Appendix B: Glossary

* **DBA / Database Analyst**: Person responsible for setting policies for and maintaining a database. A good DBA knows how make individual queries run faster and use fewer hardware resources.
* **Query Execution Plan**: A sequence of steps used by a database to execute a particular SQL statement in an efficient manner. Among other things, it describes the indexes used to efficiently join tables and apply WHERE clauses to locate and filter out data. The tools and techniques used to display a query’s plan vary between vendors, as does the syntax of the plan.
* **APM Tool / Application Performance Management Tool**: Stores and displays a large variety of current and historical performance data used to diagnose poor performance and other functional and availability issues. Often provides a facility to create a page (called a dashboard) of user-selected graphs of performance data from various sources. Data is gathered from operating system tools, Java JMX, Containers and other facilities. AKA, Application Performance Monitoring Tool.
* **Java Management Extensions / JMX**: A protocol and set of client and server Java libraries defined by JSR-160. Most Java **Containers** implement a server-side JMX facility to provide performance data. Clients like APM tools query JMX performance data to display for end users and.
* **Software Development Lifecycle / SDLC**: A rough, sequenced and team-oriented plan used to orchestrate many software development activities, such as design, code, test, and delivering software. Important in this book because traditionally, performance tuning was scheduled to be done so late in the cycle that there was not enough time or budget to fix all the performance issues before going into production.
* **Container**: Tomcat, WebSphere, WebLogic, Jetty and JBoss are all ‘containers’ that launch a Java server-side web or SOA application. It is debatable whether Spring Boot is a container, because it merely acts as a launcher/packager for other containers (like Tomcat and Jetty). The lines are somewhat blurry between “container” and “application server.”
* **Headless**: An approach for running software which is characterized as having little or no graphical user interface (GUI). For example, when you run JMeter by double clicking on jmeter.bat or jmeter.sh, it launches a multi-purpose GUI that records and creates load scripts, starts and stops load tests, and displays graphs/metrics from load tests. Alternatively, if you launch JMeter “headlessly” from a command prompt with certain parameters (-n and -t). This runs a load test and avoids any performance issues with the JMeter GUI. Also, headless mode is used to run various programs on Unix/Linux machines without a display/monitor. Most “server-side software”, the topic of this book, is also headless, like web and **SOA** applications.
* **Lowest hanging fruit**: An expression used to indicate just the small set of performance issues that are both significant/measurable drags on performance and also relatively easy to fix. This is analogous to the small set of fruit on a fruit tree that is both ripe for good eating and easy to pick because it is low to the ground and within reach.
* **JMS**: Defined in JSRs 343, 914, 368. A Java API for processing messages using queueing systems like Apache ActiveMQ.
* **Service Oriented Architecture / SOA**: A Java-servlet based application that receives and processes ‘service’ requests (often in json or xml) over protocols like HTTP/S and JMS. Other than a small console GUI, SOA apps are generally headless.
* **Correlation Variable**: A variable used in a **Load Script** that is used to capture the result of one request (perhaps HTTP) so it can be presented as input to a subsequent request. When Load Script is initially recorded, it can replay the verbatim activity of the single user that recorded the script. Correlation Variables are important because they help enhance a load script from being a verbatim recording of a single user’s traffic into a data-driven script that is capable of generating load of many difference users. See chapters 4 and 7 for how to add them to your load script.
* **Stack trace**: The ordered list of Java class/method invocations that led up to the currently executing Java class/method. Stack traces normally show up in your container’s application log files when a Java Exception occurs, but also when the JDK’s jstack is invoked to capture a **Thread Dump**.
* **Thread Dump**: a textual listing of all current running threads and a s**tack trace** for each which shows exactly the class/method name that was running in each thread at the time the thread dump was captured (often using the JDK’s jstack).
* **Server-side software**: Refers to Java software like web-applications and **SOA** systems that run on a server in a data center and not on an end user’s desktop/workstation. This book focuses on tuning server-side Java software.
* **I/O – Input/Output**: Disk or network access. I/O generally takes much more time than processing data already in memory, so it is a prime candidate for tuning. Displaying data on a screen is also I/O, but out-of-scope because server-side applications don’t write to displays. Tuning the web browser part (IE, Chrome, Firefox, etc...) of a web application is also out-of-scope (mostly) for this small book.
* **Load Generator**: A load generator is a network traffic generator used to stress out a network-enabled software application (like a SOA or the server-side portion of a web-application) to assess performance under production-like circumstances. When using using a load generator, the code that runs in the browser is generally not exercised and therefore should be performance tested using a different approach, such as with Selenium or Quick Test Pro – this kind of testing that actually launches real browser processes is out of scope for this book. AKA, network load generator. JMeter and HP Performance Center (previously Load Runner) are both load generators.
* **Load Script**: A script that details the precise network requests to be run by a network load generator. (AKA load plan).
* **Incremental Load Test**: A load test used to assess performance under increasing amounts of load. For example, a test that adds 5 threads of load every 4 minutes is an incremental load test.
* **Steady State**: A portion of a load test, after test warmup, where the same number of load generator threads is applied with constant think/pause time. If a test runs with 10 users for 10 minutes, then that test (excluding the warmup) is a steady state test. An incremental load test that loads 5 addition users every 5 minutes is not a steady state test – it is incremental.
* **Heap dump**: a point-in-time dump of all heap memory in the JVM. Heapdumps are key to solving memory leaks, as detailed in chapter 12.