大規模開発プロジェクトにおける性能テストの実践方法

～失敗しない効率的な負荷テスト実施のコツ～

2010年1月29日

TIS株式会社
技術本部　先端技術センター
東條　義幸

© 2010 TIS Inc. All rights reserved.

TIS株式会社　会社概要

社名	TIS株式会社（TIS Inc.）
設立	昭和46年（1971年）4月28日
資本金	231億円（平成21年3月31日現在）
代表者	代表取締役社長　藤宮宏章
従業員	2,844名（平成21年3月31日現在）
主要取引銀行	三菱東京UFJ銀行、三菱UFJ信託銀行
本社所在地	東京、大阪

ITホールディンググループの中核として「最適」なシステムを提供します。

© 2010 TIS Inc. All rights reserved.

AGENDA

- 大規模プロジェクトの特徴
- プロジェクトでの課題
- 性能確保のための作業
- 重点対策
- 今後の予定
- まとめ

© 2010 TIS Inc. All rights reserved.

大規模プロジェクトの特徴

- プロジェクト全体
  - 今回の対象は某社向け基幹システムの更改
  - 構想2年、開発4年
  - 全体で数千人月規模

    ピーク時には数社、数百人が複数の拠点にて同時作業
  - 開発はコンポーネント化を推進
  - Web画面、外部システム連携、バッチ処理が同時稼動

- 性能テスト観点
  - Web画面に限っても3000画面以上
  - 性能目標値を満たさない場合はC/O不可
  - 定められたテスト期間は厳守

© 2010 TIS Inc. All rights reserved.

プロジェクトでの課題

システム構成、画面構成を全て把握している人はいない

担当者を十分数アサインするのは難しい

3000画面をどうやってテストする？

人の作ったコンポーネントは良く分からない

期間厳守！！

性能要件厳守！！

© 2010 TIS Inc. All rights reserved.

性能確保のための作業

・性能要件の設定
- レスポンスタイム
- スループット
- ユーザ数

・性能テスト実施方法の検討
- 体制、大日程、など

・環境設計
- 開発環境／本番環境／テスト環境

・テストツールの選定（自作含む）

・コンポーネント単体性能検証

・機能単位性能テスト
- 画面単体での性能テスト

・負荷テスト
- Web画面として最大負荷日の処理を想定した負荷テスト
- システム全体で最大負荷日の処理を想定した負荷テスト

© 2010 TIS Inc. All rights reserved.

