リアルタイムアプリケーション用 並列コンピュータ“ギガボックス”の開発

Development of Parallel Computer “GIGABOX” for Realtime Applications

エレクトロニクス事業部　溝口正信*1　笠井茂雄*1
濱村秀彦*2　村田英明*2
技術本部　堀　慶一*3

プラント制御，監視システムなどのリアムタイム アプリケーションにおいて，コンピュータの高速化は重要な課題であり，近年，その有効な手段として並列化が注目されている．当社では，並列コンピュータ“ギガボックス”を開発した．ギガボックスは，RISC 型プロセッサ“i860 XP”を最大 64 個まで搭載し，9 600 MOPS（=10⁶命令/s）の処理能力を有する．加えて，100 MByte/s の高速データ転送バス“カスタムバス”により高速入出力が可能である．さらに，ギガボックス上で動作する“負荷分散ソフトウェア”も併せて完成した．ギガボックスは当社製品“X 線 CT スキャナシステム”で実用化され，処理時間の大幅短縮に寄与した．

Faster computers are required for such applications as controlling and monitoring plant. To meet this demand, we have developed a parallel computer, the GIGABOX, with up to 64 i860 XP processors, which demonstrates 9 600 MOPS (Million Operations Per Second) processing performance. It also enables fast I/O with a high-speed data transfer bus, the Custom -Bus. In addition, software which can dynamically distribute its load to the parallel processors has been implemented. The GIGABOX, used in an X-ray CT scanner system, has dramatically reduced the processing time.

1. まえがき

CPU チップの高速化には目を見張るものがあり，15 年前に主流であった 8086 プロセッサではわずか 1 MIPS（Million Instructions Per Second）ほどであった処理速度が，現在の i860 XP プロセッサでは 150 MOPS［Million Operations Per Second (=MIPS+MFLOPS)(注)］にまで引き上げられた．それにより，エンジニアリング ワークステーション，パーソナルコンピュータなどの演算速度は飛躍的に上昇した．しかし，新型のコンピュータが発表されても，たちまちアプリケーションはそれを上回る演算能力を要求するまで成長し，“より早いコンピュータ”を求める傾向は慢性的に続いている．その結果，コンピュータの並列化の研究が進み，現在では，実用可能な技術として着実に産業界に根付きつつある．

(注) Million FLoating-point Operations Per Second

当社においても，プラント制御システムを始めとする各種制御，監視，シミュレーションシステムにおける処理の高速化，特にリアルタイム化を実現するため，高速コンピュータのニーズが高まっている．そこで，これらのアプリケーションに最適な並列コンピュータ“ギガボックス”を開発した．ギガボックスの特徴は，i860 XP プロセッサを最大 64 個搭載し，最大 9 600 MOPS の処理能力を有することと，アドレス変換機構を備え，100 MByte/s の高速データ転送が可能な“カスタムバス”を装備したことである．ギガボックスは，本報で紹介する当社製品“X 線 CT スキャナシステム”で実用化されたのをはじめ，幾つかのシステムへの適用が実現しつつある．

本報では，ギガボックスのハードウェア構成，負荷分散ソフトウェアの概要，及び適用例としての“高速 X 線 CT スキャナシステム”について述べる．

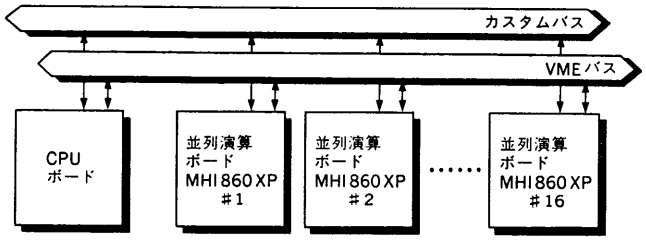


図 1　“ギガボックス”のシステム構成　ギガボックスのハードウェア構成を示すブロック図．
Structure of GIGABOX

2. 並列コンピュータ“ギガボックス”

