Best Practices for Improving Application Performance and Lowering Cost by Managing MIPS

EXECUTIVE SUMMARY

This paper will guide the reader through a suggested eight step methodology to obtain optimum results with MIPS Management. From performance objectives to illustrating return on investment, this paper will walk the user through the key steps necessary to get effective results with Compuware's solution for MIPS Management (MM).

Application throughput is a major priority for IT organizations. In some instances CIOs are compensated on application results; they have a keen interest in ensuring applications run smoothly. But having a need or desire and knowing how to satisfy that need or desire are two different things. This paper concentrates on two critical components of application throughput: **Fault Management (FM)**, ensuring that programs complete successfully; and **Application Performance Management (APM)**, ensuring that programs perform well. We will call this combined discipline **MIPS Management**.

One reason why organizations fail to adequately perform anything approaching MIPS Management is that no formal process exists. This paper will address that deficiency by providing a roadmap using Compuware's solution for MIPS Management.

There are two overwhelming reasons why IT organizations should want to practice MIPS Management. The first reason is to save money; the second is to increase customer satisfaction. Poor performance results in high costs. Among those costs are millions of dollars spent annually on unnecessary processing due to poorly tuned application software. Also included are costly hardware upgrades due to excessive CPU utilization, increased batch processing time and/or unacceptable response times. Whether applications perform well or poorly, program failures incur additional costs to restore, restart and rerun. Jim Schesvold has observed, "System abends in particular incur heavy overhead, but a variety of other abends, resource depletions or other error processing consume system resources, making them unavailable for where they're needed most" ("CICS Concepts in Action," *IBM Systems Magazine*, May/June 2007).

THE BENEFITS OF MIPS MANAGEMENT

MIPS Management can reduce the amount of CPU used by reducing the demand on system resources. You don't have to pay for the CPU cycles that are no longer used. Over time, this can result in tremendous savings. Reducing the failures that cost CPU cycles and elapsed time saves additional resources wasted restoring files and rerunning (or restarting) failed batch jobs or online transactions. Tuning applications properly and avoiding faults means lower CPU consumption. The result can be to delay or avoid a CPU upgrade.

This not only saves money in hardware costs, but it also eliminates the very costly software licensing fee increases that accompany such an upgrade.

While customer satisfaction has always been important, it is even more so in today's digital economy built on e-commerce and web-enabled applications. Customers accessing information via the Internet are not concerned about the technologies that serve their needs; they just want fast responses. A slow customer-facing application raises the risk of lost business opportunities. If a web interaction takes longer than eight seconds to deliver an answer, the typical end user will perceive the application as unavailable or unusable. Customers experiencing such slow response times will move quickly to other, more responsive sites, with a resulting loss in business.

THE MIPS MANAGEMENT CAPABILITY MATURITY MODEL

Some years ago the Software Engineering Institute at Carnegie Mellon University produced a model representing the various levels of maturity in the software development process. The Capability Maturity Model for Software (SW-CMM) identifies five levels of commitment to disciplined software development that can occur in an organization.

Compuware has devised a similar model to represent MIPS Management maturity, based on the discipline with which organizations continuously practice MIPS Management and the process they use to implement it. This model is illustrated in Figure 1.

Compuware

Just as a martial arts student only earns a black belt by passing through various levels of knowledge and skill, so too must an organization pass through various levels of techniques and processes to reach the highest level of MIPS Management maturity. The levels are described below.

Level 1—Reactive (Chaos)

Level 1 is entitled *reactive*, but could just as easily be labeled *chaos*. The organization is spending most of its time reacting to performance crises and production application failures as they occur. This ad hoc fire-fighting is costly in both time and effort, because key personnel are taken off their normal tasks and thrown into the crisis under extreme pressure to find and fix problems. Productivity decreases as stress levels increase.

Level 2—Repeatable (Proactive)

There are as many ways to write a program to solve a business requirement as there are programmers, but only one method will produce efficient code that uses the least amount of computer resources to get the work done. Quite often an application program consumes excessive CPU cycles or incurs unnecessary wait time in its processing and data access. In the vast majority of cases these inefficiencies can be removed, and usually quite easily.

