マイクロコンピュータと
ワークステーション
（１９８８．９．２７）
52－5

並列処理ワークステーション（ＴＯＰ－１）のアーキテクチャ

黒川　利明　　清水　茂則

日本アイ・ビー・エム株式会社　東京基礎研究所

本稿では、日本アイ・ビー・エム株式会社 東京基礎研究所で研究試作した高速並列処理ワークステーション（ＴＯＰ－１，　ＴＯｋｙｏ　Ｒｅｓｅａｒｃｈ　Ｐｒｏｃｅｓｓｏｒ　－　１）のアーキテクチャについて、研究開発の背景なども含めて述べることにする。

最初に研究の背景と研究の主要目的について概要する。

次にＴＯＰ－１のアーキテクチャの主要な特長について述べる。

特にスヌープ・キャッシュについてそのプロトコルやキッシュ・バス・システムの実現方法についても述べる。

ハードウェアの概要を説明したあと、アプリケーションやソフトウェアについての現在の取り組みについても簡単に報告する。

Architecture of TOP-1, A Parallel Processor Workstation

Toshiaki Kurokawa, Shigenori Shimizu

IBM Research, Tokyo Research Laboratory

5-19 Sanbancho, Chiyoda-ky, Tokyo 102　JAPAN

In this paper reported is architectural aspect of TOP-1 (TOkyo Research Processor), a prototype of high-performance, parallel processor workstation developed at IBM Research, Tokyo Research Laboratoy.
We will first describe the background of the project, its objectives and its approaches. Then we will focus on major points of its architecture, especially on its snoop cache, its protocol, and its cache-bus system realization.
After describing the overview of the hardware, we explain briefly about its application and software.

１．はじめに

日本アイ・ビー・エム株式会社 東京基礎研究所で研究試作した高速並列処理ワークステーション（ＴＯＰ－１，ＴＯｋｙｏ　Ｒｅｓｅａｒｃｈ　Ｐｒｏｃｅｓｓｏｒ　－　１）　については、すでに一部で報道されてもいるが[1]、本稿では、このＴＯＰ－１のアーキテクチャについて、研究開発の背景なども含めて述べることにする。

最初に研究の背景と研究の主要目的について概説する。

次にＴＯＰ－１のアーキテクチャの主要な特長について述べる。

特にスヌープ・キャッシュについてはそのプロトコルやキャッシュ・バス・システムの実現方法についても述べる。

ハードウェアの概要を説明したあと、アプリケーションやソフトウェアについての現在の取り組みについても簡単に報告する。

２．研究の背景と目的

2.1 研究の背景

ＴＯＰ－１を試作しようとする研究プロジェクトを起こした最大の理由は、より高性能なワークステーションがユーザには必要だということである。

後でアプリケーションの項目でも述べるが、プロフェッショナル・ユーザが用いるワークステーションの最大の用途は、プログラム開発環境やＣＡＤであり、こういった分野だけを見ても、現在のワークステーションを実現するためには、より高速高性能なマイクロプロセッサを開発するだけでなく、並列処理技術を用いて、さらに性能を向上させる必要がある。

2.2 研究の目的

したがって、本研究のプロジェクトの目的は、ワークステーション規模での並列処理技術、言い替えれば、小規模並列処理技術の確立にある。

小規模並列処理技術と一口に言っても、ハードウェア技術、ソフトウェア技術、さらに性能評価技術のそれぞれを確立する必要がある。

小規模並列処理のハードウェア技術においては、試作機を実際に開発して、ソフトウェア技術やアプリケーション技術の開発のための研究用に提供し、そういった実用を通じてアーキテクチャを再評価することが重要である。また、そのためには一台だけでなく、ある程度の台数を造り、管理していく技術も要求される。

ソフトウェア技術の研究には、並列処理用オペレーティング・システム、並列処理用プログラミング言語、さらには並列処理アプリケーション・ソフトウェアの研究が含まれる。

性能評価は、並列処理システムの研究において重要な項目で、ハードウェア性能だけでなく、ＯＳを用いたシステムとしての性能、プログラミング言語プロセッサの性能、さらには、アプリケーション・ソフトウェアまでのせた場合のシステム全体としての性能などを、なるべく多くの事例について評価する必要がある。

