株式会社インディバル様「Yahoo!リクナビ　ショットワークス（Shotworks）」サイトの構築事例 ～Linuxを使ったハイアベイラビリティDBシステムの実装～

Case Study: Construction of Website “Shotworks in Yahoo!Rikunabi” for Indival ～ Mounting of The High Availability Database System Using Linux ～

細田　隼平　Junpei Hosoda
水上　純治　Junji Mizukami

概要

当社は、株式会社インディバル様（ヤフー株式会社と株式会社リクルートとの合弁会社）が提供する求人求職マッチングサービス「ショットワークス（Shotworks）(*1)」のシステム構築を担当した。本システムはWebアプリケーションASP(*2)システムであり、DBシステムにはハイアベイラビリティ(*3)（高可用性）とハイスケーラビリティ(*4)（高拡張性）が求められた。これらを安価に実現するため、Linuxを採用しHA(*5)構成や記憶装置のパーティション構成に工夫を施した。また、パフォーマンスを事前に検証するためサービスの負荷分布を予測し、実機を手配する前に検証環境にてパフォーマンスを評価した。

これにより適正なサーバー構成を決定でき、高度なマッチング検索機能を高速かつ安価に提供することができたと考える。

1. はじめに

昨今、求人広告市場は、その展開の場をインターネット上にもひろげ、アルバイト・パート分野を中心に急成長を続けている。ヤフー株式会社様と株式会社リクルート様とは、両社の合弁会社である株式会社インディバルを設立し、短期単発アルバイト・パートを対象とした求人求職マッチングサービス「ショットワークス（Shotworks）」を立ち上げ、インターネットにおける求人事業の拡充に乗り出した。当社は、「ショットワークス（Shotworks）」のシステム基盤であるASPサイトの構築を受託し、業務アプリケーションの他に、Linuxを使ったハイアベイラビリティDBシステムの実装に取り組んだ。

本サービス（図1）の利用者は、「求人企業」と一般個人である「求職者」に分けられ、数十万人規模の利用を想定して構築されている。求人企業は本サービスを通じて、リアルタイムで求人情報をサイトに掲載できる。また求人条件にマッチした求職者を検索し、該当者に求人情報を配信することもできる。一方、求職者は自身の求職情報をサイトに登録し、希望する条件（日程・地域など）にあった求人情報を検索、本システムを通じて応募することが可能となる。その他に、本サービスでは応募者の管理、採用・不採用の通知連絡などの採用管理機能を備えている。また、今後は勤怠管理機能や給与支払機能を順次サービス展開していく予定である。

本サービスの重点は、求人企業と求職者が膨大な登録データの中から最適な求人求職情報を見つける高度なマッチング検索機能にあり、その優劣がサービスの優劣を左右する。本サービ

(*1) http://shotworks.yahoo.co.jp/
(*2) ASP（Application Service Provider）：アプリケーションの機能を企業に提供する事業者。
(*3) ハイアベイラビリティ（High Availability）：システムが利用不能となることが少ないということ。
(*4) ハイスケーラビリティ（High Scalability）：システムの利用者や負荷の増大に応じて、柔軟に性能や機能を向上させられること。
(*5) HA（High Availability）

