2004－EVA－9　(8)
2004／6／11

しきい値を設定しない相対的なキャパシティ管理の有用性

株式会社アイ・アイ・エム　技術部　塩川　孝雄

分散システムのサーバ群においては、ネットワークの拡大やシステムのブラックボックス化が進み、開発段階からの負荷予測が困難になってきている。これには、しきい値によるキャパシティ管理手法に加え、稼働実績値を基にしたしきい値の無い相対的な管理手法を併用することが効果的と考える。本稿では開発と運用部門が管理ルールを共有すべき必要性とその管理例を示し、これに加えてＣＰＵ使用率による相対的な管理手法を提示して、その有用性を示す。

The effectiveness of the relative capacity management without threshold criteria

Takao Shiokawa, Engineering Deopartment,IIM Corporation

Because of the escalation in both network scale and degree of the systems being a Black Box, it's becoming harder to forecast the load on the server group in distributed systems at the development phase. To break this difficulty, it is quite effective to use the relative management technique based on the actual performance values at running phase, in addition with the traditional management technique setting the threshold from the development phase. I show the merit of sharing management rule between developing teams and operating teams, and give a sample case. Moreover, I present a relative management technique using CPU utilization to show its effectiveness.

1. はじめに

ホスト機のＣＰＵやディスクの平均使用率は、一般的に７０～８０％と一定の高水準を維持しているのに対し、分散システムのサーバ群では平均３０～５０％と概して低いのに関わらず極端な低負荷から高負荷までのバラツキが大きくなっている。

これは、システムのブラックボックス化や複合化により負荷予測が困難化しているのに加え、ハードの低価格や台数の多さによる全数管理の困難さを理由としたキャパシティ管理を軽視しがちな傾向が反映されているものと思われる。

このようななかで、負荷異常に早期に気付くにはどのような事前・事後の方策が考えられるか。

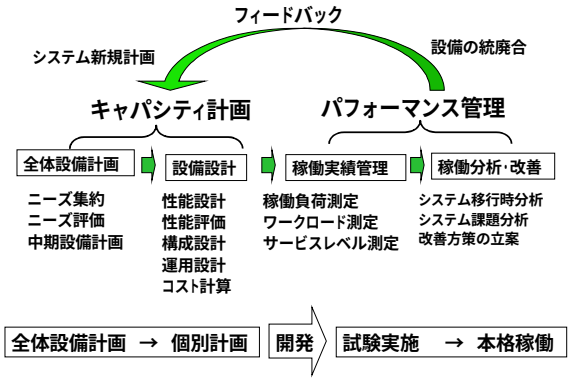
前半で開発・運用間の部門を越えた共通ルールとしきい値によるキャパシティテ管理の事例を述べ、後半でしきい値を設定しない相対的な管理手法を提案し、ＣＰＵ使用率の負荷モデルにより検証して有用性を示す。

2. 開発・運用部門共通のキャパシティ管理ルール

2.1 キャパシティ管理サイクル

ＩＴの急速な進歩によりシステム内部構造の変化が激しいため、処理増等による単純な負荷予測が出来なくなって来ている。しかし、システムの安全性を継続して維持するには、システムの計画・開発段階での「キャパシティ計画」と本格稼動後の「パフォーマンス管理」の全体をキャパシティ管理のフィードバックのサイクルとして考え、リソースの適正規模を維持することに心掛けるべきである。

キャパシティ計画は、各企業が予算作成前に行なう中長期の「全体設備計画」と当年度のシステム開発時における「個別の設備設計」に分けられる。また、運用段階のパフォーマンス管理は、日常運用での「稼働実績管理」と「分析・改善」に分けられる。



2.2 開発段階での設備設計

全面的な先端技術や新製品での構成に対し、実績の多

い技術や製品により構成する場合には省略することもあり得るが、システム開発における設備設計は一般的に次の４ステップにより行われる。

① 計画時:机上検討が中心

システム化の方針や開発要件の仮設定、設備形成方針などを基に、求められる性能や信頼性などを仮定して、設備構成・規模を想定して設計する。この際、過去のシステムやベンチマークのデータを参考にする。

