4E-01

IT システム構成可視化のためのノード配置適正化方法

鈴木克典[†1]　金子聡[†2]　林真一[†3]
（株）日立製作所

1. はじめに

オンプレミスデータセンタの大規模 IT システムでは，障害発生時に，まず IT システムの構成や各リソースの状態を基に原因箇所を特定する．同時に障害の影響を受けるアプリケーション（以降 App と記載）を確認し，外部への影響の大きさを把握する．これらを踏まえて対応方針や，複数障害発生時の対応優先度付けを決定する．これらの調査を素早く行うためには IT システムの構成や状態の可視化が有用であり，システムを構成するリソースをノード，リソース間の関連をエッジとするグラフで IT システムの構成と状態を表示するようなシステム構成表示が用いられている．これにより，全体から障害箇所部分へグラフの特定部分を拡大し，表示範囲を絞り込んでいくことで，全体構成を把握しつつ，徐々に詳細情報を得ていくことが可能となる．

そこで，本稿では大規模 IT システムであっても，障害対応などの運用に適したシステム構成表示を行うためのグラフレイアウト手法を検討した結果を報告する．

2. IT システム構成表示における課題

IT システムの構成表示ためには，IT システムを示すグラフについて，全体的に高い可読性を保ちつつ，障害原因ノード付近を拡大した際に原因ノードと関連 App ノードを同一画面で表示できるグラフレイアウトが必要となる．

可読性の高いグラフレイアウトを作成する問題は NP 困難な最適化問題として定式化されるが，これを効率的に解く方法として杉山フレームワークが知られている．杉山フレームワークを適用することで，ノードが一様に配置され，かつ辺の長さが短くなるグラフレイアウトが作成できる[1].

一方，このグラフレイアウトを IT システム構成表示に用いた場合，ノードを一様に配置しようとするため，障害原因リソースと，関連するアプリケーションの距離が離れる場合があった．その場合，障害原因リソースと，関連するアプリケーションを同時に一画面に表示できず，利便性が低下してしまう．そこで，本稿ではインフラリソースの状態やコストなどの観点を加味し，ユーザが着目しやすいリソースの近くに，関連するアプリケーションを配置するグラフレイアウト方法を提案する．

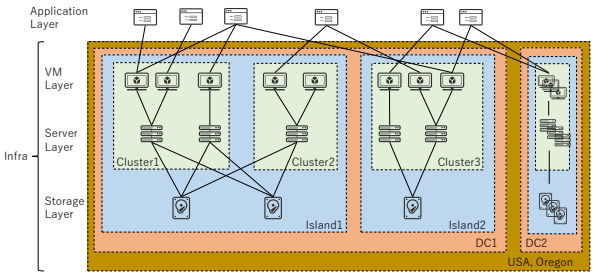

図 1　IT システムの例

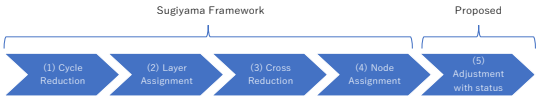

図 2　グラフレイアウト計算の流れ

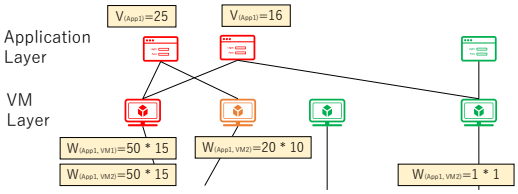

図 3　App ノード座標調整計算の例

3. IT システムの特長と想定するグラフ構造

図 1 に想定するシステムの例を示す．想定システムは Web サービスなどのアプリケーションや，VM，サーバ，ストレージなどのインフラリソースから成る．また，システム構成表示で描画するグラフには以下の特長を想定する．

- 一般的にシステム構成図では VM はサーバの上方に記述するなどの慣例がある．そのため，App – VM – Server – Storage の順のように，ノードはリソース種類毎に決められた高さに配置される．
- インフラリソースはサーバクラスタやなどの管理グループ，また，データセンタ（DC）や国など地理的な単位でグルーピングされて管理される．そのため，インフラリソースノードは入れ子構造のグループに属し，グループ毎に分割されてノードが配置される．
- App ノードはインフラリソースの各管理グループに依存せずに配置される．

A Node Assignment Method for Visualization of IT System
†1 MASATO TERADA, Hitachi Ltd.
†2 SATOSHI KANEKO, Hitachi Ltd.
†3 SHINICHI HAYASHI, Hitachi Ltd.

