

WHITE PAPER

Modernizing Application Platforms for Cost-Effective Cloud Readiness

Sponsored by: VMware

Al Hilwa

Maureen Fleming

June 2011

EXECUTIVE SUMMARY

Application platform modernization has shifted from an optional activity to a requirement as a result of increasing challenges and complexity that cannot be sustained without structural change.

We are living in an era that is characterized by an ever-quicken pace of technological change. We will witness exceptional disruptive transformations in both the front end and the back end of computing systems and the architectures of applications that run on them.

Because enterprises cannot throw out existing investments in their production applications, application platform modernization is an evolutionary process. Efforts tend to occur in a forced march over a series of phases over a period of time. By the end of the evolution, enterprises will have application platforms ready for cloud enablement.

Modern application platform offerings from software vendors require functionality across many areas, including lightweight frameworks, support of at least one type of container to house business logic, in-memory data management, queuing and messaging middleware, and monitoring and application management. This collection of needed functionality is offered by VMware under the vFabric Cloud Application Platform.

While enterprises face several challenges in evolving to a modern application platform, the benefits in cost savings, greater agility, and responsiveness far outweigh the sum of the challenges.

SITUATION OVERVIEW

Enterprise application platforms built and assembled over the past two decades began to evolve a few years ago, and the pace of change is accelerating and will continue to accelerate over the next several years. Application platform modernization is driven by structural need caused by a crisis of complexity, involving:

- ☒ Increasingly stringent customer requirements, such as faster cycle times, tougher service-level agreements, and more customization and broad adoption of consumer technology, that are forcing changes to the support and delivery of goods and services to customers
- ☒ Varied growth-oriented business goals requiring technology enablement

- ☒ Ongoing cost-cutting and efficiency efforts aimed at containing IT spending
- ☒ More regulations that need to be supported technologically

These competing demands are unsustainable without rethinking application platforms and how applications are constructed. Many IT organizations are embarking on a journey to take costs out of their application platforms and improve responsiveness through the adoption of new architectures as well as approaches to managing an application life cycle.

This is a journey because we are living in an era that is characterized by an ever-quicken pace of technological change. We will witness exceptional disruptive transformations in both the front end and the back end of computing systems and the architectures of applications that run on them. The next five years will bring changes in the way users access applications through mobile devices and interact with each other through socially enabled, cloud-backed services. In particular, there are four important areas of influences and pressures on application architectures:

- ☒ **Mobile devices.** Smartphones and tablets are packed with sensors and input equipment that add high levels of context to enhance how users interact with applications.
- ☒ **Social networking.** New and evolving modes of interaction are forcing changes in the way users consume data and content and at the same are generating copious amounts of it. These new social modes of interaction are making their way into applications and will increasingly put new burdens on back-end data processing and analysis.
- ☒ **Cloud services.** The explosion in devices and the new modes of interaction have generated demand for connectivity and new services, which must be available everywhere, available at high levels of reliability and performance, and available through agile and elastic provisioning mechanisms.
- ☒ **Big data.** This is a new class of technologies that enable management, access, and analysis of much larger sets of data than had been conventionally possible. Big data technology is used to solve new classes of problems that create new application functionality. Examples include the ability to analyze large volumes of data to identify and respond to trends as well as support high volumes of transitory data used in Web applications and caching.

These broad areas of change arrive at a time when IT organizations operate in a continuing mode of maximizing efficiencies, improving responsiveness, and reducing costs.

Modernizing Application Platforms

Because enterprises cannot throw out existing investments in their production applications, application platform modernization is an evolutionary process. Efforts tend to occur in a forced march over a series of phases over a period of time. While these phases may not be in the exact order described and some of the phases occur simultaneously, they provide a useful description of the modernization evolution taking place in the market.

Phase 1: Shift from Proprietary De Facto Standard Application Servers to Multiple Frameworks and Heterogeneous Containers

With the introduction of the J2EE application server in the late 1990s, enterprises began to adopt and standardize on a single application server product from one of many application server vendors. For each enterprise, these choices effectively resulted in a de facto standard product to deploy large and important enterprise applications.

Over the next several years, the market for application servers consolidated down to a few choices, and as a result of the relative lack of competition and the importance of the application server, prices rose to the point where the annual license and maintenance costs captured the attention of IT.

Enterprises found they were spending more on licenses and maintenance to support an application server that they were locked into, that was often more feature rich than they needed, and, at the same time, that required them to pay a premium price for developers with skills in the particulars of their de facto server.

