CANONS, CONTROVERSIES AND EVOLUTION OF E-BUSINESS MULTI-TIERED ARCHITECTURE

Chris Leowski, Ph.D.
University of Toronto
Toronto, Ontario, Canada
chris.leowski@utoronto.ca

April 2006

Abstract: Based on the author's many years as IT director at the University of Toronto, and as a systems consultant in large banks and brokerage houses, the paper offers a review of accepted truths, as well as controversies and less orthodox views, surrounding multi-tier architectures for e-business environments. What is the accepted canon, and to what extent practical considerations can undermine it and lead to its modifications? How different groups within the enterprise cooperate, but also compete, in building a functioning e-business system, and how their perceptions and actions affect the company's business and systems models and the quality of implemented solutions? The paper examines to what extent practical systems architectures – and ecommerce systems in particular – follow the well-known principles of multi-tiered design, what are theoretical and practical justifications for separating system components into multiple tiers, and under what circumstances those rules can be, and often are, circumvented or abandoned. Finally, the paper traces the evolution of multi-tiered e-business systems architecture towards the zones/container-based solutions made available by the latest versions of some operating systems.

1 INTRODUCTION

The universally accepted cannon of systems design calls for separation of the presentation, business logic, and database layers – what is often referred to as the classic 3-tier architecture - and is, in most cases, religiously followed by systems architects around the world. When asked to justify the added cost and complexity of multi-tiered systems deployed on multiple layers of hardware, the same architects usually put forward the site's security as the primary reason. Firewalls between functionally distinct layers of servers, restricted access rules, authentication and authorization at every stage, can certainly enhance security. And yet, the argument is not only overemphasized, but is not even the primary reason for implementing multi-tiered systems architecture. In fact, security considerations are often separate and are only loosely related (though related nevertheless) to design concepts that see the number of layers expanded or collapsed for totally different reasons. In many practical implementations, in brokerage and banking in particular, security itself is isolated into a separate layer, or even multiple layers.

Banks and brokerage houses are probably in the forefront of e-business technology solutions. The amounts of money involved, and of potential losses in case of a disaster, are huge, and the availability and reliability of portals and of the entire middle tier and backend infrastructures to customers – whether institutional or individual – must not be compromised. What is less emphasized, however, is the fact that different information technology groups and business groups within the same institution often approach the task of designing, building and running e-business systems in a different way. Individual business groups usually focus on the system's functionality to the exclusion of almost everything else. They are interested in what the proposed solution can do for their customers and clients and at what performance levels. They take it for granted that what gets implemented will be secure, efficient, and will comply with the organization's business and system models – without being aware of what the latter are or even whether they exist. The very

concept of an SLA (Service Level Agreement) was to introduce a "contractual" obligation of IT groups (providers) to satisfy functionality and performance level requirements of business units (consumers). Software developers, often working against impossible deadlines, concentrate their efforts on delivering the required functionality without much regard for how their newly developed modules fit into the overall enterprise architecture. Selection of design and coding tools is often based on what resources happen to be available at the moment. Security departments have their lists of what is and what is not allowed once business proposals become IT projects, and security walkthroughs are designed to enforce those rules. Systems architects are about the only group to see all ramifications of newly approved projects and their place within the overall enterprise architecture. They are the very people whose task is to ensure that the project's development cycle conforms to the company's standards and is consistent with the enterprise software development model and – equally important – with the enterprise systems model. It is that latter model – developed over the years through a combination of theoretical principles and practical requirements – that defines how services are grouped into separate tiers, how those tiers interact with one another and with all other systems within the enterprise (McGovern, 2003) . Companies that lack a strong systems architecture group are easy to spot – their IT systems are a collage of point solutions developed in response to diverse and often short-term requirements, deploying many incompatible technologies, working with overlapping and badly synchronized data in various parts of the company, and generally reflecting the "silos mentality" of individual business units and their respective IT fiefdoms. An external systems consultant can easily identify this mess – a working mess, but nevertheless a mess, and a costly one for that.

