計算機アーキテクチャ　83－22
（１９９０．７．１８）

# 並列計算機ＣＡＰ－Ⅱの並列ソフトウェアシミュレータ

池坂　守夫　　堀江　健志

㈱　富士通研究所

本論文では，高並列計算機ＣＡＰ－Ⅱの並列実行をシミュレートする並列ソフトウェアシミュレータＣＡＳＩＭについて述べる．ＣＡＰ－Ⅱは，数値計算や映像生成の高速実行と，並列ソフトウェアの研究環境を実現することを目的とした，分散メモリ型の高並列計算機である．ＣＡＳＩＭは，並列ソフトウェア開発を支援するため，ＣＡＰ－Ⅱでの並列実行を並列ライブラリレベルでシミュレートするツールであり，ワークステーション上で稼動する．時刻管理を基に並列実行をシミュレートすることによって，非同期性に対応したデバッグや評価の有効性を高めた．合わせて，ＣＡＳＩＭでは，ワークステーションで提供された充実したプログラミングツール（デバッガ，ウインドウ）を利用でき，並列ソフトウェア開発に威力を発揮する．

# Parallel Software Simulator of Cellular Array Processor CAP-II

*Morio Ikesaka, and Takeshi Horie*

*FUJITSU LABORATORIES LTD.*

We developed an parallel software simulator CASIM of Cellular Array Processor CAP-II. CAP-II is a highly parallel computer for high-speed numerical calculation and image generation. CAP-II also provides an environment for parallel processing to research parallel softwares. CASIM is a software tool for parallel programming. CASIM runs on a workstation, and simulates parallel libraries of CAP-II. Application programmers can exploit parallel programs before having CAP-II hardware. They can debug for parallel algorithms using tools on workstation. They can also analyze behavior and evaluate performance roughly.

## 1. はじめに

我々は，数値計算や映像生成の高速実行と，様々な並列ソフトウェアの研究環境を実現することを目的として，高並列計算機CAP-Ⅱの開発を進めている[1,2,3,4,5]．CAP-Ⅱは，最大1024台のプロセッサ（以下セルと呼ぶ）を，ネットワークで結合した分散メモリ型の並列計算機である．

このような並列システム上に，幅広い分野のアプリケーションを開発していくためには，各々の並列アルゴリズムに対応できる種々の通信方式を実現する並列ライブラリの充実が必要である．さらに，並列プログラムのデバッグや評価を支援する環境が必須である．

我々は，CAP-Ⅱでの並列ソフトウェア開発を支援する環境として，CAP-Ⅱ並列ソフトウェアシミュレータ（以下ではCASIMと呼ぶ）を開発した．CASIMは，Sunワークステーション上で稼動し，並列プログラムをSunのプロセスに対応させ，並列ライブラリレベルでCAP-Ⅱの並列実行をシミュレートする．デバッグの際には，Sunの充実したプログラミングツールを利用でき，並列プログラム開発で威力を発揮する．また，CAP-Ⅱでの並列処理の性能見積もりを与える機構も有する．

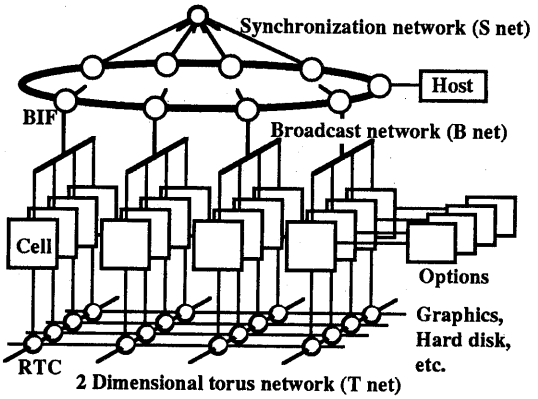
ここでは，まず，第2章でCAP-Ⅱアーキテクチャの概要を述べ，次に，第3章でCASIMの開発方針，第4章でその実現について述べる．