Level 2 is called *repeatable*, and the basic characteristic of this level is being *proactive*. In a repeatable and systematic way, the analyst reduces excessive resource demands on the system, whether it be CPU cycles or unnecessary I/O activity. The analyst is proactive in not waiting for a crisis to develop, but instead intends to reclaim resources that would otherwise be used unnecessarily. This is an improvement over the previous level.

Informal studies done by users of performance tools have indicated that approximately 80 percent of modifications made to improve performance do not require major code changes. In "Performance Tuning Mainframe Applications Without Trying Too Hard," (CMG, 2002), Tony Shediak lists common performance problems, including:

- inefficient use of the programming language data types
- data conversions caused by mixing data types unnecessarily
- inefficient compiler options
- inefficient initialization of large structures/tables
- inadequate VSAM or QSAM buffering
- inefficient file block sizes.
- inefficient SQL statements

At this level, we highly recommended you initiate one or more MIPS Management projects with defined goals and objectives. Each project should be assigned to appropriate staff members with sufficient time and authority to ensure capture and correction of all application performance and failure issues. For more specific details, see the section entitled "An MM Project Roadmap."

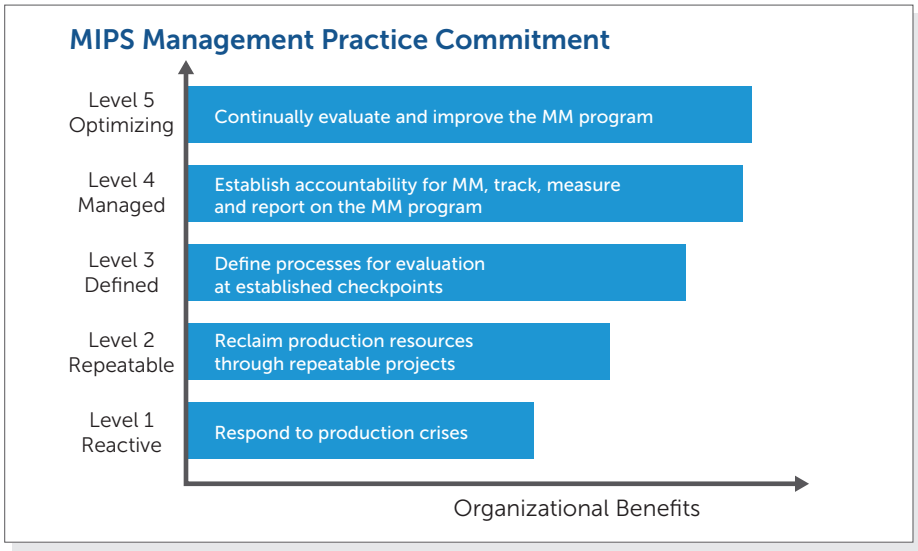

Figure 1: MIPS Management Process Maturity

Compuware

Level 3—Defined (Process-oriented)

The third level of maturity is called **defined**. At this level, a consistent, standard **process** exists for MIPS Management with established checkpoints throughout. For example, the move-to-production function may require providing proof of adequate performance before any code is put into production. In this way, operations personnel are assured no inefficient code is being migrated into the production environment that can impact existing performance thresholds. The idea here is to prevent poor performance from being introduced into the production environment, rather than just reacting to it when it happens. Other checkpoints to consider might include those of unit test, system test and even design.

Compuware iStrobe and AutoStrobe can help ensure "performance creep" does not impact current acceptable performance levels. The Auto Measure function will automatically measure any new or changed load modules. iStrobe provides the ability to compare one run to previous runs on the Jobstep performance report using past SMF data.

Level 4—Managed (Disciplined)

One challenge of performance management is that it often involves many silos within an organization, but no single person or team holds the ultimate responsibility.

A solution to the dilemma of responsibility can be found at Level 4, the **managed** level. Perhaps a better term for it would be **disciplined**. At this level of maturity, upper management has established accountability for MIPS Management throughout the organization, from development and test groups through production control and operations. Each key department involved in the application lifecycle should have an MM goal assigned to it. The goal should be well-documented and well-publicized. This accountability is recognized at the organizational and/or departmental level. Additionally, individual performance goals may be assigned to key personnel. For example, an individual application support analyst may be required to reduce total CPU cycles by 10 percent within a defined time period and also reduce production failures by the same amount for his or her area of responsibility.

