International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Management and Utilization of Database Connection Pool for Quality of Services

Tanuja Sharma

Department Of CSE (M.Tech), MDU Rohtak, Haryana, India

Abstract: ***It is quite difficult to maintain the performance of the web application server resources, that are held exclusively by a service request for the duration of its execution such exclusively-held server resources become performance bottleneck points, with failures to obtain such a resource constituting a major portion of request rejections due to server overload conditions. Because: (1) complex software systems are component middleware that using several independently maintained server resource management mechanisms; (2) session-oriented client behavior of complex data access patterns which makes it difficult to know the impact of tuning these mechanisms has on application behavior; and (3) huge ranging execution having complexity of different types request that exhibit complex structural organization behavior of component based internet services themselves. In this paper we show methodology to compute the optimal pool sizes of two resources i.e. thread and data base connection .How clients use Internet services enables mechanisms that achieve two interconnected goals: (1) providing improved QoS to the service clients, and (2) optimizing server resource utilization. In this paper review on improve server request on web application, optimization of server resources pool, data base connection pooling and quality of service.***

Keywords: client behavior, optimization of server resource pool, quality-of-service, web application server performance; database connection pooling

1. Introduction

Modern Internet services such as e-mail, banking, online shopping, and entertainment web sites have become common place in recent decade. Providers of these services often implement them as applications hosted on web application (middleware) servers, such as IBM Web Sphere, Oracle Web Logic, and Red Hat JBoss. The emergence of these web portals marked a shift from monolithically structured web sites with static read-only content, which were typical for the early Internet, to complex services that provide richer functionality, and dominate the modern Internet.

It become critical for service provider to deal with when server get over load due to it service performance of server get degraded (decrease).Complex software systems of middleware platforms are that expose to system administrators several mechanisms that can be independently maintained to improve Internet application performance and optimize server resource utilization[10]. On middleware layer CPU scheduling, memory management etc work to control over low level resource and on high level threads, data base connection work, etc. number of user shared some resources concurrently, some part of resource get shared and other which is non shared become failure to obtain such resources constitute major portion of resource get rejected due to overload. It creates a pools limited number of thread and data base and also schedule them in order to the incoming request. Optimal means, using more thread and data base connection to increase the execution parallelism. But it can degrade the performance due to thread context switching and increase data and lock the contention in the database. Optimal number is used to execute requests of different set of application components and middleware services. Component-based internet services are difficult to maintain the performance because (1) complex software systems are component middleware that expose several independently maintained in an attempt to improve application performance and optimization server resource management mechanisms; (2) complex session-oriented client behavior of complex data access patterns which makes it difficult to know the impact of tuning these mechanisms has on application behavior; and (3) huge ranging execution complexity of different types request that exhibit complex structural organization behavior of component based internet services themselves .

TPC-W application-To test the compute optimal values of critical resource pools, we use the TPC-W transactional web e-Commerce benchmark [9], The TPC-W specification describes in detail the application data structure and the web invocations that constitute the web site functionality, and defines how they change the application data stored in the database. A typical TPC-W user session consists of the following requests: user starts web site navigation by accessing the Home page, searches for particular products (Search), retrieves information about specific items, adds some of them to the shopping cart, initiates the check-out process, registering and logging in as necessary, and finally commits the order.

These services on internet share several common characteristics have implement the way they are structured and use by their clients.

- Session-oriented usage by clients: The typical interaction of users with such services is organized into sessions, a sequence of related requests, which together achieve a higher-level user goal.
- Complex data access patterns: In certain service requests not only read but also write application data Moreover, concurrent requests coming from different clients can access and modify shared application data. Data access patterns become even more complicated when the application data is replicated .The outcome of a request execution depends on the data sources it

accesses, and may be influenced by concurrent user requests.

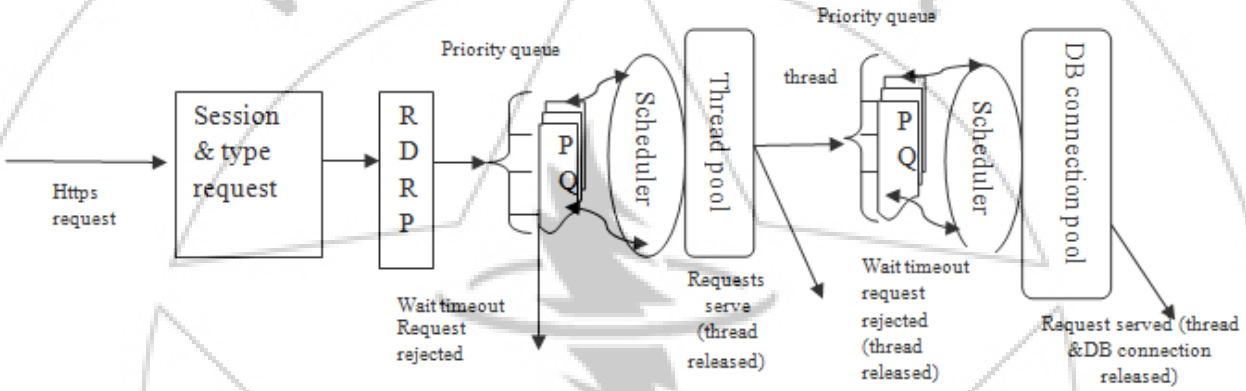
- Use of component middleware: A growing number of services utilize component middleware such as the Java Platform Enterprise Edition (Java EE) framework [Java EE 2011] as their building platform. Consequently, a web application's behavior depends not only on the way it is programmed, but also on the way it is assembled, deployed, and managed at runtime.

