Fault-Tolerant Scheduling

Bala Kalyanasundaram[*]
Computer Science Department
University of Pittsburgh
kalyan@cs.pitt.edu

Kirk R. Pruhs[†]
Computer Science Department
University of Pittsburgh
kirk@cs.pitt.edu

Abstract

We study fault-tolerant multiprocessor scheduling under the realistic assumption that the occurrence of faults can not be predicted. The goal in these problems is to minimize the delay incurred by the jobs. Since this is an online problem we use competitive analysis to evaluate possible algorithms. For the problems of minimizing the makespan, and minimizing the average completion time (for static release times), we give nonclairvoyant algorithms (both deterministic and randomized) that have provably asymptotically optimal competitive ratios. The main tool used by these algorithms to combat faults is redundancy. We also show that randomization has the same effect as redundancy.

[*]Supported in part by NSF under grant CCR-9202158.
[†]Supported in part by NSF under grant CCR-9209283.

1 Introduction

1.1 Problem Statement

The scheduling of tasks in a multiprocessor system has been recognized as an important problem and has been extensively studied (see [6] for a survey). In a large multiprocessor system, processor faults are inevitable and fault-tolerance is a significant issue [11]. The vast majority of previous work on scheduling either assumes that there are no faults, or gives minimal analysis. In this paper we begin a theoretical investigation of the effect of processor faults on scheduling by considering several standard scheduling problems modified to allow faults. We assume that the pattern of faults can not be predicted by the online scheduling algorithm. We then compare the schedules produced by the online algorithm to the schedules produced by an omniscient algorithm.

The setting for the generic multiprocessor scheduling problem is a collection $P_1, \ldots, P_m$ of processors. Each processor P_j has a speed s_j. The processors are given a collection $J_1, \ldots J_n$ of jobs. Each job J_i has a *release time* r_i that is the time that the online scheduling algorithm is first aware of J_i's existence, and is the earliest time that J_i can begin execution. Furthermore, each J_i has a length x_i, and running J_i on P_j takes x_i/s_j units of time.

There are many variants of the multiprocessor scheduling problem depending on what assumptions one makes about the jobs and processors, and how one measures the desirability of a schedule. If all release times are 0, we say the release times are *static*. Otherwise, the release times are said to be *dynamic*. In the *identical processors case* all the processor speeds are equal. Otherwise, it is called the *related processors case*. A scheduling algorithm is *clairvoyant* if it learns each x_i at time r_i, and is *nonclairvoyant* if x_i can not be deduced until J_i has been fully executed. A *preemptive* algorithm is allowed to suspend an executing job and later restart the job from the point of suspension. A *preemptive-with-restart* algorithm must begin the job afresh after suspending it. The *completion time* c_i of a job J_i is the time at which J_i has been allocated enough time to finish execution. The *completion time* of J_i is $w_i = c_i - r_i$. In the paper we primarily consider two scheduling measures, the *makespan* of a schedule, which is the maximum completion time, and the *average completion time*, which is $\frac{1}{n}\sum_{i=1}^{n} w_i$.

We assume that when a processor P_j faults it is immediately detectable, and that the job currently being run on P_j must be restarted from the beginning at some later point in time. A fault at P_j can be classified as either *permanent*, in which case no more jobs may be run on P_j, or *transient*, in which case P_j is inoperative for some finite period of time [11]. Note, *nonclairvoyant* algorithm (a.k.a. *online* algorithm) detects faults only at the time of their occurrence.

The *competitiveness* (or *competitive ratio*) of a deterministic algorithm for a particular problem and measure $\mathcal{M}$ is the supremum over all instances $\mathcal{I}$, of the ratio of the value of $\mathcal{M}$ for the schedule produced by the online algorithm given $\mathcal{I}$ as input to the optimal value of $\mathcal{M}$ for a schedule of $\mathcal{I}$ [9]. We can assume that the optimal value of $\mathcal{M}$ was computed by an off-line algorithm with full advance knowledge of all the information about the jobs and the pattern of faults. For randomized algorithms we assume an oblivious adversary [2], that is, the input (length of jobs and where, when and how long fault happens) must be specified in advance, and may not be modified as a result of random events internal to the randomized algorithm. Thus the competitive ratio of a randomized algorithm on a particular problem and measure $\mathcal{M}$ is the supremum over all instances $\mathcal{I}$, of the ratio of the expected value of $\mathcal{M}$ for the schedule produced by the online algorithm given $\mathcal{I}$ as input to the optimal value of $\mathcal{M}$ for a schedule of $\mathcal{I}$.

The online algorithms that we give are nonclairvoyant, although, because our lower bound constructions all use equal length jobs, we can conclude that clairvoyance would not asymptotically help the online scheduling algorithm in the worst case. We always assume preemption-with-restart. As in most settings where fault-tolerance is an issue, the main tool available to combat faults is redundancy [11]. In this setting this generally means running multiple copies of the same job on different processors.

1.2 Results

The main results for the case of identical processors and preemption-with-restart are summarized in the table below. We use λ to denote the total number of faults the scheduler will see over the entire length of its schedule. In the case of permanent faults we use $\eta = m - \lambda$ to denote the number of nonfaulty processors. The constant ϵ satisfies $0 < \epsilon < 1$. So as to determine the effect of the duration of faults on competitiveness, we assume that the duration of each transient fault is 0. That is, after a processor P_j experiences a transient fault, a job may immediately be restarted on P_j. The results in this table suggest that for a moderate number (say less than ϵm) of faults, the duration of the faults does not asymptotically effect the achievable competitive ratios. The function $\log^* x$ is defined to be the minimum i such that $\log_2^{(i)} x \leq 1$, where $\log_2^{(i)}$ is the log function iterated i times.

Optimal Competitive Ratios for Identical Processors			
	Makespan		Average Completion Time (Static Case)
Faults	Deterministic	Randomized	Deterministic and Randomized
$\lambda = 0$	$\Theta(1)$[10]	$\Theta(1)$ [10]	$\Theta(1)$ [8]
$\lambda > 0$ Permanent	$\Theta(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$	$\Theta(\max(\log^* m - \log^* \frac{m}{\lambda}, \frac{m}{\eta}))$	$\Theta(\frac{m}{\eta})$
$\lambda = \epsilon m$ Permanent	$\Theta(\frac{\log m}{\log \log m})$	$\Theta(\log^* m)$	$\Theta(1)$
$\lambda > 0$ Transient	$\Theta(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{\lambda}{m}))$	$\Theta(\max(\log^* m - \log^* \frac{m}{\lambda}, \frac{\lambda}{m}))$	$\Theta(\max(\frac{\lambda}{m}, 1))$

The results for no faults come from [10, 8]. If one expects that permanent faults are largely independent, then the number of faults is likely to be approximately ϵm, where the constant ϵ is the probability that a particular processor faults. So in practice the most relevant line of the table is probably where $\lambda = \epsilon m$; Note that the multiplicative constants here depend on ϵ. These results show that the competitive ratios are effectively bounded in practice. The competitive ratio for average completion time is a constant, and the randomized competitive ratio for makespan is effectively bounded since $\log^* m$ is such a slowly growing function.

In the general related processors case, almost all the optimal competitive ratios listed in the table for permanent faults increase by a multiplicative factor of R, where R is the ratio of the speed of the fastest processor to the speed of the slowest processor. For no faults, [10] showed a $\Theta(\min(\log R, \log m))$ bound on the optimal deterministic competitive ratio, with respect to makespan, in the related processors case. In [8] it is implicitly shown that there exists a constant competitive algorithm for minimizing the average completion time in the general related processors case. The result is only explicitly stated for the identical processors case. This shows that as the system becomes more heterogeneous, the degradation of the optimal competitive ratio is much more rapid in the case of permanent faults.

The results for makespan with permanent faults are given in section 2, the results for average completion time with permanent faults are given in section 3. In section 4, we consider transient faults for the identical processors case. In section 5, we consider the effect of not allowing the online algorithm to run multiple copies of the same job at the same time. It is at least possible that this may be required in some situations where the programs have side effects or use external resources. We show that this restriction really cripples deterministic algorithms in that the optimal competitive ratio for both makespan and average completion time is $\Theta(\lambda)$, for permanent faults. We also show that the bounds on the competitive ratio, with respect to both makespan and average completion time, for randomized algorithms in the case of no redundancy are exactly the same as for deterministic algorithms that use redundancy.

We know of no previous theoretical investigations of this kind into fault-tolerant scheduling. Nonclairvoyant scheduling without faults is discussed in [5, 7, 8, 10]. In [8] it is shown that with dynamic release times and without faults, a nonclairvoyant deterministic (randomized) algorithm can not be better than $\Omega(n^{1/3})$-competitive ($\Omega(\log n)$-competitive), with respect to average completion time, even allowing preemptions. This shows that online scheduling algorithms face a much more daunting task when trying to minimize the average completion time of jobs with dynamic release times. For a general survey of fault tolerant scheduling see [11].

2 Makespan

2.1 Lower Bounds

We use the following adversary to prove a lower bound on the competitive ratio, with respect to makespan, of any deterministic online algorithm. We define the *multiplicity* $\phi(J)$ of a job J at some point in time to be the number of processors on which a copy of J is being run at that time.

2.1.1 Deterministic Adversary

Adversary DETERMINISTIC-DETER$(m, n, \lambda, \ell, \mathcal{A})$: Here m is the number of unit speed processors, λ is the number of faults, n is the number of jobs, ℓ is the length of the jobs, and $\mathcal{A}$ is a deterministic online algorithm. We assume that ℓ is much larger than the length of a clock cycle. At time zero, the adversary presents n jobs of length ℓ to $\mathcal{A}$. In the rest of the description, time is divided into stages, where the duration of each stage is ℓ. Let $U(i)$, $1 \leq i \leq k$, be the collection of jobs that are not completed by $\mathcal{A}$ before the end of the ith stage. So $U(1)$ is the set of all n jobs. We denote the cardinality of $U(i)$ by $u(i)$. We use $f(i)$ to denote the number of faults during the ith stage; We set the value of each $f(i)$ later. Intuitively, the goal of the adversary is to fault $f(i)$ working processors running jobs with lowest multiplicity just before the end of every stage i (except possibly the last stage), so that the number of unfinished jobs at the end of the stage is maximized for $\mathcal{A}$. Observe that in order to force $\mathcal{A}$ not to finish a job J, the adversary must fault all of the machines running J.

