Analysis of Non-Work-Conserving Processor Partitioning Policies [⋆]

E. Rosti[1], E. Smirni[2], G. Serazzi[3], L.W. Dowdy[2]

[1] Dipartimento di Scienze dell'Informazione
Università di Milano, Italy
rose@dsi.unimi.it

[2] Department of Computer Science, Vanderibilt Univeristy
Nashville TN 37235, USA
esmirni,dowdy@vuse.vanderbilt.edu

[3] Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy
serazzi@elet.polimi.it

Abstract. In multiprocessor systems, a reasonable goal of the scheduler is to keep all processors as busy as possible. One technique for doing this is to allocate all available processors to the jobs waiting for service. Techniques which allocate all available processors are known as work-conserving policies. In this paper, non-work-conserving policies are examined. These policies keep some number of processors idle (i.e., unallocated) even when there are parallel jobs that are waiting for service. Such non-work-conserving policies set aside idle processors for anticipated new job arrivals or for unexpected system behavior. Two classes of non-work-conserving space-sharing policies are examined. One policy class keeps a certain percentage of the processors free. The other policy class makes an allocation decision based on previously observed system behavior. Two non-work-conserving policies, each selected from the two classes, are evaluated against their work-conserving counterparts. It is demonstrated that non-work-conserving policies can be particularly useful when the workload or the system behavior are irregular. Variability in the workload behavior including bursty arrivals, a high coefficient of variation in the workload execution time, unstable systems with processor failures are among the situations where non-work-conserving policies improve performance.

1 Introduction

General purpose multiprocessor systems offer considerable computational power which can be used to solve problems with large computational requirements.

[⋆] This work was partially supported by Italian M.U.R.S.T. 40% and 60% projects, and by sub-contract 19X-SL131V from the Oak Ridge National Laboratory managed by Martin Marietta Energy Systems, Inc. for the U.S. Department of Energy under contract no. DE-AC05-84OR21400.

However, it is not always possible to efficiently exploit all available processors. The nature of the problems to be solved, the architectural characteristics of the parallel system (e.g., topology, interconnection network, memory architecture), the application's computation and communication requirements, are among the factors that limit the efficient use of massively parallel systems. Multiprogramming is a viable way to improve system utilization while preserving individual application performance. However, multiprogramming complicates scheduling since an allocation policy is needed to determine the number of processors which should be allocated to each parallel program.

Preemptive or non-preemptive [SRDS93] processor allocation policies have been proposed. Preemptive policies [Oust82, MEB88, PD89, FR90, DCDP90, LV90, ZM90, MVZ93, CMV94, MZ94] allow processor redistribution upon job arrivals and departures or when a time quantum expires. Processors may be reclaimed from an executing job's assignment and distributed to newly arrived jobs, or additional processors may be added to an executing job's assignment when processors become available. Dynamic space-sharing policies and time-sharing policies are considered preemptive. Non-preemptive policies [Sev89, ZM90, GST91, MEB91, RSDSC94, Sev94, CMV94] keep the number of processors assigned to a job constant during execution. Processor allocation decisions occur only before execution starts. Static and adaptive space-sharing policies are considered non-preemptive. Non-preemptive policies are characterized by low overhead and easy implementation.

In this paper, the focus is on non-work-conserving adaptive space-sharing policies for general purpose multiprocessor systems[4]. These are policies that keep processors idle even in the presence of jobs waiting for service or policies that set aside processors for anticipated job arrivals. Traditionally, processor scheduling policies are devised with the goal of maximizing system utilization by assigning all available processors as soon as possible [ZM90, GST91, CMV94]. Work-conserving policies are natural in uniprocessors where there is no advantage from keeping the processor idle. When there are multiple processors, non-work-conserving policies may be effective.

A number of non-work-conserving policies have appeared in the literature and have proved to perform well. Restricting the number of processors assigned to a job up to the job's maximum parallelism has been used in the run-to-completion (RTC) policy presented in [ZM90]. The concept of processor working set (PWS) is used as a configuration parameter for several adaptive policies [GST91]. Among them, two non-work-conserving policies (FF and FF+LA) restrict the number of allocated processors allocated to the jobs' PWS. A policy that restricts the number of processors as a function of the average, minimum, and maximum parallelism of the application and the variance in parallelism has been proposed in [Sev89]. Three families of non-work-conserving policies have been presented in [RSDSC94]: the insurance policies (IP) that always attempt to "save" a fixed percentage of the free processors for anticipated future arrivals,

[4] Dedicated supercomputers are not the target machines of the policies investigated here. In these systems, policies that maximize throughput are commonly used.

the equal-partitioning-with-maximum (EPM) policies that restrict the number of processors assigned to each job according to a predefined MAX parameter, and the adaptive policies (AP) that resort to non-work-conserving decisions depending on limited knowledge of the system history. A uniform comparison of the adaptive space-sharing policies that appeared in the literature has been presented in [CMV94]. From this comparison, the ASP-MAX policy is distinguished. The ASP-MAX policy resorts to non-work-conserving scheduling decisions by using the available parallelism of each parallel application and a fixed percentage parameter.

A preliminary study of the advantages of leaving idle processors has been conducted in [SRSDS95]. In this work, the properties of one specific non-work-conserving strategy were investigated. It was found that performance improvements can occur from using a non-work-conserving policy when: 1) the workload does not scale linearly, 2) there exists a large variability in the workload inter-arrival times, and 3) the workload is comprised of multiple classes with different computational requirements.

