WHERE AND HOW PERFORMANCE TESTING MADE A DIFFERENCE FOR THE COMMON OPERATING ENVIRONMENT (COE)

System Description:

The software under development was the Common Operating Environment (COE) sponsored by the Defense Information Systems Agency (DISA). It provides the foundation for the Services to build their Command and Control systems. The COE is not a system itself but provides many functions that are commonly used by these other systems, such as, security, system administration, mapping, and communications and data fusion. As a foundation it was critical that the software performs all the functions expected and that it also performs efficiently enough so that sufficient computer resources are left so that all the other components can operate the way they needed to.

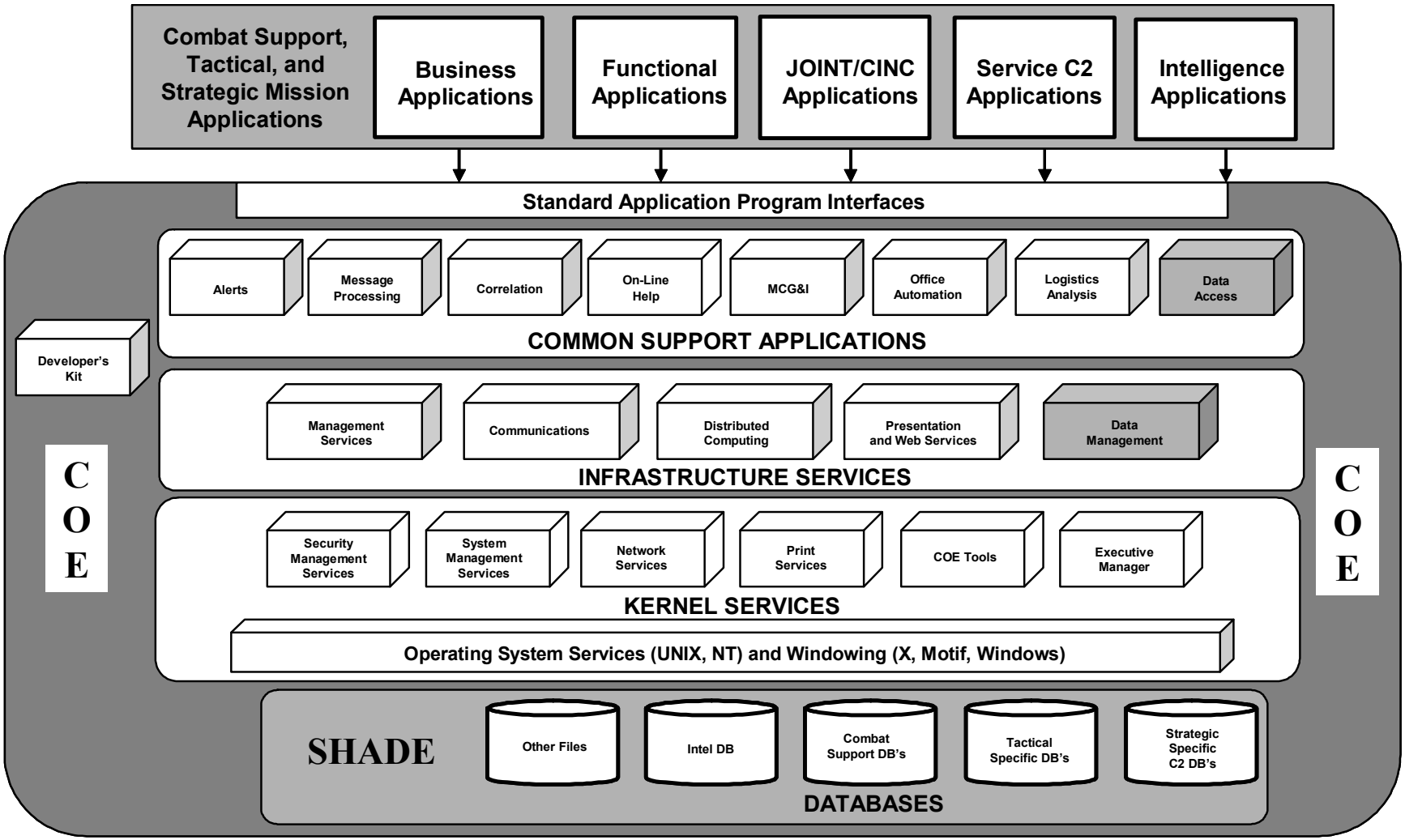
The new software was being updated to take advantage of new technology, both hardware and software. Computers are more powerful and cheaper than they were 15 years ago and many of the young service people are familiar with computers and the standard Windows interface. New programming languages promised better performance, flexibility and the ability to interact easily with commercial products.

