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．

クラスタシステムの利点である優れた高いコストパフォーマンスを維持するためには，ハードウェアの冗長化は最小限に抑え，システムソフトウェアによる冗長度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るが，一方で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．そこで，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．

チェックポインティングを提供するシステムソフトウェアとして，SCore クラスタシステムがある．本年度は，SCore が提供するチェックポインティング機構の性能上の問題点を分析し，その結果を踏まえて改良を行った．また，評価尺度として performability を用いてその改善度を定量的に評価した．

2　SCore

2.1　SCore Cluster System

SCore Cluster System Software[1] は Real World Computing Partnership 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．SCore-D はチェックポインティングの中心コンポーネントであり，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　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を各ノードのローカルディスクに保存すると共にパリティを別ノードに保存することでデータの冗長性を確保している．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．

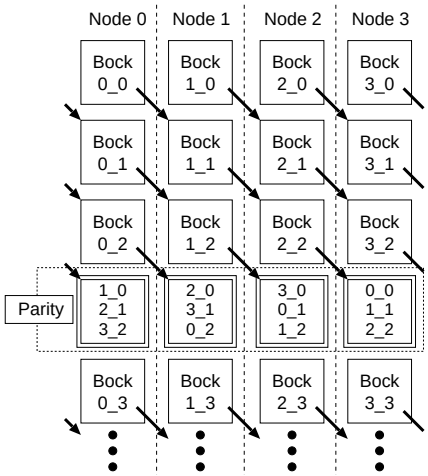

図 1: パリティの生成

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）

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は，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.

チェックポイント時間の理論式 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（Tc_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（Tc_d），およびパリティ演算に要する時間（Tc_p）の 3 つの要素から主に構成され，式 (1) で与えられる.

$$Tc_t = Tc_n + Tc_d + Tc_p \tag{1}$$

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

$$Tc_d = \frac{\frac{N}{N-1} \cdot D_c}{B_{dw}} \tag{3}$$

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

$$Tc_t = (\frac{1}{B_n} + \frac{1}{B_{dw}} + \frac{1}{B_x}) \times D_c + \frac{1}{N-1}(\frac{1}{B_{dw}} - \frac{1}{B_x}) \times D_c \tag{5}$$

D_c	checkpoint data per node without parity [MB]
N	the number of nodes
Tc_n	time for network transfer[sec]
Tc_d	time for disk write [sec]
Tc_p	time for parity operation [sec]
B_x	parity throughput [MB/sec]
B_n	bandwidth of network [MB/sec]
B_{dw}	bandwidth of disk write [MB/sec]

図 2: チェックポインティングに要する時間の理論式

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：Tc_n は式 (2)）で与えられる．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：Tc_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：Tc_p は式 (4) で表される．以上より，一回のチェックポインティングに要する時間 Tc_t は式 (5) で与えられる.

2.3 回復機構

障害発生時には，障害ノードは予備ノードと交換されなければならないが，予備ノード上に，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

$$Tr_t = Tr_n + Tr_d + Tr_p + Tr_g \quad (6)$$

$$Tr_n = \frac{D_c}{B_n} \quad (7)$$

$$Tr_d = \frac{\frac{N}{N-1} \cdot D_c}{B_{dr}} \quad (8)$$

$$Tr_p = \frac{1}{B_x} \times \frac{1}{N-1} \cdot D_c \times (N-1) = \frac{1}{B_x} \cdot D_c \quad (9)$$

$$Tr_g = \frac{D_c}{B_n} \quad (10)$$

$$Tr_t = (\frac{1}{B_n} + \frac{1}{B_{dr}} + \frac{1}{B_x}) \times D_c + \frac{1}{N-1} \cdot \frac{1}{B_{dr}} \times D_c \quad (11)$$

D_c	checkpoint data per node without parity [MB]
N	the number of nodes
Tr_n	time for network transfer[sec]
Tr_d	time for disk read [sec]
Tr_p	time for parity operation [sec]
Tr_g	time for gathering recovered data [MB/sec]
B_x	parity throughput [MB/sec]
B_n	bandwidth of network [MB/sec]
B_{dr}	bandwidth of disk read [MB/sec]

図 3: 回復処理に要する時間の理論式

がある．このアルゴリズムは以下のようになる．まず全てクリアされたデータを予備ノード上に用意する．次に 2.2 節の **(step 2)** から **(step 5)** を，予備ノードも含めて実行する．その後，各ノードは受け取ったデータと自分が保持するパリティデータの間のパリティ計算を行う．その結果が障害ノードのチェックポイントデータなのでそれを予備ノードに転送する．

回復時間の理論式　2.2 節と同様に考えると図 3 の Tr_t として与えられる．

3　SCore の改良とその有効性

3.1　実装上の問題とその改良

現在の SCore の性能上の問題を分析するために、図 1 に示すクラスタ上で，ノード数，チェックポイントデータ量を変化させてチェックポイントに要する時間を測定した．また，その時間の内訳を分類するために，パリティ計算のみ省略，パリティ計算とネットワーク転送を省略，した場合に要する時間も測定することで図 2 の Tc_n，Tc_d，Tc_p を特定し B_x，B_n，B_{dw} を推定した．その結果，Tc_p は無視できるほど小さいこと，B_{dw} は 30MB/sec 程度で妥当である一方，Tc_n はわずか 80Mbps でありピーク性能の 1Gbps に比べ遥かに低いことがわかった．この原因は，ネットワーク転送のデータサイズが，PM II の最大転送サイズである 1400B と等しいためであることがわかった．そこで転送サイズを 10 倍の 14000B にしたところ，Tc_n が 400Mbps まで改善した．

