C-023

投機的ビザンチンアルゴリズムの P2P システムへの適用

A Speculative Byzantine Algorithm for P2P System

松本 祐亮† 小林 洋†
Yusuke Matsumoto 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，同一データベースの複製(replica)を準備し，その間で投機的(speculative)ビザンチンアルゴリズムを用いるという方法が注目されるようになって来ている．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も有望な方法である．本研究では，そのアルゴリズムの中で処理効率が良いと言われている Zyzzyva と呼ばれるアルゴリズムを P2P システムへ適用するための改良を行った．P2P システムへの適用にあたって，全てのクライアントがサーバにもなりうるピュア型 P2P を想定した場合には効率が悪いので，本研究では監視用のノードを用いたハイブリッド型 P2P への適用を試みた．

2. Zyzzyva

ビザンチン故障は，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，これに対応できれば他のクラスの故障に対しても対応できることになる．ビザンチンアルゴリズム[1-3]は，ビザンチン故障に対処するためのアルゴリズムである．その中で最近注目を集めている投機的ビザンチンアルゴリズム[4-6]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アルゴリズムに Zyzzyva がある．Zyzzyva では，通常のビザンチンアルゴリズム同様，f 個の故障ノードに対処するためには 3f+1 個のノードが必要となる．非同期システムでは，1 個のクラッシュ故障に対してさえ対処することが出来ないことが知られているが，Zyzzyva では実用上の観点からタイムアウトの設定により対処している[4-5]．Zyzzyva では，先ず，クライアントから 3f+1 個の同一データを保管したサーバ群のプライマリサーバ役のサーバに対して要求が出される．次にプライマリサーバはそのレプリカに対してこの要求を伝える．すると，プライマリサーバとレプリカは，クライアントに対してレスポンスデータを返し，これらのレスポンスデータを基にクライアントで合意が取られる．この時，3f+1 以上からのレスポンスデータが同一であれば，合意が成立し，その結果は正しいと考える．しかしながら，合意が成立しなければ，次に故障対処の処理を行なう．

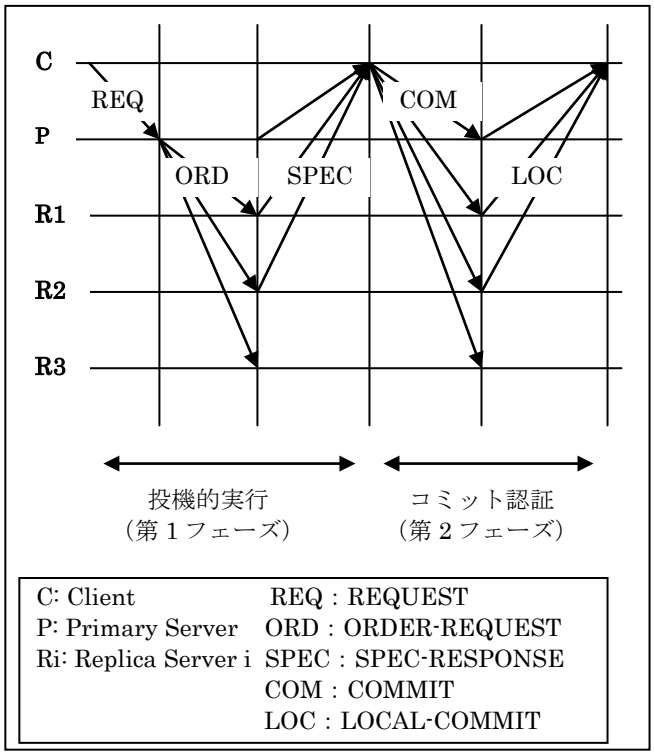

図 1　Zyzzyva での動作

Zyzzyva での処理を以下に示す（図 1 参照）．Zyzzyva では，クライアント C がプライマリ 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，

（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，引き続きクライアント 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 ROCAL-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

