システムソフトウェアと
オペレーティング・システム 65−16
（１９９４．７．２２）

大域的仮想仮想記憶 (GVVM) の RWC-1 上での性能予測

平野 聡　田沼 均　一杉 裕志　須崎 有康　塚本享治

電子技術総合研究所

hirano@etl.go.jp

超並列システムのための大域的仮想仮想記憶 (GVVM) は、PE 空間全体でのページ使用頻度に基づくページング、及び、メモリ負荷の低い PE 上のメモリをスワップ領域として用いる事により、メモリ負荷の自動負荷分散を行なう。

RWC-1 は新情報処理開発機構 RWCP において開発されている超並列マシンである。本論文では、1024 プロセッサの RWC-1 のシミュレーションが可能なネットワークレベル・シミュレータ ERSE の概略と、ERSE 上に実装した GVVM 処理系を用いて行なった予備的な性能評価の結果について述べる。

Global Virtual Virtual Memory (GVVM): Preliminary Performance Evaluation on a RWC-1 Simulator

HIRANO Satoshi　TANUMA Hitoshi　ICHISUGI Yuuji
SUZAKI Kuniyasu　TUKAMOTO Michiharu

Electrotechnical Laboratory, Japan

hirano@etl.go.jp

The Global Virtual Virtual Memory (GVVM) intends load balancing of memory for massively parallel multicomputer systems by system wide LRU paging and network paging onto other PE's. This paper shows that the GVVM indicates good performance on the RWC-1, that is a MPP machine wich 1024 processors under construction at the Real World Computer Program.

1 はじめに

我々は超並列システム用オペレーティングシステム「超流動OS」を開発している[12]。超流動OSの大域的仮想仮想記憶(GVVM)[10]は、分散メモリアーキテクチャ、及び分散共有メモリアーキテクチャの超並列システム上で、高速な仮想記憶とメモリ負荷の自動的な負荷分散を行なう。

これまでに、2次元トーラス構造のネットワークを有する並列マシン・シミュレータをテストベッドとして、シングルプロセス環境における複数のメモリ負荷分散戦略の評価[10]、プロセスが動的に生成消滅するマルチプロセス環境での評価[11]を行なってきた。

本論文では、新情報処理開発機構RWCPで開発中の超並列RWC-1[8]を対象としてGVVMの予備的な性能評価を行なう。RWC-1はメッセージによる高速なスレッド切替を可能とするRICAアーキテクチャ、PEアドレスの仮想化と大域的なアドレッシング、及び、バンヤン結合をキューブ結合で3次元に結合するCCCBトポロジのネットワークを主な特徴とする。

RWC-1の完成は数年先になるため、RWC-1のネットワークレベル・シミュレータERSE(ETL RWC-1 Simulation Environment)を開発した。ERSEはマルチスレッドで並列動作するC++オブジェクトにRWC-1のCCCB網を介したメッセージ・パッシング機能を提供する。ERSEはアプリケーション・プログラムをネイティブ・コードで並列実行するため、高速である。

ERSE上にプロセス割り付けアルゴリズム、超流動OSの管理情報共有機構MetaShare[7]、GVVM処理系を実装し、それらの予備的な性能評価を行なった。評価の結果、GVVMを用いるとGVVMを用いない場合と比較して大幅な性能向上が可能であること、MetaShareにはRWC-1上でのメモリ負荷分散能力があることがわかった。

以下、2章でGVVMの基本概念についてごく簡単に述べた後、3章でRWC-1の特徴と今回作成したRWC-1のネットワークレベル・シミュレータERSEについて説明する。4章と5章でシミュレータ上に実現したプロセス割り付けアルゴリズムとGVVM処理系の性能評価の結果を示す。

2 大域的仮想仮想記憶の基本概念

本章ではGVVMの基本概念について述べる。複数のプログラムが混在して動作するOSのもとでは、PE内でのメモリアクセスの局所性と共に、複数のプロセスのメモリ使用量の差により生ずるPE空間内でのメモリ使用量の偏りも期待できる。

