筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

InfoFrame Relational Store を利用した研究開発 － 性能予測モデルの策定およびマイルストーンを用いた進捗管理 －

菊池大輔

修士（工学）

(コンピュータサイエンス専攻)

指導教員　田中二郎

２０１３年　３月

概要

企業や社会で扱われるデータの量は年々増加している．その中で，データ量の急激な増加やアクセス数増加に対応できるようなデータベースシステムが求められている．その様なシステムの 1 つとして，NEC 様にてデータベースシステム InfoFrame Relational Store（以下 IRS）が開発された．このシステムは，スケールアウト性が高く，SQL を扱う既存システムを活用することも可能である．我々は，IRS の高いスケールアウト性を更に活かすために，IRS の性能予測システムと，予測値導出のためのモデルを作成する予測式計算システムの開発を行った．このシステムにより，IRS を使用したい顧客の方々や，IRS の開発者，現場の SE の方々が，IRS のサーバ台数による性能見積りを行なうことが可能となる．筆者は本プロジェクトで，主にマネジメント担当として活動し，プロジェクトの進捗を管理，決定することで，プロジェクト活動の円滑化を行った．また，システム開発においては性能予測モデルの策定を行うことで，本プロジェクトで開発する予測式計算システムにおいて IRS をモデル化するモジュールと，性能予測システムにおける性能値計算を可能にした．

目次

図目次

表目次

第1章　はじめに

企業または社会が扱うデータの量は年々増加している．それに伴い，データベースの性能に対する要求も高くなっている．従来から長く用いられている RDBMS(Relational Data Base Management System)[1]では，データの整合性を保つことは可能であるが，サービスの大規模化による急激なデータ量増加に耐えられないため，データをある程度捨てて対応する必要がある．従来は，データの取捨選択を行なって対応をしていたが，データ量が増加した現在では，あらゆる情報資産を利活用した価値の創出がビジネスに大きな影響を与える．よって，急激なデータ量の変化に対応でき，かつ BigData で扱われる大容量データを，整合性を保ちつつ，また可用性を考慮しながら保持できる様なシステムが必要となる．

データベースの処理能力や格納できるデータ量といった性能の向上を行なう方法として，スケールアップとスケールアウトの 2 つの考え方がある．

スケールアップとは，データベースサーバの性能自体を高めることによって性能を上げる方法である． RDBMS では，この方法を用いて性能の強化を行なっている．RDBMS はデータの整合性を保つ仕組みを有しているが，スケールアップによる性能の強化に上限がある上，ハードウェアに高性能を求める場合高価になりやすくコストがかかるといったデメリットがある．

一方，スケールアウトとは，ノードを増やすことにより性能を上げる方法である．分散 KVS(Key-Value Store)[2]ではこの方法が用いられる．Amazon の社内プロジェクトである Dynamo[3]では，この方法を用いて，大容量データの保持と可用性の向上を実現している．この方法であれば，保持可能なデータ容量の大きさの頭打ちが起こりにくいという特徴があるが，データの整合性を保つことが難しいという弱点もある．

InfoFrame Relational Store（以下 IRS）[4]は，NEC 様が開発したデータベースシステムであり，スケールアウトという形で性能強化を行なう上，トランザクションを処理可能なシステムである．データ量増加に伴って柔軟に拡張可能で，かつデータの整合性を保つ仕組みを有している．この度，筑波大学高度 IT コース博士前期課程 2 年次の科目である研究開発プロジェクトに対し，NEC 様より，IRS を利用した研究開発というテーマをご提供頂いた．そのテーマを選択した筆者を含む 3 人の学生によって，プロジェクトを遂行する．

本プロジェクトの目的は，IRS を用いるか，または IRS を扱う上で役に立つシステムを，NEC 様に対して提案し，開発することにある．

- IRS を活用する
- 新規性がある
- 大規模データを活用する

本プロジェクトは，IRS の開発元である NEC 様の助言，要望を頂きながら，筑波大学の学生チームがシステムの開発を行う．本開発の関係者とその役割を表 1-1 に示す．開発に際して，NEC 様より技術的支援と助言を頂く．筑波大学からは，NEC 様に対し開発するシステムの提案を行なった後，実際に開発した成果物を提出する．メンバ内では，それぞれの仕事，学んだ技術等について共有しながら，システムを提案し開発を行なう．

本プロジェクトでは，IRS の性能を予測するためのシステムの開発を行なう．IRS は複数サーバで構築され, サーバ台数や, 各サーバに割り当てる役割によって処理性能が決定する．我々は，それらの構成を入力として，性能予測値を出力するシステムを開発することで，IRS のスケールアウト性をさらに活かすことが出来ると考えた．

本報告書は，本プロジェクトの開発する IRS 性能予測システムと，プロジェクトの進捗，筆者個人のプロジェクトへの貢献についての報告書である．本報告書ではまず，IRS の特徴について述べる．次に，本プロジェクトで開発するシステムについて説明する．その後，本プロジェクトの進捗について述べ，最後に，筆者がプロジェクトで貢献した事，プロジェクトの振り返りについて述べる．

表 1.1　開発体制図

所属	氏名
NEC 様	白石 雅己様
	川畠 輝聖様
筑波大学	村上 貴裕　　(リーダー・技術担当)
	菊池 大輔(筆者) (マネジメント担当)
	程　 鵬　　(成果物担当)
	早瀬 康裕先生　(情報工学域 助教)

第2章　IRS の特徴

IRS は，RDBMS の信頼性と，KVS の拡張容易性を併せ持ったデータベースシステムである． RDBMS は高い信頼性を持つ一方，データ量の増加，アクセス数の増加といった場合の拡張に対応できない．また，KVS は，データ量増加には対応できるが，データの整合性を保つ面で弱い．IRS は，それらの利点を取り入れ，データ量増加等への対応ができ，かつ高い信頼性を持つシステムである．

2.1　IRS の構成と特長

IRS は，Partiqle サーバ，トランザクションサーバ，ストレージサーバの 3 層で構成されている． IRS の構成を図 2-1 に示す．IRS の現行バージョンである v2.1 では，どのサーバも Red Hat Enterprise Linux 6.1 x64 上で動作し，必要メモリ 16GB 以上，必要ディスクは 100GB 以上となっている．IRS は高信頼ネットワーク内で構成され，各種サーバが複数台ずつ稼働する．以下，各サーバについて説明する．

Partiqle サーバは，アプリケーションと IRS を SQL インターフェースでつなぐ役割を担っている．まず，アプリケーションに対し，SQL インターフェースを提供する．また，アプリケーションが発行する SQL を解析し，KVS へのアクセスへと変換し，トランザクションサーバに転送する．

トランザクションサーバは，トランザクションデータをある一定の単位毎に保持するデータストアである．ログ先行書き込み(以下 WAL)手法[5]によって，トランザクションデータをインメモリ DB に格納し，その後，ストレージサーバへと非同期にデータの更新を行う．また，保持しているデータについて Partiqle サーバへとデータの供給を行なう．トランザクションデータを保持するためのデータベースとして，分散 KVS であり，Amazon Dynamo のクローンである Voldemort[6]が動作する．TAM セットと呼ばれる 3 台以上のノードのクラスタを定義し，その中で 3 台ずつのレプリケーショングループを定義する．レプリケーショングループは，トランザクションサーバ内に定義する WAL インスタンスによって形成さ

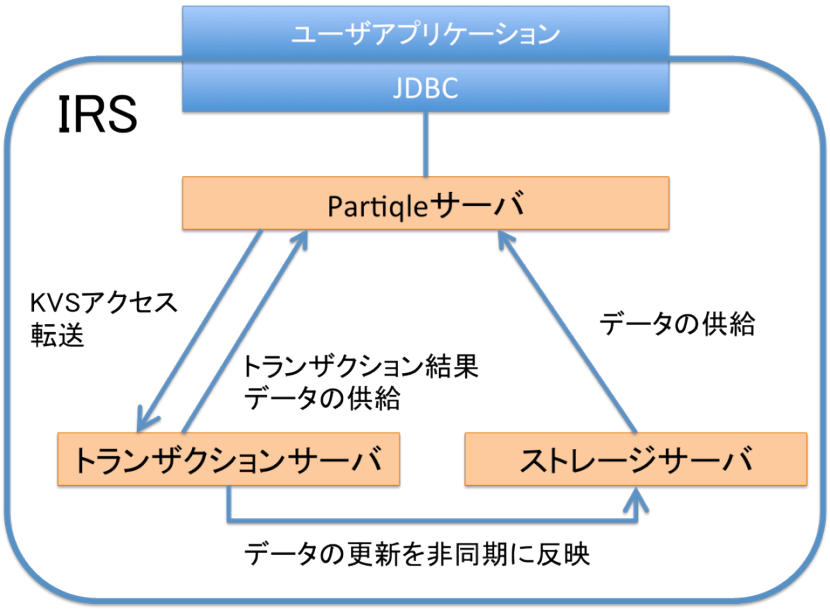


図 2-1　IRS の構成

れる．ストレージサーバへの反映が終了した後，トランザクションサーバ内の反映が終了したデータを破棄する．

ストレージサーバは，実データを格納するデータストアである．トランザクションサーバから非同期で転送されたログデータを反映する．また，Partiqle サーバに対して，その要求に応じてデータの供給を行う．KVS には，Voldemort と，ストレージとして Oracle Berkeley DB[7]を用いる．複数サーバによるクラスタ構成としており，実データを多重保存する．

IRS には，高いスケールアウト性，更新性能，システム構築の容易性，基幹業務に耐える信頼性といった特徴がある．

高いスケールアウト性については，サーバの増強が必要となった際に，ハードウェアを追加して数 10 分で完了する．スケールアウトの概要を図 2-2 に示す．各サーバの負荷が大きくなった場合に，追加したいサーバを，IRS を構成するネットワーク内に配置し，コマンドを実行することで，サーバが追加され，さらに負荷分散の機能によって，追加したサーバも含めた処理の最適化を行う．

また，IRS は，トランザクション処理を高速に行う仕組みを有しており，更新性能に優れている．ユーザアプリケーションから SQL を受けた Partiqle サーバは，トランザクションサーバに対し，受けた SQL に対応する KVS アクセスを転送する．トランザクションサーバでは，複数レコードの更新イメージを WAL にため，コミットを実行した時にそれらをまとめて 1 回の書き込み操作でトランザクションを行う．それにより，スループットが向上している．

システム構築の容易性については，IRS が SQL インターフェースを提供しているため，SQL を扱う既存資産を有効活用することが可能となっている．

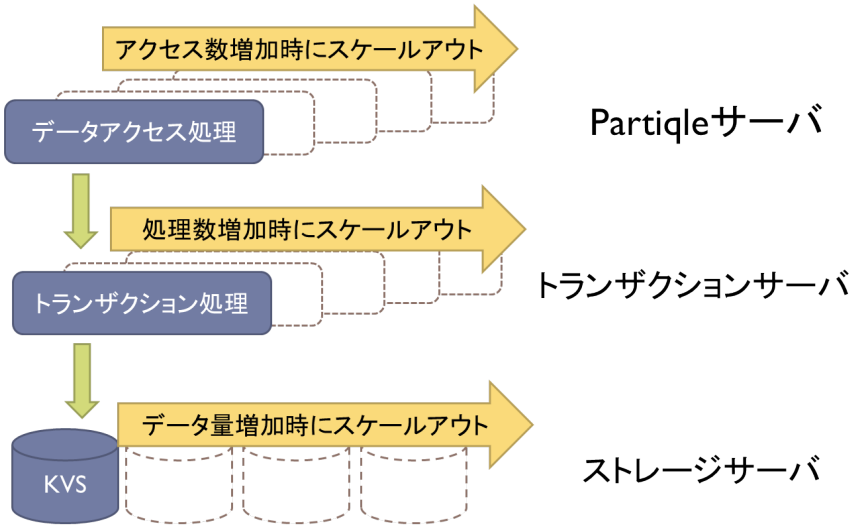


図 2-2　IRS のスケールアウト概要

2.2　IRS のスケールアウト性

IRS は，特にスケールアウト性に優れたシステムであり，各サーバの台数を増加させることが可能である．

Partiqle サーバは，スケールアウトによってアプリケーションからの接続数増加に対応可能である．以下，Partiqle サーバのスケールアウトの概要図を図 2-3 に示す．Partiqle サーバが増加すると，各サーバの処理が均等に割り振られるため，各サーバの負荷が小さくなる．Partiqle サーバは，アプリケーション側から SQL のクエリを受け取った後，KVS アクセスに変換し，トランザクションサーバに KVS の要求を渡す．Partiqle サーバの台数が少ない場合は，発行される SQL クエリが多くなった場合に，それを変換する処理や，トランザクションサーバに要求を渡す処理による負荷が高くなるが，Partiqle サーバを増やすことによって，その負荷を分散することが可能である．

トランザクションサーバは，スケールアウトによってトランザクション量増加に対応可能である．以下，トランザクションサーバの構成を図 2-4，トランザクションサーバのクラスタ内に定義する TAM セットの構成を図 2-5 に示す．トランザクションサーバは，TAM セットと呼ばれるクラスタが幾つか集まって構成される．TAM セット内には，3 台以上のノードが存在する．1 台のノードは幾つかのインスタンスを保持しており，TAM セット内の 3 インスタンスによってレプリケーショングループが構成される．

トランザクションサーバの処理は，レプリケーショングループ単位で行われる．つまり，レプリケーショングループの数が増加すれば，IRS のトランザクション処理性能が上昇することとなる．

トランザクションサーバをスケールアウトさせる方法には 2 つの方法がある．TAM セットと呼ばれるノードの集合を増やす方法と，TAM セット内にスレーブインスタンス用のノードを追加して，マスタインスタンスの負荷を減らす方法である．TAM セットと呼ばれる

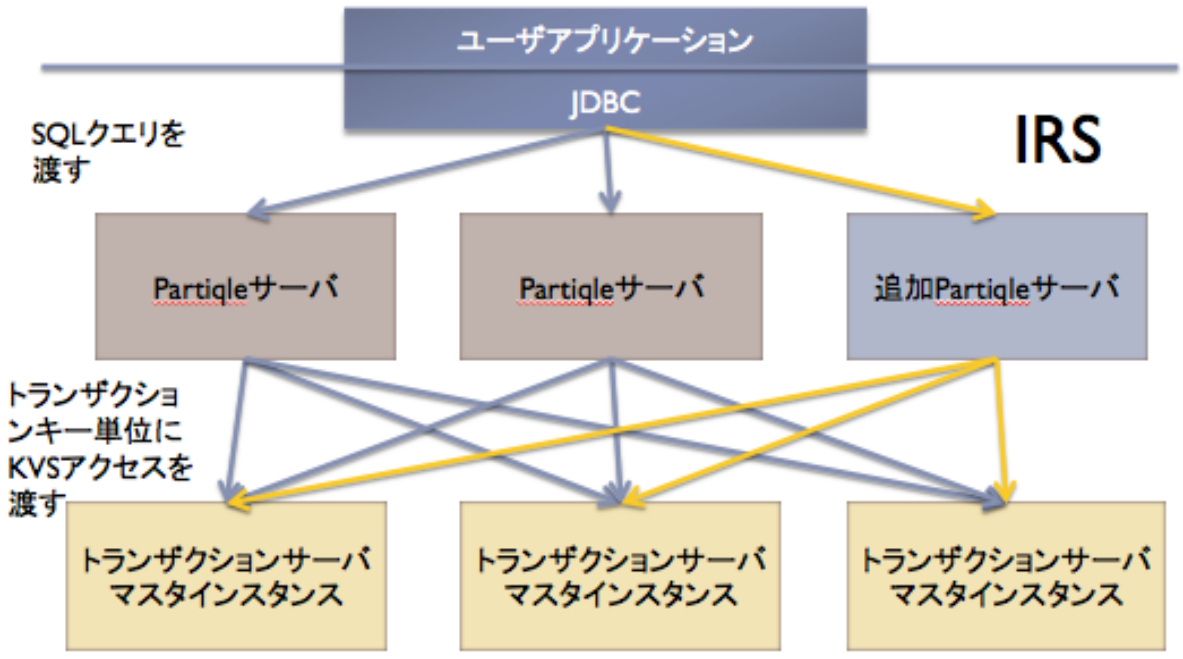


図 2-3　Partiqle サーバのスケールアウト

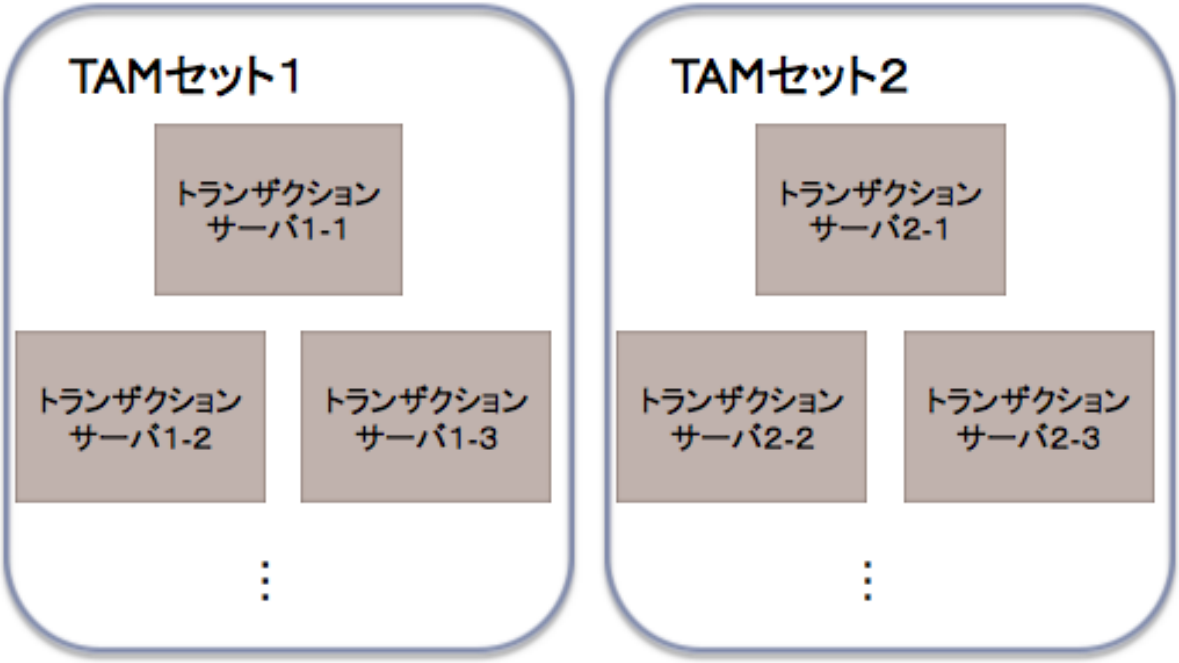


図 2-4　トランザクションサーバの構成

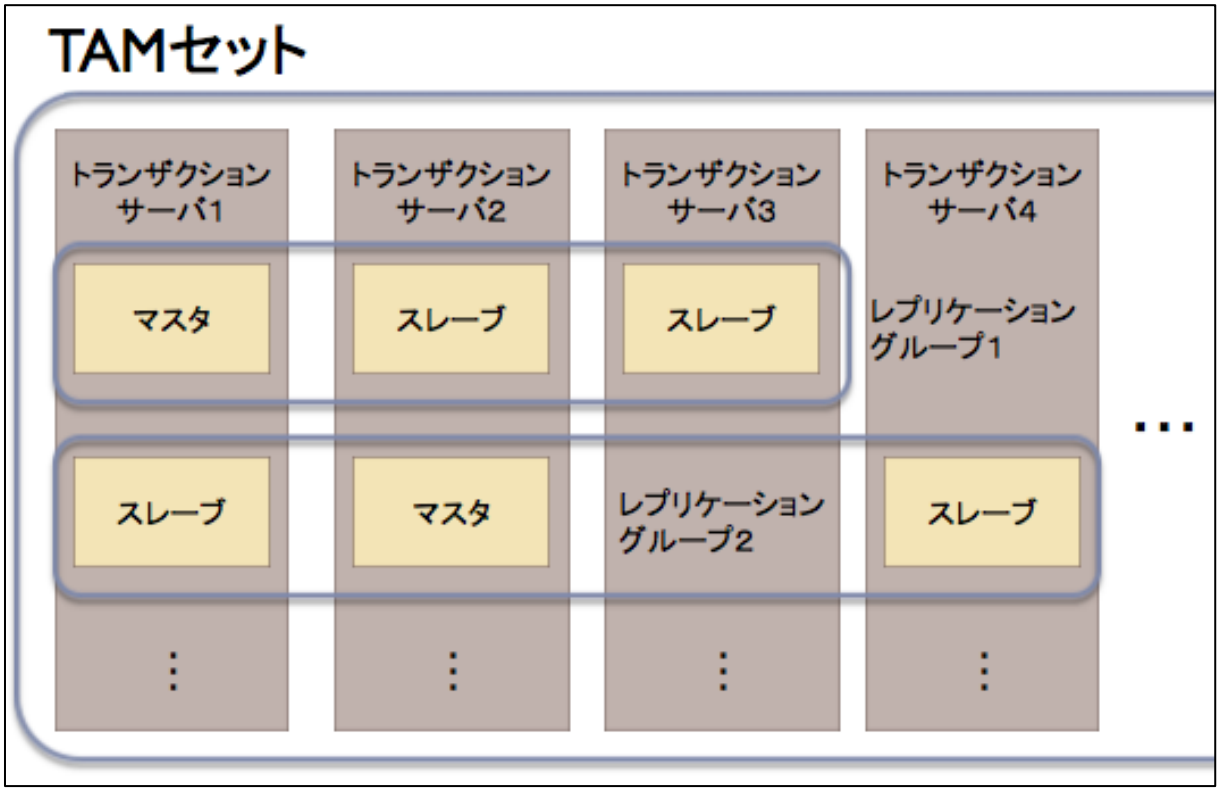


図 2-5　TAM セット内の構成

ノードの集合を増やす方法では，レプリケーショングループを構成するために最低 3 台ずつ増やす必要がある．この方法は，用いているサーバのメモリ容量が不足するか，そもそも搭載メモリ容量が小さく，TAM セット内でレプリケーショングループの増加が不可能な場合に用いる．TAM セット内にスレーブインスタンス用のノードを追加し，スレーブインスタンスを移動することによって負荷を減らす場合では，現在構成されている TAM セット内にノードを追加し，負荷分散の計算を行なうことによって，マスタインスタンスとスレーブインスタンスがどちらも動作するマシンの負荷を軽くすることが可能である．

