災害管理アプリケーションプラットフォームの自律的な障害復旧を実現する資源管理システムの構築

松井 祐希[1,a)]　渡場 康弘[2]　伊達 進[2]　下條 真司[2]

キーワード：災害管理，アプリケーションプラットフォーム，資源管理システム

1. 研究背景

地震や台風といった自然災害が猛威を振るう昨今，災害管理と呼ばれる災害発生前後の一連の活動が重要性を増している．災害管理に参加する組織間の連携を支える災害管理アプリケーションを構築する上でのコンセプトがInformation-as-a-Service（InfaaS）である．InfaaS に則る上で災害管理アプリケーションが満たすべき要件は情報の明瞭性・同期性・継続性の 3 つである．この 3 つの要件はアプリケーションレベルでの対応のみならず IT インフラストラクチャレベルでの対応が必要であり，災害管理アプリケーションが個別に対応するのが困難である．われわれは，開発及び運用の基盤となる InfaaS 対応型アプリケーションプラットフォームを開発することで，その上で稼働する災害管理アプリケーションでの InfaaS の要件達成を図った．

InfaaS の要件のうち，明瞭性と同期性は分散可視化システムの仕組みを導入することで達成した．継続性を実現するために今日までにわれわれが研究開発を進めてきたのが Software-Defined IT Infrastructure（SDITI）である．SDITI は InfaaS 対応型アプリケーションプラットフォームにおける IT インフラストラクチャをソフトウェアから動的に制御する仕組みである．SDITI により，InfaaS 対応型アプリケーションプラットフォームでは今日までに構成要素単位での可用性の向上を実現してきたが，実用に即した継続性を実現するためには，構成要素同士が連携して自律的に障害から復旧する仕組みが必要である．

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[2] 大阪大学 サイバーメディアセンター
Cybermedia Center, Osaka University
[a)] matsui.yuki@ais.cmc.osaka-u.ac.jp

2. 提案手法

本研究では，InfaaS 対応型アプリケーションプラットフォームにおける自律的な障害からの復旧に向け，構成要素を一元的に制御する資源管理システムを提案した．

提案資源管理システムの構成を図 1 に示す．資源管理システムは，計算ノード・Docker・Software-Defined Networking（SDN）・ストレージそれぞれに対応した 4 つのマネージャとそれらを統括するマスターオーケストレータによる構成とした．マスターオーケストレータはインターフェースを介してトップダウン式に各マネージャを制御する．マスターオーケストレータに操作を集約することで，構成要素間の連携を実現して状況に応じた自律的な復旧を促す．

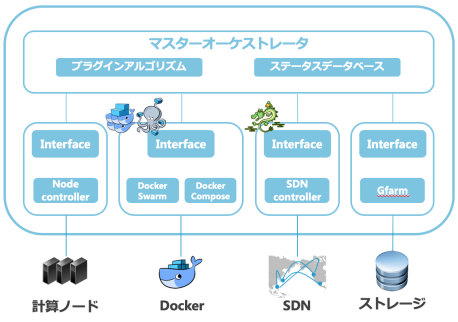

図 1　提案資源管理システムの構成．

資源管理システムは，物理ネットワーク基盤を管理下に置いて InfaaS 対応型アプリケーションプラットフォームとして運用する．資源管理システムは，物理ネットワーク基盤から任意の資源（計算ノード・ネットワーク経路）を仮想的に切り出してプレーンを構築する．プレーンでは，

© 2018 Information Processing Society of Japan

所属する計算ノードのうちいずれか 1 台に Docker コンテナを展開し，その上で可視化処理用のミドルウェアである SAGE2 のアプリケーションサーバを起動する．それぞれのプレーンには災害管理アプリケーションが配備され，プレーン内の資源を利用して運用される．

資源管理システムは，InfaaS 対応型アプリケーションプラットフォームの構成要素の状態を常に監視し，障害の発生に備える．構成要素の被災により発生した障害を検知した際には，資源管理システムは資源選択アルゴリズムに基づいて障害を受けた資源の代わりとなる代替資源を選択する．選択された代替資源をもとに，資源管理システムはネットワークの再構成や SAGE2 アプリケーションサーバの再配備を実施する．以上の動作を自動的に実施することで，資源管理システムによる InfaaS 対応型アプリケーションプラットフォームの自律的な復旧が実現する．