そこで、大域的仮想仮想記憶(Global Virtual Virtual Memory, GVVM)は、

- 二次記憶装置へのページアウトの対象となるページを、システム全体のPE上のメモリを対象にして決定する大域的仮想記憶

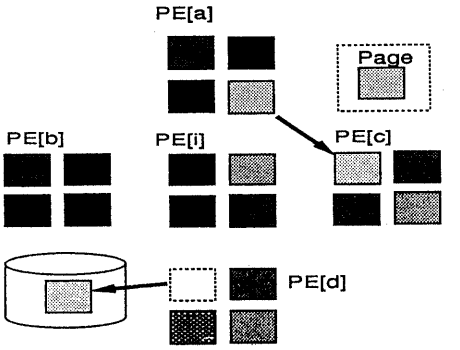

図 1: 大域的仮想仮想記憶の概念

- 他PE上に未使用のページがある場合、そのページをスワップ領域として用い仮想的にページアウトを行なう仮想仮想記憶

を融合し、ネットワークの高速性を生かした高速な仮想記憶機構とメモリの自動負荷分散を実現する。

図1にGVVMの概念を示す。四角形の箱はページを表し、色の濃さはそのページの使用頻度を表している。

PE[a]においてページフォルトが発生したとする。その際、PE[a]上で未使用の実記憶がなかった場合、最も使用頻度の低いページをページアウトし、ページフォルトが発生したページを読み込むための領域を確保しなければならない。PE[a]はシステム中のいくつかのPEを対象として「入札」を行い、自PEよりメモリ負荷(使用頻度の低いページが多い場合負荷が低いとする)の低いPEが発見されたらそのPE上のメモリをスワップ領域としてページアウトを行なう。スワップ領域を提供するPE[c]をスワップ先PE、PE[a]がPE[c]へページアウトすることを仮想ページアウトと呼ぶ。PE[a]で該当ページに対するページフォルトが発生した場合、PE[c]から仮想ページインを行なう。

自分よりメモリ負荷の低いPEを発見できなかった場合はPE[d]のように二次記憶へのページアウトを行なう。

他数のPEを備える超並列システムの場合、PE[a]はシステム中の全てのPEのメモリ負荷を調べるわけにはいかない。そこで、どのPEに対し入札を行なうかを決定する負荷分散戦略が重要である[10, 11]。5章では3種類の負荷分散戦略を比較する。

3 RWC-1とネットワークレベル・シミュレータERSE

RWC-1は新情報処理開発機構RWCPで開発中の分散メモリ型の汎用超並列マシンで、以下のような

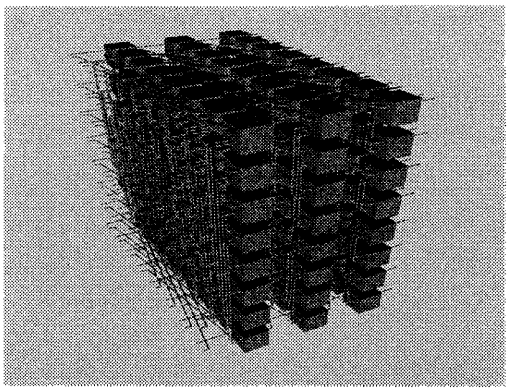

図 2: CCCB 網

特徴を有する [8]。

- RICA, Riduced Interprocessor Communication Architecture
- CCCB トポロジの相互結合網
- PE 番号や大域的なアドレスの仮想化など超並列 OS の支援機構
- I/O ネットワーク

超並列アーキテクチャの並列処理性能を引き出す上で、プロセッサ間のメッセージ交換に関わるオーバーヘッドの存在は最も本質的な問題である。RICA は、プロセッサの RISC パイプラインと、プロセッサ間のメッセージの循環路、即ち、PE に内蔵されたメッセージハンドリング・ハードウェアを融合することにより高速なメッセージ交換を実現する。プロセッサによって処理されるのは、メッセージによって駆動される RISC 命令列である。これは、並列オブジェクトをスレッドとするソフトウェアモデルと高い適合性を有する。作成される RWC-1 の PE 台数は最大 1024 台が見込まれている。

RWC-1 のネットワークトポロジは Multidimensional Directed Cycles Ensemble の一形態として、X-Y 平面をバンヤン結合 (Circular Banyan)、X-Z 平面をキューブ結合 (Cube Connected Cycles) とする CCCB 網である。CCCB 網は次数が小さく、大規模化しても直径が小さいといった特徴を有する [6]。図 2に文献 [6] より引用した CCCB 網の概観を示す。X 次元のプロセッサはまとめてクラスタと呼ばれ、I/O ネットワークとディスクを共有する。

