computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
**International Scientific Journal of Computing**

# ANALYTICALLY MODELING UNRELIABLE PARALLEL PROCESSING SYSTEMS WITH GENERAL TASK TIME DISTRIBUTIONS

**Pierre M. Fiorini [1], Robert W. Rowan [2]**

[1] University of Southern Maine, Department of Computer Science, Portland, ME, pfiorini@usm.maine.edu
[2] Nationwide Payment Solutions, 400 US Rt. 1, Falmouth, ME 04105, wrowan@getnationwide.com

**Abstract**: *For many computing systems, failure is rare enough that it can be ignored. In other systems, failure is so common that the recovery procedure can have a significant impact on the performance of the system. In this paper, assuming a computing system is unreliable, we discuss how heavy-tail or power-tail job completion time distributions can appear in an otherwise well-behaved task stream. This is an important consideration since it is known that power-tails can lead to unstable systems. We then demonstrate how to obtain performance and dependability measures for a class of computing systems comprised of P unreliable processors and a finite number of tasks, N, given different recovery policies. Finally, we discuss the effects of checkpointing on the job completion time distribution.*

**Keywords:** *Performance and Dependability Modeling, Parallel & Distributed Systems, Queueing Theory, Heavy-Tails*

## 1. INTRODUCTION

This paper discusses an analytic approach that can be used to compute expected performance, dependability, and performability measures for unreliable parallel processing systems (PPS) given the following recovery policies: *Resume* (*prs* – preemptive resume same), *Replace* (*prd* – pre-emptive repeat different), and *Restart* (*pri* pre-emptive repeat identical), although other recovery policies could be considered. We also show how *heavy-tail* or *power-tail* job completion time distributions can appear in an otherwise well-behaved task stream. This is significant since it is well known that power-tails can lead to unstable job completion times [5] [6]. The PPS can be comprised of any number of processing elements (PEs) and tasks. We assume that failure and repair rates are exponentially distributed, while task service times can be generally distributed.

The model consists of a queueing system that characterizes the system's performance, dependability, and recovery policy. The model is solved at task completion points from which performance, dependability, and performability measures can be ascertained. One well-known issue with these problems is the potentially large state-space required to characterize these types of analytic models. The technique described in this paper addresses the state space problem in two ways: 1) we utilize an *epoch* approach; and, 2) take advantage of symmetry in the system structure and task stream by assuming the tasks in the job stream are homogenous enabling us to use a *reduced Kronecker product space*.

With the epoch approach each task completion in the job is analyzed independently. We calculate the distribution and expected values of various metrics for each epoch, then sum over all epochs to obtain cumulative metrics for the entire job. This approach significantly reduces the state space since it requires that only state information of the current epoch be stored. Using this approach, performance measures such as the mean job completion time can be calculated as well as dependability measures such as the *system availability*, the *mean time to failure* (MTTF), the *mean time between failures* (MTBF), and *the mean time to repair* (MTTR) given a recovery policy. Furthermore, the *Work* done by the system, which is performability measure, can also be computed. Our base system can be thought of as a PPS with *P* identical processing elements. The base workload on the system is comprised of *N independent and identically distributed* (iid) tasks which are all present at time $x = 0$. The service time of each task has a *matrix exponential* (ME) representation given by an *m*-dimensional vector-matrix pair, <**p**, **B**> (see [1]), such that its *Cumulative distribution function*, CDF, is given by:

$$F(X) = \Pr(X \le x) = 1 - \mathbf{p}\exp(-x\mathbf{B})\mathbf{e}'$$

where $\varepsilon'$ is an *m*-dimensional column-vector of ones.

## 2. PREVIOUS WORK

The analysis of completion times when the policy was *Resume* or *Replace* was carried out by Kulkarni et. al. [2]. The analysis of mixed polices was also carried out by Kulkarni et. al. [3]; and, the task distribution for the *Restart* policy was examined by Kulkarni et. al. [2]. The work by Kulkarni et. al. [2] [3], and Bobbio and Trivedi [4] clearly suggested that the resulting service time for the *Resume* and *Replace* resumption policies could be represented by ME distributions; however, they found that the distribution of the *Restart* policy (whatever it was) could not.

Regarding our analytic model, the work of Bobbio and Trivedi is among the most relevant [4]. It examines a system whose work requirement for jobs can be represented by a *phase distribution* (PH) and focuses on computing the distribution of the completion time of the job. Our approach, by contrast, is intended to investigate the expected behavior of a job running on a system at various points in the task stream from which performance and dependability measures can be generated.

Of interest in this paper is the demonstration of how heavy-tail (hereafter referred to as *power-tail*) distributions can appear in a distribution that has an ME representation (i.e., any distribution with a *rational LaPlace transform* (LPT) ). This is important since it is well known that power-tails can lead to unstable systems and job completion times [5] [6]. Furthermore, assuming the computing system is unreliable and the recovery policy is *Restart*, we demonstrate how this behavior occurs and how it can be modeled.

Additionally, a number of researchers have observed that much computer system related phenomena (e.g., CPU process lifetimes) exhibit properties consistent with power-tail distributions [7]. Some researchers have suggested the distribution of run times for jobs in parallel processing systems are power-tail distributed. Interestingly, one reason why most sites have not observed this behavior is because of limitations on the allowed length of a job [8]. Consequently, users that need to run very long jobs resort to making a checkpoint whenever they run out of time, and then restart the job later. This precludes observations of job runtimes that are in the tail of the distribution.

Applications of our model include performance and dependability measures for unreliable distributed and parallel systems. For instance, in some distributed applications when jobs running on hosts fail they must "*Restart*" elsewhere, which causes job times to be power-tail distributed if certain conditions are satisfied.

## 3. DISTRIBUTION OF NUMBER OF FAILURES FOR TASKS

In the following sections we summarize various properties of the *Resume*, *Replace*, and *Restart* resumption policies for tasks in a job stream. First we supply some definitions.

