Practical Capacity Planning

Dave Wagoner
LexisNexis
david.wagoner@lexisnexis.com

"*The plan is nothing, the planning is everything*."
- Dwight D. Eisenhower

The goal of capacity planning is to ensure that the right computing resources are available in the right place in the computing environment and at the right time. Capacity planning efforts are cost-justified in environments where significant revenue streams are generated by the use of software applications and where capital expenditures on computing and network resources used to operate software applications are substantial.

Capacity planning is in many situations performed based solely on resource consumption projections. Instrumentation reflecting business activity and customer behavior can be integrated to yield dramatically better planning than otherwise possible. Examples from a large UNIX environment are provided.

Introduction

Computer based applications and services are adopted *initially* because of the functionality they provide; *continued* use depends largely on the performance of those applications and services. This applies in particular to web-based applications. Performance, or at a minimum the perception of performance, is therefore a priority in computing environments. Performance is most often quantified in terms of transaction response times and transaction throughput rates.

One potential source of performance problems is a shortage of computational or network resource capacity. In such a situation, some resource (CPU, DASD bandwidth, etc.) will have become saturated and transaction response times will be affected as queuing is incurred. Note that response times are partitioned into *service time*, the time spent performing a particular operation, *and queue time*, the time spent waiting to gain access to the resource to perform that particular operation. Queuing will occur when transactions are arriving faster than the operation to service those transactions can be performed (i.e. service rate).

The terms *capacity* and *performance* are often used interchangeably; such interchangeable usage is somewhat imprecise as there are qualifiers of both of these words that result in terms that warrant further distinction. First, *capacity planning* is the art of forecasting future resource demands and identifying what capacity should be available, where it should be positioned, and when it should be available. That is, capacity planning is performed to ensure that the *right computing resources* are available in the *right place* and at the *right time*. The focus of capacity planning is the future, based on what is known in the present and what has been experienced in the past.

By contrast, *capacity management* has as its goal the optimal use of existing resources to best meet existing needs such that performance is not impacted due to one or more computational resources being exhausted or saturated causing queuing and elevated response times. Such activities include the redeployment of hardware from one area to another, system tuning and the redistribution of data across DASD devices to reduce device utilization and consequently overall response times.

Performance analysis is the art of determining where time is spent at the level of the individual transaction; here the mantra is "find the bottleneck" which is responsible for the greatest contribution to response time. It is worthwhile to note that every system by definition has at least one bottleneck that is preventing transactions from being serviced infinitely fast in zero time. The removal of one bottleneck means simply that response times can be reduced and throughput can be increased until the next bottleneck is encountered.

If workloads remained constant and certain, capacity management would be sufficient and capacity planning would be unnecessary. However, there are numerous instigators of capacity demand change. These include:

- long term usage growth and seasonal usage patterns
- changes in application transaction resource consumption

- technology updates, namely:
  - infrastructure software upgrades
  - hardware replacements
- new versions of application software
- workload migration from one platform to another
- server consolidation
- new projects and products

Metrics reflecting growth and seasonal usage patterns can be collected, which make capacity changes that can be attributed to this factor predictable, as demonstrated below (see examples.) The capacity impacts of technology updates, new versions of software, and to a lesser degree, workload migration from one platform to another, can be relatively well understood and sedately addressed in capacity plans.

Changes in the resource consumption of application transactions often represent a surprise. The addition of features and functions, the addition of data, “enhancements” to output, and very rarely non-optimal code (a.k.a. bugs, design flaws, etc.) can cause transaction resource consumption to increase. One of the inherent duties of staff tasked with capacity management and planning is the monitoring of transaction resource consumption and making known the opportunities for improvement to the pertinent staff members. On rare occasions transaction resource consumption decreases, affording more capacity for other capacity demand drivers.

The most challenging area of capacity planning is incorporating the demands of new projects and products. There is often little known of the behavior of these new demand drivers yet there is typically an expectation of development and business staff that resource consumption can be determined and therefore capital can be apportioned prior to the existence of application source code. This is done to complete cost-benefit analyses and justify labor and capital expenditures. The capacity planning challenge of facilitating these new workloads is particularly prominent with large organizations where numerous disparate groups sponsor new projects and products in a fashion almost seemingly coordinated to thwart with unerring accuracy even the most exacting and comprehensive capacity plans.

The creation of a single capacity plan is not sufficient. Because of the number of independent instigators inducing changes in capacity demands, contingency capacity plans are no longer a luxury but are necessitated, notably in large organizations where capital expenditures are large but the budgets for such expenditures strictly are upper-bounded.

Capacity planning is an ongoing process (figure 1). Areas of concern are monitored to determine which components of hardware, system software, or application software warrant scrutiny; data is then gathered to quantify conditions; analysis follows, including trending of existing workloads as well as the integration of expectations of demand changes from other sources. Next, appropriate resources are deployed if necessary, and monitoring is then performed to examine the impact of changes.

