Optimal Job Fragmentation

[Extended Abstract]

Jayakrishnan Nair
Electrical Engineering
California Institute of Technology
ujk@caltech.edu

Steven H. Low
Computer Science and Electrical Engineering
California Institute of Technology
slow@caltech.edu

1. INTRODUCTION

It has been recently discovered that on an unreliable server, the job completion time distribution function (df) can be heavy-tailed (HT) even when job size df is light-tailed (LT) [1, 5]. [1] A key to this phenomenon is the RESTART feature where if a job is interrupted in the middle of its processing, the entire job needs to restart from the beginning, i.e., the work that is partially completed is lost.

A standard mechanism for reducing the job completion time in an unreliable service environment is checkpointing [3, 4, 6]. We view checkpointing as a job fragmentation operation, where the server processes one fragment of the job at a time. If the server becomes unavailable, say due to failure, then only the work corresponding to the fragment being processed at the time of failure is lost. In this paper, we are motivated by the question: Can fragmentation 'lighten' the tail df of the completion time? In Section 3, we provide sufficient conditions on the fragmentation policy that gives rise to LT completion time so long as the job size df is LT. We then characterize the optimal fragmentation policy seeking to minimize the expected job completion time. This policy requires a priori knowledge of the job size. We then describe a sub-optimal fragmentation policy that is blind to the job size and is provably very close to optimal. We also describe the asymptotic tail behavior of the job completion time df under both policies. Assuming the server unavailability periods are LT, both policies produce LT completion times when the job size df is LT. For the case of regularly varying job size df, the job completion time under both policies is regularly varying with the same degree - this is the lightest possible asymptotic tail behavior (in the degree sense). [2]

2. MODEL

2.1 Problem formulation

Our model is adapted from that of [1, 5]; we will elaborate in Section 2.2 below.

We are given a job of possibly random total size $L > 0$. We will fragment the job into smaller chunks and submit them successively to a server for processing. The server alternates between states of availability and unavailability. Without loss of generality, we assume the server has a unit service rate when available. Let n index the submission of job fragments, and $(x_n, n \geq 1)$ denote the size of the fragment for the nth submission. To each fragment is added a fixed fragmentation overhead ϕ so that the total size (processing time) of the nth submission is $x_n + \phi$. The nth job fragment is submitted immediately after the previous submission completes successfully, provided the server remains available. After the nth job is submitted, it either completes successfully after $x_n + \phi$ amount of time or fails after a random amount $A_n < x_n + \phi$ of time. The former happens with probability $\text{Prob}(A_n \geq x_n + \phi)$ and the latter with probability $\text{Prob}(A_n < x_n + \phi)$. As soon as the nth job fragment fails after A_n amount of time (server becomes unavailable in the middle of processing the nth job fragment), the system idles for U_n amount of time after which the server becomes available again. The $n+1$st job fragment with size $x_{n+1} + \phi$ is then immediately submitted, and the cycle repeats. Note that x_{n+1} may be equal to x_n (e.g., if the same job fragment is submitted again) or not (e.g., if a different fragmentation is chosen).

The time between the nth and $n+1$st submission is the cost at the nth stage: if the nth submission is successful, the cost is the job completion time $x_n + \phi$; otherwise, it is the time $A_n + U_n$ from the nth submission time to the beginning of the next availability period. We make the following assumptions.

1. The random variables $(A_n, n \geq 1)$ are iid and exponentially distributed with mean $1/\mu$.
2. The random variables $(U_n, n \geq 1)$ are iid.
3. L, $(A_n, n \geq 1)$ and $(U_n, n \geq 1)$ are mutually independent.

As we will see below, however, U_n only enters our model on the event $\{A_n < x_n + \phi\}$.

We are interested in understanding the impact of the fragmentation policy on the job completion time. To model the completion time precisely, let $x = (x_n, n \geq 1)$ denote the control (fragmentation) policy. Let the state be l_n, the remaining job size right before the nth submission, that evolves according to, for $n = 1, 2, \ldots$,

$$l_{n+1} \quad = \quad l_n - x_n \, \mathbf{1}\left(A_n \geq x_n + \phi\right). \tag{1}$$

Note that $l_1 = L$. Let the cost for the nth submission be τ_n, the time between the nth and $n+1$st submission under x,

[1] A non-negative random variable X is heavy-tailed if $\overline{\lim}_{t\to\infty} e^{\theta t} P(X > t) = \infty$ for all $\theta > 0$. Conversely, X is light-tailed if there exists $\theta > 0$ such that $\lim_{t\to\infty} e^{\theta t} P(X > t) = 0$.

