**The Association of System Performance Professionals**

The **Computer Measurement Group**, commonly called **CMG**, is a not for profit, worldwide organization of data processing professionals committed to the measurement and management of computer systems. CMG members are primarily concerned with performance evaluation of existing systems to maximize performance (eg. response time, throughput, etc.) and with capacity management where planned enhancements to existing systems or the design of new systems are evaluated to find the necessary resources required to provide adequate performance at a reasonable cost.

This paper was originally published in the Proceedings of the Computer Measurement Group's 2009 International Conference.

**For more information on CMG please visit http://www.cmg.org**

# How do you measure and analyze 100,000 servers - daily?

Charles Loboz, Steve Lee, James Yuan

Microsoft Corporation, Global Foundation Services

Online service provisioning requires large numbers of servers. It is necessary to monitor them, evaluate if and when they are overloaded and how many of them are underutilized. This is an enormous logistical and analytical task. Commercially available tools are unsuited for such unique needs, and human analysis of large-scale systems is problematic. We have designed a system capable of collecting data from a large number of servers and providing a meaningful and timely analysis - on a daily basis.

## 1. Analysis on a large scale

Online service provisioning requires tens to hundreds of thousands of computers. The capital expenditure costs hundreds of millions of dollars. Keeping track of how that investment is really used and utilized is necessary - but hard. Commercial tools were not designed nor adjusted for the very specific logistical requirements large-scale server environments. The sheer volume of performance data amounts to terabytes per day. This huge volume of data presents problems both in data storage and in reliable, efficient data analysis.

The main goal of any analysis of server performance is usually the same as years ago: to find out how well the server is utilized. In a data center we want to know which servers are idling, which are running dangerously close to impact service levels - and what are performance bottlenecks. This goal is difficult, if not impossible, using traditional methods.

### 1.1. Traditional approach to analysis

The traditional approach to analyzing server performance is centered on a single server and uses averages and daily histories of key performance indicators – usually utilizations of key server components: processors, disks, network cards [1],[2].

It is known that averages may be hiding dangerous (to service level) peaks. Therefore, traditional performance analysis packages provide daily histories for all server components, similar to the ones on the plot below. System administrators will inspect such daily histories for processor, disks, and network cards. Human inspection of several plots per server is tedious when tens of servers are involved - and impossible for data centers with tens of thousands of servers.

Daily histories also have the potential to mislead, because of the aggregation level. On the plot below (Illustration 1) the hourly history shows the peak hour utilization of 0.5 - relatively safe. But a look at the same data, aggregated over 120 second intervals shows large number of samples above 0.8 utilization - potentially disturbing. The hourly plot looks clearer, but misses a large number of data important situations when the processor utilization was well above 80% range.

We have to keep in mind that even a small number of samples with high utilization is dangerous - the misleading-by-aggregation can be deadly. For one day, seven samples constitute 1% of total samples. If we have seven samples with over 90% utilization in one day, it is highly likely that transactions executed in that period will have extremal response time. Thus we have high probability of failing a service level agreement stating that 99% of transactions should have response time below some limit...

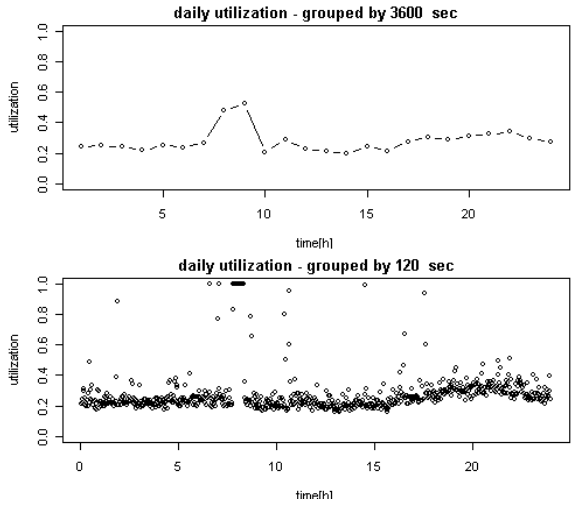


*Illustration 1: Daily utilization history at various aggregation levels*

True, the cost of missing something important is usually worth a fraction of a server's cost - and one server is cheap. But for 100,000 servers even a fraction of their cost amounts to millions of dollars. Thus the traditional approach to server performance analysis does not scale.

