CMG JOURNAL

2018 | ISSUE 1

THIS ISSUE:

CMG
Computer Measurement Group

Editor: Stephen Guendert, Ph.D.
Issue Editor: Sarah Guendert

The Computer Measurement Group, Inc. (CMG) is the most influential organization worldwide for the exchange of information among Computer Performance Evaluation (CPE) professionals. To keep CPE professionals on top of their field, CMG is dedicated to providing its members the education, networking, and leadership opportunities vital to success in today's competitive industry.

As a benefit to its members, CMG is pleased to publish this CMG Journal and it complimentary publications, the Proceedings, Amplify Blog, and MeasureIT Newsletter.

For more information about CMG and the benefits of membership, please visit www.cmg.org.

INFORMATION FOR AUTHORS

The Journal of Computer Resource Management (CMG Journal) is distributed only to members of the Computer Measurement Group, Inc. (CMG). CMG membership consists primarily of practicing professionals in the area of computer performance, analysis, and capacity planning. The objective of the CMG Journal is to bring the membership information of current importance involving:

- performance characteristics of computer systems of all sizes and architectures
- techniques to measure, analyze, compare, predict, and report performance
- techniques for managing computer performance, including capacity management, computer cost accounting, and computer performance reporting
- integration of disparate components into hardware and software systems to meet organization needs.

The intent of CMG is to publish relevant papers as rapidly as possible, preferably within six months of completion. Authors can assist CMG in achieving this objective by submitting papers in a uniform format.

To submit your paper for consideration in the CMG Journal, it must follow a format consistent with CMG guidelines. For complete instructions on paper submittal you should download two documents from https://www.cmg.org/publications/cmg-journal/ located on the CMG Website. These documents are 'Paper Guidelines and Instructions' and 'CMG License to Publish.' The first document will provide guidelines on format and text; the second document is a required form that must be signed/returned to CMG that grants your permission for CMG to publish your paper.

Here are some highlights on what is expected from authors:

- Camera-Ready Papers, in Microsoft Word, should be sent via email to cmgjournal@cmg.org
- Electronic copies of the paper are preferred; send two hardcopies of the paper in the required format (see 'Instructions' document for details) if an electronic copy is not possible. Hardcopies must be camera ready Papers should be submitted three months prior to the Journal publication date
- Please indicate if you wish to review the paper after editing, prior to publication
- Refer to all tables and figures by number

Authors must read and follow the 'Instructions for Preparing Papers' document. The above highlights do not include all the guidelines required for submitting your paper. Papers not complying with the instructions may be rejected for inclusion in the CMG Journal.

Published by the Computer Measurement Group, a not for profit Illinois membership corporation.

CMG Journal

A Publication of the Computer Measurement Group

2018 Issue 1

Table of Contents

CMG Journal 2018 Issue #1: Letter from the Editor

Welcome to the first issue of CMG Journal for 2018. It's hard to believe that CMG Impact 2017 is already two months behind us. Many of us are in the middle of winter's deep freeze. What better way to get the brain warmed up than by reading this, the latest issue of CMG Journal. This month we have four papers, and a book review for your enjoyment.

Our first paper was written by the 2009 CMG AA Michelson Award Winner, Bruce McNutt. *System Utilization: Keeping the Glass Half Full,* is a second edition of the same title paper that Bruce presented at the CMG 2016 Conference. The paper describes how the black box model, helps to plan and operate systems that grow gracefully. This comes about through a better understanding of system utilization. Earlier papers showed that the black box model can provide automated utilization measurements. This paper applies the same model more broadly to the planning and operation of a system. The goal is to grow the productivity of the system up to the level defined in the capacity planning process, with minor delays due to queueing and/or contention. This updated edition of the original paper offers simplified calculation methods for working with system utilization compared with those provided in the original version.

Our second paper, *Machine Learning for Predictive Performance Monitoring: Predicting Near-Future Total CPU Utilization using the SAS/STAT® GLMSELECT Procedure,* written by Tim Browning, presents techniques based on statistical machine learning for predicting near future CPU utilization based on the current system state as defined by selected features, feature extractions, and time classifications. After assessment of several linear models, the proposed solution uses the Least Angle Regression (LAR) technique within SAS/STAT® software and provides significant accuracy for short term predictions. In addition, the application has the ability to learn and adapt to changing conditions by reiterating the model build process.

This issue's third paper, written by Todd Havenost, was a winner of the Best Paper award at the recent CMG Impact 2017 Conference. *Achieving CPU (& MLC) Savings through Optimizing Processor Cache,* introduces key processor cache concepts and metrics, laying the foundation for understanding the vital role processor cache plays in CPU consumption. Those metrics are then leveraged to identify specific ways to improve processor cache efficiency by optimizing LPAR topology and maximizing work executing on Vertical High CPs. Throughout the paper, Todd uses several examples from real-life situations to illustrate both the opportunities and the potential impacts of tuning actions.

The final paper of the issue is *FICON CUP Diagnostics and the IBM Health Checker for z/OS,* written by Steve Guendert. This is a paper based on a chapter from the new book Steve just completed writing. As IBM Z environments have grown, and configurations have become more complex, FICON fabric issues can result in unacceptable Input/Output (I/O) service times. Resource Measurement Facility (RMF) device activity reports might show average service times that are higher than normal, while matching I/O queuing reports show abnormally high "initial command response" times on a subset of the paths to a device. Steve's paper reviews the basic functionality of FICON CUP and then introduces the new FICON Management Server (FMS) and CUP Diagnostics capabilities now available to be used by the IBM Health Checker for z/OS to gain better insight into the health and robustness of FICON Storage Area Network (SAN) fabrics.

Last but not least, is a book review of Greg Schulz's latest work, *Software-Defined Data Infrastructure Essentials.* I highly recommend this book to all our members, especially those of you interested in storage and I/O.

We are actively soliciting papers for future issues of CMG Journal. We are planning on publishing four issues per year, with a goal of four high quality papers per issue. Our next issue is planned for May 2018 and we are currently looking for and reviewing submissions. Please consider writing a paper for the CMG Journal. You can submit your papers, as well as feedback to us at cmgjournal@cmg.org.

Thanks again for reading. We hope you enjoy this issue.

Stephen R. Guendert, Ph.D.

System Utilization: Keeping the Glass Half Full

Bruce McNutt
IntelliMagic, Inc.

Abstract

This paper describes how the black box model, introduced in earlier papers, helps to plan and operate systems that grow gracefully. This comes about through a better understanding of system utilization. Earlier papers showed that the black box model can provide automated utilization measurements. This paper applies the same model more broadly to the planning and operation of a system. Our goal is to grow the productivity of the system up to the level defined in the capacity planning process, with minor delays due to queueing and/or contention.

This paper is a second edition of the paper with the same title presented at the 2016 CMG Conference. The purpose of the second edition is to offer simplified calculation methods for working with system utilization, compared with those provided in the original version.

1 Introduction

Previous papers have shown that by concealing the details of the server configuration, it is possible to develop a simple "generic" model of system queueing, that fits well with a variety of more detailed queueing models [1]. Also, the resulting *black box* model provides a powerful technique for measuring the utilization of a system based upon its external behavior, even when traditional utilization measurements based upon cycle counts are not applicable [2].

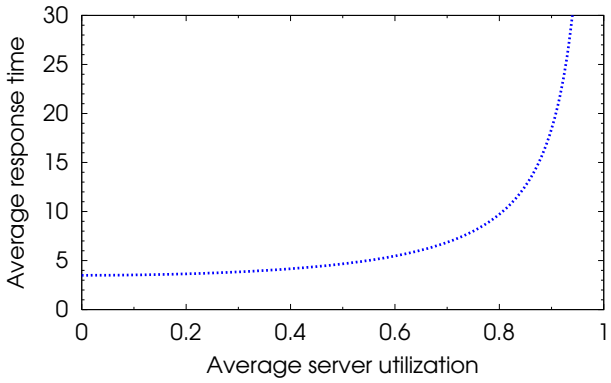
Just as the black box model helps with the measurement of utilization, it also sheds light on the implications of utilization to system performance and capacity planning. The purpose of this paper is to explore more fully those capabilities of the black box model. This paper is a second edition of the paper with the same title presented at the 2016 CMG Conference. The purpose of the second edition is to offer simplified calculation methods for working with system utilization, compared with those provided in the original version.

Ideally, we would like the increase in system use that comes from growth to occur gracefully. When this happens, utilization is a half-full, rather than a half-empty glass. The productivity of the system grows until its use reaches the level initially defined during the capacity planning process. Meanwhile, the system continues to perform well, with relatively minor delays due to queueing and/or contention.

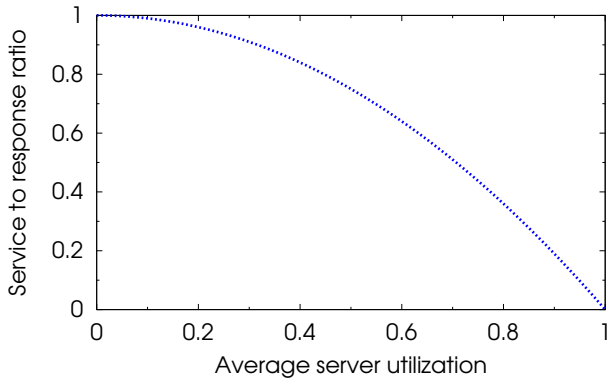
Unfortunately, the glass can sometimes become half empty. Systems can grow past their realistic limits. At that point, the use of a system for its intended purpose may become a struggle, due to severe delays.

This paper describes how the black box model provides a quantitative mechanism to plan and operate systems so that the glass remains half full. Section 2 puts in place a simple, geometrically oriented definition of the desired operating region, together with multiple criteria that can be used to verify whether a given capacity plan, or a given operational environment, falls within that region. Sections 3 and 4 then follow up on the actual use of the proposed criteria in, respectively, capacity planning as well as day to day system management.

Although the black box model provides the underpinning of the proposed scheme, its value comes from the criteria provided in this paper for assessing utilization. We find that it is surprisingly easy to to decide whether the use of a system exceeds the level that can be accomplished gracefully. In the capacity planning process, this assessment is based upon the estimated system resources and processing requirements. For a running sys-



a. Response time/throughput curve.



b. System map.

Figure 1 *The same M/M/2 queueing model, presented in two ways.*

tem, the same assessment is based upon live measurements.

2 Region of Graceful System Growth

The mathematics of the black box model are tied closely to the *system map*, an idea first suggested by Allen [3]. Figure 1 presents an example of how the system map works. The example shows the behavior of the M/M/2 model, although the exact model chosen does not matter for the purposes of the present discussion.

Figure 1a presents the average response time R as a function of the system utilization ρ. This is a typical example of a so-called response time/throughput curve. For very low system utilizations, little waiting occurs, and the response time is dominated by the time spent actually receiving service; that is,

$$R = Q + s$$

where the average queue time Q is small compared with the average service time s.

The drawback of Figure 1a comes at very high system utilizations. In this region of the figure, the time spent waiting for service becomes unbounded, so no matter what range is chosen for the vertical axis, it is impossible to show the entire curve.

The corresponding *system map*, as shown in Figure 1b, solves this problem by presenting the *ratio of service time to response time*. For the sake of having a convenient term, let us call this quantity the system map ratio:

$$M = \frac{s}{R} = \frac{s}{Q+s}$$

In the system map, this ratio gradually falls from unity (no waiting) to zero (infinite waiting); in this way, the entire curve can be shown, including the behavior as the system utilization approaches 100 percent.

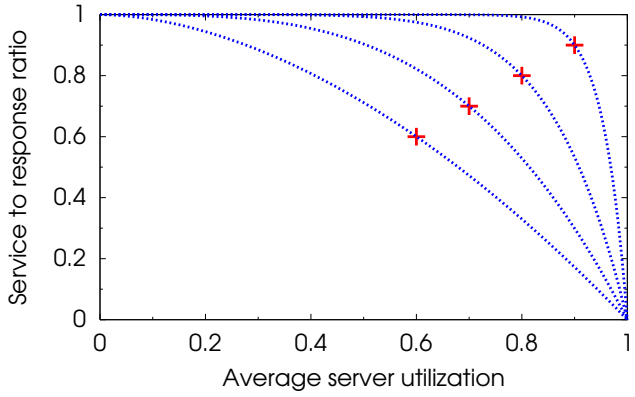


Figure 3 *Family of system maps produced by the black box model.*

As a thought exercise, imagine a blank system map. Suppose we know that a particular system's range of

performance includes some point on that map – for example, any of the points marked "+" in Figure 3. How much can we conclude from this minimal amount of information?

Remarkably, it seems to be possible to conclude a great deal. A variety of different queueing models, if they are forced to produce a curve on the system map that passes through an identified point, tend to produce very similar results. Thus, given a point on the system map, it is possible to estimate fairly well the remainder of the map, *without needing to know all the details of the queueing mechanism*.

The easiest way to produce the needed estimate is to use a family of curves that is not a valid solution to any well-known queueing model. This family, given by the general formula

$$M = 1 - \rho^c \tag{1}$$

is, however, very simple mathematically. Figure 3 presents the inferred system map behavior, based upon (1).

The black box model is an approximate mathematical model that can be invoked to explain the success of (1). For this paper, we do not actually require the full black box model; instead, we begin with (1), and reason based upon that starting point.

Assume, then, that the system map of a given system is specified by (1) for some number of servers $c \geq 1$. The number of servers may require estimation, and is not necessarily assumed to be an integer. We now assert that the range of system operating conditions over which growth can occur in a graceful manner corresponds with the condition $M \geq \rho$ as shown on the system map. In geometric terms, the desired operating region is the portion of the system map at or above a diagonal line extending from the origin.

To see that this is the case, consider the specific utilization defined by $\rho_k = (c+1)^{-1/c}$. This helpful point of reference is always close to, but no greater than, the utilization that occurs on the diagonal.

By (1), the corresponding system map ratio M_k is given by $c/(c+1) >= \rho_k$. Thus, the fraction of time spent in the queue, waiting for service, is given by $1 - M_k = 1/(c+1)$. This quantity is equal to $\frac{1}{2}$ when $c = 1$, and becomes less and less significant as c increases.

Since $M_k >= \rho_k$ we may draw a vertical line from the point (ρ_k, M_k) downward, meeting the diagonal; then another line toward the right, meeting the curve again at a point such that $M = \rho_k$. Clearly, this point is at or below the diagonal. In addition, the quantity $1 - \rho_k$ has the same desirable properties as those just described at the end of the previous paragraph. This quantity is *also* equal to $\frac{1}{2}$ when $c = 1$, and becomes less and less significant as c increases.

The value of M that occurs at the diagonal can be bounded between the two calculations of M just given; therefore, the fraction of time spent in the queue, for an operating point along the diagonal of the system map, has the same desirable properties as it does for the two bounding cases. It is equal to $\frac{1}{2}$ when $c = 1$, and becomes less and less significant as c increases.

On the other hand, by the time the curve of performance as shown on the system map crosses the diagonal, it has already turned downward. Any further gains in utilization must come at the expense of significant additions to queue time. This situation calls to mind the old saying "quit while you are ahead", and explains the recommended geometry for defining when the utilization glass can be considered half full.

The core idea of this paper is that a surprising variety of mathematical criteria can be used to identify the desired operating region. They are equivalent to each other in the sense that they produce the same pass/fail grade when applied to a given set of operational conditions. However, each criterion uses different information, reducing the amount that must be known about the environment.

Let x represent the average rate of system requests per second, and N the average population of outstanding system requests. Note also that by Little's law, these quantities are related by $N = xR$. Then within the framework of the black box model, all five of the following criteria are equivalent:

$$M \geq \rho \tag{2}$$

$$\rho \leq \frac{2M_k + \rho_k}{3 + M_k - \rho_k} \tag{3}$$

$$N \leq c \tag{4}$$

$$1 - \left(\frac{xs}{c}\right)^c \geq \frac{xs}{c} \tag{5}$$

$$\left[1 - \frac{s}{R}\right]^{\frac{1}{N}} \leq \frac{s}{R} \tag{6}$$

Each equation gives a different statement of the region that lies on or above the diagonal on the system map.

For completeness, it should be noted that the bounding threshold (3) for system utilization is stated to three digit precision. The exact boundary is given by the limit

$$\rho \leq \cdots (1-(1-(1-\frac{c}{c+1})^{\frac{1}{c}})^{\frac{1}{c}})^{\frac{1}{c}} \cdots \qquad (7)$$

The much simpler approximate form (3) is close enough for all practical purposes.

As an example of how to reason about the black box model based upon a variety of criteria, consider again the point of reference that occurs when the utilization is given by $\rho_k = (c+1)^{-1/c}$. As we observed in a previous paragraph, this point reflects conditions where the criterion (2) holds; but we might ask, how close is that criterion to failing?

