数値計算ライブラリー

MATRIX/MPP, MATRIX/MPP/SSS, MSL2 のご紹介

（株）日立製作所

１．はじめに

MATRIX/MPP, MATRIX/MPP/SSS, および MSL2 は, SR11000 上で利用可能な数値計算ライブラリーです。行列計算, 関数計算, 統計計算といった数値計算の分野でよく使われる計算をサポートしています。SR11000 向けにチューニングしており, ライブラリーを使用することで利用者プログラムの高速化ができます。

MATRIX/MPP, MATRIX/MPP/SSS, および MSL2 は, 共有メモリー型並列に対応した機能を有しており, MATRIX/MPP および MATRIX/MPP/SSS は, 分散メモリー型並列に対応した機能も有します。

ここでは, MATRIX/MPP, MATRIX/MPP/SSS, MSL2 の機能, およびインターフェイスの概要についてご紹介いたします。

２．機能概要

ライブラリー体系を図 2.1 に示します。

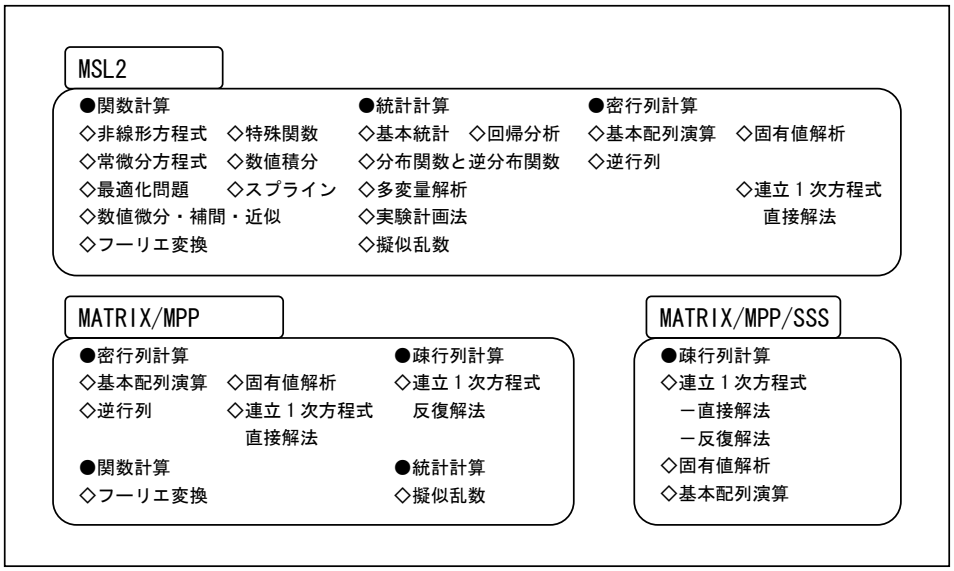


図 2.1 ライブラリー体系

MSL2 は, 行列計算, 関数計算, 統計計算といった数値計算の分野でよく使われる計算をサポートした数値計算副プログラムライブラリーです。

MATRIX/MPP および MATRIX/MPP/SSS は，行列計算副プログラムライブラリーです。MATRIX/MPP は，MSL2 で扱う分野のうち，行列計算分野と，関数計算分野のフーリエ変換機能および関数計算分野の擬似乱数機能を，SR11000 向けにチューニングしたライブラリーです。MATRIX/MPP/SSS は，構造解析等の分野で扱う大次元疎行列に対するライブラリーです。

行列計算副プログラムライブラリーは継続して機能を拡張しています。SR11000 向けライブラリーでは，SR8000 向けライブラリーに対して主に次の機能を新規に追加しました。

(1) MATRIX/MPP
  (a) 基本配列演算
    ・複素数行列の行列乗算
    ・実転置行列の行列乗算
  (b) 固有値・固有ベクトル
    ・エルミート行列に対する一般固有値・固有ベクトル
  (c) 高速フーリエ変換
    ・2 次元および 3 次元の実フーリエ変換 (Real ⇔ Complex)

(2) MATRIX/MPP/SSS
  (a) 連立 1 次方程式－直接解法
    ・エルミート疎行列に対するスパースダイレクトソルバー
  また，次の分野を新たに追加しました。
  (b) 連立 1 次方程式－反復解法
  (c) 固有値・固有ベクトル
  (d) 基本配列演算

３．特長

３．１　各ライブラリーの特長

以下に主な特長を示します。

(1) MATRIX/MPP, MATRIX/MPP/SSS
  ・SR11000 向けにチューニングしたライブラリー
  ・分散メモリー型並列に対応した機能をサポート
  ・7 基底まで対応した混合基底の高速フーリエ変換機能をサポート
  ・非ゼロ要素の構造に着目し，ゼロ要素との演算を抑止することで，スカイライン法に比べ演算量を大幅に削減したスパースダイレクトソルバー機能をサポート
  ・スパースダイレクトソルバーの前処理において，ゼロ要素が計算過程で非ゼロ要素となる Fill-in 数を少なくするオーダリング機能をサポート

