2010年度クラウドコンピューティング トライアルプロジェクトの総括

2011／2／25

電子情報通信学会SWIM研究会
於
機械振興会館

発表：宮西洋太郎　㈱アイエスイーエム
廣瀬修　㈱エクスウェア
坂下善彦　湘南工科大学
梶功夫　宮城大学
須栗裕樹　宮城大学
片岡信弘　東海大学

概要

- 企業や行政における情報処理経費(IT cost)節減，開発期間短縮，規模拡張性などのためクラウドコンピューティングが注目と期待を集め，普及し始めている．
- 電子情報通信学会SWIM研究専門委員会では2010年度，クラウドコンピューティングトライアルプロジェクト（CCTP）を発足させた．
- 目的
  - –エンタープライズ用途について，クラウドコンピューティングの適用可能性を見定め，
  - –適用可能ならば適用時の問題点を探り，
  - –将来性のある有望技術ならば，早急に実践的システム構築の知見を獲得することである．
- 方法
  - –ある程度の規模のアプリケーションを試作開発し，開発過程を通して，知見を獲得するという方法をとった．
  - –開発は，参画者のなかで自発的な集団を構成し，複数の開発を並行して行うこととした．
- 本稿では本プロジェクトの総括報告として，4件の試行結果を報告する．
- 結論的にはエンタープライズ用途として，トランザクション処理やクラウド提供者の特性などまだ十分に検証したわけではないが，エンタープライズ用途としても，十分な可能性をもつ技術であり，今後本プロジェクト終了後もさらに検討を深めていきたいと考えている．

内容

- 1．はじめに
- 2．トライアルプロジェクト
- 3．実績報告
  - 要件定義グループ
  - 構築･実装1
  - 構築･実装2
  - 構築･実装3
  - 構築･実装4
- 4．CEATECへの対応
- 5．全体まとめと今後

1．はじめに（1）

- クラウドコンピューティングという処理形態が大きなトレンドとなりつつある．
- 経費の節減，開発期間の大幅な短縮，部品ソフトウェアの利用による情報システム構築，規模拡張性（スケーラビリティ）の大幅な拡大，といった様々なメリットが，情報を他者に委ねることによる不安などのデメリットよりも重く評価されている結果であろう．
- SWIMでは，クラウドコンピューティングはビジネス用途に使えるものであろうかという疑問から始まった．
- この疑問を解き明かすためには，文献での調査や検討もさることながら，実際に，ある程度の規模の情報システムを開発して，開発過程を通して疑問を解決していかなければならないと考えた．

1．はじめに（2）

- クラウドコンピューティングは，
  - 自社内にサーバ機能を所有しない形態（パブリッククラウド），
  - または多数のサーバを少数に集約する形態（プライベートクラウド）
  - 情報処理に要する**経費（所有，運用維持管理のコスト）を大幅に削減できる可能性**をもっている．
- パブリッククラウドにおいては，情報処理機能の**寡占化をもたらす可能性が高く**，寡占化の側に立てば，大きなビジネス展開の可能性があるが，逆に寡占化の外側に置かれれば，**ソフトウェア開発ビジネス（各企業として，業界として）自体が衰退する恐れ**もある．
- ソフトウェア産業界は，その**存亡の危機**にあるのではないだろうか．

1．はじめに（3）

- ソフトウェアのもつ本質的な重要性
- 全製造業雇用の約10%以上にあたる約86万人の雇用をもつ業界であるソフトウェア産業界（ITソリューション産業の雇用数）[1]が衰退することは決して望ましいことではない
- Web2.0の考え方に合致するWikipediaの実例でも，**多数の叡智の集約**が，より有効であることが実感されている．
- 小規模なソフトウェア会社が多数存在していることは，わが国の強みでもある．
- ソフトウェア産業界は，クラウドコンピューティングという大波に飲み込まれることなく，むしろ**ビジネスチャンス**として，この大波を乗り切っていってほしい．

1．はじめに(4)

- そのためには，まず，クラウドコンピューティングなるものを深く理解することが肝要と考え，SWIM研究専門委員会では運営委員会の議をもって，クラウドコンピューティングトライアルプロジェクト(CCTP)を立ち上げることにした[2][3][4].
- 本稿では，その概要と，今年度の総括結果を報告する．以下，本稿の構成
  - 2.はプロジェクト全体の計画，
  - 3.は，筆者らが取り組んだプロトタイプ開発の概要を紹介する．
  - 4.ではCEATEC連携について，
  - 5.では全体をまとめる．

2．トライアルプロジェクト(1)

- 2．1プロジェクトの目的
  - ①クラウドコンピューティングのビジネス用途への適用性を見極め,
  - ②肯定的見極めがつけば問題点を探り,
  - ③システム構築のための実践的知見を早急に獲得すること,
- が大きな目的である．これをブレークダウンして以下に挙げる[4].

2．トライアルプロジェクト (2)

- プロジェクトの目的 (ブレークダウン)
  - (1) エンタープライズ用途へのクラウドコンピューティング技術あるいはサービスの適用についての実践的知見 (クラウドコンピューティング技術全般的についての知見，サービス提供者ごとについての知見，情報システム構築上の利点，留意点，制約事項，構築実践方法などポジティブ面，ネガティブ面ともに) を獲得し，広く公開する．
  - (2) サービス提供者側ではなく，サービスを利用してエンドユーザの要求に対応するであろう大多数のソフトウェア会社にとって，クラウドコンピューティングは，チャンス (opportunity，機会) なのか，スレット (threat，脅威) なのか見極める．スレットならば，チャンスに変える方策について探る．

2．トライアルプロジェクト (3)

- プロジェクトの目的 (ブレークダウン)
  - (3) 複数存在するクラウドコンピューティングサービス提供者のサービスを試験的に (トライアルとして) 使って，同一の要件が定義されたビジネスアプリケーションシステム (小規模プロトタイプシステム) を構築することにより，エンタープライズでの**各社のクラウドコンピューティングサービスの利用方法**についての実践的知見をSWIM研究専門委員会として，獲得する．
  - (4) 特に研究専門委員会としては，技術的課題を整理する．**システム構築に関する技術的課題** (分散データベース，並列処理など)，**システム運用に関する技術的課題** (セキュリティや認証など) を整理する．
-

2．トライアルプロジェクト(4)

- プロジェクトの目的(ブレークダウン)
  - (5) 本プロジェクト計画書は，最初の年度(2010年度)について，および，ある少数のアプリケーションについての計画である．
  - (6) これらの知見は，ソフトウェア業界に広く公開し，ソフトウェア会社がシステム構築する際の参考に供するものとする．クラウド時代でのチャンスとして，各社が競争力獲得の一助とする．
  - (7) 知見を積み重ね，クラウドシステム構築のための参照モデルあるいはガイドラインへ．
  - (8) また，中間段階の要所要所で，適宜，SWIM研究専門委員会が開催する研究会やCEATEC2010などの場で，プロジェクト成果を発表する．

2．トライアルプロジェクト(5)

- **2.2. プロジェクトの推進方法**
  - (1) 本プロジェクト計画書をSWIMフォーラム，Webなどで広く公開し，参画者（個人，大学･企業などの組織）を募る．
  - (2) 参画者（メンバー）が固まった後，以下のプロジェクト活動を行う．
  - (3) 要件定義の確立，ビジネス系の要求仕様書（要件定義書，要求定義書，システム要求仕様書，本稿ではシステム仕様書と呼ぶことにする）をメンバーの間で検討し，確立する．
  - (4) メンバーは，独自の判断で，クラウドコンピューティングサービス提供者を選定し，同一共通のシステム仕様書に従って，提供されているクラウドサービスを有効に利用して，情報システムを構築する．クラウドコンピューティングサービス提供者を複数選択することはメンバーの自由である．

2．トライアルプロジェクト(6)

- **2.2. プロジェクトの推進方法**
  - (5) このトライアルとして，クラウドコンピューティングサービス提供者をできるだけ網羅することが望ましいので，上記の選択結果を持ち寄り，研究専門委員会として相談し，**調整のうえ，決定する**．ただし，研究専門委員会の性格から，各参画者の自主的な開発であり，開発契約のような**強制的義務を負うものではない**し，その代わり**開発の対価もない**．参画者それぞれの自主的な負担によるものである．打ち合わせ会議場などの手配と費用負担は，通常の研究専門委員会と同様にSWIM研究専門委員会とする．
  - (6) メンバーによるシステム構築を行う．
  - (7) 途中結果での，発表，検討，相談を行い，相互の疑問点の解決や実践的知見の獲得を加速する．研究会の場を用いることや，別途検討会議を開催する．

2．トライアルプロジェクト（7）

- **2.3. ワーキンググループの構成**
  - **(1) ステアリング委員会（PMO Project Management Officeを兼ねる）（共通）**
    プロジェクトの方向性を検討し，決定する．プロジェクト計画書の維持保守を行う．プロジェクト管理を行う．ただし強制力はもたない．構成メンバーは各参画者（組織）の代表者，個人参画者は当人．委員長は，委員からの互選とする．
  - **(2) 要件ワーキンググループ（共通）**
    構築すべき情報システムの架空の要求について検討と定義を行い，決定する．システム仕様書，システムテスト(総合テスト)仕様書を作成する．構成メンバーはステアリングWGからの指名，または参加希望者．WG長は，WGメンバーからの互選とする．
  - **(3) 構築ワーキンググループ（個別）**
    参加メンバーごとに，独立に，実際にトライアルとしてシステム構築するワーキンググループである．

2．トライアルプロジェクト(8)

