RC23416 (W0411-054) November 8, 2004
Mathematics

IBM Research Report

Stochastic Analysis of Scheduling in Parallel Processing Systems

Mark S. Squillante
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Stochastic Analysis of Scheduling in Parallel Processing Systems

Mark S. Squillante
Mathematical Sciences Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA
mss@watson.ibm.com

This paper is dedicated to Professor Ken Sevcik on the occasion of his 60th birthday.

Abstract

We derive an exact matrix-analytic analysis of a general stochastic model of parallel processing systems under dynamic spacesharing, which yields a closed-form solution for certain model instances. An analysis of a general nearly completely decomposable model of parallel program memory reference behavior is also derived, which provides measures of the distribution of the memory overhead incurred by a parallel program as a function of its server allocation. These theoretical results can be exploited to investigate the design and performance space of parallel processing systems with respect to fundamental tradeoffs related to the server and memory allocation strategies and their interactions.

1 Introduction

Stochastic modeling and related queueing-theoretic results have played a fundamental role in the design of scheduling strategies of both theoretical and practical interest. This has been especially the case in single-server queues; refer to [16, 38, 45, 19, 58] and the references cited therein. Some important principles have also been derived for parallel-server queues, but these results have been restricted to certain parallel processing systems, such as those with customers having serial service demands (e.g., [5, 57, 51, 4, 40]) or parallel fork-join service demands (e.g., [28, 23, 24, 44, 52]).

With the significant advances of computer and communication technology and the considerable development of parallel scientific and engineering applications, several classes of scheduling strategies have emerged over time each differing in the way the parallel servers are shared among the parallel jobs submitted for execution. Our present study seeks to derive stochastic modeling and related queueing-theoretic results for some of these classes of scheduling strategies. Of particular theoretical and practical interest, and the focus of this paper, is the class of spacesharing strategies that share the parallel servers in space by partitioning them among different parallel jobs. This is based on the widespread use of spacesharing in parallel processing systems, possibly in combination with other strategies [47, 15]. We do not consider the classes of scheduling strategies that combine spacesharing with different forms of timesharing, where the servers

are shared by rotating them among a set of jobs in time. See [43, 53, 41, 55] and the references therein for a stochastic analysis of various scheduling strategies based on both spacesharing and timesharing.

A number of empirical studies have demonstrated the benefits and superiority of dynamic spacesharing over other forms of spacesharing strategies in single-class parallel processing systems where jobs are statistically identical and have sublinear speedup characteristics, and no prior knowledge of service time characteristics is available or used in scheduling decisions; refer to [31, 13, 59, 25, 26, 18] and the references cited therein. We therefore focus on this important class of scheduling strategies in which the number of servers allocated to a parallel job can be adjusted dynamically throughout its execution in response to changes in the state of the system. In particular, dynamic spacesharing tends to decrease the number of servers allocated to jobs with increasing system loads in order to increase the efficiency of parallel execution at heavier traffic intensities. The assumption that parallel applications are malleable, in the sense that their server allocations can be changed at any time, provides the greatest flexibility with respect to the scheduling of such applications. (See [47, 15] and the references therein for a discussion on other types of parallel application models.) This flexibility and its benefits, however, are gained at some cost as the reallocation, or reconfiguration, of the servers among a set of jobs under dynamic spacesharing causes the system to incur various overhead penalties. In fact, the fundamental parallel scheduling tradeoff between the benefits of dynamic spacesharing and its costs is central to the design and performance of dynamic scheduling in parallel processing systems.

The memory requirements of parallel applications are another critical factor in the design and performance of dynamic scheduling in parallel processing systems; refer to [35, 42, 27, 33, 32] and the references cited therein. More specifically, the working set size of a parallel job on each of the servers allocated to it increases with reductions in the size of this server allocation, since the job's memory requirements are spread over fewer servers. Reducing the number of servers allocated to a sufficiently large job below a certain point can cause excessive memory overhead as the job's working set on each server no longer fits in the server's local memory, thus increasing the number of page faults incurred by the job. While dynamic spacesharing strategies decrease the server allocation size with increasing system loads, memory considerations in parallel processing systems tend to argue in the opposite direction. Hence, there is a fundamental tradeoff between the impact of program efficiency towards allocating fewer servers to jobs at heavier traffic intensities and the performance impact of memory requirements towards larger allocations. This tradeoff is also central to the design and performance of dynamic scheduling in parallel processing systems. We therefore derive stochastic modeling and related queueing-theoretic results on the memory requirements of parallel applications at each of the servers allocated to them in our analysis of dynamic spacesharing.

In this paper we formulate a general stochastic model of parallel processing systems under dynamic spacesharing. An exact matrix-analytic analysis of this model is derived, which yields closed-form solu-

tions under certain instances of the model based on their probabilistic structure. We also derive expressions for several measures of interest, including the tail distribution of the queue length process and its asymptotic decay rate, the mean sojourn time of parallel jobs, and the long-run proportion of time the system spends reconfiguring the allocation of servers among the jobs. These stochastic modeling and analysis results generalize and extend those presented in [50]. In this paper we also formulate and derive an analysis of a general nearly completely decomposable model of parallel program memory reference behavior that introduces and analyzes the inter-locality miss ratio which includes the memory overhead caused by the reloading of pages upon return visits to localities (especially in iterative programs), as well as a more accurate representation of the page faults due to transitions between localities. This stochastic analysis solves a fundamental problem with previous nearly completely decomposable models and analyses that remained open since the early 1970s. The results of our analysis are then used to construct a probability distribution for the total memory overhead incurred by a parallel program as a function of its server allocation, which can be directly incorporated in our analysis of parallel processing systems under dynamic spacesharing. These stochastic modeling and analysis results generalize and extend those presented in [35].

The remainder of the paper is organized as follows. In Section 2 we present our parallel processing system model. A matrix-analytic analysis of this model and a nearly completely decomposable analysis of the corresponding memory overheads are derived in Sections 3 and 4, respectively. Our concluding remarks are provided in Section 5. Throughout this paper we shall use $\mathbb{Z}^+$ and $\mathbb{Z}_+$ to denote the set of positive and non-negative integers, respectively, with $\mathbb{R}^+$ and $\mathbb{R}_+$ denoting the corresponding subsets of the real numbers. Furthermore, the term partition will be used to refer to a (disjoint) set of servers allocated to a single parallel job.

2 Model of Parallel Processing Systems

Consider a parallel processing system that consists of P identical servers, each having its own local memory, under a single-class parallel application workload with sublinear speedup characteristics. The servers are scheduled according to a dynamic spacesharing strategy that attempts to equally allocate the servers among a set of parallel jobs, where no information about service time characteristics is used in making scheduling decisions. Let U denote the minimum number of servers allocated to any parallel job, and therefore the maximum number of server partitions is given by $N = \lfloor P/U \rfloor$. Upon the arrival of a job when the system is executing i jobs, $0 \leq i < N$, the P servers are repartitioned among the set of $i+1$ jobs such that each job is allocated a server partition either of size $\lfloor P/(i+1) \rfloor$ or of size $\lceil P/(i+1) \rceil$. Arrivals that find $i \geq N$ jobs in the system are placed in a first-come first-serve infinite-capacity queue to wait until a server partition becomes available. When one of the i jobs in the system departs, $2 \leq i \leq N$, the system reconfigures the server allocations among the remaining $i-1$ jobs such that there are only partitions of sizes $\lfloor P/(i-1) \rfloor$ and

$\lceil P/(i-1) \rceil$. A departure when the system contains more than N jobs simply causes the job at the head of the queue to be allocated the available server partition, and no reconfiguration is performed. Every parallel job is executed to completion without interruption and all of the servers in the partition are reserved by the application throughout this duration.

Parallel jobs are assumed to arrive from an exogenous source according to a Markovian Arrival Process (MAP) [30, 20] having descriptors $(\mathcal{S}_0^A, \mathcal{S}_1^A)$ of order $m^A < \infty$ with (positive) mean rate $\lambda = (\mathbf{x}\mathcal{S}_1^A\mathbf{e})^{-1} < \infty$, where $\mathbf{e}$ denotes the column vector of appropriate dimension containing all ones and $\mathbf{x}$ is the invariant probability vector of the generator $\mathcal{S}^A = \mathcal{S}_0^A + \mathcal{S}_1^A$, i.e., the (unique) solution of $\mathbf{x}\mathcal{S}^A = \mathbf{0}$ and $\mathbf{x}\mathbf{e} = 1$. (Our results can be easily extended to handle a batch MAP (BMAP) arrival process using standard methods [30, 20, 50].) The parallel service times depend upon the number of servers allocated to the jobs. When executed on partitions of size p, the service times of the parallel jobs are assumed to be independent and identically distributed (i.i.d.) following an order $m_p^B < \infty$ phase-type distribution [29, 20] with parameters $(\underline{\beta}_p, \mathcal{S}_p^B)$ and (positive) mean $\mu_p^{-1} = -\underline{\beta}_p(\mathcal{S}_p^B)^{-1}\mathbf{e} < \infty$, $p \in \{U, \ldots, P\}$. The times required to repartition the servers among the i jobs to be executed (either due to a departure when the system contains $i+1$ jobs or an arrival when the system contains $i-1$ jobs) are assumed to be i.i.d. according to a phase-type distribution having parameters $(\underline{\zeta}_i, \mathcal{S}_i^C)$ of order $m_i^C < \infty$ with (positive) mean $\gamma_i^{-1} = -\underline{\zeta}_i(\mathcal{S}_i^C)^{-1}\mathbf{e} < \infty$, $i \in \{1, \ldots, N\}$. All of the above stochastic processes are assumed to be mutually independent.

