SAP R/3 in a UNIX Environment:
A Case Study for Client/Server Capacity Planning
Glynn B. Giacone and Javier Munoz
BGS Systems

SAP R/3 is a multi-tier client/server application which supports and simplifies business applications in areas of finance, production, distribution, sales, and human resources. It recently has become very popular worldwide, and in many cases drives the purchase of significant amounts of UNIX hardware. Its multi-tier architecture typifies the strategies of mission critical applications implemented for UNIX and Windows NT, where workstations communicate with application servers which in turn communicate with database servers such as Oracle, Informix, and Sybase. This paper is a client/server case study of what instrumentation is available from UNIX, what instrumentation is required from a client/server application such as SAP R/3, and how system and subsystem metrics can be used in an analytic model for sizing and capacity planning.

WHAT IS SAP R/3?

SAP R/3 is an application developed by SAP, a German company based in Walldorf, which supports and simplifies all business processes in a company and which has recently become very popular worldwide, driving in many cases, significant investments in UNIX hardware. SAP R/3 is a tightly integrated solution which is composed of optional modules to perform specialised functions, such as finance, production, distribution, sales, materials management, and human resources. It also contains applications and tools to manage, control and monitor the system. It is implemented according to a software-oriented multi-tier client/server principle for open systems (UNIX, OpenVMS, Windows NT and the AS/400), with application servers communicating with database servers across the network. The inherent portability of SAP R/3 lets a variety of databases be utilised, including Oracle, Informix, Sybase and Software AG, and permits the presentation interface front end to run Windows, OS/2, OSF/Motif and Macintosh.

What is the SAP R/3 Architecture?

The SAP R/3 presentation, application, and database server architecture enables an uncoupling of the application logic from the presentation level and the database, a prerequisite for distributing tasks in a client/server environment. While the most likely installation of R/3 will resemble that in Figure 1, it is also possible to find single-system configurations where the interface, application and database servers execute under the same UNIX system.
Figure 1 - SAP R/3 Three-tier Client/Server
How Does Work Flow in SAP R/3?

Normally, a user logs on to the SAP system, calls up a series of transactions through data entry panels, and then logs off. Programs invoked by these transactions in the middle tier (application server) implement business logic. These programs are interpreted by the R/3 runtime system installed on every application server. Business transactions are functionally cohesive processing units that make changes to the database, e.g., posting credit and debit items, creating an order and reservation, etc.

From the R/3 viewpoint, a transaction is a sequence of dialogue steps that logically follow each other, with each dialogue step representing a “dynamic program” (dynpro) or screen. An SAP transaction typically consists of a number of dialogue steps that can be processed by different work processes. Asynchronous updates permits dialogue and database-updating handling by different work processes, or even on different computers.

![SAP R/3 Process Architecture](image-url)

The process architecture, shown in figure 2, depicts that an application server receives requests from the connected presentation servers and routes these requests to the database server via work and database server shadow processes. Communication among all servers is realised by a message server. When a request arrives to an application server, it is placed in its dispatcher queue, and it is not dispatched until a work process becomes free. Each work process, which is the basic “working engine” within R/3 and corresponds to one UNIX process, supports a special service (e.g., dialogue and background). The number and type of work processes that run within a given application server is a configurable entity. When a work process becomes free, the user’s context is placed in the corresponding working area of the work process (a task known as roll-in). Each work process has a...
corresponding “shadow” process on the database server to handle all database accesses. Local buffers exist at each application server to minimise database accesses.

An instance defines an application server with an SAP R/3 client/server environment. For each instance, the user defines the instance name, the host name where it runs, the services it provides (e.g., dialogue, background), and the number of work processes to run of each type. This information, together with the size of the local buffers, the roll area, and other details is stored in the profile for the instance (a UNIX file of the form /usr/sap/<sid>/SYS/profile/<instance name>).

The number of instances and their configuration may change during the day to divide, for example, the on-line and the nightly operation. During the on-line operation, there could be more work processes configured to accept dialogue services (i.e., data entry panels), while during off-hours (nightly operation), more background work processes could be configured to handle lengthy database update operations and the generation of reports.

A job scheduler assigns jobs to execute in the background work processes based on a scheduling table and on the priority class of the job (class A has higher priority than class B which in turn has higher priority than class C). When possible, the scheduler attempts to balance the workload among the different background processes.

