Personalization and Performance

A White Paper

Dan Greening, Pritham Shetty, Glenn Carroll, Stephen Dorato
Andromedia

November 4, 1999

Executive Summary

E-commerce vendors face two important challenges: driving up purchases and maintaining customer loyalty. Only 2.7%* of browsers buy from a site and only 15%* of those buyers return to buy again (Forrester Research, Inc. 1999). To succeed, e-marketers must find ways to keep visitors on the site. They must make the visitors' experience convenient, satisfying and personally relevant. Above all, they must entice Web visitors to come back for more.

Personalization dramatically improves web site revenue and customer loyalty. In particular, personalization has been shown to increase page views per visit, repeat visit rate, and revenue per visit for e-commerce sites.

Adaptive personalization is a popular way to increase revenue and customer loyalty. However, many adaptive personalization systems slow down when faced with high traffic. This paper shows that LikeMinds Personalization Server scales to meet the needs of the most demanding sites, on relatively inexpensive hardware. LikeMinds distributed processing architecture scales nearly linearly with additional machines, making it the most efficient and flexible choice for e-marketers.

Personalization Defined

Personalization comes in many forms. *Customization* allows visitors to change pages on a web site to fit their needs, such as specifying what stocks are interesting and what sports scores to report. This works as long as visitors know what they want.

Rules-based personalization allows a marketer to specify fixed rules to change a site based on visitor behavior. For instance, a marketer might implement a rule that if a visitor buys a digital camera, the site should up-sell additional memory for the camera. BroadVision, a popular e-commerce application server, supports rules-based personalization.

Rules are effective when the marketer understands customers and products well enough to predict each visitor's response. However, rules-based personalization falters when a marketer can't easily predict the response to an offer. Rules-based systems don't learn or adapt to user behavior in realtime. If no rule anticipates an important observed behavior, then a rules-based system provides the default. Because products, customers and business models change, rules-based systems require constant maintenance.

When a site has many content items or products to offer, *adaptive personalization* is more effective than rules-based personalization. This is because adaptive systems—such as the collaborative filtering system used by Andromedia's LikeMinds Personalization Server—can learn from observed behavior and, based on that behavior, select the right content to present or the appropriate product to recommend.

Revisiting the previous example, a marketer may not know ahead of time that buyers of digital cameras are also likely purchasers of high-quality color printers, better video cards, or more disk storage. The site would not recommend these items because no rule told the system to do so.

On the other hand, a site with an adaptive personalization system such as LikeMinds could observe that buyers of digital cameras also purchase these additional items. It would then automatically start recommending these items to digital camera buyers when appropriate—without requiring any marketer intervention. Sites with adaptive personalization capture additional wallet share from online customers. Satisfied by the shopping experience, customers come back for more. For these reasons, adaptive personalization systems have become popular features of e-commerce Web sites.

The Need for Speed

As a Web site becomes more popular, the performance of the personalization system becomes increasingly critical to the site's ability to satisfy visitors. Online shopping succeeds when it makes shopping easier and more satisfying than driving to a brick-and-mortar store. Online visitors put a premium on finding and getting what they want as conveniently and rapidly as possible. If personalized recommendations are slow or inaccurate then the convenience is lost and visitors are very likely to take their business elsewhere—the competition is only one click away.

For these reasons, Andromedia made performance and accuracy paramount in designing LikeMinds Personalization Server. These design imperatives have won many customers for Andromedia. Companies that anticipate high demand and perform competitive tests consistently choose Andromedia's LikeMinds personalization system for its ability to deliver accurate recommendations rapidly on high-traffic sites.

How Much Speed Does a Site Need?

Performance tests show LikeMinds meets the requirements of the highest traffic e-commerce sites.

To help customers choose the right hardware configurations, Andromedia created a performance laboratory to "torture test" LikeMinds on four different configurations: LikeMinds hosted on one dual-processor NT machine, on two dual-processor NT machines, on one dual-processor UltraSparc II machine, and on two dual-processor UltraSparc II machines. All four configurations interacted with an external database machine.

