B-017

分散システム開発でのディペンダブルなプロトコルの SPIN による検証

Verification of a Dependable Protocol Using SPIN in Distributed System Development

及川 雄也† 小林 洋†
Yuuya Oikawa Hiromi Kobayashi

1. はじめに

最近，Web アプリケーションサービスの普及と共に，クライアントからサーバにデータの問い合わせを行う際のサーバの故障の対処方法として，複数個のデータベースのレプリカ(replica)を用意し，投機的(speculative)ビザンチンアルゴリズムと呼ばれるディペンダブルなプロトコルを用いる方法が注目されている．これは，クラウドコンピューティングにおけるサービス側のデータ管理方法として有望な方法である．我々は，先の研究でこの中の一つである Zyzzyva と名付けられたプロトコルを基に P2P 向けのプロトコルを開発した．この研究では，この改良プロトコルを，モデル検査ツール SPIN を用いて検証を行うことを最終目的にしているが，その前に，オリジナルの Zyzzyva について SPIN により検証を行ったので，その結果やそれによって得られた知見について報告する．

2. Zyzzyva

ビザンチン故障[1-2]は，分散システムにおいて異なるノードに対して異なる出力応答をするような故障のことで，分散システムでの故障クラスの中では最も厳しい状況における故障で，これに対処できれば他のクラスの故障にも対処できることになる．投機的ビザンチンアルゴリズム[3-5]は，複数個のデータベースの複製(replica)を用意し，データベースサーバのビザンチン故障に対処するプロトコルである．但し，有名なビザンチン将軍問題[1-2]とは，問題が異なるので注意を要する．このプロトコルは楽観的(optimistic)アルゴリズムとも呼ばれ，現在のコンピュータシステムにおいて故障はごく稀にしか起きないことから，通常は故障がないものとし実行を行ない，故障が検出された場合にのみその対処を行う方法である．その中で処理効率が良いと言われているものに Zyzzyva[3-5]と呼ばれるプロトコルがある．Zyzzyva では，f 個の故障ノード（サーバ）に対処するためには 3f+1 個のノードが必要となる．システムが非同期の場合には，このような故障に対しては理論上は対処出来ないことが知られているが，Zyzzyva ではタイムアウトの設定により実用上支障がない方法で対処している．

Zyzzyva では，図 1 に示すように，クライアント C がレプリカを含むサーバ群の中のプライマリ P に要求を出すことから開始される．クライアント C からの要求を受けたプライマリ P は，レプリカ R1~R3 に対して要求命令(ORDER-REQUEST)を送る．すると，プライマリ P を含めた全レプリカがクライ アント C に対し返答(SPEC-RESPONCE)を返す．この際，

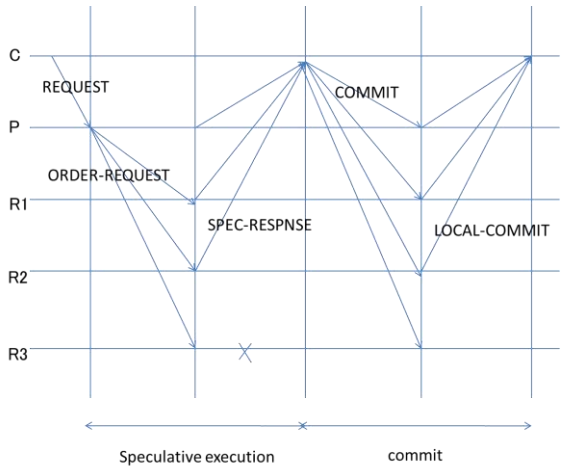

図 1 Zyzzyva での動作

（1）クライアント C のところでプライマリ P も含めて 3f+1 個のレプリカからの結果が一致していれば合意が取れたと見なし，全てのレプリカが正常にメッセージの送受信を行なっていたと判定する．この場合は 1 フェーズで処理は終了となる．故障は稀にしか起きないので，通常はこの第 1 フェーズで終了する．

（2）クライアント C に SPEC-RESPONSE が 2f+1 個しか戻って来なかった場合には，コミット認証フェーズと呼ばれる第 2 フェーズへ進む．この場合，引き続きクライント C は全てのレプリカに対して，COMMIT を送る．これに対して全てのレプリカは LOCAL-COMMIT を返し，クライアント C がその戻り値を確認する．これによりフェーズ 1 で SPEC-RESPONSE を返して来なかったレプリカが故障しているのか，又はプライマリ P 自身が故障しているのかを判別することが出来る．プライマリ以外のレプリカ Ri が故障していた場合には処理は行なわない．なお，図 1 は R3 から SPEC-RESPONCE が返って来なかった場合における Zyzzyva での処理を示している．

（3）クライアント C において，プライマリがビザンチン故障を起こしあるノードに対してだけ異なるデータを送っていると判別された場合には，ビュー変更(view change)と呼ばれるプライマリの変更処理が行なわれる．なお，クライアント C から送られて来た COMMIT によってレプリカ Ri 自身でプライマリ P から正しいデータが送られて来ていないと判断できることがある．この場合には，レプリカ Ri はクライアント C を含む全レプリカに対して告発(accusation)と呼ばれるメッセージを送り，ビュー変更を促す．ビュー変更が決定したら，レプリカ Ri の中から新しいプライマリが選択され，故障と判断された前プライマリ P はシステムから除外されることになる．

†東海大学大学院工学研究科 Tokai University

活性:クライアント Cが正常ならばリクエストは
いつか完了する.
(1)LTL での記述
(クライアント C=正常)→F(クライアント C=完了)