## 2. 高並列計算機CAP-Ⅱのアーキテクチャ

### 2.1 ハードウェア

CAP-Ⅱは，セルと呼ぶ高性能プロセッシングエレメントを，最大1024台，3種類のネットワークで接続した分散メモリ型の並列計算機である[1,2,3,4]．図1に，CAP-Ⅱのハードウェア構成を示す．

セルは二次元トーラス状の通信ネットワークで接続され自動ルーティング機能により，隣接セルだけではなく遠隔セルとも高速に通信できる．ホストと全セルは，もう一つの通信ネットワークであるブロードキャストネットワークで接続され，ホストやセルからの放送を基本にした効率の良い通信が可能である．また，ホストとセルでの即時同期を実現するため，ホストと全セルは，同期専用のネットワークで接続されている．




Cell: Cell processor BIF: Broadcast network interface
RTC: Routing controller

図1 CAP-Ⅱのハードウェア構成

### 2.2 基本ソフトウェア

CAP-Ⅱでは，メッセージ通信や同期など並列処理を記述する並列プログラミングインタフェースが，アプリケーションに提供されるソフトウェア環境の基本である．

この環境を実現する基本ソフトウェアとして，CAPドライバとセルOSがある．図2に，CAP-Ⅱのソフトウェア構成を示す．

セルOSでは，一つのセル内での複数機能の実現[5]のために，マルチタスク環境を構築する．複数タスクのスケジューリングでは，メッセージ到着によりタスクが活性化される，メッセージ駆動を基本とする．

アプリケーションでは，C言語とFORTRAN言語でプログラムを作成でき，通信や同期等の並列処理特有の機能は，並列ライブラリとして提供された関数やサブルーチンを呼び出すことによって実現する．並列ライブラリでは，アプリケーションでの並列アルゴリズムに柔軟に対応できるように，ネットワーク構成を反映した種々のメッセージ通信をサポートしている．

## 3. ＣＡＳＩＭの開発方針

CAP-Ⅱで実行する並列アプリケーションのデバッグや評価といった並列ソフトウェア開発を支援するために，次の5点に留意して，ＣＡＳＩＭを開発した．

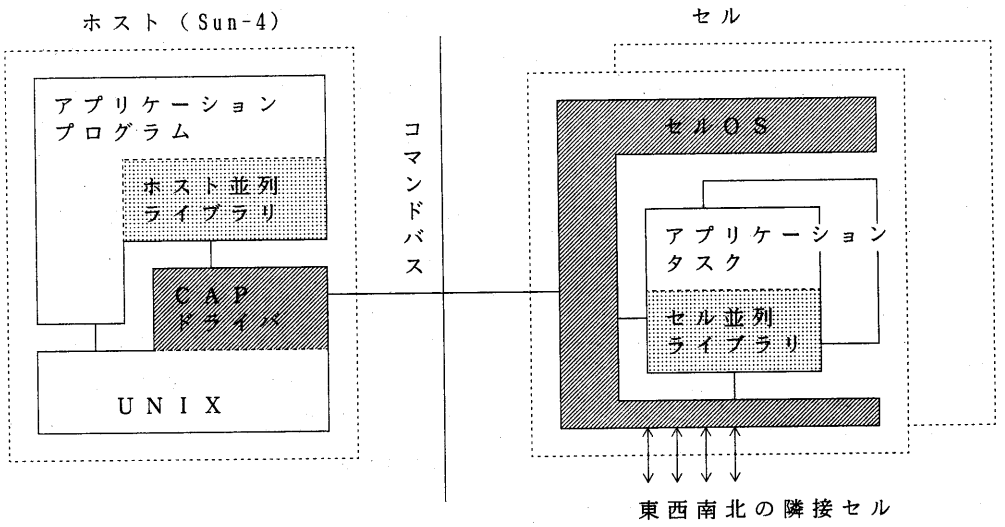



図２　ＣＡＰ－Ⅱのソフトウェア構成

(1)ワークステーション上に構築

並列プログラムの作成やデバッグを行う際，ワークステーションで提供されている充実したプログラミングツール（デバッガ，ウインドウ）を利用できる環境を構築する．

(2)並列ライブラリレベルでのシミュレーション

アプリケーションでの並列プログラムのデバッグで注目すべきは，メッセージ通信や同期などの並列処理特有の機能の論理的な動作の検証である．従って，CAP-Ⅱで提供している並列ライブラリを，シミュレートする．

