特別研究報告

題目

ベイジアンネットワークとクラスタリング手法を用いた Web 障害検知システムの開発

指導教員
井上 克郎 教授

報告者
爲岡 啓

平成 26 年 2 月 14 日

大阪大学 基礎工学部 情報科学科

平成 25 年度 特別研究報告

ベイジアンネットワークとクラスタリング手法を用いた Web 障害検知システムの開発

爲岡 啓

内容梗概

社会基盤としての Web システムは，不特定多数のクライアントを抱える重要な役割を果たしており，安定した長期稼働が要求される．しかし，今後システムの複雑化，クライアントの増加によって，システムに障害が起こる可能性は高まっていくことが予想される．そのような障害を未然に防止する手段として，障害を，その発生よりも前に検知することが考えられる．

Web システムの障害検知に関しては，少数の計測メトリクスを用いて，管理者が異常かどうか，経験に基づいて判断するという形が一般的であり，ソフトウェアによる体系的な検知は行われていない．

そこで，本研究では，早期に障害を発見するための検知システムを開発した．具体的には，複数のメトリクスを入力とし，ベイジアンネットワークおよびクラスタリングという，2 つのデータ分析技術を用いて分析した上で，システムに異常が起こる確率を算出する手法を考案した．実際に Web 上で稼働するサーバ・クライアントシステムを仮想計算機上で実現し，その稼働記録を対象として，検知システムの評価を行った．評価の結果，本システムを用いた障害検知が有効であることを示した．

主な用語

Web システム
障害検知
ベイジアンネットワーク
クラスタリング

目 次

1　はじめに

社会基盤としてのWebシステムは，コンピュータ資源やサービスを一括で管理しており，安定した長期稼働が要求される．しかし，クライアントの増加，システムの複雑化によって，システムに障害が起こる可能性は高まっている．たとえば，障害の原因として，クライアントからサーバへのアクセス過多や，サイバー攻撃などが挙げられる．Webシステムの障害が起これば，そのクライアント数や扱うデータ量に伴い，大きな被害が見込まれる．また，このような障害は時間，場面を問わず起こりうるものであるから，その障害をいかに早く発見し，対処するかということが，Webシステムの長期安定稼働に不可欠である．

ハードウェアの障害検知に関しては，物理的な変化に対して障害を検知し，障害が起きた際には，その系を切り替えることでシステムの稼働を維持する，という方法をとっている．一方で，ソフトウェアの障害検知においては，システム管理者が，サーバマシンのCPUやメモリの利用率など，少数の計測メトリクスの推移を見てシステムに一定時間内に不具合が発生するかどうかを経験的に判断することが一般的である．

一方で，システムが稼働した際に生成される種々の動作状況を示すデータ量は膨大である．また，不具合が発生する際には，個々のメトリクスが複雑に連携していることが考えられる．したがって，そのようなデータを人間が処理し，不具合が起こるかどうかという結論を導き出すことは非常に困難である [8]．また，人間が限られた時間内に閲覧できる一部のデータを基に行う判断は推測の域を出ず，その結論の質を測る尺度も存在しない．さらに，手動で行うことは時間と労力を消費し，いつも可能であるとは限らない [9]．

そこで，本研究では，複雑に連携するメトリクスを，ベイジアンネットワーク，クラスタリングという，2つのメトリクス分析技術を用いてモデル作成し，データログを入力として，人間にとって直感的に理解しやすい確率として出力するシステムを開発した．また，本システムを用いて得られた確率が，正しく障害を検知できているか，実験によって確認した．

その結果，検知回数に対しては，検知成功回数の割合は半分以下であったが，障害が起こる回数に対しては半分以上であり，この結果はWebシステムの障害検知としては有効であるということを示した．

以降，2節では，本研究の背景と目的について説明する．また，その目的を達成するために，2つのメトリクス解析技術について説明し，それらを組み合わせて障害を検知する方法を考える．3節では，2つのメトリクス解析技術を組み合わせた障害検知を実際に行うために構築した障害検知システムについて説明する．4節では，構築した障害検知システムに対して行った実験について説明する．5節では，2つのメトリクス解析技術を組み合わせた障害検知が有効であるかどうかを調査するために行った，評価実験について説明する．6節ではまとめとして，今後の課題について述べる．

2　背景

本研究の背景として，クライアント-サーバシステムについて，その問題点と，解決方法について説明する．

2.1　クライアント-サーバシステム

一般に，クライアント-サーバシステムとは，クライアント部とサーバ部の 2 つの部分で処理を分担するネットワークシステムのことを指す．クライアント部では，ユーザからの要求をサーバに渡し，サーバではその要求に従って処理を行う．その結果をクライアント部に返すことによって，ユーザに提示する，という処理の流れになる．図 1 にクライアント-サーバシステムの概略図を示す．

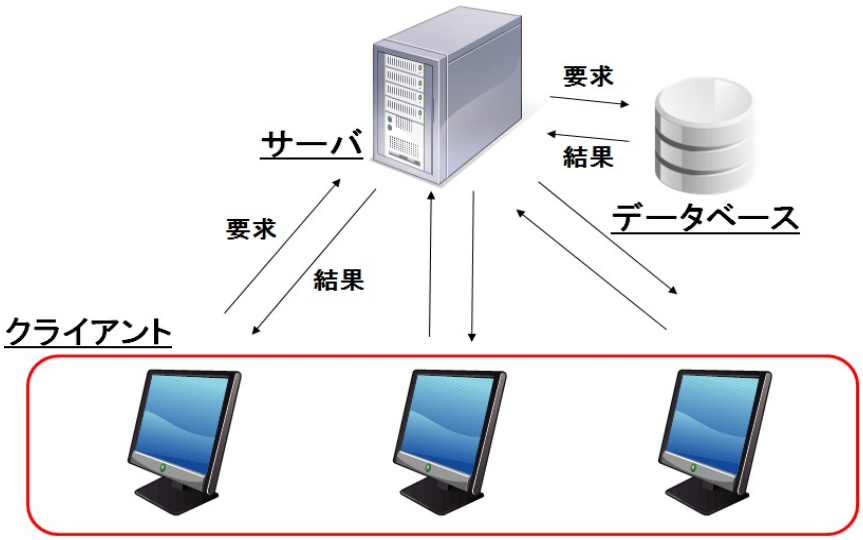

図 1: クライアント-サーバシステムの概略図

クライアント-サーバシステムのメリットとして，サーバが管理するプリンタ，ハードディスク等のハードウェア資源や，データ，ファイル，プログラムなどのソフトウェア資源を，各クライアントが共有利用することができる点が挙げられる．また，サーバアプリケーションの種類により，Web サーバ，メールサーバ、データベースサーバなどとして機能し，多様なサービスを提供することができる．さらに，上記を含め，ユーザの情報等を統一的に管理することができる．

一方で，サーバは資源やサービス，管理情報を一挙に担っているので，サーバに障害が起

こったときの被害は，P2P 等の通信方式と比べて甚大になる． 以上のような理由から，このような Web システムの長期安定稼働を目的とした障害検知を行う場合，サーバの障害に対して検知を行うことが非常に重要である．

サーバの障害検知の方法としては，ハードウェアを計測対象とした障害検知，ソフトウェアを計測対象とした障害検知の 2 種類が存在する．

ハードウェアを対象とした障害検知

ハードウェアを対象とした障害検知は，主にシステムの電気，熱など，物理的な変化に対して障害を検知し，障害が起きた際には，その系を切り替えることでシステムの稼働を維持する，といった方法を取っている．

ソフトウェアを対象とした障害検知

ソフトウェアを対象とした障害検知は，サーバ管理者が，サーバマシンの CPU やメモリの利用率，ディスク利用量，ネットワークトラフィックなど，個々の計測メトリクスの推移を一定の時間間隔で計測し，それをもとに，システムに今後不具合が発生するかどうかを判断する方法が一般的である．

本研究では，ソフトウェアを対象とした障害検知に着目する．社会基盤としての Web システムは，クライアント数やその規模に伴う大量のメトリクスを抱えており，また，それらは複雑に相関している．管理者がそれらのメトリクスを調査し，相関関係を見出し，障害を検知するということは困難であり，その判断は人間の勘や経験のみに依存する．

そこで本研究では，複雑に連携するメトリクス群を，解析技術を利用して処理し，障害検知を自動化することを考える．複雑に連携するメトリクス群を処理する解析技術として，以下のような解析技術が存在する．

2.2 ベイジアンネットワーク

ベイジアンネットワーク (Bayesian network)[] とは，不確実性を含む事象の予測や，観測結果の原因探索手法として利用することのできる確率モデルである．複数の確率変数の間の定性的な依存関係をグラフ構造によって表し，個々の変数の間の定量的な関係を条件付確率で表すことができる．これを用いた確率計算によって，不確実性を含む事象の予測やシステムの制御，障害診断等の知的情報処理に利用することができる．

例えば，データベースサーバに関するベイジアンネットワークは図 2 のようになる．データベースの読み書きやスワップなどの動作が Web サーバの CPU やメモリの利用率に影響し，それらが応答時間の遅延の原因となっている．

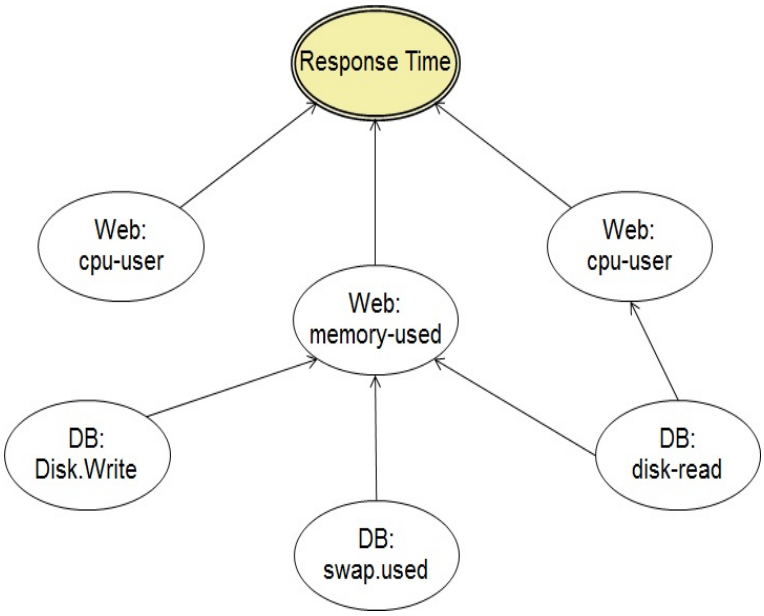

図 2: ベイジアンネットワークのグラフ構造図

2.3 クラスタリング

クラスタリング (clustering)[11] とは，分類対象の集合を，内的結合と外的分離が達成されるような部分集合に分割することであり，基本的なデータ解析手法としてデータマイニングの分野で利用されている．本研究では，データ解析におけるクラスタリング手法として，K 平均法について説明する．