なお、I/O ネットワークの仕様は定まっていないため、本論文で行なう評価は CCCB 網を用いる。

3.1 RWC-1 のネットワークレベル・シミュレータ ERSE

RWC-1 のシミュレータとして、ハードウェアの検証用のレジスタトランスファーレベルのシミュレータや、RWC-1 のインストラクションを実行可能な機能レベル・シミュレータが作成されている。これらは、RWC-1 開発のための、1PE から数 PE の精密なシミュレーションに向いている。一方で、OS やアプリケーションの開発では、RWC-1 システム全体を含む大規模なシミュレーションを行なう必要がある。そのシミュレータは、

- PE 数が多量であっても、許容できる処理時間でアプリケーションの実行が行なえること。
- RWC-1 の RICA やネットワークトポロジの性能上の特徴がシミュレーションに反映されること。
- 現在存在するの計算機上、できれば一般的に使用可能な計算機上で実行可能なこと。

を満たす必要がある。そこで今回、1000PE から 10000PE 規模のシミュレーションを行なうことを目的としたネットワークレベル・シミュレータ ERSE (ETL RWC-1 Simulation Environment、エルス) を開発した。

3.1.1 ERSE の全体構成と実行モデル

ERSE は、アプリケーションに対して RICA や CCCB 網の一部の機能を提供する。アプリケーションは、ERSE を動作させる計算機のネイティブコードで走るため、インストラクションレベルのシミュレーションを行なう機能レベルシミュレータと比較して高速である。

ERSE の構成を図 3に示す。各要素の役割は以下の通り。

アプリケーション・プロセス ERSE のアプリケーション・プロセスは並列オブジェクト (スレッド) の集合である。RWC-1 のシステム記述言語である MPC++[9] との互換性、移行性のためにマルチスレッド実行が可能な C++ で記述する。

超流動 OS プロトタイプ 超流動 OS プロトタイプ[1]はプロセス・スケジューラ、GVVM 処理系[2]、管理情報共有機構「MetaShare」を含んでいる。これらも ERSE のプロセスとして実装されている。プロセス・スケジューラはプロセスを PE 空間に割り付ける。次章でプロセス・スケジューラと GVVM の性能評価結果を示す。

PE 空間 1 台から 10000 台程度の仮想プロセッサ (PE) である。プロセスには複数の PE が割り当てられる。プロセス内の各スレッドにはネットワーク・アクセスのための API が提供される。

ネットワーク RWC-1 の CCCB 網をシミュレートする。ネットワークはシステムオブジェクト群として実装されている。ネットワークトポロジは実行時に選択可能である。現在 CCCB、二次元トーラスが使用できる。

[1]RWC-1 には RWCP の SCORE カーネルがのる。

[2]この GVVM 処理系は評価用で実際にプロセスに仮想記憶を提供するわけではない。

オブジェクトスケジューラ 並列オブジェクトの仮想時間スケジューリングにはC++のマルチスレッド・ライブラリであるAwesime[2]を改造して用いている。

実プロセッサ群 ERSEはUnix上の1プロセスである。ERSEは、SunOS4.1.3の場合は1台、Solaris2の場合は複数の実プロセッサを使用してオブジェクトを実行する。

共有メモリ ERSEプロセス用のメモリ。アプリケーションのデバッグや評価に便利なよう、全てのプロセス、オブジェクトからアクセス可能な共有メモリである。

オブジェクト(スレッド)はERSEが提供する仮想時間を基準として動作する。ある仮想時間に動作可能なオブジェクトの集合をイベントセットと呼ぶ。オブジェクトスケジューラはイベントセット内のオブジェクトを実プロセッサに割り付け実行する。イベントセットが空になると仮想時間が進行し次のイベントセットに移行する。オブジェクトが発生する非同期メッセージは網に送られ、通信先のオブジェクトに到着する。メッセージを待っていた受け側オブジェクトは動作を開始する。

ERSEの実行時には上記の構成要素全てを一緒にリンクし、Unixの一つのプロセスとして実行する。現在ERSEは、Sunのシングル及びマルチプロセッサのワークステーションとCRAY CS6400 (Solaris2互換機)上で動作している。1024PE構成のRWC-1をシミュレートするERSEは約200MBのメモリを消費する。我々が使用しているCRAYの場合、実メモリは4GBであるため、装備する32台のプロセッサを使用して16000PE程度までのシミュレーションが可能である。