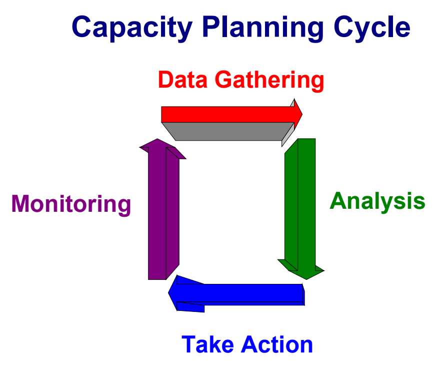


Figure 1

The frequency with which new capacity plans are to be created has a number of determinants. These include:

- Capital budget cycle
- Seasonal workload patterns
- Software licensing cycle
- Hardware vendor discount cycles

The above items represent strategic motivators to the timing of plans. In addition to these factors, acquisition and deployment periods are tactical factors that dictate how far in advance decisions are to be made with regard to acquiring resources.

Essentials of Useful Capacity Planning

Capacity planning for computing environments is commonly done by projecting future consumption of resources (CPU, memory, I/O bandwidth, DASD space, network bandwidth, etc.) based on prior consumption during recent months. Minimally, capacity planning is merely evaluating the current resource consumption to determine if additional resources are required *now*. For systems of low priority with regard to revenue generation and capital resource commitment, this may indeed be the level of capacity planning that is appropriate. For customer facing systems that are used to generate revenue or require substantial capital commitment, a higher degree of capacity planning is warranted. For such systems the resource consumption data is important but is only half of what is needed to produce a useful capacity plan.

In using resource consumption data as the sole basis for capacity planning, there is no integration of the associated application activity (i.e. transactions) in the derivation of capacity demand projections. The material impact of application activity is what is needed to produce a useful capacity plan. This data can be manifest as instrumentation data emitted from locally written applications, log files, or activity from external application monitors. The idea is to correlate overall resource consumption with application activity to determine transaction costs (*cost* in this context refers to the consumption of resources and not to a financial cost). Once transaction costs are obtained*, business drivers such as annual growth, seasonal trending, the acquisition of large blocks of new customers, etc., can be used to more accurately impact the future capacity demand projections*. This idea is paramount to performing useful, practical capacity planning, as this makes it possible to establish the ramifications of activities of seemingly distant organizations such as sales and marketing upon computing environment infrastructures. It is worthy of emphasis that such capacity ramifications can be anticipated well in advance of their materialization.

Accumulation of transactional load data, optimally over a period of years, allows seasonal transaction loads as well as annual growth rates to be identified. Seasonal patterns and growth rates are decomposed, projected, summed, and then combined with transaction costs. These results are integrated with any relevant changes to market or application structure to determine future capacity demands.

The correlation of customer activity and resource consumption has been repeatedly applied for several years at LexisNexis with great success, despite the nature of the transactions being driven by a large, uncoordinated, globally located (and therefore in disparate time zones), web-based customer population. In January of 1999, the peak hour of activity for all of 1999 and its corresponding activity level were predicted. The actual hour of peak activity was exactly as predicted and the transaction level was within 4% of what was predicted[1].

There is a substantial secondary benefit to the awareness of transaction costs. Tracking transaction costs with high frequency makes evident the ramifications of infrastructure or application changes. That is, if changes to add new features and functions to an application are made and transaction costs are thereafter substantially elevated, the tracking of costs yields evidence of performance tuning opportunities or the need to revise capacity plans to account for the subsequent increased demand. In this way, the ongoing inspection of transaction costs provides a potential indication of either performance tuning opportunities or discrepancies that will need to be addressed so that capacity plans conform to resource demands[2].

Further, tracking transaction resource consumption combined with capacity plans may expose "breakpoints" where particular resources are not able to scale and some architectural adjustment must be made. An example is a server consolidation effort where resources and workloads on a single system image are increased. What is often overlooked is that the network interface bandwidth is upper-bounded; upgrading of the interface or the use of additional interfaces via trunking[3] may provide relief in such a situation.

A common misconception is that capacity shortages are the only source of poor application performance. Low resource utilization does not necessarily equate to optimal or even acceptable performance, as poor performance can occur when no resource is at saturation. An example of this is an application that is performing I/O operations unnecessarily. The devices that are facilitating the I/O operations may be quite distant from saturation and have ample capacity available; however, the fact that I/O is being performed at all impacts transaction response time.

The time required to tune applications is frequently overlooked. The actual tuning of the application itself is not always the issue, as conflicting priorities with software development and support organization cause delays in focusing resources on tuning issues. New products, features, and functions are often higher priority than performance. Convincing these organizations to dedicate resources to performance tuning is at times nontrivial. Catastrophic performance issues that directly affect revenue streams often precipitate such tuning efforts.

Financial Motivation for Capacity Planning

Capacity planning is an investment; the currency of such an investment is primarily staff time and effort. The return from an investment in capacity planning can easily be translated into monetary gains in both near and distant futures.

Staff time dedicated to capacity planning may appear at first to be a luxury. Capacity planning duties

[1] In 2000, due to the unprecedented irregularities with the US Presidential election and the corresponding transaction activity, the deviation between predicted transaction levels and actual transaction levels increased to 15%.

[2] A somewhat subtle source for a transaction cost increase without any code change is a limitation in the scalability of the application. Such limitations often originate from thread related activity. Scalability limitations of this nature are detected as the ratio of System CPU time to User CPU time increases.