The simplest way to answer becomes apparent by examining instead criterion (4). The value of N at the point of reference is given by $N_k = (c+1)\rho_k$. When c is an integer, that same value can be stated approximately as $N_k \approx c - H_{c+2} + H_3$, where H_i represents the ith harmonic number. Keeping in mind that $c \geq 1$, we thus see that the value of N_k is equal to that of c when $c = 1$, and falls below that of c for $c > 1$.

Turning now to the offset δ between N_k and c, we have the relationship $\delta \approx H_{c+2} - H_3 << c$. For this reason, the value of N_k always comprises by far the largest portion of c. It is fair to say that the point of reference corresponds to a case where the criterion (4) just barely passes. By extension, the same statement applies to the other four criteria as well.

The remaining sections of the paper exploit (5) and (6) by applying them respectively to the capacity planning and the system monitoring phases of the system management life cycle.

3 Capacity Planning

The capacity plan for a complex system proceeds by examining that system's most important resources. The black box model is intended to help with the analysis of resources that are used temporarily to process a particular request, such as processor cores or system ports. The requirement for a resource of this type is driven by the rate x of requests against it, as well as the average service time s that will be required to complete any single request, when running by itself.

Given an estimate of both of the quantities just identified, it is then necessary to assess how many individual units of the given resource should be configured. Ordinarily, system performance will suffer unless *more* units of the resource are provided than the average number required by the rate of requests. The criterion (5) provides a simple and effective way to address the question of *how many* additional units must be added to the capacity plan, beyond those called for directly by the estimated demand.

For example, consider the number of 8 Gbps fibre channel ports being provided with a Linux server. Based upon past levels of load, we estimate that the server's I/O rate when running its maximum intended level of application work is 50,000 I/Os per second, with an average transfer size of 16 KB. Assuming the nominal 8 Gbps fibre channel protocol, this implies that the number of ports in active use will be $(16/1024) \times 50,000/800 = 0.976$ on average. It is easy to guess that we ought to configure at least 2 ports, based upon this average level of port demand. But are two ports sufficient?

A simple way to address this question is to apply the criterion (5). By that criterion, 2 ports *are* sufficient, since $1-(0.976/2)^2 = 0.762 > 0.976/2$ (the criterion passes).

On the other hand, in the analysis just presented, we assumed that the port can run at the full 8 Gbps permitted by the fiber channel protocol. Suppose, more conservatively, we wish to assume that the actual effective speed of the port is 75 percent of this theoretical maximum. This then implies that we need 1.30 ports on average. Applying the criterion (5), we now obtain $1-(1.30/2)^2 = 0.577 < 1.30/2$ (the criterion fails).

We therefore arrive at the conclusion that two ports *may* be sufficient, but only if the performance of the port technology runs at close to the theoretical maximum. A sensible next step may therefore be to investigate further any performance data that is available for the affected type of port hardware. Alternately, we could choose to configure 4 ports. By (5) this will be sufficient even if the port is only 75 percent efficient, since $1-(1.30/4)^4 = 0.989 > 1.30/4$ (the criterion passes).

4 System Monitoring

For many systems today, it is possible to directly measure the throughput x and response time R during any defined measurement interval throughout the day. In addition, the quantity $N = xR$ can be obtained using Little's Law. To apply criterion (6), however, we also require data for the average service time s.

One method to measure s is to instrument the system with the *General Purpose Utilization Monitor*, which performs measurements of s in each measurement period in addition to producing estimates of system utilization [2]. In this paper, however, we do not assume that instrumentation of that type is available. A reasonable alternative is to estimate s by identifying a measurement period in which the mix of applications is similar to that running during the daily peak, but the system load is less. In a well chosen interval as just described, queueing may be light enough so that the measurement of R can also be taken as the approximate value of s.

Assuming that a reasonable value for s can be identified, it is then possible to apply criterion (6) to assess based on measured data whether the utilization glass is half full.

It is important to note that the application of (6) does *not* require a value for c. Instead, this criterion provides an independent check on actual delivered concurrency of the system. If for any reason the value of N exceeds the effective system concurrency during some periods, then we should expect that the criterion (6) will fail in those periods.

For example, consider the same Linux system as before, configured with two ports. Using data from a shoulder period as well as a peak period, where the average transfer size was 16K in both cases, we conclude that the minimum time required to complete such a transfer is 26 microseconds, but under peak conditions the average response time is 35 microseconds. Also, the peak throughput is 40,000 I/Os per second, hence $N = 40,000 \times 0.000035 = 1.4$. Based on that information, the criterion (6) gives $[1 - 26/35]^{1/1.4} = 0.379 < 26/35$ (the criterion passes).

Although the system has been configured with two ports, this does not *necessarily* mean that it can always support two concurrent transfers without the use of a queue. For example, if both ports are in the same adapter, the processor provided in the adapter may not be able to accept interrupts from both ports at the same time. Also, under some conditions, the adapter may break a single host transfer into more than one physical transfer. The reverse case can also sometimes occur, in which multiple transfers are consolidated. The black box model cannot help with capturing such effects in detail, but it provides a way to gauge their impact. Effects of this kind may influence the effective concurrency of the system.

In the example just given, we applied (6) to a single measurement period. If, however, we observe the system through a large number of measurement periods, and assuming that we are able to capture wide swings in the load level, we can then assess the maximum value of N that is still capable of passing the criterion (6). That maximum value provides an independent check on the actual concurrency delivered by the configured pair of ports.

Suppose, then, that the two ports do deliver a concurrency of $c = 2$. Given that fact, we can then use the black box model to estimate the average port utilization during a given, specific measurement interval. Graphically, the condition $c = 2$ allows us to draw a curve on the system map that belongs to the family given by (1). We then superimpose, on the same map, a straight line that corresponds to the condition $M = s/R = sx/N = \rho c/N$. To estimate the utilization in a given interval, we proceed by finding the utilization level at which the curve intersects with the straight line. To three digit precision, the solution for system utilization, as just described, is given by

$$\rho = \frac{N}{c} \, \frac{1 + (c-1)\rho_k^c \nu}{1 + N\rho_k^{c-1} \nu^{(c-1)/c}} \tag{8}$$

where

$$\nu = \min(\alpha, \alpha^c) \tag{9}$$

and where

$$\alpha = 1 + c \, \frac{\max(N, N_k)}{c^2 + N^2} \, \max(N - N_k, \frac{N - N_k}{N_k}) \tag{10}$$

Continuing the example of the two port system, we have already concluded that $c = 2$; this, in turn, implies that $\rho_k = (c+1)^{-1/c} = 3^{-1/2} = 0.5773$ and $N_k = (c+1)\rho_k = 1.732$. Also, in the earlier example we measured $N = 1.4$. Since $N < N_k$, the factor N_k appears in both the numerator and denominator of (10). Dropping

that factor, α takes the form $1+c(N-N_k)/(c^2+N^2) = 1+2\times(1.4-1.732)/(2^2+1.4^2) = 0.8103$ (the quantity α is less than unity whenever N is less than N_k). Since $\alpha < 1$, we compute $\nu = \alpha^2 = 0.6566$. Finally, the overall black box utilization is given by

$$\frac{1.4}{2} \times \frac{1+1\times 0.5773^2 \times 0.6566}{1+1.4\times 0.5773\times 0.6566^{1/2}} = 0.515$$

to three digit precision.

Some room for *graceful* growth remains at the utilization level $\rho \approx 0.515$ just obtained. However, the opportunity for such growth extends only until we reach the condition $N = c = 2$. We can obtain the corresponding threshold for the utilization level either by plugging the assumption $N = 2$ into (8), or by applying (3). Using either approach, we find that growth should not continue past the utilization level given by $\rho_{at\_c} = 0.618$.

5 Conclusions

This paper follows up on earlier papers about the black box model, and uses that model to provide important guidelines for system management. In particular, we have examined the question of whether the utilization of a given, specific system is excessive. Within the framework of the black box model, this condition has a simple geometric interpretation.

The most important conclusion of the paper is that the black box model provides several distinct but equivalent mathematical tests for excessive system utilization. These tests can simplify both the capacity planning as well as the performance measurement phases of the system management life cycle.

The General Purpose Utilization Monitor, described in an earlier paper, can assist in the management of system utilization. It can take needed measurements, report system utilization and dynamically maintain an estimate of the concurrency of the system. This paper does not assume that a monitor with these functions is in use, but clarifies the underlying ideas behind the use of such a monitor.

References

[1] B. McNutt, “Waiting for a Black Box”, CMG Proceedings, Nov. 2013.

[2] B. McNutt, “Meter Anything from Component to Cloud: A General-Purpose Utilization Monitor”, CMG Proceedings, Nov. 2014

[3] A.O. Allen, *Probability, Statistics, and Queueing Theory*, Academic Press, 1978. See particularly pp. 221-223.

Machine Learning for Predictive Performance Monitoring

Predicting Near-Future Total CPU Utilization using the SAS/STAT® GLMSELECT Procedure

Tim Browning, IT Consultant
Kimberly-Clark Corporation

Predictive monitoring is becoming an important component of data center management to support operational stability. A key element is to understand if a system will be exhibiting CPU constraint in the near future, or operating under more optimal conditions. If the system is expected to have a predictable range of CPU activity, mitigating operational actions can be implemented such as deferring scheduled CPU intensive work during times of high CPU usage or submitting work during predicted low utilization time periods.

This paper presents techniques based on statistical *machine learning* for predicting near future CPU utilization based on the current system state as defined by selected features, feature extractions, and time classifications. After assessment of several linear models, the proposed solution uses the *Least Angle Regression* (LAR) technique within SAS/STAT® software and provides significant accuracy for short term predictions. In addition, the application has the ability to learn and adapt to changing conditions by reiterating the model build process

In a series of experiments, many predictive algorithms were applied to production data in a SAP Z/OS-based operating environment to: (1) select the appropriate features and extracted features so as to predict CPU utilization in future 5-minute intervals, and (2) utilize cross validation to reduce over fitting and improve accuracy when using new data. Using error estimations provided by modeling and validation process, we select the linear regression model and features that provide the highest accuracy for out-of-sample data.
Results showed highly accurate short-term predictions as well as acceptable levels of accuracy for some longer term prediction.

1. Introduction

There has been considerable interest in the application of machine learning statistical techniques for predicting the operational performance of critical components in large complex data centers.
As computer systems have evolved into more complex networked landscapes, automatic prediction of system near-future activity may become an important system administration goal to assure operational stability. Predictive monitoring would enable intelligent scheduling for planned work as opposed to fixed time schedules.

It may not be possible to change the future, much as we cannot change the weather, but forewarned is forearmed. If we know it's going to rain, we may want to carry an umbrella. If we know the computer system will be very busy in the next twenty minutes, we may defer scheduling additional work at a more favorable time.

The approach discussed in this paper utilizes SAS/STAT® software to develop and deploy a machine learning application for the purpose of predicting CPU utilization in the near future. The Central Processing Complex (CPC), using z/OS MVS operating system, functions as the heart of an SAP application landscape by serving as the database server (DB2). This infrastructure implementation of the application environment supports extremely high availability and scalability of SAP workloads characterized by high relative I/O content - the proportion of I/O work per CPU cycle - as well as efficient parallelism in I/O operations. The architecture is typical of large SAP-based environments used by major corporations in the manufacturing sector.

2. Avoiding High CPU Utilization

High CPU utilization often results in queuing delays and becomes both a performance and a capacity issue. Such conditions are a threat to operational

stability and often lead to poor responsiveness and/or availability, ultimately resulting in extensive loss to the business. Statistical monitors are configured to raise an alarm by issuing a 'ticket' when these conditions are in effect. However, a more proactive process needs to be deployed that will predict the near term CPU utilization with an acceptable level of accuracy based on current system state as defined by performance features and time classification contexts.

Because of virtualization and partitioning of severs, many system images can run on a shared platform with shared CPUs. High utilization of the shared CPUs can globally impact many servers and their associated applications.

3. Analytics Background

Linear regression was developed within the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by *machine learning*. It is both a statistical algorithm and a machine learning algorithm.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

3.1 System Features

The system under study is an IBM mainframe - Z13 2964 - functioning as a large database server (DB2) in an SAP distributed application architecture. As such, the central processing complex (CPC) has high *relative I/O content* – amount of I/O per CPU second – and thus we expect that current I/O activity may have predictive value for CPU utilization. Features of the system include the total CPU utilization of the complex, CPU usage by workloads, total system I/O rate, and I/O response times.

3.2 Feature Extractions

Feature selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. For example, I/O rate is a "feature" and the LOG or reciprocal of I/O rate is a "feature extraction". Feature extraction can be computed from a series of features such as the moving average or information entropy, or by interaction effects as a result of multiplying one feature by one or more other features.

Several feature extractions were included in the modeling process that incorporate predictive analytics based on past patterns of activity. These include the time series *moving average* of CPU utilization - a lagged trend indicator - and the *expected value, or mean, from a multivariate adaptive statistical filter (MASF) reference set* using time classifications (time-of-day, day-of-week). MASF is effective in anomaly detection by exploiting detected cyclic periodicities in CPU demand relative to a recurring temporal context establishing upper and lower limits of the expected CPU utilization. In effect, the MASF *reference set* provides a probabilistic distribution of values based on parametric or non-parametric measures of central tendency and variation relative to specific time classifications.

In addition, the *moving average* over the last few time intervals was calculated using the outlier-resistant geometric moving average. In this paper, the *geometric moving average* is the geometric mean of the previous n-minute intervals, i.e. not the average of geometric means. All numeric features in this project are greater than zero, thus a useful formula for the geometric mean of a time series feature set x can be defined as:

$$\bar{x} = \sqrt[n]{\prod_{i}^{n} x_i} = \sqrt[n]{x_1 . x_2 . x_3 \dots x_n} = exp\left[\frac{1}{n}\sum_{i=1}^{n} log(x_i)\right]$$

Additional polynomial transforms of the original data included exponentiation, logarithmic, and reciprocal function transforms.

In the case of relative I/O content, i.e. the proportion of I/O per CPU second, both CPU utilization (between zero and 100) and I/O (in thousands of bytes per second) were rescaled to a common scale using z-scores. Each observation is recalculated as the mean minus the standard deviation, divided by the standard deviation. This was done to better detect measures of variability. To avoid negative relative I/O content when comparing on the z-score transformed data, the absolute values were used.

Probabilistic aspects of CPU utilization uncertainty were included as extractred features such as the

information entropy of the current I/O rate and CPU usage. An assumption is made is that there are potential *interaction effects* between entropy measures and current activity. Entropy increases the uncertainty of the expected the CPU utilization.

Information entropy measures the uncertainty of a collection of data observations. Like MASF, information entropy exploits frequencies of occurrence of the response variable based on specific classifications (e.g. CPU utilization is counted in ranges or bins, with classifications of time-of-day in 5 minute increments, and day of week).

Calculation of information entropy:

Definition: *For a dataset X where each data item belongs to a specific classification (e.g., hour of day, day of week, range of values), the entropy of X relative to that class is defined as*

$$H(X) = \sum P(x) log \frac{1}{P(x)}$$

where P(x) is the probability of x in X.

In addition to constructed multivariate polynomial effects, *basis spline effects*, a special function defined piecewise by polynomials, were used to approximate complex shapes often seen in performance and capacity utilization curves. Since there is no theoretical basis for choosing a fitting function, the regression curve may be fitted with a spline function composed of a sum of B-splines. This study used spline functions up to degree=3 for CPU, I/O rate, and workload CPU demand as well as the information entropy of these features.

3.3 Requirements of the Linear Model

Linear Assumption. Linear regression assumes that the relationship between input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when there are a lot of attributes, *there may be a need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).*

Data Transformations for Linear Regression

Method	Transformation(s)	Regression equation	Predicted value ($\hat{y}$)
Standard linear regression	None	$y = b_0 + b_1x$	$\hat{y} = b_0 + b_1x$
Exponential model	Dependent variable = log(y)	$log(y) = b_0 + b_1x$	$\hat{y} = 10^{b_0 + b_1x}$
Quadratic model	Dependent variable = sqrt(y)	$sqrt(y) = b_0 + b_1x$	$\hat{y} = (b_0 + b_1x)^2$
Reciprocal model	Dependent variable = 1/y	$1/y = b_0 + b_1x$	$\hat{y} = 1 / (b_0 + b_1x)$
Logarithmic model	Independent variable = log(x)	$y= b_0 + b_1log(x)$	$\hat{y} = b_0 + b_1log(x)$
Power model	Dependent variable = log(y) Independent variable = log(x)	$log(y)= b_0 + b_1log(x)$	$\hat{y} = 10^{b_0 + b_1 log(x)}$

Figure 1 Data Transforms for Linear Regression

Remove Noise. Linear regression assumes that input and output variables are not noisy. This is most important for the output variable and *outliers were removed in the output variable.* Since the response variable is normally distributed, *outliers were defined as the mean +/- 3*standard deviation.* In the case of moving averages, where outliers may be present in the most recent monitor data, the geometric mean is used, as it is more resistant to distortion than the arithmetic mean.