- **2.3. ワーキンググループの構成**
  - **(4) 評価ワーキンググループ (個別→共通)**
  - **ここでの評価は，完成した情報システムの動作をテストすることではなく (それは各参画者が行う)，各サービス提供者について，システム仕様書とおりの情報システムを開発するにあたっての実践的知見を獲得するための評価である．各参画者からの研究成果を収集，評価，とりまとめる．本プロジェクト外への発表について扱う．構成メンバーはステアリングWGからの指名，または参加希望者．WG長は，WGメンバーからの互選とする．**

2．トライアルプロジェクト（9）

- **2.3. ワーキンググループの構成**

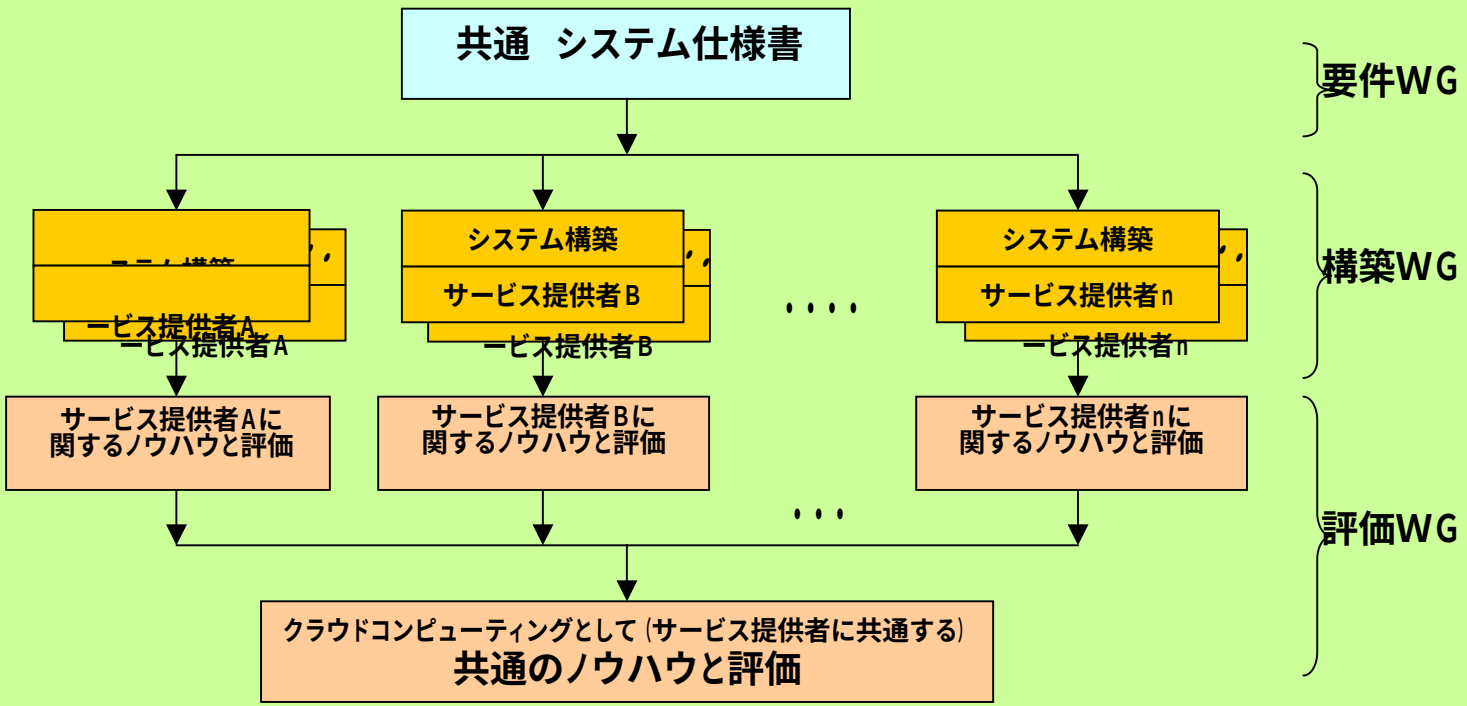


図1　ワーキンググループと分担範囲

2．トライアルプロジェクト（１０）

- 2.4. WBS(第1レイヤのみ)

  (1) クラウドコンピューティングを実現している要素技術の調査
  ・従来RDBからKeyValueデータストア（KVS）へ，
  ・弱い一貫性(Eventually Consistency)，
  ・従来ACIDからBASEトランザクションへ，
  ・並列処理(MapReduce)，
  - ・仮想化技術，
  - ・スケールアウト技術など．

  (2) 各クラウドコンピューティングサービス提供者のサービス内容の調査
  - ・Google社の技術　Google App Engine(GAE)　（PaaS），（データストアの仕組みGoogle File System(GFS), Big Table，並列処理の仕組み　MapReduce）
  - ・Amazon　EC2の技術　（IaaS），
  - ・MicroSoft社の技術　Azure　（PaaS），
  - ・salesforce.com社の技術（SaaS），など．

2．トライアルプロジェクト（11）

- 2.4. WBS(第1レイヤのみ)
  - (3) システム仕様書の作成
  - (4) 各参画者のサービス提供者の選択／選択調整
  - (5) 各参画者の情報システム開発（動作テスト含む）
  - (6) 各参画者の成果のまとめと発表
  - (7) 対外的発表の準備
  - (8) 対外的発表

2．トライアルプロジェクト（12）

- 2.5. マイルストーン
  - (1) プロジェクト計画書発行，参画者の応募，　2009/12
  - (2) 参画者の確定，参画者が事前調査を完了，　2010/2
  - (3) プロジェクトスタート，　2010/4
  - (4) システム仕様書，不明点の解決，　2010/5
  - (5) 各参画者の情報システム開発開始　2010/6
  - (6) 各参画者の情報システム開発完了，　2010/8
  - (7) CEATECでの発表　2010/10
  - (8) 成果の検討，議論，今年度のまとめ，　2011/2

2．トライアルプロジェクト（13）

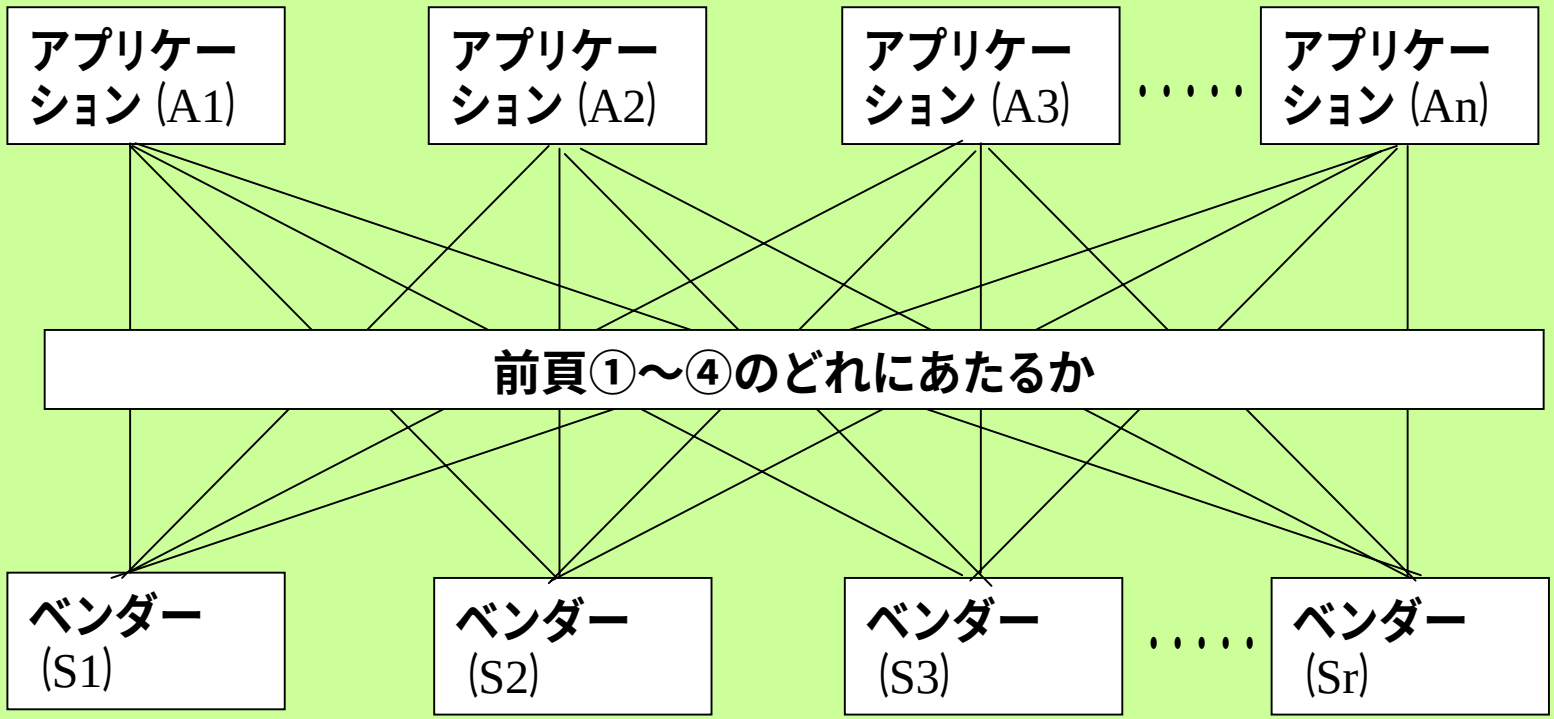
- 2.6. 予想成果
- 獲得するノウハウの例として，
  - (1) クラウドコンピューティング要素技術の利用方法や注意点の把握
  - (2) クラウドコンピューティングサービス提供者とアプリケーション種別との相性の把握
  - アプリケーションとサービス提供者との相性（得意/不得意）が考えられる．アプリケーションをA1,…,Anとし，サービス提供者をS1,…,Srとする．相性を例えば次のように分類する．