The execution behavior of the parallel jobs on each possible server allocation is an important aspect of any parallel processing model. Our formulation takes the general approach of modeling the execution behavior of the parallel jobs by a separate probability distribution for each possible number of servers p, $p \in \{U, \ldots, P\}$. This supports the inclusion of any factors that may affect the execution behavior of the parallel applications of interest (e.g., see [14, 46, 22, 47, 36, 37] and the references therein), especially given the known closure properties of phase-type distributions such as the convolution of a finite number of phase-type distributions being phase-type [29, 20]. The only requirement we have is that there exists a probabilistic mapping from the residual life of the service demands on each server allocation at the point of a reconfiguration to any other possible server allocation within the same collection of phase-type distributions. Using this mapping, the details of the relatively complex server allocation decisions that can be made by the dynamic spacesharing strategy in each case, as well as the overheads of making these decisions and of reconfiguring the applications involved, are captured in the probability distributions and the state transitions of the corresponding stochastic process. Moreover, the memory overheads incurred by a parallel job as a function of the dynamic scheduling decisions can be captured directly in the corresponding service time distributions as described in Section 4.

The use of MAPs and phase-type distributions for all model parameters is of theoretical importance in that we exploit their properties to derive an exact solution of our general parallel processing model. It is

also of practical importance in that, since the class of phase-type distributions is dense within the set of probability distributions on $[0, \infty)$, and since the class of MAPs provides a general framework for capturing the dependence structure and variability of a process, any stochastic process on this space for the parallel processing systems of interest can in principle be represented arbitrarily closely by a MAP or a phase-type distribution. Moreover, a considerable body of research has examined the fitting of phase-type distributions and MAPs to empirical data, and a number of algorithms have been developed for doing so; refer to [1, 21] and the references cited therein. This includes recent work that has considered effectively approximating long-range dependent and heavy-tailed behaviors with instances of the classes of MAPs and phase-type distributions in order to analyze performance models. By appropriately setting the parameters of our model, a wide range of parallel application and system environments can be investigated [47, 15]. In particular, the subexponential tails of the marginal interarrival distribution and the dependence structure of the interarrival process found in some parallel processing systems [54] can be captured by a MAP. Furthermore, most of the results of our analysis in the next section will continue to hold when the order of the MAP and phase-type distributions m^A, m_p^B and m_i^C are infinite, under a proper formulation.

3 Analysis of Dynamic Spacesharing

The parallel processing model of the previous section can be represented by a continuous-time stochastic process $\{X(t)\,;\, t \in \mathbb{R}_+\}$ on the state space given by $\Omega = \bigcup_{i=0}^{\infty} \Omega_i$ where $\Omega_0 \equiv \{(0, \overline{j}^A) \mid \overline{j}^A \in \omega^A\}$, $\Omega_i \equiv \{(i, \overline{j}^A, \overline{j}^B_{-,\ell}, \overline{j}^B_{+,\ell}, j_\ell^C) \mid i \in \mathbb{Z}^+, \overline{j}^A \in \omega^A, \overline{j}^B_{w,\ell} \in \omega^B_{w,\ell}, w \in \{-,+\}, \ell = \min\{i, N\}, j_\ell^C \in \{0, \ldots, m_\ell^C\}\}$, $\omega^A \equiv \{(j_c^A, j_h^A, j_p^A) \mid j_c^A \in \{1, \ldots, m_c^A\}, j_h^A \in \{1, \ldots, m_h^A\}, j_p^A \in \{1, \ldots, m_p^A\}\}$, $\omega^B_{w,\ell} \equiv \{(j^B_{w,\ell,1}, j^B_{w,\ell,2}, \ldots, j^B_{w,\ell,m^B_{\hat{n}}}) \mid j^B_{w,\ell,k} \in \{0, \ldots, \mathcal{P}^\ell_w\}, k \in \{1, \ldots, m^B_{\hat{n}}\}, \sum_{k=1}^{m^B_{\hat{n}}} j^B_{w,\ell,k} = \mathcal{P}^\ell_w, \hat{n} = n^\ell_w, w \in \{-,+\}\}$, $n^\ell_+ \equiv \lceil P/\ell \rceil$, $n^\ell_- \equiv \lfloor P/\ell \rfloor$, $\mathcal{P}^\ell_+ \equiv \text{mod}\,(P/\ell)$, $\mathcal{P}^\ell_- \equiv \ell - \mathcal{P}^\ell_+$, for $\ell \in \{1, \ldots, N\}$. The state-vector variable i denotes the total number of parallel jobs in the system; $\overline{j}^A$ denotes the state of the MAP arrival process; $j^B_{w,\ell,k}$ denotes the number of jobs executing on n^ℓ_w servers whose service time process is in phase k; and j_ℓ^C denotes the phase of the reconfiguration overhead process when $j_\ell^C \in \{1, \ldots, m_\ell^C\}$ and it indicates no reconfiguration when $j_\ell^C = 0$ (i.e., the ℓ parallel jobs are executing). Moreover, n^ℓ_- and n^ℓ_+ represent the two partition sizes when the system contains i jobs, for $\ell = \min\{i, N\}$, and $\mathcal{P}^\ell_w$ denotes the number of such partitions of size n^ℓ_w, for $w \in \{-,+\}$.

Let $x_{i,d} \in \Omega_i$, $d \in \{1, \ldots, D_i\}$, $i \in \mathbb{Z}_+$, be a lexicographic ordering of the elements of Ω_i, and define $D \equiv \sum_{i=0}^{N-1} D_i$, where $D_i = |\Omega_i|$. The set Ω_i is also called the ith level of the process. We then define $\boldsymbol{\pi} \equiv (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$, $\boldsymbol{\pi}_i \equiv (\pi(x_{i,1}), \pi(x_{i,2}), \ldots, \pi(x_{i,D_i}))$ and $\pi(x_{i,d}) \equiv \lim_{t\to\infty} \mathbb{P}[X(t) = x_{i,d}]$, $i \in \mathbb{Z}_+$, $x_{i,d} \in \Omega_i$, $d \in \{1, \ldots, D_i\}$. The limiting probability vector $\boldsymbol{\pi}$ is the stationary distribution for

the stochastic process $\{X(t)\,;\, t \in \mathbb{R}_+\}$. Assuming this process to be irreducible and positive recurrent, the invariant probability vector is uniquely determined by solving the global balance equations $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ together with the normalizing constraint $\boldsymbol{\pi}\mathbf{e} = 1$, where $\mathbf{Q}$ is the infinitesimal generator matrix for the process.

The generator matrix $\mathbf{Q}$, organized in the same order as the elements of $\boldsymbol{\pi}$, has a structure given by

$$\mathbf{Q} = \begin{bmatrix} \mathbf{B}_{00} & \mathbf{B}_{01} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}_{10} & \mathbf{B}_{11} & \mathbf{A}_0 & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{1}$$

where $\mathbf{B}_{00}$, $\mathbf{B}_{01}$, $\mathbf{B}_{10}$, $\mathbf{B}_{11}$ and $\mathbf{A}_n$, $n = 0, 1, 2$, are finite matrices of dimensions $D \times D$, $D \times D_N$, $D_N \times D$, $D_N \times D_N$ and $D_N \times D_N$, respectively. The matrices corresponding to the non-homgeneous boundary of the state space Ω have the structures

$$\mathbf{B}_{00} = \begin{bmatrix} \Psi_0 & \Lambda_0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \Phi_1 & \Psi_1 & \Lambda_1 & \mathbf{0} & \cdots & & \\ \mathbf{0} & \Phi_2 & \Psi_2 & \Lambda_2 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \Phi_{N-1} & \Psi_{N-1} \end{bmatrix}, \quad \mathbf{B}_{01} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \Lambda_{N-1} \end{bmatrix}, \tag{2}$$

$$\mathbf{B}_{10} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \Phi_N \end{bmatrix}, \qquad \mathbf{B}_{11} = \Psi_N, \tag{3}$$

where Φ_i, Ψ_i and Λ_i have dimensions $D_i \times D_{i-1}$, $D_i \times D_i$ and $D_i \times D_{i+1}$, respectively. The matrix Φ_i defines the transitions from states in Ω_i to states in Ω_{i-1}, $i \in \{1, \ldots, N\}$, Λ_i defines the transitions from states in Ω_i to states in Ω_{i+1}, $i \in \{0, \ldots, N-1\}$, and the off-diagonal elements of Ψ_i define the transitions between states within Ω_i, $i \in \{0, \ldots, N\}$. These level-dependent matrices define the exact server allocation behavior of the dynamic spacesharing strategy under consideration, the arrival, service and reconfiguration processes of the workload being modeled, and the various interactions of each of these aspects of the system. The same holds true for the $\mathbf{A}$ matrices with respect to the homogeneous portion of the state space.

Let $\mathrm{sp}(\mathbf{C})$ denote the spectral radius of a matrix $\mathbf{C}$ and let $\mathbf{I}$ denote the identity matrix of appropriate dimension. The stationary probability vector $\boldsymbol{\pi}$ of the process $\{X(t)\,;\, t \in \mathbb{R}_+\}$ with generator $\mathbf{Q}$ then can be obtained from the following standard matrix-analytic results (see Theorems 1.7.1 and 1.5.1 in [29]).