Three developer organizations were involved in building different parts of the foundation, operating under three different contracts. There appeared to be little interest in them cooperating or coordinating their efforts which led to several stalemates in progress of the development. This Tiger Team effort was the result of one of the stalemates.

People who attended the Fall 2003 Workshop on Performance and Reliability may remember the presentation Mr. Chris Johnson gave. This paper is discussing the same project but is expanding the discussion to cover how we organized our testing problem, why we chose to measure the kinds of activities that we did and what the effects of those decisions were. The test effort took place from January through May of 2002.

The following figure shows the conceptual organization of the COE software components and how they are expected to relate to other, add-on components.

COE Foundation Components



Problem Statement:

The published release date was rapidly approaching and many problems still existed. For example,

- The foundation components wouldn't run continuously for 48 hours without crashing,
- the user initialization time was greater than 5 minutes when a modest number of tracks were displayed on the World Map,
- it appeared that parts of the new user interface were generally much slower to respond than the currently deployed system and
- the new user interface required more mouse-clicks to get to the same features.

The objective of the Tiger Team was to get the software to a point that the services could begin formal integration, certification, operational test and fielding. The sponsor required that the performance criteria be that

- The newly developed software performs "at least as well as the currently deployed system".

The developers pointed out how quickly background processes completed and how quickly data could be moved from one internal component to the next. Proponents of the deployed system talked about how hard it would be for people to learn the new user interface (that was substantially different from the old).

Meeting the performance criteria was complicated by several facts, among them, the new software was required to handle substantially more data, the symbology was more complex to render, and no baseline performance data existed for deployed system. There was also the problem of defining what performance we wanted to measure and improve.

We chose to define "System Performance" as made up of the following three parts,

- End-user experience
- Resource Usage
- Endurance

We selected these areas to concentrate on because a good end-user experience would be necessary to get buy-in by the user community; good resource utilization was required for the follow-on components to be able to run; and adequate endurance would allow for operational stability.