② 設計時:机上＋ベンチマークテスト

対象システムのＳＬＡや要件仕様により、レスポンス時間、ＣＰＵ処理単価、処理件数、必要なリソース量（ＣＰＵ、メモリー、ディスク等）の設計基準値を決め、費用対効果を算出してシステムの規模・構成を設計する。

③ 開発時:実プログラムによる性能評価

予想されるワークロードのピークを考慮した性能試験とストレス試験を行なう。また、レスポンス時間とリソース使用量の調査、およびシステムの処理能力の限界を確認し、ミドルプログラムやアプリケーションプログラム（ＡＰ）をチューニングする。

④ 試験期間:運用開始一定期間での実負荷分析

ＣＰＵ、メモリー、ディスク等のリソース使用率とワークロード件数との傾向分析、およびユーザ・コマンド別のＣＰＵ使用率のシステム特性分析を行なう。またＳＬＡの維持について将来予測を行なう。

2.3 運用段階での稼働分析･改善

システム改訂や予測できない負荷変動等により、開発段階だけでなく運用段階においても性能試験やチューニングおよび設備規模の見直しや導入が繰り返される。

開発と運用段階での分析項目は基本的に同じであるが、稼働実績の反映や回線負荷分析等の要素を加える。

一方、ネッワークやD／B技術、ＧＵＩ技術の進展により大規模で複合的なシステム開発が可能になるにつれ、かつてはシステムの川下にいた現場の運用ＳＥは、分散システムでの運用で必要な技能が蓄積され、キャパシティ管理での役割も大きくなっている。

今後も、ＳＬＡの維持や課題解決において開発ＳＥや利用部門と連携して能動的に機能する運用ＳＥが望まれる。

2.4 可視化の有用性

昨今は稼働情報をグラフィックで表現し、Ｗｅｂ方式等により情報を共有することが容易である。

可視化により、開発時点でのシステム内部の動きや全サーバの負荷状態および利用部門でのレスポンス時間等を迅速に把握でき、異常負荷やリソースの余り具合に気付き易くなる。

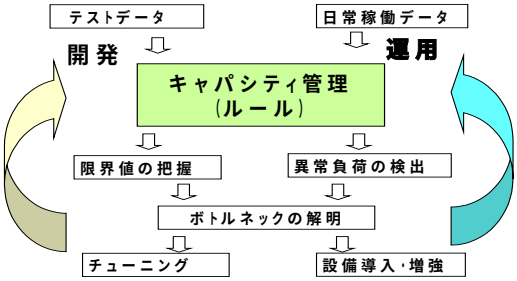
また、関係者間での可視情報の共有は、他システム間のリソースの融通を容易化し、企業全体の長期的なコストを抑制する効果が大きい。

しかし、キャパシティ管理の各段階において、各企業が実状に合わせて稼働情報をどのように可視化するかは今後の課題といえる。

2.5 キャパシティ管理ルールの共通化

キャパシティの管理サイクルを開発と運用の側面でみると相似性があることが分る。

これは、開発と運用部門のボーダレスなナレッジマネジメントにより、ＡＰによるデータ採取やツール類の採用およびデータ蓄積、可視化などを共通の管理ルールとすることで、管理サイクル全体として効率的なキャパシティ管理が可能になることを意味する。



2.6 開発･運用部門による共通改善項目

多くの企業において、従来からのホスト機の管理ルールが分散システムのキャパシティ管理に継承されているのかと危惧される。

このため、ホスト機では目新しいことではないが、前項までを踏まえて、あえて今後の分散システムでのキャパシティ管理で主要と思われる改善点を挙げてみる。

① データ採取

・市販の管理ツールで採取できない必要な管理項目は、開発段階でＡＰ等へ組み込み、その採取機能は運用段階へ引き継ぐ

（例：処理件数、レスポンス時間、エラー件数等）

② 可視化

・開発段階でのシステム内部の動きを可視化し、機能を運用段階へ引き継ぐ

・運用段階ではＷｅｂ方式などにより業務部門別、管理個所別に自動開示する

③ 蓄積

・稼働データは少なくとも１年分を蓄積する

④ 管理ツール

・可能な限り開発と運用段階で同じ管理ツールを使用する（データ発生、採取、分析、可視化）

⑤ しきい管理値

・開発運用部門の共同により、日常管理可能なしきい値を設定する（手順例を３項にて後述する）

・処理件数予測と処理単位から限界時期を明確にする

3. しきい値の設定による管理手法

しきい値は、キャパシティ管理ツールなどの警報機能やグラフ表示により、実際の日常運用で適用できなければ意味をなさない。このため、しきい値の設定は開発と運用部門間で協調して行なう必要がある。

