市場メカニズムに基づく計算資源割り当て方式

松本 尚

■研究のねらい

単体の計算機の能力を越えた性能を安価に実現するために、複数の計算機を一つの大きなシステムとして使用する計算機クラスタシステムが実用化されつつある。計算機クラスタシステムは図1のように、オフィスや研究所にある計算機を安価な標準品のネットワークによって接続して構成される。このクラスタシステムの上で効率良くかつ公平に仕事を行うことを可能にする計算資源の割り当て方式を構築することが本研究の目標である。つまり、あるマシンはいくつもの仕事を抱えているのに一部のマシンにはまったく仕事がない状態や、あるマシンはメモリが足りなくてディスクアクセスが頻繁に起こっているのに他のマシンは搭載されたメモリの一部しか使用されていないといった状態を回避する資源割り当て方式を研究開発する。

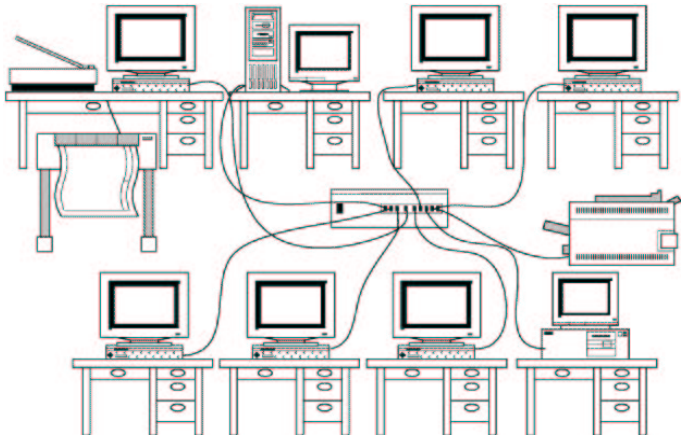

図 1: クラスタシステム

従来のシステムではある特定のマシンに中央集権的な資源配分管理プログラム（中央集権スケジューラ）が存在し、そのスケジューラが各仕事への資源配分をすべて調停していた。中央集権的な資源割り当て方式では各仕事の計算資源に対する要求にきめ細かに対応することが難しく、すべての仕事をこのスケジューラ経由で計算機に割り当てるため会話的な環境実現に向いていない。この問題点の解決のために、本研究では公平性を侵さない範囲で、アプリケーション（仕事）自らが自分の資源割り当てを決定することが可能となる自由主義的な資源割り当て方式を提案し実装することを目標とした。

目標とする方式として「自由主義」的な資源割り当て方式という言葉を使用したが、理想の市場主義経済体制がそうであるように、決して無政府状態の自由放任方式を意味しているわけではない。ある種の公平性は保証しなくてはならず、自己責任でスケジューリングの判断を下すためには判断を下すのに十分な情報が低コストで入手可能でなくてはならない。結局、本研究では、「低コストでアクセスできる情報開示機構」を持ち、「公平性を守るための基本ルールを確立」した自由主義的な資源割り当て方式の研究開発に取り組んだ。詳細については研究成果において述べる。

■研究成果

本研究の主要な成果は、**自由市場原理に基づくスケジューリング方式（FMM 方式**: Free Market Mechanism 方式）の提案と、 FMM 方式を実装した汎用オペレーティングシステムの実現である。また、 FMM 方式と共に効率良く使用可能な入出力装置の実現方式に関する発明を副産物として行った。

1. 自由市場原理に基づくスケジューリング方式

さきがけ 21 の研究期間を通して FMM 方式の理解が進み、従来の中央集権的な資源割り当て方式との違いが明らかになった。新しい理解に基づいて、本研究で提案した FMM 方式の特徴を以下にまとめる。

1. システム提供の中央集権スケジューラを持たない
   (a) マシン単位の資源割り当て
   (b) 全体的な資源調停機能はユーザレベルで実現
2. 公平性維持／性能低下防止のための基本ルール
   (a) マシン単位で資源割り当ての公平性を調節
   (b) すべての資源競合を資源割り当てに考慮
3. ユーザアプリケーションによる低コストの状況判断
   (a) 低コストでアクセスできる情報開示機構

以下、それぞれの項目について従来の中央集権的な資源割り当て方式と対比しながら FMM 方式の特徴を明らかにする。

1.1. システム提供の中央集権スケジューラの廃止

クラスタシステムにおいて、長期的な負荷分散を考慮した時分割による資源割り当てを行おうとする場合、システム全体を統括する中央集権スケジューラを用意することが一般的である。中央集権スケジューラはシステム全体の資源の使用状況を把握し、資源を要求する複数のユーザアプリケーションを調停し、実資源を配分する。

