Vol.2016-IOT-32 No.19
2016/3/3

任意の期間に発生した複数の性能障害に対する対策案生成方式に関する検討

鈴木 克典[1]　金子 聡[1]　江丸 裕教[1]

概要：　IT システムにおいて性能低下が発生すると，発生時の稼働情報に基づいて構成変更等の対策を行い，性能を改善する．稼働情報は IT システムに対するワークロードによって常に変化するため，ある時点の稼働情報に基づいて作成した対策が，他の時点においても同様の効果があるとは限らない．そのため，ユーザが複数時点で発生した性能低下を改善しようとしても，それらの全て時点で一様の効果をもたらす対策を取ることができない場合がある．そこで，複数時点においてユーザが設定した性能目標値を満たす PMSE 法 (method for Plan-creation to Minimize Side-Effect) を提案する．評価の結果、複数時点の性能情報を用いることで，効果の確実性と副作用のバランスの取れた対策案を生成できることを示す．

A Method for Plan-creation to Solve Performance Problems within Any Period of Time

Katsunori Suzuki[1]　Satoshi Kaneko[1]　Hironori Emaru[1]

Abstract: If performance failures are occurred, IT system administrator should consider a counter measure plan by passed performance and configuration data to solve performance failures. However, performance data is constantly changed affected by any workloads in IT system. So, a plan to solve performance failures based on performance data at a certain time point is not necessarily be effective too at an on other certain time point. Therefore, administrator can not always make a plan to improve performance evenly at more than two time points. To solve this problem, we propose the PMSE method (Plan-creation to Minimize Side-Effect) which generates a plan to improve system performance to a user defined objective at multiple time points. Evaluation shows that proposed method can generate the reconfiguration plan which keep a balance between effectiveness and minimization of side effect.

1. はじめに

近年の情報化社会の発展に伴い，情報システムは金融，交通，流通など多様な分野を支えるインフラとなっている．そのため，情報システムの障害が社会に与える影響は大きくなっており，情報システムの運用ではサービスレベルの維持が重視されている．

一方で，情報サービスの需要増加に伴い，データセンタの大規模化が進んでいる．また，クラウドコンピューティングの利用拡大により，これまで分散して設置されていたストレージやサーバ等のデータセンタへの集約が進んでいる．このような中，データセンタにおいては集約率の更なる向上や、柔軟性の向上のため、仮想化技術の利用が進んでいる．このように，データセンタの大規模化と仮想化が進んだことで，障害や，障害対策の影響の波及範囲が広範囲となり，その結果サービスレベルの維持は難しい問題となっている．

例えば，複数の VM(Virtual Machine) はサーバやストレージを共有するため，VM のワークロードが重なった結果，特定のサーバやストレージに負荷が集中してしまい，予期せず複数 VM の性能が低下することがある．このような性能低下に対し，管理者は将来的に同様の負荷が発生しても性能低下が再発しないよう対策を講じる．具体的に

[1] 日立製作所研究開発グループ
Center for Technology Innovation
Research & Development Group
Hitachi,Ltd.

は，性能低下発生時のサーバやストレージの稼働情報を基に，この時点における性能を改善する対策案を立案する．VM のワークロードは常に変化するため，性能低下発生時点だけではなくその前後も含め，考慮する時点が多いほど複数の VM のワークロード変化に対応した対策案を作成でき，将来的に性能低下の再発を防げる可能性が高まる．

一方で，考慮する時点が多いほど，全時点で性能低下を解消するためにはより多くの VM の配置変更が必要となる可能性がある．その結果，VM の配置変更先のサーバやストレージにおいて，既存の VM とワークロードが重なり二次的に性能劣化が発生する可能性が高まる．

そこで，本稿では管理者によって指定された複数時点の性能低下を解消しつつ，対策を実行することに伴い発生する二次的な性能劣化を最小限とする対策案の生成方式について提案する．

以降，本稿ではまず 2 章で性能を管理するための一般的な運用手順について述べた後，性能管理における課題を示す．そして，3 章で課題を解決するための提案方式である PMSE 法 (method for Plan-creation to Minimize Side-Effect) について述べ，4 章で PMSE 法を評価する．最後に 6 章で考察し，6 章で結論を述べる．

2. 性能管理と課題

2.1 性能管理

情報システムの運用においては, 管理者はシステム利用者に対して一定の性能や品質を提供するために SLO(Service Level Objective) を提示し，この SLO を守るように日々運用を行うことが一般的である [1][2].

