Enterprise Applications in the Cloud:

Non-virtualized Deployment

Leonid Grinshpan, Oracle Corporation (www.oracle.com)

Subject

The cloud is a platform devised to support a number of concurrently working applications that share the cloud's resources; being a platform of common use, the cloud features complex interdependencies among hosted applications, as well as among applications and the underlying hardware platform.

Enterprise Applications (EAs) can be deployed in the cloud in two ways:

1. **Non-virtualized setup** hosts on the same physical servers different EAs without logical borders between them (no partitions, virtual machines or similar technologies in place).

2. **Virtualized arrangement** separates EAs logically from each other by employing the above-mentioned techniques.

Both deployment models have advantages and disadvantages. The performance penalty introduced by virtualization (which we will analyze in the next article) prevents many EA vendors from recommending EA deployment in virtual environments. As an example, here is a policy of Thomson Reuters Elite business (*http://www.elite.com/virtualization_servers/*):

Elite generally recommends against using virtualization environments (e.g., Virtual Machines from VMware or Microsoft Virtual Server) for primary production servers hosting Elite products. Elite makes no performance warranties in relation to Elite applications hosted on Virtual Machines.

In non-virtualized clouds an allocation of resources for different EAs is carried out by operating systems that provision software processes representing EAs. This environment makes all processing power of the physical servers available to the applications. Furthermore, it enables collection of reliable performance metrics by

directly monitoring the server's counters. It is quite possible that for those reasons, Google applications are not embedded into virtual environments. Another example of a non-virtualized cloud is the popular project management and collaboration tool Basecamp [*Is Virtualization a Cloud Prerequisite?* http://gigaom.com/2009/08/30/is-virtualization-a-cloud-prerequisite/]

One shortcoming of non-virtualized cloud is obvious: Instability of any EA resulting in hardware downtime affects availability of all EAs. But what happens much more often is that an EA might suffer performance degradation in response to the changes in workload and service demand experienced by any other EA. We analyze that phenomenon simulating happenings in the cloud using queuing models of EAs; methodological foundation for EA performance analysis based on queuing models can be found in the author's book [*Leonid Grinshpan. Solving Enterprise Application Performance Puzzles: Queuing Models to the Rescue, Willey-IEEE Press; available in bookstores and from Web booksellers from January 2012*].

Performance Impact of Workload Fluctuations

The queuing model on Figure 1 represents simplified three-tiered Cloud with Web, Application, and Database servers. The cloud hosts three EAs (App A, App B, App C) serving three user groups, one group per EA. Each server corresponds to the model's node with a number of processing units equal to the number of CPUs in a server. The users of each EA, as well as the network, are modeled by dedicated nodes. All servers are physical ones without any partitioning among applications. Web and Application servers have 8 CPUs each; Database server has 16 CPUs.

