

# Data Engineering

Mar 2020 Vol. 43 No. 1



IEEE Computer Society

---

## Letters

Letter from the Editor-in-Chief . . . . .	Haixun Wang	1
Letter from the Special Issue Editor . . . . .	Philippe Bonnet	2

---

## Opinions

Entities with Quantities . . . . .	Gerhard Weikum	4
------------------------------------	----------------	---

---

## Special Issue on Data Management at Exascale

Experiences in Exascale Scientific Data Management . .	Mario Lassnig, Martin Barisits and Dimitrios Christidis	9
Advancing RPC for Data Services at Exascale . . . . .	Jerome Soumagne, Philip Carns and Robert B. Ross	23
Extending the Publish/Subscribe Abstraction for High-Performance I/O and Data Management at Extreme Scale . . . . .	Jeremy Logan, Mark Ainsworth, Chuck Atkins, Jieyang Chen, Jong Choi, Junmin Gu, James Kress, Greg Eisenhauer, Berk Geveci, William Godoy, Mark Kim, Tahsin Kurc, Qing Liu, Kshitij Mehta, George Ostrouchov, Norbert Podhorzski, David Pugmire, Eric Suchyta, Nicolas Thompson, Ozan Tugluk, Lipeng Wan, Ruonan Wang, Ben Whitney, Matthew Wolf, Kesheng Wu and Scott Klasky	35
SUQ <sup>2</sup> : Uncertainty Quantification Queries over Large Spatio-temporal Simulations . . . . .	Noel Moreno Lemus, Fabio Porto, Yania M. Souto, Rafael S. Pereira, Ji Liu, Esther Pacciti, and Patrick Valduries	47
Networking and Storage: The Next Computing Elements in Exascale Systems? . . . . .	Alberto Lerner, Rana Hussein, Andre Ryser, Sangjin Lee, Philippe Cudre-Mauroux	60

---

## Conference and Journal Notices

TCDE Membership Form . . . . .		72
--------------------------------	--	----

## Editorial Board

### Editor-in-Chief

Haixun Wang  
WeWork Corporation  
115 W. 18th St.  
New York, NY 10011, USA  
haixun.wang@wework.com

### Associate Editors

Philippe Bonnet  
Department of Computer Science  
IT University of Copenhagen  
2300 Copenhagen, Denmark  
  
Joseph Gonzalez  
EECS at UC Berkeley  
773 Soda Hall, MC-1776  
Berkeley, CA 94720-1776  
  
Guoliang Li  
Department of Computer Science  
Tsinghua University  
Beijing, China  
  
Alexandra Meliou  
College of Information & Computer Sciences  
University of Massachusetts  
Amherst, MA 01003

### Distribution

Brookes Little  
IEEE Computer Society  
10662 Los Vaqueros Circle  
Los Alamitos, CA 90720  
eblittle@computer.org

### The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is <http://tab.computer.org/tcde/index.html>.

### The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at [http://tab.computer.org/tcde/bull\\_about.html](http://tab.computer.org/tcde/bull_about.html).

## TCDE Executive Committee

### Chair

Erich J. Neuhold  
University of Vienna

### Executive Vice-Chair

Karl Aberer  
EPFL

### Executive Vice-Chair

Thomas Risse  
Goethe University Frankfurt

### Vice Chair

Malu Castellanos  
Teradata Aster

### Vice Chair

Xiaofang Zhou  
The University of Queensland

### Editor-in-Chief of Data Engineering Bulletin

Haixun Wang  
WeWork Corporation

### Awards Program Coordinator

Amr El Abbadi  
University of California, Santa Barbara

### Chair Awards Committee

Johannes Gehrke  
Microsoft Research

### Membership Promotion

Guoliang Li  
Tsinghua University

### TCDE Archives

Wookey Lee  
INHA University

### Advisor

Masaru Kitsuregawa  
The University of Tokyo

### Advisor

Kyu-Young Whang  
KAIST

### SIGMOD and VLDB Endowment Liaison

Ihab Ilyas  
University of Waterloo

## Letter from the Editor-in-Chief

How to efficiently and effectively manage large-scale data is a critical challenge in data management, scientific computing, machine learning, and many other fields. In this issue, we look into this problem from two angles.

Gerhard Weikum’s opinion piece titled “Entities with Quantities” highlights development along the direction of querying the Web as a database. We have come a long way in keyword based Web search: Today, all major search engines support entity based question/answering to certain extent (e.g., returning “Eiffel Tower” for query “the highest building in Paris”). Weikum is taking one important step towards the goal of querying the Web as a database. In the article, he discusses what it takes to find all entities that satisfy a quantity-based search condition, for example, “buildings taller than 500m” or “runners completing a marathon under 2:10h.” It is clear that this requires much advanced data preprocessing (e.g., information extraction, entity linking, etc.), but more importantly, it requires that at least part of the data on the entire Web needs to be organized as a database.

Philippe Bonnet put together the current issue consisting of 5 papers from leading researchers in the high performance computing and data management communities on the topic of data management at Exascale. Advances in exascale computing on petascale supercomputers are pushing the frontier of scientific computing that requires complex simulation, benefiting applications ranging from astrophysical discovery to drug design. But with increasing amounts of data, the gap between computation and I/O has grown significantly wider, which makes data management a big challenge. This timely issue answers many questions in this domain.

Haixun Wang  
WeWork Corporation

## Letter from the Special Issue Editor

Scientific computing used to be based on numerical simulations run on mid-range warehouse scale computers. This is no longer the case, due to the combination of strong application pull and technology push.

In order to get realistic models of a phenomenon in natural or engineered systems, scientists must analyze unprecedented volumes of data generated by new generations of instruments and experiments. In addition, they must run simulations at higher spatial resolutions, for longer simulation times and with higher dimension models, possibly combining multiple physical models of a phenomenon, or study multiple simultaneous phenomena. These new computational challenges stemming from scientific applications have triggered a convergence of traditional numerical simulation with machine learning and high-performance data analytics. Put differently, data science and eScience are merging.

The technology push is due to the planned transition to Exascale systems. Strictly defined, Exascale computers are capable of  $10^{18}$  floating points operations per second (flops). More interestingly, they are three orders of magnitude faster than the High-Performance Computers deployed a decade ago. The first Exascale systems are expected in the coming year. In the US, three systems are being deployed: Aurora at Argonne National Lab, Frontier at Oak Ridge National Lab and El Capitan at Lawrence Livermore Lab. In China three existing pre-exascale systems are being extended: Sunway at the National Research Center of Parallel Computer Engineering and Technology (NRCPC in Wuxi, Jiangsu), Sugon (installed at the Shanghai Supercomputer Center) and Tianhe at the National Center of Defense Technology (NUDT in Changsha, Hunan). In Japan, Riken and Fujitsu have designed the Fugaku Exascale computer, which has been announced for 2021, 2022. It will be hosted at the RIKEN Center for Computational Science in Kobe. In Europe, three pre-exascale computers are under construction: Mare Nostrum 5 at the Barcelona Supercomputing Center, Leonardo at Bologna's CINECA and LUMI at the CSC Data Center in Kaajani, Finland.

In 2008, Kogge et al. surveyed the technology challenges in achieving Exascale systems. The main roadblock they identified was *transporting data from one site to another: on the same chip, between closely coupled chips in a common package, or between different racks on opposite sides of a large machine room*. Put differently, minimizing data movement is the key challenge on Exascale systems. This is a challenge in terms of architecture, but it is also a challenge for data management.

In this issue, leading researchers from the HPC and database communities present their work on data management at Exascale. The papers will give readers an insight in the nature of the application pull and technology push sketched above. They contain the lessons learnt at the forefront of scientific data management. They are very interesting points of departure for future work.

Mario Lassnig from CERN and his co-authors review their experience with the Rucio system, developed at CERN, to handle data in the ATLAS experiment. They detail the challenges they faced and how Rucio addresses them. They report on recent efforts to adapt Rucio in the context of other large-scale scientific projects.

Jerome Soumagne from HDF Group and his co-authors tackle the issue of performance and resilience for data services at Exascale. They propose Remote Procedure Call as a building block for such data services. The paper describes the design of Mercury, a new form of Remote Procedure Call adapted to large data transfers on low-latency network fabrics.

Jeremy Logan from Oak Ridge National Lab and his co-authors focus on ADIOS, the Adaptable I/O System, that provides a publish/subscribe abstraction for high-performance data services. The paper describes its design and its use in the context of near Exascale use cases. Based on lessons learnt and examples from a range of different projects, the authors discuss challenges and opportunities for future work on data management at Exascale.

Noel Moreno Lemus from LNCC (National Lab for Scientific Computing in Rio de Janeiro, Brazil) and his co-authors tackle the issue of large-scale spatio-temporal simulations. More specifically, they focus on answering uncertainty quantification queries over such simulation results. This is a great example of the convergence of numerical simulation and query processing.

Finally, Alberto Lerner from the eXascale Infolab at U.Fribourg and his co-authors present their vision for in-network computing and near-storage processing. These techniques are crucial for bringing computation closer to data and thus tackle the issue of data movement. Their vision is based on a thorough analysis of the current generation of platforms and of the computation tasks that could be brought closer to data at rest or in movement.

These papers addresses several aspects of the state of the art in data management at Exascale and they outline a range of challenges. There are many opportunities for the database community to engage with the high-performance computing community to tackle these challenges. As Jim Gray once wrote: *The next decade will be exciting!*

Working on this issue has been a privilege. I would like to thank the authors, and specially the five contact authors for their diligence and express my admiration for the quality of their work. I would also like to thank Haixun Wang for his kind and efficient management and Pinar Tözün for her feedback. Finally, I would like to thank David Lomet for the opportunity to act as editor of this special issue.

Philippe Bonnet  
IT University of Copenhagen

## Entities with Quantities

Gerhard Weikum  
Max Planck Institute for Informatics

### The Web as a Database

Unstructured content, like text in web pages, and semistructured content, like HTML tables in web pages, has much weaker search functionality compared to structured databases. For example, joins between text documents via co-occurring entity mentions or attribute values are infeasible, unless major efforts are taken to create mark-up for a structured view. As for web tables, filters on entity mentions allow users to look up data, but results are noisy and error-prone because of ad-hoc choices for names of entities and value encodings, with huge heterogeneity across tables and often even within a table.

In the last few years, large knowledge graphs (KG), machine learning (ML) techniques and advances in entity linking algorithms [12, 17] have enabled search engines to overcome these issues, to a large degree [13]. By detecting entity mentions in web content and normalizing them onto KG entries, it has become possible to answer entity-centric queries about people, places and products almost as precisely and concisely as a database query. The following examples work with all major search engines and return crisp entity-level answers:

Query	Results(s)
Height of the Eiffel Tower	324 meters
highest building in Paris	Eiffel Tower
CEO of Amazon	Jeff Bezos
Bezos worth	108.9 Billion USD
CEOs of IT companies	Jeff Bezos, Sundar Pichai, Ginny Rometti, Zhang Yong, ...

Search engines leverage look-ups in back-end knowledge graphs, and run entity detection on both user inputs and page contents to provide these answers. It seems that entity-centric search on the web has become as easy and as effective as querying a structured and curated database!

The same methodologies, particularly, entity linking, are also key to joining data for the same entity across web tables and within heterogeneous data lakes [8, 19].

### Quantity Queries

On the disillusioning side, there is an interesting and challenging type of queries that is underexplored and hardly supported: searching with *quantities*: quantitative measures of entities that capture financial, physical, technological or environmental properties. Examples are: a celebrity’s personal wealth, a company’s quarterly revenue, a car’s energy consumption, a material’s thermal conductivity, or the usual and maximal dosage of a medical drug. Quantities can be represented as  $\langle \text{measure}, \text{value}, \text{unit} \rangle$  triples, such as  $\langle \text{height}, 8848, \text{meter} \rangle$ . The units can be simple, such as meters, light-years, US dollars, Euros etc., with well-defined conversion rules between different units for the same measure. But they can also be quite sophisticated such as  $kWh/100km$  for a car’s energy consumption or  $W/(mK)$  for the thermal conductivity of materials, with more complex conversion rules, e.g., between  $kWh/100km$  and MPG (miles per gallon) for electric, hybrid and fuel-based cars. Conversions often require context information, such as date for currency conversions, or location for car properties

---

*Copyright 2020 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*

**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

---

(incl. carbon footprint). The International System of Units (SI) is a rich reference for measures and conversions ([https://en.wikipedia.org/wiki/International\\_System\\_of\\_Units](https://en.wikipedia.org/wiki/International_System_of_Units)).

Search engines perform well on looking up quantities for given entities, such as retrieving the height of the Eiffel Tower. In this regard, quantity properties are not different from other properties such as city or architect. The pain point, however, is *finding* all entities (of a certain type) that satisfy a *search condition for a quantity of interest*, for example, buildings taller than 500m or runners completing a marathon under 2:10h. With few exceptions where explicit lists are available, search engines fall back to returning page links only. The following examples illustrate this disappointing behavior.

Query	Results(s)
people worth 50 Billion USD	link to “List of Americans by net worth - Wikipedia”
... more than 50 Billion USD	links to pages such as “Meet the world’s 50 richest billionaires in 2019”
... between 10 and 50 Billion Euros	links to pages such as “Inequality and Wealth Distribution in Germany”

Search engines do not understand numbers and units (with a few exceptions regarding dates and money, sometimes). For example, “15 kW” and “15.000 W” are two different strings. Units like “l/100km”, “MPG”, “MPGe” and “kWh/100km” are also just strings, and the systems are ignorant about unit conversions.

These queries would be trivial to handle if all data resided in a single database with well-designed schema, standardized value encodings, and high-quality curation. However, these databases do rarely exist, or are outdated or incomplete. One would hope that this is where encyclopedic knowledge graphs kick in, such as DBpedia, Wikidata or Yago. However, quantitative properties are very sparse in these KGs, and often represented just as strings, e.g., “250 mi  $\pm$  10” for the range of a car model. Only Wikidata contains triples for the range of cars, but only for 4 models (as of Dec. 2019). As for other measures, like engine power, energy efficiency, carbon footprint etc., none of these KGs has any data. Only the Web as a whole contains the wealth of information that is needed to compute accurate and complete answers to many kinds of quantity queries.

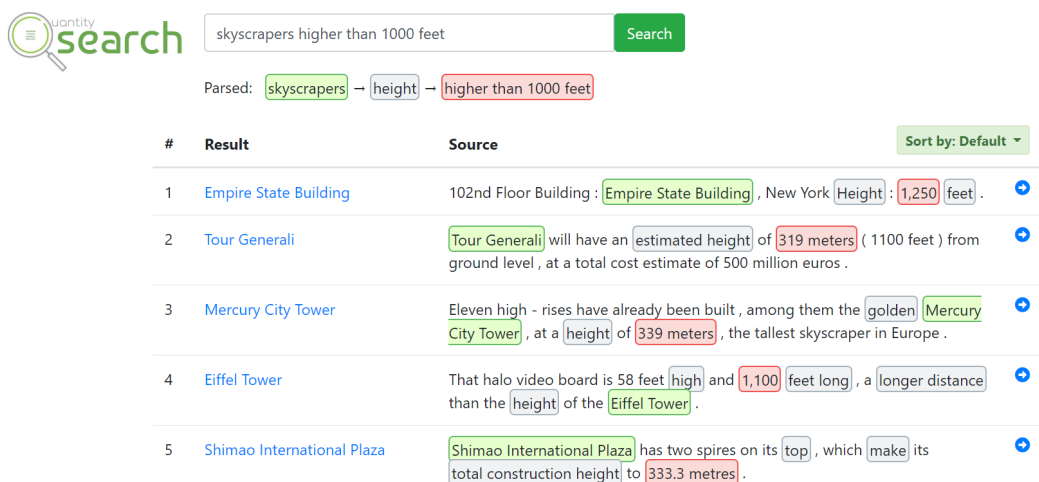
## Initial Proof of Concept

Supporting quantity queries is easy over a single well-curated database. It is challenging over web page contents, web table collections or data lakes. In the latter cases, we need to overcome the obstacles of highly heterogeneous schemas, diverse and noisy value encodings, and widely varying degrees of coverage [10].

As an initial effort, we devised methods for a limited class of quantity queries over text document collections such as Wikipedia articles or news corpora. This work has led to an early prototype system, called *Qsearch* [4, 5]. The system consists of a *data preparation* stage with quantity extraction and indexing, and a *query processing* stage with matching and ranking. A Qsearch demonstrator is accessible at <https://qsearch.mpi-inf.mpg.de>. Figure 1 shows the top-ranked answers for an example query about buildings higher than 1000 ft.

**Information Extraction:** Qsearch uses machine learning for sequence tagging. It trains an LSTM neural network with distant supervision, and applies the learned model to tag each word in a sentence, identifying three components: i) an *entity* of interest, ii) a *quantity* that refers to this entity, and iii) *context cues* that capture what exactly the quantity denotes. For example, from the sentence “The hybrid Prius is sold in Germany for less than 30 thousand, and has a battery only range of 60 km.”, Qsearch extracts two assertions: first, related to price: i) Toyota Prius as key entity, ii) 30,000 Euros (upper bound) as quantity, iii) “sold in Germany” as cue words, and second, related to range: i) Toyota Prius as entity, ii) 60 km as quantity, iii) “battery only range” as cue words.

**Query Analysis:** At query time, Qsearch analyzes telegraphic queries or full questions and decomposes them into three components: *semantic target type* (e.g., buildings or hybrid cars etc.), *quantity condition* of the form  $\langle comparison, value, unit \rangle$  (with comparisons like  $\leq$ ,  $\geq$ , between, etc.), *context cues* that candidate results should match (e.g., “electric range in city traffic” for a query about hybrid cars).



Qsearch

skyscrapers higher than 1000 feet Search

Parsed: skyscrapers → height → higher than 1000 feet

Sort by: Default

#	Result	Source
1	Empire State Building	102nd Floor Building : Empire State Building , New York Height : 1,250 feet .
2	Tour Generali	Tour Generali will have an estimated height of 319 meters ( 1100 feet ) from ground level , at a total cost estimate of 500 million euros .
3	Mercury City Tower	Eleven high - rises have already been built , among them the golden Mercury City Tower , at a height of 339 meters , the tallest skyscraper in Europe .
4	Eiffel Tower	That halo video board is 58 feet high and 1,100 feet long , a longer distance than the height of the Eiffel Tower .
5	Shimao International Plaza	Shimao International Plaza has two spires on its top , which make its total construction height to 333.3 metres .

Figure 1: Screenshot of Qsearch answers to query about buildings higher than 1000 ft

**Matching and Ranking:** Query processing aims to match all components of an assertion against the components of the query: the entity must be of the right type, the quantity condition must be satisfied, and the context cues must match as well as possible (leveraging word embeddings, e.g., to capture the relatedness of “battery only” and “electric range”). As the latter comes with uncertainty, Qsearch employs language-model-style ranking to compute the best answers.

## Challenges and Opportunities

**Quantity Filters:** Even basic filters over quantities still pose enormous challenges. The extraction from text often faces complicated and misleading inputs, such as “The battery of the hybrid Toyota Prius lasts well over 100,000 miles” as a spurious candidate for the electric range of this car. For more sophisticated measures such as the CO2 footprint of cars, it is crucial to consider elaborate context like the source of energy for electric cars, the driving situations (city vs. highway, summer vs. winter), and more. This will rarely be fully captured in a single sentence; so we need information extraction that combines and reconciles cues from entire paragraphs or even multiple documents. State-of-the-art work on quantity detection and extraction [1, 4, 6, 14, 15, 16] has disregarded such advanced settings so far.

Major sources for quantity information are also web tables and Open Data accessible on the Internet (e.g., [www.data.gov](http://www.data.gov), [data.gov.uk](http://data.gov.uk), [data.europa.eu](http://data.europa.eu), etc.). Tapping into this kind of (semi-) structured web data comes with huge challenges. Despite prior work on annotating cells in ad-hoc tables with entities and types [2, 3, 9, 18], understanding quantities and their relations to entities in this kind of online contents is way underexplored. The most notable prior endeavor is the work of Sarawagi et al. [16], which focused on a limited range of query types over web tables. Note that besides HTML tables in web pages, this direction should also consider spreadsheet data in enterprises as well as highly heterogeneous data lakes like Open Data. In addition, combining tables with cues from their surrounding text (in web pages or enterprise documents) could potentially be a powerful asset [7].

**Quantity Joins:** A next step would be tackling comparisons between quantities, either for the same entity or for different entities. For example, we could ask for 100m sprinters whose best time in Olympics finals is their personal record, or for such athletes whose time in the Olympics was worse than their personal best of the same year. These comparisons entail joins over the quantity values, in the second case even a non-equi join. The example may appear very special (of interest only to sports aficionados), but similarly structured queries appear in other domains as well; examples are comparing medical drugs and their usage (e.g., anti-coagulants for which



the standard dosage is higher in the US than in the EU), or environmental properties of fuel-based, hybrid and electric cars in different geo-regions.

These queries are easy to express in SQL if the data resides in a single high-quality database. The challenge lies in applying them to extractions from text and tables (incl. scientific literature such as PubMed, ClinicalTrials, etc.) and to ad-hoc collections of many databases.

**Quantity Aggregation:** Given the inherent noise in extractions and the incompleteness of tables, it is often necessary to aggregate quantity information from multiple sources. For example, we may have to compute unions of entity sets as a basis for grouping and aggregate comparisons, or we have to combine many extractions to approximate proper values.

Such aggregations can be amazingly difficult even for seemingly simple cases. Already basic counting can be painful and challenging [11]. Consider the example of computing the total number of World Championship medals that Usain Bolt has won in his career (answer is 14). We may obtain cues from text and tables such as: he has won 100m three times, he won 11 gold medals between 2009 and 2017, he helped the Jamaican team to win the 4x100m relay race four times, 200m@2007 2nd place: Usain Bolt, 100m@2017 3rd place: Usain Bolt, etc. Can we infer the total, or at least lower and upper bounds? For prominent cases like Usain Bolt, this is not really necessary, as there are high-quality tables and lists already and we can look up the total rather than computing it. However, for less popular entities, the accessible information is often partial and spread across many sources. One difficulty is to avoid over-counting by disregarding that the 11 gold medals already include the four medals for the relay race. If we first specified a rule system, about sports medals, we could use reasoning to infer totals, but we want a solution that works out-of-the-box for all possible domains. Can we use machine learning to predict bounds for totals and other aggregates, with as little supervision as possible?

Obviously, the task gets only harder once we tackle quantities with units for realistic use cases. For example, what are the average blood lab values for diabetes patients of certain age groups in different parts of the world (as reported in clinical studies at PubMed, and other online sources)?

## An Analyst's Dream

Quantity queries are often part of high-stakes information needs by advanced users, such as analysts, journalists, scientists and other knowledge workers. Ideally, an analyst would run her entire data analysis over web contents as easily as posing a keyword query or single-sentence question:

- Which runners have completed 10 marathons under 2 hours 10 minutes?
- Which is the most energy-efficient hybrid car model?
- How does the carbon footprint of Japanese cars compare to US-made cars when driven in the Bay Area?
- Which vaccinations have more than 80% coverage in the 20 population-wise largest countries?

The envisioned solution should support search engines over textual contents, web tables as well as heterogeneous data lakes. The key issues of extracting, normalizing, matching, ranking and aggregating quantities are the same regardless of whether we tap into textual contents or structured but fairly raw data.

More than 30 years ago, Bill Gates promised that “all information is at your fingertips” and Larry Page foresaw that “the ultimate search engine would understand exactly what you mean and give back exactly what you want”. We have gone a long way towards these goals, but there are still many obstacles. This opinion paper is a call to overcome these issues for an interesting and valuable slice of information needs.

## References

- [1] Omar Alonso, Thibault Sellam: Quantitative Information Extraction From Social Data. SIGIR 2018

- [2] Chandra Sekhar Bhagavatula, Thanapon Noraset, Doug Downey: TabEL: Entity Linking in Web Tables. ISWC 2015
- [3] Michael J. Cafarella, Alon Y. Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, Eugene Wu: Ten Years of WebTables. PVLDB 11(12), 2018
- [4] Vinh Thinh Ho, Yusra Ibrahim, Koninika Pal, Klaus Berberich, Gerhard Weikum: Qsearch: Answering Quantity Queries from Text. ISWC 2019
- [5] Vinh Thinh Ho, Koninika Pal, Niko Kleer, Klaus Berberich, Gerhard Weikum: Entities with Quantities: Extraction, Search, and Ranking. Demo Paper, WSDM 2020
- [6] Yusra Ibrahim, Mirek Riedewald, Gerhard Weikum: Making Sense of Entities and Quantities in Web Tables. CIKM 2016
- [7] Yusra Ibrahim, Mirek Riedewald, Gerhard Weikum, Demetrios Zeinalipour-Yazti: Bridging Quantities in Tables and Text. ICDE 2019
- [8] Oliver Lehmberg, Christian Bizer: Stitching Web Tables for Improving Matching Quality. PVLDB 10(11), 2017
- [9] Girija Limaye, Sunita Sarawagi, Soumen Chakrabarti: Annotating and Searching Web Tables Using Entities, Types and Relationships. PVLDB 3(1), 2010
- [10] Renee J. Miller, Fatemeh Nargesian, Erkang Zhu, Christina Christodoulakis, Ken Q. Pu, Periklis Andritsos: Making Open Data Transparent: Data Discovery on Open Data. IEEE Data Eng. Bull. 41(2), 2018
- [11] Paramita Mirza, Simon Razniewski, Fariz Darari, Gerhard Weikum: Enriching Knowledge Bases with Counting Quantifiers. ISWC 2018
- [12] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, Vijay Raghavendra: Deep Learning for Entity Matching: A Design Space Exploration. SIGMOD Conference 2018
- [13] Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, Jamie Taylor: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM 62(8), 2019
- [14] Subhro Roy, Tim Vieira, Dan Roth: Reasoning about Quantities in Natural Language. TACL 3, 2015
- [15] Swarnadeep Saha, Harinder Pal, Mausam: Bootstrapping for Numerical Open IE. ACL 2017
- [16] Sunita Sarawagi, Soumen Chakrabarti: Open-domain quantity queries on web tables: annotation, response, and consensus models. KDD 2014
- [17] Wei Shen, Jianyong Wang, Jiawei Han: Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. IEEE Trans. Knowl. Data Eng. 27(2), 2015
- [18] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, Chung Wu: Recovering Semantics of Tables on the Web. PVLDB 4(9), 2011
- [19] Erkang Zhu, Dong Deng, Fatemeh Nargesian, Renée J. Miller: JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. SIGMOD 2019

# Experiences in exascale scientific data management

Mario Lassnig\*  
CERN

Martin Barisits  
CERN

Dimitrios Christidis  
UT Arlington

## Abstract

*Scientific data management has become an increasingly difficult challenge. Modern experiments and instruments are generating unprecedented volumes of data and their accompanying dataflows are becoming more complex. Straightforward approaches are no longer applicable at the required scales and there are few reports on long-term operational experiences. This article reports on our experiences in this field: we illustrate challenges in operating the exascale dataflows of the high energy physics experiment ATLAS, we detail the concepts and architecture of the data management system Rucio that was purposely built to take up these challenges, we describe how other experiments evaluated, modified, and adopted the Rucio system for their own needs, and we show how Rucio will evolve to cope with the ever increasing needs of the community.*

## 1 Introduction

Many large scale scientific experiments are reaching a breaking point where the growth rate of the collected data greatly exceeds the growth rate of the infrastructure behind them. In the next few years, large instruments similar in scale to the Large Hadron Collider (LHC) [1] and its High-Luminosity upgrade HL-LHC [2] are coming online, such as the Deep Underground Neutrino Experiment (DUNE) [3], or the Square Kilometre Array (SKA) radio observatory [4]. Throughout their lifetimes, these collaborations anticipate massive increases both in the number of data objects they need to handle as well as the total volume of data they need to store. Additionally, increasingly complex computational workflows result in similarly complex dataflows to support them, which can rapidly lead to science-inhibiting complications. Examples include congestion on networks, disorderly space allocations on storage, or erratic transfer schedules. At the same time, there are many smaller communities and experiments who do not want to lose efficient access to the same shared storage and network resources but do

---

Copyright 2020 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

---

\*E-Mail: Mario.Lassnig@cern.ch

not possess a similar level of effort to ensure their sustainability. This brings a variety of challenges to the field of data management as a whole to ensure fair use of all available resources, leading to the need to orchestrate and synchronize dataflows across multiple experiments with potentially competing characteristics.

In this article we address four topics relevant to these challenges: in Section 2 we describe first-hand experiences developing and operating an exascale data management system for high energy physics, in Section 3 we describe the data management system that was used to tackle these challenges, in Section 4 we discuss how a diverse set of scientific communities evaluated, modified, and adopted the system for their own needs, and in Section 5 we propose possibilities how to integrate and interconnect widely distributed data management solutions as a cooperating ensemble. We conclude in Section 6 with a summary and an outlook on future work.

## 2 The data challenge in High-Energy Physics

The LHC at CERN hosts four major experiments, ATLAS [5], CMS [6], ALICE [7], and LHCb [8]. Both ATLAS and CMS are general-purpose particle physics experiments and are designed to exploit the full discovery potential and the huge range of physics opportunities that the LHC provides, whereas ALICE and LHCb focus on detailed precision studies in their respective fields. All experiments are run by large international collaborations. The experiments track and identify particles to investigate a wide range of physics topics, from the study of the Higgs boson [9] to the search for supersymmetry [10], quark-gluon plasma [11], b-physics [12], extra dimensions [13], or potential particles that make up dark matter [14].

For the remainder of the article, we will focus on the data management aspects of the ATLAS experiment, as it presents the most diverse dataflow requirements across the LHC experiments. As such, many of the experiences presented are similarly applicable to the other experiments, scaled to their respective experiment sizes. At its current scale, ATLAS is currently managing more than 1 billion files in active use comprising almost 500 Petabytes of data. For scientific integrity and reproducibility, the experiment also needs to keep track of data that has been deleted, which amounts to an additional 1.5 billion historical files. The interaction rate of operations with the data management system are typically beyond 200 Hz and reach up to 500 Hz. This includes diverse operations such as registering new files, searching for data, scheduling old files for deletions, or removing unwanted datasets. The experiment utilizes 120 data centers globally, including 5 supercomputing centers (HPCs), and connects to scientific and commercial cloud storage. There are more than 1000 active users, who in turn require data transfer and deletion rates at 500 Petabytes/year, plus an additional 2.5 Exabytes/year of data access for their ongoing analyses.

