

Oracle Workload Measurement

Andy Rivenes
AppsDBA Consulting

1. Abstract

This presentation will explore Oracle workload measurement and how it relates to Oracle system management. Our profession confuses the processes of performance tuning and workload management and this presentation will try to draw out the distinctions between the two. It will also address what "system performance" really means and then will explore how workload measurement can be used and what economic benefits can be achieved. Real world examples will be used to illustrate how workload measurement can be used as inputs to capacity planning and workload optimization. Although not a "tool" presentation, a couple of options will be addressed to allow the audience to implement workload measurement in their environments.

2. Introduction

This paper is about Oracle workload measurement. Workload measurement is fundamentally different from performance tuning. In my opinion one of the big debates in the database community centers around the approach to performance tuning. There are those who believe performance tuning is best performed by starting at the "system" level and moving down¹ through the layers to find the bottleneck(s), and there are those that believe that it is more efficient and accurate to address performance problems by starting with the problem itself, as in slow end user response time for a particular task. Where the debate seems to get bogged down is in the desire to have a one size fits all approach. In other words, most DBAs seem to view workload management and performance tuning as the same thing. However, workload management and capacity planning are not "performance tuning". So what is performance tuning? Performance tuning is the optimization of an end user process to meet a specific business need. This definition creates a measurable goal and is specific enough to understand. Where then, does that leave "system performance"? Of course the answer is workload management. Workload management leads to a process of understanding a "system's" capacity and workload by measuring it. If a system has excess capacity, then the waste caused by sub-optimal workloads may not be a problem. If the impact on business processes is within service level requirements then it may not be economically beneficial to correct them. On the flip side of this is the system where the workload consists of highly optimized jobs, but has simply outgrown the resources available. Without workload

¹ Or up depending on who you're talking to.

measurement, there is no way to know where you've been and where you're heading.

So, let's get back to the issue of approach. Is looking at "system" level "performance" wrong? No, it's just misguided. This paper will attempt to show that at the system level it's not performance tuning, it's workload management. Workload management can be used to balance workload through work shifts, target optimization efforts based on existing resource utilization, and provide input to capacity planning. The arguments about whether ratios are valid or whether wait based tuning is the right approach become easier to sift through if one can accept that optimization occurs at the session or transaction level and workload management occurs at the system level. Together they form a symbiotic relationship that becomes a part of Oracle system management.

With workload measurement I'll address what the "system" is, the types of workload that are generated, and how to measure and quantify that workload. I will also attempt to address some of the economic benefits of workload measurement along with its limitations. You'll notice that space usage and growth have been left out of this paper. Although important to capacity planning and general database administration, when measuring workload it's the file I/O rates that we're concerned with and not the storage patterns.

3. Capacity Planning, Throughput and Workload

Capacity planning is the process of *predicting* when future load levels will *saturate* the system and determining the most cost-effective way of delaying system saturation as much as possible [Menasce and Almeida (2002), 12-13]. However, most capacity planning efforts seem to focus on new system rollouts and whether there is enough hardware to support the application. Capacity planning should be an ongoing process, and part of the input to a good capacity planning methodology is past workload.

Let's define some terms. In the Practical Performance Analyst, Neil Gunther defines utilization to be the time the system or resource was busy during a measurement period [Gunther (2000), 42]. He further defines throughput as a direct measure of the number of completions [Gunther (2000), 42]. Capacity then, is the maximum amount of "service" that a system or resource can perform, and workload is the amount of capacity that a system or resource consumes. Therefore, since we can quantify the amount of service time an Oracle system or resource consumes, and the throughput based on that service time, we can begin to make capacity predictions as we observe these quantities over time.

In computer based systems, system capacity is generally categorized into four areas: CPU, memory, I/O subsystem, and network. In an Oracle database, workload is typically motivated by the execution of SQL, but can also be created

by Java and PL/SQL programs that may or may not invoke SQL calls to the database. In all these cases CPU is required to perform the work, with memory, I/O, and network components supporting the production of this work. Therefore, workload can be considered as the measurement of the “rates” of work being performed. Service time is "... the total amount of time required by a transaction during its execution at the resource" [Menasce and Almeida (2000), 238]. From this definition, we can see that service time is not limited to just CPU time (e.g. YAPP definition), disk I/O is also a form of service time. In fact, any resource can provide service time and since resources have finite capacity they can also be sources of queue time. In fact both Cary Millsap [Millsap (2003)] and Craig Shallahamer [Shallahamer (2004)] spend time describing this in detail.

The ability to measure workload rates has been instrumented into the Oracle database through various system and session level "statistics". These statistics are, in general, fairly accurate and it will be shown that they can be used to develop a good characterization of database workload. One way of verifying this accuracy, something that will be shown later for CPU service, is to use operating system measurements as a check against the Oracle statistics.

4. Workload Measurement

If workload is really the rates of work being performed, then in an Oracle database there are two basic types of rates to be measured. The first is CPU rates and the second is file I/O rates. When considering workload measurement service times, or rates, we are not concerned with synchronization issues or queueing. At the system level it is not possible to accurately measure wait events, and we'll explore this further in a later section. In the Oracle database, the statistic “CPU used by this session” in `v$sysstat` can be used to capture total CPU service time. There are also other CPU related statistics that can be used to further breakdown CPU time into sub-components [Kolk and Yamajuchi (1999)].

When considering file I/O rates, there are really two types of I/O to be considered, and they are measured differently in the Oracle database. Redo log file I/O and data file I/O are the key areas to consider because they directly affect foreground processes and the work being performed. There are other types of file I/O that are performed, specifically control file updates and archive log file creation, but in most cases their volume and affect on user transactions are not significant, and can usually be accommodated through placement tuning and operating system I/O measurements.

The online redo log files exist to allow for recovery of the database in the event of an instance or media failure. They provide a record of all changes made to the database, and as a result they can also be a performance bottleneck since any committing transaction must wait for it's redo information to be flushed from the in memory log buffer to the disk based online redo log file. This is also a major area

of Oracle file I/O writing not done in the background². The executing transaction, when it commits, incurs a foreground wait until its information is flushed to disk. This directly affects transaction time. For this reason, it is very important to insure that LGWR can flush the log buffer sufficiently fast enough to accommodate system workload and meet service level requirements.

Data file writing is typically done in the background (e.g. DBWR writing changes to datafiles) and reads have the advantage of being cached in the database buffer cache. Data file I/O also generally constitutes the largest amount of I/O performed by the database.

Oracle records redo log and data file I/O differently. Redo log file I/O is captured in statistics in v\$sysstat, and database file I/O details are captured in the v\$filestat and v\$tempstat views, although summaries are also reported in v\$sysstat. For data files Oracle will record additional file I/O timings as well as I/O calls and amounts.

There are of course, other types of information that may be of interest as well. Things like number of users connected to the database, number of commits completed, sorting to memory and disk, and various other statistical information may be helpful in characterizing the type(s) of workload that are occurring.

5. Economic Benefits

Are there economic benefits to workload measurement? Since capacity planning requires past workload as an input if it is available, and since capacity planning is critical to optimizing the allocation of resources to insure adequate performance, I believe that there are economic advantages to workload measurement. The primary motivation of capacity planning is to only purchase enough capacity to service peak workloads, and not over buy and therefore waste money on excess capacity. If there is a risk to the business of not having enough capacity then there is a benefit to the business in knowing that there will be enough capacity to meet workload demand. Another benefit from measuring workload on a constant basis, is that it becomes possible to quantify the results of response time optimizations in the savings of system resources. Since this can help justify delaying or eliminating the need for purchasing new equipment there is an economic benefit to the business.

Without venturing into the performance tuning debate, there is also another case where workload measurement can help quantify, and therefore justify, the optimization of database workload in the absence of direct end user motivation. When a system has enough capacity to satisfy the current workload, it is possible for sub-optimal work to occur unnoticed. This may be the result of batch

² I'm ignoring direct I/O for sorting and data loading for the sake of simplicity.

workloads, end user tolerance, or simply overhead processes (e.g. the PeopleSoft scheduler and the Oracle Applications concurrent manager come to mind). The question becomes, is it economically feasible to fix obvious problems that might not be directly affecting business processes? Workload measurement can help answer this question because it becomes possible to quantify the benefit of the workload reduction and then translate that into a percentage of capacity savings.