単体マシンのスケジューラと比べて、クラスタシステム用の中央集権スケジューラは資源割り当ての最適化に関して考慮すべき要因が大幅に増えている。これらの要因を以下に列挙する。

- 各マシンのプロセッサ負荷状況
- 各マシンのメモリ負荷状況
- 各マシンのネットワーク負荷状況
- 各マシンの二次記憶アクセス負荷状況
- 関連する仕事の配置（通信コスト）
- 使用中の実メモリの配置（マイグレーションコスト）
- 関連する仕事の実行時間の同期
- マシン間共有資源の競合解消

上記の要因におけるどの項目をどの程度重視する必要があるかは、それぞれのユーザアプリケーションによって事情が異なる。

中央集権スケジューラをシステムが提供し、最適化も十分に行おうとする場合は、複数の要因と個別の事情を反映して資源を割り当てるアルゴリズムを構築する必要がある。このアルゴリズム作成自体が非常に難しく、個別のアプリケーションから見れば提供されるアルゴリズムは必ずしも最適とは限らない。また、システム提供の中央集権スケジューラは様々の要因に関する個別のアプリケーションの事情を伝達可能なインタフェースを設定する必要がある。このインタフェースを変更するとアプリケーションプログラムにまで影響がでるため、インタフェースは一度設定すると容易には変更できない。このため、考慮すべき新たな要因が判明しても、システム提供の中央集権スケジューラに反映するのは難しい。

FMM 方式では発想を転換して、システム全体にわたる資源調停の役目をユーザアプリケーション自身が担うことにしている。ただし、資源の保護と仮想化は汎用システムとして存続しなければならないため、マシン単位のシステム提供のスケジューラは残る。どのマシンでどの時点で走行したいかについてユーザアプリケーションが自分で判断し、システムに対してリクエストを出すことにする。マシン内での資源競

合は調停されなければならないため、走行時刻の要求は必ずしも満たされるとは限らない。しかし、システム提供の中央集権スケジューラと同じ状況判断をすべてのアプリケーションが行えば、中央集権スケジューラと同レベルの資源割り当てが可能である。逆に言えば、スケジューリングに自由競争原理を持ち込んで、各アプリケーションがお互いに自分の走行が有利になるように切磋琢磨してもらおうという考え方がFMM方式の基本である。

1.2. 公平性を守るための基本ルール

各ユーザアプリケーションが自分の走行場所や走行時刻を指定して自由競争を行う場合に、資源割り当ての公平性を守るための基本ルールが不可欠である。なぜなら、厚かましくよりたくさんのマシンで走行することを指定すればするほど有利になるような自由放任状態では、正常にシステムを運営できない。多くのアプリケーションが厚かましく振舞えば、システムの資源供給能力が飽和してしまいシステムの全体性能が低下してしまう。多くのアプリケーションはつつましく適度に資源要求したとしても、一部に厚かましいアプリケーションが存在すれば、つつましいアプリケーションはなかなか資源の割り当てを受けられないことになってしまう。

FMM方式ではマシン単位で公平性を守る機構を導入することによって基本ルールを設定している。1台のマシンしか使用しないアプリケーションは100台のマシンを使用するアプリケーションよりも優先して実行されるべきであるという考え方も可能であるが、FMM方式ではこの立場には立たず、空いているマシンがあれば有効活用すべきという方針にたっている。不幸にも資源要求が競合してしまったマシンでのみ公平性が保たれればよいという考え方に基づいて、マシン単位で公平性を管理している。このマシン単位での公平性維持という方針のおかげで、システム提供の中央集権スケジューラが完全に不要となっている。

最近のシステムでは、ノンブロッキングI/Oがユーザに提供され、低オーバヘッドで通信や外部記憶アクセスが可能となっているものがある。このノンブロッキングI/Oを活用したアプリケーションでは性能がプロセッサ能力ではなくI/O能力で制約されるものが少なくない。このため、FMM方式ではプロセッサ資源の割り当ての公平性のみではなく、I/O資源割り当ての公平性をマシン単位のスケジューリングにおいて考慮する。同時に、I/O資源の資源競合を考慮することにより、I/O資源の一部の能力が飽和することによりシステム全体の性能が大幅に低下することを防止する。