Theorem 3.1. *Let $\mathbf{Q}$ be irreducible and in the form of (1). This stochastic process is positive recurrent if and only if sp$(\mathbf{R}) < 1$, where $\mathbf{R}$ is the minimal non-negative matrix that satisfies*

$$\mathbf{R}^2\mathbf{A}_2 + \mathbf{R}\mathbf{A}_1 + \mathbf{A}_0 = \mathbf{0}. \tag{4}$$

Furthermore, there exists a positive vector $(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$ *such that*

$$(\boldsymbol{\pi}_0,\, \boldsymbol{\pi}_1,\, \ldots,\, \boldsymbol{\pi}_N) \begin{bmatrix} \mathbf{B}_{00} & \mathbf{B}_{01} \\ \mathbf{B}_{10} & \mathbf{B}_{11} + \mathbf{R}\mathbf{A}_2 \end{bmatrix} \;=\; \mathbf{0}. \tag{5}$$

The remaining components of the invariant probability vector $\boldsymbol{\pi}$ *are then given by*

$$\boldsymbol{\pi}_{N+n} \;=\; \boldsymbol{\pi}_N \mathbf{R}^n, \quad n \in \mathbb{Z}_+, \tag{6}$$

where $(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$ *is normalized by*

$$(\boldsymbol{\pi}_0,\, \boldsymbol{\pi}_1,\, \ldots,\, \boldsymbol{\pi}_{N-1})\, \mathbf{e} \;+\; \boldsymbol{\pi}_N (\mathbf{I} - \mathbf{R})^{-1} \mathbf{e} \;=\; 1. \tag{7}$$

The time and space complexities of computing equations (5) and (7) can be significantly reduced with the use of the following matrix-analytic results (refer to [49, 17, 55]).

Theorem 3.2. *Let* $\mathbf{Q}$ *be irreducible and in the form of (1) through (3). If the minimal non-negative solution* $\mathbf{R}$ *of equation (4) satisfies sp*$(\mathbf{R}) < 1$*, and if there exists a positive probability vector* $(\boldsymbol{\pi}_0, \ldots, \boldsymbol{\pi}_N)$ *satisfying equation (5), then the components of this probability vector are given by*

$$\boldsymbol{\pi}_n \;=\; -\,\boldsymbol{\pi}_{n+1}\, \Phi_{n+1}\, \widetilde{\mathbf{R}}_n^{-1}, \qquad n \in \{0, \ldots, N-1\}, \tag{8}$$

$$\boldsymbol{\pi}_N \;=\; -\,\boldsymbol{\pi}_{N-1}\, \Lambda_{N-1}\, \widetilde{\mathbf{R}}_N^{-1}, \tag{9}$$

where $\widetilde{\mathbf{R}}_0 \equiv \Psi_0$*,* $\widetilde{\mathbf{R}}_n \equiv \Psi_n - \Phi_n \widetilde{\mathbf{R}}_{n-1}^{-1} \Lambda_{n-1}$*,* $n \in \{1, \ldots, N-1\}$*,* $\widetilde{\mathbf{R}}_N \equiv \Psi_N + \mathbf{R}\mathbf{A}_2$*. Furthermore, when* $N > 1$*, the vector* $\boldsymbol{\pi}_{N-1}$ *can be determined up to a multiplicative constant by solving*

$$\boldsymbol{\pi}_{N-1} \left[\Psi_{N-1} \;-\; \Phi_{N-1}\, \widetilde{\mathbf{R}}_{N-2}^{-1}\, \Lambda_{N-2} \;-\; \Lambda_{N-1} \widetilde{\mathbf{R}}_N^{-1}\, \Phi_N \right] \;=\; \mathbf{0}, \tag{10}$$

$$\boldsymbol{\pi}_{N-1}\, \mathbf{e} \;=\; \theta, \qquad \theta \;>\; 0. \tag{11}$$

Otherwise, when $N = 1$*, the vector* $\boldsymbol{\pi}_1$ *can be determined up to a multiplicative constant by solving*

$$\boldsymbol{\pi}_1 \left[\widetilde{\mathbf{R}}_1 \;-\; \Phi_1\, \widetilde{\mathbf{R}}_0^{-1}\, \Lambda_0 \right] \;=\; \mathbf{0}, \tag{12}$$

$$\boldsymbol{\pi}_1\, \mathbf{e} \;=\; \theta, \qquad \theta \;>\; 0. \tag{13}$$

In either case, the vector $(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$ *then can be obtained from (8), (9) and the normalizing equation (7).*

3.1 Irreducibility and Stability Criterion

While the irreducibility of the stochastic process $\{X(t)\,;\, t \in \mathbb{R}_+\}$ is often obvious, it can be verified for any specific instance of our model by determining whether the first $N+2$ levels of the state space (the boundary plus the first level of the homogeneous portion) and the corresponding transitions among these states, when viewed as a directed graph, is strongly connected. This follows directly from the fact that transitions between the states of levels $N+1$ and $N+2$ are identical to those between the states of levels N and $N+1$, and the fact that the process is skip-free to the right and to the left with respect to levels. Hence, upon verifying that the non-homogeneous boundary plus the first level of the homogeneous portion is irreducible, it follows that the entire process is irreducible.

The stability conditions for the system are provided in Theorem 3.1. When the generator matrix $\mathbf{A} \equiv \mathbf{A}_0 + \mathbf{A}_1 + \mathbf{A}_2$ is irreducible, Neuts [29] determined the following necessary and sufficient conditions for $\text{sp}(\mathbf{R}) < 1$.

Theorem 3.3. *Let the generator* $\mathbf{A}$ *be irreducible. Then* $sp(\mathbf{R}) < 1$ *if and only if*

$$\mathbf{y}\mathbf{A}_0\mathbf{e} \;<\; \mathbf{y}\mathbf{A}_2\mathbf{e}, \tag{14}$$

where $\mathbf{y}$ *is the stationary probability vector of* $\mathbf{A}$.

3.2 Special Case: Exponential Model Parameters

We now consider instances of the model where U evenly divides P and the interarrival time, service time and reconfiguration overhead distributions are all exponential, and obtain closed-form expressions for the elements of the $\mathbf{R}$ matrix satisfying (4), $p \in \{U, \ldots, P\}$, $i \in \{1, \ldots, N\}$. In this case, we have $\mathbf{A}_0 = \lambda\mathbf{I}$, $\mathbf{A}_1 = \begin{bmatrix} -(\lambda+\mu_N) & 0 \\ \gamma_N & -(\lambda+\gamma_N) \end{bmatrix}$ and $\mathbf{A}_2 = \begin{bmatrix} \mu_N & 0 \\ 0 & 0 \end{bmatrix}$. Note further that the generator $\mathbf{Q}$ has the form given in (1) with $D_0 = 1$, $D_i = 2$, $i \in \{1, \ldots, N\}$, and $D = 2N-1$.

The probabilistic significance of the element $r_{u,v}$ of the matrix $\mathbf{R} \equiv [r_{u,v}]_{u,v\in\{0,1\}}$ is the expected amount of time spent in the state $(i+1, v)$ before the first return to any state of level i, expressed in units of the mean sojourn time for the state (i, u), given that the process started in state (i, u) [29], $i \geq N$. Upon uniformizing $\{X(t)\,;\, t \in \mathbb{R}_+\}$ to obtain an equivalent discrete-time version $\{X_t^0\,;\, t \in \mathbb{Z}_+\}$ with the corresponding $\mathbf{R}^0 \equiv [r^0_{u,v}]_{u,v\in\{0,1\}}$, where

$$r_{u,v} \;=\; \left(\frac{r^0_{u,v}}{\delta_{v,v}}\right) \delta_{u,u}, \quad u, v \in \{0, 1\}, \tag{15}$$

and $\Delta \equiv [\delta_{u,v}]_{u,v\in\{0,1\}} = -\mathbf{diag}(\mathbf{A}_1)$, the probabilistic interpretation of the element $r^0_{u,v}$ is the expected number of visits to the state $(i+1, v)$ before the first return to any state of level i, given that the process

started in state (i,u) [29], $i \geq N$. We then have the following result for the exponential case of our parallel processing model.

Theorem 3.4. *Let* $\mathbf{Q}$ *be irreducible and in the form of (1), under exponential distributions for the model parameters and with* U *evenly dividing* P*. Then this stochastic process is positive recurrent if and only if*

$$\frac{\lambda}{\mu_N} < 1, \tag{16}$$

and the elements of the minimal non-negative matrix $\mathbf{R}$ *satisfying (4) are given by*

$$r_{0,0} = \frac{\lambda}{\mu_N}, \tag{17}$$

$$r_{0,1} = 0, \tag{18}$$

$$r_{1,0} = \frac{\lambda}{\mu_N}, \tag{19}$$

$$r_{1,1} = \frac{\lambda}{\lambda + \gamma_N}. \tag{20}$$

Proof: Equation (18) follows directly from the probabilistic interpretation of $r_{0,1}$ and the fact that there are no paths from the states $(i,0)$ to the states $(u,1)$, $N \leq i < u$, which do not first visit a state of level N. Consider the uniformized process $\{X_t^0\,;\, t \in \mathbb{Z}_+\}$ starting in state $(i,0)$, $i > N$, observing that the only transitions affecting $r_{0,0}^0$ in this process are from $(v,0)$ to $(v+1,0)$ with probability $\lambda(\lambda+\mu_N)^{-1}$ and from $(v,0)$ to $(v-1,0)$ with probability $\mu_N(\lambda+\mu_N)^{-1}$, $v \geq i$. The expected number of visits to $(i+1,0)$ before the first return to $(i,0)$ in this case is then given by $\mu_N(\lambda+\mu_N)^{-1}\sum_{n=0}^{\infty} n(\lambda/(\lambda+\mu_N))^n$, and thus $r_{0,0}^0 = \lambda/\mu_N$. From (15) we have $r_{0,0} = r_{0,0}^0$, which yields equation (17). Equations (19) and (20) then follow by direct calculation upon substituting (17) and (18) into equation (4). The matrix $\mathbf{R}$ is lower triangular, and thus its eigenvalues are given by its diagonal elements. Since $r_{1,1}$ is always less than 1 with $\gamma_N > 0$, it follows from Theorem 3.1 that the stability criterion is given by (16). □

Intuitively, the condition $\lambda < \mu_N$ ensures a positive recurrent process because, given $\gamma_N > 0$, the expected drift of the system in the homogeneous portion of the state space is toward level N when $\lambda < \mu_N$.