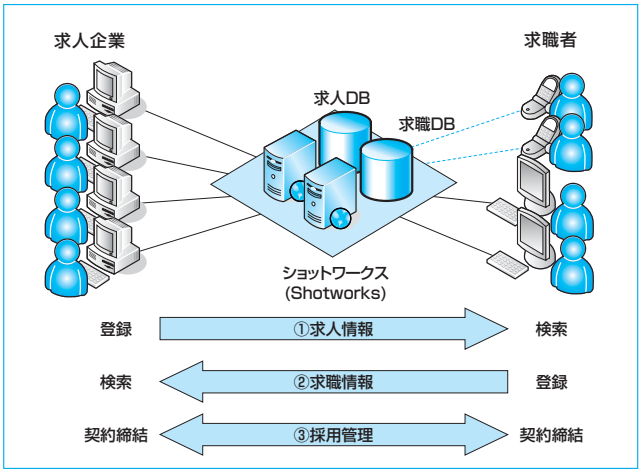

図1　サービス概要

スは、大規模なデータベースを持ち、ASPでオンラインサービスを提供している。従って、この膨大なデータ量に対し、如何に高度なマッチング検索機能を高速かつ安価に実現できるかが最大の課題であった。さらに、サービスはインターネット上で展開されるためノンストップ運用のハイアベイラビリティと、急激な利用者増に適応するハイスケーラビリティを備える必要があった。

当プロジェクトではこれらの課題に対し、Linuxを使ったIA(*6) サーバーを採用し高性能かつ安価なDBシステムを実装することで対応した。以下ではそのDBシステムの構成の一部とパフォーマンス検証の事例を概説する。

2. システム構成

今回構築したシステムの全体概要図を図2に示す。本稿で焦点となるDBシステムはサーバー3台構成で、OSにRed Hat Enterprise Linuxを採用し、共有ディスクはファイバーチャネルを利用したSAN(*7) 構成となっている。RDBMS(*8) にはIBM社のDB2 Universal Databaseを採用した。

本DBシステムでは前述した課題に対応するため、以下の対策を施している。

（1）ノンストップ運用を実現するため、HAソフトウェアとしてSteelEye社のLifeKeeperを採用した。また、ハードウェアを含め低コストを実現できるHA構成とした。

（2）DBシステム拡張時に最小限のサービス停止時間でサービス再開できるよう、パーティション分割を工夫した物理設計を行った。

（3）高速なDB接続を実現するため、アプリケーションサーバーとの接続では当社が独自開発したソケット通信を採用した。

2.1 ハイアベイラビリティと低コストを両立させるHA構成

LifeKeeperはハイアベイラビリティを実現するためのソフトウェアである。具体的には障害発生時にアプリケーション、共有ディスク、IPアドレスなどのリソースを障害発生機からStandby機に移動させ、サービスを継続させるためのソフト

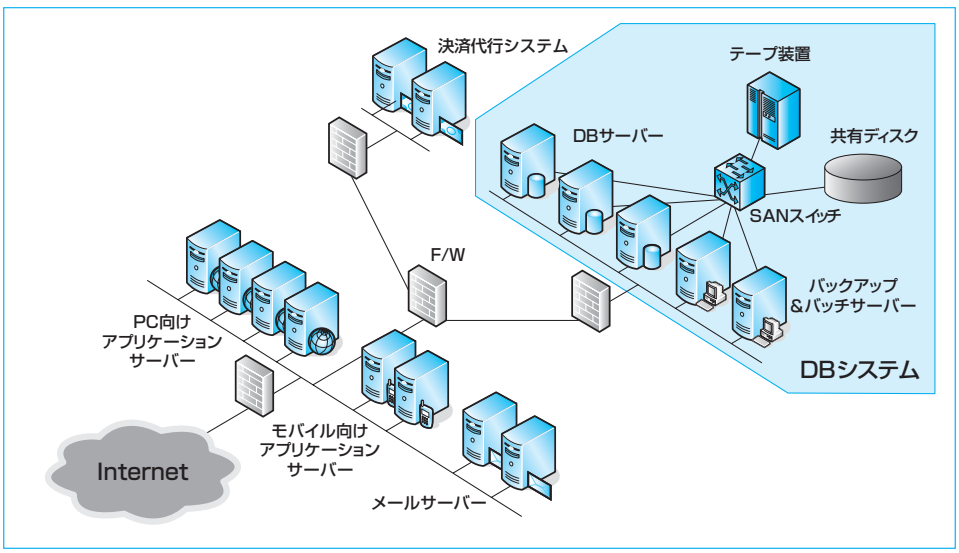

図2　システム全体概要図

(*6) IA（Intel Architecture）：Intel社マイクロプロセッサの基本設計。IAサーバーとは、Intel社製CPUまたは互換CPUを搭載したサーバー、UNIXサーバーと比較し安価。
(*7) SAN（Storage Area Network）：ストレージをファイバーチャネルなどにより接続したネットワーク。複数のサーバーからアクセスできる。
(*8) RDBMS（Relational DataBase Management System）：リレーショナルデータベースを管理するソフトウェア。