トランザクションサーバのレプリケーショングループが増加すると，KVS アクセスの受け取りや，ストレージサーバへの反映性能が上昇する．スケールアウトの概要を図 2-6 に示す．トランザクションサーバは，Partiqle サーバから KVS アクセスを受け取り，それをマイクロシャード単位でまとめ，ストレージサーバへと非同期で一括更新する．これらの処理は，トランザクションサーバ内のレプリケーショングループ単位で行われる．レプリケーショングループが少なければ，Partiqle サーバから渡される KVS アクセスが集中することとなり，かつ 1 台のノードがストレージサーバに反映出来るデータ量も多くなる．レプリケーショングループを増やすことで，それらの処理を分散させることができるので，トランザクションの処理性能が上昇する．

ストレージサーバは，スケールアウトによって実データ量の増加に対応可能である．以下，ストレージサーバのスケールアウトの概要図を図 2-7 に示す．

ストレージサーバが増加することで，保存可能なデータ量が増加することや，反映するデータを分散させることができるので，トランザクションサーバからの反映が効率的に行えることが考えられる．ストレージサーバでは，トランザクションサーバが持つログデータを非同期で反映し，同じデータについて 3 台のストレージサーバで多重保存する．ストレージサーバの台数が少ない場合，トランザクションサーバから反映されるデータの容量が大きくなり，各ストレージサーバにかかる負荷が大きくなる．ストレージサーバを増やすことで，データを保存するための容量を増やすだけでなく，トランザクションサーバから渡されるデータが分散されるため，各ストレージサーバの負荷が小さくなる．

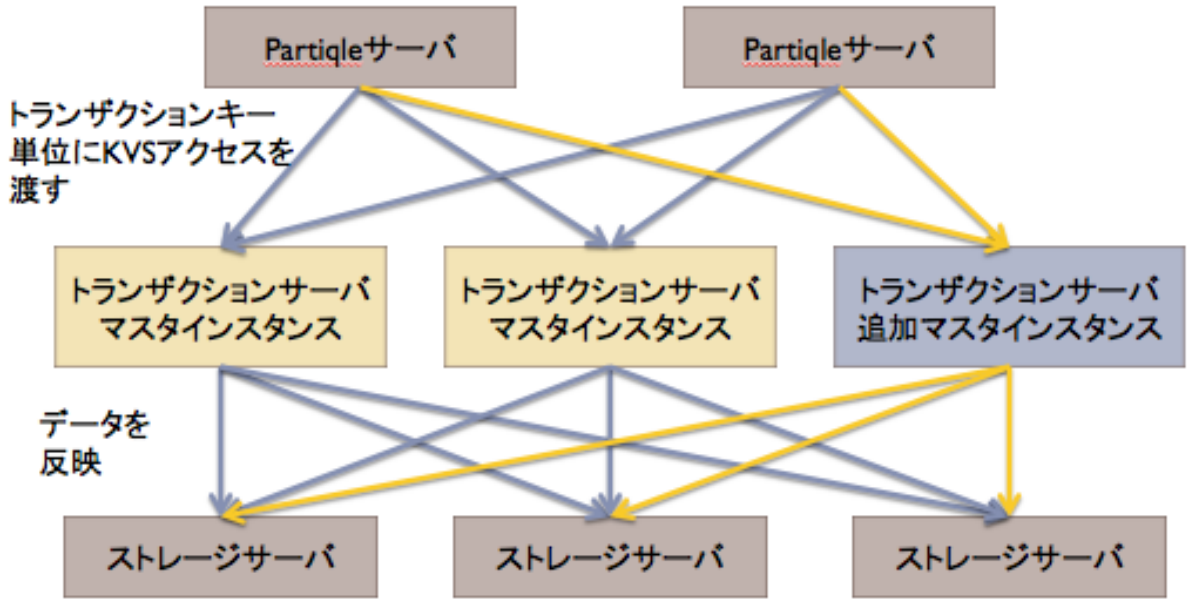


図 2-6　トランザクションサーバのスケールアウト

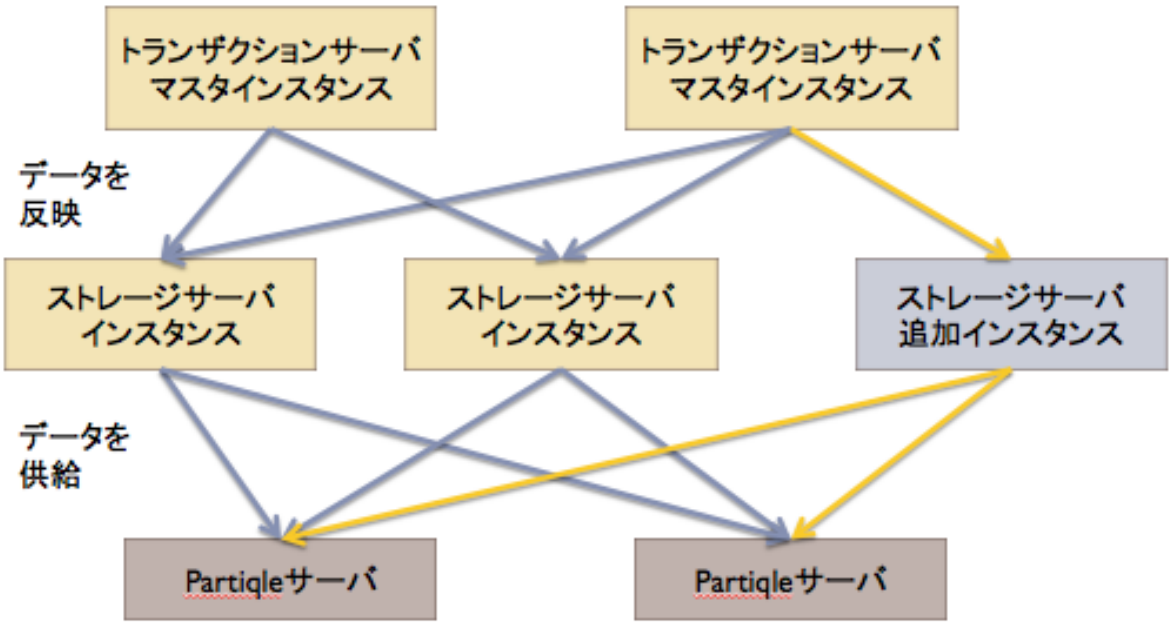


図 2-7　ストレージサーバのスケールアウト

2.3 IRS の性能

本節では，各サーバが IRS の全体的な性能に影響を及ぼす要素を挙げる．

Partiqle サーバでは，Partiqle サーバの台数，1 台のノードが SQL クエリを KVS アクセスに変換する性能，ネットワークトポロジ，帯域，CPU やメモリ等のノード性能が，IRS 全体の性能に関わる要素となっている．Partiqle サーバの台数は，ユーザアプリケーションからの SQL クエリの受け取りと，それを KVS アクセスに変換する性能，そしてトランザクションサーバに渡す処理性能に関わる．1 台のノードが SQL クエリを KVS アクセスに変換する性能は，Partiqle サーバがユーザアプリケーションから SQL を受け取ってから，KVS アクセスをトランザクションサーバに渡す時間に関わる．ネットワークトポロジや帯域は，ユーザアプリケーションから SQL アクセスを受け取る時間や，その量に関わる．CPU やメモリ等のノード性能は，ノード 1 台が 1 度に変換処理できる SQL クエリの量や，その処理時間に関わる．

トランザクションサーバでは，トランザクションサーバの台数，1 台のサーバに定義するインスタンス数，ネットワークトポロジ，帯域，CPU やメモリ等のノード性能が，IRS 全体の性能に関わる．トランザクションサーバの台数と 1 台のサーバに定義するインスタンス数は，レプリケーショングループ数に関わり，即ち Partiqle サーバからの KVS 受け取りやストレージサーバへの反映の性能に関わる．ネットワークトポロジや帯域は，Partiqle サーバやストレージサーバとの通信，トランザクションサーバのレプリケーションにおける速度や容量，ストレージサーバへの反映速度に関わる．CPU やメモリ等のノード性能は，1 台のノードで定義できるインスタンス数や，トランザクション処理速度，ストレージサーバへの反映速度に関わる．

ストレージサーバは，保存可能なデータ量，トランザクション処理性能に関わると考えられる．ストレージサーバの台数増加によって，保存可能データ量が大きくなる．この容量は，用いるノードのディスク容量に従って増加量が決定される．また，トランザクション実行時に，アプリケーションからトランザクションサーバが保持していないデータの参照をする際，Partiqle サーバへとデータの転送が行われるので，ストレージサーバの台数，ネットワークトポロジや帯域が，トランザクション処理性能に関わる．

第3章　開発するシステム

本章では，本プロジェクトで開発を行なうシステム，IRS 性能予測システムについて説明する．

我々は NEC 様に対する提案の結果，IRS の性能予測システムを開発することとした．IRS の持つスケールアウト性をさらに活かすために，性能の見積りを行なうシステムがあると良いという NEC 様からのご要望を受け取り，提案を行ったものである．このシステムは，例えば，新規に IRS を導入しようとしている方々，または IRS にサーバを追加しようとしている方々に対して，各サーバ台数による性能予測値を提供する．その他に，IRS の開発者の方々が，IRS のバージョンが上がった時に，旧来のバージョンと性能値を比較したい時に用いることが可能であると考える．

しかし，性能を予測するためには，IRS のモデルを作成する必要がある．そこで，性能予測システムの他に，性能予測モデルを作成するためのシステムを開発することとした．性能予測モデルを作成するためには，処理性能の実測値を測定し，回帰分析等の手法を用いて式を導出する必要がある．よって，我々は本プロジェクトで，サーバ台数を自動的に自由に変化させながら，IRS の性能計測と，モデル化を行った後，性能予測を行なうシステムの開発を目指した．

本章では，開発するシステムの概要について述べた後，実際に開発したシステムについて述べる．

3.1　システムの概要

本システムは，性能予測システムと予測式計算システムからなる．本システムの概要を図 3-1 に示す．予測式計算システムでは，IRS のベンチマークを実行し，その結果からモデルのパラメータ推定を行なう．性能予測システムでは，ユーザから入力されたサーバ構成に対

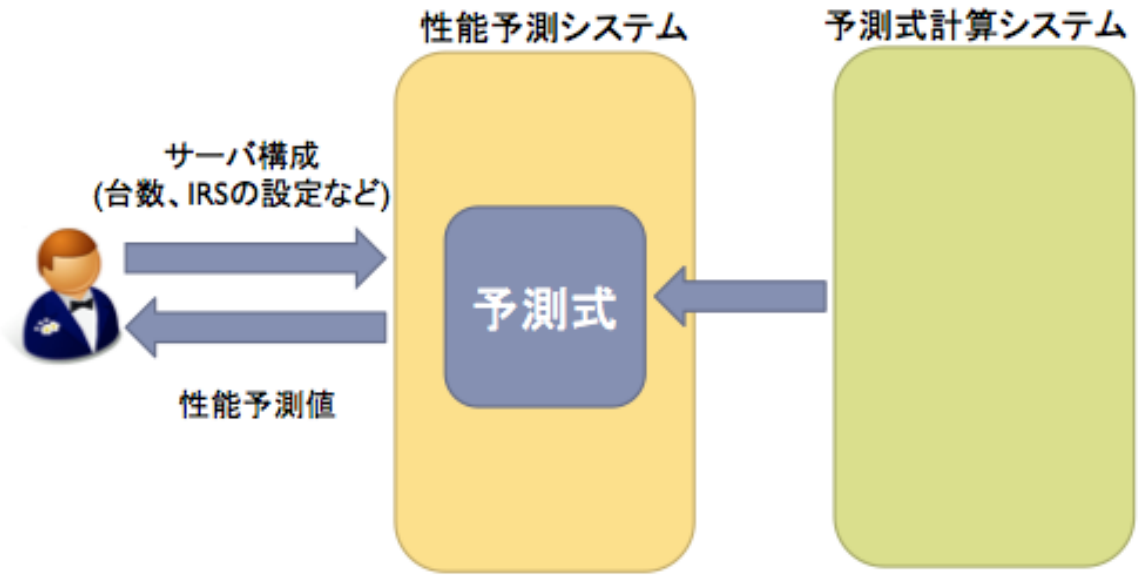


図 3-1　開発するシステムの概要

して，予測式計算システムで得られた予測式で計算を行なって，ユーザに対し性能予測値を出力する．

予測式計算システムは，ベンチマーク実行部とパラメータ推定部からなる．予測式計算システムの概要を図 3-2 に示す．

ベンチマーク実行部では，IRS のサーバ台数を自動的に変えながら，IRS に対してベンチマークを行ない，性能を実測する．ベンチマーク実行部の概要を図 3-3 に示す．ベンチマーク実行部では入力として，ベンチマークを行なう際の構成の範囲と，性能評価を行なう際の指標を受け取る．例えば，構成の範囲とは，あるサーバの台数を 2 台ずつ増やしながら計測を行なう，等である．性能評価を行なう際の指標は，TPC-C 等を挙げる．受け取った構成から，IRS 自動構成システムによって，IRS のサーバ構成を変化させながら，計測モジュールで性能を実測する．

パラメータ推定部では，ベンチマーク実行部から得られた性能実測値を集計し，モデルのパラメータ推定を行なう．パラメータ推定部の概要図を図 3-4 に示す．

パラメータ推定部では入力として，ベンチマーク実行部で計測した実測値と，モデルを作成するための雛形となる式を受け取る．集計モジュールで，実測値をまとめ上げた後，モデル化モジュールを用いて，雛形に従ったパラメータ推定を行なう．それによって作成した性能予測モデルを，性能予測システムに渡す．

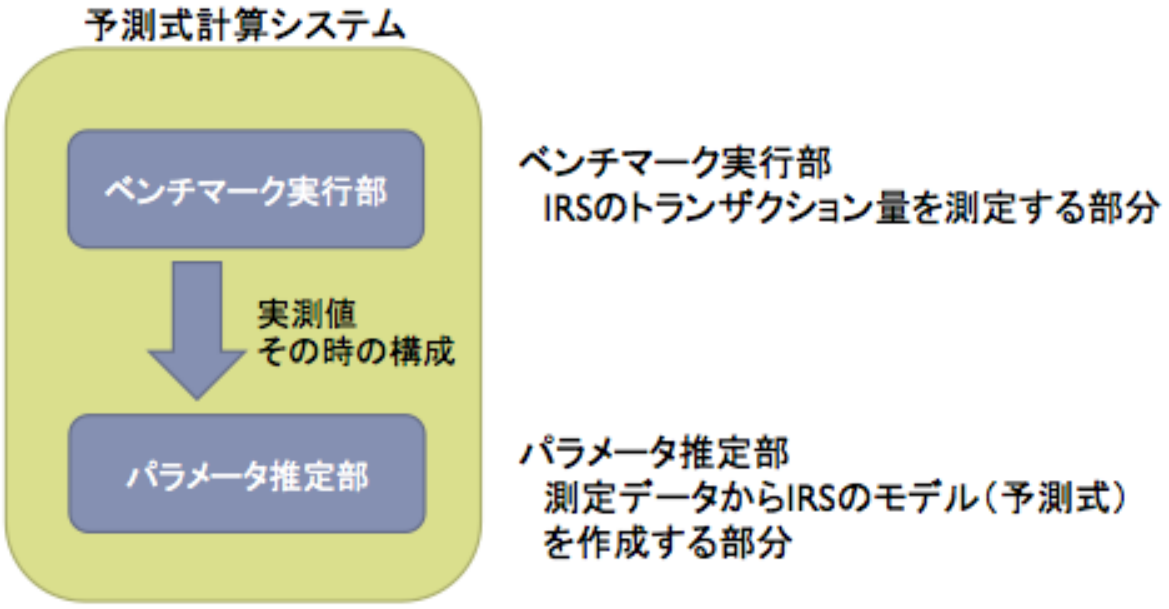


図 3-2　予測式計算システム

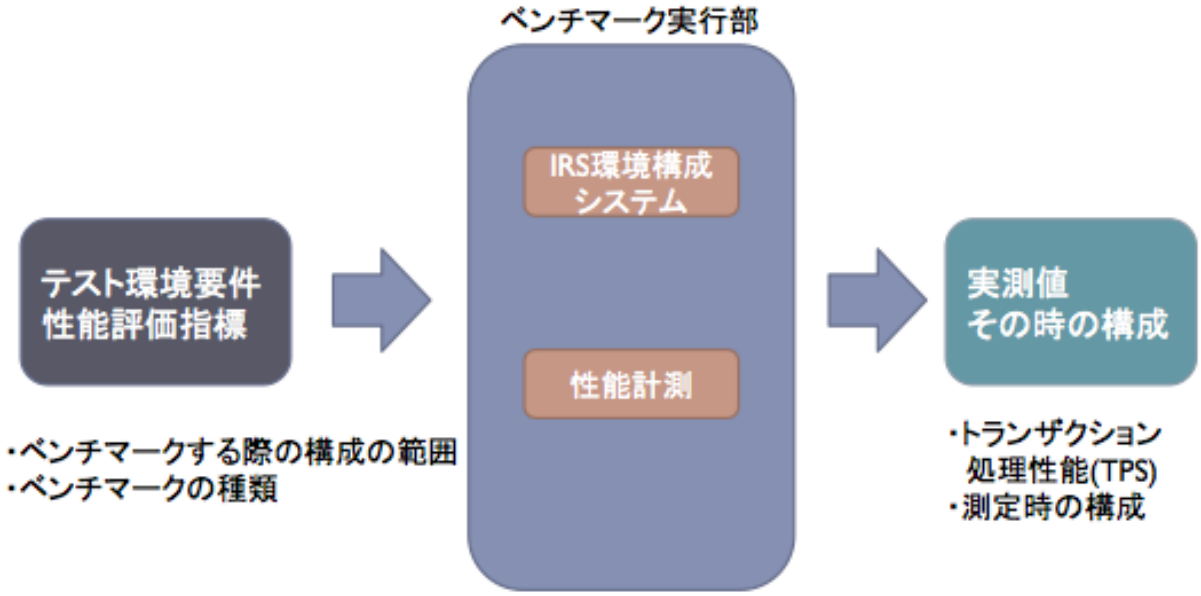


図 3-3　ベンチマーク実行部

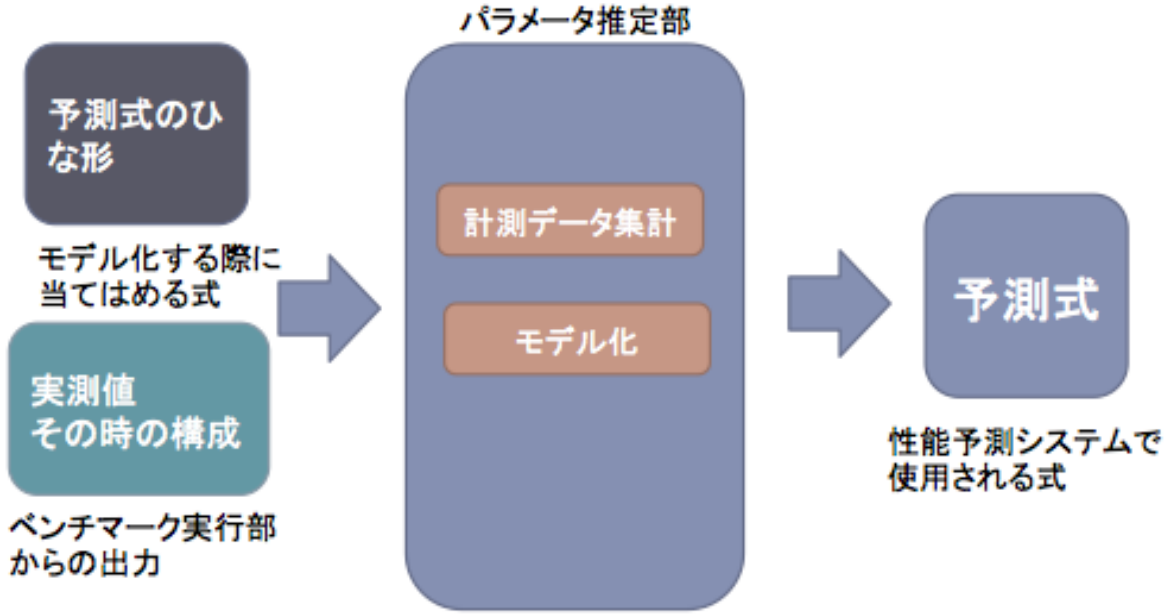


図 3-4　パラメータ推定部

以下，システムに用いる技術を述べる．

まず，ベンチマークの実行には，JdbcRunner[8]を用いる．JdbcRunner は，あらかじめ性能測定方法をシナリオという形で定義し，それに従ってデータベースの性能測定を行なうアプリケーションである．IRS には JDBC ドライバが搭載されており，それを用いてアプリケーションとの通信を行なう．そこに JdbcRunner を接続し，性能測定を行なうこととした．