管理者は SLO を満たした状態を維持するために情報システムの性能を監視し，性能低下が見られた際には性能改善のために構成変更等の操作を行う．以下に性能監視から性能改善までの運用について一般的な手順を示す．

(1) 性能監視と異常検知

情報システムの性能監視には Zabbix[3] や Hinemos[4] といった監視ツールが広く使われている．管理者は SLO を守るために，これらのツールを用いてサーバ内のメモリや CPU，ストレージ内のボリュームといったリソース（以降，単にリソースといった場合はこれらを指す）の負荷 (例：利用率や IOPS) に対してしきい値を設定して監視する．このとき，SLO よりも一段階高い基準の値を警告しきい値として設定することで，余裕を持って性能監視を行う．

(2) 異常原因の分析

性能低下検知後，管理者は適切な対策をとるために情報システムの状態を分析し，性能低下の原因である箇所とその内容を把握する．これは，データセンタが大規模化，複雑化している現状において困難な問題であることから，従来より研究 [5] がなされており，原因分析の支援ソフトも存在する．

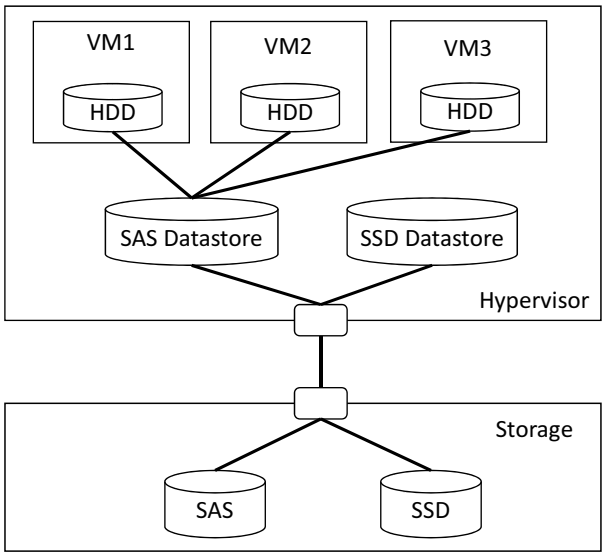

図 1 情報システムの構成例

(3) 対策案の立案と実行

管理者は性能低下の原因に基づき，将来同様の負荷が発生しても問題が再発しないように，性能改善のための対策を実行する．対策の立案も原因分析と同様もしくはそれ以上に困難であることから，事前に原因やシステムの状態に対する対策案をパタン化し，障害発生時に管理者に対して提案する製品が登場してきている [6].

図 2 は情報システムが**図 1** に示す構成である場合における，VM とストレージ装置内のボリュームに対する I/O 量の経時変化の例である．SAS Volume に対して設定された警告しきい値を超えた場合に管理者にアラートが発行される．その後，管理者は SAS Volume に対する I/O 集中を将来的に防止するために，過去の VM の稼動情報を確認し，VM を異なるデータストアに移動させる等の対策案を立案し，実行する．

2.2 性能管理の課題

性能低下が発生した際には同様の負荷発生時に性能低下事象が再現しないように，性能低下事象が発生している過去時点においてリソース負荷が警告しきい値を下回る状態となるように対策講じる．しかしながら，VM のワークロードは時間と共に変化していくため，ある時点の性能情報では適切な対策が，別の時点でも有効であるとは限らない．例えば，図 2 では警告しきい値を 2 回超えている．期間 A のみに着目した場合には VM1 を移動させれば警告しきい値を下回る状態となるが，2 回目の警告しきい値違反

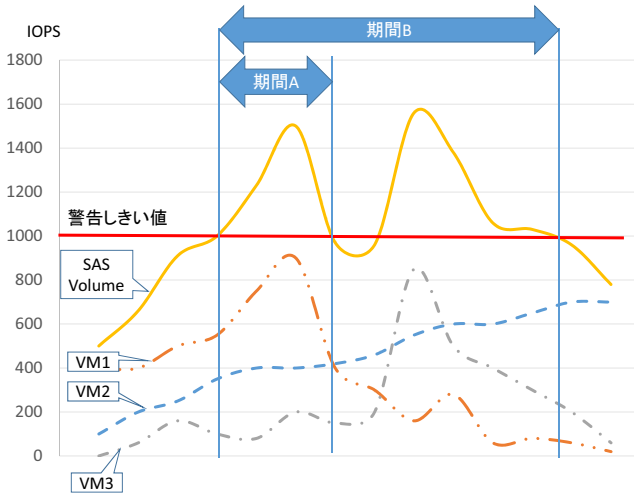

図 2 情報システムの性能情報例

を解消することはできない．しかしながら，期間 B 以降に対策を検討する場合には，管理者は過去に起きたすべての問題が再現しないようにすることを望むと考えられる．このためには，管理者は期間 B に発生した複数の性能低下事象を考慮して対策案を生成する必要がある．

