The Ratio Modeling Technique

Quickly Performing Low Confidence Capacity Predictions Using Ratio Modeling

This page has been intentionally left blank.

Quickly Performing Low Confidence Capacity Predictions Using Ratio Modeling

David R. Cook (dcook@us.oracle.com)
M. Ellen Dudar (mdudar@us.oracle.com)
Craig A. Shallahamer (cshallah@us.oracle.com)
Orig. March 11, 1997 - v2b (July 1997)

Abstract

This paper formally defines the Ratio Modeling technique as set of steps enabling you to quickly perform low confidence capacity predictions, which are required when budgeting hardware, assessing technical risk, validating alternative technical architecture designs, sizing packaged applications, and predicting production system capacity requirements. This technique enables you to define the relationship between process categories (e.g., batch processes) and a specific system resource (e.g., CPU). The Ratio Modeling technique has been validated by many of the largest Oracle Manufacturing Applications sites, is currently being used by many technical architects, and is being embraced by Hewlett Packard's Oracle Applications sizing consultants. This papers details step-by-step how you can determine ratio categories, calculate the ratios, make capacity predictions using the ratios, and validate both the ratios and the resulting predictions on your production system.

1. Table Of Contents

2. Introduction

Imagine the following two situations.

1. *You just arrived at the office and your boss informs you the budget for next year's hardware purchases is due, today by close of business.*
2. *You are creating a number of alternative technical architectures for a very large and technically complex application servicing thousands of users around the world. Your task is to recommend three technical architectures with their associated costs, risks and benefits. Keep in mind the combination of the number of users, the number of batch processes, the number of interfaces, and the number of geography's served, thrusts this project into an area of extremely high risk with a very high project failure impact.*

These are real situations our group grapples with daily. I think we would all agree these situations or some form of these situations find their way into every IT organization. The problems are centered around defining a technical architecture that supplies sufficient capacity to meet the requirements of the user community. However, as our two above cases show, it is extremely difficult, yet it must be done.

Putting together their combined experiences, independent study, and the problems facing them, Dave Cook, Ellen Dudar, and Craig Shallahamer created this new modeling technique. Their goal was to develop a technique for quickly predicting capacity requirements at a low precision level. The core model is deceptively simple and looks like this;

$$S = C_1 / R_1 + \ldots + C_n / R_n \ +/- \ K$$

The variable *S* is the predicted computing system resource usage requirement such as the number of CPUs, *C* is a count of some defined workload category, *R* is the related ratio, and *K* is the error.

At first glance, this may not seem like a discovery of any significance, but looking closer, and particularly if you have come face to face with the above situations, you will see the wide applicability of this modeling technique. We have successfully used the Ratio Modeling technique for;

- Budgeting hardware,
- Assessing where significant technical risk resides,
- Validating alternative technical architecture designs,
- Sizing packaged applications,
- Predicting current production system capacity requirements, and

This paper was written to formally define the Ratio Modeling technique, to provide a single definitive source for the technique, and to provide its readers with enough understanding to actively apply this technique to their own unique situations. It should be noted the Ratio Modeling technique has been used and validated at many of Oracle's largest and most complex Oracle Applications sites, is currently being used by many technical architects, and is being embraced by Hewlett Packard's Oracle Applications sizing consultants.

3. Definitions

While the Ratio Modeling technique is very powerful, if a few concepts are not well understood and respected, the predicted results will be completely bogus and should be thrown out. As you will learn from this paper, a bogus prediction is not the result of an apparent flawed Ratio Modeling technique, but rather an inappropriate use of the modeling technique.

The concepts we will briefly discuss are system capacity requirements, system capacity capability, prediction precision, and the relationship between time, effort, and prediction precision.

3.1. Requirements And Capacity

The simplest way to explain the relationship between requirements and capacity is to define each and then explain the importance of their relationship. Requirements are the different computing system resources types necessary to enable satisfied users.

With the requirements known, if our goal is satisfied users, we can either design a system to meet or exceed the predicted requirements, alter the requirements by managing the workload, or by tuning processes. Put another way, if the system capacity is cup of water, we need to ensure the requirements are less then a cup of water. If not, users will not be satisfied with the system performance.