2. Background-Methodology

In this work we focus on solutions that target the following interconnected goals:

1. Model use for request execution with 2 tier exclusive resources holding (1st tier thread & 2nd database). Request middle database connection caching with the standard transaction wide database connection caching.
2. Compute the optimal number of database connection and thread for a given internet application its server and database environment and specific user load.
3. providing improved QoS[2] guarantees to the clients of Internet services;
4. achieving optimal server resource utilization; and
5. Providing application developers with the guidelines for natural application structuring that enable efficient use of the proposed mechanisms for improving service performance.

1. Model Use for Request Execution with 2 Tier Exclusive Resources Holding



Request is executed by web application server and connection is established if time out for obtaining the thread time request is rejecting with message. Data base connection is obtained by request available for the exclusive use by the request until the request is processed.

How it works: To access the data base request obtained a data base connection, from data base connection pools. After request work is done over the data base connection and it returned to the resource pool. It can request from the pool many times during the execution of a single service request. This process also reduces the number of connection revoked from the data base. 2-level model of request execution and the flow of a request through the system, with the adopted request-wide database connection caching

2. Compute the optimal number of database connection and thread for a given internet application its server and database environment and specific user load

Here we compute optimal number- where M is number of threads and N is number of database connection where (M ≥ N), In our 2-tier for request execution model, request execution time can be represented as follows:

$$t = w^{THR} + p + w^{DB} + q$$

Where w^{THR} times spend by request in waiting for thread, p is the time the request spends on processing before getting a database connection, w^{DB} is the time spent by the request in waiting for a database connection, and q is the time the request spends processing with a database connection in its possession. Note that for requests that do not access the database, $w^{DB} = q = 0$.

In an optimal server configuration we would want to achieve a balanced utilization of server threads and database connections. This means that under maximum sustained user load, we would want all threads and database connections to be fully utilized. Number of idle threads or idle database connections in situations where database connections or threads (respectively) become the resource bottleneck is a waste of server resources and will cause performance degradation.

Step1- Obtain the values of p_i and q_i for some subset of possible values of M and N. Where i is request types in a session.
Step 2- The obtained data points for functions $p_i(M,N)$ and $q_i(M,N)$ for domain function.

$(M \geq N > 0)$

Step3-we get the value of maximum sustainable session throughput $\lambda(M,N)$, for every possible combination of M and N. We compute session throughput by following equation.

$\lambda(M,N) = \min\{\lambda DB(M,N), \lambda THR(M,N)\}$.

For functions pi(M,N) and qi(M,N).with only N database we cannot compute, on average not more than session per

unit time **λDB(M,N) =N/ i Viqi** and similarly for M database **λTHR(M,N) =M/i Vi(pi + qi)**

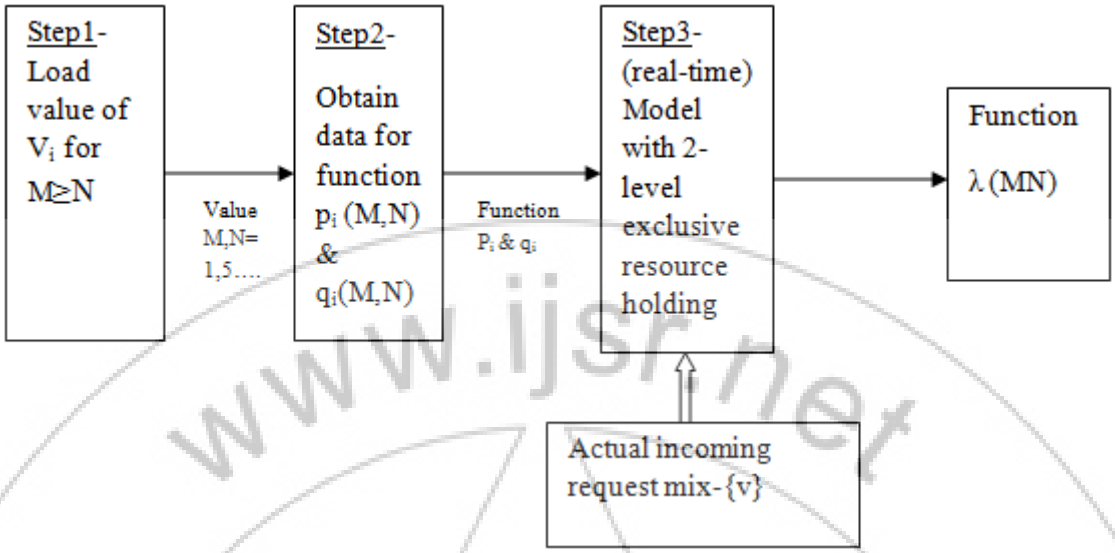


Figure 2: Steps of to compute the optimal number of server threads and database connections