Let $T$ be the *random variable* (rv) denoting the time for a task to execute without failures, with *probability density function* (pdf), $f(t)$ and mean, $E(T) = \tau$. Let $X$ be the rv denoting the total time a task spends executing because of failures, but not including the time it spends waiting. The pdf for $X$ is $g(x)$. Let $t$ be the time needed by a particular task, i.e., let $T = t$. We assume that failures occur randomly with exponential inter-occurrence times, with parameter, $\beta$. That is, $1/\beta$ is the mean time between failures (MTBF). Then

$$d(\beta) = \int_0^\infty e^{-\beta t} f(t)dt$$

is the probability that a task will be fail at least once. It also happens to be the *LaPlace* LPT of $f(x)$ evaluated at $\beta$. In our further discussions, we let

$$\gamma = \beta\tau, \text{ and } \lambda = 1/\tau.$$

In Table 1, the distribution of the number of failures, first for a particular task of time $t$, then averaged over all task times is shown. We give the values with exponentially distributed inter-occurrence times, which can be used to find the total time spent by a task in waiting for the failure to end.

**Table 1. Distribution of Number of Failures**

| **Object** | *Resume* | *Replace* | *Restart* |
|---|---|---|---|
| $P(n \mid t) =$ $P[N(t) = n]$ | $\frac{(\beta t)^n}{n!} e^{-\beta t}$ | $e^{-\beta t}(1-e^{-\beta t})^n$ | $e^{-\beta t}(1-e^{-\beta t})^n$ |
| $E[N(t)]$ | $\beta t$ | $(e^{-\beta t}-1)$ | $(e^{-\beta t}-1)$ |
| $P(n)$ | $\frac{\beta^n}{n!}\int t^n e^{-\beta t} f(t)$ | $[1-d(\beta)][d(\beta)]^n$ | $\int e^{-\beta t}(1-e^{-\beta t})^n f(t)dt$ |
| Mean (*general*) | γ | $\frac{1-d(\beta)}{d(\beta)}$ | $\int e^{-\beta t} f(t)dt - 1$ |
| $P(n)$ (*exponential*) | $\left[\frac{1}{1+\gamma}\right]\left[\frac{\gamma}{1+\gamma}\right]^n$ | $same\left(d(\beta) = \frac{1}{1+\gamma}\right)$ | $\sum_{\ell=0}^{n} \binom{n}{\ell} \frac{(-1)^\ell}{1+(\ell+1)\gamma}$ |
| Mean (*exponential*) | γ | γ | $\left[\frac{\gamma}{1-\gamma}\right], \quad \gamma < 1$ |

If a task resumes where it left off, and times between failures are exponentially distributed (the assumption in this paper), then the arrival of failures is a Poisson process. Since this is also a memoryless process, the time it takes for the task to continue does not affect the time until the next failure. Therefore, the number of failures in time $t$ satisfies the Poisson distribution, as shown in the table. For *Replace*, if each task is replaced by one of the same length, then the number of failures is geometrically distributed, with probability $(1 - e^{-\beta t})$ that a failure will occur before the task finishes. This is the same for *Restart*, since the task is always "replaced" by itself after a failure.

## 4. DISTRIBUTION OF THE NUMBER OF FAILURES FOR TASKS GIVEN THE *Restart* POLICY

Table 2 gives formulas for the time spent by a given task in actually using the resource. Recall that *Replace* involves several different tasks (one for each failure), whereas the other two involve the same task. In some cases, the best we can do is to find the LPT of the distribution, and this allows us to find the mean and variance.

The *Restart* procedure requires additional explanation. Suppose that a task has a time $t$ remaining. Then the probability density that the task will fail at time $x$, given a failure occurs is given by:

$$h(x \mid t) = \frac{\beta e^{-\beta t}}{1 - e^{-\beta t}}, \quad \text{for} \quad 0 \le x \le t.$$

Its LPT is

$$H^*(s \mid t) = \int_0^t e^{-st} h(x \mid t) dx = \frac{\beta}{\beta + s}\left[\frac{1 - e^{-(\beta+s)t}}{1 - e^{-\beta t}}\right].$$

The density function $g_n(x|t)$ for the time the task would take to finish, given that it failed $n$ times before it ran successfully, can by found by taking the convolution of $h(x|t)$ with itself $n$ times, and then with $\delta(x - t)$. This can be done since it is well known that the LPT of a convolution of functions is equal to the product of the transforms. Thus, we can find $G_n^*(s|t)$, the LPT of $g_n(x|t)$, since we already know $H^*(s|t)$. We can then average over the number of failures, using $P(n|t)$ from Table 1. This yields an explicit expression, as shown in Table 2. The LPT of the distribution of the time to finish a *Restart* task can be found by averaging over the task-time distribution. That is,

**Table 2. Distribution of Number of Failures for** ***Restart***

| **Object** | *Restart* |
|---|---|
| $g_n(x \mid t)$ (Fixed $t$) | $G_n^*(s \mid t) = \left[H^*(s \mid t)\right]^n e^{-st}$ |
| $g_n(x \mid t)$ (Fixed $t$) | $G^*(s \mid t) = \sum_{n=0}^{\infty} P(n \mid t)\, G_n^*(s \mid t) = \frac{(\beta + s)\, e^{-(\beta+s)t}}{s + \beta\, e^{-(\beta+s)t}}$ |
| Density, $g(t)$ (*general*) | $G^*(s) = \int_0^\infty G^*(s \mid t) f(t) dt$ |
| Density (*exponential*) | $G^*(s) = \lambda \int_0^\infty G^*(s \mid t)\, e^{-\lambda t}\, dt$ |
| Mean (*general*) | $-\int_0^\infty G^{*'}(s = 0 \mid t) f(t) dt$ |
| Mean (*exponential*) | $\frac{\tau}{1-\gamma}, \gamma < 1$ |
| $E(X^2)$ | $\int_0^\infty G^{*''}(0 \mid t) f(t) dt$ |

$$G^*(s) = \int_0^\infty G^*(s \mid t) f(t) dt = \int_0^\infty \frac{(\beta + s)\, e^{-(\beta+s)t}}{s + \beta\, e^{-(\beta+s)t}} f(t) dt$$

