Vol.2013-MBL-67 No.10
Vol.2013-CDS-8 No.10
2013/9/13

AWS を用いたスケーラブルな大規模安否情報システムの開発

永田正樹[†1†2]　金原一聖[†2]　竹村美彩都[†2]　峰野博史[†3]

概要：筆者らは小規模から大規模まで災害規模・災害者数に係わらず安定稼動する web 安否情報システムを提案してきた．本論文では，システム稼動基盤として AWS(Amazon Web Services) を用い災害規模に依存しない web 大規模安否情報システムの実装を行う．安否情報システムへのアクセス数・負荷は，災害規模や災害該当ユーザ数により変化する．しかし，発生する災害規模の事前想定が困難であるため，安否情報処理に係る受け皿となるシステムリソースの適切な事前見積りが困難である．予め固定されたリソースではそのキャパシティを超えるアクセスを被った場合，応答性の低下など不安定な状況を招きリクエスト処理が満足に行えず安否情報処理が完遂できない危険性がある．つまり災害規模の事前予測が困難である以上，どのようなリソースサイズであってもリソース固定は根本的解決には至らない．このため大規模安否情報システムの要件として，状況に応じて変化・拡充し災害状況如何を問わず持続的サービスを可能とする必要がある．そこで本研究では AWS のスケーラブル基盤を利用し災害状況によりシステム基盤の自動拡張／収縮可能な安否情報システムを構築する．本実装を行うことで平常時のコスト低減化及び災害時のシステムの安定稼動性を飛躍的に向上させることが可能である．

キーワード：安否情報システム，AWS，スケーラブル，負荷分散

The Development of the scalable large-scale Safety Information System using AWS.

MASAKI NAGATA[†1†2]　ISSEI KINPARA[†2]　MISATO TAKEMURA[†2]
HIROSHI MINENO[†3]

Abstract: We have proposed the Safety Information System which carries out stable operation irrespective of a disaster scale and the number of disaster persons from a small scale to large-scale. In this paper, implementation of the Safety Information System which does not ask a disaster scale is performed using AWS(Amazon Web Services). Access and Load to a Safety Information System is changed by a disaster scale and the numbers of applicable users. However, since the system cannot predict a disaster scale, the beforehand preparations of the system resource for Safety Information collection processing are difficult. In a fixed resource, when access exceeding the capacity is received, it cannot process by causing the fall of a response, and there is a danger that Safety Information processing cannot be completed. That is, since prediction of a disaster scale is difficult, in a fixed resource, it is unsolvable. The requirements for a large-scale Safety Information System are that a system resource expands and contracts according to a situation, and enables continuous service. In this study, the system built which enables automatic extension / reduction of a system resource according to a disaster situation using AWS. This system realizes cost reduction at the time of non-disaster, and improves the system stability at the time of a disaster.

Keywords: SafetyInformationSystem, AWS, Scalable, LoadBalancing

1. はじめに

災害対策のアプローチとして「災害に対する抵抗力向上」，「災害に対する回復力向上」がある[1]．前者は建物の耐震性を物理的に向上する方策で，後者は災害発生後の復旧策を迅速かつ効果的に講じる方策である．安否情報システムは後者に相当するものであり，人命安否の迅速な把握，事業継続可否の判断など，災害対応能力・機構の確立及び向上は近年における最重要課題の１つになっている[2][3]. 安否情報システムは，災害時においてシステムに予め登録された組織人員の安否登録・確認等の安否情報を効率的に収集し，集計・閲覧することを目的とし，災害発生後，次策を講じるための情報収集の仕組みとして有益である[4]．昨今ではインターネット上での web アプリケーションでの実装が主流となっており，教育機関，企業体，自治体など，各組織に浸透している状況である[5]．筆者らが所属する組織においても開発・製品運用が行われており，主に企業向けに展開している．

筆者らが開発した安否情報システムの基本構造は，インターネット上のパブリッククラウドサーバーを用い，web サーバー，DB サーバーの 2 台構成のごく一般的な web システムである．システムは PC やスマートフォン等のクライアント端末より安否情報登録・閲覧等のリクエストを受け，web サーバーが DB サーバーに登録されているクライ

†1 静岡大学 創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University
†2 株式会社アバンセシステム
AvanceSystem Corporation
†3 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University

アント情報を，CGI・PHP 等を経由し取得し，安否情報を html としてクライアント端末へ返すものである．シンプルな構成のため，初期開発コストの軽減化，メンテナンス性の高さや管理運用面が軽易であるなど，それなりのメリットもある．しかしこの構成ではサーバー数や CPU，メモリなどのシステムリソースを予め見積もった値に固定することになり，キャパシティ以上のアクセスを被った場合，システム稼動が不安定になる危険性がある．

