IPA

2010 年度 未踏 IT 人材発掘・育成事業 採択案件評価書

1. 担当 PM

藤井 彰人 PM
（グーグル株式会社 エンタープライズ プロダクト マーケティング マネージャー）

2. 採択者氏名

チーフクリエータ：玉井 森彦（奈良先端科学技術大学院大学 助教）
コクリエータ：酒井 憲吾（株式会社構造計画研究所）

3. 委託金支払額

2,880,000 円

4. テーマ名

プロセスの仮想化による分散システム開発支援ソフトウェア

5. 関連 Web サイト

http://takt-dev.net/

6. テーマ概要

Gmail, Mixi, Twitter など、近年の有用なアプリケーションの多くは、クラウドコンピューティングの技術を利用して実現されており、人と人とをつなぐコミュニケーションツールとして、我々の社会活動になくてはならない基盤サービスとなっている。今後さらに進むクラウド上でのサービス展開と、それを利用する膨大なユーザ数を支え、社会インフラとしての安定的なサービスの提供を行うためには、クラウドの信頼性の向上は必要不可欠な社会的課題である。

クラウドの実体は、データセンタ内に設置された多数のコンピュータ上で動作する分散システムであり、代表的なソフトウェアとして、Hadoop, memcached,

kumofs などが存在する。これら分散システムは、多数のプロセス、またはスレッドといった並列実行される処理の流れと、それらの間の通信に基づく協調動作により実現されている。このような複雑に絡み合う並列性に起因するバグは、その再現が非常に困難であり、デバッグの際には、バグが再現するまでシステムを何度も再実行する、というような非生産的な作業に囚われる。また、バグを未然に防ぎ、システムの信頼性をあらかじめ向上させるためには、十分なテストを行う必要があるが、統合テストのためのテストケースの記述は難しく、分散システムを十分に長い間（例えば一晩中）動作させておき、その間不具合が生じなかったらテストをパスしたとみなす、というような曖昧な基準に基づくテストが行われがちである。

本プロジェクトでは、分散システムの開発におけるデバッグ、テストの工程を容易化することを目的に、

(1) 分散システムを確定的に実行する機能
(2) 分散システムを様々なタイミングで網羅的にテストする機能
(3) 分散システムの実行、およびテストの様子を可視化する機能
(4) 分散システムの信頼性を表す指標を提示する機能

を備えた分散システム開発支援ソフトウェア「takt」を開発した。

7. 採択理由

昨今、仮想化や分散システムの構築が広がり、このような環境における障害検出、解析作業は今後その需要を増していくことが想定される。本プロジェクトは、まさにこれらを支援するためのツールであり、仮想時間上で実行されるシミュレータの実装というアイデアがユニークである。開発の根幹にかかわる技術であり、未踏として採択すべきと考えた。プロジェクト期間中に広く使われている分散ソフトウェアの障害検出なども視野に入れた。

8. 開発目標

以下の機能を持つ分散システム開発支援ソフトウェアを開発することを目標とした。

- バグの再現性を確保するため、プロセス（スレッド）の走行タイミングを集中管理し、バグ発生時と同一タイミングでシステム全体を確定的に実行する機能
- 統合テストを容易化するため、分散システムを様々なタイミングやフォールト（システムコールのエラーなど）のもとで網羅的にテストする機能

- 開発者が直感的にデバッグ、テストを行えるようにするため、分散システム全体の実行の様子、および、網羅テストの様子を可視化して表示する機能
- 網羅テストをどの程度行えば、システムの信頼性が確保できたといえるか、その判断の指標を与えるため、テスト系列に対する分散システムの振る舞いのパターンを記録し、そのバリエーションが十分に出尽くしたかどうかをもとに、システムの信頼性を定量化して提示する機能