This paper focuses on the potential benefits of two classes of non-work-conserving scheduling policies. The performance of these policies for various execution time distributions and for bursty arrivals is studied. The impact on performance of non-work-conserving decisions in the presence of processor failures is also explored. Measures to assess the degree of non-work-conservingness for each policy (e.g., the number of processors left idle with respect to the current assignment) are presented.

The paper is organized as follows. Non-work-conserving policies are described in Section 2. Section 3 illustrates the results of performance analysis for variable execution time distributions, arrival bursts, and processor failures. In Section 4 quantitative measures of the degree to which a policy is non-work-conserving are presented. Section 5 concludes the paper and summarizes the findings.

2 Non-Work-Conserving Policies

In this section the concept of non-work-conserving policies is illustrated. Two families of non-work-conserving space sharing policies found in the literature are described. In both cases, the policies schedule jobs in FIFO order, applying different non-work-conserving strategies for processor allocation, and have minimum overhead since they are of space sharing type.

In classical scheduling theory, non-work-conserving policies keep the resource idle or partially idle in the presence of work to be done. In uniprocessor systems, such a characteristic is harmful to performance when the cost of waiting for data (e.g., from cache) exceeds the cost of context switching. In multiprocessor systems, scheduling the available processors is a more difficult problem. Under space-sharing, multiple programs can execute simultaneously on disjoint subsets of processors called partitions. Unlike uniprocessor systems, not all available processors have to be assigned to achieve performance improvements.

At each scheduling round, a non-work-conserving decision is made when some of the available processors are not assigned and either 1) there are jobs still waiting for service in the queue or 2) the waiting queue is empty but the newly allocated partitions have been restricted such that some processors are left idle. The system is said to be in a non-work-conserving state if a non-work-conserving decision has just been made. Not all states with unallocated processors are considered to be non-work-conserving. For example, when processors become idle after a job finishes execution and the waiting queue is empty, the released processors are unallocated and are not redistributed among the executing jobs. Such system states are considered work-conserving since they are not the consequence of a non-work-conserving decision but rather of the non-preemptive characteristic of the policy. In contrast to non-work-conserving strategies, a work-conserving policy does not leave processors idle if there are jobs waiting for service. The number of scheduled jobs as well as the job partition size depend upon the processor allocation algorithm.

Prior work suggests that non-work-conserving policies may perform well when the system and workload behavior is irregular. These cases include:

- limited workload scalability (i.e., workloads with non-linear speedups),
- fluctuations in the arrival process of the parallel jobs, especially when the arrival process is bursty,
- multiclass workloads, composed of parallel applications which impose different demands on the system, and
- unstable systems where processor failures are possible.

In real environments, the interarrival time distribution of arriving jobs can be irregular. Arrival bursts can occur when users submit batches of jobs to the multiprocessor. In environments that suffer from processor failures, the interrupted jobs must be rescheduled. Recovery from processor failures allows the newly recovered processors to be reallocated. Multiclass workloads can place different demands on the system. Thus, wide variability exists and it is this variability that can be exploited by non-work-conserving policies.

The idea of keeping some processors idle is not new. However, it has not been analyzed in detail. In this paper, two families of non-work-conserving policies are examined. They represent two distinct ways of making non-work-conserving decisions.

2.1 The ASP-MAX Family

The ASP-MAX (Adaptive Static Partitioning with a Maximum) family of policies offers a complete range of policies whose performance is a function of the MAX parameter [CMV94]. The goal of the policy is to equally distribute all free processors to the jobs waiting in the queue. One constraint applies: the number of processors assigned to the job must be less than or equal to the minimum of the job's available parallelism and the parameter MAX. The parameter MAX is a fraction $p\%$ of the system size. When a job's available parallelism (i.e., the maximum number of processors the job can effectively use) is equal to the system size, the range of policies defined by the parameter MAX spans from non-work-conserving (when MAX $< 100\%$) to work-conserving (when MAX $= 100\%$). The policy performance is sensitive to the selection of the parameter MAX. It has been shown that an effective rule of thumb is to set MAX to 20% [CMV94].

If a job arrives at an empty system and its available parallelism is equal to the system size then the job is assigned a number of processors MAX$\times$`system_size`. Non-work-conserving decisions are made every time the system empties out and a new job arrives with an available parallelism equal to the system size. In the analysis presented in this paper, it is assumed that the the jobs' speedup curves are monotonically increasing. Therefore, the jobs' available parallelism exceeds the system size. The policy performance with various distributions of the available parallelism has been analyzed in [CMV94]. With ASP-MAX policies, the number of processors kept idle can be considerable (e.g., up to 80% of the system size when MAX $= 20\%$). Such a situation does not occur if there are jobs waiting for service in the queue. All waiting jobs are scheduled as long as there is a sufficient number of processors.

2.2 The PSA Family

An alternative way of being non-work-conserving is implemented by the PSA (Processor Saving Adaptive) policy [SRSDS95]. This policy does not force any a priori constraint on the size of the partition. Non-work-conserving decisions are made based upon the recent past system behavior.

