DEIM Forum 2009 E7-2

入れ子トランザクションの性能評価

櫻井　雅志† 　三浦　孝夫†

† 法政大学 工学部　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †masashi.sakurai.jk@eng.hosei.ac.jp, miurat@k.hosei.ac.jp

あらまし　一般に長時間トランザクションは、異常終了時の回復コストが非常に高く，この対応として入れ子トランザクション概念が提案された。しかし，入れ子トランザクションの処理性能は状況や環境に大きく依存するため，処理性能の予測が困難である。本稿では、入れ子構造の複雑さ、通信状況や競合の相互依存性の影響を調べるため、分散環境下での性能評価を行う。

キーワード　入れ子トランザクション 性能評価

Performance Evaluation of Nested Transaction

Masashi SAKURAI† and Takao MIURA†

† Dept.of Elect.& Elect. Engr., HOSEI University　3-7-2, KajinoCho, Koganei, Tokyo, 184–8584 Japan
E-mail: †masashi.sakurai.jk@eng.hosei.ac.jp, miurat@k.hosei.ac.jp

Abstract　Huge transactions take large costs in the abort. Therefore, there has been proposed nested transaction. But a processing performance of nested transaction depends on situations and environment. It is difficult to predict response times. In this paper, we simulate nested transaction in distributed computing environment, and show influence of structures, situations and conflicts.

Key words　Nested Transaction, Performance Evaluation

1. 前　書　き

トランザクション機構は、現在さまざまなシステムで用いられている。トランザクションは分散環境で障害に対処でき、高い信頼性を得ることができる。[2]

トランザクションは、信頼性を得るために ACID と呼ばれる特性が提案されている。。

また、この特性を満たすために、施錠、時刻印、2 相コミット、シャドウページングなどが用いられている。しかしまだなお、さまざまな問題が指摘されている。例えば、長時間トランザクションでは、数分から数時間かかるトランザクションの最後にアボートされた場合、今までの処理は無かったことになる。また、長時間施錠し続ける。

そこで Moss [1] らにより、入れ子トランザクションが提案されている。入れ子トランザクションは従来のトランザクションとは異なり、階層構造を持つ。また、この階層構造は木構造としてとらえることができる。この特徴により、子トランザクションに応じた処理が可能になり、部分的再実行、代替処理、きめ細かい同時制御などを行うことができる。

入れ子トランザクションでは、従来のトランザクションと共通する問題もあるが、入れ子トランザクション特有の現象や問題がある。例えば、子がトランザクションがコミットしたあとに、親トランザクションがアボートしたとき、従来の ACID 特性では、他のトランザクションから見た親トランザクションの原子性と子トランザクションの永続性は相反する。[3] このため、単純なトランザクション機構の処理性能を予測することが容易ではなく、パラメタ設定などで大きな効率の差を生む。

入れ子トランザクションは内部状態に応じて処理を行うため、状況や環境に大きく依存する。本稿では、入れ子トランザクション構造、通信や競合の相互依存性の影響を調べるため、分散環境下での性能評価を行う。

2 章では入れ子トランザクションの概略を述べ、3 章で入れ子トランザクション実現機構について述べる。4 章で性能評価方法を示し、5 章で実験・考察を行い、6 章で結びとする.

2. 入れ子トランザクションと従来のトランザクション機構

入れ子トランザクションの概要について述べる。

2.1 トランザクション

従来のトランザクション機構は、以下の ACID 特性を満たす。原子性 (A) とは、トランザクションが、成功（コミット）するか失敗（アボート）するかのどちらかであり、アボートは無かったことにしなければならない性質のことであり、一貫性 (C) とは、トランザクションは整合条件を満たす性質である。

また、隔離性 (I) とはは、並列動作するトランザクションが、他のトランザクションから見えない性質である。耐久性 (D) は、コミットした場合、結果は永遠に消えない性質である。

この 4 つの ACID 特性により、複数のトランザクションが同時に実行する環境では、それが直列した実行と同じ効果を生むという直列可能性が、トランザクション機構の結果の正しさを保証する。

ACID 特性を満たすために、排他制御に施錠を用いることが多い。施錠は以下のルールが用いられる。読み込み、書き込みには、それに応じたロックを持っている時に行なう。誰もロックを持っていなければ、書き込みロックを持つことができる。誰も書き込みロックを持っていなければ、読み込みロックを持つことができる。ロックはいつでも開放できる。これらのルールを用いた施錠により、他のトランザクションは待機させられ、競合状態が生じないことが保障される。

