一般
投稿論文

テスト環境の開発アクティビティにおける大規模分散処理システムのシステム検証の効率化

梅田 昌義[†1]

[†1] 日本電信電話（株）

近年，ビッグデータの管理と処理を行うために，スケールアウトの容易な大規模分散処理システムの必要性が増してきている．本稿では，大規模分散処理システムの検証における2つの課題である「性能確認と機能確認の効率化」，「故障検知と故障対応の迅速化」に対して，「独自ベンチマークツールの機能の作成」，「故障マシンの入れ替え方法」の解決方法を提案する．これらの解決方法をシステム検証に適用した結果，検証の手戻りがなくなり効率的に検証を実施できたプラクティスについて述べる．

1. はじめに

近年，大規模分散処理システムによるビッグデータのさらなる活用が期待されている．大規模分散処理システムを実現する代表的なソフトウェアとしては，Apache プロジェクトの Hadoop[1]がある．Hadoop以外にもいくつかの大規模分散処理ソフトウェアが開発されており，筆者らがプロジェクトで用いた大量の非構造化データを管理する Key-Value Store システムである CBoC タイプ 2（Common IT Base over Cloud Computing タイプ 2）[2]もその1つである．商用システムのソフトウェア開発の場合，運用前の最終確認となるシステム全体の検証（以降，システム検証と呼ぶ）は，実運用時に発生する問題を事前に発見し改修するという点において重要であるが，Hadoop やCBoC タイプ2 のような大規模分散処理ソフトウェアは，その特性上，一般的なソフトウェアとは異なったシステム検証が求められる．

本稿では，検証の工程の分類をアクティビティに基づいて説明する．具体的には，IEEEで作成したソフトウェアエンジニアリングの基礎知識体系のSWEBOK[3]に記述されている図1の工程の分類に基づいて説明を行う．筆者らは大規模分散処理システムのシステム検証における計画とテストケース生成のアクティビティで発生した課題の解決方法について整理してきた[4][5]．本稿では，効率的に実施できる検証の環境を整える工程全体にかかわる作業である「テスト環境の開発」のアクティビティに着目する．

図 1　SWEBOK のアクティビティ

大規模分散処理システムのテスト環境は，マシン台数が多く検証対象のプロダクトが複数あり，扱うデータ量が大量であるため，効率よくかつコントロールができるように準備する必要がある．CBoCタイプ2では，マシンはIntelのCPU, DIMMのメモリ, SAS/SATAのハードディスクを搭載した汎用PCであり，その汎用PCを20台搭載したラックを複数ラック使って運用している．また扱うデータ量は100テラバイト級で大量である．このようなCBoCタイプ2のシステム検証における「テスト環境の開発」アクティビティにおいて，1カ月の期間を要する問題が発生した．それを分析した結果，検証期間の短縮と検証作業の効率化の2点の必要性が確認された．これらの問題の概要を表1に示す．

問題の1点目は，数多くのマシンがあるために，同時に性能確認や機能確認をする際に検証の同期合わせの手間と，集計するデータを決めて分散しているマシンから収集する手間とを要することである．この問題の

表 1　「テスト環境の開発」アクティビティで発生する問題の概要と具体的な課題

作業分類	発生した問題の概要	具体的な課題
検証実施	検証で実施する読み書きの性能確認と機能確認に手間を要する	性能確認と機能確認の効率化（課題 1）
環境確認	通信障害などマシン故障により機能確認や性能測定の結果が想定と異なる	故障検知と故障対応の迅速化（課題 2）

2017 ©Information Processing Society of Japan

具体的な課題は，性能確認と機能確認の効率的な実施である．

問題の2点目は，日々，マシンの故障が発生するため，検証結果にその影響がおよび，機能の動作や性能測定の測定値が要件と異なり，原因の調査や再試験など工数を要してしまうことである．この問題の具体的な課題は，マシン故障に伴う検証への影響を回避する，速やかな故障検知と故障対応である．

本稿は，商用の大規模分散処理システムのシステム検証におけるテスト環境の開発の課題を抽出・整理し，解決策を導き出したものである．本稿の構成を以下に示す．第2章では，関連する技術に基づき従来の手法を紹介し，第3，4章では，課題に対する実際のシステム検証において有効性を確認した実践的な解決方法を提案し，第5章では，まとめと今後の課題について述べる．

