ハイパフォーマンス
コンピューティング 93－5
（2003．3．11）

SCoreクラスタシステムにおけるチェックポインティング機構の性能評価

林田卓朗[†] 鈴木優介[†] 近藤正章[†,††]
今井雅[†] 中村宏[†]
南谷崇[†] 堀敦史[†††]

クラスタシステムは，その高いコストパフォーマンスから近年広く活用されている．大規模なクラスタシステムでは構成要素となる商用既製品の数が多く，システムの故障率も大きくなる．しかしこれまでは大規模クラスタシステムの信頼性についてはさほど考慮されていなかった．長時間に及ぶ大規模科学技術計算においてクラスタシステムを活用するためには，システムソフトウェアによりシステムの信頼性を向上することが必要不可欠となっている．

SCoreクラスタシステムはチェックポインティング機能を備える公開された高性能並列プログラミング環境である．そこで我々はクラスタシステムの高信頼化における問題点を整理するために，SCoreのチェックポインティング機構を定量的に評価した．その結果，チェックポインティングに要する時間はノードの総数に依らずほぼ一定であること，しかし現在の実装では低いネットワーク転送の実効性能がチェックポインティングのボトルネックになっていることが明らかとなった．

Performance Evaluation of Checkpointing Mechanism on SCore Cluster System

Takuro HAYASHIDA,[†] Yusuke SUZUKI,[†]
Masaaki KONDO,[†,††] Masashi IMAI,[†] Hiroshi NAKAMURA,[†]
Takashi NANYA[†] and Atsushi HORI [†††]

Cluster systems are getting widely used because of good performance / cost ratio. However, little attention has been paid for their reliability so far. As the number of commodity components in a cluster system gets increased, it is indispensable to support reliability by system software.

SCore cluster system software is a parallel programming environment which is open to public and provides checkpointing mechanism. Towards highly reliable cluster systems, we evaluate and analyze the checkpointing mechanisms of SCore quantitively. The experimental results reveal that the required time for checkpointing scales very well in respect to the number of computing nodes. However, the required time is quite long the current implementation cannot utilize potential ability of networks.

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，システムの故障率も大きくなる．しかし現在までは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．従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[1),2)]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．

我々は，HPCクラスタシステムのための高速チェックポインティング機構の実現を目指している．本稿で

† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology. The University of Tokyo
†† 科学技術振興事業団 JST
††† スイミー・ソフトウェア株式会社
Swimmy Software, Inc.

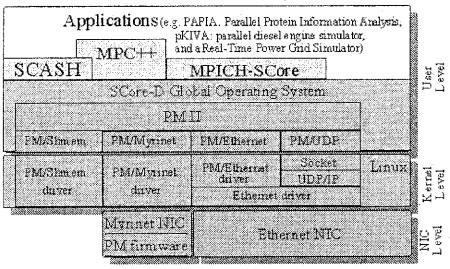

図 1 SCore Software Architecture (cited from[3])

は，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取り上げ，そのチェックポインティング機構を検証し，性能上の問題点を分析する．

2. SCore

2.1 SCore Cluster System

SCore Cluster System Software[1),2)] は Real World Computing Partnership[3)] 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．

図 1 に SCore のソフトウェアアーキテクチャを示す．SCore は *PM II* と呼ばれる高性能通信ライブラリ，*MPICH-SCore* と呼ばれる PM II 上に実装された MPI，*SCore-D* と呼ばれるグローバルオペレーティングシステム等のコンポーネントから構成される．以下にこれらについて簡単に述べる．

PM II PM II（PM-version 2）は，多くの種類のネットワークで使用可能なクラスタコンピューティング専用の高性能通信ライブラリである．Myrinet，Ethernet，UDP，共有メモリインターフェース用の PM II ドライバが実装されている．

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ネットワークバンド幅を向上させる Network Trunking 機構を備えている．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ため，パケットサイズ☆を越えるデータ量を一度に転送しないとその効果は現れないと考えられる．

