IT agility. Your way.®

Empowered by Innovation NEC

ゼロ・ダウンタイムを実現する
高可用アプリケーションサーバシステムの構築

CLUSTERPRO × BIG-IP® Local Traffic Manager

連携検証レポート

2012年7月

日本電気株式会社
F5 ネットワークスジャパン株式会社

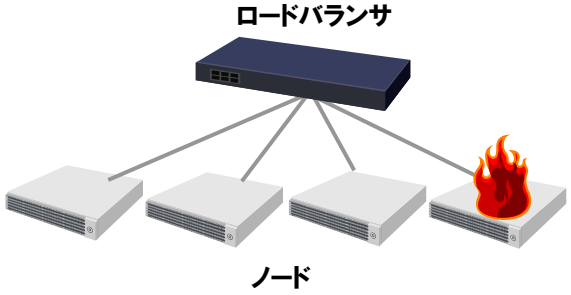
目次

1．　はじめに

1. はじめに

1.1 背景

一般的にロードバランサを用いることで、ユーザリクエストを複数のノードに分散させて、負荷を平準化させるとともに、障害が発生したノードの処理を別のノードで引き継ぐことでサービスの可用性を高めることができる。しかし、ロードバランサだけでは、アプリケーションレベルで発生する予期せぬ問題には対応できないため、システムダウンに陥る可能性がある。



ロードバランサ単独での課題

- アプリケーションレベルの負荷状況は考慮せずにリクエストを送信
- サーバの異常が検知されるまで、継続してリクエストを送信

システムダウンを防止し、『ゼロ・ダウンタイム』を実現するには？

ポイント ①
アプリケーションレベルで高可用性を確保
→ JavaベースのアプリケーションにはJava VMの監視で安定稼働を実現

ポイント ②
ロードバランサとアプリケーションサーバの密な連携
→ アプリケーションの稼働状況に応じた連携制御で安定稼働を実現

1. はじめに

1.2 目的

本検証では、BIG-IP Local Traffic Manager（以下 BIG-IP LTM）と CLUSTERPRO X との連携により、各ノードでアプリケーションの障害（※）の予兆を検出し、事前に予防措置を行うことで『ゼロ・ダウンタイム』を実現するアプリケーションサーバシステムの有効性の確認を目的とする。

※ 主にJava VMの障害

【連携概要】

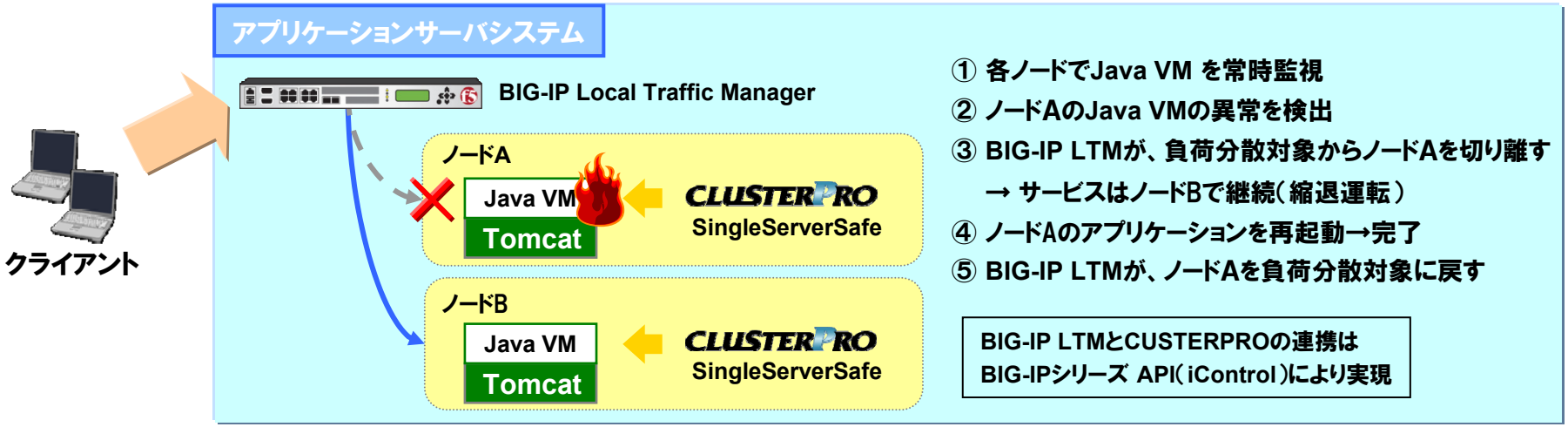
アプリケーションサーバシステムを構築する際に、以下の製品を連携させることでシステム全体の可用性を向上させる。

