システムソフトウェアと
オペレーティング・システム　89－6
シ　ス　テ　ム　評　価　2－6
（２００２．２．１４）

Web サイトのアーキテクチャの再考とそれに伴うサイジングについて

長谷川 泰章[1)]　熊澤 公平[1)]　鈴木 勝典[2)]
[1)]株式会社リクルート　[2)]株式会社シーマーク

Web サイトの構築において，システムアーキテクチャの設計とサイジングはその後の運用・保守に与える影響が大きいだけに慎重に取り組まなければならない．本報告ではリクルートの商用 Web サイト「イサイズ・ドット・コム」が抱える種々の課題を解決するための取り組みの中で，上記の 2 点に関する事例を紹介する．1 つは旧来のアーキテクチャが持つ欠点を解決するために設計した新しいアーキテクチャについて説明する．もう 1 つはアーキテクチャの見直しに伴って考案した簡易サイジング方法について説明する．

Examples of Reconsideration of Web Site Architecture and A Simple Sizing Method

Hiroaki Hasegawa[1)]　Kohei Kumazawa[1)]　Katsunori Suzuki[2)]
[1)]RECRUIT Co.,Ltd.　[2)]SEAMARK Co.,Ltd.

Designing system architecture and sizing are very important processes on every development of web site. This report shows two examples that we have tried to overcome many problems of our commercial web site "www.isize.com". One is a new architecture that we have designed in order to resolve a lot of weak points that an old architecture has. Another is a new simple and easy method for sizing of web site.

1. はじめに

1.1. 背景

Web サイトの構築には技術的にさまざまな難しさがある．その中でもシステムアーキテクチャの設計とサイジングは，その後の運用・保守に与える影響が大きいだけに慎重に取り組まなければならない．

本報告で取り上げているイサイズ・ドット・コム（http://www.isize.com/，以降 ISIZE と略称）は，リクルートが 1999 年 1 月にサービスを開始した大規模商用 Web サイトである．現在までの約 3 年間に増えつづける利用者数に対応するため，繰り返しハードウェアやネットワークの増強，あるいはアプリケーションやデータベース（以降 DB と略称）の見直しなどを行ってきた．しかしこれはもともと 1995 年ごろに，小さな UNIX サーバ 1 台で運用を開始した MixJuice という実験的 Web サイトを起源としており，基本的なシステムアーキテクチャはその当時からほとんど変っていない．そのため規模が年々大きくなるにつれて，スケーラビリティの不足，リソースの無駄使い，システム変更に対する柔軟性のなさなど，古いアーキテクチャの限界が顕在化してきた．

1.2. 目的と目標

検討の目的は前述の課題を解決することである．そのために現在のアーキテクチャを抜本的に見直し，Web サイトの再設計と移行方法の検討を行うことにした．具体的な目標は次のとおりである．

(1) 低コスト，短期間で効率的に構築でき，性能とリソース消費のバランスに優れたシステムアーキテクチャとは何かを明らかにする．

(2) 現行のシステムを移行する上で不可欠なサイジング設計をできるだけ容易に実施できる方法を考案する．

2. Web サイト構築上の課題

2.1. システムアーキテクチャに関する課題

ISIZE は窓口となる 1 つのポータルページと，その配下に仕事，旅行，住まい，クルマなどシーンと呼ばれる各事業ごとのシステムが 20 前後存在するという構成になっている（図 1）．

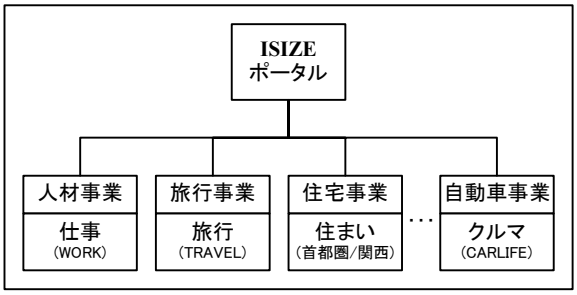

図 1. ISIZE サイト構成

共通ルールやライブラリなどは用意してあるが，各事業ごとのシステムは別々に構築されている．図 2は現在の ISIZE のごく基本的なサーバ構成である．多少のバリエーションはあるものの，第 1 階層にあたる複数 Web サーバ，及び第 2 階層にあたる複数 DB サーバ

／複数 NFS サーバの 2 階層型の構造になっている．静的ページは NFS サーバに格納されており，Web サーバからアクセスされる．動的ページは基本的にはすべて CGI をベースにしており，Web サーバ上の CGI プログラムが DB サーバに対して SQL を発行する．