At each scheduling round, the policy tries to maintain an equipartitioning scheme. The number of partitions is computed as a function of the number of jobs waiting in the queue. The partition size assigned to a waiting job is determined by the total system size divided by the number of partitions. If the number of free processors is smaller than the computed partition size, then a non-work-conserving decision is made. No job is scheduled and the free processors are kept idle. As the waiting queue increases, the partition size decreases proportionally. If the queue length is smaller than the current number of partitions and there are at least *two* free partitions of the previously computed partition size, then the partition size is increased. An exception is given by the case when the system becomes completely idle. As an illustration, consider the case when a single job is executing in the system. If the number of processors allocated to the a job is smaller than the whole system and the job completes before an arrival occurs, then the policy "remembers" that the system has been divided into more than one partition but only one was used. The new job that arrives will not be assigned the whole system, even though the whole machine is idle. This is in anticipation of another job arriving shortly, as occurred in the recent past. If no arrivals occur during the execution of the newly arrived job, the next incoming job (i.e., in the next scheduling round) will be assigned the entire system and the entire past history is erased. A detailed algorithmic description of the PSA policy is presented in [SRSDS95]. Under the PSA policy, memory of the system history is kept for one scheduling round. The severity of non-work-conserving decisions can be extended by increasing the number of scheduling rounds that keep track of the system history.

The focus of this paper is to investigate the impact of the different non-work-conserving strategies on system performance. The ASP-MAX and the PSA policies are analyzed. The next section identifies the conditions under which non-work-conserving policies perform well.

3 Performance Analysis

In this section, the performance of the ASP-MAX and PSA policies is investigated. The performance metric adopted is the response time ratio, defined as the ratio of the average response time under a given policy to the average response time under a reference policy [CMV94]. The absolute comparison of the ASP-MAX and PSA policies is not the purpose of this paper[5]. Since the goal of the paper is to study the impact of non-work-conserving decisions, the effects of such decisions must be isolated from those due to a different allocation strategy by using a different reference policy for each policy family. For each policy analyzed (i.e., ASP-MAX 20% and PSA), the reference policy considered is the corresponding work-conserving version (i.e., ASP-MAX 100% and the work-conserving PSA version).

Particular interest is devoted to policy performance under irregular workload behavior. Apart from the base case where exponential assumptions are made for the workload interarrival and service time, cases are analyzed where there is significant variability in the workload arrival and service processes. The results of bursty arrivals and limited availability in the processor set are also investigated.

A simulation study of systems with various sizes, namely 32, 64, 128, and 256 processors, is conducted. The size of the systems, as well as the policy complexity, prohibits the use of analytic models. Results are reported here for the 64 processor case only because they are representative of all system sizes. A set of 5 workloads with different speedups, spanning from nearly linear to almost flat is considered. Intermediate speedups are referred to as *concave* m% where m is the workload efficiency for the given system size. As an illustration, on a system with 128 processors, a *concave* 75% workload has a maximum speedup of 96. The maximum speedup is achieved with the total number of system processors (i.e., the speedup is monotonically increasing).

3.1 The Base Case

For the base case experiment, exponential distributions for the job interarrival and service times are assumed. In this experiment, the system does not suffer

[5] Because of the way partition size is computed, the ASP-MAX and PSA policies make non-work-conserving decisions differently. The ASP-MAX policies distribute the *free* processors to the jobs waiting in the queue under the constraint of a maximum partition size. Unequal partition sizes are allowed to co-exist. The PSA policies compute the partition size as the *total* number of system processors divided by the number of waiting jobs to limit the coexistence of unequal partition sizes.

from processor failures. Figure 1 plots the response time ratio as a function of the system utilization for the ASP and PSA policies. For the ASP policy, the parameter MAX is set to 20%. Experiments have been conducted with various MAX parameters but the results are not reported here for the sake of brevity.

Curves above the reference line (dashed line) indicate that the work-conserving policy is better. Curves below the dashed horizontal line indicate that the non-work-conserving policy is better. Five workload types are used, spanning from flat (*concave* 10%) to linear. The policy performance depends upon the system load and the workload speedup characteristics. If the workload scales well (i.e., above *concave* 50%), a fixed constraint such as the one imposed by the ASP-MAX policy hurts performance significantly. Non-work-conserving decisions yield an artificial inflation of response time since a possibly considerable amount of resources is wasted due to the MAX constraint. PSA yields better performance relative to its work-conserving counterpart. When the workloads do not scale well (i.e., *concave* 10% and *concave* 25%), a clear benefit is noted. The ASP-MAX policy can reach a performance gain of about 50%.

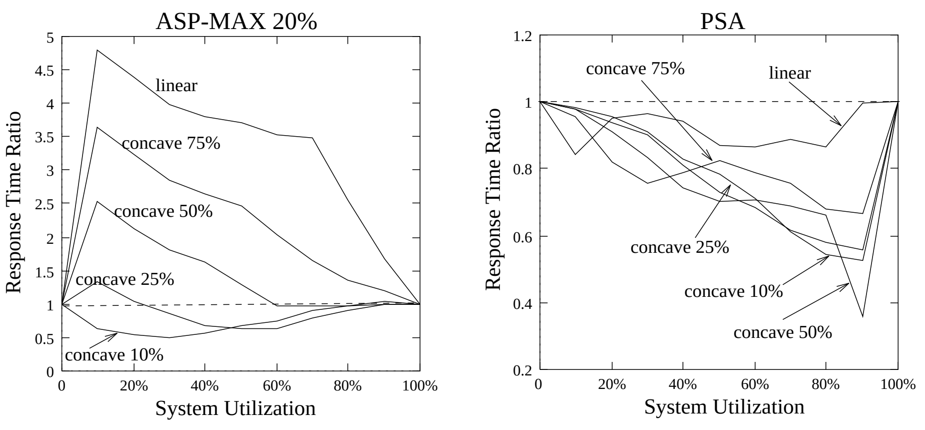

Fig. 1. Response time ratio for the ASP-MAX 20% and PSA policies with respect to the relative work-conserving policy for various workloads.