BIG-IP Local Traffic Manager

アプリケーションを理解したインテリジェントな配信を実現するアプリケーション・デリバリ・コントローラ
サービスを止めないきわめて高い信頼性を実現する製品

CLUSTERPRO X SingleServerSafe 3.1

サーバのHW/SW監視により高可用性を実現する製品



1. はじめに

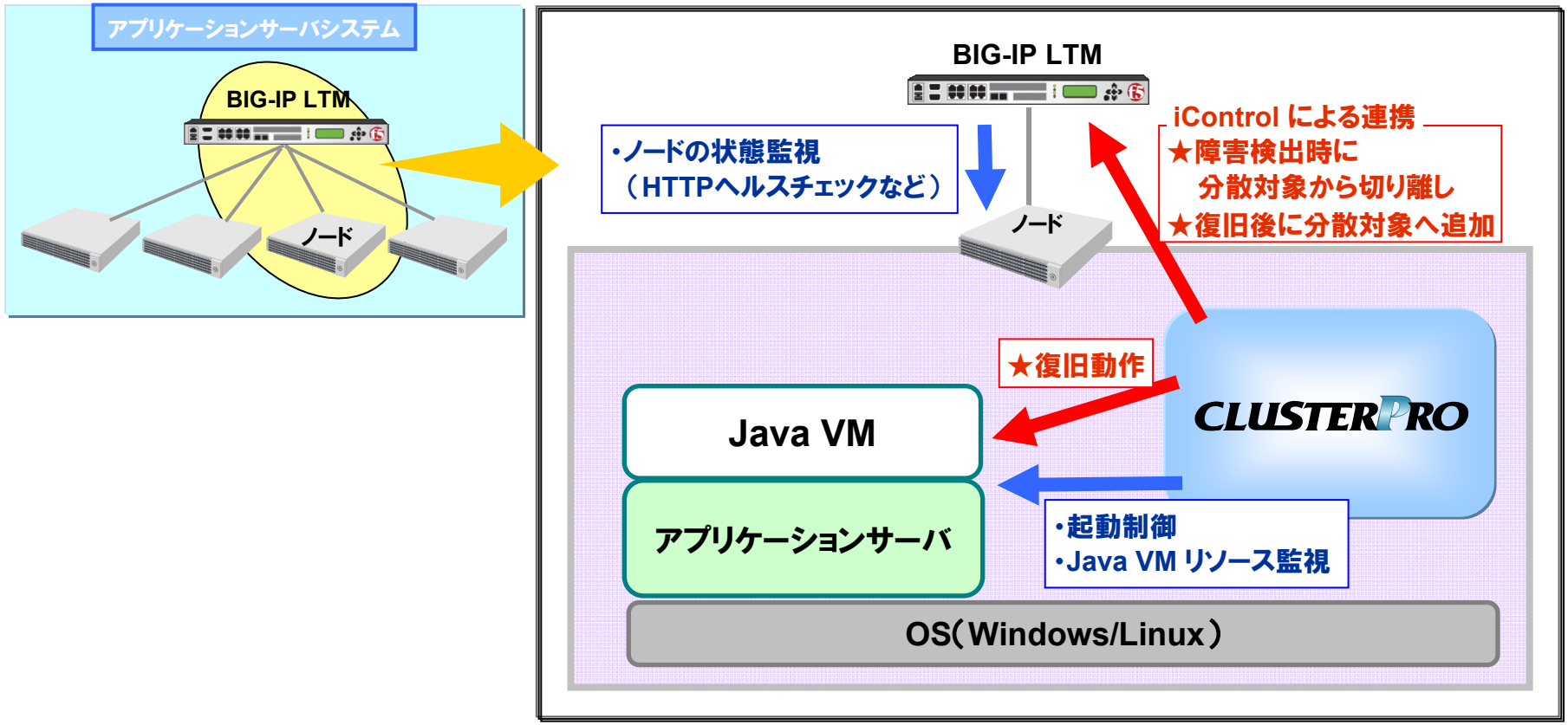
連携により期待される主要な効果

製品 / 主要効果	BIG-IP Local Traffic Manager		BIG-IP Local Traffic Manager × CLUSTERPRO X	
高負荷による障害（メモリ不足など）の予防	○	•HTTPヘルスチェックにより障害発生の検知が可能 •障害が発生したノードを負荷分散対象から切り離すことが可能	◎	•アプリケーション（Java VM）の状態を監視し、障害発生前に負荷分散対象から切り離すことにより、問題のあるノードへのリクエストを他ノードへ分散することが可能
レスポンス悪化への有効性		•スループット計測により、レスポンス悪化の検知が可能 •応答速度が速いノードで処理させることが可能		•ノードの負荷がアプリケーションレベルで高負荷な場合、予防措置として負荷分散対象から切り離し、安定した他ノードに分散させることで、レスポンス悪化への影響を抑えることが可能
				•レスポンス悪化したノードを自動復旧することで元の安定した状態に戻すことが可能
アプリケーションのデッドロック（プログラム異常）発生への有効性		•デッドロック発生により応答が返せなくなると、HTTPヘルスチェックにより障害発生の検知が可能 •障害が発生したノードを負荷分散対象から切り離すことが可能		•アプリケーション（Java VM）の状態を監視し、デッドロック発生を検知する。それにより、応答待ちで滞留しているリクエストをいち早く解放し、エラー通知をすることでクライアント側のエラー処理を迅速に実行することが可能
				•デッドロックが発生したノードを自動復旧することで元の安定した状態に戻すことが可能
通報/管理		•管理画面からの状態の確認が可能 •SNMPトラップ、syslogによる通知が可能	◎	•CLUSTERPROから通報メールを送信することで、システム管理者が問題発生時に迅速に認識することが可能。 •任意の復旧アクションを実行可能

1. はじめに

連携概要①

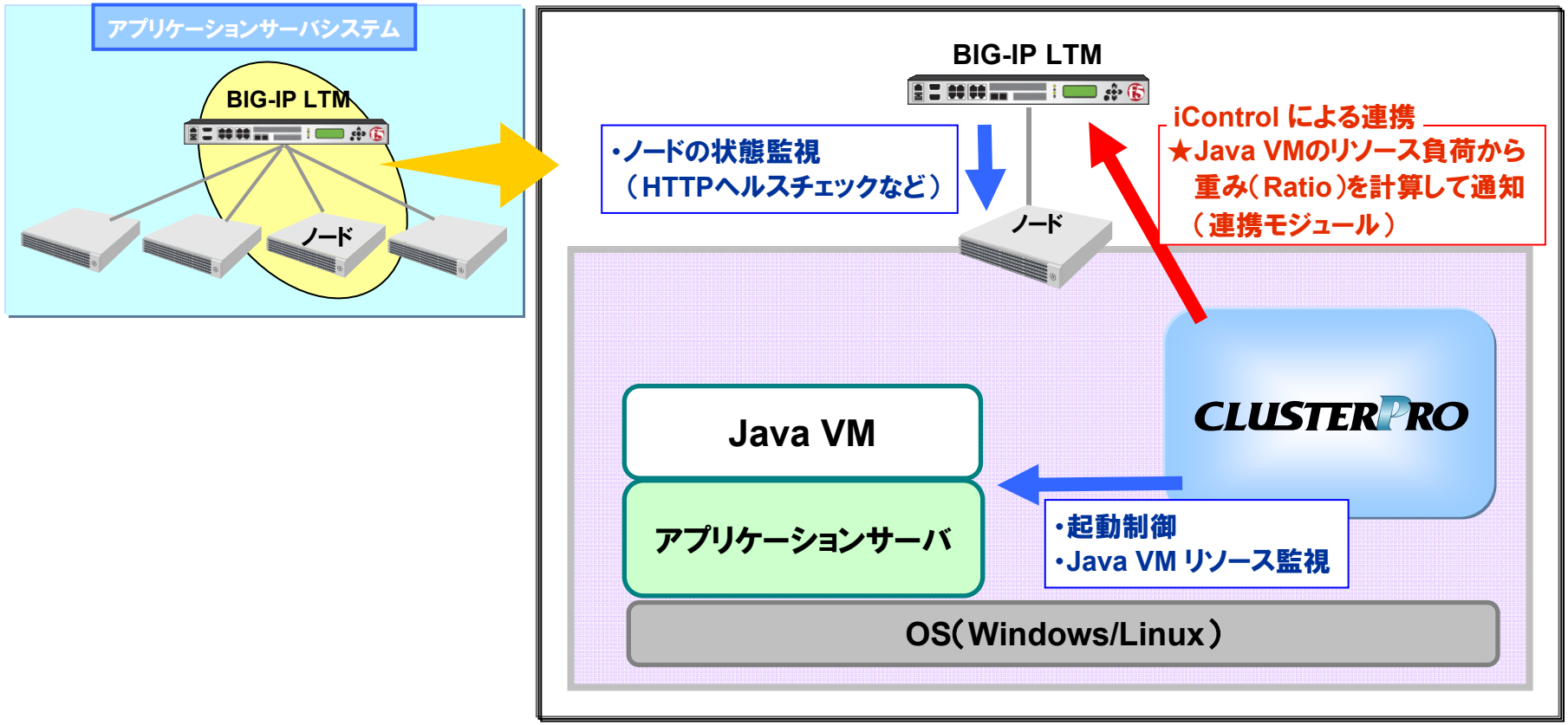
- Java Resource Agent にて障害予兆を検知し、事前対処を実施
  ⇒ ノードダウンを防止し、システムのレスポンスを維持



1. はじめに

連携概要②

- アプリケーションサーバ（Java VM）の負荷に応じた負荷分散を実施
  ⇒ 負荷の偏りを防止し、ノードダウンのリスクを軽減



1. はじめに

1.3 検証の観点

以下の観点で検証を実施

本資料での章

➢アプリケーションの高負荷による障害(メモリ不足など)の予防
- Java VMの監視により障害を予兆し、対象のノードを分散対象から切り離した後、自動復旧することができるか
- BIG-IP LTMと連携することでシステムとしてのダウンタイムをゼロにできるか

→ 3.1の検証

➢レスポンス悪化への有効性
- GC(ガベージコレクション、メモリ解放処理)の頻発によるレスポンス悪化を検出後、障害発生ノードを切り離し、自動復旧することができるか
- BIG-IP LTMと連携することでシステムとしてのダウンタイムをゼロにできるか

→ 3.2の検証

➢アプリケーションのデッドロック(プログラム異常)発生への有効性
- アプリケーションのデッドロックを検出後、障害発生ノードを切り離し、自動復旧することができるか
- BIG-IP LTMと連携することでシステムとしてのダウンタイムをゼロにできるか

