2003－OS－94　(13)
2003／8／4

分散処理システムにおける処理順序保証法

横山 和俊†　　谷口 秀夫††
†NTT データ 技術開発本部　　††岡山大学 工学部

近年の Web システムでは、提供するサービスの高度化に伴い、異なる機能を提供する Web サーバを連携させて実現する必要がある。このようなシステムでは、各 Web サーバへの処理依頼の到着順と処理順が逆転することを考慮する必要がある。一方、各 Web サーバは、性能向上の目的で負荷分散を行う分散処理システムとして構築されることが多い。したがって、各 Web サーバでは負荷分散を行いながらリクエストの処理順序を保証する制御が必要である。ここでは、負荷分散を伴う場合の処理順序保証について、分散処理システム全体の性能向上を目指したシステム構築法を提案する。また、提案方式の基本性能を示す。

Request Ordering Mechanism in Distributed Systems

Kazutoshi YOKOYAMA†　and　Hideo TANIGUCHI††
†Research and Development Headquarters
††Faculty of Engineering, Okayama University

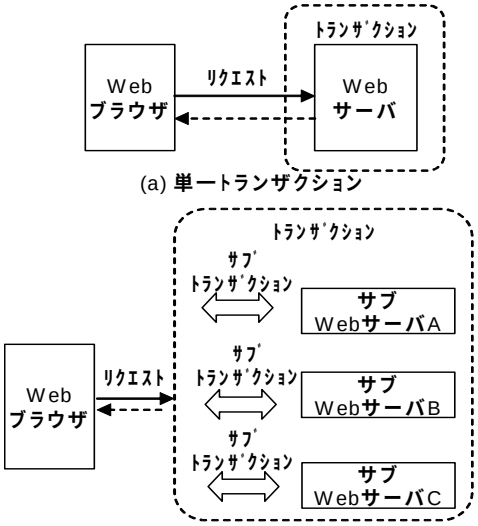
In recent years, the Web system is built combining several Web servers which offer different function, in order to offer advanced services. In each Web server, it is necessary to take into consideration that the order of arrival and the order of processing of a processing request are reversed. On the other hand, since each Web server is built as a distributed processing system with a load balance function. Therefore, it needs to perform control which guarantees the processing order of a request in the environment where load balance is performed. This paper discusses the system configuration to improve performance, which guarantees a processing order with the load balance function. Moreover, the result of performance evaluation of the proposal method is shown.

1. はじめに

近年のインターネットの進展に伴い、多くのサービスがインターネットを利用した Web システムとして提供されている。最近では、情報提供システムだけではなく、例えばチケット予約システムやインターネット証券システムのように性能と信頼性が求められるトランザクションシステムも Web システムの形態で実現されている。

Web システムは、要求（リクエスト）を出す依頼側システムの Web ブラウザと、要求を受け付け処理する Web サーバから構成される。この様子を図１に示す。図 1(a)は最も基本的な形態であり、Web ブラウザと Web サーバが１対１の関係を持ち通信している。このため、Web ブラウザからの 1 つのリクエストが Web サーバ上で 1 つのトランザクションの処理として行われる。つまり、1 トランザクション処理単位で通信パスの設定と開放が行われるため、トランザクション処理の途中で障害が発生しても、その影響を最小限化することができる。一方、提供するサービスの高度化に伴い、Web サーバが行う 1 つのトランザクション処理量が増加している。このため、Web サーバを複数のサブ Web サーバで構成した形態が考案されている[1,2]。ここでは、この形態をサブ Web サーバシステムと名付ける。つまり、図 1(b)に示すように、Web ブラウザからの 1 つのリクエストに基づく処理を複数のサブ Web サーバが行う。この形態では、1 つのトランザクションを複数のサブトランザクションに分割し、各サブトランザクションを処理する複数種類のサブ Web サーバが連携して処理を行う必要がある。この形態では、サブ Web サーバ間の連携内容が Web サーバ全体の機能や性能に大きな影響を与える。

