Application Performance and a Cup of Milk

By Sai Subramanian, Cognizant Technology Solutions

After dabbling for a while with application performance testing, bottleneck identification and tuning, it helps to think of application performance with what I call a “Milk analogy”. In this analogy, the application is the milk, and the application infrastructure is the receptacle that holds the milk. Also, performance testing with a defined workload profile can be equated to the act of controlled heating of the milk. I am now going to try to make the analogy work for you and in the process share my thoughts related to Application Performance.

The simple (or complex) goal behind Performance Engineering is to analyze the adequacy of the application and the infrastructure to handle the expected load and develop a better understanding of its capabilities. Here’s an attempt to look at performance engineering from a different perspective.

Application (Milk)

In this paper, we’ll confine our focus to 3 tier web applications though the concepts are equally applicable to other architectures. Typically a web application has a presentation tier, application tier and a data tier. Determining the application’s performance issue at different tiers in isolation before integration may be helpful and preferred as it may be easier to focus on a particular tier at one time, but it is often neglected in a typical software lifecycle for lack of time or resources. Usually a comprehensive load test is performed for an application that has no critical or major functional defects.

Infrastructure (Receptacle)

The three application tiers mentioned above reside on a set of servers, namely web server, application server and database server. As every application is unique, the containers that hold them have to be configured uniquely to extract the best performance. It is also necessary to monitor the different servers and their behavior to tune the application and the infrastructure.

All the servers reside on an operating system that sits on the real hardware and communicate across networks. The performance of every component involved is important for the application to perform optimally, or as we all know Murphy’s law will have a perverse way of proving its validity.
Performance Testing (Boiling the milk)

Why do we need performance testing?

In the case of milk, it’s easier to determine the physical properties of milk at different temperatures by actually heating the milk than to determine/model it by investigating the constituents of milk.

Similarly, it is easier to determine performance capability of an application and its infrastructure by subjecting it to a workload profile that can be expected in production. It may be necessary to delve deep into the application or the infrastructure to determine certain performance issues but a carefully designed performance test would expose most of the issues that an application would experience in production. Fixing the issue is a different game altogether.

The quality of the performance test results entirely depends on the quality of test conducted. There are a variety of tools available in the market that will enable the performance engineers to understand the real user behavior to determine the workload profile.

Monitoring

In the case of our analogy, we have a receptacle with milk that is boiling, we need to make sure that we’re heating the container in a controlled manner, and we have sufficient tools to monitor the property of milk at different temperatures say, a hydrometer-to determine the purity of milk and a thermometer-to determine the temperature.

Whereas, for an application the performance objective can be summarized in terms of achieving a certain response time, throughput and utilization while the application is subjected to an expected production load.
Response time: It’s the time taken by the server to respond to a request, it can be either TTFB (Time to First Byte) or TTLB (Time to Last Byte). There are many performance testing tools available in the market that can provide these numbers.

Throughput: It’s usually measured by the number of requests per unit time or the number of bytes downloaded per unit time. As mentioned above many of the performance testing tools can provide this metric.

Resource Utilization: There are a variety of tools that are available to determine the resource utilization for a workload profile. The primary resources that are typically monitored are:

- CPU
- Memory
- Disk I/O
- Network I/O

Analysis

A detailed analysis of the test results from different sources has to be performed and correlated to determine the performance capability of the application in the defined infrastructure. There are a number of tools available to perform the analysis and there are performance testing tools that can be integrated to get a holistic picture. Based on the criticality of the performance issue, granularity of profiling and tracing will be used.

Profiling is the summary statistics of performance metrics that includes

- Method level invocation statistics (Number of times a routine was invoked, Structure of invocations, etc.)
- Heap size statistics (Memory, GC, message communication sizes, etc)
- Thread level statistics (Thread synchronization)

Tracing is a process of determining, when and where events took place along a global timeline

- Time-stamped log of events
- Message communication events (sends/receives) are tracked
- Shows when and from/to where messages were sent

Based on the analysis performed and the applications’ deviation from the expectation either the application is approved to production or further tuning of the application is recommended.

Performance Tuning

On boiling the milk a thin layer containing cream and casein is formed that creates pressure below it which eventually results in the milk overflowing out of the container. Overflow of milk can be avoided by either using a very large container or by just popping the thin layer on top continuously or using simple techniques to release the steam that is formed below the thin layer of cream and casein.
Similarly, in the case of a software application either infrastructure can be beefed up or simple performance tuning techniques can be used such as GC tuning, SQL tuning or changing few parameters with the JVM, operating system etc.

One exception to the analogy is the end result. As we keep boiling the milk for a long duration we get an empty container that requires a lot of effort to clean whereas at the end of the Application Performance Engineering effort, we get a nicely tuned application that can boldly face production.

Pour yourself a cup of milk!

Citations:

The operating system placement image by Golftheman is available under Creative Commons Attribution-Noncommercial license.

The image of the milk is from http://www.breakingtheviciouscycle.info/beginners_guide/yoghurt/heating_pad_yoghurt.htm