Metrics and Methods that Avoid the ITR Trap

Joe Temple
Low Country North Shore Consulting LLC
and Coastal Carolina University

Abstract: The are several written several Linked in Posts about "The ITR Trap". You can also find an article here on the subject: http://www.ibmsystemsmag.com/mainframe/Business-Strategy/Competitive-Advantage/ITR-performance/
This paper will develop metrics, estimators, and methods by taking a deeper dive into some utilization measurements, to provide insight into how to avoid the ITR trap. *The author would like to thank Len Santlucia and Marty Deitch and Alex Kim of Vicom Infinity for sharing the utilization data used in this paper.*

The ITR Trap

ITR is an acronym for "Internal Throughput Rate". It is a performance metric that indicates the maximum achievable throughput of a server. ITR metrics are very common, because they reflect the hardware vendors view of performance. A machine with higher ITR can potentially do more work than a machine with lower ITR. Examples of ITR metrics are MIPS, tpmC, TPS, SAPS, SPECint Rate, FLOPS, SPECjbb, SPECJapp, etc. They are typically derived by a benchmark test that drives machines with a "Typical" or "Representative" load.

To show the maximum potential throughput, these test loads avoid the causes of delay or bottlenecks, like I/O wait states, and paging. In some cases, they are designed to minimize contention between parallel threads and provide a "balanced load" which optimizes multiprocessor scaling. To create repeatable results these tests also minimize random load variability. This also keeps the wait time from queueing low allowing the machines to run to their full potential.

The problem is that real workloads can have I/O wait states, some paging, contention and unbalanced parallelism. Accepting an ITR metric as the preferred performance indicator without considering the difference between the real load and the benchmarks from which the metric is derived will lead to a bad estimation of the servers' performance. This is particularly true when comparing server types for IT solutions. The consequences are called "The ITR Trap".

Three Points of View

To develop better metrics than ITR we need to understand the performance from three points of view. They are:

1. The Vendor Point of View: How much work can this machine potentially do?
2. The Provider Point of View: How much work can I get out of this machine over the long run?
3. The User Point of View: How fast will my work get done?

The Vendor Point of View is well represented by ITR metrics and that is what they typically will provide. However, the Provider and User Vies are not well represented by ITR metrics unless the load closely matches the behavior of the benchmark used to derive the metric. That is:

1. The load does not create I/O wait states
2. The parallelism in the load is well balanced
3. The "threads" of the load do not share a lot of updated data
4. The variability of the load is low
5. The User can drive the machine and experience the full ITR

The Provider View

The Provider is interested in the business value of the work that the enterprise can drive through the machine. Users and business processes often create a randomly variable load to the server, they may also drive enough I/O to generate I/O wait states. Both situations will cause long term utilization of the server to be lower than the utilization achieved during the benchmark tests that generated the ITR. We define External Througput Rate or ETR as average long term throughput. It should be apparent that ETR is a function of ITR and Utilization.

ETR = f(ITR,U) (1)

If we assume that throughput is a linear function of Utilization and that ITR is the throughput rate at 100% utilization we get the estimator:

ETR ~ Uavg * ITR (2)

In today's world of multi-core and multi-threaded processors this assumption is not as good as it once was because of saturation (causes underestimation) and how core utilization interacts with thread activity (causes over estimation). However, for a starting point the estimator is good enough. It is also easy to understand and calculate.

ETR or its estimator should be used to calculate the business value of the machine to the provider

The User View

Users are not interested in how much work flows through the system; users care about how fast their work gets done. The metric for this is response time. If we think of tasks being accomplished by executing threads of work, then the users view of performance is the somewhere between the "Thread Speed" and the ITR" In most Cases the users of a server see performance that is closer to Thread Speed than the ITR.

In performance modeling the user performance metric is "Response Time". Response Time is the sum of Service Time and Response time. We write this as

Tr = Ts + Tw (3)

From the discussion about we also have:

1/ITR <= Ts <= n/ITR (4)

where n is the number of threads that can run in parallel on the machine and Ts is the "Service Time" of the work on the machine.

Wait time is the amount of time a task sits on a queue waiting for the processor to execute it. It is a function of service time, variability and utilization. We write this as

$Tw = Ts * c^2 * u/(1-u)$ (5)[1]

From this and equation (3) we have the following equation for Response Time

$Tr = Ts (1+ c^2 * u/(1-u)$ (6)

[1] This is the simplest "queueing model" that takes variability into account.

We now have a model for the user view of performance. This curve of this equation has a well known "hockey stick" shape. If we normalize Ts to one we can plot this as a function of utilization with c as a parameter.