2. 関連技術

大規模分散処理システムにおける，システム検証の効率化という課題を，テスト環境の開発アクティビティの課題「性能確認と機能確認の効率化」，「故障検知と故障対応の迅速化」の観点から，それぞれについて関連する技術を示す．

(1)性能確認と機能確認の効率化に関する技術

システムの性能確認と機能確認を効率的に実施するツールとして，ベンチマークツール（Test Program）がある．一般的なツールとしてはApacheプロジェクトのHadoopのTestDFSIO，TeraSort，MRBench [6]やYahoo!が公開しているYCSB[7]があり，異なる大規模分散処理システム間での測定を実現している．しかし，これらのツールは，データサイズが小さい場合，データの条件を変化させた性能の測定ができない上，測定の中断からの再開の機能も持っていない．一方，一般的な情報システムのデータウェアハウス向けベンチマークであるTPC-Hを，大規模分散処理システム向けのHive上で動作するようにしたツール[8]がある．しかし，Apacheプロジェクトのプロダクトに特化しているため，ほかの大規模分散処理システムに対応するには，汎用化するための改造が必要である．

(2)故障検知と故障対応の迅速化に関する技術

現在の大規模なマシンの監視は，技術誌のツールの紹介記事[9]にあるように，Nagios，Hobbit，ZABBIX，Hinemos等の既存の監視OSSで運用されている．ただし，大規模分散処理システムで一部のプロセスが動作していないため，性能が出ない上，特定プロトコルの通信障害やマシンの性能劣化などは，既存の監視OSSの機能では監視ができない．故障対応においては，故障個所の交換方法として，故障交換，個別交換，一斉交換と分類し，その分類に故障分布をあてはめて，どの方式が効果的かを評価した研究[10]がある．しかし，交換作業自体をできるだけ短くする方法については記述がない．

以上のように，大規模分散処理システムのシステム検証における課題の解決方法は，オープンクエスチョンである．

3. 性能確認と機能確認の効率化（課題 1）

大規模分散処理システムの検証では，基本機能である書き込みや読み出し処理の確認と，これらの処理性能を測定する必要がある．たとえば，CBoCタイプ2は，クローラにより収集されたインターネット上のWebデータを蓄積し，Webデータを検索するサービスを提供するKey-Value Storeのシステムである．そのため，検証のユースケースとしてAP（Application Program）によるデータの読み書きには，シーケンシャルリード，ランダムライト，ランダムリードと種類があり，そのデータについてもテキストデータもあれば画像などのマルチメディアデータもある．これらの処理を実現し，さらに機能確認や問題解析の検証で必要な書き込むデータと書き込まれたデータの整合性の確認や，データサイズの変更などデータ条件を細かく変化させた測定ができるツールが必要となる．

CBoCタイプ2では，マシンを20台搭載したラックを複数使っており，マシン総数は数百台規模となる．この数百台のマシンに対して，全データ量が100テラバイト級となるデータに対する読み書きを一斉にコントロールしたり，機能確認やデータの整合性確認を自動で行う機能が求められる．さもなければ検証時間は長くなり，与えられた期間で実施するほかの作業を圧迫してしまう．

多くのマシンに対して一斉にデータの書き込み／読み出しを可能とし，システム検証で利用できるツールを表2に示す．

1点目の実アプリケーション（AP）は，商用で運用されるアプリケーションそのものである．商用のユースケースにおける処理を忠実に再現でき，機能を正確に確認することができる．しかし，インターネット上のデータを取得するには，グローバルなIPアドレスを利用し

たインターネット環境が必要となるなど利用に際しての制約がある場合がある．また，たとえば書き込みデータ量が対象となるサーバごとに異なるなど，性能検証時にデータ量がコントロールできない場合もある．