システムリソース固定の問題点を解決するため，余裕を持ったリソースを状況如何に関わらず提供可能な仕組みが渇望される．「余裕を持ったシステムリソース」の捉え方について，キャパシティ管理として最も安直な施策は予め大容量なサーバーリソースを用意・準備しておくことである．しかし以下の問題がある．1 つは度外視された費用コストであり，1 つはアクセス負荷の事前見積りは災害という特異な状況下において非常に困難であり，どの程度が「余裕を持ったシステムリソース」かの判断が困難であるという点である．またユーザからの安否情報リクエストは災害発生後，時間経過や災害状況により変化するためシステムも変化・拡充し状況如何を問わず持続的サービスを可能とする必要がある．

本論文では，平常時または災害時の状況及び組織規模に依らず安定稼動し，災害時のシステム高負荷状態においても持続的サービス可能な安否情報システム実装について述べる．システム稼働基盤は AWS[6]を用いる．AWS はスケーラブルにリソースを拡張・収縮可能であり，状況により消費リソースが著しく変化する安否情報システムの特性に適している．2 章では開発システムのベースとなる静岡大学での安否情報システムについて述べ，3 章で現状システム仕様とその課題を提起し，4 章で開発システムの概要を述べ，5 章でまとめと今後の課題について述べる．

2. 関連研究

2.1 静岡大学での安否情報システム

東海大地震の危険地域に位置する静岡大学では，2009 年 5 月より全学に安否情報システムを導入し，約 4 年の運用を経ている[7][8]. 静岡大学の安否情報システムの仕組みは，災害時に高い可用性を担保しつつ多くの安否情報回収を実現するためのシンプルな構成である．主な構成内容は，システムが稼働するサーバーのクラウドコンピューティングによる遠隔地設置, 認証コード付き URL をユーザ毎に送信し URL をクリックするだけでアカウント ID や PW を入力せずにシステムへログイン可能な簡易自動認証，web サイト上でボタンクリックのみによる簡易かつ迅速な安否情報登録，安否登録指示メールの 1 秒間隔での時間差配信，気象庁 RSS からの自動地震情報取得などがある．簡易自動認証・簡易安否情報登録は災害時の混乱において，アカウント ID・PW 入力の煩雑さ排除や，ID・PW を失念しているユーザも安否登録が可能となり，災害時の安否情報収集率向上に効果的である．

筆者らが開発した安否情報システムは，静岡大学の安否情報システムの基礎技術を流用[a]しており，構成内容は大規模組織向け安否情報システムにおいても有益なため基本構成は踏襲している．しかし大学などの教育機関組織において組織人員数は年度毎の入れ替えはあるものの，総数は年を重ねても大きくは変化しない．つまり総数に応じたシステムリソースを予め見積もることが可能である．一方，筆者らが提案する大規模安否情報システムは，一定組織のみでなく様々な組織に対して随時ユーザ登録を実現するため，その総数は変化していく．つまりシステム稼働時に一定のシステムリソースで固定されたキャパシティでの運用は，随時増加するユーザによりあるタイミングで飽和する危険性がある．

3. 現状システムとその課題

3.1 現状の安否情報システム基本構造

安否情報システムとは，災害時において予めシステムに登録しているユーザの安否情報を収集・公開・閲覧する web サービスである．一般的仕組みは，災害発生後システムから予め登録済みのユーザへメール・掲示板 URL などを配信し，ユーザはそれに対してリアクションとして安否情報登録・閲覧を行うものである（図 1）.

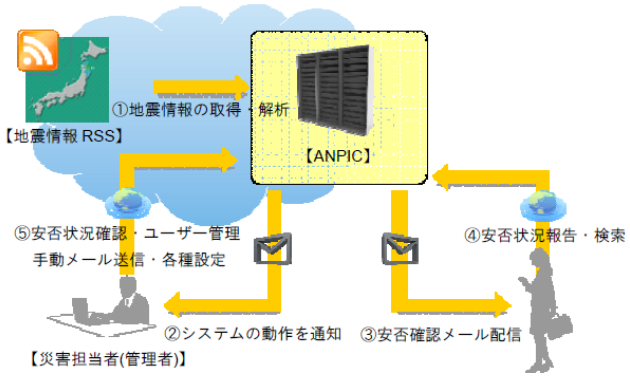


図 1 安否情報システム概要
Fig.1 Outline of the Safety Information System

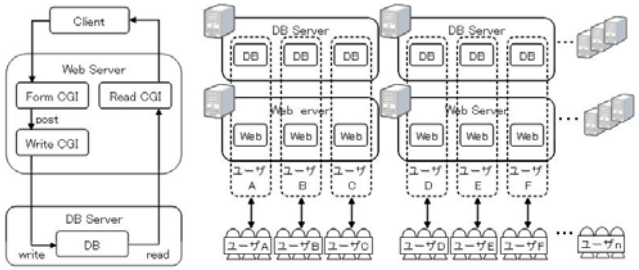
筆者らが開発した安否情報システムは上述の仕組みに対して，AWS のパブリッククラウドサーバーAmazon EC2 に Linux CentOS 6.4[9]，web サーバーに Apache 2.2.15[10]，DB は PostgreSQL 8.4.12[11]を用いて実装している．システムは PC やスマートフォン等のクライアント端末より安否情報登録・閲覧等のリクエストを受け，web サーバーが DB サーバーに登録されているクライアント情報を，PHP・CGI 等を経由し取得し，安否情報を html としてクライアント端末へ返すものである（図 2（a））.