Figure 1 shows the cumulative data volume from ATLAS starting from 2008, when the distributed computing infrastructure was commissioned. The data growth in high energy physics is not exponential, rather there are linear growth intervals at varying intensities. It can be seen that the data growth of LHC Run 1 from April 2011 until September 2012 is distinctively higher than the following LHC shutdown period until April 2015, during which only simulation data was produced. The first major deletion campaign to free up space on the available storage marks the beginning of the four-year-long LHC Run 2. The vastly increased data rate after 2015 reflects the higher intensity of the LHC. Notably, the growth rate of the simulated data also increased, which required a second major deletion campaign at the beginning of the current LHC shutdown period. We anticipate a similar deletion campaign in late 2020 before the start of LHC Run 3.

Figure 2 shows the weekly ATLAS data transfers and downloads. The transfers between data centers are enacted whenever required by the computational workflow, and the downloads subsequently occur from the storage to the node which does the actual computation. Users downloading data directly to their workstations accounts for roughly 12 percent of the overall download volume. The colors indicate the 12 geographical regions of the experiment, out of which 4 larger regions are responsible for half of the total capacity. The global weekly transfer volume is typically more than 50 Petabytes, but can burst significantly above 70 Petabytes. There is a prominent dip at the end of the year which corresponds to the two-week closure of the CERN facilities during

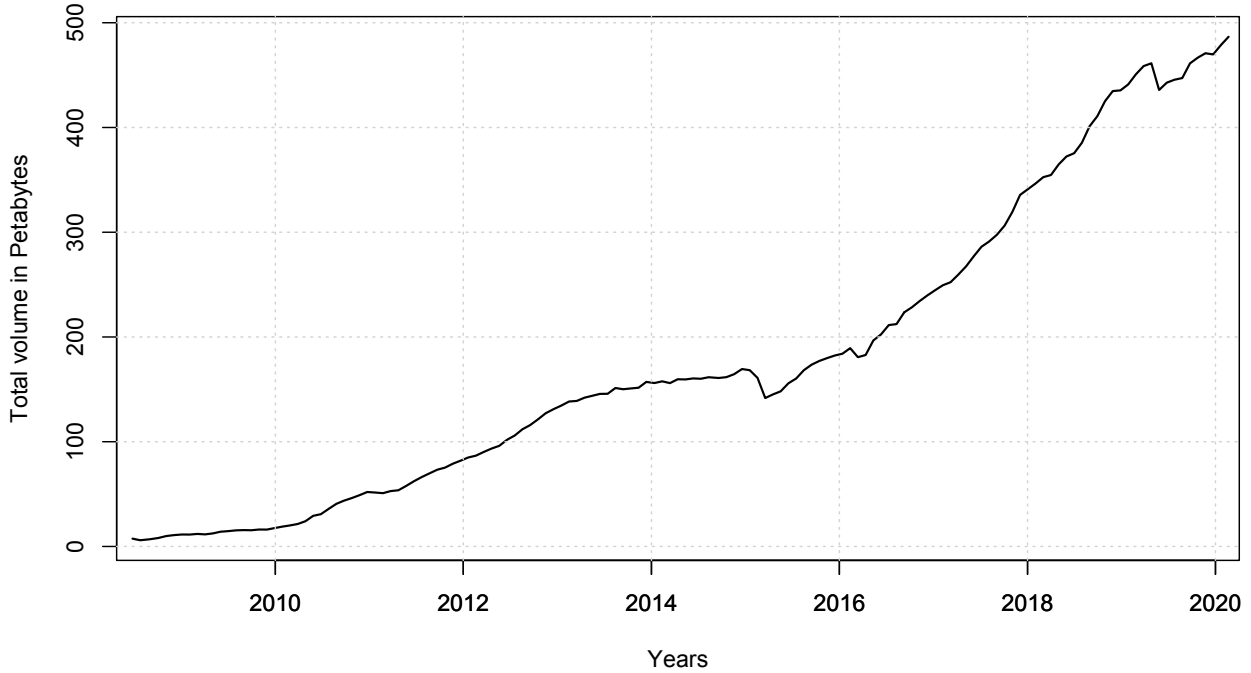


Figure 1: The cumulative ATLAS data volume approaches 500 Petabytes in early 2020. Growth has been linear with respect to the scale of the experiment, with considerable data deletion before longer observation periods.

the festive season. The number of files transferred per week is typically beyond 50 million. The number of file deletions is equivalent to roughly 10 Petabytes per week, arising from the needs of the embarrassingly parallel computational workflow, which results in many temporary intermediate files. The number of files deleted per week is typically more than 20 million. In terms of transfer failures, the typical rate is more than 4 million failures per week, mostly due to faulty hardware such as broken disks or electrical problems. Deletion failures are typically rarer, below 1 million per week, mostly due to the implementation of the WebDAV-based deletion protocol in the storage systems. Scheduled deletion of files which were already removed from storage do not count as failures. Major deletion failures occur rarely, on the order of once per quarter, which typically points to hardware failures at a data center with massive data loss. One of the major successes of the ATLAS data management system is that it can handle a large variety of these failures transparently, that is recover and restore data from alternative, pristine sources to ensure global data safety.

We now highlight three examples from data management operations in ATLAS, which help to achieve the experiment’s needs: the data lifetime model [15], sliding window processing [16], and data recovery [17]. As the first example, the lifetime model works as follows. If left unchecked, the creation of new data within the experiment would quickly exhaust all available storage space. Members of the operations team, together with other groups from the collaboration, need to spend considerable effort to identify and delete datasets that are no longer needed, mostly via consensus across scientific groups. Multiple procedures have been developed over the years to alleviate this, however the most prominent and effective is the lifetime model. Within ATLAS, each dataset is annotated with a wide variety of metadata. Policies are defined based on these metadata that dictate when they are expected to expire. For example, raw data coming out of the ATLAS detector are kept forever, whereas data in analysis object formats are kept for two years and log files are kept for only one year. These

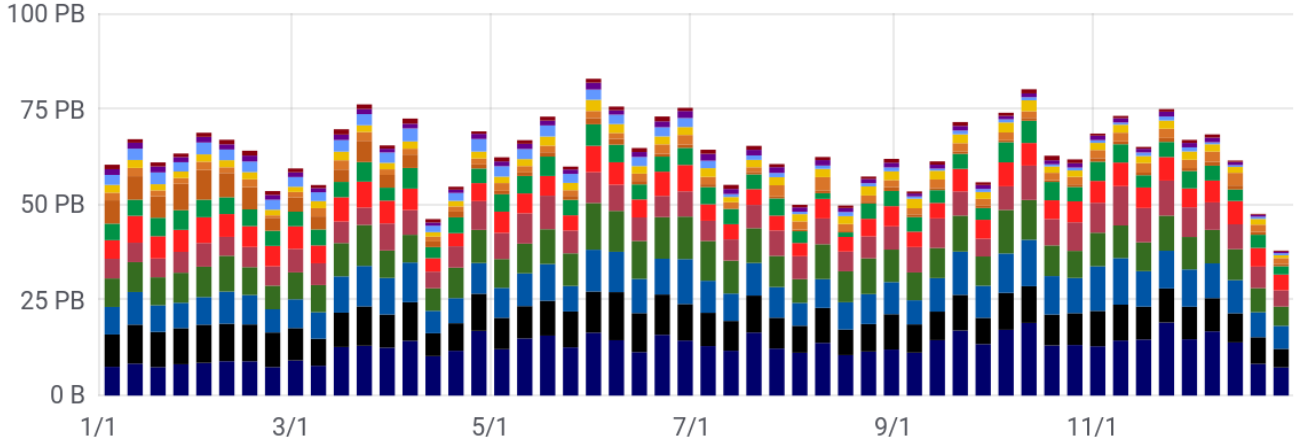


Figure 2: ATLAS data transfers and downloads are regularly above 50 Petabytes per week throughout 2019. The prominent dip during the festive season is due to CERN closure. Colors indicate geographical regions.

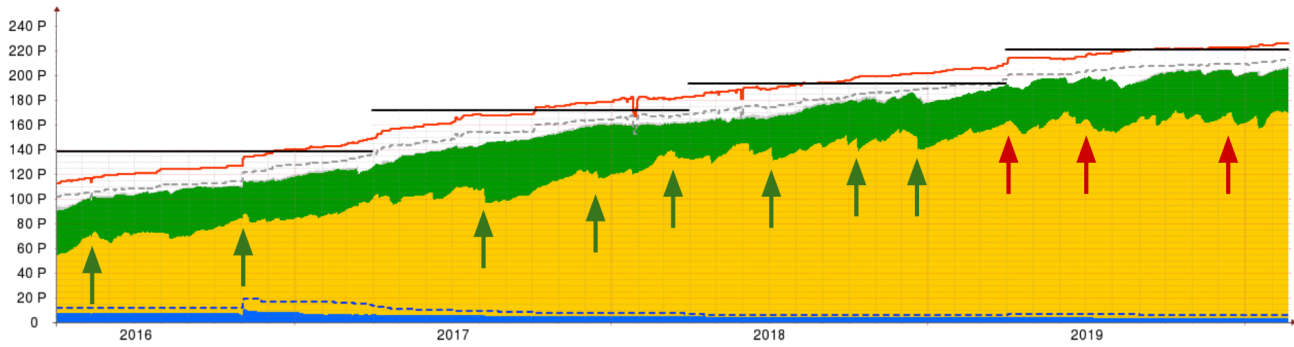


Figure 3: The green and red arrows indicate the application of the lifetime model for scheduled deletion of unused data on pledged disk resources in ATLAS. Green arrows are rule-based deletion, red arrows are file-based deletion. Black lines indicate data center pledges, the red line actual available storage. The yellow data volume indicates primary data, the green data volume is cached data.

policies allow for the lifetime of the dataset to be further extended based on the last access date, which is based on a distributed data access tracing system. This allows the deletion of data to be postponed beyond its expected lifetime as long as it is in constant use. The lifetime model itself does not however initiate the deletion of the expired datasets. Instead, a procedure is set where an announcement is made which datasets are scheduled for expiry, and users are given a period to submit exception requests. These requests are reviewed by the operations team, and the final selection is then scheduled for deletion. Applications of the lifetime model are shown in Figure 3, indicated by the green and red arrows. Typically multiple iterations are necessary per year to keep enough storage space available for distributed processing given the available storage capacity of the data center. Originally, only rules enforcing dataset distribution would be removed, that is the files themselves would remain as cached copies on storage and would be deleted only when the data center they were stored at was running out of space. In late 2018, the execution of the lifetime model was adjusted such that the rules were removed and that the files were immediately purged. The reason for this was to reduce the latency between the deletion requests and the space being made available for new data. This can be seen by the gradual decrease of the yellow primary data volume, with no correlating increase of the green cached data volume.

The second example is the sliding window process, internally called the Data Carousel. In ATLAS, typical processing workflows meant recalling all necessary data from tape onto disk, and then starting the processing.

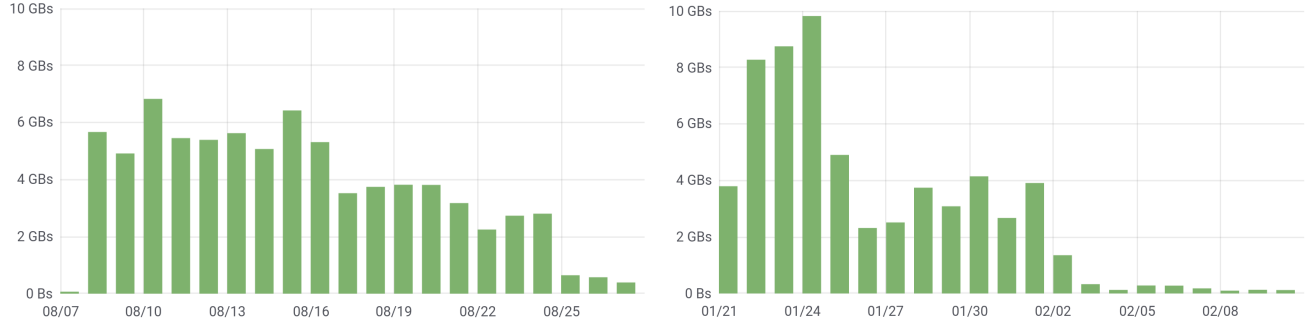


Figure 4: Data Carousel tape recall rate. On the left, measurements from the 2019 exercise with slow ramp-up and 5 GB/s throughput. On the right, the 2020 exercise with fast ramp-up and doubling of throughput.

These recalls might take from many days up to several weeks but were necessary to ensure constant high CPU utilization. With recent advances in the workflow management system and data management system, a new processing method was developed which allows the use of a sliding window process to ensure only data necessary for a particular step in the processing was recalled while still maintaining high CPU utilization. The main benefit is in the cost reduction for data centers, because instead of purchasing expensive disk-based storage a large fraction of funding can now be shifted to tape with a significantly lower cost per byte ratio. Figure 4 shows the improvements in the process from 2019 to 2020. On the left hand side, the first iteration of the process showed a significant ramp-up time of more than one day before stabilizing at 5 MBps. This functional test demonstrated that the process is feasible but required significant improvements in terms of throughput performance. During the next months, several strategies were developed, including tuning of the tape systems by the data centers, latency reduction in communication, and most importantly the exploitation of the dataset namespace to schedule the tape-writing and their recalls in groups beneficial to the computation. As seen on the right hand side, these improvements caused immediate ramp-up and a doubling of throughput capacity. At the time of writing, the process is now in use in production by ATLAS and has already been used in a significant reprocessing campaign. In this campaign, which involves the processing of 5.7 Petabytes of data, never more than 1 Petabyte is resident on disk and several hundred Terabytes are processed and removed in daily cycles.

The third example is the data recovery process. Given the number of files registered in the data management system and the number of ongoing transfers, data corruption or loss is unavoidable. The reasons can vary from corrupted network packets to faulty disk and tape drives, or even natural disasters, such as flooding of data centers. The data management system is flexible and easily allows multiple file replicas to be stored in different countries and on different storage media. For the long-term archival of raw data, experience has shown that maintaining two copies on tape storage in two different locations, one of which is at CERN, is sufficient. Each file registered in the system has a defined checksum, typically Adler-32. The reason for using Adler-32 is the algorithm's performance and cumulative property, which allows in-flight calculation. The checksum is propagated to the transfer mechanism so that if a file is corrupted, either at the source or the destination, then the transfer will immediately fail. These transfer failures can occur for a variety of different reasons, so pattern matching is applied to the error message to provide an indication where corruption occurs. Should the source file appear to be corrupted, then the replica is marked as suspicious. The data management interface collects and lists these suspicious replicas and provides a simple mechanism for operators to declare them lost. If there are other replicas of a file then a transfer will be automatically scheduled to recover it. The process is partially automated: if a file has multiple transfer failures and more than one replica, then it will be automatically declared as lost. The data management system provides configurable thresholds to fine-tune this process. The patterns themselves are also configurable. The same treatment also applies to missing files: the mechanism can trigger the files to be declared as suspicious and potentially lost. However, automated consistency checks are also conducted as a proactive measure, using periodic extracts from the data management system catalog. These are

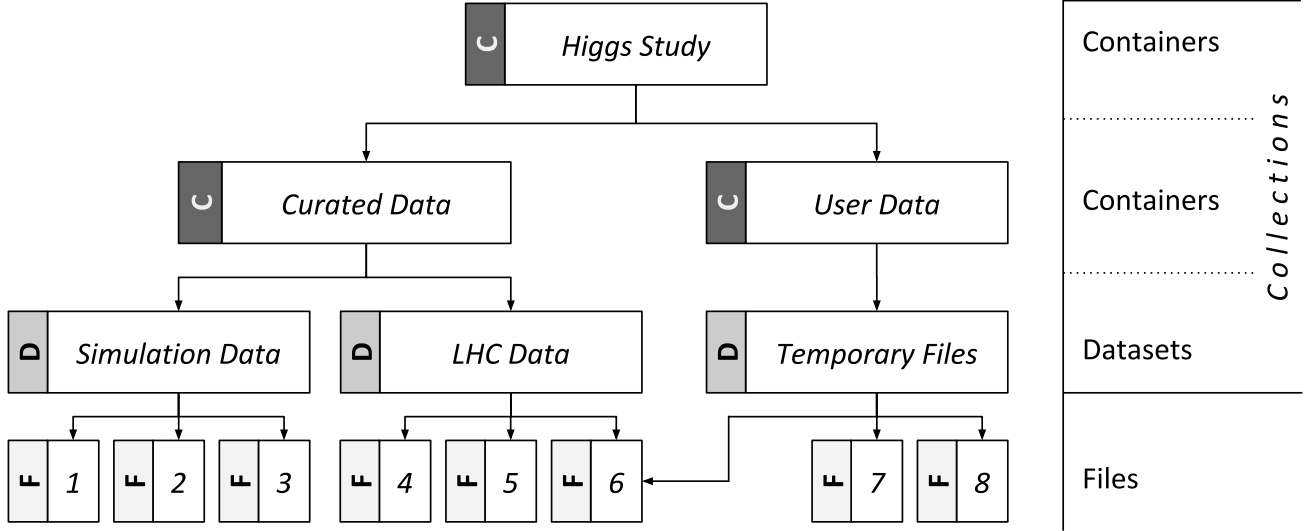


Figure 5: The namespace is organized via dynamic collections of three different types, called *Data Identifiers (DIDs)*: containers, datasets, and files. Only datasets can contain files, but files can be in multiple datasets. All operations can be enacted on all DIDs, regardless the type, and are respectively resolved.

compared against periodic extracts from the storage systems in the data centers. If a file exists in the former but is missing from the latter, then it is marked as suspicious. If a file exists in the latter but not in the former, then it is marked as dark data, which means it is occupying space without being able to be used by the experiment. The data management system ensures that such files are deleted from the data centers. This is a crucial process, which aims to prevent the accumulation of non-addressable data and in turn loss of available storage space.

### 3 The Rucio system for scientific data management

The ATLAS experiment has developed the Rucio system to handle all its distributed data needs. An extensive summary article describes Rucio in great detail [17], therefore we only give an overview here.

Rucio manages location-aware data in a heterogeneous distributed environment, including creation, location, transfer, deletion, and annotation. Declarative orchestration of dataflows with both low-level and high-level policies is the main mode of operation. The software is free and open-source, licensed under Apache v2.0, and makes use of established open-source toolchains. In terms of functionality, Rucio provides a mature and modular scientific data management federation, including seamless integration of scientific and commercial storage and their corresponding network systems. Data is stored in files, but can contain any potential payload. The storage facilities can be distributed at multiple locations belonging to different administrative domains, which makes it particularly useful for large collaborations. It was designed following more than a decade of operational experience in large-scale data management [18] and has dataflow automation as its guiding principle.

Rucio is organized in a scoped namespace of Data Identifiers (DIDs). DIDs are unique and can be files or collections. Collections can be datasets which then contain files, or containers which consist of other containers and datasets. Figure 5 shows an example namespace of several DIDs, notably the possibility to have overlapping contents across collections. Datasets and containers are logical units which usually share some scientific context, for example, files with results from a specific study, or data taken in during a specific interval. These collections also enable the user to execute certain bulk operations such as transfers or downloads in a convenient way, as Rucio correctly interprets operations on DIDs depending on their type.



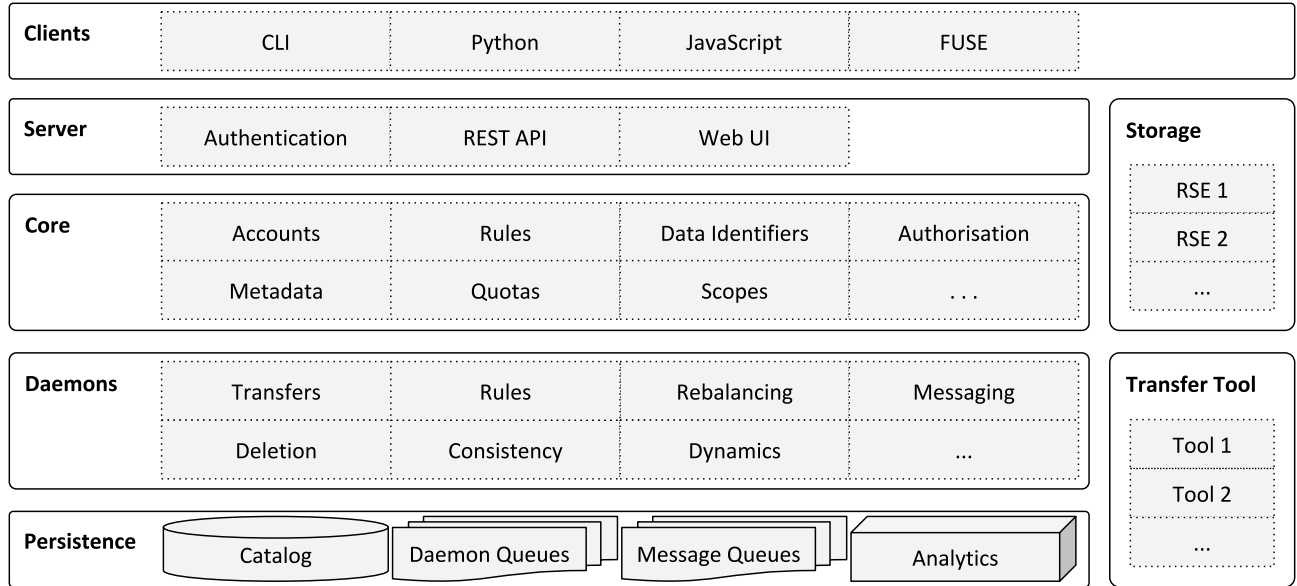


Figure 6: Rucio is loosely-coupled, with its catalog and state persisted in a transactional database. Daemons coordinate their work in shared queues, external systems are notified via message queues, and analytics storage is kept separately. The storage and transfer tools are modular and integrated vertically across the layers.

The logical representation of storage in Rucio is called a Rucio Storage Element (RSE). An RSE is a description of network-accessible storage with certain attributes, such as hostname, port, protocols and priorities. An RSE is only a logical abstraction of storage space, there is no need to run any Rucio software at a data center. Simply providing the RSE configuration to Rucio is sufficient and thus allows to federate a wide variety of heterogeneous storage systems. Interaction with the physical storage is orchestrated via the published protocols, for example gsiftp, WebDAV, root, S3, and many more. Thus RSEs can be any type of storage, typically disk or tape systems, scientific and commercial cloud providers, or even supercomputers. The physical representations of files on storage are called replicas. A replica is always associated to a specific RSE. Files can have one, or many replicas depending on the policies and access patterns of the organization. Files without replicas are marked as deleted and are kept in the historical namespace for reference. The orchestration of dataflows in Rucio is done via a descriptive concept called replication rules. These rules define policies on DIDs to ensure that a certain number of replicas are made available on RSEs matching a defined policy for a certain amount of time. Thus replication rules serve the purpose of transferring data to an RSE but also to protect the data from deletion until the lifetime of the rule expires.

The Rucio architecture, as shown in Figure 6, is based on a distributed design split into a multiple layers, each with their own set of components: the clients, server, core, daemons, and persistence layer. While details of this architecture are discussed in the summary article it is noteworthy that the architecture is fully horizontally scalable while at the same time responsive to the required work. The system can thus be used from single data center deployments up to globally distributed federations. Starting from the most basic data cataloging requirements, more advanced features can be activated selectively depending on the needs of the experiment. For example, the system supports dynamic schema-free and schema-based metadata collection and queries, data transfers between heterogeneous facilities, diverse authentication and authorization mechanisms, web user interfaces, API and CLI integrations, extensive monitoring of dataflows, and expressive high-level and low-level data policy engines. As previously mentioned, automatic data corruption identification and recovery is one of the most appreciated features of the system.

## 4 Establishing a community

As the scientific community has become more aware of the success of Rucio within ATLAS, several experiments asked if it would be possible to evaluate and potentially adopt the system for their needs. This section distills the reports from a variety of sciences from the corresponding article [19].

The first scientific experiment adopting Rucio as a data management system was the AMS-02 [20] collaboration. AMS is a particle detector attached to the International Space Station measuring antimatter in cosmic rays. There are many scientific goals of AMS, but one of the major objectives is the search for evidence of dark matter. The Rucio system was installed and managed by the Taiwanese Academy of Science (ASGC). This was done in collaboration and support of the core Rucio team at CERN. ASGC team members spent many weeks at CERN to learn from the direct operational experience of Rucio at CERN and to work together with the developers to adapt the system to scientific usecases beyond high energy physics. During this collaboration multiple features were added to the Rucio code base and the ASGC team developed a powerful web interface extension serving their local community. The Rucio system at ASGC is now in use not only for AMS, but a variety of smaller experiments hosted at their institute.

The XENON dark matter experiment [21] is operated in the underground research facility Laboratory National del Gran Sasso (LNGS) in Italy. It is aiming to directly detect weakly interacting massive particles (WIMPs). The first stage are raw data, which are distributed with Rucio among grid computing facilities within the European Grid Infrastructure (EGI) and the Open Science Grid (OSG) in the US, including SDSC's Comet Supercomputer and the HPC campus resources. Second and third stage are processed data which are kept at the Research Computing Center (RCC) in Chicago, which is also its main data analysis center. XENON1T has taken 800+ TB of raw data and ran multiple re-processing campaigns for improving data quality in ongoing data analysis tasks [22]. The upcoming XENONnT upgrade will take 1 Petabyte per year. Processing and Monte Carlo simulation campaigns are planned at the major infrastructures of EGI and OSG. A newly developed tool integrates Rucio in the XENONnT data flow and data product locations are registered. All data products will be distributed within Rucio to the connected grid computing facilities for storage. Tape storage will be integrated in Rucio this time and dedicated grid locations are reserved to store the raw data product. XENONnT is the first hard Python3 dependency on Rucio. For analysts, the RCC in Chicago is again the main data analysis centre and provides user access to high level data products at a nearby location. Analysts can also define and produce own data products for analysis purposes outside the run database or grid storage at any time.

CMS performed an evaluation of data management systems from early 2018 through summer 2018, and eventually selected Rucio. The plan is to have Rucio deployed and ready for LHC Run 3: The transition period will last from 2018-2020, and the CMS team has expressed excitement about participating in a sustainable community project. The production infrastructure is based on Docker, Kubernetes, Helm, and OpenStack, with the official Rucio Helm charts customized with minimal configuration changes for CMS. The zero-to-operating cluster timing including dependencies is in the order of tens of minutes which allows fast and easy integration with CMS software and infrastructure; Rucio upgrades are nearly instantaneous. This also allows CMS to have its production and Rucio testbed on a shared set of resources. The developer's environment is identical to various flavors of central clusters, which makes integration easy. In 2019, the first full-fledged test distributed 1 million files between all CMS T1 and T2 data centers. The critical factor for data management scalability is the number of files, not the actual volume of data to be moved. The entire successful test took 1.5 days, and was purely driven by dataset injection rate. It ran in parallel to regular experiment activity.

The Deep Underground Neutrino Experiment (DUNE) is a neutrino experiment under construction, comprising a near detector at Fermilab and a far detector at the Sanford Underground Research Facility (SURF) that will observe neutrinos produced at Fermilab. DUNE's data management challenges are unique because they have multiple geographically separated detectors asynchronously collecting data, at an expected rate of tens of Petabytes per year. DUNE is also sensitive to supernovae, which potentially produce hundreds of Terabytes over a 100 second period. It is a large collaboration that intends to store and process data at many data centers

worldwide, and the current ProtoDUNE prototype detector has already recorded 6 Petabytes of reconstructed data. The next test beam run for both single and dual phase prototypes is expected in 2021-22. DUNE has a Rucio instance at Fermilab with a PostgreSQL backend, and has contributed several database extensions to Rucio. So far, 1+ million files have been cataloged from ProtoDUNE, including raw and reconstructed data. Rucio is being used to distribute ProtoDUNE data from CERN and FNAL to other data centers for analysts. The replication rules make this easy; making a rule for a dataset and data center or group of data centers eliminates operational overhead for DUNE. The current integration plan is to progressively replace the legacy data management system, and transition to a purely Rucio based solution. The main challenge is that DUNE intends to make significant use of HPC resources, and the data management system needs to integrate with many very heterogeneous supercomputing data centers. This is in line with the global HEP move towards using more HPC resources. Additionally DUNE data will benefit from fine grained object store style access, however it is not clear how to combine this with the traditional file based approach. The DUNE community has expressed that they are interested to contribute to these developments in the near future.

The Square Kilometre Array (SKA) is an intergovernmental radio telescope project to be built in Australia and South Africa. With receiving stations extending out to a distance of more than 3'000 kilometers from a concentrated central core, it will allow astronomers to create the most sensitive images of the Universe. The SKA Regional Centers will provide a platform for transparent data access, data distribution, post-processing, archive storage, and software development. Up to 1 Petabyte will be ingested from each telescope, and made available for access and post-processing around the globe. SKA will therefore need a way to manage data in a federated way across many physical data centers transparent to the user. SKA has begun evaluating Rucio for SRC data management. Data has been uploaded, replicated, and deleted from storage systems using custom replication rules and sustained data transfers have already been demonstrated from South Africa to the United Kingdom. A full mesh functional test has been put in place and is demonstrating connectivity. Tests were conducted using data from the LOFAR telescope, an SKA pathfinder instrument. Currently, the Elasticsearch/Logstash/Kibana (ELK) monitoring stack [23] is being set up up, and already 8M data operation events over more than one year of testing have been ingested. The evaluation experience using Rucio has been positive and is now formalized through the H2020 ESCAPE project, the European Science Cluster for Astronomy and Particle Physics ESFRI research infrastructures [24]. The main findings from the test include the arduous need for X.509 certificates across storage systems, which is now being addressed via alternatives such as token-based authentication and authorization. In addition, an in-depth look at the ELK monitoring and dashboards will be performed to see where they still need to be extended. Another major point is the integration with the DIRAC [25] workflow management system, matching the Belle II experiment needs, for a full end-to-end use case. Another use case will be similar to LHC Tier-0 processing with event-driven data management and processing. The inclusion of Australian storage for long-distance tests with a focus on network optimization is also upcoming.

The Laser Interferometer Gravitational-Wave (LIGO) Observatory [26], based in the US, is a large-scale observatory to detect cosmic gravitational waves and to develop gravitational-wave observations as an astronomical tool. Virgo [27] is the European equivalent interferometer, based in Italy at the European Gravitational Observatory (EGO). LIGO and Virgo are building the International Gravitational Wave Network (IGWN), with a combined 20 Terabytes of astrophysical strain data and 1 Petabyte of raw data per instrument per observing year. A data management solution is needed for offline deep searches and parameter estimation, as well as support for dedicated and opportunistic resources, as well as archival data. Rucio now enhances the IGWN data management through a large choice of protocols, an accessible catalog, comprehensive monitoring and support for detector data flows. This includes domain-specific daemons that register new dataframes in the Rucio catalog and then create rules to trivially implement dataflow to the archives and resources. IGWN has stated that they will investigate many opportunities beyond this, as well as being happy to update to a modern, high-availability version of existing functionality. Rucio is now being used in production for limited frame data replication to volunteering data centers, and a transition away from LDR is expected over the coming months. Upcoming work includes integration of existing data discovery services and remote data access, for example, enhanced database

redundancy, and management of new data products, for example, analysis pipeline data products. A mountable Rucio POSIX namespace is under development as an alternative for gravitational wave software distribution.

