システムライフサイクルと性能管理

住友邦男

株式会社　アイ・アイ・エム

システムライフサイクルの各フェーズにおいて性能管理のもつ役割は小さくない。システム企画段階からＳＬＡとして性能を定義するユーザも増えつつある。非機能化要件として性能管理が取り上げられる機会が増え、システムに携わる関係者の関心も高くなりつつある。一方、システム運用において「性能管理の必要性に疑問を投げかける」、「性能管理の工数を確保できない」と否定的な声があるのも現実である。ここでは本番システムカットオーバー以降のシステムにおける性能管理の事例を通じ、システムのライフサイクルにおける性能管理を考察する。

Case study: System Life Cycle and Performance Management

Kunio Sumitomo
IIM Corporation

Performance management plays an important role on each phase of a system's life cycle. There are more users who define systems' performance using SLA at the planning phase. As it becomes a more popular topic of discussion as non-functional requirements, not only systems engineers, but also executives and end-users are beginning to realize the importance. On the other hand, systems engineers often express their negative opinion on performance management because they think its necessity is questionable or they simply have no time. Through the various case studies after systems' cutover, I will observe performance management on each phase of the system's life cycle and discuss its ideal implementation and issues to be solved.

１はじめに

「性能管理」難しい響きのある言葉である

汎用機全盛の頃は専門家の手によって稼動統計データから評価・分析・性能チューニングを行うという印象があった。また業務主管、運用主管が明確に分かれていた。これは汎用機が全業務の共通基盤であったためである。

システムのオープン化に伴い事情が変わってくる。コンピュータの運用に関わる担当者が増えた。また運用の専門家ではなく、業務主管が運用を含め責任を担っていることが多い。故に性能管理の必要性や効果に疑問を抱く業務担当者も多い。

本稿では、本番システムカットオーバ後を4つのフェーズに分け、各フェーズにおける性能管理の事例を通じてシステムのライフサイクルにおける性能管理の実情を考察する。

２システムライフサイクル

ここでは本番システムのライフサイクルを4つのフェーズ分け次のように定義付ける。

導入期
　業務システムのリリース後、エンドユーザによる使用が開始され初期トラブルが収まるまでの時期
成長期
　初期トラブルが収まりシステム運用が定常的な管理に向け日々改善している時期
安定期
　システム運用も軌道に乗り、安定運用の時期
終焉期
　業務機能要件がユーザ要求に応えられない、ハードウェアの耐用年数が過ぎ故障件数が多くなる時期

３導入期

導入期に性能管理が意識されることは少ない

3.1よくある実情

通常、システムのテスト完了後、本番環境にリリースされる。システムテスト段階において、ストレステスト、応答性能テスト、バッチ処理時間などさまざまな検証が実施され、本番環境へ移行される。しかしながら、今までの実績をみる限り順調に問題なく本番環境が稼動することは少ない実情がある。

3.2想定の甘さ

システム企画・設計・テスト段階では想定していない事象が本番環境において発生することがある。たとえば利用ユーザのシステム利用特性があげられる。ある程度、分散され利用されると予想していても、業務の都合上、予想を上回る同時利用が発生する。システムリソースの過負荷な利用が待ちをつくる。ユーザからみれば接続待ちとなる。システム切り替え直後はユーザも不慣れなこともあり、何度も何度も同じリクエストを繰り返す。これがさらにシステムの過負荷状態を進行させる。

3.3総合テストの重要性

たとえば、同時接続ユーザ数、単位時間あたりの処理トランザクション数、データ量、回線帯域など、本番環境と同じ条件で総合テストを行っておけば、想定外の性能問題は発生しない。コスト・時間・人などの制約事項があるため、実施できないことが多い。開発進捗の遅れなども大きな要因となっている。

制約事項が解決できているユーザの場合、想定の数倍の負荷をかけて総合テストを行っている。当然ながら本番を迎えても性能上の問題は発生しない。

3.4問題発生ケース１

次のようなケースがあった。単位時間当たりのトランザクション処理量のピーク時にWEBレスポンスが悪化していた。ハード、OS、DB、業務アプリケーションの総合テスト結果からみて発生しない問題であった。結局のところ結論から言えば回線帯域が設定の帯域を持っていなかったためである。10Mbpsと設定されているはずが実際は1Mbpsであった。既存回線であったため確認を行っていなかったところに原因がある。