3. 評価

評価では，提案資源管理システムの性能評価として，障害復旧に要する時間を実施した．設計をもとに資源管理システムを実装し，InfaaS 対応型アプリケーションプラットフォームをローカルのテストベット環境に構築した．テストベッド環境上で SAGE2 アプリケーションサーバを起動してコンテンツ配信をさせておき，計算ノードに意図的に障害を発生させることで資源管理システムが代替の計算ノードを選出してサービスを復旧する様子を観察した．

テストベッド環境を図 2 に，初期状態として構築したプレーンを図 3 示す．テストベッド環境は 13 台の計算ノードと 6 台のスイッチ，それらを結ぶネットワークで構築した．13 台の計算ノードのうち 1 台（m）は管理用計算ノードであり，実装した資源管理システムを配備した．初期状態として構築したプレーンは 4 台の計算ノードとそれらを結ぶ最低限の経路から構築した．このプレーンで，Docker コンテナが配備されている計算ノード（n9）に意図的に障害を復旧させることで，プレーンが再構築されて SAGE2 アプリケーションサーバが再配備されるまでの時間を計測した．

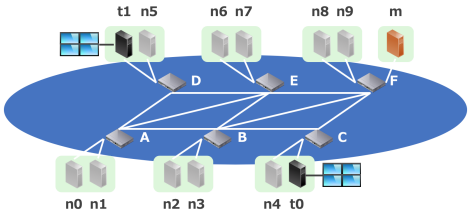

図 2　テストベット環境.

資源管理システムが障害検知アルゴリズムを呼び出す間隔は 3 秒とした．障害検知アルゴリズムは Docker Swarm のヘルスチェック機能の結果を参照するアルゴリズムを，

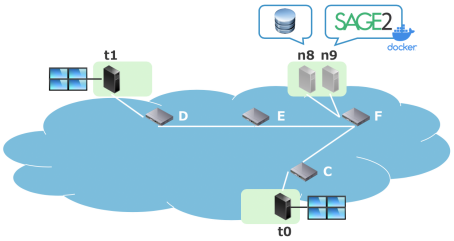

図 3　構築したプレーン.

表 1　資源管理システムの障害発生から復旧までの所要時間

動作	時間 (s)
障害検知	12.2
ステータスデータベース参照	0.108
資源選択アルゴリズムの実行	1.81×10^{-4}
通信経路のプレーンへの追加	0.311
TDW 用計算ノードとの接続	6.01
Swarm クラスタへの追加	3.50
Docker コンテナの操作	6.68
合計	28.8

資源選択アルゴリズムはプレーンに割り当てられていない計算ノードの中から無作為に 1 台の計算ノードを選ぶアルゴリズムをそれぞれ採用した．プレーンに割り当てる経路は，ホップ数が最も少ない経路を選択するものとした．試行は 10 回実施し，平均値を採用した．

結果は表 1 に示す通りとなった．時間的にもっとも高い割合を示したのは障害検知であるが，これは Docker Swarm のヘルスチェック機能が障害を検出するまでの時間に依存したものであったため，別の障害検出アルゴリズムを構築することで時間の削減が見込める．一方，資源選択に要した時間は他の処理に比べて小さく抑えられていたが，今回使用したランダムな計算ノードの選択が運用上望ましい計算ノードを選出するとは限らない．展開後のサービスの運用を考慮した資源選択アルゴリズムの考案が重要であると同時に，その際に発生すると考えられる資源選択の実行時間の増加は許容する必要があると考える．

4. 総括

本研究では，InfaaS 対応型アプリケーションプラットフォームにおける自律的な障害からの復旧を目指し，構成要素を一元的に制御する資源管理システムを提案した．評価では，性能評価として障害復旧に要する時間を計測した．

将来展望として，国際的な SDN テストベッド環境を利用したより実践的な性能評価を検討している．また，実用に即した障害検知ならびに資源選択アルゴリズムの検討が必要である．

謝辞

本研究の一部は JSPS 科研費 JP16H02802，JP17K00101 の助成を受けたものです．

© 2018 Information Processing Society of Japan