6J-01

モバイル Android クラスタにおける動的負荷分散のための チェックポインティング処理の実現

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

1 はじめに

近年，Android OS を搭載したモバイル端末 (Android 端末) は，マルチコアプロセッサの搭載などにより大きく性能が向上しており，並列分散処理アプリケーションの開発基盤として注目されている．そのような開発基盤の一つとして，我々は規模を自動的に変化させることができる Android クラスタシステムを開発している [1]．本システムはノード間の通信方式に無線通信を使用し，また MPI アプリケーションを動作対象アプリケーションとしている．しかし，MPI アプリケーションの実行中に一部のノードコンピュータが脱退した場合，アプリケーションは通常は中断することになる．そこで，MPI アプリケーションの実行中にチェックポイントを取得しておき，ノード脱退による処理中断が発生した際にクラスタに残ったノードでチェックポイントから処理を再開する手法を提案する．本稿ではその概要を述べる．

2 Android クラスタシステム

複数のノードを使用してクラスタを構築する場合，ノード間の通信性能がクラスタ全体の処理性能に大きく影響する．本システムでは，最も高速かつ手軽に利用することができる Wi-Fi をノード間の通信手段とする．並列分散処理フレームワークには，Message Passing Interface (MPI) の実装の一つである Open MPI[2] を使用する．さらに，Open MPI，クラスタ上で動作する MPI アプリケーションは，Android Native Development Kit (NDK) を用いてビルドする．一般的な Android アプリケーションは Dalvik VM 仮想マシン上で動作するので実効速度が遅くなるが，NDK により CPU ネイティブのアプリケーションを開発する．本クラスタシステムは通信途絶やノードの移動によるクラスタからの脱退，および新たなノードのクラスタへの参入を自動で検知する機能を備える．参入処理は，クラスタへ参入するノードが，参入リクエストメッセージをブロードキャスト通信により送信する．それを受け取ったクラスタ内ノードは応答を返し，そのノードをクラスタに加える．また，クラスタ内ノードは相互に生存確認・応答のメッセージをやりとりすることで，他のノードが現在クラスタに生存しているか否かの確認をおこなう．

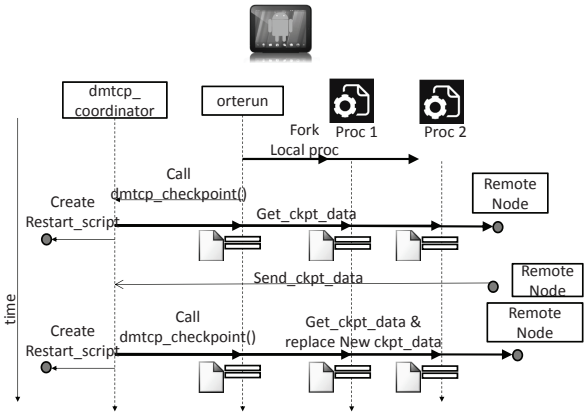

図 1: チェックポイント時の動作

本クラスタシステムにおける必須機能として，MPI アプリケーションの実行中にノードが脱退が発生し，実行中のアプリケーションが中断するような状況においても，チェックポインティングとリスタートにより処理を継続できることが挙げられる．

3 チェックポイント処理

本システムのチェックポインティングソフトウェアには，x86 や ARM 等の様々な命令セットに対応し並列プロセスのチェックポイント処理が可能な点から，Distributed MultiThreaded Checkpointing (DMTCP)[3] を利用する．図 1 にチェックポインティング時のノード内プロセスの動作を示す．図は MPI アプリケーションを起動するノード (ホスト) 上の各プロセスの動作を表している．MPI のデーモンプロセス `orterun` により MPI プロセスが立ち上がりクラスタ内の他のノード (リモート) とともに並列処理が開始されると，`orterun` は一定時間 (現在は 30 秒) 毎に `dmtcp_coordinator` に対して DMTCP のチェックポイント取得処理を依頼する．これにより，その時点でのプロセスイメージがチェックポイントデータとして，各ノードのあらかじめ決められたローカルディレクトリに保存される．チェックポイントデータが保存されたら，リモートノードはホストノードへと自ノードで生成したチェックポイントデータを送信する．

4 リスタート処理

図 2 にリスターティング処理時の各プロセスの動作を示す．リモートノード内で動作しているプロセスとの通

An Implementation of Checkpointing for Dynamic Load Balancing in Mobile Android Cluster System
†Yusuke Arai, ††Yuki Sawada, †Kanemitsu Ootsu, †Takashi Yokota and †Takeshi Ohkawa
Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University (†)
Department of Information Science, Faculty of Engineering, Utsunomiya University (††)

信ができないことを，ホストノード内のプロセスが検知すると，検知したプロセスは orterun に対して，通信途絶が発生した旨のシグナルを送信する．それを受信した orterun は，dmtcp_coordinator に対してリスタートを要求する．それを受け取った dmtcp_coordinator は MPI アプリケーションが動作しているノード上の全 MPI プロセスを kill する．その後，ホストノードはクラスタからのノードの脱退確認をおこなう．脱退がなかった場合はそのまま処理を再開する．

