PERFORMANCE OPTIMIZATION OF PARALLEL-DISTRIBUTED PROCESSING WITH CHECKPOINTING FOR CLOUD ENVIRONMENT

TSUGUHITO HIRAI AND HIROYUKI MASUYAMA
Graduate School of Informatics, Kyoto University
Yoshida-Hommachi, Sakyo-ku, Kyoto 606-8501, Japan

SHOJI KASAHARA
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

YUTAKA TAKAHASHI
Graduate School of Informatics, Kyoto University
Yoshida-Hommachi, Sakyo-ku, Kyoto 606-8501, Japan

(Communicated by Wuyi Yue)

Abstract. In cloud computing, the most successful application framework is parallel-distributed processing, in which an enormous task is split into a number of subtasks and those are processed independently on a cluster of machines referred to as workers. Due to its huge system scale, worker failures occur frequently in cloud environment and failed subtasks cause a large processing delay of the task. One of schemes to alleviate the impact of failures is checkpointing method, with which the progress of a subtask is recorded as checkpoint and the failed subtask is resumed by other worker from the latest checkpoint. This method can reduce the processing delay of the task. However, frequent checkpointing is system overhead and hence the checkpoint interval must be set properly. In this paper, we consider the optimal number of checkpoints which minimizes the task-processing time. We construct a stochastic model of parallel-distributed processing with checkpointing and approximately derive explicit expressions for the mean task-processing time and the optimal number of checkpoints. Numerical experiments reveal that the proposed approximations are sufficiently accurate on typical environment of cloud computing. Furthermore, the derived optimal number of checkpoints outperforms the result of previous study for minimizing the task-processing time on parallel-distributed processing.

1. Introduction. Cloud computing, a new usage of computing resources, has rapidly spread in this decade. According to [14], cloud computing is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared
pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. These beneficial characteristics enable us to use enormous computing resources conveniently with a low usage fee.

The most successful application framework in cloud computing is a parallel-distributed processing scheme, such as MapReduce [6]. This scheme handles an enormous amount of data with a huge number of computing resources in parallel-distributed processing fashion, and is used for a lot of applications such as data mining, machine learning and social network analysis [3]. In the following, we refer to this processing mechanism as large-scale parallel-distributed processing.

In large-scale parallel-distributed processing, an enormous task is split into a number of subtasks and those are processed independently in parallel on a cluster of machines referred to as workers. The task completes when all the subtasks have finished, and thus subtasks which are involved in worker failures cause a large processing delay of the task (the issue of stragglers) [6]. Worker failures occur frequently in large-scale parallel-distributed processing because of its huge system scale [8]. For example, it is reported in [7] that there are thousands of hard drive failures for a new cluster in the first year. In addition, the use of commodity machines for reducing hardware cost decreases the mean time between consecutive failures in data centers [1, 7]. Therefore, dealing with worker failures is an important issue. In the following, we refer to the time to complete a task (resp. subtask) as the task-processing (resp. subtask-processing) time.

Checkpointing [13] is one of the well-known solutions for alleviating the impact of worker failures. In this method, the progress of the processing (checkpoint) is periodically saved during a period of the processing of a subtask. When worker failure occurs, the subtask processed by the failed worker is resumed by other worker from the latest checkpoint. Checkpointing prevents the failed subtask executing from the beginning of the processing and reduces the increase in the subtask-processing time. However, excessive creation of checkpoints increases system overhead, whereas a long checkpoint interval wastes computation resources when worker failure occurs. Therefore, deriving the optimal checkpoint interval is a crucial subject.

In this paper, we evaluate the effect of checkpointing method on the task-processing time and consider the optimal number of checkpoints which achieves the shortest task-processing time. We construct a stochastic model for large-scale parallel-distributed processing with checkpointing, in which a task accepted by the service facility is split into subtasks of equal size, and the processing of the task ends when all of the subtasks are completed. Note that the assumption of equally sized subtasks becomes reasonable when a huge amount of input data is split into data pieces of approximately equal size [6]. Moreover, each subtask’s checkpoints are made periodically during its processing, and a subtask is resumed by other worker from the latest checkpoint when the dedicated worker fails. We assume that the time intervals between consecutive failures of each worker follow an exponential distribution. This distribution is one of the most commonly used distributions due to its tractability and there are various discussions on its validity [2, 9, 11]. We discuss the validity of this assumption in numerical examples.

For this system, we approximately derive explicit expressions for the mean task-processing time and the optimal number of checkpoints. In numerical examples, we investigate the accuracy of the derived approximations in comparison with the results of Monte Carlo simulation. Moreover, we show the usefulness of the derived
optimal number of checkpoints by comparing with the result of previous study. Finally, we validate through Monte Carlo simulation experiments the assumption of an exponential distribution for the time intervals between consecutive worker failures.