†東海大学 大学院 工学研究科　Tokai University

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．

3. Zyzzyva の P2P への適用

Zyzzyva では，本来はサーバのビザンチン故障に対処するために，サーバを複数のレプリカにより構成していたが，本研究では，Zyzzyva を全ノードがクライアント兼サーバとして働く P2P システムに適用する．アプリケーションとしては，ノード即ちピア（peer）同士が行動データ（アクションデータ）をゲームの各ステップ毎に送り合いながら処理を進める分散オンラインゲーム[7-8]のようなものを想定する．但し，ピュア型 P2P では処理効率が悪いため監視ピアを導入したハイブリッド型 P2P を想定する．開発したアルゴリズムの主要部分を図 2 に示す．

今，ピア P1，P2，P3，P4 がデータ d1，d2，d3，d4 を保持しており，P1 がプライマリとしてデータを送る場合を例として，処理の流れを以下に示す．なお，実際には，P2, P3, P4 も各々プライマリとして同様の処理を行い合意を取る，いわゆるインタラクティブコンセンサス問題[1]の処理を行う必要があるが，この処理についての説明は煩雑になるので省略する．

プライマリ P1 は，レプリカ P2，P3，P4 に行動データを送った後，監視ピア M にレスポンスとしてレプリカに送った行動データと同じデータを送る．レプリカ P2，P3，及び P4 は，レスポンスとして監視ピア M にプライマリ P1 から送られて来たデータを送る．なお，処理の効率化のため各ピアでは他のピアからの行動データが全てそろったらベクトル化し，監視ピア M に送ることにする．監視ピア M で全てのベクトル化されたデータが一致すれば処理は終了する．

[第 1 フェーズ]（投機的実行）

（1）P1 が P2，P3，P4 に対してデータを送る．

P1→P2，P3，P4：　＜ u, n, hn, d, o, t, p ＞

但し，u：ビュー番号，n：シーケンスナンバ，
hn：履歴（hn=H(hn-1,d)），o：行動データ，
t：タイムスタンプ，p：行動データの送信ピアの ID，
d：o, t, p からハッシュ化したデータ（d=H(o, t, p)）．

```
variables
 Pi: sender peer,  Pj: receiver peer j ( j != i )
 M:monitor peer ,  V: view number
 Si, Sj: sequence number in peer i, j
 NHi, NHj: history of peer i, j
 H():hash function
 Oi:operation data of Pi in game
 TSi, TSj: time stamp of i, j
 LRP:list of replied peer ID
 CountERRi , CountERRj:error count in peer i, j
 Quej:queue of Pj
 r: received data from sender
     ( equivarent to H(Oi,TSi,Pi) if correct )

@request peer // Pi
(1) Pi send < Pi,Si,NHi,H(Oi,TSi,Pi),Oi,TSi, V >  to all Pj, M;
(2) if(Pi cannot receive COMMIT from M within timeout )
(3)      Oi is stored in vector data;
(4) else
(5)      Pi send LOCAL-COMMIT to M;
(6)      Oi is stored in vector data;
(7) endif

@each receiver peer // Pj
(8)  Pj receive < Pi,Si,NHi,H(Oi,TSi,Pi),Oi,TSi , V > from Pi;
(9)  <TSi,Pi>  put into Quej;
(10) Pj send < Pj,Sj,NHj,H(r),TSj,r,Pi,Si,NHi,H(Oi,TSi),Pi, V >
                                                          to M;
(11) if( Pj cannot receive COMMIT from M within timeout)
(12)      Oi is stored in vector data;
(13) else
(14)     Pj send LOCAL-COMMIT to M;
(15)     if(Pj receive COMMIT Ⅱ from M)
(16)           Pj send LOCAL-COMMIT Ⅱ to M;
(17)           Oi is stored in vector data;
(18)     endif
(19)endif

@monitor peer
(20) if(M cannot receive true response from all peers)
(21)  M send  <COMMIT, M, LRP>  to all peers;
(22)     if(M receive all true LOCAL-COMMIT)
(23)          CountERRj = CountERRj ＋ 1;  (Pj not within LRP);
(24)     else
(25)          CountERRi = CountERRi ＋ 1;
(26)          M send
                 < COMMIT Ⅱ ,Si,NHi,H(Oi,TSi,M),Oi,TSi ,V>
                                      to all peers not within LRP;
(27)     endif
(28) endif
```