It is extremely important that the process at this maturity level contain two elements: reporting results and rewarding efforts. If management is not made aware of the savings and improvements realized by the project, including all the money saved and all the customer satisfaction generated, support for the initiative will begin to fade away. This can result in a cancellation of the MIPS Management program with the resultant return to lower and less beneficial maturity levels.

Using Abend-AID Fault Analytics, analysts can observe the cost savings realized by eliminating failures in the worst-abending applications. They can also compare the frequency and resource cost of application failures in one time period compared to another, and report the cost savings to management.

Level 5—Optimizing (Kaizen)

Level 5, called **optimizing**, is the ultimate maturity level. It is equivalent to the Japanese concept of continuous improvement, known as **kaizen**.

A philosophy of ongoing improvement, kaizen involves every position from top manager to entry-level. The point of kaizen is to recognize that any task can be improved. In order to improve, one must first perceive the need for change, recognize the problems and finally work to solve them. It might be useful to challenge all employees to brainstorm how they might improve their current job process, and then take appropriate action to realize the better solution. Continuous improvement treats every variance from the target as a problem to be solved and everyone as a responsible contributor.

In other words, make MIPS Management part of the culture. A list of suggestions about how to do this includes, but is not limited to:

- promote the optimal use of the Compuware MIPS Management solution
- train developers and analysts on the latest and greatest MM techniques
- establish mentoring programs to ensure skills transfer and guidance
- sponsor regular technical updates about the performance aspects of languages, subsystems and vendor products
- review technical standards to ensure currency

To assist clients with this MM process, Compuware provides extensive Strobe and Abend-AID training.

BENEFITS OF THE MIPS MANAGEMENT CAPABILITY MATURITY MODEL LEVELS

Each MM maturity level has corresponding benefits, each one increasing the benefits realized at the levels below. These benefits have been summarized in Figure 2. This paper is intended to provide a path for an organization to ascend from its current maturity level to the highest.

Level 1—Reactive (Chaos)

Because all organizations start at Level 1 by default, the benefits are necessarily the smallest, but do exist nonetheless. Organizations can benefit from using Strobe for application performance problems and Abend-AID for application failures, on an as-needed basis. Reacting to production crises can help to eventually resolve the application performance and failure issues, but at a greater cost, since resources are already consumed.

Level 2—Repeatable (Proactive)

Moving one step up the maturity ladder, organizations reap greater benefits at Level 2 because there is a repeated and persistent effort to take back wasted resources consumed by tasks that complete successfully, as well as by those that fail.

At this level of maturity, some of the major benefits of MIPS Management come into play, such as deferring a CPU upgrade, reducing MSU during peak R4HA periods and increasing customer satisfaction. An upgrade deferral and reducing MSU avoids not only the hardware cost, but also the accompanying increase in software licensing fees. Furthermore, rather than expending scarce developer resources in testing applications on the new hardware, managers can deploy development staff to meet business requirements.

Level 3—Defined (Process-oriented)

At Level 3, greater benefits are realized because IT staff time is not used in fire-fighting efforts. This means the cost to maintain a production system is vastly reduced. By utilizing checkpoints throughout a well-defined process, the organization can prevent inefficient or faulty applications being deployed into the production environment. Savings at this level include the cost of staff time. With less time spent on fire-fighting application failures and performance crises, there is also an increase in staff productivity.

Level 4—Managed (Disciplined)

Because everyone is accountable for MIPS Management at Level 4, the benefits realized here include a reduction of costs to achieve and maintain application efficiency. Programmers have an incentive to be concerned about performance in design and unit-test environments, thus reducing the chances that a performance bug will make it further along the application lifecycle. A study done by Accenture found performance problems that are identified later in the lifecycle can be much more costly to fix. Inefficiencies introduced in design can cost twice as much to fix during development, four times more during system testing and eight times more when the application enters production.