A significant part of planning capacity is continually gathering and predicting future requirements and then making the necessary adjustments to ensure the workload is smaller than the available capacity. Making the adjustments is where capacity planning experience, business process knowledge, and creativity come together.

When faced with the prospect of requirements exceeding capacity, we have three main alternatives. We can either tune, thereby reducing the workload, we can buy additional hardware, thereby increasing the capacity (i.e., get a bigger cup) or we can balance the workload, reducing the workload at strategic times. The figure below shows these options.

An entire paper should be written about the "tune, buy, or balance" options. Consistently analyzing performance related issues using this method is awesome. Using this analysis method, we can more effectively communicate risk, costs, benefits, impact of a risk event occurring, and effort for both technicians and management thereby allowing IT to maximize their return on investment.

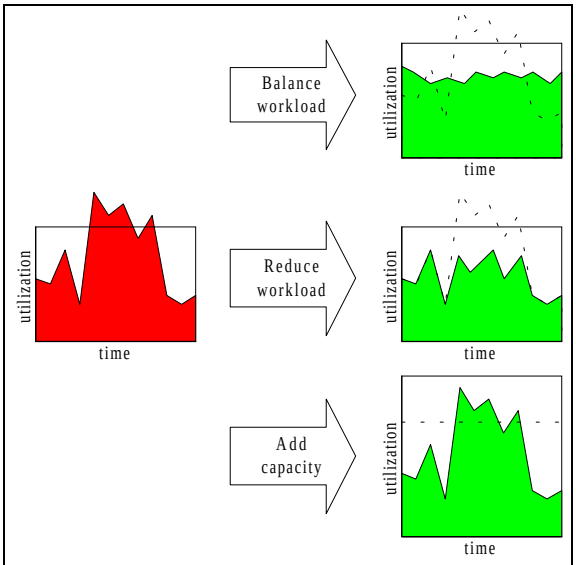

Figure 1. Sometimes requirements exceed available capacity. There are three basic options available. They are to improve workload management thereby reduce the workload, tune the application thereby reducing the workload, or re-architect or buy more hardware thereby increasing system capacity.

3.2. Prediction Precision, Time, and Effort

When I'm asked why so many predictive studies fail, my response is usually centered around precision mismatch. Put another way, the expected predicted precision was higher then the available precision. For example, suppose my customer needs a ballpark capacity prediction for budget reasons. Servicing my client with his best interest in mind, I'll put together a project that will just surpass the required predictive precision. This allows minimizes time and effort, which translates in to thousands of dollars in savings.

A common trap is to believe a high precision prediction can be appropriately completed in two weeks while the capacity planner is fully expecting to provide architectural direction. In this case, the prediction precision requirements exceed prediction precision capabilities. It is the job and in the best interest of both the technical architect and the client to fully understand precision.

As a consultant, my job is to listen closely and to understand how precise my predictions must be and to effectively articulate the associated precision level available at various costs. We use the picture below to help our clients understand the relationship between predicted precision, the time it will take to make the resulting predictions, and the associated effort.

4. The Appropriateness Of Ratio Modeling

Models are reality representations. They are not real in themselves because detail is lost, but they do represent reality. As we discussed above, there are times when high precision predictions are required and there are times when low precision predictions are required. Our responsibility is to determine the appropriate precision required and then select the corresponding appropriate modeling technique.

There are many different modeling techniques available to the capacity planner and the technical architect. We have successfully used queuing theory, regression analysis, cluster analysis, simple math, probability statistics, and ratio modeling. Ratio Modeling is unique since it allows us to relatively quickly make capacity predictions at low precision levels.