These workloads can take limited advantage from extra processors. In such cases, it is better to keep idle processors for future arriving jobs. At medium load, the PSA gain over its work conserving counterpart is around 20% and increases up to about 40% as the system utilization increases. Since the amount of processors kept free with ASP-MAX is larger than with PSA, the effects are more dramatic in terms of both losses and gains for all workload types.

3.2 Arrival and Departure Processes

Non-work-conserving policies are expected to perform well in cases when irregularities are present in the workload behavior. To validate such a hypothesis, the coefficients of variation (CV) of the distributions of the job interarrival and execution time are varied. For the first sensitivity analysis experiment, the distribution of the workload execution time is assumed exponential. The impact of various interarrival time distributions is investigated for CV's in the range of [0.15, 25]. The second sensitivity analysis experiment assumes exponential interarrival times for the incoming jobs while the coefficient of variation of the execution time is varied over the range [0.5, 10].

In Figure 2 the response time ratio of the two policies is reported as a function of the coefficient of variation of the arrival process for a fixed system utilization (50%). Although production systems usually operate at high utilization, a medium utilization level was selected to be able to observe the effects of non-work-conserving decisions. Since at high utilization non-work-conserving decisions seldom apply, their effects are more pronounced when the system does not operate close to saturation. For the sake of clarity, three representative speedups were selected out of the five speedups used in the base case experiment.

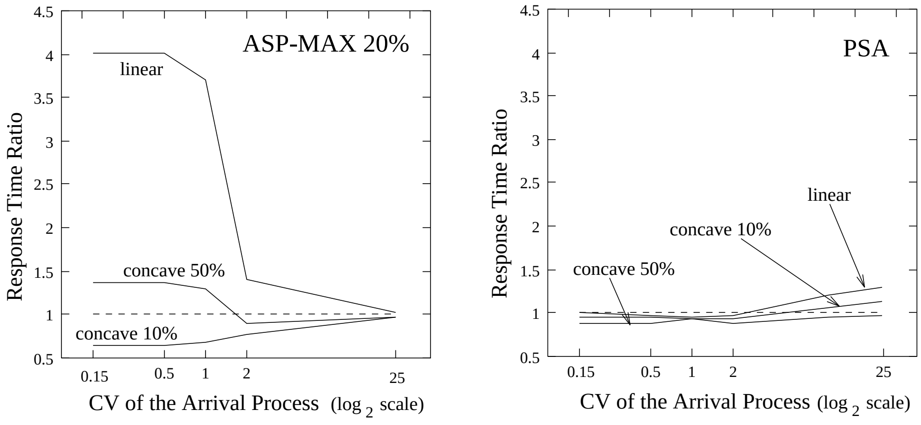

Fig. 2. Response time ratio for the ASP-MAX 20% and PSA policies w.r.t. the corresponding work-conserving policy versus the CV of the interarrival time distribution for various speedups. The system utilization is set to 50%.

As the figure shows, the increase in the arrival process variability (i.e., higher CV's) positively affects performance under the ASP-MAX policy and for workloads that scale well. As the CV of the arrival process increases, ASP-MAX is more likely to outperform the work-conserving case (i.e., dashed reference line). When the CV is small, the arrival process is more regular and the performance of the ASP-MAX policy is not good for workloads that scale well. When the workload scales poorly (flat speedup), the performance is affected in a negative way as the arrival process CV increases. Under the PSA policy the performance is relatively insensitive to the speedup characteristics of the workload. Performance improves (relative to the work-conserving case) as the CV increases but the gain is minimal. For CV=25, the curves start increasing. In this case, the PSA policy is not able to absorb the high variability of the arrival process. The amount of processors saved for future arrivals is not enough to accommodate the incoming requests.

For the second sensitivity experiment, the impact of different computational requirements from the submitted jobs is investigated by changing the coefficient of variation of the execution time distribution over the range $[0.5, 10]$. The arrival process is assumed exponential. In Figure 3 the response time ratios of the ASP-MAX and PSA policies are reported as a function of the CV of the workload execution time. The system utilization is fixed at 50%. Consistent with

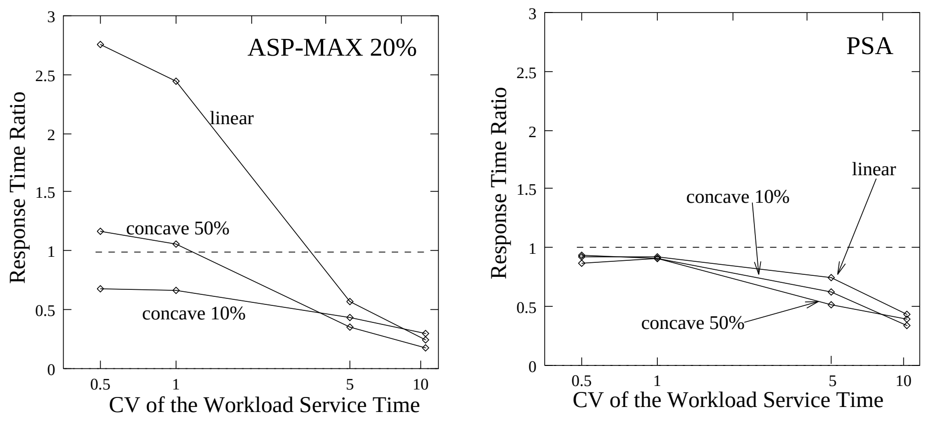

Fig. 3. Response time ratio for the ASP-MAX 20% and PSA policies w.r.t. the corresponding work-conserving policy versus the CV of the execution time distribution ($\log_2$ scale) for various speedups. The system utilization is set to 50%.

the previously observed behavior, the performance of the ASP-MAX policy improves as the CV increases. When the CV is 10, the performance improvement is around 70% with all workload types. Similarly, the PSA policy performance also improves as the CV increases. Unlike ASP-MAX however, PSA does better than its work-conserving counterpart even for small CV's. As before, the performance is relatively insensitive to the speedup characteristics of the workload. The maximum response time improvement over the work-conserving counterpart policy is about 60%.