### 1.2. The PIF approach

We have developed an approach which enables us to quickly identify underutilized or problematic servers in our data centers with minimal human intervention.. The approach consists of multiple interacting strategies. It is very efficient operationally, both in processing time and storage.

*Section 2* describes the data collection and reduction stage. *Section 3* describes the definition and use of Performance Impact Factor (PIF), a single metric summarizing server stress levels. *Section 4* describes how PIF is used for summarizing multiple servers and *Section 5* discusses properties of PIF as a metric.

# 2. Data collection and reduction

### 2.1. Data volume

Monitoring tens of thousands of servers creates daily terabytes of raw data – that needs to be managed and analyzed. Such data volume creates a problem for analysis - it will not even fit comfortably into a database [3],[4],[5]. To extract data of one server from that ocean of data, process it - and do it 100,000 times a day creates some challenges. This is especially so when we need to produce – daily - a report from several data centers in multiple time zones – and we have a deadline.

### 2.2. Data collection and histograms

On each data center server we have deployed a small utility - Data Collection Agent. It collects the data at specified intervals, currently every 15 seconds, and writes them to a disk file. The utility has minimal impact on server performance - less than 0.1% of processor utilization on the server. Depending on server configuration we collect from 100 to 500 counters per server.

The hourly data files from all servers are collected into the central store (called Data Garage). For each server we compute a daily and hourly synopsis - and this is where the difference from the traditional approach begins. For key server components - processors, disks, and network cards - we compute a 10-bin histogram. For example, for the processor, we summarize how many times we had a sample with utilization within defined ranges (0...10%, 10...20% etc) – for examples see the picture in section1.3.

### 2.3. How does it differ from the traditional approach?

Traditionally we would try to keep both the daily histories (requiring the storage for 5760 values per day) and compute a synopsis. Traditional synopsis would consist of average, minimum, maximum, occasionally a standard deviation of the original data points. Unfortunately, such numbers are of limited use in analysis of computer systems. Daily averages are hiding important peaks; minimum utilization is likely to be 0% and maximum 100% for most computers for data collected at small time intervals. Standard deviations are useful for normal distributions, but distributions of most utilization coefficients in computer systems are anything but normal.

A histogram gives us the shape of the distribution – in particular the fraction of time spent in extreme situations. If we collect, say 240 data points per hour, the histogram can tell us that 20 of them were above 90% utilization [interpretation: 5 minutes of this hour was spend in overload] and 200 data points had utilization below 10% [interpretation: server mostly idling]. Histograms allow us spot the proportion and severity of overload situations better than daily histories and, also, to differentiate between situations seemingly identical when looking just at average utilizations.

A histogram can be computed on-the-fly, even before storing the data in the database - like traditional statistics. (This is operationally important, since computing statistics on data already stored in the database is costly in large databases). A Histogram

does not increase processing nor storage requirements - daily histogram requires only 20 bytes for storage. Storing classical information requires 4 to 8 bytes per number, so for raw data we need 20-40kb and for typical synopsis 40-60 bytes.

Traditional history plots do show when peaks occurred during the day. Histograms inform us only how bad the peaks were. But traditional plots require storage of thousands of data points when a detailed picture is needed, while histograms require only 10 small integers. In addition, from the system administrator point of view - to spot the overload or to support the next buying decision - the size of the peaks and how long they lasted (which we can get from histograms) is more important than when do they happen during the day.

### 2.4. Problems with histogram analysis

Illustration 2 shows daily histograms for two servers. The data was collected at 15-second intervals, giving 5760 samples for each server for each day.

Each histogram bin contains the number of samples within given utilization range. The top picture (Illustration 2) shows that for server A there were about 200 samples with utilization between 0% and 10% (the leftmost bar). The bottom picture shows that, for server B, we had about 3500 samples with utilization below 10% - so the server B was idling most of the day.