一方で，当然に対策の実施によって性能低下が発生していないリソースに対して新たな性能低下の問題を引き起こしてはいけない．しかしながら，対策生成時に考慮する時点の数が増えるほど，複数の VM のワークロードに対応するためにより多くの VM 移行が必要となり，VM 移動先のリソースに対してより多くの負荷をかけてしまう．例えば，図 2 について，期間 A に対応するためには 1 個の VM を移動させればよいが，期間 B に対応するためには少なくとも 2 個の VM を移動させる必要がある．これにより，より長い期間の性能低下事象を考慮して対策を生成するほど，対策の実施によって VM 移動先リソースの負荷が警告しきい値を超えてしまう確率が高まる．

以上より，対策を実施することによる二次的な性能低下をできる限り小さくしつつ，複数の性能低下事象に対する対策を生成することが課題である．

3. PMSE 法

本章では 2.2 節で述べた課題を解決する PMSE 法について述べる．まず，3.1 節で対策案を生成する際の方針を述べる．次に，3.2 節で以降の議論で用いる用語を定義する．そして，3.3 節，3.4 節で対策案の生成方式を述べる．

3.1 対策案生成の方針

2.2 節で述べた課題を解決するため，任意の期間を入力とし，その期間内に発生した性能低下事象をすべて解消する対策案を生成するシステムを提案する．以下に，提案システムの方針を示す．

方針 1　性能改善の対象のリソースについて，指定された期間において，負荷が警告しきい値を下回る状態とする．

これは，管理者が性能改善の目的で期間を指定して性能低下の対策を行っている以上，管理者はその期間において負荷が警告しきい値を下回る状態となることを期待していると考えられるためである．

方針 2　性能改善の対象外のリソースについては，対策実施による二次的な性能低下を最小限にする．

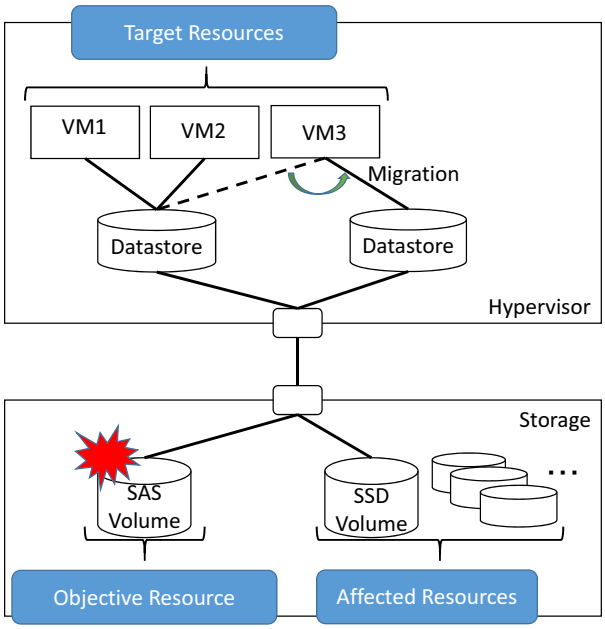

図 3 用語に対する具体例

3.2 用語の定義

以下の通り次節以降で用いる用語を定義する．

Target Resource　警告しきい値超過が検知され，対策を行うことで負荷を警告しきい値まで低下させるリソース．

Candidate Resource　Target Resource の負荷を低下させるための対策として行われる構成変更または設定変更の対象となるリソース．

Affected Resource　対策を行うことで，二次的に性能劣化が発生する可能性があるリソース

図 1，図 2 の構成，性能変化のケースを用いて，上記用語の具体例を**図 3** に示す．このケースでは，ストレージ装置内の SAS Volume への負荷が警告しきい値を超えており，その負荷低減のために性能対策を立案する．そのため，Target Resource は SAS Volume となる．ここで，SAS Volume の負荷を低減するために VM が配置されるデータストアを変更する構成変更を行う場合，Candidate Resource は移行する VM となる．また，VM を移行することで Target Resource 以外のストレージ装置内のボリュー

