Towards a Scalable Infrastructure for Advanced E-Services

Michael Carey and the Propel Platform Team*
Propel
1010 Rincon Circle
San Jose, CA 95131
www.propel.com

Abstract

In this paper, we provide a brief overview of the Propel Distributed Services Platform, the infrastructure component of the recently announced Propel Commerce System. The Propel Distributed Services Platform is Internet infrastructure software that provides an integrated set of core services for developers of mission-critical, data-centric, e-business applications. It is designed to provide the foundation for future as well as current Propel product offerings. We list the current problems and key requirements that have driven the design of the Propel Platform and then describe the resulting system architecture and highlight its feature set.

1 Introduction

At Propel, we are currently putting the finishing touches on our initial product offering, the Propel Commerce System. Figure 1 gives a high-level view of the overall system. The Propel Commerce System is a full-featured e-commerce software product made up of three key elements: The *Propel E-Commerce Suite* is designed to support customizable retail Web sites for multi-channel (e.g., Internet, physical stores, catalog) retailers for whom integration with existing back-office systems is essential. The *Propel Interaction Designs* top off the E-Commerce Suite with a set of pre-tested design patterns, JSP templates, and tools for use by shoppers, merchandisers, customer support staff, and e-commerce Web site administrators. The *Propel Distributed Services Platform*, an integrated set of core services that provides a scalable, fault-tolerant infrastructure for e-business applications, is the topic of the remainder of this paper.

2 Building E-Services Today

The majority of e-business applications today are based on a variant of the three-tier Web site architecture depicted in Figure 2. (Not shown are the hardware load balancers or firewall boxes that are also part of an e-business Web site.) End users interact with the e-business application from their Web browsers via http requests.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Nicolas Adiba, Michael Blow, Michael Carey, Steven Kirsch, David Li, James Lucky, Sambavi Muthukrishnan, Ivan Oprencak, Rajendra Panwar, Runping Qi, David Rieber, Mary Roth, John Shafer, Brian Sterling, Tolga Urhan, Bert Van der Linden, Brian Vickery, Dan Wineman, and Kuan Yee.

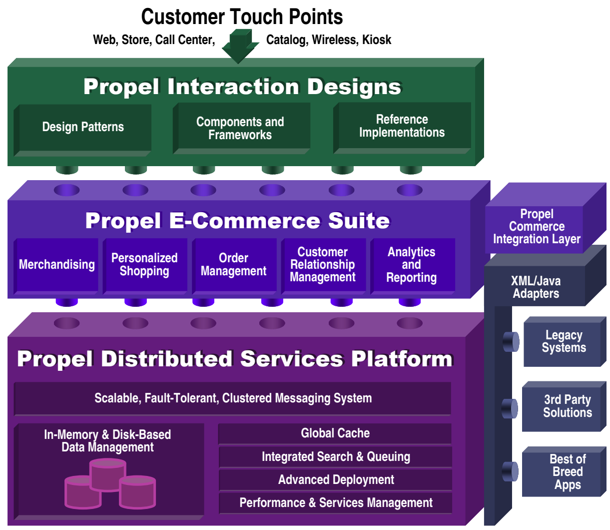

Figure 1: The Propel Commerce System.

These requests are fielded by an array of Web servers that can serve static pages and route requests to the application server tier. Load balancing is usually accomplished across the Web server tier via a hardware load balancer made by a vendor such as Cisco or Alteon. The application server tier hosts the application's business logic, which is often replicated across a set of application servers for scalability. Load balancing at this level is typically accomplished via Web server plug-ins that direct requests initially to any application server and then perform "sticky routing" to direct subsequent requests for the same session back to that server. The data server tier contains the data management facilities required to support the application, which can often include a database management system, a search engine (for full-text searching), a persistent message queuing product, and connections to back office systems such as the enterprise's ERP system, existing data warehouse, and other legacy data systems.