The rest of this paper is organized as follows. Section 2 reviews previous studies on checkpointing. In Section 3, we illustrate the analytical model for large-scale parallel-distributed processing with checkpointing. For this model, we derive the mean task-processing time and the optimal number of checkpoints in Section 4. In Section 5, we show numerical examples of the derived measures. Finally, Section 6 concludes the paper.

2. Related work. Large-scale parallel-distributed processing is required to minimize the performance degradation due to worker failures. In [4, 13, 17], various implementations of large-scale parallel-distributed processing are proposed, and they apply checkpointing method to reduce the processing delay of a task caused by worker failures. However, none of these studies fully discuss the optimal checkpoint interval.

From the viewpoint of mathematical modeling, Young [16] proposes the model for single processing with checkpointing, and derives the optimal checkpoint interval with a first order approximation. His result is applied to the determination of checkpoint interval not only for single processing but also for parallel-distributed processing [2, 15]. Daly [5] extends Young’s result to a higher order approximation, and shows that the optimal checkpoint interval is not affected by the time required to resume a failed task. The results of Young and Daly are simple and insightful. However, it becomes difficult to apply their results to parallel-distributed processing as the system scale grows because their models do not take into account the issue of stragglers.

Fialho et al. [10] and Jin et al. [12] focus on checkpointing method for one of the parallel-distributed processing schemes, Message Passing Interface (MPI), and derive the optimal checkpoint interval. However, in MPI, failed subtask restarts or stops the processing of other subtasks because subtasks are dependent each other by data exchange. The results of the above two studies are assumed this dependency. On the other hand, in MapReduce, data exchange between subtasks is not required during Map and Reduce steps, and failed subtask does not bother the processing of other subtasks, but becomes a straggler. To the best of our knowledge, the effect of checkpointing method for parallel-distributed processing such as MapReduce has not been fully studied yet. In this paper, therefore, we focus on parallel-distributed processing in which subtasks are processed independently, and derive the optimal number of checkpoints. The derived optimal number of checkpoints is as simple as Young’s formula [16], while our formula takes into account the issue of stragglers.

3. Analytical model. In this section, we describe the analytical model consisting of two parts: large-scale parallel-distributed processing and the processing of a subtask with checkpointing.

3.1. Model descriptions of large-scale parallel-distributed processing. We describe a stochastic model for large-scale parallel-distributed processing. When a task is accepted by the server, the task is divided into $M$ ($M = 1, 2, \ldots$) subtasks, and each subtask is assigned to a dedicated worker. Let $S_i$ ($i = 1, 2, \ldots, M$) denote the subtask-processing time of the $i$-th subtask. The subtask-processing times
\{S_i; i = 1, 2, \ldots, M\} are independent and identically distributed (i.i.d.) random variables which follow a common distribution function \(F\), i.e.,
\[ P(S_i \leq t) = F(t), \quad t \in \mathbb{R}. \]

We refer to the distribution \(F\) as the subtask-processing time distribution.

We now define \(T\) as the task-processing time, which is equal to the maximum of \(M\) subtask-processing times, i.e.,
\[ T = \max_{1 \leq i \leq M} S_i. \]

We also define \(G\) and \(g^{(1)}\) as the distribution function and mean value of the task-processing time \(T\). It then follows that
\[ G(t) = P(T \leq t) = \{F(t)\}^M, \quad t \in \mathbb{R}, \]
\[ g^{(1)} = E[T] = \int_0^\infty t dG(t) = \int_0^\infty t d\{F(t)\}^M. \tag{1} \]

In what follows, we refer to \(g^{(1)}\) as the mean task-processing time.

### 3.2. Model descriptions of the processing of a subtask with checkpointing

We introduce some symbols and assumptions to describe the processing of a subtask with checkpointing (see Fig. 1). That is, the subtask-processing time \(S_i (i = 1, 2, \ldots, M)\) follows these assumptions.

(a) We define \(b\) \((b > 0)\) as the subtask-processing time without checkpointing and worker failures, i.e., the actual processing time of a subtask. We also define \(c\) \((c > 0)\) as the time required to make one checkpoint. Moreover, let \(K\) \((K = 0, 1, \ldots)\) denote the number of checkpoints being made during a period of the processing of a subtask.