The Belle II experiment [28] is a particle physics experiment designed to study the properties of B mesons. The data requirements include 70 storage systems with around 200 Petabytes of data expected by the end of data-taking, with 2 replicas distributed over 6 data centers. Physics data taking started in 2019. Belle II's current distributed data management uses a bespoke design with adequate performance and supports up to 150'000 transfers/day. Some scalability issues in the system were addressed, but others are inherent to the design of the data management system, most importantly the lack of automation: this means that data distribution and deletion are done by experts at a very fine granularity. The Belle II team at Brookhaven National Laboratory (BNL) are evaluating Rucio as an alternative and all studies so far look promising. Most importantly, the performance on the PostgreSQL database at BNL shows capabilities beyond the Belle II requirements. Integration of Rucio with the rest of the Belle II distributed computing system, based on DIRAC, is planned in two stages. In the first stage the current data management APIs are replaced with an implementation that uses Rucio under the hood. This is mostly transparent to the rest of Belle II and allows bi-directional transition between the two implementations. However, this still relies on a legacy file catalog, and does not take full advantage of Rucio and its functionalities, being limited to the current APIs by definition. Nevertheless this stage allows the BNL team to gain experience in a production environment of using the DIRAC WFMS with Rucio. The second stage integration leads to an eventual migration that will use Rucio as the master file catalog, using a new DIRAC plugin to remove the dependency on the legacy file catalog.

## 5 Towards a common approach

During the evaluation together with these communities, Rucio has established a fully community-driven development process. Requirements and issues are publicly discussed via weekly development meetings, on GitHub [29], and group messaging based on Slack [30]. The project also hosts a yearly community workshop for developers and users to meet and to discuss the evolution of the software stack. The core development team provides guidance on design, architecture, as well as tests, and integration and evolves the development environment and continuous integration framework. Whereas packaging and high-level release planning is done by members of the core development team, the development is largely driven by contributors from the community. Contributions are reviewed by both the community as well as the core development team. Recent improvements in containerization and testing frameworks have significantly lowered the barrier to entry for newcomers. However, while contributing to the project has gotten simpler the project is specifically looking for maintained feature developments, thus sustainability is an important factor discussed with every contribution. One interesting aspect is that communities have started to help each other without the involvement of the core team, leading to a self-sustaining process that has been effective across time zones due to multi-continent involvement.

Although recent developments with containers and Kubernetes has made the deployment of Rucio very simple, the operation and maintenance of a data management system is still a significant effort for smaller scientific communities, which very often operate with very little personpower but still have significant data requirements. The UK Science and Technology Facilities Council (STFC) has been developing an enhancement to offer a data management service for multiple communities based on a single Rucio instance. This feature enables one Rucio instance to be virtually partitioned to serve multiple organizations, thus enabling communities to benefit from Rucio services while keeping the operational footprint low.

The need for operational cooperation has been acknowledged by many experiments, and a cross-experiment Operational Intelligence [31] effort has been started. Thousands of tickets are filed in the issue tracking system per year, which have to be followed by the operations teams of the various systems. In the context of Rucio, this effort seeks to exploit the wealth of dataflow traces to increase the level of automation. The first proposed models apply to the prediction of intelligent data placements and access patterns, time-series analyses to estimate of the

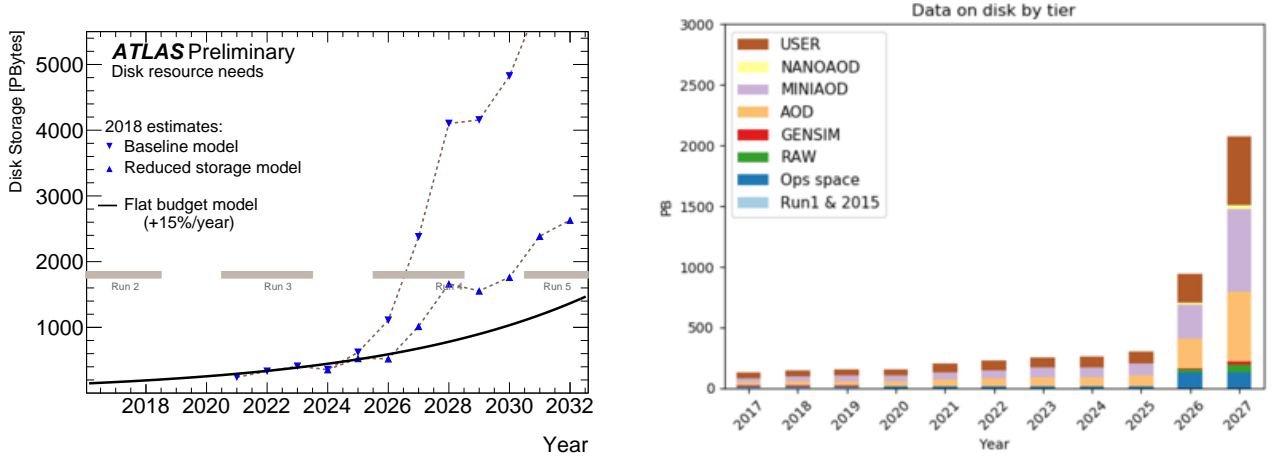


Figure 7: On the left, the estimated total disk resources from ATLAS in Petabytes needed for the years 2018 to 2032. On the right, the equivalent plot from CMS for the years 2017 to 2027. The expected data rates from HL-LHC will exceed funding capabilities even with reduced storage models after 2026.

time needed to complete transfers, or anomaly detection techniques to predict network failures. Recording and analyzing operator actions can be used to automate tasks and to suggest possible solutions to repeated issues.

Another oft-requested feature from the data centers was support for cloud storage to allow more dynamic possibilities for capacity increases. This was developed within the Data Ocean project [32], which was an R&D effort between ATLAS and Google. The idea of the project was to enable ATLAS to explore the use of different computing models by using Google resources, to allow ATLAS users to benefit from the Google infrastructure, and at the same time give Google real science use cases to improve their cloud platform. The project has been highly successful and follow-up developments enhanced the Rucio interfaces to be cloud provider agnostic. Now Rucio can serve as an enabler for scientific experiments who want transparent access to a multitude of diverse cloud storage providers.

Finally, the two major topics in scientific data management in the next years are the immediate growth rates of single experiments, and the resulting contention across experiment infrastructures on a limited set of shared storage and network resources. As shown in Figure 7 for ATLAS and CMS, the data growth at the HL-LHC beyond 2026 is significantly above any potential infrastructure growth. Therefore, the high energy physics community has prepared a white paper [33] to lay out the plans to tackle this challenge. At its core, it employs multiple strategies for data organization, management and access (DOMA) [34] that include mechanisms such as analysis model changes, dynamic use of storage quality of service, transparent distributed caching, network flow control using SDNs, and much more. Many of these strategies rely on having Rucio as a common data management system, with the objective to help steer dataflows across multiple experiments as a cooperating ensemble. The development of the exchange of dataflow state as well as cross-experiment namespace and scheduling will be crucial.

## 6 Conclusions

The LHC data needs have been driving a variety of data management developments for more than two decades. Throughout many attempts, the collected experiences led to the development of the Rucio system, which has proven flexible, efficient, and robust. The openness of the system, the autonomous declarative way of handling dataflows, the transparent handling of data incidents, and the capability to monitor the flows in detail have all contributed the success of Rucio. Many communities have now joined and are actively contributing. In

conclusion, Rucio is a successful collaborative open source project that is rapidly developing into a common standard for scientific data management.

## Acknowledgments

This work has been done as part of distributed computing research and development programs across many collaborating institutes, organizations, agencies, and communities. We thank all our colleagues for their support.

## References

- [1] L. Evans and P. Bryant. LHC Machine. JINST 3:0 S08001, 2008. <https://doi.org/10.1088/1748-0221/3/08/S08001><https://doi.org/10.1088/1748-0221/3/08/S08001>
- [2] G. Apollinari, O. Brüning, T. Nakamoto and L. Rossi. High Luminosity Large Hadron Collider HL-LHC. CERN Yellow Rep. (2015) no.5, 1. <https://doi.org/10.5170/CERN-2015-005.1><https://doi.org/10.5170/CERN-2015-005.1>
- [3] R. Acciarri et al. Long-Baseline Neutrino Facility (LBNF) and Deep Underground Neutrino Experiment (DUNE). 2016. <https://arxiv.org/abs/arXiv:1601.05471><https://arxiv.org/abs/arXiv:1601.05471>
- [4] A. Chrysostomou, R. Bolton, G. R. Davis. The Square Kilometre Array: Challenges of distributed operations and big data rates. Proc. Observatory Operations: Strategies, Processes, and Systems VII; Volume 10704:1070419 (2018). <https://doi.org/10.1117/12.2309554><https://doi.org/10.1117/12.2309554>
- [5] ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. JINST, 3:0 S08003, 2008. <https://doi.org/10.1088/1748-0221/3/08/S08003><https://doi.org/10.1088/1748-0221/3/08/S08003>
- [6] CMS Collaboration. The CMS Experiment at the CERN LHC. JINST, 3:0 S08004, 2008. <https://doi.org/10.1088/1748-0221/3/08/S08004><https://doi.org/10.1088/1748-0221/3/08/S08004>
- [7] ALICE Collaboration. The ALICE experiment at the CERN LHC. JINST, 3:0 S08002, 2008. <https://doi.org/10.1088/1748-0221/3/08/S08002><https://doi.org/10.1088/1748-0221/3/08/S08002>
- [8] LHCb Collaboration. The LHCb Detector at the LHC. JINST, 3:0 S08005, 2008. <https://doi.org/10.1088/1748-0221/3/08/S08005><https://doi.org/10.1088/1748-0221/3/08/S08005>
- [9] ATLAS and CMS Collaborations. Combined measurement of the Higgs boson mass in pp collisions at  $\sqrt{s}=7$  and 8 TeV with the ATLAS and CMS experiments. Phys.Rev.Lett., 114:0 191803, 2015. <https://doi.org/10.1103/PhysRevLett.114.191803><https://doi.org/10.1103/PhysRevLett.114.191803>
- [10] ATLAS Collaboration. Summary of the ATLAS Experiment’s sensitivity to supersymmetry after LHC Run 1 - interpreted in the phenomenological MSSM. JHEP, 1510:0 134, 2015. [https://doi.org/10.1007/JHEP10\(2015\)134](https://doi.org/10.1007/JHEP10(2015)134)[https://doi.org/10.1007/JHEP10\(2015\)134](https://doi.org/10.1007/JHEP10(2015)134)
- [11] P. Braun-Munzinger and J. Stachel. The quest for the quark-gluon plasma. Nature 448 (2007), 302-309. <https://doi.org/10.1038/nature06080><https://doi.org/10.1038/nature06080>
- [12] M. Pepe Altarelli and F. Teubert. B Physics at LHCb. Int.J.Mod.Phys A23 (2008), 5117-5136. <https://doi.org/10.1142/S0217751X08042791><https://doi.org/10.1142/S0217751X08042791>
- [13] ATLAS Collaboration. Search for TeV-scale gravity signatures in high-mass final states with leptons and jets with the ATLAS detector at  $\sqrt{s} = 13$  TeV. Phys.Lett. B, 760:0 520–537, 2016. <https://doi.org/10.1016/j.physletb.2016.07.030><https://doi.org/10.1016/j.physletb.2016.07.030>
- [14] ATLAS Collaboration. Constraints on mediator-based dark matter and scalar dark energy models using  $\sqrt{s} = 13$  TeV pp collisions at the LHC with the ATLAS detector. ATLAS-CONF, 051, 2018. <https://inspirehep.net/record/1702555><https://inspirehep.net/record/1702555>

- [15] A. Filipčič. ATLAS Distributed Computing Experience and Performance During the LHC Run-2. J.Phys.Conf.Ser. 898(5):052015. <https://doi.org/10.1088/1742-6596/898/5/052015><https://doi.org/10.1088/1742-6596/898/5/052015>
- [16] X. Zhao, A. Klimentov et al. ATLAS Data Carousel. Submitted to EPJ Web.Conf. (2020). <https://indico.cern.ch/event/773049/contributions/3474425><https://indico.cern.ch/event/773049/contributions/3474425>
- [17] M. Barisits, T. Beermann, F. Berghaus et al. Rucio: Scientific Data Management. Comput Softw Big Sci (2019) 3:11. <https://doi.org/10.1007/s41781-019-0026-3><https://doi.org/10.1007/s41781-019-0026-3>
- [18] M. Branco et al. Managing ATLAS data on a petabyte-scale with DQ2. J.Phys.Conf.Ser. 119 (2008) 062017. <https://doi.org/10.1088/1742-6596/119/6/062017><https://doi.org/10.1088/1742-6596/119/6/062017>
- [19] M.Lassnig et al. Rucio beyond ATLAS: Experiences from Belle II, CMS, DUNE, EISCAT3D, LIGO/VIRGO, SKA, XENON. Submitted to EPJ Web.Conf. (2020). [https://indico.cern.ch/event/773049/contributions/3474416/attachments/1937611/3211761/Rucio\\_CHEP19.pdf](https://indico.cern.ch/event/773049/contributions/3474416/attachments/1937611/3211761/Rucio_CHEP19.pdf)<https://indico.cern.ch/event/773049/contributions/3474416>
- [20] AMS-02 RICH Collaboration. The AMS-02 RICH detector: Status and physics results. Nucl.Instrum.Meth. A952 (2020) 161797. <https://doi.org/10.1016/j.nima.2019.01.024><https://doi.org/10.1016/j.nima.2019.01.024>
- [21] E. Aprile et al. Dark Matter Search Results from a One Ton-Year Exposure of XENON1T. Phys.Rev.Lett. 121 2018 111302. <https://doi.org/10.1103/PhysRevLett.121.111302><https://doi.org/10.1103/PhysRevLett.121.111302>
- [22] D. Ahlin et al. The XENON1T Data Distribution and Processing Scheme. EPJ Web Conf. 214 2019 03015. <https://doi.org/10.1051/epjconf/201921403015><https://doi.org/10.1051/epjconf/201921403015>
- [23] O. Andreassen, C. Charrondière and A. De Dios Fuente. Monitoring Mixed-Language Applications with Elastic Search, Logstash and Kibana (ELK). <https://doi.org/10.18429/JACoW-ICALEPCS2015-WEPGF041><https://doi.org/10.18429/JACoW-ICALEPCS2015-WEPGF041>
- [24] ESCAPE - European Science Cluster of Astronomy & Particle physics ESFRI research infrastructures. H2020-INFRAEOSC-2018-2. Grant agreement ID: 824064. <https://projectescape.eu><https://projectescape.eu>
- [25] F. Stagni et al. DIRAC in Large Particle Physics Experiments. J.Phys.Conf.Ser. 898 (2017) no.9, 092020. <https://doi.org/10.1088/1742-6596/898/9/092020><https://doi.org/10.1088/1742-6596/898/9/092020>
- [26] J. Aasi et al. Advanced LIGO. Class. Quantum Grav. 32, 074001 (2015). <https://doi.org/10.1088/0264-9381/32/7/074001><https://doi.org/10.1088/0264-9381/32/7/074001>
- [27] F. Acernese et al. Advanced Virgo Status. J. Phys. Conf. Ser. 1342 (2020) no.1, 012010. <https://doi.org/10.1051/epjconf/201818202003><https://doi.org/10.1051/epjconf/201818202003>
- [28] T. Kuhr. Belle II at the Start of Data Taking. EPJ Web Conf. 214 (2019) 09004. <https://doi.org/10.1051/epjconf/201921409004><https://doi.org/10.1051/epjconf/201921409004>
- [29] Rucio Repository <https://github.com/rucio/rucio><https://github.com/rucio/rucio>
- [30] Rucio Team Slack <https://rucio.slack.com><https://rucio.slack.com>
- [31] ATLAS and CMS Collaborations. Operational Intelligence. Submitted to EPJ Web.Conf. (2020). [https://indico.cern.ch/event/773049/contributions/3473362/attachments/1937928/3212143/CHEP\\_2019\\_Operational\\_Intelligence\\_-\\_rev05.pdf](https://indico.cern.ch/event/773049/contributions/3473362/attachments/1937928/3212143/CHEP_2019_Operational_Intelligence_-_rev05.pdf)<https://indico.cern.ch/event/773049/contributions/3473362>
- [32] ATLAS Collaboration. The Data Ocean Project: An ATLAS and Google R&D collaboration. EPJ Web Conf. 214 (2019) 04020. <https://doi.org/10.1051/epjconf/201921404020><https://doi.org/10.1051/epjconf/201921404020>

- [33] HEP Software Foundation Collaboration. A Roadmap for HEP Software and Computing R&D for the 2020s. *Comput.Softw.Big Sci.* 3 (2019) no.1, 7 <https://doi.org/10.1007/s41781-018-0018-8><https://doi.org/10.1007/s41781-018-0018-8>
- [34] D. Berzano et al. Data Organization, Management and Access (DOMA) White Paper. <http://arxiv.org/abs/arXiv:1812.00761><http://arxiv.org/abs/arXiv:1812.00761>



# Advancing RPC for Data Services at Exascale

Jerome Soumagne<sup>1</sup>, Philip Carns<sup>2</sup>, and Robert B. Ross<sup>2</sup>

<sup>1</sup>The HDF Group

<sup>2</sup>Argonne National Laboratory

## Abstract

*Remote Procedure Call (RPC) has long been an inherent component of parallel file systems and I/O forwarding middleware in high-performance computing (HPC). RPCs are used in this environment to issue I/O operations and transfer data from compute nodes to gateway and server storage nodes. With HPC systems becoming more heterogeneous, data volumes reaching new thresholds, and I/O standing as the main bottleneck, there is a growing need in the HPC community to build distributed services and adopt new workflows that are, nonetheless, no longer dictated by monolithic parallel file systems. These include specialized storage, data analysis, and telemetry services that can be adapted to fit application needs. Parallel file system RPC facilities have never been exposed to service or middleware developers, however, leaving them with two choices: MPI or the low-level fabric network protocol. In this article, we show how an independent RPC framework can be used as a building block for developing user-level data services at exascale. We identify the design choices that must be considered in terms of both performance and resilience for HPC data services, and we discuss the directions taken to palliate current HPC system constraints.*

## 1 Introduction

High-performance computing (HPC) facilities have traditionally been designed around *monolithic* file systems, which are tailored to scientific HPC workflows comprised of computation, storage, and data analysis. Scientific application users, whose needs depend on the application's domain, have been constrained to conform to system precepts and this standard workflow. While this has been a viable (but increasingly limiting) option for pre-exascale systems, increasing data volumes and increasing system complexity with emerging hardware are now forcing application users to adopt new *specialized* workflows. These specialized workflows not only achieve sustainable performance and perform data analysis in a timely manner at an increasing scale, but also better respond to application needs and provide data insights, for example through monitoring and telemetry service.

Creating specialized workflows requires the introduction of a collection of *data services* to the HPC ecosystem that must interact with both the system components (hardware and software) and the application. While some of those services may be provided by the system, the vast majority of data services are user-level services that are developed to augment the original HPC system software stack and better serve the application's performance or functionality needs. Data services (system-provided or user-provided) must respond, in most cases,

---

Copyright 2020 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

to the same user prerequisites by ensuring performance, resilience, and ease of deployment. These prerequisites introduce engineering challenges that must be overcome when creating a new HPC data service—by no means an easy task. One such challenge is communication: data exchange between services is a critical aspect of specialized workflows that are composed of multiple services interacting with each other. Developing the messaging part of a data service component on an HPC machine can, for a new service developer, involve either using the low-level network fabric API, which requires a significant amount of work and expertise, or using the vendor installed MPI library [1] that takes advantage of the underlying network fabric. MPI itself, however, is not very suitable for developing such dynamical services that may come and go [2]. MPI implementations have consistently prioritized use by applications and not by service libraries.

Data services are already a well-established technology in the cloud, where remote procedure call (RPC) is the main technique used for sending messages to remote components. Google gRPC [3] or Facebook Thrift [4] are good examples of such frameworks. However, they are not well-suited to run on HPC systems because they (1) rely on the TCP/IP stack and do not take advantage of the low latency/high bandwidth HPC fabrics and (2) are not designed for exchanging very large amounts of data, a task that is left to the user. In contrast, RPC has been used as the communication pillar of distributed file systems (e.g., Lustre Networking (LNET) [5], Panasas [6]) and I/O forwarding layers (e.g., IOFSL [7]) that are specifically designed to send I/O requests on top of the underlying network fabric. The Network File System (NFS) [8] is also a good example of the use of RPC with large data transfers and therefore close to the use of RPC in an HPC system. The internal RPC facilities of these file systems (with the exception of NFS) have, nonetheless, never been directly exposed to users; instead, they have been deeply buried in the monolithic file system software stacks that often extend into kernel space. Other parallel file systems have implemented their own network abstraction layer to support multiple network fabrics and provide messaging capabilities that can support data services. However, they are not general-purpose RPC frameworks, and in most cases cannot be easily extracted from the file systems that they were designed for.

Based on both of those technologies and past experience with I/O forwarding, we introduced in [9] an RPC framework, called Mercury, that takes advantage of low-level HPC network fabrics and facilitates the development of user-level data services. Mercury is part of a more comprehensive suite of components named Mochi [10] that provides a collection of service components for the creation of specialized data services. We present in this paper how some of the design choices made for Mercury are essential for building an heterogeneous service workflow in an exascale HPC environment. In Section 2, we present some of the work that is similar to Mercury and approaches that we take to develop user-level data services. In Section 3, we give a brief overview of Mercury’s architecture before focusing in Section 4 on the specific design points that make an RPC framework usable for HPC data services, supporting our claims by evaluation results. In Section 5, we present some of the data services that are successfully being deployed using Mochi and Mercury. In Section 6, we summarize our conclusions.

## 2 Related Work

A few other frameworks and suites of HPC data service components have been proposed using an approach similar to the one we used in Mercury. We present here three of the most notable frameworks.

*DataSpaces* [11] implements a scalable, semantically specialized shared-space abstraction that is dynamically accessible by all components and services in an application workflow, supporting both application/system-aware data placement and data movement. It relies on the *Decoupled and Asynchronous Remote Transfers* (DART) [12] layer, which is not defined as an explicit RPC framework, although it allows transfer of large amounts of data using a client/server model from applications running on the compute nodes of an HPC system to local storage or remote locations, in order to enable remote application monitoring, data analysis, code coupling, and data archiving. The key requirements that DART seeks to satisfy are minimizing data transfer overheads on the application; achieving high throughput, low latency data transfers; and preventing data losses.

To this end, DART is designed so that dedicated nodes (i.e., separate from the application compute nodes) asynchronously extract data from the memory of the compute nodes using remote direct memory access (RDMA).

The *Scalable Observation System* (SOSflow) [13] provides a broad set of online and in situ capabilities, including code steering via remote method invocation, data analysis, and visualization. SOSflow can couple together multiple sources of data, such as application components and operating environment measures, with multiple software libraries and performance tools. SOSflow’s communication mechanism relies both on TCP sockets for on-node communication and on MPI for off-node communication. Its main communication pattern is a publish-and-subscribe mechanism and relies on a daemon that is launched as a background process in user space at the start of a job script, before the scientific workflow begins.

*Faodel* [14] provides a set of services for data management and exchange in HPC workflows. Three major components of Faodel are Kelpie, Opbox, and Lunasa. Kelpie provides a key-blob abstraction. OpBox is a library for implementing asynchronous communication between multiple entities in a distributed application, and provides the user with primitives for expressing a protocol as a state machine that the communication layer can process in an asynchronous manner. It also provides a naming service to locate components of an application. Lunasa provides user-level network memory management services and effectively acts as a memory registration cache for doing RDMA. Faodel relies on an evolution of the NNTI layer from the *Network Scalable Service Interface* (Nessie) [15] RPC library. It provides an asynchronous RPC solution, designed to overlap computation and I/O. Nessie also provides a mechanism to handle bulk data transfers, which can use RDMA to transfer data efficiently from one memory to the other, and supports several network transports. Nessie uses the RPC interface to push control messages to the servers and exposes a separate one-sided API that is used to push or pull data between client and server.

### 3 Overview and Considerations

Mercury is designed around three key paradigms: provide reliable RPC functionality, support large data arguments, and take advantage of the HPC network fabrics. In terms of functionality, much more is needed when developing distributed HPC data services; but as opposed to RPC frameworks that are part of monolithic software stacks, Mercury remains as thin as possible in order to allow for reusability between various service components that must support different needs.

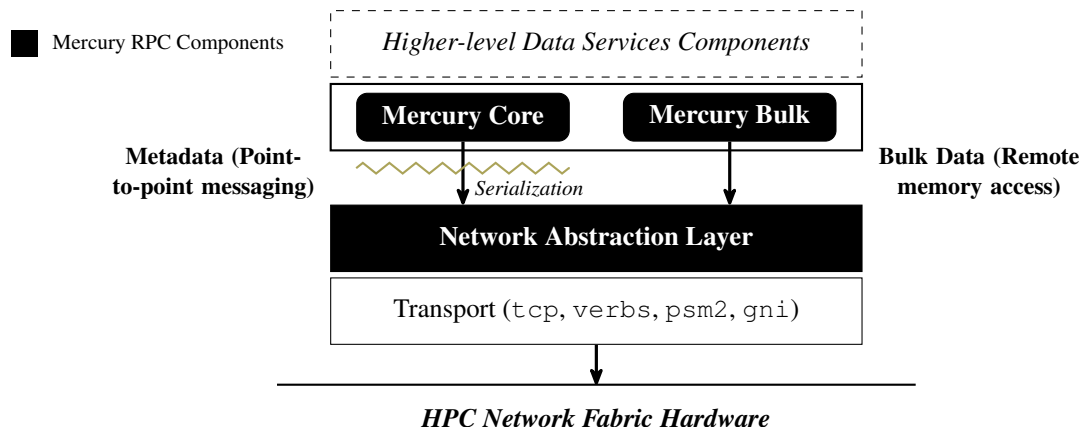


Figure 1: Overview of Mercury RPC components in the software stack.

As shown in Figure 1, Mercury is composed of two service-level components: a core RPC component, which is designed to serialize function arguments and send them to a remote target for execution using point-to-point messaging, and a bulk component, which is designed to handle large arguments (i.e., arguments that

are generally larger than 4KB depending on the underlying protocol being used). This latter component enables the creation of memory descriptors that can be sent along with the other arguments to the RPC target to initiate raw memory transfers (without serialization) using remote memory access (RMA). In Section 4.1, we detail this scenario and its benefits. In order to support a large variety of HPC network fabrics, both of these components interface with a network abstraction layer that provides a minimum set of network primitives for both point-to-point messaging and one-sided RMA communication operations. Moreover, in order to reduce the burden of connection handshakes when the underlying network does not necessarily request it (also essential for scalability) and to support services that may come and go, remote peers are addressed through unconnected endpoints. Furthermore, in order to maximize throughput, all communication is made nonblocking through a callback-based approach that we detail in Section 4.5.

While these points describe the overall architecture of an RPC framework for HPC, additional key items can rapidly become prerequisites for creating an RPC framework that is designed to support data services. These include maximizing throughput, providing scaling, enabling flexibility, and ensuring resilience. In the following section we describe how one can enhance RPC to (1) leverage RDMA-capable networks; (2) support node-local service scaling and leverage multi-core processors; (3) enable flexible, node-local deployment scenarios and service composition; (4) bridge nodes between multiple HPC networks; (5) enable fault tolerance.

## 4 Enabling RPC for HPC Data Services

We do not compile an exhaustive list of features in this section. Instead, we focus on those features that are necessary to enable strong service scaling, performance, flexibility, and resilience for data services on emerging large-scale computing platforms.

### 4.1 HPC Network Support

As opposed to cloud-based RPC solutions that rely on TCP networking, HPC network fabrics provide dedicated solutions that offer both low latency and high bandwidth. To take advantage of these solutions, however, an RPC framework must leverage low-level vendor APIs such as InfiniBand™ Verbs, Intel® Performance Scaled Messaging 2 (PSM2), and Cray® Generic Network Interface (GNI). Rather than implementing Mercury’s network abstraction layer directly on top of those APIs, we currently use OFI libfabric [16] as the intermediate layer that abstracts RDMA capabilities for RDMA-capable networks or emulated RMA (over point-to-point) for noncapable networks. Exposing native RDMA primitives is essential for taking full advantage of RDMA capable networks so that a data service can, for large data, leverage zero-copy transfers from the application’s memory from/to the storage.

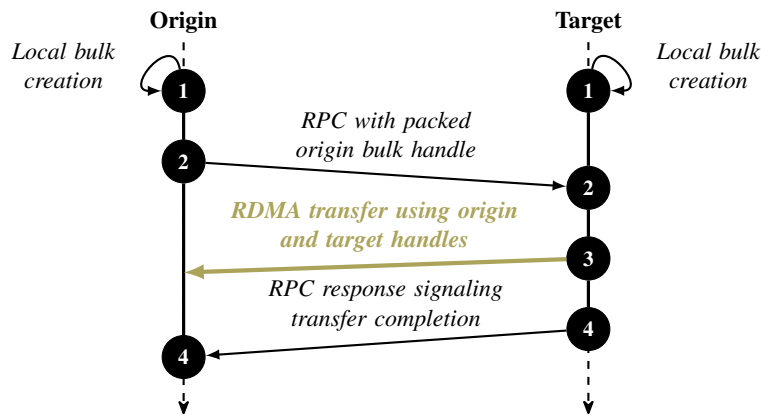


Figure 2: Four-step process of Mercury’s bulk RDMA transfers.

