Literature review:

A Comparative Analysis of LAMP and Microsoft.NET Frameworks.

By: Christo Crampton
Supervisor: Madeleine Wright

Abstract:

This paper seeks to critically evaluate contemporary literature regarding the topic. To assist an objective analysis, we first examine some important issues regarding the development of large-scale web-applications namely optimization, security, architecture and web services. Thereafter, literature regarding each of the components which make up the LAMP and .NET web development frameworks, is objectively and critically analysed, and an argument highlighting both the strengths and weaknesses of the technologies is presented. Finally a conclusion detailing the relative strengths, weaknesses and suggested application of these technologies, based on the literature, is presented.

Table Of Contents:

1.) Introduction

In my review of the literature regarding the *Comparative Analysis of the LAMP and Microsoft.NET Frameworks* I have adopted a two-pronged approach**.**

First I have covered the generic areas of importance to the topic – the *principles of the web based enterprise*. These areas; such as optimization, security and architecture are relevant to enterprise web applications in general and a firm understanding of the intricacies of these subjects aids comprehensive analysis of the frameworks.

Finally I have gone into some detail on the specific components which make up the LAMP and .NET stacks respectively – namely: PHP (Hypertext Pre-Processor), MySQL database server, Apache web server – the LAMP stack; and ASP.NET, Microsoft SQL Server and Internet Information Service (IIS) – the ASP.NET stack.

2.) Understanding the Web-based Enterprise

2.1) Defining the Web-based enterprise

"Enterprise Systems (ES) are sold as comprehensive software solutions that help to integrate organizational processes through shared information and data flows [Shanks and Seddon, 2000]. The integration of core business functions, including finance, logistics, and human resources, is achieved through the creation of a single system with a shared database" [editorial. 2004].

The web-based enterprise represents the public interface of an enterprise with the world through the internet. Using the internet, the enterprise is able to distribute its services widely throughout the world. Web applications enable successful communication with customers and businesses through a controlled interface. Shared

online databases with server-side front-ends improve accessibility of data, and facilitate multi-user concurrent interaction and access to data.

Furthermore, recent developments with web services and service-oriented architecture (SOA) mean that the web and enterprise web applications are playing an increasingly important role in facilitating B2B (Business to Business) and B2C (Business to Customer) services which can aid interoperable communication between existing legacy and client-side business components using the internet.

3.) Principles of the Web-based enterprise

3.1.) Optimization

Optimization becomes of paramount importance when delivering enterprise level products over the internet. Although modern advances in computing often mean that improved performance can be achieved by simply improving hardware, given a certain hardware configuration further optimization can be achieved through good software configuration and intelligent coding.

Although optimization is important, Fuerks[2] [2003. pp: 269-70] stresses that, particularly on the coding side, optimization comes at a cost, and we need to be careful not to *over*-optimize at the expense of readability, maintainability and security. He suggests that optimization should not be the sole aim but suggests that, when applications are deployed online and performance becomes an issue, this should be dealt with as a final iteration of the design process.

Certain optimizations can however improve performance and security without adversely affecting design; according to Soukup [1997. pp: 399-403] systems should be designed so as to decrease the need for client/server communication. Herrington [2003] elaborates: “By using a logical three-tier architecture and by reducing the number of queries and commands sent to the database, we can get web applications that scale” – Well-implemented 3-tier design (separate logic, data and presentation layers) improve performance. Furthermore, optimized applications place a smaller

load on web and database servers, hence *decreasing* the likelihood of over-burdening the server, and thus improving security. Database features such as stored procedures, triggers and advanced SQL functions make it easier to produce fine-grained n-tier applications and reduce client/server chatter.

Load tests (using Application Center Test (ACT) for ASP.NET applications) [Webb, J. 2003 pp: 538-9] and debugging tools (such as XDebug for PHP) can help identify bottlenecks in code and ease the process of performance tuning applications.

No discussion of web application optimization would be complete without a reference to caching. Caching of web pages is automatically performed by your browser. However, when serving dynamic pages, server-side caching must be used to cache dynamic data such as database recordsets. Both PHP and ASP.NET support a variety of caching functionality, including the ability to cache certain parts of the page. Pages which change infrequently should be cached so as to improve the server's ability to quickly serve content to the user [Webb, J. 2003. pp: 635-51].

Finally, the platform for deployment of an application can also affect the performance of that application. This is something in particular which needs to be considered when evaluating benchmarking data. LAMP (Linux, Apache, MySQL and PHP) and the .NET framework both represent optimal frameworks for performance and interaction of components.

In conclusion: With the massive improvements in computing hardware the emphasis on optimization has decreased with regards to programming tasks. Furthermore, as the complexity of the application increases, so it becomes more important to produce well-designed and maintainable solutions. Good optimization should not come at the cost of design or security, but should be implemented so as to improve and aid design, security and obviously performance.

3.2.) Security

The web-based enterprise commonly represents the most public interface of the enterprise with the world. Unlike offline or internal network applications, web applications are very likely to come into contact with hostile users in addition to the well-intentioned users for whom the application was intended. Therefore, security is of critical importance to web-based applications, and failure to enforce suitable security can be both very serious, and very public.

Shiflett [2004] describes some common attacks on web applications such as: spoofing, cross-site scripting, SQL injection, session fixation and hijacking. Howard and LeBlanc [2003. pp: 366-4] go further, also mentioning canonical representation attacks. The severity of these attacks can range from violation of private information and data – or identity theft - to the loss of or manipulation of data (particularly with SQL injection), and even hijacking of the host machine. Because web applications are deployed in the public domain (the internet), attacks occur publicly and reflect badly on the vulnerable enterprise.

Although security must be understood and implemented, Shiflett [2004 p 4] concedes that there are performance, design and usability overheads associated with security, and these too must be considered. He concludes that the level of security adopted is dependant on the project, but emphasises that at the very least correct data validation should be implemented. Howard & LeBlanc [2003 pp: 207-258; 535-6] elaborate on other standard security-coding implementations such data filtering – using the whitelist approach of assuming all data is invalid till proven valid - secure storage of secrets and sensitive configuration details (such as database authentication details), employing valid and industry-standard cryptography and hashing methods and *running on least privileges* so as to reduce the severity of attacks if and when they do occur.

On top of taking the valid security precautions mentioned above, Howard. H & LeBlanc [2003. pp: 568-9] emphasize the importance of suitable security testing of applications to ensure that sufficient and elegant error handling is in place, and that applications are not susceptible to compromise given unexpected data or activity. When deploying applications it is important that errors are handled subtly and if possible suppressed as they can often reveal useful information which can make

hacking easier [Howard. H & LeBlanc. D. 2003. pp: 561-4]. Furthermore, since a production application is likely to experience problems different to those of the development application (such as problems incurred due to concurrency issues or heavier load), it is also wise to have monitoring and logging tools in place or even built into the application.

3.3.) Architecture

Development for the web typically requires a diverse skill set. The programming and logic skills required by a developer are very different from those of the designer (HTML, CSS, and graphics and aesthetics skills) which are again very different from those of the Database administrator (SQL, DBMS (Database Management System) skills). Consequently it is important that these users can work on their respective components separately and with as little effect on other components as possible. Hence fine-grained architectures are desirable.