次に、開発・運用部門の共同による絶対的な管理例として、ＣＰＵ使用率のしきい値設定での７ステップの具体的な手順例を示す。(試験期間または運用段階を想定)

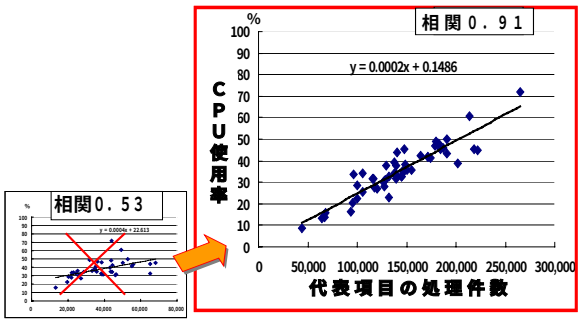
① 異常負荷の摘出･排除

システムループ等の異常負荷分を除外する。

② CPU使用率に相関する処理項目の見極め

ＣＰＵ使用率と出来るだけ相関の高い数量的に予測できるものを代表処理項目として選択する。(トランザクション件数、お客様件数など)

代表処理項目の１件あたりのＣＰＵ使用量(使用率×時間：ｍｓ／件)を計算し、⑥の負荷予測に使用する。



③ CPU使用率のロードカーブ

年間の変化、月内の変化、標準的な１日の負荷推移などのピーク負荷を予測する。

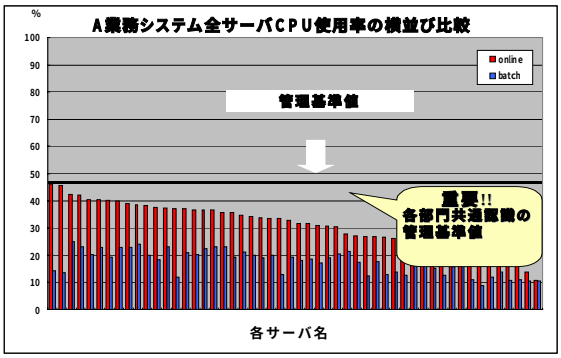
④ 単位時間平均の上限のCPU使用率

ＳＬＡを満足するレスポンス時間内となるよう、ＣＰＵ使用率の上限値を実測により見極める。

ベンチマークテストの必要があれば実施する。

⑤ 日常運用での管理基準値

③と④の単位時間あたりの上限ＣＰＵ使用率とロードカーブを考慮して、日常運用で実際に管理可能な管理基準値を決める。(処理時間帯別での日・月の平均値等)



ＣＰＵ使用率の高い順に横並び棒グラフで表現し、関連部門へ開示することにより、この管理基準値を超える場合に共同で対策がとれる。

⑥ 将来のCPU使用率を予測

代表処理項目の増加件数と②で計算した１件あたりのＣＰＵ使用量からＣＰＵ使用率の変化を予測し、いつ頃にこの基準値を越えるか判別する。

⑦ 設備構成、システム効率改善の規模

低負荷サーバから高負荷サーバへのＣＰＵ基板の融通やシステム自身の処理効率の改善によりコスト増の抑制を図る。必要があれば設備を増強する。

そしてまた①の手順へ戻り、ＰＤＣＡを廻す。

4. しきい値を設定しない相対的な管理手法

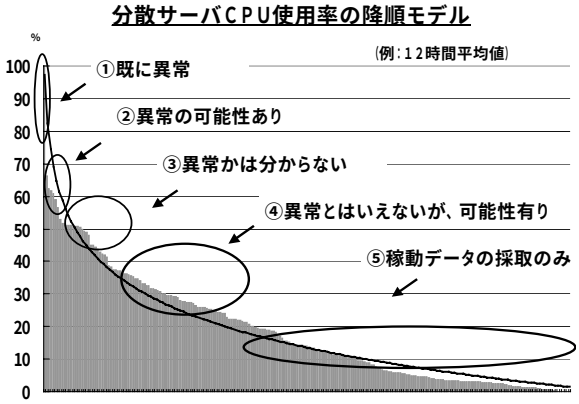
しきい値が決められていない場合においても、システム管理者は、何らかの方法で早期に問題サーバに気付き、対応することが求められる。

