大規模基幹システムの
Web・オープン化に伴う性能評価について

富士ソフト（株）

■ 執筆者Ｐｒｏｆｉｌｅ ■

伊原　智博

2000 年　富士ソフトＡＢＣ（株）入社
　　システム開発担当
2002 年　現在のプロジェクトにて性能評価・
　　サイジングを主にサーバ構築に従事
2006 年　現在 IT 事業本部産業ｼｽﾃﾑ事業部
　　横浜ｼｽﾃﾑ部第 5 技術ｸﾞﾙｰﾌﾟ所属

■ 論文要旨 ■

近年発達した Web システムはビジネスのフロントエンドでは不可欠となっている．また，汎用機と比較しオープン系システムはコストを抑えて導入できることから，Web・オープン化は大きな流れとなっている．しかし汎用機を用いた基幹システムの Web・オープン化は技術的な問題以外に信頼性の確保が大きな障害となる．

ある全国規模の保険システムにおいても，基幹システムの Web・オープン化を行うこととなり，高いレベルの信頼性を要求された．基幹システムの Web・オープン化の信頼性確保として，性能評価は欠かすことのできない要素であり，特に下記ポイントを留意して検証を進める必要があった．

・全国に展開する数万台の端末からオンライン処理を受付ける大規模システム
・開発規模が大きく，多くのアプリ開発グループに別れている
・サブシステムも複数存在し，業務内容の全体把握は困難

ここでは，大規模基幹システムの Web・オープン化に伴う性能評価について，効率的かつ高い精度の評価を目指した取り組みと，その際に発生した問題点について述べる．

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

私は４年程前から性能評価に従事しているが，性能評価を専門とした書籍は少ないと感じている．実際のシステム開発においても性能評価を専門としている方は少ないのではないだろうか．幸い私はその機会に恵まれ，性能評価やサイジングと言った作業を中心に行っている．

作業にあたっては，富士通殿事業部よりご協力頂いており，拝見した性能評価手順・過去の事例については，大変参考となった．そこで性能評価についても手順化と過去事例の蓄積は有益であると考えており，あるシステムの開発に取り組む際は，性能評価の手順書と同種の過去事例は利用すべきと考えている．

今回行った性能評価については，トレンドの Web 化・オープン化であるとともに保険業務における大規模基幹システムという高信頼性が求められるシステムが対象となる．今回取り組んだ評価方法については，同種の他システムにおいても共通すると考えられ，その際に性能評価を実施される方の参考になれば幸いである．

１．１　システム概要

今回性能評価を行ったシステムは，保険業務の基幹システムであり，全国の支店に配置されている数万台の端末が接続する大規模オンラインシステムである．また，処理はホスト端末エミュレータ（以下エミュレータと記す）を使用したホスト処理業務とクライアントサーバ処理業務で連携して行われている．構築タイミングが異なっていたため，複数のオンラインシステムが存在しており，ユーザの利便性において統一が図られること，保守・運用コストが増大していることが課題となっていた．

基幹業務部分の機器は５年以上前に本稼動されており，機器のリプレースに合わせ，基幹となる２業務について，次に示す変更が計画され，コスト削減や利便性向上，システム拡張性の確保を行った．

① エミュレータを使用したホスト処理業務

(移行のイメージを「図 1-1-1 ホスト処理業務の移行概要図」に示す)