図 2　開発したアルゴリズムの主要部分

（２）P1 と，データを送られた P2，P3，P4 は監視ピア M に SPEC-RESPONSE を送る．
P1，P2，P3，P4→M：
＜ SPEC-RESPONSE, u, n, hn, r, H(r), p, t, I, r, OR ＞
但し，r：応答（P1 が送った来た内容，つまり d1），
i：自分のピア ID，OR：P1 の< u, n, hn, d >

監視ピア M が，全てのレプリカ（P1～P4）から正しい SPEC-RESPONSE を受け取ることが出来たのなら処理は第 1 フェーズで終了する．しかし，いずれかのピアから SPEC-RESPONSE を受け取ることが出来なかった場合，又は，受け取ることは出来たが内容が正しくなかった場合には第 2 フェーズを行う．

[第 2 フェーズ]（異常発生報告と故障ピア診断）
監視ピア M が任意のピア，例えば P4 から SPEC-RESPONSE を受け取ることが出来なかったとする．この場合，データの送り主である P1 か，SPEC-RESPONSE を返して来なかった P4 のいずれかが故障していたことになる．
（１）監視ピア M が全てのレプリカ（P1～P4）に対して COMMIT を送ることで，異常が発生したことを伝える．
M→P1,P2,P3,P4：＜ COMMIT, m, CC ＞
但し，COMMIT：m：監視ピアの ID,
CC：SPEC-RESPONSE を返してきた 2f＋1 個のピア ID のリスト．

（２）COMMIT を受け取った全てのレプリカ（P1～P4）は，監視ピアに対して LOCAL-COMMIT を送る．
P1,P2,P3,P4→M＜LOCAL-COMMIT,u,d,hn,I,m＞
LOCAL-COMMIT を受け取ることによって，監視ピア M は P1 と P4 のいずれが故障していたかを判別することができる（理由は付録参照）．
もしここで，P1 の故障のために P4 が正しいデータを受け取っていなかったと判別された場合には，第 3 フェーズに進む．

[第 3 フェーズ]（プライマリピア故障の場合の対応）
行動データの送信者である P1 の故障のために P4 が正しいデータを受け取っていなかったと判別された場合には，監視ピア M が P4 に対して P1 から送られるはずだった行動データを送る．
（１）監視ピア M が P4 に正しいデータを送る．
M→P4：＜ COMMITⅡ, u, n, hn, d, o, t, m ＞

（２）P4 は COMMITⅡへの応答として，
LOCAL-COMMITⅡを送る．
P4→M：＜ LOCAL-COMMITⅡ, u, n, hn, m, t, o, r, i ＞