そこで、ここでは、サブ Web サーバで Web システムを構成する場合について、課題と対処を示し、効果的な構成法を述べる。



(a) 単一トランザクション

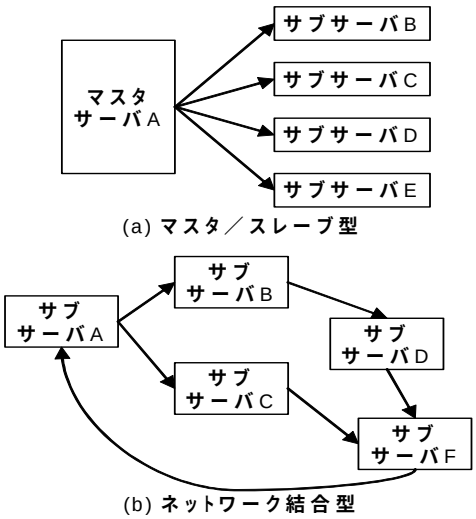
(b) 複数サブトランザクション

図1 Webシステムの形態

2. サブ Web サーバシステム

2.1 基本構成

1 つのトランザクションを複数のサブトランザクションに分割して処理を行うサブ Web サーバシステムの構成法は、図 2 に示すように大きく２つに分類できる。図 2(a)は、各サブトランザクションを行うサブサーバとトランザクション全体を管理するマスタサーバからなる構成であり、マスタサーバは各サブサーバと 1 対１の通信を行う。この構成をマスタ／スレーブ型と呼ぶ。一方、図 2(b)は、各サブトランザクションを行うサブサーバを有機的に結合する構成であり、ネットワーク結合型と呼ぶ。マスタ／スレーブ型は、サブトランザクションの処理に関しマスタサーバとサブサーバが１対１の関係を持つため、障害に強い。しかし、通信量が多い欠点がある。また、マスタサーバに負荷が集中しやすい。一方、ネットワーク結合型は、通信量を抑制でき、かつ各サブサーバの負荷を分散できる。したがって、高性能なシステム構築には、ネットワーク結合型が有効であるといえる。しかし、サブトランザクションの相関を考慮して、サブサーバ間を有機的に結合することが必要である。



(a) マスタ／スレーブ型

(b) ネットワーク結合型

図 2 複数サブトランザクションの実現方式

2.2 課題

ネットワーク結合型のサブ Web サーバシステムが抱える課題を明らかにするため、システム例を取り上げて説明する。

ネットワーク結合型のシステム例として、図３に示す電子受発注システムがある。システムは、発注 Web サーバ、決済 Web サーバ、および受注 Web サーバの 3 つのサブサーバで構成され、受発注サービスを実現している。受発注サービスは、各 Web サーバ上のサブサービス（支払いサービスなど）が結合されて実現されている。外部からのリクエストを契機に受発注サービス実行の実体である受発注トランザクションが開始される。さらに、各 Web サーバ間のリクエストの交換により、各サブサービスでサブトランザクションが実行される。

受発注トランザクションは、以下の手順で処理が行われる。

①受注 Web サーバの「見積り受付サービス」へリクエストを送信する。

②発注 Web サーバの「見積り取得サービス」に見積書を送信する。

③決済 Web サーバの「決済サービス」を経由して、受注 Web サーバの「支払い受付サービス」へリクエストを送信する。

④受注 Web サーバの「発注受付サービス」へ、リクエストを送信する。

⑤決済 Web サーバからの支払い確認を受領する。発注受付を実行し、その後、発注確認書を発注サービスへ送信する。

このような、ネットワーク結合型のシステムでは、サブ Web サーバへのリクエストの到着順序が逆転することがある。例えば、③のリクエストは、決済 Web サーバを経由して受注 Web サーバ

