Scheduling Adaptively Parallel Jobs

by

Bin Song

A. B. (Computer Science and Mathematics), Dartmouth College (1996)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1998

Author . . . .
Department of Electrical Engineering and Computer Science
November 6, 1998

Certified by . . . .
Charles E. Leiserson
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Scheduling Adaptively Parallel Jobs
by
Bin Song
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science
at the
Massachusetts Institute of Technology

Abstract

An adaptively parallel job is one in which the number of processors which can be used without waste changes during execution. When allocating processors to multiple adaptively parallel jobs, a job scheduler should attempt to be *fair*—meaning that no job gets fewer processors than another, unless it demands fewer—and *efficient*—meaning that the scheduler does not waste processors on jobs that do not need them. Moreover, the scheduler should adapt quickly and be implementable in a distributed fashion.

In this thesis, I present and analyze a randomized processor allocation algorithm, the *SRLBA* algorithm, which allocates processors to adaptively parallel jobs in a distributed system of P processors and J jobs. The algorithm consists of *rounds* of *load-balancing steps* in which processor migration may occur. In the case that each job has a demand which is more than its fair share P/J of the processors, I show that after $O(\lg P)$ rounds, the system is in an *almost fair and efficient* allocation with high probability.

To analyze the algorithm, I use a two-phase analysis with a potential-function argument. In Phase 1, I show that after $O(\lg P)$ rounds every job has at least a constant fraction of P/J processors. Then I show that in Phase 2, after an additional $O(\lg P)$ rounds, the system converges to an almost fair and efficient configuration.

Finally, I conclude my thesis with some directions for future work.

Thesis supervisor: Charles E. Leiserson
Title: Professor of Computer Science and Engineering

Acknowledgments

First I would like to thank my advisor, Professor Charles E. Leiserson. Charles first introduced me to the field of parallel job scheduling, and has guided my research in the area ever since. He has provided valuable guidance and good ideas to improve my research, in addition to always challenging me to do my best. Charles has also read my thesis over and over, helping me to present it better.

The algorithm and analysis presented in this thesis are joint work with Drs. Robert Blumofe, Charles Leiserson, Aravind Shrinivason, and David Zuckerman. I have gained valuable knowledge and inspiration from working with them.

This research was supported in part by the Defense Advanced Research Projects Agency under Grant N00014-94-1-0985. I thank DARPA for the support.

I would also like to thank my family for their moral support.

Finally, I want to thank my fiance, Matthew Cheyney. Matt is also a computer science student. He listened to me talking about my research, and pushed me to justify or change some of the more unrealistic assumptions of my model, resulting in a stronger paper. He also helped me tremendously with the writeup and the English therein.

Contents

Chapter 1

Introduction

Multiprocessor systems have moved from a rare research tool to a commercially viable product within the last ten years. Unfortunately, no clear consensus has emerged about how to schedule jobs on these systems. We believe that any good solution to scheduling should have the three characteristics. First, the system should adapt as the parallelism of jobs changes. Second, the system should converge to the desired processor allocation quickly. Third, the system should be distributed, since it should be easily scalable.

This thesis studies the problem of how to allocate processors to parallel jobs in a distributed multiprocessor system. In particular, we are concerned with ***adaptively parallel jobs:*** those for which the number of processors which can be used without waste changes during execution. I shall refer to this maximum number of efficiently usable processors for a job as the ***desire*** of the job. A scheduling algorithm can allocate a job more processors than it desires, in which case some of the processors will be underutilized. Similarly, a scheduling algorithm can allocate a job fewer processors than it desires, in which case some of the job's parallelism will not be exploited.

We call the problem of how to allocate processors to adaptively parallel jobs the ***adaptively parallel processor-allocation problem***. As multiprocessor systems become more common and as the number of processors in them increases, the the adaptively parallel processor-allocation problem also increases in importance. There is no obvious solution to this problem in a distributed system. Much research has been done on this problem [4, 7, 10, 11, 12, 14]. Some results concern static scheduling schemes [12]; others emphasize specific network topologies [9], such as a mesh; while others [7, 14] study the performances of different scheduling polices on shared memory multiprocessors. Most existing research is empirical. In this thesis, we consider the problem theoretically.

The results presented here represent joint work with Drs. Robert Blumofe, Charles Leiserson, Aravind Shrinivason, and David Zuckerman. We have developed and analyzed a distributed randomized dynamic processor-allocation algorithm which can bring the system to a "fair" and "efficient" configuration quickly as jobs enter and leave the system and change their parallelism.

In the remainder of this chapter, I define the notion of a "macroscheduling system" and the meaning of "fair" and "efficient" in the context of the processor allocation problem. Then I briefly describe the *SRLBA* processor allocation algorithm and the main result of this paper—the running time of the algorithm.

In the remainder of this paper, we use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$. To model the adaptively parallel processor-allocation problem, we define a ***macroscheduling system*** with P processors and J jobs to be a distributed system with a set of processors $[P]$, and a set of adaptively parallel jobs $[J]$. At any time, each job $j \in [J]$ has a desire d_j. In such a system, we wish to allocate an ***allotment*** m_j of processors to each job j so that the resulting allocation is "fair" and "efficient". An allocation which is "fair", but not necessarily "efficient", might give each job P/J processors, assuming P is a multiple of J. Whenever a job desires fewer than P/J processors, however this fair allocation is inefficient, because the excess processors might be put to a more productive use on another job. This observation leads to an intuitive definition of "efficient". An allocation is ***efficient*** if no job receives more processors than it desires. An allocation is ***fair*** if whenever a job receives fewer processors than it desires, then no other job receives more than one more processor than this job received. (Exactly equal allotments may be impossible due to integer roundoff.) We desire processor allocations that are both fair and efficient. An example of a fair and efficient allocation is illustrated in Figure 1-1.

In a macroscheduling system, the desires of the adaptive parallel jobs may change over time. A good scheduling algorithm should adapt as quickly as possible to changes. Since it is difficult to analyze the behavior of a scheduling algorithm in the midst of changing job desires, our analysis focuses on how rapidly the scheduler adapts to a fair and efficient allocation, assuming that the jobs' desires do not change during that adaption.

One simple scheme to solve the adaptive processor-allocation problem is ***equipartition-***

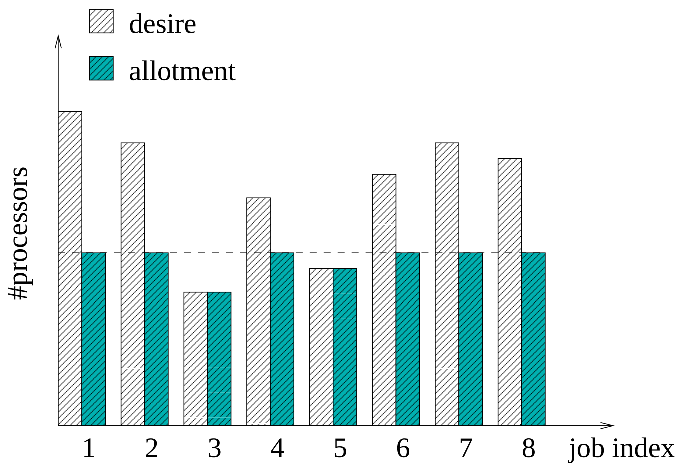

Figure 1-1: An example of the adaptively parallel processor-allocation problem. Each job $j = 1, 2, \ldots, 8$ has a desire d_j. A fair and efficient allocation m_j for these desires is shown.

ing [11, 14]. A centralized scheduler initially allocates one processor to each job. Any jobs that have reached their desires drop out. The scheduler repeats with the remaining jobs until either there are no remaining jobs or all processors have been allocated. An allocation which is produced by equipartitioning is both fair and efficient. Equipartitioning is hard to implement in a fully distributed fashion, however. It must be rerun whenever a job changes its desire or a job is created or destroyed.

Our first distributed scheduling algorithm for the adaptively parallel processor-allocation problem is called the ***RLB*** (Randomized Load-Balancing) ***algorithm***. It works roughly as follows. Each processor periodically pauses its computation and initiates a ***load-balancing step*** with a ***mate***. This mate is another processor chosen uniformly at random, independently of any other load-balancing step. Suppose that a processor $p \in [P]$ working on job k chooses processor $r \in [P]$ working for job j as a mate. If the allotment of job k exceeds the allotment of job j by 2 or more and job j's allotment is less than its desire, then processor p migrates to work on job j. Conversely, if the allotment of job j exceeds the allotment of job k by 2 or more and job k's allotment is less than its desire, then processor r migrates to work on job k. Otherwise, no processor migration occurs. Figure 1-2 illustrates such a load-balancing step, in which processor p migrates to work for job j.

Like equipartitioning,the *RLB* algorithm converges to a fair and efficient processor allocation. Unlike equipartitioning, however the *RLB* algorithm does not need centralized

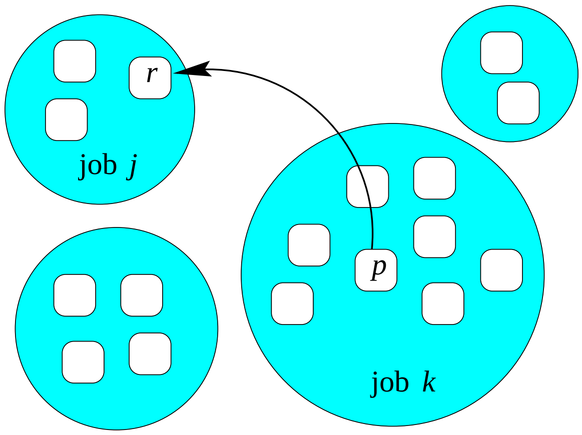

Figure 1-2: A load-balancing step. Circles denote jobs. Rounded rectangles are processors. This load-balancing step is initiated by processor p which is working for job k. Processor p chooses a processor r working for job j as its mate. Suppose job j's allotment is less then its desire, then, since job k's allotment is larger than job j's allotment by more than 2, processor p migrates to work for job j.

information on all jobs. It only requires that each job maintain its processor allotment. An empirical study [8] of this algorithm has already been performed. Simulation shows that this algorithm is efficient and stable when selected migrations are "damped" by allowing them to proceed only with a certain probability.

Although the structure of the RLB algorithm is suitable for a distributed environment, a distributed setting introduces several complicating issues. For example, what should happen when two processors randomly select the same mate? How does one define a job's processor allotment when several of its processors are attempting to balance loads simultaneously? Because of these complications, we have chosen to analyze this algorithm in a simpler model that highlights the behavior of the load-balancing step without introducing the full complexity of a distributed setting.