To achieve fine grained systems layered architectures are adopted. The two most common n-tier architectures are the 3- and 5-tier architectures. The figures below represent 3-tier architectures using the LAMP stack with the Smarty templating engine (http://smarty.php.net) (**Error! Reference source not found.** [Herrington, J. 2003]); and the .NET stack (Figure 2. [Leake, G. and Duff, J. 2003])

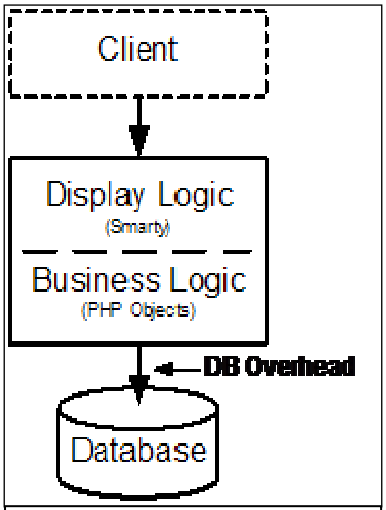


Figure 1 - 3-tier LAMP stack

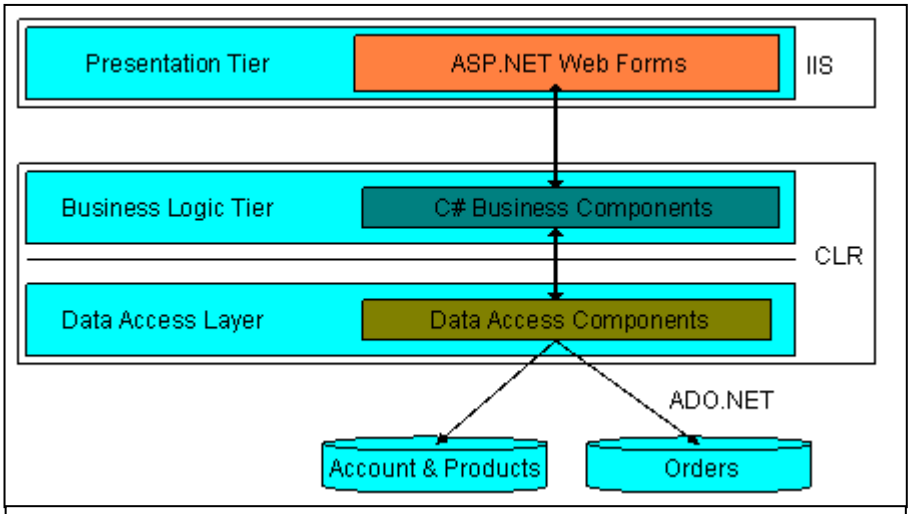


Figure 2 - 3-tier .NET stack

Figure 3 represents a possible 5-tier implementation, illustrating its use within the LAMP and .NET frameworks.

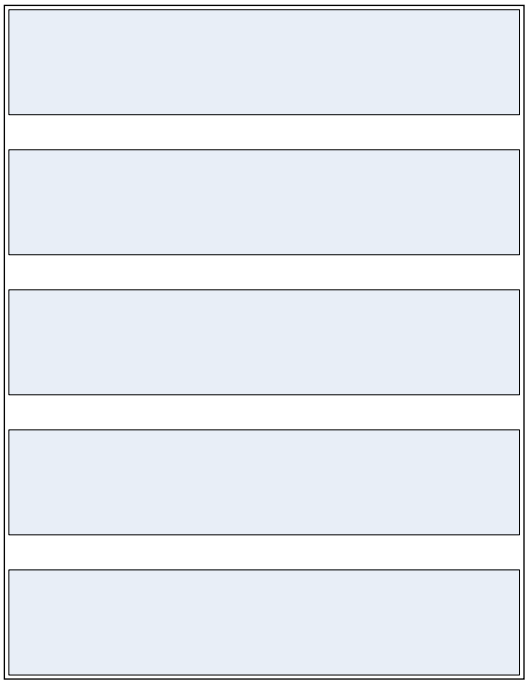


Figure 3 - 5-Tier Stack

In a layered architecture each tier should be able to exist on a physically independent system, and be able to communicate over a network – making it distributable. Layers should exchange information only with the layers directly above and below them through an interface described by an API (Application Programming Interface). Implementation on the layers must be independent of specific technologies and should be easily replaceable with equivalent technologies - for example, the MySQL database in the data layer should be interchangeable with an Oracle or SQL Server database without requiring a substantial change to the data access layer. [Fuerks, H[2]. 2003. pp: 277-9].

Fuerks, H^2 [2003. pp: 279-80] proposes that layered-architecture design improves scalability of applications, and increases robustness in terms of future changes by facilitating a 'plug-in' architecture in which components are (fairly) easily interchangeable.

This fine-grained approach and separation of logic makes applications easier to maintain and debug. Another advantage of fine-grained design is evident from studying the implementation of the ***DotNet Pet Store*** where the application was made more robust, scalable and secure by deployment on multiple servers – enabling load balancing across separate servers. (Figure 4)

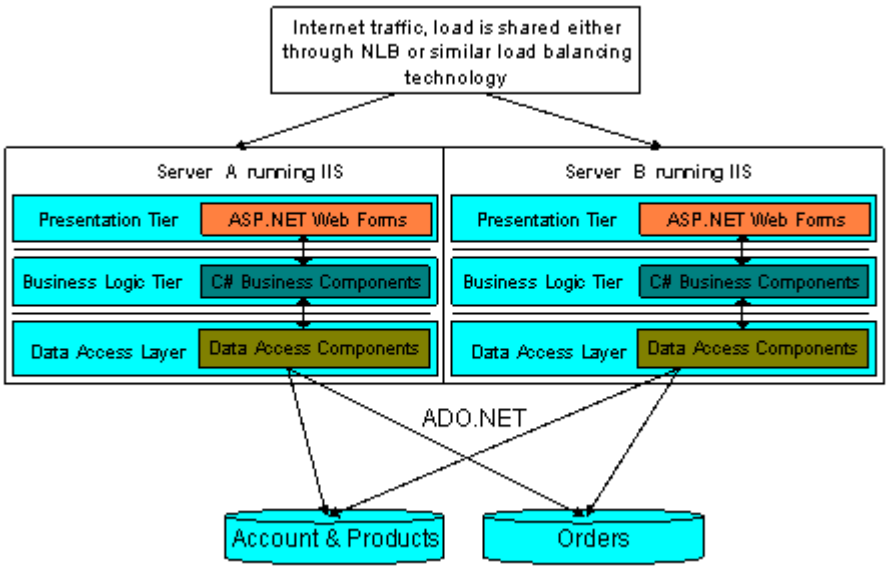


Figure 4 - Physical deployment of the ASP.NET Pet Store

3.4.) XML, Web Services and the Semantic Web

"The XML Web services programming model enables you to build highly scalable, loosely coupled, distributed applications using standard Web protocols such as HTTP, XML, and SOAP" [Trenary, J. 2002: p 369].

Yank [2002], in his simplistic and introductory article acknowledges an important shortcoming of web services: "Everyone will tell you that they are important and great, but no one can tell you why". In response to this, Bussler *et al* [2003] elaborate

on the purpose of web services, suggesting that web services "will make Enterprise Integration (EI) dynamically possible for all types and sizes of enterprise". In addition, Bussler *et al* suggest that web services will make enterprise integration more reliable and easier to achieve. Finally Trenary [2002: p. 370], reveals that: "Web services enable disparate applications to exchange messages using standard protocols such as HTTP, XML, XSD, SOAP, and Web Services Description Language (WSDL)" (see Figure 5).

.