2.2 入れ子トランザクション

入れ子トランザクションは、内部に別のトランザクションを含むことができる。また、この構造は木構造としてとらえることができる。つまり、トランザクションは、複数の子トランザクションを持つことができ、ただ一つだけ親トランザクションを持つ。子を持っていないトランザクションを葉トランザクションと呼び、子を持つトランザクションを内部トランザクションと呼ぶ。また、深さが 0 レベルのトランザクションをトップレベルトランザクションと呼ぶ。

2.3 入れ子トランザクションの ACID 特性

入れ子トランザクションでは、以下の ACID 特性に拡張される。原子性は従来と同様であり、一貫性はトランザクションは整合条件を満たす性質である。隔離性は、拡張され親トランザクションは自分の子トランザクションに限り見ることができる。耐久性では、トップレベルトランザクションがコミットした場合に限り永続的である。

また、施錠は以下のように拡張されたのルールが用いられる。読み込み、書き込みは、それに応じたロックを「持っている」時に行なう。そして、誰もロックを持っていなく、かつ、全てのロックを直系の先祖が「維持」していれば、書き込みロックを持つことができる。誰も書き込みロックを持っていなく、かつ、全ての書き込みロックを直系の先祖が維持していれば、読み込みロックを持つできる。子がコミットするとき、子が「持っている」ロックと「維持」しているロックを親に継承する。親は継承したロックを維持する。トランザクションがアボートするときは、単純にロックを開放する。

例えば、以下の木構造を考える。T1 はトップレベルトランザクションであり、T1 は T1.1 と T1.a の子トランザクションを持ち、T1.1 は T1.1.b と T1.1.c の子トランザクションを持つ。T1.1.b がコミットしたとき、その結果は T1.1 と T1.1.c からは見えるが T1、T1.a や他のトランザクションからは見えない。

また、ロックの流れは以下の通りである。また図 1 に示す。T1.1.b が書き込みロックを取得し、「持つ」。T1.1.b がコミットし、T1.1 はロックを継承し、ロックを「維持」する。このとき T1.1 がアボートすると T1.1 は維持しているロックを開放する。T1.1 がアボートしないときを考える。T1.1.c は T.1 からロックを取得し持ち、T1 はロックをそのまま維持する。T1.1.c がコミットすると、T1.1 はロックを継承し、ロックを「併合」する。

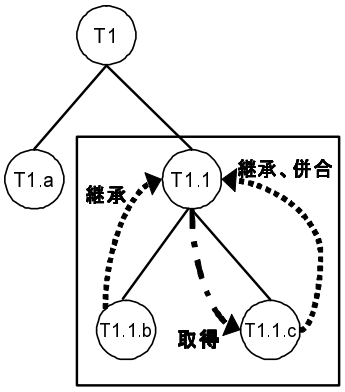

図 1　ロックの流れ

3. 入れ子トランザクション実現機構

本章では、入れ子トランザクションの実現機構について述べる。

3.1 実 現 機 構

システムには、複数のノードが存在する。ノードは、演算装置、通信回線、揮発性メモリ、永続性メモリからなる。ノードは、複数のトランザクションと一つのトランザクションマネージャが存在する。

葉トランザクションは、演算処理を行うことができる。この演算処理は、各ノードで 1 サイクルあたり 1 回行うことができる。演算処理を行う必要が無ければ、行わない。また葉トランザクションは、データの読み込み及び書き込み、コミット、アボートを行う。読み込み、書き込み、コミットはブロックされる可能性がある。

内部トランザクションは、さまざまな情報を元に判断を行うことができる。例えば、子トランザクションの状態や終了状態、経過時間やデッドライン、ノードの状態や負荷状態がある。しかし、その情報を元にできる判断は多くはない。内部トランザクションが行う判断は、わずか 4 つだけである。子を生成するかアボートさせる。自分をコミットするかアボートする

各ノード、トランザクションはメッセージを送受し、トランザクションを実行していく。まず、ノードが「create transaction」メッセージを受け取り、トップレベルトランザクションが生成される。トップレベルトランザクションは、「create subtransaction」により、内部か葉トランザクションを生成する。葉トランザクションは「lock」、「unlock」、「inheritance」といったメッセージをやり取りする。最後に、コミットした場合は親に「commited」を通知し、親トランザクションは判断を行う。