One simplifying assumption is that all load-balancing steps occur serially, and that the allotment of each job involved in such a step is updated promptly. We call this model the ***sequential perfect-information model***. Under the sequential perfect-information model, we can view the RLB algorithm as proceeding in a sequence of ***rounds***. Each round consists of P sequential load-balancing steps, in which each of the P processors has a chance to initiate a load-balancing step. To analyze the worst-case behavior of the system, we also introduce an ***adversary*** into the model. The adversary, in each step i of the round, having observed the previous $i-1$ steps, chooses a processor not previously chosen in this round to initiate the ith load balancing step.

We call the variant of the *RLB* algorithm that operates under the sequential perfect-information model and under the assumption of the adversary the ***SRLBA*** (Sequential Randomized Load-Balancing with Adversary) ***algorithm***. We define the ***absolute average allotment*** in a macroscheduling system with P processors and J jobs as $\mu = P/J$. For simplicity, we assume that μ is an integer in the remainder of this paper. We show that with the *SRLBA* algorithm, if each job has a desire which is more than the absolute average allotment and each job has at least one processor to start with, then within $O(\lg P)$ rounds, every job has within 1 of P/J processors with high probability. Using the assumption that the jobs' desires are more than the absolute average allotment is a simplification step of our analysis. We justify this assumption in Chapter 5. In the remainder of this paper, unless otherwise stated, we assume that every job has a desire that is more than the absolute average allotment.

The proof uses a two-phase analysis. Initially, some jobs may have very few processors. Phase 1 shows that with high probability, after $O(\lg P)$ steps, all jobs have at least $\epsilon P/J$ processors, where $0 < \epsilon < 1$ is a constant. Phase 2 shows that in an additional $O(\lg P)$ rounds, with high probability, every job has within 1 of P/J processors.

In each phase, we use a "potential-function" argument, in which we associate the system state with a nonnegative real-valued potential function. The potential is 0 if and only if the system reaches the desired processor allocation of the corresponding phase. We show that in each phase, the potential goes to 0 within $O(\lg P)$ rounds.

The remainder of the thesis is organized as follows. In Chapter 2, I give a formal description of the *SRLBA* algorithm and present the potential-function argument. In Chapter 3 and Chapter 4, I analyze Phase 1 and 2 of the algorithm using the corresponding potential-function arguments. Finally, in Chapter 5, I summarize our results, and point to some future work.

Chapter 2

The *SRLBA* Algorithm and the Potential-Function Argument

In the first part of this Chapter, I introduce the notion of "system configurations" to formally describe the *SRLBA* algorithm for the adaptive parallel processor allocation problem. In the second part of this chapter, I present a potential-function argument which is used in the two-phase analysis of the *SRLBA* algorithm in Chapter 3 and Chapter 3.

2.1 The *SRLBA* Algorithm

In this section, I model the macroscheduling system as a series of "configurations". I introduce the notion of an "almost fair and efficient" configuration. In addition, I describe the *SRLBA* algorithm precisely under this model.

The system we consider is a macroscheduling system with P processors and J jobs. In this system, at any time, each processor is allocated to one and only one job, and it works exclusively on that job. We say that the job *owns* the processor. We also say that the processor *belongs to* the job. We assume that initially, the allotment of every job is at least 1.

At any given time, we define the system ***configuration*** M to be the mapping

$$M : [P] \to [J] \cup \perp \ ,$$

where $M(p) = j$ if and only if processor p belongs to job j at that time. If processor p does not belong to any job, then $M(p) = \perp$. We define the ***configuration space*** $\mathcal{C}_{P,J}$ to be the set of all legal configurations.

We define the ***configuration vector*** V of a configuration $M : [P] \to [J]$, to be the J-tuple

$$V = \langle m_1, m_2, \ldots, m_J \rangle \ ,$$

where $m_k = |\{p : M(p) = k\}|$ is the allotment of job k. Since there are P jobs, it follows that $\sum_{k \in J} m_k \leq P$. We define a ***legal configuration*** to be a configuration in which each job owns at least one processor. Thus, for a legal configuration, the configuration vector has no zero entries. In the remainder of this paper, when we talk about a configuration, we assume that the configuration is legal, unless otherwise noted.

The *SRLBA* algorithm takes the system through a sequence of configurations by performing rounds of load-balancing steps. More precisely, each round consists P load-balancing steps. We begin the round with a configuration $M^{(0)}$. A generic round operates as follows. At the beginning of step i, $i \in [P]$, the system is in configuration $M^{(i-1)}$. The adversary, having observed the first $i-1$ steps of the round, picks a processor $p^{(i)}$ that has not yet been chosen this round. That is, $p^{(i)} \neq p^{(j)}$ for $j \in [i-1]$. Processor $p^{(i)}$ then performs a load-balancing step. The load-balancing step begins with processor $p^{(i)}$ selecting a mate $r^{(i)} \in [P]$ uniformly at random, independently of any other load-balancing steps. The two processors compare the processor allotments of the two jobs to which they belong. If the discrepancy in the processor allotments is 2 or more, then either $p^{(i)}$ migrates to $r^{(i)}$'s job or $r^{(i)}$ migrates to $p^{(i)}$'s job, whichever diminishes the discrepancy. At the end of the load-balancing step, processor allotments of the two jobs involved are updated appropriately to produce configuration $M^{(i)}$. We call this new configuration $M^{(i)}$ the ***successor*** of configuration $M^{(i-1)}$. Thus, during the round the adversary and the load-balancing algorithm together determine a ***trajectory*** $\langle M^{(0)}, M^{(1)}, \ldots, M^{(P)} \rangle$ of the system through the configuration space.

And example of a round is illustrated in Figure 2-1, where we have a macroscheduling system with 9 processors and 3 jobs. We assume that the processor desire of each job is larger than 9.

In the remaining of the analysis, we assume that the absolute average allotment μ is an integer that is at least 2. Under the assumption that each job has desire larger than μ, a

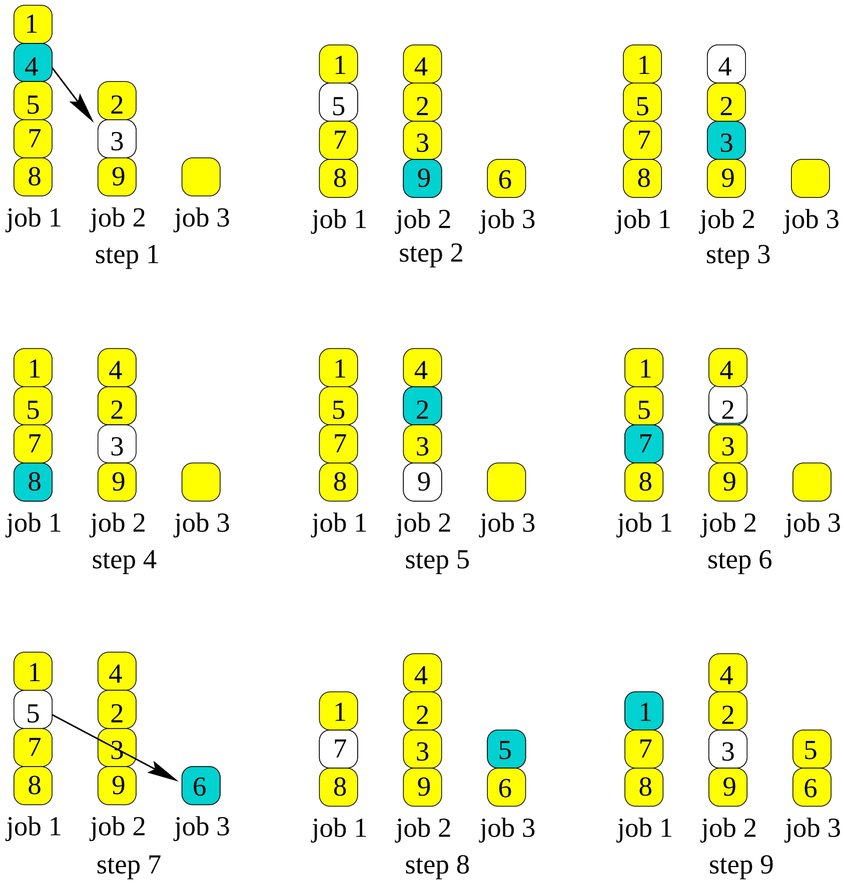

Figure 2-1: An example of a round in a macroscheduling system with 9 processors and 3 jobs. The rounded squares represent processors. In each load-balancing step, the adversary chooses a processor (the dark square) to initiate the step. This selected processor randomly chooses a processor as a mate (the light square). The arrows show the directions of processor migrations.

fair and efficient configuration is one in which every job has exactly $\mu = P/J$ processors. Moreover, for any configuration vector $\langle m_1, m_2, \ldots, m_J \rangle$, we have $\sum_{j\in[J]} m_j = P$.

Although a fair and efficient configuration is desirable, it may take the system a relatively long time to converge to an exact fair and efficient configuration using the *SRLBA* algorithm.[1] Thus, we introduce the notion of an ***almost fair and efficient configuration***, in which the processor allotment of every job belongs to the set $\{\mu-1, \mu, \mu+1\}$. An almost fair and efficient configuration is a very good approximation to the exact fair and

[1]Consider the case where all jobs have exactly μ processors except 2 jobs, where one of them has $\mu+1$ processors and the other has $\mu-1$ processors. To get to the exact configuration the expected number of rounds could be $\Theta(P)$ when $J = \Theta(P)$.

efficient configuration.

When there are changes in the system, a good scheduling algorithm should bring the system to a configuration as close to the fair and efficient configuration as soon as possible. We demonstrate this property of the *SRLBA* algorithm by proving the following result. *In a macroscheduling system with P processors and J jobs, if each job has desire larger than the absolute average allotment, and each job starts with at least* 1 *processor, then using the SRLBA algorithm, after $O(\lg P)$ rounds, with probability at least $1 - 1/P$, the system is in an almost fair and efficient configuration.*

The proof runs roughly as follows. We utilize a two-phase analysis of the algorithm. Initially, some jobs may have very few processors. We show that with high probability, after $O(\lg P)$ rounds, all jobs have at least $\epsilon\mu$ processors, where $0 < \epsilon < 1$ is a constant. We call these $O(\lg P)$ rounds ***Phase 1*** of the analysis. We show that in an additional $O(\lg P)$ rounds, with high probability, every job has within 1 of μ processors. These additional $O(\lg P)$ rounds constitutes ***Phase 2*** of our analysis. In each phase, we associate a nonnegative potential function with the configurations of the system. In Phase 1, the potential function is 0 if and only if every job has at least $\epsilon\mu$ processors. In Phase 2, the potential function is 0 if and only if the system is in an almost fair and efficient configuration. For both phases, we are able to prove that after $O(\lg P)$ rounds, with high probability, the potential goes to 0.

