簡単なシミュレーションによる

ボトルネック箇所の特定と改善効果の判定

河野 知行、 森本 舞
株式会社 アイ・アイ・エム 技術部 テクニカルサポートセンター

各種パフォーマンスツールを利用することにより、システム稼動状況を示すパフォーマンスデータを取得することができる。これらのパフォーマンスデータを解析しボトルネック箇所を指摘するのは容易であるが、チューニング策の効果を判定するには待ち行列などの知識が必要となる。本研究では、簡単なシミュレーションプログラムを作成し、この改善効果の策定を手助けする手法の可能性を探る。

Simple simulation program may helps

Bottleneck Analysis and Tuning Simulation

Tomoyuki Kawano, Mai Morimoto
Technical support Center, Engineering Department, IIM Corporation

We can easily collect performance data, when we use performance measurement tool on any platforms. Those performance data may lead us to know which part of system resource may become a performance bottleneck and what kind of tuning actions we should take. However do so, it is very difficult to figure out which tuning action may be most effective among these. If we have enough knowledge of Queuing Theory, it may help us choose most effective one. This paper describes our experience to use hand made small simulation program to evaluate an effectiveness of several tuning actions. From the experience, we may say, simulation program helps us lot to understand how tuning action resolve the problem with out deep knowledge of Queuing Theory.

1 はじめに

性能予測を行う場合に使用されるテクニックを大別すると、5種類に分類できる。経験則、線形解析、待ち行列、シミュレーション、ベンチマークの5つである。

最初の経験則とは、今までの経験を基に確立されたものである。最も有名なものは、ディスクの使用率が30%を超えると、そのレスポンス時間が悪化すると言うものである。

線形解析とは、複数のパフォーマンス指標間の相関判定を行ったり、回帰分析を行ったりするものである。つまり統計的な手法により、システム負荷が増加した際に生じるであろう問題点を推測しようとするものである。線形解析では現在の状況（傾向）を伸ばすことしかできない。このため、ある負荷量から生じ始める問題（ページングのようなもの）に関しては、推測が難しい。そのような待ち事象を数学的にアプローチするのが待ち行列技法である。

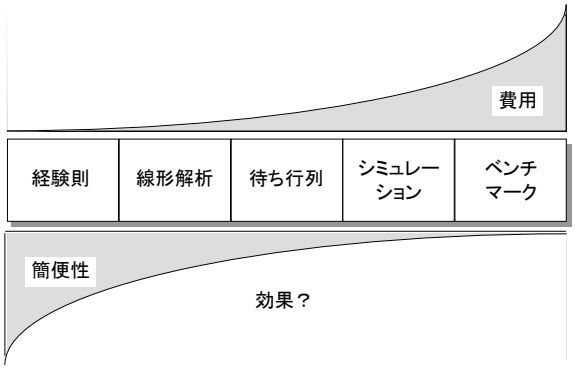


図1 性能予測テクニック

この待ち行列技法を使いこなすためには、数学的な知識が要求される。また、システムの細かな振る舞いを数式

化することが極めて困難であるため、業務要件を加味した予測を行うことが難しいとされている。そこで利用されるのが、シミュレーション技法である。

性能予測で最も正しい結果を得ることができるのがベンチマーク。簡単に言うと実機でのテストである。ストレステストと呼ばれる負荷テストを行う場合と、システム統合などの運用形態変更などを実機で行うものである。

2　シミュレーション

シミュレーションとは、どうのようなものであろうか。３０年ほど前からＧＰＳＳ（General Purpose Simulation System）と呼ばれるものがあった。シミュレーションを行うための言語環境を提供するためのものである。現在ではＧＰＳＳ以外にも、数多くのシミュレーションツールが販売されている。

このシミュレーション、別にツールを購入しなければ行えないものではない。事実、多くのＳＥ（System Engineer）はＦＯＲＴＲＡＮでシミュレーションプログラムを作っていた。今回の研究では、ＶＢ（Visual Basic）でシミュレーションプログラムを作成した。

シミュレーションに関する基本的な考え方を紹介しよう。シミュレーションプログラムは一つ一つの事象の発生を、時間の流れとともに追跡しようと言うものである。ここでは、その例としてディスクアクセスについて考えて見る。