(b) The processing work of a subtask is split into \(K+1\) segments of equal size, and a checkpoint is created at the end of each segment except the last one. Let \(\sigma_k (k = 1, 2, \ldots, K+1)\) denote the \(k\)-th time interval including the processing time of the \(k\)-th segment and the time of creating a checkpoint (if any). We then have
\[ \sigma_k = \begin{cases} b/(K+1) + c =: \sigma, & k = 1, 2, \ldots, K, \\ b/(K+1) =: \sigma', & k = K + 1. \end{cases} \tag{2} \]

We refer to \(\sigma_k\) as the \(k\)-th segment-processing time. Let \(\tau\) denote the subtask-processing time with \(K\) checkpoints and no worker failures. We then have
\[ \tau = \sum_{k=1}^{K+1} \sigma_k = K\sigma + \sigma' = b + Kc. \tag{3} \]

(c) The time intervals between consecutive failures for each worker are i.i.d. with an exponential distribution having mean \(f\) \((f > 0)\). When worker failure occurs during a segment-processing time, the subtask involved in failure is resumed by other worker from the checkpoint recorded just before the segment-processing time. We here define \(r\) \((r \geq 0)\) as the time required to resume the failed subtask.

(d) The magnitudes of the parameters \(b, c, K, f\) and \(r\) satisfy
\[ Kn, r \ll b \ll f, \tag{4} \]
which is a typical relation for parallel-distributed processing with checkpointing on large-scale computing resources \([2, 7]\).
We note that from these assumptions, the subtask-processing time is given as the sum of \( b, Kc \), the total computation time lost by worker failures and the total time required to resume the failed subtask.

4. Analysis. In this section, we first consider the worker failure probability and the processing delay (defined in Subsections 4.1 and 4.2, respectively). We then derive the subtask-processing time distribution \( F \) and the mean task-processing time \( g^{(1)} \). Finally, we propose the optimal number of checkpoints which minimizes \( g^{(1)} \). In the analysis, in order to express the optimal number of checkpoints as a simple formula, we apply some approximations.

4.1. Worker failure probability. For simplicity, we refer to an arbitrarily chosen subtask as a tagged subtask. Let \( p \) denote the probability that the tagged subtask is involved in worker failures during its processing, i.e., at least one worker failure happens in the processing of the tagged subtask, which is referred to the worker failure probability. We assume that the processing of the tagged subtask begins at time 0. We then define \( T^{(j)}_{\text{fail}} \) \( (j = 1, 2, \ldots) \) as the time at which the \( j \)-th worker failure happens to the tagged subtask. It follows from assumption (c) that the time intervals between consecutive worker failures, \( T^{(1)}_{\text{fail}}, T^{(2)}_{\text{fail}} - T^{(1)}_{\text{fail}}, T^{(3)}_{\text{fail}} - T^{(2)}_{\text{fail}}, \ldots \), are i.i.d. with an exponential distribution having mean \( f \). We also define \( N_{\text{fail}} \) as the number of worker failures happening to the tagged subtask during its processing. From these definitions, we have

\[
p = P\left(N_{\text{fail}} \geq 1\right) = P\left(T^{(1)}_{\text{fail}} \leq \tau\right) = 1 - \exp\left\{-\frac{\tau}{f}\right\} = 1 - \exp\left\{-\frac{b + Kc}{f}\right\}, \tag{5}
\]

where the last equality is due to (3). Note here that (4) implies \( b + Kc \ll f \) and thus \( (b + Kc)/f \ll 1 \). Therefore, using the linear approximation, the worker failure probability \( p \) in (5) can be estimated in the following way.

\[
p \approx \frac{b + Kc}{f} \ll 1. \tag{6}
\]

Next, we estimate the probability \( P(N_{\text{fail}} \geq 2) \).

\[
P\left(N_{\text{fail}} \geq 2\right) = \sum_{k=1}^{K+1} P\left(T^{(1)}_{\text{fail}} \leq \tau_k, N_{\text{fail}} \geq 2\right)
= \sum_{k=1}^{K+1} P\left(T^{(1)}_{\text{fail}} \leq \tau_k\right) P\left(N_{\text{fail}} \geq 2 \mid T^{(1)}_{\text{fail}} \leq \tau_k\right). \tag{7}
\]
where
\[ \tau_k = \begin{cases} k\sigma, & k = 0, 1, \ldots, K, \\ K\sigma + \sigma' = \tau, & k = K + 1. \end{cases} \] (8)

