Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Optimizing checkpoint for scientific simulations*

Xi-sheng XIAO[1,2], Ying-ping HUANG[‡3], Xi-hui ZHANG[3]
([1]Economics & Management College, Southwest Jiaotong University, Chengdu 610031, China)
([2]Industrial and Commercial College, Guizhou University of Finance and Economics, Guiyang 550003, China)
([3]College of Business, University of North Alabama, Florence, AL 35632, USA)
E-mail: davidshiau@qq.com; yhuang@una.edu; xzhang6@una.edu

Received May 12, 2012; Revision accepted Sept. 3, 2012; Crosschecked Nov. 12, 2012

Abstract: It is extremely time-consuming to restart a long-running simulation from the beginning when a failure occurs. Checkpointing is a viable solution that enables simulations to be resumed from the point of failure. We study three models to determine the optimal checkpoint interval between contiguous checkpoints so that the total execution time is minimized and we demonstrate that optimal checkpointing can facilitate self-optimizing. This study greatly advances our knowledge of and practice in optimizing long-running scientific simulations.

Key words: Checkpoint, Long-running, Optimizing, Simulation
doi:10.1631/jzus.C1200135 **Document code:** A **CLC number:** O242

1 Introduction

Many scientific simulations are long-running and computationally expensive. It is very time-consuming to restart a simulation from scratch if it dies prematurely. To prevent restarting from the beginning, a mechanism called checkpointing is used to save the state of the simulation periodically. Checkpoint and restart strategies have been under continuous investigation in the simulation, systems, and database communities. Chandy (1975) and Nicola (1995) provided excellent overviews of checkpointing and recovery strategies.

Long-running scientific simulations would benefit from this simple checkpointing mechanism that provides automatic restart or recovery in response to faults and failures, and enables dynamic load balancing and improved resource utilization through simulation migration (Kohl and Papadopoulas, 1998; Huang *et al*., 2004).

However, it is typically not a trivial task to determine the optimal interval between contiguous checkpoints. Excessive checkpointing would result in performance degradation and thus longer completion time, while deficient checkpointing would incur expensive recovery overheads and thus again longer completion time. Therefore, a trade-off must be made to determine the optimal checkpoint interval so that the total execution time can be minimized. We define checkpoint interval to be the time between two consecutive checkpoints. In this paper, we present mathematical models that analytically determine the optimal checkpoint interval.

2 Execution of a simulation

Fig. 1 shows the execution of a long-running scientific simulation, where each *xc* or *rxc* is called an execution segment. We call it an *xc*-segment or *rxc*-segment, respectively. The execution lifecycle of a simulation may include a sequence of checkpoints and possible restarts. Failures may occur any time during the execution of simulations. Once a failure occurs, the failure is detected and the simulation is

‡ Corresponding author

* Project supported by the National Science Foundation of USA and the Information Technology Research (ITR/AP-DEB) (No. 0112820)

restarted from the most recent checkpoint. Sometimes, we may even want to continue a previously completed simulation so that more simulation data can be produced and studied. When a simulation completes, its final state is checkpointed (Fig. 1).

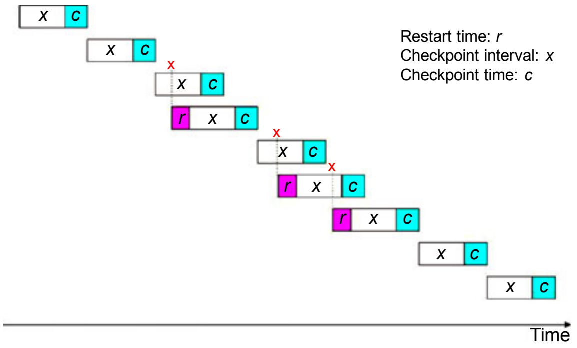

Fig. 1 Execution of scientific simulation

The total execution time of a simulation can be partitioned into the following four parts:

1. Work time (denoted by T_{work}): time needed to complete a simulation based on the assumption that the simulation never crashes and thus no checkpoint is necessary.

2. Checkpoint time (denoted by $T_{\text{checkpoint}}$): time spent to write checkpoint data to files or databases.

3. Redo time (denoted by T_{redo}): time spent to redo the simulation from the most recent checkpoint to the point of failure.

4. Restart time (denoted by T_{restart}): time needed to detect the failures and restore simulation states from checkpoint data (either data files or databases) so that redo can proceed.