2 WHY MULTIPLE TIERS?

How many tiers should be used for deployment of a typical e-business application, and why? Let us first examine a more comprehensive justification for systems tier separation both at the logical and the physical levels. The latter distinction is important, because multiple logical tiers often reflect business application's components, and do not have to be implemented on distinct physical layers (Harmon, 2001). Although the discussion can be equally heated in both cases, it is the physical tier separation that is of primary concern to us, since it often has far reaching implications for the application's security, functionality, performance, and scalability.

We already stated that security concerns are at best one, and not necessarily the most obvious reason, and that security modules are often developed as a separate tier or tiers to interact with all others. Functionality makes obviously a convenient and convincing argument. Grouping the system's functions along logical and practical lines makes for a neat design and helps us understand all interactions, as well as helps in future maintenance of the resulting solution. Usually those functions rely on widely deployed system components and off-the-shelf, though often customizable, applications. Presenting users with choices and actions is accomplished by using web server technology, while performing all necessary massaging of data and complex checks and calculations is delegated to various middleware systems, with actual data storage and retrieval being handled by the backend systems that, in most cases, deploy relational database technology. Thus we get the "classic" three-tier architecture. In practice, though, things often are not that simple and clear cut.

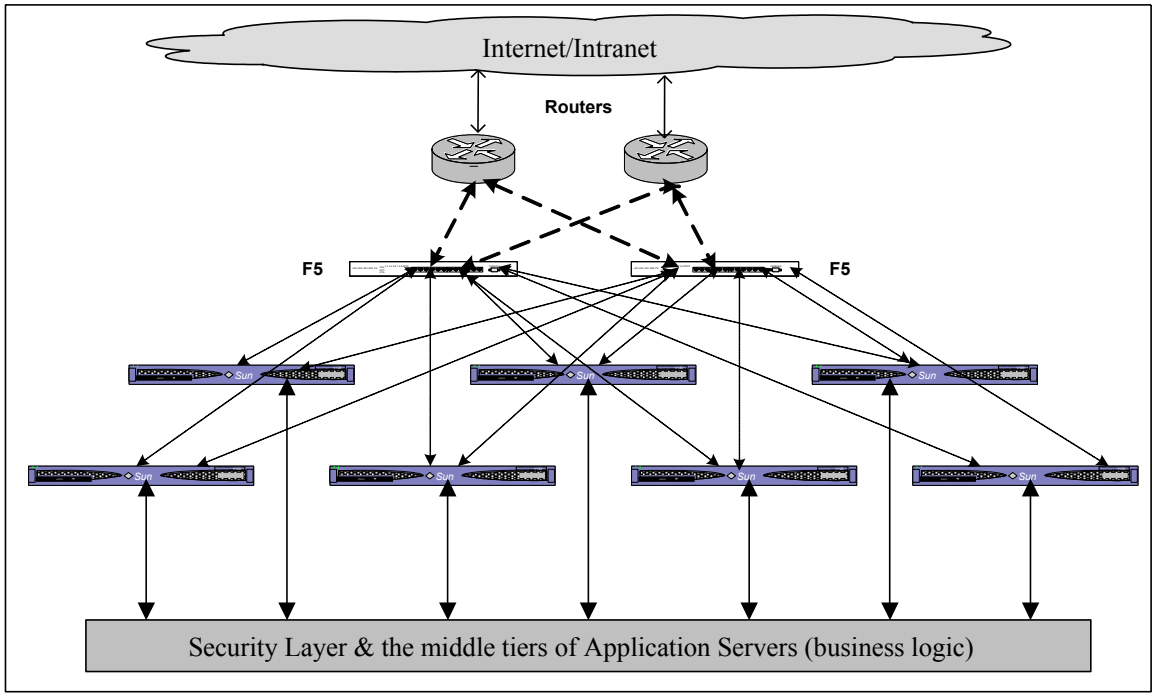
While distinct functionality is an important and valid argument for tier separation, enterprises deploying e-business solutions found that scalability and reliability were the two preponderant factors in developing enterprise systems architecture models. Both scalability and reliability constitute a *conditio sine qua non* of a successful e-business solution – the first one, because as the numbers of sessions and transactions grow, the SLA mandated performance levels have to be maintained; the second, because as individual components may fail, an intra-session continuity and full in-memory persistence of transactions – in addition to database persistence - have to be safeguarded. The main problem encountered by systems architects reflect the fact that different systems and applications that typically constitute components of an e-business solution scale differently and require different mechanisms to ensure their reliability. Standard choices between horizontal and vertical scalability - or a combination of both - apply differently across services and components, creating the need to balance the entire system while at the same time safeguarding its reliability expressed as ability to fail over within each service and component, as well as between services and components. Thus, **system tiers reflect**

relative homogeneity of components and services in relation to how they scale and how they fail over.

