A What-if Engine for Cost-based MapReduce Optimization

Herodotos Herodotou
Microsoft Research

Shivnath Babu
Duke University

Abstract

The Starfish project at Duke University aims to provide MapReduce users and applications with good performance automatically, without any need on their part to understand and manipulate the numerous tuning knobs in a MapReduce system. This paper describes the What-if Engine, an indispensable component in Starfish, which serves a similar purpose as a costing engine used by the query optimizer in a Database system. We discuss the problem and challenges addressed by the What-if Engine. We also discuss the techniques used by the What-if Engine and the design decisions that led us to these techniques.

1 Introduction

Modern organizations are collecting data ("big data") at an unprecedented scale. MapReduce systems like Hadoop have emerged to help these organizations store large datasets and run analytics on the data to generate useful insights [15]. Getting good performance from a MapReduce system can be a nontrivial exercise due to the large number of tuning knobs present in the system. Practitioners of big data analytics like business analysts, computational scientists, and systems researchers usually lack the expertise to tune these knobs.

The Starfish project at Duke University aims to provide MapReduce users and applications with good performance automatically, without any need on their part to understand and manipulate the numerous tuning knobs. Starfish uses a "profile-predict-optimize" approach:

1. *Profile*: Starfish observes and records the actual run-time behavior of MapReduce workloads executing on the system. This part is handled by the *Profiler* in Starfish.
2. *Predict*: Starfish analyzes and learns from these observations, and reasons about hypothetical tuning choices. This part is handled by the *What-if Engine* in Starfish.
3. *Optimize*: *Optimizers* in Starfish choose appropriate settings for the various tuning knobs in order to improve workload performance along one or more dimensions (e.g., completion time, resource utilization, pay-as-you-go monetary costs on cloud platforms).

Optimizers have been developed or are being developed in Starfish for setting a number of tuning knobs including: (i) choice of query execution plans for SQL-like queries run over MapReduce, (ii) degree of task-level parallelism for MapReduce jobs, (iii) physical design choices such as partitioning, ordering, and compression,

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

(iv) provisioning of resources on the cluster, (v) selection of the virtual machine type on cloud platforms like Amazon Web Services, (vi) choice of job and task scheduling policies, and others.

The effectiveness of the Optimizers in finding close-to-optimal settings of the various tuning knobs depends heavily on the accuracy of the What-if Engine in predicting performance for different settings of these knobs. Thus, the What-if Engine is a critical component in a big data ecosystem. Unfortunately, the importance of the What-if Engine is often underestimated and undervalued.

In this paper, we focus on the "predict" piece of the "profile-predict-optimize" approach and describe Starfish's What-if Engine. We will first describe the overall problem and challenges that the What-if Engine addresses, and then outline the approach taken to solve them. We also report an experimental evaluation of the What-if Engine.

2 Design of Starfish's What-if Engine

2.1 Problem and Design Decisions

Consider a MapReduce job $j = \langle p, d, r, c \rangle$ that runs program p on input data d and cluster resources r using configuration parameter settings c. Job j's performance can be represented as:

$$perf = F_1(p, d, r, c) \tag{1}$$

Here:

- *perf* is some performance metric (e.g., execution time) of interest for jobs that is captured by the *performance model* F_1.
- The MapReduce program p is written directly in terms of map and reduce functions by a user or generated from a higher-level query specified in languages like HiveQL and Pig Latin [15].
- The input dataset d is represented by information such as d's size, the block layout of files that comprise d in the distributed file-system used by MapReduce, and whether d is stored compressed.
- The cluster resources r are represented by information such as the number of nodes and node types in r, the number of map and reduce task execution slots per node, and the maximum memory available per task slot.
- The job configuration c includes parameters such as the number of map and reduce tasks, the partitioning function, settings for memory-related parameters, the use of compression, and the use of the combine function [15].