The total execution time (denoted by T_{total}) is thus

$$T_{\text{total}} = T_{\text{work}} + T_{\text{checkpoint}} + T_{\text{redo}} + T_{\text{restart}}. \quad (1)$$

For scientific simulations, T_{work} is either explicitly specified by the user as an input or can be derived implicitly based on some terminating conditions. We denote T_{work} by N, and the checkpoint interval to be determined by x. To analytically derive the optimal checkpoint interval x^*, we make the following assumptions:

1. The average time before a crash occurs is M, and crashes occur according to a Poisson process with rate $1/M$. More precisely, (1) crashes occur randomly, but with a long-term average of one crash per M time units; (2) the likelihood of a crash is independent of the past history; (3) crashes are rare in a very short time interval, and there is a negligible chance of more than one crash in a very short time interval. This assumption is widely used in the literature related to checkpointing strategies, such as Tantawi and Ruschitzka (1983), Gelenbe and Hernandez (1990), and Kwak *et al*. (2001). Based on this assumption, the probability that a simulation successfully completes t time units is $p(t)=\text{e}^{-t/M}$.

2. Assume the checkpoint time is c and the restart time is r, where c and r are constants for all xc- or rxc-segments.

Let n be the expected number of failures occurring during the execution of a simulation, and let f be the fraction of redo time over the time of an execution segment when a crash occurs. At this point, we also assume that crashes do not occur during an rxc-segment; i.e., crashes do not occur immediately after a restart. However, this assumption will be removed in Section 5 when we present the best model. Based on the above assumptions, we have the following facts:

$$T_{\text{work}}=N,\ T_{\text{checkpoint}}=Nc/x,$$
$$T_{\text{redo}}=(x+c)fn,\ T_{\text{restart}}=rn.$$

Note that the number of execution segments without failures is N/x; thus, the total checkpoint time is $T_{\text{checkpoint}}=Nc/x$.

The expected total execution time is now

$$T_{\text{total}}=N+Nc/x+(x+c)fn+rn. \quad (2)$$

We need to derive n and f, so that we can analytically determine x to minimize T_{total}. We obtain the following:

1. The probability of successfully completing an xc-segment without a crash is $p(x+c)=\text{e}^{-(x+c)/M}$.

2. Therefore, the expected number of execution segments to complete N time units is

$$\frac{N}{xp(x+c)} = \frac{N}{x}\text{e}^{(x+c)/M}.$$

3. Thus, the expected number of failures n is

$$n = \frac{N}{xp(x+c)} - \frac{N}{x} = \frac{N}{x}(\text{e}^{(x+c)/M} - 1).$$

4. Suppose z is the time of an execution segment (which could be either $x+c$ or $r+x+c$). The distribution of failures occurring at t after the most recent checkpoint is

$$d(t)=\sum_{i=0}^{\infty}\mathrm{e}^{-(t+iz)/M}/M=\mathrm{e}^{-t/M}\Big/\left(M(1-\mathrm{e}^{-z/M})\right).$$

5. Therefore, the expected point of failure between 0 and z is

$$E(z)=\int_0^z td(t)\mathrm{d}t=\frac{\int_0^z t\mathrm{e}^{-t/M}\mathrm{d}t}{M(1-\mathrm{e}^{-z/M})}=M+\frac{z}{(1-\mathrm{e}^{z/M})}.$$

6. Thus, the expected fraction of redo over z time units is

$$f(z)=M/z+(1-\mathrm{e}^{z/M})^{-1}. \qquad (3)$$

Before we substitute n and f into Eq. (2), let us examine some properties of f. Letting $y=M/z$ and $g(y)=y+(1-\mathrm{e}^{1/y})^{-1}$, we have the following lemma:

Lemma 1 $g(y)=y+(1-\mathrm{e}^{1/y})^{-1}$ ($y>0$) is monotonically increasing and $\lim_{y\to\infty}g(y)=1/2$.

Proof To prove that $g(y)$ is monotonically increasing, it suffices to prove $\frac{\mathrm{d}g(y)}{\mathrm{d}y}>0$. In fact,

$$\frac{\mathrm{d}g(y)}{\mathrm{d}y}>0\Leftrightarrow 1-\frac{\mathrm{e}^{1/y}}{\left(y(1-\mathrm{e}^{1/y})\right)^2}>0$$
$$\Leftrightarrow y(\mathrm{e}^{1/y}-1)>\mathrm{e}^{1/(2y)}.$$

Expanding both sides of the last inequality using Taylor series, then

$$y(\mathrm{e}^{1/y}-1)=1+1/(2y)+\sum_{i=2}^{\infty}\left((i+1)!y^i\right)^{-1}$$

and

$$\mathrm{e}^{1/(2y)}=1+1/(2y)+\sum_{i=2}^{\infty}(2^i i!y^i)^{-1}.$$

It is easy to see $2^i>i+1$ when $i\geq 2$, and thus $(i+1)!y^i<2^i i!y^i$. Therefore $y(\mathrm{e}^{1/y}-1)>\mathrm{e}^{1/(2y)}$, and hence $g(y)$ is monotonically increasing. And

$$\lim_{y\to\infty}g(y)=\lim_{y\to\infty}\frac{(\mathrm{e}^{1/y}-1)y-1}{\mathrm{e}^{1/y}-1}$$
$$=\lim_{y\to\infty}\frac{\left(\sum_{i=0}^{\infty}(i!y^i)^{-1}-1\right)y-1}{\sum_{i=0}^{\infty}(i!y^i)^{-1}-1}$$
$$=\lim_{y\to\infty}\frac{\sum_{i=1}^{\infty}\left((i+1)!y^i\right)^{-1}}{\sum_{i=0}^{\infty}(i!y^i)^{-1}}$$
$$=\lim_{y\to\infty}\frac{1/2+\sum_{i=1}^{\infty}\left((i+2)!y^i\right)^{-1}}{1+\sum_{i=1}^{\infty}\left((i+1)!y^i\right)^{-1}}=1/2.$$

