Scheduling Tradeoffs for Heterogeneous Computing on an Advanced Space Processing Platform

Ian Troxel and Alan D. George
{troxel, george}@hcs.ufl.edu

High-performance Computing and Simulation (HCS) Research Laboratory
Department of Electrical and Computer Engineering, University of Florida
Gainesville, Florida, 32611-6200

Abstract

Future spaceborne platforms will require expanded onboard processing payloads to meet increasing mission performance and autonomy requirements. Recently proposed spacecraft systems plan to deploy networked processors configured much like commodity clusters for high-performance computing (HPC). Just as robust job management services have been developed and are required to optimize the performance of ground-based systems, so too will spaceborne clusters require similar management services, especially to meet real-time mission deadlines. In order to gain insight into how best to address the challenge of job management in high-performance, embedded space systems, a management service has been developed for a NASA New Millennium Program (NMP) experiment for the ST-8 mission slated for launch in 2009. This paper presents an overview and analysis of the effects on overall mission performance of adding priority and preemption to a baseline gang scheduler employing opportunistic load-balancing (OLB) on a heterogeneous processing system for space. Experiments are conducted with two mission scenarios including planetary mapping and object tracking.

Keywords: Gang Scheduling, Preemption, Embedded Space Systems, Real-time Systems, Heterogeneous Computing

1. INTRODUCTION

The relatively limited processing capability available on satellites, deep-space probes, and space exploration vehicles requires the careful orchestration of applications in order to use resources as efficiently as possible. Space-system design requirements such as limited power, volume, and weight, and tolerance of radiation, temperature, and vibration effects tend to further reduce the performance offered to space-science applications. For example, the Radioisotope Thermoelectric Generator on the Cassini spacecraft, launched in 1997 to explore Saturn, is one of the most sophisticated power plants ever placed on a space system yet only provides a nominal output of several hundred Watts of electric power [1]. By comparison, a single server node in a typical HPC cluster draws approximately the same amount of power. Space platforms often feature a combination of radiation-hardened components, which are typically much more costly and less powerful than their Commercial-Off-The-Shelf (COTS) counterparts, as well as redundant components as active or passive spares. Success with space processing missions is dictated by results from the execution of one or more critical applications often with real-time deadlines, thereby making scheduling and time-slicing of resources a challenge. As such, scheduling techniques for space systems are often deterministic (i.e. with strict synchronization), performed *a priori*, and tailored specifically to both application requirements and the underlying customized platform. However, there has been a recent trend at NASA to deploy space platforms with COTS components to reduce custom design costs and complexities and increase system performance and cost-effectiveness.

Embedded COTS clusters for space such as the Dependable Multiprocessor (DM), in development at Honeywell Inc. and the University of Florida for NMP supported by NASA JPL, seek to provide a cost-effective, standards-based processing platform with a relatively seamless transition from ground-based computational clusters to space systems [2]. The DM platform features a cluster of COTS data-processing nodes equipped with PowerPC processors and Field-Programmable Gate Array (FPGA) devices as coprocessors, a pair of radiation-hardened system controller nodes, a Gigabit Ethernet data network, a standard Linux distribution, a fault-tolerant, lightweight version of the Message-Passing Interface (MPI), and a robust job and system management service. Along with managing fault detection and correction on radiation-susceptible COTS devices, a key research topic being addressed by the DM system management service is toward meeting scientific mission requirements and scheduling critical applications with real-time deadlines on a system with heterogeneous resources. In addition, the job management task is further complicated by the need to design the system to allow mission developers the option of customizing the scheduler and management service for their particular mission, yet providing the flexibility to allow the DM system to be tailored to meet the needs of a variety of missions.

The remainder of this paper examines the design, performance, and flexibility of the DM system's job manager and is divided as follows. Section 2 provides a background of related research, and Section 3 provides an overview of the DM management software. The prototype system on which the DM job management concepts are being evaluated is described in Section 4, and Section 5 presents experimental performance results for two mission scenarios. Finally, conclusions are enumerated and directions for future research briefly cited in Section 6.

