

Postgres Plus[®] and JBoss[®]

New Division of Labor for Enterprise Applications

Objective

The objective of this white paper is to help you understand the opportunities provided by two complementary open source infrastructure solutions – the Postgres Plus RDBMS from EnterpriseDB and the JBoss Application Server from Red Hat[®]. Postgres Plus is a high-powered, feature rich commercial distribution of PostgreSQL, the world’s most advanced open source database. JBoss is by far the world’s leading open source JEE application server.

Traditional Web and enterprise application architectures leverage JBoss extensively to service compute-, display- and data-intensive operations. But Web 2.0 and other advanced applications require much higher volumes of transactional data, and their scaling requirements are indeterminate. This paper suggests a rebalancing of responsibilities between JBoss and Postgres Plus that improve database performance and consistency, while delivering more resources to the application’s computational and UI operations.

Audiences

The intended audiences for this document are software architects, developers and database administrators. The document is oriented toward transactional applications, but it is also generally applicable to advanced enterprise applications of any type.

Introduction

The success of social networking sites such as Myspace, Facebook, and hi5 Networks is instructive on two dimensions. First, these sites prove that large, interactive communities can be cultivated using application components such as blogs, wikis, chat servers and user forums. Second, the usage patterns and volumes of social networking sites have exposed critical computing infrastructure requirements that do not exist in the world of static content and shopping carts (i.e., Web 1.0). Although these lessons may seem primarily useful to architects and developers of commercial Web 2.0 applications, in reality Web 2.0 architectures provide a roadmap for any enterprise application that needs to perform well, scale effectively and deliver critical intelligence to business stakeholders.

Traditional Web and enterprise application architectures leverage JBoss extensively to service compute-, display- and data-intensive operations. But Web 2.0 and other advanced applications require much higher volumes of transactional data, and their scaling requirements are indeterminate. This paper suggests a rebalancing of responsibilities between JBoss and Postgres Plus that improve database performance and consistency, while delivering more resources to applications' computational and UI operations.

Postgres Plus, JBoss and Hibernate

PostgreSQL, Postgres Plus and JBoss are longstanding open source favorites for building transactional applications. JBoss is the leading JEE application server, and Postgres solutions deliver the world's most advanced open source database management. Combined, Postgres Plus and JBoss offer powerful solutions for implementing a wide variety of transactional applications including Web 2.0, SaaS and general business applications.

JBoss' core data abstraction solution is Hibernate, an open source technology that's part of the JBoss family. Hibernate is a powerful, high performance object/relational persistence and query service. Hibernate lets developers implement persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate supports query predicates expressed in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and Example API. Hibernate includes a JBoss Certified dialect for Postgres. The Postgres dialect allows Hibernate to exploit the syntax, semantics and optimizations of PostgreSQL and Postgres Plus databases. Figure 1 illustrates the relationship between JBoss and Postgres Plus at a high level.

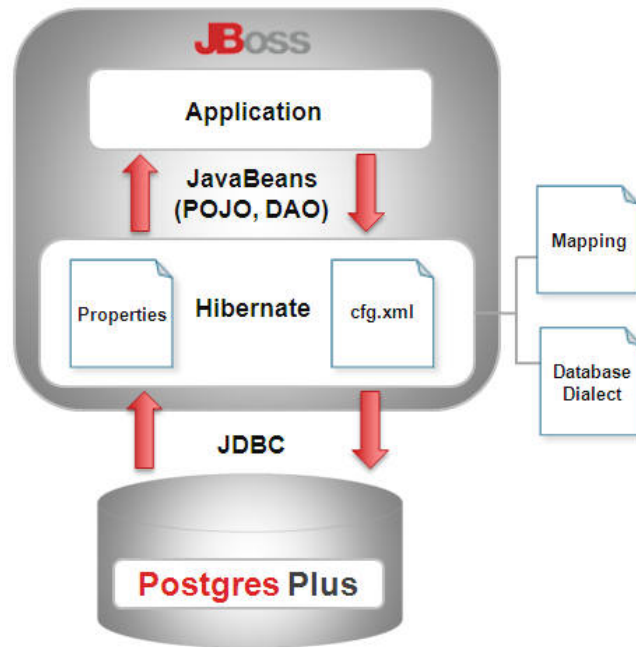


Figure 1 – High-level Postgres Plus and JBoss Architecture

Based on the Hibernate dialect for Postgres Plus, developers can create persistent logic and map Java classes to Postgres Plus database columns/types through Hibernate's mapping configuration file. In addition, Hibernate has a rich set of query and data manipulation capabilities that dramatically reduce Java developer time. Finally, Hibernate uses JDBC for connectivity to Postgres Plus. Hibernate's goal is to reduce the amount of time developers spend on data related code by 95%.

Postgres Plus and JBoss in a Web 1.0 World

Online stores such as Amazon.com and Dell.com are good examples of the Web 1.0 application paradigm. These sites provide extensive product catalogs that are viewed in high volume, but the catalog content is fairly static. Although some Web 1.0 applications support large visitor populations, dynamic site data is limited to features such as custom personal profiles, shopping carts and self-service.