It is not possible to find $g(t)$ itself, since $f(t)$ is not specified, and in any case, we have not been able to take the inverse LPT of $G^*(s)$. However, as well known, the LPT is a *Moment generator*, so,

$$E(X) = -\left[\frac{d}{ds} G^*(s)\right]_{s=0} = \int_0^\infty \left[\frac{d}{ds} G^*(s|t)\right]_{s=0} f(t) dt$$

An explicit expression for $G^{*'}(s|t)$ can be found from Table 2:

$$G^{*'}(s \mid t) = \frac{d}{ds} G^*(s \mid t) = \frac{d}{ds} \frac{(\beta + s)\, e^{-(\beta+s)t}}{s + \beta\, e^{-(\beta+s)t}}$$
$$= \frac{\left[\beta + s(s + \beta)\, e^{-(s+\beta)t}\right] e^{-(s+\beta)t}}{\left[s + \beta e^{-(s+\beta)t}\right]^2}$$

For $s = 0$, this reduces to a simple expression, namely:

$$G^*(s = 0 \mid t) = -\frac{e^{\beta t} - 1}{\beta}$$

Observe the positive exponential function in the numerator. Thus, if follows for any function,

$$E(X) = \int_0^\infty \frac{e^{\beta t}-1}{\beta} f(t)dt \qquad (1)$$

Clearly, $E(X)$ must be infinite for all distributions that go to zero more slowly than exp(-$\beta t$).

A similar situation occurs for $E(X^2)$, and thus for the variance of $g(t)$. Without going through the details, it can be shown that

$$G^{*''}(s=0\,|\,t) = \frac{2}{\beta^2}\left[e^{2\beta t} - (1+\beta t)\, e^{2\beta t}\right], \qquad (2)$$

and since

$$E(X^2) = \int_0^\infty G^{*''}(0\,|\,t) f(t)dt,$$

\

it is clear that the variance must be infinite for any function that goes to 0 more slowly than exp(-2$\beta t$).

Since $g(x)$ is the pdf of the time needed to complete a task with *Restart*, it is the inverse LPT of $G^*(s)$. Now let $R(x)$ be the *Reliability* or *Complementary distribution function* for $X$. That is,

$$R(x) := P(X > x) = \int_x^\infty g(x)dx.$$

It can be seen that each successive derivation of $G^*(s)$ introduces another factor of exp(-$\beta t$), so

$$G^{*(n)}(0\,|\,t) \to c\; e^{n\beta t},$$

implying that if

$$\lim_{x\to\infty} e^{\lambda x} f(x) = \infty \qquad (3)$$

for some $\lambda > 0$, then $g(x)$ has infinite moments. Now suppose that $\lambda_{\max} > 0$ has the property that for all $\lambda > \lambda_{\max}$, the above equation is true, but for $\lambda < \lambda_{\max}$, the limit is bounded. Next let

$$\alpha := \lambda_{\max} / \beta \qquad (4)$$

Then for all $\ell \ge \alpha$

$$\int_0^\infty x^\ell\, g(x)dx = \infty. \qquad (5)$$

But this is just the property of power-tail distributions. Since we are not able to find $g(x)$ from $G^*(s)$ at this time, we claim that

$$\lim_{x\to\infty} R(x) \Rightarrow \frac{c}{x^\alpha} \qquad (6)$$

## 5. SIMULATIONS OF TASKS WITH THE *Restart* POLICY

To test our results concerning the behavior of tasks that must *Restart*, Sheahan et. al. [9] carried out a set of simulation runs for three different functions, $f(t)$, all with mean time $\tau$ = 1, but with different variances $\sigma^2$. They are: the Erlangian-3 ($E_3$)

$$f_{E_3}(t) = \frac{3(3t)^2}{2} e^{-3t}, \quad \text{with} \quad \sigma^2_{E_3} = 1/3,$$

the exponential (with $\sigma^2$ = 1), and Hyperexponential-2 ($H_2$), given by:

$$f_{H_2}(t) = p\,\lambda_1 e^{-\lambda_1 t} + (1-p)\,\lambda_2 e^{-\lambda_2 t}$$

where $p = (3+\sqrt{6}) = 0.9082848\ldots$, $\lambda_1 = 2p$ = 1.816..., and $\lambda_1 = 2(1-p)$ = 0.1835034... .

The simulation of *Restart* operated as follows. First, the failure time distribution and the task time distribution are initialized. For the examples here, the failure time distribution is exponential and the failure rate, $\beta$, is 0.5. The task time distributions and parameters used here have all been chosen to have a mean time of 1.0 to illustrate that the shape not the average is the distinguishing feature. All simulation runs examined $10^7$ tasks.

The best way to illustrate power-tail behavior is to plot them on log-log scale. For instance, consider power tail functions of the form in Equation (6). Taking the logs of both sides we get:

$$\lim_{x\to\infty} \log[R(x)] \Rightarrow \log(c) - a\log(x).$$

That is, $R(\cdot)$ functions approach a straight line with slope $-\alpha$ when viewed on log-log scale. This is seen in Fig. 3, where only the *Restart*-time distributions, coming from the task time distributions $E_3$, $M$, and $H_2$ (i.e., the functions with exponential tails), have this behavior. From the definition of $\lambda_{\max}$ earlier, it follows that

$$\lambda_{\max} = 3.0,\ 1.0,\ 0.184...,$$

for $E_3$, $M$, and $H_2$ respectively.

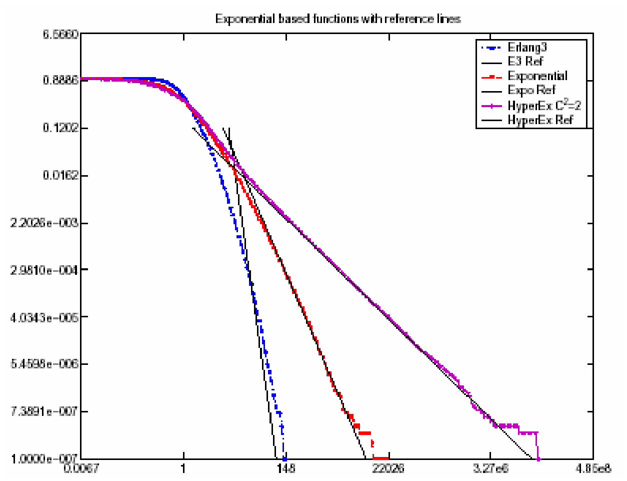

**Fig. 1 - Illustrating the power-tailed behavior of the *Restart* or *pri* recovery policy.**

When $\beta = 0.5$, then:

$$\alpha = 6.0,\ 2.0,\ 0.367001...,$$

for $E_3$, $M$, and $H_2$ respectively.