Recall here (see assumption (c)) that \( T_{\text{fail}}^{(1)} \) and \( T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} \) are i.i.d. with an exponential distribution having mean \( f \), and the event \( \{N_{\text{fail}} \geq 2\} \) is equivalent to the event \( \{T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} < r + \tau - \tau_{k-1}\} \), given that \( \{\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\} \). Based on these facts, we obtain
\[
P\left( N_{\text{fail}} \geq 2 \bigg| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \right) = P\left( T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} < r + \tau - \tau_{k-1} \bigg| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \right)
= 1 - \exp\left\{-\frac{r + \tau}{f}\right\}
= 1 - \exp\left\{-\frac{r + b + Kc}{f}\right\} =: p',
\] (9)

where we use (3) in the last but one equality. Substituting (5) and (9) into (7) yields
\[
P\left( N_{\text{fail}} \geq 2 \right) \leq p' \sum_{k=1}^{K+1} P\left( \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \right) = p' P\left( T_{\text{fail}}^{(1)} < \tau \right) = p'p. \tag{10}
\]

In addition, proceeding as in the derivation of (6), we can readily obtain
\[ p' \approx \frac{r + b + Kc}{f} \ll 1. \tag{11} \]

Applying (6) and (11) to (10) leads to
\[ P\left( N_{\text{fail}} \geq 2 \right) \ll p \ll 1, \]

and thus
\[ P\left( N_{\text{fail}} = 0 \right) = 1 - p, \quad P\left( N_{\text{fail}} = 1 \right) \approx p. \tag{12} \]

In the rest of this section, we assume that the probability \( P(N_{\text{fail}} \geq 2) \) is negligible.

4.2. Processing delay. We consider the processing delay of the tagged subtask caused by worker failure, under the assumption that \( N_{\text{fail}} = 1 \). The processing delay is defined as the sum of the time required to resume a failed subtask and the computation time lost by worker failure, i.e., the time between completion of creating the latest checkpoint and worker failure. Given that \( \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \) \( (k = 1, 2, \ldots, K + 1) \) and \( N_{\text{fail}} = 1 \), the processing delay is equal to \( r + T_{\text{fail}}^{(1)} - \tau_{k-1} \). It then follows from assumption (c) that
\[ P\left( r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t \bigg| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1 \right) \]
From (12), we have

\[ \text{distribution} \]

Applying (15) to (13) and (14), we have, for using the linear approximation, we have

\[ c \]

Note here that (2) and (4) imply

\[ \text{where the last equality is due to } \tau_k = \tau_{k-1} + \sigma \text{ (see (8)). Similarly, for } r \leq t < r + \sigma', \]

\[ \mathbb{P} \left( r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t \left| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau, N_{\text{fail}} = 1 \right. \right) = \frac{1 - \exp\left(-\frac{(t-r)/f}{\sigma'}\right)}{1 - \exp\left(-\sigma/f\right)}. \]  

(14)

Note here that (2) and (4) imply \( c \ll b/(K+1) \) and \( \sigma = b/(K+1) + c \ll f \), and using the linear approximation, we have

\[ \sigma' = \frac{b}{K+1} \approx \frac{b}{K+1} + c = \sigma, \quad 1 - \exp\left\{-\frac{-t}{f}\right\} \approx \frac{t}{f}, \quad 0 \leq t < \sigma. \]  

(15)

Applying (15) to (13) and (14), we have, for \( k = 1, 2, \ldots, K \) and \( r \leq t < r + \sigma \),

\[ \mathbb{P} \left( r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t \left| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau, N_{\text{fail}} = 1 \right. \right) \approx \frac{t-r}{\sigma}. \]  

(16)

4.3. Mean task-processing time. We first consider the subtask-processing time distribution \( F \). Let \( S \) denote the subtask-processing time of the tagged subtask. From (12), we have

\[ F(t) = \mathbb{P}(S \leq t) \]

\[ = \mathbb{P}(N_{\text{fail}} = 0)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) + \mathbb{P}(N_{\text{fail}} = 1)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 1) \]

\[ + \mathbb{P}(S \leq t, N_{\text{fail}} \geq 2) \approx (1 - p)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) + p\mathbb{P}(S \leq t \mid N_{\text{fail}} = 1) \]

\[ = (1 - p)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) \]

\[ + p \sum_{k=1}^{K+1} \mathbb{P} \left( S \leq t \left| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau, N_{\text{fail}} = 1 \right. \right) \]

\[ \times \mathbb{P} \left( \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau \left| N_{\text{fail}} = 1 \right. \right). \]  

(17)

Note here that

\[ \mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) = \left\{ \begin{array}{ll} 0, & 0 \leq t < \tau, \\ 1, & t \geq \tau. \end{array} \right. \]  

(18)