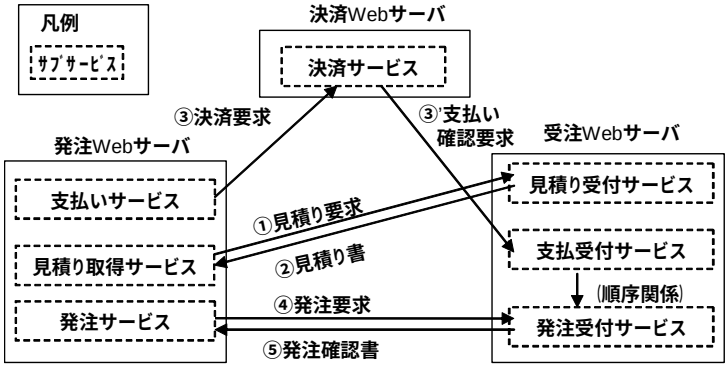


図3 ネットワーク結合型での電子受発注システム

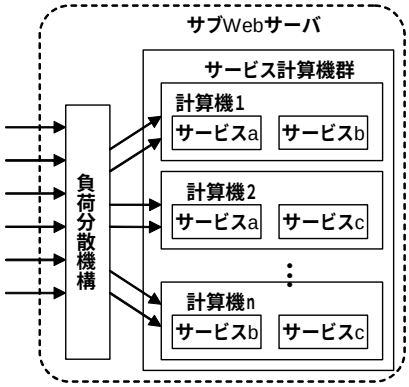


図4 負荷分散が必要な複数サブWebサーバ構成

へ送信されるため、発注リクエスト④と③'の到着が逆転する場合がある。しかし、サービス提供のためには予め定められた処理の順序にしたがって処理する必要がある。つまり、処理順序を保証する制御を行うことが必要である（課題１）。一方、ネットワーク結合型のサブ Web サーバシステムにおいては、図 3 の受注 Web サーバのように、複数のサブトランザクションを行うサブ Web サーバの負荷が大きくなりやい。このため、システム全体の性能向上のためには、高負荷になりやすいサブ Web サーバを、複数の計算機で構成し負荷分散を制御する必要がある（課題２）。この様子を図４に示す。負荷分散を行うサブ Web サーバは、サービスを提供する計算機群と、計算機への負荷分散を行う負荷分散機構から構成される。

なお、Web サーバを複数の計算機で構成した際の負荷分散方式として、様々な方式が提案されている[3,4,5]。最も単純な方式であるラウンドロビン方式は、各計算機が処理するリクエスト数が等しくなるようにリクエストを分散する。また、各計算機の現在の処理量（CPU 使用量など）をもとに負荷分散する方式がある。しかし、これらの方式は、図 1(a)に示した Web サーバを複数の計算機で構成した際の負荷分散方式であり、Web サーバが一種類のサービスを提供するシステムへの適用である。このため、図４に示したように、複数の計算機で複数種のサービスを提供するシステムには適用できない。言い換えれば、これらの負荷分散方式は、単一トランザクションを対象にしたものであり、1 つのトランザクションが複数のサブトランザクションで構成されることを考慮していない。

3. 処理順序保証と負荷分散

ここでは、ネットワーク結合型のサブ Web サーバシステムにおいて、負荷分散（課題２）を行いながらリクエストの処理順序保証（課題１）を行うサブ Web サーバの構成法について述べる。

3.1 要求条件

負荷分散と処理順序保証を行うサブ Web サーバの構成法には、以下の要求がある。

（要求１）複数のサービス種別への対応

Web システムを構成する各サブ Web サーバは、特定のサービスのみを対象として実現されることは少ない。例えば、前節の決済 Web サーバは、様々なサービスで共通的に利用できる機能である。このため、同じサブサービスが、異なるサービスで利用される。結果として、サブ Web サーバでは、サービスの種別により、サブサービスの処理順序が異なってくることを考慮する必要がある。したがって、サブサービスを実現する応用プログラムを変更することなく、サービス種別に応じたサブサービスの処理順序を制御できる必要がある。

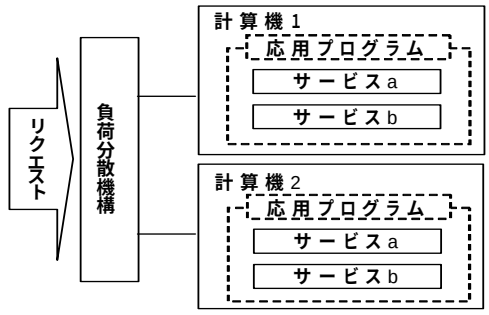
（要求２）サブ Web サーバの独立性への対応

各サブ Web サーバは、異なる組織により構築・運用されることがある。このため、サービス種別の追加や変更が、各サブ Web サーバ内の局所的な変更で行えることが必要である。

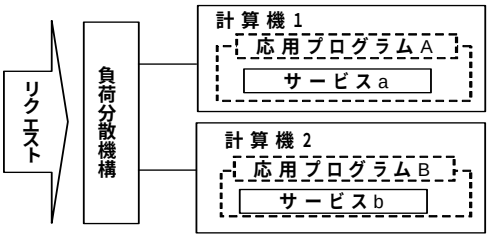
なお、以降では、サブサービスをサービスとして、サブトランザクションをトランザクションと記述している。

