

Total Performance Management

An introduction to the method.

ORACLE

TOTAL PERFORMANCE MANAGEMENT

Craig A. Shallahamer
Oracle Consulting Core Technologies
Portland, Oregon USA
voice 503/220-1678
Email cshallah@us.oracle.com

Revision 2.71 February 6, 1995

1994 OAUG Paper #49 (V1.0)
Oracle Magazine May/June and July/August 1995 (V2.69)

Abstract

Success brings increased responsibility. As management expectations upon the database administrator (DBA) increase,

1. Contents

1. CONTENTS	4
2. LIST OF FIGURES.....	5
3. INTRODUCTION.....	6
4. PERFORMANCE MANAGEMENT	7
5. HOLISTIC PROBLEM ISOLATION METHOD	9
6. STAGE 1: AUDIT.....	11
7. STAGE 2: TUNING.....	13
8. STAGE 3: PERFORMANCE MANAGEMENT	15
8.1. THE MECHANICS.....	16
8.2. DEVELOPING A PERFORMANCE MANAGEMENT STRATEGY.....	17
9. REFERENCES	21
10. ACKNOWLEDGMENTS.....	22
11. ABOUT THE AUTHOR.....	22

2. List of Figures

FIGURE 1: AS DATABASE SIZE AND COMPLEXITY INCREASES, SO DOES THE IMPORTANCE OF MANAGING PERFORMANCE.7

FIGURE 2: TOTAL PERFORMANCE MANAGEMENT IS A LIVING HOLISTIC METHODOLOGY THAT CONTINUALLY CYCLES THROUGH THREE STAGES: THE AUDIT STAGE, THE TUNING STAGE, AND THE PERFORMANCE MANAGEMENT STAGE.8

FIGURE 3: A HOLISTIC PROBLEM ISOLATION METHOD WILL INVESTIGATE FROM THREE ANGLES; OPERATING SYSTEM, APPLICATION SYSTEM, AND THE ORACLE SYSTEM. INVESTIGATING A PROBLEM FROM DIFFERENT VIEWPOINTS REDUCES ERROR AND INCREASES ACCURACY.10

FIGURE 4: A PERFORMANCE MANAGEMENT STRATEGY EXCERPT.20

3. Introduction

Since 1990, the Oracle Core Technologies Team¹ has visited hundreds of sites performing a wide variety of performance related engagements. In 1990, Cary Millsap developed a configuration standard for architecting complex industrial strength database systems. First presented at the 1990 IOUG conference, the Optimal Flexible Architecture [3], or OFA for short, has become a world-wide standard. As system performance and availability radically improved, so did the number and complexity of applications.

Success brings increased responsibility. As management expectations upon the database administrator (DBA) increased, the need to maintain performance, with every increasing application complexity and increasing data volume, has placed many DBAs in a frenzy. As a result, many DBAs have not taken a step back and understood the changes that are beginning to occur.

Rather than *managing performance*, overall tuning has continued at the micro level. In a desperate rush to maintain performance, DBAs are now rushing to purchase system management tools. As a result, we now have a mishmash of system management tools, tuning tools, and techniques with no holistic method to manage performance.

Total Performance Management (TPM) is a *living holistic methodology*. Total Performance Management is *not* a top-down or a bottom-up approach to maximize performance. It is holistic because problems are attacked from multiple fronts. It is a method because it provides a structured approach to synthesize the use of tools (system management tools, O/S tools, Oracle tools, Application tools) and techniques in a dynamic computing environment. It is living because its programs evolve to meet changing requirements and are usually run selectively and purposely—not routinely. The programs make a variety of administrative and performance related alterations and alert others to perform similar tasks. This paper concentrates on the method not on the evolving programs.

¹The Oracle Core Technologies Team has a long and rich history. In 1989 the National Technical Response Team (NTRT) was founded by Mr. Robert Rudzki and Mr. Cary Millsap. In 1992 the team was renamed the National Product Specialist Team (NPST) and in 1993 the Core Technologies Team (Core Team).

4. Performance Management

Performance management empowers the performance specialist to distance himself from an environment characterized by explosive surprises into an environment characterized by early problem detection and automated resolution. More simply, a site that manages performance has successfully transitioned from a reactive environment into a proactive environment.

As open systems increase in complexity so must their management tools. While many system management tools are gaining popularity, simply installing a tool is equivalent to simply tuning a database. Tuning and system management are pieces of the much larger performance management task.