9. 進捗概要

分散システム開発支援ソフトウェア「takt」を開発した。takt は以下の要素から成る。

- takt Core
  分散システムの確定的実行機能を提供する。この機能は、
  (1) 分散システムを構成する各プロセス（スレッド）が呼び出すシステムコールを全て捕捉し、それらを OS 内の実システムコールとは無関係に takt Core 内部で仮想的に実行する機能
  (2) 各プロセス（スレッド）を takt Core 内部で管理する仮想時間上で、並列性を排除しつつスケジューリングする機能
  により実現される。
- テストシナリオライブラリ
  takt Core 上でのテストシナリオの記述を容易化するための API を提供する。
- ビジュアライザ
  以下の機能を提供する。
  (1) 分散システム全体の実行の様子を可視化する機能
  (2) 網羅テストの実行に伴うシステムの信頼性の向上の様子を可視化、定量化して表示する機能

テストシナリオライブラリを用いて記述された網羅テストの例を図 1 に示す。この例では、memcached（http://memcached.org/）に対し、システムコール accept を 1/100 の確率で失敗させ、かつ、スレッドのスケジューリングパターンを変えながら網羅テストを実行している。網羅テストの過程で検出されたバグは、takt Core の確定的実行機能により、何度でも再現させることが可能である。

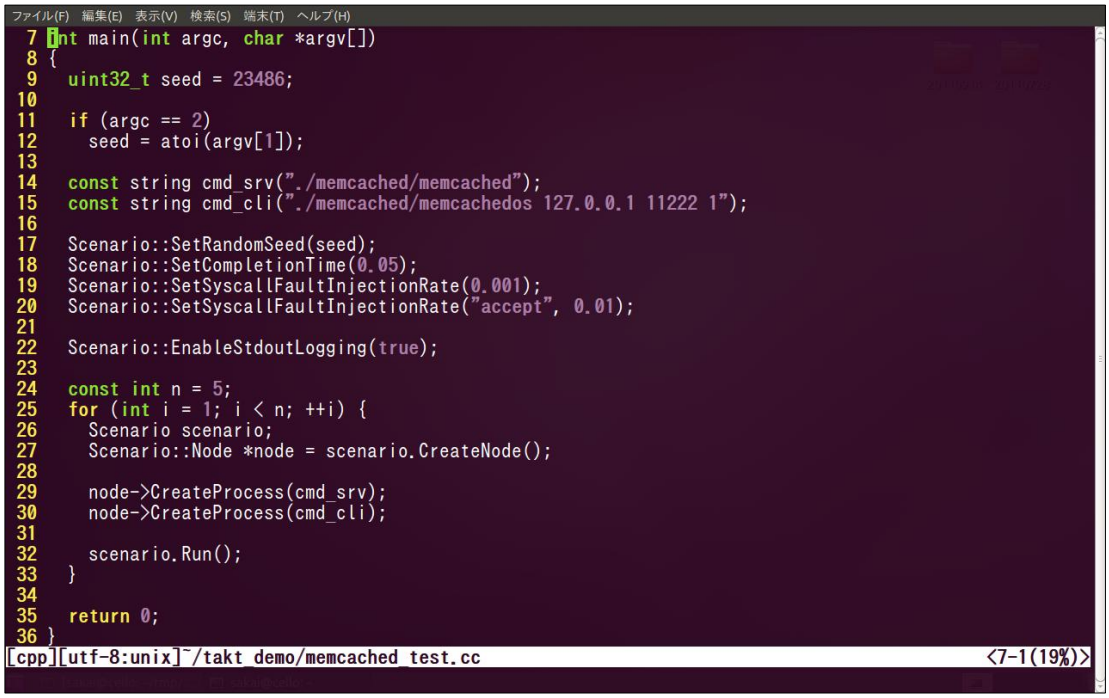

図 1. テストシナリオライブラリを用いて記述された網羅テストの例

ビジュアライザを用い、分散システムを構成するプロセス（スレッド）の時間軸上での振る舞い（呼ばれたシステムコールや、発生したエラー）を可視化した様子を図 2 に示す。画面奥から手前に向かう各線が、各々スレッドを表しており、各線上で、対応するスレッドが呼び出したシステムコールの種類や検出されたエラーを円で表示している。本機能を用いることで、あるスレッドで発生したエラーが、時間軸上で、他のスレッドとどのような処理の経過を経て発生したものか、その全体像を把握することが可能となる。