We often hear low precision predictions are not appropriate. But we tenaciously disagree. We have found one of the gapping holes in this line of work is quickly making predictions, even at low precision levels, is not just a necessity but a requirement. Low precision predictions are just, if not more, as important as high precision predictions. High precision predictions are very expensive and take months to complete. Many times

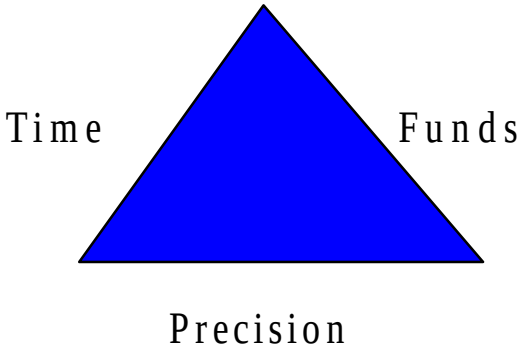

Figure 2. Time spend on a project, the funds required to support the project, and modeling precision can provide are directly related. Changing one component must change the other.

decisions must be made relatively quickly and a low precision prediction is all that is necessary. Budgeting hardware and quickly validating alternative technical architecture designs are two examples of how important it is to produce relatively quick capacity predictions. Fining ourselves in situations like these drove us to develop the Ratio Modeling technique.

5. Introductory Ratio Modeling

Using Ratio Models and the technique to derive ratios is relatively simple. Look at it. It's so simple, its power is deceptive!

$$S = C_1 / R_1 + \ldots + C_n / R_n \ +/- \ K$$

And that is it's weakness. That is, the tendency to misuse the ratios and to not thoughtfully supply the correct information into the model. This is why this section starts off with a discussion about characterizing a workload. If a workload is not characterized properly, as with any modeling technique, the resulting prediction will be bogus and can have a disastrous affect.

By the end of this section you should fully understand how to define ratios and their associated constants, and effectively use the Ratio Modeling technique. After understanding the subsequent sections, the flexibility and wide appeal of the this technique should become apparent.

5.1. Categorizing A Workload

This paper is not intended to be a definitive guide to workload characterization. However, as with any capacity planning study, properly characterizing a workload is enormously important. In particular to the Ratio Method, the desired ratio types will drive how the workload is characterized. Therefore, our objective is to present a simple technique for gathering, organizing, and analyzing usage information, thereby categorizing a workload, so you can quickly derive the required Ratio Model inputs.

Specific usage occurrence counts are one Ratio Model input parameter type. For example, a typical ratio model requires workload characterization by the number of concurrent OLTP users and the number of concurrent batch processes. More specifically for example, at peak processing time, there was found to be 510 concurrent OLTP users and five concurrent batch processes running.

One common method for characterizing workloads is to perform a *User/Functional Analysis*. A *User/Functional Analyses* analyzes *when* certain processes will occur and *how many* processes will be occurring. We typically use a workbook approach. That is,

OLTP Requirements Data Entry Sheet

Process	Login Time	Logoff Time	Number of Occurrences
gl	0	9	0
gl	9	18	425
gl	18	25	50
ap	0	6	0
ap	6	15	150
ap	16	25	75

Figure 3. Every capacity analysis requires a workload requirement figure. Completing a user/functional analysis is simple, yet can be a very complete, way to gather workload requirements. The workload view provides easy data entry but is not very useful in transforming the data into useful information.

we use one worksheet to enter the information and then another worksheet to view the information from another dimension. This other dimension just so happens to provide us with the workload characterization figures we need. The figure below graphically demonstrates our point.

OLTP Usage View

Start Hour	Number of Occurrences
0	0
1	0
2	0
3	0
4	0
5	0
6	150
7	150
8	150
9	575
10	575
11	575
12	575
13	575
14	575
15	425
16	500
17	500
18	125
19	125
20	125
21	125
22	125
23	125
24	125

Figure 4. This figure is simply a different data presentation view than in the previous figure. Notice this view provides the necessary information to quickly spot the peak OLTP load which will be used as part of our ratio model input parameters.

Once we are presented with a *usage view*, we can quickly determine the peak OLTP activity. In our example, peak OLTP activity occurs between 9 AM and 2:59 PM when a predicted 575 concurrently active OLTP users will be on-line. The 575 figure is called a workload constant and would typically be entered into a Ratio Model.

5.2. Calculating Ratios

Understanding how the ratios are calculated is understanding Ratio Modeling essence. Simple ratios are the enablers to make relatively quick capacity predictions.

The best way to explain how the ratios are calculated is to continue using the example from the previous subsection. So far in our example the follow facts are considered known.