The proof is presented in three parts. Section 2.2 provides two lemmas concerning potential functions that will be used to analyze the phases. Chapter 3 analyzes the first phase, and Chapter 4 analyzes the second phase.

2.2 The Potential Function Arguments

The running-time analysis of the *SRLBA* algorithm is based on potential functions. The basic idea is that we associate a real-valued potential function with each system configuration. When the potential function reaches 0, then a desired configuration has been attained. In this section, we provide two lemmas that give conditions under which the potential functions used in the two-phase analysis converge quickly to 0.

For a macroscheduling system with P processors and J jobs, a ***potential function*** $\Phi : \mathcal{C}_{P,J} \to \mathbb{R}$ is a mapping from the configuration space to the real numbers such that

$\Phi(M) \geq 0$ for all configurations $M \in \mathcal{C}_{P,J}$. The following lemma shows that if a single load-balancing step decreases the expected potential of the system configurations by a factor of $1/P$, then in one round, we expect the potential to decrease by a constant factor.

Lemma 1 *Suppose that the load-balancing steps of a round take a macroscheduling system with P processors and J jobs through the trajectory $\langle M^{(0)}, M^{(1)}, \ldots, M^{(P)} \rangle$, and suppose that for some $\alpha > 0$, a potential function $\Phi : \mathcal{C}_{P,J} \to \mathrm{I\!R}$ can be shown to satisfy*

$$\Phi(M^{(i-1)}) - \mathrm{E}\left[\Phi(M^{(i)})\right] \geq (\alpha/P)\Phi(M^{(i-1)}) \tag{2.1}$$

for $i = 1, 2, \ldots, P$. Then, we have

$$\mathrm{E}\left[\Phi(M^{(P)})\right] \leq e^{-\alpha}\Phi(M^{(0)}) \ .$$

Proof: Rewriting Inequality (2.1), we obtain

$$\mathrm{E}\left[\Phi(M^{(i)})\right] \leq (1 - \alpha/P)\Phi(M^{(i-1)}) \ .$$

By linearity of expectation and iterating, we obtain

$$\begin{aligned} \mathrm{E}\left[\Phi(M^{(P)})\right] &\leq \Phi(M^{(0)})\,(1 - \alpha(1/P))^P \\ &\leq e^{-\alpha}\Phi(M^{(0)}) \ , \end{aligned}$$

since $1 + x \leq e^x$ for all $x \in \mathrm{I\!R}$. ■

We will show that the potential functions used to analyze the two phases of the load-balancing algorithm satisfy the property that they are polynomially bounded in the number P of processors. Moreover, they satisfy a "gap" property: if a potential function is sufficiently small, it is in fact 0. The next lemma shows that if in each round, the expected decrease in potential is a constant factor, then after $O(\lg P)$ rounds, the potential is 0 with high probability.

Lemma 2 *Suppose that T rounds of load balancing take a macroscheduling system with P processors and J jobs through the trajectory $\langle M^{(0)}, M^{(1)}, \ldots, M^{(T)} \rangle$, where $M^{(0)}$ is the* *initial configuration and $M^{(i)}$ is the configuration at the end of round i for $i = 1, 2, \ldots, T$. Let $\Phi : \mathcal{C} \to \mathbb{R}$ be a potential function that satisfies*

$$\mathrm{E}\left[\Phi(M^{(i)})\right] < \beta\Phi(M^{(i-1)}) \ , \tag{2.2}$$

where $\beta < 1$ is a positive constant, and suppose that for any configuration M of the system, there exist positive constants a and b such that

1. *$\Phi(M) < P^a$,*
2. *$\Phi(M) < 1/P^b$ implies $\Phi(M) = 0$.*

Then, for any $\delta > 0$, if $T \geq \log_{1/\beta}(1/\delta) + (a+b)\log_{1/\beta} P$, we have $\mathrm{E}\left[\Phi(M^{(T)})\right] = 0$ with probability at least $1 - \delta$.

Proof: By linearity of expectation and iterating, we obtain

$$\begin{aligned} \mathrm{E}\left[\Phi(M^{(T)})\right] &\leq \beta^T \Phi(M^{(0)}) \\ &\leq \beta^{\log_{1/\beta}(1/\delta) + \log_{1/\beta} P^{a+b}} \Phi(M^{(0)}) \\ &= \delta P^{-a-b} \Phi(M^{(0)}) \\ &\leq \delta P^{-a-b} P^a \\ &= \delta P^{-b} \ , \end{aligned}$$

since $\Phi(M^{(0)}) < P^a$ and $T \geq \log_{1/\beta}(1/\delta) + (a+b)\log_{1/\beta} P$.

Markov's inequality [5, p. 114] asserts that for any nonnegative random variable X with finite expectation, we have $\Pr\{X \geq t\} \leq \mathrm{E}[X]/t$ for all $t > 0$. Consequently, by Condition 2 we have

$$\begin{aligned} \Pr\left\{\Phi(M^{(T)}) > 0\right\} &= \Pr\left\{\Phi(M^{(T)}) \geq P^{-b}\right\} \\ &\leq \mathrm{E}\left[\Phi(M^{(T)})\right] / P^{-b} \\ &\leq \delta \ , \end{aligned}$$

which completes the lemma. ∎

In the following two sections, we will use the potential function argument to help prove the lower bound on the running time of our algorithm. Our analysis has two phases.

Initially, some jobs may have very few processors. We show that after the $O(\lg P)$ rounds of Phase 1, with high probability, every job has at least $\mu/8$ processors. We show that in the additional $O(\lg P)$ rounds of Phase 2, with high probability, the system is in an almost fair and efficient configuration. We use different potential functions in the analysis of the two different phases. Chapter 3 presents Phase 1, and Chapter 4 resents Phase 2.

Chapter 3

Analysis of Phase 1 of the *SRLBA* Algorithm

In this chapter, we consider Phase 1 of the *SRLBA* algorithm for the adaptive processor allocation problem. We show that, in a macroscheduling system with P processors and J jobs, if every job has a desire which is more than the absolute average allotment and each job starts with at least one processor, then under the *SRLBA* algorithm, with high probability, within $O(\lg P)$ rounds, every jobs has at least $\epsilon\mu$ processors, where $\epsilon \geq 1/8$ is the "Phase-1 constant". We define the ***Phase-1 constant*** to be $\epsilon = \lceil \mu/8 \rceil / \mu$. Although the definition of ϵ depends on μ, once μ is decided, then ϵ is a constant. For $2 \leq \mu \leq 8$, we have $\epsilon = 1/\mu$, in which case, $\epsilon\mu = 1$. For $\mu > 8$, we have $1/8 \leq \epsilon \leq 1/4$ and $\epsilon\mu$ is an integer $\geq \mu/8$.

We call a configuration a ***desired Phase-1 configuration*** if every job has at least $\epsilon\mu$ processors. Since every configuration considered here is a legal configuration in which each job owns at least one processor, we obtain that, if $2 \leq \mu \leq 8$, every configuration is automatically a desired Phase-1 configuration. Thus, in the remainder of the section, we only consider the systems where $\mu > 8$ is an integer. Thus, the corresponding Phase-1 constant ϵ is in the range $[1/8, 1/4]$.

Let M be a configuration of the system in Phase 1. Let $V = \langle m_1, m_2, \ldots, m_J \rangle$ be the configuration vector of M, where $m_i = |\{p : M(p) = i\}|$ is the processor allotment of job i. We define the ***Phase-1 partition*** of configuration M to be the following four disjoint sets of jobs A, B, C, and D:

$$
\begin{aligned}
A &= \{j \in [J] : m_j \le \epsilon\mu - 1\} \ , \\
B &= \{j \in [J] : m_j = \epsilon\mu\} \ , \\
C &= \{j \in [J] : m_j = \epsilon\mu + 1\} \ , \\
D &= \{j \in [J] : m_j \ge \epsilon\mu + 2\} \ .
\end{aligned}
$$

We want to associate some potential function with Phase 1, such that the potential function is 0 if and only if the desired configuration of Phase 1 is achieved. Thus, no jobs in partition B, C, and D should contribute to the potential, and at the same time, every job in partition A should contribute positively to the potential. In addition, if a configuration is far from the desired Phase-1 configuration, the potential of this configuration should be big. So, if a job j has a small allotment m_j, the job should contribute substantially to the potential. At the same time, this job j may increase its allotment quickly during a round relative to its allotment before the round. This rapid increase occurs, because a processor p working for job j is likely to choose a processor working for a job with bigger allotment as a mate during the load-balancing step initiated by processor p. Thus, every processor working for job j is likely to recruit another processor to migrate to work for job j. Therefore, intuitively the allotment m_j of job j is likely to double after a round. This doubling process indicates a good progress towards the desired Phase-1 configuration. Thus, for a job j in set A with allotment m_j, we would like the term $1/m_j$ to appear in the potential function. When m_j is small, $1/m_j$ is big; when m_j gets doubled, $1/m_j$ is halved. Using this intuition, we define the potential function Ψ of configuration M to be

$$\Psi(M) = \sum_{j \in A} (1/m_j - 1/\epsilon\mu) \ ,$$

where $1/\epsilon\mu$ is a normalization factor. Consequently, when $m_j = \epsilon\mu$, job j's contribution to the potential is 0. The function $\Psi(M)$ represents the relative discrepancy of M from the desired Phase-1 configuration. In addition, we define the potential function Υ of configuration M to be

$$\Upsilon(M) = \sum_{j \in A \cup B} (\epsilon\mu - m_j + 1/2) \ ,$$

which represents the absolute discrepancy of M from the desired Phase-1 configuration. Note that for any configuration M, the functions $\Psi(M)$ and $\Upsilon(M)$ are nonnegative.

The potential function which we shall use for Phase 1 is the product $\Psi\Upsilon(M) = \Psi(M)\Upsilon(M)$ of the two potential functions. We show that $\Psi\Upsilon(M) = 0$ if and only if configuration M is a desired Phase-1 configuration. And we show that in $O(\lg P)$ steps, $\Psi\Upsilon = 0$ with high probability.

Here is the outline of the argument. Consider any round of the algorithm. In each step of the round, the adversary must choose a processor owned by a job that either belongs to set $A \cup B$ of the Phase-1 partition or belongs to set $C \cup D$ of the Phase-1 partition to initiate a load-balancing step. We show that if a processor owned by a job that belongs to $A \cup B$ is chosen by the adversary, then there is enough expected decrease in Υ; if a processor owned by a job that belongs to $C \cup D$ is chosen by the adversary, then there is enough expected decrease in Ψ. In either case, we show that the expected decrease of the product of $\Psi\Upsilon$ is large enough in each step, so that after a round, we expect $\Psi\Upsilon$ to decrease by a constant factor. Then, applying Lemma 2, we obtain that within $O(\lg P)$ rounds, $\Psi\Upsilon = 0$ with high probability.

