科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　１０　日現在

機関番号：　１２５０１
研究種目：　基盤研究(B)
研究期間：　2011～2013
課題番号：　２３３００００４

研究課題名（和文）分散チェックポインティングを用いたネットワークアプリケーションのモデル検査

研究課題名（英文）Model Checking Network Applications using Distributed Checkpointing

研究代表者
山本　光晴（Yamamoto, Mitsuharu）

千葉大学・理学（系）研究科（研究院）・准教授

研究者番号：００２９１２９５

交付決定額（研究期間全体）：（直接経費）　　14,800,000円、（間接経費）　　4,440,000円

研究成果の概要（和文）：我々の先行研究課題の成果であるネットワークアプリケーションのモデル検査手法をさらに発展させ、分散チェックポインティングを利用することで、より複雑かつ実際的な環境の下で動作するネットワークアプリケーションの検証を可能にした。これは分散チェックポインティングにおける通信の復元機能を実現することによって達成された。また、ネットワークAPIをより実際的なネットワークアプリケーションで用いられているものに対応させ、より広い範囲のアプリケーションの検査を可能にした。

研究成果の概要（英文）：With distributed checkpointing, we enhanced our previous results on model checking network applications so that it can check network applications on more complicated and practical environments. This was accomplished by implementing communication resumption on distributed checkpointing. We also adopted newer network API that is used in practical network applications so as to check wider range of applications.

研究分野：　総合領域

科研費の分科・細目：　情報学・ソフトウエア

キーワード：　モデル検査　チェックポインティング

１．研究開始当初の背景

マルチスレッドプログラムに代表される並行プログラムの作成においては、デッドロックなどのバグを作り込みやすく、さらにスケジューリングの非決定性により、テストなどの手法では潜在的バグが顕在化しにくいという問題がある。この種の問題を解決するための一つの手法として、モデル検査が知られている。
モデル検査とは、システムの状態空間を系統的・網羅的に探索することによって検証を行う手法である。並行プログラムに対するモデル検査においては、スケジューリングが持つ非決定性を網羅的に探索し、どのようなスケジューリングを行ってもシステムが不適切な状態に陥らないことを検証することがその目的となる。
旧来のモデル検査の対象は専用の言語で記述された擬似的なプログラムであったが、近年は実際のプログラムを検査対象とするソフトウェアモデル検査が研究対象となってきている。さらに実環境ソフトウェアモデル検査においては、検査対象のプログラムを、それが実際に使用される状況に近い環境で実行しながら検査を行うことにより、実際に起こりうる不具合を早期に発見することを目的としている。
我々は、平成 20 年度～22 年度の科研費研究課題「仮想計算機によるコミュニケーションバックトラッキングとモデル検査への応用」(以下、先行研究課題とする)において、並行プログラムとして書かれたネットワークアプリケーションのモデル検査に取り組んだ。検証対象のプログラムは Java プログラムのモデル検査器である Java Pathfinder(JPF)の上で動作し、スケジューリングの非決定性が系統的に制御される。一方で、検証対象に対する通信相手となるプログラムは、JPF とは別に通常のプログラムとして動作する。このハイブリッドな構成は、通信相手も含めてモデル検査器の上で動作させる「中央集権化」による手法と比較して、通信相手側の実装言語を問わずに実環境のプログラムを用いることができ、また状態空間の爆発が抑えられるなどの利点がある。
上記先行研究課題で提案・実装した手法では、通信相手側に送受信メッセージの依存関係に関する非決定性が存在する場合にも検査を中止しないようにするために、通信相手の状態を保存・復元するようになっている。当初は研究課題名のとおり、仮想計算機が提供する、システム全体の状態の保存・復元機能を利用して実装を行っていた。しかし、その後の実験・調査により、仮想計算機よりも軽量な、プロセスのチェックポインティング機構を用いても我々が必要とする機能は実現できることが分かり、先行研究課題の期間の途中からはチェックポインティングを用いた実装に軸足を移した。
チェックポインティングとは、実行中のプロセスの状態を、復元可能な形で 2 次記憶装置などに保存することである。我々がチェックポインティングに利用した MTCP は、Linux 上でマルチスレッドプログラムの単一プロセスに対してチェックポインティング機能を提供する実装である。これは、カーネルやチェックポインティング対象のプログラムに一切変更を必要とせず、さらに一般ユーザ権限で導入・実行が可能であるという特徴を持つ。
通信相手の復元には、単なるプロセスの復元だけではなく、検証対象と通信相手との間の通信接続の復元も必要となる。しかし MTCP 自体は通信接続の復元に対応しておらず、また TCP をトランスポート層に用いる場合、プロセス復元時にシーケンス番号の不整合などの問題を生じてしまう。そのため、我々は独自にトランスポート層を実装し、さらにネットワーク関連のライブラリ関数の呼び出しに介入することによって、これらの問題を回避していた。
しかしながら、通信接続の復元は検査対象と通信相手との間に限られており、さらに MTCP が単一プロセスのチェックポインティングに特化されているため、複数の通信相手を同時にチェックポインティングしなければならない場合に対応できていない。また、トランスポート層を独自に実現しているが故の限界として、応用範囲が比較的単純な構成のクライアント・サーバプログラムの検証に限られてしまうという問題がある。例えば、Web クライアントを検証対象とし、Web サーバを通信相手とする場合を考える。このとき、Web サーバがデータベースサーバとも通信を行う構成になっている場合は、現状では両者を同時にチェックポインティングすることができない。また、通信相手のプログラムが一旦子プロセスを作成した後に、子プロセス側で検証対象と通信を行うような場合も、現状では対応できない。

