TeamQuest

How to Manage IT Resource Consumption

At an Application Level with TeamQuest Workloads

In this paper, John Miecielica of Metavante, provides a step-by-step example showing how he uses TeamQuest software to analyze IT resource consumption at an application level. This key capability is especially important in today's environments where multiple applications run on a server or multiple servers might be required to implement one application. It allows Metavante to solve complex performance and capacity management issues crucial to keeping their business running smoothly and efficiently.

About the Author

John Miecielica is currently managing the capacity planning, monitoring and performance teams at a large financial group in the Milwaukee, Wisconsin area. The current environment consists of a large IBM mainframe complex as well as more than 1,200+ UNIX, Windows and Linux Servers. John also provides key input into the overall systems architecture.

Previously, John created and managed the open systems disciplines at his current company. He has more than 20 years experience in technical support and management on levels of systems ranging from large IBM mainframe complexes, many varieties of UNIX implementations down to desktop and server environments running Windows OS. John holds a BA in Mathematics and Computer Science, and an MS in Computer Science.

Managing a Consolidated Environment: Issues and Challenges

Metavante has a server farm with more than 1500 Windows, Linux and Unix servers supporting a vast array of Java, .Net and legacy applications. When first hosting applications on Windows and Unix servers in the 1990s, the standard mode of operation was one application per server. In this current era of virtualization, grid computing, and server reduction initiatives, the situation is now more complex. One physical server might host multiple applications, or one application could span many servers.

One of the core activities for any capacity planning and performance management team is to track physical and logical resource consumption. Physical resources include CPU, memory, and I/O; logical resources include number of application threads and number of available database connections. In a shared resource environment, it becomes vitally important to correlate the resources consumed with the application or system process consuming them for a variety of reasons:

- Tracking resource consumption over time provides input to capacity planning activities at an application level.

  • Tracking resource consumption is important not only after an application is in production, but also during the pre-deployment testing phase to ensure that it meets design specifications.

  • Correlating resource utilization with other metrics on a given server allows us to troubleshoot performance issues. We can identify the busiest I/O device when an application stops consuming user CPU for a short period of time, for example

Tracking Resource Usage with Workloads

TeamQuest workloads allow me to correlate the system resources consumed with the applications, departments, or processes consuming them.

A *workload* is a logical classification of work on a computer system defined around a common characteristic, such as by application or department.

A *workload* set is several workloads that together represent the resources used by all activities of the systems being analyzed. Each workload set defines a different way of looking at performance.

Workloads and workload sets allow me the flexibility to view resource consumption in a variety of ways. I can track resource consumption by application, for example, to measure how well we're meeting service levels and to uncover potential performance or capacity concerns before users are affected.

Defining Workloads: A Drill Down Approach

Due to the nature and complexity of the environment being managed, a drill down approach to defining workloads is appropriate. For example, a high level view of a given environment would consist of the following defined workloads:

- Operating Environment
- Application Component 1
- Application Component 2
- Application Component 3

However, if we encounter either a capacity concern or a performance issue, we may need to quickly re-characterize work into the following categories:

- Operating Environment
- Operating System
- Disk and Backup
- Network
- Security
- Application Component 1
- Application Component 2
- Application Component 3

Or perhaps

- Operating Environment
- Application Component 1
- Application Component 2
- Application Component 2a
- Application Component 2b
- Application Component 2c
- Application Component 3

I set out to see if we could meet these business objectives with TeamQuest Manager, TeamQuest View and TeamQuest Model. Here is what I found.

Finding Answers with TeamQuest Manager and TeamQuest View

Step 1: Set-up

My research indicates that we need access to the field fullcmd. The default is for TeamQuest to collect the first 72 characters of this field. Some of our applications have a fullcmd field of more than 1000 characters, so we modify the collection agent to collect just the last 128 characters of fullcmd. This is done via the TeamQuest administrative interface.