3.5問題発生ケース２

OS、DBともバージョンアップしておりハード性能は1.5倍相当であった。休日の本番移行が無事に終了し、本番の第一日目を迎えた。午前中は特に問題はなかったが、午後の処理ピーク時間に差しかかった頃、処理遅延問題が発生した。システム状況を確認したところ、プロセッサ使用率が100%、空きメモリが枯渇、ページングが多発しスラッシング状態になり、スワップスペース使用率が90%を超え、システムダウン一歩手前の状態であった。I/O処理活動の多い業務システムであったため、I/O領域としてメモリを大量消費し、ワーキングセット領域が枯渇していた。幸いI/O領域の大きさを制限することでシステムダウンを回避できたのである。

導入期において本番環境と同条件で総合テストを実施できなかったために発生する問題は、性能問題の大半を占める。しかし導入期の性能問題は、本番環境と同条件で総合テストを実施することにより回避できるが、コスト・時間・人の制約事項のためできないのが実情である。

3.6即効性のある解決策

ここでは本番環境であるため、即効性のある対処が求められる。ハードウェアの増強、ソフトウェアのチューニングなどが考えられるが、大抵のユーザではハードウェア増強で解決を試みることが多い。これはハードウェア増強することで処理能力が確実にアップするからである。実際、プロセッサの追加やメモリの増強で、初期の性能問題を乗り切った事例は多い。ソフトウェアのチューニングについては、稼動検証が必要となり工数面で即効性に欠ける。またパフォーマンスが悪化することも考えられる。

４成長期

性能管理を運用に定着させる

ようやく初期トラブルが落ち着きだした頃、性能管理に意識が向く。システム企画・設計の段階で稼動統計データの取得や業務実行ログを取得する仕組みが織り込まれるシステムは多くなっている。しかしながら、取得すべき稼動統計データや業務実行ログを性能管理として活用する仕組みが設計されていることはまだまだ少ない。背景には「本番システム運用は正常に稼動して当たり前」という固定概念があるように考える。

4.1性能管理の目的

本番システム運用において性能管理を行う目的は、利用者であるエンドユーザ業務に支障がないようにサービスを提供することにある。安全・安心・安定・信頼などの言葉が並ぶ場合が多い。ここで問題になるのはサービスレベル（以降、SLAと呼ぶ）が具体的に設定されていないケースが多いことである。そのため性能管理の項目と基準となる数値を設定することから始めなければならない。SLAを決めずにマシンサイジングができているのかと疑問になる瞬間である。

4.2性能管理の項目

目的にあった性能管理の項目が取得可能であるかを確認する必要がある。取得項目が存在しない場合は、存在する性能管理の項目の中から代替となる項目を選択する。

性能管理の項目の代表格としてオンライン・トランザクションのレスポンス時間、夜間バッチ処理の締め切り時間の２項目を設定することが多い。両方とも業務系の統計データである。故にシステム設計時からこれらの時間をログとして記録する仕組みが必要となる。

WEBサーバの場合はURL単位にHttpアクセスログとしてレスポンス時間を取得できる。ただしWEB表示画面そのもののレスポンス時間はHttpアクセスログには出力されない。WEBサーバ以外の場合、トランザクションのレスポンス時間を取得することが困難であることが多い。できればトランザクションの開始、終了を業務ログとして記録する仕組みがほしい。記録できない場合はシステムリソースの稼動統計データをみなし値として性能管理を行うことになる。夜間バッチ処理の締め切り時間についても同様に業務ログとして記録できる仕組みがほしいが、ジョブスケジューラ等を使用している場合はジョブスケジューラから取得できる。

4.3性能管理の流れ

大枠の工程は次のようになる

①統計データの取得
　↓
②性能評価・分析・レビュー
　↓
③改善計画の立案・実施可否の判断
　↓
④実施・効果検証

①は継続的に自動収集する仕組みを構築する。②は定期的に行う。実施サイクルは日次、週次、月次ベースなどで業務の重要度に依存して行う。性能評価・分析・レビューの結果、特に問題がなければ、③以降のアクションはない。③では改善計画を立案し、予測される効果度合い、実施の容易性、コスト性、緊急度などの項目を判断材料として、優先度を決め実施の可否を判断する。④は確実に改善計画を実行し、その効果を統計データから検証する。