[3] "Trunking" is a method used to combine multiple communication paths via software so that they appear as a single network connection such that the resulting bandwidth is higher for the combined connections than for an individual connection.

typically fall to system administrators who are already overburdened with periodic maintenance activities, pursuing functionality and performance issues, installation of new hardware and software, and even database and application administration. Compounding this is the estimation that the average system administrator has less than two years of experience. It is no wonder that capacity planning is viewed as an afterthought amongst these other rather prominent priorities.

In this context, it may at first appear to be difficult to justify the dedication of substantial staff time to capacity planning for a computing environment. The reality is that where annual capital budgets are substantial, an investment in capacity planning efforts is easily justified.

First, capacity planning is necessary to protect the revenue stream by ensuring that adequate capacity is available for revenue generating activities. Shortages in resources normally result in queuing of activity and degraded response times. Internet customers are typically tolerant of websites that are occasionally unavailable, as this is seen as an intermittent and correctable issue; these customers are far less tolerant of websites that suffer slow response time. (Outages of websites that are time sensitive, such as *eBay*, are notable exceptions to this). It is no leap in logic to anticipate that the departure of customers in the short term due to poor performance will lead to the loss of those customers (and corresponding revenue) over the long term. Reputations of websites, like those of people, are somewhat difficult to restore.

The second way in which dedicated staff time for capacity planning is justified is somewhat subtle: capacity planning is also performed to determine when it is appropriate to *defer* purchases to conserve financial resources. Recall that the goal of capacity planning is to ensure that adequate resources are present in the right amounts in the right place and at the right time; that is, not too late but also not *too early*.

There are three immediate ways in which the deference of acquisitions is important. First, the price/performance ratio of hardware is typically decreasing over time (Moore's law). Deferring a purchase therefore means that acquisitions costs will be reduced for a fixed amount of hardware. Alternatively, deferring a purchase also means that the same acquisition funding will purchase more resources.

Second, deferring purchases until absolutely necessary provides application support organizations time to tune applications. If applications can be tuned, fewer computational resources will be needed to support those applications. A purchase prior to a tuning effort may result in surplus (depreciating) hardware that is not contributing to the revenue stream. This should not be underrated; application tuning can reduce resource consumption by orders of magnitude – for large configurations the saving can be quite staggering.

A cautionary note is appropriate here regarding the timing of purchases – purchasing schedules must incorporate acquisition, deployment, and integration time for new hardware resources to ensure that resources are available when they are needed. Deployment and integration times for new resources can easily exceed acquisition times at large sites that have disciplined and rigorous change control processes.

Third, deferring purchases has a financial impact on the time value of money, an impact familiar to accountants. If a loan is needed to acquire the resources, interest costs are incurred. If a loan is not needed but corporate resources are dedicated, those resources cannot be earning interest or dedicated to other business needs.

One or two success a year in protecting revenue streams or by realizing substantial cost savings by the deferring of a hardware/software purchase can easily have an impact on the corporate bottom line sufficient to justify staff time. As an example, just three of the activities of capacity planners for the UNIX environment at LexisNexis during the first half of 2001 have yielded return in excess of twice their annual staffing costs of slightly under $1 million. First, capacity planning was performed for a back-office decision support system. The organization using and funding the systems was advised that an investment of $2 million was necessary. However, a capacity plan was completed that showed that this was not the case and indeed any purchase in this area was unnecessary.

Further, there were two distinct occasions where revenues were seriously threatened by software issues discovered during capacity planning processes; these issues would not have been otherwise noticed until the revenue losses materialized. A single day of revenue far exceeds the annual staffing cost for the capacity planners in the UNIX environment.

One strategy to avoid capacity planning is to purchase and stockpile resources well in advance of their potential need. For the three reasons outlined above, diminishing hardware costs, the impact of tuning efforts, and the time value of money, this strategy does not derive maximum benefit from acquired resources. This is in effect an "insurance policy approach" to capacitating a computing environment, and may indeed work in small computing environments with modest capital budgets. However, successful computing environments and their capital budgets rarely stay small, and the consequences of purchasing substantial resources too far in advance of when the resources are needed exceeds the cost of the staff time to perform capacity planning sufficient to avoid such purchases.

Determining Staffing Requirements

Large consulting firms are fond of using the number of computer systems in an environment as a guideline to determine staffing requirements for administration of those systems. The reality is that there is substantial variance in the level of effort required to maintain systems within an environment. There are typically systems in an environment that run with minimal administrative requirements for months whilst other systems demand considerable attention. That is, the "80-20 Rule" applies: approximately 80% of the staff time is concentrated on approximately 20% of the systems in the environment. Any validity to the guidelines used by large consulting firms regarding staffing levels is due to the proportions being similar across different environments rather than solely on the number of systems.

There is a temptation to similarly estimate staff time necessary for capacity planning as a function of the number of computer systems in an environment. The 80-20 Rule and the number of systems can be applied here, but perhaps more telling metrics are the periodic *capital budget* for expenditures on computing resources and the anticipated periodic *revenues*. Since capacity planning is focused on the future, using the current number of systems is perhaps a distorted staff level estimator. Further, since the two principle areas in which capacity planning can benefit an organization are by protecting the revenue stream and by reducing resource acquisition costs, it would seem appropriate to establish the level of effort dedicated to capacity planning as a function of these quantities.

