ということです（線形回帰が前提としているのは加法的関係です）。言い換えれば、二つの要因 *a* と *b* の効果を掛け合わせると、以下のモデルになります。

$$y = ab \tag{19}$$

この関係は線形回帰でそのままモデル化するには不適当です。しかし、両辺を自然対数で表すと、以下が得られます。

$$\ln y = \ln a + \ln b \tag{20}$$

このような方法でモデル化すると、加法的効果に真数を適用して乗法的効果を決定できます。

落とし穴13（乗法的効果）

ベンチマークや、容量計画の実施において、乗法的関係は陥りやすい落とし穴といえます。Jain [20, pp. 304]は、要因としてワークロードサイズとCPU速度を含む実験計画を一例として挙げています。単純なシステムでは、これら二つの要因の相互関係は明らかに乗法的です。しかし、データベースアプリケーション・ベンチマークのようにシステムが複雑になると、その効果は必ずしも乗法的、加法的ではなく、より複雑な関数が関わります。これは、CPU速度はデータベースアプリケーションのパフォーマンスに間接的な効果を与えることが多いからです。前述のように、*すべてのCPUは一定の処理速度に従い待機します*。

状況によっては、検討中の要因を単純な方法で分析すると、加法的モデルの放棄につながる恐れがあります。また、乗法的効果の可能性を示す指標がいくつかあります。一つは、試験スペクトラムにわたって応答変数 *y* の最大値と最小値の比率が大きいことです。この比率が大きい場合（たとえば、最大値より一桁大きい、またはそれ以上）、対数変換を考慮する必要があります。対数変換を考慮するその他の指標は、残差が応答変数と同じ桁である、または残差の分位点分位点プロットが正規分布に従わない場合です[20, pp. 304–8]。

2^k 実験計画の統計分析

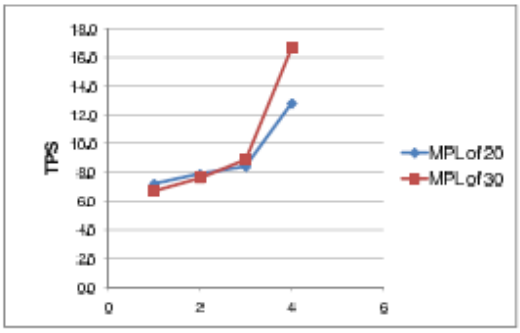
前項において省略した形ではありますが、2^2 実験計画の結果を表す線形回帰モデルについて述べました。ここでは、*k* 個の要因の効果を測定するためにそのモデルを一般化し、それぞれの要因を二つのレベルで一般化します。二つのレベルそれぞれでの *k* 個の要因の分析には、2^k 回の実験が必要です。

例6（仮定的 2^k 実験）

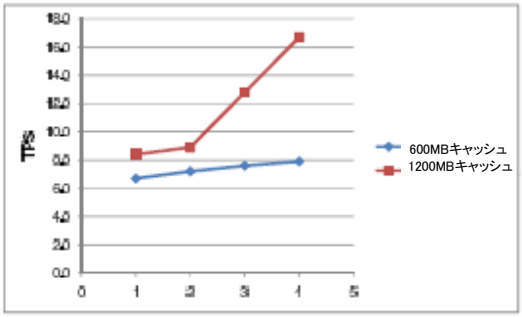
この例では、*k*=2 に追加要因としてクライアント接続数を追加した *k*=3 実験計画に変更します。サーバーハードウェアの特性など他の二次的要因は一定のままと想定します。今回の容量計画の検討では、システムパフォーマンスの全体に与えるデータベース・キャッシュサイズ、サーバー・マルチプログラミングレベル、およびクライアント接続数の相対的効果を測定します。今回も、1秒当たりのトランザクション処理件数で表すトータルスループット（応答変数）で測定します。