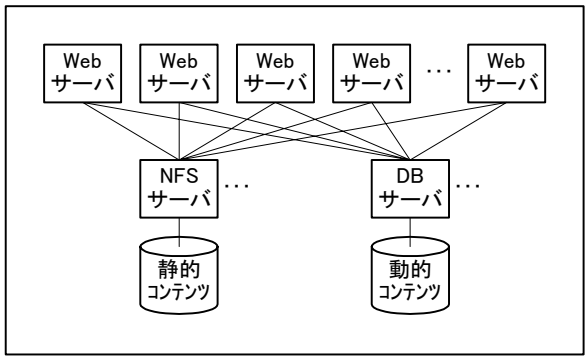

図 2. 基本的サーバ構成

動的ページにおけるプログラム言語はほとんどが RPerl[1]で，一部 C/C++も利用している．Perl と CGI は多くの Web サイトで用いられており，今ではもっとも枯れた技術の 1 つで安心感がある．しかし基本的にプロセス生成型の実行環境であるため，多くのシステムリソースが必要な上，性能が出にくい，本質的にスケーラビリティを確保しにくいなどの欠点がある．RPerl は長年利用してきて十分に慣れていること，インタプリタ言語特有の扱いやすさやプログラマ層の厚さなどから，開発や保守の生産性はかなり高くなっているが，前述のような欠点がある．

ISIZE のアーキテクチャ上のもう 1 つの特徴として，動的ページが多いことがあげられる．つまりコンテンツの多くを DB に格納し，CGI 経由でアクセスして動的に HTML を生成して表示する形式になっている．中には 8 割以上が動的ページというシーンも存在する．動的ページは共通ライブラリなどを利用すれば手軽に Web ページの開発ができる，DB 内のコンテンツの一元管理がしやすい，セキュリティを確保しやすいなどのメリットがある．しかし CGI の割合が大きいことは前述のとおりの欠点がある上，多くのコンテンツを DB に格納するため，DB サーバの負荷が必要以上に高くなり，性能上のデメリットが大きい．

また動的ページの中でも DB 更新処理が多いサービスや，ログイン ID 等の認証機能や個人情報保護機能などのセキュア環境を必須とするようなサービスもある．そのためアーキテクチャが複雑になっている．加えて度重なる追加開発で規模が大きくなってきたため，共用している DB サーバを中心として，シーン間の相互依存性が高くなり，保守のためのサービス停止の調整などに余計な時間がかかる上，実際の保守作業もたいへんな手間がかかるようになっている．

[1] Recruit Perl：リクルートが独自にカスタマイズした perl 言語

2000 年までのインターネットブームの中では，迅速な Web サイト構築がビジネス上もっとも重要であった．そのため CGI の利用や動的コンテンツの多さによる欠点は，あり余るほどのシステムリソースへの投資のおかげで，それほど問題にならなかった．しかしブームが去った現在では，コストの最適化と迅速性の両方を満たすアーキテクチャを模索する必要がある．

2.2. サイジング方法に関する課題

初期の ISIZE の構築においては，システム構築担当者が，予想されるアクセス数，コンテンツの量やサイズ，プログラムのステップ数や本数，製品の性能などと，これまでの経験値に基づいて，ハードウェアやネットワークのサイジングを行った．しかしこれはある意味勘に近いものがあり，システマティックなサイジングは行っていない．またかなり規模が大きくなってきた現在でも，ほとんどの構築現場では明確なサイジング方法を確立していない．それでもこれまで何とか運用できているのは，初期の段階で余裕を持ったハードウェアとネットワークの構成にしていたことと，性能悪化やリソース不足が発生するたびに十二分な投資を行い，それらを増強してきたからに他ならない．しかしコストの最適化を強く求められている現状では，そのような方法は採用できない．したがって最初の構築の段階でできるだけ正確なサイジングを行う必要がある．もちろん精度が高くてもコストが高くつくのでは意味がないので，バランスのとれた方法を考案すべきである．

3. アーキテクチャの再考

3.1. 検討の方向性

検討の方向性は，コストの最適化と迅速性の両方を満たすアーキテクチャを模索することである．基本的には現在のアーキテクチャが持つ欠点をほとんど解消できるものを採用すべきであると考える．具体的には下記のような方向性で検討を進めた．

(1) 低コストで構築が可能
(2) システムリソースの消費が少ない
(3) パフォーマンスがよい（特に高負荷時）
(4) 本質的にスケーラビリティに富む
(5) 業界標準的な技術を採用する
(6) DB サーバの負荷を下げる
(7) シーン個別の機能はできるだけ少なくする
(8) 開発生産性を向上させやすい
(9) 柔軟性が高い（変更に強い）

3.2. 新しいアーキテクチャ

検討の結果決定した新しいアーキテクチャの概要について述べる．

3.2.1. サービス構成とインフラに関する考え方

新アーキテクチャに基づいた全体サービス構成の概

要は図 3のとおりである.

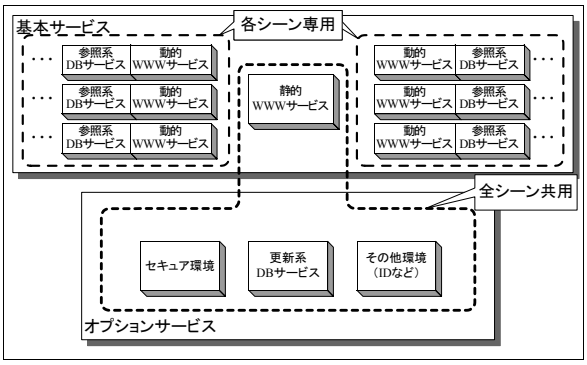

図 3. 全体サービス構成の概要

(1) **サービスを基本とオプションの 2 つに分ける**: シーンごとに異なる要件をサービスの組合せで実現でき，負担コストも明確にしやすい.