2．トライアルプロジェクト（14）

- 2.6. 予想成果
- (2) クラウドコンピューティングサービス提供者とアプリケーション種別との相性の把握，
  - ①アプリケーションとサービス提供者の組み合わせ(Ai, Sj)は，どのように工夫しても，ほとんど実用にならない．（例：リアルタイムで，厳密なトランザクション処理を行うアプリケーションをKeyValueデータストアで実装する場合）
  - ②アプリケーションとサービス提供者の組み合わせ(Ai, Sj)は，要求仕様を緩めることによって実用になる．（例：ユーザの待ち時間の許容値を大きくする）
  - ③アプリケーションとサービス提供者の組み合わせ(Ai, Sj)は，データの持ち方を工夫することによって実用になる．（例：更新データと読み取り専用データの分離）
  - ④アプリケーションとサービス提供者の組み合わせ(Ai, Sj)は，なんら構築上の工夫を要することなく実用になる．（例：大量情報のKeyValueデータストアによる検索）
  - 上記のような使用方法に関するノウハウが獲得できれば，情報システム構築者として，大いに有益であろう．またクラウドとオンプレミスとの使い分けや，ハイブリッドクラウドの検討にも有益であろう．

2．トライアルプロジェクト（15）

- **アプリケーションタイプとベンダーの相性**



3．実績報告(1)

- 3.1. 参画者の募集
- SWIM研究専門委員会委員へのメールによる案内，およびSWIM幹事からの個別の案内により，約50名の参画者があり，活動を開始した．
- 特に仙台地区については，筆者の数人に地縁があり，宮城県情報サービス産業協会(MISA)様のご協力で多くのソフトウェア企業から参加いただいた．
- 大手ソフトウェア企業は，単独でクラウドコンピューティング時代に備えていることと思われ，このような集団的取り組みの必要性は少ない．
- 一方中小規模の企業では集団的取り組みが有効と考えられるが概して，日常的に研究的業務を行う余裕がなく，参画しても積極的な活動が困難という事情がある．

3．実績報告（2）

- 3.2. プロジェクトのスタート
- 参画者が確定した段階で，東京地区では2010年3月27日に，多数の参画者に集まっていただいき，キックオフ会議を行った．ここでは，
- ステアリンググループ長（宮西），
- PMO責任者（片岡），
- 要件定義グループ長（廣瀬），
- 構築グループ長（坂下），
- 評価グループ長（須栗）
- の各（ワーキング）グループ長を決定した．仙台地区では，2010年4月20日にキックオフ会議を行い，
- 仙台地区での推進体制（リーダ梶）を決定した．

3．実績報告(3)

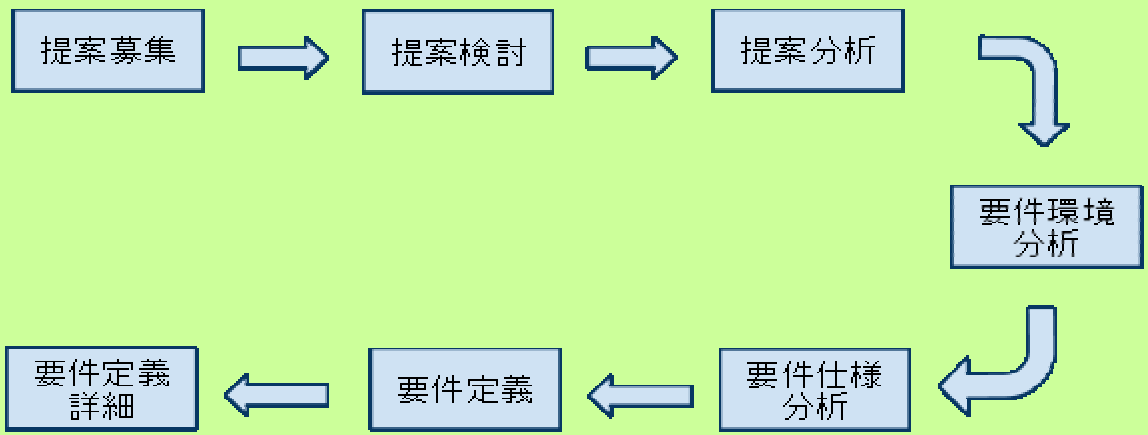
- 3.3. 要件定義グループの報告(廣瀬)
- 参画者からの提案を求め，7つのアプリケーションの提案があり，2010年5月7日，10日に要件定義グループ長主催の会議を行い，3つのアプリケーションに絞り込んだ．
- 3.3.1. 要件定義作業の前提
- 　今回のプロジェクトはクラウドシステムの実験的開発という意味を持って開始されたものであることから，要件定義グループは，一般的な要件定義作業の前々段階ともいえる，開発するシステムそのものの募集から始めることとした．また，利用するクラウド環境(PaaS)の選定も実験の一つと考え，どれという特定をせず自由に選定できるものとして始めた．

3．実績報告（4）

- 3.3.2. 募集にあたっての提案発表環境の提供
- 募集するにあたって提案の促進を図るため，必要な項目を埋めることによって提案を完成するようにスプレッドシートによるドキュメント仕様を作成した．
- またその更新にあたっては，アップロードの即時性，会員への通知の煩雑さの回避を大きなメリットとし，かつ厳格ではないが，基本的に会員のみが閲覧可能という性格を持たせられるところから，ツールとしてGoogle社のGoogleDocを用い，ブラウザからそれを開けば提案者は自由にいつでも提案をアップし，また会員はいつでも最新の情報を閲覧できるようにした．
- そのスプレッドシートは現在も下記サイトで閲覧可能である．（現在も更新は可能であるが，更新は本プロジェクトとしては既に意味をなさない．）
- https://spreadsheets.google.com/ccc?key=0AjPcRg8i-yvcdFZ0Um9NTHdqa05fLXotMmkzb2tKVmc&hl=en#gid=0

3．実績報告(5)

- **3.3.3. 要件定義グループの作業の流れ**
- **要件定義グループは冒頭に話したように本来要件定義の範囲を超えて作業を行ったわけであるが，流れとしては約2か月を掛け図2のように進める予定であった．しかし要件仕様分析のドキュメント仕様の作成(RA1-a)までで終わってしまった．(上記GoogleDocサイトを参照)**
- **途中までの作業の報告になるが，以下順次述べることとする．**



- **図2　要件定義グループの作業フロー**

3．実績報告（6）

- (1) 提案募集
- 2010年3月27日の最初の提案から5月7日の最後の提案まで7件の提案があった．（応募提案内容については，上記GoogleDocのシート提案一覧を参照．）

- 提案募集はプロジェクトの期間を考慮して締切りを設けた．しかし，多くが集まったとは言えなく，かつ締切りを厳守できなかった．

- その結果スケジュールが後ろに押される形となった．一般に上流工程ほど変動要素が大きいため時間的猶予が必要であるが，これは今後の課題の検討材料としたい．

3．実績報告（7）

- (2) 提案検討および分析
- 2010年5月10日要件定義グループとしてのミーティングを開き，提案検討に入った．
- 提案検討においては，提案内容に対して，そのメリット，デメリット，想定する利用技術，開発難易度，大体工数見積りなどについて分かる範囲において記述をお願いし，より客観的な提案となり，また相互提案の比較検討もできるようにドキュメントを工夫した．（シート提案一覧を参照）
- 分析については，利用目的ないし用途および，想定する利用対象，非クラウド環境との相違を操作性，利便性，保守性，運用性の視点から洗い出すようにし，また見積もりも人，装備，費用，開発期間などから，より具体的かつ客観的な判断できるようにし，それらの総合評価として5段階評価をつけ，開発判断を提供するようにした．（シートPAを参照）

3．実績報告(8)

- (3) 要件環境分析，要件仕様分析
- 要件環境および要件仕様の各分析については，ミーティングを開くことができず，ドキュメント仕様を纏めるだけに終わったが，以下に示す．
- ① 環境分析
- どのPaaSを利用するかをそれぞれの長所や制限を検討しながら分析を加え，提案内容をより具体的構造的にイメージすることで，開発環境としての要件定義を構築することを目途に，各PaaS毎に検討を加えられるように，想定されるPaaSエンジンGAE，SalesForce，Azure，その他(Hadoop，Romaなど)を列挙して一覧できるようにドキュメント仕様を纏めた．(シートRA1を参照)
- ② 要件仕様
- 環境分析を前提として，何をどう実現させるのかを箇条書きにして要件仕様を明確にすることを目途として，実現項目を箇条書きにし，項目毎の矛盾を防ぐためのフラグ，また実施条件の整理が客観的にできるようにドキュメント仕様を纏めた．(シートRA1-aを参照)

3．実績報告(9)

- (4) 要件定義，要件定義詳細
- 要件定義については，ドキュメント仕様も進まなかったが，上記の要件環境分析，要件仕様分析から，実現環境，実現内容，実施条件等を明記して，基本設計に資する材料を提供しようとした．実現環境としては，主として利用するPaaSとその利用環境の仕様と可用性を考え，想定される利用形態についての概要を，実現内容については，箇条書きにして，DB設計に資するような体裁をもって記述し，実施条件等についてはセキュリティその他の制限に対する対応について記述する予定であった．