We now show more formally how the adversary faults processors in each stage. Consider the multiplicity of jobs in $U(i)$ just before the end of the ith stage. Assume that the jobs in $U(i)$ are numbered and ordered such that $\phi(J_1) \leq \phi(J_2) \leq \ldots \leq \phi(J_{u(i)})$. Find the largest number s such that $\sum_{j=1}^{s} \phi(J_j) \leq f(i)$. Let $U(i+1) = \{J_x : 1 \leq x \leq s\}$. Note that $s = u(i+1)$. Just before the end of the ith stage, the adversary faults every processor running a job in $U(i+1)$. Notice that since all the copies of jobs in $U(i+1)$ are terminated just before the end of the ith stage, by induction, no job in $U(i+1)$ can be completed before the end of the $(i+1)$st stage. Finally, let

$$D(i) = \begin{cases} U(i) - U(i+1) & \text{if } \sum_{j=1}^{s} \phi(J_j) = f(i) \\ U(i) - U(i+1) - \{J_{s+1}\} & \text{otherwise} \end{cases}$$

We now explain how to set the value of the $f(i)$'s.

Case 1: If $m - \lambda \geq \frac{m \log\log m}{\log m}$ then each $f(i) = \lfloor \lambda / \log m \rfloor$ for $i \leq \log m$, and $f(i) = 0$ for $i > \log m$.

Case 2: If $m - \lambda < \frac{m \log\log m}{\log m}$ then $f(1) = f(2) = m/4$. In the permanent fault case, $f(3) = \lambda - m/2$, and $f(i) = 0$ for $i \geq 4$. In the transient fault case, $f(i) = m$ for $3 \leq i \leq 2 + \lfloor (\lambda - m)/m \rfloor$, and $f(i) = 0$ for $i > 2 + \lfloor (\lambda - m)/m \rfloor$.

2.1.2 Analysis

We first prove two simple lemmas that we will use often.

Lemma 1 Let $\alpha_1, \ldots, \alpha_k$ be a collection of nonnegative reals such that $\sum_{i=1}^{i=k} \alpha_i = \alpha$. Then there are at least $k/2$ integers j such that $1 \leq j \leq k$, and $\alpha_j \leq 2\alpha/k$.

Proof: To reach a contradiction, assume that for at most $k/2$ of the j's we have $\alpha_j \leq 2\alpha/k$. Hence, there are at least $k/2$ j's such that $\alpha_j > 2\alpha/k$. Therefore, the sum of those α_j's must be strictly larger than $(\frac{k}{2})(\frac{2\alpha}{k}) = \alpha$, a contradiction. ∎

Lemma 2 Let $g(i)$ be the number of working processors at the start of the ith stage. The adversary DETERMINISTIC-DETER guarantees that if $u(i) \geq \frac{g(i)}{c} + \frac{f(i)}{c-1}$, for some integer $c \geq 1$, then $u(i+1) \geq \lfloor \frac{f(i)}{c-1} \rfloor$.

Proof: Consider some time just before the end of the ith stage. At this time there can be at most $g(i)/c$ jobs with multiplicity c or greater. Hence, since $u(i) \geq \frac{g(i)}{c} + \frac{f(i)}{c-1}$, there must be at least $\lfloor \frac{f(i)}{c-1} \rfloor$ jobs will multiplicity $c-1$ or less. The result then follows by the definition of DETERMINISTIC-DETER. ∎

Lemma 3 Assume $\lambda \leq m - \frac{m \log\log m}{\log m}$, and $m \geq n \geq \lambda \geq 1$. Then for any online deterministic algorithm $\mathcal{A}$, the number of stages forced by the adversary DETERMINISTIC-DETER$(m, n, \lambda, 1, \mathcal{A})$ is $\Omega(\frac{\log m}{\log(\frac{m \log m}{\lambda})})$.

Proof: Notice that the faults occur just before the end of every stage. We will assume that all m processors are available to $\mathcal{A}$ at the beginning of every stage even if the faults are permanent. Therefore, the lower bound we will establish under this strong assumption holds for both permanent and transient fault cases. Notice that the required bound is constant if $\lambda \leq \sqrt{m}$. So it suffices to consider the case where $n \geq \lambda \geq \sqrt{m}$.

Consider the time just before the end of the ith stage. By the definition of $D(i)$, $m - \frac{\lambda}{\log m} \geq \sum_{J \in D(i)} \phi(J)$ and $|D(i)| \geq [u(i) - (1 + u(i+1))]$. The minimum multiplicity of a job in $D(i)$ is at least the maximum multiplicity of a job in $U(i+1)$, which is in turn at least $\frac{\lambda}{\log m} \cdot \frac{1}{1+u(i+1)}$. Therefore,

$$m - \frac{\lambda}{\log m} \geq \sum_{J \in D(i)} \phi(J) \geq [u(i) - (1 + u(i+1))] \frac{\lambda}{\log m} \frac{1}{1 + u(i+1)}$$

Simplifying, we get, $1 + u(i+1) \geq u(i)(\frac{\lambda}{m \log m})$. By observing that $2u(i+1) \geq 1 + u(i+1)$ for $u(i+1) \geq 1$, we get $u(i+1) \geq u(i)(\frac{\lambda}{2m \log m})$. Since $u(1) = n$, by induction we get $u(i+1) \geq n(\frac{\lambda}{2m \log m})^i$.

At the end of the last stage, say k, we have $u(k+1) < 1$ and $u(k) \geq 1$. So

$$2 \geq 1 + u(k+1) \geq u(k)(\frac{\lambda}{m \log m}) \geq 2n(\frac{\lambda}{2m \log m})^k$$

Equivalently, we get $k \log(\frac{2m \log m}{\lambda}) \geq \log n$. Therefore, $k = \Omega(\frac{\log n}{\log(\frac{m \log m}{\lambda})})$. The result follows since $\log n \geq \frac{\log m}{2}$. ■

Lemma 4 Assume that λ satisfies $\lambda \geq m - \frac{m \log \log m}{\log m}$. Then for all deterministic online algorithms $\mathcal{A}$, the number of stages forced by the adversary DETERMINISTIC-DETER$(m, m, \lambda, 1, \mathcal{A})$ is $\Omega(\frac{m}{\eta})$ for the permanent fault case and $\Omega(\frac{\lambda}{m})$ for the transient fault case.

Proof: Note $f(1) = m/4$ and there are m working processors at the start of the first stage. Applying lemma 2 with $c = 2$, $\mathcal{A}$ has at least $m/4$ jobs not completed by the start of the second stage. Note $f(2) = m/4$ and there are at most m working processors at the start of the second stage. Applying lemma 2 with $c = 6$, $\mathcal{A}$ has at least $m/20$ jobs not completed by the start of the third stage. Now consider the permanent fault case. Since there are exactly η working processors at the end of third stage, and at least $m/20$ unfinished jobs before the end of the third stage, the number of stages is $\Omega(m/\eta)$. In the transient fault case, all processors are faulting for the next $\Omega(\lambda/m)$ stages. ■

Theorem 5 In the case of identical processors, and static release times, the competitive ratio, with respect to makespan, of any deterministic clairvoyant algorithm $\mathcal{A}$ is $\Omega(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$, for $\lambda \geq 1$ permanent faults.

Proof: Apply DETERMINISTIC-DETER$(m, m, \lambda, 1, \mathcal{A})$. Observe that the adversary faults a total of at most $m/2$ processors during the first two stages. As a consequence, at least $m/2$

processors do not experience a fault during the first two stages. These $m/2$ processors can complete m jobs at the end of the second stage. Therefore, OPT is at most 2. The result follows from lemma 3 and lemma 4. ■

Theorem 6 In the case of identical processors, and static release times, the competitive ratio, with respect to makespan, of any deterministic clairvoyant algorithm $\mathcal{A}$ is $\Omega(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{\lambda}{m}))$, for $\lambda \geq 1$ transient faults.

Theorem 7 In the case of related processors, the optimal deterministic competitive ratio, with respect to makespan, is then $\Omega(R \cdot \max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$ for $\lambda \geq 1$ permanent faults.

Proof: Let $\mathcal{A}$ be the given online algorithm. First we assume that $\lambda \leq \sqrt{m}$, where the lower bound we wish to establish is $\Omega(R)$. There are 3 processors with speed R and $m-3$ processors with speed 1. Initially, two jobs of length R arrive. Just before the end of the first (unit time) stage, one speed R processor faults so that there is an unfinished job at the end of the stage. At the end of the first stage, the remaining two speed R processors fault. It is not hard to see that $OPT = 1$ and the online makespan is $\Omega(R)$.

We now assume that $\lambda > \sqrt{m}$. There are $\lambda/2$ processors with speed R and $m - \lambda/2$ processors with speed 1. There are exactly $\lambda/4$ jobs of length R. Using lemma 1, one can see that the adversary can fault $\lambda/4$ speed R processors just before the end of the first stage such that at least $\lambda/16$ jobs are unfinished by the given online algorithm $\mathcal{A}$. The remaining speed R processors fault right after time 1. Since $\lambda/4$ speed R processors work for one full stage, $OPT = 1$. Now apply DETERMINISTIC-DETER$(m - \lambda/2, \lambda/16, \lambda/16, R, \mathcal{A})$. ■

Note that in these lower bounds, the online algorithm algorithm is faced with a plethora of faults after the optimal schedule has finished, a phenomenon that may not be quite acceptable. It is not clear how to formulate an alternative reasonable model that avoids this phenomenon.

2.1.3 Randomized Case

Using Yao's technique [3, 12], it suffices to prove the lower bound on the expected competitive ratio of any deterministic algorithm when faults occur according to a fixed probabilistic distribution. We first describe the instance of the input and the probability distribution on the faults. It is important to notice that the description of the following adversary is independent of the online algorithm.

Adversary OBLIVIOUS-DETER(m, n, λ, ℓ) : Here m is the number of unit speed processors, λ is the number of faults, n is the number of jobs, and ℓ is the length of the jobs. At time zero, the adversary releases n jobs of length ℓ. In the rest of the description, time is divided into stages, where the duration of each stage is ℓ.

Case 1: Suppose $m - \lambda \geq \frac{m}{\log^* m}$. Just before the end of every stage (except possibly the last stage), each working processor will fault independently with probability $\frac{\lambda}{2m \log^* m}$.

Case 2: Suppose $m - \lambda < \frac{m}{\log^* m}$. Right before the end of the first two stages, each working processor faults independently with probability $1/5$. In the permanent fault case, just before the end of the third stage, all but an arbitrary $m - \lambda$ processors fault. In the transient fault case, all processors fault just before the end of stages 3 through $2 + \lfloor (\lambda - m)/m \rfloor$.