Remove Collinearity. Linear regression will over-fit your data when there are highly correlated input variables. *Pairwise correlations of input features were performed to remove the most highly correlated metrics.*

Here is an example of part of the feature correlation matrix. Excel was used with conditional formatting to quickly find highly correlated features.

Feature Correlation Matrix

	cpu_future	cpu_now	entropy_cpu	masf_cpu	cpu_recp	cpu_sqrt
cpu_future	1	0.79	-0.15	0.87	-0.79	0.8
cpu_now	0.79	1	-0.19	0.69	-0.93	1
entropy_cpu	-0.15	-0.19	1	-0.18	0.15	-0.18
masf_cpu	0.87	0.69	-0.18	1	-0.67	0.7
cpu_recp	-0.79	-0.93	0.15	-0.67	1	-0.95
cpu_sqrt	0.8	1	-0.18	0.7	-0.95	1
entropy_rioc	-0.02	-0.01	0	-0.02	0.02	-0.01
entropy_sio	-0.17	-0.2	0.23	-0.2	0.19	-0.2
entropy_workload	-0.04	-0.07	0.45	-0.05	0.04	-0.06
IOrate_geomean	0.26	0.34	-0.04	0.27	-0.33	0.34
IOrate_weighted_mean	0.02	0.03	-0.02	0.05	-0.03	0.03
IOrate_sqroot	0.15	0.17	0.03	0.08	-0.16	0.17
ma_cpu	0.87	0.89	-0.13	0.75	-0.91	0.91
ma_response_time	0.03	0.02	-0.01	0.03	-0.02	0.02
ma_rioc	0.04	0.03	0	0.05	-0.06	0.04
ma_siorate	0.63	0.65	-0.07	0.54	-0.64	0.65
ma_workload	0.79	0.85	-0.14	0.68	-0.81	0.85
Response_time_geomean	0.02	0.05	-0.04	0.09	-0.05	0.05
Response_time_log	0.09	0.09	-0.04	0.17	-0.1	0.1
Response_time_sqrt	0.06	0.06	-0.02	0.11	-0.07	0.06
RIOC	0.02	0.02	0	0.03	-0.03	0.02
siorate	0.59	0.72	-0.12	0.51	-0.68	0.72
workload	0.74	0.93	-0.18	0.64	-0.86	0.93

Figure 2 Feature Correlation Matric

An example of highly correlated features is I/O Intensity – defined as the product of response time

and I/O rate. It is thus highly correlated to both I/O rate and I/O response time.

Gaussian Distributions. Linear regression will make more reliable predictions if input and output variables have a Gaussian (normal) distribution. In this study, both CPU and most predictor variables were Gaussian. For example, see Figures 3 and 4.

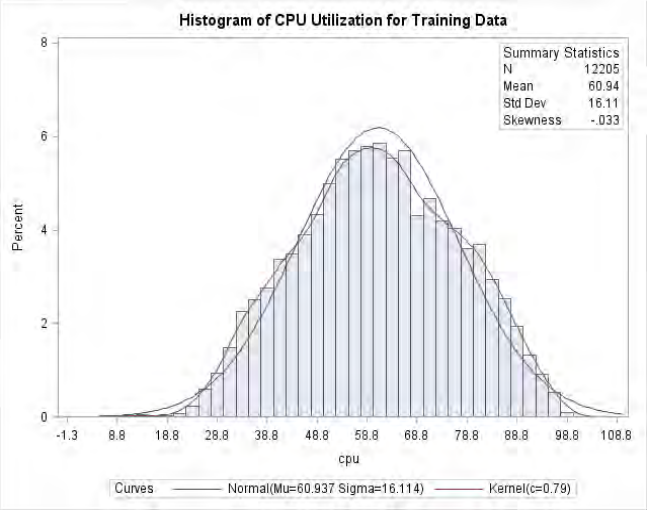


Figure 3 Gaussian Distribution of CPU Utilization

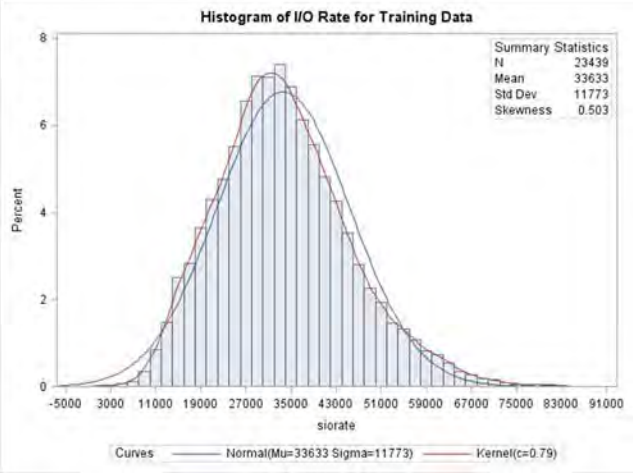


Figure 4 I/O Rate Gaussian Distribution

Rescale Inputs: Linear regression will often make more reliable predictions if the *input variables are rescaled using standardization or normalization.*

Lastly, linear regression assumes that the relationship between input and output is linear. All the above metrics and probabilistic analytics, were subject to logarithmic, reciprocal, and polynomial transforms to convert potential non-linear relationships into a linear model. (See Figure 1).

Memory usage metrics are not included as this system has zero paging rates for most workloads.

3.4 Effect Selection Process

The *effects* for model construction can be summarized into these categories:

Time Effects (Used in MASF)

- Time of Day
- Day of Week

Polynomial Effects

- Transforms of the feature metrics, CPU, I/O, etc.

Basis Splines

- Multivariate polynomial effects

Regressor Effects

- The lagged data from the previous time period for CPU busy, I/O rate, Workload Busy, and I/O response time.

Interaction Effects with Entropy State

- The product of current CPU utilization and CPU Entropy
- The product of current I/O Rate and I/O Entropy
- The product of Workload CPU Usage and Entropy of Workload CPU Usage
- The product of current Relative I/O Content (RIOC) and Entropy of RIOC

Data available for the project includes 45 days of 5-minute interval performance data collected from the RMF Portal. After data cleaning, there were 11,983 observations available. Including classification splits, there are 889 parameters in the model for selection.

These observations were partitioned into lagged datasets where the CPU utilization was paired with lagged features of the system at different past intervals. Random samples were taken of the total amount of data and 9,587 observations were used for

the TRAINING dataset, and 2,397 observations were used for VALIDATION, and TESTING datasets.

Large corporate data systems are typically characterized by scheduled work that repeats in patterns of activity (temporal context) over time. These patterns can be exploited for identifying context anomalies. An individual data item can be anomalous *within a specific context;* which requires an analytical *notion of context*. For time series data it is the change or pattern variation that is of interest. Typically, we expect a negative coefficient for high entropy predictive features. As entropy increases (uncertainty), then predictive utility decreases in the linear model.

4. Model Development Approach

For standard regression, the flagship SAS/STAT procedure for model building is the GLMSELECT procedure. This procedure selects effects in general linear models of the form

$$Y_i = \beta_0 + \beta_1 x_1 + \cdots \beta_{ip} + \varepsilon_i, i = 1, \ldots n$$

where the response yi is continuous. The predictors $x_{i1}, \ldots x_{ip}$ represent main effects (or features) that consist of continuous or classification variables and their interactions or constructed effects.

If there are too many predictors, the model can *overfit* the training data, leading to poor prediction when the model is applied to future data. To mitigate this problem, the GLMSELECT procedure supports a variety of model selection methods, including the LASSO and LAR methods.

4.1 Model Types and Measures of FIt

The first objective is to determine what features to select for the model – effect selection - and the second objective is to determine the best model that fits the data, but does not overfit the data. The modeling procedure GLMSELECT was used to enable the following requirements: parameterizations for classification effects (time and day of week), interaction effects, partitioning of data into training, validation, and testing roles, and provision of stopping rules based on a variety of model evaluation criteria.

Selection methods used included the following:

Forward selection:
starts with no effects in the model and adds effects.

Backward elimination:
starts with all effects in the model and deletes effects.

Stepwise regression:
is similar to forward selection except that effects already in the model do not necessarily stay there.

Least angle regression (LAR):
is similar to forward selection in that it starts with no effects in the model and adds effects. The parameter estimates at any step are "shrunk" when compared to the corresponding least squares estimates.

The results indicated very little difference between Backward and Forward regression so only Backward is summarized.

In every case, the criteria for model fit included R-Squared (coefficient of determination) and RMSE (Root Mean Square Error). The criteria for selection of LAR is based on utility, i.e. how well does the model fit new data not seen before.
The following is the SAS documentation describing how the LAR algorithm is implemented:

> "*The algorithm starts by centering the covariates and response, and scaling the covariates so that they all have the same corrected sum of squares. Initially all coefficients are zero, as is the predicted response. The predictor that is most correlated with the current residual is determined and a step is taken in the direction of this predictor. The length of this step determines the coefficient of this predictor and is chosen so that some other predictor and the current predicted response have the same correlation with the current residual. At this point, the predicted response moves in the direction that is equiangular between these two predictors. Moving in this direction ensures that these two predictors continue to have a common correlation with the current residual. The predicted response moves in this direction until a third predictor has the same correlation with the current residual as the two predictors already in the model. A new direction is determined that is equiangular between these three predictors and the predicted response*

moves in this direction until a fourth predictor joins the set having the same correlation with the current residual. This process continues until all predictors are in the model."

[reference: http://documentation.sas.com/?cdcId=statcdc&cdcVersion=14.2&docsetId=statug&docsetTarget=statug_glmselect_details09.htm&locale=en].

As you can see in Figure 5, even though the Stepwise and Backward regressions better fit the training data on several measures of fit, the LAR method was vastly superior with recent out-of-sample data. Model fitness is determined by how well the predictions fit out-of-sample data from the last eight hours.

A separate model was developed for each 5-minute lag into the past from 5 minutes to 30 minutes:

5 Minute Prediction Modeling Results

	Traing		Out of Sample1		Out of Sample2		Mean	
Model Type	**RSq**	**RMSE**	**RSq**	**RMSE**	**RSq**	**RMSE**	**RSq**	**RMSE**
STEPWISE	0.89	5.84	0.17	13.20	0.37	8.34	0.27	6.79
BACKWARD	0.89	5.84	0.51	9.47	0.38	8.24	0.45	4.93
LAR	0.62	10.624	**0.98**	**4.91**	**0.98**	**4.03**	0.98	2.95

10 Minute Prediction Modeling Results

	Training Result		Sample1		Sample2		Mean	
Model Type	**RSq**	**RMSE**	**RSq**	**RMSE**	**RSq**	**RMSE**	**Rsq**	**RMSE**
STEPWISE	0.81	7.53	0.15	11.60	0.24	12.80	0.20	5.92
BACKWARD	0.83	7.16	0.14	11.6	0.22	12.90	0.18	5.91
LAR	0.45	12.712	**0.96**	**6.12**	**0.95**	**7.48**	0.96	3.54

15 Minute Prediction Modeling Results

	Training Result		Sample1		Sample2		Mean	
Model Type	**RSq**	**RMSE**	**RSq**	**RMSE**	**Rsq**	**RMSE**	**Rsq**	**RMSE**
STEPWISE	0.81	7.50	0.23	13.10	0.48	9.72	0.36	11.41
BACKWARD	0.83	7.13	0.11	15.40	0.30	10.40	0.21	12.90
LAR	0.48	12.44	**0.94**	**7.70**	**0.85**	**6.30**	0.90	7.00

20 Minute Prediction Modeling Results

	Training Result		Sample1		Sample2		Mean	
Model Type	**RSq**	**RMSE**	**RSq**	**RMSE**	**RSq**	**RMSE**	**Rsq**	**RMSE**
STEPWISE	0.88	6.07	0.02	13.00	0.53	8.22	0.28	6.77
BACKWARD	0.89	5.77	0.03	12.80	0.09	16.40	0.06	6.45
LAR	0.45	12.708	**0.49**	**9.950**	**0.98**	**4.20**	0.74	5.47

25 Minute Prediction Modeling Results

	Training Result		Sample1		Sample2		Mean	
Model Type	**RSq**	**RMSE**	**RSq**	**RMSE**	**RSq**	**RMSE**	**Rsq**	**RMSE**
STEPWISE	0.77	8.37	0.12	14.10	0.03	16.00	0.08	7.07
BACKWARD	0.78	8.15	0.10	14.50	0.03	15.90	0.07	7.27
LAR	0.40	13.33	**0.74**	**10.200**	**0.55**	**11.60**	0.65	5.38

30 Minute Prediction Modeling Results

	Training Result		Sample1		Sample2		Mean	
Model Type	**RSq**	**RMSE**	**RSq**	**RMSE**	**RSq**	**RMSE**	**Rsq**	**RMSE**
STEPWISE	0.75	8.61	0.09	12.30	0.17	13.70	0.13	6.24
BACKWARD	0.77	8.28	0.09	12.30	0.04	16.30	0.07	6.17
LAR	0.41	13.13	**0.81**	**8.72**	**0.66**	**10.60**	0.74	4.69

Figure 5 Model Comparisions

As expected, predictive accuracy drops the further away in time the model has to predict.

One way to visualize the superior results from LAR as compared to the STEPWISE method is to use a *prediction ellipse*. The actual vs. predicted data is plotted with an ellipse encapsulation showing the percent of sample data falling with a specific confidence level interval. Because the center of the ellipse is the sample mean, a prediction ellipse can give a visual indication of skewness and outliers in the data. A prediction ellipse also displays linear correlation: If you standardize both variables, a skinny ellipse indicates highly correlated variables, whereas an ellipse that is nearly circular indicates little correlation. Figure 6 shows the LAR results; Figrue 7 the STEPWISE results.

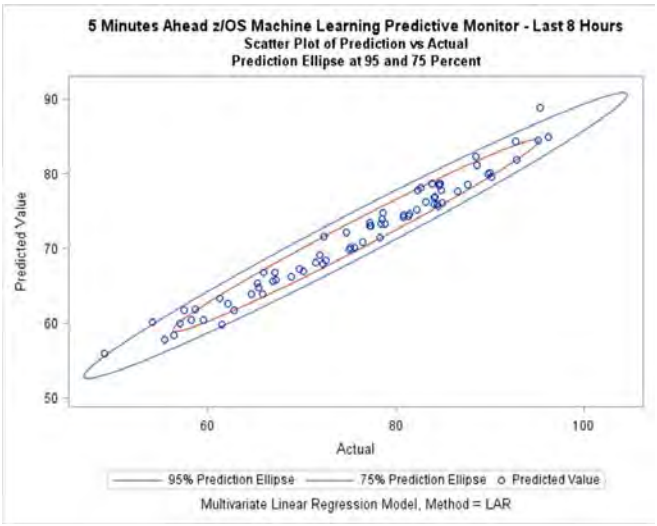


Figure 6 Prediction Ellipse of LAR results for 5-minute predictions

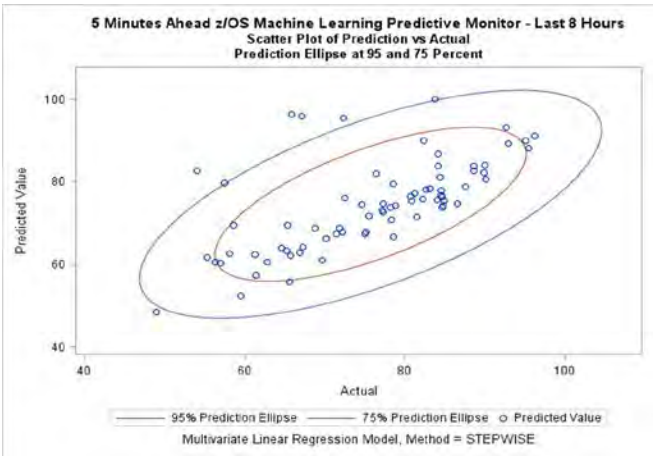


Figure 7 Prediction Ellipse STEPWISE results for 5-minute predictions

Notably, in the coefficient and feature selection process, the MASF reference set mean is the first to enter the model, followed by LOG(cpu), the moving average of the total workload, and the reciprocal (1/CPU).

None of these are actual measure features, but feature extractions.