また，パラメータ推定には，R[9]と，それを Java で扱うための rJava[10]というライブラ

リを用いる．R は，S 言語と呼ばれる統計プログラミング言語をオープンソースとして実装したアプリケーションである．R では，コンソールを用いて対話的に統計プログラミングを行なうことが可能であるが，今回は rJava を用いて，Java から R の機能を用いることとした．

開発言語は，コマンドの実行等にシェルスクリプト，性能計測モジュールにおける JDBCRunner のシナリオ作成に JavaScript，モデル化モジュールの実装を Java，R での計算に S 言語をそれぞれ用いる．

開発は，性能予測システムと，予測式計算システム内にある各モジュール単位に行なう．

3.2 開発したシステム

本節では，開発した性能予測システムについて述べる．

本プロジェクトで開発した性能予測システムについて，各モジュールが単体で動作するところまで完成しており，IRS の各種サーバ台数による性能予測値を出力することが可能である．これにより，3 章冒頭で述べた NEC 様への価値の提供が可能となった．

IRS 環境構成システムでは, IRS を構築するサーバへの自動 OS インストールと設定, IRS の自動インストールと設定の 2 点を行なう．OS インストールについては，性能計測を行なう上で必要な設定までを行った OS 環境の提供が可能となっている．IRS の自動インストールや設定については，まだ手動で行なう必要がある．自動的に IRS の環境を変更する部分については，ch-root を用いた再配置による方法を考えているが，IRS の設定については手動で行なう. それらの部分について開発を継続して行ない, 本年 2 月中に完成予定となっている．詳細は，本プロジェクトメンバである村上の特定課題研究報告書[11]を参照されたし．

性能計測モジュールでは，JdbcRunner を用いた IRS の性能計測を行なうことが可能である．本プロジェクトでは，IRS の性能を測定するためのデータモデル，トランザクションを策定した．それらを用いて，IRS のトランザクション処理性能 Transaction Per Second(以下 TPS)を測定する．現状では，ストレージサーバの影響を加味しない，Pertiqle サーバの台数とトランザクションサーバのレプリケーショングループ数による性能測定が可能となっている．今後は，JdbcRunner での性能計測の結果を集計モジュールに渡すためのインターフェースの開発と, ストレージサーバの性能も含めた TPS 測定のための測定方法の策定を行なう．詳細は本プロジェクトメンバである程の特定課題研究報告[12]を参照されたし．

集計モジュールについては，現在は開発されておらず，計測モジュールの結果を手動で CSV 形式ファイルにまとめている．今後，IRS 環境構成システムから受け取った現在の台数構成と，性能計測モジュールから受け取った実測値を CSV 形式ファイルにまとめ上げ，モデル化モジュールに渡す部分について開発を行なう．

モデル化モジュールでは，予測モデルの導出が可能となっている．性能計測で得られた実測値がまとめられた CSV ファイルと，本プロジェクトで策定したモデルの雛形となる式を入力とし，R による非線形回帰分析を行なうことで，パラメータ推定を行なう．そこで得られたパラメータを，予測値計算やグラフ化を自動で行なうことが可能となるように参照等の設定を施した Open Document Spreadsheet(以下 ODS)[13]形式のファイルに書き込む．モデル化モジュールについては，筆者担当部分である．個人の貢献として 5 章 1 節にて詳細に説明している．

性能予測システムについては，モデル化モジュールの出力として得られる ODS 形式ファイルがその役割を担っている. モデル化モジュールから出力された ODS 形式ファイルでは，性能予測値とグラフの閲覧が可能となっている．これによって，各種サーバの台数によるボトルネック発生を判断することや, 各バージョンについて測定, モデル化が行われていれば，IRS のバージョンによる性能値の変化を確認することが可能である．現状では ODS 形式ファイルにて予測値を閲覧する以外の方法はないが，予測を見たい台数構成を入力として，その構成での性能予測値を出力する GUI アプリケーションや Web アプリケーションを開発予定である.

第4章　　プロジェクトの進捗

本章では，本プロジェクトのスケジュールと実際の作業期間を最初に示した後，プロジェクトの進捗について説明する．

4.1　　スケジュールとその実績

本プロジェクトの長期スケジュールとその実績を図 4-1 に示す．

我々は，イテレーティブ開発手法を用いて本プロジェクトで開発を行う計画であった．これを採用した理由としては，学生チームの人数が 3 人と少人数であるため，3 章に示した各モジュールやシステムを，段階的に少しずつ 3 人で分担して実装に取り組んだ方が，見通しの良い開発が出来ると考えたからである．プロジェクトの計画としては，提案の後，実装を 4 つフェーズに分けて，機能を段階的に実装していくものとなっている．開発フェーズのスケジュールを図 4-2 に示し，以下，プロジェクト当初に計画した開発フェーズにおける各実装フェーズについて説明する．

1. 仮想マシンを自由に増減できる機構の部分の開発を行う．
2. 作成した仮想マシンに実際に IRS をインストールして，自動構成で動作させる．
3. 説明変数を台数として，IRS の処理可能トランザクション量の台数特性，データ採取を行う．
4. 採取したデータの表示方法について考え，実装する．

オプション：説明変数を増やして測定する．説明変数の例としては，CPU のコア数や，メモリ容量等を挙げる．

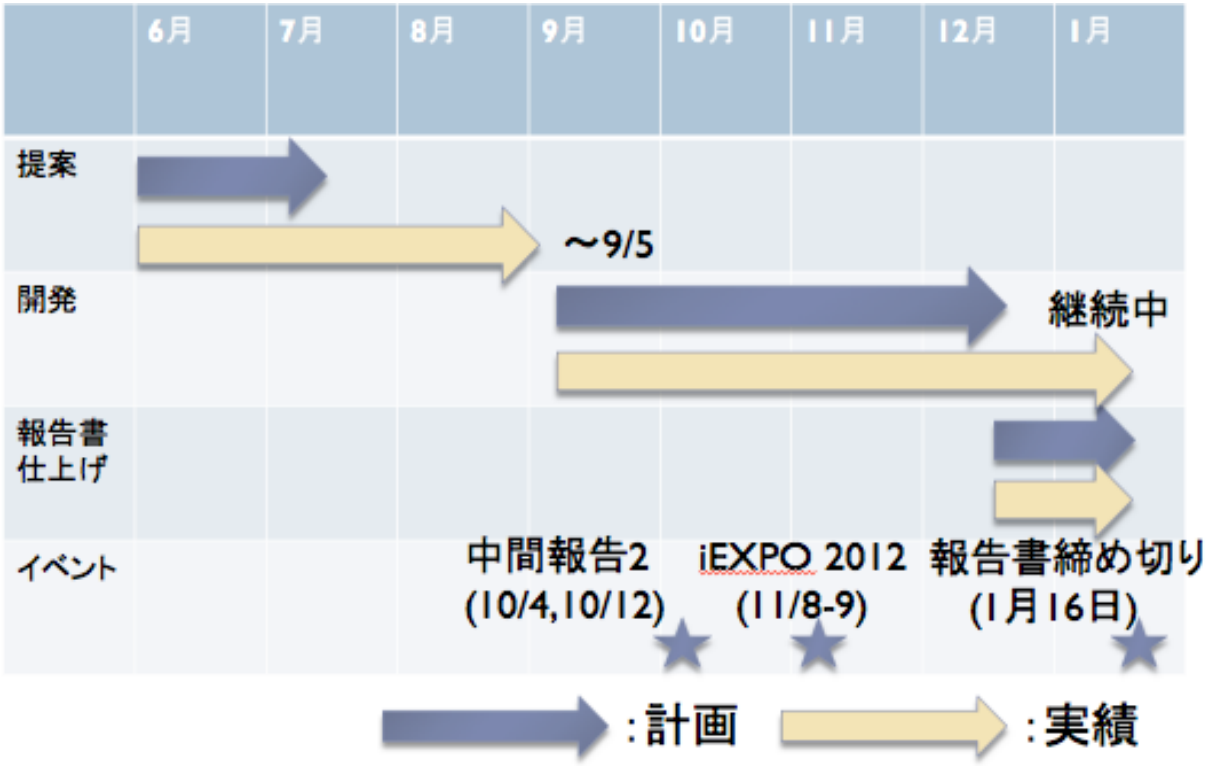


図 4-1　長期スケジュール

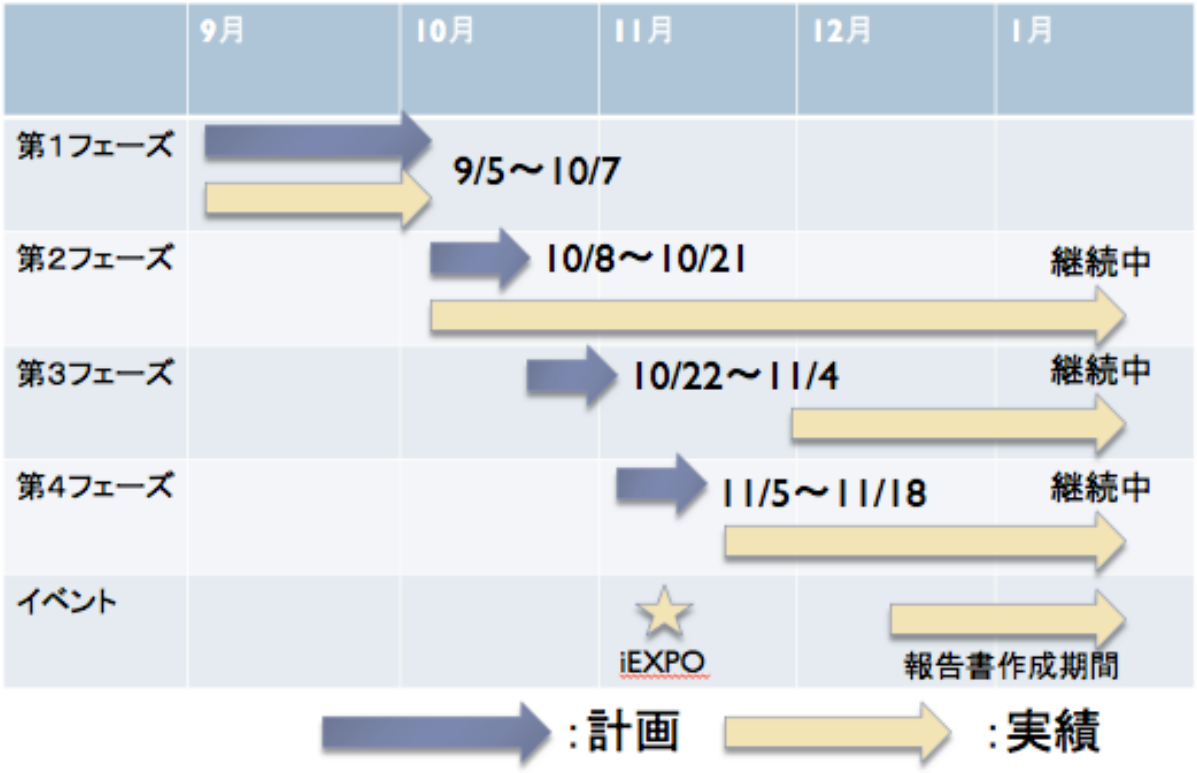


図 4-2　開発フェーズのスケジュール

図 4-2 を見ると，実績として，第 2 フェーズが大幅に遅延している．これは，当初予定していた仮想マシン上での IRS 構築から，物理マシン上での構築へと方針を変更したことと，物理マシン上での IRS インストール，構築に多大な時間を費やしたからである．

この原因としては，我々が用意したマシンが，IRS を構築するには搭載メモリ容量が少なかったことが主に挙げられる．IRS の構築に用いるマシンに推奨される搭載メモリ容量は 16GB であり，仮想マシンに割り当てられるメモリ容量が不足したことや，我々が用意可能であった物理マシンの搭載メモリ容量が最大 4GB であったことから，インストールや，起動において問題が発生した．

第 2 フェーズは当初 3 人で分担して作業を行なっていたが，第 4 フェーズのモデル化モジュール開発の作業を IRS インストール完了前に前もって開始し，IRS インストール後に第 3 フェーズの性能計測の部分が開始した．

現在の状況としては，3 章 2 節で述べたように，各モジュールが単体で動作する部分まで開発が行われた．今後，更に開発を進め，2 月中に NEC 様の環境にて実験を行った後，成果物一式について納品を行なう予定である．

4.2 提案

この節では，学生チームが顧客である NEC 様に行った提案について説明する．

我々は，システムの開発に先立ち，開発するシステムについて NEC 様に対して幾つか提案をさせて頂いた．

4.2.1 最初の提案

初めに，IRS を用いたアプリケーションについてチームで意見を出し合った後，実際に NEC 様に対し提案を行った．提案の内容は以下の通りである．

1. IRS に関する性能予測ツール
   サーバの台数や CPU の性能，メモリやディスク容量によってトランザクション処理性能がどのように変化するか，という部分について，モデル化を行うツールに関して提案を行った．
2. 自動学習型位置推定システム
   スマートホンに搭載された GPS や高度計といった各種センサからデータを収集して，総合的に判断し位置推定を行うシステムの開発について提案した．
3. クラウドストレージサービス
   ローカルに実ファイルを保存する形式ではなく，リモートでファイルを管理するクラウドストレージサービスを提案した．
4. Twitter をデータソースとするシステム
   Twitter の API によって，大規模データを収集し，日付，地理など詳細な条件を指定した tweet の抽出や，tweet よりユーザの属性・好みを推定するシステムを開発する．

最初に行った提案では，開発するシステムの決定まで至らなかった．理由としては 2 点挙げられる．

1 つ目の理由として，IRS の特性についてよく理解できていなかった事が挙げられる．IRS は，トランザクション処理性能に優れているが，今回提案したシステムはそれを活かしたシステムではなかった．スケールアウト性に優れており，大容量のデータが扱える，といった要素から提案を行ったため，トランザクション処理性能を活かしたものを提案できなかった．

2 つ目の理由としては，多く提案することを目標としたため，個々の提案についての考察が不十分となったことが挙げられる．

提案の結果から，NEC 様より，上記 1 番の IRS 性能予測ツールに関しての要望が挙げられた．しかし，IRS の特性について理解した上で再度提案を行った方が良いと考えたため，開発するシステムの決定を見送った．

4.2.2 再提案

3 章 1 節 1 項で説明した最初の提案を基に，NEC 様から頂いた要望，意見を参考にしながら，IRS についての性能予測ツールに関する提案を再度行った．これは，IRS を構成するサーバのメモリや CPU 性能，各サーバの台数から，IRS の性能が分かればさらに IRS を活かすことができる，といった NEC 様の要望から，最初の提案を考察し，再提案を行ったものである．

再提案の結果，IRS についての性能予測を行なうためのシステムの開発を決定した．システムの詳細については第 3 章を参照されたし．

4.3 IRS 環境の構築

この節では，我々の手元に IRS 環境を用意した際の流れについて説明する．

IRS の性能予測システムの開発を行なうに先立ち，我々の居室環境にて IRS の構築を行った．当初，物理マシンの用意が不可能だと考えたため，仮想マシンによる構築を考えたが，上手くいかなかった．その後，高度 IT コースの学生から，使用していないデスクトップ PC を借用することで，物理マシンによる IRS 環境を構築した．

スケジュールが遅れてしまった原因は，主にこのフェーズにあったと考えられる．我々は IRS 環境の構築を 1 週間で終わると見込んでいたが，実際には 2 ヶ月もの時間を消費してしまった．この原因としては，NEC 様とのコミュニケーション不足，特に，原因が分からない場合に，我々が試行錯誤し過ぎてしまい，NEC 様への連絡が遅れてしまったのが大きな要因であると考える．

4.3.1 仮想マシン上での構築

当初, 我々のチームでは, 物理マシンでの IRS 環境の用意が不可能であると判断したため, 仮想マシンを用いて IRS 環境を用意することとした. IRS はスケールアウト性に優れるシステムのため，ノードとなるマシンが多数必要となる．開発当初，必要となるマシンを用意する手立てを持っていなかったため，仮想マシンを多数起動することでこれが満たせると考えた．

仮想マシンを用意する環境としては，KVM(Kernel-Based Virtual Machine，以下 KVM)[14]を用いることにした．これは，Linux 系 OS で用いられる yum コマンドによって容易にインストールできること，筆者が KVM を扱った経験があることから採用した．

しかし，仮想マシン上で IRS を動作させるにあたり，様々な問題が発生した．IRS，特にトランザクションサーバでは，トランザクションデータをメモリ上で保持する．よって，大きなメモリ容量が必要となる．NEC 様の資料では，トランザクションサーバを起動させるために，16GB のメモリを搭載するマシンを推奨環境としていた．我々が用意できるマシンは，一番大きくてもメモリ 4GB を搭載するマシンであり，仮想マシンに割り振ることの出来るメモリ容量も小さくせざるを得なかった．そのため，メモリ不足によってトランザクションサーバの起動ができないという事態が起こった．

結果として，仮想マシン上ではなく，物理マシン上での IRS インストールをするように方針が変更となった．

4.3.2 物理マシン上での構築

仮想マシン上での環境用意が不可能であると判断した後，我々の居室内での物理マシンの IRS 環境の用意を試みた．物理マシンでの環境は，高度 IT コース学生に貸与されているデスクトップ PC を集め，用意した．但し，いずれのマシンも搭載メモリ容量が 2GB～4GB と，IRS の推奨環境とされるメモリ 16GB には足りないマシンである．

我々が用意した物理マシン環境では，IRS の動作は確認できるものの，トランザクションサーバをスケールアウトする際の問題が発生した．用意したマシンの搭載メモリ容量が不足していることにより，トランザクションサーバのインスタンスが，サーバ 1 台に対し 1 個ずつとなっている．さらに，インスタンスが使用するメモリ容量を，IRS の初期設定で定められた標準的な容量よりも少々小さくせざるを得なかった．