3.1.2 ERSEのネットワークの構成

ネットワーク中でX-Y平面を構成するバンヤン結合とY-Z平面を構成するキューブ結合を図4に示す。直接網であるため、ルータはPEに内蔵される。ルータは入力4本、出力4本(PEから網へ/網からPEへ、X次元方向の並行リンク、Y次元方向のクロスリンク、Z次元方向のクロスリンク)を有する。Store & Forwardデッドロックを防ぐため、PE用リンク以外の各リンクは3系統の入力バッファを有しリンクを共有している。リンクの使用権の競合はアービタによって解決される。現在の実装ではアービタは先着順アービトレーションを行なう。RWC-1は網からPEへ入るリンクに4種類のプライオリティ付メッセージキューを設けているが、ERSEでは実装の都合上、各スレッドにメッセージキューを設けている。オブジェクトの発生するメッセージは1ワード(ERSEでは32bit)から16ワードの大きさのパケットに分解されて網に投入される。ルータ内ではアービトレーションに1仮想時間かかり、リンクの通過にパケット長×1仮想時間かかる[3]。パケットの転送はCut & Throughであるため、リンクの先のバッファが空の場合はすぐに次のルーティングが始まる。

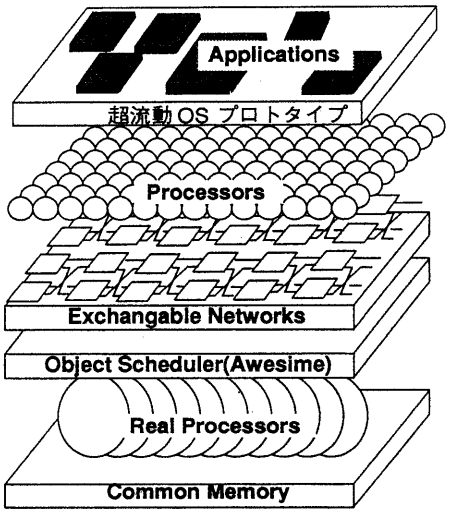

図 3: RWC-1ネットワークレベル・シミュレータERSEの構成

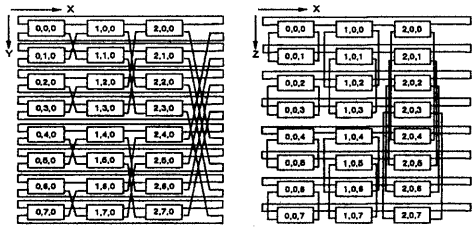

図 4: X-Y平面のバンヤン結合、X-Z平面のキューブ結合

4 プロセス割り付けアルゴリズムとその性能評価

本章ではERSE上に実現したプロセス割り付けアルゴリズムについて説明し、性能評価の結果を示す。

GVVMは複数のプロセスが動作する環境で、メモリを多量に使用するプロセスに、PE空間上で空いているPE領域やメモリ負荷の低いPEのメモリ

[3]RWC-1では1パケットワードが32bitより大きくなる可能性がある。

を提供することによってメモリ負荷の分散を図る。従って、PE空間の空きの割合はどの程度か、即ち、プロセスをRWC-1の空間内にどのように割り当てるかが問題となる。

4.1 RWC-1上でのプロセス割り付けアルゴリズム

到着順にプロセスをPE空間に割り付け、タイムシェアリングは行なわない簡便なプロセス割り付けアルゴリズムを5種類実装し、プロセッサ使用率の測定を行なった。

投入するプロセスの形状は3次元の直方体とする。プロセスに割り付けられたPE領域をパーティションと呼ぶ。「閉パーティション」とはパーティション外部を通らずにメッセージをルーティングできるパーティションである。RWC-1のCCCB網で閉パーティションを作成する場合、パーティションのY次元、Z次元の分割単位は2のべき乗となる(図4)。X次元はPE数が少ないため分割せず、X次元内の全てのPEを1パーティションに割り当てる。従ってパーティショニングはY-Z平面(図2の左側の面である)の2次元空間での分割方法が問題となる。今回は以下のアルゴリズムを実装した。

Quad 2D Buddy Y-Z平面を等面積の矩形に4分割して行く[3]。(図5左)

Binary 2D Buddy without Process Rotation Y-Z 平面を等面積の矩形に2分割して行く。(図5右)

Binary 2D Buddy with Process Rotation Y-Z 平面を等面積の矩形に2分割して行く。(図5右) ただし、プロセスはX次元を軸として90度回転(縦横を交換)してもよいものとする。