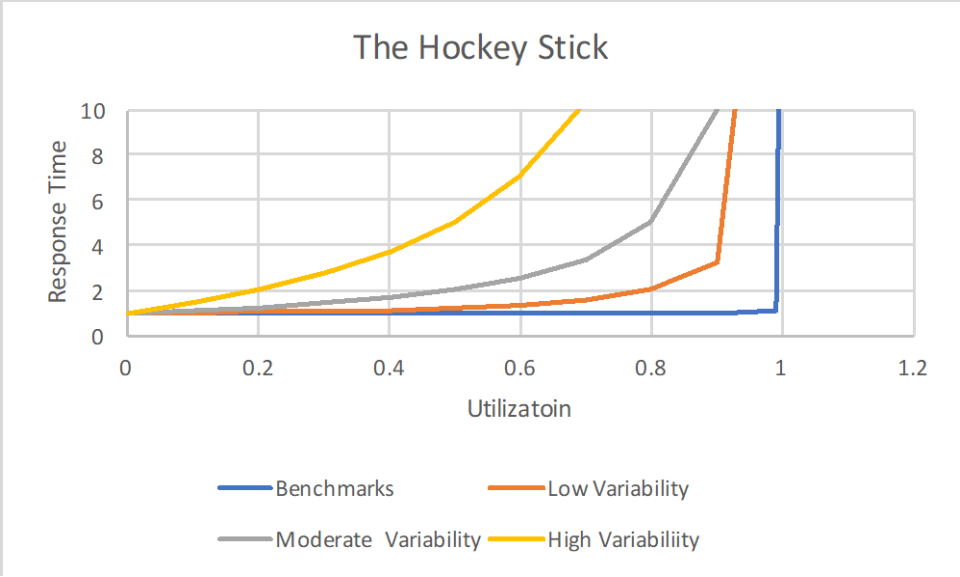


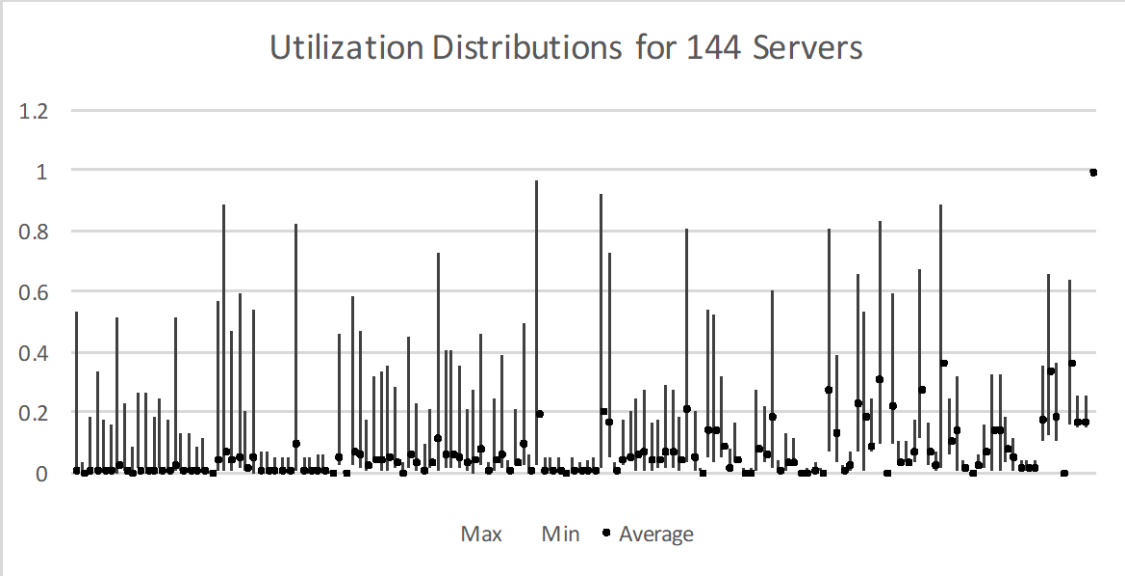
Chart 1.

When utilization is very high the response time becomes unbounded because of the (1 – u) in the denominator of the wait time model. At very low c, such as occurs in benchmarks, Response Time remains low until utilization nears 100%. At high variability (c >= 2), the Response time is more ramp like and becomes unacceptable at much lower utilization.

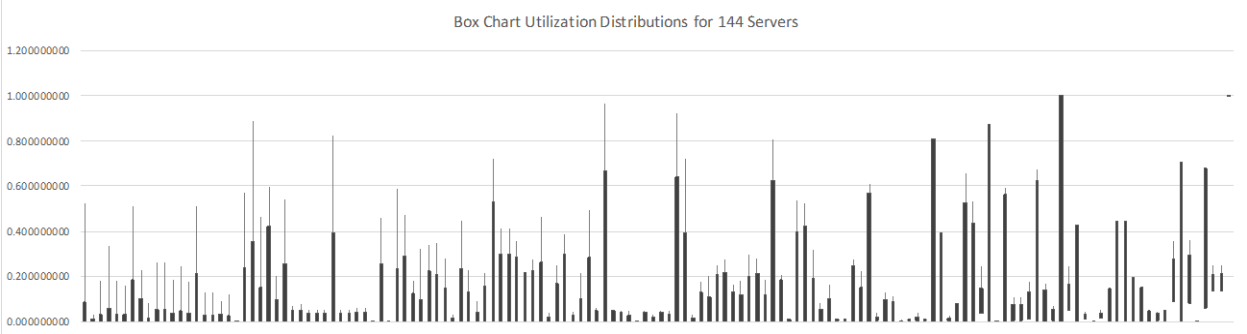
To avoid the ITR Trap we must understand utilization, because the metrics for both the Provider and the User depend on it. In fact, they must be traded off by design. This is because ETR→ITR when utilization is high and Tr→Ts when utilization is low. The only way to get both High ETR and Low Tr is to eliminate variability from the load. The load variability will drive utilization variability. This means we must understand more than the average and the peak, but also the distribution over time.

A Utilization Study

This paper will examine 144 servers and their utilization. Here are the distributions of utilization on the servers:



Notice how low the averages are. Almost all of the deviation from the average is in the "upper" tails. Here is another view of the distributions. These have a box from the minimum of k=3 or 1 and the maximum of k = -3 or 0. The maximum and minimum data points are represented by whiskers that are outside of the box.



Note that there are long whiskers between the box and the max data point for a significant number of these servers. Note that for almost all of the servers "0" is contained in the box.

The presence of the whiskers indicates that the maximum measured value of "k" for these distributions is often greater than 3.

Variability, Maximum Deviation, and Headroom

This brings us to the variability of the distributions called "c" in the equations above. The parameter c is called the "Coefficient of Variability" and is defined as the Standard Deviation / Average. We write this as:

c = Stdev(u)/Uavg (7)

"Rogers' Equation" relates c to Uavg and k_{max}.