２．研究の目的

本研究課題の目的は、我々の先行研究課題の成果であるネットワークアプリケーションのモデル検査手法をさらに発展させ、分散チェックポインティングを利用することで、より複雑かつ実際的な環境の下で動作するネットワークアプリケーションの検証を可能にすることである。分散チェックポインティングとは、複数ノードで動作しているかもしれない複数のプロセスに対し、相互の通信接続も含めて、保存・復元を行う機構である。例えば、MTCP を利用した分散チェックポインティングの実装として、DMTCP がある。分散チェックポインティングを用いることで、以下のような環境の下で動作するネットワークアプリケーションの検証を可能にすることを目指す。 1) 通信対象が複数プロセ

スからなる環境、2) 通信対象が複数ノードで動作している分散環境、3) 通信対象が子プロセスの生成を伴うような環境。

３．研究の方法

まずは DMTCP を含めた分散チェックポインティング実装の調査から始める。DMTCP では、チェックポインティングの対象となるプロセスを coordinator と呼ばれるプログラムの制御下におくことによって、(場合によっては複数ノードで動作している)複数プロセスの保存・復元を可能にしている。これらのプロセスの間の通信接続も保存・復元される。しかし、DMTCP における通信接続の復元は、coordinator の制御下にあるプロセス同士の間に限られる。一方で、検証対象のプロセスを実行するモデル検査器は、検証の際に探索済みのスケジューリング履歴などの大域的な情報を保持しなければならないため、保存・復元の対象プロセスではなく、よって coordinator の制御下にはない。従って、DMTCP をネットワークアプリケーションのモデル検査に応用するためには、coordinator の制御下にあるプロセスとないプロセスとの間の通信接続の復元にも対応させなければならない(図 1: 復元されない通信接続)。

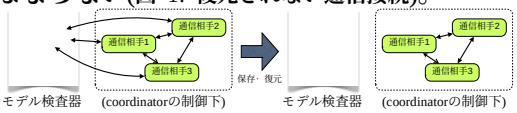

図 1: 復元されない通信接続

最初に予備実験として、本番の実装において想定される問題点の洗い出しとパフォーマンス測定を目的に、coordinator の制御下にあるプロセスとないプロセスとの間の通信接続の復元を実現するプロトタイプの作成を行う。さらにプロトタイプ作成で得た知見を元にして、我々の先行研究課題「仮想計算機によるコミュニケーションバックトラッキングとモデル検査への応用」で提案したネットワークアプリケーションのモデル検査手法に、分散チェックポインティングを組み込む。具体的には、通信接続の復元部分を、独自のトランスポート層実装とネットワーク関連のライブラリ関数呼び出しに対する介入によるものから、分散チェックポインティングによるものに置き換える。これは Java Pathfinder の拡張として我々が先行研究課題で設計・実装した、I/O キャッシュ層を変更することによって行う。

