２００７年度オープンソースソフトウェア活用基盤整備事業

自治体等におけるオープンソースソフトウェア活用に向けての導入実証

～OSSによる統合DBを介した基幹システムと業務システム連携の実証～

実証事業実施報告書

２００８年2月

株式会社 ＢＳＮアイネット

用語定義

用　語	意　味
API	OSなどが用意したソフトウェアを開発する際に使用できるファイル制御、ウィンドウ制御等の命令や関数の集合のこと。ソフトウェアの開発者がすべての機能をプログラミングするのは困難で無駄も発生するため、共通して利用する機能が、OSやミドルウェアなどの形でまとめて提供されている。個々の開発者は機能を参照するだけで、簡単にその機能を利用したソフトウェアを作成できる。 （Application Programming Interface）
APPLIC	財団法人全国地域情報推進協会のこと。地方公共団体の情報システムの抜本的改革や、地方公共団体内外の地域における多数の情報システムをオープンに連携させるための基盤の構築を推進する組織。 (The Association for Promotion of Public Local Information)
Axis	SOAPエンジンの一つで、クライアント、サーバ、ゲートウェイなどのSOAP処理を構築するためのフレームワーク。Axisの現在のバージョンはJavaで書かれている。
cron	Linux系のシステムで、コマンド（ジョブ）を定期的に自動実行するための機能。管理を行なう場合、ジョブを定期的に自動実行するため、この機能を使う。
CSV	カンマ(",")で列が区切られ、改行で行が区切られているファイル形式。表計算ソフトやデータベースを保存する際に使う形式である。異なる種類のアプリケーション間のデータ交換に使われることが多い。 (Comma Separated Values)
DDL	データベースのデータ構造を定義するための言語である。使用することでテーブル全体の作成・変更・削除など行うことが出来る。 (Data Definition Language)
EA	企業や政府機関・自治体などの組織（enterprise）の手順やシステムの標準化、組織の最適化を進め、効率よい組織の運営を行うための方法論のこと。 (Enterprise Architecture)
EXE形式	Windowsなどで実行可能なプログラムが収められたファイル。ファイル名は「.exe」という拡張子がつけられる。
J2EE	「Java 2（Javaの第2版）」のプラットフォームの一つで、企業業務システムや向けのサーバに必要な機能をまとめたもの。 (Java 2 Enterprise Edition)
Java	Sun Microsystems社が開発したプログラミング言語であり、完全なオブジェクト指向性を備え、標準でセキュリティ機能やネットワーク関連の機能も豊富に搭載されている。Javaではプラットフォームの非依存を目

	標としており、特定のOSに依存することなく、基本的にはどのようなプラットフォームでも動作する。
JDBC	Javaとデータベースを接続するためのAPI。 （Java DataBase Connectivity）
LDAP	ネットワークを利用するユーザーやコンピュータなど様々な情報を管理する「ディレクトリサービス」へアクセスするためのプロトコルのこと。 (Lightweight Directory Access Protocol)
OpenStandia （オープンスタンディア）	株式会社野村総合研究所が提供しているサービスで、独自に選定したOSS製品の組み合わせを事前に検証し、セキュリティや性能チューニングなどの基盤関連パラメータの設計と部品群の実装を行うものである。今回の実証では、以下のOSSを使用したため、それらの構築支援およびサポートサービスの提供を受けている。 （各機能概要については、1.3.3に記述） ・Cent OS ・Apache ・mod_proxy_balancer ・Heartbeat ・JBoss AS ・PostgreSQL+ Pgpool
OSAC	自治体のオープンスタンダードを技術的に支援する企業・団体など17団体で構成されたコンソーシアム。電子自治体アプリケーション・シェア推進協議会や地域ITベンダー等に対して、具体的な支援等を行うことを目的として設立された。 (Open Standardisation Support Consortium)
RFP	情報システムを導入するにあたり、ユーザーよりベンダーに提供する、システム調達仕様書。 (Request For Proposal)
SOAP	ネットワーク上のアプリケーション間（オブジェクト間）の情報をHTTPなどの通信手段を用いて交換し合うためのプロトコル。このプロトコルを表現するための手段としてXMLが利用されている。 (Simple Object Access Protocol)
SQL	リレーショナルデータベースの操作を行なうための言語の一つ。 (Structured Query Language)
Unicode	米国の情報関連企業が中心となって提唱し、標準化された文字コード体系。一部は国際規格であるISO/IEC 10646として定められている。
VB.NET	Microsoft Visual Basic .NET（マイクロソフト ビジュアル ベーシック ドットネット）の略称で、マイクロソフトが開発したプログラミング言語。 Visual Basic（VB）はVB.NETの前のプログラミング言語で、Visual Ba

	sic6.0までバージョンアップされている。 .NET Frameworkとは各種の.NET対応サービス・ソフトウェアが利用可能となる基盤ソフトウェアのこと。Windowsアプリケーションだけでなく XML・Webサービスや Webアプリケーションなど Webベースのアプリケーションなどに対応。
Webサービス	HTTPなどのインターネット関連技術を応用して、SOAPと呼ばれるXML形式のプロトコルを用いて、メッセージの送受信を行う技術、またはそれを適用したサービス。
WS-Reliability	Webサービスを提供する際に、オープンで信頼性のあるメッセージングを実現するための標準化をめざした仕様。メッセージの送達保証、重複防止、順序保証の機能を提供するもの。
WS-Security	Microsoft社、VeriSign社、IBM社が共同して提唱している、SOAPメッセージの信頼性を保証するための仕様。XML Signatureや XML Encryptionといった、XML文書のセキュリティ保護のための仕様を、SOAPで使用する方法を定義したもの。
XML	文書やデータの意味や構造を記述し汎用的に使用することの出来るのマークアップ言語の一つ。マークアップ言語とは、構造をテキストファイルで記述することができ、「タグ」と呼ばれる特定の文字列で地の文に構造を埋め込んでいく言語のことで、ユーザーが独自のタグを指定できることから、マークアップ言語を作成するためのメタ言語とも言われる。 (Extensible Markup Language)
学童保育支援システム	一部の自治体では、就労等により昼間保護者のいない家庭の小学校低学年児童の健全育成を図ることを目的に、保護者が帰宅するまでの間、児童を預かるための施設を設置しているが、その施設利用料を徴収・管理するシステム。
共通基盤	システム間連携や認証といった共通機能を、共通基盤という１つのシステムとしたもの。共通基盤を導入すれば、各業務システムは個々に連携する必要がなく、共通基盤とのみ連携すればよいため、連携に伴う初期コスト、保守コストを抑制することができる。APPLICが提唱している地域情報プラットフォームも共通基盤の仕様を定めたものといえる。
地域情報プラットフォーム	自治体が持つ情報システムをはじめとした、地域内外のあらゆる情報システムを全国規模で連携させるための共通基盤のこと。地域情報プラットフォームでは、XMLなどの技術を活用することによって、プラットフォームやデータ形式などが異なるシステム間でのシームレスな連携を可能とし、類似したデータや機能の重複を排除することを目的としている。
電子自治体アプリケーション・	効率的な自治体システムの構築を目的として設立された組織で、アプリケーション・シェアの実現を目指しており、現在8県2市が参加している。

シェア推進協議会	**アプリケーションをみんなの共有財産として開発･利活用するための議論を重ねており、福岡県が開発した「電子自治体共通化技術標準」をベースに各種研究を重ねている。**

【目次】

はじめに

現在、自治体においては、逼迫する財政状況の中、部分的な情報システムの整備を積み重ねてきた結果、庁内には様々なシステムが混在し、情報共有化を困難なものとしている。

また、オープンシステムにより基幹系システムを構築している場合にも、そのシステムにおける設計内容等がブラックボックス化され、個別業務システムの調達に際して、メインフレーマーと同等に、地元ITベンダーに広く提案の機会を提供するまでに至っていないのが現状である。

さらに、システム調達にあたっては、必ずミドルウェア等のライセンスロイヤリティが発生し、調達コストを引き上げる要因にもなっている。

このような状況下、課題の解決策の一つとして登場してきたのが、地域情報プラットフォームという発想であり、庁内の各種システム間の連携やデータの相互活用を、アプリケーションに共通する機能や、アプリケーション間の連携機能を持つ共通基盤を実装して行う。

しかし、これまでの共通基盤には、商用OSやミドルウェアを前提に構築しているためのライセンスコストの問題や統合データベース(以下、「統合DB」という)についての実装事例がないなど、地域情報プラットフォームの普及の課題となっている。

そうした中で、今回の導入実証では地域情報プラットフォームの発想に基づくシステム構築に際して、ライセンスフリーのオープンソースソフトウェア(以下、「OSS」という)の採用を、重要なテーマとして位置づけ、実証フィールドの提供等上越市の全面的な協力をいただき行うことができた。

具体的には、基幹系システムとシステム連携を行うための統合DBをOSSで構築し、さらにその統合DB上で個別業務システムとして学童保育支援システムを稼動させるものである。また、各システム間の連携は、オープンスタンダード化支援コンソーシアム（以下「OSAC」という）が提唱する共通基盤システムを介して行うもので、これもOSSで構築する。

目的として、共通基盤コードのフルオープンソース化、統合DBの実装、ドキュメント・ライセンスの整備の３点に重点を置くことにし、各種実証に取り組んだ。具体的な実証内容とその結果および実証作業中に発生した問題とその対応策については本編で述べているが、基本的にはOSS化したシステムについては、動作検証、性能評価を実施した結果、十分実運用に耐えられる成果がでたものといえる。

しかし、まだOSSには様々な普及上の課題があることも事実である。具体的には、OSSの高負荷時の性能や信頼性の問題、保証とサポートの問題、ライセンスが複雑であるとういう問題、外字コードの問題等である。

今回の実証事業では、今後自治体が実運用を行う上での課題を明らかにし、解決策についても提言している。また、OSS化の推進に関する基本的な方向性の提言と、それに沿った、株式会社ＢＳＮアイネットの今後の活動方針についても言及している。

今回の実証事業で、OSS化したシステムを全国の自治体が自由に利用できることにより、今後一層自治体におけるOSS導入が加速するものと期待したい。

１．　導入実証の目的と概要

１．１　導入実証の背景

１．１．１　自治体が目指すIT化の方向性

自治体においては、「e-JAPAN戦略」に次ぐ「IT新改革戦略」や電子自治体システムの構築、それに伴う情報共有化の流れにより、従来以上に全庁的な情報共有・情報の一元管理の必要性が高まっており、下記のような方向性を持って、IT化を推進しようとしている。

（１）電子自治体の構築
　・市民の視点に立った電子自治体の構築
（２）情報システムの全体最適化の実現
　・自治体業務を効率化できる全体最適を実現する
　・情報の共有化、一元化の実現
（３）IT投資の適正化
　・総務省の「情報システムに係る政府調達の基本指針」に準じた分割調達
　・運用、保守、ライセンス料等のコストの適正化
（４）ITベンダーの育成
　・地域産業活性化の視点からの地元ITベンダーの育成

１．１．２ 自治体におけるIT化の現状

前節のような方針を具体的に実施していくにあたっては、自治体が現状抱えるいくつかの阻害要因が存在する。特に重要な影響を与えている制約としては次のようなものがある。

（１）財政面での制約
財政面で余裕のある自治体はごく少数であり、自主財源によるIT化や情報通信環境の整備が困難であるのが実情である。
（２）技術面での制約
最新のIT技術や、セキュリティ対応などは専門的な知識・スキルが必要であり、こうした専門家の支援を受けずに構築・運用を進めていくのは困難である。
（３）人材面での制約
自治体においては、IT化やネットワークなどの知識を有する人材を確保することが困難である。また、情報システム調達の際、適切なRFPを作れる人材がいないことが多い。

１．１．３　自治体情報システムの課題

現状の自治体の情報システムは、限られた予算と技術の変遷に合わせた段階的かつ部分的なシステム整備が行われてきた結果、次のような課題を抱えている。

（1）ベンダーロックインとライセンスコストの増大

・個別にシステムを調達してきた結果、さまざまなベンダーの製品が乱立し、ブラックボックス化によって開発ベンダーへのロックインが発生

・システム数の増加とともに、商用OS・ミドルウェア等の商用製品（以下、商用ソフトウェアという）のライセンスコストが増大

（2）データ活用が困難

・データベース仕様が公開されず、データがあるのに活用できない

・ニーズに応えたシステムの分割発注が困難

（3）運用負荷の増大

・ユーザー、組織の管理や、権限の管理がシステムごとに行われ、人事異動時などに多大な管理負荷が発生

・エンドユーザーの使い勝手もシステム数の増加とともに悪化

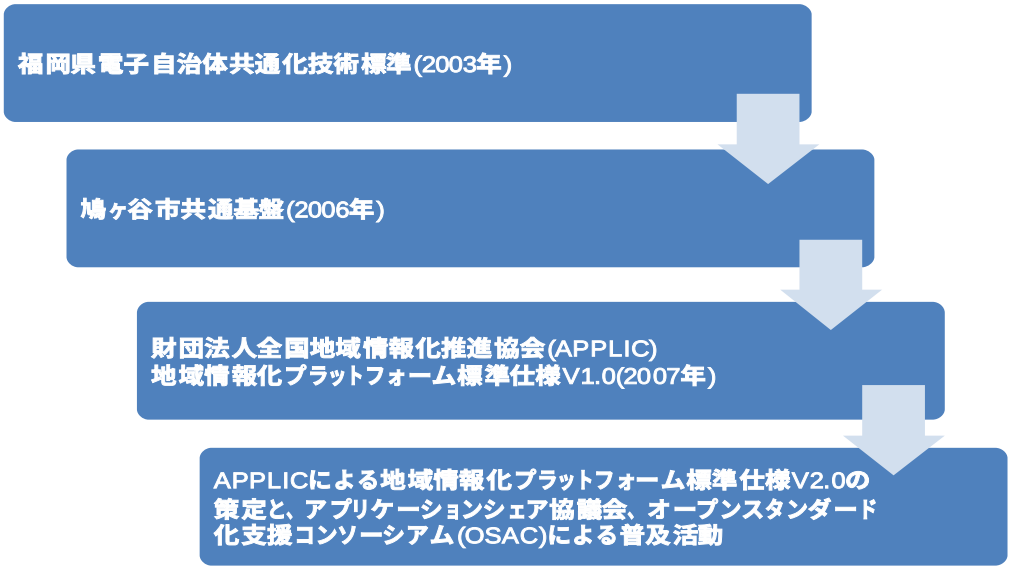
１．１．４　地域情報プラットフォームの誕生

前節のような課題の解決策の一つが、統合DBによるデータ統合であった。これは、住民記録や税などの基幹情報システムのデータを統合DBに格納し、他のシステムでの活用を図るものであり、株式会社ＢＳＮアイネットも2002年に新潟市に「汎用データベースシステム」として提供してきた。

しかしながら、前述の、ユーザー、組織、権限の管理や利用者の使い勝手については、統合DBだけでは限界がある。こうした背景と、組織内の情報システムの全体最適化を志向するEA（エンタープライズ・アーキテクチャ）の観点から生まれてきたのが、地域情報プラットフォームという考え方である。

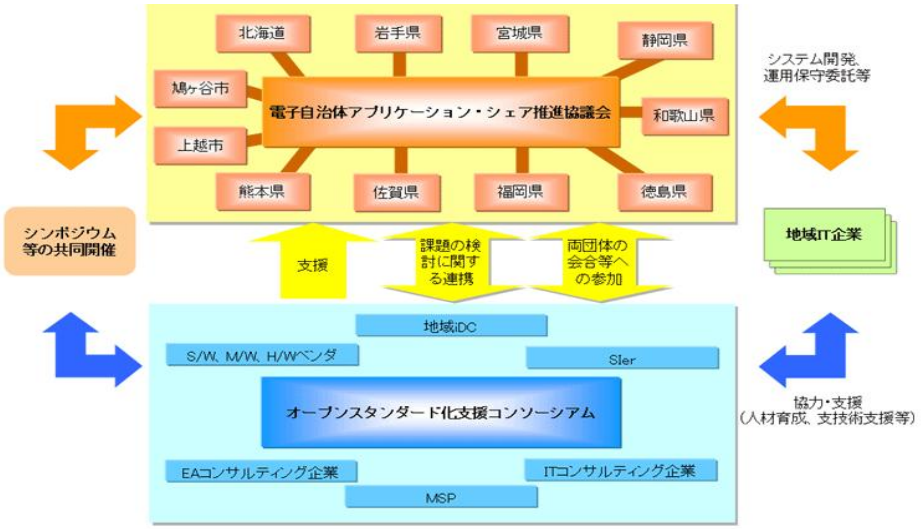
地域情報プラットフォームは、2007年7月に財団法人全国地域情報化推進協会（以下「APPLIC」という）が地域情報プラットフォーム標準仕様V1.0をリリースしている。この考え方の先がけとしては、2003年に実施された、福岡県電子自治体共通化技術標準（以下「福岡基盤」という）の構築がある。その後、埼玉県鳩ケ谷市がこの福岡基盤の仕様及びプログラムに機能追加を行い、鳩ケ谷市共通基盤（以下「鳩ヶ谷基盤」という）としてリリースを行っている。その後APPLICでは、業務ユニットに追加を行った標準仕様V1.5を2007年9月にリリースし、現在はさらに仕様を追加した標準仕様V2.0の策定を進めている。

■図表1-1　地域情報プラットフォーム誕生の経緯



一方、上越市も参加している電子自治体アプリケーション・シェア推進協議会や、OSACが、全国の自治体に向けた地域情報プラットフォームの普及活動を行っている。

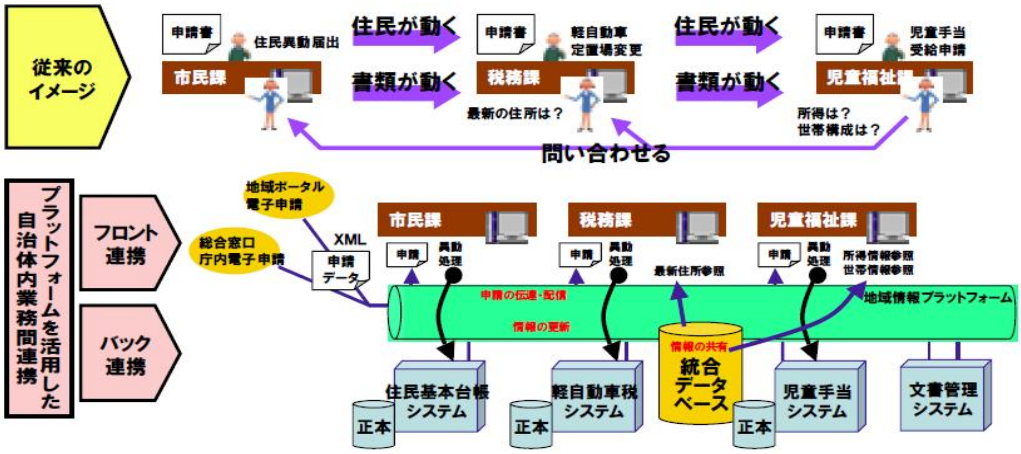
■図表1-2　電子自治体アプリケーション・シェア推進協議会とOSACの関係



※OSACホームページより引用
http://www.osac.info/gaiyou/index.html

地域情報プラットフォームは、図表1-3のとおり、自治体内における各種システム間の連携やデータの相互活用を推進する目的で構築されているが、将来的には自治体間の情報の受け渡しや、住民の引越しなどの際の民間も含めたサービス間連携も目指しているものである。

■図表1-3　地域情報プラットフォームを活用した業務間連携



※財団法人 全国地域情報化推進協会「地域情報プラットフォーム基本説明書Ver.2.0」より引用

１．１．５　上越市の状況

上越市においては、合併と併せたシステム調達に際して、もとよりオープンシステムにより基幹系システムを構築しており、他自治体とは若干状況は異なるものの、情報システムの最適化、情報の共有化・一元管理という点では、まだ見直す余地も数多く存在している。

特に、システム設計等がブラックボックス化され基幹系システムとその関連システムは特定ベンダーに依存せざるを得ず、結果的に自由な調達を阻害している面がある。また、大半のシステムがWindows環境に限定されているため、ライセンスやバージョン管理が必要であり、状況に応じてカスタマイズも発生する。さらに、ミドルウェアを指定しているものが多く、ミドルウェアのライセンスコストを含め、高価な運用・保守等の費用が必要であり、多くの問題が存在する。

そのため、上越市では次期リプレースを数年後に控え、「IT投資の適正化」や「市の知財・人材の継承」という視点、また更なる市民サービスの向上や県内ITベンダー育成としての、ベンダーロックインを廃した分割発注の視点等もあわせた、総合的な庁内情報システムの最適化やシステム調達の検討が必要であると考えていた。

１．１．６ 地域情報プラットフォーム普及の課題

これまで述べてきたように普及が期待される地域情報化プラットフォームであるが、その普及のカギを握る実装版の共通基盤にも、普及を阻害する問題が存在する。

（１）コスト面での課題

福岡基盤の成果物や、鳩ケ谷基盤はソースコードを含めて無償提供されているが、商用OSやミドルウェアを前提に構築されているため、ライセンスコストが発生し、当初導入コストが高いという問題が存在する。

（２）統合DBの公開実装が存在しない

地域情報プラットフォーム標準仕様V1.0では、統合DBについての実装仕様がない。鳩ケ谷基盤の統合DBは地域情報プラットフォームの仕様策定前に実装されている。

（3）情報開示の不足

一般に、OSSとして公開されているソフトウェアの中には、ドキュメントが限定されており、活用できるかどうかの判断をするためにもソースコードの解析が必要なものが存在する。そのためOSSを導入活用しようとする自治体等のユーザー側では、こうした点が普及の妨げになっている。

１．２　導入実証の目的

１．２．１　目的の設定

今回の導入実証は、こうした課題を解決し、自治体における地域情報プラットフォームの普及を促進することにある。

１．２．２　課題解決のための仮説

今回の導入実証事業は、大仮説として「前述の課題解決により、自治体におけるOSSの検討促進は可能である」を設定し、下記の３つの切り口で、大仮説をより細分化した仮説を設定することにした。

＜仮説１＞共通基盤コードをOSS化し、業務ユニットとの連携が図られれば、ライセンスコストの削減が可能である。

＜仮説２＞地域情報プラットフォーム標準仕様に基づく統合DBをOSSで実装すれば、地域情報プラットフォームの普及促進に有効である。

＜仮説３＞ドキュメント・ライセンスを整理し、それを公開すれば、自治体においてもOSSが選択肢となり得る。

１．２．３　成果の活用イメージ

（1）上越市における次期基幹システム構築時の共通基盤として活用

今回の実証事業で構築したシステムは、実証事業終了後も上越市において学童保育支援システムの基盤として運用が継続される予定である。

また、上越市が数年後に予定している次期基幹システム導入において、今回構築した共通基盤が果たして有効なのかどうか評価を行う予定である。

（2）他の自治体への普及

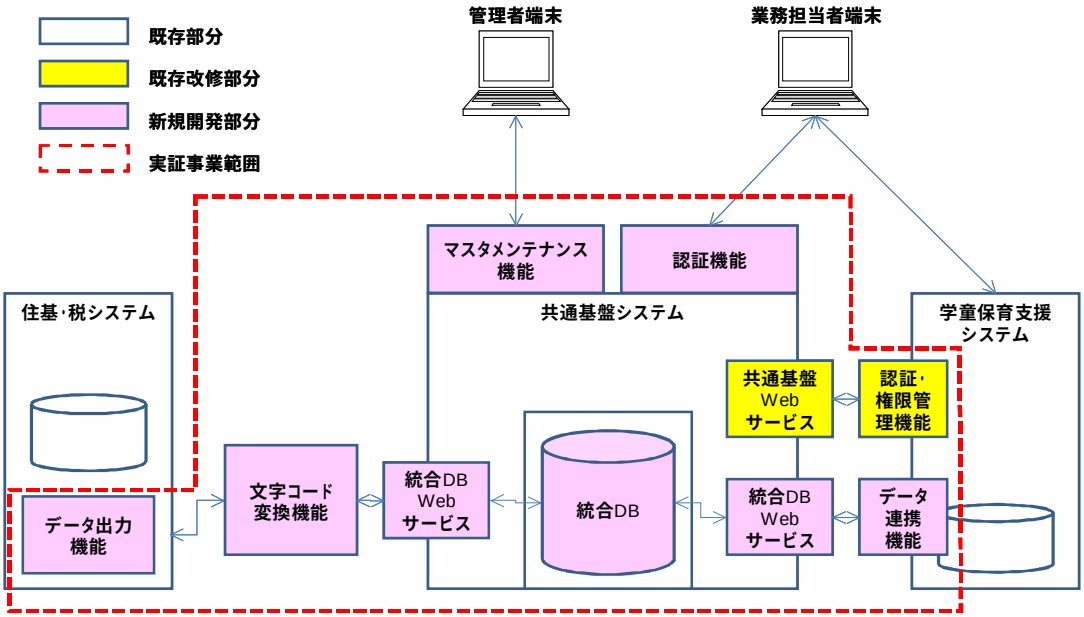
電子自治体アプリケーション・シェア推進協議会およびOSACを通じ、全国の自治体への普及が見込まれる。