要因	レベル 0	レベル 1
マルチプログラミングレベル A	20	30
データベース・キャッシュサイズ B	600MB	1200MB

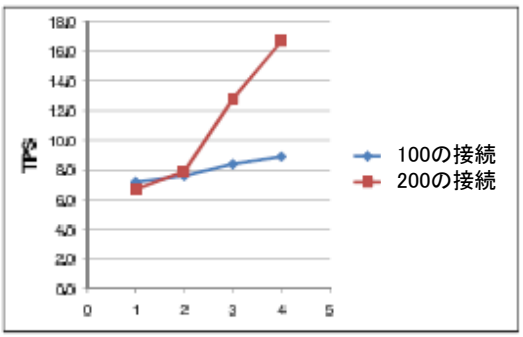
クライアント接続数 C	100	200



(a) マルチプログラミングレベルに関する
TPS散布図



(b) データベースキャッシュサイズに関する
TPS散布図



(c) 接続数に関するTPS散布図

図2 2^3 実験データの散布図

表6は、2^3 の仮定的実験の結果として、8 個の実験結果を示しています。

キャッシュサイズ (MB)	20 の MPL		30 の MPL	
	100 の接続	200 の接続	100 の接続	200 の接続
600	7.2	7.9	7.6	6.7
1200	8.4	12.8	8.9	16.7

表6 2^3 実験結果（1秒当たりのトランザクション処理件数）

落とし穴14（多重共線性）

2^k *実験計画*やその他の実験計画を考慮するときに、分析内にできるだけ多くの要因を含めようとしてしまうかもしれません。*k* 値が大きい場合の一つの落とし穴は、必要な実験数が飛躍的に増えてしまうことです。しかしながら、線形回帰分析に共通する他の問題として、*多重共線性*、つまり二つ以上の要因間の相関関係が挙げられます[20, pp. 253-4]。例6では、クライアント接続数を要因セットに追加することが問題となります。たとえば、データベースインスタンスのサイズの場合は、分析の主要因として含めるべきか、などが問題となります。前述の「代表的ワークロードの作成」では、実際の状況を適切にモデル化するためのワークロードを慎重に作成する重要性について論じました。クライアント接続数を倍にすると、代表的データベースのサイズも相応に大きくなる可能性があります。この相関関係は、同じ分析においてその二つの要因が両方ともに考慮され、そしてそれらの（測定された）相関関係が有意である場合、矛盾する有意性結論に至る可能性があります。

2^k 計画の線形回帰分析は前述の 2^2 実験計画の分析を一般化したものです。2^2 実験と同様に、方程式(5)は重線形回帰モデルを表します。*k* 個の要因では、交互作用の複数の組み合わせを考慮することになり、複雑さが増します。つまり、k > 2 の場合は、*k* 個の主効果、$\binom{k}{2}$ 個の2因子交互作用、$\binom{k}{3}$ 個の3因子交互作用などが発生します。幸いにも 2^k 実験計画では、以前に述べた符号テーブル方式を利用し、モデルの係数を決定するために一連の一次方程式をより簡単に解くことができます。

サーバー・マルチプログラミングレベルとデータベース・キャッシュサイズをそれぞれ表す変数 X_A と X_B（方程式8）に加えて、クライアント接続数である追加要因 *C* を表すもう一つの変数が必要になります

$$X_C = \begin{cases} \text{100クライアントの接続の場合は−1} \\ \text{200クライアントの接続の場合は+1} \end{cases} \qquad (21)$$

2^2 の符号テーブルを作成したときと同じように 2^3=8 の結果を表す符号テーブルを作成します。この仮定的例の結果となる符号テーブルを表7に示します。

実験	*I*	*A*	*B*	*C*	*AB*	*AC*	*BC*	*ABC*	*y*
1	1	-1	-1	-1	1	1	1	-1	7.2
2	1	1	-1	-1	-1	-1	1	1	7.6
3	1	-1	1	-1	-1	1	-1	1	8.4
4	1	1	1	-1	1	-1	-1	-1	8.9
5	1	-1	-1	1	1	-1	-1	1	7.9
6	1	1	-1	1	-1	1	-1	-1	6.7
7	1	-1	1	1	-1	-1	1	-1	12.8
8	1	1	1	1	1	1	1	1	16.7
	76.2	3.6	17.4	12	5.2	1.8	12.4	5	合計
	9.525	0.45	2.175	1.5	0.65	0.225	1.55	0.625	合計/2^3