But wait a minute. We're talking about reducing the workload for a resource. Is that the same as an upgrade? Will we get into the trap of increasing the competition for a non-bottleneck resource and possibly degrade our system's performance [Gunther (2000), 117-122]? Of course not, because by reducing workload we will always move to the left of the performance curve by reducing the competition for all system resources. See figure 1 for an M/M/1 system. Clearly if we reduce workload (e.g. the arrival rate of work or requests) we will move to the left of the curve as our utilization drops, and we will have greater capacity than we had before. "Results from eliminating unnecessary workload often result in better-than-linear performance improvement." [Millsap (2004)]

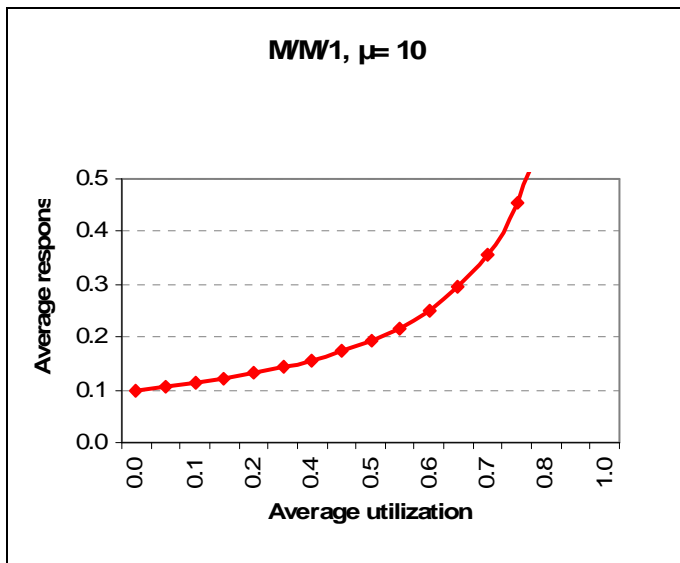


Figure 1. M/M/1 System

6. Rates Versus Waits

The YAPP paper [A. Kolk, S. Yamaguchi (1999)] helped quantify that in an Oracle database, a user's response time is equal to their service time plus their wait time. There are at least three problems that occur when this definition is applied at the system level. The first is that at the system level one cannot measure response time. Response time is the measure of the duration of some action. Typically response time is represented as $R = S + W$, where R is response time, S is service time, and W is queue time [Gunther (2000)]. The

problem with this definition at the system level is that there are a multitude of actions taking place in many service centers. To make matters worse, requests will overlap and happen in parallel. How can you possibly associate response times with each action, especially if you're looking at aggregated data (e.g. `v$sqlsystem_event`)? The answer is that you can't.

In release 10g Oracle has taken a slightly different approach. With the Automatic Workload Repository, Oracle attempts to establish a separate time line for each session, and then correlate each session's response time components. The sum total of all these time lines is then the system "response time". Oracle calls this "Time Model Statistics". Craig Shallahamer proposes the same kind of idea in his paper "Response Time Analysis for Oracle Based Systems" [Shallahamer (2004)]. Unfortunately this doesn't help us with our workload measurements because we're not interested in individual response times. We're trying to measure the total rates of work being performed. As far back as the early Oracle8 documentation, the concept of elapsed time representing "response time" at the system level has been recognized. At the system level we can accurately measure elapsed time and total service time if we're using interval based snapshots, but as we will see in the next sections we cannot measure total system wait time directly. We can only derive system wait time based on interval elapsed time and total service time. At the system level though, system wait time just winds up being excess capacity.

The second issue involves the problem of measuring timed events at the system level. The YAPP paper equated wait time with Oracle wait events. However, wait events, or more accurately timed events, cannot be accurately measured at the system level. There is an infinite capacity to wait at the system level [Rivenes (2003), Millsap (2003)], and Oracle does not provide us with the ability to uniquely measure the time that is captured. As I'll show in a later section, this results in multiple counting of these events when they're aggregated.

The third issue with system level analysis of timed events is that there is no time boundary. There is no way to correlate an elapsed time with the duration of the timed events. In other words, in order for the event to have any context, you have to know the total elapsed time associated with the session that contributed the timed event. Without that information you have no ability to associate the "significance" of the event. There is no way to create an accurate system level resource profile.

6.1 System Level Rates

In the Oracle database there are multiple types of services and associated rates. In fact, many wait events are really their own service channels³. These so called

³ The term service channel is used to denote some resource that provides service (e.g. CPU or a disk drive)

wait events are only "waits" from the perspective of the process that is running. Work is being performed (e.g. service time is being consumed) and then a "wait" is encountered until some "event" occurs that will allow continued processing. At the system level though, many of these events involve service time to access other resources. Disk I/O is probably the best example. A wait event on "db file sequential read" is a wait while an I/O call is issued for a database block to be read into the buffer cache. This event measures resource service time and queue time, and is really a timed event at the system level. The amount of I/O generated is a rate. Since there are many types of I/O that occur in an Oracle database there are also many types of I/O rates that can be measured.

Since service time is not limited to just CPU time, the key to measuring system workload is to understand that it's the "rates" of work that are being evaluated, and that in order to measure rates there must be an ability to accurately distinguish between service time and queue time.

6.2 System Level Wait Time

Wait time is not a finite resource at the system level. For any given session, response time will be composed of service time plus wait time. At the system level elapsed time will replace response time, but wait time cannot be derived directly. Oracle will record total wait times, through the `v$system_event` view, for all events that sessions waited on. Wait time in this context is infinite. This occurs because there is no limit to the number of sessions that can connect and disconnect during any interval period. However, each session that is connected during the interval can contribute to the total wait time of the various timed events. This is also true for service time, but as we have said, that resource is finite and as such access is serialized. Not so with timed events.

The effect of this is that while system level elapsed time is still composed of service time plus wait time, only service time and elapsed time can be derived directly. Wait time, because it is composed of all sessions contributing time in a non-serialized fashion, will be the sum of all session elapsed times minus service time. At the system level this number is of no use. Another way to look at it is that each session's elapsed time parallels the system elapsed time. The example in figure 2, based on a one CPU system, attempts to show what happens. Notice that the sum of each session's elapsed time and wait time, from a system perspective, is greater than the system's total elapsed time and wait time. This is what happens when looking at the totals for wait time in `v$system_event` versus the total CPU time from `v$sysstat` (CPU used by this session) and elapsed time (wall clock).

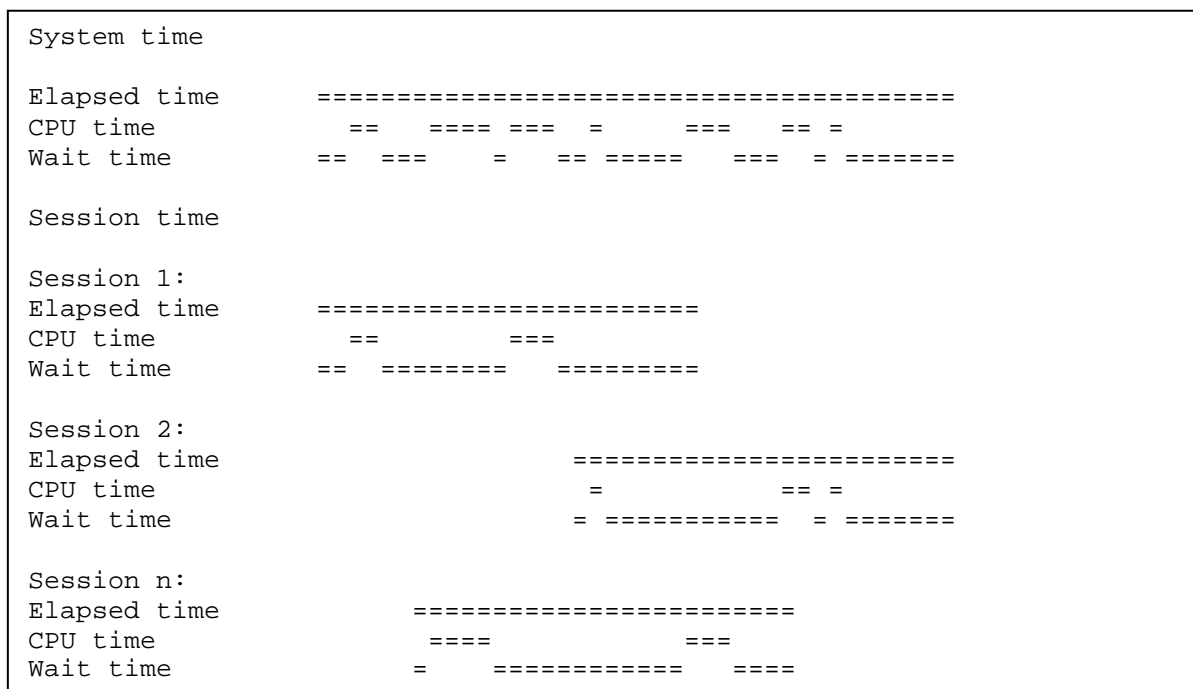


Figure 2. Timing boundaries.

The following will show the four distinct cases that need to be considered when trying to use interval based monitoring at the system level for timed events. Also shown is a fifth case that elaborates on how multiple counting can occur. See figure 3 for each case type for a given time interval bounded by intervals t0 and t1. This behavior has been verified through Oracle 10g (10.1.0.3) and also affects the active session history feature in Oracle 10g (e.g. v\$active_session_history).

