Performance Engineering for Free

By Andrew Lee
Whitney Associates Ltd

Introduction

There has been a growing increase in the interest and use of OpenSource or free software over the recent years. The most popular is probably Linux, the free Unix-like operating system.

As organisations attempt to trim their IT costs, the attraction of "something for free" has helped see the adoption of this type of software into mainstream commercial organisations. This paper will examine whether free software is available and usable as part of the performance engineering process.

What is free software?

Obviously any software you do not have to pay a purchase price for is free. However, as this paper will consider, there are two major types of free software - OpenSource and Commercial-driven.

OpenSource software is where a group of developers or an organisation will freely distribute the source code of the software they have developed. In addition people can usually join the group of developers and help make additions or perform bug fixes to the software. The web has been a great enabler for OpenSource software as it allows a geographically separated group of developers to collaborate. Organisations, such as SoftwareForge.net, exist to host the collaborative working environment for a group of software developers.

Commercial-driven software is where a commercial software company may make available a software product free to its customers, either to gain market share or to help these customers achieve certain objectives. The classic case of using free software to gain market share is Microsoft who distributed their *Internet Explorer* browser software free with their operating system.

What are the implications of free software?

The section below identifies some of the advantages/disadvantages of free software.

Advantages

- It's Free – There is no charge for the software, although some organisations may charge for technical support.
- Easy Access – Often the tools are available for download from the web so you can easily download and try them and distribute as many copies as you want in your organisation.
- Rapid Upgrades – Where there is a large group of developers working on a project, bug fixes and patches can be available in days.
- No Hidden Secrets – Since the source code is available, you can review the code and understand exactly what it does (if you have the time).

Disadvantages

- No Technical Support – Technical support is often only available by posting a question to a newsgroup where the usefulness and timeliness of the response will vary.
- No Liability – It makes no guarantees; if the product doesn't live up to your expectations, then tough!
- Poor Usability – Often the tools are not as slick as commercial products, particularly in areas such as installation scripts and user interfaces.

The Performance Engineering Life Cycle

The performance engineering lifecycle is a series of steps that, when undertaken during the system development process, will ensure a system meets the users' performance requirements.

The steps are briefly outlined below:

1) Performance Goals
   Collect volumetric information and set user performance-requirements.
2) Risk Assessment
   Review the project and identify the best method for reducing performance risk.
3) Performance Modelling
   Use simulation or analytical models to analyse the ability of the system design to meet the performance goals.
4) Performance Budgets
   Assign individual performance goals at the code level to ensure the developers understand the performance implications of the code they are writing.
5) Benchmarking
   During development, performance-critical components may need to be benchmarked to help de-risk the project.
6) Performance Testing
   Finally, the system must be tested to show that it meets the performance goals.

The two drivers for performing a performance engineering project are the costs of the staff and the cost of the tools. This paper will examine only the possibility of reducing the cost by using free tools. Therefore, the tools examined in this paper are for:

- Performance Modelling
- Profiling
- Performance Testing

Performance Modelling Tools

It is difficult to find any comparable Opensource tools that match the functionality of commercial modelling tools such as *Hyperformix Integrated Performance Suite (IPS)*. However, when considering only low-level modelling tools it is possible to compare commercial and free modelling tools. This section compares the commercially available *Hyperformix Workbench* with the free simulation tool called *Ptolmey*. In addition, some comments on Microsoft research in the area is given.

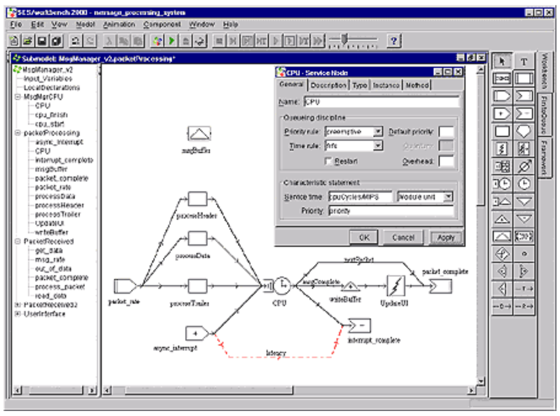
Hyperformix Workbench

Hyperformix Workbench is a system simulation tool designed to model concurrent systems. It is aimed at developers and system engineers who wish to assess real system designs. Originating from the University of Texas as a simulation tool to model concurrent systems, it has progressed to be a commercially-available tool.

A *Workbench* model is a hierarchy of sub-models. Each sub-model is represented by an extended directed graph. A directed graph is made up from 24 defined nodes that are connected together to form a model or a subsystem model. The 24 defined nodes break down into 9 resource management nodes, 9 flow control nodes, 4 sub model nodes and 2 declaration nodes. Transactions flow around the directed graphs using the modelled resources. The ability of using sub-models allows top/down and bottom/up design methods to be used.