Lemma 3 and Lemma 4 provide some basic properties of the potential functions Ψ and Υ.

Lemma 3 *Consider a macroscheduling system with P processors. Under the SRLBA algorithm, given any configuration M, the potential function Ψ has the following properties:*

(i) if M' is a successor of M, then $\Psi(M') \leq \Psi(M)$;

(ii) $\Psi(M) \leq P$;

(iii) $\Psi(M) = 0$, if and only if M is a desired configuration of Phase 1;

(iv) if $\Psi(M) < P^{-2}$, we must have $\Psi(M) = 0$.

Proof: Let ϵ be the Phase-1 constant. Let $\mu = P/J$ where J is the number of jobs in the macroscheduling system. We first show part (i). After a load-balancing step, Ψ changes only when a processor previously working for a job $l \in [J]$ migrates to work for a different job $k \in [J]$. According to the different Phase-1 partitions to which job l and job k may belong right before the load-balancing step, we have the following three cases.

- Before the load-balancing step, job l and job k both belong to $A \cup B$ of the Phase-1 partition of M. Let m_l and m_k be job l and k's processor allotment in configuration M, and m'_l and m'_k be job l and job k's processor allotment in configuration M'. If $|m_l - m_k| < 2$, then, no migration happens and $\Psi(M') = \Psi(M)$. Otherwise, without loss of generality, we assume that $m_l - m_k \geq 2$. Thus, $m'_l = m_l - 1$ and $m'_k = m_k + 1$. We obtain

$$\begin{aligned} \Psi(M) - \Psi(M') &= \left(\frac{1}{m_l} + \frac{1}{m_k}\right) - \left(\frac{1}{m'_l} + \frac{1}{m'_k}\right) \\ &= \left(\frac{1}{m_l} + \frac{1}{m_k}\right) - \left(\frac{1}{m_l - 1} + \frac{1}{m_k + 1}\right) \\ &= \frac{1}{m_k(m_k+1)} - \frac{1}{m_l(m_l-1)} \\ &> 0\ . \end{aligned}$$

  The last inequality holds because $m_k < m_l - 1$ and $m_k + 1 < m_l$, since $m_l - m_k \geq 2$.

- Before the load-balancing step, job l and job j both belongs to the set $B \cup C \cup D$ of the Phase-1 partition of M. In this case, Ψ does not change.

- Before the load-balancing step, job l belongs to set $C \cup D$ of the Phase-1 partition and job k belongs to set A. In this case, job l cannot be in set A of the Phase-1 partition of the successor configuration M', and thus job l contributes 0 to the potential in both configurations. Job k will have one more processor, and therefore its contribution to the potential will decrease. Thus Ψ decreases.

In any case, we have proved that Ψ never increases. Thus, part (i) is true.

To prove part (ii), we only need to observe that or any job $j \in A$, we have $1/m_j - 1/(\epsilon\mu) < 1$. Since $\mu \geq 2$, we have $J < P$. Thus, $\Psi(M) \leq P$ for all configurations M.

For part (iii), from the definition of Ψ, we know that $\Psi(M) = 0$ if and only if A is empty, which means that a desired configuration of Phase 1 has been achieved.

To show the last property, notice that if $\Psi(M) \neq 0$, then there exists a job $j \in A$. Job j's contribution to the potential is at least

$$\begin{aligned} 1/m_j - 1/\epsilon\mu &\geq 1/(\epsilon\mu - 1) - 1/\epsilon\mu \\ &\geq 1/(\epsilon\mu)^2 \end{aligned}$$

$$> \quad P^{-2} \ .$$

Therefore, If $\Psi(M) < P^{-2}$, then $\Psi(M) = 0$. ∎

Lemma 4 *Consider a macroscheduling system with P processors. Under the SRLBA algorithm, given any configuration M, we have*

- *(i) $0 \leq \Upsilon(M) < \epsilon P < P$;*
- *(ii) if $\Upsilon(M) < 1/2$, then $\Upsilon(M) = 0$;*
- *(iii) if M' is a successor of M, then $\Upsilon(M') \leq \Upsilon(M)$;*
- *(iv) if M' is a successor of M and $\Upsilon(M) - \Upsilon(M') > 0$, then $\Upsilon(M) - \Upsilon(M') \geq 1/2$.*

Proof: Let ϵ be the Phase-1 constant. Let $\mu = P/J$ where J is the number of jobs in the macroscheduling system. Let $\langle m_1, m_2, \ldots, m_J \rangle$ be the configuration vector of M. For part (i), since for all job $j \in [J]$, $m_j \geq 1$, we have

$$\begin{aligned} \Upsilon(M) &\leq J(\epsilon\mu - 1/2) \\ &< P\epsilon \\ &< P \ . \end{aligned}$$

For part (ii), we know that if there exists a job j belonging to set $A \cup B$, then job j at least contributes $1/2$ to Υ. Thus, if $\Upsilon(M) < 1/2$, it must be 0.

As for parts (iii) and (iv), consider the load-balancing step between configuration M and its successor M'. If there is no processor migration at all, or if a processor migration takes place between two jobs that both belong to $C \cup D$ of the Phase-1 partition of configuration M, or if a processor migration takes place between two jobs that both belong to set $A \cup B$ of the Phase-1 partition of configuration M, then $\Upsilon(M') = \Upsilon(M)$.

Another case is to consider is when a migration takes place between a job j belonging to set $A \cup B$ of the Phase-1 partition of M and a job k belonging to set D of the Phase-1 partition of configuration M. Since $m_k \geq \epsilon\mu + 2$, after the migration, in the successor configuration M', job k cannot be in the set $A \cup B$. Thus, if job j belongs to set A of the Phase-1 partition of configuration M, we have $\Upsilon(M') = \Upsilon(M) - 1$; if j belongs to set B of the Phase-1 partition of configuration M, then $\Upsilon(M^{(i)}) = \Upsilon(M^{(i-1)}) - 1/2$.

Since no migration happens when a load-balancing step occurs between a job belonging to set B and a job belonging to set C, the only other case left is when a migration takes place between a job j belonging to set A of the Phase-1 partition of M and a job k belonging to set C. After the migration, job j will get one more processor, which will decrease the potential by 1; job k will be in set B of the Phase-1 partition of the successor configuration M', and it contributes 1/2 to the potential. In this case, we have $\Upsilon(M') = \Upsilon(M) - 1/2$.

Therefore, in all cases, $\Upsilon(M^{(i-1)}) - \Upsilon(M^{(i)}) \geq 0$, proving part (iii). Moreover, if $\Upsilon(M^{(i-1)}) - \Upsilon(M^{(i)}) > 0$, then $\Upsilon(M^{(i-1)}) - \Upsilon(M^{(i)}) \geq 1/2$, proving part (iv). ■

With the properties of Ψ and Υ above, we shall show in Lemma 5 and Lemma 6 that after any given step $i \in [P]$, either the expected value of Ψ decreases by a factor of $1/2P$ or the expected value of Υ decreases by a factor of $1/2P$.

Lemma 5 *In a macroscheduling system of P processors and J jobs with $\mu > 8$, consider the SRLBA algorithm. If in the load-balancing step right after configuration M is obtained, the adversary chooses a processor p which works for a job belonging to $C \cup D$ of the Phase-1 partition of configuration M, then for the successor configuration M', we have* $\mathrm{E}[\Psi(M')] \leq \Psi(M)(1 - 1/2P)$.

Proof: When $\mu > 8$, for the Phase-1 constant ϵ, we have $1/8 < \epsilon < 1/4$. For any job k in set A of the Phase-1 partition of M, processor p will choose r which belongs to job k with probability m_k/P, where m_k is the allotment for job k; if this happens, processor p migrates to work for job k, and $\Psi(M') = \Psi(M) - 1/m_k + 1/(m_k+1) = \Psi(M) - 1/m_k(m_k+1)$. Thus, for $j \in [\epsilon\mu - 1]$, if there are n_j jobs in set A of the Phase-1 partition of M with j processors each, then

$$\Psi(M) = \sum_{j=1}^{\epsilon\mu-1} n_j(1/j - 1/\epsilon\mu) ,$$

and

$$\begin{aligned}\Psi(M) - \mathrm{E}[\Psi(M')] &= \sum_{j=1}^{\epsilon\mu-1} jn_j/Pj(j+1) \\ &= \sum_{j=1}^{\epsilon\mu-1} n_j/P(j+1) .\end{aligned}$$

Thus,

$$\frac{\Psi(M) - \mathrm{E}\left[\Psi(M')\right]}{\Psi(M)} = \frac{\sum_{j=1}^{\epsilon\mu-1} n_j/P(j+1)}{\sum_{j=1}^{\mu\epsilon-1} n_j(1/j - 1/\epsilon\mu)} . \tag{3.1}$$

What choice of the n_j's minimizes the righthand side of Equation (3.1)? Recall that if $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are nonnegative constants, then the minimum value of $(\sum_j a_j x_j)/(\sum_j b_j x_j)$, subject to the condition that the x_j's are nonnegative (and that not all of them are 0), is $\min_j \{a_j/b_j\}$. If we let $x_j = n_j$, $a_j = 1/P(j+1)$ and $b_j = 1/j - 1/\epsilon\mu$, then the righthand side of Equation (3.1) is at least

$$\min_j \frac{1/P(j+1)}{1/j - 1/\epsilon\mu} = (1/P) \min_j \frac{1/(j+1)}{1/j - 1/\epsilon\mu} .$$

Now,

$$\begin{aligned} \frac{1/(j+1)}{1/j - 1/\epsilon\mu} &= \frac{\mu\epsilon j}{(j+1)(\epsilon\mu - j)} \\ &= \epsilon\mu\left(1 - \frac{1}{j+1}\right)\left(\frac{1}{\epsilon\mu - j}\right) . \end{aligned}$$

Since $1 - 1/(j+1)$ and $1/(\epsilon\mu - j)$ are both increasing functions of j, we conclude that the minimum occurs at $j = 1$. Thus, the righthand side of Equation (3.1) is at least $(1/2P)/(1 - 1/\epsilon\mu) \geq 1/2P$. Thus, $\mathrm{E}\left[\Psi(M')\right] \leq \Psi(M)(1 - 1/2P)$. ∎

Lemma 6 *In a macroscheduling system of P processors and J jobs with $\mu > 8$, consider the SRLBA algorithm. If in the load-balancing step right after configuration M is obtained, the adversary chooses a processor p which works for a job belonging to set $A \cup B$ of the Phase-1 partition of configuration M, then for the successor configuration M', we have* $\mathrm{E}\left[\Upsilon(M')\right] \leq \Upsilon(M)(1 - 1/2P)$.