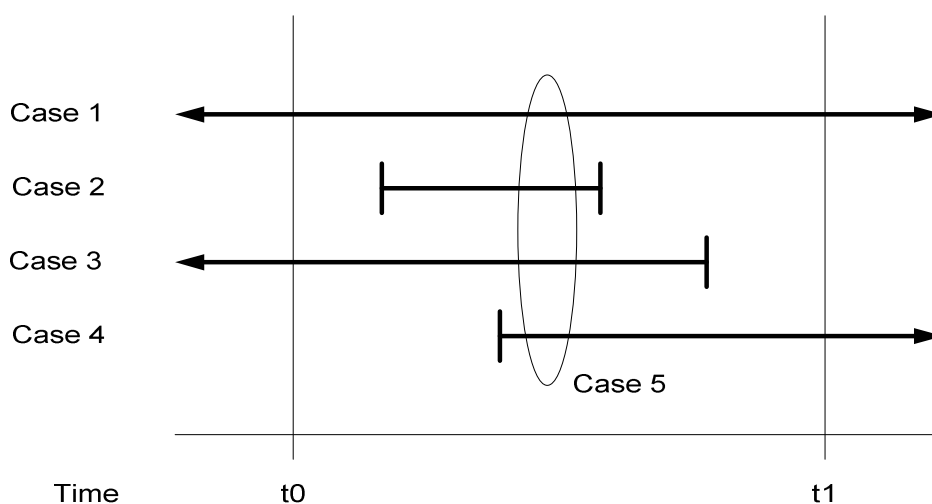


Figure 3. Timing boundaries and multiple counting.

In case 1 we see that the session existed during both time t0 and t1. This is the only case where we can get accurate⁴ timing. We can definitively associate an elapsed time in this case (this assumes we could track it individually of course).

Case 2 shows that the session started and ended between time t0 and t1. This will not show up in v\$session_event at the time of the interval snapshot, but will contribute to events in v\$system_event. We don't know it's elapsed time.

Case 3 shows that the session started before time t0 but ended before time t1. This is like case 2. It will contribute time to v\$system_event but will not be trackable through v\$session_event so no elapsed time can be established.

Case 4 shows a session that starts after time t0 and still exists at time t1. The event time can be captured since all time will be accurate at the t1 snapshot but the elapsed time is not known.

Case 5 shows the overlap that can occur when two or more sessions are waiting on the same event, and therefore the multiple counting that can occur because all wait time will be counted towards the specific event(s) that occur. These examples show that if all of these sessions were waiting on the same resource, which is entirely possible in Oracle – think enqueue, then we would have a portion of each session's event contributing four times to the same total in v\$system_event. The question might be raised, are not multiple waiters for the same resource representative of unsatisfied demand for the resource in question? The answer is sometimes yes and sometimes no. In the case of the timed event "db file sequential read" this may be true if they're waiting on the same disk, but in the case of "SQL*Net message from client", probably not.

So, how serious is this problem? The answer of course is that it depends, but since there are obvious measurement problems and multiple counting of the same events, the answer is that the information is unreliable and therefore shouldn't be used at the system level at all.

7. What is System Performance?

It must be human nature, but everyone always refers to the "system" as being slow whenever there is a perceived problem. In fact, to the end user the "system" is quite different than it is to the database administrator. An end user probably considers everything to be the system from the time he or she hits enter to the time a response appears on the screen, or until they get their report. For our purposes though the "system" will be considered the entire database server. This includes the hardware the database is running on, along with the operating system and peripherals such as storage systems since they are all directly

⁴ This may still not happen for CPU time since long running (e.g. spanning intervals) recursive calls will not report CPU stats until finished.

related to the functioning of the Oracle database. When discussing technical architecture with system administrators I have often said that we are architecting a database server, not a UNIX server. This is usually met with skepticism, but the fact is that Oracle can, and very well may, use an entire machine's resources.

To be precise, an Oracle database system is made up of one or more service channels providing service time with the inherent queueing time and service time capacity based on the arrival rate of requests. These service channels form a system of queues that are synchronized by the Oracle kernel to insure consistent data presentation to business transactions.

What then is system performance? If the system is the database server and performance is defined as response time and throughput that meets desired or acceptable levels [Menasce and Almeida (2000), 216], then how can a system have performance? If we cannot measure a system's response time, then a system's performance can only be measured in terms of its throughput. If throughput is a measure of completions or work performed, and workload is the measurement of the rates of work then system performance can be thought of as the system's ability to provide throughput. In other words, system performance only has context in the framework of measuring its workload. Based on this definition, system performance can also be thought of as the aggregate perception of the users to service level requirements. This can be a useful distinction since it is entirely possible for system throughput to be high, but interactive users to find response time totally unacceptable. Recall that our end user considers everything to be the system from the time he or she hits enter to the time a response appears on the screen.

8. Do Any Tools Exist To Capture Workload?

So, how do we capture workload information? There are lots of tools available, since what we're really talking about is just data from the "v\$" views provided by Oracle. The point is not so much collecting the data, but what you do with it once you've got it. How many times have you read on a newsgroup, or perhaps even witnessed first hand, the DBA who has a database with a problem, runs a Statspack report or perhaps looks at the v\$system_event view, and concludes that their problem is with event x, and wants to know what to do about it? Invariably someone will ask, "Well, what did it look like before the problem started"? Typically the DBA will have no idea because they aren't measuring workload. In the absence of knowing whether the workload has changed, it becomes a difficult call to start diagnosing things that may have nothing to do with the root problem. If the workload has changed, or a specific problem can be identified, then attack the problem at hand with response time optimization. Again, you can't "tune" performance with system level information, you can only manage the workload.

8.1 Statspack

I used the example of a Statspack report above, and there is nothing wrong with Statspack, after all it just collects v\$ data, right? Yes, that's true. However, what's the first thing people look at after they run a Statspack report? It's usually the "Top 5 Timed Events" section, which is misleading. After all, it's based on system wide timed events, and we've already shown that measuring these events at the system level is unreliable. But wait, Oracle added CPU time in Oracle9i, so doesn't that fix the problem? Well no, because we still don't have a reliable measure of timed event duration, and even if we did, we couldn't tie it back to a particular process. As you may have guessed, this is the wrong information and the wrong process to use.

There always seems to be the argument, "... but I fix problems with Statspack reports all the time." That may be true, but for every success story there are countless failures to identify the real problem with Statspack, or any other interval based tool measuring system wide data. In fact, I would go so far as to say, that unless the bottleneck skews the timed event information as to make it so obvious as to what the problem is, that you're just guessing. So, is Statspack useless? No, but the Statspack report is misleading in several areas. It's definitely not a performance tuning tool because it relies on system wide data, and it's report format is of dubious workload measurement value, because it includes timed event information, which I think I've shown is unreliable.

8.2 10g Automated Workload Repository

Oracle 10g has a new feature, the Automated Workload Repository. This is an interval based snapshot of statistic and timed events that has been built in to the database. New views and packages have been added to the database and a re-worked "report" has been added as well, essentially what was the Statspack report. While all of the Statspack code remains, it is clear that Oracle intends the AWR, which is also integrated into Enterprise Manager, to become its workload measurement tool since all mention of Statspack has been removed from the 10g Performance Tuning guide.

However, AWR still suffers from the same problem of trying to create a resource profile at the "system" level. Notice that in figure 4, which is a section of the new Automated Workload Report called "Time Model Statistics", the percentages are based on a new concept called "DB Time". Oracle defines DB time as:

"The most important of the time model statistics is `DB time`. This statistics represents the total time spent in database calls and is a indicator of the total instance workload. It is calculated by aggregating the CPU and wait times of all sessions not waiting on idle wait events (non-idle user sessions)."⁵

⁵ Time Model Statistics, Oracle Database Performance Tuning Guide 10g Release 1 (10.1)

```

-> ordered by Time (seconds) desc

```

Statistic Name	Time (seconds)	% Total DB Time
background elapsed time	64,201.37	8230.25
DB time	780.07	100.00
DB CPU	766.47	98.26
sql execute elapsed time	693.29	88.88
PL/SQL execution elapsed time	300.85	38.57
background cpu time	273.90	35.11
parse time elapsed	179.37	22.99
hard parse elapsed time	165.02	21.15
PL/SQL compilation elapsed time	18.71	2.40
hard parse (sharing criteria) elapsed time	9.08	1.16
connection management call elapsed time	5.76	.74
Java execution elapsed time	1.58	.20
hard parse (bind mismatch) elapsed time	.95	.12
sequence load elapsed time	.53	.07
inbound PL/SQL rpc elapsed time	.00	.00
failed parse elapsed time	.00	.00
failed parse (out of shared memory) elapsed t	.00	.00

Figure 4. Time Model Statistics section of an Automated Workload Report

The problem of course is that we still have the same problem of session level measurements being applied at the system level that we talked about in the section "[Rates versus Waits](#)". However, at least Oracle has now named the information for what it is, which is "Workload".

There are other problems with AWR which existed with Statspack as well. The first and foremost one being in retention of data. The sheer volume of data gathered may be too large for any practical long term storage. Oracle states that

"By default, the snapshots are captured once every hour and are retained in the database for 7 days. With these default settings, a typical system with an average of 10 concurrent active sessions can require approximately 200 to 300 MB of space for its AWR data. It is possible to change the default values for both snapshot interval and retention period."⁶

They go on to state that it may be possible to reduce space consumption by measuring less data, less frequently. The bottom line appears to be that if you want to use AWR for effective long term workload measurement that you will have to tailor it to your storage needs.