1.3. ユーザアプリケーションによる低コストの状況判断

各アプリケーションが自己責任で資源割り当ての判断を下すためには判断を下すのに十分な情報が低コストで入手可能でなくてはならない。

実行時に動的に負荷分散を自らの判断で行うためには、自分が現在走行しているマシンの負荷情報のみではなく、他のマシンの負荷情報も必要である。また、システムが非均質であれば、マイグレーション可能なマシンを知るために各マシンのプロセッサの種類といった情報も必要になる。これらの情報を獲得するために、各アプリケーションが毎回他マシンに通信を行うのではコストが大き過ぎる。他マシンも含めた資源の使用状況や割り当て情報がユーザアプリケーションから低コストでアクセスできる情報開示機構が必要である。この情報開示機構として筆者が*SSS–CORE*で提案した情報開示機構を使用する。この情報開示機構において、カーネルが管理している資源量、資源の種類、資源の機能、資源割り当て情報や資源使用状況といった情報を格納しているメモリ領域がユーザ空間にread-onlyでマップされており、これらの情報がユーザから低コストで参照できる。そして、情報開示機構は自分のマシンの情報のみではなく、他マシンの情報を適宜交換し、ユーザアプリケーションに対して公開している。

2. FMM方式の実装

2.1. *SSS–CORE* の ULTRA 60 への移植

本研究より以前に筆者が開発した汎用スケーラブルオペレーティングシステム*SSS–CORE* Ver.1.x（図2）はSun Microsystems社のSPARCstation 20をEthernetまたはFast Ethernetで接続したクラスタ

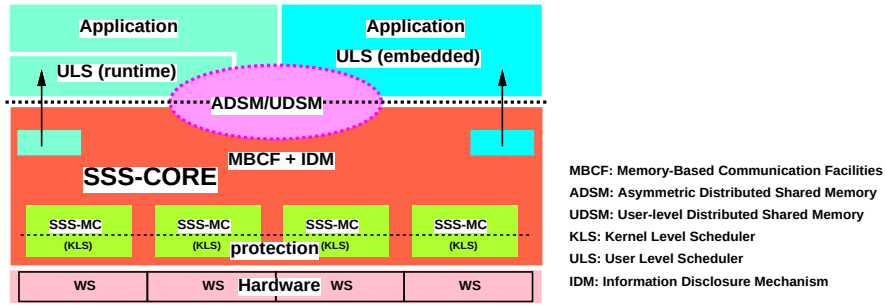

図 2: *SSS–CORE* の構成

環境で動作する。 SPARCstation 20 は平成 7 年度に発売されたワークステーションであり、プロセッサは SuperSPARC である。現在、 Sun Microsystems 社の最新鋭ワークステーションに搭載されているプロセッサは UltraSPARC-II であり、単体性能は SuperSPARC の数倍高速である。本研究においてクラスタ構築用に購入したマシンは UltraSPARC-II プロセッサを搭載した ULTRA 60 ワークステーションである。このため、 *SSS–CORE* を ULTRA 60 へ移植する必要があった。

UltraSPARC はユーザレベルのコードに関して、 SuperSPARC と互換性があるものの、カーネルレベル（Supervisor mode）のアーキテクチャと命令体系は大幅に改良されている。このため、プロセッサ依存のコードを大幅に変更する必要がある。また、 UltraSPARC が使用された Ultra シリーズのワークステーションは Boot ROM が Open BOOT ver.3.x になっており、 SPARCstation 20 の Open BOOT ver.2.x と大幅に異なっている。さらに、 ULTRA 60 は I/O 用の拡張バスも SBus から PCI バスに変更されており、 I/O 周りのコードを変更する必要もあった。これらの事由により移植作業は非常に困難であったが、平成 12 年度上半期には Ultra 版 *SSS–CORE* Ver.2.2 を ULTRA 60 で動かすことに成功した。

2.2. マイグレーション機能の開発

FMM 方式を *SSS–CORE* において実現するためにマイグレーション機能を実装した。マイグレーション機能とはあるマシン上で走行中のプログラムを他のマシンに移送して実行を継続させる機能である。 FMM 方式の場合、アプリケーションが主体となって実行場所を指定するため、 *SSS–CORE* に新たにマイグレーション要求用のシステムコールを実装した。ただし、今回実装したマイグレーション機構にはプログラムの中断に関して何も制約がないため、システム側からの強制的なマイグレーションにも対応できる。

SSS-CORE の最大の特徴である MBCF は通信相手がマシンを含めて仮想化されており、 MBCF によって通信同期を行っているプログラムはそのまますべてマイグレーション可能である。マイグレーションされるプログラムの新しいマシンにおける再起動に平成 10 年度に実装した遠隔実行機構を利用することにより、移送されたプログラムは標準入出力を継続的に使用可能である。マイグレーションされたプログラムは遠隔実行機構の標準出力機構を使って、標準出力を起動された端末画面またはコンソール画面に出力する。標準入力も同様に起動された端末キーボードまたはコンソールキーボードから移送後も入力を続けることが可能である。

