4H-07

動的なノード数変更に対応した MPI並列処理のための負荷分散手法の実装

澤田 祐樹 † 大津 金光 † 大川 猛 † 横田 隆史 †
† 宇都宮大学大学院工学研究科情報システム科学専攻

1 はじめに

近年モバイル端末は高性能化しており，並列分散アプリケーションのためのプラットフォームとして新たに注目されている．我々は手軽に並列計算による大規模演算を行うために，Android OS を搭載した端末を計算ノードとして活用し，MPI アプリケーションを並列処理するクラスタシステムを開発している [1]．本システムは端末の脱退，参入に伴うノード数の動的変化に対応するためにチェックポイントデータの取得と，取得したチェックポイントデータからの並列処理のリスタートが可能である．しかし本システムでの従来のリスタート機能において，ノード数が動的に変化した場合，特定のノードが脱退したノードで行っていた並列処理を全て引き継ぐため，クラスタ内のノード間における並列処理の負荷に偏りが生じる場合があった．そこで，リスタート時に各ノードにおける並列処理の負荷をプロセス単位で分散化するプロセスの割り当て手法を開発する．本稿ではその負荷分散手法の実装について述べる．

2 Android クラスタシステム

ノード間の通信は高速な通信が可能で，多くの端末で利用できる Wi-Fi を用いる．また並列処理の基盤として Open MPI を用い，MPI アプリケーションを並列実行の対象とする．本システムは移動体であるモバイル端末を計算ノードとしているため，クラスタからのノードの脱退や，クラスタへのノードの参入により，ノードが動的に変化することが想定される．

ノードが動的変化した場合でも MPI アプリケーションの実行を保証するためにチェックポインティング/リスタート機能を採用している．チェックポインティング/リスタート機能は，Android OS が正式にサポートしている x86，ARM，MIPS の各命令セットに対応している点から Distributed MultiThreaded Checkpointing (DMTCP) を使用して実現する．DMTCP の管理下で MPI アプリケーションを実行するとホストノードでは，全プロセスを管理する dmtcp_coordinator プロセスと，MPI 並列プロセスを生成する oretrun プロセスが立ち上がる．また orterun プロセスは各リモートノードに orted プロセスを生成し，orted プロセスは，MPI 並列プロセスを生成する．現在 orterun プロセスは 30 秒ごとに dmtcp_coordinator プロセスへチェックポイント取得要求を行う．dmtcp_coordinator プロセスは各プロセス内で起動しているチェックポイントスレッドへ取得要求メッセージを発行し，チェックポイントデータを作成させる．ノードが脱退した場合は，クラスタに残っているいずれかのノードに対して脱退ノード内で作成されたチェックポイントデータを送信し，リスタート時にそのチェックポイントデータを活用することで脱退ノードの並列処理を引き継ぐことができる．

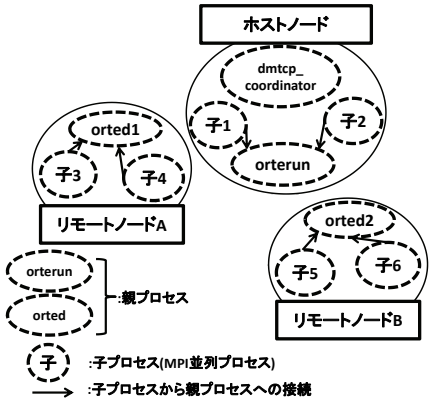

図 1: 3 ノード構成クラスタシステム

3 動的なノード数変更時の負荷分散の現状と課題

図 1 は 3 ノード構成のクラスタを示しており，各ノードでそれぞれ 2 つの MPI プロセスが実行されている．このとき，リモートノード B が脱退した場合に，脱退したノードで実行されていたプロセスをクラスタ内に残ったノードへと移動することを考える．使用している DMTCP (バージョン 2.3.1) ではノード単位の割り当てとなり，クラスタに残った 2 ノードのうちリモートノード A が脱退ノードの並列プロセス全てを引き継いでリスタートし，リモートノード A だけ並列処理の負荷が増加する．そのため脱退による処理性能の低下が著しい．もしプロセス単位で移譲が可能であれば，リモートノード B の 2 プロセスを，クラスタに残った 2 ノードにそれぞれ 1 プロセスずつ割り当て，並列処理の負荷を分散させ，処理性能の低下を緩和できると考える [2]．しかし負荷分散手法を適用する場合，移行するプロセスの通信相手先がチェックポインティング時とは一部異なる．

チェックポインティング時，各プロセスの通信相手は記録され，リスタート時には記録された情報を元にプロセス間通信の再構築を開始する．各プロセスは自

Implementation of Balancing Method for MPI Parallel Computation Allowing Dynamic Change of the Number of Nodes

†Yuki Sawada, †Kanemitsu Ootsu , †Takeshi Ohkawa , and †Takashi Yokota
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)