9. Measuring Workload

Although I've specifically avoided making this a "tool" paper, the discussion of workload measurement can benefit from some concrete examples and a real world "how to". With that in mind, are there any tools that capture workload

⁶ Automatic Workload Repository, Oracle Database Performance Tuning Guide 10g Release 1 (10.1)

information? Sure, but most suffer from the deficiency that they aren't designed to handle long term data storage. Take the discussion of Statspack and AWR in the previous sections for example. The problem with these tools is that they capture each interval of data essentially in the same way that they are stored in the database. In other words you get a row for every statistic in v\$sysstat. This results in a large volume of data that is cumbersome to summarize over multiple intervals. This is not to say that it's not possible, and there are people who do it, but it either requires extra work or lots of space.

Another approach is to process each interval up front and only store the interesting information, and do it in a "wide" or de-normalized approach. The advantage of this method is that it makes it much easier to create reports on wide date ranges (e.g. large interval periods as in weeks or months). This is the approach taken in SYSMON⁷.

There are certainly other tools available as well, but the point is to capture the information, not how to do it. The examples in the following sections show the types of data that can be captured. All data was captured using the SYSMON utility off of a real production database.

9.1 CPU Utilization

At the system level CPU capacity is a finite resource. There are a finite number of CPU instructions that can be executed by a CPU in a given amount of time. There are also a fixed number of CPU(s) available on a given machine at any given point in time. Therefore, the maximum possible CPU service time is the elapsed time times the number of CPUs. Any unused CPU time is measured as wait time at the system level. In an Oracle database CPU service time can be measured for a specific time interval with the v\$sysstat statistic "CPU used by this session". Although it may not be accurate enough for response time optimization [Millsap, 2003] it is almost always accurate enough over longer interval periods for workload measurement. When coupled with corresponding OS data, CPU measurements can be used with a high degree of confidence in measuring system workload.

9.1.1 Granularity

"Amount of CPU time (in 10s of milliseconds) used by a session from the time a user call starts until it ends. If a user call completes within 10 milliseconds, the start and end user-call time are the same for purposes of this statistics, and 0 milliseconds are added."⁸

⁷ SYSMON is an AppsDBA Consulting workload measurement tool available at www.appsdba.com/utilities_sysmon.htm

⁸ Oracle Database Reference, 10g Release 1 (10.1), Part Number B10755-01

This tells us that it is possible that CPU time could be missed. It turns out that at the system level this does not seem to be much of a problem. Since over the long run the measurements, both high and low, seem to balance out. Cary Millsap attempts to explain this behavior with a phenomenon known as "quantization error" [Millsap (2003)].

9.1.2 Timing Boundaries

Another problem with the reliance on CPU statistics in Oracle is that Oracle only updates the statistic 'CPU used by this session' when a user call completes. This means that long running queries and PL/SQL processes, which use CPU, will NOT show up in the CPU statistic until the query or outermost PL/SQL block completes. This can therefore skew a given interval period if one or more calls complete that spanned one or more previous periods. The previous periods will also have lower times than they should. This can be detected to some extent by the statistic 'CPU used when call started'. Oracle will mark the CPU time used when a call starts in this statistic. Unfortunately this statistic is only of value at the session level, since the system level statistic is updated every time another session starts a call.

Another issue to consider is that the init.ora parameter "resource_limit", when set, may have the side effect of causing "CPU based statistics to be updated more frequently than usual in order to ensure CPU resource limits are not exceeded."⁹

The following shows an example of what can occur with the statistic 'CPU used by this session' when a call crosses interval boundaries. This example shows the use of a PL/SQL loop to generate CPU usage.¹⁰ This example was originally run back in 2001 against an 8i database, but the test has been verified against a 10.1.0.3 database as well. Two database sessions were used, one to run the test with before and after statistics shown, and a second to observe the first session's statistic values during the test. Notice that the CPU values are not updated until the procedure completes.

⁹ Oracle Corporation, Note: 30800.1, "Reference Note for Init.Ora Parameter "RESOURCE_LIMIT", this note is old and may or may not still apply.

¹⁰ The scripts used, `cpu_bounday.sql` and `cpu_boundary_system`, can be found at appsdba.com.

Session View:

Notice that the stats query requires approximately 30 100ths of a second.

SID	Time Stamp	Stat Name	Value
37	10/12/01 08:38:42	CPU used by this session	7050
37	10/12/01 08:38:45	CPU used by this session	7080

SQL> @cputest

SID	Time Stamp	Stat Name	Value
37	10/12/01 08:39:01	CPU used by this session	7109

PL/SQL procedure successfully completed.

SID	Time Stamp	Stat Name	Value
37	10/12/01 08:40:17	CPU used by this session	14133

System View:

SID	Time Stamp	Stat Name	Value
37	10/12/01 08:38:58	CPU used by this session	7109
37	10/12/01 08:39:06	CPU used by this session	7139
37	10/12/01 08:40:11	CPU used by this session	7139
37	10/12/01 08:40:19	CPU used by this session	14163

* Notice the change as the procedure completes.

What this tells us is that when monitoring CPU usage we need to be careful to monitor the entire period, or elapsed time, to minimize the possibility of missing the statistics of processes that have not completed or that started before our measurements began.

Figure 5 shows the CPU utilization for a database averaged over a 30 day period. Both database CPU, as measured from the Oracle statistics, and OS CPU as measured with "sar" are shown for each day. As you can see, the two statistics are very close and provide a nice check on accuracy. The overall downward trend is due to a workload reduction project that was in progress at the time of the measurement.

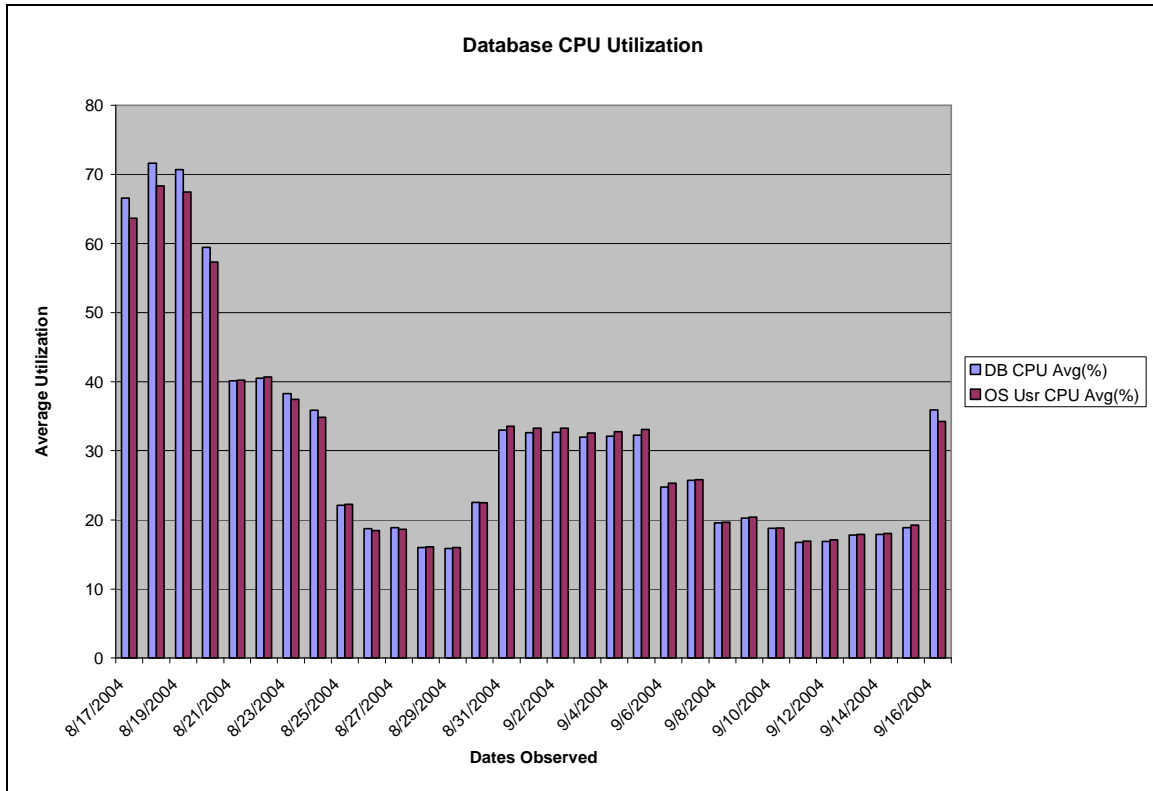


Figure 5. Database CPU utilization measurement.

Figure 6 shows total database CPU usage plotted against OS user mode CPU usage over a several month period. Notice that in general, the two match up very closely and that the database measurements show a pretty accurate picture of CPU usage on this database server.