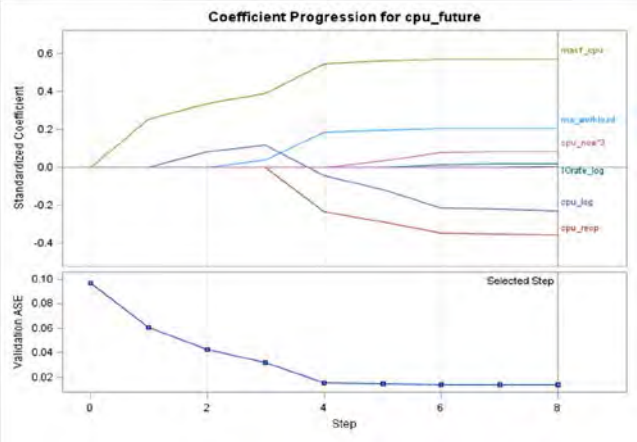


Figure 8 Coefficient Progression plot visualizes the selection process

A plot of the actual vs predicted over the last 8 hours, using the 5-minute look ahead model demonstrates a very accurate predictive history:

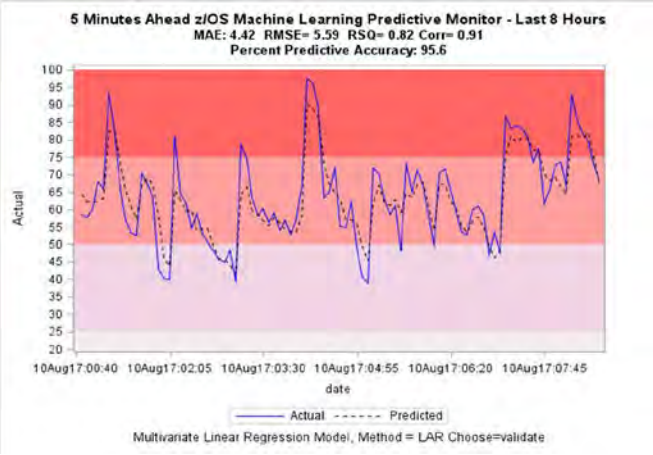


Figure 9 Plot of Actual vs Predicted using a 5-minute look ahead model

A plot of the actual vs predicted, using a 30-minute lag, shows a decline in predictive accuracy (but still quite impressive):

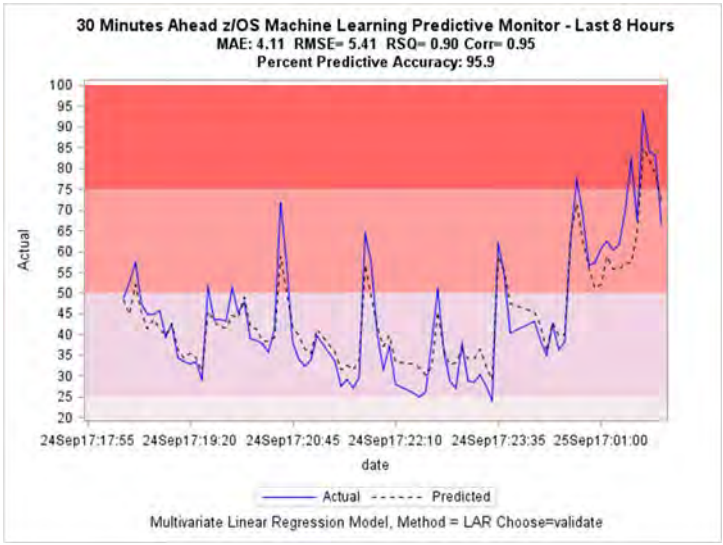


Figure 10 Plot of Actual vs Predicted using a 30-minute look ahead model

The prediction ellipse for the 30-minute look ahead is also quite impressive:

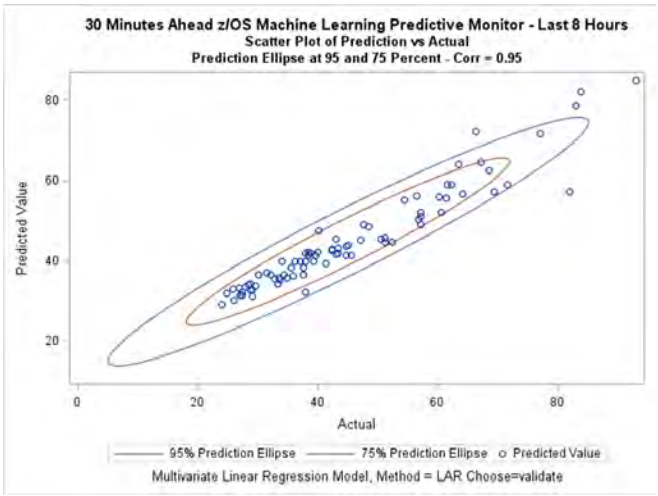


Figure 11 Prediction Ellipse - 30 minute look ahead

5. Discussion

As we would expect, the prediction accuracy declines as we look further into the future. In addition, the types of features and number of features changes.

In Figure 12, the red line shows the drop in accuracy but the number of features selected varies considerably.

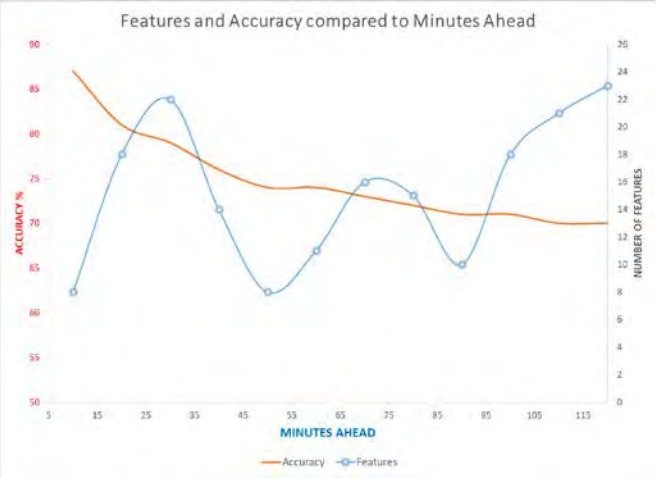


Figure 12 Predictive Accuracy and Number Features vs minutes ahead

Each lagged time period is 5 minutes with models developed incrementally from 5 to 120 minutes into the future. Thus, there are a total of 12 models for each run cycle. The model build and reporting takes about 3 minutes and runs every 5 minutes as RMF/SMF data is captured through the RMF Data Portal.

Machine learning, as used in this study, is the application of some forms of *adaptive statistical modeling*. Machines don't actually "learn" nor do statistical algorithms represent some mechanistic disembodied intelligence. However, human learning and intelligence is greatly assisted by statistical modeling in much the same way that optics technology assists vision.

As a practical matter, dynamic adaptive work scheduling using predictive monitoring should improve efficiencies and stability for CPU-intensive workloads where there is some time flexibility for scheduled work.

6. Conclusions and Limitations

The GLMSELECT procedure, after extensive customization, has produced exciting results with high levels of accuracy. However, each time it runs there are changes made to the models both in terms of model features selected and coefficients. Likewise, predictive accuracy varies for different time periods and days of the week. This is most likely a result of the different times of day it is running (time classification) and changes in entropy for that time period (level of uncertainty) as well as the presence of outliers in recent data.

It is unknown if this approach would also be successful with non-mainframe systems or systems that are compute intensive rather than I/O intensive.

7. Useful Related Materials

Buzen, Jeffrey and Annie Shum, "MASF -- Multivariate Adaptive Statistical Filtering", CMG1995 Proceedings, pp. 1-10.

Efron, Bradley, Trevor Hastie, Iain Johnstone and Robert Tibshirani, Statistics Department, Stanford University, Jan 9, 2004, "Least Angle Regression".

Efron, B., Hastie, T. J., Johnstone, I. M., and Tibshirani, R. (2004). "Least Angle Regression (with Discussion)." Annals of Statistics 32:407–499.

Funes, F., Penalized Regression Methods for Linear Models in SAS/STAT®, SAS Institute

Harrell, F. E. (2001). Regression Modeling Strategies. New York: Springer-Verlag.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001). The Elements of Statistical Learning: Data Mining, Inference,and Prediction. New York: Springer-Verlag.

Matsunaga, Andréa and José Fortes, "On the use of machine learning to predict the time and resources consumed by applications", 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

Simpson, P., *et. al*., "Assessing Model Fit and Finding a Fit Model", Sugi 29 Proceedings.

Tibshirani, R. (1996). "Regression Shrinkage and Selection via the Lasso." Journal of the Royal Statistical Society, Series B 58:267–288.

Trubin, Igor, "Exception Based Modeling and Forecasting", CMG2008 Proceedings

Trubin, Igor, "Capturing Workload Pathology by Statistical Exception Detection System", CMG2005 Proceedings.

Achieving CPU (& MLC) Savings through Optimizing Processor Cache

Todd Havekost, IntelliMagic

Customer experiences with z13 processors have confirmed that delivered capacity is more dependent than ever before on effective utilization of processor cache. After providing the framework for understanding and interpreting the enlightening metrics available from the SMF 113 records, this paper identifies potential ways to leverage those metrics to improve processor cache efficiency, thereby reducing CPU consumption and MLC software expense. Insights into the potential impact of various tuning actions are demonstrated using data from several real-life case studies.

1. Introduction

Processor cache utilization plays a significant role in CPU consumption for all z processors, but that role is more prominent than ever on z13 models. Achieving the rated 10% capacity increase on a z13 processor vs. its zEC12 predecessor (despite a clock speed that is 10% slower) is very dependent on effective utilization of processor cache.

This paper will introduce key processor cache concepts and metrics, laying the foundation for understanding the vital role processor cache plays in CPU consumption. Those metrics will be leveraged to identify specific ways to improve processor cache efficiency by optimizing LPAR topology and maximizing work executing on Vertical High CPs.

Throughout the paper several examples from real-life situations will be used to illustrate both the opportunities and the potential impacts of tuning actions.

2. Key Metrics (CPI and RNI)

The first key metric to consider is Cycles Per Instruction (CPI). CPI represents the average number of processor cycles spent per completed instruction. Conceptually, processor cycles are spent either productively, executing instructions and referencing data present in Level 1 (L1) cache, or unproductively, waiting to stage data due to L1 cache misses.[1] Keeping the processor productively busy executing instructions is highly dependent on effective utilization of processor cache.

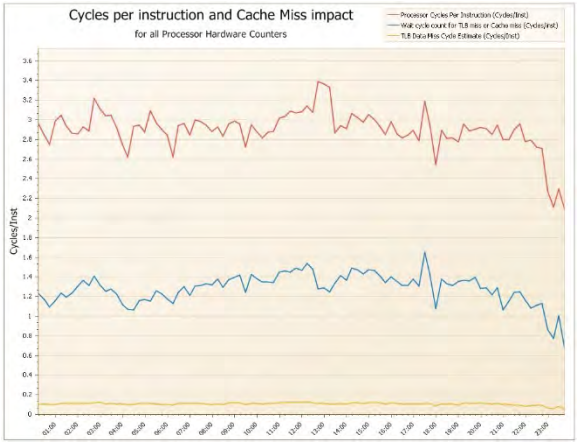


Figure 1. Cycles per instruction (CPI).

Figure 1 breaks out CPI into the "productive" and "unproductive" components just referenced.[2] The space above the blue and below the red line

[1] While "waiting to stage data" z processors leverage Out of Order execution and other types of pipeline optimizations behind the scenes seeking to minimize unproductive waiting.

[2] Another small component of CPI, Translation Lookaside Buffer (TLB) misses while performing Dynamic Address Translation, is represented by the yellow line on Figure 1.

reflects the cycles productively spent executing instructions for a workload. This would be the CPI value if all required data and instructions were always present in L1 cache, and it reflects the mix of simple and complex machine instructions in a business workload.

But since the amount of L1 cache is very limited,[3] many machine cycles are spent waiting for data and instructions to be retrieved from processor cache or memory into L1 cache. Note that these "waiting on cache" cycles, represented by the area under the blue line, represent a significant portion of the total, typically 35-55% of total CPI.[4] This highlights the magnitude of the potential opportunity if improvements in processor cache efficiency can be achieved and the importance of having clear visibility into key cache metrics.

A second key metric and one that correlates closely with the unproductive cycles represented by that blue line is Relative Nest Intensity (RNI). RNI quantifies how deep into the shared processor cache and memory hierarchy (called the nest) the processor needs to go to retrieve data and instructions when they are not present in L1 cache (see Figure 2). This is very important, because access time increases significantly for each subsequent level of cache and thus results in more waiting by the processor.

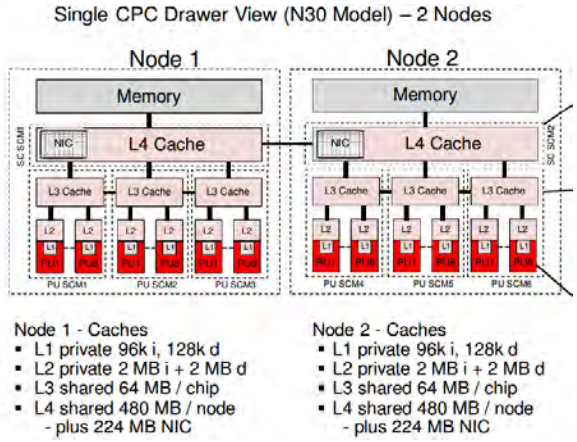


Figure 2. Architecture of the z13 cache [Sinram2015].

The formula to calculate RNI as provided by IBM is processor dependent and reflects the relative access times for the various levels of cache. Here is the z13 RNI formula [Kyne2017].

2.3 * (0.4*L3P + 1.6*L4LP + 3.5*L4RP + 7.5*MEMP) / 100

L3P = % of L1 misses sourced from the shared chip-level L3 cache
L4LP = % of L1 misses sourced from L3 or L4 cache in the same (local) node
L4RP = % of L1 misses sourced from L3 or L4 cache in a remote node or drawer
MEMP = % of L1 misses sourced from memory

Note that retrievals from memory (MEMP) have a weighting factor almost nineteen times higher than the factor for L3 cache (7.5 vs. 0.4), reflecting how many more machine cycles are lost waiting for data when it is not found anywhere in processor cache and must be retrieved from memory. IBM defines a threshold value for RNI of 1.0 above which a workload is considered to place a high demand on cache and thus may not achieve the rated capacity of a processor. But no matter the current RNI value, reducing it means less unproductive waiting cycles and thus less CPU consumption.

3. HiperDispatch

A key technology to understand when seeking to reduce RNI is HiperDispatch (HD). HD was first introduced in 2008 with z10 processors, but it plays an even more vital role on z13 models where cache performance has such a big impact.

With HD, the PR/SM and z/OS dispatchers interface to establish affinities between units of work and logical CPs, and between logical and physical CPs. This is important because it increases the likelihood of units of work being re-dispatched back on the same logical CP and executing on the same (or nearby) physical CP. This optimizes the effectiveness of processor cache at every level, by reducing the frequency of processor cache misses, and by reducing the distance (into the Nest) required to fetch data.

When HD is active, PR/SM assigns logical CPs as Vertical Highs (VH), Vertical Mediums (VM), or Vertical Lows (VL) based on LPAR weights and the number of physical CPs. VH logical CPs have a 1-1 relationship with a physical CP. VMs have at least a 50% share of a physical CP, while VLs have a low share of a physical CP (and are subject to being "parked" when not in use).

Work executing on VHs optimizes cache effectiveness, because its 1-1 relationship with a physical CP means it will consistently access the

[3] 96 KB for instructions and 128 KB for data per processor core on z13 models [Sinram2015].

[4] Based on the author's analysis of CPI data from 45 sites to date. This leads to the rule of thumb that a 10% reduction in RNI correlates to a 5% reduction in CPU consumption.

same processor cache. On the other hand, VMs and VLs may be dispatched on various physical CPs where they will be contending for processor cache with workloads from other LPARs, making the likelihood of their finding the data they need in cache significantly lower.

This intuitive understanding of the benefit of maximizing work executing on VHs is confirmed by multiple data sources. One such source is calculated estimates of the life of data in various levels of processor cache (derived from SMF 113 data). Commonly, cache working set data remains in L1 cache for less than 0.1 millisecond (ms), in L2 cache for less than 2 ms, and in L3 cache around 10 ms. This means that by the time an LPAR gets re-dispatched on a CP after another LPAR executed there for a typical PR/SM time-slice of 12.5 ms, its data in L1, L2 and L3 caches will all be gone. The good news is those working sets will be rebuilt quickly from L4 cache, but the bad news is that each access to L4 cache may take 100 or more cycles [for more detail on this analysis see Houtekamer2016].

This thesis that work executing on VHs experiences better cache performance is further substantiated by analyzing RNI data at the logical CP level. At the beginning of the case study presented in Figure 3, the Vertical CP configuration for this system consisted of three VHs and two VMs. Note that the RNI values for the two VM logical CPs (CPs 6 and 8) were higher than the RNIs for the VHs.[5]

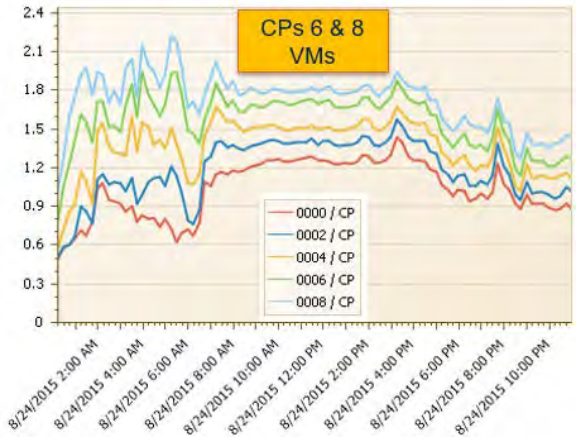


Figure 3. RNI by Logical CP Sys3 (3 VHs,2 VMs).

