ISSN 1349-8541

Cybermedia Center
Osaka University

大阪大学サイバーメディアセンター

計算機利用ニュース

Vol. 4 No.2　2008.12
第7号
Cybermedia Center, Osaka University

特集：SX－9

大阪大学サイバーメディアセンター
大規模計算機システムホームページ:http://www.hpc.cmc.osaka-u.ac.jp/j/

利　用　案　内

主なサービス内容 （設置機種）	係・室名等	連絡先・電話番号 （大阪大学内からは下4桁が内線番号となります）	月～金	土･日･祝日
●開館サービス時間				
センター利用者の呼出 センター見学の申込等一般受付	情報推進部情報企画課 総　務　係 soumu@cmc. （吹田本館 1F）	06-6879-8804	8:30～12:15 13:00～17:15	閉室
会計事務一般	情報推進部情報企画課 会　計　係 kaikei@cmc. （吹田本館 1F）	06-6879-8810		
利用者受付 利用案内、利用申請、利用者講習会、広報誌の発行､利用負担金、図書の閲覧、貸出	情報推進部情報企画課 情報企画班 usersv@cmc. （吹田本館 1F）	06-6879-8808		
大規模計算機システムの運用･管理､セキュリティに関すること	情報推進部情報基盤課 研究系システム班 system@cmc. （吹田本館 2F）	06-6879-8813		
ネットワークの運用･管理､セキュリティに関すること	情報推進部情報基盤課 研究系システム班 network@cmc. （吹田本館 2F）	06-6879-8816		
プログラムのメール等相談	メール利用相談員 toiawase@cmc. 情報企画班 （吹田本館1F）	Tel： 06-6879-8808 Fax： 06-6879-8814	常時	

●計算機運転サービス時間

オンライン･サービス〈センター外端末からの利用〉（注）		
	すべての計算機	終　日　運　転

（注）＊平日の17時15分以降及び土曜日･日曜日･祝日は自動運転を行っていますが、障害が発生した場合は、その時点でサービスを中止することがあります。

＊スーパーコンピュータは、毎月の第1月曜日8:30～11:00に保守点検のため停止します。

＊サービス時間は原則として上記の時間となっていますが、状況によって変更する場合がありますので、ご了承ください。

＊E-mailアドレスの末尾にはosaka-u.ac.jpをお付け下さい。

特集　ＳＸ－９

スーパーコンピュータ SX-9 のハードウェア

稲坂　純　　萩原　孝
日本電気株式会社　コンピュータ事業部

1. はじめに

2008年7月よりセンターに導入された新しいスーパーコンピュータシステムである SX-9 のハードウェアについて紹介します。SX-9 は、旧システムの SX のアーキテクチャを継承し、単一 CPU 性能 102.4GFLOPS、ノード性能 1638.4GFLOPS に引き上げています。また従来の SX 向けに開発したソフトウェア資産を継承するシステムです。ノードあたり 16 台の中央処理装置（Central Processing Unit、CPU）を有し、ノード間を専用の超高速のノード間接続装置（Inter-node Crossbar Switch、IXS）により接続し、システム全体で 10 ノード、総 CPU 数 160、総合演算性能 16.3TFLOPS、総メモリ容量 10T バイトを有する、実効演算性能、スケーラビリティ、及び使いやすさに優れたスーパーコンピュータです。

特長は以下の通りです:

① CPU あたり 100GF を超えるベクトル演算性能
② ノードあたり 1.6TFLOPS の演算性能と 1TB の大容量メモリ
③ 専用のノード間接続装置（IXS）により、ノードあたり 64GB/s×2 のノード間データ転送バンド幅
④ 65nm 銅配線技術を用いた超高速、高集積 CMOS LSI による高密度実装
⑤ 省電力設計による消費電力・発熱量の削減、及び高密度実装による設置面積の削減

本稿では、上記特性を備えた SX-9 のシステム構成（アーキテクチャ）、及びテクノロジの概要について説明します。

2. SX-9 システム構成

SX-9 は、CPU と主記憶装置（Main Memory Unit: MMU）を密結合した共有メモリ型のシングルノード 10 台中 8 台を超高速なノード間接続装置（IXS）にクラスタ接続することにより、分散並列処理を可能としたシステムです。SX-9 のシステム諸元を表1、システム構成を図 1 に示します。

表 1　SX-9 システム主要諸元

中央処理装置（CPU）	CPU 数	160
	ベクトル性能	16.3TFLOPS
主記憶装置（MMU）	容量	10TB
	データ転送性能	40 TB/s
入出力機構（IOF）	スロット数	240
	総入出力性能	128 GB/s
ノード間接続装置	データ転送性能	64 GB/s×2／ノード

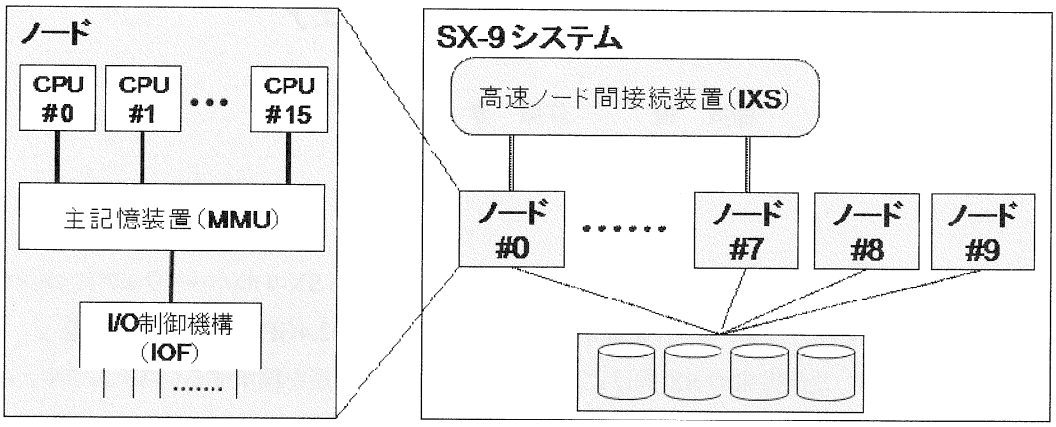


図1　SX-9 システム構成

各ノードは CPU 数 16、総合演算性能 1638.4GFLOPS、主記憶容量 1T バイト、8 スロットの入出力用の PCI スロットを持ち、演算性能、メモリスループット性能、入出力性能などのトータルバランスに優れ、高い実効性能を実現します。また、ノード内の共有メモリを利用した並列処理用に通信レジスタ(Communication Register:CR)を備えており、自動並列処理や、Open MP 指示行による並列処理時の CPU 間同期制御を高速に実行することが可能です。

SX-9 システムは、10 ノード中 8 ノードを超高速に接続する専用の IXS によりクラスタ接続し、表 1 に示すように総 CPU 数 160、総合主記憶容量 10T バイトの構成であり、総合演算性能 16.3TFLOPS という高い演算性能を実現します。

また SX-9 の 1 ノードにおける保守エリアを含む設置面積、及び消費電力はそれぞれ約 $2m^2$、及び約 30KVA であり、設置環境及び性能あたりの消費電力は従来のシステムと比較して大幅に改善しています。次節以降、SX-9 システムのハードウェアについて説明します。

中央処理装置(CPU)

SX-9 の CPU は従来の SX アーキテクチャを継承しつつ、さらなる機能・性能の強化を図っています。図 2 に CPU の構成を示します。CPU は、スカラユニット部、及びベクトルユニット部により構成され、プロセッサ／メモリ間ネットワークを介して MMU と接続されます。スカラユニットは命令の解読、ベクトルユニットへのベクトル命令の供給・起動、及びスカラ命令の実行を行います。一方、ベクトルユニットはベクトル演算部、及びベクトル制御部から構成されます。ベクトル演算部は 8 セットのベクトルパイプラインを備え、各ベクトルパイプラインは、乗算器×2、加算／シフト演算器×2、及び除算／平方根演算器×1、論理演算器×1 の 6 種のそれぞれ独立に動作可能な演算パイプライン、マスク演算パイプライン、ロード／ストアパイプライン、マスクレジスタ、及びベクトルレジスタにより構成されています。したがって、科学技術計算で主に利用される乗算と加算は、3.2ns のクロックサイクルあたり 32 個が同時に処理することが可能であり、最大 102.4GFLOPS のベクトル演算性能を実現します。

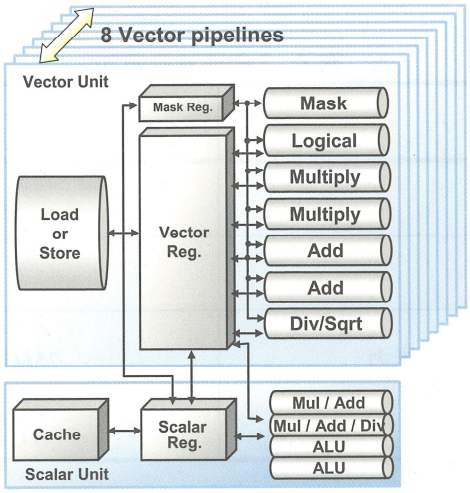


図 2　CPU の内部構成

スカラユニットは 64 ビット RISC アーキテクチャであり、L1 キャッシュ、命令同時デコード数 4、最大命令同時発行数 6 であるスーパスカラアーキテクチャ、アウトオブオーダー実行、及び命令投機実行の採用により短ベクトル時のベクトル命令発行性能、及びスカラ性能を向上させ、3.2GFLOPS の最大演算性能を実現しています。

主記憶装置（MMU）

スーパーコンピュータにおいて高い実効性能を実現するためには、高い演算性能に見合うだけの高いデータ供給性能が必要となります。SX-9 の MMU は、高速、かつ均一にメモリアクセス可能な共有メモリ方式を採用しています。MMU は 32768 個のメモリバンクを 64 バンク毎にメモリバンクを管理する制御部で管理し、その制御部にメモリバンクキャッシュ機構を備えてバンク競合を最小限に抑えつつ、高いメモリスループット性能を実現しています。

これにより SX-9 は表 2 に示すように、シングルノードにおいて、1T バイトのメモリ容量、及び 4T バイト／秒のメモリスループット性能、システム全体では 10T バイトの総メモリ容量、及び 40T バイト／秒の総合メモリスループット性能を実現します。

また、SX-9 は、従来 SX シリーズ同様に 3 次元実装構造の MMU カード実装を引き続き採用しています。これにより、RAM とメモリ制御部間の物理的距離を短くし、RAM の高速動作、及び RAM と CPU 間の高速信号伝送を実現しています。一方、メモリ信頼性確保のための ECC（Error Check and Correct）符号、タイミング、パリティ、2 重化回路などの採用による高いメモリ故障検出率の実現、擬似障害によるチェック回路の診断機能、エラー内容から即座にエラー箇所を指摘するビルトイン機能などにより、RAS（Reliability, Availability, Serviceability）機能の充実を図り信頼性を高めています。

表 2　主記憶装置諸元

	諸元
主記憶装置容量	1T バイト
インターリーブ数	32768
CPU あたりのデータ供給能力	256G バイト／秒
最大データ供給能力	4T バイト／秒

入出力機構（Input/Output Features：IOF）

入出力機構はシステムのスループットを高く保つために、SX-9 の高いプロセッサ性能、及びメモリ性能に見合った高速なデータ転送性能を備えており、ノードあたり 24 スロット（但し、1 スロットはシステム制御用）、16G バイト／秒、SX-9 システム全体では 230 スロット、160G バイト／秒の性能を有します。入出力インタフェースは、ファイバチャネル（Fiber Channel: FC）、シリアル SCSI（Serial Attached SCSI: SAS）などの汎用インタフェースをサポートしており、様々な周辺機器を接続することが可能です。また、ネットワークインタフェースとして、ジャンボフレームに対応したギガビット・イーサネットを備えています。

I/O 処理において、CPU は全ての I/O 装置に対して対等にアクセスすることが可能であり、実行負荷の低い CPU を I/O 制御に割り当てるなど、CPU の効率的利用を可能としています。

ノード間接続装置（IXS）

SX-9 は図 1 で示したように、共有メモリ型のシングルノード 10 台中 8 台を超高速専用クロスバスイッチを介して結合することにより、分散並列処理を可能としています。各ノードは、ノード間通信制御部（Remote Control Unit: RCU）を介して IXS とケーブル接続され、分散並列処理において低通信レイテンシ、及び高通信スループットを実現します。

RCU のデータ受信部、及び送信部はそれぞれ独立に動作可能であり、ノードあたり 64G バイト／秒×2（双方向）の通信バンド幅を実現します。また、RCU は CPU とは独立に動作するデータムーバを持つことにより、異なるノードのメモリ間でデータ転送を行なうリモートメモリアクセスを CPU 動作とは完全に独立して行なうことが可能です。SX-9 の IXS は、従来の SX シリーズの IXS で採用していた回線交換型から、パケット交換型にデータ交換方式を変更しました。これにより、従来の回線制御オーバーヘッドを削減し、小データサイズの転送性能の向上を図っています。

3. SX-9 のテクノロジ

LSI 技術

これまで SX シリーズでは CMOS テクノロジによる高集積化，及びプロセッサのパラレル化により高性能化を実現しつつ、コストパフォーマンスを向上させてきました。

SX-9 では、さらに高い性能を実現するために LSI 技術及び回路技術を発展させています。 また、システムの性能向上のためには、LSI 間信号伝送の高速化も非常に重要です。SX-9 では、新規のマルチチャネル・シリアル・インタフェースを開発し、LSI 間の高速データ転送を実現しています。また、インタフェース回路の低消費電力化、小面積化により LSI への多チャネルの搭載を実現しています。表 3 及び図 3 に SX-9 の CPU LSI 諸元、及び CPU チップ外観をそれぞれ示します。

表3　CPUチップ諸元

テクノロジノード	65nm
搭載トランジスタ数	3億5千万トランジスタ
電源電圧	1.0V
ピン数（内信号ピン）	8,960（1,791）
配線層構成	銅11層
I/Oインタフェース	CML（Current Mode Logic）
実装形態	ベアチップ実装

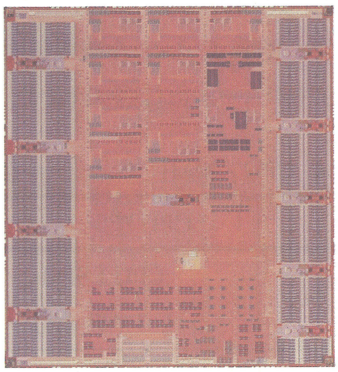


図 3　CPU チップ外観

SX-9 に使われる LSI の共通仕様として、65nm CMOS プロセス、11 層銅配線、及び低誘電率層間絶縁膜などの採用による配線遅延の改善、MIM（Metal-insulator-metal）プロセスの開発による大容量オンチップキャパシタの実現、ゲート酸化膜の薄膜化による高性能な低電圧電源の実現などを行なっています。さらに、新規のマルチチャネル・シリアル・インタフェースの開発により、CPU-MMU 間の低転送レイテンシを実現し、同時に LSI 上の電源分離、アナログ回路の削減、制御信号のデジタル化などによりノイズ耐力及びエラーレートの格段の向上を実現しています。

LSI 内部の RAM 回路は、デバイス性能を最大限引き出すために専用設計されています。また、LSI の低消費電力化のために、読み出し回路のダイナミックパワーの削減、及びリーク電流の少ないトランジスタの使い分けによるスタティックパワーの削減を実現しています。また、高速なクロック動作を実現するために、LSI 外部からクロックを逓倍する APLL（Analog Phase-Locked Loop）回路を採用しています。

高速システムにおける処理能力の向上には、LSI 内信号伝送の高速化とともに、LSI 間信号伝送の高速化が必要となります。同様に、信号伝送の高速化を妨げる要因となる電源ノイズ対策も重要です。SX-9 では高速、かつ安定し

た信号伝送を実現するために、信号伝送時の減衰が小さい低損失材料を使用した基板、伝送信号の波形を改善するイコライズ機能を備えた回路、及び波形ひずみの少ないソケットやコネクタなどを採用しています。また、トランジスタが高速化し、電源電流の時間変化が大きくなることにより電源ノイズが増加するため、デカップリング用コンデンサの搭載数最適化などにより電源ノイズの低減を実現しています。

実装技術

SX-9は、世界最高性能、高コストパフォーマンス、及び優れた設置性を実現するために、高密度LSI実装技術、高密度接続技術、高効率冷却技術、及び高性能電源モジュール技術により、従来のワンチッププロセッサをさらに進化させました。