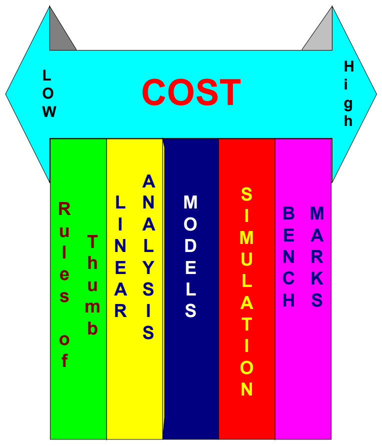


Figure 2 - The Capacity Planning Analysis Cost Continuum

Another determinant in the staff resources required for capacity planning is the depth of detail with which the analysis portion of the capacity planning cycle is performed. This is reflected in figure 2 (derived directly from work done by Leroy Bronner circa 1982.)

"Cost" in this context reflects the level of resources, both in staff hours and supporting hardware and software, translated into a monetary value. Rules of Thumb (ROTs) are useful in determining high-level approximations. Examples such as "10% of the activity on any given day occurs during peak hour" and "0.5% of the activity in any given month will occur during the peak hour for the month" enable capacity planners to quickly determine if a proposed product's activity will have a substantial impact on existing resources.

On the high-cost end of the continuum is the use of benchmarks or "load tests." A remark concerning benchmarks is appropriate here: the only truly relevant benchmark is the running of one's own application. Using SPEC, TPC-x, dhrystone, and a host of vendor benchmarks are useful to obtain a general sense of what may happen under a limited set of conditions, but to truly capture what will happen for a specific application, the application itself must be run and it must be done so in conditions that would exist when the application is actually used[4].

Benchmarks can (and are) run on systems used in production (i.e. used to service customer activity), typically during off peak hours. Ideally a parallel environment is used to not interfere with customer activity, but duplicating the production environment and maintaining it can be costly. A scaled-down version of a production environment is of some value, but typically bottlenecks are discovered in the actual production environment because the benchmarking environment is not sufficiently capacitated to expose these benchmarks[5].

There exists a substantial staff dedicated to capacity planning and performance analysis at LexisNexis. A pattern has been established that there is slightly less than one capacity planner dedicated to the UNIX environment for every $1 million of the UNIX environment capital budget[6] (this number may be different for OS390 and Win2K environments.) Duties

[4] In addition to running the actual application in production environment conditions, the application must be run using data and transactions that would actually occur in use. Capturing customer activity for "replaying" during a benchmark is ideal but non-trivial.

[5] Applications running on systems in a load test environment that are smaller than those in a production environment are not always able to generate the transaction rate, and consequently the I/O load, the network traffic, and memory and buffer space consumption as observed in production.

[6] Revenue streams are not made public in the case of LexisNexis and therefore can not be used for the purposes of describing staffing efforts.

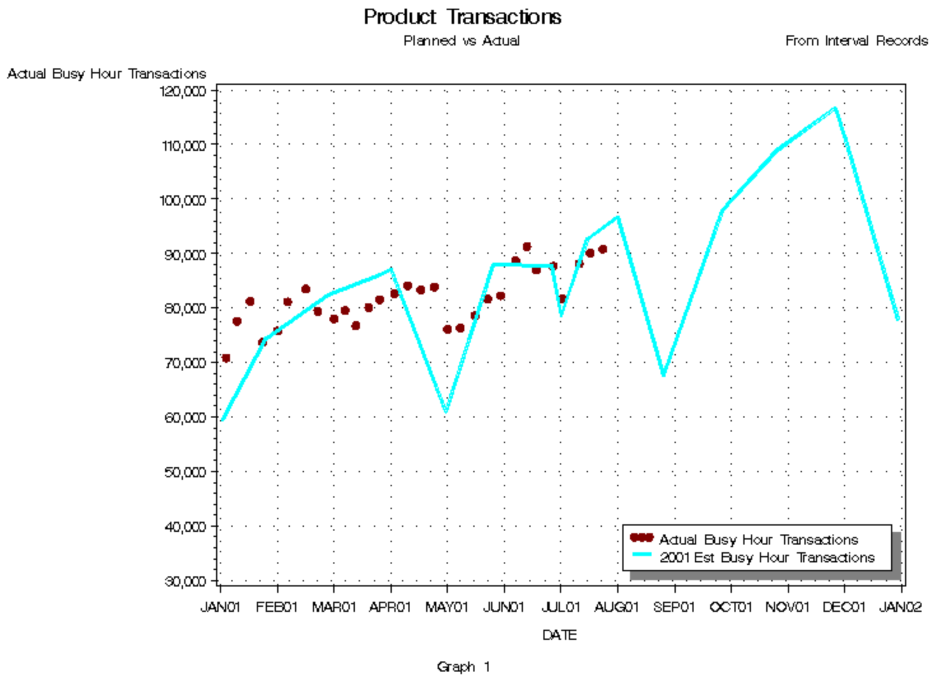
of this staff are primarily capacity planning, performance analysis, maintaining vast volumes of application instrumentation to support planning efforts, and special studies in capacity and performance areas.