ギガボックスは，50 MHz で動作するインテル社製の RISC (Reduced Instruction Set Computer) 型プロセッサ“i860 XP”を最大 64 個搭載した並列コンピュータシステムである．ギガボックスは，図 1 に示すように，最大 16 枚までの i860 XP 並列演算ボード（各ボードに 4 プロセッサ搭載）“MHI 860 XP”と，それらを管理する 1 枚の CPU ボードが，VME のシャーシに格納された構成をとる．その処理能力は最大で 9 600 MOPS となる．

以下に，ギガボックスの核となる演算ボード MHI 860 XP，及び高速入出力を可能とする高速データ転送バス“カスタムバス”について述べる．

2. 1 “MHI 860 XP”のハードウェア

i860 XP 並列演算ボード“MHI 860 XP”は，VME 9 U サイズのボード上に四つの PE (Processing Element) が搭載されている．ボードの写真を図 2 に，ハードウェアブロック図を図 3 に

*1 技術部エレクトロニクス技術センター
*2 技術部情報・制御技術センター
*3 高砂研究所燃焼・伝熱研究室主務 工博

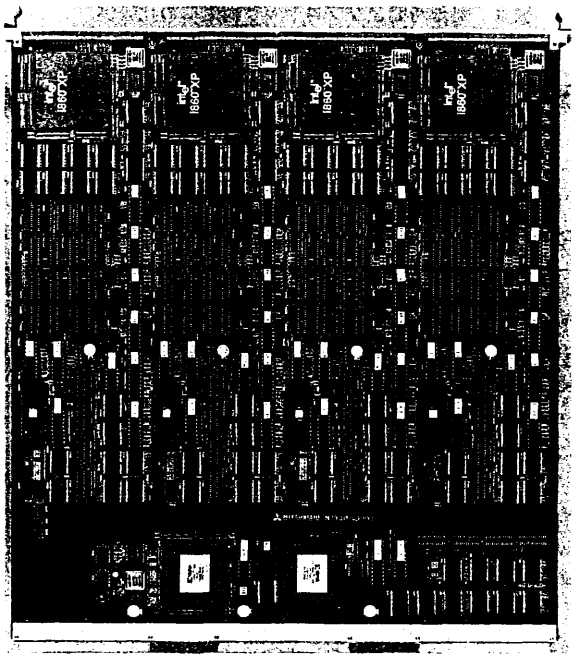


図2　i860 XP 並列演算ボード“MHI 860 XP”の写真
MHI 860 XP : i860 XP parallel processing board

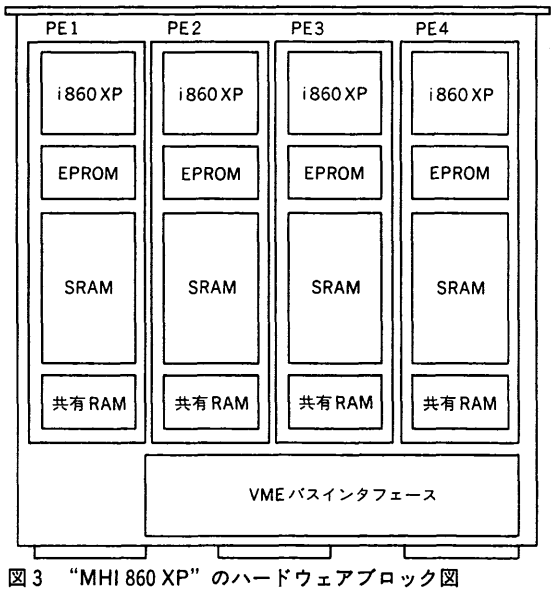


図3　“MHI 860 XP”のハードウェアブロック図
i860 XP 並列演算ボード“MHI 860 XP”のハードウェア構成を示す.
Block diagram of MHI 860 XP

示す. PE は, i860 XP プロセッサ, メモリ, VME バスインタフェースなどから構成される演算処理の最小ユニットであり, **表1**に示す性能を有する.

ギガボックスでは, アプリケーションの要求する処理能力に合わせたシステム構築が, MHI 860 XP の増設のみで実現され, 非常にコストパフォーマンスに優れる.