Note also that \( S = \tau + r + T_{\text{fail}}^{(1)} - \tau_{k-1} \) for \( k = 1, 2, \ldots, K+1 \), given that \( \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau \) and \( N_{\text{fail}} = 1 \). Combining this fact and (16), we have

\[ \mathbb{P} \left( S \leq t \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau, N_{\text{fail}} = 1 \right) \]

\[ = \mathbb{P} \left( r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t - \tau \left| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau, N_{\text{fail}} = 1 \right. \right) \]

\[ \approx \frac{t - \tau - r}{\sigma}, \quad k = 1, 2, \ldots, K + 1, \quad \tau + r \leq t < \tau + r + \sigma, \]
and thus
\[
\sum_{k=1}^{K+1} P \left( S \leq t \left| \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1 \right. \right) P \left( \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \left| N_{\text{fail}} = 1 \right. \right) \\
\approx \frac{t - \tau - r}{\sigma} \sum_{k=1}^{K+1} P \left( \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \left| N_{\text{fail}} = 1 \right. \right) \\
= \frac{t - \tau - r}{\sigma} \left( T_{\text{fail}}^{(1)} < \tau \left| N_{\text{fail}} = 1 \right. \right) \\
= \frac{t - \tau - r}{\sigma}, \quad \tau + r \leq t < \tau + r + \sigma. \tag{19}
\]

Applying (18) and (19) to (17) yields
\[
F(t) \approx \begin{cases} 
(1 - p), & \tau \leq t < \tau + r, \\
(1 - p) + p(t - \tau - r)/\sigma, & \tau + r \leq t < \tau + r + \sigma.
\end{cases} \tag{20}
\]

Next, we derive an approximation of the mean task-processing time \( g^{(1)} \). Substituting (20) into (1) and using the binomial theorem, we have
\[
g^{(1)} \approx \tau + r \left\{ 1 - (1 - p)^{M} \right\} + Mp\sigma \sum_{i=0}^{M-1} \binom{M-1}{i} \frac{p^i(1-p)^{M-1-i}}{i+2} . \tag{21}
\]

Note here that on the right side of (21), the second and third terms mean the expected values of the time required to resume a failed subtask and the computation time lost by worker failure, respectively. We now define \( R(z) \) as
\[
R(z) = \sum_{i=0}^{M-1} \binom{M-1}{i} \frac{p^i(1-p)^{M-1-i}z^{i+1}}{i+2} = z(1-p+pz)^{M-1}.
\]

It then follows from (21) that
\[
g^{(1)} \approx \tau + r \left\{ 1 - (1 - p)^{M} \right\} + Mp\sigma \int_{0}^{1} R(z)dz \\
= \tau + r \left\{ 1 - (1 - p)^{M} \right\} + \sigma \left\{ 1 - \frac{1 - (1 - p)^{M+1}}{(M+1)p} \right\}.
\]

Substituting the definitions (2), (3) and (5) of \( \sigma, \tau \) and \( p \), we have
\[
g^{(1)} \approx b + Kc + r \left\{ 1 - \left( \exp \left\{ - \frac{b + Kc}{f} \right\} \right)^{M} \right\} \\
+ \left( \frac{b}{K +1 } + c \right) \left\{ 1 - \frac{1 - \left( \exp \left\{ -(b + Kc)/f \right\} \right)^{M+1}}{(M+1)(1 - \exp \left\{ -(b + Kc)/f \right\})} \right\} \\
=: \tilde{g}^{(1)}(K). \tag{22}
\]

The accuracy of the approximation (22) is numerically investigated in Subsection 5.1.