JBoss is often the power plant of Web 1.0 application architectures. To mitigate database contention, frequently-accessed content is stored in JBoss' memory cache

and delivered to site visitors on demand. Since the data is relatively static, synchronizing the cache with content changes is not highly burdensome on the infrastructure. When the application applies a database transaction (insert, update or delete), JBoss synchronizes its cache with the new content, and then propagates changes to other servers in the infrastructure. This cache coherency approach scales acceptably for applications that have high query, low transaction workloads.

Web 2.0 Applications Pose Data Scaling Challenges

Social networking is the central theme of Web 2.0 applications. Wikis, blogs, chat servers and community forums are examples of social networking application components that provide high levels of user-driven interactivity. These interactive components place new demands on the application infrastructure. What's more, social networking sites face a thorny scaling dilemma – they must accommodate explosive growth, even though the magnitude and timing of that growth is difficult to predict. Myspace.com and Hi5 Networks.com, for example, now have more than 300 million and 75 million registered users, respectively, and face tremendous scaling challenges as their communities evolve.

Many Web 2.0 applications are highly transactional in nature, with community members frequently adding and updating site data. These users have very high expectations. Any latency or inconsistency is noticeable, and may cause users to discontinue visiting the site. Both Postgres and JBoss have a proven history of reliability and the ability to handle large volumes of complex transactions, making them ideal infrastructure components for Web 2.0 applications.

By combining high levels of interactivity and ACID transaction integrity, sites like hi5 Networks aim to capture and retain the attention of large user populations. The goal is to provide an experience that encourages users to visit often, stay connected for long periods and interact continuously with other community members. Thus, Web 2.0 applications pose important infrastructure challenges not present in their first generation counterparts, including the need to:

1. Handle large amounts of dynamic, user-driven data
2. Scale the underlying infrastructure in the face of uncertain demand
3. Maintain durable connections and sessions as site usage increases

As described earlier, Web 1.0 architectures overcome database contention by maintaining static and semi-static data in the application server (JBoss) cache, with cache coherency managed primarily by the JBoss instances themselves. Web 2.0 applications process much higher volumes of dynamic data – blog posts, wiki entries, chat archives – that can swamp the application server tier. As application usage increases, simply adding more JBoss servers to the mix is not an effective scaling solution. The overhead of maintaining consistent cache data across an expanding server farm increases latency and starves JBoss servers of cycles that are needed in other areas of the application (e.g., to manage rich user interfaces). The solution to these new architectural challenges is a high performance, massively scalable Postgres Plus database tier that complements and enhances the JBoss infrastructure.

Database Scaling with Postgres Plus

As the load of a Web 2.0 application grows, response times will begin to deteriorate unless the infrastructure is appropriately scaled. The data intensive nature of these applications correctly suggests that significant scaling benefits can be achieved at the database tier. Thus, the database tier must be scaled vertically and/or horizontally. Vertical scaling involves increasing the computing power (CPU, memory and disk resources) available to a database server. Horizontal scaling involves partitioning and/or replicating the database across multiple instances using commodity hardware and software. It is often the most cost effective means through which to build a Web 2.0 infrastructure, but requires the database to efficiently perform complex distributed computing operations. Postgres Plus delivers several capabilities that are critical to both vertical and horizontal scale-out of the database tier.

1. **Infinite Cache.** Many data-intensive applications cache frequently-accessed data at the app server tier. As discussed earlier, however, Web 2.0 data is often dynamic in nature. Thus, Web 2.0 applications burden JBoss with the chores of maintaining coherent, synchronized cache data. As application usage grows, the overhead of managing an increasingly distributed JBoss cache can increase latency and limit JBoss' ability to service the application's compute- and display-intensive operations. The optimal division of labor is for JBoss servers to handle compute- and display-intensive operations and Postgres Plus to handle distributed cache operations.

Postgres Plus provides special-purpose Infinite Cache servers that offload cache management chores from the application server tier without adding overhead to the primary database (see Figure 2). Importantly, Infinite Cache allows cache invalidation to be triggered from the database, ensuring that the application does not return stale data even if the data is altered by external systems. By isolating resource intensive cache management operations to Infinite Cache data servers, Postgres Plus allows JBoss to service that application's growing computational and UI demands.