- ホスト(前処理)部分をオープン化
- エミュレータから Web へフロントビューを変更

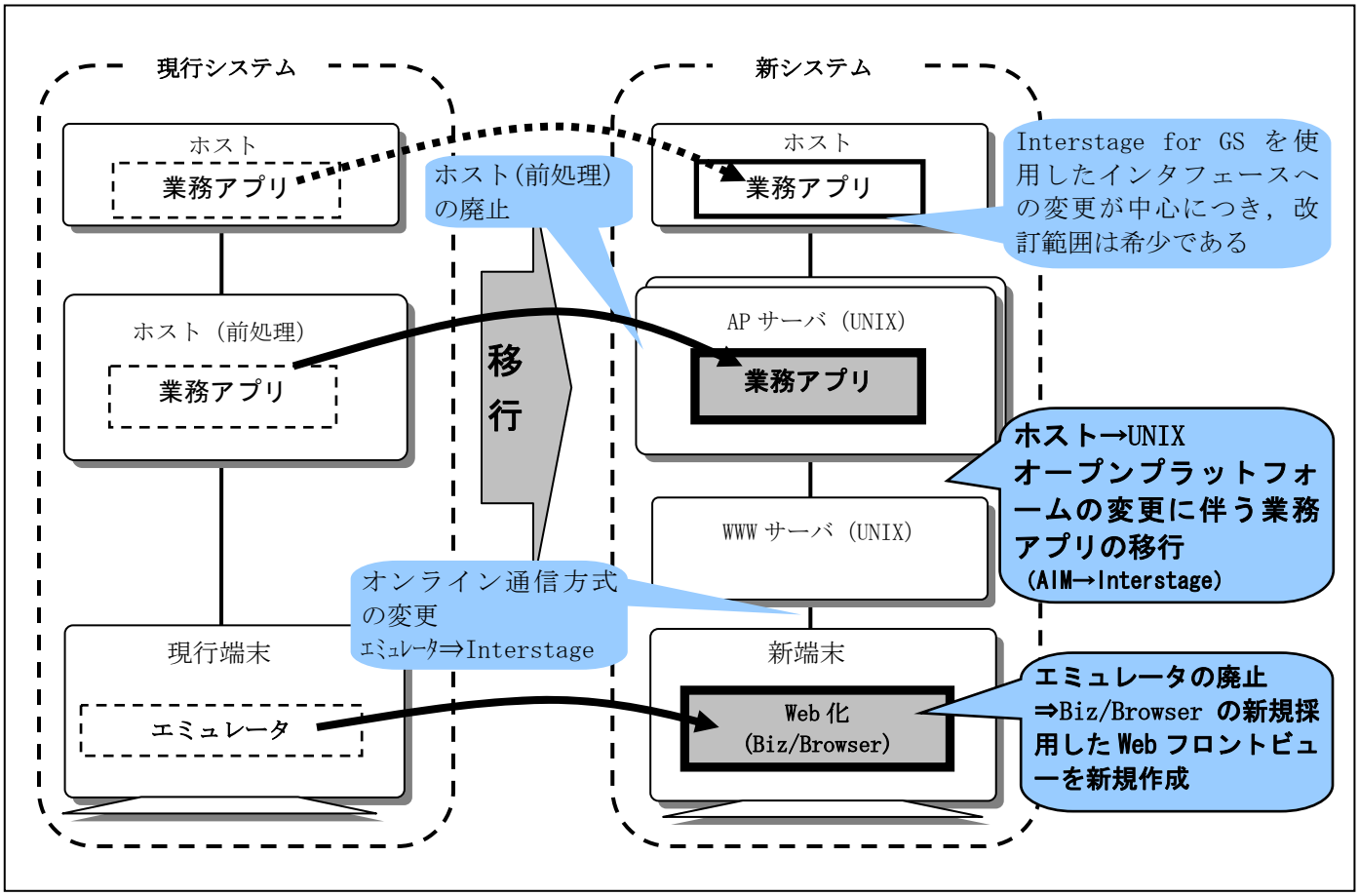

図 1-1-1 ホスト処理業務の移行概要図

② クライアント・サーバ処理業務

- 新規サーバ機器の導入
- 複数の県毎に配置していたＤＢを統合

移行に際しては基幹システムであることから高い信頼性が求められる．特に Web 化・オープン化を行ったホスト処理業務については，オープン化による信頼性の低下が懸念されることから，十分な対策を講じる必要があった．信頼性として欠かすことができない要素として性能評価があり，本文ではトレンドである Web 化・オープン化を行った①の性能評価について記載することとする．

１．２　性能の確認ポイント

性能評価にはいくつかの役割があるが，集約すると下記の確認ポイントを評価する事であると私は考えている．また，確認ポイントのイメージを「図 1-2-1 性能評価の役割」に示す．

① ユーザが満足できるレスポンスが得られるか
② バッチ処理時間が与えられた運用時間内に収まるか
③ システムが常に①②を維持できるキャパシティを確保しており，かつサイジングが成されているか