1 システムについて web サーバー1 台，DB サーバー1 台

a ソフトウェア使用許諾契約（国立大学法人 静岡大学，株式会社アバンセシステム）

Vol.2013-MBL-67 No.10
Vol.2013-CDS-8 No.10
2013/9/13

で構成し，1 システムに収容可能なユーザ数を固定しているため，ユーザ数が増加すれば上述の構成をマニュアルで構築し追加していく人的手間が発生し，さらにキャパシティ固定の課題がある（図 2（b））．



(a)処理フロー　　(b)サーバー構成

(a) Processing flow　　(b)Server composition

図 2 現状システム構成

Fig.2 Present System configuration

3.2 平常時，災害時のキャパシティ管理

安否情報システムが高負荷を被る状況は，主に災害時のユーザからのアクセス集中時であり，平常時でのシステム負荷は低い．原因は災害時には安否登録・閲覧などシステムへアクセスするユーザが増加しシステムへの関心が高まるが，平常時はそもそも被害を被っていないため安否情報を登録・閲覧する必要がなくシステムへの関心が低いためである．つまり災害時と平常時ではシステム負荷の差が激しい．また，災害規模の事前予測は困難であるため災害に対して最適なシステムリソースの事前見積りも同様に困難である．

以上のように災害規模やその災害対象となるユーザ数によりシステムへのアクセス数及び負荷状況に差がある安否情報システムの特性上，予め一定量のサーバーリソースをシステムのキャパシティ管理として画一的に施しておくことは得策ではない．

3.3 主要機能のシステムに与える負荷

現状システムの構成にてユーザ数 2 万名規模での安否確認訓練を実施したところ，複数ユーザの同時アクセスによりサーバー応答が不能になる事態が発生した．対応手段として，一旦システムを停止し AWS 機能によるサーバースペックのスケールアップを実施した．応答不能状態は復旧したが，スケールアップに伴うシステムの再稼働は一時的にサービスを停止する問題がある．またサーバーのキャパシティは限りがあるため，1 台のサーバーでの限界値を超えるユーザリクエスト対応は不可能である．この課題を解決するため単体サーバーのスケールアップではなく，サービスを継続しつつサーバー台数を増減することによりキャパシティを増減可能なスケールアウト・イン構成にてシステムを構築する．まず 1 台で処理可能な負荷を算出しその値を基にスケールアウト・インを実行するため現状システムの評価を行った．

本システムの主要機能である「安否情報登録」，「安否情報閲覧」の 2 つの機能にてシステムに与える負荷を計測した．（表 1）に評価に用いたサーバー環境を示す．テストツールに Apache JMeter[12]を使用した．評価内容は，クライアント PC の JMeter 上にて複数の擬似ユーザから本システムへ同時にアクセスし「安否情報登録」，「安否情報閲覧」の各機能をそれぞれ行い，本システムが稼動するサーバー上にて Linux コマンドの「top[13]」，「sar[14]」，「vmstat[15]」等を使用しシステム負荷を計測した．徐々に同時接続ユーザ数を増加したところ 200 名の擬似ユーザ（スレッド）からのアクセスにて web サーバーのロードアベレージが 1.0 以上となりシステムの高負荷状態が認められた．CPU 負荷，メモリ負荷，Disk I/O，ネットワーク I/O 等の調査の結果，この時点でのボトルネック主要因は CPU 負荷であったため CPU 負荷について詳細計測を行った．

インスタンスタイプ	t1.micro
OS	CentOS 6.4
CPU	1.0-1.2 GHz
コア	1コア
RAM	615MB
HDD	16GB

表 1 評価環境

Table 1 Evaluation environment

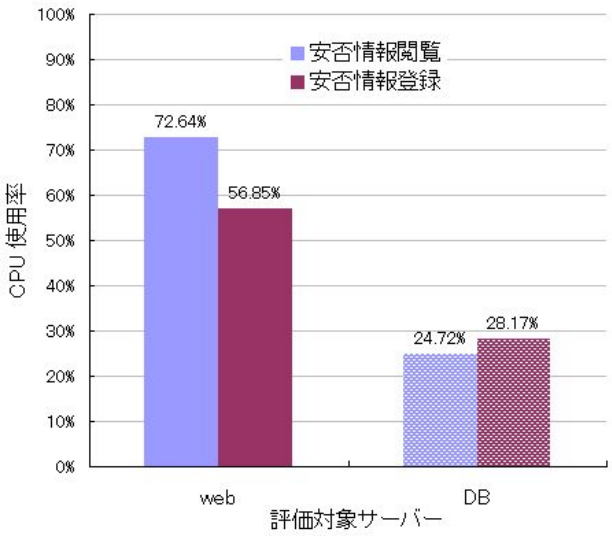


図 3 主要機能の CPU 負荷

Fig.3 CPU usage of a prime function

評価結果の値は上述した 2 つの機能を実行中の CPU 使用率の平均値である（図 3）．「安否情報閲覧」実行中，web サーバーの CPU 使用率が 100%超を継続する期間もあり高負荷状態が認められたが，DB サーバーは最大でも 40%程度となり負荷は低い．

