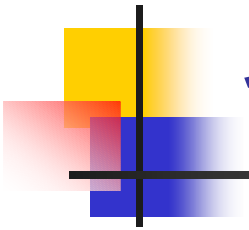
大規模科学計算向け
汎用数値ソフトウェア基盤の開発

西田 晃
中央大学21世紀COEプログラム ／ JST CREST
額田 彰　小武守 恒　梶山 民人
JST CREST ／ 東京大学



背景

- 従来の研究手法
  ～インターネット登場以前～
  - 各分野で独自にライブラリを開発
    （実は共通の問題）
  - 大型計算機センター内で公開･登録ユーザが利用
  - 商用ライブラリ
    - 計算機ベンダ製ライブラリ
      （特定機種向け）
    - サードパーティ製汎用ライブラリ
      （ワークステーション･小規模共用環境向け）
- 計算機環境の大規模化･高並列化
  - スケーラブルなソフトウェア共通基盤の普及が急務

これまでの研究開発

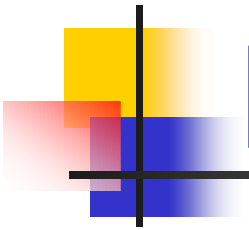
- 海外（主に米国）
  - 汎用数値ライブラリ
    - Netlib (BLAS, LAPACK, etc.) , Matlab
    - PETSc, Aztec/Trilinos, FFTW, ScaLAPACKDOE SciDAC, DARPA HPCS
  - 構造解析
    - NASTRAN
- 国内
  - 汎用数値ライブラリ
    - PARCEL（Fortran77/C）
  - 構造解析
    - GeoFEM, ADVENTURE, HPC-MW, PCP
  - 流体
    - UPACS

計算機環境の変化(1)

- 高速ネットワークの普及
  - コンピュータの処理能力は18ヶ月で2倍に
    (ムーアの法則)
  - 通信網の帯域幅は6ヶ月で2倍に
    (ギルダーの法則)
- デスクトップ PCからスーパーコンピュータまで，多様な資源がネットワーク上で共有できるようになった
- オープンな環境で研究開発が可能に
  - GNU
  - Linux
  - グリッドコンピューティングなど

計算機環境の変化(2)

- 並列処理の普及
  - 従来
    - プロセッサ単体の性能向上
    - ベクトルスパコン — 国産ベンダが市場を席巻
  - MPI の策定
    - PC クラスタの台頭 — 米国ベンダが市場を席巻
  - マルチコア化
    - Cell — 今後は？
- 並列計算機上での科学技術計算が一般的に
  - 広く使ってもらうには
    - 移植性をどう確保するか
    - 実装手法に関する研究（コンピュータサイエンス）の必要性
  - まずは主要な分野から国産ライブラリを
    - 行列反復解法
    - 高速関数変換(FFTなど)
- 平成14年度に JST CREST プロジェクトとして採択



開発方針

- 海外では？
  - US DOE
    - ASCI [Accelerated Strategic Computing Initiative]
      -> ASC [Advanced Simulation & Computing]
    - SciDAC [Scientific Discovery through Advanced Computing]
      -> 2006年から SciDAC-2 が発足
  - US DOD
    - HPCS [High Productivity Computing Systems] Program
    - 2006年12月に Cray, IBM を開発ベンダとして選定
- 本プロジェクトでは…
  - ソフトウェア開発に限定
  - 開発プラットフォームをどうするか
    - オープンな開発環境が提供されていること（移植性の観点から）
    - コストパフォーマンスが良いこと（将来的に普及する可能性が高いため）

実施体制

- メンバー
  - 専任研究員 ＋大学，研究機関からの参加メンバー
- 開発環境
  - 5年後に広く普及しているアーキテクチャを選ぶ必要
  - 原則として実装手法研究グループが CREST 予算により選定・導入
  - 一部は企業との共同研究契約による無償利用
- 開発スタイル
  - web サーバを設置
  - フリーソフトウェアとして2005年9月に初版を公開･国内向けに ML 等でアナウンス
  - 基礎研究と国内外のユーザからのフィードバックをもとに改良･機能追加
  - 2006年11月に SC06 向けに新版を公開
  - 国内･海外向けに ML 等でアナウンス