The three PT curves, together with their respective straight line asymptotes are plotted in Fig. 1, which demonstrates the asymptotic behavior of these functions, which is clearly power-tail.

One objective of this work was to employ a ME representation for the *Restart* recovery distribution so that performance, dependability, and performability measures could be generated using analytic queueing models for jobs that implemented this policy. By constructing the state transition diagram for the *Restart* recovery policy, it can be shown that there is no ME representation since the number of states required to characterize this process is infinite [11]. Given this, our approach was to utilize a suitable ME approximation that asymptotically emulates important statistical properties of PT distributions. To do this we used *Truncated power-tail* (TPT) distributions appropriately parameterized with $\alpha$ and $\tau$, which do have ME representations [12]. Essentially, TPTs are Hyperexponential distributions that are "truncated" after a predetermined number of exponential phases. Their ME representation is given by <**p**($T$), **B**($T$)>, where $T$ is the *truncation parameter* or the number of phases in the Hyperexponential distribution. The entrance vector, **p**($T$), can be constructed by

$$\mathbf{p}(T) = \frac{1-\theta}{1-\theta^2}\left[1,\ \theta,\ \theta^2, \ldots, \theta^{T-1}\right],$$

and the generator **B**($T$) by

$$\mathbf{B}(T) = \mu \cdot \mathrm{diag}\left[1,\ \frac{1}{\gamma},\ \frac{1}{\gamma^2}, \ldots, \frac{1}{\gamma^{T-1}}\right].$$

The mean for <**p**($T$), **B**($T$)>, $1/\mu(T)$, can be fixed using

$$\mathbf{p}(T) = \frac{1-\theta}{1-\gamma\theta} \cdot \frac{1-(\theta\gamma)^T}{1-\theta^T},$$

Important parameters of the TPT distribution are $\alpha$ and $\gamma$, which can be set (and determined) by the relation

$$\theta\gamma^{\alpha} = 1,\ \ or\ \ \alpha = -\frac{\log(\theta)}{\log(\gamma)},$$

where $\mu$ (in practice) is usually fixed to 0.5. Observe that $\alpha$ can be determined by appropriate use of Equation (4).

Another consideration we have addressed is that although the asymptotic properties of *Restart* case can be reasonably modeled using TPT distributions, its non-asymptotic behavior cannot. To handle this, we have investigated approaches that fit the non-asymptotic properties of the *Restart* distribution to an ME distribution and combine this with a TPT distribution, which captures its asymptotic behavior .

## 6. GENERATING PERFORMANCE, DEPENDABLITY AND PERFORMABILITY MEASURES

Recall that our base system can be thought of a parallel system consisting of $P$ identical processing elements executing a finite set $N$ independent and iid tasks as shown in Fig. 2. We assume the service time distributions for tasks can be represented by some $m$-dimensional ME vector-matrix pair, <**p**, **B**> [1].

To generate expected performance and dependability measures for unreliable computing systems, we use a G/C queueing system that incorporates the recovery policy of the system and task service times that can be generally distributed. The model is then solved at task completion points from which performance, dependability, and performability measures can be calculated.

As mentioned earlier, one well-known issue is the potentially large state space required to characterize these types of problems. We address the state space problem in two ways: 1) we utilize an *epoch* approach; and, 2) take advantage of the symmetry in system structure and task stream by assuming tasks in the job are homogenous, which enables us to use a *reduced product space*.

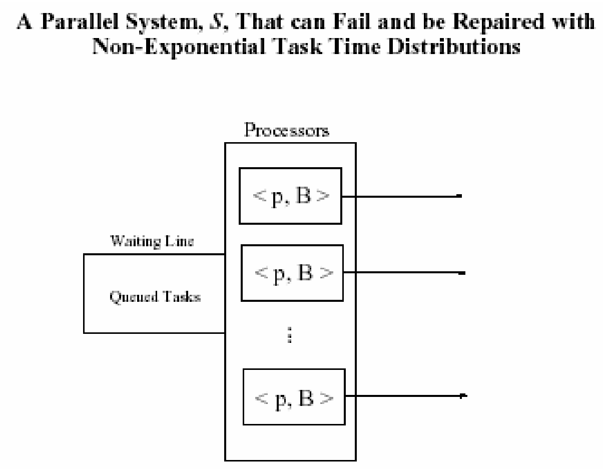
A Parallel System, S, That can Fail and be Repaired with Non-Exponential Task Time Distributions
Processors
< p, B >
< p, B >
< p, B >
Waiting Line
Queued Tasks

**Fig. 2 - This figure shows our base model, a parallel processing system with non-exponential task time distributions where failures and repairs can occur.**

We define an epoch as the period between the completion of one task and the completion of the next task (i.e., these are the embedding points - see Fig. 3). Except at the end of a job, there are usually *P* processors and tasks running during an epoch. If one formulated this as a *direct product* (or *Kronecker product*) space, then one would need $m_r^P$ states to represent all possibilities during an epoch. However, in our model we assume that processors and tasks are statistically identical, thus one does not need to keep track of which processor/task failed, but merely how many processors are running, and how many tasks are in each state of execution. In our method we treat each epoch separately (taking due account of the dependence of each epoch on the previous one).