Level 5—Optimized (Kaizen)

On Level 5, continuous, incremental improvements in the process itself will ensure MIPS Management remains an integral part of the IT organization. A solid plan is just the entry point to maintaining a competitive edge, especially in this era of web-enabled, customer-facing applications. The question that CIOs need to ask is: "What is the cost of not doing MIPS Management?"

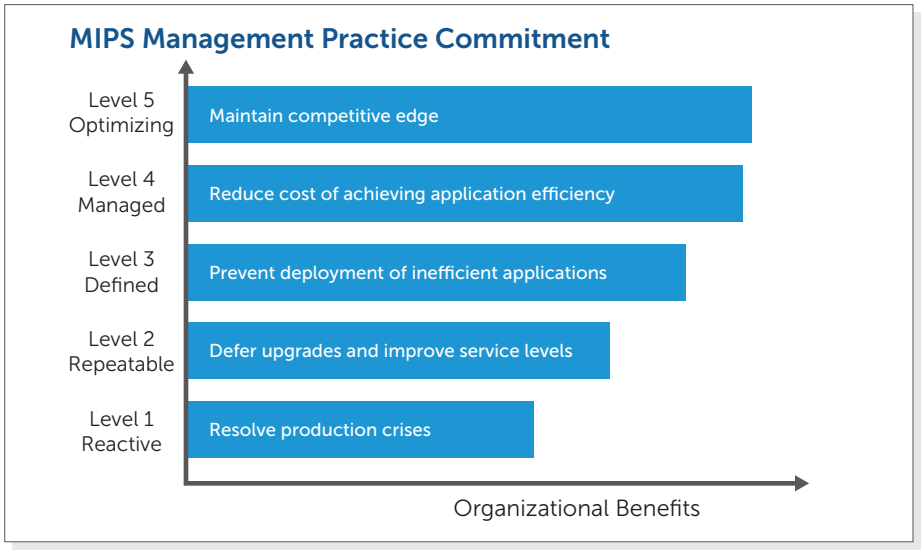

Figure 2: Benefits of MIPS Management

Compuware

AN MM PROJECT ROADMAP

What follows is a suggested MIPS Management project outline, with examples of tasks that need to be completed. Project managers can use this template as a starting point for their projects. This roadmap is intended to assist an organization in moving up the MM maturity model in the most efficient manner. It assumes at least some familiarity with the functions of Compuware products like Strobe, iStrobe, Abend-AID Fault Analytics and Abend-AID. For further assistance, please contact Compuware at **compuware.com**.

Step 1—Consider the Objectives

It is axiomatic that those who fail to plan, plan to fail. Any MIPS Management project is doomed to failure, or at least to realize less than optimum results, if the team does not define concrete, quantifiable objectives. MM objectives can include one or more of the following: (1) reducing CPU cycles consumed by the application, (2) reducing the elapsed time of the application and (3) reducing the frequency of application failure.

CPU reduction is usually of highest priority when the processor is running at "five nines" (99.999 percent full capacity), a CPU upgrade is imminent or MLC is consistently exceeding the monthly budget. Reducing CPU cycles can delay such a costly upgrade in the near to medium term, saving the organization thousands of dollars in hardware and software costs. On the other hand, if batch window constraint is the issue, then the focus would be on reducing elapsed time, accomplished mainly through reducing I/O by various means. In either case, reducing the frequency and number of application failures will increase the return on the investment in data processing resources. Identifying the leading cause of the high MLC and reducing its impact can have an immediate savings.

In many cases, all three objectives are desirable. It is of the utmost importance, however, that the objectives be quantifiable, specific and well-documented. Then there will be no question whether the project met its goals. The best place to record the objectives is in a project plan. An example of a specific goal might be to reduce the CPU consumption of Program X by 10 percent, decrease the monthly peak rolling four hour average MSUs by 10 percent, improve the elapsed time of Job ABC by 15 minutes or reduce the number of failures by 20 percent next month.

Step 2—Define the Project Scope

At first glance, a MIPS Management project can appear to be overwhelming. A large mainframe installation may run hundreds of definable applications, comprised of thousands of programs across several platforms. Unless a project manager can focus on what matters most, he or she is subject to the dreaded "paralysis by analysis" syndrome. To narrow down the relevant details needed to avoid this syndrome, Step 2 involves gathering easy to understand information using Abend-AID Fault Analytics, AutoStrobe and Strobe Insight.