現状システムでは web サーバーが DB サーバーより負荷が高まる傾向にあり，システム全体としてみれば web サーバーがボトルネックになっている．

3.4 2 つのユーザ特性によるアクセスの傾向

安否情報システムへのアクセスは上述の通り主に災害

時に集中するが，以下の通りユーザ特性によりその傾向に差が生じる．

(1) **受動的ユーザ（一般ユーザ）**

災害時，システムからのメール受信後それをきっかけとして受動的にシステムへアクセスし自身の安否情報を登録するユーザ．操作権限は安否登録・自組織内のみの安否情報閲覧などの機能に制限される．組織内において安否情報管理をされる側である．

(2) **能動的ユーザ（管理ユーザ）**

システムからユーザに安否情報収集メールが届いていない状態においても能動的にシステムへアクセスし，自身の安否情報登録や組織人員の安否情報を閲覧・管理するユーザ．システムへのアクセスは災害時だけでなく，平常時にユーザ登録・組織管理・安否集計なども行う．管理ユーザは複数人登録可能であり，組織において安否情報を管理する側である．

受動的ユーザと能動的ユーザのアクセス数を比較すると，能動的ユーザは安否情報を管理する管理者である場合が多く，主に安否情報を登録するのみの受動的ユーザより数が少ない．つまりシステムへのアクセス集中は災害時の受動的ユーザからのアクセスが圧倒的に多いため，災害時の受動的ユーザへの対応が本システムの負荷対策の主題となる．

3.5 現状の負荷対策と課題

システム高負荷の原因となるアクセス集中は 2 つのユーザ行動パターンから推測可能である．それぞれの行動パターンに対して以下の対策を実施している．また課題を提起する．

(1) **受動的ユーザアクセス負荷への対策**

システムへアクセスが集中し高負荷となる原因は複数ユーザが同時に一斉アクセスするためである．そのため同時アクセス数を低減する仕組みを講じることでアクセスを分散する．受動的ユーザはシステムからメール受信後にアクションを起こすため，メール配信に対策を講じることで負荷分散を実現する．対策内容は，個々のユーザへのメール配信を 1 秒間隔としユーザへのメール受信の時刻を分散させ，システムへのアクセス時刻も同様に分散させることでアクセス集中を軽減する．メール配信が一斉に行われた場合，ユーザはほぼ同タイミングにメールを受信する．経由するメールサーバー，DNS サーバーなどの関係上全く同タイミングではないが，配信元からの送信時刻が全ユーザで同一の場合とユーザ毎にシステムにて明示的に時間差をつけて送信した場合とでは，ユーザへの受信到達時刻は後者が到達時刻の差が大きい．本対策はメール受信時間差を利用する．

しかし以下にあげる課題がある．1 秒間間隔でのメール送信は 10000 名組織の場合，最終ユーザへ到達する時刻は 10000 秒≒3 時間となる．メール送信中の 3 時間の間に災害により通信インフラが遮断された場合，メール不達ユーザが発生する．組織人員全員にメールが行き届かなければその組織の正確な安否情報収集に影響を及ぼす可能性がある．

(2) **能動的ユーザアクセス負荷への対策**

システム負荷はアクセス数だけでなくシステムを操作する処理にも関連する．プログラム的に重い処理を実行すればシステム負荷が増加する．能動的ユーザ（管理ユーザ）の操作内容は，ユーザ登録・組織管理・安否集計など多岐に渡り，受動的ユーザの主操作である安否情報登録・閲覧と比較して，個々の処理がプログラム的に複雑であり重い．また能動的ユーザ（管理ユーザ）の操作内容の特性上，比較的平常時のアクセスが多いため，平常時に対しても負荷対策を講じる必要がある．

対策内容は，事前に能動的ユーザ（管理ユーザ）が行う個々の操作に対して消費リソースを算出・把握し，操作内容に応じてシステムキャパシティを決定する．現状システムでは，安否情報システムを導入する組織人員数・管理者ユーザ数の事前把握が可能であり，またシステムへの登録人数を制限している．さらにユーザ数に応じてシステム構築を行っているため，少なくとも平常時の能動的ユーザのアクセス集中による負荷は上記から算出可能である．具体的施策としてシステムへの登録ユーザ数に応じて AWS EC2 のスケールアップで対応している．