プロセスが起動しているノードの IP アドレスと通信用ポート番号を dmtcp_coordinator プロセスにあらかじめ登録し，dmtcp_coordinator プロセス経由で通信相手先プロセスの IP アドレスやポート番号を取得し，接続を行う．図 1 において，リモートノード B が脱退し，子プロセス (子 5，子 6) を残りの 2 ノードにそれぞれ移行する場合を考える．子プロセス (子 5，子 6) はチェックポインティング時には親プロセス (orted2) と通信していたため，dmtcp_coordinator プロセス経由で脱退した通信相手 (orted2) の IP アドレスやポート番号を取得できず，プロセス間通信の再構築に失敗する．そのためリスタート時におけるプロセス間通信の再構築処理を一部変更する必要がある．

4　プロセス間通信の再構築

チェックポインティング時とは通信相手が異なる場合でもプロセス間通信の再構築を実現できるようにする．必要な要件は以下の 2 つである．

- 要件 1. チェックポインティング時には通信していなかったプロセスとの通信を可能にする事
- 要件 2. チェックポインティング時には通信していたプロセスからの接続要求がない場合，脱退したと判断し通信の再構築を行わない

要件 1 によって，負荷分散によって移行した子プロセス (MPI 並列プロセス) は，移行先ノード内の親プロセスとの通信が可能となる．要件 1 を満たすために，dmtcp_coordinator プロセスは IP アドレス・ポート番号取得要求に対する通信相手情報の検索に失敗した場合，要求元ノードの IP アドレスを元に対応する親プロセスの IP アドレス・ポート番号を移行された子プロセスへ伝達するよう実装した．また，チェックポインティング時には通信していなかったプロセスから接続要求を受け取った親プロセスは接続要求を受け付けるように実装した．

要件 2 によって，クラスタに残っているプロセス (orteurn プロセス) と脱退ノード内の親プロセス (orted プロセス) との間の通信エラーを解消できる．要件 2 を満たすために，接続要求の受信に失敗した通信の要求元が，脱退ノードの orted プロセスの場合はエラー処理に移行せず，再構築しない通信として扱うように実装した．これにより，プロセス間通信を再構築でき，MPI アプリケーションのリスタートが可能となった．

5　予備評価

ノード脱退後のリスタートにおいて，負荷分散手法を適用する事によって MPI アプリケーションの計算時間を削減し，処理性能の低下を緩和しているか評価する．3 ノード構成のクラスタにおいて，各ノードで 4 つの MPI プロセスが実行されているものとする．このとき，1 つのノードが脱退した状況を想定する．脱退したノードで実行されていた 4 プロセスを，クラスタに残った 2 ノードに任意のプロセス数割り当て，計算時間を測定する．現在，Android OS 上で DMTCP を動作させる環境を実現できていないため，Linux PC (動作周波数:3.4GHz，コア数:4，メモリ:32GB) クラスタを用いて予備評価を行う．評価には MPI 並列板の N クイーンプログラム (クイーン数:19) を使用する．リスタート時における負荷分散手法適用による効果を明らかにするために，N クイーンプログラムがリスタートしてから解を求めるまでの時間 (計算時間) を比較する．割り当てプロセス数ごとの計算時間を図 2 に示す．

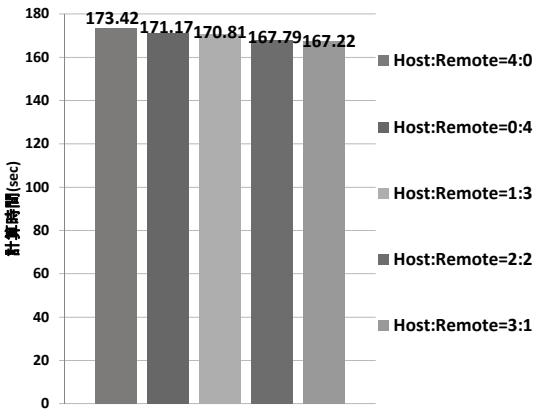

図 2: 割り当てプロセス数ごとの計算時間

図 2 中の凡例は，dmtcp_coordinator プロセスが起動している PC (Host) と，リモート PC (Remote) それぞれに割り当てたプロセス数であり，左 2 本のバーがノード単位によるプロセス割り当て，残りの 3 本が負荷分散手法を適用した場合の計算時間を示している．図 2 より，ノード単位によるプロセス割り当て時の計算時間と比較して，負荷分散手法を適用することで計算時間を削減し，処理性能の低下を緩和することができた．

6　おわりに

本稿では，プロセス単位による並列処理タスクの割り当てを可能とする負荷分散手法を実装し，予備評価を行った．今後の課題はノードの性能を元に割り当てるプロセス数を決定する機構を追加することである．

謝辞

本研究は一部 JSPS 科研費 24500055，15K00068 の助成による．

参考文献

[1] Y.Sawada，Y.Arai，K.Ootsu，T.Yokota，T.Ohkawa “An Android Cluster System Capable of Dynamic Node Reconfiguration”，Proc.7th International Conference on Ubiquitous and Future Networks (ICUFN 2015)，pp.689-694，2015.

[2] 澤田 祐樹，荒井 裕介，大津 金光，大川 猛，横田 隆史 “動的なノード数変更に対応した MPI 並列処理のための負荷分散手法の提案”，情報処理学会 第 77 回全国大会講演論文集，pp.1-113〜1-114，2015.