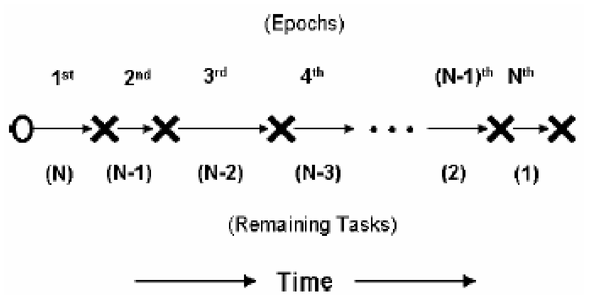

**Fig. 3 - The job starts at time zero. Each "X" represents a task completion. When the final task completes, the job is completed.**

One could argue that the epoch approach provides less information than the approach that directly integrates the *Chapman-Kolmogorov* (CK) formula, which is one way to handle these types of problems; however, this is not true in principle. The two domains can be thought of duals of each other, analogous to a function and its LPT. For instance, if one specifies the time that a system has been running, then one gets from the CK solution the *probability* that the $\mathtt{l}$th task has finished. If one specifies the epoch ($\mathtt{l}$ – 1) tasks have already been completed), then one gets the *mean* time it took to get there.

By extendeing results from the G/C queue (namely, suitable modifications for the mean time for the queue to drain – see [1] and [9]), we define a vector-matrix pair $<\wp(\mathtt{l}), \mathcal{B}(\mathtt{l})>$, that characterizes the distribution of the completion time for the $\mathtt{l}$th epoch, where $\wp(\mathtt{l})$ is the entrance vector and $\mathcal{B}(\mathtt{l})$ is the infinitesimal generator matrix of the completion process for the $\mathtt{l}$th epoch. We construct $\mathcal{B}(\mathtt{l})$ from the underlying Markov chain and service time distribution.

The epoch number, $\mathtt{l}$, denotes the number of tasks remaining in the system. We combine this information with that garnered from the state space to represent the system. Hence, the state-space tells us how many processing elements (PEs) have and have not failed so that we can determine how many PEs are busy, how many PEs are idle, and how many PEs are down.

The inverse of the $\mathcal{B}(\mathtt{l})$ matrix is the service time matrix, $\mathcal{V}(\mathtt{l})$. Elements of $[\mathcal{V}(\mathtt{l})]_{ij}$ represent the amount of time during the $\mathtt{l}$ th epoch that the system spends in state *j*, given the system began the epoch in state *i*. The service time matrix is the inverse of $\mathcal{B}(\mathtt{l})$; that is, $\mathcal{V}(\mathtt{l}) = [\mathcal{B}(\mathtt{l})]^{-1}$. To calculate the entrance vector, $\wp(\mathtt{l})$, we compute the *conditional probability transition matrix*, $\mathcal{Y}(\mathtt{l})$, by

$$\mathcal{Y}(\mathtt{l}) = \mathcal{V}(\mathtt{l})\mathcal{M}(\mathtt{l})\mathcal{Q}(\mathtt{l})\mathcal{R}(\mathtt{l}),$$

where $\mathcal{M}(\mathtt{l})$ is the *transition rate matrix*. $\mathcal{Q}(\mathtt{l})$ represents the state transition on an epoch completion and $\mathcal{R}(\mathtt{l})$ represents state transitions due to a task arrival from the queue. Their construction depends upon the recovery policy, the task distribution, and the modeling situation.

the entrance vector, $\wp_I = \wp(1)$, can be determined from the modeling situation and $\wp(\mathtt{l})$ is calculated by the following:

$$\wp(\mathtt{l}) = \wp(\mathtt{l} - 1)\mathcal{Y}(\mathtt{l})$$

Finally, we can compute $T(\mathtt{l})$, the mean time to complete the $\mathtt{l}$th epoch,

$$T(\mathtt{l}) = \wp(\mathtt{l})\, \mathcal{V}(\mathtt{l})\varepsilon',$$

and $T_N$, the mean time to finish all *N* tasks,

$$T_N = \sum_{\ell=1}^{N} T(\ell).$$

Expected dependability (i.e., the system availability, the mean time to fail (MTTF), the mean time between failures (MTBF), the mean time to repair (MTTR) ), and performability measures (i.e., *Work*) can be calculated in a similar manner with suitable matrix operators. For more information, the reader is referred to Rowan [10].

## 7. AN EXAMPLE – A SINGLE PROCESSOR THAT CAN FAIL AND BE REPAIRED

In this section, we derive an analytic formulation to compute the mean time to complete a job consisting of *N* tasks, with independent and identically distributed non-exponential completion times, running on 1 processor that can fail and be repaired. We consider the *Resume*, *Replace*, and *Restart* policies.

In all cases, the non-exponential task times are represented by an *m*-phase ME distribution which is characterized by the vector-matrix pair <**p**, **B**>, where **p** is a row vector of dimension *m* and **B** is a square matrix of size *m* × *m* (see [1]).

Processor failures occur at rate $\beta$, which have exponentially distributed inter-occurrence times. When the processor fails, it gets repaired at rate $\alpha$, which is also exponentially distributed.

The behavior of these systems is as follows. Suppose the processor has *N* tasks to complete and assume the processor is operational when the job begins. The system services tasks for some period of time until either the job (all the tasks) completes or a failure occurs (at rate $\beta$). When this happens, no more tasks can complete until the system is repaired and this occurs at rate $\alpha$. Once this happens, the processor resumes servicing tasks. This cycle continues until all *N* tasks have finished.

The differences between the *Resume*, *Replace*, and *Restart* policies are characterized by the matrix representations shown below. For example, the completion rate matrix, $\mathbf{B}_{Resume}$, for the `l`th epoch for tasks executing on a single processor, which after a failure, continues later where it left off (i.e., the *Resume* policy) is

$$\mathcal{B}_{Resume}(l) = \begin{bmatrix} \mathbf{B}+\beta\mathbf{I} & -\beta\mathbf{I} \\ -\alpha\mathbf{I} & \alpha\mathbf{I} \end{bmatrix}.$$

Given a failure, the matrix representation of a task using the *Replace* policy after a failure (a new task time from the distribution) is given by

$$\mathcal{B}_{Replace}(l) = \begin{bmatrix} \mathbf{B}+\beta\mathbf{I} & -\beta\mathbf{I}\mathbf{e}' \\ -\alpha\mathbf{p} & \alpha \end{bmatrix}.$$

The matrix representation of a single task given the *Restart* policy is

$$\mathcal{B}_{Restart}(l) = [\mathbf{B}_{TPT}],$$

where $\mathbf{B}_{TPT}$ is an ME representation of a power-tail distribution.

