高可用性グリッドシステムのノード配置に関する研究

山田博之† 南澤吉昭† 大場善次郎†
Hiroyuki Yamada† Yoshiaki Minamisawa† Zenjiro Ohba†
†北海道大学大学院情報科学研究科
†Graduate School of Information Science and Technology, Hokkaido Univ.

要旨
社会を担う情報システムは、重度な障害に対してもシステムが稼働し続けることが求められる。このような高可用性を実現可能なグリッドシステムは、システム内の大多数のノードが適切に配置されていなければ高可用性を実現できない。本研究では、多数のノードから構成される高可用性グリッドシステムのノード配置を検討するために、システムの定式化および、可用性の指標であるシステムの稼働率を計算するためのシミュレーション手法を提案する。

1. はじめに

近年、社会基盤として重要な役割を果たしている情報システムは、障害発生時においても高い可用性を維持することが求められ、情報システムにおいて障害対策は考慮すべき問題となっている。

この問題を解決するための手段の一つとして、グリッドと呼ばれるシステム構成が存在する。従来は可用性の高いシステムを構築する為に、ノードやノード間回線の多重化という構成がとられてきた。一方、グリッドを適用したシステムは可用性が高まるだけでなく、全体の負荷の最適化が可能であることや拡張性に優れているという従来の構成にはないメリットが存在する。

しかし、グリッドは「多数のノードからなるシステム構成」という性質を持つため、ノード配置の組み合わせが非常に多くなってしまう。そのため高可用性を持つように、適切にシステム内のノードを配置することは非常に複雑な問題である。そこで本論文では、グリッドシステムのノード配置を検討するために、ノード配置から稼働率を求めるためのシミュレーション手法を提案する。

2. 関連技術

高可用性システム[1]を実現するための多重化の手法として、フェイルオーバが最も一般的である。運用サーバと待機サーバから構成される、フェイルオーバを用いたシステムは、障害時の業務の引き継ぎ中はサービスを提供することはできないので、引き継ぎ時間を短くすることが可用性を高めるための条件となる。現在ではこの引き継ぎ時間を短くするための様々な研究がなされている[2,3]。

3. グリッドシステム

グリッドは「複数の組織から動的に構成される一つの仮想的な組織体の中での、リソース共有と問題解決のための協調」と定義されている[4]。本論文では、グリッドシステムとはグリッドを適用したシステムと定義する。

グリッドは、システムを構成するノードのリソースを仮想的に共有する性質から、高可用性システムを構築することが可能である。それだけではなく、複数のプログラムの負荷を動的に変更したり、状況に応じてノードを動的に追加したりすることが可能である。このようなシステムはコスト面から、ビジネスの分野で有用である。また、グリッドシステムを構成するためのツールの一つである、Globus Tool Kit[5]はインターネットを介したグリッドシステムの構築が可能である。そのため、高価な専用線を用いる必要がない。

以上のように、高可用性グリッドシステムはフェイルオーバと比較して、コスト面に優位点が存在する。しかしグリッドシステムは、多数のノードから構成される場合、高可用性を持つように適切にシス

テム内のノードを配置することは非常に複雑な問題である。本論文ではこの問題を解決するために、グリッドシステムを構成するノード配置に対しての稼働率を求めるためのシミュレーション環境を構築する。

4. シミュレーション

4.1. アプローチ

システムは複数のコンポーネントから構成される。コンポーネントとは、システムを運用するために動作するプログラムのことである。各コンポーネント間で特定の協調を行うことにより、システムは運用状態となる。システム運用に必要なコンポーネント協調が一つでも行われていないと、システムは非運用状態である。協調を行うには、コンポーネント間で通信が行えなければならない。

システムを構成するノードは、単一もしくは複数のコンポーネントとして動作する。コンポーネント間で通信が行えるということは、それぞれのコンポーネントとして動作しているノード同士で通信が行えなければならない。ノードおよびノード間の通信回線の故障が生じてしまうと、それらのノードが担当しているコンポーネント間の通信が行えないということになる。ノードおよびノード間通信回線の故障は、ハードウェアの故障などの内的要因および災害などの外的要因によって生じる。

図 1 のような、Web サーバ、アプリケーションサーバ、DB サーバの 3 つのコンポーネントから構成される Web アプリケーションシステムを例にとる。Web サーバの役割をするノードのどれかと、アプリケーションサーバの役割をするノードが通信可能で、かつアプリケーションサーバの役割をするノードと DB サーバの役割をするノードが通信可能な状態であれば、このシステムは運用状態である。

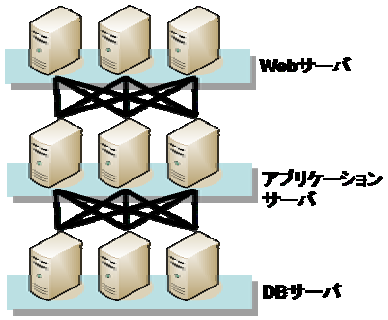

図 1　Web アプリケーションシステム

本論文では、システムの稼働率をシミュレートするために、システムに関連する「ノード」、「コンポーネント」、「回線」、そして稼働率に関連する「障害」の定式化を行う。

4.2. 定式化

ノード

ノードとは任意の処理を行うことが可能で互いに通信することが可能なコンピュータのことである。通信は回線を介して行われる。ノードは $n \in N$ と表す。ただし、システムにおけるノード全体の集合を N とする。ノードは一定の確率で故障してしまう。その故障率を求めるための関数を break_node と定義する。すなわち

$$\text{故障率} = break\_node(n) \qquad n \in N \tag{1}$$

である。ノード n が動作可能なとき、論理式 $IsRunning(n)$ は真であるとする。

回線

ノード間を 1 対 1 で接続するものを回線と定義する。現実には、ノード間にはルータやスイッチなどを介することがあるが、それらは考えないものとする。ノード $n_a, n_b \in N$ を接続する回線を $c_{ab} \in C$ と