On the most expensive test configuration Andromedia tried—LikeMinds Personalization Server running on a distributed-processing environment composed of two dual-processor UltraSparc IIs with a separate database machine—LikeMinds delivered up to 26,894,400 personalized page views daily, while remaining within an acceptable latency range. This is comparable to traffic seen on large non-personalized portals, such as www.weather.com, impressive performance on hardware costing less than $84,000.

In the least expensive test configuration—LikeMinds running on one dual-processor NT machine—the system delivered up to 8,609,143 personalized pages per day. This performance easily meets the requirements of most e-commerce sites. The test shows online

businesses can rapidly deliver highly accurate personalized recommendation with hardware that costs less than $15,800.

The testing scenarios detailed in this report are very conservative. For example, the measured latency includes network delays between LikeMinds and the web server. Each simulated page generated four recorded events, and requested 100 recommended items. Typical sites experience lower network delays and interact with fewer parameters, obtaining better throughput and latency than reported here.

The test results also demonstrate that LikeMinds multi-threaded, distributed-processing architecture scales nearly linearly with increases in computing power, meaning that the system has extremely high capacity and scalability.

Introduction

Personalization Improves Retention, Drives Loyalty, and Increases Revenue

A salesperson that presents each and every customer with personally relevant products is likely to sell more products, and gain more repeat business. That's why many successful Web sites and call centers now incorporate adaptive personalization technology. Adaptive personalization builds a behavior or interest profile for each Web visitor, and then dynamically changes the online experience for each visitor based on that profile. Personalization increases important e-marketing metrics, such as time on the site, number of pages viewed, rate of return visits, and average spend rate per visit.

Online merchants frequently use personalization systems based on collaborative filtering, such as Andromedia's LikeMinds Personalization Server. LikeMinds records a person's behavior, identifies other people that have similar behaviors (called "mentors"), and uses these mentors to predict content or product suggestions that are of interest to that individual Web visitor. For e-commerce sites, this personally relevant cross-selling increases the revenue gained from Web visitors.

LikeMinds has been proven to have a high ROI in the field. Levi-Strauss & Co. ran a comparison trial that showed that LikeMinds personalization increased the average online customer spend rate by 33%, the average time on the site by 75%, and repeat visitation by 225% over a control group.

Popular Sites Need High Throughput and Scalability

There is an obvious correlation between the accuracy of personalized recommendations and the revenue that an e-commerce site will derive from them. Accuracy counts—the better the recommendations, the higher the revenue and better the repeat visit rate. However, there is usually a tradeoff between performance and accuracy. Few realtime adaptive personalization systems can deliver both high performance and high accuracy, while accommodating the traffic seen on premier e-commerce sites. This should be a major concern, as slow personalization can try a visitor's patience, result in lost revenue, and—worst of all—drive visitors to competitors' sites.

In initial deployments, low traffic may hide throughput limitations in personalization systems. Problems may not occur until a site gains more traffic. In some cases, extremely popular e-commerce sites have dropped their initial personalization system choice because

the software was unable to support high demand. The wrong system can be fast in low traffic and a dog in high traffic.

Every e-commerce site, even those with low traffic today, should plan for future success by estimating traffic levels one, two and three years down the road. Since effective personalization is often pervasive, it can be difficult to switch to another vendor. A slow site could damage an online merchant's brand before they have time to replace or remove the system.

Personalization Metrics

Interaction with a personalization server can be broken into two parts: recording events, and getting predictions. In LikeMinds, application programming interfaces (APIs) provide *addTransaction* function calls to notify the system of events (ratings, purchases, product views, shopping cart inserts/deletes, etc.). These events tell the personalization system that a visitor has done something of interest. The APIs also provide *queryPrediction* function calls to predict what a visitor will do, or recommend personally relevant products or content. A typical personalized page requires one addTransaction function call that passes multiple events, and one queryPrediction function call that returns multiple predictions.

The important performance metrics in personalization are throughput and latency. Throughput is how many function calls can be performed in a second under sustained load. Latency is the average time required by a function call.

To compute the performance a site requires, it is necessary to first determine the *acceptable latency* and *peak personalized traffic*. *Acceptable latency* is the amount of time one can allocate per page to provide personalization. 300mS or less is generally regarded as imperceptible to visitors. In recommended configurations, LikeMinds performs one addTransaction call plus one queryPrediction call in less than 150mS total at peak loads.