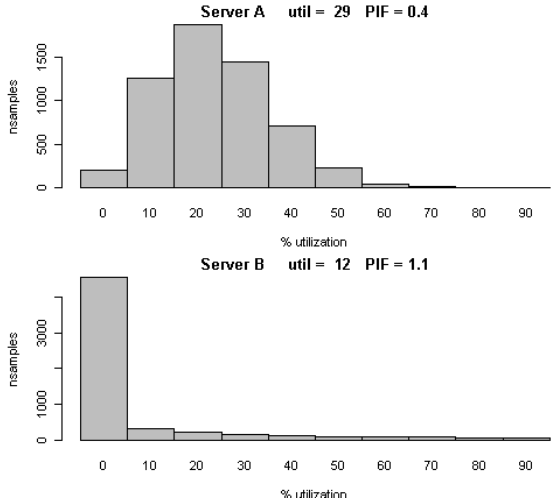


*Illustration 2: Utilization histograms for two different servers*

System administrators have to eyeball such histograms and manually decide which ones are problematic. This can lead to misinterpretations, as shown on the two histograms below.

A casual glance could suggest that Server A, with the average daily utilization of 29%, is worth more attention than server Server B – with the average daily utilization of only 12%. But, for a trained system administrator, the histogram of Server B will look alarming. And not because it has a very small – barely visible - number of high utilization samples above 80% level.

The distribution of Server B utilization data has what is known as a 'fat tail'. The fat tail concept is not related to the number of samples at the tail end of the distribution – but to how quickly the number of samples at the tail end of the distribution drops off in comparison normal (Gaussian) distribution. The presence of fat tail signals that the distribution of incoming requests is neither Gaussian nor Poisson – with all the mathematical and practical consequences of it. In system administrator's terms it is suggesting that the the requests to the system are arriving in irregular bunches. This, in turn, means that any increase in load preserving the load distribution is likely to grow disproportionately the right side of the histogram. (This effect is slightly similar to cascading locking in databases – when the number of locks goes over some critical threshold, the flow of processing goes haywire).

In other words, for Server A, the growth in load will most likely result in gradual and orderly shift of the histogram to the right and will keep the fraction of catastrophic overloads low. For Server B any growth in the load is likely to instantly increase the fraction of dangerous overload situations;

Note that so far we have described the analysis of a server by looking at one histogram – preferably the one for the most critical component of the server. But to find out which component is most critical we need to analyze all server components – processors, disks, network cards – thus we have to look at multiple histograms per server.

The smaller question is: how many system administrators are trained to spot such situations? The big question is: can anyone do that for 100,000 computers - daily?

# 3. Performance Impact Factor

The underlying problem is that histograms come in all shapes and we want to translate that variety into a single number - so we can reduce the need for human visual inspection. This number should summarize histogram information and alert us to server performance problems *in all situations when a trained system administrator would do so while eyeballing the histograms*.

The difficulty is that utilization is related to performance (response time, service level) in convoluted, nonlinear and non-intuitive ways. Importantly, we do not really care that much about

utilization - we care a lot about the impact of utilization on performance.

The naive approach would be to construct a set of rules, like 'we have a problem if 1% of counts is in 90%+ utilization range'. Naturally, that would require defining also similar threshold for 80% etc. And how to evaluate situations when we have some in 90% range but none in 80% range and plenty in 70% range?

## 3.1. Definition of PIF

We define a metric, called Performance Impact Factor (PIF), giving a one-number summary of the 'meaning' of the histogram to a system administrator. *PIF is a normalized weighted average of utilization.* (In this section we consider PIF for a single server component only – for example a processor).

That normalization and weighting are applied to histogram counts and normalized on per-count basis, so we can compare PIF for histograms with differing total number of counts. To keep the index values small despite response time growing faster-than-linear with utilization at high utilizations, we take logarithm base 10 of that weighted average:

$$PIF = \log 10(\frac{\sum_{k=1}^{10} w_k h_k}{\sum h_k}),$$

where $h_k$ is the number of counts in the histogram bin and $w_k$ is the weight associated with that histogram bin.

The key concept here is that *we are after the possible impact of utilization on performance* and not after average hourly or daily utilization.

From queue-theoretic considerations, as well as from daily experience, we know that the performance impact depends non-linearly on utilization. So we construct a set of weights for the number of counts in each bin and take the logarithm of weighted average.