3．実績報告（10）

- 3.3.4. 要件定義グループのまとめ
- 当グループは上記のとおりその作業半ばで構築グループに引き継いた形となっており，中途半端に終わってしまった．
- このような結果となったことについては，要件定義グループのミーティングが各位のスケジューリング調整，作業分担が実務上うまく進められなかったことと（実際は1回しか集合をかけられなかった．），何よりグループ長である筆者（廣瀬）自身が会社の業務に忙殺され，メンバー各位との十分なコミュニケーションがとれなかったことに大きく起因すると考える．また，これは終わった後で言えることであるが，時間的制約を考えむしろ実験としては予めするべきものを決め，その要件定義だけに絞って進めた方がよかったかもしれないと感じもした．結果論である．
- これは個人的なことながら，長としてその甘さは大いに自省すべき点であり，解決する方法としては，自身がアシストする人間を今後確保しなければ実質不可能に近いとの判断をした．

3．実績報告（11）

- 3.4. 構築グループの報告
- 上記のプロセスで選定した3つのアプリケーションのどれかを，自発的な開発チームを構成して，並行して開発し，結果を持ち寄ることになっていた．しかしこの段階から，実際に開発に取り組むチームは，なかなか多くはなかった．
- そのような中でも，仙台地区においては，アプリケーション開発のほかに独自のテーマ（「リレーショナルデータベースからKeyValueデータストアへの移植ノウハウ整理」など）を設け，積極的に取り組んだ．
- プロジェクト全体としては，多種（クラウドサービス提供者）のプラットフォーム（PaaS）について，試行する予定であったが，実際には，Google社のPaaSを試行するのみとなってしまった．
-

3．実績報告（12）

- 3.4.1. 構築･実装1 ：Google App Engine を用いた意見収集･形成システムの試作(miyanishi班) [12]
- 3.4.1.1. アプリケーションとサービス提供者の選択
- アプリケーションとしては，3つのアプリケーションのうちの1つ「意見収集･形成システム」を選択し，クラウドインフラはGoogle社のGAEによるPaaSを用いてアプリケーションシステムを開発することにした．GAEを選択した理由は，
- ①クラウドの原点ともいえる要素技術で成り立っていること，
- ②小規模な利用には料金がかからないこと，
- の2点であった．

3．実績報告（13）

- 3.4.1.2. 要件の確定と非クラウド版の開発（1／2）
- 当アプリケーションについては筆者（宮西）の提案であったので，システム仕様書を作成したが，細部の仕様については，文章で表現するよりも，動作するプログラムを作成したほうが早道であると判断し，GAEで使用可能なプログラミング言語であるJava言語を用いてシステムを作成することにした．（非クラウド，JSP，サーブレットによるシステム）
- システム仕様書は2010年5月末にできあがり，Javaで動作するシステム（非クラウド版システムと本稿では称する）もほぼ同時期（6月初旬）に作成を終わった．これについては文献[5]を参照されたい．
- その後，多少改訂した仕様書及びソースコードをWebに公開している[6][7]．また，余談になるが，Java版とは別にPHP版も作成した．PHP版はJava版システムの機能を絞っている．同じくWebにソースコードを公開している[8]．

3．実績報告（14）

- 3.4.1.2. 要件の確定と非クラウド版の開発（2／2）
- 非クラウド版システムが動作したことにより，このシステムをクラウド化すること（非クラウドからクラウドへの移植）が本プロジェクトとしてのプロトタイプ開発となった．
- 変則的な開発プロセスになったが，今回については，適切な方法であったと判断している．
- なぜならば，本来のシステム開発における不確実さと，GAEという新しい技術を使うことによる不確実さの両方を抱えてしまうより，本来のシステム開発での不確実さを無くする（システムとして動作している状態）ことにより，GAEの利用方法にのみ注力できるからである．

3．実績報告（15）

- 3.4.1.3. GAE環境の整備と動作確認（1／2）
- 2010年7月中旬から下旬にかけて，参考文献[9]およびGoogle社のオンラインチュートリアル[10]を参考に，GAEのローカル環境を自己のPCに設定した．
- この作業は，筆者には大変な作業であった．注意すべき点は，JavaのバージョンとEclipseのバージョンで，この点に苦労が多くあった．筆者のバージョンは，Javaが1.5.0_09であり，Eclipseが3.5.2である．まず，Googleのアカウントを取得し，GAEへのサインアップを行い，Javaの古いバージョンの消去と1.5.0_09のインストール，Eclipseは古いバージョン（3.2）を残したまま，3.5.2をインストールした．そして，Eclipse3.5用のプラグインをインストールした．

3．実績報告（16）

- 3.4.1.3. GAE環境の整備と動作確認（2／2）
- GAE環境整備後，すぐにオンラインチュートリアルのサンプルプログラムGreetingの動作をローカル環境で確認し，これを少し拡張してBookCommentシステム（「書評システム」）を2010年8月上旬に作成した．8月下旬に，Googleのappspot.comにdeployして，クラウドコンピューティングとして動作することを確認した．ローカルからクラウドにdeployすることによる問題はなかった．ソースコードは8月上旬にWebに公開した．現在は8月下旬に行った小修正後のバージョンを公開している[11]．この小アプリケーションにより，KeyValueデータストアにおける，全件表示，1件削除，1件修正，1件追加，1件詳細表示（id指定の特定）の方法を確認した．この小アプリケーションによりKeyValueデータストアの使い方の概要を理解でき，それほど困難なものではないことを実感した．
- 後に判明したが，id指定だけではなく，SQLでの複数属性による検索も可能であるが，この段階で実現していなかった．しかし，それもさほど困難なものではなかった．

3．実績報告（17）

- 3.4.1.4. データストアハンドリングプログラムの作成
- GAEには，MySQLのようにDBMS単独でテーブルを設定したり，レコードを検索，追加，削除するユーティリティプログラムは用意されていない．しかしシステム開発上，このような機能は必須であるので，2010年8月上旬に，自作した．
- この機能は後の開発工程のテスト時に役立った．
- 3.4.1.5. データストアの見直しと移植（1／2）
- 3.4.1.2. で述べた非クラウド版システムでのデータベース設計は，KeyValueデータストアを意識した設計としていたが，やはり実際にKeyValueデータストアを使う段階になると，若干の修正が必要となった．また，この後のプログラム開発においても，データベースの設計（クラウドでは，データストアの設計と称すべきかもしれない）は，クラウドの場合でも従来と同様に重要であると実感した．KeyValueデータストアでは，RDBのレコード（行，ライン，タプル）に相当するものは，1つのオブジェクトであり，スキーマは不要ということになっている．しかしプログラミングの際には，そのオブジェクトの属性を常に意識する必要があり，RDBのスキーマを確定しておくことと，ほぼ同様と思われた．

3．実績報告（18）

- 3.4.1.5. データストアの見直しと移植（2／2）
- 見直しの骨子は，Keyによってのみ検索可能ということであり，特別の（Primaryの）Keyを設けることが必須であると考えた．しかしGoogleのKeyValueデータストアでは，Key以外の属性によるwhere指定ができるので，ほとんどのデータストア検索の場面で，Keyにたよらざるを得ないということはなかった．ただし，Keyをもっておいて便利な場面もあった．
- GAEデータストアではJoin操作ができないことへの対策は，非正規化（データの重複保持）によって行った．システム規模が大きくなってくると注意が必要な点である．
- 移植を終わって[12]，今回のシステムを，http://opvtsystem.appspot.com/に公開している．ソースコードについては，[13]に公開している．また，CEATEC発表の後，簡易的な在庫管理システムを開発し，http://inventory-management-system.appspot.com/に公開している．

3．実績報告(19)

- 3.4.1.6. 評価(1／3)
- 今回の経験について現状の考えをまとめる.
- (1) プロジェクトの技術的目的･目標に対して
- KeyValueデータストアの実際の使い方の習得については，目的を達したが，その他のクラウド技術については，実験を行うことができなかった. 例えば，
- ･弱い一貫性(Eventually Consistency)管理とBASE (Basically Availability, Soft state, Eventually consistency)トランザクション処理，
- (従来はACID(Atomicity, Consistency, Isolation, Durability)トランザクション)
- ･並列処理(MapReduce)，
- について，ビジネス系として今後，利用方法の習得と実証を行う必要がある.
- クラウドサービス提供者の評価についても，次の(2)で述べるが，開発チーム数が多くなかったので，プロジェクトで当初意図した目的は達成できなかった.

