学位論文　博士（工学）

ウェブアプリケーションにおける
性能異常の検出と原因究明に関する研究

2011 年度

慶應義塾大学大学院理工学研究科

岩 田　聡

ウェブアプリケーションにおける
性能異常の検出と原因究明に関する研究

岩田　聡

論文要旨

近年，オンラインバンキングや証券取引など大きな責任を伴うサービスがウェブアプリケーションで提供されるようになってきた．しかし，システム障害が度々報告されているように，その信頼性は十分とはいえず，対策が求められている．システム障害の一つとして近年特に注目を集めているのが応答時間の増大やスループットの低下などの性能異常である．現在ウェブアプリケーションは非常に複雑になっており，性能異常の発生を事前に防ぐのは非現実的である．

本研究では，発生した性能異常から早期に復旧し，被害を最小限に抑えることを目的とする．そのためには異常発生をいち早く検出し，さらに原因究明に有益な情報を管理者へ提供する手法が必要になる．この異常検出，原因究明支援手法には，精度だけでなく導入する際のシステムへの変更を最小限に抑えることも要求される．なぜなら，近年システムの大規模化，複雑化が進み，システムに変更を加えるには専門的な知識と技術を要するためである．これまでは検出，原因究明の精度か導入の容易さかのどちらか一方を重視した手法が提案されてきており，2 点を共に満たす手法はなかった． 精度を重視した手法の例としてはコンポーネントごとに処理時間を監視する手法がある．ウェブアプリケーションはさまざまなソフトウェアコンポーネントを組み合わせて構成されており，即座に異常を絞り込むことは復旧にかかる時間の大きな短縮につながる． しかしその一方でコンポーネントごとに処理時間を監視する機構を導入するために，システムへの大きな変更を要する．

そこで，本研究では検出と原因究明の精度が高いだけでなく，対象システムへの導入が容易な手法を提案する．具体的には，処理時間の監視単位をリクエストの種類ごとにやや粗くすることで導入を容易にする．一般的なウェブサーバではリクエストごとに URL と処理時間を記録する機能が備わっており，対象システムへ変更を加えることなく，本手法を実現可能である．実際に Apache ウェブサーバでもこの機能を備えている．

一方で，比較的粗い監視情報から精度高く検出を行うために，管理図という既存の統計手法を応用する．ウェブアプリケーションの処理時間は一定ではなく，正常時でも揺らぎが生じる．この揺らぎと異常な変化の見極めを管理図の応用によって行う．管理図は過去のデータと現在のデータを統計的に比較し，分布が異なると判断した場合に警告を発する．過去のデータとして正常時のリクエスト処理時間を与え，運用時の処理時間と比較することで早期に性能異常を検出する．また，リクエストは種類ごとに異なるコンポーネントを用いて処理が行われるため，異常が発生しているリクエストの種類に注目することで，原因の絞込みを行うことができる．

さらに，本研究では複数の性能異常が同時に発生した場合にも対応する．異常を同時に複数検出した場合，それらが同じ原因により発生したものかどうかを判定することが求められる．同じ原因によるものだと判断できれば，そのリクエストの種類が共通して利用しているコンポーネントが原因である可能性が高い．逆に異なる原因によるものだと判断できれば，個々のリクエストの種類が個別に利用しているコンポーネントが原因の可能性が高い．

提案手法では，処理時間の変化傾向に着目することで，性能異常の原因が同じか異なるかを自動的に判定する．まず処理時間の分布の変化傾向を棒グラフの形式で抽出し性能異常のシグネチャとする．そして，シグネチャ同士を比較し類似度を計算する．最後に，この類似度を基に階層的クラスタリングの群平均法を用いてクラスタリングを行う．その後クラスタリング結果を参考にしながら管理者が原因究明を行う．リクエストの種類が同じクラスタに分類されれば，それらに発生している性能異常は同じ原因である可能性が高く，異なるクラスタに分類されれば，それらに発生している性能異常の原因は異なる可能性が高い．

提案手法を，オークションサイトを模したベンチマークウェブアプリケーション RUBiS に適用したところ，性能異常の早期検出と原因究明支援を行うことができた．クライアントの増加に伴い発生した性能異常を処理時間が約 100 ミリ秒増大した段階で検出できた．さらに，提案手法により得られた情報を利用し，異常の原因の絞込みをサーバ単位や，Java のクラス，メソッド単位で行うことができた．

Research for Detection and Diagnosis of Performance Anomalies in Web Applications

Satoshi Iwata

Abstract

Today, responsive services such as stock trading and online banking are becoming to be provided via web applications. However, they are not dependable enough and so, performance anomalies, such as increase of response time or decrease of server throughput, are often reported. It is required to recover a faulty system as quick as possible with early detection and diagnosis of anomalies, to prevent the service from getting severe damage.

Although several methods have been proposed to achieve the goal, none of them have fulfilled both of two major requirements. Not only should detection and diagnosis be done effectively, a method should be deployed easily to a target system. If not so, practical use of a method is hindered.

We propose a method which fulfills both requirements. Our method observes processing times with the granularity of request types, which can be differentiated via URL. As modern server software usually has a function to log URL and processing time of each request, our method can be applied easily to existing systems. For example, Apache web server has the functionality. Moreover, our method is effective to determine root causes, since we can narrow down root causes to components which have been used to process faulty request types.

To effectively detect and diagnose anomalies with coarse-grained monitoring, we take an advantage of statistical analysis. We apply *control charts* and monitor processing times. Control charts enable us to detect anomalies without being confused by natural fluctuations in processing time. Control charts compare current data to data which has been gathered during past normal operation. If control charts detect statistical differences between two distributions of data, they raise a warning. In our method, control charts monitor four types of statistics, i.e., average, maximum, median, and minimum of requests' processing time.

When several request types are simultaneously detected as faulty, we have to determine if their root causes are the same or not. Our method automates this task by clustering them based on the similarity in deviations from the non-anomalous distribution of measurements. Our clustering method involves three steps. First, we distill a *performance anomaly signature* from the processing time of requests. A performance anomaly signature characterizes how the "distribution" of the processing time has changed after an anomaly has occurred. Second, we calculate the *similarity* in the signatures. The similarity is a scalar that represents the degree to which two signatures, i.e., two bar graphs, overlap each other. Finally, we cluster anomalies based on the similarities. If two or more anomalies are clustered together, this implies that they are affected by the same root cause.

The results from case studies, which were conducted using an auction prototype *RUBiS*, are encouraging. The increases of processing times were around 100 milliseconds when our method detected anomalies. Afterwards, guided by the results of our clustering method, we determined suspicious components, such as server software, Java classes, or methods in a Java class.