3.2 実現上の課題

サブ Web サーバを複数の計算機で実現する構



(a) 均一配置方式



(b) 機能別配置方式

図5 応用プログラムの配置方針

成方法として、サービスを実現する応用プログラムの配置方針により、大きく 2 つに分類できる。一つは、各サービスを提供する応用プログラムをすべての計算機に配置する均一配置方式である。もう一つは、特定のサービスを実現する応用プログラムを各計算機に配置する機能別配置方式である。これらの方式の様子を図 5 に示す。

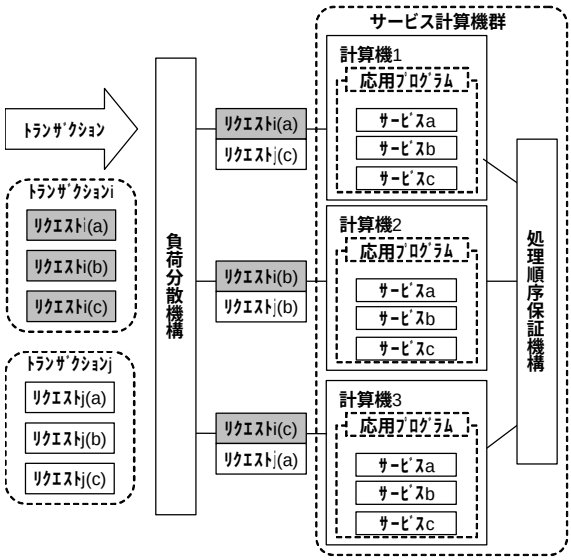
アクセス数増加に伴い計算機を追加する場合、均一配置方式は、現行の計算機と同じ応用プログラムが配置された計算機を追加すればよく、拡張が容易である。一方、機能別配置方式は、応用プログラムの配置を見直す必要があり、かつ、現行の計算機の応用プログラムの構成にも影響があり、均一配置方式に比べ、計算機の拡張が複雑である。このため、システムの再構成を伴うような負荷変動が大きい Web システムの場合、均一配置方式の方が可用性と拡張の容易性にすぐれている。耐故障性の観点では、均一配置方式は、同じサービスを提供する応用プログラムがすべての計算機に配置されているため、1 つの計算機に障害が発生した場合でも、他の計算機で同じサービスの継続が可能である。一方、機能別配置方式は、ある計算機に障害が発生した場合、特定のサービスが中断する。このため、均一配置の方が耐故障性の観点からも優れている。以上のことから、サブ Web サーバを複数の計算機で実現する構成方法としては、均一配置方式が良いといえる。

そこで、以降では、均一配置方式による構成を対象に議論を進める。

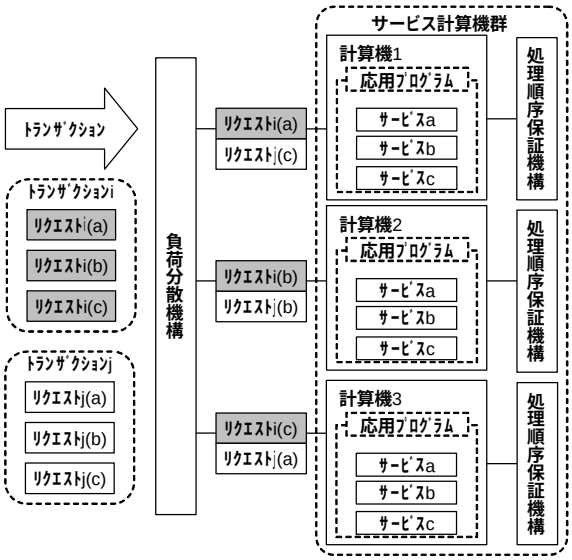
次に、複数の計算機で負荷分散を行う方式は、負荷分散を行う処理の単位により、以下の 2 つに分類できる。