(2) MSL2
  ・行列計算，関数計算，統計計算といった数値計算の分野でよく使われる計算をサポート

３．２　SR11000 向け最適化方式

SR11000 のハードウェア性能を引き出すために，ライブラリーでは次のような手法を採用しています。

・並列性の高いアルゴリズムの採用

・キャッシュブロッキング

・スーパースカラの有効利用（ループアンローリングなど）

以下に，並列性の高いアルゴリズムの採用例とキャッシュブロッキングの例を示します。

(1) 並列性の高いアルゴリズムの採用

アルゴリズムを改善し並列性を高めることで，処理を高速にしています。

ここでは一例として，1 次元フーリエ変換を 2 次元フーリエ変換に変形するアルゴリズム[1]について説明します。

フーリエ変換は一般に，

$$C_k = \sum_{j=0}^{N-1} f_j \exp(-\frac{2\pi kj}{N} i) \qquad (k = 0,1,\cdots, N-1) \qquad \cdots\cdots\cdots\cdots\cdots\cdots \text{式 1}$$

で表されます。

ここで，$N = Nx * Ny$ と因数分解できるとするとき，式 1 における j および k を，

$j = sNx + r$　（ただし，$0 \leqq s < Ny$，$0 \leqq r < Nx$）

$k = pNy + q$　（ただし，$0 \leqq p < Nx$，$0 \leqq q < Ny$）

と表すと式 1 は，次のように表すことができます。

$$C_{pNy+q} = \sum_{r=0}^{Nx-1} \sum_{s=0}^{Ny-1} f_{sNx+r} \exp(-\frac{2\pi (pNy+q)(sNx+r)}{N} i) \qquad \cdots\cdots\cdots\cdots\cdots\cdots \text{式 2}$$

ここで，三角関数の周期性から，

$$\exp(-\frac{2\pi i}{N}(pNy+q)(sNx+r)) = \exp(-2\pi i(\frac{pr}{Nx} + \frac{qs}{Ny} + \frac{qr}{N}))$$

が成り立つので式 2 は，次のように表すことができます。

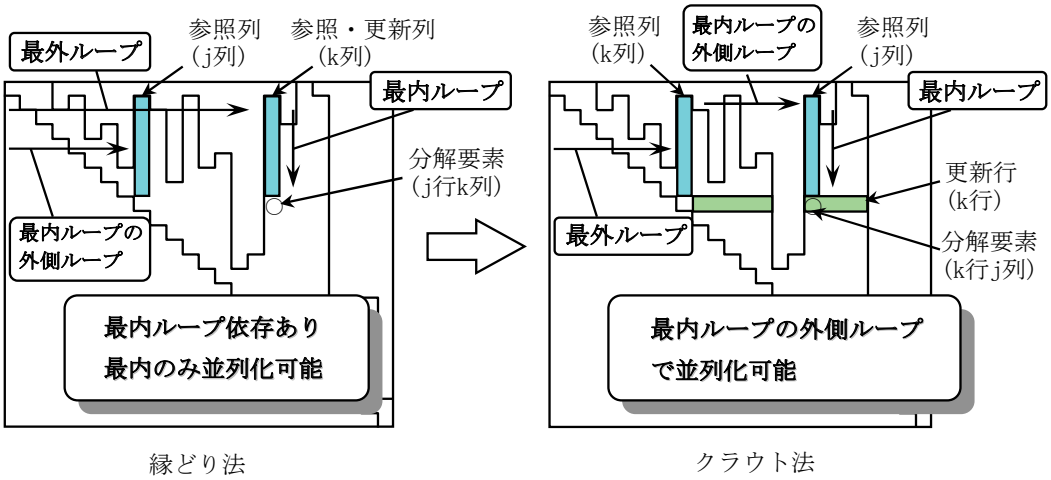
$$C_{pNy+q} = \sum_{r=0}^{Nx-1} \sum_{s=0}^{Ny-1} f_{sNx+r} \exp(-\frac{2\pi pr}{Nx} i) \exp(-\frac{2\pi qs}{Ny} i) \exp(-\frac{2\pi qr}{N} i)$$

$$= \sum_{r=0}^{Nx-1} \left[\sum_{s=0}^{Ny-1} (f_{sNx+r} \exp(-\frac{2\pi qs}{Ny} i)) \exp(-\frac{2\pi qr}{N} i) \right] \exp(-\frac{2\pi pr}{Nx} i) \quad \cdots \text{式 3}$$

①　　②

式 3 で，N 点の 1 次元フーリエ変換が Nx 組 Ny 点の 2 次元フーリエ変換と Ny 組 Nx 点の 2 次元フーリエ変換に分離されます。分離されることで，①が r に依存しないため Nx に対して並列化可能になり，②が s に依存しないため Ny に対して並列化可能になります。

次に一例として，スカイライン法での手法を図 3.1 に示します。



縁どり法　　クラウト法

図 3.1 ライブラリーでの並列化手法

図 3.1 に示す縁どり法[2]では，1～(k-1)列を参照し，参照・更新列(k 列)を更新します。図は，参照列(j 列)を参照し，分解要素(j 行 k 列の要素)の更新を表しています。参照列が次の参照列(j+1 列)となるとき，前の処理で更新した分解要素(j 行 k 列の要素)を参照するため，1～(k-1)列の最内ループの外側ループにまたがり更新と参照の依存関係があります。そのため最内ループのみが並列化可能です。

上記の縁どり法での最内ループの外側ループ繰り返し依存の関係を解消するために，ライブラリーではクラウト法[2]を採用しています。図 3.1 に示すクラウト法では，k+1～N 列を参照し，更新行(k 行)を更新します。参照列(j 列)と更新行(k 行)には依存がないため，最内ループの外側ループで並列化可能となります。

