Assessment of the Effect of Memory Page Retirement on System RAS Against Hardware Faults

Dong Tang, Peter Carruthers, Zuheir Totari, Michael W. Shapiro
Sun Microsystems, Inc.
dong.tang{peter.carruthers, zuheir.totari, michael.shapiro}@sun.com

Abstract

The Solaris 10 Operating System includes a number of new features for predictive self-healing. One such feature is the ability of the Fault Management software to diagnose memory errors and drive automatic memory page retirement (MPR), intended to reduce the negative impact of permanent memory faults that generate either correctable or uncorrectable errors on system reliability, availability, and serviceability (RAS). The MPR technique allows memory pages suffering from correctable errors and relocatable clean pages suffering from uncorrectable errors to be removed from use in the virtual memory system without interrupting user applications. It also allows relocatable dirty pages associated with uncorrectable errors to be isolated with limited impact on affected user processes, avoiding an outage for the entire system. This study applies analytical models, with parameters calibrated by field experience, to quantify the reduction that can be made by this operating system self-healing technique on the system interruptions, yearly downtime, and number of services introduced by hardware permanent faults, for typical low-end and mid-range server systems. The results show that significant improvements can be made on these three system RAS metrics by deploying the MPR capability.

1. Introduction

Modern operating systems (OS), especially those running on computer servers, are incorporating more and more self-healing techniques [3, 6]. By self-healing, we mean the system is able to diagnose and recover from hardware and software component level failures automatically, without causing any apparent interruption to most applications. In order for a system to be self-healing, it must have robust failure detection, diagnosis, isolation, and handling capabilities. The self-healing for hardware problems is typically done by taking the failed component, such as a CPU, memory region, or I/O channel, off line while continuing to operate with the remaining system resources, achieving a graceful degradation rather than an undesired disruption of the entire system.

A recent study [5] of field failure data collected by Microsoft from Windows XP machines reveals that memory is the most frequently failing component among hardware failures. This conclusion matches the result of a case study of Sun low-end and mid-range servers in the field that memory is at the top of the system downtime pareto for all hardware problems. Mitigating the disruption of memory failures on system operation is thus expected to have significant effect on the system reliability and availability. This issue has been realized by major server vendors and addressed in advanced UNIX systems. For instance, dynamic configuration of memory blocks has recently been implemented in the AIX system [4] and automatic page retirement has recently been implemented in the Solaris system [1].

The Solaris memory page retirement (MPR) capability is a self-healing technique that removes a physical page of memory from use by the system in response to Error Correction Code (ECC) errors associated with that page. The enhanced MPR technology in Solaris 10 is driven by a Fault Manager containing diagnosis software [7] that examines any memory correctable errors (CEs) and uncorrectable errors (UEs) detected by the underlying hardware. The diagnosis software can retire memory pages containing CEs and relocatable clean pages (a clean page can be refilled from its backing object) containing UEs without interrupting user applications. It can also isolate relocatable dirty pages containing UEs with limited impact on affected user processes to avoid an outage for the entire system.

The Solaris MPR technology is also hardware independent: the implementation is all common code in the virtual memory subsystem, and the diagnosis software has been implemented for multiple variants of the UltraSPARC processor as well as the AMD Opteron™ family of processors. This technology could easily be adopted by other operating systems with a modern virtual memory subsystem. Therefore, a quantitative RAS analysis of MPR to measure the benefit of this technology is of interest to the OS developers, system designers, and end users. This study, using analytical models combined with field or engineering data, quantifies the reliability, availability, and service cost benefits of deploying MPR on typical low-end and mid-range data center servers.

The evaluation methodology used in this study is the hierarchical Markov modeling approach

implemented in RAScad [8], a Sun internal RAS architecture modeling tool with the automatic model generation capability. The tool supports integrated modeling of availability, performability and service cost. The rest of the paper is organized as follows. Section 2 gives an overview of MPR technique. Section 3 discusses the modeled configurations and associated assumptions. Section 4 presents the models and parameters. Section 5 analyzes results generated from the models. Section 6 concludes this study.