CPU、及びMMUモジュールの諸元を表4に、CPU、及びMMUモジュールの外観を図4にそれぞれ示します。超高速動作が要求されるCPU／MMUモジュールは、高密度実装により大型で多ピンのLSIを搭載可能としています。CPUモジュールは、CPU LSIをビルドアップ基板表面にベアチップ実装し、裏面には高速シリアル信号を伝送するケーブルを接続するための高密度コネクタを搭載しています。MMUモジュールは、メモリ制御用のHUB LSIとSDRAMをプリント配線基板に実装しています。

次にシステム実装技術について述べます。SX-9はルータ(RTR)モジュールと呼ばれるメインボードならびに高周波多芯ケーブルを介して、CPUモジュールとMMUモジュールを相互接続するケーブルインタフェース接続構造を採用しています。これにより、メモリユニットの高密度化とCPUモジュールとMMUモジュール間の効率的な接続を実現することができました。RTRモジュールは図5に示すように32枚のMMUモジュールと2個のルーティングスイッチLSIを搭載しています。MMUモジュールから送信されたデータは ルーティングスイッチLSIでルーティングされ、高速シリアルインタフェース信号に変換された後、RTRモジュール裏面に搭載されたコネクタを経由し、高周波信号伝送用に開発した多芯ケーブルを介してCPUモジュールに伝送されます。

表4 CPU／MMUモジュール諸元

項目		CPUモジュール	MMUモジュール
搭載LSI(形態)		CPU LSI×1 (ベアチップ)	HUB LSI×1 (FBGA)
	ピン数	8960	840
	IOピッチ(μm)	168	168
搭載RAM		—	μBGA×24
配線基板	種類	ビルドアップ プリント配線基板	プリント配線基板
	基板サイズ(mm)	140×112.5	110×65
	基板厚(mm)	1.6	1.56
	基板層数	4ビルド－8コア－4ビルド	12
	配線密度(μm)	配線幅／間隙 ＝18／20	配線幅／間隙 ＝80／80
モジュール	入力端子数	2628	170
	冷却	空冷	
	消費電力(W)	240	41

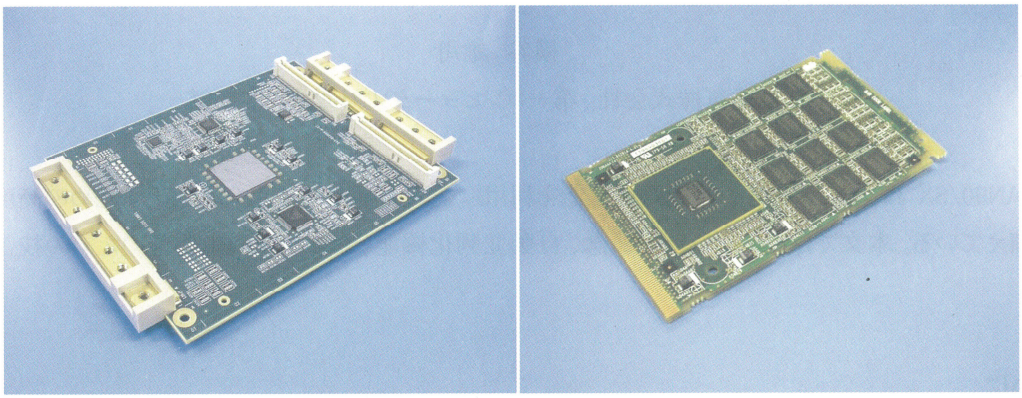


図 4　CPU モジュール、MMU モジュールの外観

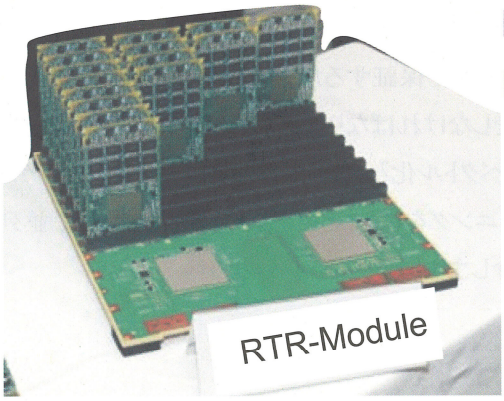


図 5 RTR モジュールの外観

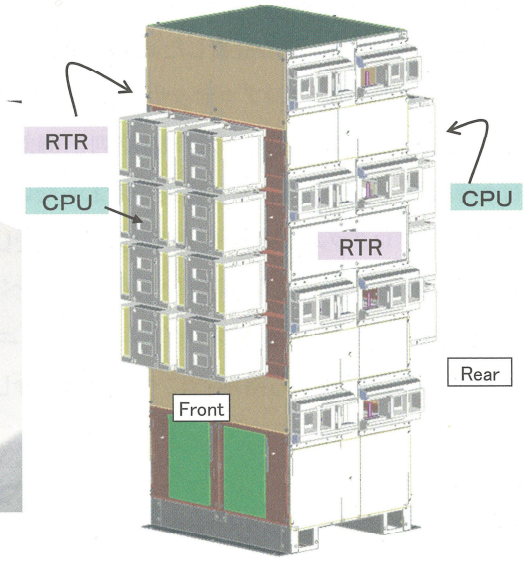


図 6　布線ボックスの外観図

CPU モジュールと RTR モジュール間の高密度多芯ケーブルは図 6 に示す布線ボックスと呼ばれるユニットに収納されています。このケーブルはノード内に搭載される CPU モジュール 16 ユニットと RTR モジュール 16 ユニット間を 1 対 1 で相互に接続することで、CPU チップと MMU モジュール間は独立した専用の高速信号伝送媒体を有することが可能となり、高性能 CPU と大容量のメモリモジュール間の広帯域のデータ転送バンド幅を実現し、SX-9 システムの高性能化に寄与しました。

4.　おわりに

本稿では SX-9 のハードウェア概要について説明しました。SX-9 は、スーパーコンピュータの要件である CPU の高い演算性能と、それに見合う主記憶からのデータ供給能力のバランスを重視し、ユーザに使いやすい大規模な分散共有メモリ型スーパーコンピュータとして開発しました。NEC は、今後も様々な研究分野の発展を支える強力なツールであるスーパーコンピュータを開発していきます。

SX-9 FORTRAN90/SXの自動並列化機能

横谷　雄司

日本電気株式会社　第一コンピュータソフトウェア事業部

概　要

FORTRAN90/SX は、SX-9 のもつ並列処理機能を利用して、その性能を十分に引き出すための高度な自動並列化機能を備えている。本文では、並列処理の概念、自動並列化機能、および並列化促進のための技法について紹介する。

1.　はじめに

SX-9 は、ベクトル処理に並列処理機能を融合した「スケーラブル・パラレル・スーパーコンピュータ」です。このハードウェアのもつ高い能力を発揮させるためには、コンパイラのベクトル化/並列化機能が重要な役割を果たします。

FORTRAN90/SX は、SX のハードウェアに密着した高度な最適化、ベクトル化、並列化機能を有した Fortran95 コンパイラです。SX-9 は、1 ノード内 16 台のプロセッサの範囲内では、利用しやすい共有メモリ方式を採用しており、FORTRAN90/SX は、この共有メモリ方式を使った自動並列化機能を提供しています。

一般に、並列化を行う時には、並列実行しても結果が変わらないことを保証するために、データの依存関係の解析を行い、細心の注意を払ってプログラムの変形や指示行の挿入をしなければなりませんが、自動並列処理機能を利用すると、それらの作業を自動的にコンパイラが行います。また、ベクトル化と同様に、効果的に並列化を行うためのオプションや指示行が用意されており、十分にプログラムのチューニングを行うことが可能です。本稿では、並列処理の概念と、自動並列化機能および関連する指示行についてご紹介します。

2.　並列処理とは

並列処理とは、1 つの仕事をいくつかの小さな仕事(タスク)に分割し、それを複数の CPU で並列に実行することです。FORTRAN90/SX コンパイラが備えている自動並列処理機能とは、「コンパイラがプログラムを解析して、並列に実行可能なループや文の集まりを抽出し、ループの繰り返しや文の集まりを複数のタスクに自動的に割り当てて実行時間を短縮する機能」です。

コンパイラが、do ループを 4 つのタスクに分割して実行するイメージは、次の図のようになります。この例では、外側ループの 100 回の繰り返しを 4 つに分割して、各 CPU 上で各々を並列に実行します。

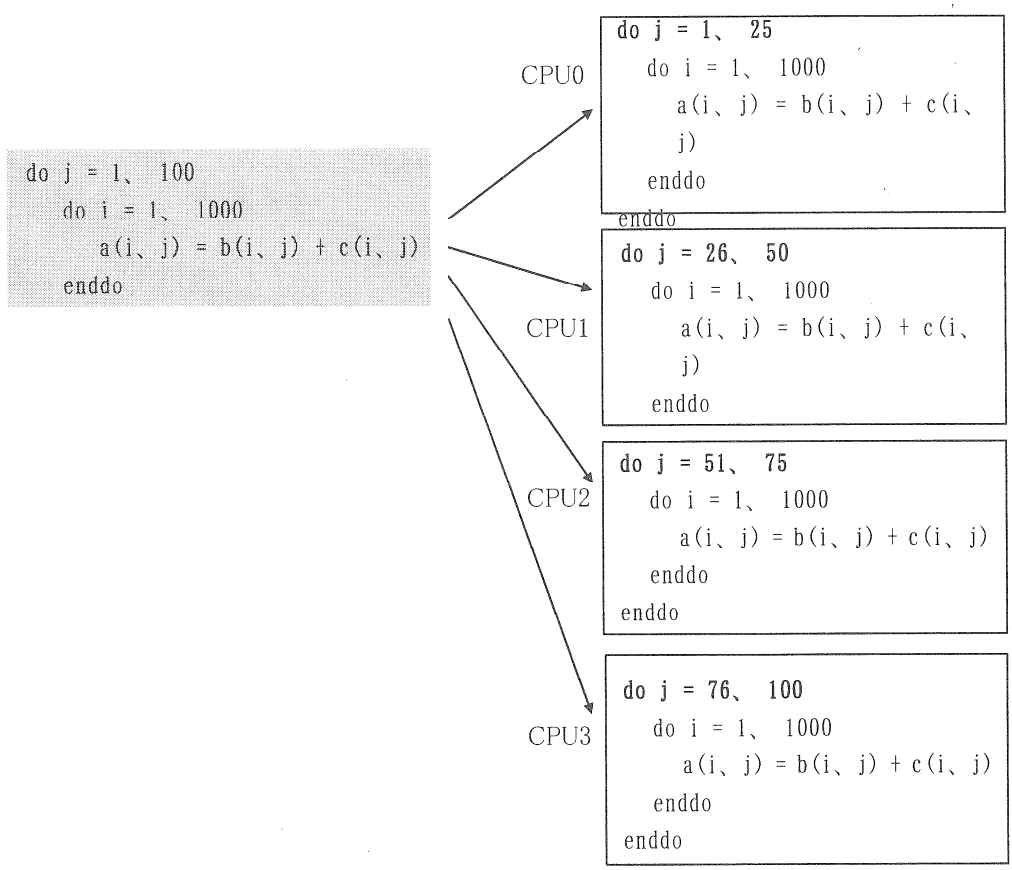


図 1 並列実行のイメージ

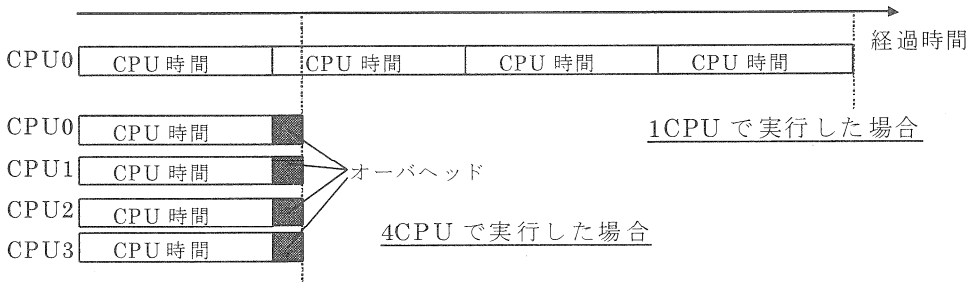
この例の配列 a は分割されて、各タスク毎に値(a(i、1)～a(i、25)、 a(i、26)～a(i、50)、 a(i、51)～a(i、75)、 a(i、76)～a(i、100))が計算され、定義されるので、a は各タスクから共通に参照できるグローバルな領域に割り当てられます。このような各タスクから共通に参照できるデータをタスク間共有データと呼びます。これに対して、並列実行される各タスクから、非同期に定義/参照を行うと、結果が不正になってしまうようなデータ(上例では i)は、各タスク毎にローカルな領域(スタック)に割り当てられ、タスク固有なデータと呼びます。自動並列化機能は、このようなデータの割当ても適切に行います。

この自動並列化機能は、オプション"-P auto"を指定するだけで利用可能です。

```
sxf90  -P auto  program.f
```

ここで、"実行時間の短縮"には注意が必要です。並列処理は、1 つの仕事を分割して、並列に実行を行うわけですから、CPU 時間が削減されるわけではなく、経過時間が短縮されることになります。また、仕事を各タスクで並列実行させるための処理(オーバーヘッド)も必要となり、CPU 時間は、かえって増加することになります。たとえば、CPU 時間と経過時間の関係は、以下のようになります。

図 2 並列処理における CPU 時間と経過時間



2.1 並列処理とベクトル処理

ここで、“ベクトル化による実行時間の短縮”と“並列化による実行時間の短縮”との相違を明確にしたいと思います。ベクトル処理とは、規則的に並んだ複数個の配列データを一度に演算する高速なベクトル命令を使って処理を行うことであり、この場合、CPU 時間が短縮され、同時に経過時間も短縮されます。これに対して、並列処理では、先に述べた通り、合計の CPU 時間は並列化のオーバヘッドにより、単一 CPU で実行した時よりも増加することになりますが、経過時間を短縮することによって高速化を図ります。したがって、ベクトル化の場合は、単一 CPU での実行ですが、上手にベクトル化できれば、スカラで実行した時よりも、一般的に 10 倍以上の性能向上が期待できます。並列化の場合に期待できる性能向上の効果は、最大で使用可能な CPU の個数倍となります。

これらのことより、基本的には、ベクトル化と並列化を組み合わせて利用し、多重ループの内側ループについてはベクトル化を行い、外側ループを並列化することが、高速化を図る最善の方法となります。

また、並列処理した場合には、オーバヘッド時間が加わりますので、並列に実行される仕事量(粒度と呼びます)が十分に大きくなければ、並列化の効果は期待できません。当然ですが、並列処理のオーバヘッド時間よりも並列実行される部分の実行時間の方が小さければ、並列化したことにより、実行時間(経過時間)がかえって多くなってしまうことになります。

ベクトル化できるプログラムは、ベクトル化すればほとんど全ての場合に性能向上が図れますが、並列化できるプログラムは、並列化したからといって必ずしも性能が向上するとは限らないことになります。すなわち、どんなプログラムでも自動並列化すれば性能が向上するということではないことに注意して下さい。

以下では効果的な並列化の方法について説明をしていきたいと思います。

3. 自動並列化

自動並列化機能を使用すれば、その名の通り、並列用の指示行を直接使って並列プログラミングをする場合に比べ、格段に容易にプログラムを並列化することができます。自動並列化機能は、プログラムを解析し、並列化した場合の効果も調べて、並列化を行います。たとえば、並列化すれば十分性能が向上するだけの粒度をもっているか、do ループの繰り返しを並列実行しても結果不正になるような文を含んではいないかなどを調査し、可能な場合は、並列化できるようにループやデータの定義を書き直します。

自動並列化は、内部的に図 3 のようなイメージで行われます。

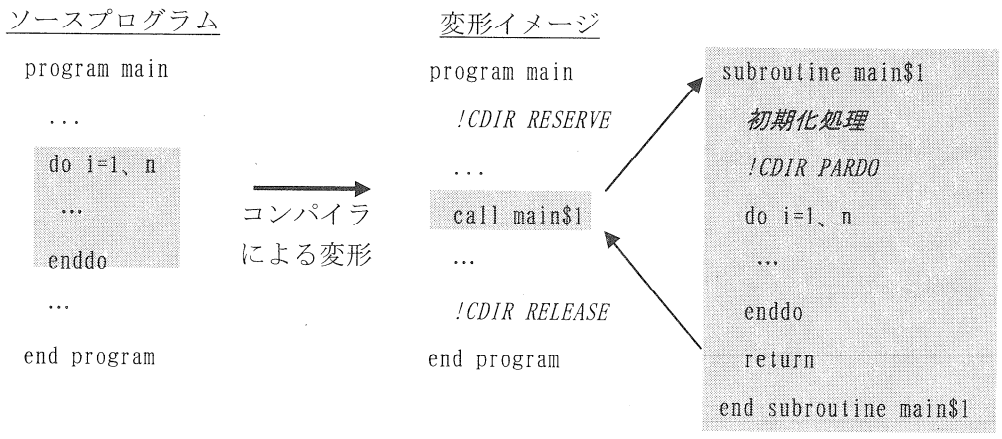


図 3 自動並列化におけるコンパイラの変形イメージ

コンパイラは並列可能なループを検索し、そのループを新たにサブルーチンとして切り出して並列化します。並列実行されるループをサブルーチンとして切り出すことによって、並列化効率を最大限に引き出すことが可能となります。上記例の!CDIR で始まる行はコンパイラによって挿入された並列制御のためのもので、RESERVE はタスクの確保、RELEASE はタスクの解放、PARDO はループを並列実行することを指示します。また切り出されたサブルーチン名には、元のサブルーチン名に$1、$2、...とサフィックスが付きます。コンパイル時に、オプション-R1 を指定することによって、並列化された様子を示すリスト(変形リスト)を参照することができます。

