International Journal of Research
Available at
https://www.internationaljournalofresearch.org/
ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

Forecasting of Internet Performance

Ms. BODAPATI KAVERI **Mr. G.DILIP KUMAR**

B. Tech Student, Department of CSE,
Sri Mittapalli College of Engineering, Tummalapalem, NH-16, Guntur, Andhra Pradesh, India

Assistant professor, Department of CSE,
Sri Mittapalli College of Engineering, Tummalapalem, NH-16, Guntur, Andhra Pradesh, India

Abstract:-

In the application, we use the BIGRQUERY package to query and download results. The BIGRQUERY package provides an interface to Google BIGRQUERY which hosts NDT results along with several other datasets. However, R users will need to first set-up a BIGRQUERY account.

Once done, SQL-like queries can be run from within R. The results are saved as a data frame on which further analysis can be performed. Aside from the convenience of working within the R environment, the BIGRQUERY package has another advantage: the only limitation to the size of the query results that can be saved for further exploration is the amount of available RAM.

Keywords*-Big social data, Social set analysis, Social business, Visual analytics, geo-spatial, GIS, Taxi, Green cabs, Uber..*

I INTRODUCTION

Internet-based (or network-centric) applications have experienced incredible growth in recent years and all indications are that such applications will continue to grow in number and in importance. How such applications should be structured and how they should interact with operating systems is the subject of much activity in the research community, where it is commonly believed that existing interfaces are ill-suited to supporting such applications [4],[18], [14].

Current approaches to building Internet server software suffer from the problem that if the demand from client applications exceeds the server's ability to handle the demand, the performance of the server and hence the performance seen by the clients degrades dramatically. That is, existing servers are not wellconditioned to load. In fact, in such systems the throughput of the server approaches zero as the number of simultaneous requests to that server continues to grow. This is reflected in unpredictable and extremely long wait times, or a complete lack of response for many of the users of such systems. It is precisely during these

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

periods of high demand when being able to service customers may be most important to those who are relying on the server.

Examples of such periods occur during sharp changes in the stock market, breaking news events, and the Christmas shopping season.

Unfortunately, it is not practical or cost effective to provision a system in order to handle peak demands because the peak demand on web servers can be several to hundreds of times higher than the average demand [1] [17].

The goal of this work is to examine the impact of various design considerations on a simple select-based web-server to determine how such an application can be better conditioned to load.

II RELATED WORK

Current approaches to implementing high-performance web servers require special techniques for dealing with high levels of concurrency. This point is illustrated by first considering the logical steps taken by a webserver to handle a single client request, as shown in Figure 1.

Note that nearly all Internet-based servers and services follow similar steps.

1. Wait for and accept an incoming network connection.
2. Read the incoming request from the network.
3. Parse the request.
4. For static requests, check the cache and possibly open and read the file.
5. For dynamic requests, compute the result.
6. Send the reply to the requesting client.
7. Close the network connection

Several of these steps can block because they require interaction with a remote host, the network, a database or some other subsystem, or potentially a disk. Consequently, in order to provide high levels of performance the server must be able to simultaneously service partially completed connections and to quickly and easily multiplex those connections that are ready and able to be serviced (i.e., those for which the application would not have to block and wait). This may result in the need to be able to handle several thousands or tens of thousands of simultaneous connections [4].

Initial attempts to implement web servers handled concurrency issues by creating a separate thread of control for each new connection and relying on the operating system to automatically block and unblock threads appropriately. Unfortunately, threads consume significant amounts of resources and server architects found that it was necessary to restrict the number of executing threads [8] [4].

More recent approaches to high-performance server design treat each connection as a finite state machine (FSM) with transitions between states being triggered by the event being processed. Several connections are managed simultaneously by multiplexing between different connections (FSMs) with the server working on the connection on which forward progress can be made without invoking an operating system call that would block to wait

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

for a result. This is accomplished by tracking the file and socket descriptors of interest and periodically querying the operating system for information about the state of these descriptors (using a system call like select or poll). The results of these calls indicate to the application which operations can be performed on which descriptors without causing the application to block. Obtaining this information is the key component in providing the ability to multiplex between outstanding connections.

Significant research has been conducted into improving web server performance by improving both operating system mechanisms and interfaces for obtaining information about the state of socket and file descriptors from the operating system [3] [12] [2] [4] [13] [14] [6]. These studies have been motivated by the belief that under high loads with a large number of concurrent connections, the overhead incurred by select (or similar calls) is prohibitive to implementing high-performance Internet servers. As a result they have mainly developed improvements to select, poll and sigwaitinfo by reducing the amount of data that needs to be copied between user space and kernel space or by reducing the amount of work required by the kernel to perform such operations (e.g., by only delivering one signal per descriptor in the case of sigwaitinfo).