2.1.4 Analysis

Before analyzing this adversary we need the following definitions and facts.

Definition 8 For a nonnegative integer k, a real $\alpha \geq 2$ and a real $z \geq 1$, we inductively define

$$tower(\alpha, k, z) = \begin{cases} z & \text{if } k = 0 \\ \alpha^{tower(\alpha, k-1, z)} & \text{otherwise} \end{cases}$$

Lemma 9 $(k-1) + \log^* \alpha^z \leq \log^*(tower(\alpha, k, z)) \leq 2(k-1) + \log^* \alpha^z + O(1)$.

Proof: Notice that $2^{2^x} \geq \alpha^x$ if $x \geq \alpha \geq 2$. Therefore, for $k \geq 2$, we have

$$tower(2, (k-1), \alpha^z) \leq tower(\alpha, k, z) = tower(\alpha, k-1, \alpha^z) \leq tower(2, 2(k-1), \alpha^z).$$

The result then follows since $\log^*(tower(2, k, z)) = k + \log^* z + O(1)$. ■

Fact 10 If $p(x)$ is some fixed polynomially bounded function, then $\log^* p(x) = \log^* x + O(1)$,

Fact 11 Let y be a random variable with expectation μ and variance σ^2. Then Chebyshev's inequality states that $Prob[|y - \mu| \geq \alpha\mu] \leq \frac{\sigma^2}{\alpha^2 \mu^2}$. If $\sigma^2 \leq \mu$ then $Prob[|y - \mu| \geq \alpha\mu] \leq \frac{1}{\alpha^2 \mu}$.

Lemma 12 Assume $\lambda \leq m - \frac{m}{\log^* m}$, and $n = \Theta(m)$. Then for any deterministic online algorithm $\mathcal{A}$, the expected number of stages forced by the adversary OBLIVIOUS-DETER$(m, n, \lambda, 1)$ on $\mathcal{A}$ is $\Omega(\log^* m - \log^* \frac{m}{\lambda})$.

Proof: In this proof we will assume that all m processors are available to $\mathcal{A}$ at the beginning of every stage. Therefore, the desired bound holds for both permanent and transient fault cases. Notice that the required lower bound is $\Omega(1)$ if $\lambda \leq m/\log m$. So it suffices to consider the case where $\lambda > m/\log m$.

Consider a time just before the end of the ith stage. At this time let $u(i)$ be a random variable representing the number of jobs that are not be completed by $\mathcal{A}$ before the end of the ith stage. We say that an unfinished job J is *weak* if $\phi(J) \leq \frac{2m}{u(i)}$ at this time. Using lemma 1, one can see that there are at least $u(i)/2$ *weak* jobs among these $u(i)$ jobs. We now show that with very high probability $u(i+1) \geq (\frac{u(i)}{4})(\frac{\lambda}{m \log^* m})^{\frac{2m}{u(i)}}$. Since processors fault independently with probability $\frac{\lambda}{2m \log^* m}$, the probability that no copy of a *weak* job J is completed at the end of the ith stage is at least $(\frac{\lambda}{2m \log^* m})^{\phi(J)} \geq (\frac{\lambda}{2m \log^* m})^{\frac{2m}{u(i)}}$. Hence by linearity of expectation,

$$E[u(i+1)] \geq (\frac{u(i)}{2}) \cdot (\frac{\lambda}{2m \log^* m})^{\frac{2m}{u(i)}}.$$

Since $u(i+1)$ is the sum of independent 0/1 valued random variables, the variance of $u(i+1)$ is at most the expected value of $u(i+1)$ (see [1]). So applying Chebyshev's inequality,

$$Pr[|u(i+1) - E[u(i+1)]| \geq E[u(i+1)]/2] \leq \frac{4}{E[u(i+1)]}.$$

We define a stage i to be *good* if $|u(i+1) - E[u(i+1)]| \leq E[u(i+1)]/2$, and we define a stage i to be *correct* if the number of faults during that stage does not exceed $\lambda / \log^* m$. Assuming that all the stages are good and correct, we first show that there are $\Omega(log^* m - \log^* \frac{m}{\lambda})$ many stages. We then show that with probability $1 - o(1)$ the first $\Theta(log^* m - \log^* \frac{m}{\lambda})$ stages are good and correct.

If all stages are good as well as correct then

$$u(i+1) \geq \frac{E[u(i+1)]}{2} \geq (\frac{u(i)}{4}) \cdot (\frac{\lambda}{2m \log^* m})^{\frac{2m}{u(i)}}$$

Let $r(i) = 2m/u(i)$, and so $r(1) = 2m/u(1) = 2m/n \geq 2$. Furthermore,

$$r(i+1) \leq 4r(i) \cdot (\frac{2m \log^* m}{\lambda})^{r(i)}$$

Since $r(i) \geq 2$, $\lambda \leq m$ and we have

$$r(i+1) \leq 2^{2+r(i)} (\frac{2m \log^* m}{\lambda})^{r(i)} \leq (\frac{2m \log^* m}{\lambda})^{3r(i)}.$$

Let $\alpha = (\frac{2m \log^* m}{\lambda})^3 \geq 8$. Therefore, we get $r(i+1) \leq \alpha^{r(i)}$, and so by induction we also get $r(i+1) \leq tower(\alpha, i, r(1))$.

Let k be an integer such that $E[u(k)] > \sqrt{m}$ and $E[u(k+1)] \leq \sqrt{m}$. Our goal is to show that $k = \Omega(log^* m - \log^* \frac{m}{\lambda})$. Since $|u(k+1) - E[u(k+1)]| \leq E[u(k+1)]/2$, we get $u(k+1) \leq 3\sqrt{m}/2$. As a consequence, $r(k+1) \geq 4\sqrt{m}/3$. Therefore, it must be the case that $tower(\alpha, k, r(1)) \geq 4\sqrt{m}/3$. Applying lemma 9 we get that

$$\log^*(tower(\alpha, k, r(1))) \leq 2(k-1) + \log^* \alpha^{r(1)} + O(1) \quad \text{and}$$

$$\log^*(tower(\alpha, k, r(1))) \geq \log^*(4\sqrt{m}/3) \geq \log^* m$$

Then by fact 10

$$k \geq \frac{1}{2}(\log^* m - \log^* \alpha^{r(1)}) + O(1) = \frac{1}{2}(\log^* m - \log^* \frac{m \log^* m}{\lambda}) + O(1)$$

We now consider two cases. If $\lambda \leq m/\log^* m$ then $m/\lambda \geq \log^* m$. Thus

$$\log^* \frac{m \log^* m}{\lambda} \leq \log^* (\frac{m}{\lambda})^2 = \log^* \frac{m}{\lambda} + O(1)$$

If $\lambda > m/\log^* m$ then $\log^* m - \log^* \frac{m \log^* m}{\lambda} = \Theta(\log^* m)$ and $\log^* m - \log^* \frac{m}{\lambda} = \Theta(\log^* m)$.

Let us now fix an x such that $x \leq k$ and $x = \Theta(\log^* m - \log^* \frac{m}{\lambda})$. We now complete our proof by showing that the probability that the first x stages are good and correct is $1 - o(1)$. The probability that there exists a stage between 1 and x that is not good is at most

$$\sum_{i=1}^{x} \frac{4}{E[u(i+1)]} \leq 4x/\sqrt{m}.$$

Hence, the probability that all the first x stages are good is at least $1 - o(1)$.

Since processors fault independently with probability $\lambda/2m \log^* m$, the expected number of faults per stage is at most $\lambda/2 \log^* m$, and the variance does not exceed the expected number of faults. Applying Chebyshev's inequality, we get that the number of faults per stage is at most $\lambda/\log^* m$ with probability at least $1 - 2(\log^* m)/\lambda$. The probability that there is a stage between 1 and x that is not correct is thus at most $2x(\log^* m)/\lambda$, which is $o(1)$ since $\lambda > m/\log m$. ∎

We will use the following lemma repeatedly.

Lemma 13 Suppose that there are $n = \epsilon m$ unfinished unit length jobs, and m idle working processors at some time t. Let $\mathcal{A}$ be an assignment of these jobs to these processors. Further suppose that each processor faults independently with probability p just before time $t+1$, where $p < 1$ is a constant. Then with probability $1 - O(1/m)$, the number of processors faulting just before time $t+1$ is at most $(1 + \frac{1}{4})pm$. Also, with probability $1 - O(1/m)$, the number of jobs unfinished before time $t+2$, using $\mathcal{A}$, is at least $\frac{\epsilon}{4} m p^{2/\epsilon} = \Theta(m)$.

Proof: Since p is a constant, $q = 1 - p$ is also a constant. Let x be the random variable representing the number of successes in m Bernoulli trials, where the success probability is p. The expected number of success $E(x)$ is mp and the standard deviation $\sigma(x) = \sqrt{mpq} \leq \sqrt{E(x)}$. Applying this to our case, the expected number of faults $E(x)$ is mp. Therefore, applying Chebyshev's inequality, the number of faults does not exceed $(1 + \frac{1}{4})pm$ with probability $1 - 16/(p^2 E(x)) = 1 - O(1/m)$.

Let K be the set of jobs with multiplicity at most $2/\epsilon$ just before time $t+1$ in $\mathcal{A}$. Applying Lemma 1, we know that $|K| \geq n/2 = \epsilon m/2$. Since processors fault independently with probability p just before time $t+1$, the probability that a job $J \in K$ is unfinished at time $t+1$ is $p^{2/\epsilon}$. Moreover, the probability that a job $J \in K$ is unfinished is independent of the probability that some other job is finished. Therefore, the expected number of jobs in K that are not finished at time $t+1$ is at least $F = \frac{\epsilon}{2} m p^{2/\epsilon} = \Theta(m)$. Since the variance is smaller than the expected value, applying Chebyshev's inequality (see fact 11), we get that with probability $1 - 4/F = 1 - O(1/m)$, the number of unfinished jobs just before time $t+2$ is at least $F/2 \geq \frac{\epsilon}{4} m p^{2/\epsilon} = \Theta(m)$. ■

Lemma 14 Assume $\lambda \geq m - \frac{m}{\log^* m}$ and $n = \Theta(m)$. Then for every deterministic online algorithm $\mathcal{A}$, the expected number of stages forced by the adversary $\text{OBLIVIOUS-DETER}(m, n, \lambda, 1)$ on $\mathcal{A}$ is $\Omega(\frac{m}{\eta})$ for the permanent fault case and $\Omega(\frac{\lambda}{m})$ for the transient fault case.