Peak personalized traffic is the number of pages per second during peak times. To calculate this, divide total pages per day by 86,400 (the number of seconds per day), then multiply by the peak traffic per hour and divide by the average traffic per hour.

For example, suppose a site gets 1,000,000 page views daily, with the ratio of peak-to-average hourly traffic at five-to-one. Then peak personalized traffic is about 58 pages per second. If every page is personalized with two function calls per page, the site requires 116 calls per second—a demand easily satisfied by a modest Windows/NT implementation of LikeMinds.

Scalability is the ability of a system to gracefully accommodate more traffic with additional computers. To achieve high scalability, the system must be designed from the ground-up to work in parallel. Many personalization systems in use today aren't designed for scalability.

The rest of this paper discusses LikeMinds' highly scalable architecture, and describes the performance testing results for single and distributed configurations of LikeMinds on both Windows/NT and Solaris. The tests use conservative assumptions, so e-marketers can confidently use these results to determine the hardware and software configuration needed to personalize their Web sites.

LikeMinds Architecture

Andromedia's LikeMinds Personalization Server was designed from the ground-up for low latency, high throughput, high scalability and high accuracy. These performance goals are essential for popular sites. Online businesses that directly compared LikeMinds performance and accuracy with that of competitive products have selected LikeMinds.

This paper also discusses how LikeMinds Personalization Server achieves its high performance. The server was tested on different hardware and software configurations in the Andromedia Performance Lab. Results are shown for Windows/NT and Solaris, on single- and distributed processing configurations, running Oracle and Microsoft SQL Server database back-end software.

LikeMinds predictive modeling is based on an innovative and patented form of collaborative filtering. LikeMinds first tracks user behavior, finds mentors with similar behaviors, and then uses the behaviors and preferences of those mentors to recommend new items.

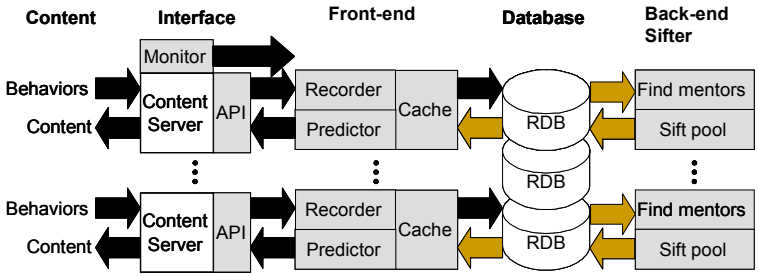

Figure 1. Distributed Processing Architecture

Figure 1, above, shows the architecture of the LikeMinds system. Behaviors—such as clicking on a link, purchasing a product, or specifying a preference for something—are interactions between the visitor and the content server (which can be a Web server, a call center application, etc.).

The API library is a software interface installed on the same machine as the content server. It simply translates method calls to a network protocol. The API has several language realizations: Java, JavaScript, Visual Basic, C++, C. Multiple API libraries, installed on different machines, can communicate with a single front-end.

The front-end has two functions: record behaviors and predict behaviors (or recommend things). It can be hosted on a different machine than the content server.

When a visitor does something relevant, which is reported through the API, the front-end records that behavior in a lazy write-through cache. This type of cache makes writing into databases extremely fast. Several behaviors can be recorded in a single call.

A relational database stores data about visitors, items to be recommended, mentors and relationships. An ODBC interface accommodates any relational database server. LikeMinds also has an Oracle native interface for higher performance needs.

The sifter process constantly runs in the background to determine who's like whom. It matches visitors to other visitors or archetypes in the database using up-to-the minute behavioral data provided by the front-end process. Each visitor gets a set of mentors. Each mentor's contribution to the recommendation is weighted according to the mentor's similarity to the target.

When a Web application requests a recommendation through the API, the front-end fetches mentors and their behaviors for the visitor (often from the cache), constructs a prediction vector for the subset of items recommended, sorts the results by value and confidence, and returns the recommendation. The API allows multiple recommendations to be returned in a single call, including information about the items, predicted values, and confidence.

Performance Features

Three features contribute to LikeMinds performance: An application-specific cache, pervasive multithreading and a ground-up design supporting distributed processing.