$Uavg = 1/(1+ k_{max} * c)$ (8)

Recognizing that the short term peak of a utilization distribution is typically 1 and rearranging equation 8 we get

Upeak/Uavg = 1 + k_{max} * c (8.1)

Check: Upeak = Uavg * (1+ k_{max} * c) → Upeak = Uavg + k_{max} * Stdev which is how we define a random variable. Noting that k_{max} * Stdev is the maximum measured deviation above the average, we define this value as the "Headroom". We write this as

Headroom above average = k_{max} * Stdev (9)

We then divide Headroom by utilization to get normalized headroom:

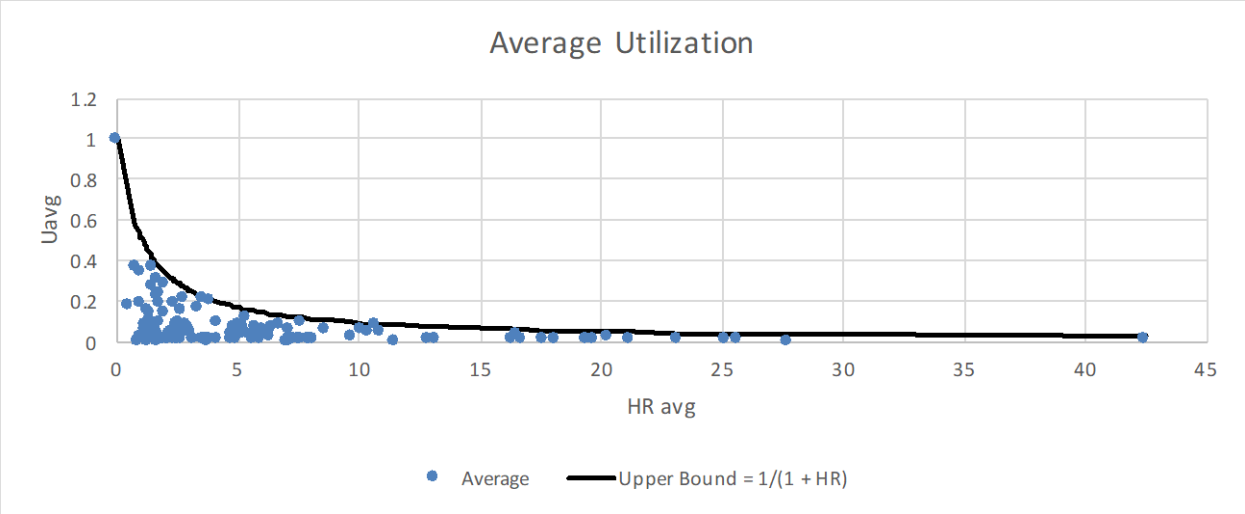
HRavg = k_{max} * c (10)

It can also be shown that the HR = k * c at any value of utilization u, is

HR(u) = (1-u) / u (11)

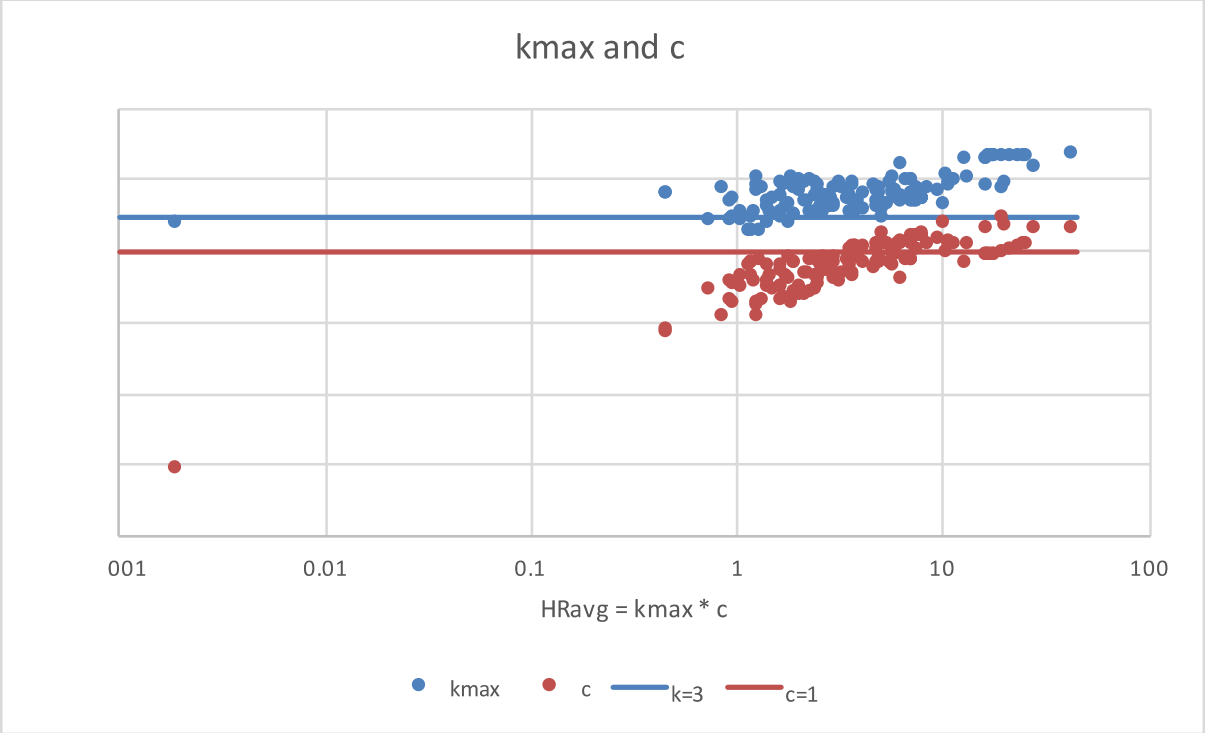
This is the inverse of the utilization term in the queueing model for Wait Time (Equation 5). This means that HR(u) has an inverted hockey stick shape.