次に完成したモデル検査器を用いて、ネットワークアプリケーションを応用例として用いた評価を行う。単純な構成のクライアント・サーバプログラムの検証であれば、先行研究課題での独自のトランスポート層実装による通信接続の復元と、本研究課題での分散チェックポインティングによるものと間でパフォーマンス比較が可能なはずである。また、本研究課題の目的は分散チェックポインティングを用いることにより、モデル検査が適用可能なネットワークアプリケーションの範囲を拡大することであったので、単純な構成のクライアント・サーバプログラムを超えた環境で動作するアプリケーションに対して、本研究課題での手法が実際に適用可能かどうか実験によって評価する。

４．研究成果

(1) 分散チェックポインティングにおける通信接続復元の実現とモデル検査への応用

分散チェックポインティング実装としては当初の予定通り、DMTCP を使用することにした。DMTCP では、チェックポインティングの対象となるプロセスを coordinator と呼ばれるプログラムの制御下におくことによって、(場合によっては複数ノードで動作している)複数プロセスの保存・復元を可能にしている。これらのプロセスの間の通信接続も保存・復元される。 しかし、DMTCP における通信接続の復元は、coordinator の制御下にあるプロセス同士の間に限られる。一方で、検証対象のプロセスを実行するモデル検査器は、検証の際に探索済みのスケジューリング履歴などの大域的な情報を保持しなければならないため、保存・復元の対象プロセスではなく、よって coordinator の制御下にはない。従って、DMTCP をネットワークアプリケーションのモデル検査に応用するためには、coordinator の制御下にあるプロセスとないプロセスとの間の通信接続の復元にも対応させなければならない。この問題を解決するため、我々は coordinator の制御下に新たにプロキシプロセスを導入した。モデル検査器のかわりにプロキシプロセスが通信相手との間の通信接続を確立し、UNIX ドメインソケットを通じてソケットディスクリプタをモデル検査器に渡す。保存時に一時的にモデル検査器側のソケットは切断されるが、プロキシプロセスが同じソケットを保持しているため、復元時には通信接続も復元される、これを再度 UNIX ドメインソケット経由でモデル検査器に渡す。

実際にこのモデル検査器を用いて、Java による Secure Shell 実装である JSch に含まれるファイル転送プログラム ScpTo を検証し、スレッドが適切に同期されていないことによる競合条件を発見することに成功した。また、バグを修正した ScpTo で検証が成功することも確認した。

本成果をまとめた論文はソフトウェア工学分野におけるトップカンファレンスの一つである ASE(Automated Software Engineering)に採択された。また、我々の先行研究課題におけるチェックポインティングを利用したモデル検査器の成果と、本課題における分散チェックポインティングに関する部分とをまとめたものを雑誌論文として投稿し、その論文は本課題研究期間中に TSE(IEEE Transactions on Software

Engineering)に採録が決定した。その後、本課題研究期間終了直後の 2014 年 5 月に出版されている。

(2) Java PathFinder モデルクラスとしての Java.nio クラスの実装

我々の先行研究課題「仮想計算機によるコミュニケーションバックトラッキングとモデル検査への応用」では、我々がモデル検査器の基盤として用いている Java PathFinder がネットワーク API として使用するモデルクラスとして Java.io クラスの実装を与えていたが、より多くのネットワークアプリケーションに対する検証を可能とするためには、ノンブロッキング I/O への対応が不可欠となる。