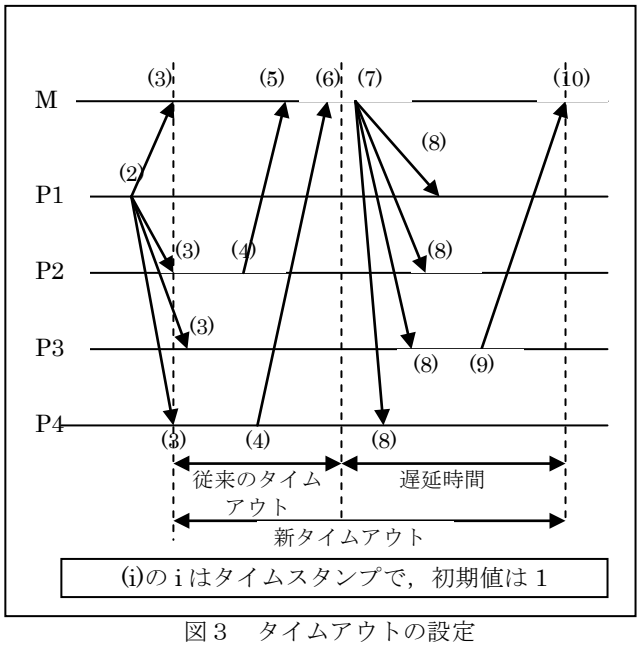

図３　タイムアウトの設定

非同期システムでは，１個のクラッシュ故障に対してさえ対処することが出来ないことが知られているが，Zyzzyva では実用上の観点からタイムアウトの設定により対処している[4-5]．この場合問題は，タイムアウトの時間の設定方法だが，本研究においては，図３のようにタイムスタンプを利用することにより，対処することにした．図３では，P3 が SPEC-RESPONSE を送って来ないために，故障が発生したとして判断され，監視ピア M から COMMIT が送られている．だが，P3 は実際には故障していたわけではなく遅延していただけであったために，後に遅れて SPEC-RESPONSE を返してくる場合を想定している．この場合，監視ピア M では，P3 の処理の遅延のために SPEC-RESPONSE を返してこないのか，或いは，P3 での故障のために送り返してこないのかを判断することができない．そこで，タイムアウトを設けて，ある一定時間が経過した時点で SPEC-RESPONSE 返しこないなら P3 を故障と見なし COMMIT を送ることにする．タイムアウトの設定は，初期値として適当な値を用意しておき，もし，この値を超えて P3 から SPEC-RESPONSE が返されてきた場合には，この時間を新たなタイムアウト時間とする．

5. おわりに

本研究では，クライアントと複数のレプリカからなるサーバ群との間でのビザンチン故障に対処するために提案されている，投機的ビザンチンアルゴリズムの１つである Zyzzyva を，P2P システムに適用するためのアルゴリズムの改良を行った．但し，処理の効率化のために監視ピアを設けており，ハイブリッド型 P2P を想定している．このアルゴリズムの適用領域としては，ピア同士が行動データ（アクションデータ）を送り合いながら処理を進める分散オンラインゲームのようなものを想定している．

今後の課題としては，まず，故障を起こしたピアに関する処理方法について更なる検討が必要であると考えられる．また，全体の動きを検証するために，シミュレーションを行う予定である．

参考文献

[1] 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.

[2] L. Lamport, R. Shostak and M. Pease, The Byzantine generals problem, ACM Trans. Program. Lang. Syst., 4, 3, pp.382-401, 1982.

[3] M. Pease, R. Shostak and L. Lamport, Reaching agreement in the presence of faults, JACM, 27, 2, pp.228-234, 1980.

[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, and: Zyzzyva: Speculative Byzantine Fault Tolerance, Proc. SOSP'07, pp.45-58, 2007.

[6] A. Wright: Contemporary Approaches to Fault Tolerance, CACM, 52, 7, pp.13-15,2009.

[7] N.E. Baughman and B.N. Levine: Cheat-proof playout for centralized and distributed online games, Proc. IEEE INFOCOM2001, pp.104-113,2001.

[8] E. Cronin, B. Filstrup and S. Jamin: Cheat-proofing dead reckoned multiplayer games, Proc. 2nd Int'l conference on ADCOG, 2003.

付録

LOCAL-COMMIT を受け取ることによって，監視ピア M は P1 と P4 のいずれが故障していたかを判別することができるのは次のような理由による．

LOCAL-COMMIT に含まれている d は，P1 から受け取ったオペレーション，タイムスタンプ，オペレーションを送ったピア名を基にハッシュ化したものであり，行動データの送信者 P1 からデータを受け取ったという証拠である．もしも P4 が P1 からデータを受け取っていないのなら，正しい d を監視ピア M に返すことは不可能であり，逆に正しい d を返して来たということは P1 からデータを受け取っているということになる．P1 からデータを受け取ることのできるのは監視ピア M の介入以前では，最初に P1 が他の全てのピアにデータを送った第１ステップのみである．この時に P1 からデータを受け取れているということは，P1 が故障を起さずに P4 に正しいデータを送ったためということになる．つまり，P4 は正しいデータを受けているのに SPEC-RESPONSE を返さない故障ピアであったと判断できる．

逆に，P4 の返してきた d が正しくなく，hn が他ピアより 1 つ分前のものであった場合には，P1 が最初に行動データを P4 に送っていなかったために，P4 では今回の d が得られないため履歴に d を加えることができず，hn が 1 つ分前のままであったということになる．この場合は，行動データを全てのピアに送っていなかったということから，P1 が故障していたものと判断できる．