（1）リクエスト単位（R 単位）方式
リクエスト単位で各サーバに負荷分散する。例えば、ラウンドロビン方式により、リクエストの数のみに着目して負荷分散を行う。

（2）トランザクション単位（T 単位）方式
トランザクション単位で各サーバに負荷分散する。具体的には、リクエストの中身を解析し、



(a) リクエスト単位（R単位）



(b) トランザクション単位（T単位）

図6 負荷分散と処理順序保証の構成

同じトランザクションが割り当てられているサーバにリクエストを転送する。

上記2つの方式の様子を図6に示し、負荷分散と処理順序保証の構成方法を以下に説明する。図6では、2つのトランザクションiとjを受け付けたときの動作を示している。トランザクションiとjは、3つのサブトランザクションa,b,cから構成されている。a,b,c の実行要求であるリクエストi(a),i(b),i(c)とj(a),j(b),j(c)により、対応するサービスa,サービスb,サービスcが実行される。R単位方式では、トランザクションを構成するサブトランザクションを意識しないため、異なる計算機に i(a)～i(c)が負荷分散される場合がある。このため、i(a)～i(c)の処理順序を保証するためには、計算機間にまたがる処理順序保証機能が必要である。一方、T単位方式では、i(a)～i(c)は、同じ計算機に負荷分散される。そのため、計算機毎に処理順序保証機能を配置することができる。

表1にR単位方式とT単位方式の特徴を示す。R単位方式は、サービスの実行要求であるリクエストが、異なった計算機へ転送される可能性があり、結果として、異なる計算機上のサービス間での処理順序を調整する必要がある。このため、処理順序保証のための同期オーバヘッドが大きくなる。一方、T単位方式は、負荷分散機構が、トランザクション単位で各計算機に負荷分散する。このため、1つの計算機内でサービスの処理順序を調整すれば良く、同期オーバヘッドは R 単位に比べて小さい。しかしながら、負荷分散機構でリクエストの内容を解析する必要があり、負荷分散機構のオーバヘッドが大きくなる。

計算機間の同期処理は、通信や共有記憶装置を用いて行なうことが必要であり、負荷分散を行うためのリクエスト解析処理に比べて処理時間が大きい。このため、トランザクション単位での負荷分散方式が、性能の観点で有効であると考えられる。

表1 負荷分散と処理順序保証の構成方法の比較

負荷分散方式	長所	短所
リクエスト単位	負荷分散処理において、リクエストの解析が不必要なため、制御が簡単であり、オーバヘッドが小さい。	1つのトランザクション中のリクエストが複数の計算機で処理されるため、複数計算機間での同期処理が必要である。
トランザクション単位	負荷分散処理において、リクエストの解析が必要なため、オーバヘッドが大きい。	1つのトランザクション中のリクエストが必ず同じ計算機で処理されるため、計算機間の同期処理が必要ない。

4. 実現方式

4.1 提案方式

T 単位方式によりリクエストの処理順序を保証する方式を提案する。提案方式でのリクエストの形式を図7に示す。

ServiceName は、サービス種別を示している。TransactionId は、トランザクション識別子を表し、サービス種別内で一意である。RequestName は、リクエスト種別を表している。Data は、サービスへの応用プログラムデータである。

負荷分散機構は、図8に示すトランザクション分散管理テーブル（TLB_Table と略す）を保持する。負荷分散機構は、リクエストを受け取ると、以下の動作を行う。

[Step1] リクエストを解析しサービス種別とトランザクション識別子を取得する。

[Step2] ステップ1で取得したサービス種別とトランザクション識別子に該当する項目が、TLB_Table 内に存在するかを判定する。

[Step3-A] TLB_Table に存在する場合、該当する計算機名を取得し、リクエストを計算機に転送する。

[Step3-B] TLB_Table に存在しない場合、テーブルにエントリを追加し、計算機を選択し、リクエストを転送する。

```
<Transaction>
<ServiceName>サービス種別</ServiceName>
<TransactionId>ﾄﾗﾝｻﾞｸｼｮﾝ識別子</TransactionId>
<RequestName>ﾘｸｴｽﾄ種別</RequestName>
<Data>APﾃﾞｰﾀ</Data>
</Transaction>
```

図7 リクエスト形式

サービス種別	ﾄﾗﾝｻﾞｸｼｮﾝ識別子	計算機名
i	1	計算機1
	･･･	･･･
	n	計算機n
j	1	計算機2
	･･･	･･･
	m	計算機m

図8 トランザクション分散管理テーブル

処理順序保証機構は、リクエスト処理テーブル（R_Table）、リクエスト保存キュー（R_Queue）を保持する。R_Table の構成を図 9 に示す。R_Table には、リクエストの処理順序を定義する順序定義部と、処理を定義する処理定義部がある。例えば、図 9 のサービス種別 i については、i(a)→i(b)→i(c)の順序で処理することを意味している。また、図 10 に示す R_Queue は、受信したリクエストを一時的に保存するために用いる。R_Queue の次リクエスト種別は、次に処理するリクエスト種別を示しており、受信したリクエストが次リクエスト種別と同じ場合に、リクエストが処理される。処理順序保証機構は、リクエストが到着すると R_Table と R_Queue を用いて以下の処理を行う。

