大域ディペンダブルプロジェクト
～ディペンダブルアーキテクチャグループ 南谷·中村研究室～

南谷崇　中村宏

情報理工学系研究科システム情報学専攻 (先端科学技術研究センター)

1　はじめに

クラスタシステムは極めて低コストに高性能計算環境が構築可能であるため，大規模科学技術計算を中心に近年広く利用されている．一度の計算時間が数時間から数日に及ぶことも少なくない大規模科学技術計算では，この間の障害発生に対処する高信頼化技術は重要なものとなってきている．チェックポインティングは特別なハードウェアを必要としないためクラスタシステムの高信頼化手法として有望であり，実際にクラスタシステム用ミドルウェアの SCore[1] ではチェックポインティングがサポートされている．

チェックポインティング手法の研究では，システム内での故障率の均一性が前提とされている．故障率の均一性とは，システムを構成するノードの故障率が共通であること，実行時間中のどの時点でも故障率が一定であることである．しかし，チェックポインティングが用いられる対象である計算機クラスタシステムでは今後，利用者の増加や利用分野の拡大，システムの大規模化が進むと予想され，そのような多様な環境では一システムの中でも故障の集中するノード，そうでないノードの分類が生じると考えられる．例えば，運用中のシステムの拡張により故障率の異なるノードが混在したり，アプリケーションによっては，計算負荷・ネットワーク負荷の集中が起き実行時間中に故障率が変動することがある．

このような故障率が非均一なときのチェックポインティングの最適化に関する研究はこれまで行われていない．そこで我々は，故障率に空間的な異なりがある場合に着目し，このような場合に効率良くチェックポインティングを行う手法を提案検討した．

2　提案手法

2.1　方針

本論文では，1MIR[2] の coordinated チェックポインティングに焦点を絞り，故障率に空間的な異なりがある場合に有効な 1MIR coordinated チェックポインティング手法を新しく提案する．

冗長度 1 の 1MIR の他に，冗長度を多くした MIR[2] や，故障しない安定なファイルサーバを外部に設置し，全ノードのチェックポインティングデータをそのサーバに保存する CFS[2] という手法がある．多冗長化は，大規模システムで問題となる多重故障に対応するための手法であり，一般に，冗長度 1 の 1MIR では多重故障を回復することはできない．しかし，冗長度を多くした MIR では，チェックポインティングに要する時間が冗長度に比例して長くなる問題がある．

我々はすでに文献 [3] において，多重故障を考慮した場合でも，冗長度の多い MIR や CFS よりも，1MIR においてチェックポインティングデータの保存先を工夫したほうが良いという結果を報告している．そこで本論文でも，システム内で最も多く発生する 1 ノードのみの単独故障に対応でき，しかも，チェックポインティングに要する時間が短い 1MIR に焦点を絞り，故障率に空間的な異なりがある場合に多重故障の確率を抑えるように 1MIR を工夫することにした．

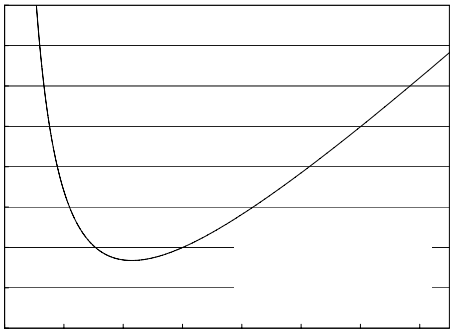

図 1: チェックポインティング間隔とオーバーヘッドの関係

工夫すべき点は，チェックポインティング間隔の最適化である．coordinated チェックポインティングにおいては，その時間間隔は総実行時間に大きな影響を与えるため，故障率に応じた適切な値を選択する必要がある．ところが，ノードごとに故障率が異なるシステムでは，全ノード共通の最適チェックポインティング間隔を何らかの方法で決定しなければいけない．そこで，システムの平均故障率を用いて，この最適チェックポインティング間隔を簡便に決定する手法を提案する．

2.2 最適チェックポインティング間隔の決定

チェックポインティングを行う際にはその間隔が重要なパラメータとなる．チェックポインティング間隔の長短とオーバーヘッドの関係を図 1 に示す．故障発生時のロールバック量を抑えるためにチェックポインティングを頻繁に行うと，計算停止時間が増えオーバーヘッドが増加する（グラフの左側）．しかし，チェックポインティングのオーバーヘッドを抑えるために頻度を少なくするとロールバック量が増加し計算完了まで時間がかかる（グラフの右側）．チェックポインティング間隔はこの両者のトレードオフの最適点を選ぶ必要がある．

最適なチェックポインティング間隔 I と，そのときのオーバーヘッドは markov モデルを用いて求めることが可能であり [3]，次のものに依存する．

- アプリケーションの実行時間 Υ
- ノード数 N
- 故障率 λ
- チェックポインティング時間 C
- リカバリ時間 R

故障率 λ に依存するということは，ある故障率に対して最適なチェックポインティング間隔が一意に定まるということである．しかし，故障率が異なるノードが混在するシステムは，そのままでは最適チェックポインティング間隔を求めることができない．そこで，システムの平均故障率を用いる手法を提案する．