(2) **静的と動的ページのサービスを分割する**: 特性の大きく異なるコンテンツの性能をそれぞれ最大限に発揮させるため.

(3) **静的 WWW サービスは全シーンで共用する**: 静的ページは比較的性能の低いハードウェアでも十分な性能を発揮させられる. 分けるとサーバ台数が増えて運用コスト的に不利である.

(4) **動的 WWW サービスはシーンごとに用意する**: 動的ページのアプリケーションやコンテンツはシーンごとに特性がかなり異なるので, 分けたほうが開発・保守上管理がしやすい.

(5) **更新系 DB や高いセキュリティをあまり必要としない**: 更新系 DB やセキュリティ機能はコストがかかる．できるだけ ID や個人情報を必要としない構成・コンテンツなどにしたい.

(6) **静的コンテンツは静的 Web サーバのローカルディスクに格納する**: これまでは NFS サーバを利用していたが, 現行のコンテンツ量と最近のサーバスペックから見て, ローカルディスクで十分な容量と性能を確保できる.

(7) **参照系 DB のリカバリに時間がかかってもよい**: 短時間でリカバリできる構成にするとコスト高になるので，ある程度のレベルに抑えたい.

3.2.2. アプリケーションアーキテクチャ

基本的なアプリケーションアーキテクチャの概要は図 4のとおりである.

(1) **バッチ生成した静的ページで 85%のページビュー(以降 PV)をカバーする**: 静的 HTML は処理性能を高めやすいので, 全体の費用対効果を向上させるためにも，できるだけ比率を高くしたい.

(2) **動的ページはサーバサイド Java 技術(サーブレット/JSP)で開発する**: Perl や C による CGI と比較して高負荷時の処理性能が高くスケーラビリティに富む．業界標準でもある．EJB はまだ開発生産性を高められる状況ではない上, 性能容量的にも不利になる場合が多いので当初は採用しない．将来改善されれば採用もあり得る.

(3) **動的ページの 90%は参照系 DB(MySQL)を使用する**: 高い比率でフリーソフトを使うことでコストを下げられる．参照系のみなら Oracle よりも処理性能が高い場合が多い.

(4) **Java 開発フレームワークの採用**: 開発・保守の生産性を向上させるため，シーマーク社の J2EE フレームワークである logicletBox[3]をリファレンスとして採用している.

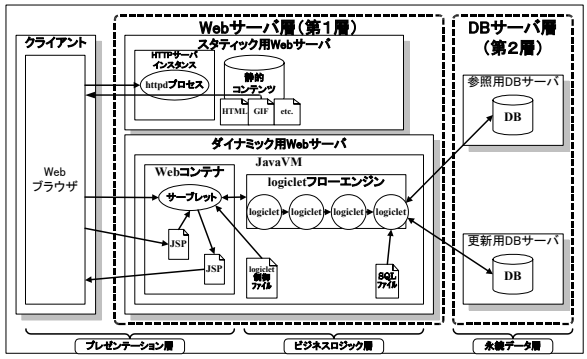

図 4. アプリケーションアーキテクチャの概要

3.3. 制約事項

新アーキテクチャは従来の欠点をほぼ解消できるかわりに，ある程度の制約がある．その代表的なものは次のとおりである.

(1) **DB 検索パターンなどの自由度が低くなる**: DB 検索は原則として, あらかじめバッチジョブで作り置きした静的ページを配信する形式である. そのためレスポンスタイムは格段に短くなるが, 多様な検索パターンには対応できない.

(2) **完全な即時 DB 更新ができなくなる**: ユーザからの DB 更新要求は何らかの仕組みでいったんキューイングされ, その後あるタイミングでまとめてバッチ更新する形式になる. 原稿入稿や掲示板の書き込みなどがすぐには反映されなくなるので, 場合によってはユーザの心理的負担が大きくなる可能性がある.

これらの制約は難しい問題であり，ビジネス上必須かどうかの判断が必要になる．場合によっては新しいアーキテクチャへの移行が不可能な場合もあり得る.

4. サイジング方法の検討

4.1. 検討の前提

サイジング方法の検討の前提は次のとおり.

(1) 低コストで容易であること
(2) 現行 Web システムの移行を前提とする
(3) Web サーバのサイジングに焦点を絞る
(4) 性能設計より必要容量見積りを重視する

次の4.2以降では今回考案したサイジング方法の基本的な考え方と手順について述べる．新アーキテクチャのサービス構成に基づき，静的と動的 Web サーバのサイジングは原則として分けて考える．

4.2. Web サイトの分析

移行対象 Web サイトを URL 単位で分析する．つまり静的・動的ページのそれぞれを，サイト構造とアクセスログから特徴付ける．具体的には静的ページや CGI 単位で分類して，単位時間当りの PV 数，各種データ量，SQL 等 DB 関係のデータやアプリケーションなどを調査する．転送データ量などは実ページサイズだけでなくアクセスログも参考にする．最終的に PV/sec，PV 率，累積 PV 率，HTML 平均サイズなどで整理して一覧表にする．今回のサイジングの焦点は負荷のピーク時の容量設計なので, PV などのデータはピーク時間帯のものを中心に分析する．