Proof: Notice that the adversary (applying case 2) faults each working processor independently with probability 1/5 during the first two stages. So applying lemma 13 twice, we get

- With probability $1 - O(1/m)$, the number of faults does not exceed $m/4$ during each of the first two stages.
- With probability $1 - O(1/m)$, the number of jobs that cannot be finished before the end of the second stage is $\Omega(m)$.
- With probability $1 - O(1/m)$, the number of jobs that cannot be completed before the end of third stage is $\Omega(m)$.

The desired bound for the permanent fault case follows, since with probability $1 - o(1)$, there are exactly η working processors and $\Theta(m)$ unfinished jobs just before the end of the third stage. In the transient fault case, just before the end of each of the next $\lfloor (\lambda - m)/m \rfloor$ stages all m processors fault. ■

Theorem 15 In the case of identical processors, and static release times, the competitive ratio, with respect to makespan, of any randomized clairvoyant algorithm $\mathcal{A}$ is $\Omega(\max\ (\log^* m - \log^* \frac{m}{\lambda}, \frac{m}{\eta}))$, for $\lambda \geq 1$ permanent faults.

Proof: In the arguments of lemma 12 and lemma 14, we also showed that at least $m/2$ processors are working during the first two stages with probability $1 - o(1)$, and hence $OPT = 2$ for these cases. For other cases, OPT is at most the makespan of $\mathcal{A}$. Therefore, $E[\frac{\text{makespan of } \mathcal{A}}{\text{makespan of OPT}}] = \Omega(\max\ (\log^* m - \log^* \frac{m}{\lambda}, \frac{m}{\eta}))$. Applying Yao's principle (see [3]), it is the case that the competitive ratio (against oblivious adversary) of an online algorithm is not smaller than $E[\frac{\text{makespan of } \mathcal{A}}{\text{makespan of OPT}}]$. ■

Theorem 16 In the case of identical processors, and static release times, the competitive ratio, with respect to makespan, of any randomized clairvoyant algorithm $\mathcal{A}$ is $\Omega(\max\ (\log^* m - \log^* \frac{m}{\lambda}, \frac{\lambda}{m}))$, for $\lambda \geq 1$ transient faults.

Theorem 17 In the case of related processors, the optimal randomized competitive ratio, with respect to makespan, is $\Omega(R)$ for $\lambda \geq 1$ permanent faults.

Proof: Once again we use Yao's technique. There are 3 processors with speed R and $m-3$ processors with speed 1. Initially, 2 jobs of length R arrive. Just before the end of the first unit length stage, the adversary faults one of the three speed R processors uniformly at random. As a consequence, with probability 1/3 there is an unfinished job at the end of the first stage. Now, at the end of the first stage, the remaining two speed R processors fault. It is not hard to see that $OPT = 1$ and the expected makespan of any online algorithm is $\Omega(R)$. ∎

Theorem 18 In the case of related processors, the optimal randomized competitive ratio, with respect to makespan, is $\Omega(R \max(\log^* m, \frac{m}{\eta}))$ for $\lambda = \Theta(m)$ permanent faults.

Proof: There are $\lambda/2$ processors with speed R and $m - \lambda/2$ with speed 1. There are exactly $\lambda/4$ jobs of length R. Just before the end of the first stage, the adversary faults each of the speed R processors independently with probability 1/3. Applying lemma 13 we get:

- With probability $1 - O(1/\lambda)$, the number of faults does not exceed $\lambda/4$ during the first stage.
- With probability $1 - O(1/\lambda)$, the number of jobs that cannot be finished before the end of the second stage is $\Omega(\lambda)$.

Since $\lambda = \Theta(m)$, with probability $1-o(1)$ the number of unfinished jobs at the beginning of the second stage is $n = \Theta(\lambda)$ and at least $\lambda/4$ speed R processors ran without fault for one full stage, and hence $OPT = 1$. At the end of the first stage, the adversary faults the remaining speed R processors. Notice that $n \leq m$. Now the adversary applies the strategy, Oblivious-Deter$(m - \lambda, \min(n, \lambda/2), \min(n, \lambda/2), R)$. Applying lemma 12 and lemma 14, we get that with probability $1 - o(1)$, the competitive ratio is $\Omega(R \cdot \max(\log^* m, \frac{m}{\eta}))$. Otherwise, the competitive ratio is at least 1. We get the desired result since the competitive ratio (against an oblivious adversary) is at least $E[\frac{\text{makespan of } \mathcal{A}}{\text{makespan of OPT}}]$ ∎

2.2 Upper Bounds for Makespan

We define a randomized algorithm, which we call Rotary, and analyze its deterministic worst-case performance and expected performance against permanent faults. We assume identical processors throughout this section.

Definition 19 The algorithm Rotary breaks time into stages.

1. Let S_i denote the ith stage.

2. Let $u(i)$ denote the number of unfinished jobs at the beginning of S_i.

3. Let $g(i)$ denote the number of working processors at the beginning of S_i.

4. Let $c(i)$ denote the number of distinct jobs completed during S_i.

5. Let $f(i)$ denote the number of faults experienced during S_i.

6. Let OPT be the optimal makespan.

2.2.1 Online Algorithm

ALGORITHM ROTARY: The algorithm divides time into stages. The first stage starts when the first job is released. We maintain the invariant that all working processors are idle at the start of a stage. At the beginning of a stage S_i, the algorithm initializes a queue Q_i to contain the collection of jobs that have been released but not finished by the start of S_i. The ordering of the jobs in Q_i is arbitrary. The rest of the execution of ROTARY depends on the relationship between $u(i)$ and $g(i)$.

Suppose $u(i) > g(i)$. Whenever a processor P is idle, P removes a job at the front of the queue Q_i and runs it. If Q_i is empty P remains idle for the rest of this stage. Stage S_i ends at the earliest time such that every job in Q_i has run to completion or has experienced a fault.

Now, suppose $u(i) \leq g(i)$. The algorithm selects uniformly at random an ordering of the $g(i)$ working processors, and then assigns jobs to processors in this order. When considering a processor P, ROTARY assigns P the job J currently on the front of Q_i, and then moves J to the back of Q_i. No processor is assigned a second job. Stage S_i ends at the earliest time such that every processor has either finished a job or has faulted.

2.2.2 Basic Properties of ROTARY

Note that ROTARY often forces a processor to remain idle for the remainder of a stage after having completed a job. This enables us to simplify many of our definitions and proofs. The asymptotic bounds do not change if we allow some arbitrary collection of idle processors to run any unfinished job. Observe that when a new job arrives in the middle of a stage, it will not be added to the queue. It will considered only during the following stages.

Our first goal is to show that by some reasonable time ROTARY has reduced the number of unfinished jobs to m.

Fact 20 For all jobs J_i, $x_i \leq OPT$.

Lemma 21 ROTARY guarantees that the length of any stage i is at most $(\frac{m}{\eta} + 1)OPT$.

Proof: If $u(i) \le g(i)$, then applying fact 20, the length of the stage is at most OPT. Suppose $u(i) > g(i)$. Since there are always at least η fault-free processors, a simple counting argument shows that for every job J in the queue, some processor must start executing a copy of J by at least time $m \cdot OPT/\eta$ after the start of the stage. Using fact 20, our claim follows. ■

Lemma 22 ROTARY guarantees that there will be at most m jobs left unfinished $(2\frac{m}{\eta}+2)OPT$ time units after the release of the last job.

Proof: By lemma 21, a new stage S starts no later than $(\frac{m}{\eta}+1)OPT$ time units after the release of the last job. Since there are at most m faults in stage S, at most m jobs are left unfinished after stage S, which lasts at most $(\frac{m}{\eta}+1)OPT$ units of time. ■

Now we focus our attention on the case where the number of unfinished jobs is at most m. Intuitively, we will argue that for a stage in which the multiplicity of a job is not too small and not too many processors fault, the online algorithm makes good progress. On the other hand, we will also argue that there cannot be too many stages with either small multiplicity or too many faults.

Definition 23

1. Let $\phi(i)$ denote the minimum, over all unfinished jobs J at start of stage S_i, of the number processors that ran a copy of J in stage i.

2. Let stage S_β be the first stage, after the release of the last job, for which the number of jobs left unfinished at the start of this stage is at most m.

3. Let $C = \{S_i : i \ge \beta\}$.

4. Let $F = \{S_i \in C : \phi(i) \le 2\}$. F is the set of all stages in C such that the algorithm ran at most two copies of some unfinished job during that stage.

5. Let $H = \{S_i \in C - F : f(i) \le g(i)/4\}$. H is the set of all stages in $C - F$ such that the ratio of the number of working processors to the number of processors experiencing a fault is at least 4.

Lemma 24 ROTARY guarantees that the number of copies of any unfinished job that are started during S_i is either $\phi(i)$ or $\phi(i)+1$.

Since η processors never experience a fault, for any stage in F, at least $\lceil \eta/3 \rceil$ jobs will be completed. Therefore $|F| \le 3\lceil \frac{m}{\eta} \rceil$. On the other hand, we will derive a bound on $|C - F - H|$, since the number of faults in each stage in $C - F - H$ is large.

Lemma 25 ROTARY guarantees that if the length a stage $S_i \in F$ is $\alpha \cdot OPT$, then $\lfloor \alpha - 1 \rfloor \le 3c(i)/\eta$.

Proof: Assume that stage S_i finished at time t. The last job assignment in this stage can not be made earlier than time $t - OPT$ by fact 20. Observe that any processor that does not experience a fault during a stage cannot be idle for more than OPT time units. Thus the η nonfaulty processors were all running jobs at all times before $t - OPT$, and thus by fact 20 each of these processors must complete at least $\lfloor \alpha - 1 \rfloor$ jobs. Since $S_i \in F$, $\phi(i) \leq 2$ and so by applying lemma 24 we conclude that $c(i) \geq \eta \lfloor \alpha - 1 \rfloor / 3$. ∎

Lemma 26 ROTARY guarantees that the total length of the stages in F is at most $9\lceil \frac{m}{\eta} \rceil$

Proof: First observe that η nonfaulty processors complete a job in each stage. For a stage in F, at most three processors finish the same job. Since the number of jobs in any of the stages in C is at most m, $|F| \leq 3\lceil \frac{m}{\eta} \rceil$. Let $\alpha_i OPT$ be the length of stage $S_i \in F$. Consider the sum

$$\sum_{S_i \in F} \alpha_i OPT \leq OPT \cdot \sum_{S_i \in F} (\lfloor \alpha_i - 1 \rfloor + 2)$$

Applying lemma 25, this sum is at most $OPT \cdot (2|F| + 3\sum_{S_i \in F} c(i)/\eta)$. Applying the bound on $|F|$, the result follows since $\sum_{S_i \in F} c(i) \leq m$. ∎

Lemma 27 ROTARY guarantees that the length of a stage in $C - F$ is at most OPT.

