性能評価の定石

3L-8

ー業務システム開発における性能設計の勧めー

青木　研治
富士通株式会社

【はじめに】

システム開発において、性能評価と言う分野についての解釈は様々であり、まだ何をするべきかと言った面では統一されていない。

そこで、私の経験した大規模開発での経験をもとに、システム開発の中で、性能評価をどの様に行うべきかを述べたい。

【業務開発の問題点】

企業にとっての情報化の重要性の認識が高まるとともに、コンピュータを取り巻く技術革新によってＣＰＵ／周辺機器の性能が向上し、大規模ネットワーク構築や大容量データの集中管理が可能になってきており、ＳＩＳ (Strategic Information System) やＣＩＭ (Computer Integrated Manufucturing)といった複雑に情報が入り組んだ業務開発が行われるようになってきている。

しかし、業務開発量と比較して開発作業を行うＳＥやプログラマーが不足しているため、開発の効率化に目が向けられ、結果として出来上がったものに対する効率／性能については二の次になっている場合が多い。

そのため、テスト段階及び本稼働になってから大きな問題となるケースが多々見受けられる。

この理由として、①業務要件のみを中心としたシステム設計　②性能面からの制約に対する情報不足　③効率／性能に対する無関心　などがあると考えられる。

【性能評価とは】

言葉の定義として、『性能評価』とは一体なにを示すのか自体あまり明確では無いが、この論文では『あるシステム環境のなかで、システムの提供できる資源を効率良く使用して、ユーザの求める機能を実現するための方法』といった意味で使用する。

【性能評価のながれ】

性能評価という作業の中には　①性能見積もり　②稼働分析　③システムチューニング　④キャパシティプランニングと言った様々な要素が含まれているが、それぞれの作業が業務の開発スケジュールの中でどの様な役割を持つかは明確ではないので、業務開発に対する性能評価のフェイズを考え、どの様なことをすべきかの整理を行った。

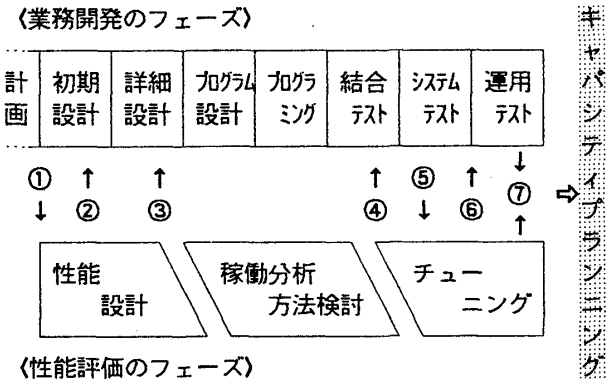

図１．業務開発と性能評価

１．性能評価の４つのフェーズ

(1)性能設計

開発される業務の要件と対象となるユーザ資源（データ量・トランザクション量等）をベースに必要となるシステムの資源を積み上げ、使用されるアプリケーションの機能を明確にする。（①②）業務設計の性能面からの指針を明確にすると共に、設計された業務（初期設計／詳細設計）に対して性能面からのレビューを行う。（③）

(2)稼働分析方法検討

開発される業務システムがどの様な状態で稼働しているかを分析する方法と、個別の業務プログラム（オンライン／バッチ）が資源をどの様に使用しているかを把握する方法を検討する。

必要があれば解析ＴＯＯＬを開発する。

(3)チューニング

プログラミングが終了し、テストが始まったフェーズからプログラム・ＪＯＢの稼働状態を把握し、必要に応じてシステム全体の稼働状態と問題となるプログラム・ＪＯＢの解析結果から必要となるチューニングを行う。（④⑤⑥）

業務プログラム・ＪＯＢの稼働状況とシステム全体の負荷状況から、性能面から見て本稼働に耐えるかどうかを評価する。（⑦）

(4)キャパシティプランニング

稼働中の資源使用状況の推移を把握し、どの時点でどのような資源強化が必要かを検討する。

The method of development for application system without performance troubles
Kenji Aoki
FUJITSU LIMITED