In addition, the complexity of enterprise Java made it much more difficult and expensive to develop applications.

Given the increasing use of application servers to run Web applications and the corresponding cost pressures, many enterprises realized they could no longer afford to support this approach and needed greater choice. They responded by reexamining the framework of choice — J2EE — and began adopting lighter-weight options, such as Spring.

In addition, they began to identify alternative, lower-cost application servers that could support the lighter-weight choices.

That caused a shift away from de facto standard products to a policy of supporting a portfolio of Java-based application servers and frameworks. In addition, lower-cost, open source alternatives evolved to become powerful enough to run large, mission-critical applications and lightweight enough to run in a highly distributed fashion on inexpensive hardware. During this time, Spring became a popular lightweight Java-based framework.

Currently, many enterprises support both heterogeneous application server environments and frameworks. On the Java front, there is an effort to drive down the cost of development through an emphasis on pure Java skills for developers and pure support of frameworks. In numerous discussions with enterprises about this evolution, we've found consistent benefits, including:

- ☒ Lower license and maintenance costs for application servers
- ☒ Ability to develop on the same environment as production, reducing the cost of testing and bug fixes and improving time to value
- ☒ Easier troubleshooting by developers because they understand the application architecture at a lower level
- ☒ Lower labor costs for development because enterprises can reduce the number of developers on staff who are specific application server specialists

Phase 2: Standardize Application Life-Cycle and Runtime Operations

Driving down the cost of software and development while increasing the flexibility of adopting the best approach for an application project satisfied both developers and finance, but it caused problems for operations that resulted in the next phase of application platform modernization.

The rule of thumb in operations is that standardized processes are less expensive than nonstandardized processes. Introducing more choice around application servers and frameworks shifted a single set of standards to multiple products and development approaches. To regain efficiency, application life-cycle and application operations teams needed to refactor, shifting from a focus on tooling that supports a single application server or server cluster to standardized application life-cycle and management processes.

In discussions with enterprise application platform teams working on this type of evolution, we've found they are heavily focused on a least-cost strategy around standardizing their processes, including greater automation. One aspect of this is standardizing the development, testing, and production environments associated with an application.

Tomcat is the most frequently deployed Java application server and is often the server of choice for application development. Traditionally, enterprises move to a different application server in production, which can lengthen the development cycle as testing occurs in the shift from development to production.

Tomcat is now hardened and supported by vendors, which means enterprises are able to develop, test, and run applications using the same environment. This provides a low-cost offering measured by software costs, developer productivity, scalability, and time to value.

Ultimately, there is a drive to reduce the number of containers supported by IT but an acknowledgement that multiple containers are required to service different application needs. Custom applications tend to be built on the least-cost product that meets the application requirements, and implementations of packaged applications are built on the container specified by the application provider.

Phase 3: Adopt Lightweight Container Architecture and Virtualize

Traditional Web applications are built with a single application server or server cluster accessing a single database or database cluster. The business logic of an application is often tied to the database. Enterprises are shifting to an approach that moves the business logic to containers.

They are making this move to prepare for virtualizing their application platform to automate the management of compute capacity across the environment. As part of the effort, they focus on lightweight containers such as Spring, which reduces their power consumption, starts up faster, improves throughput and scalability, and allows the platform to dynamically scale based on application capacity needs.

A large European media organization realized it needed to improve the scalability of its online service when viewership began to grow rapidly. The service was built on a proprietary source application server accessing a database. The development team began the process of reducing dependency on the database by adopting Hibernate for object-relational mapping and experienced performance benefits. But once its online service experienced a 150% increase in traffic, the service effectively hit a performance wall.

The team decided to shift the business logic to the lighter-weight Spring framework and tested VMware's tcServer, finding that start-up time was less than a minute compared with 15 minutes on its current JEE application server. The team found that it was able to scale to meet demand on fewer servers.

Start-up times were important because the team wanted to virtualize the application platform running its online service and automate the commissioning and decommissioning of compute capacity based on load. When the team had to wait 15 minutes, it was difficult to meet demand.

Phase 4: Rethink Data Strategy to Improve Performance

The database tier of applications is also undergoing significant change in application modernization for speed, scalability, and cost reasons. Classically, a relational database is used to store the data of record in addition to temporary data used for an application session. Enterprises began experimenting with shifting read-only data to caches to improve scalability and performance and also to reduce database license and maintenance costs.

