

4D-2 動的モジュール追加機構におけるモジュール状態の可視化

佐藤元信† 高野了成† 早川栄一‡ 高橋延匡‡

†東京農工大学工学部 ‡拓殖大学工学部

1. はじめに

利用形態の多様化やモバイル環境のような動的環境にシステムが対応できるように、動的拡張機構を持ったオペレーティングシステム (OS) が増えている。このような動的拡張機構は、カーネルに対するモジュール追加、独立したプロセスやスレッドによるサーバ、ユーザレベルタスクに対するダイナミックリンクなど形態は様々である。実行時に資源やモジュールの対応を決定するシステムでは、ソースコードだけから資源の利用状況を把握することは困難であり、開発段階では実際にどのモジュールがロードされ、どのモジュールがリンクしているかという全体像を把握したい。

本稿では実行時にモジュールのロード、リンクに関する情報を収集し、モジュール構成などを可視化するシステムについて述べる。このシステムではモジュールに含まれる手続きの呼出し回数などをもとにしてモジュール同士の結合度を計算し、それに応じてモジュールの関係を図示することで直感的な理解の手助けをしている。

2. 研究の目的

動的にモジュール同士の結合を決定するようなシステムでは、手続き呼出しが実際にどのモジュールにリンクされるかを知ることができない。特に開発中のシステムでは環境をすべて把握することは難しく、開発者が想定していたモジュールとは別のモジュールが利用されているようなことがある。また性能を改善するとき、モジュール同士の結合の強さや、どのモジュールが頻繁に利用されているかといった情報が必要になる。そこで、実行時にモジュールの利用状況などに関する情報を収集し、単に数値の表としてではなく開発者に分かりやすい形で図示することを本研究の目的とする。

3. 設計方針

(1) 文字に頼らない直感的な表示

表示すべき情報はモジュールの数に比例して多くなる。このような情報を図示する際、その意味を数値や文字で表現してしまうと、ただでさえ状況を把握するのは困難となる。そこで、モジュール名や関数名と

いった文字列でしか表現できないもの以外は、リンク数やモジュールの大きさといった情報も数値として示すのではなく、色や大きさといった図形表現に置き換える。

(2) 情報の段階的な表示

収集した情報をすべて同時に表示しようとすると、細かな情報まで目についてしまい全体像が把握できなくなる。そこで、モジュールの持つ情報を階層化し、各層ごとに段階的に表示して注目したい部分だけを見やすくする。

4. 設計

4.1 全体構成

可視化システムの全体構成を図 1 に示す。このシステムは情報収集部と可視化ツールから構成される。収集した情報はその場で可視化ツールに渡して可視化することもできるし、ファイルに保存しておいて後で解析することもできる。

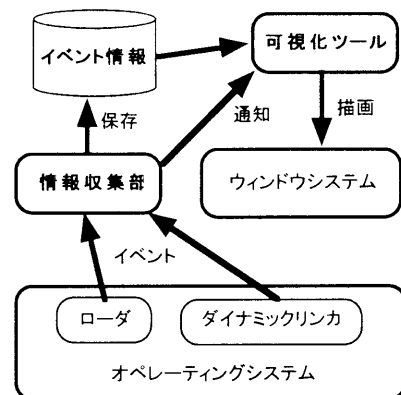


図1 モジュール可視化システムの構成

4.2 情報収集部

情報収集部ではモジュールの状態が変化するイベントを捕らえることでモジュールの情報を収集する。モジュールの状態が変化するイベントは (a) モジュールのメモリへの展開、(b) 展開されたモジュールのメモリからの削除、(c) 他モジュールへの参照 (ダイナミックリンク) の三つである。(a) (b) はローダやメモリ管理の処理であり、(c) はダイナミックリンクの処理である。これらの処理を行う手続きを監視するこ

とで、モジュールに関する情報を得ることができる。

4.3 可視化ツール

可視化ツールでは収集された情報を元に、ユーザが見やすい形で情報を提示する。GUIによりモジュールや手続きのリンク状態を表示し、ユーザが着目したい部分をクリックすることでより詳細な情報を得ることができる。