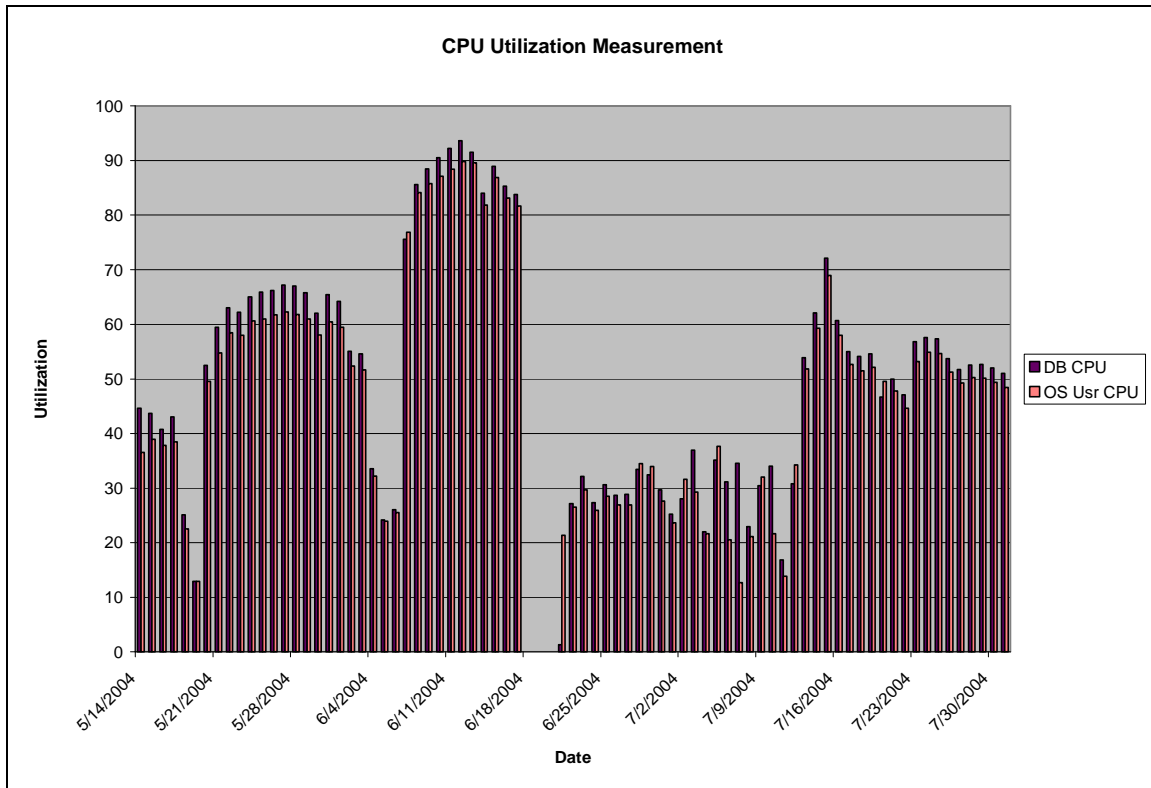


Figure 6. Historical CPU utilization trend.

Figure 7 shows a break down of CPU usage for individual intervals (e.g. hourly for one day). This can help illustrate when peak periods of utilization occur. In other words, if the daily average utilization is 32.18%, but peak periods get as high as 94.61%, then CPU saturation may be occurring that would be missed if looking at just daily averages. And depending on the type of workload, see [Types of Workload](#), this may occur at pretty much any interval granularity.

Interval Date	CPU Elapsed Time(Sec)	Total CPU Time(Sec)	Parse CPU Time(Sec)	Recursive CPU Time(Sec)	Other CPU Time(Sec)	DB CPU Avg Util(%)
6/23/2004 0:50	28,904	8611.06	3.74	22.36	8584.96	29.79
6/23/2004 1:50	28,920	8469.10	3.92	20.39	8444.79	29.28
6/23/2004 2:51	28,904	8457.01	2.38	19.71	8434.92	29.26
6/23/2004 3:50	28,416	8240.44	1.97	19.18	8219.29	29.00
6/23/2004 4:50	28,896	8062.04	1.36	18.13	8042.55	27.90
6/23/2004 5:50	28,904	8038.07	0.96	17.80	8019.31	27.81
6/23/2004 6:50	28,896	8038.57	0.96	16.04	8021.57	27.82
6/23/2004 7:50	28,416	7909.93	1.10	16.42	7892.41	27.84
6/23/2004 8:50	28,904	8082.88	1.08	16.33	8065.47	27.96
6/23/2004 9:50	28,896	8108.19	1.49	18.99	8087.71	28.06
6/23/2004 10:50	28,912	8116.23	2.37	17.93	8095.93	28.07
6/23/2004 11:51	28,896	8313.63	1.78	19.49	8292.36	28.77
6/23/2004 12:50	28,424	7945.35	1.10	16.14	7928.11	27.95
6/23/2004 13:50	28,928	8082.06	0.96	16.94	8064.16	27.94
6/23/2004 14:50	28,920	9309.38	1.07	16.63	9291.68	32.19
6/23/2004 15:51	29,032	10723.92	1.08	16.59	10706.25	36.94
6/23/2004 16:50	28,480	11275.53	6.84	23.40	11245.29	39.59
6/23/2004 17:50	28,944	27382.99	14.24	17.27	27351.48	94.61
6/23/2004 18:51	28,944	9729.93	1.11	13.77	9715.05	33.62
6/23/2004 19:50	28,424	7824.25	0.89	13.81	7809.55	27.53
6/23/2004 20:50	28,896	7975.59	1.08	14.03	7960.48	27.60
6/23/2004 21:50	28,896	7951.50	0.99	14.25	7936.26	27.52
6/23/2004 22:51	28,896	7945.91	1.01	14.90	7930.00	27.50
6/23/2004 23:50	28,424	7782.37	0.97	14.58	7766.82	27.38

Figure 7. Single day CPU interval usage.

9.2 I/O Rates

Database workload is composed of more than just CPU usage. As stated at the beginning the I/O subsystem is another area of workload that occurs in an Oracle database. When considering I/O workload, we're typically most concerned about tracking database file I/O and redo generation since this makes up the bulk of the I/O call and transfer rates, and has the largest effect on I/O capacity. The following charts show the trends on a real Oracle database over a several month period. During this period several attempts were made at reducing workload by "tuning" specific processes. This workload information helped show the results on the entire "system" in addition to the fact that the specific processes that were optimized had shorter run durations.

Figure 8 shows the general trend of database file I/O. While this may be most useful for a very high level view, it does give an interesting picture of overall file I/O volume. For effective capacity planning I/O call and transfer rates must be transferred down to the device level, but the point is that this data can be available if the database workload is being measured.

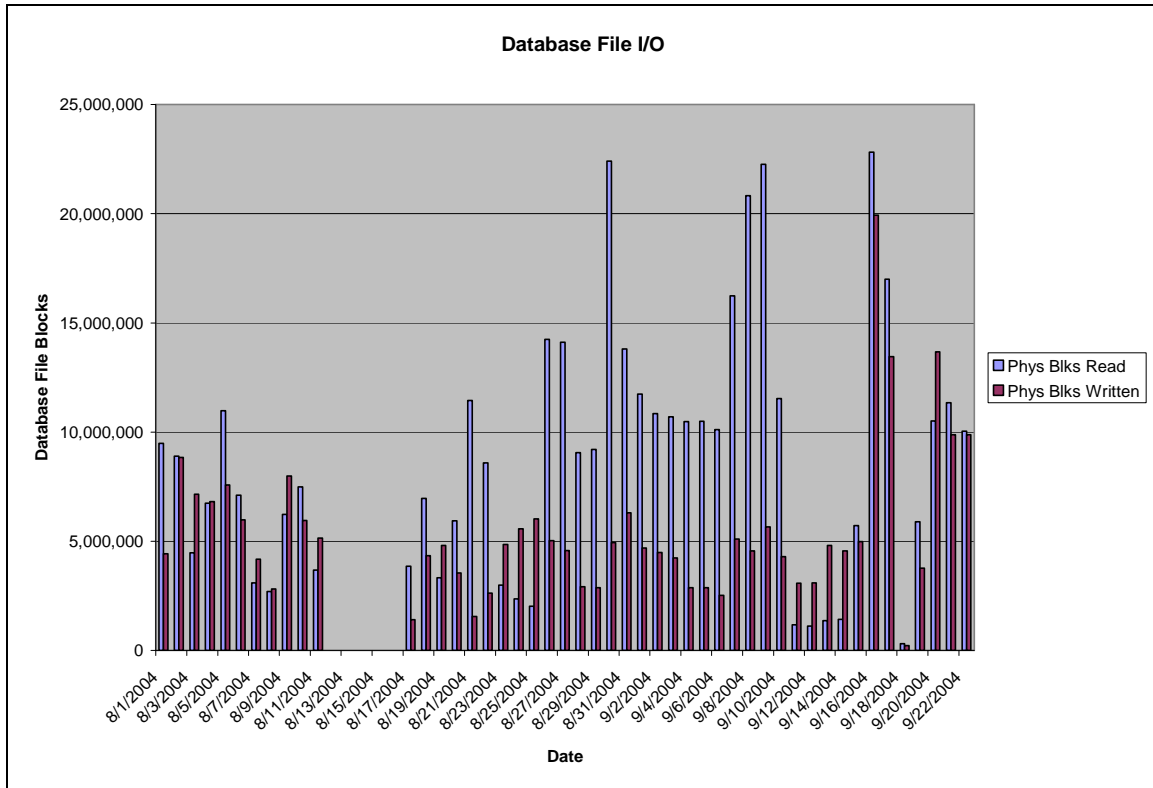


Figure 8. Database file I/O rates trend

Figure 9 shows redo generation rates. This information can also be further broken down into rates that can be used to properly size and track the online redo log I/O sub-system. It is important to capture this information since it is entirely possible for the online redo generation to "heat"¹¹ an I/O sub-system and cause I/O bottlenecks for the rest of the database. This information is not available in the v\$filestat views and is sometimes overlooked. It must be mined from v\$sysstat info.