表 1: 実験に用いたクラスタの仕様

# of nodes	8
CPU	Pentium4 Xeon 2.4GHz
Memory	DDR SDRAM 2048MB
Network	Gigabit Ethernet (1Gbps in peak)
Switch	Netgear GS524T
HDD	Ultra-160 SCSI
OS	Linux kernel 2.4.18-2SCORE SCore 5.2.0

図 4 に，ノード数 4，チェックポイントデータサイズ 500MB の時に，改良前後で 1 回のチェックポイントに要する時間の内訳を示す．図中の “Original” は改良前，“Modified” は改良後を表わす．この図からわかるように，ネットワーク転送の改良で総実行時間が半分以下に短縮していること，改良前はネットワーク転送が大半の時間を締めていたが，改良後はディスク書き込みの方が総実行時間に締める割合が大きく，更なる改良にはディスク I/O の改良が必要であることがわかる．

図 5 に，ノード数 4 の時に，回復処理に要する時間を示す．この図からわかるように，ネットワーク転送の改良により，回復処理に要する時間が約 1/4 に短縮していることがわかる．

3.2　改良による信頼性向上

前節の改良がシステムの信頼性に与える影響を評価するために，システムの availability と実効性能を合わせた指標である performability [2] を用いる．直感的には，performability は性能と availability の積の期待値であり，定期的な間隔でチェックポインティングを行う場合には，performability

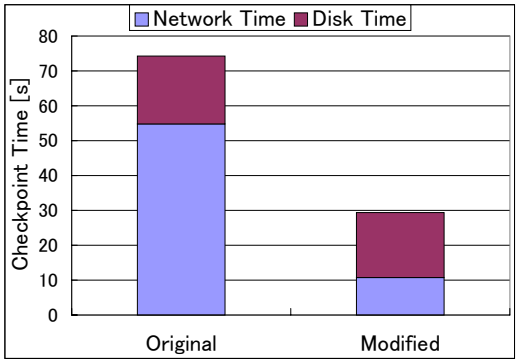

図 4: チェックポイントに要する時間の内訳

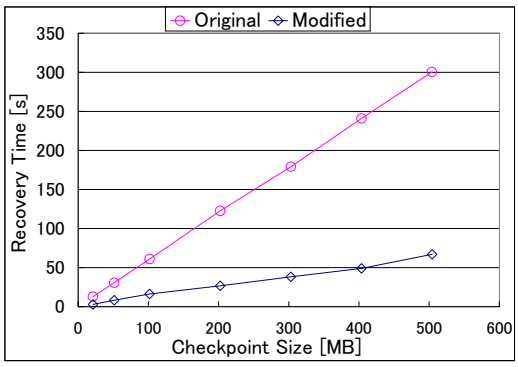

図 5: 回復処理に要する時間 (ノード数:4)

は以下の式 12 で与えられる [3]．ここで，CP_itvl と CP_time は各々，チェックポイント間隔時間と，チェックポインティングに要する時間を表わす．また，$MTTR'$ は障害発生時のロールバックで失われる時間を含み，“$\frac{CP\_itvl}{2} + Recovery\ time$” で与えられる．

$$Performability = \frac{CP\_itvl}{CP\_itvl + CP\_time} \times \frac{MTBF}{MTBF + MTTR'} \quad (12)$$

図 6 に，各ノードの MTBF を 5 年 (=43800 時間) とした場合の，ノード数 1024 からなるクラスタを想定し，チェックポイントデータ量が 1GB と仮定した場合の，改良前後の performability を示す．横軸はチェックポインティングを実行する間隔を表わす．この図から，チェックポインティング間隔が短すぎると実効性能が低下することにより performability が低下すること，チェックポイ

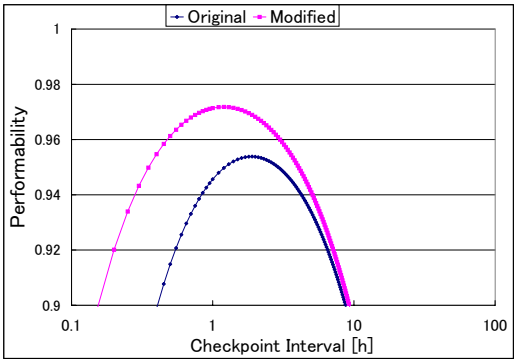

図 6: Performability の向上 (ノード数:1024，1 台の MTBF:5 年)

ンティング間隔が長すぎると障害発生時のロールバックが大きくなり，$MTTR'$ が大きくなるためにやはり performability が低下すること，がわかる．また，改良前後を比較すると，チェックポインティングを高速化することで，性能への悪影響を抑えながらチェックポインティング間隔を短くすることができ，performability の向上をもたらすことがわかった．

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改良，およびその改良による信頼性の向上を定量的に示した．来年度以降は，この改良で得た知見を元に，さらなる高信頼化手法を提案し，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] J.F. Meyer, “On Evaluating the Performability of Degradable Compting Systems”, IEEE Transactions on Computers, Vol.C-29, No. 8, pp.720–731, Aug. 1980.

[3] M.Kondo, T.Hayashida, M.Imai, H.Nakamura, T.Nanya, and A.Hori ”Evaluation of Checkpointing Mechanism on SCore Cluster System”, IEICE Trans. on Inf. & Syst, Vol.E86-D, No.12, pp.2553-2562, 2003