Developing a High Traffic, Read-Only Web Site

John Nauman
Infoseek Corporation
1399 Moffett Park Drive
Sunnyvale, CA, 94089
1-408-543-6773
jnauman@infoseek.com

Ray Suorsa
Infoseek Corporation
1399 Moffett Park Drive
Sunnyvale, CA, 94089
1-408-543-6773
res@infoseek.com

ABSTRACT

In this paper, we describe some of the considerations for designing highly trafficked web sites with read-only or read mostly characteristics.

Keywords

Highly trafficked web sites, web site caching, stable sockets

1. INTRODUCTION – THE PROBLEM

There are several different areas that need to be considered in providing a viable commercial web site/service. These include the capital investment (hardware and connectivity costs necessary to support the service levels desired), as well as successful marketing of the web site to differentiating it and gain share of mind. If these two items are successfully accomplished and traffic begins to flow to the site, the technology used to deliver the service quickly becomes crucial to its success.

There are three major technology challenges for a highly trafficked web site: availability, speed and scalability. This ordering is intentional, and important. Once the hardware and connectivity for the web site are in place and once the marketing efforts have been successful in getting folks to try it, the site must be up and available for people to be able to use. However a site that takes too long to respond, whether the responses are search results or a catalog of merchandise, will have people abandon it in favor of sites that can respond to the requests faster. Finally, in the current web environment of hyper-growth, a site must be scalable to allow it to grow as the web expands and as users figure out how to do more with it. In this paper we discuss how each of these factors is dealt with at Infoseek (www.infoseek.com), one of the premier search engines on the web.

2. SERVICE OVERVIEW

Before going into detail on how we address the three areas above, a brief system overview is in order. While we looked at alternatives when the original Infoseek site was developed, no off-the-shelf web-servers were available that met our requirements. As a consequence we designed essentially all of the website service components ourselves. Each of the layers outlined below was architected and developed to have the necessary availability, performance and scalability.

The initial layer of the service is the Proxy layer. When a user comes to www.infoseek.com, a user connection to one of the proxies is established via DNS round-robin (to distribute the traffic). Each of these proxies is responsible, once assigned, for servicing the user requests.

Behind the proxies are Page-Builders. The page-builders are responsible for constructing the page containing the user's results, as well as related information and an advertisement, in HTML and JavaScript. After construction the page-builders return the page, via the proxy, to the user. Since Infoseek is a search engine and since most requests cannot be anticipated we dynamically construct virtually all the pages we deliver from the service. Caching of specific query responses as well as titles information about the documents indexed is done in the page-builder to improve performance.

In the event that the response to the query is not already in its cache, the page-builder makes a request for information (results) from a set of Cache Machines. These cache machines serve as an intermediate stage for query results and titles information (which are cached separately) that might be used in response to multiple queries. Caching query results and titles information separately provides better performance than a common cache due to the different access statistics of the two types of data. Non-query content such as related news or business information is stored in databases at this level as well.

If the cache machine is unable to provide the results it makes a request of the Search Engine for the information. The search engine is essentially the heart of the system. The search engine is built around patented distributed-search technology which is significantly more advanced that that employed by other search engines. It hosts Infoseek's web index and does the searching across the distributed indexes comprising the WWW collection. While the search engine also maintains a cache to avoid disk access, this is the place the index is actually searched if that is necessary.

3. AVAILABILITY

Clearly the most important aspect of a high traffic web site is availability. If the site is not there when the user wants to visit, then the likelihood of a repeat visit will drop significantly. At Infoseek we take an n+1 redundant/failover approach to availability. The four classes of systems outlined above provide the search service and each of these systems is replicated both in hardware and software. In addition, we have multiple T3 connections to the Internet through multiple ISPs. That's the easy

part. The challenge is to actually have the service reconfigure itself in the face of a failure so that availability isn't compromised. In addition to architecting the service components to reconfigure, we tuned the default TCP parameters of the OS (Solaris) to enable our approach.