Proof: Suppose $S_i \in C - F$. By the definition of F, $\phi(i) > 3$. This can happen only when $u(i) \leq g(i)$. Under this situation, according to the description of ROTARY, no processor is assigned a second job during this stage. Applying fact 20, the result follows. ∎

Lemma 28 ROTARY guarantees that $|C - F - H|$ is at most $\log_{4/3} \frac{m}{\eta}$.

Proof: Observe that faults are permanent, and that initially there are m working processors. Also, for any stage $S_i \in C - F - H$, $f(i) \geq \frac{1}{4} g(i)$. As a consequence, after y stages in $C - F - H$, the number of remaining working processors is at most $m/(3/4)^y$. The result then follows, since η processors never fault. ∎

Lemma 29 The cumulative length of the stages not in H is $O(\frac{m}{\eta} OPT)$.

Proof: Follows from lemma 22, lemma 26, lemma 27 and lemma 28. ∎

In order to prove the desired upper bound for ROTARY, it now suffices to prove that the cumulative length of the stages in H is not large. Since $H \subseteq C - F$, by lemma 27 it suffices to show an upper bound on the number of stages in H.

2.2.3 Deterministic Bound

In this subsection, we will analyze the worst-case deterministic performance of ROTARY. We assume that the ordering of the processors in ROTARY is determined by some arbitrary deterministic procedure. We call the resulting algorithm DETERMINISTIC ROTARY.

Definition 30 For $1 \leq j \leq \lfloor 1 + \log \frac{m}{\eta} \rfloor$,

1. Let $N_j = \{S_i \in H : m/2^j < g(i) \leq m/2^{j-1}\}$.
2. Let λ_j be the total number of faults experienced during stages in N_j, i.e., $\lambda_j = \sum_{S_i \in N_j} f(i)$.
3. Let $n_j = |N_j|$.

Lemma 31 ROTARY guarantees that $\lambda_j \leq m/2^{j-1}$.

Proof: This follows since faults are permanent and $g(i) \leq m/2^{j-1}$ for any stage $S_i \in N_j$. ■

Lemma 32 DETERMINISTIC ROTARY guarantees that $|H| = O(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$.

Proof: We say that a stage $S_i \in N_j$ is *good* if $f(i) \leq 2\lambda_j/n_j$. By lemma 1 there can be at most $n_j/2$ stages in N_j that are not good. Hence, at least $\frac{1}{2}|H|$ stages in H are good. From now on, we consider only good stages in H. Let k be the total number of good stages in H.

Consider a good stage $S_i \in N_j$. The algorithm guarantees that $\phi(i) \geq \lfloor g(i)/u(i) \rfloor$. Notice that $H \cap F = \emptyset$. So $S_i \notin F$, and as a consequence $2\phi(i) \geq \phi(i) + 1 \geq g(i)/u(i)$. Since $m/2^j < g(i) \leq m/2^{j-1}$, we get $\phi(i) \geq m/(2^{j+1}u(i))$.

Since the multiplicity of a job is at least $\phi(i)$, it must be the case that $\phi(i)u(i+1) \leq f(i)$. Since S_i is good, $f(i) \leq 2\lambda_j/n_j$, and so $u(i+1) \leq (\frac{2\lambda_j}{n_j})/\phi(i)$. Now substituting $\phi(i) \geq m/(2^{j+1}u(i))$ into this equation we get

$$u(i+1) \leq u(i)(\frac{2\lambda_j}{n_j})(\frac{2^{j+1}}{m}) = u(i)8(\frac{\lambda_j}{n_j})(\frac{2^{j-1}}{m}) \tag{1}$$

Let $\ell = \lfloor 1 + \log_2 \frac{m}{\eta} \rfloor$. We now measure the progress made by good stages in H. We set $u'(i)$ to be equal to $u(x)$, where S_x is the chronologically ith good stage. Applying definition 23, the number of unfinished jobs in any stage in $H \subset C$ is at most m. Therefore, we have $u'(1) \leq m$. Since at least $n_j/2$ stages in N_j are good, applying induction we get

$$u'(k) \leq m8^k \prod_{j=1}^{\ell} (\frac{\lambda_j}{n_j} \frac{2^{j-1}}{m})^{n_j/2} \tag{2}$$

Recall that k is the number of good stages in H. First assume that $\eta > m/2$. Notice that then $\ell = 1$, $\lambda_1 \le \lambda$ and $k \le n_1 \le 2k$. Observe that $u'(k) \ge 1$, otherwise there will be at most $k-1$ stages. Then equation 2 implies

$$m8^k(\frac{\lambda}{n_1 m})^{k/2} \ge 1 \quad \text{or} \quad \log m \ge \frac{k}{2}\log(\frac{n_1 m}{2\lambda}) - 3k \ge \frac{k}{2}\log(\frac{km}{2\lambda}) - 3k$$

Therefore, $k = O(\frac{\log m}{\log(\frac{k \log k}{\lambda})})$.

We now consider the case that $m/2 \ge \eta \ge m/\log m$. Call an n_j *big* if $n_j \ge k/(2\ell)$. Observe that by lemma 31, $\lambda_j \le \frac{m}{2^{j-1}}$. Notice that the $u'(i)$'s do not increase as we move from one stage to another. As before, applying the fact that $u'(k) \ge 1$, we will show that k is not large. In order to estimate k, we consider the rate at which the $u'(i)$'s decrease during a good stage where n_i is *big*. So, applying $u'(k) \ge 1$, we can conclude that

$$m8^k \prod_{\text{big } n_j} (\frac{2\ell}{k})^{n_j/2} \ge m8^k \prod_{\text{big } n_j} (\frac{\lambda_j}{n_j}\frac{2^{j-1}}{m})^{n_j/2} \ge m8^k \prod_{j=1}^{\ell} (\frac{\lambda_j}{n_j}\frac{2^{j-1}}{m})^{n_j/2} \ge 1 \tag{3}$$

In the above equation, the first and the second inequality follows since $n_j \ge k/(2\ell)$ and for any j in $[1,\ell]$, $\frac{\lambda_j}{n_j}\frac{2^{j-1}}{m} \le 1$. Notice we can use lemma 1 to show that $\sum_{\text{big } n_j} n_j \ge k/2$. Therefore, equation 3 implies

$$m8^k \prod_{\text{big } n_j} (\frac{2\ell}{k})^{n_j/2} = m8^k (\frac{2\ell}{k})^{\sum_{\text{big } n_j} n_j/2} \le m8^k (\frac{2\ell}{k})^{k/4} \tag{4}$$

As a consequence, we get

$$m8^k(\frac{2\ell}{k})^{k/4} \ge 1 \text{ or } \frac{k}{4}\log\frac{k}{4 \cdot 8^4 \ell} \le \log m \tag{5}$$

Then by the definition of ℓ, $\ell = O(\log\log m)$. Then equation 5 implies $k = O(\log m/\log\log m)$. Note that $\log m/\log\log m = O(\frac{\log m}{\log(\frac{m \log m}{\lambda})})$ if $m/2 \ge \eta \ge m/\log m$.

Now assume $\eta < m/\log m$. Since $\ell = O(k)$, equation 5 implies $k = O(\log m)$. Note that $\log m = O(\frac{m}{\eta})$ for $\eta < m/\log m$. The result follows since $k \ge \frac{1}{2}|H|$. ■

Theorem 33 In the case of permanent faults and identical processors, the competitive ratio, with respect to makespan, of DETERMINISTIC ROTARY is $O(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$.

Proof: The result follows from lemma 29, lemma 32 and lemma 27. ■

2.2.4 Randomized Bound

We now examine the expected performance of the randomized algorithm ROTARY.

Lemma 34 *The expected number of stages in H for* ROTARY *is* $O(\log^* m - \log^* \frac{m}{\lambda})$.

Proof: In the following proof, we will not use the fact that $\lambda \leq m$. As a consequence, this lemma is also true for transient faults case where λ could be significantly larger than m. We will appeal to this fact when we discuss transient faults.

Let S_i be a stage in H. For $1 \leq j \leq \phi(i)$, let $P_j(i)$ be the set of processors that were assigned a job from the jth time through the queue in S_i, e.g. the first $u(i)$ processors assigned jobs are in $P_1(i)$. Let $f_j(i)$ be a random variable representing the number of faults experienced by processors in $P_j(i)$ during stage S_i. Recall that $f(i)$ is the total number of faults during stage S_i.

For $1 \leq a \leq u(i)$, let x_a be a random variable equal to 1 if all copies of the ath job starting S_i experience a fault during stage S_i, and 0 otherwise. Observe that processors are randomly ordered before they start selecting a job from the queue. As a consequence, the probability that a job J_a is not completed by a processor that selected J_a during the jth time through the queue is $\frac{f_j(i)}{u(i)}$. Then

$$E[x_a] = Pr[J_a \text{ not finished during } S_i] \leq \prod_{i=1}^{\phi(i)} \frac{f_j(i)}{u(i)} \tag{6}$$

For a fixed number δ of faults during the stage S_i and $a \neq b$, notice that

$$Pr[x_a = 1 \mid x_b = 1, f(i) = \delta] \leq Pr[x_a = 1 \mid f(i) = \delta]$$

Therefore, we can conclude that

$$Pr[x_a = 1, x_b = 1 \mid f(i) = \delta] \leq Pr[x_a = 1 \mid f(i) = \delta] \cdot Pr[x_b = 1 \mid f(i) = \delta]$$

Since the x_i's are 0/1-random variables, $\mathrm{Cov}[x_a, x_b] = Pr[x_a = 1, x_b = 1] - Pr[x_a = 1] \cdot Pr[x_b = 1]$. As a consequence, for $a \neq b$, the covariance $\mathrm{Cov}[x_a, x_b] \leq 0$. For 0/1 random variables x_a's (see [1]),

$$\mathrm{Var}[\textstyle\sum_{j=1}^{u(i)} x_j] \leq E[\sum_{j=1}^{u(i)} x_j] + \sum_{a \neq b} \mathrm{Cov}[x_a, x_b] \tag{7}$$