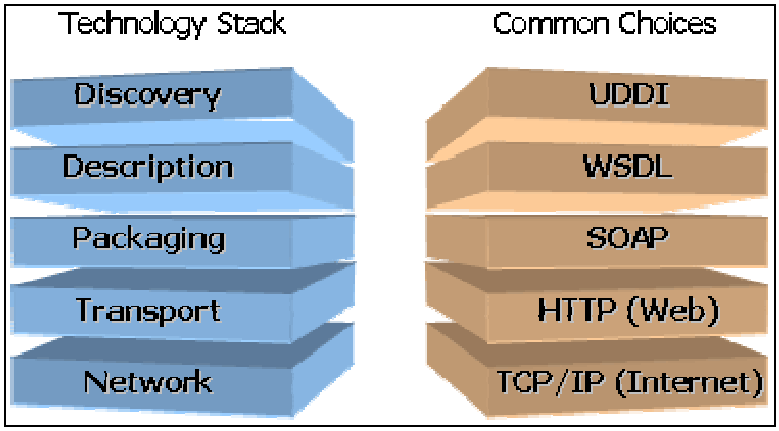


Figure 5 – the Web services technology stack

Yank [2002] refers to web services as "a network interface to applications functionality, based on standard internet technologies" (see Figure 4). As such, web services are highly abstracted business logic. Ternary [2—2: p. 370] explains: the client does not need to know the language in which XML web services are implemented, but only needs to know the location of the service, and the methods which that service makes available. So, using web services, companies can make their API (Application Program Interface) available over the internet or intranet for consumption and use by other client applications of any language and on any platform.

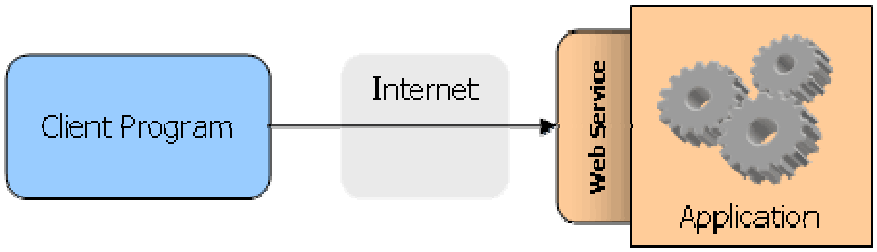


Figure 6 - distributed application functionality over the internet using web services

In conclusion, web services play an important role in providing distributed and scalable applications. As a heavily standardized technology, and with major backing from key vendors such as Microsoft and IBM, we can only expect the use and importance of web services to grow. Potentially web services could serve as an incredibly effective tool to support fine grained B2B and B2C interaction and EI through a controlled API.

4.) Development frameworks

4.1.) LAMP

4.1.1.) PHP 5 Scripting Language

PHP 5 is the latest stable release of PHP, with the latest version, 5.0.4, being released in March 2005. PHP 5 represents a huge improvement in functionality over PHP 4, and has been much anticipated as PHP's answer to the .NET revolution.

Zend [2005], the commercial counterpart to PHP which is largely responsible for promoting PHP as an enterprise-ready web development tool, touts PHP 5 as: "the glue that ties together functionality from the diverse systems to address pressing business needs" [Zend Technologies. 2005]. Although PHP doesn't come with the extensive frameworks for client and server application development that accompany ASP.NET and J2EE, Zend [2005] suggest that PHP represents a complete, efficient and easy solution for rapid web application development which directly interfaces with code written in a number of other languages and databases. Further: "The widespread deployment of PHP…ensures that it has been proven in almost any deployable scenario" [Zend Technologies[2]. 2005]. Zend notes that while PHP 3 and 4

stamped PHP's authority on the internet, it remains to be seen if the improvements in PHP 5 will have the same effect on the enterprise.

Gutmans *et al* [2005] elaborate on the new features in PHP 5: an improved Object Oriented (OO) model; exception handling; support for XML and web services; improved interoperability; a MySQLi (MySQL improved) extension plus extensions for PERL. Gutmans *et al* [2005] also mention the improved infrastructure of the new engine as well as a new and better memory manager.

The new features and improvements in PHP 5 are very exciting and suggest PHP's move towards maturity. However, compared to ASP.NET and J2EE, these new features are still in their infancy, and it is likely that there will be a number of development iterations before these features can be considered mature.

Although it is clear that PHP has its weaknesses, there is little valid academic literature on the topic. Some of the arguments below reflect the opinions of respectable and experienced developers in the industry as posted on technical forums regarding PHP and web development.

Wright [various authors] points out that PHP is a scripting language. Consequently he suggests that languages such as ASP.NET which are compiled (Just-In-Time) by the CLR (Common Language Runtime) - .NET's equivalent of the JVM (Java Virtual Machine) - to an IL (Intermediate Language) enjoy performance gains over scripting languages such as PHP. Fuecks [various authors] responds, mentioning scripting language acceleration engines (PHP accelerators) which effectively cache code on the local disk and speed execution time with similar results to ASP.NET's compiled code. There are numerous benchmarks which have been performed, with varying results on both sides, so that it is difficult to conclusively suggest that one is faster. Therefore we can conclude that the performance difference is negligible.

More convincing arguments regarding PHP's value in the enterprise question not its performance or scalability, but its architecture. Vossos [Vossos, G] talks of companies which: "trade-off good design and architecture for fast-tracked HTTP scripting solutions like PHP". Johanssen [various authors] points out that PHP shifts a lot of the

responsibility onto the programmer rather than the interpreter or compiler because it is a much more tolerant language than other strongly typed languages such as Java in particular, but also C# and some other ASP.NET languages. Crane [2002] elaborates on some of the shortcomings of PHP's tolerance when working in a team of developers: firstly, a member of the team can redefine a function written by another member of the team; secondly, a team member can redefine a *built-in* function used in code written by another developer. He also mentions the fact that, by default, undeclared variables are approved by the interpreter with no warning messages.

This leads onto another major issue regarding PHP as a base for the development of large applications - its lack of distinct or inherent patterns of best practices for development. Johansson [various authors] proposes that whereas ASP.NET will tie you into a strong design pattern for enterprise development with its features such as code behind (code separation), web forms and components, and centralized configuration, PHP leaves this responsibility mostly up to the developer. As Crane [2002] remarks: the result of this is that developers or teams of developers must essentially implement their own techniques using their own tools or the array of tools available as modules for PHP. This lack of an industry standard or standards detracts from PHP's standing as an enterprise-ready technology, and promotes *ad hoc* development. Wright [various authors] and Crane [2002] advance these arguments to suggest that though php *can* be used to develop big web applications, it is not the best-suited tool for the job.

Finally, Crane [2002] suggests that "the language was accreted, rather than ever having been designed". In a similar vein, Voostind [various authors] attacks the lack of infrastructure, framework and toolkit for the PHP language. Both Crane [2002] and Voostind [various authors] criticise the 'cobbled together' nature of the PHP engine and 'class library'. Wright and Voostind [various authors] also point out that PEAR (PHP Extension and Application Library), PHP's 'class library', is a miniscule and poorly structured library by comparison to the .NET class library.

In conclusion, despite its shortcomings, PHP, especially with the major updates in PHP 5 is definitely ready for enterprise development. PHP represents a truly cross-platform, lightweight development language, which - despite its light stature - is

extremely efficient and at the same time powerful. It enables experienced developers to leverage powerful web applications quickly across multiple platforms, easily and with few lines of code. The major shortcoming of PHP is that it allows poor coding practices, which lead to poor design and projects which are hard to maintain. PHP can be criticized for placing too much responsibility on the programmer, and as such may become unsuitable or undesirable for development of large-scale applications by teams of developers.

4.1.2.) MySQL Database Server

In the word of Jack Urlocker [Mytton, D. 2004], Vice President of Marketing at MySQL, the principle strengths of MySQL are that it is: “fast, reliable and easy to use”. Indeed, speed is MySQL’s prominent feature, and has won it high profile customers such as Yahoo! Adobe, Sabre, NASA, Associated Press, Google, Evite/CitySearch and Lufthansa [Mytton, D. 2004].

MySQL’s ease of use is also a feature important in its leverage in the enterprise, says Yahoo!’s Zawodny [Gilmore W.J. 2001]: “We went from experimental to mission-critical in a couple of months. Once others saw it, they jumped on board”.

The current version of MySQL (version 4.1) represents a somewhat featureless yet high performance and reliable database engine, which, in the words of Urlocker [Mytton, D. 2004] does not suffer form from “server bloat”.

