修士学位論文

題目

クラスタリングを用いたベイズ学習モデルを動的に更新するソフトウェア障害検知手法

指導教員
井上 克郎 教授

報告者
爲岡 啓

平成 28 年 2 月 9 日

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 ソフトウェア工学講座

平成 27 年度 修士学位論文

クラスタリングを用いたベイズ学習モデルを動的に更新するソフトウェア障害検知手法

爲岡 啓

内容梗概

ウェブサーバ群に起きうる障害を未然に発見することは，システムの可用性向上の観点などから重要である．既存研究として，ウェブシステムから収集したメトリクス群に対して，解析技術を用いて処理することによって，管理者の知識や経験に依存せず障害検知を行う手法が提案されている．しかし，そのような手法は，大量のデータを学習する必要があり，学習に膨大な時間がかかるため，効率的に障害検知を行うのは困難であった．

そこで，クラスタリングとベイズ学習モデルという，機械学習を用いた 2 つの解析技術を組み合わせて，メトリクス群を効率的に処理する手法について研究を行った．その結果，より少ないデータ量で，全データを学習した場合と同様の効果を得られる障害検知を行うことができることがわかった．

本論文では，実用化の観点からその手法を発展させ，クラスタリングによる学習用データ選定を自動化することで，状況に即してベイズ学習モデルを動的に更新する障害検知手法を提案する．また，手法を適用したシステム開発を行い，その出力結果に対して評価を行った．その結果，ウェブシステムに対して，逐次的に精度の高い障害検知を行えることをがわかった．

主な用語

障害検知
クラスタリング
ベイズ学習モデル

目 次

1　はじめに

社会基盤としてのウェブシステムは，不特定多数の利用者にサービスを提供す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，クライアント数の増加によって，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．たとえば，障害の原因として，ウェブサーバへの過負荷や，セキュリティの脆弱性を突いた攻撃などが挙げられる [32]．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，ウェブシステムの長期安定稼働に必要である [31]．

現状のウェブシステムにおいては，システム管理者がサーバマシンのメモリや CPU など少数のメトリクスの推移を確認して，一定時間内にシステムに障害が発生するかどうかを判断する [5]．システム管理者の扱うデータ量は，システム規模の増大に伴い今後増加の一途を辿ると考えられる．不具合が発生するとき，個々のメトリクスが非常に複雑に連携している．したがって，膨大なデータログを人間が処理し，不具合が起こるかどうかという結論を導き出すことは困難であり，それは人間の処理限界を超えている [7]．また，人間が限られた時間内に確認できる一部のデータに基づいた判断は推測の域を出ず，経験者の勘に頼るところが大きい．さらに，手動で行うことは時間と労力を要し， かつ常に可能であるとは限らない [2]．

近年，収集した大量のデータを，機械学習等の統計処理を施すことで，複雑に絡み合ったメトリクス間の関連をモデル化し，システムの異常判定，ひいては，異常の根本原因究明に活用する手法が提案されている．しかし，そのような手法の多くは，膨大な学習データを必要とするため，高精度，高効率を両立する障害検知を行うことは困難である．

私の所属する研究グループで行っている過去の研究において，クラスタリングとベイズ学習モデルという，機械学習を用いた 2 つの解析技術を組み合わせて，メトリクス群と障害発生の関係を学習することにより，より少量の学習データを用いて，全学習データを用いた場合と同様の効果が得られる障害検知を行う手法を考案した．また，その手法の評価を行い，効率良く障害検知が行えていることを確認した．

そこで，本論文では，手法を実在するウェブシステムに適用することを目的として，学習データ選定の工程を自動化し，逐次的な障害検知を行うことができる手法を提案した．学習データが少量であれば，ベイズ学習モデルにおける計算時間は短くなるため，短いサイクルでの再学習が可能となり，より現状に即した検知が行えると考えた．それを実証するため，学習データとして組み入れるかどうかを判断するプログラムを含む，全工程を自動化した検知システムを開発した．このシステムをウェブシステムに対して適用し，出力結果に対して評価を行った．10 回の実験に対する，全学習データを用いた場合と提案手法を用いた場合

の結果の比較を，一定時間継続する障害に対して，どちらが正しく検知を行えているか，というよりより実用的な観点で行った．評価の結果，本手法を用いた検知システムが，実用性の高いものであるということを確認した．

以降 2 章では，本研究の背景と，機械学習を用いた既存のデータ解析技術，また，その問題点について説明する．3 章では，提案手法の有効性を確認する予備実験について説明する．4 章では，提案手法とその実験内容，結果について説明し，その評価を行う．5 章ではまとめと今後の課題について述べる．

2　研究背景

2.1　ウェブシステムにおける障害

社会基盤としてのウェブシステムは，不特定多数の利用者にサービスを提供す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ている．そのため，このようなシステムにおいては，保守に関する規定を厳密に示した Service Level Agreement(以下 SLA) が存在する．SLA とは，サービスを提供する事業者が契約者に対し，どの程度の品質を保証するかを明記した合意書であり，混雑時の通信速度や処理性能の最低限度や，障害やメンテナンス等による利用不能時間の年間上限等，サービス品質の保証項目を定めている．また，それらを実現できなかった場合の利用料金の減額等の補償規定を含む [34].

例えば，NTT の提供する VPN サービス，Arcstar Universal One においては，サービス利用者との間に SLA が定められている [33]．ここでは，ある対象区間の平均伝送遅延時間を指標とし，これが基準値を超えた場合、利用料金を利用者に返還するという規定を設けている．具体的には，月単位で，ある対象区間の平均伝送遅延時間が 35msec を上回った場合，料金の 10%が利用者に返還される．このように，システム障害とは，システムが完全に利用停止状態となった場合だけでなく，最低品質を保証することによって，それ以外の状態を障害と定義する場合がある．

2.2　障害検知

先の例のように，利用者との契約において厳密に定義された障害を検知することが，ウェブシステムに求められる重要な課題である．障害を検知する方法として，大きく分けて，ハードウェア，ソフトウェアの，それぞれを対象とする 2 つのアプローチが存在する．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，系を切り替えることでシステムの稼働を維持する，といった方法を用いる．一方，ソフトウェアを対象とした障害検知においては，システムの管理者が，CPU 利用率やメモリの利用量，ディスク利用量，ネットワーク送受信量など，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，それらをもとに，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．本論文では，ソフトウェアを対象とした障害の検知に着目する．

2.3　ソフトウェアの障害検知における問題

システム管理者の扱うメトリクスの量，種類は，システム規模の増大に伴って増加していく．また，不具合が発生するとき，個々のメトリクスが非常に複雑に連携しているため，膨

大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困難であり，それは人間の処理限界を超えている [7]．人間が限られた時間内に確認できる一部のデータに基づいた判断は推測の域を出ず，経験者の勘に頼るところが大きい．さらに，手動で行うことは大きな時間と労力を要する [2]．そこで近年，収集した大量のデータを，機械学習等の統計処理を施すことで，複雑に絡み合ったメトリクス間の関連をモデル化し，システムの異常判定，ひいては，異常の根本原因究明に活用する手法が提案されている．

2.4 データ解析技術

これまでに提案されている，機械学習による障害検知に用いられる技術として，クラスタリングとベイズ学習モデルという，2 つのデータ解析技術を紹介する．

2.4.1 クラスタリング

クラスタリング (以下 CL) は，観測対象の過去の状態をグルーピングし，いくつかのクラスタに分割する方法である [13]．入力 (学習) データとして正常時のデータのみを与え，最近傍クラスタからの距離が閾値を超える観測値を異常と判定する診断方法が知られているが，正常範囲の一部のみが学習データに含まれる場合，正常であるにもかかわらず異常と判定してしまう可能性がある．このため，異常データを含まず，かつ正常範囲内でなるべく広い範囲を含む入力データが必要となる．この要件が十分に満たされないと，検知漏れや誤報を引き起こす可能性が高くなる [21]．図 1 では，CL の概念図を示している．ここではメトリクス A，B，C の 3 次元空間において，正常時クラスタの重心と，現在の状態との距離計算を行っている．

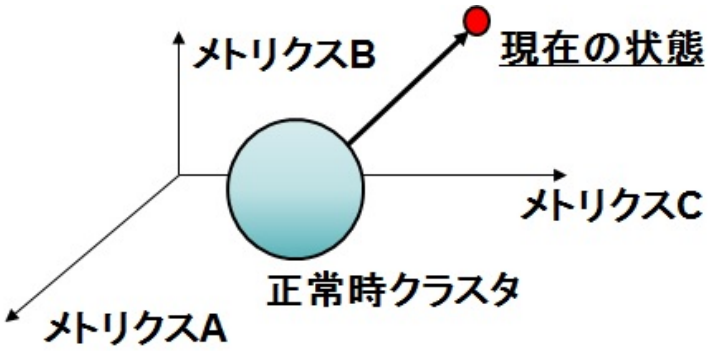

図 1: クラスタリングの概念図

本研究では，CPU 利用率，メモリ利用量など，n 種のメトリクスについて，時間変化の系列をデータとして取得し，n 次元空間上にプロットする方法をとる．正常時の点のみを含む点の集合において，その重心を中心とするクラスタの情報を持つモデルを生成する．このモデルを利用して，新たに取得した実時間データをプロットした点と，正常時クラスタの重心との距離をとることで，正常時との差を距離として算出することができる．