Capacity Planning Examples

A number of examples of capacity planning excerpts follow. They are not entire examples of "capacity plans," but rather key excerpts that present noteworthy aspects of capacity planning. In reading the examples, two key points must be noted. First, the time period of focus is the *peak hour* of activity on a daily basis, in terms of transactions[7]. It is this hour for which capacity should be available, rather than average, median, or some other hour of activity[8]. Using this increment of time is sufficient in these examples, but may not be appropriate for capacity planning in all environments.

What constitutes success in capacity planning may not be the same in every situation. In these examples, maintaining CPU utilization below designated thresholds during peak hours of activity to minimize queuing is the objective. Memory, I/O bandwidth, or network bandwidth could also be resources for which planning is performed; it so happens that for the OLTP applications in the examples below that CPU resources are exhausted prior to any of the other resources. Other plans, such as those for decision support systems, may have success defined as limiting the number of hours that systems are utilized above a particular threshold or some other relevant criterion.

Second, the majority of examples relate to Sun Microsystems servers. Capacity ratings for these servers are expressed in terms of "M-values" or "M-quanta." This is based on the work of Bill Walker of Sun Microsystems[9]. M-values are the best measurement of aggregate system capacity known to date. This measurement incorporates not only the number of CPUs and clock speed of the CPUs, but also the system type (to incorporate backplane impacts such as memory latency), operating system version, and L2 cache sizes.



Graph 1
Example 1

Example 1

The first two examples show customer activity in terms of transactions that materialized compared to what was planned. Note that the customers generating the traffic are web-based, have "bursty" (Erlangian) arrival rates, and constitute a large population that is geographically dispersed and therefore in differing time zones.

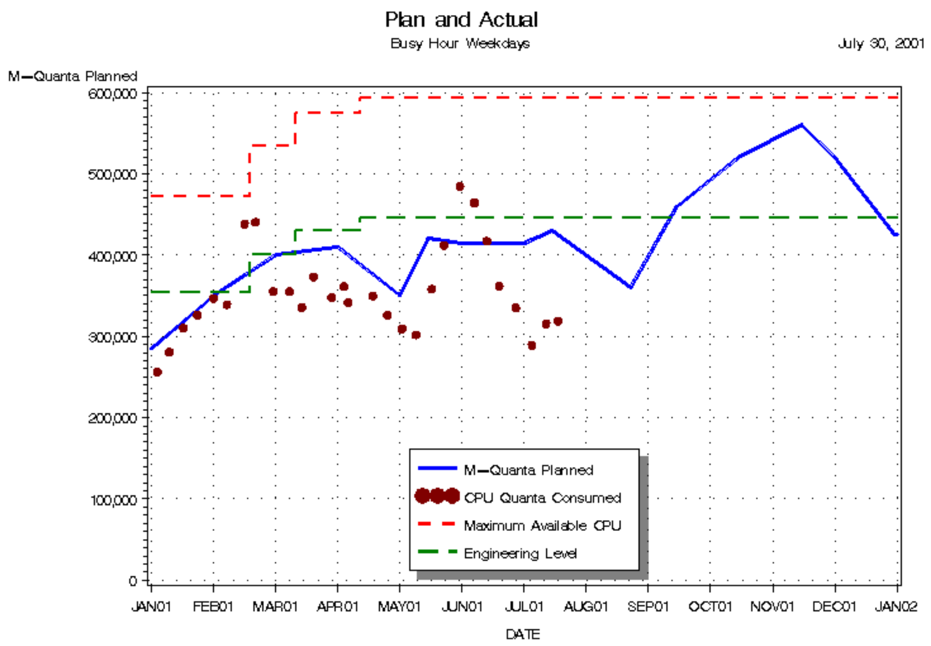
In graph 1, peak hours of activity for a set of combined products during each week are plotted against a "plan", where the plan is an integration of a projection of transaction activity based on prior years of data and what is anticipated regarding changes in business activity. Activity from prior years is decomposed into a seasonal pattern and a growth pattern. Extrapolations on each of these components are made and the results recombined and then integrated with anticipated impacts resulting from business changes to produce the plan as seen in the graph.

Deviations from the plan warrant scrutiny. In this example, transactions are slightly higher in January than planned – quite possibly in the aftermath of the unprecedented activity associated with the 2000 U.S. Presidential election. Activity in May decreases as expected, but not as much as expected for the seasonal pattern. A review of what changes in the marketplace and in products, as well as an evaluation

[7] The peak hour of transactions does not always mean the peak hour of resource consumption. For applications run at LexisNexis and shown in these examples, the peak hour of transactions and resource consumption do indeed match.

[8] Dynamic Reconfiguration capabilities to dynamically add and remove resources are available with some platforms; at this point in time there are still issues with such technology, but this could eventually alter this practice.

[9] The concept on which Bill Walker's work is based originated from Joe Major's work on mainframe systems in this area.



Graph 2
Example 2

of the likelihood of their recurrence, will provide the feedback needed to refine future revisions of the plan. Continuous improvement is an underlying theme in capacity planning activities.

Example 2
Resource consumption is associated with the transactions in example 1. For this particular set of applications, CPU consumption warrants the most attention from capacity planners. Actual CPU consumption is plotted against the planned consumption in graph 2.