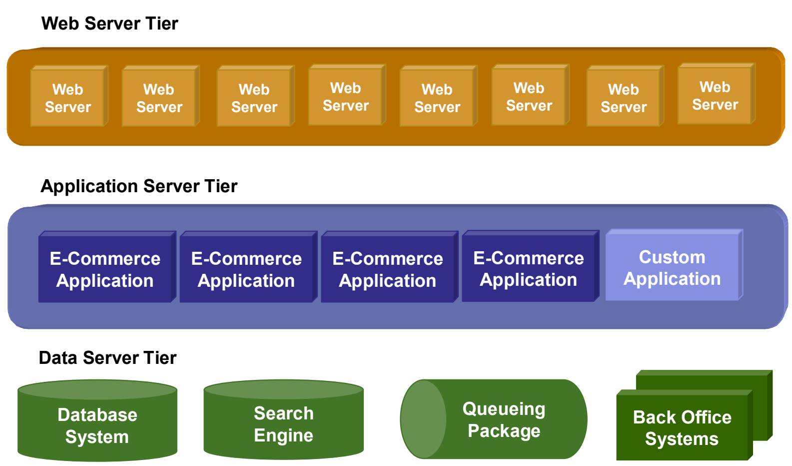

Figure 2: Typical Three-Tier Web Site Architecture.

The standard three-tier Web site architecture, while workable, has a number of limitations. A few of its more significant limitations are:

1. Scalability is provided at the Web server tier by adding Web servers, and (to a somewhat lesser extent) at the application server tier by adding more application servers running copies of the application. However, scalability is limited by the data server tier. There, the application is at the mercy of the individual components' (database system, search engine, queuing system) abilities to scale. When it is even possible, scaling at the data server tier often requires rolling in expensive, high-end SMP systems and software.

2. The e-commerce application developer is required to programmatically piece together data drawn from the various data sources on the data server tier. For example, to perform tasks that require access to database data, full-text search, and queue entries, the developer must use three different APIs, hand-code the equivalent of query execution plans, and utilize careful programming techniques to try and avoid cross-source data consistency problems.

3. For application-level scalability, the e-commerce application developer often ends up having to statically partition their data into multiple databases at the application level (e.g., one database for current auctions, a separate database for recent auctions). Again, this can result in the developer having to hand-code query execution plans when the need arises to operate on data that crosses partitioned database boundaries.

4. Managing the system's configuration and day-to-day operation, as well as doing performance tuning, is tricky at the data server tier. In addition to managing the Web server and application server tiers, these tasks all require the system administrator to deal with the administration and monitoring aspects of a number of disparate data systems.

3 E-Service Software Infrastructure Requirements

The Propel Distributed Services Platform has been developed to address the shortcomings of these existing architectures. Our goal was to develop an integrated software infrastructure that e-business application developers could build on rather than having to piece together and then attempt to scale, administer, and tune a set of individual (and often otherwise unrelated) components. We started with a clean slate and the following high-level requirements:

1. The Platform should provide "access to data". That is, it should be designed for use in developing e-business Web sites and other advanced e-services for which data (and thus dynamically generated content) is central.

2. The Platform should be *incrementally* scalable so that a given e-business site can be scaled up cost-effectively over time in direct proportion to increasing demands for the services provided by the site.

3. The Platform should be highly-available. It should be designed for a 24x7 world, avoiding downtime in the face of component failures and/or the need to deploy new versions of data or code.

4. The Platform should provide high performance (subject to satisfying the previous goals). More specifically, the primary performance goal should be to provide fast, stable response times under high load conditions.

5. The Platform should exploit modern commodity hardware trends. Two such trends are the ease of assembling a large amount of processing capacity via a large "shared nothing" network of inexpensive machines (Sun Netras or rack-mountable Intel boxes) and the low cost of relatively large main memories for such machines.

6. The Platform should make it "easy" to build and administer e-business applications that are themselves scalable and highly-available.

7. The Platform should be developed using Java, for Java applications, in order to achieve a balance of time-to-market, robustness, and ease of evolving the system over time.

8. The Platform should be able to work together with an application developer's preferred Web server, application server, and relational database system products.

