「データ宝石箱 II」を用いた分散プロセスの可視化

伊藤 貴之○，山口 裕美 (日本アイ・ビー・エム(株))

Visualization of Distributed Processes Using "Data Jewelry Box II" algorithm

Takayuki ITOH, and Yumi YAMAGUCHI

ABSTRACT

It is important to visualize status of distributed processes for the purpose of management of very large-scale distributed computing systems. This paper proposes the architecture for visualization of distributed processes using "Data Jewelry Box II" algorithm, which is suited to visualize seamless time-varying data. This architecture categorizes a set of processes, and creates hierarchical structure to monitor status of processes by applying "Data Jewelry Box II". Our implementation classifies processes into "assigned process" and "unassigned process", and again classifies the assigned processes according to the instances which processes are assigned. Then, our implementation displays whole of distributed processes in one window, because “Data Jewelry Box II” is suited to visualize large-scale hierarchical data in a small display area.

Keywords: Visualization, Distributed processes, Data Jewelry Box

1. はじめに

ネットワーク上にあるヘテロなコンピュータ資源をフレキシブルに利用できる新しい大規模分散計算環境として、Grid Computing [Grid] が近年注目されている。Grid Computing に新しく導入される Open Grid Services Architecture (OGSA)[OGSA]では、サービス(サーバー上に公開されているソフトウェアコンポーネント)の実体を動的に確保することができる。そのため、あらかじめ計算機資源の利用状況が把握できれば、ユーザはサービスの実体を複数の計算機上に分割して確保することで、計算機資源を効率よく利用することができるようになる。

計算機資源を適応的に活用できるOGSAのような分散計算環境が普及すると、計算機資源の利用状況は管理者が予測できないほど複雑になることが予想される。そこで著者らは、大規模な分散計算環境の管理・設計において、分散プロセスをリアルタイムに監視することが有用であると考えた。実際に、Hewlett Packard 社の OpenView[hp]のように、グラフィックス技術を用いて分散環境を監視するツールが製品化されている例がある。

一方、著者らは、大規模な階層型データの全貌を視覚化する新しい技術として、「データ宝石箱」アルゴリズムを提案している[Ito01][Ito03]。この手法は、階層構造を長方形の入れ子構造で表現することで、限られた画面空間に大規模データの全貌を高速配置している。図 1 は、「データ宝石箱」を用いて、ウェブサイトを構成するウェブページ群をディレクトリ構造で階層化したデータを視覚化したものである。著者らは「データ宝石箱」を拡張し、時間変化に沿って微量ずつ変化するデータをシームレスに表現する「データ宝石箱 II」アルゴリズムを提案している[Yam02][Yam03]。

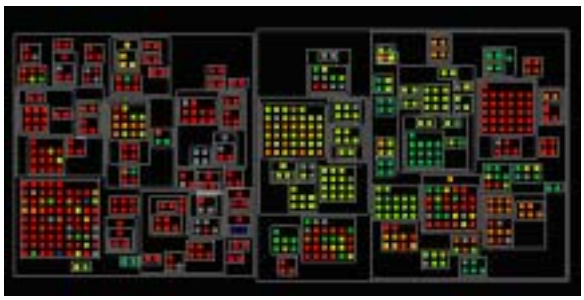

Fig 1. 「データ宝石箱」アルゴリズムを用いて画面配置した階層型データ

著者らは「データ宝石箱 II」を用いて、刻々と変化する分散計算環境の利用状況を視覚化するためのプロトタイプを開発した。本報告では、OGSA 上で開発した分散プロセス監視ツールと、それによる視覚化例を紹介する。

著者らは「データ宝石箱 II」について、以下の点において分散プロセスの監視に適していると考えている。

[特徴 1: 高速処理] 分散プロセスの分布は刻々と変化するので、それに瞬時に応答して現時点のプロセス分布を表示する必要がある。「データ宝石箱 II」は数千ノードを持つ階層型データを数秒の計算時間で画面配置できるので、この要求を満たすことができる。