2. Overview of MPR

The MPR technique itself is quite simple: if a physical memory page is believed to be affected by an underlying hardware fault (e.g., a weak cell or faulty row in a memory chip or DRAM), the affected page can be retired by relocating its content to another physical page, and placing the retired page on a list of physical pages that should not be subsequently allocated by the virtual memory system. If the underlying fault manifests itself as one or more CEs to the operating system, the page retirement can be completed immediately by copying its content to another physical page and updating the virtual memory page translation tables.

If the underlying fault manifests as a UE on a clean page, the page can be retired and the OS can take a subsequent page fault to allocate a new physical page and re-read the contents of the page from the associated backing object. If a UE affects a dirty page and the UE is detected on a cache write back, Solaris marks the page as having a UE but defers action until the page is subsequently accessed (hoping the page will instead be freed). Finally, if a UE affects a dirty page and the error is detected on an access, the operating system forcibly terminates the affected process, retires the affected page, and then restarts the affected service. Therefore, only pages that are not relocatable at all, such as those used within particular regions of the kernel itself, are not retirable.

The diagnosis software uses knowledge of the hardware memory layout and processor memory error syndromes to drive its decision-making as to the use of MPR. If sufficient pages within a Dual In-line Memory Module (DIMM, the basic field replaceable memory component) have been retired, the software can publish a message to a human administrator indicating that it is time for the DIMM to be replaced.

MPR is implemented entirely in software, and is architecture-independent. It provides a cost-effective, complementary technique to hardware memory reliability technologies such as chipkill ECC and redundant memory channels and chips. It leverages the underlying ECC features and provides an additional recovery action for the system through the software approach. In particular, MPR provides a very cost-effective solution to memory failures that are isolated to small regions of cells: no additional hardware cost and performance cost are incurred, memory capacity gracefully degrades by only the size of a physical page (typically as small as 4K or 8K bytes), and memory error recovery is transparent to user applications.

3. Configurations and assumptions

In this study, two typical server systems, a low-end data center server (System 1) and a mid-range data center server (System 2), are selected for investigating the effect of the Solaris MPR feature. The studied platforms do not have sophisticated fault tolerant memory architectures such as chip sparing and automatic chip reconfiguration, other than the regular ECC (single-bit error correction and double-bit error detection) support. Some of the current generation server systems have incorporated the chipkill-correct ECC (which can correct multiple erroneous bits from one memory chip) technology [2]. The models discussed in this paper are also applicable to such memory architectures with relevant parameters (e.g., Fraction of Correctable Errors) adjusted to reflect the reduced percentage of uncorrectable errors due to the chipkill technology.

Some of the configurable components and their RAS features in the two systems are described below.

System 1 includes the following components.

- Four processor chips (degradable, no hot swap)
- 32 GB memory (degradable, no hot swap)
- Four hard disks (redundant, hot swappable)
- Four power supply units (redundant, hot swappable)
- Two PCI cards (non-redundant)

System 2 includes the following components.

- 24 processor chips (degradable, hot swappable)
- 192 GB memory (degradable, hot swappable)
- Six power supplies (redundant, hot swappable)
- Eight PCI cards (redundant, hot swappable)

Two models, one for the system without MPR and one for the system with MPR, are developed in this study to do quantitative comparisons. To simplify the study, both models assume constant failure and repair rates. In addition, the model for the system without MPR is based on the following assumptions:

- Repeated CEs caused by a memory fault does not disrupt the system but a scheduled service is required to replace the faulty memory component.
- UEs caused by a memory fault is handled by the automatic system recovery (ASR) functionality of the server architecture. That is, the system will de-configure the memory bank containing the faulty memory component by rebooting itself and return to a degraded operational mode. Later on, a scheduled service is required to replace the faulty memory component.

The model for the system with MPR is based on the following assumptions:

- When CEs or UEs occur, if all pages involved are retired successfully, the CEs/UEs will not disrupt the system and the faulty memory component will either not be replaced or be replaced (if the number of retired pages exceeds a threshold) in a deferred service action, therefore reducing unexpected system interruptions and downtime as well as the number of services.
- When CEs or UEs occur, if not all pages involved are retired successfully, the effect of CEs or UEs is

the same as that described above for the system without MPR.