4 The Propel Distributed Services Platform

The Propel Distributed Services Platform provides an integrated set of services for use in building enterprise-level Java applications. Figure 3 lists the key components of the Propel Platform and illustrates how its use advances the standard three-tier Web site architecture shown in Figure 2. The Propel Distributed Services Platform enhances the site infrastructure, working together with industry-leading SQL database servers, J2EE application servers, and popular Web servers to provide a runtime environment that has been designed to significantly simplify the development and deployment of scalable, highly-available, e-business applications. Its major components are:

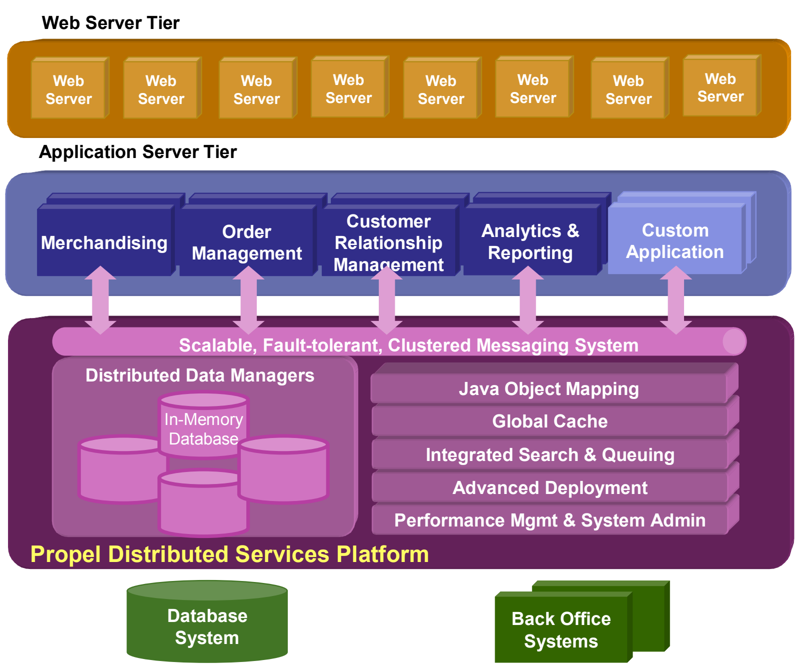

Figure 3: Propel Commerce System Web Site Architecture.

Clustered Messaging System: This component is the heart of the Propel Platform. The Clustered Messaging System provides a set of Java IPC APIs that are used by Platform services (and made available to interested application services) for inter-service communications. It supports a model where messages may be sent either to a named service or to a specific registered instance (provider) of a named service. It performs automatic load-balancing and failure management for registered services, and provides several different request routing protocols (master/slave, multi-master, and subscription). It runs on two or more nodes ("message hubs") of the site and is itself a clustered, scalable service - additional reliability and message throughput can be achieved by scaling up the number of message hubs. Its logically centralized nature contributes to system manageability and enables optimizations intended to minimize physical resource consumption and boost throughput under high loads (by reducing the number of required TCP/IP connections and allowing service-to-service message batching).

Distributed Data Management: This component provides an incrementally scalable framework for managing e-business data. It uses a collection of in-memory data managers (the TimesTen ^{TM} database management system is a component of our system) in concert with the application developer's preferred disk-based database system to store and manage e-business data. The in-memory data managers store tables in main memory rather than on disk to reduce query latency, using checkpoints and logging to ensure data durability in the event of system failures. Performance critical tables are placed under the control of one or more in-memory data managers, while less frequently accessed (or extremely large) data sets continue to reside in the disk-based database system. Data replication is utilized to ensure high availability and access-handling scalability for the in-memory data, and the system has been designed to permit partitioning of tables across multiple data managers for added scalability and capacity. Data placement is a physical design decision that is completely transparent to application programmers - applications can access Platform-managed data via the provided data access APIs as though it were contained in a single data manager. This eliminates any need for application-visible database partitioning or manual programming of cross-database queries.

Integrated Search: Also included in the Distributed Data Management component of the Propel Distributed Services Platform is support for full-text indexing and querying of the text columns of Platform tables. This enables a single Platform query to combine both text search and traditional (parametric) database search predicates, which can significantly simplify the development of e-business search features. Match-any and match-all searches are supported, as are application-provided tables containing stop words, synonyms, and common misspellings for any text column. Results are relevance-ranked, and ranks are made accessible for use in sorting and presenting query results. Full and incremental text index refresh modes are supported; per-transaction refresh is planned as well. Internally, the integrated search facilities ride on the in-memory Distributed Data Management infrastructure to achieve fast, reliable, scalable text searching.