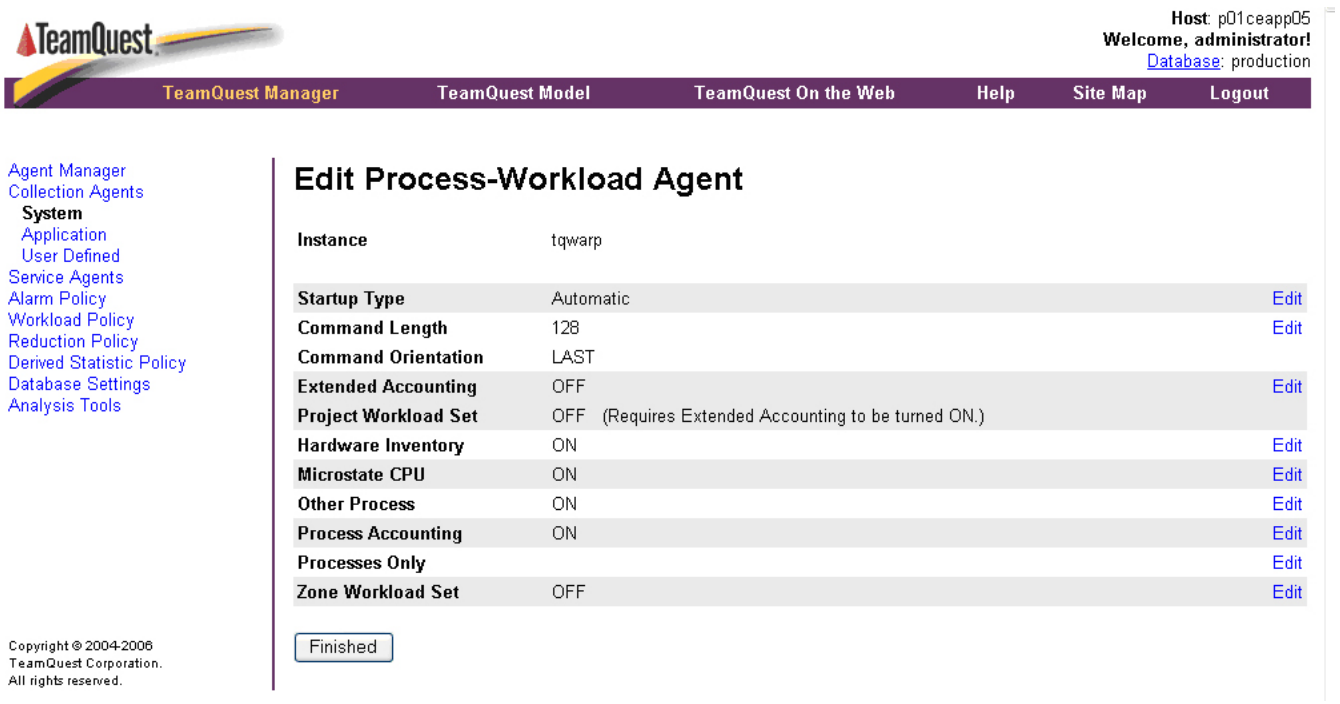


Figure 1
TeamQuest Performance Software administrative interface allows you to make changes to information collected by the Process-Workload Agent

Step 2: Understanding our environment and practicing workload definitions

The next step is to get a list of the most active processes and start classifying them into the categories that we want to monitor. One of the easiest ways to do this is with TeamQuest View and the process table.

Start with a CPU chart and then drill down into the process table.

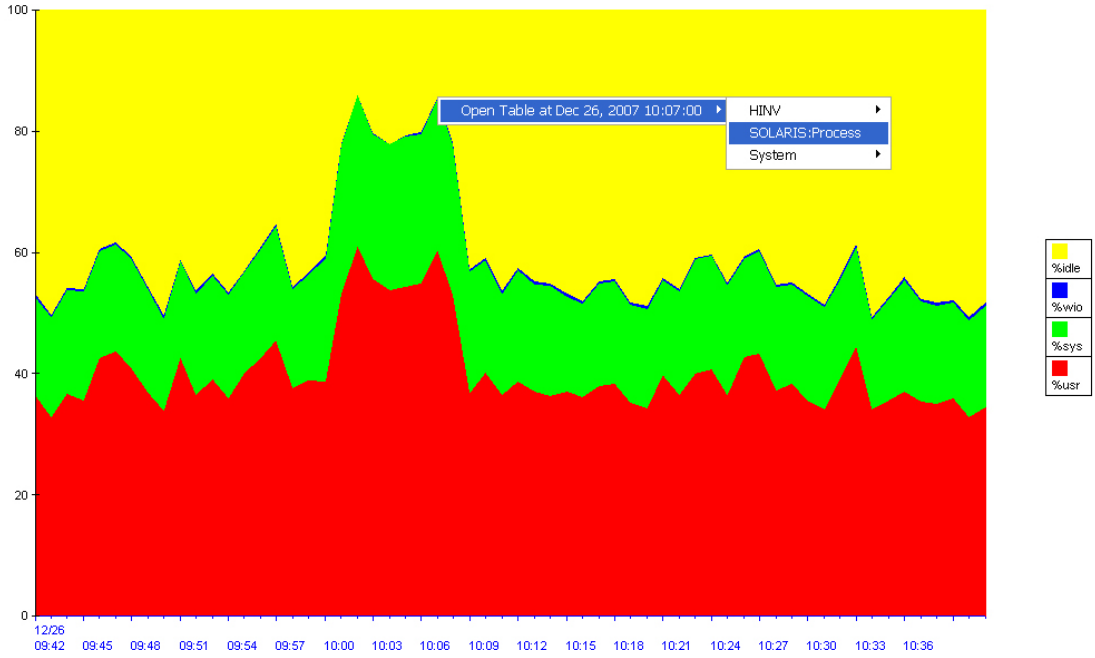


Figure 2
You can right-click on any area of a chart to drill down to the process table for more detailed information.

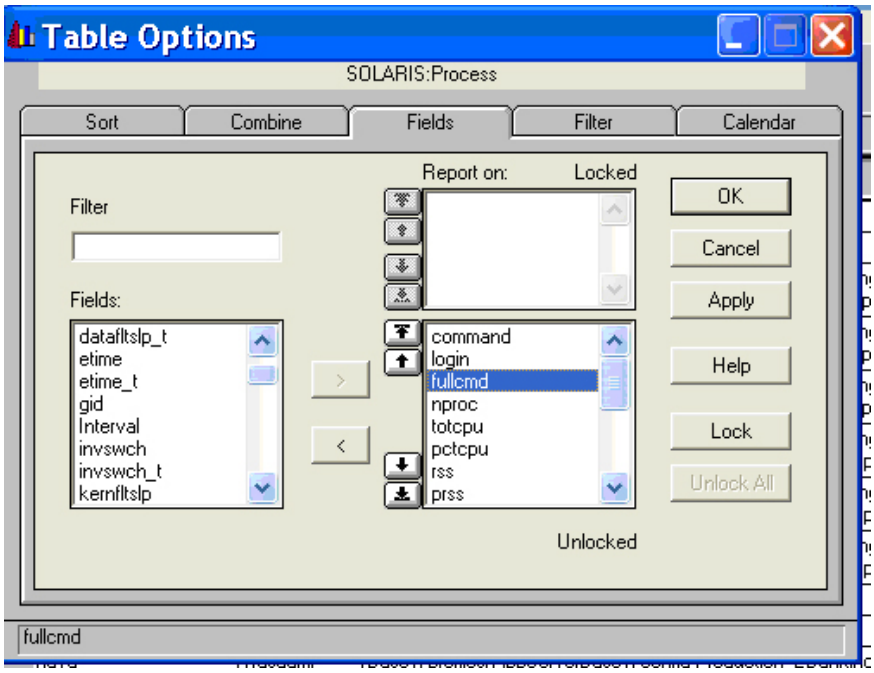


Figure 3
TeamQuest View's Table Options dialog box makes it easy to add, delete and move columns in your progress report.