The weights must satisfy certain obvious conditions imposed by the problem we are trying to solve - the weights should be somewhat proportional to the impact that component utilization has on performance.

The starting point could be queueing-theoretical formula for response time $rt(u) = 1/(1-u)$, but we are mindful of the importance of the variance of response time, so weight coefficients proportional to $1/(1-u)^2$ are more useful starting point. Also, these basic formulas apply to single-proccessing-units only – the formulas for multiprocessing units are less adapted to practical use – but can be approximate by using simply higher power.

There are various other factors which push the practical weight function higher than the power of two - like the prevalence of multi-processor servers in data center populations, existence of disk arrays and some characteristics of disk performance counters. We have also observed in practice that using $1/(1-u)^3$ gives better differentiation in the middle utilization range while retaining good differentiation at the edges.

This set of weights can be replaced with many others, depending on more specific needs. Calculation of weight coefficients can be made more precise and server-element dependent, or even customized for given server taking into account details of its configuration and technical characteristics of its elements. Number of processor cores, size of L2 and L3 caches, number of spindles in the disk arrays – all that could be translated into weight coefficients.

## 3.2. Examples of PIF

Our set of weights is computed using the cube-proportional function, for utilization values at the left-end of each histogram bin. That gives $w_1 = 1$ for histogram bin counting utilizations between 0% and 10% and $w_{10} = 1000$ for histogram bin counting utilizations between 90% and 100%. Normalization by the total number of counts and use of logarithm base 10 results in the PIF range from 0 to 3.

To facilitate more intuitive communication we call the PIF range from 0 to 0.1 'stone cold', from 0.1 to 0.3 'cold', up to 1.0 'warm', up to 2.0 hot and over 2.0 'overloaded' – the ranges are somewhat arbitrary, but align well with our experience.

The previous histogram picture (Illustration 2) had shown daily histograms for two servers: Server A has average daily utilization of 29%, but its PIF value is 0.4 - in the lower range of 'warm' category. Server B has much lower average daily utilization of 12% but its PIF value is 1.1 - in the 'hot' category. That is in line with what a trained system administrator would conclude, as per earlier discussion.

The picture below (Illustration 3) shows daily histograms for two servers with identical average daily utilization of 45%. However, for server C the PIF value is only 0.8 (warm) while for server D the PIF value is 2.0 - in the 'overloaded' range.

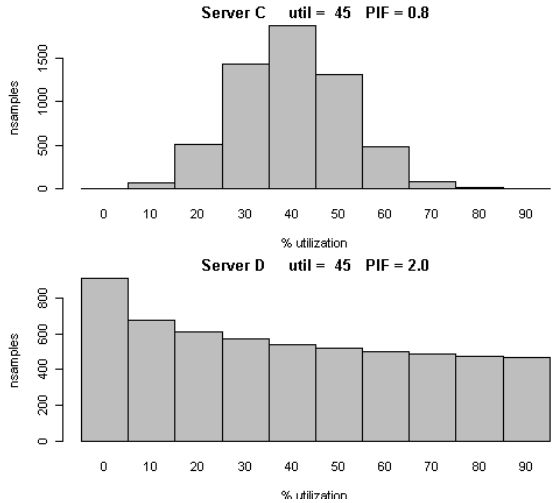


*Illustration 3: Same daily utilization - warm and hot servers*

### 3.3. Additional considerations

There are some matters of general importance, but outside the scope of this paper.

Firstly, the weights can be chosen based on some other considerations, not necessarily related to queuing. For example we could commercial loss associated with decreased service level.

Secondly, the weights can be applied directly to raw 15-second data samples – but that would limit the flexibility. If we wanted to compute PIF using some other formula we would have go go back to raw data, which would be costly. It is much cheaper to compute on histograms than on raw data – and we still retain the option of applying multiple ways of PIF computation to the same base data.

Thirdly, introduction of the logarithm base 10 is for the readability only, since the base weighting is strongly non-linear with underlying measure. The essence of the approach is to combine a weighting function with the base utilization.

## 4. Analysis of multiple servers

How do we compare multiple servers? A server has multiple components - processors, disks and network cards. We compute PIF for each such component separately. How do we compare the set of multiple PIFs from one server with similar set of PIFs for another server?