2点目の汎用ベンチマークツールは，OSSのベンチマークが主流であり，特定の条件の書き込みや読み出しの基本性能の把握ができる．また，OSS公開されているものは，利用者が多く，その利用者が測定した類似システムの結果が公開されているため，そうした類似システムの結果と比較して自システムの優劣の確認ができる．しかし，その結果は，特定条件の書き込みや読み出しの基本性能値しか取得できないため，実際のデータ特性を模擬した性能や，過負荷時や運用によるマシンメンテナンス作業など，商用での実利用を想定した性能値を取得することができない．さらに，問題発生時の解析に必要な機能は具備していない．

3点目の独自ベンチマークツールは，商用で運用されるAPの処理を模擬したTP（Test Program）であり，実利用を想定した性能要件や限界性能の特性を把握することができる．一般的な情報システムのシステム検証では，主に汎用ベンチマークツールと実アプリケーションを利用することが多い．しかし，大規模分散処理システムのシステム検証においては，汎用ベンチマークツールでは商用の実利用を想定した性能が取得できないこと，APでは環境の制約やデータ量のコントロールができないことから，これらを解決する独自ベンチマークツール（TP）の作成が必要となる．しかし，新たにTPを作成するには，開発リソースを計画に入れる必要があり，TP開発の負荷を見積もっておく必要がある．そこで作業見積もりをしたところ，TPの作成は，実アプリケーションを工夫して利用し，かつ取得できない性能値を手作業で確認するのに必要な期間（我々の経験では1カ月）よりも短い期間で作成可能であり，期間短縮の効果が期待できることが判明した．

本章では，商用のシステム検証で必要なベンチマークツールの機能を明らかにするため，CBoCタイプ2の性能測定兼システム検証用に独自開発した独自ベンチマークツール（ランダムライト，ランダムリード，シーケンシャルリード）の機能をどのように決定したか説明する．

表2　データの書き込み／読み出しのツール

	ツールの説明	主な用途
実アプリケーション	商用で運用される実アプリケーション（AP）そのものを用いる	ユースケース性能要件達成確認
汎用ベンチマークツール	OSS等のベンチマークツールを目的に合わせて取捨選択（TestDFSIO[☆1]，TeraSort[☆2]，MRBench[☆3]，YCSB[☆4]，TPC-H-Hive[☆5]など）	類似システムとの性能比較
独自ベンチマークツール	実利用を模擬したデータ発生用のテストプログラム（TP）を独自に開発	・ユースケース性能要件達成確認 ・限界性能特性の把握

☆1 Hadoop付属の分散ファイルシステムベンチマークツール
☆2 Hadoop付属の巨大ファイルソートプログラム，http://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html
☆3 Hadoop付属のMapReduceベンチマークツール
☆4 Yahoo!が公開しているNoSQL用ベンチマークツール
☆5 データウェアハウス向けベンチマークであるTPC-HをHive上で動作できるようにしたツール，https://github.com/rxin/TPC-H-Hive/tree/master/

3.1　独自ベンチマークツールの機能の検討

独自ベンチマークツール（TP）を作成する際に，以下の2つの案を検討した．

(1) ツール自身で書き込みのデータを作成し，書き込まれたデータを読み出すツールを作成する
(2) APで蓄積された実際のデータを別ディスクに保管し，そのデータを蓄積データとして書き込みしなおし，その蓄積されたデータを読み出すツールを作成する

この2つの案を比較したところ，(1)，(2) 両方ともツールを作成する工数を要する点は変わらない．しかし，(2) は蓄積されたデータが100テラバイト級であるため，別ディスクに保管するだけで数週間を要した．その上，蓄積するデータとして書き込みしなおすのに，さらに数週間を要した．(1) は (2) と比較してデータを作成する機能を新たに作成する工数を要するが，一度作成すればよく再利用ができる．以上より，(1)の案がよいと判断した．以降，(1) の案でTPを作成する場合に必要な機能について示す．

TPについては，APの処理を模擬し性能を測定するため，以下の機能を具備する必要がある．

A) 複数台のTPを集中制御できる仕組み
B) 書き込みデータの内容をパラメータ等で柔軟に変更できデータ内容が確認できる仕組み
C) スループットやデータのチェックができる仕組み