The service time matrices for the *Resume* and *Replace* policies can be computed as $\mathcal{V}_{Resume/Replace/Restart}(\mathtt{l}) = [\mathcal{B}_{Resume/Replace/Restart}(\mathtt{l})]^{-1}$. To calculate performance measures for both the *Resume* and *Replace* policies, the following matrices and matrix operators are required.

Following Section 6, the conditional completion probability matrices need to be determined, and they can be defined for the *Resume* and *Replace* resumption policies by

$$\mathcal{Y}(\mathtt{l}) = \mathcal{V}(\mathtt{l})\mathcal{M}(\mathtt{l})\mathcal{Q}(\mathtt{l})\mathcal{R}(\mathtt{l}),$$

where $\mathcal{M}$ is the state departure rate matrix defined by

$$\mathcal{M}_{Resume}(l) = \mathrm{diag}(\mathbf{B}+\beta\mathbf{I}, \beta\mathbf{I}),$$

and

$$\mathcal{M}_{Replace}(l) = \mathrm{diag}(\mathbf{B}+\beta\mathbf{I}, \beta)$$

It is important to remark that with the *Restart* case, for all `l`, $\mathcal{Y}(\mathtt{l})$ is equivalent to the **Q** matrix as defined in [1] (i.e., $\varepsilon' \wp_{\mathbf{1},Restart}$ – see below).

$\mathcal{Q}$ is a conditional probability matrix representing the state of the system upon a task (or epoch) completion for all policies. It is a column vector of dimension [(2*m*) × 1] in the *Resume* scenario and [(*m* + 1) × 1] in the case. Assuming task times in the job are Hyperexponentially distributed and with both recovery policies, the *i*th element of $\mathcal{Q}$ is equal to $\mu_i = (m\mu_i + \beta)$, which is the probability of a task completion prior to failure given the system is in internal state *i* for all $1 \le i \le m$ and is 0 otherwise.

Lastly, $\mathcal{R}$ represents the transition of the internal state of the system upon an arrival. $\mathcal{R}$ is a row vector of dimension [1 × (2*m*)] (*Resume*) and [1 × (*m* + 1)] (*Replace*) where the *i*th element is equal to $[\mathbf{p}]_i$ (the probability of an entering task entering phase *i*).

Observe that in the single-server case, both $\mathcal{Q}$ and $\mathcal{R}$ are one dimensional matrices. However, when more PEs are modeled in the system, these matrices become two dimensional.

Before performance measurements can be calculated, the initial state vector, $\wp_{\mathbf{1}}$, representing the initial state of these systems must be determined.

Assuming the processor is operational when the job begins (other assumptions are valid as long as $\wp_{\mathbf{I}}\,\boldsymbol{\varepsilon}' = 1$), we have

$$\wp_{\mathbf{I},Resume} = [\mathbf{p}\ \mathbf{o}];\ \ \wp_{\mathbf{I},Replace} = [\mathbf{p}\ \mathbf{o}];\ \ \wp_{\mathbf{I},Restart} = [\mathbf{p}_{TPT}],$$

where **o** is an *m* dimensional row vector of 0's.

To compute performance measures (i.e., the mean time for *N* tasks in a job to complete), suitable modifications regarding the mean time for to drain a *finite* G/1 queue are utilized (see Section 6, [1], and [9]).

## 8. NUMERICAL EXAMPLES

In this section, we present some examples using our analytic model for two non-exponential task time distributions: 1) Hyperexponential distributions; and, 2) *m*-phase TPT distributions. Fig. 4 shows various systems where the workload having task execution times sampled from an Hyperexponential-8 ($H_8$) distribution and a *Replace* task resumption policy. The number of tasks in the workload equals the number of processing elements (PEs) in the system (i.e., $N = P$), and that value is varied and the job completion time is shown for three different failure rates.

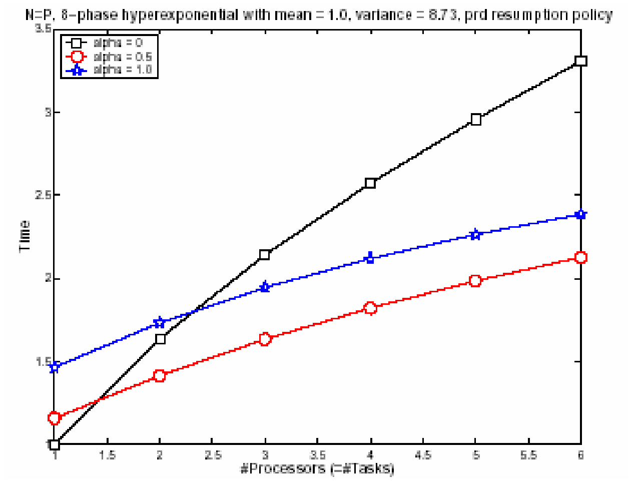

**Fig. 4 – Performance measures using the *Replace* resumption policy where the workload has task execution times sampled from an $H_8$ distribution**

Comparing the curve when the failure rate is 0 (i.e., no failures) with the curves with failures shown in Fig. 4 demonstrates one property of the *Replace* resumption policy; performance can improve when failures are introduced into the system. This is because tasks from the longer phase are more likely to fail (simply because they take longer to execute), and when the failed task is restarted it has a good chance of restarting in a faster phase. Observing the two lines with failures, we see this property holds until performance degrades with increasing failure rate.