### 4.1. Multiple components of a single server

The description so far dealt with PIF of a single-component. Each server contains multiple components – we must consider at least processors, disks, and network cards. How do we compute PIF for each component and for the total server?

The processor. We compute one PIF for all processors in the server. We consider multiple processors or processor cores as a single device with multiple queuing stations. The operating system balancing is usually so good, that treating each processor separately is rarely justified at these timescales. Processor utilization is the base measurement here.

The disk. We compute one PIF for each physical disk in the system. We need to treat each disk as a separate queuing entity, as the operating system does not balance disk accesses between disks. Disk utilization is more difficult to define and measure than processor utilization – at least for performance impact purposes. We use percent of the time the disk channel is busy as a proxy for disk utilization.

The network. We compute one PIF for each network card. Network card's utilization is defined as the ratio of the number of bytes transferred to the maximum number of bytes that could have been transferred (based on card's rated speed).

Thus for each server we have several PIF values: one for processors, one for each physical disk, one for each network card. We need to combine these component-PIFs to obtain one PIF for the whole server – the server-PIF.

Computer systems are mostly queuing systems, which means that the main throughput/service level constraint comes from the most overloaded component. So the algebra on component-PIFs is simple - for each server we can consider only the maximum value from its set of component-PIFs. Thus, regardless whether the server is bottlenecking on the processor or network card, we simply identify the overall server performance impact with the performance impact of its maximally stressed component.

On the other end of the spectrum, a server with very low maximum PIF will have to have all its PIFs very low - thus a low maximum PIF is a good indicator of an underutilized server. Importantly, because of the way PIF is computed, very low PIF means not only that the server utilization is low but also assures us that there are no dangerous peaks hidden in the average utilization.

Having just one number indicating server's potential performance problems allows us to avoid visual

inspection of components of each individual server. We do not have to look at daily histories – nor even at histograms. PIF allows us to be very selective about what information needs to be looked at.

### 4.2. Multiple servers

We sort all servers by decreasing server-PIF - the servers at the top of the list will need tuning, expansion – or some offloading. The ones at the bottom of the list will need additional load assigned.

We tabulate the servers by their function (like Web server, database), owner (like Hotmail, Messenger), physical location (data center, rack) - or all of them - and establish precisely the overload or under-utilization areas.

PIF makes creation of an overview of 100,000 servers easy and fast. A database table containing 100,000 rows per day is trivial - laptop-size. Even the database table of hourly summaries needs to hold only 2,400,000 rows per day - a trivial amount by database standards. Even keeping all component-level PIFs requires database size of few megabytes per day.

Since the daily synopsis of all servers is so small, we can easily keep the historical information and produce weekly, monthly or yearly summaries as well as analyze daily and hourly trends. That small representation allows us also to define new trending summaries for services (Line of Business) and data centers. We can analyze daily growth of the number of overloaded servers in each group as well as the drop in the number of underutilized servers.

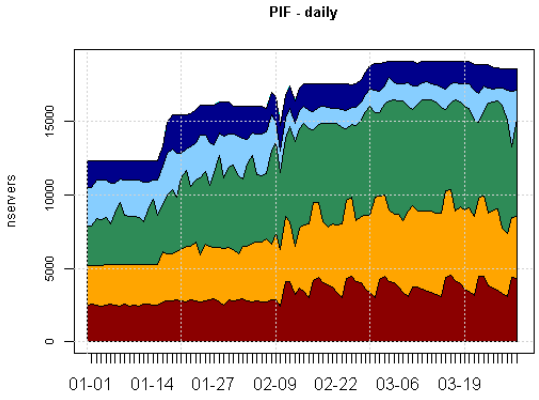


*Illustration 4: quarterly history of daily PIF distribution for one sever group*

The picture above (Illustration 4) shows the daily results for a group of about 20,000 servers over three months. Each vertical line consists of several segments with different colors – at the top deep blue means stone-cold servers, at the bottom red means overloaded servers. We can see the day-to-day variability of each type of server stress - together with the number of servers in that group. The relatively constant thickness of top band shows that the number of stone-cold servers had remained constant during the quarter. The bottom band shows that the number of overloaded servers grew, so did the number of 'hot' servers (second band from the bottom).