3.2 実現機構の問題

従来のトランザクション処理システムには無い問題が生じる。一つ目は、トランザクションが構造を持つことにより、その構

造内での通信の必要性が生じる。このとき、自分の子トランザクションが同一ノードにいる保障はなく、他ノードであれば遅延が発生する。また、施錠におけるロックの取得、継承も同様の問題が起こる。二つ目に、木の構造、形状による差異が存在する。内部や葉トランザクションの数、偏りや深さによって差が生じる。

4. 性能評価

我々は、シミュレーションにより、性能評価を行う。シミュレーションは、MPI の実装である MPICH2 を用いて行った。MPI により各ノードの時刻やメッセージの同期を行い、性能等に依存しない評価を行うことができる。

4.1 評価方法

我々は、葉トランザクションの深さを変えて、トップレベルトランザクションのコミット時のサイクルを比較する。同じシミュレーションを 8 回行い、コミット時間の平均を各深さで比較する。

以下に我々が用いるシミュレーションについて述べる。

4.2 仮想時間

我々は、本シミュレーションに仮想的な時間「サイクル」を導入する。サイクルは整数である。システムの全ての処理時間や遅延時間といった全ての時間が、全てサイクルで表される。サイクルを導入することにより、コンピュータの性能や回線の帯域に依存せずに評価することができる。本実験ではサイクルを消費するのは葉トランザクションの演算処理とした。例えば、回線遅延が 3 サイクルとする。サブトランザクションの生成では、自ノードであれば、子トランザクションは直ちに生成される。しかし他ノードでは、回線による遅延が起き、生成されるのは 3 サイクル後である。また、同時に複数のサブトランザクションを生成する場合でも、同様に 3 サイクル後である。しかし、葉トランザクションの演算処理は異なる。同一ノードでは、並列演算は起きない。7 サイクルと 5 サイクルの長さを持つ葉トランザクションは 12 サイクル後には両方終了している。ここで、スケジューラの問題が起こるが、本シミュレーションでは単純に先に生成された実行可能なトランザクションを実行する。異なるノードで実行された場合、7 サイクル後及び 5 サイクルに終了するがその終了を他ノードが知るのは 10 サイクル及び 8 サイクル後である。

4.3 木構造

入れ子トランザクションの木構造はさまざまな形状が考えられる。本シミュレーションでは、特に深さによる影響を調べる。そこで次の簡潔な構造を考える。

- サブトランザクションは、その全ての子トランザクションを同等に扱う。
- 葉トランザクションは全て同一レベルである。
- 木構造に偏りが少ない。

木構造に偏りがまったく無いのは完全 N 分木であるが、様々な条件を満たしながら深さを変えることはできない。実験では、全ての内部トランザクションが持つ子は N 又は N-1 の時に木構造に偏りが少ないとし、これを用いる。

また、本シミュレーションでは、子トランザクションの終了状態に着目して実験を行う。終了状態とは、コミット・アボート・未終了である。子の終了状態は判断される情報の中でもっとも基本的かつ重要なものである。子の数と終了状態を元にアクション表が書けるが、子トランザクションを同等に扱う場合は以下 3 つのルールいずれかと 2 つのパラメタで記述できる。パラメタ α と β は次の不等式を満たす。

$0 \leqq \alpha \leqq \beta \leqq$ 子トランザクション数

（1） 即時再実行

コミットした子トランザクションが α を超えるまで、アボートした子トランザクションを即再実行する。

（2） 最低限再実行

アボートした子トランザクションが $\beta - \alpha$ を超えたら、その分だけ子トランザクションを再実行する

（3） 再実行なしアボートした子トランザクションが $\beta - \alpha$ を超えたら、アボートする。

上記に加え全ルールで、コミットした子トランザクションが β 以上なら他の子トランザクションをアボートさせて、直ちにコミットする。

また、コミットしたとき、次の不等式を満たす。

$\alpha \leqq$ 子トランザクションのコミット数 $\leqq \beta$

5. 実　　験

5.1 準　　備

本実験では、表 1 のパラメタを用いてシミュレーションを行った。また、施錠の競合による待ちを想定せず、データの読み込み及び書き込みによる待ちはメッセージの遅延時間に依存する。これは、トランザクションは常に悲観的ロックを用いるとは限らないためである。

表 1　パラメタ

並列度	3[ノード]
メッセージ遅延	1-5[サイクル]
葉トランザクションの処理時間	15-25[サイクル]
深さ	1-5[レベル]
葉トランザクションの合計	100
トランザクションの合計	101〜
α	50[%]
β	50[%]

5.2 実験結果

各実験を各深さで 8 回行った。その結果を表 2, 図 2 に示す。今実験では、図 2 より明らかに子トランザクションが深くなるほど、コミットに至るまでの平均時間が急激に早くなっていることが確認できる。また、最小値や最大値を見ても平均時間が早くなっていることが明らかである。