4. Models and parameters

Figure 1 shows the top level Markov model for the system without MPR deployed. In the diagram, each circle represents a system state. If a state is marked by 1, it is an up state. If a state is marked by 0, it is a down state. When the system goes from an up state to a down state, a system interruption occurs. The expression beside a transition arrow represents the transition rate for the system to go from one state to another state. The rate out of a down state reflects downtime the system stays in that state. The notation used in the models is explained below.

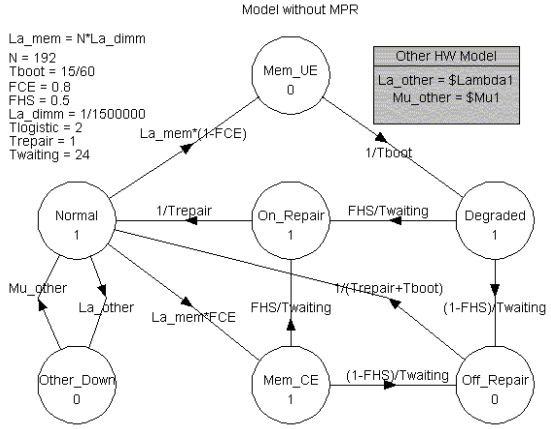

Figure 1. Top level model without MPR

- Normal: State in which the system is functioning properly (no faults)
- Mem_UE: State in which the system experienced UEs and is automatically restarting itself to de-configure the faulty memory bank
- Degraded: State in which the system is running in a degraded mode (after de-configuring the faulty memory bank) and waiting for replacement of the faulty memory component
- Mem_CE: State in which the system is experiencing repeated CEs and a service is scheduled to replace the faulty memory component
- On_Repair: State in which the faulty memory component is being replaced on line
- Off_Repair: State in which the faulty memory component is being replaced off line
- Other_Down: State in which the system is down due to faults of hardware other than memory
- La_mem: Permanent fault rate for entire memory
- La_dimm: Permanent fault rate for a 1GB DIMM (1/1,500,000 hours, provided by vendors)
- N: Number of 1GB DIMMs in the system
- FCE: Fraction of correctable errors among all memory errors (default = 80% for platforms with regular ECC, estimated from field data)
- FHS: Fraction of the faulty memory component replacements that are done by hot swap on line without interrupting the OS and user applications
- Tboot: System reboot time (5 minutes for System 1 and 15 minutes for System 2)
- Tlogistic: Logistic time – time for service personnel to arrive at the scene when the system is down (2 hours, determined by the best service contract)
- Twaiting: Service waiting time – waiting for off-peak hours to repair the system (24 hours)
- Trepair: Time of replacing the faulty memory component (1 hour, a conservative estimate
- La_other: System failure rate due to other hardware problems than memory faults
- Mu_other: System recovery rate from the above failure

When a CE occurs, the system goes into state Mem_CE where the fault may generate multiple CEs. In this state, the system no longer has the capability to correct an additional single-bit error in the checkword that already contains an erroneous bit from the faulty memory component. To avoid the potential hazard caused by this loss of memory redundancy, the system administrator is assumed to schedule a service to replace the faulty memory component at an off-peak time (Mem_CE -> Repair_Mem), to restore the redundancy. In the time window waiting for service, a second single-bit (hard or soft) error occurring in the same checkword would generate a UE. A calculation based on engineering standards shows that the occurrence probability of such an event is extremely low in the time window (24 hours) so that it is negligible and hence not shown in the model.

When a UE occurs that requires a system reset, the ASR functionality executes after the reboot but before the OS kernel initializes to de-configure the faulty memory bank. After ASR, the system runs in a degraded mode, waiting for an administer to replace the faulty memory component (Normal -> Mem_UE -> Degraded).