3.3 Performance Measures

Using the components of the invariant vector $\boldsymbol{\pi}$, we can obtain various performance measures of interest. In particular, the tail distribution of the queue length process can be expressed as

$$\mathbb{P}[Q > z] \;=\; \begin{cases} \displaystyle\sum_{n=z+1}^{N-1} \boldsymbol{\pi}_n \mathbf{e} + \sum_{n=N}^{\infty} \boldsymbol{\pi}_n \mathbf{e} \;=\; \sum_{n=z+1}^{N-1} \boldsymbol{\pi}_n \mathbf{e} + \boldsymbol{\pi}_N (\mathbf{I}-\mathbf{R})^{-1}\mathbf{e}, & 0 \le z < N-1, \\ \displaystyle\sum_{n=z+1}^{\infty} \boldsymbol{\pi}_n \mathbf{e} \;=\; \boldsymbol{\pi}_N \mathbf{R}^{z+1-N} (\mathbf{I}-\mathbf{R})^{-1}\mathbf{e}, & z \ge N-1, \end{cases} \tag{21}$$

with the corresponding expectation given by

$$\mathbb{E}[Q] \;=\; \sum_{n=1}^{N-1} n\boldsymbol{\pi}_n \mathbf{e} + \sum_{n=0}^{\infty} (N+n)\boldsymbol{\pi}_{N+n}\mathbf{e} \;=\; \sum_{n=1}^{N-1} n\boldsymbol{\pi}_n \mathbf{e} + N\boldsymbol{\pi}_N (\mathbf{I}-\mathbf{R})^{-1}\mathbf{e} + \boldsymbol{\pi}_N \mathbf{R}(\mathbf{I}-\mathbf{R})^{-2}\mathbf{e}. \tag{22}$$

The expected sojourn time of a job can then be calculated using Little's law [58] and (22), which yields

$$\mathbb{E}[T] \;=\; \lambda^{-1} \left(\sum_{n=1}^{N-1} n\boldsymbol{\pi}_n \mathbf{e} + N\boldsymbol{\pi}_N (\mathbf{I}-\mathbf{R})^{-1}\mathbf{e} + \boldsymbol{\pi}_N \mathbf{R}(\mathbf{I}-\mathbf{R})^{-2}\mathbf{e} \right). \tag{23}$$

The asymptotic decay rate of the tail distribution of the queue length process is another measure of interest. Let η denote this decay rate, which is also called the caudal characteristic [30]. Note that $\eta = \mathrm{sp}(\mathbf{R})$ under the assumptions of this paper. Let $\mathbf{u}$ and $\mathbf{v}$ be the left and right eigenvectors corresponding to η that are strictly positive and normalized by $\mathbf{u}\mathbf{e} = 1$ and $\mathbf{u}\mathbf{v} = 1$. Under the assumptions herein, it is well known that [39]

$$\mathbf{R}^z \;=\; \eta^z\, \mathbf{v}\cdot\mathbf{u} \;+\; \mathrm{o}(\eta^z), \quad \text{as } z \to \infty,$$

which together with equation (6) yields

$$\boldsymbol{\pi}_{N+z}\mathbf{e} \;=\; \boldsymbol{\pi}_N \mathbf{v}\, \eta^z \;+\; \mathrm{o}(\eta^z), \quad \text{as } z \to \infty. \tag{24}$$

It then follows that

$$\mathbb{P}[Q \ge N+z] \;=\; \frac{\boldsymbol{\pi}_N \mathbf{v}}{1-\eta}\, \eta^z \;+\; \mathrm{o}(\eta^z), \quad \text{as } z \to \infty, \tag{25}$$

and thus

$$\lim_{z\to\infty} \frac{\mathbb{P}[Q \ge N+z]}{\eta^z} \;=\; \frac{\boldsymbol{\pi}_N \mathbf{v}}{1-\eta}, \tag{26}$$

or equivalently

$$\mathbb{P}[Q \ge N+z] \;\sim\; \frac{\boldsymbol{\pi}_N \mathbf{v}}{1-\eta}\, \eta^z, \quad \text{as } z \to \infty, \tag{27}$$

where $f(z) \sim g(z)$ denotes that $\lim_{z\to\infty} f(z)/g(z) = 1$. Hence, in addition to providing the stability criterion for the parallel processing system, η is indicative of the tail behavior of the stationary queue length distribution.

Note that the caudal characteristic can be obtained without having to first solve for the matrix $\mathbf{R}$. Define the matrix $\mathbf{A}^*(s) \equiv \mathbf{A}_0 + s\mathbf{A}_1 + s^2\mathbf{A}_2$, for $0 < s \leq 1$. When the generator matrix $\mathbf{A}$ is irreducible, this matrix $\mathbf{A}^*(s)$ is irreducible with nonnegative off-diagonal elements. Let $\chi(s)$ denote the spectral radius of the matrix $\mathbf{A}^*(s)$. Then, under the above assumptions, η is the unique solution in (0,1) of the equation $\chi(s) = 0$. This solution can be directly computed, although more efficient methods have been developed [3].

Another performance measure of interest is the long-run proportion of time the system spends executing reconfigurations. This can be expressed as

$$p_r \;=\; (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{N-1})\nu_b + \sum_{n=0}^{\infty} \boldsymbol{\pi}_{N+n}\nu_r \;=\; (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{N-1})\nu_b + \boldsymbol{\pi}_N(\mathbf{I} - \mathbf{R})^{-1}\nu_r, \qquad (28)$$

where the v^{th} position of the vector ν_b (respectively, ν_r) contains a 0 if $j_\ell^C = 0$ in the corresponding state of the boundary (respectively, homogeneous portion) and contains a 1 otherwise (i.e., when $j_\ell^C \in \{1, \ldots, m_\ell^C\}$).

4 Analysis of Memory Reference Behavior

Our analysis of dynamic spacesharing in the previous section does not explicitly include the memory requirements of parallel applications as a function of the number of servers allocated to the applications. This requires a stochastic analysis of the memory reference behavior of the parallel programs at each server allocated to them, which in turn depends upon the parallel processing system and parallel applications of interest.

In this section we derive such an analysis for a general class of parallel processing systems and parallel applications whose memory reference behaviors are consistent with the findings of many empirical studies since the early 1970s of serial applications on single-server systems. Specifically, the parallel program behavior tends to consist of (not necessarily disjoint) locality phases that clearly dominate the fraction of total memory references and transitions that account for a considerable fraction of the program's page faults. Phase periods tend to be of much longer duration than transition periods, but both phases and transitions are of equal importance in the program's memory reference behavior. While these properties may differ from those found in parallel environments where the entire memory reference set at each server must be entirely loaded in the physical memory of the server [42, 27, 6, 33, 32], they indeed have been shown to hold for an important class of computationally-intensive parallel applications that consist of numerical computations on large amounts of data in an iterative manner [34, 35].

A classical approach to derive a stochastic analysis of memory reference behavior consists of formulating a nearly completely decomposable model of program behavior that exploits its locality properties and obtaining an approximate solution of the model using the methods of decomposition and aggregation; see [9, 2] and the references therein. The model solution is based on extensions of the Simon-Ando approximations for the stationary probabilities of the corresponding Markov chain. This solution is then used to estimate the miss ratios of a program as a function of the physical memory capacity, which are computed as weighted sums of the miss ratios for the individual locality phases where the weights correspond to the long-run proportion of time the program spends in the respective localities. Error bounds also can be obtained within this framework; e.g., refer to [10, 56].

On the other hand, the iterative nature of the parallel applications of interest cause program localities to be visited multiple times in some, potentially non-deterministic, manner. This aspect of parallel programs is not taken into account in any of the previous models of program behavior, even though the impact of this behavior on system performance can be quite significant [35]. The reason for this is easily illustrated with a simple example. Consider a program consisting of multiple localities whose sizes are individually less than, but cumulatively greater than, the physical memory capacity. Previous modeling approaches, including those in [9, 2] and their references, will yield a miss ratio of zero in this case because the miss ratio for each locality is zero and thus the weighted sum (for any set of weights) is zero. However, if the program iterates among the different locality phases then it can experience a significant memory overhead as potentially large portions of each locality are reloaded upon return to the locality. This represents a fundamental problem with previous nearly completely decomposable models and analyses of the memory reference behavior of programs that remained open since the early 1970s until it was first observed and addressed in [35]. While our focus here is on parallel processing systems and parallel applications, our solution to this longstanding open problem and corresponding results are completely general and can be applied in any computing environment with the foregoing program behavior properties, especially given the large size of physical memory in contemporary computer systems.