プロセスの大きさは必ずしも2のべき乗ではないため、割り当てられたパーティション中に、プロセスによって使用されないプロセッサが存在する。これを「内部フラグメンテーション」と呼ぶ。

また、参考のために2のべき乗にとらわれずに2次元空間をパーティショニングするアルゴリズムも追加する。ただし、RWC-1上では閉パーティションとならない。これらのアルゴリズムでは内部フラグメンテーションは発生しない。

First Fit 最初に発見した投入可能な空間にプロセスを投入する[5]。

Adaptive Scan 最初に発見した投入可能な空間にプロセスを投入する。ただし、プロセスの投入ができなかった場合はプロセスの縦横を交換して再び試みる[1]。

Best Fit フラグメンテーションが最小となる空間にプロセスを投入する[5]。

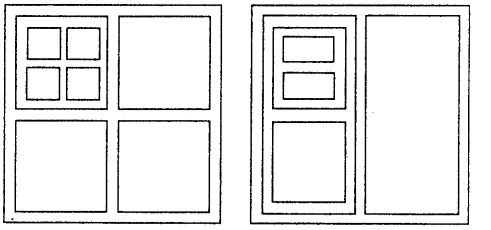

図5: Quad 2D Buddy、Binary 2D Buddy

4.2 プロセス割り付けアルゴリズムの性能評価結果

上記のアルゴリズムを用いてプロセッサ使用率を測定する実験を行なった。実験条件は以下の通り。

- Y-Z平面の大きさは32×32PE。
- プロセスの大きさは最小1×1PE、最大16×16 PEの一様分布。
- プロセス到着時間とサービス時間は指数分布。それらの平均値の比をシステム負荷と呼ぶ。
- プロセス数は200。

図6に結果を示す。横軸はシステム負荷、縦軸はプロセッサ使用率である。プロセッサ使用率は全ての時間、全てのPEにプロセスが割り付けられている場合を100%とする。まず、全般に2のべき乗でパーティショニングを行なうBuddyアルゴリズムは、2のべき乗にとらわれないアルゴリズムと比較して性能が低い。Buddyアルゴリズムの中では、平面を2分木に分割して行く方が4分木に分割するよりも性能が良く、更にプロセスを回転させる方が良い。

そこで、GVVMの評価には閉パーティションを実現可能なBuddyアルゴリズムの中で最も性能の良い"Binary 2D Buddy with Task Rotation"を用いる[4]。このアルゴリズムのプロセッサ使用率は45%ほどである。従ってPE空間の半分程度は空いていることになる。

5 GVVMの性能評価

本章ではERSE上に実現したGVVMについて説明し、性能評価の結果を示す。性能評価の目的は、RWC-1上において、GVVMを用いるとGVVMを用いない場合と比べてどのくらいの性能向上があるかを調べることである。

[4]ただし、プロセスを回転することにより性能低下が発生しうる。例えば、バンヤン結合を仮定して高度に最適化してあるプロセスは回転してキューブ結合に割り当てられると性能が低下する。

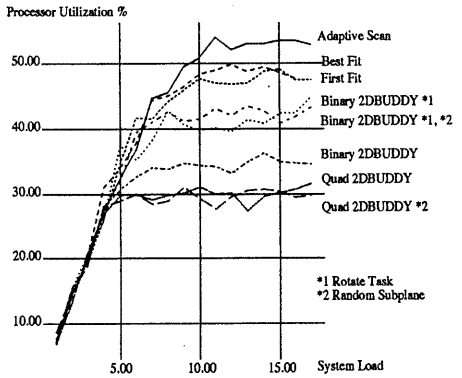

図 6: プロセッサ使用率

5.1 負荷分散戦略

GVVM 処理系は上記の PE 空間の空きを発見し、そこへ不要なページを仮想ページアウトする。仮想ページアウトされる PE をスワップ先 PE と呼ぶ。空きを発見するメモリ負荷分散戦略として、以下の 3 つを用いた。

ランダム探索法 ある PE でページアウトが必要となりスワップ先 PE を決める際、その PE は乱数によって選んだ一定数の PE に対してメモリ負荷の比較を行なうための入札を行なう。入札の結果、より低いメモリ負荷の PE が複数存在した場合、最短距離の PE を選択し、スワップ先 PE とする。