4.4. Optimal number of checkpoints. In this subsection, we show a simple and explicit formula for the optimal number of checkpoints which is expected to minimize the mean task-processing time \( g^{(1)} \). To this end, we simplify the approximate
expression (22) of $g^{(1)}$. From (4), we have $(b + Kc)/f \ll 1$ and $Kc \ll b$, which lead to
\[
\exp\left\{ -\frac{b + Kc}{f} \right\} \approx 1 - \frac{b + Kc}{f} \approx 1 - \frac{b}{f}.
\]
Applying the above approximation to (22), we have
\[
g^{(1)} \approx \tilde{g}^{(1)}(K) \approx b + Kc + r \left\{ 1 - \left( 1 - \frac{b}{f} \right)^M \right\} + \left( \frac{b}{K + 1} + c \right) \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f} \right\} =: \tilde{g}^{(1)}(K).
\]
It is easy to see that
\[
\frac{d}{dK} \tilde{g}^{(1)}(K) = c - \frac{b}{(K + 1)^2} \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f} \right\},
\]
\[
\frac{d^2}{dK^2} \tilde{g}^{(1)}(K) = \frac{2b}{(K + 1)^3} \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f} \right\} \geq 0.
\]
Therefore, $\tilde{g}^{(1)}(K)$ takes its minimum value for $K$ such that
\[
\frac{d}{dK} \tilde{g}^{(1)}(K) = 0.
\]
We now define $K^{\ast}_{\text{prop}}$ as the number such that $(d/dK)\tilde{g}^{(1)}(K)|_{K=K^{\ast}_{\text{prop}}} = 0$, i.e., the solution to (23). We then have
\[
K^{\ast}_{\text{prop}} = \sqrt{\frac{b}{c} \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f} \right\}} - 1.
\]
We propose $K^{\ast}_{\text{prop}}$ as the optimal number of checkpoints. Since we have made several approximations in obtaining $K^{\ast}_{\text{prop}}$, the proposed number $K^{\ast}_{\text{prop}}$ is not, in general, equal to the exact optimal number of checkpoints, denoted by $K^{\ast}$, which minimizes $g^{(1)}$. Nevertheless, under those approximations, we can expect $K^{\ast}_{\text{prop}}$ to be a good approximation to the exact optimal number $K^{\ast}$, i.e.,
\[
K^{\ast} \approx K^{\ast}_{\text{prop}} = \sqrt{\frac{b}{c} \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f} \right\}} - 1.
\]
In Subsection 5.2, we investigate the accuracy of the approximate formula (24) through numerical experiments, where $K^{\ast}_{\text{prop}}$ is rounded to the nearest integer because the number of checkpoints must be integer.

Finally, we note that the approximation formula (24) is independent of $r$. It is reported by Daly [5] that for single processing, the optimal number of checkpoints is not affected by the time required to resume a failed subtask. Therefore, the formula (24) implies that this insensitivity against the time required to resume a failed subtask is also applicable for parallel-distributed processing. Moreover, we also note that if $M = 1$ then
\[
K^{\ast}_{\text{prop}} = \frac{b}{2\sqrt{cf}} - 1 =: K^{\ast}_{\text{young}},
\]
where $K^{\ast}_{\text{young}}$ is the optimal number of checkpoints according to Young [16]. Therefore, our result can be considered an generalization of Young’s result.
Table 1. Parameter set.

| Parameter | Description                  | Value              |
|-----------|------------------------------|--------------------|
| M         | Number of workers            | 10 to 1,000        |
| b         | Subtask-processing time      | 6 to 120 [hour]    |
| c         | Time to make a checkpoint    | 30 to 3,000 [sec]  |
| f         | Mean time between worker failures | 7 to 180 [day]   |
| r         | Time to resume a failed subtask | 30 to 3,000 [sec] |
| K         | Number of checkpoints        | 0 to 30            |

5. Numerical examples. In this section, we show some numerical examples. First, we verify the proposed approximations of the mean task-processing time and the optimal number of checkpoints in comparison with the results of Monte Carlo simulation. In addition, we show the usefulness of the derived optimal number of checkpoints by comparing with the result of previous study. Finally, we validate through Monte Carlo simulation experiments the assumption of an exponential distribution for the time intervals between consecutive worker failures. Table 1 shows the parameter values used in the numerical experiments. We set these parameters according to the literature [2, 7] in order to reflect values which are used in real systems.

5.1. Approximation accuracy of the mean task-processing time. In this subsection, we investigate the approximation accuracy of the mean task-processing time. We apply some approximations to derive the mean task-processing time in Section 4. To verify these approximations, we compare the analytical result (22) with the result of Monte Carlo simulation. We calculate the 95% confidence interval of the mean task-processing time in simulation experiments.

Figures 2 to 6 show the mean task-processing time with respect to the number of checkpoints for various system parameters. In Fig. 2 (resp. Figs. 3, 4, 5 and 6), the parameter $M$ (resp. $b$, $c$, $f$ and $r$) is set to different three values. In these figures, the analytical result gives a good approximation of the mean task-processing time for most parameter values. On the other hand, we observe the gaps between the result of analysis and that of simulation experiment for $M = 1,000$ in Fig. 2, $b = 120$ [hour] in Fig. 3 and $f = 7$ [day] in Fig. 5. These gaps are caused by the ignorance of twice or more worker failures happening to each subtask on analysis in Subsection 4.1. However, for such parameter values, qualitative trend against the number of checkpoints is described well by the approximation.

Moreover, in Fig. 6, the mean task-processing time takes the minimum value at $K = 13$ for each $r$. This trend means that the optimal number of checkpoints is not affected by the time required to resume a failed subtask, and agrees with the observation of the approximation formula (24).