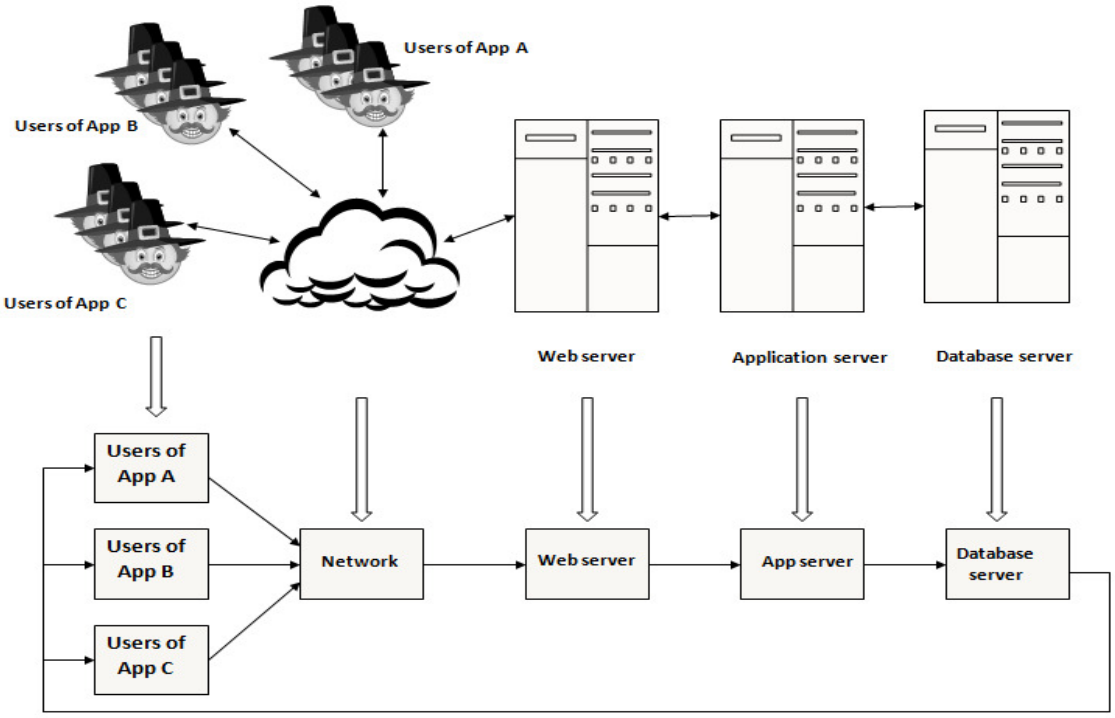

Figure 1 Model 1 of the cloud hosting three enterprise applications

The models in this article were analyzed using TeamQuest solver [*http://teamquest.com/products/model/index.htm*]. Here is a description of Model 1 components in TeamQuest terms:

Active Resource	Equipment Name	Equipmer Type	Path	Active Resource Attributes
Web server	Intel XeonMP 2.8GHz/0.5MB	CPU		Servers = 8; Hyper-threading = On; Type = THREAD; Discipline = PPRI
Application server	Intel XeonMP 2.8GHz/0.5MB	CPU		Servers = 8; Hyper-threading = On; Type = THREAD; Discipline = PPRI
Database server	Intel XeonMP 2.8GHz/0.5MB	CPU		Servers = 16; Hyper-threading = On; Speed Factor = 0.77932; Type = THREAD; Discipline = PPRI
Users of App A	THINK Queue			Servers = 1; Discipline = IS
Users of App B	THINK Queue			Servers = 1; Discipline = IS
Users of App C	THINK Queue			Servers = 1; Discipline = IS
Network	DELAY Queue			Servers = 1; Discipline = IS

Workload 1 for Model 1 is characterized in Table 1. For each application it is represented by transactions identified by application name. A user initiates transaction a number of times indicated in column *"Number of transaction executions per user per hour."* We analyze the model for 200, 400, 600, and 800 users.

Table 1

Workload 1 for Model 1

Transaction name	Number of users					Number of transaction executions per user per hour
	Total 3	Total 200	Total 400	Total 600	Total 800	
App A transaction	1	100	200	300	400	10
App B transaction	1	50	100	150	200	20
App C transaction	1	50	100	150	200	5

To solve the model we have to specify the profile of each transaction (Table 2). The Transaction Profile is a set of time intervals (service demands) a transaction has spent in all processing units it has visited while served by application.

Table 2

Transaction Profiles (seconds)

	Time in Network node	Time in Web server node	Time in App server node	Time in Database server node
App A transaction	0.001	0.2	1.0	5.0
App B transaction	0.0015	0.1	0.5	2.5
App C transaction	0.003	0.2	5.0	5.0

Model 1 estimates that transaction times for **all** applications will start increasing when the number of users exceeds 400 (Figure 2).

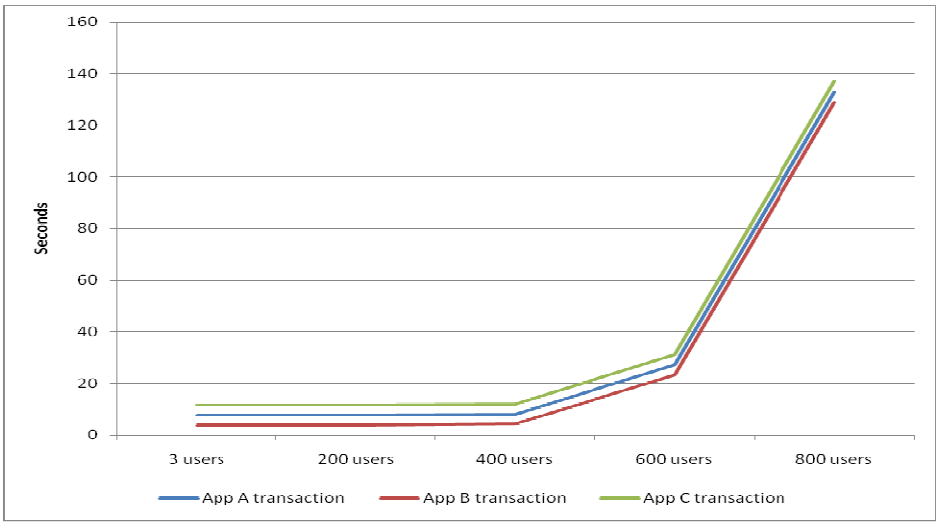

Figure 2 Transaction response times for three applications

To find a reason for transaction time degradation, let's look at utilization of the cloud's servers (Figure 3).