表7 2^3 実験計画の符号行列方式

表7で、平均パフォーマンスは 9.525TPS で、三つの要因（マルチプログラミングレベル、キャッシュサイズ、および接続数）の効果はそれぞれ q_A=0.45、q_B=2.175、q_C=1.5 になります。興味深いことに、この一連の実験においてはキャッシュサイズと接続数の交互作用効果（列 *BC*）は1.55であり、接続数単独の効果より大きくなります。この実験計画の全変動(SST)は、以下のとおりです。

$$\begin{aligned}\mathrm{SST} &= 2^3(q_A^2 + q_B^2 + q_C^2 + q_{AB}^2 + q_{AC}^2 + q_{BC}^2 + q_{ABC}^2) \\ &= 8(0.2025 + 4.73 + 2.25 + 0.4225 + 0.05 + 2.4 + 0.39) = 83.595. \quad (22)\end{aligned}$$

各要因に割り当てられてた変動部分は以下のとおりです。

要因	SST コンポーネント	変動部分
マルチプログラミングレベル A	1.62	1.62/83.595 = 1.93%
データベース・キャッシュサイズ B	37.85	37.85/83.595 = 45.27%
接続数 C	18	18/83.595 = 21.53%
AB	3.38	4.04%
AC	0.405	0.004%
BC	19.22	22.99%
ABC	3.125	3.73%

この調査から、サーバー・マルチプログラミングレベルを20から30に上げても、このワークロードに与える効果は比較的重要ではないと結論づけられます。これは、図2(a)でも示されているとおりです。100および200のクライアント接続の応答変数(TPS)を表す二つの線はほぼ同じです。しかし、このワークロードでは、データベース・バッファープールのサイズはパフォーマンスの変動の45%を占め、接続数は22%弱を占めます。さらに、これらの二つの要因はあわせて23%の変動を占めます。以上のことから、バッファープール・サイズはこのワークロードのサーバーパフォーマンスにおいては重要な要因であると結論づけられます。

2^k_r 実験計画の分析

2^k_r 実験計画の背景にある考えは、実験誤差を推計できるように各実験を *r* 回繰り返すということです。

線形回帰モデルの実験誤差は方程式(5)の *e* 項によって示されます。

落とし穴15（実験誤差の査定）

実験誤差を査定するために、実験を繰り返すメリットは重要視されないことが多いようです。実験の繰り返しに反対する代表的な主張は、その費用を除けば、被試験ソフトウェアが決定的であり、各実験で同一結果が得られるはずというものです。この主張の欠点となるのは、複雑なシステムの場合、すべての潜在的なソフトウェアの交互作用、とくにビジネスアプリケーションにおいては一般的である日付や時間帯に依存するソフトウェアの交互作用を予想するのは困難であるということです。

符号行列方式を使って *r* 回繰り返された実験を活用する簡単な方法は、各2^k 実験の y_{ij} の *r* 回行われた観測それぞれの平均 $\bar{y}_i$ を取ることです。y_{ij} は *i* 回目の実験の *j* 番目のレプリケーションを示します。*j* 回目の実験それぞれの応答変数として $\bar{y}_i$ を使用し、2^2 実験計画の線形回帰モデルは、以下のように*期待*応答値 $\hat{y}_i$ を導きます。

$$\hat{y}_i = q_0 + q_A X_{A_i} + q_B X_{B_i} + q_{AB} X_{A_i} X_{B_i} \qquad (23)$$

要因 *A* と *B* は、X_{Ai} および X_{Bi} の値をとります。*r* 回実行された観測それぞれとその期待値の差は実験誤差を表します。

$$e_{ij} = y_{ij} - \hat{y}_i = y_{ij} - q_0 - q_A X_{A_i} - q_B X_{B_i} - q_{AB} X_{A_i} X_{B_i}. \qquad (24)$$

2^2 の場合の係数計算（方程式11を参照）と同様に、各 q_j を計算する数式は以下のとおりです[20, pp. 309]。

$$q_j = \frac{1}{2^k} \sum_{i=1}^{2^k} S_{ij} \bar{y}_i \qquad (25)$$

S_{ij} は、その実験の一連の観測 y_i *に対して*、符号テーブルの行 *i* 、列 *j* の項目を示します。

変動の割り当て　各 $2^k r$ 観測の実験誤差を求めるために方程式(24)を使用できます。定義によれば、誤差の合計は 0 になります。誤差の分散および効果の信頼区間を推計するため、残差平方和（sum of the squared errors、SSE）を以下のように計算することができます。

$$\mathrm{SSE} = \sum_{i=1}^{2^k} \sum_{j=1}^{r} e_{ij}^2 \qquad (26)$$

SSEは、変動割り当ての今回の査定における一つのコンポーネントです。

$$\mathrm{SST} = \mathrm{SSA} + \mathrm{SSB} + \mathrm{SSAB} + \mathrm{SSE}. \qquad (27)$$