The replacement of faulty memory components as well as CPU's and associated boards can be done on line (no downtime) if the dynamic reconfiguration and hot swap functionality is provided by the hardware platform. Some users prefer the off-line repair actions (stop the system and incur downtime) to avoid possible damage caused by human error and other imperfect on-line repair problems. Thus, the parameter FHS is introduced in the model to account for the percentage of repairs that do not incur downtime. This parameter determines the fraction of the rate from state Degraded to states On_Repair or Off_Repair, both of which are service states. For the System 2 model, FHS is set to 50%, estimated based on our field observation, and a parametric analysis is done later on this parameter. For System 1, FHS is set to 0 because it has no hot swap capability for memory.

If a system failure occurs due to hardware problems other than memory faults, the system goes from the Normal state to the Other_Down state. The associated failure rate (La_other) and repair rate (Mu_other) are evaluated from a submodel called Other HW Model shown in the gray color rectangle box which represents the interface between the parent model and the submodel. The parameter's bindings are defined in the box, i.e., La_other and Mu_other, are bound to

the submodel output Lambda1 and Mu1 which are the equivalent failure rate and repair rate [8] of the submodel. Details of the submodel are not further discussed in this paper.

Figure 2 shows the top level Markov model for the system with MPR deployed. Compared with the model without MPR, this model has two more states: UE_Crash and CE_Remain to account for unsuccessful page retirement. The additional notation used in the model is explained below.

- UE_Crash: State in which the system is restarting itself due to unsuccessful attempts to isolate and recover from an uncorrectable error
- CE_Remain: State in which correctable errors remain in the system after unsuccessful attempts to retire pages associated with correctable errors
- Tpr: Time to page retirement (5 seconds, a conservative upper bound)
- Ps_ce: Probability of successful page retirement for correctable errors (default = 0.75, estimated from field data)
- Ps_ue: Probability of successful page retirement for uncorrectable errors (default = 0.5)
- FDR: Fraction of deferred replacement of faulty memory which occurs when the number of pages retired exceeds a threshold (default = 50%, estimated from field data)

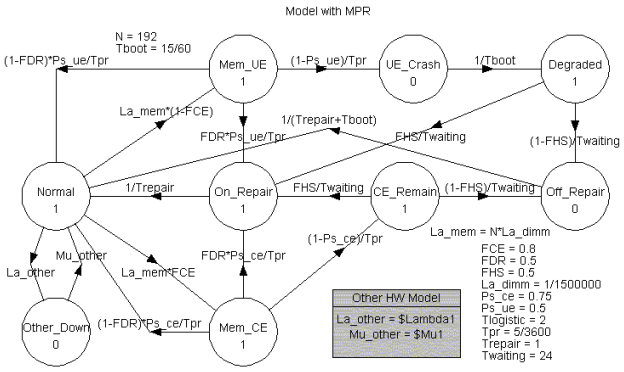

Figure 2. Top level model with MPR

When multiple CEs occur (a single CE event does not trigger MPR to avoid retiring pages as the result of a soft upset), the system tries to retire the affected pages. In most cases, pages containing CEs are retired successfully, thus avoiding a downtime and unexpected interruption to replace the faulty memory. If the number of pages retired exceeds a threshold, the associated faulty memory components can be replaced in a periodic maintenance event which is not counted as an interruption or downtime, or equivalent to an on-line repair in the model. This event is modeled by the parameter FDR which is set to 50%, an estimate from field data. However, the page retirement could fail due to various reasons such as imperfect handling of storms of correctable errors (Mem_CE -> CE_Remain). The probability of successful page retirement for correctable errors (Ps_ce) is set to 0.75, based on the field performance.

When a persistent UE occurs, the system tries to retire the affected page in state Mem_UE (notice this is no longer a down state). If the page is associated with a user process, whether it is clean or dirty, the page is either transparently retired (for the clean case), or the related user processes are killed and the corresponding service is automatically restarted by the Solaris Service Manager. In both cases, the system is kept in the up state and the page retirement is considered successful (Mem_UE -> Normal). Since the faulty memory bits have now been isolated, it is not yet necessary to schedule a service to replace the faulty memory component. If the page is associated with the kernel itself and is not relocatable, the system cannot retire the page and has to restart itself to de-configure the faulty memory bank (Mem_UE -> UE_Crash). After the restart, the system runs in a degraded mode, waiting for replacement of the faulty memory component. The probability of successful page retirement for UEs (Ps_ue) is assumed to be 0.5. In the uncertainty analysis discussed later, the parameter is varied in a wide range to show its effect on system availability.