[2] A df F is regularly varying with index/degree $\alpha > 0$ (denoted $F \in \mathcal{RV}(\alpha)$) if $\overline{F}(x) = x^{-\alpha}\chi(x)$ where $\chi(x)$ is a slowly varying function, i.e., $\chi(x)$ satisfies $\lim_{x\to\infty} \frac{\chi(xy)}{\chi(x)} = 1 \; \forall \; y > 0$ [2]. We will abuse notation and say a random variable $L \in \mathcal{RV}(\alpha)$ to mean that its df $F_L \in \mathcal{RV}(\alpha)$.

that is defined by

$$\tau_{n+1} = ((x_n + \phi)\mathbf{1}(A_n \geq x_n + \phi) + (A_n + U_n)\mathbf{1}(A_n < x_n + \phi)) \mathbf{1}(l_n > 0). \quad (2)$$

We emphasize that the state sequence $(l_n, n \geq 1)$ and the cost sequence $(\tau_n, n \geq 1)$ depend on the control policy $(x_n, n \geq 1)$, although this is not explicit in the notation. The job completion time T is given by

$$T = \sum_{n=1}^{\infty} \tau_n. \quad (3)$$

In Section 3, we present our results regarding the impact of the fragmentation policy on T.

2.2 Model interpretation and motivation

As mentioned above, our model is an adaptation of the following model in [1, 5] where a server alternates between states of availability and unavailability as per a semi-Markov process. This can model, for example, a server that is prone to failure: the unavailability period corresponding to the server downtime after a failure. The server availability (unavailability) periods are distributed as A (U) respectively. A job of random size L, independent of the server availability process is to be processed by the server. If the server becomes unavailable when the job is still being processed, we assume that the job needs to be restarted from the beginning, i.e., the work that is partially completed is lost. This is the RESTART model in queueing literature (see [1]). Recently, the following result has been proved about the job completion time under RESTART [5, 1].

THEOREM 1 ([5, 1]). *Under RESTART, if distribution of the job size L has unbounded support, then the job completion time is HT.*

This means completion times may be HT even for very LT workloads. Intuitively, the reason we get HT completion times is that large jobs get restarted many times before they complete and therefore have very large completion times.

Checkpointing is a standard mechanism for reducing the completion time of a job on an unreliable server [3, 4, 6]. It involves intermittently saving the state of the job to a reliable storage; the points at which the state is saved are called checkpoints or rollback points. If the job is interrupted by server unavailability, processing can resume from the last saved checkpoint once the server becomes available again.

Our model in Section 2.1 captures the processing of a job with checkpointing, assuming the server availability periods are exponentially distributed (with parameter μ). Submitting a fragment of size x_n to the server may be interpreted as a decision to establish the next checkpoint after time x_n of processing. ϕ may be interpreted as the time required to establish the checkpoint. The checkpoint is established successfully if the server remains available for a time $x_n + \phi$. When a fragment completes successfully (i.e., a checkpoint is established successfully), we allow immediate submission of the next fragment. Since we make the assumption that the server availability periods are exponentially distributed, the memoryless property of exponential distribution means that when a job fragment finishes and the next fragment is submitted, the remaining time to server unavailability is again exponentially distributed with the same parameter. The random variables $(U_n, n \geq 1)$ are distributed as the server unavailability period U. Note that we make no assumptions regarding the distribution of U. Once the server becomes available after a period of unavailability, there is typically a recovery period associated with reloading the saved job state before useful processing can resume. This recovery time can conveniently be incorporated into the unavailability period U. Note that the model can easily capture interruptions during the recovery process, by appropriate redefinition of the unavailability period df [6].

We now briefly describe some applications that motivate the model.

1. Servers that fail: The server could be unreliable and might fail from time to time. Job fragments are submitted successively as soon as the previous fragment completes successfully provided the server remains available. In this scenario, U_n denotes the down-time after the server failed in the middle of the nth submission.

2. Priority queue: Consider a queue that serves jobs of two priority levels. Low priority (LP) jobs use the server when there are no high priority (HP) jobs in the system. If an HP job arrives when the server is processing an LP job, the LP job is pre-empted and needs to be restarted. In this scenario, U_n denotes the busy period induced by HP jobs that arrived in the middle of the nth LP job. We are interested in the problem of how the LP jobs should be fragmented. The time to the start of the next HP busy period will be exponential if HP jobs arrive according to a Poisson process.

3. File fragmentation in cognitive radio setting: Consider a secondary user who is allowed to use a wireless channel to transfer her file of size L whenever primary (high priority) users are not using it. The secondary user must abort her transmission whenever primary users want to use the channel. We are interested in the problem of how the secondary user should size her packets. The availability period for the secondary user will be exponentially distributed if the primary users initiate transfers according to a Poisson process.

3. RESULTS

In this section, we present our results on the job fragmentation model. The following theorem gives sufficient conditions on the fragmentation policy for LT L to imply LT completion time.