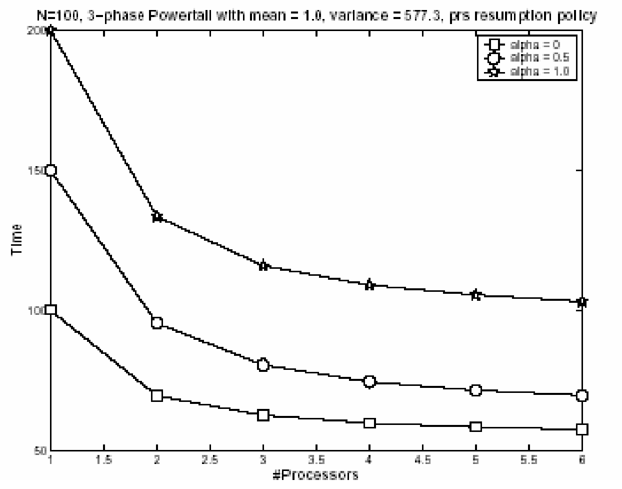

**Fig. 5 – Performance measures using the *Replace* resumption policy where the workload has task execution times sampled from an TPT distribution**

Fig. 5 shows various systems with a workload comprised of 100 tasks sampled from a 3-phase TPT distribution with a *Resume* task resumption policy. Again, the expected job completion time is shown as a function of the number of processing elements in the system for three different failure rates.

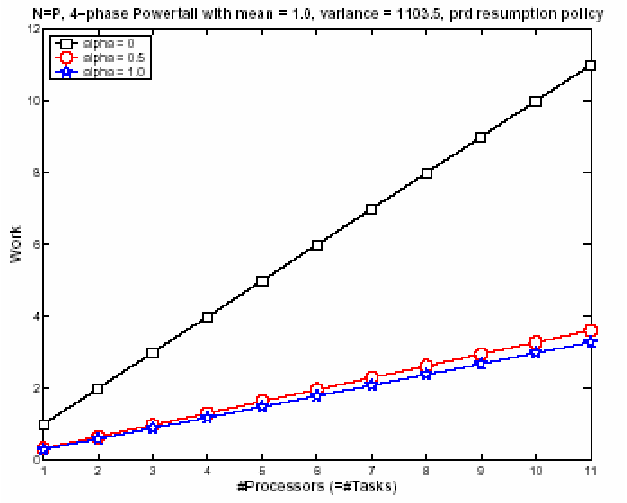

**Fig. 6 – The expected work done by the system as a function of PEs for various failure rates where the number of tasks in the workload is equal to the number of processors. The task execution times are sampled from a 4-phase TPT distribution.**

Fig. 6 shows expected work done by the system as a function of the PEs for various failure rates. We are again looking at a system where the number of tasks initially in the workload is equal to the number of processors. The task execution times are sampled from a 4-phase TPT distribution with a *Replace* resumption policy.

We again see the result of a decrease in the amount of work done by the system when failures are introduced. This is again due to the fact that the tasks with long execution times are likely to fail and be replaced (when restarted) by a task with a shorter execution time from another phase. Notice that we now have a strict decrease in the amount of work done by the system with increasing failure rate.

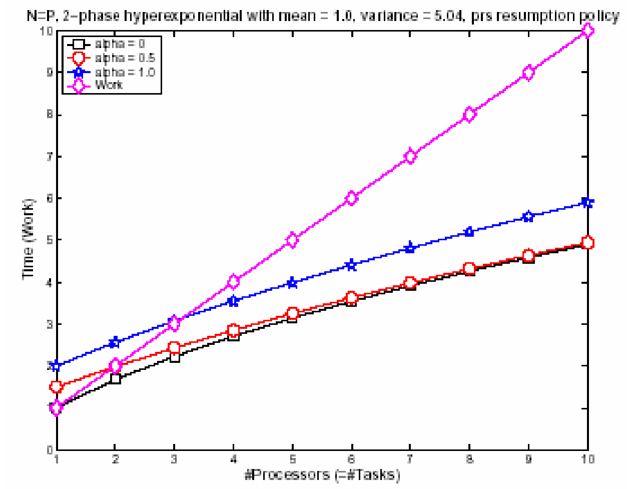

**Fig. 7 – The expected work done by the system as a function of PEs for various failure rates where the number of tasks in the workload is equal to the number of processors. The task execution times are sampled from a 4-phase TPT distribution.**

Fig. 7 shows both expected work done by the system and expected job completion time as a function of PEs for various failure rates. We are again observing a system where the number of tasks initially in the workload is equal to the number of processing elements in the system. The task execution times are sampled from a $H_2$ distribution with a *Resume* resumption policy.

Interestingly, we observe that work is independent of the failure rate for the *Resume* task resumption policy, despite a strict increase in the expected job execution time with increasing failure rate. We attribute this to the Markovian property of each phase. Once each task starts, it executes with some exponentially distributed rate until completion. If the task fails, the Markovian property tells that, on average, the time remaining is equal to the expected time of a new sample. That is, when the task resumes the new completion time is equal to the time that had been remaining prior to failure, on average. The result is that expected work done by the system is independent of failures since work neglects time spent by tasks in the waiting queue.

## 9. ON-GOING WORK – CHECKPOINTING AND TASK-TIME DISTRIBUTIONS

The purpose of checkpointing is to prevent a task from having to start again from the beginning if it fails. This involves interrupting the task, recording its internal state, and then resuming where it left off. If, at any time before the next checkpoint operation, the task should fail, then it can restart later at the most recent checkpoint. This may turn out to be a costly procedure and aversely affect system performance. Thus, the question how often checkpointing should be done, or even if it should be implemented at all is a legitimate concern. Clearly, the results of this paper indicate that not checkpointing can be destabilizing for tasks (and jobs comprised of 1 or more tasks) that are required to finish.

One possibility investigated by [13] [14] would be for a task to checkpoint a fixed number of times. This would make sense for a task or job which is itself made up of a fixed number of *sub-tasks*. Then the task could checkpoint immediately after the execution of each sub-task. But if the distribution of task times is ME then at least one of the sub-tasks will be as well. Thus, the power-tail behavior of $X$ will not go away. Indeed, [13] showed that the mean time to complete the task (with $k$ sub-tasks) would be

$$\overline{T}(t \mid k) = k \cdot \frac{e^{\beta \frac{t}{k}} - 1}{\beta}.$$

Thus, if we averaged over all task times, it can be shown that the effective $\alpha$ becomes larger. For instance, suppose that the tasks can be executed with $k$ checkpoints per task. Then, $\alpha$ for the sub-task time distributions would be scaled accordingly; that is,

$$\alpha \approx k \times \lambda_{\max}.$$

Observe that power-tail behavior is still present using this checkpointing strategy since the existence of $\alpha$ implies the task time distribution has infinite moments (see [13]).