Once in the process table, turn on the field "fullcmd" and organize it so it appears toward the left side of the display. This is especially important in the Unix environment where the differentiators between processes are only found in the fullcmd field. To accomplish this, I simply click on options and use the standard GUI to change and organize the display fields.

The resultant process tables looks like this.

Row	command	login	fullcmd	nproc
Summary	<Multi>	<Multi>	<Multi>	13
1	<Other>	<N/A>	<Other>	
2	java	wasadmin	rBase1/profiles/AppServerBase2/config Production_EBanking_Cell p01ceapp05_base1_node2 CEBNG_E_Prd_P01ceapp05_ApplicationServer_	
3	java	wasadmin	rBase1/profiles/AppServerBase2/config Production_EBanking_Cell p01ceapp05_base1_node2 CEBNG_E_Prd_P01ceapp05_ApplicationServer_	
4	java	wasadmin	rBase1/profiles/AppServerBase2/config Production_EBanking_Cell p01ceapp05_base1_node2 CEBNG_E_Prd_P01ceapp05_ApplicationServer_	
5	java	wasadmin	rBase1/profiles/AppServerBase1/config Production_EBanking_Cell p01ceapp05_base1_node1 CEBNG_D_Prd_P01ceapp05_ApplicationServer_	
6	java	wasadmin	rBase1/profiles/AppServerBase1/config Production_EBanking_Cell p01ceapp05_base1_node1 CEBNG_D_Prd_P01ceapp05_ApplicationServer_	
7	java	wasadmin	rBase1/profiles/AppServerBase1/config Production_EBanking_Cell p01ceapp05_base1_node1 CEBNG_D_Prd_P01ceapp05_ApplicationServer_	
8	fsflush	root	fsflush	
9	agent	root	/opt/Entercept/agent -d	
10	java	wasadmin	Base1/profiles/AppServerBase1/config Production_EBanking_Cell p01ceapp05_base1_node1 CEBNGADM_D_Prd_P01ceapp05_ApplicationServe	

Figure 4
Process table reports show information about the processes active within a particular time interval.

The report shows several of the top CPU-utilizing processes are Java processes which all look similar to each other. However, there is one differentiator. In the middle of the fullcmd string, note that some are CEBNG_D where as others are CEBNG_E. I asked our application team contact about this and he explained that the application is organized into "containers" and that the containers on this server are labeled "D" and "E." I further noted that it is important when troubleshooting issues to distinguish between the containers. We discussed that this differentiation would be very important in capacity planning, as well.

Before defining workloads, I find it best to test my selection criteria again using the process table in TeamQuest View. From within the process table view, I select "Options" and then "Filter."

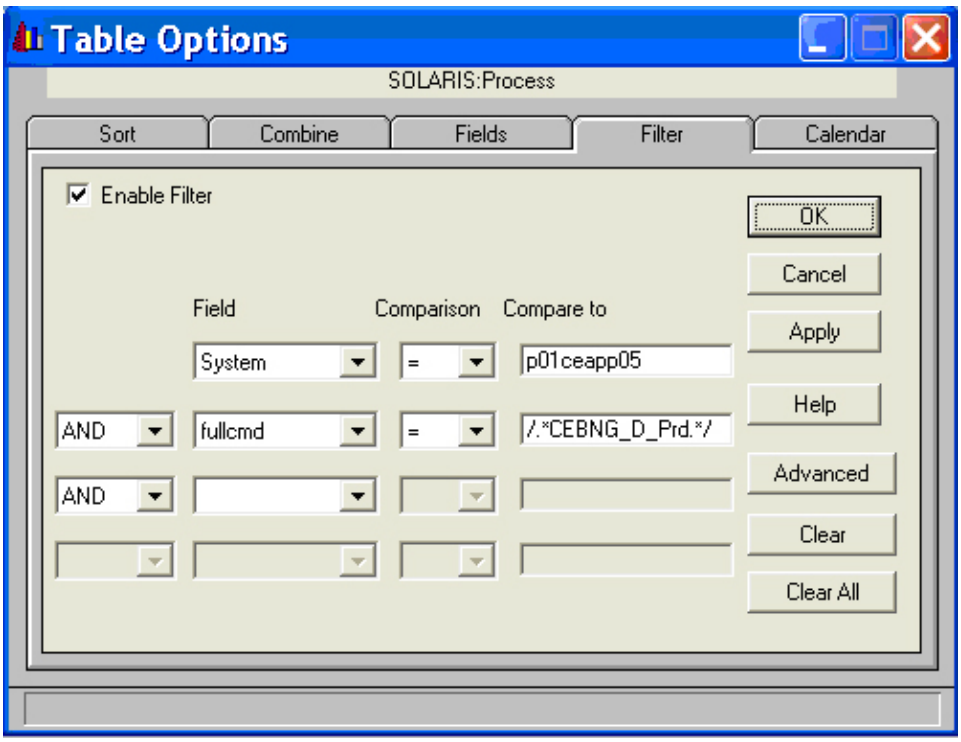


Figure 5
Processes can be easily filtered as a tool for evaluating a potential workload definition.

The resultant process table view confirms that we capture just the processes we want.