After the deployment of additional hardware caused the Vertical CP configuration to change to five VHs (Figure 4), the greatest reductions in RNI occurred for CPs 6 and 8. Now that they had also become VHs and were no longer experiencing cross-LPAR contention for processor cache, their RNIs decreased to a level comparable to the other VHs [Havekost2016].

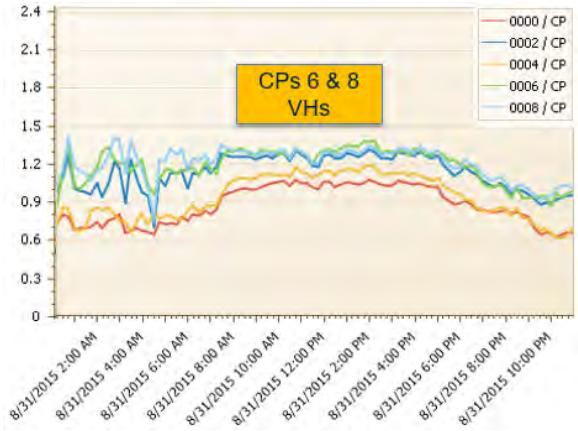


Figure 4. RNI by Logical CP Sys3 (5 VHs).

Up to this point we have seen that CPI decreases when there is a reduction in unproductive machine cycles waiting for data and instructions to be staged into L1 cache. Reducing CPI translates directly to decreased CPU consumption, which when occurring at peak periods results in reduced IBM Monthly License Charge (MLC) software expense.

Opportunities to optimize processor cache are worth investigating, because those unproductive waiting cycles typically represent at least one third of overall CPU, and often one half or more. Fortunately, the mainframe is a very metric rich environment, and the RNI metric is available that correlates to those unproductive waiting cycles.

Note that optimizing processor cache can have a particularly big payoff on z13 processors, which are more dependent than ever before on effective processor cache utilization. Two primary ways to reduce RNI will now be considered, by optimizing LPAR topology, and by maximizing work executing on VHs.

[5] RNIs for VMs and VLs are routinely higher than for VHs throughout the dozens of use cases reviewed by this author.

4. LPAR Topology

PR/SM dynamically assigns LPAR CPs and memory to hardware chips, nodes and drawers seeking to optimize cache efficiency. This topology can have a very significant impact on processor performance, because remote cache accesses can take hundreds of machine cycles.[6] The reader is encouraged to reference Figure 2 to provide a framework for the following discussion.

A z13 processor has one to four CPU drawers (along with some number of I/O drawers), and two nodes in each drawer. Each active physical CP (labelled "PU" on Figure 2) has its own dedicated L1 and L2 cache.

When a unit of work executing on a CP on a given Single Chip Module (SCM) accesses data in L3 cache, the first level that is shared (and thus is part of the "nest" and RNI metric), that access can be on its own SCM chip, "on node" (on a different SCM within this node), "on drawer" (on the other node in this drawer), or "off drawer". The more remote the access, the greater the number of machine cycles required, often hundreds of cycles. Similarly, access to L4 cache and to memory can be "on node", "on drawer", or "off drawer."

A first example showing the impact of LPAR topology on RNI is begins with Figure 5.

	Chip 2
Drawer 2 Node 1	SY22 VM00
	SY22 VM01
	SY22 VL02
	SY18 VH00
	SY18 VH01
	SY18 VH02

	Chip 2	Chip 3
Drawer 3 Node 1	SY18 VM03	**SY03 VH00**
	SY18 VM04	**SY03 VH01**
	SY16 VM00	**SY03 VH02**
	SY16 VM01	SY03 VM03
	SY16 VL02	SY03 VM04
	SY16 VL03	SY16 VL04
	SY20 VM00	
	SY20 VL01	

Figure 5. LPAR Topology – Example 1.

Each entry in this diagram represents a general purpose logical CP and indicates the system identifier, polarity (VH, VM or VL), and relative logical CP numbered starting from 0. VHs are highlighted in bold. zIIP CPs have been removed because this analysis is focused on general purpose CPs.

In this scenario, the logical CPs from several LPARs are competing for physical CPs on three chips. Green and orange shading has been added to compare two primary Production systems that execute similar data sharing workloads, each having three VHs and two VMs.

Note that the VHs and VMs for System 3 (shaded green) are collocated on the same chip. The "RNI by Logical CP" chart for this system appeared earlier in Figure 3. The RNIs for the two VM CPs on that system were higher than for the VHs, but not dramatically so.

Contrast that with the topology of System 18 (in orange), which has three VHs in Drawer 2, but its two VMs reside in a different drawer, Drawer 3. These VMs are located not on a different chip in the same node, not on a different node in the same drawer, but in the most remote possible location, in a different drawer.

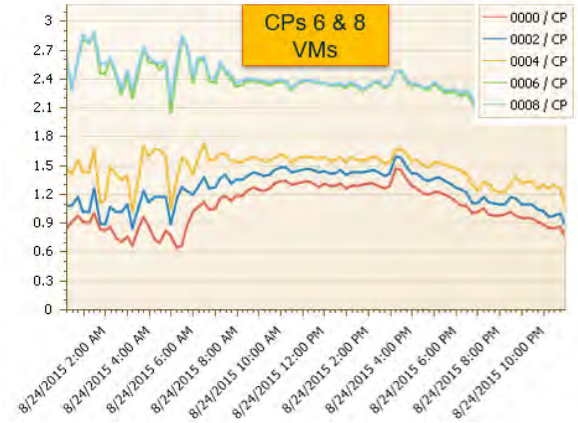


Figure 6. RNI by Logical CP Sys18 (3 VHs,2 VMs).

Figure 6 shows the extent of the negative impact to RNI that results from cache references for the VMs on system 18 routinely travelling across drawers. The gap in RNI between the two VMs (CPs 6 and 8) and the three VHs is much larger on this system with the adverse LPAR topology than it was on system 3 (from Figure 3), where the VHs and VMs were collocated on the same chip. And Figure 7 shows the improvement in RNI when CPs 6 and 8 changed from VMs to VHs was also greater, because it is very likely that all five of those VHs are now collocated on the same chip.

[6] LPAR topology data is provided by the SMF Type 99 Subtype 14 record. As opposed to some SMF 99 subtypes which can generate overwhelming volumes, the volume of subtype 14 data is very manageable, one record per logical CP every five minutes, making this another data source that warrants collection and analysis.

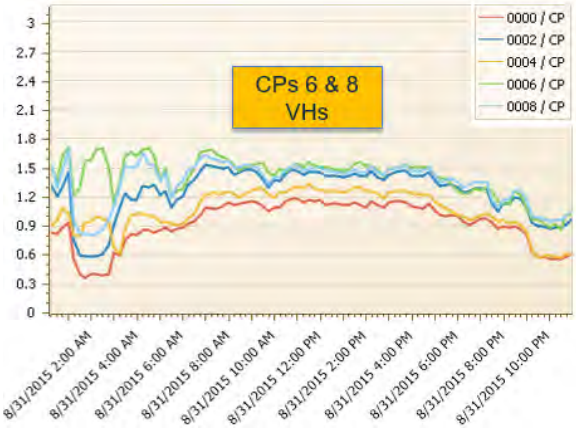


Figure 7. RNI by Logical CP Sys18 (5 VHs).

Additional detailed analysis in Figure 8 breaking out the waiting cycles of CPI by the various causes shows the impact of all that far slower off-drawer access on system 18's CPI. Note the significant increases in cycles spent on off-drawer accesses for system 18 for L3 cache (gray), L4 cache (light blue), and memory (dark blue). The cumulative impact of the L3 and L4 off-drawer accesses added more than an extra half a cycle (0.57) per instruction to system 18.

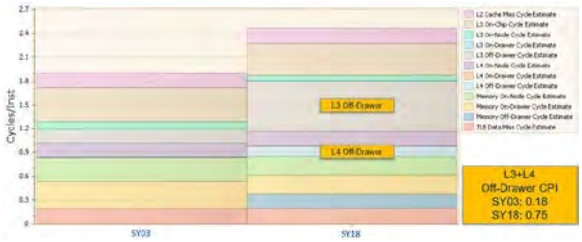


Figure 8. Estimated Impact Cache Misses.

Drilling further into this cache miss data on system 18 by logical CP (Figure 9) demonstrates how those off-drawer accesses predominantly occurred on the two VM logical CPs, CPs 6 and 8, as expected. Note that work executing on those two VMs required two full additional cycles per instruction compared to work executing on the VHs.

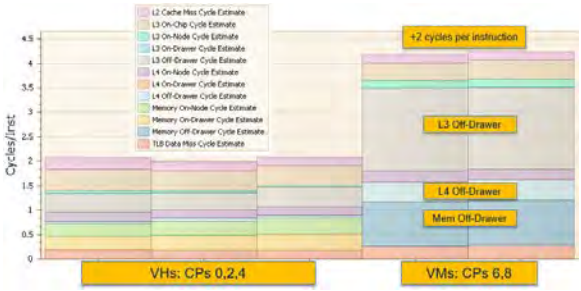


Figure 9. Cache Miss Impact by Logical CP Sys18.

When there is a significant disparity between the RNIs for VMs and VHs on the same system, an investigation of the LPAR topology is warranted. This analysis also highlights another benefit of having more work executing on VHs. In addition to avoiding cross-LPAR contention for cache from other LPARs (as identified earlier), since PR/SM is very likely to collocate VHs, the incremental work now executing on these VHs is unlikely to be subject to the cross-node or cross-drawer access times often experienced on VMs.

A second LPAR topology scenario involves an entirely different kind of opportunity. In the use case depicted in Figure 10, PR/SM configured all ten VH CPs from two Production LPARs in the same node on a single drawer. The outcome of this configuration was that the two LPARs were sharing the 480 MB L4 local cache of that single node between them.

Drawer 2 Node 1	Chip 1	Chip 2
Before: 1 node 480 MB L4L cache	SY07 VH00	SY17 VH00
	SY07 VH01	SY17 VH01
	SY07 VH02	SY17 VH02
	SY07 VH03	SY17 VH03
	SY07 VH04	SY17 VH04

Figure 10. LPAR Topology – Shared Node.

Increasing the memory allocation for both LPARs caused them to exceed the memory available in a single drawer and forced PR/SM to assign them to separate drawers and thus separate nodes. Now that these two LPARs had the L4 local cache of two nodes available to them (960 MB), we would expect more misses to be resolved from there. (The z13 RNI formula is repeated here to highlight how much lower the weighting is for L4 Local accesses than L4 Remote and Memory.)

2.3 * (0.4*L3P + 1.6*L4LP + 3.5*L4RP + 7.5*MEMP) / 100

	L4LP	L4RP	MEMP	RNI
Before	4.38%	0.91%	4.85%	1.48
After	5.84%	0.59%	3.82%	1.31

Figure 11. Impact of LPAR Topology Change.

Figure 11 shows that an additional 1.5% of L1 misses were now sourced from L3 or L4 local

node cache (L4LP in Green), reducing the frequency of accesses from L3 or L4 remote cache and especially from memory. This resulted in an 11.5% reduction in RNI and corresponding 6% increase in effective capacity. Note that this change did not require deploying any additional hardware, but instead involved utilizing the existing hardware more effectively.[7]

5. Maximizing Work Executing on VHs

A second way to reduce RNI is to maximize work executing on VHs. The two variables determining the Vertical CP configuration are (1) LPAR weights and (2) the number of Physical CPs. Each of these will be considered in detail.

5.1. Setting LPAR Weights

There are several ways to maximize work on VHs through setting LPAR weight values.[8] One is to adjust LPAR weights to increase the number of VHs for high CPU LPARs that currently have a significant workload executing on VMs and possibly even VLs. In Figure 12, a small weight change on a large LPAR changing the LPAR weight percentage from 70 to 71 percent increased the number of VHs from seven to eight.

12 CPs	% Shr	CP Shr	VHs	VMs	VLs	RNI
Before	70%	8.4	7	2	3	
After	71%	8.52	8	1	3	-2%

Figure 12. LPAR Weight Change on Large LPAR.

This resulted in a measured decrease in RNI of 2% for a given measurement interval, which correlated to a CPU reduction of 1%. 1% less CPU on a large LPAR can translate into a meaningful reduction in MLC software expense, especially when compared with the level of effort required to identify and implement this type of change. Benefits from tuning LPAR weights typically produce single digit percentage improvements as in this case, but there can be larger opportunities as we will observe.

A second way to increase work executing on VHs involves tailoring LPAR weights to increase the overall number of VHs assigned by PR/SM on a processor. The LPAR weight configuration of 30/30/20/20% in Figure 13 appears routine, but unfortunately on a z13 it results in zero VHs.[9]

6 CPs	% Shr	CP Shr	VHs	VMs	VLs
LP01	30%	1.8	0	2	2
LP02	30%	1.8	0	2	2
LP03	20%	1.2	0	2	1
LP04	20%	1.2	0	2	1

Figure 13. LPAR weight configuration with 0 VHs.

As Figure 14 shows, relatively small LPAR weight changes could increase the number of VHs from zero to two, and increase the amount of work eligible to execute on VHs from 0% up to 33%. On a comparable system in this environment, the RNI for work executing on VHs was 20% lower than work on VMs, so it is likely this change would significantly reduce CPU consumption on this processor.

6 CPs	% Shr	CP Shr	VHs	VMs	VLs
LP01	34%	2.04	1	2	1
LP02	34%	2.04	1	2	1
LP03	16%	0.96	0	1	2
LP04	16%	0.96	0	1	2

Figure 14. Adjusted LPAR weights creating 2 VHs.

If the characteristics of the workloads on LPARs change between shifts, automating LPAR weight changes corresponding to those shifts may be another way to increase the workload executing on VHs. And finally, fewer, larger LPARs may be a configuration option to increase the size of the workload executing on VHs.

[7] IBM specialists assisting us at my former employer identified this opportunity and the workaround to create the desired topology, and measured the increase in effective capacity (which aligned very well with the RNI rule of thumb).

[8] The specifics of how PR/SM determines Vertical CP assignments based on LPAR weights and the number of physical CPs is beyond the scope of this paper [for details see Havekost2017].

[9] This is the configuration after an IBM June 2016 z13 microcode change [see Snyder2016 for details].

5.2. Increasing Number of Physical CPs

Along with LPAR weights, the second variable in the equation PR/SM uses to assign VHs is the number of physical CPs. We will now consider the options and considerations for maximizing work executing on VHs through increasing the number of physical CPs.

One way to increase the number of physical CPs is to utilize sub-capacity processor models. These models provide additional physical CPs for the same MSU capacity, which translates into more VHs without incurring additional hardware expense.

If single engine speed requirements or other considerations in your environment do not require full capacity models, sub-capacity models could be selected when upgrading to new generation technologies. This approach adds physical CPs, along with the associated L1 and L2 processor cache, without incurring additional hardware expense. This would result in more VHs and thus greater processor cache efficiency.

Total	zEC12-711	z13-710	z13-617	z13-525
MSUs	1593	1632	1610	1603

Figure 15. z13 models with similar capacity ratings.

Figure 15 shows various z13 models that have similar MSU capacity ratings as a zEC12-711. Opting for sub-capacity models in this example would result in seven to fifteen more physical CPs than the full capacity z13-710, significantly increasing the available processor cache and workload executing on VHs. Selecting a sub-capacity model would likely result in additional realized capacity due to the CPU savings resulting from increased processor cache efficiency, even though the rated capacity would be comparable to the z13-710.

A second way to increase the number of physical CPs is to install or deploy additional capacity, leveraging a one-time hardware expense to achieve greater, recurring MLC software expense savings.

Traditional mainframe capacity planning has been predicated on the assumption that running mainframes at high utilizations is the most cost-effective way to operate. The results of this processor cache analysis and the case study we will consider shortly challenge that approach. They indicate that in many cases significant reductions in MSU consumption and thus MLC software costs may be achieved by operating at lower utilizations due to the increased impact that cache effectiveness has on z13 processors.

Understandably, developing the business case to acquire additional hardware capacity may be challenging. But many sites would not require that business case because they already have a business practice of pre-purchasing additional capacity they do not immediately deploy.[10] But even these sites tend to deploy that previously acquired capacity in a "just in time" manner, rather than reaping the recurring benefits of MLC software savings by deploying surplus capacity as soon as it is acquired.

In either case, whether acquiring additional hardware capacity or deploying previously acquired capacity, the framework for reevaluating this approach is that in most mainframe environments software represents a much larger expense than hardware, with MLC software typically constituting the single largest expense line item. If a one-time hardware acquisition expense achieves software savings which are realized on a recurring basis year-after-year, there may be a strong business case for acquiring or deploying hardware capacity that significantly exceeds the business workload requirements.