→ 3.3の検証

➢通報/管理(上記3つの観点に共通)
- 管理画面・通報メールにより管理者は状況を認識できるか
- チューニングなどの根本解決に有効な情報を入手できるか

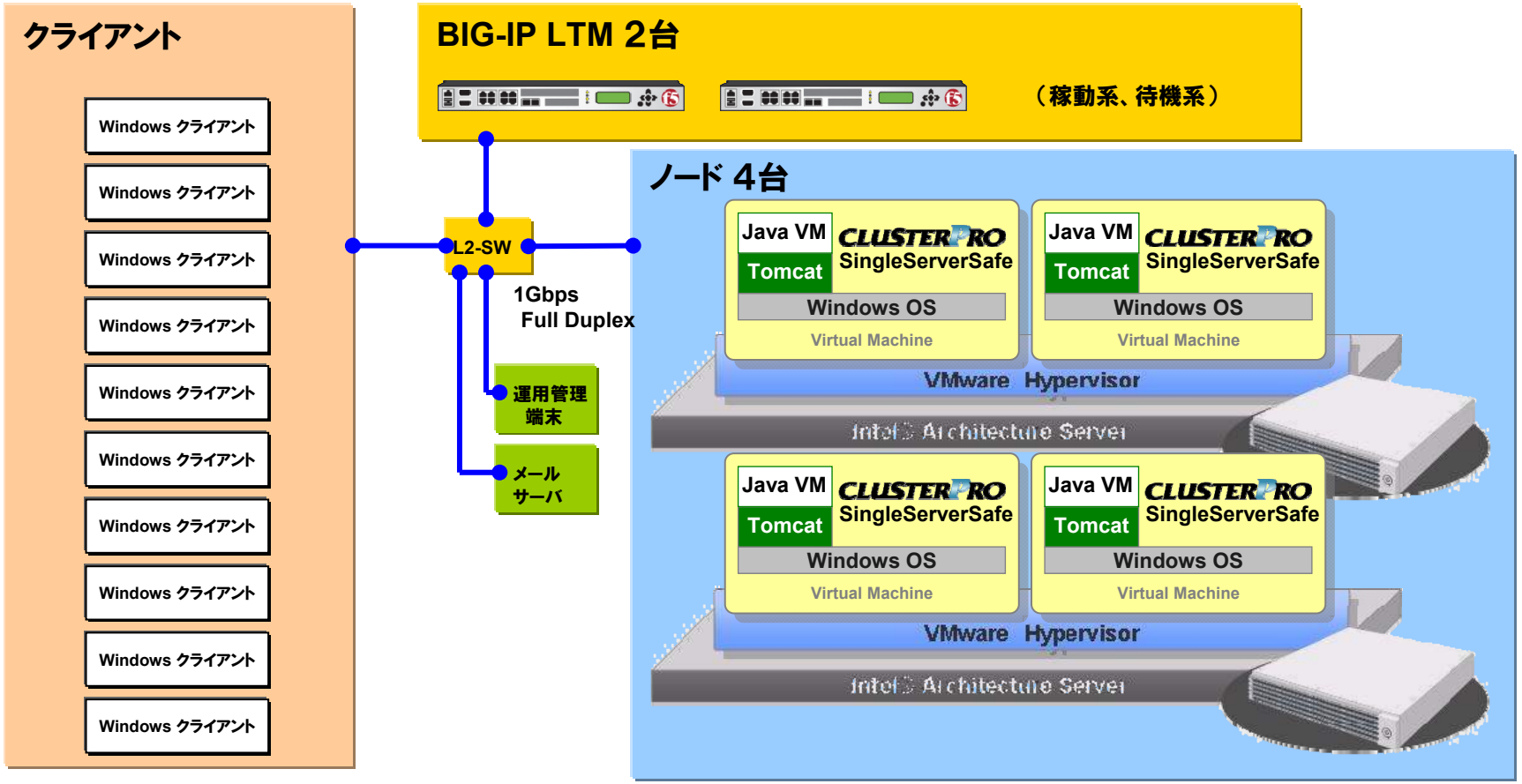
→ 3.4の検証

2.　検証環境

2. 検証環境

2.1 ネットワーク構成

以下に、ネットワーク構成図を示す。



※実運用では各種PC、端末など
様々なクライアントが想定される。

※各ノードはVMware上に仮想マシンとして構築。
実運用では物理・仮想環境は特に問わない。

2. 検証環境

2.2 BIG-IP LTM構成

本検証で使用した BIG-IP LTM は下記の通りである。
なお、稼動系、待機系の2台とも共通の構成となる。

ハードウェア	BIG-IP Local Traffic Manager 3900
ソフトウェアバージョン	10.2.3
ロードバランシング方式	Ratio(node)

2.3 アプリケーションサーバ構成

本検証で使用したアプリケーションサーバ（以下 APサーバ）は下記の通りである。
なお、ノードの構成は4台(APサーバ#1/#2/#3/#4)とも共通の構成となる。

OS	Windows Server 2008 R2
Javaランタイム	JDK 6 Update 31
アプリケーションサーバ	Tomcat 6.0
高可用性ソフトウェア	CLUSTERPRO X SingleServerSafe 3.1 CLUSTERPRO X Java Resource Agent 3.1（オプション製品） CLUSTERPRO X Alert Service 3.1（オプション製品） ※本検証で用いる連携モジュールについては、お問い合わせください。

2. 検証環境

2.4 クライアント・検証ツール構成

本検証で使用したクライアント・検証ツールは下記の通りである。
なお、10台とも共通の構成となる。

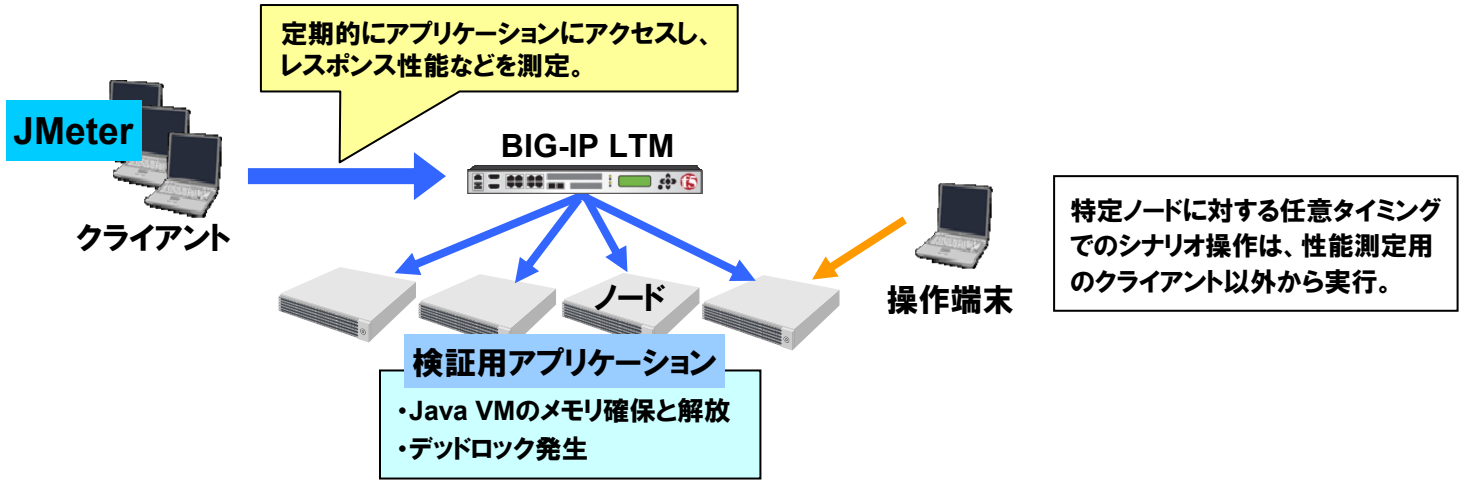
OS	Windows XP
Javaランタイム	JDK 6 Update 31
検証ツール	Apache JMeter 2.3.4

検証ツールは、検証シナリオに基づいたクライアントからAPサーバへの定期的なアクセスを実行し、
アクセス結果やレスポンス性能などのデータ測定をするために使用する。

2. 検証環境

本検証の動作シナリオは下記の通りである。
検証用アプリケーションはTomcat の Java VM 上に配備し、クライアントから Web ブラウザを経てアクセスする。
アクセスによって Tomcat の Java VM のリソースが消費される。

検証項目	動作シナリオ	確認する観点
1	Webアクセスの増加により、Web コンテナ上で巨大なオブジェクトが多数生成され、Java VM のメモリプール領域の使用量が上昇する。	・高負荷による障害の予防（3.1） ・レスポンス悪化への有効性（3.2）
2	アプリケーションの不具合によりデッドロックが発生する。	・デッドロック発生への有効性（3.3）



3.　検証/結果

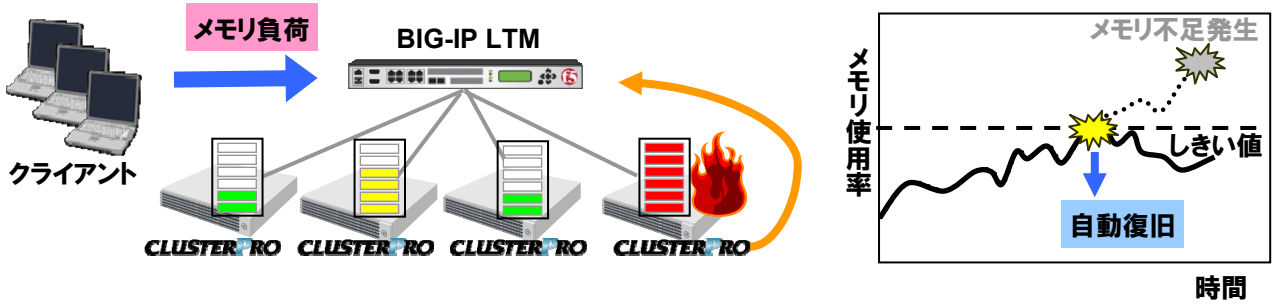
3. 検証/結果

3.1 高負荷による障害の予防

3.1.1 検証

課題

HTTPヘルスチェックにより障害発生を検知し、障害発生ノードを負荷分散対象から切り離すことが可能であるが、障害検知されるまでは障害発生ノードにリクエストが分散されてしまう。
メモリ高負荷状態でのリクエストによりメモリ不足が発生してしまうと、ユーザリクエスト処理異常となってしまう。



検証内容

クライアントからBIG-IP LTM経由で検証用アプリケーションに定期的にアクセスすることで、各APサーバのメモリの使用率を上昇させていく。
BIG-IP LTMのみの環境と、BIG-IP LTM と CLUSTERPRO を連携させたときの環境で検証(クライアントからのアクセスのレスポンスタイム計測)を行い、連携機能の効果を確認する。

