サーバ・クライアントの依存関係を用いた分散システム障害管理方式

5U－1

米田　健＊　谷林　陽一＊＊　宮内　直人＊　中川路　哲男＊　勝山 光太郎＊
＊三菱電機情報システム研究所　＊＊三菱電機情報システム製作所

1. はじめに

現在急速に普及し始めたクライアントサーバシステムにおいては、クライアントプロセスがサーバプロセスに対してRPC[1]を用いて処理を依頼するという形態をとる。そして、クラアントプロセスから処理を依頼されたサーバプロセスがさらに異なるサーバプロセスに処理を依頼するケースも大規模分散処理システムでは一般的になると思われる。このような場合、クライアントプロセスの依頼した処理に対しての応答が遅い、クライアントプロセスの依頼した処理が異常終了する、といった事態が生じた場合にその原因を把握しにくい。そこで、クライアントプロセスの要求の結果RPCがどのサーバプロセスに発行され、RPCが伝えられるまでの時間、RPCにより、実行される処理の所要時間、その処理の正否を把握することで、応答が遅い場合のボトルネックの検出、障害時の障害箇所の検出が可能な２つの方法を提案し比較検討を行った。

2.クライアントサーバモデルにおける処理の依頼のネスト

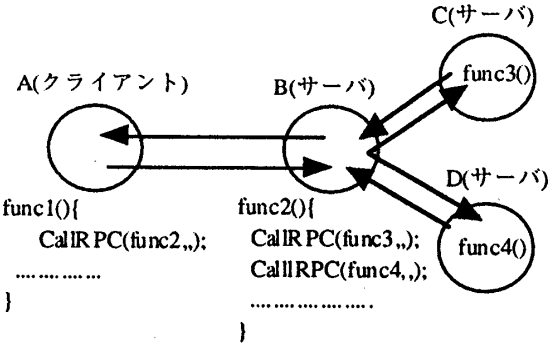

図1ネストされた処理の依頼

図1では、クライアントプロセスAがプロセスBに処理の依頼をし(RPCによるfunc2()の呼び出し)、サーバプロセスBはその処理の中でさらにサーバプロセスC,DにRPCにより処理を依頼している(func2()内でのRPCによるfunc3(),func4()の呼び出し)。今、プロセスA,B,C,Dはすべてネットワーク上の異なるマシン上に存在すると仮定する。Aで実行されたfunc1()の終了までに異常に時間がかかったり、func1()が異常に終了する場合、その原因の所在場所としては、A,B,C,Dの存在するマシン内、それぞれのマシンをつなぐネットワークが考えられる。しかし、原因の存在場所を特定することはきわめて困難である。その理由として以下のような問題点が挙げられる。

[問題点1]サーバの関数はさまざまなプロセスからRPCに呼ばれるが、どのプロセスから呼ばれたかを容易に把握する手段は提供されていない。

[問題点2]クライアントプロセスがサーバプロセスに処理を依頼し、処理を依頼されたサーバプロセスが異なるサーバプロセスに処理を依頼する場合、クライアントプロセスから直接処理を依頼されたサーバプロセス、間接的に処理を依頼されたサーバプロセスがすべてクライアントプロセスの要求した処理にかかわっていることを容易に把握する手段が提供されていない。

[問題点3]処理にかかわったサーバ、クライアントプロセス間でやりとりされるRPCのパケットのネットワーク伝播時間、RPCによってコールされた関数の処理時間を容易に把握する手段が提供されていない。

これらを解決する方法として以下に示す、２つの方法を提案する。

3.　RPC駆動型管理情報収集方式(A方式)

この方式においては、プロセスがRPCのパケットを送受信する度に、クライアントの処理要求を識別するID、RPCの発行元、宛先、時刻などの付加的情報をRPCの利用状況を監視するRPC管理サーバに送り、RPC管理サーバが送られてきた情報をもとに、障害箇所、ボトルネック検出を行う。

(1)RPCパケットフォーマット

GPID	SrcID	DstID	従来のRPCパケットの情報

図2RPCパケットフォーマット

GPID(Global Processing ID)…あるクライアントからの処理の要求により引き起こされるすべてのRPCに対して共通につけられるIDであり、はじめに処理を依頼するマシンIDとそのマシン内で発行されるRPCをユニークに識別するIDを組み合わせたIDである。

SrcID…RPCの要求の発行元を識別するID

DstID…RPCの要求の受付先を識別するID

A distributed system fault management method using relations among client and server processes

Takeshi Yoneda*,Youichi Tanibayashi**,
Naoto Miyauchi*,Tetsuo Nakagawaji*,Koutaro Katsuyama*
*Mitsubishi Electric Co. Info. Sys. Lab.
**Mitsubishi Electoric Co. Info. Sys. Eng. Center