2.3. 公平性維持用エイジング方式の実装

FMM 方式では、先に述べたように、競合する資源要求を発行したプログラムにペナルティを課すことでマシン単位で公平性を維持する。 *SSS–CORE* 内で資源割り当てが制御されている資源はプロセッサのみである。一定期間毎に最も高い優先度のプログラムがマシンのプロセッサを割り当てられる。プロセッサ以外の資源に関しても、プロセッサを割り当てない限り、新たな使用要求を出すことは不可能であるため、プロセッサの割り当てを制御することにより使用頻度を制御できる。

SSS–CORE のマシン内のプロセッサ割り当ては割り当て時刻をベースにした優先度を使用している。この方式は計算量を増やさずに優先度の発散を防止することができる。 FMM 方式ではこの方式をベースに、使用要求が飽和した資源へのさらなる要求に対するペナルティを加味することで公平な資源割り当て

を実現する。どの飽和資源に対する要求に対して、どの程度のペナルティを課するかという問題は、どういう公平性を実現したいかと言うシステムの全体の目的と関わる問題であり、システム管理者の可変パラメータとして実装されている。*SSS–CORE* の現在の FMM の実装ではプロセッサ資源とネットワーク資源の割り当ての公平性にのみ対応している。FMM 方式の枠組において、メモリ資源の割り当ての公平性に対応するためには、メモリの二次記憶へのスワッピング機能を *SSS–CORE* に実装する必要がある。

2.4. ユーザレベルの FMM 用基本戦略

ユーザアプリケーション自らが自分が走行するマシンをマイグレーション機能によって指定するため、ユーザレベルで実行マシンを選定する処理を行う必要がある。もしも、複数のプログラムが負荷が軽い同一のマシンに同時に移動した場合には、かえって負荷がアンバランスになる可能性がある。そこで、ethernet の CSMA/CD プロトコルからヒントを得た以下のアルゴリズムに従って、移動先を選定する。

1. 適当な粒度（数百 msec）で本来の仕事を実行
2. 情報開示機構に問い合わせて一番軽いマシン M1 を調べる
3. M1 と自マシンの負荷を比較し、移ることが得がどうか判断
4. 非移動が得である場合は 1. に戻る
5. サイコロを振り、移送猶予期間 T1 をランダムに決定
6. さらに T1 の時間を過ぎるまでの間、本来の仕事を実行
7. 情報開示機構に問い合わせて一番軽いマシン M2 を調べる
8. M2 と自マシンの負荷を比較し、移ることが得がどうか判断
9. 非移動が得である場合は 1. に戻る
10. M1 と M2 が一致の場合、移送システムコール発行、戻ったら 1. へ
11. M1 と M2 が不一致の場合、M2 を新たな M1 として 5. へ

なお、*SSS–CORE* の情報開示機構は 100msec 毎にマシン間の情報を交換している。負荷の軽いマシンを得るための計算方法はアプリケーション依存であり、CPU 負荷、ネットワーク負荷、メモリ負荷の値を組み合わせて計算した値を比較に使用する。移送猶予期間の T1 は情報開示機構の更新間隔と処理に関わるプログラム数（並列度）の積の数倍に、0 から 1 の間の乱数を掛けた時間。

このアルゴリズムにより、同一の低負荷のマシンに移動する確率を低くして、負荷分散が可能になる。また、情報開示機構から得られる負荷情報を使って計算するマシン単位の負荷の計算式を、アプリケーションに応じて調整することによりきめ細かな負荷調整が可能となる。

