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
—Toward More Robust Cloud Applications—

1. 背景

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．

2. 目的

上記の問題を改善するため，以下の機能を持つ開発支援ソフトウェアを開発する．

- 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

3. 開発の内容

上記の目的を実現するソフトウェア「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，および，(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**：(1) 分散システム全体の実行の様子を可視化し，また，(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を提供する．

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．

```
ファイル(F)  編集(E)  表示(V)  検索(S)  端末(T)  ヘルプ(H)
 7 int main(int argc, char *argv[])
 8 {
 9   uint32_t seed = 23486;
10
11   if (argc == 2)
12     seed = atoi(argv[1]);
13
14   const string cmd_srv("./memcached/memcached");
15   const string cmd_cli("./memcached/memcachedos 127.0.0.1 11222 1");
16
17   Scenario::SetRandomSeed(seed);
18   Scenario::SetCompletionTime(0.05);
19   Scenario::SetSyscallFaultInjectionRate(0.001);
20   Scenario::SetSyscallFaultInjectionRate("accept", 0.01);
21
22   Scenario::EnableStdoutLogging(true);
23
24   const int n = 5;
25   for (int i = 1; i < n; ++i) {
26     Scenario scenario;
27     Scenario::Node *node = scenario.CreateNode();
28
29     node->CreateProcess(cmd_srv);
30     node->CreateProcess(cmd_cli);
31
32     scenario.Run();
33   }
34
35   return 0;
36 }
[cpp][utf-8:unix]~/takt_demo/memcached_test.cc                    <7-1(19%)>
```