Enabling RMA capabilities through Mercury’s bulk component (see Figure 1) is a four-step process (see Figure 2). First, *bulk handles*, which are abstract memory descriptors, must be created on both origin and target processes. During handle creation, memory regions are registered (which in most cases corresponds to a physical hardware registration); this allows for the higher level data service to only expose memory pages that it wishes to access in either read-write or read-only mode. Second, an RPC is issued from the origin process to the target process with the serialized bulk handle of the origin process; this handshake allows the target process to gather virtual address information, registration keys, and so forth, which are necessary for the underlying protocol to post an RDMA operation. Third, the actual RDMA operation is posted using both the target’s local bulk handle and the origin’s handle that was transmitted through the RPC. Since bulk handles are abstract memory descriptors, more complex scenarios such as scatter/gather can be transparently implemented and even delegated to the hardware if the hardware provides this support, allowing for more efficient transfers. Finally, the RPC response is sent, effectively signaling the origin of the transfer completion. This server-driven four-step process is the most conventional model for data transfers in Mercury, but client-driven transfers are legal as well. The former is more commonly recommended for two reasons. First, it enables servers to throttle or re-order transfers according to load. Second, it makes the clients lighter weight and more scalable, since they do not have to track the state of server resources.

**Evaluation.** To show the importance of supporting this capability, we compare the RPC performance and “RPC with bulk” performance on an InfiniBand cluster (Cooley) that is equipped with 4X FDR Infiniband cards (56 Gb/s). Compared to TCP over the same network, our approach improves RPC throughput with close to  $9 \times 10^5$  operations per second and close to 6,000 MB/s average throughput when performing RPC and bulk transfer through the native verbs interface. Note that the previous results do not use any multi-threading capabilities. We maintain a number of 32 RPCs in-flight to ensure sustained performance. Multi-threading support is discussed in the next section.

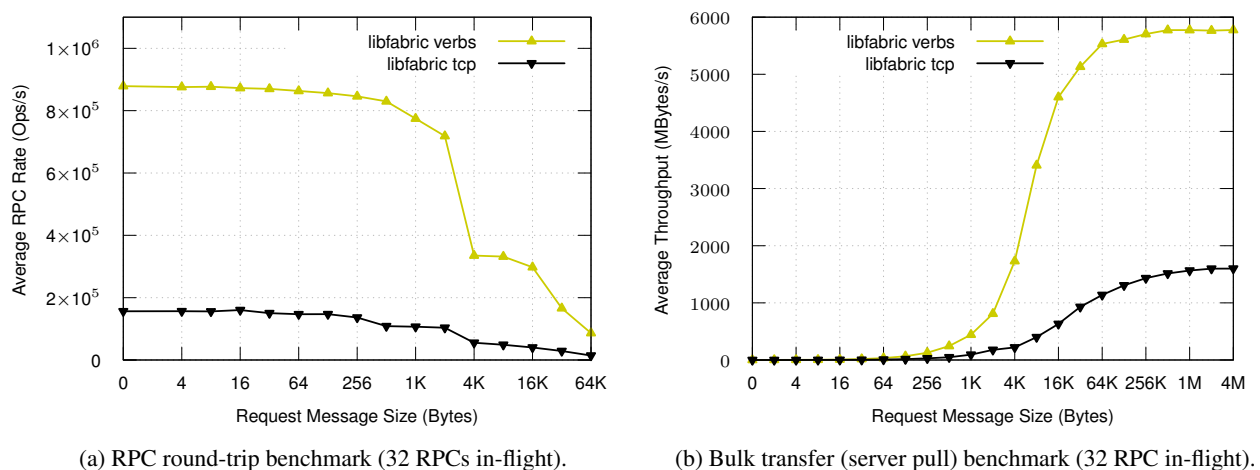


Figure 3: Effect of leveraging RDMA network on InfiniBand cluster (FDR InfiniBand).

## 4.2 Multi-Core Architecture Support

With CPUs experiencing increasing core count and lower frequencies per core, data services are expected to take advantage of these architectures by either distributing the load of incoming RPCs across cores or by running multiple services co-located within the same node. Communication frameworks typically adopt one of two progress models: either *explicit* or *implicit*. *Explicit* progress implies that the user will regularly make progress

calls to effectively check into network completion queues, poll file descriptors, etc. In contrast, an *implicit* progress model will make progress in background without any need for the user to be involved. However, this usually involves a background progress thread running to make progress while operations are being posted. While this may seem convenient, this “hidden” thread can become detrimental when running concurrently with other user’s threads, leading to unexpected scheduling issues. Therefore, to prevent this type of issues and give data services sufficient flexibility in how progress is ensured, we follow an explicit progress model. RPC is not only about messaging and communication, it is also about execution of user-defined function calls. When making progress, therefore, it is often desirable to decouple the RPC execution activities from the network progress activities, which leads us to actually adopt a *progress-and-trigger* model that gives services more control over the placement of the progress and execution threads. In this approach, implicit progress can be accomplished by the user by having a thread calling progress in background.

In a typical scenario, an RPC listener service will start posting RPC receive operations with memory bound to the thread posting the operations. Distributing the execution of these incoming RPCs across multiple threads (e.g., using a thread pool) can lead to several context switches at a significant performance penalty. To prevent this scenario, take advantage of multi-core architectures, and allow for node-local service scaling without costly creation of separate endpoints per thread, we make use of *scalable endpoints* (SEP) when available. Scalable endpoints are provided through libfabric [16] but can be extended through our network abstraction layer. Scalable endpoints allow for sharing a single endpoint resources between threads by assigning separate transmit and receive contexts (including completion queues) to each thread. When SEPs are used, context switches between threads no longer exist—a fundamental advantage for RPC multi-core architectures.

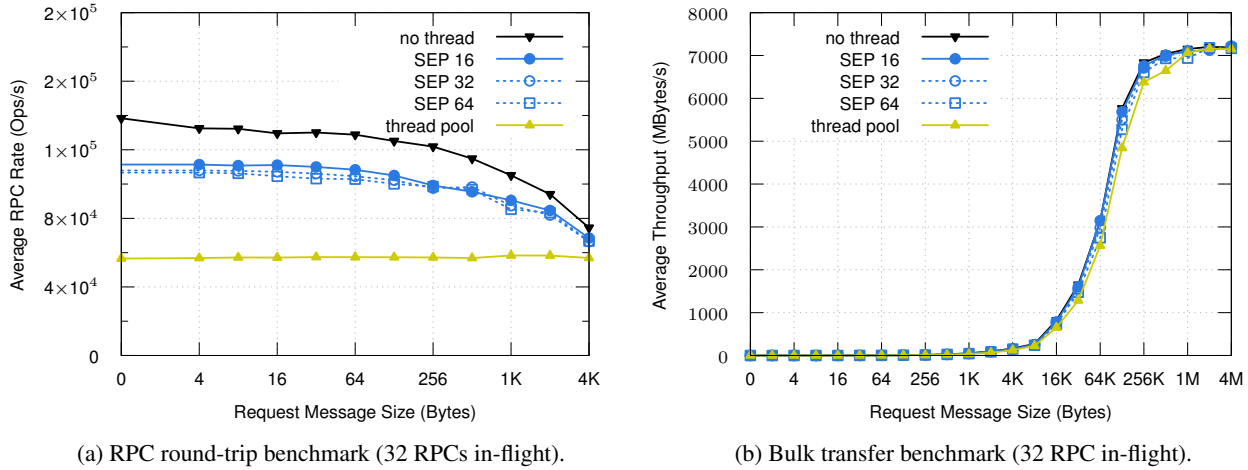


Figure 4: Effect of using scalable endpoints on Cray XC40 (Aries interconnect).

**Evaluation.** To demonstrate the impact of context switches and emphasize the benefits of scalable endpoints, we run two benchmarks on the Theta supercomputer at the Argonne Leadership Computing Facility (ALCF). Theta is a Cray XC40 system with a second-generation Intel Xeon Phi processor and Cray Aries interconnect. Each compute node is a single Xeon Phi chip with 64 cores, 16 GB of Multi-Channel DRAM (MCDRAM), and 192 GB of DDR4 memory. Users typically take advantage of this architecture by either deploying multiple data services locally or by distributing incoming RPCs across cores. In order to do so using SEP, we assign each core to make progress and trigger calls on their own receive context. As shown in Figure 4, using SEP provides close match (in terms of operations per second) to the performance of workloads that do not use multi-threading. Distributing requests using a thread pool, in contrast, has a significant detrimental impact on RPC rate. Note

that in all cases bulk transfers exhibit similar overall bandwidth, as context switches only represent a portion of the time spent when large data is transferred over the network.

### 4.3 Flexible Provisioning and Topology

In the preceding section, we demonstrated node-level scaling when RPCs are made between separate nodes using the native interconnect. Additional optimization can be made, however, by being aware of node-local process placement, in order to ensure efficient composition of services.

#### 4.3.1 Transparent Node-Local Deployment

When deploying data services, it is common for some of these services to either issue RPCs to other local services (i.e., separate processes within the same node) or to send RPCs back to themselves (i.e., within the same process). The latter typically arises out of convenience, rather than creating a separate code path for that case. To achieve the former, Mercury can make use of shared-memory transparently by detecting that the target address is on the same node. Using lockless shared ring buffers and lockless queues, it is possible to achieve lockless transfers with very low latency. For bulk data transfers and to prevent any intermediate *memcpy*, zero-copy transfers (i.e., one single and direct copy from origin to target buffer) can be achieved using the Linux Cross-Memory Attach mechanism.

To achieve the latter, Mercury detects when the target address is the same as the origin address and sends RPCs using the same argument packing mechanism, by immediately queuing the RPC into a local completion queue, internally signaling completion to wake up any potential thread waiting in a progress call. Likewise, bulk data transfers are realized through a *memcpy* between source and destination buffers.

This combination of transparent shared-memory transfers between separate processes, loopback redirection within the same process, and over-the-wire transfers has shown substantial benefits when deploying data services in terms of performance and flexibility, since data services can treat all three scenarios identically.

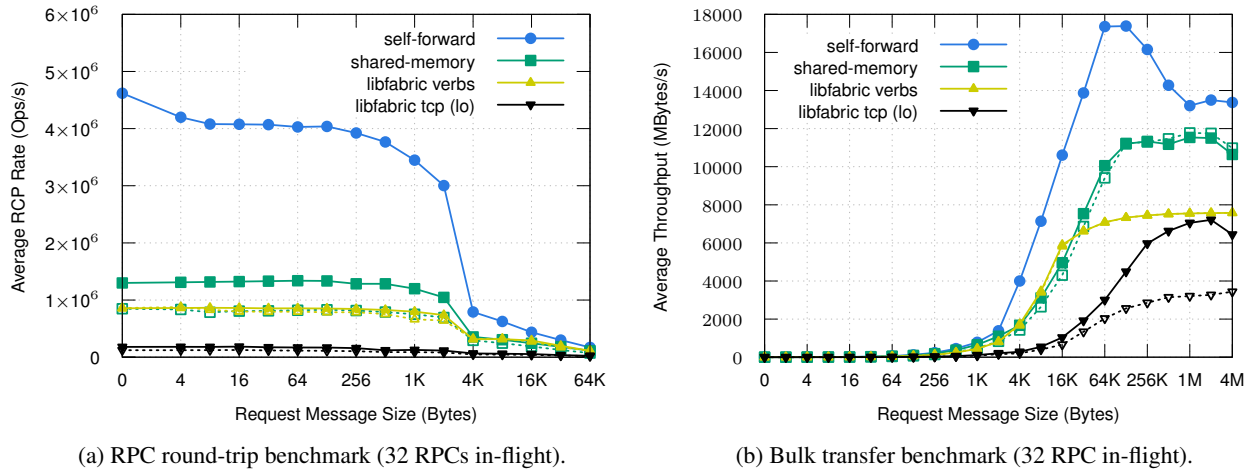


Figure 5: Comparison between node-local RPC mechanisms on InfiniBand cluster (FDR InfiniBand).

**Evaluation.** To illustrate this scenario on our InfiniBand cluster (Cooley), we compare in Figure 5 our two local RPC communication mechanisms to issuing RPCs either through the native interconnect (in this case InfiniBand Verbs) or through TCP and the loopback interface. The latter is one of the fallback mechanisms



typically used when not using shared-memory. Cooley is equipped of dual-sockets nodes with Intel Xeon E5-2620 v3 CPUs. Consequently, performance varies depending on process placement and the NUMA nodes being used—performance when running on separate NUMA nodes is represented by a dotted line in Figure 5. In terms of both RPC operations per second and bulk throughput, these two mechanisms are very valuable, providing much better performance than both the native interconnect and TCP (1.3 MOps/s for shared-memory and more than 4 MOps/s for loopback execution). When running on separate NUMA nodes, shared-memory performance is naturally impacted, though RPCs with bulk transfer still perform at a much higher rate due to the use of Linux Cross-Memory Attach (CMA).

### 4.3.2 Service Composition

With node-level scaling and transparent node-local deployment in place, composing data services seems the next natural step. In order to provide flexible composition, the RPC API must not be specific to any implementation but rather rely only on *origin* and *target* concepts. The RPC mechanism then can be consistently employed to communicate between different service “servers” and “clients”.

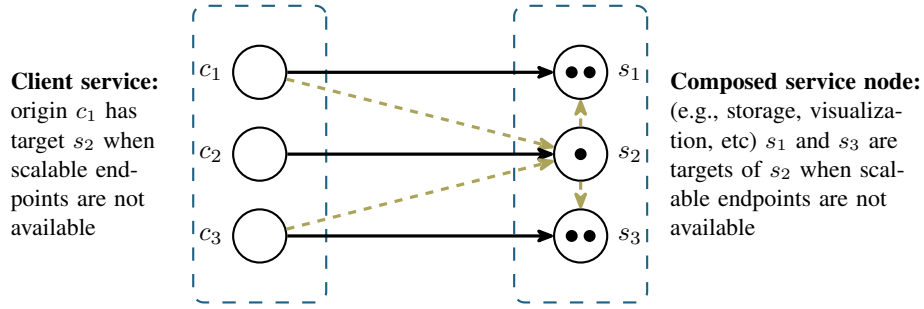


Figure 6: Composition of services with and without scalable endpoints.

When multiple services are colocated, there is also a need for addressing specific services and efficiently making progress. As shown in Figure 6, this can then be accomplished by using a “delegator” service, which can potentially become a bottleneck, or by using scalable endpoints addressing specific receive contexts directly through an ID that can be defined for each data service. When there is no hardware support for scalable endpoints, however, this functionality must be emulated by embedding a service ID into the RPC header and using that ID to distribute RPC requests to the corresponding service through that delegator. An alternative is to create multiple endpoints, one for each data service; but this is usually not recommended due to hardware resource limitations.

## 4.4 Multi-Network Support

As we bridge local and nonlocal communication mechanisms, supporting multiple fabrics follows a similar approach and relies on the same supporting components described in Sections 4.2 and 4.3. Mercury’s architecture defines *classes* that physically correspond to one endpoint and *contexts* that correspond to completion queues and locally allocated resources. When using scalable endpoints as described in Section 4.2, we are in a scenario with one class (one endpoint) and multiple contexts (multiple completion queues) that share the same endpoint. When bridging multiple fabrics, we are in a scenario with multiple classes (multiple endpoints) and one or more contexts (completion queues) associated with each class.

The challenge is efficiently making progress over these separate classes and contexts. To facilitate this, Mercury provides two progress mechanisms, allowing for a service to either busy spin on each of these contexts to process requests as quickly as possible (at the cost of using more CPU resources), or to wait and sleep



on this set of contexts until a new request arrives. In the latter case, we rely on Linux’ file descriptor and *epoll* mechanism to wait. This allows for monitoring of both local event notifications and hardware queue notifications. This transparent notification mechanism allows a data service implementation to simply wait on a file descriptor rather than manually making progress on each of the interfaces/endpoints.

## 4.5 Resilience and Fault Tolerance

When supporting data services at scale, there are multiple approaches that one can take to define a resilient RPC mechanism (for instance, guaranteed delivery). One of the primary requirements for an RPC component is to allow services to recover after a fault has occurred (e.g., node failure, unresponsiveness of a service component) without compromising performance, by simply providing robust support for canceling operations that are pending. This implies reclaiming local resources that RPC operations have previously allocated and gracefully recovering from faults. It is important to note that we assume in that discussion the use of *reliable* unconnected endpoints in the transport layer, hence RPC requests do not get “lost”. Ordering and tag matching are not critical for the transport to provide though (Mercury matches messages itself when needed). Mercury itself only provides *at-most* once semantics: nonblocking RPC requests are sent and a nonblocking response is sent back (unless it is explicitly stated not to do so). It is then up to services to make their own decision on how to react (e.g., retry, fail over, initiate a rebuild, etc). Both RPC and bulk data transfer operations may be interrupted if any of the peers involved no longer responds, in which case pending operations must be canceled. Canceling an operation that cannot complete, either because a fault has occurred or a timeout has been reached, is necessary in order to reach proper completion.

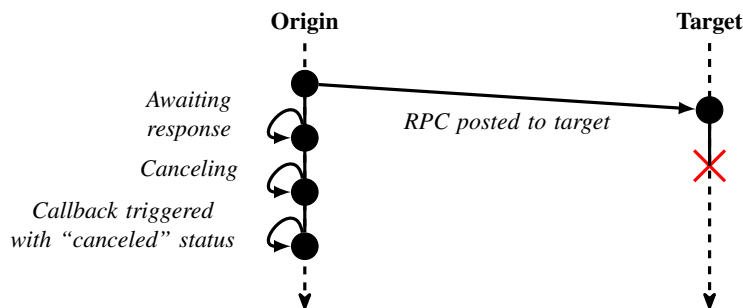


Figure 7: Cancellation of an RPC operation.

Cancellation of operations in Mercury is always an *asynchronous* and *local* operation. As shown in Figure 7, forwarding an RPC request is a nonblocking operation. Therefore, since Mercury follows a callback-based mechanism, completion of that operation is known from a user’s perspective only when the callback that is associated to that operation is pushed to the local completion queue and later triggered after making both progress and trigger calls. When that callback is triggered, the state of the operation is reported to the user as *canceled*. Since operations are nonblocking, keeping cancellation an asynchronous operation instead of an operation that completes immediately is essential. This mechanism protects against races in the event that a peer response arrives after local cancellation has already succeeded. After the callback is triggered, it is then safe to re-use the existing RPC request handle to issue a retry for example.

Cancellation is the foundation for implementing timeout scenarios in data services in order to recover from a fault. When an internal fault occurs, however, cancellation of the operation is not always necessary if the RPC has not yet been posted, in which case that operation can simply be directly retried. This scenario is similar for all other nonblocking operations in Mercury, including bulk data transfers.

## 5 Applications and Use Cases

As mentioned in Section 1, Mercury is part of the Mochi suite of service components. Mochi provides additional features on top of Mercury such as the notion of group membership, transparent user-level thread semantics, key/value stores, C++ and Python bindings. This work is further described in [10] along with additional use cases, including the following:

Intel’s *Distributed Application Object Storage* (DAOS) [17] project provides a transactional and multidimensional object store for use in large-scale HPC environments with embedded storage directly attached to the compute fabric. DAOS is a vendor-backed push to provide an alternative to the traditional parallel file system and has the potential to extract higher performance out of emerging low latency storage technology by running in user space. DAOS is envisioned as a multiuser and persistent volume available to all applications. It therefore encompasses a variety of system management capabilities, including distributed authentication and device provisioning.

The *Unify* project, the successor to BurstFS [18], implements a temporary high-performance file system using local resources on nodes in the HPC system. In Unify, data is explicitly staged between the temporary Unify file system and the “permanent” parallel file system. The Unify team is exploring specialization in the form of multiple flavors of file systems, such as *UnifyCR* for checkpoint/restart workloads and a separate specialized version for machine learning workloads. This backend specialization allows Unify to optimize for different use cases without sacrificing the portability and common toolset advantages of a POSIX interface. UnifyCR, for example, uses user-space I/O interception, scalable metadata indexing, and colocated I/O delegation to optimize bursty checkpoint workloads while still presenting a traditional file system view of the data.

*GekkoFS* [19] implements a temporary and highly scalable file system providing relaxed POSIX semantics tailored to the majority of HPC applications. This type of specialization allows applications using the existing POSIX interface (under specific constraints) to see dramatic performance improvements as compared with file systems supporting the complete specification. The GekkoFS team has demonstrated millions of metadata operations per second, allowing it to serve applications with access patterns that were historically poor matches for file systems, and the team has shown rapid service instantiation times allowing new GekkoFS volumes to be started on a per-job basis.

*Proactive Data Containers* (PDC) [20] provides a data model in which a container holds a collection of objects that may reside at different levels of a potentially complex storage hierarchy and migrate between them. A PDC volume is instantiated for an application workflow and sized to meet workflow requirements for data storage and I/O. Objects can hold both streams of bytes and KV pairs, and additional metadata can be associated with objects as well. Unlike GekkoFS and UnifyCR, PDC does not present a conventional file system interface but instead provides a way of unifying application’s memory and storage by providing object mapping semantics, which hide actual I/O transfers between storage hierarchies from the user.

## 6 Conclusion

To support data services at scale, a re-usable RPC component must be able to provide performance by enabling the use of all the underlying hardware and network fabrics, flexibility by facilitating service composition, and resilience by providing support for local cancelation. Mercury in that regard is already providing this functionality and is on the path of being used on production systems, to enable not only file system capabilities but to also provide specialized data service workflows as part of the Mochi suite of components.

We are also considering how to make use of collectives through Mercury and how to provide data services with optimized collective RPC operations (such as RPC broadcasts) that do not only rely on point-to-point messaging, which is a limitation when an RPC must be sent to a large number of targets. Furthermore, with accelerators (e.g., GPUs) that are now part of the HPC ecosystem, there is a growing interest in how to make

efficient use of RDMA and address the accelerator’s memory directly from a remote target. These are two future directions that we are considering to further evolve our RPC framework.

## Acknowledgments

This material was based upon work supported in part by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program, under Contract No. DE-AC02-06CH11357; in part supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration. responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative; and in part supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program.

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

The authors would like to thank Howard Pritchard for his help on successfully porting this software to Cray GNI-based systems.

## References

- [1] Argonne National Laboratory, “MPICH,” 2013. [Online]. Available: <http://www.mpich.org>
- [2] J. Zounmevo, D. Kimpe, R. Ross, and A. Afsahi, “On the Use of MPI in High-Performance Computing Services,” in *Recent Advances in the Message Passing Interface*, 2013.
- [3] Google Inc, “Protocol Buffers,” 2012. [Online]. Available: <https://developers.google.com/protocol-buffers>
- [4] M. Slee, A. Agarwal, and M. Kwiatkowski, “Thrift: Scalable Cross-Language Services Implementation,” Facebook, 156 University Ave, Palo Alto, CA, Tech. Rep., 2007.
- [5] F. Wang, S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang, “Understanding Lustre Filesystem Internals,” Oak Ridge National Lab., National Center for Computational Sciences, Tech. Rep., 2009, ORNL/TM-2009/117.
- [6] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, “Scalable Performance of the Panasas Parallel File System,” in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, ser. FAST’08. USA: USENIX Association, 2008.
- [7] N. Ali, P. Carns, K. Iskra, D. Kimpe, S. Lang, R. Latham, R. Ross, L. Ward, and P. Sadayappan, “Scalable I/O Forwarding Framework for High-Performance Computing Systems,” in *IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–10.
- [8] R. Sandberg, D. Golgberg, S. Kleiman, D. Walsh, and B. Lyon, *Design and Implementation of the Sun Network Filesystem*. USA: Artech House, Inc., 1988, pp. 379–390.
- [9] J. Soumagne, D. Kimpe, J. Zounmevo, M. Chaarawi, Q. Koziol, A. Afsahi, and R. Ross, “Mercury: Enabling Remote Procedure Call for High-Performance Computing,” in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2013, pp. 1–8.
- [10] R. B. Ross, G. Amvrosiadis, P. Carns, C. D. Cranor, M. Dorier, K. Harms, G. Ganger, G. Gibson, S. K. Gutierrez, R. Latham, B. Robey, D. Robinson, B. Settlemyer, G. Shipman, S. Snyder, J. Soumagne, and Q. Zheng, “Mochi: Composing Data Services for High-Performance Computing Environments,” *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 121–144, Jan 2020. [Online]. Available: <https://doi.org/10.1007/s11390-020-9802-0>
- [11] C. Docan, M. Parashar, and S. Klasky, “DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows,” *Cluster Computing*, vol. 15, no. 2, pp. 163–181, Jun. 2012. [Online]. Available: <https://doi.org/10.1007/s10586-011-0162-y>

- [12] —, “Enabling High-speed Asynchronous Data Extraction and Transfer Using DART,” *Concurr. Comput. : Pract. Exper.*, vol. 22, no. 9, pp. 1181–1204, 2010.
- [13] C. Wood, S. Sane, D. Ellsworth, A. Gimenez, K. Huck, T. Gamblin, and A. Malony, ““a scalable observation system for introspection and in situ analytics”,” in *Proceedings of the 5th Workshop on Extreme-Scale Programming Tools*, ser. ESPT ’16. IEEE Press, 2016, pp. 42–49.
- [14] C. Ulmer, S. Mukherjee, G. Templet, S. Levy, J. Lofstead, P. Widener, T. Kordenbrock, and M. Lawson, ““faodel: Data management for next-generation application workflows”,” in *Proceedings of the 9th Workshop on Scientific Cloud Computing*, ser. ScienceCloud’18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: <https://doi.org/10.1145/3217880.3217888>
- [15] J. Lofstead, R. Oldfield, T. Kordenbrock, and C. Reiss, “Extending Scalability of Collective IO through Nessie and Staging,” in *Proceedings of the Sixth Workshop on Parallel Data Storage*. New York, NY, USA: ACM, 2011, pp. 7–12.
- [16] P. Grun, S. Hefty, S. Sur, D. Goodell, R. D. Russell, H. Pritchard, and J. M. Squyres, “A Brief Introduction to the OpenFabrics Interfaces - A New Network API for Maximizing High Performance Application Efficiency,” in *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, Aug 2015, pp. 34–39.
- [17] Intel Corporation, “DAOS: Revolutionizing High-Performance Storage with Intel Optane Technology,” <https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/high-performance-storage-brief.pdf>, Jun. 2019.
- [18] T. Wang, K. Mohror, A. Moody, K. Sato, and W. Yu, “An Ephemeral Burst-Buffer File System for Scientific Applications,” in *SC ’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2016, pp. 807–818.
- [19] M. Vef, N. Moti, T. Süß, T. Tocci, R. Nou, A. Miranda, T. Cortes, and A. Brinkmann, “GekkoFS—A Temporary Distributed File System for HPC Applications,” in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2018, pp. 319–324.
- [20] J. Mu, J. Soumagne, H. Tang, S. Byna, Q. Koziol, and R. Warren, “A Transparent Server-Managed Object Storage System for HPC,” in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2018, pp. 477–481.

# Extending the Publish/Subscribe Abstraction for High-Performance I/O and Data Management at Extreme Scale

Jeremy Logan<sup>†</sup>, Mark Ainsworth<sup>§</sup>, Chuck Atkins<sup>‡</sup>, Jieyang Chen<sup>†</sup>, Jong Choi<sup>†</sup>, Junmin Gu<sup>★</sup>, James Kress<sup>†</sup>, Greg Eisenhauer<sup>⊙</sup>, Berk Geveci<sup>‡</sup>, William Godoy<sup>†</sup>, Mark Kim<sup>†</sup>, Tahsin Kurc<sup>†</sup>, Qing Liu<sup>⊃</sup>, Kshitij Mehta<sup>†</sup>, George Ostrouchov<sup>†</sup>, Norbert Podhorzski<sup>†</sup>, David Pugmire<sup>†</sup>, Eric Suchyta<sup>†</sup>, Nicolas Thompson<sup>†</sup>, Ozan Tugluk<sup>§</sup>, Lipeng Wan<sup>†</sup>, Ruonan Wang<sup>†</sup>, Ben Whitney<sup>†</sup>, Matthew Wolf<sup>†</sup>, Kesheng Wu<sup>★</sup>, and Scott Klasky<sup>†</sup>

<sup>§</sup> Brown University, Providence, RI, USA

<sup>⊙</sup> Georgia Institute of Technology, Atlanta, GA, USA

<sup>‡</sup> Kitware, Inc., Clifton Park, NY, USA

<sup>★</sup> Lawrence Berkeley National Laboratory, Berkeley, CA, USA

<sup>⊃</sup> New Jersey Institute of Technology, Newark, New Jersey, USA

<sup>†</sup> Oak Ridge National Laboratory, Oak Ridge, TN, USA

## Abstract

*The Adaptable I/O System (ADIOS) represents the culmination of substantial investment in Scientific Data Management, and it has demonstrated success for several important extreme-scale science cases. However, looking towards the exascale and beyond, we see the development of yet more stringent data management requirements that require new abstractions. Therefore, there is an opportunity to attempt to connect the traditional realms of HPC I/O optimization with the Database / Data Management community. In this paper, we offer some specific examples from our ongoing work in managing data structures, services, and performance at the extreme scale for scientific computing. Using the publish/subscribe model afforded by ADIOS, we demonstrate a set of services that connect data format, metadata, queries, data reduction, and high-performance delivery. The resulting publish/subscribe framework facilitates connection to on-line workflow systems to enable the dynamic capabilities that will be required for exascale science.*

---

Copyright 2020 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

---

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

# 1 Introduction

While exascale systems are poised to arrive at Leadership Computing Facilities and provide never-before-seen capacity for sheer computational ability, a new generation of science applications are evolving that will leverage these systems to push the boundaries of science. Changes in the way that science applications are built and used are creating new challenges to the Applied Math and Computer Science communities which must be met in order to fully utilize these new HPC systems. There are several important categories of changes that are occurring in this context. First, as scientists seek to extend the impact of simulation, there is an increased need to couple together multiple separate applications. One variant of this is multi-physics code coupling, where several established simulation codes are made to share data in order to work together to accomplish a larger task. This type of coupling, known as *strong coupling*, typically involves applications that are interdependent and rely on data from each other in order to continue. This contrasts with *weak (in situ) coupling*, where data producers do not have any dependence on consumers and can continue working unfettered regardless of the progress or presence of subsequent workflow components. Weak coupling capabilities are increasingly in demand as a technique for applying machine learning/AI techniques to understand the data as it is being produced. Another increasingly prevalent technique, seen in areas such as molecular dynamics and seismology, involves the use of large ensembles of individual simulations, along with the subsequent statistical analysis of the large set of resulting data. Finally, federating large-scale experimental and observational facilities with exascale simulation environments to provide real-time analysis and steering of ongoing measurements will be essential for leveraging these facilities to best benefit science. The need to extend the existing high performance I/O and data management capabilities to support these new categories is currently a major driver of our work.