2.5. FMM 方式による資源割り当ての実施例

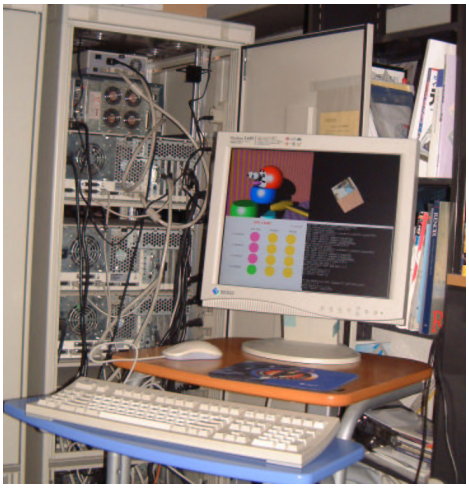

図 3: *SSS–CORE* さきがけクラスタ

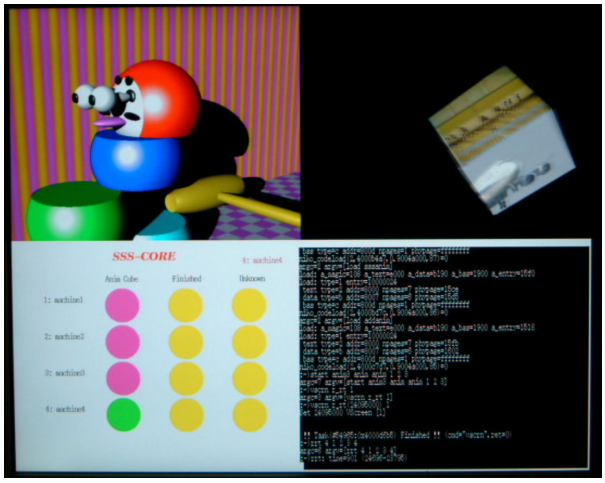

図 4: animcube デモンストレーション

図 3 に本研究で用いた ULTRA 60 によるワークステーションクラスタシステムを示す。本実施例では ULTRA 60 マシン 4 台を Gigabit ethernet によって接続している。図 4 に animcube と呼ばれるデモプログラムを実行中のコンソール画面を示す。デモ画像は右上のスクリーンに表示され、左下のスクリーンはシステムモニタを示す。図 4 は 4 番マシンのコンソール画面であり、システムモニタの一番左の列に animcube の実施状態が示される。システムモニタの各行は各マシンに対応し、丸印の色によって対応するマシンにそのアプリケーションに関連する仕事がいくつ割り当てられているかを示している。黄色の丸は 0 個、緑の丸は 1 個、青紫の丸は 2 個、ピンクの丸は 3 個、赤の丸は 4 個以上の仕事があることを示している。 animcube は立方体の表面に動画像を表示するデモプログラムであり、その立方体が自転ならびに公転軌道によって動き回る。 1 番マシン、 2 番マシン、 3 番マシンの各マシンにおいて動画像再生の 2 個の仕事（立方体の表裏の二面に対応）とその再生画像を透視変換する 1 個の仕事の計 3 個の仕事（ピンクの丸印に対応）を行っていて、 4 番マシンでは他のマシンから送られてきた計算結果を 1 画面ずつ同期を取って右上のスクリーンに表示する 1 個の仕事（緑の丸印に対応）が実行されている。

図 5: 8 並列レイトレ起動時

図 6: 仕事 1 個移送後

animcube のデモが 4 台のマシンで並列実行されている状態で、さらに並列レイトレーシングプログラム（以下レイトレ）を 8 個の仕事による並列処理（8 並列）によって起動する（表示マシンは 4 番マシン）。このレイトレプログラムは複数の 3 次元モデルを繰り返し計算して表示し続けるデモプログラムであり、FMM 方式に基づくマイグレーションコードがユーザレベルで埋め込まれている。ただし、初期状態としてすべての仕事を 4 番マシンで走行させる。図 5 はこのレイトレを起動直後の 4 番マシンのスクリーンである。レイトレ画像は左上に表示されている。レイトレに関する仕事の配置状態はシステムモニタの真中の列に表示され、 8 個の仕事が動き出した 4 番マシンにが赤丸（4 個以上の仕事を持つ状態）で表示されている。レイトレの起動によって、 4 番マシンに負荷が偏った状態になるため、比較的空いているマシンを情報開示機構によって見つけて、 1 個の仕事が自らを 3 番マシンに移送した直後を示すのが図 6 の画面である。レイトレ画像で突出して計算が先行している何本かのラインが 3 番マシンにおけるレイトレ計算に対応する。

図 7 はさらに比較的空いている 2 番マシンを見つけて、 1 個の仕事が自らを 2 番マシンに移送した直後の画面である。このようにして仕事（プログラム）が自ら空いているマシンを見つけて移動して、最終的には各マシンの仕事量が均等化する。この実施例では animcube も時分割による並行動作で動き続けており、レイトレの 8 個の仕事は 1 番マシンに 2 個（青紫）、 2 番マシンに 2 個（青紫）、 3 番マシンに 1 個（緑）が移動し、 4 番マシンに 3 個の仕事（ピンク）が残った時点で、負荷が均衡して安定状態に達した（図 8）。 animcube の仕事も含めた各マシンの仕事量は、 1 番マシンが 5 個、 2 番マシンが 5 個、 3 番マシンが 4 個、 4 番マシンが 4 個であり、均衡している。

animcube の実行を停止し、全仕事を終了させたところ、 4 番マシンから 3 番マシンにレイトレの仕事が 1 個移動して、レイトレの仕事は各マシン 2 個（青紫）ずつで完全に均衡化された（図 9）。ここで 2 番