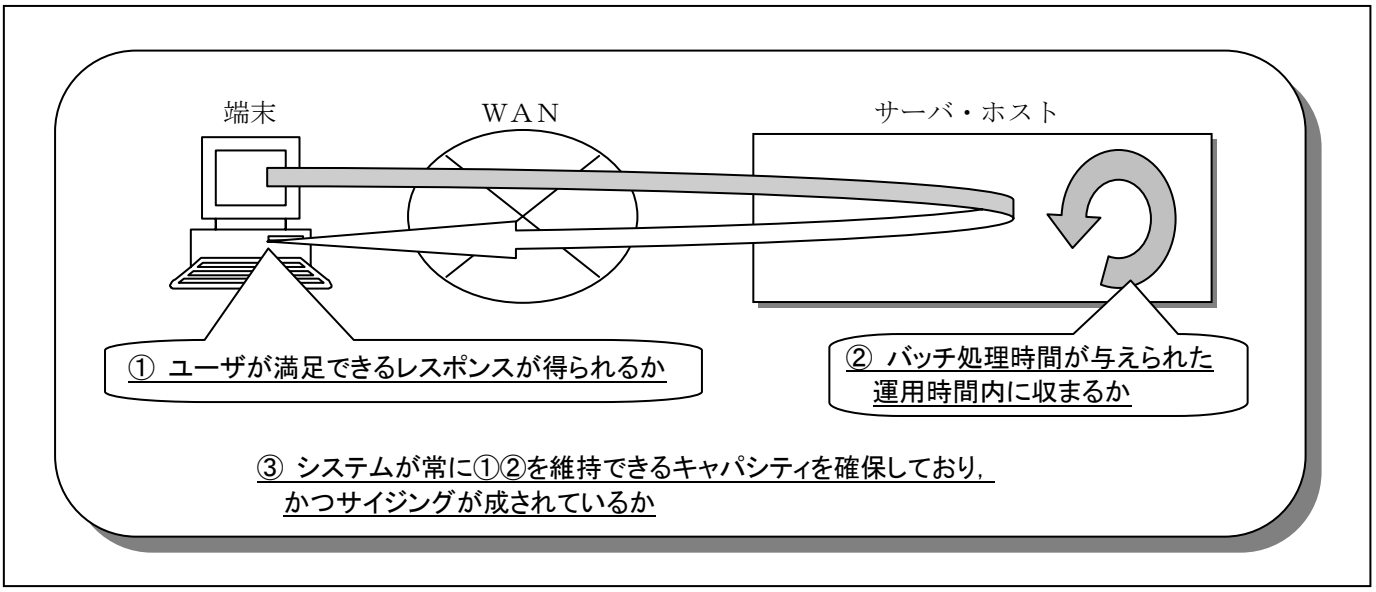

図 1-2-1 性能評価の役割

今回のシステム開発において，①～③をどのようにして，効率的かつ高い精度で実施するかを課題とし，性能の確認方法について検討した．

そこで，本システムの性能確認を行うにあたっては，下記の特徴十分に考慮した上で検証を進める必要があると考えた．

・全国に展開する数万台の端末からオンライン処理を受付ける大規模システム
・開発規模が大きく，多くのアプリ開発グループに別れている
・サブシステムも複数存在し，業務内容の全体把握は困難

２．本論

２．１　基本方針

前項で述べた本システムの留意すべきポイントを踏まえ，下記を性能評価の基本方針とすることで，システムに求められる十分な精度の評価を効率的に実施し工数を抑えることを図った．

【性能を考慮した開発手法の標準化】

業務アプリに，共通ライブラリと業務アプリのスケルトンを提供することで，下記の向上を図った．

・処理の均一化による安定した性能の確保
・パターンを縮小化しテスト範囲を標準化

【性能評価の標準化】

性能評価方法の標準化を行い，手順書を提示することで下記の向上を図った．

・アプリ開発者で共通のテストを実施して頂き一定基準の性能を確保
・手順書を提示することで，テストをスムーズに実施
・机上試算による性能検証を行うことで早い工程での性能問題を検出し，手戻りを抑える

上記基本方針を基に行った性能評価の実施内容について次に記述する．また，性能評価を進める中で平行して検討した，性能面から見たシステムの改善提案についても記述する．

２．２　性能評価の実施

２．２．１　性能を考慮した開発手法の標準化

本システムにおいては，大きく基盤部分と業務アプリ部分に分け開発を行った（「図 2-2-1-1 部品化による提供部分」を参照）．基盤部分の役割としては大きく２つあり，ライブラリの提供とスケルトンの提供である．

ライブラリの提供では，業務アプリが直接ミドルウェアを呼び出すことをせず，用意した共通ライブラリを呼び出すことで，複数の業務取引について１トランザクションで処理することを制限したり，１回の通信データ量の上限とするなどの制限を設けた．

また，スケルトンの提供では，業務アプリ部分の共通処理部分について雛形を用意し，各アプリ開発グループはその雛形を基に開発を行うこととした．

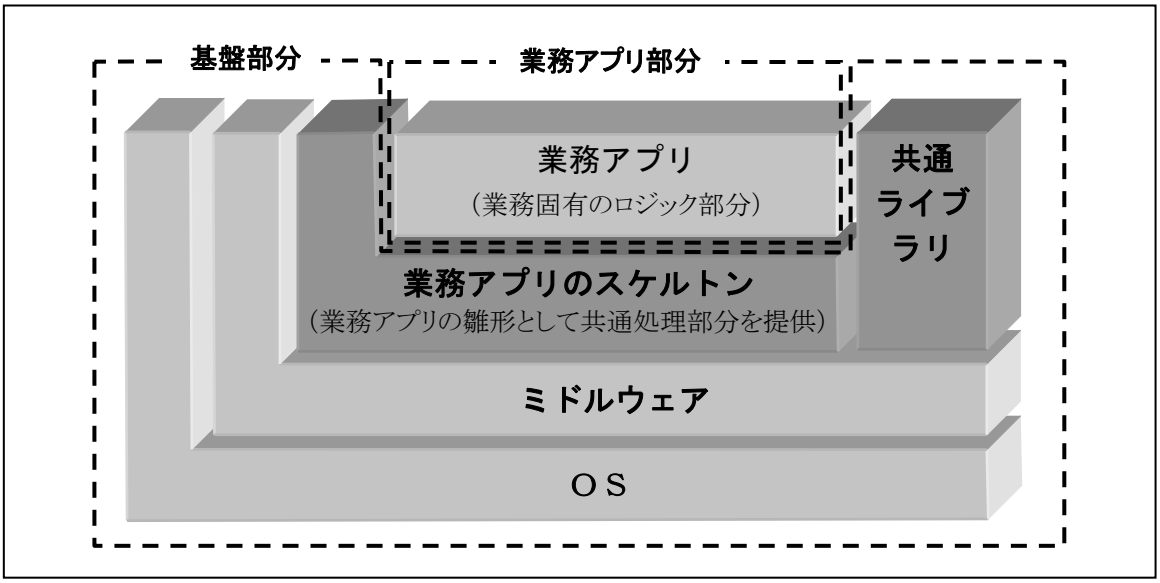

図 2-2-1-1 部品化による提供部分

業務アプリが使用するライブラリ・スケルトンを部品として提供することで，業務ロジックの作り込み部分を最小化した．

(1)　実施内容

性能評価の実施については，大きく下記「図 2-2-1-2３段階の性能評価」に示す３段階に分けテストを行った.