Aの機能は，TP数が数百におよぶ場合には1つ1つ手作業で起動や管理することは時間がかかるため，起動や停止を集中管理するコントロールサーバ上のTPの一括起動や停止をするツールを動作させることにより，各TPを一括管理する．また設定変更もコントロールサーバで実施する．この機能により，APの処理と同じように，書き込みや読み出しのTPの起動や終了タイミングを合わせることができ，APの模擬ができる．

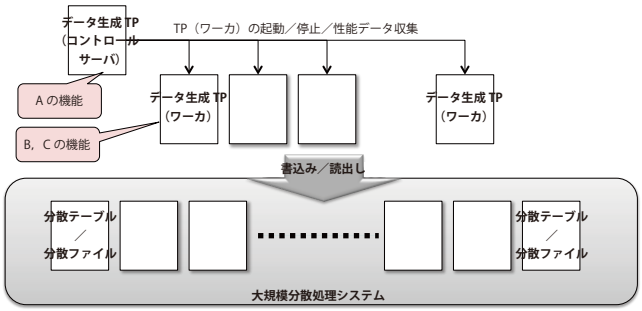

図 2　TP の構成と集中制御

Bの機能は，データの内容を固定ではなく変更できるようにすることにより，データの内容によるシステムの性能への影響を確認できる．またAPのデータ内容を模擬することで，実際のシステムを模擬した機能確認ができる．

Cの機能は，各TPのログ等からの結果の集計を仕組みとして持つことにより，数百台規模のマシンから時刻を合わせて集計した結果を容易に確認することができる．これにより，逐次性能を確認することができ，ログのメッセージや監視ツールの突合による問題発生時の即時対応ができる．

これらを実現すると，図2のように1カ所のコントロールサーバに配置したデータ生成TPから各ワーカに配置したTPを制御することができる．この考え方は，TPの作成というアプローチであり，APによる読み書きを実施するほかの大規模分散処理システムにも適用できる．次の節でA，B，Cの各機能の詳細について記述する．

3.2　検討結果からの具体的な解決方法

Aの機能については，機能として必要な「ランダムライト／ランダムリード」と「シーケンシャルリード」を独自ツールのTPとして作成し，集中制御した．「ランダムライト／ランダムリード」と「シーケンシャルリード」とを別々に作成するのは，APがそれぞれを別サービスとして処理するためで，その結果，それぞれが別プロセスで動作するからである．TPの構成は，図2のように1台のコントロールサーバと複数台の「ランダムライト／ランダムリード」と「シーケンシャルリード」の処理をするワーカで構成される．オペレーションはすべてコントロールサーバで行い，コントロールサーバから各ワーカに対して起動／停止／性能データ収集の制御を行った．

Bの機能について，書き込みデータは，表3に記載する「ランダムライト／ランダムリード」のTPを使ったデータ生成により自動生成した．生成したデータにおいて，検索されるキーとなるデータのRowKeyは，複数のサーバ／プロセス／スレッドによる書き込み／読み出しの処理が分散されるように，該当のサーバやプロセスの情報を用いたハッシュによる変換処理を行い算出した値で作成する．また，そのデータ自体は，Rowkeyで使用した該当のサーバやプロセスの情報からハッシュ算出した値で作成するとともに，データの書き込み時の圧縮機能により，指定した圧縮率で作成する．この手法により，生成されたデータの内容を把握した検証ができる．たとえばデータを書き込む際の異常系の検証では，書き込むデータと書き込まれたデータの整合性がチェックできる．「ランダムライト／ランダムリード」TPと「シーケンシャルリード」TPにおけるデータの書き込み／読み出しの管理の仕組みで工夫した点を以降に示す．これらの工夫は汎用ベンチマークツールでは実現されていない．

表 3　「ランダムライト／ランダムリード」の TP におけるデータ生成方式

生成項目	内容
RowKey	処理を行うサーバの IP アドレス，シーケンス番号，プロセス ID，スレッド ID からハッシュ算出
テキストデータ	処理を行うサーバの IP アドレス，シーケンス番号，プロセス ID，スレッド ID，世代番号（パラメータ指定）からハッシュ算出 圧縮機能により，データ書き込み時に指定する圧縮率になるようにデータを作成．圧縮率はパラメータとして指定
マルチメディアデータ	処理を行うサーバの IP アドレス，シーケンス番号，プロセス ID，スレッド ID，世代番号（パラメータ指定）からハッシュ算出