ムに対する負荷が増加する．そのため，Affected Resource としてはストレージ装置内の Target Resource 以外のボリューム，及び対応するデータストアがある．

3.3 対策としてとり得る対策の整理とモデル化

対策案の生成アルゴリズムを構築するにあたり，まず性能低下を解消するためにとり得る対策手段を分類・整理する．

対策としてとられる手段は大きく分類すると，(1)Target Resource の性能を増強する手段，(2) システムのワークロード総量は変えずにワークロードの配置を変更する手段，(3) ワークロード総量を削減する手段，の 3 種類が考えられる．

以下，それぞれ手段についてパラメータを求めるためのモデルを示す．

(1) Target Resource の性能を増強する手段

ハードディスクの追加といったリソースの量を増加させる変更の場合では，指定されたすべての時点で警告しきい値を下回る状態となるために必要な追加リソース量を求める．

(2) ワークロードの配置を変更する手段

VM の移動といったワークロードの配置を変更する手段では，以下の手順で操作対象の Candidate Resource を決定する必要がある．

(a) 方針 1 を満たすために，指定されたすべての時点で負荷が警告しきい値以下となる Candidate Resource の組み合わせパタンを求める．

(b) 上記 Candidate Resource の組み合わせパタンより，方針 2 に従って適切な組み合わせを一つ求める．

(3) ワークロードの総量を削減する手段

VM の削除や I/O 量制限といったワークロードを削減する手段では，Target Resource の負荷が警告しきい値を下回る状態とするために必要な制限量や Candidate Resource の組み合わせを求める．

上記の 3 手段では，(1) のみが VM への性能劣化を起こさずに性能対策を行うことができる．しかし，機器に関するコストの削減を進めている通常のデータセンタでは，性能低下時の追加用にリソースを遊休させておくとは考えにくい．そのため，(1) をとるためには新規にリソースを調達する場合が多く，実施に時間がかかる対策となる可能性が高い．

そこで，次善の対策として (2),(3) が候補となるが，(3) の方が既存の業務への影響が大きくなる可能性がある．例えば，(3) の手段例としてはディスクへの I/O を制限する方法がある．VM のディスクへの I/O を 20%制限した場合には，単位時間当たりに実施可能な I/O 数が減るため大量のデータを読み書きする際に要する時間が増加してしまう．そのため，(2),(3) の比較では，まずは (2) を行うのがよいと考えられる．

本稿では次節以降，ワークロードに対する影響を最小限に抑えることができる (2) の手段について述べる．

3.4 対策副作用の評価方式

本節では前節で述べた手段 (2) において，ワークロードの配置を変更する際に，ユーザにより指定された期間における各リソースの負荷平準化と各リソースの警告しきい値違反点の最小化を両立する，PMSE 法 (method for Plan-creation to Minimize Side-Effect) を提案する．

リソースに与えられる負荷は VM のワークロードの重ねあわせによって決まり，VM のワークロードはアプリケーションの種類やエンドユーザの利用状況の変化による．そのため，対策実行後の Affected Resource の負荷は多様な傾向となり得る．そこで対策案を比較する上では，以下の観点で各 Affected Resource の負荷を比較し，評価する．

観点 1 可能な限り対策実行後の各リソースの負荷が平準化されていること．

特定の Affected Resource のみに負荷が集中すると，各 Affected Resource を利用する VM 間で性能差が生じ不平等となる．これを防止するために，各リソースの負荷を平準化する．

観点 2 対策実行後の各リソースの負荷が警告しきい値を下回ること．

しかし，VM のワークロードによっては上記の両方を満たす対策を生成することができず，各 Affected Resource の負荷を平準化できない対や，策負荷が警告しきい値を超えてしまう Affected Resource が発生する対策となってしまう場合がある．

そこで，そのような場合であっても可能な限り上記 2 つの観点に従った対策を選択するために，上記の各観点について以下 2 つの計算方法で対策案の評価値をそれぞれ計算し，2 つの評価値を利用して対策案の比較を行う．

