数理解析研究所講究録
1526巻 2006年 20-24

動的計画法による最適チェックポイント間隔に関する考察

岩本 一樹[†], 岡村 寛之[†], 円尾 忠司[†], 土肥 正[†]

[†]広島大学大学院工学研究科情報工学専攻

1. はじめに

ソフトウェアシステムのディペンダビリティを向上させるソフトウェアフォールトトレランス技術にチェックポインティング [1] と呼ばれる手法がある．チェックポイントとは，処理したデータあるいは処理中のデータを主記憶から安定な二次記憶媒体に保存する時期のことである．これらを配置することにより，システム障害が発生した場合，直前のチェックポイントまで後ろ向き回復（ロールバックリカバリ）を行い，そこから処理を再開することが可能となる．これにより，システム障害による再処理の無駄を軽減することができる．さらに，処理過程のログファイルを生成しておくことにより，チェックポイントから障害発生時までの処理を復元することが可能になる（ロールフォワード）．しかしながら，チェックポイントの生成には若干のコスト（処理時間や処理負荷など）が発生する．つまり，過剰なチェックポイントの生成を行うと，障害が発生した場合における再実行に要するコストは少なくて済むが，チェックポイントを生成するためのコスト（チェックポイントオーバーヘッド）が増大してしまう．逆にCP配置が疎な場合は，障害時のロールバックリカバリおよび処理の再実行に要するコストが増大することになる．従って，それらのトレードオフを考慮した上で，ある評価規範にしたがって最適なチェックポイント生成時間間隔（CP系列）を求めることは重要な問題となる．

Ozaki *et al.* [2] は連続的に稼動するシステムに対して，min-max法を用いて期待費用を最小にする最適チェックポイント配置アルゴリズムを提案している．しかしながら，ここで提案されたアルゴリズムは収束条件や探索範囲に関してあいまいな部分が多いヒューリスティックなアルゴリズムである．Toueg *et al.* [3] は計画期間を離散化した動的計画法を用いることで期待処理終了時間を最小にする最適なCP系列を導出している．しかし，離散化によるアプローチは近似的な手法であり，得られる最適CP系列が刻み幅に大きく依存する．一方，Okamura *et al.* [4] は厳密に連続時間の最適CP系列を離散化によるものと全く異なる動的計画法によって導出する手続きを提案している．そこで本稿では，Toueg *et al.* の離散化によるアプローチとOkamura *et al.* による厳密解を導出する手法を紹介し，数値例を通じてこれらの違いを検証する．

2. 有限計画期間におけるCP配置問題

システムは時刻 $t=0$ から稼動を開始し，時間区間 $[0,T]$ の間にチェックポイントを配置するものとする．また，各チェックポイントは非周期に配置されていくものとし，そのスケジュールは $\pi=\{t_1,t_2,\ldots\}$ とする．チェックポイントの配置には $\mu_c(>0)$ のオーバーヘッドがかかるものとする．時刻 T を経過した時点で，データの書き出しやシステムのメンテナンスなどを行うことで障害年齢は元に戻るものとする．システム障害の発生時間は確率分布 $F(t)$ に従い，$f(t)$ をその密度関数とする．システム障害が発生した場合，直ちにロールバックリカバリを行う．そのオーバーヘッドは直前のチェックポイントからの経過時間 x に依存した関数 $\rho(x)$ で与えられるものとする．このとき障害年齢も元に戻るものとする．ここで，システムの稼動開始から時間 T 経過後およびロールバックリカバリ後の障害年齢が再生する時点までを1サイクルとして定義する．チェックポイント配置問題の概念図を図1に示す．