K 平均法 [12] とは，まず，n 種類の取得データを 1 点として，n 次元空間上にプロットする，それらを K 個のクラスタに分類したモデルを生成し，入力データとの距離を計算するという手法である．

例えば，図 3 では，CPU 利用率，メモリ利用量，ネットワーク送受信量の 3 種類のメトリクスに対して，正常時の点をクラスタとしてモデル化し，入力データとの距離計算を行っている．

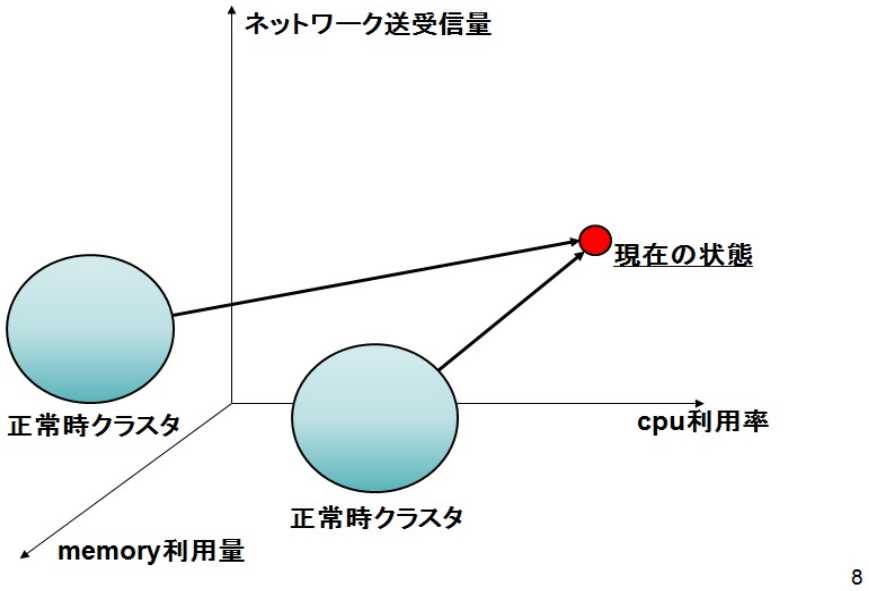

図 3: クラスタリング図

2.4 解析技術の比較

Web システムの障害を検知する，という観点から，2 つの解析技術を，利点と欠点を挙げて比較する．

ベイジアンネットワーク

ベイジアンネットワークの利点としては，メトリクスの相関関係をモデル化できる [10] ことが挙げられる．この利点は，複雑性の高いメトリクスを処理するという，本研究に目的に合致している．また，人間にとって直感的に理解しやすい確率という形で出力が行われるため，検知の際の閾値を設定しやすいという良さもある．

一方で，ベイジアンネットワークの欠点としては，学習情報が少ない場合には障害検知が困難であるという点が挙げられる．相関関係を基に確率を算出するため，過去のデータに障害が起こった事例がない場合には検知することが不可能である．

クラスタリング

クラスタリングの利点としては，入力メトリクスとクラスタ間の距離計算という単純な処理であるため，学習情報が少量であっても検知が可能であることが挙げられる．この利点によって，ベイジアンネットワークが検知できない場合にも，クラスタリングによる検知を行

うことができる．

一方で，クラスタリングの欠点としては，学習情報の選定が困難であるという欠点がある．クラスタリングは，正常時のメトリクスをモデル化して，異常時との差を距離として算出するため，異常時のデータを持つメトリクスを学習した場合，算出距離が短くなり，検知が不可能になる．したがって，モデル化の際，異常時のデータの除去が必要になる．

そこで，本研究では，2 つの解析技術を組み合わせて，障害検知を自動化する方法を考える．そのために，以下の手順で実験，評価を行う．

1. Web システムを実装する．
2. 実装した Web システムに負荷を注入し，そのメトリクスを収集する．
3. メトリクスを基にベイジアンネットワーク，クラスタリングそれぞれのモデルを生成する．
4. 生成したモデルを利用して，実際に障害検知ができるか実験，評価を行う．

3　障害検知システム

本研究では，クライアント-サーバシステムという，一般的な Web システムを対象として，ベイジアンネットワーク，クラスタリングという，2 つのメトリクス解析技術を利用した障害検知システムが有効であるか

3.1　Web システム構成

本研究で実装する Web システムの環境について説明する．実装環境を図 4 に示す．

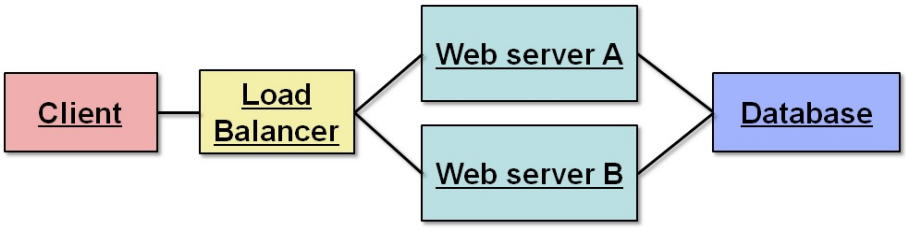

図 4: 実装する Web システムの構成

実装環境は，クライアント，ロードバランサ，Web サーバ，データベースの 4 つのコンポーネントで構成する．Web サーバは，一般的な Web システムには複数存在することを鑑み 2 台，その他のコンポーネントは，それぞれ 1 台の仮想計算機に対して実装する．以下に各システムの役割について説明する．

クライアント

役割

Web サーバにリクエストを送信し，その結果を受信する．

利用アプリケーション

クライアント用のアプリケーションとして，Apache JMeter を用いる．

Apache JMeter[3] は，クライアント-サーバシステムのパフォーマンス測定および負荷テストを行う Java アプリケーションである．Web ページを指定して，アクセスクライアント数，間隔，ループ回数等を設定することによって，クライアントからサーバへのアクセスを見せかけることができる．

ロードバランサ

役割

クライアントから送られたリクエストを Web サーバへと中継する．複数の Web サーバの状態をモニタリングして，リクエストによる負荷を分散させることができる．

利用ツール

ロードバランサの実装ツールとして，Apache mod_proxy，balancer を用いる．

Apache mod_proxy，balancer[1] は，Apache が提供する負荷分散ツールであり，クライアントからのアクセスを複数のサーバに分散することができる．本研究では，2 台の Web サーバに対して，均等に負荷分散を行っている．

Web サーバ

役割

ロードバランサから受け取ったクライアントからの要求を，データベースを用いて処理し，その結果を返す．

利用アプリケーション

サーブレットコンテナとして Apache Tomcat，サーバアプリケーションとして JPetStore を利用する．

Apache Tomcat[2] とは，Java Servlet や JavaServer Pages を実行するためのサーブレットコンテナである．Apache Tomcat を利用することによって，サーバ上で HTML などの Web ページを動的に生成する Java サーブレットを動作させることができる．

また，JPetStore とは，Java サーブレットを利用してサーバ上で動作するアプリケーションである．データベースの情報を基に架空のペットストアを Web ページとして表示することができる．

データベース

役割

サーバから受け取った要求を基にデータの検索，抽出を行い，その結果をサーバへと返す．

利用データベースシステム

データベースシステムとして，MySQL を利用する．

MySQL[5] とは，Oracle が開発するリレーショナルデータベースを管理，運用するためのアプリケーションである．本研究では JPetStore のデータをデータベースとして登録し，Web サーバからアクセスすることによって，データベースサーバを実装する．

各アプリケーション，ツールをシステムに反映し，図 5 に示す．

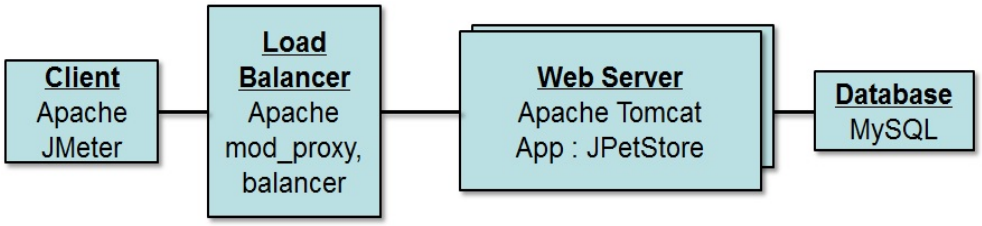

図 5: Web システム

3.2 メトリクス収集

メトリクス収集ツールとして，collectd を利用する．

collectd

collectd[6] は，Web サーバの CPU 利用率，メモリ利用量等の各要素を一定の時間間隔でメトリクスとして収集するデーモンプログラムである．このプログラムを用いて，Web サーバへの負荷に関するデータを収集する．

3.3 負荷

Web システムに対して負荷をかけるツールとして，stress を利用する．

stress

stress[7] は，システム用の作業負荷ジェネレータである．指定した量の負荷を CPU，I/O，RAM 及び HDD に一定時間強制的に発生させることができる．

3.4 解析ツール

植田のツールは，ベイジアンネットワークとクラスタリングを利用した，障害検知のための解析ツールである．ある区間のメトリクス群を入力として，ベイジアンネットワーク，ク

ラスタリングのモデルを作成し，入力データの確率，距離計算を行うことができる．

4　実験

本研究で行う実験について述べる．

4.1　実行手順

まず，本研究の実験プロセスについて説明する．実験プロセスを図 6，図 7 に示す．

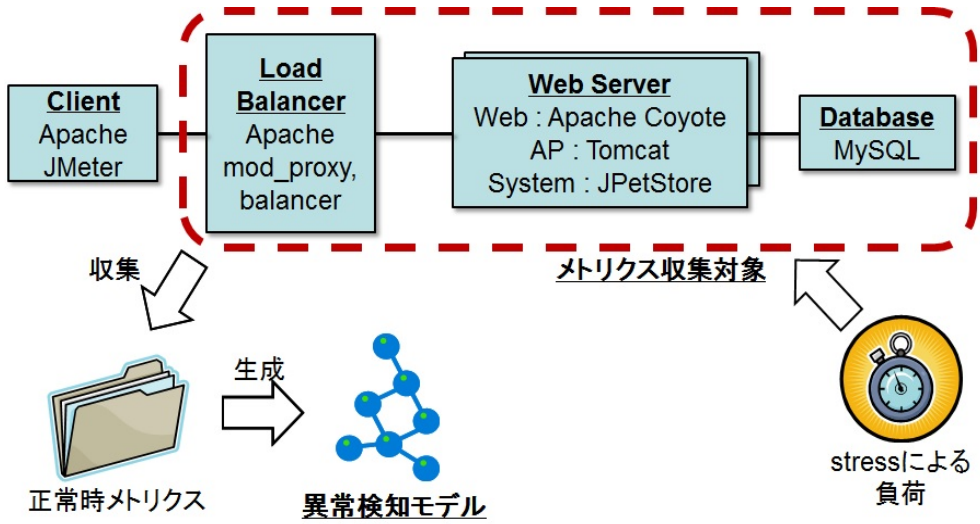

図 6: プロセス 1

まず，クライアントから Web サーバへのアクセスを継続した状態で，Web サーバに一定時間負荷を加える．次に，Web システムの構成要素であるロードバランサ，Web サーバ，データベースを収集対象として，メトリクス収集を行う．さらに，そのメトリクス群を入力として，ベイジアンネットワーク，クラスタリングそれぞれの異常検知モデルを生成する．