主な研究成果

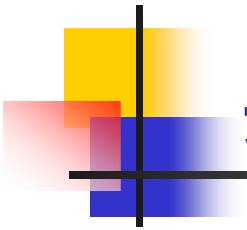
- 論文発表
  - 27件
- 口頭発表
  - 国内36件，海外16件
- 特許出願
  - 国内･海外各1件

実装手法(1)

- 開発環境の選択基準
  - Linux OS の積極採用
    - 移植性, オープン性を評価
      - SGI Altix, Cray XT3, IBM Blue Gene, IBM OpenPower, SCE PLAYSTATION3
      - PC クラスタ
  - コストパフォーマンス
    - 将来的な普及率を念頭に
      - SGI Altix (2003年7月導入. Intel Itanium ベース. SGI はその後 MIPS から Itanium に完全移行. 2006/09/11)
      - Cray XT3 (2005年3月導入. AMD Opteron ベース. Red Storm の商用化. Cray はその後 XT4 を発表. 2006/11/13.)
    - 消費電力
      - IBM Blue Gene (2003年11月 IBM Watson 研究所と共同研究開始. IBM PowerPC 440 搭載.)
      - マルチコア化の進展(SGI Altix で共有メモリ環境に対応済み)

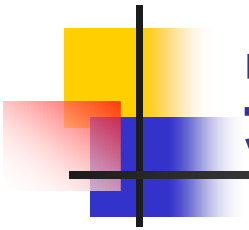
実装手法(2)

- TOP500 Supercomputer Sites (2006年11月版)
  - 1 DOE/NNSA/LLNL United States BlueGene/L - eServer Blue Gene Solution IBM
  - 2 NNSA/Sandia National Laboratories United States Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core Cray Inc.
  - 3 IBM Thomas J. Watson Research Center United States BGW - eServer Blue Gene Solution IBM
  - 4 DOE/NNSA/LLNL United States ASC Purple - eServer pSeries p5 575 1.9 GHz IBM
  - 5 Barcelona Supercomputing Center Spain MareNostrum - BladeCenter JS21 Cluster, PPC 970, 2.3 GHz, Myrinet IBM
  - 6 NNSA/Sandia National Laboratories United States Thunderbird - PowerEdge 1850, 3.6 GHz, Infiniband Dell
  - 7 Commissariat a l'Energie Atomique (CEA) France Tera-10 - NovaScale 5160, Itanium2 1.6 GHz, Quadrics Bull SA
  - 8 NASA/Ames Research Center/NAS United States Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband SGI
  - 9 GSIC Center, Tokyo Institute of Technology Japan TSUBAME Grid Cluster - Sun Fire x4600 Cluster, Opteron 2.4/2.6 GHz and ClearSpeed Accelerator, Infiniband NEC/Sun
  - 10 Oak Ridge National Laboratory United States Jaguar - Cray XT3, 2.6 GHz dual Core Cray Inc.
- トップ10はすべて対応済みのアーキテクチャ



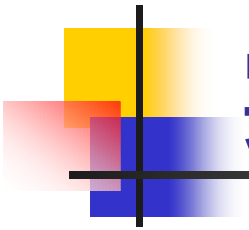
実装手法(3)

- スケーラビリティ
  - どこでテストするか
  - IBM Blue Gene → 65536 CPU まで拡張可能（LLNL IBM Blue Gene/L）
  - NEC SX → 5120 CPU まで拡張可能（地球シミュレータ ）
- IBM Blue Gene → IBM T. J. Watson Research Center, ニイウス株式会社との共同研究契約により対応
- 地球シミュレータ → NEC SX-6i の導入，共同利用プロジェクト（平成18年度）により対応



実装手法(4)