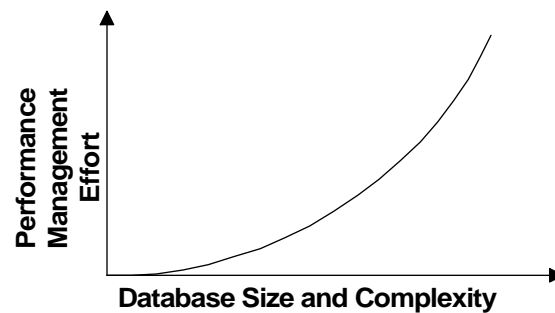


Figure 1: As database size and complexity increases, so does the importance of managing performance.

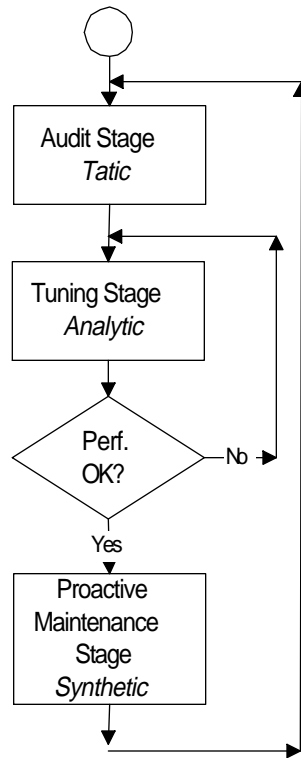


Figure 2: Total Performance Management is a living holistic methodology that continually cycles through three stages: the audit stage, the tuning stage, and the proactive maintenance stage.

Performance management is a process. Most sites are nowhere near managing performance. However, operating in a reactive environment can be optimal. Figure 1 shows that as a database(s) grows and application complexity increases, so must the performance management effort. For example, suppose a 200 megabyte application is servicing ten users. If a problem arises, it may not be an issue if users alert you of a problem. However, the performance manager of multiple database systems servicing hundreds of users cannot allow performance to degrade and cannot allow users to be the first to detect potential problems.

When down time is not scheduled and problems are not anticipated, the time to “do the job right” may not be available. In contrast, a site which seeks out potential problems and schedules administration down time tends to solve small problems before they turn into big problems.

The TPM transitions a system through three distinct yet dependent stages (Figure 2). Each stage has very specific objectives with tactics to meet those objectives. This paper will address each stage at a high level.

The first stage is the *audit* stage. During the audit stage, the problem is defined, the system is quickly observed, changes are possibly made, and the overall project is scoped. The second stage is the *tuning* stage. During the tuning stage, the problem is resolved by performing a detailed analysis and implementing a variety of recommendations. The third stage is the *proactive maintenance* stage. This is when the performance game transitions from a reactive mode into a proactive mode. Tools are put into place to detect potential problems, alert responsible parties, and automatically resolve most issues.

As mentioned above, managing performance is a process that takes time, careful planning, technical expertise, and powerful tools.

5. Holistic Problem Isolation Method

Each TPM stage uses a simple yet very powerful approach when isolating performance bottlenecks. To optimize a database system one must understand each subsystem: the database, the operating system, and the application. By discovering where each system is being stressed or bottlenecked, observing the overlap, and re-focusing the search, the problem possibilities can be narrowed very quickly.

Unfortunately, most performance specialists tend to focus on one of the three main subsystems mentioned above. This results in an ill-defined problem that will translate into a lop sided solution. In many cases, the solution, while appearing to solve the problem from one subsystem’s viewpoint, actually compounds the problem from other viewpoints resulting in overall system performance degradation.

Each subsystem’s components need to be investigated and any bottleneck contributors must be “placed on the table.” Picture your system as if it were a card game, and each subsystem were a player with a pile of chips. The database system player holds the “datablock buffer size” chips and the operating system player holds some of the I/O chips. Whenever a possible bottleneck contributor is found, a chip is thrown into either the CPU, I/O, memory, or network pile. When the game is over (the investigation complete), not only is the bottleneck identified but a number of other issues will have been identified.