Tuning knobs to optimize the performance of the MapReduce job $j = \langle p, d, r, c \rangle$ are present in each one of p, d, r, and c. For example, if p represents a join operation, then a map-side join or a reduce-side join can be done [1]. The data layout can be tuned by using row-based or column-based layouts [8]. The cluster resources can be tuned by choosing appropriate virtual machine types [6]. The job configuration can be tuned by adjusting the number of reduce tasks [5].

If the function F_1 from Equation 1 can be estimated accurately by the *What-if Engine*, then Starfish's Optimizers can be very effective at finding optimal settings for the above tuning knobs. However, accurate modeling of F_1 has to overcome many challenges that arise in distributed processing such as task parallelism, scheduling, and interactions among the large number of tuning knobs involved.

Furthermore, MapReduce jobs usually run as part of a MapReduce workflow W, where W is a directed acyclic graph (DAG) G_W that represents a set of MapReduce jobs and their dataflow dependencies. Consider a MapReduce workflow W that runs the programs $\{p_i\}$ from the corresponding jobs j_i in G_W on input base datasets

	What-if Questions on MapReduce Job Execution
WIF_1	How will the execution time of job j change if I increase the number of reduce tasks from the current value of 20 to 40?
WIF_2	What is the new estimated execution time of job j if 5 more nodes are added to the cluster, bringing the total to 20 nodes?
WIF_3	How will the execution time of job j change when I execute j on the production cluster instead of the development cluster, and the input data size increases by 60%?
WIF_4	What is the estimated execution time of job j if I execute j on a new cluster with 10 EC2 nodes of type m1.large rather than the current in-house cluster?

Table 1: Example questions the What-if Engine can answer.

$\{b_i\}$ and cluster resources r using configuration parameter settings $\{c_i\}$. W's performance can be represented as:

$$perf = F_2(\{p_i\}, \{b_i\}, r, \{c_i\}) \tag{2}$$

Building the *performance model* F_2 to predict workflow performance brings in additional complications due to concurrent execution of independent jobs in W on the cluster resources r.

It is quite clear that performance prediction is a very hard problem. At the same time, it is a problem that needs a reasonable solution if we want to generate good settings for tuning knobs automatically. One of the contributions we have made in Starfish is to identify an abstraction of the prediction problem that strikes a good balance between being applicable and admitting to good solutions. This abstraction can be characterized by the following three design decisions:

1. Address *relative* prediction problems instead of *absolute* prediction problems.
2. Develop *profiles* that capture the salient aspects of MapReduce execution. Use profiles consistently to represent the observed performance of actual MapReduce jobs that run as well as the predicted performance of hypothetical MapReduce jobs.
3. Break the overall prediction problem into composable components and attack each component with the best modeling technique available among analytical models, black-box models, and simulation models.

2.2 Relative What-if Questions

Starfish's What-if Engine can answer any *what-if question* of the following general form:[1]

> Given the profile of a job $j = \langle p, d_1, r_1, c_1 \rangle$ that runs a MapReduce program p over input data d_1 and cluster resources r_1 using configuration c_1, what will the performance of program p be if p is run over input data d_2 and cluster resources r_2 using configuration c_2? That is, how will job $j' = \langle p, d_2, r_2, c_2 \rangle$ perform?

Note the relative format in which the what-if question is specified. Being relative here means that the What-if Engine is given complete information about the execution of the job j, and then asked about how a related, but different, job j' will perform. Intuitively, relative what-if questions are easier to answer than absolute questions. In the relative case, only an estimate of the change in performance is needed; whereas the absolute case as represented by Equation 1 needs a prediction starting from scratch with limited input information provided.

Table 1 lists several interesting what-if questions that users or applications can express directly to the Starfish What-if Engine. For example, consider question WIF_1 from Table 1. Here, the performance of a MapReduce

[1] For simplicity of presentation, this specification considers a single MapReduce job instead of a workflow.