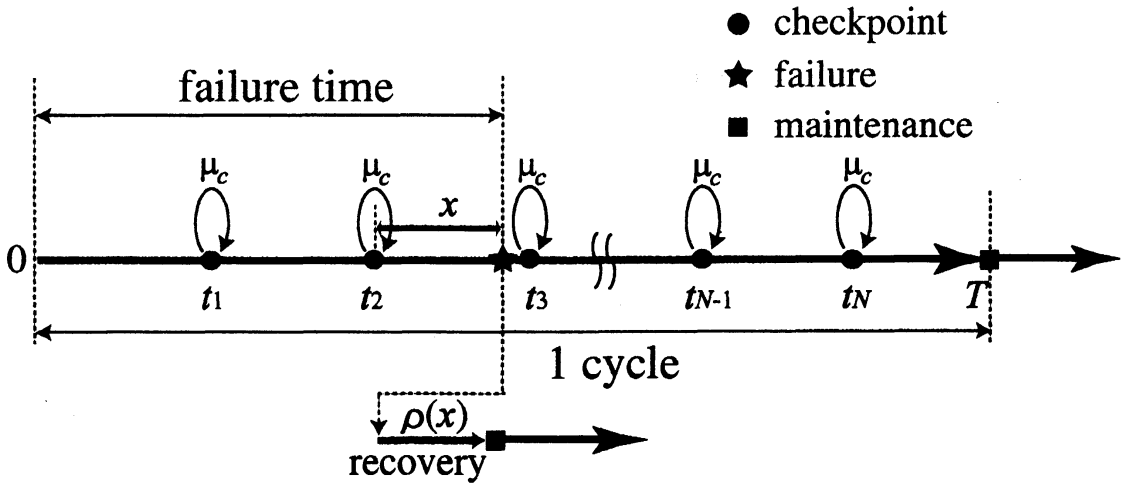

図 1: CP 配置問題の概念図.

3. 動的計画法による最適チェックポイント配置

3.1 離散化によるアプローチ

処理限界時間 T を n 個の区間に分割し，それぞれの区間において処理開始時にチェックポイントを配置するかどうかの決定を行う．これにより，チェックポイント配置時間は離散化することになる．$T^k_{i,j}$ を，区間 $[i,j](1 \le i \le j \le n)$ においてチェックポイントを最大 k 個配置したときの期待無駄時間，$L^k_{i,j}$ をそのときのチェックポイント配置系列 $\{u_1,\ldots,u_k\}(1 < u_1 < \ldots < u_k \le j)$ と定義する．このとき，

$$T^0_{i,j} = \frac{\int_0^{t_{i,j}} \rho(t)dF(t)}{1 - F(t_{i,j})} \tag{1}$$

となる．ここで，$t_{i,j}$ は区間 $[i,j]$ の時間間隔である．これにより，最適性の原理から以下の最適性方程式が得られる．

$$W = \min_{1<i\le j}(T^{k-1}_{1,i-1} + T^0_{i,j} + \mu_c). \tag{2}$$

これらにより，離散化された時間上での最適 CP 系列を導出する動的計画法アルゴリズムは以下のように得られる．

Algorithm 0:

Step 1: $T^0_{i,j}$ を計算する．$i = 1,2,\ldots,n,\ j = i, i+1,\ldots,n.$

Step 2: 初期値を求める．$T^k_{1,1} = T^0_{1,1},\ k = 1,\ldots,n-1$

Step 3: すべての $k = 1,\ldots,n-1, j = n, n-1,\ldots,2$ に対して

$$W = \min_{1<i\le j}(T^{k-1}_{1,i-1} + T^0_{i,j} + \mu_c). \tag{3}$$

ここで，$W < T^{k-1}_{i,j}$ ならば $T^k_{1,j} = W$ とし，得られた i を $L^k_{1,j}$ に追加する．そうでなければ $T^k_{1,j} = T^{k-1}_{1,j},\ L^k_{1,j} = L^{k-1}_{1,j}$ とする．

3.2 厳密解

動的計画法を用いて，連続時間環境下でシステムの期待無駄時間を最小にする最適チェックポイント配置時刻列を導出することを考える．いま，$T_i(t_{i+1}|t_i)$ を，i 番目のチェックポイントが時刻 t_i で生成されたもとで，$i+1$ 番目の CP を時刻 $t_i \le t_{i+1} \le t^*_{i+2}$ で生成し，以降は最適な配置 π^* に従ってチェックポイントを生成したときの期待無駄時間，v^*_i を $i+1$ 番目の CP 配置時における期待最小無駄時間と定義する．このとき，最適性の原理から次の最適性方程式が得られる．

$$v^*_i = \min_{t^*_i \le t_{i+1} \le t^*_{i+2}} T_i(t_{i+1}|t^*_i), \tag{4}$$

$$\begin{aligned} T_i(t_{i+1}|t_i) &= \int_{t_i}^{t_{i+1}} \{\mu_c(i+1) + \rho(t-t_i)\} dF(t) \\ &\quad + v^*_{i+1}, \qquad i = 0, 1, \ldots, N-1. \end{aligned} \tag{5}$$