表す。ただし、システム内の全回線を表す集合をCとする。回線は一定の確率で故障し、その回線に接続されているノード間での通信が行えなくなる。その故障率を求めるための関数を break_connection と定義する。すなわち

$$\text{故障率} = break\_connection(c) \qquad c \in C \tag{2}$$

である。回線cが通信可能なとき、論理式$IsConnecting(c)$は真であるとする。

コンポーネント

システムを構成するコンポーネントを$com \in Com$と表す。ただし、システムを構成する全てのコンポーネントをComとする。また、コンポーネントcomとして動作可能なノードの集合を$\{n_1, n_2, \ldots, n_k\}$とする。ただし、$1 \le k \le$「$N$の要素数」とする。

システム運用のために、コンポーネントcomとして動作するべきノードを求めるための写像$Assign$を

$$Assign : Com \to N \tag{3}$$

とする。

コンポーネント協調

コンポーネントcom_1, com_2が協調するとき、その協調関係を

$$r = (com_1, com_2) \qquad com_1, com_2 \in Com \tag{4}$$

と定義する。$r \in R$とする。ただし、システム全体におけるコンポーネント協調集合をRとする。

システム

ノード集合N_s、回線集合C_s、コンポーネント集合Com_s、コンポーネント協調集合R_sからなるシステムを

$$S = (N_s, C_s, Com_s, R_s) \tag{5}$$

と定義する。

運用

システムが運用状態かどうかを論理式$IsOperating(S)$と定義すると、

$$\begin{aligned} &\forall r = (com_a, com_b) \in R_s; \\ &n_a = Assign(com_a) \wedge n_b = Assign(com_b) \\ &\wedge IsRunning(n_a) \wedge IsRunning(n_b) \wedge IsConnecting(c_{ab}) \\ &\Rightarrow IsOperating(S) \end{aligned} \tag{6}$$

であるとき、システムは運用状態である。システムの障害復旧時に写像$Assign$の値域が変化することがある。

障害

一定の故障率によってノード、回線が内的要因により故障することは先に述べた。ノード、回線はさらに外的要因による障害のため、故障することがある。障害Dは

$$D = (N_D, C_D) \quad N_D \subset N, C_D \subset C \tag{7}$$

と定義する。ただし、N_D, C_D はそれぞれ障害 D により故障するノード、回線の集合である。また、N, C はそれぞれシステム S を構成するノード、回線の集合である。内的要因による故障の場合、N_D もしくは C_D の要素数は 1 となる。

復旧

故障および障害が起こったノード、回線を復旧するのには時間がかかる。復旧にかかる時間を求める関数を *Recover(e)* と定義する。ただし、e はノードもしくは回線である。

4.3. 実現

図 2 のフローチャートで表現されている処理を実装することで、シミュレーションを実現する。

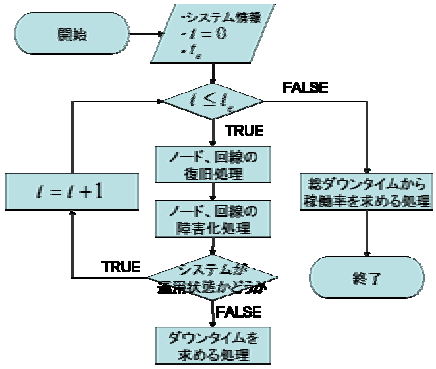

図 2　シミュレーションフローチャート

入力データのシステム情報は、ノード集合 N_s、回線集合 C_s、コンポーネント集合 Com_s コンポーネント協調集合 R_s である。また、t_e はシミュレーション終了時間である。「ノード、回線の復旧処理」にて一定確率でノードと回線を故障させる。また、一定確率で障害 D を発生させる。故障、および障害時に復旧にかかる時間を用いて、以下の式で稼働率を求める。

$$\text{稼働率} = 1 - \frac{\text{総ダウンタイム}}{t_e} \tag{8}$$

5. まとめと今後の課題

本論文では、グリッドシステムの稼働率を求めるためのシミュレーション環境を構築するために、システムの定式化を行った。また、定式を用いてシステムの稼働率を求める手法の提案を行った。

今後は、グリッドシステムのノード、回線、プロセス、コンポーネントを与えて、実際に稼働率を求めるシミュレーションを行う。また、シミュレーション結果を元に、高可用性グリッドシステムを構築するためのノード配置について考察を行う。

参考文献

[1] Jim Gray, Daniel P. Siewiorek, “High-Availability Computer Systems,” Computer vol. 24 no. 9, IEEE, 1991, pp.39-48.

[2] Alex C. Snoeren, David G. Andersen, and Hari Balakrishnan, “Fine-Grained Failover Using Connection Migration”, USITS '01 Paper, 2001, pp.221–232.

[3] Yuanyuan Zhou and Peter M. Chen and Kai Li, “Fast Cluster Failover Using Virtual Memory-Mapped Communication”, ICS '99: Proceedings of the 13th international conference on Supercomputing, ACM Press, 1999, pp.373-382.

[4] Ian Foster and Carl Kesselman and Steven Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, Int. J. High Perform. Comput. Appl., Vol.15 No. 3, 2001, pp.200-222.

[5] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," The Grid: Blueprint for a New Computing Infrastructure, I. Foster and C. Kesselman, eds., Morgan Kaufmann, San Francisco, 1999, pp. 259-278.