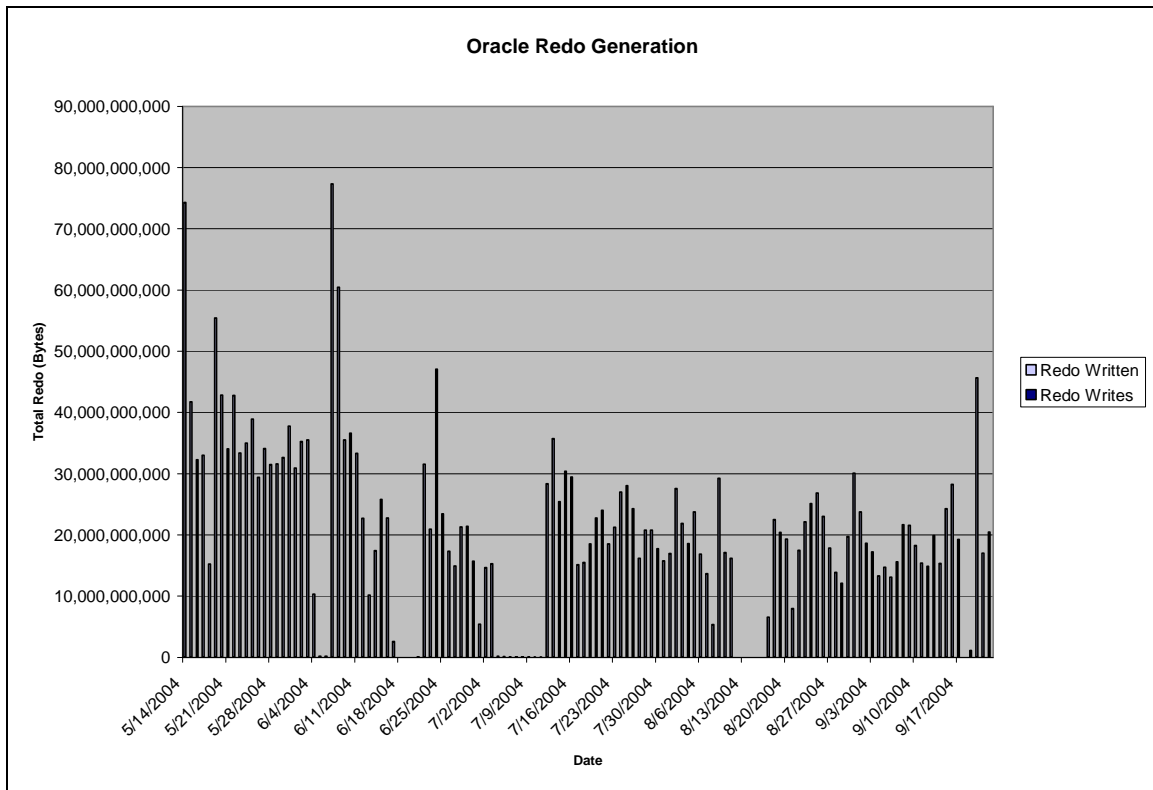


Figure 9. Redo generation trend

Figure 10 shows an interesting view of file I/O throughput for physical block reads and writes, by hourly interval, for each file system. This graph uses an Excel pivot table to generate the stacked bar graph for all file systems. This chart can help illustrate how file system load is occurring.

¹¹ I first heard this term used by Mark Farnham to explain how a few hot spots can cause the whole I/O system to be impacted.

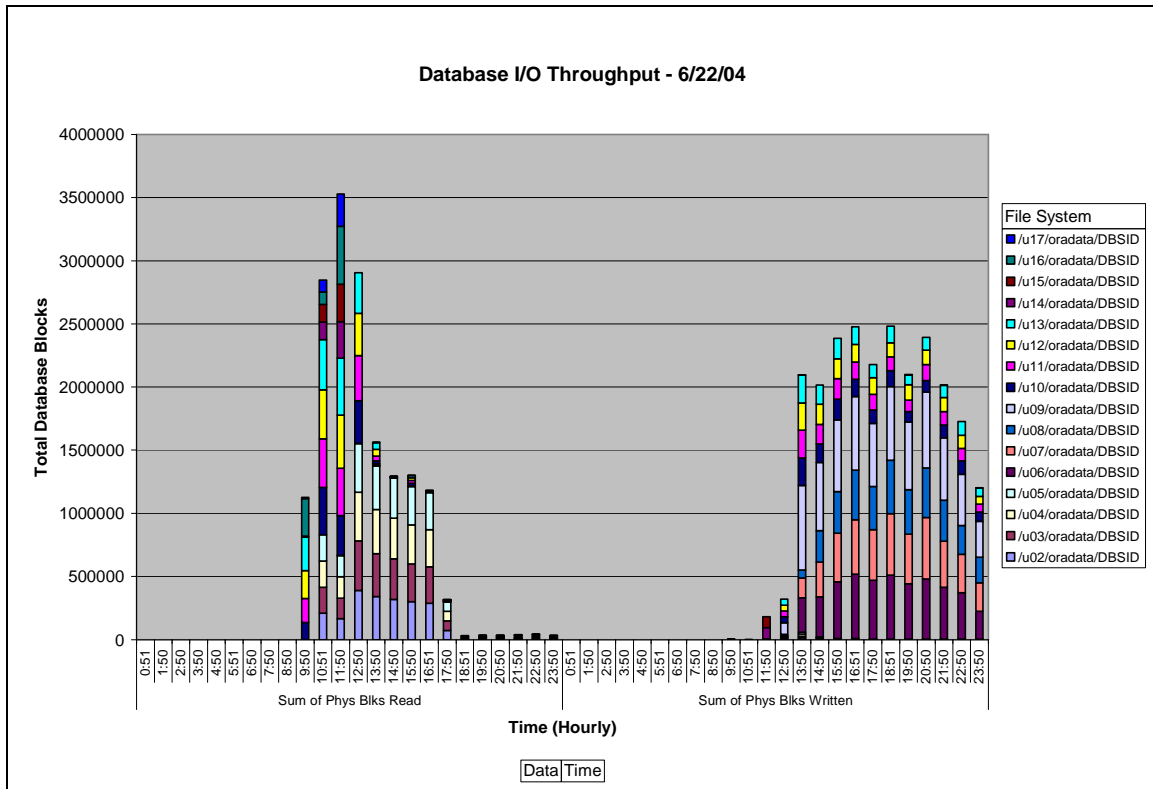


Figure 10. Database File I/O Throughput

10. Supplement To Method R

The following section will propose a supplement to Method R. For review, Method R from the hotsos.com web site:

"... using the approach that we call Method R:

1. Select the top user actions for which the business needs improved performance.
2. Collect properly scoped diagnostic data that will allow you to identify the causes of response time consumption for each selected user action while it is performing sub-optimally.
3. Execute the candidate optimization activity that will have the greatest net payoff to the business. If even the best net-payoff activity produces insufficient net payoff, then suspend your performance improvement activities until something changes.
4. Go to step 1."¹²

At the "system" level however, it is possible to have "excess" workload that is not directly impacting business processes, but is wasting system utilization. Worse yet, this latent workload will only make an impact at the worst possible time,

¹² Hotsos, <http://www.hotsos.com/education/method-r.html>

when system utilization peaks. A question was asked in the [Economic Benefits](#) section, "is it economically feasible to fix obvious problems that might not be directly affecting business processes?" The classic answer is no. After all, the costs of the machine(s) have already been paid, so wasting some resource that has excess capacity isn't costing the business anything additional. For that matter, having DBAs fixing inefficient SQL that isn't impacting business processes isn't costing the business additional money either unless they're contractors, working overtime, or neglecting their other duties. The cost of their salary is relatively fixed as well. So how do we deal with this dilemma? Is it irresponsible to reduce workload in the absence of a clear impact to business process throughput?

I believe that if the effort to reduce the workload is less than the cost of that workload to the business in the future then it is a responsible endeavor. We can look at two issues here. The first is the cost of additional hardware as the capacity of the current hardware is exhausted. More importantly though is the cost to the business during peak workloads for the loss of the utilization that the latent workload is consuming. This is utilization that would have been available to the system during the peak workloads. In other words, if there is a 20% utilization waste on a system that averages 60% utilization, does it matter if the average utilization drops to 40%? Perhaps not, but what about a 20% utilization waste on a system that has now grown to an average utilization of 80%? Since most computer purchases have large upfront costs, any delay in incurring these costs is usually an advantage to the business.

This is my motivation in proposing a supplement to Method R for workload management. The premise is that it's always OK to reduce workload. So, in the absence of end user complaints¹³:

1. Identify the most wasteful, previously unoptimized workload over a workload cycle.
2. Measure the *benefit* of reducing or eliminating that workload waste versus the *cost* of reducing or eliminating it.
3. If economically beneficial then fix it, otherwise stop until something changes.¹⁴
4. Repeat step 1.