4.1 Program Behavior Models

We now develop models of memory reference behavior that are used to approximate the paging overhead incurred by a program on each server allocated to it. This consists of modeling the memory reference behavior of the program at each server and deriving approximations for the corresponding miss ratio realized by the program when executed with a finite capacity memory. The page replacement policy defines which page currently residing in memory will be replaced upon the reference to a page that is not in memory, and thus it determines the set of pages that are in memory at any given time. Computer systems, including the servers comprising parallel processing systems, often employ (or approximate) the least recently used (LRU)

replacement algorithm in which the page chosen for replacement is the one that has not been referenced for the longest time. We therefore assume an LRU page replacement algorithm.

There has been a significant amount of research concerning the stochastic modeling and analysis of the page reference behavior of programs; see [11, 8, 9, 48, 12, 2] and the references therein. The independent reference model (IRM) has many advantages from the mathematical modeling viewpoint, but this model can produce very poor approximations in practice because it does not capture any concept of locality. In fact, IRM goes to the other extreme by assuming that the probability of the next page reference is independent of the page currently being referenced. The LRU stack model (LRUSM), on the other hand, specifies the distribution for the inter-stack distances in an LRU stack. While this model can be parameterized to give a high probability for referencing the pages at (or near) the top of the stack, it still does not capture the idea that there could be sudden jumps between localities, which has been empirically shown to occur in real programs, nor does it capture changing locality sizes. In our study, we use the Markov reference model (MRM) of program behavior because it captures quite well the concepts of program locality and dynamic transitions between localities during program execution.

The page reference string generated by a program, denoted by a sequence $\{Y_n\,;\, n \in \mathbb{Z}_+\}$, is assumed to be a realization of a time-homogeneous Markov process with irreducible page transition matrix $[p_{ij}]$ where

$$p_{ij} = \mathbb{P}[Y_{n+1} = j \mid Y_n = i].$$

The miss ratio is defined as the ratio of the total number of page faults to the total number of references. Let $LRU(n)$ denote the set of all pages in the LRU stack after the nth reference. Then we can calculate the page fault probability as

$$\lim_{n\to\infty} \mathbb{P}[Y_{n+1} \notin LRU(n)].$$

Assuming the measurement interval to be sufficiently long, the miss ratio converges to the page fault probability.

Computing the page fault probability directly from the probability transition matrix $\mathbf{P} = [p_{ij}]$ can be prohibitively expensive both in terms of time and space. This is due to the fact that the computation involves a sum over each of the states of the LRU stack, which grows exponentially. This direct approach, however, ignores the important properties of program behavior. The structural properties of this behavior result in a type of system that has been termed nearly completely decomposable by Simon and Ando [9]. We reduce the complexity of the problem at hand by exploiting these structural properties to facilitate our analysis of the dynamic behavior of parallel programs.

Assume for the moment that the program localities are disjoint and that no additional pages are referenced in transition from one locality to the next. Arrange the rows and columns of the probability transition matrix $\mathbf{P}$ such that the pages of each locality are grouped together consecutively. We then block partition $\mathbf{P}$

according to these locality sets, which yields the following structure

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1L} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2L} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}_{L1} & \mathbf{P}_{L2} & \cdots & \mathbf{P}_{LL} \end{bmatrix}$$

where $\mathbf{P}_{IJ}$ defines transitions from pages in locality $I \in \{1, \ldots, L\}$ to pages in locality $J \in \{1, \ldots, L\}$, and L denotes the number of program localities. It follows from the locality property of program behavior that the submatrices along the main diagonal (i.e., $\mathbf{P}_{11}, \cdots, \mathbf{P}_{LL}$) consist of relatively large probability measures, while the elements of the other submatrices are very small in comparison (i.e., $\mathbf{P}_{IJ} \approx \mathbf{0}, J \neq I$).

Matrices of this type are called nearly completely decomposable [9]. In general, the matrix $\mathbf{P}$ can be written in the form

$$\mathbf{P} = \mathbf{Q}^* + \varepsilon \mathbf{C} \tag{29}$$

where $\mathbf{Q}^* = \textbf{diag}(\mathbf{Q}_1^*, \mathbf{Q}_2^*, \cdots, \mathbf{Q}_L^*)$, the matrices $\mathbf{Q}_I^*$ are stochastic, $I \in \{1, \ldots, L\}$, ε is small compared to the elements of $\mathbf{Q}^*$, and $|c_{ij}| \leq 1$. The matrix $\mathbf{Q}^*$ is said to be completely decomposable because all elements off the main diagonal of submatrices are zero. We refer the interested reader to [9] for a comprehensive treatment of completely and nearly completely decomposable matrices and their solutions.

Intuitively, we decompose the probability transition matrix $\mathbf{P}$ into a macro model and L individual micro models. This is consistent with the use of macro and micro models in [12]. The key idea is that the macro model characterizes the transitions between localities, while each micro model characterizes the references within a particular locality. From this perspective, the matrices $\mathbf{Q}_I^*$ are the probability transition matrices for each of the individual micro models. The macro model is defined by the matrix $[\widehat{p}_{IJ}]$ of dimension $L \times L$ whose elements represent the probability of referencing some page in locality J on the next memory reference, given that the current reference is to some page in locality I. More formally, we have

$$\widehat{p}_{IJ} = \mathbb{P}[Y_{n+1} \in J \mid Y_n \in I].$$

Note that the subscripts are capitalized to emphasize that they refer to the macro model. Letting $\underline{\psi}_I$ denote the invariant probability vector of the matrix $\mathbf{Q}_I^*$, the elements $\widehat{p}_{IJ}$ are then computed as

$$\widehat{p}_{IJ} = \sum_{i \in I} \sum_{j \in J} \psi_{I,i}\, p_{ij}\ . \tag{30}$$

In the interest of space, we do not describe here our approaches for adding within the above framework the case of overlapping localities and of pages that are referenced in transitions between localities. These technical details can be found in [34]. We note, however, that the page faults due to transitions between

localities, together with those resulting from multiple visits to localities in an iterative manner, are included in our analysis of the macro model (see Section 4.3).

As a consequence of the Simon-Ando theorems [9], we can make the following important observations regarding the model of program behavior developed above. In the short-term period, equilibrium is (approximately) reached separately by each locality $\mathbf{Q}_I^*$. These equilibrium states are (approximately) preserved over the long-term period such that the distribution of references among the different localities approaches the distribution $\widetilde{\boldsymbol{\pi}} = (\widetilde{\pi}_1, \widetilde{\pi}_2, \ldots, \widetilde{\pi}_I, \ldots, \widetilde{\pi}_L)$, where $\widetilde{\pi}_I$ is the stationary probability that a memory reference is directed to locality I.

We next consider the miss ratio and memory overhead realized by the program at each server within the context of our program behavior model. Let M denote the number of pages comprising the finite capacity memory available to a parallel application at each of the V servers allocated to it. To elucidate the exposition, we divide the page faults incurred by a program at each server into two different classes. The first consists of faults that occur within each locality, whereas the second class of page faults consists of those resulting from transitions between localities and from the iterative nature of parallel programs in which localities are visited multiple times in some (non-deterministic) fashion. We refer to the miss ratios and memory overheads due to these different types of page faults as the intra-locality and inter-locality miss ratios and memory overheads, respectively, and consider each type in turn. These methods are then used to approximate the total memory overhead incurred by a parallel application as a function of its server allocation.

4.2 Intra-Locality Memory Overhead

Given a nearly completely decomposable model of program behavior with irreducible page transition matrix $\mathbf{P} = [p_{ij}]$ of the form in equation (29), we compute the intra-locality miss ratio $\mathcal{R}^{w\ell}$ (i.e., the miss ratio $w$ithin a ℓocality) under the LRU page replacement algorithm as follows. Let $F(\mathbf{P}, M)$ denote the stationary page fault probability for a program model $\mathbf{P}$ and a finite memory of M pages. More formally,

$$F(\mathbf{P}, M) = \lim_{n \to \infty} \mathbb{P}[Y_{n+1} = i \mid i \notin LRU(n)].$$

It follows from the properties of the nearly completely decomposable matrix $\mathbf{P}$ that the page fault probability can be approximated as

$$F(\mathbf{P}, M) \approx \sum_{I=1}^{L} \widetilde{\pi}_I F(\mathbf{Q}_I^*, M), \tag{31}$$

the accuracy of which is known to be (at least) within $\mathrm{O}(\varepsilon)$ [9]. Due to the locality property exploited in the construction of the matrix $\mathbf{P}$, such errors are expected to be very small. The intra-locality miss ratio is then obtained from equation (31) by recalling that $\mathcal{R}^{w\ell}$ converges to $F(\mathbf{P}, M)$ for sufficiently long measurement intervals.