1. At peak, there will be a predicted 575 concurrent OLTP users, that is C_1 equals 575.
2. At peak, there will be a predicted six concurrent batch processes, that is C_2 equals 6.
3. We will assume there is no model error, that is *K* equals 0.
4. Our objective is to determine the number of CPUs required, *S*, to meet the peak predicted user load.

Our base Ratio Model is,

$$S = C_1 / R_1 + \ldots + C_n / R_n \ +/- \ K$$

Since we are gathering data from a real production system, the number of CPUs required to meet the load, *S*, is equal to the number of CPUs, *N*, multiplied by the average CPU utilization, *U*. Mathematically, our expanded Ratio Model is,

$$N * U = C_1 / R_1 + \ldots + C_n / R_n \ +/- \ K$$

For our example,

$$N * U = 575 / R_1 + 6 / R_2$$

What remains unknown are the two associated workload ratios, R_1, which we call the oltp-to-cpu ratio, and R_2, which we call the batch-to-cpu ratio, the number of system CPUs, *N*, and the average CPU utilization, *U*.

Ratios are calculated values based upon the information gathered from real life production systems, an extremely realistic benchmark, or if we have no other options, then a combination of the above sources and vendor specifications. Gathering multiple data points from a real life production system will provide all information we need to derive our unknowns. By understanding the production application and then very carefully choosing when to gather your data, you can zero out all but one of the constants, that is, a characterized workload slice. For example, if the two characterized workloads are OLTP and batch usage, there is a good chance OLTP usage will at some point be equal to zero…like in the early morning when only batch processes are running. In this case, our ratio model is further simplified to,

$$N * U = C_2 / R_2$$

Since we can gather the number of system CPUs, *N*, their associated average utilization, *U*, and the number of concurrent batch processes, C_2, simple algebra will yield our batch-to-cpu ratio, R_2.

Let's assume we gathered many samples. There were 12 CPUs, on average 6 batch processes concurrently running, and the average CPU utilization was 45%. That is, number of system CPUs, *N*, is 12, their associated average utilization, *U*, is 45%, and the number of concurrent batch processes, C_2, is 6. Solving for the batch-to-cpu ratio, R_2 is 1.11. Substituting our batch-to-cpu ratio into our expanded formula, we have,

$$N * U = C_1 / R_1 + C_2 / 1.11$$

Sampling from a production environment with both OLTP and batch processes running, we can gather the number of CPUs, *N*, the average CPU utilization, *U*, the number of concurrently running batch processes, C_2, and the number of concurrent OLTP connections, C_1. With this information, we can to derive the oltp-to-cpu ratio, R_1. Suppose, there were 12 CPUs, we sampled an average CPU utilization of 65%, 200 concurrent OLTP connections, and 5 concurrent batch processes. Substituting these values into our expanded model we now have,

$$12 * 65\% = 200 / R_1 + 5 / 1.11$$

Basic algebra derives the oltp-to-cpu ratio, R_1, to be 60.61. That is, each fully utilized CPU can service 60.61 OLTP sessions.

At the beginning of this section we presented a scenario where we wanted to calculate the number of CPUs required to meet the processing requirements of 575 concurrent OLTP connections and 6 concurrent batch processes. Based upon our previous examples, the batch-to-cpu ratio is 1.11 and our oltp-to-cpu ratio is 60.61. Substituting these values into our Ratio Model we have,

$$S = 575 / 60.61 + 6 / 1.11$$

Simplifying for *S*, *S* equals 14.90. What this says is using the Ratio Modeling technique, we quickly predicted, at a low precision level, that 14.90 CPUs, running at 100% utilization, will be required to service 575 concurrent OLTP users and 6 concurrent batch processes. Does our analysis stop here? No, as we will discuss, it has just begun.

5.3. Scrubbing The Data

In our experiences, the data gathered is never perfect. If we were to calculate the ratios without closely examining the data and removing bogus data points, our ratios would be skewed and our predictions flawed. Over time we have developed a short checklist to allow us to quickly identify misleading data points and what to do about them. Below is the checklist.