MetaShare 探索法 超流動 OS の管理情報共有機構である「MetaShare[7]」の機能を使用する。MetaShare はシステム中でプロセッサ負荷、メモリ負荷等の管理情報を管理するサブシステムである。 MetaShare は GVVM からの「メモリ負荷の小さい PE はどこにあるか」という Query に対し、いくつかの候補を提示する。

DVM 探索法 仮想ページアウトはネットワークで直接隣接する PE のうち最もメモリ負荷の低い PE へ行なう。1 ホップずつの転送を繰り返すことにより、よりページ負荷の低い PE 領域への負荷拡散が発生することを期待している [4]。

5.2 管理情報共有機構「MetaShare」

メモリ負荷の低いスワップ先 PE を探す際、その PE は仮想ページアウトを行なう PE に近いことが望ましい。一方、近くによい場所がない場合、より大局的観点から PE 空間内の空き領域を探すことも必要である。また、スワップ先 PE を探しているプロセス群の間で空き領域の使用区分を調整する事も必要である。

MetaShare は各 PE に存在する GVVM オブジェクトが発生する Query に対し低メモリ負荷の PE 群

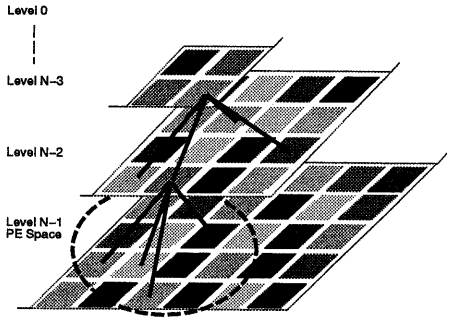

図 7: MetaShare の分散データ構造

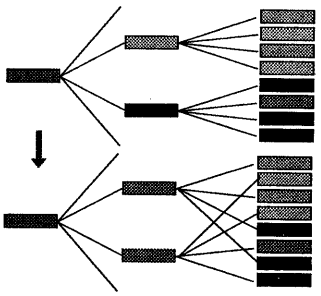

図 8: 木の再構成

を提示する。図 7に MetaShare が PE 空間上に構成する分散データ構造を示す。図中、最下層 [Level n-1] が PE 空間の Y-Z 平面に相当する。最下層の 4 つの領域 (PE) のメモリ負荷値を抽象化 (平均) し次の層 [Level n-2] の 1 領域 (木のノードに相当) を形成する。この操作を繰り返し、ルートに至る木を作る。ルートは領域をひとつ含み、それは PE 空間全体のメモリ負荷の平均を表している。

MetaShare オブジェクトは Query に対し木の最下層から木を辿り、低メモリ負荷の PE を探索する。PE[x] から木を遡り [Level i] 層に達しているとする。[Level i] 層の領域は [Level i+1] 層の 4 つの子の領域を含んでいる。そのうちのひとつは PE[x] を含む領域である。残り 3 つの領域に PE[x] を含む領域よりメモリ負荷値の低い領域があれば、木を下ってその領域の中を探索する。なければ、[Level i-1] 層へ上がる。

木は PE 空間のメモリ負荷値の変動を反映するよう定期的に再構成される。その際、ある階層のある領域は自分の含む子の領域のメモリ負荷値が平均化されるよう、孫の領域の子の領域への所属を組み替える (図 8)。即ち、メモリ負荷が高い領域と低い領域が組を作る。また、領域には直径が決められており、ネットワークトポロジ上遠くはなれた領域が同一の親領域を持つことはない。これにより、メモリ負荷の高い領域は、組になっている低負荷領域を近

くに低いコストで発見することが可能である。

5.3 GVVM の性能評価結果

各負荷分散戦略の空き領域を見つける能力を比較するため、プロセスはメモリを多量に使用するものとする。そこで、評価の条件としては厳しい条件であるが、投入するプロセスの各 PE の仮想空間の大きさは実空間の大きさの 2 倍とし、全てのプロセスをメモリ飢餓状態に設定した。PE 上のオブジェクトは仮想空間を正規分布に従ってアクセスする。その他の条件は以下の通り。