１．３　導入実証の概要

１．３．１ 導入実証の全体構成

今回導入実証におけるシステム全体構成は下図のとおりである。

■図表１４　システム全体構成図



１．３．２ 実証内容

＜仮説１＞共通基盤コードをOSS化し、業務ユニットとの連携が図られれば、ライセンスコストの削減が可能である。

＜仮説１に関する実証＞

（１）鳩ケ谷基盤をベースに、OS/ミドルウェアをOSSベース（Linux/Apache/JBoss）に書き換える。

（２）今回の実証に必要な部分について鳩ケ谷基盤に含まれる商用ソフトウェアモジュールを排除し、代替機能をOSSで構築する。

（３）共通基盤が業務ユニットと連携可能であることを確認するため、以下の作業を行う。

①業務ユニットとしては、株式会社ＢＳＮアイネットの既存パッケージシステムである学童保育支援システムを選定し、これと共通基盤を連携するためのモジュールの開発を行う。

②開発した地域情報プラットフォーム対応版学童保育支援システムを上越市において試験運用し、実運用に支障がないことを検証する。

③なお、実証事業を行う上で必要であるが共通基盤に含まれない、マスタメンテナンス機能、認証連動機能は新たに開発する。

④構築した共通基盤において、Webサービスの性能試験を実施し、自治体での実運用に耐えることを検証する。このため、負荷分散構成での試験を行う。また、自治体規模に対応したハードウェア構成の見積もりを行う。

なお、学童保育支援システムを実証対象として採用した理由は、

・自社開発パッケージのため、ブラックボックスなく、仕様を把握していること

・一般的に流通しているWindowsベースのシステムであること

・VB.NETにより開発された業務ユニットがJ2EEで構築される地域情報プラットフォームの共通基盤と連携可能であることを示すこと

である。

＜仮説２＞地域情報プラットフォーム標準仕様に基づく統合DBをOSSで実装すれば、地域情報プラットフォームの普及促進に有効である。

＜仮説２に関する実証＞

（１）地域情報プラットフォーム標準仕様に基づく統合DBの論理設計を行う。

（２）（１）で行った論理設計を元に、オープンソースデータベースであるPostgreSQLを想定した物理設計を行う。

（３）（２）の物理設計を元に、PostgreSQL上に統合DBを生成するDDLを記述する。

（４）統合DBは自治体情報システムの中核となるため、実運用に必要な可用性及びセキュリティを実装する。

（５）データ提供側である基幹業務システムからデータを抽出し、プラットフォーム依存の文字コードを外字領域のマッピングを含めて地域情報プラットフォームに適したUTF-8に変換する文字コード変換システムの設計・実装を行う。

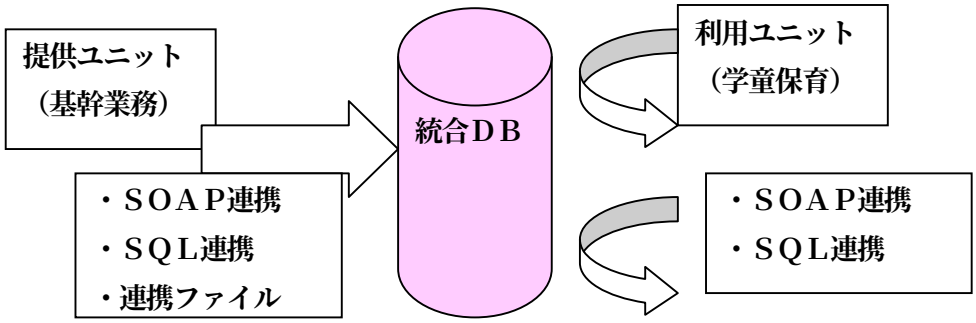
（６）構築した統合DBと業務ユニット側の連携のため、統合DB側に地域情報プラットフォーム標準仕様に基づくWebサービスインターフェースの実装を行う。

（７）統合DBの稼働実証のため、対象となる学童保育支援システム上に、統合DBからデータを取得するインターフェースモジュールの実装と、これに関連する既存モジュールの修正を行う。

（８）OSSで構築した統合DBが自治体の実運用に耐える性能を持つかどうかを検証するため、SQLおよびWebサービス経由での性能試験を行う。

統合DBと業務ユニットとの連携に関する概念図を図表1-5に示す。

■図表1-5　統合DBとの連携方式



＜仮説３＞ドキュメント・ライセンスを整理し、それを公開すれば、自治体においてもOSSが選択肢となり得る。

＜仮説３に関する実証＞

（１）＜仮説１に関する実証＞の（１）、（２）の作業とともに、共通基盤に不足する開発者向けドキュメントの整備を行う。

（２）上記の作業に加え、実証環境構築の際に共通基盤を導入する際の設定手順書を整備する。

（３）今回実証で利用するオープンソースについて、複雑なライセンス形態を整理して、ドキュメントとしてまとめる。

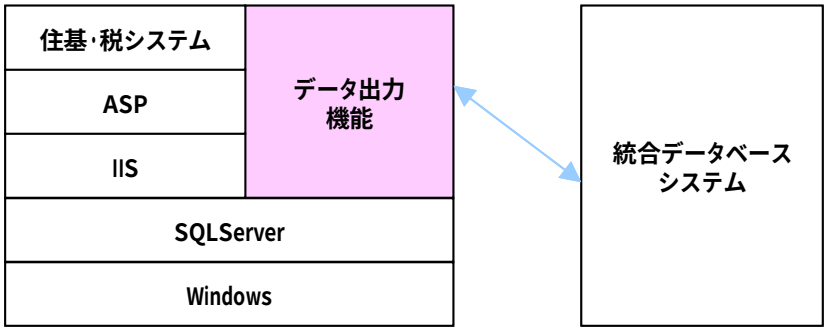
１．３．３　実証機能

基幹連携において、新規作成する機能は以下の通りである。

機能	概要
基幹系システム連携データ出力機能	基幹系システムより出力される「宛名」「住民基本台帳」「住民税」の日々の異動分のデータを連携ファイルデータとして、テキスト形式にて特定のフォルダへ出力する機能を実装する。
統合DBへの連携データ取込み機能	上記の機能により出力された連携データを統合DBへ取り込みを行う機能を実装する。
文字コード変換機能	基幹系システムより出力される異動データに関して、文字コードを変換(UTF-8)する機能を実装する。

データ提供側である基幹系システムのシステム構成図を図表1-6に示す。

■図表1-6　データ提供側システム構成



鳩ヶ谷基盤から利用する機能は以下の通りである。

機能	概要
ログ管理	業務システムにおける照会・更新におけるログ情報管理
システム連携	Web連携、DB連携、ファイル連携

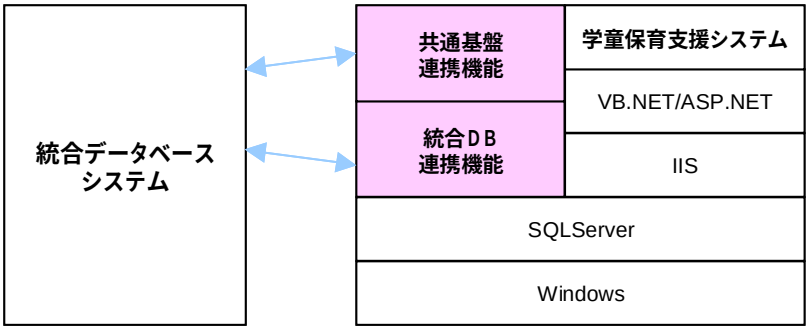
新規作成する機能は以下の通りである。

機能	概要
業務システム連携（共通基盤側）	共通基盤には業務ユニット向けWebサービスインターフェースを定義する。
共通基盤・統合DB連携（学童保育支援システム側）	学童保育支援システム側に、共通基盤及び統合DBの機能を利用するためのインターフェースを実装する。業務ユニットの機能に依存せず汎用性があり実装例として公開する部分と、学童保育支援システムに依存する部分（実証事業範囲外）に分離して開発を行う。
マスタメンテナンス機能	人事異動に伴う職員情報の変更など、学童保育支援システムで必要とする統合DB上のマスタ情報の更新機能を提供する。なお、既存の共通基盤では統合DB上におく情

機能	概要
	報は発生源で管理することを前提とするため、住基・住民税システムから統合DBに提供するデータは当該システムで更新することとし、統合DBに対するメンテナンス機能は提供しない。

データ利用側である学童保育支援システムのシステム構成図を図表1-7に示す。

■図表1-7　データ利用側システム構成



１．３．４　導入実証全般に関する補足

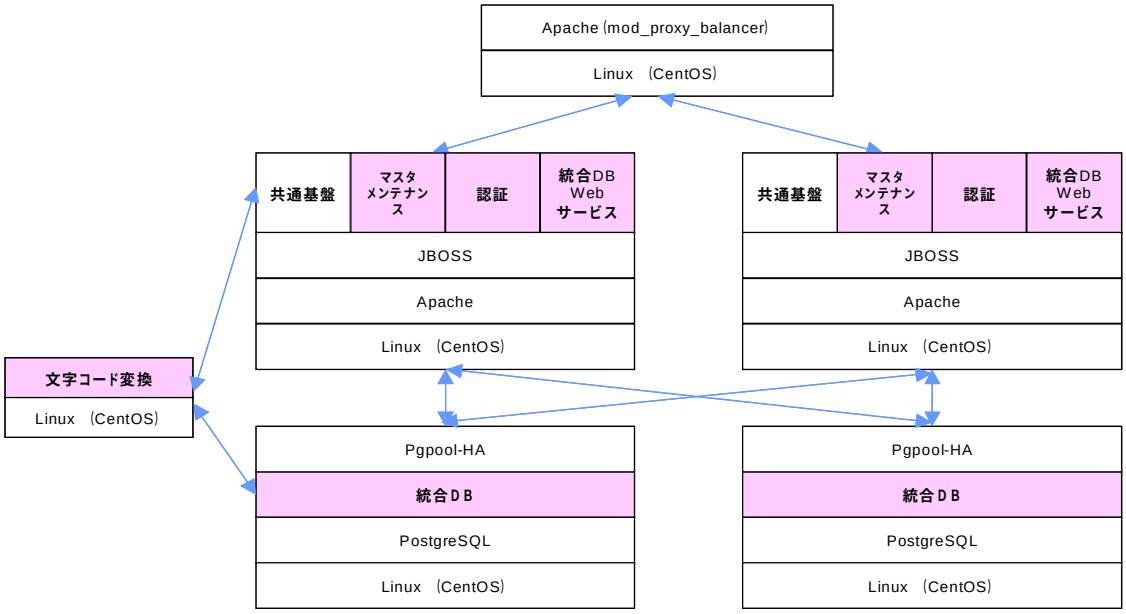
（1）導入実証におけるネットワーク環境

本実証に必要なネットワーク環境は、実証用に構築したギガビットイーサネットのネットワークで行う。ただし、担当課による実際の稼働試験は、上記のネットワークを上越市の基幹業務ネットワーク（LAN　100Mbps）に接続して行う。

（2）導入実証に必要なシステム構成

システム全体構成図を図表1-8に示す。

■図表1-8　共通基盤及び統合ＤＢのシステム構成図



各ソフトウェアの機能概要については、下表の通りである。

ソフトウェア	機能概要
CentOS	RedHat Enterprise Linuxと高い互換性を持つLinux OS
Heartbeat	サーバの二重化を実現するクラスタ化ソフトウェア
Apache	世界中で広く利用されているWebサーバ
mod_proxy_balancer	Apacheのプラグインとして負荷分散機能を提供するモジュール
JBoss	Javaベースのアプリケーションサーバ
PostgreSQL	DBサーバ
Pgpool -HA	PostgreSQLのフロントエンドとしてレプリケーション機能を提供する

本実証実験で使用するサーバに想定される要件は以下の通りである。

◇ ロードバランサ

各業務ユニット及びエンドユーザーを含む全てのクライアントアクセスの入口として動作する。Webサービスの負荷試験はこのサーバ経由で行われるため、システム全体のパフォーマンスに対するボトルネックとならないよう高いCPU性能と、多くのクライアントアクセスを処理する為に十分なメモリ容量を必要とする。

◇ Web/APサーバ

共通基盤及びデータベースに対するWebサービスインターフェースを提供する為の実行基盤としてJavaベースのアプリケーションサーバの利用を前提とする。

Javaベースのアプリケーションサーバは良好なパフォーマンスを得る為には強力なCPUを必要とし、且つ当サーバがシステム全体のボトルネックにならないようにするためには複数個のCPUを搭載する必要がある。

また、当サーバ上にて統合DBに対する共通インターフェースを実装することからも、当サーバに障害が発生した場合にはシステム全体がサービス不能となる為、サーバの二重化は必須である。今回の実証実験では、負荷分散構成をとることで効率的なハードウェア活用を前提とした負荷試験を行う。

◇DBサーバ

本システムの核となるサーバであり、業務ユニットを含む各クライアントが必要とする業務データは全て当サーバ上に存在する為、結果としてデータ照会要求は全て当サーバに集中する。

データアクセスの遅延がシステム全体のパフォーマンスに対するボトルネックとならないよう高いCPU性能と多くのメモリ容量を必要とする。特に、データベースのパフォーマンスを大きく左右するI/O処理の負荷を軽減する為には、より多くのメモリ容量を必要とする。

また、業務データが当サーバ上に保存されることからも、当サーバに障害が発生した場合にはシステム全体がサービス不能となる為、サーバの二重化は必須である。

◇ 文字コード変換サーバ

既存の住基・税システムからのデータ同期を実施するサーバとして動作する。住基・税システムより差分データを受信して、文字コードの変換を行い、データベースの更新を実施する為、この更新用アプリケーションが支障なく良好なパフォーマンスで動作するのに十分なCPU性能とメモリを必要とする。

本実証実験で使用するサーバの台数及びハードウェア構成は以下の通りとする。

□ サーバ台数

ロードバランサ	1台
Web/APサーバ	2台
DBサーバ	2台
文字コード変換サーバ	1台
合計	6台

□ ハードウェア構成

◇ ロードバランサ/文字コード変換サーバ　共通

CPU	デュアルコア 3.0GHz × 1
メモリ	2GB
ディスク	73GB SAS 1500rpm × 2
CD/DVD	DVD
NIC	1000BASE-T対応イーサネット × 2

◇ Web/APサーバ

CPU	デュアルコア 3.0GHz × 2
メモリ	2GB
ディスク	73GB SAS 1500rpm × 2
CD/DVD	DVD
NIC	1000BASE-T対応イーサネット

◇DBサーバ

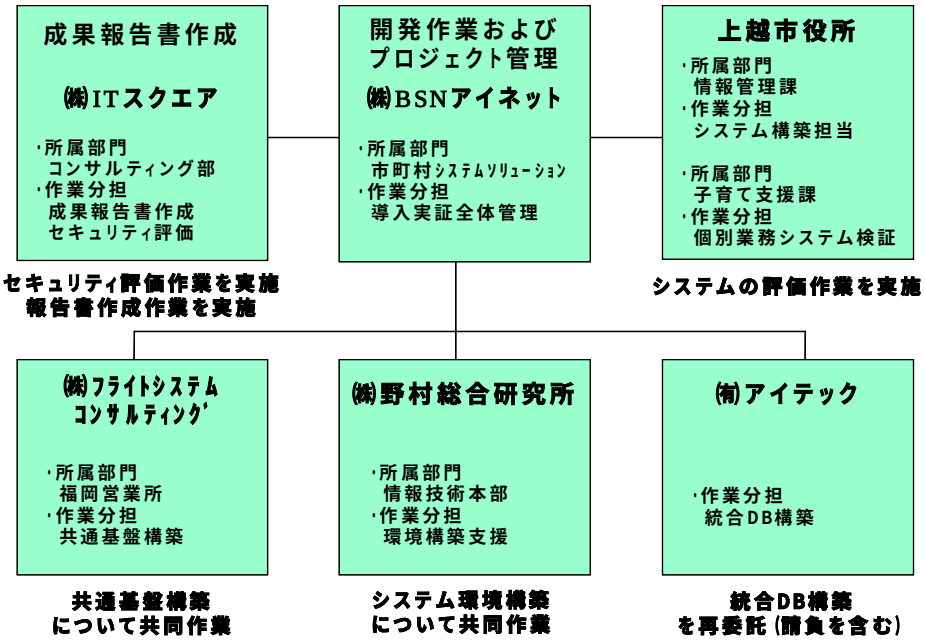
CPU	デュアルコア 3.0GHz × 1
メモリ	4GB
ディスク	73GB SAS 1500rpm × 2
CD/DVD	DVD
NIC	1000BASE-T対応イーサネット × 2

１．４　実施体制

１．４．１　実施体制スキーム

本導入実証全体のスキームを以下に示す。

■図表1-9　実施体制スキーム



１．４．２　開発体制

開発体制を以下に示す。

■図表1-10　開発体制

企業名	株式会社ＢＳＮアイネット
主な実施場所	〒950-0916　新潟市中央区米山２丁目５番地１

企業名	株式会社ITスクエア
主な実施場所	〒950-0916　新潟市中央区米山１丁目１１番地１１　昴ビル３Ｆ

企業名	株式会社フライトシステムコンサルティング
主な実施場所	〒812-0054　福岡市東区馬出２丁目１番地7号 福岡ことぶきビル２Ｆ

企業名	株式会社野村総合研究所
主な実施場所	〒240-0005　横浜市保土ヶ谷区神戸町134

企業名	有限会社アイテック
主な実施場所	〒950-0148　新潟市西区巻甲３１７４－３

１．４．３　自治体における実施体制

自治体における実施体制を以下に示す。

■図表1-11　自治体における実施体制

自治体	新潟県上越市　情報政策課
	新潟県上越市　子育て支援課
主な実施場所	〒943-8601　上越市木田一丁目一番三号

１．４．４　導入実証作業での自治体の役割

導入実証作業における上越市の役割としては、実証実験フィールドの提供と実証評価である。
その他、実証評価のための学童保育支援システムの運用評価を子育て支援課で実施していただいた。
実証作業項目ごとの自治体の役割を下記に示す。

■図表1-12　実証作業項目ごとの自治体の役割

番号	実証作業項目	自治体の役割
1	共通基盤に関する実証	・現行業務／システムの情報開示 ・システム機能要件の提示
2	統合DBに関する実証	・現行業務／システムの情報開示 ・統合DBの必要項目提示 ・システム機能要件の提示
3	導入実証全般	・実証評価 ・上越市側実証体制の情報開示 ・機器設置場所等の情報開示

１．５　導入実証スケジュール

１．５．１　スケジュール表

それぞれの実施スケジュールを以下に示す。

■図表1 -13　実施スケジュール

システム開発等 作業項目	2007/ 9月	10月	11月	12月	2008/ 1月	2月
１．物品調達						
２．共通基盤						
３．統合DB構築						
４．導入実証全般						
５．報告書作成						

１．６　普及活動･活用計画

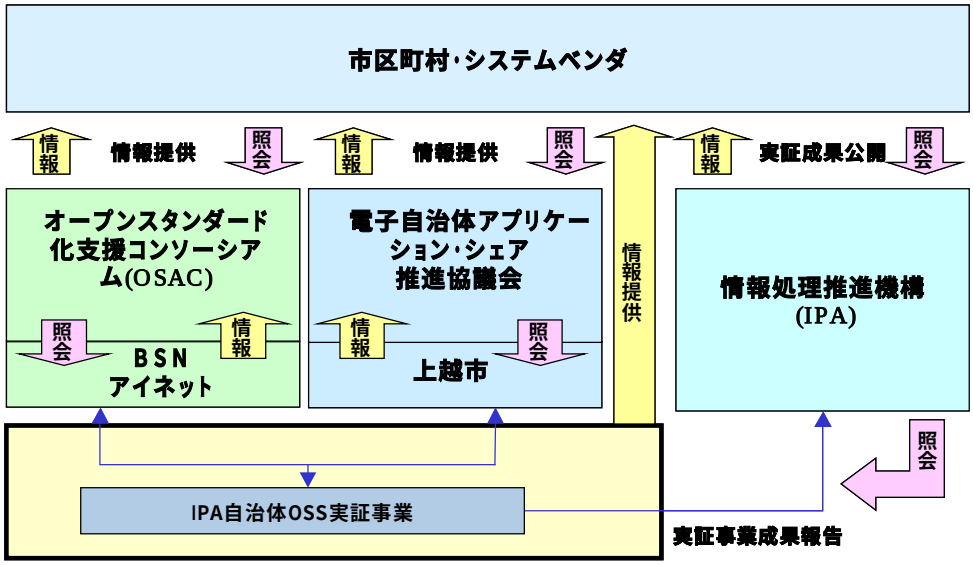
１．６．１　普及戦略

統合DBの導入ニーズは、情報化に積極的な市区町村において具体的な調達フェーズへと移りつつある。そのような中、本導入実証の成果を独立行政法人情報処理推進機構（以下「IPA」という）の広報活動を通じ、ソースコードの公開も含めて行っていくことで、当該市区町村のRFP作成における参考事例としての利活用が見込まれる。

本事業により作成された共通基盤や統合DBと、それらを実装するための開発者ドキュメントの公開に当たり、上越市および株式会社ＢＳＮアイネットが、参加しているそれぞれの団体への情報提供の役割を担い、各団体での検討事例として位置づけていただくことが可能となる。結果的に各団体の活動を通して、各団体と全国の市区町村・システムベンダーとの情報交換が活発になることが予想される。

IPA、上越市、株式会社ＢＳＮアイネットの大きく3つの機関の普及活動を基本とし情報提供を進め、市区町村への啓蒙を進める一方、今後県外も含め統合DBの調達案件に積極的に参加し、上越市に次ぐ事例を早期に創っていく。

■図表1 -14　情報提供の流れ



平成20年度においては、OSACの情報提供活動に当該実証内容も含めて頂き、広く市区町村への啓蒙を進めていくこととする。

また、情報提供による内容照会などには個別に対応を行い、当該実証内容の実証結果について、理解を深めていただき、具体的な調達フェーズへ進むためのサポートを行う。

１．６．２　メンテナンス・バージョンアップ・機能拡張等の計画

上越市における共通基盤システム、統合DB、基幹連携、学童保育支援システムの運用において、上越市職員向けの問い合わせ窓口（ヘルプデスク）は株式会社ＢＳＮアイネット側で設けるものとする。利用したOSSでのセキュリティパッチの対応、リビジョン／バージョン管理、バージョンアップ対応についても、株式会社ＢＳＮアイネット側で実施する。運用後に上越市側から出されるであろう要望事項については、システムとして対応するもの、上越市側の運用で対応するものを明確に区分し、必要に応じてシステムの機能拡張を実施していくこととする。

１．６．３　自治体における活用計画

上越市においては、現時点で本実証終了後も学童保育支援システムを実業務で利用する方針である。また、数年後の基幹システムリプレースに向けた検討のため、本実証環境の継続利用を考えている。

本実証環境にて調達手法の検討を継続して行い、導入コストや移行コストなどの検討内容に合わせたシステムの見直しなども視野にいれた運用を考えている。

２　事前準備

本章では、本実証を行うにあたっての事前準備作業について記述する。

２．１　自治体との事前調整

本実証においては、先ずテーマがあり、そしてそれを実施する候補となる自治体へのアプローチといった流れで進めていく事となった。

上越市については、株式会社ＢＳＮアイネットより税システムを導入頂いていること、また「電子自治体アプリケーション・シェア推進協議会」にも所属していることから、有望な候補として考えており、早々に企画概要について資料を作成し提案をさせて頂いた。

２．１．１　0SS実証の内容に関する事前調整

（１）0SS実証事業の提案

上越市においては、平成19年度から現システムで基幹系業務を運用しており、まだ次期リプレースに期間はあるものの、情報管理課としては、早期より次期リプレースに向けた検討を進めていく方針を持っており、今回提案はその方向性と合致したものとしてご理解を頂いた。