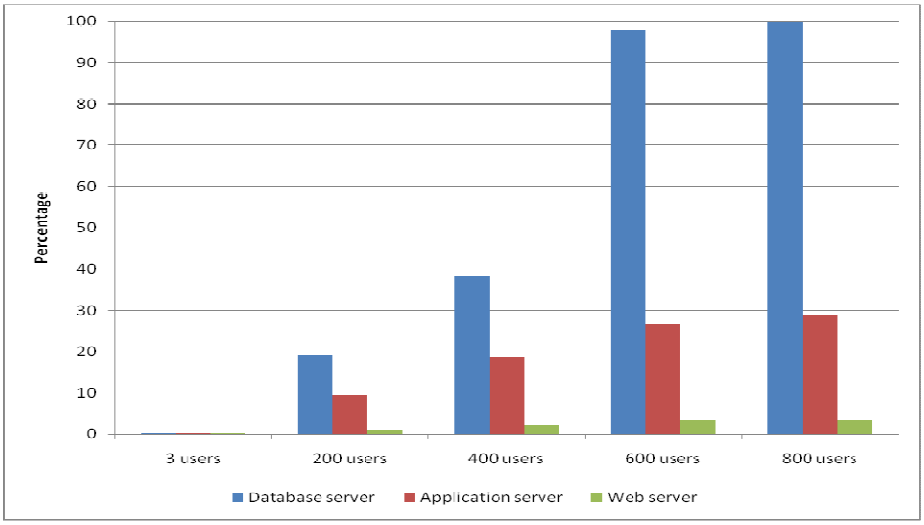

Figure 3 Utilization of the cloud's servers

When the number of users is close to 600, utilization of Database server exceeds 85% and causes noticeable increase in transaction times for all applications. Any further growth in the number of users maxes out the Database server and results in exponential explosion of transaction time. Non-virtualized cloud does not discriminate—it punishes all applications by increasing their transaction times.

Model 1 helps to determine the contribution of each application into Database server utilization (Figure 4).

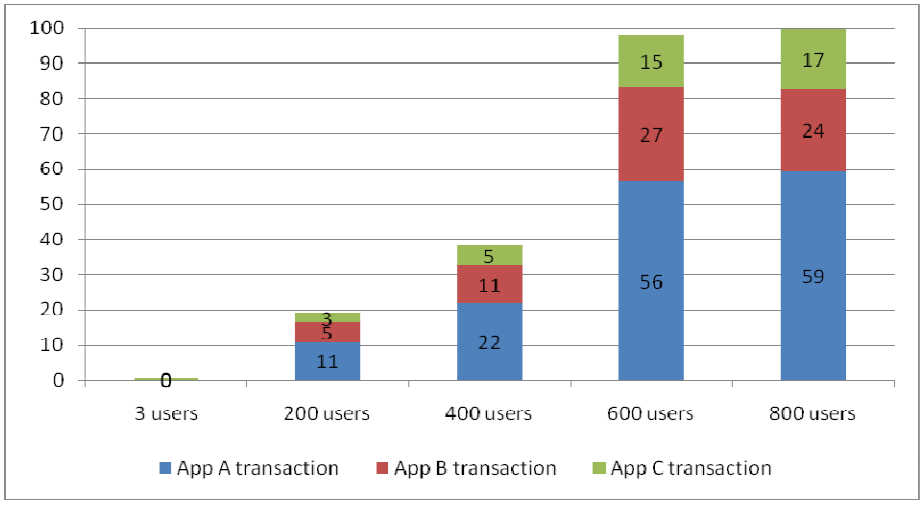

Figure 4 Breakdown of utilization of Database server by App A, B, and C (percentage)

Per Figure 4 the largest "consumer" of Database server capacity is App A. We will analyze how the decrease of App A number of users affects all other applications. Workload 2 in Table 3 shows that when we keep the number of users of App A limited to 100, for two other applications they remain the same as was specified in Table 1.