Fig. 2 confirms that for fixed checkpoint interval x, the average fraction of redo over an execution segment converges to 1/2 as M increases to positive infinity. Fig. 2 is generated through experiments on running simulations. Next, we derive our models to determine the optimal checkpoint intervals for autonomic Web-based simulations.

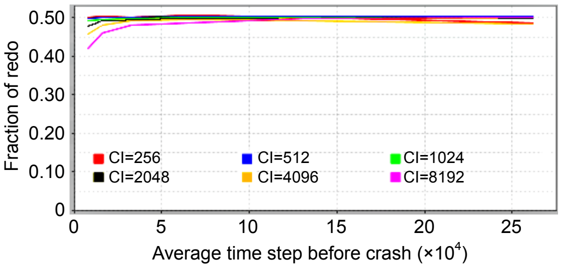

Fig. 2 Average fraction of redo over an execution segment
CI: checkpoint interval

3 Model I

We assume that M is sufficiently large compared to x, c, r. By Lemma 1, we see that $(x+c)f$ is approximately 1/2. Substituting $f=1/2$ and $n=(N/x)(\mathrm{e}^{(x+c)/M}-1)$ into Eq. (2), we obtain

$$T_{\text{total}}(x)=N+\frac{Nc}{x}+\left(\frac{x+c}{2}+r\right)\frac{N}{x}(\mathrm{e}^{(x+c)/M}-1). \qquad (4)$$

We need to find x^* so that $T_{\text{total}}(x)$ is minimized, which suffices to calculate x such that $\frac{\mathrm{d}T_{\text{total}}(x)}{\mathrm{d}x}=0.$

We then have

$$\mathrm{e}^{(x+c)/M}\left(x^2+(c+2r)x-(c+2r)M\right)+(2r-c)M=0. \tag{5}$$

It is unlikely to find an exact solution for this equation analytically. To solve it asymptotically, we need the following lemma:

Lemma 2 $-\ln y \approx 1-y$, if $0<y<1$ and $y\approx 1$.

Proof Use $\lim\limits_{x\to 0}\dfrac{\ln(1-x)}{x}=-1$.

By Lemma 2, Eq. (4) can be written as

$$\begin{aligned}\frac{x+c}{M}&=-\ln\frac{x^2+(c+2r)x-(c+2r)M}{(c+2r)M}\\&\approx 1-\frac{x^2+(c+2r)x-(c+2r)M}{(c-2r)M}.\end{aligned}$$

With standard algebraic calculations, the above equation can be simplified to $(x+c)^2=2(M+r)c$. Thus,

$$x=\sqrt{2(M+r)c}-c. \tag{6}$$

Hence, we have the following:

Theorem 1 The optimal checkpoint interval for minimizing total execution time is

$$x^*=\sqrt{2(M+r)c}-c,$$

and the expected total execution time is $T_{\text{total}}(x^*)$.

4 Model II

Instead of approximating f with 1/2, we now substitute $f=M/(x+c)+(1-\mathrm{e}^{(x+c)/M})^{-1}$ and $n=(N/x)(\mathrm{e}^{(x+c)/M}-1)$ into Eq. (2), and we obtain

$$\begin{aligned}T_{\text{total}}(x)=N+\frac{Nc}{x}+&\left(\left(\frac{M}{x+c}+\frac{1}{1-e^{(x+c)/M}}\right)(x+c)+r\right)\\&\cdot\frac{N}{x}(\mathrm{e}^{(x+c)/M}-1).\end{aligned}$$

After standard algebraic transformations, the above equation can be simplified as

$$T_{\text{total}}(x)=\frac{N(M+r)}{x}(e^{(x+c)/M}-1). \tag{7}$$

Again, to minimize $T_{\text{total}}(x)$, we take the first derivative of Eq. (7) and let it be zero, or equivalently,

$$(x+c)/M=-\ln(1-x/M). \tag{8}$$

If x^* is the solution to the above equation, then $\dfrac{\mathrm{d}^2T_{\text{total}}(x^*)}{\mathrm{d}x^2}=\dfrac{N(M+r)}{Mx^{*2}(M-x^*)}>0$. Hence, x^* achieves the minimum for $T_{\text{total}}(x)$. This is based on the fact that if the first derivative $f'(x)=0$ and the second derivative $f''(x)>0$, then x is a local minimum. It is far from trivial to solve Eq. (8) analytically. However, it is extremely simple to solve it numerically based on the fact in the following lemma:

Lemma 3 Eq. (8) has one and only one solution in the interval $(0, M)$.