3 HOW DO THEY SCALE?

Viewed in this light, web services tier – or the part of it known as the presentation tier – scales very well both horizontally and vertically. That is, the systems architect has an option of adding multiple web servers (horizontal scaling), migrating to more powerful web servers (vertical scaling) or a combination of both. In practice, horizontal scaling adds the bonus of multiple access channels, assuming that appropriate load balancing and session fail over devices (e.g. F5) are deployed in front of the web servers. In extreme cases multiple (usually two) paths from independent ISPs lead to the same set of servers and services, so that customer traffic can be routed through the auxiliary channel if the primary ISP fails. As long as the "traffic cop" devices distribute incoming requests, there seem to be no obvious limits to horizontal deployment of additional servers, save for the cost of maintaining a larger server farm in multiple locations. A typical scenario is illustrated in Figure 1.

Figure 1.
*Typical horizontal scaling within the **presentation layer**: multiple independent web servers balance the load using round-robin or weighted algorithms, and providing automatic failover. Multiple routers and load management devices offer alternate access channels.*



Business logic tier usually scales well horizontally – but only up to a point. The requirement to maintain the full session persistence means that application servers that form part of the business logic tier have to work as a cluster – with intra cluster socket-to-socket and multicast communication. Maintaining in-memory session replication across large clusters involves substantial overhead, and from a certain size of the cluster vertical scalability may be a far better option to solve increased load and performance issues than adding more servers to the cluster. In practice, this tier is particularly tricky. Scaling vertically means more powerful hardware running larger application servers. The latter, at least those commercially available, run as Java applications within their own Java Virtual Machine (e.g. BEA's WebLogic, IBM's Websphere, and many other application servers from SUN, Oracle, etc., including the free Tomcat from Apache). JVM's memory management mechanism, relying on creation of new objects in the 'eden' part of the heap, subsequent copying of objects to other parts of

the heap, and eventual garbage collection of de-referenced objects, although easy on developers, can be a nightmare for those responsible for performance tuning of large e-business applications. It has been frequently observed that a full-scale randomly occurring garbage collection within a large JVM can stop a sensitive application dead in its tracks for that extra second or two, spelling the difference between acceptable and unacceptable performance levels. Peak load on any given application means high numbers of new requests resulting in accelerated creation of new objects within the application server's JVM, during which the usage of the heap shoots up to the levels that – if not carefully controlled –trigger full and/or frequent garbage collection. Given that such peak loads coincide with peak usage of many other components of the company's IT systems, the resulting degradation of overall performance can be quite severe, triggering – for example - angry phone calls from the trading floor and escalations up the company's command chain.

Figures 2 and 3 illustrate typical scenarios for vertical and horizontal scaling of the middle tier of application servers working in a cluster. The left side of each figure shows the "before" state, while the right side shows the "after" state of each solution. In the vertical scaling case the cluster is simply redeployed on more powerful hardware, and each member of the cluster now runs in a larger JVM and has access to more of other system resources (CPU, I/O channels, etc.). Multiple business applications address the same cluster of application servers, and the performance gain reflects a more powerful cluster after redeployment. In the typical horizontal scaling scenario (Figure 3) a second similar cluster is deployed on additional hardware with the same or similar characteristics as the hardware that runs the original cluster. Available resources are doubled, but more importantly business applications are now split between the two clusters. Although these scenarios serve only as illustrations, they are not very far from what takes place in the real world. The number of additional clusters in the horizontal scaling scenario can vary, with multiple clustered groups of application servers running often on the same grouped set of hardware – usually one deployed at the primary business site and the other at the "business continuity" site, or – in case of geographically distributed businesses – in different locations (e.g. some members of each cluster in Toronto and others in New York). In many cases business expansion leads to a combination of horizontal and vertical scaling solutions within the middle tier, often incorporating disaster recovery and business continuity solutions at the same time.