大規模なサーバ群の実運用では、メモリー、ディスク等の膨大な管理項目を常時把握管理することは困難であるが、稼働状況を代表する最低限の管理項目として少なくとも「ＣＰＵ使用率」だけは常時採取すべきである。この際、セキュリティ対策などを理由に採取しないサーバを特例として許さないことが必要である。

以下に、複数・多種の大規模サーバ群を集中管理する場合を想定し、「全サーバの階層的表示例」、「問題サーバの洗い出し」、および「絞り込みによる優先順位付け」の順に、稼働実績に基づくしきい値レスでの相対的な管理手法について述べる。

4.1 階層的表示による管理

日常運用でのシステム管理の感覚によりＣＰＵ使用率別にサーバ群を分けると、５ランクに分類できる。



システム管理者が異常負荷の発生に気付き易やすいよう、負荷の大きさの他、システムの重要度合いや安定度合いなどにより、階層的に仕分けして可視化する。次にその表示例を示す。

① 高負荷順グラフ（全サーバ）

全サーバを串刺しで管理する方法として、全サーバのＣＰＵ使用率を高負荷順にソートし棒グラフで表示する。（例えば３００台中２０～５０台程度）

１日に数回更新することにより、システム管理者は高負荷の発生を確認できる。

② CPU使用率推移グラフ（30%程度のサーバ）

新規導入サーバや負荷が高いサーバを選別してＣＰＵ負荷率の推移グラフをリアルタイムに表示する。

③ 重点管理対象サーバ（10%程度のサーバ）

特に重点的に管理するサーバは、ＣＰＵ使用率、処理件数、レスポンス時間の推移やタイムアウトエラー等の推移グラフを自動作成して表示する。

4.2 問題サーバの洗い出し

ＣＰＵ使用率が低い場合においては、予定外の処理増やプログラムループによる負荷増、メモリー不足などの異常負荷サーバに気付かない場合も多く見受けられる。

このため、「問題のサーバを自動的に洗い出す管理手法」について、筆者の運用管理の実務経験をもとに、ＣＰＵ使用率を活用して行なう手順を考え、モデルデータにより検証した。

（ＣＰＵ使用率１時間間隔・１２時間平均のデータを想定して説明）

① 各種統計値の使用

しきい値が設定されていない場合、システム運用管理者が「異常負荷に気付く」には、過去の実績か他のサーバ負荷との比較に頼ることになる。

例えば、ＣＰＵ使用率の急激な変動、一定期間での大幅な増加や低減、他サーバとの比較での異常な高低および負荷の変化方向の異なり等である。

これ等を模擬して、次の様な統計値の組み合わせにより、異常負荷かどうかを自動的に判定を行なう。

- 平均値、最高値、最小値（負荷の大きさを表現）
- 連続負荷（複数インターバルにより負荷の連続性を表現）
- 変動係数（負荷の変化度合いを表現）
- 相関係数（負荷の相似度合いを表現）

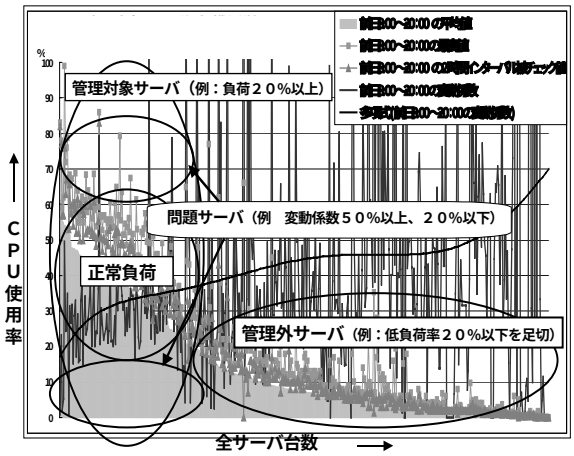
② 問題サーバの負荷の位置付け

システム管理者が経験的に感じている問題サーバの位置付けを、上記の統計値により設定する。

平均使用率が２０%以下のように極端に低いサーバは、統計値が不安定となり問題サーバとして洗い出される可能性があるので対象外とする。これによる運用上の不都合は少ないと思われる。

