Adapting eFinance Web Server Farms to Changing Market Demands

Ronald Moore/Achim Müller/Ralf Müller/Klaus Temmen
IS Innovative Software AG
Sandweg 94, 60316 Frankfurt am Main
{ronald.moore | amueller | ralf.mueller | klaus.temmen}@isg.de

Abstract: The evolution of the eFinance Market presents new and changing requirements on web server farm architecture. Web server farms built during the boom years of the web were designed to provide service to a very large number of users, where each request however placed relatively little load on the system. Further, the requests displayed a large amount of statistical similarity, so that *caching* mechanisms could be successfully applied. However, as professional eFinance tools migrate to web-based ASP technology, and as different companies cooperate to build these tools using the *Web Services* paradigm, these basic assumptions no longer hold: The number of users decreases, while the demands placed by each user increases. Further, each user places highly specialized demands on the application, decreasing the similarities between different requests. The success of conventional caching techniques drops even further if the web server farm is now called upon to provide XML-based *Web Services* instead of HTML web pages.

This paper analyzes these new requirements on the basis of a case study: We present the design process behind the addition of an *On The Fly Calculation Server* to IS Innovative Software's Web Server Farm architecture. This new server has been developed to provide advanced financial (MPT) calculations to professional users. The resulting design represents a form of *intelligent memory*, optimized to minimize the movement of data and thus request latency. The stages in the development of this new server, and the integration of this server into the existing web server farm, are presented.

Keywords: Web Services, eFinance, high performance web applications, caching mechanisms, web server farm architecture.

1 Introduction

Changing market conditions are leading to new requirements for eFinance web servers, and the architecture of financial web server farms is changing to meet these requirements. As a case in point, this paper reviews the design of a new financial calculation server, and shows how this design mirrors new developments in the eFinance market.
In a previous paper [CMM02], we have shown how the concept of *active caching* plays a central role in the financial web sites developed, hosted and presented by the web server farm of IS Innovative Software. This cluster has been developed to meet the demands of financial web sites for low latency and high availability despite very high peak loads. This paper shows how the migration of financial applications onto web-based, ASP (Application Service Provider) platforms presents new challenges and requirements, and how these challenges in turn influenced the design of a new calculation sever for advanced financial applications.

Further, just as the concept of *cache memory* was taken from traditional computer architecture and adapted to the field of web server farm architecture, the new extensions to web server architecture are reminiscent of the computer architecture concept of *intelligent memory* (IRAM) [Pa97], in which the distinction is dropped between data servers and computation servers.

1.1 Changing Market Forces

Market forces are driving professional eFinance applications onto the web server platform and into the ASP paradigm.
This has two sides: Web site providers are facing an increasingly competitive market for conventional web sites, and thus must search for growth potential in up-scale, value-added professional markets. At the same time, providers of professional applications are drawn to the ASP platform, because its inherently networked nature allows for the quick and cost-effective dissemination not only of rapidly changing data, but also of the applications that operate on that data.
Further, networked platforms offers companies new ways to cooperate when building such distributed applications, with interfaces defined and implemented on the basis of XML and the *Web Services* paradigm [Ce02].

1.2 Changing Challenges for Web Server Farms

Conventional financial web sites are called upon to provide web pages with a minimal latency to very high numbers of users. Further, financial web sites are called upon to provide dynamically changing, often personalized, information (most typically, but not only, current prices). Conventional database systems cannot meet these requirements.
Fortunately, the similarities between the requests made by different users present ideal conditions for *cache memory techniques*. Cache memory logic works only when statistical regularities exist in the memory usage: cache designers speak of *temporal* and *spatial locality* [Mo02]. These techniques apply well to conventional websites, due to the strong statistical properties of their usage, as long as advanced techniques are used to accommodate the demands for fast invalidation and updating of dynamically changing data. Application-specific techniques can be used to capture statistical regularities unique to different types of web traffic [CMM02].
When however conventional web server farms are extended to serve XML as a part of ASP applications, new conditions prevail. The number of users is significantly lower, while the statistical similarity of their demands on the application decreases, since each user is *working* with data, and not just *browsing* it. This means that both the temporal locality displayed by one user's requests, and particularly the temporal locality displayed in the requests of *groups* of users, decreases. Further, XML interfaces are much less predictable than HTML interfaces. The constraints of good web page design do not apply. This makes it harder to characterize the *granularity* of the data movement.

2 The Design of the "On The Fly Calculation" Server

This section illustrates the consequences of the changing market forces on the development of IS Innovative Software's new *On The Fly Calculation* (OTFCalc) Server.