MPICH-SCore MPICH-SCORE は MPICH をベースとした，プロセス間通信に PM II を用いた MPI ライブラリである．

SCore-D SCore-D はプロセッサやネットワークのようなクラスタのリソースを管理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である．PM

☆ PM/Ethernet ドライバが用いるパケットサイズは 1400byte である

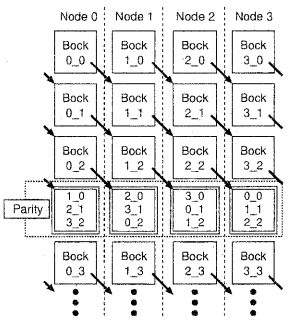

図 2 ノード数が 4 の場合のパリティの生成方法

ネットワークデバイスは SCore-D によって管理され，利用可能となる．

SCore-D はチェックポインティングの際に中心的な役割を果たす．全てのユーザプログラムは，時間間隔を指定することで SCore-D により一貫したチェックポインティングが行われる．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[4)] 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 SCore におけるチェックポインティング機構

SCore では各ノードのチェックポイントデータはそれぞれのノードでローカルディスクに並列に保存される．そのため，あるノードに永久障害が発生した場合，障害が発生したノードが保持していたチェックポイントデータは失われてしまう．SCore では他のノードにチェックポイントのパリティデータを保存することによりデータの冗長性を確保し，この永久障害の問題に対処している．

図 2 にノード数が 4 の場合のパリティ生成方式を示す．縦の一列が一つのノードのパリティを含むチェックポイントデータを示す．各ノードが並列に以下のステップを実行することによりパリティブロックが生成される．

- （Step1）チェックポイントデータを小さなブロックに分割する．
- （Step2）最初のブロックを隣りのノードに転送する．
- （Step3）受信したブロックと自分が保持する次のブロックとのパリティ演算を行う．
- （Step4）生成されたパリティブロックを更に隣りのノードに転送する．
- （Step5）Step3 と Step4 を *(N-2)* 回繰り返す．但し総ノード数を N とする．
- （Step6）受信したブロックをパリティブロックとして最後のノードのローカルディスクに保存する．
- （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，あるノードが隣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（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，図 3 中の式 (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隣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 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 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．

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

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

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

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

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

D_c	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
Tc_n	データ送受信に要する時間 [sec]
Tc_d	ディスク書き込みに要する時間 [sec]
Tc_p	パリティ演算に要する時間 [sec]
B_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_{dw}	ディスク書き込みのバンド幅 [MB/sec]

図 3 チェックポインティングに要する時間の推定

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障害発生前と同じノード上で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，故障ノードは予備ノードと交換される．

各計算ノードでチェックポイントからメモリイメージが再構成されるとユーザプログラムは再開する．予備ノード上で再開する場合は，まず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故障ノードが保持していたチェックポイントデータを回復する必要がある．

故障ノードのチェックポイントデータ回復処理は以下の手順で実行される．まず，代替ノードに全てのデータが 0 のチェックポイントデータを保持させる．次にパリティ生成処理の（Step2）〜（Step5）を代替ノードを含めた全ノードで同様に実行する．最後のノードで受信したブロックと，そのノードに保存されているパリティブロックとのパリティ演算を行うと，生成されたブロックは故障ノードが保持していたチェックポイントブロックとなる．回復されたブロックは代替ノードに順次転送される．

回復処理に要する時間 Tr_t は，チェックポインティングにかかる時間の推定とほぼ同様に定式化できる（図 4）．まずブロック送受信のための通信にかかる時間（Tr_n），チェックポイントデータをローカルディスクにから読み込むのに要する時間（Tr_d），およびパリティ演算に要する時間（Tr_p）の 3 つの要素に加え，回復されたブロックを代替ノードに集めるのに要する時間（Tr_g）から構成され，式 (6) で与えられる．