Plotting Uavg v HRavg we get



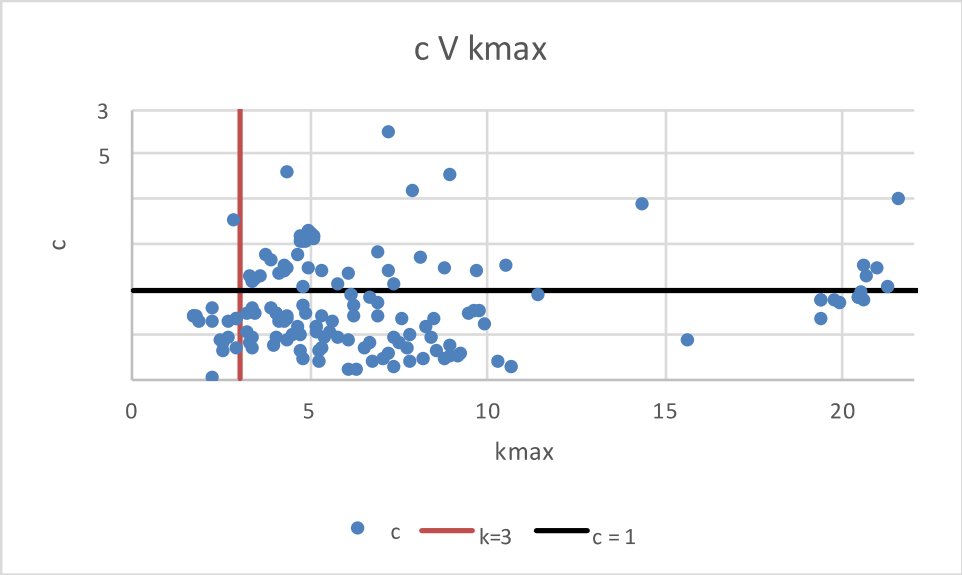
Utilization has an inverse relationship with HR. Since the maximum Peak is 1 then 1/(1+HR) is the upper bound on utilization. If the peak is at less than 1 the average utilization will fall below the line. Here it very clear that based on peak utilization, many of these servers will have ETR well below their ITR.

Plotting k and c v HRavg for our servers we get the following chart.



This shows that kmax is well above 3 for most of the servers, and c is quite low. High k with low c is an indicator of "bursty" load driving high spikes but averaging low utilization. This is one of the sources of Low ETR on distributed servers and a driver of the emergence of virtualization.

This can be seen by plotting c v kmax

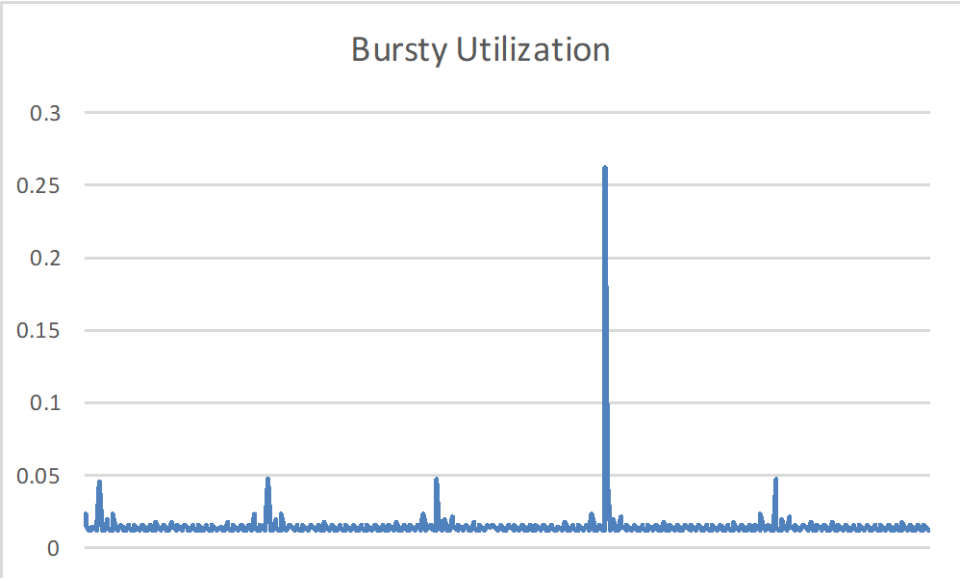


Points to the left of k = 3 are to the left of the red line. These servers may have distributions that approach "Lognormal" distributions. Points below c = 1 are below the black line. These servers have distributions with low variability. Points to the right of k = 3 and below c = 1 are bursty loads. These loads show low variability most of the time but have spikes of high utilization and temporarily have high variability near the spikes. Points to the right k = 3 and above c =1 have high variability and low average utilization.

The points to the right of k=3 have "Fat" and/or long upper tails. This drives a need to be conservative in attempting to drive ETR toward ITR. The problem is that the relatively high probability of large deviations above the average makes Response Time issues more likely than desired if the utilization is pushed too high.
For this type of usage pattern it is best to design for kmax = 4 or 5 or higher rather than k = 2 or 3.