The key to the Method R supplement is to define the workload cycle(s) and create a simple methodology to estimate the costs of both the workload waste and the fix to remove the waste. In identifying wasteful workload and the cost to fix it, it may also be beneficial to consider the current system workload and take into account work with a high usage of the resource with the least amount of

¹³ This implies that there is currently no over utilized resource (e.g. there is "headroom" on all resources).

¹⁴ Thanks to Jared Still and Carel-Jan Engel for pointing out that Step 3 should have an explicit "stop" to avoid the desire to slip into Compulsive Tuning Disorder.

remaining headroom. It may be the case that there is an opportunity to consume less of a resource with little headroom even though the total "cost" may increase with additional use of resources having higher headroom.

One of the common causes of latent workload is where a relatively efficient query is executed so many times that it uses a lot of resource when considered in total, or in a system wide context. This may be a case where making the query more efficient, or just running it fewer times, results in a huge savings. The following example will hopefully illustrate this phenomenon.

This story begins with an initial investigation of why interval level chained row counts (i.e. the "table fetch continued row" statistic) were so high on a system I was asked to manage, see figure 11. This information was gathered at the system level and therefore was part of a workload characterization of the system. At this point I was simply making a workload analysis, there was no user complaint or business need driving my investigation.

```

SYSMON Rates and Waits Database Report

Interval: 18-AUG-2004 00:52 - 23:52

< information deleted >

Statistics By Interval:

```

Intvl End	Tot Conn	Rdo Wt [~0]	LibC [<1%]	Disk Sorts	DBWR [<1%]	Cnt Row [~0]	Buff Wt [~0]	User Commits
01:52	422	0	.00	0	5.00	25,634,788	596	900
02:53	422	0	.00	3	5.00	24,302,813	1,101	878
03:52	423	0	.00	3	3.41	24,518,068	797	933
04:52	423	2	.00	0	6.29	23,949,996	699	992
05:53	423	0	.00	0	6.20	25,318,348	1,180	944
06:52	423	0	.00	0	4.17	24,169,216	422	868
07:52	423	5	.00	0	5.50	24,748,742	486	888
08:52	426	0	.00	0	3.73	24,474,731	443	1,158
09:52	423	22	.00	1	4.79	23,162,886	594	1,228
10:52	423	2	.00	1	4.29	23,019,521	381	1,172
11:52	423	3	.00	2	3.67	23,662,244	574	1,199
12:52	423	11	.00	2	3.67	23,071,683	992	1,203
13:53	423	0	.00	1	3.62	23,187,165	519	1,250
14:52	422	2	.00	2	3.75	23,166,212	374	1,286
15:52	422	7	.00	3	3.96	23,363,204	754	1,266
16:53	421	3	.00	1	3.95	22,736,668	700	1,283
17:52	421	4	.00	5	4.08	22,830,318	559	1,219
18:52	420	0	.00	1	4.48	23,090,924	1,023	1,248
19:52	420	3	.00	0	4.60	23,437,407	387	1,300
20:52	420	0	.00	0	4.07	23,850,547	399	1,352
21:52	420	2	.00	0	4.66	24,052,979	431	1,314
22:53	420	4	.00	0	3.89	24,015,585	276	1,274
23:52	420	0	.00	1	2.98	23,416,752	344	1,226

Figure 11. Workload Summary excerpt

At this point the investigation changed from workload analysis, and into a classic performance tuning exercise. Workload measurement had simply illuminated a high rate of chained row accesses. A search of active sessions contributing

values to this statistic quickly identified the problem table. However, in reviewing the SQL to identify the table(s) being accessed, I also noticed unusually large execution counts and buffer gets, and this led to reviewing the most expensive SQL currently in the system with the following SQL statement:

```
select EXECUTIONS,
       BUFFER_GETS,
       DISK_READS,
       ROWS_PROCESSED,
       CPU_TIME,
       ELAPSED_TIME,
       HASH_VALUE
from v$sqlarea
where buffer_gets > 100000
order by BUFFER_GETS,
         CPU_TIME
```

This statement generated 7,161 rows with the following being the largest:

EXECUTIONS	DISK_READS	BUFFER_GETS	ROWS_PROCESSED	ADDRESS	HASH_VALUE	CPU_TIME
75816	3584526	1655868594	750	00000005BCB401D0	2314851399	2.6400E+11
1009436	0	1690376983	1687348669	00000005DB24D700	229119125	2.2929E+10
72491	3393689	1949629162	732	00000005BFAF37A8	2550865707	2.5962E+11

As you can see, statement 2550865707 produced the largest number of buffer gets and was executed 72,491 times returning only 732 rows. Further investigation showed that the text of the SQL involved the updating of an informational table (i.e. a table that the application was using to track loading information). This was in fact the same table that initiated the chained row investigation. In reviewing the SQL from some of the other statements, this table showed up in almost all of them. Further review of the captured data showed that there were two variations of SQL statements and they could benefit substantially from a change in indexing. This was captured with an extended SQL trace of one of the processes accessing the table in question. From the trace file it was clear that the optimizer was favoring full table scans rather than using an existing index. When the column ordering was reversed on the primary key and an additional index was added with the second column from the primary key as the leading edge a dramatic improvement occurred. Additional columns were also added to this second index to allow that query to be satisfied entirely from the index.

Workload measurement now allowed me to quantify the savings. Figure 12 shows a substantial reduction in CPU utilization and logical reads after the changes were made on the 24th of August.

Date	Total Elapsed Time(Sec)	CPU Elapsed Time(Sec)	DB Logical Reads	DB CPU Time(Sec)	DB CPU Avg(%)	OS Usr CPU Avg(%)	OS Usr CPU Min(%)	OS Usr CPU Max(%)
08/18/04	86,352	690,816	7,902,803,855	494,594	71.60	68.33	61.00	73.00
08/19/04	86,431	691,448	7,903,129,277	488,810	70.69	67.46	63.00	72.00
08/20/04	86,395	691,160	7,874,800,107	411,003	59.47	57.33	41.00	67.00
08/21/04	86,382	691,056	7,604,332,092	277,520	40.16	40.21	39.00	42.00
08/22/04	86,398	691,184	7,667,811,263	279,899	40.50	40.67	39.00	43.00
08/23/04	86,405	691,240	5,173,811,098	264,530	38.27	37.42	22.00	52.00
08/24/04	86,389	691,112	3,823,823,589	247,723	35.84	34.83	22.00	49.00
08/25/04	86,437	691,496	2,272,546,617	152,783	22.09	22.25	15.00	30.00
08/26/04	86,383	691,064	1,568,055,602	129,397	18.72	18.46	16.00	23.00
08/27/04	86,382	691,056	1,700,447,592	130,179	18.84	18.63	16.00	24.00
08/28/04	86,436	691,488	1,506,743,354	110,404	15.97	16.08	16.00	17.00
08/29/04	86,373	690,984	1,500,723,241	109,477	15.84	16.00	16.00	16.00
08/30/04	86,390	691,120	1,664,563,387	155,658	22.52	22.50	16.00	32.00
08/31/04	86,402	691,216	1,978,556,216	228,044	32.99	33.54	32.00	37.00
09/01/04	86,146	689,168	1,816,337,865	224,887	32.63	33.29	32.00	36.00
09/02/04	86,380	691,040	1,809,854,359	225,660	32.66	33.25	32.00	35.00
09/03/04	86,382	691,056	1,868,711,926	221,125	32.00	32.58	23.00	35.00
09/04/04	86,435	691,480	1,831,948,026	221,945	32.10	32.79	32.00	34.00
09/05/04	86,375	691,000	1,830,712,911	222,928	32.26	33.08	33.00	34.00
09/06/04	86,431	691,448	1,724,152,638	171,116	24.75	25.29	12.00	33.00
09/07/04	86,396	691,168	1,691,018,286	177,717	25.71	25.83	12.00	89.00
09/08/04	86,376	691,008	1,756,856,856	135,388	19.59	19.63	18.00	21.00
09/09/04	86,383	691,064	1,854,180,735	139,994	20.26	20.38	19.00	23.00
09/10/04	86,434	691,472	1,719,237,070	129,735	18.76	18.83	16.00	21.00
09/11/04	86,373	690,984	1,471,529,971	115,685	16.74	16.92	16.00	18.00
09/12/04	86,433	691,464	1,535,727,550	116,771	16.89	17.08	17.00	18.00
09/13/04	86,374	690,992	1,607,176,065	122,982	17.80	17.92	17.00	19.00
09/14/04	86,429	691,432	1,533,241,650	123,720	17.89	18.04	17.00	20.00
09/15/04	86,378	691,024	1,666,641,611	130,282	18.85	19.25	17.00	28.00

Figure 12. Workload Reduction

11. Limitations

Are there limitations to workload measurement? Of course. One of the biggest limitations is in the nature of the workload itself and what it tells us about system utilization. The other big limitation is in our ability to measure workload. And finally, understanding that workload measurement cannot help us understand performance. At best it can provide an indicator of performance issues, but even that is problematic at best.