3．実績報告（19）

- 3.4.1.6. 評価（2／3）
- (2) プロジェクト推進について
- プロトタイプ開発を実施するか否かは，参画者の自主的判断にお任せしていたので，仙台地区以外では，開発活動は極めて低調であった．この種の**ボランタリー活動の限界**を感じた．しかし，プロジェクトは失敗であったかと問われれば，そうではなかった．実際に開発に携わった技術者には，確実にクラウド技術が身についたであろうし，プロジェクトとして同じ目的に向かったということで，質問や協力しあうことにより，開発の励みになり，開発も促進された．
- **仙台地区での推進方法**が今後の参考になろう．まだまだ検討すべき積み残しも多く，プロジェクトが仙台地区を中心として継続されることを望むものである．

3．実績報告（20）

- 3.4.1.6. 評価（3／3）
- (3) プログラミング
- プログラミング上では，GAEが扱うデータタイプに制限があること，nullの扱い，JSP→サーブレット，サーブレット→JSPのパラメータの受け渡し，Error500の場合の原因追求方法，デバッグ方法など不慣れな面があり，困難性もあった．
- 筆者の方法は，全体的には，データストアおよび図3の画面遷移について，印刷物を常時かたわらに置き，プログラミングを行った．トラブルで多かったのは，上記のプログラム間でのパラメータの受け渡しの間違い，「｝」のもれといったGAE特有の問題ではなく，通常のプログラミング上のミスが多かった．GAE特有の問題として，KeyValueデータストアの使い方については，データストアアクセス方法を何種類かトライすることにより，使用方法を比較的容易に習得できた．それ以外には文字コードについて，注意すべき点があった．
- プログラミング上のミスは，3.4.1.2で述べたように，非クラウド版システムを動作させてからクラウド化する方法をとったにも関わらず，この傾向がでたということは，3.4.1.2によらない場合，一層の苦労が思われる．

3．実績報告（21）

- 3.4.1.7. 今後の課題
- 3.4.1.6.(1)で述べたように，ビジネス系にクラウドコンピューティング方式を適用するには，まだ技術的な見極めで残しているものもある．これらを見極めるために，本プロジェクトを継続したいと考えている．また，開発に着手する個人や組織を増加するための方法について検討する必要がある．

- 今回の報告で，GAEは，おそらく他のクラウドサービスも，取り組むのに，さほどの困難がないことを多くの方に理解いただき，トライアルの輪が広がることを期待する．

3．実績報告(22)

- 3.4.2. 構築･実装2 Google App Engine を用いた意見収集･形成システムの試作(suguri班) [14]
- 筆者(須栗)らは通常のJSP/Servlet/JDBCで作成された意見収集･形成システム [7] を，Google App Engine (GAE) 上に移植した [14]．このシステムは，何らかの話題について提言された意見に対して，コメントを付加し，投票を行うものである[5]．この移植の概要を述べる．

- GAEでも，JSPとServletを利用できる．この点は，従来のWebアプリケーションを開発するのと違いはない．大きな違いは，従来JDBCでRDBを使っていたところで，GAEのデータストアであるBigTableを使うことである．BigTableはRDBとは大きく異なる．ここが，GAEのプログラミングの肝である．

3．実績報告（23）

- 3.4.2.1. 開発環境（1／2）
- IDE として Eclipseを用いた. Eclipse のバージョンはHelios Release (3.6) である． OS には，32-bit の Windows XP, 64-bit の Windows 7, Mac OS X Snow Leopard (Eclipse は 64-bit) を使用した. OSの違いによる開発上の差異は-ほとんど無かった. EclipseでGAE を利用するためには, Google Plugin と Google App Engine for Java SDKを別途インストールする必要がある.

3．実績報告（24）

- 3.4.2.1. 開発環境（2／2）
- Eclipse でまずローカルにプログラムを作成した．ローカル環境でのテストのために，開発用サーバが用意されている．
- 次にJava プログラムをGAE のウェブサイトに配置して動作させた．このサイトを作成するためには，Google のアカウントが必要である．
- GAE のサイト上で動作を確認していたところ，BigTable を使用するとうまく動作しないという現象が発生した．これは，GAE 上でデータストアの作成を行うために必要な時間が，サイトの作成とは別に数時間程度かかるためであった．GAE の管理ページでデータストアの状態を確認できる．ここで作成が完了している状態になれば，配置した Web アプリケーションを動作させられる．また，これ以外の理由でも，実際にGAEのウェブサイト上で動作させるとローカル環境とは異なるエラーを起こす場合もあった．この理由は不明なことが多い．デバッグは試行錯誤になるため困難であった．

3．実績報告（25）

- 3.4.2.2. BigTable
- 既存のアプリケーションをクラウドに移行する際に，どのような問題が発生するかを知る必要がある．とりわけ，データの永続層として，RDBからKey-Value Store (KVS) への移行は，困難が予測される．GAEで採用しているKVSであるBigTableをデータの面から理解することを進めた．BigTableは，永続化オブジェクトと呼ばれるインスタンスを経由して，画面とデータストアの間で表示データをやり取りしている．
- JDBCを用いて作られたアプリケーションをGAEで実行する際に，SQLの代わりにGQLという言語を用いてデータストアに対するクエリーを送信して結果を得る．GQLはSQLに類似した構文を持つ．しかし，GAEでは，RDBでは可能であるjoinを行うことができない．そのため，データストアをリレーショナルに正規化した場合，検索を分けて行う必要があり，効率が悪くなる．BigTableでのデータ構造を設計する際に，joinの代替となる手段を考える必要がある．

3．実績報告（26）

- 3.4.2.3. Joinの代替手段（1／2）
- この代替手段として，次の2点を試行した．
- (1) データ構造を非正規化する．
- (2) GAEのMerge Join機能を利用する．
- (1) は，joinした後と同じ状態の構造を持つテーブルを使う．設計する際に，あまり考えなくて済むので，短時間でデータクラスを作成できる．こうすることの問題は，正規形を崩すため，重複するエンティティが出てくることである．GAEの背後では膨大な数のマシンが稼働しているため，このようにマシンパワーを必要とする処理でも実行させることはできる．しかし，データ更新時の整合性やセキュリティリスクは，開発者が自前で管理する必要がある．また，CPU時間やデータ量をより多く使用するので，より多く課金されることになる．

3．実績報告(27)

- 3.4.2.3. Joinの代替手段(2／2)
- (2) は，複数のプロパティインデックスをマージしながら検索する方法である．こうすれば，リレーションを定義し，データクラス間の整合性を取ることが出来る．これは一般的な関係モデルとは異なる思想のデータ構造である．従って，従来のRDBの設計とは適合しなかったデータの表現ができる可能性もある．しかし，設計のために割かれる労力は増す．費用対効果について注意を払う必要がある．
- (1) と (2) のいずれの方法でも，RDBの設計をそのまま移行することはできない．テーブルの構造については新しく設計しなおす必要がある．それに従って，アプリケーションのロジックも単純に移植するわけにはいかない．前述の通り，(1) と (2) の双方にメリット及びデメリットが存在する．従って，ケースバイケースで手法を選択する必要がある．設計を行うためには，BigTableの構造と機能を理解し，できることとできないことの制限について知る必要がある．実際に業務で使用するプログラムでは，特に更新系のコミットタイミング等で，工夫が必要になると考えられる．

3．実績報告（28）

- 3.4.2.4. 評価
- データストアを上手に利用出来るか否かでGAEの価値が決まる．この仕組みを上手く活用できなければ，GAEを使う利点は存在しない．RDBを用いた厳密な処理が必要か，KVSによる軽量だが精度の低い処理が適しているか，アプリケーションの要求仕様の段階から考える必要がある．
- 現在のところ，提供されている低水準APIだけを用いてアプリケーションを開発又は移植するのは困難である．具体的には，主キーの扱いが難しい，文字コードの変換方法が不明という問題がある．
- GAEにおける開発は，現在のところまだ困難であるが，今後は，更に多様なフレームワークや生産性の高いツールの出現を期待する．また，設計，製造，試験，運用を含めた，総合的なシステムのコストパフォーマンスについて，従来型のシステム開発との比較を行う必要がある．

3．実績報告（29）

- 3.4.3. 構築･実装-3 ：SQLデータベースのGAE環境への移行(kaji班) [15]
- 既存アプリケーションをGoogle Application Engine（GAE）に移行するユーザニーズが今後増加することが予想される．その際に，アプリケーションロジックのみならず，既存SQLデータベースを移行することも大きな作業である．筆者（梶）らはデータベース移行に焦点を当て，GAEでのアプリケーション構築について技術的課題を明確にし，JDOやLowLevelAPIなど異なるアクセス方法について現時点での評価を試みた．
- 3.4.3.1. 研究の目的（1／3）
- 現在利用可能なパブリッククラウドの利用としてGoogle GAEによるPaaSの利用，マイクロソフト提供のWindows Azureによるアプリケーション開発,アマゾン提供のIaaSの利用などが考えられる．

3．実績報告（30）

- 3.4.3.1. 研究の目的（2／3）
- Windows Azureの利用は，従来のWindowsアプリケーション開発スキルがそのまま継承できるメリットがあり，Windows開発に親しんできたベンダーにとっては期待の持てるサービスであるが，Windows Azureの日本での利用可能が2010年2月からと，比較的最近であり，まだSDKの利用や開発に関するマニュアルの提供などがまだ十分といえない現況である．その点ではアマゾン提供のIaaSは現在でも利用可能で，既存の自前で稼動しているシステム環境のイメージをアマゾン上に作成し，移行すればそのまま稼動することができる．新たな開発スキルを特に必要としないのが特徴である．一方Google GAEは開発言語がPythonというなじみの薄いものであったが2010年4月からはJava言語をサポートするようになり，Javaスキルを活かすことが可能となり，技術的観点では興味のある開発環境である．

3．実績報告（31）

- 3.4.3.1. 研究の目的（3／3）
- このような観点から，既存SQLデータベースをGAE環境のKVS(Key Value data Store)テーブルに変換するという前提の下に，技術的留意点，導入上のhint&tips，SQLと比較したときのコーディング上の違い，各データベースの特性などの観点から実証的に取り組み，得られた知見について報告し，スキルと経験知を共有することを目的とした．
- なおこのテーマに対して仙台地元企業4社と宮城大学が参画し，上記の観点から取り組みを分担し，活動を開始した．分担したサブテーマは以下である．
- (SB-1) SQLデータベースによる既存販売管理システムをGAEに移行するという条件の下，移行手順に関する技術的留意点を明確にする．
- (SB-2) 実業務で使用されている自治体の選挙事務システムSQLをKVSに移行するという条件の下，SQL言語がKVSではどのように行なわれるか，比較検討する．
- (SB-3) KVSに対する3つのアクセス方法(JDO，JPA，Low Level API)の特色を明確にし，適用限界について考察する．
- 以下に各々について，報告をまとめる．

3．実績報告(32)