These results imply that the derived approximation (22) is sufficiently accurate to find the number of checkpoints which minimizes the mean task-processing time when the system parameters do not significantly deviate from the approximation that twice or more worker failures on each subtask can be ignored.

5.2. Usefulness of the proposed approximation for the optimal number of checkpoints. In this subsection, we discuss the usefulness of the proposed approximation for the optimal number of checkpoints. To this end, we compare the mean task-processing times calculated by three different ways: the approximation
The mean task-processing time [sec]
The number of checkpoints

Figure 2. Mean task-processing time with respect to the number of checkpoints for various $M$ ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

Figure 3. Mean task-processing time with respect to the number of checkpoints for various $b$ ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

Based on Young [16], the proposed approximation (24) and Monte Carlo simulation. In simulation experiments, the mean task-processing time is calculated with the 95% confidence interval for each number of checkpoints, and the minimum one is chosen as the result of simulation. Note here that the approximation derived by
Figure 4. Mean task-processing time with respect to the number of checkpoints for various $c$ ($M = 100, b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

Figure 5. Mean task-processing time with respect to the number of checkpoints for various $f$ ($M = 100, b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

Young [16] is extended to a higher order one by Daly [5]. However, it is reported in [5] that there is no significant difference between the results of Young and Daly when $b \ll f$ holds. Moreover, as mentioned in Section 2, Young’s approximation is still
applied to the determination of the number of checkpoints for parallel-distributed processing [2, 15]. Therefore, we choose the result of Young as a counterpart.

Figures 7 to 11 illustrate the mean task-processing times calculated by three different ways. The horizontal axis represents the parameter \( M \) (resp. \( b, c, f \) and \( r \)) in Fig. 7 (resp. Figs. 8, 9, 10 and 11). Figures 7 to 11 indicate that the result of the proposed approximation agrees well with that of simulation, and outperforms the result of Young. This is because Young’s approximation does not take into account the issue of stragglers. We can confirm in Figs. 7 and 9 that this disadvantage appears remarkably for large \( M \) and \( c \). These results suggest that the proposed approximation (24) for the optimal number of checkpoints is significantly useful to minimize the mean task-processing time.

5.3. Assumption validation of the analytical model. In this subsection, we validate the assumption of the proposed analytical model. We assume in our analytical model that the time intervals between consecutive worker failures follow an exponential distribution to simplify the analysis. However, there are various discussions on its validity, and some studies point out that an exponential distribution is poorly fitted to the distribution for the time intervals between consecutive worker failures [2, 9, 11]. Therefore, we calculate the mean task-processing time through Monte Carlo simulation when the time intervals between consecutive worker failures follow an exponential distribution, gamma distribution with shape parameter 0.35 or Weibull distribution with shape parameter 0.48. We choose the latter two distributions according to the measurement of real systems [11]. The mean task-processing time is calculated with the 95% confidence interval for each number of checkpoints, and the minimum one is chosen as the result for each distribution.
Figures 7 to 8 show the mean task-processing time when the time intervals between consecutive worker failures follow the exponential, gamma or Weibull distribution. The horizontal axis represents the parameter $M$ (resp. $b$, $c$, $f$ and $r$)
Figure 9. Mean task-processing time with respect to $c$ for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.

Figure 10. Mean task-processing time with respect to $f$ for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.

in Fig. 12 (resp. Figs. 13, 14, 15 and 16). In these figures, there is no significant difference among the mean task-processing time for three distributions, and we can
The mean task-processing time [sec] vs. The time to resume a failed subtask [sec]

**Figure 11.** Mean task-processing time with respect to $r$ for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $b = 24$ [hour], $f = 30$ [day]): Comparison between the results of previous and proposal analyses and simulation.

The mean task-processing time [sec] vs. The number of workers

**Figure 12.** Mean task-processing time with respect to $M$ for the optimal number of checkpoints ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

Consider that the assumption that the time intervals between consecutive worker failures follow an exponential distribution is reasonable.
To consider this insensitivity against the shape of the distribution, we illustrate the mean task-processing time with respect to small $f$ in the cases of three distributions in Fig. 17. In this figure, the difference between the result of the exponential
The mean task-processing time [sec]
The mean time between worker failures [sec]
Exponential
Gamma
Weibull

Figure 15. Mean task-processing time with respect to $f$ for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

The mean task-processing time [sec]
The time to resume a failed subtask [sec]
Exponential
Gamma
Weibull

Figure 16. Mean task-processing time with respect to $r$ for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $b = 24$ [hour], $f = 30$ [day]): Comparison among three distributions for the time intervals between consecutive worker failures.

distribution and that of the gamma or Weibull distribution decreases with the increase in the mean time between worker failures. This implies that the shape of the distribution does not significantly affect the mean task-processing time when the
The mean task-processing time [sec]  

The mean time between worker failures [sec]  

Exponential  
Gamma  
Weibull

The mean time between worker failures is sufficiently larger than the subtask processing time (i.e. $b \ll f$).