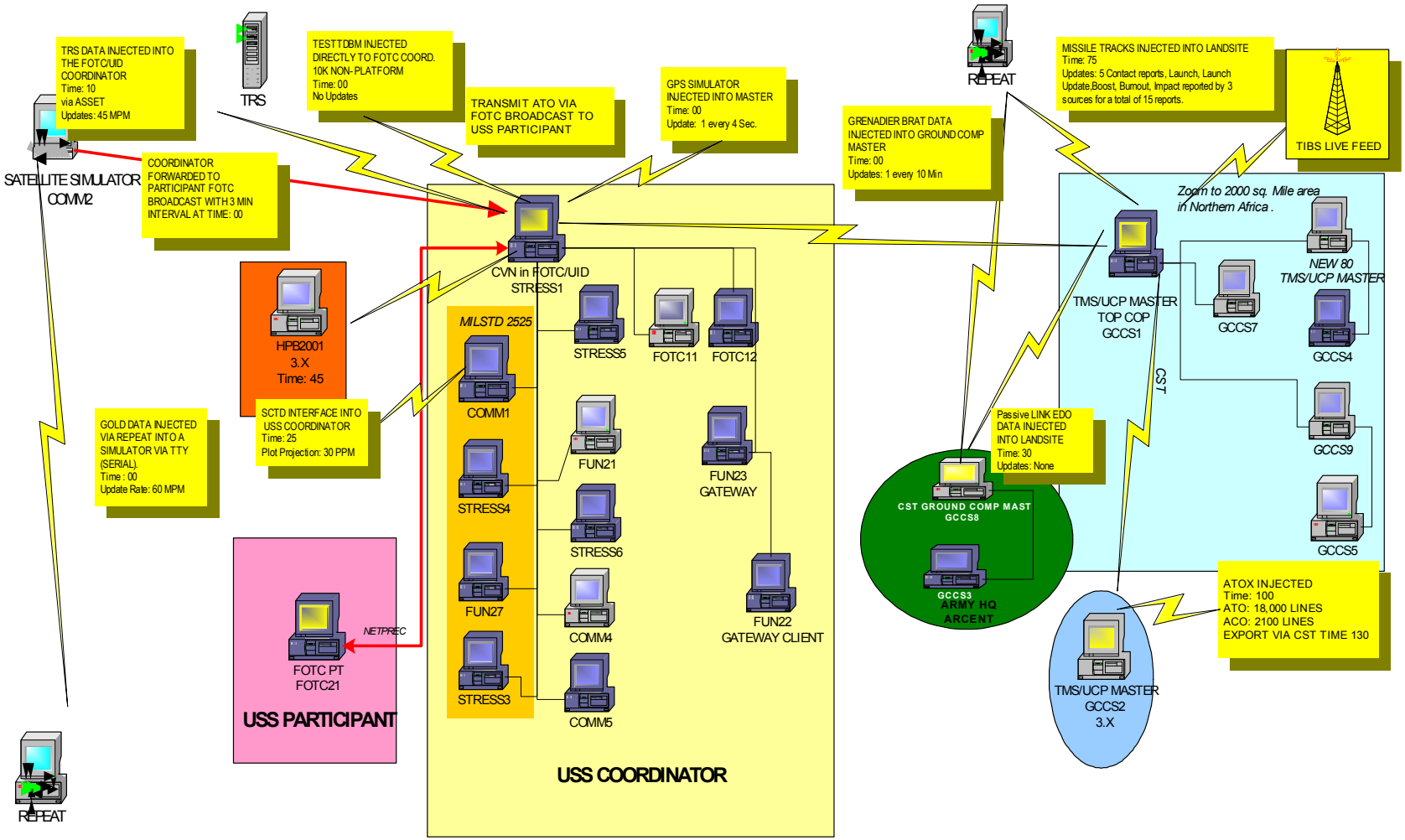
In addition we needed to capture the end-user performance parameters of the deployed system for comparison purposes.

Lab Setup:

We needed to emulate key components of a very large wide-area network (WAN) composed of many local area networks (LANs) that are linked in order to share data. Servers capture data from a variety of sensors and other data sources, perform basic analysis and "data fusion" functions and then update the location and other data for an "object" on land, sea or in the air. The servers periodically update other servers and clients "down-stream" with the data that they have requested. The updates may occur as late as every 3 minutes or as frequently as whenever data has changed on the primary server.

We simulated 3 different types of LAN connected to a "TOP COP" server that was programmed to update the down-stream servers anywhere from 0 to 5 minutes. (The TOP COP server is the primary point of entry for data into the WAN.) We simulated the different basic message types the foundation was expected to process. We used the Team Quest performance-monitoring tool to capture CPU and memory usage and process queue length. (Team Quest was selected for its relative ease of use and easy-to-use data analysis and reduction tools.) Our WAN had 9 servers and 15 client workstations. All the client workstations had operators executing a set of actions and recording computer clock-time, as did some of the servers. The operators were recruited from other DoD test facilities; the other test organizations were interested in learning what they could about the new software since they would be responsible for testing with it in the near future. We did not have load-testing tools at the time and couldn't justify the time and expense since the duration of this test effort was relatively short. (Not enough time to acquire, learn, and deploy any new tool in the 4-6 month window of the Tiger Team.)

