A Brief Primer on Performance Testing

Summary:

Performance Testing was always a part of Software Testing whether explicitly stated or not. The world of software testing is becoming smarter and mature. With the changing landscape of Computing and a paradigm Industry wise shift towards SOA/web based applications has brought a forefront focus on Performance Testing and the expectations saught from this kind of testing are measurable numbers. I have tried to extract the entire essence in terms of minimal words which could serve the purpose to cut-short the learning time.

Performance Testing is about to measure the performance of an application under test whether the application is an embedded, desktop or a distributed enterprise application. However, these are the enterprise based applications/architectures where lies the prime focus of performance testing. Expectations saught from AUT is to measure the performance numbers and ensuring that it conforms to the expectations.

Goals of Performance Testing

The business goal of performance testing is to measure the application performance and ensure that the numbers conform to the Service Level Agreements. Goals can be internal (if the application is an in house project), or external (when the SLA's define the objectives).

External Goals: Conforming to the Service Level Agreements (SLA's). The SLA's at the highest level consists of the following parameters (whether or not explicitly stated).

- Application Response time.
- Application throughput.
- Maximum number of concurrent users
- Resource utilization in terms of various performance counters for example: CPU, RAM, network I/O, and disk I/O.

- Soak Tests[1] under varied workload patterns including normal load conditions, excessive load conditions, and conditions in between. This can include increase in the number of users, amount of data and so on.

Internal Goals:

- Application Crash. The application crash translates into a condition where the application either hangs or stops responding to requests. Some of the symptoms of breaking point include 503 errors with a "Server Too Busy" message, and errors in the application event log that indicate that the ASPNET worker process recycled because of potential deadlocks.
- Symptoms and causes of application failure under stress conditions. Recoverability options, whether the application recovers after a crash or not.

[1] Soak Testing is about measuring application performance over long periods of test run typically one would expect in a real production/live environment.

More importantly to ensure that there is no data loss when the application crashes and application recovers gracefully.

- Known issues/bugs in the AUT.

Performance Objectives

Most of the performance tests depend on a set of predefined, documented, and agreed-upon performance objectives. Knowing the objectives from the beginning helps make the testing process more efficient. You can evaluate your application's performance by comparing it with your performance objectives. One should by all means just run Ad-hoc tests randomly without any specific objectives (**Old Principle**: how many bugs were discovered just by executing the test cases).

As a thumb of rule, following are the performance expectations from the Application Under Test

I. **Application Response Time:** This is the most fundamental parameter which ideally is the second nature of the **performance tester.** Application Response time is the amount of time taken to respond to a request. You can measure response time at the server or client as follows**:**

II. **Response Time at the server**. This is the time taken by the server to complete the execution of a request. This does not include the client-to-server latency, which includes additional time for the request and response to cross the network.

III. **Response Time at the client**. The latency measured at the client includes the request queue, plus the time taken by the server to complete the execution of the request and the network latency. You can measure the latency in various ways. Two common approaches are time taken by the first byte to reach the client (*time to first byte*, TTFB), or the time taken by the last byte of the response to reach the client (*time to last byte*, TTLB). Generally, you should test this using various network bandwidths between the client and the server.

By measuring latency, you can gauge whether your application takes too long to respond to client requests.

1. **Application Throughput**

Throughput is the number of requests that can be served by Application Under Test per unit time. It is measured in terms of transactions per second or orders per second. The throughput varies largely due to the type of load applied, volume of load applied etc. The various examples include credit card transactions, the number of concurrent users, volumes of downloads and so on. A larger parameter however in this case also happens to be the network connection. For example, in terms of numbers lets say, there are 1000 users with an average page request data of 5k for every 5 minutes.
The throughput would be = 1,000 x (5x1024x8) / (5 x 60)

2. System Resource Utilization

Typically the following system parameters are measured in performance testing:

I. Memory Utilization
II. CPU Utilization
III. Disk I/O
IV. Network I/O

You can identify the resource cost on a per operation basis. Operations might include browsing a product catalog, adding items to a shopping cart, or placing an order. You can measure resource costs for a given user load, or you can average resource costs when the application is tested using a given *workload profile*.