Proof: If in the load-balancing step, processor p picks a processor which belongs to a job in set D of the Phase-1 partition of configuration M, then a processor migration will take place, and Υ decreases at least by 1/2. Since $\mu > 8$, we have $\epsilon < 1/4$. We obtain

$$\begin{aligned} \sum_{k \in D} m_k &\geq P - (\epsilon\mu + 1)J \\ &\geq P/2 . \end{aligned}$$

Thus, we have

$$\begin{aligned}\Upsilon(M) - \mathrm{E}\left[\Upsilon(M')\right] &\geq (1/2)\sum_{k\in D} m_k/P \\ &\geq (1/2)(P/2)/P \\ &= 1/4\ .\end{aligned}$$

By part (i) of Lemma 4, $\Upsilon(M) < \epsilon P \leq P/4$, we obtain

$$\begin{aligned}\mathrm{E}\left[\Upsilon(M')\right] &\leq \Upsilon(M) - 1/4 \\ &\leq \Upsilon(M)(1-1/P) \\ &\leq \Upsilon(M)(1-1/2P)\ .\end{aligned}$$

■

Putting together Lemma 5 and Lemma 6 we have the following lemma about the product of Ψ and Υ.

Lemma 7 *In a macroscheduling system of P processors and J jobs with $\mu > 8$, consider the SRLBA algorithm. For any given round, if $M^{(0)}$ is the initial configuration of the round and $M^{(P)}$ is the configuration at the end of the round, then we have*

$$\mathrm{E}\left[\Psi\Upsilon(M^{(P)})\right] \leq e^{-1/2}\Psi\Upsilon(M^{(0)})\ .$$

Proof: For any $i \in [P]$, let $M^{(i)}$ be the configuration at the end of step i. If in step i the adversary picks a processor p which works for a job in set $C \cup D$ of the Phase-1 partition of configuration $M^{(i-1)}$, then since Υ is nonincreasing, by Lemma 5 we have

$$\begin{aligned}\mathrm{E}\left[\Psi\Upsilon(M^{(i)})\right] &\leq \mathrm{E}\left[\Psi(M^{(i)})\Upsilon(M^{(i-1)})\right] \\ &= \mathrm{E}\left[\Psi(M^{(i)})\right]\Upsilon(M^{(i-1)}) \\ &\leq \Psi(M^{(i-1)})\Upsilon(M^{(i-1)})(1-1/2P) \\ &\leq \Psi\Upsilon(M^{(i-1)})(1-1/2P)\,.\end{aligned}$$

Similarly, if the adversary picks a processor p which works for a job in partition $A \cup B$ of configuration $M^{(i-1)}$, then since Ψ is nonincreasing, by Lemma 6 we have

$$\begin{aligned} \mathrm{E}\left[\Psi\Upsilon(M^{(i)})\right] &\leq \mathrm{E}\left[\Psi(M^{(i)})\Upsilon(M^{(i-1)})\right] \\ &= \mathrm{E}\left[\Psi(M^{(i-1)})\right]\Upsilon(M^{(i-1)}) \\ &\leq \Psi(M^{(i-1)})\Upsilon(M^{(i-1)})(1-1/2P) \\ &\leq \Psi\Upsilon(M^{(i-1)})(1-1/2P)\,. \end{aligned}$$

Thus, in any case, we have

$$\Psi(M^{(i-1)})\Upsilon(M^{(i-1)}) - \mathrm{E}\left[\Psi(M^{(i)})\Upsilon(M^{(i)})\right] \geq (1/2P)\Psi(M^{(i-1)})\Upsilon(M^{(i-1)})\,.$$

Together with Lemma 1, we obtain the desired result. ■

Lemma 7 shows that in every step of any round, the expected value of $\Psi\Upsilon$ decreases by a factor of $1/2P$. Together with the lemmas proved in Section 2.2, we obtain the following corollary.

Lemma 8 *In a macroscheduling system of P processors and J jobs, consider the SRLBA algorithm. Given any positive $\gamma < 1$, with probability at least $(1-\gamma)$, every job has at least $\epsilon\mu$ processors after $2\ln(1/\gamma) + 8\ln P$ rounds, where ϵ is the Phase-1 constant.*

Proof: If $\mu \leq 8$, we are done. Otherwise, we have $\mu > 8$ and $1/8 < \epsilon \leq 1/4$. By Lemma 3 and Lemma 4, we know that for any configuration M, we have $\Psi\Upsilon(M) < P^2$. Also, if $\Psi\Upsilon(M) < P^{-2}/2 < P^{-2}$, then $\Psi\Upsilon(M) = 0$. For $i \in \mathbf{N}$, let $M^{(i)}$ denote the configuration after round i, and let $M^{(0)}$ be the initial configuration before the first round. By Lemma 7, for any $i \in \mathbf{N}$, we have $\mathrm{E}\left[\Psi\Upsilon(M^{(i)})\right] < e^{-1/2}\Psi\Upsilon(M^{(i-1)})$. Thus, by Lemma 2, after

$$\begin{aligned} T &= \log_{e^{1/2}}(1/\gamma) + (2+2)\log_{e^{1/2}} P \\ &= 2\ln(1/\gamma) + 8\ln P \end{aligned}$$

rounds, with probability at least $1-\gamma$, we have $\Psi\Upsilon(M^{(T)}) = 0$. Notice that $\Psi(M^{(T)})\Upsilon(M^{(T)}) = 0$ if and only if $\Psi(M^{(T)}) = 0$, which means that each job has at least $\epsilon\mu$ processors. ■

In Lemma 8, if we choose γ to be $1/P^c$, for any constant c, then with probability

$(1-1/P^c)$, after $2(c+4)\ln P$ rounds, every job has at least $\mu/8$ processors. Therefore with high probability, after $O(\lg P)$ rounds, we obtain the desired Phase-1 configuration.

Chapter 4

Analysis of Phase 2 of the *SRLBA* Algorithm

In this chapter, we consider Phase 2 of the *SRLBA* algorithm. For a macroscheduling system of P processors and J jobs with Phase-1 constant ϵ, we show that under the *SRLBA* algorithm, if every job has a desire which is more than the absolute average allotment μ, and each job has at least $\epsilon\mu$ processors, then with high probability, in $O(\lg P)$ rounds, the system is in an almost fair and efficient configuration.

We first establish the potential functions that are used in the analysis of this phase. We assign a given round to one of two possible cases according to its initial potential. We analyze each of the two cases. Finally, we conclude that Phase 2 takes $O(\lg P)$ rounds.

4.1 The Potential functions for Phase 2

This section introduces the notion of "Phase-2" partition. It then describes the potential functions used in Phase 2.

In the *SRLBA* algorithm, once every job has more than $\epsilon\mu$ processors, no job will ever have less than $\epsilon\mu$ processors. Henceforth, we assume that, in Phase 2, every job has at least $\epsilon\mu$ processors in every configuration we consider, where $1/8 \le \epsilon \le 1/2$.

Let M be a configuration of the system in Phase 2. Let $\langle m_1, m_2, \ldots, m_J \rangle$ be the configuration vector of M, where $m_i = |\{p : M(p) = i\}| > \epsilon\mu$ is the processor allotment of job i. We define the ***Phase-2 partition*** of configuration M to be the following disjoint sets of jobs:

$$
\begin{aligned}
A &= \{j \in [J] : m_j < \mu - 1\} \ , \\
B &= \{j \in [J] : m_j = \mu - 1\} \ , \\
C &= \{j \in [J] : m_j = \mu\} \ , \\
D &= \{j \in [J] : m_j = \mu + 1\} \ , \\
E &= \{j \in [J] : m_j > \mu + 1\} \ .
\end{aligned}
$$

We define $\lambda_j = |m_j - \mu|$ to be the ***discrepancy*** of job j. Given the configuration M, we define the potential functions $\overline{\Lambda}$ and Λ to be

$$
\begin{aligned}
\overline{\Lambda}(M) &= \sum_{j \in A \cup E} (\lambda_j - 1) \ , \\
\Lambda(M) &= \sum_{j \in [J]} \lambda_j \ .
\end{aligned}
$$

Both $\overline{\Lambda}$ and Λ are potential functions, as they are both nonnegative. Moreover, they both are nonincreasing, have an upper bound P, and if they are not zero, they must be at least 1. The potential function $\overline{\Lambda}(M)$ represents the discrepancy of M from the almost fair and efficient configuration, whereas $\Lambda(M)$ represents the discrepancy of M from the absolute fair and efficient configuration. To simplify our notation, for any configuration M, we define

$$
\Lambda\overline{\Lambda}(M) = \Lambda(M)\overline{\Lambda}(M) \ ,
$$

which is also a potential function. Moreover, $\Lambda\overline{\Lambda}(M) = 0$ if and only if the system is in an almost fair and efficient configuration.

In this section, we shall show that during Phase 2, after each round, the expected decrease in the potential $\Lambda\overline{\Lambda}(M)$ is at least a constant fraction of what it was before entering the round. Then, using the potential-function argument established in Section 2.2, we show with high probability that after $O(\lg n)$ rounds, $\Lambda\overline{\Lambda}(M) = 0$.

To further simplify the notation, we define $\overline{\Lambda}_A$ and $\overline{\Lambda}_E$ to be

$$
\overline{\Lambda}_A(M) = \sum_{j \in A} (\lambda_j - 1) \ ,
$$

$$\overline{\Lambda}_E(M) \;=\; \sum_{j\in E}(\lambda_j - 1)\;.$$

We have $\overline{\Lambda}_A(M)+\overline{\Lambda}_E(M) = \overline{\Lambda}(M)$. The function $\overline{\Lambda}_A(M)$ represents the number of processors wanted by those jobs whose allotments are fewer than $\mu-1$, whereas $\overline{\Lambda}_E(M)$ represents the extra processors to be taken from those jobs whose allotments are more than $\mu+1$. Intuitively, if $\overline{\Lambda}_A(M)$ is much larger than $\overline{\Lambda}_E(M)$, then many jobs in the system have exactly $\mu+1$ processors; conversely, if $\overline{\Lambda}_E(M)$ is much larger than $\overline{\Lambda}_A(M)$, then many jobs in the system have exactly $\mu-1$ processors. If both $\overline{\Lambda}_A(M)$ and $\overline{\Lambda}_E(M)$ are 0, then the configuration M is an almost fair and efficient configuration.

Consider any round. If R is the initial configuration of the round, then at least one of the following two cases is true:

(a) $\overline{\Lambda}_E(R) \geq \overline{\Lambda}(R)/2$,

(b) $\overline{\Lambda}_A(R) \geq \overline{\Lambda}(R)/2$.

In the follow two subsections, we shall prove that in either of the two cases, the expected value of $\Lambda\overline{\Lambda}$ decreases by a constant fraction after a round of load-balancing steps.