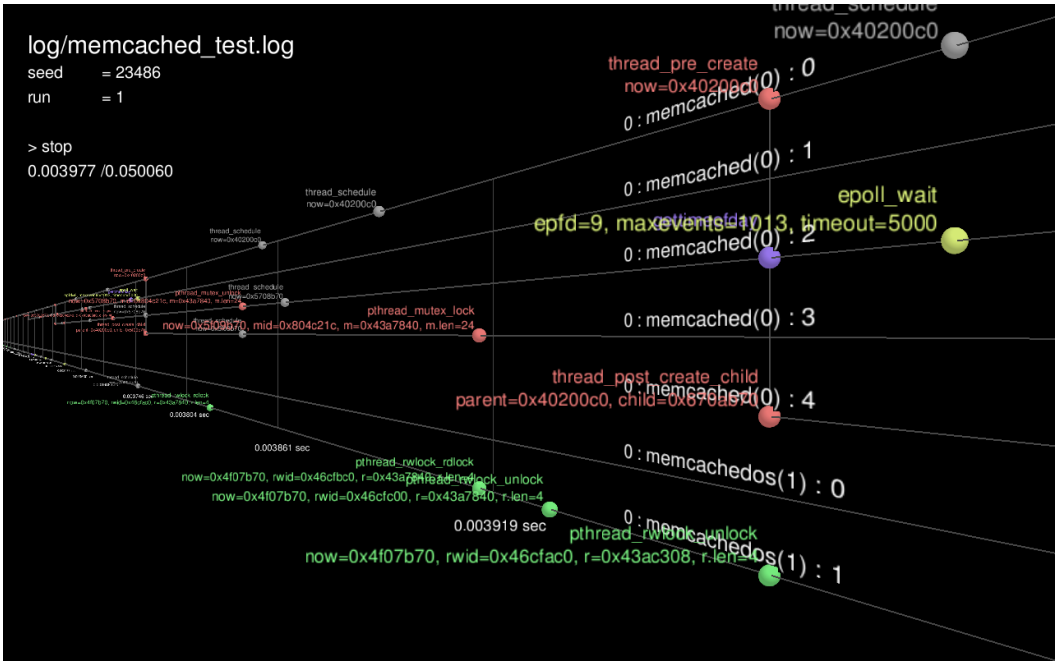

図 2. 分散システムを構成するプロセス（スレッド）の時間軸上での振る舞いの可視化

ビジュアライザを用い、網羅テストの経過に伴うシステムの信頼性の向上度合を可視化した様子を図 3 に示す。二次元平面上に配置された各球は、各テストの実行結果におけるシステムの振る舞いを意味しており、似通った振る舞いのものは近くに、異なった振る舞いのものは遠くに配置される。新たな球が、その周りに他の球が全く存在しない位置に出現した場合、過去のテストでは発見されなかった、新たなシステムの振る舞いが見つかったことを意味する。このような球の発生確率は、網羅テストを経るにつれ、だんだん減少するため、システムを様々なスケジューリングのパターンでテストし尽したことが直感的に把握可能となっている。また、画面左上の「coverage」の項目には、この性質を 0～100 の数値で定量的に表示しており、この値が大きくなるにつれ、システムの信頼性が向上したこと、すなわち、網羅テストの「やめ時」が判断可能となっている。