File Edit View Format Data Tables Window Help

Lock Unlock Filter Combine Set Options Stop

Dec 27, 2007 14:52:00 - 14:53:00

Row	command	login	nproc	totcpu	pctcpu	rss	prss	majflt	pid	pri	lioch	avgmem_t	ttyname	group
Summary	<Multi>	<Multi>	129	02:10.06	54.13	10109424	<N/A>	4	<Multi>	53	79746.7	12732832	<Multi>	<Multi>
1	java	wasadmin	1	00:14.21	5.91	1233984	<N/A>	0	16778	71	9735.9	1434312	?	wasadm
2	java	wasadmin	1	00:12.85	5.35	1219752	<N/A>	1	17484	71	8553.3	1409456	?	wasadm
3	java	wasadmin	1	00:11.96	4.98	1213584	<N/A>	1	17159	71	6849.0	1409528	?	wasadm

Figure 6
This process report shows only those processes that meet the filter criteria.

Step 3: Set up workloads

After I've defined all workloads inside TeamQuest View, I find it very trivial to set these workloads up. To start, I go to the TeamQuest administrative interface on the target server and select "Workload Policy."

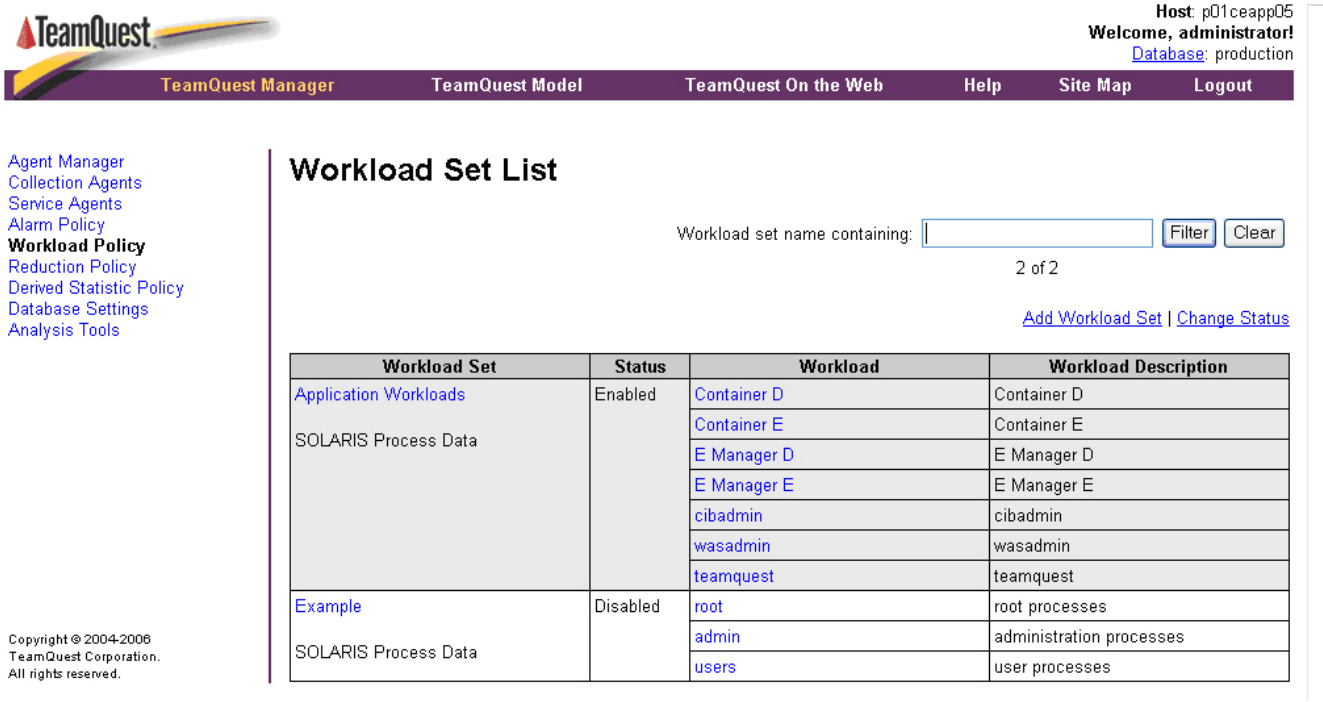


Figure 7
Workload sets are easily created or edited via the TeamQuest Performance Software administrative interface.

I can select standard features such as adding a new workload set, or I can click on an existing workload to modify it.

Once in the workload set, I can add various workload definitions or modify existing ones.