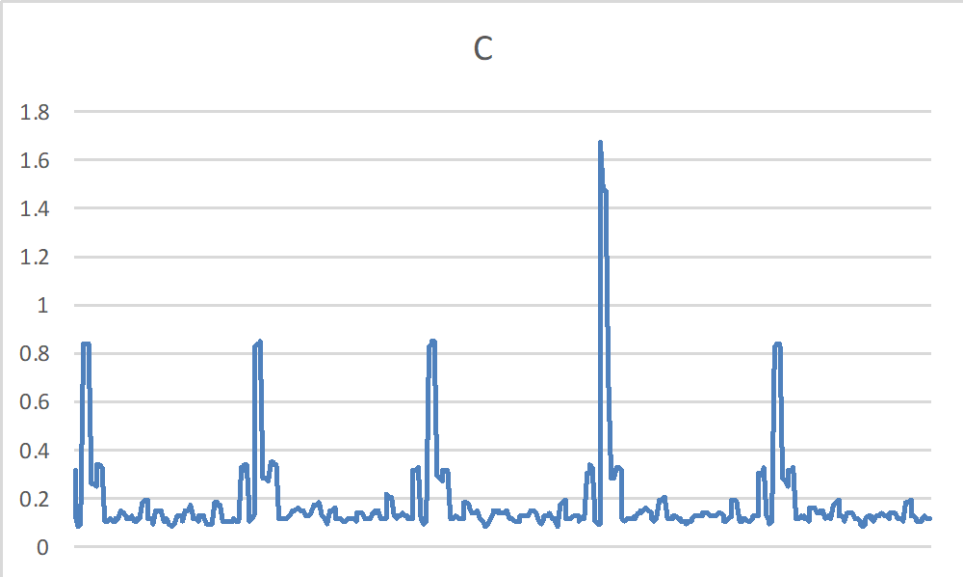
For the bursty distributions it becomes necessary to look at the intervals near the bursts and ignore the long intervals of low utilization when determining the utilization statistics from the data. This will avoid underestimating c and Uavg as they make a difference to the user experience.

Here is a plot of a bursty workload time series.



- The average utilization is very low: Uavg = 0.0137
- Umax is also quite low Umax = 0.2615
- The Standard Deviation is low: Stdev = 0.0121
- c is < 1 c = 0.8808
- kmax is very high kmax = 20.51

Here is how c varies over time. We plot a 1 hour running value of c based on 4 fifteen minute intervals.



Notice that c reaches nearly 1.7 during the time near the highest spike of utilization. Clearly variability is high during the period that matters. Here are the statistics for the peak hour:

- The average utilization is higher but still low: Uavg = 0.0745
- Umax is unchanged Umax = 0.2615
- The Stdev is higher Stdev = 0.1247
- c > 1 c = 1.6726
- k = is much lower kmax = 1.5

For bursty loads, we must design for the peak and average of busy periods rather that the peak and average of the entire set of intervals.

Response Time is Hard to Measure

It is often difficult to determine service time from production data like utilization or even throughput numbers. A performance indicator that does not depend on measurement of Service Time while indicating the effect of utilization on Response Time would be very helpful. The people at TeamQuest created such a metric which they call TPI[2] that does just that. They use a sophisticated queueing model and extensive real time collection of

[2] TeamQuest also refers to this as the "Stretch Factor"
http://www.teamquest.com/en/products-services/teamquest-predictor/

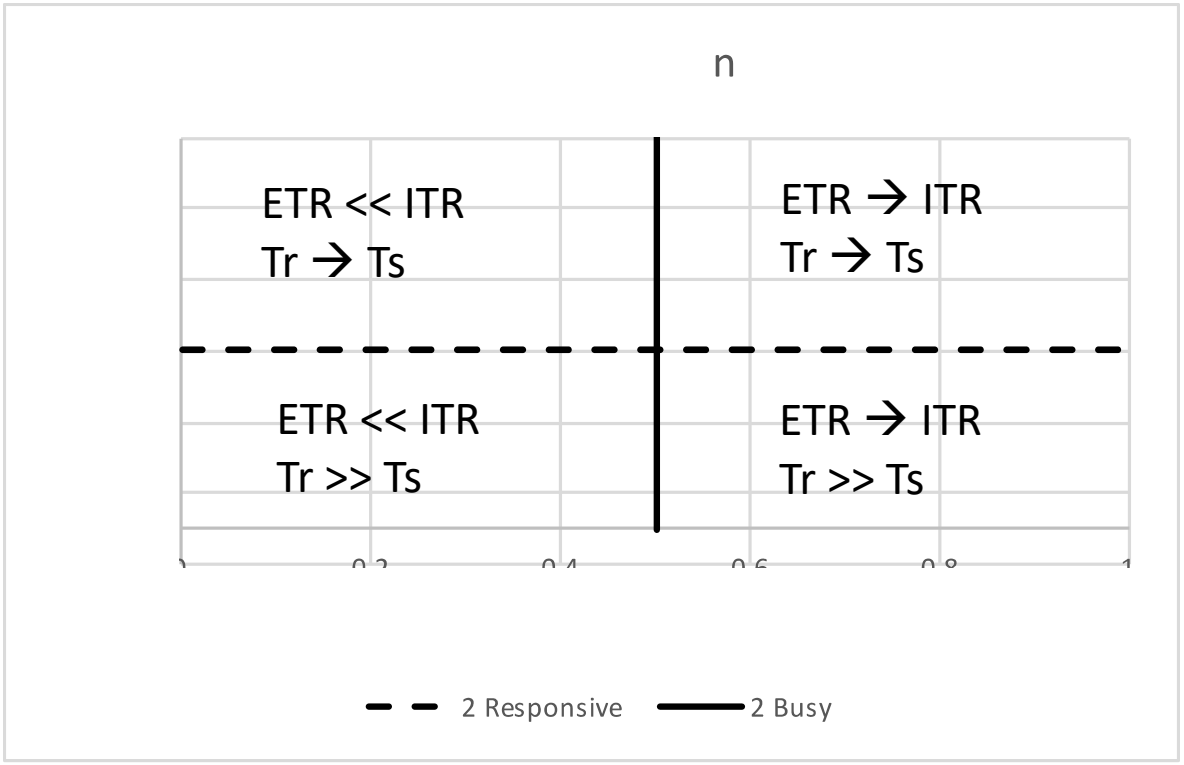
production data to create it. Using our simple queuing model for Wait time (Equation 5), we can create an estimator which is useful for gaining insight into how a machine will behave under load.