The intra-locality miss ratio of the program model $\mathbf{P}$ is therefore obtained by solving L much smaller micro models to calculate their corresponding miss ratios (i.e., $F(\mathbf{Q}_I^*, M)$, $I \in \{1, \ldots, L\}$) and combining these results with the solution to the macro model (i.e., $\widetilde{\boldsymbol{\pi}}$). If the matrices $\mathbf{Q}_I^*$ are fairly small, then the intra-locality page fault probability can be computed by analyzing the Markov chain over the state space of all possible LRU stacks. While this Markov chain is also irreducible, since the $\mathbf{Q}_I^*$ are irreducible, and a closed-form solution exists for the page fault probability, the state space can become prohibitively large even for moderate size $\mathbf{Q}_I^*$ matrices. Some of the methods in Section 3 can be exploited to address this problem when the appropriate structure exists. However, it is important to note that there is significant evidence [2, 12, 34] suggesting that in the short-term (micro model) the IRM and LRUSM assumptions may be satisfied. This is especially the case for the LRUSM in part because sudden jumps between localities are directly handled outside of the micro model for each locality. We further note that closed-form expressions exist for $F(\mathbf{Q}_I^*, M)$ when $\mathbf{Q}_I^*$ corresponds to the IRM or LRUSM [2].

The intra-locality memory overhead of the program model $\mathbf{P}$ as a function of the memory capacity M on each of the V servers allocated to the program, $\mathcal{M}_{VM}^{w\ell}$, is then obtained in terms of the corresponding miss ratio as

$$\mathcal{M}_{VM}^{w\ell} \;=\; \mathcal{R}_{VM}^{w\ell}\; R_{VM}\, \mathcal{C}, \tag{32}$$

where R_{VM} is a non-negative discrete random variable representing the total number of memory references for the program, and $\mathcal{C}$ is the cost to load a page in memory. Note that the product $R_{VM}\mathcal{R}_{VM}^{w\ell}$ represents the number of page faults incurred by the program.

Equation (31) is a very valuable result in that it greatly simplifies the computation involved in calculating $\mathcal{R}^{w\ell}$. This approximation, however, does not account for the page faults generated during repeated visits to each locality. The importance of these effects were previously noted above, although now the problem can be understood even more precisely. Consider a program that visits each of its localities multiple times, where every locality fits in memory but the total number of pages far exceeds the capacity of memory. Thus, $F(\mathbf{Q}_I^*, M) = 0$, $I \in \{1, \ldots, L\}$, and from equation (31) the miss ratio $F(\mathbf{P}, M)$ is equal to 0, which is clearly not correct given the multiple visits to localities with the number of pages comprising the program far exceeding M. We capture this important aspect of program behavior in the inter-locality miss ratio and memory overhead.

4.3 Inter-Locality Memory Overhead

We develop an approximation to the inter-locality miss ratio $\mathcal{R}^{a\ell}$ and inter-locality memory overhead $\mathcal{M}^{a\ell}$ (i.e., the miss ratio and memory overhead $a$cross ℓocalities) under LRU page replacement based primarily on the macro model. Since the program reference behavior is assumed to be Markovian, the process repeats itself (i.e., regenerates) on each return to locality I. Thus, the measures we need to consider concern the

number of pages in locality I that must be reloaded on return to locality I.

Let C_I be a non-negative discrete random variable that represents the number of pages in locality I that have to be reloaded on return to I, and let N_I^* be a non-negative discrete random variable that represents the number of returns to locality I, for $I \in \{1, \ldots, L\}$. The inter-locality memory overhead then can be calculated as

$$\mathcal{M}^{a\ell} \;=\; \sum_{I=1}^{L} \sum_{j=1}^{N_I^*} C_{I,j}, \tag{33}$$

where $C_{I,1}, C_{I,2}, \ldots, C_{I,N_I^*}$ is a sequence of i.i.d. random variables such that $C_{I,j} \stackrel{\mathrm{d}}{=} C_I$, $j = 1, 2, \ldots, C_{I,N_I^*}$. Under the assumptions herein, the random variable N_I^* is a stopping time for this sequence and thus the moments of $\mathcal{M}^{a\ell}$ can be determined by applications of Wald's equation; e.g., see [58]. Dividing the inter-locality memory overhead by the total number of references yields the corresponding inter-locality miss ratio. All that remains is the calculation of measures of the random variables N_I^* and C_I, $I \in \{1, \ldots, L\}$.

4.3.1 Calculation of N_I^*

We can calculate measures of N_I^* from the distribution of first passage times and the corresponding recurrence time random variable τ_I^* representing the elapsed time between first entry into locality I and the next return to I. Following [35], we focus on the recurrence time τ_I^*. Let $\{Z_n\,;\, n \in \mathbb{Z}_+\}$ be a Markov chain with one-step transition matrix corresponding to the macro model of Section 4.1, defined over the state space $\mathbf{S} = \{1, 2, \ldots L\}$ of localities. The mean recurrence time to state I, $\mathbb{E}[\tau_I]$, is then defined as

$$\mathbb{E}[\tau_I] \;=\; \sum_{n=1}^{\infty} n\mathbb{P}[Z_n = I, Z_\nu \neq I, 0 < \nu < n \mid Z_0 = I]. \tag{34}$$

Note that τ_I is not the measure we are interested in because it includes events of the form $[Z_1 = I \mid Z_0 = I]$, which clearly do not imply any change of locality, and because it does not include the time spent in locality I upon first entry. Instead we need τ_I^* whose expectation is defined by

$$\begin{aligned}
\mathbb{E}[\tau_I^*] \;&=\; \sum_{n=1}^{\infty} n\mathbb{P}[Z_n = I, Z_\nu \neq I, 1 < \nu < n \mid Z_1 \neq I, Z_0 = I] \;+\; \sum_{n=1}^{\infty} n\widehat{p}_{II}^{\,n}(1 - \widehat{p}_{II}) \\
&=\; \sum_{n=1}^{\infty} n\mathbb{P}[Z_n = I, Z_\nu \neq I, 1 < \nu < n \mid Z_1 \neq I, Z_0 = I] \;+\; \frac{\widehat{p}_{II}}{1 - \widehat{p}_{II}}.
\end{aligned}$$

Equation (34) can be rewritten as

$$\begin{aligned}
\mathbb{E}[\tau_I] &= \mathbb{P}[Z_1 = I \mid Z_0 = I] + \sum_{n=2}^{\infty} n\mathbb{P}[Z_n = I, Z_\nu \neq I, 0 < \nu < n \mid Z_0 = I] \\
&= \widehat{p}_{II} + \sum_{n=2}^{\infty} \Big\{ n\mathbb{P}[Z_n = I, Z_\nu \neq I, 1 < \nu < n \mid Z_1 \neq I, Z_0 = I]\mathbb{P}[Z_1 \neq I \mid Z_0 = I] \Big\} \\
&= \widehat{p}_{II} + (1 - \widehat{p}_{II})\Big(\mathbb{E}[\tau_I^*] - \frac{\widehat{p}_{II}}{1 - \widehat{p}_{II}}\Big),
\end{aligned}$$

from which it follows that

$$\mathbb{E}[\tau_I^*] = \frac{\mathbb{E}[\tau_I]}{1 - \widehat{p}_{II}} . \tag{35}$$

For a recurrent Markov chain, $\mathbb{E}[\tau_I] = 1/\widetilde{\pi}_I$ where $\widetilde{\pi}_I$ is the invariant probability for state I of the Markov chain Z_n. Upon substituting for $\mathbb{E}[\tau_I]$ in equation (35) we obtain

$$\mathbb{E}[\tau_I^*] = \frac{1}{\widetilde{\pi}_I - \widetilde{\pi}_I \widehat{p}_{II}} . \tag{36}$$

4.3.2 Calculation of C_I

The calculation of measures of C_I can be obtained exactly or approximately by an analysis of the program reference matrix $\mathbf{P}$ and its properties developed above. Since the details of this analysis can depend upon the specific details of the matrix $\mathbf{P}$, in what follows we calculate general upper and lower estimates of the moments of C_I.

To illustrate our general approach, consider a particular sample path that takes us through the localities $K_1, K_2, \ldots, K_r$ in any order and any number of times, before returning to locality I. Since $\mathbf{P}$ is an irreducible and nearly completely decomposable Markov reference matrix, it is reasonable to assume that the program spends enough time in each of the localities $K_1, K_2, \ldots, K_r$ to have referenced all of their pages at least once. Let $\mathbf{G} = \{K_1, K_2, \ldots, K_r\}$, and define $\|\mathbf{G}\|$ to be the number of distinct pages in all of the localities comprising $\mathbf{G}$. Observe that if all of the localities in $\mathbf{G}$ were referenced at least once before a return to locality I, then $\|\mathbf{G}\|$ pages would have been loaded in memory. If $\|\mathbf{G}\| + \|\{I\}\| \leq M$, no pages need to be reloaded on return to locality I, whereas if $M - \|\{I\}\| < \|\mathbf{G}\| < M$, then the number of pages to be reloaded on return to locality I is between $\|\mathbf{G}\| + \|\{I\}\| - M$ and $\|\{I\}\|$. Finally, if $\|\mathbf{G}\| \geq M$, then all $\|\{I\}\|$ pages have to be reloaded on return to I.