Points are plotted by day rather than week in this graph. Deviations worthy of scrutiny occurred in mid-February and in early June. Two features of this graph warrant comment. First, the uppermost capacity line represents 100% CPU utilization; it is not possible to consume more than this level. Somewhat below this is what is referred to as the "engineering level" of capacity. An objective is to avoid exceeding the engineering level for sustained periods.

The difference between 100% utilization and the engineering level is driven by two characteristics. First, there is a utilization upper-bound (UB) which an individual system should not exceed. When a system exceeds this threshold, queuing ensues and transaction times increase. The utilization upper-bound can be thought of as a service level objective. The second contributor to the engineering limit is referred to as the *failover* capacity limit. This is the limit above which capacity should not be exceeded so that should there be a failure of one system among all systems concurrently offering a particular service, the surviving systems will be able to absorb the full workload of the surviving host and still operate below UB.

For example, assume two systems are operating in tandem ("active-active" configuration) to provide a service and are configured such that if one fails the other absorbs the load. If the upper-bound capacity limit for an individual system is 90% utilization, then the engineering limit for the composite set of coordinated systems is 90%/2 = 45%.

This percentage can be calculated for N similarly configured systems with upper-bound capacity for individual systems UB by the equation:

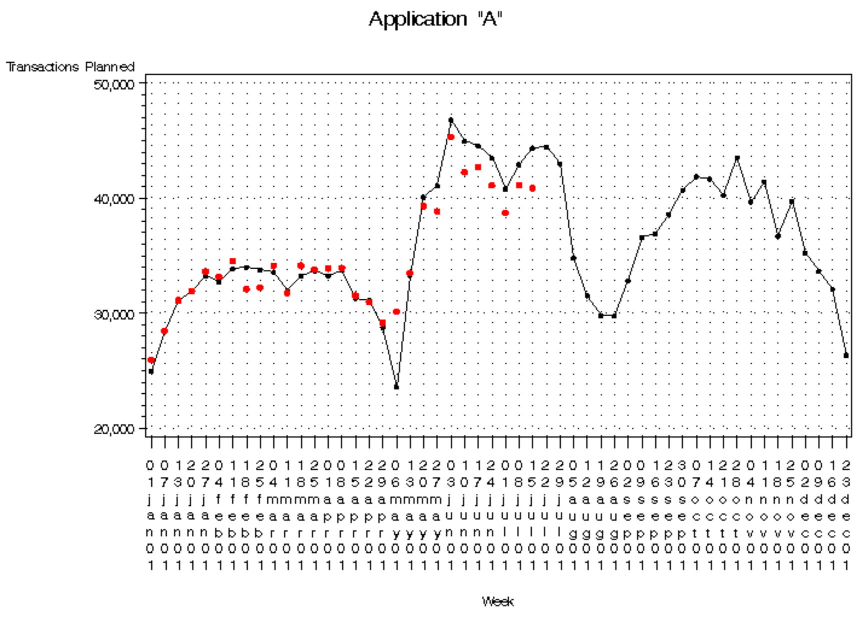
$$\text{Engineering Limit} = \frac{\text{UB x (N -1)}}{\text{N}}$$

For Sun Microsystems servers, UB is typically between 85% and 90% CPU utilization. Note that this is for a peak hour of activity and represents an average utilization for that hour. Experience at LexisNexis has shown that response times for applications can double if UB is consistently exceeded, even though CPU utilization remains below 100%.

Example 3
Achieving high accuracy in planning for peak hour transactions in a dynamic business environment can be challenging, as demands are driven by more than just seasonal patterns and growth. A common situation is that a small number of products contribute the most impact on overall capacity and the relative load associated with all other products is comparatively inconsequential (80-20 Rule). Therefore, focusing planning efforts on dominant products will likely yield the greatest accuracy for the overall plan.

Graph 3 shows actual transaction levels during peak hour of each week compared to plan levels for one of the major products in the example environment. Prior graphs used primarily a resolution of months along the X-axis for plan levels; with sufficient data, this resolution can be improved to weeks with great effect. Recall that the customer population is large, web-based, and geographically distributed, so this degree of accuracy is impressive.

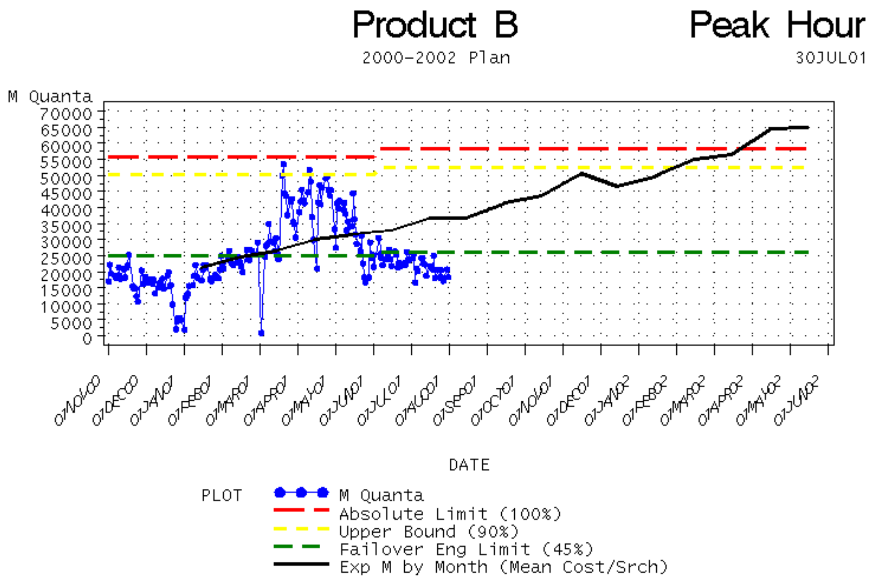
Example 4
Three tiered architectures are popular in environments hosting web products. The three tiers in the architecture are web-servers, application servers, and database servers. The prior examples focused on



Graph 3
Example 3

application servers. Graph 4 shows the CPU resource consumption on a database server for a product.