Interestingly, recent work by Chandra and Mosberger [6], in addition to introducing operating system modifications designed to improve application performance, demonstrates that a rather simple modification to a select-based web-server (with a stock operating system) provides better performance than their attempts and the attempts of others to improving performance by modifying the operating system. They refer to this server as a "multi-accept" server because upon learning of a request for a new incoming connection, rather than accepting a single connection, it attempts to accept as many incoming connections as possible. Calls to accept are repeated until a value of EWOULDBLOCK is returned, which indicates that there are currently no more outstanding connections that can be established. The results of Chandra and Mosberger's experiments were contrary to conventional wisdom which believed that select-based servers perform poorly under high loads.

Our work in this paper is largely motivated by this recent work by Chandra and Mosberger [6]. We believe that their work demonstrates that even simple server designs exhibit a wide range of variation in performance that is not well understood. In this paper we attempt to characterize the behaviour of some of these design options and use these results to gain insight into some of the issues affecting server designs in general.

Our work differs from previous work in that we investigate a variety of techniques for improving the performance of web-servers by concentrating mainly on the software architecture of the server. Specifically, we are interested in determining which aspects of different server application designs contribute to, or help to prevent, server meltdown during periods of high load. The focus

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

of much of this paper is on the interplay between how the server accepts new incoming connections, obtains information about kernel events, and uses that information to process existing connections. We believe that this approach has provided us with a better understanding of techniques that can be deployed within the application to significantly improve performance.

III METHODOLOGY

In this study, we use as a starting point the micro-servers developed by Chandra and Mosberger [6] to examine scalable event-dispatching mechanisms in Linux. We focus on their "multi-accept" server which provides the highest performance of all of the servers and kernel dispatching mechanisms considered in their study. We use the core of this server to create a new, highly parameterized micro-server that is designed to permit us to quickly and easily explore a wide variety of options with respect to implementing various aspects of the web server.

This approach is both necessary and important. In addition to creating a framework within which different options can be explored, it ensures that any differences in performance are actually due to differences in the software architecture of the server and not due to other artifacts of the implementations being compared (e.g., differences in the caching algorithms or numbers of file descriptors being used). While this approach is attractive, it is not without its drawbacks. Perhaps the main drawback is that it is not feasible to examine and compare all combinations of parameters. Fortunately we have found that we've been able to apply insights gained in some experiments to eliminate the need to explore some combinations of options. Although we have not done an exhaustive study of all combinations of options or performed a completely systematic elimination of different combinations, we have been able to explore combinations of options that result in quite significant improvements in server throughput. Interestingly, we've found some seemingly minor modifications to the server can have a significant influence on the resulting performance of the server.

IV SERVER IMPLEMENTATION

We have designed and implemented our server to permit the exploration of several issues related to the implementation of high-performance web-servers. Again, our goal is to be able to provide a controlled environment in which we can fairly and accurately compare various design and implementation options.

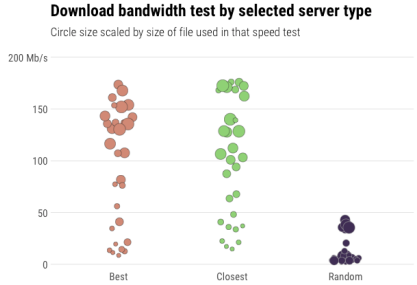
Among some of the issues we have examined (not all are reported on in this paper) are how performance is impacted by: aggressively accepting new connections, the order in which the open socket descriptors are processed, the size of the listen queue used in accepting new connections, and how that choice interacts with the size of the TCP SYN queue. We have also explored how caching impacts the design of high-performance web-servers.

.

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

V EXPERIMENT RESULTS

```
servers <- spd_servers(config=config)

closest_servers <- spd_closest_servers(servers, config=config)

only_the_best_severs <- spd_best_servers(closest_servers, config)

glimpse(spd_download_test(closest_servers[1,], config=config))

## Observations: 1

## Variables: 15

## $ url "http://speed0.xcelx.net/speedtest/upload.php"

## $ lat 42.3875

## $ lng -71.1

## $ name "Somerville, MA"

## $ country "United States"

## $ cc "US"

## $ sponsor "Axcelx Technologies LLC"

## $ id "5960"

## $ host "speed0.xcelx.net:8080"

## $ url2 "http://speed1.xcelx.net/speedtest/upload.php"

## $ min 14.40439

## $ mean 60.06834

## $ median 55.28457

## $ max 127.9436

## $ sd 34.20695
```



Internet performance of the local servers.