4.2 Case (a)

In this section, we shall prove that if $\overline{\Lambda}_E(R) \geq \overline{\Lambda}(R)/2$ for a configuration R at the beginning of a round, then at the end of the round, the expected decrease in $\Lambda\overline{\Lambda}$ is at least a constant fraction of $\overline{\Lambda}\Lambda(R)$. The proof involves a sequence of four lemmas, in the last of which we derive the desired result.

Given any configuration M during Phase 2, the following lemma gives a lower bound on the number of processors working for jobs in set $A \cup B$ of the Phase-2 partition.

Lemma 9 *In a macroscheduling system with P processors and J jobs, during Phase 2 of the SRLBA algorithm, for any configuration M with configuration vector $\langle m_1, m_2, \ldots, m_J\rangle$, we have*

$$\sum_{j\in A\cup B} m_j \geq \frac{\epsilon}{2(1-\epsilon)}\Lambda(M)\;.$$

Proof: Since for any job $j \in A \cup B$, we have $\epsilon\mu \leq m_j \leq \mu$, it follows that

$$\begin{aligned} \lambda_j &= \mu - m_j \\ &\leq (1-\epsilon)\mu \ , \end{aligned}$$

and hence, $m_j/\lambda_j \geq \epsilon/(1-\epsilon)$. Thus, since $\sum_{J \in A \cup B} \lambda_j = \Lambda/2$, we have

$$\begin{aligned} \sum_{J \in A \cup B} m_j &\geq \frac{\epsilon}{(1-\epsilon)} \sum_{J \in A \cup B} \lambda_j \\ &= \frac{\epsilon}{2(1-\epsilon)} \Lambda \ . \end{aligned}$$

■

Given any configuration M of phase 2, the following lemma shows that if the contribution to the potential $\overline{\Lambda}$ from jobs in partition E is sufficiently large, then the expected potential of M's successor M' decreases by a factor of at least $1/7P$.

Lemma 10 *In a macroscheduling system with P processors and J jobs, during Phase 2 of the SRLBA algorithm, consider a round whose initial configuration $M^{(0)}$ satisfies the condition $\overline{\Lambda}_E(M^{(0)}) \geq \overline{\Lambda}(M^{(0)})/2$. Let M be any configuration of this round, and M' be a successor of M. If $\overline{\Lambda}_E(M) > (1/7)\overline{\Lambda}(M^{(0)})$, then*

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M')\right] \leq (1 - 1/7P)\Lambda\overline{\Lambda}(M) \ .$$

Proof: Let $\langle m_1, m_2, \ldots, m_J \rangle$ be the configuration vector of M. For job $j \in [P]$, let λ_j be the discrepancy of job j in configuration M. Suppose in the load-balancing step after configuration M, the adversary picks up a processor p that belongs to job k. We have two cases.

- In the first case, job k belongs to $A \cup B \cup C$ of the Phase-2 partition of configuration M. We call the load-balancing step a success if processor p proposes to a processor r which works for a job in E of the Phase-2 partition of M. If the load-balancing step is a success, we have $\overline{\Lambda}(M) - \overline{\Lambda}(M') \geq 1$, and $\Lambda\overline{\Lambda}(M) - \Lambda\overline{\Lambda}(M') \geq \Lambda(M)$. The probability of a success is at least

$$\begin{aligned}
\sum_{j \in E} m_j/P &> \sum_{j \in E} (\lambda_j - 1)/P \\
&= \overline{\Lambda}_E(M)/P \\
&> (1/7P)\overline{\Lambda}(M^{(0)}) \\
&\geq (1/7P)\overline{\Lambda}(M)
\end{aligned}$$

since $\overline{\Lambda}$ is nonincreasing. Thus,

$$\begin{aligned}
\Lambda\overline{\Lambda}(M) - \mathrm{E}\left[\Lambda\overline{\Lambda}(M')\right] &\geq \left((1/7P)\overline{\Lambda}(M)\right)\overline{\Lambda}(M) \\
&= (1/7P)\Lambda\overline{\Lambda}(M) \; .
\end{aligned}$$

- In the second case, job k belongs to $D \cup E$ of the Phase-2 partition of configuration M. In this case, we call the load-balancing step a success if processor p chooses processor $r \in A \cup B$ of the Phase-2 partition of M. If the load-balancing step is a success, we have $\Lambda(M) - \Lambda(M') = 2$, and $\Lambda\overline{\Lambda}(M) - \Lambda\overline{\Lambda}(M') \geq 2\overline{\Lambda}(M)$. Since in Phase 2, every job has at least $\epsilon\mu$ processors, by Lemma 9, we have

  $$\sum_{j \in A \cup B} m_j \geq \frac{\epsilon}{2(1-\epsilon)}\Lambda(M) \; .$$

  Since, out of the P total processors, there are at least $(\epsilon/2(1-\epsilon))\Lambda(M)$ processors in $A \cup B$, the probability of a success is at least $(\epsilon/2(1-\epsilon))\Lambda(M)/P$. Therefore, the expected decrease in $\Lambda\overline{\Lambda}$ is

  $$\Lambda\overline{\Lambda}(M) - \mathrm{E}\left[\Lambda\overline{\Lambda}(M')\right] \geq \frac{\epsilon}{1-\epsilon}\Lambda\overline{\Lambda}(M)/P \; .$$

When $1/8 \leq \epsilon \leq 1/2$, we have $\min\{\epsilon/(1-\epsilon), 1/7\} = 1/7$, which completes the proof. ∎

In a given round, let $M^{(0)}$ be the initial configuration, and for any $i \in [P]$, let $M^{(i)}$ be the configuration after step i of the round. For each $i \in \{0, 1, \ldots, P-1\}$, define the events $X^{(i)}$ and $Y^{(i)}$ as follows:

$$X^{(i)} = \left\{\overline{\Lambda}_E(M^{(i)}) \geq \overline{\Lambda}(M^{(0)})/7 \text{ and } \Lambda\overline{\Lambda}(M^{(i-1)}) > \Lambda\overline{\Lambda}(M^{(0)})/2\right\} \; ,$$

$$Y^{(i)} \quad = \quad \left\{\overline{\Lambda}_E(M^{(i)}) < \overline{\Lambda}(M^{(0)})/7 \text{ or } \Lambda\overline{\Lambda}(M^{(i-1)}) \leq \Lambda\overline{\Lambda}(M^{(0)})/2\right\} \ .$$

By definition $\Pr\left\{X^{(i)}\right\} + \Pr\left\{Y^{(i)}\right\} = 1$, for every step $i = 0, 1, \ldots, P-1$. The following lemma shows the relationship between the expected decrease of $\Lambda\overline{\Lambda}$ and the events $X^{(i)}$'s, for $i = 0, 1, \ldots, P-1$.

Lemma 11 *In a macroscheduling system of P processors and J jobs, consider a round of Phase 2 under the SRLBA algorithm. Suppose $M^{(0)}$, the initial configuration of the round, satisfies the condition $\overline{\Lambda}_E(M^{(0)}) \geq \overline{\Lambda}(M^{(0)})/2$. For $i \in [P]$, let $M^{(i)}$ be the configuration after step i. Then, we have*

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i)})\right] \leq \mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)})\right] - \Pr\left\{X^{(i-1)}\right\} \Lambda\overline{\Lambda}(M^{(0)})/(14P) \ ,$$

and

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(P)})\right] \leq \Lambda\overline{\Lambda}(M^{(0)}) \left(1 - \left(\sum_{i=0}^{P-1} \Pr\left\{X^{(i-1)}\right\}\right) /14P\right) \ .$$

Proof: In the beginning of step i, if the event $X^{(i-1)}$ is true, that is, $\overline{\Lambda}_E(M^{(i-1)}) \geq \overline{\Lambda}(M^{(0)})/7$ and $\Lambda\overline{\Lambda}(M^{(i-1)}) > \Lambda\overline{\Lambda}(M^{(0)})/2$, then by Lemma 10, the following must hold:

$$\begin{aligned}
\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)}) - \Lambda\overline{\Lambda}(M^{(i)})\right] &\geq (1/7P)\Lambda\overline{\Lambda}(M^{(i-1)}) \\
&\geq (1/14P)\Lambda\overline{\Lambda}(M^{(0)}) \ ,
\end{aligned}$$

which is equivalent to

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i)})\right] < \mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)})\right] - (1/14P)\Lambda\overline{\Lambda}(M^{(0)}) \ .$$

Otherwise, the event $X^{(i-1)}$ is not true. In this case, using the fact that the potential function $\Lambda\overline{\Lambda}$ is nonincreasing, we have $\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i)})\right] \leq \mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(0)})\right]$. Thus, in general, we have

$$\begin{aligned}
\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i)})\right] &\leq \Pr\left\{X^{(i-1)}\right\}\left(\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)})\right] - (1/14P)\Lambda\overline{\Lambda}(M^{(0)})\right) \\
&\quad + \left(1 - \Pr\left\{X^{(i-1)}\right\}\right)\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)})\right] \\
&= \mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)})\right] - \Pr\left\{X^{(i-1)}\right\}\Lambda\overline{\Lambda}(M^{(0)})/(14P) \ .
\end{aligned}$$

By induction, we obtain

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(P)})\right] \leq \Lambda\overline{\Lambda}(M^{(0)})\left(1-\left(\sum_{i=0}^{P-1}\Pr\left\{X^{(i-1)}\right\}\right)/(14P)\right) .$$

∎

The next lemma proves that, given any configuration R in the beginning of a round, if $\overline{\Lambda}_E(R) \geq \Lambda(R)$, then at the end of the round, the expected decrease in $\Lambda\overline{\Lambda}$ is at least a constant fraction of $\overline{\Lambda}\Lambda(R)$.