It is often maintained that the middle tiers (plural!) are where all action takes place – everything else is either presentation or persistence (data storage). Simplistic as such opinions can be – for example, a well designed presentation tier can be a very complex, multi-channel, itself often a two-tier, load balanced and with multiple fail over safeguards, system – there is a fair amount of truth in them. The level of complexity within business logic tiers of modern e-business applications is often staggering. Multiple asynchronous operations need to be performed to check the client's authenticity and authorization to perform certain actions, check the client's credit, notify various parties to the deal, generate many audit threads, back-off to well pre-defined points in case of inconsistencies or failures, restore a session, request manual intervention, verify and store transactions, etc. Asynchronous as they may be, they all need to be eventually synchronized within a completed unit of operation (whatever the latter is – a security trade; a bank deposit; a foreign exchange transaction; etc.). Multiple message queues can get overcrowded, retrievals from external data repositories (e.g. a credit check) can be slower than usual, synchronization of a particular transaction with other business areas of the company can fail and needs to be stored and queued for a later update; etc. All this underlines criticality of middle tiers to the well being of the entire e-business solution. We will see later how various IT groups within the company can view this criticality in different, and often contradictory, ways.

Finally, the database (backend) tier scales well only vertically. Contrary to claims of many database vendors, horizontal scalability of relational database management systems has not been so far a great success story. Influenced often by the pricing models of the relational database systems vendors, companies tend to minimize the number of primary database servers by running multiple database instances on more powerful hardware. Thus primary vertical scaling is combined with a version of horizontal scaling defined as multiple independent instances on the same physical hardware, rather than a truly distributed database on multiple hardware (what would constitute a classic horizontal scaling within the database tier). A typical deployment would see major business applications each having its own application server cluster (middle tier) and a

separate database instance on a large common database server, with database replication (but not concurrence) on a fail over server. All of them, however, might address the same cluster of highly protected security servers (a separate layer within the middle tier) for authentication and authorization.

4 COLLAPSIBLE TIERS?

Practical considerations often play havoc with theoretical designs, and e-business is no exception to that. We have already mentioned that different groups within the company may approach the task of a successful deployment of a business application in a different way. Controversies abound and pitched battles are often fought around the number and delineation of tiers, not to mention detailed decisions regarding individual software and hardware technologies. Each successful deployment often reflects inter-group cooperation worked out over the prolonged conflict of ideas and objectives. On one hand, systems architects tend to push for an expansion and separation of layers based on their functionality and scalability in order to conform to their view of the enterprise systems and business models. On the other hand, powerful business and applications development groups tend to lean towards collapsing the number of layers and lowering the number of hardware and software components based on their view of performance targets and the desire to minimize the number of potential points of failure. Their argument often runs along the lines that if a single component's (e.g. network; a single server; a message queue, etc.) failure can doom the application, then the result is not much different from a failure in a "combined" solution (e.g. a powerful server), where all tiers (presentation, business logic, databases) run within the same hardware. Even if the argument is grossly overstretched – each tier, when properly built, has enough redundancy to keep the business running in case of component failures, and tiers may have to be separated for many other reasons – proponents of centralized deployment (collapsed tiers) may still score points on performance measures. This author spent many months capturing and correlating business performance metrics (each set of metrics individually designed, measured, and captured for each business application) with traditional systems measurements in order to re-balance the tiers to address performance bottlenecks. Network latencies in the order of milliseconds between fully separated tiers, when added over hundreds of checks and queries comprising a single unit of work, were often found to be unacceptably long.

The range of possible practical solutions, reflecting different philosophies, systems architecture approaches, or just day-to-day pressures, of different groups (architects, developers, business groups, managers, quality control, etc.) is indeed large. It is not uncommon to find senior managers even in very e-business oriented enterprises – usually from IT development and from business groups - arguing earnestly for "collapsed layers" solutions, often to the despair of, and strong opposition from, systems architects, security experts, data modelers, and the like. On the other extreme, full tier separation model can result in 6 to 8 separate hardware and software tiers for each business application: from routing and 'traffic cop' tier, through static web servers tier, security 1 (authentication), dynamic web servers tier, security 2 (authorization), multiple business logic tiers (depending on the complexity of the underlying business), down to the persistence layer.