2. RELATED RESEARCH

There have been several significant efforts to develop COTS cluster platforms in space. The Remote Exploration Experimentation (REE) project championed by NASA JPL sought to develop a scalable, fault-tolerant, high-performance supercomputer for space composed of COTS processors and networking components. The REE system design deployed Software-Implemented Fault Tolerance (SIFT) middleware layer, known as the Adaptive Reconfigurable Mobile Objects of Reliability (ARMOR), between a commercial operating system and the applications. This approach offered a customizable level of fault tolerance based upon reliability and efficiency requirements. Within ARMOR, a centralized fault-tolerance manager oversees the correct execution of applications through remote agents that are deployed and removed with each application execution [3]. The preliminary implementation of this system was an important first step and showed promise with testing performed via a fault-injection tool. Unfortunately, to the best of our knowledge, a detailed analysis of how the system would schedule multiple applications was not undertaken or published. In our work with developing the DM middleware, insight was gleaned from the REE system, and the DM design will address some of the perceived shortcomings in terms of the potential scalability limitations of the REE middleware.

The Reconfigurable Scalable Computing (RSC) project, led by NASA LARC, proposes a compute cluster for space composed entirely of triple-redundant MicroBlaze softcore processors running uC-Linux and hosted on Xilinx Virtex-4 FPGAs [4]. Replication is a popular and worthy approach and alternative to SIFT, but of course such hardware redundancy can impose a large overhead on resources. Like DM, RSC is a good example of an embedded cluster for space that will employ commodity components. By contrast, the DM system will also employ FPGAs as a primary means to improve performance, and will use a combination of SIFT, internal FPGA mechanisms, and periodic scrubbing to overcome faults.

Much research has been conducted to improve the performance and fault-tolerance of conventional HPC clusters, ideas that can be leveraged for embedded space systems. Although ground-based systems do not share the same strict power and environmental constraints as space systems, improving fault-tolerance and availability is nonetheless important as such systems frequently execute critical applications with long execution times. For example, the Dynamic Agent Replication eXtension (DARX) framework provides a simple mechanism for deploying replicated applications in an agent-based distributed computing environment [5]. The modular and scalable approach provided by DARX would likely provide performance benefits for ground-based systems but may be too resource-intensive to appropriately function on resource-limited embedded platforms. The Comprehensive Approach to Reconfigurable Management Architecture (CARMA), under development at the University of Florida, aims to make executing FPGA-accelerated jobs in an HPC environment as easy and reliable as it currently is to execute jobs in traditional HPC environments [6]. Elements of each of these services have influenced the DM middleware design; in particular, CARMA's ability to seamlessly integrate FPGA devices into the job scheduling system partially formed the basis of the DM scheduler.

In addition to management services, numerous job scheduling techniques commonly used in large-scale cluster systems have been investigated in the literature. Some of the areas include batch scheduling (i.e. applications scheduled to nodes for the duration of their execution time without preemption), gang scheduling (i.e. applications scheduled in a lock-step fashion where preemption is used to optimize resource utilization), and coscheduling [7] whereby application metrics are observed dynamically at runtime to continuously optimize job deployment. In order to provide a method to optimally utilize system resources with as little overhead as possible, the DM system employs a gang scheduler that uses OLB to quickly deploy parallel and serial applications. Batch scheduling was not chosen, since the inability to preempt applications would waste limited processor cycles, and coscheduling was not selected due to the overhead that would be imposed on the system. Due to the level of checkpointing required in the system to overcome faults due to radiation, parallel preemption is a relatively overhead-free operation (i.e. a simple kill of all parallel tasks). In addition, the heterogeneity of processing resources (e.g. availability of both CPUs and FPGAs) is taken into account when scheduling applications (dictated by a job's description file) to improve the quality of service that the applications observe.

3. DM MIDDLEWARE DESIGN

The DM middleware has been designed with the resource-limited environment typical of embedded space systems in mind. A top-level overview of the DM software architecture is illustrated in Figure 1. To achieve a standard runtime environment with which scientific application designers are accustomed, a commodity operating system (e.g. Linux variant) forms the basis for the software platform on both controller and data processors. Providing a COTS runtime system will allow scientists to develop their applications on inexpensive HPC clusters on the ground and then transfer their applications to the flight system with minimal effort. Such a flexible path to flight application deployment will reduce project costs and development time, ultimately leading to more science missions deployed over a given period of time.

The *Mission Manager* (MM) executes on a radiation-hardened processor and provides an abstract view of the DM system to ease mission development. Mission developers define the collection of applications they wish the DM system to execute through a select list of features such as prioritization, real-time deadlines, replication requirements, etc. Through this interface, the control features provided by the DM system can be tailored for any specific mission without lengthy custom development. The MM handles all

high-level scheduling decisions such as priority deployment and when preemption is appropriate to meet critical real-time deadlines. The MM also directs system-wide, fault-recovery schemes in order to ensure the overall mission requirements are met at all times including the deployment of application replicas with voting if deemed necessary to meet a mission's reliability requirements.