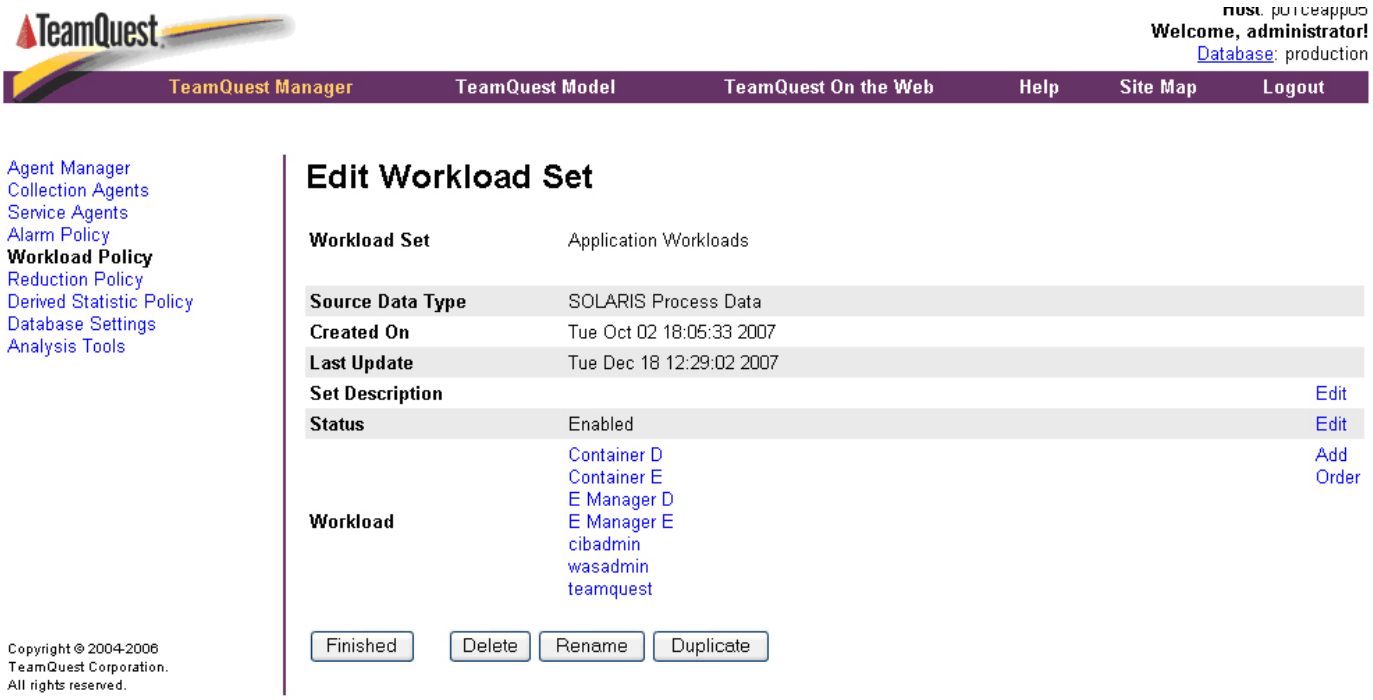


Figure 8
You can edit a workload by clicking its name.

Below is an example of a workload definition:

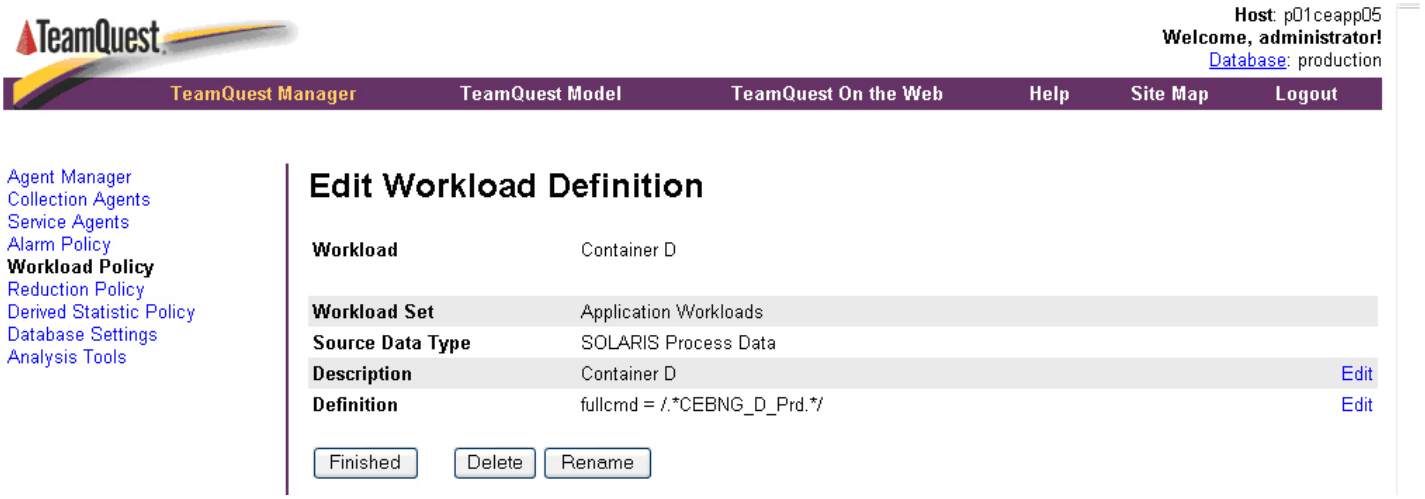


Figure 9
This screen shows a typical workload definition.

Note that the expression syntax used in the “Definition” field is identical to what we tested in TeamQuest View. Once I have all the workloads defined, I simply activate my changes and TeamQuest Manager applies the new workload definitions to newly collected performance data.

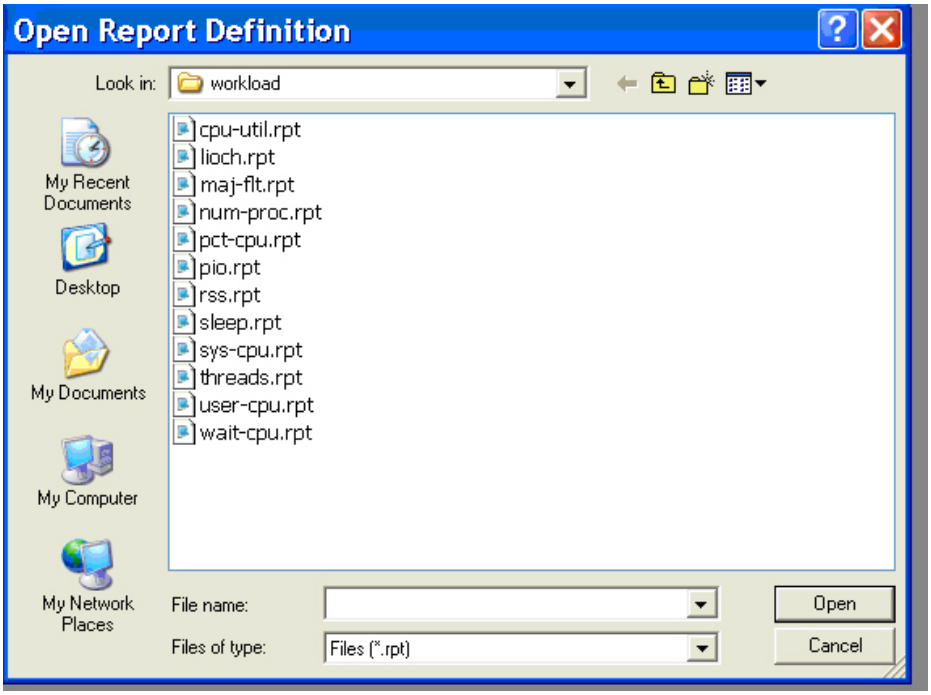


Figure 10
TeamQuest software provides a variety of predefined reports for analyzing workload activity.