Tr_n，Tr_d についてはチェックポインティング処理の場合と全く同様である（式 (7)，式 (8)）．一つのパリティブロックから *(N-1)* 回のパリティ演算により，一つのチェックポイントデータブロックが回復されるので，Tr_p は式 (9) で表される．回復されたチェックポイントブロック全てが代替ノードに転送されるので Tr_g は，式 (10) で表される．式 (6)〜式 (10) より，回復処理に要する時間は式 (11) で与えられる．

図 4 の各式で示した時間は，回復処理に要する時間のうち，故障ノードが保持していたチェックポイントの回復処理にかかる時間だけを定式化したものである．実際には回復処理にはこれ以外にもチェックポイントデータからのメモリイメージの回復処理等多くの要素が関わる．

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

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

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

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

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

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

Tr_n	データ送受信に要する時間 [sec]
Tr_d	ディスク書き込みに要する時間 [sec]
Tr_p	パリティ演算に要する時間 [sec]
Tr_g	回復されたブロックを転送するのに要する時間 [sec]
B_{dr}	ディスク読み込みのバンド幅 [MB/sec]

図 4 回復処理に要する時間の推定

表 1 Platform Specification

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

3. 実　　験

3.1 実験方法

我々は表 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[6] の IS を用いた．SCore ではユーザアプリケーションの使用メモリ量がチェックポイントデータサイズに比例する．ここで IS を用いたのは，IS は実行中にメモリ使用量がほとんど変化しないため，チェックポイントデータサイズが毎回一定の値となり，今回の実験に適しているためである．

本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．比較のためにネットワークは Gigabit Ethernet と Fast Ethernet のそれぞれにおいて計測を行った．また回復処理に要する時間についても

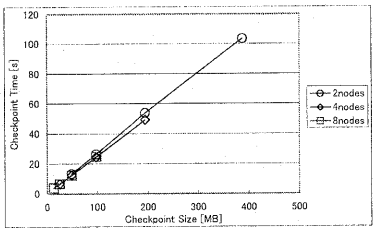

図 5 チェックポイントデータサイズとチェックポインティングに要する時間との関係

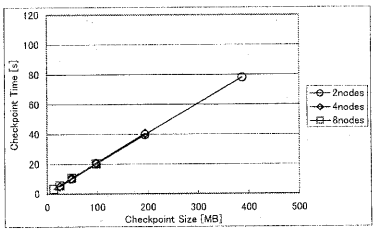

図 6 ディスク書き込みを行わない場合のチェックポイントデータサイズとチェックポインティングに要する時間との関係

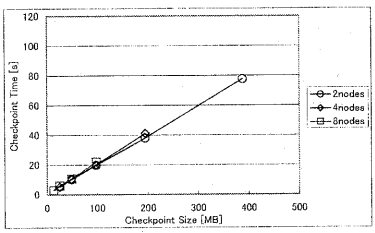

図 7 ディスク書き込みとパリティ演算を行わない場合のチェックポイントデータサイズとチェックポインティングに要する時間との関係

計測した．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．

3.2 チェックポインティング処理に要する時間

図 5 に Gigabit Ethernet を用いた場合の，各ノード数におけるノード当りのチェックポイントデータ量（図 3 の D_c に相当）と CP 時間の関係を示す．この結果より，CP 時間はデータ量にほぼ比例しているが，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．

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．クラスタシステムを実際に利用する際には，多くの場合総メモリ量のほとんどを使用する．例えばメモリサイズが 2GB の場合では，一回のチェックポインティングに 7 分以上の時間がかかり，その間ユーザプログラムの処理が停止してしま

うことになる．

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 Tc_d が長くなるためであると考えられる．

次にディスク書き込みの影響を排除することで，ディスク書き込みに要する時間を推定するために，SCore-D をディスク書き込みを行わないように変更して実験を行った．変更後は Tc_d は 0 となり，CP 時間 Tc_t は式 (5) より，次式で与えられる．

$$Tc_t = (\frac{1}{B_n} + \frac{1}{B_x}) \times D_c - \frac{1}{N-1}\frac{1}{B_x} \times D_c \quad (12)$$