Table 3

Workload 2 for Model 1

Transaction name	Number of users					Number of transaction executions per user per hour
	Total 3	Total 200	Total 300	Total 400	Total 500	
App A transaction	1	100	100	100	100	10
App B transaction	1	50	100	150	200	20
App C transaction	1	50	100	150	200	5

Transaction times and utilization of the cloud's server for Workload 2 are pictured on Figures 5 and 6; charts suggest that **all** applications now provide acceptable services for their users.

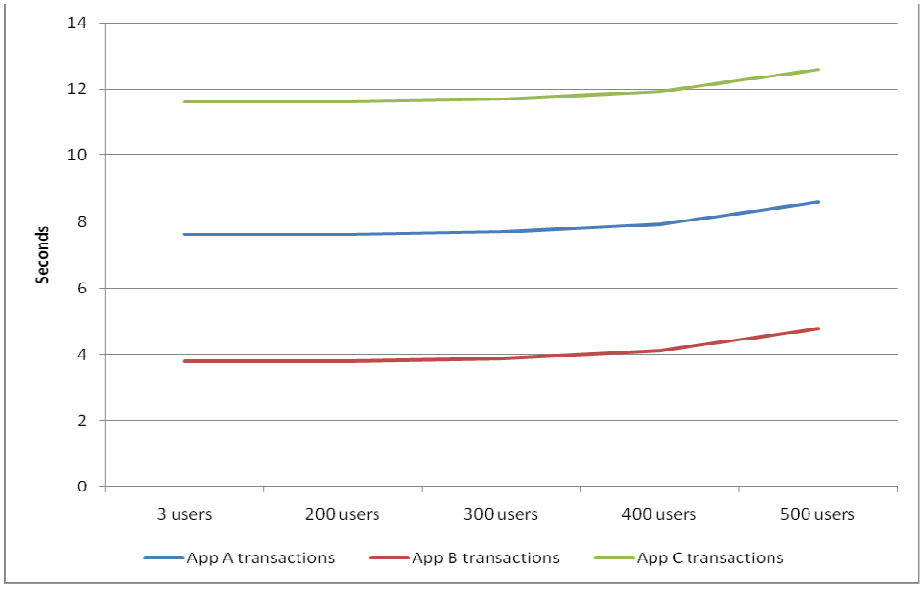

Figure 5 Transaction response times
for three applications for Workload 2

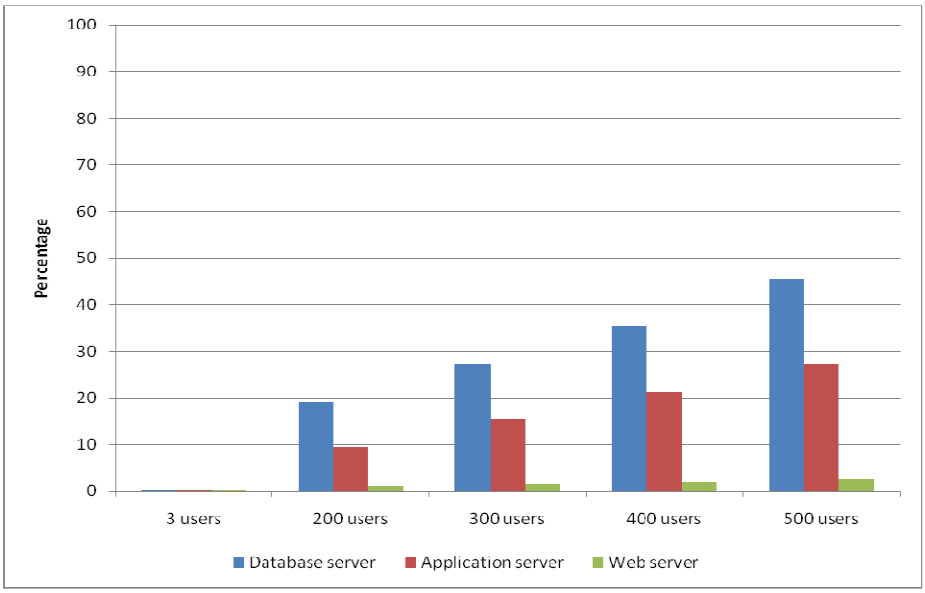

Figure 6 Utilization of the cloud's servers for Workload 2

Performance Impact of Service Demand Fluctuations

The transaction's service demand for a particular hardware component characterizes the time this component has to spend to process the transaction. A transaction can be in two states: either waiting for resource or using resource. A service demand for a particular resource means that a time transaction is using this resource. Service demand in general depends on two parameters:

- Resource's processing speed
- Volume of data to be processed by resource

The first parameter characterizes hardware resource (for example, disk transfer rate is 1000 Mbit/second). For a resource such as CPU, a processing speed depends on clock speed (for example, 3 GHz), as well as on software algorithms. Resource processing speed is a constant for given hardware component and software release.