The component level failure rates are calculated using methods described in Telcordia TR-NWT-000332 [10] with part-level (ICs, resistors, capacitors, etc.) failure rates adjusted based on field data, or directly estimated from field data, or provided by the OEM vendors. The field data used to calibrate component failure rates were collected from tens of thousands of field machines with billions of cumulative operating hours. In addition, two field case studies were conducted to estimate MPR specific parameters (FCE, Ps_ce, etc.). One study was based on the data representing 376,000 system hours of Sun mid-range servers for which detailed scenarios of CE and UE events as well as MPR actions were recorded. Another study was based on the data collected from 16,000 platforms of Sun high-end servers for about one year period, which showed that the number of DIMM dispatches to the field had been reduced by about 35% for systems with the deployment of MPR. In the next section, the most sensitive parameters are identified and an uncertainty analysis is conducted on these parameters.

5. Analysis of results

Tables 1 and 2 show results generated from the models with and without MPR discussed above, for the two systems. With MPR deployed, system reliability, availability, and service cost can all be improved. In particular, the system interruption rate can be reduced by 42% to 45%. The system yearly downtime can be reduced by 37% to 54%. The number of services can be reduced by about 20%.

Table 1. RAS results for system 1

Model	*Yearly Interruptions*	*Yearly Downtime*	*Yearly Services*
Without MPR	0.351	23.23min.	0.319
With MPR	0.202	14.65min.	0.253
Reduction by MPR	42.5%	36.9%	20.7%

Table 2. RAS results for system 2

Model	*Yearly Interruptions*	*Yearly Downtime*	*Yearly Services*
Without MPR	1.106	56.81 min.	1.918
With MPR	0.604	25.86 min.	1.528
Reduction by MPR	45.4%	54.5%	20.3%

The variance on RAS improvements can be explained by the RAS architectures of these systems. For a redundant component, the downtime associated with replacing the faulty component is just the repair time plus reboot time. For a non-redundant component, the failing of the component would keep the system down while waiting for the arrival of the service personnel, which makes the downtime much longer. That is, the more redundancy in the system, the less the fraction of downtime caused by other HW faults or the more effective the MPR on improving system availability, as seen from the downtime reduction for the two systems.

The above results were evaluated based on the parameters defined in the previous section. Some of these parameters are configuration or workload dependent (e.g., Tboot and La_dimm) and some are the FMA implementation related (e.g., Ps_ce and Ps_ue). What are the most sensitive parameters in terms of system downtime? A parameter sensitivity analysis was performed to identify sensitive parameters. The analysis results indicate that the system downtime is sensitive to the following parameters (ordered by sensitivity): Ps_ce, FHS, Trepair, FCE, La_dimm, Tboot, and Ps_ue. Among these parameters, La_dimm is independent of MPR technology, system configuration and maintenance and is discussed later. FHS is specific to System 2 (which has the memory hot swap capability) and is also discussed later. For all other sensitive parameters, we do an uncertainty analysis to investigate the impact of variance of these parameters on system RAS.

The uncertainty analysis is supported in RAScad and has been used in evaluating availability for the Sun Java System Application Server system [9]. The method performs random sampling from parameter ranges defined by the user and can address questions such as: Assume we have n (sample size) systems with each system's parameters selected by randomly sampling from possible ranges in customer sites or determined by workloads, configurations, and other factors, what are the average system availability and confidence intervals? The ranges for the selected parameters are defined as follows.