**How does SAP R/3 Architecture affect Performance?**

As with most UNIX interactions, the response time of the SAP R/3 transactions are affected by the load of the UNIX nodes where the application servers reside, the load on the database and its server, and the load on the network. Additionally, SAP R/3 transactions are also affected by the SAP architecture itself (e.g., the scheduler queue, roll-in, etc.) and the configuration selections for the instances (e.g., number of dialogue and background processes, size of the different buffers, etc.).

Typical capacity planning questions raised in this environment are:

- What is the utilisation of the application and database servers? Are there any bottlenecks?
- How many users work on the various instances? Is the load well balanced? How many can the target or production system handle?
- Which are the most frequently executed transactions? How is their response time? What happens to response time as the number of users increase?
- Which processes are causing the load? Are they SAP processes? Oracle processes? What else is running?
- Are there variations in the response time for the various instances?
- How has the workload changed during the day? How about the response time? Are there any peaks?
- Are there any applications with much higher I/O access times than others? Are they located on slower disks?
- How has the load changed during recent days? Were there any special actions performed?

A key aspect in understanding and reporting the performance of an SAP R/3 installation is response time at the SAP transaction level (e.g., the response time for a data entry screen or a background program). Response time for an SAP R/3 transaction is composed of

- wait time: time spent in the dispatcher queue until a work process becomes free
- dispatch time: time spent in the work process such as:
- CPU time: CPU time spent by the work process
- Load time: time spent loading system data, programs, dialogues,...
- DB request time: time spent by the shadow process to deliver a request
- Other: other miscellaneous time (roll time, enqueue time,...)

Fortunately, SAP R/3 collects performance data at this level to ease the analysis task. It also collects detailed data about OS, database, network, and buffer statistics. User access to this performance data is done via **CCMS** (Computer Center Management System) which is an integral component of every SAP R/3 installation.

**What is the Purpose of CCMS?**

In the mission statement of CCMS we can find the following goals:

- Enable the unattended and continuous operation of an SAP R/3 installation.
- Provide the tools for analysing the workload, the resource consumption in the different system components and the efficiency of all tunable system parameters.
- Provide, when possible, automatic workload balancing.
- Enable remote operation

The functionality of CCMS can be grouped into two major areas, controlling and monitoring.

The controlling tasks focus on: (a) SAP system management (configuration, reconfiguration, startup, shutdown, system alerts, and backup), (b) background processing and scheduling of jobs, and (c) workload distribution and balancing. For example, through CCMS, the SAP R/3 administrator can define “operation modes” to fit the workload requirements of on-line and nightly operations. Job executions can be scheduled to occur at certain times and to respond to external events. The administrator can also be alerted whenever an instance is not running or when backups have not been performed in a timely fashion.

The monitoring tasks focus on: (a) performance monitoring and alerting for SAP applications, database, operating system, and the network, (b) performance analysis of troubled areas and components, and (c) historical performance trending and analysis (a performance database that spans days, weeks, and months).

**What are the CCMS Monitoring and Performance Analysis Capabilities?**

As with all other functions within CCMS, access to the monitoring and analysis tasks is done through a GUI, and selections are made from pull-down menus and data entry forms. Alerts triggered by the SAP R/3 data collectors are viewed in tabular and graphical formats. Performance reports at various levels of detail can be sorted, summarised, and permit drill-down to focus on additional detail. The data access forms also have powerful selection capabilities to accommodate queries of the type “show me all transactions with a response time above 2 seconds” or “show me how many times transaction SQ00 has been executed this week”. The performance data collected can be enhanced with SAP R/3 SQL and ABAP program trace tools to analyse all problem areas in as much detail as is required.

The performance data collected for real time and historical consumption belong to one of: (a) UNIX system and network metrics, (b) database metrics, (c) metrics on buffers and other SAP R/3 objects, and (d) workload metrics. Figure 3 lists some of the metrics collected from each category.
Alert thresholds for all real-time metrics are customizable for each instance. There is a three-level alert scheme for each alert: green-OK, yellow-Warning, and red-Problem. When an alert is triggered, you can acknowledge it (accepting it temporarily), in which case a new alert will occur if a new maximum value is reached, or you can reset it (deleting the local alert), in which case will occur if the threshold is reached again.