To the contrary, the second parameter's prevailing trend is an increase in data volume, because over time, business accumulates more data (for example, more sales executed in June than in January). If EA transaction represents a financial report on sales volume, then generation of the report's data for June will take more Database server time than generation of the report's data for January. We analyze an impact of service demand fluctuations on EA performance using Model 2.

Model 2 has the same topology and the same workload as Model 1 (Figure 1 and Table 1); the difference is that we analyze Model 2 for three values of service demands from App C transaction for processing in Database (5, 8, and 10 seconds; see Table 4).

Table 4

Transaction Profiles (seconds) with Different Service Demands

	Time in Network node	Time in Web server node	Time in App server node	Time in Database server node
App A transaction	0.001	0.2	1.0	5.0
App B transaction	0.0015	0.1	0.5	2.5
App C transaction	0.003	0.2	5.0	Model analyzed for 5.0, 8.0, 10.0 seconds

Model 2 predicts that times of **all** application transactions degrade faster with an increase of service demand from **one** application (Figure 7).

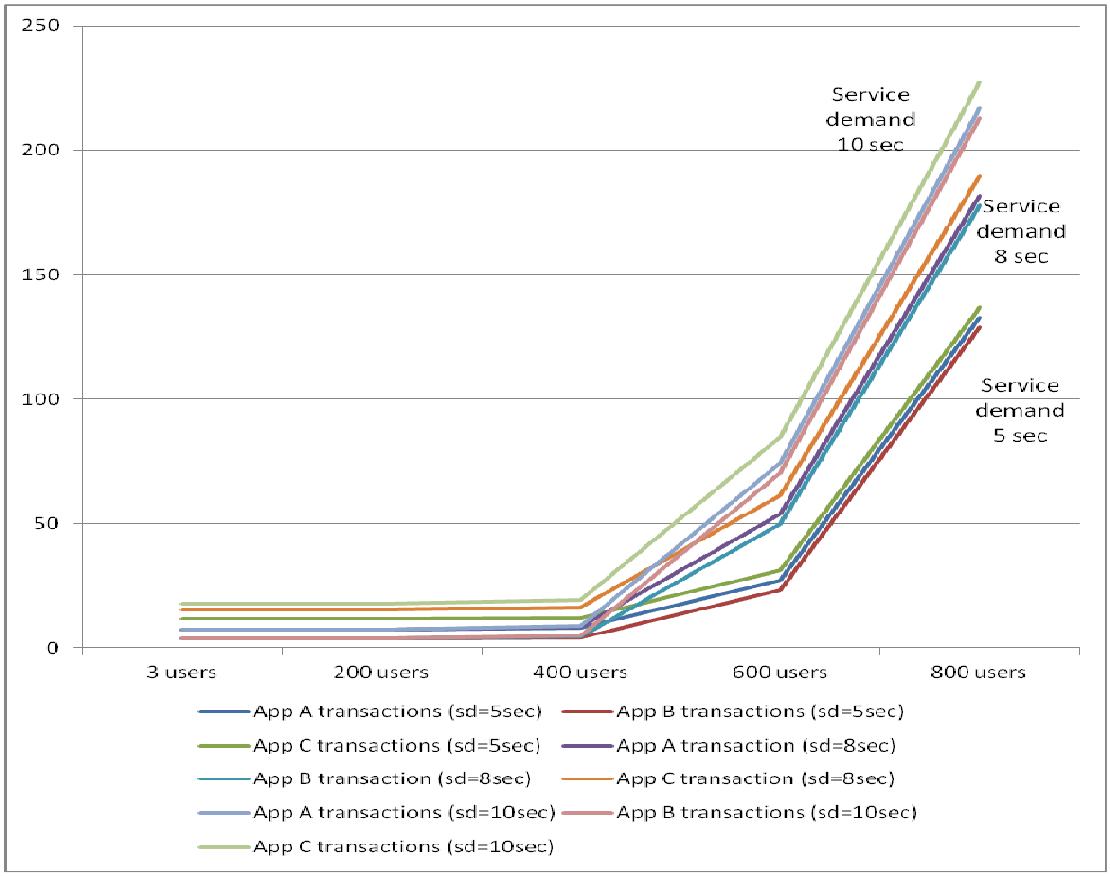

Figure 7 Transaction response times for different service demands (sd in the legend stands for service demand)