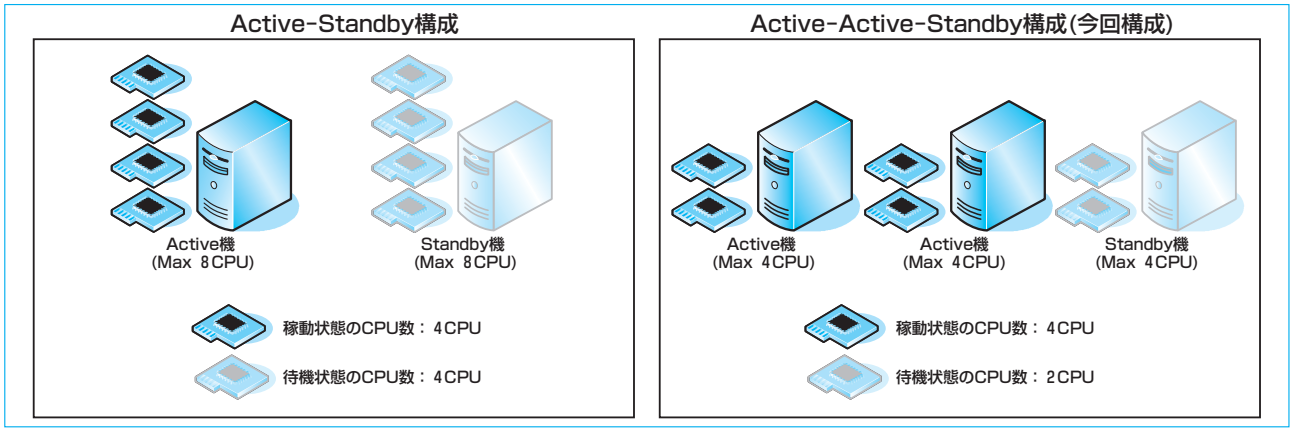

図3　Active-Standby構成とActive-Active-Standby構成の比較

ウェアである。初期設定から導入後の設定変更に至るまでGUIで容易に操作できる。また、各サーバー間でのリソースの移動も柔軟に実施可能である。

HA構成として、もっともシンプルな構成はActive-Standby構成である。この構成は1台を稼動状態、もう1台を待機状態にする。構成がシンプルであるため、構築も管理も容易である。しかし、高性能のサーバーが2台必要となるため高コストとなる。そこで本DBシステムで採用した構成は、3台のサーバーによるActive-Active-Standby構成である。2台のサーバーを稼動状態とし処理を分散させ、残り1台を待機状態とする構成である。

後述するパフォーマンステストの結果、稼動状態のサーバーに必要となるCPU数は計4CPUと試算していた。将来的なCPU増設も視野に入れると、Active-Standby構成の場合、最大8CPUをサポートするハイエンドサーバーが2台必要となる。Active-Active-Standby構成の場合、最大4CPUをサポートするミドルレンジサーバーが3台必要となる（図3参照）。サーバーに掛かるコストを比較すると、Active-Active-Standby構成の方が約35%コストダウンできるという結果となった。

また、Active-Active-Standby構成ではサーバー1台あたりのコストが低いためサーバーをこまめに増設できる点や、2台のサーバーで同時に障害が発生した際にリソースを残り1台のサーバーに集約してサービスを継続できる点も利点として挙げられる。1台で稼動させる場合、性能は平常運用時の約半分になるが、利用ピーク時に合わせてシステムをサイジングしているため、ほとんどの場合において遅延なくサービスを継続できると判断した。