1. Remove all data points where less than three batch processes are running. The fewer concurrent batch processes running, the more significant the overhead (operating system, Oracle database system, application, etc.). Allowing a significantly higher overhead percentage into our calculations than will appear in production skews a ratio lower. That is, the skewed ratio will predict a CPU can process fewer batch processes than it really can.

2. Remove all outliner data points. For our purposes, outliners are data points which fall outside of a data point cluster or trend line. Referencing our example, data points gathered while current jobs were greater than ten and a ratio greater than three were considered outliners and removed. As with any scientific study, outliners should be investigated as to why they exist. If many outliners exist, additional data points should to be gathered to ensure the outliners don't represent some special event which should be investigated.

3. Remove all data points where other significant computing activity was occurring while the data points where gathered. For example, was a backup running, were there massive data loads or data extraction programs running, where there a significant number of OLTP users online during batch-to-cpu ratio data gathering? If so, remove the data point. While gathering ratio data points, we always run the UNIX **ps** command, log the output, and review it for significant system resource consumers.

Suppose we gathered the below data from a production system early one morning.

Batch-to-CPU Ratio

No. of Batch Procs.	CPU Utilization	No. of CPUs	Batch / CPU Ratio	No. of OLTP Procs.
1	13%	12	0.641	0
1	80%	12	0.104	10
1	90%	12	0.093	15
2	12%	12	1.389	0
2	14%	12	1.190	0
3	38%	12	0.658	0
3	23%	12	1.087	0
3	14%	12	1.786	3
3	32%	12	0.781	0
3	26%	12	0.962	0
4	42%	12	0.794	0
4	20%	12	1.667	0
4	21%	12	1.587	0
4	25%	12	1.333	1
5	20%	12	2.083	0
5	36%	12	1.157	0
6	46%	12	1.087	0
6	39%	12	1.282	1
6	40%	12	1.250	0
7	45%	12	1.296	0
8	49%	12	1.361	0
9	47%	12	1.596	0
9	31%	12	2.419	0
15	20%	12	6.250	0
19	47%	12	3.369	0
23	42%	12	4.563	2

Figure 5. This figure shows gathered data points along with their calculated ratios. A simple spreadsheet for data entry allows for easy mathematical computation and visual graphing.

To spot the trends we graph the data as follows.

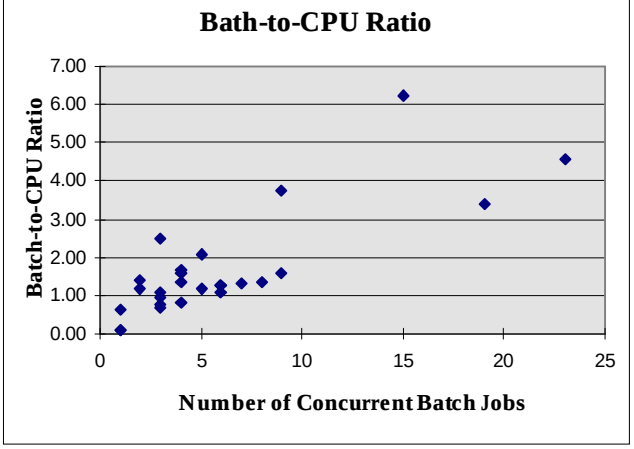

Figure 6. As with regression modeling, visually inspecting the data is very important. Our eyes and minds can spot trends and outliners which most complex routines and statistical analysis can not.

After removing the outliners, the data points and the resulting graph look very different.