- ページアクセスの書き込み率は 30% とする。
- ディスクは 8PE のクラスタ単位で配置される。RWC-1 の I/O ネットワークの仕様は定まっていないため、ディスクは CCCB 網を介してアクセスする。ネットワークとディスクの転送速度比は 25:1。ネットワークの転送速度をリンクあたり 400MB/秒[5]、ディスク 1 台の転送速度は 16MB/秒とする。バッファによりディスクへの書き込みとネットワークからのデータ転送はオーバーラップする。
- 投入するプロセスの形状は直方体とする。X 次元の大きさは 4PE で固定する。Y,Z 次元の大きさは、システムの辺長を n とすると $U(1, n \times 0.5)$ の一様分布とする。例えば、$X = 4, Y = 16, Z = 16$ $(N = 1024)$ の構成の場合、プロセスは最小で 4x1x1 = 4PE の大きさ、最大で 4x8x8 = 256PE の大きさとなる。
- プロセスは 40 個を予め生成しておき、Binary 2D Buddy プロセス回転付で割り当てる。全てのプロセスが終了するまでの時間を実行時間とする。実行時間には MetaShare のコスト、仮想ページングのコスト、実ページングのコスト等の I/O コストは入るが、CPU コストは入っていない。
- システムを構成する各次元の PE 数は以下の通り。

PE 数	X	Y	Z
256	4	8	8
512	4	16	8
1024	4	16	16

1024PE システムを対象として、GVVM 未使用の場合の実行時間を 1 とした時の実行時間の比較を図 9に示す。負荷拡散型の DVM はページを 1PE づつ移動する必要があるため、よい性能が得られないのに対し、空き領域に直接仮想ページアウトを行なうランダム探索法や MetaShare 探索法はよい性能を示している。

[5] ネットワークのリンクの幅を 32bit と仮定。48bit 幅であれば、600MB/秒となる。

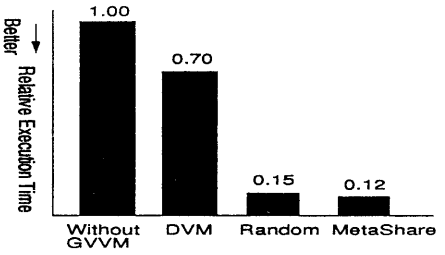

図 9: 1024PE 時の相対処理時間

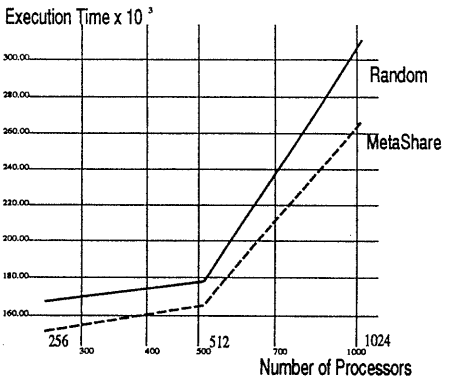

図 10: PE 数に対する実行時間

PE 数を変化させたランダム探索法と MetaShare 探索法の性能比較結果を図 10に示す。横軸は PE 数(対数表示)、縦軸は実行時間である。これら二つの負荷分散戦略の性能の差は、発見した空き領域の近さの差が一因である。その様子を図 11(ランダム探索法) と図 12(MetaShare 探索法) に示す。図中の直方体は各 PE のメモリ使用量を表している。まず、ランダム探索法の図 11を見て頂きたい。大きな直方体が並んでいるプロセスの割り当て状況が見てとれる(この中には 8 つのプロセスが存在する)。そして、空き領域に仮想ページアウトにより分散されたメモリが散在している。一方、図 12の MetaShare 探索法では、プロセスの形状ははっきりしていない。これは各プロセスは、まず、自らに最も近い空き領域である内部フラグメンテーションにメモリ負荷を分散してゆくため、プロセスの回りにメモリ負荷が染み出すためである。自分の近くに空きがなくなったら、より遠くの、プロセスの割り付けられていない空き領域を使用する。