ここでは、変動係数を例に問題サーバの位置付けについて説明する。変動動係数（標準偏差値を平均で割り１００倍して百分率で使用）は、平均値の大きさに関係なくバラツキの大きさを比較するのに都合が良い。

ランダムなトランザクションのオンライン負荷の変動係数は２０～５０%の範囲内になることが多い。極端に負荷が高い場合、またはループ負荷のように変化しない負荷がある場合はこの範囲外になる傾向がある。このような数値を設定して問題サーバの洗い出で使用する。



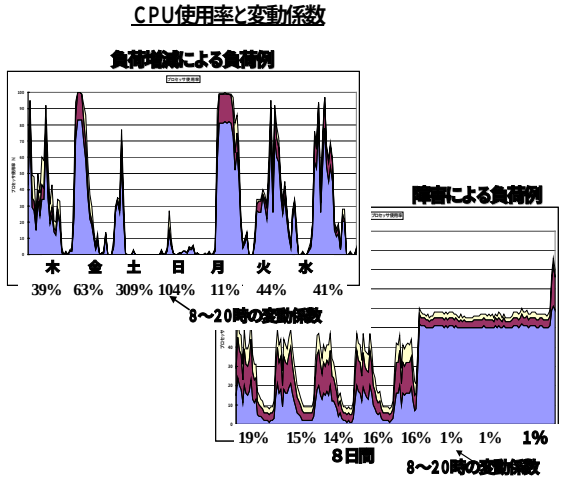
③ 標準負荷の仮定

システムの運用管理者がＣＰＵの負荷パターンを記憶して日常の運用判断することを模擬し、過去（前年同月、最近の数日分等）の負荷推移や同系列サーバの現在の負荷推移を標準負荷と仮定する。

この標準負荷と該当サーバ負荷との差が大きいものを「問題サーバ」として洗い出すため、一定の比率を仮定する。

例えば、１週間の平日の平均負荷に対し、５０%以上の変化があれば異常とする。また、相関係数では０．７以下は異常として選択する等である。

なお、休日や月次処理日、障害発生日などの特例日は除外する必要がある。検証では手動で除外したが、ＣＰＵ使用率や変動係数が平常日に比べ明らかに異なること等を利用して自動除外することが可能と思われる。



④ 問題サーバの洗い出しの手順

前記①②③により、問題サーバの洗い出しのステップを次のように想定することが出来る。

＜負荷の位置付けによる洗い出し＞

第一段階：ＣＰＵ使用率が一定以下を除外

第二段階：ＣＰＵ使用率が一定時間に一定以上

第三段階：変動係数が一定以下、一定以上

＜標準負荷との差による洗い出し＞

第四段階：平均値の増率が一定以上

第五段階：変動係数の増減が一定以上、一定以下

第六段階：過去との相関係数が一定以下

第七段階：現在の他サーバとの相関が一定以下

（第二～七段階は全てＯＲ条件）

⑤ 負荷モデルによる検証

以上の段取りにより、サーバ数百台の５日間のＣＰＵ使用率をモデルデータ（ＥＳ／１　ＮＥＯにより採取）としてパソコンへ展開し、上記のステップに準じて洗い出しを試みた。