[特徴 2: 大規模データへの適用] 分散プロセスの数は大規模な計算環境では数千、数万に及ぶことがあり、この

全体を一画面に配置できることが監視ツールの重要な要求である。「データ宝石箱 II」は、できるだけ小さい画面空間にデータ全体を配置することで、この要求を満たすことができる。

[特徴 3: 構造型データの表現] 数千、数万に及ぶ分散プロセスの概略を理解するには、プロセス分布の構造を表現できることが望ましい。「データ宝石箱 II」は階層型データを表現する手法なので、この要求を満たすことができる。

[特徴 4: 類似データの類似表現] 分散プロセスの時間変化をシームレスに監視するためには、類似するデータに対して類似した表示結果が得られることが望ましい。「データ宝石箱 II」では 2 章に示すとおり、直前時刻における画面表示結果を参照して現在時刻におけるデータを画面配置するので、この要求を満たすことができる。

2.「データ宝石箱 II」アルゴリズム

「データ宝石箱」[Ito01][Ito03]は図 1 にも示した通り、階層型データを構成する葉ノードを長方形のアイコンで表現し、枝ノードを長方形の枠で表現する。また、長方形の枠の入れ子構造によって、データの階層構造を表現する。そして、以下の条件

[条件 1] 長方形群の占有領域が小さくなること
[条件 2] 長方形どうしが重ならないこと
[条件 3] 占有領域の縦横比ができるだけ 1 になること
[条件 4] 高速に配置すること

を満たすように、階層型データを表現する長方形群を画面空間に配置する。

「データ宝石箱」の拡張手法である「データ宝石箱 II」[Yam02]は、上記の条件に加え

[条件 5] ユーザが配置したい位置にできるだけ近くに配置させること

を条件として長方形群を画面空間に配置する。

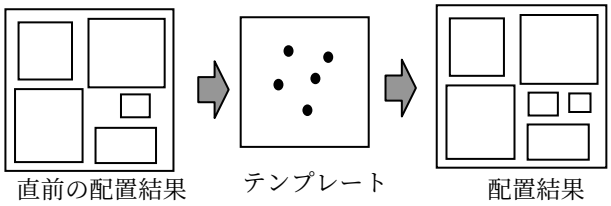

Fig 2.「データ宝石箱 II」アルゴリズムの概要

「データ宝石箱 II」では、あらかじめユーザがノードを配置したい位置を「テンプレート」ファイルに記述しておき、これを参照しながら長方形群の画面上の位置を決定する。本報告のように、時間変化に沿って変化するデータをシームレスに表示したい場合には、直前の配置結果をテンプレートに格納する（図 2 参照）。

以下、階層型データ中の 1 階層を構成するノード群の画面配置手順について説明する。まず、面積が最大となる長方形を、テンプレートに記述された位置に配置する。2 個目以降の長方形の配置順番は、テンプレートに記録された位置が、最初に配置した長方形に近い順、とする。この配置順番により、長方形の画面上の配置関係が崩れるのを防ぐことが出来る。画面上の適切な位置を高速に決定するために本手法では、占有領域の 4 頂点と、既に配置されている長方形の中心点で Delaunay 三角メッシュを生成し、三角メッシュを参照しながら[条件 1]〜[条件 5]を満たす位置を検出する。詳細は参考文献[Yam02][Yam03]を参考にされたい。

3．分散プロセスの視覚化への適用

本章では、「データ宝石箱 II」を OGSA 上で稼動する分散プロセス監視に適用したアーキテクチャについて述べる（図 3 参照）。本アーキテクチャ上で稼動するサービスは以下の通りである。

[親サービス] 分散プロセスの管理、モニタリングサービスへの分散プロセス管理結果の通知、クライアントとの通信、を担当する。
[子サービス] アプリケーションが必要とする各処理を担当する。
[モニタリングサービス] 「データ宝石箱 II」による分散プロセスの視覚化を担当する。