2. 2　カスタムバス

並列コンピュータを制御, 監視システムなどへ適用する際, 入出力の速度はシステムの性能を大きく左右する. 並列コンピュータのアプリケーションは通常, (1)多量の物理データを処理し, 少量の結果を出力するアプリケーション（X 線 CT など）, (2)少量の論理データから, 多量の物理データを生成するアプリケーション（画像合成）のいずれかに分類される. そのどちらの場合

表1　PE の仕様
Specifications of PE

CPU	i 860 XP (50 MHz 動作)
ローカルメモリ	SRAM 4 MB EPROM 128 KB
共有メモリ	1 MB
処理能力	150 MOPS (50 MIPS + 100 MFLOPS)
VME バスインタフェース	VME Revision C.1 準拠

表2　カスタムバスの仕様
Specifications of Custom-Bus

データ転送速度	100 MB/s
データ幅	32 bit
最大ノード数	80 ノード
バス信号レベル	NTL (NMOS Transistor Logic)

でも多量のデータ入出力が伴い, その速度によりシステムのスループットが決定される.

ギガボックスの入出力用高速データ転送バスとしては, 100 MByte/s の転送能力を有する“カスタムバス”を開発し装備した. このカスタムバスは, VME の P2 バックプレーンを伝送路とし, 並列コンピュータシステムのプロセッサ-入出力間, 及びプロセッサ-プロセッサ間のデータ転送をサポートする. その性能を**表2**に示す.

カスタムバスには, 高速性を更に高めるメカニズムである“アドレス変換機構”が装備されている. カスタムバスにおけるデータ転送では, 送信局は送信先のアドレスをデータに付加して送出し, 受信局はアドレス変換機構を用いてそのアドレスを解析し, データ受信を実行するか否かを決定する. よって, アドレス変換機構の設定によって, (1)送信局から単一受信局へのデータ転送, (2)送信局から複数受信局へのデータ転送が自由に選択できる. その結果, 後者では, 最大 100 MByte/s の N（受信局数）倍の高速転送能力が得られるのである.

3. 負荷分散ソフトウェア

3. 1　静的負荷分散及び動的負荷分散

並列コンピュータでは, 並列化の対象となるジョブを複数のタスクに分割し, これらを各プロセッサに割当てて処理を行う.

ここで, 並列処理の開始前に各タスクの処理時間が確定していれば, ジョブ全体の処理時間が短くなるように, あらかじめ各プロセッサにタスクを割当てることができる. これを静的負荷分散と呼ぶ.

しかしながら, 並列処理の実行時に各タスクの処理時間が変動する場合, 最短時間になるような負荷分散を, 並列処理の開始前には行うことができない. また静的負荷分散を行うと, 実行時にプロセッサのアイドル状態が発生しプロセッサ稼働率が低下する.

このような場合には, 実行時にプロセッサ稼働率が向上するように負荷分散を行うことが, 高速化に有効である. これを動的負荷分散と呼ぶ.

既存の商用並列コンピュータでは, タスク割当て・実行機能とプロセッサ間通信機能のみが提供されている. したがって, 並列プログラム作成者自身が, プロセッサ稼働率を向上させるための動的負荷分散を並列プログラムの記述に含める必要があった. このことは, 記述の複雑化と工数の増加を招いていた.

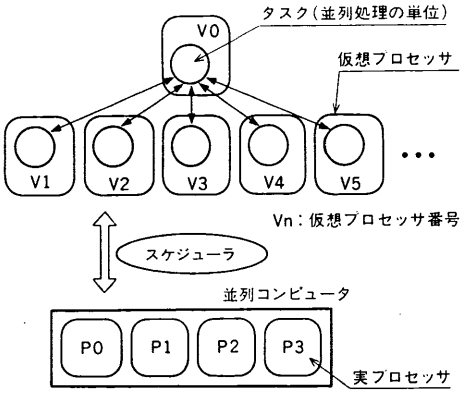


図4　仮想プロセッサと実プロセッサ　仮想プロセッサと実プロセッサの関係図.
Virtual processors and real processors

ギガボックスにおいては,

- 負荷変動のあるアプリケーションの高速化
- 並列プログラミングの容易化

の両立を目的として，負荷分散スケジューラ並びに負荷分散ライブラリを開発し標準装備した.