Monitoring Resource Consumptions by Applications

In a non-virtualized environment, each application is "materialized" by the operating system as the number associated with its processes. To find out resource consumptions by application, we have to monitor its processes. Figure 8 pictures Windows Task Manager reporting performance data for two processes belonging to the same application: Process *beasvc.exe* is an application server and process *oracle.exe* is a database. *CPU* column shows CPU usage as a percentage of time that process used the CPU since last update; *Mem Usage* column delivers a size of current working set of a process in kilobytes; *Threads* column counts a number of threads running in a process; remaining columns relate to I/O system.

More detailed information on processes can be collected using Windows Performance Monitor. For example, Performance Monitor reports for each process a speed of I/O operations, number of I/O operations per second, memory paging information, thread count and state of each thread, usage of virtual memory, etc. Performance Monitor can save collected data into log files; analyses of such log files discover trends in data collected in performance counters.

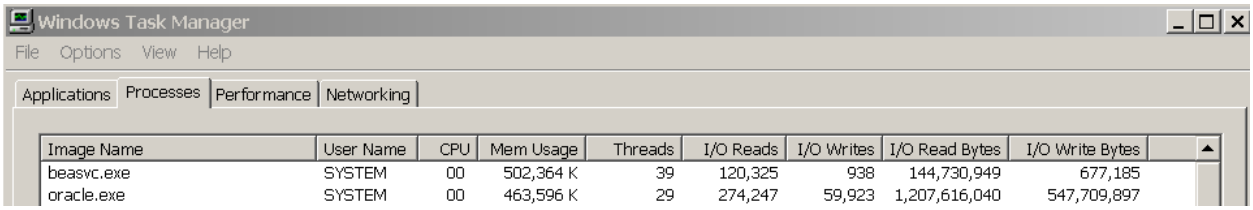

Figure 8 Performance data reported by Windows Task Manager for different processes

In UNIX environment process level data can be examined, for example, by using *prstat –a* command (Figure 9): Process ID is reported in first column PID, columns *SIZE* and *RSS* specify memory usage and column *CPU* delivers CPU utilization. This command helps to associate process name (last column) and process ID.

```
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
245393 pbfhypts 4496K 4160K cpu63   54    0   0:15:04 0.2% prstat/1
  2105 root       27M   22M sleep   59    0  91:45:30 0.0% adclient/8
233504 pbfhypts 1573M 1527M sleep   59    0   0:16:42 0.0% ESSSVR/562
232052 pbfhypts  453M  375M sleep   58    0   0:06:00 0.0% ESSBASE/162
419077 oracle   3277M 3262M sleep   59    4  16:20:11 0.0% oracle/1
246026 oracle   3273M 3262M sleep   46    0   0:00:02 0.0% oracle/1
  2037 root       13M 8448K sleep   59    0   3:57:49 0.0% automountd/3
  2917 root       33M   19M sleep   59    0  47:45:00 0.0% sysedge.sol210-/1
```

Figure 9 Process information in UNIX environment
as reported by *prstat –a* command

If we have to log data on process resource utilization over some period of time, we can use the following command for a process named <process name>:

while true; do ps -eo vsz,rss,pcpu,comm| grep <process name> >> log.txt; sleep 5; done

This command collects the counters every five seconds:

vsz - Total size of the process in virtual memory, in kilobytes.

rss - Resident set size of the process, in kilobytes.

pcpu - Percentage of CPU utilization.

We mentioned a few basic ways of providing insight into resource consumption by each hosted application; operating systems and third party monitoring products offer a broad spectrum of means to collect process-level data to make confident decisions on cloud capacity management.

Take Away from the Article

1. In non-virtualized cloud performance, bottlenecks created by a shortage of hardware resources affect all applications by increasing their transaction times.
2. Cloud-wide bottleneck can be caused by increase in workload of any hosted application, as well as by fluctuations in its service demand while processing large or complex data.
3. A contribution of each application to resources utilization can be found by monitoring software processes created by the operating system for a particular application. Queuing models provide estimates of such contribution.

About the Author

During the last fifteen years as an Oracle consultant, the author was engaged in hands-on performance tuning and sizing of enterprise applications for various corporations (Dell, Citibank, Verizon, Clorox, Bank of America, AT&T, Best Buy, Aetna, Halliburton, Pfizer, Astra Zeneca, Starbucks, etc).