2.2 将来の拡張を容易にさせるDB2のパーティション構成

DB2は複数のサーバーを組み合わせて、全体で1つのデータベースとして見せるパーティションデータベースという機能を有している。各サーバーは0個以上のパーティションを担当し、クライアントからの要求を分担して処理している。

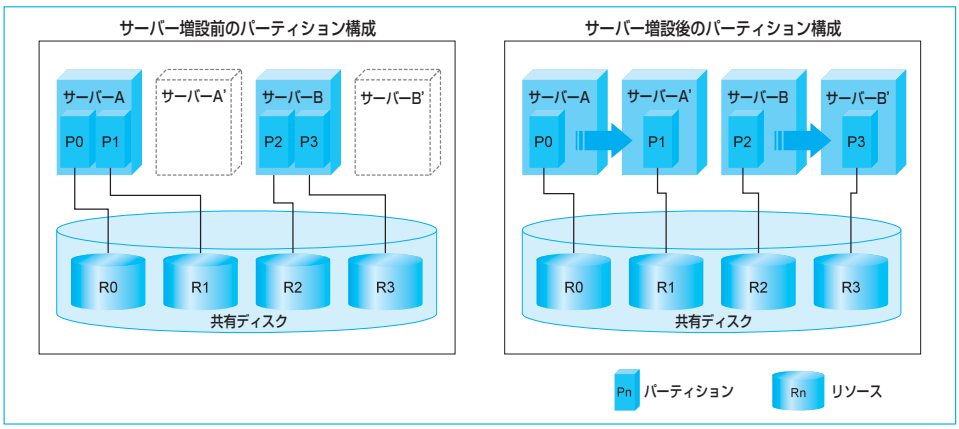

図4　サーバー増設時の移行イメージ

本システムはリリース後2年までの利用者増加を想定しサイジングされている。サービス機能や利用者の増加への対策のひとつとしてサーバー増設が挙げられるが、24時間サービスを提供している本システムでは、安全かつ短時間に増設作業を行える必要があった。そこで、通常2サーバーの場合は2パーティションで構成するのに対し、本DBシステムでは2サーバーに対し4パーティションで構成している。これにより、既存パーティションを分割せず、サーバー間での共有ディスクリソースの移動とIPアドレスの割り当て等の最小手順でサーバー増設を実現でき、サービス停止時間の短縮化を図ることができる（図4参照）。

2.3 ソケット通信を用いたアプリケーションサーバーとの接続

本システムのアプリケーションサーバーは、OSがFreeBSD(*9)、アプリケーション開発言語がPHP(*10)となった。DB2ではFreeBSD上でのDB2純正クライアントソフトが用意されておらず、DB接続実現に向け次のように幾つかのアプローチを行った。

（1）Linux版DB2純正クライアントソフトの活用
- インストールスクリプトを書き換えてFreeBSDへインストール
- FreeBSDのLinuxエミュレータ上での稼動
- Linuxに一旦インストールしたものをFreeBSDへコピー

（2）JDBC(*11)接続を利用
- PHPからのDBアクセスをJava経由とし環境依存性の低いドライバーを使用

（3）他社製ソフトウェアの活用

（4）接続ソフトウェアの独自開発
- SOAP(*12)を利用したJavaプログラムでの連携
- ソケット通信(*13)を利用したJavaプログラムでの連携
- ソケット通信を利用したCプログラムでの連携

結果、（1）（2）ではいずれもDB接続が実現できなかった。（3）については、ODBC(*14)をサーバー間で橋渡しするブリッジソフトウェアが存在することがわかった。特徴としては、インストールのみで導入が可能なためコストが少ない反面、ODBCとODBCとのブリッジ接続に注力されたソフトウェアであるため、パフォーマンス面で期待が持てない点が挙げられた。インターネット向けシステムにおいて、パフォーマンス面で不安が残ることは大きなリスクであった。一方、（4）については、本システムに不要な処理の省略や、パーティションデータベース環境下で効力を発揮するようなアルゴリズムの組み込みなどDB2に特化したパフォーマンスチューニングが施せる反面、開発コストが増大するという点と、一からの開発であるため新規開発リスクが伴うという点が考えられた。