Another possibility would be for the task to be interrupted at fixed intervals of time $\Delta$. It can be shown that the mean time to complete the task or job using this approach, averaged of all possible tasks times, is proportional to

$$\overline{T}(t \mid \Delta) \propto \frac{e^{\beta\Delta} - 1}{\beta\Delta},$$

which clearly indicates this procedure *does* break up power-tail behavior [14].

## 10. CONCLUSIONS

The performance of tasks in jobs that restart after having failed or been interrupted is an important topic of research. It is of particular importance in the execution of parallel programs on computers, where PEs may fail, or in distributed applications where tasks can fail (or be interrupted) and must run at least as long as they did before they failed (i.e., the *Restart* policy). Tasks that can resume from the point where they were before the interruption have been described in the literature, but if a task must start over from the beginning, little is known regarding their analytic properties. We discussed that if the task time distribution has an exponential tail (which includes all ME, or RLT functions), then the completion time distribution is power-tailed. This can have serious consequences for the performance of the system if $\alpha$ is too small. In fact, when $\alpha \leq 1$, the time to completion has infinite mean, and when $\alpha \leq 2$ it has infinite variance. Thus, via our analytic and simulation results, we can say that the *Restart* recovery mechanism can be very unstable regarding the execution times of jobs.

We also discussed an analytic approach that generates performance and dependability measures for parallel jobs running on unreliable systems whose tasks times are non-exponentially distributed using the *Resume*, *Replace*, and *Restart* recovery polices, although other recovery policies could be considered. One well-known problem with these types of analytic approaches is their (potentially) large state-space requirements. This was addressed in two ways: 1) analyzing the system via epochs since the only storage requirement is the current epoch to generate performance and dependability measures; and, 2) using a reduced product space.

Finally, we discussed on-going work regarding power-tail behavior and tasks or jobs that checkpoint on unreliable systems. It was discussed that when a task or job checkpointing after a sub-tasks completes *does not* eliminate power-tail behavior when one of the sub-tasks is ME distributed. However, checkpointing at fixed intervals of time *does* eliminate power-tail behavior

## 11. REFERENCES

[1]. L. Lipsky. *Queueing Theory: A Linear Algebraic Approach*. McMillan. NY. 1992.

[2]. V. Kulkarni, V. Nicola, and K. Trivedi. On Modeling the Performance and Reliablility of Multimode Systems, *The Journal of Systems and Software* 20 (1986).

[3]. V. Kulkarni, V. Nicola, and K. Trivedi. The Completion Time of a Job on a Multimode System, *Advances in Applied Probability* 19 (1987).

[4]. A. Bobbio and K. Trivedi. Computation of the Distribution of the Completion Time When the Work Requirement is a PH Random Variable, *Communications in Statistics- Stochastic Models* 6 (1990).

[5]. M. Greiner, M. Jobmann, and L. Lipsky. The Importance of Power-Tail Distributions for Modeling Queueing Systems, *Operations Research* 47 (2) (1999).

[6]. P. Fiorini, L. Lipsky, and M.Crovella. Consequences of Ignoring Self-Similar Data Traffic In Communications Modeling. "*10th International Conference on Parallel and Distributed Computing (PDCS-97)*", New Orleans, USA 1997.

[7]. W.E. Leland and T. J. Ott. Load-balancing heuristics and process behavior, in "*SIGMETRICS Conf. Measurement & Modeling of Comput. Syst.*", 1986.

[8]. D. Feitelson. Sensitivity of Parallel Job Scheduling to Fat-Tailed Distributions. unpublished manuscript, School of Computer Science and Engineering, The Hebrew University of Jerusalem, 2000.

[9]. R. Sheahan, L. Lipsky, and P. Fiorini. The Effect of Different Failure recovery Procedures On the Distribution Of Task Completion Times," in *19th IEEE International Parallel and Distributed Processing Symposium*," Denver, CO 2005.

[10]. R. W. Rowan. Modeling Unreliable Parallel Systems with Non-Exponential Task Time Distributions, M.S. Thesis, University of Southern Maine, 2005.

[11]. P. Fiorini, R. Sheahan, and L. Lipsky. On Unreliable Computing Systems when Heavy-Tails Appear as a Result of the Recovery Procedure, *Performance Evaluation Review* 33 (2) (2005).

[12]. M. Greiner, M. Jobmann, and L. Lipsky. The Importance of Power-Tail Distributions for Modeling Queueing Systems. *Operations Research* 47 (2) (1999).

[13]. P. Fiorini and C. Bossie. On Checkpointing and Heavy-Tails in Unreliable Computing Environments. to appear in *Performance Evaluation Review* (2006).

[14]. P. Fiorini and L. Lipsky. Comparing Checkpointing Strategies in Unreliable Computing Environments. Unpublished manuscript, 2005.

***Pierre M. Fiorini*** *is an Assistant Professor of Computer Science at the University of Southern Maine. He received the Ph.D. degree from the University of Connecticut in Computer Science & Engineering (1998), an M.S. in Computer Science & Engineering from the University*

*of Connecticut (1995), and a B.S. in Computer Science from Trinity College (1989). His research interests include Queueing Theory, Computer Performance Modeling, Network Modeling, Stochastic Processes, and Artificial Intelligence. He is a member of the IEEE and ACM.*

***Robert W. Rowan** received his M.S. degree in Computer Science from the University of Southern Maine in 2004. After earning a B.S. in Mechanical Engineering from Cornell University in 1999, he went work for MicroStrategy Incorporated, a database analysis software vendor in Vienna, Virginia. He currently manages information systems for Nationwide Payment Solutions, a provider of electronic payment transaction processing and services.*