複数のアプリケーションが一台のディスクを共有し、ランダムにアクセスしているとする。ディスクは同時に一つのアクセス要求しか処理できないとすると、そのレスポンス時間はどのように変化するのか。

この例は、待ち行列のM／M／１による予測と同じである。ここでM／M／１の意味を説明しておこう。最初のMは、ディスクのアクセス時間間隔は一定ではなく、ある平均値となるランダムさを持った値であることを意味している。より厳密に定義すると、ポイソン分布に従うランダムさと定義されている。２つ目のMは、サーバ（今回の場合はディスク）のサービス時間がポイソン分布に従うランダムさを持ったものであることを意味している。最後の１は、サーバが同時には一つのアクセス要求しか受付ないことを意味している。

3　プログラムステップ

このシミュレーションを行うには、図２に示すようなプログラムを作成する。このプログラムを簡単に説明しよう。このプログラムでは４つの変数を使用する。現在の時刻、ディスクのサービス時間、ディスクの動作終了時刻、ディスクのレスポンス時間の４つである。

最初のステップ①では、ディスクアクセスが行われる時間間隔を決定する。ランダムにアクセスするとの想定であるため、目標となる平均時間間隔が達成されるように乱数を発生させる。具体的にはポイソン分布を使用するために単純な乱数の自然対数を求め、それに平均時間間隔を掛け算して求める。ＶＢ言語で書くならば、次のような計算式となる。

時間間隔=ABS(平均値*LOG(RND))

ステップ②では、そのディスクアクセスに必要なサービス時間を決定する。こちらもステップ①と同様に、自然対数を用いた乱数を発生させる。ステップ①と②で、M／M／１の２つのMの処理を行ったことになる。

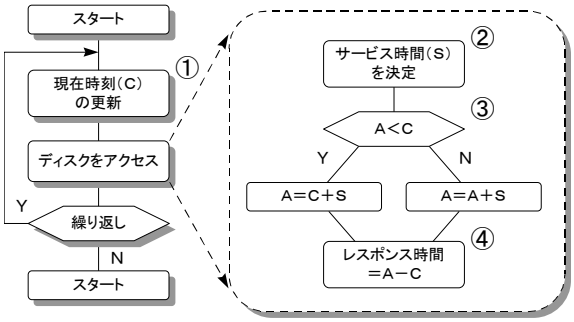
ステップ③では、ディスクの動作終了時刻と現在の時刻を比較して以降の処理を決定する。「Y」へ分岐するのは、ディスクの動作完了時刻が現在の時刻よりも小さい時、つまり現在ディスクが動作していない場合である。その場合、現在の時刻にサービス時間を加えた値を新しい動作終了時刻とする。

もしディスクが動作しているタイミングでアクセス要求が来ると「N」へ分岐する。この場合、ディスク動作が完了するのを待ってから、次のディスクアクセスを行う。このため、このシミュレーションにおいては動作終了時刻にサービス時間を加えた値を新しい動作終了時刻とする。

ステップ④では、そのアクセスのレスポンス時間を計算する。このステップは簡単であり、ディスクの動作終了時刻から現在の時刻を引き算した値をレスポンス時間とする。

ステップ②で決定したサービス時間と、ステップ④で求められたレスポンス時間を蓄積し、ディスクの使用率や、平均サービス時間、平均レスポンス時間を求める。

以上の処理は、ディスクアクセスを忠実になぞったものであり、充分な回数だけ繰り返し実行することにより正しいシミュレーション結果を得ることができる。



A:ディスクの動作終了時刻　C:現在の時刻　S:サービス時間

図2　シミュレーションのフロー

4　シミュレーションの実行

シミュレーションの実行を時間の流れに沿って考えて見てみよう。図3に示した3つのディスクアクセスを考える。アクセス①と②は、ディスクが動作していないタイミングでのディスクアクセスである。これらのアクセスをシミュレーションする場合、図2のステップ③では「Y」へ分岐する。