Algorithm for predicting MapReduce workflow performance

Input: Profile of jobs in workflow, Cluster resources, Base dataset properties, Configuration settings
Output: Prediction for the MapReduce workflow performance

For each (job profile in workflow in topological sort order) {
    Estimate the virtual job profile for the hypothetical job (Sections 3.1, 3.2, and 3.3);
    Simulate the job execution on the cluster resources (Section 3.4);
    Estimate the data properties of the hypothetical derived dataset(s) and the overall job performance;
}

Figure 1: Overall process used by the What-if Engine to predict the performance of a MapReduce workflow.

job $j = \langle p, d, r, c \rangle$ is known when 20 reduce tasks are used to run j. The number of reduce tasks is one of the job configuration parameters in c. WIF_1 asks for an estimate of the execution time of a hypothetical job $j' = \langle p, d, r, c' \rangle$ whose configuration c' is the same as c except that c' specifies using 40 reduce tasks. The MapReduce program p, input data d, and cluster resources r remain unchanged in j'.

As indicated in Table 1, the What-if Engine can answer questions on real and hypothetical input data as well as cluster resources. Furthermore, as we will describe in Section 3, the What-if Engine can answer what-if questions for MapReduce workflows composed of multiple jobs that exhibit producer-consumer relationships or are run concurrently.

2.3 Profiles

Note that a relative what-if question needs the profile of a MapReduce job as input. A profile consists of fields that together form a concise summary of the job's execution. These fields are partitioned into four categories:[2]

- *Cost fields:* These fields capture information about execution time at the level of tasks as well as phases within the tasks for a MapReduce job execution. Some example fields are the execution time of the Collect and Spill phases of a map task.
- *Cost statistics fields:* These fields capture statistical information about execution time for a MapReduce job—e.g., the average time to read a record from the distributed file-system, or the average time to execute the map function per input record—that is expected to remain unchanged across different executions of the job unless the CPU and/or I/O resources available per node change.
- *Dataflow fields:* These fields capture information about the amount of data, both in terms of bytes as well as records (key-value pairs), flowing through the different tasks and phases of a MapReduce job execution. Some example fields are the number of map output records and the amount of bytes shuffled among the map and reduce tasks.
- *Dataflow statistics fields:* These fields capture statistical information about the dataflow—e.g., the average number of records output by map tasks per input record (the Map selectivity) or the compression ratio of the map output—that is expected to remain unchanged across different executions of the MapReduce job unless the data distribution in the input dataset changes significantly across these executions.

3 MapReduce Performance Prediction

Figure 1 shows the What-if Engine's overall process for predicting the performance of a MapReduce workflow. The DAG of job profiles in the workflow is traversed in topological sort order to ensure that the What-if Engine

[2]Due to space constraints, we refer the reader to [4] for the full listing and description of fields in all four categories.

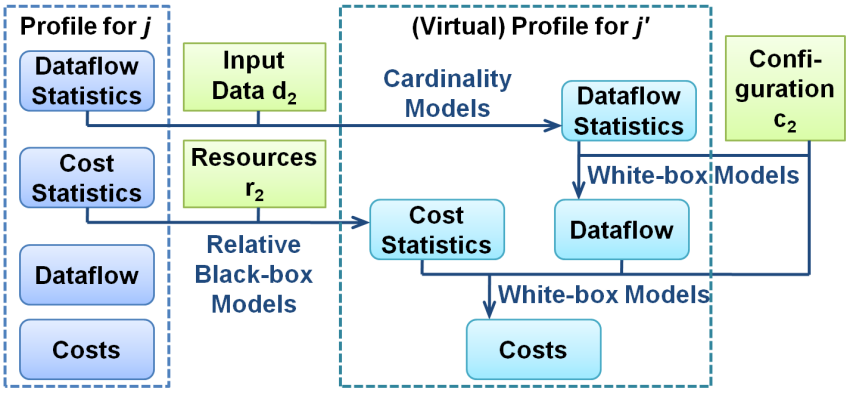

Figure 2: Overall process used by the What-if Engine to estimate a virtual job profile.

respects the dataflow dependencies among the jobs. For each job profile, a *virtual job profile* is estimated for the new hypothetical job j' based on the new configuration settings, the cluster resources, and the properties of the data processed by j' (Sections 3.1, 3.2, and 3.3). The virtual profile is then used to simulate the execution of j' on the (perhaps hypothetical) cluster (Section 3.4). Finally, the simulated execution is used to estimate the data properties for the derived dataset(s) produced by j' as well as the overall performance of j'. The overall workflow performance is predicted by combining the performance predictions for each job in the workflow.