しかし能動的ユーザの平常時での事前リソース見積りは不可能ではないが，安定稼動を担保するためには登録されている全能動的ユーザ（管理ユーザ）数に対して余裕を持ったキャパシティを事前確保しておかなければならない．全能動的ユーザ（管理ユーザ）が常時システムへアクセスしている訳ではないため，通常，アクセスが少ない状況では過剰キャパシティとなりコスト面での課題がある．またキャパシティの事前固定は本質的な解決にはならず，状況に応じてシステムをスケーラブルに拡大・縮小可能な大規模安否情報システムを実現するためには大きな課題となる．

3.6 課題と要求事項の整理

ユーザ数や災害規模に依らず安定稼働する大規模安否情報システムを開発するにあたり，課題から要求事項の整理を行う．まずアクセス集中及び高負荷状況は下記となる．

- 平常時，主に能動的ユーザのプログラム処理が重いシステム設定操作．
- 災害時，主に受動的ユーザの安否登録・閲覧などのアクセス集中．
- 災害時の上記 2 ユーザによる混在アクセス．

上記から，平常時の高負荷処理と災害時の規模・該当ユーザからのアクセス集中に応じて最適なリソースを自動拡張・収縮を可能とするスケーラブルな処理基盤が必要である．またシステムへ登録するユーザ数を固定せず事後追加が可能であることも必要である．

Vol.2013-MBL-67 No.10
Vol.2013-CDS-8 No.10
2013/9/13

4. AWS を用いた大規模安否システム

平常時・災害時のユーザ特性，登録ユーザ数などの状況如何に依らず持続的サービスを可能とし，また当該状況において適切なリソースをシステム自らスケーラブルに自動拡張・収縮する仕組みの処理概要を述べる[16][17].

4.1 AWS での自動拡張・縮小の概要

本システムが稼動する基盤として AWS を用いる．AWS は Amazon.com の保有するコンピュータリソースを，インターネットを通じてユーザに提供する遠隔コンピューティングサービスである．AWS を利用すると投資リスクを抑えつつスケーラブルなシステム構築が可能となる．

本システムでの平常時と災害時のシステム負荷差は災害時が高いため，システムのキャパシティ管理としては，平常時のリソースは少なく見積もり，災害時にはリソース増強を行う，というアプローチが効果的である[18]．AWS を利用し平常時では最低限のリソースで稼動し，災害時にはその規模に応じたリソースを自動拡張・収縮する仕組みを検討する．処理概要は以下となる．

1． Cloud Watch[19]を用いシステム負荷（CPU，メモリ，ディスク I/O，ネットワーク I/O）の監視及びリソース値に閾値を設定する．
2． 閾値以上の負荷を被った場合，閾値をトリガーにして Cloud Watch はリソース拡張･収縮を Auto Scaling[20]に指示する．
3． Auto Scaling は Cloud Watch からの指示により予めシステムに定義された拡張・縮小ポリシーに沿って web サーバー群（予め web サーバーのマシンイメージを EC2 AMI として作成しておく）を起動する．
4． 起動された web サーバー群を Elastic Load Balancing[21]がシステムのロードバランシンググループに設定し，クライアントからのアクセス負荷分散を実施する．

以上が AWS 基盤での自動拡張・収縮の基本構造である（図 4）．[22].

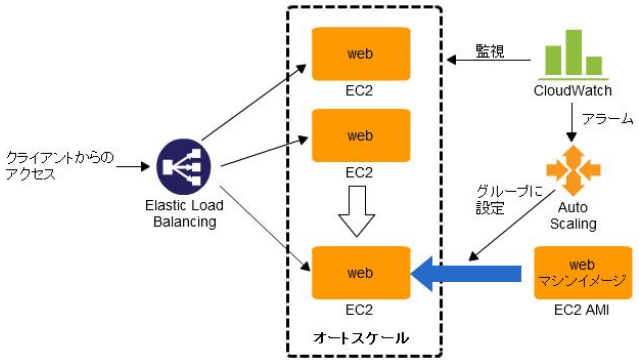


図 4 AWS 自動拡張・縮小概要
Fig.4 The outline of automatic extension / reduction

4.2 システム構成

本項以降にて開発システムの構成を解説する（図 5）.

本システムは可用性を考慮し平常時においても web サーバー2 台の冗長化構成を敷く．それぞれの web サーバーが稼働するデータセンター（アベイラビリティゾーン）を別とすることで一方のデータセンター施設に問題が生じた場合においてもシステム継続稼働を可能とする．複数 web サーバーへのセッション維持は，セッション情報を保持するステート管理サーバーを別途稼働する．それぞれの web サーバーはステート管理サーバーにセッション情報を保存・取得しステート管理を行う．web システムの高い応答性を考慮するとセッション情報の書き込み・読み込みも高速性が求められるため，ステート管理サーバーにはオンメモリデータベースとして ElastiCache[23]（memcached）を採用する．