Step 4: Using the workload definitions in TeamQuest View

Now I can open predefined workload reports in TeamQuest View.

The report shows the CPU utilization broken down into the workloads we defined.

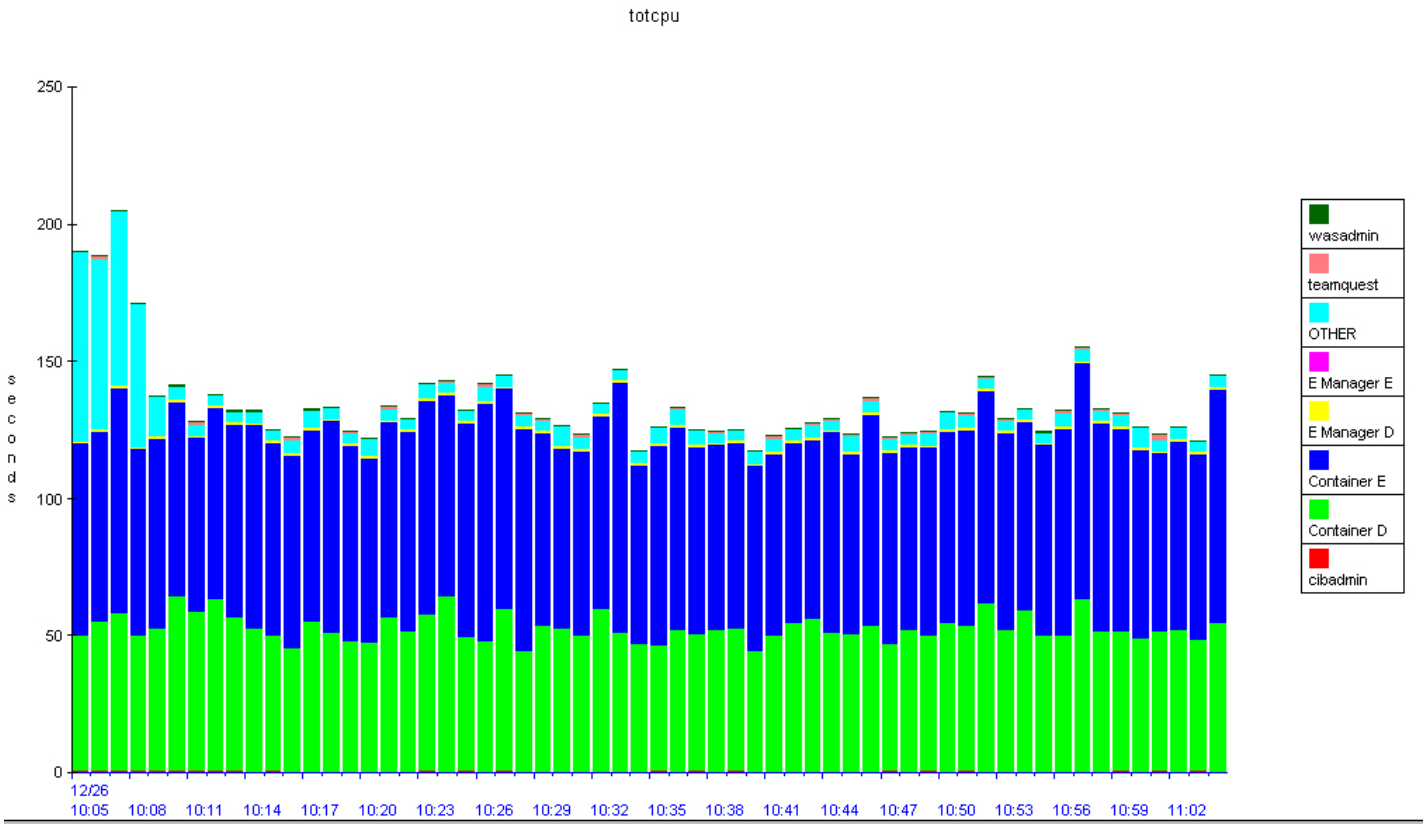


Figure 11
This report shows the breakdown of the workloads consuming CPU over a one-hour period.

I then can drill back down to the process table and include “workload: Application Workloads” as a field. Then, I click on the new column header and then click “Set.” I get the following report.

Row	workload:Application Workloads	command	login	nproc	totcpu	pctcpu	rss	prss	majflt	pid	pri	lioch	avgn
Summary	<Multi>	<Multi>	<Multi>	136	02:34.05	64.11	9925080	<N/A>	80	<Multi>	54	120757.8	12
1	Container E	java	wasadmin	3	00:56.24	23.41	3953840	<N/A>	4	<Multi>	71	51495.7	4
2	Container D	java	wasadmin	3	00:47.96	19.96	3462224	<N/A>	0	<Multi>	71	36392.6	4
3	OTHER	<Multi>	<Multi>	104	00:46.03	19.16	238696	<N/A>	0	<Multi>	52	85.3	
4	teamquest	<Multi>	<Multi>	12	00:02.24	0.93	48448	<N/A>	75	<Multi>	48	32740.2	
5	E Manager D	java	wasadmin	1	00:00.87	0.36	1122776	<N/A>	0	20195	71	6.0	2
6	wasadmin	java	wasadmin	2	00:00.70	0.29	1088184	<N/A>	1	<Multi>	70	37.7	1
7	cibadmin	<Multi>	cibadmin	11	00:00.00	0.00	10912	<N/A>	0	<Multi>	56	0.2	

Figure 12
A drill down from a workload chart to the process table shows the top consumers of various IT resources.