The process of virtual profile estimation forms the foundation on which Starfish's ability to answer what-if questions is based. Specifically, given the profile of a job $j = \langle p, d_1, r_1, c_1 \rangle$ and the properties of input data d_2, cluster resources r_2, and configuration parameter settings c_2 of a hypothetical job $j' = \langle p, d_2, r_2, c_2 \rangle$, the virtual profile of j' has to be estimated. Our solution for virtual profile estimation is based on a mix of black-box and white-box models. The overall estimation process has been broken down into smaller steps as shown in Figure 2, and a suitable modeling technique was picked for each step. These smaller steps correspond to the four categories of fields in a job profile. We use conventional cardinality (white-box) models from database query optimization to estimate dataflow statistics fields (described in Section 3.1), relative black-box models to estimate the cost statistics fields (described in Section 3.2), and new analytical (white-box) models that we developed to estimate the dataflow fields, and in turn, the cost fields (described in Section 3.3).

3.1 Cardinality Models to Estimate Dataflow Statistics Fields

Database query optimizers keep fine-grained, data-level statistics such as histograms to estimate the dataflow in execution plans for declarative queries. However, MapReduce frameworks lack the declarative query semantics and structured data representations of Database systems. Thus, the common case in the What-if Engine is to not have detailed statistical information about the input data d_2 in the hypothetical job j'. By default, the What-if Engine makes a *dataflow proportionality assumption* which says that the logical dataflow sizes through the job's phases are proportional to the input data size. It follows from this assumption that the dataflow statistics fields in the virtual profile of j' will be the same as those in the profile of job j given as input. When additional information is available, the What-if Engine allows the default assumption to be overridden by providing dataflow statistics fields of the virtual profile directly as input.

3.2 Relative Black-box Models to Estimate Cost Statistics Fields

Clusters with identical resources will have the same CPU and I/O (local and remote) costs. Thus, if the cluster resource properties of r_1 are the same as those of r_2 in the what-if question, then the cost statistics fields in the virtual profile of the hypothetical job j' can be copied directly from the profile of job j given as input. This copying cannot be used when $r_1 \neq r_2$, in particular, when job j' will run on a *target cluster* containing nodes with a different type from the *source cluster* where job j was run to collect the profile that was given as input.

The technique we use when $r_1 \neq r_2$ is based on a *relative* black-box model $M_{src \to tgt}$ that can predict the cost statistics fields CS_{tgt} in the virtual profile for the target cluster from the cost statistics fields CS_{src} in the job profile for the source cluster.

$$CS_{tgt} = M_{src \to tgt}(CS_{src}) \tag{3}$$

Generating training samples for the $M_{src \to tgt}$ model: Let r_{src} and r_{tgt} denote the cluster resources respectively for the source and target clusters. Suppose some MapReduce program p is run on input data d and configuration parameter settings c on both the source and target clusters. That is, we run the two jobs $j_{src} = \langle p, d, r_{src}, c \rangle$ and $j_{tgt} = \langle p, d, r_{tgt}, c \rangle$. From these runs, we can generate the profiles for each individual task in these two jobs by direct measurement. Therefore, from the i^{th} task in these two jobs,[3] we get a training sample $\langle \mathrm{CS}_{src}^{(i)}, \mathrm{CS}_{tgt}^{(i)}, \rangle$ for the $M_{src \to tgt}$ model.

The above training samples were generated by running a related pair of jobs j_{src} and j_{tgt} that have the same MapReduce program p, input data d, and configuration parameter settings c. We can generate a complete set of training samples by using a *training benchmark* containing jobs with different $\langle p, d, c \rangle$ combinations. Selecting an appropriate training benchmark is nontrivial because the two main requirements of effective black-box modeling have to be satisfied. First, for accurate prediction, the training samples must have good coverage of the prediction space. Second, for efficiency, the time to generate the complete set of training samples must be small. Both actual and synthetic jobs can be used to generate the training samples. Different methods to generate the training benchmark are described in [4].