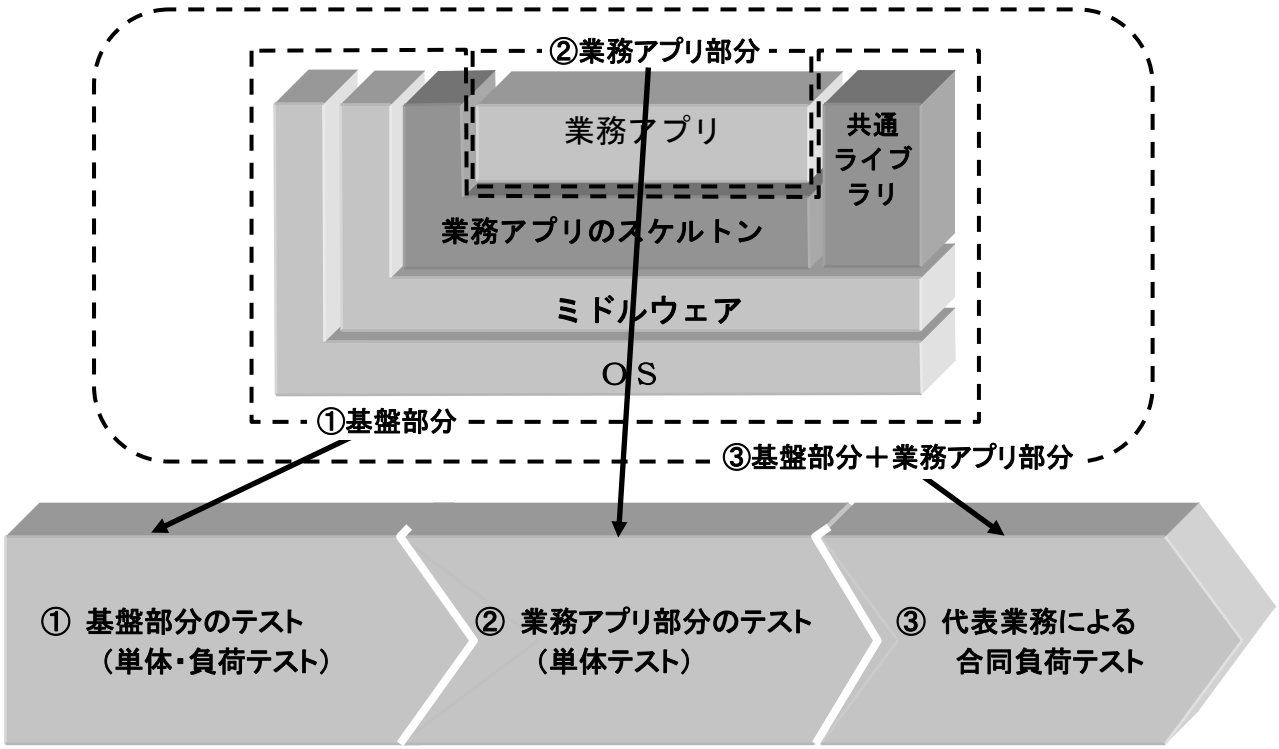

図 2-2-1-2　３段階の性能評価

①　基盤部分のテスト

先行開発業務アプリ（業務アプリの開発手法の確立のため，１業務を先行して開発）とテスト用業務アプリ（実業務で使用する共通部品を使用することで，実業務に近い性能検証が実施可能）をモデル業務として，先行して単体・負荷テストを実施し，基盤部分の性能を確認.

②　業務アプリ部分のテスト

確認が完了した基盤上で，各業務アプリの単体テストを実施し，それぞれの業務アプリに大きな性能劣化が発生していないことを確認.

③　代表業務による合同負荷テスト

使用頻度の高い業務を中心に代表業務を選定し，業務アプリ部分と基盤部分を合わせた負荷テストを行い，より精度を高めたシステム全体を通しての最終的な確認を実施.

(2)　効果

業務ロジックの作り込み部分を最小化することで開発・テスト工数を抑えるとともに，性能評価についても，下記の点について大きな効果を得ることができた．

・ 安定した性能を確保

ライブラリとして提供することで処理方式を制限し，開発者の違いによる性能のバラつきを抑えることができた．また，スケルトンを提供することで，更にコーディング時の開発者による違いを抑えることができ．これらにより，システム全体として安定した性能が得られた．

・ テスト範囲・パターンの縮小化

およそ500画面に上る業務固有ロジック部分は各アプリ開発者が性能評価を実施し，基盤部分は基盤開発グループが性能評価を実施することで，評価する範囲・パターンを大幅に絞り込むことができた．また，共通部分となる基盤を集中的にテストすることで，システム全体としては精度の高い評価を実施することができた．

(3)　実施時の問題とその対応

④ モデル業務を使用しテストすることで，レスポンスを確保するためには，画面項目数を数百項目に抑えることや，印刷形式を限定しなければならないことが判明した．しかし，これらの問題を業務アプリの開発規約にフィードバックすることで，全体的な業務アプリ開発へ活かすことができた．また，テスト時に発生した問題を元に次項に示す評価手順の改善にも繋げることができた．

⑤ 先行して実施する基盤部分のテストについてはモデル業務を使用するため，モデル業務の選定によっては評価結果がブレる可能性がある．今回はモデル業務に重めの処理をさせ，キャパシティに余裕のある基盤を確保した．そして合同負荷テストにより，ブレが用意したキャパシティ以内に収まっていることを最終確認した．

⑥ 開発工程の時期が異なる業務が存在する場合，別途テストを実施する必要が発生する．本システムにおいても，一部業務については，合同負荷テストとは別に負荷テストを別途実施した．