4.3. モデル素の考え方

各 URL が，「モデル素」と呼ぶ, 1 つまたは複数の，これ以上分解できない基本的要素の組合せで構成されていると考える．表 1は今回の題材の ISIZE スポーツを分析した結果である．BN，E，Sc は他のモデル素から導出されたものなので, 本質的なものは, A, As, B, BA，BP，C，D，S の 8 つである．

表 1. モデル素一覧（ISIZE スポーツ）

ﾓﾃﾞﾙ素	説明
A	DB **無**，**セッション無**，**整数演算処理**：2KB HTML，フィボナッチ数列を計算，10 から 20 項目までと 92 項目の値を表示．現在時刻を出力.
As	DB **無**，**セッション有**，**セッション情報の読み書き処理**：1KB HTML，セッションを保持し，数値をインクリメントし，その値を出力する.
B	DB **有(軽量参照)**，**セッション無**，**単一コネクション**：1KB HTML，単独テーブルに対し 1 件 select で出力.
BA	DB **有**，**セッション無**，**比較的大きな連想配列処理**：2KB HTML，DB から Select したデータを要素 2000 個の連想配列に保持，10 個のキーで要素を抽出，ページを出力.
BN	DB **有(軽量参照)**，**セッション無**，**コネクション複数回実行**:2KB HTML, B の処理を 10 回繰り返す. Select するデータは 1 回毎に別．導出モデル素(B).
BP	DB **有(軽量参照)**，**セッション無**）：2KB HTML，単独テーブルに対し 20 件 select，レコード内容を出力.
C	DB **有(重い参照)**，**セッション無**：10KB HTML，3 テーブルから JOIN した select をかけ，10 件を表示.
D	DB **有(更新)**，**セッション無**：1KB HTML，単独テーブルに対する update を 1 回行う.
E	DB **有(参照＆更新)**，**セッション無**：1KB HTML，2 種類の単独テーブルを 1 回ずつ select，別の単独テーブルに update を 1 回行う．導出モデル素(B,BP,D).
S	**静的ページ配信**，**サーバサイドキャッシュ無**：4KB HTML，同一静的ファイルの取得.
Sc	**静的ページ配信**，**サーバサイドキャッシュ有**：4KB HTML，複数静的ファイルの取得．導出モデル素.

4.4. ベンチマークテストプログラムの作成

Web サイトの分析によって得られた各モデル素に対応するベンチマークテストプログラムを作成する．基本的にはモデル素の処理内容と特性を，新しいアーキテクチャに適合させて実装する.

4.5. 静的 Web サーバのサイジング

結論から述べると，動的 Web サーバのように詳細な検討を行っていない．というのもベンチマークテストの結果, 静的 Web サーバの性能は, 現在の代表的 UNIX サーバの中でかなり性能の低いものを利用しても軽く 2000 リクエスト／秒以上の性能が出ることがわかったので，あまり精密に行う意味がないからである．この値はほとんど 100M イーサネットの性能限界で，これ以上は向上させようがない.

また参考として 1G イーサネットを利用し，かつ OS のサーバサイド･キャッシュ機能を利用した場合, 7000～12000 リクエスト／秒程度まで性能を高められることがわかっている．しかしインターネット接続部分が 1Gbps というのは当面あり得ないので，この結果はサイジング設計上は考慮しない.

4.6. 動的 Web サーバのサイジング

ここからは動的 Web サーバのサイジングに関する考え方と手順について述べる.

4.6.1. アプリケーションの分析と特徴付け

各 CGI に関して 2 段階の特徴付けを行う．1 つめは 1 次近似で，1 つの CGI が 1 つのモデル素と対応する場合である．2 つめは 2 次近似で，CGI が複数のモデル素の組合せと対応する場合である．調査も含めてごく簡単にサイジングを行う場合は 1 次近似のみを用い, もう少し詳細に行いたい場合は 2 次近似を用いる．当然 1 次近似のほうが精度は低くなる.

1 次近似の判別方法は, 例えば DB 処理がなければ A または As．そのうち Cookie 等のセッション管理を使用していなければ A，使用していれば As となる．2 次近似では，例えば軽い select が 2 回なら「B×2」，重い select が 4 回なら「C×4」. 軽い select が 2 回で update が 1 回なら「B×2＋D×1」となる.

4.6.2. ベンチマークテスト

サイジング計算のための基礎データを得るためベンチマークテストを行う．テストでは基本的に各モデル素プログラム単独の処理性能を測定する．ターゲットとなる運用環境を考慮してリファレンスマシンと構成を決定すべきである．測定時は処理エラーが発生しない範囲で，サーバマシンの CPU 使用率を安定的に 100%状態にできる程度の負荷をかける．今回は最大負荷時の容量設計が焦点なのでこのような方法を取っている．つまり外部負荷の変動に対する CPU 負荷や処理性能の変動を詳しく調べるのではなく，リファレンスマシンの最大能力での測定を行うことにより，測定値の正規化を行っていると見なしている．この状態で各プログラムの最高性能値をサイジングに使用する．ま

たWebサーバのCPU使用率の内訳（sysとuser）は考慮しない. 両者合わせて100%になっているようにする. つまりサーバ内部における CPU やプロセスのスイッチなどシステムのオーバーヘッドは無視する．最終的な計算を容易にするため, Webサーバの CPU 使用率は, どのモデル素の負荷変動に対しても直線的に比例すると仮定する.