2.4.2 ベイズ学習モデル

ベイズ学習モデル (以下 BN) は，個々の事象の因果関係を条件付き確率で表すモデルで，観測対象の過去の状態を学習し，観測対象がある状態にある時の注目事象 E の発生確率 $P(E)$ を算出する事ができる [14]．例えば，障害検知においては,「応答時間が 3 秒以上となる」という事象の発生確率を算出する，という方法で利用される．ただし，学習データに含まれる過去の状態と同じとみなされる既知の状態での注目事象の発生確率は計算できるが，学習データに含まれない未知の状態下での注目事象の発生確率は正しく計算できない．注目事象の発生確率を正しく計算するためには学習データの量を増やすことが効果的である [5][19]．しかし，データ量が増えると学習処理にかかる時間が増大し，実用的な時間で完了しなくなるという課題がある．学習データに含まれる観測項目数を一定にした場合，計測回数に応じて学習時間が長くなる [26]．図 2 では，BN の概念図を示している．事象 B，C の発生確率 $P(B)$，$P(C)$ に対し，事象 B，C と因果関係のある事象 A が起こる条件付き確率 $P(A|B,C)$ を，BN を用いて算出することができる．

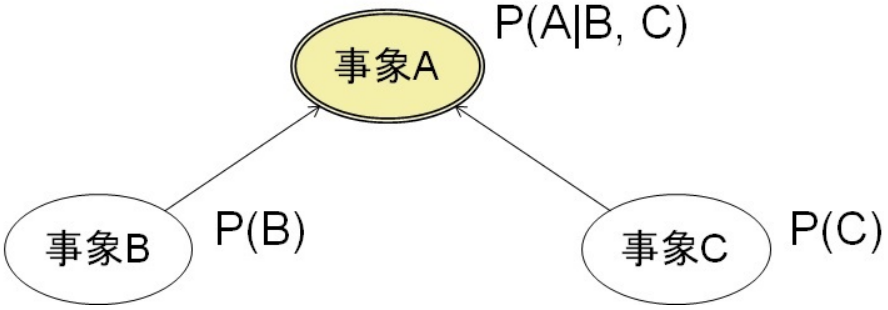

図 2: ベイズ学習モデルの概念図

たとえば，何らかの要因で，Web サーバの CPU 利用率が 100%に近い値を示すと，処理性能が劣化し，結果としてクライアントからのリクエストに対して応答する時間が遅延す

る．こういった因果関係を，データを入力することによってモデル化し，確率という人間にとって比較的理解しやすい指標で出力されるというのが BN の特徴である．

2.5　既存手法の問題点

CL，BN を含め，機械学習を活用する方法の多くは，期待する効果を得るために適切に学習データを選定しなければならない，という共通の課題がある．学習データは，効果があり，かつそのデータ量が少なければ少ないほど効率的である．現在は，学習パラメータの調整と合わせて，学習データの選定を人間が試行錯誤で行う事で，この問題に対処している [16][15][5][19][12].

2.6　所属する研究グループにおけるこれまでの成果

このような既存手法の問題に対処するため，私が所属する井上研究室では研究グループを組織して研究を行っている．これまで研究グループでは，学習データの選定を自動化する手法を提案した [20]．BN の学習データに，同じ状態が繰り返し出現する回数を減らしつつ，異なる状態を多数含めることができれば，より少ないデータ量で，全データで学習した場合と同等の学習効果を得る事ができると考えた．正常時を学習した CL モデルを用いた判断プログラムを用いて，BN に追加学習する必要のあるデータかどうかを判断し，学習が必要なデータのみを再学習する，という一連の動作を繰り返せば，より少量で，BN にとって未知の情報を多く含む学習データを作り上げることができるという仮説を立て，この手法の実装，評価を行った結果，手法の効果が実証された．この実証実験では，検知システムによる処理を実験終了後に一括で行う診断を行っており，取得したデータをリアルタイムに処理する逐次的な診断は行っていない．

そこで，本論文では，手法を実在するウェブシステムに適用するため，逐次的な障害検知を行うことができる手法を提案した．学習データが少量であれば，ベイズ学習モデルにおける計算時間は短くなるため，短いサイクルでの再学習が可能となり，より現状に即した検知が行えると考えた．それを実証するため，学習データとして組み入れるかどうかを判断するプログラムを含む，全工程を自動化した検知システムを開発した．このシステムを実際にウェブシステムに対して適用し，出力結果に対して評価を行った．

3　予備実験

逐次的な障害検知手法では CL，BN を組み合わせたとき，高効率であるという前提が不可欠であるため，その効果を実証するための予備実験を行う．

3.1　実験内容

まず，正常時のデータを CL に初期データとして与えてモデルを生成し，診断用のデータを入力として，その距離を算出する．距離が大きい区間，すなわち，CL が異常と検出した区間のデータのみを選定し，BN に学習させることで，検知精度を維持したまま，学習データ量を削減できるはずである．この考えに沿って実験を行い，また，検知精度を維持できているかどうか評価を行う．方法としては，CL が異常と判定した区間を含む，様々なパターンの区間を学習データとして，BN の出力確率を算出する．これらの出力に対して，性能劣化と強い因果関係を持つ平均応答時間との相関係数を計算することによって，学習データについての優劣の比較を行い，CL による異常判定が適切であることを示す．

3.2　検知システムの入出力手順

本予備実験のために開発した検知システムについて，その入出力の手順を以下に示す．

手順 0

初期データとして正常時のデータの系列を与え，それらがプロットした点の集合のクラスタの情報を持つ CL モデルを予め生成しておく．

手順 1

学習データの系列と CL モデルを距離演算器に入力し，学習データの時刻ごとのクラスタとの距離を演算し，演算結果を出力する．

手順 2

演算結果として取得した距離を見て，予め設定した閾値を超える時刻のデータを選び，それらの系列を選定データとする．なお，この作業は手作業で行っている．

手順 3

選定データを学習器に入力し，メトリクス同士の因果関係の情報を持つ BN モデルを生成する．

手順 4

診断したいデータの系列と BN モデルを分類器に入力し，時刻変化に対する障害発生確率の系列を出力する．

手順 2 における選定によって削減されるデータは，正常時の状態であり，正常時には同じ状態が繰り返し出現していると考えられるため，BN モデルに及ぼす影響は少ないと考える．よって，手順 4 における出力結果は，全学習データを利用した場合との強い相関が見られると予想される．手順をまとめたものを図 3 に示す．これらの手順によって，診断データを入力することで，障害の発生確率を出力することができる．

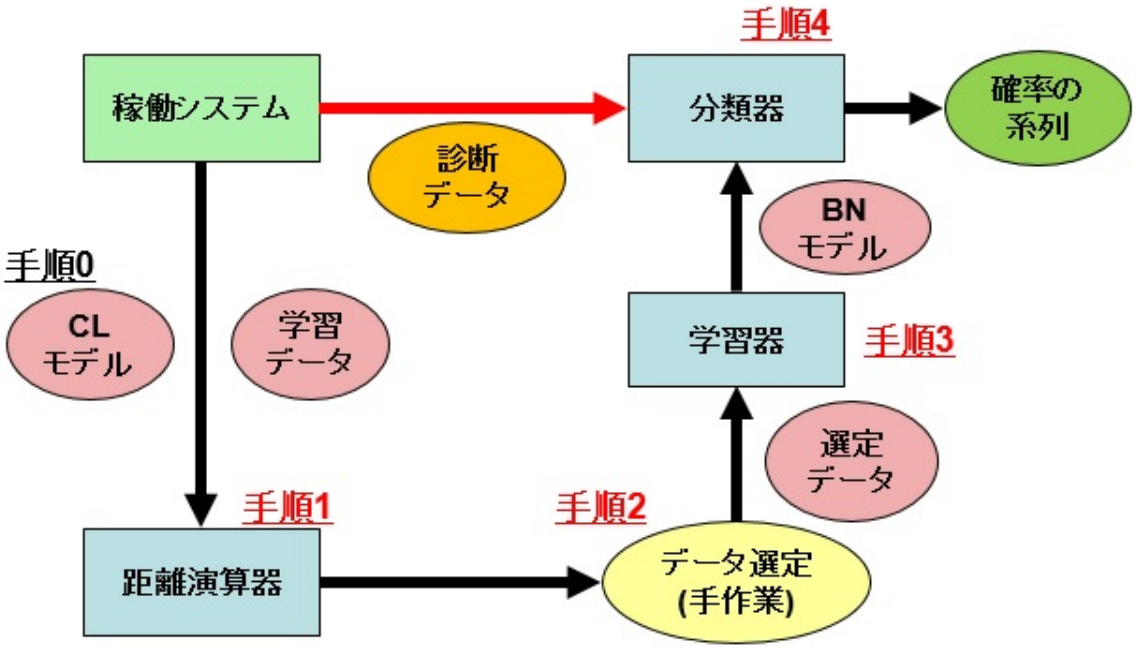

図 3: 入出力手順

距離演算器，学習器，分類器については，所属する研究グループにおいて開発を行った．距離演算器の実装に際して，CL 用の R 標準ライブラリ k-means[9]，学習器の実装に際して，BN 用の統計解析環境 R 用ライブラリ bnlearn[26] を用いた．