V CONCLUSION

In this application, we are analysing the performance of the network providers on the basis of the upload and download speed . By this analysis we will provide the performance of best servers within the local servers. This analysis helps the novice users to know the best service provider within their area.

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

VI FUTURE ENHANCEMENTS

We will provide an analysis report on the performance of the servers regarding upload and download speed. By designing an interface to this application helps the more users to know their network speed and enhance their performance.

VII REFERENCES

[1] M. Arlitt and T. Jin. Workload characterization of the 1998 world cup web site. IEEE Network,
14(3):30–37, May/June 2000.
[2] G. Banga, P. Druschel, and J.C. Mogul. Resource containers: A new facility for resource management
in server systems. In Operating Systems Design and Implementation, pages 45–58, 1999.
[3] G. Banga and J.C. Mogul. Scalable kernel performance for Internet servers under realistic loads. In
Proceedings of the 1998 USENIX Annual Technical Conference, New Orleans, LA, 1998.
[4] G. Banga, J.C. Mogul, and P. Druschel. A scalable and explicit event delivery mechanism for UNIX.
In Proceedings of the 1999 USENIX Annual Technical Conference, Monterey, CA, June 1999.
[5] P. Bhoj, S Ramanathan, and S. Singhal. Web2K: Bringing QoS to web servers. Technical report, HP
Laboratories, HPL-2000-61, May 2000.
[6] A. Chandra and D. Mosberger. Scalability of Linux event-dispatch mechanisms. In Proceedings of the
USENIX Annual Technical Conference, Boston, MA, 2001.
[7] S.L. Graham, P.B. Kessler, and M.K. McKusick. gprof: a call graph execution profiler. In Proceedings
of the SIGPlan '82 Symposium on Compiler Construction, pages 120–126, Boston, MA, 1982.
[8] J. Hu, I. Pyarali, and D. Schmidt. Measuring the impact of event dispatching and concurrency models
on web server performance over high-speed networks. In Proceedings of the 2nd Global Internet
Conference. IEEE, November 1997.
[9] J. Mogul and K. Ramakrishnan. Eliminating receiver livelock in an interrupt-driven kernel. In Proceedings of the USENIX Annual Technical Conference, pages 99–111, San Diego, CA, 1996.
[10] D. Mosberger and T. Jin. httperf: A tool for measuring web server performance. In First Workshop on
Internet Server Performance, pages 59—67, Madison, WI, June 1998. ACM.
[11] M. Ostrowski. A mechanism for scalable event notification and delivery in Linux. Master's thesis,
Department of Computer Science, University of Waterloo, November 2000.
[12] V. Pai, P. Druschel, and W. Zwaenepoel. Flash: An efficient and portable web server. In Proceedings
of the USENIX Annual Technical Conference, Monterey, CA, June 1999.
[13] N. Provos and C. Lever. Scalable network I/O in Linux. In Proceedings of the USENIX Annual
Technical Conference, FREENIX Track, June 2000.
[14] N. Provos, C. Lever, and S. Tweedie. Analyzing the overload behavior of a simple web server. In
Proceedings of the Fourth Annual Linux Showcase and Conference, October 2000.
[15] K.K. Ramakrishnan. Scheduling issues for interfacing to high speed networks. In Proceedings of the
IEEE Global Telecommunications Conference, pages 622–626, Orlando, FL, 1992.

ISSN: 2236-6124
Volume 09 Issue 01
Jan-June 2019

[16] T. Voigt, R. Tewari, D. Freimuth, and A. Mehra. Kernel mechanisms for service differentiation in
overloaded web servers. In Proceedings of the USENIX Annual Technical Conference, Boston, June
2001.
[17] L.A. Wald and S. Schwarz. The 1999 Southern California seismic network bulletin. Seismological
Research Letters, 71(4), July/August 2000.
[18] M. Welsh and D. Culler. Virtualization considered harmful: OS design directions for well-conditioned
services. In Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS VIII), Schloss Elmau, Germany, May 2001.

AUTHORS:

A.THRIVENI :- B. Tech Student, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem,NH-16, Guntur Andhra Pradesh, India.

CH.SUDHEER:- B. Tech Student, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, NH-16, Guntur Andhra Pradesh, India.

M. PAVAN KUMAR:- B. Tech Student, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem, NH-16, Guntur Andhra Pradesh, India.

B PRIYANKA : B. Tech Student, Department of CSE, Sri Mittapalli College of Engineering, Tummalapalem NH-16, Guntur Andhra Pradesh, India.