以上の調査結果から、アクセス数の少ないサービス開始初期は他社製のブリッジソフトウェアを利用し、その間に接続ソフトウェアを独自開発することと決定した。なお、開発する接続ソフトウェアとしては、SOAP通信やJavaプログラムでのソケット通信ではレスポンス劣化や成熟度に不安があるため、ソケット通信を利用したCプログラムで実装した。また、サービス稼動途中でブリッジソフトウェアから独自開発の接続ソフトウェアへ入れ替えとなるため、アプリケーションに対するAPI(*15)はブリッジソフトウェアが採用しているODBC準拠のAPIに合わせることとした（図5参照）。

接続ソフトウェアの開発が順調に進んだことにより、ブリッジソフトウェアの移行はシステムテスト期間中に実施することができた。移行ではスケジュールに影響を与えるような大きな

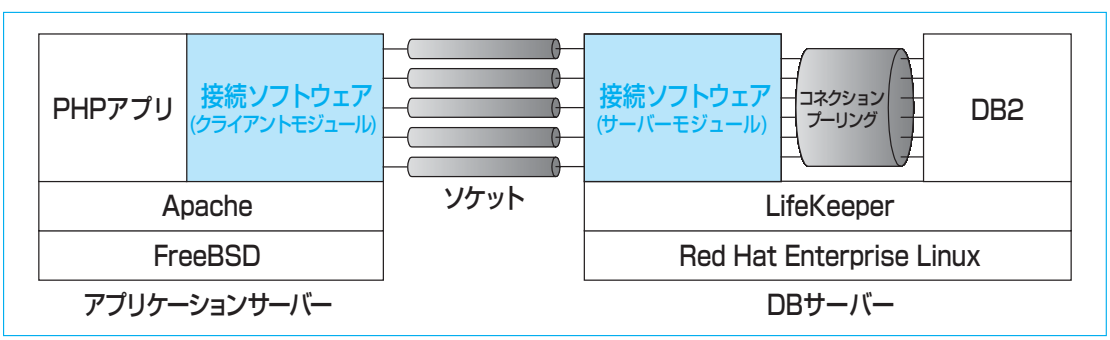

図5　独自開発の接続ソフトウェア概念図

(*9) FreeBSD: PC/AT互換機用のUNIX互換OS。オープンソースソフトウェア。
(*10) PHP（PHP:Hypertext Preprocessor）：Webアプリケーション開発用のスクリプト言語。オープンソースソフトウェア。
(*11) JDBC（Java DataBase Connectivity）：Javaプログラムからデータベースにアクセスするための API。
(*12) SOAP（Simple Object Access Protocol）：他のコンピューターにあるオブジェクトにアクセスするためのプロトコル。XMLをベースとし、HTTPなどのプロトコル上で利用できる。
(*13) ソケット通信: TCP/IPにおいて利用できるネットワーク用API。ソケットはIPアドレスとポート番号を組み合わせたネットワークアドレス。
(*14) ODBC（Open DataBase Connectivity）：Microsoft社が提唱しているデータベースにアクセスするための API。
(*15) API（Application Program Interface）：OSやミドルウェアがアプリケーションに対して公開している命令や関数、またそれらを使用するための手続き規約。

問題は発生しなかったが、いくつかのアプリケーション修正は発生した。DB2純正クライアントソフトや他社製ブリッジソフトウェアなどのパッケージ製品は、アクセスするアプリケーション側に、ある程度の曖昧さを許してくれた。教科書どおりの接続手順やSQL文でなくても、補正しエラーを吸収してくれる処理が組み込まれていると思われる。一方、開発した接続ソフトウェアにはそのような機能を組み込んでいないため、SQL関連のエラーが顕在化することとなった。接続ソフトウェアは優先順位の低い機能を捨てて高速化を目指しているので、この問題にはWebアプリケーション側の改修で対応した。