3．2　負荷分散スケジューラによる動的負荷分散の実現

ギガボックスに実装した負荷分散スケジューラは，仮想プロセッサと実プロセッサのマッピングの概念に基づいて構築しており，以下のように動的負荷分散を実現した（図4）.

並列プログラム作成者は，仮想プロセッサに対して並列処理の各タスクを割当てるような並列プログラムを記述する．ここで仮想プロセッサとは，タスクの割当て対象として，負荷分散スケジューラが提供するものである．仮想プロセッサは，負荷分散ライブラリを通じて無制限に得ることができ，1個の仮想プロセッサに対して1個のタスクを割当てることができる．したがって並列処理の開始前に，すべてのタスクを仮想プロセッサへ割当てることができる．この仮想プロセッサに対して，実際に並列計算機で利用可能なプロセッサを実プロセッサと呼ぶこととする.

並列処理を開始すると，負荷分散スケジューラは仮想プロセッサと実プロセッサのマッピングを行う．実プロセッサにマッピングされた仮想プロセッサは，割当てられたタスクを実行する．実行が終了すると，マッピングが解かれ，実プロセッサはアイドル状態となる．この実プロセッサは，実行可能であるにもかかわらずいまだタスクを実行していない仮想プロセッサと直ちにマッピングされる.

このように，実プロセッサのアイドル状態にある時間を少なくするよう次々とマッピングすることにより，動的負荷分散を実現した.

また，並列プログラム作成者にとっては，仮想プロセッサへの動的負荷分散を行うだけであり，その記述は容易である.

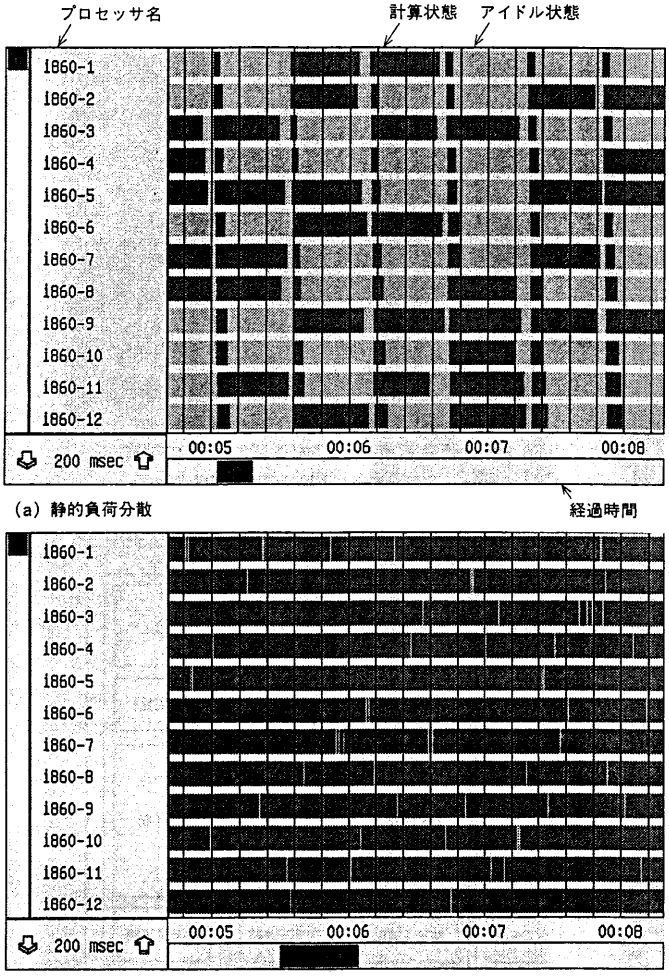
負荷分散スケジューラ及び負荷分散ライブラリは，いずれもC言語で記述しており，実装に必要なプロセッサ間通信については，PVM仕様[1]準拠のプロセッサ間通信ライブラリを利用した．PVM仕様のプロセッサ間通信ライブラリは，事実上の標準として既に多くの並列コンピュータシステムで稼働しているため，本スケジューラ並びにライブラリを移植することは容易である.

3．3　適用例による評価

モンテカルロ法による解析計算の一つに適用し，負荷分散スケジューラの評価を行った.