The major shortcoming of MySQL when considering its role in the enterprise is its small feature-set and the fact that the current version does not support triggers, stored procedures, views or events [MySQL. 2005. 1.7.5.4; 1.7.5.6], all of which are considered significant in the development of enterprise applications. Borland, in a comparison with their own database server (InterBase), also criticise MySQL’s user-level role management and ‘poor monitoring tools’ [Todd, B 2004. pp: 11:14]. A report by Troels [Troels A.] also suggests that MySQL deviates from the SQL standard in particular by not supporting views, using abnormal joining mechanisms and with the LIMIT workaround. This produces problems with layered architectures

as Hartman [Hartman H. 2001] describes with regard to swapping a feature rich Oracle 9i database for the MySQL equivalent.

A notable feature of MySQL is its support for an array of different table types. The two which are used commonly are MyISAM and InnoDB. According to the MySQL certification guide [DuBois, P.2004, pp: 91-96]; MyISAM tables are incredibly fast for SELECT queries because they feature table locking which essentially converts data to a fast compressed read-only table. However, MyISAM tables are slow in terms of UPDATE, INSERT and DELETE queries, and they don't support data integrity measures – most notably foreign keys. This is a very serous omission, and in my opinion makes the use of MyISAM tables almost redundant in a serious database as they require application programming to ensure referential integrity. This is unreliable, and requires overly tight-coupling with the application layer; InnoDB tables are slower than MyISAM tables (although according to Horn (2003), even the 'slower' InnoDB tables are comparably faster than most other database engines). InnoDB tables support data-integrity features, use row level locking, and are the most attractive table type to use when dealing with large MySQL databases.

Finally, the new MySQL 5 (currently in BETA) is set to offer a number of the features (see Table 1) which have been sorely lacking in older versions, namely, stored procedures, triggers and views as well improved support for clustering and a data dictionary. However, even though MySQL 5 is due to ship shortly, these features are still very new in the MySQL engine and it will be for a while before they can be considered as solid and reliable as features which have existed in other databases such as *Microsoft SQL Server* for a long time.

Table 1 - MySQL Roadmap

Feature	**MySQL Series**
Unions	4.0
Subqueries	4.1
R-trees	4.1 (for `MyISAM` tables)
Stored procedures	5.0
Views	5.0
Cursors	5.0

Foreign keys	5.1 (implemented in 3.23 for `InnoDB`)
Triggers	5.0 and 5.1
Full outer join	5.1
Constraints	5.1

(MySQL[2]. 2005)

In conclusion, MySQL excels in terms of speed, and reliable data storage. However, its shortcomings in terms of important features such as stored procedures, views and triggers make it somewhat less of a viable contestant in the enterprise database space, although it does have its strengths and has been adopted by some notable companies on a fairly serious level. The new features of MySQL 5 will push it into the domain of other more prominent database servers, but it will be some time before it can hold its own against database servers like SQL Server and Oracle.

4.1.3.) Apache Web Server

According to Netcraft, the Apache web server is currently the most popular web server in the world dominating roughly 70% of the server market (see Figure 7)

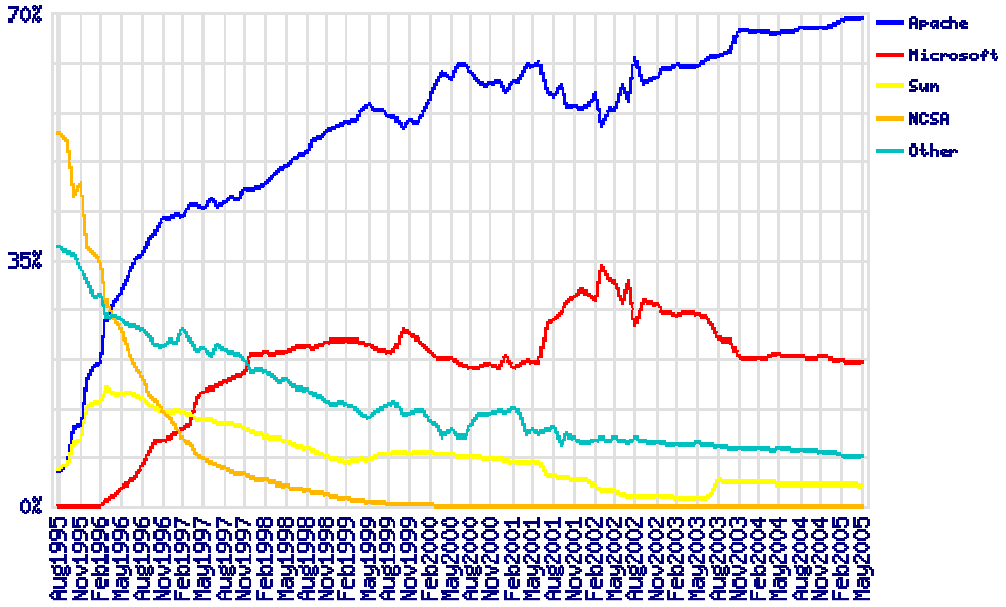


Figure 7 - Netcraft Server Survey - Market Share for Top Servers Across All Domains August 1995 - May 2005

Apache gains its popularity from its tight security record (in particular compared to Microsoft's IIS (Internet Information Services) - the next most popular server) and its low price (Apache is open source and free). Apache is cross-platform and supports a number of different languages (PHP, Perl, Python, Java, C/C++) and databases (MySQL, PostgresSQL and Oracle) [Varner, P. 2001].

The major short-coming of Apache is the relative difficulty of administration due to the lack of GUI administration tools for the server. Most configurations take place in a central httpd.conf file or on a folder level using .htaccess files

4.1.4.) Conclusions on the LAMP development framework

The LAMP stack represents a lightweight high-performance solution to web development. The version 5 releases of both PHP and MySQL (still in BETA) show a drive towards increasingly sophisticated architecture and functionality. Although these technologies are somewhat behind in terms of this, the lightweight framework *is* extensible and powerful enough to be used to develop sophisticated applications, and eases the development process. At the same time it allows the developer flexibility as applications can be effectively leveraged over many platforms. Thus LAMP provides the platform independency of a J2EE application without the bulk and complexity overhead. However, with the LAMP stack, the developer must forgo the extensive architecture and intricate frameworks which ease the process of enterprise development and design.

4.2.) Microsoft .NET Framework

4.2.1.) ASP.NET

Firstly, when talking of ASP.NET it is important to define exactly what ASP.NET is, as failure to do so can result in quite a level of confusion. Firstly, ASP.NET is not .NET. .NET represents the entire .NET framework which is Microsoft's future

platform for development. ASP.NET is the part of the .NET framework responsible for dealing with web applications, web services and XML. In particular ASP.NET uses the `System.web` namespaces which deal with web-specific items [Webb, J. 2003. p. 5].

A major improvement of ASP.NET on classic ASP is the CLR, which is responsible for compiling ASP.NET pages to IL. According to Hartman [2001]: "Microsoft claims that ASP.NET pages are five to seven times faster than classic ASP pages". The controversial Pet Shop applications produced using the .NET and Java technologies also shows the impressive speed of operation of the .NET platform with some benchmarks claiming ridiculous performance advantages over J2EE as Wampler [2001] points out: "Microsoft claims that the .NET version requires one-third the lines of code (LOCs) and provides *28 times* faster average response times (for 450 concurrent users)". Although these benchmarks are almost definitely proprietary hype, they do suggest that major improvements to performance have been made.

Performance aside, ASP.NET also comes with a much improved architecture. Anderson *et al* [2002. p 6] elaborate on the improved architecture in ASP.NET which is much more modularized and component based: "Every page becomes a programmatically accessible, fully compiled object, and takes advantage of techniques like object-oriented design, just-in-time compilation, and dynamic caching" [Anderson *et al* 2002. p 6]. With ASP.NET and its modularized approach Microsoft are beginning to blur the lines between windowed application development and web development.