There are numerous points worth mentioning in this example. First, the CPU resource consumption plan is derived by determining the amount of CPU consumed on a database system for customer transactions that arrive at the application server (i.e. “cost” or “work” per transaction), and then multiplying this by the peak hour number of transactions for each week.



Graph 4
Example 4

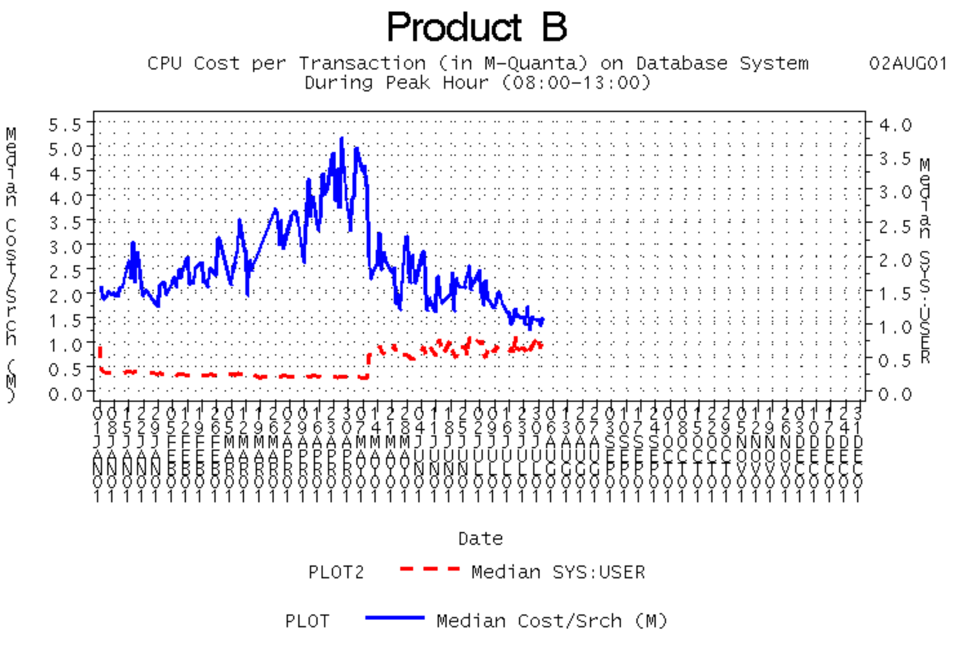
Determining the cost can be accomplished via regression analysis if the nature of the activity performed by the database server in response to the transactions is distinguishable and diverse. If the various database activities for transactions are approximately uniform with respect to CPU consumption, then it is sufficient to compute the “cost per transaction” by dividing the CPU consumed by the number of transactions for a large number of time intervals (preferably during peak and near peak hours) on the database system and performing a univariate statistical analysis. Median values worked well in this example as opposed to averages, which were generally higher due to outliers[10]. Note that this assumes an approximately constant cost per transaction as activity levels increase. This is likely to not be true in general as workloads increase and scalability limitations are discovered.

This plan was derived in February and early March of 2001. The plan shows that the available CPU resources are not only insufficient to provide failover capacity early in the year, but during the first week in September the upper-bound capacity for an individual system will be exceeded without the addition of hardware resources or application tuning. These findings were sufficient to inspire the latter approach. These conversations began in late February when actual CPU consumption was approximately at the failover capacity limit.

Actual CPU utilization tracks well with plan until mid-March, when a major revision to the software running on the application server was made. The impact on the database server was dramatic, as CPU consumption was not only above the failover capacity of 45%, but it exceeded the upper-bound capacity limit of 90%. Transaction response times during this period were notably longer.

The CPU consumption dropped to zero one day in early March. This was the result of a system failure where this workload had been

[10] A conservative plan should also be derived using perhaps 90th percentile cost per search values and a risk assessment made as to which plan should apply.



Graph 5
Example 4

accommodated elsewhere as the product continued to operate.

To aid developers in tracking the impact of their application tuning efforts on the database system, daily tracking of the cost per transaction was charted. Graph 5 shows an increase in the cost per transaction as a result of the mid-March software changes. Tuning efforts brought the cost down, but suspicion was raised as the system to user CPU ratio dramatically increased. A high system to user CPU ratio is an early warning indicator of potential scalability limitations. As a result, this metric was added to the daily chart for tracking purposes.

<u>Relevant Digressions</u>

There are a number of slightly digressional capacity planning topics that warrant at least brief mention. Insufficient attention to these points is guaranteed to condemn even exceptional capacity planning efforts to an abrupt and fiery ruination.