方法 1 負荷平準化評価法

対策実行後の Affected Resource について，指定された期間内の時点毎に，最も負荷の大きい Affected Resource の負荷と，最も負荷の小さい Affected Resource の負荷を比較し，負荷の差を計算する．計算した負荷の差を，全時点分足し合わせた値を評価値とする．

この評価値は，より小さな値を持つ対策案を選択することで，期間内においてより狭い範囲に各リ

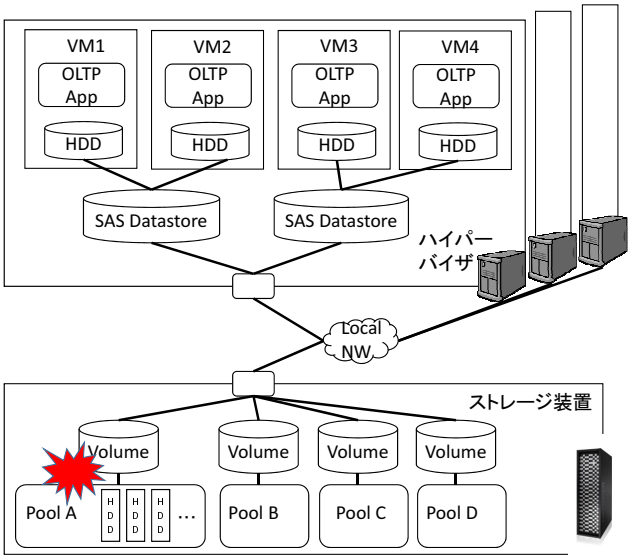

図 4 シミュレーションの想定環境と想定シナリオ概要

ソースの負荷が収まり，平準化されていることを示す．

方法 2　しきい値違反点最小化評価法

指定された期間内に，対策実行後の各 Affected Resource の負荷が警告しきい値を超える回数を計算し，全ての Affected Resource についてその回数を足し合わせた値を評価値とする．

最終的には，両方式の評価結果を正規化し，スケールを合わせた後で和をとった値が最も小さくなる対策案を選択する．両方式で計算した評価値を足し合わせるのは，各計算方法は単体では評価に差がつかず対策案を絞り込めない場合や，極端な対策案となってしまう場合があるためである．たとえば，すべての対策案について対策実施後の Affected Resource の値が，全時点で警告しきい値を下回っている場合には方法 2 の評価値は 0 となり，対策案毎に差がつかない．一方で，方法 1 では Affected Resource 間の負荷の差以外は考慮していないため，必ずしも Affected Resource の負荷の絶対値が小さくなる対策案が選ばれない．そのため，両観点でバランスの取れた解を安定的に生成するために，両方の計算方法の評価値を用いる．

4. 評価

本章では，3 章で述べた性能改善の対策案生成方式を評価した結果を示す．

4.1 シミュレーション概要

金融機関におけるワークロードを想定し，シミュレーションを行った．**図 4** に想定環境を示す．近年では仮想化の性能向上が進んでいることから，仮想化環境を用いていることを想定する [7]．想定環境では，ストレージ装置にハイパーバイザが接続されており，ハイパーバイザ上で複数の VM が稼動している．また，VM 上では金融機関における代表的なワークロードである OLTP アプリケーションが稼働している．各 VM の OLTP アプリケーションは同様のものであり，負荷分散を容易とするために複数の VM に分散配置されている．ストレージ上には複数のプールが用意されており，プールからはボリュームが作成されている．そして，各ボリュームはハイパーバイザにマウントされており，データストアとして利用可能となっている．このような環境において，プールには IOPS に警告しきい値が設定されており，あるプール (Pool A) で警告しきい値超過が発生したために，Pool A に対する IOPS を低下させる対策を立案するシナリオを想定する．