(2) キャッシュブロッキング

キャッシュメモリーは実メモリーに比べアクセス速度が高速のため，キャッシュメモリーに入ったデーターを繰り返し利用できれば，実メモリーのデーターを用いるより計算処理が高速になります。キャッシュメモリーは容量が少ないため，計算に必要な全てのデーターを持つことはできません。そのため行列データーを，キャッシュメモリーに入るよう一定の大きさにブロッキングします。

ライブラリーでは，SR11000 に適したキャッシュブロッキングを採用することで，高速な演算を行っています。行列乗算 C=A×B の計算処理におけるブロッキング例を図 3.2 に示します。

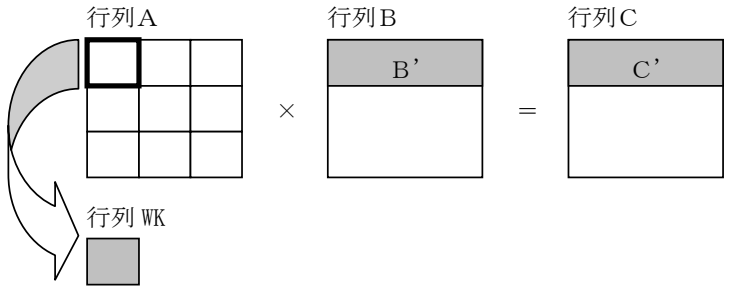


図 3.2 行列のブロッキング例

行列 WK は，行列 A の 1 ブロックを転置コピーしたものです。行列乗算処理では，行列 WK と行列 B の B’部分を参照して行列乗算を行い，行列 C の C’部分に格納します。このとき行列 WK はキャッシュメモリーに入った状態で繰り返し利用されます。

一般固有値の標準化変換における前進消去演算でのブロッキング例を図 3.3 に示します。

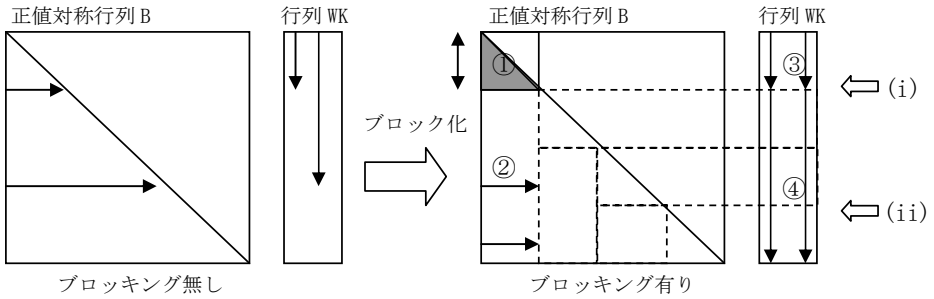


図 3.3 一般固有値でのブロッキング例

図 3.3 の行列 WK は，圧縮 1 次元配列で格納された実対称行列 A を，連続にアクセスするためにコピーした 2 次元配列です。右のブロッキング有りの例では，次のように処理します。

(i) 正値対称行列 B の対角ブロック①と行列 WK のブロック③を参照して，行列 WK のブロック③を更新

(ii) 次に，更新が終わった行列 WK のブロック③と正値対称行列のブロック②を参照して，行列 WK のブロック④の消去計算を実施

上記のように処理することで，参照する行列データーを常にキャッシュメモリーに入れておき，高速な前進消去演算を実現しています。

４．インターフェイス仕様

４．１　並列化の概要

並列化にはノード内並列化とノード間並列化があり，それぞれに対応したインターフェイスがあります。概要を図 4.1 に示します。

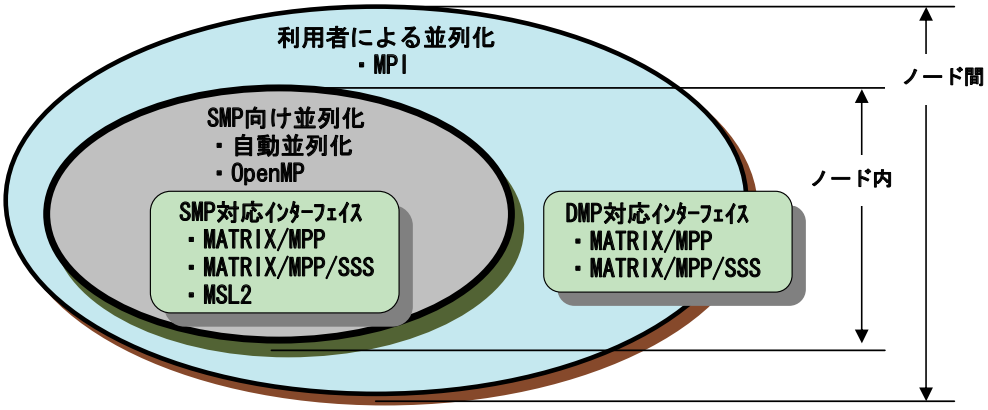


図 4.1 並列化の概要

SR11000 のノード内並列化は共有メモリー型並列（SMP:Shared Memory Parallel）で，ノード間並列化は分散メモリー型並列（DMP:Distributed Memory Parallel）です。

ノード内並列化は自動および OpenMP で並列化します。ノード間並列化は利用者プログラムで MPI(Message Passing Interface)を用いて並列化します。

