A-001

ランポート時計を利用したハイブリッド P2P 型 分散チェックポイントアルゴリズムの評価

○東 瞭斗[*1]　守屋 宣[*2]
Ryoyo Azuma　Sen Moriya

1. はじめに

分散アプリケーションに耐故障性を持たせる手法として，定期的にプロセスの状態保存 (チェックポイントの取得) を行う手法について考察する．プロセス間で相互に通信を行う分散アプリケーションでは，各プロセスが独立にチェックポイントの取得をするだけでは，状態の一貫性を維持したままチェックポイント状態へロールバックさせることができない．そのため，プロセス間で連携してチェックポイントを取得する必要がある．

ハイブリッド P2P システムのサーバと連携し，Lamport が提案した論理時計 [1](以下，ランポート時計) を利用してチェックポイントを取得する手法を文献 [2] で提案している．時計を利用してチェックポイントを取る手法は，チェックポイントのためのメッセージ交換が必要ない．また，物理時計を使わずに論理時計を利用する手法では，アプリケーションの動作がない限りはチェックポイントを取得しなくてよく，時計の値が大きくなりすぎることがないといった特徴がある．文献 [2] の手法に基づいて文献 [3] でサーバが複数の場合における考察を行った．

本稿では，文献 [3] に基づいてシミュレーション実験を行い，サーバが複数の場合における分散チェックポイントアルゴリズムについて評価を行った．

2. 諸定義

メッセージ交換により相互に通信を行う複数のピアとサーバから構成されるハイブリッド P2P システム (図 1) を考える．ピアとサーバの間の通信およびサーバ間の通信の通信遅延は無視できるほど小さいが，ピア間の通信はメッセージが欠落することはないが通信遅延の保証はないとする．各ピアは物理時計を参照することはないが，タイマーにより一定時間 τ_{slp} の間にアプリケーションの動作が全くなかったことを検知できるとする．

システムの状況の遷移をシステムの実行，システム内のピアのメッセージ送受信や内部状態の変化をイベントとよぶ．システムの実行に現れるイベント間の前後関係 [1] を，以下のように定義する．

定義 1　実行 E に現れるイベントの集合を $\mathcal{E}^E$ とする．このとき，イベント $e, e' \in \mathcal{E}^E$ に対し，(1)〜(3) のいずれかが成り立つとき $e \to e'$ とする．(1) e と e は同じプロセスで e, e' の順に生起している．(2) e が送信イベント，e' が同じメッセージの受信イベントである．(3) (1)(2) の推移律が成り立つ．□

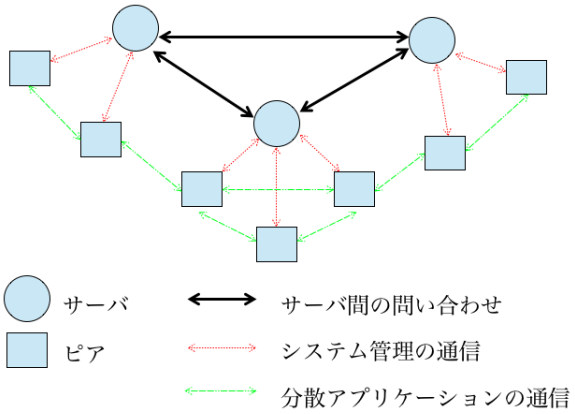

図 1　ハイブリッド P2P システム

このとき，無矛盾な切断を次のように定義する．

定義 2　実行 E の無矛盾な切断 c とは，

$$\forall e, e' \in \mathcal{E}^E : (e \in \mathcal{E}^E_{F(c)}) \wedge (e \to e') \Longrightarrow e' \in \mathcal{E}^E_{F(c)}$$

を満たすような $\mathcal{E}^E$ の 2 つの部分集合 $\mathcal{E}^E_{P(c)}$ と $\mathcal{E}^E_{F(c)}$ への分割である．□

無矛盾な切断は，各ピアで生起するすべてのイベントに対し，あるイベント e_t 以前に生起するイベントを $\mathcal{E}^E_{P(c)}$ に，e_t より後に生起するイベントを $\mathcal{E}^E_{F(c)}$ に分割することにより考えられる．このとき，各ピアの e_t 直後の状態に基づいて，必ずしもシステムの状況として現れているわけではないがピア間のメッセージ送受信の一貫性が維持されている，**無矛盾な状況**を構成できる．無矛盾な状況を構成できるならば，あるピアで故障が発生したとしても，故障発生前の無矛盾な状況にロールバックができるようになる．

システムの任意の実行 E のイベント e に対し，

$$\forall e, e' \in \mathcal{E}^E, e \to e' \Longrightarrow c(e) < c(e')$$

を満たすように e に割り当てられる値 $c(e)$ を論理時計と呼ぶ．論理時計をもとにすると，ある論理時計値 T に対し，実行 E のイベント集合 $\mathcal{E}^E$ の $c(e) < T$ なるイベント e は $\mathcal{E}^E_{P(c)}$ に，$c(e) \geq T$ なるイベント e は $\mathcal{E}^E_{F(c)}$ に属するような切断 c は無矛盾な切断となる．代表的な論理時計に，ランポート時計がある．ランポート時計は非負整数であり，送受信されるメッセージにその送信イベントの時計の値を添付する．各ピアの送信イベント，内部イベントのときの時計値は，そのピアの直前の