The nodes in a directed graph are used to model the path and processes that a transaction can follow. There are the standard nodes such as serviced queue, a set delay, loops and branches. They also included interrupt nodes, resource nodes and nodes to create sibling concurrent processes. For each node there is a predefined list of options, which can be selected to configure the node. For example, a service node can choose the service time and a corresponding distribution as well as the queue discipline. The real advantage of *Hyperformix Workbench* is that, if no predefined options exist for a node, a user can specify his own by adding a C code procedure to that node.

The tool can be run in the batch mode or in an animation mode. When the model is run in batch mode various statistics can be collected and a report is generated at the end of the simulation run. The simulation reports collects the statistics that have been requested by the user. For each element, various statistics can be collected on the type of node. For



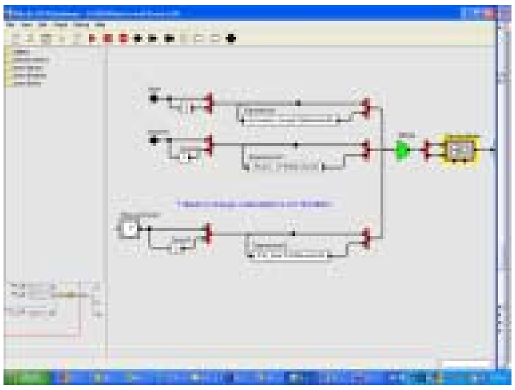
example, at a service node data can be collected on queue population, service time, etc. Certain statistics can be saved to a file and used to create more detailed plots. In animation mode a user can see transactions moving through the model. In addition various monitor points can be attached and on-screen graphs can be displayed as statistics are being collected.

The list price for *Hyperformix Workbench* is unavailable.

http://www.hyperformix.com

Ptolemy

The *Ptolemy* project aims to build Java-based tools that allow modelling and simulation of concurrent distributed systems. The work is mainly performed by the Department of Electrical Engineering and Computer Science at the University of Berkeley under the direction of Professor Edward Lee (no relation to the author). The focus is primarily on embedded systems, particularly those that mix technologies, and include, for example, analog and digital electronics, hardware and software, and electronics and mechanical devices. Therefore, *Ptolemy* provides different domain for different analysis approaches such as discrete event simulation, continuous simulation, finite state machines and Petri-nets. For the purpose of this paper we shall concentrate primarily on the discrete event simulation domain.



The discrete event simulation capability of *Ptolemy* consists of a notion of current time, and processes events chronologically in time. An event is a token with a time stamp. A 'token' is basically an occurrence, caused by a source, which changes or updates the state of the model.

ptolemy.eecs.berkeley.edu/

Microsoft Indy

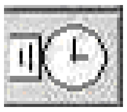
Not much is known about the *Microsoft Indy* tool other than that posted on the Microsoft research website. Microsoft describes *Indy* as "a framework for a modelling expert to develop custom model applications using pre-existing core components and services. Users can plug-in different hardware models, workload descriptions and external performance components. This lets Indy grow and adapt to the needs of its users." The primary applications for *Indy* are capacity planning tools and architecture modeling.

research.microsoft.com/sysperf/

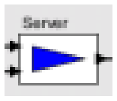
Workbench vs Ptolemy

At first glance there are many similarities between *Workbench* and *Ptolemy* but on further study the real differences seen are in terms of functionality and usability. For example, consider the functionality of a simple service centre that is fundamental to the modeling of software systems.

The *Workbench* service node looks like this -



while the *Ptolemy* service node looks like this:

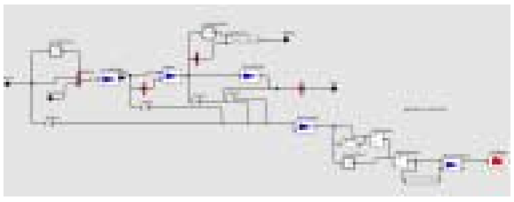


Both model a queue of service requests queuing for a service centre where they are then are delayed for the service time before departing the service node.

However, there is a much richer set of functionality with the *Workbench* service node which includes:

- Range of queue disciplines
- Choice of statistics
- Multiple service centers

You can of course use the other modeling nodes within *Ptolemy* to add this functionally, but the simple service node becomes a much more complex model. For example, to graph the queue length during the simulation and write the results to a file the model needs an additional 18 nodes as shown below.