変更後の実験結果を図 6 に示す．図 5 の変更前の結果と比較して CP 時間は 2 割程度減少していることが分かる．この差がチェックポインティングに要する時間のうちのディスク書き込みに要する時間に相当する．この結果より実際のディスク書き込み性能はおおよそ 30[MB/s] と見積もられる．この値はハードウェア環境から実効ディスク性能を考慮すると妥当な値であると考えられる．

また，データ量が同一の場合のノード数に依存した CP 時間の差はより小さくなっていることが分かる．これは，式 (12) において明らかに B_x は B_n と比較して極めて大きく，第二項は無視でき，ノード数 N に依存しない第一項が支配的となるからである．

次にパリティ演算のスループットとネットワークバンド幅をより定量的に評価するために，更に SCore-D をパリティ演算を行わないように変更して実験を行った．各計算ノードはパリティ演算を行うことなくデータブロックをそのまま転送する．この時，式 (4) の Tc_p は 0 になり，式 (12) より Tc_t は以下の式で与えられる．

$$Tc_t = \frac{D_c}{B_n} \quad (13)$$

結果を図 7 に示す．ディスク書き込みのみを行わない図 6 の結果と比較すると，ほとんど性能向上が見られない．これらの結果より，パリティ演算に要する時間はわずかであると考えられる．ゆえにネットワークバンド幅 B_n がチェックポインティングにおける主なボトルネックであることが分かる．

また，この結果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な値である．これは図 2 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活かしきれていないことが理由として考えられる．ここでブロックサイズは SCore-D により動的に決定されるが，その上限は 1400byte に制限されている．

ネットワークに Fast Ethernet を用いた場合の同様の実験結果を図 8 に示す．図 5 の結果と比較してネットワークのピーク性能は 10 分の 1 であるにも関わらず，CP 時間はほとんど増加していない．この結果も現在の SCore のチェックポインティング実装は，ネットワーク性能を有効に活用できていないことを示している．

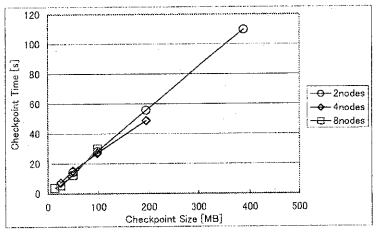

図 8 Fast Ethernet を用いた場合のチェックポイントデータサイズとチェックポインティングに要する時間との関係

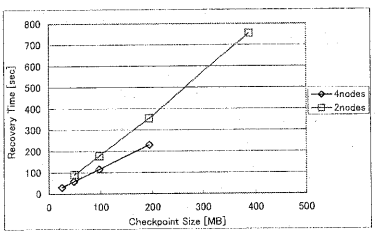

図 9 チェックポイントデータサイズと回復処理に要する時間との関係

3.3 回復処理に要する時間

図 9 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する．障害発生から SCore-D が障害を検出するまでの時間，および SCore-D を再起動するのに要する時間は含まれない．図 9 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が分かる．チェックポインティング処理は正常稼働中に頻繁に行う必要がある割り込み処理であるのに対して，回復処理は永久障害が発生したときのみに行われるものである．そのためこの回復処理に要する時間がクラスタシステムのパフォーマンスに与える影響は少ないと考えられる．

また，ノード数が少ないほど回復処理に要する時間は長くなる傾向にあることが分かる．これは，式 (11) で示されるように，ノード数が少ないほど同じチェックポイントデータサイズに対して生成されるパリティブロックのデータ量が大きくなるため，パリティデータをディスクから読み出すのに要する時間が長くなるからであると推測される．

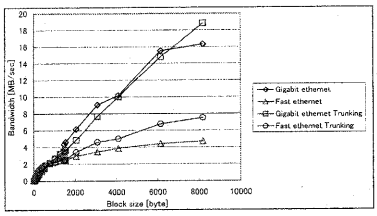

図 10 ping-pong 転送を行った場合のネットワーク性能

3.4 考　　察

本実験環境における SCore のネットワーク性能を定量的に評価するため，2 ノード間でピンポン転送によるメッセージ交換を行った場合のラウンドトリップタイムを計測した．メッセージサイズとバンド幅との関係を図 10 に示す．

