C_013

サーバ主導型リクエスト多重度決定アルゴリズムの提案とiSCSIへの適用
Proposal of Server-Driven Multiple Request Control Algorithm and the Application for iSCSI

中塚 大樹† 白木 伸二郎‡ 岩満 幸治†
Daiki Nakatsuka Shinjiro Shiraki Koji Iwamitsu

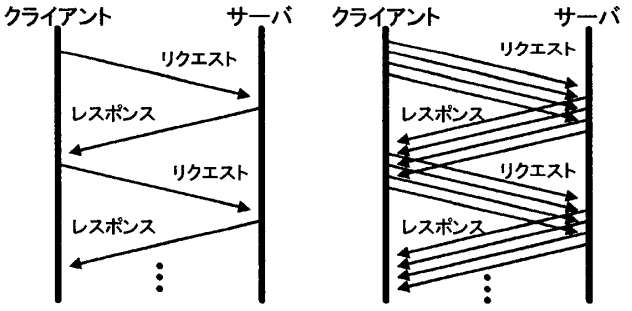
1. はじめに

ある特定の処理を実行する少数のサーバと，それを利用する多数のクライアントで構成されるクライアント・サーバシステムは，近年様々なアプリケーションで使用されている．本研究では，クライアントとサーバとの間の通信を効率化する，リクエスト多重度の決定アルゴリズムを提案し，その効果を評価した．

2. 背景

クライアント・サーバシステムでは，クライアントが，サーバへ処理の依頼を送信し(リクエストの送信)，それを受け取ったサーバは，リクエストの内容に基づき処理を行なった後，その結果を返信する（レスポンスの送信).

クライアントとサーバとの間で使用される通信プロトコルの多くは，クライアントによるリクエストの多重化を可能としている．ここで，リクエストの多重化とは，クライアントが，レスポンスの到着を待つことなく，複数のリクエストを連続して送信することを指す．また，この時に連続して送信するリクエスト数を多重度という．リクエストを多重化した時の通信シーケンスを図1に示す.



(a) 多重度1の通信シーケンス (b) 多重度4の通信シーケンス

図1 リクエスト多重化時の通信シーケンス

図1に示す通り，リクエストの多重化によってクライアントおよびサーバの処理待ち時間が減少するため，通信を効率化できる．しかし，サーバの能力(例えば受信したリクエストを格納するためのバッファ領域など)は有限であり，リクエスト数がサーバの能力を上回ってしまうと，サーバはリクエストを廃棄してしまう．この場合，クライアントは，一定時間レスポンスの到着を待った後リクエストを再送しなくてはならない．そのため，サーバによるリクエストの廃棄は，単位時間あたりの処理リクエスト数を低下させる．従って，クライアント・サーバシステムでは，サーバの処理能力を超えないように各クライアントのリクエスト多重度を決定するアルゴリズムが重要である.

† (株) 日立製作所
‡ (株) 日立コンピュータ機器

3. 課題と目的

リクエストの多重度を決定するアルゴリズムは，クライアントが多重度を決定するものと，サーバが多重度を決定するものとの2つに分類できる．本稿では，前者をクライアント主導型アルゴリズム，後者をサーバ主導型アルゴリズムと呼ぶ.

クライアント主導型アルゴリズムの例として，TCP[1]で採用されているNewReno[2]やVegas[3]などがある．TCPでは，クライアントが，これらのアルゴリズムによって送信するセグメントの多重度を決定する.

また，サーバ主導型アルゴリズムを採用する通信プロトコルの例として，iSCSI[4]がある．単純なサーバ主導型アルゴリズムとして，サーバが，レスポンスを送信する時点での残資源量を基に多重度を決定する，というアルゴリズム(単純アルゴリズム)が考えられる.

しかし，単純アルゴリズムは，サーバ資源に余裕がある時にリクエストを送信したクライアントへ大きな多重度を割り当てる一方，サーバ資源が少ない時にリクエストを送信したクライアントへ小さな多重度を割り当ててしまう(不平等問題)．また，不平等問題を避けるため，すべてのクライアントへ均等に多重度を割り当てると，リクエスト送信頻度が小さいクライアントへ，リクエスト送信頻度が高いクライアントと同じ多重度を割り当てることになり，効率が低下する(多重度浪費問題).

サーバ主導型アルゴリズムの設計には，上記の2つの問題を解決しなくてはならないという課題がある．そこで，本研究では，上記の2つの問題を解決するサーバ主導型アルゴリズムを提案し，その評価を行なった.

4. 定義

本稿では，アルゴリズムの適用対象とするクライアント・サーバシステムを以下のように定義する.

サーバに接続するクライアント数をiとし，各クライアントをCLn (n=1,2,…,i)と呼ぶ．また，サーバが有する資源の量をNとする．サーバの資源は，リクエスト受信時に一つ減少し，レスポンス送信時に一つ増加する．つまり，Nはサーバが同時に処理できるリクエスト数に等しい.

サーバは，残資源数が0のときにリクエストを受信すると，そのリクエストを破棄する．また，クライアントは，リクエスト送信から一定時間待ってもレスポンスを受信できない場合，リクエストを再送するものとする.

さらに，タイムアウト時間をTとする．クライアントが，リクエストを送信してからT秒以内にレスポンスを受信できなかった場合，もしくは，多重度が0である時間がT秒を超えた(つまり，T秒間リクエストを送信できなかった)場合，クライアントはエラーが発生したと判断し，サーバとの通信を切断する(タイムアウトエラー).