[Step1] リクエストを解析し、サービス種別、トランザクション識別子、及び、リクエスト種別を取得する。

[Step2] R_Table の順序定義部を参照し、サービス種別とリクエスト種別に該当する項目が存在するかを判定する。

[Step3-A] 存在しない場合、処理順序が規定されていないリクエストのため、処理を実行する。実行後、処理定義部の次サービス名へリクエストを転送する。

[Step3-B] 存在する場合、処理順序が規定されているリクエストのため、ステップ４以降の処理を行う。

[Step4] R_Queue の該当するトランザクション識別子のリクエスト保存部へ、リクエストを保存する。

[Step5] R_Queue の次リクエスト種別を取得し、受信したリクエストの種別と比較する。

[Step6-A] リクエスト種別が異なる場合、処理を終了する。

[Step6-B] リクエスト種別が同じ場合、処理を実行し、処理定義部で示される次サービスへリクエストを送信する。（Step7 へ）

[Step7] 次リクエスト種別を更新する。更新後、R_Queue を参照し、次リクエスト種別に該当するリクエストが存在するかを判定する。

[Step8-A] 次リクエストが存在しない場合、処理を終了する。

ｻｰﾋﾞｽ種別	順序定義部				処理定義部		
					ﾘｸｴｽﾄ種別	ｻｰﾋﾞｽ名	次処理名（Webｻｰﾊﾞ名、ｻｰﾋﾞｽ名）
i	a	b	c	null	a	ｻｰﾋﾞｽa	(D,d)
					b	ｻｰﾋﾞｽb	null
					c	ｻｰﾋﾞｽc	(E,e)
j	b	a	null	null	a	ｻｰﾋﾞｽa	(D,d)
					b	ｻｰﾋﾞｽb	null
k	c	a	b	null	a	ｻｰﾋﾞｽa	(D,d)
					b	ｻｰﾋﾞｽb	null
					c	ｻｰﾋﾞｽc	(E,e)

図9 リクエスト処理テーブル

ｻｰﾋﾞｽ種別	ﾄﾗﾝｻﾞｸｼｮﾝ識別子	次ﾘｸｴｽﾄ種別	ﾘｸｴｽﾄ保存部	
			ﾘｸｴｽﾄ種別	データ保存部
i	1	a	a	null
			b	bﾃﾞｰﾀ
			c	cﾃﾞｰﾀ
	2	c	a	aﾃﾞｰﾀ
			b	bﾃﾞｰﾀ
			c	null

図10 リクエスト保存キュー

[Step8-B] 次リクエストが存在する場合、次リクエストをリクエスト保存キューから取り出し、次リクエストを実行する。Step7 に戻り、次リクエストが存在しなくなるまで繰り返す。

4．2 提案方式の特徴

提案方式では、処理順序保証機構が、R_Table と R_Queue を用いてリクエストの処理順序制御を行う。このため、応用プログラムに変更を加えることなく、処理順序を制御することができる。例えば、図 9 に示すようにサービス種別 i,k を構成するリクエスト種別は両者とも a,b,c であるが、サービス種別 i は a→b→c、サービス種別 k は c→a→b の異なった順序が定義できる。したがって、複数のサービス種別への対応（要求１）を満足できる。また、提案方式は、各 Web サーバにおいて、トランザクションを構成するリクエストの処理順序を追加・変更する場合、該当する Web サーバの R_Table の構成を変更するのみで良い。このため、他の Web サーバへの影響が少ない。したがって、Web サーバの独立性への対応（要求２）を満足できる。