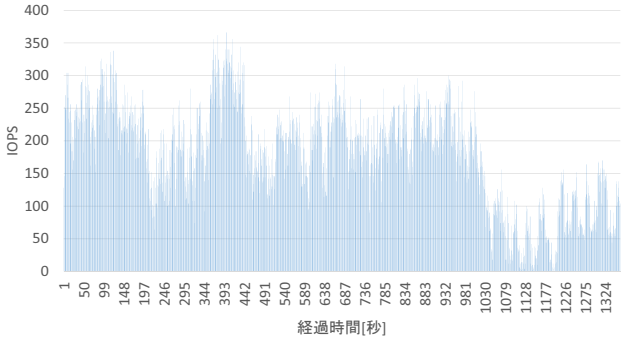

図 5 VM の IOPS 例

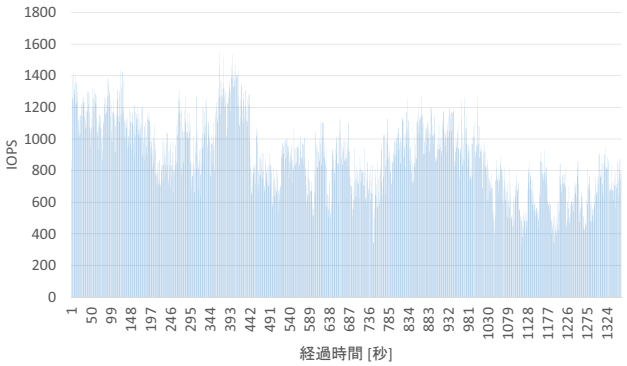

図 6 プールの IOPS 例

データはある金融機関における OLTP アプリケーションのディスク I/O をトレースした公開データ [8] を用いた．データを基に 30 秒単位で IOPS を計算したデータを**図 5** に示す．横軸は I/O 開始からの秒単位での経過時間を，縦軸は IOPS を示す．これは，VM の I/O パタンの 1 例であり，本来は VM 毎に異なる I/O パタンとなる．しかしながら，1 台の VM の I/O トレースデータしか公開されていないため，本シミュレーションでは図 5 のデータを基に図 4 中の各 VM のディスク IOPS をシミュレートすることとした．それにあたり以下の想定を置いた．

表 1　シミュレーションの設定

プールの数	5 個
プールとデータストアの対応	1：1
データストア当たりの VM 数	5 個〜8 個
プール単位 IOPS の警告しきい値	1,000 IOPS
総時点数 (秒換算)	1,360 点 (40,800 秒)

- OLTP アプリケーションは北米や欧州などタイムゾーンが異なる複数の地域からもアクセスされる．そのため，VM 毎に I/O の周期がずれることがある．
- 地域によって人口等が異なることから各 VM が発行する IOPS が異なる．
- 以上，2 点の想定によって I/O 周期のずれや，IOPS のずれが発生するものの，OLTP アプリケーション自体のアクセスパタンは地域によって変化しない．

以上の想定より，図 5 のデータを時間軸に関してスライドさせる，また，波形を維持して拡大・縮小する等の処理を行うことで，各 VM が発行する IOPS を計算した．そして，各 VM の IOPS を合算することでプールに発行される I/O の IOPS を計算した．生成したプール当りの IOPS 例を**図 6** に示す．

シミュレーションの詳細な設定は，**表 1** とした．各プールは SAS HDD から構成されているとし，ディスクの性能に合わせて VM が各データストアに配置されていると想定した．また，プールの警告しきい値は 1,000IOPS とした．

また，管理者が行う対策は VM を警告しきい値超過が発生したデータストアから，別のプールから作られているデータストアに移動させることとした（以降，VM を移動すると表現する）．本シミュレーションでは計算の簡略化のため，VM を移動するとプールの IOPS から VM の IOPS が減算され，移動先のプールの IOPS に VM の IOPS が加算されるとした．ここで，厳密には VM のワークロードによってプールへ発生する I/O 量は，キャッシュヒット率などの条件によって違いが生じる．しかしながら，キャッシュ量などの条件が各プールで同一，かつ各 VM は同様パタンのワークロードであることとしたとき，VM の移動前後で IOPS はそれほど大きく変わらず，この違いによって評価結果が大きく変わらないと考えられるため，上記のような計算の簡略化を行った．

以上の想定の下，複数時点の性能低下事象に対して有効な対策案の生成可否と，Affected Resource の性能劣化低減率の 2 つの観点について，提案方式を評価した結果を次節以降に述べる．

4.2　考慮する時点数と対策の効果

PMSE 法は、性能低下対策の対象とする時点数を増やすほど Target Resource のしきい値違反を抑制出来る一方で、Affected Resources のしきい値違反が発生する可能性