この場合には、実際に稼動しているハードウェア／ソフトウェア上での評価が、シミュレーション等による評価とともに重要となる。

３．ＴＯＰ－１のアーキテクチャ

3.1 小規模並列マルチプロセッサ

ＴＯＰ－１のアーキテクチャで目標としているのは、小規模な並列処理システムで、いわゆるワークステーションとして、作業現場におけるようなものである。

小規模並列処理というものの基本的な技術は、これまでの研究によってかなり蓄積されており、システム化の段階にあると考えている。

ここで＇小規模＇と言っているのは、具体的には数台から十数台の並列処理を指す。将来的には＇大規模＇でない

ものという方が適当かもしれない。

3.2 共有メモリ・アーキテクチャ

ＴＯＰ－１のメモリシステムは、すべての処理ノードから完全に均質に共有された共有メモリのみから構成され、ローカルメモリは含まれていない。従来、共有メモリ型マルチプロセッサは、共有メモリとローカルメモリの組合せで実現され、各処理ノードに固有なデータ及びコードをローカルメモリに割当てることによって、共有メモリ型マルチプロセッサ・システムの欠点であるバス及びメモリ競合の軽減を図るというのが一般的であった。

しかし、ローカルメモリを用いたそのような構成は、汎用的な使用を前提とした場合、静的なメモリ割当て、さらには、動的なメモリ再配置等かなり複雑な問題をメモリ管理に引き起こし、ソフトウェアの生産性を著しく低下させる危険がある。

ＴＯＰ－１ではソフトウェアの負荷を軽減するために共有メモリ・アーキテクチャを取ることにした。

3.3 スヌープ・キャッシュの導入

共有メモリ・アーキテクチャの問題点は、各プロセッサが共有メモリにアクセルする時に生じるバス・トラフィックによる性能低下である。

通常のシングル・ポート・メモリを使用していると、このバス・トラフィックのために２、３台で並列化が行き詰まってしまう。

この並列処理におけるバス・トラフィックのボトルネックをローカル・メモリによらないで共有メモリのアーキテクチャで解決するための素直な考え方は、各プロセッサにキャッシュ・メモリを配置することである。キャッシュ・メモリの内容に対してアクセスが入っている間は、バスを介して主記憶にアクセスする必要がないからである。

しかし、キャッシュをただ単に取り付けただけでは、複数プロセッサにまたがったキャッシュ内容をどうするかという、キャッシュ・コヒーレンシ（cache coherency）の問題が生じる。

すなわち、プロセッサＡのキャッシュとプロセッサＢのキャッシュとが同一のアドレスＸ番地をアクセスしたとした時、両プロセッサがＸ番地の内容を読み込んでいるだけならよいが、片方でもＸ番地の内容を更新した場合には他方のキャッシュの内容と（場合によると）主記憶の内容とを更新しなければならない。

もちろん、ある程度のインコヒーレンシ（incoherency）を許容しつつ、実行上問題が起きないようにすることも可能である。特に主記憶とキャッシュとの間では実際上そのようなアプローチを取ることが多い。

いずれにせよ共有メモリ・アーキテクチャでは各プロセッサにキャッシュをつけただけでは問題がキャッシュ・コヒーレンシに移るだけである。

キャッシュ・コヒーレンシの問題を解決するための方法の一つがスヌープ・キャッシュ（snoop cache）である。

スヌープ・キャッシュでは共有バス自体を用いて、各プロセッサのキャッシュがそれぞれキャッシュ・コヒーレンを実現させるべく働く。

具体的にＴＯＰ－１でのスヌープ・キャッシュについて次に紹介しよう。

４．ＴＯＰ－１スヌープ・キャッシュの働き

4.1 ReadヒットとコピーなしのWriteヒット

簡単な場合として、図１のようにプロセッサが必要とする命令／データがキャッシュ中にあって、そのデータをプロセッサが読み込んでいる、もしくはその番地にデータを書きにいっているのだが、その番地は他のプロセッサのキャッシュに共有されていないものとする。