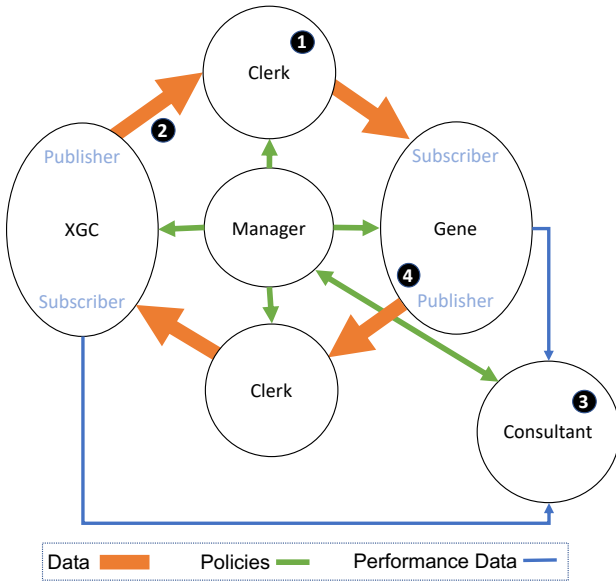


Figure 1: A logical view of a code coupling example involving two fusion codes, XGC1 and Gene, which focus on different regions of a reactor. Enhanced capabilities include 1) Data compression within a publish/subscribe stream, 2) Strong coupling provided by a pair of pub/sub streams, 3) Performance understanding provided by a Consultant capable of advising a Manager, and 4) Enhanced metadata.

The Adaptable I/O System (ADIOS) [12] has been developed to offer high-performance I/O, staging, data reduction, and data management capabilities to applications through a simple publish/subscribe oriented API. In §2 we present several of the underlying models upon which our tools are based. First, we explore some useful extensions to the publish/subscribe pattern, and describe how ADIOS is being used to support such an approach to data management. Then we introduce the Streaming Structure of Parallel Arrays (SSOPA) model that describes the underlying approach used by ADIOS to manage distributed self-describing data.

To address the changing set of usage patterns on leadership computing platforms, we present a set of capabilities that are needed to facilitate effective use of these machines by current and emerging science use cases, and in particular, how we are incorporating these capabilities into the ADIOS ecosystem. First, as I/O and storage capacity is generally failing to keep pace with advances in computational capabilities, data compression and reduction capabilities are an increasingly critical requirement for exascale. We have incorporated a variety of compression capabilities into ADIOS, and ongoing work with MGARD [1] offers a novel data refactoring capability that will allow different levels of data reso-

lution to be provided according to the user's intentions. Next, we have developed a variety of data staging

services to address different runtime needs of applications and *in situ* workflow components. Runtime performance understanding is critical for maintaining high utilization levels for exascale workflows, and we discuss our ongoing efforts to incorporate performance monitoring and analysis into ADIOS-based workflows. Finally, metadata organization for large, complex data streams must be designed to avoid management bottlenecks and poor metadata scaling, so we have concentrated some effort on metadata optimization, including recent refinements to the ADIOS-BP file format. Our work towards each of these capability classes is discussed in more detail in §4.

## 2 Building an Exascale Data Management Interface

Key to our redesign for ADIOS was the recognition that it already bore a great deal in common with publish/subscribe middleware interfaces [4]. Building from this, we proposed [9] an extended publish/subscribe metaphor that included other important functions like management, inline editing and data fusion, and resource utilization planning, in addition to the more common ideas of brokers and content managers. An example of this can be seen in Figure 1. Although ADIOS is generally billed as an I/O library, and it uses a very I/O-like interface, the key difference is in the semantics of access, and how the query metadata is maintained through the system. By breaking the POSIX I/O expectations of byte placement, users’ code be written such that it is indistinguishable if data is delivered from a live stream of data, a persistent store, or some federated view over distributed entities.

The key to understanding this is the core ADIOS data model, which we describe as the Streaming Structure of Parallel Arrays (SSOPA). The base query model that SSOPA offers is very limited – one can identify a range of values that one wants from a global array, a range of values from a local array on a particular writing process, a global variable value, or a variable that had a different value on each writing process. For each of these “local” values, ADIOS presents them as “courtesy” global arrays, where one dimension is simply the index of the writing process.

This extension of the Structure of Arrays (SOA) approach to data, but using the distributed/parallel concept of arrays, is what gives us the key reading abstraction for scientific data. The interface forces each write of data from an individual component to tag its patch with the offset and extent of the local patch within the global array, which allows us to offer a best-of-both-worlds view to the subscribers – a Structure-of-Arrays model, where you can explicitly request a view on the arrays using simple offsets, and yet performance similar to array of structures, since each writer’s

block of data is maintained as an individual column store for efficient access. Figure 2 is a schematic view of how the data looks to the publisher and the subscriber, each of which is individually a parallel code.

Concretely, this allows the interface to respect the fact that these extremely large data streams are composed of simultaneously delivered, distributed data shards. Naïvely, one could think of a large array of processes, each writing out a small patch of a large, distributed matrix. Performance in writing comes from the fact that we can make independent progress; the performance in reading the data leverages many of the same observations of performance of column stores vs row stores in database shards.

The last key to the model is that the stream of parallel arrays has explicit synchronization windows, or

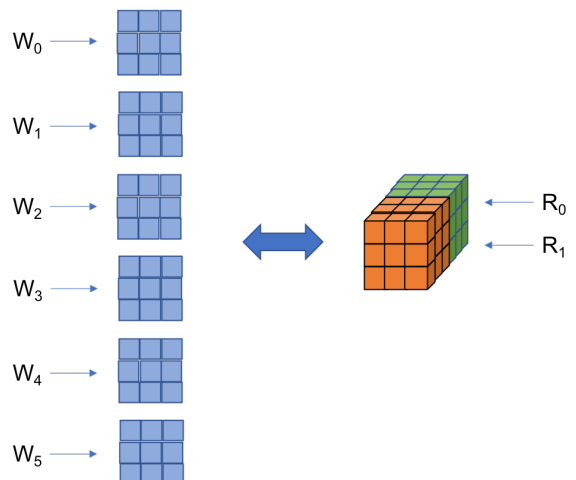


Figure 2: Local 2D arrays written by individual processes ( $W_0 - W_5$ ) appear to readers ( $R_0 - R_1$ ) as part of a 3D array where one array dimension represents writer id.

“steps” in ADIOS terminology. The sequence of these steps can be applied by a user in a number of ways – as versioned partial updates to a table, as separators between distinct items like image data, or as the sequence of time step data from a simulation or experimental science campaign. The last of these is most common for our extreme-scale science community, but all of these engineering choices are supported through the abstraction. This explicit consistency marker by the publishers of the data give the runtime system flexibility for making performance decisions; different engines (as described in §4.2) within ADIOS can implement lazy or imperative consistency as best fits their particular use case.

All together, the SSOPA model represents a base layer for the query model that has broad consensus among our user base. On-going work explores extensions of the query model and the ADIOS API in a number of ways. Staying true to the requirement for adaptability in the ADIOS interface means that we must always be evaluating new ways to tune and extend the subscription interface. However, the logic of the base model and matching challenging user requirements always guides our development process. The modular execution approaches enabled by the SSOPA model help address the needs from across the developing exascale science frontier.

### 3 Near Exascale Science Use Cases

Scientific computing has ridden along on the swell of interest in Big Data, and it is useful to use the lexicon of the 5 V’s (Volume, Velocity, Variety, Veracity, and Value), as it helps to highlight where some of the key differences are that have caused divergence in the solutions that different communities use. Although exascale computing does promise large volumes of data, that data is not like the data that drives internet-scale development. Individual data elements are themselves extremely large (multiple TBs in size), and they arrive as part of tightly synchronized, high velocity bursts. We have identified three examples of science teams that are preparing for the exascale era – some in terms of exaflop computations, and others in terms of exabyte data – that can help put the capabilities we discuss in §4 into context, as well as further clarifying how ADIOS’s data model addresses their core data management needs.

#### 3.1 Computational Fusion

XGC is advanced fusion software designed to model plasma edge physics in magnetic confinement devices and one of the largest scientific simulations running at Department of Energy Leadership Computing Facilities (LCFs). XGC has demonstrated full-machine scalability at the largest available sites, with jobs that result in multiple petabytes of data per hour over tens of hours durations. Such data generation already challenges the capacities of the file systems of current LCFs, and next generation machines will beckon even larger datasets. At these scales, it is no longer feasible to store all relevant datasets and carry out post-simulation analysis only; the time to retrieve the quantity of data from disk/tape is too great. Output from an XGC simulation must be processed *in situ* and in-transit through complex workflows for meaningful data reduction (before being written to storage) and for scientific data analysis and visualization.

Another I/O challenge facing fusion simulations is code coupling for Whole Device Modeling (WDM). By running multiple application codes concurrently, each focusing on a specific spatial region or physical phenomenon of the fusion device, researchers are building a holistic model to understand complex plasma reactions throughout the entire reactor. Execution of multiple fusion simulation codes with both tightly and loosely coupled data sharing during execution is needed. ADIOS provides performant data exchange methods to satisfy the various requirements.

Various I/O research leveraging ADIOS has been conducted to support data exchanges for loose and tight coupling scenarios, data reduction with strictly bounded physics properties, and efficient metadata management with high concurrency. Runtime performance research is necessary to insure that the coupled simulation is



resilient to jitter caused by external network contention and unexpected system abnormality, and to provide online guidance through low-latency performance monitoring and prediction.

### 3.2 Molecular Dynamics with LAMMPS

Future molecular dynamics/materials science computing addresses modeling on longer time scales; in particular, understanding how we can design and manipulate materials so that their long-term behavior meets our needs and requirements. Using the LAMMPS molecular dynamics code [13], researchers have begun studying problems such as the following: What alloy mix is most likely to maintain integrity over two decades of radiation bombardment in a fusion reactor? What mix of elements will best ensure that radioactive elements are fully contained by a glass matrix protection from the outside world and the human population?

In general, improved flexibility, durability, and hardness over time are all achievable through better design and manufacturing; however, traditionally the simulations do not explore the breadth of possibility, focusing on a more limited range of the solution design space. While simulations with millions, billions, and even trillions of atoms are now possible, particle number scaling is not the only difficulty – long-time runs combine scalable performance with the management and control of concurrently-running ensembles of such runs.

The I/O challenge for such scenarios is multi-faceted. At a base layer, one needs to accelerate the performance of each individual parallel run as much as possible. On another level, one must collect, calibrate, and make decisions upon the sets of such time-stamped data in order to direct exploration of the evolution of the system. For that to be efficient, in-memory streaming of data is needed for runtime coupling, even if the development, testing, and validation needs those same interfaces to work through file I/O. Finally, with the extent of the number of ensemble members, it is critical to manage the metadata representations so that one can not only query based on computational provenance (i.e. Run #23478) but also on location (i.e. on the burst buffer) and on relevant features (i.e. after the crystal cracked).

### 3.3 Radio Astronomy with The Square Kilometre Array

Radio astronomy data is another major use case of our I/O abstraction model. In particular, we have been focusing on data I/O challenges of the future world’s largest radio telescope, the Square Kilometre Array (SKA). SKA is an international project that will offer unprecedented views to the astronomy community. From a data perspective, it is an intimidating challenge – in the initial phase, the output from more than 130,000 antennas located in Western Australia will be streamed, condensed, analyzed, and then fed into an open science repository. In the initial phase, this will be at 10 Tbps, with over an order of magnitude increase in bandwidth requirement expected as the SKA platform fully comes on line [19].

One of the most difficult I/O challenges for the radio astronomy community is storing visibility data. It is generated at an early stage in the typical data reduction pipeline, and it is usually the largest data product across the entire workflow. For the past two decades, most such data has been stored in the MeasurementSet format through the Casacore Table Data System (CTDS) [15]. CTDS was designed under the assumption that most radio astronomy data processing algorithms can be embarrassingly parallelized, and it does not use parallel I/O.

CTDS has been sufficient for most radio telescopes prior to SKA. However, SKA will have orders of magnitude more antennas than previous radio telescopes, and the volume of visibility data scales quadratically with the number of antennas. In order to better understand the potential bottlenecks, we have simulated the full scale SKA Phase 1 Low data using most compute nodes of the world’s fastest supercomputer, Summit [19]. The results showed that the embarrassing parallel model is no longer optimal when scaling up to the full-scale SKA level, possibly explainable by producing too many small table files, which bottleneck the parallel filesystem.

We have developed an ADIOS storage manager for CTDS, which enables parallel I/O for CTDS at the column layer, and repeated the SKA workflow using it [18]. The performance is at least one order of magnitude better than the embarrassing parallel model even at the partial SKA Phase 1 scale. However, due to limitations

of the CTDS architecture, we have yet to enable parallel I/O at the table layer, which would likely improve performance even further. Doing so will also require optimizations at the metadata management layer that we will discuss in §4, in order to handle table-based data more efficiently.

### 3.4 Lessons Learned

The science use cases described above highlight several important challenges for exascale computing. The extreme data sizes being handled by the Computational Fusion and the Radio Astronomy examples will require new reduction and compression capabilities, which we discuss in §4.1. Both the Fusion and Molecular Dynamics cases call for robust coupling mechanisms for a variety of coupling situations, as we describe in §4.2. All three science examples point to the need for integrated tools for runtime performance understanding, a capability that we examine in §4.3. Finally, the complexity of data in the Molecular Dynamics case and the Radio Astronomy case both point to the need for specific metadata management capabilities, which we detail in §4.4.

## 4 Capabilities for Exascale

There are many specific technical approaches and results that have been developed in response to these science needs. What we summarize here are the broader core capabilities that we have identified as being required to address the needs of these science applications in the exascale era. For each of these capabilities, we have tried to highlight specific papers and projects that one can explore for greater technical detail.

### 4.1 Reduction and Compression Capabilities

*The purpose of computing is insight,  
not numbers.*  
—Richard Hamming

Richard Hamming’s remark [8], made over 50 years ago at the dawning of the age of large scale scientific computation, is even more relevant today as we prepare for scientific simulation at exascale. Avoiding bottlenecks in exascale scientific discovery requires research into managing, storing, and re-

trieving the large volumes of data that are produced by simulations and analyzed for months afterwards. Our main objective is to accelerate knowledge discovery, and as data sizes grow from simulations and experiments, it is imperative that we prioritize information over data.

Some of the fundamental research we have done aims to manage the overall data life cycle, including data generation (e.g., from a simulation) or acquisition (e.g., in the case of experimental and observational data), optimized data placement, runtime data management including migration, reorganization and reduction, data consumption for knowledge discovery, and purging data from the system to optimize system operation.

The classical workflow where the entire dataset is written to storage for later analysis will no longer be viable at exascale, simply because the amount of generated data will be too large due to capacity and performance limitations. In the future, it will be vital to take advantage of *a priori* user information (1) to gain higher performance and predictability of I/O, (2) to prioritize the most useful data for end users so that I/O can be finished under time constraints, and (3) to perform *in situ* operations and analysis before storing the information. A result of these requirements is a need for a set of techniques to reduce and restructure data, here referred to as Data Refactoring. In recent years, libraries such as SZ[5], ZFP[11], MGARD[1] have emerged as leading techniques for compressing and reducing large, voluminous data at extreme scales. The research using MGARD is an important part of the response to this need for data refactoring. It is a progressive compression technique, based on the theory of multigrids, that allows one to segment data into components that can be individually managed and yet also can bound the error and timeliness of your read request by only pulling the number of components from the multigrid layers that are needed for the accuracy at hand.

The challenge when applying refactoring techniques, particularly application-aware techniques, is how to incorporate sufficient knowledge in the storage system such that an arbitrary future client has sufficient information to recreate the desired information. Additionally, by spreading data across multiple different kinds of storage media that typically have independent namespaces, locating any particular data will be challenging.

*A priori* information can be provided by application scientists regarding which data should be sent where in the storage system (so that minimally, the most science-relevant data can be available for subsequent analysis. This can allow science goals to be accomplished even when the storage is busy servicing other users.

For our approach, data refactoring generates the needed prioritization classes. There are many data refactoring techniques, including re-organization and reductions, and the best choice will generally be application- and user-dependent. However, our observation is that, once the choice is settled, it will not typically change from run to run within an extended campaign. One research challenge in effectively and efficiently refactoring data is understanding when the time and resources required to identify and execute the “best methods” exceeds the gains achieved. Another critical question concerns quantifying and controlling information loss due to refactoring the data and using a reduced dataset. Broadly speaking, the path from data to knowledge consists of extracting underlying models or patterns from the datasets and interpreting the resulting models. Although scientific data generally contains random components due to finite precision and measurement and calibration effects, useful scientific data is never purely random. As such, a core concern in refactoring is understanding how much information is present in a data set and therefore which type(s) of refactoring will be most effective.

Ideally, scientists would like to perform the entire analysis *in situ*, thus avoiding intermediate data sets and effectively circumventing the large data issue completely. The catch, of course, is that this is unlikely to achieve the best science results since, by their nature, large-scale simulations aim to discover new and emergent behavior often hidden in the form of higher order effects within the data deluge. In particular, this means if data thinning or truncation is applied haphazardly, the higher order information sources may be eliminated. Typically, entire data sets cannot be stored in easily accessible storage due to its sheer size. However, the data cannot be reduced prior to archiving without risking losing information. Viewed in this way, the problem would appear intractable. As noted above, though, this is not a true impediment as long as we can incorporate a user’s *a priori* knowledge of models and effective refactoring techniques. The information needed to answer the scientist’s particular science goals is frequently significantly smaller, and using careful information-theoretic and application-given techniques the Storage System and I/O layer can exploit this. *In situ* data management and reduction pipelines comprise of multiple steps: applying reduction techniques, assessing the quality of reduction, analyzing data to ensure preservation of essential features, and assessing the overall performance of the full pipeline.

Deep application knowledge means one can sometimes achieve dramatically superior data reduction compared with what one might achieve otherwise. However, even in the absence of such high level knowledge, an I/O system must offer generic data reduction and re-organization techniques. For instance, certain basic data semantics information is needed and must be supported by the overall infrastructure. Effective *in situ* data management is a vital component of overall large data management at the exascale.

#### Key research questions addressed by ADIOS with MGARD

1. How can we initially place data so that it can be discovered and consumed efficiently?
2. How can the placement and migration of data across a multi-tiered storage hierarchy be optimized at runtime, both from the application and system perspective?
3. How can knowledge about the application used to better prepare the data for consumption?
4. When and how do we make the decision to purge data?

Staging Engine	Characteristics	Application Domain
SST	Configurable queueing, dynamic connections, multiple readers, WAN and RDMA	General use
SSC	MPI-based coupling	Optimized for tightly coupled simulation codes utilizing MPI.
BP4	File-based coupling, readers update as file is written	Useful for development and temporal analysis
Inline	In-process coupling, zero copy data delivery	Requires specially written applications, only $N \rightarrow N$ coupling

Table 1: ADIOS2 Staging Engines

## 4.2 Coupling Capabilities

Another way to understand ADIOS’s publish/subscribe view[4, 2] is to recognize that from the perspective of data producers and consumers, everything is coupled via these publish/subscribe channels. It is the user’s choice at runtime to choose a service provider, “engine”, that dictates whether this coupling happens asynchronously through files, synchronously through memory, or in a more complicated pattern. Table 1 showcases some of the available engines as of ADIOS 2.5. Though this simple description of functionality belies a host of complexity, the basic functionality of managing the data exchange defined by the SSOPA model is shared by all engines.

While all of the science cases above share a need for direct communication between running HPC programs, they do differ in the details of their needs and the nature of the data exchange required. For example Whole Device Modeling requires bi-directional coupling between components and the simulation for timestep  $N + 1$  in any component application cannot proceed without the data produced by timestep  $N$  in other coupled applications. We refer to this level of interdependency as “strong coupling” and in this case the staging system often has very little flexibility in meeting the application communication needs, as any latency or bandwidth limitations have a direct impact on the performance of the composite application. In contrast, other coupling situations are “weak” and allow the communication system more flexibility. In LAMMPS for example, the simulation and the analytics form a simple pipeline, and while the performance of that pipeline is important, there are also opportunities for techniques such as queueing and latency hiding that can enhance the overall performance of the system.

To support the strongest coupling cases, where all participating components can share a global MPI communicator via MPI’s MPMD launch mode and the application communication pattern is fixed, ADIOS features the Strong Staging Coupler (SSC) engine. In the first timestep, SSC records the geometry of the SSOPA exchange between the coupled components (what array elements are written and read where), and on subsequent timesteps that data exchange is re-enacted using one-sided MPI put and get calls. This fixed pattern of data exchange is not a universal feature of HPC applications, but it does appear in important subsets, such as XGC, and relying upon it allows SSC to exploit highly-optimized MPI implementations on emerging exascale computing platforms.

In situations where the coupled components cannot meet the criteria necessary to use SSC, ADIOS users can fall back to another staging engine, the Sustainable Staging Transport (SST). Unlike SSC, SST uses MPI only within each application, so it doesn’t require all components to be launched simultaneously with MPMD mode and uses one of several non-MPI transports to move data between applications. Currently those transports include a LibFabric-based RDMA transport for intra-cluster use, TCP- or UDP-based transports for inter-cluster or WAN communication and an experimental shared-memory transport. In support of the pub/sub model described in this paper, SST supports dynamic connection and disconnection of Reader clients, including multiple simultaneous readers. This is an important feature for visualization clients that may wish to connect and disconnect from a running analysis application, but it also has a variety of other uses, including making sure that something like the failure of a connected analysis application doesn’t interfere with an ongoing simulation. Also unlike SSC, SST doesn’t assume lockstep synchronization between readers and writers, instead it buffers timestep data

until it is required, transmits the data to readers when requested, and manages the release of buffered data after it is consumed. Because unbridled queuing can lead to memory exhaustion and some applications have little memory to spare for staging, SST also has mechanisms for limiting queue sizes and allows applications to either block or discard data on queue full conditions. Like SSC, SST can also take advantage of fixed communication patterns for the purpose of pre-loading data to the readers where it will be consumed.

While the basic design of ADIOS coupling has been flexible enough to support high performance publish/subscribe operation in today's critical applications, we anticipate changes to support the needs of emerging applications and systems. Some of these changes, like managing performance and placement as data moves through more complex storage hierarchies, likely fit within current ADIOS APIs and semantics. However, additional techniques borrowed from pub/sub systems, such as the writer-side filtering and data reduction offered by derived events[6], may require reimagining the nature of ADIOS read semantics.

### 4.3 Performance Understanding

Exascale computing will bring us unprecedented computing performance through massive new machines involving both a vast collection of CPU cores as well as cutting edge GPUs, all connected through extremely fast and flexible networks and leveraging a range of new memory and storage technologies. With well tuned codes, these machines will be capable of exascale performance, but balancing the CPU / GPU mix, and avoiding network misuse and storage bottlenecks will be more important than ever. In addition to the raw scale of the machinery, the science cases described in §3 demonstrate a range of behaviors that drive new performance needs which are difficult to meet with conventional software technologies. In particular, we have seen a change from the construction of single, monolithic codes to computational experiments composed of multiple executables, run as ensembles, coupled codes, or as pipelines of *in situ* computations. We also observe increasing need for federated computing where, for instance, large data sources such as accelerators and radiotelescopes are used to feed extreme scale computing resources that are geographically separated from the data sources. This will require that these application be tuned not just within a single supercomputer, but with tuning strategies that span multiple sites and platforms.

The tools that have been built up over the last decades for performance understanding and management of large MPI-based codes are capable primarily of static, post hoc analysis of application performance. But this style of analysis will not be sufficient for application workflows with many moving parts and requiring careful orchestration of those parts to achieve adequate performance on complex hardware platforms. Runtime availability, aggregation, and analysis of performance information will be critical for such orchestration, as will the ability to accurately model [17] and predict the performance of application components, and place processes appropriately [3]. Systems will need to address many additional constraints and concerns as a result of dealing with the measurement of collections of distinct runtimes and distinct applications, and it becomes more than what a simple patchwork of fixes can address.

Existing tracing and profiling tools and frameworks such as Scalasca [7], ScoreP [10] and Tau [14] tend to be batch oriented, with a focus on post-mortem analysis, and typically operate on a single process or MPI application. Existing runtime techniques do not offer detailed aggregate views of the performance of a suite of concurrent applications. As a result, it is difficult to get the holistic view needed at runtime for coupled codes or ensemble suites using these types of tools.

As we examine the needs of exascale performance understanding, it is clear that a variety of performance metadata from tracing and profiling must be made available at runtime. As some of these measurements can be expensive, we must have the ability to turn individual measurements on and off as needed, and to adjust frequency of measurements, and delivery criteria. At first glance, it would appear that the pub/sub tools might directly support the additional performance metadata through the same mechanisms that handle application data, however the delivery requirements of performance metadata may be quite different from those of application data, not only in terms of destination, but also delivery frequency and latency. So care must be taken not

to carelessly mix data and performance metadata, but rather to allow out-of-band approaches for expediting performance metadata when necessary. Such tools must provide appropriate flexibility while remaining user friendly and avoiding the introduction of network bottlenecks.

In [20], we presented a prototype system for collecting and aggregating performance data. The system leveraged the Tau software for tracing and profiling, and introduced an aggregation layer capable of merging this performance metadata collected across a large number of nodes into a single global view that could be queried by performance analysis components at runtime. This work demonstrated several examples of leveraging runtime performance information, including runtime application monitoring, I/O model extraction, understanding I/O variability, and dynamic process placement. As an example of the use of this sort of monitoring infrastructure, Figure 3 illustrates the variability inherent across and within runs of XGC-1 on both Titan and Theta.

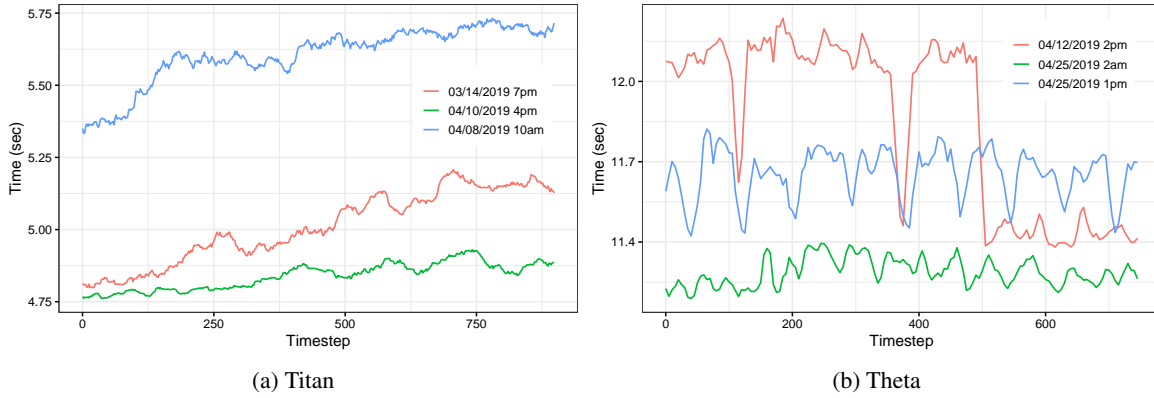


Figure 3: Performance variability of XGC-1. The results presented here for Titan (a) and Theta (b) show the degree of variability in performance both within a timestep and across separate runs.

#### 4.4 Metadata Management Capabilities

In order to make scientific data easy to retrieve, reuse and re-evaluate over its lifetime, plenty of metadata is also generated. For example, a simulation with thousands of MPI processes for millions of steps needs to output data every 100 or 1,000 steps. The output data contains variables that are global arrays and each MPI process is a writer which outputs a specific block of each global array. In this case, we need metadata for each block to track which process generates it and at which simulation step, which variable it belongs to, where it is located in the global array, where to find it in actual file, etc. As we can see, even for writing out a single variable, the amount of metadata needed can be huge. If a simulation outputs hundreds of variables, the number of metadata items will further increase, which not only leads to extra I/O and storage overhead, but also exacerbates the computation and communication overhead since the metadata needs to be gathered and reorganized into certain format. In this subsection, we are going to discuss several specific metadata challenges we face when we design ADIOS. Some of these challenges have been well addressed which results in some unique capabilities enabled by current ADIOS implementation, while the others are planned to be addressed by future work.

First of all, in order to achieve better I/O performance on parallel file systems, scientists often let each MPI process of their simulation output its own data, which is usually a specific data block that is part of a much larger global variable. If the simulation data is generated in this way, it requires I/O libraries such as ADIOS to be able to efficiently manage the metadata for each data block. In ADIOS, this challenge is addressed by leveraging a log-structured file format called BP, which is the ADIOS native binary file format. In BP format, the data generated by each writer are organized and stored in a block by block manner. Specifically, each writer directly writes its own data blocks to the file system, while in the meantime it maintains each data block's metadata, which will be later gathered by a specific MPI process.

This process organizes the metadata items collected from all writers into certain order and then writes them into the metadata file. Moreover, the metadata contains not only the information for locating each data block, but also some characteristics, such as minimum and maximum value, of each data block, which enables the capability of building a fast query interface. Or in other words, users can query the characteristics of each data block without touching the actual data as those characteristics have been included in the metadata.

Second, there are different ways to organize metadata items collected from all writers, the challenge is to not only reduce the communication and computation overhead of gathering and organizing the metadata items, but also provide users some flexibility so that different data access patterns can be supported. In order to reduce the overhead of constructing global metadata, we separate the metadata items of each simulation step from each other and build an extra index table to locate them in a step-by-step manner [16]. This also allows users to access data generated at any simulation step without parsing the metadata of all previous steps, which is especially useful for data streaming use cases, such as online analysis and code coupling. Moreover, since the index table is small and can be easily loaded into memory of each reader, querying metadata of each simulation step is an atomic transaction which enables lock-free data access for asynchronous readers.

As future work, we need to address two critical challenges in metadata management for scientific applications. One is to further reduce overhead of constructing metadata when simulation outputs a plethora of variables with complex attributes. The other one is to support data query across multiple storage tiers. With the development of data storage technology, HPC systems are equipped with complex I/O subsystems that usually consists of multiple storage tiers. For instance, besides center-wide parallel file systems and tape storage systems, all DoE's leadership supercomputers have an intermediate storage tier called burst buffer. Scientific applications might write data to all these storage tiers based on their needs. Therefore, it is critical to design an efficient metadata mechanism that can track data objects and accelerate data movement across all storage tiers.

## 5 Conclusion

Exascale computing is bringing new usage patterns that will open up new areas of scientific discovery. The traditional monolithic simulation code is being replaced with coupled codes, *in situ* workflows, and large ensembles of independent tasks. In the face of these new usage patterns, bespoke custom workflows are becoming increasingly unwieldy. Monolithic solutions are unlikely to be suitable, as portions of workflows will need to be performed across different resource scales. Systems that provide efficient and reusable support for the new usage patterns are needed.