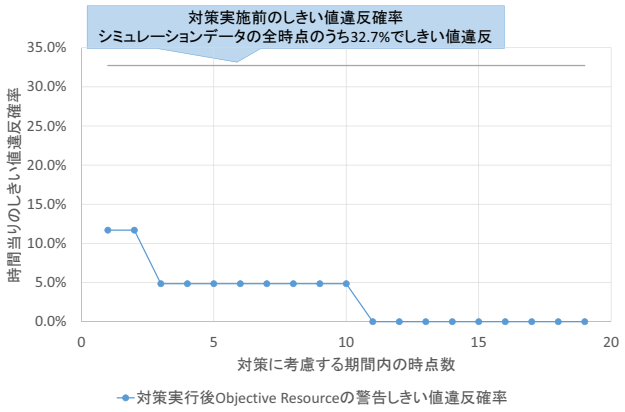

図 7　時点数変化に伴うしきい値違反発生確率の変化

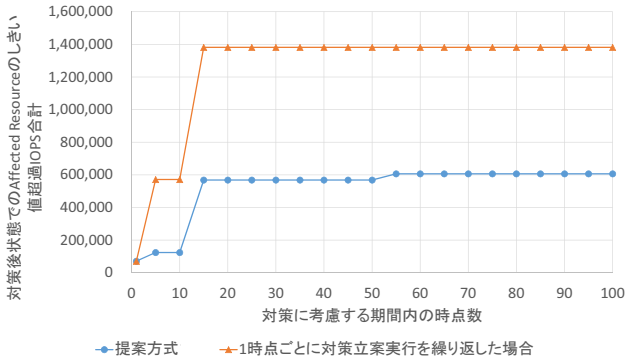

図 8　時点数変化に伴う Affected Resources のしきい値超過 IOPS 累計の変化

が高まると考えられる．そこで、性能低下対策の対象とする時点数と，生成した対策実行後に Target Resource にて IOPS 警告しきい値超過が再発する確率の関係を評価した結果を，**図 7** に示す．

横軸は対策の生成に用いた時点の数である．縦軸は生成された対策実行後に，Target Resource(Pool A) で警告しきい値超過が再発する確率を示す．これは，対策生成に用いていないデータも含むシミュレーションデータにおいて，性能低下が発生する確率であり，将来同程度の負荷が再発した場合に，対策実行後の Target Resource において性能低下が再発する可能性を示している．

対策を実施する前では，シミュレーションデータの全時点のうち 32.7%の時点でプールの IOPS が警告しきい値を超えていた．一方，性能低下を検知した時点 1 点のみを計算に用いて生成した対策を実施した場合は，警告しきい値を超えた時点の数は全時点のうち 12%となった．そして，性能低下検知時点から 11 時点のデータを用いて対策を生成することで，対策実施後の警告しきい値を超える時点は 0 となった．

4.3　考慮する時点数と対策の副作用

次に，性能低下対策の対象とする時点数と，対策案を実

行した場合の Affected Resource に対する負荷増加の関係について評価した結果を，**図 8** に示す．横軸は対策の生成に用いた時点の数である．縦軸は，生成された対策を実行した後の状態において，Affected Resource である全プール (Pool B,C,D,E) の各時点の IOPS の警告しきい値超過分を，シミュレーションの全時点に渡って合計したものである．

図 8 では PMSE 法の比較対象として，指定された期間中の全ての時点毎に独立に負荷を警告しきい値以下とする対策を生成し，それらの重複を排して最終的な対策案を生成する方式 (以下，事象間独立対策方式と記載) の結果も示す．事象間独立対策方式では，指定された期間の最初の時点から 1 時点ずつ性能状態を確認し，Target Resource の負荷が警告しきい値を超えていた場合には，当該時点において最も IOPS の大きい VM を，最も IOPS の少ないプールに移動させる対策を立案することを繰り返す．

図 8 より，PMSU 法と事象間独立対策方式のそれぞれで生成した対策案を実施した後の状態を比較すると，Pool B,C,D,E の IOPS の警告しきい値超過分の全時点累計が 2 倍から 5 倍程度の比となっており，差は最大 800,000 IOPS(時点数 11 から 50) であった．すなわち，シミュレーションの総時点数 1360, 各 Pool の警告しきい値 1000 IOPS, Affected Resource のプール数 4 であることから，対策生成に用いる時点数 11 として PMSE 法で生成した対策案実施後の Affected Resource(Pool B,C,D,E) の IOPS 総計は 6,040,000 IOPS となり，事象間独立対策方式では 6,840,000 IOPS となる．