More precisely, we define a lower estimate of the overhead $\alpha_I^l(\mathbf{G})$ as

$$\alpha_I^l(\mathbf{G}) \equiv \begin{cases} 0, & \|\mathbf{G}\| + \|\{I\}\| \leq M, \\ \|\mathbf{G}\| + \|\{I\}\| - M, & M - \|\{I\}\| < \|\mathbf{G}\| < M, \\ \|\{I\}\|, & \|\mathbf{G}\| \geq M, \end{cases}$$

and an upper overhead estimate $\alpha_I^u(\mathbf{G})$ as

$$\alpha_I^u(\mathbf{G}) \equiv \begin{cases} 0, & \|\mathbf{G}\| + \|\{I\}\| \le M, \\ \|\{I\}\|, & \text{otherwise.} \end{cases}$$

Now let ${}_{\mathbf{G}}\xi_{II}$ denote the probability that upon leaving locality I, all (and only) the localities in $\mathbf{G}$ are visited in any order and any number of times before returning to locality I. More formally,

$$\begin{aligned} {}_{\mathbf{G}}\xi_{II}^{(n)} &= \mathbb{P}[Z_n = I, \{Z_\nu : 1 < \nu < n\} = \mathbf{G} \mid Z_1 \neq I, Z_0 = I], \\ {}_{\mathbf{G}}\xi_{II} &= \sum_{n>1} {}_{\mathbf{G}}\xi_{II}^{(n)} \, . \end{aligned}$$

With this formulation we can compute an upper and lower estimate for $\mathbb{E}[C_I^k]$ as follows

$$\sum_{\mathbf{G}:\mathbf{G}\subset\mathbf{S}, I\notin\mathbf{G}} {}_{\mathbf{G}}\xi_{II}(\alpha_I^l(\mathbf{G}))^k \le \mathbb{E}[C_I^k] \le \sum_{\mathbf{G}:\mathbf{G}\subset\mathbf{S}, I\notin\mathbf{G}} {}_{\mathbf{G}}\xi_{II}(\alpha_I^u(\mathbf{G}))^k \, . \tag{37}$$

Note that the summation in equation (37) can be reduced to only those $\mathbf{G}$ for which $\alpha^u(\mathbf{G}) > 0$.

To calculate ${}_{\mathbf{G}}\xi_{II}$ we define a measure related to taboo probabilities [7]. Consider a set $\mathbf{H}$ of localities (we are actually interested in $\mathbf{H} = \mathbf{G}^c$). For all $J \in \mathbf{S}\backslash\mathbf{H}$ and $n \in \mathbb{Z}^+$, define

$${}_{\mathbf{H}}P_{JI}^{(n)} \equiv \mathbb{P}[Z_n = I, Z_\nu \notin \mathbf{H}, 1 < \nu < n, Z_1 \neq J \mid Z_0 = J]$$

and

$${}_{\mathbf{H}}P_{JI} \equiv \sum_{n\in\mathbb{Z}^+} {}_{\mathbf{H}}P_{JI}^{(n)} \, . \tag{38}$$

We then can decompose ${}_{\mathbf{H}}P_{JI}^{(n)}$ by the method of first entrance (refer to [7]) as follows

$$\begin{aligned} {}_{\mathbf{H}}P_{JI}^{(n)} &= \sum_{\substack{K\notin\mathbf{H} \\ K\neq J}} \widehat{p}_{JK} \sum_{\nu=0}^{n-2} \widehat{p}_{KK}^{\,\nu} \, {}_{\mathbf{H}}P_{KI}^{(n-\nu-1)}, \qquad n \ge 2; \\ \sum_{n\ge 2} {}_{\mathbf{H}}P_{JI}^{(n)} &= \sum_{n\ge 2} \sum_{\substack{K\notin\mathbf{H} \\ K\neq J}} \widehat{p}_{JK} \sum_{\nu=0}^{n-2} \widehat{p}_{KK}^{\,\nu} \, {}_{\mathbf{H}}P_{KI}^{(n-\nu-1)}, \qquad (39) \\ &= \sum_{\substack{K\notin\mathbf{H} \\ K\neq J}} \widehat{p}_{JK} \sum_{\nu=0}^{\infty} \sum_{n\ge\nu+2} \widehat{p}_{KK}^{\,\nu} \, {}_{\mathbf{H}}P_{KI}^{(n-\nu-1)}, \qquad (40) \\ &= \sum_{\substack{K\notin\mathbf{H} \\ K\neq J}} \frac{\widehat{p}_{JK}}{1-\widehat{p}_{KK}} \, {}_{\mathbf{H}}P_{KI} \, . \qquad (41) \end{aligned}$$

Rewriting the left hand side of equation (41) in terms of equation (38) yields a system of $\|\mathbf{S}\backslash\mathbf{H}\|$ equations which can be solved for the same number of unknowns. That is, for each $J \in \mathbf{S}\backslash\mathbf{H}$, we obtain

$$ {}_{\mathbf{H}}P_{JI} = \begin{cases} \sum\limits_{\substack{K \notin \mathbf{H} \\ K \neq J}} \dfrac{\widehat{p}_{JK}}{1-\widehat{p}_{KK}} \, {}_{\mathbf{H}}P_{KI} + \widehat{p}_{JI}, & J \neq I, \\ \sum\limits_{\substack{K \notin \mathbf{H} \\ K \neq J}} \dfrac{\widehat{p}_{JK}}{1-\widehat{p}_{KK}} \, {}_{\mathbf{H}}P_{KI}, & J = I. \end{cases} $$

To compute ${}_{\mathbf{G}}\xi_{II}$, we actually need

$$ \begin{aligned} \sum_{n \geq 2} \mathbb{P}[Z_n = I, Z_\nu \notin \mathbf{H}, 1 < \nu < n \mid Z_1 \neq I, Z_0 = I] &= \frac{{}_{\mathbf{H}}P_{II}}{\mathbb{P}[Z_1 \neq I \mid Z_0 = I]} \\ &= \frac{{}_{\mathbf{H}}P_{II}}{1-\widehat{p}_{II}} \, . \end{aligned} \tag{42} $$

Upon substituting $\mathbf{H} = \mathbf{G}^c$ in equation (42), ${}_{\mathbf{G}}\xi_{II}$ can be calculated via the following recursion

$$ {}_{\mathbf{G}}\xi_{II} = \frac{{}_{\mathbf{G}^c}P_{II}}{1-\widehat{p}_{II}} - \sum_{\mathbf{F}:\mathbf{F} \subsetneq \mathbf{G}} {}_{\mathbf{F}}\xi_{II} \, . \tag{43} $$

Hence, we calculate the upper and lower estimate for moments of C_I by substituting (43) in equation (37). Once again, we only need to compute ${}_{\mathbf{G}}\xi_{II}$ in (43) for those $\mathbf{G}$ for which $\alpha^u(\mathbf{G}) > 0$. As previously noted, one can often obtain a more accurate approximation for the moments of C_I by exploiting properties of the matrices $\mathbf{P}$ and $\mathbf{Q}_I^*$ in the above analysis. In fact, exact values of $\mathbb{E}[C_I]$ are obtained for the parallel applications considered in [34, 35], and the same approach can be exploited to obtain higher moments of C_I in such cases.

4.4 Total Memory Overhead

The total memory overhead of the program model $\mathbf{P}$ as a function of the memory capacity M on each of the V servers allocated to the program, $\mathcal{M}_{VM}$, can be expressed in terms of the corresponding intra-locality and inter-locality memory overheads as

$$ \mathcal{M}_{VM} \;=\; \mathcal{M}_{VM}^{w\ell} \;+\; \mathcal{M}_{VM}^{a\ell}. \tag{44} $$

Assuming $\mathcal{M}_{VM}^{w\ell}$ and $\mathcal{M}_{VM}^{a\ell}$ are independent, the moments of $\mathcal{M}_{VM}$ can be obtained from the analysis in Sections 4.2 and 4.3. We then construct a phase-type distribution with parameters $(\underline{\beta}_V, \mathcal{S}_V'^B)$ of order $m_V'^B < \infty$ by matching as many moments (and/or density function) of $\mathcal{M}_{VM}$ as are of interest, using any of the best known methods for doing so; e.g., see [1, 21] and the references therein.

The total service times of parallel jobs when executed on partitions of size V are comprised of the combination of their corresponding original service requirements and total memory overheads. By exploiting the known closure properties of phase-type distributions [29, 20], this combined service time distribution can be appropriately constructed as a (possibly more compact) phase-type distribution based on $(\underline{\beta}_V, \mathcal{S}_V^B)$ and $(\underline{\beta'}_V, \mathcal{S'}_V^B)$ with mean $\mu'^{-1}_V = \mu_V^{-1} + \mathbb{E}[\mathcal{M}_{VM}]$, $V \in \{U, \ldots, P\}$, and substituted into the dynamic spacesharing model of Section 2. The analysis of this dynamic spacesharing model that includes the memory overheads incurred by each parallel application as a function of its server allocation is then directly given by our analysis derived in Section 3.

5 Conclusions

In this paper we have presented a mathematical analysis of server and memory resource allocation in parallel processing systems. First, a general stochastic model of parallel processing systems under dynamic spacesharing was formulated, and an exact matrix-analytic analysis of this model was derived. Our results included closed-form solutions for certain model instances based on their probabilistic structure and closed-form expressions for several measures of interest, such as the tail distribution of the queue length process and its asymptotic decay rate, the mean sojourn time of parallel jobs, and the long-run proportion of time the system spends reconfiguring the allocation of servers among the jobs. Second, a general nearly completely decomposable model of parallel program memory reference behavior was formulated, and a stochastic analysis of this model was derived. Our results included probability measures of the total memory overhead incurred by a parallel program as a function of its server allocation, which in turn can be directly incorporated in our stochastic analysis of parallel dynamic spacesharing systems. Moreover, we solved a longstanding open problem with previous nearly completely decomposable models and analyses by introducing and deriving results for the inter-locality miss ratio in addition to the classical intra-locality miss ratio. The collection of theoretical results presented in this paper can be exploited to investigate the design and performance space of parallel processing systems with respect to fundamental tradeoffs related to the server and memory allocation strategies and their interactions.