Now, I can quickly identify top CPU workload, top I/O workload, etc.

Another useful graph I found was the lioch, which is a pre defined workload report.

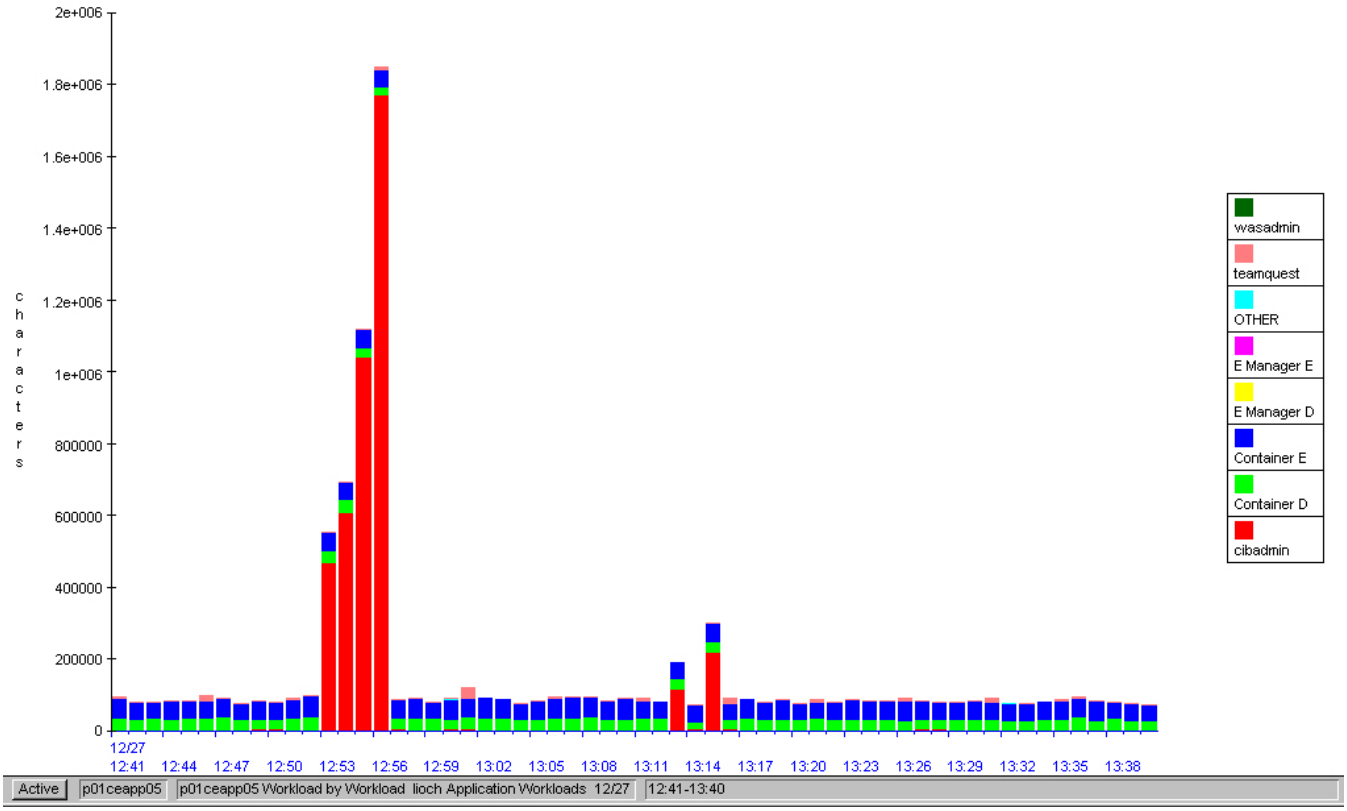
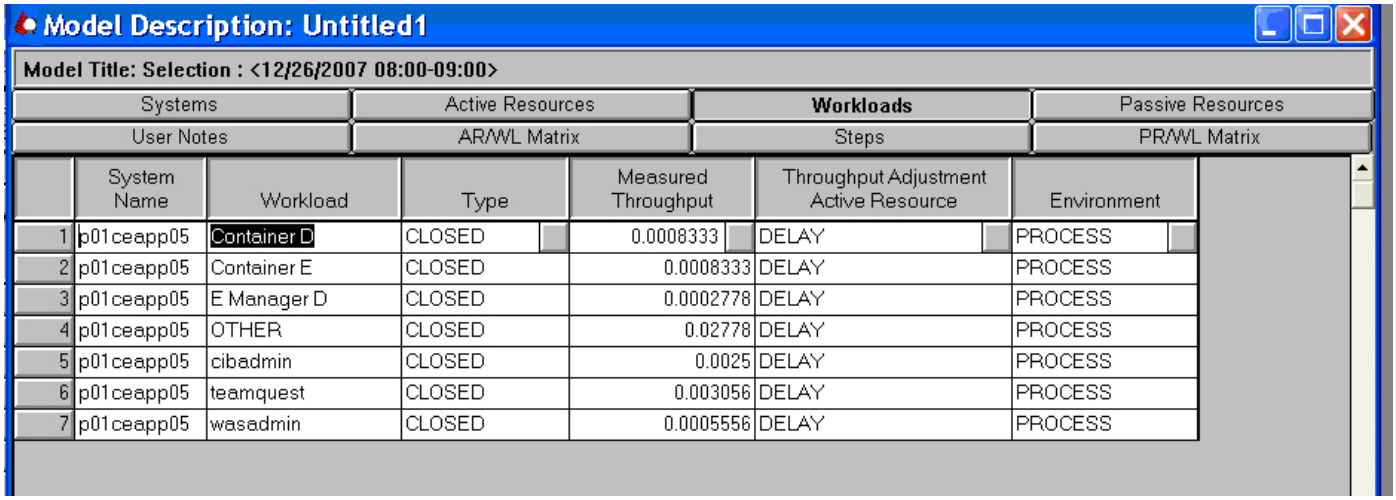


Figure 13
This chart clearly shows which workload is responsible for high I/O wait.