Learning all the $M_{src \to tgt}$ models needed: It is important to note that the training benchmark has to be run only once (or with a few repetitions) per target cluster resource; giving only a linear number of benchmark runs, and not quadratic as one might expect from the relative nature of the $M_{src \to tgt}$ models. The training samples for each source-to-target cluster pair are available from these runs. For example, to address question WIF_3 from Table 1 (i.e., planning for workload transition from a development cluster to production), one run each of the training benchmark on the development and the production cluster will suffice.

Once the training samples are generated, there are many *supervised learning* techniques available for generating the black-box model in Equation 3. Since cost statistics are real-valued, we selected the *M5 Tree Model* [12]. An M5 Tree Model first builds a regression-tree using a typical decision-tree induction algorithm. Then, the tree goes through pruning and smoothing phases to generate a linear regression model for each leaf of the tree.

In summary, given the cost statistics fields CS_{src} in the job profile for the source cluster r_{src}, the relative black-box model $M_{src \to tgt}$ is used to predict the cost statistics fields CS_{tgt} in the virtual profile for the target cluster r_{tgt} when $r_{src} \neq r_{tgt}$.

3.3 Analytical Models to Estimate Dataflow and Cost Fields

The What-if Engine uses a detailed set of analytical (white-box) models to calculate the dataflow fields in the virtual profile given (i) the dataflow statistics fields estimated above, and (ii) the configuration parameter settings c_2 in the hypothetical job j'. These models give good accuracy by capturing the subtleties of MapReduce job execution at the fine granularity of phases within map and reduce tasks. The current models were developed for Hadoop, but the overall approach applies to any MapReduce implementation. A second set of analytical models combines the estimated cost statistics and dataflow fields in order to estimate the cost fields in the virtual profile. The full set of models is described in [4].

[3] Ensuring that the tasks have the same input data d and configuration parameter settings c ensures that there is a one-to-one correspondence between the tasks in these two jobs.

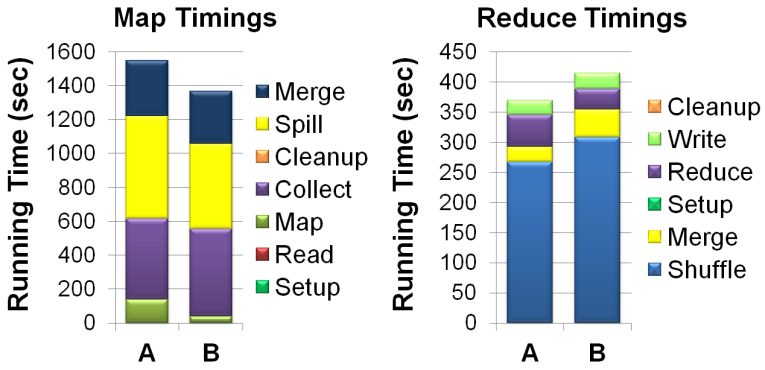

Figure 3: Map and reduce time breakdown for Word Co-occurrence jobs from (A) an actual run and (B) as predicted by the What-if Engine.

3.4 Simulating the Execution of a MapReduce Workload

The virtual job profile contains detailed dataflow and cost information estimated at the task and phase level for the hypothetical job j'. The What-if Engine uses a *Task Scheduler Simulator*, along with the job profiles and information on the cluster resources, to simulate the scheduling and execution of map and reduce tasks in j' (recall Figure 1). The Task Scheduler Simulator is a pluggable component that takes into account the dataflow and resource dependencies among the MapReduce jobs in a workflow. Our current implementation is a lightweight discrete event simulation of Hadoop's default FIFO scheduler.

The final output—after all jobs have been simulated—is a description of the complete (hypothetical) workflow execution in the cluster. The desired answer to the what-if question—e.g., predicted workflow running time, amount of local I/O, or a visualization of the task execution timeline—can be computed from the workflow's simulated execution.