注：世代はデータの版数を意味する．

(1)「ランダムライト／ランダムリード」のTPのデータ生成に必要な項目を表4の上部に示す．データ書き込みの量を容易に増やしたり，データの生成バリエーションを増やすため，以下の機能が必要である．

①スレッド数／プロセス数やデータの送信間隔で読み込み／書き込みデータの発生量を調整

②書き込むデータサイズは一様ではなく正規分布やF分布，ポアソン分布のようにデータサイズを分布に基づいて書き込み

③書き込みを中断した場合に中断点からの再開（レジューム機能）

必要な項目は，書き込み／読み出しのデータとそのデータ生成を指定するための情報である，RowKeyサイズ，データサイズ分布，データサイズ，レジューム，およびスループット計測周期とした．これらの項目について，汎用ベンチマークツールである

YCSB，TestDFSIO，TeraSortではレジュームやスループットの計測周期を指定することができない．たとえば，詳細な分析が必要な場合でも計測周期は短くすることができず，途中で中断した場合は最初からの処理になりその時間が無駄になる．

(2)「シーケンシャルリード」TPのデータ生成に必要な項目を表5の上部に示す．必要な項目は，読み出しのカラムとそのカラムを指定するための情報である，検索対象となるデータのtimestamp，同時読み出しプロセス数とした．これらの項目について汎用ベンチマークツールでは読み出しカラムの指定，つまり，1カラムのみを指定して読み出しができない．

Cの機能は，スループットやデータ内容の確認ができるようにするもので，ログに必要な情報を記録する仕組みと，その記録を集計する仕組みからなる．複数のワーカに配置したTPで出力されたログをコントロールサーバで集計できるように作成することで作業の効率化を図った．「ランダムライト／ランダムリード」TPと「シーケンシャルリード」TPにおけるデータの書き込み／読み出しの計測の仕組みで工夫した点を以降に示す．

(1)「ランダムライト／ランダムリード」TPの性能を計測する仕組みとして必要となる主なログを表4の下部に示す．特に性能の特性を確認するために工夫した項目は，以下の2点である．

①各処理の平均処理時間に加え，最大／最小／分散等を出力

②レスポンスタイムがある閾値以上となったデータの個数を出力

表4 「ランダムライト／ランダムリード」TPで必要な項目と主な出力ログ

必要な項目	データ発生量（スレッド数，総件数）
	RowKeyサイズ
	データサイズ分布
	データサイズ
	レジューム
	スループット計測周期
主な出力ログ	ログ出力時刻
	アクセス数
	平均処理時間
	処理時間の分散
	処理時間の最大値
	処理時間の最小値
	スループット
	処理時間が閾値以上の件数
	読み出しデータなしの件数
	読み出しデータの不一致の件数

表5 「シーケンシャルリード」TPで必要な項目と主な出力ログ

必要な項目	読み出しカラム
	検索対象となるデータのtimestamp
	同時読み出しプロセス数
主な出力ログ	読み出し開始／終了時刻
	読み出しデータ量
	読み出し件数
	読み出しスループット
	読み出しArea（水平分割したテーブル）数

具体的な出力ログの項目は，ログ時刻，アクセス数，平均処理時間，処理時間の分散，処理時間の最大値，処理時間の最小値，処理時間のスループット，処理時間の閾値以上の件数，読み出しデータなしの件数，および読み出しデータの不一致の件数とした．

(2)「シーケンシャルリード」TPの性能を計測する仕組みには，読み出しによる結果の確認のためのログ出力が必要である．主なログを表5の下部に示す．「シーケンシャルリード」のTPについては，読み出し開始／終了時刻，読み出しデータ量，読み出し件数，読み出しスループット，および読み出しArea（水平分割したテーブルの一部）数を主な出力ログとした．

3.3 適用結果

この手法を適用した具体的な実例と結果について示す．作成したTPでは，APの利用による環境制約の考慮やデータ量コントロールの工夫が必要なく，図2に示したように1カ所のコントロールサーバに配置したデータ生成TPから各ワーカに配置したTPを制御して，機能検証と非機能検証ができた．特に，汎用ベンチマークツールでできなかった，書き込んだデータのチェックや，データの書き込みの中断や，データによる性能の変化を100テラバイト級のデータに対して確認できるようになった．この手法では，汎用ベンチマークで確認できなかった作業を確認するのに，これまで約1カ月要していたところを計画どおり1日で完了し，検証時間の短縮を図ることができた．