2．性能評価の重点ポイント

上記の4つのフェーズのうち、(2)の稼働分析方法の検討と、(3)のチューニングは様々なプロジェクトで実施されているが、どちらかと言えば出来上がった『物』を上手く動かすための手段であり、性能問題を解決するための方策ではない。

最も効率良く性能問題を解決する方法は、初期の設計時点で性能に対する目標と方法論を決めておくことである。

その具体的な方法としては、設計者に対して性能面から見た指針を与えることであり、内容としては以下の様なことが必要である。

①初期設計の段階で、性能面から考慮する重点ポイント。（データの量・構造の分析など）

②設計段階で行う性能目標実現のためのレビューでのチェックポイント。
（ＤＢのアクセス経路・目標レスポンスなど）

③プログラマーに対し、分かり易い形で所定の性能目標を提示する方法。
（ＤＡＳＤアクセス回数の制限など）

【性能指針の作成】

1．性能指針作成の手順

初期の設計段階でどれだけ性能面からの情報を提示できるかが最も重要である。

本来はプランニングの段階で情報を分析しておくことが必要なのかも知れないが、業務の要件がまとまった段階から使用するハード・ソフトを検討することが多く、性能面の情報を分析することは後手に回りがちである。

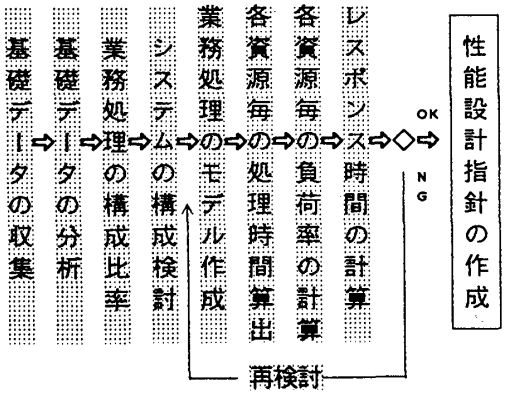

図2．性能指針の作成手順

性能指針を作成するためには、先ず事前にシステムを構成する様々な資源の特性を把握すべきであり、作業のフローとしては図2の様になる。

2．性能指針のポイント

業務開発の初期の段階で性能設計のための基礎となるデータを揃えることは容易でない。

しかし、少なくとも業務データの収集・分析と、以下に示す4点のポイントについて、検討すべきである。

(1)データ属性の分析（質・量）

データの発生頻度や蓄積方法・必要となるデータの容量などを考慮すると、属性として

①トランザクション型：日々発生するトランザクションに依存し発生の波が最も大きい。ピークを想定して設計する必要がある。

②マスター型：更新／追加の頻度はほぼ決まっており容量としてもあまり変化はない。

③履歴型：日々のトランザクションに比例してデータが発生し、蓄積が必要なため最も容量を必要とする。使用頻度は多くない。

の3パターンに分類できるので、属性に応じたデータ編成と効率面からの評価を行う。

(2)資源能力の具体的提示

プログラムのロジックは業務の要件により千差万別であり一概に規制できないが、1回のトランザクションで発行するｉ／ｏの回数の制約を明確にする事などにより対処が容易になる。
（15 i/o／1 TR、70 ms ／1 i/o など）

(3)プロトタイピングの実施

業務設計を行うＳＥやプログラマーに対して分かり易い形で問題の解決方法を示すためには、業務プロトタイピングを実施し、以下の事項を明らかにする。

①基本ハード・ソフトに関する特性および機能

②業務処理のパターンと使用しなければならない機能（運用面からの機能も含む）の検証

その業務プロトタイピングの結果を基礎に業務開発の設計段階で性能設計指針を提示出来ない場合、性能面からの手戻りが発生する。

(4)性能見積りワークシート

プログラム設計が完了すると、プログラムがどの様な命令を発行しているかが明確になる。

そのフェーズまでに、ロジック／マクロ命令よりどの位の資源を使用するか（ダイナミックステップ・ｉ／ｏ命令の発行回数など）を容易に見積れるワークシートを提供すべきである。

これにより、異常なレスポンスが予想されるものが早期に把握でき、対処方法を検討することができる。

【終わりに】

業務開発の早い時点で性能指針を明確にすることにより未然に問題発生を防止できると考える。

ぜひ初期設計段階からの性能設計の実施を試行してみて頂きたい。