Lemma 12 *In a macroscheduling system with P processors and J jobs, consider a round of Phase 2 under the SRLBA algorithm. Suppose $\overline{\Lambda}_E(R) \geq \overline{\Lambda}(R)/2$, where R is the initial configuration of the round. If R' is a configuration at the end of the round, then*

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] \leq (27/28)\Lambda\overline{\Lambda}(R) .$$

Proof: In the beginning of the round, we have $\overline{\Lambda}_E(R) \geq \overline{\Lambda}(R)/2$. If at the end of the round the event $Y^{(P)}$ is true, then either $\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}(R)/2$ or $\overline{\Lambda}_E(R') < \Lambda(R)/7$. If $\overline{\Lambda}_E(R') < \Lambda(R)/7$, then the decrease in $\overline{\Lambda}$ is at least

$$(1/2)\overline{\Lambda}(R) - (1/7)\overline{\Lambda}(R) = (5/14)\overline{\Lambda}(R) .$$

Thus, $\Lambda\overline{\Lambda}(R') \leq \frac{9}{14}\Lambda\overline{\Lambda}(R)$. Otherwise, $\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}(R)/2$. In either case we have

$$\Lambda\overline{\Lambda}(R') \leq \frac{9}{14}\Lambda\overline{\Lambda}(R) ,$$

since $\max\{9/14, 1/2\} = 9/14$. If the event $Y^{(P)}$ is not true , since the potential function $\Lambda\overline{\Lambda}$ is nonincreasing, we have $\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}$. So, when $\Pr\left\{Y^{(P)}\right\} \geq \epsilon$, we have

$$\begin{aligned}
\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] &\leq (1-\Pr\left\{Y^{(P)}\right\})\Lambda\overline{\Lambda}(R) + \Pr\left\{Y^{(P)}\right\}\frac{9}{14}\Lambda\overline{\Lambda}(R) \\
&= \Lambda\overline{\Lambda}(R) - \Pr\left\{Y^{(P)}\right\}\frac{5}{14}\Lambda\overline{\Lambda}(R) \\
&\leq \Lambda\overline{\Lambda}(R) - \frac{5}{14}\epsilon\Lambda\overline{\Lambda}(R)
\end{aligned}$$

$$= \left(1 - \frac{5}{14}\epsilon\right) \Lambda\overline{\Lambda}(R) \ .$$

Now consider the case where $\Pr\left\{Y^{(P)}\right\} < \epsilon$. Since $\overline{\Lambda}_E$, $\overline{\Lambda}$, and Λ are nonincreasing, we have $\Pr\left\{Y^{(i-1)}\right\} \leq \Pr\left\{Y^{(i)}\right\}$ for all $i \in [P]$. Because $\Pr\left\{Y^{(P)}\right\} < \epsilon$, we have $\Pr\left\{Y^{(i-1)}\right\} < \epsilon$ for all $i \in [P]$. Since $\forall i \in [P]$, $\Pr\left\{X^{(i-1)}\right\} + \Pr\left\{Y^{(i-1)}\right\} = 1$, we obtain $\sum_{i=0}^{P-1} \Pr\left\{X^{(i)}\right\} > (1-\epsilon)P$. Using this and Lemma 11, we have

$$\begin{aligned} \mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] &\leq \left(1 - \frac{(1-\epsilon)P}{14P}\right) \Lambda\overline{\Lambda}(R) \\ &\leq \frac{13+\epsilon}{14}\Lambda\overline{\Lambda}(R) \ . \end{aligned}$$

Since the Phase-1 constant ϵ satisfies the condition $1/8 \leq \epsilon \leq 1/2$, we have

$$\max\left\{(13+\epsilon)/14, (1 - 5\epsilon/14)\right\} \leq 27/28 \ .$$

It follows that

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] \leq \frac{27}{28}\Lambda\overline{\Lambda}(R) \ .$$

■

4.3 Case (b)

In this subsection, we shall prove that, for a configuration R at the beginning of a round, if $\overline{\Lambda}_A(R) \geq \overline{\Lambda}(R)/2$, then at the end of the round, the expected decrease in $\Lambda\overline{\Lambda}$ is at least a constant fraction of $\overline{\Lambda}\Lambda(R)$. The proof involves a sequence of four lemmas, in the last of which we derive the desired result.

Given any configuration M of phase 2, the following lemma gives a lower bound on the number of processors working for jobs in the partition $A \cup B$ in terms if $\Lambda_A(M)$.

Lemma 13 *In a macroscheduling system with P processors and J jobs, during Phase 2 of the SRLBA algorithm, for any configuration M with the configuration vector $\langle m_1, m_2, \ldots, m_J\rangle$, we have*

$$\sum_{j \in A} m_j \geq \frac{\epsilon}{1-\epsilon}\overline{\Lambda}_A(M) \ .$$

Proof: Since for any job $j \in A$, we have $m_j \geq \epsilon\mu$, it follows that

$$\begin{aligned} \lambda_j - 1 &< \lambda_j \\ &= \mu - m_j \\ &\leq (1-\epsilon)\mu \ , \end{aligned}$$

and hence, $m_j/(\lambda_j - 1) \geq \epsilon/(1-\epsilon)$. Thus, we have

$$\begin{aligned} \sum_{J \in A} m_j &\geq \frac{\epsilon}{(1-\epsilon)} \sum_{J \in A} (\lambda_j - 1) \\ &= \frac{\epsilon}{1-\epsilon} \overline{\Lambda}_A(M) \ . \end{aligned}$$

■

Given any configuration M during phase 2, the following lemma shows that if the contribution to the potential $\overline{\Lambda}$ from jobs in partition A is sufficiently big, then the expected potential of M's successor M' decreases by a factor of at least $1/28P$.

Lemma 14 *In a macroscheduling system with P processors and J jobs, during Phase 2 of the SRLBA algorithm, consider a round whose initial configuration $M^{(0)}$ satisfies the condition $\overline{\Lambda}_A(M^{(0)}) \geq \overline{\Lambda}(M^{(0)})/2$. Let M be any configuration of this round, and M' be a successor of M. If $\overline{\Lambda}_A(M) > \overline{\Lambda}(M^{(0)})/4$, then*

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M')\right] \leq (1 - 1/28P)\Lambda\overline{\Lambda}(M) \ .$$

Proof: Let $\langle m_1, m_2, \ldots, m_P \rangle$ be the configuration vector of M. For job $j \in [P]$, let λ_j be the discrepancy of job j in configuration M. Suppose that during the load-balancing step following configuration M, the adversary chooses a processor p belonging to job k to initiate the load-balancing step. We have two cases.

- In the first case, job $k \in C \cup D \cup E$ of the Phase-2 partition of configuration M. We call the load-balancing step a success if p proposes to processor r which works for a job in A of the Phase-2 partition of M. If the load-balancing step is a success, we have

$$\overline{\Lambda}(M) - \overline{\Lambda}(M') \geq 1$$

and

$$\Lambda\overline{\Lambda}(M) - \Lambda\overline{\Lambda}(M') \geq \Lambda(M) \ .$$

By Lemma 13 the probability of a success is at least

$$\begin{aligned}\sum_{j \in A} m_j/P &> \frac{\epsilon}{1-\epsilon}\overline{\Lambda}_A(M)/P \\ &> \frac{\epsilon}{4(1-\epsilon)}\overline{\Lambda}(M^{(0)})/P \\ &> \frac{\epsilon}{4(1-\epsilon)}\overline{\Lambda}(M)/P \ .\end{aligned}$$

Thus,

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M)\right] - \Lambda\overline{\Lambda}(M') \geq \frac{\epsilon}{4(1-\epsilon)}\Lambda\overline{\Lambda}(M)/P \ .$$

- In the second case, job $k \in A \cup B$ of the Phase-2 partition of configuration M. In this case, we call the load-balancing step a success if p proposes to a processor r working for a job in $C \cup D \cup E$ of the Phase-2 partition of M. If the load-balancing step is a success, we have

  $$\Lambda(M) - \Lambda(M') = 2$$

  and

  $$\Lambda\overline{\Lambda}(M) - \Lambda\overline{\Lambda}(M') \geq 2\overline{\Lambda}(M) \ .$$

  Since

  $$\begin{aligned}\sum_{j \in C \cup D \cup E} m_j &\geq \sum_{j \in C \cup D \cup E} \lambda_j \\ &= \Lambda(M)/2 \ ,\end{aligned}$$

  the probability of a success is at least $\Lambda(M)/2P$. Therefore, the expected decrease in $\Lambda\overline{\Lambda}$ is

  $$\Lambda\overline{\Lambda}(M) - \mathrm{E}\left[\Lambda\overline{\Lambda}(M')\right] \geq \Lambda\overline{\Lambda}(M)/P \ .$$

Since the Phase-1 constant ϵ satisfies $1/8 \leq \epsilon \leq 1/2$, we obtain $\min\{\epsilon/4(1-\epsilon), 1\} \geq 1/28$.

Thus, we complete the proof. ∎

In a given round, let $M^{(0)}$ be the initial configuration, and for any $i \in [P]$, let $M^{(i)}$ be the configuration after step i of the round. For each $i \in \{0, 1, \ldots, P-1\}$, define the events $X^{(i)}$ and $Y^{(i)}$ to be

$$\begin{aligned} X^{(i)} &= \left\{\overline{\Lambda}_A(M^{(i)}) > \overline{\Lambda}(M^{(0)})/4 \text{ and } \Lambda\overline{\Lambda}(M^{(i)}) > \Lambda\overline{\Lambda}(M^{(0)})/2\right\} , \\ Y^{(i)} &= \left\{\overline{\Lambda}_A(M^{(i)}) \leq \overline{\Lambda}(M^{(0)})/4 \text{ or } \Lambda\overline{\Lambda}(M^{(i)}) \leq \Lambda\overline{\Lambda}(M^{(0)})/2\right\} . \end{aligned}$$

By definition $\Pr\left\{X^{(i)}\right\} + \Pr\left\{Y^{(i)}\right\} = 1$, for every step $i = 0, 1, \ldots, P-1$. The follow lemma shows the relationship between the expected decrease of $\Lambda\overline{\Lambda}$ and the events $X^{(i)}$'s, for $i = 0, 1, \ldots, P-1$.

Lemma 15 *In a macroscheduling system of P processors and J jobs, consider a round of Phase 2 under the SRLBA algorithm. Suppose $M^{(0)}$, the initial configuration of the round, satisfies the condition $\overline{\Lambda}_A(M^{(0)}) \geq \overline{\Lambda}(M^{(0)})/2$. For $i \in [P]$, if $M^{(i)}$ is the configuration after step i, then we have*

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i)})\right] \leq \mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(i-1)})\right] - \Pr\left\{X^{(i-1)}\right\}\Lambda\overline{\Lambda}(M^{(0)})/56P ,$$

and

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(M^{(P)})\right] \leq \Lambda\overline{\Lambda}(M^{(0)})\left(1 - \left(\sum_{i=0}^{P-1}\Pr\left\{X^{(i-1)}\right\}\right)/56P\right) .$$

Proof: Similar to the proof of Lemma 11. ∎

The next lemma proves that, given any configuration R in the beginning of a round, if $\overline{\Lambda}_A(R) \geq \Lambda(R)$, then at the end of the round, the expected decrease of $\Lambda\overline{\Lambda}$ is at least a constant fraction of $\overline{\Lambda}\Lambda(R)$.