Another major drive within ASP.NET is towards web services. Microsoft use web services extensively in their own technology (a good example is the Microsoft .NET passport which uses web services to store user credentials across the Microsoft web farm). Microsoft has played a major role in promoting web service technology. Furthermore, Microsoft also supplies a repository of their own web services at http://uddi.microsoft.com/ [Parihar, M. 2002. p 400]. With Visual Studio, Microsoft also provide tools to easily create, consume and discover web services which hide a number of complexities concerned with web service creation, consumption and

deployment [Trenary, J. 2002]. These tools automatically create and store various files such as .wsdl (Web Service Discovery Language), .disco (discover documents) and .vsdisco (Visual Studio discover) files. This eases web service applications development by allowing the developer to concentrate on the business logic instead of the intricacies of deployment, *but* also allows developers to deploy web services without properly understanding what they are doing. In particular, .vsdisco files (which enable Visual Studio to search and find your web services) may not be desirable with a web service deployment, but these are added by default when services are created with Visual Studio tools. Microsoft's strong support of XML web service technology is not surprising as it represents an opportunity to improve cross platform as well as cross language interoperability between Microsoft products and other platforms and technologies.

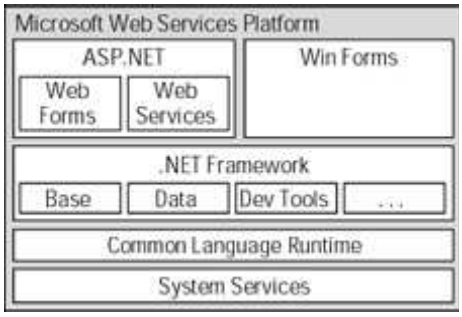


Figure 8 - Microsoft web services platform architecture

Web services are enabled though the `System.Web.Services.*` namespaces. Support for web service discovery and creation is built into ASP.NET and requires no third party packages or tools. [Parihar *et al.* 2002. pp 406-7].

The major short-coming of ASP.NET is that it is (functionally) tied to the Windows operating system and IIS. Although there is work on the Mono project and Rotor, a CLR for Linux, none of these technologies is near production release). This is a major drawback to the success and usefulness of the technology - especially considering that 60% of web servers are running Apache not IIS.

ASP.NET presents a complete web-application development suite for the Microsoft platform which is extremely powerful, yet easy to configure, install and use for development. Although ASP.NET is not as easy to learn as scripting languages such as ASP or PHP, ASP.NET eases the development of complicated and large-scale web

applications by using neat modularization and componentization (with the help of web controls) and a complete and well documented class library - all of which promote good design practises. ASP.NET represents a very viable option for large-scale web-application development within a Microsoft environment, but unfortunately is not an option under other environments.

4.2.2.) MS SQL Server

Microsoft SQL Server is a fully featured database server. Soukop [1997. pp 449-527] speaks of some of the more important features namely support for batches, ACID (Atomicity, Consistency, Isolation, Durability) compliant transaction management, stored procedures and triggers. These features are well established in SQL Server and can be considered reliable. Soukop [1997. pp 399-488] also mentions SQL Server's support for Transact-SQL, a SQL-Server pseudo programming language which allows the user to "use standard SQL as a programming language to write logic within the database engine" Soukop [1997. p. 399]. All these tools improve the ease of producing a layered architecture and reduce the need for client/server communication in applications. Additionally, Howard and LeBlanc [2003. pp: 402-3] suggest that stored procedures are a useful means of helping prevent SQL injection.

Rankins *et al* [2002. pp 45-6] elaborate on some of the new features in the 2000 edition (SQL Server 7.0), importantly XML support, indexed views, multiple SQL Server instances and some new data types. Microsoft currently has SQL Server 2005 in BETA 2 production.

SQL Server is a full-featured enterprise database capable of performing extremely well on the windows platform; however, once again, the server is (functionally) tied to the Windows platform. Furthermore, application performance is highly optimized towards Microsoft technologies and the .NET framework (see

APPENDIX III – Database benchmarking). From the benchmarking performed by e-week.com we can see that SQL Server performance was improved from serving roughly 200 web pages per second using a Java script to 700 using ASP.NET, similarly response times dropped from 100+ to 24. Using these results we can deduce that within the correct (Microsoft) environment SQL Server can perform equitably if not preferably to other databases, yet its performance is hampered greatly once it is taken out of its preferred environment.

In conclusion, similar to ASP.NET, SQL Server is a robust and feature-rich database server, and when used within its optimal environment performs incredibly well, but it is tightly tied with Microsoft technologies, not only to the platform, but even the language or runtime environment of the application which is accessing it. Hence, SQL Server is a good option for building .NET enterprise applications, but is unsuitable for development outside of Microsoft proprietary systems.

4.2.3.) IIS – Internet Information Systems web server

IIS web server is definitely the weakest component in the .NET framework for web application development and deployment. Of particular concern is its poor security track record (see Figure 9).

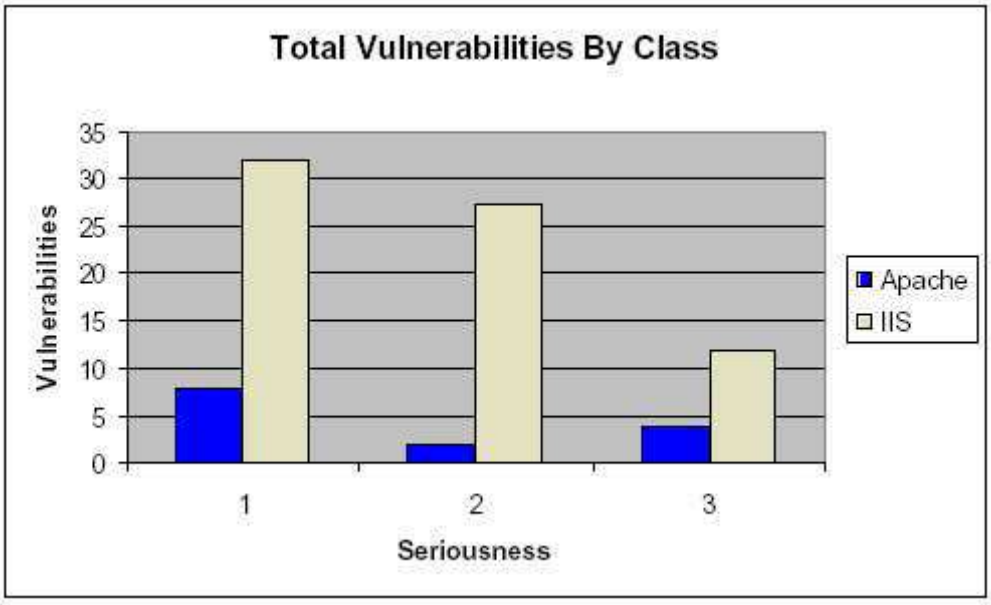


Figure 9 - Apache vs. IIS vulnerabilities (1 is most serious)

Although the latest version of IIS (version 6) has addressed many of the issues - hence the recent 0.25% increase in relative popularity against Apache [Netcraft. 2005] - IIS still has a reputation for poor security.

On the positive side however, IIS does offer simple administration and monitoring tools, and is easy to manage through the IIS GUI.

4.2.4.) Conclusions on the .NET web development framework

The .NET web development framework provides a complete development solution with solid performance and architectural infrastructure for the development of scalable high performance large-scale and maintainable web applications within a Microsoft environment. Unfortunately the performance of .NET components outside the Microsoft environment is substantially lowered, even dysfunctional. Although this isn't a concern if all development is within the .NET framework, it does tie resources to Microsoft and in such reduces flexibility for future changes.