**What about SAP R/3 Sizing and Capacity Planning?**

While CCMS gives monitoring and analysis capabilities, every installation has to be prepared to size capacity for a newly acquired set of modules, and to plan for the impact of growth on remaining capacity. To accomplish this goal, there needs to be an ability to correlate and synchronize SAP CCMS performance reports, and UNIX/RDBMS collected performance metrics to an SAP business unit mapping scheme. This mapping will need to take dialogue step activity counts and performance data and aggregate this information to something that the potential user cares about - answering the what-if impact of additional users running an existing SAP application. Workload characterisation at the SAP application module can be expressed in terms of resource consumption, e.g. CPU msec and physical I/O’s, for an “average” dialogue step. This dialogue step, or transaction, needs to have attributes of existing or anticipated arrival rates, and a resource consumption profile. These attributes can be utilized by analytic modeling, to predict, as a function of additional users, response times, utilizations, and throughput for SAP R/3 application modules.

**What about SAP R/3 Workload Characterisation?**

All work within SAP can be categorised into different levels according to the following hierarchy:

→ Application Area
How can an Analytic Model be used for SAP Sizing and Capacity Planning?
An analytic model can be used to answer “what-if” questions to evaluate alternatives for current performance problems and plan for expected capacity needs. It can “predict” how configuration and workload changes affect system performance. An analytic model for SAP sizing and capacity planning needs to include characteristics such as:
• Representation of multi-vendor heterogeneous systems
• Memory in the system
• Uni and Multi processors
• Disk drives
• Logical volumes
• Client server workloads and transactions
• Several different types of transactions in the same workload
• Independent and dependent (triggered) transactions
• Synchronous and asynchronous dependent transactions
• Amount of private and shared memory required by a transaction
• The way memory is used by transactions
• CPU priority scheduling
• Multiprogramming Constraints to deal with finite number of servers
• Networks of different types and speeds

Analytic modeling is based on proven queuing theory to define and model workloads, along with their resource requirements and usage constraints. “What-if” analyses can be performed to evaluate changes to CPU, memory, I/O, scheduling priority, transaction rates and mixes, the number of concurrently running transactions, file server routing, transaction routing among nodes, and other attributes of system and application environments. A model for a multi-tier SAP R/3 domain of nodes needs to have the following attributes:

• Processing demands - SAP R/3 application module workloads and transaction definitions with measured/anticipated arrival rates
• Usage constraints - UNIX process scheduling and application multi-threading
• Processing resources - system configurations with CPU’s, memory, disks, network, etc.

SAP R/3 models can be built from UNIX, SAP R/3 CCMS, and RDBMS measurements of production/test systems. Each SAP application module needs to have its dialogue step processing broken down by four key components:

• CPU (including non-I/O portion of paging delay)
• I/O (user and system rates, NFS and Logical Volume details)
• Network (TCP/UDP packet flow statistics)
• Wait for servers (multi-programming constraint) where transactions run

CCMS Measurements can be used to derive service, e.g. CPU msec, and queuing theory predicts performance, i.e. workload and transaction response times, throughputs, and resource utilizations such as disk, memory, and network utilizations and throughput rates.

Figure 4 shows how UNIX can measures SAP activity in the form of a combination of UNIX processes. However, this is not sufficient to accommodate the type of questions asked by SAP administrators. As indicated previously, workloads in an SAP model must represent SAP applications, whose transactions relate directly to the number of dialogue steps reported in CCMS.
The basic process to create an SAP model consist of the following steps:

(1) gather data from all the UNIX servers that constitute the SAP installation for a selected interval, (e.g. from 10-11 a.m.)

(2) using CCMS, generate reports at the beginning and end of the same interval for each SAP instance at the SAP application level,

(3) create SAP workloads by combining resources consumed for UNIX processes,

(4) apportion UNIX process resource consumption to accommodate the workloads using the CCMS application level data from step 2.

Gathering UNIX data for step 1 can come from UNIX utilities such as SAR, or commands such as ps, vmstat, iostat, etc. Alternatively, commercial system management tools access the kernel directly can reduce measurement overhead.

For step 2, CCMS can produce a report to aggregate tcodc-level data to SAP application-level data. However, cumulative data is reported, therefore one must take the deltas between the values at end and the beginning of the interval to determine dialogue step interval processing.

For each SAP instance “i” and for each SAP application “a” running on instance “i”, we have:

- \( CCMS\_ARR(a,i) \): number of dialogue steps per hour
- \( CCMS\_RESP(a,i) \): response time per dialogue step
- \( CCMS\_CPU(a,i) \): CPU consumed per dialogue step
- \( CCMS\_DBR(a,i) \): Num. of DB requests per dialogue

Steps 3 and 4 form the bulk of the process to create SAP workloads and build the initial model.