MATRIX/MPP，MATRIX/MPP/SSS，および MSL2 は，ノード内並列化への対応として SMP 対応インターフェイスを有しています。

MATRIX/MPP および MATRIX/MPP/SSS は，SMP 対応インターフェイスに加え，ノード間並列化への対応として DMP 対応インターフェイスを有しています。

４．２　利用形態

ライブラリーでは利用形態に応じて，SMP 対応インターフェイスとして逐次処理用インターフェイスを，DMP 対応インターフェイスとして並列処理用インターフェイスを提供しています。次に各インターフェイスの概要を示します。

(1) 逐次処理用インターフェイス

1 ノード内の複数プロセッサーを使用して実行する場合に利用します。逐次に実行する利用者プログラム，ノード内およびノード間で並列に実行する利用者プログラムから利用可能です。ライブラリー内部ではノード間通信を行いません。

逐次処理の利用形態を図 4.2 に示します。

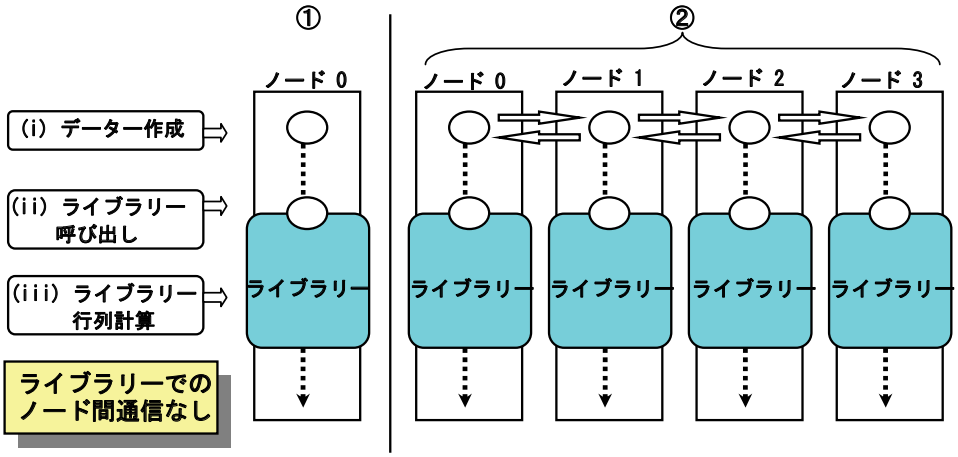


図 4.2 逐次処理の利用形態

図 4.2 において，(i)は利用者プログラムによるデーター作成，(ii)は利用者プログラムからのライブラリー呼び出し，(iii)はライブラリー内での行列計算を示します。

①利用者プログラムを 1 ノードで実行

利用者プログラム，ライブラリーともに，ノード間通信なし。

②利用者プログラムを 4 ノード間で並列に実行（逐次処理用インターフェイスを使用）

(i) データー作成は，利用者プログラムがノード間でデーター通信

(ii) 各ノードでライブラリーを呼び出す

(iii)ライブラリーはそれぞれのノード内で動作し，ライブラリーの内部ではノード間通信なし

(2) 並列処理用インターフェイス

複数ノードを使用して実行する場合に利用します。ノード間で並列に実行する利用者プログラムから利用可能です。計算の対象となるデーターを分散メモリー上に分散して配置し利用するインターフェイスで，ライブラリー内部でノード間のデーター通信を行います。

並列処理の利用形態を図 4.3 に示します。

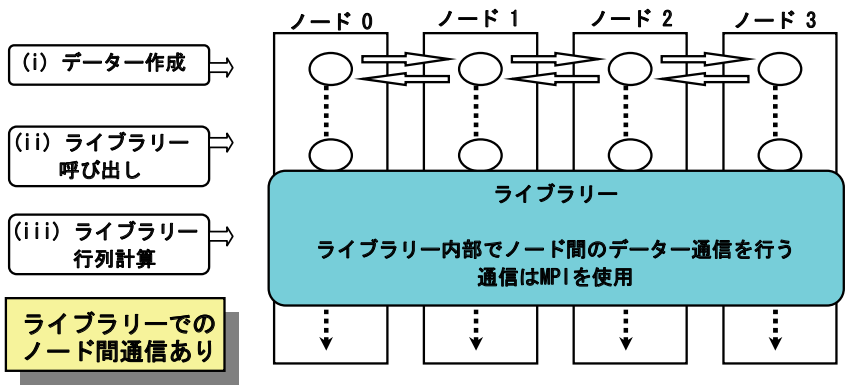


図 4.3 並列処理の利用形態

図 4.3 は，利用者プログラムを 4 ノード間で並列に実行し，並列処理用インターフェイスを使用する例です。

(i) データー作成は，利用者プログラムがノード間でデーター通信

(ii) 各ノードでライブラリーを呼び出す

(iii)ライブラリーの内部で MPI を用いてデーター通信

４．３　データーの与え方

ライブラリーでは利用するインターフェイスにより，行列種別ごとにデーターの与え方が異なります。データーの与え方を次に示します。

(1) 逐次処理用インターフェイス

逐次処理用インターフェイスのデーターの与え方は，表 4.1 のように分類されます。

表 4.1 逐次処理用インターフェイスのデーター格納配列

行列種別	データー格納配列
密非対称行列	1 次元配列
密対称行列	圧縮 1 次元配列
帯行列	圧縮 2 次元配列
疎行列	圧縮 1 次元配列 圧縮 2 次元配列
スカイライン行列	圧縮 1 次元配列
ベクトル	1 次元配列