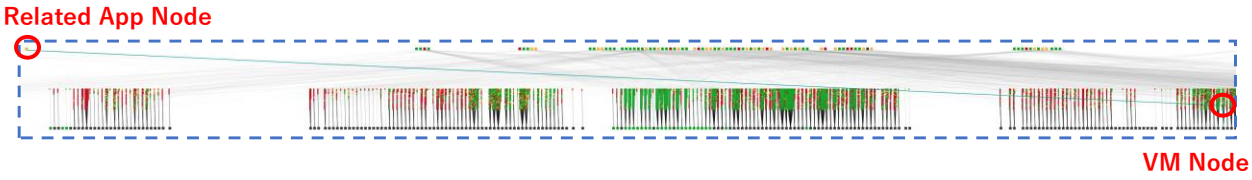

図 4　ノード配置調整適用前の App ノード配置例

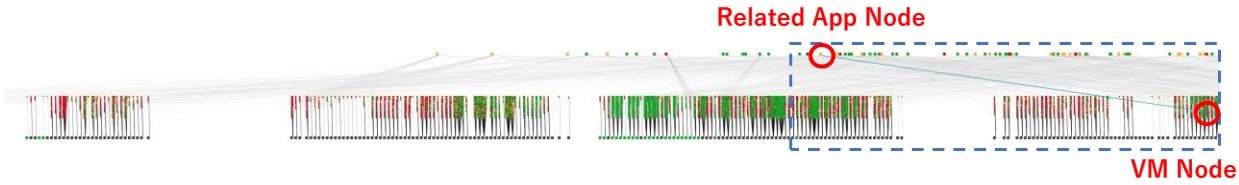

図 5　ノード配置調整適用後の App ノード配置例

4. 提案グラフレイアウト方法

図 2 に提案手法の流れを示す．提案手法では杉山フレームワークを拡張し，グラフ全体のレイアウトを計算後，インフラリソースの情報を基に App ノードの配置を調整する．

（1）閉路除去，（2）階層割当のフェーズは，本稿で想定するグラフが閉路を含まず，また，ノード種類によって配置される高さが決まることから特別な処理はない．（3）交差削減と（4）ノード座標割当は文献[2]と文献[3]に記載される方法で計算する．次に（5）ノード配置調整で App ノードの座標を調整する．手順は以下の通りである．

1.　インフラリソースノードの重み付け

App ノード n_i から見た，関連するインフラノード n_j の重み $W_{(i,j)}$を求める．本稿では $W_{(i,j)}$をノード状態 Status とコスト Cost から求めた．ノード状態とは例えば性能異常の有無である．本稿では，図 3 に示す様に異常状態のノードの場合には Status = 50 等の一定値を計算に用いた．コストとはリソースに要する費用であり，リソースの重要性を示す指標として利用した．なお，App はリソースの能力を共有していることから，コストは App 数で分割して計算に用いた．

$$W_{(i,j)} = Status_{(j)} * Cost_{(i,j)} \quad (1)$$

2.　App ノード座標の更新

App ノード n_i の x 座標 x_i を，関連するインフラノード n_j の x 座標 x_j の加重平均として求める．

$$x_{(i)} = \sum (w_{(i,j)} * x_{(j)}) / \sum (w_{(i,j)}) \quad (2)$$

3.　App ノードの間隔調整

適切な間隔を保つように，App ノードの x 座標を調整する．App ノード n_i の重要度 V_i を計算し，重要度が高い App から順に，すでに x 座標調整済みの App ノードから一定距離以上離れつつ，最もステップ 2 で求めた重心に近い点を調整後 x 座標とした．本稿では重要度 V_i を App ノード n_i の関連するインフラノード n_j のコスト $Cost_{(i,j)}$の総和とした．

5. 提案手法の適用結果

提案手法の（5）ノード配置調整を行わない場合（適用前）と，行う場合（適用後）で IT システムのグラフをレイアウトした例を図 4 と図 5 に示す．グラフの大きさは，縦 800px，幅 14,600px であった．これは App 数 100 個，VM 数 12,000 個，物理サーバ数 350 個の IT システムである．図 4 と図 5 には，状態の悪い VM と，関連する App の組のうち最も表示上の距離が近づいた例を示す．本例では状態の悪い VM と，関連する App との距離が適用前後で約 1/2 (5,855px)となっていた．これは，適用前は関連する全 VM を同等に考慮して App の配置が決定されていたためである．

なお，適用前での状態の悪い VM と，App との距離の平均値は 3,298px，適用後では 2,346px であった．

6. おわりに

大規模 IT システム構成の可視化向けに，グラフをレイアウトする方法を検討した．IT システムの障害対応では原因リソースの特定と，影響するアプリケーションの特定を迅速に行う必要があり，システム構成可視化でも原因リソースとアプリケーションを同一画面で確認できる必要があった．そこで，杉山フレームワークを拡張し，リソースの状態やコストを加味してグラフレイアウトを計算する手法を提案した．大規模 IT システムの実データを用いて評価した結果，異常状態の VM ノードと App ノードの間の表示上の距離が，提案手法の適用前後で 1/2 となる例を確認した．

参考文献

[1] 尾上洋介，階層グラフの可視化，日本オペレーションズ・リサーチ学会 機関紙 Vol. 63 No.1 (2018), PP. 20-26
[2] Forster M., “Applying Crossing Reduction Strategies to Layered Compound Graphs”, Graph Drawing 2002. Lecture Notes in Computer Science, vol 2528, pp. 276-284
[3] Sander, G. “Layout of Compound Directed Graphs”, 1996, UNIVERSITAT DES SAARLANDES