２．２．２　性能評価方法の標準化

部品化により業務アプリ部分のテスト範囲の最小化を図ったが，およそ 500 画面に上る業務が存在し，多くのアプリ開発グループに別れている．更にサブシステムも複数存在することから，業務内容の全体把握は困難であり，各アプリ開発者に評価を実施して頂く必要があった．

しかし，性能評価を実施した際に評価方法がバラバラになることが予想されることから，性能要件や各工程での評価手順を手順書として明記し，各アプリ開発者へ提示することで，評価方法を標準化し，一定の性能を確保することとした．

また，手順書には，IT 工程の単体性能測定の前に，UI～PT 工程で机上試算によるレスポンス評価を加えることとした．机上試算による評価を加えることで，早い工程で性能問題を検出し手戻りを抑えることとした．

(1) 実施内容

各アプリ開発者に性能評価を実施して頂くに当たり，標準化した手順として「図 2-2-2-1 性能評価手順書評価内容」に示す内容を提示し，工程別に３段階に分けテストを行って頂いた．

工程	評価内容
UI～PT	下記項目を記入することで端末レスポンスを机上算出するExcelのマクロを使用した試算シートとその使用方法を提示し，端末レスポンスを評価して頂いた（机上算出については，基盤部分の実測結果と平均的なモデル業務を使用して測定した値を基礎値として算出した）． ・１画面の項目数 ・データ通信量 ・サーバ業務アプリステップ数 ・ホスト処理時間（移行対象でないため現行システムの処理時間を記入） ※ 机上試算に使用したExcelの試算シートについては「図2-2-2-2 端末レスポンス試算シート」にイメージを記載.
IT	タイムスタンプを用いた測定方法・測定に使用するツールの使用方法等を記載し，各アプリ開発者にて単体テストによるレスポンスの実測とリソース使用量の測定を行って頂いた． また，ITのテスト環境では本番環境と機器構成・ネットワーク等が異なる環境であり，測定条件を机上試算と組み合わせ，端末レスポンスの評価を評価頂いた．
ST	代表業務を多重実行した際のレスポンス・リソース評価について指針を記載. ※ 負荷テストは手順化せず，作業要員を集め自チーム内で責任をもって負荷がけを行いテストを実施した.

図 2-2-2-1 性能評価手順書評価内容

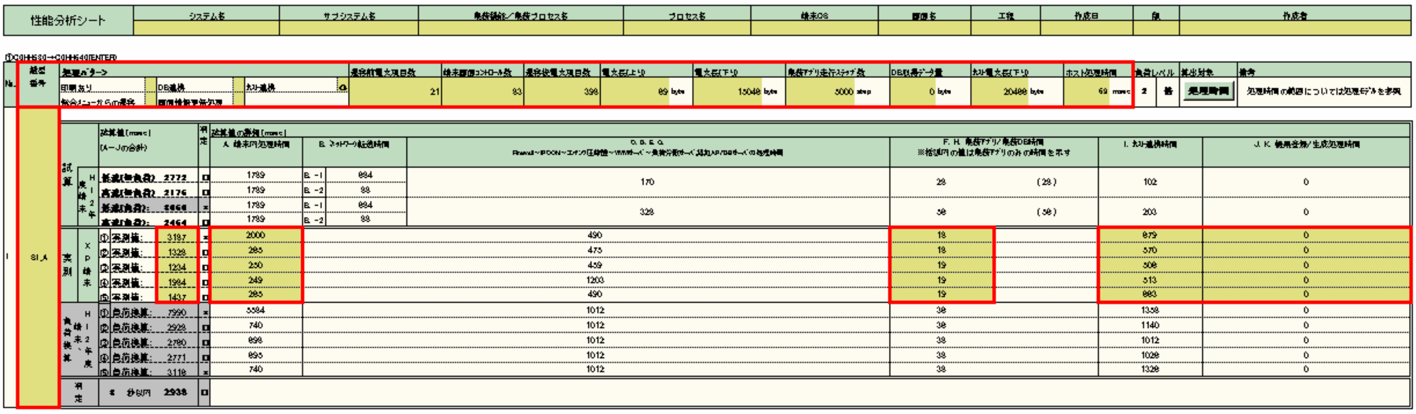

図 2-2-2-2 端末レスポンス試算シート

なお，UI～PT 工程，IT 工程においては机上試算を用いているが，本システムが部品化により共通部品を使用していることで，安定した性能により机上試算による試算が可能となったと考えており，共通部品を使用していない場合は，机上試算による端末レスポンスの試算は難しいと考えている．

(2) 効果

早い工程で性能評価を実施し手戻りを少なくし，業務アプリ改善を実施することが可能となった．また，工程が進むごとに精度を高くすることで，最終的に求められる精度の性能評価を実施することができた．