6 おわりに

以上、大域的仮想仮想記憶 (GVVM) の基本概念、GVVM を RWC-1 上で評価するために作成した RWC-

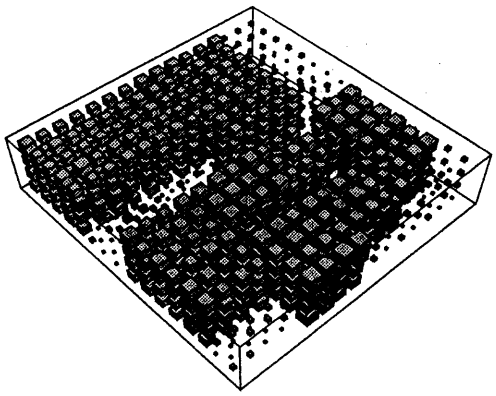

図 11: ランダム探索法の負荷分散の様子

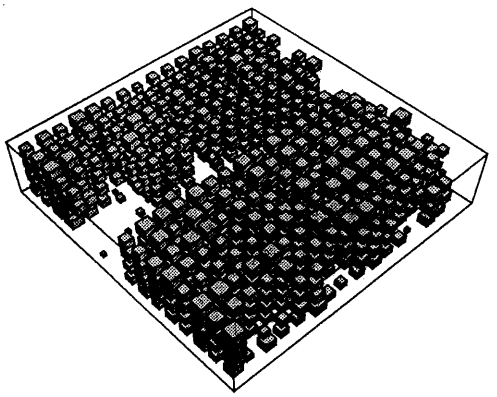

図 12: MetaShare 探索法の負荷分散の様子

1 のネットワークレベル・シミュレータ ERSE、ERSE を用いて行なったプロセス割り付けアルゴリズムと GVVM の評価の結果について述べた。

評価の条件としてページングが多発する厳しい条件を用いたにもかかわらず、1024PE 構成の RWC-1 上で GVVM を用いた場合、GVVM を用いなかった場合と比較して、約 8 倍の性能が得られることがわかった。また、超流動 OS の管理情報共有機構 MetaShare が負荷分散に有効な情報を提供できることがわかった。

MetaShare には情報収集間隔など各種パラメータを与える必要がある。これを、マシンの状況に合わせて自動的に決定することが今後の課題である。また、タイムシェアリングを行なう環境での GVVM の評価を行ないたい。

謝辞

本研究の一部は RWC 計画の一環として「超並列システムアーキテクチャに関する研究」で行なわれたものである。関係各位に感謝する。特に、CCCB 網に関してご教示頂くと共に、図 2の引用をご許可下さった RWCP の横田氏に深謝する。

参考文献

[1] J. Dign and L. Bhuyan. An Adaptive Submesh Allocation Strategy for Two-Dimensional Mesh Connected Systems. *Proc. of International Conference on Parallel Processing*, pp. 193–200, 1993.

[2] D. Grunwald. A Users Guide to Awesime: An Object Oriented Parallel Programming and Simulation System. *Univ. of Colorado Boulder Tech. Report CU-CS-552-91*, 1991.

[3] K. Li and K. Cheng. A Two-Dimensional Buddy System for Dynamic Resource Allocation in a Partitionable Mesh Connected System. *J. of Parallel and Distributed Computing*, pp. 79–83, 12 1991.

[4] M. Malkawi, D. Knox, and M. Abaza. Dynamic Page Distribution in Distributed Virtual Memory Systems. *Proc. of 4th ISMM Int. Conf. on Parallel and Distributed Computing and Systems*, pp. 87–91, 1991.

[5] Y. Zhu. Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers. *J. of Parallel and Distributed Computing*, pp. 328–337, 16 1992.

[6] 横田隆史, 松岡浩司, 岡本一晃, 廣野英雄, 坂井修一. 超並列向け相互結合網 MDCE の提案と評価. *JSPP'94*, pp. 121–128, 1994.

[7] 田沼均, 平野聡, 一杉裕志, 須崎有康, 塚本享治. 管理情報共有機構 (MetaShare) を大域的仮想仮想記憶で利 用した場合の基本性能評価. 情報処理学会研究会報告 *93-OS-65(SWoPP'94)*, 1994.

[8] 坂井修一, 岡本一晃, 横田隆史ほか. 超並列計算機 RWC-1 の基本構想. *JSPP'93*, pp. 87–94, 1993.

[9] 石川裕, 堀敦司ほか. 並列プログラミング言語 MPC++ の実現. *JSPP'94*, pp. 105–112, 1994.

[10] 平野聡, 一杉裕志, 田沼均, 須崎有康. 超流動 OS の大域的仮想仮想記憶におけるページ探索法の比較. 情報処理学会研究報告 *93-OS-61 (SWoPP'93)*, pp. 65–72, 1993.

[11] 平野聡, 一杉裕志, 田沼均, 須崎有康, 塚本享治. 大域的仮想仮想記憶 (GVVM) のマルチプロセス環境での評価. *JSPP'94*, pp. 365–372, 1994.

[12] 平野聡, 田沼均, 須崎有康, 濱崎陽一, 塚本享治. 超並列システム用オペレーティングシステム「超流動 OS」の構想. 情報処理学会研究報告 *93-OS-58*, pp. 17–24, 1993.