2^2 実験計画での計算と同様に、代数を使用して、SSTやその他の平方和を求める各種の数式は以下のとおりです[20, pp. 309]。

$$\mathrm{SSY} = \sum_{i=1}^{2^k} \sum_{j=1}^{r} y_{ij}^2 \tag{28}$$

$$\mathrm{SS0} = (2^k r) q_0^2 \tag{29}$$

$$\mathrm{SST} = \mathrm{SSY} - \mathrm{SS0} \tag{30}$$

j=1,2,..2^k-1の場合すべてに対して $\mathrm{SS}j = (2^k r) q_j^2$ (31)

$$\mathrm{SSE} = \mathrm{SST} - \sum_{j=1}^{2^k-1} \mathrm{SS}j \tag{32}$$

j 番目の効果による変動のパーセンテージは以下の比率から計算されます。

$$\frac{\mathrm{SS}j}{\mathrm{SST}} \times 100\%. \tag{33}$$

定義によれば、誤差の合計は0にならなければなりません。なぜなら誤差は、各実験の *r* 回の観測それぞれと応答変数の平均値 $\bar{y}_i$ との差から計算されるからです。その誤差が正常に分配されると仮定すると、2^k_r 実験計画の誤差の変動は次の数式を使用してSSEから推計することができます。

$$s_e^2 = \frac{\mathrm{SSE}}{2^k(r-1)}. \tag{34}$$

各効果の信頼区間の測定を含め、上記の結果をベースとした追加の統計テストを実行することができます。Jain [20,pp. 298]は線形回帰モデルにより、全効果の変動は $s_e^2/(2^k r)$ と等しいことを示しています。従、各効果の標準偏差は $s_e/\sqrt{(2^k r)}$ なります。スチューデントのt分布を使用し、各効果の信頼区間は以下の式で求められます。

$$q_j \pm t_{[1-\alpha/2;2^k(r-1)]} s_{q_j}. \tag{35}$$

パフォーマンス評価ツール

様々なパフォーマンス評価ツールが市販されており、データベースアプリケーションの合成ワークロード作成の支援や、様々な実験の実行と実験結果のトラッキングに利用されています。市販ツールの中で人気が高いのは、高額ですが、Mercury Software (http://www.mercury.com)製のLoadRunnerスイートがあります。それとは対極の位置にあるのが、SQL Anywhere標準装備のTRANTESTユーティリティープログラムです。TRANTESTはマルチユーザーのクライアントサーバー・システムのパフォーマンス評価を行うには単純ですが使いやすいツールです。TRANTESTには、ユーザーシンクタイムを取り入れるメカニズムが単純であるなどの欠点があります。

Sybase iAnywhereのコンサルティング・サービスグループは、LoadRunnerの機能と似た機能を提供するツールFloodgateを提供しています。Floodgateは、双方向ワークロードを完全に再現するために、複数のクライアントを発生させ、異なるスクリプトを実行することができます。また、レスポンスタイムや他のパフォーマンス指標を測定し計算する測定機能も提供します。

上記のツールやSQL Anywhere同梱のSybase Centralグラフィカル管理ツールに加えて、その他の方法でもSQL Anywhereサーバーからパフォーマンス統計データを取得することができます。Microsoft Windowsの環境で最も一般的な方法としては、Windows NT Performance Monitorがあります。SQL Anywhereサーバープロセスは Performance Monitorにサーバー固有およびデータベース固有の各種パフォーマンスカウンターを提供します。たとえば、1秒当たりのページ読み込み量、動作中および待機中の要求の平均数などです。この情報は、パフォーマンス評価におけるパフォーマンスボトルネック診断の際やモデルのワークロード有効化を支援する際に非常に重要になる可能性があります。しかし、パフォーマンスボトルネックの診断は非常に時間を要する作業であり、相当の熟練を要するということを警告しておきます。パフォーマンス評価調査のプロジェクトマネージャーは、必要に応じてこの診断を実行するための十分なリソースが利用できるようにしておかなければなりません。