3.1 自動並列化の条件

自動並列化の条件は、以下の通りです。

表 1 FORTRAN90/SX 自動並列化対象

対象となる構文	DO ループ 配列式
対象となるデータの型	構造型、文字型以外の型(4 倍精度も可能)
対象となるループ中に 許される文	代入文、IF 文、GOTO 文、CONTINUE 文、 CALL 文、SELECT 構文
対象となる演算	加減乗除算、べき算、論理演算、関係演算、 型変換、組込み関数

- ループ中のデータに依存関係がある場合に、ベクトル化や並列化ができなくなることがありますが、その条件には違いがあります。次の例では、ベクトル化は可能ですが、並列化はできません。配列 a の要素がループのある繰り返し回で定義され、別の繰り返し回で参照される(たとえば a(2)は i=2 の繰り返し回で参照され、i=3 の繰り返し回で定義される)ため、ループの繰り返しを並列実行すると、結果不正になる可能性があります。

```
do i = 2,  n
  a(i-1) = a(i) * b(i) + c(i)
enddo
```

  並列化の場合は、ループの繰り返し間でデータの依存関係があると、並列化できなくなるため、ベクトル化よりも条件が厳しくなります。

- CALL がループ中にある場合、ベクトル化はできませんが、呼び出すサブルーチンが並列実行可能であれば、そのループを並列化することができます。
- ベクトル化では、4 倍精度用のベクトル命令がないため、4 倍精度のベクトル化は不可能でしたが、並列化においては、ハード的な制約はないため、並列化が可能です。

3.2 自動並列化方法

並列実行される場合は、先にも述べた通り、粒度が十分に大きくなければその効果が期待できません。また、SX-9 はベクトルマシンであるため、別稿でご説明したように、ベクトル化が性能向上には欠かせません。ベクトル化は CPU の処理時間を短縮、すなわち粒度を小さくします。したがって、ベクトル化を行っても並列の粒度をなるべく大きく保てるように考慮する必要があります。自動並列化機能は、基本的には、多重ループの内側ループをベクトル化し、外側ループを並列化します。内側ループがベクトル化できない場合は、スカラコードのまま外側ループが並列化されます。

```
subroutine sub(a、h)
      real a(600、100、100)
      integer h(100)
      do i = 1、100                          並列化
        do j = 1、100
          do k = 1、599
            a(k、j、i) = (a(k+1、j、i) + a(k、j、i)) * 0.5
          enddo                              ベクトル化
        enddo
        h(i) = h(i) + 1
      enddo
      end
```

さらに、コンパイラは、並列化の効果を高めるため、可能であればループ変形などの最適化を行い、並列化を促進します。これらの並列化の工夫について、いくつか例を紹介します。

◆ 一重ループの場合

基本的には、ベクトル化を行いますが、ループのコストが大きい、つまり粒度が大きい場合は、ループを分割して、ベクトル化＋並列化を行います。ベクトル化できない場合は、並列化だけが行われます。

```
do i= 1、100
  a(i) = sqrt(b(i)**2 + c(i)**2)      ベクトル化
enddo
```

```
do i = 1、10000
  a(i) = sqrt(b(i)**2 + c(i)**2)      ベクトル化＋並列化
enddo
```

◆ ループ融合や一重化が可能なループの場合

ループ融合や一重化などのループの最適化を行った後に、並列化を行います。

```
subroutine sub(a、b、c)
real a(10000、4)、b(10000、4)、c(10000、
4)
do j= 1、4
  do i=1、10000
    a(i、j) = sqrt(b(i、j))
  enddo
enddo
do j= 1、4
  do i=1、10000
    b(i、j) = c(i、j) - a(i、j)
  enddo
enddo
return
end
```

ループ融合
＋
一重化

```
do j=1、10000*4
    a(j、1) = sqrt(b(j、1))
    b(j、1) = c(j、1) - a(j、1)
enddo                          並列化
```

◆ 条件並列化

ループ長(粒度)あるいは依存関係が不明で、並列化の効果がコンパイル時に判断できない場合、実行時に粒度や依存関係を調べて並列コードを実行するかどうかを選択できるように条件並列化を行います。

次の例では、nx*ny>n によって、粒度が並列化するのに十分かどうかを調べています。これは、ループの繰り返し数が十分に大きいかどうかを実行時にチェックしているわけですが、このとき、コンパイラは単純に繰り返し数だけではなく、do ループ中の各演算(加算や乗算など)に対して重み付けを行い、do ループの演算コストから並列化の効果が十分に期待できる値(n)を計算して、条件並列化を行います。

また、ループ中のデータに繰り返しにまたがった依存関係がある場合は並列化すると正しい結果が得られませんが、この例では、配列 y を定義している y(ic+i)と y(id+i)の添字に現れる変数 ic と id において

id=ic または id と ic の値の差が nx 以上

という条件が満たされていれば、ループの実行中に y(ic+i)と y(id+i)により値が変更される領域に重なりがないことが保証できます。

この条件(id-ic=0 .or. abs(id-ic)>=nx)を実行時に調べ、並列実行可能な場合のみ並列コードを実行するようにしています。

```
do i = 1、nx
  aa = a
  bb = b
  do j = 1、ny
    aa = aa + x(i+1) * g(j-1)
    bb = bb + x(i-1) * g(j-1)
  enddo
  y(ic+i) = -aa
  y(id+i) = -bb
enddo
```

条件並列化

```
if (nx*ny > n .and.
   (id-ic = 0 .or. abs(id-ic) >= nx) then
    並列コード
  else
    非並列コード
endif
```

◆　手続き呼び出しを含むループの場合

ループ内に手続き呼び出しがある場合、その手続き呼び出しにより、ループの繰り返しにまたがるデータの依存関係が生じていなければ、後で説明する CNCALL 並列化指示行を指定するか、手続きをインライン展開することによって並列化を行うことができます。以下に示す例の 2 つのサブルーチン abc、 def 中のループは、展開イメージからわかるように、データに依存関係が存在しないため、並列化が可能です。

```
real x(100、 100)、 y(100、 100)、 z(100、 100)
do i = 1、 100
  call abc(x(1、 i)、 y(1、 i)、 100)
  call def(y(1、 i)、 z(1、 i)、 100)
enddo
end

subroutine abc(a、 b、 n)
real a(n)、  b(n)
do i = 1、 n
  b(i) = a(i) + 1.0 / a(i)
enddo
end

subroutine def(a、 b、 n)
real a(n)、  b(n)
do i = 1、 n
  b(i) = b(i) + sqrt(a(i))
enddo
end
```

インライン展開

```
do i = 1、 100
 !サブルーチン abc を呼び出したイメージ
  do i1 = 1、 100
    y(i1、 i) = x(i1、 i) + 1.0 / x(i1、 i)
  enddo
 !サブルーチン def を呼び出したイメージ
  do i2 = 1、 100
    z(i2、 i) = z(i2、 i) + sqrt(y(i2、 i))
  enddo
enddo
```

並列化イメージ

```
do i = 1、 100
  do i1 = 1、 100
    y(i1、 i) = x(i1、 i) + 1.0 / x(i1、 i)
  enddo
  do i2 = 1、 100
    z(i2、 i) = z(i2、 i) + sqrt(y(i2、 i))
  enddo
enddo
end
```

◆　配列構文の場合

配列構文の場合は、内部的には、do ループのイメージに展開され、その後で、ベクトル化、並列化が行われます

```
real a(2000、 1000)、 b(2000、 1000)、 c(2000、 1000)
a(1:1000、 :) = b(1:1000、 :) * c(1:1000、 :)
b(1:1000、 :) = sin(c(1:1000、 :))
```

並列化
イメージ

```
do j = 1、 1000          並列化
  do i = 1、 1000
    a(i、 j) = b(i、 j) * c(i、 j)
    b(i、 j) = sin(c(i、 j))
                         ベクトル化
  enddo
enddo
```

4. 並列化の阻害要因と並列化指示行

4.1 並列化の阻害要因

先にも述べた通り、ループの繰り返し間にデータの依存関係がある場合は、並列化はできません。並列化を妨げる要因をいくつか紹介します。

◆ 添字に重なりがある場合

```
do i = 1、n
  a(i) = b(i+1)
  b(i) = c(i)
enddo
```

この例も 3 節で述べた例と同様に、ループの繰り返し間にデータの依存関係があるために並列化ができない例です。ただし、ベクトル化は可能です。依存関係によるベクトル化可/並列化不可の理由をもう少し具体的に説明します。

ベクトル化の場合は、ループの繰り返しの実行順序は保証されますが、並列化の場合は、ループの実行順序は保証されません。上記例においてループの繰り返しと配列 b の定義/参照関係に着目すると以下のようになります。この例では、ベクトル化の場合は、参照と定義の順番は保証され、 たとえば、b(3)の値は必ず参照してから定義されることになります。

ループの繰り返し	参照	定義
1	b(2)	b(1)
2	b(3)	b(2)
3	b(4)	b(3)
...	....	....

すなわち、ループの繰り返しにまたがっての定義/参照関係は、プログラム通りの正しい関係が保持されることになります。次に並列化の場合ですが、簡単のために、ループの繰り返し 2 回毎に並列化する場合を例に考えてみます。

ループの繰り返し	参照	定義	
1	b(2)	b(1)	タスク 1 で実行
2	b(3)	b(2)	
3	b(4)	b(3)	タスク 2 で実行
4	b(5)	b(4)	
...	....	....	....

この場合は、タスク1、タスク2、… が並列に実行されることになるため、b(3)の値がタスク 1 で参照されるタイミングとタスク 2 で定義されるタイミングの順序は保証できません。すなわち、先にタスク 2 で定義された値(c(3)の値)をタスク 1 で参照して、a(2)に代入してしまう可能性があるわけです。

◆ 定義と引用が閉じていない場合

```
do i = 1、n
  c(i) = t
  t = b(i)
enddo
```

ループ中で、変数tを引用してから、定義しているため、ループの繰り返しを並列実行すると、結果不正となります。文の意味は変わってしまいますが、次のように変数tを定義してから、引用していれば、並列化が可能です。

```
do i = 1、n
  t = c(i)
  …
  b(i) = t
enddo
```

◆ IF文下にDO変数以外のインデックス変数がある場合

```
do j = 1、n
  do i = 1、n
    if (a(i、j) >= del) then
     ii = ii + 1
     ic(ii、j) = ii
    endif
  enddo
  do i = 1、ii
    b(i、j) = ic(i、j) + sin(c(ii、j))
  enddo
enddo
```

iiの更新(IF文THEN節の実行)がa(i、j)の値に左右され、iiは外側ループの繰り返しにおいても加算されていきますが、外側ループで並列化された場合、各タスクで並列実行されるループ毎にiiが加算されてしまい、iiの値が正しく計算されなくなり、bの結果が不正となります。

◆ ループからの飛び出しがある場合

```
do j = 1、n
  do i = 1、n
    a(i、j) = sqrt(b(i、j))
    if (a(i、j) >= del) go to 100
    if (c(i、j) >= 0) then
     b(i、j) = c(i、j) - a(i、j)
    else
     b(i、j) = c(i、j) + a(i、j)
    endif
  enddo
enddo
100 continue
```

並列実行されると、ループの繰り返しの実行順序が保証されないため、ループから飛び出すタイミングがプログラム通りにならない場合があり、本来値が変更されないはずの配列 b の要素の値が書き換えられてしまう可能性があります。

4.2 指示行の利用とソースプログラムの書き換えによる並列化促進

並列化の阻害要因は、上記以外にもありますが、比較的発生し易い状況は、依存関係によって並列化が妨げられる場合や、各タスクで同じデータに書き込みを行ってしまうことによって結果不正を引き起こしてしまうような場合です。これらは、プログラミング上の工夫によって、回避することが可能な場合も多々あります。また、ベクトル化の場合と同じように、プログラマには、依存関係がないことがわかっており、それをコンパイラに教えてやることによって並列化が可能になる場合もあります。このような状況に対応できるように、FORTRAN90/SX コンパイラには、指示行が用意されています。本節では、指示行の紹介と利用法、およびプログラミング上の工夫の例を紹介します。

(1) 並列化指示行

並列化指示行は、カラム 1 から

```
!CDIR オプション
```

の形式で指定します。並列化用の主な指示行のオプションとその利用法は、次の通りです。

◆ CONCUR / NOCONCUR

直後のループを自動並列化の対象とする/しないを指定します。

たとえば、並列化するとかえって性能が劣化するループをプログラムが含んでいる場合に、そのループの先頭に NOCONCUR を指定します。

◆ INNER / NOINNER

最内側ループあるいは一重ループを自動並列化の対象とする/しないを指定します。

最内側ループは、既定値では自動並列化の対象とはならないため、最内側ループを並列化できると効果がある場合に、INNER を指定します。また、一重ループの場合も、並列化効果がコンパイル時に不明な場合は、並列化の対象とはなりませんが、INNER を指定することによって並列化をすることが可能となります。

```
!CDIR INNER
      do i = 1, n
        a(i) = a(i)+b(i)*c(i)
      enddo
```

→ イメージ

```
if (n > 1000) then
  ベクトル+並列コード
else
  ベクトルコード
endif
```

この例では、コンパイラは、ループ中のコストを計算して、条件並列化を行っています。

◆ NOSYNC

ループ中の配列要素に重なりがないことを指定します。

次の例で、k1 と k2 の値がコンパイル時に不明な場合は、k1=k2 であった場合に結果不正になる可能性があるため、このループを自動並列化することはできません。k1 と k2 の値が同じでないことがわかっている場合は、NOSYNC を指定することによって並列化が可能となります。

```
!CDIR NOSYNC
      do j = 1, ny
        do i = 1, nx
          a(i, k1, j+1) = a(i, k2, j) + b(i)
        enddo
      enddo                    並列化
```

◆ SELECT(CONCUR)

多重ループにおいて、指定されたループを優先して並列化します。

多重ループにおいて、並列化した場合に最も効率のよいループを指定することができます。たとえば、最外側ループの繰り返し回数が 1 や 2 である場合、最外側ループで並列化しても、その効果は期待できません。このような場合は、次の例のように、 SELECT(CONCUR)を指定することによって、より効率よく並列化することが可能となります。

```
      do k = 1, nz
!CDIR SELECT(CONCUR)
        do j = 1,  ny
         do i = 1, nx
           c(i) = b(i, j, k) / dble(nx)
           a(i, j, k) = a(i, j, k) + c(i) / 2.0
         enddo
        enddo
      enddo
```

イメージ

```
      do k = 1, nz
        do j = 1,  ny
         do i = 1, nx
           c(i) = b(i, j, k) / dble(nx)
           a(i, j, k) = a(i, j, k) + c(i)
/ 2.0                    並列化
         enddo
        enddo
      ..
```

◆ CNCALL

手続き呼び出しを含むループを並列化してもよいこと指定します。

あらかじめ、ループ中で呼び出しているサブルーチンが並列実行されても問題がないことがわかっている場合、CNCALL を指定することによって並列化が可能となります。自動並列化機能では、引数として渡す変数が、ループ中で明示的に定義されていない(ループ中で値が代入されない)場合は、タスク間共有データ(タスク間での共通の領域を参照するデータ)となるため、渡されたデータがサブルーチン中で更新されないということが、並列実行しても結果不正にならない重要な条件となります。

```
!CDIR CNCALL
      do i = 1, n              並列化
        call sub(a(i),  x)
      enddo
```

たとえば、この例の場合、x はタスク間共有データであるため、複数のタスクでサブルーチン sub が実行され、それぞれで x が更新されると、結果不正を引き起こすことになります。したがって、x が sub 中で変更されない場合のみ、CNCALL 指示行を指定して並列化することが可能です。

◆ 強制並列化指示行

上記指示行等を指定してもコンパイラにそのループが並列化可能であることが認識できず並列化が行なわれない場合でも、利用者には、そのループが並列化可能であることがわかっている場合があります。このような場合にコンパイラに強制的に並列化を行なわせることを指定する指示行が、強制並列化指示行です。