We do this by dividing Response Time by Service Time, creating a ratio metric which is based only on utilization and variability. We call this metric UPI (Usage based Performance Indicator).

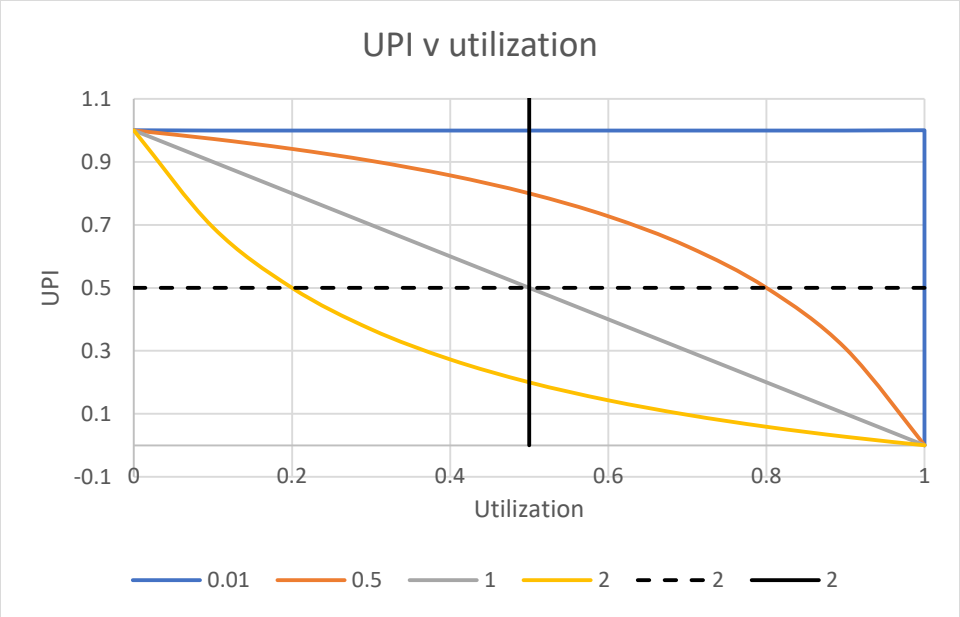
UPI = Ts/Tr (12)

The idea here is that UPI will approach 1 as queueing effects diminish and approach 0 as they increase. We apply this to our simple queueing model.

UPI = Ts/Ts * (1 + c^2 * u/(1-u)) = 1/(1 + c^2 * u/(1-u)) (13)



If we plot UPI v utilization and divide the chart up into 4 quadrants, it is quite clear that when both UPI and utilization are above 0.5 the ETR will approach ITR and Tr will approach Ts. If possible, we want our machines to operate in the upper right corner. Here is a plot of UPI(u) with c as a parameter.



Notice that a machine will operate in the upper right corner only if the effective variability is below 1. Based on this chart, it would seem that the servers we are studying have fallen into the ITR trap. The question is how do we reduce the variability and move into the preferred quadrant of operation?

Consolidation

Part of the problem comes from the very nature of distributed systems. It turns out that the variability of the load decreases with consolidation and increases with distribution. Recall Rogers' Equation (8)

Uavg = 1 / (1 + k_{max} * c) (14)

If we distribute a piece of work on s servers this becomes:

Uavg = 1 / (1+ k_{max} *c * SQRT(s))) (15)

This assumes that the work on the original server is divided evenly among the s identical servers.

If on the other hand we have n identical server loads that we put one server the euation becomes

Uavg = 1/ /(1+ k_{max} * c / SQRT(n)) (16)

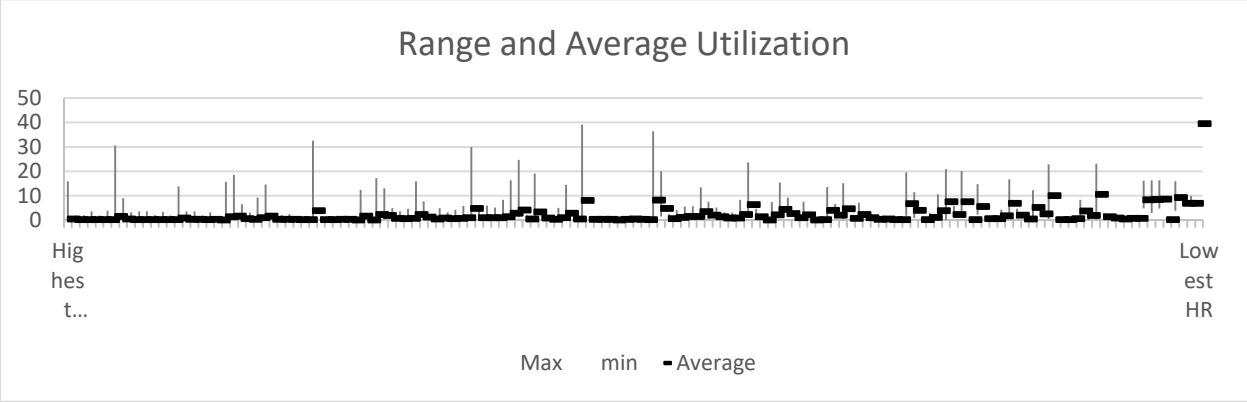
This is the same as saying that the c gets smaller when loads are consolidated. In some cases this is enough to push the c below 1 and move the operation into the preferred quadrant of our chart.