また，物理マシンの台数を十分に確保できなかった．高度 IT コース学生に貸与されているデスクトップ PC は，半数は搭載メモリ容量 2GB のマシンである．今回我々は，搭載メモリ容量が 4GB のマシンで，学生が使用していないものについて 8 台確保することができたが，それでも IRS のトランザクション処理性能を測定するには不十分である．IRS 環境の構築方法としては，我々の居室内に構築する他，クラウドサーバを活用する方法等が挙げられるが，ネットワーク接続の面で不安が生じるため，採用しなかった．

4.4 各システムの実装

この節では，IRS 構築後の性能予測システムの実装について説明する．

まず，学生チームの実装担当の割り振りについて説明する．実装は，モジュール毎に開発担当者を決めて行った．担当者は以下の表の様になっている．当初の予定では，モジュール毎に開発担当者を決める予定ではなく，それぞれのモジュールについて細かく担当者を決めて開発を行なう予定だった．しかし，4 章 3 節に示した遅延によって，それぞれ並行して開発を行なうことを余儀なくされた．

実装は，モジュール毎に担当者を決めて行った．各モジュールに対する担当者を表 4.1 に示す．村上は，サーバの構築やクラスタの構成の知識を多く有していたことから，IRS 自動構成システムの実装を担当した.程は,既に性能計測に関する調査を行なっていたことから，性能計測モジュールの実装を担当した．筆者は，前もって R やモデル策定について調査を始めていたことから，モデル化モジュールの実装を担当した．

表 4.1 各モジュール開発の担当者

担当者	担当モジュール
村上	IRS 自動構成システム
程	性能計測モジュール
菊池(筆者)	モデル化モジュール

4.5　NEC 様の環境下での性能実測

この節では，NEC 様が保有するサーバを用いて行った性能実測について述べる．

我々は，NEC 様からお借りした環境を用いて，トランザクション処理量の実測を目指したが，借用期間ではそれを達成することができなかった．

NEC 様にお借りした試験環境について，表 4.2 に示す．我々はこの環境で，IRS の本番環境におけるトランザクション処理量の実測を目指した．IRS はスケールアウト性に優れるシステムであり，処理性能の実測にはサーバの台数と，多くの時間が必要となる．よって，サーバ台数は，NEC 様から借用できる上限の 12 台，期間は土曜日と日曜日を挟んだ 5 日間と設定した．

しかし，NEC 様に環境をお借りした 5 日間では，当初の目的を達成できなかった．これの理由としては，性能計測を行なう指標について，当初予定していた TPC-C を準拠した性能計測が，IRS には合っていないという結論に至ったことが挙げられる．IRS は，例えばオークションでの入札処理や落札処理といった単純なトランザクションの処理に長けており，TPC-C の様な複雑さを持ったトランザクションの処理には適さない．よって，性能指標の変更を余儀なくされ，結果として，計測が思うように出来ず，当初の目標であったトランザクション処理性能の実測を行なうことができなかった．

表 4.2　NEC 様より借用した環境

サーバ	Express5800/B120d（ブレード）
台数	12 台
メモリ	32GB(8GB×4)
HDD	600GB(300GB，2.5 型 SAS 15,000rpm)×2
ネットワーク規格	1000BASE-T　4 ポート
借用期間	12 月 13 日～12 月 19 日(15 日(土)，16 日(日)を除く)

第5章　筆者のプロジェクトに対する貢献

この章では，筆者がプロジェクトに対し貢献した部分について述べる．まず，筆者の主要な貢献であるマネジメントと性能予測モデルの策定について述べ，次にその他の貢献を述べる．

5.1　性能予測モデルの策定

筆者は，本プロジェクトにおいて，IRS の性能を予測するためのモデルの策定を行うことで，予測式計算システムにおけるモデル化モジュールと，性能予測システムにおける予測値導出の部分の開発に寄与し，本プロジェクトで開発するシステムにて性能予測を行なうことが可能となった．

性能予測モデルの策定は，4 章 1 項で述べたように，IRS の性能実測よりも先立って行われた．理由としては，IRS のインストール時に発生した遅延のため，時間がかかると考えられたモデル策定を先に行ったこと，モデル策定のための関数に調査を要したことと，R を用いたパラメータ推定について調査する必要があったことが挙げられる．

よって筆者は，IRS の特徴や構成から，トランザクション性能が上昇する要素の洗い出しを経て，性能上昇の傾向を考察し，それを基に性能予測モデルの策定を行った．今回策定した性能予測モデルは，各種サーバの台数またはインスタンス数によって，線形的に性能が上昇すると仮定している．また，各種サーバのいずれかの数が少ない時，そのサーバによって，性能上昇が滞る，即ちボトルネックが起こるような部分について表している．モデルは，IRS の各種サーバが全て IRS のトランザクション処理性能に関わると仮定した 3 変数モデルと，各種サーバのうち 2 種のサーバがトランザクション処理性能に関わると仮定した 2 変数モデルの 2 つを策定した．

今回策定したモデルの評価は，我々が構築した IRS の環境における台数不足，性能測定方法の不備により，十分に行えていない．トランザクションサーバが動作するには最低でも搭載メモリ容量が 4GB のノードを用いる必要があるが，我々にはそのようなノードを十分に用意することが不可能であった．また，現状の測定方法では，IRS の性能測定を十分に行なうことが出来ない．具体的には，ストレージサーバを無視し，Partiqle サーバとトランザクションサーバによる性能測定を行なうような測定方法となっている．

この節ではまず，IRS の構成から定義したモデル式について述べる．次に，そのモデルの導出を行った際の考察について述べる．その後，R を用いたパラメータ推定，モデル式の考察と再策定，実測データを用いた性能予測モデルの作成について述べる．最後に，性能予測値を出力する部分の開発について述べる．

5.1.1 定義した性能予測モデル

この項では，今回定義した性能予測モデルについて述べる．性能予測モデルの策定における考察については，5 章 1 節 2 項で述べる．

筆者は，IRS の各種サーバの台数を説明変数とする 3 変数モデルの策定を行った．この策定を行った理由としては，IRS のトランザクション処理性能が，IRS の各種サーバの台数によって決定されると考察したためである．

今回策定したモデルは，以下の様なものとなっている．

$$(e - a_p p)(e - a_t t)(e - a_s s) = C \quad \cdots (1)$$

(ただし，C は負の定数)

ただし，eはトランザクション処理量予測，p は Partiqle サーバの台数，tはトランザクションサーバのレプリケーショングループ数，sはストレージサーバの台数，Cは負の定数とする．このモデルは，各種サーバのいずれかのみが増加したとき，他のサーバ数の少なさがボトルネックとなり，それ以上性能が上昇しないことを考慮したものとなっている．また，各種サーバが増加した時の性能上昇が線形的であると仮定して策定を行った．

また，3 種のサーバのうち 2 種のみがトランザクション処理性能に関わっている場合に備えて，3 変数モデルを策定する前に考察を行った以下の 2 変数モデル

$$(e - a_x x)(e - a_y y) = C \quad \cdots (2)$$

(ただし，C は正の定数)

も，性能予測モデルの作成に用いることとする．

5.1.2 性能予測モデル策定における考察

この項では，5 章 1 節 1 項で述べた性能予測モデルを策定する際の考察について述べる．

モデルの策定ではまず，3 台のサーバ各々の性能を考え，IRS の性能に大きく関わる要素について洗い出しを行った．各種サーバに共通して性能に関わる要素と言えるのが，各ノードの台数もしくはインスタンス数と，各ノードが接続するネットワークのトポロジ，帯域である．このうち，ネットワークによる性能の変化については，IRS が高信頼な LAN 内に構成されるということや，Partiqle サーバに最大接続数が設定されている事等，ネットワークボトルネックが起こりにくい構成となっていることから，今回策定するモデルには含めないこととした．よって，今回のモデルは，IRS の各種サーバの台数またはインスタンス数を説明変数として策定を行った．

次に，IRS の各サーバ台数によって性能がどのように変化するか，ということについて考察を行った．以下，各種サーバについて考察を述べる．

Partiqle サーバは，台数が増えることで最大接続可能数が大きくなるため，トランザクション処理性能として線形に増加すると考えられる．

トランザクションサーバについては，レプリケーショングループの増加によってトランザクション処理の効率化が可能となるが，1 台のトランザクションサーバに対し複数インスタンスを定義可能であることから，線形的もしくはやや線形的に性能が上昇すると考えられる．

ストレージサーバは，トランザクションサーバからのデータ反映処理を分散させることが可能であるが，ノード追加によって Partiqle サーバからのデータ参照についてスループットがある程度減少すると考えられるため，線形的もしくはやや線形的に性能が上昇すると考察した．

ただし，例えばトランザクションサーバのレプリケーショングループが少ない状態で，

Partiqle サーバ数が増加した場合，レプリケーショングループが少ないことによる，トランザクション性能上昇についてのボトルネックが起こり得る．その他のサーバについても，あるサーバが少ないことによるボトルネックの発生が予想できる．

以上から，予測モデルの要件は以下の様である．

- Partiqle サーバの台数，トランザクションサーバのレプリケーショングループ数，ストレージサーバの台数の全てもしくは一部を説明変数とし，トランザクション処理性能を被説明変数とするモデル
- Partiqle サーバ，トランザクションサーバ，ストレージサーバのいずれかを増やしただけでは，増やしていないサーバがボトルネックとなり，全体的な性能が上がらないようなモデル

この要件を満たした場合に，1 種類のサーバの台数に着目した時の性能変化を表すグラフの概形を図 5-1 に示す．この図では，緑色が着目するサーバ単体の性能変化，青色が多種のサーバによって決定する性能上限，赤色が性能値を表している．今回の策定では，各サーバ・インスタンスの増加による性能上昇がそれぞれ線形的になると仮定して策定を行った．

まず, 要件として挙げた説明変数のうち, 2 つを説明変数とする 2 変数モデルを策定した．全てを説明変数とする 3 変数モデルは，2 変数モデルの拡張として得ることが可能であると考えたためである．

2 変数の場合のモデル策定では，双曲線関数を用いることとする．これは，ボトルネックについて考慮する必要があり，ある値に漸近する様な関数が必要となるためである．

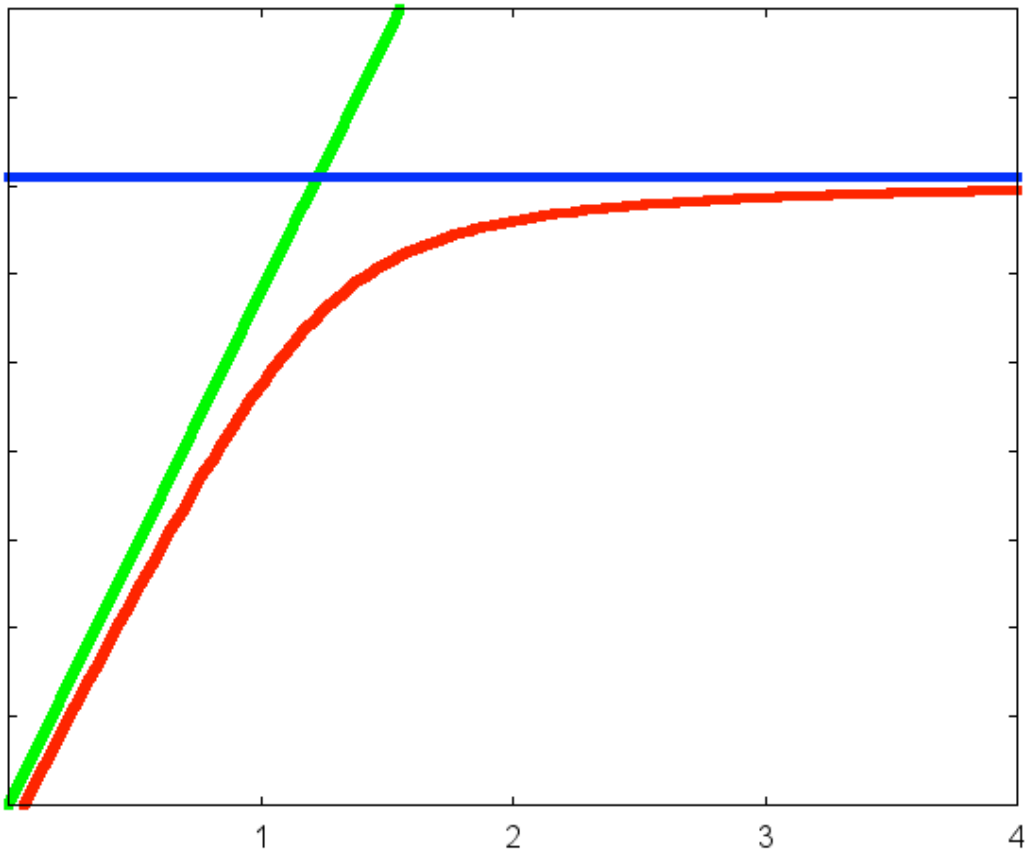


図 5-1　1 種類のサーバに着目した場合の台数による性能変化

双曲線関数は，以下のような式

$$(e - A)(e - B) = C$$

$$\begin{pmatrix} A, B\text{は漸近線のとる値で，定数もしくはそれを決定する関数} \\ C\text{は定数} \end{pmatrix}$$

で表現することが可能である．今回策定しようとしているモデルでは，説明変数の増加による性能上昇が線形的であると仮定しているため，漸近線は $A = a_x x$，$B = a_y y$と定義できる．よって，2 変数モデルが，（2）式

$$(e - a_x x)(e - a_y y) = C$$

と定義できる．この式の中のa_xとa_y，Cを決定できれば，モデルの作成が可能となるが，図 5-1 の様なグラフになるためには，C が正の定数でなければならない．

続いて， 3 変数の場合のモデルについて，2 変数モデルの拡張を行ない，策定を行った．3 変数でのモデルは，2 変数の場合に導出された式を拡張することで得る．トランザクション処理量予測を e，Partiqle サーバの台数を p，トランザクションサーバの台数を t，ストレージサーバの台数を s とする．2 変数の場合に導出された式を 3 変数に拡張した結果，3 変数モデル（1）式

$$(e - a_p p)(e - a_t t)(e - a_s s) = C$$

(C は負の定数)

が導出可能である．よって，IRS の性能実測値を用いて，a_p，a_t，a_s，Cを決定できれば，それが性能予測モデルとなる．

5.1.3　　R を用いたパラメータ推定

この項では，前項までに述べた性能予測モデルを用いたパラメータ推定方法とその考察について述べる．まず，策定した 3 変数の性能予測モデルを R で用いるための計算について述べる．その後，R を用いたパラメータ推定方法について述べる．

性能予測モデル作成のためのパラメータ推定は，統計のためのツールである R を用いた非線形回帰分析によって行なう．しかし，R を用いてモデルを作成するには，上記の式を性能値 e の方程式として解く必要がある．よって，上記式を e について変形することを考える．以下，2 変数モデルと 3 変数モデルについて，式変形について述べる．

まず，2 変数モデルのパラメータ推定を行なうために，式（2）を変形する．変形は，2 次方程式の解の公式を用いる．式（2）を展開し，2 次方程式とみなせば，

$$e^2 + a_1 e + a_0 = 0$$

但し，

$$a_1 = -(a_x x + a_y y)$$

$$a_0 = a_x a_y xy - C$$

と置くことが出来る．よって，解の公式を用いれば，

$$e = \frac{-a_1 \pm \sqrt{{a_1}^2 - 4a_0}}{2}$$

と変形可能である．評価の結果，性能予測に適当な式は，

$$e = \frac{-a_1 - \sqrt{{a_1}^2 - 4a_0}}{2} \quad \cdots (3)$$

と求まったので，これを用いて R でパラメータ推定を行う．

次に，3 変数モデルのパラメータ推定を行なうために，式（1）を変形する．変形は，3 次方程式の解の公式であるカルダノの公式[15]を用いて行なう．カルダノの公式は，以下の様なものである．

3 次方程式を$x^3 + ax^2 + bx + c = 0$とする時，その 3 つの解は，

$$x_1 = \omega_1 \sqrt[3]{-q + \sqrt{q^2 + p^3}} + \omega_1 \sqrt[3]{-q - \sqrt{q^2 + p^3}} - \frac{a}{3}$$

$$x_2 = \omega_2 \sqrt[3]{-q + \sqrt{q^2 + p^3}} + \omega_3 \sqrt[3]{-q - \sqrt{q^2 + p^3}} - \frac{a}{3}$$

$$x_3 = \omega_3 \sqrt[3]{-q + \sqrt{q^2 + p^3}} + \omega_2 \sqrt[3]{-q - \sqrt{q^2 + p^3}} - \frac{a}{3}$$

と表すことが出来る．但し，

ω_1，ω_2，ω_3は 1 の 3 乗根

$$\left(\omega_1 = 1,\ \omega_2 = \frac{-1 + \sqrt{3}i}{2},\ \omega_3 = \frac{-1 - \sqrt{3}i}{2}\right)$$

$$p = \frac{3b - a^3}{9},\quad q = \frac{27c + 2a^3 - 9ab}{54}$$

である．

今回の場合，解くべき方程式は，5.1.1 項で述べた式（1）を展開することで得られる．展開した結果，

$$e^3 + a_2e^2 + a_1e + a_0 = 0$$

但し，

$$a_2 = -(a_pp + a_tt + a_ss)$$

$$a_1 = a_pa_tpt + a_ta_sts + a_sa_psp$$

$$a_0 = a_pa_ta_spts - C$$

が得られた．これを解の公式に当てはめれば，3 つの式が導出される．そのいずれかを用いて R で非線形回帰分析を行うということを考えた．

R にて 3 次方程式の解の公式を扱うには，2 つの方法がある．カルダノの公式で得られた式をそのまま使う方法と，3 次方程式の判別式を用いた場合分けによる方法である．

カルダノの公式をそのまま扱う方法では，R での計算を複雑にしなくて済むが，複素数の 3 乗根を計算するにあたって留意する点が存在する．留意する点とは，3 乗根が一般に 3 つあることによる．実数の 3 乗根については，R による 3 乗根計算で実数値を得ることが出来るが，複素数の 3 乗根について，対象の複素数によって出力される値が定まらない．複素数 z^3の 3 乗根の 1 つを z とすれば，残り 2 つの共役複素数は

$$z\omega_2,\ z\omega_3$$

$$\left(\omega_2 = \frac{-1 + \sqrt{3}i}{2},\ \omega_3 = \frac{-1 - \sqrt{3}i}{2}\right)$$

で与えられる．R では，3 乗根の値のうち 1 つを出力するが，複素数の 3 乗根を求めたい時に，その結果で z が出力される時，$z\omega_2$，$z\omega_3$が出力される時と，場合によって異なるため，解の公式から導出される 3 式のどれを用いればよいか特定することが不可能である．よって，モデル式の特定方法を調査することが必要となる．

3 次方程式の判別式を用いた場合分けによる方法は，複素数の 3 乗根の計算を介さずに済むが，その分 R での式の定義が複雑になってしまう．3 次方程式の判別式は，カルダノの公式で用いた p，q を用いて，

$$D = -q^2 - p^3$$

で与えられる．3 次方程式の各項の係数が実数であれば，判別式 D によって以下のように解の判別が可能である．

$D \geq 0$：3 つの解がいずれも実数解．$D = 0$であれば重解を持つ．

$D < 0$：1 つの実数解と 2 つの共役な複素数解を持つ．

しかし，この方法を取る場合，場合分けによる分岐を R で定義する必要があるため，モデル式が複雑になってしまうという問題も挙げられる．

以上 2 点の方法から，3 次方程式の判別式を用いた場合分けによる方法を選択した．以下，それぞれの場合について式の考察を行なう．

● $D \geq 0$ の場合

$D \geq 0$ の場合，3 つの解がいずれも実数となるが，$D \neq 0$ の場合，計算の過程上で必ず 1 組の共役な複素数が現れる．以下，カルダノの公式で得られる式で説明する．式の 1 つ，

$$x_1 = \sqrt[3]{-q + \sqrt{q^2 + p^3}} + \sqrt[3]{-q - \sqrt{q^2 + p^3}} - \frac{a}{3} \quad \cdots (4)$$

を見ると，3 乗根の中に$\sqrt{q^2 + p^3}$ という平方根がある．$D > 0$ の時，即ち $-q^2 - p^3 > 0$ の時，これは負数の平方根となり，虚数となる．しかし，解は実数となることから，第 1 項と第 2 項の和が実数となることがわかる．これはつまり，第 1 項の 3 乗根とそれぞれ共役な複素数の組となるということである．このことから，この式の簡略化を考える．