2.1 The Architectural Context: Existing Financial Calculations Servers

Conventional web users are usually content with a fixed set of analytic figures, as long as these figures are updated on a regular basis. As such, calculations for the web market can be performed in an *event-driven* fashion, every time new price data arrives from the data feeds, and *not* on demand. This led to the calculation architecture illustrated in the Figure 1.

Financial data streams (e.g. prices) are parsed and interpreted by *feed handlers* and/or *importers.* The data is first stored in a "Tick Server (TickServ)", specialized to support very fast retrieval, sorting and ranking of dynamically changing data. End-of-day histories are extracted and stored in a specialized "*History Server* (HistServ)". Analytic figures are derived from these histories and added back to the tick server, where the web

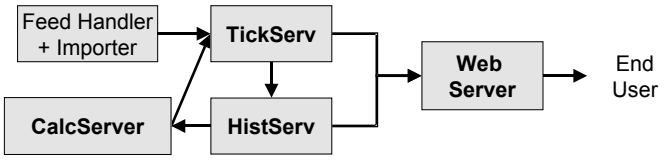

Figure 1: The portion of the previous architecture responsible for financial calculations. Arrows show dataflow (the flow of requests is not shown).

server can obtain them upon request. The web server includes a variety of application-specific cache memories designed to reduce the amount of requests actually sent to the tick server. All of the hardware components are replicated to provide both fail-over security, as well as parallel computation for increased performance. The architecture is, in other words, highly scalable.

Since the set of all calculations provided is of a reasonable size, it is possible and efficient to update all the financial figures as soon as new end-of-day prices are added to the history Server. Much care is taken to invalidate out-dated calculations in the web servers' caches [CMM02].

2.2 Functional Specification of the On The Fly Calculation Server

As part of a cooperation with a supplier of professional financial analysis applications for the funds market, IS Innovative Software AG needed to adapt its web server farm architecture (as described in [CMM02]) to allow professional users to perform highly variable financial calculations on funds performance data. The new service needed to be able to supply two forms of output: performance histories in arbitrary currencies, and *calculation matrices*.

Each row of a calculation matrix represents one fund's history, and each column represents a different calculation. Thus, each element of the matrix contains the results of applying one calculation to one fund's history.

The calculations included both basic investment scenarios (e.g. savings plans, withdrawal plans) and MPT (Modern Portfolio Theory) [FGM02] calculations. Each calculation (column) is specified by parameters such as currency, sample frequency, time range (start and end dates) and choice of benchmark.

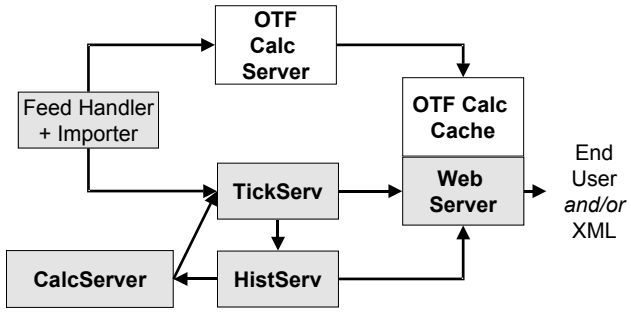

Figure 2: The Extended Architecture.
New portions are white, previously existing portions are in gray.

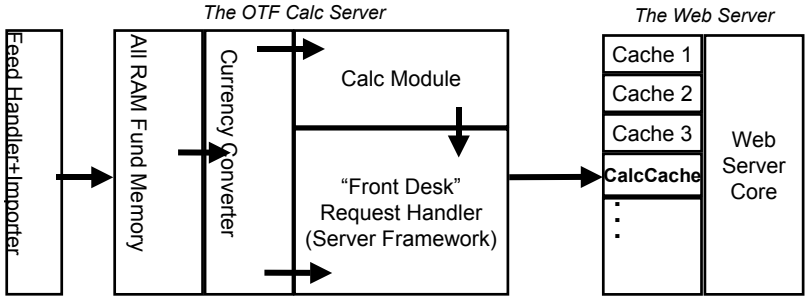

Figure 3: The Components of each Server.

The basic requirement for performance was that a matrix containing several hundred rows (funds), and some dozen columns (calculations), must be returned to the user in well under five seconds.

2.3 The Design of the OTFCalc Server

Since professional users need to be able to vary any and all of the parameters of each calculation (e.g. currency, benchmark), a new server is required to produce results *on the fly*. The space of all possible permutations of the parameters (and thus the number of figures which the architecture must be prepared to serve) is much too large to provide in an *event-driven* manner.