3.3 Performance Analysis with Arrival Bursts

In this section the policy behavior with respect to bursty arrivals is analyzed. At each arrival time, an arrival burst of a given size or a single arrival can occur with probability p and $1-p$, respectively. Bursts of size 2, 5, and 10, each with probability 0.1, 0.5, and 0.9, are considered. Figure 4 plots the response time ratios for the ASP-MAX policy as a function of the burst size for the three probability values considered. As in the previous experiments, the system utilization is fixed at 50%. With a burst size equal to 0, only single arrivals are allowed. In this experiment, the workload interarrival and service times are ex-

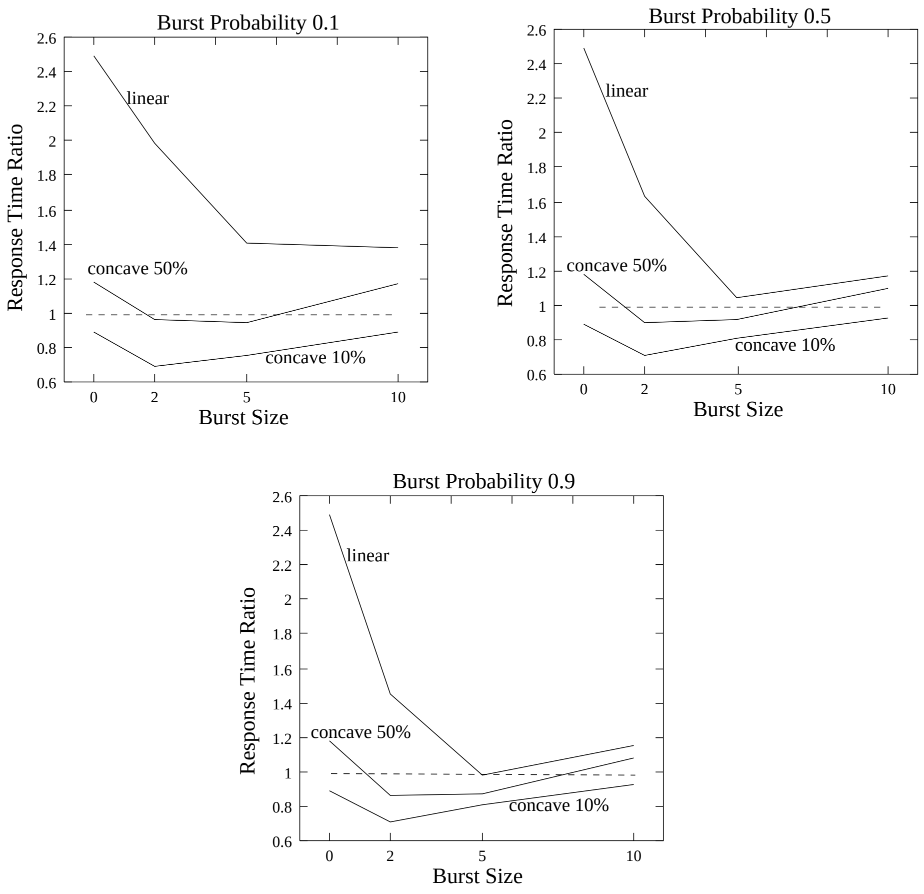

Fig. 4. Response time ratios for the ASP-MAX 20% policy with arrival bursts and different burst probabilities at system utilization 50%.

ponential. A trade-off is observed among the burst size, the burst probability, and the performance improvement. Figure 4 indicates that small burst sizes are absorbed easily by the ASP-MAX policy. The large number of unassigned processors can be employed to handle such bursts. As the bursts become larger, the non-work-conserving choice remains preferable although its advantages are significantly reduced.

The same experiments were conducted with the PSA policy. In Figure 5 the response time ratios at system utilization 50% for the PSA policy as a function

