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，近年広く利用されている．科学技術計算で利用される大規模な HPC（High Performance Computing）クラスタシステムにおいては，構成要素となる商用既製品の数が多く，障害発生率も大きくなるにもかかわらず，現在までは HPC クラスタシステムの信頼性についてはあまり考慮されてこなかった．

大規模科学技術計算においては，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く，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．しかし従来のハードウェア構成に冗長性を導入する高信頼化技術は，高いコストパフォーマンスというクラスタの利点を損なう．したがってシステムソフトウェアにより高い信頼性を実現することが必要不可欠である．

代表的なソフトウェアによる高信頼化技術としてチェックポインティングがあり，例えば SCore クラスタシステムにおいて実現されているが，チェックポインティングに要する時間が長いと実効性能の低下を招く．従って，クラスタシステムにおいて高速なチェックポインティングを実現することが重要となる．

本研究では，高信頼 HPC クラスタシステムの実現を目指し，そのためにまず高速チェックポインティング機構の実現を目指す．しかしながらクラスタシステムは構成要素に商用既製品が多く用いられ，チェックポインティング機構の速度を理論式だけで検討することは現実的ではない．そこで，クラスタシステムにおけるチェックポインティングの問題点を整理するために，ソースが一般に公開されている SCore クラスタシステムをまず取り上げ，そのチェックポインティング機構を検証し，その速度の理論式と実際値を比較検討することで，性能上の問題点を分析を開始する．

2　SCore

2.1　SCore Cluster System

SCore Cluster System Software[1] は Real World Computing Partnership [2] により開発された，ワークステーションおよび PC クラスタ用の高性能並列プログラミング環境である．

SCore クラスタシステムは多種にわたるネットワークインターフェイスをサポートする高性能通信ライブラリ *PM II*，プロセス間通信に PM II を用いた高性能 MPI ライブラリ *MPICH-SCore*，クラスタのリソースを利用するユーザレベルのグローバルオペレーティングシステム *SCore-D* 等のコンポーネントから構成される．PM II は複数のネットワークインターフェイスを同時に用いることにより通信バンド幅を向上させる Network Trunking 機構を備えている．

SCore-D はチェックポインティングの中心コンポーネントとなる．全てのユーザプログラムは，時間間隔を指定するだけで SCore-D により一貫したチェックポイントが提供される．SCore-D のチェックポインティングは，全てのプロセスが同期をとって実行する coordinated-checkpointing で

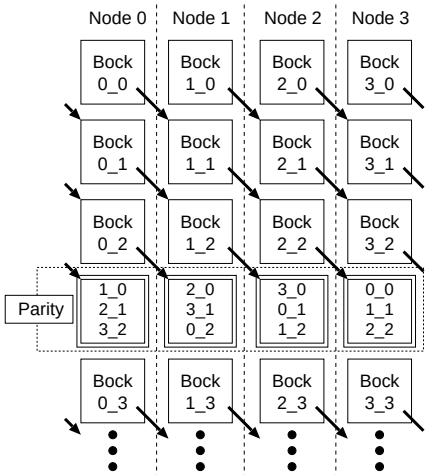

図 1: パリティの生成

あるため，チェックポインティング時には全てのプロセス間通信は停止される．

2.2 SCore におけるチェックポインティング機構

SCore ではチェックポイントデータは各ノードのローカルディスクに並列に保存される．そのためノードに永久障害が発生した場合，故障ノードが保持していたデータも失われてしまう．SCore では別ノードにパリティを保存することでデータの冗長性を確保し，この問題に対処している．図 1 にノード数が 4 の場合のパリティ生成方式を示す．各ノードが並列に以下のステップを実行してパリティが生成される．

step1 チェックポイントデータを小さなブロックに分割．

step2 最初のブロックをとなりのノードに転送．

step3 受信したブロックと自分が保持する第 2 ブロックとのパリティ演算．

step4 生成されたパリティブロックを更にとなりのノードに転送．

step5 step3 と step4 を *(N-2)* 回繰り返す．（総ノード数を N とする）

$$T_t = T_n + T_d + T_p \tag{1}$$

$$T_n = \frac{D_c}{B_n} \tag{2}$$

$$T_d = \frac{\frac{N}{N-1} \cdot D_c}{B_d} \tag{3}$$

$$T_p = \frac{1}{T_x} \times \frac{1}{N-1} \cdot D_c \times (N-2) \tag{4}$$

$$T_t = (\frac{1}{B_n} + \frac{1}{B_d} + \frac{1}{T_x}) \times D_c + \frac{1}{N-1}(\frac{1}{B_d} - \frac{1}{T_x}) \times D_c \tag{5}$$