Lemma 16 *In a macroscheduling system with P processors and J jobs, consider a round of Phase 2 under the SRLBA algorithm. Suppose $\overline{\Lambda}_A(R) \geq \overline{\Lambda}(R)/2$, where R is the initial configuration of the round. If R' is a configuration at the end of the round, then*

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] \leq \frac{111}{112}\Lambda\overline{\Lambda}(R) .$$

Proof: In the beginning of the round, we have $\overline{\Lambda}_A(R) \geq \overline{\Lambda}(R)/2$. If at the end of the round the event $Y^{(P)}$ is true, then either $\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}(R)/2$ or $\overline{\Lambda}_A(R') \leq \overline{\Lambda}(R)/4$. If $\overline{\Lambda}_A(R') \leq \overline{\Lambda}(R)/4$, then the decrease in $\overline{\Lambda}$ is at least $\overline{\Lambda}(R)/4$, and hence,

$$\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}(R)/4 \ .$$

Since $\max\{1/2, 1/4\} = 1/2$, we have

$$\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}(R)/2 \ .$$

If the event $Y^{(P)}$ is not true , since the potential function $\Lambda\overline{\Lambda}$ is nonincreasing, we have $\Lambda\overline{\Lambda}(R') \leq \Lambda\overline{\Lambda}(R)$. So, if $\Pr\left\{Y^{(P)}\right\} \geq \epsilon$, we have

$$\begin{aligned}
\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] &\leq (1 - \Pr\left\{Y^{(P)}\right\})\Lambda\overline{\Lambda}(R) + \Pr\left\{Y^{(P)}\right\}\Lambda\overline{\Lambda}(R)/2 \\
&= \Lambda\overline{\Lambda}(R) - \Pr\left\{Y^{(P)}\right\}\Lambda\overline{\Lambda}(R)/2 \\
&\leq (1 - \epsilon/2)\,\Lambda\overline{\Lambda}(R) \ .
\end{aligned}$$

Now consider the case where $\Pr\left\{Y^{(P)}\right\} < \epsilon$. Since $\overline{\Lambda}_E$, $\overline{\Lambda}$ and Λ are nonincreasing, we have $Y^{(i-1)} \leq Y^{(i)}$ for all $i \in [P]$. Since $Y^{(P)} < \epsilon$, we have $Y^{(i-1)} < \epsilon$ for all $i \in [P]$. Since $\forall i \in [P]$, $X^{(i-1)} + Y^{(i-1)} = 1$, we have

$$\sum_{i=0}^{P-1} x^{(i)} > (1-\epsilon)P \ .$$

Using this and Lemma 15, we obtain

$$\begin{aligned}
\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] &\leq \left(1 - \frac{(1-\epsilon)P}{56P}\right)\Lambda\overline{\Lambda}(R) \\
&\leq \frac{55+\epsilon}{56}\Lambda\overline{\Lambda}(R) \ .
\end{aligned}$$

Since the Phase 1 constant ϵ satisfies the condition $1/8 \leq \epsilon \leq 1/2$, we obtain

$$\max\{(55+\epsilon)/56, (1-\epsilon/2)\} \leq 111/112 \ ,$$

and thus, in either case, we have

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(R')\right] \le (111/112)\Lambda\overline{\Lambda}(R) \ .$$

■

4.4 Combined analysis of cases (a) and (b)

Combine Lemma 12 and Lemma 16, we obtain the following lemma for Phase 2.

Lemma 17 *In a macroscheduling system of P processors and J jobs, under the SRLBA algorithm, once all jobs have no less than $\epsilon\mu$ processors, with probability at lest $1-\delta$, it takes $(\ln(112/111))^{-1}(\ln(1/\gamma)+2\ln P)$ rounds for the system to converge to an almost fair and efficient configuration.*

Proof: For $i \in \mathbf{N}$, let $R^{(i)}$ denote the configuration of the system after round i, and let $R^{(0)}$ denote the initial configuration of the system. Since $111/112 > 27/28$, by Lemma 12 and Lemma 16, we obtain

$$\mathrm{E}\left[\Lambda\overline{\Lambda}(R^{(i)})\right] \le (111/112)\Lambda\overline{\Lambda}(R^{(i-1)}) \ .$$

Also, for any configuration R, we have $\Lambda\overline{\Lambda}(R) < P^2$. And if $\Lambda\overline{\Lambda}(R) < 1 = P^0$, then $\Lambda\overline{\Lambda}(R) = 0$. By Lemma 2, after

$$\begin{aligned} T &= \log_{112/111}(1/\gamma) + (2+0)\log_{112/111} P \\ &= \left(\ln\frac{112}{111}\right)^{-1}(\ln(1/\gamma)+2\ln P) \end{aligned}$$

rounds, with probability at least $1-\gamma$, we have $\Lambda\overline{\Lambda} = 0$, which implies that the system converges to an almost fair and efficient configuration. ■

Chapter 5

Conclusion and Future Work

In this Chapter I conclude my thesis with the main result of the analysis of the *SRLBA* algorithm. I briefly describe the result when some of the jobs desires are smaller than the absolute average allotment. Then, I discuss some directions for future work under the imperfect information model. Finally, I briefly describe a system implementation of the adaptive processor allocator for the multithreaded language Cilk[1, 2, 6, 13].

Theorem 18 *In a macroscheduling system of P processors and J jobs, if each job has a desire which is more than the absolute average allotment P/J, then given any initial configuration, under the SRLBA algorithm, the system arrives at an almost fair and efficient configuration after $O(\lg P)$ rounds with probability at least $1 - 1/P$.*

Proof: Combine Lemma 8 and Lemma 17. ■

Theorem 18 uses an simplifying assumption that each job has a desire which is more than the absolute average allotment of the macroscheduling system. Although this assumption may not hold in a real system, the method we used to prove the theorem under this assumption is still applicable to situations where some jobs have smaller desires than the absolute average allotment. I have a sketch of an analysis of the *SRLBA* algorithm without this simplifying assumption which, at this time, is not ready for publication. The analysis employs the same potential function arguments with two phases, except that the second phase is more complicated than without the simplifying assumption. The result is as follows.

Consider a macroscheduling system with P processors. Let Q be the maximum total number of processors which work for jobs whose desires are more than their allotment in any fair and efficient configuration of the system. Then with probability at least $1 - 1/P$, within $O(P \lg P/Q)$ rounds, the system converges to the almost fair and efficient configuration. Moreover, if Q is a constant fraction of P, we obtain that within $O(\lg P)$ rounds, the system is in a fair and efficient configuration with high probability.

Another important direction of work is to extend our analysis to a system with imperfect information. In the sequential perfect-information model, we assume that the load-balancing steps occur in serial and that for each processor, information on job allotment is updated promptly. In a real distributed system, however, it is very hard to serialize the load-balancing steps and update the information of a job's allotment quickly.

Some related work has been done by Robert Blumofe and David Zuckerman on a case of the parallel imperfect information model [3]. They consider a system of P processors with two parallel jobs both with infinite processor desires. Their algorithm proceeds in rounds, each of which consists P parallel load-balancing steps. A load-balancing step is modified so that the processor working on the job with larger allotment migrates to work on the job with smaller allotment with a certain "damping probability". This damping probability depends on the allotments of the two jobs. They show that if the allotments of the jobs are updated at the end of every round, the two-job system converges to the fair and efficient configuration after $O(\lg P)$ rounds, where P is the number of processors.

Our analysis of the *SRLBA* algorithm, together with the two-job imperfect information analysis, form a first step in the analysis of the more general and difficult processor allocation problem. We hope that our analysis or the methods employed therein may provide some insight into a general and practically implementable solution to the adaptively parallel processor allocation problem.

I have implemented an adaptive processor allocator for the multithreaded language Cilk [1, 2, 6, 13] base on the work presented in this thesis. We call this processor allocator the Cilk Macroscheduler, in contrast to the Cilk runtime system internal scheduler, the microscheduler. Traditionally, in order to run a Cilk program, the user must specify the number of processes that the program uses. With the Macroscheduler, the system allocates processors to concurrently running Cilk jobs in a fair and efficient fashion. As the current version of Cilk runs on an SMP (symmetric multiprocessor), this Macroscheduler is able to use a centralized algorithm instead of a distributed one.

One crucial part of the implementation the manner in which the Macroscheduler obtains the processor desire of user programs. The Macroscheduler gets this information through the interaction with the Cilk microscheduler. In a Cilk job, processors that are idle attempt to "steal" work from other processors. If the allotment of a job is more than its desire, then its processors spend a large portion of the total running time stealing. If the allotment is much less than the desire, then they will not spend much time stealing. Thus, we can use the steal information provided by the microscheduler to estimate a job's desire. We have both theoretical and empirical results to support this heuristic. This idea of using steal information to estimate the parallelism is based on an idea due to Charles Leiserson and Robert Blumofe. Charles Leiserson and I are currently preparing a separate paper about the Cilk Macroscheduler for publication.

Bibliography

[1] Robert D. Blumofe. *Executing Multithreaded Programs Efficiently.* PhD thesis, Massachusettes Institute of Technology, September 1995.

[2] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yli Zhou. Cilk: An efficient multithreaded runtime system. *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Proctice of Parallel Programming (PPoPP)*, pages 207–216, July 1995.

[3] Robert D. Blumofe and David Zuckerman, 1996. Private communication.

[4] S.-H. Chiang, R. K. Mansharamani, and M. K. Vernon. Use of application characteristics and limited preemption for run-to-completion parallel processor scheduleing policies. *Proceedings of the ACM SIGMETRICS Conference*, June 1988.

[5] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[6] Matteo Frigo, Charles E. Leiserson, and Keith H. Randall. The implementation of the Cilk-5 multithreaded language. Submitted for publication, 1997.

[7] S. T. Leutenegger and M. K. Vernon. The performance of multiprogrammed multiprocessor scheduling policies. *Proceedings of the ACM SIGMETRICS Conference*, May 1990.

[8] Philip A. Lisiecki. Macro-level scheduling in the Cilk network of workstations environment. Master's thesis, Massachusettes Institute of Technology, May 1996.

[9] Virginia Lo, Kurt Windisch, Wanqian Liu, and Bill Nitzberg. Non-contiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Computing*, 1996.

[10] S. Majumdar, D. L. Eager, and R. B. Bunt. Scheduling in multiprogrammed parallel systems. *Proceedings of the ACM SIGMETRICS Conference*, May 1988.

[11] Cathy McCann, Raj Vaswani, and John Zahorjan. A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors. *ACM Trans. on Computer Systems*, 11(2):146–178, May 1993.

[12] K. C. Sevcik. Characterization of prallelism in applications and their use in scheduling. *Proceedings of the ACM SIGMETRICS Conference*, May 1989.

[13] Supercomputing Technologies Research Group, MIT Laboratory for C omputer Science, 545 Technology Square, Cambridge, Massachusetts 02139. *Cilk 5.1 Reference Manual*, September 1997. *http://theory.lcs.mit.edu/~cilk*.

[14] Andrew Tucker and Anoop Gupta. Process control and scheduling issues for multiprogrammed shared-memory multiprocessors. *Proceedings of the 12th ACM Symposium on Operating System Principles*, pages 159–166, December 1989.