Proof Let $g(x)=(x+c)/M+\ln(1-x/M)$. Then $g(0+)=c/M>0$ and $g(M-)=(M+c)/M+\ln(0+)=-\infty<0$.

Since $g(x)$ is continuous, there exists x in the interval $(0, M)$ such that $g(x)=0$.

Furthermore, $\dfrac{\mathrm{d}g(x)}{\mathrm{d}x}=\dfrac{1-(1-x/M)^{-1}}{M}<0$ for any $x\in(0, M)$, which means that $g(x)$ is monotonically decreasing in the interval $(0, M)$. Hence, there exists one and only one x such that $g(x)=0$. Thus, the lemma holds.

From the proof of the above lemma, given values of M and c, we can numerically solve equation $g(x)=0$ using a simple bisection algorithm. In the following algorithm, ε is typically chosen as 0.0001:

```
set $t_{\text{lo}}=0$ and $t_{\text{hi}}=M$
while ($t_{\text{hi}}-t_{\text{lo}}>\varepsilon$) do
    $t_{\text{mi}}=(t_{\text{lo}}+t_{\text{hi}})/2$
    if $g(t_{\text{lo}})g(t_{\text{mi}})>0$ then
        $t_{\text{lo}}=t_{\text{mi}}$
    else
        $t_{\text{hi}}=t_{\text{mi}}$
    end if
end while
```

Even if we cannot solve Eq. (8) analytically, we can solve it asymptotically. To find an asymptotic solution, we consider two cases:

Case I: $c/M \to 0+$, or $c \ll M$, i.e., the checkpoint time c is much less than the average time before crash M. Then $x/M \to 0+$, according to Eq. (8). Thus,

$$(x+c)/M = -\ln(1-x/M) \approx x/M + x^2/(2M^2),$$

and then we obtain

$$x = \sqrt{2Mc}. \tag{9}$$

Case II: Now we consider the general case. Let $d = \sqrt{2c/M}$ and $y=x/M$. Write y as

$$y = \sum_{n=0}^{\infty} a_n d^n. \tag{10}$$

From Case I, we see that $y \to d$ as $d \to 0$. Thus, we have $a_0=0$ and $a_1=1$. Expanding $\ln(1-x/M)$ using Taylor series, Eq. (8) becomes

$$d^2/2 - \sum_{n=2}^{\infty} (y^n/n) = 0. \tag{11}$$

To obtain an asymptotic solution, we now let $y=d+a_2d^2+a_3d^3+o(d^3)$ (where a_2 and a_3 are the coefficients to be determined and $o(d^3)$ is the higher order of d^3) and expand Eq. (11). Note that we are certainly able to approximate the solution to higher orders. However, this suffices to demonstrate the idea of obtaining an asymptotic solution. Equating the terms of powers with d, we have $a_2=-1/3$ and $a_3=1/36$. Thus, $y=d-d^2/3+d^3/36+o(d^3)$. Therefore,

$$x \approx \sqrt{2Mc}\left(1-\sqrt{2c/M}/3 + c/(18M)\right). \tag{12}$$

Now let x^* be the only root of Eq. (8). Then we have

$$T_{\text{total}} = N(M+r)\frac{e^{(x^*+c)/M}-1}{x^*} = N\frac{1+r/M}{1-x^*/M}.$$

Hence, we have the following theorem:

Theorem 2 The optimal checkpoint interval x^* that minimizes the total execution time is asymptotically $\sqrt{2Mc}\left(1-\sqrt{2c/M}/3 + c/(18M)\right)$. The expected total execution time is $N(1+r/M)/(1-x^*/M)$. Note that the optimal checkpoint interval is independent of the restart time r.

5 Model III

In Models I and II, we assume that no failure occurs during rxc-segments. In this section, we remove this assumption so that crashes may occur in both xc- and rxc-segments.

The probability that an rxc-segment completes without a failure is $\mathrm{e}^{-(r+x+c)/M}$. Thus, the probability that a failure does occur in an rxc-segment is $1-\mathrm{e}^{-(r+x+c)/M}$. For a simulation with total execution time units T_{total} and average time before crash M, the expected number of failures is T_{total}/M.