4.4統計データの取得

自動的に24時間/365日、継続して取得できる仕組みを構築する。

システムリソース系の統計データは取得するインターバル、タイミングを揃えておくのが良い。性能評価・分析の際に相関判定をできるようにしておくためである。

業務系の統計データは処理単位に処理時間（レスポンス時間）が取得できるのが良い。

4.4.1性能管理の項目（システムリソース系）

次にシステムリソースの性能管理の代表項目を列記する。

・プロセッサ：使用率、ランキュー状況
・メモリ：フリーメモリ量、ページング活動、
　　スワップファイル空き容量
・I/O：ビジー率、レスポンス時間
・プロセス：プロセッサ使用時間、メモリ使用量
・DB：バッファヒット率、論理リード回数

OS標準機能で計測している値を取得できるのが良い。これは本番システムにおいて統計データを取得する負荷をできるだけ小さくするためである。また値の精度を揃えることができる。これは評価値として使用する条件になる。

UNIX環境の場合、sar、vmstat、iostatコマンドなどの出力値になる。Windows環境の場合はレジストリに記録されているシステムカウンタ値になる。

4.5性能評価・分析・レビュー

4.5.1サイクル

評価サイクルとその利用方法について運用を取り決めておくことが必要である。

日次ベースでは監視の意味合いが強くなる。各

統計値があらかじめ定めた閾値を越えていないかを確認する。閾値を越えた場合、必要に応じて対応することになる。

月次ベースでは月次評価、ピーク日評価、前月対比評価、前年同月対比評価などが行われる。長期レンジでは年次、3ヵ年の期間を評価することもある。これらは稼動傾向を掴むために行われる。

基本的に、月次ベースの評価サイクルが多い。

4.5.2統計データの可視化

性能評価分析のため取得した統計データを時系列のグラフとして加工する。各統計データをカテゴリ別にインターバルを揃えてグラフ化する。また閾値判定をするため、グラフに閾値線を引くことで、より明確に問題の有無を判断できる。

可能であれば統計データの加工においても自動化されていることが望ましい。稼動統計データは取得していても、グラフ化する工数がかかる。またはツールがないため、稼動統計データが活用されていないケースがある。できればシステム設計の一部として、性能データを加工し、可視化する手段を含むべきである。

4.5.3性能評価分析

システムリソースの統計データから性能評価をするには基本的なOSの動きを理解しておく必要がある。判定したいことは、どのような動きが処理遅延を引き起こすのか、また、引き起こす可能性があるか、ということである。性能評価分析では、既に処理遅延が発生している状況でボトルネックを見つけだす。また現状に問題がなければ、潜在しているボトルネックの発見が必要となる。

運用担当者は運用業務には長けているが、システムリソースを評価することには不慣れなケースが多い。この成長期においては運用担当者が性能管理の重要性や性能評価分析手法、システムリソースの動きを学び、性能管理のノウハウを培うよい機会でもある。

4.5.4稼動値

誰もが同じように判断できる数値作りが必要である。

ユーザとシステムリソースの動きや評価分析手法などについて話していた時のことである。次のような要望があった。プロセッサ、メモリ、I/O、DBと複数の項目を見て判断するのではなく、日経平均のようにその数値をみれば、すぐに判断できる数値はないのか、もしくは新たに作ることは出来ないのかとのことであった。誰がみても同様に判断できる数値づくりを要望された。

該当システムを性能評価したところ、メモリ、I/O、DBともに業務負荷のピーク時において性能上、問題がないと判断できた。プロセッサの稼動をもとに基準値づくりを始めた。稼動状況を表す数値として平均値、パーセンタイル値、標準偏差値などを候補とした。一つの値だけでは変化がわかりにくいと考え、次のような計算値を採用した。

プロセッサ稼動値＝
　平均使用率＋50p＋80p＋90p＋最大値

ここではこの値をプロセッサ稼動値と名づけた。実際に使用した値は15分インターバルのプロセッサ使用率である。1日に96サンプルある。この稼動値により1日のプロセッサ使用率の変化がよりわかりやすくなる。