D_c	ノード当りのパリティを含まない チェックポイントデータ量 [MB]
N	ノード数
T_n	データ送受信に要する時間 [sec]
T_d	ディスク書き込みに要する時間 [sec]
T_p	パリティ演算に要する時間 [sec]
T_x	パリティ演算のスループット [MB/sec]
B_n	ネットワークのバンド幅 [MB/sec]
B_d	ディスク書き込みのバンド幅 [MB/sec]

図 2: チェックポインティングに要する時間の推定

step6 受信したブロックを最後のノードのローカルディスクに保存．

step7 全てのチェックポイントデータに対して，step2 から step6 を同様に実行する．

このアルゴリズムにおいて，*(N-1)* 個のチェックポイントデータブロックに対して，*(N-2)* 回のパリティ演算により一つのパリティブロックが生成される．また，あるノードがとなりのノードに転送するデータ量の合計は，総ノード数に依らず 1 ノード当りのパリティを含まないチェックポイントデータサイズに等しい．

2.3 チェックポインティングに要する時間の推定

チェックポインティングに要する時間は，ブロック送受信のための通信にかかる時間（T_n），チェックポイントデータをローカルディスクに書き込むのに要する時間（T_d），およびパリティ演算に要する時間（T_p）の 3 つの要素から主に構成され，式 (1) で与えられる．

1 ノードは最終的に全ての自分のチェックポイントデータをとなりのノードに送信するので，ブ

ロック送受信に要する時間：T_n はパリティブロックを含まないチェックポイントデータ量をネットワークのバンド幅で割った値となる．(式 (2)) 各ノードはパリティを含むチェックポイントデータをディスクに書き込むのでディスク書き込みに要する時間：T_d は一ノード当りの全てのデータ量をディスク書き込みのバンド幅で割った値となる．(式 (3)) パリティブロックの総データ量は $\frac{1}{N-1}D_c$ であり，一つのパリティブロック生成のために *(N-2)* 回のパリティ演算が必要なので，パリティ演算にかかる時間：T_p は式 (4) で表される．式 (1)-式 (4) より一回のチェックポインティングに要する時間は式 (5) で与えられる．

2.4 故障検出・回復機構

SCore-D はサーバノードが計算ノードに巡回させたメッセージの返答をタイムアウト期間内に受け取らなかった場合に障害発生と判断し，回復処理を開始する．

回復処理ではまず全ての計算ノードを再検査する．障害が一時的で故障ノードが再び応答するようになったら，ユーザプログラムは単に最後のチェックポイントから再開される．故障ノードが依然応答しない場合は永久障害と判断され，障害ノードは予備ノードと交換されなければならない．

各計算ノードでメモリイメージがチェックポイントから再構成されるとプログラムは再開する．予備ノード上で再開する場合は，他の正常なノードのチェックポイント，およびパリティデータからメモリイメージを再構成する必要がある．

3 実験

3.1 実験方法

表 1 に示される環境において SCore のチェックポインティング機構の評価を行った．実験には Nas Parallel Benchmark の IS を用いた．

実験では一回のチェックポインティングに要する時間を，ワーキングセットサイズおよびノード数を変化させて測定した．また回復処理に要する時間の計測も行った．以降，一回のチェックポインティングに要する時間を CP 時間と表記する．

表 1: Platform Specification

# of nodes	8
CPU	Pentium4 Xeon 2.4GHz
Memory	DDR SDRAM 2048MB
Network	Gigabit Ethernet x2 PM/Ethernet Trunking
HDD	Ultra-160 SCSI
OS	Linux kernel 2.4.18-2SCORE SCore 5.2.0

3.2 実験結果

図 3 に Gigabit Ethernet を用いた場合の，ノード数をパラメータとしたノード当りのチェックポイントデータ量（図 2 の D_c に相当）と CP 時間の関係を示す．この結果より，CP 時間はデータ量に比例するが，データ量が同一の場合，CP 時間はノード数に依らずほぼ一定であることが分かる．これは SCore のチェックポインティングアルゴリズムはノード数に対してスケーラブルであることを示す．