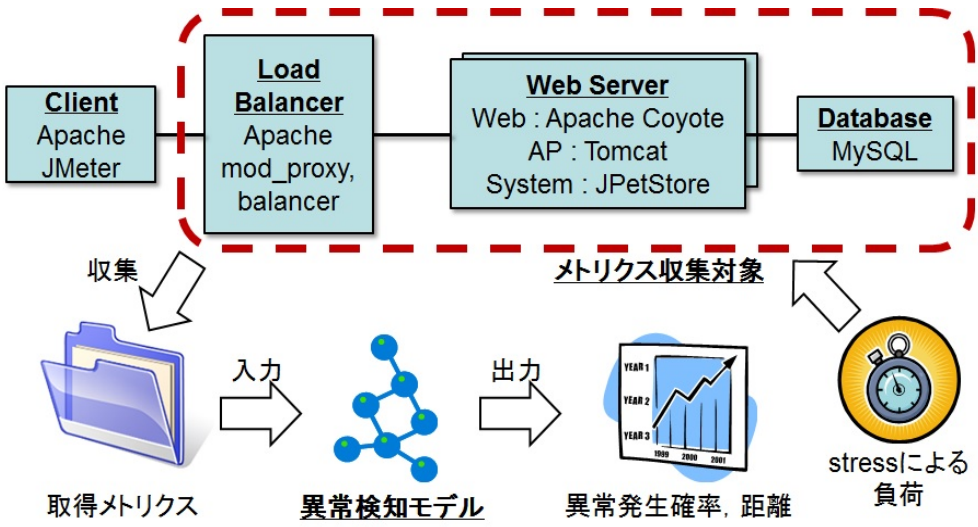

図 7: プロセス 2

異常検知モデル生成後，再び Web サーバに負荷をかけて，メトリクス収集を行う．その新たなメトリクス群を入力とし，生成したモデルを利用して，異常発生確率，距離の算出を行う．

4.2 計測メトリクス

異常検知モデル生成のために利用するメトリクス群を表 1 に示す．

メトリクス名	意味	単位
CPU-user	CPU 利用率	%
Memory-used	メモリ利用量	byte
Disk-ops	DiskI/O オペレーション数	ops/sec
Network	ネットワーク送受信量	byte/sec

表 1: 計測メトリクス

4.3 仕様

本研究で利用する各仮想計算機について，その仕様を表 2 に示す．

	CPU	メモリ	ストレージ	アプリケーション	OS
クライアント	2core	1GB	30GB	Apache JMeter 2.10	CentOS Ver6.4 x64
ロードバランサ	1core	1GB	30GB	Apache mod_proxy, balancer	
Web サーバ A	1core	1GB	30GB	Apache Tomcat 6.0.24-57	
Web サーバ B	1core	1GB	30GB	iBatis JPetStore 4.0.5	
データベース	1core	4GB	110GB	MySQL 5.1.69-1	

表 2: 各仮想計算機の仕様

4.4 メトリクス取得手順

Web システムのメトリクス取得方法について説明する．メトリクス収集は 15 分単位で行い，Web サーバ A，B それぞれに時間をずらして負荷をかける．時間単位のプロセスを表 3 に示す．

時間	要件
0：00〜3：00	負荷注入なし
3：00〜6：00	Web サーバ A に負荷注入
6：00〜9：00	Web サーバ A，B 両方に負荷注入
9：00〜12：00	Web サーバ B に負荷注入
12：00〜15：00	負荷注入なし

表 3: 実験プロセス

さらに，表 4 の区間に沿ってメトリクス収集し，それを入力として異常検知モデル作成を行う．そのモデルを利用して，メトリクス群をを入力とした確率，距離計算を行う．

区間数	メトリクス取得時間
1	0：00〜3：00
2	0：00〜6：00
3	0：00〜9：00
4	0：00〜12：00
5	0：00〜15：00

表 4: モデル生成の区間

4.5　結果

各メトリクスの時間変化の 1 例をグラフとして示す.

取得メトリクスの変化グラフ

Web サーバ A のメトリクスグラフを図 8，図 9，図 10，図 11 に，Web サーバ B のメトリクスグラフを図 12，図 13，図 14，図 15 に示す.