A workload profile consists of an aggregate mix of users performing various operations. For example, for a load of 200 concurrent users (as defined below), the profile might indicate that 20 percent of users perform order placement, 30 percent add items to a shopping cart, while 50 percent browse the product catalog. This helps you identify and optimize areas that consume an unusually large proportion of server resources and response time.

3. Workload

Simultaneous users have active connections to the same Web site, whereas concurrent users hit the site at exactly the same moment. Concurrent access is likely to occur at infrequent intervals. Your site may have 100 to 150 concurrent users but 1,000 to 1,500 simultaneous users.

When load testing your application, you can simulate simultaneous users by including a random think time in your script such that not all the user threads from the load generator are firing requests at the same moment. This is useful to simulate real world situations.

However, if you want to stress your application, you probably want to use concurrent users. You can simulate concurrent users by removing the think time from your script.

Measuring Numbers in Performance Testing

When you need to measure how many system resources your application consumes, you need to pay particular attention to the following:

- **Memory**. Amount of available memory, virtual memory, and cache utilization.
- **Processor**. Processor utilization, context switches, interrupts and so on.
- **Network**. Percent of the available bandwidth being utilized, network bottlenecks.
- **Disk I/O**. Amount of read and write disk activity. I/O bottlenecks occur if read and write operations begin to queue.

The next sections describe the performance counters that help you measure the preceding metrics.

Memory

To measure memory utilization and the impact of paging, you can use the following counters:

- **Memory\Available Mbytes**

  Cut-Off Threshold: A consistent value of less than 20 to 25 percent of installed RAM is an indication of insufficient memory.

  Importance: indication of the amount of physical memory available to processes running on the computer.

- **Memory\Page Reads/sec**

  Cut-Off Threshold: Sustained values of more than five indicate a large number of page faults for read requests.

  Importance: This counter indicates that the working set of your process is too large for the physical memory and that it is paging to disk. It shows the number of read operations, without regard to the number of pages retrieved in each operation. Higher values indicate a memory bottleneck.

  If a low rate of page-read operations coincides with high values for **Physical Disk\% Disk Time** and **Physical Disk\Avg. Disk Queue Length***,* there could be a disk bottleneck. If an increase in queue length is not accompanied by a decrease in the pages-read rate, a memory shortage exists.

- **Memory\Pages/sec**

  Cut-Off Threshold: Sustained values higher than five indicate a bottleneck.

  Importance: This counter indicates the rate at which pages are read from or written to disk to resolve hard page faults. Multiply the values of the **Physical Disk\Avg. Disk sec/Transfer** and**Memory\Pages/sec** counters. If the product of these counters exceeds 0.1, paging is taking more than 10 percent of disk access time, which indicates that you need more RAM.

- **Memory\Pool Nonpaged Bytes**

  Cut-Off Threshold: Watch the value of Memory\Pool Nonpaged Bytes for an increase of 10 percent or more from its value at system startup.

  Importance: If there is an increase of 10 percent or more from its value at startup, a serious leak is potentially developing.

- **Server\Pool Nonpaged Failures**

  Cut-Off Threshold: Regular nonzero values indicate a bottleneck.

  Importance: This counter indicates the number of times allocations from the nonpaged pool have failed. It indicates that the computer's physical memory is too small. The nonpaged pool contains pages from a process's virtual

address space that are not to be swapped out to the page file on disk, such as a process' kernel object table. The availability of the nonpaged pool determines how many processes, threads, and other such objects can be created. When allocations from the nonpaged pool fail, this can be due to a memory leak in a process, particularly if processor usage has not increased accordingly.

- **Server\Pool Paged Failures**

  Cut-Off Threshold: No specific value.

  Importance: This counter indicates the number of times allocations from the paged pool have failed. This counter indicates that the computer's physical memory or page file is too small.

- **Server\Pool Nonpaged Peak**

  Cut-Off Threshold: No specific value.

  Importance: This is the maximum number of bytes in the nonpaged pool that the server has had in use at any one point. It indicates how much physical memory the computer should have. Because the nonpaged pool must be resident, and because there has to be some memory left over for other operations, you might quadruple it to get the actual physical memory you should have for the system.

- **Memory\Cache Bytes**

  Cut-Off Threshold: No specific value.

  Importance: Monitors the size of cache under different load conditions. This counter displays the size of the static files cache. By default, this counter uses approximately 50 percent of available memory, but decreases if the available memory shrinks, which affects system performance.