4.3.1 モジュール情報の階層化

見たい情報だけを見やすくするために情報を階層化し、階層ごとに表示する。ここではモジュールに関する情報を次のように階層化した。

- (1) モジュールとモジュールのリンク状態
- (2) あるモジュールの手続きのリンク状態
- (3) ある手続きの呼出し状況

(1) で着目したいモジュールを選択すると、そのモジュールに含まれる手続きがどのモジュールとリンクされているかを示す (2) の段階になる。(2) である手続きを選択するとその手続きをどのモジュールがどれくらい呼び出しているかを示す (3) の段階になる。

4.3.2 情報と図形の意味

モジュール名や手続き名のような文字列でしか表現できないもの以外は、すべて図として表現する。そこで図における各要素に次のような意味付けを行った。

(1) 形状

矩形をモジュール、楕円形を手続きとして区別した。また、リンク状態はこれらの図形を結ぶ直線として表現する。

(2) 位置

図形の絶対的な位置には意味を与えない。その代わり、図形同士の距離で関連の強さを表現する。つまり手続き呼出しが頻繁に行われているもの同士が近くに集まるようにする。

(3) 大きさ

図形の大きさはモジュールや手続きのコードの大きさを示す。また、図形同士を結ぶ線の太さはリンクの多さを示している。

(4) 色

モジュールや手続きのシステム階層ごとに色を変えることで、そのモジュールがどの空間にマップされているかを示す。

5. 実現

前述の設計に基づいてモジュールの可視化システムを実現した。情報の収集は、ダイナミックリンクを行うとき特定の手続き呼出しをフックすることで行って

いる。この方法であれば、ダイナミックリンク機構を持つOSであればどのOSにも適応できる。フックする手続きは、モジュールのロード、ダイナミックリンクの処理であり、この二つをイベントとして捕らえることができるようにした。

具体的にはPC/AT互換機上で動作するOS/omicon V4用を実現し、手続き呼出しのフックにはスタブ機構 [1] を利用した。また、可視化ツールは未ウィンドウシステムのアプリケーションとして実現した。今回はイベント情報のディスクへの保存部分は実装していないので、可視化ツールがダイナミックリンクなどの手続きをフックし、情報を直接収集している。この実行画面を図2に示す。

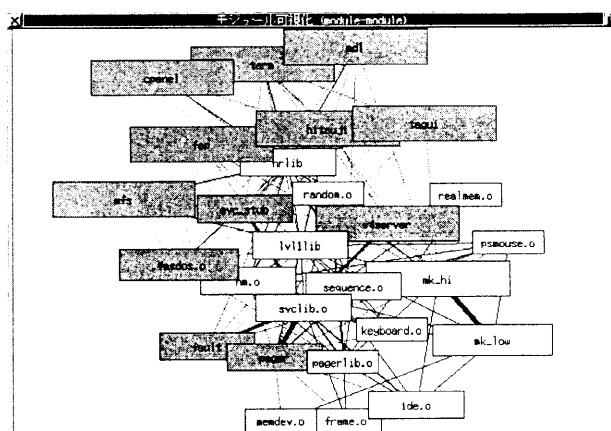


図2 可視化ツールの実行画面

6. おわりに

本稿では動的に拡張可能なシステムにおけるモジュール状態の可視化システムの設計と実現について述べた。このシステムを実装した結果、実行時にモジュールのロードやリンクを監視することで情報を収集し、その情報を直感的に分かりやすいように数値を用いずに図形として提示することができた。今後は未実装であるファイルへの情報の保存の実現や、実際に利用してもらっての評価などを行う予定である。

参考文献

- [1] 佐藤元信, 高野了成, 早川栄一, 高橋延匡: “OS/omicon V4におけるデバッグ支援環境の設計”, 情報処理学会研究報告, 2000-OS-83, pp.19-24, 2000
- [2] Jochen L., “Toward real microkernels”, Commun. ACM 39, pp. 70-77, 1996.
- [3] J. Lamping, R. Rao, and P. Pirolli, “A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies”, CHI'95 Proceedings, 1995.