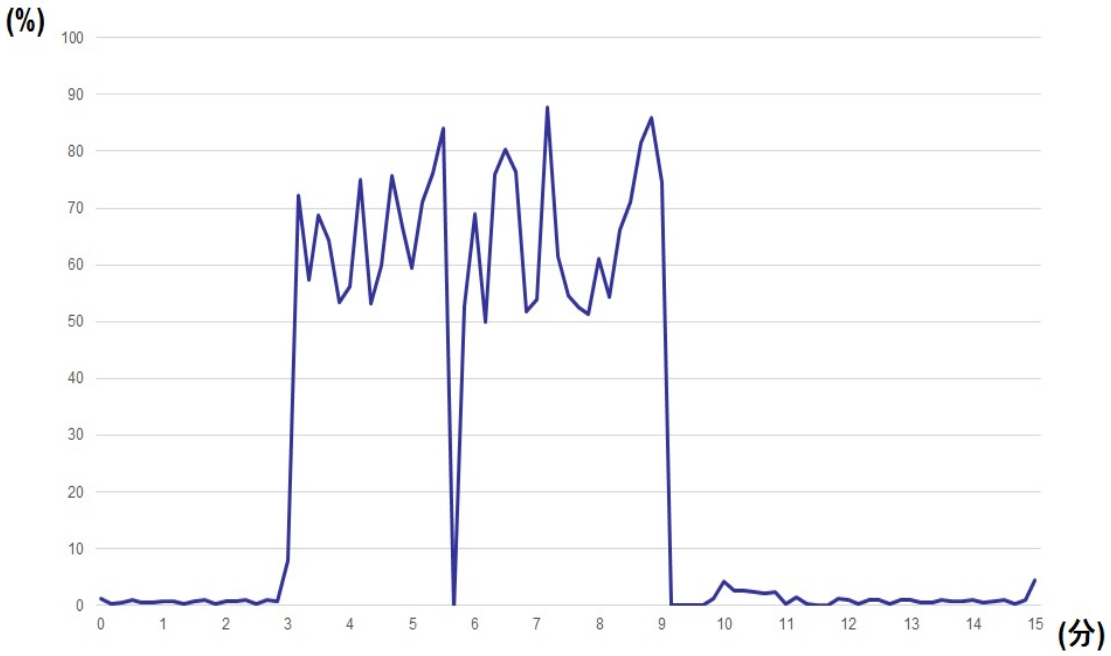

図 8: Web サーバ A CPU 利用率

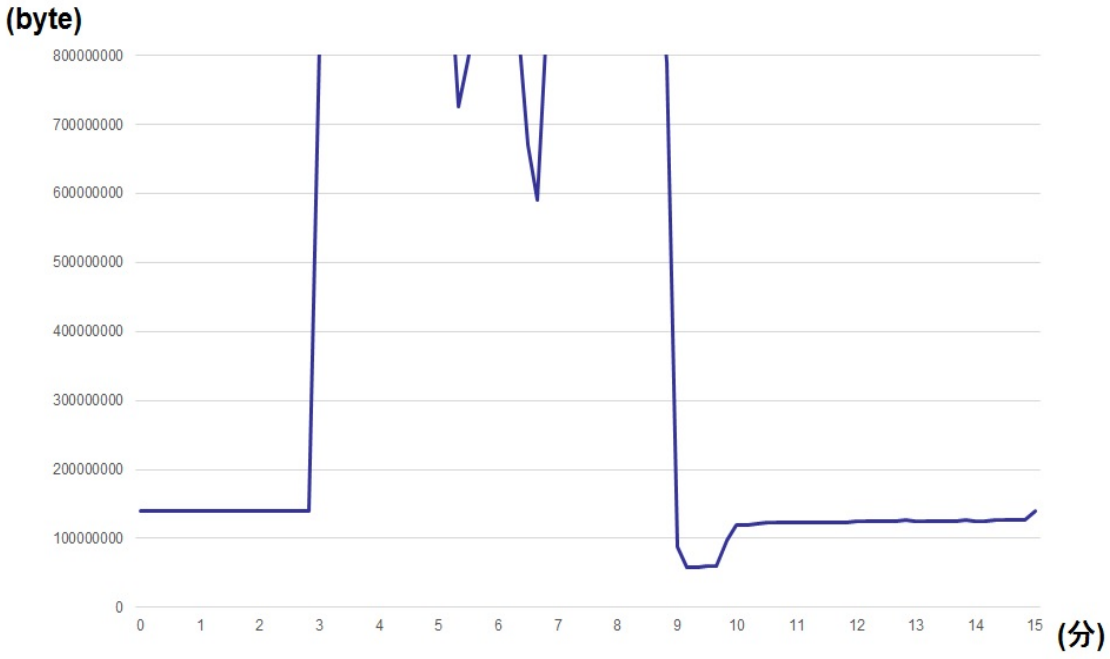

図 9: Web サーバ A メモリ利用量

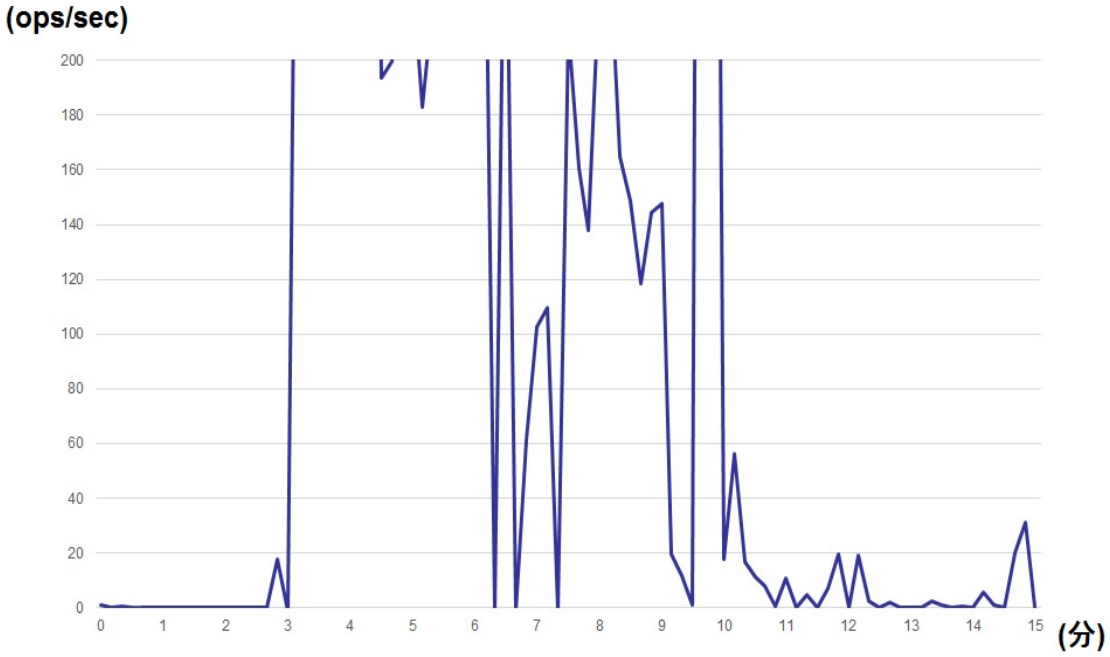

図 10: Web サーバ A ディスクオペレーション数

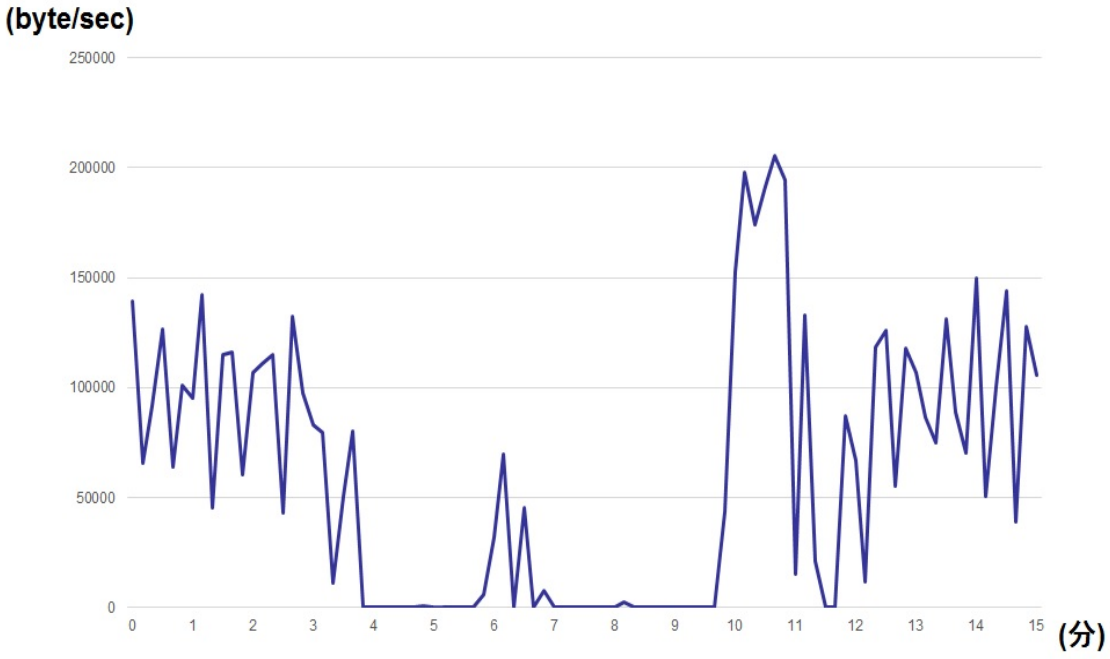

図 11:
Web サーバ A ネットワーク送受信量

CPU 利用率，メモリ利用量，ディスクオペレーション数については，負荷注入を行った 3 分から 9 分の区間で顕著な上昇が見られる．ネットワーク送受信量については，負荷注入を行った区間での低下が見られるが，その理由として，Web サーバ A に負荷が注入されたため，クライアントのリクエスト処理が滞ったためであると考えられる．