3.3 稼働システムの環境

本予備実験における，稼働システムの環境について説明する．

稼働システム環境は，クライアント，ロードバランサ，ウェブサーバ，データベースの 4 つのコンポーネントで構成する．ウェブサーバは，一般的なウェブ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．

3.3.1 クライアント

ウェブサーバにリクエストを送信し，その結果を受信する．クライアント用のアプリケーションとして，Apache JMeter[23] を用いる．Apache JMeter 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．ウェブページを指定して，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擬似的につくりだすことができる．

3.3.2 ロードバランサ

クライアントから送られたリクエストをウェブサーバへと中継する．複数のウェブ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．ロードバランサの実装ツールとしては，Apache の mod_proxy_balancer[24] を用いる．mod_proxy_balancer は，Apache が提供する負荷分散に利用するモジュールであり，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 台のウェブサーバに対して，均等に負荷分散を行っている．

3.3.3 ウェブサーバ

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．サーブレットコンテナとしては Apache Tomcat[25]，サーバアプリケーションとしては iBatis JPetStore[30] を利用する．Apache Tomcat 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り，これを利用することによって，サーバ上で HTML などのウェブ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 ウェブページとして表示することができる．

3.3.4 データベース

ウェブ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，その結果をウェブサーバへと返す．データベースシステムとしては，MySQL を利用する．MySQL とは，Oracle[29] 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 のデータを登録し，ウェブ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た図 3.3.4 を以下に示す．

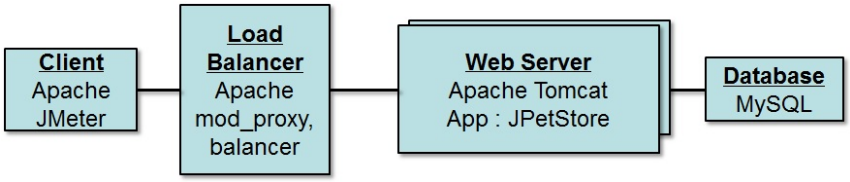

図 4: ウェブシステム全体図

本予備実験では 4 台の仮想計算機を利用した．そのうち 3 台については，CPU 1 コア，メモリ 1GB，HDD 20GB，1Gbps ネットワークとし，ウェブサーバ，ロードバランサとした．データベース用計算機については，CPU 2 コア，メモリ 4GB，HDD 110GB，1Gbps ネットワークとした．OS は全計算機において，Linux(CentOS 6.4) を採用した．これらの仕様をまとめたものを表 1 に示す．

表 1: 各仮想計算機の仕様

	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	
ウェブサーバ A	1core	1GB	30GB	Apache Tomcat 6.0.24-57	
ウェブサーバ B	1core	1GB	30GB	iBatis JPetStore 4.0.5	
データベース	1core	4GB	110GB	MySQL 5.1.69-1	

各仮想計算機には CPU，メモリ，ネットワーク，Disk の利用状況データを 10 秒間隔で収集するための監視エージェント，collectd[27] が組み込まれており，観測データを取得することができる．クライアントにおいては，例えばデータベース上の商品を検索し，商品をカートに入れ，チェックアウトを行う，などといった，あらかじめ与えたシナリオと負荷パターンに応じて，ユーザの挙動を模擬するリクエストを発生させる．

3.3.5 収集対象メトリクス

収集対象となるメトリクスの一覧を表 2 に示す．本実験では，ロードバランサで表 1 における 1 から 3 までと 5 の 4 項目，2 台のウェブサーバそれぞれで 1 から 4 までの合計 8 項目，データベースサーバで 1(2 コア分) と，2, 3, 4, の合計 5 項目，計 17 項目について，システ

ム監視の際よく利用されるメトリクスを収集した．

表 2: 監視対象メトリクス

	リソース名	監視対象
1	CPU(データベースのみ 2 つ)	利用率 (%)
2	メモリ	利用量 (bytes)
3	ネットワーク	送受信量 (bytes/sec)
4	Disk(ロードバランサ以外)	I/O オペレーション数 (ops/sec)
5	Web Access (ロードバランサのみ)	リクエスト数， 最大，平均応答時間

3.4 評価

CL を用いて学習データの選定を行った場合の BN の出力確率を全学習データを用いた場合と比較し，その検知精度が劣化していないことを示すため，BN の出力結果である障害発生確率の評価を行う．

3.5 障害発生確率の優劣

大規模システムにおける障害のひとつとして，その性能劣化が挙げられることが多く，クライアントのリクエストに対するウェブサーバの応答時間は，性能劣化と強い因果関係を持つ．ゆえに，BN による障害発生確率の時間変化と，その時間変化する区間における平均応答時間との相関係数を計測することによって，学習データについての優劣の比較を行うこととした．

平均応答時間と確率の時間変化との相関係数 $COR(0 \leq COR \leq 1)$ はそれぞれのデータの系列の 1 要素を x, y として，以下の式で与えられる．

$$COR = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

相関係数 COR の値によって，学習データの優劣を決定する．

一般に，学習データの量は大きければ大きいほど，注目事象の発生確率は正しく計算されるため，より長期間の学習データを用いた出力結果のほうが優れているはずである．CL で選定した学習データを用いた場合の COR が他の学習データを用いた場合よりも高い値を示せば，CL によって，より少ないデータ量で高精度な検知を行えることを示せる．

3.6　評価手順

BN による障害発生確率の評価方法を手順として以下に示した．

手順 1

ウェブシステムから取得した学習用データを，時間変化に対していくつかの区間に分割する．分割した様々な区間を基にして，連続区間のパターンを作成し，これらの学習データ群全てについて，それぞれ BN モデルを生成する．このパターンの中には，CL で選定した場合に取得できる学習データも含まれる．

例えば，15 分間の実験を行い，5 分ごとにデータを 3 つに分割したとき，3 つの区間に対しては区間 1，2，3 それぞれのみと，区間 1-2，区間 2-3，区間 1-3(全学習データ) の 6 パターンがある．CL による選定プロセスを経て得られた区間が区間 1-2 であったとき，それはこの 6 パターンに含まれる．

手順 2

BN モデル群それぞれを分類器に入力し，障害発生確率群を出力する．

手順 3

それぞれの障害発生確率と実験時の平均応答時間の系列との相関係数を取り，それらの順位付けを行う．その結果，学習データの優劣を決定することができる．

評価手順をまとめたものを図 5 に示す．

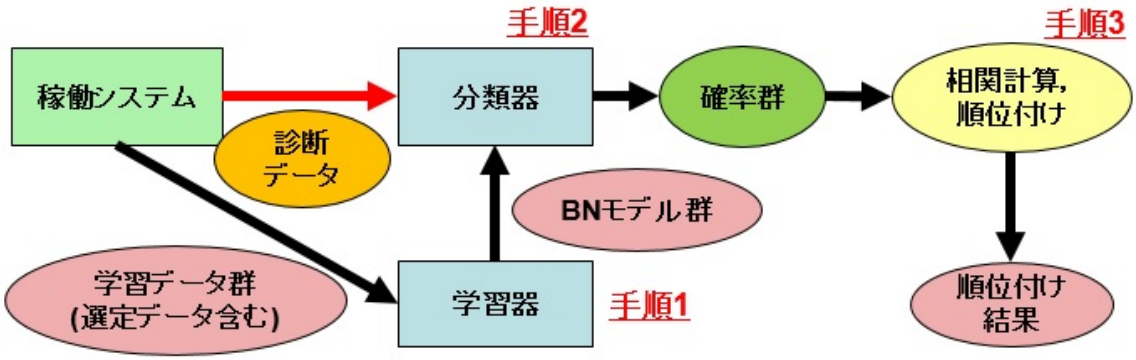

図 5: 評価手順

3.7　実験方法

評価実験は 35 分単位で行い，5 分ごとに区間を分け，区間 1-7 を定める．ウェブサーバ上で稼働する JPetStore へ，クライアントからの擬似的なバックグラウンドトラフィックを与えながら，システムの異常状態を模擬する負荷を，ウェブサーバに与える．Apache JMeter を用いて，クライアントからサーバへ継続的にリクエストを発生させた状態で，システムの異常状態を模擬するために stress コマンド [30] にて負荷を与え，データ収集を行う (図 6)．また，学習用データには 7 つの区間があるため，それらの連続区間のパターン総数は 28 である．これら全てのパターンに対して BN での障害発生確率を算出し，平均応答時間との相関係数を取る．