- Tboot: 2 - 10 min. for System 1, 10 - 20 min. for System 2
- Trepair: 0.5 - 1.5 hours
- FCE: 80% - 90%
- Ps_ce: 0.6 – 0.95
- Ps_ue: 0.25 – 0.75

The sample size is 10,000. In each sampling, RAScad randomly picks up a value for each parameter from the above ranges to calculate system results. The 90% confidence interval (90% CI) can be determined from the 10,000 points in the RAScad output, as shown in Tables 3 and 4. Taking these 90% CI's into account, the improvement made by MPR ranges from 37% to 59% for system interruptions, from 27% to 73% for system downtime, and from 17% to 26% for system services, depending on platforms, configurations and workloads.

Table 3. Uncertainty analysis results for System 1

Yearly Interrupts		*Yearly Downtime*		*Yearly Services*	
90% CI	*Reduc-tion*	*90% CI (min.)*	*Reduc-tion*	*90% CI*	*Reduc-tion*
0.161	54.1%	12.26	47.2%	0.237	25.7%
0.220	37.3%	16.99	26.9%	0.263	17.6%

Table 4. Uncertainty analysis results for System 2

Yearly Interrupts		*Yearly Downtime*		*Yearly Services*	
90% CI	*Reduc-tion*	*90% CI (min.)*	*Reduc-tion*	*90% CI*	*Reduc-tion*
0.452	59.0%	15.03	73.5%	1.430	25.4%
0.663	40.1%	34.96	38.5%	1.588	17.2%

The failure rate of memory components determines the memory contribution to the overall system failures and thus directly affects the results of the MPR RAS analysis. Figures 3 and 4 show how the variance of memory component failure rate in a possible range would change the results for Systems 1 and 2. When the DIMM MTBF (1/La_dimm) varies from 1,000,000 to 5,000,000 hours, the effect of MPR on system RAS can decrease by over 50%. A recent case study on a sample of mid-range data center servers in the field indicates that the average DIMM MTBF is at the low end of the above range (close to the vendor supplied data). However, we cannot exclude the possibility of higher MTBF numbers to be seen for DIMMs of better quality or running on different workloads. This parametric analysis provides an insight into the effect of the memory failure rate on the benefit of MPR.

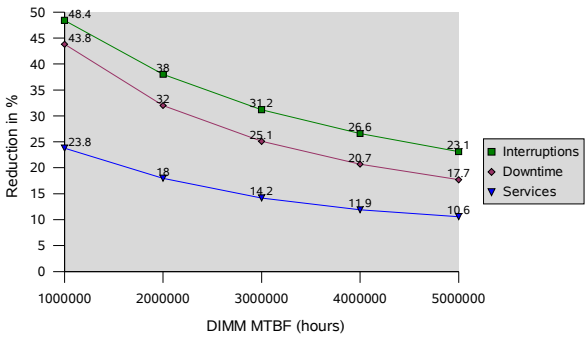

Figure 3. Effect of DIMM MTBF on system 1

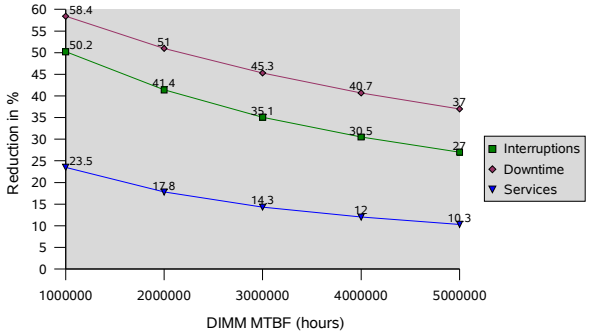

Figure 4. Effect of DIMM MTBF on system 2

Finally, we investigate the effect of a parameter specific to System 2, FHS, the fraction of hot swap replacement of faulty memory components. Figure 5 shows how this parameter affects the analysis results. The lower the FHS, the more the impact of MPR on system reliability and availability. That is, systems using less the hot swap capability would be more beneficial from the MPR deployment. This is because less hot swap translates to more system interruptions and downtime.