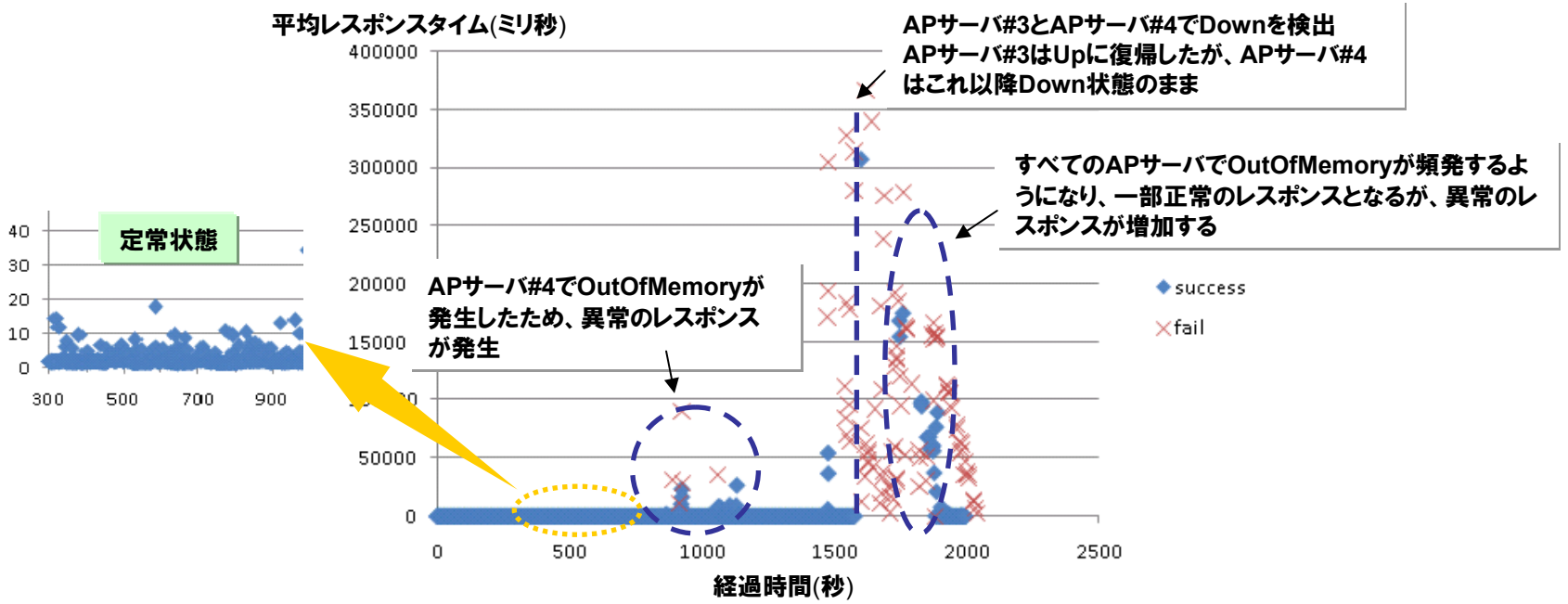
検証手順

(1) BIG-IP LTM のみの環境(連携機能未使用状態)で、負荷をかけ、障害を発生させる。
　①現象を発生しやすくするために、JMeterを用いて、APサーバ4台のJavaヒープメモリを約 50% 消費。
　②JMeterを用いて、BIG-IP LTM経由で、APサーバ4台のJavaヒープメモリの占有と解放を行うアクセスを繰り返し、
　結果的にJavaヒープメモリ使用率が上がっていく状況を作り出す。
　③Javaヒープ枯渇を経て、OutOfMemory が発生するまでアクセスしつづける。
(2) BIG-IP LTM と CLUSTERPRO を使用した状態(連携機能使用状態)で、手順(1)と同じ負荷をかける。
　手順(1)と異なり、自動復旧が動作するので、その経過を記録する。
(3) 手順(1)と(2)でレスポンスタイムと経過時間を軸にしたグラフを作成する。

3. 検証/結果

3.1.2 BIG-IP LTMのみの場合

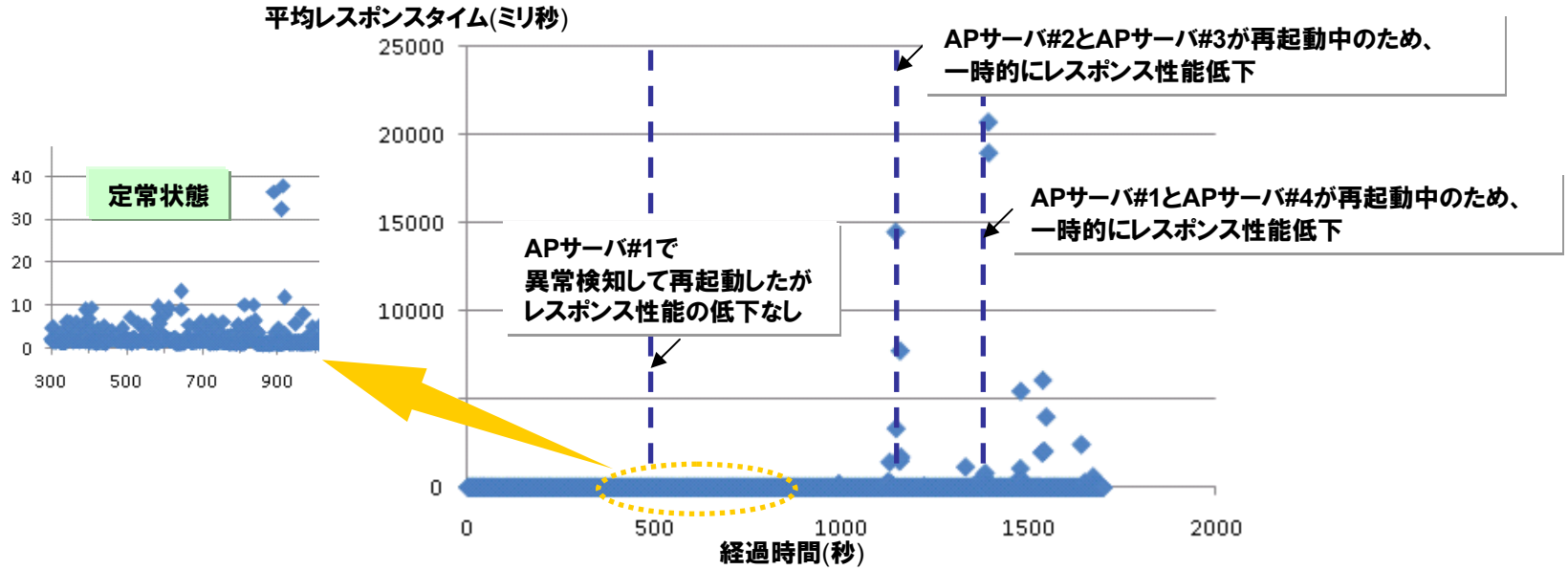
- 900 秒頃から1台のノードでメモリ不足（OutOfMemory）が発生し、レスポンスタイムが悪化。HTTPヘルスチェックのDown検知により障害発生ノードが分散対象から切り離されたことで、大きなレスポンス悪化にはならず。
- 1000 秒頃に同じノードでOutOfMemoryが発生し、これ以降は分散対象から切り離された状態となる。残り3台での負荷分散となるが、しばらくはレスポンス安定。
- 1500 秒頃に2台目のノードでOutOfMemoryが発生。残り2台のノードにアクセス集中することになり、メモリ負荷が加速し、レスポンスの悪化とあわせてリクエストエラーが頻発。2000秒頃に全てのノードでOutOfMemoryが発生し、システムダウンの状態となる。



3. 検証/結果

3.1.3 BIG-IP LTMとCLUSTERPRO を連携させた場合

- 500秒頃に1台のノードでJavaヒープメモリ使用率がしきい値の80%を超過。レスポンス悪化前に負荷分散対象から切り離して復旧動作（APサーバの再起動）を実行したことで、**レスポンス遅延は見られなかった**。
- 1200秒頃と1400秒頃に一時的なレスポンス遅延が見られたが、1台のノードで復旧動作中に他のノードがしきい値超過を検出したタイミングであった。一時的に2台のノードが切り離される状態となるが、復旧動作完了後に負荷分散対象に組み込まれるため、**継続したレスポンス遅延は見られなかった**。
- しきい値超過を検出すると、BIG-IP LTMのノードのステータスをdisableに変更して継続のHTTPリクエストが割り振られないようにしたことで、リクエストエラーの発生はなく、**全てのリクエストが正常に処理**されたことを確認。