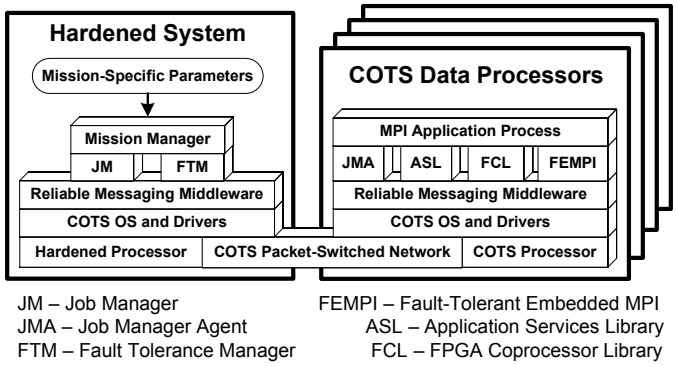

Figure 1. DM Middleware Architecture

The primary functions of the *Job Manager* (JM) are low-level application scheduling, dispatching processes, and directing application recovery based upon user-defined policies and direction from the MM. The JM employs an OLB scheduler, with gang scheduling for parallel jobs, which receives frequent system status updates from the Fault-Tolerance Manager (FTM) in order to maximize system availability. In addition, the scheduler optimizes the use of heterogeneous resources such as FPGA accelerators with strategies borrowed from the CARMA runtime job management service [6]. Jobs are described using a Directed Acyclic Graph (DAG) and are registered and tracked in the system by the JM via tables detailing the state of all jobs, be they pending, currently executing, or suspected as failed and under recovery. These various job buffers are frequently checkpointed to enable seamless recovery of the JM and all outstanding jobs. The JM heartbeats to the FTM via reliable messaging middleware to ensure system integrity and, if an unrecoverable failure on the control processor occurs, the backup controller is booted and the new JM loads the checkpointed tables and continues job scheduling from the last checkpoint.

4. EXPERIMENTAL SETUP

For the current phase of the DM project, a prototype system designed to mirror when possible and emulate when necessary the features of a typical satellite system has been developed. As shown in Figure 2, the prototype hardware consists of a collection of COTS Single-Board Computers (SBCs) executing Linux, one augmented with an FPGA coprocessor, a reset controller and power supply for power-off resets, redundant Ethernet switches. Six SBCs are used to mirror the specified number of data processor boards the flight experiment (four) and also to emulate the functionality of radiation-hardened components (two) currently under development. Each SBC is comprised of a 650MHz PowerPC, memory and dual 100Mbps Ethernet interfaces. One data processor is equipped with an Alpha Data ADM-XRC-II accelerator card featuring a Xilinx Virtex-II 6000 FPGA, thus making the system heterogeneous. The SBCs are interconnected with two COTS Ethernet switches and Ethernet is used for all system communication. A Linux workstation emulates the role of the *Spacecraft Command and Control Processor*, which is responsible for communication with and control of the DM system but is outside the scope of this paper.