この二つの場合には、各プロセッサにキャッシュを付けた効果が直接出ていて、バスを介した共有メモリへのアクセスは生じない。

また、これらの場合には、他のプロセッサが同時に働いていて構わない。

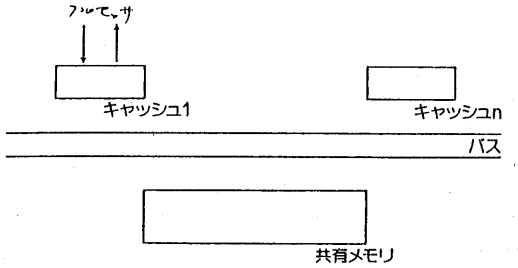

図１．　Readヒットまたはコピーなしの Writeヒット

4.2 Readミス

次にプロセッサが必要とする番地の命令／データがそのプロセッサのキャッシュになかったとする。その番地が他のプロセッサのキャッシュ中になければ、単一プロセッサのキャッシュの場合と同じで、共有メモリからデータをキャッシュに読み込み、アクセスするだけである。

他のプロセッサのキャッシュにその番地があるとスヌープの効果が現れる。この場合には大別して二つの場合がある。

一つは図２のように他のキャッシュに同じ番地のデータがあるが、その内容自体は共有メモリと同じ場合である。このような場合をＴＯＰ－１ではクリーン・シェアと呼んでいる。そして、このような場合にはデータのキャッシュ間共有が生じていなかったと同様に、データは共有メモリからもってこられる。

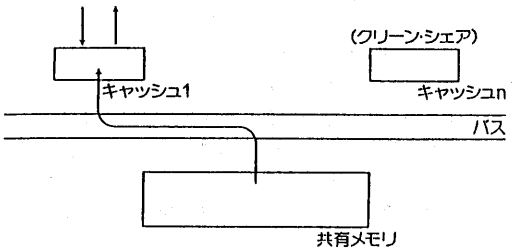

図２．　クリーン・シェアなので共有メモリからもってくる。

一方、他のキャッシュも同じ番地のデータをもっていて、しかもそのデータの内容が共有メモリと異なっている場合もある。これは基本的には図１のコピーなしのWriteヒットによって生じる。

このような場合には、共有メモリの内容をもってくるわけにはいかない。図３のように他のプロセッサのキャッシュからそのデータを自分のキャッシュにもらってくることとなる。この場合をＴＯＰ－１ではダーティ・シェアと呼ぶ。共有メモリの内容には手が触れられず、共有バスを介してキャッシュのデータが転送される。

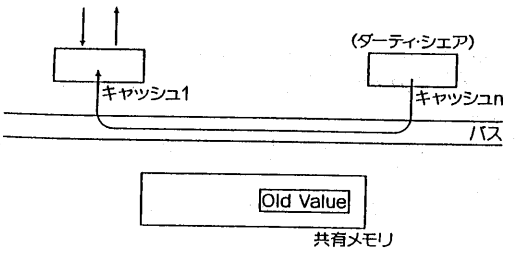

図３．　ダーティ・シェアでキャッシュ間転送をする。

4.3 他のキャッシュにコピーがある場合のWrite

他のキャッシュにコピーがある場合でのWriteヒットにも二つの場合がある。一つはキャッシュの内容とメモリの内容とが一致してるクリーンな場合で、もう一つはキャッシュの内容とメモリの内容とが一致していないダーティな場合である。

しかし、ＴＯＰ－１ではどちらの場合にも、共有メモリの内容もキャッシュの内容もすべて更新する。効果として、キャッシャ間で共有されているデータに対する書き替えは、キャッシュの内容をクリーンにする。

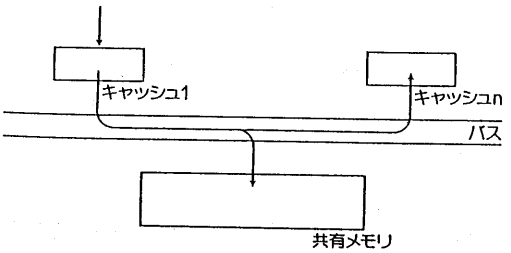

図４．　キャッシュの内容も共有メモリの内容も更新する。

ここで述べたのはＴＯＰ－１でのスヌープ・キャッシュのいわばプロトコルである。スヌープ・キャッシュのプロトコルの実現には他にも種々の方法があるだろう。ＴＯＰ－１の場合にはこのスヌープ・キャッシュと次に述べるようなキャッシュ・バス・システムとにより、キャッシュ・ヒット率が９０％以上の場合には１０台程度までほぼプロセッサ数にリニアな性能向上が期待されることとなった。