過去の計測値から閾値として、プロセッサ稼動値330以上を警戒レベル、400以上を警告レベルと設定した。（図１）

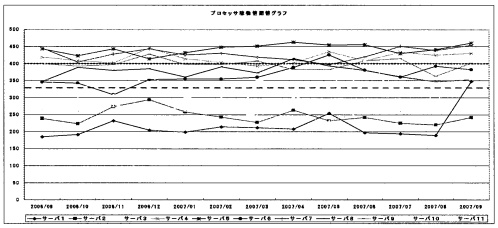

図１

4.5.5閾値の見直し

性能評価分析は決められたサイクルで繰り返し実施される。設定された値で継続的に判断することに意義があり、人に依存しない評価結果を得ることができる。ただこの設定された値が現状にそぐわなくなることもある。閾値の見直しも定期的に実施する必要がある。

手段としては、最初は全サーバに共通の閾値を設定する。性能評価分析を繰り返すことで個々のシステム、サーバの様子がわかってくる。次に共通の閾値から、システム単位、またはサーバ単位へと、より個別に対応できる閾値に調整する。

4.5.6CPUループの発見

プロセッサ使用率の閾値を設定する際に必ず

登場するのがCPUループをどのように発見するか、という課題である。

たとえば、単純にプロセッサ使用率の閾値＝70%と設定しても、1CPU環境であればCPUループを発見することはできるが、複数CPU環境であれば閾値の篩に掛けることはできない。搭載CPU数を考慮した閾値設定が必要である。

一例をあげる。搭載CPU数＝4のサーバで、夜間バッチ終了からオンラインの立ち上げまでの時間帯は、必ずプロセッサ使用率が数%になる。この時間帯の閾値を24%に設定し、プロセッサ使用率が24%を越えると、管理者に通知する仕組みを構築した。これによりCPUループの発見につながった。

4.6改善計画の立案・実施可否の判断

性能評価分析の結果、システム改善の必要ありと判断すると改善計画を立案する。実施可否の判断に必要な項目は次の通りである。

・具体的な改善案
・期待される効果
・必要とされる工数
・必要とされる人
・必要とされる技術
・コスト
・リスク度合い
・緊急度合い

複数の改善案を立案し、上記項目をもとに実施優先度を計算する。場合によっては見送ることもある。

実施する改善案はできるだけ少ないのが望ましい。これにより正確な効果検証ができる。また実施の際のリスクを軽減することにも繋がる。

4.7実施・効果検証

改善案実施にあたっては計画のとおり実施する。

4.7.1テスト環境での改善策の実施・効果検証

通常は、テスト環境においてその効果を検証する。検証では２つの要点を持つ。一つは本番システムに悪影響を与えないか。端的に言えば今まで通り処理を正常に行えるということである。もう一つは、ほんとうに改善効果が出るかということである。

この際に肝心となることは本番と同じ環境をいかに正確に準備できるかである。可能であれば本番環境と同じ性能問題を再現し、改善案を適用し、その効果を検証するほうが望ましい。

4.7.2本番環境への改善策の適用

テスト環境での検証の結果により、悪影響をあたえることなく、性能改善効果を確認できれば本番環境に適用する。改善後の本番環境の統計データを取得することで効果を検証することができる。

５安定期

システム運用も軌道に乗り、安定的な時期を迎える。

5.1データ量増加によるシステムへの影響

システムが順調に稼動しおおよそ１～２年経過するころ、取り扱いデータ量の増加により応答性能に影響が出始める。日々に発生するデータ量が増加傾向でなくても、保持しているデータ量が大きくなることで応答性能に変化がでる。変位は徐々であるため、月次ベースの性能評価分析では気が付かないかもしれない。この場合、年次ベースで傾向分析するのがよい。リソースの統計データでみれば、I/Oレスポンス時間、アクセス待ち個数の変位を確認することで、システムへの影響度合いを測ることができる。

5.2性能を左右する要因

取り扱いデータ量の増加によるシステムへの影響を受けるのはDBシステムである。DBシステムの安定した性能を保つためには日常のメンテナンス運用が大事である。性能を左右する要因として最も大きな影響を与えるのはアクセスパスである。いかに効率よく最短のアクセスで対象となるデータを獲得するかが重要となる。更新処理が行われると、どうしてもデータブロックの分割やデータブロック内の断片化、利用できないスペースなどが出来てしまう。これらは直接、アクセスの効率性を悪くしてしまう。コストベース運用であれば定期的な分析処理（analyze）は必須である。知らぬ間にアクセスパスの効率が落ち、業務処理遅延の結果を招く。

