2R-5

Webサーバの負荷情報に基づいた動的パラメータチューニング

南野　久美子†　　佐藤　喬†　　多田　好克†
電気通信大学　大学院情報システム学研究科†

1　概要と目的

近年、インターネットの中心的サービスであるWWW (World Wide Web)は爆発的に発展してきている。それにともない、インターネットの利用者数の増加、Web利用によるビジネスチャンスの増加など、Webサーバが安定したサービス供給をおこなう必要性が高まっている。こうしたことからWebサーバの管理者が、Webサーバの性能測定や、パフォーマンスをあげるためのパラメータチューニングを重要視する傾向がある[1][2]。

Webサーバにおいて、より少ない資源環境でより高速なレスポンスを確保するために、Webサーバの管理者によるサーバのパラメータチューニングは欠かせない。しかし、パフォーマンスをあげるためのチューニング作業には、次のにあげるような行程を、管理者が繰り返しおこなう必要がある。

1）Webサーバへの負荷状況の取得
2）負荷の分析
3）Webサーバのパラメータの変更
4）パラメータの妥当性の検証

これらの作業は、多くの時間を費やし、管理のコストが増える要因でもある。また、Webサーバへの負荷は常に変化するものであり、高いパフォーマンスを維持するためには1度のチューニング作業で対応できるものではない。パラメータの誤った設定が原因で、サーバの障害発生やパフォーマンスの低下がおこるとも考えられる。

そこで、稼動中のWebサーバの負荷状況に応じて、自動的にパラメータチューニングを行うシステムを提案する。本システムを用いることによって、サーバ管理者の作業負担を最小限に抑えたパフォーマンスチューニングを可能にすることが目的である。

2　システム全体の構成

本システムの対象サーバソフトウェアはApache[3]とする。Apacheは機能ごとに分割されたモジュールで構成される。Webサーバとして必要最低限な基本的な機能をhttp_coreモジュールとしてまとめ、その他の機能は取り外し可能なモジュールとして実装している。本チューニング機構は図1のように、Apacheモジュールとして実装する。モジュールとして実装することにより、Apache内部に保持している情報を利用することができ、オーバヘッドもほとんど生じない。また、既存のシステム(OSやApache)に変更を加えることなく、本システムを利用することができる。

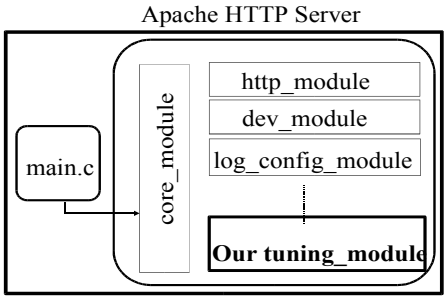

図1　システムの構成

3　対象パラメータ

本システムでは、チューニング対象のパラメータをKeepAliveTimeoutとMaxClientsの2つとした。

・KeepAliveTimeout

HTTP[4]では、サーバとクライアント間で持続的なコネクションが可能である。この持続的なコネクションのことをKeep-Alive接続と言う。KeepAliveTimeoutは、Keep-Alive接続中に次のリクエストがくるまでサーバが接続を持続しつづける時間を決めるパラメータである。

・MaxClients

MaxClientsは、応答することのできる同時リクエスト数を決めるパラメータである。プロセスベースのサーバにおいてMaxClientsは、クライアントからのリクエストに応答するために起動される子プロセスの最大数を決定する。

Apacheに対してベンチマークを用いて実験を行い、KeepAliveTimeoutとMaxClientsがサーバの性能に及ぼす影響を確認した。実験のサーバ計算機には、Linux2.4、Apache2.0.55を用いた。搭載メモリ量は256MBである。クライアント計算機はサーバ計算機と同スペックのものを用意した。ベンチマークには、JMeter[5]を用いた。同時接続数を500と設定したベンチマークテストをそれぞれ5回行い、平均値をくらべたものが図2になる。

“Dynamic parameter tuning based on load factors of the Web server”

†　Kumiko Minamino, Takashi Satou, Yoshikatsu Tada.
Graduate School of Information Systems ,
The University of Electro-Communications.