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．

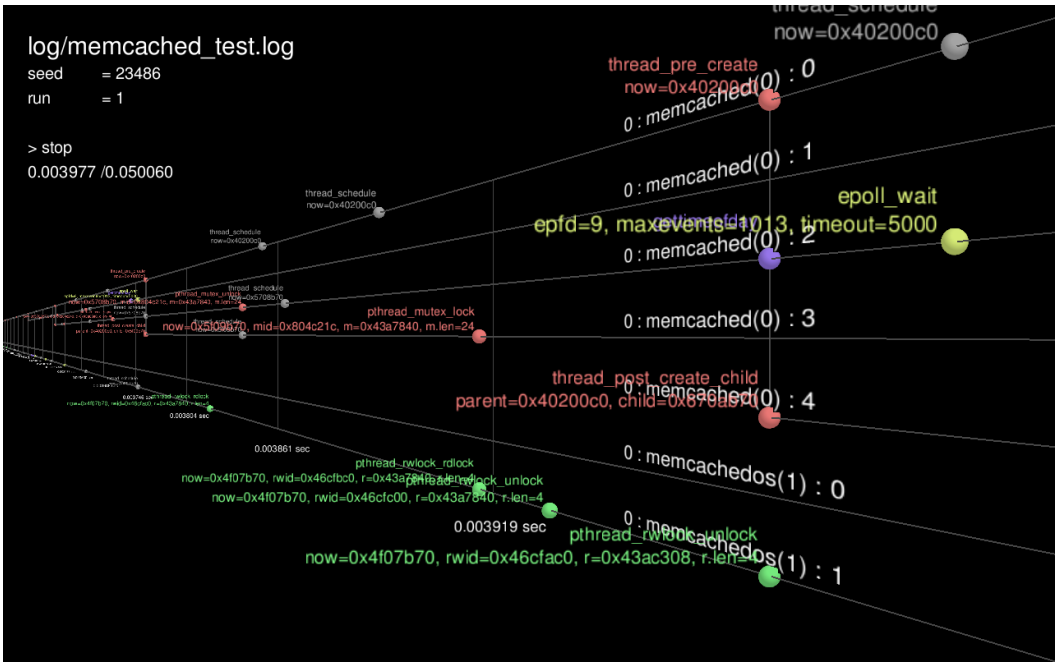

図 2 分散システムを構成するプロセス（スレッド）の時間軸上での振る舞いの可視化

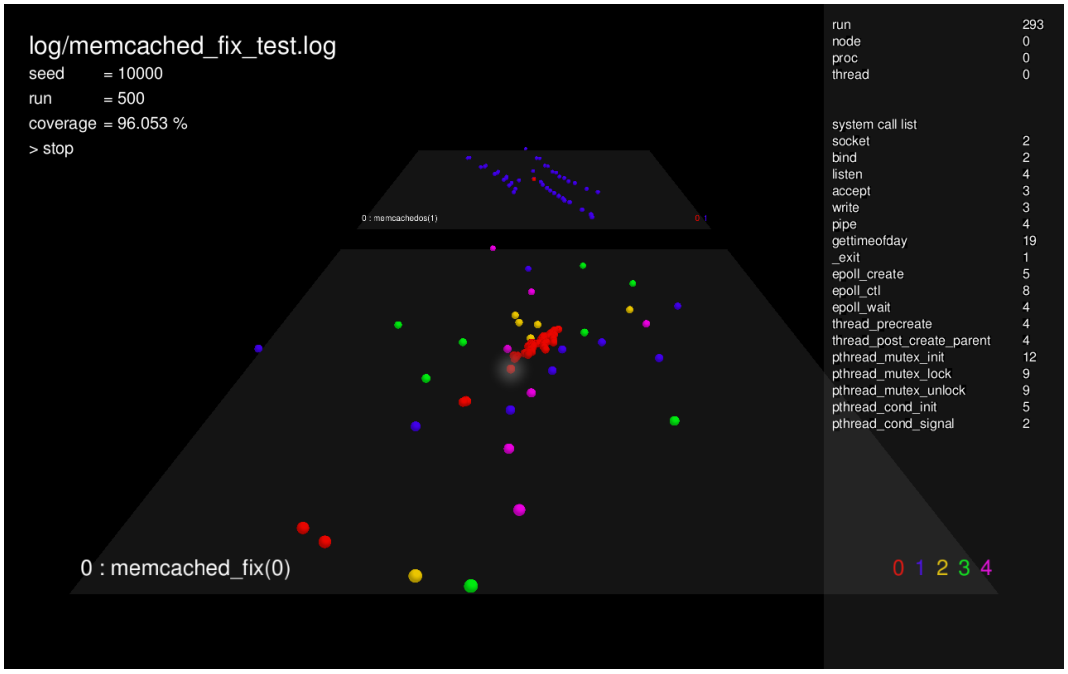

図 3 網羅テストの経過に伴うシステムの信頼性向上度合の可視化と定量化

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.

4. 従来の技術(または機能)との相違

gdb, Valgrind, SystemTap に代表される既存の開発支援ツールでは, 分散システムの実行は OS のスケジューリングや実ハードウェアの関与のもとに行われるため, 並列性に起因するバグの再現性が不確実な状況下でのデバッグを強いられ, 作業効率を著しく低下させる. また, xUnit のような単体テストのためのフレームワークは, 統合テスト(システムレベルのテスト)を支援するものではなく, また, システムの信頼性に対し, 定量的な評価指標を与えるような仕組みも存在しない.

5. 期待される効果

本プロジェクトで開発したソフトウェアを用いることで, 分散システムのデバッグ, およびテストの工程において, 特に並列性に起因するバグの原因の早期特定, さらには, 網羅テストによるバグの洗い出しによるシステムの信頼性の大幅な向上が期待される.

6. 普及(または活用)の見通し

多くのユーザに利用されている既存の分散システム(memcached, flare, kumofs など)に対し, 開発したソフトウェアを用いてバグの発見を行い, 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. 短期的, 中期的には, このような活動を地道に行い, 本プロジェクトの成果を徐々に広めていく. また, 長期的には, ネットワーク上のサービス, もしくはオープンソースソフトウェアとして一般に利用可能なかたちでソフトウェアを公開し, 分散システム, ひいてはクラウドの信頼性向上に寄与することで, 本プロジェクトの成果を社会還元しようと考えている.

7. クリエータ名(所属)

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)
コクリエータ: 酒井 憲吾 (株式会社構造計画研究所)

(参考)関連 URL

http://takt-dev.net/