まず，（4）式に存在する 2 個の 3 乗根のうち，片方の値について考える．複素数の 3 乗根の 1 つは，絶対値がその複素数の 3 乗根で，偏角が$1/3$のものとして求めることが可能である．その他の 3 乗根についても，求まった偏角に$2\pi/3$，$4\pi/3$を加えることで得られる．上式の第 1 項について，$D > 0$ の条件下では，

$$\sqrt[3]{-q + (\sqrt{-q^2 - p^3})i}$$

と置くことが出来る．このことから，実部が$-q$，虚部が$\sqrt{-q^2 - p^3}$ の複素数の 3 乗根を求めれば良い．この複素数の絶対値は，

$$\sqrt{(-q)^2 + \left(\sqrt{-q^2 - p^3}\right)^2} = \sqrt{q^2 + (-q^2 - p^3)}$$

$$= \sqrt{-p^3} \quad \cdots (5)$$

と求まる．また，偏角は，

$$\tan^{-1}\frac{\sqrt{-q^2 - p^3}}{-q} \quad \cdots (6)$$

となる．これらの値から，この複素数の 3 乗根を求める．絶対値は（5）式の 3 乗根をとって，

$$\sqrt[3]{\sqrt{-p^3}} = \sqrt[3]{\sqrt{(-p)^3}}$$

$$= \sqrt[3]{(\sqrt{-p})^3}$$

$$= \sqrt{-p}$$

と求まる．また，3 乗根の偏角は（6）式を用いて，

$$\frac{1}{3}\tan^{-1}\frac{\sqrt{-q^2 - p^3}}{-q} + \frac{2\pi n}{3} \quad (n = 0, 1, 2)$$

と求まる．

続いて，2 個の 3 乗根の和について考えた．第 1 項と第 2 項は共役な複素数の組となるため，結果はそれぞれの実部の 2 倍となる．第 1 項の実部は，絶対値と偏角を用いて，

$$\sqrt{-p}\cos\left(\frac{1}{3}\tan^{-1}\frac{\sqrt{-q^2 - p^3}}{-q} + \frac{2\pi n}{3}\right) \quad (n = 0, 1, 2)$$

と表すことが出来る．$D = -q^2 - p^3 = 0$の場合においてもこの式は適用可能である．よって，$D \geq 0$ の時，3 次方程式 $x^3 + ax^2 + bx + c = 0$ の 3 つの解は，

$$2\sqrt{-p}\cos\left(\frac{1}{3}\tan^{-1}\frac{\sqrt{-q^2-p^3}}{-q} + \frac{2\pi n}{3}\right) - \frac{a}{3} \quad (n = 0, 1, 2) \quad \cdots (7)$$

と表すことが可能である．

最後に，3 つの解のどれを選ぶかについて考察した．パラメータを適当に定め，考察を行った結果，

$$2\sqrt{-p}\cos\left(\frac{1}{3}\tan^{-1}\frac{\sqrt{-q^2-p^3}}{-q} + \frac{2\pi}{3}\right) - \frac{a}{3}$$

が解として適切だと判断した．

- $D < 0$の場合

$D < 0$の場合，実数解 1 つと，共役な複素数 2 つが解として得られるが，ここでは，実数解を求めることを考える．$D < 0$ の条件下では，$\sqrt{q^2 + p^3}$は実数となる．よって，2 つの 3 乗根からそれぞれ実数として値を得ることができれば，解は実数となることがわかる．R での 3 乗根計算は，値が実数であれば実数値が出力される．よって，

$$\sqrt[3]{-q + \sqrt{q^2 + p^3}} + \sqrt[3]{-q - \sqrt{q^2 + p^3}} - \frac{a}{3} \quad \cdots (8)$$

をそのまま計算すればよい．

（7）式，（8）式より，今回策定した 3 変数モデルについて当てはめた結果，

$D \geq 0$ の時

$$e = 2\sqrt{-p}\cos\left(\frac{1}{3}\tan^{-1}\frac{\sqrt{-q^2-p^3}}{-q} + \frac{2\pi}{3}\right) - \frac{a_2}{3}$$

$D < 0$の時

$$e = \sqrt[3]{-q + \sqrt{q^2 + p^3}} + \sqrt[3]{-q - \sqrt{q^2 + p^3}} - \frac{a_2}{3}$$

但し，

$$p = \frac{3a_1 - {a_2}^3}{9}, \quad q = \frac{27a_0 + 2{a_2}^3 - 9a_1a_2}{54}$$

$$a_2 = -(a_p p + a_t t + a_s s)$$

$$a_1 = a_p a_t pt + a_t a_s ts + a_s a_p sp$$

$$a_0 = a_p a_t a_s pts - C$$

を用いて，R によるパラメータ推定を行なうこととした．

R を用いたパラメータ推定は以下のように行なう．

1. モデル式を R 上で定義する
2. データを CSV ファイルから読み込む
3. 非線形回帰分析で用いるパラメータの初期値を決定する
4. 非線形回帰分析を行なう

モデル式を R 上で定義するには，R にて関数として定義する．R では，まとまった計算や場合分けについて，関数として一括計算を行なうことが可能である．以下，関数定義の書式を示す．

```
function(変数 1，変数 2，変数 3，...){
  行なう計算
}
```

また R は，CSV 形式やその他テキストファイルで表されたデータの挿入が可能である．モデル化モジュールでは，性能計測の結果を CSV 形式にまとめ，それを用いてデータ入力を行なうこととした．データの入力方法の書式を示す．

```
read.csv(データファイルのパス，header=ヘッダが存在するか,
                                    col.names=付加する列名のリスト)
```

R は，非線形回帰分析を行なうための関数が定義されている．以下，今回の分析で使用する上での関数の書式を示す．

```
nls(モデルの形式，data=データリスト，start=分析時のパラメータ初期値リスト)
```

非線形回帰を行なう際のパラメータの初期値は,今回は Excel を用いて求めることとする．実測値を散布図としてグラフ化し，同じ図に性能予測モデルの関数を描く．自分の手でパラメータを変更しながら，性能予測モデルの関数のグラフが，実測値にフィットする様なおおまかなパラメータを決定する．

5.1.4 モデル化モジュールの実装

本項では，R を用いたモデル化モジュールの実装について述べる．

今回，モデル化モジュールの実装言語として Java を用いる．これを選択した理由としては，筆者にとって慣れている言語であることや，R を用いるための rJava というライブラリが存在していることが挙げられる．

rJava では，Java と R を接続し，R 内で用いられる S 言語を Java から実行することや，値の受け渡しを行なうための機能を提供している．ここでは，筆者がモデル化モジュールを実装する上で特に使用したクラスについて簡単に説明する．

まず，Java から R に対して，実行したい S 言語を渡す処理は，Rengine クラスが担っている．Rengine クラスの eval メソッドは，引数に S 言語を文字列として渡すことで，戻り値として S 言語を R で実行した際の結果を返す．関数定義やファイルからのデータ入力，パラメータ推定に至るまで，性能予測モデルを作成するための機能は全て Java から実行することが可能である．

また，eval メソッドの戻り値として，REXP クラスのインスタンスを得る．REXP クラスは，浮動小数点数単体や配列，行列等，様々なフォーマットの値を保持している．REXP クラスでは，保持する値を Java のプリミティブ型である double 型に変換して返す asDouble メソッドや，浮動小数点数の配列に変換して返す asDoubleArray メソッド等，保持する値を Java で扱える形に変換するメソッドが多数用意されている．これによって，非線形回帰分析の結果を取得することが可能となっている．

現状では，Java のクラス非線形回帰分析を行なうメソッドを用いて，5 章 1 節 3 項で述べた手順に従ってパラメータ推定を行なうことが可能である．今後の課題としては，まだ見通しの良い実装物となっていないため，クラス分け等のリファクタリングを行なうことを考える．

5.1.5 策定したモデルの評価

本項では，モデルに対する評価について述べる．しかし，現状十分なデータ測定が行えていない．これについては 5 章 5 節に示す．よって，Partqle サーバの台数とトランザクションサーバのレプリケーショングループ数を説明変数とする 2 変数モデルに関する考察を行なう．

モデルの評価は，2 つの方法を用いて行なう．まず，IRS のトランザクション処理性能について計測したデータ(以下，実測値)を用いてパラメータ推定を行ない，その結果として作成されたモデルにより導出される予測値と実測値との比較を行なう．次に，Leave-one-out Cross Validation 法[16]による評価を行なう．この方法は，実測値のうち 1 つを取り除き，その状態で作成されたモデルによって，取り除いたデータを予測できるかを評価する．評価には，2 つの方法共に相対誤差を用いる．

IRS の性能計測は，計測モジュールを用いて行なう．計測方法としては，計測用マシンにスレッドを Partiqle サーバの最大接続数と同じ数起動し，IRS に接続してベンチマークを行なう．計測は，Partiqle サーバの台数が 1～4 台，トランザクションサーバのレプリケーショングループが 1，2 個の場合において，時間を置いて 5 回行ない，各組み合わせの中間値を計測結果としている．なお，Partiqle サーバが 4 台，トランザクションのレプリケーショングループが 2 個の場合における計測は，用意した IRS の環境での台数不足により，行なうことが出来なかった．

以下，実測値を用いたパラメータ推定について述べる．計測モジュールを用いて得られた計測結果を表 5.1 に示す．

パラメータ推定を行なう前に, Excel を用いて手動で初期値の決定を行なう. 結果として，

$$a_x = 1200,\ a_y = 775,\ C = 40000$$

を初期値として用いることとなった．実測値と，パラメータ初期値を設定した予測モデルとの比較のグラフを，トランザクションサーバのレプリケーショングループが 1 組，2 組の場合それぞれについて，図 5-2 と図 5-3 に示す．

表 5.1 IRS のトランザクション処理性能計測結果

T \ P	1	2	3	4
1	744.1	790.3	803.3	810.0
2	1193.3	1320.4	1394.5	

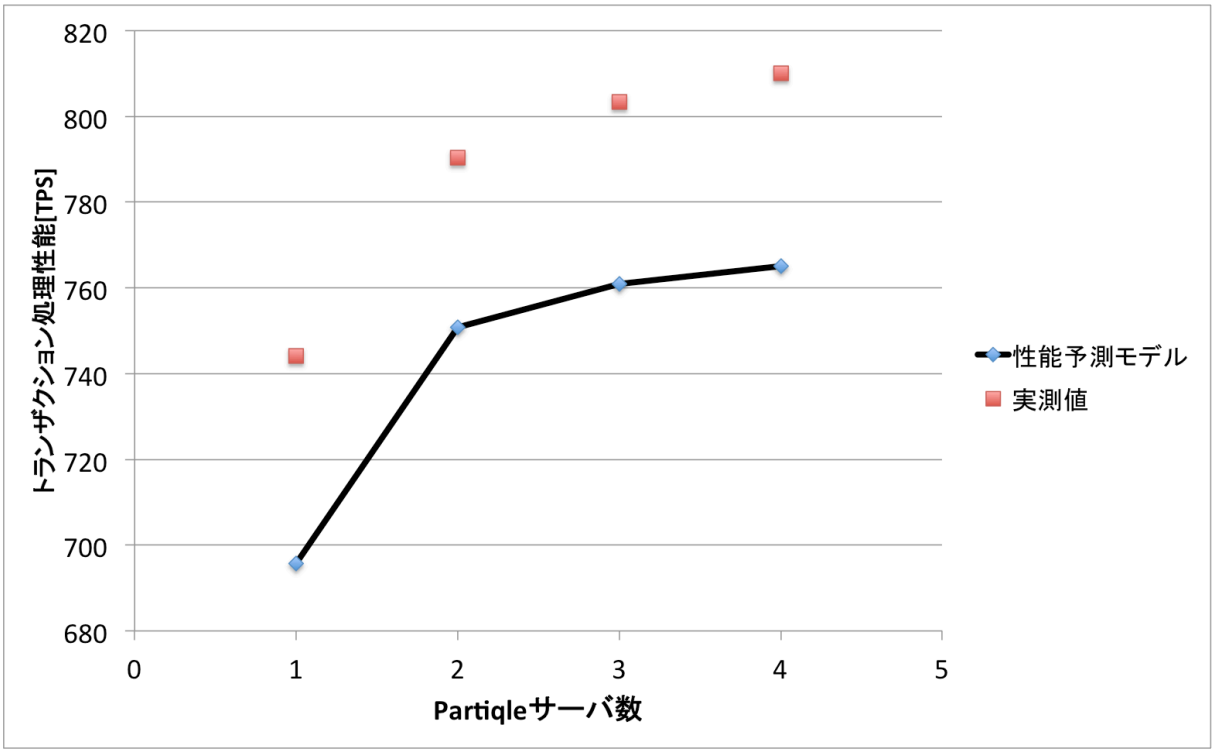


図 5-2　初期値を設定したモデルと実測値の比較(トランザクションサーバ=1)

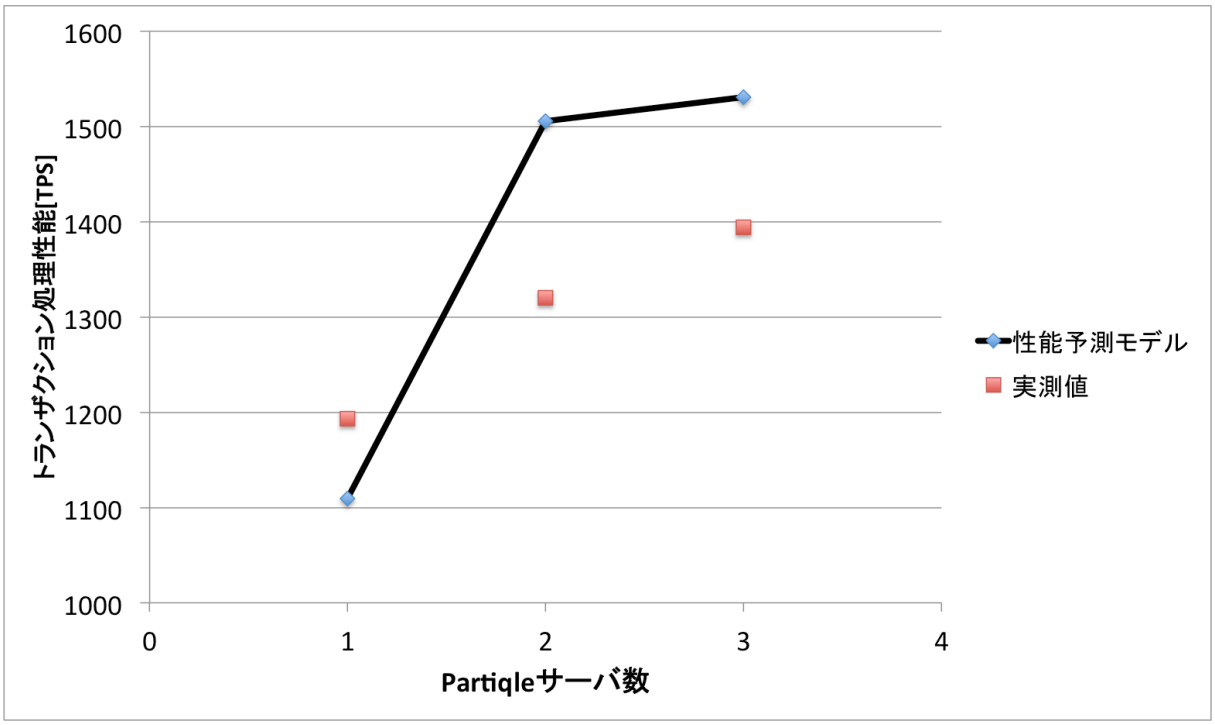


図 5-3　初期値を設定したモデルと実測値の比較(トランザクションサーバ=2)

次に，決定した初期値を用いて非線形回帰分析によるパラメータ推定を行なおうと考えたが，R による推定が上手くいかなかった．よって，異なる初期値を入れて推定を行った結果，以下の様にパラメータが推定された．

モデル式：$(e - a_x x)(e - a_y y) = C$　のパラメータa_x，a_y，Cに対し，

$$a_x = 1159,\ a_y = 710.2,\ C = -8015$$

その結果として，モデル：$(e - 1159x)(e - 710.2y) = -8015$ が導出された．ここでは，xは Partiqle サーバの台数，yはトランザクションサーバの台数としている．このモデルによる予測値と，実測値との比較を行なう．比較した表を表 5.2，比較したグラフを，トランザクションサーバのレプリケーショングループが 1 組, 2 組の場合でそれぞれ図 5-4, 5-5 に示す．

表 5.2　実測値と予測値の差異

トランザクションサーバ	Partiqle サーバ	実測値	予測値	相対誤差
1	1	744.1	728.8	2.095%
	2	790.3	715.2	10.50%
	3	803.3	713.1	12.65%
	4	810.0	712.2	13.73%
2	1	1193.3	1195	-0.09851%
	2	1320.4	1429	-7.627%
	3	1394.5	1424	-2.093%

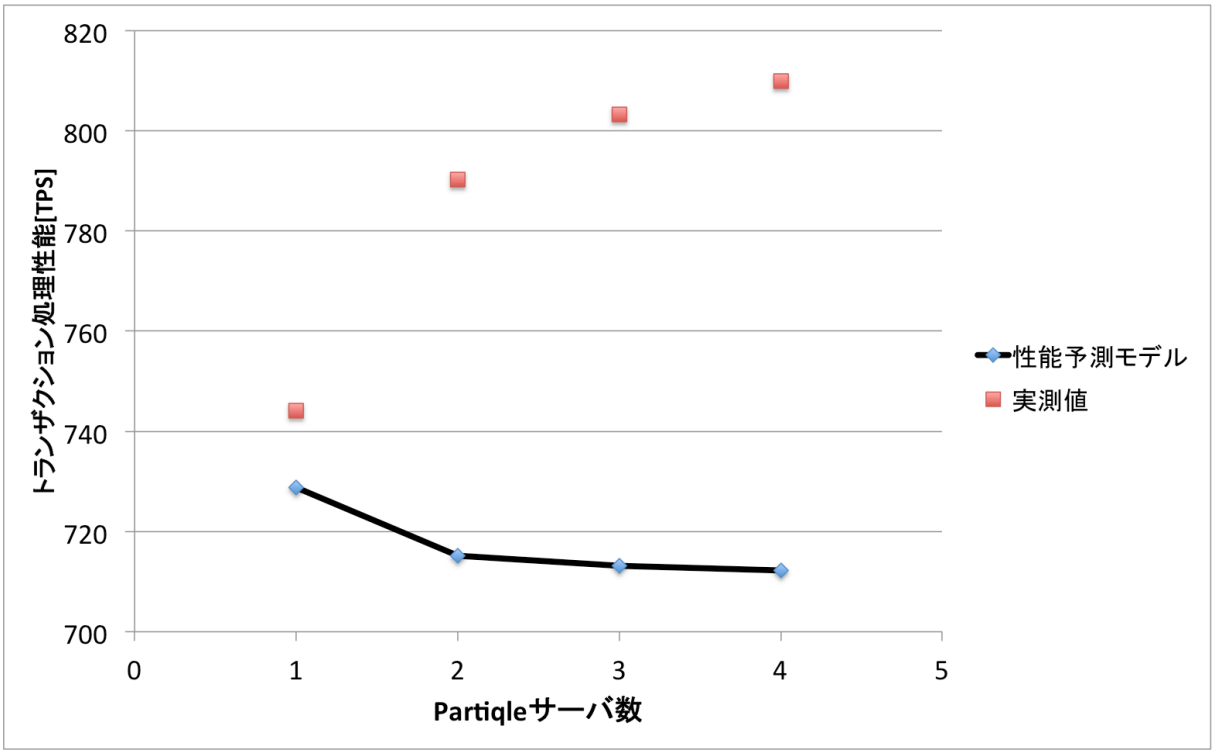


図 5-4　予測値と実測値の比較(トランザクションサーバ=1)