アクセス③はディスクが動作している最中のディスクアクセスである。このため先行したアクセスが完了するのを待つ必要がある。この場合にはステップ③で「N」へと分岐する。このアクセス③で生じる待ち時間のことを、アクセス待ち時間と呼んでいる。

アクセス③をシミュレーションする時、先行したアクセスの終了時刻はプログラムの変数「A」に、またアクセス③が実行された時刻は変数「C」に記録されている。故に、この待ち時間は「A−C」で求められる。

早速、このプログラムを実行させることにしよう。しかし、実行する際に明確にしておかねばならない条件がある。それは、アクセス時間間隔と、ディスクのサービス時間の平均値を幾らにするかと言うことである。ここでは、話しを簡単にするためにディスクの平均サービス時間を１０ミリ秒、またディスクをアクセスする時間間隔を３０ミリ秒とする。すると、このディスクの使用率は３３．３３３・・%になるはずである。

プログラム実行の結果を見てみよう。図4に、このプログラムで計算された待ち時間とサービス時間の平均値の推移を示す。この図では、横軸にプログラムの繰り返し回数を、また縦軸に時間（ミリ秒）を示す。今回は、プログラムを2万回、繰り返し実行した。その結果、平均サービス時間は１０ミリ秒で、また待ち時間は5ミリ秒で安定した。

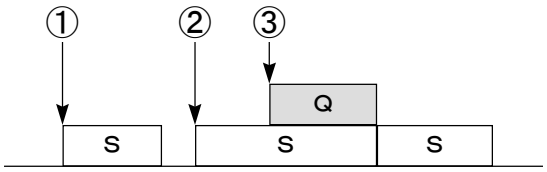


図3　アクセスタイミング

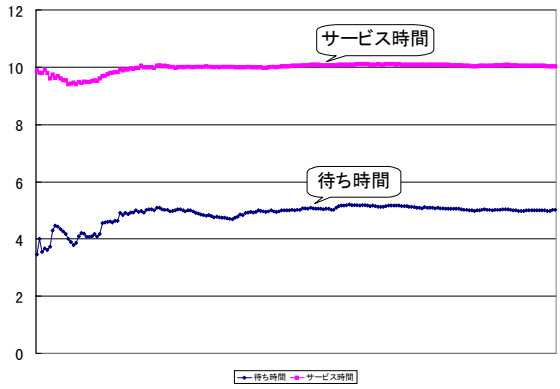


図4　サーバ動作の安定化

5　シミュレーションの基本

シミュレーションでは「資源」と「事象」そして「処理」の3つを考える必要がある。「資源」とはディスクやＣＰＵなど、待ち行列モデルを構成するサーバのことである。一方、「事象」とは、それらの資源をアクセスするアクセス要求のことである。事象が生成されると、予め定められた順序で資源をアクセスする。この資源アクセスの順序を決定するのが「処理」である。

違った言い方をするならば、「資源」はその待ち行列モデルを構成するシステムのことである。資源の一つ一つはハードウェアやソフトウェアの資源を意味する。ソフトウェア資源としては、バッファメモリなどが上げられる。「処理」はそのシステムで動作する業務プログラムであり、「事象」はそのプログラムが処理すべきトランザクションである。

事象が資源をアクセスする際、その資源の状態を見ながら事象を制御する必要がある。もし、資源が先行する事象により使用中である場合、事象はその資源を直ちに使用する事が出来ない。このような場合、その事象は資源の空くのを待つ必要がある。

図2で示したシミュレーションロジックは、非常に簡素化されたものである。「資源」と「事象」それに「処理」の3つの要素を考慮したものではない。3つの要素の内、「資源」だけに着目したものである。図2のロジックでは、新たにディスクをアクセスする事象は、先行した事象を追

い越すことは考慮されていない。つまり、事象の処理順番は、常に変わることがない。

並列処理を可能とするような資源（複数サーバを持つステーション）の場合、同時に複数の事象を処理する事が可能である。また、それぞれの事象のサービス時間は一定でないため、事象の順序が変わる可能性がある。例えば、先に到着した事象が資源を２０ミリ秒使用し、直後に到着した事象が資源を５ミリ秒使用したとすると、後に処理を開始した事象が先に資源使用を完了する事になる。並列処理を可能とする待ち行列モデルとは、M／M／2など、複数のサーバを持つ資源である。例えば、複数のＣPUを持つプロセッサなどである。