- **Memory\Cache Faults/sec**

  Importance: This counter indicates how often the operating system looks for data in the file system cache but fails to find it. This value should be as low as possible. The cache is independent of data location but is heavily dependent on data density within the set of pages. A high rate of cache faults can indicate insufficient memory or could also denote poorly localized data.

- **Cache\MDL Read Hits %**

  Cut-Off Threshold: The higher this value, the better the performance of the file system cache. Values should preferably be as close to 100 percent as possible.

  Importance: This counter provides the percentage of Memory Descriptor List (MDL) Read requests to the file system cache, where the cache returns the object directly rather than requiring a read from the hard disk.

Processor
To measure processor utilization and context switching, you can use the following counters:

- **Processor\% Processor Time**

  Cut-Off Threshold: The general figure for the threshold limit for processors is 85 percent.

  Importance: This counter is the primary indicator of processor activity. High values many not necessarily be bad. However, if the other processor-related counters are increasing linearly such as **% Privileged Time** or **Processor Queue Length**, high CPU utilization may be worth investigating.

- **Processor\% Privileged Time**

  Cut-Off Threshold: A figure that is consistently over 75 percent indicates a bottleneck.

  Importance: This counter indicates the percentage of time a thread runs in privileged mode. When your application calls operating system functions (for example to perform file or network I/O or to allocate memory), these operating system functions are executed in privileged mode.

- **Processor\% Interrupt Time**

  Cut-Off Threshold: Depends on processor.

  Importance: This counter indicates the percentage of time the processor spends receiving and servicing hardware interrupts. This value is an indirect indicator of the activity of devices that generate interrupts, such as network adapters. A dramatic increase in this counter indicates potential hardware problems.

- **System\Processor Queue Length**

  Cut-Off Threshold: An average value consistently higher than 2 indicates a bottleneck.

  Importance: If there are more tasks ready to run than there are processors, threads queue up. The processor queue is the collection of threads that are ready but not able to be executed by the processor because another active thread is currently executing. A sustained or recurring queue of more than two threads is a clear indication of a processor bottleneck. You may get more throughput by reducing parallelism in those cases.

  You can use this counter in conjunction with the **Processor\% Processor Time** counter to determine if your application can benefit from more CPUs. There is a single queue for processor time, even on multiprocessor computers. Therefore, in a multiprocessor computer, divide the Processor Queue Length (PQL) value by the number of processors servicing the workload.

If the CPU is very busy (90 percent and higher utilization) and the PQL average is consistently higher than 2 per processor, you may have a processor bottleneck that could benefit from additional CPUs. Or, you could reduce the number of threads and queue more at the application level. This will cause less context switching, and less context switching is good for reducing CPU load. The common reason for a PQL of 2 or higher with low CPU utilization is that requests for processor time arrive randomly and threads demand irregular amounts of time from the processor. This means that the processor is not a bottleneck but that it is your threading logic that needs to be improved.

- **System\Context Switches/sec**

  Cut-Off Threshold: As a general rule, context switching rates of less than 5,000 per second per processor are not worth worrying about. If context switching rates exceed 15,000 per second per processor, then there is a constraint.

  Importance: Context switching happens when a higher priority thread preempts a lower priority thread that is currently running or when a high priority thread blocks. High levels of context switching can occur when many threads share the same priority level. This often indicates that there are too many threads competing for the processors on the system. If you do not see much processor utilization and you see very low levels of context switching, it could indicate that threads are blocked.

Disk I/O

To measure disk I/O activity, you can use the following counters:

- **PhysicalDisk\Avg. Disk Queue Length**

  Cut-Off Threshold: Should not be higher than the number of spindles plus two.

  Importance: This counter indicates the average number of both read and writes requests that were queued for the selected disk during the sample interval.

- **PhysicalDisk\Avg. Disk Read Queue Length**

  Cut-Off Threshold: Should be less than two.

  Importance: This counter indicates the average number of read requests that were queued for the selected disk during the sample interval.

- **PhysicalDisk\Avg. Disk Write Queue Length**

  Cut-Off Threshold: Should be less than two.

  Importance: This counter indicates the average number of write requests that were queued for the selected disk during the sample interval.