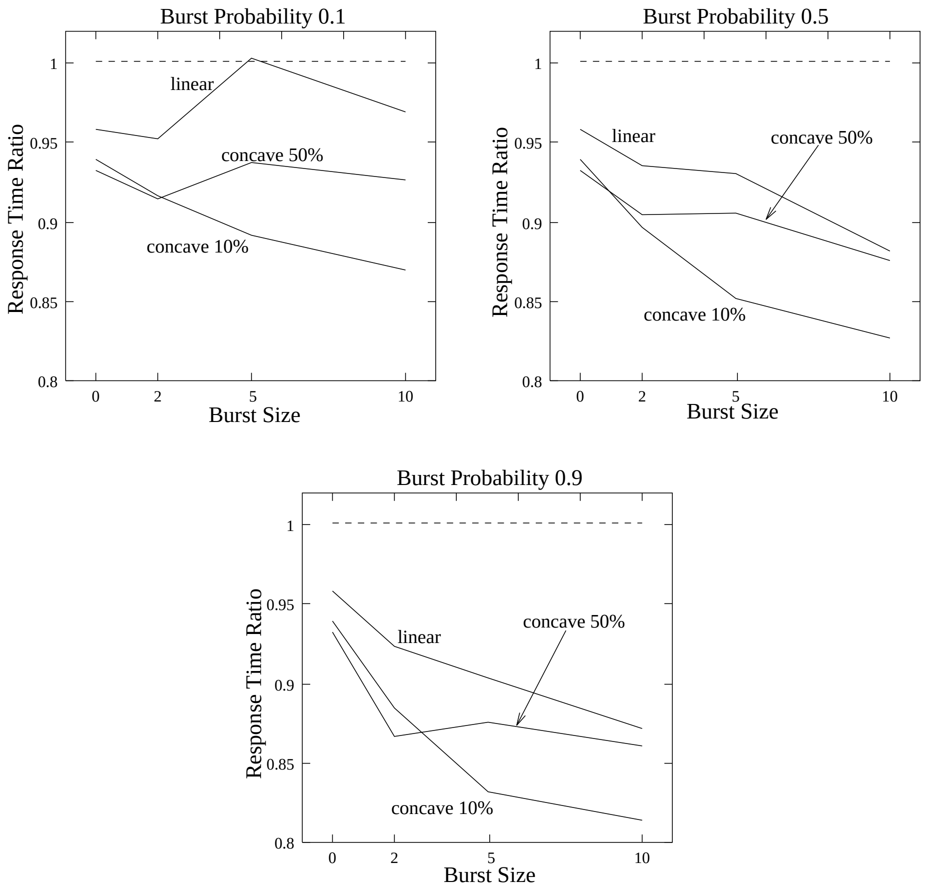

Fig. 5. Response time ratios for the PSA policy with arrival bursts and different burst probabilities at system utilization 50%.

of different burst sizes are reported. The maximum performance improvement is achieved with the largest burst size and burst probability. The trend is similar for all workload types and it is monotonic across the range of burst sizes. The basic conclusion is that as burst size increases, or as the burst probability increases, the effectiveness of non-work-conserving policies improves.

3.4 Performance Analysis with Processor Failures

In this section the system behavior in the presence of processor failures is investigated. If a processor fails while a job is in execution, the job must be restarted on a new set of processors. Two alternatives are possible: the job can be either restarted on the portion of partition that is still operating or can be rescheduled as if it had just entered the system. While the first alternative is not affected by the allocation policy types, the second alternative is expected to yield better performance under a non-work-conserving policy. If an interrupted job is to be rescheduled on a new set of processors, processors left idle during previous scheduling rounds can be effectively used for the job's new allocation.

In this analysis simultaneous multiple processor failures are not allowed. Processors fail according to a Poisson process and return to operating condition after an exponentially distributed repair time. Idle and allocated processors are equally likely to fail. When a processor fails, the number of free processors and the system size are decreased accordingly and the partition size is computed using these new values. If an idle processor fails, it is immediately removed from the set of available processors and it is assigned only after it has been repaired.

In Figure 6 the response time ratios for the ASP and PSA policies are re-

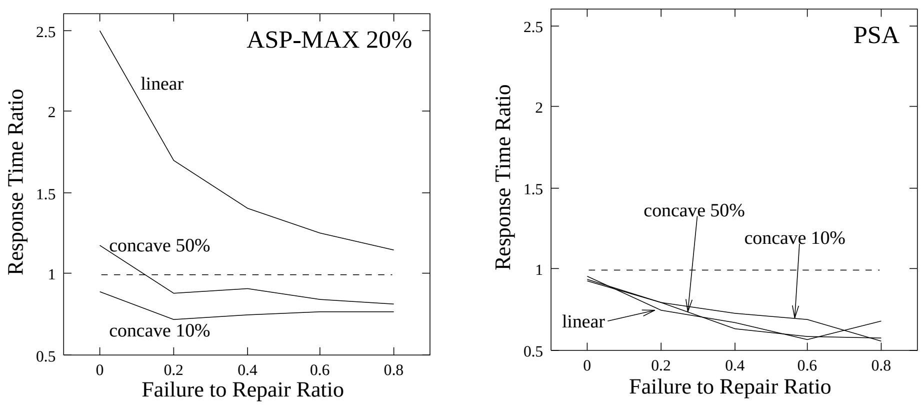

Fig. 6. Response time ratio for the ASP-MAX 20% and PSA policies with processor failures versus the failure to repair rate ratio under the rescheduling strategy for various workload types.

ported as a function of the failure to repair rate ratio for various workload types.

As in the previous experiments, the system utilization is set to 50%. Under the presence of processor failures, both ASP-MAX 20% and PSA perform well consistently. With ASP-MAX 20%, the performance improves for all workload types as the failure to repair ratio increases. The effect is more pronounced for workloads that scale well. With the PSA policy, performance improves across the entire range of failure to repair rate ratio regardless of the workload type. At high ratios, the gains are consistently in the 40% to 50% improvement range.

4 Observations

It has been shown that under irregular workload and system behavior non-work-conserving policies can yield better performance than their work-conserving counterparts. The results suggest that as the workload behavior becomes more irregular, non-work-conserving policies improve performance. Yet, there are specific cases where work-conserving policies are best (e.g., when the arrival process is hypoexponential or the workload scales well).

In Section 2 two distinct non-work-conserving policies are presented. The ASP-MAX policies make non-work-conserving allocations based on the parameter MAX. The PSA policy is non-work-conserving because it never assigns fewer processors than the computed size if such a number is not available. The PSA decisions are made based on the knowledge of previous system states. The two different strategies yield non-work-conserving states of different type and severity. Keeping many processors idle can be useful under certain circumstances. However, under other circumstances, it is better to adjust the number of idle processors according to the system state. To quantify such differences, various non-work-conserving indices are measured. The probability of being in a non-work-conserving state (i.e., the percentage of time spent in a non-work-conserving state), is an overall policy measure that captures the degree to which a policy is non-work-conserving.