- ソフトウェア工学の観点から
  - 生産性の向上
    - 再利用性，移植性を重視
  - オブジェクト指向の導入(Lis)
    - 並列処理記述部分の隠蔽
    - 逐次版，OpenMP版，MPI 版とも同一のAPIによりプログラミングが可能
  - スクリプト型プログラミング言語の開発(SILC)
    - 多様な分散計算環境，数値計算ライブラリに対応済み
  - オートチューニング機能の導入 (FFTSS)
    - 計算環境に合わせて自動的に実行プランを構築



実装手法(5)

- 今年度の計画
  - 行列計算ライブラリインタフェース SILC について、
    - ベクトル計算機
    - 大規模分散並列環境

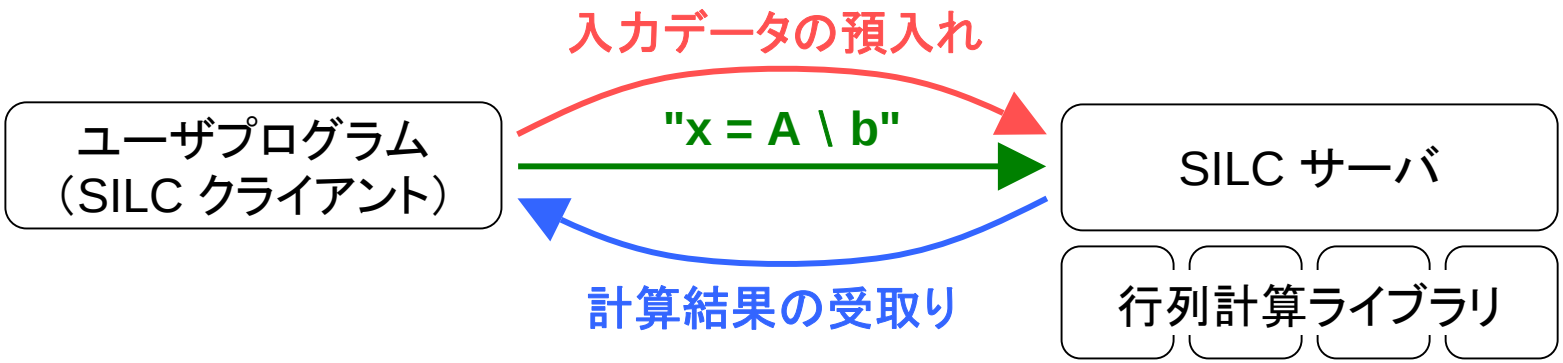
    に対応し，システムの実装を進める

実装手法(6)
行列計算ライブラリインタフェースSILC(1)

- Simple Interface for Library Collections
- 特定のライブラリ、計算環境、プログラミング言語に依存しないインタフェース
  - ターゲット:行列計算ライブラリを呼び出す C や Fortran のプログラム
  - 従来のライブラリ呼出しを置き換えるシステム
- 使いやすさ、便利さが目的

実装手法(7)
行列計算ライブラリインタフェースSILC(2)

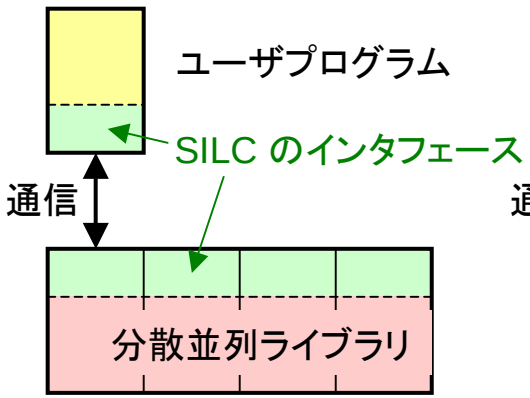
- 次の3ステップで行列計算ライブラリを利用
  1. 入力データの預入れ
  2. 文字列(数式)による計算の指示
  3. 計算結果の受取り