3. パフォーマンスの検証と環境構築リハーサルの実施

今回のシステム構築では、DBシステムの性能がサービスレベルに大きく影響するため、サーバーのサイジングに注力する必要があった。そのため、アプリケーション開発の開始前に負荷テストを行い、パフォーマンスを検証した。また、本番環境の構築前にリハーサルを行ったことも特徴である。問題発生時の手戻りコストがプロジェクトに与える影響は、基盤環境の構築では非常に大きくなる。これらのリスクへの対策として、当プロジェクトでは可能な限り事前検証作業を行った。

3.1 負荷の予測

負荷の予測を立てるに当たり、お客さまからいただいた利用者数予測資料の他、国内IX(*16)各拠点の混雑状況と、本システムの利用ケースを考慮した予測を立てた。各利用ケースのアクセス頻度をデータから類推し、必要な処理性能を導き出している。例えば、モバイル端末による勤務管理サービスのアクセス数は次のように予測した。勤務者は勤務開始時に出勤登録、勤務終了時に終了登録をモバイル端末から行う仕様となっている。そこで、約1,000件の様々な業種の勤務データを収集し、出勤時間と終了時間の累積値から各時間の勤務者数を算出した。さらに算出した勤務者数を出勤時間と終了時間で異なる分散値を用い正規分布に分散させ、1分あたりのアクセス数を算出した。図6に勤務管理サービスのアクセス数予測結果を示す。

このように、利用ケースごとにアクセス数を予測・積層してサービス全体の負荷を予測した。1日あたりのアクセス数を割り算するような単純計算では、精確な負荷を予測できない。

3.2 パフォーマンス検証

パフォーマンスの検証には、IBM社のLinuxCoC(*17)を約1ヵ月間利用した。LinuxCoCを利用したことにより、本番環境の調達スケジュールに左右されることなく本番環境規模でのパフォーマンス検証が行えた。LinuxCoCでは、サーバーはもちろんストレージやSAN環境も準備されている。また、サポート面では各種製品のプロフェッショナルが常駐しており、いつでも技術サポートを受けられる環境である。

パフォーマンス検証では、要件定義の時点で処理が重くなると想定した画面とSQLのプロトタイプを作成し、2年後を想定したデータ量のサンプルを用いて、処理が最速となるような画面・SQL・DB設計のチューニングを施した。ここでは実際に検証した例として、求人企業が300万人の求職者を対象に十数種類の条件を用い、該当する人物を検索する機能を挙げる。

パフォーマンス検証の前半として、1ユーザーが検索機能を利用した際のレスポンス時間向上のためのチューニングを行っ

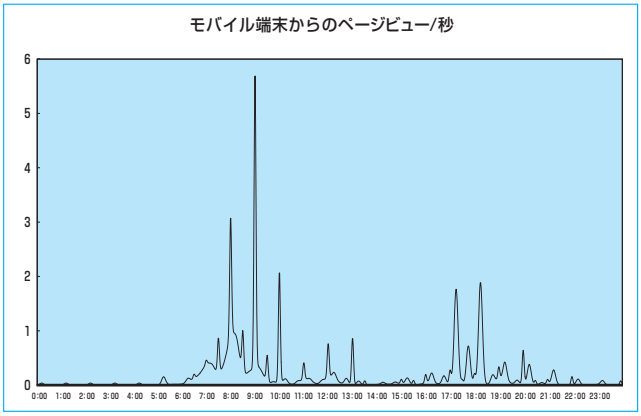

図6　負荷予測グラフ（モバイル端末）

実施事項	画面表示までの時間(秒)
実施前	32秒
SQL文の最適化 およびインデックスの設定	16秒
DB2のパラメーター調整	4秒
データのパーティション分割の最適化	2秒

図7　チューニング成果

(*16) IX（Internet eXchange）：インターネットサービスプロバイダや学術ネットワークを相互に接続するインターネット上の相互接続ポイント。
(*17) LinuxCoC（Linux Center of Competency）：Linuxコンピテンシー・センター。日本アイ・ビー・エムが保有する国内最大級のLinux環境検証施設。