(3)並列プログラミングインタフェースの保証

CASIM で開発した並列プログラムは，ソースレベルの変更なしに，CAP-Ⅱで実行できる．

(4)並列実行順序の保証

CAP-Ⅱのホストやセルを含めた多数プロセッサでの並列実行の順序を，CASIM でも保証する．並列処理，特に非同期性を持った並列処理の場合，メッセージ通信の順序が並列動作を支配し，メッセージ到着順序によってその処理内容が変化する．従って，CAP-Ⅱの多数プロセッサにわたる並列実行をできる限り忠実にシミュレートすることによって，CASIM を使ったデバッグと評価の有効性を高める．

(5)並列アルゴリズムの評価

通信のオーバーヘッド等を含めた性能予測ができる．ネットワーク競合等ハードウェアの詳細な動作を考慮した評価を行わずとも，内部処理の実行時間，通信時間，アイドル時間等の概算は，負荷分散や台数効果など，アプリケーション並列化の評価に役立ち，プログラムチューニングの指針を与えることができる．

## 4.　ＣＡＳＩＭの実現

3.の開発方針に従い，Sun ワークステーション上に，CASIM を構築した．図３に，CASIM 環境下でのソフトウェア構成を示す．

CAP-Ⅱのホストで稼動するプログラムとセルで稼動するタスクを，Sun のプロセスに対応させ，並列実行をシミュレートする（以下，それぞれを「ホストプロセス」「タスクプロセス」と呼ぶ）．

CAP-Ⅱの基本ソフトウェアである CAPドライバとセルOSの機能は，「並列実行制御プロセス（以下 CASIMサーバと呼ぶ）」で実現し，ホストやセルでのメッセージ通信などの並列実行を制御する．

CAP-Ⅱでは，セルのタスクやホストから，セルOSや CAPドライバに発行される通信や同期等のサービス要求（ＳＶＣ）は，並列ライブラリで処理される．この並列ライブラリで実行されるCAP-Ⅱの基本ソフトウェアとのインタフェースを，CASIM では，TCP/IPを使った CASIMサーバとのプロセス間通信で実現した．

「ユーザインタフェースプロセス（以下 UIFプロセスと呼ぶ）」は，CASIM サーバの生成や，ホストプロセス生成の指示を行ったり，プロセス実行のウインドウ制御，評価データ収集等を行う．

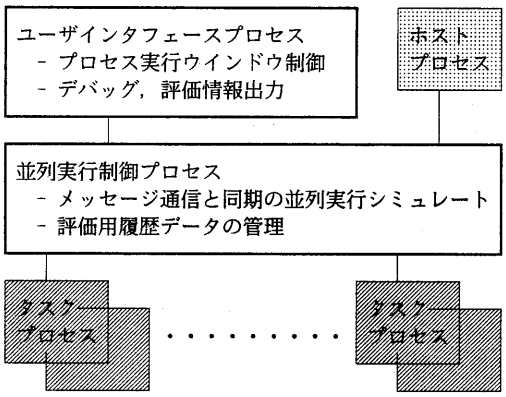



図3　ＣＡＳＩＭのソフトウェア構成

### 4.1 CASIM サーバの構成

CASIM では，CAP-Ⅱでの並列実行制御をシミュレートする CASIMサーバの実現が鍵である．

CASIM サーバでは，UIF プロセス，ホストプロセスやタスクプロセスからの処理要求に対応すべく，次のシーケンスを繰り返す．

```
・if（ UIFプロセスからの要求）｛
    UIF プロセスからの要求処理;
  ｝
・if（ホストプロセスからの要求）｛
    ホストプロセスからのＳＶＣ処理;
  ｝
・if（タスクプロセスからの要求）｛
    タスクプロセスからのＳＶＣ処理;
  ｝
・if（評価モード）｛
    ＳＶＣ処理の実行;
  ｝
```