The following diagram shows the laboratory setup and the roles and relationships between the LANs.



Laboratory Set Up

Method:

User representatives, developers and testers were organized into a Tiger Team working in the same laboratory for a period of four months. The original test team was comprised of six testers, a test lead and a test director. In order to provide the quantity of testing that we needed to accomplish testers were solicited from other DoD organizations that would eventually be involved in testing the new system after the foundation was integrated with the follow-on components. People were rotated through every 2-3 weeks. This high turnover meant that the tests had to be well documented, easy to execute and make it easy to record results.

Following is the basic organization of the test effort:

- We executed a series of tests to determine the performance baseline of the new and deployed systems, primarily the end-user experience and the server-processes for the core functions (communications, data fusion, mapping).
- We developed performance data for each individual component of the foundation
    - Communications
    - Cartographer / Symbology display
    - Data dissemination
    - Additional core components (as available)
- The developers provided new deliveries every two weeks
- We re-built the foundation components into the full system one component at a time in order to measure the effect of each component on overall performance.
- We stressed the system with user activity and increased communications input.
- We checked end-user performance using a list of standard user actions and a stopwatch
- We monitored server and client internals using Team Quest
- We maintained a separate minimum system configuration running at all times that would not be affected by the stress tests in order to watch the "endurance" of the software. That is, the length of time the software runs when users or a heavy input data load does not tax it.

The stress tests were a combination of user activities and data injections. The following table contains some typical user actions that were measured each time a new software delivery was made.

The endurance test was executed using a small LAN, one server and one client. The server was preloaded with a minimum amount of data, the data injection was started at a representative rate and the client ran Rational Robot scripts to simulate a low level of user interaction. The processes would be checked periodically throughout the day and it would be noted when processes stopped unexpectedly or the system became unresponsive.

Requirement	4.x Objective (all platforms & track loads)	Definition
Initialization:		
Reboot time from initiation to Login screen.	<= 105s	Reboot system (Solaris -->sysadmin 'restart' icon); measure time btwn initiating action and when Login Screen appears. W2K: go to Start >Restart; time from 'OK' to Login Screen.
Login to COE Processing complete	<= 40 sec	From login window, enter username/password; measure time between <CR> and Msg window 'Login to COE Processing Complete'. Enter total number of tracks in column to the left.
Login to Chart Display (Whole World)	<=120s @ 10K <= 180s @ 20K	From login window, enter username/password; measure time between <CR> and Chart / System ready to use (processing completed).
Launch 'Center-Width' Pull-down Menu selection to window display	<= 1s	Select 'Center-Width' from Map Options pull-down; measure time for GUI window to appear
Launch Message Log via Icon selection to window display	<= 2s	Launch 'Message Log' from icon; measure time from for GUI Window to appear. Different between 3.x & 4.x
Logout from select to Login Screen	<= 9 s	Solaris: from Click 'Exit' then 'OK'; measure time btwn <CR> and when Login Screen appears. W2K: from Ctl-Alt-Delete, measure time btwn clicking 'Yes" to Log Off window and when Login Screen appears.

Typical User Actions

Test Results

The major results of the performance testing are summarized below. In retrospect very little of what was discovered was "subtle" or hard to understand. The results ran the gamut from "get more capable hardware" to having SUN Microsystems provide engineering support for JAVA garbage collection problems.

Major problems and recommended solutions for end-user-experience:

- Long login times (3-5 minutes) depending on number of objects being rendered on screen;
    - Changed the way the software rendered and updated objects; suggested a CONOPS where default map and number of objects is much smaller in size.
- Minimum client configuration not powerful enough to support processing requirements;
    - Original client minimum was 256 MB RAM and 400 MHZ processor;
    - More realistic client minimum was changed to a Pentium-3 700mhz processor and 512 MB RAM.