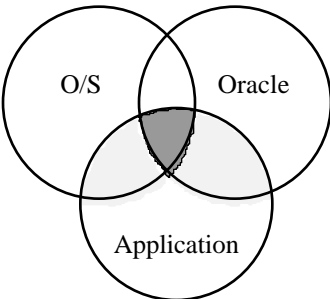


Figure 3: A holistic problem isolation method will investigate from three angles; operating system, application system, and

6. Stage 1: Audit

The first TPM stage is the audit stage. The audit stage theme is centered around defining the problem and establishing a foundation upon which to build. The key audit stage points are:

- Overall problem scoping
- High level problem detection
- Holistic problem isolation
- Two to four days effort
- Two to four page report (varies greatly)

Limiting problem detection to a high level restrains the innate desire of a technician to dig into the details. This allows the performance specialist to maintain a visionary view of the entire situation. Working in areas you are comfortable and diving into the details is fun and yields quick results. However, by avoiding project scoping and not applying a holistic problem isolation method, a complete and long term solution will most likely not be implemented. If scoping is skipped, resources necessary to complete the tuning and proactive maintenance stages may not be available.

The audit stage consists of multiple short repetitive processes that merely skim the surface looking for indications of iceberg problems. For example, running the UNIX command **vmstat** every thirty seconds during problem time is one way to detect the operating system bottleneck. A benefit of a **vmstat** like command is it dips into each operating system subsystem, it is repeatable, it presents concise information, and it places minimal strain on the operating system. Another **vmstat** benefit is its command structure is very simple yet powerful.

The audit stage must begin scoping the entire performance problem. If scoping is not properly done, this may be an indication a high level approach has taken a back seat to a more detailed approach. While some key issues will be discovered, producing in quick results, this approach may bypass long term problems that may reoccur. Consider row fragmentation. One can quickly determine if a database system is suffering from row fragmentation.[7] A neophyte solution is to rebuild the database thus eliminating the problem. Unfortunately the problem will most likely return because the quick solution treated the symptom, not the problem.

Scoping includes establishing metrics such as benchmarks. “Before and after” metrics are extremely powerful for documenting a project’s success. Without metrics a project’s success is left entirely up to how people feel a particular day. If users are still not pleased with the system yet the metrics indicate otherwise,

meet with users and possibly change the benchmark. Remember, impressive metrics mean very little to disgruntled users.

Scoping includes establishing project goals and estimating resources to successfully complete the project. Goals not only include performance metrics but documentation describing each problem and how it was eradicated. There are many different types of resources that may be required to complete the project. Resources do not only include labor but hardware and software tools. To tackle a specific problem a specific tool may be required. This tool may need to be purchased or may need to be developed. The general theme in scoping is to ensure everyone knows what you require to do your job, what your job will cover, and when the job will be completed.

Delivering the audit stage summary report may be cumbersome but it is an important deliverable. Both you and your colleagues will need the summary for future reference. The audit stage summary report not only communicates your work to your peers and to your management but it is an important reference document to be used during the tuning and the proactive maintenance stage.

The audit stage report should be very brief and direct. Leave the details for the tuning stage. Shown below is an outline which has proven successful in the field. The audit stage report typically combines the configuration investigation and performance investigation sub-sections. A typical audit stage report may range from two to four pages. Whereas a tuning stage report may range from five to fifty pages. The tuning stage report is where, using this same outline below, each claim is supported by a detailed explanation followed by supporting documentation.

- **Summary.** The *summary* section should, from a high-level, summarize your objectives, your accomplishments, and your overall comments and conclusions based upon the various sections.
- **Structure.** If your report is more than a few pages, the *structure* section serves as a report outline or table of contents.
- **Objectives.** The *objectives* section tell others what this report is trying to accomplish.
- **Scope.** Use the *scope* section to define and limit report contents.
- **Configuration Investigation Summary.** The *configuration investigation summary* quickly summarizes the entire configuration investigation and summarizes the results of each configuration investigation sub-section.
 - **Operating System Configuration Investigation.** The *operating system configuration investigation* section should explain if the O/S has sufficient horse power, has been built for ease of performance optimization, administrative ease, and production growth.