(2)SPIN での記述
p → <>q
#define p Client == Normal
#define q Client == End

図２　性質の LTL での記述と SPIN での実装

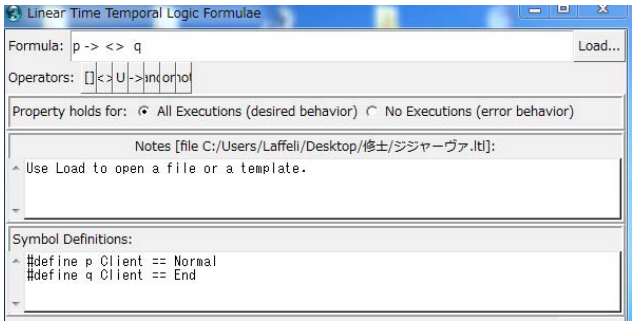

図３　SPIN の LTL 式入力フォーム

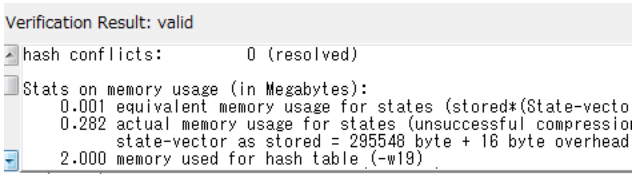

図４　実行結果の一部

3. SPIN による検証

SPIN[6-7]は，ソフトウェアのモデル検査のためのツールである． このツールでは，検査対象のシステムを抽象化し専用の言語 Promela（Process Meta Language）で記述し，検証したい性質を線形時相論理（Linear Temporal Logic :LTL）で記述することにより，制約違反の検証が可能となる．この検証では，システムをどのように抽象化するかということと，どのような性質を検証するかということがポイントとなり，その点が研究対象にもなっている．Promela は C 言語に似た文法を持つ言語であるが，C 言語や他の多くの一般的な言語と異なり，非決定性を持つ処理が実行可能である． 本研究では，Zyzzyva プロトコルの検証する性質として，活性（Livness）と安全性（Safety）を扱うことにした．この場合の活性とは，「クライアント C が正常であるならば，リクエストはいつか完了する」ということであり，安全性とは，「リクエストの履歴（history）の順序が正しい」ということとした[4-5]． この２つの性質を LTL で表した後，SPIN に実装すると，図 2 のようになる．LTL では命題論理の結合子，否定¬，論理和∨，論理積∧，含意→に加え「ずっと成り立つ」を表す G や「いずれ成り立つ」を表す F を使って性質を表現する． SPIN で記述する場合，G や f は[]，<>で表す．

検証したい性質の式を入力し，検証を開始すると SPIN は valid か not valid の値を返す． not valid の場合，SPIN は記述した論理式が成り立たない場合(反例)を一つ示す．SPIN の LTL 式入力フォームを図３に示す．

4. 実行結果

SPIN による実行結果について，以下に記す．

今回の研究で用いたマシンは以下のとおりである．

マシン名 Gatewsy EC5800-45S
CPU　　intel Centrino2
OS　　Windows7

性質検証の実行時間は，両方とも約 9 秒であり，どちらも valid という結果が返って来た． 実行結果の一部を図 4 に示す．この結果，Zyzzyva では活性と安全性の両方の性質を満たしており，クライアントが正常であれば，故障が発生してもいつか必ずプロトコルを完了することができ，また各ノードの行動履歴が途切れることがないことを検証することができた．

5. 終わりに

Zyzzyva をモデル化し，SPIN により活性と安全性について検証を行った．今回と同様の検証は，文献では見い出せなかったが，既に行われているとは考えられる．また，SPIN による検証については，ただツールを動かしただけではないかという批判が時々聞かれる．しかしながら，SPIN を用いるためには，まず，システムを抽象化して Promela により記述する必要があり，次に，検証したい性質を見出し LTL で記述する必要があり，この抽象化と検証性質を見出し記述することが研究対象となっている．

本研究の最終目的は，Zyzzyva を基に P2P 向けに我々が以前開発したプロトコル[8-9]の検証であり，現在これを行っている．

参考文献

[1] L. Lamport, R. Shostak and M. Pease: The Byzantine generals problem, ACM Trans. Program. Lang. Syst., 4, 3, pp.382-401, 1982.
[2] 米田友洋，梶原誠司，土屋達弘: ディペンダブルシステム，共立出版，2005.
[3] A. Wright: Contemporary Approaches to Fault Tolerance: CACM, 52, 7, pp.13-15,2009.
[4] R. Kotla, A. Clement, E. Wong, L. Alvisi, and M. Dahlin: Zyzzyva: Speculative Byzantine Fault Tolerance, CACM, 51,11, pp.86-95, 2008.
[5] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong: Zyzzyva: Speculative Byzantine Fault Tolerance, ACM Trans. Computer System, 27, 4, pp.1-39, 2009.
[6] G.J. Holtzmann: The Spin Model Checker — Primer and Reference Manual, Addison-Wesley, 2003.
[7]SPIN ACM Link: http://spinroot.com/spin/whatispin.html
[8]松本祐亮，小林洋: 投機的ビザンチンアルゴリズムの P2P システムへの 適用，第９回情報科学技術フォーラム (FIT2010)，第１分冊，PP.441-444 (2010.9).
[9] Y.Matsumoto, and H. Kobayashi: A Speculative Byzantine Algorithm for P2P System, Proc. IEEE 2010PRDC, pp.231-232, (2010.12).