More specialized plots (not included here) can give easy to interpret visualization of whether the servers are overloaded on disk or processors and how tunable the overloaded servers are.

## 5. PIF as a metric

Most importantly, PIF combines each component's utilization and peak information. PIF is a projection from two-dimensional space (utilization, peaks) into one-dimensional space (performance). This is augmented by the PIF algebra on server's components. As with all dimensionality-reducing projections, interpretation of the results may create pitfalls for the unwary.

For example: between-servers utilization can be sensibly averaged - but between-servers PIF obviously cannot. We can state that a server with utilization 50% is loaded twice higher than a server with utilization of 25%. We cannot apply similar reasoning to PIF values, as they include the effects of peaks, possibly differently shaped on each server. Thus PIF is a non-additive, non-linear and non-decomposable metric.

PIF is a single-server metric - there is no point comparing average PIFs for two groups of servers. What we can sensibly do is to compute, in each server group, a fraction of overloaded servers to estimate which group is overloaded most.

We need also to understand how PIF for a single server should be interpreted. The underlying characteristic of PIF is that it offers more definite information at its low values than at its high values.

Low PIF value means that the server is not doing much – so *all* its elements are not doing much.

But the high PIF value means only that *some* element or elements are overloaded. The server may have almost-idle processor, 5 almost-idle physical disks, and one physical disks being 100% busy. As we select component with maximum PIF to describe the server, this server has PIF 3.0 - maximum overload. But balancing the disk load - usually possible - could bring that server to the 'warm' category. Thus high PIF means that the server is performance-impacted but it does not mean it is not tunable.

In other words: there is only one way a server can be stone cold – *all* its elements are underused. But there

are many ways in which the server can be overloaded – any combination of its components may be overloaded. To get more information we need to look at component-level PIF.

PIF is not a silver bullet. It applies to transaction-oriented servers, but not to batch processing ones, like Web-crawlers. On batch servers, 100% utilization has different meaning and does not indicate potentially infinite response time or impacted service level. Thus the PIF computations should not be applied to data collected when the system was performing batch operations (like backup).

Sampling frequency is important - PIF value depends on it. Appropriate care must be exercised when comparing PIFs from samples collected at different sampling rates - the normal rules based on the analysis of stochastic systems apply.

# 6. Summary

We have introduced a metric – Performance Impact Factor. It summarizes in one number the way server performance may be potentially impacted – taking into account multiple server components and peaks in processing load. That metric avoids falling into the aggregation trap – missing peak load situations averaged by daily or hourly averages.

That single PIF number is used to tabulate tens of thousands of servers. It facilitates server categorization and discovery of overloaded or underutilized servers, without the requirement for human review of individual performance plots.

The computation of PIF is operationally simple. Most of that simplicity is due to (a) using histograms instead of raw data or averages and (b) PIF being a weighted sum of histogram counts. That allows us to handle the analysis of performance of tens of thousands of servers on just a few servers. The PIF data storage requirements scale to today's large-scale server populations.

The metric is flexible – both conceptually and operationally, allowing for further expansion and adjustment to individual requirements of servers and data centers.

The usefulness of that metric is being tested daily on a large population of non-homogenous servers in our data centers, in an environment where hundreds of different applications are run on a variety of server types.

# 7. Acknowledgments

We express our deep thanks to our internal reviewers and editors, Dr. Bill Jia and Mike Eck for very thorough verification and substantial comments on both the content and the form of this paper.

# 8. References

[1] C. H. Tsai, K. G. Shin, J. Reumann (2007). "Online web cluster capacity estimation and its application to energy conservation". IEEE Transactions on Parallel and Distributed Systems. 18(7). pp. 932-945.

[2] N. Xiong, X. Defago, Y. He, Y. Yang (2005). "A resource-based server performance control for grid computing systems". Lecture Notes in Computer Science. 3779, pp. 56-64.

[3] http://www.craigsmullins.com/dbta_074.htm,

[4] http://technet.microsoft.com/en-us/library/aa998066.aspx

[5] S. Johnson, H. Fangohr, and S. J. Cox (2008). "Managing large volumes of distributed scientific data." Lecture Notes in Computer Science. 5103(3), pp. 339-348.