- **Database Configuration Investigation.** The *database configuration investigation* provides details regarding database structural issues such as segment and storage management.
- **Application Configuration Investigation.** The *application configuration investigation* section provides details regarding the structural integrity (E.g., Are all files present and properly located?), performance strategies (E.g., index strategy), object correctness (E.g., Do all procedures compile error free?), and source code control.
- **Performance Investigation Summary.** The *performance investigation summary* section quickly summarizes the entire performance investigation and summarizes where each sub-system is being stressed or bottlenecked.
 - **Operating System Performance Investigation.** The *operating system performance investigation* summarizes the O/S bottleneck and then supplies supporting documentation by detailing each operating system sub-system (memory, cpu, I/O, network).
 - **Database Performance Investigation.** The *database performance investigation* summarizes where the Oracle system is being stressed or bottlenecked and then supplies supporting documentation by detailing each database sub-system (process, memory, I/O, storage management).
 - **Application Performance Investigation.** The *application performance investigation* first summarizes the application resource intensive areas or processes, their affect upon performance, and then supplies documentation to support this conclusion.
- **Reference Material.** It may be appropriate to suggest *reference materials* which further explain and support your work, and to better educate your audience.

7. Stage 2: Tuning

The second TPM stage is the tuning stage. The tuning stage theme is centered around resolving performance problems. The key tuning stage points are:

- Further define project scope
- Holistic problem isolation
- Intense tuning from all angles
- Develop, implement, and document solutions
- Three to five day effort iterations
- Five to fifty page report (varies greatly)

During the tuning stage, a holistic problem isolation method is used to further define the project scope, the performance problems, and their solutions. Now is the time to dig into the details and perform a detailed analysis of the problems

discovered during the investigation stage and any new problems discovered during the tuning stage. While the proactive maintenance stage is to follow, when the tuning stage is complete, users should be very happy with performance and to their knowledge, the “tuning” project is complete.

As Figure 2 shows, the tuning stage is an iterative process that continues until users are satisfied with the system’s performance. It is important for everyone involved to understand tuning is typically not finding the silver bullet and then shooting it. Tuning involves gathering information, performing analysis, making changes, and repeating this process. Depending upon problem severity, system response, and access to the system, the tuning stage can take days to weeks to complete.

Portions of the tuning stage are dependent upon a realistic production load. When determining if redo log allocation latch contention exists, the system needs to be under a realistic load. As with most contention related issues, as the system load nears production levels, so does the accuracy of the information gathered.

Tuning application SQL is not dependent upon a realistic production load and is best performed on a different database. A SQL statement will access the same number of database blocks regardless of the system load. In fact, running production DML statements on a production database may be disastrous. Also, running queries on a production system may adversely affect production system performance.

The system startup/shutdown cycle is a nuisance to users but so is unacceptable performance. Tuning during a realistic system load probably means production availability requirements, which may prohibit an occasional startup/shutdown cycle. Therefore, a balance must be maintained. A performance specialist must respect and delicately work with a production database. And users must realize that unless the performance specialist is allowed to do their job, the system will always be “slow.”

As mentioned above, solid metrics are especially important during the tuning stage. Whatever tools and tool sets are used one theme must persist; the results must be reliable, accurate to a known degree, understandable, and gathered within an acceptable time period. Data that are technically impressive to look at may be useless if no one knows what the numbers really mean.

Running the command **sar -A 5 2400** vividly illustrates how very detailed and very accurate data can produce useless information. Every five seconds the above **sar** command will spew forth a single 300 plus character line detailing operating system activity. The results are very detailed (2400 lines) and very accurate but frustrating to read and nearly impossible to analyze.

There are a number of tools available both commercially and publicly that empower the performance specialist. Oracle's Core Technologies team uses a combination of tools. For example, the Administration and Performance Suite (APS) tool kit [5], the Performance Monitoring (PM) tool kit [6], the Oracle Metrics tool, Oracle7 for windows, Microsoft Excel, and Microsoft Word are all used to aid in delivering a complete solution. BMC Software (Patrol) and Ecosystems offer superb system management tools (SMT). While SMTs do provide information valuable during the tuning stage, they are more appropriately implemented during the proactive maintenance stage. A quick glance through Oracle Magazine will reveal a number of performance and SMT related tools.

A thorough understanding of each system under investigation helps the performance specialist to use the best tool. Learn as much as you can about each system and its subsystems. For example, if you are investigating Oracle's shared pool, understand its components and use a tool which digs deep into the heart of the shared pool. A rash conclusion regarding an "SGA cache hit ratio" may lead one into making incorrect decisions, possibly resulting in operating system paging.

To mark tuning stage completion, an audit stage type report should be produced. The structure is similar, but the contents is more defined, more granular, and very scientific.

8. Stage 3: Proactive Maintenance

The central proactive maintenance stage theme is to actively prohibit problems from re-occurring. Once users are happy with performance, it is time to transition into the proactive maintenance stage. Stage three is very difficult for many because it is relatively new and most people do not fully understand performance management. In addition, DBAs are being bombarded with system management tool options.