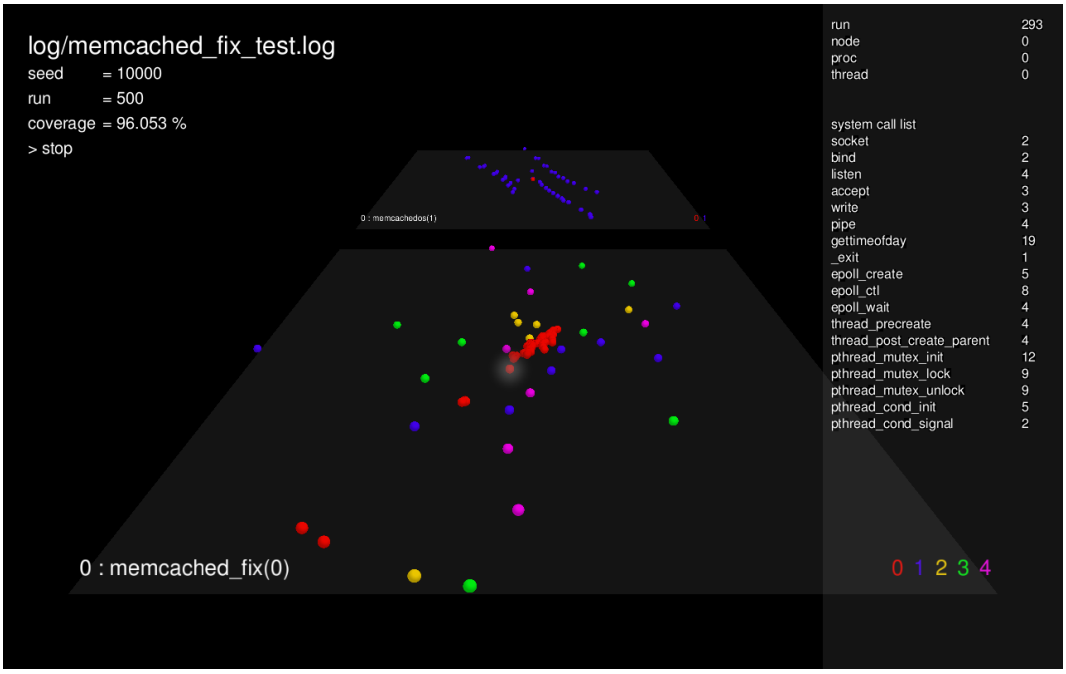

図 3. 網羅テストの経過に伴うシステムの信頼性向上度合の可視化と定量化

10. プロジェクト評価

分散システムにおけるデバッグ作業は、通常困難を極める。これは現在の分散システムがネットワークを中心に離散的に発生するイベントを取り扱う事が多いためで、タイミングの散らばりや重なりが、解析困難な問題を引き起こすためである。本プロジェクトは、これらの問題を解決する手法を提供している。

玉井、酒井両名の実経験にもとづく課題設定であったため、プロジェクトスタート当初より、しっかりとした目的意識をもって開発を進めることができ、また途中でもその目標を見失うことなく開発を粛々と進めたことを高く評価したい。その結果、基盤技術をきちんと実装しつつも、進捗状況が一目で分かるビジュアライザや、ホワイトボックステストが不可能な事象にたいして、新しい品質管理基準となり得るテストカバレッジの数字も提示するという、当初目標を上回る成果を実現した。

クリエータとコクリエータの明確な役割分担と、効率的な連携作業についても本プロジェクトの良い結果に繋がっている。彼らのチームワークも高く評価したい。

開発ツールは、一般ユーザ向けサービスやソフトウェアとは異なり、まさに IT 技術者の開発基盤を支えるテクノロジーである。未踏からこのようなソフトウェアを成果として世界に提示出来る事は非常に重要であると考える。

11. 今後の課題

今後の課題としては、本プロジェクト期間内では未実装だった、いくつかの疑似システムコールの実装を進めることがあげられる。本プロジェクト期間内では、多くのネットワークプログラミングやシステムプログラミングの教科書で解説されているような、代表的なシステムコールに関して優先的に実装を行ったため、一般的にはあまり利用されない、マイナーなシステムコールに関しては実装を見送っていた。これらマイナーなシステムコールを利用して実装されている分散システムをサポートするためには、不足している疑似システムコールの実装を進める必要がある。

また、多くのユーザによって利用されている分散システム（例えば、flare、kumofs、MongoDBなど）に対し、taktを用いて網羅的テストによるバグの検出を行うとともに、検出されたバグに対し、確定的実行機能によるバグの原因の特定を行い、そのシステムの開発コミュニティに対しバグ報告を行うことが挙げられる。