学習区間と実験における負荷パターンについてまとめ，図 7，表 3 に示した．学習データ，診断データいずれも，このパターンの負荷を与えたものを収集する．

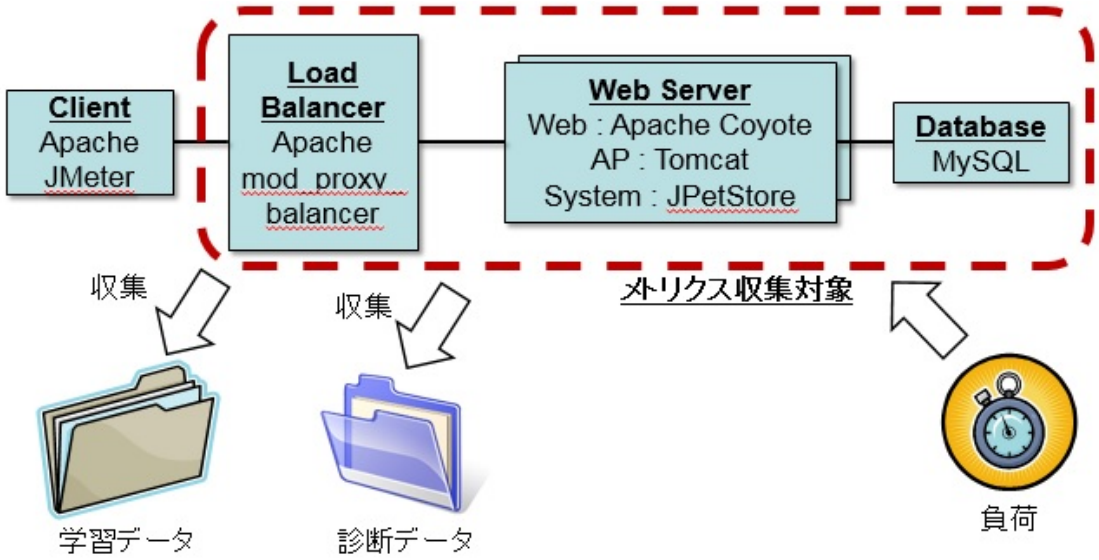

図 6: データ収集方法

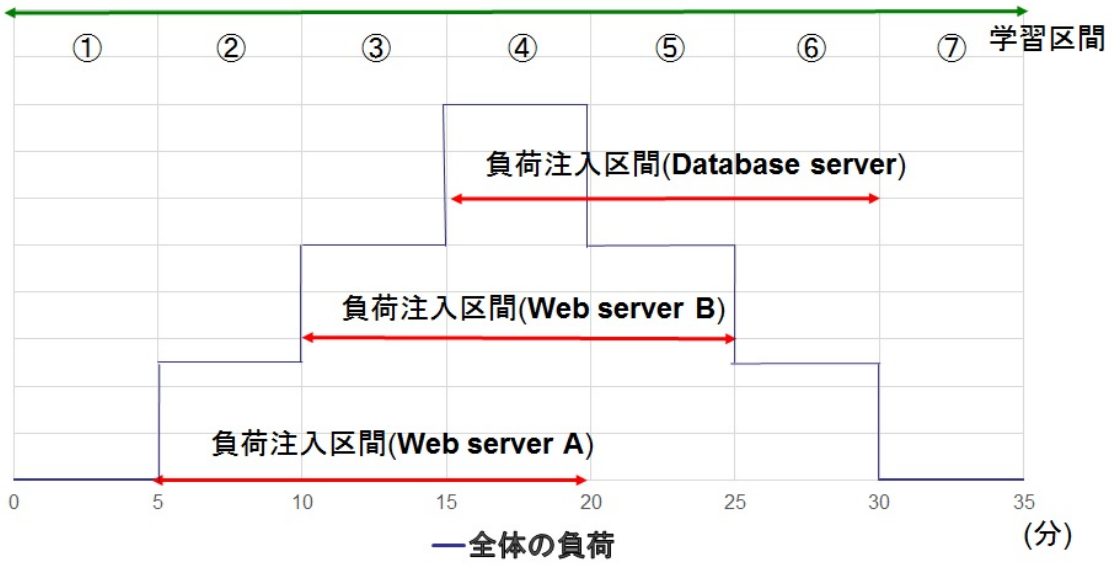

図 7: 負荷パターン

表 3: 実験プロセス

時間	要件
0：00-5：00	負荷注入なし
5：00-10：00	ウェブサーバ A に負荷注入
10：00-15：00	ウェブサーバ A，B に負荷注入
15：00-20：00	ウェブサーバ A，B，DB に負荷注入
20：00-25：00	ウェブサーバ B，DB に負荷注入
25：00-30：00	DB に負荷注入
30：00-35：00	負荷注入なし

3.8 初期データと閾値の設定

CL，BN それぞれについて，モデル生成の際に予め決定すべき事項が存在する．

CL については，正常時クラスタを生成するための初期データと，学習データの選定を行う際，CL が出力した距離に対して，正常か異常かを判断する閾値が必要である．

本予備実験では，初期データとして，クライアントからのリクエストのみがトラフィックとして存在する区間 1 のデータの系列を，正常時データとして組み入れることとする．また，実験環境において，ウェブサーバに負荷を与えてデータ収集を行った結果，CL モデルの算出距離が 500 を超えるとき，顕著に学習データとの差異を検出している，すなわち，異常を検出していることがわかった．そこで，本研究では，CL モデルの算出距離 500 を閾値とし，この閾値を超えた区間のみを，BN の学習データとして用いることとする．

BN については，事象の発生確率を計算するため，注目事象が何であるかを決定しなければならない．実験環境において，ウェブサーバに負荷を与えてデータ収集を行った結果，ある時刻における最大応答時間が 3 秒を超えた場合に，クライアントからのリクエスト処理能力が著しく低下していることがわかった．そこで，本予備実験では，最大応答時間が 3 秒を超えたとき，システムに障害が発生したと判断することとし，最大応答時間が 3 秒を超える確率を算出することとした．

3.9 実験結果

まず，図 7 のように負荷をかけた時の，診断用データにおけるリクエスト数，平均，最大応答時間の時間変化を図 8 に示す．グラフの○点が 10 秒間のリクエストの最大応答時間 (左軸 : sec)，△点が同 10 秒間の平均応答時間 (左軸 : sec)，＋点が同 10 秒間の処理リクエスト数 (右軸 : request/sec)，横軸は経過時間 (秒) を表す．グラフタイトルの MA=5 はグラフ上の点が隣接 5 データの移動平均値である事を表す．システム全体に最大負荷のかかる区間 4 を中心として，区間 2-6 に最大応答時間の大きな増加が見られる．

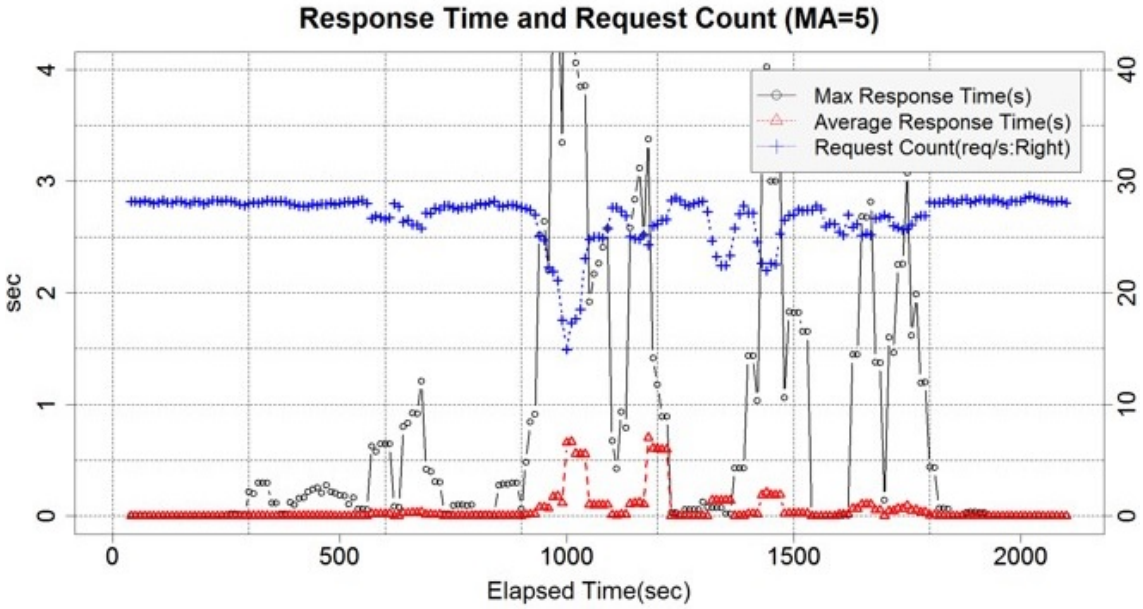

図 8: リクエスト，応答時間の時間変化

以下，検知システムの入出力手順 (3.2 章) に従い，実験結果の例を示す．

手順 1，2

学習区間を区間 1 のみとした時の BN，CL における学習データの診断結果を図 9 に示す．グラフの○点が BN による最大応答時間が 3 秒を超える確率 P(左軸)，△点が CL による最近傍クラスタからの距離 D(右軸)，横軸は経過時間 (sec) を表す．正常時のみを学習データとして組み入れているため，BN モデルは確率として 0 を継続出力している．区間 3-6 は CL モデルの閾値 500 を超えているため，この区間を選定データとする．