Latest developments in the operating systems zones or containers (the latter's definition being "zones with resource management capabilities") make it possible to merge the two approaches to a certain degree. Figure 4 illustrates one such zone based e-business solution, where each of the four tiers (presentation, security, business logic, and persistence) runs within a separate container on the same piece of hardware, with resource quotas and independent network addressing for each container. Zones are indeed fully separated from one another, although they can – but do not have to – rely on the same core set of OS binaries, and they obviously share physical resources of the underlying hardware. Simplicity of their very tightly knit network, pooling of resources while guaranteeing their fair share through zone quotas, potential savings on various software components, streamlined management of both the infrastructure and of the business application, and many other factors contribute to the relative recent popularity of this approach. Redundancy is achieved by duplicating the entire system at the business continuity site and by clustering individual components (e.g. application servers) between the two sites, thus contributing to solving two problems at the same time. This may very well signal the future for e-business systems architecture.

5 CONCLUSIONS

Systems architecture of e-business solutions in Tier One corporations (e.g. banks and large brokerage firms) has been undergoing evolution reflecting interests, perspectives, and point of view of major business and IT players. Compromise solutions are often reached regarding the number and level of logical and physical separation of different tiers. Required performance levels, security concerns, legal compliance, application development cycles, business functionality, scalability across various components, business continuity in case of a disaster – all these and many other considerations contribute to the final systems model of the corporation's e-business. It is the enterprise architect's role to reconcile often conflicting interests of various players without loosing the sight of a well functioning, fully scalable, and secure e-business solution. Recent advances in the operating system's zoning within the same hardware allow building of separately addressable containers with reserved resource allocation. This, in turn, allows systems architects to build logically separate e-business tiers within a "collapsed tiers" model from the hardware and network points of view – an example of which can be found in Figure 4. Cloning of such solutions across the boundaries of primary vs. business continuity sites, or across geographically dispersed primary corporate sites, leads to very neat and compact architectures that combine performance and high resilience to failures with a high degree of maintainability and manageability.

REFERENCES

McGovern, J., Ambler, S., Stevens, M., Linn, J., Sharan, V., Jo, E., 2003. A Practical Guide to Enterprise Architecture, The Coad Series, Prentice Hall.

Harmon, P., Rosen, M., Guttman, M., 2001, Developing E-Business Systems & Architectures, Morgan Kaufmann Publishers

Figure 2.
Vertical scaling: a 3-member middleware cluster deployed on more powerful hardware and running in a larger JVM. All applications address the same cluster of application servers.