3. 検証/結果

3.1.4 考察

3.1.2、3.1.3 **より観点毎の結果は以下のようになった。**

Java VMの監視により障害を予兆し、対象のノードを分散対象から切り離した後、自動復旧することができるか

- メモリ不足（OutOfMemory）の発生よりも早い段階でメモリの高負荷状態を検出し、APサーバの再起動により自動復旧することが可能。
- クライアント側からのアクセスを全て正常に処理することが可能。

BIG-IP LTM と連携することでシステムとしてのダウンタイムをゼロにできるか

- メモリ高負荷状態を検出した段階で、障害発生ノードへのリクエストの分散を停止し、負荷上昇にともなうレスポンス異常を抑止可能。
- 障害復旧時、APサーバの再起動において処理が完了するまで待機することで、既存のコネクションを中断することなく（※）復旧可能。

※ 既存のコネクションを中断することなく復旧動作を実行可能であるのは、BIG-IP LTM連携の優位性
※ BIG-IP LTMとAPサーバのコネクション数が0になるまで、APサーバの再起動を待機

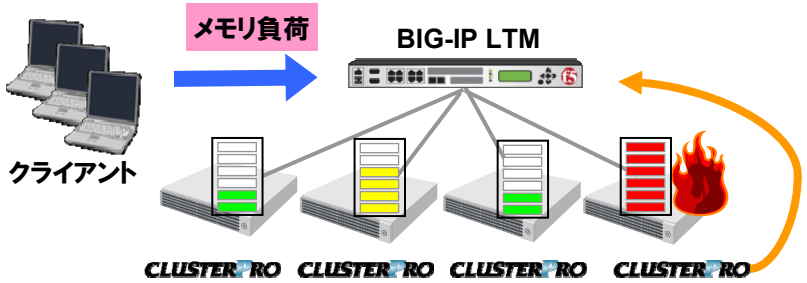
3. 検証/結果

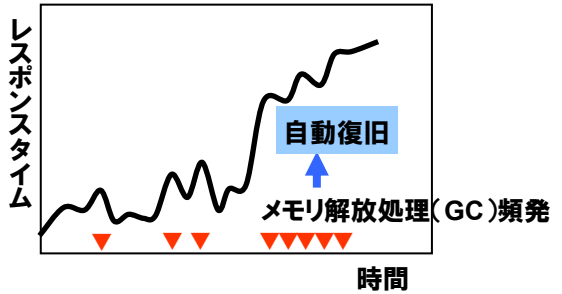
3.2レスポンス悪化への有効性

3.2.1 検証

課題

HTTPヘルスチェックにより障害発生を検知し、障害発生ノードを負荷分散対象から切り離すことが可能であるが、障害検知されるまでは障害発生ノードにリクエストが分散されてしまう。
GC(ガベージコレクション、メモリ解放処理)の頻発の影響でリクエスト遅延が発生してしまう。





検証内容

クライアントからBIG-IP LTM経由で検証用アプリケーションに定期的にアクセスすることで、各APサーバのメモリの使用率を上昇させて、メモリ解放処理(ガベージコレクション 以下GC)が頻発する状態にする。
BIG-IP LTMのみの環境と、BIG-IP LTM と CLUSTERPROを連携させたときの環境で検証(クライアントからのアクセスのレスポンスタイム計測)を行い、連携機能の効果を確認する。

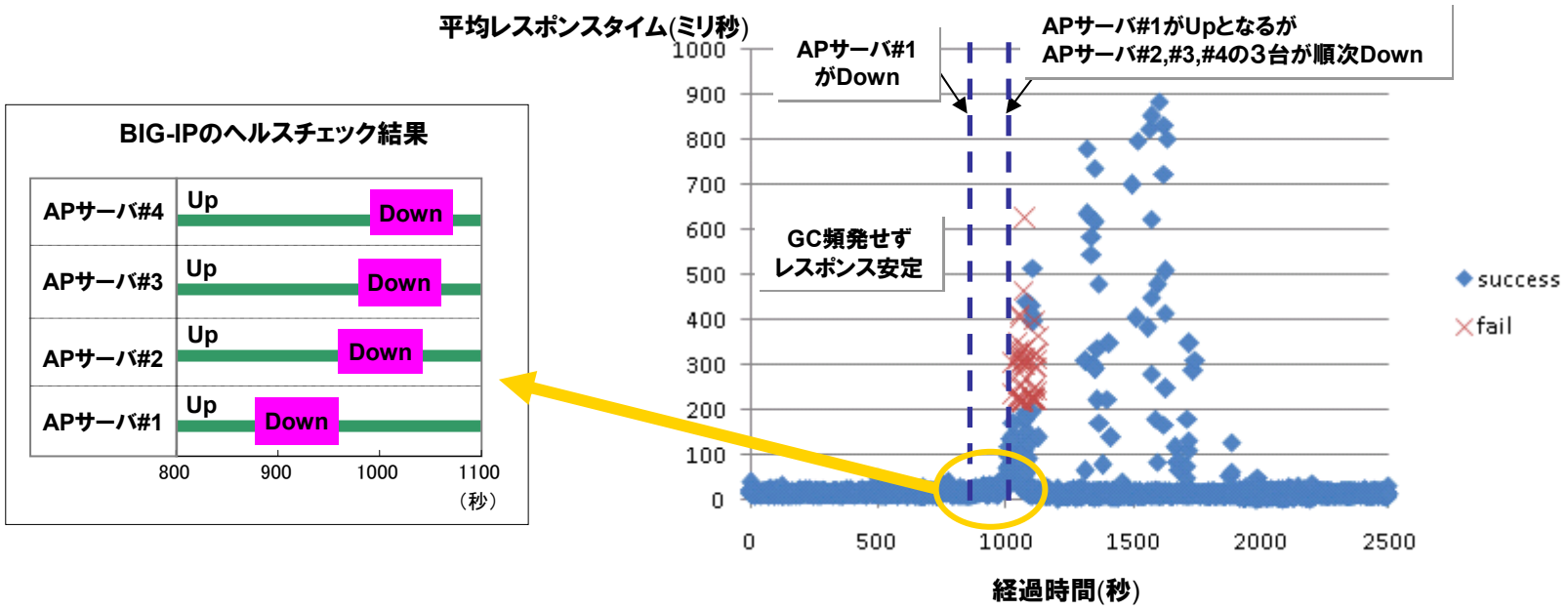
検証手順

(1) BIG-IP LTM のみの環境(連携機能未使用状態)で、負荷をかけ、障害を発生させる。
①現象を発生しやすくするために、JMeterを用いて、APサーバ4台のJavaヒープメモリを約50%ほど消費。
②JMeterを用いて、BIG-IP LTM 経由で、APサーバ4台のJavaヒープメモリの占有と解放を行うアクセスを繰り返し、
結果的にJavaヒープメモリ使用率が上がっていく状況を作り出す。
③GC連続発生、FullGC連続発生の状態となり、レスポンス悪化が発生する。
(2) BIG-IP LTM と CLUSTERPRO を使用した状態(連携機能使用状態)で、手順(1)と同じ負荷をかける。
手順(1)と異なり、自動復旧が動作するので、その経過を記録する。
なお、本検証では、Javaヒープ枯渇判定機能のしきい値を高く設定し、限界まで自動復旧が行われないようにする。
(3) 手順(1)と(2)でレスポンスタイムと経過時間を軸にしたグラフを作成する。