手順 3，4

次に，区間 3-6 を選定データとして学習器に入力し，BN モデルを生成し，診断データと共に分類器に入力した結果を図 10 に示す．各軸，値の見方については図 9 と同様である．

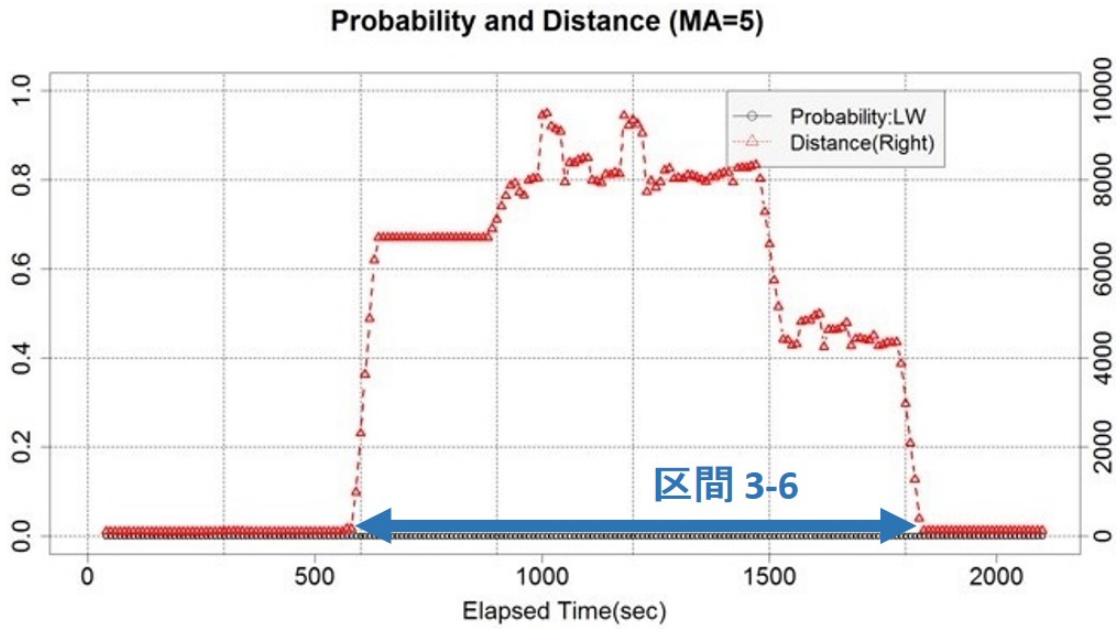

図 9: BN，CL による診断結果 (区間 1-1)

異常時を学習データとして組み入れたため，BN モデルの出力確率は図 8 における応答時間の変化に対応して変化している．BN モデルの出力確率と平均応答時間との相関係数を計測したところ，$COR = 0.88$ となった．

3.10 評価結果

評価手順 (3.6 章) の手順に従い，評価を行った結果を示す．連続区間の 28 パターン全てに対して障害発生確率と平均応答時間との相関係数を計測した．それらを値の高い順に並べた表 4 を示す．

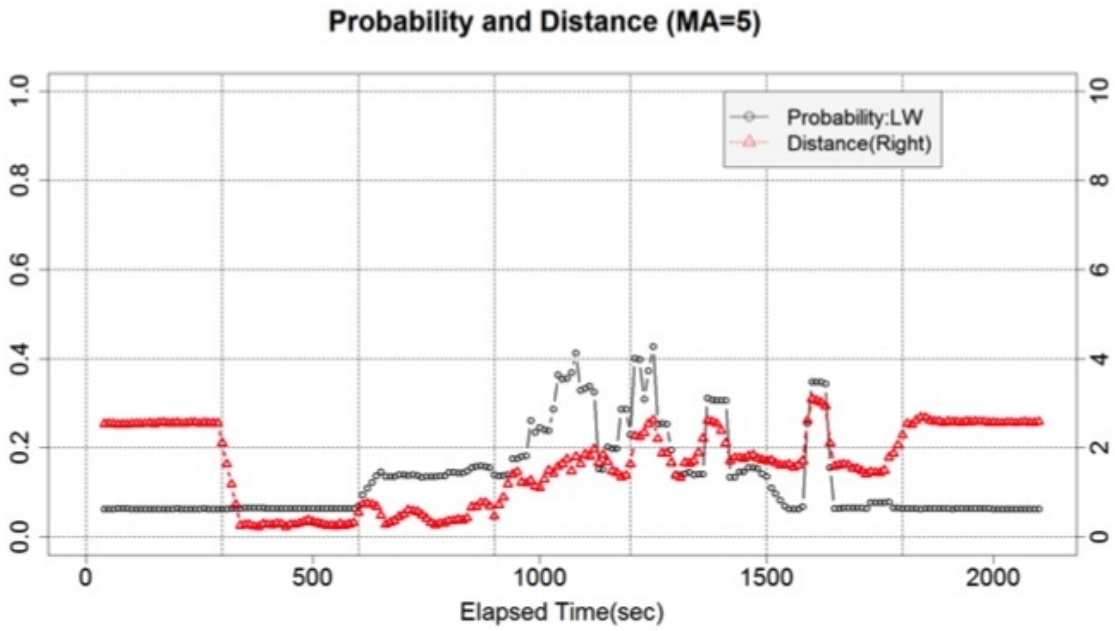

図 10: CL による選定を行った BN 診断結果 (区間 3-6)

表 4: 相関係数による評価結果

順位	区間	相関係数	区間数
1	2-7	0.907	6
2	1-7	0.893	7
3	1-6	0.892	6
4	2-6	0.89	5
5	3-7	0.888	5
6	**3-6**	**0.88**	**4**
7	4-6	0.848	3
8	5-5	0.826	1
9	4-5	0.805	2
10	3-5	0.794	3
…	…	…	…
27	6-7	0.138	2
28	1-1	0	1
平均	-	0.684	3

一般的には，学習データの量は多ければ多いほど，注目事象の発生確率は正しく計算されるため，区間数 4 の学習データの順位は，より区間数の多い学習データの数が 6 つであることから，7 位以下となるはずである．しかし，提案手法を用いた学習データは 6 位となったため，少ないデータ量で，学習データとしてより高い効果を発揮すると言える．

3.11　全学習データとの診断比較

最後に，全学習データ (区間 1-7) を用いた場合の BN の出力確率と，CL の選定したデータ (区間 3-6) を用いた場合の BN の出力確率の相関係数を取り，同等の効果をあげているかを検証する．

区間 1-7 全てを学習データとした，BN，CL の診断結果を図 11 に示す．各軸，値の見方については図 9，図 10 と同様である．

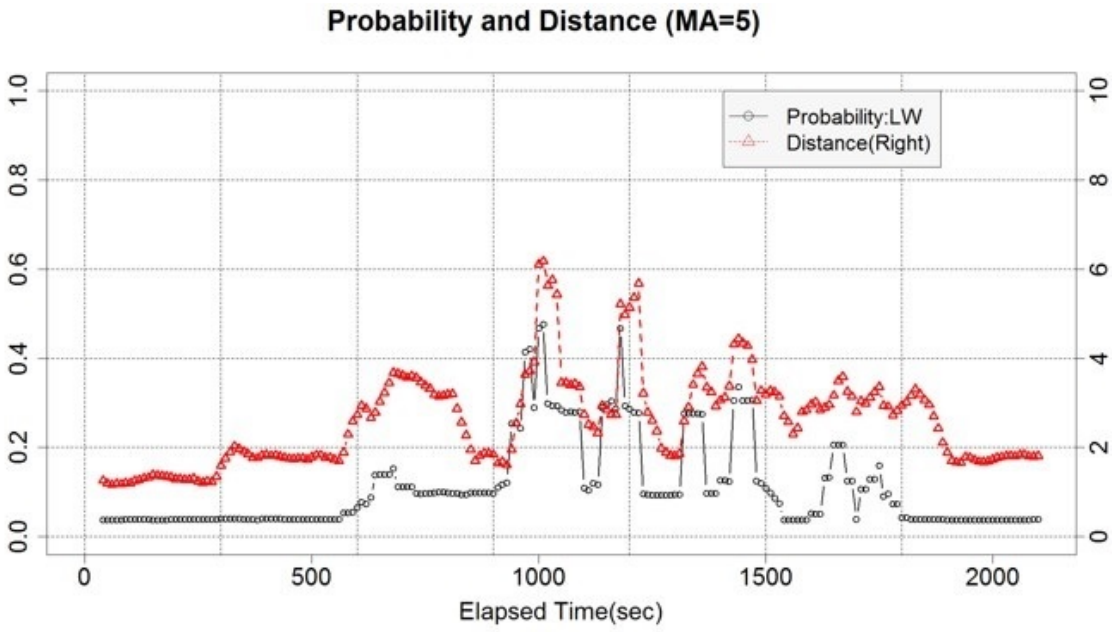

図 11: BN，CL による診断結果 (区間 1-7)

それぞれの算出確率 (図 10，11) の時間変化について，相関係数を算出したところ，$COR = 0.993$ と，高い相関値を見せた．区間数が減っても同等の結果が得られたということで，提

案手法に対して検証することができた.