To support failure detection each of the systems in the service has two sets of persistent (stable socket) connections to other systems. One of these is used for high volume transactions between systems while the other monitors to ensure the other systems are alive and functional. The number of transaction connections is varied depending on the concurrency demand between the two systems. In the event of a failure of one system the connected system notices the failure through the monitor port, close their connections to the failed system and open connections to the least loaded system providing the same service. The failed system is removed from the pool of systems providing the service. What this means is that in the event of a failure the service reconfigures itself dynamically to circumvent the failure. Since the detection and circumvention of this failure can take a few seconds, we typically experience an increase in service time in the area of the service impacted for a brief period. If a failure occurs, scripts in monitoring machines detect the failure and cause the operations organization to be alerted to the possible need for human intervention. If the failure is a transient, the failed system will re-initialize itself and return to service at which point it rejoins the servers in its class.

While all this seems relatively straightforward, the integration of the monitoring, failure detection and re-initialization has proven to be one of the more delicate pieces of the system.

4. PERFORMANCE

Once the availability of the site/service is dealt with, and the user can expect to find the site when they look for it, the issue of performance of the site rises in importance. There are a few major areas that we have found which can dramatically impact performance of our service. Since we are largely serving results from static collections we are not usually faced with the serialization and commit logic inherent in systems that support significant levels of update transactions. This means most of our I/O access is to read data (indices, titles, summaries, ...) in order to satisfy a query. This, in turn, implies that avoiding these I/Os is the way to have the most profound impact on system performance. In the limit, doing no I/O would be preferable. Unfortunately the size of the web (tens of millions of web pages indexed) and the dynamic nature of its content make it infeasible to cache all the data for most of our collections. Caching strategies are critical to our ability to dynamically construct results pages quickly. We cache in the page-builders and the search engine as well as in the cache machines to try to reduce the I/O required to satisfy a request.

While reducing I/O as much as possible is the primary means to improve performance, there are other areas that merit attention as well. We have found that the persistent (stable socket) connection protocol has generated significant savings in system overhead for setup and teardown of intra-service connections. In addition storage allocation, de-allocation, and garbage-collection are areas where we've been able to significantly streamline system performance/throughput in multi-threaded, multi-processor mode.

The server architecture is distributed and multi-threaded which yields significant latitude in handling transaction latency. We have found this architecture to be the most effective in dealing with the transaction queues generated on the Internet.

5. SCALABILITY

With the performance problem under control the next area of focus is scalability. The web has been experiencing growth rates in the 5-10% per month range for the past two years. This rate shows no sign of abating in the near future. That means any site that is providing a primary web service must be prepared to grow at these rates. This translates into the requirement for a very open-ended architecture. Two years ago the Infoseek service was capable of serving ~5M simple page views per day against an index of a few million pages. Since then the service has evolved to be capable of supporting tens of millions of complex, content-rich page views per day against an index of tens of millions of web pages. This meant evolving our system at all levels. We enhanced our page-building architecture to make it increasingly robust while, at the same time, increasing its efficiency. The search engine of two years ago was capable of supporting about 3 million indexed pages. We replaced it with an engine that exhibits exceptional relevance and recall as well as high performance, virtually unlimited index size, support for search across distributed indices and the ability to deal well with natural language.

On the overall service front, the redundancy/failover architecture described above also directly contributes to scalability. As a result of the architectural decisions around providing redundancy it is a simple matter to add a new system at any of the four system levels and have it assume a part of the service load from its peers. Resources are added only where needed, permitting fine-grain scalability. This technique has been used successfully over the past two years as we have increased our traffic and our user base at approximately the same rate that the Internet is growing. While adding additional systems is one solution to the explosive growth we have experienced, it is an expensive one. In an effort to keep costs under control we have also made several major modifications to the service to make it less resource consumptive as it grows. This is accomplished by working within one of the levels of the service to improve its internal architecture and processing while continuing to maintain its interfaces to reduce the impact of the enhancements on other parts of the service.

6. CONCLUSION

Over the past two years the Infoseek service has grown by almost a factor of 10 in terms of results served to our users. This growth has been a direct result of successfully addressing the challenges of availability, performance and scalability of the service.

(I-4)