With Strobe Insight Reports in iStrobe providing advanced analytics for the SMF 30 and 70 records, you can quickly and automatically identify the high CPU consumers in your applications. Out of the box reports will show jobs that contribute the most to the peak MSUs, jobs that use the most CPU and jobs that have the longest elapsed times.

A similar process is available for CICS transactions that consume excessive CPU or have unacceptable response times. AutoStrobe CICS Global Monitoring can build a candidate list of troublesome transactions that should be measured.

AutoStrobe Batch Global Monitoring will optionally monitor all or a subset of batch job steps requested and calculate a moving average for both CPU consumption and elapsed time. If either of these thresholds is breached during future processing, Strobe will initiate a measurement immediately to ensure that all performance metrics are captured in a timely fashion.

Abend-AID Fault Analytics provides statistics about failing tasks, including CICS and IMS transactions as well as batch job steps. For CICS transactions and job steps, the reports include the CPU seconds and elapsed time consumed prior to the failure. Using this information, an analyst can see which tasks fail most frequently, and which failing tasks consume the most resources. In this step, you will want to take an initial look at the information from Fault Analytics and then decide how to focus the view of the information relevant to your objective.

Here are some of the ways you can focus the data in Fault Analytics. You can look at:

- highest CPU consumers and/or longest runtimes
- certain applications (payroll, order entry, etc.)
- certain environments (CICS, IMS, batch)
- certain LPARs
- certain times of the day
- production only

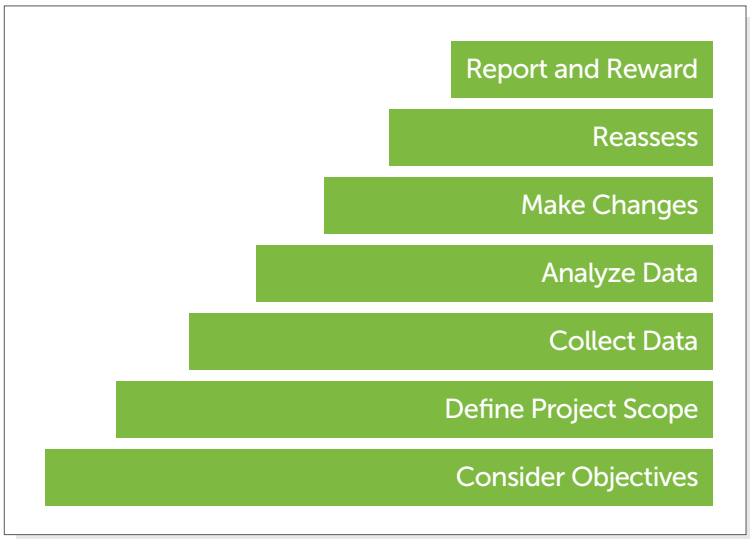

Figure 3: The Seven Steps to Highly Effective MIPS Management

Step 3—Collect the Data

This step involves customizing and reviewing Fault Analytics reports and executing Strobe measurements.

Using Fault Analytics, you can quickly identify tasks for a "hit list" of the failing tasks that consumed the most CPU seconds or had the longest elapsed times. These should be included among the first tasks that you measure using Strobe, because they are costing you the most in failures by incurring costs in resource consumption and in recovery and restart CPU usage. Using Strobe, you can measure the tasks in the candidate list.

AutoStrobe automates the measurement process. The automation built into AutoStrobe allows users to minimize the time needed to proactively manage application performance and increase the efficiency of their MIPS Management process. Analysts can create reusable groups of measurement requests and submit them with a single action. Analysts can also schedule individual measurements or groups by day, date or time of day, enabling them to easily gather application performance data at regular intervals during peak processing times.

Step 4—Analyze the data

In this step, you will use the data gathered in the previous step to determine the changes needed to reduce the resources consumed by your applications. You will analyze Fault Analytics reports, Abend-AID diagnostic reports and iStrobe performance profiles.