ただ、実証にあたっては個別業務システムとの連携が必須となることから、情報管理課以外に個別業務を主管する原課の協力が必要となる。

そのため、対象とする個別業務の選定を事前に行い、情報管理課提案時に、原課である子育て支援課に対しても、学童保育支援システムをあわせて提案し、実証事業終了後、当該成果によるシステムを子育て支援課よりご利用頂くことで、ご理解を頂くことができた。

来年度利用にあたっては、子育て支援課で別途予算措置を行って頂くこととした。

（２）実証後所有権の確認

本実証期間中及び実証終了後の所有権について整理を行うことが、一つ重要なポイントとなった。実証終了後も上越市が当該システムを運用する為には、所有権の確認により次年度に必要となる予算については、その措置を行う必要があり、先ずその整理を行った。

機器について実証期間はレンタルとし、終了後は上越市より別途予算措置を行って頂くこととした。

■図表2-1　所有権についての考え方

項目		フェーズ	経費拠出	所有権		
				上越市	BSN	IPA
統合データベース	ハードウエア	実証時	IPA			レンタル
		実証後		別途調達		
	システム	実証時				○
		実証後		OSSなので利用自由	○	
学童保育システム	ハードウエア	実証時	BSN		○	
		実証後			○	
	システム	実証時			○	
		実証後			○	
	I-F	実証時	IPA			○
		実証後			○	
基幹業務(税情報システム)	ハードウエア	実証時	上越市	○		
		実証後		○		
	システム	実証時			○	
		実証後			○	
	I-F	実証時	IPA			○
		実証後			○	

※I-F：インターフェース関連プログラム

２．１．２　実証期間の作業についての確認

上越市では、本事業において必要な情報の開示や機能要件の提示、評価への参加をお願いすることとなるが、各担当者の繁忙状況を鑑み、事前に参加頂くイベントをスケジュールに落とし、作業調整を進めて頂くようお願いをした。

２．２　システム開発・構築体制の確立

実証事業申請の企画段階より「地域情報プラットフォーム」や比較的大規模のOSSデータベース構築などの経験が必要なものについては、外部ノウハウの活用を意識しており、従前より参加していたOSACと調整を図りながら、開発の全体体制の検討を進めてきた。

また、内部体制については、実証項目により３チームを編成し、共同実施企業と連携を取りつつ事業を進めるようなかたちを作った。

２．２．１　共通基盤と業務ユニットインターフェース構築体制

社内的には、社内技術部門と自治体担当部門より構成を行ったが、鳩ヶ谷基盤による「地域情報プラットフォーム」の分析及びOSS化などのノウハウについては外部からの支援が必須と考えられた。

それらノウハウについては、実績のある企業より行って頂く必要があることから、先に鳩ヶ谷基盤と総務省の調査事業を行った実績のある株式会社フライトシステムコンサルティングにお願いする事とした。

２．２．２　統合DBと業務ユニットインターフェース構築体制

社内的には、業務系システムとの連携を行う部分であることから、自治体部門によりチームを編成した。データベース及びサーバ環境の構築において、株式会社ＢＳＮアイネットが新潟市で構築した「共通

データベース」の設計を基に、OSSによる設計が可能な県内企業を財団法人にいがた産業創造機構に相談したところ、唯一㈲アイテックより可能との返答があり今回協力を頂く事となった。

また、OSSによる高品位なシステム基盤の構築においては、OpenStandia（オープンスタンディア）の実績が高く、構築後のチューニングなどにも即対応が可能と判断し、株式会社野村総合研究所に協力をお願いした。

２．２．３　報告書作成体制

実際作業に着手している開発要員が、開発･構築を行いながら報告書を作成することは、先々困難と判断し、株式会社ＢＳＮアイネットの関連会社である株式会社ITスクエアとコンソーシアムを組みプロジェクトを進め、開発要員から口頭で伝えられる内容を報告書に纏め上げる形態とした。

さらに詳細な部分での精度を上げるべく、プロジェクトリーダー、サブリーダー、技術部隊の責任者等をアドバイザーとして体制を組んだ。

また、株式会社ITスクエアは、セキュリティについても担当した。

２．３　サービス運用体制の確立

上越市は、本実証事業終了後そのまま本格運用に移行する。このため、サービス運用体制を確立する必要があり、下記を検討することとした。

まず第一に市職員がOSSにて構築されたシステムを運用するにあたり、必要な研修を実施する。職員向けの研修に関しては、システム管理者研修とシステム操作研修の２種類を実施する。システム管理者研修は、共通基盤などOSSの中核にあたる部分について、正確な運用が保全されるように実施する。また、研修終了後、一定期間経過してのフォローアップ研修も実施する。

システム操作研修は学童保育支援システムの操作研修が中心になる。今回はじめて導入されるわけであるが、スムーズに運用できるようにする。

第二に、実証終了後のシステム本格運用のためのサポートを実施する。実証の過程で出てきた、以下の事項に対応できるようにする。

・OSS/ミドルウェアに関する問い合わせ、障害対応、改善要求対応
・共通基盤、統合DB等に関する問い合わせ、障害対応、改善要求対応
・学童保育支援システムの問い合わせ、障害対応、改善要求対応
・ハードウェア障害時の対応

なお、上記の問い合わせ等の対応については、ヘルプデスクを設置する予定である。

また、本格運用に際しては、株式会社野村総合研究所などのOSAC参加企業の協力を得て、職員が安心して利用できる運用マニュアルを用意する予定である。

３．システム開発

システム開発に関しては、大きくは以下の４つのシステムを開発した。

①共通基盤コードのフルオープンソース化

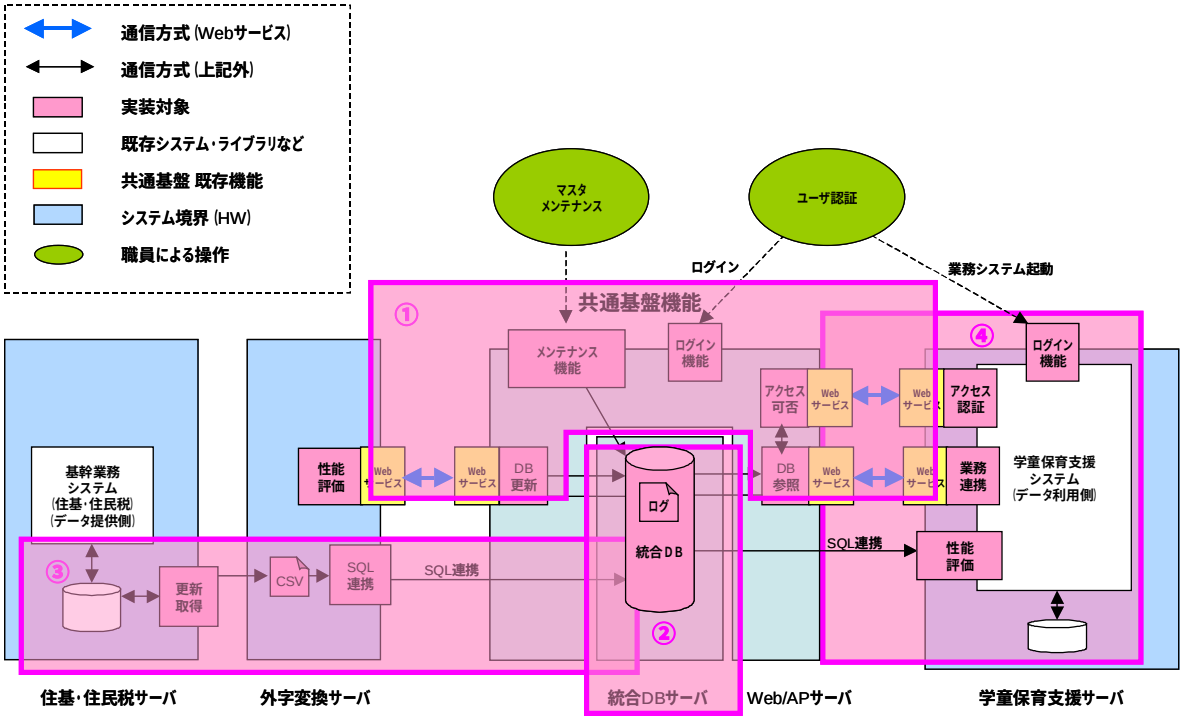
②統合DBをオープンソースで実装

③基幹業務システム（データ提供側）の連携機能

④学童保育支援システム（データ利用側）の連携機能

今回実証事業に係るシステム開発全体概要図を図表3 -1に示す。

■図表3 -1　システム開発全体概要図



３．１　共通基盤コードのフルオープンソース化

３．１．１　システムの概要

共通基盤のシステム基本構成は、OSはLinux、WebサーバはApache 、アプリケーションサーバはJBossとし、OpenStandia（オープンスタンディア）として導入実績が豊富な構成とした。

以下、作業別にシステム開発の内容を示す。

（１）鳩ケ谷基盤のフルオープンソース化

共通基盤には商用ソフトウェアに依存する機能が含まれている。この商用ソフトウェアのうち、商用ミドルウェアであるアプリケーションサーバのWebLogicはJBossに、データベースエンジンのOracleはPostgreSQLにそれぞれ代替し、共通基盤がこれらにより稼動できるよう再構築を行った。また、帳票出力といった商用モジュールについては、既存オープンソースによる代替はできないものの正規購入すれば共通基盤から再利用可能となるため、ソースコード上の商用モジュール呼出部をコメントアウトし、処

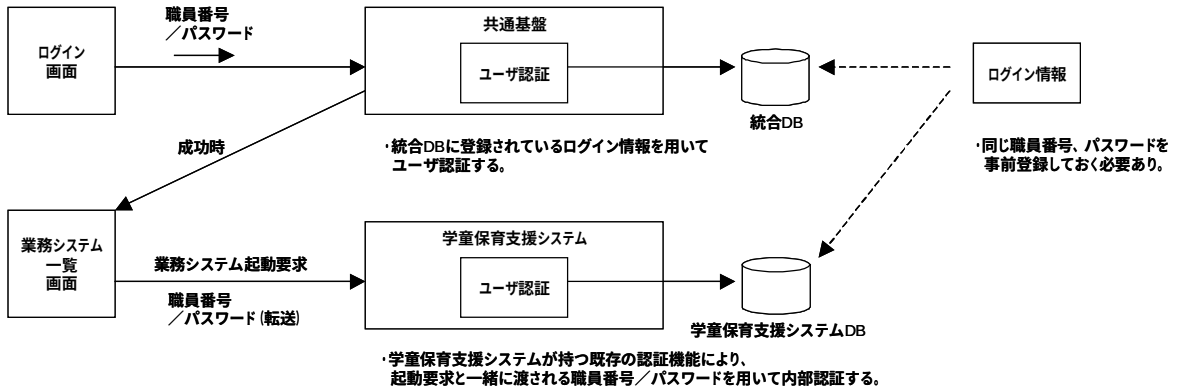
理ロジックを温存する方針とした。

（2）マスタメンテナンス機能、認証機能（ログイン）

マスタメンテナンス機能について、鳩ヶ谷基盤では機能が提供され、かつ、ソースコードが開示されているものの、当該機能が利用するデータベースのテーブル構造については一般に公開されていない。この為、学童保育支援システムで必要とする統合DB上の職員情報、およびアカウント情報の管理機能を新規作成した。

認証機能について、鳩ヶ谷基盤で機能が職員ポータルとして提供されているが、こちらも統合DBがあって初めて動作する機能である。しかし、統合DBに関する設計書等は公開されていないため、鳩ヶ谷基盤パッケージで提供するポータル認証機能の利用を断念することにし、簡易なユーザー認証機能を構築することにした。

■図表3-2　認証機能の処理フロー



（3）Webサービスインターフェース機能

共通基盤では、Webサービスにより連携する業務システムに対して、予め認可された業務システムのみ接続を許可する認証機能を提供している。これら機能を利用する場合、業務システムが送信するSOAPメッセージに必要な認証情報を付与する必要がある。鳩ヶ谷基盤ではこの作業を簡素化するため当該処理を行う組込機能を業務システム側に提供しているが、それらはJavaで実装されたもののみである。このため、学童保育支援システム向けに新たにVB.NET版の組込機能を製造し、鳩ヶ谷基盤上でJava-VB.NETといった異なる言語による連携を可能にした。

以上により、フルオープンソースソフトウェアでの共通基盤の構築を実現した。

３．１．２　システムの機能概要

共通基盤コードのフルオープンソース化に関するシステム機能概要を以下に示す。

（1）鳩ケ谷基盤のフルオープンソース化

分類	項目	№	説明
共通基盤の OSS 化	商用ミドルウェア対応	1	WebLogic を前提としたソースコードおよび設定ファイルを調査し、JBoss 上で稼動できるように再構築する。
		2	Oracle に依存したソースコード、および設定ファイルを調査し、PostgreSQL で稼動できるように再構築する。
	商用モジュール対応	3	帳票出力など OSS で代替不可な機能で本実証実験で必要とされない機能を調査し、再利用可能な状態で排除（コメントアウト）する。

（2）マスタメンテナンス機能、認証機能

分類	項目	№	説明
職員情報テーブル作成	職員情報テーブル作成	1	業務ユニット起動に必要な職員情報を格納するテーブルを提供する。
職員情報の作成	職員情報の照会、登録、更新、削除	2	職員情報の照会、および編集機能を提供する。
	権限設定(一般、管理者)	3	他者の職員情報に対する編集権限の付与を可能にする。
ログイン	ユーザー認証	4	簡易パスワード認証機能を提供する。
業務ユニット起動権限設定	起動権限の付与、取消し	5	業務ユニットの起動権限の編集機能を提供する。

（3）Webサービスインターフェース機能

分類	項目	№	説明
異言語間 Web サービス連携（Java -VB.NET 接続）	共通基盤に対する同期接続	1	共通基盤に対する VB.NET 版 Web サービス連携モジュールを提供する。
	共通基盤に対するアクセス認証	2	共通基盤に対する VB.NET 版のアクセス認証モジュールを提供することで、正規に許可された業務ユニットとしてのアクセスを可能にする。
業務ユニット向け Web サービス連携	汎用業務インタフェースの提供	3	業務テーブルに対する汎用データ格納インタフェースを提供する。
	学童育児支援システム向け業務インタフェースの提供	4	学童育児支援システムに特化したデータ照会インタフェースを提供する。
Web サービス連携性能評価ツール	学童育児支援システム向け SQL 連携	5	SQL 連携時のデータ照会性能評価ツールを提供する。
	学童育児支援システム向け Java -VB.NET 連携	6	Java -VB.NET 連携時のデータ照会性能評価ツールを提供する。
	学童育児支援システム向け Java -Java 連携	7	Java -Java 連携時のデータ照会性能評価ツールを提供する。
	住基・住民税システム向け Java -Java 連携	8	Java -Java 連携時のデータ格納性能評価ツールを提供する。

３．２　統合DBをオープンソースで実装

３．２．１　システムの概要

統合DBのシステム基本構成は、OSはLinux、データベースエンジンはPostgreSQLとし、PostgreSQLのレプリケーション機能を提供するPgpool （Pgpool -HA）を導入した。対象とする業務ユニットは「住民基本台帳」、「個人住民税」、「福祉」の３業務とした。

統合DBのテーブル設計作業では、APPLICより公開されている「自治体業務アプリケーションユニット標準仕様V1.0」（以下「APPLIC標準仕様」という）を基にして、各業務ユニットのテーブルのインターフェース仕様を精査することで、テーブル設計を行った。またテーブル規約を提示し、今後のテーブル設計を進める上で指針となるものを示すことにした。

本実証では統合DBに接続するユーザーは、管理者用、共通基盤からの接続用、直接SQLからの接続用の３つのユーザーに分類してユーザー設計を行った。管理者以外は必要な権限以上の権限はユーザーには付与しないこととし、共通基盤からの接続・直接SQLからの接続に必要なユーザーにはSELECT文以外のコマンドは実行できないこととした。

なお、上記で設計された、ユーザー、テーブル、権限を自動生成するためのスクリプトを作成した。

３．２．２　システムの機能概要

統合DBをオープンソースで実装に関するシステム機能概要を以下に示す。

分類	項目	№	説明
統合DB構築	住民基本台帳マスタ	1	住民の転入・転出・転居・出生・死亡等の異動、照会や証明書の発行・通知書の出力等の情報を格納している。　※APPLIC標準仕様に基づき設計
	個人住民税マスタ	2	個人住民税の課税対象管理・資料の管理・賦課・統計処理等を行うための情報を格納している。 ※APPLIC標準仕様に基づき設計
	福祉マスタ	3	生活相談受付、保護申請審査、支給管理、統計処理のための情報を格納している。 ※APPLIC標準仕様に基づき設計
	統合DB環境構築スクリプト	4	統合DB環境（データベース、ユーザー、テーブル、権限）を自動生成する。

３．３　基幹業務システム（データ提供側）の連携機能

３．３．１　システムの概要

基幹業務システム（データ提供側）の連携機能について、システム開発の内容を以下に示す。

（１）統合DB連携用データ出力機能

基幹業務システムから「住民基本台帳」、「個人住民税」の差分の更新データを抽出し、そのデータをFTP転送により文字コード変換システムにCSV形式で送信する。連携用データは統合DBのレイアウトに合わせた形でCSV出力し、APPLIC公開の「アーキテクチャ標準仕様V1.0」の要件を満たすような設計とした。本連携用データ出力機能は既存システムへの改修を行わず、VisualBsic6.0にて単体のEXE形式にて作成されており、Windowsのタスクからスケジュール実行されるものである。

（２）統合DBへの連携取りこみ機能

基幹業務システムから出力された更新情報（CSV形式）を統合DBへ取り込むプログラムを作成する。取り込み機能はPerlにて実装され、Linuxのcron機能にて定期的にスケジュール実行される。

設計で、統合DBのレイアウトが今後、列の変更・追加されることを考慮し、統合DBのテーブルや列の定義情報を設定ファイル(XML形式)でもつことにより、本連携用データ出力機能のプログラムを修正せずとも設定ファイルを変更することで対応できるようにした。

（３）文字コード変換機能

統合DBへのデータ取り込み機能で、設定ファイル上にある外字変換パラメータを「Y」にすることにより、外字の変換テーブルの規則に従い、対応のUnicode（UTF-8）に外字変換される。外字の変換テーブルはテキスト形式で変換の外字の対応表が記述されている。ここで外字変換をクライアントで実装せずサーバーでの機能としたのは既存システム側での改修を必要とせず、また複数のシステムを統合する際に有利と考えたからである。

（４）ログ管理機能

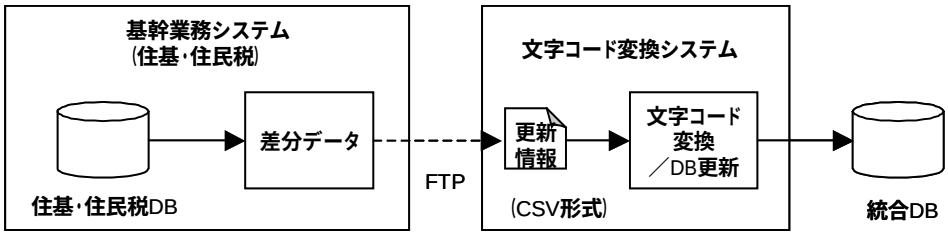
定期的に統合DBへデータ取り込み処理が行われた際に、正常・失敗の処理結果また確認用として取り込みされた更新件数を連携システムからログファイルとして出力される。ログ出力機能に関しては、出

力する内容のレベルをパラメータにて制御でき、パラメータを変更すると、より詳細な情報を出力することができる設計とした。更新時に発行されたSQLレベルでの出力も可能であり、エラー発生時に問題となるSQL文を参照し原因特定をすることができる。

（5）メッセージ通知機能

設定ファイルに通知先のメールアドレスを登録すれば、取り込みエラー発生時にメール通知を管理者に行うことができる。

■図表3-3　基幹系システムのデータ出力から統合DBへのデータ取り込みまでの処理フロー



なお、上記のシステム構成を採用した理由は以下の通りである。

・業務ユニット更新時に、統合DB側の更新も同時に行う即時連携を採用すると、データ提供側の業務ユニットのデータ更新プログラムの根幹に改修が必要となる可能性があるため、汎用性がなくなるおそれがある。

・即時連携を採用した場合、分散DBの更新処理は制御が複雑になり、トラブル発生時のデータリカバリーに困難が伴う。

・連携データをCSVで受け渡すことにより、レイアウトの変更が生じた場合も、統合DBへの更新プログラムを修正する必要がない。また、データ提供側の業務ユニットのプラットフォームへの依存度が軽減され、汎用機等の場合であっても本成果物を利用できる可能性が出てくる。

３．３．２　システムの機能概要

基幹業務システム（データ提供側）の連携機能に関するシステム機能概要を以下に示す。

（1）統合DB連携用データ出力機能

分類	項目	№	説明
連携用データ出力機能	住民基本台帳データの連携	1	基幹業務システムで更新された住民基本台帳の情報を判断し、CSV形式にて更新情報を出力する。
		2	出力された更新情報をFTP転送により文字コード変換システムに送信する。
	個人住民税データの連携	3	基幹業務システムで更新された個人住民税のデータ情報を判断し、CSV形式にて更新情報を出力する。
		4	出力された更新情報をFTP転送により文字コード変換システムに送信する。

（2）統合DBへの連携取りこみ機能

分類	項目	№	説明
連携データ統合DB取込み機能	連携データ統合DB取込み	1	基幹業務システムから出力された更新情報(CSV形式)を統合DBへ取り込む。
	外字変換機能	2	設定ファイルにて外字変換パラメータを「Y」にすることにより、外字変換テーブルを参照しshift jisからUNICODEへの変換が可能となる。
	ログ出力機能	3	連携データ統合DB取込み機能のシステムログを出力する。
	メッセージ送信機能	4	連携データ統合DB取込み機能が異常終了した場合に、エラー情報をメールにて通知する。

３．４　学童保育支援システム（データ利用側）の連携機能

３．４．１　システムの概要

学童保育支援システム（データ利用側）の連携機能について、システム開発の内容を以下に示す。

（1）共通基盤連携機能

ログイン機能は、新規開発した簡易な認証機能を使い、職員コードおよびパスワードで認証を行う。また、権限管理連携機能は、学童保育支援システムの業務メニュー制御機能で、ログインした職員の業務権限より、業務メニュー表示の制御を行うものである。

（2）統合DB連携機能

統合DB連携機能は次の４つの機能を実装する。宛名情報検索機能は、氏名カナ、住所、生年月日より、統合DBの宛名情報から検索を行い、検索結果一覧の表示を行うものである。次に宛名情報参照機能は、検索結果一覧より選択された個人の世帯コードより、該当する世帯の詳細情報の表示を行う。また、税情報取り込み機能は、選択された個人が所属する世帯の住民税情報を取得するものであり、福祉情報取り込み機能は、選択された世帯の生活保護情報を取得する。

３．４．２　システムの機能概要

学童保育支援システム（データ利用側）の連携機能に関するシステム機能概要を以下に示す。

分類	項目	№	説明
共通基盤連携機能	業務ユニットログイン機能	1	共通基盤より連携された職員IDで、業務ユニットシステム内の職員テーブルを検索し、合致する場合は業務メニューの表示を行う。
		2	共通基盤より連携された職員IDで、業務ユニットシステム内の職員テーブルを検索し、合致しない場合は業務メニューの表示を行わない。
	権限管理連携機能	3	共通基盤よりログインした職員IDの権限に応じた業務メニューの表示を行う。
統合DB連携機能	宛名情報検索機能	4	氏名カナ、住所、生年月日により、統合DBの宛名情報から検索を行い、入力された検索条件に合致するレコードの一覧の表示を行う。
		5	検索結果の件数が、設定されている最大検索数を超えた場合は、検索条件の絞り込みを促すメッセージの表示を行う。
	宛名情報参照機能	6	表示された一覧画面で選択した個人の世帯コードより、その世帯に属する世帯員情報の一覧の表示を行う。
	税情報取り込み機能	7	選択した世帯の世帯員の所得割データが取得を行う。
		8	表示された世帯員情報より、選択された個人の所得割データの表示を行う。
		9	表示された世帯員情報より、選択されていない個人の所得割データのマスキングを行う。
	福祉情報取り込み機能	10	統合DBより生活保護の情報を取得し、選択された世帯が生活保護世帯だった場合、生活保護チェックボックスに自動的にチェックを行う。
		11	統合DBより生活保護の情報を取得し、選択された世帯が生活保護世帯でない場合、生活保護チェックボックスにチェックを行わない。