These results indicate that the proposed analytical model can evaluate the mean task-processing time for parallel-distributed processing with checkpointing method even when the time intervals between consecutive worker failures follow a more realistic distribution.

6. Conclusion. In this paper, we evaluated the effect of checkpointing method on the task-processing time and considered the optimal number of checkpoints which achieves the shortest task-processing time. We constructed an analytical model for large-scale parallel-distributed processing with checkpointing, and approximately derived explicit expressions for the mean task-processing time and the optimal number of checkpoints. In numerical examples, we confirmed the accuracy of the derived approximations in comparison with the results of Monte Carlo simulation. Moreover, we showed the usefulness of the derived optimal number of checkpoints by comparing with the result of previous study. Finally, we validated through Monte Carlo simulation experiments the assumption of the analytical model that the time intervals between consecutive worker failures follow an exponential distribution. We can claim that the proposed approximations are sufficiently accurate even when the time intervals between consecutive worker failures follow a more realistic distribution, such as a gamma or Weibull distribution. Furthermore, the derived optimal number of checkpoints is as simple as Young’s formula [16], while our formula outperforms Young’s one and is useful for minimizing the task-processing time on parallel-distributed processing.
REFERENCES

[1] L. A. Barroso and U. Hökel, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, Morgan & Claypool Publishers, California, 2009.

[2] T. C. Bressoud and M. A. Kozuch, Cluster fault-tolerance: An experimental evaluation of checkpointing and MapReduce through simulation, in *Proc. the IEEE International Conference on Cluster Computing and Workshops (CLUSTER 2009)*, (2009).

[3] C. L. P. Chen and C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, *Information Sciences*, 275 (2014), 314–347.

[4] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy and R. Sears, MapReduce online, in *Proc. the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2010)*, (2010).

[5] J. T. Daly, A higher order estimate of the optimum checkpoint interval for restart dumps, *Future Generation Computer Systems*, 22 (2006), 303–312.

[6] J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters, *Communications of the ACM*, 51 (2008), 107–113.

[7] J. Dean, Designs, lessons and advice from building large distributed systems, in *Keynote Presentation of the 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS 2009)*, (2009).

[8] J. Dean and S. Ghemawat, MapReduce: A flexible data processing tool, *Communications of the ACM*, 53 (2010), 72–77.

[9] S. Di, Y. Robert, F. Vivien, D. Kondo, C.-L. Wang and F. Cappello, Optimization of cloud task processing with checkpoint-restart mechanism, in *Proc. the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 13)*, (2013).

[10] L. Fialho, D. Rexachs and E. Luque, What is missing in current checkpoint interval models?, in *Proc. the 31st International Conference on Distributed Computing Systems (ICDCS 2011)*, (2011), 322–332.

[11] B. Javadi, D. Kondo, A. Iosup and D. Epema, The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems, *Journal of Parallel and Distributed Computing*, 73 (2013), 1208–1223.

[12] H. Jin, Y. Chen, H. Zhu and X.-H. Sun, Optimizing HPC fault-tolerant environment: An analytical approach, in *Proc. the 39th International Conference on Parallel Processing (ICPP 2010)*, (2010), 525–534.

[13] A. Martin, T. Knauth, S. Creutz, D. Becker, S. Weigert, C. Fetzer and A. Brito, Low-overhead fault tolerance for high-throughput data processing systems, in *Proc. the 31st International Conference on Distributed Computing Systems (ICDCS 2011)*, (2011), 689–699.

[14] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-145, 2011.

[15] M. Tafzi, J. Y. Shi and A. Khreishah, SpotMPI: A framework for auction-based HPC computing using Amazon spot instances, in *Proc. the 11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2011)*, (2011), 109–120.

[16] J. W. Young, A first order approximation to the optimum checkpoint interval, *Communications of the ACM*, 17 (1974), 530–531.

[17] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker and I. Stoica, Discretized streams: Fault-tolerant streaming computation at scale, in *Proc. the 24th ACM Symposium on Operating Systems Principles (SOSP 2013)*, (2013), 423–438.

Received December 2016; revised August 2017.

E-mail address: tsugu@sys.i.kyoto-u.ac.jp
E-mail address: masuyama@sys.i.kyoto-u.ac.jp
E-mail address: kasahara@is.naist.jp
E-mail address: takahashi@i.kyoto-u.ac.jp