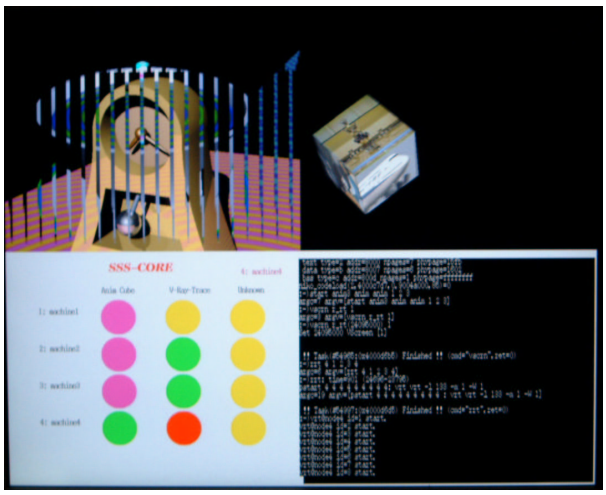

図 7: 仕事 2 個移送後

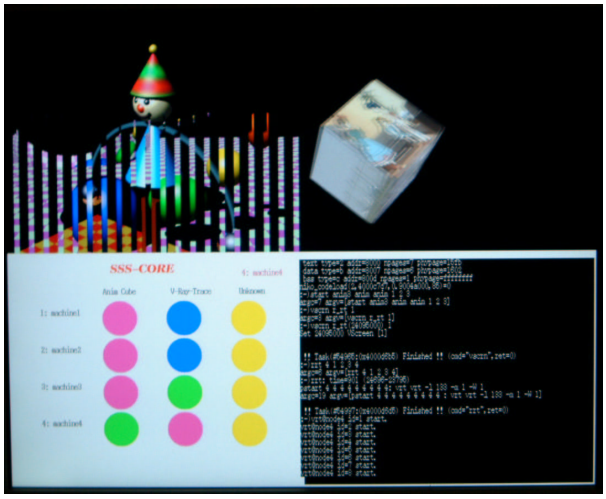

図 8: 仕事 5 個移送後（安定状態）

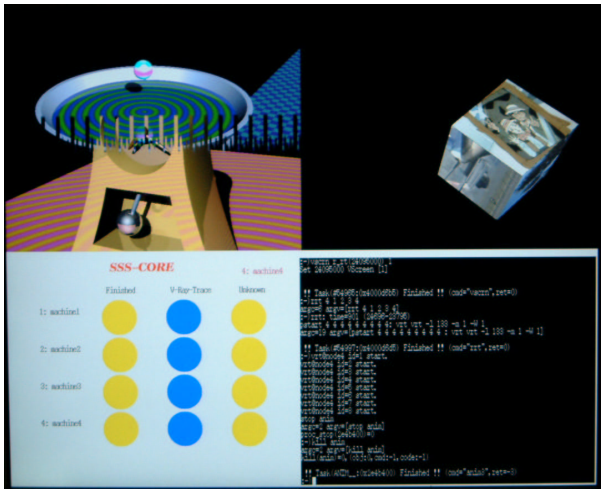

図 9: animcube 停止後（安定状態）

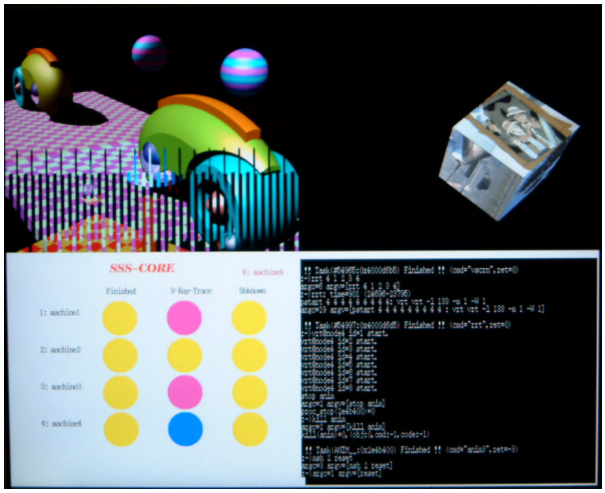

図 10: マシン No.2 停止後（安定状態）

マシンに対して停止コマンドを発行して、 FMM 方式の枠組内で 2 番マシンにおいて走行中の仕事に緊急退避させる。避難後安定状態に達した画面を図 10 は示している。これは停止前にもうすぐ停止するという情報を情報開示機構に書くことによって、アプリケーションを停止することなく一部マシンのシャットダウンが可能になることを示している。逆に、新たにマシンをクラスタに追加してオペレーティングシステムを立ち上げれば、自然に起動されたマシンにも負荷が分散して行き、システム全体の処理能力を向上させることが可能である。