本研究課題では、モデルクラスとしてノンブロッキング I/O を含む Java.nio クラスの実装を新たに与えた。これは本課題とメンバーが重なっている別課題「クラウドコンピューティングミドルウェアのソフトウェアモデル検査手法」の基盤ともなる。モデルクラスがノンブロッキング処理や例外処理を正しく処理できているか確認するために、モデルベースのテストツール Modbat を開発してそれを用いたこと、検証の実例として、Java で実装されている HTTP サーバ Rupy の実際のバグを発見したことが特徴となっている。この成果をまとめた論文も、年度は異なるが前述のトップカンファレンス ASE に採択された。

(3) その他

分散チェックポインティングおよびモデル検査環境に関わるソフトウェアとして、以下の 2 つを開発した。1 つ目は Java PathFinder の環境をまるごと仮想環境として提供するもので、利用者が複数のソフトウェアをインストールする手間を省くのみならず、利用者側の環境の差異によらず、統一的・標準的な検証環境を提供するのに役立つ。2 つ目は仮想環境を利用した分散チェックポインティングをモデル検査以外に応用したものであり、ネットワークアプリケーションに対してネットワーク切断などの稀なイベントを意図的に発生させて、正しくそれらに対処できているかを確かめるためのツールである。

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 4 件）

① W. Leungwattanakit, C. Artho, M. Hagiya, Y. Tanabe, M. Yamamoto, and K. Takahashi,
Modular Software Model Checking for Distributed Systems,
IEEE Transactions on Software Engineering,
査読有, Vol.40, No. 5, 2014, pp. 483 - 501,
DOI: 10.1109/TSE.2013.49

② C. Artho, A. Biere, M. Hagiya, E. Platon, M. Seidl, Y. Tanabe, and M. Yamamoto,
Modbat: A Model-based API Tester for Event-driven Systems,
Hardware and Software: Verification and Testing, 9th International Haifa Verification Conference, HVC 2013, Haifa, Israel, November 5-7, 2013, Proceedings,
査読有, LNCS 8244, 2013, pp. 112 - 128,
DOI: 10.1007/978-3-319-03077-7_8

③ C. Artho, M. Hagiya, R. Potter, Y. Tanabe, F. Weitl, and M. Yamamoto,
Software Model Checking for Distributed Systems with Selector-Based, Non-blocking Communication,
28th IEEE/ACM International Conference on Automated Software Engineering (ASE),
査読有, 2013, pp. 169 – 179,
DOI: 10.1109/ASE.2013.6693077

④ W. Leungwattanakit, C. Artho, M. Hagiya, Y. Tanabe, and M. Yamamoto,
Model Checking Distributed Systems by Combining Caching and Process Checkpointing,
26th IEEE/ACM International Conference on Automated Software Engineering (ASE),
査読有, 2011, pp. 103 - 112,
DOI: 10.1109/ASE.2011.6100043

〔学会発表〕（計 1 件）

① Cyrille Artho,
Modbat: A model-based API tester for event-driven systems,
第 10 回 ディペンダブルシステムワークショップ (DSW 2012),
2012 年 12 月 11 日～2012 年 12 月 12 日,
独立行政法人理化学研究所 計算科学研究機構

〔その他〕

ホームページ等

http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/net-iocache

６．研究組織

(1)研究代表者

山本 光晴（YAMAMOTO, Mitsuharu）
千葉大学・理学研究科・准教授

研究者番号：00291295

(2)研究分担者
萩谷 昌己（HAGIYA, Masami）
東京大学・情報理工学系研究科・教授
研究者番号：30156252

アルト シリル（ARTHO, Cyrille）
独立行政法人産業技術総合研究所・セキュアシステム研究部門・主任研究員
研究者番号：30462831

田辺 良則（TANABE, Yoshinori）
国立情報学研究所・アーキテクチャ科学系・特任教授
研究者番号：60443199

(3)連携研究者

なし

(4)研究協力者
レオンワタナキット ワチャリン（LEUNGWATTANAKIT, Watcharin）
東京大学→千葉大学
(平成 23 年度)

ヴァイテル フランツ（WEITL, Franz）
千葉大学
(平成 24 年度～平成 25 年度)

ポッター リチャード（POTTER, Richard）
東京大学
(平成 24 年度)

セビ ナジム（SEBIH, Nazim）
東京大学
(平成 25 年度)