システム内部での処理時間は表 2の 5 つから構成されると仮定する．これらの基礎データの設定には，ベンチマークテストで得られた測定値と，それを基にした推定値を用いる．測定上注意すべきことは，DB サーバが処理全体のボトルネックにならないよう，余裕を持って処理を行えるような構成にすることである．具体的には Web サーバの CPU が 100%状態でも，DB サーバの CPU に余裕があるようにする.

表 2. 処理時間の構成（シングルプロファイル）

No	構成要素	説明
①	pre	平均待ち時間．クライアントからのリクエストが Web サーバに到着してから実際に処理が行われるまでの待ち時間．本来は詳細に測定するか，待ち行列理論を用いて推定するべきだが，サーバ内全処理時間の中で占める割合が低いので，今回は Web サーバ付属のツールで計測した値の平均値を用いた.
②	process+jsp	①③④⑤を除く Web サーバ内部（Web コンテナを含む）での処理時間．これも本来はプロファイラ等で精密に測定できると，より精度が上がるのだが，手間がかかるので総処理時間から他の処理時間を差し引いた値を推定値として使用している．JSP の処理は実際は各種の処理が行われた後に1回だけ実行されるが，ここでは①③④⑤以外の処理にすべてまとめてしまっている．したがってサイジングの2次近似の計算では複数回評価されることになってしまうが今回は無視している．しかし将来はJSPの処理時間を分けてプロファイルを設定する方向で考えている.
③	jndi lookup	DB アクセス時にデータアクセスオブジェクトを探すために必要な処理時間．モデル素 B においてログを挿入し計測した結果.
④	db	DB 系の処理時間はそれぞれのモデル素のクエリを実行し，フェッチし終わるまでの DB 関係の処理に要した時間をログ挿入し計測した．DB サーバ上で処理中は Web サーバは結果待ちのアイドル状態にあるが, Web サーバ内では並列に処理が実行されているので完全な待ち状態になるわけではない.
⑤	transfer	Web サーバが返信用 HTML を構成した後，サーバからすべての HTML を出力するまでの時間．本来はサーバ内処理時間のみを考慮すべきだが，計測が難しいことやベンチマーク環境はLANのみで遅延が少ないことから，今回はネットワーク時間も含んでいる.

4.6.3. サイジング計算

テスト環境の待ち行列ネットワークモデルは実際には図 5上図のようになる．しかし今回は図 5下図のように簡略化して考える．すなわち到着分布も処理分布も指数分布でサーバ台数が 1 台の基本的なモデル（M/M/1/∞/∞/FCFS）として扱う.

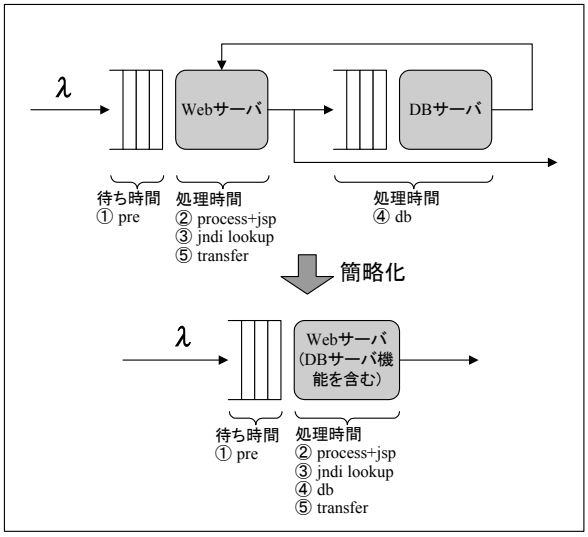

図 5. 計算のためのモデル

①～⑤の各処理時間は実際には図 5上図の状態を表している．しかし計算上は下図のように DB サーバの処理時間（④db）を Web サーバに含めて 1 つの処理装置のように単純化するわけである.

計算の基本は図 6のとおり．具体的にはλはその CGI に対する単位時間当りのリクエスト数，μは処理装置の平均処理能力を表す．ρは処理装置が空いていない確率，すなわち処理装置の利用率（%）を表す.

各 URL の $\rho = \frac{\lambda}{\mu}$ から全 URL の Σρ を算出
λ：平均到着率（req/sec）
μ：平均サービス率（req/sec）
ρ：処理装置の利用率（%）
Σ：全 URL 分の総和

図 6. 基本的な計算方法

λは各 CGI に対する毎秒当りのリクエスト数から求める．μはシングルプロファイルの①～⑤から計算する．1 次及び 2 次近似における平均サービス率μの具体的な計算方法は図 7のとおり．基本的には 1 次または2次近似で得られた総処理時間の逆数に CPU 数をかけたものと定義している．｛M 素分解｝は，各要素が 1 次及び 2 次近似の対応モデル素の数を表す 1 行 9 列の行列．その他｛foo｝は，各モデル素のシングルプロファイルにおける処理時間を要素に持つ 9 行 1 列の行列である.