4. 故障検知と故障対応の迅速化（課題2）

一般的な情報システムのシステム検証に使われるマシ

ンは数台であり，故障発生の頻度は1年に数度である．そのため，故障対応をリアクティブに実施しても検証スケジュールに影響することがなく，問題は顕著化しなかった．それに比べ，大規模分散処理システムでは利用するマシンの台数が数百台規模となり，マシンあたり1年に1度の故障率でも日々マシンの故障が発生する状況となる．この大規模分散処理システムでは故障したマシンを切り離して運用する機能が備わっているものの，マシンの性能劣化や特定のプロトコルで接続できないなど，その機能に検知されずに故障したマシンが運用される場合がある．故障したマシンの放置は，故障したマシン台数分の処理性能が落ちるだけでなく，故障したマシンの処理がボトルネックとなって機能に影響を与えることもある．そのため，マシンの故障を検証前に検知して交換等の対応をし，検証への影響を回避する必要がある．仮想化技術を利用しないCBoCタイプ2は，マシン台数が多いため，効率的に実施しないと検証開始が遅れることになってしまう．

実際，データの書き込みにおいて，図3に示すとおり63時間経過した付近と73時間経過した付近で，応答時間が長くかかっている時間帯があった．性能測定の条件としては，数百台のマシンを利用し図2の構成でTPによりランダムライトを実施していた．データサイズはクローリングされたデータと条件を合わせ平均レコードサイズ5k Byteから40k Byteとし，分布も合わせていた．

調査は応答時間とマシンのリソースの観点で行った．その結果，次の2点が明らかとなった．

(1) 特定のマシンで応答時間が遅くなっていないか調査をした結果，特定の1台ではなく，複数台で遅い傾向があった．

(2) 各マシンのリソース状況を調査した結果，特定のサーバでI/Oが一時的に高くなっていた．

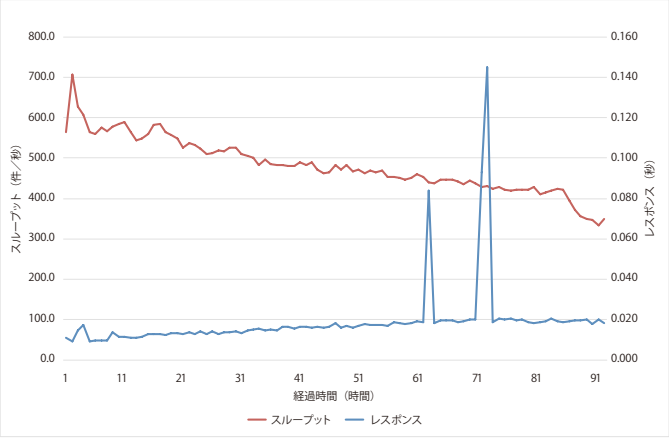

図3　書き込みデータのスループットと応答時間

これらの調査結果から，特定のサーバでI/Oの処理が一時的に遅くなり，それが全体的な応答時間の悪化につながったと考えられる．この調査には数週間を要した．

このように，検証に影響のある故障を事前に回避することが重要である．そこで，本章では商用のシステム検証で必要なマシン故障に対する検証への影響回避をどのように実現したか説明する．

4.1　故障マシンの入れ替え方法の検討

最初に故障マシンの入れ替えの方法について，表6に示すように3つの案を検討した．

1点目の案は，マシンの故障を発見した場合に，マシンの調達を行って調達できたマシンから故障機と入れ替えることである．この方法は，故障の都度にマシンを調達するので，費用は故障したマシンの分のみでよいが，調達や設置に時間を要する．当初この案で運用をしていたが，故障から復旧に2～3週間程度を要しその間の検証が止まってしまうためほかの案に替えた．商用の運用時であれば，システムが稼働していれば対応ができるが，マシン台数を揃えた機能検証や性能検証は対応ができない．