3. 検証/結果

3.2.2 BIG-IP LTMのみの場合

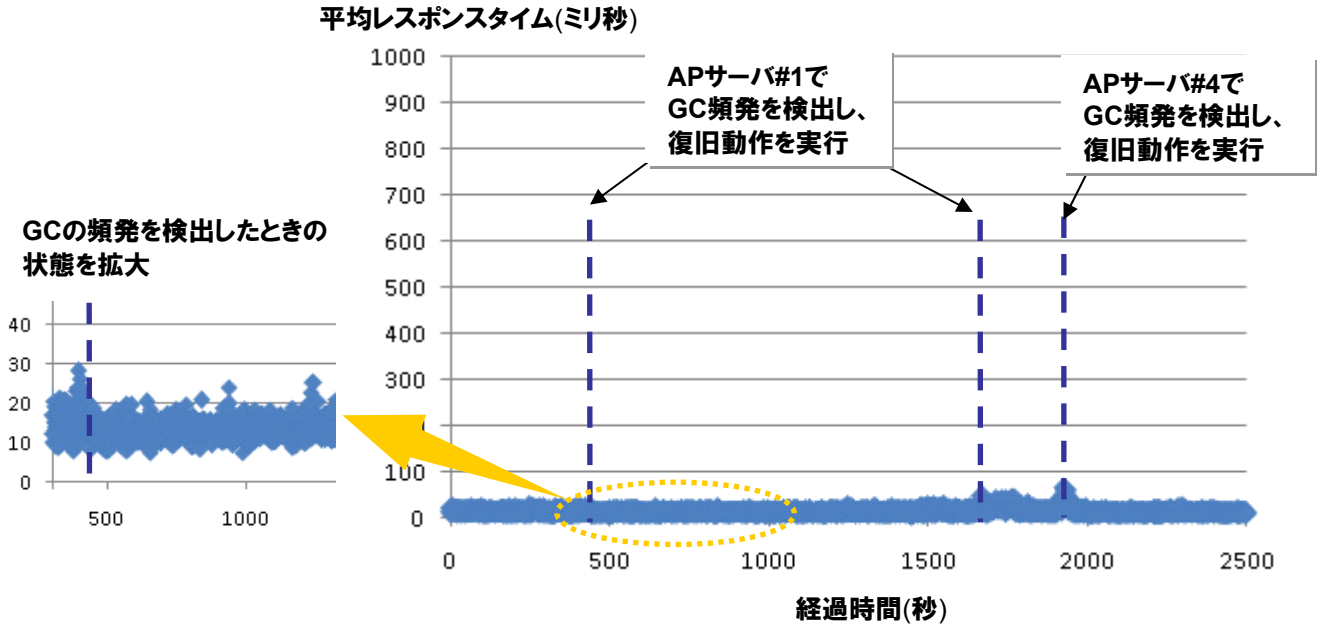
- 900秒頃から、1台のノードでメモリ解放処理（以下 GC）が頻発し、HTTPヘルスチェックで異常（Down）が検出されたが、残り3台のノードで正常にリクエスト継続できレスポンス低下は発生しなかった。
- 1000秒頃に3台のノードでGCが頻発し、HTTPヘルスチェックで異常が検出された。
  レスポンス遅延が発生するとともに、一部のリクエストのレスポンスがエラーとなった。
- 1100秒頃から、全てのノードでHTTPヘルスチェックが正常となり、一時的にレスポンス安定状態となった。
- 1300秒頃からGCが頻発しはじめたが、HTTPヘルスチェックでの異常は検出されなかった。GC頻発の高負荷状態がしばらく継続したため、レスポンスに時間が掛かるようになった。



3. 検証/結果

3.2.3 BIG-IP LTMとCLUSTERPRO を連携させた場合

- 500秒頃、1700秒頃にノード（APサーバ#1）でメモリ解放処理（以下 GC）の頻発を検出するが、負荷分散対象から切り離した後に復旧動作（APサーバの再起動）を実行したことで、レスポンス遅延は見られなかった。
- 1900秒頃にノード（APサーバ#1）の復旧動作中にノード（APサーバ#4）でGCの頻発を検出するが、レスポンス遅延は見られなかった。
- GCの頻発を検出してレスポンス悪化する前に復旧動作を実行したことで、全体を通して安定したレスポンスタイムを実現。



3. 検証/結果

3.2.4 考察

3.2.2、3.2.3 **より観点毎の結果は以下のようになった。**

メモリ解放処理(GC)の頻発によるレスポンス悪化状態を検出後、障害ノードを切り離し、自動復旧することができるか	
◎	•レスポンス遅延が発生する前のメモリ解放処理(GC)の頻発を検出した時点で、アプリケーションサーバ(以下 APサーバ)の再起動により自動復旧することが可能。 •クライアント側からの全てのアクセスを正常かつ安定したレスポンスタイムで処理することが可能。

BIG-IP LTM と連携することでシステムとしてのダウンタイムをゼロにできるか	
◎	•BIG-IP LTM のみの構成でもレスポンスタイム計測した負荷分散方式でレスポンス遅延の予防が可能であるが、CLUSTERPRO と連携することで異常状態のノードを自動復旧でき、ダウンタイムゼロを実現することが可能。 •レスポンスタイム遅延を検出した段階で、APサーバへのリクエスト割り振りを停止し、負荷上昇にともなうレスポンス悪化を抑止可能。 •障害復旧時のAPサーバの再起動は該当APサーバでの処理が完了するまで待機。これにより、既存のコネクションを中断することなく(※)復旧可能。 •※ 既存のコネクションを中断することなく復旧動作を実行可能であるのは、BIG-IP LTM連携の優位性 ※ BIG-IP LTMとAPサーバのコネクション数が0になるまで、APサーバの再起動を待機

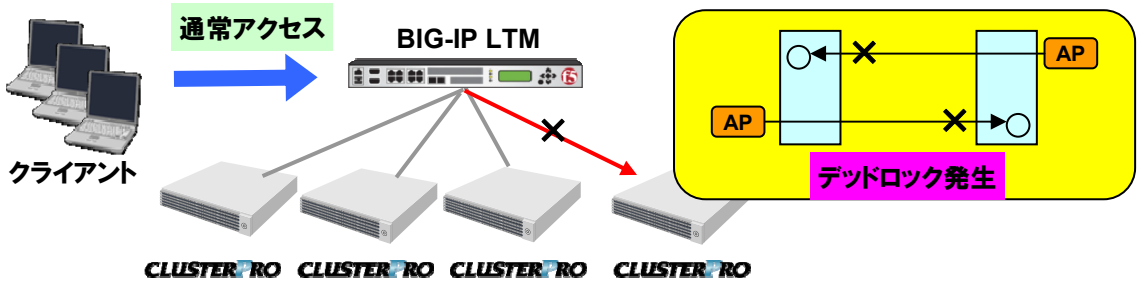
3. 検証/結果

3.3デッドロック発生への有効性

3.3.1 検証

課題

アプリケーションでデッドロックが発生したことをHTTPヘルスチェックでは検知できず、クライアントからのリクエストのタイムアウトになるまでは、異常を検出できない。



検証内容

本検証ではレスポンスタイムの測定ではなく、単位時間当たりに処理されたリクエスト数を測定し、システム全体の性能状況を確認する。
クライアントからBIG-IP LTM経由で検証用アプリケーションに定期的にアクセスし、アクセス結果を記録する。
各アプリケーションサーバに対して順番にデッドロックを発生させ、クライアントからのアクセスのリクエスト数の推移を確認し、連携機能の効果を確認する。

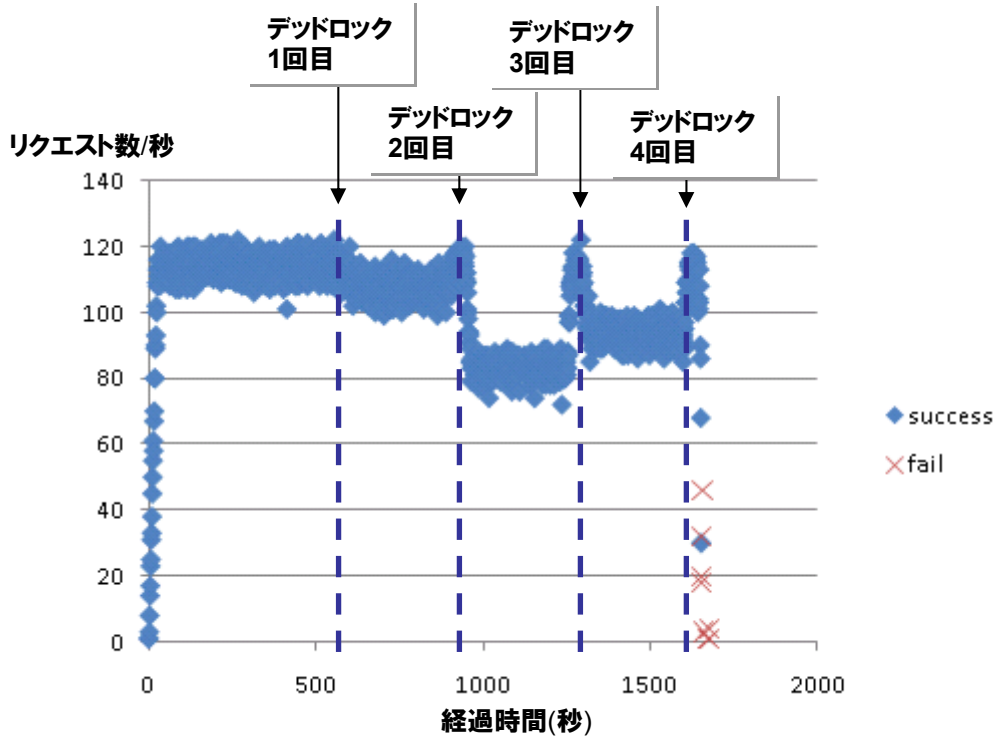
検証手順

(1) BIG-IP LTM のみの環境(連携機能未使用状態)で、負荷をかけ、障害を発生させる。
①JMeterを用いて、BIG-IP LTM経由で、APサーバ4台のJavaヒープメモリの占有と解放を行うアクセスを繰り返す。
(メモリを消費させるためではなく、レスポンスタイムを安定させるため。)
②JMeterを用いて、BIG-IP LTM経由で、APサーバ4台へアクセスを繰り返す。
(負荷をかけるためではなく、定期的にアクセスできるかを記録するため。)
③手動で、デッドロックが発生するアクセスを行う。
④6分に一度、合計4回、③のアクセスを行い、定期的なアクセスの経過を記録する。
(2) BIG-IP LTM と CLUSTERPRO を使用した状態(連携機能使用状態)で、手順(1)と同じアクセスを行う。
手順(1)と異なり、自動復旧が動作するので、その経過を記録する。
(3) 手順(1)と(2)でリクエスト数／秒と経過時間を軸にしたグラフを作成する。