全 N 台の計算機クラスタのうち，N_1 台が故障率 λ_1，N_2 台が故障率 λ_2，…，N_n 台が故障率 λ_n を持っているとする．

このとき，システムの平均故障率 λ_{ave} を次式で定義する．

$$\lambda_{\mathrm{ave}} = \frac{N_1\lambda_1 + N_2\lambda_2 + \cdots + N_n\lambda_n}{N_1 + N_2 + \cdots + N_n} \qquad (1)$$
$$\text{ただし,} N_1 + N_2 + \cdots + N_n = N$$

故障率が異なるシステムでの最適チェックポインティング間隔決定手法として，システムの全ノードが（1）式の平均故障率 λ_{ave} を持っているとみなし，markov モデルと λ_{ave} によって最適チェックポインティング間隔を決定する，という手法を提案する．

この手法では，ノードごとに故障率が異なるシステムから，故障率が共通のシステムへの等価変換を行っている．以下では，平均故障率 λ_{ave} によって等価変換が可能であることを証明する．

2.3 等価変換の証明

定期的にチェックポインティングを行うシステムの markov モデルは図 2 のように構築できる．すなわち，

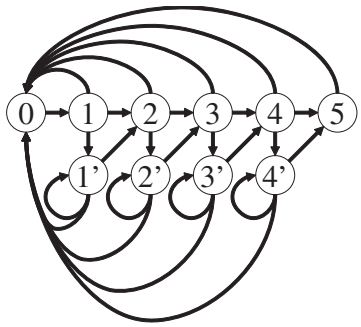

図 2: markov モデルの構築例

状態 i：i 番目のチェックポインティング完了時点から $(i+1)$ 番目のチェックポインティング完了まで
状態 i'：状態 i で故障が発生したときの，回復処理の開始から完了まで
状態 0：実行開始から第 1 回目のチェックポインティング完了まで
と定義できる．このとき，N 台のノードが共通して故障率 λ を持つとすると，状態 i から $(i+1)$ への遷移確率 $P_{i(i+1)}$ は

$$P_{i(i+1)} = \exp\bigl(-N\lambda(I+C)\bigr) \tag{2}$$

で表される [3].

一方，故障率が空間的に異なるシステムで，故障率 λ_k を持つノード群 (ノード数 N_k) が状態 i から $(i+1)$ に遷移する確率 $P_{k,i(i+1)}$ は I をチェックポインティング間隔，C をチェックポインティング時間として

$$P_{k,i(i+1)} = \exp\bigl(-N_k\lambda_k(I+C)\bigr) \tag{3}$$

ここで，状態 i から $(i+1)$ に遷移する事象は各ノード群で独立なので，システム全体が状態 i から $(i+1)$ へ遷移する確率は

$$\begin{aligned} P_{i(i+1)} &= \Pi_{k=i}^{n} P_{k,i(i+1)} \\ &= \exp\bigl(-(N_1\lambda_1+\cdots+N_n\lambda_n)(I+C)\bigr) \\ &= \exp\Bigl(-(N_1+\cdots+N_n)\frac{N_1\lambda_1+\cdots+N_n\lambda_n}{N_1+\cdots+N_n}(I+C)\Bigr) \\ &= \exp\bigl(-N\lambda_{\mathrm{ave}}(I+C)\bigr) \end{aligned} \tag{4}$$

と 1 つの式で書き表すことができ，結局，全ノードが同じ故障率 λ_{ave} を持っているのと等価である．

状態 i から i'，i' から i，0 から 0 への遷移確率式では同様の議論が可能である．

次に，状態 i' から i'，i' から 0 への遷移確率式 $P_{i'i'}$ と $P_{i'0}$ について検討する．

故障率に異なりのないシステムでは，状態 i' において同時に故障しているノード数を n_f として両者の確率は

$$P_{i'i'} = 1-\exp\bigl(-(N-2n_f)\lambda R\bigr) \tag{5}$$
$$P_{i'0} = 1-\exp\bigl(-2n_f\lambda R\bigr) \tag{6}$$

で表される．

一方，故障率の異なるシステムでは (5) 式, (6) 式は，各ノード群において故障しているノード数を $n_{1f},\cdots,n_{nf}$ として，

$$\begin{aligned} &P_{i'i'} \\ &= 1-\exp\bigl(-R\bigl((N_1-2n_{1f})\lambda_1+\cdots+(N_n-2n_{nf})\lambda_n\bigr)\bigr) \\ &= 1-\exp\Bigl(-R\bigl(N\lambda_{\mathrm{ave}}-2(n_{1f}\lambda_1+\cdots+n_{nf}\lambda_n)\bigr)\Bigr) \end{aligned} \tag{7}$$

$$\begin{aligned} &P_{i'0} \\ &= 1-\exp\Bigl(-R\bigl(2n_{1f}\lambda_1+\cdots+2n_{nf}\lambda_n\bigr)\Bigr) \\ &= 1-\exp\Bigl(-2R(n_{1f}\lambda_1+\cdots+n_{nf}\lambda_n)\bigr)\Bigr) \\ &\quad \text{ただし } n_f = n_{1f}+\cdots+n_{nf} \end{aligned} \tag{8}$$