5.) Final Conclusion:

The development of large-scale web applications is about more than just the performance of the technologies. Applications need to be manageable, robust, secure and scalable. Both the LAMP stack and the .NET framework are viable solutions for producing such applications, but as expected, both have different strengths.

The .NET web application development platform offers a fully-featured Microsoft-centric solution to enterprise web development. While this framework represents a complete development solution within a Microsoft corporate, .NET applications perform poorly, if at all, in a non-Microsoft environment. Although there is a strong drive by Microsoft towards interoperability between Microsoft and other technologies, Microsoft products still remain vendor specific and ultimately tied to the Windows operating system. Although Microsoft products are becoming increasingly interoperable, and able to interact with the technologies of other vendors, with the .NET framework environment, Microsoft effectively ties the accepting enterprise to

their products (operating system, web server, and ideally database server). Thus, although .NET is a very solid development environment, the .NET framework limits the flexibility of the enterprise with regards to the adoption of external complimentary technologies.

The LAMP stack on the other hand, while not as robust and architecturally sound as the .NET framework, is incredibly cross-platform and provides a useful tool for rapid web development, while still permitting flexibility in terms of future growth and change of technology. The performance and scalability of the LAMP stack are impressive, and the production of large-scale high-performance web-based applications is possible, however management and maintenance of LAMP projects is not usually as straightforward as with more sophisticated frameworks such as .NET

After extensive research, I feel that of the two frameworks, ASP.NET and LAMP, ASP.NET is clearly the more mature and comprehensive solution for enterprise development. However, the technologies which comprise LAMP - Linux, Apache, MySQL and PHP – are on a drive towards maturity, and certainly are capable of producing high-end solutions to big problems. While ASP.NET is better suited to large scale development, LAMP, while fully capable of tackling large scale applications, is better suited to smaller to medium sized applications, or applications which are likely to require deployment and interaction across numerous different platforms or runtime environments.

References:

1. Anderson, R., Francis, B. Homer, A., Howard, R., Susman, D., Watson, K. 2002. **Professional ASP.NET 1.0**. Wrox Press, Birmingham, United Stats
2. Berkes, D. 2002. **Survey says: PHP passes Microsoft Active Server Pages.** Available at: http://www.newsforge.com/internet/02/06/11/011243.shtml?tid=5 Accessed: 26 May 2005
3. Bussier, C., Fensel, D., Sadeh, N. 2003. **Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce.** Journal of Electronic Commerce (IJEC). Available at: http://www.w3c.or.kr/pipermail/swws-interest/att-0090/01-ijec_special_issue_cfp.pdf Accessed: 23 April 2005.
4. Crane, A. 2002. **Experiences of Using PHP in Large Websites.** Available at: http://www.ukuug.org/events/linux2002/papers/html/php/. Accessed: 31 May 2005
5. DuBois, P., Hinz, S., Pedersen, C. 2004. **MySQL Certification Study Guide**. Sams Publishing.
6. Editorial. 2004. **Understanding the contextual influences on the enterprise system design, implementation, use and evaluation.** Journal of Strategic Information Systems 13 (2004) 271-77. Available at: www.sciencedirect.com Accessed 24 May 2005.
7. Fuerks, H. 2003. **The PHP Anthology: Volume I: foundations**. Sitepoint Pty. Ltd. Collingwood, Australia
8. Fuerks, H.[2] 2003. **The PHP Anthology: Volume I: Applications**. Sitepoint Pty. Ltd. Collingwood, Australia
9. Gilmore W.J. 2001. **MySQL Powers Yahoo! Finance**. Available at: http://www.mysql.com/news-and-events/success-stories/yahoo_finance.html Accessed: 27 May 2005
10. Gutmans, A, Bakken, S, Rethans, D. 2005. **PHP 5 Power Programming. Chapter 2 – Whats new in PHP 5.** Prentice Hall PTR. Available at: http://www.zend.com/php5/andi-book-excerpt.php. Accessed: 16 May 2005.

11. Herrington, J. 2003. **The PHP Scalability Myth.** Available at: http://www.onjava.com/pub/a/onjava/2003/10/15/php_scalability.html. Accessed 24 May 2005.
12. Howard. H and LeBlanc D. 2003. **Writing Secure Code**. Microsoft Press, Redmond Washington.
13. Johansson, M. 2002. **Debate - .NET V. PHP: Top 6 Reasons to Use .NET.** Available at: http://www.sitepoint.com/article/v-php-top-6-reasons-use-net Accessed on: Accessed April 5 2005
14. Leake, G. and Duff, J. 2003. **Microsoft .NET Pet Shop 3.x: Design Patterns and Architecture of the .NET Pet Shop.** Microsoft Corporation. Available at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/petshop3x.asp Accessed: 24 May 2005
15. MySQL. 2005. **MySQL Reference Manual.** Available at: http://dev.mysql.com/doc/mysql/en/. Accessed: 26 May 2005
16. MySQL^2. 2005. **MySQL Development Roadmap.** Available at: http://dev.mysql.com/doc/mysql/en/roadmap.html. Accessed: 26 May 2005.
17. Mytton, D. 2004. **Interview – Zack Urlocker of MySQL**. Available at: http://www.sitepoint.com/article/interview-zack-urlocker-mysql Accessed: 27 May 2005
18. Netcraft. 2005. **Netcraft Web Server Survey**. Available at: http://news.netcraft.com/archives/web_server_survey.html Accessed 27 May 2005
19. Parihar, M. *et al.* 2002. **ASP.NET Bible.** Hungry Minds Inc. New York
20. Rankins, R., Bertucci, P., Jensen, P. 2002. **Microsoft® SQL Server 2000 Unleashed.** 2ed. Sams Publishing, United States of America
21. Shiflett, C. 2004. **PHP Security.** ApacheCon**.** Available at: http://shiflett.org/php-security.pdf . Accessed 2004
22. Soukup, R. 1997. **Inside Microsoft SQL Server 6.5** Microsoft Press, Redmond, Washington.
23. Trenary. J. 2002. **Developing XML Web Services and Server Components with Microsoft Visual Basic .NET and Visual C# .NET.** Microsoft Corporation. Redmond, Washington, USA.
24. Troels, A. **Comparison of different SQL implementations.** Available at: http://troels.arvin.dk/db/rdbms/ Accessed: 25 May 2005

25. Various Authors. Forum available on the internet: **Debate - .NET V. PHP: Top 10 .NET Myths Exposed.** Available at: http://www.sitepoint.com/forums/showthread.php?t=75177 Accessed: 22 May 2005
26. Varner, P. 2001. **Multi-tiered Web Architectures Using Open Source Software.** Available at: www.cs.virgina.edu/~pev56/writing/academic/oss-pres/oss-press-handout.pdf. Accessed 22 May 2005
27. Vossos, G. **Whitepaper: J2EE vs. PHP**. Available at: http://www.livetime.com/docs/PHPvsLiveTime.pdf#search='php%20OR%20shortcomings%20OR%20whitepaper%20OR%20php5'. Accessed: 31 May 2005
28. Wampler, D. 2001. **Cat Fight in a Pet Store: J2EE vs. .NET**. Available at: http://www.onjava.com/pub/a/onjava/2001/11/28/catfight.html. Accessed 27 May 2005
29. Webb J. 2003. **Developing Web Applications with Microsoft Visual Basic .NET and Visual C# .NET**. 2ed. Microsoft Press, Redmond, Washington.
30. Yank, K. 2002. **Web Services Demystified.** Available at: http://www.sitepoint.com/article/web-services-demystified. Accessed: 23 April 2005.
31. Zend Technologies. 2005. **PHP 5: Open Source Scripting for the Heterogeneous Enterprise.** Zend Technologies. Available at: http://www.zend.com Accessed 28 March 2005
32. Zend Technologies[2]. 2005. **PHP and Zend Technologies in the Enterprise.** Zend Technologies. Available at: http://www.zend.com Accessed 28 March 2005