4 Accuracy of What-if Analysis

We report some empirical results on the accuracy of the What-if Engine in predicting the overall execution time of different MapReduce workflows. In our evaluation, we used Hadoop clusters running on Amazon EC2 nodes of various sizes and node types. Figure 3 compares the actual MapReduce task and phase timings with the corresponding predictions from the What-if Engine for a Word-Cooccurrence MapReduce program run over 10GB of real data obtained from Wikipedia. Note that even though the predicted timings are slightly different from the actual ones, the relative percentage of time spent in each MapReduce phase is captured fairly accurately.

We also ran multiple MapReduce workflows under 40 different configuration settings. We then asked the What-if Engine to predict the job execution time for each setting. Figures 4(a)-(c) show scatter plots of the actual and predicted times for these 40 jobs. Observe the proportional correspondence between the actual and predicted times, and the clear identification of settings with the top-k best and worst performance (indicated by the green and red dotted circles respectively).

The fairly uniform gap between the actual and predicted timings in Figure 4 is due to inaccuracies in the profiles given as input to the What-if Engine. These inaccuracies are caused by the overhead of the Java profiling technology that Starfish uses currently. The gap is largest when the MapReduce program runs under CPU contention—which was the case when the Word Co-occurrence job was profiled. Unlike the case of Word Co-occurrence in Figure 4(a), the predicted values in Figures 4(b) and 4(c) are closer to the actual values; indicating that the profiling overhead is reflected less in the costs captured in the job profile. We expect to close this gap using commercial Java profiling technology.

Another common use case we considered in our evaluation is the presence of a development cluster, and the need to stage jobs from the development cluster to the production cluster. In our evaluation, we used a

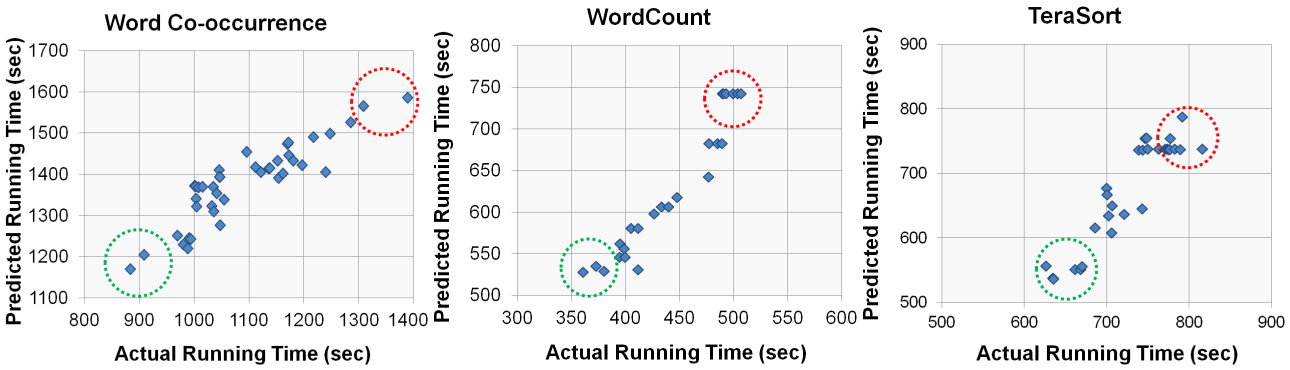

Figure 4: Actual Vs. predicted running times for (a) Word Co-occurrence, (b) WordCount, and (c) TeraSort jobs running with different configuration parameter settings.