4　提案手法

予備実験を行った結果，CL，BN を組み合わせる方法が障害検知において効果的であることを実証した．この事実をふまえ，本章では，逐次的障害検知を行うための手法を提案する．

4.1　逐次検知システムの入出力手順

本手法における入出力の手順を以下に示す．これらの手順は全て自動で行い，毎分 BN による障害発生確率の出力が行われる．

手順 0

初期データとして正常時のデータの系列を与え，それらがプロットした点の集合のクラスタの情報を持つ CL モデルを予め生成しておき，判断プログラムに利用する．

手順 1

学習用データセットの系列と CL モデルを判断プログラムに入力する．判断プログラムは，学習データセットが BN の再学習に必要かどうかを判断する．必要でないと判断した場合は入力を行わない．

手順 2

判断プログラムによって必要であると判断された学習データセットを蓄積された学習データに組み入れ，それを学習器に入力し，メトリクス同士の因果関係の情報を持つ BN モデルを生成する．

手順 3

診断したいデータの系列と BN モデルを分類器に入力し，時刻変化に対する障害発生確率の系列を出力する．

手順 1-3 は全て自動化し，一定時間ごとに学習用データ，診断用データを取得し，確率の系列出力を行う．手順をまとめたものを図 12 に示す．

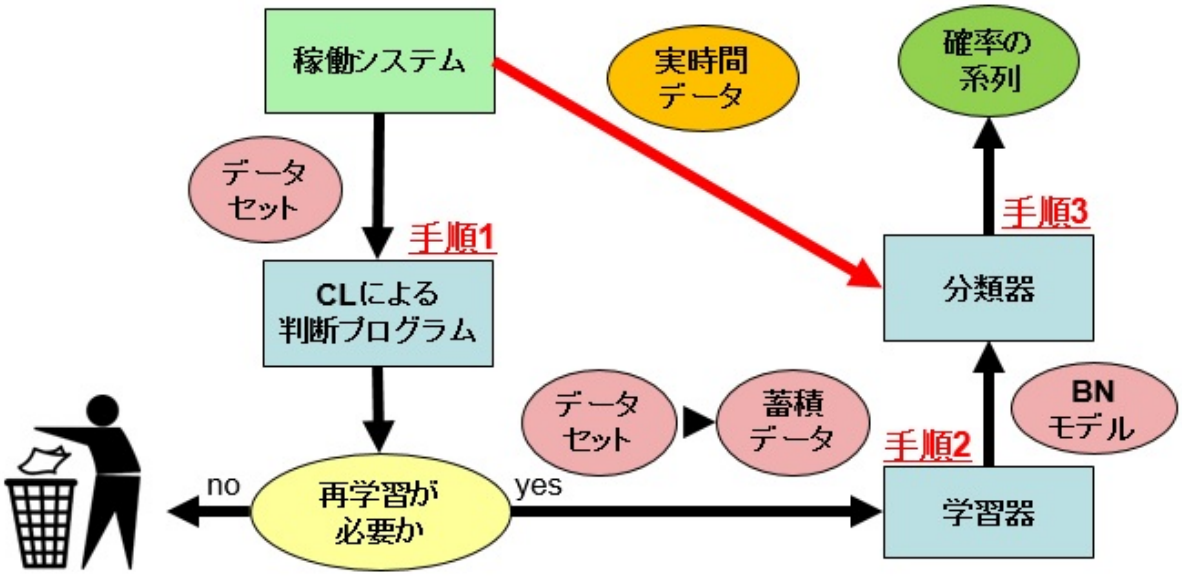

図 12: 提案手法の手順

4.2 データセット

学習用のデータセットについて説明する．実験環境において，監視エージェントとして利用している collectd では，10 秒毎にロードバランサ，ウェブサーバ，データベースそれぞれからデータ取得を行っている．手順 1-3 の処理時間は，学習データ量の削減によって高々30 秒程度に抑えられることがわかったため，1 分毎にデータセットを取得し，処理を行うことができる．そのため，1 分間の学習データ系列 6 要素を学習データセットとし，これを逐次障害検知システムの入力とする．

4.3 蓄積データ

CL による判断プログラムで再学習に必要と判断されたデータは蓄積データとして保存し，それ以降の診断に用いる．

4.4 判断プログラム

CL による判断プログラムが行う処理について説明する．

判断プログラムは，学習データの系列と CL モデルを入力として，その学習データが必要であるかを判断するプログラムである．2 つの入力を用いて，正常時クラスタとデータ系列のそれぞれの要素に対して距離演算を行い，その距離が閾値を超えている要素が多数見られた場合，その系列は学習に必要である，と判断される．例えば 1 つの学習データセットに対して距離演算を行った結果，6 要素全てが設定した閾値を超えていたとすると，その系列は学習に必要であると判断される．

4.5 実験

実験環境，実験内容については，予備実験と同様であり，35 分単位でウェブシステムに順次負荷をかける実験を行う (図 7)．

4.6 初期データと閾値の設定

3.8 章で説明したとおり，CL，BN にはそれぞれ初期データ，閾値が必要であるが，これらについても，予備実験と同様とする．

判断プログラムについては，1 つの学習データセットにおいて，CL の閾値 500 を超える要素が過半数 (6 要素のうち 4 要素以上) であった場合，BN の再学習に必要であると判断することとする．

4.7 実験結果

実験結果の一例を述べる．

まず，図 7 のように負荷をかけた時の，診断用データにおけるリクエスト数，平均，最大応答時間の時間変化を図 13 に示す．グラフの○点が 10 秒間のリクエストの最大応答時間 (左軸：sec)，△点が同 10 秒間の平均応答時間 (左軸：sec)，＋点が同 10 秒間の処理リクエスト数 (右軸：request/sec)，横軸は経過時間 (秒) を表す．

予備実験の結果と同様に，システム全体に最大負荷のかかる区間 4 を中心として，区間 2-6 に最大応答時間の大きな増加が見られる．

次に，全学習データを用いた場合の BN の出力結果を図 15 に示す．横軸は経過時間 (sec) を表し，グラフの○点，△点共に，BN による最大応答時間が 3 秒を超える確率 (左軸) を表す．○点はロジックサンプリング (LS) 手法を用いた結果，△点は尤度重み付け (LW) 手法を用いた結果であり，いずれも分類器に利用している R 言語のライブラリ bnlearn で利用

