A Study of Contemporary System Performance Testing Framework

S

Alex Ng
Federation University, Australia

Shiping Chen
CSIRO Data61, Australia

INTRODUCTION

System Performance Testing is a vital activity spanning the Software Development Life Cycle (Siva & Sharma, 2006).There are different types of test: Functionality, Graphics, Collision, Performance, Gaming, User Interface, Progression, Correlation, Numerical, Conformance, Interoperability and many more. In this chapter, we find special interest in System Performance Testing. The purpose of system performance testing is to identify bottlenecks by measuring a system's responsiveness and scalability under a certain load (Sarojadevi, 2011). Load testing provides a means for assessing the behavior of a system under varying traffic burdens. Performance testing is able to provide simulation and modeling help enterprises experiment not just with load but also with specific situations, such as large numbers of users performing the same activity conducting a search or completing an online transaction simultaneously (Menascé, 2002). Performance tests conducted over time provide a good landscape trend which helps companies plan and test for future needs. Such testing can help determine what hardware resources need to be purchased to achieve desired future capabilities (Jain, 1991).

In this chapter, we discuss some of the contemporary system performance testing frameworks and present a general-purpose testing framework for both simple and complicated performance testing. We observed that few of the system performance testing frameworks available have been widely used in industry or the research community (Yigitbasi, Iosup, Epema, & Ostermann, 2009). Our framework proposes an abstraction to facilitate performance testing by separating the application logic from the common performance testing functionalities.

The rest of this chapter is organized as follows. We first provide a definition of system performance and the requirements for system performance testing framework. Afterward, we present a discussion of some of the contemporary system performance testing frameworks. Lastly, we propose a system performance framework with a sample implementation to demonstrate the applicability of this framework.

BACKGROUND

According to (Meier, Farre, Bansode, Barber, & Rea, 2007), Performance Testing is defined as the technical investigation done to determine or validate the speed, scalability, and/or stability characteristics of the product under test. Performance-related activities, such as testing and tuning, are concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives for the application under test.

(Meier et al., 2007) classify performance metrics into the following categories:

DOI: 10.4018/978-1-5225-2255-3.ch658

- **Network-Specific Metrics:** A set of metrics about the overall behavior of the network used to support the system.
- **System-Related Metrics:** A set of metrics helps identify the resource utilization of the system.
- **Platform-Specific Metrics:** A set of metrics related to software that is used to host the application system, such as the Microsoft.NET Framework common language runtime (CLR) and ASP.NET-related metrics.
- **Application-Specific Metrics:** These include custom performance counters inserted into the application code to monitor application health and identify performance issues.
- **Service-Level Metrics:** A set of metrics help measure overall application throughput and latency, or they might be tied to specific business scenarios.
- **Business Metrics:** These metrics are indicators of business-related information, such as the number of orders placed in a given timeframe for a particular department.

There are some common system performance metrics for enterprise systems, such as: *Response Time, Latency,* and *Throughput*. In some contexts it's customary to call these things by different names: *Throughput and Response Time, or Capacity and Delay, or Bandwidth and Latency*. We provide the following definitions to avoid ambiguity:

- **Response Time (RT):** It is the total time taken by a client to wait in invoking a server function and coming back with a result. Oxford Dictionary defined RT as *the length of time taken for a person or system to react to a given stimulus or event*.
- **Latency:** It is the total time spent by a system generated message to travel from its source to the destination. According to Oxford Dictionary, "latent" means "existing but not yet developed or manifest". Together, latency and bandwidth define the speed and capacity of a network. For an online system, latency is the total time spent by a message from its sender (source) to its receiver (destination).

In a synchronous client/server-based environment, RT is equivalent to Latency, where the client is both the message sender (requester) and receiver (respondent). This is why these terms are sometimes used alternatively. However, they are different in the asynchronous mode because the sender and the receiver are two different parties. The total time spent by a client (RT) in invoking an asynchronous request and waiting for the result to come back will be different from the total time for a message (Latency) to get from the source to its destination because the message sender may be different from the message receiver.

Furthermore, there exists a term called *Processing Time* to describe the amount of time a system takes to process a given request, not including the time it takes the message to get from the user to the system or the time it takes to get from the system back to the user.

With this view, Latency + Processing Time = Response Time.

- **Throughput:** It represents the server's capability in handling a certain number of client requests (such as messages or transactions) within a unit of time (such as second or minute). This shows how well a company's facility is able to cope with the demand of its client.

From Figure 1, we can see the throughput of a typical system demonstrates the following conditions: (1) *Nominal Capacity*: maximum achievable throughput under ideal workload conditions. (2) *Usable capacity*: maximum throughput achievable without exceeding the pre-specified response-time limit, and (3) *Knee Capacity*: A condition that starts to deviate from the linear