As you analyze the Fault Analytics reports created in the previous step, you will find your high-consuming tasks fall into one of two categories:

1. Tasks that were canceled or timed out because they ran long.
2. Tasks that failed due to an error.

For tasks in the second category, Abend-AID provides detailed diagnostic information with recommended resolutions for batch and CICS abends, as well as errors in IMS, DB2 SQL and Websphere/MQ environments.

In the previous step, you ran Strobe measurements that show an individual address space and any program and/or transactions executing in that address space for the duration of the measurement. The measurement process gathers thousands of samples that are then used to create a series of hierarchical reports indicating where and how the application program spent its time. This series of reports is called a performance profile. By reviewing the performance profile, an analyst can quickly identify opportunities to make improvements.

Such a detailed profile can appear daunting, especially to new users. To address this concern, Compuware created iStrobe, a browser-based navigation tool designed to assist in profile navigation. For users of DB2, Strobe offers an SQL analysis facility that will provide more details on how to improve individual SQL performance, via built-in hints and tips. The SQL Analysis Feature works in conjunction with Strobe and iStrobe to supply access path analysis and database and SQL coding recommendations for DB2 applications measured by Strobe. Analysts using Strobe Insight Analytics in iStrobe can quickly identify items consuming the most CPU in those profiles that will offer the biggest returns from tuning.

Step 5—Make Changes

As Albert Einstein observed, insanity is doing the same thing over and over again and expecting different results. In other words, if you continue to do the things you've been doing, you'll continue to get the results you've been getting. The only way to improve application performance and reduce faults is to make changes that will reduce the demands made by the application on the system, and prevent faults from recurring.

Resolving failures in your high-consuming tasks should be your first priority, since these applications waste CPU and I/O resources, but also must be recovered and restarted. In the previous steps, you used Abend-AID Fault Analytics to identify these tasks and analyze the errors with Abend-AID. In this step, you will use that information to take action to prevent the errors from recurring.

By tuning high-consuming and failing tasks first, IT organizations realize a two-fold benefit. First, the application no longer fails, avoiding the costs associated with recovery and rerun. Second, CPU consumption is reduced by removing inefficiencies found using iStrobe reports.

Experience has shown that there are four levels of difficulty in application improvements, and approximately 80 percent of all changes fall into the first three.

The first level includes file buffering alterations, block size changes and other similar alterations that can usually be accomplished directly in the JCL. Such changes result in major improvements, but require no testing, as no application logic has been modified. These purely environmental corrections result in impressive improvements. Enhancements of this nature have been known to reduce CPU consumption by 86 percent and elapsed time by 95 percent (Shediak, op. cit).

The second type of changes include compiler parameter changes, which modify the characteristics of the resultant object and/or load modules. Refer to the vendor's documentation to get an understanding of how each parameter will affect performance. There are also conference proceedings from organizations such as CMG and SHARE that will assist in tuning compiler options. Note that in most cases, these changes do not need extensive testing since, once again, no application logic was directly modified. Performance can be vastly improved with this type of change; in fact, CPU has been reduced by up to 70 percent in such instances *(ibid)*.

The third type of alteration, involving minor code changes, includes eliminating inefficient use of programming language data types, removing data conversions caused by mixing data types unnecessarily and inefficient initialization of large structures or tables. Moving the invocation of a built-in function such as Date/Time outside of a loop will also reduce demand on system resources. Testing is desirable after completing changes such as these, but because the change is minor in nature, extensive testing is not required. Again, performance can improve greatly; CPU reduction in the range of 65 percent has been documented using these methods *(ibid)*.

The final type of alteration requires a major code change. Note that this kind of change is rarely required, and a total rewrite of the application modules is not required. Of course, a cost-benefit analysis should be completed before continuing with a major code rewrite to ensure adequate return on investment.

Think of looking for performance opportunities as peeling an onion. Although there may be many layers of opportunity to improve performance, only one particular opportunity will be topmost at any one time. This is the top layer of the onion. Make modifications to fix this one, and only one opportunity, before moving on to the next step. The reason for this is simple. Just as in problem resolution you don't try to make more than one change, so too in performance analysis. Too many changes may adversely impact each other, confusing the issue in subsequent measurements. The recommendation is to pick the best opportunity and make one change to improve the performance at a time. Then move on to Step 6.