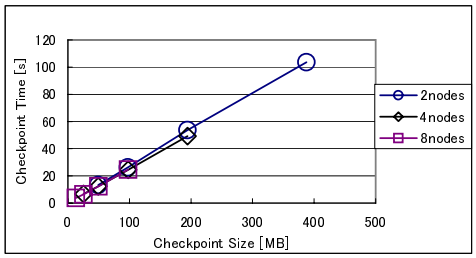

図 3: CP 時間とデータ量の関係

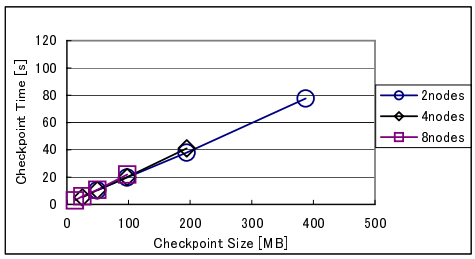

図 4: ディスク書き込みとパリティ演算を行わない場合の CP 時間とデータ量の関係

しかし CP 時間はかなり長く，例えばデータ量が総メモリ量の 4 分の 1 以下にすぎない 400MB の場合でも約 100 秒かかっている．またデータ量が同一の場合，ノード数が少ない方がわずかに CP 時間が長いことも分かる．これはノード数が少ない方が，$\frac{D_c}{N-1}$ で与えられるパリティデータ量が大きいので，式 (3) のディスク書き込みに要する時間 T_d が長くなるからである．

次に，ディスク書き込みとパリティ演算の影響を排除するために，SCore-D をディスク書き込みを行わずパリティ演算も行わないように変更した．変更後は T_d と T_p は 0 となり，CP 時間 T_t は次式で与えられる．

$$T_t = \frac{D_c}{B_n} \tag{6}$$

結果を図 4 に示す．図 3 の結果と比較すると 20% ほどしか時間が短縮されていないことがわかる．この結果より，主にネットワークバンド幅 B_n がボトルネックであることが分かる．2 つの図から，実効ネットワークバンド幅は 5[MB/s]（=40[Mbps]）と推定される．この値は Gigabit Ethernet のピーク性能と比較するとはるかに小さい．これは図 1 におけるブロックサイズが小さすぎるために，ネットワーク性能を生かしきれていないためと考えられる．ブロックサイズは SCore-D により動的に決定されるが，一度に転送可能なデータサイズは 1400byte に制限されている．

また，この実験では Network Trunking 機構を用いて二つのネットワークインターフェイスを使用しているが，Network Trunking 機構ではラウンドロビン方式に各ネットワークインターフェイスに交互にパケットを割り振る．そのため，パケットサイズを越える転送データ量でないとその効果は現れないと考えられる．PM ドライバが用いるパケットサイズは 1400byte で転送時のブロックサイズと同一であるため，Network Trunking 機構は有効に働いていないと考えられる．

3.3 回復処理

図 5 はあるノードに永久障害が発生した場合の回復処理に要する時間を示す．回復処理に要する時間とは障害が検出されてから SCore-D が正常なノードのチェックポイントデータ，およびパリティデータから障害ノードのチェックポイントを回復し，メモリイメージを再構築してプログラムの実行が再開されるまでの時間を意味し，障害検出にかかる時間は含まれない．

図 5 から，回復処理に要する時間は数分から十数分とかなり長いこと，及びノード数が少ないほど長くなることが分かる．これはノード数が少ないほどパリティブロックのデータ量が大きく，その読み出しに時間がかかるからである．

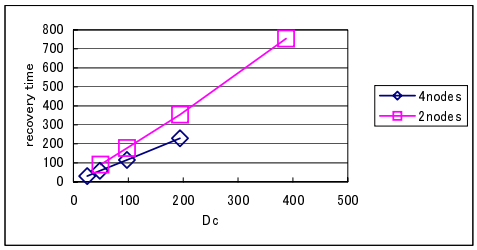

図 5: 回復処理に要する時間

4 結論

本稿では，SCore におけるチェックポインティング機構の性能解析とその実験結果を示した．SCore のチェックポインティング機構は，チェックポインティングに要する時間をノード数に関わらず一定にできるため，大規模なクラスタシステムにも適用できるという利点を持つ．一方，低い実効ネットワークバンド幅のために一回のチェックポインティングに非常に長時間かかるという欠点を持つことも明らかになった．

来年度以降は，この問題について他の環境での比較実験や SCore-D の改造により更に調査を進め，我々の最終目標である，高信頼 HPC クラスタシステムの実現を目指したい．

参考文献

[1] http://www.pccluster.org

[2] http://www.rwcp.or.jp