平均サービス率 μ（個/sec）

$$= \frac{1000 \times \text{CPU 数}}{①+\{\text{M 素分解}\} \times (\{②\}+\{③\}+\{④\}) + \text{転送率} \times \text{htmlsz}/1024}$$

{foo}	行列（{M 素分解}以外の単位はすべて msec）
CPU 数	Web サーバマシンの CPU 数（今回は 4 個）
htmlsz	HTML のサイズ（単位：byte）
転送率	1KB の HTML の転送処理時間（今回は 0.152msec/KB）

図 7. 平均サービス率の計算方法

μ にはリクエスト到着時の待ち時間①pre と DB サーバ上での処理時間④db が含まれている．したがって実際には本来の Web サーバ単体の性能ではなく，DB サーバを含んだ Web サーバのサイジングをしていることになる．この方法では，DB サーバ上での処理が軽く，全体から見た処理の比率も低い場合には，比較的精度の高いサイジングが可能である．しかし逆の場合は実体とかなり異なる結果になる可能性があるので注意が必要である．

ρ は正確には各種の待ち時間を含んだ ELAPS 処理時間に基づく処理装置の専有率であり，実 CPU 使用率ではない．しかしサーバの専有率から簡単に判断できるようにするために近似的に実 CPU 使用率と見なしている．また今回のようにサーバがマルチプロセッサの場合，厳密には M/M/m（M/M/S）モデルで計算すべきだが，ここでは M/M/1 モデルでの計算値に単純に CPU 数を乗じるという簡略化をしている．

次にプロトタイププログラムの測定値と，モデル素で表した CGI の同一シナリオでの計算値とを比較して補正係数を算出する．そして最終的に 1 次近似の場合は 1 つの CGI を 1 つのモデル素で代表させた場合の，2 次近似の場合も 1 つの CGI を複数のモデル素で代表させた場合の補正計算値を基に，各 CGI のサーバの利用率（≒CPU 使用率）を計算する．その計算値を全 CGI 分加算してサーバの総利用率（≒CPU の総使用率）を計算する．

5. 適用例

5.1. 適用対象 Web サイト

ISIZE スポーツというシーンを対象に適用を試みた．これはサッカー，プロ野球，メジャーリーグ，F1，ゴルフ，競馬などスポーツ関連の各種情報サイトで，次のような特徴がある．

(1) **負荷のピーク性が高い**：オリンピック情報サイトなどと同様，人気の高い競技の進行状況を，リアルタイムに知りたいユーザからのアクセスが特定の時間帯に極端に集中する傾向（バースト性）がある．特に国際競技で日本のチームや選手が出場している時間帯は顕著である．

(2) **情報の更新頻度は低い**：サッカーや野球など代表的な人気スポーツのスコアでも秒単位で変化するものではないため，得点時に数分から数 10 分間隔で更新されれば十分である．

(3) **ほとんどが静的ページである**：90%以上のコンテンツが，試合の状況変化時にバッチ生成された静的ページで構成されている．残り 10%は選手の成績情報や掲示板機能などで，リアルタイムの参照・更新が行われている．

5.2. Web サイトの分析とモデル素対応付け

分析はこのシーンのピーク時間帯の 1 つである平日の正午から 1 時間のデータを基にしている．表 3は各 CGI とモデル素との近似例である．

表 3. 1 次近似及び 2 次近似の適用例

No	URL	PV (pv/s)	html (byte)	select	update	insert	1 次近似	2 次近似				
								A	As	B	C	D
1	/team.html	0.37	25770	6			C				6	
2	/answer_check.cgi	0.27	7808	3			B			3		
3	/rnd_image2.cgi	0.25	2710				A	1				

5.3. ベンチマークテスト

5.3.1. テスト環境

ベンチマークテストのマシン環境は図 8のとおり．ネットワークは基本的に 100M イーサネット LAN だが，静的ページの最高性能を測定するために 1Gbps のものも併用した．

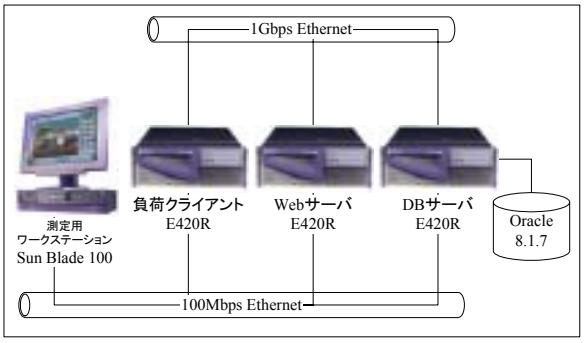

図 8. ベンチマークテストのマシン環境

ハードウェア環境は表 4のとおり．今回は実際の運用環境に近いことや，経験上コスト・パフォーマンスに優れ，新アーキテクチャのサーバ候補の 1 つでもあるマシンを選択した．