表3　処理時間及びプロセッサ稼働率の比較例
Example of comparison of processing time and operation rate

	静的負荷分散	動的負荷分散
処理時間　(s)	48.4	21.9
平均プロセッサ稼働率　(%)	46	95



(a) 静的負荷分散

(b) スケジューラ動的負荷分散

図5　プロセッサ状態履歴の視覚化　プロセッサ状態の履歴を表示した画面のコピー.
Visualization of processor histories

モンテカルロ法計算では，確率的試行の繰返し計算が処理の大半を占める．各繰返し処理単位は互いに独立で依存関係が存在しないため，タスクとして分割し並列処理を行うことができる．このとき，各タスクの処理時間は確率変数の値によって異なるため，静的負荷分散を行った場合には，プロセッサのアイドル状態が発生しプロセッサ稼働率が低下する．したがって，動的負荷分散を行いプロセッサ稼働率を向上させることが，全体の処理時間の短縮に不可欠となる.

1 000個の試行計算を対象に，ギガボックスの12個のプロセッサを用いて，静的負荷分散と動的負荷分散のそれぞれで並列処理を行った．処理時間並びに平均プロセッサ稼働率の比較を表3に示す．表3では，動的負荷分散の方が処理時間が短く，平均プロセッサ稼働率が大きいことが分かる.

また，視覚化ツールHeNCE[2]による静的負荷分散と動的負荷分散のプロセッサ状態履歴を図5に示す．図5では，負荷分散スケジューラが，プロセッサのアイドル状態の発生を抑えるように

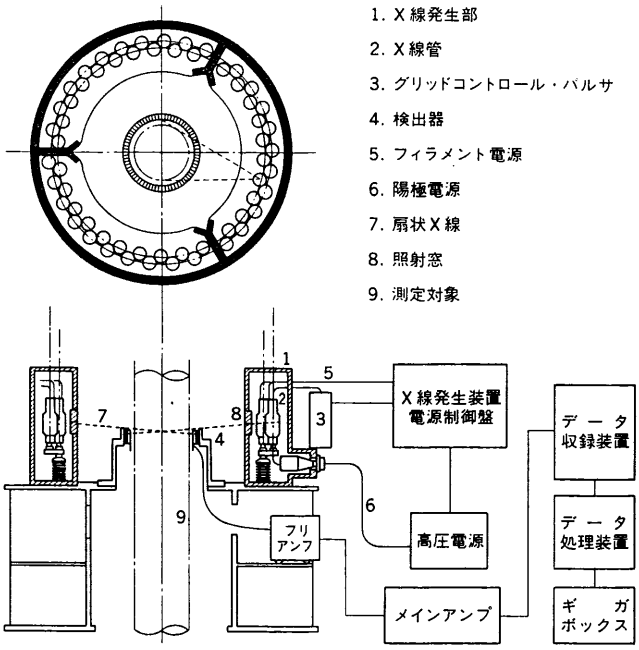


図6　高速X線CTスキャナの構成　高速X線CTスキャナの構成を示す.
Structure of high-speed X-ray CT scanner

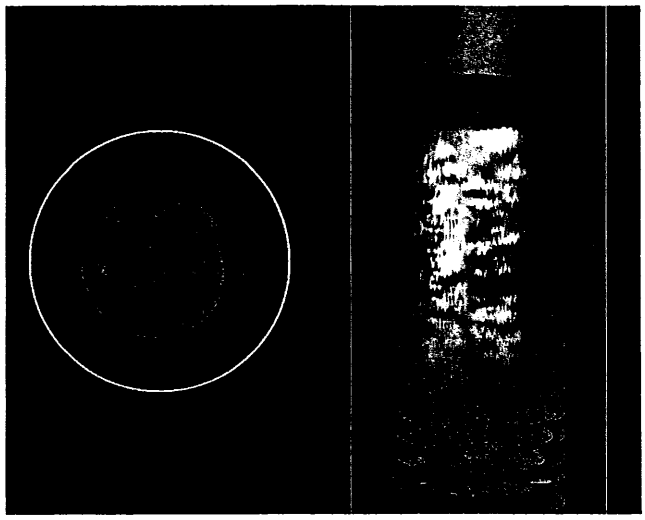


図7　気液二相流の計測例　気液二相流の二次元断面図(左)，及び合成された三次元立体図（右).
Example of gas-liquid two phase flows

動的負荷分散を行っていることが分かる.