As a consequence, we get

$$\mathrm{Var}[\textstyle\sum_{j=1}^{j=u(i)} x_j] \leq E[\sum_{j=1}^{j=u(i)} x_j] \tag{8}$$

The product in equation 6 is maximum subject to $\sum_{j=1}^{\phi(i)} f_j(i) \leq f(i)$ if each $f_j(i) = f(i)/\phi(i)$. Hence for $1 \leq a \leq u(i)$,

$$E[x_a] \leq \prod_{i=1}^{\phi(i)} \frac{f(i)}{u(i)\phi(i)} \tag{9}$$

Note $u(i)\phi(i) \geq u(i)(\frac{g(i)}{u(i)} - 1)$. Since $S_i \notin F$, it is the case that $u(i) \leq g(i)/2$, and $u(i)(\frac{g(i)}{u(i)} - 1) \geq g(i)/2$. Hence,

$$E[x_a] \leq \prod_{i=1}^{\phi(i)} \frac{2f(i)}{g(i)} \tag{10}$$

Let k be the number of stages in H, and let α be the number of faults experienced during stages in H. Since $f(i) \leq g(i)/4$ for any S_i in H, we have $2f(i)/g(i) \leq 1/2$. Call a stage $S_i \in H$ *good* if $f(i) \leq \frac{2\alpha}{k}$. Using lemma 1 we can show that at least $\frac{1}{2}|H|$ stages in H are good. So, it suffices to find an upper bound on the expected number of good stages in H. Let S_i be a good stage under consideration. Hence,

$$E[x_a] \leq (\min(\frac{1}{2}, \frac{4\alpha}{g(i)k}))^{\phi(i)} \leq (\min(\frac{1}{2}, \frac{4\lambda}{g(i)k}))^{\phi(i)} \tag{11}$$

and applying linearity of expectation we get

$$E[\sum_{j=1}^{u(i)} x_j] = E[u(i+1)] = \sum_{j=1}^{u(i)} E[x_j] \leq u(i)(\min(\frac{1}{2}, \frac{8\lambda}{g(i)k}))^{\phi(i)} \tag{12}$$

Now, applying the equation 8 we get

$$\mathrm{Var}[u(i+1)] \leq u(i)(\min(\tfrac{1}{2}, \tfrac{8\lambda}{g(i)k}))^{\phi(i)}. \tag{13}$$

Call a stage S_i in H *unlucky* if $u(i+1) > 3E[u(i+1)]$. Since

$$3u(i)(\min(\frac{1}{2}, \frac{8\lambda}{g(i)k}))^{\phi(i)} \geq 3E[u(i+1)]$$

we have

$$u(i+1) \leq 3u(i)(\min(\frac{1}{2}, \frac{8\lambda}{g(i)k}))^{\phi(i)}$$

for any *lucky* stage. By Chebyshev's inequality, the probability that a stage is *unlucky* is at most $\mathrm{Var}[u(i+1)]/(2E[u(i+1)])^2 \leq 1/4$. Since at least $k/2$ stages in H are good, with probability $1 - O(1/k)$, at least $k/4$ good stages in H are *lucky*.

Let $k' = \Theta(k)$ be the number of *lucky* stages in H. We ignore the progress made by ROTARY during stages in H that are *unlucky*. Suppose stage x is *lucky*. Then

$$u(x+1) \le 3u(x)(\min(\frac{1}{2}, \frac{8\lambda}{g(x)k}))^{\phi(x)} \le 3u(x)(\min(\frac{1}{2}, \frac{8\lambda}{g(x)k}))^{\frac{g(x)}{u(x)}} \tag{14}$$

Since $\phi(x) \le g(x)/u(x)$, by setting $r(x) = \frac{g(x)}{u(x)}$, and observing that $g(x) \ge g(x+1) = g(x) - f(x) \ge \frac{3}{4}g(x)$ for any stage $S_x \in H$, we get

$$r(x+1) \ge \frac{r(x)}{4}(\max(2, \frac{g(x)k}{8\lambda}))^{r(x)} \tag{15}$$

Since $r(1) \ge 3$, we have $r(2) \ge 4$. Therefore, for $x \ge 2$

$$r(x+1) \ge (\max(2, \frac{g(x)k}{8\lambda}))^{r(x)} \tag{16}$$

Observe that for all i, $g(i) \ge g(i+1)$ (equality holds in the case of transient faults). If k' is the number of *lucky* stages then by induction we get

$$r(k) \ge tower(\max(2, \frac{g(k)k}{32\lambda}), k'-2, r(2))$$

Notice that $r(2) \ge 1$, $r(k) \le m$ and $k \ge k'$. Notice that the required bound follows if $k \le 16$. Now consider the case where $k > 16$. Therefore, we get

$$m \ge tower(\max(2, \frac{g(k)k}{8\lambda}), k'-1, 1) \ge tower(2, k'-2, \max(2, \frac{g(k)}{\lambda}))$$

Taking $\log^*$ both sides and applying lemma 9 we get

$$log^* m \ge k' + \log^*(\max(2, \frac{g(k)}{\lambda})) + O(1)$$

Suppose $\lambda \le m/3$, then $g(k) \ge 2m/3$, and so $\frac{g(k)}{\lambda} \ge 2$. On the other hand, if $\lambda > m/3$, we have that $\max(2, \frac{g(k)}{\lambda})$ is in the range $[2, 3]$. Hence, the expected number of stages in H is $O(\log^* m - \log^*(\frac{m}{\lambda}))$. ■

Theorem 35 **The competitive ratio of ROTARY is** $O(\max(\log^* m - \log^* \frac{m}{\lambda}, \frac{m}{\eta}))$.

Proof: The result follows from lemma 29 and lemma 34. ■

3 Average Completion Time

We present a nonclairvoyant deterministic algorithm that is $O(\frac{m}{\eta})$-competitive with respect to average completion time. We then show that every randomized algorithm for this problem has a competitive ratio, with respect to average completion time, that is $\Omega(R\frac{m}{\eta})$. We assume static release times, m identical processors, and consider only permanent faults. We also assume that we have a lower bound of 1 on the length of any task. This is not an unreasonable restriction since we could take our unit of length to be the time to execute one instruction on the fastest processor.

3.0.5 Online Algorithm

Geometric Rotary: The algorithm consists of stages. Stage S_0 starts when all the jobs are released. Let $u(i)$ denote the number of jobs that have not yet been run to completion at the start of stage S_i. We call S_i *full* if $u(i) \geq 2m$.

We now define the stage S_i. We first form a queue Q of unfinished jobs. For a full stage, we add exactly one copy of each of the $u(i)$ jobs to the queue. In the case of a stage that is not full, we add $\lceil 2m/u(i) \rceil$ copies of each of these $u(i)$ jobs to Q.

During the stage when a processor P becomes idle, the next job J on Q is removed from Q and run from its beginning on P until one of the following three conditions holds:

a) job J is completed, or

b) P ran J for 2^i time units without completion, or

c) a fault occurs while P is processing J.

If case a) occurs, P immediately attempts to get another job from Q. If case b) occurs, P suspends J, and then immediately attempts to get another job from Q. If Q is empty, P idles until the end of the stage. Stage S_i ends when all processors are idle and Q is empty.

3.0.6 Analysis

Fact 36 Suppose that n jobs of length $\ell_1, \ell_2, \ldots, \ell_n$ arrive at time 0. Let $\ell_i \leq \ell_j$ for $1 \leq i \leq j \leq n$. The optimal offline average completion time on a single unit speed processor is $\frac{1}{n}\sum_{i=1}^{n}(n-i+1)\ell_i$.

First notice that doubling the size of jobs only doubles the average offline optimal completion time. This also holds for the multiprocessor case since the Shortest Processing Time algorithm is optimal. So, for the purpose of our analysis, let us assume that we round the execution time of each job up to the next highest integer power of 2. Then for every job, if the length of the job is $\ell = 2^i$, we assume that its length is 2ℓ (which may be up to

4 times its original length) when we calculate the optimal total completion time. We do so for the ease of analysis, and it is important to observe that the competitive ratio will be off by a factor of at most 2. This doubling idea has been used before, see for example [10].

Observe that Geometric Rotary "estimates" the length of a job by running it for a duration of at most 2^i during the ith stage. Based on this estimated length, we calculate a lower bound on the completion time for the offline optimal algorithm. As we move from one stage to another, the estimated lower bound increases due to increase in the estimated length of some jobs from 2^i to 2^{i+1}.

We use the terminology "increase in completion time experienced by Geometric Rotary during S_i" to mean the sum over all jobs J of the portion of the competition time for J that occurs during stage S_i.

At the beginning of a stage S_i, the estimated length of an unfinished job is 2^i. Assuming that this estimated length is correct, one can compute the optimal total completion time. At the end of the stage S_i, Geometric Rotary will increase the estimated length of some unfinished jobs to 2^{i+1}. As a consequence of this increase in estimated lengths of some jobs, there will an increase in our lower bound of the optimal total competition time. This is what we call the "increase in the estimated lower bound on the completion time of the offline algorithm during a stage".

We will now compare the increase in completion time experienced by Geometric Rotary during S_i to the corresponding increase in the estimated lower bound on the completion time of the offline optimal algorithm during this stage.

Lemma 37 If we have k jobs of length x to be run on p processors, then every schedule (that doesn't unnecessarily idle the processors) is optimal and has total completion time

$$\sum_{i=1}^{\lfloor k/p \rfloor} ipx + (k \bmod p)\lceil k/p \rceil x = \lfloor k/p \rfloor(\lfloor k/p \rfloor + 1)px/2 + (k \bmod p)\lceil k/p \rceil x$$

Proof: The ith term in the sum represents the sum of completion times of the ith jobs executed on each of the p processors. ■

Lemma 38 If S_i is full then Geometric Rotary guarantees that the increase in the total completion time experienced during S_i is at most $2^{i+1}u(i)^2/\eta$.

Proof: Since processors work in parallel, more processors imply less increase in completion time. At least η processor work without fault during this stage. The increase in the completion time is maximized if we have only η working processors. Thus the length of S_i is at most $2^i\lceil u(i)/\eta \rceil \leq 2^{i+1}u(i)/\eta$. If all jobs are delayed this amount, then the increase in total completion time is $2^{i+1}u(i)^2/\eta$. ■

Lemma 39 If S_i is not full then Geometric Rotary guarantees that the increase in total completion time experienced during S_i is at most $5u(i)m2^i/\eta$.