The publish/subscribe metaphor provides a good starting point for designing such systems to accommodate the I/O and data management needs of the exascale era. ADIOS as an Adaptable I/O System based on the publish/subscribe paradigm has been used to support some of these emerging data needs. New usage patterns will require that new capabilities be added to existing pub/sub mechanisms. Compression and reduction capabilities, like MGARD, will be required to deal with data volumes that continue to outpace memory and storage capacity of new systems. Efficient coupling capabilities are needed to support new paradigms of application coupling and *in situ* composition. Extensions to performance monitoring capabilities will enable the advanced monitoring and tuning that will be needed to keep exascale machines fully utilized, and careful adjustments to metadata management systems are needed to insure that application access patterns can be supported efficiently. As new exascale systems come online, ADIOS will continue to adapt to the changing landscape of high performance computing in order to meet new application needs.

**Acknowledgment** Without the continued support from the Department of Energy's Office of Advanced Scientific Computing Research, the projects upon which this future vision rests would not be possible. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725, as well as resources of the National Energy Research

Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

## References

- [1] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science*, 19(5-6):65–76, 2018.
- [2] J. Y. Choi, C. Chang, J. Dominski, S. Klasky, et al. Coupling exascale multiphysics applications: Methods and lessons learned. In *14th IEEE International Conference on e-Science, e-Science 2018, Amsterdam, The Netherlands*, pages 442–452. IEEE Computer Society, 2018.
- [3] J. Y. Choi, J. Logan, M. Wolf, G. Ostrouchov, T. Kurc, Q. Liu, N. Podhorszki, S. Klasky, M. Romanus, Q. Sun, M. Parashar, R. M. Churchill, and C. S. Chang. TGE: Machine learning based task graph embedding for large-scale topology mapping. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 587–591, Sep. 2017.
- [4] J. Dayal, D. Bratcher, G. Eisenhauer, K. Schwan, M. Wolf, X. Zhang, H. Abbasi, S. Klasky, and N. Podhorszki. Flexpath: Type-based publish/subscribe system for large-scale science analytics. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 246–255, May 2014.
- [5] S. Di and F. Cappello. Fast error-bounded lossy HPC data compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 730–739, May 2016.
- [6] G. Eisenhauer, K. Schwan, and F. E. Bustamante. Publish-subscribe for high-performance computing. *IEEE Internet Computing*, 10:40–47, 2006.
- [7] M. Geimer, F. Wolf, B. J. Wylie, E. Ábrahám, D. Becker, and B. Mohr. The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, 22(6):702–719, 2010.
- [8] R. W. Hamming. *Numerical Methods for Scientists and Engineers.*, McGraw-Hill, Inc., USA, 1973.
- [9] S. Klasky, M. Wolf, M. Ainsworth, et al. A view from ORNL: scientific data research opportunities in the big data age. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria*, pages 1357–1368, 2018.
- [10] A. Knüpfer, C. Rössel, D. a. Mey, et al. Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, Tau, and Vampir. In *Tools for High Performance Computing 2011*, pages 79–91, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [11] P. Lindstrom. Fixed-rate compressed floating-point arrays. Computer software. <https://www.osti.gov/servlets/purl/1231942>. Vers. 00. USDOE. 30 Mar. 2014. Web.
- [12] Q. Liu, J. Logan, Y. Tian, et al. Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks. *Concurrency and Computation: Practice and Experience*, 26(7):1453–1473, May 2014.
- [13] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Jrnl of Computational Physics*, 117(1):1–19, 1995.
- [14] S. S. Shende and A. D. Malony. The TAU parallel performance system. *The International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.
- [15] G. van Diepen. Casacore Table Data System and its use in the MeasurementSet. *Astronomy and Computing*, 12:174–180, 2015.
- [16] L. Wan, K. V. Mehta, S. A. Klasky, M. Wolf, H. Y. Wang, W. H. Wang, J. C. Li, and Z. Lin. Data management challenges of exascale scientific simulations: A case study with the Gyrokinetic Toroidal Code and ADIOS. In *The 10th International Conference on Computational Methods, ICCM’19*, 2019.
- [17] L. Wan, M. Wolf, F. Wang, J. Y. Choi, G. Ostrouchov, and S. Klasky. Analysis and modeling of the end-to-end I/O performance on OLCF’s Titan supercomputer. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1–9, Dec 2017.
- [18] R. Wang, C. Harris, and A. Wicenec. AdiosStMan: Parallelizing Casacore table data system using Adaptive IO System. *Astronomy and Computing*, 16:146–154, 2016.
- [19] R. Wang, A. Wicenec, and T. An. SKA shakes hands with Summit. *Science Bulletin*, 2019.
- [20] M. Wolf, J. Choi, G. Eisenhauer, S. Ethier, K. Huck, S. Klasky, J. Logan, A. Malony, C. Wood, J. Dominski, and G. Merlo. Scalable performance awareness for in situ scientific applications. In *2019 IEEE 15th International Conference on e-Science (e-Science)*, 2019.



# **$SUQ^2$ : Uncertainty Quantification Queries over Large Spatio-temporal Simulations**

Noel Moreno Lemus <sup>a</sup>, Fabio Porto <sup>a</sup>, Yania M. Souto <sup>a</sup>, Rafael S. Pereira <sup>a</sup>, Ji Liu<sup>c</sup>, Esther Pacciti<sup>b</sup>,  
and Patrick Valduriez <sup>b</sup>

<sup>a</sup>LNCC, DEXL, Petropolis, Brazil

<sup>b</sup>Inria and LIRMM, University of Montpellier, France

<sup>c</sup>Big Data Laboratory, Baidu Research, Beijing, China

## **Abstract**

*The combination of high-performance computing towards Exascale power and numerical techniques enables exploring complex physical phenomena using large-scale spatio-temporal modeling and simulation. The improvements on the fidelity of phenomena simulation require more sophisticated uncertainty quantification analysis, leaving behind measurements restricted to low order statistical moments and moving towards more expressive probability density functions models of uncertainty. In this paper, we consider the problem of answering uncertainty quantification queries over large spatio-temporal simulation results. We propose the  $SUQ^2$  method based on the Generalized Lambda Distribution (GLD) function. GLD fitting is an embarrassingly parallel process that scales linearly to the number of available cores on the number of simulation points. Furthermore, the answer of queries is entirely based on computed GLDs and the corresponding clusters, which enables trading the huge amount of simulation output data by 4 values in the GLD parametrization per simulation point. The methodology presented in this paper becomes an important ingredient in converging simulations improvements to the Exascale computational power.*

## **1 Introduction**

The rapid growth of high-performance computing combined with recent advances in numerical techniques increases the accuracy of numerical simulations. This leads to practical applicability in models for predicting the behavior of weather, hurricane forecasts [1] and subsurface hydrology [2], just to name a few, positioning simulations as increasingly important tools for high-impact predictions and decision-making applications.

In order to reach higher simulation accuracy of reproduced phenomena, the scientific community is leaving behind the traditional deterministic approach, which offers point predictions with no associated uncertainty [3], to move towards Uncertainty Quantification ( $UQ$ ) as a common practice over simulation results analysis. Arguing for improvements in simulation accuracy, by the assessment of uncertainty quantification of simulation results consider the extra knowledge a scientist acquires whenever the simulation behaviours become more

---

Copyright 2020 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

evident over different scenarios. The increase in simulation scenarios call for more computing power. In a subsurface seismic domain, for example, a simulation computes the wave velocity at each point of an area. A scientist is interested in a particular region of the space where a salt dome is located. She may issue a query filtering only that spatial region and checking how precise is the velocity field in that area. In order to achieve that, she could quantify the uncertainty involved in the region of interest. This type of simulation that evaluates the uncertainty by analyzing its output is referred to as *forward propagation*. In this paper, we focus on answering uncertainty quantification queries over large spatio-temporal simulations ( $UQ^2STS$ ). The  $UQ^2STS$  problem is challenging as: (1) it requires analyzing large amounts of simulation output data; (2) the uncertainty at each point may exhibit patterns too complex to be captured by low order statistical moments (such as mean and standard deviation); (3) the uncertainty behavior may vary along a simulation spatio-temporal region leading to a complex data pattern to be modelled and an uncertainty expression of difficult interpretation. Thus, the problem involves conceiving a method to accurately and efficiently solve the  $UQ^2STS$ .

We propose the  $SUQ^2$  method to solve this problem.  $SUQ^2$  is based on the adoption of the generalized lambda distribution (*GLD*) PDF type as a model of the uncertainty at each simulation computation point, which solves issue (2). Its uniform representation reduces significantly the computation of the necessary fitting function. Furthermore, by adopting a single function type, we could run clustering algorithms on the GLD parametrization, electing a representative data distribution for a large set of simulation points. This represents a huge saving in data storage, which solves issue (1). Finally, the cluster representatives are used to composed a mixture of GLDs and to measure the information entropy of the  $UQ^2STS$ , which solves issue (3). We illustrate the adoption of the  $SUQ^2$  method with a case study in seismology.

In our previous work [12], we designed a system to efficiently compute PDFs in large spatio-temporal datasets. The system implements a Spark dataflow to streamline the huge amount of PDF fitting computation. Our work extends the results of this work by adopting the GLDs as a generic model for data distribution, avoiding testing for different distribution types and uniformizing the computation of mixed PDFs in spatial-temporal regions.

To the best of our knowledge, the first effort to use the *GLD* to model uncertainty in data is the work of Lampasi et. al. [8], followed by Movahedi et. al. [9] for a task involving the computation of results reliability. The adoption of mixture of *GLDs* is motivated by the work of Ning et. al. [10]. Algorithms to use the mixture of *GLDs* to model datasets have been deployed with the **GLDEX** R package. Wellmann et al. [11] propose to use information entropy as an objective measure to compare and evaluate model and observational results. Our  $SUQ^2$  method combines these techniques.

The rest of the paper is organized as follows: Section 2 gives the problem formalization and introduces the *GLD* function. Section 3 presents the  $SUQ^2$  method and the workflow to solve the  $UQ^2STS$  problem. Section 4 gives an experimental evaluation with a use case in seismology. Section 5 concludes.

## 2 Preliminaries

In this section we define some basic concepts needed for the rest of the paper. We first formalize the problem. Next, we present the Generalized Lambda Distribution function, including a discussion on its shape and the mixture of GLDs.

A simulation is a combination of a numerical method implementing a mathematical model and a discretization that enables to approximate the solution in points of space-time. A simulation can be used for two different types of problems: *forward* or *inverse*. *Forward problems* study how uncertainty propagates through a mathematical model. In a simulation, a spatio-temporal domain is represented by a grid of positions  $(s_i, t_j) \in \mathcal{S} \times \mathcal{T} \subseteq \mathbb{R}^3 \times \mathbb{R}$ , where values of a quantity of interest (QoI), such as velocity, are computed. In a parameter sweep application, a simulation is executed multiple times, each with a different initial configuration, leading to multiple occurrences for a given domain position, in order to explore the simulation behavior under different scenarios.

A simulation can be formally expressed as  $\mathbf{q} = \mathcal{M}(\boldsymbol{\theta})$  where:  $\boldsymbol{\theta} \in \mathbf{R}^n$  is a vector of input parameters of the model;  $\mathcal{M}$  is a computational model, and  $\mathbf{q} \in \mathbf{R}^k$  is a vector that represents quantities of interest (*QoI*). In a *forward problem*, the parameters  $\boldsymbol{\theta}$  are given and the quantities of interest  $\mathbf{q}$  need to be computed. In stochastic models, at least one parameter is assigned to a probability density function (*PDF*) or it is related to the parameterization of a random variable (*RV*) or field, causing  $\mathbf{q}$  to become a random variable as well.

In order to estimate a stochastic behavior of the output solution  $\mathbf{q}$  in terms of input uncertainties  $\boldsymbol{\theta}$ , sampling methods analyze the values of  $\mathcal{M}(\boldsymbol{\theta})$  at multiple sampled conditions in the  $\Theta$  space (called stochastic space) directly from numerical simulations. Methods like Monte Carlo (*MC*) are used to randomly sample in the stochastic space, and hence many sample calculations are required to achieve a convergence of stochastic estimations. As a result, the method returns multiple realizations of  $\mathbf{q}$ . Then, other methods to measure the uncertainty need to be applied.

In a more general case, the computational model  $\mathbf{q} = \mathcal{M}(\boldsymbol{\theta})$  represents the spatio-temporal evolution of a complex systems, and the *QoI*  $\mathbf{q}$  can be represented as  $\mathbf{Q} = (\mathbf{q}(s_1, t_1), \mathbf{q}(s_2, t_2), \dots, \mathbf{q}(s_n, t_n))$ , where:  $(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n) \in \mathcal{S} \times \mathcal{T} \subseteq \mathbb{R}^3 \times \mathbb{R}$  represents a set of distinct spatio-temporal locations, and  $\mathbf{q}(s_i, t_j)$  represents a value of the *QoI* at the spatio-temporal location  $(s_i, t_j)$ .

In a stochastic problem, on each spatio-temporal location  $(s_i, t_j)$  we have many realizations of  $q(s_i, t_j)$  that can be represented as a vector  $\langle q(s_i, t_j) \rangle$ . In this context, it is frequent that more than  $10^4$  simulations are performed while exploring the model parameter space, which leads the output dataset to have a size of order  $N_s \times N_t \times N_{sim}$ , where  $N_s$  is the number of spatial locations,  $N_t$  is the number of time steps, and  $N_{sim}$  is the number of simulations. An example of the volume of data generated by these simulations is given in the experimental evaluation (see Section 4), where the output dataset is about 2.4 TB.

A simple approach to solve a spatio-temporal UQ query is to consider a simple aggregation query, computing the mean and standard deviation on the selected spatio-temporal region. This approach, albeit being simple and fast, is unable to capture patterns exhibited by the data distribution of complex phenomena. The solution we propose in [12] adopts probability density function (*PDF*) as a more accurate modeling data distribution at each point. However, the adoption of *PDFs* brings its own challenges. First, as the uncertainty may vary in different regions of the simulation, one needs to try multiple function types, such as Gaussian, Logarithm, Exponential, *etc.*, at each spatial position to find the one closest to its data distribution. This leads to a huge computation cost for each simulation spatial position. Moreover, as a region may be defined by different *PDF* types, answering solving a (*UQ<sup>2</sup>STS*) requires dealing with heterogeneous function types, making it more costly and harder to interpret the results. Thus, at the basis of the *SUQ<sup>2</sup>* method is the adoption of the *GLD* *PDF* type, which is presented in the next section.

## 2.1 Generalized Lambda Distribution

The Generalized Lambda Distribution (*GLD*) has been applied to fitting phenomena in many fields with very good results. It was proposed by Ramberg and Schmeiser in 1974 [14] as an extension of the Tukey's distribution, and it is tuned to represent different data distributions through the specification of  $\lambda$  parameter, where  $\lambda_1$  and  $\lambda_2$  determine location and scale parameters, while  $\lambda_3$  and  $\lambda_4$  determine the skewness and kurtosis of the *GLD*( $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ ). The Ramberg and Schmeiser proposal is known as *RS* parameterization. The *RS* parametrization has some constraints with respect to the values of  $(\lambda_3, \lambda_4)$ , see [4]. To circumvent those constraints, Freimer et al. [5] introduced a new parameterization called *FMKL*:  $Q_{FMKL}(y|\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \lambda_1 + \frac{1}{\lambda_2} \left[ \frac{y^{\lambda_3}-1}{\lambda_3} - \frac{(1-y)^{\lambda_4}-1}{\lambda_4} \right]$ . As in the previous parameterization,  $\lambda_1$  and  $\lambda_2$  are the location and scale parameters, but in this one  $\lambda_3$  and  $\lambda_4$  are the tail index parameters. The advantage over the previous parameterization is that there is only one constraint on the parameters, i.e.  $\lambda_2$  must be positive.

Both representations (i.e. *RS* and *FMKL*) can present a wide variety of shapes and therefore are utilized in practice; however, generally the *FMKL* *GLD* is preferred due to the ease in its use [6]. In this paper, we opt for

using the *FMKL GLD* representation.

### 2.1.1 Shapes of GLD

A *GLD* can describe a variety of shapes, such as U-shaped, bell shaped, triangular, and exponentially [7]. At the same time it provides good fits to many well know distributions.

These *GLD* properties are important to the *SUQ<sup>2</sup>* method for two reasons. First, no previous knowledge is needed to fit the *GLD* to a dataset. Second, *GLDs* can be comparatively assessed; grouped based on their shapes, which enables running clustering algorithms, electing a representative distribution, and synthesizing the data in a cluster.

The shape of a *GLD* depends on its  $\lambda$  values. In the case of the *FMKL GLD* parameterization, Freimer et al. [5] classify the shapes into five categories depending on the variety of distributions, which can be represented by several combinations of the shape parameters  $\lambda_3$  and  $\lambda_4$ .

The ability to model different shapes is critical to the *SUQ<sup>2</sup>* approach as it is the basis for the clustering algorithms (see Section 3.2).

### 2.1.2 GLD mixture

In general, a *mixture distribution* is the probability distribution of a random variable that is derived from a collection of other random variables. Given a finite set of *PDFs*  $\{p_1(x), p_2(x), \dots, p_n(x)\}$ , and weights  $\{w_1, w_2, \dots, w_n\}$  such that  $w_i \geq 0$  and  $\sum w_i = 1$ , the mixture distribution can be represented by writing the density  $f(x)$  as a sum (which is a convex combination):  $f(x) = \sum_i^n w_i p_i(x)$ . Extending this concept to *GLD*, the mixture distribution can be represented as:  $f(x) = \sum_i^n w_i GLD_i(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ . This model is used in Section 3.3 to characterize the uncertainty in a spatio-temporal region.

## 3 Simulation Uncertainty Quantification Querying (SUQ<sup>2</sup>)

In a stochastic problem, on each spatio-temporal location  $(s_i, t_j)$  we have many realizations of  $q(s_i, t_j)$ . A schema to store this information in a relational database can be:

$$S(s_i, t_j, simId, q(s_i, t_j)) \quad (1)$$

where *simId* represents the *id* of one simulation (realization).

In this section, we first show how to fit a *GLD* to a spatio-temporal dataset. Next, we present the clustering of *GLDs* using the lambda parameters. Then, we present how to compute the uncertainty in a region using a mixture of *GLDs* and information entropy. Figure 1 shows the *SUQ<sup>2</sup>* method represented by a workflow and divided into three main steps: fitting process, clustering and UQ analysis.

### 3.1 Fitting a GLD to a spatio-temporal dataset

Given a random sample  $q_1, q_2, q_3, \dots, q_n$ , the basic problem in fitting a statistical distribution to these data is the distribution from which the sample was obtained. In our approach we divide this process into three steps: (1) fitting the *GLD* to the data; (2) validating the resulting *GLD*; (3) evaluating the quality of the fit.

Algorithm 1 realizes Step 1. Before starting the fitting process, we need to group all the simulation values that correspond to the same spatio-temporal location  $(s_i, t_j)$ . As a result, we get a new dataset  $S^*(s_i, t_j, < q_1, q_2, \dots, q_n >)$ , where  $q_i, 1 \leq i \leq n$ , represents a vector of all the values of  $q$  at point  $(s_i, t_j)$ . This process is efficiently computed according to the approach developed in [12].

For each spatio-temporal location  $(s_i, t_j) \in \mathcal{S} \times \mathcal{T}$ , we use a function of the GLDEX R package [7], to fit the *GLD* to a vector  $\langle q_1, q_2, \dots, q_n \rangle$ , line 2. This is an embarrassingly parallel computation method, which we adopt in [12].

Once the fitting process in Step 1 has been applied, a fitted *GLD* is associated to each simulation spatio-temporal position. The schema in Equation 1 is modified to accommodate the *GLD* parameters in place of the list of simulation values:  $S(s_i, t_j, GLD_i, j(\lambda_1, \lambda_2, \lambda_3, \lambda_4))$ .

Finally, we need to evaluate the fit quality, which assesses whether the *GLD* probability density function (PDF) correctly describes the dataset. We use here the Kolmogorov-Smirnov test (KS-test), that determines if two datasets differ significantly. In this case, the datasets are the original dataset and a second one generated using the fitted *GLD*. As a result, this test returns the p-value, line 5. If the p-value is bigger than 0.05, lines 6-7, we store the lambda values of those *GLDs*.

---

**Algorithm 1** Fitting the *GLD* to a spatio-temporal dataset

---

```

1: function GLDFIT( $S(s_i, t_j, \langle q_1, q_2, \dots, q_n \rangle)$ )
2:    $\langle \lambda_1, \lambda_2, \lambda_3, \lambda_4 \rangle \leftarrow \text{FIT.GLD.LM}(\langle q_1, q_2, \dots, q_n \rangle)$ 
3:    $isValid_{(s_i, t_j)} \leftarrow \text{VALIDITYCHECK}(\langle \lambda_3, \lambda_4 \rangle)$ 
4:   if  $isValid_{(s_i, t_j)}$  then
5:      $[pvalue, D]_{(s_i, t_j)} \leftarrow \text{KS}(\langle \lambda_1, \lambda_2, \lambda_3, \lambda_4 \rangle_{(s_i, t_j)})$ 
6:   end if
7:   if  $pvalue_{(s_i, t_j)} > 0.05$  then
8:      $\text{STORELAMBDAS}(\langle \lambda_1, \lambda_2, \lambda_3, \lambda_4 \rangle, s_i, t_j)$ 
9:   end if
10: end function

```

---

### 3.2 Clustering the *GLD* based on its lambda values

In Section 2.1.1, we discussed the two most important parameterizations of the *GLD* and selected *FMKL* to be used for the rest of the paper. In this parametrization,  $\lambda_1$  represents the location of the *GLD* and is directly related to the mean of the distribution.  $\lambda_2$  is the scale, directly related to the standard deviation, and  $\lambda_3$  and  $\lambda_4$  represent the left and right tails of the distribution. Combinations of  $\lambda_3$  and  $\lambda_4$  can be used to estimate the skewness and kurtosis of the distribution.

As  $\lambda_2$  defines the dispersion, and  $\lambda_3$  and  $\lambda_4$  the shape of a *GLD*, the combination of these parameters determine the quantification of the uncertainty, from the *GLD* point of view.

According to Lampasi et al. [8], a particular *GLD*( $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ ) can be rewritten as:

$$GLD(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \lambda_1 + \frac{1}{\lambda_2} GLD(0, 1, \lambda_3, \lambda_4) \quad (2)$$

In Equation 2, the first term applies  $\lambda_1$  while the second involves the remaining parameters. Then, we can apply clustering algorithms only considering parameters in the second term:  $\lambda_2$ ,  $\lambda_3$  and  $\lambda_4$ . The clustering algorithm would be applied in two steps 1. The first clusters based on the  $\lambda_2$  values, according to the curve dispersion. Next, for each cluster obtained in the first step, we cluster again according to parameters  $\lambda_3$  and  $\lambda_4$ , which are the parameters that define the shape of the distribution.

Then, in this step of our workflow, we cluster the *GLDs* using  $\lambda_2$ ,  $\lambda_3$  and  $\lambda_4$  values. The final result of this step is:

$$S_C(s_i, t_j, GLD_k, clusterID) \quad (3)$$

where *clusterID* represents the ID of the cluster to which the *GLD* at the spatio-temporal location  $(s_i, t_j)$  belongs. With the domain modeled by clustered *GLDs*, we can use this result to characterize the uncertainty in a

---

**Algorithm 2** Clustering the GLD based on its  $\lambda_{(2,3,4)}$  values.

---

```

1: function GLDCLUSTERING( $S(s_i, t_j, < 0, \lambda_2, \lambda_3, \lambda_4 >)$ )
2:    $S(s_i, t_j, clusterID_I) \leftarrow \text{FIRSTSTEP}(S(s_i, t_j, \lambda_2))$ 
3:   for each  $clusterID_I$  do
4:      $S(s_i, t_j, clusterID_{II}) \leftarrow \text{SECONDSTEP}(S(s_i, t_j, < \lambda_3, \lambda_4 >), S(s_i, t_j, clusterID_I))$ 
5:   end for
6: end function

```

---



---

**Algorithm 3** GLD mixture in a region  $(\mathcal{S}_i \times \mathcal{T}_j)$ 


---

```

1: function GLDMIXTURE( $\mathcal{S}_i \times \mathcal{T}_j, C_{\mathcal{S}_i \times \mathcal{T}_j}$ )
2:   for each  $p_i$  in  $(\mathcal{S}_i \times \mathcal{T}_j)$  do
3:      $c \leftarrow \text{cluster}(p_i)$ 
4:      $w_c = w_c + 1$ 
5:      $N = N + 1$ 
6:   end for
7:   return  $\frac{1}{N} \sum_c^{C(\mathcal{S}_i \times \mathcal{T}_j)} w_c * c.\text{getGLD}()$ 

```

---



---

**Algorithm 4** Information Entropy in a region  $(\mathcal{S}_i \times \mathcal{T}_j)$ 


---

```

1: function GLDMIXTURE( $\mathcal{S}_i \times \mathcal{T}_j, C_{\mathcal{S}_i \times \mathcal{T}_j}$ )
2:   for each  $p_i$  in  $(\mathcal{S}_i \times \mathcal{T}_j)$  do
3:      $c \leftarrow \text{cluster}(p_i)$ 
4:      $w_c = w_c + 1$ 
5:      $N = N + 1$ 
6:   end for
7:    $p_c(s, t) = \frac{w_c}{N}$ 
8:    $H(s, t) \leftarrow - \sum_{c=1}^C p_c(s, t) \log p_c(s, t)$ 
9:   return  $H(s, t)$ 

```

---

particular spatio-temporal region or to measure numerically the corresponding uncertainty. In Sections 3.3 and 3.4, we describe how these approaches are implemented (see Figure 1).

### 3.3 Use of GLD mixture to characterize uncertainty in a spatio-temporal region

One of the main advantages of assessing the complete probability distribution of the outputs is that we can use the *PDFs* to answer queries. If we consider that the clustering of GLD has good quality, we can pick the GLD at the centroid of each cluster as a representative of all its members. In this context, in a particular spatio-temporal region, each cluster may be qualified with a weight given by:  $w_k = \frac{1}{N} \sum_{i=1}^S \sum_{j=1}^T w(s_i, t_j)$ , where:

$$w(s_i, t_j) = \begin{cases} 1 & \text{if } clusterID(s_i, t_j) = k \\ 0 & \text{otherwise} \end{cases} \quad \text{and } N \text{ is the number of points in the region } (\mathcal{S}_i \times \mathcal{T}_j).$$

The weight  $w_k$  is the frequentist probability of occurrence of the cluster  $k$  in the region, and complies with the conditions defined in Section 2.1.2 that  $w_k \geq 0$  and  $\sum w_k = 1$ .

Remember that the mixture of the GLDs can be written as  $f(x) = \sum_{k=1}^K w_k GLD(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ . So, if we have the weights and a representative GLD for each cluster, we have the mixture of GLD that characterizes the uncertainty in the spatio-temporal region  $(\mathcal{S}_i \times \mathcal{T}_j)$ . The GLD mixture process is summarized in Algorithm 3, which receives a spatio-temporal region and a *clusterId* associated to each spatio-temporal point. In the main loop, lines 3 to 5, the algorithm increments the number of occurrences for each clusterId and the total number of points. At line 7, the mixture expression is returned.

### 3.4 Information entropy as a measure of uncertainty in a spatio-temporal region

Now, what happens if we want to measure the uncertainty quantitatively? The information entropy is useful in this context. We use the different clusters we got in Section 3.2 as the different outcomes of the system. The information entropy is computed as follows  $H(s, t) = - \sum_{c=1}^C p_c(s, t) \log p_c(s, t)$ , where  $c$  represent a particular cluster in the set of clusters  $C$ , and  $p_c(s, t)$  represents the probability of occurrence of the cluster  $c$  in the spatio-temporal region  $(s, t)$ .

Algorithm 4 computes the information entropy in a region  $C_{(S_i \times T_j)}$ . In lines 2 to 7, we assess the probability of each cluster in the region. Using this result, we can evaluate the information entropy  $H(s, t)$ , line 8, and finally, return the result in line 9.

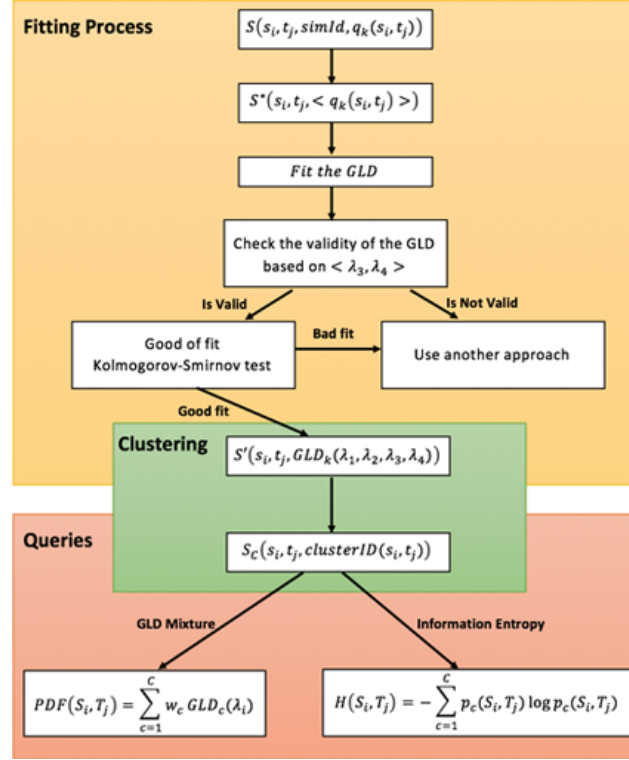


Figure 1: SUQ<sup>2</sup> workflow divided into three steps, (a) fitting process, (b) clustering of the GLDs and, (c) queries over the results of the clustering process.

## 4 Experimental Evaluation

In this section, we first introduce the data used in our experiments. Next, we discuss how we apply the fitting and clustering techniques over the experiment dataset. Then, we present the queries used in the performance evaluation and discuss the results expressed as a mixture of GLDs and values computed using information entropy.

As a case study, we use the HPC4e seismic benchmark, a collection of four 3D models and sixteen associated tests<sup>1</sup> to generate the data of a cube. The models include simple cases that can be used in the development stage of any geophysical imaging practitioner as well as extremely large cases that can only be solved in a reasonable time using supercomputers. The models are generated based on the required size by means of a Matlab/Octave script. The tests can be used to benchmark and compare the capabilities of different and innovative seismic modelling approaches, thus simplifying the task of assessing the algorithmic and computational advantages.