Caching reduces latency by keeping frequently used items in main memory. Effective cache designs strike a balance between performance and flexibility. Requiring everything to fit in main memory can reduce latency, but such systems will break when memory is exhausted, or when the number of items exceeds an upper bound.

LikeMinds puts complete flexibility first, and then offers the most efficient caching scheme within that requirement. For example, LikeMinds can run in a small amount of memory regardless of the number of items or visitors in the database. However, it runs faster when more memory is allocated for caching.

The benefits of the LikeMinds cache can extend to the Web application itself. The cache normally records visitors, items, mentors and predictions. LikeMinds can also be configured so that item-specific data—such as name, SKU, price, etc.—are cached with the item. The LikeMinds APIs let Web applications request this data. If a visitor receives a recommendation from LikeMinds, the data associated with that recommendation are in the cache. Thus, the cache not only makes LikeMinds recommendations faster, it also accelerates the display of those recommended items. No other personalization solution offers this capability.

Multithreading increases throughput, allowing LikeMinds to exploit a single CPU to the greatest extent possible. While one operation blocks waiting for a database fetch, another operation can compute a prediction, accept a new request, etc. Multithreading allows LikeMinds to exploit symmetric multiprocessors, such as Sun Microsystems Ultra IIs and Enterprise Servers or multiprocessor Pentium servers, increasing the throughput with each additional processor.

LikeMinds also supports *distributed processing* (separate machines connected over a high-speed network). Distributed processing support lets you increase throughput incrementally by adding inexpensive hardware, gaining a nearly linear increase in throughput for each additional machine.

LikeMinds software was designed from the ground-up to support both tightly coupled multiprocessing (symmetric multiprocessors) and loosely coupled multiprocessing (distributed processors) efficiently. No other personalization solution offers this flexibility and scalability.

LikeMinds Performance Tests

Test System

Andromedia conducted these performance tests using the database from a production personalization site—Movie Critic (www.moviecritic.com), the popular movie recommendation site that also serves as a demonstration of LikeMinds personalization capabilities. Industry analyst Peppers & Rogers Group recently rated Movie Critic as one of the world's best one-to-one Web sites.

When the tests were conducted, the Movie Critic database contained 166,518 registered visitors and a total of 4,568 different movie listings. Visitors averaged 46 movie ratings. Mentors in the *mentor pool* (the set of visitor profiles that may be used as mentors) had an average of 330 ratings. The system was configured for a maximum of 4,000 mentors in the mentor pool.

LikeMinds performance is independent of the number of registered users and the total number of items. Performance is roughly linearly dependent on the number of mentors in the mentor pool, the average number of recorded behaviors or ratings per mentor, and the function calls per second.

Andromedia performed simulated visitor interactions using two external "hitter machines" to simulate traffic—visitors interacting with the site and generating events as well as requesting recommendations. Each hitter machine was an UltraSparc II containing dual 400MHz CPUs. The hitters made random API calls to add visitor transactions (to enhance the visitors' profiles) and query recommendations (to create a personalized experience). It was necessary to use two machines, because one machine could not saturate the LikeMinds Personalization Server.

The test scenario simulated the arrival, site interaction, and departure of Web visitors with realistic, random behavior. Each "visitor" issued an API call every four seconds, on average, following a Poisson distribution. Calls were either *addTransactions* or *queryPredictions*, chosen at random in a 2:3 ratio. Each *addTransactions* call inserted four transactions on items selected according to the global distribution of ratings in the initial database. Each *queryPredictions* call requested the top 100 predictions (e.g. "Best Bets"). These parameters were chosen to be significantly more challenging than those of high-traffic Web sites. Most sites insert a single transaction per call and request only the top ten predictions.

The hitters affected traffic by varying the number of active visitors. The simulation started by drawing a random subset of visitors from the initial database. To simulate arrivals and departures, 5% of the visitors left the system every ten seconds, yielding an average session time of 105 seconds. The simulation replaced visitors one-for-one, 98% of the time with a registered visitor already in the database and 2% of the time with a new visitor, registering for the first time. This reflects the growth of registered visitors observed on the Movie Critic site.