One reservation frequently expressed about this approach is that Independent Software Vendor (ISV) licenses are a barrier to deploying surplus capacity, because many ISV contracts are capacity-based and not usage-based. Capacity-based ISV contracts can severely limit your operational flexibility until they are renegotiated to become usage-based. A proactive initiative to renegotiate any capacity-based ISV contracts can place you in the enviable position of having the flexibility to configure your environment in the most cost-effective manner going forward.[11]

[10] Reasons for this practice include negotiating a volume discount or long-term lease, avoiding the procurement effort involved with frequent acquisitions, or acquiring capacity required for seasonal peaks, which is then typically deactivated for the remainder of the year.

[11] This ISV contract renegotiation initiative was successfully achieved at my previous employer.

The following use case shows the potential magnitude of the impact of deploying hardware on processor cache efficiency. The case involves a high RNI workload and hardware capacity that had been previously acquired but not deployed.

Figure 16 is a weekly tracking chart that compared the Production business online workload (red line) versus CPU consumption (blue line). For the last several months in a zEC12 environment, these values tracked very closely. But after the four processors were upgraded to z13-711 models, CPU consumption far exceeded business transaction workload.

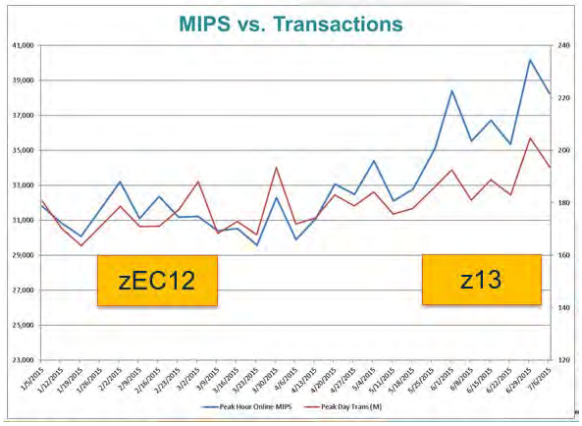


Figure 16. CPU impact of z13 implementations.

As stated earlier, the cycle speed on z13 processors is 10% slower than the zEC12, yet rated capacity is 10% higher. This high RNI (1.6) workload did not achieve the improvement in cache effectiveness the z13 relies upon to achieve this increased capacity, resulting in a capacity shortfall of 4000 MIPS versus the zEC12 configuration.

When additional capacity (z13-716 models) was deployed that enabled most of the workload to execute on VHs, MIPS consumption was reduced dramatically (see Figure 17). This resulted in 9000 MIPS savings versus the prior z13-711 configuration for equivalent business workloads.

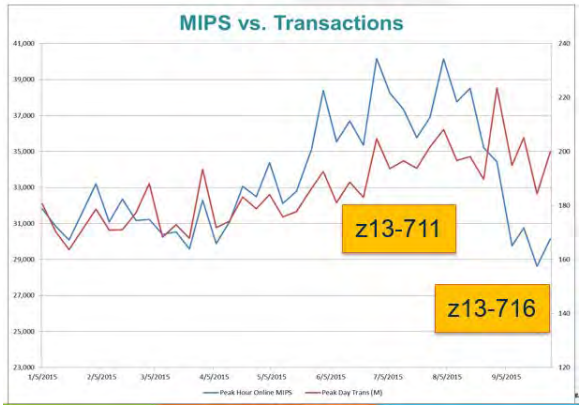


Figure 17. CPU impact of z13-716 upgrades.

Ultimately all the previously acquired capacity was deployed through upgrades to z13-726 models, resulting in a 13,000 MIPS reduction from the z13-711 configuration (see Figure 18). The primary factor driving this final round of savings was not processor cache efficiency but a reduction in the MSU/CP ratio, which we will now consider.

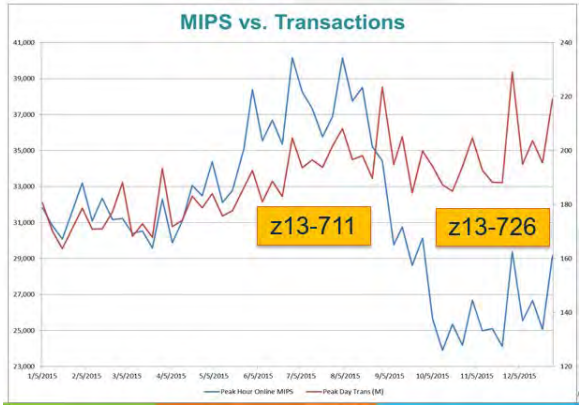


Figure 18. CPU impact of z13-726 upgrades.

6. MSU/CP Ratio

Another important factor impacting measured CPU consumption that adds to the business case for acquiring or deploying additional hardware relates to the multiprocessing (MP) effect. The MP effect applies to any hardware environment, and reflects the fact that adding cores (CPs) increases the overhead required to manage the interactions between physical processors and shared compute resources.

To account for the typical MP effect when running at relatively high utilizations, the MSU/CP ratios in IBM's processor ratings are not linear, but decrease significantly as more CPs are added as Figure 19 shows [Kyne2015].

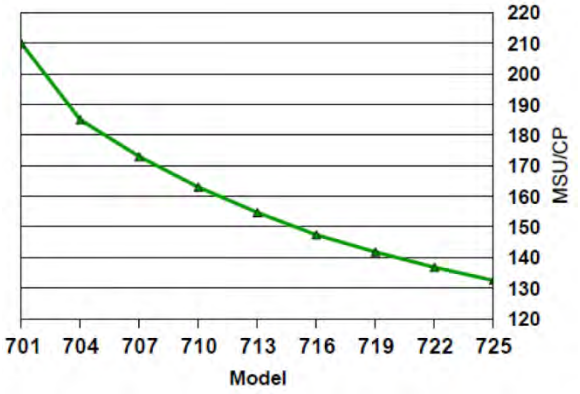


Figure 19. MSU/CP ratios for various z13 models.

But if the business workload remains the same when CPs are added, overall processor utilization will decrease, and additional overhead from the MP effect is likely to be negligible. In that case, a lower MSU/CP rating translates directly into lower MSU consumption for the same workload. Figure 20 compares the MSU/CP ratios for the four processor models in the previous use case.

CEC	MSUs/CP	vs zEC12-711	vs z13-711
zEC12-711	144.8		-9.7%
z13-711	160.4	10.7%	
z13-716	147.4	1.8%	-8.1%
z13-726	131.3	-9.3%	-18.1%

Figure 20. MSU/CP ratios from use case.

The Orange value shows the 10%+ increase associated with the original z13-711 implementation, one that the High RNI workload fell far short of achieving. The Green value shows the decrease of 18% in the MSU/CP rating for the z13-726 from the z13-711. Or expressed another way, a workload that would generate 1000 MSUs on a 711 processor would generate only 819 MSUs when run on a 726, solely due to the reduced MSU/CP rating. This is independent of the CPU savings from improved processor cache efficiency that would also accompany this change.

This means that there are two significant benefits of deploying additional hardware capacity while running at lower utilizations that combine to create even larger savings. First, less CPU is consumed due to operating efficiencies from more effective use of processor cache. And second, the CPU that is consumed translates into fewer MSUs, due to the decrease in the processor MSU/CP ratings.

7. Summary and conclusions

Processor cache performance plays a more prominent role than ever before in the capacity delivered by z13 processors. And this is a trend that is likely to continue with future processor models, due to the engineering challenges associated with increasing cycle speeds.

Since unproductive cycles spent waiting to stage data into L1 cache typically represent one third to one half of overall CPU, it is incumbent on the mainframe performance analyst to have clear visibility into the key metrics and to leverage those metrics to optimize processor cache. This paper presented methods to reduce RNI and CPU consumption, along with case studies confirming the effectiveness of these methods in real-life environments.

Another trend that is very likely to continue is that software expenses will consume a growing percentage of the overall mainframe budget, while hardware costs will represent an ever-smaller percentage. Considering these factors, it is indeed time for a thorough re-examination of the traditional assumption of mainframe capacity planning that running mainframes at high utilizations is the most cost-effective way to operate.

References

[Sinram2015] Horst Sinram, *z/OS Workload Management (WLM) Update for IBM z13, z/OS V2.2 and V2.1*, SHARE Session #16818, March 2015.

[Kyne2017] Frank Kyne, Todd Havekost, and David Hutton, *CPU MF Part 2 – Concepts*, Cheryl Watson's Tuning Letter 2017 No. 1.

[Houtekamer2016] Gilbert Houtekamer, Wim Oudshoorn, and Todd Havekost, *Getting More MIPS out of your processors*, CMG Proceedings, 2016.

[Havekost2016] Todd Havekost, *Achieving Significant Capacity Improvements on the IBM z13: User Experience*, Share Session #18345, March 2016.

[Havekost2017] Todd Havekost, *Beyond Capping: Reduce IBM z13 MLC with Processor Cache Optimization*, Share Session #20127, March 2017.

[Snyder2016] Bradley Snyder, *z13 HiperDispatch – New MCL Bundle Changes Vertical CP Assignment for Certain LPAR Configurations*, IBM TechDoc 106389, June 2016.

[Kyne2015] Frank Kyne, *A Holistic Approach to Capacity Planning*, Cheryl Watson's Tuning Letter 2015 No. 4.

FICON CUP Diagnostics and the IBM Health Checker for z/OS

Stephen R. Guendert, Ph.D.

The IBM FICON Control Unit Port (CUP) provides an enhanced communications connection between IBM z Systems hosts and FICON switching devices, over which in-band management and communications can occur. Within IBM z Systems, there are management programs that can supervise switching devices by communicating through CUP to those switches. A relatively recent enhancement to this CUP communications capability is generally known as the CUP Diagnostics function. This enhancement is a collaborative effort to provide IBM Z Systems with visibility into fabric events that the z/OS operating system has historically been unaware of.

INTRODUCTION

As IBM Z environments have grown, and configurations have become more complex, FICON fabric issues can result in unacceptable Input/Output (I/O) service times. Resource Measurement Facility (RMF) device activity reports might show average service times that are higher than normal, while matching I/O queuing reports show abnormally high “initial command response” times on a subset of the paths to a device. Before these CUP Diagnostic enhancements were introduced, it was problematic to identify a single root cause for these issues or to identify where in the fabric a problem might have originated. CUP Diagnostics enables the ability for the CUP to proactively notify the z System that there are abnormal conditions in the fabric that might impact either the performance or reliability of the I/O traffic traversing through the fabric. This paper reviews the basic functionality of FICON CUP and then introduces the new FICON Management Server (FMS) and CUP Diagnostics capabilities now available to be used by the IBM Health Checker for z/OS to gain better insight into the health and robustness of FICON Storage Area Network (SAN) fabrics.

FICON CUP BASICS

The Control Unit Port (CUP) function allows a z/OS system to communicate with the FICON Director through standard channel program protocols. This includes control functions like blocking and unblocking ports, performance monitoring, and error reporting functions. The CUP device is implemented as a standard device control unit via firmware implementation on a switch that allows a z/OS system to issue appropriate channel program Channel Command Word (CCW) commands to it. The CUP device is defined in the I/O configuration as a switch device and is brought online to z/OS. The control unit definition for the CUP device consists of one or more channel paths attached to the switch with the reserved port address destination of 0xFE, which is defined by the FICON architecture as the address of the CUP. Therefore, I/O requests routed to this destination port are directed to the CUP.

The FICON CUP provides an in-band management interface defined by IBM that defines the CCWs and data payloads that the FICON host can use for managing the switch. The protocol used is the IBM version of the ANSI FC-SB4 single-byte command code specification, which defines the protocol used for transporting CCWs to the switch, and for the switch to direct data and status back. FICON CUP becomes available when the FMS optional license is installed and enabled on FICON switching devices.

FICON CUP is a direct architectural descendant of the CUP that ran on ESCON Directors. The IBM 9032-5 ESCON Directors had an in-band management capability that utilized an embedded port in the control processing cards to provide a communications path to an MVS console. This was used for three primary purposes: first, reporting hardware (Field-Replaceable Unit [FRU]) errors up to MVS (helpdesk), second,

allowing and prohibiting ports (the world's first "zoning" mechanism) by using Prohibit Dynamic Connectivity Mask (PDCM), and third, basic performance monitoring. When switched FICON was being introduced, IBM wanted to make certain that its mainframe customers would have a consistent management look and feel between ESCON and FICON, so CUP was carried forward to FICON. Today, FICON CUP support is provided by all mainframe storage and SAN vendors, but customers must implement the optional FMS license to make it functional on their FICON switches and directors. Today's more advanced CUP is still used for the three primary functions listed above, but it has been enhanced to also provide RMF reporting for the FICON switching devices and to help implement FICON Dynamic Channel Path Management (DCM).

TRADITIONAL USES OF FICON CUP

FICON CUP provides a defined CCW and payload interface for host in-band management and collection of FICON switch performance metrics, resulting in the collection of RMF 74 subtype 7 records, more commonly known as the FICON Director Activity Report. Through the use of this record and report a user can gain a good understanding about frame pacing delay (that is, buffer credit starvation) and many other analysis capabilities. RMF provides online, interactive performance monitoring and long-term overview reporting with post-processor reports.

Some RMF reports that can assist in analysis of fabric problems are:

- Channel path activity report
- Device activity report
- I/O queueing activity report
- FICON Director activity report
- Enterprise Disk Systems (ESS) Link Statistics report

However, RMF is not so effective at helping a user troubleshoot problems such as fabric congestion, device congestion, bad cables, failing optics, or FRU problems in a switched fabric. What is really needed is broader architecture that defines a richer set of data that can be supplied from fabric components to the IBM z Systems components. That can be accomplished through a tighter integration of CUP on switching devices and enhanced host based management programs. Host-based management programs can manage FICON switches by sending CCW commands to the switch control unit defined in the I/O Configuration Data Set (IOCDS) and Hardware Configuration Definition (HCD). A FICON switching device (that is, Director-class or fixed configuration switch) that supports CUP can be controlled by one or more host based management programs or switch consoles. Control of the FICON switches can be shared between these options. CUP commands, which are channel command words (CCWs), monitor and control FICON switch functions. There are 42 CUP commands, or CCWs, for monitoring and control of FICON switch device functions. CUP commands have generally been oriented towards management of a single switch even though the use of CUP in a cascaded FICON environment is fully supported.

It has become very apparent over time that as fabrics become more complex it is essential that they become proactively monitored by an IBM Z System. Although very mature and robust by most standards, there are a few challenges in the way that z/OS has traditionally defined and worked with fabrics. For example, the z/ OS IOCDS definitions include F_Ports, relative to Control Units and Devices, but there are no definitions for InterSwitch Links (ISLs), routing topologies, or distance extension facilities. Subsequently, these fabric components are essentially invisible to z/OS and z/OS can do nothing to react to specific problems in the fabric. Rather, z/OS just suffers performance degradations and reliability exposures until the problems manifest in a secondary manner, such as a serious degradation of I/O performance or reliability, that it is capable of detecting. CUP Diagnostics is a facility that enables the CUP to notify the IBM Health Checker for z/OS host management program about fabric events while also

supporting subsequent queries by the host to provide detailed information about its fabric topologies (that is, domains, ISLs, routing groups) and the location of, and details of, errors within those topologies.

INTRODUCTION TO CUP DIAGNOSTICS

The Health Checker for z/OS is an IBM software component that is used to monitor and detect performance and Reliability, Availability and Serviceability (RAS) issues in data center components including both the Computer Electronics Complex (CEC) and its peripherals. The IBM Health Checker for z/OS reports problems to various system operations components (that is, automation components) and initiates further analysis actions and remedial actions. FICON switching devices (that is, a storage networking fabric) sit in the middle, between the computing and storage components, so switches have a critical role in transporting data between the CEC and its peripherals. But, as described earlier, the preponderance of fabric operations are invisible to z/ OS, which cannot quickly react to or troubleshoot those issues. Errors within the fabric can be caused by either software or hardware problems or by frame traffic rates that exceed available capacity. Some of the potential issues that can cause problems in the fabric are:

- Defective cables or optics
- Congestion within the fabric
- I/O or switch configuration problems, such as incorrect cabling or routing
- Channel or control unit hardware or firmware issues
- I/O configuration definition errors

IBM, in a cooperative effort with Brocade, has defined CUP Diagnostics function, a new architectural component to be implemented in the Brocade Fabric OS (Brocade FOS) FMS code. CUP Diagnostics are the newest CUP CCWs and payloads that enable the IBM Z host to receive direct alerts about fabric problems and then proactively gather additional, detailed information about the fabric (for example, components, topologies, ISL routing, and fabric problems). FOS v7.2 initiated the CUP Diagnostics function by providing some basic, but limited, responses to the IBM Z host.