- 3.4.3.2. SB-1の結果(データストアへのアクセス方法)
- 既存アプリを移行する準備のために，データストアにアクセスする方法として3種類を試行した．
- (1) JDO
- データストアを検索する方法に以下の2つの方法がある．
- ① Queryオブジェクト一つずつに条件やソート順等をメソッドで指定(推奨)．
- ② SQLに似た形．一文で記述．(制約が多い．)

3．実績報告（33）

- (2) JPA
- JPAとJDOでコード変更を要する部分は以下である．
- ① persistence.xml(src/MATE-INF) ファイル追加(JPAの使用を宣言)．
- ② EMF.java(src) ファイル追加（JPAでビックテーブルへアクセスするパス）．
- ③ entityのメソッド部分．
- ④ 各種インポート内容．
- ⑤ 検索部分（JDOのSQLに類似）．
- ・移植する前のアプリケーションがJPAを使っている等の理由がない限り，JDOで開発するのが普通．JDOに比べて説明が少ないため，エラーの対処などで困る．
- (3) Low level API
- ・オブジェクトとエンティティのマッピングが提供されていない．大規模なアプリケーションには不向き．だが，GAEの機能を理解するには有用．

3．実績報告（34）

- 3.4.3.3. SB-2の結果（移行時の技術的課題）
- (1) 単純移行
- 既存RDBをBigTable に単純に移行した場合にJDOではSQLで可能であった以下の機能については書き換えが必要であることが確認された．
- ① SQLのCOUNTと同様の処理を行う場合，プログラムでアルゴリズムを実装する必要がある．
- ② SQLのJOINと同様の処理を行う場合，プログラムでアルゴリズムを実装する必要がある．

3．実績報告（35）

- (2) BigTableの特徴を考慮した移行
- BigTableは，RDBのように正規化してなるべく重複したデータ項目を格納しないような設計を目指すのではなく，検索時間を最小にすることに重点を置いた構造になっている．したがって，プロパティが重複値を持っていても,そのほうが検索時のパフォーマンスが良いならば，それは無駄ではないとの考え方のようである．したがって，そのような構造であることを前提してエンティティのキーに何を選ぶかを決める必要がある．単にRDBの主要キーをそのままKVSのキーに設定しても良いパフォーマンスは得られない．むしろ，アプリケーションがデータをどのように参照するかに基づいてキーを決めるなどの設計が必要である．

3．実績報告（36）

- 3.4.3.4. SB-3の結果（GAEでのアプリ開発）
- (1) GAEにおけるデータ操作方法
- GAEのデータアクセスにはJDO, JPA, Low Level API(LLAPI)の3つのAPIが提供されている．各々の特色を列挙すると以下のようになる．
- ① LLAPIの場合
- ・GAE 独自の APIである．
- ・データストアのアクセスに特化しているためシンプル・高速である．
- ・Low Level なため開発が面倒である．
- ・公式ドキュメント含め情報量が少ない．
- ・「多次元ソート済みマップ」と捉えることができる．
- ② JDO, JPAの場合
- ・JCP (Java Community Process) 標準仕様に基づいている．
- ・従来のRDBのようにGAEデータストアを使うことができる．
- ・データストアの機能を十分に生かすには無理があるのでないかと思われる．
- ・spin-up をはじめ，処理が重い．
- どちらを選ぶにしてもメリット／デメリットがある上，KVSの特徴を理解した上で使いこなす必要がある．

3．実績報告（37）

- (2) GAEでのアプリケーション開発（1／2）
- フレームワークおよび開発環境，テスト方法についての特色をまとめると以下のようになる．
- ① フレームワークについて
- ・EclipseなどのSDK を使った開発は，勉強としては良くても管理など大変そうである．
- ・本格的な業務アプリを作るためには何らかのフレームワークを利用したほうがよさそうである．
- ・GAE上で使うには既存の汎用フレームワークでは無理がある．
- ・GoogleはGWT (Google Web Toolkit) を勧めているが，評価はまだ定まっていない．

3．実績報告（38）

- (2) GAEでのアプリケーション開発（2／2）
- ② 開発環境･テスト方法について
- ･テキストエディタによるCUI開発は軽く手軽だが，生産性はやはり落ちる．
- ･デプロイやリファクタリングを考えると統合開発環境(IDE)のメリットは大きい．
- ･せっかくならテストも何らかのフレームワークが要望される．
- このテーマについてはまだ十分評価できていないが，現時点で評価するならば「Eclipse + GAE/Java + GWT + Slim3」[16]が良さそうであると評価できる．今後もう少し実証評価を進めていくことが必要である．

3．実績報告(39)

- 3.4.4. 構築･実装4 ：Google App Engine を用いた部品庫システムの試作(sakashita班)
- 製造の各段階で発生し管理される生産管理データ(PD; Product Data)を一貫した方法で管理運用(PDM; PD Management)を行う部品庫システムは，製品構成と製造工程に関する基準情報および関連する履歴情報を扱うBOM(Bill of Material)の一部と位置づけられるとし，この設計と開発をGoogle App Engine (GAE) 上にて実施することは，今後各種の情報システムを検討する上で有用であると考える．

3．実績報告（40）

- 筆者（坂下）らは，通常のクライアント・サーバ形態で作成された部品庫システムを，GAE上に再設計開発した．このシステムは，製造用の設計図における部品表のデータから，その製品を製造する際に必要な部品を倉庫から引き当てることを行うものである．この開発の概要を述べる．
- 本開発では，電子回路基板設計の場面を対象とした使用部品の仕様の特定を，現在の在庫状況を参照しながら行う形態のシステム構築を行うことと，文書の形式で多く存在する各部署におけるデータ及び情報をデータベース化するツールの製作[17]，更に文書内に多く存在する表形式の関係データのデータベース化と文書へのフィードバック機構の構築を目指した．

3．実績報告（41）

- 3.4.4.1. 部品庫(PDM)システムの概要
- (1) 設計工程における部品仕様の特定
- 設計工程における基本的な機能である．ここでの主な機能は部品番号を付けることと当該部品の規格などの仕様を決定することである．多くの部品はその企業が備える部品データベース(DB)に登録されている．このDBを検索する際に使用するキーワードや種類･機能･諸特性を，同時に設計情報として部品表のデータとして利用する．
- (2) 他部門データの参照
- 関連担当部門間で互いのデータを参照する形態で，作業者の意思決定の下で実施する作業形態とした．例えば，資材部門が調達している部品情報を設計部門が参照できる形とし，設計者により，当該部品を決める仕組を提供する．

3．実績報告（42）

- (3) 部品データベース
- 企業は製造に必要となる部品は，標準在庫品としてその多くを在庫部品DBに登録して保管している．設計部門は，基本的にこの標準在庫品に登録されている部品を優先的に使用し，必要に応じて新たに購入することを選択する．（資材）管理部門は，在庫部品が枯渇しないように製造の状況を鑑みながら調達し，調達が必要になった部品を調達する．
- 以上(1)から(3)について図3に示す．

3．実績報告（43）

- 以上(1)から(3)について図3に示す．

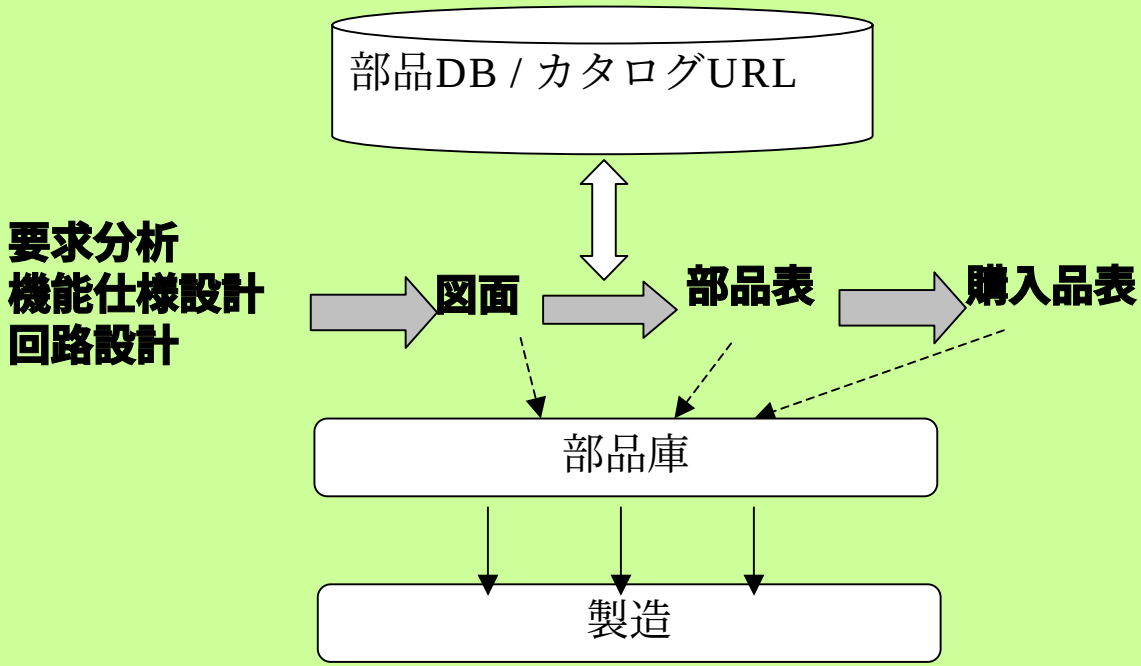


図3 部品庫システムの概要

3．実績報告（44）

- 3.4.4.2. 開発環境と手順
- (1) 開発環境
- IDE として Eclipse(Helios Release v3.6) を用いた.　EclipseでGAE を利用するために, Google Plugin と Google App Engine for Java SDKをインストールした.
- (2) 仕様定義
- 設計者は，設計した図面に必要な部品の仕様･規格と数量を決定し，部品表と呼ばれる表にまとめる．そして，製品に対する生産命令がオーダー（工事命令）として発令されると，その製品に必要な設計図面が決まり，各図面で指定されている部品の種類と数量が決まり，そして，このオーダーに必要な部品の種類と数量の全体が決まる．即ち下記の順序関係となる．
    - 部品⇒部品表⇒設計図(品番)⇒製品⇒製造オーダー