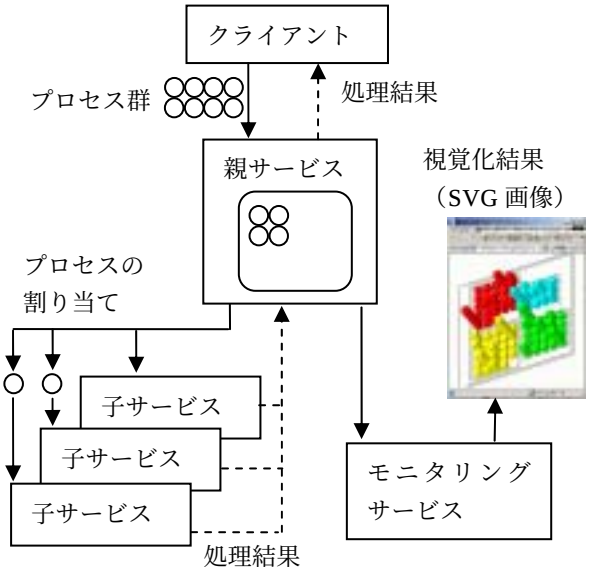

Fig 3．分散プロセスとモニタリングのアーキテクチャ

まずクライアントは親サービスと通信し、クライアントによるプロセス群のリクエストを親サービスに受け渡

す。親サービスはリクエストされたプロセス群を、順々に子サービスに分配する。子サービスは、受け取ったプロセスを処理し、親サービスに処理結果を渡す。全てのプロセス群の処理を終了したら、親サービスはクライアントに全ての処理結果を返す。以上の処理の過程で、親サービスはプロセスの処理状況をモニタリングサービスに渡し、モニタリングサービスは「データ宝石箱 II」を用いてプロセスの状況を視覚化し、その結果を Scalable Vector Graphics (SVG)画像として作成する。SVG 画像は XML に準拠した 2D の画像フォーマットであるので、ウェブブラウザ上に表示することができる。そのため、クライアントは SVG 画像で表示されたプロセス状況を、どこからでもウェブブラウザを通して確認することができる。

分散プロセスの処理状況を「データ宝石箱 II」で表現するために、分散プロセスを階層型データとして構造化する必要がある。本報告では図 4 に示すように、プロセスの集合を親サービスが階層構造に分類する。本報告では各プロセスを、子サービスで処理中または処理終了状態（既処理）か、未処理か、に二分する。プロセスが既処理の場合、割り当てられた子サービスごとにプロセスを分類する。このようにしてプロセス群を階層型データに見立てて、「データ宝石箱 II」アルゴリズムを適用する。

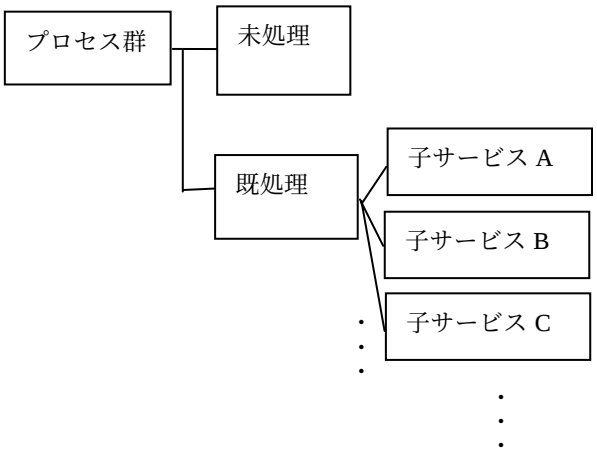

Fig 4. プロセス群の分類例

4. 実験

本章では、3 章で説明したアーキテクチャのプロトタイプを実装した結果を紹介する。本実験では、分散計算のアプリケーションとして、著者の一人が既に提案しているメッシュ分割手法を採り上げた [Shi00][Vis00]。本実験では、80 のパーツからなる機械部品をクライアントから親サービスに渡すと、親サービスはメッシュ計算を処理する 4 つの子サービスに各パーツを割り当てる。このとき、できるだけ短時間で全てのプロセスを終了するために、各パーツに対するメッシュ分割の予想所要時間を概算し、予想所要時間の大きいプロセスから順に子サービスに割り当てる。詳細は参考文献[Yam03]を参考にされたい。