2点目の案は，マシンの故障の頻度は考慮せず，利用するマシンのバックアップとしてもう一式マシンを用意することである．この方法は，マシンの故障を発見時には，バックアップマシンとすぐに切り替えができ故障から復旧までの時間は30分程度ですむ．しかし，利用する倍の台数のバックアップマシンが必要となるため，利用するマシンの倍の費用が必要になる．

3点目の案は，これまでの故障の実績を基に，これから故障するマシンの台数を割り出し，そのマシンの台数を予備機として用意し，マシンが故障した場合に，その予備機と切り替えることである．この方法は，費用としては故障する頻度が少ないときには，余分に費やしてしまい，故障する頻度が多いときには，1点目の案と同様にマシンの調達を行う必要がある．しかし，故障する頻度が予備機の数と合っている場合は，最少の費用と故障から復旧までの最短時間で故障したマシンの入れ替えが

表6　故障マシンの入れ替え方法

案	解決方法	費用	復旧期間
1	故障の都度にマシンを入れ替える	○	3週間
2	運用の倍のマシンを用意して使い捨てにする	×	30分
3	一定の割合の予備機を用意して入れ替える	△	30分

できる．

以上から1点目から3点目の中では，3点目の案がよいと判断した．以降，3点目の故障機の入れ替え案をもとに説明する．

解決の考え方としては，監視ツールによる故障監視と，予備マシンの用意による故障マシンの入れ替えで対応する．故障率はこれまでの運用の記録からある程度範囲がわかるため，それを元に想定した故障マシン数と同程度の予備マシンを準備しておく．これにより，検証に影響がないよう速やかに故障マシンを切り離し予備マシンに入れ替えることができる．また，通常のマシン監視に加え，それだけでは発見できないシステム独自で使用している特定プロトコルの故障を速やかに検知する．

この解決の考え方は，故障の即時検知というアプローチであり，マシン台数が数百台規模の場合，大規模分散処理システム一般に適用できる．

4.2 検討結果からの具体的な解決方法

具体的な解決方法としては，既存の監視OSSであるNagiosとMRTG[11]，Crane[12]に加えて，シェルスクリプトによるツールとCronにより故障を検知する手法を取った．表7に示すとおり，監視OSSによる監視は，一般的なシステムの監視内容であるCPUの負荷といったCPUの状態，メモリの使用率といったメモリの状態，ディスクI/Oやディスク容量といったディスク状態や，NWの不具合等の状態を数値の異常により監視した．加えて，利用した監視OSSでは，syslog等のシステムログに出力されたエラーメッセージだけでなく，追加の設定により大規模分散処理システムのエラーメッセージも監視した．

スクリプトやCronのツールでは，大規模分散処理システムで特有の内容である，ログメッセージやプロセス監視，特定プロトコルの疎通確認のほかに，運用に必要となる統計情報の確認を行った．スクリプトやCronのツールを定期的に実行することにより，ログメッセージやプロセスの監視や特定プロトコルの疎通確認を自動化し，問題を検知した場合，即時に監視画面等で通知した．

表7 監視概要

監視内容	監視方法
ログメッセージ	監視 OSS
プロセス状態	監視 OSS
特定プロトコル疎通確認	ツール
統計情報	ツール

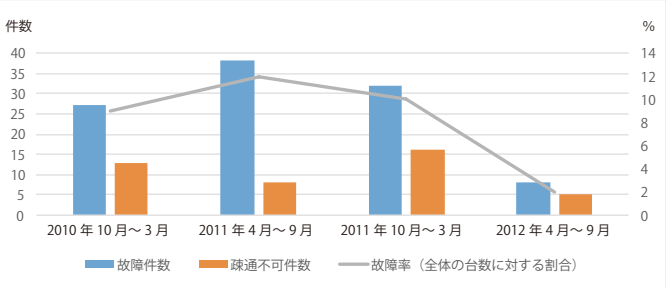

図4 適用した期間の故障の実例

商用システムのCBoCタイプ2の検証では，20台のマシンを積んだラックを15ラック利用していた．図4に2年間の故障の実例を示す．故障の発生には，当初は「ある検証を実施すると発生する」，「あるプロダクトのマシンで発生する」等の仮説を立てたが，特定の作業による故障の傾向は見られなかった．一方で，故障したマシンは再度故障するという傾向が見られた．また，特定プロトコルで接続できない状態のマシンは，1カ月あたり数%の割合で発生していた．この状況から，故障率とラックに搭載できるマシン数を考慮し，予備マシン数は全体台数の1割とした．