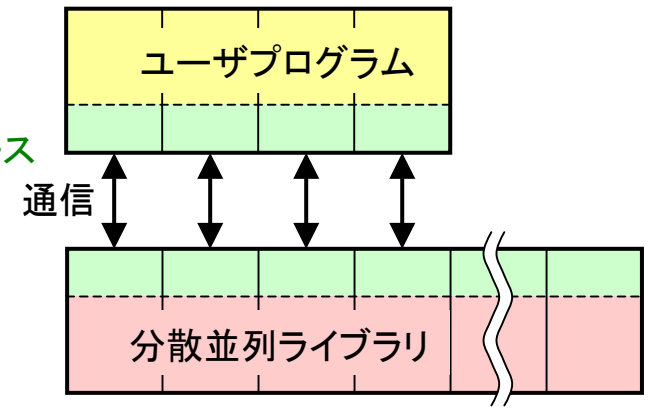
実装手法(8)
成果(1)

- NQS 等のバッチ処理システムを採用した計算環境への対応
  - クライアント・サーバ方式によらない SILC アプリ(ユーザプログラム)の作成方法の検討(下記構成(C))

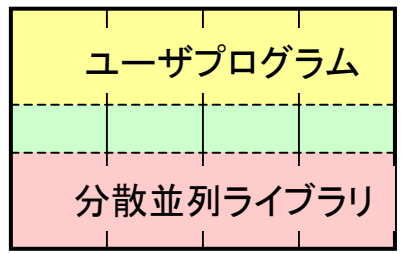
構成(A)
逐次クライアント＋分散並列サーバ



構成(B)
分散並列クライアント＋分散並列サーバ



構成(C)
SILC のインタフェースと共にライブラリをリンク



実装手法(9)
成果(2)