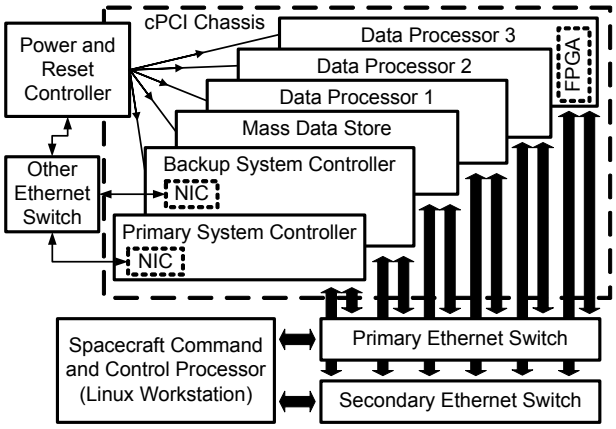

Figure 2. Testbed Configuration

Two realistic mission scenarios have been devised to test the effectiveness of the DM scheduler. The first is a *planetary mapping* mission that includes three algorithms for high- and low-resolution image processing of a celestial object. An image filtering algorithm continuously performs an edge-detection on a stream of images to provide surface contour information, while a Synthetic Aperture Radar (SAR) kernel [8] periodically processes a data cube representing a wide-swath radar image of the object's surface. When an image buffer is filled above a pre-defined level, a data compression algorithm compresses the images for downlink. The second mission performs *object tracking* with the same three algorithms used in the mapping mission but with a slight variation that dramatically changes the mission's runtime characteristics. As in the planetary mapping mission, the image filtering algorithm continuously performs an edge detection of input images. However, the SAR algorithm only executes a detailed radar scan if an object or event of interest is discovered, thus the SAR kernel is non-periodic. The characteristics of these algorithms are summarized in Table 1. The execution times, as measured on the testbed previously defined, for each algorithm executing on one or multiple SBCs with or without the use of an FPGA coprocessor are also provided.

Execution times are denoted in terms of the number and type of nodes on which the application executes. For example, *1 SW SBC* denotes the case when the application executes only on the CPU of one SBC, while *1 HW SBC* denotes execution on one SBC using its FPGA coprocessor. Similarly, *2 SW SBCs* defines the case when an application is deployed in a parallel fashion on two nodes without an FPGA. The remaining two designations in Table 1 denote cases where a parallel application executes on a mixture of processor–only and FPGA-accelerated nodes.

Table 1. Mission and Algorithm Characteristics

Mission Description	Scheduling Mode	Initial Priority Level	Execution Time Measured on Testbed (seconds)				
			1 SW SBC	1 HW SBC	2 SW SBCs	1SW and 1HW SBC	2SW and 1HW SBC
1) Planetary Mapping							
Image Filtering	Continuous	Medium	10	5	5	4	3
SAR	Periodic	High	600	30	300	25	20
Data Compression	Non-Periodic	Low	50	N/A	25	N/A	N/A
2) Object Tracking							
Image Filtering	Continuous	Medium	10	5	5	4	3
SAR	Non-Periodic	High	600	30	300	25	20
Data Compression	Non-Periodic	Low	50	N/A	25	N/A	N/A

Multi-SBC versions of the image filtering algorithm are decomposed in a coarse, data-parallel manner such that each process is independently filtering one image at a time from a stream of images. In this manner, the total number of images to filter per processing interval is evenly distributed across the coordinating SBCs, although the total number of images to process varies each time that the application executes due to the time delay between successive deployments. For clarity, execution times denoted in the table for image filtering are the effective execution time required per image. SAR is decomposed across multiple SBCs in a parallel fashion with all SBCs coordinating to complete a single iteration of the fixed-size SAR data set. Data compression is decomposed across up to two SBCs in a distributed manner where the SBCs evenly divide the fixed-size data set then compress their portions independently. An FPGA-accelerated version of the compression algorithm was not implemented, as noted with "N/A" in the table.

5. SCHEDULING ANALYSIS

Several experiments were conducted to analyze the ability of the DM scheduler to effectively meet mission deadlines and to assess each scheme's impact on performance and buffer utilization. Data Ingress defines the rate at which data (i.e. images) enter the processing system from external sensors. Images are fixed at 2.6MB each in the experiments. For Mission 1, SAR processes 50 images worth of data is therefore periodically scheduled every time that 50 images enter the system. However, the non-periodic version of SAR deployed in Mission 2 is only executed when an object of event of interest is detected, and this detection event is emulated as a Poisson distribution with a mean equal to that of the periodicity of the SAR process used in Mission 1 for a fair comparison between results. Two scheduling improvements are considered, including use of application priorities in scheduling decisions, denoted *Priority*, and use of preemption mechanisms to remove one or more currently executing job to provide additional resources to a higher-priority application. These two enhancements are compared to a Baseline approach with neither of these features.

The priority levels noted in Table 1 are based upon the criticality of the application to the overall mission. However, the DM scheduler applies a best-effort approach to quality of service whereby if an application reaches a critical threshold then its priority may be increased. For example, the image filtering application becomes high priority if the number of images to be filtered is double its allocated buffer size (i.e. 100 images), and the compression algorithm becomes medium or high priority if the data to be compressed is double (i.e. 200 images) or triple (i.e. 300 images) its allocated buffer size, respectively. Preemption occurs when an application with a higher priority has a full input buffer and preempting a lower priority application will provide the necessary system resources. Applications with equal priority levels cannot preempt each other but, in the case when two applications of equal priority are simultaneously available for deployment, the application that completed most recently will not be chosen.