During the past few years it has been painfully obvious that true performance management has not been a priority. As stated in the introduction, this has not been a problem because relational database systems were still in their toddler years, relatively simple, and contained limited data volume. However, this is no longer true.

System management tools (SMTs) are not necessarily performance management tools. Currently, a SMT leans towards availability management rather than towards performance management. Also, a SMT's probing functions are more static rather dynamic or even evolutionary. For example, a SMT's flashing red box may indicate a disk crash or a network failure. In contrast, a performance

management tool's flashing red box may indicate a possible I/O bottleneck or a slow network response is being investigated and automatically resolved. A basic evolving program will determine the best method of problem resolution based upon past experience ("life times" of experience) and the current situation.

These evolutionary tools are being built based upon research performed in a new field called Artificial Life (Alife). These "living" programs exhibit life like characteristic such as eating (consuming energy), being born, dying, improving themselves, and producing offspring. These programs adapt, change, and evolve over their lifetime and over many generations to better perform their job.

In the future, performance management tools will be developed that alter themselves and dynamically alter perhaps Oracle segment characteristics in an efficient and non obtrusive way. Subsequent papers will be presented regarding living tools and their relation to performance management.

8.1. *The Mechanics*

Stage three takes the most time to fully implement because tools must be put into place which automatically detect, alert, and fix a potentially harmful problem. Traditionally, DBAs periodically check to ensure no Oracle segments are approaching the maximum number of extents. When the DBA discovers a potentially harmful situation, the extent's storage parameters are altered or the object is rebuilt. A better solution would be to have a performance management tool alert the DBA of the potential problem and automatically adjust the segment's storage parameters.

There are thousands of potential problems. However, by understanding your systems, how they are used, and understanding their strengths and weaknesses, generalizations can be made which reduce the items that must be periodically checked. Consider the UNIX operating system. By using basic commands such as **sar**, **vmstat**, and **netstat**, each subsystem (I/O, memory, CPU, network) can be monitored.

Many of the tools used during the audit and tuning stage may be reused. While the actual code may have to be altered, the algorithm, the methods, and the underlying data may interface nicely into a SMT. There are two basic SMT tool classes. The *snap shot* type tool produces results based upon what is occurring at a particular moment. In contrast, *time span* tools produce information based upon a selected time interval. Both tool types can be very effective.

Smoothing out peaks and valleys is the strength of time span tools. When I monitor daily activity, I typically do not want hundreds of detailed **vmstat** lines.

Glancing at many lines, one naturally focuses on the unusual activity not the general activity. An hourly time span is more useful because I can spot hours of heavy activity and then quickly begin to discover the operating system bottleneck.

Snapshot tools are perfect when checking if a particular event has occurred. Unfortunately, rapidly firing snapshots can adversely affect system performance. In addition, snapshot data is often misinterpreted. For example, if the number of active Oracle shared servers is snapped each hour, one may be inclined to believe the snapped number represents the average number of active shared servers. This is incorrect. A snapshot represents a specific event at a specific point in time.

Whether a commercial SMT or a SMT you developed is being used, interfacing your tools is typically the same. SMTs prefer their input to be generated from a functional model. A function like tool accepts parameters and outputs information in one or more columns on a single line. The UNIX command **vmstat** is a prime example of a single line multiple column, time span tool.

Care should be taken not to allow the SMT to influence the system it is monitoring. Wever Heisenberg in 1926 made a very simple but profound statement regarding investigations. Briefly, he said the granularity of an investigation is proportional to the effect the investigation places on what is being investigated. By minimizing the frequency of snapshot tool execution, the effect a tool places on a system is minimized. Reducing the run frequency of tools investigating historically stable items will also decrease the SMT effect. For example, if Oracle's datablock buffer cache hit ratio is consistently 0.90 to 0.95, it would be wise to reduce run frequency. However, a balance must be maintained between gathering the required information and minimizing the monitoring effect.

A simple method to check if your SMT is influencing your system is to observe the system (e.g., by running **vmstat** or **sar**) during a steady state period while the SMT is monitoring and while the SMT is not monitoring. Comparing the information will show the SMT performance impact.

8.2. Developing a Performance Management Strategy

Developing a performance management strategy has historically been one of the most difficult tasks for the information technologist. I suspect this is due in part from our training and experience of digging deep into the details. It is difficult to focus on the vision when you are constantly staring at details.