Figure 7 illustrates the percentage of time spent in non-work-conserving states for ASP-MAX 20% and PSA as a function of the system utilization for the base case experiment of Section 3.1. As the figure shows, at low loads the ASP policy spends more time than the PSA policy in non-work-conserving states since it keeps many processors idle. At low loads, the PSA policy tends to assign the entire system to a single job rather than keeping some processors idle. However, as the load increases, small numbers of processors are left idle. At medium to high load, few processors can be left idle more frequently since the partition size is neither the largest possible (i.e., the whole system), nor has reached the minimum size (i.e., 1). This accounts for the higher percentage of time at medium load that the PSA policy is observed to be in a non-work-conserving state.

An important factor is the number of processors left idle with respect to the current partition size. If the average partition size is much larger than the number of free processors, the system performance is not hurt. On the other hand, keeping one processor free when the current partition size is 2 processors can seriously impact performance because the system is in a high load situation.

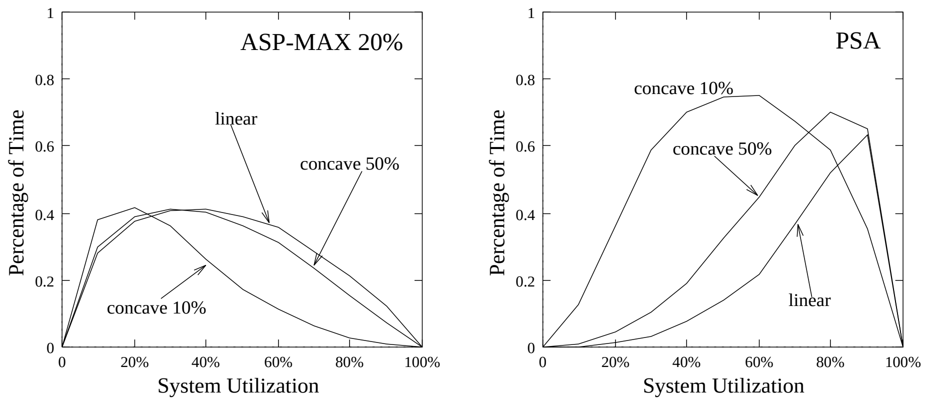

Fig. 7. Percentage of time in non-work-conserving states as a function of the system utilization for the ASP-MAX 20% and PSA policies with various workload types.

A measure that captures this effect is the ratio of the average number of processors free in a non-work-conserving state to the average partition size. Figure 8 illustrates this ratio for the ASP and PSA policies for the various workload types as a function of the system utilization. For the ASP-MAX policy, the ratio

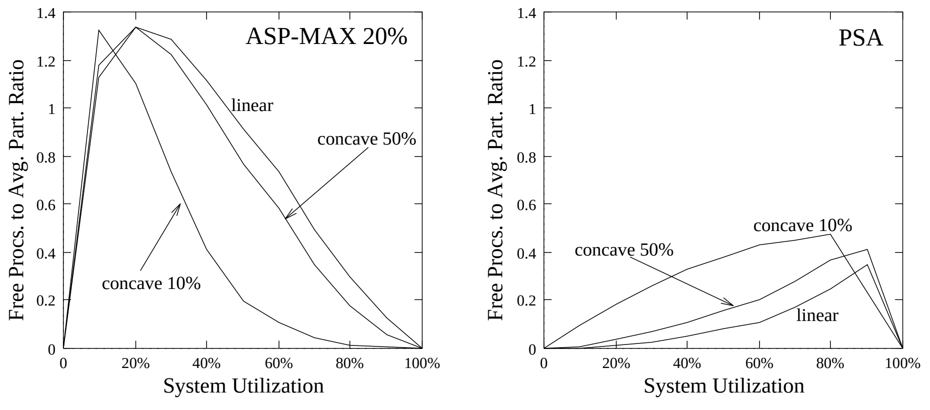

Fig. 8. Ratio of the average number of free processors to the average partition size in non-work-conserving states for the ASP-MAX 20% and PSA policies versus the system utilization for various workload types.

is greater than 1 at medium low loads since the system is likely to have only one or two jobs executing simultaneously. With a maximum allocation of 20% of the system size, the number of idle processors is likely to be larger than the average assignment.

As the system utilization increases, the number of idle processors decreases because of the higher arrival rate. Even though the maximum partition size does not change, more parallel jobs are in execution and less processors are left idle. The opposite trend is observed for the PSA policy. Since the partition size is computed based on the queue length, at low loads the number of free processors relative to the partition size is small. Most jobs execute on the entire system or on half of it. The percentage of free processors relative to the partition size is never greater than 0.5, since only fragments of partitions are kept idle. Overall, Figure 8 indicates that the PSA policy is more conservative than ASP-MAX with respect to non-work-conserving decisions. This is especially true at low to medium system loads. As the system load increases, the PSA policy resorts to non-work-conserving decisions more often.

5 Conclusions

In this paper, the concept of non-work-conserving adaptive space-sharing policies for general purpose multiprocessor systems is presented. Two families of policies are investigated which represent two distinct ways of making non-work-conserving decisions. For the ASP-MAX policy, non-work-conserving decisions are made based on the parameter MAX. The PSA policy "saves" processors according to some knowledge of previous system history. By means of a simulation study, the effectiveness of these policies are analyzed. Conditions are identified under which non-work-conserving policies are useful. Non-work-conserving policies are effective when:

- the workloads that do not scale well (i.e., workload speedup curve is sublinear),
- high variance exists in the arrival process of the workload to the system,
- high variance exists in the workload execution time (e.g., multiclass workloads),
- the workload is susceptible to bursty arrivals, and/or
- the systems is prone to processor failures.