と書き換えられる．ここで，$n_{1f},\cdots,n_{nf}$ は実行中に決まる値であるため，指数部第 2 項は λ_{ave} で表現することができない．よって，この項は誤差項となる．しかし，同時に故障しているノード数 n_f は極めて小さいため，この誤差項はほとんど影響しないと予想される．

以上の議論より，誤差項を無視できる場合，markov モデルのパラメータに平均故障率 λ_{ave} を代入して得られる最適なチェックポインティング間隔 I が，故障率が空間的に異なる場合の解となることが示された．

3 評価

提案手法の評価にはモンテカルロ法によるシミュレーションを用いた．アプリケーションの開

表 1: シミュレーションに用いたパラメータ

故障率 λ_1	10^{-6} 件/秒
故障率 λ_2	10^{-7} 件/秒
アプリケーションの実行時間	50 万秒
チェックポインティング時間	60.0 秒
リカバリ時間	38.8 秒
シミュレーション回数	5000 回

表 2: パラメータ取得実験環境

CPU	Pentium4 Xeon 2.4GHz x 2
Memory	DDR SDRAM 2GB
Network	Gigabit Ethernet
HDD	Ultra-160 SCSI
OS	Linux 2.4.18-2SCORE
ノード数	16
チェックポイント データサイズ	500MB

始から終了までをシミュレーションし，各ノードにおける故障の発生には乱数を用いた．シミュレーションに用いたパラメータを表 1 に示す．この中で，チェックポインティング時間とリカバリ時間とはチェックポインティングデータの転送／書込に要する時間であり，表 2 の環境で測定した値を利用している．1MIR チェックポインティングでは，チェックポインティング時間とリカバリ時間はノード数に依らず一定である．

先に述べたように，平均故障率を用いたシステムの等価変換には誤差項が存在するが，その影響が無視できるかどうかを評価する．すなわち，markov モデルと平均故障率を用いて導出した最適チェックポインティング間隔と，実際のそれとの差が無視できるほど小さいかどうかを検討した．

全ノード数 $N = 32$，そのうち 24 ノードが故障率 λ_1，8 ノードが λ_2 であるシステムにおいて評価を行った．このシステムでは，システムの平均故障率 $\lambda_{\mathrm{ave}} = 7.45 \times 10^{-7}$ 件/秒となり，全ノードの故障率がこの λ_{ave} であると仮定したときに，markov モデルから求めた最適チェックポインティング間隔 $I = 2207$ 秒（約 37 分）である．

チェックポインティング間隔を様々に変えてシミュレーションした結果を図 3 に示す．チェックポインティング間隔 2000 秒付近で最小値をとっており，λ_{ave} から求めた間隔 I とほとんど一致した．

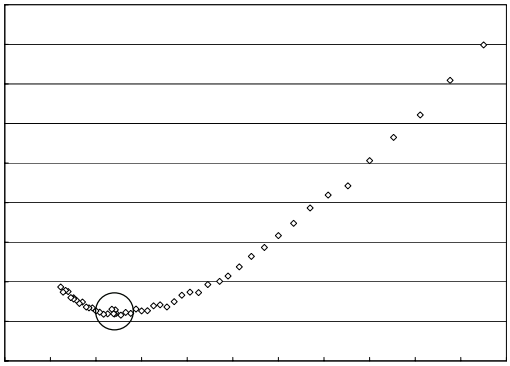

図 3: シミュレーション結果．丸印は morkov モデルでの最小値

4 まとめ

大規模化が進む計算機クラスタでは今後，システム内の均一性を前提とすることが現実的でなくなる．そこでノードごとに故障率が異なるシステムにおいて，チェックポインティング間隔の最適化により効率の良い 1MIR チェックポインティング手法を提案した．シミュレーションによりこの手法の妥当性を示した．故障率の非均一性を空間から時間に拡張し，実行中に故障率が変動するシステムの高信頼化は今後の課題である．

参考文献

[1] www.pccluster.org

[2] J. S. Plank, "Improving the performance of coordinated checkpoints on networks of workstations using RAID techniques," Proc. of SRDS'96, pp.76-85, 1996.

[3] H. Nakamura, T. Hayashida, M. Kondo, Y. Tajima, M. Imai, and T. Nanya, "Skewed Checkpointing for Tolerating Multi-Node Failures", Proceedings of IEEE SRDS'04, pp.116-125 , Oct. 2004

[4] 東美和子、近藤正章、今井雅、中村宏、南谷崇、"空間的・時間的な故障率の変動を考慮したチェックポインティング手法の初期検討", 信学技報 DC2005-14, pp.7-12, 2005