これを満たす $\{t_1, \ldots, t_N\}$ が最適な CP 系列となる．$\rho(x) = ax + b$ とし，上記の最適性方程式を解くために 2 区間のチェックポイント配置をまとめた次の関数を定義する．

$$\begin{aligned} C_i(t_{i+1}|t_i, t_{i+2}) &= a(t_{i+1} - t_i)\lambda(t_{i+1}) - \mu_c\lambda(t_{i+1}) \\ &\quad -a(1 - \overline{F}(t_{i+2})/\overline{F}(t_{i+1})). \end{aligned} \tag{6}$$

ここで，一般に $\overline{\phi}(\cdot) = 1 - \phi(\cdot)$ である．さらに，$\lambda(\cdot) = f(\cdot)/\overline{F}(\cdot)$ であり，$F(\cdot)$ の故障率を表す．これらにより，Policy Iteration を用いた最適な CP 系列を導出する動的計画法アルゴリズムは以下のように得られる．

Algorithm 1

Step 1: $k := 0, t_{N+1} := T$ とする．

Step 2: 初期 CP 系列 $\pi^{(k)} = \{t^{(k)}_1, \ldots, t^{(k)}_N\}$ と初期期待無駄時間 $w^{(k)}_i := 0$ を与える．

Step 3: すべての $i = 0, \ldots, N-1$ に対して

$$t^{(k+1)}_i := \{t^{(k)}_i \le t \le t^{(k)}_{i+2}; C_i(t|t^{(k)}_i, t^{(k)}_{i+2}) = 0\}, \tag{7}$$

$$\begin{aligned} w^{(k+1)}_i &:= \int_{t^{(k)}_i}^{t^{(k+1)}_{i+1}} \{\mu_c(i+1) + \rho(t - t^{(k)}_i)\} dF(t) \\ &\quad + \int_{t^{(k+1)}_{i+1}}^{t^{(k)}_{i+2}} \{\mu_c(i+2) + \rho(t - t^{(k+1)}_{i+1})\} dF(t) \\ &\quad + w^{(k)}_{i+2} \end{aligned} \tag{8}$$

を算出する．ここで，$w_N = w_{N+1} = 0$ である．

Step 4: 許容誤差 δ に対し，すべての $i = 1, \ldots, N$ において $|t^{(k)}_i - t^{(k+1)}_i| < \delta$ ならば終了，そうでなければ Step 3 へ．