DB サーバーは RDB を用いサーバースペックは 1 台で許容可能なユーザ数から決定する．予め複製起動のために AWS 上でマシンイメージ（AMI）を作成しておき，キャパシティを超えるユーザ登録が発生した場合，同一構成の DB サーバーを AMI から複製し起動する．安否情報システムの特性上，登録ユーザ数の増減は頻繁に発生することはなく，災害派生有無にも関連しないため（ユーザ登録・削除は平常時に管理者が行う），DB サーバーは web サーバーのような瞬発力のあるスケーリング構成は不要となる．そのため DB サーバーのスケールは平常時を想定する．既存の DB サーバーと新たに起動された DB サーバーへのクエリの振り分けはシャーディングモジュールが担当する．また DB サーバーに納められたユーザ情報は AWS SnapShot でバックアップを取る．

4.2.1 負荷予測機能

開発システムのリソース拡張・収縮はサーバー台数の増減により実現した．スケーラブルにリソース拡張・収縮が可能である AWS は，平常時・災害時で著しくシステム負荷が変化する安否情報システムの稼動特性に適している．しかし以下の課題もある．AWS での拡張・収縮は，システムリソース（CPU 使用率，Disk I/O など）の監視及び閾値アラームに依るものであるため，システムが高負荷状態となった後に実行される．高負荷状態でのユーザリクエスト処理は応答不能状態も懸念されるため，システムが負荷を被る前にリソース拡張を実現することが望ましい．

本システムでは観測時点での安否情報システムの状態によりその後の負荷状況を予測し，事前スケール可能なモジュールを開発した．コントロールサーバー（図 5）が安否情報システムのステータスを監視し状況により事前スケーモジュールがこれを行う．

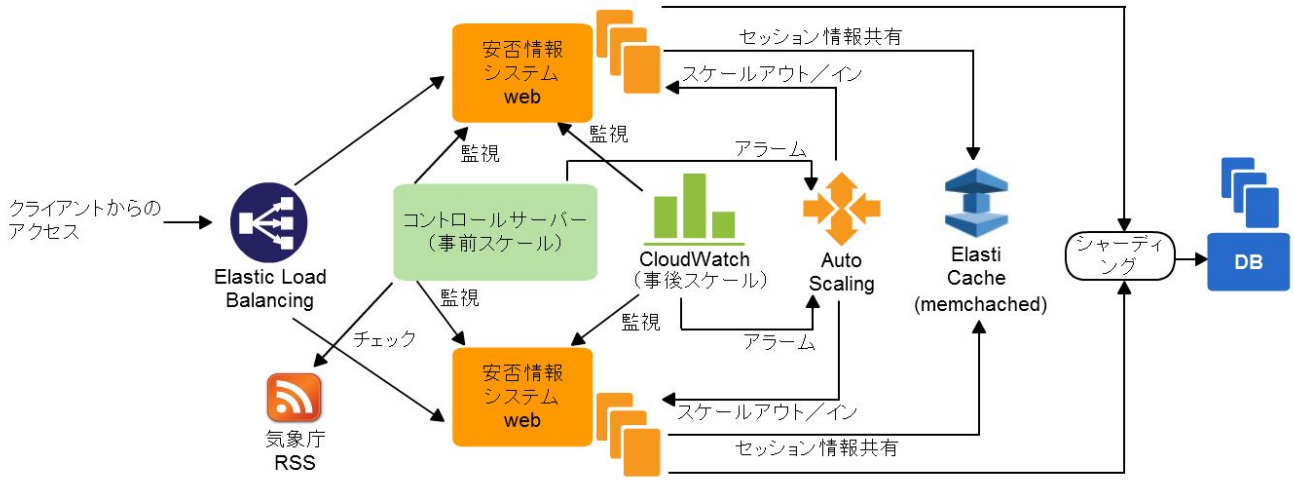


図 5 開発システムアーキテクチャ

Fig.5 Architecture of development system

4.2.2 負荷予測のケース

本システムは以下の状況をステータス値として内部に保持し，コントロールサーバーがステータス値を監視することでその後の負荷予測を実現する．以下に挙げる値は直近でのシステムへアクセスする分母（ユーザ数）となるため，この値が大きければその後の高負荷状況が予測可能となる．

- システムに与える負荷が大きい能動的ユーザ（管理ユーザ）の総ログイン数
- 一定間隔で監視している気象庁 RSS の内容に応じた災害該当ユーザ数
- メール送信モジュールの処理待ちキュー値（ケースとしては地震以外での大量手動メール送信後のアクセス集中がある）

以上の値からシステムが高負荷状況に陥った後での後追いリソース拡張ではなく，事前に拡張することによりその後集中するアクセスを許容可能とする基盤の事前構築が可能となる．

課題として挙げた時間差メール送信においても，メール送信前に予めリソース拡張を施しておくことで一斉メール配信による同時アクセスを許容可能となり，時間差メールによるメール不達問題を解決し安否情報収集速度・収集率及びその正確性がより向上する．

4.2.3 スケールポリシー（負荷集中前後のスケール）

リソース拡張・縮小の程度は平常時・災害時などの状況により変化させなければならない．どのような状況でどのようなタイミングにどの程度リソースを拡張・縮小するのかポリシー（スケールポリシー）を綿密に定義しておく必要がある．安否情報システムは平常時の負荷は低いためスケールポリシーの方針としては，平常時は最低限のリソースで，災害時は規模・対象ユーザ数に応じたリソース拡張を行う．