LikeMinds 3.1 supports temporary visitors, but the tests presented here do not include them. Temporary visitors are represented solely in the cache so as to avoid database transactions that slow the system down. However, temporary visitor data disappears when the session ends. Use of temporary visitors will allow higher traffic at lower latency than indicated by this performance test. Again, the results shown here are conservative.

Single-Host NT Configuration

The most common LikeMinds configuration in practice is the LikeMinds front-end process and sifter process sharing a dual-processor Windows/NT machine, with SQL Server 7.0 running on a separate machine. The sifter process runs at extremely low priority, and gets no CPU time when the front-end process is saturated.

Though this is a reasonable configuration for a customer, it is not a reasonable configuration for a performance test. Under heavy load, the front-end would starve the sifter of CPU time, and the sifter would not compete with the front-end for the database in a realistic way. At a constantly heavy load, sifting would not be performed and accuracy would suffer, but the front-end would be faster.

Instead, Andromedia created a configuration that places the sifter on a separate machine. In this test, the main contribution of this additional host was to slow the system down! The approach of placing the sifter on a separate machine was used in each of the four tests described in this paper.

Component	OS	System Type	CPUs	Memory
Front-end	Windows NT	Pentium III 500	2	1 GB
Sifter	Windows NT	Pentium III 500	2	1 GB
SQL Server 7.0	Windows NT	Pentium III 500	2	1 GB

Table 1. Single-Host NT Configuration

The test configuration is shown in Table 1. Other than the presence of dual processors and large amounts of memory, these machines hosting LikeMinds Personalization Server are run-of-the-mill workstations, running Windows/NT 4.0 SP5. None had RAID controllers. The SQL Server machine had SCSI drives, with software striping.

Users	Throughput (calls/sec)			Front-end Latency (milliseconds)			API Latency (milliseconds)			CPU
	QP	AT	Total	QP	AT	Average	QP	AT	Average	
400	60	40	100	36.2	8.7	25.2	47.8	11.9	33.5	30%
600	90	60	150	47.9	8.0	32.0	66.5	9.9	43.8	45%
800	118	79	197	112.0	4.0	68.8	113.9	6.6	71.0	78%
1000	133	88	221	171.5	2.3	103.8	185.3	4.7	113.0	84%
1200	133	89	222	212.9	2.4	128.7	221.6	4.7	134.9	87%

Table 2. Single-Host NT Performance

Table 2 shows the single-host NT performance test results.

The left-hand side shows the number of users driving the test.

The first set of three columns shows the system throughput in calls per second. QP indicates queryPredications calls, AT indications addTransactions calls, and total is the sum of both. One can see that the system becomes saturated at around 221 calls per second, processing an average of one call per 4.5mS.

The second set of three columns shows the latency of calls measured at the front-end process. These figures do not include the round-trip time from the API library through the network to the front-end. This is valuable because it establishes a lower bound on the latency, assuming better network hardware between API and front-end.

The third set of three columns shows the latency of calls measured before the API is called and after it returns. This is the effective latency seen by the Web application in our switched 100BaseT Ethernet environment.

The last column shows the CPU utilization.

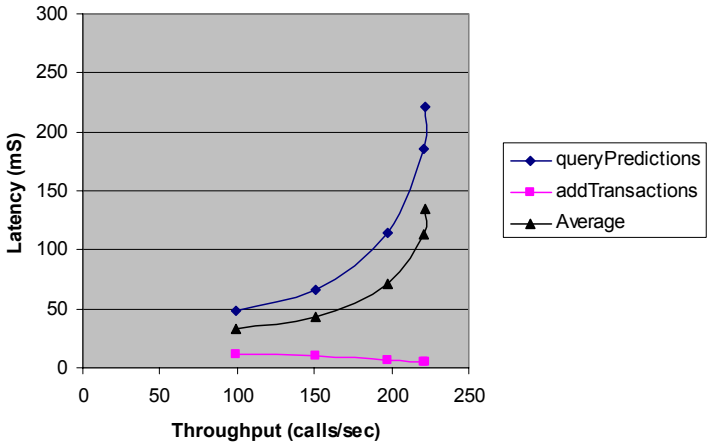

Figure 2. Single-Host NT Latency vs. Throughput

Figure 2 shows API latency plotted against throughput.