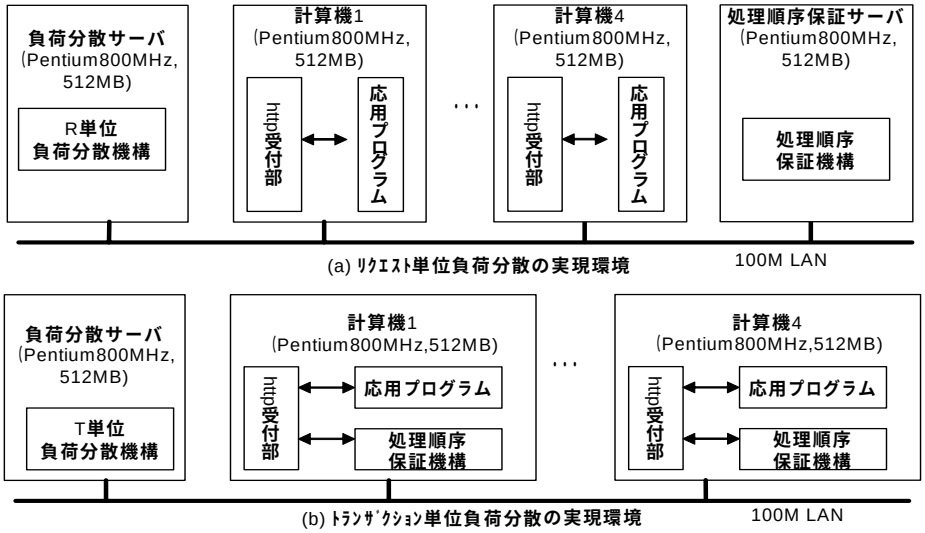


図11 評価環境

5. 実装と評価

5.1 評価条件

負荷分散を伴う処理順序保証方式について、提案方式を実現し評価を行った。また、比較のため、R 単位方式も実現した。実現したハードウェア環境を図 11 に示す。図 11(a)中の処理順序保証サーバは、R 単位方式の時に計算機間の処理順序制御を行うためのものである。負荷分散機構、http 受付部、および応用プログラムは、すべて Java により実現している。また、処理順序保証機構は Java と Oracle データベースを用いて実現した。

評価は、1 種類のサービス種別において、リクエスト数が５個（a,b,c,d,e,f）の場合について、サーバ数、データサイズ、リクエストの到着順序を可変にして実施した。リクエストの処理順序は、a→b→c→d→e→f としている。

5.2 評価結果

計算機数を増加させた場合のスループットを図 12 に示す。リクエストのデータサイズは１KB である。図中の正順は、リクエストを a→b→c→d→e→f で送信した場合であり、逆順は、f→e→d→c→b→a で送信した場合である。図により、以下のことがわかる。

（1）T 単位方式のスループットは、ほぼ計算機数倍で増加している。一方、R 単位方式は、計算機数が 2 倍になっても 25%程度の増加である。これは、計算機間にまたがる順序調整のための同期処理時間が大きいためである。

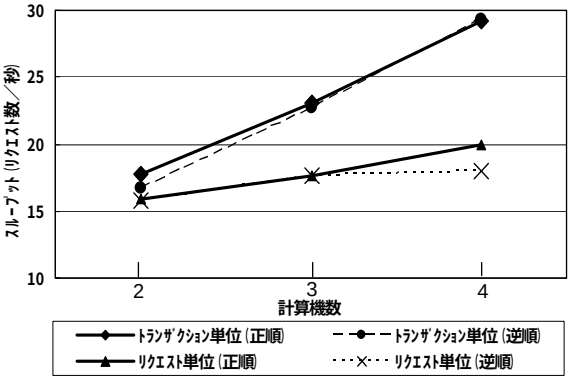


図12 計算機数によるスループット評価

（2）正順と逆順を比較すると、T 単位方式の場合は両者にほとんど差はないことが分かる。一方、R 単位方式の場合は、計算機数 4 の時、逆順のスループットが低下している。これは、逆順の場合、最初に実行されるべきリクエスト種別 a が到着すると、すでに到着している他のリクエスト b,c,d,e,f の処理が継続して実行されるためである。すなわち、処理順序保証機構での同期処理が集中して発生するため、同期に必要な処理時間がさらに増加していると推察できる。