Step 6—Reassess

To determine what impact a change has made on the application's performance, the analyst needs to re-measure the job step or online region. This step is identical to Step 3 described above. The automated scheduling and grouping functions of AutoStrobe are very useful in this step, because this process is very repetitive.

Using Abend-AID Fault Analytics, you can examine the same customized reports and compare the CPU time lost to faults between time periods and observe progress toward your goal.

Step 7—Report and Reward

Note that Steps 4 through 6 are iterative and should be done as often as necessary until the objectives are reached for that particular application or suite of programs. Step 7, on the other hand, can and should be done continually throughout the project as appropriate. The reasons for this are elaborated below.

Two major obstacles to the successful completion of a project are often overlooked simply because they are not technical in nature. The first is failure to report results accurately and in a timely fashion. Project sponsors (i.e., the people who provide the needed financial and personnel resources) want to see a quantifiable summary of progress. In such a case, no news is not good news. When management does not receive continuous updates, a project is very likely to fail through a subsequent lack of support. Always be ready to answer the questions "What have you done for me lately?" or "How is it going with that MIPS Management project?"

The other obstacle to successful completion of any project is poor morale among the participants, usually caused by a perceived lack of appreciation. If the people involved in the project do not feel valued and appreciated, interest, effort and overall morale will eventually decline.

iStrobe includes a performance database allowing analysts to report many metrics captured during previous measurements. Using a graphical interface, analysts can create specific trend and summary reports for all important tuning components to identify hot spots before they become a problem and to show performance improvements over time. The analyst can create graphs and charts, which could be used as a performance dashboard or in a management report to validate the tuning efforts and highlight future objectives.

With AutoStrobe, an analyst can collect, display and manage measurement session history information. This information enables the analyst to more easily identify trends in the performance of a particular job step, DBRM or transaction in an online region and to quantify the results of the MM program. Collecting measurement session history for the job steps, transactions and DBRMs specifically targeted for performance improvement enables the team to focus its efforts.

For example, it may be already known that the overall performance of the payroll application has improved because the CPU time and elapsed time have decreased. However, by comparing the most recent measurement session with the measurements of the job prior to implementing the coding improvements, the analyst can calculate exactly how much performance has improved. From the COST COMPARISON ISPF panel, the analyst can quickly see changes in the cost per run and the cost per year (total annualized savings) for instances of the selected job step, as compared to the baseline measurement. History records can be recorded for job steps, transactions and DBRMs.

Managers can use Abend-AID Fault Analytics to track the progress of a MIPS Management project, showing the frequency and cost of application failures throughout the project lifecycle and beyond. Fault Analytics archives the information, so it is always available for historical comparison.

Step 8—Establish Continual MIPS Management Processes

When the initial MM goals have been reached, the challenge will be to establish standards and processes to integrate MIPS Management into the application lifecycle, to ensure that resources dedicated to application processing are used efficiently in the future. There are several measures to consider:

1. Run Strobe measurements against tests at the unit, integration, regression and QA stages of testing.
2. Use iStrobe to analyze the performance profiles and make appropriate changes.

As mentioned earlier, AutoStrobe can monitor all batch job steps requested and calculate a moving average for both CPU consumption and elapsed time. If either of these thresholds is breached during future processing, Strobe will initiate a measurement immediately to ensure that all performance metrics are captured in a timely fashion. This helps to automatically prevent performance creep, saving time and money and allows analysts to move on to more important tasks.

SUMMARY

This paper has attempted to provide a clear way of thinking about MIPS Management (involving both performance and fault management), why it is important and how to go about it. It has shown how an IT organization can progress through the MM maturity model, moving from chaos to kaizen. It has offered ideas and suggestions to assist in the evolution of an MM roadmap for any organization willing to take the time and effort to realize the tremendous benefits from such a plan.

As a wise man once said, there is no silver bullet. This methodology is offered as one more item in an organization's toolbox so, in the arena of MIPS Management, the organization can move forward to better results.