サステーナブルコンピューティング・シミュレータ

池嶋 俊* 小磯 知之† 鈴木 与範‡
阿部 洋丈§ 加藤 和彦‡、§

1. はじめに

近年、インターネットを介して提供されるサービスの重要性はますます高まってきている。しかし、障害によってサービスが提供不能に陥る事態もしばしば発生し、様々な損害をもたらしている。特に、通信障害が原因のサービス不能は、その影響が大きいにもかかわらず対処が困難である。対処を困難にしている一因は、その原因の多様性にあると考えられる。インターネットにおける通信障害は、ハードウエア故障や設定ミスなどの一般的な障害に加え、地震、火事など自然災害のような外的要素にも起因し得るからである。

このような障害に対処する手法としてハードウエアの多重化などがある、しかし、人手による管理方法には限界があるという見方もある。近年のインターネットの拡大に伴い、システム管理にかかるコストは爆発的に増大しているからである。そこで、コンピュータがそれぞれ自律的に障害などの環境変化に対応し、互いに協調動作することで、全体としての正常さを保とうとするアプローチが注目を集めている。

我々は、特定のシステムを対象とした自律協調化による環境変化への対応を目指すのではなく、汎用的な自律協調型基盤システムの実現を目指す。サービスはこの基盤システムを利用することで、容易に環境変化に対応することが出来るようになる。本稿では、以降の自律的に環境変化を乗り越えることが出来るサービスを Sustainable Service と呼ぶ。

Sustainable Service の実現に向けて、個々の計算機がどのような動作を行うべきかを考察する事が必要である。現在、実機を用いて実験を行っているが、この手法では実験にコストがかかり、全体の動きを把握しにくい。そこで、個々の計算機の動作をシミュレートし、結果を可視化するソフトウエアを開発する。

2. 概観

Sustainable Service システムは、インターネットに接続された複数の計算機によって構成される。サービスプログラムは、この計算機群上で実行される仮想機械上で動作している。システムは、このサービスプログラムを実行し、クライアントからのサービス要求に対応する。以降では、仮想機械を実行しているシステムの計算機をサーバと呼び、その他の計算機のことをサーバ候補と呼ぶ。サーバとサーバ候補は Peer-to-Peer ネットワークを構成しており、必ずしも全対全の通信ができなくてもよい。システムは、サービスプログラムの状態をこの Peer-to-Peer ネットワーク上で共有し、障害によってサービス不能状態に陥った場合は、サーバ候補群から新たなサーバが選ばれてサービスの提供を継続する。

サーバは、障害が発生した後で新しいサーバがサービスの提供を継続できるように、サービスプログラムの状態を仮想機械の実行状態として取得し、サーバ候補の構成する Peer-to-Peer ネットワーク上で共有する。

サーバ候補群は、サービスプログラムが正常にサービスを提供できているかを監視している。この監視で異常が検出された場合、サーバ候補群から新しくサーバとして振る舞うサーバ候補（次期サーバ）が選出される。次期サーバは、Peer-to-Peer ネットワーク上で共有されているサービスプログラムの状態を収集し、それらを用いて障害が起こる直前

* 筑波大学 情報学類 計算機システム主専攻
† 筑波大学大学院 理工学研究科
‡ 筑波大学大学院 システム情報工学研究科
§ 科学技術振興機構

のサービスプログラムの状態を復元して、サービスの提供を継続する。

サービスプログラムは、負荷分散やスループットの向上を目的として複数のサーバ上で実行することが可能である。また、サービス要求の変化などに応じて、その規模を縮小することも可能である。

3. 実装

Sustainable Service システムの中で自律して動作する単位をノードと呼ぶ。ノードが複数集まりノード群として動作する時、各ノードがどのような動作をすればシステム全体の sustainability を持つかを考える事が重要である。

ノードを複数個シミュレータ上で動作させ、Sustainable Service としての全体の動きをシミュレートし、視覚化する。シミュレータ内の各ノードは仮想的に作成されたネットワークによって他のノードと接続している。ネットワークは通信不能になる可能性があり、この場合ノード間の通信は行えない。

シミュレータは大きく分けて次の 2 つのモデルを持つ。一つは、ネットワークのモデルである。ネットワークの分断はどの地点で発生するか予想する事はできない。そのため、本シミュレータでは、ネットワークの任意の地点で通信障害が発生するモデルを用意した。二つ目は、各ノードのモデルである。各ノードはオートマトンで記述し、自己の周りのネットワークや、自己が通信可能なノードの状態を元に状態遷移を行う。

シミュレータは、これらのモデルを動作させると共に、モデルの状態の可視化を行う。

シミュレータ全体を Java を用いて実装した。