落とし穴16（Task Managerプロセス統計の解釈）

Windowsでよくある問題は、物理メモリーがデータベースサーバーによってどのくらい使用中なのかを特定することです。システムで動作中のプロセスをすべてリストするWindows Task Managerは、“Memory Size”の列に実際のメモリー使用量を報告するのではなく、代わりに各タスクの*ワーキングセット・サイズ*を報告します。大まかに言えば、Windows XPや他の類似Windowsオペレーティングシステムにおいては、ワーキングセット・サイズはメモリーに常駐し、当該プロセスによって最近使用された物理RAM量を指します。

ここでの問題は、Windowsがその“最近使用された”をどのように定義しているかです。Windowsは定期的にプロセスのワーキングセットを“削減”します。これは、プロセスに割り当てられたメモリーのほとんどまたはすべてを“存在していない”ものとしてマーキングします（そのメモリーがスワップアウトされたかのようになります）。しかし、Windowsは実際にはそのメモリーをスワップアウトしません（ただし他の目的に真っ先に再利用されます）。もしそのプロセスがそのメモリーを再び参照する場合、そのプロセスは仮想メモリー障害を引き起こします。Windowsはこの“ソフトフォールト”を発見し、そのプロセスが実際にはメモリーの削減部分を使用していることを認識し、そのプロセスのワーキングセットを相応にインクリメントします。通常Task Managerは、あるプロセスが最小化されているとき、そのプロセスのワーキングセット・サイズを激減して報告します。つまり、Windowsは最小化されたアプリケーションはほとんどメモリーを使用しないと仮定し（データベースサーバーに対しては正し

くない）、そのプロセスのワーキングセット・サイズを大幅に削減します。結果として、Task Managerによって表示されるプロセスのワーキングセット・サイズは激しく変動する可能性があります。従、Task Managerにより報告されるワーキングセット・サイズや“VM Size”値は、サーバーのメモリー使用量を決定するときに使用する有効な測定基準ではありません(注15)。代わりに、Windows Performance Monitorの“Process/Private Bytes”および“Process/Virtual Bytes”カウンターに報告されている値を、それぞれサーバーのメモリー消費量およびアドレス空間サイズ(注16)をトラックキングするために活用することを勧めます。そしてすべてのプラットフォームにおいて、起動時にキャッシュサイズを確認するためにサーバー画面を確認したり、現在のキャッシュサイズを取得するためにSelect property ('CacheSize')を行います。

警告：Windows NT Performance Monitorを使用することで非常に有益な情報を得られますが、同時にその使用によりパフォーマンス評価の結果を混乱させる可能性もあります。なぜなら、NT Performance Monitorもある程度のサーバーリソースを消費するからです。さらに、背景知識なしにPerformance Monitor統計データを適切には解釈するのは難しい可能性があります。

市販およびフリーウェアのパフォーマンス評価ツールに関する総合的な調査については、本ドキュメントの範囲外です。

結論

コンピューターシステムのパフォーマンス評価には膨大な専門的知識が必要になり困難が伴います。評価に使用するワークロードモデルが正当かどうかを立証し、実態を再現できていることを保証してくれる絶対的な技術など存在しません[14]。また、パフォーマンス評価実験でパフォーマンスのボトルネックが存在している可能性が浮かび上がった場合はそのボトルネックがどこにあるのか、具体的にどのようなものなのか、そしてその問題に対処するためにどうしたらいいのかを特定するのに膨大な時間と労力を費やすことになります。

本文書では、ベンチマーク用のワークロード作成に関連する問題を考察し、2^k 実験計画に基づいた解析方法を述べてきました。一部実施要因計画を具体的に示す2^{k-p} 計画などの他の実験計画も可能です。2^{k-p} 計画では、必要な実験のトータル回数を減らすことができます。ただしこの場合、結果が*交絡する*リスクがあり、交互作用を確認するのに必要十分な実験結果詳細がないため、要因グループ間の交互作用が十分に示されません。

上記以外の状況下では*ネスト化された実験計画*または*ブロック計画*を用いた解析を行う価値があるかもしれません。たとえば要因の一つがハードウェアプラットフォームおよび（または）OSにある場合はブロック計画を行うのが最適です。OSの特性であるメモリー割り当てや、基礎をなすアーキテクチャー（32ビットかそれとも64ビットか）などを含めた総効果により実験結果に有意な分散が生じるため、このような場合にブロック計画を行えばモデルの正確さを飛躍的に向上させることができるかもしれません。