### 4.1 Dataset

In the HPC4e benchmark, the models are designed as sets of 16 layers with constant physical properties. The top layer delineates the topography and the other 15 delineate different layer interface surfaces or horizons. To

<sup>1</sup>The benchmark can be freely downloaded from <https://hpc4e.eu/downloads/>

generate a single cube with dimensions  $250 \times 501 \times 501$  we can use the values provided in the benchmark. For example, to generate a cube in the  $v_p(m/s)$  variable we can use the fixed values of Table 2. The first slice of this cube is shown in Figure 2.

Layer	$v_p(m/s)$	Layer	$v_p(m/s)$
1	1618.92	9	2712.06
2	1684.08	10	2532.2
3	1994.35	11	2841.03
4	2209.71	12	3169.31
5	2305.55	13	3169.31
6	2360.95	14	3642.28
7	2381.95	15	3659.22
8	2223.41	16	4000.00

Table 2: Values of  $v_p$  used in the generation of a the  $v_p$  attribute, to generate n velocity models. single velocity field cube.

Layer	PDF Family	Parameters	Layer	PDF Family	Parameters
1	Gaussian	[1619, 711.2]	9	Exponential	[3949, 394.9]
2	Gaussian	[3368, 711.2]	10	Exponential	[5983, 711.2]
3	Gaussian	[8839, 711.2]	11	Exponential	[3520, 352.0]
4	Gaussian	[7698, 301.5]	12	Exponential	[3155, 315.5]
5	Lognormal	[7723, 294.7]	13	Uniform	[2541, 396.4]
6	Lognormal	[7733, 292.2]	14	Uniform	[2931, 435.3]
7	Lognormal	[7658, 312.1]	15	Uniform	[2948, 437.0]
8	Lognormal	[3687, 368.7]	16	Uniform	[3289, 471.1]

Table 3: PDFs and the parameters used to sample

As our purpose is to study the uncertainty in the simulation output, we need the input  $v_p(m/s)$  to present a stochastic behavior. We model the input according to the distributions depicted in Table 3. Next, using a Monte Carlo method, we generate a sampling of 1000 realizations of the  $v_p(m/s)$  variable and use a Matlab script provided by the HPC4e benchmark to generate the cube data. We perform the simulations 1000 times, one for each realization, and generate 1000 cubes (230 GB) as output. The generated cubes are  $250 \times 501 \times 501$  multi-dimensional arrays. In order to simplify the computational process and visualize the results, we select the slice 200 to be used in our experiments. Then, we have 1000 realizations of a slice with size of  $250 \times 501$ . The data schema in Equation 1 can be simplified as we only have two spatial dimensions and no time domain. Thus, the dataset can be represented as  $S(x_i, y_j, simId, v_p(x_i, y_j))$ . In this new representation,  $(x_i, y_j)$  is the 2D coordinates and  $v_p(x_i, y_j)$  is the velocity value at point  $(x_i, y_j)$ . *simId* represents the Id of the simulation, ranging from 1 to 1000.

## 4.2 Fitting the GLD

The first step is to find the *GLD* that best fits the dataset at each spatial location. Running Algorithm 1, we get a new 2D array:

$$S'(x_i, y_j, GLD(\lambda_1, \lambda_2, \lambda_3, \lambda_4)) \quad (4)$$

The raw data is significantly reduced and the new dataset is characterized by four lambda values at each spatial location.

To check how good is the fit, we use the *ks.test* algorithm included in the R-package *stats* [13], which return the *p-value* at each spatial location. Our results show that the fit of the GLD is acceptable in most cases (*p-value*  $> 0.05$ ), in 82 % of the spatial locations (see Figure 3). In the 18% regions where the GLD modeling was not acceptable, some *GLD* extensions proposed in [4] could be used. Since the main purpose of this paper is to demonstrate the usefulness of the *GLD* in *UQ*, this particular problem is beyond our scope.

## 4.3 Clustering

Up to now, the dataset is characterized by the schema depicted by Equation 4. Then, using a clustering algorithm, such as k-means, we group the GLDs based on its  $(\lambda_2, \lambda_3, \lambda_4)$  values, as discussed in Section 3.2. In this paper,



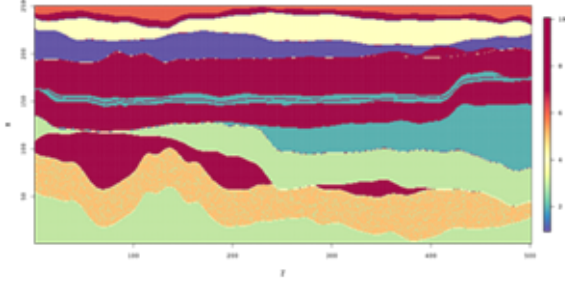


Figure 2: One slice of the  $250 \times 501 \times 501$  cube. In the slice, we can distinguish between the different layers.

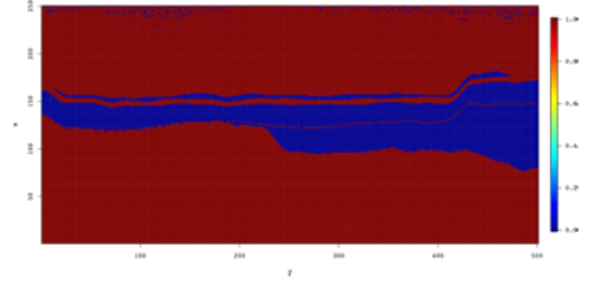


Figure 3: The red color shows where the p-value is greater than 0.05.

we use the k-means algorithm with  $k = 10$ . The choice of the clustering algorithm and the parameterization is subject for further investigation and is beyond the scope of this paper.

Once the clustering algorithm is applied, a new dataset is produced. In the new dataset, for each spatial location, a label indicates the cluster the *GLD* belongs to, as shown (see the schema at Equation 5) in Figure 4. Note that in Figure 4, the blue region corresponding to cluster 11 is not a cluster itself. It is rather the region where the *GLD* is not valid (see Section 4.2).

$$S_C(x_i, y_j, clusterID, GLD_{x_i, y_j}) \quad (5)$$

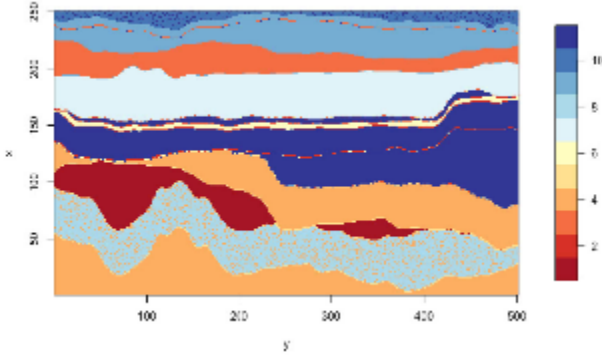


Figure 4: Result of the clustering using k-means with  $k = 10$ .

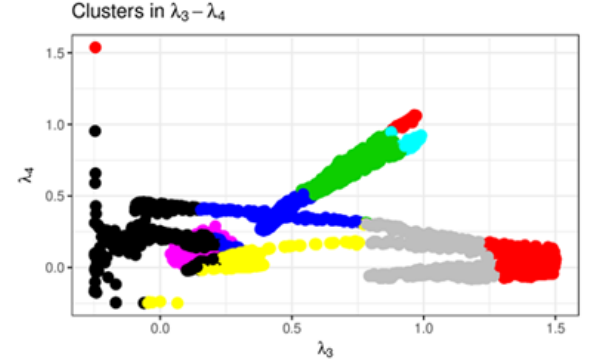


Figure 5: Distribution of the clusters in the  $(\lambda_3, \lambda_4)$  space. The points that belongs to a same cluster are one near the others, as was expected.

If we visually compare Figure 2 with Figure 4, we observe a close similarity.

Another interesting result is shown in Figure 5, where we plot the clusters in  $(\lambda_3, \lambda_4)$  space. As we mentioned in Section 2.1.1, the shape of the *GLD* depends on the values of  $\lambda_3$  and  $\lambda_4$ . In this scenario, the expected result is that the members of the same cluster share similar values of  $\lambda_3$  and  $\lambda_4$ . This is exactly the result we can observe in Figure 5.

To further corroborate this fact, Figure 6 shows the *PDFs* of 60 members of the 10 clusters. Visually assessing the figures gives an idea of how similar are the shapes of the members of a same cluster and how dissimilar are the shapes of the members of different clusters. This suggests that our approach is valid. A product of these observations is that we can pick one member of each cluster (the centroid) as a representative

of all the members of this cluster, Table 4. The selected member is going to be used to answer the queries in the next Sections.

Cluster	$\lambda_2$	$\lambda_3$	$\lambda_4$	Cluster	$\lambda_2$	$\lambda_3$	$\lambda_4$
1	0.0013937313	0.9585829	1.04696461	6	0.0003894541	1.4076354	-0.01925743
2	0.0005291388	1.1633978	-0.07162550	7	0.0021972784	0.3253562	0.01493809
3	0.0020630696	0.1349486	0.17305941	8	0.0015421749	0.9491101	0.86699555
4	0.0016238358	0.8653824	0.83857646	9	0.0018672401	0.2176002	0.17862024
5	0.0027346929	0.5084664	0.39199164	10	0.4856397733	0.1404140	0.14011298

Table 4: Clusters centroids.

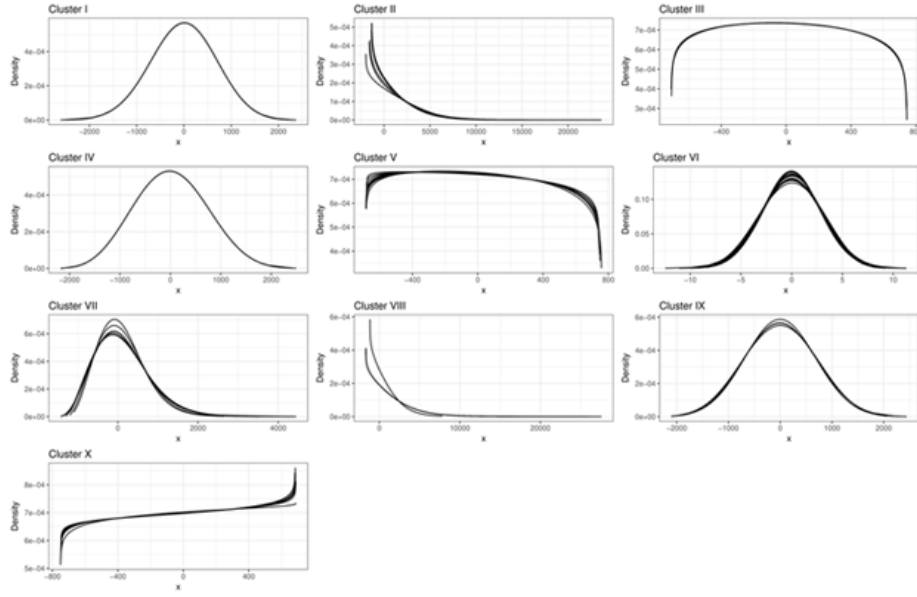


Figure 6: PDFs of 60 members of the 10 clusters obtained using k-means over the  $(\lambda_2, \lambda_3, \lambda_4)$  values.

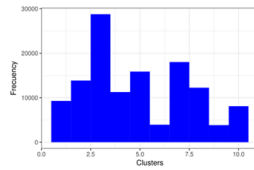


Figure 7: Distribution of the 125250 points by clusters.

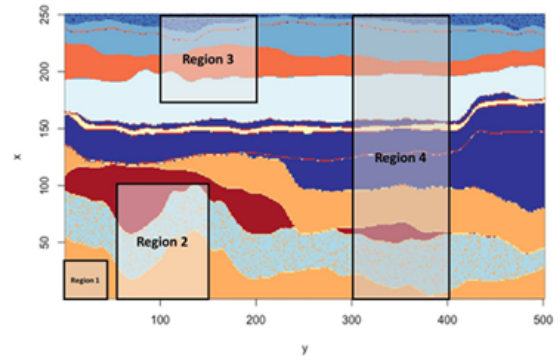


Figure 8: Analysis Regions. The four regions where selected intentionally in this way to warranty different distributions of the clusters inside it.

The 125250 points of the slice are distributed through the clusters following the histogram of Figure 7.

## 4.4 Spatio-temporal queries

Up to this point, the initial dataset is summarized as depicted by the schema in Equation 5. It can be used to answer queries and validate our approach, by comparing the results with the raw data.

First of all, we select four spatio-temporal regions of the dataset where the clusters suggest us different behaviors. The regions are shown in Figure 8. With these four regions, we assess the adoption of the *GLD* mixture to obtain the *PDF* that characterizes the uncertainty in a specific region (see Sections 4.4.1 and 4.4.2). We use the information entropy to assign a value that measures the uncertainty at each region. In Section 4.4.1, we expect the *GLD* mixture to characterize well the raw data, and in Section 4.4.2, we hope that the information entropy is zero in region 1 and increases between regions 2, 3 and 4.

### 4.4.1 GLD mixture

In this experiment, we use the representative *GLDs* at each cluster and the weight associated to it in the region. Using these parameters, we can build a *GLD mixture* that characterizes the uncertainty on that region. We use Algorithm 3 described in Section 2.1.2. First, we query the region to find the clusters represented inside it, and how they are distributed. The retrieved results are shown in Table 5. If we divide the columns of Table 5 by the sum of the elements of each column, we get the weight needed to formulate the *mixed GLDs*. It is clear that the *GLD* in region 1 is represented by the *GLD* of cluster 4. In the other three cases, we get what is shown in the set of equations 6.

Cluster	Region 1	Region 2	Region 3	Region 4
1	0	2250	0	979
2	0	0	0	268
3	0	0	2596	1468
4	1640	4467	0	5173
5	0	149	0	269
6	0	0	0	416
7	0	0	1967	3920
8	0	3335	0	3432
9	0	0	1918	3280
10	0	0	901	583

Table 5: Distribution of the clusters by regions. The four regions are selected intentionally this way to warrant different distributions of the clusters inside it.

$$\begin{aligned}
GLD_{region2} &= 0.22GLD_{c1} + 0.44GLD_{c4} + 0.014GLD_{c5} + 0.33GLD_{c8} \\
GLD_{region3} &= 0.34GLD_{c3} + 0.26GLD_{c7} + 0.25GLD_{c9} + 0.12GLD_{c10} \\
GLD_{region4} &= 0.22GLD_{c1} + 0.44GLD_{c4} + 0.014GLD_{c5} + 0.33GLD_{c8}
\end{aligned} \tag{6}$$

Now, we need to evaluate whether the *mixture of GLDs* correctly models the uncertainty in a region. We perform the same *ks-test* used to evaluate the good quality of the fit, described in Section 3.1. Based on the *p-value*, Table 6, we can conclude that in all 4 regions the *mixture of GLDs* is a good fit to the raw data.

Metrics	Region 1	Region 2	Region 3	Region 4
p-value	0.73	0.56	0.34	0.08

Table 6: p-values by regions.

#### 4.4.2 Information Entropy

Based on the distribution of clusters inside the regions (see Table 5), we can compute the entropy.

entropy	Region 1	Region 2	Region 3	Region 4
value	0	1.122243	1.41166	2.024246

Table 7: Information Entropy by regions.

As we expect (see Table 7), the entropy in region 1 is zero, because the region contains only members of the cluster 4. On the other regions the entropy increases from region 2 to region 4, as expected. The information entropy is a very good and simple measure of the uncertainty, and here it is demonstrated its usefulness combined with the *GLD*.

## 5 Conclusion

In this paper, we proposed  $SUQ^2$ , a method to answer uncertainty quantification (UQ) queries over large spatio-temporal simulations.  $SUQ^2$  trades large simulation data by probability density functions (PDFs), thus saving huge amount of storage space and computational cost. It enables complex data distribution representation at each simulation point, as much as a spatio-temporal view of simulation uncertainty computed by mixing spatial point PDFs. We evaluated  $SUQ^2$  using a seismology use case, considering the computation of uncertainty in regions of a slice of the seismic cube. The results show that  $SUQ^2$  method produces an accurate view of the uncertainty in regions of space-time while considerably saving storage space and reducing the cost associated with the PDF modeling of the dataset. To the best of our knowledge, this is the first work to use GLD as the basis for answering UQ queries in spatio-temporal regions and to compile a series of techniques to produce a query answering workflow.

## Acknowledgments

This work has been funded by CNPq, CAPES, FAPERJ, the Inria HPDaSc and SciDISC Associated Teams and the European Commission (HPC4E H2020) project.

## References

- [1] D. R. Tobergte and S. Curtis. Workshop on Quantification, Communication, and Interpretation of Uncertainty in Simulation and Data Science. in *Journal of Chemical Information and Modeling*, 53(9):1689–1699, 2013.
- [2] G. Baroni and S. Tarantola. A General Probabilistic Framework for uncertainty and global sensitivity analysis of deterministic models: A hydrological case study. in *Environmental Modelling and Software*, 51:26–34, 2014.
- [3] R. H. Johnstone, E. T. Y. Chang, R. Bardenet, T. P. de Boer, D. J. Gavaghan, P. Pathmanathan, R. H. Clayton, and G. R. Mirams. Uncertainty and variability in models of the cardiac action potential: Can we build trustworthy models? in *Journal of Molecular and Cellular Cardiology*, 96:49–62, 2016.
- [4] Z. A. Karian and E. J. Dudewicz. *Handbook of fitting statistical distributions with R*. 2011.
- [5] M. Freimer, C. T. Lin, and G. S. Mudholkar. A Study Of The Generalized Tukey Lambda Family. in *Communications in Statistics - Theory and Methods*, 17(10):3547–3567, 1988.
- [6] C. G. Corlu and M. Meterelliyoz. Estimating the Parameters of the Generalized Lambda Distribution: Which Method Performs Best? in *Communications in Statistics: Simulation and Computation*, 45(7):2276–2296, 2016.

- [7] S. Su. Fitting Single and Mixture of Generalized Lambda Distributions to Data via Discretized and Maximum Likelihood Methods: GLDEX in R. *Journal of Statistical Software*, 21(9), 2007.
- [8] D. A. Lampasi, F. Di Nicola, and L. Podesta. Generalized lambda distribution for the expression of measurement uncertainty. *IEEE Transactions on Instrumentation and Measurement*, 55(4):1281–1287, 2006.
- [9] M. M. Movahedi, M. R. Lotfi, and M. Nayyeri. A solution to determining the reliability of products Using Generalized Lambda Distribution. *Research Journal of Recent Sciences Res.J.Recent Sci*, 2(10):41–47, 2013.
- [10] W. Ning, Y. Gao, and E. J. Dudewicz. Fitting mixture distributions using generalized lambda distributions and comparison with normal mixtures. *American Journal of Mathematical and Management Sciences*, 28(1-2):81–99, 2008.
- [11] J. F. Wellmann and K. R. Lieb. Uncertainties have a meaning: Information entropy as a quality measure for 3-D geological models. *Tectonophysics*, 526-529:207–216, 2012.
- [12] J. Liu, N. Lemus, E. Pacitti, F. Porto, P. Valduriez. Parallel computation of PDFs on big spatial data using spark. in *em Distributed and Parallel Databases*, pp. 1-28. In press, 10.1007/s10619-019-07260-3, 2019.
- [13] R. H. C. Lopes Kolmogorov-Smirnov Test. in Lovric M. (eds) *International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg, 2011.
- [14] J. S. Ramberg, and B. W. Schmeiser. An approximate method for generating asymmetric random variables. in *em Communications of the ACM*, 17(2), 78–82. doi:10.1145/360827.360840, 1974.

# Networking and Storage: The Next Computing Elements in Exascale Systems?

Alberto Lerner<sup>1</sup> Rana Hussein<sup>1</sup> André Ryser<sup>1</sup> Sangjin Lee<sup>2\*</sup> Philippe Cudré-Mauroux<sup>1</sup>

<sup>1</sup>*University of Fribourg  
Switzerland*

<sup>2</sup>*Hanyang University  
South Korea*

## Abstract

*Many large computer clusters offer alternative computing elements in addition to general-purpose CPUs. GPU and FPGAs are very common choices. Two emerging technologies can further widen the options in that context: in-network computing (INC) and near-storage processing (NSP). These technologies support computing over data that is in transit between nodes or inside the storage stack, respectively. There are several advantages to moving computations to INC and NSP platforms. Notably, the original computation path does not need to be altered to route data through these subsystems; the network and the storage are naturally present in most computations.*

*In this paper, we describe the evolutionary steps that led to INC and NSP platforms and discuss how they can improve critical computing paths in large-scale database systems. In the process, we comment on the constraints that the current generation of these platforms present as well as expose why we believe them to be relevant to the next generation of exascale platforms.*

## 1 Motivation

The networking and storage stacks have always carried some computing power. Network switches can triage billions of packets per second. Solid-state drives (SSDs) can scramble and encode gigabytes of data per second (for error correction purposes [9]). Despite such computing power, applications have no access to how the devices process the data, other than by issuing IO requests. The functionality of these devices has been closed to changes. Opening them would require supporting a certain level of *programmability*.

Nonetheless, the benefits of executing application logic close to networking and storage devices are known [17, 40]. New algorithms that take advantage of proximity to the data become viable and provide both performance and power consumption gains. Figure 1(a) illustrates how FPGAs and network “middle-boxes” can create these opportunities.

The lack of programmability in network and storage devices has impacted not only applications. It has also hindered the advancement of these platforms. On the networking side, new protocols emerge that depend heavily on hardware to run at high-speeds. *Fixed-function* devices, the ones that have their functionality “baked”

---

*Copyright 2020 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.*

**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

---

\*Work done while visiting the University of Fribourg.

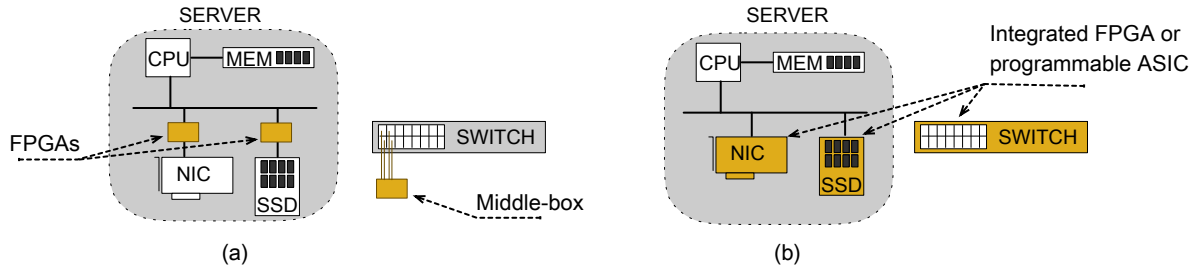


Figure 1: Alternative in-network computing and near-storage processing platforms. (a) Using “bump-in-the-wire” FPGAs or network “middle-boxes” to create early application logic sites. (b) Leveraging programmability within the device for application logic.

into the hardware, may need to undergo a full, lengthy development cycle before they can run new protocols. Recently, a solution emerged where a new class of networking devices started supporting programmability (e.g., in programmable switches [6] and “smart” NICs [44]). The protocols these devices run are expressed as software programs.

The need for programmability also emerged on storage platforms recently. SSDs are designed to shield applications from the intricacies of managing flash memory. However, evidence appeared that SSDs often miss the right performance decisions because of the separation between it and the applications [3]. A more *modular* SSD design, deemed *open-channel*, was proposed that allows a host (a server), rather than the SSD itself, to make certain low-level decisions. The modules that run on the host are software-based, therefore programmable.

Developers soon realized that the programmability could also be used by applications. It made it possible to inject application code directly into the networking and storage stacks. Figure 1(b) depicts such a scenario. *In-network computing* and *near-storage processing* emerged as the disciplines that leverage processing power in the network and storage stacks, respectively, for application purposes.

An application that benefits from INC or NSP contains, by definition, algorithms running in different computing elements of a system. These algorithms ought to communicate at high speed. The most commonly used interconnect for such systems has been based on the PCIe bus standard [8]. At this time, the standard is being revised to operate at higher speeds. PCIe is also being extended with mechanisms to support cache coherence protocols to run atop of it [12, 14].

Despite their potential, INC and NSP platforms are not without challenges. INC is a more mature technology with increasingly popular programmable models that shield the developer from the intricacies of the devices. Such programming models, however, are quite different from a general-purpose CPU’s. In contrast, NSP often uses general-purpose processors—but much less powerful ones than server-class CPUs. In both cases, porting algorithms to these platforms requires rethinking the algorithms to fit either a different model or a less-powerful environment.

In this paper, we elaborate on the above challenges and opportunities of adopting INC and NSP as platforms to improve application performance. We summarize the contributions of this paper as follows:

- We discuss the evolution and recent advancements of INC and NSP platforms;
- We present specific computation tasks that can benefit from them;
- We describe the challenges that still remain in order to use INC and NSP as application platforms;
- Finally, we present the benefits of adopting the current generation of INC and NSP.

The rest of this paper is structured as follows. We discuss the programmability of the network and storage stacks in more detail in Section 2. We show examples of computing paths that can leverage INC and NSP capabilities in Section 3. We comment on the evolution of the PCIe interconnect and the possibilities it opens

in Section 4. We discuss the challenges and opportunities of adopting INC and NSP platforms in Section 5. We elaborate on how ready for adoption the technologies are in Section 6, before concluding in Section 7.

## 2 An Abridged History of INC and NSP

There have been many attempts to conciliate speed and flexibility in networking hardware devices. Figure 2 depicts the main ideas behind the relevant milestones in that context.

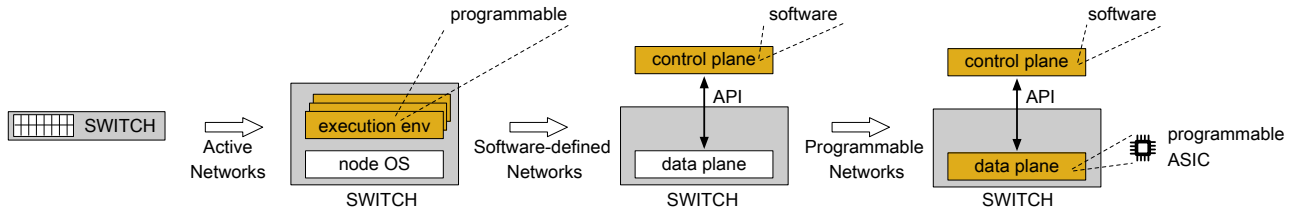


Figure 2: Evolution of the network stack towards programmability.

*Active Networks*, one of the first attempts, proposed a standard architecture for more flexible network devices [10]. The goal of the architecture is to enable changes to the device behavior—typically to implement new protocols—via the execution of user-driven customization. There are two main components in such a device. First, a node operating system that manages basic IO functionality, including processing, storage, and transmission. Second, multiple Execution Environments (EE) that support the execution of programs containing packet-processing logic. Applications can inject code into the device through an API, which generates special packets for the device to execute. Active networks proved to be very flexible; however, they ran considerably slower than their fixed-function counterparts.

*Software-Defined Networks* (SDN), a later attempt at a more flexible network, is also centered around a standard device architecture [23]. The main feature of the architecture is the separation of the control plane (policy definition) from the data plane (policy implementation). The control plane would contain information such as routing tables; the data plane would use that information to forward packets without sacrificing speed. An SDN device could adopt a new protocol if the SDN standard recognizes that protocol. Introducing new protocols, however, still required changes in the SDN standard itself and sometimes also on the devices. The main issue was that SDN only supports custom logic on the control plane. Protocols that depended on special per-packet logic need to send these packets to the control plane, which in turn processes them, pushing the results back to the data plane. The additional path cannot sustain *line-rate*, as one calls the individual port speed of a switch.

In a more recent development, *Programmable Dataplanes* (also referred to as *Programmable Networks*) allowed custom logic on the data plane [2]. The cornerstone of this technology is a generation of programmable ASICs (chips) that supports a certain degree of stateful packet processing without compromising speed [6]. This balance is captured by a programming model called Protocol Independent Switch Architecture (PISA) [38]. PISA shields the programmer from various device intricacies and, because it was designed to be protocol-independent, it proved adept at expressing application computations as well. Programmable networks created the conditions for a new generation of applications to push specialized logic into the network, i.e., to perform *in-network computing*.

We now turn our attention to the evolution of the storage stack. Just as with networks, the idea of near-storage processing is not new, but the driver for altering this stack has been more application-centric than in networking. In the early 2000s, a seminal proposal, called *Active Disks*, gained traction that aimed at transforming hard drive controllers into data-processing platforms [35]. These controllers often carried a small, general-purpose



processor that was somewhat over-dimensioned for its purpose. Applications could place logic on that processor, and access/modify the data blocks that were streaming in and out of the hard drive. The computing power of these controllers ultimately proved to be underwhelming, and the industry saw little need to improve them.

Eventually, SSDs based on NAND-flash displaced hard drives in many applications. Managing flash memory requires considerably more computing power than a magnetic medium, and flash memory error rates are very high, which requires computational-intensive error correction techniques [9]. Moreover, the industry decided that SSDs should be drop-in replacements for hard drives. SSDs execute internally a compatibility layer called Flash Translation Layer (FTL) for that purpose [13]. All these factors turn SSDs into relatively powerful embedded devices.

A proposal soon emerged to leverage SSD controllers for database query processing using *smart SSDs* [15]. As with active disks, the computing capacity on early devices was not sufficient for most computations in practice [16]. It would take a new generation of devices to accommodate fast database logic [21]. However, the industry focus was more on making devices faster than on making them smarter. Performance wasting was an issue, as the FTL not always makes the best decisions from an application’s point of view. Some efforts emerged that tried to make SSD architectures more open to configurations. Figure 3 captures the essential steps of this effort.

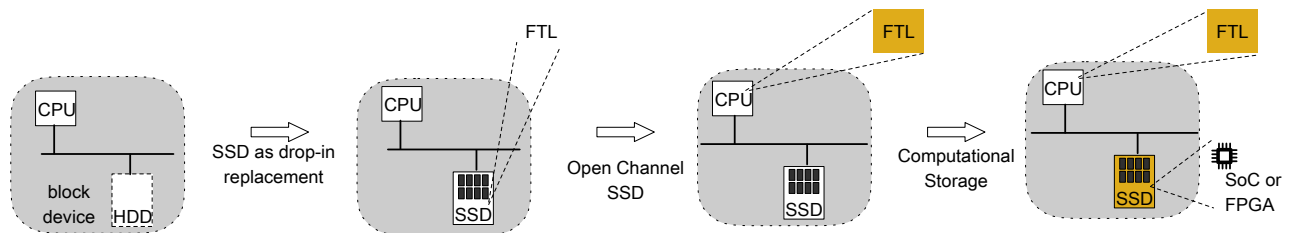


Figure 3: Evolution of the storage stack towards programmability.