ただし、この指示行を指定した場合、コンパイラはデータの依存関係などのチェックは行なわずに並列化をするので、並列化したときの動作の妥当性については利用者が保証しなければなりません。

```
!CDIR PARALLEL DO PRIVATE(wk)
  do j = 1、 10
    do i = 1、 100
      wk(i) = a(i) + b(i)
    enddo
    call sub(x(j)、 wk)
  enddo                    並列化
```

PARALLEL DO 指示行は直後のループを並列実行することを指定します。各ループの繰り返しで作業用に使用している配列(wk)のようにタスク固有のデータは、PRIVATE オプションで指定します。ただし、do 変数はコンパイラが自動的にタスク固有データと解釈するので、明示的に PRIVATE で指定する必要はありません。PRIVATE に指定されなかったデータはタスク間共有データとなります。

```
!CDIR PARALLEL DO
    do i = 1、 100
      call sub(a(i)、 b(i)、 x)
!CDIR ATOMIC
      s = s + a(i) * b(i)
    enddo                  並列化
```

ATOMIC 指示行は、PARALLEL DO 指示行で並列化されたループ中の総和や内積など、複数のタスクが同時に実行してはならない代入文に指定します。

(2) プログラミング上の工夫による並列化促進

◆ ループの入れ換えによる並列化促進

次のループは外側ループで並列化すると、a(i、j)と a(i、j-1)の間に依存関係があり、正しい実行結果が得られません。この場合は、ループを入れ換えることによって並列化が可能となります。

```
do j = 2、m
  do i = 1、n
    a(i、j) = a(i、j-1) * b(i、
j)
  enddo
enddo
```

→

並列化可能

```
do i = 1、n
  do j = 2、m   ループ入れ換え
    a(i、j) = a(i、j-1) * b(i、
j)
  cnddo
enddo
```

◆ 仮引数の配列サイズ変更による並列化促進

引数として渡ってきたデータは、タスク間で共有となるため、次の例では、配列cがタスク間共有変数となります。このため、最外側ループ(do k=1、nz)で並列化を行うと、配列 c の領域を各タスクで書き換えることになり、結果不正となってしまいます。

そこで、配列 c の次元の宣言を変更し、最外側ループで異なる領域を使用することにすれば、並列化が可能となります。もちろん、この sub を呼び出しているサブルーチン中の対応する配列に対しても修正が必要です。

```
subroutine sub(a、b、c、nx、ny、nz)
real*8 a(100、100、100)、b(0:100、100、100)
real*8 c(0:100)
do k = 1、nz
  do j = 1、 ny
   do i = 1、nx
     c(i) = b(i、j、k) / dble(nx)
   enddo
   do i = 1、nx
     a(i、j、k) = a(i、j、k) + (c(i-1)+c(i)) /
2.0
   enddo
  enddo
enddo
```

→

```
subroutine sub(a、b、c、nx、ny、nz)
real*8 a(100、100、100)、b(0:100、100、100)
real*8 c(0:100、100)
do k = 1、nz
  do j = 1、 ny
   do i = 1、nx
     c(i、k) = b(i、j、k) / dble(nx)
   enddo
   do i = 1、nx
     a(i、j、k) = a(i、j、k) + (c(i-1、k)+c(i、k))
/ 2.0
   enddo
  enddo
enddo
end
```

◆ 作業領域の受け渡しをしないことによる並列化促進

上の例において、サブルーチン sub を呼び出している側が、配列 c を参照していない、すなわち配列 c を単なる作業領域として確保している場合、配列 c を引数として渡さずに、サブルーチン sub 側で配列宣言することによって、並列化が可能となります。

```
subroutine sub (a、 b、 nx、 ny、 nz)                    ←―― 引数 c の削除
real*8 a(100、 100、 100)、 b(0:100、 100、 100)
real*8 c(0:100)
do k = 1、 nz
  do j = 1、  ny
   do i = 1、 nx
     c(i) = b(i、 j、 k) / dble(nx)
   enddo
   do i = 1、 nx
     a(i、 j、 k) = a(i、 j、 k) + (c(i-1)+c(i))/2.0
   enddo
  enddo
enddo
end
```

5. 並列プログラムの性能解析ツール

並列プログラムの実行性能を調べる機能として、プログラム特性情報(proginf 情報)が利用できます。

proginf 情報は、プログラムの実行時に環境変数 F_PROGINF を YES あるいは DETAIL と設定することによって採取できます(大阪大学では既定値として DETAIL が設定されています)。以下に、並列プログラム実行時の proginf 情報の出力例を示します。なお、図中の ※ は、並列処理使用時に表示される情報です。

```
   ******  Program Information  ******
Real Time (sec)              :          28.779867   経過時間 (秒)
User Time (sec)              :         115.000177   ユーザ時間 (秒)
Sys  Time (sec)              :           0.039784   システム時間 (秒)
Vector Time (sec)            :         112.660629   ベクトル命令実行時間 (秒)
Inst. Count                  :      47717311106.    全命令実行数
V. Inst. Count               :      24518067652.    ベクトル命令実行数
V. Element Count             :    6144579540224.    ベクトル命令実行要素数
FLOP Count                   :    3521968996499.    浮動小数点データ実行要素数
MOPS                         :       53632.776441   MOPS 値
MFLOPS                       :       30625.770224   MFLOPS 値
MOPS   (concurrent)          :      214494.483217   MOPS 値   (実行時間換算)          ※
MFLOPS (concurrent)          :      122482.168428   MFLOPS 値   (実行時間換算)        ※
A. V. Lenfth                 :         250.614348   平均ベクトル長
V. Op. Ratio (%)             :          99.623864   ベクトル演算率 (%)
Memory Size (MB)             :        1600.000000   メモリ使用量 (MB)
Max Concurrent Proc.         :                4.    最大同時実行可能プロセッサ数   ※
 Conc. Time(>= 1)(sec)       :          28.754953     1 台以上で実行した時間 (秒)  ※
 Conc. Time(>= 2)(sec)       :          28.753148     2 台以上で実行した時間 (秒)  ※
 Conc. Time(>= 3)(sec)       :          28.751822     3 台以上で実行した時間 (秒)  ※
 Conc. Time(>= 4)(sec)       :          28.742525     4 台以上で実行した時間 (秒)  ※
Event Busy Count             :                0.    イベントビジー回数             ※
Event Wait (sec)             :           0.000000   イベント待ち時間 (秒)          ※
Lock Busy Count              :                0.    ロックビジー回数               ※
Lock Wait (sec)              :           0.000000   ロック待ち時間 (秒)      ※
Barrier Busy Count           :                0.    バリアビジー回数               ※
Barrier Wait (sec)           :           0.000000   バリア待ち時間 (秒)      ※
MIPS                         :         414.932501   MIPS 値
MIPS (concurrent)            :        1659.446674   MIPS 値 (実行時間換算)         ※
I-Cache (sec)                :           0.005766   命令キャッシュミス (秒)
O-Cache (sec)                :           0.190763   オペランドキャッシュミス (秒)
Bank Conflict Time                                      バンクコンフリクト時間
 CPU Port Conf. (sec)        :           1.380597   CPU ポート競合 (秒)
 Memory Network Conf. (sec)  :          10.024346    メモリネットワーク競合 (秒)

Start Time (date)    :  Mon Jun 30 18:07:03 JST 2008   開始時刻(日付)
End   Time (date)    :  Mon Jun 30 18:07:10 JST 2008   終了時刻(日付)
```

図 4 並列処理使用時の proginf 情報

並列処理の性能を分析するときに重要な情報は、proginf 情報の Conc. Time です. 図 4 の例では、このプログラムが 4 つの CPU で約 29 秒づつ実行されたことが分かります。

この例では、Conc. Time がほぼ均等となっているので、うまく並列化されていると言えます。これが、以下の例

```
Conc. Time(>= 1)(sec):        74.154168
Conc. Time(>= 2)(sec):         8.549322
Conc. Time(>= 3)(sec):         8.292376
Conc. Time(>= 4)(sec):         8.071275
```

のように、Conc.Time(>=1)だけ実行時間が大きく残りが小さい場合は、プログラムが十分並列化できていないと言えます。このようなプログラムは、4. 2 で紹介した方法などを用いて、より多くのループが並列化されるようにする必要があります。

6. あとがき

以上、FORTRAN90/SX で利用可能な自動並列化機能について、簡単に説明いたしました。

並列化は、ベクトル化と異なり、すべてを自動まかせにしても、必ず性能が向上するというわけにはいきません。したがって、SX の並列処理方式を理解した上で、よりよい並列処理が行われるようにプログラムのチューニングを行うことが、性能向上には大変重要です。

本稿が、自動並列化利用の手助けになれば幸いです。

SX-9 FORTRAN90/SX の自動ベクトル化機能

横谷　雄司
日本電気株式会社　第一コンピュータソフトウェア事業部

概　要

SX-9 システムは豊富なベクトル演算命令を有しており，そのハードウェア性能を十分に引き出すためには，コンパイラの自動ベクトル化機能によりプログラムのベクトル化を行い，ベクトル命令によって処理される時間の割合(ベクトル化率)をできるだけ高くする必要がある．本稿では，SX-9 システムの FORTRAN90/SX コンパイラがもつ自動ベクトル化機能について，その特長と性能向上の観点について紹介する．

1. はじめに

SX-9 システムは 16 個の CPU をもつノードを複数結合した「スケーラブル・パラレル・スーパーコンピュータ」であり，そのハードウェア性能を十分に引き出すためには，

・複数の CPU を効率的に使用する，**並列化**

と共に，

・個々の CPU の中で効率的に計算を行う，**ベクトル化**

が大変重要となります．

FORTRAN90/SX は，ハードウェアに密着した高度な最適化，ベクトル化，並列化機能を有した Fortran95 コンパイラです．言語仕様としては，Fortran95(JIS X3001-1:1998)をサポートしています．

並列化につきましては別稿に譲り，本稿では，FORTRAN90/SX コンパイラがもつ**自動ベクトル化機能**についてご紹介いたします．

2. 自動ベクトル化機能

SX-9 システムでのベクトル化技法は，SX-8R システムの場合と基本的には同じですが，今回はスーパーコンピュータを初めて使われる方にもご理解いただけるよう，ベクトル化の基本概念からご紹介させていただきたいと思います．

2.1 ベクトル化の基本概念

通常の演算命令は，一度に一組のデータに対する演算処理を行います．(これを，ベクトル命令と対比させるために**スカラ命令**と呼びます．

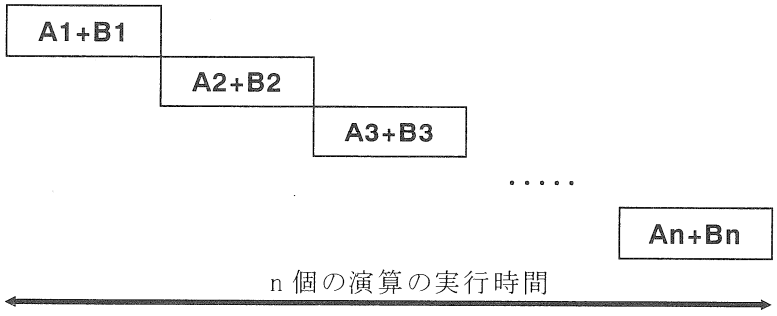


図 1　スカラ命令(スカラ加算)の実行イメージ

これに対して，ベクトル命令は，複数の組のデータに対する演算処理を一つの命令で一度に行うことができます

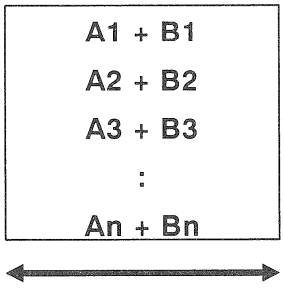


図 2 ベクトル命令(ベクトル加算)の実行イメージ

ループ中で計算される行列の要素など，規則的に並んだ配列データ(ベクトルデータ)に対してベクトル命令を適用することを自動ベクトル化と呼び，ベクトル化することによって高速な演算が可能となります．

2.2 自動ベクトル化の例

例えば，次の DO ループは，2 つの配列からのデータのロード，ベクトル加算，メモリへのストアの 4 つのベクトル命令で実行されます．

例 1 配列代入文と，ベクトル化されて生成される命令の概念

```
DO I=1,100
    C(I) = A(I) + B(I)
END DO
 ↓
VR1    ← 配列 A      (配列Aからベクトルレジスタにデータをロード)
VR2    ← 配列 B      (配列Bからベクトルレジスタにデータをロード)
VR3    ← VR1 + VR2   (ベクトル加算)
配列 C ← VR3         (配列 C に結果をストア)
                                  VRn:ベクトルレジスタ
```

2.3 ベクトル化の対象範囲

FORTRAN90/SX コンパイラは，Fortran プログラムの以下の範囲を対象として自動ベクトル化を行います．

表 1　FORTRAN90/SX 自動ベクトル化対象

ベクトル化の対象となるループ	配列式，DO ループ，DO WHILE ループ， FORALL ループ，IF 文と GOTO 文によるループ
ベクトル化の対象となる文	代入文，CONTINUE 文，GOTO 文， CYCLE 文，EXIT 文，IF 文，SELECT 構文 （CALL 文，入出力文　等は不可）
ベクトル化の対象となるデータの型	4 バイト／8 バイトの整数型・論理型 単精度／倍精度の実数型・複素数型 （文字型，4 倍精度，2 バイト整数型　等は不可）
ベクトル化の対象となる演算	加減乗除算，べき算，論理演算，関係演算， 型変換，組込み関数 （利用者定義演算，ポインタ代入　等は不可）

2.4 ベクトル化可能な条件

しかし，上記の範囲内であれば，どんなプログラムでもベクトル化できる訳ではありません．次の例を見てください．

例 2

```
DO I=1,99
    A(I) = 2.0                ! 代入文 2-1
    B(I) = A(I+1)             ! 代入文 2-2
END DO
```

図 3 に，このプログラムをベクトル化しない場合，ベクトル化した場合の演算の実行順序を示します．

ベクトル化しない場合の実行順序

```
A(1)=2.0      ! 代入文 2-1
B(1)-A(2)     ! 代入文 2-2
A(2)=2.0      ! 代入文 2-1
B(2)=A(3)     ! 代入文 2-2
:
A(99)=2.0     ! 代入文 2-1
B(99)=A(100)! 代入文 2-2
```

ベクトル化した場合の実行順序

```
A(1)=2.0      ! 代入文 2-1
A(2)=2.0      ! 代入文 2-1
                  :
A(99)=2.0     ! 代入文 2-1
B(1)=A(2)     ! 代入文 2-2
B(2)=A(3)     ! 代入文 2-2
:
B(99)=A(100)! 代入文 2-2
```

図 3　ベクトル化の有無による実行順序の相違

例2のループをベクトル化して実行すると，全ての要素に対して2-1の代入がベクトル命令で実行され，次に2-2の代入がやはり全ての要素に対して実行されます．このため，配列B(1)～B(98)の値は全て2.0となり，プログラムの意図と異なる結果となってしまいます．

また，N回目の繰り返しで定義したスカラ変数をN+1回目の繰り返しで引用する，例3のようなプログラムも，全ての要素に対して3-1の代入が先に実行されるため，ベクトル化すると配列A(1)～A(100)の値が全てゼロとなってしまいます．

例3

```
S=0.0
DO I=1,100
  A(I) = S                          ! 代入文 3-1（スカラ変数Sの引用）
  S = B(I)+C(I)       ! 代入文 3-2（スカラ変数Sの定義）
END DO
```

これらのプログラムのように，ベクトル化によって配列の定義・引用関係に変化が生じてしまう場合には，ベクトル化することができません．

FORTRAN90/SXの**自動ベクトル化機能**は，コンパイラがソースプログラムを解析して，ベクトル命令で実行できる部分を自動的に検出するとともに，必要ならベクトル化に適合するようにプログラムを変形して，その部分に対してベクトル命令を生成します．

2.5 配列構文のベクトル化

例2のプログラムは，DOループでしたが，配列構文ではどうでしょうか．

例4

```
A(1:99)-2.0                         ! 代入文 4-1
B(1:99)=A(2:100)       ! 代入文 4-2
```

配列構文はDOループと異なり，それぞれの文毎に全ての要素を演算することが，規格で定められています．即ち例4では，まず代入文4-1によってA(1:99)=2.0を全て実行してから，代入文4-2を実行します．これはすなわち図3に示した「ベクトル化した場合の実行順序」と同じです．したがって配列構文では，例2のように複数の文の前後関係によってベクトル化が阻害されることはありません．(もちろん他の要因でベクトル化できないことはあります．) なお，配列構文については「FORTRAN90/SX 言語説明書」の「1．1 代入文」および「3．5 式」を参照してください．