2^{k-p} 実験計画、およびさらに複雑なその他の実験計画の解説・解析については本書では割愛します。詳細に関しては参考文献[20, 22]をご覧ください。

注

注1. 全レベルのデータスキューが規定されるのはTPC-DSベンチマーク案[29]のみであり、その他の業界標準ベンチマークテストにおいてデータベースデータは一様分布で生成されますのでご注意ください。

注2. この作業をどれほど簡単な方法で行ったとしてもそれがいかに大変な作業になるかは参考資料[9]の9章をご覧頂ければお分かりになると思います。

注3. SQL Anywhereはダイナミック・キャッシュサイジングをサポートしています。この機能により必要に応じてサーバーのバッファープール・サイズが自動で調整され、測定実験中に制御される他のワークロードパラメーターが付加されます。

注4. SQL Anywhere 9.x、10.xのドキュメントで記述されているソフトおよびハードメモリー制御リミット情報は誤りです。

注5. キャッシュの自動リサイズが-ca 0コマンドライン・スイッチにより無効となっている場合、ハードリミットがその時点のバッファープール・サイズを上回ることがあります。

注6. 32ビットWindows Advanced Serverプラットフォームの場合、データベースサーバー・プロセス（Advanced Windowing Extensionsなし）で使用可能な最大アドレス空間はさらに約1ギガバイト（合計2.6ギガバイト）増えます。

注7. 要求レベルのログではアプリケーションサーバーのアクティビティーがキャプチャーされますが、アプリケーション要求の結果ストアドプロシージャーまたは当該機能内で発生した一連のSQL要求はキャプチャーされませんのでご注意ください。ただしSQL Anywhereバージョン10で提供されるアプリケーションプロファイリング機能ではこの機能が提供されます。アプリケーションプロファイリングおよび要求レベルログどちらに関しても、パフォーマンス評価に直接使用するのにふさわしい形では出力されないため、スクリプト内で若干の後処理が必要になります。

注8. SQL Anywhere 10ではクエリー間の平行性もサポートし、より複雑な拡張性分析を実現しています。一つの要求を算出するために、一つまたはそれ以上のプロセッサーを同時に使用します。

注9. この例では、デュアルプロセッサーPentium XEON 2.2GHz上のSQL Anywhere 8.0.2 (4294)で、データをフルキャッシュした状態でパフォーマンス結果を取得しています。またクエリーは、FETCHTSTユーティリティーを使用して計測しています。厳密には、値はシステムによって異なります。

注10. 2台のマシン間のネットワーク待ち時間は、SQL AnywhereのPINGユーティリティーが報告する往復時間を利用して概算できます。この往復時間は、送信側から受信側へデータパケット一つを送受する時間に、受信側から送信側へアクノリッジメントを返す時間を足した時間となります。

注11. ネットワークスループットを測定する一つの方法として、比較的大きなサイズのファイルを1台のマシンからもう1台のマシンへコピーし、その所要時間を測定するやり方があります。

注12. SQL Anywhereは、キャッシュサイズをサーバー要件と他のアプリケーションのニーズに合わせて変化させる、動的バッファープール・サイジングをサポートしています。この自己管理機能は組み込みアプリケーション環境では必須であることが多いが、バッファープールの可変性がパフォーマ

ンス分析（特に実験の再現性）を困難にする可能性があります。従、パフォーマンス分析には固定サイズのバッファープールを推奨します。

注13. Microsoft Excelをご使用の場合、この列掛け算はSumProduct機能で行えます。

注14. k 個の要因では、k > 2 の場合、散布図は多次元になります。

注15. Windows AWE拡張を使用して、データベース・キャッシュページはメモリーに物理的に固定されておりスワップアウトは不可能です。従、データベース・キャッシュページはプロセスのワーキングセットの一部として認識されず、タスクマネージャーによって報告されません。実際に、内蔵されているWindowsユーティリティーは、Address Windowing Extensionsを使用するプロセスのメモリー使用量を適切に報告するために存在するわけではありません。

注16. これらの値はどちらもAddress Windowing Extensions用のメモリー容量を含みません。

参考文献

[1] Luiz André Barroso, Kourosh Gharachorloo, and Edouard Bugnion. Memory system characterization of commercial workloads. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 3–14, Barcelona, Spain, June 1998. IEEE Computer Society Press.