また，机上試算のシートを各アプリ開発者が記入することで，何を改善することが有効であるかを認識して頂くことができた．

(3) 実施時の問題とその対応

① 各アプリ開発グループが個別に性能問題に取り組むことは技術的に難しく，各グループのテスト結果をまとめ，分析することが必要となった．

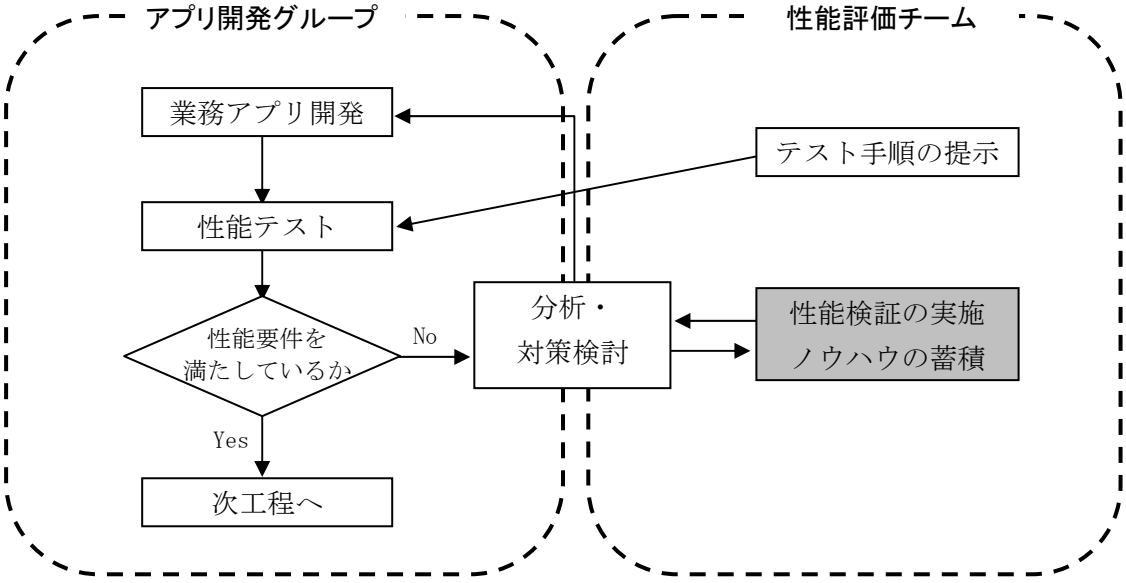

図 2-2-2-3 性能評価の流れ

そこで，各グループのテスト結果を集め，性能検証を実施することによりノウハウを蓄積した専門チームを設置した（「図 2-2-2-3 性能評価の流れ」を参照）．性能問題が発生した際は，アプリ開発グループと専門チームで協議を行い，問題の分析・対策検討を実施することで問題を乗り切ることができた．

② 机上試算を行う場合，平均的なモデルを用いるため，業務アプリの作りによっては試算値との乖離が大きくなる場合が発生した．特に端末内処理時間については端末スペックが低いこともあり，大きい場合は数秒程度のプレが発生してしまった（サーバ上の業務アプリについては業務 DB へのアクセスがほとんど存在しないことから，業務ロジックによる違いは 100 ミリ秒程度に収まった）．

２．２．３　性能面からのシステム改善の検討

性能評価を実施する場合，システム全体を通した性能を考慮する必要があるが，システムが大規模になり，各グループが性能評価を検討すると，全体を見渡した性能改善は難しくなる．そこで，前項までに記述した基本方針以外に，性能評価を実施して行く中で，システム全体を通した性能について改善点が無いかについても検討を行っておりいくつかの改善を提案・実施している．下記に例を２つ記載する．

(1) 画面の集約

トランザクションを制御するミドルウェア（Interstage）では，主にトランザクション件数に比例してリソース使用量が増減する傾向があった．そこで，トランザクション件数を減少させることで，それに比例し，特に WWW 層・AP 層のサーバのキャパシティを抑制することが可能となる．また，これまでのエミュレータを使用したユーザインターフェースから Web 化を実施したことで，１画面に表示可能な項目を増やすことが可能となった．

そこで，複数画面を１画面へ集約し，１業務を行う際の画面遷移回数を減らし，クライアントとサーバ間のトランザクション件数の抑制を図り，キャパシティを抑えることを提案した．更に今回のシステムではリッチクライアントの機能を有していることから，業務によっては複数画面をタブで構成とすることでも，トランザクション件数の抑制することを提案した．適用は各アプリ開発者の開発工数を踏まえ実施された．

(2) 拡張性の確保

UI 工程では帳票サーバは AP/DB の８台構成を予定しており，拡張性については，スケールアウト可能な構成としていた．しかし要件が整理されるに従い，帳票系の業務量が増加することとなり，帳票サーバの能力を増加させるため，最終的に数十台のサーバが必要となった．

AP/DB 構成でスケールアウトした場合，DB の台数も増加する．DB 部分については，OracleRAC 構成を用いていることもあり，特に保守時の運用費用の増加が懸念される．また，性能評価を実施する中で，帳票サーバのリソース使用の大部分は AP 処理であることが判明したことから，AP と DB サーバを分割してサーバに配置し，スケールアウトした場合でも DB の台数を数台とすることで，全体の保守費用の抑制を提案し，構成の変更を行った．

２．３　本稼動を受けて

今回性能評価を実施したシステムについては，平成 18 年４月に本稼動を開始しており，４月を終えた現時点まででシステム全体が停止することはおきていない（最も業務量が多いピーク日は４月の稼動当初）．概ね順調な稼動を迎えており，レスポンスが遅いとのユーザからの報告も発生していない状況である．

機器のキャパシティについては，余裕を持った機器構成としたこと，画面集約が図られたことによりトランザクションが大幅に減少しており，CPU 使用率はサーバによって異なるが数十パーセント程度に収まっている．余裕があるキャパシティについては，当初から次年度についても機器の増強が予定されおり，次年度の機器増設分へ転用できることから，お客様にも了承を頂いている．