Integrated Queuing: The Distributed Data Management component also includes built-in support for persistent queues. Queues are active, table-like data structures managed by the in-memory data managers for performance. Like tables, queues have schemas, and queue entries are typed. The Propel Platform's queues enable distributed services to reliably enqueue information for one another, allowing them to communicate asynchronously yet be assured that information loss will not occur in the event of system failures. Two forms of dequeue operation are supported, both of which use query predicates and ordering to specify the entries that an application wishes to dequeue. One form atomically removes the dequeued entries from the queue and returns them to the application, while the other form atomically updates their state in an application-specified manner but leaves them in the queue for later use. Since queues are part of the Propel Platform's Distributed Data Management framework, the Platform's transaction management capabilities apply to queues, and queues can be queried and joined with data from regular tables and other queues.

Java Object Mapping: To provide a natural programming model for use by Java application developers, the Propel Distributed Services Platform includes a layer of Java APIs known as the Java Object Mapping layer (OM layer). This layer provides a high-level Java object "view" of all Platform-managed data (including both data in tables and entries in queues). Its APIs are based on the widely used Java bean programming paradigm. The beans used by an application are automatically generated by the OM layer via an XML-based tool included with the Platform. The supported mappings include a variety of relationship types based on underlying referential integrity constraints and application-provided XML annotations. The OM layer provides access to related objects as interconnected Java beans. Both static and dynamic query-based access to data is provided. Anticipated queries can be specified at bean generation time and used to create simple-to-call methods that fully encapsulate complex database predicates (including both local predicates and path expressions). In addition, ad hoc queries can be built and executed programmatically at runtime. A set of generic data access and transaction management APIs are also provided by the OM layer.

Global Cache: The Global Cache is a hierarchical caching service intended for caching such objects as fragments of generated dynamic HTML content. For the e-business applications that the Propel Platform aims to serve, most HTML pages are dynamically computed based on Platform data content and on characteristics of the user for whom the page is being assembled. Site scalability can be significantly enhanced through the caching of such content, as it is very often reusable for a period of time (e.g., until the next deployment of product catalog or pricing information in the case of a product information page). Page fragments rather than just whole pages are cached to support reuse of dynamic content despite full personalization. The Global Cache's architecture consists of a level 1 cache on each application server plus a level 2 cache server that runs on one or more nodes of the site. The level 2 cache server is scalable through partitioning, and a least-recently used policy is used to manage the placement of fragments within the distributed cache. This architecture permits the caching capacity of the system to be scaled much more economically than architectures based solely on application server caches.

Advanced Deployment: High availability requires that new data and new code be deployable to a live e-business site without downtime or interference with ongoing sessions. To support this requirement, the Propel Platform includes an Advanced Deployment service that manages a multi-stage (offline, staging, and production) Web site model. For deployment of new application code, the Propel Advanced Deployment service interacts with the Clustered Messaging System to incrementally apply and then enable changes to services and servers within the site. For deployment of new database content, the OM Layer provides built-in support for differential data versioning. A table can be designated as versioned, and the OM Layer will then transparently modify the application's data accesses (on the fly) to select only that data associated with the session's target version. The Propel version representation requires space proportional just to the degree of change, not to the overall cardinality of a given versioned table. Both forward and backward rolling of deployed data changes are supported, and emergency change support is provided as well.

Distributed Services Manager: Last but certainly not least, the Propel Distributed Services Platform includes a Web-based system administration console called the Propel Distributed Services Manager. Through this console, an e-business site operator can configure and monitor the health and performance of all aspects of the site – including hardware (load balancers, Web servers, application services, and other machines within the site) as well as software (both Platform services and application services). The console includes pages that provide real-time performance reporting as well as viewing and filtering of the system-wide event logs and support for automated alerts. Database administration pages are provided as well, and the console has an extensible internal organization that permits new application services to be made known to the console, either for generic administration through default pages or for custom administration through JSP pages written specifically to manage a given service.

5 Summary and Future Plans

In this short paper, we have provided a brief summary of the motivation for and an overview of the Propel Distributed Services Platform. The system integrates a number of ideas from the distributed database and transaction processing worlds and has been designed to enable a "distributed system services approach" to the development of mission-critical e-business Web sites. In particular, it provides a set of components and interfaces intended to significantly simplify the problem of constructing and managing data-intensive Web applications as sets of communicating, scalable, highly-available services. Our future plans include additional work on object and query caching, flexible (XML) data management, geographically distributed e-business Web sites and applications, and hands-off performance management.