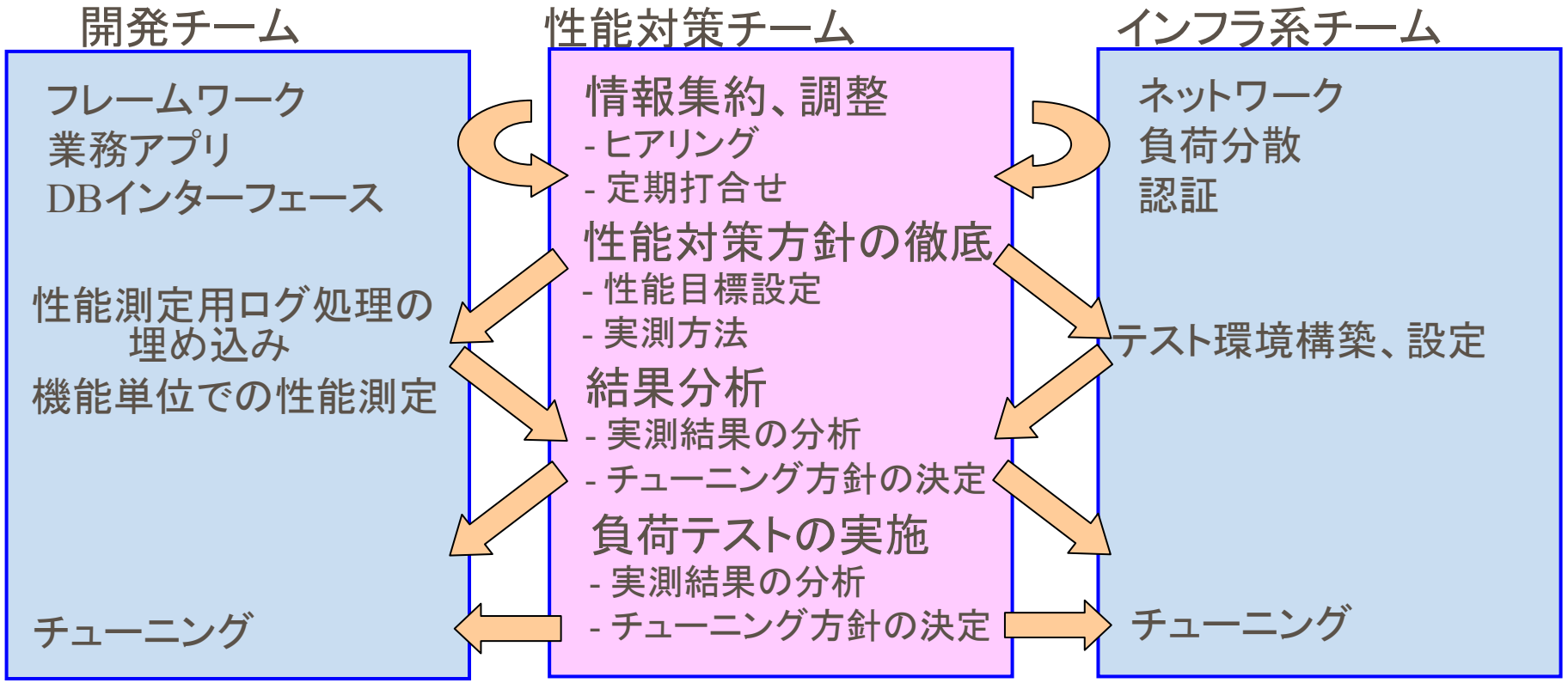
重点対策

1. 体制作り
   ～孤独感からの脱却～
2. 業務分析による重点対象の絞込み
   ～力点の調節～
3. 性能目標値（レスポンス）の設定
   ～ゴールはどこ？～
4. 性能問題の早期発見
   ～不発弾を取り除く～
5. 本番運用を想定した負荷テストの実施
   ～ツールは考えてはくれない～

© 2010 TIS Inc. All rights reserved.

重点対策1：体制作り

- 性能対策専任チームによる情報集約／共有と分析の実施



効果

1. プロジェクト全体で統一した方針、対応が取れる
2. 定期的な情報交換、共有により、作業スケジュールをたてやすくなる
3. 早期に良好な関係を作っておくと、開発終盤のシビアな要求も対応して貰いやすい

© 2010 TIS Inc. All rights reserved.