Table of Figures

Tables:

APPENDIX I – Example PHP 5 code

XML Support with SimpleXML:

Consider the following XML file:

```
<clients>
<client>
    <name>John Doe</name>
    <account_number>87234838</account_number>
</client>
<client>
    <name>Janet Smith</name>
    <account_number>72384329</account_number>
</client>
</clients>
```

The following piece of code prints each client's name and account number:

```
$clients = simplexml_load_file('clients.xml');
foreach ($clients->client as $client) {
    print "$client->name has account number $client-
>account_number\n";
}
```

(Gutmans *et all*. 2005)

Web Services:

Consuming a web service couldn't be easier:

The following calls `SomeFunction()` defined in a WSDL file:

```
$client = new SoapClient("some.wsdl");
$client->SomeFunction($a, $b, $c);
```

(Gutmans *et all*. 2005)

Exception Handling:

"PHP 5 adds the ability for the well known try/throw/catch structured exception handling paradigm. You are only allowed to throw objects which inherit from the Exception class.

```
class SQLException extends Exception {
    public $problem;
    function __construct($problem) {
        $this->problem = $problem;
    }
}

try {
    ...
    throw new SQLException("Couldn't connect to database");
    ...
} catch (SQLException $e) {
    print "Caught an SQLException with problem $obj->problem";
} catch (Exception $e) {
    print "Caught unrecognized exception";
}
```

"Currently for backwards compatibility purposes most internal functions do not throw exceptions. However, new extensions are making use of this capability and you can use it in your own source code. Also, similar to the already existing `set_error_handler()` you may use `set_exception_handler()` to catch an unhandled exception before the script terminates."

[Gutmans *et all*. 2005]

Interoperability:

Java

"This example shows how straightforward it will be for PHP code to use a Java Bridge to access a database using the JDBC APIs in J2EE:

```
<?php
$host = 'localhost';
$db = 'test';
$user = 'test';
$pwd = '';
Java("com.mysql.jdbc.Driver"); // init
$conn = Java("java.sql.DriverManager")-
>getConnection("jdbc:mysql://$host/$db", $user, $pwd);
$stmt = $conn->createStatement();
$rs = $stmt->executeQuery("SELECT * FROM news");
while($rs->next()) {
      printf("On %s article %s\n", $rs->getDate("when"),
      $rs->getString("title")
);
}
?>
```

“Java objects are simply invoked using the *Java()* function directly from the PHP code. Once a Java object is instantiated and assigned to a PHP variable, then methods can be invoked on that object as if it were a PHP object, completely transparent to the programmer.” [Zend Technologies. 2005]

PERL

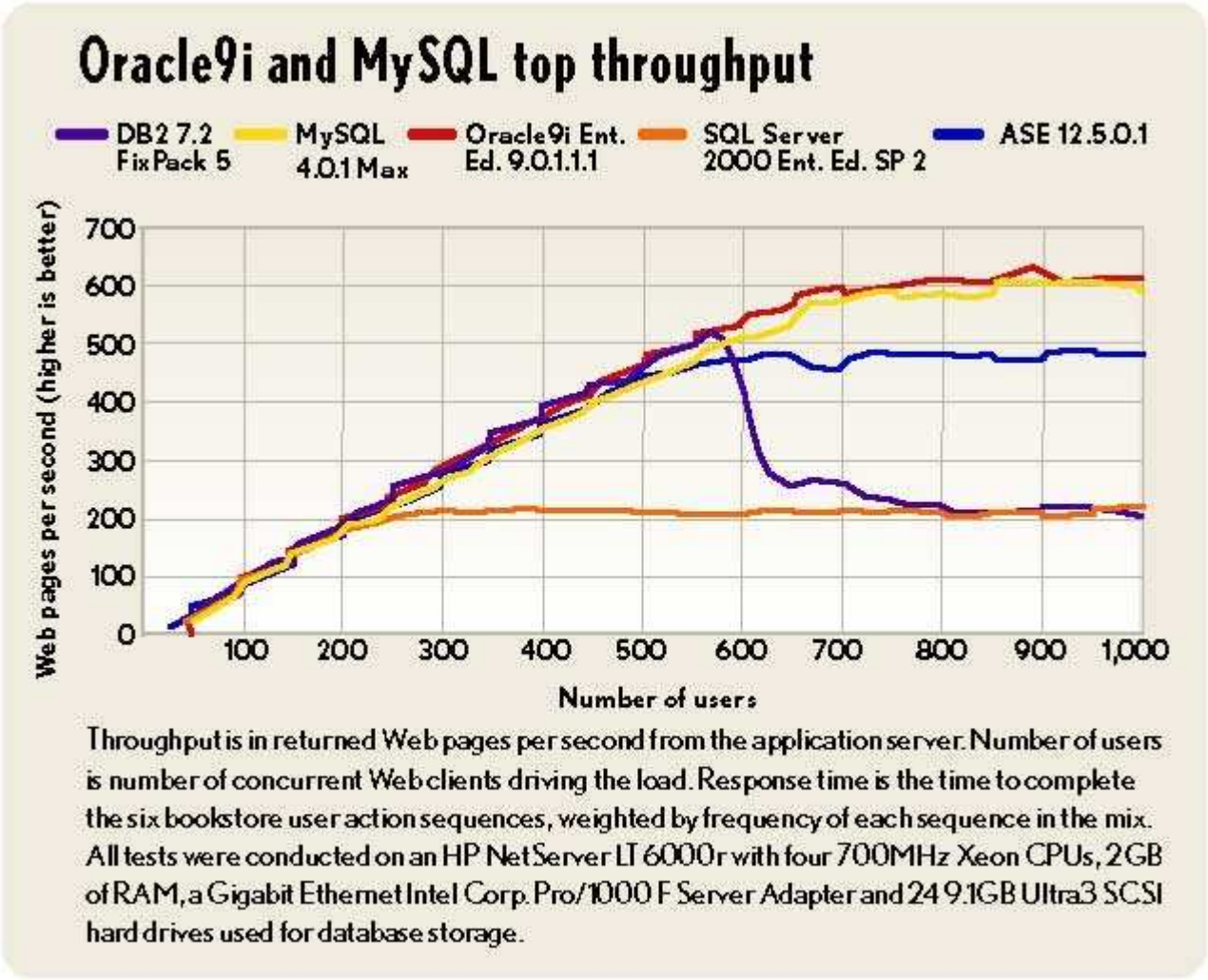
```
<?php
print "Hello from PHP!\n";
$perl = new Perl();
$perl->eval('print "Hello from Perl!\n"');
print "Bye!\n";
?>
```

[Zend Technologies. 2005]