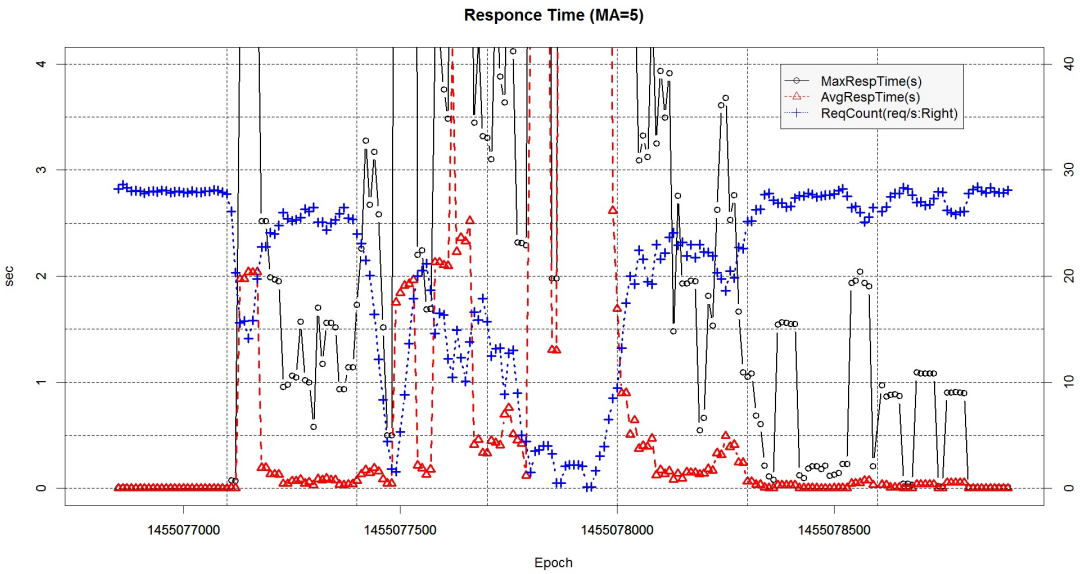

図 13: 実際にかかった負荷

できる．実験の結果，LW 手法の出力確率のほうが高精度であると判断し，LW 手法を採用した．

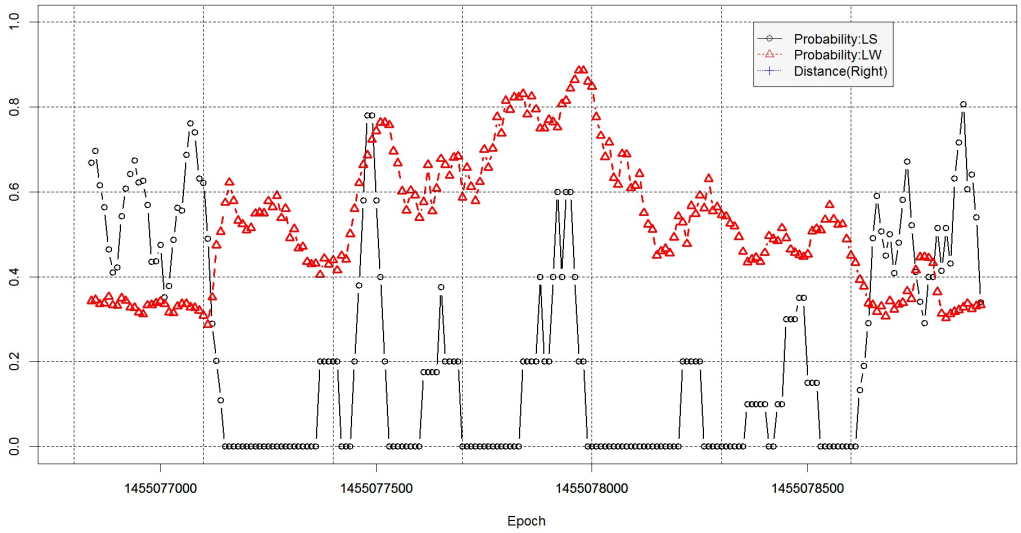

図 14: 提案手法による出力結果

LW 手法による出力確率は，図 13 における応答時間の変化に対応して変化しているが，システムに負荷がかかっているときとそうでないときの確率の値に顕著な差がみられない．そのため，実用化に際して，確率の閾値の決定が困難であると考えられる．

最後に，提案手法を用いた逐次検知システムの出力結果を図 15 に示す．各軸，値の見方については図 14 と同様である．

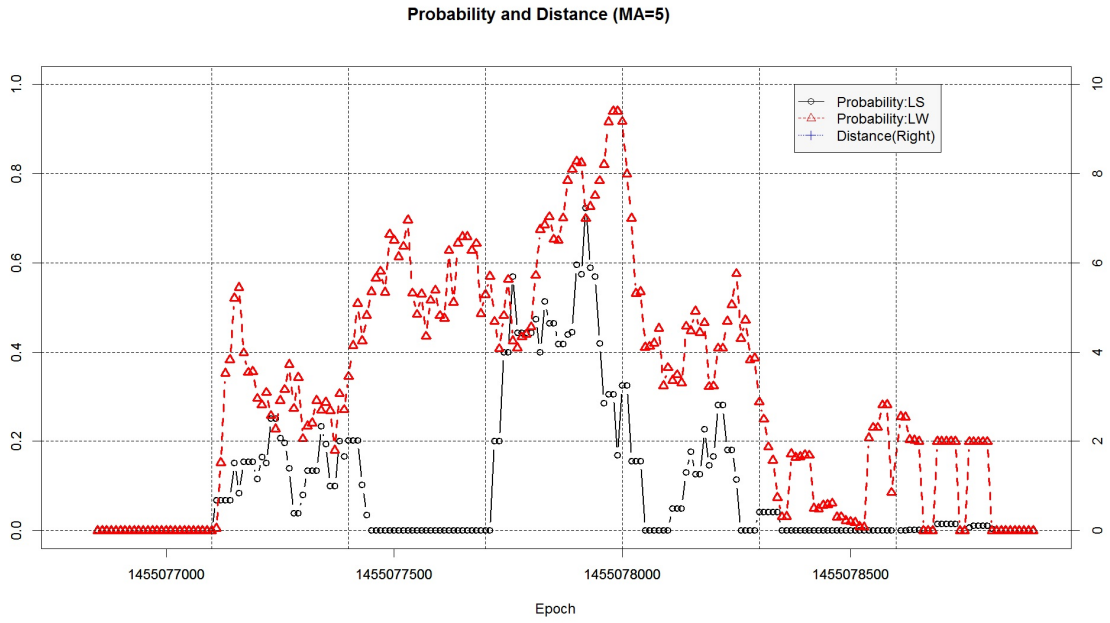

図 15: 提案手法による出力結果

LW 手法による出力確率は，図 13 における応答時間の変化に対応して変化している．全学習データを用いた場合に比べ，システムに負荷がかかっているときとそうでないときとで確率の値に顕著な差がみられるため，実用化に際して，確率の閾値の決定が容易であると考えられる．

4.8 評価

これらの結果に対して評価を行う．具体的には，10 回の実験に対する，全学習データを用いた場合と提案手法を用いた場合の結果の比較を行う．

4.8.1 比較方法

一定時間継続する障害に対して，どちらが正しく検知を行えているかを評価する．BN の出力確率に対して閾値を設け，まず，閾値を超えた時，その時刻から 1 分間をアラート区間とする．1 分後に閾値を超えている場合，アラート区間を 1 分延長する．また，障害についても，最大応答時間が 3 秒を上回ったとき，その時刻から 1 分間を障害発生区間とし，1 分後に最大応答時間が 3 秒を上回っている場合，障害発生区間を 1 分延長する．アラート区間と障害発生区間に重なりが見られれば，検知成功とする．

4.9 比較項目

検知成功の内容を細かく見るため，比較項目について，その数値を比較することとします．比較項目は障害発生回数，検知された障害数，アラート回数，正解数の 4 つである．

障害発生回数とは，最大応答時間が 3 秒を超えた回数である．検知された障害数とは，障害発生区間がアラート区間と重なった回数である．アラート回数とは，BN による診断確率が閾値を超えた回数であり，正解数とは，アラート区間が障害発生区間と重なった回数である．

計 10 回の実験結果に対してこの回数を調べ，検知された障害数に対する，障害発生回数の割合と，アラート回数に対する正解数の割合を算出し，その値を比較する．

検知された障害数と正解数の違いをわかりやすくするため，比較方法の例を図 16 に示す．

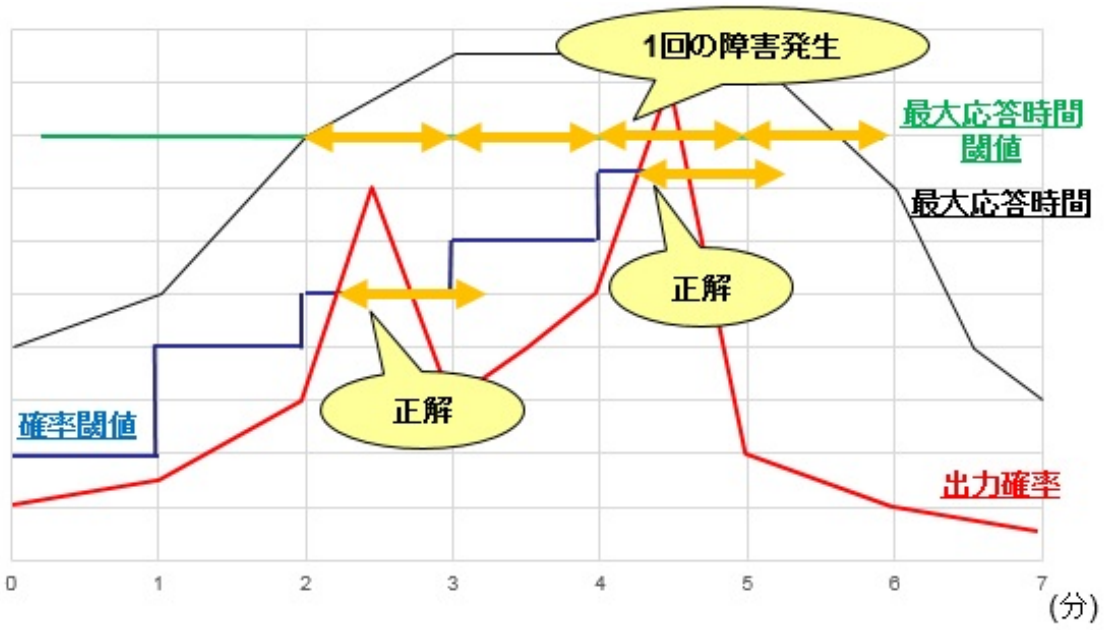

図 16: 評価方法の例

これは，BN による診断結果と最大応答時間のグラフを示している．横軸は時間変化を分単位で表す．時刻 $t = 2$(分) において，最大応答時間が閾値を超えているため，ここから 1 分間，障害発生区間が生じる．1 分後，2 分後と，最大応答時間は閾値を超え続けているので，障害は計 4 分間発生している．しかし，継続して発生しているため，この時の障害発生回数は 1 回と数える．また，出力確率が閾値を超えている 2 箇所は，継続していないので，アラート数は計 2 回と数える．さらに，障害発生中に出力確率が閾値を超えているため，正解数を数える．この際，障害発生区間は連続しているのに対し，アラート区間は連続していない．このような長区間の障害発生に対して，正解数を何回とするか，という問題がある．本研究で行った評価方法では，正解数は 2 回と数えることとする．逆も同様で，アラートが長時間鳴っている状態で障害が複数回発生した場合，それらは全て検知された回数として数えることとする．

4.9.1 評価結果

全学習データを用いた場合と提案手法を用いた場合を比較した評価結果を表 5 に示す.