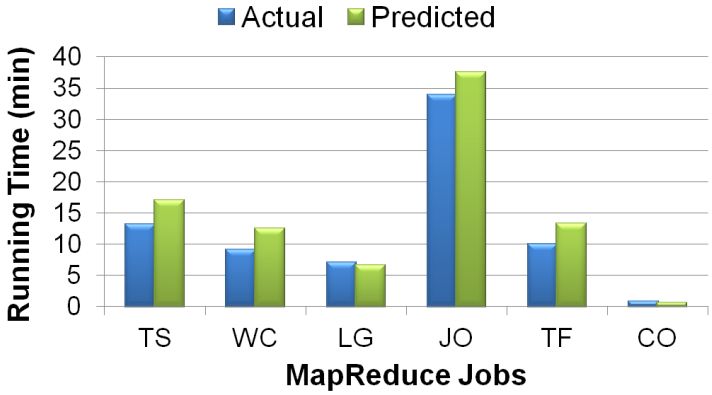

Figure 5: Actual and predicted running times for MapReduce jobs when run on the production cluster. The predictions used job profiles obtained from the development cluster.

10-node Hadoop cluster with m1.large EC2 nodes (4 EC2 units CPU[4], 7.5 GB memory, 850 GB storage) as the development cluster, and a 30-node Hadoop cluster with m1.xlarge EC2 nodes (8 EC2 units CPU, 15 GB memory, 1690 GB storage) as the production one. We profiled a number of MapReduce programs on the development cluster. We then executed the MapReduce programs on the production cluster using three times as much data as used in the development cluster.

Figure 5 shows the actual and predicted running times for each job when run on the production cluster. The What-if Engine used job profiles obtained from the development cluster for making the predictions. Apart from the overall running time, the What-if Engine can also predict several other aspects of the job execution like the amount of I/O and network traffic, the running time and scheduling of individual tasks, as well as data and computational skew.

Overall, the What-if Engine is capable of capturing the performance trends when varying the configuration parameter settings or the cluster resources.

5 Related Work

MapReduce is emerging rapidly as a viable competitor to existing systems for big data analytics, even though it currently trails existing systems in peak query performance [11]. When users are asked to tune the performance of their MapReduce workloads, they have to rely on their experience, knowledge of the data being processed, and rules of thumb from human experts or tuning manuals to complete the task [15]. Automated approaches for such

[4]One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz processor.

performance tuning are based typically on white-box models, black-box models, or simulation. Our approach is unique as it using a carefully-designed combination of all three aforementioned techniques to achieve the task.

White-box models are created by human experts who have detailed knowledge of how workload execution is affected by the properties of cluster resources, input data, system configuration, and scheduling policies. If the models are developed correctly, then the predictions from them will be accurate; as is the case in the *epiC* system, which supports System-R-style join ordering in Hive using white-box modeling [16]. However, white-box models can quickly become inaccurate if they have to capture the impact of evolving features such as hardware properties [6].

There has been considerable interest recently in using black-box models like regression trees to build workload performance predictors [2]. These models can be trained automatically from samples of system behavior, and retrained when major changes happen. However, these models are only as good as the predictive behavior of the independent variables they use and how well the training samples cover the prediction space. Relative (fitness) modeling is a black-box approach proposed recently for complex storage devices [9]. These models drastically simplify the model-learning process as long as actual workload performance can be measured on the devices. The What-if Engine borrows ideas from relative modeling and applies them to MapReduce clusters.

Simulation is often a valid alternative to modeling, but developing accurate simulators faces many of the same challenges as analytical white-box models. Some discrete event simulators have been proposed for Hadoop (e.g., *Mumak* [13] and *MRPerf* [14]), but none of them meet the needs of the What-if Engine. Mumak needs a job execution trace as input, and cannot simulate job execution with a different cluster size, network topology, or even different numbers of map or reduce tasks from what the execution trace contains. MRPerf is able to simulate job execution at the task level like our What-if Engine. However, MRPerf uses an external network simulator to simulate the data transfers and communication among the cluster nodes; which leads to a per-job simulation time on the order of minutes. Such a high simulation overhead prohibits MRPerf's use by a cost-based optimizer that needs 100-1000s of what-if calls done over different configuration parameter settings.