Once application platforms began to be virtualized, there was also a need to abstract access to data, and enterprises began using data services to connect application logic to data. Once the direct connection between a container and its data was severed, enterprises were then able to further improve the scalability and performance of applications by moving data into in-memory data fabrics. Unlike data caches, data fabrics manage data in memory across servers, supporting both relational and object-oriented models. Once data is inserted, updated, or deleted, those changes are replicated to its peers to maximize availability and scalability.

Data fabrics can also be virtualized on commodity servers, with all the cost savings and benefits of elastic availability of resources based on workloads. Because the data is accessed in memory, the throughput of an application is much faster and there is a tendency to require fewer application servers.

In addition, the use of data fabrics provides the opportunity to build newer, decision-centric applications. These types of applications consume large volumes of data at high speeds to perform the analysis needed to detect when a decision needs to be made and also perform analysis to determine best options for the decision.

Phase 5: Modernize the Application Platform Around Utility Services in a Private Cloud-Like Environment

Amid the modernization of an enterprise's application platform, there is often a major initiative around a mission-critical area that requires an entirely new approach and new application architecture. In discussions with enterprises, we've found this often happens when:

- ☒ An enterprise runs into a performance wall around changing customer requirements
- ☒ Change management becomes prohibitively expensive
- ☒ Cycle times speed up to the point where IT can no longer support business without making a significant change

At that point, modernization moves into high gear. There is a realization that the development team needs to minimize redundancy and increase throughput across its portfolio of related applications by creating standardized, services-oriented layers to handle processing between customers (whether internal or external) and the applications that manage and fulfill the requirements of the customer relationship.

The focus moves away from an application to a set of utility services that handle inputs and outputs and manage processing in between. In these scenarios, there is a move away from single application workloads to a processing backbone that handles all workloads, breaking processing down into a series of redundant steps. This backbone receives requests or events, processes them, and feeds necessary data or tasks to the appropriate application.

Restaurant Chain Improves Scalability of Custom Point-of-Sale Application

Originally, developers for this restaurant chain franchisee built their applications on an application server running Unix. While the same number of workers used the applications, the developers were constantly building more application functionality, keeping more staff on the system longer.

When performance began to stall for an important point-of-sale (POS) application, they upgraded the underlying hardware to new high-end servers and did not get the boost in software performance they expected. The developers made a decision to move away from a centralized application server running sticky sessions. Instead, they built a new application around what they called private cloud architecture, consisting of:

- ☒ An event-driven, stateless, highly parallelized, task-oriented environment
- ☒ tcServer to serve Web applications and also run Spring applications when needed
- ☒ A homogeneous set of resources available to do work
- ☒ A generic framework of listeners that listen on demand, using RabbitMQ for queuing

Under the new architecture, the developers gained the scalability, availability, and performance they needed, running workloads split up over a dozen low-cost servers instead of a couple of high-end servers. The change in application architecture also meant their core services do not suffer from downtime when someone has a machine locked up.

Financial Services Firm Builds Execution and Settlement Backbone

A specialist in high-volume, high-speed trade execution and settlement realized it needed to improve the flexibility of supporting its customers while also improving the efficiency of the processes involved with its core business. The company connected to many exchanges worldwide, and integration with each exchange was implemented as a point-to-point connection, developed over a long period of time. There are as many ways to clear and settle as there are exchanges.

The financial services firm changed this by creating an event-driven processing backbone that maximizes the reusability of processing steps by breaking transaction processing down into a series of components with attached queues all managed in memory. A component begins processing when a message moves into the queue, and when it is finished, it sends work to the next step in the process until processing is completed. Eventually, the work is integrated into the basic applications that handle the business services the customers contract for with this financial services provider.

Only one step in the process connects to each exchange, which isolates change management to just one area of the backbone. By the same token, the connection to application endpoints is isolated as well, also in support of faster and more accurate change management.

Application Platform Modernization Leads to Cloud Readiness

Enterprises will move to the cloud in many ways. They may decide to use the cloud for seasonal extra capacity in a hybrid model. They may shift applications entirely to the cloud or parts of applications that benefit by being in the cloud.

It is clear that enterprises focused on modernizing their application platforms are far along the path to cloud readiness. Building applications on lightweight containers, virtualizing, and automating management all help to achieve elastic scalability.

Separating business logic from data and connecting through Web services support the needs for high throughput, availability, and efficiency.

Rearchitecting to create utility-based application services supports the move to cloud-like utility models. Building this on top of an event-driven architecture not only provides greater throughput and elastic scaling but also establishes a path to route work as directed to whatever location is available for processing.