どのようにして、この事象の追い越しを管理するのか。図2のシミュレーションプログラムでは、記憶域としてディスク（つまり資源）に関する変数域しか準備していなかった。レスポンス時間（R）、サービス時間（S）、ディスクの動作終了時刻（A）、それに現在の時刻（C）である。

事象の追い越しを管理するには、事象に関する変数域を準備する必要がある。先ず、その事象が生成された（誕生した）時刻である。その事象がシミュレーションプログラム内で生成されてから、全ての処理が終了されるまでが、その事象のエラップス時間である。このエラップス時間を求めるために、事象の生成時刻を記録しておく必要がある。

また、資源をアクセスしている事象を個別に管理する必要がある。具体的には、資源を配列定義し今資源を使用している事象、待ち状態になっている事象を個別に管理する。

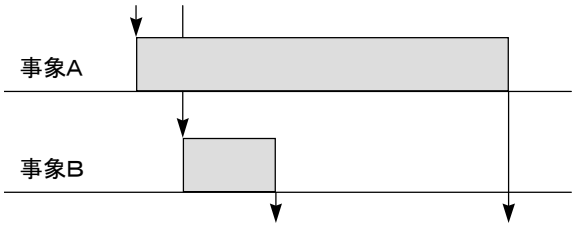


図5　事象の追い越し

6　キャッシュ制御のシミュレーション

ＣPUやディスクなどの資源は、前述した資源の考え方で対応できる。しかし、バッファメモリーやキャッシュメモリー（以降キャッシュと略する）は、ちょっと異なった制御を必要とする。

キャッシュには、頻繁に使用するデータを蓄積しておく。もし、キャッシュ内に目的のデータが存在しなければ、ディスクからの読み込み動作が必要となる。もし、目的のデータがキャッシュ内にあればディスクのアクセスは必要でない。

この操作をプログラミングすることを考えてみよう。最初にキャッシュが存在しないプログラムを考える。つまり、常にディスクへのアクセスが行われるものである。このロジックにキャッシュを想定した処理を埋め込む。例えば９０%のヒット率をシミュレーションする場合、１０%の要求に対してのみディスクをアクセスするロジックを付け加えれば良い。非常に簡単である。

7　問題点の整理と情報収集

とあるお客様のＤBサーバで、特定時間帯にレスポンス時間が悪化すると言う問題が発生していた。色々と調査しておられたが、今一つ問題点がスッキリしない。そこで我々も参加させていただくことになった。

先ずはシステム構成をインタビュー。このＤBサーバは４ＣPU構成でＯｒａｃｌｅが動作。ディスクはＲＡＩD構成で巨大なキャッシュを搭載している。ＤBサーバとＲＡＩDディスク間は４本のＳＣＳＩ（Small Computer Systems Interface）で接続されている。

問題発生時、ＣPU使用率は７０%でありＯｒａｃｌｅバッファヒット率は９０%、何れも問題ある状況とは考えられない。またＲＡＩDディスクのキャッシュヒット率は７７%程度であり、ディスクベンダーも問題ないとしている。しかし、最もＩ／O回数が高いＨDＤ（Hard Disk Drive）には、秒あたり５５回程度のＩ／Oが出ている様子。そのＨDDの仕様によると平均サービス時間は１０ミリ秒程度であるが、こちらも大きな問題ではないとされている。

大まかに整理すると以上のような状況であった。その後、お客様の担当者とのディスカッションを行った。担当者によると、現在の４ＣPU構成を５ＣPU構成に増強するには予算上、問題が多すぎる。ＲＡＩDディスクを入れ替えるにも予算の問題が生じるとのこと。もう一つ、システム上の制限がありＳＱＬのチューニングを行う事は出来ない。何とかチューニングで問題を回避することは出来ないであろうかとのことであった。