How can SAP Workloads be built in the Initial Model?

While SAP installations can present configurations that may span from one to many UNIX servers, we will use as our sample configuration (see Figure 5) one having a UNIX server (named “as” for application server) with one SAP instance executing several SAP applications and another UNIX
server (named “ds” for database server) running the database instance (e.g., using Oracle). Assume that the presentation server is a Windows PC. Extending the example to multiple SAP instances on the same or on different application servers is a straightforward exercise.

The first step is to collect data for the UNIX processes in this environment that results in two files (one for each UNIX server for the period of interest). Similarly, we will have one CCMS report for the SAP instance covering this same period.

![Sample configuration used](image)

The second step in the creation of the initial SAP model will be to identify the composition of the initial workloads, which we will create and name as follows:

**Workload SAP_WORK:** Consists of two transactions, one independent and the other dependent. Independent transaction AS_SAP_WP consists of all SAP work processes running in node as (the corresponding UNIX PIDs can be seen through CCMS). The dependent transaction, DS_SAP_DB, consists of all shadow processes in node ds that perform the database operations on behalf on the SAP work processes.

**Workload SAP_REST:** It has two independent transactions. AS_SAP_REST groups all other SAP processes running in node as, while DS_SAP_REST consists of all other SAP/RDBMS processes running in node ds.

**Workload OTHER:** It also has two independent transactions. AS_OTHER consists of all remaining processes running in node as, and DS_OTHER consists of all remaining processes running in node ds.

Note that with this initial (and simple) 3-workload characterisation, with all three workloads visiting both UNIX nodes, we are now ready to use the CCMS output to “break-up” workload SAP_WORK into multiple SAP applications (one workload per SAP application).

**How can CCMS data be used to Apportion the Initial Model?**

First, we need to make some base assumptions about SAP’s consumption of system resources. You will need to assess the risks involved if some of these assumptions are invalidated.
All SAP applications are equally affected/rewarded by SAP’s local buffers, making the ratio of database requests to physical databases accesses equal for all applications within a given UNIX node.

Work performed by the RDBMS on behalf of the SAP applications is directly related to the number of database requests issued by the SAP applications.

All local/remote I/O performed by the SAP applications is directly related to the number of database requests.

Given that all applications within an SAP instance use the same work processes, the memory/working set requirements for all applications is the same within a given UNIX node.

By the same venue, CPU priorities for all applications within the same instance is the same.

The consumption of resources by the SAP processes which are not work processes is independent of SAP application activity within each UNIX node.

Similarly, all resource consumption by the RDBMS processes which are not the shadow processes of the SAP work processes is independent of SAP application activity.

With these base assumptions in place, we can apportion the initial model by employing the following disaggregation procedure on initial workload SAP_WORK.

Arrival rates for workload SAP_APP(a) : Represent the number of users as a function of measured dialogue counts per user so that the arrival rate of the independent transaction of SAP_APP(a) is equal to CCMS_ARR(a)

Set the arrival rate of the dependent transaction of SAP_APP(a) to 1 (i.e., every independent issues one dependent).

CPU service for workload SAP_APP(a) : Use the ratio of CCMS_CPU(a) to the sum of CCMS_CPU(a) for all applications (a), to break up the independent and dependent transactions’ own CPU service time.

I/O count for workload SAP_APP(a) : Use the ratio of CCMS_DBR(a) to the sum of CCMS_DBR(a) for all applications (a), to break up the independent and dependent transactions’ own I/O count.

Setting the other workload parameters for workload SAP_APP(a): Carry the initial values for the independent and dependent transactions of initial workload SAP_WORK to SAP_APP(a) for memory usage, working set, page I/O counts, and network traffic, making sure to accommodate for the difference in transaction rates so that total paging and network usage remain unchanged from the initial model.