４．　導入実証結果

前述の１．３において仮説に対しての実証概要を記載しているが、本章では、実際に実証作業を行った結果をまとめ、考察を加えたものを記載する。

仮説１に関する検証は以下の通りである。

（１）共通基盤コードのフルオープンソース化の検証

（２）共通基盤と業務ユニットとの連携の検証

（３）Webサービス連携の性能評価の検証

仮説２に関する検証は以下の通りである。

（１）地域情報プラットフォームに基づく統合DBの実装の検証

（２）基幹業務システムデータ連携機能・動作の検証

（３）統合DBの性能評価の検証

仮説３に関する検証は以下の通りである。

（１）開発者向けドキュメント・導入手順書等の整備の検証

（２）ライセンスの整理・調整の検証

導入実証全般に関する検証は以下の通りである。

（１）システム環境構築等インフラ整備の検証

以下順次検証結果を示すことにする。

４．１　共通基盤コードのフルオープンソース化の検証

４．１．１　実証概要

共通基盤コードのフルオープンソース化に関する検証を以下の内容で実施した。

（１）鳩ケ谷基盤のフルオープンソース化

鳩ケ谷基盤をベースに、商用OS/ミドルウェアをOSS（Linux/Apache/JBoss）に書き換え、動作検証した。

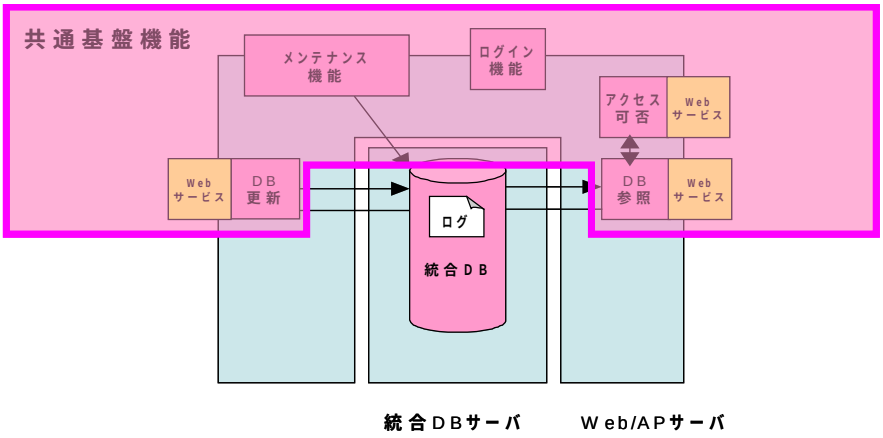
（２）商用モジュールの排除

帳票出力といった商用モジュールについては、今回実証事業の範囲外であったため、ソースコード上の商用モジュール呼出部をコメントアウトし、排除した。

（３）マスタメンテナンス機能、認証機能の実装

追加開発機能として、実証事業を行う上で必要であるマスタメンテナンス機能と認証機能を新たに開発し、動作検証を実施した。

■図表4-1　共通基盤コードのフルオープンソース化に関する検証



４．１．２　実証結果

鳩ヶ谷基盤の商用OS/ミドルウェア依存部分の改修を行い、OSS環境（Linux/Apache/JBoss/PostgreSQL）で動作することを検証した。

鳩ヶ谷基盤の中には、上記の商用OS／ミドルウェア以外にも、開発ベンダーの商用ポータル製品への認証機能等の依存部分や、商用印刷モジュールへの依存部分が含まれていたため、これらの排除を行った。

マスタメンテナンス機能、認証機能については、簡易な代替機能を新規開発した。

以上の作業により、共通基盤コードのフルオープンソース化を実現することができた。

４．１．３　考察

OSSとして公開されているソフトウェアの中には、ドキュメントが未整備なものや、一部機能を商用製品に依存しているため、ライセンス購入やサポート契約が必要になることもあり、OSSを導入活用する自治体等のユーザー側では、こうした点について十分な調査を行う必要がる。

一方、自社製品をオープンソースとして公開する側からすると、ビジネスモデルとして、導入サービスやサポートサービスで収益を上げる必要があることが多く、戦略上限定的な情報開示とならざるを得ないことは理解できるが、ビジネスの拡大のための普及活動としての側面を考えて、情報の提供に取り組む必要があると考える。

なお、商用製品に対応するOSS製品の留意点および評価に関して以下に示す。

■図表4-2　商用製品に対応するOSS製品の留意点・評価

項目	サーバ	商用製品	OSS製品	OSS導入の留意点	評価
可用性	APサーバ（負荷分散）	NetScaler	Apache ＋ mod_proxy_balancer	振り分け先となるAPサーバがTomcatまたはJBoss以外の場合、アプリケーションによっては正常に動作しない危険性がある（Webアプリケーション上でのセッションが維持できない危険性がある）	OSS版は振り分け先となるAPサーバを選ぶ（つまりTomcat/JBoss以外の場合には注意が必要）点に関して多少汎用性に乏しいところはあるが、一方でAPサーバがTomcat/JBossの場合には非常に適しているともいえる。それほど複雑でないルールに従ってJBoss上で実行されるWebアプリケーションへの負荷分散を行う場合には、処理が軽く高速に動作するOSS版は十分利用に値すると考えられる。
	DBサーバ（クラスタ）	Oracle ＋ RealApplicationCluster	Pgpool ＋ Heartbeat	データの同期を行う為のレプリケーションの方式が「同じ更新処理を複数回実行する」という方式の為、更新系トランザクションだけで比較すると、純粋にクラスタを構成するDBサーバが増えれば増えるほど一回のトランザクションの処理時間が遅くなってしまう。 参照系については、台数が増えても極端に遅くなることはない。	OSS版は単純な機能の組み合わせによるクラスタ化の為、機能面だけで見ると少々物足りなく見えるが、反面構成がシンプルな為構築しやすいと考えられる。高価なHWや複雑なソフトウェア上の依存関係、制約等に縛られる事が無い為、比較的容易にDBの二重化が実現できる。台数の増加によるスケーラビリティの向上はあまり見込めないが、耐障害性の向上という観点で見れば2台のDBサーバによるクラスタで十分な為、更新系トランザクションの遅さもある程度までは許容可能であると考えられる。
運用	APサーバ	WebLogicAS	JBossAS		通常運用のみで考えると、OSS版でも遜色ないと考えられる
	DBサーバ	Oracle	PostgreSQL	更新系トランザクションが多い場合には、データベースの肥大化と適切なパフォーマンスを保つためにも定期的なVACUUM処理が必要である。 VACUUM中はパフォーマンスが悪化すると言われている為、処理のタイミングについては注意する必要がある。	OSS版は一般的なRDBMSでは必要のない処理を行う必要があるが、一方で一般的なRDBMSが定期的なメンテナンス処理として必要なタスク（インデックスの再構築や統計情報の更新等）をこの一つの処理でまかなう事が可能な為、VACUUMだけ実行すればよいというメリットもある。オンライン中のメンテナンス処理には適さないが、逆に多くのシステムが業務時間外にメンテナンス処理を行う事が多い為、VACUUM中のパフォーマンス悪化が大きな問題になることはあまりないと考えられる。
監視	APサーバ	WebLogic	JBossAS	商用に多く見られるような、きれいな画面（グラフ化、要点の抽出等）での監視は、OSS版では行えない。	わかり易い監視GUIはあるにこしたことはないが、運用が始まるとそれらの画面を見る事はほとんどない。障害発生時などもテキストベースのログファイルを追う事が多いため、結局GUIを利用する機会というのは初期の導入時やシステムテストの時のみであることが多い。 一般的に運用が始まった後のサーバの監視は別途専用のシステムを使用して自動化して行う多い為、商用に付属するGUIを監視に使用することはほとんどないと考えられる。
	DBサーバ	Oracle	PostgreSQL	同上	同上
システム評価ツール	APサーバ	LoadRunner	JMeter	ネイティブなプロトコルを使用するソフトウェアやWindowアプリケーション等の負荷テストは行えない。	一般的なWebアプリケーションの負荷テストで使用するのであれば、機能的には十分である。

	DBサーバ	LoadRunner	pgbench	PostgreSQL以外には使用できない	
セキュリティ	APサーバ	WebLogic	JBossAS		Webアプリケーションのセキュリティは APサーバ自身の機能というよりAPサーバ上で実行するWebアプリケーションでの実装に大きく依存する。 そういった意味では、商用版、OSS版共にJavaのライブラリとして提供されている多くのセキュリティ機能をWebアプリケーション側が組み込むことは可能な為、この点ではOSS版も商用版と遜色ないと考えられる。
	DBサーバ	Oracle	PostgreSQL	OSS版でのデータの暗号化はアプリケーション側での対応が必須であり、またかなりのパフォーマンス悪化を招くため注意が必要。 監査ログについても全てのログを取るか、取らないかのいずれかの選択になる為、取得する場合には膨大な量のログが記録される危険性がある。	監査ログについては商用版の方が柔軟性があるが、一般的にWebアプリケーションではDB側で記録される監査ログに記載されるユーザと実際に処理を行っているエンドユーザーとを紐づける事はかなり難しい為、DBの監査ログを取ることはあまり多くなく、アプリケーション側のログ機能を充実させて監査ログとする事の方が多い。 同様の理由で、ユーザ毎のデータマスク処理についてもDB側の機能ではなくアプリケーション側で実装されることの方が多いと考えられる。 また、暗号化についてもアプリケーション側での対応を必要とされる事が多く、商用といえどもそう簡単に導入できるものではないと考えられる。 その為、商用ではセキュリティ機能が充実しているように見えてもそれらをほとんど使用していないのが実情だと考えられる。
障害復旧	APサーバ	負荷分散装置 ＋ WebLogic	mod_proxy_balancer ＋ JBossAS		障害から復旧したAPサーバを手動/自動で振り分け先に追加させることができる為、OSS版も商用と遜色なく利用可能であると考えられる。
	DBサーバ	Oracle ＋ RealApplicationCluster	Pgpool ＋ PostgreSQL	障害復旧時には必ずクラスタを構成する全てのデータベース（PostgreSQL）を停止させる必要がある。	完全に無停止でのサービスを提供する必要があるシステムではOSS版の利用はむずがしいが、そうでない場合には通常障害復旧時にはある程度のシステム停止が許容されることが多いと考えられる。そのようなシステムの場合、OSS版の制約は大きな問題にはならないと思われる。

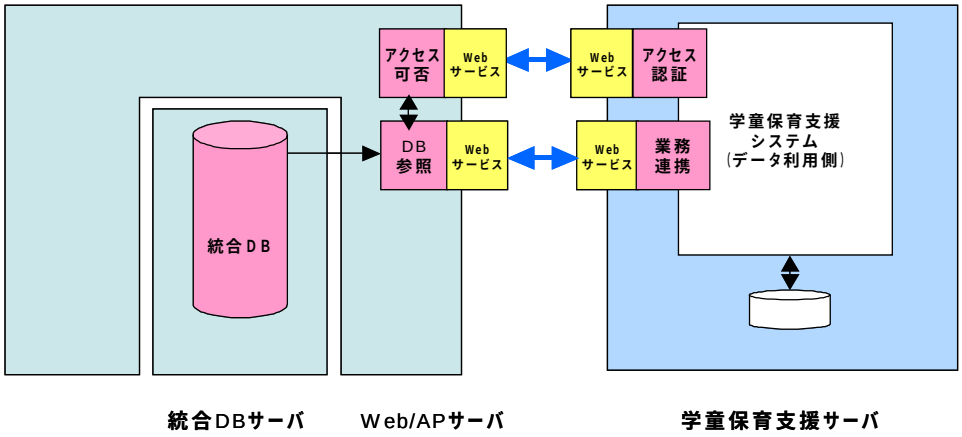
純粋にフルスペックで比較すると機能面では商用製品の方が確かに優れているが、上記のように実際のシステム構築においては、商用製品の機能をフル活用することはほとんどないと言ってよく、基本的な機能の組み合わせによってシステムを実装している事が多い。従って、基本的な機能のみで比較するとすればOSS製品は商用製品と遜色ないと考えることができる。

４．２　共通基盤と業務ユニットとの連携の検証

４．２．１　実証概要

共通基盤と業務ユニットとの連携に関する検証を以下の内容で実施した。
（1）今回の実証で構築した共通基盤と業務ユニットである学童保育支援システムとをSOAP連携させるための、共通基盤側のWebサービスインターフェース機能を開発し、動作検証した。
（2）業務ユニットである学童保育支援システムを採用し、これを共通基盤と連携するようモジュールを開発し、動作検証を行った。

■図表4-3　共通基盤と業務ユニットとの連携に関する検証



４．２．２　実証結果

本実証で利用した共通基盤のWebサービス連携機能は、鳩ヶ谷基盤のベースとなった福岡基盤に含まれる機能である。この機能では、Webサービス連携時のアクセス認証を簡易に実現するため、クライアント側に組み込むべき機能を提供している。なお、現状、この機能部品はJavaで記述されたもののみの提供であり、他言語版は提供されていない。一方、本実証実験で共通基盤と連携する業務ユニットとなる学童保育支援システムは、Windows上のWebアプリとしてVB.NETで構築されている。

このため、Javaで実装されている処理内容を解析してVB.NETで独自に実装しなおすとともに、学童保育支援システム側でのログイン機能やデータ取得機能の修正を行うことで、異言語間（Java-VB.NET）でも共通基盤と業務ユニットの連携が可能であることが実証できた。

４．２．３　考察

共通基盤でシステム間連携に使用されているSOAPによるWebサービスは、当初から実装言語を問わない分散システム通信仕様として設計されており、APPLICが地域情報プラットフォームの標準規格としてSOAPによるWebサービスを採用しているのもこの視点によるものと思われる。この実証においても、SOAPによるWebサービスであればAPI仕様書から異言語間通信を実装するのはそれほど困難ではないと認識していた。

もともとSOAPは異言語間の相互運用性を考慮した仕様であったため、Java-VB.NET間通信の実現もそれほど困難は伴わないと予想していた。しかしながら、共通基盤のWebサービス連携機能を実装する場合、

モジュールの詳細仕様が不明であったため、同機能のクライアント側モジュールをVB.NETで実装するために、送受信されるSOAPメッセージをモニタリングしながら、詳細な分析を行う必要があった。なお、現在のAPPLIC仕様は1.0であるが2.0の仕様も策定中である。APPLIC2.0では1.0でオプションとなっていたWebサービス上のセキュリティ（WS-Security）や信頼性（WS-Reliability）を確保する機能が必須となる可能性が高い。これらもSOAPと同様、仕様上、異言語間の相互運用性は想定されているものの、相互運用性を確保できるかは当該仕様の実装モジュールに左右されることが多い。共通基盤のように、技術規格としては異言語間、異なるプラットフォーム間の連携を想定した製品であっても、実際には異言語間、異プラットフォーム間の動作検証が十分行われているとは限らず、また、実装がそれを意識したものになっているという保証はないため、導入にあたっては十分な調査と検証を行う必要がある。

次に、統合データベースを用いたデータ連携であるが、今回、上越市での動作検証では、APPLIC標準仕様で規定している「住民基本台帳ユニット」等の標準インターフェースを使用せず、学童保育支援システムに特化したインターフェースを使用している。これは次の理由による。

・標準インターフェースが汎用のため、学童保育支援システムには不要の項目が多数含まれており、オーバーヘッドが大きい。
・学童保育支援システムで必要な情報を取得するには、複数の標準インターフェースによる呼び出しを行う必要があり、レスポンス低下の原因となる。

今回の実証にあたり、事前調査で行った測定結果を下記に示す。

【事前調査の前提条件】

評価環境は、実証環境ではサーバOSはLinux(CentOS)となるが、開発作業への着手も考慮するとサーバ搬入・サーバ構築を待つ時間がない、統合DBへ構築したテーブル項目を全て取得した場合/サブセットで取得した場合での速度差を評価できれば良いという理由から暫定措置としてWindows環境にてサーバを稼動させて評価を行った。

○評価環境（クライアント側）
・Windows XP SP2
・J2SE 1.4.2 update16
・評価ソースコード （後述:AxisClientEval）

○評価環境（サーバ側）
・Windows XP SP2
・JBoss 4.2.1GA
・JavaSE 5.0 update12
・評価ソースコード （後述:AxisServerEval）

サーバ／クライアント側で利用するJavaバージョンが異なる理由は以下の通りである。

(１)JBoss4.2.1GAをJavaSE5.0で動作させるのが本実証上、正しい前提であるため、サーバ側はJavaSE5.0で稼動させる。
(２)共通基盤（福岡基盤）ではSOAP通信用ライブラリとしてAxis1.1が利用されているがAxis1.1はJavaSE5.0での動作保証がされない。このため、動作が保証されるJ2SE1.4.2にてクライアント側を稼動させ、評価値を取得する。

【評価方針】

クライアント／サーバ側間メッセージ送受信における通信速度評価を行う。

サーバ側は次の前提で定義する。

(１)評価対象は住民基本台帳（jumin）とする。

(２)属性（DBカラム相当）は文字列とし、全項目に"1234567890"(10バイト長)を設定する。

(３)クライアントに返却するデータはサーバ側でダミーを生成する。DBアクセスは行わない。

(４)返却データは２種類（フルセット／サブセット）を用意する。

(５)フルセットは、juminテーブルが持つ全カラムを対象とする。

(６)サブセットは、juminテーブルが持つデータの一部とし、学童保育支援システムが処理上必要とするカラムのみに限定する。

(７)クライアントからの要求に合わせて、返信するデータ数を変更可能にする。

クライアント側は次の前提で定義する。

(１)サーバに対する取得データ件数を指定可能とする。検索条件などその他の情報は送信しない。

(２)サーバに対する処理要求の送信間隔（インターバル）を指定可能とする。

(３)サーバに対する呼び出し回数を指定可能とする。

【評価結果】

次の前提で評価を行った。

①１実行あたり、100回データ取得処理を実行する。

②データ取得件数は100,300,600,1000で計測する。

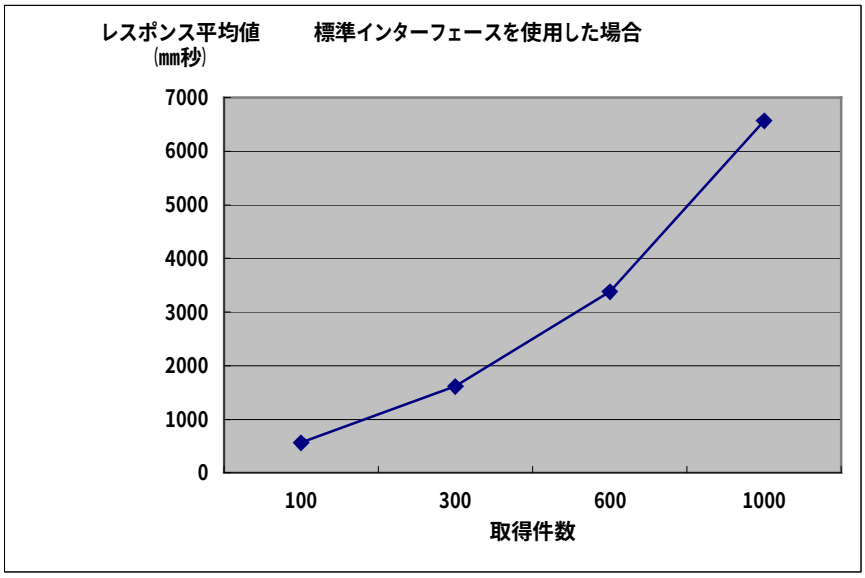
③データ取得間隔は1秒とする（値の選択は適当）

④100,300,600,1000を１セットとし、フルセット／サブセットで検索する。

⑤計測時のゆらぎを緩和させるため、フルセット、サブセットの検索を２順させる。

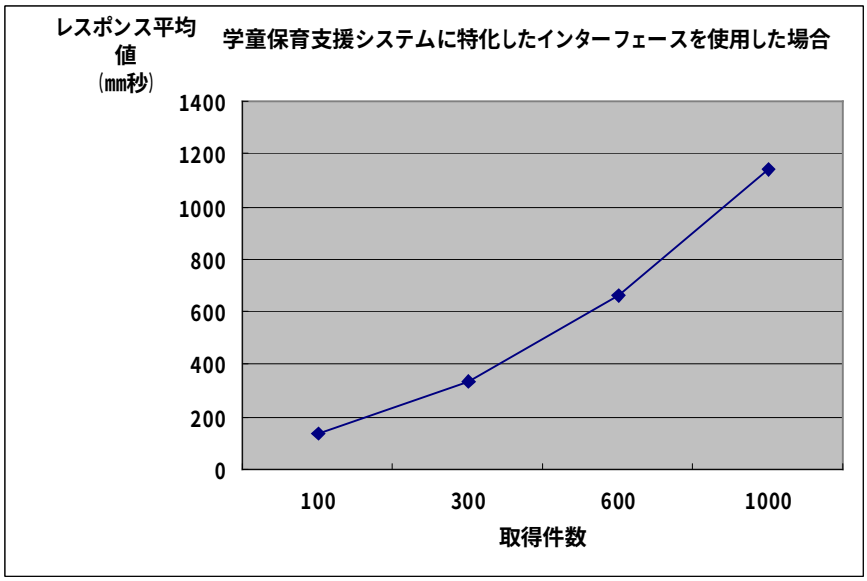
（１） 標準インターフェースを使用した場合

■図表4 4　標準インターフェースを使用した場合のレスポンス



（2） 学童保育支援システムに特化したインターフェースを使用した場合

■図表4-5 学童保育支援システムに特化したインターフェースを使用した場合のレスポンス



統合DBに構築されたテーブル内の全項目を取得したレスポンスと、学童保育支援システムが必要とする情報のみ取得したレスポンスを比較したところ、全項目を取得したレスポンスの場合では4倍から6倍程のレスポンスの低下が確認できたため、本実証では実運用での学童保育支援システムの処理レスポンスを考慮して、標準インターフェースを利用せず、学童保育支援システムに特化したインターフェースを使用して実証を行うこととした。

標準インターフェースで構築する場合と学童保育支援システムに特化したインターフェースで構築する場合のメリット・デメリットについて、下記の表で示す。

■図表4-6 インターフェースの相違によるメリット・デメリット

	標準インターフェースのみで構築する場合	学童保育支援システムに特化したインターフェースを使用する場合
メリット	・データ提供側インターフェースの汎用性が増す ・同一機能を提供する業務ユニット間の互換性が確保できる ・異なる自治体に業務ユニットを提供する場合のカスタマイズが減少する	・業務ユニットが必要としている情報のみ取得でき、パフォーマンスが向上する ・新しい業務や業務の多様化に柔軟に対応できる
デメリット	・利用しないデータも送信するため、パフォーマンスが低下する ・各種業務ユニットごとに標準インターフェースを策定する必要があり、新規業務の追加、業務の多様化についていけない	・業務ユニットの導入ごとに共通基盤側の機能追加が必要となる ・類似したインターフェースがいくつもできる。

また、業務ユニット側の設計として、統合データベースに存在するデータの複写を独自データベースに保持する場合と、必要に応じて共通基盤側に問い合わせる場合がある。今回の実証では、学童保育支

援システムは必要になる都度、共通基盤側に問い合わせる設計にした。

双方のメリット、デメリットについて下記の表で示す。

■図表4-7　データ保持の相違によるメリット・デメリット

	独自データベースに保持	必要の都度問い合わせ
メリット	・パフォーマンスが確保できる ・共通基盤の障害時にも稼働可能	・最新の情報が取得できる ・業務ユニット側でデータを管理する必要がない
デメリット	・最新のデータが取得しにくい	・パフォーマンスが低下 ・共通基盤の障害時に動作不可

業務ユニットが独自にデータベースを持ち、統合データベースの情報の複写を保持することは、パフォーマンス等の面でメリットもあるが、最新の情報を取得しにくいというデメリットがある。

特に、APPLIC標準仕様V1.0では、情報の更新について通知は行わないという規定になっているため、最新の情報を取得するのは業務ユニット側の責任となる。しかしながら、標準インターフェースにおいては、最終更新日時等の情報により、更新分だけを取得するような機能を定義しておらず、部分的なデータの同期がとりにくいという問題が存在する。