CUP DIAGNOSTICS AND FOS 7.1 AND FOS 7.2

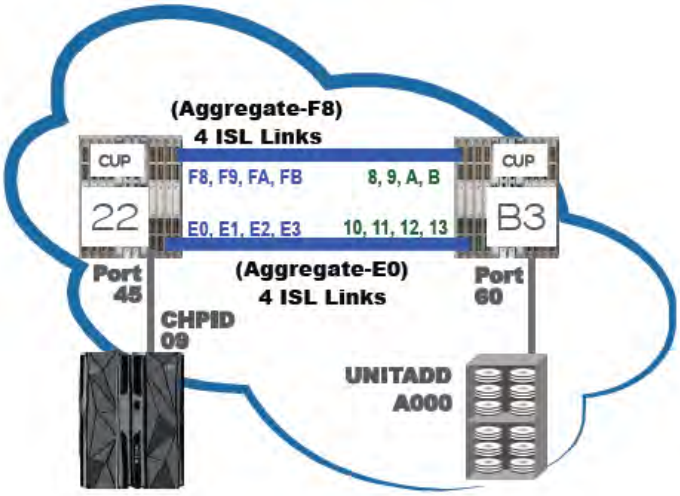
In March 2014, two new functions became available in FOS v7.2 that were engineered into the CUP functionality. ROUTE and HEALTH keywords were provided, in order to be functional with the Display Matrix z/OS command. While z/OS has historically provided sophisticated I/O monitoring and recovery, this modified command exposes newly available fabric diagnostic data, provided directly from the switch, and makes some of the fabric operations become evident that previously were invisible to z/OS.

The ROUTE function provides important information back to the command issuer regarding the specific path that frames are traversing between a mainframe Channel Path ID (CHPID) and a device (that is, the to device TODEV keyword) or from a device to a mainframe CHPID (that is, the FROMDEV keyword). In a parallel development effort, two new IBM z/OS Health Checks were added so that z/OS could recognize some specific problems within the fabric. These functions are supported on all switches beginning with FOS v7.1.0c and Network Advisor 12.0.2 software levels or later.

On the IBM Z side, z/OS 1.12 and later, plus some required Authorized Program Analysis Reports (APARs) and Program Temporary Fixes (PTFs), are required with FOS v7.1.0c to support the CUP Diagnostics and IBM Health Checker for z/OS features added at that firmware level. Important Note: Even if the end user has no plans to use these new CUP features, the IBM Health Checker for z/ OS feature uses a previously undefined bit in the Sense data. If the user does not implement these APARs and PTFs, then any time the channel issues a Sense command to the CUP, and there is health information to report, an error message indicating an invalid command code is posted to the z/OS console. This might become annoying to the user.

To make good use of CUP Diagnostics, the user interface is often the ROUTE enhancement to the z/OS Display Matrix command. The DISPLAY M=DEV command allows a user to display the route through the SAN fabric for a specific device and channel path, by specifying the ROUTE keyword. The routing information returned includes all of the domains and ports that are in the path from the channel to the device. Since the usual intent is to find the frame routing that is being used in a cascaded fabric, the reporting of routing information is performed only when the channel is connected to a switch and the control unit definition for the channel path is defined in the I/O configuration with a two-byte link address (that is, cascaded FICON).

See the example in Figure 1. Here, a user wants to see how frames are being sent from the mainframe (CHPID 09) through the fabric in order to reach one of the disk arrays, A000. The user also wants a report about the health of the fabric along that path. The user issues the Display Matrix command with a ROUTE parameter and a HEALTH parameter. You can see the output that is displayed when the CUP Diagnostics function is executed on their behalf via the Display Matrix command.



Refer to Figure 1: Issuing the command: D M=DEV(A000,(09)),ROUTE=TODEV, HEALTH IEE583I hh.mm.ss DISPLAY M 058 DEVICE sdddd STATUS=ONLINE

```
Source to destination routing information follows:

 Switch Domain=22, Type=Source Director
                            Group
  Port Type  From   To     Agg Dyn Speed Misc
   45  Entry Chan   Agg-F8 ..  ..   16G  Static
   80  Exit  A000   B3OA   ..  ..   16G

 Switch Domain=B3, Type=Source Director
                            Group
  Port Type  From   To     Agg Dyn Speed Misc
   0A  Entry 22FA   B-360  F8  ..   16G  Static
   60  Exit  A000   B3OA   ..  ..   16G  .....
```

Figure 1. CHPID 9 using entry point 45 on CUP22 to send and receive frames to CU A000

In the routing portion of the display output, information for each switch in the path is displayed, which includes its domain and switch type. The definition of the switch type depends upon the number of switches in the path between the channel and device, and the position of the switch within the path. If there is only one switch, the switch type is shown as Only Director. If there is more than one switch, the first switch is known as the Source Director, and the last switch is known as the Destination Director. If there are any switches between the source director and the destination director (see Figure 2), each of those switches is known as an Intermediate Director.

From the example output, you can see that the route from the channel to the device travels through two switches, Domain ID 22 (that is, source director) and Domain ID B3 (destination director). The channel is connected to entry port 45 on switch Domain ID 22, as indicated by the “From” column for that port. The “To” column for the same port indicates that I/O requests originating from this port are routed to ISL aggregate FA (that is, four trunked links, as in Figure 2). Brocade FOS v7.2 did not make any distinction between trunked and nontrunked ISL connections. Therefore, only a single ISL link is shown in the output. Fabric Shortest Path First (FSPF) provided CUP Diagnostics with a list of possible egress ports, and CUP Diagnostics simply chose the first egress port shown in that list as the master ISL link.

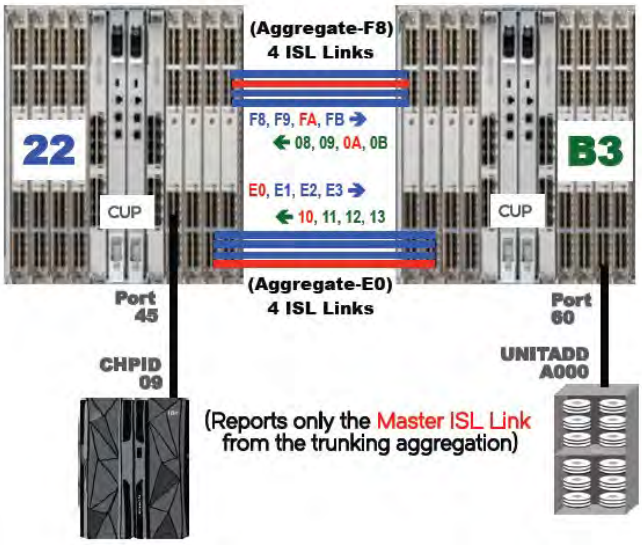


Figure 2. CHPID 9 could use a link on one of two hardware trunks to send and receive frames to CU A000.

For entry ports, the physical connection appears under the “From” column and represents either a channel, a control unit, or a single port on another switch. The logical connection appears under the “To” column and represents a single port or a group of ports on the same switch, depending on the routing and grouping methods. For exit ports, the logical connection appears under the “From” column and always represents a single entry port on the same switch. The physical connection appears under the “To” column and represents either a channel, control unit, or a single port on another switch. FOS v7.3 and beyond provide additional information about individual trunk ISL links that FOS 7.2 does not provide. If you examine Figure 2, you see that there are two trunks of four ISL links each. Each trunk is an aggregation of ISL links and is identified using the lowest port number in the group. In this case, the source switch is Domain ID 22, so the aggregation designations are for Domain ID 22 flowing frames to Domain ID B3. That makes the first ISL aggregation for Domain ID 22 aggregation-F8 and the second ISL aggregation for Domain ID 22 aggregation-E0.

If the command “D M=DEV(A000,(09)),ROUTE=FROMDEV, HEALTH” is issued, then the trunking aggregations are for connections from Domain ID B3 flowing frames to Domain ID 22. Its first ISL

aggregation is Domain ID B3 aggregation-08, and the second ISL aggregation is Domain ID B3 aggregation-10. The speed listed is the negotiated speed, in gigabits per second.

You can interpret the information for the destination director B3 in a similar manner. Entry port 0A is within the aggregation group F8, and it routes the I/O requests to port 60 on the same switch. Once again, there is no distinction between trunked and nontrunked ISL connections. Therefore, only a single ISL link is shown in the output chosen from the FSPF list. Although port 60 has data routed to it from multiple ISL entry ports on the switch, it is not possible to indicate that to the user in this version of Brocade FOS. Port 60 is not part of an aggregate group, therefore it contains ".." in that column. The storage device control unit is connected to port 60. Note that in this static routing example (that is, Port-Based Routing [PBR]), the information in the "Misc" column indicates that I/Os are being statically routed. In the version of CUP Diagnostics for Brocade FOS v7.2, there is no recognition of Device-Based Routing (DBR) or FICON Dynamic Routing (that is, Exchange-Based Routing) capabilities on ISL connections.

The following output shows the "health" display for the example above:

```
Health information follows:

  Fabric Health=No health issues

  Switch Domain=22, Health=No health issues
                                %Util  %Delay   Error Count Opt Signal
  Port Health                  Trn/Rcv Trn/Rcv   Trn/Recv    Trn/Recv
   45  Port Normal              99/99    0/0       0/0      -4.8/-6.6
   FA  Port Normal              99/99    0/0       0/0      -4.9/-6.6

  Switch Domain=B3, Health=No health issues
                                %Util  %Delay   Error Count Opt Signal
  Port Health                  Trn/Rcv Trn/Rcv   Trn/Recv    Trn/Recv
   0A  Port Normal              99/99    0/0       0/0      -2.1/-2.4
   60  Port Normal              99/99    0/0       0/0      -2.4/-2.1
```

Figure 3. Example health display output

For the health portion of the display, a description of the health of the fabric, each switch, and each port is displayed, as well as the following information for each port:

- The % transmit/receive utilization indicates the percent utilization of the total transmit or receive bandwidth available at the port.

- The % transmit delay is the percent of time that the frame transmission was delayed because no buffer credits were available on the port. The % receive delay is the percent of time that the port was unable to receive frames because all receive frames were utilized.

- The error count is the number of errors detected on the port that affect the transmission or receipt of frames on the port. This is a sum of the errors counted over the fabric diagnostic interval, which is set to 30 seconds by z/OS.

- The optical signal column indicates the signal strength of the fiber optic signal being transmitted/received by this port, in units of dBm (that is, denotes an absolute power level measured in decibels and referenced to 1 milliwatt)

This example of the health data reveals that there are no health problems within the fabric. The text provided in the fabric health, switch health, and port health is switch-vendor-specific. If any data is not valid, then ".." is displayed in the appropriate column.

BROCADE FOS 7.3 ENHANCEMENTS TO THE CUP DIAGNOSTIC CAPABILITY

In October 2014, FOS v7.3.0b became available for mainframe enterprises, and it hosted a plethora of new capabilities for CUP Diagnostics. Among its enhancements was the ability to provide information about complex routing topologies. This section describes these changes.

In the previous version of CUP Diagnostics (that is, FOS v7.2) only the first FSPF link of an ISL trunking aggregation was listed in the report that was provided in response to a CUP Diagnostics query. With the release of FOS v7.3, CUP Diagnostics was capable of providing a much more robust display of all of the members of an ISL aggregation/trunk. A common Aggregation Group Number (AGN) is assigned to indicate which ISL links are contained within the same trunk aggregation. Each port in the trunk aggregation is displayed with its own statistics. As an example, referring to Figure 4, a user would like to see how frames are being sent from the mainframe (CHPID 09) through the fabric, in order to reach one of their disk arrays, A000, similar to what was done at FOS v7.2. The user also wants a report about the health of the fabric along that path. The user issues the same Display Matrix command with a ROUTE and a HEALTH parameter. You can view the enhanced output displayed from CUP Diagnostics, via the Display Matrix command, as follows.

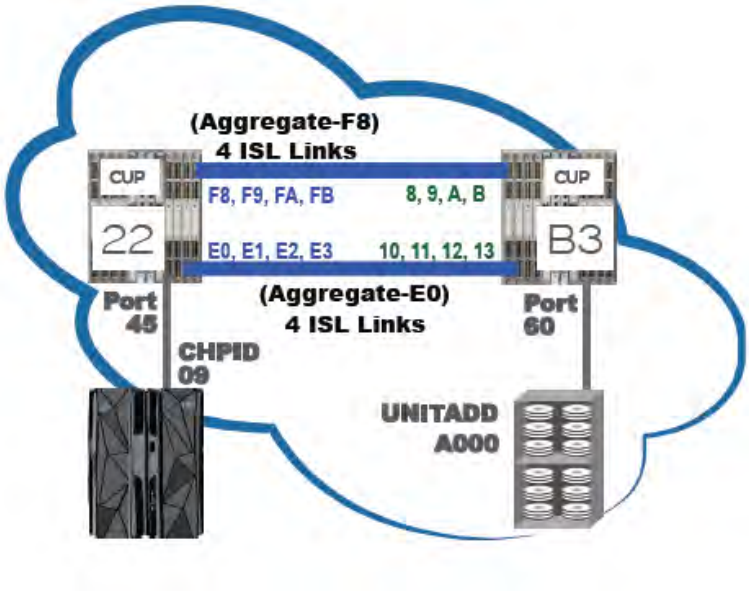


Figure 4a. Trunk (that is, aggregation) groups are assigned the lowest port in the trunk as the group number.

Referring to Figure 4: Issuing the command: D M=DEV(A000,(09)),ROUTE=TODEV, HEALTH IEE583I hh.mm.ss DISPLAY M 058 DEVICE sdddd STATUS=ONLINE

```
Source to destination routing information follows:

 Switch Domain=22, Type=Source Director
                                Group
  Port Type  From   To     Agg Dyn Speed Misc
   45  Entry Chan   Agg-F8 ..  ..    8G  Static Alt=1
   F8  Exit  A000   B3O8   F8  ..   16G
   F9  Exit  A000   B3O9   F8  ..   16G
   FA  Exit  A000   B3OA   F8  ..   16G
   FB  Exit  A000   B3OB   F8  ..   16G


 Switch Domain=B3, Type=Source Director
                                Group
  Port Type  From   To     Agg Dyn Speed Misc
   08  Entry A001   B360   ..  ..    8G  Static Alt=1
   09  Exit  A002   B360   F8  ..   16G  Static Alt=1
   0A  Exit  A003   B360   F8  ..   16G  Static Alt=1
   0B  Exit  A004   B360   F8  ..   16G  Static Alt=1
   60  Exit  Mult   CU     F8  ..   16G  .....
```

Figure 4b. Trunk (that is, aggregation) groups are assigned the lowest port in the trunk as the group number.

From the example output, you can see that the route from the channel to the device travels through two switches, Domain IDs 22 and B3. Domain ID 22 is the source director, and Domain ID B3 is the destination director. The channel is connected to entry port 45 on switch Domain ID 22, as indicated by the “From” column for that port. The “To” column for the same port indicates that I/O requests originating from this port are routed to ISL aggregate FA (that is, four trunked links; see Figure 5). From the “Agg” column, you can also determine that aggregate F8 consists of exit ports F8, F9, FA, and FB on the switch. Each of these exit ports is connected to a different port on switch domain B3, as shown in the “To” column for the ports. For example, port F9 routes I/O to port 09 on domain B3.

When a port is connected to a channel, “Chan” appears under the “From” or “To” columns of the display, depending on which direction was specified. Likewise, when a port is connected to a control unit, “CU” appears in the display. When an entry or exit port is connected to a single switch port, the domain and port number are displayed as a four-digit number under the “From” or “To” columns. When an entry port is connected to multiple ports on the current switch, a dynamic or aggregate group number is displayed under the “To” column, depending on the routing and grouping methods used.

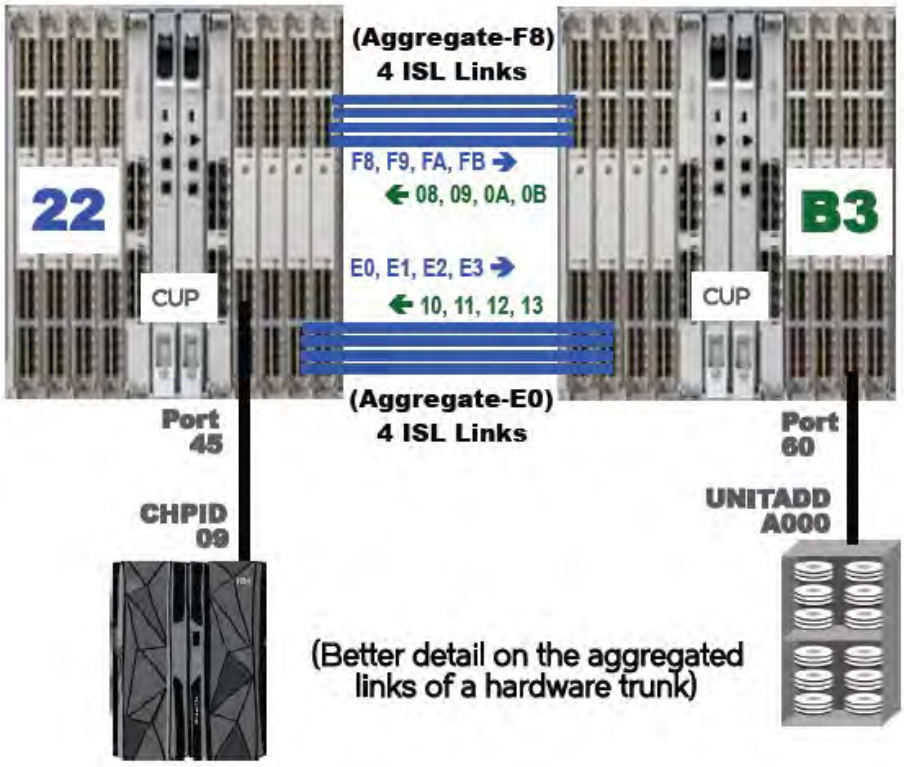


Figure 5a. A more detailed look at the ISL links contained in the Aggregation Groups.