SUMMARY - SAP R/3 MODELING AND CAPACITY PLANNING

Figure 9 shows an example of model response time results when UNIX system measurement data apportioned by CCMS aggregated tcode performance reports are used to build a performance model. In this example, the model workloads include SAP application modules FI, MM, SD, BC, AM, etc. For each workload, the model calculated response time between 10% and 20% of measured response time from CCMS. Figure 10 shows CPU utilisation, not available from CCMS reports, for each of the SAP R/3 application modules.
<table>
<thead>
<tr>
<th>Workload</th>
<th>Total</th>
<th>CPU</th>
<th>WAIT</th>
<th>CPU</th>
<th>WAIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM@cares</td>
<td>0.29</td>
<td>0.16</td>
<td>0.03</td>
<td>0.05</td>
<td>0.00</td>
</tr>
<tr>
<td>BC@cares</td>
<td>0.26</td>
<td>0.14</td>
<td>0.02</td>
<td>0.05</td>
<td>0.00</td>
</tr>
<tr>
<td>FI@cares</td>
<td>0.19</td>
<td>0.09</td>
<td>0.01</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>MM@cares</td>
<td>1.55</td>
<td>0.20</td>
<td>0.36</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SD@cares</td>
<td>0.33</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Shared@cares</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 9 - SAP R/3 Response-time Modeling
<table>
<thead>
<tr>
<th>Node/Workload</th>
<th>Count</th>
<th>% Util</th>
<th>out of</th>
<th>Q Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>cares</td>
<td>1.0</td>
<td>9.68</td>
<td>100</td>
<td>0.11</td>
</tr>
<tr>
<td>AM@cares</td>
<td></td>
<td>0.74</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>BC@cares</td>
<td></td>
<td>0.43</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>FI@cares</td>
<td></td>
<td>0.71</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>MM@cares</td>
<td></td>
<td>0.19</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>SD@cares</td>
<td></td>
<td>1.57</td>
<td></td>
<td>0.02</td>
</tr>
<tr>
<td>Shared@cares</td>
<td></td>
<td>0.70</td>
<td></td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 10 - SAP R/3 CPU Utilisation Modeling

With these initial modeling results, and additional apportioning of some of the background CCMS processing, using an analytic model for SAP R/3 sizing and capacity planning demonstrates proof of concept. It has the added benefit of being able to answer “what-if” questions such as the following:

- How will the SAP R/3 system perform with increased application usage?
- Will users have acceptable performance when SAP R/3 is fully deployed?
- How many app servers are needed, and what size should they be?
- Will the target database server deliver acceptable performance?

What is missing from SAP R/3 and CCMS is an alternative approach to complement and reduce the excessive amounts of time put into benchmarking efforts that may or may not establish accurate deployment or growth performance expectations. SAP R/3 represents significant investments and corporate strategies. It is a complex system, where it is important to know what’s going on today, but equally important to be able to forecast capacity requirements for tomorrow. A modeling approach in complex distributed heterogeneous environments can identify resource consumption in SAP R/3 business oriented groupings, can correlate resource usage to delivered performance, can predict the impact of user growth on existing applications, and can size demands for new SAP R/3 applications prior to deployment.

We have demonstrated in this paper that modeling technology can be applied to answering a series of questions which normally arise in SAP R/3 installations.
Appendix A: Sample SAP R/3 CCMS Reports

<table>
<thead>
<tr>
<th>Program or Tcode</th>
<th>Dialogue steps</th>
<th>Response time tot(s) avg(ms)</th>
<th>CPU time tot(s) avg(ms)</th>
<th>Wait time tot(s) avg(ms)</th>
<th>DB time tot(s)</th>
<th>Requested kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>TOTAL</em></td>
<td></td>
<td>4.239</td>
<td>2.544</td>
<td>600</td>
<td>378</td>
<td>89</td>
</tr>
<tr>
<td>Spool</td>
<td></td>
<td>1.067</td>
<td>1.319</td>
<td>1.237</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Bckgrd</td>
<td></td>
<td>1.006</td>
<td>657</td>
<td>653</td>
<td>172</td>
<td>171</td>
</tr>
<tr>
<td>MainMenu</td>
<td></td>
<td>554</td>
<td>16</td>
<td>30</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>ST03</td>
<td></td>
<td>388</td>
<td>142</td>
<td>365</td>
<td>40</td>
<td>104</td>
</tr>
<tr>
<td>ABZK</td>
<td></td>
<td>128</td>
<td>27</td>
<td>215</td>
<td>13</td>
<td>101</td>
</tr>
<tr>
<td>ABZV</td>
<td></td>
<td>110</td>
<td>26</td>
<td>235</td>
<td>13</td>
<td>120</td>
</tr>
<tr>
<td>V+03</td>
<td></td>
<td>105</td>
<td>35</td>
<td>334</td>
<td>11</td>
<td>106</td>
</tr>
<tr>
<td>AS02</td>
<td></td>
<td>102</td>
<td>10</td>
<td>98</td>
<td>5</td>
<td>49</td>
</tr>
<tr>
<td>F-43</td>
<td></td>
<td>80</td>
<td>14</td>
<td>175</td>
<td>6</td>
<td>75</td>
</tr>
<tr>
<td>ST04</td>
<td></td>
<td>75</td>
<td>22</td>
<td>297</td>
<td>8</td>
<td>102</td>
</tr>
<tr>
<td>ABZO</td>
<td></td>
<td>73</td>
<td>17</td>
<td>231</td>
<td>6</td>
<td>78</td>
</tr>
</tbody>
</table>