次に、データサイズを可変にした場合の、スループットを図 13 に示す。リクエストの順序は正順である。図 13 より以下のことがわかる。

（1）両方式ともデータサイズ増加に伴い、スループットが低下している。これは、リクエスト保存キューにリクエストを保存する処理時間が増加しているためである。

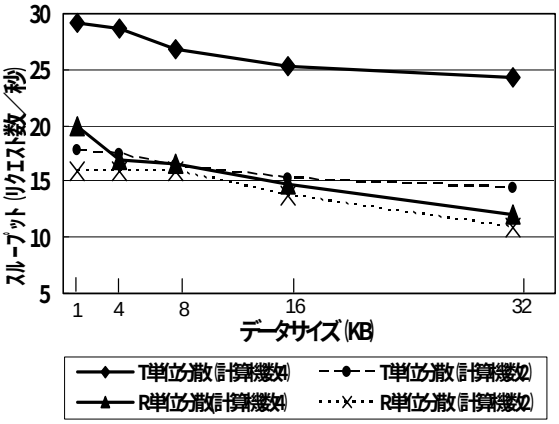


図13 データサイズによるスループット評価

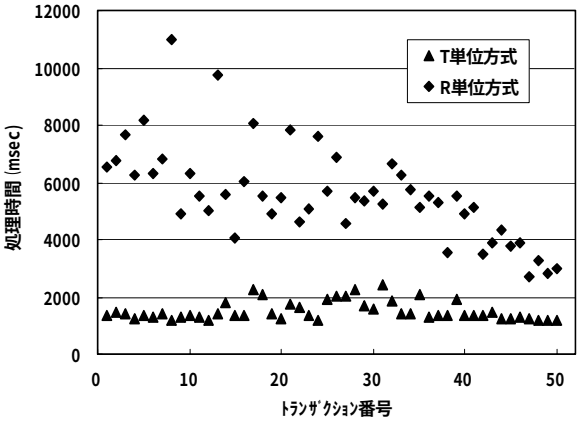


図14 トランザクション処理時間の分布

（2）T 単位方式（計算機数 4）に関して、1KB と 32KB を比較すると、約 17%スループットが低下している。また、R 単位方式（計算機数 4）は、40%程度も低下する。これは、リクエスト保存キューへのリクエスト保存処理時間が大きくなるつれ、処理順序調整のための同期待ち時間が増加するためである。つまり、T 単位方式は計算機毎に処理順序保証機構を配置しているが、R 単位方式では、計算機間で１つの処理順序保証機構を使用するため、低下率が大きくなる。

（3）R 単位方式の低下率が大きいことから、データサイズの増加に伴う負荷分散処理の処理時間は、処理順序保証のための同期処理時間より小さいことが分かる。

最後に、T 単位方式と R 単位方式のトランザクション処理時間の分布を示す。図 14 は、データサイズが１KB、逆順の場合の処理時間の分布を表している。図 14 より、T 単位方式のトランザクション処理時間は、R 単位方式の約 1/5 である。また、R 単位方式の処理時間のばらつきに比べ、T 単位方式の処理時間のばらつきが小さいことが分かる。

以上の評価より、T 単位方式は、計算機数に比例してスループットが向上し、またデータサイズの増加にもスループットの低下が小さい。したがって、大規模かつ高負荷な Web システム構築に適していると言える。

6．まとめ

負荷分散と処理順序保証の構成法として、述べた複数の Web サーバを連携させて処理を行う Web システムの性能向上を可能にする構成法を提案した。提案方式は、トランザクション単位での負荷分散方式により処理順序保証機構を各計算機上に配置する。また、提案方式では、応用プログラムがリクエストの処理順序を意識する必要がない特徴を有する。これにより、複数種類のトランザクションでの利用や、トランザクション内の処理順序変更に容易に対応できる。さらに、提案方式の性能評価を行い、従来のリクエスト単位の負荷分散を用いた場合と比較を行った。評価の結果、提案方式は、計算機数に比例してスループットが向上し、またデータサイズの増加にもスループットの低下が小さい。これにより、提案方式は、大規模かつ高負荷な Web システム構築に適していることを示した。

残された課題として、同期処理時間と負荷分散機構の処理時間を可変にした場合の評価、リクエストの到着時間を可変にした場合の評価がある。

参考文献

[1] 関：” ビジネス IT インフラでのグリッドコンピューティング”, 情報処理, Vol44 No.6, pp.581-587 (2003)

[2] 青山：”Web サービスネットワークとサービスグリッドを統合した Web サービスグリッドネットワーク”, 情報処理学会シンポジウムシリーズ, Vol.2003, No.5 pp.43-44 (2003).

[3] 井上, 山口：”NAT における WWW サーバの負荷分散機構の実装”, 情処研報 DPS-78-004 (1996).

[4] 潤田, 弓場, 佐藤：”種々の並列・分散アプリケーションに対して容易に統合化な動的ロードバランサ pDLB の提案と実装”, 情処研報 DPS-102-26 (2001).

[5] 大平, 松本, 平木：”汎用クラスタ上の資源情報を用いた HTTP サーバにおける負荷分散性能の評価”, 情処研報 OS-85-005 (2000).