Therefore, the expected number of failures occurring in rxc-segments is $n_{rxc}=T_{\text{total}}(1-\mathrm{e}^{-(r+x+c)/M})/M$, and the expected number of failures occurring in xc-segments is $n_{xc}=T_{\text{total}}\mathrm{e}^{-(r+x+c)/M}/M$. Note that a restart time r is added after a failure in an xc-segment, while a restart time r is not added after a failure in an rxc-segment since it is already included. Therefore the total execution time is now

$$T_{\text{total}}(x)=N+Nc/x+[(x+c)^2f+r]n_{xc}+(r+x+c)^2fn_{rxc}.$$

Substituting f, n_{xc}, and n_{rxc} into the above equation and simplifying it, we obtain

$$T_{\text{total}}(x)=MN\mathrm{e}^{r/M}(\mathrm{e}^{(x+c)/M}-1)/x. \tag{13}$$

As before, we take the first derivative and let it be zero. Then we obtain $(x+c)/M=-\ln(1-x/M)$.

Surprisingly, we have the same minima for Model III as for Model II. Hence, we have the following theorem:

Theorem 3 The optimal checkpoint interval that minimizes the total execution time is asymptotically $x^* = \sqrt{2Mc}\left(1-\sqrt{2c/M}/3 + c/(18M)\right)$. The expected total execution time is $N\mathrm{e}^{r/M}/(1-x^*/M)$. Note that the optimal checkpoint interval is independent of the restart time r.

We find that T_{total} in Models II and III achieves its minimum values at the same argument value (more information can be found at http://en.wikipedia.org/

wiki/Arg_max). Let T_2=$T_{\text{total}}(x^*)$ in Model II and T_3=$T_{\text{total}}(x^*)$ in Model III. Then

$$\frac{T_3}{T_2}=\frac{e^{r/M}}{1+r/M}=\frac{1+r/M+\sum\limits_{n=2}^{\infty} r^n/(n!M^n)}{1+r/M}>1.$$

This means that although Models II and III reach minima at the same argument value, the expected total execution time is longer in Model III than in Model II. Note that $\lim\limits_{x\to 0}\frac{e^x}{1+x}=1$. Thus, $T_2\approx T_3$ if the restart time r is far less than M.

6 Experiments and discussion

A Java simulation is developed to experiment with and evaluate the above three models. The simulation generates random points of failure according to a Poisson process with rate $1/M$, and then outputs the total execution time as a function of N, M, r, x, and c. We run the simulation 1000 times for each distinct combination of N, M, r, x, and c. The complete code that generates all experiment data can be downloaded from http://bizresearch.una.edu/research/checkpointcode.zip.

Now we present the following experiments:

Experiment 1 We set N=10 000, M=2000, r=20, and c=10. The experiment results are shown in Fig. 3. Since r and c are very small compared to M, we find that all three models are good matches of the simulation results. In other words, there is little difference between the three models if r/M and c/M are small. The predicted optimal checkpoint interval for Model I is 191, and the predicted optimal checkpoint intervals for Models II and III are 193.

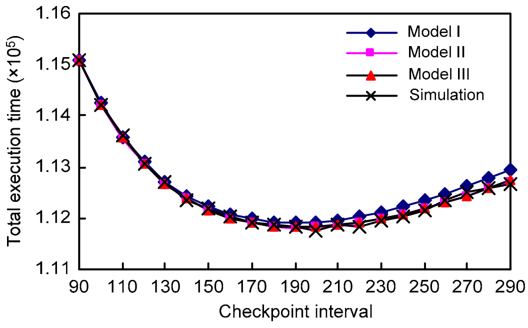

Fig. 3 Experiment 1 with *N*=10 000, *M*=2000, *r*=20, and *c*=10

Experiment 2 Now we increase the checkpoint time c in Experiment 1 and set N=100 000, M=2000, r=20, and c=100. The experiment results are shown in Fig. 4. We find that Model I deviates from the simulation results. However, both Model II and Model III are still in good agreement with the simulation results. Since $r\ll M$, there is no distinguishable difference between Models II and III. The predicted checkpoint interval for Model I is 536. The predicted optimal checkpoint intervals for Models II and III are 568.

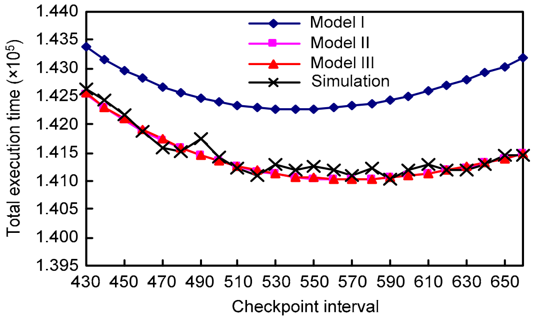

Fig. 4 Experiment 2 with *N*=100 000, *M*=2000, *r*=20, and *c*=100

Experiment 3 We increase the restart time r in Experiment 2 and set N=100 000, M=2000, r=200, and c=100. The experiment results are shown in Fig. 5. We find that both Model I and Model II deviate from the simulation results. However, Model III is still in good agreement with the simulation results. The predicted checkpoint interval for Model I is 563. The predicted checkpoint intervals for Models II and III are 568.