5.1. Mission 1: Planetary Mapping

In the experiments, real-time deadlines are treated as a dependent variable, thus providing a measure of the ability of one scheme versus another. The actual real-time deadline range for the applications highlighted in these experiments vary from 0.5 to 100 seconds depending upon the criticality of mission and other system-specific parameters. The ability of the DM scheduler to meet the real-time deadline requirements of the planetary mapping mission is illustrated in Figure 3. The Minimum Real-Time Deadline is a measure of the maximum amount of time that the SAR algorithm required to execute, measured from the time its processing period began until the time when the output was written to the data compression buffer. The results show that the Baseline approach is very poor at meeting the real-time deadline requirement due to an inefficient use of processing resources. The Priority scheme provides a marked improvement over the Baseline system and the Preemption provides additional improvement.

By contrast, the results in Figure 4 show that the Preemption scheme requires a larger buffer to be deployed in the system as data ingress increases, due to resource contention. The Priority scheme requires the system to deploy less buffer space when data ingress increases because applications are scheduled more efficiently without resource contention due to thrashing.

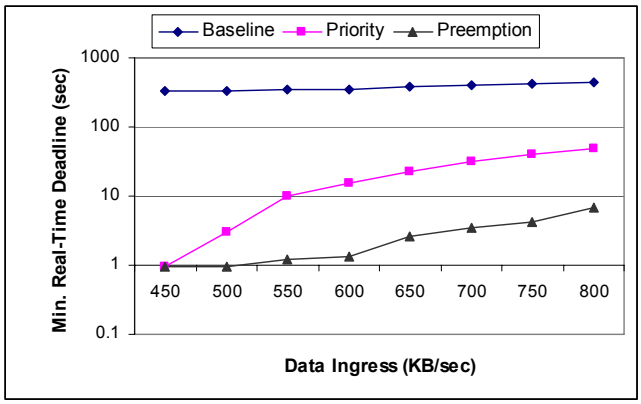

Figure 3. Minimum Real-Time Deadline vs. Data Ingress

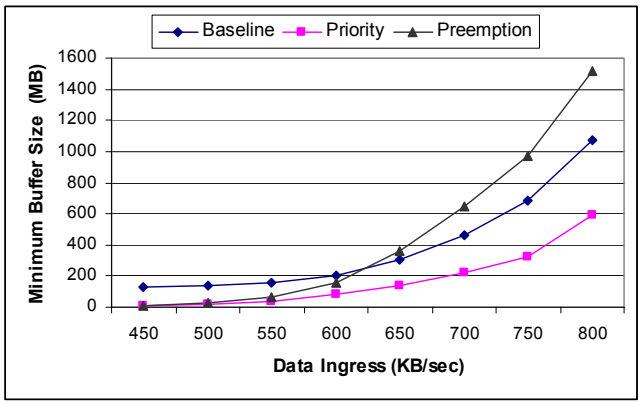

Figure 4. Buffer Utilization vs. Data Ingress

Overall mission performance results are provided in terms of throughput in Figure 5. The Preemption strategy provides the best performance up to 650KB/sec of ingress rate, but shows a continual decrease in performance due to thrashing. Essentially, the compression algorithm's buffer quickly fills up above this rate and becomes elevated to high priority in order to make room for more output data. The Priority scheme does not suffer the same thrashing effects because preemption is not permitted, so the performance of this scheme matches the Baseline above the critical point. Above the ingress rate of 650KB/sec, overhead due to the DM system middleware (i.e. monitoring, deploying, checkpointing, etc.) begins to affect the execution of Mission 1. Future improvements to the system will allow greater data ingress rates to be achieved without penalty.

These results suggest that the Preemption strategy provides performance improvements for critical applications with a periodic deployment interval to a point at which the performance of the overall system is reduced to keep meeting the critical application's deadline. The Priority scheme provides an improvement over the Baseline scheme in all cases and mitigates the resource thrashing problem that degrades performance of the Preemption method.

5.2. Mission 2: Object Tracking

In contrast to Mission 1, the ability of the DM scheduler to meet the real-time deadline requirements of the object tracking mission, one which contains a non-periodic application, is illustrated in Figure 6. The results show that the Baseline approach is also very poor at meeting the SAR real-time deadline requirement due to an inefficient use of processing resources. The Priority scheme still provides a marked improvement over the Baseline system but, in contrast to the results from Mission 1, the ability to meet the real-time deadline is somewhat reduced due to the unpredictable nature of when the SAR application will be executed. Without the ability of the Priority scheme to give adequate precedence, the SAR algorithm spends more time waiting in the job queue for available resources. This problem is averted by allowing SAR to displace already deployed applications in the Preempt scheme.