2.6 ベクトル長

個々のベクトル命令は，演算処理に入る前にある程度の準備処理が必要となります．この準備処理に要する時間を**立ち上がり時間**と呼びます．即ち，実際のベクトル演算に要する時間が余りにも小さいと，立ち上がり時間の影響が大きくなり，ベクトル化による高速化が行えません．

ベクトル化した場合とベクトル化しない場合とで実行時間が等しくなるループの繰り返し数(**ループ長**)を**交叉ループ長**と呼び，立ち上がり時間と交叉ループ長には，図4に示す関係があります．

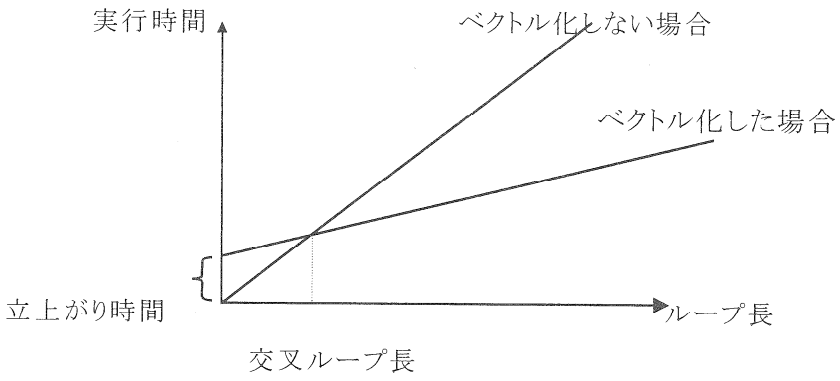


図 4　立上がり時間と交叉ループ長

この図を見ると，ループ長をできるだけ長くした方が，ベクトル化による高速化の効果が大きいことが，おわかりいただけると思います．

FORTRAN90/SX コンパイラの自動ベクトル化では，ループの繰り返し回数が 5 未満の場合には，ベクトル化による効果が少ないと判断し，ベクトル化していません．

3. 拡張ベクトル化機能

前述の通り，FORTRAN90/SX の自動ベクトル化機能は，ベクトル命令で実行可能な部分を検出しますが，そのままではベクトル化できない場合にプログラムを変形してベクトル化したり，プログラムを変形することによってベクトル化の効果をさらに高めたりします．これを**拡張ベクトル化機能**と呼びます．ここでは，FORTRAN90/SX が持つ種々の拡張ベクトル化機能のうち主なものを紹介します．

3.1 文の入れ換え

先ほどの例 2 は，「ベクトル化できない」と書きましたが，二つの文を入れ換えるとベクトル化できることに気づかれたでしょうか？

例 5　ソースプログラム

```
DO I=1,99
   A(I)=2.0      ! 代入文 5-1
   B(I)=A(I+1)   ! 代入文 5-2
 END DO
```

コンパイラによる変形

```
DO I=1,99
      B(I)=A(I+1)      ! 代入文 5-2
      A(I)=2.0         ! 代入文 5-1
END DO
```

このプログラムを右のように変形すると，ベクトル化した場合でも配列の定義・引用関係が保存されますので，FORTRAN90/SX コンパイラは自動的に 5-1 と 5-2 の文を入れ換えてベクトル化を行います．

3.2 ループの一重化

ループ長が長いほどベクトル化の効果が高いとご説明しましたが，例 6 のように多次元配列を 1 次元配列と見なすことができる場合には，多重ループを一重ループに変形することでループ長を拡大すると同時に，外側ループの繰り返し制御の時間を削減します．

例 6

```
DIMENSION A(M,N), B(M,N), C(M,N)
DO I=1,M
   DO J=1,N
      A(J,I) = B(J,I) + C(J,I)
```

```
   END DO
END DO
        ↓ コンパイラによる変形のイメージ
DO IJ=1,M*N
   A(IJ,1) = B(IJ,1) + C(IJ,1)
END DO
```

3.3 ループの入れ換え

多重ループの場合には，通常は最も内側のループをベクトル化します．しかし，ループを入れ換えることにより定義・引用関係の矛盾が解消されてベクトル化できるようになったり，内側のループよりも外側のループの方がループ長が長く，入れ換えた方が速いと判断した場合には，コンパイラがループを入れ換えてベクトル化を行います．

例 7　配列 A に依存関係がありベクトル化できない

```
DO J=1,M                          ! 外側ループ
   DO I=1,N                       ! 内側ループ
      A(I+1,J) = A(I,J) + B(I,J)
   END DO
END DO
```

↓

依存関係がなくなりベクトル化できる

```
DO I=1,N                ! 元の内側ループ
   DO J=1,M             ! 元の外側ループ
      A(I+1,J) = A(I,J) + B(I,J)
   END DO
END DO
```

3.4 部分ベクトル化

ループ構造や配列式に，ベクトル化できる部分とベクトル化できない部分が含まれている場合，

ベクトル化可能な部分と不可能な部分に分割し，可能な部分だけをベクトル化します．

例 8　4 倍精度の式が含まれているため，ベクトル化できない

```
REAL, DIMENSION(M) :: A, B, C
REAL(KIND=16), DIMENSION (M) :: Q
A(1:M) = B(1:M) * REAL(C(1:M) * Q(1:M))
```

↓

二つの配列式に分割することでベクトル化する．

作業用の配列 WK はコンパイラが用意する．

```
WK(1:M) = REAL(C(1:M) * Q(1:M))    この配列代入文はベクトル化不可
A(1:M) = B(1:M)*WK(1:M)            この配列代入文をベクトル化する
```

3.5 条件ベクトル化

配列の依存関係がベクトル化に適合しているかどうかコンパイル時に不明であったり，ループ長がコンパイル時に不明で，ベクトル化したほうが速いかどうかわからない場合に，ベクトル化したコードとベクトル化しないコードの両方を生成しておき，プログラムを実行するとき，そのどちらかを選択して実行するものです．

例 9 K が 0 以上であるか，または K が-10 未満ならばベクトル化しても配列 A の各要素の定義参照順序が変わらないのでベクトル化できる

```
DO I=N, N+10
    A(I) = A(I+K) + B(I)
END DO
      ↓     コンパイラによる変形のイメージ
IF (K .GE.0 .OR. K.LT.-10) THEN
    A(N:N+10) = A(N+K:N+10+K)+B(N:N+10)    ベクトル化する
ELSE
    DO I=N, N+10                           ベクトル化しない
      A(I) = A(I+K) + B(I)
    END DO
END IF
```

例 10 繰り返し回数がコンパイル時に不明なため，ベクトル化したほうが速いかどうか判断できない場合

```
A(1:N) = B(1:N) + C(1:N)
                ↓     コンパイラによる変形のイメージ
IF (N .GE. 5)  THEN
    A(1:N) = B(1:N) + C(1:N)        ベクトル化する
ELSE
  DO i=1, N
      A(i) = B(i) + C(i)    ベクトル化しない
  END DO
END IF
```

3.6 マクロ演算の認識

次のようなパターンは，変数や配列要素が繰り返しにまたがって定義・引用されるため，本来はベクトル化できませんが，コンパイラが特別なパターンであることを認識し，SX のハードウェアが持つ専用のベクトル命令を用いることで，ベクトル化を行います．

例 11 総和: ひとつ前の繰り返しで定義したスカラ変数 S の値を，次の繰り返しで引用するため，通常はベクトル化できないが，配列 A の総和を求めるパターンであるとコンパイラが認識して，総和ベクトル命令を用いることでベクトル化する

```
DO I=1, N
   S = S + A(I)
END DO
```

例 12 漸化式: 配列Aのひとつ前の繰り返しで定義された値を次の繰り返しで参照するので, 専用のベクトル命令を用いてベクトル化する

```
DO I=1, N
   A(I) = A(I-1) * B(I) + C(I)
 END DO
```

備考:もし A(I-1)=A(I)*B(I)+C(I)であれば, 前の繰り返しで定義した値を後の繰り返しでは参照しないため, そのままベクトル化できる.

例 13 最大値, 最小値を求める

```
DO I=1, N
   IF (XMAX .LT. X(I)) THEN
      XMAX = X(I)
   END IF
END DO
```

3.7 ループ融合

拡張ベクトル化機能ではありませんが, コンパイラは同じ形状(次元数と各次元のサイズ)を持つ複数の配列式, 同じ繰り返し回数を持つ複数のループ構造を一つにまとめてベクトル化します. これを**ループ融合**と呼びます.

例 14 次の二つの配列代入文は形状が一致しているので, 下の二重ループと同じように解釈, 最適化されてベクトル化される.

```
A(1:M, 1:N) = B(1:M, 1:N) + C(1:M, 1:N)
D(1:M, 1:N) = E(1:M, 1:N) * F(1:M, 1:N) + S
            ↓   コンパイラによる変形のイメージ
DO J=1,N
   DO I=1,M
      A(I, J) = B(I, J) + C(I, J)
      D(I, J) = E (I, J) * F(I, J) + S
   END DO
END DO
```

コンパイラは, 同じ形状の配列式・ループ構造が連続していれば融合しますが, 間に形状の異なる配列式・ループ構造や, 他の文があると融合できません.

高速化のためには, 出来るだけ同じ形状の配列式・ループ構造を連続させるようにしてください.

例 15　以下の3つの配列代入文は，二つの配列代入文の間に形状の異なる配列代入文があるために融合されない．

```
A(1:M, 1:N) = B(1:M, 1:N) + C(1:M, 1:N)
X(1:L) = 0.0
D(1:M, 1:N) = E(1:M, 1:N) * F(1:M, 1:N) + S
```

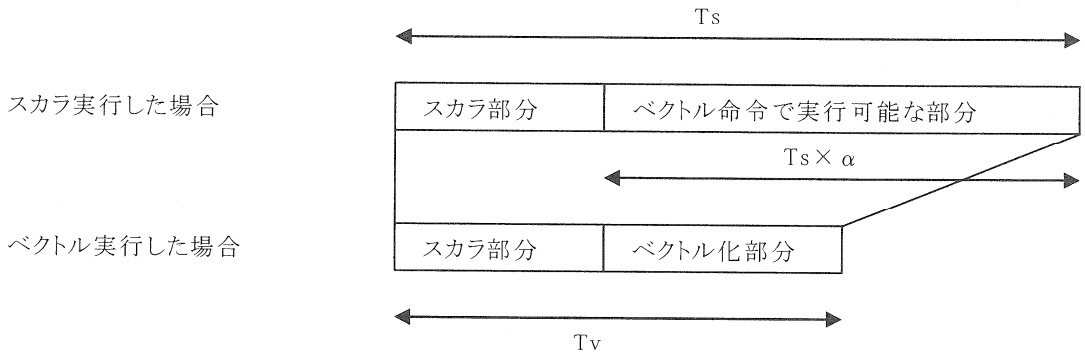
以下のように，文の順序を入れ換えて同じ形状の配列代入文が連続するように書き換えると，最初の2つの配列代入文がループ融合される

```
A(1:M, 1:N) = B(1:M, 1:N) + C(1:M, 1:N)
D(1:M, 1:N) = E(1:M, 1:N) * F(1:M, 1:N) + S
X(1:L) = 0.0
```

4. ベクトル化率向上のための手法

4.1 ベクトル化率

プログラムをスカラ命令だけで実行させた場合の実行時間に占める，ベクトル命令で実行可能な部分の時間の割合を**ベクトル化率**と呼びます．



α：ベクトル化率
Ts：スカラ実行したときの実行時間
Tv：ベクトル実行したときの実行時間

図 5　ベクトル化率

プログラムをベクトル化することにより実行性能が向上しますが，プログラムの一部だけをベクトル化しても，ベクトル化の効果はあまり期待できません．ベクトル化部分の実行時間が非常に小さくても，ベクトル化率が50%程度では，高々2倍の性能にしかならないことは，図5からもわかると思います．一般にはベクトル化率が90～95%以上ないと，ベクトル化による大きな性能向上は期待できません．

すなわち，ベクトル化による高速化技法の一つは，ベクトル化率を高め，100%に近づけることにあります．

しかし，一般にベクトル化率を正確に求めることは困難であるため，SX-9では，ベクトル化率に近い値として，プログラム特性情報(proginf)に表示される**ベクトル演算率**を用いています(5.1 proginf情報参照)．ベクトル演算率は，実行された命令の数をハードウェアがカウントすることで，プログラムで処理された全演算数に占める，ベクトル演算命令で処理された数の割合を求めたものです．

SX-9では，この**ベクトル演算率**を高くすることを目標に，チューニングを行ってください．

ここでは，ベクトル化率(ベクトル演算率)を向上させる手法の一つとして，コンパイラ指示行についてご紹介します．

4.2 コンパイラ指示行の挿入による高速化

コンパイラは，プログラムに対して自動的に最適なベクトル命令を生成しますが，例えば変数のもつ値のように，コンパイラがプログラムを解析してもわからない情報があると，必ずしも十分なベクトル化が行われるとは限りません．

このような場合に，コンパイラが知り得ない情報を利用者が与えることにより，ベクトル化の効果を一層促進させるものが，**コンパイラ指示行**です．

コンパイラ指示行は，

```
!CDIR オプション[,オプション]
*CDIR オプション[,オプション](ソースが固定形式の場合のみ)
```

のいずれかの形式で，!CDIR, *CDIR は，1 桁から 5 桁に書かなければなりません．

詳しくは「FORTRAN90/SX プログラミングの手引 3.2 コンパイラ指示行」を参照してください．

以下に，主な指示行のオプションとその使い方について説明します．

a) VECTOR/NOVECTOR

直後の配列式 または DO ループを自動ベクトル化の対象とする(VECTOR)か，対象としない(NOVECTOR)ことを指定します．

一般に VECTOR を指定する必要はありませんが，ベクトル長が小さく，ベクトル化しない方が効率の良いことがわかっていような場合に，NOVECTOR を指定します．

例 16 例えば「M は，1 又は 2 にしかなり得ない」ことを利用者が知っている場合，以下の様に NOVECTOR を指定してベクトル化を抑止したほうが効率がよい

```
!CDIR NOVECTOR
      A(1:M) = B(1:M) * C(1:M) + D(1:M) * E(1:M) - F(1:M) * G(1:M)
```

b) NODEP

配列の定義・引用関係がコンパイル時に不明であるため，コンパイラが自動ベクトル化できない場合に，利用者が「ベクトル化してもよい定義・引用関係であることを保証するので、ベクトル化するように」指示するものです．

例 17 NK の値が正であればベクトル化できる．条件ベクトル化で NK の値を判断するコードが出力されるが，「NK の値が常に正である」ことを利用者が知っている場合，NODEP を指定することにより無条件にベクトル化される．

```
!CDIR NODEP
      DO I=1, N
         A(I) = A(I + NK)
      END DO
```

例 18 IP(I)の値に重複するものが無ければベクトル化できるが，もし重複するものがある場合にはベクトル化できない．通常コンパイラにはどちらか判断できず，またこの場合には条件を判断する条件ベクトル化もできない．

もし利用者が，「IP(I)の値に重複するものがない」ことを知っている場合には，NODEPを指定することによってベクトル化することができる．

```
!CDIR NODEP
      DO I=1, N
         A( IP(I) ) = A( IP(I) ) + B(I)
      END DO
```

c) SELECT(VECTOR)

指定したDOループを優先してベクトル化することを指示します．

多重ループでベクトル化可能な DO ループが複数存在するとき、ベクトル化するループを指定したい場合に使用します。

例 19 ループの一重化ができない場合，コンパイラには変数 L,M,N の値がわからないので，通常最内側ループでベクトル化する．

しかし，例えばLの値がとても大きく，MやNの値がごく小さいことを利用者が知っているならば，次のようにSELECT(VECTOR)指示行を挿入することにより，右のように変形されて，ループ長の長いDO I=1,Lのループでベクトル化される．

```
!CDIR SELECT(VECTOR)
      DO I = 1, L
         DO J =1, M
            DO K=1,N
               A(K, J, I)=B(K, J, I)+C(K, J, I)
            END DO
         END DO
      END DO
                  ↓ コンパイラによる変形のイメージ
   DO J = 1, M
         DO K =1, N
            DO I = 1,L          ! このループがベクトル化される
               A(K, J, I)=B(K, J, I)+C(K, J, I)
            END DO
         END DO
   END DO
```

上の例で，もし，Mの値がとても大きく，LやNの値が小さいならば，DO J=1，M の直前に

```
!CDIR SELECT(VECTOR)
```

の指示行を挿入すればよいことは，もうおわかりのことと思います．

d) SHORTLOOP

SXシステムのベクトル演算命令では，一つの命令で一度に最大256要素のデータを処理することができます．それでは，処理したいデータの要素数が256個を超える場合にはどうでしょうか？この場合には次のように256回ずつの繰り返しを持つループ(例21では配列式)に分割してベクトル化を行います．