(2) 並列処理用インターフェイス

並列処理用インターフェイスでは複数ノードで実行するため，各ノードにデーターを分割して与えます。データー分割方式を表 4.2 に示します。

表 4.2 並列処理用インターフェイスのデーター分割方式

行列種別	データー分割方式
密非対称行列	Scattered Square 分割 ブロックサイクリック列分割
密対称行列	Scattered Square 分割 ブロックサイクリック列分割
帯行列	Scattered Square 分割
疎行列	差分法領域分割による配列 ブロック行分割による圧縮型 2 次元配列
スカイライン行列	Scattered Square 分割 ブロックサイクリック列分割
ベクトル	Scattered Square 分割

４．４　並列処理用インターフェイスのデーター割り当てと動作例

並列処理用インターフェイスではノード間でデーター通信を行います。並列処理用インターフェイスでのデーター分割と動作の例として，Scattered Square 分割とブロックサイクリック列分割の概要を次に説明します。

(1) Scattered Square 分割

(a) データー分割

ブロック形式 Scattered Square 分割の，16 ノード使用時の構造例を図 4.4 に示します。

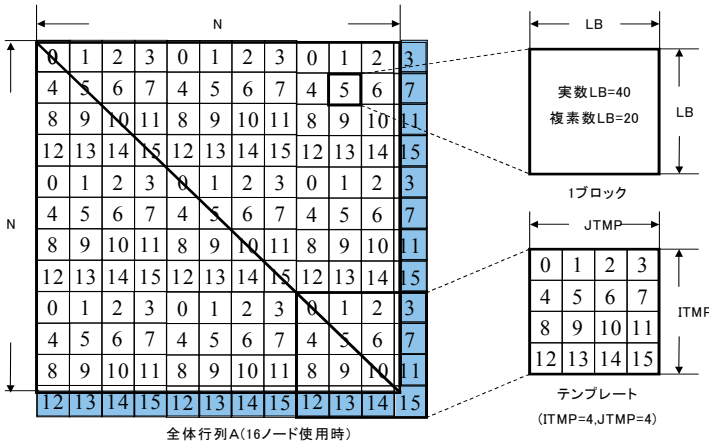


図 4.4 ブロック形式 Scattered Square 分割の構造例

ブロック形式 Scattered Square 分割では，全体行列をブロックの集まりとして捉えます。1 ブロックの LB（ブロックサイズ）は，実数では 40，複素数では 20 です。

図 4.4 では，図中の数字は，データーを割り当てるノード番号です。行列データーの分配では，使用するノード台数によってテンプレートを決定し，そのテンプレートにしたがって行列データーを分配します。

テンプレートの形状である ITMP と JTMP の大きさは，使用するノード台数を NPU とすると，NPU=ITMP*JTMP の関係です。このため，ITMP と JTMP の大きさは，計算で使用するノード台数によって決まります。この例ではノード台数が 16 であることから，ITMP=4, JTMP=4 となります。