When developing a basic strategy, solicit input from a variety of individuals. Involving management, IS personnel, and the user community will help ensure

everyone understands what is deemed acceptable performance. Constantly ask yourself, “Is this issue important to my users?”

A performance management strategy (Figure 4) is composed of a one to many, multilevel hierarchy consisting of a mission statement, goals, objectives, and tactics. A few goals should be developed which when completed will signal a successful mission. Objectives are a further breakdown of the goals and tactics are the measurable events that can be checked off when completed.

Below are a few guidelines to help prepare your performance management strategy.

- **Mission Statement**
 - is visionary
 - is not specific
- **Goals**
 - are short and action oriented
 - lead one to naturally think, “How can this be accomplished?”
 - Minimize the number of goals
- **Objectives**
 - are higher level action items
 - can usually be check marked when completed
 - must be complete, to ensure the goal will be completed when all the objectives are met
- **Tactics**
 - are usually time based action items
 - are easily assigned to an individual
 - can be check marked when completed
 - are step-by-step instructions to ensure a specific objective will be completed

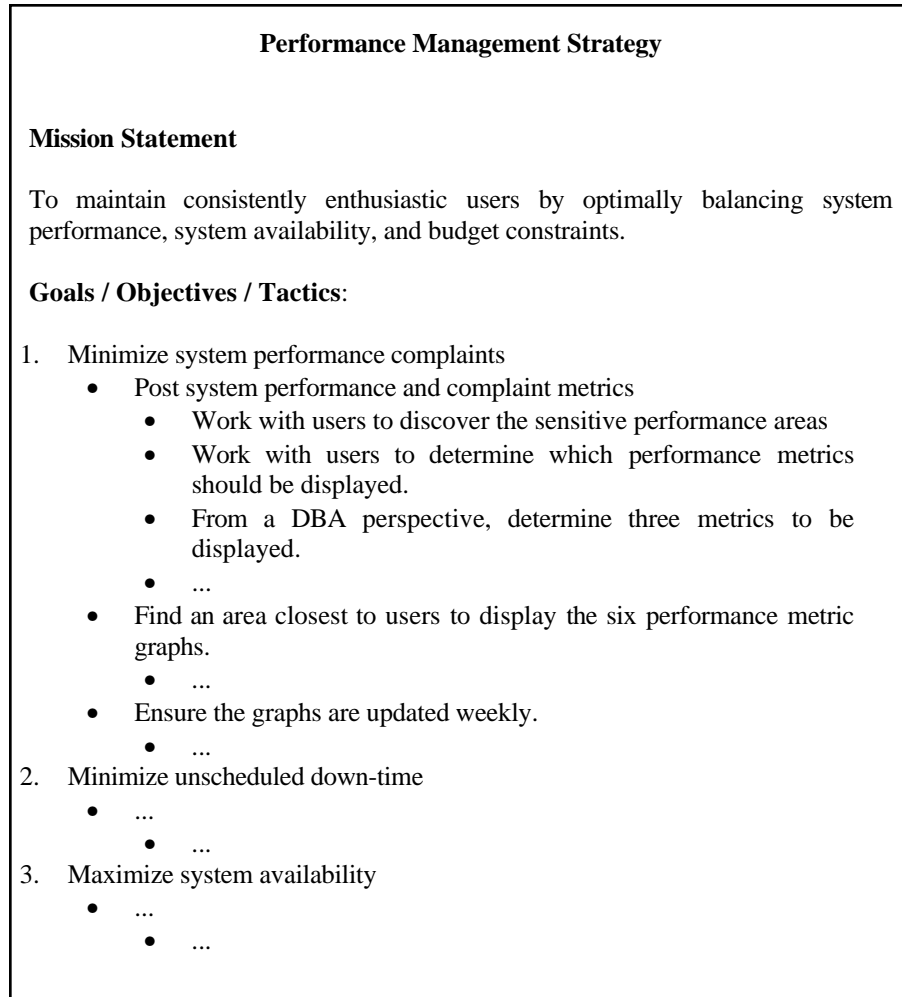


Figure 4: A performance management strategy excerpt.

9. References

1. Millsap, Cary V., Shallahamer, Craig A., Adler, Micah (1993). Predicting

10. Acknowledgments

This work was supported by Oracle Corporation and was conducted while the