5. 提案アルゴリズム

本研究で提案するサーバ主導型アルゴリズムを，図 2に示す．また，提案アルゴリズムが使用するパラメータを表1にまとめる．以下，提案アルゴリズムを説明する．

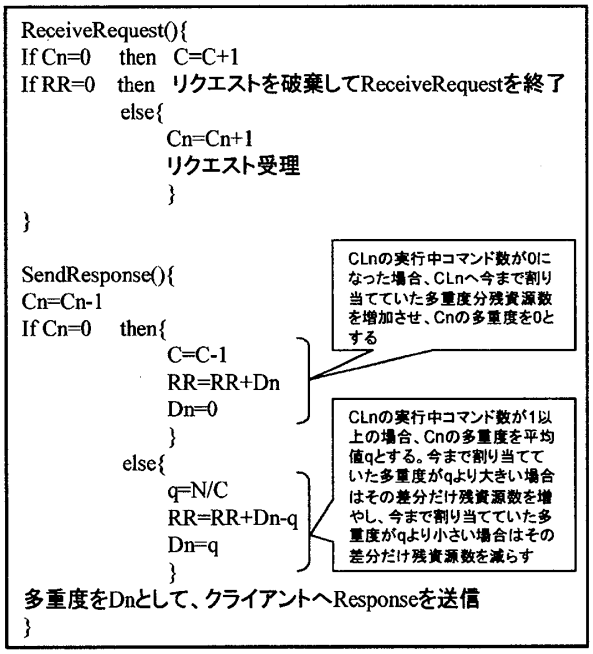


図 2 提案アルゴリズム

表 1 提案アルゴリズムのパラメーター覧

#	パラメータ名	意味	初期値
1	C	リクエストを送信しているクライアント数	0
2	Cn(n=1,2,…,i)	CLn の実行中リクエスト数	0
3	RR	サーバの残資源数	N
4	q	N をリクエスト送信中のクライアント数で等分した値	0
5	Dn(n=1,2,…,i)	CLn へ割りあてる多重度	0

提案アルゴリズムは，図 2の通り 2 つの関数からなり，リクエスト受信時に ReceiveRequest 関数を実行し，また，レスポンス送信時に SendResponse 関数を実行する．SendResponse 関数は，サーバ主導型アルゴリズムの 2 つの問題を解決するため，以下の動作を行なう．

まず，多重度浪費問題への対策として，サーバは，CLn へレスポンスを送信することで CLn が実行中のリクエスト数が 0 になった場合，CLn の多重度を 0 とし，減らした分 RR(残資源数) を増加させる．つまり，サーバは，実行中のリクエスト数が 0 になったクライアントはしばらくリクエストを送信しないものとみなし，そのクライアントへ割り当てた資源を回収する．

また，不平等問題への対策として，サーバは，CLn の実行中リクエスト数が 1 以上の場合，CLn の多重度を q とする．ここで，q は，資源量をリクエスト送信中のクライアント数で等分した値である．これにより，サーバは各クライアントの多重度を均等にすることができる．

6. 評価

本章では，提案アルゴリズムと単純アルゴリズムを実装し，それらの性能を比較した結果を記す．なお，クライアント・サーバ間の通信プロトコルは iSCSI を利用した．

6.1. 評価環境

提案アルゴリズムと単純アルゴリズムを実装した iSCSI サーバへ，100 台の iSCSI クライアントを接続した(i=100)．サーバの資源量は 512 とした(N=512)．また，タイムアウト時間は 10 秒とした(T=60)．

6.2. 評価指標

多重度浪費問題が解決すると，サーバの性能を有効利用でき，サーバが 1 秒間当たりに処理できるリクエスト数(IOPS)が増加する，と考えられる．また，不平等問題が解決すると，クライアント毎のリクエスト送信数が均等になり，T 秒間以上多重度が 0 のままとなるクライアント数が減少する，と考えられる．そこで，IOPS およびタイムアウトエラー発生クライアント数を測定した．

6.3. 評価結果

IOPSの平均値およびタイムアウト発生クライアント数を表 2に示す．なお，IOPSの平均値は，単純アルゴリズムの測定値を 1 とした比率で表す．

表 2 提案アルゴリズムと単純アルゴリズムの比較

#	比較項目	単純アルゴリズム	提案アルゴリズム
1	IOPS 平均値(相対値)	1.00	1.29
2	エラー発生数	32	0

表 2に示した通り，単純アルゴリズムに比べ，提案アルゴリズムはIOPSが約 30%向上している．また，単純アルゴリズムでは，不平等問題の発生により，32 台のクライアントでタイムアウトエラーが発生し，クライアントとサーバとの間の接続が切断された．一方，提案アルゴリズムではタイムアウトエラーが発生しなかった．

7. おわりに

本研究では，サーバ主導型の多重度決定アルゴリズムを提案し，その評価を行なった．評価の結果，提案アルゴリズムを採用することで，単純アルゴリズムを採用する場合に比べて IOPS が約 30%向上し，タイムアウトエラーの発生を防止できることを確認した．

参考文献

[1] J. Postel et al., "Transmission Control Protocol (RFC793)", IETF (1981.9).

[2] S. Floyd et al., "The NewReno Modification to TCP's Fast Recovery Algorithm(RFC2582)", IETF (1999.4).

[3] B. Lawrence et al., "TCP Vegas: New Techniques for Congestion Detection and Avoidance", ACM SIGCOMM, pages 24-35 (1994.8).

[4] J. Satran et al., "Internet Small Computer Systems Interface (RFC3720)", IETF (2004.4).