Dialogue step performance is too granular for sizing and capacity planning because the application module is not represented as a planning object.

**Figure 6:** Sample CCMS report showing Dialogue step performance.

<table>
<thead>
<tr>
<th>Application</th>
<th>Dialogue steps</th>
<th>Response time tot(s) avg(ms)</th>
<th>CPU time tot(s) avg(ms)</th>
<th>Wait time tot(s) avg(ms)</th>
<th>DB time total(s)</th>
<th>Requested kBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>TOTAL</em></td>
<td></td>
<td>4.239</td>
<td>2.544</td>
<td>600</td>
<td>378</td>
<td>89</td>
</tr>
<tr>
<td>Shared</td>
<td></td>
<td>1.640</td>
<td>1.337</td>
<td>815</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>Unknown</td>
<td></td>
<td>1.008</td>
<td>658</td>
<td>652</td>
<td>172</td>
<td>170</td>
</tr>
<tr>
<td>BC</td>
<td></td>
<td>606</td>
<td>242</td>
<td>399</td>
<td>82</td>
<td>135</td>
</tr>
<tr>
<td>AM</td>
<td></td>
<td>523</td>
<td>138</td>
<td>263</td>
<td>50</td>
<td>95</td>
</tr>
<tr>
<td>SD</td>
<td></td>
<td>245</td>
<td>95</td>
<td>388</td>
<td>35</td>
<td>142</td>
</tr>
<tr>
<td>FI</td>
<td></td>
<td>209</td>
<td>57</td>
<td>273</td>
<td>17</td>
<td>81</td>
</tr>
<tr>
<td>MM</td>
<td></td>
<td>7</td>
<td>18</td>
<td>2.560</td>
<td>5</td>
<td>764</td>
</tr>
<tr>
<td>AllAppl</td>
<td></td>
<td>1</td>
<td>0</td>
<td>427</td>
<td>0</td>
<td>150</td>
</tr>
</tbody>
</table>

**Figure 7:** Sample CCMS report showing tcode profile aggregation at the application. From this report the number of dialogue steps, CPU time, requested kBytes are helpful to derive and apportion a “dialogue” step transaction profile.

| Application | DB requests total avg. Seq. total avg. Changes total avg. Phys. reads Phys. chges Buffer reads |
|-------------|---------------------------------------------------------------|-------------------------------|---------------------|----------------|-----------------|-----------------|
| *TOTAL*     | 92.883                                                        | 5.9 ms                        | 78.323              | 4.2 ms         | 1.406           | 56.5 ms         | 26.985           | 2.884           | 7.466           |
| Shared      | 1.640                                                         | 1.337                         | 815                 | 17              | 101             | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| Unknown     | 1.008                                                         | 658                            | 652                 | 172             | 170             | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| BC          | 606                                                           | 242                            | 399                 | 82              | 135             | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| AM          | 523                                                           | 138                            | 263                 | 50              | 95              | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| SD          | 245                                                           | 95                             | 388                 | 35              | 142             | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| FI          | 209                                                           | 57                             | 273                 | 17              | 81              | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| MM          | 7                                                             | 18                             | 2.560               | 5               | 764             | 56.5 ms         | 433              | 1.796           | 160             | 33              |
| AllAppl     | 1                                                             | 0                              | 427                 | 0               | 150             | 56.5 ms         | 433              | 1.796           | 160             | 33              |

**Figure 8:** Sample CCMS report showing tcode profile aggregation I/O activity counters which can be used to establish a dialogue step I/O profile and apportion UNIX reported hdisk utilisation.