Future work includes experimentation on a real system using real workloads. This would allow the study of the impact of factors such as memory and I/O bandwidth restrictions when non-work-conserving policies are used.

References

[AMV93] R. Agrawal, R.K. Mansharamani, M.K. Vernon, "Response time bounds for parallel processor allocation policies," Technical Report # 1152, Computer Science Dept., Univeristy of Wisconsin, Madison, WI, June 1993.

[CMV94] S.-H. Chiang, R.K. Mansharamani, M.K. Vernon, "Use of application characteristics and limited preemption for run-to-completion parallel processor scheduling policies," *Proc. ACM SIGMETRICS*, 1994, pp. 33-44.

[EZL89] D.L. Eager, J. Zahorjan, E.D. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. on Computers*, Vol 38(3), March 1989, pp. 408-423.

[DCDP90] K. Dussa, B.M. Carlson, L.W. Dowdy, K.-H. Park, "Dynamic partitioning in a transputer environment," *Proc. ACM SIGMETRICS*, 1990, pp. 203-213.

[FR90] D.G. Feitelson, L. Rudolph, "Distributed hierarchical control for parallel processing," *IEEE Computer*, Vol 23(5), May 1990, pp. 65-77.

[GST91] D. Ghosal, G. Serazzi, S.K. Tripathi, "Processor working set and its use in scheduling multiprocessor systems," *IEEE Trans. on Software Engineering*, Vol 17(5), May 1991, pp. 443-453.

[GTU91] A. Gupta, A. Tucker, S. Urushibara, "The impact of operating system scheduling policies and synchronization methods on the performance of parallel applications," *Proc. ACM SIGMETRICS*, 1991, pp. 120-132.

[Int93] Intel Corporation, **Paragon OSF/1 User's Guide**, 1993.

[Klei75] L. Kleinrock, **Queueing Systems**, Vol 1, Wiley Interscience, 1975.

[LV90] S.T. Leutenegger, M.K. Vernon, "The performance of multiprogrammed multiprocessor scheduling policies," *Proc. ACM SIGMETRICS*, 1990, pp. 226-236.

[MEB88] S. Majumdar, D.L. Eager, R.B. Bunt, "Scheduling in multiprogrammed parallel systems," *Proc. ACM SIGMETRICS*, 1988, pp. 104-113.

[MEB91] S. Majumdar, D.L. Eager, R. B. Bunt, "Characterization of programs for scheduling in multiprogrammed parallel systems," *Performance Evaluation*, Vol 13(2), 1991, pp. 109-130.

[MVZ93] C. McCann, R. Vaswani, J. Zahorjan, "A dynamic processor allocation policy for multiprogrammed shared memory multiprocessors," *ACM Trans. on Computer Systems*, Vol 11(2), February 1993, pp. 146-178.

[MZ94] C. McCann, J. Zahorjan, "Processor allocation policies for message-passing parallel computers," *Proc. ACM SIGMETRICS*, 1994, pp. 19-32.

[Oust82] J. Ousterhout, "Scheduling techniques for concurrent systems," *Proc. 3rd International Conference on Distributed Computing Systems*, 1982, pp. 22-30.

[PD89] K.-H. Park, L.W. Dowdy, "Dynamic partitioning of multiprocessor systems," *International Journal of Parallel Programming,* Vol 18(2), 1989, pp. 91-120.

[RSDSC94] E. Rosti, E. Smirni, L.W. Dowdy, G. Serazzi, B.M. Carlson, "Robust partitioning policies for multiprocessor systems," *Performance Evaluation*, Vol 19(2-3), March 1994, pp. 141-165.

[SST93] S.K. Setia, M.S. Squillante, S.K. Tripathi, "Processor scheduling in multiprogrammed, distributed memory parallel computers," *Proc. ACM SIGMETRICS*, 1993, pp. 158-170.

[Sev89] K.C. Sevcik, "Characterization of parallelism in applications and their use in scheduling," *Proc. ACM SIGMETRICS*, 1989, pp. 171-180.

[Sev94] K.C. Sevcik, "Application scheduling and processor allocation in multiprogrammed multiprocessors," *Performance Evaluation*, Vol 19(2-3), March 1994, pp. 107-140.

[SRDS93] E. Smirni, E. Rosti, L.W. Dowdy, G. Serazzi, "Evaluation of multiprocessor allocation policies," Tech. Report, Computer Science Dept., Vanderbilt University, Nashville, TN, August 1993.

[SRSDS95] E. Smirni, E. Rosti, G. Serazzi, L.W. Dowdy, K.C. Sevcik, "Performance gains from leaving idle processors in multiprocessor systems," to appear in *International Conference on Parallel Processing.*

[TG89] A. Tucker, A. Gupta, "Process control and scheduling issues for multiprogrammed shared-memory multiprocessors," *Proc. of the 12th ACM Symposium on Operating Systems Principles*, 1989, pp. 159-166.

[ZM90] J. Zahorjan, C. McCann, "Processor scheduling in shared memory multiprocessors," *Proc. ACM SIGMETRICS*, 1990, pp. 214-225.

[ZB91] S. Zhou, T. Brecht, "Processor pool-based scheduling for large-scale NUMA multiprocessors," *Proc. ACM SIGMETRICS*, 1991, pp. 133-142.

This article was processed using the LaTeX macro package with LLNCS style