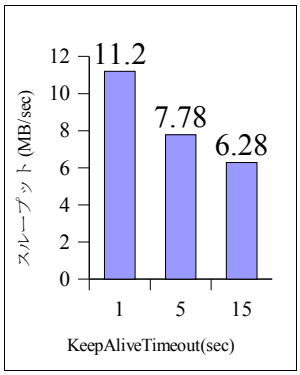

図2　KeepAliveTimeout と性能の関

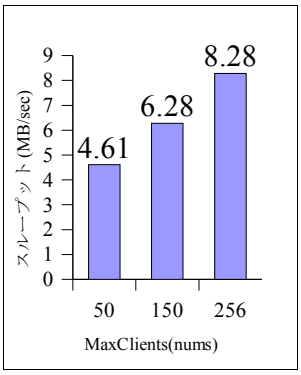

図3　MaxClients と性能の係関係

デフォルトは、KeepAliveTimeout=15、MaxClients=150 である。計測結果から、2 つのパラメータ値がサーバの性能に大きく影響を与えていることが分かる。

図 2 では、無駄にコネクションを維持している時間が長くなることによりスループットはさがっている。図 3 では、個プロセス数が増えることにより、処理待ちのリクエストが減り、結果スループットがあがっている。

4　負荷状況の取得・分析

・KeepAliveTimeout

Keep-Alive 接続中では、サーバがリクエストを処理し終わり、しばらくの間リクエストがこない時間(これを思考時間と呼ぶ)が長ければ長いほど、資源は無駄に使われていると考える。これはサーバのパフォーマンス低下の一因になる。このことから、サーバの負荷状況として、クライアントからのリクエスト間隔を計測する。リクエスト間隔が長くなったところを思考時間に入ったと判断し、Keep-Alive 接続が切断できるよう KeepAlive-Timeout を自動調整していく(図 4)。

本システムは、Apache がリクエストを accept した後のフェーズにフックし、コネクションごとの全リクエスト間隔を記録し続ける。リクエスト間隔の分布から、思考時間に入ったと仮定する思考開始時間を定め、Keep-AliveTimeout に再設定する。新たな KeepAliveTimeout は、再設定後に確立されたコネクションから有効となる。

・MaxClients

Apache はリクエストの状況によって、子プロセスを生成するが、子プロセス数が多すぎると物理メモリが不足してしまい、スワップアウトが大量に発生してしまう。反対に少なすぎると、サーバの処理能力が下がってしまう。そこで、MaxClients の最適値を

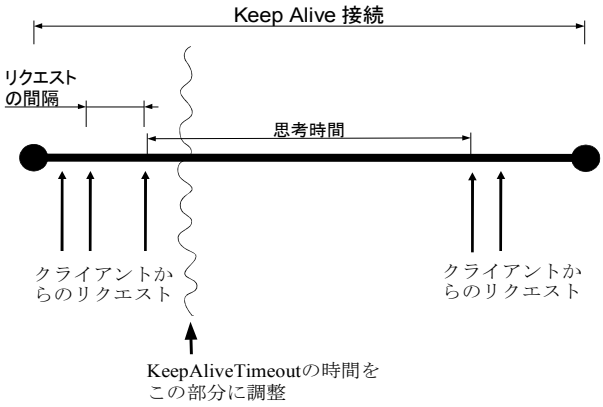

図4　KeepAlive 接続の様子

Apache が使用可能なメモリ量≒
子プロセスのメモリ消費量 × MaxClients + 親プロセスのメモリ消費量

とする。本システムでは、上記の式を満たす MaxClients をもとめるために、稼動中の Apache 内部の各子プロセスの消費メモリを proc ファイルシステムを用いて計測する。その情報をもとに、適切な MaxClients 値を算出し適用する。

5　おわりに

今回、Web サーバにおける負荷情報に基づいた自動パラメータチューニングシステムの設計方針について述べた。今後、実装したシステムの評価をふくめ、2 つのチューニングパラメータの間の影響を加味したチューニング方式等を考慮し、本システムの効果と問題点について示していきたい。

参考文献

[1] Yixin Diao, Neha Gandhi, Joseph L. Hellerstein, Sujay Parekh, and Dawn M. Tilbury: Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics With Application to the ApacheWeb Server, 7th *IEEE/IFIP Symp. on Integrated Network Management* (2001)

[2] Yiming Hu, Ashwini Nanda, and Qing Yang: Measurement, Analysis and Performance Improvement of the Apache Web Server, *18th IEEE International Performance, Computing and Communications Conference* (1999)

[3] Apache: The Apache Software Foundation: http://www.apache.org/

[4] Berners-Lee, T., Fielding, R., and H. Frystyk, Hypertext Transfer Protocol HTTP/1.0., *RFC 1945 MIT/LCS, UC Irvine* (1996)

[5] Apache Jakarta Project: Apache Jmeter: http://jakarta.apache.org/jmeter/