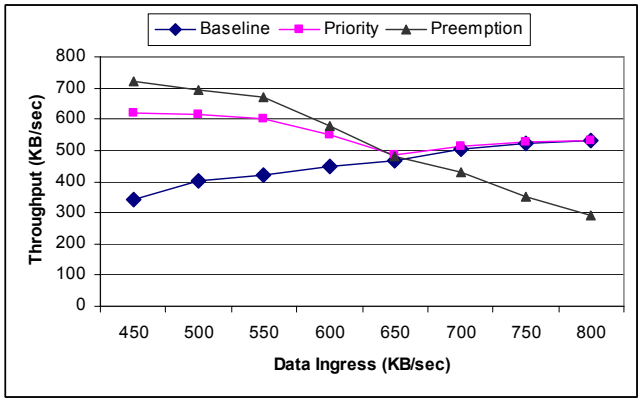

Figure 5. Data Throughput vs. Data Ingress

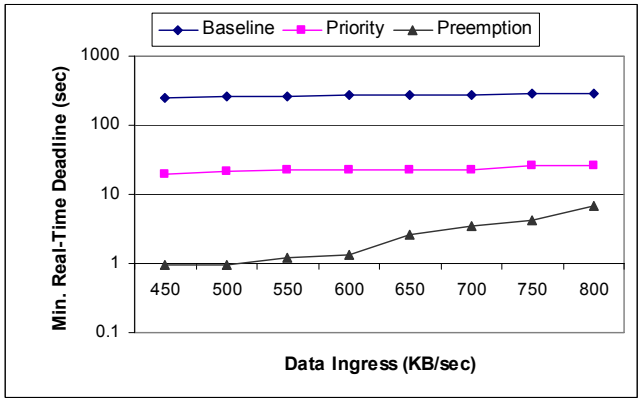

Figure 6. Minimum Real-Time Deadline vs. Data Ingress

The memory requirements for each scheme, presented in Figure 7, are dramatically reduced as compared to the results observed in Mission 1, because the SAR application is more randomly distributed, thereby reducing and amortizing the likelihood of contention. The memory requirements of the Priority scheme have increased relative to the Baseline scheme due to the increased queuing delay that SAR observes as previously described.

The results in Figure 8 highlight overall performance for Mission 2. The Preemption strategy again provides the best performance up to about 640KB/sec and still shows a continual decrease in performance due to thrashing when the compression algorithm's buffer fills. The Priority scheme does not schedule as efficiently as the data ingress rate increases and therefore no longer tracks the performance of the Baseline scheme. In fact, the Baseline

scheme demonstrates a slight improvement over the results from Mission 1, since the simple first-in first-out approach outperforms the two contrived schemes. However, overall system throughput improvement is not advantageous since the minimum real-time deadline that the SAR algorithm can expect under the Baseline scheme is in hundreds of seconds.