The design of the new, extended architecture is illustrated in Figures 2 and 3.

The first design decision was to store the fund histories separately, and to store them all in RAM. Neither conventional database servers or the existing HistServ would have been fast enough, given the large number of histories involved in a single request. Further, RAM is meanwhile so inexpensive, that the option to hold all the histories in RAM, unthinkable only a few years ago, is now easily affordable.

The next design decision was to incorporate the currency conversion functionality directly into the server which stored the funds' histories. This means that the currency exchange rate histories must be stored with the funds' histories, but eliminates the need to

move these through the network. This represents the first step to a server architecture similar to the concept of *intelligent memory* (IRAM) [Pa97]: Computation is seen as a value-added feature of a data server, and not as a server in its own right.
Various experiments were performed to establish the relationship between data transfer times and calculations times. All experiments demonstrated conclusively that the calculation times were completely negligible in comparison with the time required to move the data between the servers. This observation had two implications: First, plans to precalculate, in an event-driven fashion, the most often requested intermediate results were abandoned. Second, it was decided to integrate even the calculation of the matrices into the new history server. This resulting server design, which combines a specialized storage of the funds' histories, currency conversion and all the calculations, was named the *On The Fly Calculation* (OTFCalc) Server. This server can either be called upon to provide funds' histories, converted to various currencies, or to provide calculation matrices based on these histories (compare Figure 3).
It would have been possible to go even further and integrate a cache memory into the OTFCalc Server, but it was decided instead to integrate such a memory into the *web server*. This was done for several reasons: Most importantly, the only significant statistical regularity to be expected exists between the different requests made over time by each individual user. Since the web servers' load balancer can usually map one user's requests onto the same physical web server, the web servers have a much better chance to discover and exploit this statistical regularity.
Another reason to put the *Calc Cache* in the Web Server is the synergy between two necessary forms of preprocessing: The arbitrary size and shape of the XML requests is unsuitable for communication between the web server and the OTFCalc Server. Therefore, the requests are dissected into blocks with a given (but configurable) maximum size. This preprocessing step can be easily combined with the manipulations performed when the *Calc Cache* recognizes that individual calculations are stored in the cache. These preprocessing steps continue the trend toward specialized, application-dependant cache logic (cf. [CMM02]).

2.4 Preliminary Results and Plans

Both the *OTFCalc Server* and the *Calc Cache* component for the web server have been implemented, and while they have not yet been tested under realistic loads, preliminary tests have to date consistently demonstrated response times well under the required 5 second threshold, with matrices with over seven thousand rows. Further, one OTFCalc server can (so far) easily hold all the histories for all available European funds so far.
Planned work includes testing different data distribution methods (which will become as the number of funds in the system increases), and more advanced caching logic in the *Calc Cache*, once we have sufficient experience with realistic loads to determine what statistical regularities exist (if any).

3 Conclusion

This paper demonstrates the effects changing market conditions have had on the financial web server farm architecture developed and employed by one commercial web service provider. Market saturation in the web realm and continuing cost pressure in the financial application market motivates the migration of professional applications onto web-

based ASP platforms, which in turn accelerates the adoption of XML-based interfaces and the Web Services paradigm. These two developments change the functional requirements placed on web server farms, leading to new, streamlined architectures.
Just as conventional web server farms adapted the concept of *cache memories* from traditional computer architecture, we are now approaching an architecture very similar to the concept of *intelligent memories* (IRAM) [Pa97], in which processors and memories are merged. Affordable RAM and ever increasing CPU performance in reasonably priced (PC-class) servers leads to the melting together of calculation and data storage functionalities. Both the technological and the economic trends are thus leading to designs where calculations represent *value-added enrichments* to highly specialized data servers.

References

[Ce02] Cerami, E.: Web Services Essentials. O'Reilly UK, 2002.

[FGM02] Fabozzi, F.; Gupta, F.; Markowitz, H.: The Legacy of Modern Portfolio Theory. In:Journal of Investing, Fall 2002, p. 7-20.

[CMM02] Cotoaga, K.; Müller, A.; Müller, R.: Effiziente Distribution dynamischer Inhalte im Web. In: Wirtschaftsinformatik 44 (2002) 3, p. 249-259.

[Mo02] Moore, R.: SDAARC – A Self Distributing Associative Architecture. Shaker Verlag, 2002.

[Pa97] Patterson, D. et al.: A Case for Intelligent RAM: IRAM. IEEE Micro, April 1997.