Summary - Modeling Tools

In summary when comparing *Workbench* to *Ptolemy* you get what you pay for! With only a basic level of functionality available with *Ptolemy* it would take a considerable amount of effort to expand the tool to match the capability of *Workbench*. However, for organizations that cannot afford the high price of *Workbench* then *Ptolemy* at least offers an reasonable alternative.

Profilers

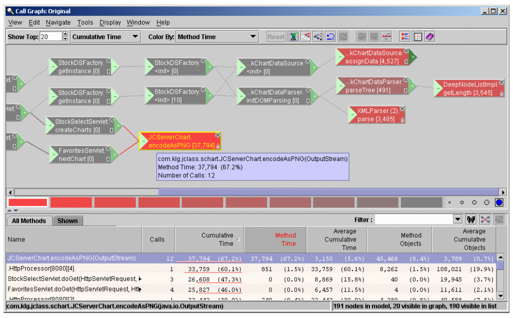
Although simulation tools will help identify the potential bottlenecks and supply the development team with performance budgets, code profilers will still be needed to identify the location of inefficient code. There are many commercial and free profilers available for many different computer languages. For the purposes of this paper we will consider those available for Java.

In the Java world there are many competing implementations of the Java engine which is the machine responsible for running the Java code. This engine is known as the Java Virtual Machine (JVM). Therefore, several vendors have profilers specifically for their JVM. For example, BEA has developed a profiler for its JRocket JVM and SUN has developed a profiler for its JFluid implementation.

This section will consider the commercial profiler *JProbe*, and the free profilers *JMP* and *Hprof*.

Jprobe

JProbe provides CPU profiling, and memory and thread analysis. As a commercial tool it aims to identify the source of the performance problem as quickly as possible. It therefore provides features such as a graphical call path trace (seen below) and integration with an IDE (integrated design environment). Through its integration with an IDE it is possible to click on a problem area identified by *JProbe* and the actual lines of code will be displayed in the developers' IDE.


Call Graph: Original
View Edit Navigate Tools Display Window Help
Show Top: 20
Cumulative Time
Color By: Method Time
Reset
StockDSFactory. getInstance [0]
StockDSFactory. <init> [0]
...kChartDataSource. <init> [0]
...kChartDataSource. assignData [4,527]
StockDSFactory. getInstance [0]
StockDSFactory. <init> [10]
...kChartDataParser. initDOMParsing [0]
...kChartDataParser. parseTree [491]
DeepNodeListImpl. getLength [3,545]
XMLParser (2). parse [3,485]
StockSelectServlet. createCharts [0]
FavoritesServlet. nextChart [0]
JCServerChart. encodeAsPNG [37,794]
All Methods
Shown
Filter :

com.klg.jclass.schart.JCServerChart.encodeAsPNG(OutputStream)
Method Time: 37,794 (67.2%)
Number of Calls: 12

Name	Calls	Cumulative Time		Method Time		Average Cumulative Time		Method Objects		Average Cumulative Objects	
JCServerChart.encodeAsPNG(OutputStream)	12	37,794	(67.2%)	37,794	(67.2%)	3,150	(5.6%)	45,468	(8.4%)	3,789	(0.7%)
.HttpProcessor[8080][4].	1	33,759	(60.1%)	851	(1.5%)	33,759	(60.1%)	8,262	(1.5%)	108,021	(19.9%)
StockSelectServlet.doGet(HttpServletRequest, H	3	26,608	(47.3%)	0	(0.0%)	8,869	(15.8%)	40	(0.0%)	19,945	(3.7%)
FavoritesServlet.doGet(HttpServletRequest, Htt	4	25,827	(46.0%)	0	(0.0%)	6,457	(11.5%)	4	(0.0%)	11,611	(2.1%)

com.klg.jclass.schart.JCServerChart.encodeAsPNG(java.io.OutputStream)
191 nodes in model, 20 visible in graph, 190 visible in list

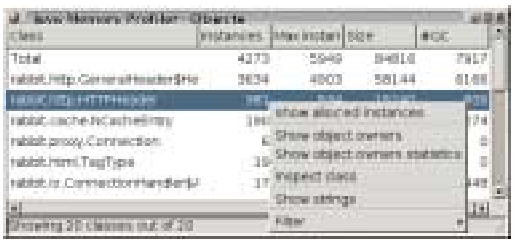
As a commercial tool the company is committed to supporting various vendors' JVMs and providing timely updated new versions of JVMs.