4.3 適用結果

商用システムであるCBoCタイプ2の検証用のマシン環境に，提案方法を適用した．予備マシンがないときはマシン調整の実施から行うため，約1カ月要していたが，提案手法を適用した結果，予備マシンが準備済みであることにより，入れ替え作業のみの短時間で復旧ができた．また，OSSによる通常の監視システムでは故障の対象にならず発見が遅れることが多い，大規模分散処理システムに特有な特定プロトコルでの接続不良などの故障にも即時対応ができ，手戻りのない検証の実現ができた．

5. まとめと今後の課題

本稿では，大規模分散処理システムのシステム検証に対して，「テスト環境の開発」アクティビティに基づく2つの技術的課題として整理し，効率的に実施できる解決方法を提案・実践してその有効性を確認した．

提案した方法は，TPの集中制御，故障検知と故障マシン入れ替えというアプローチであり，汎用のPCサーバで構成した大規模分散処理システム一般に適用できる．ただし，TPのリアルタイムの設定変更，故障の傾向の分析はできておらず，これらは今後の課題である．

課題の調査により，大きく効率化が期待できる対策と

適用結果が出た段階で，ほかの実行，問題報告／テストログ，欠陥追跡のアクティビティについて順次報告する予定である．

参考文献

1) 西田圭介 : Google を支える技術〜巨大システムの内側の世界，技術評論社 (2008).
2) 鷲坂光一，中村英児，高倉　健，吉田　悟，冨田清次：大量データ分析のための大規模分散処理基盤の開発，NTT 技術ジャーナル，Vol.23, No.10, pp.22-25 (2011).
3) 松本吉弘（監訳）：ソフトウェアエンジニアリング 基礎知識体系 SWEBOK V3.0, オーム社 (2014).
4) 坂井俊之，梅田昌義，中村英児，本庄利森：大規模分散処理システムのソフトウェア試験とその実践，情報処理学会デジタルプラクティス，Vol.4, No.1, pp.51-59 (2013).
5) 梅田昌義，鬼塚　真：計画アクティビティにおける大規模分散処理システムのシステム検証の効率化，デジタルプラクティス，Vol.7, No.3, pp.330-339 (2016).
6) White, T.，玉川竜司（訳）：Hadoop 第 2 版，オライリー・ジャパン (2011).
7) Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.：Benchmarking Cloud Serving Systems with YCSB, Proceedings of the 1st ACM symposium on Cloud computing, New York (2010).
8) Running TPC-H queries on Hive, https://issues.apache.org/jira/browse/HIVE-600
9) 寺島広大：オープンソース統合監視ツール導入指南，http://thinkit.co.jp/free/article/0706/21/1/，Think IT (2007).
10) Barlow, R., and Proschan, F.：Mathematical Theory of Reliability, 84, John Wiley & Sons, Inc. (1965).
11) Kennedy, T.：Tobi Oetiker's MRTG - The Multi Router Traffic Grapher, http://oss.oetiker.ch/mrtg/ (2011).
12) 林　憲亨，佐藤　麦：クラウド環境における運用管理ソリューション，NTT 技術ジャーナル，Vol.23, No.8, pp.19-24 (2011).

梅田 昌義（正会員）umeda.masayoshi@lab.ntt.co.jp
NTT ソフトウェアイノベーションセンタ第二推進プロジェクト主任研究員．1991 年電気通信大学電気通信学部情報数理工学科卒業．同年，日本電信電話（株）入社．現在，大規模分散処理システムの研究開発に従事．

投稿受付 : 2017 年 1 月 30 日
採録決定 : 2017 年 6 月 23 日
編集担当 : 峯　恒憲 (九州大学)

アンケートにご協力ください　https://www.ipsj.or.jp/15dp/enquete/enq_dp0804.html