Proof: The length of S_i is at most $2^i\lceil u(i)\lceil 2m/u(i)\rceil/\eta\rceil \leq 2^i\lceil 4m/\eta\rceil \leq 5m2^i/\eta$. Since at most $u(i)$ jobs are delayed by this amount, we get the desired bound. ∎

Lemma 40 Assume we double the length of k jobs from x to $2x$.

(a) If $k \geq m$ then the increase in the offline optimal total completion time is at least $k^2x/8m$.

(b) If $k < m$ then the increase in the offline optimal total completion time is at least kx.

Proof: Consider only the time required to run the additional x units of each job. By lemma 37 this is $\lfloor k/m\rfloor(\lfloor k/m\rfloor + 1)mx/2 + (k \bmod m)\lceil k/m\rceil x$. Suppose $k \geq m$, substituting $\lfloor k/m\rfloor \geq k/2m$ we get the desired result. Suppose $k < m$, we get the desired bound since $(k \bmod m)\lceil k/m\rceil x = kx$. ∎

Lemma 41 In every stage S_i, Geometric Rotary guarantees that at least half of the $u(i)$ jobs have at least one copy run the full 2^i time units on some processor.

Proof: Recall our assumption that the length of a job is a power of 2. If S_i is full then this follows because there are at most m faults. If S_i is not full then the number of jobs that can have all copies experience faults is at most $m/\lceil 2m/u(i)\rceil \leq u(i)/2$. ∎

Theorem 42 Given static release times, Geometric Rotary is $O(\frac{m}{\eta})$-competitive, with respect to average completion time, for $\lambda = m - \eta$ permanent faults.

Proof: For stage S_i, lemma 41 shows that Geometric Rotary runs at least half the jobs for their full 2^i time slice. Also, recall our assumption that whenever Geometric Rotary runs and completes a job J of length 2^i, the length of J for the adversary is 2^{i+1}, and so J can be considered as a job whose length increased from 2^i to 2^{i+1} when we apply lemma 40. As a consequence, when we apply lemma 40, we can set $k \geq u(i)/2$.

Suppose $u(i) \geq 2m$. Lemma 38 and lemma 40 show that the ratio of the increase in the completion time experienced by Geometric Rotary and the optimal offline algorithm is at most $\frac{2^{i+1}u(i)^2/\eta}{k^2 2^i/8m}$ where $k \geq u(i)/2 \geq m$. Therefore, the ratio is at most $64m/\eta$.

Suppose $m \leq u(i) < 2m$. Lemma 39 and lemma 40 show that the ratio of the increase in the completion time experienced by Geometric Rotary and the optimal offline algorithm is at most $\frac{5u(i)m2^i/\eta}{k^2 2^i/8m} \leq \frac{160m^2}{u(i)\eta} \leq 160m/\eta$.

Finally, suppose $u(i) < m$. Lemma 39 and lemma 40 show that the ratio of the increase in the completion time experienced by Geometric Rotary and the optimal offline algorithm is at most $10m/\eta$. ∎

Theorem 43 With static release times, and related processors, every randomized algorithm is $\Omega(R\frac{m}{\eta})$-competitive with respect to average completion time.

Proof: Using Yao's technique [12], it suffices to bound the expected competitive ratio of any deterministic algorithm where the occurrence of fault is selected from some fixed probability distribution.

First we assume that $\lambda \leq m/2$. The desired lower bound is $\Omega(R)$. There are 3 processors with speed R and $m-3$ processors with speed 1. Initially, 2 jobs of length R arrive. Just before the end of the first unit length stage, the adversary faults one of the three speed R processors uniformly at random. As a consequence, with probability 1/3 there is an unfinished job at the end of the first stage. Now, at the end of the first stage, the remaining two speed R processors fault. One can see that $OPT = 1$, and that the expected completion time of any online algorithm is $\Omega(R)$.

We now assume that $\lambda > m/2$. There are λ processors with speed R and $m - \lambda$ with speed 1. There are exactly $\lambda/2$ jobs of length R. Just before the end of the first stage, the adversary faults each of the speed R processor independently with probability 1/3. Applying lemma 13 we get:

- With probability $1-O(1/\lambda)$, the number of faults does not exceed $\lambda/2$ before the end of the first stage.
- With probability $1-O(1/\lambda)$, the number of jobs that cannot be finished before the end of the second stage is $\Omega(\lambda)$.

Since $\lambda > m/2$, with probability $1-o(1)$, the number of unfinished jobs at the beginning of the second stage is $\Theta(\lambda)$ and at least $\lambda/2$ speed R processors ran without fault for one full stage (and so $OPT = 1$).

At the end of the first stage, the adversary faults the remaining speed R processors. Now there are $\eta = m - \lambda$ unit speed processors, $x = \Theta(\lambda)$ unfinished jobs of length R. Therefore, the total completion time for any deterministic online algorithm is at least

$$\sum_{i=1}^{\lfloor x/\eta \rfloor} Ri\eta = \Theta(\frac{R\lambda^2}{\eta})$$

with probability $1-o(1)$. Note that for the case under consideration, the number of jobs is $\Theta(\lambda)$ and $\lambda = \Theta(m)$. Therefore average completion time is $\Omega(R\frac{m}{\eta})$. The result follows since the competitive ratio (against oblivious adversary) is at least $E[\frac{\text{makespan of } \mathcal{A}}{\text{makespan of OPT}}]$. ■

4 Transient Faults

In this section we analyze the effect of transient faults, which by definition have duration 0. Here λ denotes the total number of faults. Note that it is possible for the number of faults λ to exceed m. We assume identical processors. The lower bounds for makespan with transient faults was handled in section 2.1, so we begin with the upper bound for makespan.

We modify ROTARY to immediately end a stage S, i.e., every processor suspends its current job, if more than $m/2$ faults occur during this stage. We call the new algorithm TRANSIENT ROTARY. Analogously, we also modify DETERMINISTIC ROTARY and call this new algorithm DETERMINISTIC TRANSIENT ROTARY. Because of the great similarity with the case of permanent faults, we will be brief in our discussion. Recall definitions 19 and 23. Note that $g(i) = m$ for all i.

Lemma 44 TRANSIENT ROTARY guarantees that the length of any stage is at most $O(OPT)$.

Proof: Analogous to the proof of lemma 21. It is important to observe that all m (instead of at least η) processors are available at the start of the stage. Also at least $m/2$ (instead of η) processors are working throughout a stage. ∎

Lemma 45 TRANSIENT ROTARY guarantees that the total number of stages S_i where $f(i) \geq \frac{m}{4}$ is at most $\frac{4\lambda}{m} = O(\frac{\lambda}{m})$.

Proof: If more than $\frac{4\lambda}{m}$ stages have $\frac{m}{4}$ or more faults each then the total number of faults will exceed λ. ∎

Lemma 46 TRANSIENT ROTARY guarantees that there will be at most $m/4$ jobs left unfinished after $O(\frac{\lambda}{m})$ stages starting from the arrival of the last job.

Proof: After the arrival of the last job, consider a stage with total number of faults less than $m/4$. Clearly, there are at most $m/4$ unfinished jobs after this stage. Applying lemma 44 and lemma 45, we get the desired result. ∎

Theorem 47 The competitive ratio, with respect to makespan, of DETERMINISTIC TRANSIENT ROTARY is $O(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{\lambda}{m}))$, for λ transient faults.

Proof: Given the previous three lemmas, it suffices to bound $|H|$. We ignore the progress made by the online algorithm on stages that are not in H. Let $H = \{S_1, S_2 \ldots S_k\}$. For $1 \leq i \leq k$, the algorithm guarantees that $\phi(i) \geq \lfloor \frac{m}{u(i)} \rfloor$, and $u(i+1) \leq f(i)/\phi(i)$. Therefore, $u(i+1) \leq 2u(i)f(i)/m$. We say that a stage in H is *good* if $f(i) \leq 2\lambda/k$. By lemma 1 at least $|H|/2$ stages are good. Suppose $8\lambda/k \geq m$, then $k \leq 8\lambda/m$ and so the result follows. From now on consider the case $2\lambda/k < m/4$. So for a good stage S_i we have $u(i+1) \leq \frac{4\lambda}{km}u(i)$. Proceeding as in the proof lemma 32, we are left solving the equation $m \geq k \log \frac{km}{4\lambda}$ for k, which yields the fact that $k = O(\frac{\log m}{\log(\frac{m \log m}{\lambda})})$. ∎

Theorem 48 The competitive ratio, with respect to makespan, of TRANSIENT ROTARY is $O(\max(\log^* m - \log^* \frac{m}{\lambda}, \frac{\lambda}{m}))$, for λ transient faults.

Proof: Given the three previous lemmas, it suffices to show that $|H| = O(\max(\log^* m - \log^* \frac{m}{\lambda}, \frac{\lambda}{m}))$. Observe the fact that all m processors are available at the start of each stage. That is, for all i, $g(i) = m$ in the proof of lemma 34. The result follows from lemma 34 . ∎

As in the case of ROTARY, we modify GEOMETRIC ROTARY to repeat a stage if it experiences more than $m/2$ faults during that stage. Call the new algorithm TRANSIENT GEOMETRIC ROTARY.

Theorem 49 For static release times, the competitive ratio, with respect to average completion time, of GEOMETRIC TRANSIENT ROTARY is $O(\max(1, \lambda/m))$, for λ transient faults.

Proof: Notice that the increase in the completion time experienced by GEOMETRIC TRANSIENT ROTARY during a stage is at most the total completion time of the optimal offline algorithm. There can be at most $2\lambda/m$ stages that experience more than $m/2$ faults. For a stage that experiences less than $m/2$ faults, we can apply the analysis found in the permanent fault case where $\eta \geq m/2$. The result then follows. ∎

5 No Redundancy

In this section we look at the effect of not allowing more than one copy of a job to be run at any one time. We consider only the case of identical processors. We first show that deterministic algorithms can have very high competitive ratios.

Theorem 50 If no redundancy is allowed, then the competitive ratio with respect to makespan and with respect to average completion time, is $\Omega(\lambda)$, for λ permanent faults or λ transient faults in a system of identical processors.

Proof: Assume that we have one job J. Allow a processor to run J and fault it just before it is ready to complete the job. This process can be repeated λ times. ∎

We say that an online algorithm A is *stage-based* if:

1. The time spent by A is partitioned into stages.
2. At the start of each stage, all of the working processors are idle.
3. Jobs considered for execution during a stage are exactly those that are released before the start of the stage.
4. It is possible to determine, at the start of the stage, the number of copies of each job that will be run during the stage.
5. The multiplicities of any two jobs during a stage differs by at most 1.