A single user license of *JProbe* is approximately £1800.

http://www.quest.com

JMP

JMP is a Java Memory Profiler that has been written by Robert Olofsson. As the name suggests, it provides information on the amount of memory required by each instance of a class in a Java program. *JMP* can perform heap analysis and has the ability to show which objects own (have references to) all the objects of a specified class, thus helping to detect memory leaks,


Java Memory Profiler - Objects

Class	Instances	Max instan	Size	#GC
Total	4273	5949	84816	7917
rabbit.http.GeneralHeader$He	3634	4803	58144	6168
rabbit.http.HTTPHeader	[illegible]	[illegible]	[illegible]	[illegible]
rabbit.cache.NCacheEntry	[illegible]	[illegible]	[illegible]	[illegible]
rabbit.proxy.Connection	[illegible]	[illegible]	[illegible]	0
rabbit.html.TagType	[illegible]	[illegible]	[illegible]	0
rabbit.io.ConnectionHandler$I	[illegible]	[illegible]	[illegible]	[illegible]

Showing 20 Classes out of 20

http://www.khelekore.org/jmp/

Hprof

Hprof is a command line profiler that is released as part of the Java Software Developer Kit. It provides options to analyse CPU, memory and thread usage. Since this is a command line tool it only produces textual output, which requires further analysis

Summary – Profilers

Free profilers can be used to debug performance problems although the additional features of commercial tools may mean that there is a price worth paying.

Performance Test Tools

Performance test tools are needed to confirm that a system can support the necessary workload and meet the users' response time requirements.

This section will consider the commercial tool LoadRunner and the free tools OpenSTA, Grinder and WAS.

Mercury LoadRunner

With over 50% of the market share in performance test tool, *LoadRunner* is the dominant player in the market place. It is primarily available in two variants - a Web Test tool and a DB test tool. Performance test scripts are created by recording a user's interaction with the system. Then the script is edited and parameterised so that is can effectively emulate multiple users (a performance test would be useless if all the users logged on with the same username!). Finally, the test scripts are run against the test system. Performance test tools have the concept of a Virtual User, each of whom mimics the work of a real user. Typically, the pricing for performance tools is dependent on a base price and then extra for additional virtual users which in some cases can be leased.

Typical *LoadRunner* installations cost within the range of £50-100K depending on which additional modules, such as system monitoring components and the number of virtual users, are required.

www.merc-int.com

OpenSTA

OpenSTA was originally a commercial load testing tool produced by Cyrano, but a decision was made to release it as an Opensource product in 2000 with the option for people to pay for additional technical support. However, what started out as a promising tool does not seem to have kept pace with the developments in technology, such as the new versions of browsers.

The last release of OpenSTA was over a year ago and there is little indication of any significant development effort.

http://opensta.org/

Grinder

Grinder is a specialist tool designed for performance testing of J2EE systems, in particular, those hosted on the BEA Weblogic application servers. It has been written by Paco Gómez and Peter Zadrozny who are BEA employees. There is a good book written by the Grinder authors discussing the performance testing of J2EE systems. The *Grinder* tool has been written for a particular purpose and for a technical audience.

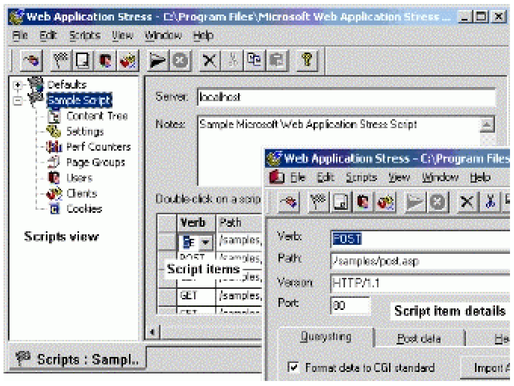
Grinder is an excellent example of a free tool that has a narrow technical market.

http://grinder.sourceforge.net/

Microsoft WAS

Microsoft offers the *WAS* (*Web Application Stress*) Tool.

WAS is a good tool for testing the basic functionality of a website, but it has limited scripting ability and therefore will not be able to create complex test scripts.



http://www.microsoft.com

Summary - Performance Test Tools

A good range of up-to-date, full-functionality, free, load-testing software does not seem to be available in the performance testing area. For simple testing and for specific technologies it may be possible to use free software, but otherwise a commercial tool will be needed to provide an accurate load test of your system.

Conclusion

Companies have been able to make cost savings by using Opensource tools such as the Linux operating system.

However, the benefits of Opensource software do not seem to be available to the performance engineering community and the reason for this seems to be that, as a small community, there does not seem to be enough "developers" who will work for free on the Opensource projects.

There seems to be some success for software tools that are produced by individuals aimed at specific technologies e.g *Grinder* or narrow functionality e.g *JMP*.

The biggest growth I expect to see is from vendors producing tools to support the use of their own existing tools. In particular, as Microsoft moves into the larger scale enterprise market it may seek to gain acceptance into this market place by distributing free tools to aid in the design and development of systems built using its technology.