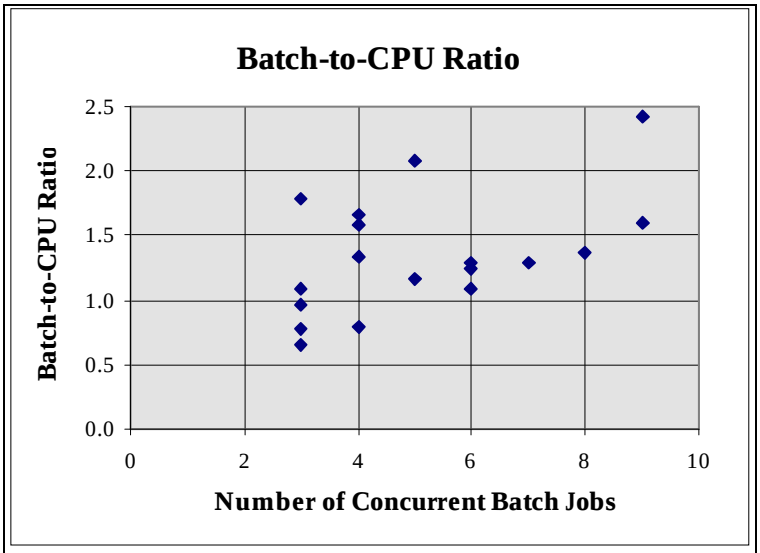

Figure 7. By visually inspecting both the data points and the numbers, we eliminated the outliners. With the outliners removed, we can spot the trend and begin to discuss what ratios numbers are appropriate.

After analyzing the graph, a number of possible interpretations arise. We will discuss a few.

1. "The batch-to-cpu ratio is linear starting at six batch processes and continuing on forever, so if we need to run 30 batch processes that should be no problem." Hardware vendors will commonly use this argument. However, carrying this argument forward will lead one to conclude the batch-to-cpu ratio is infinity. Obviously this will not occur, so this argument is invalid.

2. "Since the batch-to-cpu ratio is consistently below 2.0, then as long as we don't exceed 2.0 in our models, we are safe." This argument has some merit, except 2.0 is on the data point high side. Understanding the Ratio Modeling technique produces a very low precision prediction, that is, the statistical confidence interval is very large, we want our predictions to be very conservative. In this case, a 1.25 batch-to-cpu ratio is much more reasonable.

3. In our example, it would be best to gather more data points until both statistically and visually the trend becomes apparent. The data we have shown does not visually show a strong trend and the statistics surely do not so indicate a strong trend.

Although our sample data does not show this, when enough data points are gathered, we usually see a curve beginning with a steep slope eventually leveling off. When this

occurs, you can very be confident in your ratio number. The initial steep slope demonstrates the overhead affect with just a few concurrent batch job running, but after the number of concurrent batch jobs increases, the overhead percentage per batch job decreases.

5.4. Gathering ratio components

At this point we have introduced the Ratio Model, shown how the ratios are calculated, and provided a few examples. Now we are ready to present exactly how one gathers the data needed to calculate ratios. We will specifically address the oltp-to-cpu and batch-to-cpu ratios for an Oracle Applications system running on UNIX. While actual scripts will not be presented, with the information below in-hand, the job is not too difficult.

Number of CPUs. Many UNIX systems when presented with the **sar -M** command will detail CPU statistics for each CPU. Simply counting lines returned and subtracting any overhead will result in the number of system CPUs. Another option is to simply ask your UNIX administrator.

Average CPU utilization. Most UNIX systems when presented with the **sar -u** command will return the average CPU time in terms of system time, user time, waiting for i/o time, and idle time. Since we want our ratios to include all the activity on the system, including operating system overhead, CPU utilization equals user time plus system time.

Number of OLTP connections. How users are connected into the database system determines how this statistic will be gathered. Unless users are connected through Oracle's multi-threaded server (MTS), simply counting Oracle OLTP sessions works very well. Weather MTS or dedicated server connections, run a blind query on the v$session table and understand which sessions are OLTP, batch, background process, etc. Then construct your query (SQL, shell script, C, etc.) to only count the OLTP sessions.

Number of concurrent batch processes. If the Oracle Applications are running, you will be very aware of the Concurrent Manager. If your site is not running the Oracle Applications, the technique described below should provide enough guidance so you can create the appropriate scripts for other application environments.

We have found most sites' concurrent processing needs consume around 60% of the total CPU during normal daytime processing. Of course, your numbers may vary greatly, but our point is batch processing consumes a tremendous amount of computing system resources.

The below script will quickly return the Concurrent Manager facility's process count. There is an index on **phase_code** column which enables this query to only access a few data blocks, typically resulting in sub-second response.

```
select count(phase_code)
from applsys.fnd_concurrent_requests
where phase_code = 'R'
```

This is a good time to address sample frequency. The answer is not as simple as you might expect. The problem is centered around the fact the UNIX **sar** command presents averages over a time span and our **select count** query's reflect a single point in time. The longer our sample time span, the more likely these two numbers will not support each other. On the other hand, if we sample every 15 seconds, the data collected takes more time to analyze and the tool overhead may become noticeable. Neither of which is desirable.