また、平均との差を図 3 に示す。図 3 より、実験毎の差が大きく開かないことがわかる。

表 2　実験結果 時間と深さ（平均、最小、最大）

深さ [level]	平均 [サイクル]	最小 [サイクル]	最大 [サイクル]
1	790.6	779	811
2	608.6	545	675
3	489.4	408	567
4	345.3	302	392
5	162.4	149	173

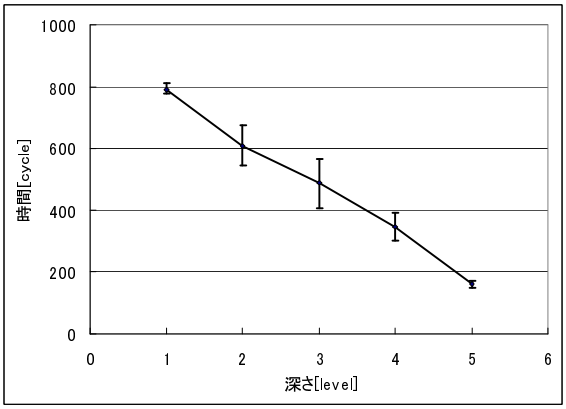

図 2　実験結果 時間と深さ（平均、最小、最大）

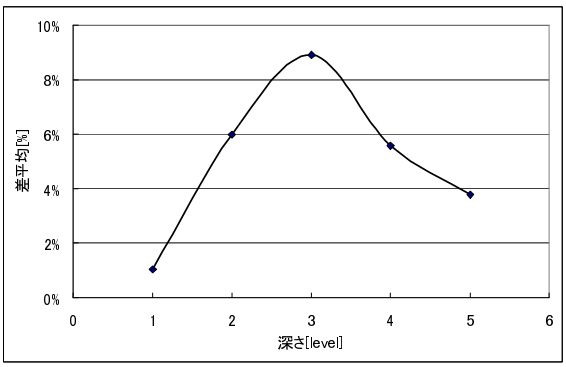

図 3　実験結果 平均との差と深さ

5.3 考　　察

葉トランザクションの数及び深くすると次のようなことが考えられる。トランザクションの生成、コミット・アボートの通知の遅延やサブトランザクション数の増大である。一般にトランザクション数の増大は性能を落とす遠因になるが、入れ子トランザクションでは逆に内部トランザクションがコミットしやすくなったと考えられる。例えば、深さが 2 レベルで葉トランザクションが 16 個であれば、最低でも 4 個のトランザクションのコミットが必要だが、深さが 4 でであれば、一つのトランザクションがコミットしてれば親トランザクションはコミットできる。

また、図 2 に線形性が見て取れる。この線形の x 切片は 6.1[レベル] である。この値は、100 個の葉トランザクションを用いて作られる 2 文木の最大の深さと関係があると考えられる。今実験では、半分の子トランザクションがコミットしたときにコミットを行う。子トランザクションが 4 個であれば 2 個コミットした時点でコミットする。そして、子トランザクションが 3 個のときに、1 個コミットした場合は、50%の確率でコミットする。さらに、子トランザクションが 1 個のときは、子がコミットしていなくても 50%の確率でコミットする。このとき、メッセージ遅延等を無視すれば 0 サイクルでコミットする。任意のトランザクションがコミットしたとき、木全体で子トランザクションを 2 個か 1 個であり、このコミットは連鎖する。子トランザクションが一つになりはじめるのは、5.6 レベルからである。メッセージ遅延によりいくらか深くなり、直線近似により 6.1 レベルという値が出てきたと考えられる。

これらよりは、メッセージの遅延より、深くし内部トランザクションコミットしやすくなったほうが、トップレベルのコミットが早くなると結論付けられる。

6. 結　　び

本稿では、サブトランザクションの有用性を示し、さらにシミュレーションにより葉トランザクション数が一定の元で、葉のレベルを深くすることによるコミットが非常に早くなることを示した。

文　　献

[1] J, Eliot. and B, Moss.: “Nested Transactions: An Approach to Reliable Distributed Computing”, In *Department of Electrical Engineering and Computer Science*, 1981.

[2] Gray, J. and Reuter, A.: “Transaction Processing, Morgan Kaufmann”, 1993.

[3] Lynch, N., Merritt, M., Weihl, W. and Fekete, A.: “Atomic Transactions, Morgan Kaufmann”, 1994.