図 4.4 のように分割したデーターは，部分行列に格納してライブラリーに引き渡します。部分行列への格納方式を図 4.5 に示します。

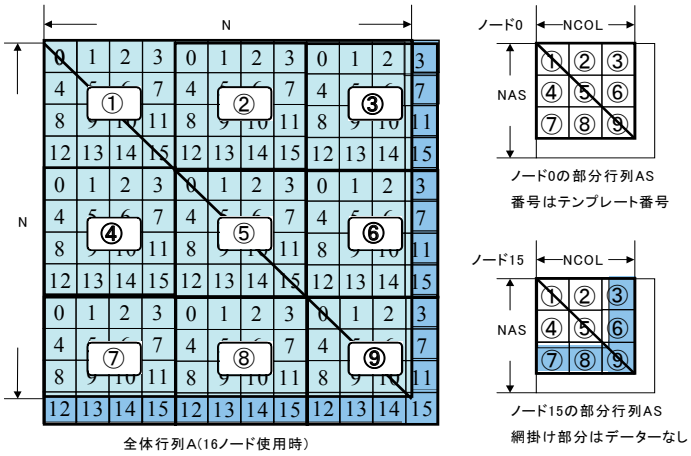


図 4.5 部分行列への格納方式

図 4.5 では，右の部分配列 AS 内の数字は，左の全体行列 A のテンプレート番号です。

網掛け部分は，データーの無い部分であり，初期化の必要はありませんが，領域は確保しておく必要があります。

(b) 動作概要

4 ノード使用時の，行列乗算の動作概要例を図 4.6 に示します。

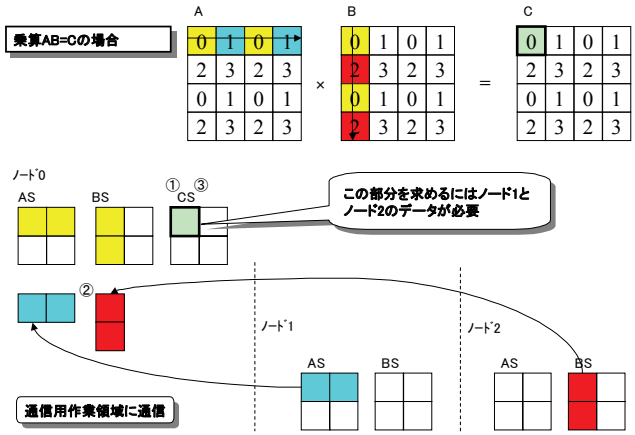


図 4.6 行列乗算の動作概要例

図 4.6 のノード 0 の動作は次のとおりです。

① 自ノードに割り当てられた AS および BS を参照し CS を計算

② 通信用作業領域を用いて，CS を求める際に必要となるノード 1 の AS とノード 2 の BS をノード 0 で受信

③ 受信したデーターを参照し CS を計算

(2) ブロックサイクリック列分割

(a) データー分割

ブロック形式サイクリック列分割の，4 ノード使用時の構造例を図 4.7 に示します。

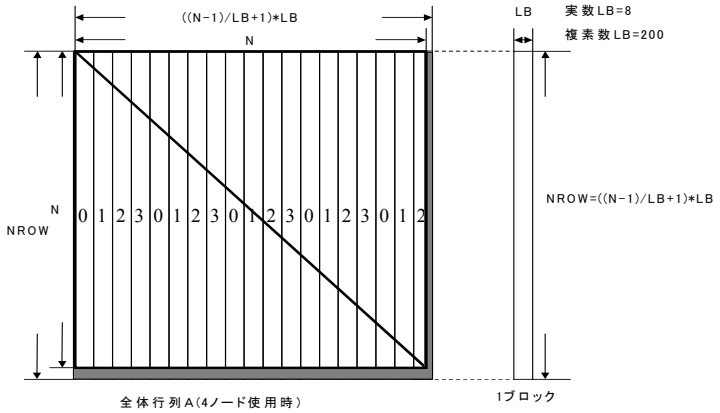


図 4.7 ブロック形式サイクリック列分割の構造例

ブロック形式サイクリック列分割では，全体行列のデーターをブロック単位に分割して，ブロックの集まりを部分行列に格納してライブラリーに引き渡します。図 4.7 の図中の数字は，データーを割り当てるノード番号です。1 ブロックの LB（ブロックサイズ）は，実数では 8，複素数では 200 です。

部分行列への格納方式を図 4.8 に示します。

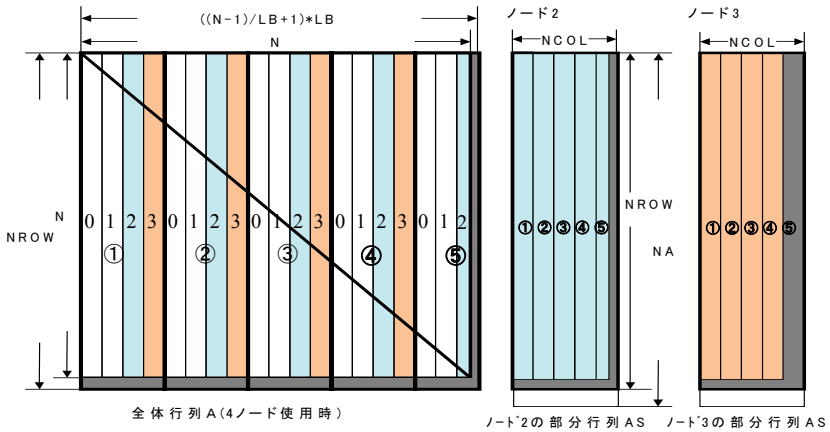


図 4.8 部分行列への格納方式

図 4.8 では，右の部分配列 AS 内の数字はテンプレート番号で，左の全体行列 A のテンプレート番号に対応した当該ノード用のデーターを集めた形となります。

(b) 動作概要

4 ノード使用時の連立 1 次方程式の LU 分解における動作概要例を図 4.9 に示します。

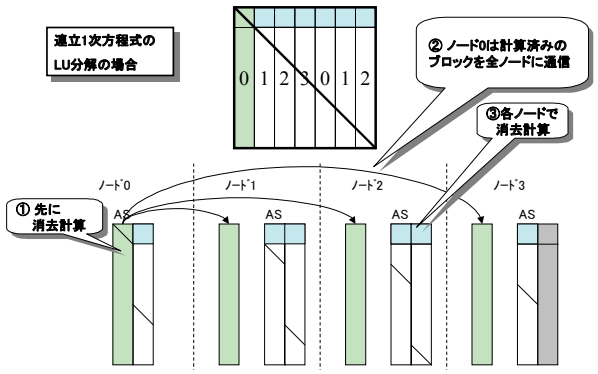


図 4.9 連立 1 次方程式の LU 分解における動作概要例

図 4.9 では，次のとおり動作します。

① ノード 0 で軸列ブロックの消去計算

② 消去計算した軸列ブロックをノード 0 から各ノードに対して送信

③ 各ノードではノード 0 から受け取った軸列ブロックを参照して消去計算

５．性能測定結果

ライブラリーはノード内並列化およびノード間並列化に対応しているため，使用するプロセッサー数およびノード数により計算時間が短縮でき，プログラムの高速化が図れます。MATRIX/MPP での性能評価結果を次に示します。

(1) 3 次元複素フーリエ変換(HZFT7M)