As one might expect, heavier demand (throughput) increased latency in the queryPredictions call. However, heavier demand *reduced* latency in addTransactions, an odd result. This is likely due to Windows/NT multithreading. The addTransactions call places the transaction on a producer-consumer queue. If the queue contains nothing when a transaction is added (such as when demand is low), this may cause a thread-switch to the consumer side of the queue, increasing latency. However, when something is on the queue, adding another transaction has no effect. Therefore, when demand is higher, addTransactions latency declines.

Using linear interpolation, we can conclude that a dual Pentium III 500 MHz system is capable of handling at least 199 API calls per second, with an average latency of 75mS per call.

This configuration is adequate for sites serving up to 8,609,143 personalized page views daily. For sites seeing a peak-to-average demand ratio of five-to-one, the configuration can provide for 1,721,829 personalized page views daily.

Dual-Host NT Configuration

The dual-host NT configuration shows the affect of adding another front-end host to the previous configuration.

Component	OS	System Type	CPUs	Memory
Front-end A	Windows NT	Pentium III 500	2	1 GB
Front-end B	Windows NT	Pentium III 450	2	1 GB
Sifter	Windows NT	Pentium III 500	2	1 GB
SQL Server 7.0	Windows NT	Pentium III 500	2	1 GB

Table 3. Dual-Host NT Configuration

Table 3 shows the hardware configuration in this test. A matched Windows/NT dual Pentium III 500MHz system was added to host another front-end process.

Users	Throughput (calls/sec)			Front-end Latency (milliseconds)			API Latency (milliseconds)			CPU
	QP	AT	Total	QP	AT	Average	QP	AT	Average	
400	60	40	99	14.4	2.9	9.8	21.6	5.5	15.1	18%
800	119	79	198	14.7	2.0	9.6	24.1	5.4	16.6	25%
1200	179	119	299	41.4	8.2	28.1	49.8	12.2	34.7	55%
1600	235	157	392	116.4	3.9	71.4	125.1	6.5	77.7	80%
2000	259	172	431	172.2	2.4	104.3	201.5	4.2	122.6	85%
2400	258	172	430	218.8	2.5	132.3	248.1	5.6	151.1	87%

Table 4. Dual-Host NT Performance

Table 4 shows the dual-host NT performance test results, in the same format as the previous test.

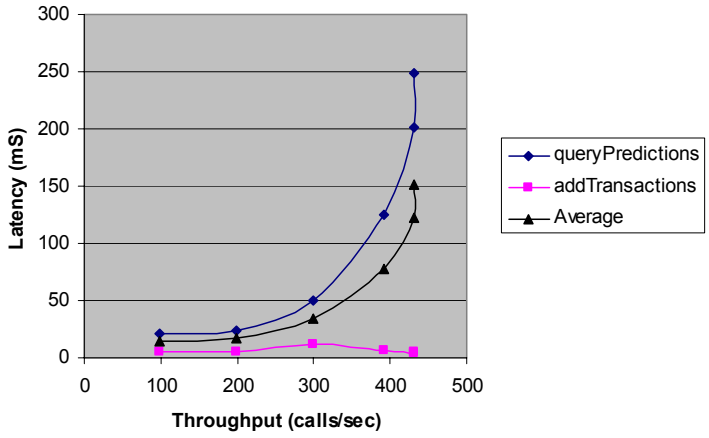

Figure 3. Dual-Host NT Latency vs. Throughput

Figure 3 plots the API latency against throughput.

Using linear interpolation with the results in Table 1, we see that at the same average API latency of 75mS in the single-host case, the dual-host configuration can process about 386 calls per second, versus 199 in the single host case. This is approximately 90% of linear scaling.

This configuration is adequate for sites serving up to 16,682,132 personalized page views daily. For sites seeing a peak-to-average demand ratio of five-to-one, the configuration can provide for 3,336,426 personalized page views daily.

This performance test shows that throughput capacity on Windows/NT scales nearly linearly with additional front-end processors.