通常は，各プロセスからの要求は，直ちに処理を行い，その結果を，要求元のプロセスに返す．

さらに，多数プロセッサの並列実行順序をできる限りCAP-Ⅱと同一にするために，時刻管理を行うシミュレーション（4.2 で後述）を実現する「評価モード」を設ける．このとき，ホストプロセスとタスクプロセスからのＳＶＣ要求に対する実行は，条件が満たされるまで延ばされる．

### 4.2 並列実行シミュレート方式

3.の開発方針で述べたように，多数プロセッサにわたるCAP-Ⅱでの並列実行順序を，CASIM でもできる限り保証するためには，対応するSun の複数プロセスの実行を制御する必要がある．

ホストとセルをSun のプロセスに対応させ単純に実行させるだけでは，その複数プロセスの実行順序は SunOSのプロセス管理に任される．すなわち，CAP-Ⅱでは起こり得ない順に，プロセスが実行される状況が発生する．

この状況を避けるために，CASIM では，時刻管理を用いたプロセス実行制御機構を，CASIM サーバに実現した．

まず，セル内複数タスクの実行については，セルOSのタスクスケジューラを，そのまま実現することによって，セル内での複数タスクの実行順序を，完全に保証する．

さらに，ホストと複数セルにわたる並列実行順序を保証するために，各プロセッサでの経過時間を基本にした時刻管理を行う．すなわち，ホストプロセスとタスクプロセスでの実行時間を計測し，これを基に，Sun での複数プロセスの実行を制御する．

この実行制御のために，「評価モード」を設け，並列実行に関わるＳＶＣに対して，次のように対応する．

(1)ホストプロセスとタスクプロセスでのＳＶＣ処理

CASIM サーバで，各プロセスの経過時間を知るために，並列ライブラリで発行される並列実行のＳＶＣ要求は，時間計測を追加した次のシーケンスに従って実行される．

```
⇨並列ライブラリでのＳＶＣ要求
・プロセス経過時間Ｔ２を求める．
・実行時間を求める（Ｔ＝Ｔ２－Ｔ１）．
・CASIM サーバに，時間Ｔを付加したＳＶＣ処理要求
  メッセージを送信する．
・CASIM サーバから，ＳＶＣ処理結果メッセージを受
```

信するまで待つ.
・ＳＶＣ処理結果メッセージを受信すると，プロセス経過時間Ｔ１を求め，再設定する.
⇦並列ライブラリからの復帰

これによって，ＳＶＣとＳＶＣの間で実行された内部処理時間を，CASIM サーバで累積することが可能となり，各プロセッサでの経過時間を求めることができる.

(2)CASIM サーバでのＳＶＣ処理

まず，ホストプロセスとタスクプロセスからのＳＶＣ要求メッセージに対応した処理の実行を，次のシーケンスに従って延期する.

⇨プロセスからの要求メッセージに対するＳＶＣ処理
・メッセージに付加された時間Ｔと，要求元プロセッサでＳＶＣ処理に要する時間ｔを，要求元プロセッサの現在の経過時間Ｅに加算する.
（Ｅ＝Ｔ＋ｔ＋Ｅ）
・メッセージに付加された時間情報を，経過時間Ｅを基に更新する．メッセージ送信要求の場合，通信時間を合わせて考慮し，メッセージ到着時刻に更新する.
・ＳＶＣ処理要求メッセージキューに，このメッセージを登録する．このとき，メッセージに付加された経過時間の小さい順に並ぶようにする.