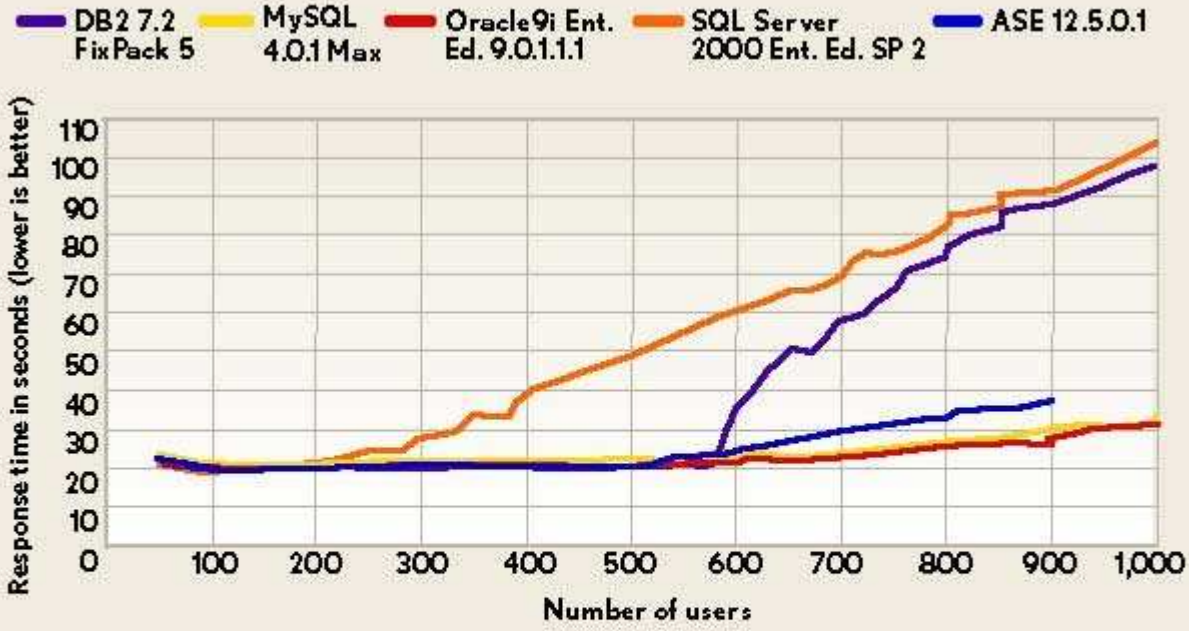
APPENDIX III – Database benchmarking

Results from benchmarking done by e-week.com
[http://www.eweek.com/slideshow/0,2394,pg=0&s=1590&a=23120,00.asp]

MySQL performs well in terms of speed. SQL Server performs poorly except when run on an all Microsoft stack, where its performance exceeds that of the other databases.

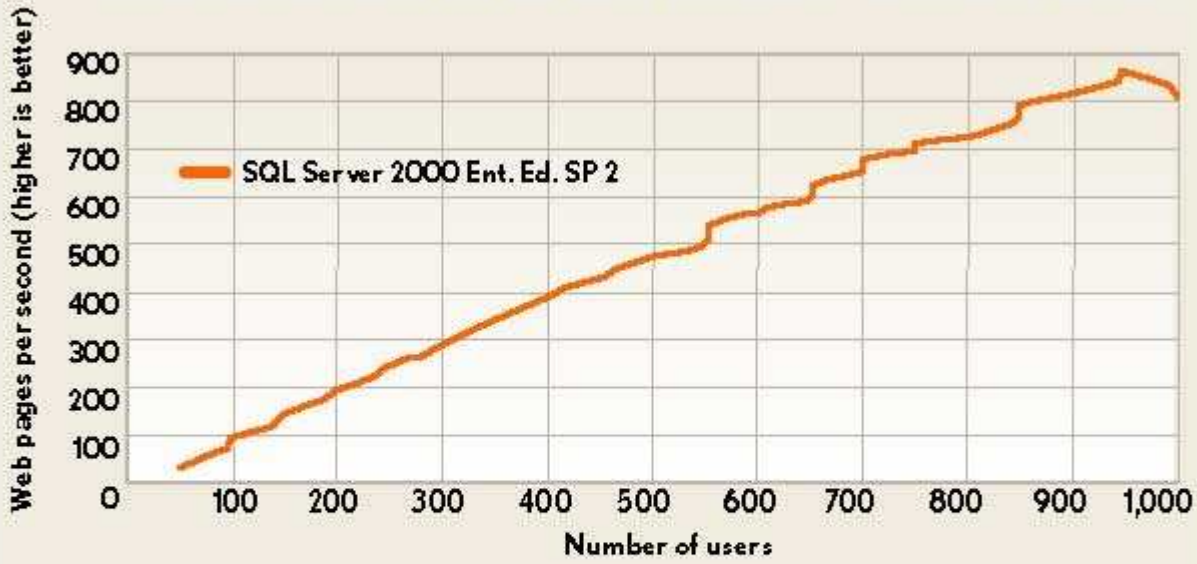


Oracle9i and MySQL offered the fastest response times



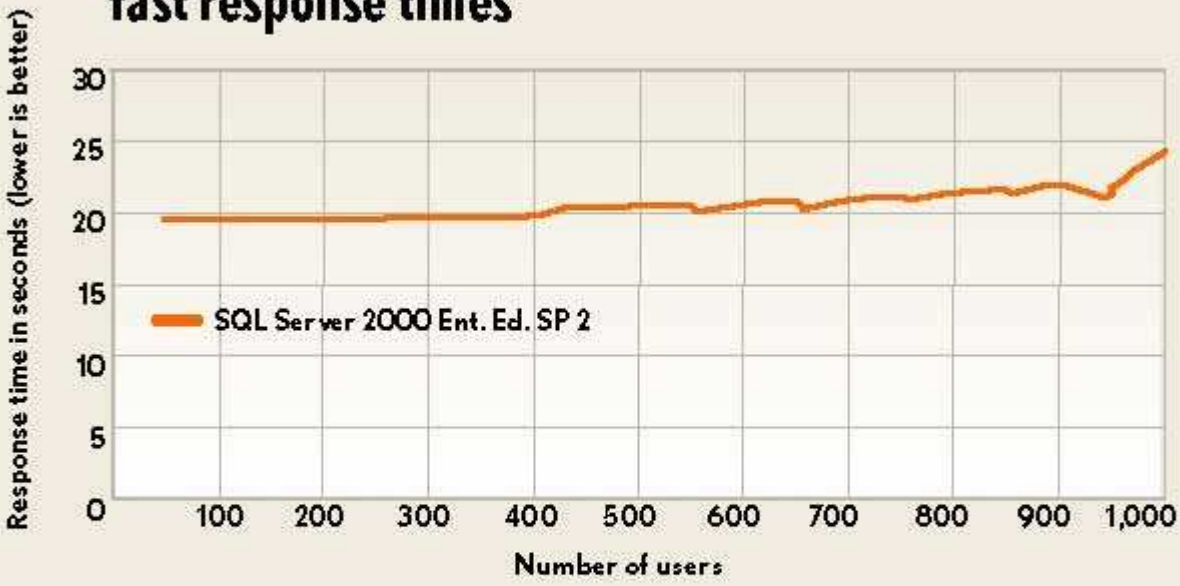
Throughput is in returned Web pages per second from the application server. Number of users is number of concurrent Web clients driving the load. Response time is the time to complete the six bookstore user action sequences, weighted by frequency of each sequence in the mix. All tests were conducted on an HP NetServer LT 6000r with four 700MHz Xeon CPUs, 2 GB of RAM, a Gigabit Ethernet Intel Corp. Pro/1000 F Server Adapter and 24 9.1GB Ultra3 SCSI hard drives used for database storage.

SQL Server 2000 performed very well in ASP .Net-based test on an all-Microsoft software stack



Throughput is in returned Web pages per second from the application server. Number of users is number of concurrent Web clients driving the load. Response time is the time to complete the six bookstore user action sequences, weighted by frequency of each sequence in the mix. All tests were conducted on an HP NetServer LT 6000r with four 700MHz Xeon CPUs, 2 GB of RAM, a Gigabit Ethernet IntelPro/1000 F Server Adapter and 24 9.1GB Ultra3 SCSI hard drives used for database storage.

SQL Server 2000 also clocked consistently fast response times



Throughput is in returned Web pages per second from the application server. Number of users is number of concurrent Web clients driving the load. Response time is the time to complete the six bookstore user action sequences, weighted by frequency of each sequence in the mix. All tests were conducted on an HP NetServer LT 6000r with four 700MHz Xeon CPUs, 2 GB of RAM, a Gigabit Ethernet Intel Pro/1000 F Server Adapter and 24 9.1GB Ultra3 SCSI hard drives used for database storage.