AutoPerf: An Automated Load Generator and Performance Measurement Tool for Multi-tier Software Systems

Shrirang Shirodkar
Department of Computer Science and Engineering
IIT Bombay
Powai, Mumbai - 400 076, India
shrirang@cse.iitb.ac.in

Varsha Apte
Department of Computer Science and Engineering
IIT Bombay
Powai, Mumbai - 400 076, India
varsha@cse.iitb.ac.in

ABSTRACT

We present a load generator and performance measurement tool (*AutoPerf*) which requires minimal input and configuration from the user, and produces a comprehensive capacity analysis as well as server-side resource usage profile of a Web-based distributed system, in an automated fashion. The tool requires only the workload and deployment description of the distributed system, and automatically sets typical parameters that load generator programs need, such as maximum number of users to be emulated, number of users for each experiment, warm-up time, etc. The tool also does all the co-ordination required to generate a critical type of measure, namely, *resource usage per transaction or per user* for each software server. This is a necessary input for creating a performance model of a software system.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Testing tools*; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Experimentation, Measurement, Performance

Keywords

load generators, profilers, capacity analysis, distributed systems

1. INTRODUCTION

Performance measurement on a system test environment, using load generator tools, is an essential step in the release cycle of any Web-based application that is built for use by a large number of simultaneous users. Performance measurement of such applications can be done with two goals: the first and common goal is to experimentally characterize the performance of the application by treating the server system as a "black box". Thus all performance measures are recorded at the clients that generate the requests. The second goal is to *profile* the server, so as to obtain measurements that can be used in a *performance model* of the application [2, 5]. For e.g., a queuing model of a software server would require as a parameter, the CPU time taken by a server to process one request. This can be obtained by profiling the server during a measurement experiment. Models can be used for extrapolating the performance of the system for scenarios which are not available in the testbed [2].

Performance measurement, commonly known as "load testing", involves synthetic load generation on the the Web-server, in a way that exercises various scenarios, which represent user behavior. A typical load testing exercise is done by testing the server system at various "load levels" (from low to high), where the load level is typically specified as the number of virtual users who carry out a request-response cycle. Measures such as the response time seen by the clients, and the maximum request rate, or maximum number of users supported by the application, can then be determined experimentally. A number of "load generator" tools [1] exist that provide a rich suite of features such as a GUI, support for a variety of protocols, management of multiple experiments using a database, and generation of graphs for various measures.

The existing tools, however, have two limitations: first, they are focussed mainly on quantifying the user- perceived performance of the system, such as response time versus number of simultaneous users, throughput versus number of users, etc. However, *server profile* measures are currently not provided directly by any of the existing tools. Generating such measures requires quite a bit of tedious co-ordination of several unrelated tools by the performance tester.

The second problem is that existing load generator tools require several configuration values to be input manually by the tool user. The correct values of these parameters (e.g. number of users that are required to stress the system or warm-up time of an experiment) need to be simply "guessed" by the performance tester.

In this paper, we present a tool, *AutoPerf* that addresses these problems. The only information that AutoPerf needs is the URLs of the transactions to be analyzed, a specification of the user behavior, and the deployment details of the system under study. The tool automatically and efficiently, runs the suite of tests needed to comprehensively characterize the performance of the application. The tool measures the client-perceived performance from low load to saturation load on its own, and in addition, generates server profile measures that can be used to build a queuing model of the system.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

2. KEY FEATURES AND MECHANISMS OF AUTOPERF

AutoPerf has several features and mechanisms that distinguish it from existing load generator tools. We list these below:

a) *Probabilistic User Sessions*: A user session in AutoPerf can be specified as a probabilistic session, termed as the "Customer Behavior Model (CBMG)" [2] (i.e. a list of URLs and probabilities of going from one URL to the other in a single session).
b) *Minimal System Specification*: AutoPerf requires only the following to be specified (in XML format): 1) The CBMG, 2) the IP addresses of the machines on which the servers are deployed and 3) the names of the server processes that are to be profiled. Then, once the profiling agents and the master controller are started - the entire process of load testing and profiling is done automatically.
c) *Automated Capacity Analysis*: Given the above input, AutoPerf reports the following measures which characterize the system capacity: the maximum throughput (in terms of requests per second) of the system and the maximum number of users after which throughput flattens or drops, response time versus number of users, throughput versus number of users, for the range of one to maximum number of users.
d) *Server-side profiling and correlating*: An essential feature of AutoPerf is the automated co-ordination of load generator and the server profilers for generation of the correlated server performance profiles. AutoPerf produces the following server-side profiles: CPU utilization (for all server processes, at each load level, where load level is the number of concurrent users), CPU ms per process per transaction, and memory required per process for each additional user.
e) *Maximum number of users:* AutoPerf uses a mechanism based on a queuing theory result [4], which estimates the "saturation number", or the maximum number of users supported by the system. Thus, this does not have to be specified manually.
f) *Determination of Load Levels:* Once the maximum number of users is estimated, AutoPerf needs to run experiments at various load levels, so that a reasonably smooth plot of response time vs number of users, or throughput vs number of users can be obtained. Here, carrying out experiments at too many load levels may result in the experiment taking too much time, whereas too few load levels may result in a plot that does not capture the behavior of the measure accurately enough. AutoPerf determines these load levels automatically, so that a smooth plot is generated, while minimizing the number of experiments required. The algorithm for determining the the load levels is described in [3]. Figure 1 shows an example of a plot of throughput versus number of users for a web calendar application. AutoPerf needs 7 experiments to generate this plot which matches very well with the one generated by running the experiment at each load level from 1 to 47).
g) *Warm-up detection:* For every load level, AutoPerf generates load for some duration, detects the point at which the system has "warmed up", and only then starts recording performance measures. This eliminates effects of any transient values in determining averages.

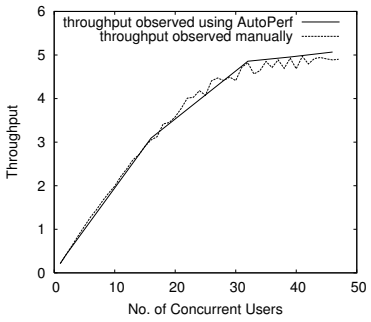

Figure 1: Throughput curves obtained manually and that using AutoPerf

h) *Determination of number of repetitions of transactions per user:* Since one of the quantities to be determined by AutoPerf is the per-transaction resource usage time, it is important to carry out a large enough number of transactions, so that numerical errors (e.g. rounding off to zero if the values are small) do not occur. AutoPerf determines the minimum number of transactions that each user should perform (termed "execution count"), so that an accurate estimate can be made of resource consumption per transaction. Currently, the execution count is determined based on CPU ms precision.

AutoPerf itself has minimal resource requirements. The load generator's memory usage increases at a rate of 520 KB per virtual user. When load is generated on a very fast server with zero think time, the CPU utilization at the load generator increases at a rate of 0.3% per virtual user on a Intel(R) Xeon(TM) dual CPU (each hyper-threaded) 2.80GHz machine.

3. SUMMARY

We have introduced a tool that requires minimal system description for carrying out a full-fledged suite of load testing experiments on an application. The tool also generates server resource usage profiles, which are required as an input to performance models. Autoperf thus has potential to be used as a part of an automated performance modeling framework, where a tool would automatically discover the data and characteristics of a system and create a performance model from such data.

4. REFERENCES

[1] R. Hower. *Web Test Tools.* http://www.softwareqatest.com/, 2007.

[2] D. Menasce and V. Almeida. *Scaling for E-Business.* Prentice-Hall, Inc., Upper Saddle River, NJ 07458, 2000.

[3] V. Selot. Automated tool for resource profiling and capacity analysis of distributed systems. Master's thesis, Indian Institute of Technology, Bombay, July 2006.

[4] K. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Pearson Education, Inc., 201 W. 103rd Street, Indianapolis, IN 46290, 2002.

[5] C.U. Smith and L.G. Williams. *Performance Solutions, A Practical Guide To Creating Responsive, Scalable Software.* John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, 2002.