例20 繰り返し回数が256を超える配列式のベクトル化の概念

```
C(1:1000) = A(1:1000) + B(1:1000)
                ↓
DO I = 1,1000,256        ! コンパイラが生成するループ
     C(I:MIN(I+255,1000)) = A(I:MIN(I+255,1000) )+B(I:MIN(I+255,1000))
                          ! この配列代入文は常にベクトル化される
END DO
```

このようにコンパイラが生成するループを，**ストリップマイニングループ**と呼びます．

しかし，ループ長が常に256以下であることがわかっていれば，このストリップマイニングループの処理が不要となりますので，このための処理時間を削減できると同時に，レジスタを効率的に使用することができます．

SHORTLOOPは，ループ長が必ず256以下であることを指示します．

例21 変数Mの値が，常に256以下であることがわかっているならば，次の指示行を挿入することで，ストリップマイニングループを作成しないようにします．

```
!CDIR SHORTLOOP
      A(1:M) = B(1:M) + C(1:M)
```

FORTRAN90/SXコンパイラのコンパイラ指示行には，ここで紹介した他にもいろいろなオプションが指定でき，ベクトル化を制御することができます．

詳細につきましては，「FORTRAN90/SXプログラミングの手引 3.2コンパイラ指示行」をご参照ください．

5. 性能解析のツール

5.1 proginf情報

先に述べた，ベクトル演算率を始め，プログラムの実行性能を調べる最も簡単で有効な情報がプログラム特性情報(proginf情報)です．

proginf情報は，プログラムの実行時に，

```
setenv F_PROGINF YES
      または
setenv F_PROGINF DETAIL
```

を指定することで，表示されます(大阪大学では既定値としてDETAILが設定されています)．

図6は，setenv F_PROGINF DETAILで採取したproginf情報の例ですが，ベクトル演算率(図6の②)，

ベクトル長(図 6 の①)ともに非常に大きく，とても効率よくベクトル化されたプログラムでの例を示しています．

もしベクトル演算率が低ければ，コンパイラや後述の簡易性能解析機能の出力情報をもとに，ベクトル化できていないループ構造を抽出し，ベクトル化できるように変形したり，指示行を挿入するなどして，ベクトル演算率の向上を図ります．

次に，平均ベクトル長が短い場合には，2 重以上のループ構造を抽出し，SELECT(VECTOR)指示行を挿入してループ長の一番長いループがベクトル化されるようにすることで，ベクトル長を拡大します．また，ループ長が極端に短いループ構造がベクトル化されている場合には，NOVECTOR 指示行を挿入して，ベクトル化を行わないことも検討します．

```
       ******     プログラム  情報                  ******
       経過時間 (秒)                      :                      30.039067
       ユーザ時間(秒)                     :                      28.297595
       システム時間 (秒)                  :                       0.017765
       ベクトル命令実行時間 (秒)          :                      28.263599
       全命令実行数                       :                 11892009287.
       ベクトル命令実行数                 :                  6099457558.
       ベクトル命令実行要素数             :               1528191109757.
       浮動小数点データ実行要素数         :                880815869041.
       MOPS 値                            :                  54208.977057
       MFLOPS 値                          :                  31126.881228
  ①→  平均ベクトル長                     :                    250.545412
  ②→  ベクトル演算率(%)                  :                     99.622385
       メモリ使用量(MB)                   :                    512.031250
       MIPS 値                            :                    420.248061
       命令キャッシュミス(秒)             :                      0.002944
       オペランドキャッシュミス(秒)       :                      0.011593
       バンクコンフリクト時間
         CPU ポート競合 (秒)              :                      0.455945
         メモリネットワーク競合 (秒)      :                      2.416176

       開始時刻 (日付)    :  2008年  6月 30日 月曜日 17:57:42 JST
       終了時刻 (日付)    :  2008年  6月 30日 月曜日 17:58:12 JST
```

図 6 proginf 情報の出力例 (F_PROGINF=DETAIL)

5.2 簡易性能解析機能(ftrace)

proginf 情報では，プログラム全体のベクトル化率(ベクトル演算率)等を知ることができますが，手続き単位の性能を知ることはできません．

これに対して，簡易性能解析機能(ftrace 機能)を使用すれば，手続き単位や指定した範囲の性能情報を表示することができ，この情報を元にプログラムの性能上の問題点を調べることができます．

簡易性能解析機能の使い方を以下に示します．

1) プログラムを -ftrace オプションをつけてコンパイル・リンク

```
% sxf90 -ftrace test.f90
```

2) できた実行ファイルを実行

```
% a.out
```

3) ftrace コマンドを実行

```
% ftrace
```

このようにするだけで，第 7 図に示すように手続き単位の平均ベクトル長，ベクトル演算率などの性能情報が出力されます．

なお，3)の ftrace コマンドを実行する代わりに，2)の実行の前に環境変数 F_FTRACE に YES を設定しておくと，プログラムの実行が終了したときに，性能情報が自動的に表示されます(大阪大学では既

定値は設定されていません).

```
*--------------------------*
  FLOW TRACE ANALYSIS LIST
*--------------------------*

Execution : Mon Jun 30 17:58:12 2008                    ベクトル演算率    平均ベクトル長
Total CPU : 0:00'28"266                                      ↘          ↙
手続き名    CALL回数  実行時間
PROG.UNIT  FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS    MFLOPS V.OP  AVER.    VECTOR I-CACHE O-CACHE  BANK CONFLICT
                      TIME[sec](  %)    [msec]                    RATIO V.LEN      TIME    MISS    MISS CPU PORT NETWORK

sub5           1275    23.255( 82.3)    18.239  61136.0  35800.2 99.62 250.2    23.248  0.0007  0.0036  0.034    0.911
sub3            500     4.784( 16.9)     9.567  20427.7   8298.5 99.69 256.0     4.781  0.0002  0.0010  0.061    1.840
sub7             24     0.147(  0.5)     6.124  72093.7  50785.4 99.93 250.3     0.146  0.0000  0.0005  0.001    0.006
main              1     0.039(  0.1)    39.140  47202.8   4145.6 99.52 256.0     0.029  0.0008  0.0040  0.000    0.001
sub3              9     0.031(  0.1)     3.432  49334.3  21004.3 99.79 256.0     0.031  0.0000  0.0000  0.002    0.013
sub6             24     0.007(  0.0)     0.301  67647.3  37418.1 99.57 256.0     0.007  0.0000  0.0000  0.000    0.002
sub4              1     0.003(  0.0)     2.501  43833.8  21620.6 99.80 250.3     0.002  0.0000  0.0000  0.000    0.000
sub10             1     0.001(  0.0)     0.892    150.6      1.8  0.96   7.6     0.000  0.0001  0.0002  0.000    0.000
sub8              1     0.001(  0.0)     0.563    150.5      0.8 39.48 244.1     0.000  0.0002  0.0002  0.000    0.000
sub1              1     0.000(  0.0)     0.245  18648.4      0.0 98.46 256.0     0.000  0.0000  0.0000  0.000    0.000
sub9              1     0.000(  0.0)     0.004    175.4     70.7 58.62  36.0     0.000  0.0000  0.0000  0.000    0.000
-------------------------------------------------------------------------------------------------------------------------
total          1838    28.267(100.0)    15.379  54268.1  31160.9 99.62 250.5    28.245  0.0020  0.0094  0.098    2.773
```

図 7 ftrace コマンドの出力例

より詳しい使い方については,「FORTRAN90/SX プログラミングの手引 10.3 簡易性能解析機能」をご参照ください.

6. おわりに

以上, FORTRAN90/SX コンパイラの自動ベクトル化機能を中心にご紹介させていただきました. FORTRAN90/SX コンパイラは, 他にも本稿でご紹介できなかった種々のベクトル化機能を持っています. 詳細につきましては,「FORTRAN90/SX プログラミングの手引」を参照してくださるようお願いいたします.

皆様が SX-9 と FORTRAN90/SX を使っていただくうえで, 本稿が, 多少なりともお役に立てれば幸いです.

センター報告

Supercomputing Contest (SuperCon)

Programming Contest for High School Students

スーパーコンピューティング・コンテスト　SuperCon2008

SuperCon は、1995 年より始まったプログラミングコンテストです。予選を通過した 10 組のチーム（1 チーム 2〜3 名）が大阪大学、東京工業大学の会場に分かれ、スーパーコンピュータを使ったプログラミングを行います。数日間かけて本選課題の問題を解くプログラムを作成し、作成最終日に提出されたプログラムの正確さ・速度を審査委員会が評価し、コンテスト最終日の成果報告会で発表します。本選課題には、科学技術の様々な分野から最先端の話題が選ばれ、それを高校生にもわかりやすい問題にして、挑戦してもらいます。

第 14 回となる今年は下記の日程で行われました。

予選課題発表：平成 20 年 6 月 2 日（月）
予選応募期間：平成 20 年 6 月 2 日（月）
〜 6 月 23 日（月）
予選通過者発表：平成 20 年 6 月 25 日（水）
本　選　期　間：平成 20 年 7 月 28 日（月）
〜8 月 1 日（金）

優勝は筑波大付属駒場 potassio チーム、準優勝は東京工業大学付属科学技術高等学校 Fungaa チーム、３位には広島学院高等学校 Maruchan チームが入賞しました。その他、学会賞として久留米高等工業専門学校　Kurofune チームが選ばれました。

大阪大学サイバーメディアセンター

平成２０年度 スーパーコンピュータシンポジウム

日時：２００８年１０月２４日（金）１０：３０～１７：４０
会場：大阪大学レーザーエネルギー学研究センター
主催：大阪大学３センター共催
サイバーメディアセンター（CMC）
レーザーエネルギー学研究センター（ILE）
核物理研究センター（RCNP）

プログラム

10:30 ～10:35	菊池 誠（大阪大学サイバーメディアセンター） はじめに
10:35 ～11:00	東田 学（大阪大学サイバーメディアセンター） SX-９の導入と大規模計算機システムについて
11:00 ～11:20	西原 功修（大阪大学レーザーエネルギー学研究センター） Workflow、XML-DB機能を有する Webポータル型シミュレーション支援システム
11:20 ～11:40	駒 佳明（沼津工業高等専門学校） SXで探る重いクォークの物理
11:40 ～12:00	小林 泰三（九州大学） 基盤センターにおけるグリッド連携
	昼食（別紙：学内お食事処 参照）
13:20 ～13:40	日本電気 SX-９のご紹介
13:40 ～14:00	小林 広明（東北大学） 実アプリーケーションを用いたSX-９の性能評価
14:00 ～14:20	柳澤 将（大阪大学産業科学研究所） 第一原理分子動力学計算による有機/金属界面のシミュレーション
14:20 ～14:40	吉田 尚史（信州大学） 流体大規模計算とスーパーコンピュータ
	休憩
14:55 ～15:15	城崎 知至（大阪大学レーザーエネルギー学研究センター） 高速点火レーザー核融合の多階層連結シミュレーション
15:15 ～15:35	梶島 岳夫（大阪大学工学研究科） FEMまたはVOF法を組み込んだ埋込境界法による可変型境界を有する多相流の解析
15:35	谷口 億宇（大阪大学核物理センター）

～15:55	原子核の構造変化
15:55 ～16:55	パネルディスカッション ■　－－－ スーパーコンピュータの将来 －－－ ・サイバーメディアセンターへの期待 ・利用者と管理者の役割 ・ベクトル化・並列化とアプリケーション ■モデレータ 外川 浩章 大阪大学核物理研究センター ■パネリスト 江川 隆輔 東北大学サイバーサイエンスセンター 松古 栄夫 高エネルギー加速器研究機構　計算科学センター 長友 英夫 大阪大学レーザー研 東田　学 大阪大学 CMC
16:55 ～17:00	竹村 治雄（大阪大学サイバーメディアセンター センター長） おわりに
17:15 ～ 17:40	サイバーメディアセンター見学
17:40～	懇親会

平成 20 年 10 月 24 日、レーザーエネルギー学研究センターにて、2008 年度スーパーコンピュータシンポジウムが開催されました。

今回は主に2008年7月に導入した大阪大学サイバーメディアセンター(CMC)　スーパーコンピュータシステム SX-9 のお披露目と利用者の情報交換を目的としたものです。

はじめに、サイバーメディアセンター菊池教授より開会の辞の後、導入済みの SX-8、PC クラスターに加え、今回導入した SX-9 を中心とした CMC の HPC システムの紹介や他機関とのグリッド連携についてサイバーメディアセンターから報告し、また、SX-9 については、NEC 社、先行導入した東北大学から実環境での性能評価について講演していただきました。一方、利用者からは、本システムを用いた研究の成果について、運用への要望も含め発表していただきました。

午後からは、スーパーコンピュータの将来について、利用者及び管理者でのパネルディスカッションがあり、活発な意見交換が行われました。

参加者は合計 58 名、盛況に実施されました。

平成20年度夏季利用者講習会アンケート　集計結果

◎受講者数、アンケート回収数等

講習会名	開催日時	申込者数	受講者数			アンケート回収数
				学内	学外	
AVS可視化処理入門 【MicroAVS入門】	8月28日 14:00～17:00	4	4	4		4
AVS可視化処理応用 【AVS＆FORTRANプログラミング入門】	8月29日 10:00～17:00	10	9	9		9
スーパーコンピュータ初心者入門	9月 1日 13:30～17:00	22	20	13	7	20
スーパーコンピュータ 【ベクトル化、並列化編】	9月 2日 9:30～12:00	10	10	10		10
スーパーコンピュータ 【パフォーマンス・チューニング編】	9月2日 13:00～15:00	10	7	7		7
IDL利用入門	9月3日 13:00～17:00	9	7	7		7
合　　計		65	57	50	7	57

[問1] あなた(参加者)は、◆本センターの利用者ですか？ ◆所属は？ ◆職種は？

講習会名	利用者			所属		職名						
	はい	いいえ	無回答	学内	学外	教員	技術職員	事務職員	院生	学生	研究生	その他
AVS可視化処理入門 【MicroＡＶＳ入門】	3	1		4		1			2			1
AVS可視化処理応用 【ＡＶＳ＆FORTRANプログラミング入門】	8	1		9		1	2		4	1		1
スーパーコンピュータ初心者入門	8	11	1	13	7		2		5	12		1
スーパーコンピュータ 【ベクトル化、並列化編】	7	2	1	10			1		5	2		2
スーパーコンピュータ 【パフォーマンス・チューニング編】	7			7					4	1		2
ＩＤＬ利用入門	5	1	1	7		1			3	1		2
合　　計	38	16	3	50	7	3	5	0	23	17	0	9

[問2, 3]

講習会名	[問2] どういう方法でこの講習会を知りましたか？(複数回答可)							[問3] 開催日時は適当ですか？			
	速報	センター掲示のポスター	その他掲示のポスター	WWW	研究室の教員	研究室の知人	その他	適当	普通	不適当	わからない
AVS可視化処理入門 【MicroＡＶＳ入門】	2	2						2		1	1
AVS可視化処理応用 【ＡＶＳ＆FORTRANプログラミング入門】	3	2				2	2	7		1	1
スーパーコンピュータ初心者入門	4	1		1	9	1	3	12	2	4	1
スーパーコンピュータ 【ベクトル化、並列化編】	3	3			1	1	1	7	1	1	
スーパーコンピュータ 【パフォーマンス・チューニング編】	1	2				1	2	6		1	
ＩＤＬ利用入門	1				4	1	1	6	1		
合　　計	14	10	0	1	14	6	9	40	4	8	3

[問4, 5] 内容等

講習会名	[問4] 希望した内容に満足しましたか？					[問5] これからの研究に役立ちますか？				
	大変満足	満足	普通	悪い	わからない	大変役立つ	役立つ	普通	役立たない	わからない
AVS可視化処理入門【MicroＡＶＳ入門】	3	1				2	2			
AVS可視化処理応用【ＡＶＳ＆FORTRANプログラミング入門】	5	4				3	6			
スーパーコンピュータ初心者入門	4	4	11			2	8	4		5
スーパーコンピュータ【ベクトル化、並列化編】	2	5	2			3	3	2		
スーパーコンピュータ【パフォーマンス・チューニング編】	4	2	1			3	3	1		
ＩＤＬ利用入門	1	3	2			1	5			1
合　　計	19	19	16	0	0	14	27	7	0	6