5.　ＴＯＰ－１のキャッシュ・システム

並列処理のためのキャッシュのスヌープ制御以外の点について述べよう。

キャッシュの容量は１ラインを６４ビットとして１６Ｋライン、すなわち１２８Ｋバイト分の容量をもつ。この容量は一般的にワークステーション・クラスのシステムでとられている１６Ｋバイト～６４Ｋバイトのキャッシュよりはかなり大きい。

そこでキャッシュのマッピング方法としてはダイレクト・マッピングを取った。シミュレーション等のデータから、この程度の大容量キャッシュの場合にはダイレクト・マッピングでもマルチ・ウェイ・アソシアティブ・マッピングとほとんど変わらないヒット率が得られるという期待があったからである。このことは、まだ限られた個数のデータではあるけれども、実際のプログラムで確かめられている。

スヌープ制御のためにＴＯＰ－１のキャッシュには３２Ｋバイトずつのタグ・メモリが２セットついている。これはプロセッサ側からと共有バス側からとの両方から同時アクセス可能にするため二重化している。

６．ＴＯＰ－１のバス・システム

共有バス／共有メモリ・アーキテクチャではスヌープ・キャッシュを用いても、（あるいは用いるからこそ）、バス・システムの高速化、大容量化が要求される。

ＴＯＰ－１では、６４ビットのデータを送る二本のシステム・バスを用意しており、システム・クロックが１６ＭＨｚの場合に８５Ｍバイト／秒の転送容量を確保している。

またバス使用権獲得のためのアービトレーションも速く処理したいので、１クロックでのアービトレーションを実現している。

この二本のバスは番地によりインタリーブされておりキャッシュ・メモリも図５のように各バス毎に用意されている。

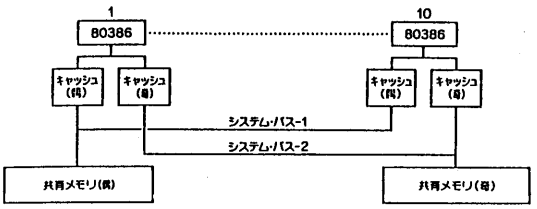

図5　ＴＯＰ－１のキャッシュ・バス・システム

７．ＴＯＰ－１のハードウェア・システム

具体的なＴＯＰ－１のハードウェア・システムでは、マイクロプロセッサとしてインテル社のi80386を使用し、浮動小数点用のエンジンとしては同じくインテル社のi80387もしくはウェイテック社の1167を使用できる。

現在標準的に考えられている構成では、プロセッサを１０台、主記憶を３２メガ・バイト、ディスク容量を３００メガ・バイト４台となっている。

ハード・ディスク・システム制御のためには制御用のカードを開発しており、これはローカル・バスを用いてハード・ディスク制御専用のプロセッサに接続される。

入出力はＩ／Ｏインタフェース・カードを用いてＰＳ／５５もしくはＰＳ／２のマイクロチャネルに接続される。

８．ソフトウェアなど

8.1 ＴＯＰ－１のアプリケーション

ＴＯＰ－１プロジェクトでは、アプリケーションを特別に限定せず、汎用の並列処理ワークステーションにすることを考えているが、強いて言えば次のようなアプリケーションが候補として上げられる。

- ロボティックス
- 自然言語処理
- グラフィックス
- プログラミング環境
- ＣＡＤ

汎用の並列処理ということで、いわゆるベクタ・プロッセサのような数値処理だけでなく、自然言語処理のような人工知能的なアプリケーションも含まれている。これは、ＴＯＰ－１のアーキテクチャとして、共有バス／共有メモリ方式を採用した一つの理由ともなっている。

8.2 ＴＯＰ－１のプログラミング言語

高速並列処理ワークステーション用のプログラミング言語として、Ｃは不可欠であろう。ＴＯＰ－１プロジェクトでは東京基礎研究所で開発したオブジェクト指向機能をもつＣ言語であるＣＯＢ（C with OBject）[2]に、さらに並列処理機能を追加することと、Common Lispを拡張して並列処理機能をもたせることを、まず当面のテーマとして取り上げている。他のプログラミング言語についても並列処理化を進めていく。

参考文献

[１] ＮＥレポート，日経エレクトロニクス 1988年6月27日号(450号) pp.100 ~ 101.
[２] 上村，手続き型言語へのオブジェクト指向の導入，情報処理，29巻4号，1988.