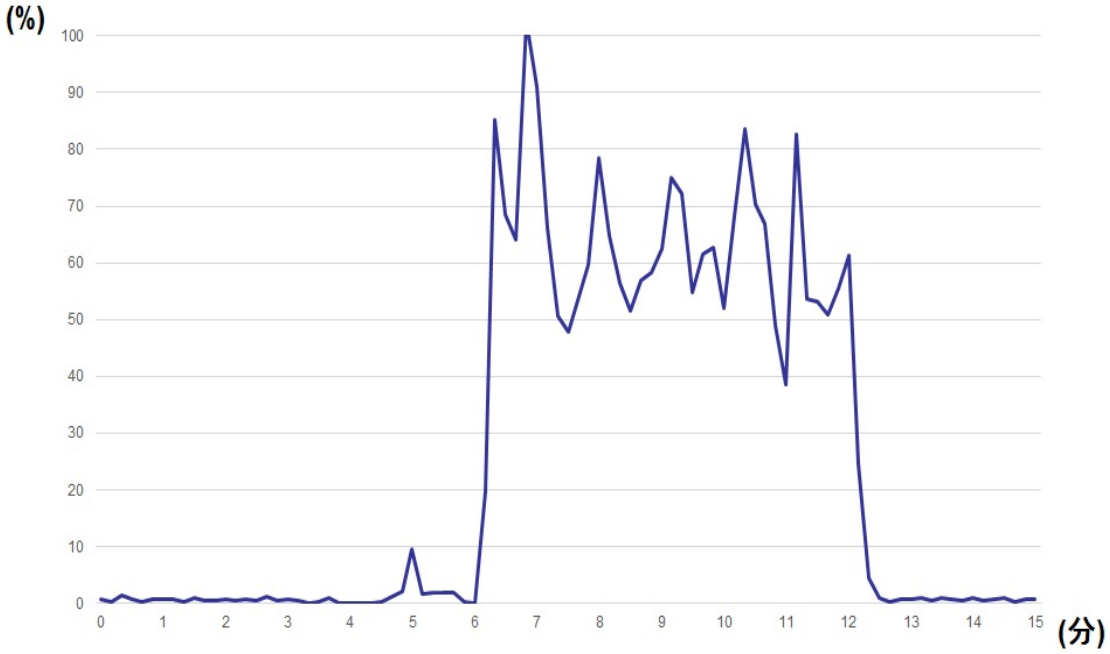

図 12: Web サーバ B CPU 利用率

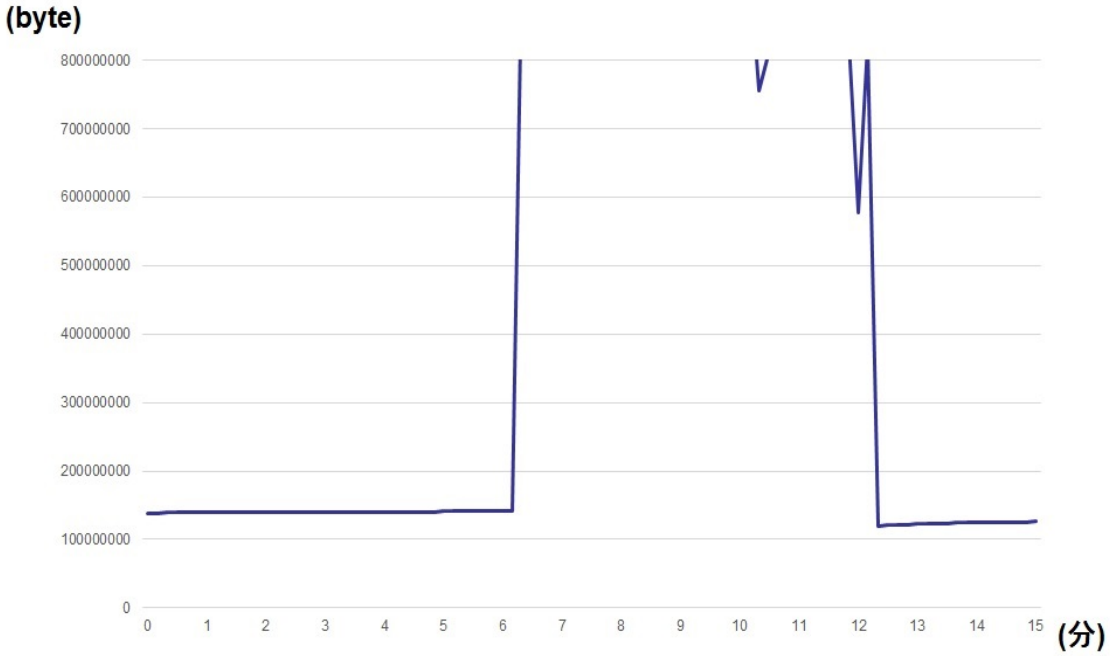

図 13: Web サーバ B メモリ利用量

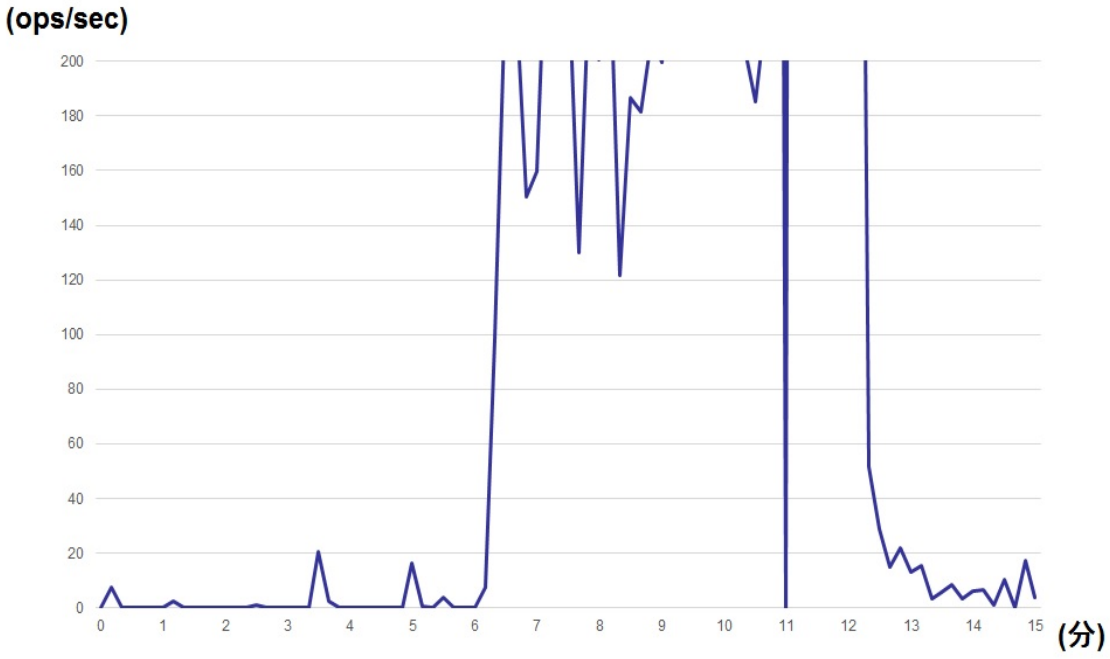

図 14: Web サーバ B ディスクオペレーション数

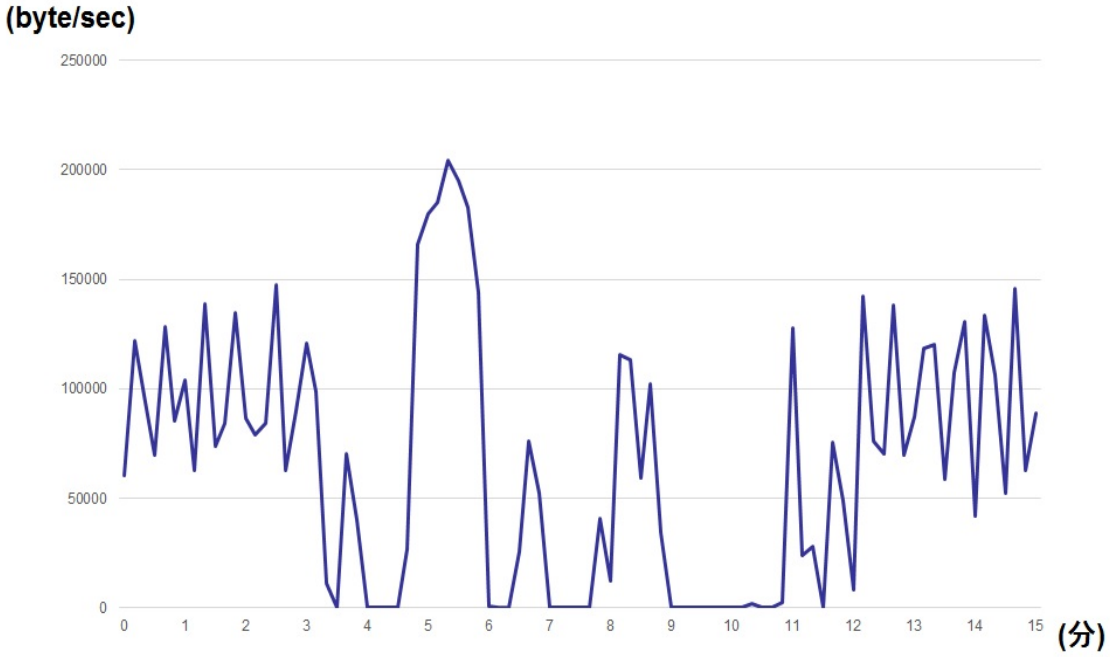

図 15: Web サーバ B ネットワーク送受信量

Web サーバ B についても，Web サーバ A のメトリクスと同様に，CPU 利用率，メモリ利用量，ディスクオペレーション数については負荷注入を行った 6 分から 12 分の区間で顕著な上昇が見られるが，ネットワーク送受信量については，負荷注入を行った区間で低下している．

ネットワーク送受信量についての結果を詳しく見るために，ロードバランサの CPU 利用率，リクエスト数，応答時間についても収集を行った．その時間変化グラフを図 16，図 17，図 18 に示す．

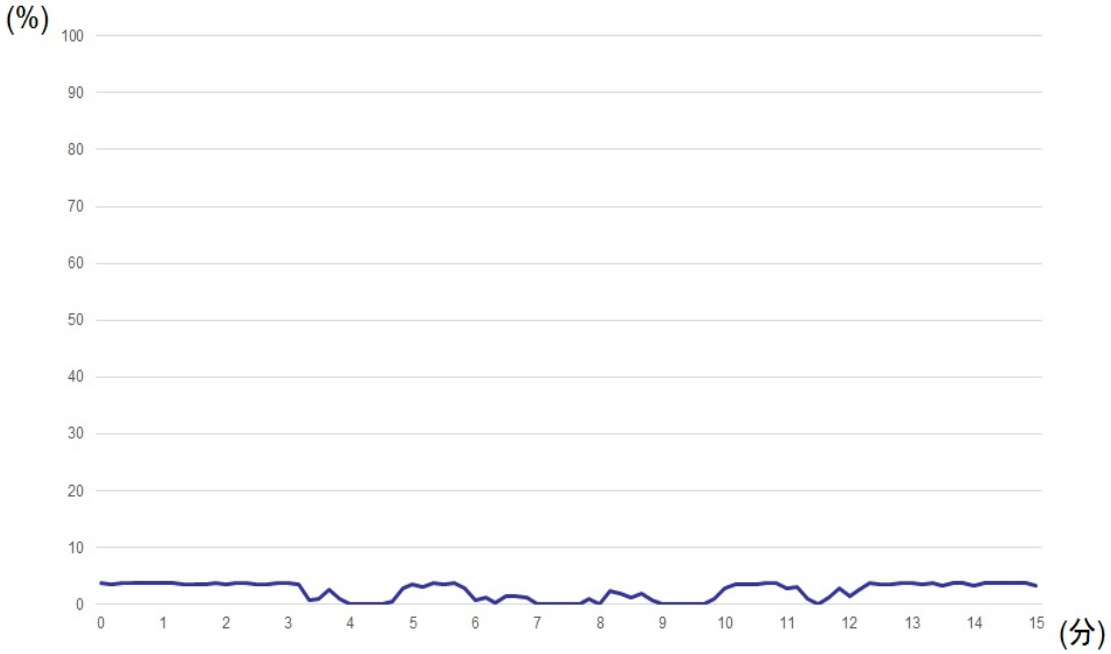

図 16: ロードバランサ CPU 利用率

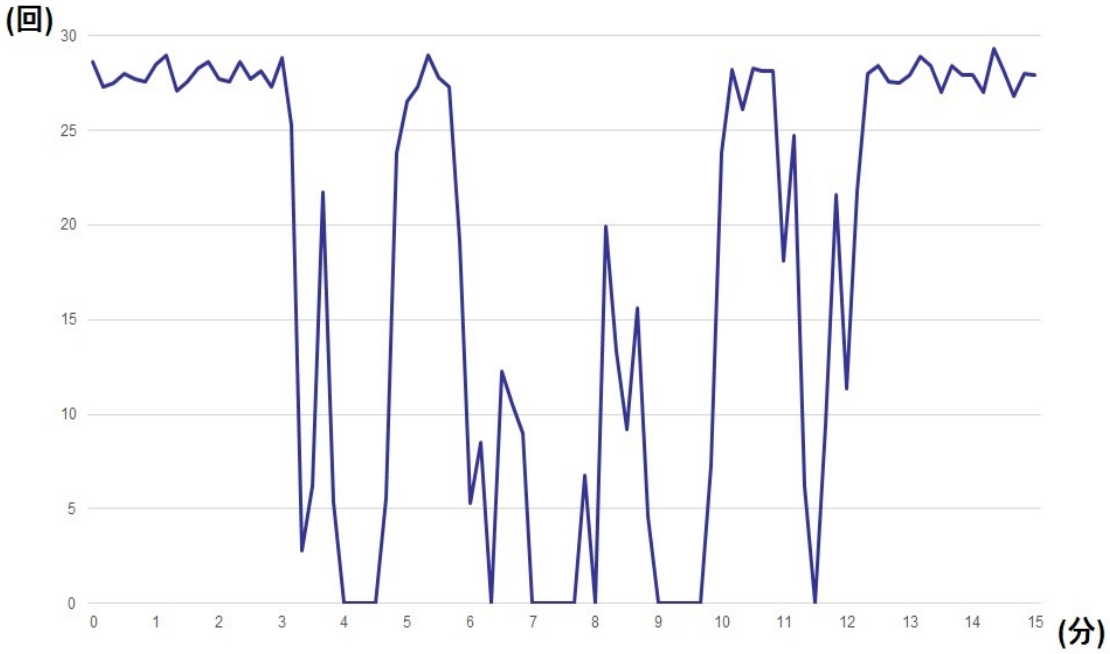

図 17: ロードバランサ リクエスト数

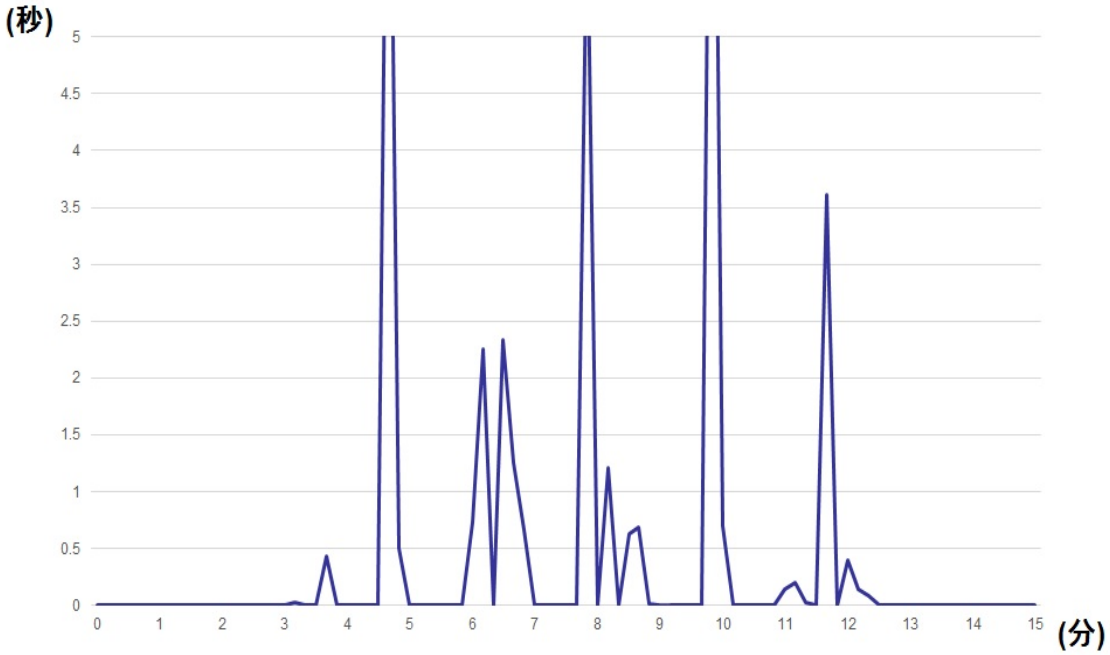

図 18: ロードバランサ 応答時間

これらを見ると，CPU 利用率は 10 %以下を一定に保っているが，リクエスト数は，Web サーバ A，B に負荷をかけている 3 分から 12 分にかけて，大きな低下が見られる．さらに，応答時間については，同時間において，極端な上昇が見られる．これらの結果からも，Web サーバ A，B への負荷注入によって，Web サーバのリクエスト処理が滞っていることがわかる．