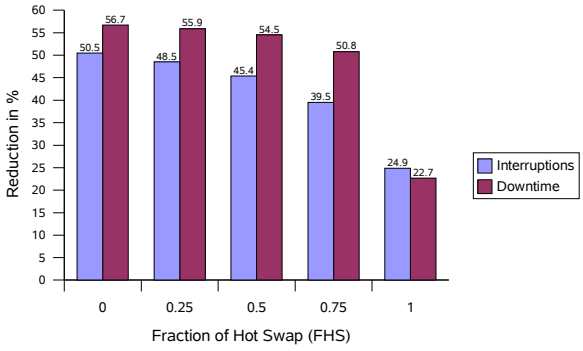

Figure 5. Effect of FHS on system 2

6. Conclusions

In this study Markov models, combined with field measurements, were used to assess the improvements made by the Solaris MPR self-healing technique on system interruptions, downtime, and service actions introduced by permanent memory faults, for typical server systems. The parameters used in the models were calibrated by field data or estimated using the engineering judgment. The uncertainty analysis was conducted to obtain the 90% confidence intervals by varying sensitive parameters in wide ranges. The parametric analysis was also done on key parameters. The results indicate that with MPR deployed, the system interruptions caused by hardware faults can be reduced by 42% to 45%, the system downtime caused by hardware faults can be reduced by 37% to 54%, and the number of system services incurred by hardware faults can be reduced by about 20%.

Since the memory configuration in the modeled platforms is maximized, the analysis results tend to be optimistic. However, these results are consistent with the field data in that memory is the top item of the system interruptions and downtime caused by hardware problems. The results are also consistent with our observation that the number of DIMM dispatches to the field has been reduced by about 35% for systems with the deployment of MPR.

Since the MPR feature is implemented entirely in software, it provides an extremely cost-effective RAS feature that complements hardware reliability features for memory such as redundant memory chips and lanes, and chipkill ECC. MPR can have a dramatic RAS improvement on systems with a large memory configuration that lacks redundancy, and therefore is especially well-suited to systems where the use of redundancy is constrained by cost, physical space, power, or cooling requirements. MPR has been successfully implemented for Solaris on both UltraSPARC and Opteron™ based systems, and can easily be added to any operating system with modern virtual memory management features.

Acknowledgments

Many thanks to Charlie Slayman, William Bryson, and Richard Elling for their valuable inputs to this study. Thanks also go to Peter Carr, Ganesh Ramamurthy, Tarek Derbas and Son Pham for providing support to this study. Thomas Simons initiated this study for addressing questions from customers in the telecommunication industry.

References

[1] T. M. Chalfant, *Solaris Operating System Availability Features*, Sun BluePrints™ OnLine, Jan. 2004.

[2] T. J. Dell, *A white Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory*, White Paper, IBM, Nov. 1997.

[3] A. G. Ganek and T. A. Corbi, "The Dawning of the Autonomic Computing Era," *IBM Systems Journal*, Vol. 42, No. 1, Jan. 2003.

[4] J. Jann, L. M. Browning, and R. S. Burugula, "Dynamic Reconfiguration: Basic Building Blocks for Autonomic Computing on IBM pSeries Servers," *IBM Systems Journal*, Vol. 42, No. 1, Jan. 2003.

[5] B. Murphy, "Automating Software Failure Reporting," *ACM Queue*, Vol. 2, No. 8, Nov. 2004.

[6] M. W. Shapiro, "Sel-Healing in Modern Operating Systems," *ACM Queue*, Vol. 2, No. 8, Nov. 2004.

[7] Sun Microsystems, *Predictive Self-Healing in the Solaris 10 Operating System – A Technical Introduction*, White Paper, June 2004.

[8] D. Tang and K. S. Trivedi, "Hierarchical Computation of Interval Availability and Related Metrics," *Proc. International Conference on Dependable Systems and Networks* (DSN-2004), June 2004.

[9] D. Tang, D. Kumar, S. Duvur, O. Torbjornsen, "Availability Measurement and Modeling for An Application Server," *Proc. International Conference on Dependable Systems and Networks* (DSN-2004), June 2004.

[10] Telcordia Technologies, *SR332 - Reliability Prediction Procedure of Electronic Equipment*, Issue 1, May 2001.