４．３　Webサービス連携の性能評価の検証

運用に向けてのWebサービス連携の評価を以下の観点により実施した。

- データをSELECTする場合の処理レスポンスの評価
- データを更新する場合の処理レスポンスの評価
- 処理を行っている時の各サーバのCPU使用率

評価内容については以下のとおりである。

（1）　Webサービス連携とSQL連携による処理性能の比較評価

【検証①】Webサービス連携とSQL連携におけるデータをSELECTする場合の処理レスポンスの評価

【検証②】Webサービス連携とSQL連携におけるデータを更新する場合の処理レスポンスの評価

【検証③】接続端末数を増加させた場合のWebサービス連携によるデータをSELECTする場合の処理レスポンスの評価

（2）　異言語間におけるWebサービス連携のデータをSELECTする場合の処理レスポンスの評価

【検証④】同一言語間、異言語間によるWebサービス連携について、データをSELECTする場合の処理レスポンスの評価

４．３．１　実証概要

Webサービス連携ではSOAPを利用して通信を行う際に、送受信するデータをサーバ側でインスタンス化し、その後、XML形式への変換を行い、さらにクライアント側で再度データのインスタンス化が行われる。このため、一度に送受信するデータ件数が多くなるほど、この処理に該当するデータ量も増大し、結果

として、処理を実行しているアプリケーションサーバでは、CPUやメモリといったリソースが多く消費され、Webサービス連携による処理レスポンスにも大きな影響を与えている。

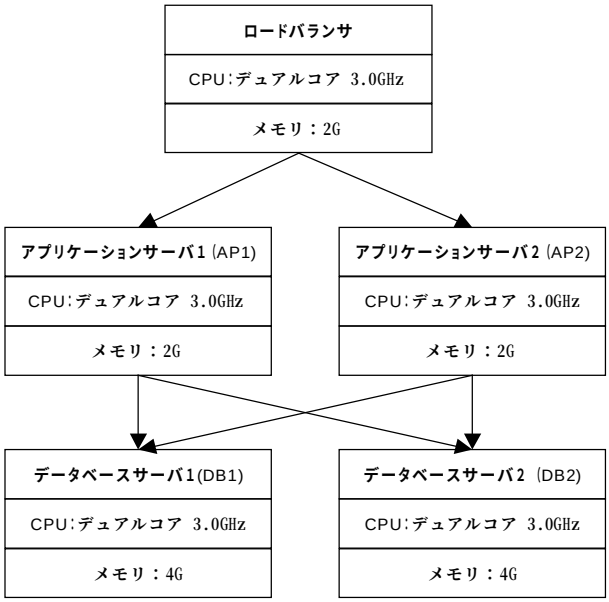
そこで本実証事業における性能評価では、Webサービス連携を利用した場合であっても業務ユニットを運用することが可能であるかを検証すると同時に、本実証事業の機器構成によるアプリケーションサーバについて実運用可能な接続端末台数についても調査した。

（1）評価環境

以下に、性能評価を実施した環境について記載する。

◇性能評価を実施したサーバ環境（図表4-8）

■図表4-8　性能評価環境の各サーバスペック



※括弧内のAP1,AP2,DB1,DB2の表記は性能評価結果の表中における各サーバ名称に対応する

◇各サーバーのコネクション数の設定値

- PostgreSQLの最大同時接続数：100
- Pgpool の最大同時接続数：70
- JBoss の同時接続数：64（アプリケーションサーバ1台あたり32）
  （JBoss が利用できるアプリケーションサーバ上のメモリの設定を1Gとする）

◇評価に利用するテストデータの件数

- 上越市様規模を想定し、テストデータの件数は50万件とした。

◇評価に利用するテストツールの仕様

- 氏名検索処理条件の配列を満たすCSVデータを作成し、データを読み込んで処理を行うメインクラスをテストツールとして作成し、評価テストを実施する。
- 氏名検索処理条件のCSVデータは5000件で、テストツールはサーバに対する処理要求の送信間隔（インターバル）を100㎜秒に設定した状態で5000件の検索処理を実行する。

・一つのテストツールが起動している状態を、稼働端末一台として評価する。

◇端末台数の増加方法

・当初起動するテストツールを5とし、検索処理を実行させる。
・テストツールの起動をタスク化し、１分後に起動テストツールの数を5追加する。
・以降、１分毎にタスク実行を繰返し、起動テストツールを5ずつ増加させる。

（２）Webサービス連携とSQL連携による処理性能の比較評価

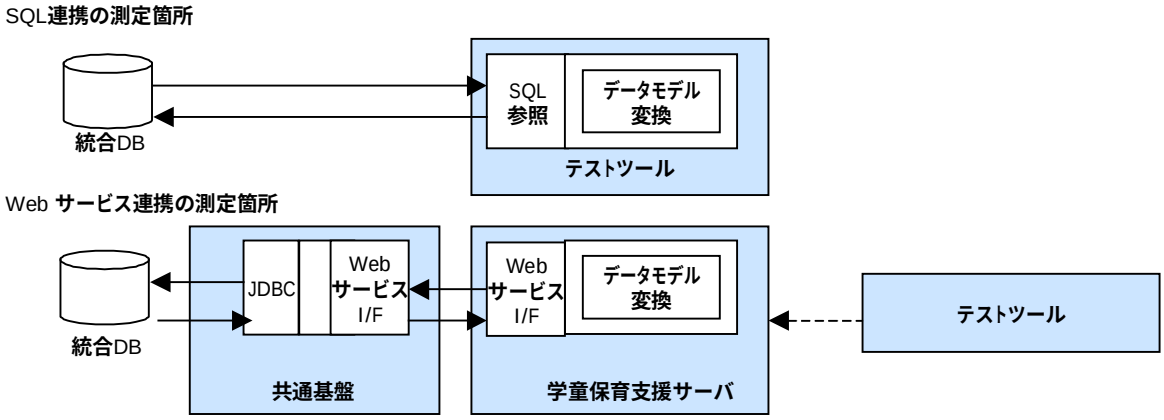
＜検証①Webサービス連携とSQL連携におけるデータをSELECTする場合の処理レスポンスの評価内容＞

Webサービス連携とSQL連携におけるデータをSELECTする際の処理レスポンスを比較し、Webサービス連携が本実証事業で実証対象としている学童保育支援システムの運用において十分な性能を維持できるかを評価すると共に、Webサービス連携による運用がどこまで可能であるかを評価する。

【検証方法】

① Webサービス連携およびSQL連携における検索処理のレスポンスについては図表4-9で示す実線区間により計測を行った。ここでは業務ユニット側がデータを取得・更新したあとの処理時間は含まれない。

■図表4-9　Webサービス連携とSQL連携の検索処理レスポンスの測定範囲



② 計測時のゆらぎを緩和させるため、検索処理を実行する端末台数を10台とし、その平均値により評価する。

③ 一度の検索処理で統合DBよりSELECTしてくるデータ件数を100,300,600,1000と増加させた場合のWebサービス連携およびSQL連携の処理レスポンスを比較調査する。

④ 評価を行っている間の端末稼働率は100%とする。（5台稼働していれば同時実行数5台となる）

＜検証①Webサービス連携とSQL連携におけるデータをSELECTする場合の処理レスポンスの評価結果＞

【検証①】の評価内容について、上記検証方法によりWebサービス連携およびSQL連携におけるデータSELECT時の処理レスポンスを計測したところ以下の結果が得られた。

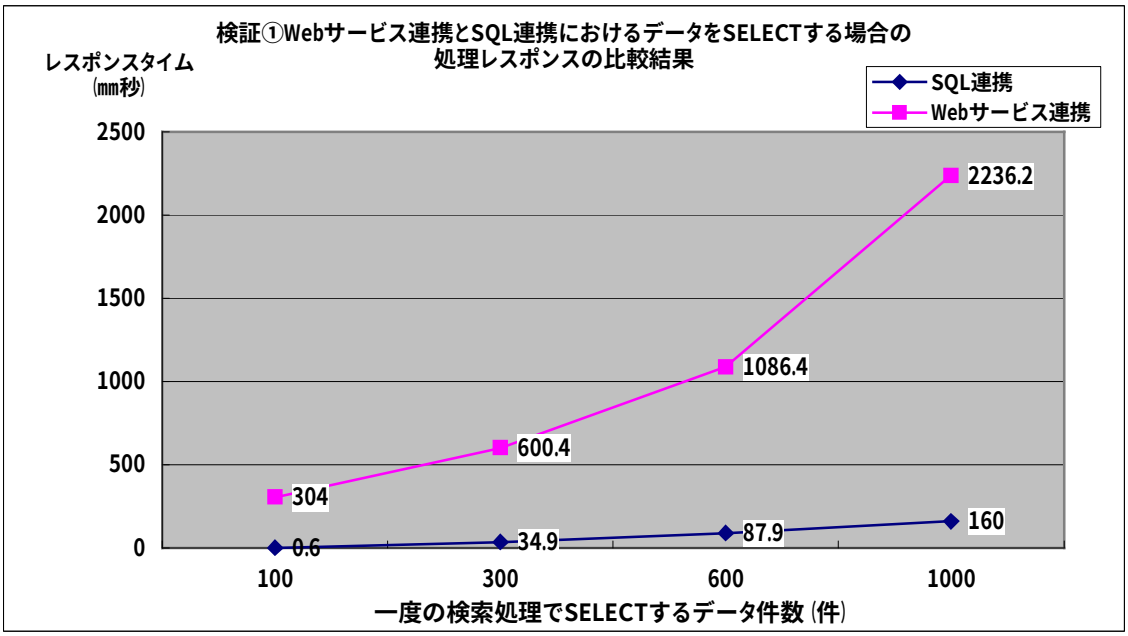
■図表4 -10　Webサービス連携におけるデータSELECT時の処理レスポンス結果

データ取得件数	データSELECT時の処理レスポンス平均値（mm秒）	CPU使用率平均値（%）				
		ロードバランサ	AP1	AP2	DB1	DB2
100	304.0	6.8	21.9	21.9	5.8	3.9
300	600.4	7.0	33.1	32.7	7.5	5.5
600	1086.4	6.9	38.2	38.1	9.9	7.5
1000	2236.2	6.9	43.8	43.8	7.5	5.3

■図表4 -11　SQL連携におけるデータSELECT時の処理レスポンス結果

データ取得件数	データSELECT時の処理レスポンス平均値（mm秒）	CPU使用率平均値（%）				
		ロードバランサ	AP1	AP2	DB1	DB2
100	0.6	—	—	—	6.2	4.2
300	34.9	—	—	—	10.2	7.7
600	87.9	—	—	—	13.8	10.7
1000	160	—	—	—	16.2	12.5

■図表4 -12　Webサービス連携とSQL連携におけるデータをSELECTする場合の処理レスポンスの比較結果



【検証①】の検証結果から、Webサービス連携では、処理経路が多岐に渡ってからデータの取得が行われるため、直接データベースを参照するSQL連携との処理レスポンスの比較では大きな差があることが確認できた。特に、一度の処理でSELECTしてくるデータ件数が600件以上となってからは両連携のレスポンス差が1秒程となったため、レスポンス差を体感するようになった。この検証結果と本実証事業で統合DBを参照する業務ユニットである学童保育支援システムの運用形態を照らし合わせて、Webサービス連携によるデータ連携が実運用に耐えられるかを評価した。

学童保育支援システムは、住民からの申請により該当児童に対してシステムへの受け付け入力を行う運用となっている。このとき、該当者の検索には児童の氏名・生年月日を入力して検索処理を行うことが中心であり、その処理により該当する児童数が600件を超えることはまず起こり得ない。Webサービス連携は処理経路が多岐に渡るため、データをSELECTする場合の処理レスポンスが懸念されたが検証結果より、一度の処理でSELECTしてくるデータ件数が300～600件であれば0.6秒～1秒程度の処理性能を発揮しており、この処理時間に学童保育支援システムが取得したデータを加工して画面表示を行う処理をする時間を考慮しても、十分なレスポンスが期待でき、実運用に十分耐えうることが可能であると言える。

ただし、上越市様の実運用では、より良いレスポンスを提供することも考慮して、学童保育支援システムの一度の処理でSELECTしてくるデータの最大件数については300件を推奨値として設定した。

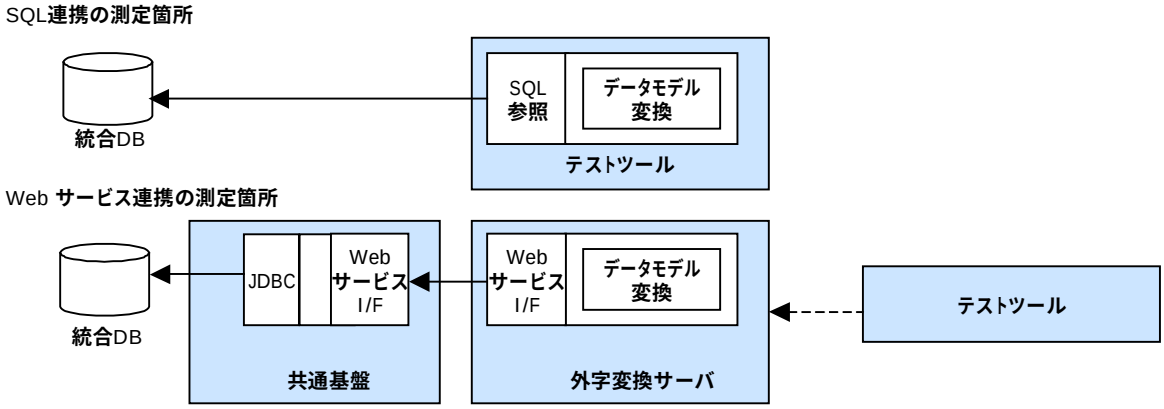
＜検証②Webサービス連携とSQL連携におけるデータを更新する場合の処理レスポンスの評価内容＞

データをSELECTする処理と同様に、データを更新する処理におけるWebサービス連携とSQL連携について処理レスポンスを比較し、Webサービス連携が本実証事業で実証対象としている基幹系システムからのデータ連携の運用において十分な性能を維持できるかを評価する。

【検証方法】

① Webサービス連携およびSQL連携におけるデータ更新処理のレスポンスについては図表4-13で示す実線区間により計測を行った。

■図表4-13　Webサービス連携とSQL連携のデータ更新処理におけるレスポンスの測定範囲



② 一度の更新処理により統合DBへ更新するデータ件数を100,300,600,1000と増加させた場合のWebサービス連携およびSQL連携の処理レスポンスを比較調査する。

③ 計測時のゆらぎを緩和させるため、測定はそれぞれ3回実施し、その平均値により評価する。

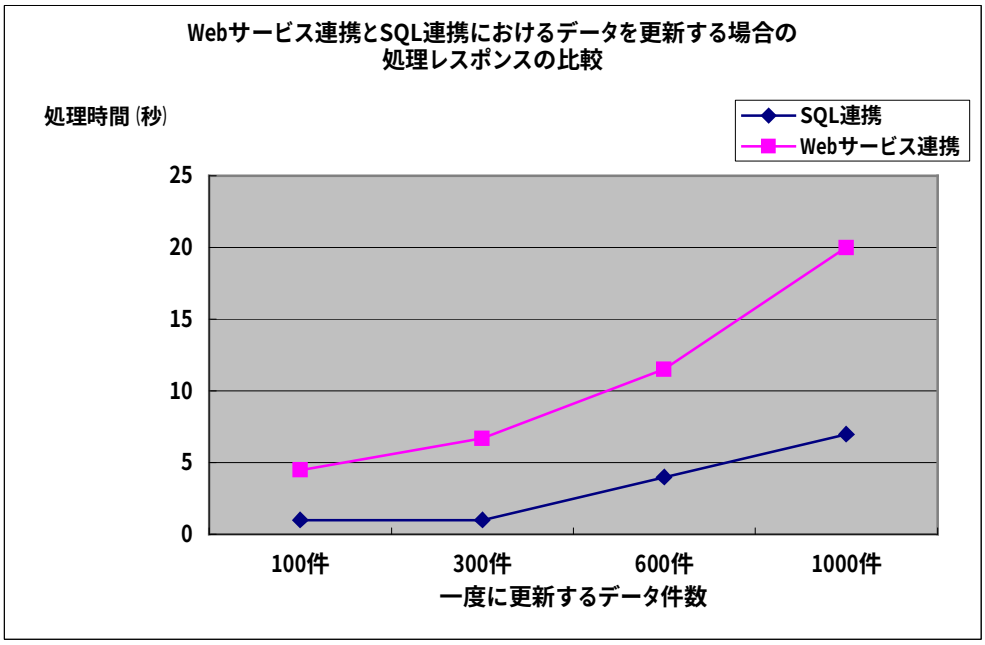
＜検証②Webサービス連携とSQL連携におけるデータを更新する場合の処理レスポンスの評価結果＞

【検証②】の評価内容について、上記検証方法によりWebサービス連携およびSQL連携におけるデータ更新時の処理レスポンスを計測したところ以下の結果が得られた。

■図表4 -14　Webサービス連携とSQL連携におけるデータを更新する場合の処理レスポンスの計測結果

一度の処理で更新するデータ件数	Webサービス連携におけるデータ更新処理平均時間（秒）	SQL連携におけるデータ更新処理平均時間（秒）
100件	4.5	1
300件	6.7	1
600件	11.5	4
1000件	20.3	7

■図表4 -15　Webサービス連携とSQL連携におけるデータを更新する場合の処理レスポンスの比較結果



本実証事業における基幹系システムからのデータ連携は、基幹系システムより発生した更新データを5分間隔で取得し、統合DBへのデータ更新を行うというデータベース間連携を行っている。本実証事業にご協力いただいた上越市様の一日の住民異動による更新データ件数は300件～500件であり、そのことも踏まえて評価については100件、300件、600件、1000件のデータ更新による評価を行った。検証②の結果として、データを取得してくる場合と同様、データベースへ直接処理を行うSQL連携がWebサービス連携でデータ更新を行う処理より短時間で処理が可能であった。

また、一度に更新するデータ量が多くなる程、両連携によるデータ更新時間の差が顕著になっていくことが確認できた。これは、これまでの検証同様、更新するデータをインスタンス化することやSOAP通信を行うためにXMLに書込む処理に時間を費やすからと言える。

しかしながら、上越市様の日常業務において5分のうちに100件以上の住民異動による更新データが発生しうる可能性はまず無いことや学童保育支援システムの運用が更新データの即時反映までを必須としないこと、今回の検証結果から一度に更新する異動データが300件程度であればWebサービス連携による更新時間も7秒程度で可能であることから、Webサービス連携によるデータ更新処理を行った場合でも十分運用が可能であると言える。

ただし、自治体内の業務処理では税業務のように、一度に数万件の更新データが発生する場合もある。

このような大量のデータ更新を扱うケースでは、Webサービス連携では処理に時間がかかる上、更新処理を実行しているアプリケーションサーバ上のメモリやCPUを非常に多く消費することが想定されるため、SQL連携により処理を行うことも選択肢として考える必要がある。

また、本実証事業で実践したデータベース連携ではなく、更新データの統合DBへの反映に即時性が必要とされる場合には、統合DBへデータを更新する業務ユニットが、自業務内のデータベースを更新すると同時にWebサービス連携により統合DBへの更新が必要となる。このような場合では統合DBへ連携する業務ユニット側の更新処理毎にシステム改修が必要となるため、データ更新の処理が汎用性を持たず、開発工数が増大することが懸念された為、本実証事業においては採用していない。

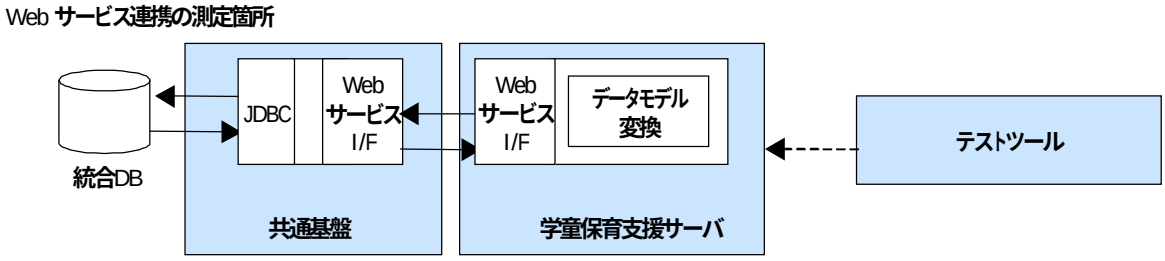
＜検証③稼働端末数を増加させた場合のWebサービス連携によるデータをSELECTする場合の処理レスポンスの評価内容＞

Webサービス連携では、データをSELECTする場合や更新する場合の処理経路が多岐に渡ることから、アプリケーションサーバへの負荷が高くなることが想定されるため、本実証事業で導入する機器構成における接続端末台数によるデータをSELECTする場合の処理レスポンスを検証し、接続可能な端末台数の最大値を評価する。また、接続端末台数の制限が各サーバのどのようなリソースに影響するかを特定することで、接続端末数を増加させる場合の機器構成について評価する。

【検証方法】

① Webサービス連携における検索処理のレスポンスについては図表4-16で示す実線区間により計測を行った。ここでは業務ユニット側がデータを取得・更新したあとの処理時間は含まれない。

■図表4-16　稼働端末数を増加させた場合のWebサービス連携によるデータをSELECTする場合の処理レスポンスの測定範囲



② 一度の検索処理で統合DBよりSELECTしてくるデータ件数を300件固定とする。

③ 当初稼働端末台数を5台とし、その後、1分ごとに5台ずつ稼働端末を増加させた場合のWebサービス連携の処理レスポンスを調査する。

④ 評価中の端末稼働率は100%とする。（10台稼働していれば同時稼働端末台数10台となる）

＜検証③稼働端末数を増加させた場合のWebサービス連携によるデータをSELECTする場合の処理レスポンスの検証結果＞

【検証③】の評価内容について、上記検証方法によりWebサービス連携における、稼働端末数を増加させた場合のデータSELECT時の処理レスポンスを計測したところ以下の結果が得られた。

■ 図表4 -17　稼働端末数を増加させた場合のWebサービス連携によるデータをSELECTする場合の処理レスポンスの検証結果（一度の処理でSELECTするデータ件数は300件）

同時稼働端末台数	データSELECT時の処理レスポンス平均値（㎜秒）	CPU使用率平均値（%）				
		ロードバランサ	AP1	AP2	DB1	DB2
5台	441.0	2.8	18.0	14.5	8.4	4.5
10台	474.3	2.7	33.1	26.7	9.5	6.5
15台	528.9	3.3	43.8	36.4	10.5	6.8
20台	639.9	3.4	49.1	43.2	11.5	7.8
25台	683.0	3.6	53.3	46.1	11.9	9.0
30台	753.4	5.0	56.5	50.3	11.2	9.2
35台	849.7	4.3	60.8	53.2	12.0	8.4
40台	911.1	3.9	61.3	53.9	11.1	9.2
45台	1001.1	3.9	62.1	53.3	12.5	9.3
50台	1101.5	5.1	61.3	51.6	11.3	8.8
55台	1182.2	4.4	62.6	54.4	12.2	8.8
60台	1254.6	4.9	62.4	54.8	11.3	9.1
65台	1348.2	4.8	61.8	55.9	12.8	8.3
70台	1349.5	4.6	62.7	55.4	11.3	9.9
75台	1555.5	4.7	64.4	57.3	10.2	8.6
80台	1800.6	3.9	62.6	55.8	11.8	7.4

評価を始めた当初は、統合DBより取得するデータを格納するテーブルへのIndexを設定せずに調査したところ、接続台数が1台でもデータベースサーバのCPU使用率が90%を超える結果となったため、Indexを設定しての調査を行った。その結果、データベースサーバについては接続する処理台数が80台となってもCPU使用率は殆ど変化することなく、安定した状態を保っていることが確認できた。

この結果より、当然のことではあるが、取得するデータを格納しているテーブルには最適なIndexの設定を行うことが必須であると言える。

一方でアプリケーションサーバでは当初、JBossが利用可能なメモリの値を256Mで設定していたが、この段階では接続端末台数が10台を超えるとアプリケーションサーバのCPU使用率が90%以上となり、データベースサーバへの接続が行えないという現象が発生することを確認した為、JBoss の利用可能メモリを1Gに変更して評価を行った。結果として接続台数が80台を超えても同様の現象が発生することはなくなった。

前述の４．３．１　実証概要で述べたように、アプリケーションサーバ上では取得したデータをインスタンス化し、またSOAP通信を行うためにXMLに書込んだデータを保持することにJBoss が多くのメモリを消費する為、初期設定の256Mではデータ処理・保持が行えずメモリ不足となり、処理の実行が行えない状態になったと思われる。サーバ機器の導入時にはアプリケーションサーバ上のメモリについてはより多くのメモリの搭載を推奨したい。