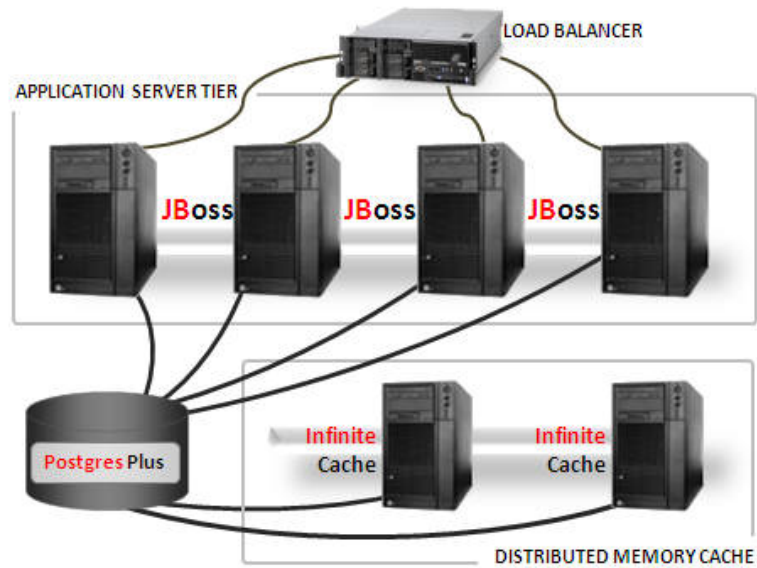


Figure 2 – Postgres Plus Infinite Cache in JBoss Cluster

- Transaction Processing.** Although advanced capabilities like Infinite Cache offload resource-intensive query operations to special purpose cache servers, optimizing the database for OLTP operations, the database itself must also have the raw power to handle transactional workloads. Postgres databases are renowned for transaction processing, functioning flawlessly as the workhorse behind many of the most demanding JBoss applications in production today. Postgres Plus includes multi-version concurrency control (MVCC) and table-level partitioning to further enhance the performance of transactional systems. MVCC is an “optimistic” concurrency strategy that allows read operations to acquire query contexts without blocking write (insert/update/delete) operations. MVCC significantly reduces data contention and, thereby, improves performance in transactional applications by minimizing locking operations. Table-level partitioning allows data to be sliced horizontally and distributed across database instances, optimizing access to frequently needed data while ensuring fast, consistent access to all information.

3. **Advanced Database-level Partitioning.** As a Web 2.0 application community grows, additional scale can be achieved by partitioning the main application database into multiple physical servers. A horizontal partitioning strategy called “sharding” is useful in an increasing number of Web 2.0 scale-outs. Hibernate Shards allow a database to be partitioned into subsets. By deploying database partitions on many physical servers, the bandwidth needed to handle growing application workloads is distributed across the infrastructure. Postgres Plus includes sophisticated database federation capabilities that support cross-database joins by linking federated tables into a single schema (see Figure 3), allowing user applications to transparently retrieve data from multiple partitions and process unified result sets. These advanced capabilities enhance Hibernate Shards by solving the thorny problem of cross-shard joins, so JBoss applications benefit automatically when Postgres Plus partitioning is in use to implement sharding strategies.

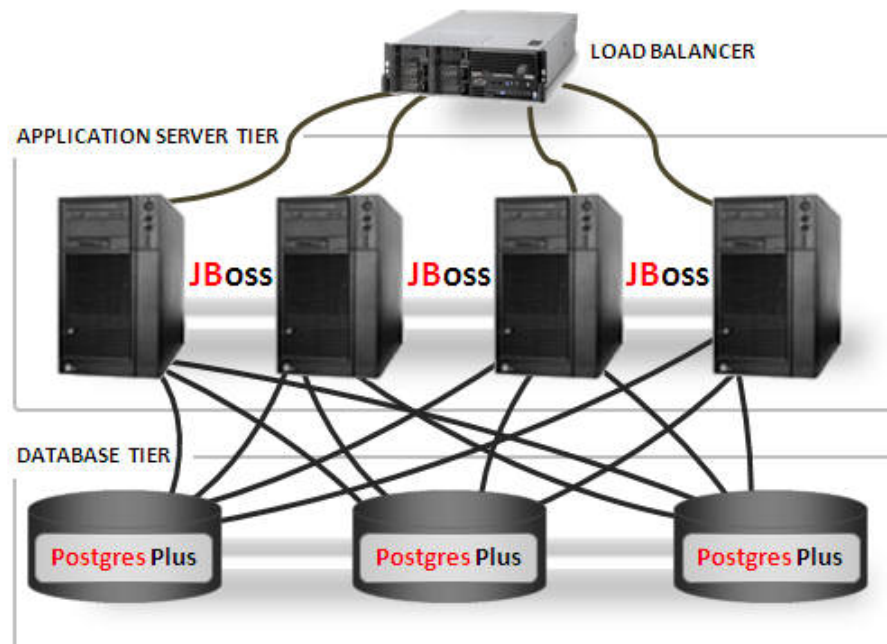


Figure 3 – Partitioned Postgres Plus Database in JBoss Cluster

Summary

Advanced enterprise applications offer exciting new ways to attract, retain and cultivate vast user communities. However, the sheer volume of visitors and the transactional nature of their interactions pose very different computing infrastructure demands when compared with predecessor application architectures. What's more, infrastructures supporting these applications must be able to scale rapidly in the face of highly uncertain growth.

The combination of JBoss Application Server and Postgres Plus RDBMS represents the leading infrastructure stack for transactional applications. However, Web 2.0 style applications pose special scalability challenges, and require a new division of labor between JBoss and Postgres. Postgres Plus can be configured to deliver advanced memory caching, transaction processing and partitioning services within the database tier, allowing JBoss to service the application's expanding compute- and display-intensive operations.

*EnterpriseDB and Postgres Plus are trademarks of EnterpriseDB Corporation. All other marks are the property of their respective owners.