- **PhysicalDisk\Avg. Disk sec/Read**

  Cut-Off Threshold: No specific value.

  Importance: This counter indicates the average time, in seconds, of a read of data from the disk.

- **PhysicalDisk\Avg. Disk sec/Transfer**

  Cut-Off Threshold: Should not be more than 18 milliseconds.

  Importance: This counter indicates the time, in seconds, of the average disk transfer. This may indicate a large amount of disk fragmentation, slow disks, or disk failures. Multiply the values of the **Physical Disk\Avg. Disk sec/Transfer** and **Memory\Pages/sec** counters. If the product of these counters exceeds 0.1, paging is taking more than 10 percent of disk access time, so you need more RAM.

- **PhysicalDisk\Disk Writes/sec**

  Importance: This counter indicates the rate of write operations on the disk.

Network I/O

To measure network I/O, you can use the following counters:

- **Network Interface\Bytes Total/sec**

  Cut-Off Threshold: Sustained values of more than 80 percent of network bandwidth.

  Importance: This counter indicates the rate at which bytes are sent and received over each network adapter. This counter helps you know whether the traffic at your network adapter is saturated and if you need to add another network adapter. How quickly you can identify a problem depends on the type of network you have as well as whether you share bandwidth with other applications.

- **Network Interface\Bytes Received/sec**

  Importance: This counter indicates the rate at which bytes are received over each network adapter. You can calculate the rate of incoming data as a part of total bandwidth. This will help you know that you need to optimize on the incoming data from the client or that you need to add another network adapter to handle the incoming traffic.

- **Network Interface\Bytes Sent/sec**

  Importance: This counter indicates the rate at which bytes are sent over each network adapter. You can calculate the rate of incoming data as a part of total bandwidth. This will help you know that you need to optimize on the data being sent to the client or you need to add another network adapter to handle the outbound traffic.

- **Server\Bytes Total/sec**

  Cut-Off Threshold: Value should not be more than 50 percent of network capacity.

  Importance: This counter indicates the number of bytes sent and received over the network. Higher values indicate network bandwidth as the bottleneck. If the sum of **Bytes Total/sec** for all servers is roughly equal to the maximum transfer rates of your network, you may need to segment the network.

- **Protocol related counters:**
  - ***Protocol_Object*\Segments Received/sec**
  - **Protocol_Object\Segments Sent/sec**

  Protocol_Object can be TCP, UDP, NetBEUI, NWLink IPX, NWLink NetBIOS, NWLink SPX, or other protocol layer performance objects.

  Importance: Protocol-related counters help to narrow down the traffic to various protocols because you might be using one or more protocols in your network. You may want to identify which protocol is consuming the network bandwidth disproportionately.

- **Processor\% Interrupt Time**

  Importance: This counter indicates the percentage of time the processor spends receiving and servicing hardware interrupts. This value is an indirect indicator of the activity of devices that generate interrupts, such as network adapters.

Success Mantras for Success in Performance Testing:

- Clear the application and database logs after each performance test run. Excessively large log files may artificially skew the performance results.
- Identify the correct server software and hardware to mirror your production environment.
- Use a single graphical user interface (GUI) client to capture end-user response time while a load is generated on the system. You may need to generate load by using different client computers, but to make sense of client-side data, such as response time or requests per second, you should consolidate data at a single client and generate results based on the average values.
- Include a buffer time between the incremental increases of users during a load test.
- Use different data parameters for each simulated user to create a more realistic load simulation.
- Monitor all computers involved in the test, including the client that generates the load. This is important because you should not overly stress the client.
- Prioritize your scenarios according to critical functionality and high-volume transactions.

- Use a zero think time if you need to fire concurrent requests,. This can help you identify bottleneck issues.
- Stress test critical components of the system to assess their independent thresholds.

Best Practices for Performance Testing

- Do not allow the test system resources to cross resource threshold limits by a significant margin during load testing, because this distorts the data in your results.
- Do not run tests in live production environments that have other network traffic. Use an isolated test environment that is representative of the actual production environment.
- Do not try to break the system during a load test. The intent of the load test is not to break the system. The intent is to observe performance under expected usage conditions. You can stress test to determine the most likely modes of failure so they can be addressed or mitigated.
- Do not place too much stress on the client test computers.