The first of these is the need for at least rudimentary social skills of those performing capacity planning whilst they operate in a professional setting. Conveying data and conducting subsequent discussion plans and their justification requires substantial communication in small groups and in public forums. If an inappropriate style (a.k.a. "career limiting feature") overshadows the content of the communication, then the entire effort of capacity planning has been wasted save for a memorable and possibly entertaining calamity. Human nature is such that the message is not easily disassociated with the messenger, and an ungraceful performance is certain to detract from the credibility of content.

Phrased another way, a degree of tact is necessary. This is truer in the area of capacity planning than in perhaps other support areas due to the influence capacity plans can have on capital resources as there is unfailingly some form of competition for these resources. Agendas of various competing parties may be contradictory and capacity plans may be used as fodder to support one position over another. There is a relevant quote regarding tact: "Tact is the art of making a point without making an enemy." (Isaac Newton). There are few supporting roles in which this is truer.

Capacity planning attempts to prepare for the future by anticipating events well before they occur. Professionals in another notable area in which this is attempted, namely weather prediction professions, suffer derision whenever their predictions are even mildly erroneous and are taken for granted when they are correct. However, these professions are generally afforded the luxury of not suffering any direct impact for forecasting conditions that did not materialize. This is not so for capacity planners; they enjoy the full benefit of the gratification presented to them by the organizations they serve whenever plans deviate from reality.

One solution to this dilemma is to ensure that capacity plans that are produced are indeed *plans* and not *forecasts*. This distinction is that forecasts are a prediction of what will happen and may be judged more or less accurate; a plan *is an agreement amongst all participating parties* to prepare for a particular set of conditions in the face of uncertainty. Parties other than those directly responsible for capacity planning, such as business or other support organizations that will in some way have a dependence on the resulting capacity plan, are engaged to provide higher quality data than otherwise likely to occur. Essentially, their assistance is enlisted to clarify and lend scrutiny to assumptions, business expectations, and data.

If the conditions that actually materialize differ from those expected by the agreed plan, then an investigation as to why the plan differed from reality should be pursued so that subsequent plans or even the planning process can be improved[11]. The principle here is that by engaging parties who will be affected by capacity plans being produced, the quality of those capacity plans is assured to be higher than otherwise possible as they will have an active role and share in the rewards.

As more organizations become involved in the process of developing capacity plans, it will become evident to capacity planners that capacity and performance are but two among several factors that ultimately are used to make decisions regarding resources to be purchased. Other critical factors include datacenter floor space, the number of systems being managed by system administration organizations, flexibility of available architectures, software compatibility, and corporate platform strategies to lower overall operating costs. For example, it is entirely possible that a platform that is technologically inferior to others available is selected to best meet the overall computing requirements. In situations where decisions are being made to the contrary of their advice, capacity planners are best served by ensuring that the arguments they have forwarded have been understood.

Summary

What has been described here is a view of capacity planning that is of practical, rather than strictly theoretical, use. Many organizations only use resource consumption data to produce capacity plans if they indeed produce capacity plans at all. The benefit of merging resource consumption data with customer transaction data would appear to be intuitive, yet this is rarely done. The suitability of the resulting capacity plans is generally far better than otherwise possible.

It must be recognized that capacity planning is concerned with the placement of *the right computing resources* in the *right place* and at the *right time*. The deployments of resources too late will likely result in poor application performance, discourage customer transactions, and eventually impacting corporate revenues. The deployment of resources too early affects capital budgets such that operating costs are higher than necessary. Capacity planners do indeed walk a tight rope. In computing environments where revenue streams and capital budgets are substantial, the investment of staff time to providing capacity plans easily yields a significant return.

[11] In cultures requiring a scapegoat, blame is shared amongst the participants when plans and reality differ. This type of culture is highly undesirable because participants may be reluctant to share useful data in the future if they are faulted and chastised. If conditions that were expected in the plan do indeed arise, then competition for the credit will of course ensue.

Acknowledgements

Recognition is in order for Dave Heald and Jim Aldrich of LexisNexis, as they (patiently and repeatedly) conveyed many of the ideas summarized here until they were adequately understood. George Dodson of candle.com endured editing duties and tolerated many questions. Thanks to Dave Heald for recreating and updating the figures based on hardcopy material. Recognition is also in order for Bill Cooke, my immediate supervisor at LexisNexis, who commissioned this work and supplied ample encouragement, only to be forced to endure the review of its drafts; no good deed goes unpunished. Many valuable comments came from those at LexisNexis that have reviewed the document; in particular Mike Natale, Dave Heald, and Jim Aldrich.

Bibliography

[1] Raj Jain, The Art of Computer Systems Performance Analysis, John Wiley and Sons, 1991, ISBN 0-471-50336-3.

[2] Daniel Menasce' and Virgilio Almeida, Capacity Planning for Web Performance, Prentice Hall, Upper Saddle River, NJ, 1998, ISBN 0-13-693822-1

[3] Daniel Menasce' and Virgilio Almeida, Scaling for E-Business, Prentice Hall, Upper Saddle River, NJ, 2000, ISBN 0-13-086328-9

[4] George I. Thompson, "Six Levels of Sophistication for Capacity Management", CMG 2000 Proceedings