[*1] 近畿大学大学院 総合理工学研究科
[*2] 近畿大学 理工学部 情報学科

アプリケーションイベント e に対して $c(e)+\alpha(\alpha \geq 1)$ となり，受信イベントではそのピアの直前のイベント e と受信メッセージに添付された時計値 $M.c$ に対して $\max(c(e), M.c)+\alpha(\alpha \geq 1)$ となる．

3. ハイブリッド P2P システムにおけるチェックポイント取得

前節で述べたとおり，ランポート時計を使えば無矛盾な状況を構成できる．そこで，あらかじめ無矛盾な状況を構成するランポート時計の間隔 t_δ を定めておき，ランポート時計が初めて $k \cdot t_\delta$ 以上 $(k = 0, 1, 2, \ldots)$ になるイベントの直前にチェックポイントを取得するようにすれば，ピアが一貫性を維持したロールバックに利用できるチェックポイントを取得できる．以下，ランポート時計 $k \cdot t_\delta$ で取得されるチェックポイントをラウンド k のチェックポイントとよぶ．

ロールバックをするときは，次のように行う．ピア P_i がラウンド k のチェックポイントを取得してからラウンド $k+1$ のチェックポイントを取得する前に故障したとする．このとき，ピア P_i 及び既にラウンド k のチェックポイントを取得しているピアは，ラウンド k のチェックポイントから動作を再開する．その他のピアは過去の状態に戻らなくても，一貫性は維持される．

ランポート時計の調整は以下のように行う．

(a) サーバを利用した調整

ラウンド k までのチェックポイントを取得済みのピアが次のチェックポイントを取得しようとするとき，サーバに既に取得されているチェックポイントのラウンドの最大値 $k_{\max}$ を問い合わせ，ラウンド $k_{\max}$ または $k+1$ のチェックポイントとして取得する．

(b) 長時間イベントがないときの調整

ピアでタイマーの時間 τ_{slp} 以上イベントが生起しなかったとき，ピアで次のイベントが生起したときにもチェックポイントを取得するようにする．

(a)(b) のとおりにチェックポイントを取得することで，あるピアで最初にラウンド k のチェックポイントが取得された実時刻から $t_\delta \cdot \tau_{\mathrm{slp}}$ 経過するまでにラウンド k のチェックポイントがロールバックに利用できるようになる [2].

サーバの負荷を小さくするために，サーバを複数用意し各サーバが管理するピアを分割した場合には，以下の調整を追加で行う [3].

(c) サーバ間による調整

(a) のとおり調整を行った際，サーバから他のサーバにも問い合わせを行うことで各サーバの $k_{\max}$ を確認し，$k_{\max}$ の最大値をそのサーバの $k_{\max}$ として調整を行う．

(d) 更新頻度による調整

(c) におけるサーバから他のサーバに対しての問い合わせを $k_{\max}$ が更新される度ではなく，一定回数更新される度にすることで，サーバ間の通信頻度を減らし、サーバの負荷を小さくする調整を行う．

4. シミュレーション実験と結果

サーバ数 1 または 5，ピア 1000 から成るハイブリッド P2P システムで，t_δ を $15, 25, 35$，τ_{slp} を 30 秒として，チェックポイント取得法を評価するシミュレーション実験を行った．各ピアは，1 回のイベントの後，$6 \sim 29$ 秒後に送信，受信，または内部イベントが生起するか，$1/25$ の確率で $30 \sim 180$ 秒の間動作をせずに停止し続けるとする．$1/100$ の確率で重要な計算が行われると仮定し，その計算が行われた場合は時計値に関係なくチェックポイントを取得するものとする．チェックポイントを取得する際に前節で述べた調整を行う．

この条件のもとで，60 分 10 回ずつ動作させる実験を行い，チェックポイント取得からロールバックに使えるまでの時間の平均を評価をした．調整は (a),(b) を行い，(c),(d) を行うかどうかで比較をする．実験結果を表 1 に示す．実験結果より，t_δ の値が 15 の場合ではサーバ 1 つが一番良い結果となっているが，t_δ の値が大きくなるにつれ，(c),(d) の調整を行う方がより新しいチェックポイントを利用可能となっている．

表 1　実験結果

サーバ	調整	$t_\delta = 15$	$t_\delta = 25$	$t_\delta = 35$
1	(a),(b)	449.3	597.0	631.5
5	(a),(b)	474.73	575.5	610.6
	(a),(b),(c)	461.56	564.71	595.57
	(a),(b),(c),(d)	473.8	569.24	592.73

5. まとめ

本稿では，ランポート時計を利用したハイブリッド P2P 型分散チェックポイントアルゴリズムに対してサーバが複数ある場合におけるシミュレーション実験，評価を行った．サーバを複数用意した場合において調整を工夫することで，サーバの負荷を小さくしつつ効率の良いチェックポイントを取得できることを確認できた．

参考文献

[1] L. Lamport: Time, Clocks, and the Ordering of Events in a Distributed System, *Communications of the ACM*, Vol. 21, No. 7, pp. 558–565 (1978).

[2] 守屋 宣：ハイブリッド P2P システムに対するランポート時計を利用したチェックポイント取得とロールバック, 信学技法 COMP2016-1, pp. 1–8 (2016).

[3] 東 瞭斗, 守屋 宣：ハイブリッド P2P システムに対するランポート時計分散チェックポイントアルゴリズムに関する考察, 平成 29 年電気関係学会関西連合大会, pp. 424–425 (2017).