There is no perfect solution to determining time span frequency. However, regardless of our sample time span, the time span versus snapshot sample error will be reduced as more samples are gathered. It's an important fact, a large time span will produce more data point outliners. If you see this occurring, consider decreasing the time span.

Another important fact is how long a workload process is running. For example, if the typical batch processes run in under five minutes, then a shorter time span is more appropriate to minimize time span error. However, if batch processes typically run for over thirty minutes and OLTP users stay connected for hours, than a 25 minute or more time span may be appropriate. We are trying to avoid missing a concurrent manger process count *rush*. This is because our snapshot type query may miss the rush while the **sar** statement will take the process "rush" into account.

So what's the answer? While there is no "answer," here's what we do. If we have only a couple nights and days to sample data, we may use a five to fifteen minute sample time span. If we have the luxury of gathering data over many days, then thirty minutes seems to work out well. While many outliners will be gathered with a thirty minute time span, the shear data point quantity allows the real trend to appear.

5.5. Validating Your Model and Statistical Correctness

Any model used to make predictions must be validated. While model validation techniques are out of scope for this paper, there are many outstanding references. Some of these references are presented our the bibliography.

It's important to understand validating a model is more than simply saying, "This model has been validated." It allows us to confidently make a statement such as, "Based upon the validated model and your anticipated workload, peak CPU utilization will range from 60% to 90%, 95% of the time." Now that's a powerful statement.

5.6. Making Capacity Predictions

In a previous section's example, we quickly predicted, at a low precision level, 14.90 CPUs running at 100% utilization will be required to service 575 concurrent OLTP users and 6 concurrent batch processes. So is configuring a system with fifteen CPUs

appropriate? Absolutely not! That would leave no room for growth, error, or unexpected processing peaks.

The answer resides in a thorough cost, risk, and the impact of a risk event occurring analysis. The answer is not simply to buy more hardware. In fact, in many cases, a bigger box may not exist. Listed below are some common options.

Buy a bigger box. For example, if six CPUs are predicted to be fully utilized, then twelve will be approximately utilized at fifty percent. The amount of hardware your "throw" in to the equation depends, again, on cost, risk, and the impact of a risk event occurring.

Reduce the workload by either tuning or workload management. When a system is in production or a good user/functional analysis has been performed, an expert performance tuner can estimate how much additional CPU time can be gained by tuning SQL and an expert performance manager can estimate how much additional CPU time can be gained by improved workload management. Be aware purchasing additional hardware attacks the problem's symptom, while performance tuning and management attack the problem and the results will typically outlive the hardware.

Re-architect. Ratio Modeling is commonly used when discovering and quickly eliminating alternative technical architectures. A seasoned architect will know recommending a technical architecture with a 90% predicted CPU utilization has a very good chance of completely saturating the CPUs and dramatically increasing response time and decreasing throughput. One alternative is to propose and quickly evaluate a different technical architecture. The bibliography lists a few references related to technical architecture design.

Perform a more detailed study. A major Ratio Modeling benefit is allowing one to quickly assess if a more detailed study is warranted. For example, if the Ratio Model predicts 50% CPU utilization, then even with a 20% margin of error the CPUs will not become saturated. In this case, it may not make sense to perform a more precise study. On the other hand, if the utilization prediction is 75% and either CPU saturation or a 300 person manufacturing floor shutdown is a real possibility, then investing in a more detailed study may be well worth the investment.

The bottom line is, the Ratio Modeling technique will quickly allow you to make better decisions and recommends to your management. Not necessarily the final recommendation but one more data point to lead you in making the best decision possible at a relatively low cost.

5.7. Using Many Ratios

A natural extension to the ratios models we have discussed thus far is to include additional ratios. For example, instead of just using the oltp-to-cpu ratio and batch-to-cpu ratio, what about using an oltp-to-cpu ratio specifically for General Ledger users, or General Ledger users in Oregon? I think you get the pattern.