以上より，PMSE 法は事象間独立対策方式に比べ，対策に用いる期間内の時点数 11 から 50 のときに，最大で対策案実施時の Affected Resource(Pool B,C,D,E) の IOPS 総計を 12%，警告しきい値超過分の累計 IOPS を 50%少なくすることが出来ている．前節の結果とあわせ，このシミュレーションでは対策生成に用いる期間内の時点数 11 以上の場合において，PMSE 法では各時点に対して独立に対策を生成した場合に比べ，Affected Resource の負荷を 12%削減しつつ，同等の性能低下改善効果を持つ対策を生成することが出来たことが分かる．

5. クラウド環境への適用可能性

クラウドの利用増加に伴い，アプリケーションの開発運用のサイクルが現状に比べ，短くなっていくことが予想されている．従来はアプリケーションのライフサイクルが長かったため，管理者がシステムのワークロードを把握し，事前に性能設計することができた．しかしながら，アプリケーションの開発運用サイクルが短くなるクラウド環境では，ワークロードの特徴を把握する時間がとれず，管理者が事前に性能設計を行うことが困難となる．そのため，今後は性能対策を自動化する重要性が増してくると考えられる．

しかしながら，現状では対策によって発生する二次的な性能劣化に対し，管理者が対処できる範囲で性能対策が行われている．そのため，自動実施された対策による二次的な性能劣化に対応することは困難である．また，従来は対策案の処理時間や，失敗のリスクといった要素も加味して管理者が対策を実行している．自動化を進めるためには，このように性能を含む多様な観点についてより高い精度で影響を見積もり，影響が最小限となる対策を確実に実施する必要がある．

以上に対し，PMSE 法を応用し，複数時点の性能情報だけでなく他のワークロードなどを考慮することで，性能劣化以外の二次的な影響を最小化することが可能である．これは，上記のクラウド環境における性能対策自動化を実現する上で必要な技術である．

6. おわりに

本稿では，任意の期間に発生した複数の性能低下事象を解消しつつ，対策を実施することに伴う二次的な性能劣化を最小限とする対策案を生成する方式である PMSE 法を提案した．本提案方式では，構成変更による影響を踏まえて対象の期間と同様のワークロードが発生した場合のリソースの負荷を推定し，その結果に基づいて各リソースの負荷が平準化され，かつ，二次的な性能劣化の発生回数が最小化されるように対策案を生成する．

PMSE 法を，実環境におけるディスクアクセス IOPS データを用いて評価した．評価の結果，1 時点のみを指定して対策案を生成した場合に 12%の確率で発生していた性能低下の再発を，適切な期間を指定することで 0 とできることを確認した．また，二次的な性能劣化を考慮せずに対策する場合に比べて，PMSE 法で生成した対策では対策実行後の各 Affected Resource の負荷を平均 12%減らすことができた．

PMSE 法を用いることで，二次的な性能低下の見積もりの手間がなくなり，かつ，より性能を安定させる対策を採ることが可能となった．現状では，対策としてワークロードの配置を変更する手段をとる場合のみを考慮しているが，今後はリソースを追加する手段などに対応していくことが課題である．これによって，二次的な性能劣化が発生する場合には，性能劣化を発生させないために必要なリソース量の提案などが可能となる．

参考文献

[1] Amazon Web Services, Inc.: Amazon EC2 Service Level Agreement, https://aws.amazon.com/ec2/sla.
[2] Lango, J.: Toward Software-defined SLAs, *Commun. ACM*, Vol. 57, No. 1, pp. 54–60 (online), DOI: 10.1145/2541883.2541894 (2014).
[3] Zabbix LLC.: Zabbix, http://www.zabbix.com/.

[4] NTT DATA Corp.: Hinemos, http://www.hinemos.info/.
[5] 崇之永井，正剛名倉：迅速な危機回復を目的とする大規模環境向け障害原因解析システム，情報処理学会論文誌，Vol. 54, No. 3, pp. 1109–1119（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009552598/⟩ (2013).
[6] VMware,Inc.: How To Troubleshoot vSphere 5.x Performance Issues Using vCenter Operations (2013).
[7] Zhang, N., Tatemura, J., Patel, J. and Hacigumus, H.: Re-evaluating Designs for Multi-tenant OLTP Workloads on SSD-based I/O Subsystems, *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, New York, NY, USA, ACM, pp. 1383–1394 (online), DOI: 10.1145/2588555.2588567 (2014).
[8] Laboratory for Advanced Software Systems, University of Massachusetts Amherst.: UMassTraceRepository, Storage OLTP Application I/O.