Single-Host Sparc

Another common LikeMinds configuration in practice is the LikeMinds front-end process and sifter process sharing a dual-processor UltraSparc machine, with Oracle 8 running on a separate machine. Again, the sifter process runs at extremely low priority, and gets no CPU time when the front-end process is saturated.

So, as with the Windows/NT configuration, this test configuration included a sifter on a separate machine to slow the system down.

Component	OS	System Type	# of CPUs	Memory
Front-end	Solaris 2.6	400 MHz UltraSparc-II	2	2 GB
Sifter	Solaris 2.6	400 MHz UltraSparc-II	1	0.5 GB
Oracle 8.0.2	Solaris 2.6	300 MHz UltraSparc-II	4	4 GB

Table 5. Single-Host Sparc Configuration

The test configuration is shown in Table 1. Other than large amounts of memory on the front-end, these machines are typical Sun workstations, running Solaris. None had RAID controllers.

The machine hosting the Oracle database is less powerful than database machines found at typical customer sites. The recommended Oracle configuration includes 9 to 18 striped disk drives with more memory. The system had two SCSI disks with no striping.

The sifter was hosted on a relatively small machine. It was clear that the sifter could easily keep up with traffic even when the front-end processes were saturated.

Users	Throughput (calls/sec)			Front-end Latency (milliseconds)			API Latency (milliseconds)			CPU
	QP	AT	Total	QP	AT	Average	QP	AT	Average	
400	60	40	100	10.0	0.6	6.2	21.1	5.6	14.9	14
800	120	80	200	18.4	0.6	11.3	30.2	8.0	21.3	31
1200	180	120	300	33.4	0.6	20.3	45.7	10.4	31.6	43
1600	220	140	360	39.6	0.7	24.5	103.9	21.7	71.9	78
1800	240	160	400	39.6	0.7	24.0	104.5	30.5	74.9	80
2000	246	159	405	40.6	0.7	24.9	107.5	41.8	81.7	89

Table 6. Single-Host Sparc Performance

Table 6 shows the performance results for the single-host Sparc configuration. This shows that Sparcs can deliver about twice the throughput as Windows/NT—at the same latency and the same processor speed.

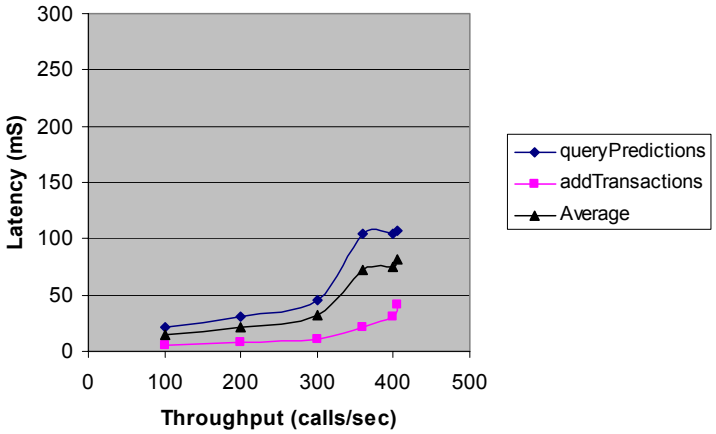

Figure 4. Single-Host Sparc Latency vs. Throughput

Figure 4 plots API latency against throughput.

This configuration is adequate for sites serving up to 17,283,176 personalized page views daily. For sites seeing a peak-to-average demand ratio of five-to-one, the configuration can provide for 3,456,635 personalized page views daily.

Dual-Host Sparc

The two-host Sparc configuration shows the effect of adding another front-end host to the previous configuration.

Component	OS	System Type	# of CPUs	Memory
Front-end A	Solaris 2.6	400 MHz UltraSparc-II	2	2 GB
Front-end B	Solaris 2.6	400 MHz UltraSparc-II	2	2 GB
Sifter	Solaris 2.6	400 MHz UltraSparc-II	1	0.5 GB
Oracle 8.0.2	Solaris 2.6	300 MHz UltraSparc-II	4	4 GB

Table 7. Dual-Host Sparc Configuration

Table 7 shows the hardware configuration in this test. A matched UltraSparc II 400MHz system was added to host another front-end process.