検証③の結果、接続端末台数が30台を超えたあたりからアプリケーションサーバのCPU使用率が50%を

超え始め、処理レスポンスについては接続台数が45台を超えたあたりから1秒を超える結果となった。今回取得したレスポンスはWebサービスの呼び出しからデータ取得までの時間であり、データ取得後に業務ユニットが自システムへのデータ表示のための加工時間も発生することを考慮するとデータ取得にかかる処理レスポンスが1秒程度で収まる40台の接続が本実証機での最大値であり、この環境下においてはWebサービス連携による運用も、十分実運用に耐えることが可能であると評価した。

また、端末台数が40台の時にアプリケーションサーバのCPU使用率は60%近い数値と高い結果となったが、今回の検証では平均端末利用率は100%であり、常に追加された端末が同時実行している環境で評価を行っている。実運用では40台の端末が常にフル稼働していることは考えられず、端末の平均利用率は30%～40%程度であることが想定されるので、アプリケーションサーバのCPU使用率も実運用上ではもう少し、低い数値になると考える。このようなことも含めて、接続端末台数を決める上での判断材料としていただきたい。

なお、処理レスポンスの低下はデータベースサーバではなく、アプリケーションサーバが第一に起因となることも確認できたことから、接続する端末台数を増加させる場合にはまず、アプリケーションサーバを追加し、その後データベースサーバの負荷状況を考察しながら、サーバに搭載するCPUやメモリの容量を検討していくことが必要になる。

（3）異言語間におけるWebサービス連携のデータをSELECTする場合の処理レスポンスの評価

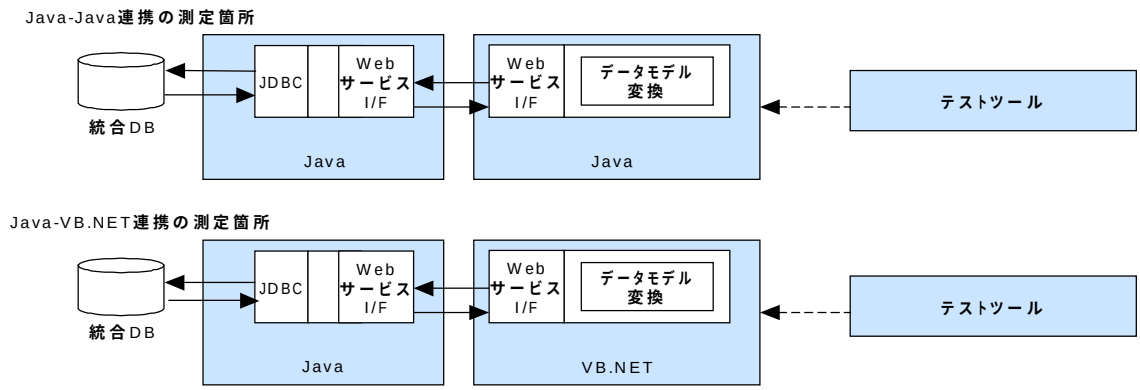
＜検証④同一言語間、異言語間によるWebサービス連携について、データをSELECTする場合の処理レスポンスの評価内容＞

本実証事業においては、共通基盤と連携する業務ユニットである学童保育支援システムがVB.NETにより構築されたシステムであることから、通常、共通基盤が提供しているJava -Java間のWebサービス連携ではなく、共通基盤側ではJava Axisを利用し、業務ユニット側はマイクロソフト社が無償提供しているライブラリを利用したWebサービス連携を開発した。今後、共通基盤と接続する業務ユニットが増加するとともに、業務ユニットを構築する言語についても多様になることが想定されるため、Webサービス連携を実現する言語の相違により、同一言語でWebサービス連携を構築した場合と比較して、顕著な性能差がないことを検証する。

【検証方法】

① Java -Java連携、Java -VB.NET連携のWebサービス連携におけるデータをSELECTしてくる場合の処理レスポンスについては図表4 -18で示す実線区間により計測を行った。

■図表4 -18　Java -Java連携とJava -VB.NET連携のデータをSELECTする場合のレスポンスの測定範囲



② 計測時のゆらぎを緩和させるため、検索処理を実行する端末台数を10台とし、その平均値により評価する。

③ 一度の検索処理で統合DBよりSELECTしてくるデータ件数を100,300,600,1000と増加させた場合のJava -Java連携とJava -VB.NET連携の処理レスポンスを比較調査する。

＜検証④同一言語間、異言語間によるWebサービス連携について、データをSELECTする場合の処理レスポンスの評価結果＞

【検証④】の評価内容について、上記検証方法によりJava -Java連携とJava -VB.NET連携における、データをSELECTする場合の処理レスポンスを計測したところ以下の結果が得られた。

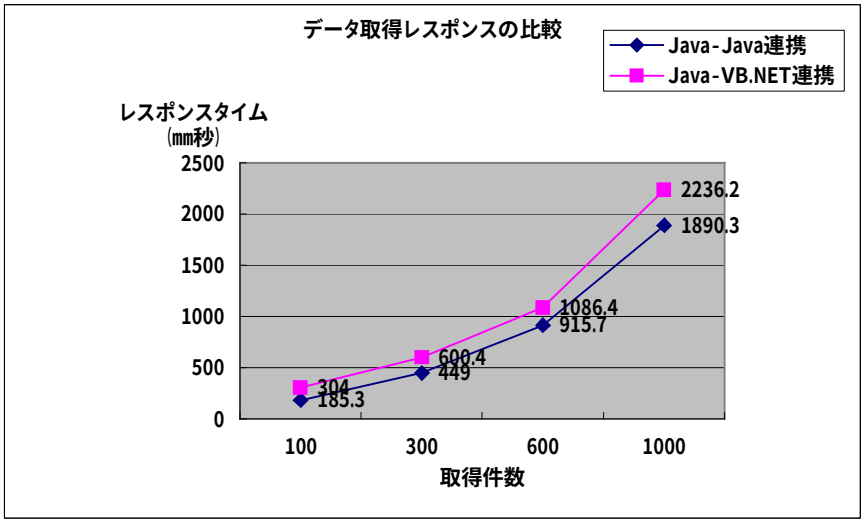
■図表4 -19　Java -Java連携における、データをSELECTする場合の処理レスポンス結果

データ取得件数	レスポンス平均値mm秒	CPU使用率平均値（%）				
		ロードバランサ	AP1	AP2	DB1	DB2
100	185.3	1.7	17.1	11.0	9.6	2.9
300	449.0	2.9	22.4	18.2	9.5	3.8
600	915.7	2.9	24.3	20.7	10.6	5.0
1000	1890.3	2.9	30.6	27.3	9.1	3.8

■図表4 -20　Java -VB.NET連携における、データをSELECTする場合の処理レスポンス結果

データ取得件数	レスポンス平均値mm秒	CPU使用率平均値（%）				
		ロードバランサ	AP1	AP2	DB1	DB2
100	304.0	6.8	21.9	21.9	5.8	3.9
300	600.4	7.0	33.1	32.7	7.5	5.5
600	1086.4	6.9	38.2	38.1	9.9	7.5
1000	2236.2	6.9	43.8	43.8	7.5	5.3

■図表4 -21　Java -Java連携とJava -VB.NET連携における、データをSELECTする場合の処理レスポンスの比較結果



検証④の結果より、Java -VB.NET連携・Java -Java連携の両処理レスポンスを比較したところ、処理レスポンスにおいては若干、Java -Java連携のほうが早いレスポンスを示す結果となったが、その差についても0.12秒～0.35秒程度の差であり、異言語間での差異は殆ど無いと言える。また、一度に取得してくるデータ件数を増加させても、その差は殆ど広がらないことも確認できた。一方で、処理レスポンス以外に着目すると、ロードバランサやアプリケーションサーバにおけるCPU使用率に相違が見られたが、この相違はレスポンス面では問題ではないが、接続する端末台数の最大値には若干の影響があるものと考えられる。

以上の結果より、異言語間で構築されたWebサービス連携であっても、同一言語で構築されたWebサービス連携と大きな性能差がなく、同等の処理が行えることが確認できたため、Java -VB.NET連携を採用した場合でも十分な性能を提供することが可能だと評価した。

４．３．２　考察

これまでの検証結果より、検証を実施する前より懸念していたWebサービス連携における処理レスポンスや、異言語間のWebサービス連携については実運用レベルでは問題がないことが確認できた。また、基幹系業務からのデータ連携など、データ提供側からのデータ連携処理で大量のデータが更新される場合にはSQL連携を利用するほうが良いという結果も得られた。レスポンス面以外では、Webサービス連携ではアプリケーションサーバのCPU負荷がネックとなり、接続する端末台数の制限について考慮する必要があることも確認できた。

このような結果より、本実証事業で評価した内容が、今後、Webサービス連携による連携手法を取り入れる際の知見とすることができたと考える。

４．４　地域情報プラットフォームに基づく統合DBの実装の検証

４．４．１　実証概要

地域情報プラットフォームに基づく統合DBの実装に関する検証を以下の内容で実施した。

（１）統合DBの設計・構築

APPLIC標準仕様で公開されている業務ユニットのテーブル定義に基づき統合DBの論理設計を行い、オープンソースデータベースであるPostgreSQLの物理設計を行った。また、PostgreSQL上に統合DB環境（データベース、ユーザー、テーブル、権限）を自動生成するスクリプトを作成した。

（２）暗号化検証

PostgreSQLではデータベースや特定の列に対する暗号化が可能かどうか、技術調査および実証を行った。これは、データの物理的な盗難・紛失への対策であり、また、統合DBの住民基本台帳の情報で扱う住基ネット番号・本籍といった重要情報に対して項目単位での暗号化を行うことによりデータを利用する側の権限を設定するためである。さらに、暗号化を列に対して行った場合に暗号・復号の処理が必要になることからも性能面での影響を評価した。この暗号化の性能評価には、PostgreSQL用のベンチマークツールであるpgbenchを使用して評価を行った。

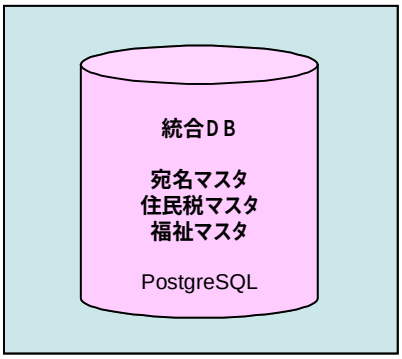
（３）クラスタ化検証

クラスタ化検証は以下の通り行った。

①本実証では統合DBをOSS製品であるPostgreSQLを選定し、統合DBの可用性を高める手段として、Pgpool（PgpoolHA）＋ PostgreSQL ＋ HeartBeatの構成でクラスタ化を実現した。クラスタ化については株式会社野村総合研究所に設計を依頼し、信頼性・可用性・処理性能の要件を踏まえて、コストパフォーマンスを最大化することを目標として設計した。また環境構築時には複雑なパラメータ設定やインストール手順が必要とされるため環境構築時に必要なパラメータの設定ファイルや環境構築手順書を準備した。

②環境構築前に統合DBの運用方法・障害対策を想定したシナリオを幾つか作成し、環境構築後に運用テストを行うことにより本構成の評価を行った。実証ではシナリオとしてネットワークの障害・プロセスの障害・ノードの障害が発生した場合をケースとして挙げ、障害を意図的に発生させることにより障害時のサーバの振る舞い及び復旧手順までを確認した。

■図表4 -22　統合DBの実装に関する検証



統合DBサーバ

４．４．２　実証結果

OracleからPostgreSQL対応については実現でき、統合DBをOSS上で実装することについては、可能であることが実証できた。

テーブル設計の項目については、APPLIC標準仕様で定義されている項目については、今回実験で使用する業務ユニット（学童保育支援システム）を対象にする限り不足している項目はないが、今後、統合DB導入の普及を考えた場合、様々な業務ユニットが共通基盤上に搭載されるが、将来的に項目が不足して再度改修しなくて済むように、明らかにカスタマイズが必要となるこが想定される項目を追加し、汎用的なテーブル設計を心がけた。なお、テーブル設計書ではAPPLIC標準仕様に対応している項目と、追加した項目が分かるように記載した。

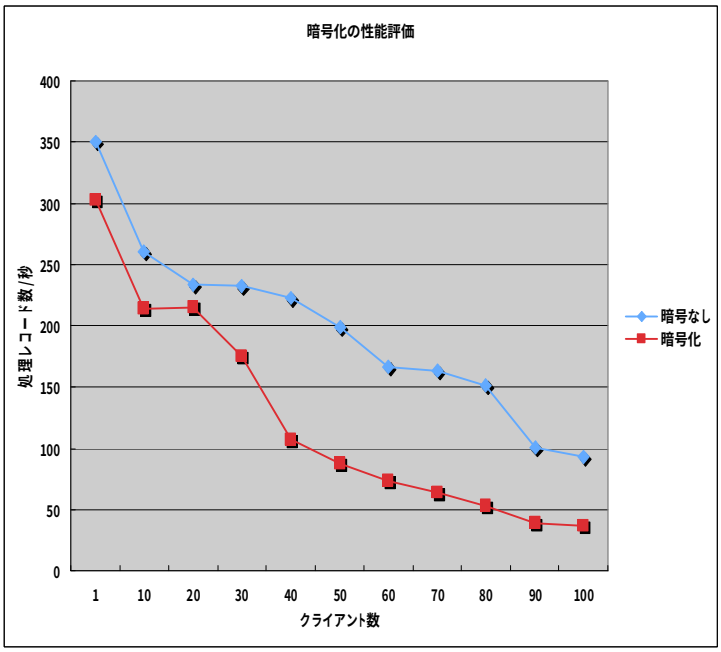
次に、テーブルのキー情報については、「住民基本台帳」の標準仕様では世帯番号と識別番号（個人番号）とで複合キーと表記されているため、それに準じて識別番号のみではなく、世帯番号と識別番号（個人番号）との複合キーとした。

暗号化検証については、PostgreSQLではデータベース自体の暗号化を行うことができ、また、列単位での暗号化についてはPostgreSQL標準の暗号化オプションを使用することにより要件を満たすことができた。暗号化の方法としては、PostgreSQLのcontrib関数ライブラリのpgcryptoによりフィールドの暗号化を行い、逆に、暗号化されたフィールドを利用する場合は、復号用のキーを使用して復号化すること

ができた。

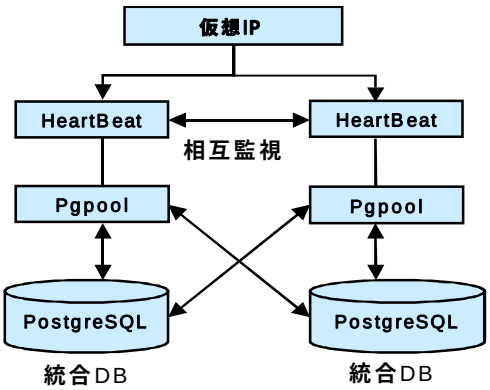
また住民基本台帳マスタの３列を暗号化（住民基本台帳マスタは全てで８２列）したことによる性能評価の結果は、図表4 -23の通りで、当初は暗号化を行わなかった場合に比べて、約85%の性能を維持していたが、クライアント数が増加するに従って性能が低下し、特にクライアント数が40台を超えると処理能力は半減することがわかった。しかし、暗号化した場合であっても、クライアント数100台で１秒あたり37件の処理能力を有しており、自治体で想定される更新件数を考えると運用上耐えられないものではないと言える。

■図表4 -23　暗号化による性能評価



クラスタ化については、本実証で要求される接続数（通常時/ピーク時の利用ユーザー数）、運用要件（サービスの稼働時間、リブート、バックアップ）を満たすことができる方式の選定を行い、設計作業から環境構築作業を実施した。環境構築作業では、Pgpool（PgpoolHA）+ PostgreSQL + HeartBeatの構成に複雑な設定が必要とされたが、環境構築の手順書を事前に作成し、Linux、PostgreSQLを初めて触れる人でも手順書に沿った環境構築を行うことができた。また、環境構築の設計段階で株式会社野村総合研究所と障害の種類やケースを想定し、設計仕様に組み込んでいたため実際の運用を想定した障害テストを行った段階でもサーバーの稼働率が確保できることが確認できた。

■図表4 -24　統合DBのクラスタ構成



４．４．３　考察

統合DBの実装に関しては、次のような知見を得ることができた。

項目の持ち方は導入自治体により可変的に対応することができることがわかった。また、前述したように、今回実証では、業務ユニットがDBを持たず、直接統合DBを参照する方式とした。なお、業務ユニットがDBを持つ場合は統合DBの差分をどのように取り込むかが課題となる。

統合DBの更新のタイミングは、基幹システムよりデータを抽出し、バッチでDB更新する方法と基幹システムの更新の都度、DB更新する方法とがあるが、今回は前者を採用することにした。

因みにデータ項目については、APPLIC標準仕様で公開されている情報のみでは不足していると思われる情報も付加し設計している。

暗号化に関してはOSSであるPostgreSQLを選定した場合でも暗号化を行うことができ、セキュリティ面を考慮した提案が可能である。但し、暗号・復号を利用するアプリケーション側で対応する必要があり、また実証結果に見られるように接続数によりレスポンスの低下につながる恐れがある為、十分な考慮が必要とされる。

クラスタ化検証に関しては、OSS製品のPgpool（PgpoolHA）+ PostgreSQL + HeartBeat　組み合わせでもクラスタ化が実現でき、障害を意図的に発生させ障害テストを行った結果も十分な稼動時間及び信頼性を確認することができ可用性が確保できたといえる。

４．５　基幹業務システムデータ連携機能・動作の検証

４．５．１　実証概要

基幹業務システムデータ連携機能・動作に関する検証を以下の内容で実施した。

（１）統合DB連携用データ出力機能

基幹業務システムから「住民基本台帳」、「個人住民税」の差分の更新データを抽出し、そのデータをFTP転送により文字コード変換システムに送信し、CSV形式で書き込まれることを確認した。

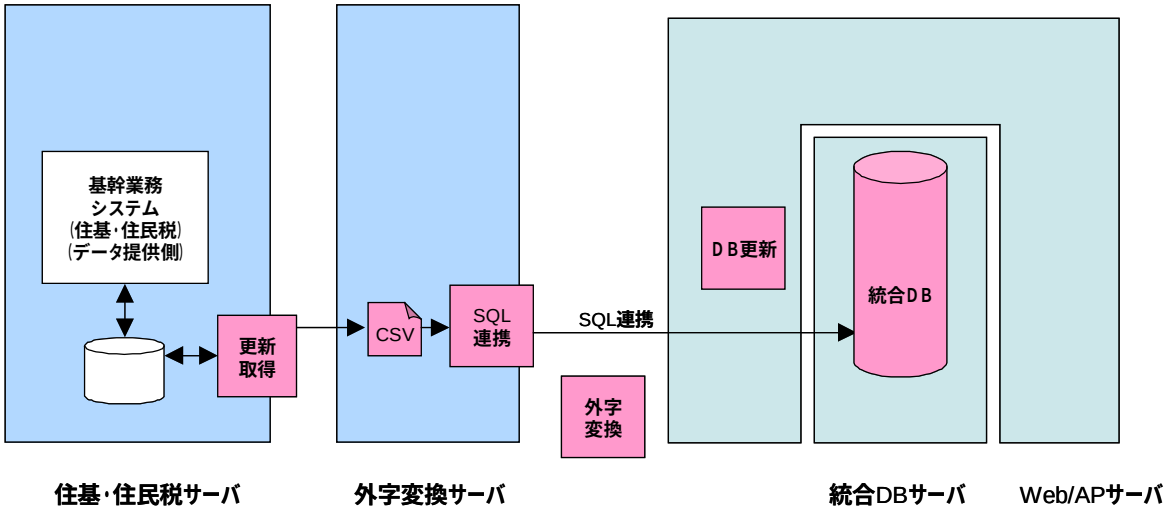
（２）統合DBへの連携取りこみ機能

上記（１）の更新情報（CSV形式）を統合DBへ取り込むプログラムを作成し、動作を検証した。

（３）文字コード変換機能

基幹業務に依存する文字コードを外字領域のマッピングを含めて地域情報プラットフォームに準拠したUNICODEのUTF-8に変換していることを確認した。

■図表4-25　基幹業務システムデータ連携機能・動作に関する検証



４．５．２　実証結果

基幹業務システムからのデータ出力機能および出力されたデータ(CSV形式)の統合DBへの取り込み機能については、動作することが確認できた。また、数分間隔・日次更新での連携も確認でき、APPLICの要件を満たすことができた。

文字コード変換はプログラムにより変換テーブルを使用し変換を行う方式とした。外字変換については、今回、上越市で利用しているShift-JISの外字格納領域をそのままUTF-8の外字格納領域として利用できたため、文字コード変換機能は動作しているが、結果は同じになった。文字コード変換は外字変換も含めて実現でき、実運用に耐えうるものとなった。

結果的に、基幹系システムから統合DBへと異なるOS間（Windows→Linux）においてもデータ連携を行うことができた。

４．５．３　考察

今回実証では、提供側の業務ユニットとしてWindows2003上で動作する基幹業務システムを対象としているが、WindowsからLinuxへと異なるプラットフォームにおいても外字変換の問題を解決することができた。

４．６　統合DBの性能評価の検証

４．６．１　実証概要

統合DBの実証実験で使用するPostgreSQLおよびPgpoolの性能評価の検証を以下の内容で実施した。性能評価は、基幹系システムより出力される更新情報のCSVデータをShift-JIS からUTF-8への文字コードの変換を行い、統合DBへ格納するまでの処理時間を統合DBへの連携プログラムを使用し以下の測定をした。

（1）本実証実験使用のクラスタ化を実現するための構成であるPgpool経由で統合DBへ取り込まれるまでの時間と、PostgreSQLに直接接続して統合DBへ取り込まれるまでの時間を比較した。これはクラスタ化を実現するためにPgpool-HAのソリューションを選定したが、Pgpool-HAは検索系の性能向上が期待できるが、更新系の性能が期待できない特徴があり、上記のテストを実測することで更新系の性能評価をし実運用に耐えうるか確認をする。
（2）基幹系システムより出力される更新情報のCSVデータを２万件、４万件、６万件、８万件、１０万件と２万件ずつ増加させた場合の統合DBへ取り込まれるまでの処理時間の変化を計測した。テストケースで更新データを最大10万件としたのは提供側の業務ユニットから出力される更新情報の件数として自治体業務を考えた上で連携件数として妥当であると考えたからである。また更新対象のマスタには住民基本台帳マスタ及び個人住民税マスタを使用して測定した。これは、住民基本台帳マスタと住民税マスタとではテーブルの列数が異なるためでテーブル設計によるレスポンスへの影響を把握するためである。
（3）PostgreSQLをデフォルトの設定値のままの状態を初期環境として処理時間を計測し、次にPostgreSQLの代表的なパラメータをチューニングした際の処理時間を測定し比較を行った。
以下は初期環境以降に変更したパラメータの例である。

◇チューニングケース１
・shared_buffers = 24MB→384MB
・temp_buffers = 64MB
・work_mem = 64MB
・log_statement = ‘all’

◇チューニングケース2
・fsync = on
・wal_sync_method = fsync
・checkpoint_segments = 32
・log_statement = ‘none’

上記の全ての性能評価では処理時間測定時にはバラツキがないことを確認するために、各テストケース毎に測定は３回行った。また、レスポンス測定時にはvacuum処理を行いデータ上の不要領域を削除し最適化した後に測定した。

４．６．２　実証結果

前述４．６．１の（1）および（2）のデータベース性能評価については、次のような結果になった。
図表4-26はpgpool経由で統合DBへの取り込みをした場合の結果で、図表4-27はPostgreSQLに対して直接取り込み処理を実行した場合の結果である。ここで「データ件数」とは、取り込み対象の更新データ件数であり、「処理時間」は更新データを統合DBへ取り込むまでの処理時間、「処理件数/秒」は１秒間に取り込めた更新情報の処理件数となる。取り込みのデータ件数を２万件ずつ増やしてレスポンス測定を行ったが、「処理件数/秒」を見ても分かるとおりデータ件数が増加しても、１秒あたりに処理できる

件数は変化していない。また、住民基本台帳マスタへの更新処理時間に対して住民税マスタの更新処理時間が多くかかっているのは、住民基本台帳マスタに比べ住民税マスタの項目数が多いこと（住民基本台帳マスタは列数82に対して住民税マスタでは列数が114）が影響していることが分かった。さらに、PostgreSQLに直接接続をし更新した場合に比べ、Pgpool経由で更新した場合には60%程度のスループットとなることが分かった。