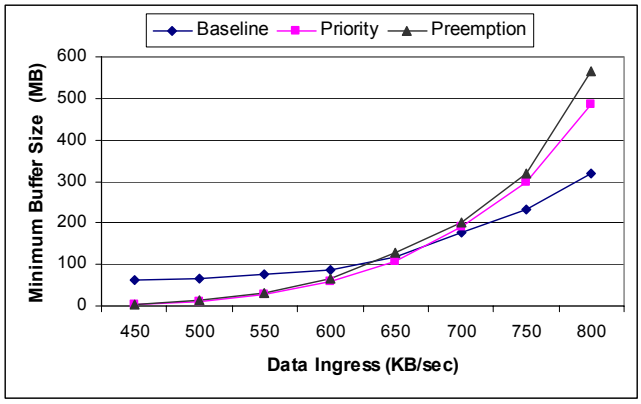

Figure 7. Buffer Utilization vs. Data Ingress

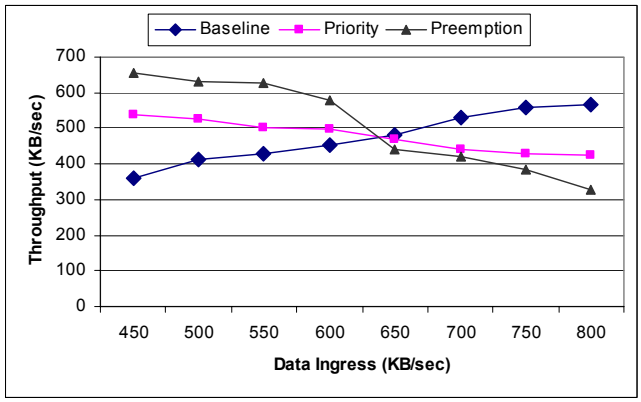

Figure 8. Data Throughput vs. Data Ingress

6. CONCLUSIONS

The Dependable Multiprocessor (DM) is a COTS-based, processing platform for scientific applications in space. The system provides a standard development and runtime environment familiar to scientific application developers, and a robust management service to overcome the Single-Event Upset (SEU) susceptibility of COTS components, among other features. Several experiments were undertaken to analyze the ability of the DM scheduler to meet real-time mission deadlines for applications in earth or space science.

The results show that the Baseline approach, an OLB-based gang scheduler, provides good overall system performance for missions having applications with random execution patterns, but cannot meet typical real-time deadline requirements of either of the case-study missions. The Priority scheme is the best overall method for missions that incorporate applications with periodic execution patterns, but it suffers a performance reduction when scheduling applications with random execution patterns. The Preemption strategy provides performance improvements for critical applications with a periodic deployment interval to a point at which the performance of the overall system is reduced, since resource thrashing ensues as the scheduler attempts to meet the most critical deadlines. Also, the Preemption strategy requires the system to deploy the most memory buffer resources. Future work for the DM project includes additional fault-tolerance and availability analysis for failure rates likely to be observed in orbital and deep-space missions. In addition, other SIFT mechanisms will be developed through the MM including spatial and temporal job replication and algorithm-based fault tolerance.

7. ACKNOWLEDGEMENTS

The authors wish to thank Honeywell Inc. and NASA JPL for their support of this research, and a special thanks to Vikas Aggarwal at Tandel Inc. and Grzegorz Cieslewski and Jim Greco of the HCS Lab at Florida for developing benchmarks used in our experiments.

8. REFERENCES

[1] R. Gibbs, "Cassini Spacecraft Design," *Proc. International Society for Optical Engineering (SPIE)*, 2803, pp. 246-258, 1996.

[2] J. Samson, J. Ramos, I. Troxel, R. Subramaniyan, A. Jacobs, J. Greco, G. Cieslewski, J. Curreri, E. Grobelny, A. George, V. Aggarwal, M. Patel and R. Some, "High-Performance, Dependable Multiprocessor," *Proc. IEEE Aerospace Conf.*, Big Sky, MT, March 4-11, 2006.

[3] K. Whisnant, R. Iyer, Z. Kalbarczyk, P. Jones III, D. Rennels and R. Some, "The Effects of an ARMOR-Based SIFT Environment on the Performance and Dependability of User Applications," *IEEE Transactions on Software Engineering*, 30(4), 2004.

[4] J. Williams, N. Bergmann, and R. Hodson, "A Linux-based Software Platform for the Reconfigurable Scalable Computing Project," *Proc. Conf. on Military and Aerospace Programmable Logic Devices (MAPLD)*, Washington, DC, September 7-9, 2005.

[5] M. Bertier, O. Marin and P. Sens, "A Framework for the Fault-Tolerant Support of Agent Software," *Proc. Symp. on Software Reliability Engineering (ISSRE)*, Boulder, CO, November 17-20, 2003.

[6] I. Troxel, A. Jacob, A. George, R. Subramaniyan and M. Radlinski, "CARMA: A Comprehensive Management Framework for High-Performance Reconfigurable Computing," *Proc. Conf. on Military and Aerospace Programmable Logic Devices (MAPLD)*, Washington, DC, September 8-10, 2004.

[7] E. Frachtenberg, D. Feitelsoná, F. Petrini, and Juan Fern andez, "Adaptive Parallel Job Scheduling with Flexible CoScheduling," in *IEEE Transactions on Parallel and Distributed Systems*, 11(16), November 2005.

[8] J. Greco, G. Cieslewski, A. Jacobs, I. Troxel, C. Conger, J. Curreri, and A. George, "Hardware/software Interface for High-performance Space Computing with FPGA Coprocessors," *Proc. IEEE Aerospace Conf.*, Big Sky, MN, March 4-11, 2006.