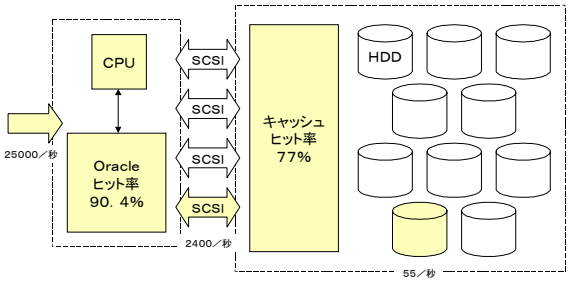


図6　問題発生のシステム

8　シミュレーションフロー

この問題を調査するにあたり、最初からシミュレーション技法の適用を考えた訳ではない。最初は、通常の問題分析アプローチの採用を検討した。例えば、各種パフォーマンスデータを時系列的に並べ、分析する方法である。

しかしこのお客様の場合、ＲＡＩＤディスクの評価も求めておられる。また、そのデータはベンダー提供の専用ツールでしか収集できない特殊なものである。このため、今回はシミュレーションによる問題分析を行うものとした。

ここでシミュレーションを行うモデル定義を行った。以下に述べる数値は、お客様から提供されたものをベースに決定されたものである。

ｎ回（実測値無し）のＯｒａｃｌｅ要求が出た際のＣＰＵ使用率は７０%である。ｎ回の内、９０%はＯｒａｃｌｅバッファでヒットする。つまり、ｎ回の内、１０%が実Ｉ／ＯとしてＳＣＳＩを経由してＲＡＩＤディスクに渡される。

この際、４本のＳＣＳＩでは負荷分散処理が行われ、それぞれのＳＣＳＩが６００回／秒のＩ／Ｏを処理する。ＳＣＳＩは一本あたり４０MB／秒の能力を持っている。

ＲＡＩＤディスクのキャッシュでは７７%がヒットする。キャッシュヒットしなかった２３%の実Ｉ／ＯがＨＤＤアクセスで処理される。最も多くの実Ｉ／Ｏを処理するＨＤＤのＩ／Ｏ回数は秒あたり５５回である。

１本のＳＣＳＩには６００回／秒のＩ／Ｏが処理されている。４本のＳＣＳＩが準備されているので、全体で２４００回／秒の実Ｉ／Ｏが行われていることになる。よって、Ｏｒａｃｌｅバッファのヒット率を勘案すると、このＤＢサーバには２万４千回分のＯｒａｃｌｅ要求が出ているものと考えられる（とした！）。

細かな事を書き始めると切りがない。概要レベルであれば、前述した程度であろう。前述の条件をフローで表したのが、図7である。このフローを見ていただくと我々が考えたことがお判りいただけると思う。ＣＰＵ，ＳＣＳＩ，ＨＤＤの３つの資源と、Ｏｒａｃｌｅバッファ、ＲＡＩＤディスクキャッシュの２種類のキャッシュ制御に着目した訳である。

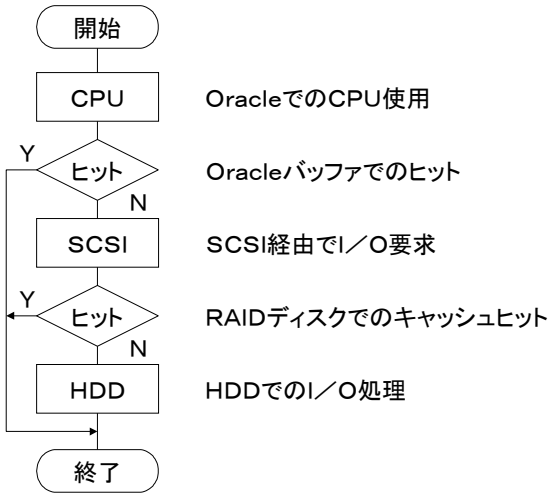


図7　シミュレーションフロー

9　シミュレーション結果

このシミュレーションを行った結果、ＨＤＤのレスポンス時間が極端に悪化する事が判った。その結果を待ち行列技法で紹介しよう。

先ずＣＰＵであるが、４ＣＰＵ構成であるため使用率が７０%である際のレスポンス時間を考えると、次式のようになる。

$$R = Q + S = Q_{[70]} + S = 0.333S + S = 1.333S$$

ＳＣＳＩは、４つのＭ／Ｍ／１の組み合わせである。それぞれのＳＣＳＩは４０MB／秒の能力を持っており、Ｏｒａｃｌｅは一回のＩ／Ｏで８ＫＢを転送していると考える。するとその使用率やレスポンス時間は、次のようになる。