3 次元複素フーリエ変換で変換点数が 512×512×512 の場合の性能評価結果を図 5.1 に示します。

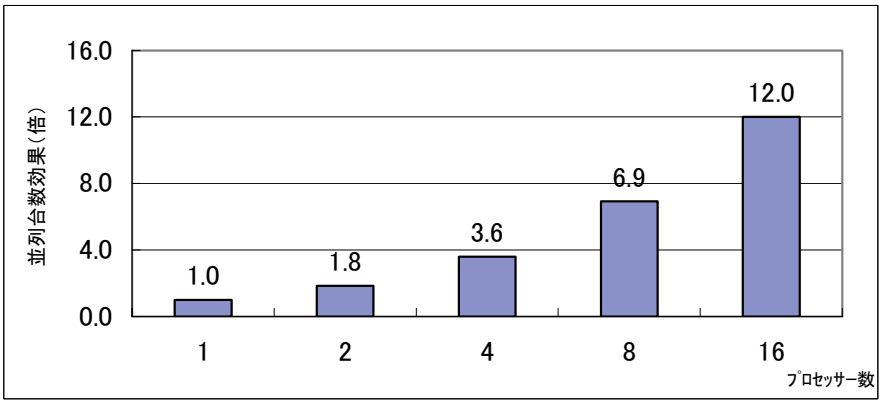


図 5.1 3 次元複素フーリエ変換(HZFT7M) 性能比較

図 5.1 は，横軸にノード内並列化による使用プロセッサー数を，縦軸に 1 プロセッサーで計算した場合の計算時間を 1 とし，1 プロセッサーで計算した場合の計算時間を複数プロセッサーで計算した場合の計算時間で割ったプロセッサー数による並列台数効果を示した図です。

1 プロセッサーに比べて，2 プロセッサーでは 1.8 倍，4 プロセッサーでは 3.6 倍，8 プロセッサーでは 6.9 倍，16 プロセッサーでは 12.0 倍となります。

(2) 連立 1 次方程式分散版(HDLPPMDP)

30000 次元の実対称行列を係数行列とする連立 1 次方程式の解を求めた場合の性能評価結果を図 5.2 に示します。

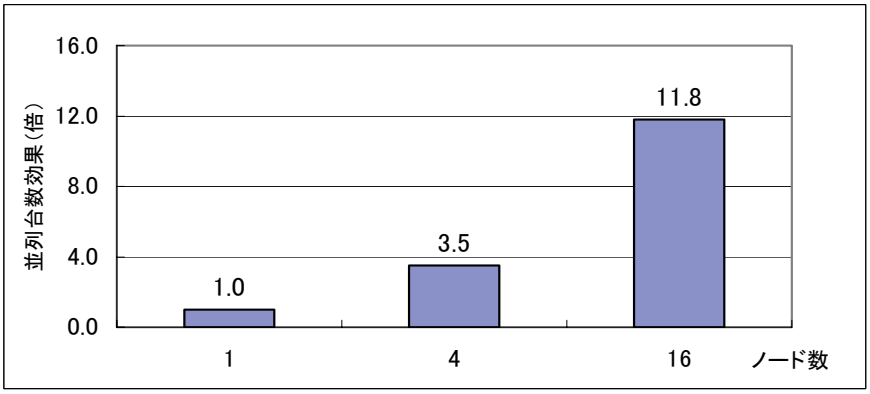


図 5.2 連立 1 次方程式（実対称）性能比較

図 5.2 は，横軸にノード間並列化による使用ノード数を，縦軸に 1 ノードで計算した場合の計算時間を 1 とし，1 ノードで計算した場合の計算時間を複数ノードで計算した場合の計算時間で割ったノード数による並列台数効果を示した図です。

1 ノードに比べて，4 ノードで計算を行うと 3.5 倍，16 ノードで計算を行うと 11.8 倍になります。

６．まとめ

以上で示しましたとおり，ライブラリーのノード内並列化およびノード間並列化に対応したインターフェイスを使用することで利用者プログラムを簡略化できるとともに，並列性の高いアルゴリズムやキャッシュブロッキング等を採用して SR11000 向けにチューニングしたライブラリーを使用することで容易に並列台数効果を得ることができます。

7．ライブラリー使用時の注意点

ライブラリーは，利用者プログラムで注意が必要な点や，指定値によって性能に差異が出る場合などがあります。ここでは例として，並列処理用インターフェイス使用時の注意点と，整合寸法指定値による性能差について示します。

(1) 並列処理用インターフェイス使用時の注意点

ライブラリーは内部でMPIを使用してノード間でデーター通信を行います。ライブラリーの初期処理用副プログラム(HMATINITまたはHCOMINIT)および終了処理用副プログラム(HMATEXITまたはHMATFINL）と，MPIの初期処理（MPI_INIT）およびMPIの終了処理（MPI_FINALIZE）との関係を表7.1に示します。

表7.1 ライブラリーの初期処理/終了処理とMPI_INIT/MPI_FINALIZEの関係

副プログラム	名称	利用者プログラムでのMPI_INITの発行	動作	注意事項
初期処理用副プログラム	HMATINITまたはHCOMINIT	発行済	MPI_INITは発行しない	－
		発行していない	MPI_INITを発行する	初期処理用副プログラムでMPI_INITを発行するため，以降は利用者プログラムでもMPI関数を使用できる
終了処理用副プログラム	HMATFINL	発行済	MPI_FINALIZEは発行しない	利用者プログラムでMPI_INITを発行した場合，MPI_FINALIZEも利用者プログラムで発行する必要がある
		発行していない	MPI_FINALIZEを発行する	終了処理用副プログラムでMPI_FINALIZEを発行するため，以降は利用者プログラムでもMPI関数は使用できない
	HMATEXIT	－	MPI_FINALIZEは発行しない	利用者プログラムでのMPI_INITの発行有無に関係しない