References

[1] S. Asmussen. Phase-type distributions and related point processes: Fitting and recent advances. In S. R. Chakravarthy and A. S. Alfa, editors, *Matrix-Analytic Methods in Stochastic Models*, pages 137–149. Marcel Dekker, 1997.

[2] O. I. Aven, E. G. Coffman, Jr., and Y. A. Kogan. *Stochastic Analysis of Computer Storage*. D. Reidel, 1987.

[3] N. G. Bean, J.-M. Li, and P. G. Taylor. Caudal characteristics of QBDs with decomposable phase spaces. In *Advances in Algorithmic Methods for Stochastic Models,* G. Latouche and P. Taylor (eds.), pages 37–55. Notable Publications, 2000.

[4] S. C. Borst. Optimal probabilistic allocation of customer types to servers. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 116–125, 1995.

[5] S. L. Brumelle. Some inequalities for parallel-server queues. *Operations Research*, 19:402–413, 1971.

[6] D. C. Burger, R. S. Hyder, B. P. Miller, and D. A. Wood. Paging tradeoffs in distributed shared-memory multiprocessors. *Journal of Supercomputing*, 8:1–30, 1996.

[7] K. L. Chung. *Markov Chains with Stationary Transition Probabilities*. Springer-Verlag, 1960.

[8] E. G. Coffman, Jr. and P. J. Denning. *Operating Systems Theory*. Prentice Hall, 1973.

[9] P. J. Courtois. *Decomposability*. Academic Press, 1977.

[10] P. J. Courtois and P. Semal. Error bounds for the analysis by decomposition of non-negative matrices. In *Proceedings of International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*, pages 253–268, 1983.

[11] P. J. Denning. The working set model for program behavior. *Communications of the ACM*, 11(5):323–333, May 1968.

[12] P. J. Denning. Working sets past and present. *IEEE Transactions on Software Engineering*, SE-6(1):64–84, January 1980.

[13] K. Dussa, B. Carlson, L. Dowdy, and K.-H. Park. Dynamic partitioning in transputer environments. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 203–213, 1990.

[14] D. L. Eager, J. Zahorjan, and E. D. Lazowska. Speedup versus efficiency in parallel systems. *IEEE Transactions on Computers*, 38(3):408–423, March 1989.

[15] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In *Job Scheduling Strategies for Parallel Processing,* D. G. Feitelson and L. Rudolph (eds.), pages 1–34. Springer-Verlag, 1997. Lecture Notes in Computer Science Vol. 1291.

[16] D. W. Fife. Scheduling with random arrivals and linear loss functions. *Management Science*, 11(3):429–437, 1965.

[17] J. L. Hellerstein, T. S. Jayram, and M. S. Squillante. Analysis of large-scale distributed information systems. In *Proceedings of International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, July 2000.

[18] N. Islam, A. Prodromidis, and M. S. Squillante. Dynamic partitioning in different distributed-memory environments. In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pages 244–270. Springer-Verlag, 1996. Lecture Notes in Computer Science Vol. 1162.

[19] L. Kleinrock. *Queueing Systems Volume II: Computer Applications*. John Wiley and Sons, 1976.

[20] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM, Philadelphia, 1999.

[21] G. Latouche and P. Taylor. *Advances in Algorithmic Methods for Stochastic Models*. Notable, 2000.

[22] S. Majumdar, D. L. Eager, and R. B. Bunt. Characterisation of programs for scheduling in multiprogrammed parallel systems. *Performance Evaluation*, 13(2):109–130, 1991.

[23] A. M. Makowski and R. D. Nelson. Distributed parallelism considered harmful. Technical Report RC 17448, IBM Research Division, December 1991.

[24] A. M. Makowski and R. D. Nelson. Optimal scheduling for a distributed parallel processing model. Technical Report RC 17449, IBM Research Division, February 1992.

[25] C. McCann, R. Vaswani, and J. Zahorjan. A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors. *ACM Transactions on Computer Systems*, 11(2):146–178, May 1993.

[26] C. McCann and J. Zahorjan. Processor allocation policies for message-passing parallel computers. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 19–32, May 1994.

[27] C. McCann and J. Zahorjan. Scheduling memory constrained jobs on distributed memory parallel computers. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 208–219, May 1995.

[28] R. D. Nelson, D. Towsley, and A. N. Tantawi. Performance analysis of parallel processing systems. *IEEE Transactions on Software Engineering*, pages 532–540, April 1988.

[29] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. The Johns Hopkins University Press, 1981.

[30] M. F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications*. Marcel Dekker, 1989.

[31] K.-H. Park and L. W. Dowdy. Dynamic partitioning of multiprocessor systems. *International Journal of Parallel Programming*, 18(2):91–120, 1989.

[32] E. W. Parsons and K. C. Sevcik. Benefits of speedup knowledge in memory-constrained multiprocessor scheduling. *Performance Evaluation*, 27&28:253–272, October 1996.

[33] E. W. Parsons and K. C. Sevcik. Coordinated allocation of memory and processors in multiprocessors. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 57–67, May 1996.

[34] V. G. Peris, M. S. Squillante, and V. K. Naik. Analysis of the impact of memory in distributed parallel processing systems. Technical Report RC 19336, IBM Research Division, October 1993.

[35] V. G. Peris, M. S. Squillante, and V. K. Naik. Analysis of the impact of memory in distributed parallel processing systems. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 5–18, May 1994.

[36] E. Rosti, G. Serazzi, E. Smirni, and M. S. Squillante. The impact of I/O on program behavior and parallel scheduling. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 56–65, June 1998.

[37] E. Rosti, G. Serazzi, E. Smirni, and M. S. Squillante. Models of parallel applications with large computation and I/O requirements. *IEEE Transactions on Software Engineering*, March 2002.

[38] L. E. Schrage and L. W. Miller. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research*, 14(4):670–684, 1966.

[39] E. Seneta. *Non-Negative Matrices and Markov Chains*. Springer Verlag, New York, second edition, 1981.

[40] J. Sethuraman and M. S. Squillante. Optimal stochastic scheduling in multiclass parallel queues. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 93–102, June 1999.

[41] J. Sethuraman and M. S. Squillante. Analysis of parallel-server queues under spacesharing and timesharing disciplines. In *Matrix-Analytic Methods: Theory and Applications*, G. Latouche and P. Taylor (eds.), pages 357–380. World Scientific, 2002.

[42] S. K. Setia. The interaction between memory allocation and adaptive partitioning in message-passing multicomputers. In *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), pages 146–164. Springer-Verlag, 1995. Lecture Notes in Computer Science Vol. 949.

[43] S. K. Setia, M. S. Squillante, and S. K. Tripathi. Processor scheduling on multiprogrammed, distributed memory parallel computers. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 158–170, May 1993.

[44] S. K. Setia, M. S. Squillante, and S. K. Tripathi. Analysis of processor allocation in multiprogrammed, distributed-memory parallel processing systems. *IEEE Transactions on Parallel and Distributed Systems*, 5(4):401–420, April 1994.

[45] K. C. Sevcik. Scheduling for minimum total loss using service time distributions. *Journal of the ACM*, 21(1):66–75, 1974.

[46] K. C. Sevcik. Characterizations of parallelism in applications and their use in scheduling. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 171–180, May 1989.

[47] K. C. Sevcik. Application scheduling and processor allocation in multiprogrammed parallel processing systems. *Performance Evaluation*, 19:107–140, 1994.

[48] J. R. Spirn. *Program Behavior: Models and Measurements*. Elsevier, 1977.

[49] M. S. Squillante. A matrix-analytic approach to a general class of G/G/c queues. Technical report, IBM Research Division, May 1996.

[50] M. S. Squillante. Matrix-analytic methods in stochastic parallel-server scheduling models. In *Advances in Matrix-Analytic Methods for Stochastic Models*, S. R. Chakravarthy and A. S. Alfa (eds.). Notable Publications, 1998.

[51] M. S. Squillante and E. D. Lazowska. Using processor-cache affinity information in shared-memory multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):131–143, February 1993.

[52] M. S. Squillante and K. P. Tsoukatos. Optimal scheduling of coarse-grained parallel applications. In *Proceedings of Eighth SIAM Conference on Parallel Processing for Scientific Computing*, March 1997.

[53] M. S. Squillante, F. Wang, and M. Papaefthymiou. Stochastic analysis of gang scheduling in parallel and distributed systems. *Performance Evaluation*, 27&28:273–296, October 1996.

[54] M. S. Squillante, D. D. Yao, and L. Zhang. Analysis of job arrival patterns and parallel scheduling performance. *Performance Evaluation*, 36–37:137–163, August 1999.

[55] M. S. Squillante, Y. Zhang, A. Sivasubramaniam, N. Gautam, H. Franke, and J. Moreira. Modeling and analysis of dynamic coscheduling in parallel and distributed environments. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 43–54, June 2002.

[56] G. W. Stewart. Computable error bounds for aggregated Markov chains. *Journal of the ACM*, 30:271–285, 1983.

[57] W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181–189, 1977.

[58] R. W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[59] J. Zahorjan and C. McCann. Processor scheduling in shared memory multiprocessors. In *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 214–225, May 1990.