3. 検証/結果

3.3.2 BIG-IP LTMのみの場合

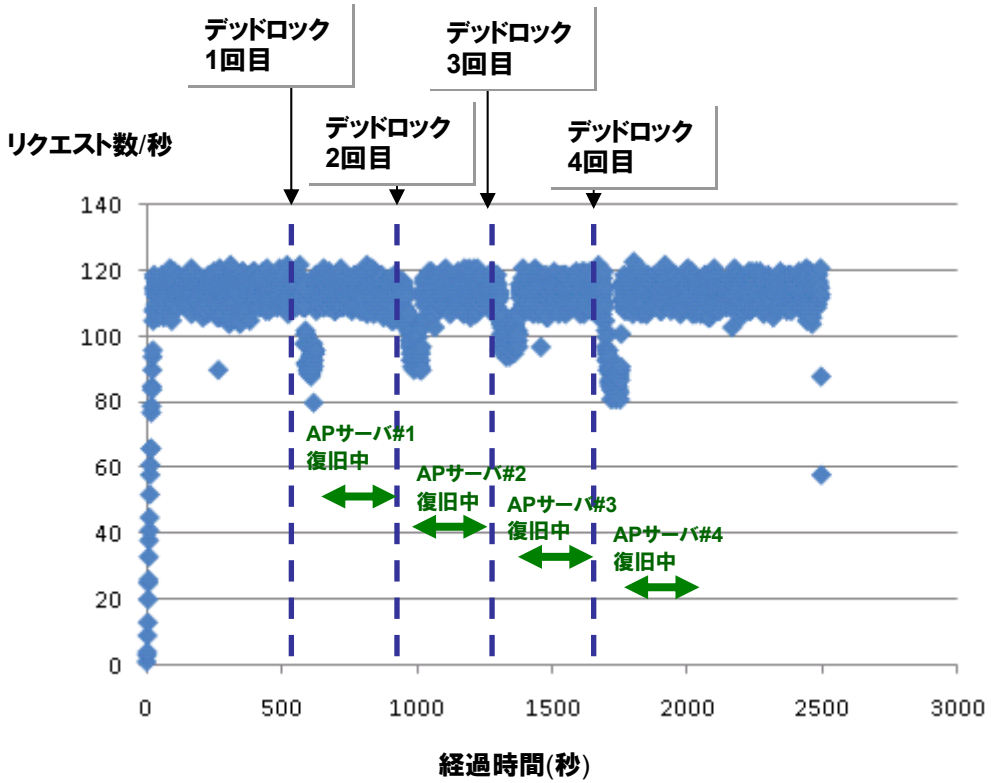
- デッドロックの発生したスレッドはレスポンス応答が不可となり、デッドロックが発生していないスレッドでリクエストを実行する。このため、**単位時間当たりのリクエスト数は低下。**
- デッドロック発生のタイミングでリクエスト要求したクライアントのセッションは、しばらく処理待ちで滞留するが、タイムアウト後にリクエストを再開できるようになる。2回目以降のデッドロック発生前に、処理待ちになっていたクライアントのセッションが全てタイムアウトしたため、定常状態のリクエスト数／秒に復帰する。
- デッドロック発生時のリクエスト数／秒の低下幅はタイミング依存であり一定ではない。
- 1600秒頃に全てのAPサーバでデッドロック状態になり検証不可能となった。



3. 検証/結果

3.3.3 BIG-IP LTMとCLUSTERPRO と連携した場合

- デッドロック発生時に単位時間当たりのリクエスト処理数は一時的に低下するが、すぐに定常状態に復帰する。
- BIG-IP LTMのみの場合と比較して、処理待ちになっていたリクエストは、APサーバの再起動により解放されるため、デッドロック発生から定常状態に戻るまでの期間が短縮した。
- デッドロック発生したノードは、負荷分散対象から切り離して復旧動作を行うため、システム全体への影響を極小化できた。



3. 検証/結果

3.3.4 考察

3.3.2、3.3.3 より観点毎の結果は以下のようになった。

アプリケーションのデッドロックを検出後、障害発生ノードを切り離し、自動復旧することができるか
◎
•デッドロックの検出を契機としたAPサーバの自動復旧が可能。 •デッドロックにより応答待ちで滞留しているリクエストをいち早く解放でき、クライアント側のエラー処理を迅速に実行することが可能。

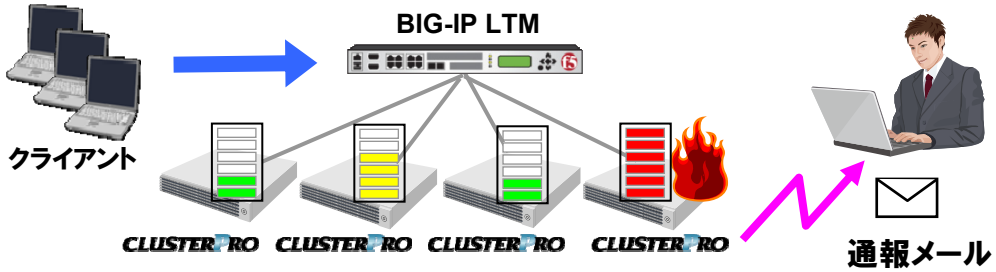
BIG-IP LTM と連携することでシステムとしてのダウンタイムをゼロにできるか
◎
•デッドロック発生を検出した時点で、APサーバへのリクエスト割り振りを停止し、早期に自動復旧を行うことで、システム全体のダウンを防止し、ダウンタイムゼロを実現することが可能。 •障害復旧時のAPサーバの再起動において、該当APサーバでの処理が完了するまで待機することで、既存のコネクションを中断することなく(※)復旧可能。 •※ 既存のコネクションを中断することなく復旧動作を実行可能であるのは、BIG-IP LTM連携の優位性 ※ BIG-IP LTMとAPサーバのコネクション数が0になるまで、APサーバの再起動を待機

3. 検証/結果

3.4 通報/管理

3.4.1 検証

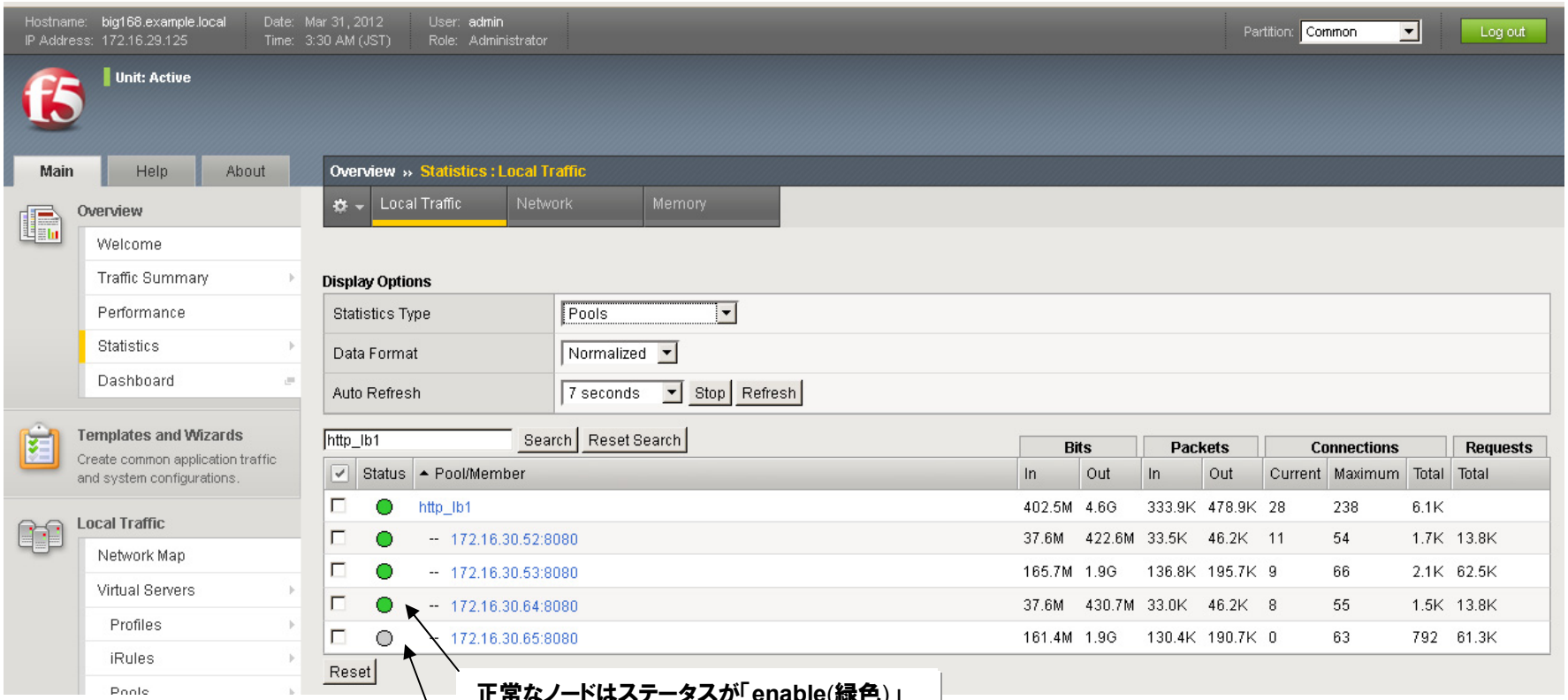
3.1～3.3までの検証を行う中で、各製品の管理画面、通報メールなどを通じて、
管理者に有効な情報が届いていたかを確認する。