4. 数値例

ここでは，提案手法を用いた最適 CP 系列の導出，及びそのときの期待無駄時間の算出例を示す．システム障害の発生時間分布を

$$F(t) = 1 - \exp\left\{-\left(\frac{x}{\eta}\right)^m\right\} \tag{9}$$

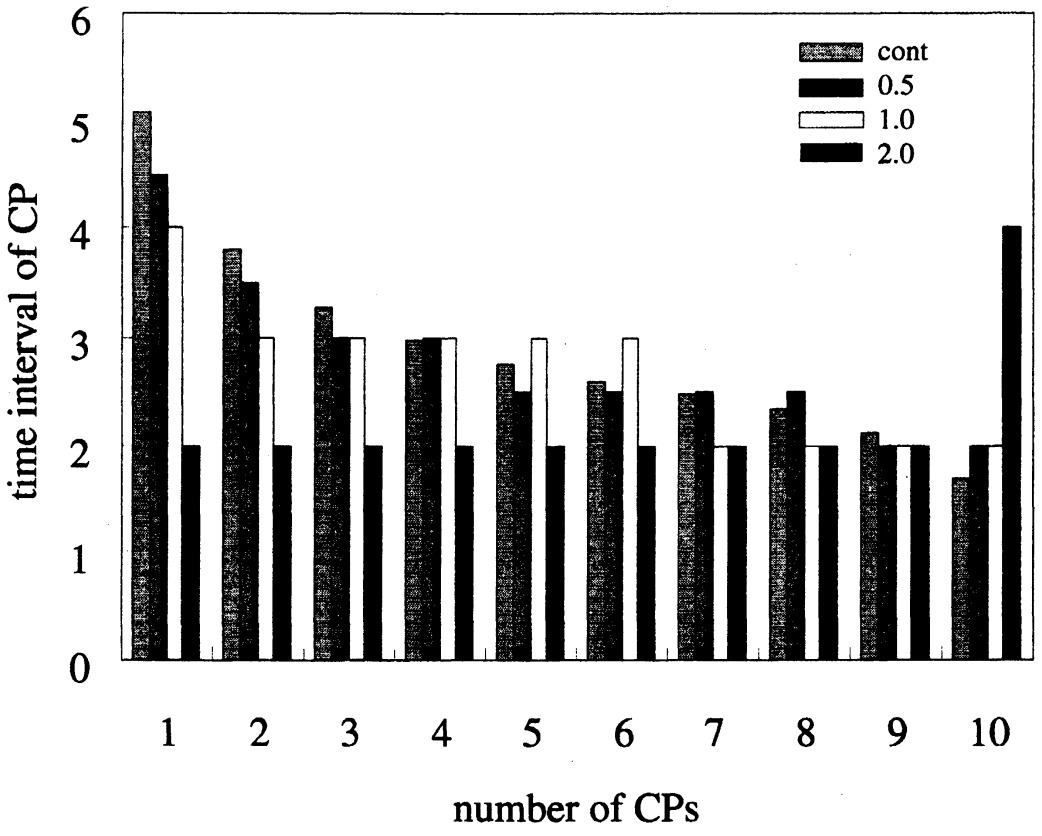

図 2: 連続，離散時間におけるチェックポイント配置間隔の比較.

と表されるワイブル分布と仮定する．ここで m は形状パラメータ，η は尺度パラメータである．また，他のパラメータとして $T = 30$, $\mu_c = 1$, $\rho(x) = x$ を設定する．$m = 2, \eta = 10$ のとき，$F(t)$ は IFR(Increasing Failure Rate) となる．図 2 は連続時間と離散時間におけるチェックポイントの配置間隔を表している．cont は連続時間環境における厳密解を表し，離散時間におけるチェックポイント配置可能時間の間隔はそれぞれ 0.5, 1.0, 2.0 と設定した．離散時間でのチェックポイント配置時間間隔が大きくなるにつれ，最適な CP 系列は連続時間からずれてくることがわかる．また，表 1 は配置チェックポイント数 N を 1 から 10 まで増やしていったときの期待無駄時間の比較を表している．これより，このパラメータ設定における期待無駄時間を最小にする最適チェックポイント数は $N = 2$ となった．また，すべての N において，Toueg *et al.* のアルゴリズムと比較して厳密解による連続時間環境での期待無駄時間が最も小さく，離散時間環境の CP 配置可能間隔が大きくなるにつれ，期待無駄時間が増加していることがわかる．これは，チェックポイント配置が離散環境下でしか配置できないことにより，厳密な最適解でチェックポイントを配置することができないためと考えられる．しかし，Toueg *et al.* のアルゴリズムは Okamura *et al.* のアルゴリズムと比較して高速にチェックポイント配置系列を導出できるため，今後計算時間を考慮した比較検証を行う必要がある．

5. まとめと今後の課題

本稿では，有限計画期間におけるチェックポイント配置問題を考え，Toueg *et al.* の提案した離散時間アルゴリズムと Okamura *et al.* の提案した連続時間アルゴリズムの比較を数値

表 1: チェックポイント数に対する期待無駄時間の比較.

N	cont	0.5	1.0	2.0
1	6.15484	6.15546	6.16806	6.16806
2	3.41670	3.45480	3.50485	3.68364
3	4.81810	4.81920	4.84639	5.01027
4	4.24589	4.34310	4.43768	4.43768
5	4.63487	4.64671	4.68948	4.70454
6	4.53411	4.56256	4.60482	4.64652
7	4.61232	4.61576	4.62875	4.71322
8	4.59948	4.62174	4.72642	4.75043
9	4.61040	4.64451	4.74492	5.19260
10	4.60940	4.64236	4.73857	5.83503

例を用いて検証した．結果として，連続時間環境で最適 CP 系列を導出することにより期待無駄時間をより削減することが可能であることがわかった．今後の課題として，それぞれのアルゴリズムの計算時間を考慮し，期待無駄時間の削減度合と計算時間の両方を包含した評価規範を設定して検証を行う予定である．

REFERENCES

[1] Chandy, K. M. (1975), A survey of analytic models of roll-back and recovery strategies, *Computer*, **8** (5), 40–47.

[2] Ozaki, T., Dohi, T., Okamura, H. and Kaio, N. (2004), Min-max checkpoint placement under incomplete failure information, *Proc. 2004 Int'l. Conf. on Dependable Systems and Networks*, 721–730, IEEE CS Press.

[3] Toueg, S. and Babaoglu, Ö. (1984), On the optimum checkpoint selection problem, *SIAM J. of Comput.*, **13** (3), 630–649.

[4] Okamura, H., Iwamoto, K. and Dohi, T. Ö. (2006), A DP-based optimal checkpointing algorithm for real-time appications, *Int. J. of Reliability, Quality and Safety Engineering.*, **13** (4), 323–340.