Software Load:		ICSF+CCE/CME: 2525	ICSF+CCE/CME: NTDS
	Own Ship Only; no Track Load	W2K	W2K
Step	Name	Average Processing Time	Average Processing Time
6	Boot, login as ICSF profiled user until map complete	102.72	101.29
8	Relogin as ICSF profiled user until map completes	64.90	65.56
9	Launch 2nd Chart	19.95	19.39
10	Close 2nd Chart	2.70	2.43
	10,000 Track Load	W2K	W2K
Step	Name	Average Processing Time	Average Processing Time
6	Boot, login as ICSF profiled user until map complete	191.42	168.72
8	Relogin as ICSF profiled user until map completes	154.07	128.78
9	Launch 2nd Chart	97.18	75.12
10	Close 2nd Chart	40.98	19.96
	20,000 Track Load	W2K	W2K
Step	Name	Average Processing Time	Average Processing Time
6	Boot, login as ICSF profiled user until map complete	281.59	247.13
8	Relogin as ICSF profiled user until map completes	245.21	209.79
9	Launch 2nd Chart	175.80	149.82
10	Close 2nd Chart	158.04	101.00

Without filtering the display

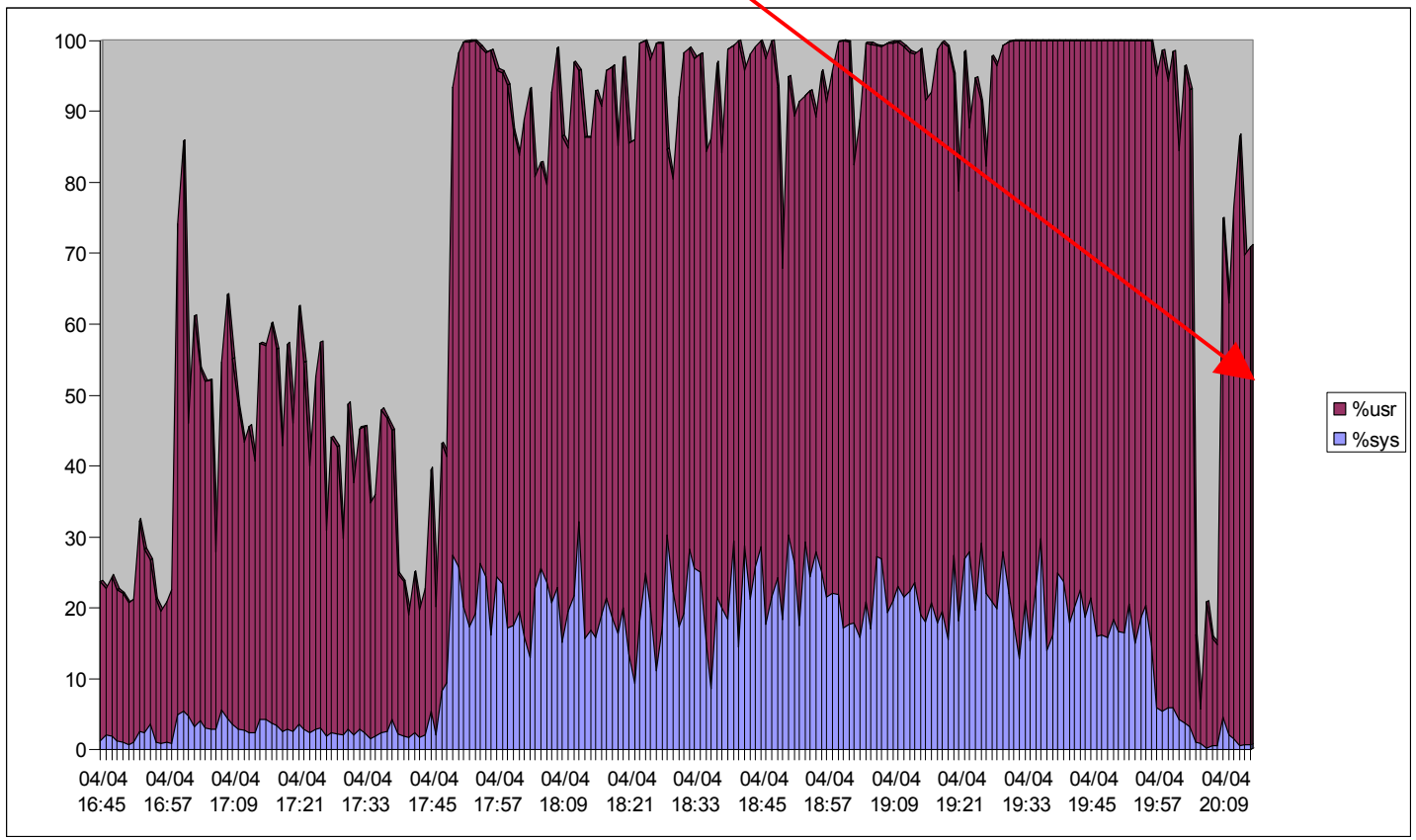
	ICSF+CCE/CME: 2525	ICSF+CCE/CME: NTDS
	W2K	W2K
Step	Average Processing Time	Average Processing Time
6	101.92	101.64
8	65.27	66.48
9	21.57	21.23
10	3.39	2.61
	W2K	W2K
Step	Average Processing Time	Average Processing Time
6	122.52	120.02
8	87.43	87.79
9	30.76	28.08
10	4.63	4.44
	W2K	W2K
Step	Average Processing Time	Average Processing Time
6	135.96	133.20
8	100.30	113.41
9	39.73	39.99
10	7.68	7.15