一方モニタリングサービスは、プロセス群の階層型データを定期的に親サービスから受け取り、「データ宝石箱 II」を用いて視覚化する。本実験では、10 秒ごとに親サービスから階層型データを受け取っている。一連の視覚化結果を図 5 に示す。図 5 では、各々の棒が各々のプロセスを表現している。左枠の中の棒は未処理プロセスの集合を、右枠の中の棒は既処理プロセスの集合を表している。既処理のプロセス群は、さらに 4 つの子サービスごとに分類表示されている。また図 5 では、未処理のプロセスを青で、その他のプロセスは、子サービスごとに色付けをしている。プロセスの高さは処理時間を示している。

図 5 から、プロセスがどのように 4 つのサービスに割り当てられ、どのサービスが処理時間の大きいプロセスを担当しているか、どのサービスがより多くのプロセスを担当しているか、といった処理状況を視覚化できていることがわかる。

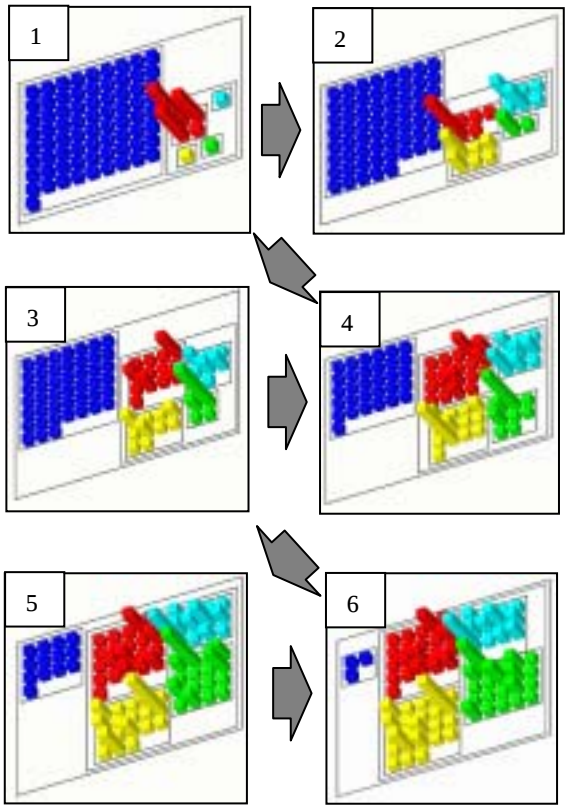

Fig 5. 分散プロセスの視覚化の様子。青い棒が未処理のプロセス、その他が子プロセスに割り当てられた既処理のプロセス。時間を追うごとに、未処理のプロセスが減少していることがわかる。

図 6 は、複数のユーザによる分散プロセスを 1 画面に

視覚化した例である。この視覚化に用いたデータは、図4 に示す階層型データをユーザごとに生成して、これを合成して 1 個の大きな階層型データにしたものである。言い換えればこの階層型データは、プロセス群をユーザごとに分類し、これを未処理と既処理に分類し、さらに既処理プロセスをサービスごとに分類したものである。このようにして複数のユーザによる分散プロセスを集めることにより、分散計算環境全体のプロセス分布を一画面に集約して監視することが可能になると考えられる。