異常検知モデルの出力結果

メトリクス群を基に，それぞれの区間に対して検知モデルの生成を行った．モデル生成は 30 のメトリクス群に対して行い，生成された 30 のモデルのうち，応答時間に対して，最も相関がとれているものを 1 つ選び，異常検知モデルとした．

そして再度 Web システムに負荷をかけ，メトリクスの収集を行い，そのメトリクスを異常検知モデルに入力して，確率，距離を算出した．ロードバランサの応答時間に対する変化をグラフに表した．

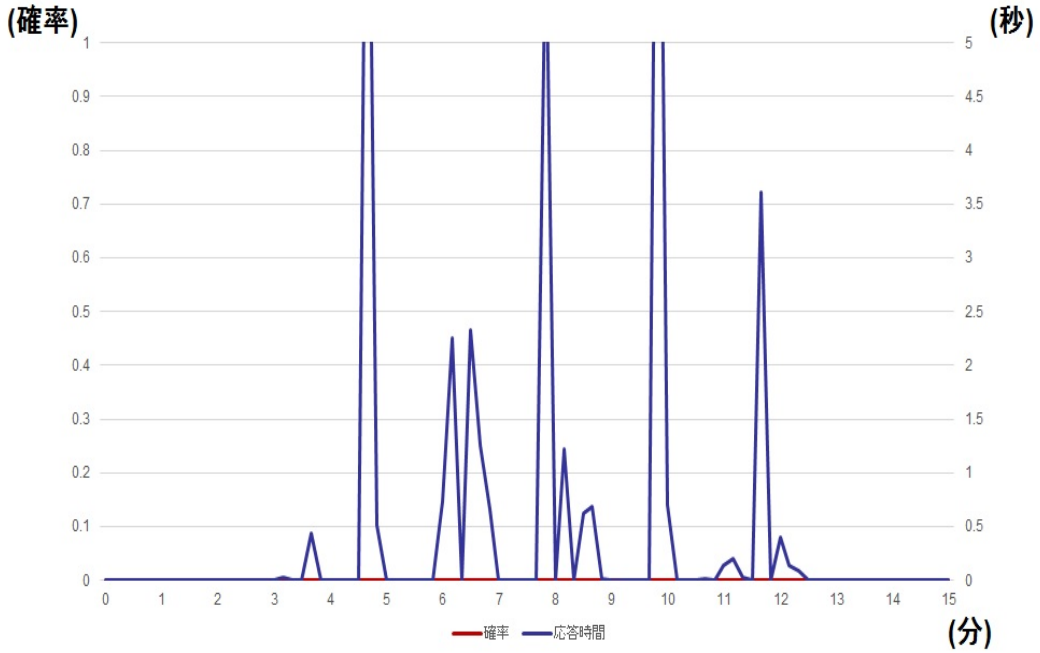

図 19: 区間 1 における確率検知グラフ

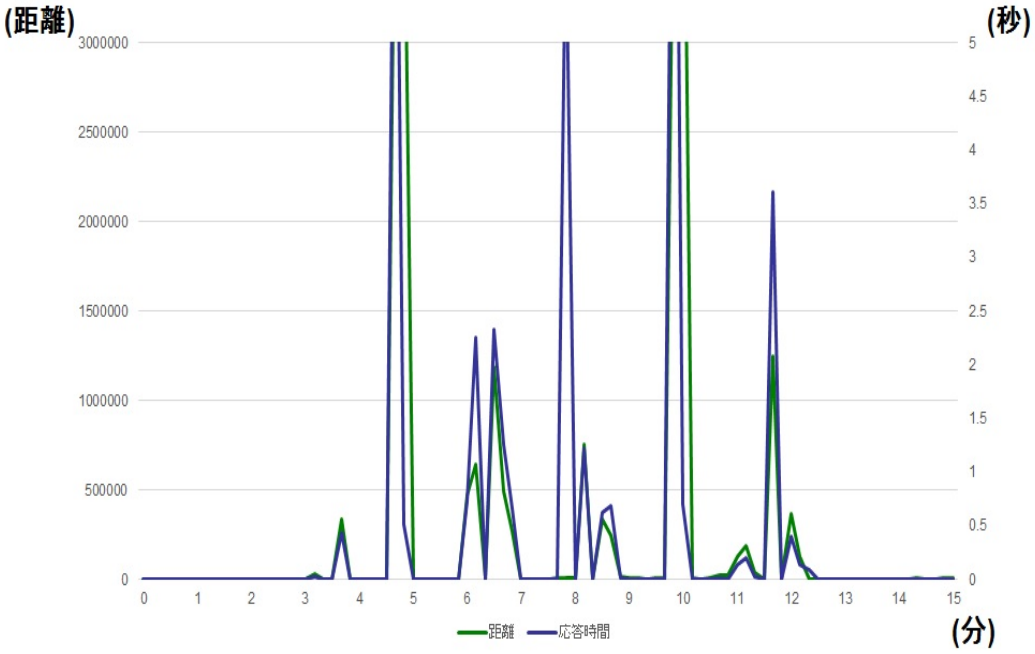

図 20: 区間 1 における距離グラフ

区間 1 は負荷が加えられていない，正常の状態が継続するため，ベイジアンネットワークは負荷の学習をしておらず，適切な検知ができていない．一方で，クラスタリングは正常時をモデルとして距離計算できるため，応答時間が上昇する時刻に対応して距離を大きくしている．