延期されたＳＶＣ要求は，ホストとセルの全てのプロセッサの経過時間が決定したとき，処理される.

全プロセッサでの経過時間が決定する時点は，全プロセッサが並列実行を要求するＳＶＣを，もはや発行しないことが確認された時である.

これは，ＳＶＣ要求を発行して結果を待っているか，メッセージ受信や同期確立などを待っているかの何れかの「待ち状態」に，全プロセスがなったときである.

全プロセスが「待ち状態」になければ，まだ稼動しているプロセッサが存在することになり，全プロセッサの経過時間を決定することはできない.

具体的には，次のシーケンスで制御される.

⇨ＳＶＣ処理の実行
・全プロセスが，「待ち状態」にあるか，その稼動状態を調べる.
・上の条件が満たされていれば，ＳＶＣ処理要求メッセージキューの先頭を一つ取り出し，対応する処理を実行し，結果を待っている要求元に返答を返す.
（これによって，要求元プロセスの実行が再開される.）
・上の条件が満たされていなければ，各プロセスからのＳＶＣ要求メッセージの到着を待つ.

### 4.3 並列実行の履歴と評価

CASIM では，4.2 で述べた「評価モード」のとき，各プロセスでの並列実行の履歴をとることができる.

このとき，時刻管理による並列実行シミュレートを行う際に用いる，プロセッサ性能，通信や同期のオーバーヘッドを規定する値（例えば，非同期通信におけるメモリコピー時間，ネットワークの通信速度など）の基礎データを，パラメータとして指定することができる.

通信や同期のオーバーヘッドを０と指定すれば，並列アルゴリズムそのものに内在する負荷分散等の基礎評価ができる．また，CAP-Ⅱに対応した基礎データを与えれば，通信や同期のオーバーヘッドを含めたCAP-Ⅱでの性能予測ができる.

但し，時間計測の精度が粗い点，ネットワーク競合等のハードウェア要因を無視している点で，性能予測には自ずと限界がある．しかし，大まかな性能予測でも，アプリケーション並列化の評価には，充分役立つ.

### 4.4 プロセス実行のウインドウ指定

CASIM では，Sun のウインドウシステムである SunView あるいは X Window システムの環境下で，ホストプロセスと各タスクプロセスの実行を行うことができ，各プロセスの標準出力は，プロセスに対応したウインドウに表示される.

SunView 環境下では，各プロセスを"dbxtool" を用いて実行することができ，デバッグに威力を発揮する.

また，X Windowシステム環境下では，CASIM がインストールされたSun ワークステーションとは異なるリモートホストから，ネットワークを通じてCASIM を使用でき，各プロセスの実行をリモートホストでのウインドウで確認することができる．

図4に，SunView 環境下での実行例を示す．通常のウインドウによる実行と，"dbxtool" を使った実行を，例示している．

## 5. おわりに

高並列計算機CAP-Ⅱの並列ソフトウェア開発を支援する並列ソフトウェアシミュレータCASIM について述べた．

CASIM では，各プロセッサの時刻を管理し，CAP-Ⅱでの並列実行のシミュレート精度を向上させることによって，デバッグや評価の有効性を高めた．また，ワークステーションのウインドウシステムを利用した，並列プログラム開発環境を構築し，使いやすくした．

今後，基本ソフトウェアを含めた並列システムとしてCAP-Ⅱを完成させる．さらに，CAP-Ⅱでのデバッグや評価を支援する環境として，実行過程や結果のグラフィックス表示などのユーザインタフェース，並列実行の再現など，並列デバッグ環境を構築する予定である．

## 参考文献


1）石畑ほか:"高並列計算機CAP-Ⅱの構成とﾒﾓﾘｼｽﾃﾑ"，情報処理学会計算機ｱｰｷﾃｸﾁｬ 研究会（SWoPP 琉球 '90).