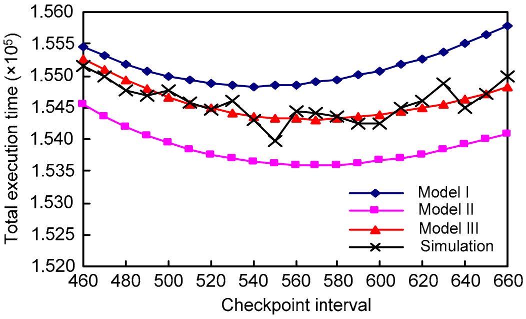

Fig. 5 Experiment 3 with *N*=100 000, *M*=2000, *r*=200, and *c*=100

From the above three experiments, we see that Model III is the best one to match the simulation results among all three models. When the restart time

r and checkpoint c are both far less than M, the three models are all in good agreement with the simulation results.

Next we experiment on how the changes of average time before crash M, checkpoint time c, and restart time r affect the choice of optimal checkpoint interval. The optimal checkpoint interval for the simulation is obtained as follows: first we calculate the optimal checkpoint interval x^* from Model II (or Model III, since they are the same); then we run the simulation 1000 times for each checkpoint interval in a wide neighborhood (for example, from x^*-100 to x^*+100) of x^*, and the checkpoint interval that results in least total execution time is chosen.

Experiment 4 We set N=100000, r=20, and c=100, and let M range from 1000 to 3000. Fig. 6 shows the experiment results of the relationship between M and the optimal checkpoint interval x^*. From the figure, we see that x^* monotonically increases as M increases for all three models. The optimal checkpoint interval calculated from Model II (or Model III) is in good agreement with that from the simulation.

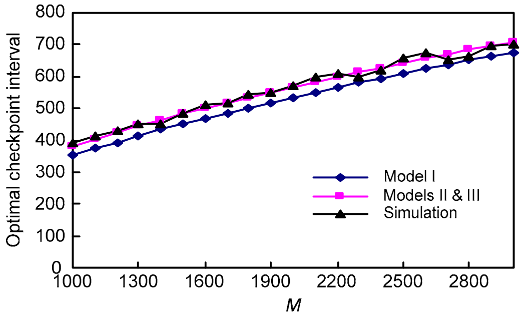

Fig. 6 Experiment 4 with N=100000, r=20, and c=100

Experiment 5 We set N=100000, M=2000, and r=20, and let c range from 100 to 300. Fig. 7 shows the experiment results of the relationship between c and the optimal checkpoint interval x^*. From the figure, we see that x^* monotonically increases as c increases for all three models. The optimal checkpoint interval calculated from Model II (or Model III) is in good agreement with that from the simulation.

Experiment 6 We set N=100000, M=2000, and c=100, and let r range from 100 to 300. Fig. 8 shows the experiment results of the relationship between r and the optimal checkpoint interval x^*. From the figure, we see that x^* monotonically increases as r increases for Model I, while staying constant for Models II and III. The simulation shows that x^* is almost constant, which confirms that the choice of optimal checkpoint interval is independent of the restart time r.

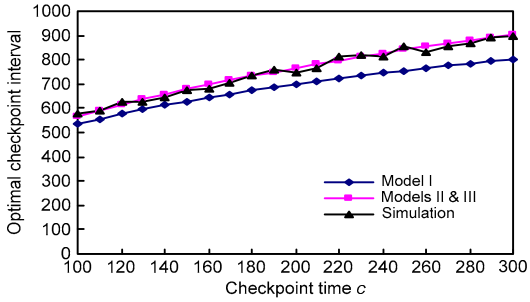

Fig. 7 Experiment 5 with N=100000, M=2000, and r=20

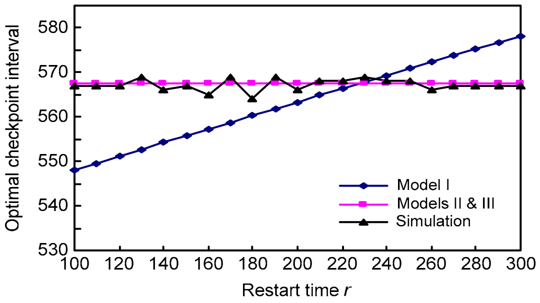

Fig. 8 Experiment 6 with N=100000, M=2000, and c=100

From these experiments, we conclude that Model III is a good model so we can use it to calculate the optimal checkpoint interval and predict the total execution time of a simulation. M and c for a specific scientific simulation can be determined empirically by running the simulation a sufficient number of times. For example, in our case study of the natural organic matter (NOM) simulation (please check http://www.nd.edu/~nom for more information about the National Science Foundation (NSF)-funded NOM project), M is determined by running the simulation many times without checkpointing, and c is determined easily by calculating the average time spent on checkpointing.