Users	Throughput (calls/sec)			Front-end Latency (milliseconds)			API Latency (milliseconds)			CPU
	QP	AT	Total	QP	AT	Average	QP	AT	Average	
800	119	80	199	23.2	0.3	14.0	28.5	7.5	20.1	11%
1600	238	159	397	49.0	0.6	29.7	76.0	10.9	49.9	25%
2400	351	234	585	48.9	0.5	29.6	116.7	21.9	73.8	41%
3200	453	302	754	38.0	0.7	23.1	112.4	29.5	79.2	59%
4000	488	325	814	49.2	0.7	29.8	141.2	45.1	102.8	66%

Table 8. Dual-Host Sparc Performance

Table 8 shows the dual-host Sparc performance test results, in the same format as the previous test.

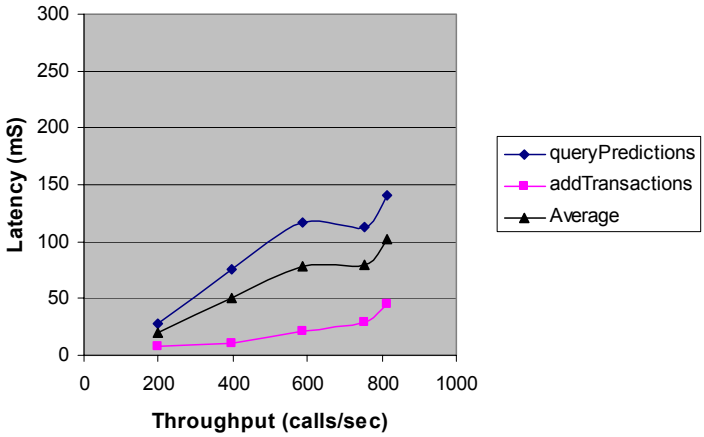

Figure 5. Dual-Host Sparc Throughput vs. Latency

Figure 5 plots API latency against throughput.

Using linear interpolation, we find this configuration provides a maximum latency of 75mS at about 623 API calls per second, versus 386 in the single-host case—about 61% of linear scaling. This lower level of scaling on Sparcs vs. Windows/NT is likely due to increased saturation of the network. Note the high latency cost attributable to the network, shown in the difference between the front-end latency columns and the API latency columns of Table 8. Nevertheless, 61% of linear scalability is quite respectable.

Multiple LAN adapters on the database server and a partitioned LAN would likely bring the scaling closer to linear. Reducing the number of predictions in queryPredictions would also bring scaling closer to linear.

This configuration is adequate for sites serving up to 26,894,400 personalized page views daily. For sites seeing a peak-to-average demand ratio of five-to-one, the configuration can provide for 5,378,880 personalized page views daily.

Conclusion

To increase revenues and retain customers, e-merchants are increasingly turning to adaptive personalization technology. However, e-businesses cannot sacrifice speed, because visitors have little patience for slow Web sites.

This paper shows that LikeMinds Personalization Server performs at high speed under very heavy traffic, making it suitable for the most demanding web sites.

System	Hosts	HW Cost	Calls/sec	Calls/day	Max pages/day
Pentium III 500, NT	1	$15,800	199	17,218,286	8,609,143
Pentium III 500, NT	2	$23,700	386	33,364,264	16,682,132
UltraSparc II 400, Solaris	1	$56,000	400	34,566,353	17,283,176
UltraSparc II 400, Solaris	2	$84,000	623	53,788,800	26,894,400

Table 9. Configuration cost and throughput

Table 9 shows the four test configurations examined in this paper, and a summary of the performance results. The HW Cost column indicates the cost of hardware to host LikeMinds and the database server. The Max pages/day column indicates the maximum number of pages per day, assuming two calls per page and a minimum response time of 75mS.

The least expensive test configuration, costing $15,800, delivered up to 8,609,143 pages per day at acceptable delay. This configuration is suitable for sites experiencing moderately high traffic.

The most expensive test configuration, costing $84,000, delivered up to 26,894,400 pages per day at acceptable delay. This configuration meets the requirements of today's highest traffic Web sites.

Your Web site's revenue and customer loyalty depend on the speed and accuracy of your personalization system. LikeMinds Personalization Server's accurate personalization easily scales to meet the most demanding needs.