(2)管理情報
RPCのパケットを送信、受信するときに以下のような情報をRPC管理サーバに送信する。

GPID	SrcID	DstID	Direction	Event	Time	OtherInfo

GPID,SrcID,DstIDはRPCのパケットの情報。
Direction…RPCの要求か、応答か。
Event…send, receive, timeout
(rpcのパケットを送信するときか、receiveするときか、TimeOutのときか)
Time…管理情報作成時の時刻
OtherInfo…マシンの任意の情報（マシンの付加情報)

図3 RPC管理サーバに送信される情報

(3)管理情報発行タイミング
管理情報は、RPCパケットを送る時、またはRPCパケットを受け取る時にRPC管理サーバに伝送される。
(4)管理情報分析手法
RPC管理サーバ側では、送られてくる管理情報をGPIDが共通のものごとに収集し、SrcID,DestIDの情報をもとにRPCがどこに発行されたかを把握する。
そして、時刻の情報をもとに、ネットワーク伝播時間、プロセス内処理時間を算出する。また、ネットワークの障害、RPCの受信プロセス、マシンの障害はEventフィールドにTimeOutが記されていることで把握できる。

4. 管理情報累積付加方式(B方式)

管理情報累積付加方式では、RPCを送信、受信するたびにそのRPCを扱ったエンティティのID(マシンID+プロセスID+ThreadID)、時刻などの情報を次々と付け加えていく。そして、クライアントにRPCの応答が戻ってきたとき、そのRPCの応答に付加された情報を検証することによりクライアントの要求した処理が、どのようなマシン上のどのプロセス、どのスレッドにより扱われ、ネットワークの伝播時間、マシン内での処理時間はどのくらいであったのかを把握することができる。
(1)RPCパケットの送信時、受信時に付加される情報

EntityID	Direction	Event	Time	OtherInfo

EntityID…RPCを送信、受信するエンティティのIDでマシンID+プロセスID+スレッドIDである。
Direction…RPCの要求か、応答か。
Event…send, receive, timeout
(rpcのパケットをsendするときか、receiveするときか、TimeOutのときか)
Time…管理情報作成時の時刻
OtherInfo…マシンの任意の情報（マシンの付加情報)

図4 RPCパケット送受信時に付加される情報

(2)付加される累積情報の具体例
図1のように4つのプロセスA,B,C,Dの間でRPCのやりとりが行われる場合、管理情報が付加されていく様子を図5に示す。図5においてはRPCの中身のデータは省略してある。

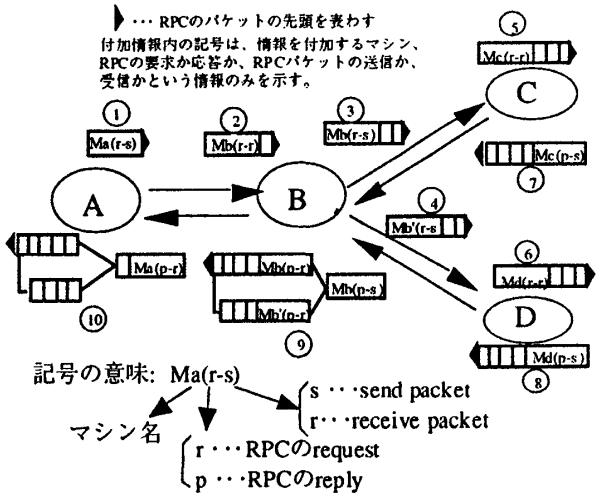

図5管理情報埋累積付加方式

5. 考察

A,B２つの方式が2章で述べた問題点に関してどのように対処しどのような課題が存在するかを述べる。

問題点1に対しては、A方式ではRPCのパケットにSrcIDをつけることにより、B方式ではRPCの発行元IDをEntityIDとして付加することで対処している。　問題点2に対しては、A方式では、クライアントの要求した処理にかかわるRPCには共通のユニークなIDをつけることにより、B方式ではクライアントの要求した処理にかかわったEntityのIDが累積的に付加されていくことにより対処している。

問題点3に対しては、A,B方式ともタイムスタンプをRPCのパケットに付加することで、対処している。

A方式の課題としては、RPC管理サーバに集まってくる管理情報の量を適切に制限する方法の考案がある。RPC管理サーバに管理情報を送るべきRPCとそうでないRPCを区別する必要がある。またB方式の問題としては処理のネストのレベルが深くなればなるほど、RPCのパケットに付加される情報が大きくなりその情報の構造も複雑になることがあがられる。そこで、付加される管理情報をできるだけ少なくする仕組が必要である。

また両方式とも分散環境における時刻の同期がとれていることが必要である。

6. おわりに

大規模分散環境における性能管理、障害管理のツールとしてRPCを利用する方式を提案した。今後は実装する場合のことを考慮し、上記課題に取り組んでいく予定である。

参考文献

[1]A.D.Birrell and B.J.Nelson,Implementing Remote Procedure Calls,ACM Trans. on Comp. Sys.pp.39-59, Feb. 1984.