In reality, most of the loads to be consolidated are not identical nor are the servers that they run on. While it is possible to work this out analytically for loads with normal independently random distributions, there is usually some correlation due to interactions and the nature of the workday. Also, as we have seen, Utilization distributions are rarely normal. Thus, any analytic solution that we have like Rogers' equation is an estimate.

We can see the effect of consolidation on adding the utilization data on an interval by interval basis and watching how consolidation unfolds as each server is added. The first thing we have to do is to convert each server

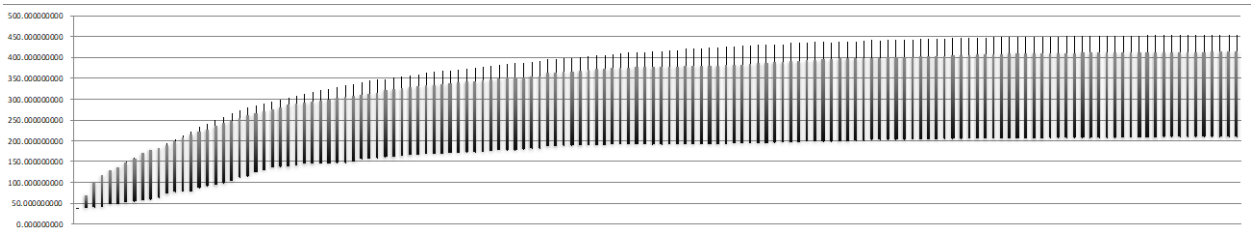
utilization into a common Throughput Metric. This can be an ITR metric which we multiply by the utilization values to get an ETR estimate for each interval. We will use a Metric called TP which is simply the cores on the server x the clock rate for the server. This is not a really good ITR metric but it will suffice for our purposes.

Here are the TP distributions for our 144 servers

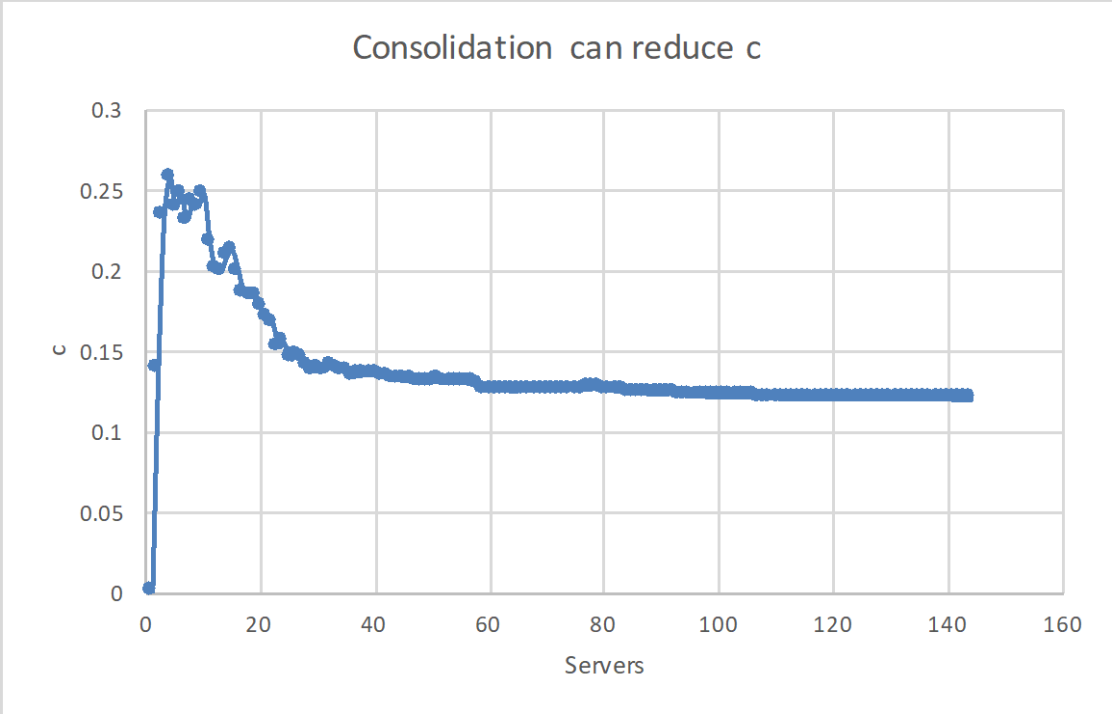


This is the same chart as the utilization distributions but it is scaled by the individual server TP ratings.
The next step is to analyze the sum of all the interval data and discover the peak interval of the sum and the individual server contributions to that peak. We then order the server data from the highest contribution to the lowest contribution. We then start with a single server which is the largest contributor. The next column contains the first server data plus the data for the next highest contributor. This is repeated until the last column contains the consolidation of all 144 servers in our study.