3. 検証/結果

3.4.2 BIG-IP LTM による通知機能

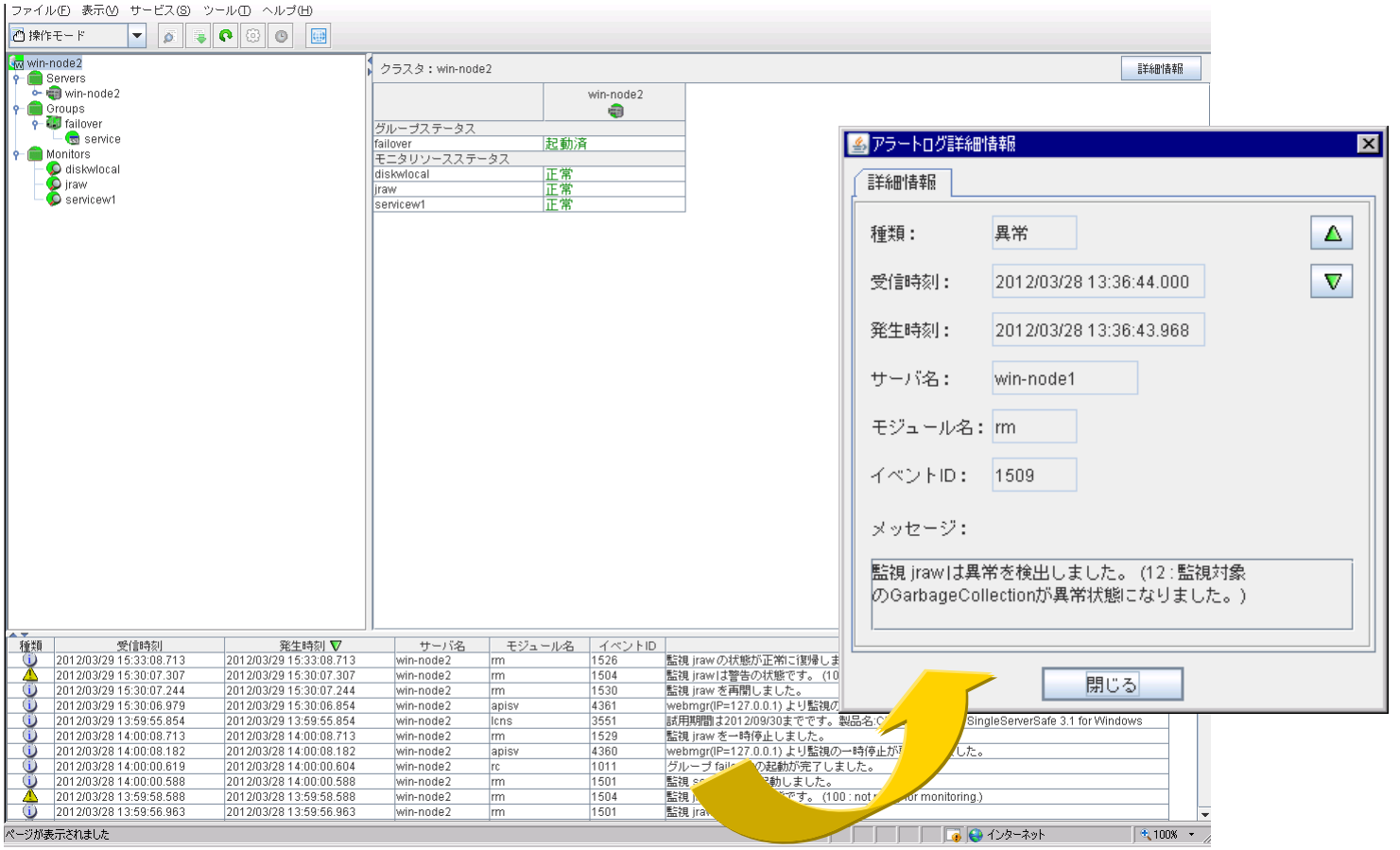
- BIG-IP LTM では、管理画面で各ノードの状態を確認できる



3. 検証/結果

3.4.3 CLUSTERPRO による通知機能

- CLUSTERPRO では、WebManager でアプリケーション（Java VM）の状態を確認できる



3. 検証/結果

3.4.3 CLUSTERPRO による通知機能

- CLUSTERPRO では、異常の発生をメールで通知することができる

メモリヒープ枯渇の場合

デッドロック検出の場合

GC 頻発の場合

メール通知機能を利用するには、
オプション製品（CLUSTERPRO X Alert Service）が必要

3. 検証/結果

3.4.4 考察

3.4.2、3.4.3 より観点毎の結果は以下のようになった。

管理画面・通報メールにより管理者は状況を認識できるか	
◎	•BIG-IP LTM、CLUSTERPRO共に管理画面・通報メールによって、管理者が状況を判断することが可能。 •BIG-IP LTM のヘルスチェックで異常を検知したノードと、CLUSTERPROで異常を検知したノード（負荷分散対象から除外されたノード）は、区別して判断することが可能。 •各ノードの状態を確認するには BIG-IP LTM の管理画面、各アプリケーション（Java VM）の状態を確認するには CLUSTERPRO WebManager、メール通知が有効である。

4.　まとめ

4. まとめ

本検証により、BIG-IP LTMとCLUSTERPRO Xの連携が、『ゼロ・ダウンタイム』を実現するアプリケーションサーバシステムにおいて有効であることを証明することができた。

本資料での章

3.1の検証	➢アプリケーション（Java VM）の状態を監視し、障害発生前に負荷分散対象から切り離すことにより、リクエストを常に健全なノードに分散することが可能。
3.2の検証	➢高負荷状態のアプリケーションの予防措置として負荷分散対象から切り離し、システムを常に健全なノードで構成することで、レスポンス悪化への影響を抑えることが可能。
3.3の検証	➢アプリケーション（Java VM）の状態を監視し、デッドロックの発生を検出し、応答待ちで滞留しているリクエストをいち早く解放し、エラー通知することでクライアント側のエラー処理を迅速に実行することが可能。
3.4の検証	➢管理画面・通報メールにより、管理者はリアルタイムな状況の認識が可能。 また、統計情報やログなどから、原因究明につながる有効な情報を入手可能。

5. お問い合わせ先

検証内容の詳細については下記までお問い合わせください。

CLUSTERPRO製品

- 製品ご紹介サイト　http://www.nec.co.jp/clusterpro/
- お問い合わせ先　日本電気株式会社
  info@clusterpro.jp.nec.com

BIG-IP製品

- 製品ご紹介サイト　http://www.f5networks.co.jp/
- お問い合わせ先　F5 ネットワークスジャパン株式会社
  F5 First Contact（F5 ファーストコンタクト）
  http://www.f5networks.co.jp/fc/

NECグループビジョン2017

人と地球にやさしい情報社会を
イノベーションで実現する
グローバルリーディングカンパニー

Empowered by Innovation

NEC