[問6, 7] 内容等

講習会名	[問6] 講習内容は理解できましたか？				[問7] 講習会資料は適当で満足しましたか？			
	よく理解できた	大体理解できた	まったく理解できなかった	無回答	大変満足	満足	普通	悪い
AVS可視化処理入門【MicroＡＶＳ入門】	2	2			2	2		
AVS可視化処理応用【ＡＶＳ＆FORTRANプログラミング入門】	4	5			4	5		
スーパーコンピュータ初心者入門	3	12	2	2	1	8	9	
スーパーコンピュータ【ベクトル化、並列化編】	5	2			2	6		
スーパーコンピュータ【パフォーマンス・チューニング編】	1	5	1		3	2	1	
ＩＤＬ利用入門	3	4			1	1	1	
合　　計	18	30	3	2	13	24	11	0

[問8] 利用予定

講 習 会 名	[問8] 差し支えなければ、どのような対象にご利用されるでしょうか？		回答内容
	回答	無回答	
AVS可視化処理入門 【MicroＡＶＳ入門】	3	1	流体計算の可視化、CT画像の再構築と実験データのマッピング、熱・流体計算
AVS可視化処理応用 【ＡＶＳ＆FORTRANプログラミング入門】	4	5	CT画像の再構築、プラズマの粒子のふるまいや輻射計算の可視化、CFD
スーパーコンピュータ初心者入門	6	14	放射線治療計画におけるモンテカルロシミュレーション、CFD、流体の数値解析、Ilestaを利用、レーザープラズマ発生時の輻射計算
スーパーコンピュータ 【ベクトル化、並列化編】	6	4	電磁波シミュレーション、レーザープラズマ輻射計算、流体解析、CFD
スーパーコンピュータ 【パフォーマンス・チューニング編】	6	1	電磁波シミュレーション、レーザープラズマ輻射計算、流体解析、CFD
ＩＤＬ利用入門	5	2	レーザープラズマの輻射計算などの可視化、リモートセンシングデータ、月周回衛星「かぐや」のデータ解析、赤外線天文衛星「あかり」の解析、標高マップの画像処理
合　　計	30	27	

[問9]

別紙「平成20年度夏季利用者講習会アンケート 集計結果([問4～7,9,11,15]に対する意見等)」に記載

[問10, 12, 13,14] 内容等

講習会名	[問10] あなたは、スーパーコンピュータを使ったことがありますか？			[問12] 今後、大規模計算機システムのスーパーコンピュータの科学技術計算を利用されますか？			[問13] 今後、大規模計算機システムのアプリケーションサーバの科学技術計算を利用されますか？			[問14] 今後、大規模計算機システムのアプリケーションを利用されますか？		
	ある	ない	その他	する	しない	その他	する	しない	その他	する	しない	その他
AVS可視化処理入門 【MicroＡＶＳ入門】		4		2	1	1	3		1	3		1
AVS可視化処理応用 【ＡＶＳ＆FORTRANプログラミング入門】	3	6		3	6		7	1	1	7	1	1
スーパーコンピュータ初心者入門	3	15		3	15		5	7	4	6	6	4
スーパーコンピュータ 【ベクトル化、並列化編】	4	4		4	4		5	1	2	5	1	2
スーパーコンピュータ 【パフォーマンス・チューニング編】	4	3		4	3		4	2	1	5	1	1
ＩＤＬ利用入門	1	6		1	6		1	4	1	1	4	1
合　　計	15	38	0	17	35	1	25	15	10	27	13	10

[問11]

別紙「平成20年度夏季利用者講習会アンケート 集計結果([問4～7,9,11,15]に対する意見等)」に記載

平成２０年度夏季利用者講習会　アンケート集計結果

（[問 4～7, 9, 11, 15]に対する意見等）

講習会名：ＡＶＳ可視化処理入門【ＭｉｃｒｏＡＶＳ入門】
（講師：(株) ケイ・ジー・ティー)

[問４] 希望した内容に満足しましたか？
・満足できた点
回答なし。
・悪い点
回答なし。

[問６] 講習内容は理解し満足できましたか？
・よく理解できた（満足できた）点
ファイル形式など自分のデータの可視化法が分かり易かった。
データの読み込み、全般。

[問９]今後、どのような機能の追加・改良を行えば、満足して利用できると思われますか。
マウスのセンターホイールを使って、スクロールバーを動かせるようになると便利。

[問１５] その他、ご意見・ご要望があれば、些細なことでも結構ですのでご記入ください。
（ソフトウェア、ハードウェア、通信関係)。
大変分かり易かったです。

講習会名：ＡＶＳ可視化処理応用【ＡＶＳ＆ＦＯＴＲＡＮプログラミング入門】
（講師：(株) ケイ・ジー・ティー)

[問４] 希望した内容に満足しましたか？
・満足できた点
研究に応用できそうな操作やモジュールの使い方を学べた。
簡単なモデルにはすぐに適用できそう。
基本的なことから説明してもらえた。

[問６]講習内容は理解し満足できましたか？
・よく理解できた（満足できた）点
全体の流れが理解できたので、自分のデータで活用できる状態になった。
ウィンドウの表示・非表示。
ソフトウェアレンダラ、ムードウェアレンダラの切替え。
テキストがまとまって分かりやすくなりました。
実習が分かりやすかった。

[問９]今後、どのような機能の追加・改良を行えば、満足して利用できると思われますか。
数値の入力欄間を Tab キーで移動できるようにしてほしい。
並列計算で出力されたファイルの可視化と動画ファイルへの出力。
時系列データのループ。

[問１５] その他、ご意見・ご要望があれば、些細なことでも結構ですのでご記入ください。
（ソフトウェア、ハードウェア、通信関係)。
後半の実習中に一度何かを失敗してしまうと、あたふたしてしまって内容が理解しにくかった。
講習会で使用するパソコンの性能を上げてほしい。

講習会名：スーパーコンピュータ初心者入門
（講師：レーザーエネルギー学研究所職員、研究系システム班職員)

[問４] 希望した内容に満足しましたか？
・満足できた点

わかりやすかった。
NQS スケジュールをさわることができた。
[問６] 講習内容は理解し満足できましたか？
・よく理解できた（満足できた）点
会話型・バッチ型について。
スーパーコンピュータの利用について。
システム構成図での説明が大変よくわかりました。
全体的に理解できた気がします。
コンパイルに気をつけないといけないこと。
・全く理解できなかった（改善すべき）点
スーパーコンピュータにプログラムを投入するところ。
[問９]今後、どのような機能の追加・改良を行えば、満足して利用できると思われますか。
PC クラスタを増設してほしい。
プロジェクタに表示されているターミナルのフォントサイズを大きくしてほしい。
[問１５] その他、ご意見・ご要望があれば、些細なことでも結構ですのでご記入ください。
（ソフトウェア、ハードウェア、通信関係）。
実際にパソコンを触るとわかりやすいです。
４～６月頃にも開催してほしい。
例を多くみせてほしい。
C 言語は使った人なら分かるが、経験がない人には少し難しい。
知らない単語ばかりで、何の話をしているかわからなくなった。

講習会名：スーパーコンピュータ【ベクトル化、並列化編】
（講師：レーザーエネルギー学研究所職員，研究系システム班職員）
[問４] 希望した内容に満足しましたか？
・満足できた点
ベクトル化の重要性が認識できた。
[問６] 講習内容は理解し満足できましたか？
・よく理解できた（満足できた）点
ベクトル化と並列化の概要が把握できた点。
ベクトルとスカラーの違い。
コンパイラがいかに賢いか。
[問９] 今後、どのような機能の追加・改良を行えば、満足して利用できると思われますか？
コンパイラのエラーで表示される行にファイルの行も表示してほしい。
[問１１] スーパーコンピュータではどんなライブラリをご使用ですか。
ASL
数学ライブラリ（Lapack、Blas）
[問１５] その他、ご意見・ご要望があれば、些細なことでも結構ですのでご記入ください。
（ソフトウェア、ハードウェア、通信関係）。
細かいエラーメッセージの説明は必要ないと思う。
大変わかりやすかったです。

講習会名：スーパーコンピュータ【パフォーマンス･チューニング編】
（講師：日本電気）
[問４] 希望した内容に満足しましたか？
・満足できた点
PROGINF 情報の見方がわかった。
[問６] 講習内容は理解し満足できましたか？
・よく理解できた（満足できた）点
並列計算の利点。

・全く理解できなかった（改善すべき）点
専門用語ばかりでついていけなかった。
[問１１] スーパーコンピュータではどんなライブラリをご使用ですか。
ASL
[問１５] その他、ご意見・ご要望があれば、些細なことでも結構ですのでご記入ください。
（ソフトウェア、ハードウェア、通信関係）。
かなり難しいように感じました。慣れるのに時間がかかりそうです。
先輩のソースを見て分からなかった点が分かりました。

講習会名：ＩＤＬ利用入門
（講師：ジクー・データシステムズ）
[問４] 希望した内容に満足しましたか？
・満足できた点
簡単なさわりかたが理解できたと思う。
プログラミングについての説明を聞けた点。
聞きたい内容であった。
[問６] 講習内容は理解し満足できましたか？
・よく理解できた（満足できた）点
プログラミングについて学生達のよい入門コースとなった。
Itools の使用法、コマンドの基礎。
[問１５] その他、ご意見・ご要望があれば、些細なことでも結構ですのでご記入ください。
（ソフトウェア、ハードウェア、通信関係）。
実習で使うパソコンの性能を上げてほしい。

大規模計算機システム　Q＆A・LINK集

1．パスワード変更をしたいのですが？

https://portal.hpc.cmc.osaka-u.ac.jp/ にWebからアクセスし、ログインID、パスワードの入力を行って、新パスワードを指定します。この時、MAC O/Sを利用されていれば、FirefoxかNetscapeでご利用ください。

2．研究室のパソコンからスパコンを利用したいのですが、どうすれば良いのでしょうか？

下記のURLにスーパーコンピュータ利用入門の講習会資料がありますので、ご覧ください。

http://www.hpc.cmc.osaka-u.ac.jp/j/tebiki/sx-kiso.pdf

3．大規模計算機システムにログインする方法を知りたいのですが？

下記のURLをご覧ください。接続方法がわかります。

http://www.hpc.cmc.osaka-u.ac.jp/j/tebiki/login.html

4．大規模計算機システムにファイル転送を行いたいのですが？

WinSCPを使用してファイル転送を行います。下記のURLを参照ください。

http://www.hpc.cmc.osaka-u.ac.jp/j/tebiki/sx5-nyumon.pdf

5．大規模計算機システムの負担金一覧表を知りたいのですが？

下記のURLをご覧ください。負担金一覧表がわかります。

http://www.hpc.cmc.osaka-u.ac.jp/j/futankin/index.html

6．大規模計算機システムの試用制度及び試用制度の申請をしたいのですが？

下記のURLをご覧ください。試用制度の申し込み及び内容を知ることができます。

http://www.hpc.cmc.osaka-u.ac.jp/j/futankin/shiyou.html

7．大規模計算機システムの申請関係の情報を知りたいのですが？

下記のURLをご覧ください。申請書及び内容を知ることができます。

http://www.hpc.cmc.osaka-u.ac.jp/j/shinsei/forms.html

8．大規模計算機システムを利用する場合の資格を知りたいのですが？

下記のURLをご覧ください。利用資格を知ることができます。

http://www.hpc.cmc.osaka-u.ac.jp/j/shikaku/index.html

9．大規模計算機システムのサポート情報を知りたいのですが？

下記のURLをご覧ください。種々の情報等を掲載していますので、ご覧ください。

http://www.hpc.cmc.osaka-u.ac.jp/j/support/inquiry.html　問合せ先情報

http://www.hpc.cmc.osaka-u.ac.jp/j/support/advisor.html　利用相談員情報

１０．他の６大学情報基盤センターの情報を知りたいのですが？

下記の URL をご覧ください。他大学情報基盤センターへリンクしています。

http://www.cmc.osaka-u.ac.jp/j/intro/link.html

１１．　Ｗｅｂ上のマニュアルを参照できませんか？

下記の URL から大規模計算機システムのポータルにログインするとスーパーコンピュータシステムなどのマニュアル参照する事ができます。

https://portal.hpc.cmc.osaka-u.ac.jp/

2008年度大規模計算機システム利用相談員

氏名	所属	職名	相談形態
高木　達也	大阪大学 大学院薬学研究科	教授	メール
山井　成良	岡山大学 総合情報基盤センター	教授	メール
武知　英夫	阿南工業高等専門学校 機械工学科	准教授	メール
柳澤　　将	大阪大学 産業科学研究所	特任研究員	メール （木曜日 14時～16時）

＊委嘱期間 2008.5.1～2009.3.31

2008年度大規模計算機システム利用指導員

氏名	所属	職名
板野　智昭	関西大学 システム理工学部	講師
藤　堅正	近畿大学 理工学部	講師
武知　英夫	阿南工業高等専門学校 機械工学科	准教授

＊委嘱期間 2008.5.1～2009.3.31

利用規程等

(規程)

(附表)

・規程関係

大阪大学サイバーメディアセンター大規模計算機システム利用規程

第1条　この規程は、大阪大学サイバーメディアセンター(以下「センター」という。)が管理・運用する全国共同利用のスーパーコンピュータシステム及びワークステーションシステム(以下「大規模計算機システム」という。)の利用に関し必要な事項を定めるものとする。

第2条　大規模計算機システムは、学術研究及び教育等のために利用することができるものとする。

第3条　大規模計算機システムを利用することのできる者は、次の各号のいずれかに該当する者とする。
(1)　大学、短期大学、高等専門学校又は大学共同利用機関の教員(非常勤講師を含む。)及びこれに準ずる者
(2)　大学院の学生
(3)　学術研究及び学術振興を目的とする国又は地方公共団体が所轄する機関に所属し、専ら研究に従事する者
(4)　学術研究及び学術振興を目的とする機関(前号に掲げる機関を除く。)で、センターの長(以下「センター長」という。)が認めた機関に所属し、専ら研究に従事する者
(5)　科学研究費補助金の交付を受けて学術研究を行う者
(6)　前各号のほか、特にセンター長が適当と認めた者

第4条　大規模計算機システムを利用しようとする者は、所定の申請を行い、センター長の承認を受けなければならない。
2　前項の申請は、大規模計算機システム利用の成果が公開できるものでなければならない。

第5条　センター長は、前条第1項による申請を受理し、適当と認めたときは、これを承認し、登録番号を与えるものとする。
2　前項の登録番号の有効期間は、1年以内とする。ただし、当該会計年度を超えることはできない。

第6条　大規模計算機システムの利用につき承認された者(以下「利用者」という。)は、申請書の記載内容に変更を生じた場合は、速やかに所定の手続きを行わなければならない。

第7条　利用者は、第5条第1項に規定する登録番号を当該申請に係る目的以外に使用し、又は他人に使用させてはならない。

第8条　利用者は、当該申請に係る利用を終了又は中止したときは、速やかにその旨をセンター長に届け出るとともに、その利用の結果又は経過を所定の計算機利用報告書によりセンター長に報告しなければならない。
2　前項の規定にかかわらず、センター長が必要と認めた場合は、計算機利用報告書の提出を求めることができる。

第9条　利用者は、研究の成果を論文等により公表するときは、当該論文等に大規模計算機システムを利用した旨を明記しなければならない。

第10条　利用者は、当該利用に係る経費の一部を負担しなければならない。

第11条　前条の利用経費の負担額は、国立大学法人大阪大学諸料金規則に定めるところによる。

第12条　前条の規定にかかわらず、次の各号に掲げる場合については、利用経費の負担を要しない。
(1)　センターの責に帰すべき誤計算があったとき。
(2)　センターが必要とする研究開発等のため、センター長が特に承認したとき。

第13条　利用経費の負担は、次の各号に掲げる方法によるものとする。
(1)　学内経費(科学研究費補助金を除く。)の場合にあっては、当該予算の振替による。
(2)　前号以外の場合にあっては、本学が発する請求書の指定する銀行口座への振込による。

第14条　センター長は、この規程又はこの規程に基づく定めに違反した者その他大規模計算機システムの運営に重大な支障を生じさせた者があるときは、利用の承認を取り消し、又は一定期間大規模計算機システムの利用を停止させることがある。

第15条　この規程に定めるもののほか、大規模計算機システムの利用に関し必要な事項は、センター長が定める。

附　則
1　この規程は、平成12年4月1日から施行する。
2　大阪大学大型計算機センターの利用に関する暫定措置を定める規程(昭和43年9月18日制定)は、廃止する。
3　この規程施行前に大阪大学大型計算機センターの利用に関する暫定措置を定める規程に基づき、平成12年度の利用承認を受けた利用者にあっては、この規程に基づき利用の登録があったものとみなす。