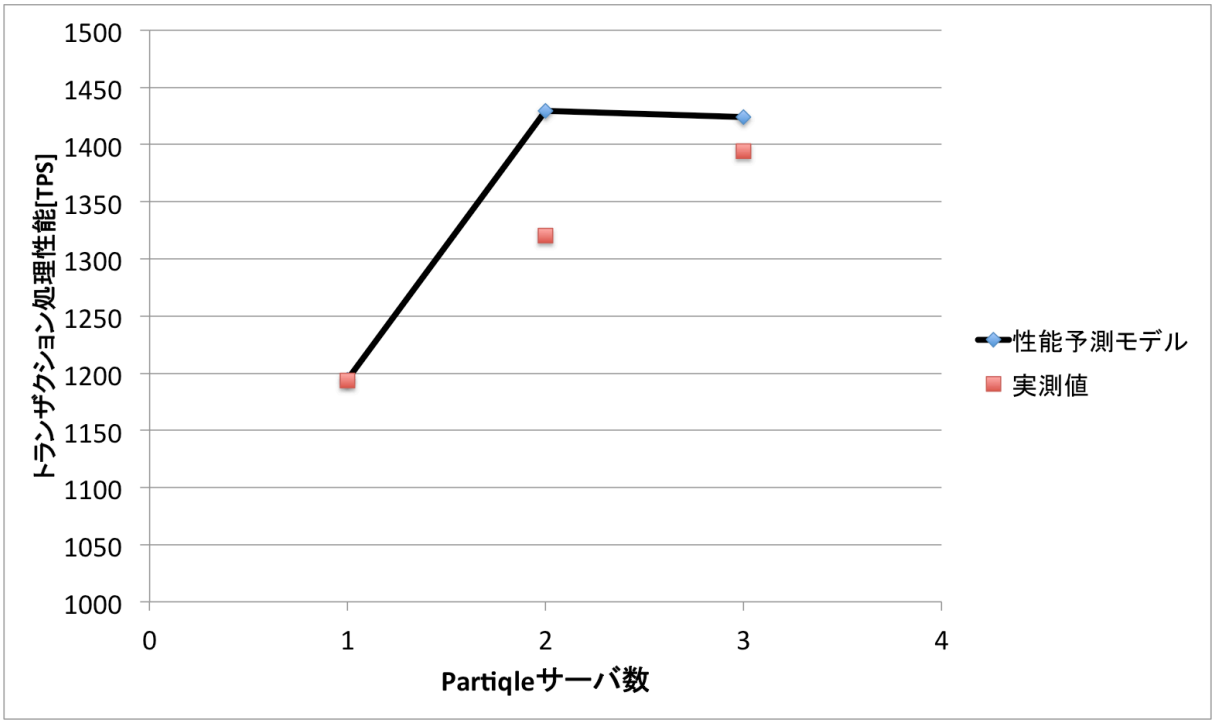


図 5-5　予測値と実測値の比較(トランザクションサーバ=2)

予測値と実測値の比較から，今回策定したモデルの評価，考察を行なう．まず，トランザクションサーバのレプリケーショングループが 1 組の場合について述べる．図 5-3 のグラフを見ると，予測値のグラフは実測値よりも低い値となっており，Partiqle サーバの台数が増加するとともに，性能が減少すると予測された．また，表 5.2 の相対誤差についても，レプリケーショングループが 1 組の場合は，Partiqle サーバの台数が増加するに従って，相対誤差も大きくなっている．レプリケーショングループが 2 組の場合は，1 組の場合よりも予測値と実測値が近い値となっている．相対誤差についても 1 組の場合よりも誤差が小さいが，全体的に負の割合となっており，即ち予測値が実測値よりも大きい値となっている．

この様な結果となった背景には，パラメータ推定の際，パラメータ C が負の値となったことが挙げられる．パラメータ C が負の値の場合において，片方のサーバの台数に着目した時の性能変化を表すグラフの概形を図 5-6 に示す．このモデルで想定していたパラメータ C は正の値であり，負の値の場合，漸近線に対する双曲線の位置が異なる．これでは，よいモデルが策定できたと言えない．

このモデルの Leave-one-out Cross Validation 法による評価について述べる．以下，取り除いた値と，その状態で作成したモデルによる予測値との比較の表を表 5.3 に示す．この表には，比較の他に作成したモデルのパラメータ C の符号も合わせて載せている．これを見ると，相対誤差が大きく，またパラメータ C の符号が 5 つの場合において負になっていることがわかる．よって，このモデルでは予測が上手くできていないと結論付けられる．

策定したモデルが不適当であると考えられるため，新たにモデルを策定することとした．

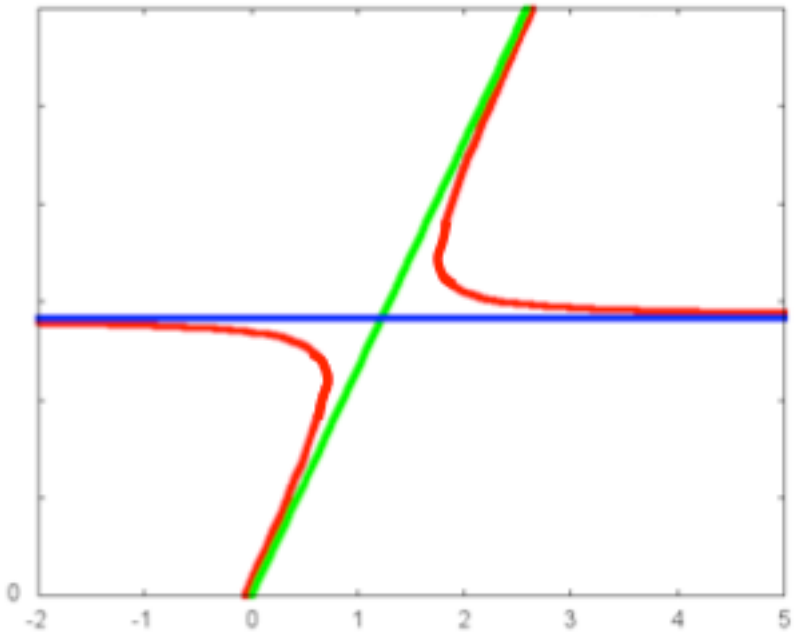


図 5-6　パラメータ C が負の場合のグラフの概形

表 5.3　Leave-one-out Cross Validation 法を用いた評価

トランザクションサーバ	Partiqle サーバ	実測値	予測値	相対誤差	パラメータ C の符号
1	1	744.1	683.2	8.910%	正
	2	790.3	708.4	11.57%	負
	3	803.3	703.7	14.16%	負
	4	810.0	700.6	15.62%	負
2	1	1193.3	1464	-18.47%	負
	2	1320.4	1484	-11.03%	負
	3	1394.5	1448	-3.680%	正

5.1.6　　性能予測モデルの再策定及び考察

この項では，策定した性能予測モデルにおいて，実測値によるパラメータ推定の結果を受けて行なったモデルの再策定について述べる．

当初の性能予測モデルでは，各種サーバの台数やインスタンス数によって線形的に性能が上昇すると仮定して策定を行なった．しかし，実測値から考察した結果，線形的にはならず，線形的よりもやや性能が低くなることが分かった．よって，今回策定した性能予測モデルにおいて線形的な性能上昇を意味する部分を，なだらかに上昇する様に変更することを考える．その場合の，1 種類のサーバの台数に着目した時の性能変化のグラフを図 5-7 に示す．

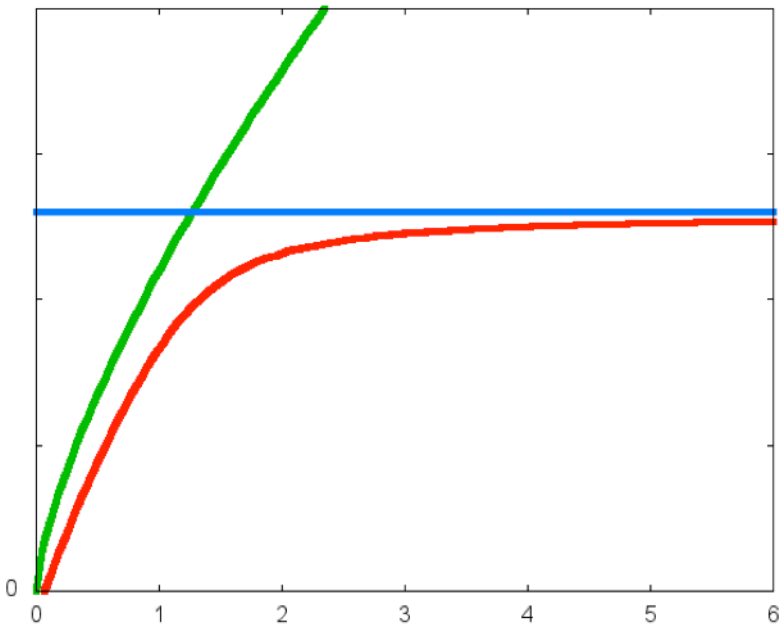


図 5-7　1 種類のサーバに着目した時の台数による性能変化（新策定）

再策定では，小数の指数関数的に性能が上昇すると仮定して策定を行なう．2 変数モデルである（2）式では，性能が線形的に上昇する部分を$a_x x$や$a_y y$と表現していたが，説明変数に指数のパラメータを付加することで，再策定の要件を満たすことが出来ると考える．

よって，再策定した 2 変数モデルが以下のようになる．

$$(e - a_x x^{p_x})(e - a_y y^{p_y}) = C$$

このモデルは，従来の 2 変数モデルの説明変数であるx，yに，それぞれp_x，p_yを指数としている．この中の 5 つのパラメータ，a_x，a_y，p_x，p_y，Cを決定出来れば，それが性能予測モデルとなる．

5 章 1 節 4 項で行なった性能予測モデルの考察と同じ手順で，新しく策定した 2 変数モデルの考察を行なう．

パラメータ推定を行なう前に，Excel を用いて手動で初期値の決定を行なう．その結果，

$$a_x = 1300,\ a_y = 810,\ p_x = 0.7,\ p_y = 0.7,\ C = 30000$$

を初期値として用いることとした．実測値と，新しく策定した性能予測モデルにパラメータ初期値を設定した予測モデルとの比較のグラフを，トランザクションサーバのレプリケーショングループが 1 組，2 組の場合それぞれについて，図 5-8 と図 5-9 に示す．

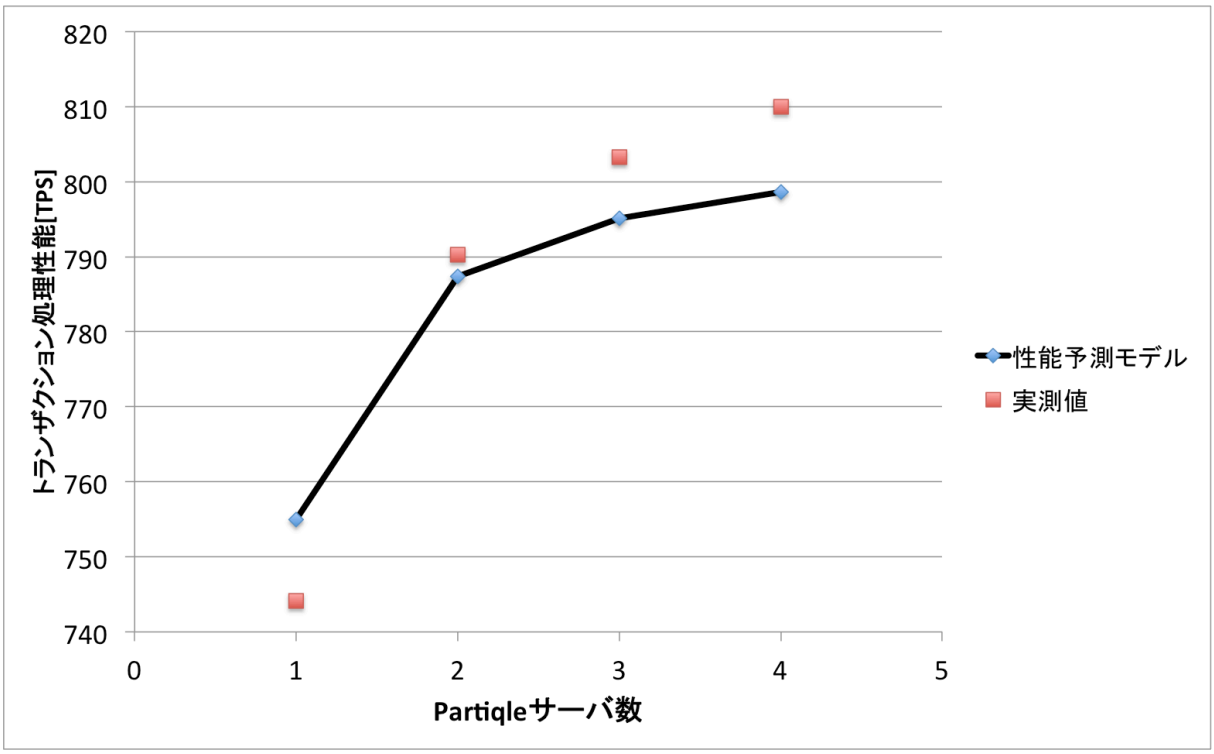


図 5-8　初期値を設定した新モデルと実測値との比較(トランザクションサーバ=1)

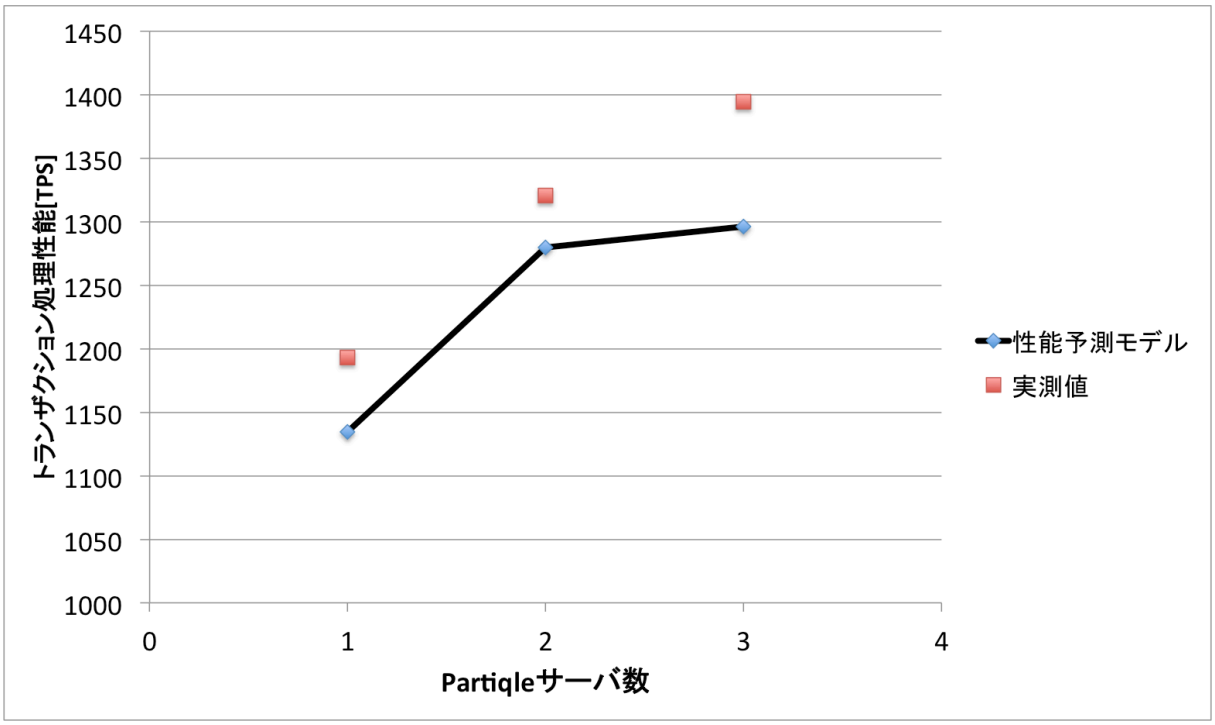


図 5-9　初期値を設定した新モデルと実測値との比較(トランザクションサーバ=2)

続いて，決定した初期値を用いて非線形回帰分析によるパラメータ推定を行なう．行なった結果，以下の様にパラメータが推定された．

$$a_x = 1410,\ a_y = 946.3,\ p_x = 0.1802,\ p_y = 0.9279,\ C = 131800$$

結果として，モデル$(e - 1410x^{0.1802})(e - 946.3y^{0.9279}) = 131800$が得られた．このモデルによる予測値と，実測値との比較を行なう．比較した表を表 5.4，比較したグラフを，トランザクションサーバのレプリケーショングループが 1 組，2 組の場合でそれぞれ図 5-10，5-11 に示す．

表 5.4　新モデルにおける実測値と予測値の差異

トランザクションサーバ	Partiqle サーバ	実測値	予測値	相対誤差
1	1	744.1	747.4	-0.4402%
	2	790.3	784.3	0.7713%
	3	803.3	802.5	0.1059%
	4	810.0	814.0	-0.4901%
2	1	1193.3	1193	0.02588%
	2	1320.4	1322	-0.1235%
	3	1394.5	1394	0.02284%

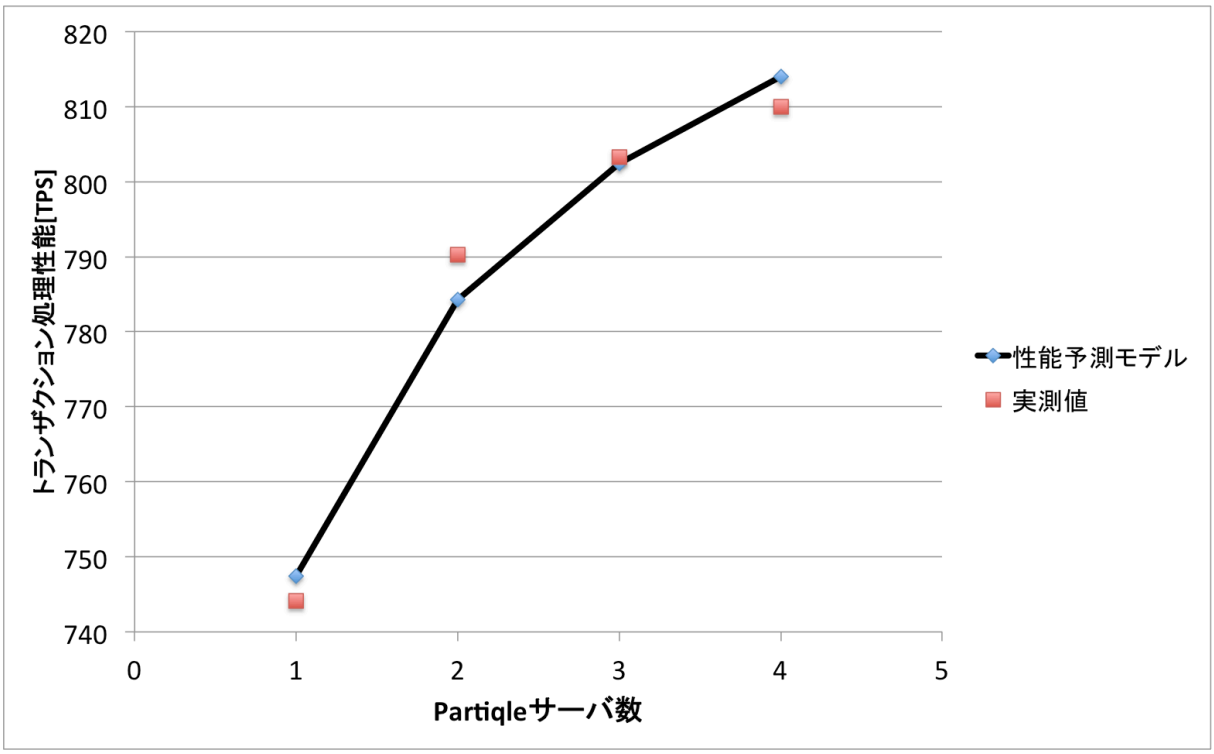


図 5-10　新モデルにおける予測値と実測値の比較(トランザクションサーバ=1)

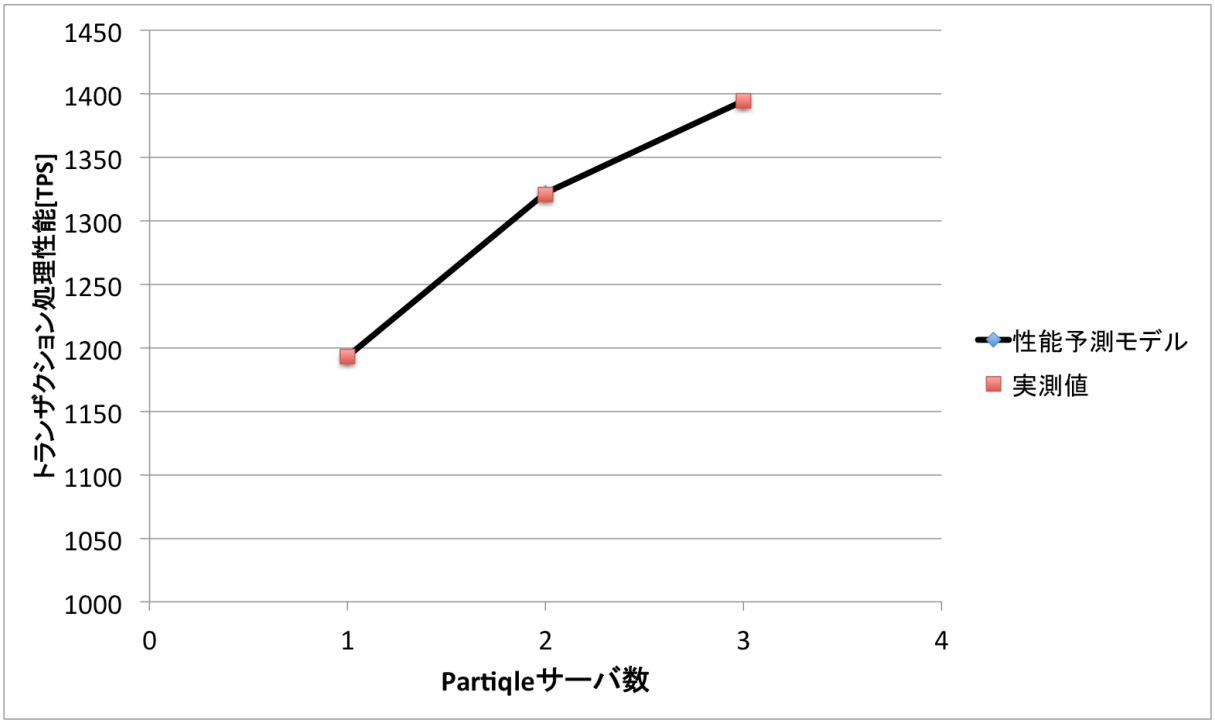


図 5-11　新モデルにおける予測値と実測値の比較(トランザクションサーバ=2)

以下, 新しく策定した性能予測モデルに関する評価, 考察を行なう. 相対誤差については, 全てが 1%以下となっている. グラフを見ても, 実測値に対して予測値が同様な値をとっていることが見て取れる. よって, このモデルは IRS のモデル化に適しているのではないかと考える.