[2] Ramesh Bhashyam. TPC-D–the challenges, issues, and results. ACM SIGMOD *Record*, 25(4):89–93, December 1996.

[3] Dina Bitton, David J. DeWitt, and Carolyn Turbyfill. Benchmarking database systems: A systematic approach. In *Proceedings of the 9th International Conference on Very Large Data Bases*, pages 8–19, Florence, Italy, October 1983. VLDB Endowment.

[4] Haran Boral and David J. DeWitt. A methodology for database system performance evaluation. In ACM SIGMOD *International Conference on Management of Data*, pages 176–185, Boston, Massachusetts, June 1984. Association for Computing Machinery.

[5] Peter M. Chen and David A. Patterson. A new approach to I/O performance evaluation–self-scaling I/O benchmarks, predicted I/O performance. ACM *Transactions on Computer Systems*, 12(4):308–339, November 1994.

[6] Stavros Christodoulakis. Implications of certain assumptions in database performance evaluation. ACM *Transactions on Database Systems*, 9(2):163–186, 1984.

[7] Xilin Cui, Patrick Martin, and Wendy Powley. A study of capacity planning for database management systems with OLAP workloads. In *Proceedings of the International CMG Conference*, pages 515–526, Dallas, Texas, December 2003. Computer Measurement Group.

[8] Steven A. Demurjian, David K. Hsiao, Douglas S. Kerr, Robert C. Tekampe, and Robert J. Watson. Performance measurement methodologies for database systems. In *Proceedings of the 13th ACM Annual Conference*, pages 16–28, Denver, Colorado, October 1985. Association for Computing Machinery.

[9] Steven A. Demurjian, David K. Hsiao, and Roger G. Marshall. *Design Analysis and Performance Evaluation Methodologies for Database Computers*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[10] David J. Dewitt, Shahram Ghandeharizadeh, and Donovan Schneider. A performance analysis of the gamma database machine. In *ACM SIGMOD International Conference on Management of Data*, pages 350–360, Chicago, Illinois, June 1988.

[11] Dror G. Feitelson. Workload modeling for performance evaluation. In Maria Carla Calzarossa and Salvatore Tucci, editors, *Performance Evaluation of Complex Systems: Techniques and Tools*, Lecture Notes in Computer Science 2459, pages 114–141. Springer-Verlag, Rome, Italy, September 2002.

[12] Domenico Ferrari. *Computer Systems Performance Evaluation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[13] Domenico Ferrari, Giusepe Serazzi, and Alessandro Zeigner. *Measurement and Tuning of Computer Systems*. Prentice-Hall, Englewood Cliffs, New Jersey,1983.

[14] Paul J. Fortier and Howard E. Michel. *Computer Systems Performance Evaluation and Prediction*. Digital Press, Burlington, Massachusetts,2003.

[15] G. Benton Gibbs, Jerry M. Enriquez, Nigel Griffiths, Corneliu Holban, Eunyoung Ko, and Yohichi Kurasawa. IBM *E-server pSeries Sizing and Capacity Planning: A Practical Guide*. IBM Corporation, San Jose, California, March 2004. IBM Redbook Order Number SG–247071, http://www.redbooks.ibm.com/redbooks/pdfs/sg247071.pdf 参照

[16] Jim Gray, editor. *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan-Kaufmann, San Francisco, California, second edition, 1993.

[17] Seungrahn Hahn, M.H. Ann Jackson, Bruce Kabath, Ashraf Kamel, Caryn Meyers, Ana Rivera Matias, Merrilee Osterhoudt, and Gary Robinson. *Capacity Planning for Business Intelligence Applications: Approaches and Methodologies*. IBM Corporation, San Jose, California, November 2000. IBM Redbook Order Number SG24–5689,