3. マルチメディア対応高速通信カードの研究開発

現在広く使用されているインターネット上の通信方式（TCP/IP プロトコル）の実装は処理オーバヘッドが大きいため、将来より大きなバンド幅のネットワークが利用可能になっても、このオーバヘッドによりユーザがバンド幅を活用できない可能性がある。クラスタコンピューティングによってシステムの性能向上を図る場合に、このオーバヘッドが処理全体のボトルネックとなり性能向上効果が得られない。さらに、この通信オーバヘッドの問題は、セキュリティを保つための暗号プロトコルを通信に使用する場合には、暗号化／復号化の処理コストが非常に大きいため、より深刻な問題になる。また、動画像データや音声データの連続メディアストリームを扱うプログラムに対して FMM 方式の資源割り当てを実際に適用するために、低コストで連続メディアストリームが扱える環境が必要であった。これらの理由から本研究内においてマルチメディア対応高速通信カードの研究開発を行った。

3.1. ユーザレベル I/O アクセス方式

入出力装置をユーザアプリケーションから使用する場合に、通常はオペレーティングシステムを介して使用する。これは故意もしくは過失に関わらず、ユーザが不当な使用方法によって、他のユーザのデータを壊したり、覗き見たり、自分のシステムや他人のシステムに迷惑をかけたりさせないためである。しかし、オペレーティングシステムを介して入出力装置にアクセスを行うと少なからずオーバヘッドが発生する。このオーバヘッドを無くすためには、ユーザアプリケーションから直接かつ安全に入出力装置にアクセスしたい。この目的を実現するのが、 Memory-based Virtual Interface (MVI) と命名したユーザレベル I/O アクセス方式である。どのユーザアプリケーションがアクセスしたかを入出力装置が識別可能であれば、保護や使用制限をかけることが可能である。このため、入出力装置のレジスタをアクセスする物理アドレスの一部（厳密にはページフレーム部分の一部）に、アプリケーションを識別する識別子を埋め込み区別する。この方式は非常に汎用性が高く、多くの高性能入出力装置に適用可能である。なお、本発明は日本ならびに米国に特許出願されている。

3.2. マルチメディア対応高速通信カードの実装

通信の処理オーバヘッドを大幅に削減するために、新しいネットワークインタフェースアーキテクチャを研究開発する必要がある。そこで、前記のユーザレベル I/O アクセス方式の採用とメモリベース通信の概念を導入することにより、処理コストの低い通信ネットワークインタフェースアーキテクチャを設計し、このアーキテクチャに基づく高速通信カードを開発した。 FMM 方式の下で、複数のユーザプログラムが効率良くかつ公平に本カードを同時使用することが可能である。本カードは、各種通信プロトコルをカード内蔵プロセッサによって処理し、データ転送、データ暗号化／復号化、マルチメディアストリームのデコードを専用内蔵ハードウェアによって実現する。よって、本カードが使用されるマシンのメインプロセッサは通信処理や通信に付随するデータ転送やデータ変換の処理を大幅に免除され、本来のアプリケーションプログラムの実行に専念できる。内蔵プロセッサのファームウェアによって、広く使われている TCP/IP および UDP/IP をサポートする。暗号プロトコルに関しては現在広まりつつあり、ネットワークインタフェースにおいて独立に処理可能な IPSec をサポートする。そして、クラスタ計算用およびソフトウェア分散共有メモリ実現のための通信方式として、 *SSS–CORE* において高性能と汎用性が実証されているメモリベース通信ファシリティ（MBCF）もサポートされる。

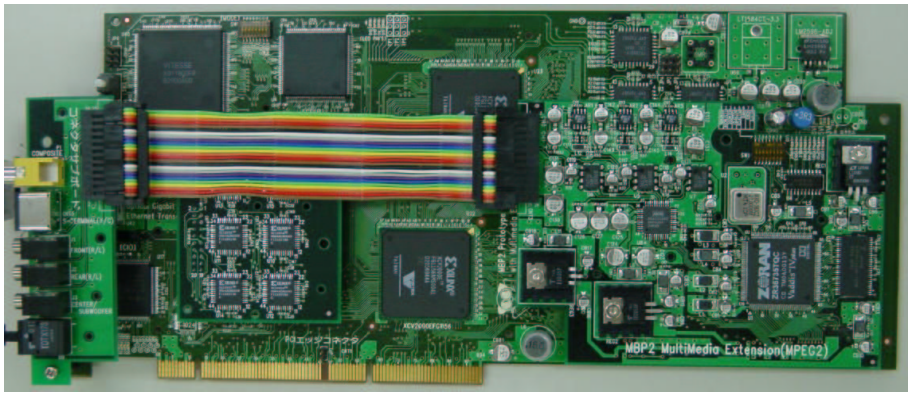

図 11: マルチメディア対応高速通信カードの外観

作製に当たり、マルチメディア用 LSI の入手が極めて困難であり、当初予定していた MPEG2 のエンコーダ LSI の入手は諦めざるを得なかった。紆余曲折の末、 ZORAN 社のデコーダ LSI のみが入手可能であった。この LSI を使用して、本カードは MPEG2 ストリームをメイン CPU のパワーを使用せずに再生可能な高機能 gigabit ethernet カードとして作製された。また、組込用マイクロプロセッサも当初予定していた MB86860 が入手困難であったため、能力が劣るが入手可能な MB86832 を使用することになった。将来的には、より高性能なプロセッサに基板上の LSI を交換する予定である。