THEOREM 2. *Assume that L, U are LT. If the fragmentation policy satisfies either of the two following conditions, then T is LT:*

1. **Independent fragments:** *$\{X_j\}_{j \geq 1}$ is an iid sequence of random variables independent of L and the server availability process satisfying $P(X_j > 0) > 0$ and $x_j = \min\{X_j, l_j\}$.*

2. **Bounded fragments:** *There exist constants a, b where $0 < a < b$ such that the fragmentation policy satisfies*

$$\min\{a, l_j\} \leq x_j \leq \min\{b, l_j\}.$$

Note that the condition that L and U be LT cannot be relaxed. It is easy to see that if either L or U are HT,

then T is necessarily HT, with or without fragmentation. The sufficient conditions of Theorem 2 admit constant fragment sizes (this corresponds to periodic checkpointing), i.e., $x_n = \min\{c, l_n\}$ for some positive constant c. Finally, we note that the condition that the fragment sizes be bounded is very reasonable. If we choose too small a fragment size, then the fragmentation overhead starts to dominate the processing time. If the fragment size is chosen too large, then its probability of failure approaches 1. Theorem 2 says that so long as we choose fragment sizes 'reasonably,' the completion time is LT if L, U are LT. In other words, (so long as U is LT) *HT completion time can occur only due to HT job sizes.*

We now describe the optimal fragmentation policy that minimizes the expected completion time. We optimize within the class of stationary Markov policies (i.e., policies where the control decision x_n in period n depends only on the state l_n and not on the discrete time index n or on past events). We need the following definitions:

1. $h(x) = (E[A] + E[U])\left(e^{\mu(x+\phi)} - 1\right)$,
2. $g(x) = \frac{h(x)}{x}$,
3. $a = \arg\min_{x>0} g(x)$.

THEOREM 3. **Optimal fragmentation policy x^*:** *The expected job completion time is minimized by fragmenting the job into $K^*(L)$ fragments of equal size $x^*(L) = \frac{L}{K^*(L)}$, where $K^*(L)$ is given by*

$$K^*(L) = \begin{cases} 1 & \text{for } L \le a \\ \underset{k \in \{\lfloor \frac{L}{a} \rfloor, \lceil \frac{L}{a} \rceil\}}{\arg\min} \; g(L/k) & \text{for } L > a \end{cases}.$$

Each fragment is (re)submitted to the server till it gets processed completely. The completion time T^* under the optimal policy has the following properties.

1. *If L, U are LT, then T^* is LT.*
2. *If U is LT, $L \in \mathcal{RV}(\alpha)$, then $P(T^* > t) \sim P\left(L > \frac{t}{g(a)}\right)$.*

We make the following observations regarding the optimal fragmentation policy x^*.

1. The optimal policy corresponds to checkpointing periodically, with a period $x^*(L)$ that depends on the job size.
2. The optimal fragmentation policy does *not* depend on the server unavailability period distribution U.
3. Assuming Poisson server failures, the problem of optimal checkpoint placement in order to minimize the expected job completion time is considered in [3]. However, the optimal policy is derived in [3] assuming that no server failures can occur during checkpointing and recovery. We make no such assumption in this paper.

Finally, it is possible to fragment very close to optimally by remaining blind to the job size. Consider the following simple fragmentation policy.

$$x_j = \min\{l_j, a\}$$

We denote this policy by a [3] and the completion time under this policy by T^a.

THEOREM 4. **Sub-optimal blind fragmentation policy a:** *The sub-optimality of policy a (in expected completion time) is bounded as follows.*

$$E[T^a | L = l] - E[T^* | L = l] \le h(a) \quad \forall \quad l > 0.$$

Moreover,

1. *If L, U are LT, then T^a is LT.*
2. *If U is LT, $L \in \mathcal{RV}(\alpha)$, then $P(T^a > t) \sim P\left(L > \frac{t}{g(a)}\right)$.*

We make the following observations regarding the sub-optimal blind policy a.

1. a corresponds to checkpointing periodically with period a.
2. The sub-optimality of a is bounded above by a constant for any job size.

We conclude with a remark regarding the tail behavior of the completion time under the fragmentation policies x^* and a. Let us assume U is LT. If L is LT, then the completion time is LT under both policies. Clearly, if L is HT, so is the completion time, with or without fragmentation. Theorems 4 and 3 indicate that if L is regularly varying, then the completion time under x^* and a is regularly varying with the same degree as L. Note that the policies x^* and a were derived seeking to minimize the expectation of the completion time. Even so, they produce as good a completion time tail behavior as possible (in the degree sense) for HT (specifically, regularly varying) L.

4. REFERENCES

[1] S. Asmussen, P. Fiorini, L. Lipsky, T. Rolski, and R. Sheahan. Asymptotic behavior of total times for jobs that must start over if a failure occurs. *Mathematics of Operations Research*, 33(4):932–944, 2008.

[2] N. H. Bingham, C. M. Goldie, and J. L. Teugels. *Regular Variation (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press, 1987.

[3] A. Duda. The effects of checkpointing on program execution time. *Inf. Process. Lett.*, 16(5):221–229, 1983.

[4] V. Grassi, L. Donatiello, and S. Tucci. On the optimal checkpointing of critical tasks and transaction-oriented systems. *Software Engineering, IEEE Transactions on*, 18(1):72–77, Jan 1992.

[5] P. R. Jelenković and J. Tan. Can retransmissions of superexponential documents cause subexponential delays? In *Proceedings of IEEE INFOCOM*, May 2007.

[6] V. G. Kulkarni, V. F. Nicola, and K. S. Trivedi. Effects of checkpointing and queueing on program performance. *Stochastic Models*, 6(4), 1990.

[3] We abuse notation and use the same symbol a to denote both the fragment size as well as the fragmentation policy.