When a set of entry or exit ports are part of an aggregate group of ports, those ports are assigned an aggregate group number. This group number appears in two places. First, it appears under the “Agg” column, to show which ports make up the aggregate group. Second, if I/O requests are being statically routed from an entry port to this aggregate set of ports, then the aggregate group number appears under the “To” column of the entry port. The value that is assigned for the aggregate group number is switch vendor-specific. Note that the definition of what is considered an entry port or exit port depends on the direction of the display request: channel to device (TODEV) or device to channel (FROMDEV). For example, when TODEV is specified, the port that is connected to the channel is considered an entry port, and the port connected to the control unit is considered an exit port. However, if FROMDEV is specified, the roles are reversed.

If the command, “D M=DEV(A000,(09)),ROUTE=FROMDEV, HEALTH” is issued, then the trunking aggregations are for connections from Domain ID B3 flowing frames to Domain ID 22. Its first ISL aggregation is Domain ID B3 aggregation-08, and the second ISL aggregation is Domain ID B3 aggregation-10. The speed listed is the negotiated speed, in gigabits per second.

The information for the destination director B3 can be interpreted in a similar manner. Entry ports 8, 9, A, and B are all within aggregate group F8, and they route the I/O requests to port 60 on the same switch (that is, Domain ID B3). Notice that port 60 can have data routed to it from multiple entry ports on the switch. Because there is not a single originating port to identify, the “From” column contains “Mult”. Port 60 is not part of an aggregate group and therefore contains “..” in that column. The storage device control unit is connected to port 60.

One of the major changes in this release of CUP Diagnostics is that both PBR and DBR are noted as “Static” under “Misc” in the report. DBR was not supported in CUP Diagnostics at Brocade FOS v7.2.

When an entry port uses static routing, the exit port or aggregate group number assigned to the port and the numbers of alternate paths are displayed. Detailed information about the alternate paths is not displayed.

Note that, in this static routing example (that is, PBR or DBR), the data for aggregate E0 is not displayed; however, the information in the “Misc” column indicates that I/Os are statically routed. One alternate route is defined, which is the aggregate E0 routing path. FICON Dynamic Routing (that is, Exchange Based Routing) is not supported at Brocade FOS v7.2, so only static routing is reported by v7.3 CUP Diagnostics.

This example of the health data shows that there are some health problems within the fabric on both switches, as shown by the fabric and switch text Port Error. You see a combination of bottleneck detected errors, slow draining device detected errors, and some ports that have been fenced. The text provided in the fabric health, switch health, and port health is switch-vendor-specific. If any data is not valid, ”..” is displayed in the appropriate column.

```
Health information follows:

  Fabric Health=Port Error

  Switch Domain=22, Health=BRD020-Port Error
                                %Util   %Delay    Error Count Opt Signal
  Port Health                  Trn/Rcv Trn/Rcv    Trn/Recv    Trn/Recv
   45  BRP031-Bottleneck Detect 99/0     0/0        0/0       -4.8/-6.6
   F8  No health issues         32/44    4/0        0/0       -4.9/-6.6
   09  BRP030-Port Fenced        0/0     0/0        0/0         ../-6.6
   FA  No health issues         32/44    4/0        0/0       -4.9/-6.6
   FB  No health issues         32/44    4/0        0/0       -4.9/-6.6


  Switch Domain=B3, Health=Port Error
                                %Util   %Delay    Error Count Opt Signal
  Port Health                  Trn/Rcv Trn/Rcv    Trn/Recv    Trn/Recv
   08  No health issues          0/0     0/0        0/0       -2.5/-3.8
   09  BRP030-Port Fenced        0/0     0/0        0/0         ../-3.3
   0A  No health issues         33/0     7/0        0/0       -2.4/-2.2
   0B  No health issues         33/0     4/0        0/0       -3.9/-2.7
   60  BRP033-SlowDrain Detect   0/0    95/0        0/0       -2.5/-3.8
```

Figure 5b. Health information for example in figure 5a

Blind Director Enhancement to CUP Diagnostics

A Blind Director is a domain where the CUP is not capable of processing Director Diagnostics Query messages, because its firmware level is prior to FOS v7.1 or it does not have FMS mode (CUP) enabled. See Figure 5 to view these two conditions. If a CUP Diagnostics query, where a domain after the Source domain is blind and incapable of responding to a CUP Diagnostic query request, it will return a code indicating the domain was down-level or that FMS mode was off. FOS v7.3 implemented several improvements to the ability to handle and report a Blind Director in the routing path of a Director Diagnostic Query.

Important Note: A user might have the FMS license installed on all FICON switching devices. Then during an upgrade from a previous version of Brocade FOS to v7.3, High Integrity Fabric (HIF) does not get set. At FOS 7.3, the FMS mode is dependent upon HIF to be enabled. If for any reason HIF is not enabled on a FOS 7.3 switch, then FMS will be disabled and a query by CUP Diagnostics will treat it as a

"Blind Director". For a cascaded CUP Diagnostics query, where the Source and Destination domains and the Source port are valid, and one of the domains after the Source domain is down-level (FOS v7.0 or earlier) or FMS mode is disabled, CUP Diagnostics indicates that the domain is incapable of allowing the query. FOS v7.3 handles domains that do not support CUP Diagnostics by supplying an appropriate reason code (for example, lower firmware level or FMS mode being off). Customers typically have cascaded switches at or close to the same Brocade FOS level.

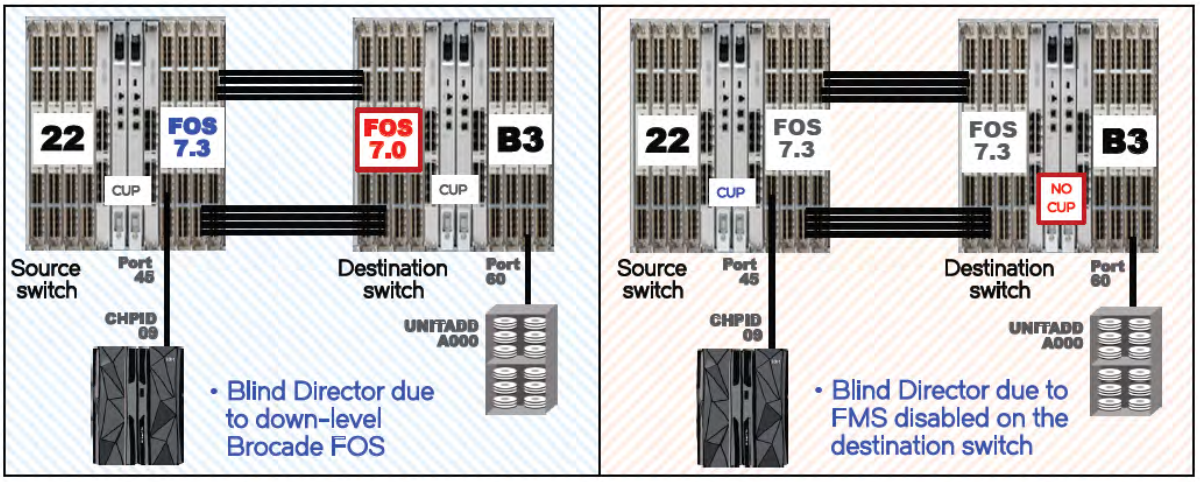


Figure 6. Blind Director is not capable of processing Director Diagnostics Query messages due to inadequate firmware levels (Left) and not having CUP enabled (Right).

BROCADE FOS 7.4 ENHANCEMENTS TO CUP DIAGNOSTICS

FOS v7.4 became available in June 2015, and this version included a very significant change to CUP Diagnostics. CUP Diagnostics was enhanced to include an interface to Brocade FOS Fabric Vision™ Monitoring and Alerting Policy Suite (MAPS) feature so that MAPS can notify FMS (that is, CUP) when an appropriately configured MAPS event is triggered. To support this CUP notification feature, MAPS enables a setting that is called FMS mode.

The rules that have an FMS notification action are part of all three of the default MAPS policies: dflt_aggressive_policy, dflt_moderate_policy and dflt_ conservative_policy. MAPS also supports the FMS action in custom rules. Some of the default rules are configured with the action FMS along with typical default actions of RASLOG, SNMP, and EMAIL. The FMS action can also be configured at the MAPS global action level. (See Table 1.) In the active policy, if the FMS notification action is configured for any triggered events, then MAPS sends the notification to CUP with event information. The following information is sent to CUP:

- Triggered event rule name
- Object and its type on which the event was triggered
- Severity of the event
- Condition on which the event was triggered

CUP uses any triggered MAPS events to create Health Check Alerts. These Health Check Alerts are sent to IBM Z hosts by using the FICON Unsolicited Alert Status mechanism, to notify the hosts of MAPS-detected conditions. The IBM Health Checker for z/OS can then issue subsequent FICON CCW

commands back to the CUP to retrieve detailed and contextual information about the alert. Here is an example of a user creating a custom rule that will send event information to CUP:

• mapsrule --create myRule -group ALL_F_PORTS -monitor CRC -timebase MIN -op ge -value 0 -action raslog,snmp,FMS

The Brocade FOS Fabric Vision MAPS module exposes a new action in the mapsconfig command. Modified definition of this command is given here:

• mapsConfig --actions If the user issues the Dashboard CLI command, it shows configured actions, including FMS, as part of the configured notification display.

Diagnostic Queries, HSC Codes, and Text Strings

When the host makes a health-type query to the CUP/Fabric, the CUP can supply HSCs for each node (Director or Port), that has an abnormal condition. Associated with each HSC is an HSC text string that describes the condition. This is the text that is displayed in the “Health” column of the MVS display. Each node can have two different types of events represented by HSC codes:

• Threshold event: - The threshold event is the result of a MAPS rule threshold being exceeded (for example, a CRC error). It is reported once and then cleared.

• Sticky event: - A sticky event is a port state that is reported by a MAPS rule being triggered (for example, traffic congestion and port fencing). A sticky state is a persistent state with a specific HSC value that is reported by CUP queries. It is continuously reported by CUP queries until another MAPS rule or event occurs that changes that state.

Traffic congestion can occur on either F_Ports or E_Ports. Bottlenecks are detected by special Brocade FOS components (that is, Bottleneck Detection or Fabric Performance Impact Monitoring) and these are now reported to CUP, which in turn alerts the z Systems of the condition. If both threshold events and sticky events exist, a query has the threshold event reported preferentially over the sticky event. Subsequent queries then report the sticky event, until it is changed by another event.

SUMMARY

When a FICON user installs and enables the optional FMS license on a Fibre Channel switching device, the user can utilize new CUP enhancements to provide a more robust fabric environment. Switch (CUP) Diagnostics provides new fabric-wide diagnostic command channel programs to enable z/OS to obtain fabric topology, collect diagnostic information such as performance data, determine the health of the fabric, generate a diagnostic log on the switch, and help users resolve problems in the fabric. A switch device can provide an unsolicited unit check, called a Health Summary unit check, which indicates that one or more switches or ports in the fabric are operating at less than optimal capability. This check triggers z/OS to retrieve the diagnostic information from the switch to further diagnose the problem. The sense data identifies the source and destination ports that are used for the query. A switch that has CUP Diagnostics support signals to z/OS that there is a health problem. The z/OS system initiates monitoring for the path, requesting diagnostic data from the switch at regular intervals. The problem might require intervention such as additional z/OS system commands or I/O configuration changes. After no errors or health issues are detected by the switch for at least two hours, the monitoring of the route is stopped and no longer appears in the report. The IBM Health Checker for z/OS reports if any switches that support CUP Diagnostics capabilities indicate unusual health conditions on connected channel paths. While z/OS has historically provided sophisticated I/O monitoring and recovery, these Health Checker reports expose newly available diagnostic data provided directly from the switch. These health checks enable additional insight into problems with a user’s fabric, such as hardware errors, I/O misconfigurations, or fabric congestion.

REFERENCES

Brocade Communications Systems. (2016). Brocade CUP Diagnostics and the IBM Health Checker for z/OS. San Jose, CA, USA: Brocade Communications Systems.

Brocade Communications Systems. (2017). Brocade Fabric OS FICON Configuration Guide, 8.1.0. San Jose, CA, USA: Brocade Communications Systems.

Guendert, S. (2018). Brocade Mainframe Connectivity Solutions. San Jose, CA, USA: Broadcom Limited.

Guendert, S., & Lytle, D. (2007). MVS and z/OS Servers Were Pioneers in the Fibre Channel World. Journal of Computer Resource Management, 3-17.

Guendert, S. (2007). To CUP or Not to CUP? That is the FICON Question! Proceedings of the 2007 Computer Measurement Group International Conference.

IBM Corporation (2015). IBM Health Checker for z/OS V2R2 User's Guide. Armonk, NY, USA: IBM Corporation.

Book Review: Software-Defined Data Infrastructure Essentials

By Stephen Guendert, Ph.D.

Software-Defined Data Infrastructure Essentials is the latest book from Greg Schulz. Greg Schulz is the Founder and Senior Analyst of the independent IT advisory and consultancy firm Server StorageIO (www.storageio.com). Greg has been a longtime supporter of CMG, and over the years he has spoken at numerous CMG conferences. I have had the privilege of knowing Greg for many years, and am the proud owner (and reader) of all three of his previous books. I reviewed two of those books in prior issues of CMG's *MeasureIT!* I jumped at the chance when Greg asked me to read and review this, his latest book.

The book, which is 600 pages in length, is in four parts. Part one focuses on "the big picture", covers industry trends and its two chapters cover 1) the fundamentals of data infrastructure and server/storage I/O technology, and 2) fundamentals of applications and data. Part two and its four chapters is a great deep dive tutorial into key concepts of server and storage I/O including software, server I/O, data organization at the bits/bytes level, and distance networking. Part three contains five chapters and continues where part two left off, with more of a focus on storage specifics: storage medium and device components, and data services functionality. Part four has three chapters tying everything together and looks at how all of the previously discussed data infrastructure components work as one.

The chapters I found the most informative were the material on PCIe and Non-volatile Memory Express (NVMe) technology. NVMe is the future of storage protocols and it is dramatically changing the storage and storage networking industry. I have read many articles and other books covering NVMe and in my opinion, *Software-Defined Data Infrastructure Essentials* is the definitive source for learning everything you need to know about NVMe and its related technology.

Just like Greg's other books, *Software-Defined Data Infrastructure Essentials* is extremely well written, and easy and enjoyable to read. Greg always does an outstanding job starting the discussion of what can be complex technology topics at a high level, and then carefully taking the reader deeper with more and more detail on each topic. This holds true for each chapter of the book. Greg's writing style is easy to follow which makes for a very positive learning experience when reading something that you may not be at all familiar with in the least.

In my experiences consulting with customers, and attending industry conferences around the world, one of the things that stands out is a rapid influx of the new generation into IT fields, especially storage. University computer science curricula do a great job teaching fundamentals and programming, but very little is done in terms of teaching I/O fundamentals. *Software-Defined Data Infrastructure Essentials* would make an outstanding textbook for a university computer science course on this topic.

2018 BOARD OF DIRECTORS

- Kevin Mobley, The Ian Thomas Group, LLC - Chair
- Anoush Najarian, MathWorks - Vice Chair
- Allan Zander, DataKenetics - Treasurer
- Nell Owens, Midwest ISO - Secretary
- John Baker, JCB Performance Solutions Inc.
- Rich Fronheiser, Syncsort
- Jonathan Gladstone, Bank of Montreal
- Steve Guendert, IBM
- Alex Podelko, Oracle
- Elisabeth Stahl, IBM
- Igor Trubin, Capital One

2017 CMG STAFF

- Amanda Hendley - President and Managing Director
- Michelle Cervantes - Operations and Events Manager
- Mary Hutson - Operations and Events Coordinator

ABOUT

CMG Journal is a quarterly members-only publication of Computer Measurement Group. Submissions to be considered for publication are accepted throughout the year at cmgjournal@cmg.org. For more information about CMG, the CMG Journal, or membership options, please visit CMG.ORG.

CMG
Computer Measurement Group

Computer Measurement Group
2720 Route 42, #121, Sicklerville, NJ 08081
856-401-1700 | cmghq@cmg.org