脱退があった場合は，脱退ノードの処理を，クラスタ内の別のリモートノードが引き継ぐようにして処理を再開する別のリモートノードへの脱退ノードの処理の移動は，脱退ノードで保存されたチェックポイントデータをすべてリモートノードへと移動することによりおこなう．脱退ノードを含めた各リモートノードのチェックポイントデータは，チェックポインティング時にホストノードへと送信されている．まず，ホストノードは脱退ノードで保存されたチェックポイントデータすべてを，クラスタ内に生存しているリモートノード 1 台へ送信する．また DMTCP では，リスタート時に各チェックポイントデータに記録されている処理の動作ノードを指定することができる．この機能を利用して，脱退ノードの処理を，ホストノードにより脱退ノードのチェックポイントデータを送信されたリモートノード上で動作するように指定する．このようにすることで，脱退ノードの処理を別のノードに引き継がせることができる．

5 機能検証

実装したチェックポインティング処理，リスターティング処理がそれぞれ図 1，図 2 のシーケンス通りに動作するかの機能検証をおこなった．今回の検証では DMTCP が Android OS 上での動作を実現できていないため，Linux OS (CentOS 6.6) 稼働している PC4 台を使用して検証した．まずチェックポインティング機能について，ホストで orterun により dmtcp_checkpoint() を呼ばれると各ノードでチェックポイントデータが生成される．各リモートノードはチェックポイントデータをホストノードへ送信する．以上の動作がおこなえれば，チェックポインティング処理が正しく機能していると言える．検証の結果，チェックポイントデータを生成できており，ホストノードが全リモートノードのチェックポイントデータを受信できていることが確認できた．

次にリスタート処理について，あるリモートノードとの通信途絶をホストノードが検知すると，全 MPI プロセスが kill される．その後，ノードの生存確認をおこない現在クラスタに生存しているノードを特定する．ノードの脱退があった場合，ホストノードは脱退ノードで保存されたチェックポイントデータすべてを別のリモートノードに送信する．その後，脱退ノードの処理をチェックポイントデータを送信したリモートノードがを継承するようにノード情報を更新し処理を再開する．検証の結果，指定したノード上で脱退したノードのプロセスが動作した状態で処理を再開でき，また正しい実行結果が出力されたことも確認できた．以上の検証から，実装したチェックポイント/リスタート機能が正しく機能していることが確認できた．

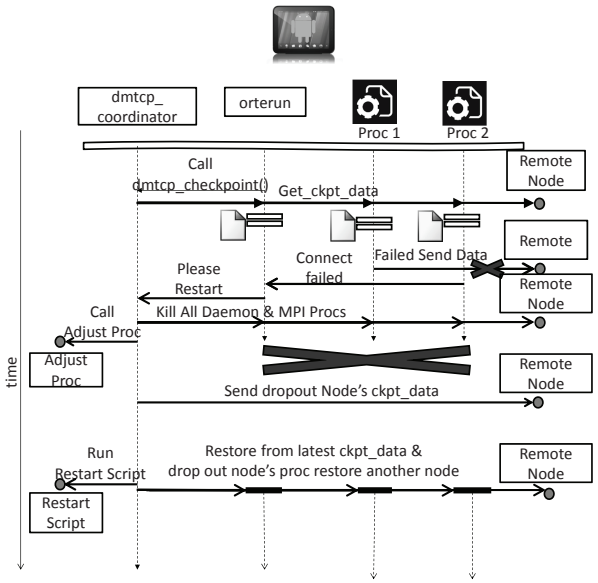

図 2: リスタート時の動作

6 おわりに

本稿では，我々の Android クラスタシステムにおいて，MPI アプリケーション実行中にノードの脱退に対応するために，DMTCP を用いたチェックポイント・リスタート機能を実装した．現状は，特定のノードが脱退ノード 1 台分の処理を引き継ぐため，その 1 台の処理負荷が他の端末よりも大きくなり，結果として全体の処理性能が低下する．そのため，今後の課題として各ノードの負荷を均一にできるような，負荷分散機能を実現する必要がある．

謝辞

本研究は，一部日本学術振興会科学研究費補助金（基盤研究 (C)24500055，同 (C)24500054，同 (C)25330055，若手研究 (B)25730026）の援助による．

参考文献

[1] 荒井 裕介ほか: “端末の動的な参加・脱退を支援する無線接続型 Android クラスタシステムの実装”, 信学技報, Vol.114, No.155, pp.143-148 (CPSY2014-34), 2014.

[2] Edgar Gabriel et al.: “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”, Proceedings, 11th European PVM/MPI Users' Group Meeting, 2004.

[3] J. Ansel et al.: “DMTCP: Transparent Checkpointing for cluster computations and the desktop”, Proceedings of International Parallel and Distributed Processing Symposium, IPDPS 2009, pp.1-12, 2009.