附　則
この改正は、平成13年1月6日から施行する。

附　則
この改正は、平成13年4月1日から施行する。

附　則
この改正は、平成14年4月1日から施行する。

附　則
この改正は、平成14年6月19日から施行し、平成14年4月1日から適用する。

附　則
この改正は、平成15年4月1日から施行する。

附　則
この改正は、平成16年4月1日から施行する。

附　則
この改正は、平成18年2月15日から施行する。

附　則
この改正は、平成19年9月28日から施行する。

大規模計算機システム利用負担額一覧

（平成20年4月1日現在）

区分	計算機資源のシェア値	スーパーコンピュータ		ファイル利用の制限	年間負担額（後期利用は半額）
		並列実行CPU数	メモリ制限		
基本負担額	1	4	16GB	10GB	0円（備考７）
	1	4	16GB	10GB	1万円
	10	4	32GB	300GB	10万円
	50	8	制限なし	1TB	50万円
	100	制限なし	制限なし	2TB	100万円
	260	制限なし	制限なし	3TB	200万円
	450	制限なし	制限なし	4TB	300万円
ファイル追加オプション	ファイル追加50GBにつき				1万円
消費税額	上記負担額で算出した合計額に100分の5を乗じて得た額				

備考

1 基本負担額は年度の最初の登録時に算出する。

2 基本負担額の制限内でスーパーコンピュータ、クラスタシステム、ファイルなど計算機資源を利用できる。なお、スーパーコンピュータ、クラスタシステムにおけるCPU・メモリなどの計算機資源は、フェアシェアスケジュール機能により設定したシェア値に応じて割り当てられる。

3 基本負担額1万円の場合、登録者数は1名とする。その他の場合、登録者数は特に制限を設けない。

4 後期（10～3月）利用の基本負担額及びファイル追加オプションは、年間負担額の半額とする。

5 上記の基本負担額以外に50万円単位での申請を1,000万円を上限として受け付ける。その場合のシェア値及びファイルシェア利用の制限の設定については以下のとおりとする。
シェア値は300万円未満が基本負担額の1.3倍、300万円以上が基本負担額の1.5倍とする。
ファイル利用の制限は、50万円につき0.5TBを加算する。

6 ファイルサーバはファイル使用量の制限内で利用できる。なお、制限値以上の利用は50GB単位での追加オプションとする。

7 別に定める試用制度による利用を認められた者は、基本負担額1万円の場合と同じ資源を、登録のあった月から、前期（4月～9月）3ヶ月間、又は後期（10月～3月）1ヶ月間無料で利用できる。ただし、当該会計年度を超えての利用はできないものとする。

8 大学院の学生が基本負担額1万円で利用する場合、負担額を半額とする優遇措置を受けられる。

9 企業利用者は、科学研究費補助金及び共同研究プロジェクトでの利用を除き負担額を3倍の設定とする。

10 先端研究施設共用イノベーション事業に係る利用期間は四半期単位とする。なお、負担額は前項の年間負担額の1/4の設定とする。

大阪大学サイバーメディアセンター大規模計算機システム試用制度利用内規

第1条 この内規は、大阪大学サイバーメディアセンター（以下「センター」という。）が管理運用する全国共同利用のスーパーコンピュータシステム及びワークステーション（以下「大規模計算機システム」という。）の試用制度を利用するための必要な事項を定める。

第2条 試用制度は、初めてセンターの大規模計算機システムを利用する者（以下「利用者」という。）に一定の期間利用させることによって、同システム利用についての知識の向上と教育研究活動と学習に役立てることを目的とする。

第3条 試用制度を利用することができる者は、次の各号のいずれかに該当するものとする。
(1) 大学、短期大学、高等専門学校又は大学共同利用機関の教員（非常勤講師を含む。）及びこれに準ずる者
(2) 大学院の学生
(3) 学術研究及び学術振興を目的とする国又は地方公共団体が所轄する機関に所属し、専ら研究に従事する者
(4) 学術研究及び学術振興を目的とする機関（前号に掲げる機関を除く。）で、センターの長（以下「センター長」という。）が認めた機関に所属し、専ら研究に従事する者
(5) 科学研究費補助金の交付を受けて学術研究を行う者
(6) 前各号のほか、特にセンター長が適当と認めた者

第4条 利用者は、所定の申請書により申請し、センター長の承認を得なければならない。ただし、上記の申請はセンターホームページから行えるものとする。

第5条 センター長は、前条の申請について適当と認めた場合は、当該利用番号を与えて承認するものとする。

第6条 利用者の有効期間は、前期（4月～9月）3ヶ月間、又は後期（10月～3月）1ヶ月間とする。ただし、当該会計年度を超えることはできないものとする。
2 基本負担額10,000円の場合と同じ計算機資源を利用可能とする。
3 利用有効期間を超えた場合は、強制的に利用を取り消すものとする。

第7条 利用者は、当該利用番号を当該申請に係る目的以外に使用し、又は他人に使用させてはならない。

第8条 センター長は、この内規に違反した場合、もしくは氏名等を偽り利用した場合、その他大規模計算機システムの運営に重大な支障を生ぜしめた場合には、当該利用の承認を取り消すことがある。

附 則
この内規は、平成12年11月30日から施行し、平成12年4月1日から適用する。
附 則
この改正は、平成13年1月6日から施行する。
附 則
この改正は、平成14年4月1日から施行する。
附 則
この改正は、平成16年4月1日から施行する。
附 則
この改正は、平成18年4月1日から施行する。
附 則
この改正は、平成19年1月5日から施行する。
附 則
この改正は、平成19年9月28日から施行する。

大阪大学サイバーメディアセンター大規模計算機システム利用相談員指導員内規

第1条 大阪大学サイバーメディアセンター（以下「センター」という。）は、センターが管理・運用する全国共同利用のスーパーコンピュータシステム及びワークステーション（以下「大規模計算機システム」という。）の共同利用の効果を高め学術研究の発展に資するため、大規模計算機システム利用相談及び指導活動（データベース開発指導を含む。）を行う。
2 前項の目的のため、センターに利用相談員（以下「相談員」という。）及び利用指導員（以下「指導員」という。）を置く。

第2条 相談員及び指導員は、共同利用有資格者の中から広報委員会が候補者を推せんし、教授会の議を経てセンター長が委嘱する。

第3条 相談員及び指導員の任期は、4月1日又は10月1日からの1カ年とする。ただし、再任を妨げない。

第4条 相談員は、電子メール等を利用しオンラインで、第1条第1項のセンター利用相談活動を行うものとする。

第5条 指導員は、所属の地区協議会連絡所において、第1条第1項のセンター利用指導活動を行うものとする。

第6条 相談員及び指導員には、センター利用相談及び指導の必要上、計算機利用のために特定の番号を与えることができる。
2 前項に係る利用経費の負担額は免除する。

第7条 センターは、相談員及び指導員に対し相談及び指導上必要な資料もしくは情報を提供するものとする。

第8条 センターは、相談員及び指導員に対する研修会並びに研究連絡会等を実施するものとする。
2 前項の企画及び実施に当たっては、広報委員会が企画・立案し、教授会の承認を得るものとする。

第9条 相談員には、第6条第1項の目的以外においても、一定量の大規模計算機システム使用にかかるジョブ優先処理等の特典を与えることができる。

第10条 この内規に定めるもののほか、必要な事項については広報委員会で検討後、教授会の議を経てセンター長が別に定めるものとする。

附 則
この内規は、平成12年11月30日から施行し、平成12年4月1日から適用する。
附 則
この改正は、平成19年9月28日から施行する。

大型計算機利用大阪地区（第6地区）協議会規程

第1条　大型計算機利用大阪地区（第6地区）協議会（以下「本会」という。）は、大阪大学サイバーメディアセンターが管理・運用する全国共同利用のスーパーコンピュータシステム、コンピュータシステム及び関連するネットワーク（以下「大型計算機システム等」という。）の利用を希望し、本会に所属するものの利便をはかることを目的とする。

第2条　本会の事務局を大阪大学サイバーメディアセンター内に置く。

第3条　本会は、大阪、和歌山、奈良、兵庫、岡山、香川、愛媛、高知及び徳島の9府県内にある連絡所をもって会員とする。
2　上記以外で、理事会が特に認めた連絡所は会員とすることができる。

第4条　連絡所を設けようとするものは、責任者を定め、連絡所登録申請書を本会事務局へ提出し、理事会の承認を受けなければならない。
2　前項の連絡所の廃止をするものは、連絡所廃止届を本会事務局へ提出しなければならない。
3　連絡所の責任者は、その連絡所に所属し、大型計算機システム等を利用するものを代表して、必要な事務を処理する。

第5条　本会は、第1条に示された目的を達成するため、次の事業を行う。
一　会員の登録承認
二　大阪大学サイバーメディアセンターと会員間の連絡及び調整
三　他の地区協議会との事務連絡及び情報交換
四　その他理事会が必要と認めた事項

第6条　本会に会長1名、理事若干名の役員を置く。
2　本会に幹事若干名を置き、役員を補佐せしめることができる。
3　幹事は、理事会の承認を経て、会長が委嘱する。

第7条　会長は本会を代表し、本会の業務を総括する。
2　会長は理事の互選によって定める。
3　会長の任期は2年とし、再任を妨げない。ただし、任期途中で交代した会長の任期は、前任の会長の残任期とする。

第8条　理事は会員の互選によって定める。
2　理事の任期は2年とし、再任を妨げない。ただし、任期途中で交代した理事の任期は、前任の理事の残任期とする。

第9条　会長は理事会を招集し、その議長となる。
2　理事会は次の事項を審議する。
一　連絡所の設置の承認
二　事業計画の立案並びに実行
三　その他会長が必要と認めた事項
3　理事会は、理事現在数の2分の1以上の出席がなければ開催することができない。ただし、あらかじめ委任状を提出したものは出席者とみなす。
4　理事会の議事は、出席者の過半数をもって決し、可否同数のときは、議長が決する。

第10条　会長は年1回以上総会を招集し、その議長となる。
2　総会は次の事項を審議する。
一　本会規程の改廃
二　事業報告
三　事業計画
四　その他理事会が必要と認めた事項
3　総会は、会員現在数の5分の1以上の会員が出席しなければ開催することができない。ただし、あらかじめ委任状を提出したものは出席者とみなす。
4　総会の議事は、出席者の過半数をもって決し、可否同数のときは議長が決する。

第11条　本会は、特定事項の審議等のため、必要に応じて専門部会を置くことができる。
2　専門部会に関し必要な事項は、本会が別に定める。

附　則
この改正は、平成12年10月4日から施行し、平成12年4月1日から適用する。
附　則
この改正は、平成14年10月15日から施行し、平成14年4月1日から適用する。
附　則
この改正は、平成17年10月14日から施行し、平成17年4月1日から適用する。

ネットワーク専門部会内規

第1条　大型計算機利用大阪地区（第6地区）協議会（以下「第6地区協議会」という。）規程（以下「協議会規程」という。）第11条に規定する専門部会として、ネットワーク専門部会（以下「専門部会」という。）を置く。

第2条　専門部会は、学術研究、教育活動等を支援するネットワークの情報交換等の便宜を図り、地域に貢献することを目的とする。

第3条　専門部会は、次の各号に掲げるものをもって構成する。
1　協議会規程第3条に規定する会員
2　その他専門部会が必要と認めた者

第4条　専門部会に部会長を置き、第6地区協議会会長が指名する。
2　部会長は、専門部会を招集し、その議長となる。

第5条　専門部会は、通常は年1回、第6地区協議会の開催に併せて開催することとし、必要に応じて開催することができる。

附　則

この内規は、平成 14 年 10 月 15 日から施行し、平成 14 年 4 月 1 日から適用する。

・附表

大規模計算機システム　ホスト一覧

サーバ名	ホスト名
ログインサーバ	login.hpc.cmc.osaka-u.ac.jp
ファイル転送サーバ	ftp.hpc.cmc.osaka-u.ac.jp
Mail サーバ	mail.hpc.cmc.osaka-u.ac.jp
大規模計算機システムホームページ	http://www.hpc.cmc.osaka-u.ac.jp
Web ポータル	https://portal.hpc.cmc.osaka-u.ac.jp

スーパーコンピュータなどの演算システムへは、ログインサーバ経由での接続となります。
（ホスト一覧表には明記していません）

SX-8R、SX-9 及び PC クラスタのジョブクラス一覧

スーパーコンピュータと PC クラスタのジョブ資源制限値は次のとおりです。

クラス	経過時間		使用 CPU 数		主記憶(GB)	
	既定値(分)	最大値(時間)	既定値	最大値	既定値	最大値
DBG	1(1 分)	1(10 分)	1	4	1	16
SX8F(SXF)	1	24	1	8	1	120
SX8L(SXL)	1	240	1	32	1	1000
SX8L(届出制)	1	240	1	64	1	2000
DBG9	1(1 分)	1(10 分)	1	4	1	128
SX9	1	24	16	64	64	4000
SX9(届出制)	1	240	16	128	64	8000
PCC	1	740	4	---	16GB/ノード	

募　　集

大規模計算機システムを利用して行った研究･開発等の記事の募集について

センターでは、大規模計算機システムを利用して研究したことを主体とする内容の広報誌の発行を予定しています（6月、12月）。この広報誌に掲載する次の内容の記事を募集しますので、皆さんのご投稿をお待ちします。

1. 随筆
2. 大規模計算機システムを利用して行った研究・開発の紹介
3. プログラムの実例と解説
4. その他、広報誌に掲載するにふさわしいもの

＊投稿いただいた方には、ご希望により掲載した広報誌5部を進呈いたします。

【原稿執筆に当たってご注意いただく事項、及び原稿の提出方法】

a) Texの場合（オフセット印刷可能なもの）

・Latex のソースファイルは、MS-DOS 形式のフロッピーあるいは電子メールで提出してください。
・図面・画像は、別途提出してください。また、図表や写真の挿入箇所を指示しておいてください。
・書式は、
　フォントサイズ：10point
　スタイルファイル：jarticle
　上余白：20mm、下余白：20mm
　左余白：20mm、右余白：20mm

b) ワードプロセッサ出力（オフセット印刷可能なもの）

・テキストファイルは、MS-DOS形式のフロッピーあるいは電子メールで提出してください。
・図面及び画像は、別途提出してください。また、図表や写真の挿入箇所を指示しておいてください。
・書式は、
　フォントサイズ：10point
　フォント：MS 明朝
　上余白：20mm、下余白：20mm
　左余白：20mm、右余白：20mm

a,b 共通）

・1頁1段の行数：50行
　1行の文字数：24文字
　1頁2段書き、フォントはMS明朝
・数字、英字は半角、フォントはTimes New Roman
　数字、英字を括弧で閉じる場合は、括弧も同様に半角
・文字、漢字は全角。文字、漢字を括弧で閉じる場合の括弧は全角
・文中の半角の“, ”や“. ”は、全て全角の“、”や“。”とする。
・原稿の題名は、MS ゴシックで14Point。執筆者氏名は、姓と名の間及び所属機関・学部・専攻名の間を半角スペースとし、フォントサイズは10Point、フォントはMS明朝とする。

・注意事項

a) 投稿原稿は原則として、サイバーメディアセンターホームページに公開されますことをご了承ください。なお、公開を希望されない場合には、その旨をご連絡してください。また、原稿の返却はいたしません。
b) 執筆者には、ご希望により冊子を5部進呈します。
c) 初回の校正は、誤植の防止をはかるため執筆者にお願いします。
d) 原稿の送付先は、下記のとおりです（大阪大学内の方は、学内便でお願いします）。

郵送：〒567-0047
大阪府茨木市美穂ヶ丘5－1
大阪大学サイバーメディアセンター
情報推進部情報企画課
情報企画班
TEL　06-6879-8808　FAX　06-6879-8814
E-mail：usersv@cmc.osaka-u.ac.jp

広報委員会委員

小田中　紳　二（委員長、大阪大学サイバーメディアセンター）
藤　　堅　正（近畿大学理工学部）
豊　永　昌　彦（高知大学理学部）
前　迫　孝　憲（大阪大学大学院人間科学研究科）
養　老　真　一（大阪大学大学院法学研究科）
小　郷　直　言（大阪大学大学院経済学研究科）
阿　部　浩　和（大阪大学サイバーメディアセンター）
清　川　　清（大阪大学サイバーメディアセンター）
竹　蓋　順　子（大阪大学サイバーメディアセンター）
時　田　恵一郎（大阪大学サイバーメディアセンター）
馬　場　健　一（大阪大学サイバーメディアセンター）

大阪大学サイバーメディアセンター
計算機利用ニュース　2008年度　Vol.4 No.2
2008年12月発行

編集者　大阪大学サイバーメディアセンター
広報委員会

発行者　大阪府茨木市美穂ヶ丘5-1（〒567-0047）
大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University
Tel：06-6879-8808
URL：http://www.hpc.cmc.osaka-u.ac.jp/j/

印刷所　大阪市福島区玉川3-6-4
阪東印刷紙器工業所

目　　次

（お願い）
このニュースは、本センター利用者（利用登録者）の皆様に配布しています。
お近くの研究者・大学院生の方にも、このニュースをご回覧くださるようお願い申し上げます。