表 4. ベンチマークテストハードウェア環境

種別	型名	CPU	ﾒﾓﾘ	ﾈｯﾄﾜｰｸ	OS&ﾐﾄﾞﾙ
Web サーバ	Sun E420R	USparcII×4 (450MHz)	4GB	100M & 1G ｲｰｻﾈｯﾄ	Solaris8 iWS6
DB サーバ	↑	↑	↑	↑	Solaris8 Oracle8
ｸﾗｲｱﾝﾄ	↑	↑	↑	↑	Solaris8

ソフトウェア環境は表 5のとおり．HTTP サーバは数種類の中からベンチマークテストで総合的にもっとも性能の高かったものを，また DBMS は使い慣れてい

るものを選択した．

表 5. ベンチマークテストソフトウェア環境

種別	ベンダー	製品名	種別
Web サーバ	iPlanet E-Commerce Solutions	iPlanet Web Server 6.0E	HTTP サーバ
DB サーバ	Oracle	Oracle 8.1.7	RDBMS
ｸﾗｲｱﾝﾄ	Apache Software Foundation	Apache Benchmark Tool	Web アプリ用負荷ツール

5.3.2. 測定結果

表 6は測定結果に基づく基礎データ表である．

表 6. 基礎データ表

ﾓﾃﾞﾙ素	① pre	② procjsp	③ jndi	④ db	⑤ trans	total (ms)	html (byte)	実測値 (req/s)
A	0.6	3.96	0	0	0.29	4.85	1955	825.19
As	0.6	4.10	0	0	0.16	4.85	1046	824.37
B	0.6	3.23	6	5	0.16	14.99	1052	266.86
BA	0.6	361.51	6	430	0.29	798.40	1962	5.01
BN	0.6	24.44	30	50	0.28	105.32	1864	37.98
BP	0.6	262.66	6	405	0.28	674.54	1861	5.93
C	0.6	7.05	6	38	0.37	52.02	2480	76.89
D	0.6	14.81	6	6	0.21	27.62	1400	144.82
E	0.6	273.16	18	416	0.21	707.96	1400	5.65
S	0.6	0.29	0	0	0.60	1.49	4051	2682.14

5.4. プロトタイプによる補正

サイジング計算結果の補正を行うため，現行システムの中から PV 比率の高い CGI を数本抽出してプロトタイププログラムを作成しベンチマークテストを実施した．表 7は各プロトタイププログラムをモデル素で 2 次近似したものである．表 8はプロトタイプの測定結果と 2 次近似によるサイジング計算の結果から求めた補正係数である．

表 7. プロトタイプの 2 次近似

No	プロト	A	As	B	C
1	opta	3		4	2
2	quiz(1)	2	1	1	
3	quiz(2)	2		2	
4	quiz(3)	3	1	2	

表 8. 2 次近似用補正係数

No	プロト	① pre	② proc jsp	③ jndi	④ db	⑤ trans	total (ms)	html (byte)	計算値 (req/s)	実測値 (req/s)	補正係数
1	opta	0.6	38.91	36	96	3.81	175.33	25700	22.81	27.09	1.187
2	quiz(1)	0.6	15.24	6	5	1.16	28.00	7800	142.85	166.55	1.166
3	quiz(2)	0.6	14.38	12	10	1.16	38.14	7800	104.88	101.16	0.965
4	quiz(3)	0.6	22.43	12	10	1.16	46.19	7800	86.59	101.18	1.168

サイジング用補正係数⇒1.12

標本数は少ないが，全般的に計算値のほうが性能が低めになる傾向がある．標準偏差は 0.105 で比較的ばらつきは少ない．そこで計算値に平均補正係数 1.12 を乗じて最終的なサイジング計算値を求めることにした．この係数に関しては検定を行っていないが，今後さまざまなサイトの移行におけるサイジングを考える上では必要だと考える．

図 9は quiz(1)というプロトタイプに関して，表 6～表 8の各種データを基にサイジング計算（予測性能計算）を実施した例である．

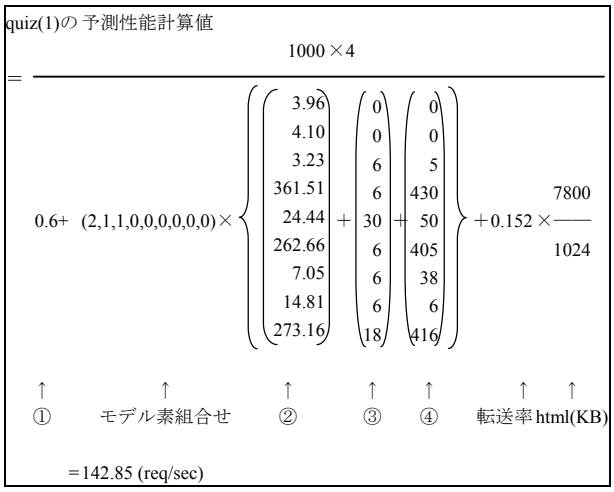

図 9. quiz(1)の 2 次近似の予測性能計算の具体例

5.5. サイジング計算

5.5.1. 計算結果

表 9は最終的なサイジング計算を行った結果である．169 種類の CGI アプリケーションに対して，それぞれ平均到着率 PV(pv/sec)でリクエストがあった場合の 1 次及び 2 次近似それぞれの Web サーバの利用率を計算し，それを全 CGI 分合計している．