2）堀江ほか:"高並列計算機CAP-Ⅱのﾙｰﾃｨﾝｸﾞｺﾝﾄﾛｰﾗ"，同上.

3）加藤ほか:"高並列計算機CAP-Ⅱのﾌﾞﾛｰﾄﾞｷｬｽﾄﾈｯﾄﾜｰｸ"，同上.

4）清水ほか:"高並列計算機CAP-Ⅱのﾒｯｾｰｼﾞｺﾝﾄﾛｰﾗ"，同上.

5）佐藤ほか:"高並列計算機CAP-Ⅱによる3次元ｸﾞﾗﾌｨｯｸｽ"，電子情報通信学会ｺﾝﾋﾟｭｰﾀｼｽﾃﾑ研究会（SWoPP 琉球 '90).

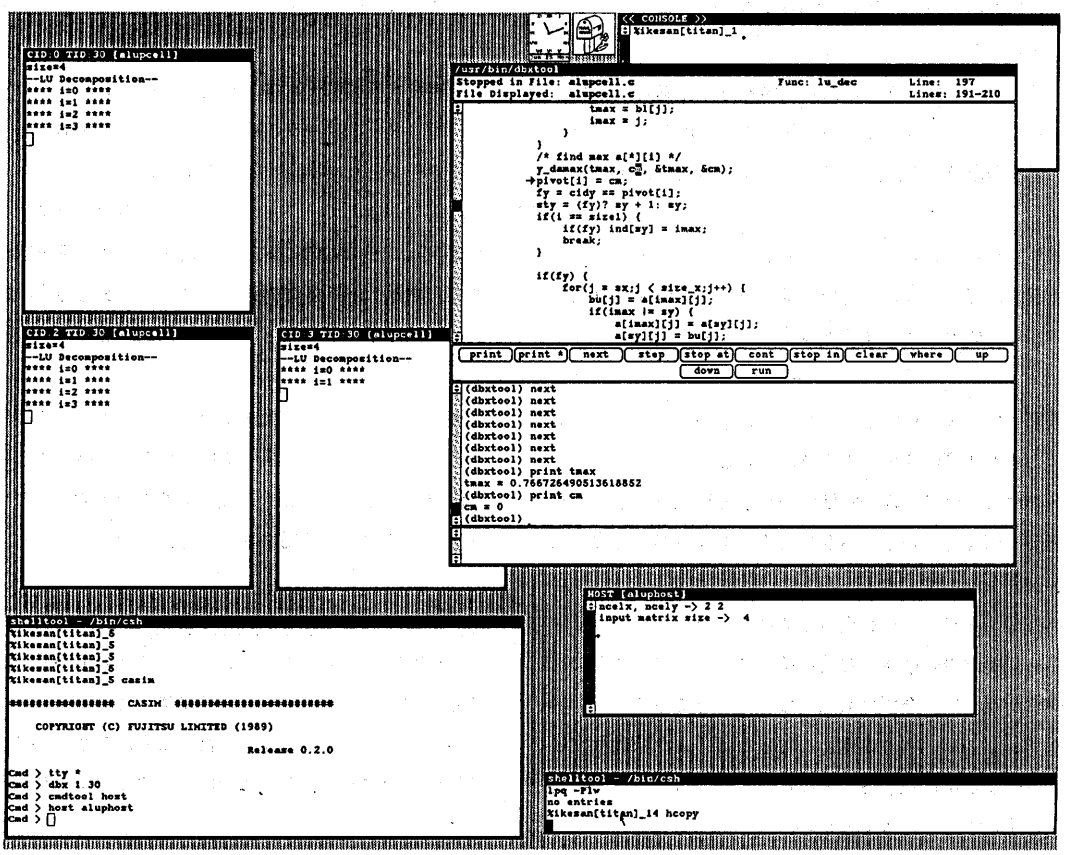




図4　ＣＡＳＩＭでの実行例