Lemma 51 Rotary, Transient Rotary, Geometric Rotary and Geometric Transient Rotary are stage-based algorithms.

For any *stage-based* deterministic algorithm $\mathcal{A}$, we define a randomized algorithm One-Copy-$\mathcal{A}$ that does not run multiple copies of a job simultaneously. At the start of a stage S, assume that $\mathcal{A}$ plans to run k copies of a job J_i during S. One-Copy-$\mathcal{A}$ selects an integer uniformly at random from the range 1 to k inclusive, and assigns it to a variable α_i. One-Copy-$\mathcal{A}$ then simulates $\mathcal{A}$ and when $\mathcal{A}$ makes the α_ith assignment of J_i, One-Copy-$\mathcal{A}$ makes its only assignment of J_i during this stage. The following lemma is the key to analyzing One-Copy-$\mathcal{A}$.

Lemma 52 Let $\mathcal{A}$ be a deterministic *stage-based* online algorithm that starts a stage S_i with $u(i)$ jobs. Assume that for any pattern of $f(i)$ faults during this stage, $\mathcal{A}$ guarantees that at most $u'(i)$ jobs are unfinished at the end of the stage. Then the expected number of unfinished jobs by One-Copy-$\mathcal{A}$ is $O(u'(i))$ and the variance of the number unfinished jobs is not greater than the expected value.

Proof: Let $\mathcal{J} = \{J_1, J_2, \ldots J_{u(i)}\}$ be the set of unfinished jobs at the start of the stage S_i. Let $\phi(J_j)$ be the number of copies of a job $J_j \in \mathcal{J}$ ran by $\mathcal{A}$ in this stage. Without loss of generality, let $0 < \phi(J_a) \leq \phi(J_b)$ for $1 \leq a \leq b \leq u$. Let $x_a = 1$ if J_a is not finished by One-Copy-$\mathcal{A}$ during the stage, and $x_a = 0$ otherwise. Let $x = \sum_{i=1}^{u} x_a$ denote the number of jobs that are not completed during the stage by One-Copy-$\mathcal{A}$. Let $f_j(i)$ be the number of processors experiencing a fault while running a copy of J_j using algorithm $\mathcal{A}$. First

$$E[x] = \sum_{j=1}^{u(i)} E[x_j] = \sum_{j=1}^{u(i)} \frac{f_j(i)}{\phi(J_j)} \leq \sum_{j=1}^{u(i)} \frac{f_j(i)}{\phi(J_1)}$$

Note that since in the worst case the adversary for $\mathcal{A}$ will fault all the low multiplicity jobs,

$$\sum_{j=1}^{u'(i)+1} \phi(J_j) > f(i) = \sum_{j=1}^{u(i)} f_j(i)$$

Since the multiplicity of each job is at most $\phi(J_1) + 1$, we get that

$$E[x] \leq \frac{1}{\phi(J_1)} \sum_{j=1}^{u(i)} f_j(i) \leq \frac{1}{\phi(J_1)} \sum_{j=1}^{u'(i)+1} \phi(J_j) \leq \frac{1 + \phi(J_1)}{\phi(J_1)} (u'(i) + 1) \leq 4u'(i).$$

For fixed $f_j(i)$'s, the x_j's are independent. As a consequence, the variance of x (the sum of 0/1-random variables) does not exceed the expected value of x. ■

Theorem 53 The randomized competitive ratio, with respect to makespan, of One-Copy-Deterministic Rotary is $O(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$ for permanent faults, and $O(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{\lambda}{m}))$ for transient faults in a system of identical processors.

Proof: The analysis is analogous to the analysis of ROTARY. The major difference is in applying the proof of lemma 34 which establishes a bound on H.

Of course, the bound we wish to establish is different from the one proved in lemma 34. The recurrence relation established in the equation 14 will be different in the current proof. We continue to follow the notation established in the proof of lemma 52 and in the proof of lemma 34. Notice that $E(x) \leq \frac{1}{\phi(J_1)} f(i)$. Let $g(i)$ be the number of processors that did not experience a fault during the ith stage (i.e., S_i). Notice that $g(i)/u(i) \geq \phi(J_1) \geq \lfloor g(i)/u(i) \rfloor \geq 3$. As a consequence $E(x) \leq \epsilon u(i) \frac{f(i)}{g(i)}$ where $\epsilon < 3/2$.

Recall that a stage $S_i \in H$ is defined to be *good* (in the proof of lemma 34) if $f(i) \leq 2\alpha/k \leq 2\lambda/k$. We define a stage to be *unlucky* if $u(i+1) > 2E[u(i+1)]$. Therefore, at the end of a *lucky* stage $u(i+1) \leq 2\epsilon u(i) \frac{f(i)}{g(i)}$. Therefore, $u(i+1) \leq 2\epsilon u(i)\ (\min(\frac{1}{4}, \frac{4\lambda}{kg(i)}))$.

As before, at least $k/2$ stages in H are *lucky*. Consider the rate at which the $u(i)$'s decrease on *lucky* stages. Applying the fact that $u(1) \leq m$ and $u(k) \geq 1$, we get $1 \leq u(k) \leq u(1)[2\epsilon \min(\frac{1}{4}, \frac{4\lambda}{kg(k)})]^{k/2}$. Solving for k we get $k = \frac{\log m}{\log(\frac{m \log m}{\lambda})}$. ∎

Theorem 54 **If no redundancy is allowed then the optimal randomized competitive ratio, with respect to makespan, is $\Omega(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{m}{\eta}))$ for permanent faults, and $\Omega(\max(\frac{\log m}{\log(\frac{m \log m}{\lambda})}, \frac{\lambda}{m}))$ for transient faults in a system of identical processors.**

Proof: We convert DETERMINISTIC-DETER$(m, m, \lambda, 1, \mathcal{A})$ into a randomized adversary by selecting the faults in each stage uniformly at random among all working processors.

Suppose $\lambda \leq m - \frac{m \log\log m}{\log m}$. Therefore, case 1 of the adversary applies. Let $u(i)$ be a random variable that represents the number of unfinished jobs at the start of the ith stage. Since $\lambda/\log m$ processors are chosen uniformly at random, the probability that a given processor experiences a fault during this stage is at least $\frac{\lambda}{m \log m}$. Therefore $E[u(i+1)] \geq u(i)\frac{\lambda}{m \log m}$ and $\text{VAR}[u(i+1)] \leq E[u(i+1)]$.

Let k be an integer such that $E[u(k)] > \sqrt{m}$ and $E[u(k+1)] \leq \sqrt{m}$. We now argue that with probability $1 - o(1)$, $k = \Omega(\frac{\log m}{\log(\frac{m \log m}{\lambda})})$. As in the proof of lemma 14, we define a stage i to be *good* if $|u(i+1) - E[u(i+1)]| \leq E[u(i+1)]/2$. Applying Chebyshev's inequality, with probability $\sum_{i=1}^{x} \frac{4}{E[(u(i+1)]} \leq \frac{4x}{\sqrt{m}}$, we have $|u(i+1) - E[u(i+1)]| \leq \frac{1}{2}E[u(i+1)]$ for all i in the range 1 through x $(\leq k)$. Substituting $u(1) = m$, we get $u(i+1) \geq \frac{u(i)}{2}\frac{\lambda}{m \log m} \geq m(\frac{\lambda}{2m \log m})^i$. Since $u(k+1) \leq \sqrt{m}/2$, we get $k = \Omega(\frac{\log m}{\log(\frac{m \log m}{\lambda})})$. The result follows since OPT is $O(1)$.

Now, suppose $\lambda > m - \frac{m \log\log m}{\log m}$. Notice that during the first two stages, the probability that a working processor faults is $\Theta(1)$. Therefore with probability $1 - o(1)$, $\Theta(m)$ jobs will not be completed at the end of the second stage. The desired bound for the permanent fault case follows, since there are exactly η working processors and $\Theta(m)$ unfinished jobs. On the other hand, in the case of transient faults, no progress is made for the next $\lfloor(\lambda - m)/m\rfloor$ stages since all m processors fault just before the end of each of those stages. The result then follows. Note that the analysis is analogous to that in the proof of lemma 4. ∎

Theorem 55 If no redundancy is allowed then the optimal randomized competitive ratio, with respect to average completion time, is $\Theta(\max(\frac{m}{\eta}, 1))$ for permanent faults.

Proof: The lower bound follows from theorem 43. Applying lemma 52, one can use the analysis of GEOMETRIC ROTARY to get the desired bound. ■

Acknowledgments: We thank Taieb Znati for pointing out to us the need for a theoretical analysis of fault tolerant scheduling. We also thank Joel Wein, Sundar Vishwanathan, Sunondo Ghosh and Daniel Mosse´ for helpful discussions. Finally, we would like to thank the referees for many useful comments and suggestions.

References

[1] N. Alon and J. H. Spencer, *The Probabilistic Method,* John Wiley & Sons, Inc., 1992.

[2] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson, "On the Power of Randomization in On-line Algorithms," *Algorithmica*, **11**, 2–14, 1994.

[3] A. Borodin and R. El-Yaniv, "On Randomization in On-line Computation", *Proceedings of IEEE Conference on Computational Complexity*, 1996.

[4] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[5] A. Feldmann, J. Sgall, S. Teng, "Dynamic scheduling on parallel machines", *Theoretical Computer Science*, **130**, 49–72, 1994.

[6] R. Graham, E. Lawler, J. Lenstra, and A Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey", *Annals of Discrete Mathematics*, **5**, 287 – 326, 1979.

[7] T. Matsumoto, "Competitive analysis of the round robin algorithm', *International Symposium on Algorithms and Computation*, 71 – 77, 1992.

[8] R. Motwani, S. Phillips, E. Torng, "Non-clairvoyant scheduling", *Theoretical Computer Science*, **130**, 17–47, 1994.

[9] D. Sleator, and R. Tarjan, "Amortized efficiency of list update and paging rules", *Communications of the ACM*, **28**, 202 – 208, 1985.

[10] D. Shmoys, J. Wein, and D. Williamson, "Scheduling parallel machines on-line", *Proceedings of IEEE Symposium on Foundations of Computing*, 131 – 140, 1991.

[11] J. Vytopil (ed.), *Formal Techniques in Real-time and Fault-tolerant Systems*, Kluwer Academic Publishers, 1993.

[12] A. Yao, "Probabilistic computations: towards a unified measure of complexity", *Proceedings of IEEE Symposium on Foundations of Computing*, 222 – 227, 1977.