先に述べた負荷予測による事前スケールだけでなく，システムへのアクセスは災害発生後，時間経過や災害状況により増加する可能性もあるため事後拡張処理（CloudWatch での負荷監視）も実装する．

システムのスケールポリシーは事前・事後両者を定義し様々な状況で対応可能な構成とする．また高負荷状況に陥る前のリソース拡張は敏速に，負荷軽減時の収縮は緩慢に行いシステムキャパシティに余裕を持たせる（図 6）．

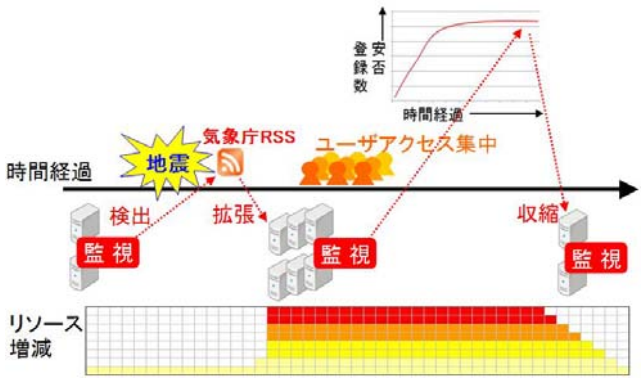


図 6 リソース拡張・収縮

Fig.6 Resource extension / reduction

4.2.4 スケーリングフロー

平常時のデフォルト構成は，ElasticLoadBalancing でのロードバランシンググループ配下に 2 台の web サーバーが稼働している．スケールするタイミングは，コントロールサーバー（事前スケール）及び CloudWatch（事後スケール）が，監視している 2 台の web サーバーの負荷が設定された閾値に達した場合，閾値をトリガーとして AMI に複製済みの web サーバーを起動する．この時 2 拠点のデータセンターにて 1 台ずつ合計 2 台の web サーバーを起動する．理由は ElasticLoadBalancing がトラフィックを公平に配分するためである．スケールした状態においてさらに負荷閾値に達した場合，再度上述のフローを行い，2 台，4 台，6 台と

web サーバーを増加していく．減少は逆の手順となる．

4.3 web 分散構成での検証

スケーリング後を想定し web サーバー4 台での分散環境にて負荷評価を行った．評価環境・内容は「3.3 項 主要機能のシステムに与える負荷」に準ずる．

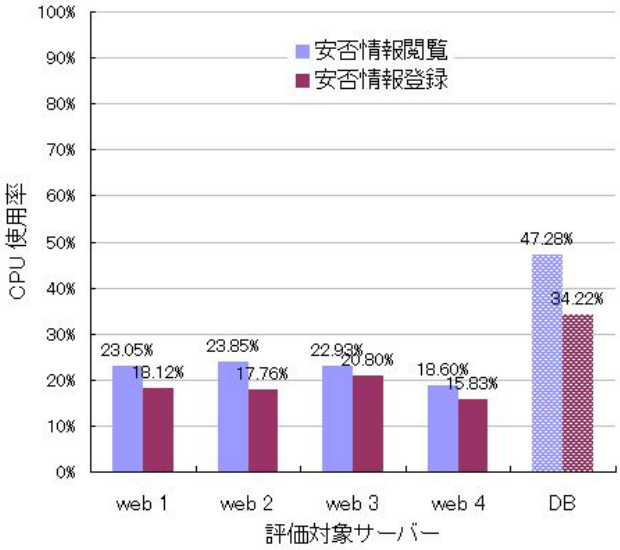


図 7 web サーバー4 台での CPU 負荷

Fig.7 CPU usage of 4 Web Servers

「3.3 項主要機能のシステムに与える負荷」の web サーバー1 台構成では「安否情報閲覧」に平均 70%を超える CPU 負荷を被ったが 4 台構成ではアクセスが分散され，各サーバーに係る負荷もほぼ均等に分散されている（図 7）．また web サーバーの負荷が下がりタスク待ちが減少しリソースに余裕が発生したため，その分 web サーバーから DB サーバーへの問い合わせが増加している．web サーバー1 台構成と比較し DB サーバーの負荷がキャパシティの範囲内で上昇していることは，DB サーバーのリソース活用においても有効である．

5. まとめと今後の展望

本論文では，AWS を用い特定の状況だけでなく災害前後に想定される様々な状況において，災害内容・規模に応じてシステムリソースを自動拡張・収縮する大規模安否情報システムを開発した．本システムは安否情報を管理・操作する web システムとそれを稼動するスケーラブルな拡張・収縮基盤で構成される．本システムの効果は，災害において拡張・収縮するアーキテクチャとしての側面と，状況に即したシステムキャパシティを調整することにより平常時では低スペック・低コストのシステム運用が可能なコスト面の側面とがある．