A MapReduce program has semantics similar to a *Select-Project-Aggregate* (SPA) in SQL with User-defined functions (UDFs) for the selection and projection (map) as well as the aggregation (reduce). *HadoopToSQL* and *Manimal* perform static analysis of MapReduce programs in order to extract declarative constructs like filters and projections, which are then used for database-style optimizations [3, 7]. Manimal does not perform profiling, what-if analysis, or cost-based optimization; it uses rule-based optimization instead. *MRShare* performs multi-query optimization by running multiple SPA programs in a single MapReduce job [10]. MRShare proposes a (simplified) cost model for this application. Overall, previous work related to MapReduce job optimization targets semantic optimizations for MapReduce programs and is either missing or lacks comprehensive profiling and what-if analysis.

6 Summary

In this paper, we described the design and implementation of the Starfish What-if Engine. Such a What-if Engine is an indispensable component of any optimizer for MapReduce systems, just like a costing engine is for a query optimizer in a Database system. The uses of the What-if Engine go beyond optimization: it may be used by physical design tools for deciding data layouts and partitioning schemes; it may be used as part of a simulator that helps make critical design decisions during a Hadoop setup—like the size of the cluster, the network topology, and the node compute capacity; or it may help make administrative decisions—like how to allocate resources effectively or which scheduling algorithm to use.

References

[1] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A Comparison of Join Algorithms for Log Processing in MapReduce. In *Proc. of the 2010 ACM SIGMOD Intl. Conf. on Management of Data*, pages 975–986. ACM, 2010.

[2] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson. Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters. In *Proc. of the 1st USENIX Conf. on Hot Topics in Cloud Computing*. USENIX Association, 2009.

[3] M. J. Cafarella and C. Ré. Manimal: Relational Optimization for Data-Intensive Programs. In *Proc. of the 13th Intl. Workshop on the Web and Databases*, pages 10:1–10:6. ACM, 2010.

[4] H. Herodotou. *Automatic Tuning of Data-Intensive Analytical Workloads*. PhD thesis, Duke University, May 2012.

[5] H. Herodotou and S. Babu. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. *Proc. of the VLDB Endowment*, 4(11):1111–1122, 2011.

[6] H. Herodotou, F. Dong, and S. Babu. No One (cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics. In *Proc. of the 2nd Symposium on Cloud Computing*. ACM, 2011.

[7] M.-Y. Iu and W. Zwaenepoel. HadoopToSQL: A MapReduce Query Optimizer. In *Proc. of the 5th European Conf. on Computer Systems*, pages 251–264. ACM, 2010.

[8] A. Jindal, J.-A. Quian-Ruiz, and J. Dittrich. Trojan Data Layouts: Right Shoes for a Running Elephant. In *Proc. of the 2nd Symposium on Cloud Computing*. ACM, 2011.

[9] M. Mesnier, M. Wachs, R. Sambasivan, A. Zheng, and G. Ganger. Modeling the Relative Fitness of Storage. *SIGMETRICS*, 35(1):37–48, 2007.

[10] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas. MRShare: Sharing Across Multiple Queries in MapReduce. *Proc. of the VLDB Endowment*, 3(1):494–505, 2010.

[11] A. Pavlo, E. Paulson, A. Rasin, D. Abadi, D. DeWitt, S. Madden, and M. Stonebraker. A Comparison of Approaches to Large-Scale Data Analysis. In *Proc. of the 2009 ACM SIGMOD Intl. Conf. on Management of Data*, pages 165–178. ACM, 2009.

[12] R. J. Quinlan. Learning with Continuous Classes. In *Proc. of the 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348. Springer Berlin / Heidelberg, 1992.

[13] H. Tang. *Mumak: Map-Reduce Simulator*, 2009. `https://issues.apache.org/jira/browse/MAPREDUCE-728`.

[14] G. Wang, A. Butt, P. Pandey, and K. Gupta. A Simulation Approach to Evaluating Design Decisions in MapReduce Setups. In *Proc. of the IEEE Intl. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 1–11. IEEE, 2009.

[15] T. White. *Hadoop: The Definitive Guide*. Yahoo! Press, 2010.

[16] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi. Query Optimization for Massively Parallel Data Processing. In *Proc. of the 2nd Symposium on Cloud Computing*. ACM, 2011.