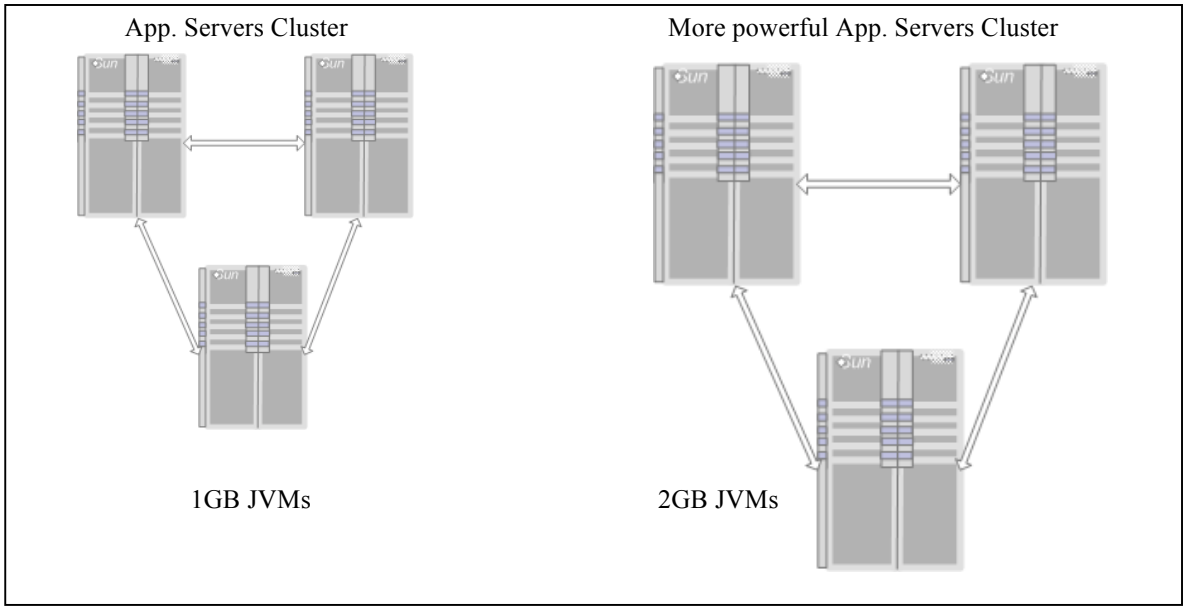


Figure 3.
Horizontal scaling: one 3-member cluster replaced with two such clusters (on identical hardware and running identical JVM). Individual applications address different clusters.

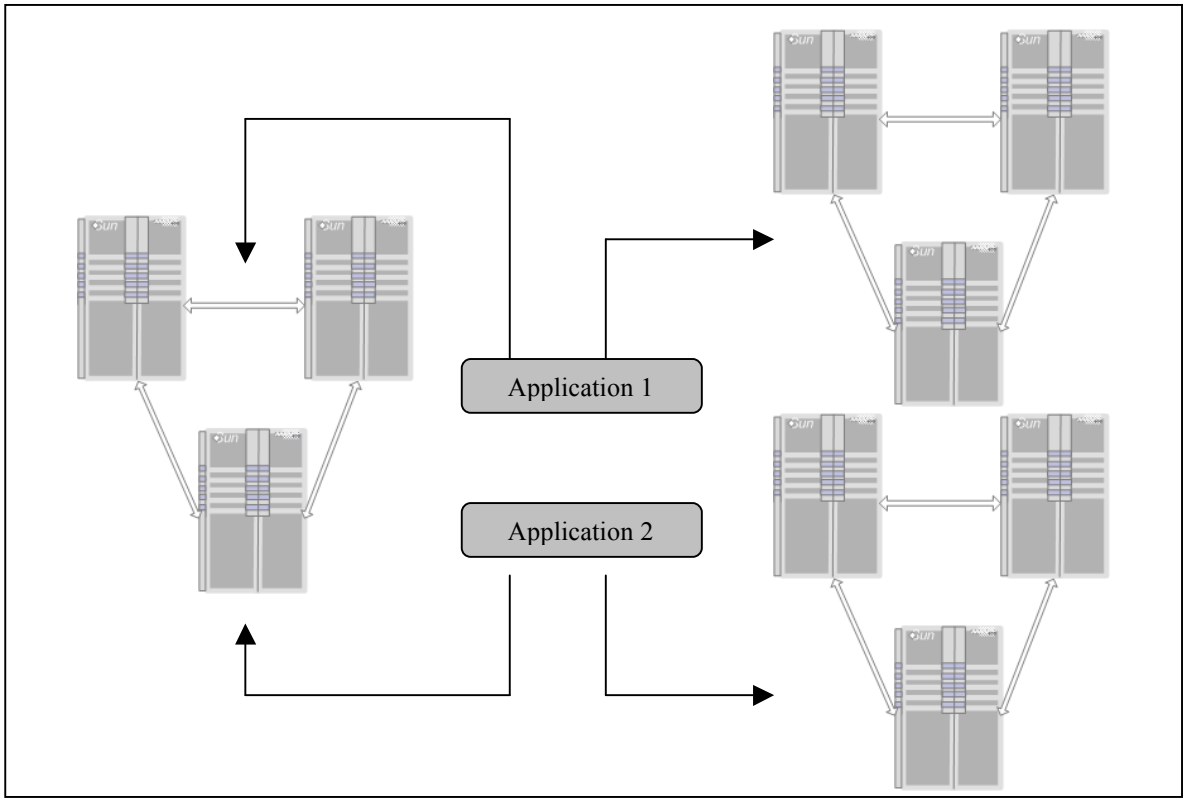


Figure 4.

Operating system's zone/container based e-business solution (all tiers collapsed to single hardware). Each zone/container has a separate IP address and is independent of other zones. Resource quotas are assigned to each container.