$$\rho_{SCSI} = \frac{8 \times 600}{40 \times 1024} = 0.117$$

$$R = \frac{S}{1-\rho} = \frac{S}{1-0.117} = 1.133S$$

次にＨＤＤについて考えて見よう。ＨＤＤには、Ｍ／Ｍ／１が適用できる。その使用率とレスポンス時間は、次のようになる。一回のＩ／Ｏでのサービス時間は、お客様

が調査された１０ミリ秒を採用した。

$$\rho_{HDD} = \frac{10 \times 55}{1000} = 0.55$$

$$R = \frac{S}{1-\rho} = \frac{S}{1-0.55} = 2.222S$$

ＣＰＵ、ＳＣＳＩ、HDDのサービス時間は、それぞれ固有のものである。また、Ｏｒａｃｌｅバッファ やRAＩＤキャッシュがあるため、Ｉ／O回数はＯｒａｃｌｅ＞ＳＣＳＩ＞HDDの規則が成立する。

Ｏｒａｃｌｅレベルでは２４０００回、ＳＣＳＩレベルで６００回、HDDレベルで５５回のＩ／O要求があったとしている。これらの条件を基に、待ち行列計算を行えば、それぞれの資源のレスポンス時間を求める事が出来る。また、ＲＡＩＤディスク全体のレスポンス時間を求めるには、HDDの負荷分布状況を加味する必要があることを忘れてはならない。

前述したように、この問題の原因は「HDD」であった。HDDの待ち時間が増大し、これ以上のＯｒａｃｌｅ要求が来てもレスポンス時間が悪化するだけであり、スループットが向上する事はない。待ち行列であれシミュレーションであれ、同じ結論に達する。

１０　チューニングの方策

このHDDの問題を解決するために、如何なるチューニング方法が考えられるか。キャッシュ効果を高め実Ｉ/O回数を削減するか、それともHDDの性能を高めるべきか。お客様の要望によりＳＱＬチューニングは行うことが出来ないとする。また、お客様はＣＰＵ能力が足りないのではと危惧しておられるので、その点も確認したい。

そので、次のチューニングによる効果予測をシミュレーションで行った。

Ｏｒａｃｌｅバッファヒット率を９０%から９５%へ
ＲＡＩＤキャッシュヒット率を７７%から８２%へ
HDD性能を５%改善
ＣＰＵを４枚から５枚へと増強

その結果、トータルレスポンス時間の改善度合いは、次のようになった。

項目	エラップス（ms）	改善率
現状	0.67	0
Oracleバッファ	0.33	0.507
RAIDキャッシュ	0.46	0.313
HDD能力	0.61	0.09
CPU枚数	0.65	0.03

図8　チューニング手法とその改善率

１１　今後の課題

待ち行列計算でも同じ結果を得ることは出来るが、お客様への説明となると「難しい！」、「正しいのか！」などとの反応が多い。今回の予測結果の報告では、シミュレーションプログラムをデモで見ていただきながら説明したところ「非常に理解し易い」とのお客様のコメントを頂いた。

シミュレーションプログラムによる性能予測の疑似体験が、プレゼンテーションの助けになったと確信する。またその場でお客様から出されたチューニング案のシミュレーションを、その場で行うことが可能となり好評であった。

当社では、線形解析によるシステム特性判定ツール、ルールベースによる性能分析ツール、そして待ち行列モデルやシミュレーションによる性能予測ツールを自社開発している。これらのツールにはそれぞれ長所・短所がある。それぞれの特長を活かした形で、よりユーザフレンドリーなプロダクト開発に努めたい。

１２　参考文献

野瀬純郎：情報システムの性能評価技法、
ＩＩMレポート１９９８　NO１０５～１０９

以上