重点対策2：業務分析による重点対象の絞込み

限られた期間、リソースで、全てを同じ優先度で対策を講じることは不可能

対策

画面別に、過去の利用実績データを、新システム画面へマッピングし、以下の観点で重点対策対象を絞り込む

- 過去実績のアクセス頻度がTOP40に入る画面
- エンドユーザより特に希望のあった画面のうち、アクセス頻度が上位の画面

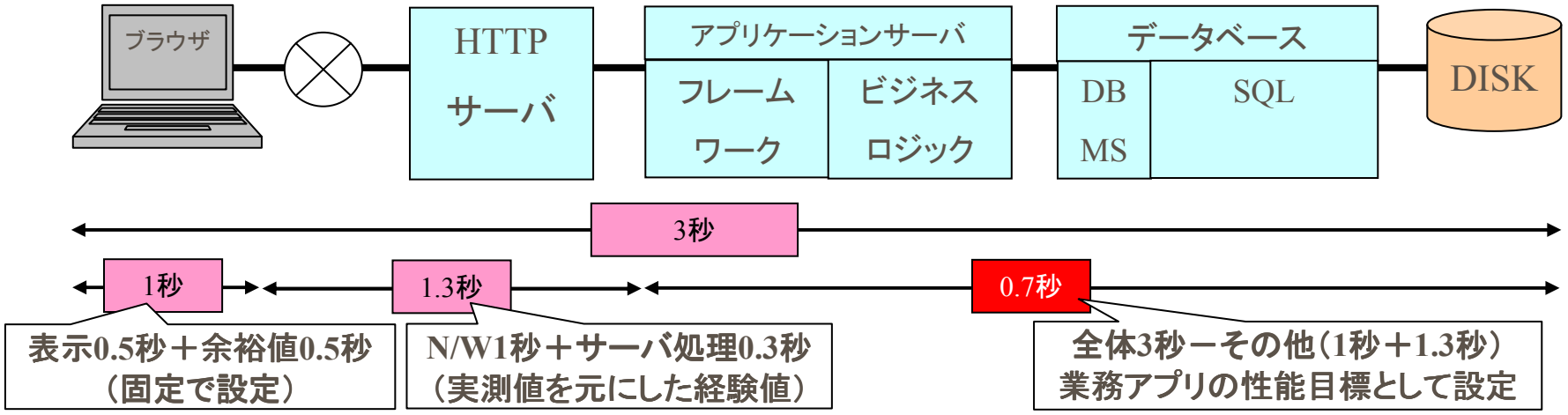
効果

1. この絞り込み条件で、想定画面アクセス量の80%をカバーでき、少ないリソースで重点集中の効率的なテスト実施ができた

© 2010 TIS Inc. All rights reserved.

重点対策3：性能目標値（レスポンス）の設定

- 初期設定時（要件定義時）には、現行システムのレスポンスを参考に、クライアントにて一律3秒以内（95パーセンタイル値）と設定
- 構成要素毎に目標値を個別設定



- 明らかに目標達成が不可能なものについては、後工程にて（仕様FIXまでは）必要に応じて個別に目標値を設定

効果

1．レイヤ毎の定量値を明確に示すことで、ゴールがはっきり見えた

© 2010 TIS Inc. All rights reserved.

重点対策4：性能問題の早期発見

（1） 机上での性能検証

設計がある程度進んだ段階で、机上検証を実施

・前提

・最も頻度の高いデータ、検索パターンを想定する
・ハード依存の基本データはベンダ調整の上決定
・問題発生が多く、チューニング効果の高いDBアクセスに着目して検証

①設計内容より、画面別に処理内容を見積もる

・SQL発行回数
　SELECT/FETCH/INSERT/UPDATE/DELETE
・データヒット件数

②処理内容を元に性

・SQL処理時間＝SQL発行回数×SQL平均単価×余裕率＋競合コスト
・DISK I/O時間＝（データヒット件数より算出したI/O回数）×I/O単価
　※索引はキャッシュあり、データはキャッシュなしと想
・全体処理時間＝SQL処理時間+DISK I/O時間

プロトタイプ実測値より
0.46ms

固定値
1.5

SQL種別により
0.02ms～0.05ms
×SQL回数

ベンダ想定値より
10ms

© 2010 TIS Inc. All rights reserved.

重点対策4：性能問題の早期発見

③性能目標値を満たさない画面に対する対策

- DB設計の見直し（索引追加、など）
- アプリロジックの変更
- 業務仕様変更
- 性能目標値の再設定

※性能目標値を満たす目処がたつまで、先の工程には進めない

効果

開発担当者からは苦情多数  しかし、

1. 約2000パターン中の約1/4に対して、改善指摘を実施
   明らかに設計に問題があるものについて、製造に本格着手する前に
   対策を講じることができた
2. 設計段階から、性能の意識を植え付けることができた

© 2010 TIS Inc. All rights reserved.