4. X線CT画像処理の高速化への適用

4.1 高速X線CTスキャナシステム

1971年に頭部診断医療用として，X線を用いたEMI社のスキャナのプロトタイプがCTスキャナとして登場し，急速な技術進歩と相まって広範に普及した．現在，医療分野ばかりでなく，非接触・非破壊的に内部が観察できる利点から，工業分野での非破壊検査や計測に，X線やγ線のCTスキャナが利用されている.

しかしながら，これらのCTスキャナは線源と検出器を機械的に駆動する方式であるため，一断面画像を得るために必要な撮像時間が数s以上で，流れの時間変動のような高速現象の計測には適用できなかった．当社の高砂研究所では，気液二相流の時間変動の動的な計測を主目的に，撮像時間が0.5 msの世界最高速のX線CTスキャナを開発した．撮像時間の大幅な短縮によって，写真撮影でシャッタ時間を短くするのと同様に，ぼけの少ない鮮明な断面画像を得ることができるようになった．加えて，ギガボックスの利用によって，データの高速処理が可能となった.

このCTスキャナは，高速化の障害となっている機械的駆動部分をなくし，電気的にX線の発生部を高速で切換える方式を採用している．図6に，高速X線CTスキャナの構成を示す．X線発生管から射出され，測定対象を透過した扇状X線，すなわち投影データは，向かい側に置かれた検出器によって検知される．検出器からの計測信号はX線発生用の制御パルス信号と同期してギガボックスを内蔵したデータ収録装置に転送されて，コンボリューションバックプロジェクション法と呼ばれている再構成手法によって画像再構成が行われる．再構成画像から，流れの瞬時の密度分布や相分布が得られる.

図7は，空気と水の混合流すなわち気液二相流の計測例で，測定面での空気と水の瞬時の分布を表す二次元断面画像が，時間的に連続に得られている．また，これらの二次元断面画像を重ね合わせて三次元立体像を構成することができ，時間変動する流動状況を立体的に把握できる．図7の例では，気液二相流においてスラグ流と呼ばれている流れの状況が把握されている.

4.2 画像再構成の並列処理化

撮像時間が0.5 msの高速X線CTスキャナからは，毎秒2 000枚の断面画像を構成できる多量の投影データが得られる．したがって，このシステムでの計測時間を10 sとすると，断面画像20 000枚分の再構成処理が必要となる．この画像再構成処理を，画像サイズ256×256（画素）で，EWS（SUN SPARC Station 5）を用いて実行しようとすると，10 s分のデータ処理に約5 hかかることになり，実用上，処理時間が問題となる.

これに対しギガボックスを用いて処理の高速化を実現した．一断面画像分の投影データは，相互に依存関係がないため，各プロセッサに一断面画像ごとの再構成処理を順次割当てる方法で並列化を行った，その結果，プロセッサを64個用いた場合，処理時間が7 min程度まで短縮されることとなった.

5. むすび

コンピュータが高速化されれば，制御，監視システムの精度が向上するのみならず，新たなるアプリケーションの創造も期待できる．そして，並列コンピュータはその旗手となるべき存在である.

まだ実用化の域に入ったばかりの並列コンピュータは，無限の発展性を秘めている．当社においても，今後，ギガボックスのプロセッサ間通信の強化，ソフトウェアの拡充などを図り，より高速で，信頼性が高く，操作性に優れた並列コンピュータへと改良するため，鋭意努力する所存である.

参考文献

(1) Geist, G. A., Beguelin. A., Dongarra, J. J., Manchek, R. and Sunderam, V. S., PVM 3 User's guide and reference manual, Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, September 1994

(2) Beguelin, A., Dongarra, J. J., Geist, G. A., Manchek, R. and Sunderam, V. S., Graphical development tools for network-based concurrent supercomputing, In Proceedings of Supercomputing 91, Albuquerque, 1991 p.435-444