＜設定した定数例＞

a. 負荷２０%以下は判定対象から除外

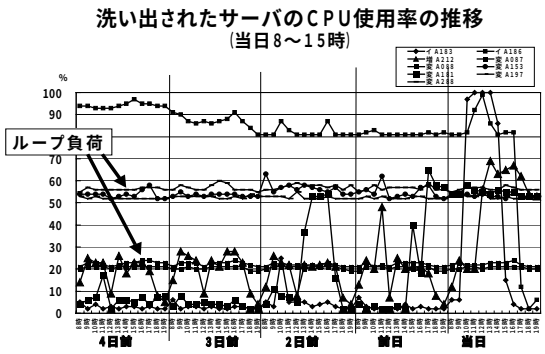
b. 変動係数５０%以上、２０%以下は異常

c. ２時間連続で８０%以上の場合は異常

d. ５０%以上の負荷増は異常

e. 変動係数の増減比が２５０%以上、
２０%以下は異常

この結果、異常サーバが１０台（０．５%）洗い出され、このうち半数はループ障害であることが確認出来た。



4.3 絞り込みによる優先順位付け

複数のサーバが同時に洗い出された場合、システム管理者は運用対応の優先順に迷うことがある。また狼少年的な警報は管理理不在となる危険性もある。

このため、「最も優先して早期に対応すべき問題サーバ」を順位付けして表示する方法の手順を考え、モデルデータにより検証することにした。

① 想定した可視化の様式

複数のサーバを管理する場合、数字の羅列は管理しにくいものである。

このため、洗い出して絞り込んだサーバの各棒グラフの下部に「警報理由」と「優先順位」をコメントする可視化の方法をとり、これを定期的に出力することで、詳細分析の要否をシステム管理者が迅速に判断できることを想定した。

② 優先順位付けの考え方

問題サーバの洗い出し手順は前記 4.2 と同じである。

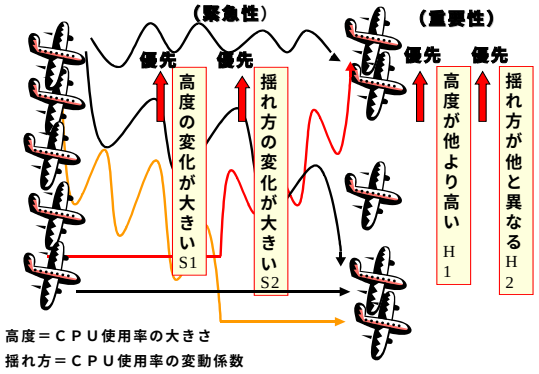
ターゲットの問題サーバの負荷が、過去（実績値）および現在の全体サーバの平均に比べ、どの程度の位置付けにあるかを数値で表し、優先度＝重要＋緊急性とする。