重点対策4：性能問題の早期発見

（2） 画面単体での性能検証

画面が完成した段階で、画面単体の性能実測を実施

- 処理パターン
  - ・アクセス頻度最大パターン（標準的なデータ量、ロジックの重さ）
  - ・処理の重いパターン（最大データ量、最も重いロジック）
- 性能測定箇所
  - ・フレームワークにログ出力処理を埋め込み、アプリ部分の処理時間を測定
- 業務チーム、性能対策チーム共同での実測
  - ・DB標準ツールによるSQL単位の情報
  - ・フレームワークから出力されるログ情報
- 結果分析
  - ・全ての出力データを集計、分析するツールをperl+Excelにて自作
  - ・主にDBアクセス（SQL呼び出し回数、DISK I/O回数、索引使用など）に着目して分析

© 2010 TIS Inc. All rights reserved.

重点対策4：性能問題の早期発見

・性能目標値を満たさない画面に対する対策

テスト実施114パターン中、68パターンがNG

・アプリロジックの変更（45%）
・DB設計の見直し（統計情報更新、索引追加、など）（20%）
・性能目標値の再設定（5%）
※残り30%はテスト環境依存の問題であったため、対策はなし

※性能目標値を満たす目処がたつまで、先の工程には進めない

効果

1．痛みは伴うが、まだ致命傷にはならない
　ここで業務アプリの性能に関する問題をほぼ全て解消することができた

© 2010 TIS Inc. All rights reserved.

重点対策5：本番運用を想定した負荷テストの実施

- 負荷テストは本番運用に近い状況での性能確認

・通常、画面単体での実行は不可。ログインからの画面遷移により対象処理が行われる
・同じ画面でも、検索条件などにより処理の重い／軽いが存在する

・アクセス比率が同程度の画面をグループにまとめ、グループ単位で画面遷移を決定する。（テスト対象画面を7グループへ分類）
・重い処理／軽い処理のアクセス比率をヒアリングし、データ検索パターンをシナリオに盛り込む

スクリプト1			スクリプト2		
1	ログイン	-	1	ログイン	-
2	メニュー遷移	-	2	メニュー遷移	-
3	会員情報検索	30%	3	ポイント情報検索	10%
4	履歴情報検索	25%	4	売上げ情報検索	10%
5	請求情報検索	30%	5	ログアウト	
6	ログアウト	-			