3．実績報告（45）

- 仕様定義した主なものは以下である．
- ① オブジェクト図
- ② JSP
- ③ Server 機能
- ④ データ定義
- 図4に，部品表登録と部品項目データ入力の流れをJSPに適用した一部を示す．

3．実績報告（46）

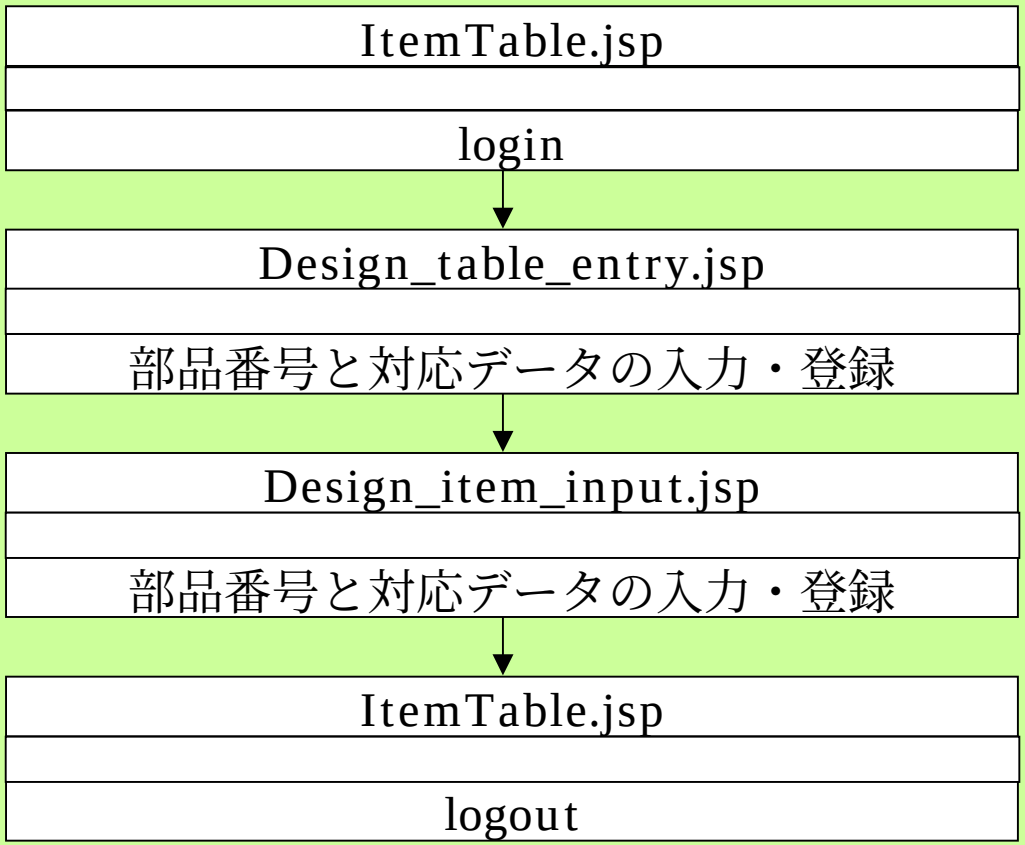


図4 部品表登録と部品項目データ入力の流れ

3．実績報告（47）

- (3) BigTableへの対応
- 既に多くの個所で指摘されている課題である．本開発で対象とした部品表とデータベースにおいては，極めて単純な検索の範囲内で済んでいる．BigTableに対しては，従来のRerational　DBMSの場合とは大きく異なり，後での検索のし易さを考慮した形態のタプル形式のデータにして，格納した．利用側のアプリケーションは，キー（この場合は，規格）になる注目するデータ項目に対応するデータをデータベースから引き出し，必要に応じて必要なデータ部分をそのタプルから取り出す，という極めて単純な方式を採用した．その一例を表1に示す．
- なお，BOMを扱う場合は不定貫のことを考慮するのが一般であるが，今回の実装実験では考慮外とした．

3．実績報告（48）

表１ 部品庫のデータ（一部）

規格（仕様）	名称	種類	在庫数	メーカ名	メーカ型番
120Ω1/4W	角型抵抗	抵抗	123	N	ERJ3GEYJ102V
22Ω1/2W	金属皮膜抵抗	抵抗	240	F	ERX12LJ22M
422μF	フィルム型	コンデンサ	250	P	ECHU1C121GX5
220μF	電界型	コンデンサ	223	P	EEFFD0D101R

3．実績報告（49）

- 3.4.4.3. GAEを利用する際の留意事項
- 現在の経験量では，我々は，普遍的な事柄を述べるレベルにはまだないが，開発を通して気がついたことを以下に挙げる．
- (1) 開発の方法
- 基本的には，開発環境側で従来のクライアント･サーバ型のシステムを構築し，動作確認を行ってから，クラウド側にdeployするので，従来方法とは大きく変わらない．このことは，ある意味ではクラウドを利用して開発するという状況にはなってないので，開発手法としては，クラウドの環境を生かしてないといえる．即ち，直接開発の方法を考えたい．

3．実績報告（50）

- (2) データストアとしてのBigTable
- 本開発では，前述のように一般的な意味でのデータ操作を行った探索は行ってない．直にこのBigTableをアプリケーションが操作する場合は，今回のように，他が利用しやすい形態のタプルの形にするのが安全であろう．しかし，従来のRDBを駆使するサービスを組む場合は，正規化されたDBとこのBigTableを操作する仲介役のAPIの類が必要と思われる．これは，半構造を扱うXMLデータベースの領域でも指摘されていることにも通じる．
- 表1で示した方式では，部品表へのエントリ作業では要求を満たすが，実作業では設計者が様々な項目をキーにしてカタログあるいは部品データベースを多面的に探索する．この場合は，インデックスの構造に留意して設計する必要がある．

3．実績報告（51）

- (3) JDOとエンティティのライフサイクル
- JavaBeansとは異なりエンティティにライフサイクルがあり，その状態は以下である．
- ①　transient→データストアに反映される前の状態．
- ②　persistent→データストアに反映された状態．
- ③　detached→”detached”が設定されている状態で，PersistentManagerをcloseした状態．
- ④　hollow→”detached”が設定されてない状態で，PersistentManagerをcloseした状態．
- データがデータストアに反映されるのは，トランザクションをコミットしたとき，およびPersistent Managerをcloseしたときとなる．また”detached”の状態では，エンティティはセッションに格納することによって複数のリクエストにまたがってその状態を維持できるようだが，このことは未実施である．
- JDOを使用したデータストア操作では，このライフサイクルに注意する必要があると予想される．

3．実績報告（52）

- 3.4.4.4. まとめ
- 部品庫システムという極めて小さい簡単なシステムをGAE環境を利用して，試験的に実装した．十分な経験とは言えないが，GAEの一端に触れることができた．一方で，我々は類似のシステムを別の環境であるsalseforce.com上に実装する試みも行っている[18]．
-

3．実績報告（53）

- 3.5. 今後の展開（1／2）
- 今年度は，GAEを用いた実装を4件行うことによって，これに関する一定の知見を得ることができた．他方「共通の要求仕様を異なるクラウドプラットフォームで実装することによって各々の利点及び欠点を知る」という目標は達成できなかった．参加者の関心が，GAEに集中していたためである．
- 今後の活動予定として，次の案がある．一部は仙台地区を中心に既に試行を開始している．
- (1) GAE
- ・Pythonによるアプリケーション開発．
- ・Slim3フレームワーク [19] によるアプリケーション開発．
- ・Slim3に替わるフレームワークの開発．
-

3．実績報告（54）

- 3.5. 今後の展開（2／2）
- (2) GAE以外
- ・Hadoopによる分散処理.
- ・EucalyptusによるAmazon互換IaaS.
- ・Windows Azureの使用.
- ・分散セキュリティ [20].
- 新たな参加者, 及び新たなテーマは常時募集している.

4．CEATECへの対応（1）

- 4. CEATECへの対応(片岡)
- CEATEC JAPANの開催期間中 (2010年10月5日(火)～9日(土)幕張メッセ) に電子情報通信学会の研究会が開催された.
- CEATEC JAPANは，電子情報通信系最大規模の展示会であり，これと連携することで，電子情報通信分野で活躍する企業技術者に本学会活動をPRすることを目的としたものである.
- さらに，CEATEC JAPAN展示会に参加した技術者が研究会に参加する，あるいは，研究会に参加した研究者がCEATEC JAPAN展示会に参加するといった相互交流を期待したものである. 広く電子情報通信分野の活性化につながる試みとして行われた.