11.1 Types of Workload

The type of workload being measured can affect the ability to analyze capacity. Transactions that are short and bursty in nature may not lend themselves well to an ability to determine true system utilization. It may be that overall utilization is very low, but at the time of critical workload all resources are consumed. There may also be issues caused by a relatively small number of resource intensive tasks as opposed to a large number of low resource usage tasks. Knowing

and/or categorizing the types of workload that occur is needed to accurately assess system utilization.

11.2 Workload Balancing

Workload balancing is the process of re-scheduling work to different time periods. This is typically done by shifting work to periods when demand is lighter. The concept of work shifts has been around since the early days of mainframes, and most scheduling systems have the ability to define work shifts.

In Oracle 10g a new feature has been added to allow the definition of application services. This feature is meant to allow better management of workloads within the database.

11.3 Interval Measurement

Interval measurement is directly related to knowing the type(s) of workload being measured. If intervals being used to measure workload are too long or too short then the ability to accurately identify utilization may be compromised. Intervals that are too short risk affecting system utilization and intervals that are too long risk skewing utilization averages to the point of being useless or even misleading.

Referring to Figure 13, during interval t1 a transaction may be too slow because the CPU is not fast enough but system utilization may be low. During intervals t2 through t5 utilization may be high, but transaction throughput may be perfectly acceptable. The key is in knowing the workload.

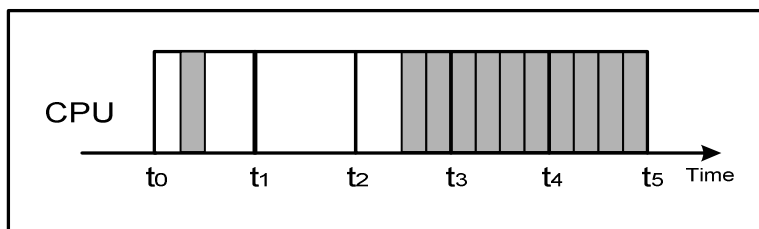


Figure 13. Interval Measurement

Assuming processes have been optimized as much as is economically feasible, it is possible that a system's workload can exceed its capacity. In the absence of "performance tuning" it now becomes a capacity issue to provide adequate resources for the given workload.

11.4 Utilization Measurement

As I have stated, workload measurement is not a method of performance tuning. At the system level it is just not possible to accurately determine performance. For example, is 100% CPU utilization bad? Let's take an example of a single CPU system and a multi-CPU system. If the system is running one long running batch program then on the single CPU system the utilization could be 100% if the job is CPU bound. Does that mean there's a performance problem? It might, but what are the real factors? If the process response time is meeting the business' requirements and there are no other demands for capacity that are not being satisfied then it may be perfectly acceptable to have 100% utilization. On the other hand it is possible that a multi CPU system could be experiencing an aggregate utilization of much less than 100%, but not be meeting a single processes response time due to a single CPU not being fast enough to meet the businesses response time requirement. Figure 14 attempts to show the utilization of a multi-cpu system and a single CPU system for a specific interval using this example.

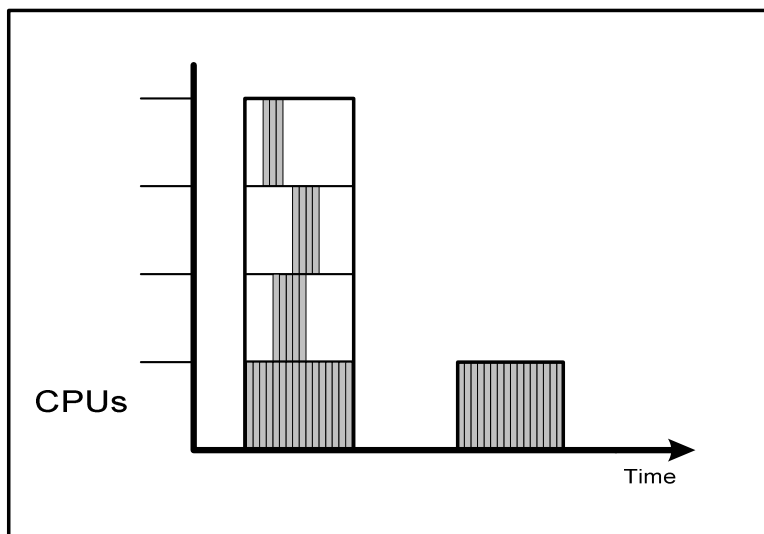


Figure 14. CPU utilization

The bottom line is that you can't determine the "value" of utilization. 100% utilization may be fine for one system's workload and disastrous for another. Conversely it is also possible to have horrible performance and still have excess capacity.

12. Conclusion

Hopefully I have shown that workload measurement can provide several benefits. One benefit is the ability to perform trend analysis on workload utilization. By capturing historical information, one input to a capacity planning model can be derived. It can be difficult to know where you're going when you don't know where you've been. Another benefit is the ability to identify and measure change. Sudden changes in workload can be used to trigger questions to the user

community about business process response time, indicate possible changes in workload due to code changes, or show the increase in capacity of a hardware upgrade. Gradual changes or trends can also be identified and used as input to help stay ahead of the capacity "curve". Indeed, proper workload measurement should help you avoid altogether "Phase I" of Method R on existing, stabilized systems.

In addition to an input to capacity planning, I believe it is also feasible to use workload information to show cost savings, by delayed capacity upgrades, when new efficiencies are discovered. This paper illustrated an example of a single, relatively efficient, SQL query that was being executed so often that it consumed almost 40% of the CPU capacity of the machine. By itself, the query attracted little attention, but when considered over the course of a larger time interval it used a significant amount of CPU resources. With workload measurement it was possible to show a quantifiable savings that reduced CPU workload by 40% and pushed out a hardware upgrade much further into the future.

A recent poster to the Oracle-L list¹⁵ posed the question, "how can I tell if my server is being utilized efficiently?" I think I've shown that this is the wrong question. The question should be "how can I tell how much my server is being utilized?" By introducing efficiency into the question the poster has posed a question for which there is no answer. Instead, we can either focus on the performance of a specific user action, or the capacity of the system providing the throughput for that action. But we cannot make the value judgment of whether the system is running "efficiently".

13. About The Author

Andy Rivenes is an Oracle DBA who has been working with Oracle products since 1992. Mr. Rivenes has specialized in Oracle Applications systems as well as custom Oracle environments. He has worked for Oracle Corporation and currently works for Lawrence Livermore National Laboratory as an Oracle DBA. Mr. Rivenes has given several presentations at OAUG conferences and chaired the OAUG Database SIG. Mr. Rivenes maintains the AppsDBA.com web site and currently focuses on topics related to Oracle system management. Mr. Rivenes is also active in the baseball community and is a certified pitching coach with the National Pitching Association and a member of the American Baseball Coaches Association. When not talking about Oracle topics he can be easily engaged into discussing the latest pitching techniques currently being researched through the NPA. Mr. Rivenes can be reached at andy@appsdba.com.

¹⁵ The Oracle-L list is an email based forum for the discussion of anything to do with using Oracle databases. More information can be found at www.freelists.org.

14. Updates

2.8 – Original release for the Hotsos 2005 Symposium

2.9 – Updated the "Supplement to Method R" to incorporate an explicit stop at step 3.

15. References

- [1] N. J. Gunther, "The Practical Performance Analyst", Authors Choice Press, Lincoln, NE, 2000.
- [2] C. V. Millsap, "Optimizing Oracle Performance", O'Reilly & Associates, Inc., Sebastopol, CA, 2003.
- [3] A. S. Rivenes, "SYSMON, Oracle Workload Management Utility", AppsDBA Consulting, www.appsdba.com.
- [4] A. S. Rivenes, "Oracle Workload Management Using Time Based Optimization Techniques", AppsDBA Consulting, www.appsdba.com, 2003.
- [5] D. A. Menasce and V. A. Almeida, "Scaling for E-Business, Technologies, Models, Performance, and Capacity Planning", Prentice Hall, Upper Saddle River, New Jersey, 2000.
- [6] D. A. Menasce and V. A. Almeida, "Capacity Planning for Web Services, Metrics, Models, and Methods", Prentice Hall, Upper Saddle River, New Jersey, 2002.
- [7] E. M. Goldratt, "The Goal", North River Press, Great Barrington, MA 1986.
- [8] A. Kolk, S. Yamaguchi, J. Viscusi, "Yet Another Performance Profiling Method (or YAPP-Method), Oracle Corporation, Redwood Shores, CA, 1999.
- [9] C. V. Millsap, "Profiling Oracle: How it Works", Hotsos Enterprises, Ltd., 2004.
- [10] Oracle Database Performance Tuning Guide 10g Release 1 (10.1), Part Number B10752-01
- [11] Oracle9i Database Performance tuning Guide and Reference Release 2 (9.2), Part Number A96533-02
- [12] C. A. Shallahamer, "Response Time Analysis for Oracle Based Systems", Orapub, Inc., 2004.
- [13] C. V. Millsap, "Batch Queue Management and the Magic of '2'", Hotsos Enterprises, Ltd., 2000.