This report allows me to easily identify which workload is responsible for high I/O waits.

Step 5: Capacity Planning and Modeling

These workloads are not only available in TeamQuest View for performance analysis, but also are carried over into TeamQuest Model. We use TeamQuest Model for what-if analysis as part of the capacity planning process.



	System Name	Workload	Type	Measured Throughput	Throughput Adjustment Active Resource	Environment
1	p01ceapp05	Container D	CLOSED	0.0008333	DELAY	PROCESS
2	p01ceapp05	Container E	CLOSED	0.0008333	DELAY	PROCESS
3	p01ceapp05	E Manager D	CLOSED	0.0002778	DELAY	PROCESS
4	p01ceapp05	OTHER	CLOSED	0.02778	DELAY	PROCESS
5	p01ceapp05	cibadmin	CLOSED	0.0025	DELAY	PROCESS
6	p01ceapp05	teamquest	CLOSED	0.003056	DELAY	PROCESS
7	p01ceapp05	wasadmin	CLOSED	0.0005556	DELAY	PROCESS

Figure 14
TeamQuest Model allows what-if analysis on workloads.

Since these workloads are now discrete objects, I can change the rate of growth on each one independently.

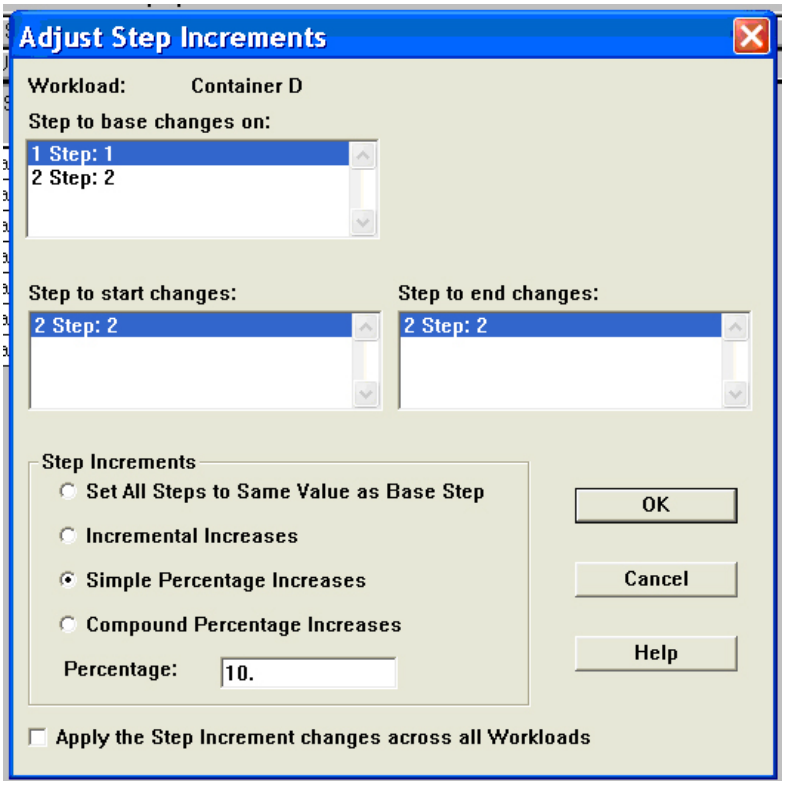


Figure 15
TeamQuest Model allows users to see the impact of growing workloads over time.

I used this dialogue to grow the Container D workload by 10%. I could also create a model that moves Container D (or any other workload) to a different server to see what the effects will be.

Summary

1. Moving Forward

TeamQuest workloads allow me to reprocess previously collected data into newly defined workloads. With this capability, we will be creating a drill-down process for workload definitions. Right now, we run most workload definitions generically between infrastructure components and major application components. However, if a noticeable change occurs or problem develops in the infrastructure workload, we reprocess the data using different workload definitions to show which technical discipline (system, storage, security, etc) needs to review the issue.

2. Hidden Gem — Input to Chargeback Systems

In order to apply this workload methodology across our server farm, a strong naming convention is needed to identify processes and the workload to which they belong. This can be derived from either the command or fullcmd fields. Once the naming convention is in place, we can use the performance data as input to our chargeback system because we will be able to identify which application is driving resource utilization on a given server where multiple applications share a common server.

3. Conclusions

Strong workload analysis is key to solving business issues in performance analysis, capacity planning and chargeback. The ease of use and flexibility of TeamQuest software help solve these key issues effortlessly. The key differentiators with TeamQuest that I have found are:

- Ease of use – All of the above outlined steps worked the first time I tried them and I did not have to call TeamQuest Support for even so much as a clarification question. It is extremely easy to implement and very intuitive.
- Flexibility – The ability to post process data so that multiple workloads can be applied against the same data is key in that you never know which workload you may need to subdivide to resolve business issues.

I have tried other performance data collection software and have found the workload classification process in those products so cumbersome and inflexible that we abandoned its use.

TeamQuest Corporation

www.teamquest.com

Follow the TeamQuest Community at:

Americas

One TeamQuest Way
Clear Lake, IA 50428
USA
+1 641.357.2700
+1 800.551.8326
info@teamquest.com

Europe, Middle East and Africa

Box 1125
405 23 Gothenburg
Sweden
+46 (0)31 80 95 00
United Kingdom
+44 (0)1865 338031
Germany
+49 (0)69 6 77 33 466
emea@teamquest.com

Asia Pacific

6/F CNT Commercial Centre
No. 302 Queen's Road, Central
Hong Kong, SAR
+852 3571-9950
asiapacific@teamquest.com