Here is the box chart of the progressive consolidation distributions

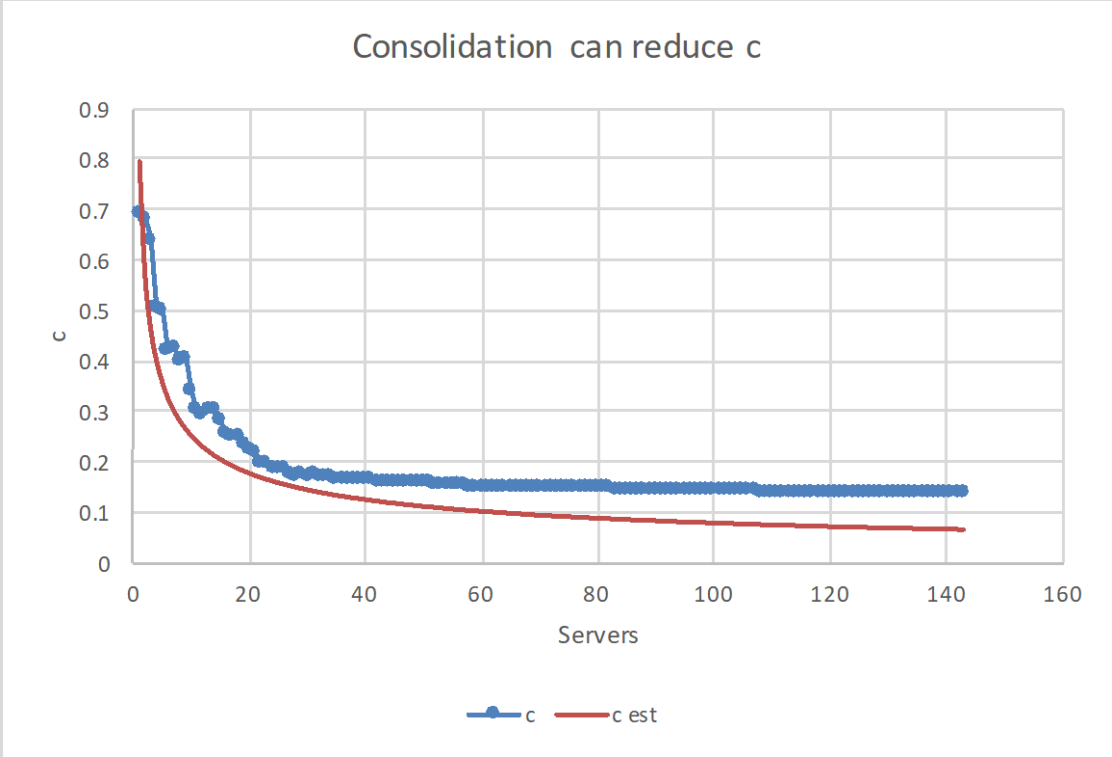


Notice that there are still whiskers indicating a fat tail, but they are much shorter than the unconsolidated data.



This is what happens to c as the consolidation progresses. The reduction of c for this workload is not as it would have been estimated by Rogers' equation. This is because the machine with the largest contribution to the peak runs "hot" throughout the whole measurement period and therefore has very, very low c as an initial value. This causes the variability to climb until about 5 servers are consolidated, and then c begins its inverse slope. Notice that all these consolidations would run in the "preferred quadrant" for ITR and Response Time.

Here is the plot if we leave off the first server from the consolidation.



The estimated variability c est is calculated by multiplying the average c of the individual servers by the square root of the number of servers consolidated (horizontal axis). Note that the estimate is optimistic. It is a lower bound. The data will approach it but correlation of workloads, varied size and variability of the individual loads and distributions that are not normal, keep the consolidation from reaching the estimated value.

Transition: Thread Limited to Cache/Memory Limited Performance

Consolidating loads affects more than the core utilization. For example, each server's workload uses memory space and cache space. Memory and Cache space utilization is less variable than core utilization. As a result, consolidation of workloads causes average cache and memory usage to grow faster than core usage. Eventually the cache miss rate increases and the memory and cache interconnect "nest" delays increases. Eventually, paging (swapping) starts to occur. This reduces scaling and thread speed causing the machine to saturate at lower utilization. These effects eventually cause a transition from being CPU thread bound to being memory/cache bound.

"Enterprise" class servers such as Power 8, SPARC M Series and System z Mainframes have significantly more cache and/or memory than two socket SPARC Tseries, Intel Servers, MIPS, and ARM servers. Eight Socket Intel servers fall between the distributed and Enterprise servers, because they can have large memory but still run out of cache space. Enterprise Class servers also have more I/O connections and bandwidth. As a result, Enterprise Class servers can consolidate more workloads per CPU than the typical mass produced "2 socket" servers used in distributed clusters and some major public cloud offerings.

This defines our final metric that we call N. We use the size of the highest level cache as an estimator and choose a base machine to normalize N to 1. All machines with more cache than the base machine will have N > 1. One could use total memory or some more sophisticated measurement for N but this seems drive enough insight particularly for planning, design and analysis.

Four Estimators used to avoid the ITR Trap

ITR – This is the benchmark based result that leads you to the trap if you don't use the others

The estimator is TP:

TP = Cores * Clock Rate

TP = Threads * estimated Thread Speed

ETR – This represents the Providers' view of performance. It is aligned with business value.

The estimator is ETR(u)

ETR = ITR * Uavg

Tr – Response Time represents the users' view of performance

The Estimator is UPI:

$UPI = Ts/Tr = 1/(1 + c^2 * u/(1-u))$

N – Capacity represents the number of normalized virtual machines the server can contain

The estimator is:

Total Cache/Total Cache of base machine

We have shown that plotting UPI v Utilization with c as parameter, that c must be low to operate at high ETR with low Tr. We also showed that consolidation reduces variability, allowing workload to move into the preferred operating quadrant of UPI v utilization. From Rogers's equation, we can understand average utilization. Then ETR can be Estimated, based on the ability to consolidate N servers. We have shown that this is an estimate and that the actual consolidation of load can indeed reduce c. However, quantification is best done by using the summation of interval data rather than analytic math.