Display of objects filtered

Major problems with resource usage:

- Several processes used more than 50% of the available CPU each
    - The developer was able to rework the code to push CPU usage reliably below the 50% threshold
- Several processes had memory leaks that caused system failure after running overnight
    - In each case controlling the introduction of new programs while monitoring the memory and CPU usage allowed specific processes to be identified as the offenders and fixed
    - Problems with JAVA garbage collection were documented so that SUN engineers could suggest alternative switch settings and also said we should upgrade to the new version

stress1 SOLARIS:Process	Apr 04, 2002 20:10:00 - 20:11:00	
command	pctcpu	fullcmd
<Multi>	63.18	<Multi>
java	49.16	/h/COTS/JAVA2/bin/../bin/sparc/native_threads/java -Dafw -Xss1280k -XX:-UseComp
java	3.96	/h/COTS/JAVA2/bin/../bin/sparc/native_threads/java -Xss768k -Xmx24M -cp /h/COE/
Tdbm	2.27	Tdbm
Xsun	2.15	/usr/openwin/bin/Xsun :0 -nobanner -auth /var/dt/A:0-BlaGbb
CSTTrkDec	1.6	CSTTrkDec CSTTCP
Cartographer	0.57	Cartographer
CSTTCP	0.51	CSTTCP CSTTCP
java	0.48	/h/COTS/JAVA2/bin/../bin/sparc/native_threads/java -Dtdbm.alert -cp /h/COE/Comp



Major problem of system endurance:

- System didn't run for more than 24 hours at a time because of the CPU and memory usage problems discussed above.
    - Endurance was increased from 24 hours to 384 hours while increasing the communications load from nil to "normal" levels

Conclusions / Recommendations

By the end of the Tiger Team effort we had not identified or fixed all the problems (either performance or feature related). The following were some recommendations to the sponsor for items to concentrate on:

- Addressing any remaining/new high priority trouble reports against the COE Foundation
- JVM/garbage collection issue (excessive heap growth/memory leak)
- System login/initialization issue (excessive time from login to useable display)
- Second chart issue (time to bring-up second chart and having to close primary chart to delete second chart)
- Complete detailed functional testing of critical add-on software components and address all high priority trouble reports for them
- Prioritize and address priority-3 (user-interface) trouble reports and change requests that are required by COE systems

Lessons Learned:

- Forced period of intense testing and fixing highly useful to bring together all the groups involved solving a well-defined problem
- Using Team Quest to provide solid, repeatable, detailed information on performance helped bring developers and testers together to solve the problem rather than continuing an acrimonious relationship of finger pointing.