One more interesting finding about Model III is the following lemma:

Lemma 4 Let

$$T(x)=NM\mathrm{e}^{r/M}(\mathrm{e}^{(x+c)/M}-1)/x, \tag{14}$$

where c>0, r>0, and $x\in(0,M)$. Supposing $x^*=\arg\min_{0<x<M} T(x)$, and then for any t>0 such that $0<x^*-t<$

$x^*+t<M$, we have $T(x^*-t)>T(x^*+t)$.

Proof The proof is omitted since it is straightforward.

Lemma 4 means that an under-predicted checkpoint interval results in longer total execution time than an equivalent over-predicted one. This conclusion can be confirmed from Figs. 3–5.

7 Real application of Model III

Model III has been extensively applied in the NSF-funded interdisciplinary project—the NOM project (http://www.nd.edu/~nom). NOM is a heterogeneous mixture of organic molecules found in terrestrial and aquatic environments—from forest soils and streams to coastal rivers and marshes to the open ocean. NOM plays a vital role in ecological and biogeochemical processes, including chemical buffering, mineral dissolution/precipitation, photochemistry, and microbial nutrition. This project consists of an interdisciplinary team of environmental (biology, chemistry, geology) and IT scientists that is developing a stochastic model for the time-dependent evolution of NOM in the environment. The scientific objectives are to produce both a new methodology and a specific program for predicting the properties of NOM over time as it evolves from precursor molecules to eventual mineralization. The methodology being developed is a mechanistic, stochastic simulation of NOM transformations, including biological and non-biological reactions, as well as adsorption, aggregation, and physical transport. It employs recent advances in agent-based simulation, Web-based deployment of scientific applications, collaboratory for sharing simulations and data, and scalable Web-based database management systems to improve the reliability of the stochastic simulations and to facilitate analysis of the resulting large datasets using data-mining techniques. The stochastic synthesis model of NOM evolution is being developed as a dialogue between coding and numerical testing on the one hand and environmental data-based testing on the other. Initially, a simple model program was coded to simulate 'steady state' transformations of NOM in a stable environment. This program was made available to applications scientists in biology and geochemistry, who tested the behavior of the model for fidelity to laboratory data and field observation, and who directed the parameterization of the model and the inclusion or deletion of various molecular transformations (e.g., hydrolysis, photolysis, absorption, and microbial consumption). The program is being modified as indicated by these tests, and additional code modules will be added to simulate NOM transport (in soil, ground, or surface waters) and the response of the biological community. We make the Web-based simulation available to external investigators.

The Web-based system delivers long-running scientific simulations available to scientists across the US. The system was carefully designed to be highly reliable and scalable using grid computing technologies. In other words, the system was intended to be self-healing, i.e., being able to recover from errors. The state data of the simulation were periodically archived to a database and the simulation will be able to restart from the most recent checkpoint to the failure point. The checkpoint interval was determined by using Model III. The simulation system has been approved to be very reliable and have served external investigators since 2004.

8 Related works on checkpointing

Analytically determining optimal checkpoint dates back to as early as 1974 when Young (1974) presented a first-order approximation to the optimum checkpoint interval. The first-order approximation was $\sqrt{2cM}$, which coincides with the special case in Eq. (9). Our work extends this result to more general cases.

In Duda (1983), Shin *et al.* (1987), and Kulkarni *et al.* (1990), a model with Poisson failure is considered to determine the optimal number of checkpoints, which minimizes the expected execution time of a program, with an assumption that no failure occurs during the checkpoint and restart phases. The same model was extended by Grassi *et al.* (1992) in which the optimal number of checkpoints relies on the distribution of the program execution time. Tantawi and Ruschitzka (1983) considered a model with a general distribution of failures and allowed failures to occur during the checkpoint and restart phase. This generality yields a model that needs to compute an infinite number of integrals, which is computationally intractable. A simpler model is proposed by preventing failures from occurring during checkpoint and restart. However, this simplification still results in computing an infinite number of non-linear equations. Thus, an even simpler model is then proposed by imposing more constraints which assume that the execution time between two successive checkpoints is constant and that the expected restart time equals the mean checkpointing time. With this simplification, an iteration algorithm with dynamic programming is used to compute an approximation of the optimal number of checkpoints. Ling *et al.* (2001) used a variational calculus approach to derive an explicit formula that links the optimal checkpointing frequency with a general failure rate, with the objective of globally minimizing the total expected cost of checkpointing and recovery. The results show that the optimal checkpointing frequency is proportional to the square root of the failure rate.

More recently, Kwak and Yang (2012) presented an analytical model to derive the optimum number of checkpoints with the objective of maximizing the probability of completing all tasks within deadlines, where they assumed that the transient faults follow a Poisson distribution. Ji *et al.* (2011) proposed a heuristic algorithm for checkpoint placement with the objective to improve application performance. This checkpoint mechanism was inserted into application source code so that the application can restart automatically and transparently. Cao *et al.* (2011) took advantage of the frequently consistent applications to develop checkpoint recovery algorithms that obtain lower overheads and reduce latency by trading space in main memory.