図 11に基板の外観を示す。平成 13 年 9 月末現在、基板および基板上の 2 個の大容量 FPGA(XCV2000E-7FG1156) の回路実装と動作テストが済み、基板上のファームウェアと OS のドライバを開発中である。

■今後の展開

最大の目標であった FMM 方式を汎用スケーラブルオペレーティングシステム *SSS–CORE* に実用レベルで組み込むことに成功した。現在、 FMM 方式が *SSS–CORE* のマシンを跨ぐ唯一の資源割り当て方式である。しかし、 *SSS–CORE* の ULTRA 60 への移植とマルチメディア対応高速通信カードの開発に予定よりも多くの時間を要したため、 FMM 方式の実証実験を研究期間内に十分に行うことができなかった。また、作成した高速通信カードを使って連続メディアストリームを扱うアプリケーションの資源割り当てに関する研究を行う時間もとれなかった。これらの課題は将来課題として残されている。

今回の研究開発の中で考案した組込装置向けの三件の発明（うち出願二件）は、低消費電力で高性能な組込マイクロプロセッサの研究開発の基本技術となって、新たな研究開発プロジェクトに引き継がれて行く。また、 FMM 方式のために開発されたマイグレーション機能の応用として、ノンストップコンピューティング環境を実現することの重要さを本研究中に認識した。このため、 *SSS–CORE* を高信頼性および高可用性を持つ汎用オペレーティングシステムに育てる研究開発を新たなプロジェクトとして行うことになった。

FMM 方式自体も非常に優れた資源割り当て方式であり、 *SSS–CORE* のスケジューリング方式として活躍している。 *SSS–CORE* の完成度とユーザの使い勝手を高め、キラーアプリケーションを開拓および開発することにより、 *SSS–CORE* オペレーティングシステムと共に FMM 方式の資源割り当て方式を世の中に普及させて行きたいと考えている。

■成果リスト

- 松本 尚, 平木 敬: 自由市場原理に基づくスケジューリング方式. 電子情報通信学会研究会報告, Vol.99, No.251, pp.63–70 (August 1999).
- 松本 尚: アクセス方法及びアクセス処理プログラムを記録した記録媒体. 科学技術振興事業団, 特願平 11-255272 (September 1999).
- 松本 尚: プロセッサ. 科学技術振興事業団, 特願平 11-354203 (December 1999).
- Sasaki, S., Matsumoto, T., Hiraki, K.: On the Schedulability Conditions on Partial Time Slots. *Proceedings of 6th Int. Conf. on Real-Time Computing Systems and Applications (RTCSA'99)*, pp.166–173 (December 1999).
- 松本 尚: NIC を活用したネットワーク RAID 方式の提案. 情報処理学会研究会報告 Vol.2000, No.74, pp.79–84 (August 2000).
- 大平 怜, 松本 尚, 平木 敬: 汎用クラスタ上の資源情報を用いた HTTP サーバにおける負荷分散性能の評価. 情報処理学会研究会報告 Vol.2000, No.75, pp.31–38 (August 2000).
- 松本 尚: 高性能組み込み用プロセッサアーキテクチャの検討. 電子情報通信学会技術研究報告, CPSY, Vol.100, No.248, pp.17–24 (August 2000).
- 田中 清史, 松本 尚: 実時間処理 RISC コア Casablanca の評価. 電子情報通信学会技術研究報告, CPSY, Vol.100, No.248, pp.25–32 (August 2000).
- Matsumoto, T.: A Study on Memory-Based Communications and Synchronization in Distributed-Memory Systems. *Dissertation Thesis*, Graduate School of Science, University of Tokyo (February 2001).
- 丹羽 純平, 松本 尚, 平木 敬: ソフトウェア DSM 機構を支援する最適化コンパイラ. 情報処理学会論文誌 Vol.42, No.4, pp.879–897 (April 2001).
- 松本 尚: メモリベース通信ファシリティの評価. 電子情報通信学会技術研究報告, CPSY, Vol.101, No.329, pp.31–40 (October 2001).