■図表4-26　pgpool経由の場合の更新処理結果

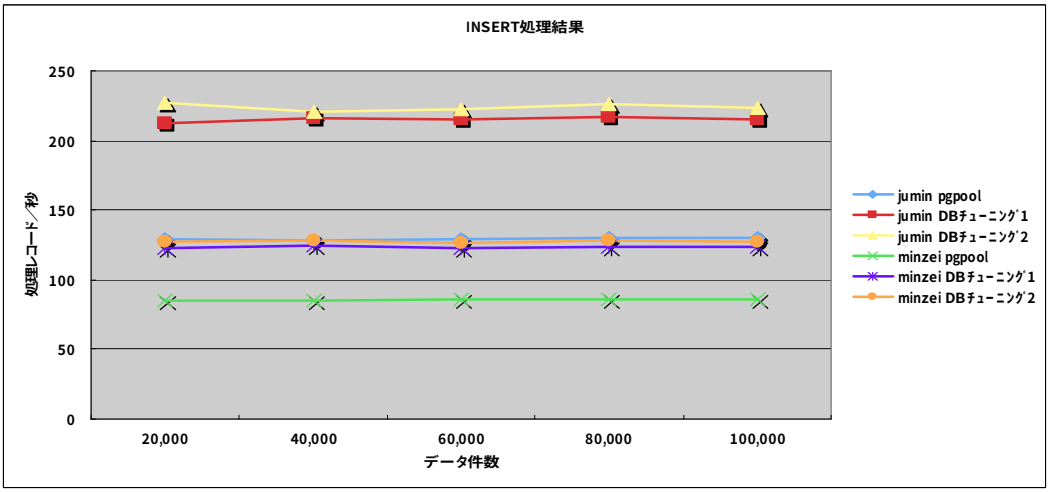
データ件数	処理時間（秒）		処理件数/秒	
	住民基本台帳マスタ	住民税マスタ	住民基本台帳マスタ	住民税マスタ
20,000	155	235	129	85
40,000	311	468	129	85
60,000	464	700	129	86
80,000	614	934	130	86
100,000	764	1,157	131	86

■図表4-27　PostgreSQL直接接続の場合の更新処理結果

データ件数	処理時間（秒）		処理件数/秒	
	住民基本台帳マスタ	住民税マスタ	住民基本台帳マスタ	住民税マスタ
20,000	94	164	213	122
40,000	185	323	216	124
60,000	279	489	215	123
80,000	369	649	217	123
100,000	464	810	216	123

前述４．６．１の（３）の検証結果については図表4-28に示す通りである。設定値が初期環境の場合とPostgreSQLの一般的な設定値をチューニングした場合の処理時間を比較すると、チューニング後は初期環境に比べ、住民基本台帳マスタでは約1.8倍、住民税マスタでは約1.5倍の処理速度の向上がみられた。

■図表4-28　Pgpool 経由のINSERT処理結果



※juminpgpool：チューニング前の住民基本台帳マスタ
※juminDBチューニング1：チューニング後の住民基本台帳マスタ
※juminDBチューニング2：チューニング後の住民基本台帳マスタ
※minzei Pgpool ：チューニング前の住民税マスタ
※minzeiDBチューニング1：チューニング後の住民税マスタ
※minzeiDBチューニング2：チューニング後の住民税マスタ

４．６．３　考察

今回の実証実験では統合DBをOSS製品であるPgpool（PgpoolHA）＋PostgreSQL＋HeartBeatの構成で可用性を高めるために構築したが、PostgreSQLに直接接続してクエリ実行した場合に比べ、Pgpool経由で接続してクエリ実行した場合では60%程度のスループットであり、期待する結果を大幅に下回るものであった。ただし、チューニングをした結果、大きな性能向上を確認でき地域情報化プラットフォームにて扱われる提供ユニット側からの更新情報や更新頻度を考えても十分な処理能力であるといえる。提供ユニットからの更新情報を統合DBへ取り込むより、統合DBへ利用ユニットが参照する頻度が多い場合には本統合DBの環境でも問題ないが、解消するためにはコストが高くなるが共有ディスク型のクラスタ構成にすることや、Pgpool以外のクラスタを実現できるソリューションを利用することも選択肢の一つである。

また、更新情報の件数を2万件から10万件と2万件ずつ増やして性能評価をした結果では、処理件数が増えたことによるレスポンスの悪化が発生せず、処理件数と処理時間は単純にリニアな関係となった。これは本実証で扱われる統合DBのサーバのメモリ容量が10万件程度の更新件数であれば十分であることの証左と言える。

４．７　開発者向けドキュメント・導入手順書等の整備の検証

４．７．１　実証概要

OSSとして公開されているソフトウェアの中には、実際の利用に必要な導入手順書や、そのソフトウェアを利用してアプリケーションを構築する場合の開発者向けドキュメントがなかったり、不充分だったりするものが存在する。

今回の実証の中心となった鳩ヶ谷基盤についても、元となった福岡基盤において、物理データベース設計書が公開されていなかったり、鳩ヶ谷基盤で提供されているモジュールについてのドキュメントが未整備であったりする状況が見られた。

今回の実証事業では、こうした状況を改善するため、OSS以外のソフトウェアに依存する部分の書き直し作業と並行して、ドキュメント類の整備を行った。

４．７．２　実証結果

今回の実証事業で整備を行った開発者向けドキュメント、導入手順書類は、「９.付属資料一覧」に示している通りである。

４．７．３　考察

開発者向けドキュメント、導入手順書等の整備を行った結果、共通基盤を導入しようとする自治体等や、これを活用した製品あるいはソリューションを提供しようとする事業者が、共通基盤の評価や採用検討をする際に、十分参考になるものと考えられる。

４．８　ライセンスの整理・調整の検証

４．８．１　実証概要

OSSとして公開されているソフトウェアの中には、明確な使用許諾契約や、著作権の行使についての表明がされていないものが存在する。

今回の実証の中心となった鳩ヶ谷基盤では、元となった福岡基盤については、福岡県による利用許諾契約が明示されていたが、鳩ヶ谷基盤で開発された部分については、利用許諾契約等が明示されていなかった。

利用許諾条件等の明示がないソフトウェアを使用することは、自治体にとっても一般企業にとっても許容できないリスクとなるため、当該ソフトウェアの採用を断念する大きな要因となりうる。

また、OSSのライセンスの中には、図表4-29のように、そのソフトウェアを組み込んだ側のソースコードの開示を条件としたり、配布に際して、同時に配布される他のソフトウェアについてもソースコードの開示を求めたりするものが存在する。

こうしたライセンス間の「互換性」の問題は、当該ソフトウェアを組み込んだソフトウェアを配布しようとする自治体や事業者にとって大きな制約となりうる。

■図表4 -29　主要OSSライセンス比較

		複製・再頒布	改変	改変部分のソース公開	組み込んだ他のコードのソース公開
オープンソースソフトウェア	GPL類型	可能	可能	必要	必要
	MPL類型	可能	可能	必要	不要
	BSD類型	可能	可能	不要	不要
フリーウェア		可能	不可		
プロプライエタリ・ソフトウェア		不可	不可		

※財団法人ソフトウェア情報センター「オープンソースソフトウェアのライセンス契約問題に関する調査」より引用

今回の実証事業の目的の一つは、福岡県および鳩ヶ谷市から共通基盤ソフトウェアの配布を委託されているOSACや、開発元と協議・協力の上、当該ソフトウェアの利用許諾条件を明確にするとともに、今回新たに開発するソフトウェアについて、これらのライセンスと矛盾しないライセンスを採用することで、当該ソフトウェアの普及を促進させようとするものである。

４．８．２　実証結果

実証事業において、ライセンス条件の確認及び明確化を行った結果、下図のとおりとなった。

■図表4 -30　ライセンス整理表

分類	ソフトウェア	機能	著作権者	配布者	提供状況・ライセンス		
					福岡県	鳩ヶ谷市	上越市
プラットフォーム通信機能	SOAP通信機能	ネットワーク上のアプリケーション間の情報を交換し合うための通信機能。この機能を表現するための手段としてXMLが利用されている。	福岡県	OSAC	福岡県ライセンス	福岡県ライセンス	福岡県ライセンス
ビジネスプロセス管理機能	BPEL対応SOA機能	複数のWebサービスを連携させることで、複雑なプロセスフローを定義することができる機能。	福岡県	OSAC	福岡県ライセンス	福岡県ライセンス	
共通機能ライブラリ	職員認証	LASDECが提供する職員認証システム。	総務省	LASDEC	非オープンソース		
	シングルサインオン	LASDECが提供するシングルサインオンの機能。	総務省	LASDEC	非オープンソース		
	自治体情報	自治体固有情報の登録機能。	RKK	OSAC		福岡県ライセンスに準拠	福岡県ライセンスに準拠
	職員情報	職員情報の登録機能。	RKK	OSAC		福岡県ライセンスに準拠	福岡県ライセンスに準拠
	スケジュール登録	職員等のスケジュールを登録する機能。	RKK	OSAC		福岡県ライセンスに準拠	福岡県ライセンスに準拠
	バッチ処理	一定量のデータを集め、まとめて一括処理を行なう処理機能。	RKK	OSAC		福岡県ライセンスに準拠	福岡県ライセンスに準拠
	ログ管理	業務システムにおける照会・更新についてのログ情報の管理機能。	RKK	OSAC		福岡県ライセンスに準拠	福岡県ライセンスに準拠
	システム連携	Web連携、DB連携、ファイル連携の機能。	RKK	OSAC		福岡県ライセンスに準拠	福岡県ライセンスに準拠
	公的個人認証	公的個人認証サービスを個人が利用するための機能。	RKK	OSAC		福岡県ライセンスに準拠	
	職員ポータル	各種業務システムの職員認証を「職員ポータル」として統合管理する機能。	RKK	OSAC		福岡県ライセンスに準拠	
	共通電子決裁	業務システムと連動して行政文書に関わるすべての決裁業務を電子的に処理するための機能。	RKK	OSAC		非オープンソース	
	ログイン機能	システムログイン時の簡易認証機能。	BSNアイネット	BSNアイネット			BSDライセンス
	マスタメンテナンス機能	システム起動時に職員情報が必要になるが、この職員情報の変更等を行う機能。	BSNアイネット	BSNアイネット			BSDライセンス
統合データベース	統合データベース	地域情報プラットフォームに基づく統合DBを構築するためのテーブル設計の機能。	BSNアイネット	BSNアイネット		非オープンソース(RKK)	BSDライセンス
	統合データベースWebサービスAPI	統合DBと共通基盤とをWebサービスにより連携させるためのインターフェース機能。	BSNアイネット	BSNアイネット			BSDライセンス
	VB.NET用Webサービスアクセスモジュール	共通基盤とVB.NETで開発されている業務ユニットとをWebサービスにより連携させるためのインターフェース機能。	BSNアイネット	BSNアイネット			BSDライセンス

上越市での実証事業で使用あるいは改修、作成した共通基盤関連ソフトウェアのライセンスは下記のとおりである。

①福岡県ライセンス

福岡県がシステムの技術基盤、構築方法を統一化・標準化した「福岡県電子自治体共通化技術標準」

の成果物配布に適用しているライセンスで、正式には「福岡県電子自治体共通化技術標準プログラム利用許諾書」である。オープンソースではあるが、どの類型に属するか不明である。基本的には複製・再頒布、改変は可能であり、MPLライセンスに近いと推察できる。なお、ドキュメントについては「福岡県電子自治体共通化技術標準（標準化文書）利用許諾書」が適用される。

②福岡県ライセンスに準拠

福岡県の著作物ではないが「福岡県電子自治体共通化技術標準プログラム利用許諾書」に準ずるライセンスで提供されるもので、上記利用許諾書の「福岡県」を原著作権者に読み替えたものである。

③BSDライセンス

今回実証事業でＢＳＮアイネットが新規に開発したソフトウェアに適用するライセンスである。OSI認定のライセンスの一つであり、複製・再頒布、改変は可能であるが、改変部分のソース公開および組み込んだ他のコードのソース公開は不要である。

４．８．３　考察

今回の実証事業において、不明となっていた鳩ヶ谷基盤の利用許諾条件を明確化することができ、自治体等が導入を行う上での大きな障害が取り除かれたと考えられる。

４．９　システム環境構築等インフラ整備の検証

４．９．１　実証概要

システム環境構築等インフラ整備の検証について以下に示す。

（１）機器調達

今回実証実験にて必要な機器の調達を以下の手順で実施した。

①実証実験を実現するためのミドルウェア選定

②ミドルウェアがどのサーバに必要か選定

③サーバのリソースを算出

④機器調達

（２）環境構築

以下の情報を基に株式会社野村総合研究所と協議したうえで、サーバ設計を行い、設定した環境パラメータ及びインストール手順書に基づき、サーバ機の環境構築を行った。

①性能、ボリューム要件

・通常時の平均利用ユーザー数

・ピーク時の最大利用ユーザー数

・単位時間当たりの処理リクエスト数

・単位ユーザー当たりのリクエスト数

②耐障害性要件

・ロードバランサ障害時の対応

・バックアップについて

③拡張性要件の確認

・将来、サーバ増設などの拡張

④外部接続要件の確認

（３）ネットワーク設計

以下の手順でネットワーク設計を行った。

①サービス（実証実験）を実現するためのミドルウェア選定

②ミドルウェアがどのサーバに必要か選定

③サーバの選定

④既存ネットワーク構成を考慮した、ネットワーク設計

４．９．２　実証結果

サーバ環境構築、ネットワーク設計等のインフラ整備については、手順通りに作業を実施することで、特に大きなトラブルもなく行うことができ、今回実証事業の環境での、共通基盤、統合DBの動作検証および性能評価に関しても実施することができた。

４．９．３　考察

サーバ環境設計書、サーバ設定記録、構築手順書により、Linuxの最低限の知識しか有していない者でも環境構築をすることができた。

また、機器調達時にサーバのリソースを見積もり、サーバ選定を行ったが、どの程度のスペックのハードウェアを必要とするのか、選定するのが非常に困難であった。共通基盤、統合DBのシステム評価や負荷試験を実施することで、本構成においても十分実運用に耐えうることが分かった。今後はスペックの異なるサーバで実証を積み重ねることにより、最適なサーバを選定することができるものと考えられる。

５．　導入実証の評価

５．１　上越市からの評価

今回の実証事業に関して、上越市の担当者から評価を行っていただいた結果を以下に示す。

（１）情報管理課の評価

①機能面について

統合データベースと業務ユニットの連携については、商用製品にひけをとらず、十分運用が可能である。

②性能面について

本実証事業で実施したWebサービス連携の性能評価については、各種パフォーマンス測定値やOSSで構築した統合DBと業務ユニットとの連携など、業務ユニットの運用を通して、連携したデータが違和感なく利用でき、十分運用に耐えるものと思われる。

③その他

本実証事業で構築した環境のコストと従来の商用製品を利用した場合の推定コストの比較評価については、システムを構築するための直接的なコスト（ハードウェア、ソフトウェア、構築費等）は安価であることがわかったが、導入以前のコンサルティング経費の比較もして欲しい。

「ライセンス整理表」については十分参考になる。

（２）子育て支援課の評価

OSSで構築した統合DBと業務ユニットである学童保育支援システムとの連携については、レスポンス面の問題もなく、データ参照などスムーズに利用できることから、十分運用に耐えられると感じられた。

以上から、上越市とは入念な事前準備を行ってきたわけであるが、基本的には今回実証事業で開発・導入したシステムの機能・性能に関しては、実運用に耐えうると評価され、また、ライセンスに関する整理表についても、今後のOSS導入について、十分に役立つものであると考えられる。

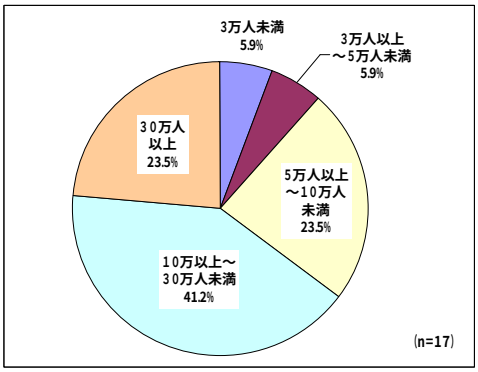
５．２　全国自治体からの評価

今回の実証事業について、OSACと関係のある全国自治体に対して、アンケート調査を実施し、得られたデータを基に集計・分析を行った。

アンケートは、全国２４市・区に配布し、有効回答は１７市・区となった。その内、直接訪問し、ヒアリングを実施したところは３市・区ある。以下に集計・分析した結果とまとめを示すこととする。

なお、対象自治体の人口規模は図表5-1の通りである。

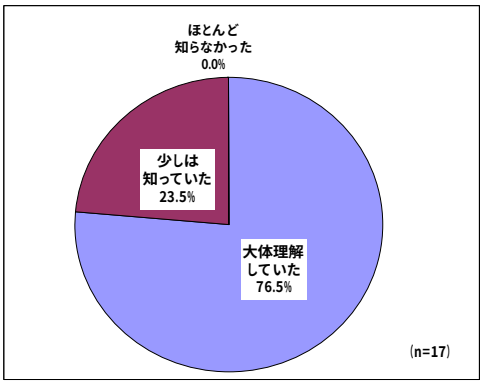
■図表5-1　回答した市・区の人口規模



（1）APPLICが提唱している「地域情報プラットフォーム」について

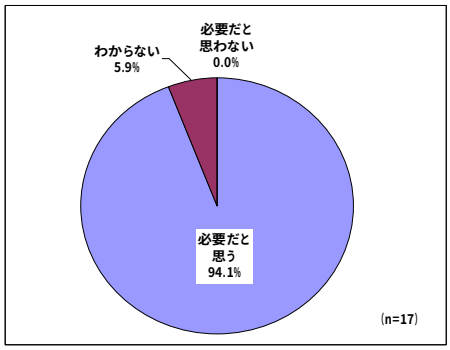
地域情報プラットフォームの認知度について聞いたところ、図表5-2のように、７割以上の市が「大体理解していた」と回答し、「少しは知っていた」を合わせると10割になり、認知度が非常に高いことが分かる。

■図表5-2　地域情報プラットフォームの認知度



次に、地域情報プラットフォームの必要性について聞いたところ、「必要だと思う」が９割以上と回答し、その必要性は十分に認識されていると言って良い。また、既に地域情報プラットフォームの標準仕様に準拠したシステムを開発中の市もあった。

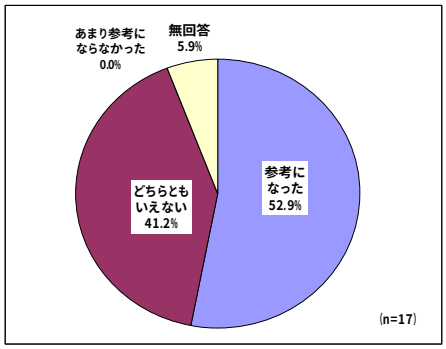
■図表5-3　地域情報プラットフォームの必要性



（2）共通基盤について

今回実証した共通基盤をOSSで実装したことに関して聞いたところ、「参考になった」と５割以上が回答している。しかし、「どちらともいえない」という回答も4割以上あり、また、単一業務だけではなく、複数業務の連携を見ないと参考にならないという意見もあり、今後の実証に期待するところである。

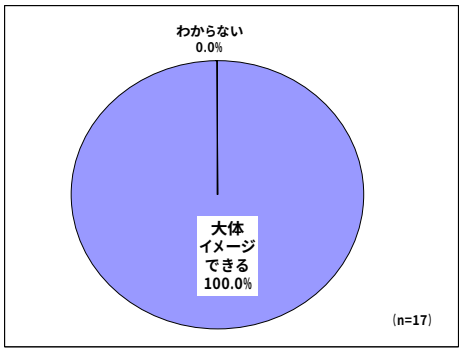
■図表5-4　共通基盤をOSSで実装したことの参考度



（3）統合DBについて

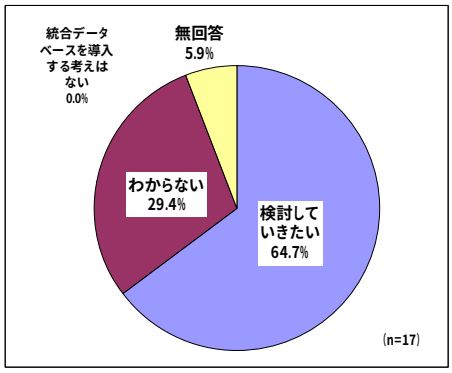
統合DBの活用方法について聞いたところ、「大体イメージできる」と10割が回答しており、その重要性は理解されていると言って良い。特に住民サービスの視点から、統合DBは全庁のワンストップ・サービスを実現する上で必要不可欠なものであるという意見があった。

■図表5-5　統合DBの活用方法の理解



また、統合DBの導入について聞いたところ、「検討していきたい」と6割以上が回答しており、前向きであることが分かる。

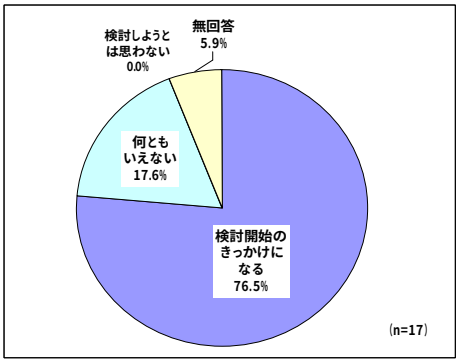
■図表5 -6　統合DBの導入の意向



（4）コスト比較について

商用版とOSS版の推定コスト比較の内容について聞いたところ、「検討開始のきっかけになる」と８割が回答しており、参考になったものと思われる。

■図表5 -7　推定コスト比較の参考度



（5）ライセンスについて

ライセンスについては内容の説明が必要なことから、直接訪問した3市にのみ聞いている。ライセンス整理表について聞いたところ、「大体理解できる」と2市が回答している。

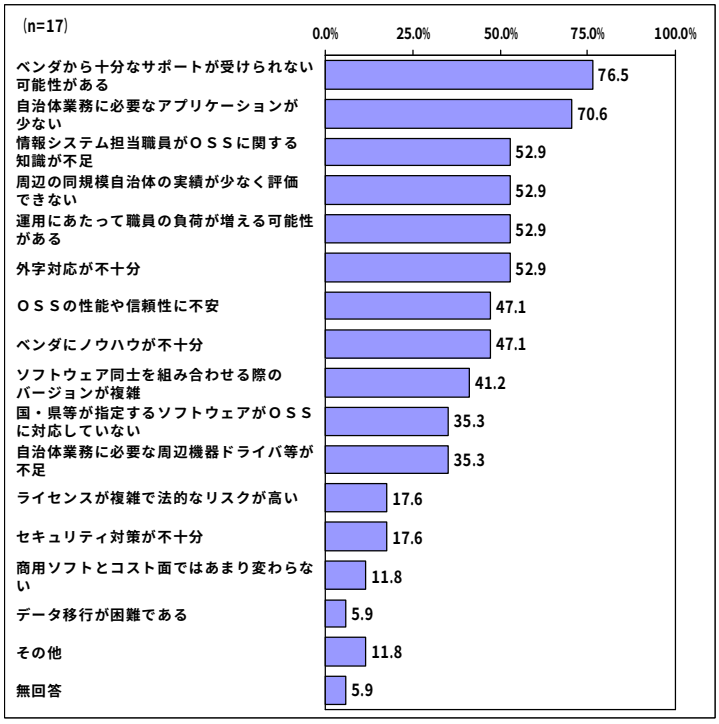
また、記述内容について聞いたところ、「何ともいえない」、「整理がたりない」と回答しており、今後、内容の充実を図っていく必要がる。

さらに、今後導入の判断材料になるかと聞いたところ、「十分なると思う」との回答があり、参考になったものと思われる。

（6）OSS導入全般について

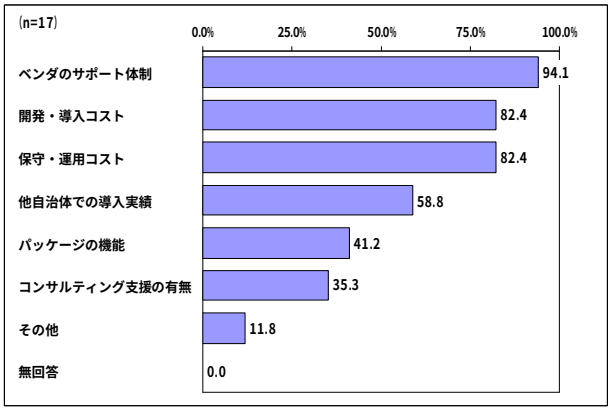
OSS導入上の課題等に聞いたところ、図表5 -11のような回答になり、「ベンダから十分なサポートが受けられない可能性がある」、「自治体業務に必要なアプリケーションが少ない」、「周辺の同規模自治体の実績が少なく評価できない」、「情報システム担当職員がOSSに関する知識が不足」という回答が多かった。