http://www.redbooks.ibm.com/redbooks/pdfs/sg245689.pdf 参照
[18] Richard A. Hankins, Trung A. Diep, Murali Annavaram, Brian Hirano, Harald Eri, Hubert Nueckel, and John P. Shen. Scaling and characterizing database workloads: Bridging the gap between research and practice. In *Proceedings of the 36th International Symposium on Microarchitecture*. IEEE Computer Society Press, December 2003.
[19] Paula B. Hawthorn and Michael Stonebraker. Performance analysis of a relational data base management system. In *ACM SIGMOD International Conference on Management of Data*, pages 1–12, Boston, Massachusetts, May 1979. Association for Computing Machinery.
[20] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, New York, New York, 1991.
[21] K. [Krishna] Kant. *Introduction to Computer System Performance Evaluation*. McGraw-Hill, New York, New York, 1992.
[22] Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models*. McGraw-Hill International, New York, New York, fifth edition, 2005.
[23] Jack L. Lo, Luiz André Barroso, Susan J. Eggers, Kourosh Gharachorloo, Henry M. Levy, and Sujay S. Parekh. An analysis of database workload performance on simultaneous multithreaded processors. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 39–50, Barcelona, Spain, June 1998. IEEE Computer Society Press.
[24] Winfried Materna. An approach to the construction of workload models. In M. Arató, A . Butrimenko, and E. Gelenbe, editors, *Performance of Computer Systems*, pages 161–177. North-Holland, Vienna, Austria, February 1979.
[25] Ann Marie Grizzaffi Maynard, Colette M. Donnelly, and Bret R. Olszewski. Contrasting characteristics and cache performance of technical and multi-user commercial workloads. In *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 145–156, San Jose, California, October 1994. Association for Computing Machinery.
[26] Ian McHardy. Optimizing Adaptive Server Anywhere performance over a WAN. Technical whitepaper, Sybase iAnywhere, Waterloo, Ontario, 2005. http://www.ianywhere.com/downloads/whitepapers/wan tuning.pdf 参照
[27] Daniel A. Menascé and Virgilio A. F. Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice-Hall, Upper Saddle River, New Jersey, 2001.
[28] David S. Moore and George P. McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman and Company, New York, New York,1989.
[29] Meikel Poess, Bryan Smith, Lubor Kollar, and Paul [Per-Åke] Larson. TPC-DS, Taking decision support benchmarking to the next level. In *ACM SIGMOD International Conference on Management of Data*, pages 582–587, Madison, Wisconsin, June 2002. Association for Computing Machinery.
[30] Viswanath Poosala. Zipf's law. Technical report, University of Wisconsin, Madison, Wisconsin, 1995. http://www.bell-labs.com/user/poosala/pub.html 参照
[31] Parthasarathy Ranganathan, Kourosh Gharachorloo, Sarita V. Adve, and Luiz André Barroso. Performance of database workloads on shared-memory systems with out-of-order processors. In *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 307–318, San Jose, California, October 1998. Association for Computing Machinery.
[32] Tom Sawyer. Doing your own benchmark. In Gray [16], pages 543–561.
[33] Dennis Shasha and Philippe Bonnet. *Database Tuning*. Morgan-Kaufmann, San Francisco, California, 2003.
[34] Standard Performance Evaluation Corporation, Warrenton, Virginia. SPEC *jAppServer2004 Benchmark Specification, Revision 1.08*, December 2006. http://www.spec.org/jAppServer2004 参照.

[35] G. C. Steindel and H. G. Madison. A benchmark comparison of DB2 and the DBC/1012. In *Proceedings, International Conference on Management and Performance Evaluation of Computer Systems*, pages 360–369, Orlando, Florida,1987. The Computer Measurement Group.
[36] Liba Svobodova. *Computer Performance Measurement and Evaluation Methods: Analysis and Applications*. Elsevier, New York, New York, 1976.
[37] Transaction Processing Performance Council, San Jose, California. *TPC Benchmark D (Decision Support) Standard Specification, Revision 2.0.0.12*, June 1998.
[38] Transaction Processing Performance Council, San Jose, California. *TPC Benchmark H (Decision Support) Standard Specification, Revision 1.1.0*, June 1999.
[39] Transaction Processing Performance Council, San Jose, California. *TPC Benchmark R (Decision Support) Standard Specification, Revision 1.0.1*, February 1999.
[40] Ted J. Wasserman, Patrick Martin, and Haider Rizvi. Developing a characterization of business intelligence workloads for sizing new database systems. In *Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP*, pages 7–13, Wasington,D.C., November 2004. Association for Computing Machinery.
[41] Ted J. Wasserman, Patrick Martin, and Haider Rizvi. Sizing DB2 UDB servers for business intelligence workloads. In *Proceedings of the 2004 Conference of the IBM Centre for Advanced Studies on Collaborative Research*, pages 135–149, Markham, Ontario, October 2004. IBM Corporation.

法的注意

アイエニウェア・ソリューションズ株式会社
http://www.ianywhere.jp/