この結果から SCore-D がチェックポインティングの際にデータ転送のブロックサイズとして用いている 1400byte でのメッセージ交換では十分なネットワーク性能が出ていないことが分かる．1400byte のメッセージサイズで Gigabit ethernet を用いた場合のネットワークバンド幅は約 5[MB/sec]，Fast ethernet の場合は 2.5[MB/sec] 程度となっていることが分かった．これらの値は 3.2 節までの実験結果から得られる値とほぼ一致するものである．

また，図 10 の結果からデータ転送時のデータサイズを大きくすることで，チェックポインティングにおけるネットワークの実効バンド幅を改善できることが予測される．例えば 8000byte のブロックサイズで Gigabit Ethernet を用いる場合，16[MB/sec] のバンド幅が得られ，ネットワーク転送に要する時間は現在の 3 分の 1 程度になると期待される．現在の SCore の実装では，チェックポインティング時には Network Trunking を利用できるようにはなっていないが，更にこれを利用できれば約 18[MB/sec] までの性能向上が期待できると考えられる．

ネットワークバンド幅が改善されれば，ディスク書き込みもまたボトルネックとなり得ることも予測される．例えばノード数が 2 でチェックポイントデータサイズが 400MB の場合，ディスク書き込みには図 5 の結果と，図 6 の結果を比較することにより，25[sec] 程度かかっていることが分かる．仮にネットワークバンド幅が 18[MB/sec] に改善された場合，データ転送には 22[sec] かかることになる．この場合，よりディスク書き込みに多く時間がかかることになり，チェックポインティングにおけるボトルネックとなる．ゆえに高速なチェックポインティング機構の実現のためにはディスク書き込みにかかる時間を削減することも課題である．

4. 結　　論

本稿では *SCore Cluster System Software* におけるチェックポインティング機構の性能評価を行った．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，大規模クラスタシステムにも適用できる利点を持つ．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．実効ネットワークバンド幅はブロックサイズを最適化することにより改善される可能性があることを示した．

また回復処理に要する時間については，現時点では不明確ではあるがディスク読み込み性能の影響が大きく，パリティデータ量がより大きくなるノード数が少ない場合の方が長時間必要であることも分かった．

今後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る予定である．同時に，高い実効ネットワークバンド幅が得られるようチェックポインティング機構の改善に取り組み，その性能評価を通して更なるチェックポインティングの高速化を目指す予定である．我々の最終目標は HPC クラスタシステムのための高速チェックポインティングを提案し，実装することである．

参 考 文 献

1) Y. Ishikawa, H. Tezuka, A Hori, S. Sumimoto, T. Takahashi, F. O'Carroll, and H. Harada, "RWC PC Cluster II and SCore Cluster System Software – High Performance Linux Cluster", In Proc. of the 5th Annual Linux Expo, pp.55–62, 1999.
2) http://www.pccluster.org
3) http://www.rwcp.or.jp
4) M. Elnozahy, L. Alvisi, Y.M. Wang, and D.B.Johnson. A Survey of Rollback-Recovery Protocols in Message-Passing Systems. Tech. Rep CMU-CS99-148, Carnegie Mellon University, June 1999.
5) T, Nishioka, A Hori, and Y. Ishikawa, "Consistent Checkpointing for High Performance Clusters", In Proc. of Internationall Conference of Cluster Computing 2000, pp.367–368, 2000.
6) D. Bailey, T. Harris, W. Saphir, R. Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0", NASA Ames Research Center Report, NAS-05-020, 1995.

謝辞
本研究の一部は，ＮＥＤＯ民間基盤技術研究支援制度（課題名「大規模・高信頼性サーバの研究」）によるものである．