■図表5 -11　0SS導入の課題



また、今後0SSを導入するとしたら、どのような点を重視するか聞いたところ、「開発・導入コスト」、「保守・運用コスト」、「ベンダーのサポート体制」を重視している回答が多かった。

■図表5 -12　0SS導入で重視する点



以上から、今回アンケート調査した市・区は、0SSに関して関心が高く、導入検討に積極的な意見が多くあり、本実証事業の内容が非常に興味を持たれていることが感じられた、自治体における0SS導入の検討のきっかけになったものと思われる。

なお、今後の課題として次のような意見が寄せられていたので以下に記載しておく。

・共通基盤に関しては、住民記録情報を中心とした複数の周辺業務連携の実証が必要である。
・基幹業務での0SS利用に当たっては、長期間の稼動保証がなければならない。
・自治体業務に対応した0SSパッケージの拡充が望まれる。

５．３　自社他部門からの評価

今回の実証事業に関して、株式会社ＢＳＮアイネットのパッケージ開発部門により評価を実施した。その結果を以下に示す。

（１）機能面について

システム監視、データベースの管理、バックアップ管理、システム管理者権限の設定、パフォーマンスデータの収集等の機能面について、今回実証事業で開発・導入したOSS版と商用版との比較結果は、ほとんど同じであり、運用に耐えうると思われる。

（２）性能面について

システム性能試験の検証に立ち会った結果、パフォーマンスについて十分な性能が得られている。

（３）ドキュメントについて

開発者ドキュメント、導入手順書等については、本資料により、今回の実証実験の範囲である共通基盤、統合DBを同じように構築できると思われる。

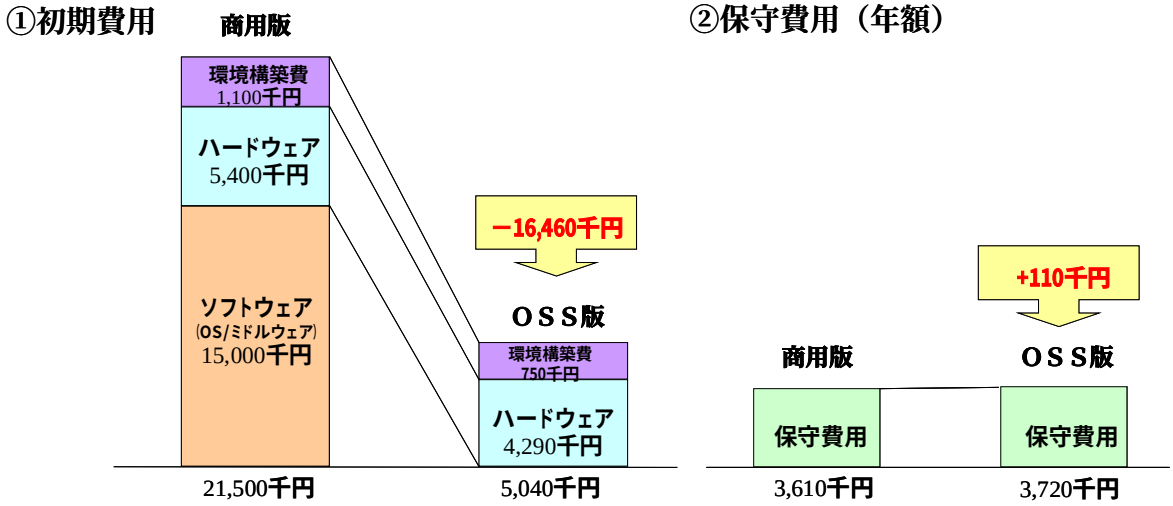
また、学童保育支援システム以外の他のパッケージを共通基盤に接続することに関しても、同様に可能である。

以上から、自社他部門からの評価ではあるが、基本的には今回実証事業で開発・導入したシステムの機能・性能に関しては、実運用に耐えうるものであり、ドキュメントについても十分に役立つという評価であった。

５．４　推定コストに関する評価

商用モデルとOSSモデルとの推定コスト比較を行った結果、下図の通りとなった。

■図表5 -13　推定コスト比較



コスト比較を行った前提条件を以下に記す。

（１）ソフトウェアの構成

項目	商用版	OSS版
ロードバランサ	負荷分散アプライアンス	Apache + mod_proxy_balancer
OS	Solaris	CentOS
アプリケーションサーバ	BEA WebLogic Server	JBoss AS
データベース	Oracle	PostgreSQL
データベース冗長化	Oracle Real Application Cluster	Heartbeat + Pgpool

（２）ハードウェアの構成

サーバ	台数
文字コード変換サーバ	1台
ロードバランサ	1台
アプリケーションサーバ	2台
データベースサーバ	2台

（３）保守費用の内訳

項目	商用版		OSS版	
	保守費用（年額）		保守費用（年額）	
ハードウェア	※1	\240,000		\170,000
ソフトウェア	※1	\3,370,000	※2	\3,550,000
合計		\3,610,000		\3,720,000

※1：価格比較を行う為のサンプル構成であり、動作保証を行うものではない。

※2：OSS版のソフトウェア保守は株式会社野村総合研究所が提供するサポートサービスを使用した場合の概算

初期費用については、OSS版が商用版に比べて約1,600万円安くなっているが、大きな要因はOSS版の方がOS/ミドルウェアの費用がかかっていないためである。ハードウェア、環境構築費については、OSS版の方が若干安価になっているがほとんど同額とみてよい。今回はサーバのみの比較になっているが、

クライアントも含めた価格で比較するとさらに、その差は大きくなると考えられる。

実際、今後どのようなアプリケーション・ソフトウェアを導入していくかにより、それを含めたトータルコストがどの程度変動するか見極めて行く必要がある。

しかし、今回の試算でわかるように、上越市の開発成果を利用する場合であれば、最小限の業務アプリケーションのカスタマイズのみで対応でき、基盤であるOS/ミドルウェアは無償で利用することができる。全国の自治体がOSS版を活用することで、システム全体の導入コストの高騰を回避することが可能となる。

また、保守費用について、今回のケースではほとんど差異は見られなかった。システム費用について、OSS版では本来ライセンス料の支払い等が不要になり、商用版に比べて低廉化できることが一般的であるが、今回はOS/ミドルウェアの組み合わせが複雑で多岐にわたっているため、信頼がおける株式会社野村総合研究所が提供するサポートを受けることにして、参考費用を記述している。

株式会社野村総合研究所が提供する主な保守サービスの内容は以下の通りである。

・定期メンテナンスサービス・・・障害の予兆を早期に発見するため、定期的な検査を実施
・問い合わせ対応・・・障害原因調査・復旧支援等
・サマリレポート・・・バグ修正、セキュリティアラートの情報提供

６．　実証に関する総括

仮説１は「共通基盤コードをOSS化し、業務ユニットとの連携が図られれば、ライセンスコストの削減が可能である」と設定したが、前述４．１～４．３にあるように、共通基盤のフルオープンソース化、共通基盤と業務ユニットとの連携については、実現でき正常に動作することが検証できた。また、ライセンスコストの削減については、５．４で述べたように、実現可能であることがわかった。以上から、仮説１は検証できたものと言える。

仮説２は「地域情報プラットフォーム標準仕様に基づく統合DBをOSSで実装し正常に動作することが検証できれば、地域情報プラットフォームの普及促進に有効である」と設定したが、統合DBをOSSで実装することについては、実現でき、実運用で使用できることが確認できた。また、これらの導入実証を全国の自治体に評価してもらったところ、「統合DBの導入を検討していきたい」という意見が多かった。以上から、仮説２は検証できたものと言える。

仮説３は「ドキュメント・ライセンスを整理し、それを公開すれば、自治体においてもOSSが選択肢となり得る」と設定したが、共通基盤の統合DBの実装を通じてドキュメント・ライセンスを整理することができた。ドキュメント整理に関しては自社他部門からよく整理されているという評価であった。また、ライセンスの整理に関しては、全国の自治体に評価してもらったところ、「大体理解できる」という意見が多かった。以上から、仮説３は検証できたものと言える。

前述の５．２で示したように、今回の実証事業の内容を全国の自治体に評価していただいたわけであるが、「検討開始のきっかけになった」という回答が多数あり、自治体におけるOSS導入の検討のきっかけになりうるものとの感触を得た。

以上から、大仮説として設定した「前述の課題解決により、自治体におけるOSSの検討促進は可能である」ということについては、基本的に検証できたものと言える。

７．　導入実証から得られた知見

７．１　共通基盤

７．１．１　Webサービスの異種言語対応

APPLIC標準仕様では、通信プロトコルにSOAPを用いたWebサービスが採用されている。共通基盤に対してWebサービス連携を行う場合、業務システム側にも相応の作りこみが必要となる。今回利用した共通基盤では、この作業を簡略化させるためのクライアントモジュールが提供されている。

ただし、Java版のみの提供であるため、本実証のように他言語（VB.NET）で連携させたい場合はやはりクライアント側を製造する必要がある。

実際、共通基盤との連携処理部には独自仕様もふくまれていたため、連携の実現にはソースコードやSOAPメッセージの解析作業が必要だった。

従って、共通基盤側で、あらゆる言語のクライアントモジュールを提供することは不可能であるが、部品クラスの詳細設計情報を開示することにより、他言語での開発が容易になるのではないかと考えられる。

なお、APPLICの定める業務ユニット側インターフェースでは、Webサービスのメッセージ名、エレメント名等に日本語を使用している。「今後の電子自治体を推進するには、業務に関する名称をすべて英語にすることは現実的ではない。」との判断によるものであり、環境等の制約により英字での実装を排除するものではないが、英字での代替標準は策定されていない。

Webサービスの標準エンコーディングとしては、妥当な判断と言えるが、Webサービスが、実装言語上のクラスやメソッドにマッピングされるのが一般的なことと、プログラミング言語によっては、日本語のクラス名やメソッド名を使用できないことを考えると、英数字のみによる代替標準の策定が望ましいのではないかと考える。

７．１．２　ユーザー認証機能

情報システムにおいてユーザーの認証や権限管理は、セキュリティ面からも重要性の高い機能であるが、人事異動等の際に変更処理の業務負荷が大きくなる原因ともなっている。

したがって、地域情報プラットフォームに接続する業務ユニットは、共通基盤で定める共通の認証機能と連動できることが望ましい。

今後の地域情報プラットフォーム標準仕様では、こうした中核となる共通仕様についても標準を策定することが望ましいと考える。

７．１．３　自治体、組織、職員等の基本情報の取り扱い

鳩ヶ谷基盤では、前述のように自治体情報、職員情報管理モジュールがOSSとして提供されているものの、データベース設計情報が提供されていなかったため、今回の実証では独自の職員情報管理機能を作成した。

こうした情報については、前述の認証機能と同様に、自治体の業務ユニットが共通で必要とする情報

であるが、APPLIC標準仕様では、人事給与システムの標準仕様に職員情報についての定義が見られるものの、共通基盤として共通の定義がなされていない。今後は、人事給与システムとの連携にも考慮した共通基盤の標準仕様の定義が望まれる。

７．１．４　共通基盤導入のポイント

以上のようなことを考慮すると、共通基盤を自治体が導入するに当たっては、少なくとも次のようなことに留意する必要がある。

・システム全体における共通基盤の果たすべき役割を検討する。
・共通基盤にある機能のうちどの機能を利用するか決定する。
・共通基盤（統合データベース）で管理する項目を決定する。
・既存ポータルを利用するのか、新規でポータルを構築するのか決定する。
・認証方式を決定する。
・共通基盤と連携する業務ユニットを選定する。
・業務ユニットと共通基盤の連携手法（標準インターフェースのみでよいのか、業務ユニットに特化したインターフェースを用意するのか。データの複製を業務ユニット側に持たせるのか否か。複製を持たせた場合の同期方法をどうするのか等）を決定する。

７．２　統合DB

宛名データは複数の業務から連携されることが想定されるので、今回の実証範囲では考慮が不要であったので実装していないが、複数経路で更新を行う場合に排他を連携業務にて意識する必要が出てくる。

同じく、今回の実証範囲では考慮が不要であったので実装していないが、重複した同一の宛名情報については同一人特定を行う機能が必要となる。

文字コード変換に関する今後の課題としては次のものがあげられる。

・外字変換は統合DB側、業務ユニット側のどちらが意識すべきかを明確にする。
・本実証のように一括でデータ連携する場合の手法にするか、オンラインで統合DBのデータを更新・参照する手法にするかを決定する。
・UTF-8に変換された統合DBのデータを業務ユニット側で利用するためには、逆の文字コード変換が必要（UTF-8→Shift-JIS）になるがその手法を決定する。

７．３　ドキュメント・ライセンス

ドキュメントの整備については、当初想定したよりも多くのドキュメント類の整備が必要となった。また、既存のドキュメントの様式や記述レベルがまちまちであったため、一貫性の確保に多くの労力を必要とした。

ライセンスの整理については、ソースコード自体に著作権の表示がないなど、ソースコード自体の由来を確認するのに多大な時間を費やした。また、著作権者と使用許諾条件の明確化の交渉をするにあたり、著作者側の利害と、活用のための制約の最小化との間の調整に、ある程度の期間が必要であった。

８． 今後の取り組み

８．１　導入実証を行ったうえでの提言

今回の実証で得られた知見を元に、自治体における情報システム整備とOSSの果たすべき役割について述べておきたい。

いわゆる「平成の大合併」により、全国の市町村では大型の合併が促進され、数多くの新しい市、新しい町が誕生した。こうした自治体が合併の効果を挙げていくためには、情報システムの整備が不可欠であるが、統合にかかるコストが大きいため、合併前の自治体の情報システムを暫定的に使い続けている自治体も少なくない。また、自治体の中には、厳しい財政事情により、情報システムの整備を進めたくとも進められないところも存在する。限られた予算の中で、いかに情報システムの整備・運用を進め、住民サービスの向上につなげていくかは、全国自治体共通の課題といえる。APPLICの提唱する「地域情報プラットフォーム」構想は、自治体システムのインターフェースや機能を標準化し、相互活用を図ることで、このような課題を解決しようとする取り組みのひとつである。

今回の実証事業に際して行った自治体アンケートでも、ほとんどの自治体がこのような共通化の必要性を感じていることが見て取れ、本事業のようなOSSによる共通基盤の構築に期待する声も多かった。

今回の実証事業で、OSSで構築した共通基盤・統合DBが、自治体の情報システムの構築基盤として、十分実用に耐えうることを実証できたと信じているが、なお自治体の担当者が採用を躊躇させる要素がいくつかあるのも事実である。それは、①サポートに対する不安、②本当にコストが削減できるのかどうか、そして、③身近な導入事例の不足である。

一つ目のサポートに対する不安は、自治体に限らず、OSSの導入を検討する情報システム担当者が共通に抱くものである。これまで、ハードウェアベンダーやソフトウェアベンダーが提供する商用製品に慣れてきたものにとって、開発者が「保証をしないこと」を前提に公開されているOSSの利用に不安を覚えるのは当然と思われる。

しかしながら、実際にソフトウェアが原因の障害が発生した場合のことを考えると、ソースコードが公開されない商用ソフトウェアにおいては、ユーザーあるいは開発元以外のシステム提供側技術者は、障害の解決をベンダーに依頼し、開発元のベンダーの技術者が解決するのを待つことしかできない。これに対して、ソースコードが公開されたOSSでは、ユーザーあるいは提供側ベンダーの技術者が、問題の所在を突き止め、修正することが可能であり、また、ユーザーや開発者のコミュニティの手助けを得て、問題を解決できることを考えれば、商用ソフトウェアのほうが必ずしも安心とは言い切れないと考えられる。ただし、こうしたOSSの利点を生かしていくためには、ユーザーはともかく、提供側となる我々サービス提供事業者が技術力向上が不可欠のものと考えられる。

二つ目のコスト削減効果に対する疑問であるが、これは、OSSの有償サポート料金が、商用ソフトウェアの有償サポート料金に比べて割高になっていることが多いことに起因していると思われる。このため、導入当初費用は、ライセンス料の発生しないOSSが安価なものの、長期に運用した場合には、サポート料の高いOSSが結果的には割高になるのではないかと考えられている節もある。しかし、今回の実証でのコスト比較でもわかるとおり、確かに有償サポートは、商用ソフトウェアを上回ることが多いが、情報システムのライフサイクルを通じたコストを考えると、OSSのほうが割高というのはあたらない。

OSSのサポートコストが割高に見えている背景には、現状では技術者の数が少ないことと、有償サポートを利用するユーザーが少ないため、「スケールメリット」が出にくいことが一因となっているとも考えられ、今後は解消に向かっていくと考えられる。

三つ目の、身近な事例が少ないことに関しては、今回の実証事業のように、地理的に自治体に近く、業務ノウハウも豊富な地方のIT企業が、OSSに豊富な事例と技術力の蓄積を持つベンダーの協力を得て、事例を積み重ねていくのが有効ではないかと考える。こうした意味から、今回協力をいただいたOSACのような活動は有益であり、また、本実証事業のように、OSS導入のリスクを軽減しながら、自治体が継続して運用をしていくことができる制度も、引き続き重要であると考える。

８．２　株式会社ＢＳＮアイネットの今後の活動方針

今回の実証事業を踏まえ、株式会社BSNアイネットとして、今後OSSの普及・推進活動を展開するに当たっての課題及び方針について下記に述べる。

今回の実証事業においては、OSACの活動や、地域のオープンソース関連コミュニティを通じて、参加・協力企業の力を結集することができたが、前述のような、ソースコード公開のメリットを生かすためには、当社自身が技術力を高め、障害が発生した場合などに適切な対応ができるようにする必要がある。これまでともすればハードウェアベンダー、ソフトウェアベンダーに依存しがちであった体制を改め、オープンソースの基礎とも言えるOSを中心に、技術者の育成に努めていきたい。

また、シングルベンダー、オールインワンでの提供が多い商用ソフトウェアと異なり、OSSでは、複数のソフトウェアを組み合わせて活用することが必要であり、当社単独では限界がある。また、OS・ミドルウェアのソースコードの解析や修正を行うことができる高度な技術者は、短期間に養成できるものではない。このため、OSSの普及に際しては、高度な技術力や事例を豊富に持つ企業との連携が重要であると考えている。この点では、OSACのメンバーを中心とした企業と今後も連携を図り、ユーザーに安心していただけるサポートを提供できるような体制を構築しなければならない。

さらに、OSSでは、ユーザー及び開発者のコミュニティが重要な役割を担っている。コミュニティへの参加は、技術者の相互研鑽にとって非常に効果的なものであり、また、OSSで発生した問題や、開発ノウハウの共有には、前記の協力企業と並んでコミュニティが不可欠な要素となる。当社も、NPO法人新潟オープンソース協会をはじめとするコミュニティ活動に協力していきたい。

一方、今回の実証の成果物の普及・推進活動については、次のように考える。

まず、上越市においては、今回の実証事業の成果を基にしたシステムが、2008年5月に本稼動を迎える。稼動に際しては、上越市の内外に広報・情報発信をしたいと考えている。

他の自治体への普及策については、営業活動として今回の実証成果の説明と、同様なシステムの構築提案を行う予定である。新潟県内はもとより、県外についてもOSACおよび電子自治体アプリケーション・シェア協議会と協力し、活動をしていきたい。また、IPAの広報活動にも積極的に協力していきたい。

さらに、地域のIT企業との連携も重要であると考えている。共通基盤・統合DBの導入により、既存の基幹システム提供ベンダーにとっては、システムの囲い込みが行いにくくなる。業務アプリケーション提供側にとっては、競争が激化し、利益を上げにくくなる側面がある一方で、自治体側にとっては、これまでコスト面で構築をあきらめていた新しいシステムの構築が行いやすくなる。こうして構築された新しい業務ユニットは、共通基盤・統合DBの導入先自治体に広く普及させること

が、本実証の趣旨をさらに発展させていくものと考える。地域IT企業との連携によって、こうしたアプリケーションの流通促進を図っていきたい。

９．付属資料一覧

本報告書の付属資料一覧を図表８-１に示す。

■図表８-１　付属資料一覧

分類	番号	付属資料	詳細内容
共通基盤	１	共通基盤設計書	・共通基盤機能概要・機能一覧 ・API仕様書 ・Javadoc ・OSS改修報告書 ・追加開発機能改修報告 ・Webサービスインターフェース開発仕様書 ・Web サービス連携インターフェース開発説明書兼手順書
	２	試験結果報告書	・単体試験報告書 ・結合試験報告書 ・総合試験報告書
	３	性能評価報告書	・性能評価報告書
	４	環境構築手順書	・共通基盤導入手順書
統合 DB	５	データベースユーザー一覧	・データベースユーザー一覧
	６	テーブル一覧	・テーブル一覧
	７	テーブル関連図（ER 図）	・テーブル関連図（ER 図）
	８	テーブル設計書	・テーブル設計書
	９	統合 DB 環境構築スクリプト	・統合 DB 環境構築スクリプト
	１０	統合 DB クラスタ設定シート	・基盤環境設計書 ・基盤方式設計書 ・基盤環境構築手順書 ・DB環境構築手順書(PostgreSQLのみ) ・設定ファイル
	１１	統合 DB 環境構築シート	
	１２	統合 DB 運用・利用者向けマニュアル	・基盤運用手順書
	１３	統合 DB 運用・管理者向けマニュアル	
	１４	性能評価報告書	・性能評価報告書
	１５	環境構築手順書	・基盤環境構築手順書 ・基盤方式設計書 ・基盤環境構築手順書 ・DB 環境構築手順書 ・設定ファイル
学童保育	１６	業務ユニット側開発設計書	・ログイン連携設計書 ・申請書入力処理画面設計書 ・保護者・児童選択画面設計書 ・住基情報選択画面設計書 ・税情報取得画面設計書
基幹連携	１７	基幹系システム連携設計書	・基幹系システム連携設計書
	１８	外字変換機能設計書	・外字変換機能設計書

おわりに

今回の導入実証の目的は、共通基盤コードのフルオープンソース化、統合DBの実装、ドキュメント・ライセンスの整備を通して、最終的に自治体における地域情報プラットフォームの普及を促進することにあった。

本編で述べているように、共通基盤コードのフルオープンソース化については、システム開発を行い、動作検証、性能評価を実施した結果、十分実運用に耐えられる成果が出たものといえる。また、OSS化によりライセンスコストの削減が可能であるということもわかった。

次に、統合DBの実装であるが、これも結果的に標準仕様を包括した汎用的なテーブルを構築したが、実運用に耐えうるパフォーマンスを得ることができた。今後統合DBの実装事例が蓄積されていく中で、地域情報プラットフォームの普及が進むものと考えている。

また、ドキュメントについては、今後、第三者でも今回実証事業と同様の作業を遂行できるように整備したと考えている。ライセンスについては、ライセンス整理表を作成したが、十分役立ったものと考えている。

以上のように今回の実証事業で掲げた目標については、十分達成され、自治体における地域情報プラットフォームの普及に向けて一歩前進したものと思っている。

今後とも利用者側および提供者側双方がしばらくは手探りの状態が続くものと思われるが、大きな流れとしては確実に自治体でのオープンソースの利用が進むものと思われる。財政健全化の流れから自治体での情報システムのコスト削減策として、オープンソースによる開発費やライセンス費用の削減に取り組むところが増えてくることは間違いないと言える。また、コスト削減だけではなく、地域経済の活性化の視点から、分割発注による地元中小ベンダーからの調達の動きも進むものと考える。

本報告書の内容が少しでも多くの自治体の方々の目に触れていただけることを願い、今後のOSS普及の一助となれば幸甚である。

最後になるが、本導入実証を共に検討していただき、共同作業を実施していただいた上越市情報管理課および子育て支援課の職員の方々に深く感謝するものである。併せて、今回実証事業に参画いただいた協力会社の皆様にも感謝したい。また、本導入実証全般にわたり全面的にご指導いただいたＩＰＡの職員の皆様にも感謝する次第である。