た。チューニングの手法としては、SQL文の最適化およびインデックスの設定、データベースパラメータの変更を順次実施した。図7にチューニングの成果をまとめた。

パフォーマンス検証の後半では、Microsoft社の負荷テストツール「WAST(*18)」を用いて負荷テストを行った。負荷テストの目的は、数十ユーザーによる同時アクセス環境下でDBシステムが余裕のある挙動を取れることを確認することである。WASTではセッション(*19)の管理に作り込みが必要であったため、セッションの管理が必要無いよう、1画面のみの構成とした。また、負荷をかける際の検索条件は毎回条件が変化するよう、アプリケーションサーバー側でランダムに生成するようにした。

負荷テストの結果からDBシステムには、メモリの使用状況は十分に余裕があり、ディスクI/Oにも余裕があることを確認できた。しかし、実行待ちプロセス数が多いことからCPU不足であると判断できるため、今後のマシン増強の際にはまずDBサーバーのCPU増設、次いでDBサーバーの増設という順序でCPU増強対応の可能性が発生すると予測している。

3.3 環境構築リハーサルの実施

当プロジェクトではパフォーマンス検証の他、IBM社プリセールスチームの技術支援の下、本番環境構築のリハーサルも行った。リハーサルの目的は以下のとおりである。

（1）ハードウェア納品受け入れ後のセットアップ時間短縮

- パーティションデータベース、LifeKeeperによるHA化は経験が浅かったため、その技術仕様・実装方法の理解と習熟
- Linux等既知の技術の仕様・実装方法の確認
- 各種ソフトウェアの自動インストール・設定スクリプトの作成

（2）未知の問題のあぶり出し

- 実機が無ければ直面しない問題の洗い出しと解決

リハーサルの実施はハードウェア納品後から基盤セットアップ完了までの期間を短縮するための工夫である。さらに、リスクを事前に把握する、ないしは解消することにもつながる。残念ながら今回は、本番環境とまったく同一の環境を揃えることができなかったため、本番環境セットアップにおいてもいくつかの問題は発生したが、リハーサルを実施した価値は大きかったと考える。

4. おわりに

本DBシステムを構築するにあたっての最大の課題は、如何にして高度なマッチング検索機能を高速かつ安価に実現できるかであった。また、インターネットというチャネルの特性上、ノンストップ運用のハイアベイラビリティと柔軟な増強を可能とするハイスケーラビリティを求められるシステムでもあった。

当プロジェクトでは、DB接続ソフトウェアを独自に開発すると共に、早い時期でのパフォーマンス検証を実施し、DB処理のチューニングを行うことにより高速化へのアプローチを図った。また、Active-Active-StandbyというHA構成によりハイアベイラビリティ環境を低コストで実現し、DBパーティション構成の工夫によりハイスケーラビリティも確保した。

つい数年前まで、オープン系においてハイアベイラビリティなシステムを構築するには、UNIXを採用することが普通であった。現在、その選択肢にLinuxが加わったと考えている。それは今回の構築を通して、LinuxがOSとしての高い完成度を備えており、今回の適用範囲においてはUNIXと比較しても何ら遜色が無いことを実証できたからである。そして、スケーラビリティやコストという側面に対しても策を講じることを可能とし、その策を施すことにより大きな課題を解決できるという可能性を確認できたと考える。

細田　隼平
Junpei Hosoda

・ビジネスソリューション事業本部
CRMソリューション部
・基盤構築を始めとするWebシステム開発に従事

水上　純治
Junji Mizukami

・ビジネスソリューション事業本部
CRMソリューション部
・開発プロジェクトにおけるリーダー業務に従事

(*18) WAST（Web Application Stress Tool）：Microsoft社が無償で提供しているWebシステム用の負荷テストツール。
(*19) セッション：Webシステムにおいて、1人のユーザーが行動する際の一連の行動（画面遷移）。