表 9. 最終的なサイジング計算結果

No	CGI	PV (pv/s)	html (byte)	1 次近似 (req/s)	1 次近似 ρ (%)	2 次近似 (req/s)	2 次近似 ρ (%)
1	/team.html	0.37	25770	79.14	0.47	14.44	2.58
2	/answer_check.cgi	0.27	7808	274.26	0.10	100.91	0.27
3	/rnd_image2.cgi	0.25	2710	882.02	0.03	904.61	0.03
省略	←	←	←	←	←	←	←
11	/usa_thanks.cgi	0.08	5328	6.33	1.18	63.28	0.12
省略	←	←	←	←	←	←	←
169	/Aimage_edit.html	0.00	2996	872.44	0.00	896.93	0.00
	合計→	3.18		合計→	3.49	合計→	6.03

5.5.2. 考察

今回のサイジング設計に用いたサーバマシンの場合，ピーク時間帯の総負荷 3.18 リクエスト／秒に対して，1 次近似では 3.49%，2 次近似では 6.03%の使用率になる．実際の移行後のデータがあるわけではないので，現時点ではどの程度正確なのかは判断できない．しかしベンチマークテストとプロトタイピングの結果からすると，それほど的外れではないと思われる．なぜならもっとも処理負荷の高いモデル素 BA，BP，E の場合でも CPU 使用率を 100%にするには，表 6のように 5～6 リクエスト／秒の負荷が必要だが，全 CGI に対するピーク時の総負荷でもその 2 分の 1 強にしかならないし，実際に ISIZE スポーツにはこれらの重い処理は

ほとんどないからである．サイト分析での比率は A：26%，As：5%，B：38%，C：27%，D：5%で，重い処理は合わせても 1%未満である．これと各モデル素の最高性能値を考え合わせると 6.03%というのは妥当な値だと思われる．ただ逆にこの使用率は ELAPS 時間ベースであること，①pre と④db の処理時間も含めたものなので実際より多めに計算されていることから，危険側より安全側に振れすぎている可能性はある．

以上のことから ISIZE スポーツの動的ページを移行する場合，このサーバマシンでは負荷のピーク時でも 6～7%程度しか使用しないことになる．一般的に Web サイトの最大負荷時の CPU 使用率は平均して 70～80%程度はないと無駄なので，この程度のサイトであれば性能的に今回のサーバの 10 分の 1 程度で十分である．以上の考察から，実際の移行計画ではリファレンスマシンの 5 分の 1 程度の性能のサーバマシンを用意すればよいという判断をした．それ以上性能の低いサーバマシンは存在しないというのが事実である．

6. まとめ

現行の商用 Web サイトのアーキテクチャの古さに起因する種々の課題を解決するために，抜本的な見直しを行い，新しいアーキテクチャの方向性を明らかにした．これはこれまでの多くの課題を解決できるには違いないが，逆にあきらめなければならない部分もある．しかしこれまで Web サイトの構築に過大な投資してきたことを反省すべき時期だと考え，あえて本報告のようなコストパフォーマンスを重視したアーキテクチャの Web サイトを提案したいと考える．

次にこれまで技術者の経験と勘及び潤沢な投資に頼っていた Web サイトのサイジング設計に関して，比較的容易に見通しを立てられる方法を考案した．詳細な評価は今後の実際の移行を待たねばならないが，ある程度は使えると考えている．

7. 今後の課題

新しいアーキテクチャもサイジング方法も実際の移行に伴う分析・評価ができていないので，まずそれが一番の課題である．特にサイジングに関してはまだまだ課題が多い．

サイジング方法の考案は初めての試みのため，考え方も手法も洗練されていないところが多い．まず考え方に根本的な誤りがないかどうか精査が必要だが，特に簡略化のし過ぎによる実態からの乖離は避けなければならない．したがって基本的なサイジング計算モデルそのもの，種々の Web サイトの特性に対応できる計算方法と測定方法，シングルプロファイルの定義と設定方法，過渡的負荷状態や処理のオーバーヘッドの取り扱いなどに関しては今後見直したい．

しかし重要なのは，本当の意味で簡単・実用的かつそれなりの精度を持った方法にすることである．この辺のバランスは難しいが，それぞれの Web サイト構築の事情に合わせた方法を採用すべきである．今回の検討には 3 ヵ月弱の期間と 10 人月余りの工数がかかっているが，今後実際のサイジングを行う場合は，より短期間・低コストで行えるようにしなければならない．そのためには理論的基盤の確立はもちろんのこと，作業の効率化が必須になる．特に Web サイトの分析やベンチマークテストの効率化のための工夫，シミュレーションツールによる机上検証の導入，システマティックなサイジング計算方法の確立などが必要である．

参考文献

[1] 亀田壽夫，紀一誠，李頡著：性能評価の基礎と応用，共立出版

[2] IBM 編：拡張のための計画他，developerWorks

[3] シーマーク編：logicletBox フレームワーク説明書，http://www.seamark.co.jp/