*Open Channel SSDs* appeared from a growing understanding that many of the FTL tasks were better performed leveraging application knowledge [3]. In an open-channel SSD, some of the FTL responsibilities are removed from the device. Instead, they run on the host housing the SSD or even inside individual applications [30]. Naturally, these devices expose a lower-layer view of the underlying flash medium.

With more exposure to SSD architectural details, several works emerged to build tooling around these devices. These efforts range from frameworks to program SSD controllers [33], to performance measurement tooling [26], to even full SSD rapid prototyping platforms [24]. At the same time, a class of works appeared that deploy application code on SSDs, making them a *Computational Storage* platform [32, 37, 43]. The common thread in these works is that they increase the existing computing power of the devices by building them on platforms such as a System-on-a-Chip (SoC) or an FPGA. The tooling and the additional computing capacity paved the way for *Near-Storage Processing*.

One problem that remains open, however, is the lack of a consensus around a programming model. While there are recent proposals in that sense (e.g., in [18]), these models leave many aspects undefined. For instance, they have not addressed how to interface application logic with internal device mechanisms. An application that wishes to influence the IO scheduling policy of a device directly has no means to do so. There is also no consensus on how to shield an application developer from specific aspects of different devices.

### 3 Alternative Computing Paths

INC and NSP provide alternative sites beyond a CPU where applications can place logic. This fundamentally changes the traditional data movement patterns we see in CPU-centric algorithms, potentially bringing performance and power savings benefits. We illustrate these possibilities in this section by presenting and discussing

several use cases in large-scale data management.

### 3.1 In-Network Data Aggregation

Data aggregation is a very common operation in data management. Aggregation involves grouping data by specific criteria and calculating summaries over each group. A typical example is the `GROUP BY` clause in `SQL`. When the data volume is large, the operation can involve several servers, e.g., as in a rack-wide computation. Figure 4(a) illustrates one possible way to solve such a distributed aggregation.

The servers agree on how to partition the data, taking into consideration the grouping key. This divides the work into disjoint, independent tasks. Each server reshuffles its data while performing the aggregation over its assigned partition, as data arrives. Eventually, all the machines send their aggregation results to an elected server, which combines them into a global result. Note that throughout all these interactions, the switch performs data transmissions only.

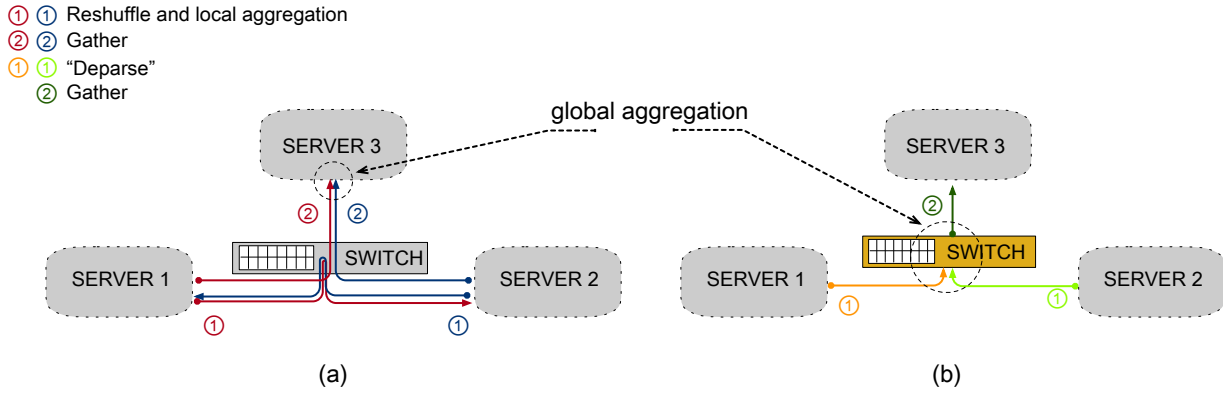


Figure 4: Data aggregation scenario (e.g., `SQL GROUP BY`). (a) With a “passive” switch, the servers first repartition the data and work on aggregating its assigned partition locally. Then, an elected server gathers all partitions and performs the global aggregation. (b) On an INC platform, an “active” switch can, in typical situations, perform the global aggregation in one step and transmit the results to an elected server. We use a given color to represent each unique data stream in the computation.

In Figure 4(b), we show that using the switch as an active element simplifies the entire computing path [25]. We assume that the switch is programmable and that it can be leveraged to hold an entire aggregation table. The size of such a table is often manageable, as it is proportional to the number of groups involved, rather than the size of the dataset. Note that the servers need not reshuffle the data nor perform an aggregation on a partition. Because programmable switches perform computations at line-speed, the aggregation table on the switch will be completed as soon as the last server finishes transmitting its tuples, and can then be sent immediately to an elected server.

Not all computations are suitable for INC deployment. The important caveats include: (a) the number of steps the switch can execute over each packet is limited; (b) the kind of instructions the switch can perform is also constrained; and (c) programmable network devices adhere to a programming model that imposes a *forward logic*-style onto algorithm design. Loops and complex branching are strongly discouraged, although possible. In practice, these restrictions reduce the choices of data structures the switch can support. In particular, the aggregation described in Figure 4(b) requires a hash table to be adapted to INC constraints. In this case, the number of collisions can be handled on the switch up to a certain bound. Relaxing this constraint involves using a technique called “overflowing” [25], which allows treating long collision chains outside the switch without a noticeable performance penalty in most cases.

These limitations notwithstanding, a large class of computations can benefit from INC platforms [34]. Moreover, processing data on the switch tends to scale well as the number of servers grows in the cluster. The number of switches naturally grows with the number of servers.

### 3.2 Near-Storage Checkpoint Derivation

We now describe an opportunity that arises specifically in an in-memory database system. Like most DBMSs, in-memory databases guarantee durability using a persistent transaction log. Because the log can get arbitrarily long, it would be impractical to recover from a crash just by replaying it. Therefore, databases also perform a periodical checkpoint (e.g., by taking a snapshot) of the current memory state and write it to persistent storage, often SSDs. A recovery algorithm can load the snapshot and replay the portion of the log acquired after the checkpoint. In in-memory databases, the log and checkpoint are the only workloads written to disk.

The logging and checkpointing processes compete for both disk and memory bandwidth. Figure 5(a) shows these contention points. The checkpointing process reads the database contents from the main-memory while it is being queried/modified. The checkpointing process also issues write requests to the SSD, which have to be scheduled along with the logging workload. The contention is responsible for the throughput reduction that many systems experience during checkpoints.

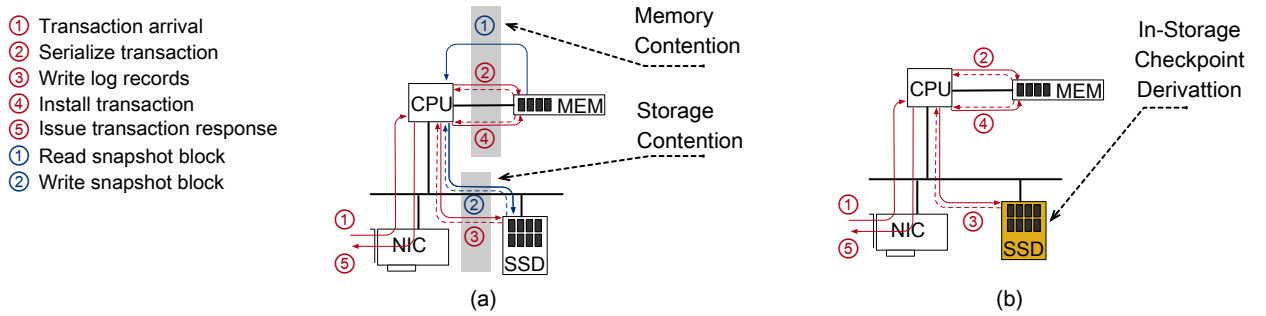


Figure 5: Checkpoint computation scenario. (a) Transaction logging (red) and checkpoint (blue) are two parallel processes. They compete for memory and disk bandwidth. (b) A checkpoint could be derived by processing the transaction log inside an SSD. The solid lines represent data; the dashed ones, control and/or return.

The key observation here is that partial snapshots, which can serve as checkpoints, can be derived from the log stream directly. We believe that such derivations may very well occur inside a smart SSD. We show in Figure 5(b) that once we move that process to the device, the contention points disappear.

Note, however, that the processing power on an SSD is far smaller than that of a general-purpose CPU. We cannot possibly expect to move the same algorithm we used on a CPU into an NSP platform and obtain similar performance results. Creating a checkpoint derivation algorithm for an SSD requires finding snapshot approximations that the device can process at the necessary pace. We comment on Section 5 how specific architectural changes on smart SSDs can make this task easier.

### 3.3 Low Latency Database Replication

Another code path that can benefit from either INC or NSP is that of a replica node in a database system. On a master node, transactions need to be serialized via a concurrency control algorithm. The latter typically runs on a CPU. The replica node code-path is simpler because the serial order of the transactions was already determined. The CPU takes the modifications coming from the network card and persists them on disk in the same order it received them. Subsequently, it updates its data structures and sends a notification to the master node that it accepted the transaction. Figure 6(a) depicts such an interaction.

- ① Replication request arrival
- ② Write log records
- ③ Install mutations
- ④ Send replication ack
- ① Replication request arrival
- ② Write log records
- ③ Log records written
- ④ Send replication ack
- ① Read request
- ② Install mutations
- ③ Ack mutations

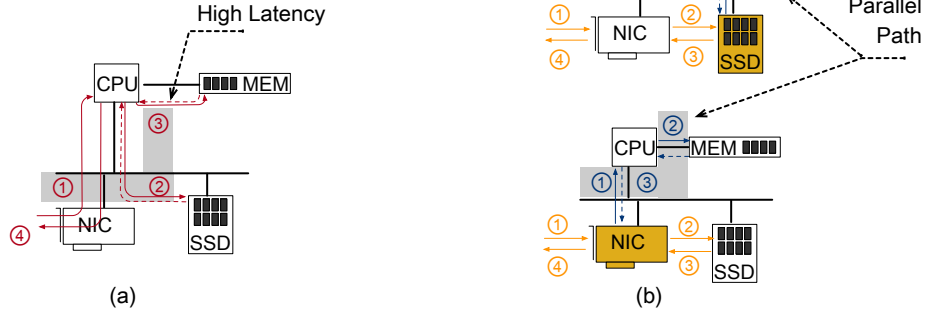


Figure 6: Database replica node scenario. (a) Each transaction log entry is first persisted in storage and then applied to memory, as the CPU coordinates the replication. (b) If an “active” NIC or SSD coordinates the process, the persistence and memory update paths can proceed in parallel.

Note that the replica code path incurs in latency. The master node may be withholding the original transaction until the replication path is completed. We can optimize this code path in at least two ways. Using NSP, the NIC would send the modification stream directly to a smart SSD. In turn, the SSD could be programmed to coordinate the interaction instead of the CPU. The SSD would notify the NIC after it persisted the changes, reducing the latency. It would also, in parallel, allow the CPU to read (and perform) the modifications. An alternative path exists where the NIC itself would perform the coordination. Figure 6(b) shows both cases. This scenario involves having the NIC and the SSD communicate without any CPU intervention. This type of communication is called peer-to-peer DMA [8] and it has been used in other contexts before, such as direct access to a remote disk (e.g., NVMe-over-Fabrics [19]).

## 4 Upcoming Interconnects

INC and NSP platforms rely on an interconnect to communicate with other computing elements in a system. PCIe has been the *de facto* interconnect for quite a while [8]. PCIe is a point-to-point bus with a variable number of lanes dedicated to devices. Each lane can transfer close to 1GB/s on the current standard version, Gen 3. Cards attached to the bus can use  $1\times$ ,  $2\times$ ,  $4\times$ ,  $8\times$ , or  $16\times$  lanes to achieve a theoretical maximum of 16 GB/s bidirectional bandwidth.

Newer devices are creating the need for faster speeds. For instance, the standard NIC port speeds are about to go from 100 Gb/s to 400 Gb/s, which a single Gen 3 PCIe slot can no longer support. The PCIe standard had moved to 2 GB/s lanes on Gen 4. The following iteration of the standard, Gen 5, brings 4 GB/s lanes with a theoretical bi-directional bandwidth of 64GB/s for  $16\times$  device.

There is another compelling feature that new versions of PCIe buses will bring: cache coherence. This feature allows computing elements to negotiate who may be caching a given portion of memory at any given time. It also determines who has the license to write to that memory address. Coordinating access to single memory address allows all the computing elements to share a single view of memory, as Figure 7(a) shows. There are two flavors of coherency: one in which the CPU has a prominent role in controlling the memory, called *asymmetric*, and one in which all computing elements play a similar role, called *symmetric*.

At the time of writing, there are four upcoming interconnects offering both high speed and coherence: CAPI [39], CCIX [12], CXL [14] and Gen-Z [22]. CCIX and Gen-Z support symmetric coherence, unlike CAPI or CXL, which support asymmetric coherence. CAPI is a competing standard to PCIe. CCIX and CXL use only

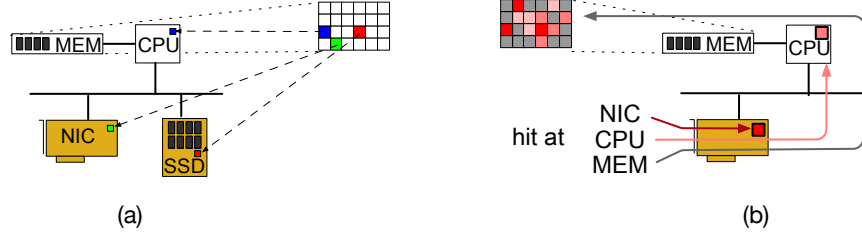


Figure 7: Cache coherency scenarios: (a) a NIC and an SSD may be responsible for writing to specific addresses in memory that can be read by other elements, and (b) a NIC can be responsible for caching (and writing to) very hot pages while leaving the CPU responsible for less accessed pages.

PCIe, while Gen-Z can use ethernet as well as PCIe. The range of PCIe is short enough to work only within the confines of a chassis, while ethernet allow to connect across chassis. This opens up the possibility of CCIX or CXL being applied simultaneously with Gen-Z to form coherence domains that cross server boundaries.

Cache coherency allows for new INC and NSP design use cases. For instance, one may extend the caching hierarchy beyond the CPU. Figure 7(b) illustrates this possibility. Some distributed applications may cache hot data items in the NIC [41]. With coherence, the NIC can also request write access to such an item and update it. Any other computing element that requests to cache that address would then see the updates.

## 5 Challenges and Opportunities

We mentioned above some immediate difficulties that arise when deploying current generation INC and NSP platforms. In this section, we elaborate on further challenges and opportunities that we believe would unlock more of the potential of these platforms.

**In-Network Stateful Computations.** Data management primarily involves stateful computations, i.e., algorithms that access a result from a previous iteration and generate a new result at the next one. The aggregation operation discussed in Section 3.1 is one such algorithm. Maintaining state in-network, particularly in switches, is currently very challenging. Programmable switches rely on high-speed memories that, for cost reasons, are present in limited amounts. Moreover, switches have a very strict “allowance” of how many instructions they can execute per packet and what those instructions can do. These restrictions make maintaining data structures other than a hash table on a switch possible but difficult. Even some common operations on hash tables, such as collision management, require some adapting, as we discussed above.

We miss a computing model that can relax those restrictions to a certain extent while still keeping the ability to run logic at line-speed. Supporting such a model would most likely require introducing new hardware onto programmable switches. We believe such a model is possible, but the field of in-network computing is still new. There is not yet a clear list of missing operations from applications beyond networking protocols on which to base new switch hardware capabilities.

**SSD-Application Interface.** A regular SSD makes several decisions such as IO scheduling and page mapping based solely on observing the stream of IO commands it receives [28]. When application logic executes inside the device, it becomes an additional stream of IO commands. The internal and external streams would likely compete with one another. A straightforward way to manage the new internal streams is to pretend they came from outside and to proceed as usual.

An alternative way would be to allow the application logic and the device to interact. Consider again the checkpoint derivation scenario we describe in Section 3.2. It generates new IO requests at a given pace. Depending on the current load on the SSD, the checkpoint stream can be too fast or too slow for the device to handle.

Ideally, we want a means for the device and the application logic to negotiate that pace. There is an opportunity here to establish a communication model that supports this kind of interaction.

**SSD Channel Architecture.** An SSD’s bandwidth and capacity are a result of combining many relatively limited NAND flash *packages* (chips) [27]. A typical arrangement is to group the packages in disjoint sets called *channels*. There are anywhere between 8 to 32 channels in a typical SSD. This arrangement has proved adequate when data pages are transferred in and out of the device.

The introduction of application logic into an SSD, however, can cause data pages to be moved *across packages*—possibly between channels. For instance, we describe in Section 3.2 how an NSP process can read pages from a transaction log and write them, after some manipulation, as pages of a checkpoint. In a traditional SSD architecture, this pattern would consume bandwidth from both the origin and the destination channels. We believe that there may be other package interconnect architectures beyond channels that are more suited to the data movements above. To the best of our knowledge, there has been no study about interconnects that involve architectures other than fixed channels.

**Code-Variant Generation and Selection.** INC and NSP platforms introduce additional hardware heterogeneity to computing platforms. An algorithm implementation that works well on a given switch may not work on a different one, due to architectural differences. The generation and selection of variant implementations is a known problem; it appeared before in domains such as GPUs [36].

To complicate matters, some algorithms are flexible enough to be deployed either on a NIC or on the switch to which it connects. The selection of the most appropriate platform constitutes an optimization step that compounds to the variant selection problem above. Currently, the programmer is responsible for making such choices. There is an opportunity for creating higher-level tooling that would aid in these decisions.

**Runtime Resource Management.** In a regular setting, the operating system mediates every single network and storage IO between applications and devices. The OS does so by interposing itself between the two. What, then, should be the role of the OS when a portion of the applications reside *inside* the device?

One potential solution is to accept that applications may want to access a device directly, without mediation. Such is the premise of Arrakis [31], which tricks an application into interacting with virtual versions of the devices, and redesigns the kernel to provide the expected protections under such assumptions.

## 6 Discussion

Notwithstanding the challenges and opportunities described in the previous section, INC and NSP platforms are a reality. The question arises as to whether they are ready for adoption. In this section, we break this question into a list of sub-questions and answer each one in turn.

**Are there diverse offerings of INC and NSP platforms?** INC can be realized by many different hardware targets, both on switches and NICs. In terms of programmable switches, there are at least three silicon manufacturers producing programmable ASICs: Barefoot (Intel) Tofino [1], Broadcom Trident 4 [5], and Cavium (Marvel) Xpliant [11]. Together, these chips appear in several commercial offerings from companies such as Cisco and Arista. SmartNICs are also widely available, ranging from FPGA-centric accelerators such as Xilinx Alveo [42] to more software-oriented platforms such as Netronome [29]. The offering for programming languages and abstractions is also diverse. A prevalent language is P4 [7], but there are variations such as NPL [4] and even C libraries in some cases [29].

NSP is also a rich environment with many prototyping and commercial platforms available. For instance, SSDs are starting to emerge that introduce application specializations. Samsung has recently announced the KV-SSD, which incorporates Key Value store logic within its firmware [20]. One issue with such offerings is that they add application functionality but do not expose programmability. The interested reader should look

at [33] for an extensive list of specialized SSD and NSP-platforms available at the time of writing.

**What are the advantages of INC and NSP compared to “CPU-centric” alternatives?** While there is no study yet of an exacale platform that resorts to both INC and NSP, the expected benefits are a combination of increased performance, lower energy expenditure, and lower CPU utilization. There are numerous works that provide partial results.

For INC, data manipulations such as the one described in Section 3.1 were studied before [25]. We can expect to see the performance of many typical queries improve by a factor of  $2\times$  by using programmable switches at moderate network speeds. The power consumption in these switches is estimated to be only 12.4% higher than their fixed-function counterparts, and, anecdotally, the switches cost about the same. The operations-per-Watt ratio on some INC platforms such as smartNICs was also studied before [41]. For instance, a variant of the caching scenario we discussed in Section 4 reports a  $17\times$  better power utilization as compared to a regular CPU.

NSP platforms deliver similar benefits. According to [21], the performance of scans and joins can see performance improvements between  $5\times$  and  $47\times$ , the cost of equipping an SSD to support near-storage processing is less than 1% of its total cost, and the energy-efficiency compared to performing those operations on a CPU can be up to  $45\times$  better.

**Are there standards in place to guarantee the portability and longevity of the solutions?** There is a big difference from a standardization point of view between INC and NSP technologies. As mentioned above, INC has broadly embraced  $\mathcal{P}4$  as a programming language. The latest edition of the language,  $\mathcal{P}4_{16}$ , allows different target platforms to express their capabilities. A compiler can generate specific code from a unique source to different, maybe even disparate, devices. These features make the language flexible enough to work on future devices, which can only help its adoption.

In contrast, there is not yet a consensus on how computational storage should be programmed. We even miss a standard around what an open-channel SSD should be, which would arguably be a necessary step.

## 7 Conclusion

In this paper, we reviewed how the evolution of the network and storage stacks have unlocked their computing power. Applications can not only request services from these stacks, but also embed logic into them. We showed that with a careful redesign, algorithms running on INC or NSP platforms could present a reduced amount of data movement, lower CPU utilization, less energy consumption, or a combination of these.

We also discussed several challenges that INC and NSP still face. Despite these limitations, we commented on a current generation of INC- and NSP-enabled devices that are available off-the-shelf. We believe that the emergence of the network and storage stacks as computing elements creates promising ways to scale typical data management computations.

**Acknowledgments.** This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement 683253/GraphInt).

## References

- [1] Barefoot Tofino and Tofino 2 Switches. <https://www.barefootnetworks.com/products/brief-tofino-2/>.
- [2] R. Bifulco, and G. Rétvári. A Survey on the Programmable Data Plane: Abstractions, Architectures, and Open Problems. *HPSR*, June, 2018.
- [3] M. Bjørling, J. González, and P. Bonnet. Lightnvm: The Linux Open-Channel SSD Subsystem *FAST*, February, 2017.
- [4] Broadcom NPL. Network Programming Language. <https://nplang.org/>.



- [5] Broadcom. Broadcom Trident 4. <https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56880-series>.
- [6] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz. Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN. *SIGCOMM CCR*, 43(4):99–110, 2013.
- [7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *ACM SIGCOMM CCR*, 44(3):87–95, 2014.
- [8] R. Budruk, D. Anderson, and E. Solari. PCI Express System Architecture. *Pearson Education*, 2003.
- [9] Y. Cai, S. Ghose, E.F. Haratsch, Y. Luo, and O. Mutlu. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives. *Proc. of the IEEE*, 105(9), 1666–1704, 2017.
- [10] K.L. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz. Directions in Active Networks. *IEEE Comm. Magazine*, 36(10):72–78, 1998.
- [11] Cavium. XPlaint Ethernet Switch Product Family. [www.cavium.com/XPlaint-Ethernet-Switch-Product-Family.html](http://www.cavium.com/XPlaint-Ethernet-Switch-Product-Family.html)
- [12] CCIX Consortium. An Introduction to CCIX. *White Paper*, <https://www.ccixconsortium.com/wp-content/uploads/2019/11/CCIX-White-Paper-Rev111219.pdf>
- [13] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee, and H.-J. Song. A Survey of Flash Translation Layer. *J. of Syst. Archit.*, 55(5–6):332–343, 2009.
- [14] D.D. Sharma. An Introduction to Compute Express Link. *White Paper*, [https://docs.wixstatic.com/ugd/0c1418\\_d9878707bbb7427786b70c3c91d5fbd1.pdf](https://docs.wixstatic.com/ugd/0c1418_d9878707bbb7427786b70c3c91d5fbd1.pdf).
- [15] J. Do, Y.S. Kee, J.M. Pzatel, C. Park, K. Park, and D.J. DeWitt. Query Processing on Smart SSDs: Opportunities and Challenges. *SIGMOD*, June, 2013.
- [16] J. Do, S. Sengupta, and S. Swanson. Programmable Solid-State Storage in Future Could Datacenters. *CACM*, 62(6):54–62, 2019.
- [17] J. Fang, Y.T.B. Mulder, J. Hidders, J. Lee, and H.P. Hofstee. In-Memory Database Acceleration on FPGAs: A Survey. *VLDB Journal*, October, 2019.
- [18] B. Gu, A.S. Yoon, D.H. Bae, I. Jo, J. Lee, and J. Yoon. Biscuit: A Framework for Near-Data Processing of Big Data Workloads. *SIGARCH Comp. Arch. News*, 44(3):153–165.
- [19] Z. Guz, H. Li, A. Shayesteh, and V. Balakrishnan. Performance Characterization of NVMe-over-Fabrics Storage Disaggregation. *ACM Trans. on Storage*, 14(4):1553–3077, 2018.
- [20] Y.-S. Ki. Key Value SSD Explained – Concept, Device, System, and Standard. *SNIA SDC*, September, 2017.
- [21] S. Kim, H. Oh, C. Park, S. Cho, S.-W. Lee, and B. Moon. In-Storage Processing of Database Scans and Joins. *Inf. Sci.*, 327(C):183–200, 2016.
- [22] P. Knebel, D. Berkram, A. Davis, D. Emmot, P. Faraboschi, and G. Gostin. Gen-Z Chipset for Exascale Fabrics. *HotChips*, August, 2019.
- [23] D. Kreutz, F.M.V. Ramos, P.E. Veríssimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proc. of the IEEE*, 103(1):14–76, 2015.
- [24] J. Kwak, S. Lee, K. Park, J. Jeong, and Y.H. Song. Cosmos+ OpenSSD: Rapid Prototype for Flash Storage Systems. *ACM Trans. on Storage*, to appear.
- [25] A. Lerner, R. Russein, and P. Cudré-Mauroux. The Case for Network Accelerated Query Processing. *CIDR*, January, 2019.
- [26] A. Lerner, J. Kwak, S. Lee, K. Park, Y.H. Song, and P. Cudré-Mauroux. It Takes Two: Instrumenting the Interaction between In-Memory Databases and Solid-State Drives. *CIDR*, January, 2020.
- [27] R. Micheloni, A. Marelli, and S. Eshghi. Inside Solid State Drives (SSDs). *Springer*, 2012.



- [28] E.H. Nam, B. S. J. Kim, H. Eom, and S. L. Min. Ozone (O3): An Out-of-Order Flash Memory Controller Architecture. *IEEE Transactions on Computers*, 60(5):653–666, 2011.
- [29] Netronome. Agilio CX SmartNICs. <https://www.netronome.com/products/agilio-cx/>
- [30] J. Ouyang, S. Lin, S. Jiang, Z. Hou, Y. Wang, and Y. Wang. SDF: Software-Defined Flash for Web-Scale Internet Storage Systems. *ASPLOS*, 2014.
- [31] S. Peter, J. Li, I. Zhang, D.R.K. Ports, D. Woos, A. Krishnamurthy, T. Anderson, and T. Roscoe. Arrakis: The Operating System Is the Control Plane. *ACM TOCS*, 33(4), 2015.
- [32] I.L. Picoli, P. Bonnet, and P. Tözün. LSM Management on Computational Storage. *DaMoN*, July, 2019.
- [33] I.L. Picoli, N. Hedam, P. Bonnet, and P. Tözün. Open-Channel SSD (What Is It Good For). *CIDR*, January, 2020
- [34] D.R.K. Ports, and J. Nelson. When Should The Network Be The Computer. *HotOS*, May, 2019.
- [35] E. Riedel, C. Faloutsos, G.A. Gibson, and D. Nagle. Active Disks for Large-Scale Data Processing. *IEEE Computer*, 34(6):68–74, 2001.
- [36] V. Rosenfeld, M. Heimel, C. Viebig, and V. Markl. The Operator Variant Selection Problem on Heterogeneous Hardware. *ADMS*, August, 2015.
- [37] Z. Ruan, T. He, and J. Cong. INSIDER: Designing In-Storage Computing System for Emerging High-Performance Drive. *Usenix ATC*, July, 2019.
- [38] A. Sivaraman, A. Cheung, M. Budiu, C. Kim, M. Alizadeh, H. Balakrishnan, G. Varghese, N. McKeown, and S. Licking. Packet transactions: High-level programming for line-rate switches. *SIGCOMM*, August, 2016.
- [39] J. Stuecheli, B. Blaner, C.R. Johns, and M.S. Siegel. CAPI: A Coherent Accelerator Processor Interface. *IBM Journal of Research and Development*, 59(1):1–7, 2015.
- [40] J. Teubner, and L. Woods. Data Processing on FPGAs. *Morgan & Claypool Publishers*, 2013.
- [41] Y. Tokusashi, H. Matsutani, and N. Zilberman. LaKe: The Power of In-Network Computing. *ReConFig*, December, 2018.
- [42] Xilinx. ALVEO Adaptable Accelerator Cards for Data Center Workloads. <https://www.xilinx.com/content/xilinx/en/products/boards-and-kits/alveo.html>
- [43] L. Woods, Z. István, and G. Alonso. Ibex: An Intelligent Storage Engine with Support for Advanced SQL Offloading. *Proc. of the VLDB*, 7(11):963–974.
- [44] N. Zilberman, Y. Audzevich, G.A. Covington, and A.W. Moore. NetFPGA SUME: Towards 100 Gbps as Research Commodity. *IEEE Micro*, 34(5):32–41, 2014.



## Data Engineering

It's FREE to join!

# TCDE

[tab.computer.org/tcde/](http://tab.computer.org/tcde/)

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

## Join TCDE via Online or Fax

**ONLINE:** Follow the instructions on this page:

[www.computer.org/portal/web/tandc/joinatc](http://www.computer.org/portal/web/tandc/joinatc)

**FAX:** Complete your details and fax this form to **+61-7-3365 3248**

Name   
IEEE Member #   
Mailing Address   
  
Country   
Email   
Phone

### TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

### Membership Questions?

#### Xiaoyong Du

Key Laboratory of Data Engineering and Knowledge Engineering  
Renmin University of China  
Beijing 100872, China  
[duyong@ruc.edu.cn](mailto:duyong@ruc.edu.cn)

### TCDE Chair

#### Xiaofang Zhou

School of Information Technology and Electrical Engineering  
The University of Queensland  
Brisbane, QLD 4072, Australia  
[zxf@uq.edu.au](mailto:zxf@uq.edu.au)



IEEE Computer Society  
10662 Los Vaqueros Circle  
Los Alamitos, CA 90720-1314

Non-profit Org.  
U.S. Postage  
PAID  
Los Alamitos, CA  
Permit 1398