Requirements of a Modern Application Platform Offering

Modern application platform offerings from software vendors require functionality across many areas, including:

- ☒ Lightweight frameworks
- ☒ Support of at least one type of container to house business logic
- ☒ In-memory data management
- ☒ Queuing and messaging middleware
- ☒ Monitoring and application management

This collection of needed functionality is offered by VMware under the vFabric Cloud Application Platform. The ingredients of vFabric include:

- ☒ **Spring Framework.** This Java framework simplified the coding of enterprise applications by integrating important design patterns into a single coherent open source project. Spring popularized the idea of configuring program components through inversion of control and declarative transaction management and made it possible for many enterprise applications to be developed and deployed without an application server. The framework has been available as Apache licensed open source code since 2002 and has evolved to encompass many of the most productive new ideas in programming over the past decade, such as the MVC design pattern and aspect-oriented programming. The impact of Spring on the Java world has been significant, stimulating JEE and EJB to adopt many of the same concepts and to evolve to be simpler and cleaner.
- ☒ **vFabric tcServer.** This Apache Tomcat-based lightweight application server includes a variety of technologies for managing applications built with the Spring framework and leveraging the underlying vSphere virtualization engine. tcServer adds enterprise capabilities to Tomcat, such as management and deployment across multiple nodes, sophisticated diagnostic functionality, and optimizations for virtualized environments, making it well-suited for cloud delivery.
- ☒ **vFabric GemFire Database System.** This is a distributed data database system with elasticity and reliability features that inherently leverage cloud architectures, providing options for modern application requirements. GemFire brings high data management capabilities, such as parallel disk persistence and wide-area data distribution, to Web applications with specific optimizations for Spring and tcServer.
- ☒ **vFabric Hyperic Management Tools.** This performance management solution for applications in virtualized and cloud environments provides monitoring and control across multiple layers of the service stack. The Hyperic tools are essential for managing private and public application platform clouds.

- ☒ **RabbitMQ Queuing System.** This is a high-performance, open source implementation of the AMQP specification, which is the modern de facto standard protocol for interoperable, reliable, and secure messaging technologies. Messaging patterns such as request-response and publish-subscribe have become essential ingredients inside cloud application architectures and are supported by RabbitMQ.
- ☒ **vFabric Cloud Foundry.** This core cloud infrastructure ties layers of the application platform together into a platform as a service (PaaS) and provides the key provision and management functions for applications. Cloud Foundry includes the Cloud Controller, the Health Manager, Routers, and Droplet Execution Agents (DEAs), as well as a variety of other services.

FUTURE OUTLOOK

IDC believes that many traditional custom-built and ISV applications will be rearchitected over the next five years to take advantage of a modern application platform supporting cloud software architectures. Enterprises that are evolving to cloud application platforms face several challenges as well as significant opportunities.

Key challenges include:

- ☒ IT organizations may be concerned that they have more urgent priorities that preclude them from learning a new framework or style of application.
- ☒ The time, effort, and cost to rebuild configuration and deployment tools may make it difficult for the IT team to justify an investment.
- ☒ Concerns over a steep learning curve to rearchitect and rebuild applications to support cloud enablement may cause enterprises to delay adoption.
- ☒ Organizations with a strong culture of implementing packaged applications may not see the benefit of a modern application platform. They may shift directly to a software-as-a-service offering, bypassing modernization and development on a PaaS.

Key opportunities include:

- ☒ Future proofing the enterprise by supporting an agile environment capable of speeding up the process of change
- ☒ Significant cost savings through each evolutionary phase of modernization
- ☒ Private and public cloud readiness for custom-developed applications

CONCLUSION

VMware has moved strategically to assemble and integrate a strong stack under the vFabric brand, leveraging the competency of its SpringSource division in open source project and community management and governance offerings. In building vFabric, VMware has sought to enter the application platform fray and act as a new center of gravity for Java-based research and development, establishing itself at the forefront of taking Java to a modern application platform, then to the cloud.

VMware's integrated stack approach and cloud optimizations bring a higher level of coherence across stack members than has typically been seen in the Java space, serving to attract a significant contingent of developers numbering in the millions.

Copyright Notice

External Publication of IDC Information and Data — Any IDC information that is to be used in advertising, press releases, or promotional materials requires prior written approval from the appropriate IDC Vice President or Country Manager. A draft of the proposed document should accompany any such request. IDC reserves the right to deny approval of external usage for any reason.

Copyright 2011 IDC. Reproduction without written permission is completely forbidden.