While this may initially appear like a terrific idea, keep in mind by simply adding more ratios, model precision does not necessarily increase. In fact, don't be surprised if the model precision actually decreases. It takes time to plan, gather, and analyze data to properly create more ratios. Just as when using regression models, simply adding more independent variables does not necessarily increase model predictability. In fact, using an inappropriate independent variable, such as the outside temperature in Boring, Oregon, will only reduce the model's predictive power. The bottom line is to keep it simple, understand the model's precision capabilities, understand the precision requirements, and ensure the predictive precision exceeds the predicted chosen model's capabilities.

6. Concluding Thoughts

Much more could be said about Ratio Modeling. This paper will not doubt be updated and expanded in the years to come. But I hope our efforts over the last few years, as explained in this paper, has provided you with enough information to begin exploring the possibilities, usefulness, and appropriateness of low precision models. Thank you for your time.

7. References

Chatterjee, S.; Price, B. *Regression Analysis by Example*. John Wiley & Sons, 1991. ISBN 0-471-88479-0

Cook, C. *System Sizing Standards for Oracle Applications.* Oracle Corporation White Paper, 1995. http://www.europa.com/~orapub

Jain, R. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991. ISBN 0-471-50336-3

Levin, R.; Kirkpatrick C.; Rubin, D. *Quantitative Approaches to Management*. McGraw-Hill Book Company, 1982. ISBN 0-07-037436-8

Menascé, D.; Almeida, V.; Dowdy, L. *Capacity Planning and Performance Modeling*. PTR Prentice Hall, Englewood Cliffs NJ, 1994. ISBN 0-13-035494-5

Millsap, C. *Designing Your System To Meet Your Requirements.* Oracle Corporation White Paper, 1995. http://www.europa.com/~orapub

Shallahamer, C. *Predicting Computing System Throughput and Capacity*. Oracle Corporation White Paper, 1995. http://www.europa.com/~orapub

Shallahamer, C. *Total Performance Management*. Oracle Corporation White Paper, 1994. http://www.europa.com/~orapub

8. Acknowledgments

This work was supported by Oracle Corporation and was conducted while the authors were working for (and still are) the Oracle Services System Performance Group. We would like to thank the many individuals who encouraged us to further develop and refine the modeling technique. The result has been a much deeper understanding of Ratio Modeling. A special thanks is very appropriate to everyone in the Americas System Performance Group for their constant support and furthering of performance modeling and performance prediction.

9. About the Authors

All three authors are Oracle Consulting Services, System Performance Group (SPG) team members and founding members of SPG's technical architecture and capacity planning service line. SPG is dedicated to architecting, tuning, managing, and operating optimally performing Oracle based systems around the world.

David R. Cook is a Principal Consultant for the Oracle Consulting Services, System Performance Group. Mr. Cook has over six years experience with Oracle products and specializes in performance management and capacity planning. David is based in Salt Lake City, Utah and can be reached at 801.595.5674 or via e-mail at **dcook@us.oracle.com**. David's papers are available via the Web at **http://www.europa.com/~orapub**.

M. Ellen Dudar is a Managing Principal Consultant for the Oracle Consulting Services, System Performance Group. Ms. Dudar has over ten years of extensive object oriented software development and database experience. Ellen has co-authored a paper entitled *A Life-cycle Based Approach To Ada Software Configuration Management* (IEEE Computer Society Press, 1989). Ellen is based in Houston, Texas and can be reached at 713.658.6948 or via e-mail at **mdudar@us.oracle.com**.

Craig Shallahamer is a Practice Manager for Oracle Services System Performance Group. Since joining Oracle in 1989, Mr. Shallahamer has worked at hundreds of client sites around the world. His specialization is performance tuning, performance management, capacity planning, and technical architecture design related research and on-site engagements. As a result, Mr. Shallahamer has had the pleasure to publish a number of papers at the EOUG, OAUG, IOUW, Asia Open World, and *Oracle Magazine*. Craig is based in Portland, Oregon and can be reached at 503.220.5122 or on the internet at **cshallah@us.oracle.com**. Craig's papers are available via the World-Wide Web at **http://www.europa.com/~orapub**.