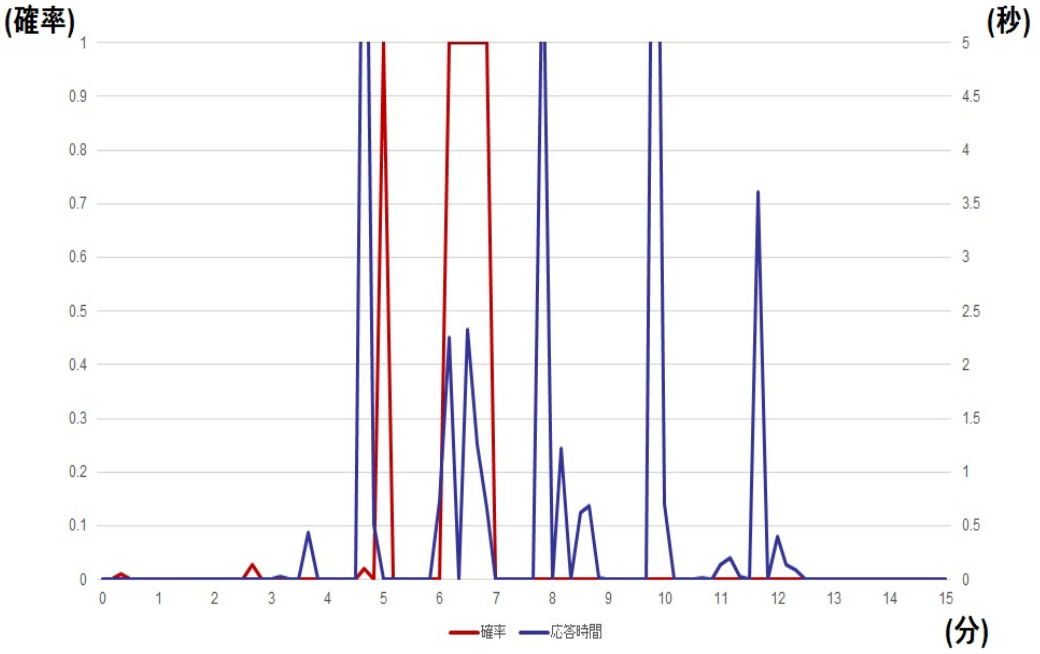

図 21: 区間 2 における確率検知グラフ

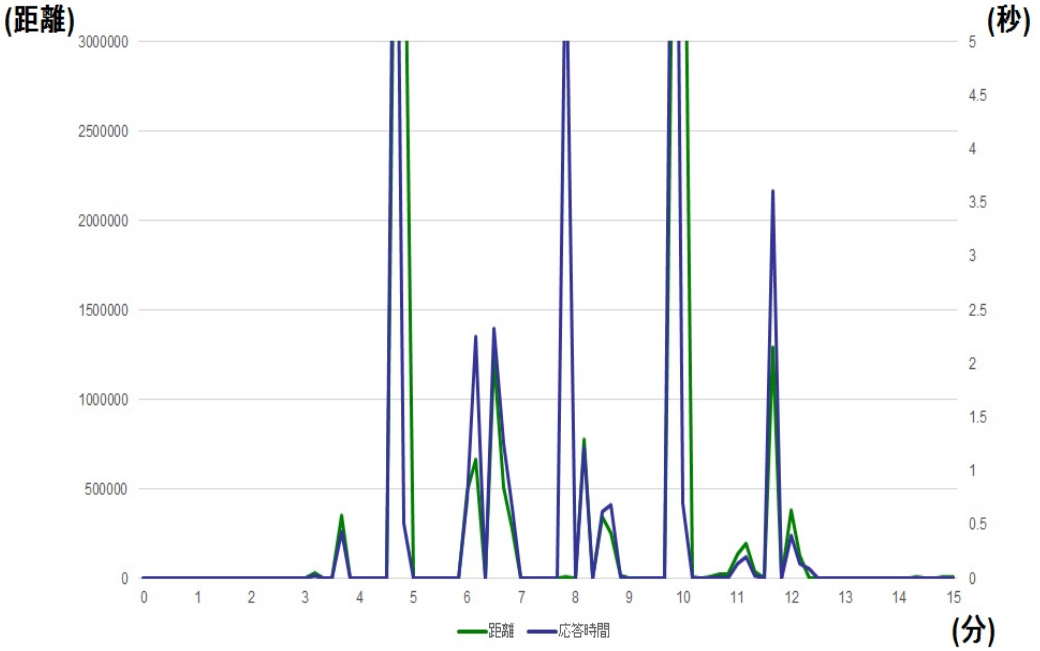

図 22: 区間 2 における距離グラフ

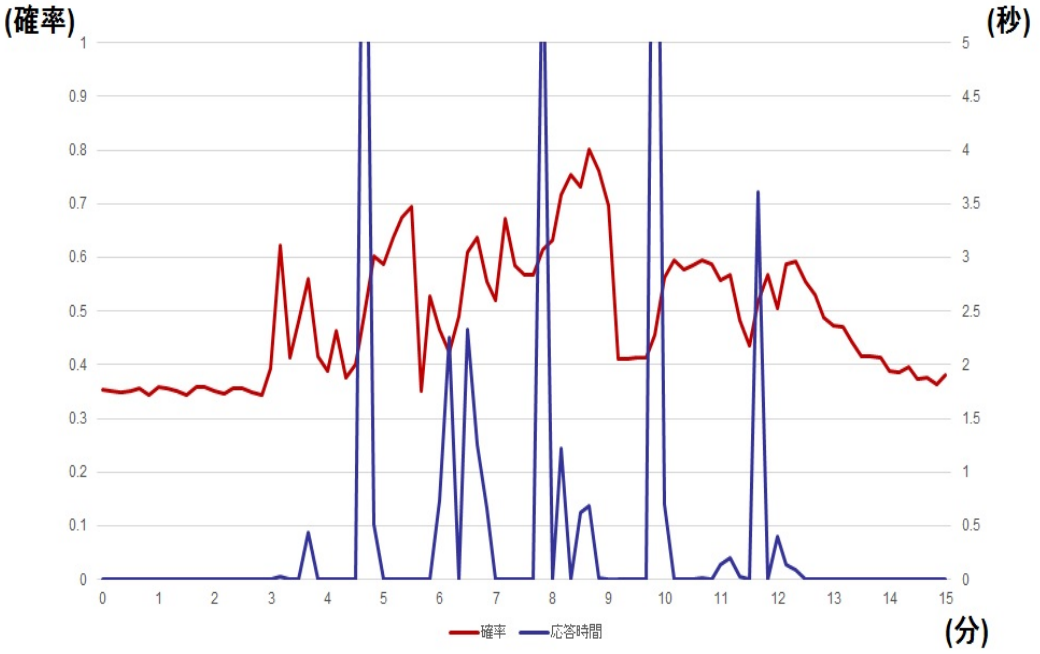

図 23: 区間 3 における確率検知グラフ

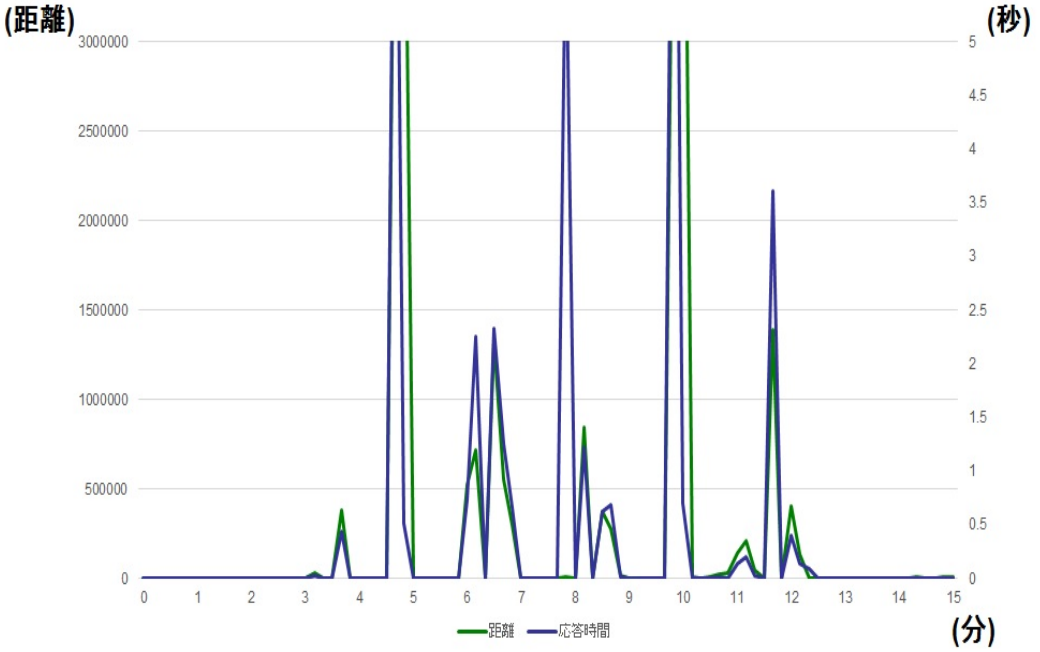

図 24: 区間 3 における距離グラフ

区間 2，3 では，ベイジアンネットワークモデルが負荷の学習をしたことによって，徐々に応答時間に対応したグラフを示すようになっている．クラスタリングは正常時，異常時の混在したメトリクスを利用してモデル生成を行っているが，障害を正確に検知している．この理由としては，応答時間の上昇が非常に大きいためであると考えられる．