３．まとめ

３．１　今回の取組みと今後の課題について

基本方針に従い性能評価を実施したことで，当初の予定した効果は十分に得られた．しかし，課題も残っており，次年度の開発に向けて改善を図って行きたい．

① 部品化について

今回の大規模基幹システムでは，性能評価を実施する上でも部品化の実施は，作業の効率化・性能の安定に非常において有効であった．今後とも処理の共通化を図りたいと考えている．

② 手順書による性能評価について

手順書作成が遅れたことで机上試算による評価を一部実施することが出来なかったが，当初予定した全画面の単体性能テストを完了することができ，一定の性能を確保することができた．机上試算による性能評価については，一部実測結果と大きな乖離が発生していることから，次年度に予定されている業務移行に向けて，机上試算方法の見直しを図りたい．

③ 性能面からのシステム改善検討について

今回は画面集約による改善により，キャパシティを大きく抑えることができた．性能面からのシステム改善の検討は有効であり，システム全体を見渡した性能改善を提案していきたいと考えている．

３．１．１　その他の問題事例

(1) 障害発生によるテストの遅延

今回の性能評価で，最も作業が滞ってしまったのは，基盤部分の単体・負荷テストの実施であった．基盤部分の性能テストは業務アプリが完成していない状態で先行して実施することから，システム全体を通したトランザクションの流れのテストは十分に実施されていなかった．そのため，先行開発業務とテスト用業務アプリを使用して行った性能テストでは，グループ間の連携不足による下記のような障害が多数発生し，性能測定の行えない状況が何度も発生した．

・ 認証処理で失敗し目的の処理が行えない．
　→ テスト用の負荷がけツールから送信する端末情報とサーバ上の端末情報が矛盾しており，認証処理が正常に行えずエラーが発生していた．
・トランザクションが予定していたモジュールで処理されず，性能測定を行うことができなかった．
　→ 担当者が処理モジュールを決定する際の判定条件を誤って認識していた．
・ ホストとの連携処理でエラーが発生．
　→ ホストの必要なサービスが起動されていなかった．
etc...

性能テストは環境の占有が必要ということもあり，夜間・休日のテストがメインとなる．そのためもあり，障害発生時に担当者の不在が多く，調査・対処が持ち越しになり，テストが進まない状況が続くことになった．テスト環境の調整には時間を要することから，障害の長期化はテスト実施に致命的な遅れとなり，性能検証は度々スケジュールの見直しを行うこととなってしまった．

(2) テスト体制の見直し

開発規模が大きく開発担当者が多岐に渡ることから，最初テストはテスト対象となる開発担当者とともにテストの実施を行っていたが，前項のような問題が多発するため，各グループより性能テストの窓口となる人を割り当てて頂き，ワーキンググループとして性能テストに当たった．

各窓口に対応頂くことで，テスト準備もスムーズに行うことができ，問題発生時の調査，問題部分の切り分けについて迅速に行うことが可能となった．障害対処については，担当者が不在のため持ち越しになったものの，即時対応が可能な障害の多くは窓口の方に対応頂きテストを進めることができた．

ワーキンググループの設置は，各グループから協力して頂かなければならないが，テストの全体的な進捗は大きく改善しており，今回の性能テストを進める上では欠かすことができなかった．

３．２　大規模システムの性能評価の実施

今回の性能評価の実施にあたっては，各グループがバラバラに行動していては十分に検証することができなかった．大規模システムで性能評価を行うためには，システム全体を見渡し，性能に関する開発の標準化，評価の標準化を実施できる，下記「図 3-2-1 性能評価の体制」に示すような体制が必要と考えられる．