3. To Provide Solution to Last Three Targeted Interconnected Goal

Session-oriented mechanism is use for component-based Internet services by their clients that helps to (1) improve QoS delivered to the clients; (2) optimizing management and utilization of server resource; and (3) application developers that provide the guidelines for natural application structuring mechanisms for improving service performance.

Reward-driven session prioritization (RDRP)[11] mechanisms that try to maximize reward attained by the service, by dynamically assigning higher execution priority values to the requests whose sessions are likely to bring more reward. It schedule the incoming request available to thread and DB connection with the request priority set by RDRP algorithm (FIFO is used).Request which is unable to connect to thread and database connection within predefined time interval, it get reject with an explicit message. RDRP algorithm work as follows.

1. For every incoming request, it observe sequence of request already exist in the session and compare it with incoming sequence of request with knows CBMG structure of session type
2. For each session type of CBMG (probability of session type of customer behavior modern graph), it compute its expected reward and execution cost. This information is used to get non-conditional values for future session request.
3. Priority of request is defined; its session reward is divided by its expected execution cost for future session requests.
4. Finally priority of request is define, which determine and control the schedule in the existed thread and DB connection to incoming requests.

3. Future Scope

Internal complexity of centralized data Internet services and their hosting environments will grow in the future; the only way to focus efficient management and beneficial utilization of these systems is to curtail their external complexity, as it is exposed to the system administrators. To limit the exposed management complexity of the service is to automate some of the service management aspects, which contrast with current primarily manual approaches for the same task.

4. Conclusions

Specifically, the contributions of this work are the following:

- ✓ For maximizing profit attained by the service in the overload situations and improving the QoS delivered to users by using Reward-driven session prioritization schemes
- ✓ For a given maximum of client requests using model of request execution with 2-tier exclusive server resource holding (threads, database connections),
- ✓ Analytical models of concurrent web session execution with bounded inconsistency in shared application data, which are able to accurately predict the values of QoS metrics of interest.

References

[1] T. F. Abdelzaher, K. G. Shin, and N. Bhatti. "Performance guarantees for web server end-systems": A control-theoretical approach. IEEE Transactions on Parallel and Distributed Systems, 13(1):80–96, 2002.
[2] T. F. Abdelzaher and K. G. Shin. "QoS provisioning with Contracts in web and multimedia servers". In

Proceedings of the IEEE Real-Time Systems Symposium (RTSS'99), December 1999.
[3] J. Almedia, M. Dabu, A. Manikntty, and P. Cao. "Providing differentiated levels of service in web content hosting". In Proceedings of the First Workshop on Internet Server Performance, June 1998.
[4] J. Almedia, M. Dabu, A. Manikntty, and P. Cao. "Providing differentiated levels of service in web content hosting". In Proceedings of the First Workshop on Internet Server Performance, June 1998.
[5] H. Chen and P. Mohapatra."Session-based overload control in QoS-aware web servers". In Proceedings of the IEEE Conference on Computer Communications
[6] L. V. Ahlfors. Complex Analysis. McGraw–Hill, 1979.
[7] ACM/IFIP/USENIX International Middleware Conference. Lecture Notes in Computer Science, vol. 2672, Springer, Berlin.
[8] TPC-W. 2005."Transaction Processing Performance Council". Transactional web e-commerce benchmark. http://www.tpc.org/tpcw/.
[9] TPC-W-NYU. 2006. "A Java EE implementation of the TPC-W benchmark." http://www.cs.nyu.edu/totok/professional/software/tpcw/tpcw.html.
[10] TOTOK, A. AND KARAMCHETI, V. 2010a. Optimizing utilization of resource pools in web application servers". Concurrency Comput: Pract. Exper. 22, 18, 2421–2444.
[11] TOTOK, A. AND KARAMCHETI, V. 2010b. RDRP: "Reward-driven request prioritization for e-commerce websites"". Electron. Commerce Res. Appl. 9, 6, 549–561.
[12] L. Cherkasova. "Scheduling strategy to improve response time for web applications". In Proceedings of the International Conference on High Performance Computing and Networking (HPCN Europe'98), April 1998

Author Profile

Tanuja Sharma, Pursing Mater of Technology from Department of Computer Science & Engineering ,World College of Technology and Management(Approved by AICTE Govt. & Affiliated to Maharshi Dayanand University,Haryana, India