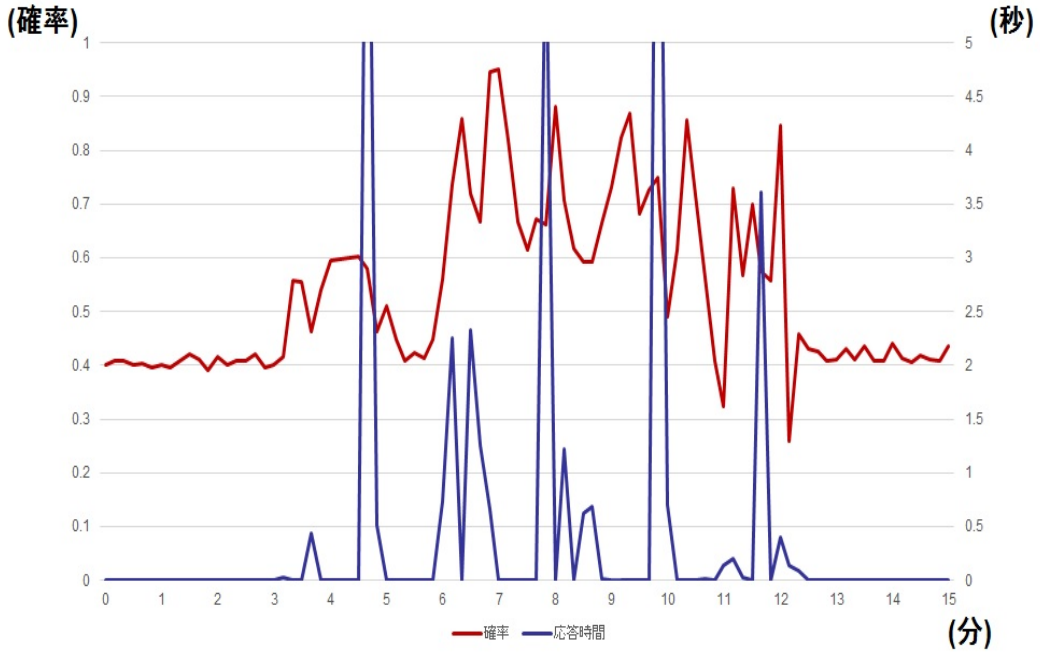

図 25: 区間 4 における確率検知グラフ

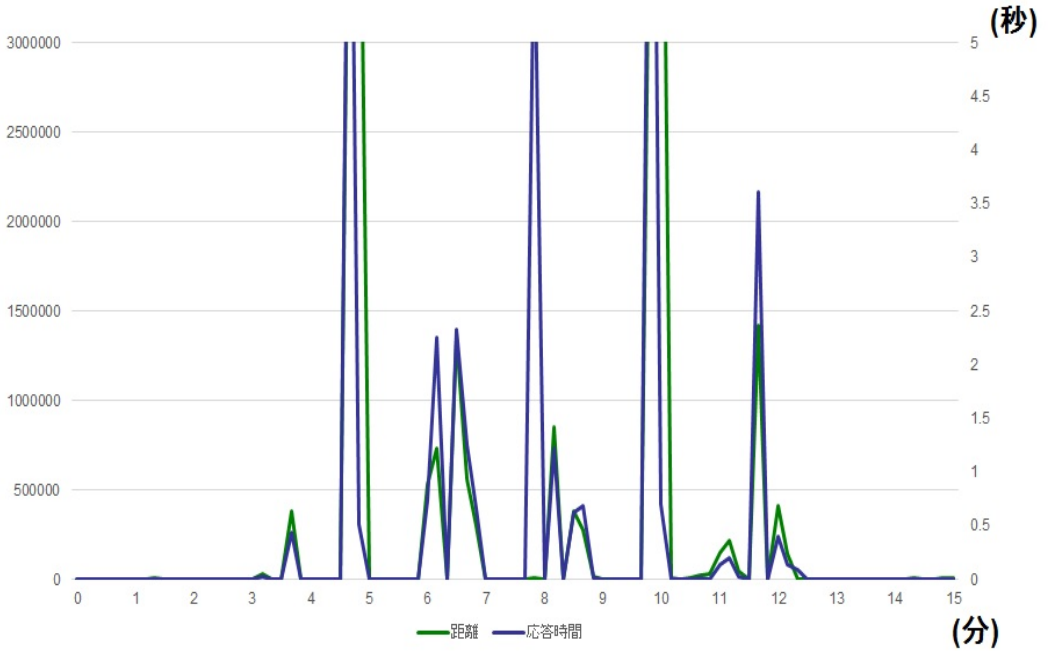

図 26: 区間 4 における距離グラフ

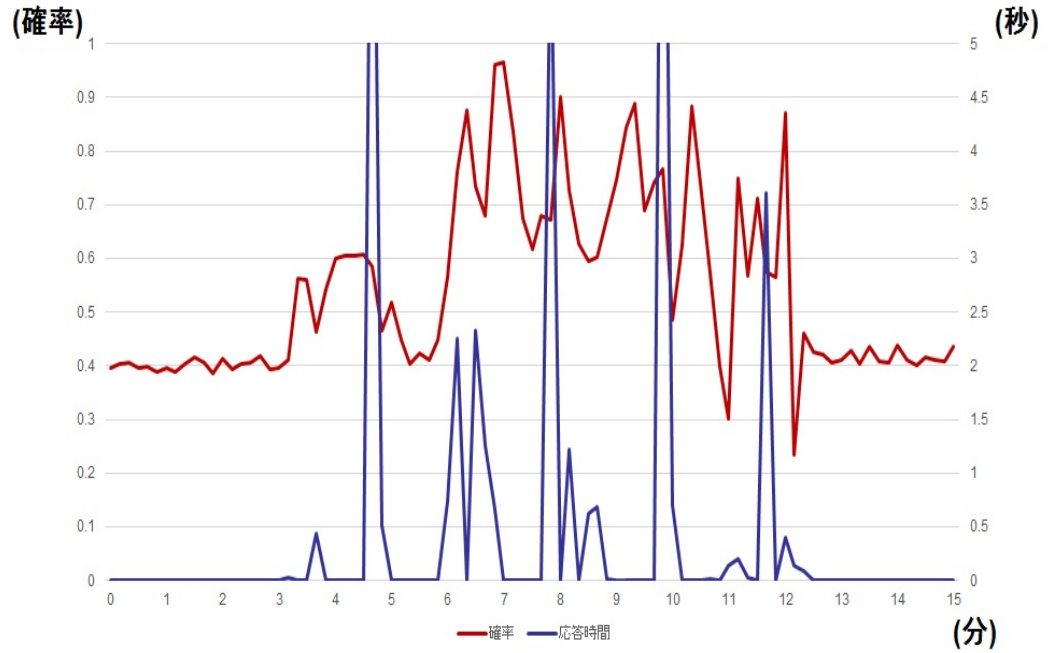

図 27: 区間 5 における確率検知グラフ

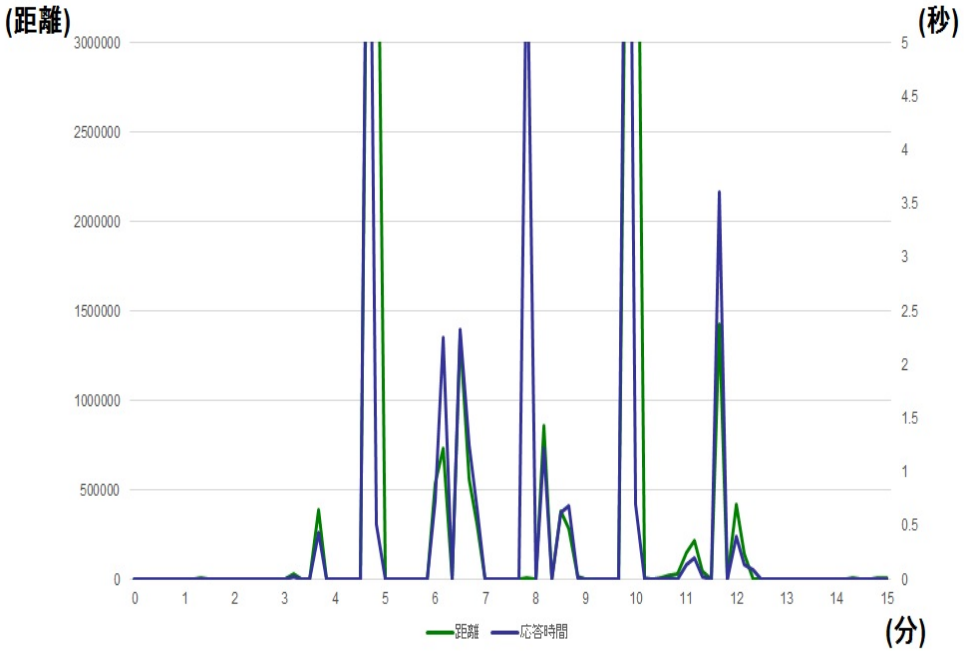

図 28: 区間 5 における距離グラフ

区間 4，5 では，学習情報を増やしたベイジアンネットワークモデルが，応答時間の上昇を事前に検知し，それを確率として反映している．クラスタリングは区間 2，3 の結果と同様に，正常時，異常時の混在したメトリクスを利用してモデル生成を行っているが，応答時間に対応した結果を示している．

総じて，ベイジアンネットワークは，少量のデータを基にモデルを生成する場合，メトリクスの相関関係に関する情報が少ないため，その検知精度は低いが，様々な状況のメトリクスを取得，学習することによって，その検知精度を上げることができる．

一方で，クラスタリングは，正常時のデータを基にモデルを生成する場合，障害検知に利用できる．また，異常時のデータが混在している場合にも，障害を検知できることがある．

5　評価

実験時に生成した異常検知モデルが，実際に障害検知をできることを，評価実験を行うことによって示す．

5.1　障害の定義

評価実験における，Web システムの障害として，ロードバランサの応答時間が 3 秒を超えた時，障害が起こった，と定義する．

5.2　障害検知の定義

次に，本研究における障害検知を定義する．以下の 2 つの条件を同時に満たす場合に，検知を行うものとする．

1. ベイジアンネットワークモデルで算出した確率の値が 0.6 を超える，
2. クラスタリングモデルで算出した距離の値が 100000 以上である．

5.3　評価項目

この定義のもと，検知開始から 1 分以内に障害が発生すれば検知成功，しなければ検知失敗とする．さらに，検知成功時，障害が起こった時刻と検知開始時刻との差を対処時間とし，障害を検知してから，それに対処するためにかけられる時間を調べる．

5.3.1　評価結果

これらの評価条件で，異常検知モデル生成に利用したメトリクスとは別に，15 のメトリクス群を収集し，それらに対して評価実験を行った．その結果を表 5 に示す．

平均障害回数	平均検知回数	平均検知成功回数	平均検知失敗回数	平均対処時間
3.47 回	4.40 回	1.87 回	2.53 回	20.6 秒

表 5: 評価結果

また，この結果に対する適合率，再現率を表 6 に示す．

適合率	再現率
42.5 %	53.9 %

表 6: 適合率，再現率

5.4 考察

表 5 を見ると，平均検知回数に対して，平均検知成功回数の割合が 50 %以下と低いが，その原因としては，障害が連続して起こり，確率が高い値を出し続けること，検知しても平均応答時間が閾値に到達しないことが主に挙げられた．

また，表 6 を見ると，再現率が適合率を上回っている．本研究は，社会基盤である Web システムに対して求められる，長期安定稼働を動機としている．そのため，検知漏れを許すが障害を確実に検知する (適合率が高い) よりも，起こる障害のうち，できるだけ多くのものを検知する (再現率が高い) ほうが，障害検知として優れていると考える．評価結果から，再現率は適合率を上回っており，この結果は，本研究における障害検知がその目的に合っていることを示している．

6　まとめと今後の課題

本研究では，Web システムに対する障害検知の自動化を目的として，ベイジアンネットワークとクラスタリングという，2 つのメトリクス解析技術を組み合わせて利用する方法を検討した．2 つの解析技術のモデルを用いた評価実験では，定義した障害に対し，目的に則した障害検知ができることを示し，この手法が有効であることを示した．

今後の課題としては，パラメータを変更して実験し，メトリクスの最適化を行うこと，対処時間を伸ばすことが挙げられる．評価実験の結果，本研究の定義では連続した障害を検知することが難しいことがわかったため，障害と強く関連するメトリクスを選択することによって，検知成功率を向上させることが必要である．

また，対処するためにかけられる時間が 20.6 秒と短く，現状での実用性は低いため，モデル生成のパラメータを変更する，学習データを増やす，学習データの内容について検討するなどして対処時間を伸ばすことが必要である．

最後に，本研究の結果を基に，自作の Web システムでなく，社会的に利用されている Web システムに対して利用可能な障害検知を行うツールを作成し，性能評価を行いたいと考えている．

謝辞

本研究について，常に適切な御指導および御助言を賜りました大阪大学大学院情報科学研究科コンピュータサイエンス専攻井上克郎教授に心より深く感謝いたします.

本研究において，様々な観点から適切な御指導および御助言を賜りました大阪大学大学院情報科学研究科コンピュータサイエンス専攻松下誠准教授に深く感謝いたします.

本研究において，適切な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻石尾隆助教に深く感謝いたします.

本研究の遂行にあたり，熱心かつ丁寧な御指導およびご助言頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻ソフトウェア設計学講座 井垣宏 特任准教授に深く感謝いたします.

本研究において, 様々な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻後藤祥氏に心より感謝いたします.

本研究において，客観的な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻鬼塚勇弥氏に深く感謝いたします.

本研究において，常に丁寧な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻山中裕樹氏に深く感謝いたします.

最後に，その他様々な御指導，御助言等を頂いた大阪大学大学院情報科学研究科コンピュータサイエンス専攻井上研究室の皆様に深く感謝いたします.

参考文献

[1] Apache. http://httpd.apache.org/

[2] ApacheTomcat. http://tomcat.apache.org/

[3] Apache JMeter. https://jmeter.apache.org/

[4] iBatis. https://ibatis.apache.org/

[5] Oracle. http://www.oracle.com/

[6] collectd. http://collectd.org/

[7] stress project page. http://people.seas.harvard.edu/ apw/stress/

[8] A. Fox and D. Patterson. Self-repairing computers. Scientific American, June 2003.

[9] G. A. Alvarez, E. Borowsky, et al. An automated resource provisioning tool for large-scale storage systems. ACM Transactions on Computer Systems (TOCS), 19(4):483.518, November 2001.

[10] Sebastian Thrun, Chair Christos Faloutsos, Andrew W. Moore, Peter Spirtes, Gregory F. Cooper. Learning Bayesian Network Model Structure from Data, May 2003.

[11] Satoshi Iwata, Kenji Kono: Clustering Performance Anomalies Based on Similarity in Processing Time Changes, IPSJ Transactions on Advanced Computing Systems, Vol.5 No.1 1-12, January 2012.

[12] Arthur D, Manthey. B, Roeglin. H. k-means has polynomial smoothed complexity, 2009.

[13] Friedman N, Linial M, Nachman I, Pe'er D. Using Bayesian Networks to Analyze Expression Data. Journal of Computational Biology 7 (3?4): 601-620, 2000.

[14] T. F. Abdelzaher, K. G. Shin, et al. Performance guarantees for Web server end-systems: A controltheoretical approach. IEEE Transactions on Parallel and Distributed Systems, 13(1):80.96, January 2002.

[15] Fayyad, Usama, Piatetsky-Shapiro Gregory, Smyth Padhraic. From Data Mining to Knowledge Discovery in Databases, December 2008.

[16] Liu, Bing. Web Data Mining: Exploring Hyperlinks, Contents and Usage Data, Springer, ISBN 3-540-37881-2, 2007.