使用データにより、各処理の中で、重さ比率を調整

同時実行数	
スクリプト1	400
スクリプト2	100

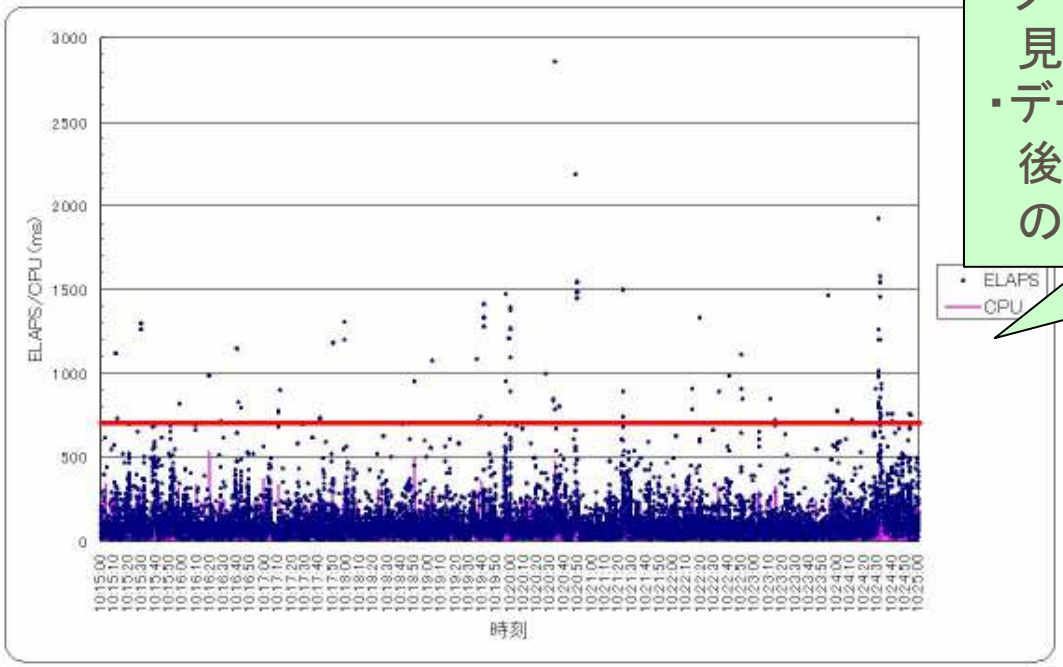
スクリプト単位の同時実行数により、画面の実行数を調整

© 2010 TIS Inc. All rights reserved.

重点対策5：本番運用を想定した負荷テストの実施

- 結果分析

  サーバ上のログを取り込み、レスポンス分布をグラフ化、集計するツールを自作（awk+Access+Excel）



・グラフ化すると、全体の傾向が見える
・データをDBに入れておくと、後から様々な解析に利用できるので便利

個々の画面毎の判定には集計データを使用

画面名称	実行数	目標以内件数	平均ﾚｽﾎﾟﾝｽms	達成比率	判定
会員情報照会	1,164	1,165	127.7	99.23%	○
履歴情報照会	:	:	:	:	:
請求情報照会	:	:	:	:	:
ポイント情報照会	:	:	:	:	:
:	:	:	:	:	

© 2010 TIS Inc. All rights reserved.

重点対策5：本番運用を想定した負荷テストの実施

実際には・・・今回の負荷テストでは、
- 画面処理自体に起因する問題の検出はほぼゼロ
- OS、ミドルウェア起因による問題の検出のみ

アプリ大改造に至る諸問題を早期に解決できたため、
致命的な問題発生を回避することができた。
⇒性能問題によるプロジェクト進捗の遅れはなし

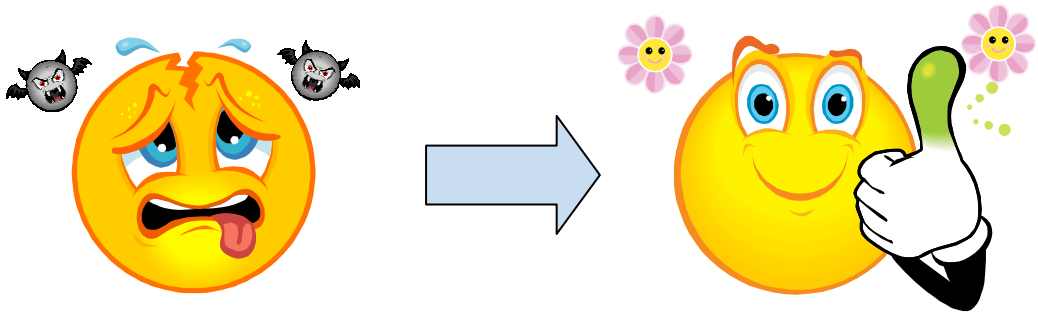
© 2010 TIS Inc. All rights reserved.

今後の展開

今回の手法を一過性のものにせず、普及、徹底させる

- ノウハウの可視化、ナレッジ化を進める
  誰もが参照できる形での情報展開

- より小規模なプロジェクトへの横展開
  全てのプロジェクトで、フルセットの対策を採ることは現実的には困難を伴うが、必須事項は実施しなければならない。



© 2010 TIS Inc. All rights reserved.

まとめ

- 性能問題解決に特効薬はない
- より早い段階から、工数を掛けてでも、メンバへの意識付け、及び、必要な対策を講じていけば、プロジェクト終盤のデスマーチを回避できる
- 実施よりも準備がカギ

  いかに十分な準備ができるかによって、スムーズな実施ができるかが決まる

  負荷テストだけを考えるのではなく、その準備（前提）として単体性能テストを実施すると考えるべき

© 2010 TIS Inc. All rights reserved.

ご清聴ありがとうございました

© 2010 TIS Inc. All rights reserved.