Our work differs from the aforementioned work in several ways: (1) our simple assumption (failures occur according to a Poisson process) yields a simple cost function which is easy to solve numerically using a simple bisection algorithm, and (2) experiments on simulation show that the model is in good agreement with simulations.

9 Summary

In this paper, three models are discussed to determine the optimal checkpoint interval and predict total execution time for long-running scientific simulations. We can draw the following conclusions from the discussions:

1. Model III is the best model to calculate the optimal checkpoint interval and predict total execution time.

2. The choice of checkpoint interval is independent of the restart time *r*; however, the predicted total execution time is exponentially dependent on *r*. Therefore, the total execution time can be reduced dramatically if a failure can be detected and restart can be accomplished quickly.

3. An under-predicted checkpoint interval results in longer total execution time than an equivalent over-predicted one; therefore, we would rather choose a larger checkpoint interval if it is not possible or is too difficult to calculate the optimal.

The ability to restart an abortive simulation from the most recent checkpoint is a basic requirement for developing and deploying autonomic Web-based simulations (Huang and Madey, 2005). In particular, it helps to achieve self-healing and self-optimizing features of autonomic Web-based simulations.

References

Cao, T., Vaz Salles, M., Sowell, B., Yue, Y., Demers, A., Gehrke, J., White, W., 2011. Fast Checkpoint Recovery Algorithms for Frequently Consistent Applications. Proc. ACM SIGMOD Int. Conf. on Management of data, p.265-276. [doi:10.1145/1989323.1989352]

Chandy, K., 1975. A survey of analytic models for rollback and recovery strategies. *Computer*, **8**(5):40-47. [doi:10.1109/C-M.1975.218955]

Duda, A., 1983. The effects of checkpointing on program execution times. *Inf. Process. Lett.*, **16**(5):221-229. [doi:10.1016/0020-0190(83)90093-5]

Gelenbe, E., Hernandez, M., 1990. Optimum checkpoints with age dependent failures. *Acta Inf.*, **27**(6):519-531. [doi:10.1007/BF00277388]

Grassi, V., Donatiello, L., Tucci, S., 1992. On the optimal checkpointing of critical task and transaction-oriented systems. *IEEE Trans. Software Eng.*, **18**(1):72-77. [doi:10.1109/32.120317]

Huang, Y., Madey, G., 2005. Autonomic Web-Based Simulations. Proc. 38th Annual Simulation Symp., p.160-167. [doi:10.1109/ANSS.2005.15]

Huang, Y., Xiang, X., Madey, G., 2004. A Self Manageable Infrastructure for Supporting Web-Based Simulations. Proc. 37th Annual Simulation Symp., p.149-156. [doi:10.1109/SIMSYM.2004.1299478]

Ji, Y., Jiang, H., Chaudhary, V., 2011. A heuristic checkpoint placement algorithm for adaptive application-level checkpointing. *Int. J. Appl. Sci. Technol.*, **1**(6):50-61.

Kohl, J., Papadopoulas, P., 1998. Efficient and Flexible Fault Tolerance and Migration of Scientific Simulations Using CUMULVS. Proc. SIGMETRICS Symp. on Parallel and Distributed Tools, p.60-71. [doi:10.1145/281035.281042]

Kulkarni, V.G., Nicola, V.F., Trivedi, K.S., 1990. Effects of checkpointing and queuing on program performance. *Commun. Stat. Stoch. Models*, **6**(4):615-648. [doi:10.1080/15326349908807166]

Kwak, S., Yang, J., 2012. Optimal checkpoint placement on real-time tasks with harmonic periods. *J. Comput. Sci. Technol.*, **27**(1):105-112. [doi:10.1007/s11390-012-1209-0]

Kwak, S.W., Chio, B.J., Kim, B.K., 2001. An optimal checkpointing strategy for real time control systems under transient faults. *IEEE Trans. Reliab.*, **50**(3):293-301. [doi:10.1109/24.974127]

Ling, Y., Mi, J., Lin, X., 2001. A variational calculus approach to optimal checkpoint placement. *IEEE Trans. Comput.*, **50**(7):699-708. [doi:10.1109/12.936236]

Nicola, V., 1995. Checkpointing and the Modeling of Program Execution Time. *In*: Lyu, M.R. (Ed.), Software Fault Tolerance. John Wiley & Sons, Chichester, England, p.167-188.

Shin, K.G., Lin, T., Lee, Y., 1987. Optimal checkpointing of real-time tasks. *IEEE Trans. Comput.*, **36**(11):519-531.

Tantawi, A.N., Ruschitzka, M., 1983. Performance Analysis of Checkpointing Strategies. Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, p.129.

Young, J.W., 1974. A first order approximation to the optimum checkpoint interval. *Commun. ACM*, **17**(9):530-531. [doi:10.1145/361147.361115]