Leave-one-out Cross Validation 法を用いた評価について述べる. 以下, 最初に策定したモデルの評価の際と同様に, 除いた値と, その状態で作成したモデルによって導出された予測値との比較について表 5.5 に示す. この表を見ると, 最初のモデルで問題となっていたパラメータ C の符号は全て正となった. また相対誤差も, 1 つを除いて 5%以内となっている. ここで特に注目したいのが, トランザクションサーバのレプリケーショングループが 1 組, Partiqle サーバの台数が 4 台の時について, それよりも Partiqle サーバの台数が少ないデー

表 5.5　Leave-one-out Cross Validation 法を用いた新モデルの評価

トランザクションサーバ	Partiqle サーバ	実測値	予測値	相対誤差	パラメータ C の符号
1	1	744.1	764.9	-2.72%	正
	2	790.3	781.8	1.09%	正
	3	803.3	801.9	0.17%	正
	4	810.0	819.1	-1.11%	正
2	1	1193.3	1046	14.12%	正
	2	1320.4	1365	-3.28%	正
	3	1394.5	1344	3.76%	正

タを用いて予測が出来ていることである．よって，このモデルは現実を反映したモデルとなっている様に見える．

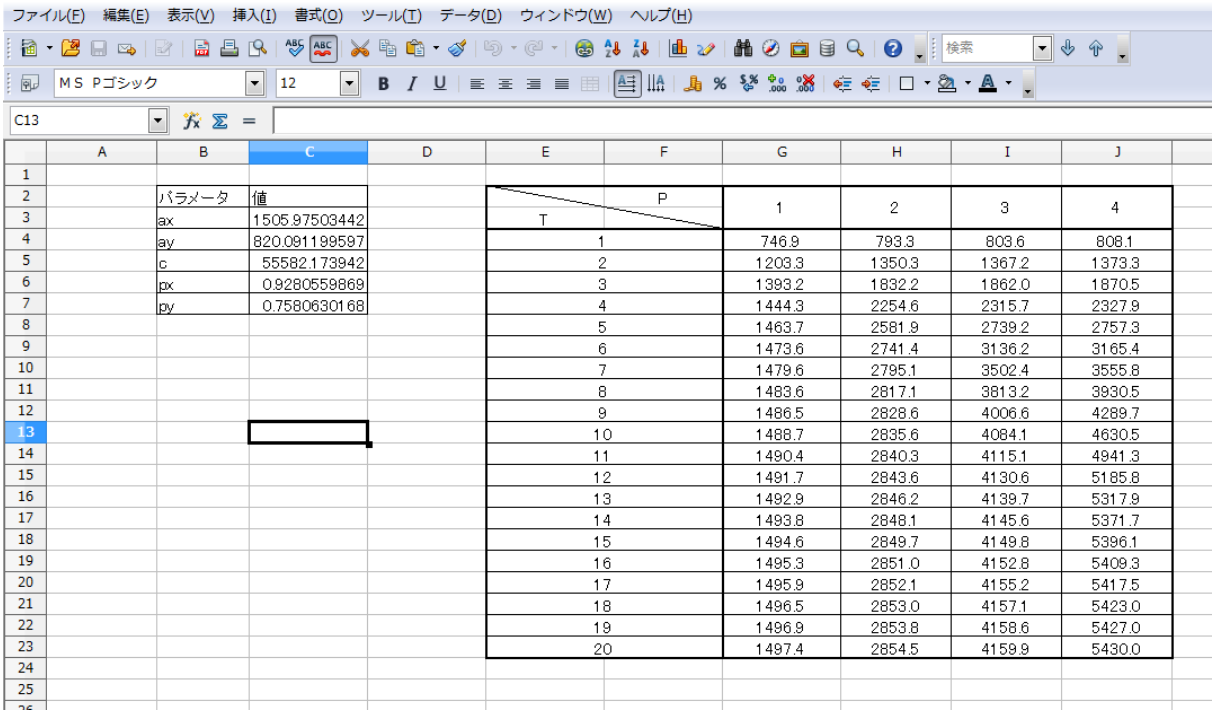
ただし，現状ではモデルの評価を行なうにはデータ数が少ない．新しいモデルのパラメータ数は 5 つあり，統計学的に考えると最低でもその 2 倍のデータ数があるべきであったが，現状では 7 つのデータを得られているのみである．よって，完全な評価を行なうためには，もっと多くのデータを測定する必要がある．また，ストレージサーバを含めた性能の計測，3 変数モデルの評価も行なえていないため，2 月中に行なわれる NEC 様環境での再実験によって，モデルの評価を完了する予定である．

5.1.7 性能予測値の出力を行なう部分の実装

本項では，ODS 形式ファイルを用いた性能予測値出力について述べる．

モデル化モジュールでは，推定されたパラメータによる性能予測値計算やそのグラフが閲覧できる ODS 形式ファイルを出力する．ODS 形式ファイルとは，OASIS(Organization for the Advancement of Structured Information Standards，構造化情報標準促進委員会)[17]によって開発され，国際標準 ISO/IEC 26300 に定められた，表計算ソフトフォーマットである．Apache OpenOffice Calc[18]で作成した表計算ファイルは，この形式で保存され，Microsoft Office の Excel で閲覧，編集をすることが可能である．その実体は圧縮形式である ZIP 形式ファイルであり，表の値や文字のフォント等を設定する XML ファイルが内包されている．それを展開し，内容を編集することによって，表示される値の変更を行なうことが可能である．よって，あるセルを参照する計算式や，グラフのデータ範囲を前もって指定しておくことで，該当部分の編集によって，その編集に則した計算値導出やグラフ作成をすることが可能である．それを利用し，モデル化モジュールで推定したパラメータについて，該当する XML ファイルに書き込み，再圧縮することで，性能予測値やそのグラフの閲覧が可能となる．

パラメータによる性能予測値計算やグラフ閲覧が可能な ODS ファイルを作成するために，ODS 形式ファイル内にある XML ファイル content.xml に対して編集を行なう．このファイルには，表に表示するセルのデータが格納されている．まず，編集を行なうために，あらかじめ準備した ODS 形式ファイルを解凍し，content.xml の，推定したパラメータを入力するべきセルの部分に一意な文字列を設定しておく．次に，モデル化モジュールでその文字列に対し推定したパラメータを置換する．その後，再圧縮を行ない，ODS 形式ファイルに戻す．そのファイルを開くと，性能予測値の表とグラフを閲覧することが可能である．モデル化モジュールから出力された ODS 形式ファイルの，性能予測値の表を閲覧する部分と，グラフを閲覧する部分についてそれぞれ図 5-12 と図 5-13 に示す．パラメータは，図 5-12 のセル C3～C7 に入れるよう，content.xml の編集を行なっている．右の性能予測値の表は，C3～C7



パラメータ	値
ax	1505.97503442
ay	820.091199597
c	55582.173942
px	0.9280559869
py	0.7580630168

T \ P	1	2	3	4
1	746.9	793.3	803.6	808.1
2	1203.3	1350.3	1367.2	1373.3
3	1393.2	1832.2	1862.0	1870.5
4	1444.3	2254.6	2315.7	2327.9
5	1463.7	2581.9	2739.2	2757.3
6	1473.6	2741.4	3136.2	3165.4
7	1479.6	2795.1	3502.4	3555.8
8	1483.6	2817.1	3813.2	3930.5
9	1486.5	2828.6	4006.6	4289.7
10	1488.7	2835.6	4084.1	4630.5
11	1490.4	2840.3	4115.1	4941.3
12	1491.7	2843.6	4130.6	5185.8
13	1492.9	2846.2	4139.7	5317.9
14	1493.8	2848.1	4145.6	5371.7
15	1494.6	2849.7	4149.8	5396.1
16	1495.3	2851.0	4152.8	5409.3
17	1495.9	2852.1	4155.2	5417.5
18	1496.5	2853.0	4157.1	5423.0
19	1496.9	2853.8	4158.6	5427.0
20	1497.4	2854.5	4159.9	5430.0

図 5-12　ODS 形式ファイルでの性能予測値の表の閲覧部分

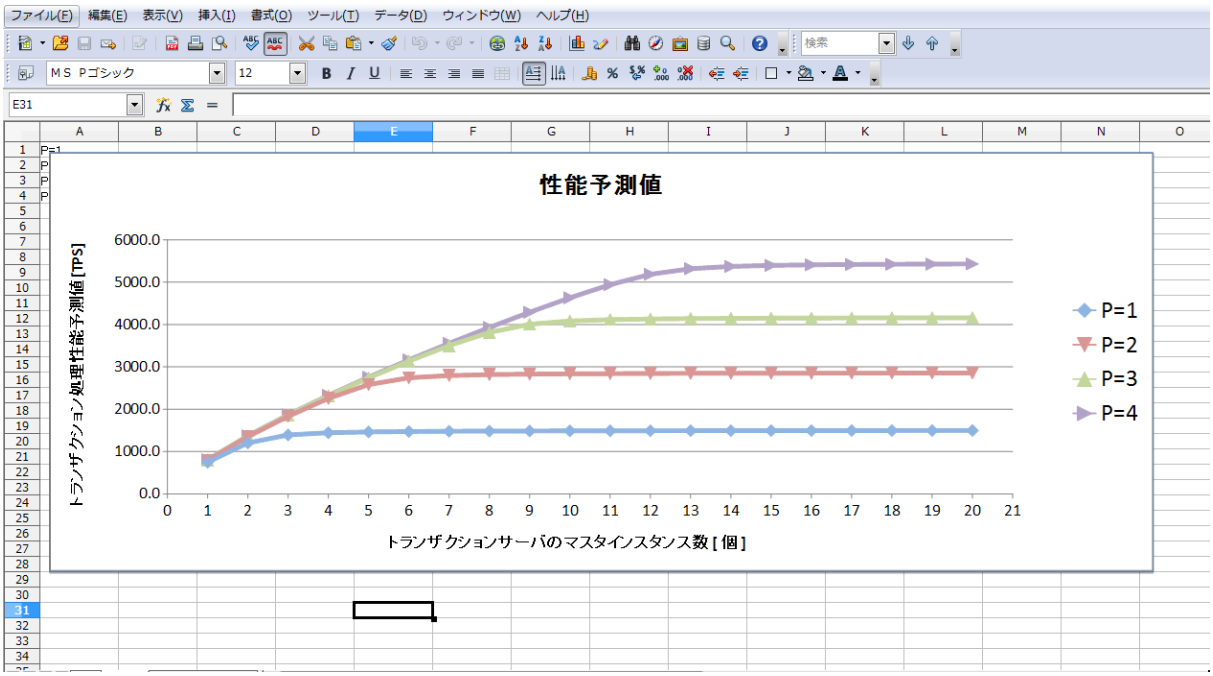


図 5-13　ODS 形式ファイルでの性能予測値のグラフの閲覧部分

を参照して計算をしているため，推定結果が反映されるようになっている．また，図 5-13 に示したグラフは，図 5-12 の性能予測値の表を参照し描かれる．よって，パラメータの部分の編集により，そのパラメータに則したグラフ描画が可能となっている．

この部分についての今後の予定を述べる．現在は特定の台数構成による性能予測値出力を行えない．また，今回は 2 変数モデルで作成されたモデルに関してのみ，ODS 形式ファイル

の出力を行なうことが可能となっている．以上から，Web アプリケーションや GUI アプリケーションによる，特定の台数構成を指定した場合にその構成の性能予測値が得られるシステムや，3 変数モデルを用いた場合の ODS 形式ファイル出力について実装することで，性能予測システムとして使いやすく，役立つものになるのではないかと考える．

5.2 マネジメント

筆者は，プロジェクト全体を通してマネジメント担当として活動し，主に進捗についての決定を行なうことで，プロジェクトの進行が円滑化した．

5.2.1 プロジェクト開始時におけるマネジメント案策定

まず，プロジェクト開始時に，行なうべきマネジメントや，チームとしてどのように活動していくかについて考え，チームが一体となってプロジェクトを推進するための基盤を作ることで，メンバ各々が自分のやるべき作業を考え，プロジェクトを遂行することが出来た．

まず，開発手法としてアジャイルソフトウェア開発手法[19]を採用することを考えた．採用を考えた理由としては，チームメンバが 3 人と少人数なことから，アジャイルソフトウェア開発手法のプラクティスが行ない易いということと，このテーマが自由提案テーマということで，良いアイデアを途中で適用し易く出来るということが挙げられる．

また，進捗マネジメントの方法として，マイルストーンを置いて段階的に実装を進めるという方法を提案した．これを採用しようとした理由としては，メンバ各々が，プロジェクトの作業について確認をしつつ作業が出来，大きな仕様変更等を起こりにくくするという狙いがある．上で述べたチケット駆動開発と合わせて，プロジェクト内の計画を逐次見直すことが可能となる．

それに並行して，チケット駆動開発プロセス[20]の導入を考えた．メンバ個々の作業について，チケットという形で発行することで，設定したマイルストーンに向けて，各々がするべき作業を明確にすることが可能なだけではなく，これを用いた進捗管理が可能である．

以上により，メンバ個々が，プロジェクト全体の進捗を逐次把握することが出来，また一体的にプロジェクトを推進させるための基盤を作ることができた．

5.2.2 マネジメントの実践

本項では，作成したマネジメント案に対し，その実践について述べる．

マネジメント案を策定し，それに沿ってマネジメントや手法を実践したが，結論から言えば,マネジメント案で策定したものを全て実践できたわけではなかった.主な原因としては,4.3 項で述べた環境構築時の遅延が挙げられる．

まず，チケット駆動開発プロセスについては，各人の作業について筆者が Redmine[21]に登録するという形をとった．この形をとった理由として，筆者がメンバに対し，現在行なっている作業は何か,ということについて常に質問を行なっていたことから,その回答を基に,次の作業は何か，どんな作業が必要かということについて把握しやすいことを挙げる．

マイルストーンを用いた進捗マネジメントでは，フェーズ毎にマイルストーンを置くことで，機能を段階的に実装するための目標の設定を行った．まず，2 週間で終わると考えられる粒度で各フェーズを定めた．プロジェクト進行中では，随時プロジェクトの進め方や作業に対するメンバの割り振りについて，積極的に発言を行った．

アジャイルソフトウェア開発手法に関しては，実践できなかった．当初の予定では，開発するモジュール毎にフェーズを定めており，モジュールを更に細分化してメンバに開発を割り振る手法を考えていた．しかし，遅延によって各モジュールで担当者を決めて開発を行なうこととしたため，アジャイルソフトウェア開発に必要な情報共有やコミュニケーションが取りづらくなったため，実践不可能となった．

5.2.3 マネジメントの考察

マイルストーンを用いた進捗管理については，メンバ個々が目標に向かって開発を行なうことが出来たこと等，効果が得られたと考える．チケット駆動開発プロセスも合わせて，マイルストーンに向かって行なう作業について,細かい粒度で作業を明確化することによって,メンバが次に何をするべきか，今何をしているかということに関してある程度共有ができたと考える．

マイルストーンを用いた進捗管理の反省点としては，期間の設定が過度に楽観的だったことと，筆者が Redmine に登録することに限界があったことの 2 点挙げられる．

まず，期間の設定が過度に楽観的だったことについて述べる．今回遅延が発生したのは，第 2 フェーズの環境構築の部分で，2 週間で完了する予定だったが，実際は 2 ヶ月もの時間を費やしてしまった．計画の時点で，この環境構築が上手くいかないことをリスクとして挙げられておらず，対応策を決定していなかったことも，遅延の原因である．またこの遅延によって，筆者を含めたメンバが開発作業に傾倒し，思うように進捗マネジメントができなかったことも，反省として挙げられる．

筆者による Redmine へのチケット登録については，メンバの行なっている作業はメンバ自身がよく知っており，それを聞いて登録することは，逆にオーバーヘッドが大きくなってしまうこととなった．Redmine は Web サービスであるので，個々が責任をもって自分の作業や，プロジェクトとしてするべき作業についてチケットを発行するべきだったと考える．

5.2.4 マイルストーンを用いた進捗管理について

筆者は，メンバが情報共有しつつ見通しのよい開発を行なうために，マイルストーンを用いた進捗管理を行なうことを考えた．この方法では，開発対象モジュールの完成目標毎にマイルストーンを置き，その中で細かく作業を分担する．さらに，1 つのマイルストーンを達成した後，次のマイルストーンについてリスケジュールを行ない，プロジェクトの進捗を管理する．この方法によって，プロジェクトが遅延や障害に対して柔軟に対応できると考えていたが，4 章 3 項に述べた IRS 環境構築において発生した遅延によって，結果として大きな遅延を生んでしまった．

遅延が発生してしまった理由としては，IRS の構築に時間がかかることについてリスクとして考えていなかったことによる．この部分については，我々が用意可能であった物理マシンの性能等を考えると，IRS を構築するのに性能的に不十分であることは明らかであったので，リスクとして考えておく必要があったのではないかと考える．

また，各フェーズを 2 週間毎に設定したことについては，ある程度妥当な判断であったと考える．遅延が発生した第 2 フェーズから第 4 フェーズは，期間として各 2 週間を割り当てており，合計 6 週間の期間ということになる．メンバ各々から現状を聞いた所，それぞれが開発するモジュールについて，2 月末までのプロジェクト完了に向けた開発終了に対する見通しが立っている状態である．実装は，IRS のインストールと起動確認の完了を待って実質 12 月に開始された．更に，メンバが自らの担当部分における実装に傾倒してしまい，モジュール毎の詳細な情報共有や，仕様についての相談等が十分に行われなかった．よって，情報共有や仕様についての決定がメンバ一体で行われていれば，マイルストーンを 2 週間毎に達成することは十分に考えることが可能である．

5.3 トランザクションサーバに関する調査

筆者は，プロジェクト開始時，トランザクションサーバに関する調査を行うことで，トランザクションサーバがトランザクションを処理する方法やスケールアウトによる性能向上の方法についてメンバ間で共有することで，メンバがトランザクションサーバについてある程度理解することが出来た．

この調査は，IRS の 3 種のサーバについて，メンバが 1 つずつ担当して行ったものである．筆者はこのうち，トランザクションサーバの担当として，主にトランザクションサーバがトランザクションを処理する方法について調査を行った．

反省としては，性能計測に関わる細かい部分までの共有を行なうことができなかったことが挙げられる．例えば，トランザクションサーバのスケールアウトに関しては，この調査では完全に理解し，共有することができなかった．これについて深く理解することができていれば，性能計測をスムーズに行なうことができたのではないかと考える．