また、判断材料に重み付けして数値化することにした。

重要性　Ｈ１＝ＣＰＵ使用率の位置付け
　　Ｈ２＝変動係数の位置付け

緊急性　Ｓ１＝ＣＰＵ使用率の変化の位置付け
　　Ｓ２＝変動係数の変化の位置付け

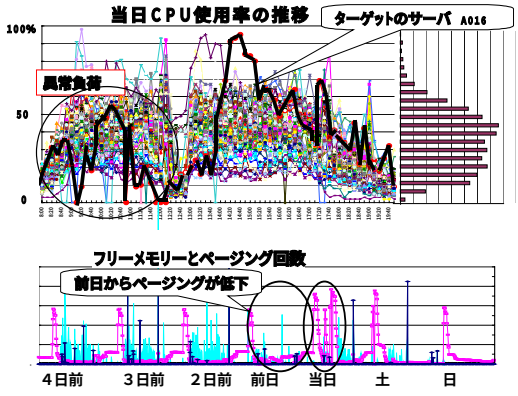
この 4 つの判断材料は編隊を組んで飛行するジェット機に例えることができる。



③ 負荷モデルによる検証

以上の順位付けの考え方により、同系列業務の高負荷のサーバ群（月～金曜日の５日間×５０台、１０分間隔のＣＰＵ使用率）をモデルデータとして検証した。

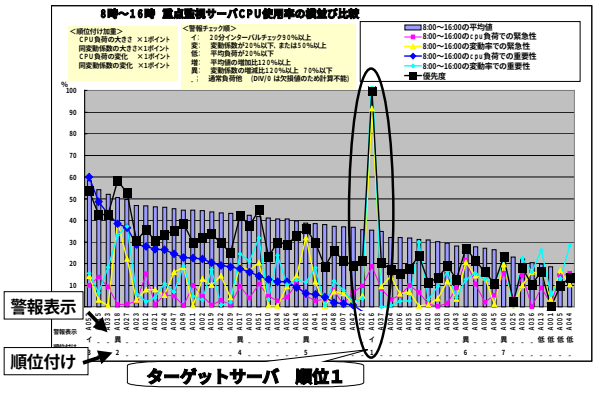
ターゲットサーバの負荷



当サーバ群の中には、当日にメモリー不足によりスローダウンした１台のサーバがある。のターゲットサーバが「優先順位１」として表示されるかを試みた。

この結果、検証前には異常ではないと思われた前日からの異常負荷の兆候も含め、ほぼ全時間帯での優先付けが可能であった。

検証結果グラフ（当日8～16時）

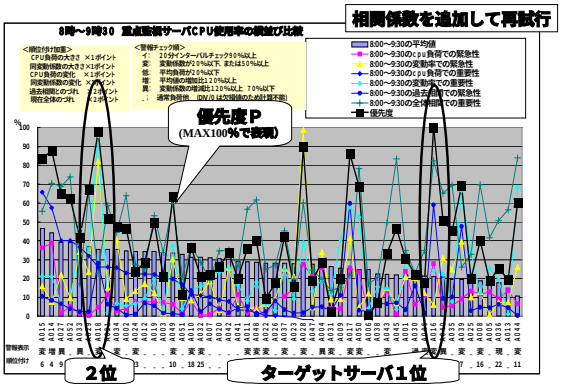


しかし、負荷が一度落ち込み復帰した時間帯（早朝１時間半）での絞り込みが出来なかった。

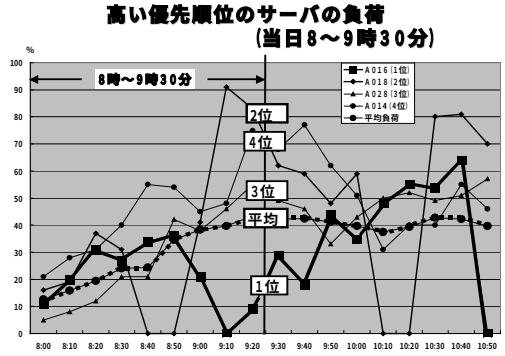
前記 4.3.②で示した４つの判断材料加え、標準負荷との相似度合いを判断する材料として、「緊急性：過去平均との相関係数」と［重要性：他サーバとの相関係数］を新たに追加し、重み付けも高くして再計算した。

この結果、短時間での異常負荷も優先順位１位として絞り込めた。

検証結果グラフ（当日8～9時30分）



高い優先順位のサーバの負荷
（当日8～9時30分）



検証で使用した詳細の計算式と重み付けの例を次に示す。

優先順位付けの計算式

		過去	現在
CPU使用率	対象サーバ	A	C
	全サーバ（平均）	B	D
同上変動係数	対象サーバ	a	c
	全サーバ（平均）	b	d
同上相関係数	過去との相関		E
	全サーバとの相関		e

① 重要性H ＝ H1＋H2＋H3
＝｜(C／D)－1)×100 ×加重h1＋｜((c／d)－1)×100｜ ×加重h2
＋(－e＋1) ×50×加重h3

② 緊急性S ＝ S1＋S2＋S3
＝｜((C／A)／(D／B))－1)×100｜×加重s1＋｜((c／a)／(d／b))－1)×100｜
×加重s2＋(－E＋1)×50×加重s3

（加重点数）

・CPU負荷の大きさH1　×1ポイント
・同変動係数の大きさH2　×1ポイント
・同全体との相関係数H3　×2ポイント　} 重要性H

・CPU負荷の変化S1　×1ポイント
・同変動係数の変化S2　×1ポイント
・同全体との相関係数S3　×2ポイント　} 緊急性S

優先の大きさp＝ 重要性H ＋ 緊急性S
＝ (H1×1+H2×1+H3×2) + (S1×1+S2×1+S3×2)
－100～+∞の範囲

優先度P ＝((p－minp)／(max p－min p)) ×100
0～+100の範囲

5．まとめ

本稿では、大規模なサーバ群の管理を想定し、問題サーバのＣＰＵ使用率を統計処理して相対的に絞り込むプロセス例について、モデル負荷により検証を試みている。この結果、固定値や重み付けを該当管理サーバ群の負荷実態に合わせて設定する等の工夫を必要とするが、複雑な負荷においてもほぼ洗い出しと優先順位付けが可能であった。

これにより、しきい値による管理と同様に相対的な管理手法も有用であることを示すことが出来たと思う。

相対的な管理手法は、一層拡大するコンピュータネットワークのキャパシティ管理の一手法として、多くの関係者の経験と知恵により洗練されていくべきものと考える。

参考文献：上野淳三、広田直俊、白井伸児「システム設計の考え方」
宮川公男「基本統計学」