表 5: 提案手法の評価結果

＼	全学習データ	提案手法
障害発生回数	107	
検知された障害数	60	90
検知障害数/発生回数	0.561%	**0.841%**
アラート回数	64	102
正解数	57	90
正解数/アラート回数	0.891	**0.882**

まず，アラート回数に対する正解数についての項を見ると．全学習データを用いた場合に対して，アラート回数に対する正解数の項目では多少劣るものの，102 回のアラートを発生させ，そのうち 90 回は障害発生に対応したものであった．また，障害発生回数に対する検知された障害数の項目では，大きな差をつけている．

さらに，正解数のうち，障害発生の 10 秒前から 0 秒前までに検知成功したものを直前検知回数とすると，全学習データを用いた場合では 22 回，提案手法を用いた場合では 48 回，直前検知が行われたことがわかった．

本システムでは，障害発生を防ぐことを最終的な目標としているため，これらの評価結果から，本手法を用いた障害検知システムが実用性の高いものであることが確認できた．

5　まとめと今後の課題

本論文では，手法を実在するウェブシステムに適用することを目的として，学習データ選定の工程を自動化し，逐次的な障害検知を行うことができる手法を提案した．学習データが少量であれば，ベイズ学習モデルにおける計算時間は短くなるため，短いサイクルでの再学習が可能となり，より現状に即した検知が行えると考えた．それを実証するため，学習データとして組み入れるかどうかを判断するプログラムを含む，全工程を自動化した検知システムを開発した．このシステムをウェブシステムに対して適用し，出力結果に対して評価を行った．具体的には，10 回の実験に対する，全学習データを用いた場合と提案手法を用いた場合の結果の比較を，一定時間継続する障害に対してどちらが正しく検知を行えているか，というよりより実用的な観点で行った．評価の結果，本手法を用いた検知システムが，障害検知に対して実用性の高いものであるということを確認した．

今後の課題としては，大きく 2 つ挙げられる．1 つ目に，評価実験の回数や，負荷パターンを増やすことが挙げられる．実際の障害に近い負荷を発生させることによって，様々な状況においての逐次的障害検知システムの有効性を確認できる．2 つ目に，システム管理者を対象としたユーザインタフェースの実装が挙げられる．本手法は実在するウェブシステムへの適用を目的としており，本手法を利用して，パラメータの設定，出力など，管理者が利用しやすいツールを実装することは，ウェブシステムへの適用に不可欠であると考える．

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．
本研究において，常に適切な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，植田良一様に深く感謝いたします．
本研究の遂行にあたり，実験環境における熱心かつ丁寧な御指導およびご助言を頂きました大阪工業大学情報科学部情報システム学科 Software Development and Analysis 研究室井垣宏准教授に深く感謝いたします．
本研究において，客観的な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，神田哲也様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] T. F. Abdelzaher, K. G. Shin, et al, “Performance guarantees for Web server endsystems: A controltheoretical approach”, IEEE Transactions on Parallel and DistributedSystems, 13(1), pp80-96, 2002.

[2] G. A. Alvarez, E. Borowsky, et al. “An automated resource provisioning tool for large-scale storage systems”, ACM Transactions on Computer Systems (TOCS), pp483-518, 2001.

[3] D. Arthur, B. Manthey, H. Roglin, “k-means has polynomial smoothed complexity”, 2009.

[4] C. Bernard, “An optimal convex hull algorithm in any fixed dimension ”, 1993.

[5] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, J.S. Chase, “Correlating instrumentation data to system state: A building block for automated diagnosis and control”, USENIX Association OSDI’04: 6th Symposium on Operating Systems Design and Implementation, 2004.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth “From Data Mining to Knowledge Discovery in Databases”, 2008.

[7] A. Fox and D. Patterson, “Self-repairing computers”, Scientific American, 2003.

[8] N. Friedman, D. Geiger, M. Goldszmidt, “Bayesian Network Classifiers”, Machine Learning Volume 29, Issue 2-3, pp131-163, 1997.

[9] N. Friedman, M. Linial, I. Nachman, D. Pe’er, “Using Bayesian Networks to Analyze Expression Data”, Journal of Computational Biology 7, pp601-620, 2000.

[10] S. Iwata, K. Kono, “Clustering Performance Anomalies Based on Similarity in Processing Time Changes”, IPSJ Transactions on Advanced Computing Systems, Vol.5 No.1 1-12, 2012.

[11] J. B. MacQueen, “Some Methods for classification and Analysis of Multivariate Observations”, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability,University of California Press, pp281-297, 1967.

[12] T.H.D. Nguyen, M. Nagappan, A.E. Hassan, M. Nasser, P. Flora, “An Industrial Case Study of Automatically Identifying Performance Regression-Causes”, MSR Hyderabad, India, 2014.

[13] Y. Okada, T. Sahara, S. Ohgiya, T. Nagashima, “Detection of Cluster Boundary in Microarray Data by Reference to MIPS Functional Catalogue Database”, The 16th Int. Conference on Genome Informatics, Japanese Society for Bioinformatics, Proc. of The 16th Int. Conference on Genome Informatics, Tokyo, Japan, 2005.

[14] J. Pearl, “Bayesian Networks, a Model of Self-Activated Memory for Evidential Reasoning”, Proceedings, Cognitive Science Society pp329-334, 1985.

[15] E. Stehle, K. Lynch, M. Shevertalov, C. Rorres, and S. Mancoridis, “On the use of Computational Geometry to Detect Software Faults at Runtime”, 7th International Conference on Autonomic Computing, ICAC, Washington, DC, USA, 2010.

[16] E. Stehle, K. Lynch, M. Shevertalov, C. Rorres, and S. Mancoridis, “Diagnosis of Software Failures Using Computational Geometry”, 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, Nov., 2011.

[17] R. Taylor, E. Rdcs. “Interpretation of the Correlation Coefficient: A Basic Review”, JDMS1, pp35-39, 1990.

[18] S. Thrun, C. C. Faloutsos, A. W. Moore, P. Spirtes, G. F. Cooper, “Learning Bayesian Network Model Structure from Data” 2003.

[19] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, A. Fox, “Ensembles of Models for Automated Diagnosis of System Performance Problems”, The International Conference on Dependable Systems and Networks, Yokohama, Japan, 2005.

[20] 植田 良一, 角井 健太郎, 爲岡 啓, 松下 誠, 井上 克郎, “Web サービスシステムの応答性能劣化診断のための学習データ自動選定方法”, 電子情報通信学会論文誌, Vol.J99-D, No.1, pp.100-108, 2016.

[21] 鈴木 英明, 内山 宏樹, 湯田 晋也, “データマイニングによる異常検知技術”, 日本オペレーションズ・リサーチ学会, pp.506-511, 2012.

[22] 爲岡 啓, 植田 良一, 松下 誠, 井上 克郎, “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システム障害検知システムの有効性検証”, 情報処理学会第 188 回 SE 研究発表会, pp1-8, 2015.

[23] Apache JMeter, https://jmeter.apache.org/.

[24] Apache mod_proxy_balancer, http://httpd.apache.org/docs/2.2/ja/mod/mod_proxy_balancer.html/.

[25] Apache Tomcat, http://tomcat.apache.org/.

[26] bnlearn - an R package for Bayesian network learning and inference, http://www.bnlearn.com/.

[27] collectd, http://collectd.org/.

[28] iBatis JPetStore, http://sourceforge.net/projects/ibatisjpetstore/.

[29] Oracle, http://www.oracle.com/.

[30] stress project page, http://people.seas.harvard.edu/-apw/stress/.

[31] 加藤 清志, “サービスの安定稼働を阻むサイレント障害”, http://thinkit.co.jp/article/1089/1.

[32] 谷 誠之, “年々難しくなる「障害検知」のコツ”, http://www.itmedia.co.jp/im/articles/1005/19/news106.html.

[33] VPN サービス Arcstar Universal One SLA, http://www.ntt.com/vpn/data/sla.html.

[34] Web 担当者 Forum, SLA とは, http://web-tan.forum.impressrd.jp/g/sla.