図２は業務遅延が発生した時に取得した統計データをもとにグラフ化したものである。DBメンテナンスを行っていなかったために、Diskのアクセス待ち個数が200個/秒以上となっていた。そこ

でINDEXの再編成を行うことで50～60個/秒程度まで待ち個数が減り改善した。

担当者はメーカが提供している自動チューニング機能を信じ、DBメンテナンスを行っていなかったことが要因であった。

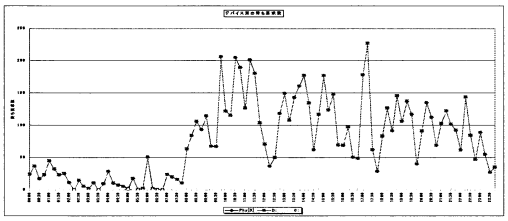

図２

保持するデータ量を小さくすることでアクセス効率を上げる方法もある。保存期限の満了したデータのアーカイブなどが上げられる。本番稼動初期の段階で、データアーカイブまで運用に含めスケジュールされていることは稀である。当然ではあるが対象となるデータが存在しないためである。酷いケースではそのまま忘れ去られ抜け落ちていることがある。

保持するデータ量が大きくなると応答性能に影響する一因はデータのハイウォーターマーク値が上げられる。このハイウォーターマーク値の影響を受けにくくする仕組みとしてパーティショニングを使用することができる。

図３は安定期に入ってからパーティショニングを適用した事例である。

WebAP-DB構成のシステムである。DBのパーティショニングを行った結果、応答時間が1/2～1/3程度まで改善された。

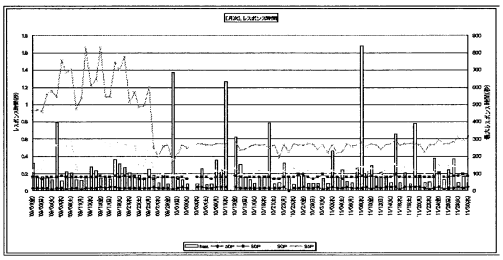

図３

5.3接続ユーザ数の増加による影響

同時接続ユーザ数の増加により同時接続セッション数が増加すると、システムへの負荷も大きくなり、応答性能に影響がでる。このような変化には性能管理の範疇で対応できる。一番影響が大きいのは、接続不能になることである。ユーザとの接続を受け持つミドルウェアは最大接続数を持っている。システムが安定運用されているため、システムの入り口での受付容量の確認を忘れがちである。

WEBサーバでの実例である。安定稼動していたWEBサーバに接続できないとの連絡があった。システムリソース、ネットワークまた該当プロセスには問題がなかった。数時間の調査の結果、同時接続セッション数に達していたことがわかった。管理体制に問題があったことは否めない。

６終焉期

新しい業務要件に応えられない、ハードの故障回数が増えてきた頃にはシステムは終焉期に入る。ほぼ５年以上を経過するころと想定される。

次期システムへの移行を踏まえ、性能管理情報として蓄積したものを整理することになる。

6.1長期傾向分析

このフェーズの長期傾向分析の目的は次期システムサイジングのベースデータとして利用するためである。本番開始からの統計データを使用し次のような整理を行う。

・稼動傾向
・障害発生件数
・故障件数
・利用ユーザ数の推移
・運用上の問題点

７最後に

「当たり前のことを当たり前にする」ことができれば、本番環境における性能に関する問題の発生頻度は削減できるものと考えられる。

システムの設計段階からシステムに性能管理要件を織り込むことが先ず必要である。システム発注から本番稼動までの短期間化が求められて、非機能化要件のひとつである性能管理が業務開発に比べ後回しにされがちである。

システムのライフサイクルは開発期間より、本番運用期間が長い。この本番運用期間のシステム品質を保つためにも性能管理の重要性を認識し、システムのライフサイクルを意識した質の高い性能管理の実践が不可欠である。

以上