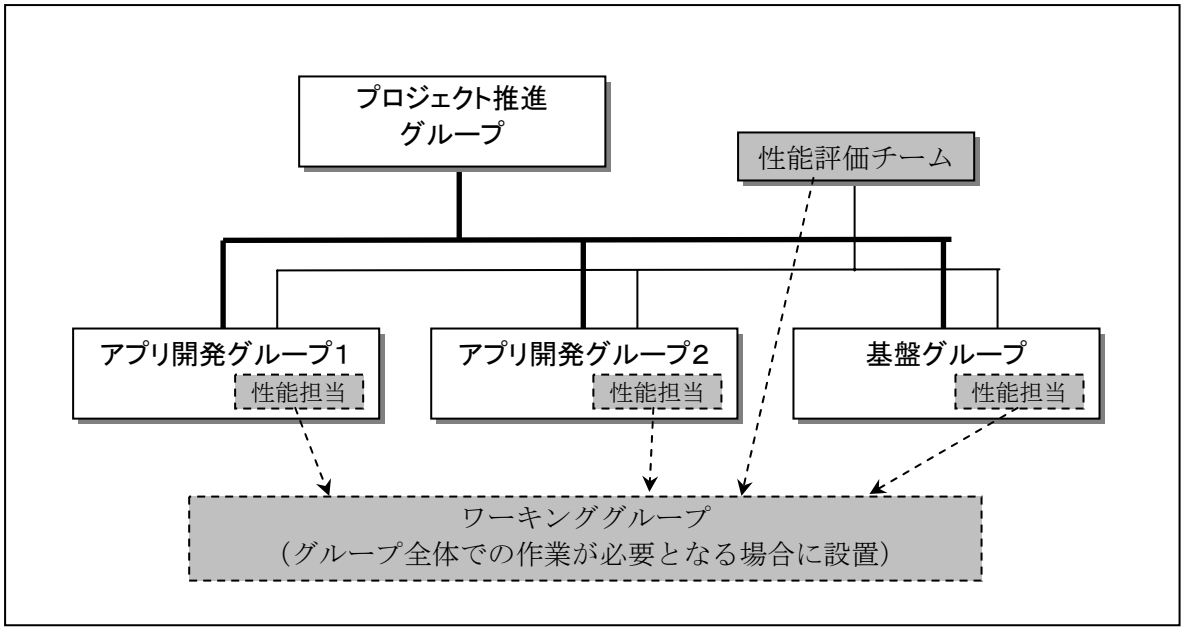

図 3-2-1 性能評価の体制

性能評価のポイントについては開発標準などで定義されている場合もあるが，実際に大規模システムで各グループが一定レベルの性能を確保することは困難であり，性能評価を専門としたチームが必要である．

また，負荷テスト等で各グループの協力が必要となる場合は，性能評価のみを目的としたワーキンググループを作り，各グループの性能に関する有識者を集めることで，より体制を強化して実施を図る必要がある．

この体制を整えたことで，今回のプロジェクトは成功に至った．

最後に，性能評価を行う上で，留意しておいた方が良いと考えるポイントについて記載する．

① システム全体を見て性能評価を行う

レスポンスタイムが要件を満たしていない場合，チューニングにより短縮を図ることになるが，何処をチューニングするのが最も効果的であるかは，システム全体を考えて実施する必要がある．また，今回の画面集約による性能改善等は，アプリ開発グループ，基盤グループが共同で取り組む必要があった．

性能改善を検討する際は，グループに捕らわれないよう，システム全体の利益を考え改善・提案ができるよう，できる限りシステム全体を監視するグループが性能改善についても検討を実施すべきである．

② 重要なポイントを中心に評価を実施する

性能評価を実施する際，すべての処理について厳密に評価することは，開発期間を考慮すると難しい．そこで性能評価は，どのポイントを中心にテストを実施するか，事前に検討する必要があると考えている

性能要件が厳しい業務や共通処理部分，使用頻度の高い業務等を中心に性能評価を実施し，重要度が低いものについては，多少精度が落ちても問題が無いよう安全係数をかける等の余裕を持った評価を行い，システム全体として十分な性能を確保すべきである．

③ 過去事例の活用

性能評価の手法はシステムの特性に合わせ毎回変更しなければならない．今回実施した標準化の手法についても，対象システムの規模や開発グループ構成によっては有効に働かなくなる．そこで，類似したシステムの事例を参考とすることで，より効率的な評価が可能になる．また，類似したシステムで発生した性能問題は，同様に発生する可能性が高いことから，過去事例は確認しておくべきである．

今回の性能評価に当たっても，同システムの過去の評価手法・性能問題を参考とし，事業部からは過去のトラブル事例，他システムの評価方法等をご教授頂いた．それにより，多くの性能問題を抽出することができ，効率的にテストを進めることができた．

今回のような大規模基幹システムの性能評価という，信頼性の確保として重要な作業に携わることができ，得難い経験をさせて頂くことができた．この経験を通じて，より確実に性能を満足できるシステムを顧客に提供できると考えている．

用語説明

AIM (Advanced Information Manager)

富士通(株)が開発した総合的なオンラインデータベースシステム．オンラインネットワーク機能，データベース機能，トランザクション管理機能などを結合したソフトウェア製品．

Biz/Browser

アクシスソフト(株)が開発したリッチクライアント業務用 Web ブラウザ．キーボードフォーカス，入力フィールドごとの入力制御，数値のカンマ表示など，業務システム向けの機能をサポートしている．また，画面情報のキャッシュ化により高速な画面表示レスポンスを提供する．

Interstage

富士通(株)が開発したシステム構築用のミドルウェアの総称．基幹系システム構築・運用のために用意された製品群であり，汎用機水準の高信頼，高性能技術を Windows，Linux，Solaris 系システムの構築技術に反映させ，安定したレスポンスや TCO 削減を提供．

Oracle RAC (Real Application Clusters)

クラスタ内の全ノードでバッファ・キャッシュを共有し，高速クラスタ間相互接続によるデータの同期「キャッシュ・フュージョン」することで，高可用性とリニアなスケーラビリティを両立させたクラスタ・アーキテクチャ．

UI、PT、IT、ST

各開発プロセスの略称を示しており，それぞれの意味は下記の通り．

UI（ユーザインタフェース設計工程）
- ・業務システム仕様（プロセス機能，データ構造，画面，帳票）を設計．
- ・システム方式設計の詳細化，および運用・移行方法を設計．
- ・全体テスト計画を立案．

PT（プログラムテスト）
- ・プログラムテスト仕様に従ってテストを実施し，品質を検証．

IT（結合テスト）
- ・プログラムを結合して，プロセス単位のテストを実施し，品質を検証．
- ・外部システムとのインタフェースを含むすべてのプロセス間のインタフェーステストを実施．
- ・必要に応じて結合レベルや処理形態例（オン/バッチ）でIT工程を分割．

ST（システムテスト）
- ・実機上で業務システム機能をテスト．
- ・性能，信頼性，運用性，セキュリティなど，システム全体の検証を実施．