4．CEATECへの対応（2）

- 4.1. 研究会開催までの経緯
- 2009年9月7日理事会で「企業向け学会価値向上策検討WG」での検討結果として2010年10月のCEATECで電子情報通信学会の研究会の開催を試行することが決定された．11月26日のISS拡大運営委員会，技術会議において，各研究専門員会に予算的な負担を掛けないことを前提に，ISSの各研究専門委員会に企画の提案依頼がだされた．企画は専門委員会の活動が，企業，社会に広く伝わり，会員確保につながることを狙ったものである．
- SWIM研究専門委員会は，2009年12月10日の専門委員会で参加を決定し，2009年度より開始したSWIM研究専門委員会のプロジェクトである，CCTP(Cloud Computing Trial Project)の成果を中心に発表することとした．テーマ名を「エンタープライズにおけるクラウドコンピューティングの『真髄』（後に『適用可能性』に変更）（チュートリアル，パネル議論）」とした．
- 2010年1月にSC(サービスコンピューティング)研究専門委員会も類似のテーマがあるとのことで，合同での開催に変更した．研究会としは，SCが第2種研究会(時限研究会)のため，今回の共催研究会は第2種として開催することとなった．
- 開催は，2010年10月7日(木)に幕張メッセの国際会議場で開催され普段の研究会の倍程度の80名弱の参加者を得ることができた．

4．CEATECへの対応（3）

- 4.2. 連携企画の内容
- 連携企画全体としは，下記の5つの基調テーマのもとで研究会が開催された．
- 信学会全体として4つのテーマ
- (1) 基調テーマ1：Digital Harmonyを支える次世代ディスプレイ
- (2) 基調テーマ2：次世代ネットワークから新世代ネットワーク
- (3) 基調テーマ3：クラウドコンピューティング
- (4) 基調テーマ4：福祉と見守りのための画像音声処理
- 当研究会は，第2種研究会として開催したため各発表は学会の出版物となっていないが，発表のPPTに関しては参考文献[21]から[23]を参照されたい．

4．CEATECへの対応（4）

- 4.3. 研究会発表概要
- 基調テーマ3のもと，SCと合同で3つのセッションを行った．
- (1) セッション1：SC研究専門委員会主体
- テーマ 「Webサービス，クラウドの先へ：サービスコンピューティング研究が拓く世界」 4件の発表[21]．
- (2) セッション2：SWIM研究専門委員会主体
- テーマ 「エンタープライズにおけるクラウドコンピューティングの適用可能性(チュートリアル招待講演)」および3件の発表[22]．実際は，チュートリアルではなく事例発表とした．
- (3) セッション3：SC研究専門委員会，SWIM研究専門委員会合同パネルデスカション[23]．
- 当研究会は，第2種研究会として開催したため各発表は学会の出版物となっていないが，発表のPPTに関しては参考文献[21]から[23]を参照されたい．

4．CEATECへの対応（5）

- 4.4. パネルディスカッションの内容紹介
- 各発表内容は紙面の都合上割愛するが，パネルデスカション各パネリストの発表内容について以下紹介する．
- ・司会 浦本直彦氏(日本IBM)
- ・青山幹雄氏(南山大学)（SC側）
- 「クラウドコンピューティングの現状と今後」と題してクラウドの進化モデルとSOC/SOAによるクラウド統合のモデル，またSOAによるクラウド統合の事例，統合の課題についての議論があった．
- ・横山重俊氏(国立情報学研究所)（SC側）
- 「使いやすいアカデミッククラウドを目指し」と題して国立情報学研究所で運用中のedubase Cloudの構築思想や状況について議論があった．他人に影響を与えることなく自由に自分のクラウドを構築できることが特長である．
- ・林雅之氏(NTTコミュニケーションズ)（SWIM側）
- 「クラウドコンピューティングについての国内外の最新状況と今後」と題して広く国内外の状況の説明があった．特に米国政府に対するGoogleやMicrosoftの取り組みの議論があった．
- ・森正勝氏(日立製作所)（SWIM側）
- 「クラウドコンピューティングの現状と今後」と題し日立製作所が目指す社会インフラとしてのクラウドの説明があった．特に，知識指向サービス(KaaS)の提供についての議論があった．

5．全体のまとめ(5)

- 5. 全体としてのまとめと今後
- 今回のCCTPはプロジェクトとしては，必ずしも所期の成果を得ることができなかった．理由は資金，権利，義務とほとんど関係のないボランタリーとしてのプロジェクトの限界でもあろう．
- しかし，仙台地区のように，産業界から多くのかたがたが本プロジェクトに参加したケースもあった．
- 本文にも述べたように，クラウドコンピューティングがエンタープライズ用途で実用になるためには，
- ・GAE以外のクラウドプラットフォームについては，本プロジェクトとしては，未経験である．さらなる知見が必要である．
- ・リアルタイムでの厳密な整合性 (consistency) への要求にどのように答えるか．
- ・外部に預けているデータとプログラムをどのようにして保護するか (セキュリティ問題) ．
- ・効率的なソフトウェア生産方法，あるいは効率的なソフトウェアの再利用，組合せ方法 (ソフトウェア部品組合せ，またはサービス組合せによるシステム構築方法) ．
- ・異種クラウド間のインターオペラビリティ．
- ・異種クラウド間での共通的なソフトウェア生産，再利用方法
- ・広大な範囲からのソフトウェア部品の探索方法．
- ・ソフトウェア部品の汎用的な登録方法．開発されたソフトウェア部品のサービス化．
- といった今後とも我々技術者が取り組むべき研究課題がまだ多く残されている． 関係各位のさらなる取り組みを望むところである．

参考文献（1）

- [1] http://www.meti.go.jp/committee/materials2/downloadfiles/g100423a07j.pdf
- [2] http://www.ieice.org/~swim/jpn/CCTPPlan.pdf
- [3] 宮西洋太郎, “クラウドコンピューティングとどう向き合うか～クラウドはビジネス系情報システム構築技法の革命となりうるか～, ”電子情報通信学会研報, Vol.110, No.70　pp.1-6, June 2010
- [4] 宮西洋太郎, “クラウドコンピューティングトライアルプロジェクトの概要～ソフトウェア産業の繁栄のために～, ”電子情報通信学会研報, Vol.110, No.70　pp.51-56, June 2010
- [5] 宮西洋太郎, “世論形成のための意見収集･形成システムの試作～汎用的意見交換Webサイト～, ”電子情報通信学会研報, Vol.110, No.184　pp.17-22, August 2010
- [6] システム仕様書「意見収集･形成システム」バージョンJ http://isem.co.jp/documents/cloudcomputing/OpinionGatheringSystemJ.pdf

参考文献（2）

- [7]「意見収集・形成システム」ソースコード（zipファイル）非クラウド Java 版 http://isem.co.jp/documents/cloudcomputing/OpinionGatheringSystemV1.5_20100709_workspace.zip（保存をクリック）
- [8]「意見収集・形成システム」ソースコード（zipファイル）非クラウド PHP版 http://isem.co.jp/documents/cloudcomputing/OpinionVoteSystemPHPVersion2.1_20100718.zip（保存をクリック）
- [9] (株)グルージェント，「Google App Engine for Java [実践]クラウドシステム構築」技術評論社，2009年9月
- [10] グーグル社オンラインチュートリアル http://code.google.com/intl/ja/appengine/docs/java/gettingstarted/
- http://code.google.com/intl/en/appengine/docs/java/gettingstarted/
- [11]「書評システム」ソースコード（zipファイル）クラウド版 http://isem.co.jp/documents/cloudcomputing/workspaceforGAEBookCommentSystem20100828.zip（保存をクリック）

参考文献（3）

- [12] 宮西洋太郎, “クラウドコンピューティングによるアプリケーション試作開発事例～トライアルプロジェクト中間報告～, ” 電子情報通信学会 CEATEC JAPAN 2010 連携企画研究報告, pp. 23－30, October 2010.
- [13]「意見収集･形成システム」ソースコード（zipファイル）クラウド版 http://isem.co.jp/documents/cloudcomputing/workspaceforGAE_OpinionGatheringSystemVG1.0_20100902.zip（保存をクリック）
- 動作するシステム： http://opvtsystem.appspot.com
- 追加したアプリケーション, 在庫管理システム：http://inventory-management-system.appspot.com/
- [14] 須栗裕樹, 庄司諒, 前野亨, 柴崎健一, 齋正寿, 庄司健太, “Google App Engine を用いた意見収集･形成システムの試作, ” 電子情報通信学会 CEATEC JAPAN 2010 連携企画研究報告, pp. 31－35, October 2010.

参考文献(4)

- [15] 曽根碧, 金村昌秀, 地主雅信, 梶功夫, “GAE環境におけるSQLデータベース移行に関する技術的考察,” 電子情報通信学会 CEATEC JAPAN 2010 連携企画研究報告, pp. 36－40, October 2010.
- [16] ひがやすを, 小川 信一, “Slim3 on Google App Engine for Java, ” 秀和システム, 2010年7月
- [17] 坂下善彦, 齊藤充, 東海林和弘, “製造関連データを生産物データベースへ登録する変換ツール, ”情報処理学会第68回全国大会, March 2006
- [18] 坂下善彦, 平雄佑, “SaaS環境のクラウドコンピューティングのPDM (部品表) の試作, ”電子情報通信学会技術研究報告, Vol.110, No.427, pp.45-50, February 2011
- [19] http://sites.google.com/site/slim3appengine/
- [20] 宮西洋太郎, “クラウドコンピューティングにおける高度セキュリティ実現方式の提案　～ 情報処理委託内容の秘匿方式 ～,” 電子情報通信学会技術研究報告, Vol. 110, No. 302, pp.13-17, November 2010.

参考文献（5）

- [21] Webサービス、クラウドの先へ：サービスコンピューティング研究が拓く世界
- http://langrid2.nict.go.jp/sc/seminar_list.html#ceatec2010
- [22] エンタープライズにおけるクラウドコンピューティングの適用可能性
- http://www.ieice.org/iss/swim/jpn/presentations/
- [23] パネルデスカション：クラウドコンピューティングの現状と今後
- http://www.ieice.org/iss/swim/jpn/presentations