5.4 IRS インストール時の貢献

筆者は，IRS インストール時に，仮想マシンでのテストの提案と，物理マシン上での IRS インストールを行なうことで，システム開発の環境を用意し，システムの開発を進行させることに寄与することが出来た．

筆者は，IRS 環境を手元に容易する際に，仮想マシン上で，環境として KVM でテストすることを提案することで，物理サーバの環境があまり用意できない中で，IRS の構成に必要な仮想マシンの用意を行なうことが可能になったと考える．

また，仮想マシンを実現する環境として，KVM の使用を提案し，使用を決定した．これを採用した理由としては，筆者が KVM を用いた仮想マシン環境構築を行った経験があることや，CentOS 等で用いられる yum コマンドで容易に構築が可能な事，また，CUI による仮想マシンのコピーや削除が標準で可能であることが挙げられる．以下，仮想化技術の比較について述べる．

- KVM と VMWare シリーズとの比較

KVM と VMWare シリーズを比較した場合，KVM の導入容易性を挙げることが出来る．KVM の導入には yum コマンドを用いるだけでよく，仮想マシンを CUI で操作するためのパッケージを他にインストールする必要がない．VMWare(例えば，ESXi)を用いる場合には，CUI で仮想マシンを操作するためのパッケージをインストールする必要がある．本開発では，自在に仮想マシンを増減するプラットフォームを目指したため，仮想マシンの clone から起動，シャットダウン，そして削除までを CUI で行なうことが必要となっている．よって，本開発では KVM を用いるのがふさわしいと考えた．

- KVM と Xen の比較

KVM と同じような機能を有するものとして Xen[22]が挙げられ，どちらも Linux カーネルに組み込まれている．調査の結果，KVM と Xen の間には，ディストリビューション毎によるサポートの有無に違いがあるのみで，CUI 上での操作や，機能については大きな違いは見られなかった．筆者は，インターンシップにて KVM の操作経験があり，Xen についてはメンバ内に経験者がいなかった．よって今回は，使用になれている KVM を用いることとした．

以上に関連して，仮想マシン上でIRSの構成を変える際，仮想マシンの増減を行なう部分についての実装を行った．概要図を図5-14に示す．

まず，CentOSのマシン上にKVM環境を構築する．次に，コピー元となる仮想マシン（以下マスタVM）を1つ手動でインストールをしておく．その後，KVMのコマンドを用いてマスタVMのクローンを行なうことで，仮想マシンの数を増やすことが可能である．クローンした後，作成された仮想マシンに対して静的IPアドレスを割り当てることで，作成後すぐにssh等でリモートログインすることが可能となっている．仮想マシンを削除する際も，KVMのコマンドを用いることで行なう．

筆者は，この機構の開発において，仮想マシンのクローンや削除等，KVMのコマンドを用いた部分について実装を行った．

この機構により，IRSを構成するために必要な数の仮想マシンを手早く用意することが可能となった．しかし，前述の通り，メモリ容量に関する問題が顕在化してしまったので，この機構については現在使用されていない．

物理マシンで環境を用意する方針となってからは，まずIRSのインストールを担当していた．

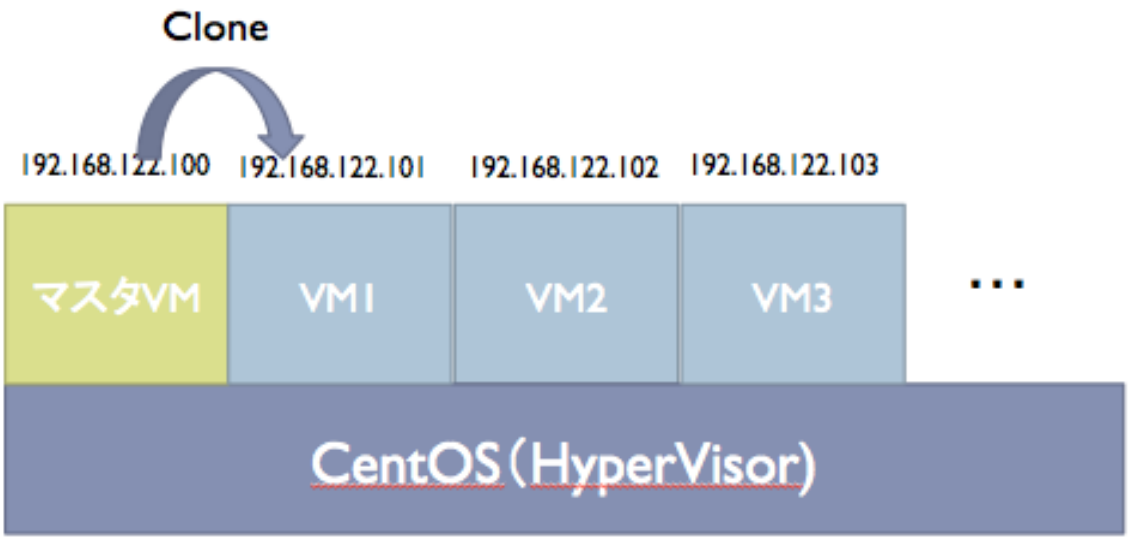


図 5-14　仮想マシンを増加させる部分の概要

5.5 性能計測時の貢献

筆者は，IRS の性能計測を行なう方法の提案を行なうことで，性能計測に必要な要素についてメンバで共有し，理解を深める事ができたと考える．

性能計測について提案を行ったのは，筆者が性能予測モデル策定を担当していたことが理由である．今回，サーバ台数やインスタンス数を説明変数とした 2 変数モデルと 3 変数モデルを策定した．よって，それに必要なデータ数や，サーバの構成について，性能予測担当のメンバと考察を行なう必要があった．

現状の測定方法は，主に Partiqle サーバとトランザクションサーバによるトランザクション処理性能を測定する形となっている．以下測定の流れについて説明する．

1. IRS にデータを 100 万レコード挿入する
2. ストレージサーバへの反映を待つ
3. JDBCRunner を用いて性能を計測する

性能を計測する指標，データモデル等は，程の特定課題研究報告書を参照されたし．

この方法では，Partiqle サーバからストレージサーバへの参照について，精密に性能を測定することが出来ないと考える．理由としては，100 万レコードのデータは容量 250MB と，現環境で用いているストレージサーバの HDD 容量 250GB の 1000 分の 1 となっていることを挙げる．これでは，データの検索について十分な測定を行なうことができない．よって，さらに多くのデータをストレージサーバに挿入して計測を行なう必要があると考える．

筆者が提案した方法は以下の通りである．

1. トランザクションサーバの搭載メモリ容量の半分程になるくらいデータを挿入する．
2. ストレージサーバへの反映を待つ
3. キーが異なるデータについて 1 を行なう
4. ストレージサーバへの反映を待つ
5. データ挿入処理と，トランザクションを同時並行で行なうようなシナリオで性能計測を行なう

この方法であれば，ストレージサーバの処理も考えつつ，IRS 全体の処理性能を測ることが出来ると考える．

しかし，IRS にデータを挿入してから，ストレージサーバへ反映するまで，データ量によって膨大な時間が必要となる．よって，今回はこの方法で行わず，従来通りの方法で計測を行なうこととなった．

5.6 その他の貢献

筆者は，学内での中間報告及び学外での発表にて，発表者を担当することで，プロジェクトについて行なっている事について対外的に分かりやすく報告することができた．

学外での発表では，NEC 様のプライベートセミナーである iEXPO 2012 に参加させて頂き，本プロジェクトの内容について 10 分程度の発表を行った．その結果，NEC 様の顧客先企業に務める聴者の方から開発システムについての質問，アドバイスを頂いた．

iEXPO で聴者から頂いた質問，アドバイスは，今回開発しているシステムが，どちらかというと SE が使用するものに感じたという内容だった．本プロジェクトで開発している性能予測システムでは，トランザクション処理性能の性能予測を行なうが，実際に IRS をサービスに導入するユーザの方々からすると，トランザクション処理性能よりも，どの程度のサービスに適用可能かという部分が知りたいのでは，ということである．そこで，今後の展望として，性能予測システムを用いてトランザクション処理性能から，どの程度のサービスに適用可能かを予測するツールを開発できるのではと考える．

第6章　プロジェクトの振り返り

本プロジェクトは，筆者にとって未知な，プロプライエタリシステムを用いたプロジェクトということで，それに関する難しさを知る良い機会だったと考える．以下，本プロジェクトの遅延と，性能計測に関する部分で得た教訓について述べる．

今回のプロジェクトで一番時間がかかった部分は，本プロジェクトの本質ではない IRS のインストールの部分だった．インストールが上手くいかないときに，学生チーム内で試行錯誤をし過ぎてしまったことが，本プロジェクトの大きな遅延を生んでしまった．インストールが思うように進まなくなった段階で NEC 様に問い合わせをすることが，学生チームには必要だった．この経験から，プロプライエタリシステムを用いるプロジェクトにおいては，その開発元との密接なコンタクトを取るべきであるという教訓を得た．

また，性能計測の部分においては，従来のデータベースを測定する方法ではうまくいかず，そこでの試行錯誤に苦労した．性能計測モジュール等の機能の実装は，計画ではモジュールを更に細分化し，メンバ全員で機能を 1 つずつ実装していく方法を取る予定であったが，実績ではモジュール毎に担当者を分けて開発を行った．そのため，性能計測の部分を担当者にほぼ任せることになってしまった．そこである程度共有が出来ていれば，性能計測をさらにスムーズに行なうことが可能であったと考える．インストール時に発生した遅延を取り戻すために，メンバ各々が各自の作業に集中し過ぎってしまったことが，十分な共有ができなかった原因と考える．これは，筆者についてもモデル策定に傾倒し過ぎたことで，進捗マネジメントが上手く行えなかったことも 1 つの原因であると言える．この経験から，チームでの開発における情報共有の大切さに改めて気づくことができた．

筆者が担当した性能予測モデル策定は，出来れば性能計測の結果を待って始めたい部分であった．未経験の R を用いたパラメータ推定や，数学を用いたモデル策定に調査がかかるため，前もって始めたのは良かったと考えるが，性能計測の結果から，トランザクション処理性能の傾向を見て，それからモデル化を行なうことができていれば，モデル策定をもっとスムーズに，方向性を定めながら進めることができたと考える．

第7章　おわりに

本プロジェクトでは,「InfoFrame Relational Store を利用した研究開発」というテーマの元，NEC 様に対し IRS 予測システムについて提案し，開発を行なった．

筆者は開発において，IRS の性能を予測するためのモデル策定と，統計処理アプリケーションである R を用いた非線形回帰分析でパラメータ推定を行なうモデル化モジュールの実装を担当した．今回は，2 変数モデルと 3 変数モデルについて策定を行ない，性能計測方法や環境の不足等を理由として，2 変数モデルの評価と考察を行なった．その結果，策定したモデルが IRS のモデル化に適さないと結論付けた．そこで，新たにモデルを策定し再評価，考察を行なった．結果としては，誤差も十分に少なく，データのばらつきを吸収していると評価出来る．しかし，まだデータ数が絶対的に少ないため，今後更なるデータ測定を行ない，そのデータを用いて評価を行なう必要がある．

筆者は，プロジェクト内で主にマネジメント担当として活動した．本プロジェクトでは，マイルストーンを用いた進捗管理を取り入れて開発を行なったが，遅延によって開発方法の変更が発生した．遅延は IRS 構築の部分で発生したが，前もってリスクとして挙げていなかった部分であった．プロジェクトで発生し得るリスクを回避するために，進捗管理と合わせてリスク管理を行ない，マイルストーン達成毎にリスクの評価をし直すことが必要だったと考える．

謝辞

日本電気株式会社の白石雅己様，川畠輝聖様には，テーマ提供並びに iEXPO での発表のご支援，数々のアドバイスを頂き，プロジェクト全体に渡るご指導，ご協力を頂けましたことを深く感謝致します．

本プロジェクトを進めるにあたり，指導教員の田中二郎教授，ならびに課題担当教員の早瀬康裕助教，山戸昭三教授には，多くの助言，ご指導を頂きました．心より感謝致します．

また，共にプロジェクトを遂行した村上貴裕氏，程鵬氏には，プロジェクトを通してとても良い経験をさせて頂きました．誠にありがとうございます．

参考文献

[1] Codd, E., F.: “A Relational Model of Data for Large Shared Data Banks”, Communications of the ACM, Vol. 13, No. 6, pp. 377-387, June 1970

[2] 首藤一幸, “key-value ストアの基礎知識,” Software Design, 2010 年 2 月号, p.14-21, (株)技術評論社, Jan.2010.

[3] DeCandia, G., Hastorun, D., Jampani, M. Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: ”Dynamo: amazon’s highly available key-value store”, SOSP ’07 Proceedings of twenty-first ACM SIGOP symposium on Operating systems principles, pp. 205-220, October 2007, ISBN 978-1-59593-591-5.

[4] NEC, ”InfoFrame Relational Store”, http://www.nec.co.jp/pfsoft/irs/

[5] Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., Schwarz, P.: “ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging”, ACM Transactions on Database Systems, Vol 17, No. 1, pp. 94-162, March 1992

[6] “Project Voldemort: A distributed database.” http://project-voldemort.com/

[7] M.I. Seltzer, “Berkeley DB: A Retrospective”, In Proceedings of IEEE Data Eng. Bull., 2007, 21-28.

[8] Vector. “JdbcRunner – 汎用データベース負荷テストツール” http://hp.vector.co.jp/authors/VA052413/jdbcrunner/

[9] Michael J.Crawley, 野間口 謙太郎, 菊池 泰樹, ”統計学:R を用いた入門書”, 共立出版(株), May.2008

[10] “CRAN – rJava Package” http://cran.r-project.org/web/packages/rJava/index.html

[11] 村上 貴裕, ”InfoFrame Relational Storeを利用した研究開発 –性能計測を目的としたIRS自動構成システムの開発および開発と意見調整を進めるための取り組み–“, 平成24年度筑波大学特定課題研究報告, March 2013.

[12] 程 鵬, ”InfoFrame Realational Storeを利用した研究開発 –性能計測指標の選択と計測システムの開発および成果物管理“, 平成24年度年度筑波大学特定課題研究報告, March 2013.

[13] “OASIS Open Document Format for Office Applications (OpenDocument) TC” https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

[14] 平 初他，”KVM 徹底入門 Linux カーネル仮想化基盤構築ガイド”，(株)翔泳社, Jun.2010

[15] “三次方程式の解の公式[物理のかぎしっぽ]”
http://hooktail.sub.jp/algebra/CubicEquation/

[16] “Cross Validation”
http://www.is.doshisha.ac.jp/report/2011/9/29/2011045007/index.html

[17] “OASIS”, https://www.oasis-open.org/jp/

[18] “無料総合オフィスソフトウェア – Apache OpenOffice 日本語プロジェクト –“
http://www.openoffice.org/ja/

[19] Alister Cockburn, “アジャイルソフトウェア開発”，ピアソン・エデュケーション, Aug.2002

[20] 小川明彦, 阪井誠, ”チケット駆動開発 –BTS で Extreme Programing を改善する”, 日科技連ソフトウェア品質シンポジウム 2009, 2009

[21] “Redmine”， http://redmine.jp/

[22] Barham, P., Dragovic, B., Fraser, K., Hand, S., and Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A., “Xen and the Art of Virtualization”, 19th ACM Symposium on Operating Systems Principles (SOSP ’03), pp. 164–177, 2003.

付録

付録一覧

付録として以下のドキュメントを添付する.

- プロジェクト初期におけるマネジメント案
- 筆者が担当したモデル化モジュールの定義について

プロジェクト初期における
マネジメント案

マネジメント案

作成者：菊池大輔

作成日：2012 年 7 月 15 日

手法

- アジャイルソフトウェア開発 + チケット駆動開発

  XP(eXtreme Programing)の 5 つの価値からリスクを分析

  - コミュニケーション

    コミュニケーションについてのリスク

    - 開発方針の食い違いが起こる可能性
    - 開発の進捗具合が見えなくなる可能性
    - 顧客とのコミュニケーション不足になる可能性

    コミュニケーションに関する対策

    - SSH + Screen または tmux

      ターミナルの共有により，開発に関する意識合わせを積極的に行なう．
    - Redmine を用いたチケット駆動

      仕事を細分化し，チケット単位にする．やるべき仕事を明確にすることで，方針の食い違いを低減し，進捗具合を把握しやすくする．
    - 定期的な顧客とのコミュニケーション

      隔週で電話会議，もしくは NEC 本社でのミーティングの機会を設ける．

- シンプル

  シンプルさについてのリスク

  - 可読性に優れないコードを書く可能性
  - ソースコードの版数について把握不可能になる可能性

  シンプルさに関する対策

  - 詳細なコーディング規約の設定

    フィールド，メソッド，クラス，ファイル名に至るまで，命名規則を設ける．また，if やメソッドの使い方等は意識合わせを行なう．
  - Git を用いた版数管理

    分散バージョン管理 Git を用いて版数を明確にする．
- フィードバック

  フィードバックに関するリスク

  - 自分がやったことについてフィードバックを忘れる可能性

  フィードバックに関する対策

  - チケット駆動開発

    チケット駆動開発により，どの作業が終了したか，もしくは作業中かを明確にする
  - ターミナル画面共有

    ターミナルの画面共有により，個々の進捗具合を随時把握する
- 勇気

  勇気に関するリスク

  - ソースコードの変更について責任を持つことを恐れ，手が進まない可能性

  勇気に関する対策

  - コーディング規約

    コーディング規約に，ソースコードの変更を行う際の手順を明記しておく
  - Skype

    Skype を用いることにより，メンバの意識を統一し，1 人で迷うことを少なくする

- 尊重

  尊重に関するリスク

  - 個々がチームメンバの意見を認めず, 勝手に作業を進める可能性

  尊重に関する対策

  - 個々の意見を尊重

    出た意見に関して, 自分の意見との折衝についてきちんと考えて発言する．意見については，論理的であることが望ましいが，ある程度直感的に出た意見も考慮する

マネジメント

- 進捗マネジメント
  - 方法
    - チケット駆動開発

      チケット駆動とすることによって，仕事の進捗を明確にする．(ex.どの仕事を誰が担当しているか，どの仕事が完了しているか)
    - 反復スケジュールの作成

      アジャイル手法では，1 反復(イテレーションという)を 2 週間程で行なう．1 反復での仕事もそうだが，全体としてどれくらいずつ作業を行なうかを把握する必要がある．
  - 課題
    - チケットの粒度による作業時間の増減

      チケットの粒度に大小があると，チケットによって作業時間が増えたり減ったりすることが容易に想像できる．よって，チケットの粒度をなるべく均一にするか，もしくはチケットにそれぞれ計画出来高(PV)を設定する必要がある．

- リスクマネジメント
  - 方法
    - XP の「5 つの価値」の分類でのリスク管理

      アジャイル開発手法である XP の 5 つの価値は，アジャイル開発を行なう上で必要な事や価値観を十分に示している．そこで，その価値毎にリスクの管理を行なうことで，よりアジャイル開発に身近な管理が可能である．
    - 締め切りの徹底

      締め切りに対する意識を上げる．具体的には，提出物の締め切りを把握し，それの確認に掛かる時間を推測した上で，第 1 版提出等の予定を立てる．
  - 課題
    - 「5 つの価値」分類での分類漏れ

      5 つの価値で分類した時に, そのどれにも属さないリスクが存在し得る．そうしたときに，新しい分類を MECE 的に考える必要がある．

モデル化モジュールについて

モデル化モジュールについて
作成者：菊池大輔

定義

　モデル化モジュールは，性能予測システムで用いるモデルを作成するモジュールである．集計モジュールにて集計した性能計測の実測値と予測モデルの式の雛形を入力として，式の雛形に対しパラメータ推定を行なう．結果として導出されたパラメータを出力とし，性能予測システムに渡す．

要件

　モデル化モジュールの要件は以下のようなものとなる

- IRS の性能をよく表すことが出来るようなモデルが策定されること
- 回帰分析によるパラメータ推定を行なうこと
- 集計モジュールから集計された実測値を受け取ること
- 性能予測システムへ予測モデルもしくはパラメータを渡すこと

仕様

- 開発言語は主に Java，または R を扱うための S 言語
- Java から R を用いるために，rJava ライブラリを使用
- 性能予測モデルは双曲線関数を基にした 2 変数モデルと，それを拡張した 3 変数モデル
- パラメータ推定方法は R を用いた非線形回帰分析を採用
- モデル化モジュールへの実測値の入力には CSV 形式ファイルを使用