しかし以下の課題・懸念事項もある．

- コントロールサーバー，シャーディングモジュールステート管理サーバーなどの単一障害点
- RDB の高度かつ複雑な機能がボトルネックになる恐れ

以上のような課題を解決するために各モジュールの冗長化やデータ特性の検証が必要である．

また災害時においては安否情報システムだけでなく全ての情報システムの負荷が上昇する恐れもあり他システムとのリソース競合等も注意しなければならない．クラウド基盤でのリソース配分は有限のため，システム分散箇所を複数地点設けるなどの検討も必要である．

今後は上記課題解決を含む本論文で開発したシステムの評価・検証を行う予定である．現状はアクセス集中や機能毎の負荷値把握について定量的評価が乏しいため，システム内に負荷値可視化の仕組みを実装し評価を行う．得られた負荷値に対してシステムが安定稼動するための最適なリソース値算出を検証し，実装・評価を行う．

謝辞 本システムの母体となった静岡大学の安否情報システム開発者 長谷川孝博准教授に深く感謝する．

参考文献

1) 林能成，梶田将司，太田芳博，若松進，木村玲欧，飛田潤，鈴木康弘，間瀬健二: 組織特性を考慮した大学向け災害時安否確認システムの開発，安全問題研究論文集 Vol.3（2008 年 11 月），土木学会
2) 松本直人: 事例に学ぶ東日本大震災における情報発信，情報処理学会論文誌，Vol.54， No.3，1021-1027(Mar.2013)
3) 佐藤聡，杉木章義，陳漢雄，古瀬 一隆，片岸 一起，中井 央，秋川 友宏，前田 敦司，和田 耕一: 東日本大震災時の筑波大学情報インフラにおける対応と課題，情報処理学会論文誌，Vol.54，No.3，1038-1049(Mar.2013)
4) 畑山満則，安藤恵: 地域防災計画における情報伝達の機能的障害の発見手法の開発，情報処理学会論文誌，Vol.54，No1，202-212(Jan.2013)
5) 太田芳博，梶田将司，林能成，若松進: 名古屋大学安否確認システムの構築と運用，電子情報通信学会技術研究報告，IA，108(409)，77-82(2009-01-21)
6) Amazon Web Services(online), from http://aws.amazon.com/ (accessed 2013-04-22)
7) 長谷川孝博，井上春樹，八巻直一: 低コスト運用でユーザフレンドリな安否情報システムの開発, 学術情報処理研究, 13, 91-98, 2009
8) 金子朋広，長谷川孝博，瀬野忠愛，八巻直一: クラウドを用いた大規模安否情報システムの構築と運用，スケジューリング・シンポジウム論文集，巻 2011，103-108，20110924
9) The CentOS Project: CentOS(online), from http://www.centos.org/ (accessed 2013-04-22)
10) The Apache Software Foundation: Apache HTTP SERVER PROJECT(online), from http://httpd.apache.org/ (accessed 2013-04-22)
11) The PostgreSQL Global Development Group: PostgreSQL(online), from http://www.postgresql.org/ (accessed 2013-04-22)
12) The Apache Software Foundation: Apache JMeter(online), from http://jmeter.apache.org/(2013-08-15)
13) top: Linux man pages(online), from http://linux.die.net/man/1/top(accessed 2013-08-15)
14) sar: Linux man pages(online), from http://linux.die.net/man/1/sar(accessed 2013-08-15)
15) vmstat: Linux man pages(online), from http://linux.die.net/man/8/vmstat(accessed 2013-08-15)

16) 松原正純，鈴木和宏，勝野昭: 自律コンピューティングに向けた HPC 向け動的負荷分散機構，情報処理学会論文誌，Vol.44，No.SIG11(ACS3)，Aug.2003
17) 松本亮介，川原将司，松岡輝夫: 大規模共有型 Web バーチャルホスティング基盤のセキュリティと運用技術の改善，情報処理学会論文誌，Vol.54，No.3，1077-1086(Mar.2013)
18) 江丸裕教，高井昌彰: 動的配置法によるハイブリッドクラウドの運用管理コスト最小化，情報処理学会論文誌，Vol.54，No.4，1581-1591(Apr.2013)
19) Cloud Watch, Amazon Web Services(online), from http://aws.amazon.com/cloudwatch/ (accessed 2013-04-22)
20) Auto Scaling, Amazon Web Services(online), from http://aws.amazon.com/autoscaling/ (accessed 2013-04-22)
21) Elastic Load Balancing, Amazon Web Services(online), from http://aws.amazon.com/elasticloadbalancing/ (accessed 2013-04-22)
22) Jurg van Vliet，Flavia Paganelli，Steven van Wel，Dara Dowd，玉川憲 監修, 玉川竜司 訳: Amazon Web Services プログラミング，OREILY，2012
23) ElastiCache, Amazon Web Services(online), from http://aws.amazon.com/elasticache/(accessed 2013-08-15)