並列処理用インターフェイスの副プログラムを実行する場合は，副プログラムの実行前に初期処理用副プログラムで環境設定をしなければなりません。

また副プログラムの実行終了後には，終了処理用副プログラムの実行が必要です。終了処理用副プログラムにより，初期処理で設定した環境を解放します。

HMATFINLでは，それ以前にMPI_INITが発行されているかを判別し，判別結果により必要に応じてMPI_INITおよびMPI_FINALIZEを発行します。

利用者プログラムでMPI_INITを発行している場合は，HMATFINLではMPI_FINALIZEを発行しません。そのため，MPI通信関数を使用しなくなる時点で，利用者プログラム内でMPI_FINALIZEの発行が必要です。

MPI_FINALIZEを発行すると，それ以降は並列処理用副プログラムだけでなく利用者プログラム中でもMPI関数が使用できなくなりますので，注意が必要です。

(2) 整合寸法指定値による性能差

配列の大きさによっては,キャッシュラインのスラッシング[3]の影響で性能に差異が出ます。そのため，整合寸法の大きさによって性能に差異が出ることがあります。

3 次元複素フーリエ変換での入出力データー形式の例を図 7.1 に示します。

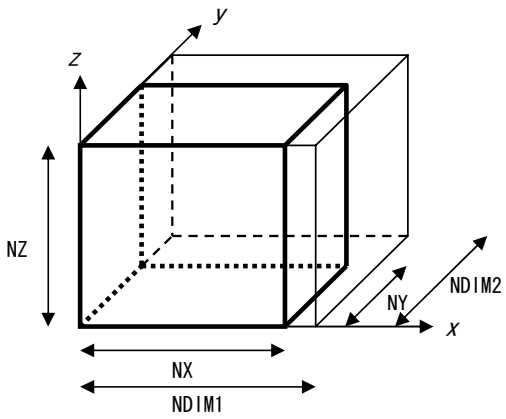


図 7.1 入出力データー形式の例

図 7.1 に示すとおり，実際に変換を行う 3 次元配列の範囲は NX×NY×NZ ですが，NX に対して NDIM1(NDIM1≧NX),NY に対して NDIM2(NDIM2≧NY)を使用し,確保する配列の大きさは NDIM1×NDIM2×NZ で指定します。

NX=NY=NZ=512 の場合の性能比較を図 7.2 に示します。

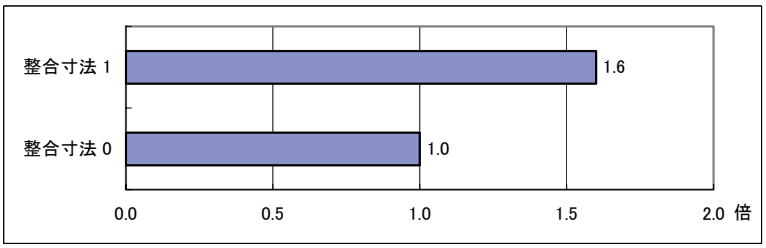


図 7.2 性能比較

図 7.2 は，整合寸法 0 が NDIM1=NX,NDIM2=NY，整合寸法 1 が NDIM1=NX+1，NDIM2=NY+1 で，縦軸に整合寸法の違いを，横軸に整合寸法 0 の場合の計算時間を 1 とし，整合寸法 0 の場合の計算時間を異なる整合寸法の場合の計算時間で割った整合寸法による性能改善効果を示した図です。整合寸法を調整することで，整合寸法 0 に比べて整合寸法 1 では約 1.6 倍となります。

2 基底の 3 次元複素フーリエ変換では，変換点数が 2 べキとなるためにキャッシュラインのスラッシングが発生して性能が劣化する可能性が高くなります。そのため，整合寸法に奇数値を指定することでキャッシュラインのスラッシングを大幅に低減することができます。

7．参考文献

[1] 森　正武，名取　亮，鳥居　達生，岩浪講座　情報科学－18　数値計算，岩波書店(1982)，pp.160-162

[2] 小国　力，村田　健郎，三好　俊郎，Dongarra,J.J.，長谷川　秀彦，行列計算ソフトウェア，丸善(1991)，pp.105-111

[3] Kevin Dowd 著，久良知　真知子訳，ハイ・パフォーマンス・コンピューティング，オーム社(1992)，pp.53-60

[4] (株)日立製作所，SR11000 行列計算副プログラムライブラリ MATRIX/MPP (3000-3-C87)，(2007)

[5] (株)日立製作所，SR11000 行列計算副プログラムライブラリー疎行列解法 MATRIX/MPP/SSS (3000-3-C88)，(2007)

[6] (株)日立製作所，SR11000 数値計算副プログラムライブラリ MSL2 行列計算 (3000-3-C83)，(2004)

[7] (株)日立製作所，SR11000 数値計算副プログラムライブラリ MSL2 関数計算 (3000-3-C84)，(2004)

[8] (株)日立製作所，SR11000 数値計算副プログラムライブラリ MSL2 統計計算 (3000-3-C85)，(2004)

[9] (株)日立製作所，SR11000 数値計算副プログラムライブラリ MSL2 操作 (3000-3-C86)，(2004)