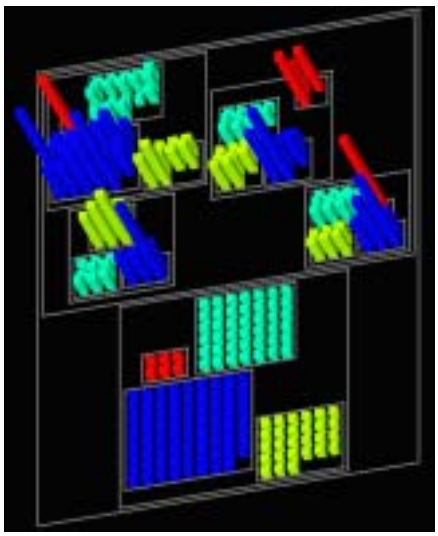

Fig 6. 複数のユーザによる分散プロセスの視覚化。

5. おわりに

本報告では、「データ宝石箱 II」を用いて、OGSA 上で稼動する分散プロセスを監視するアーキテクチャを紹介した。子サービスに割り当てたプロセス群の管理を親サービスに一括させることで、モニタリングサービスは、親サービスと通信するだけで分散プロセスの状況を視覚化することができる。また、「データ宝石箱 II」による視覚化結果を SVG 画像フォーマットで出力することで、誰でもどこからでも分散プロセスの処理状況を把握することができる。

「データ宝石箱 II」は、できるだけ小さい画面空間に階層型データ全体を配置することを特徴としており、大規模な分散プロセス分布の表現に適していると考えられる。また、時間変化に沿って微量ずつ変化する階層型データのシームレスな表現を特徴としており、分散プロセスの刻々とした時間変化の監視に適していると考えられる。

本実験による成果の実用化を検討するために、以下の項目を今後の課題としたい。

- プロセス群を表現する棒の分類、色算出、高さ算出、の基準となる属性に何を選べば効果的な監視ができるか検討したい。まずは、プロセスの特性やメモリ使用量、ディスク使用量など、計算機資源に関するあらゆる属性を用いて、実験を重ねたい。
- 実際に運用されている分散計算環境上で実験し、実際に運用されているプロセス分布に対する監視結果を検証したい。
- モニタリング画像を GUI に拡張し、プロセスの取り消しといった操作をクライアントが直接操作できるようになると、分散環境の管理操作が可能になり、さらに有益になると考えられる。

また関連研究として、Treemaps [Joh91]を用いた Grid Computing の計算機資源の視覚化事例が報告されている[Hei03]。これらの関連研究との比較についても検討を進めたい。

参 考 文 献

[Grid] Foster I., Kesselman C., Tuecke S., The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of High Performance Computing Applications, Vol. 15, No. 3, pp. 200-222, 2001.

[hp] Hewlett Packard, OpenView, http://www.openview.hp.com/

[OGSA] Foster I., Kesselman C., Mick J. M., Tuecke S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.

[Ito01] 伊藤, 梶永, 池端, データ宝石箱：大規模階層型データのグラフィックスショーケース, 情報処理学会グラフィクス&CAD 研究会, 2001-CG-104, 2001.

[Ito03] Itoh T., Yamaguchi Y., Ikehata Y., Kajinaga Y., Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm, IEEE Transactions on Visualization and Computer Graphics, in process.

[Yam02] 山口, 伊藤, データ宝石箱 II ～ 位置情報を用いた大規模階層型データのグラフィックスショーケース, 情報処理学会グラフィクス&CAD 研究会, 2002-CG-108, 2002.

[Yam03] Yamaguchi Y., Itoh T., Visualization of Distributed Processes Using “Data Jewelry Box II” Algorithm, CG International 2003, to be presented on 2003/7/10.

[Shi00] Shimada K., Yamada A., Itoh T., Anisotropic Triangulation of Parametric Surfaces via Close Packing of Ellipses, International Journal on Computational Geometry & Applications, Vol. 10, No. 4, pp. 417-440, 2000.

[Vis00] Viswanath N., Shimada K., Itoh T., Quadrilateral Meshing with Anisotropy and Directionality Control via Close Packing of Rectangular Cells, 9th International Meshing Roundtable, pp. 227-238, 2000.

[Joh91] Johnson B., Shneiderman B., Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space, IEEE Visualization ’91, pp. 275-282, 1991.

[Hei03] Heisig S., Treemaps for Workload Visualization, IEEE Computer Graphics & Applications, Vol. 23, No. 2, pp. 60-67, 2003.