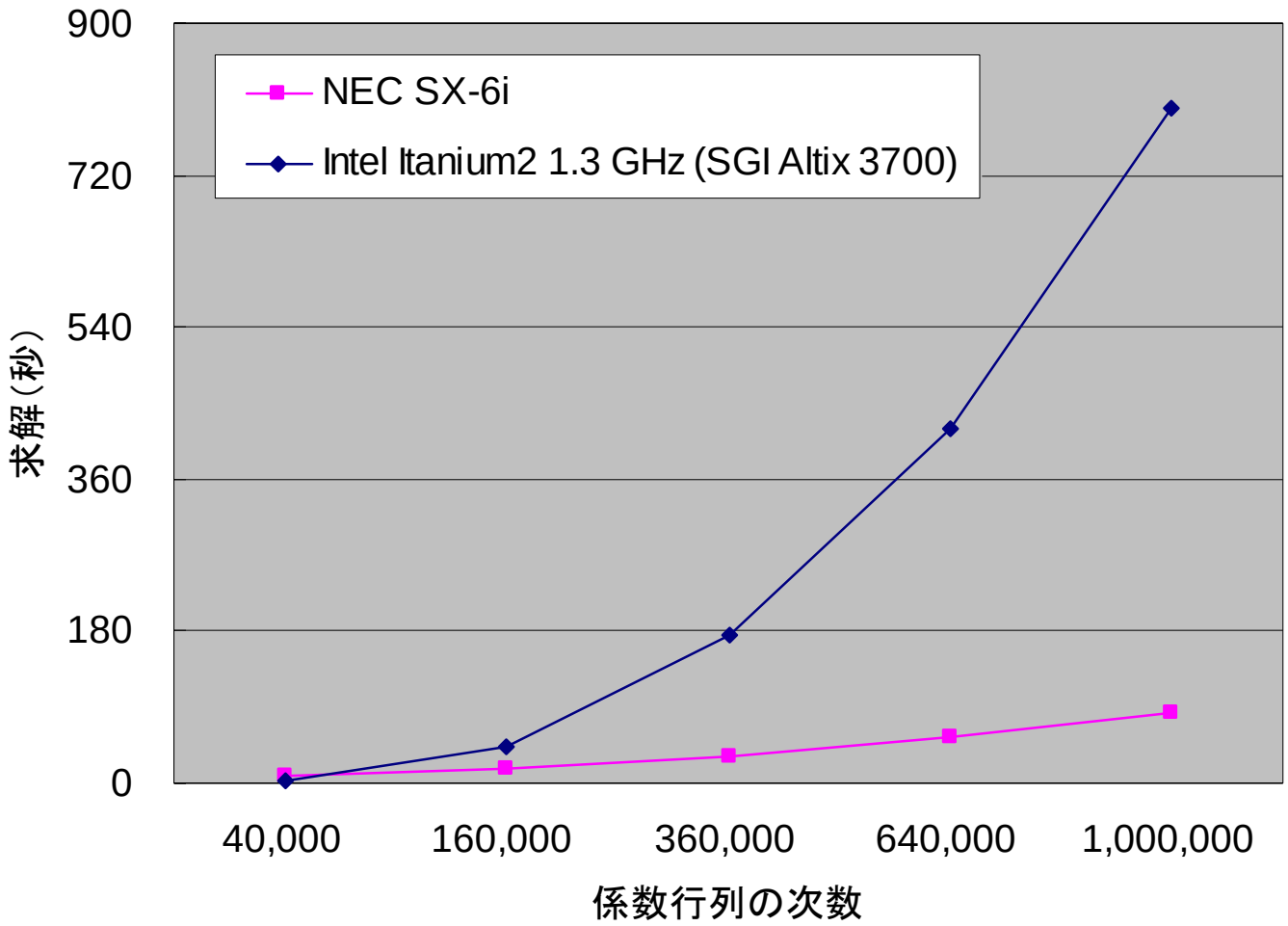
- 計算ノードの OS の機能に制限事項の多い計算環境への対応
  - マルチスレッド機能および共有ライブラリの動的リンク機能を用いない SILC サーバの開発
- ベクトル計算機への対応
  - ベクトル計算機向きの行列の格納形式 Jagged Diagonal Storage (JDS) 形式のサポート
  - SILC サーバのベクトル化：ソースコードへのベクトル化指示子の挿入
  - 実機 (NEC SX-6i) での動作検証および予備性能評価

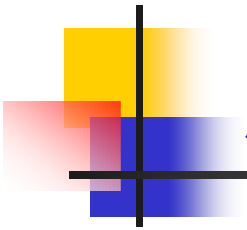
実装手法(10)
予備性能評価(1)

- SILC の数式で実現した CG 法
  - 反復と条件分岐はユーザプログラムの記述言語で実現
  - 行列 *A* の格納形式や精度によらず動作
- 数値実験
  - 2次元のラプラス方程式を5点中心差分で離散化
  - 行列の格納形式は JDS 形式

```
/* A: 係数行列, b: 右辺ベクトル */
SILC_EXEC("rho_old = 1.0");
SILC_EXEC("n = length(b)");
SILC_EXEC("p = zeros(n, 1)");
SILC_EXEC("x = zeros(n, 1)");
SILC_EXEC("r = b");
SILC_EXEC("bnrm2 = 1.0 / norm2(b)");
for (i = 1; i <= maxiter; i++) {
  SILC_EXEC("rho = r' * r");
  SILC_EXEC("beta = rho / rho_old");
  SILC_EXEC("p = r + beta * p");
  SILC_EXEC("q = A * p");
  SILC_EXEC("alpha = rho / (p' * q)");
  SILC_EXEC("r = r - alpha * q");
  SILC_EXEC("nrm2 = norm2(r) * bnrm2");
  SILC_EXEC("x = x + alpha * p");
  /* 収束判定 */
  object.v = &nrm2;
  SILC_GET(&object, "nrm2");
  if (nrm2 <= EPSILON)
    break;
  SILC_EXEC("rho_old = rho");
}
/* x: 解ベクトル */
```

実装手法(11)
予備性能評価(2)





反復解法(1)

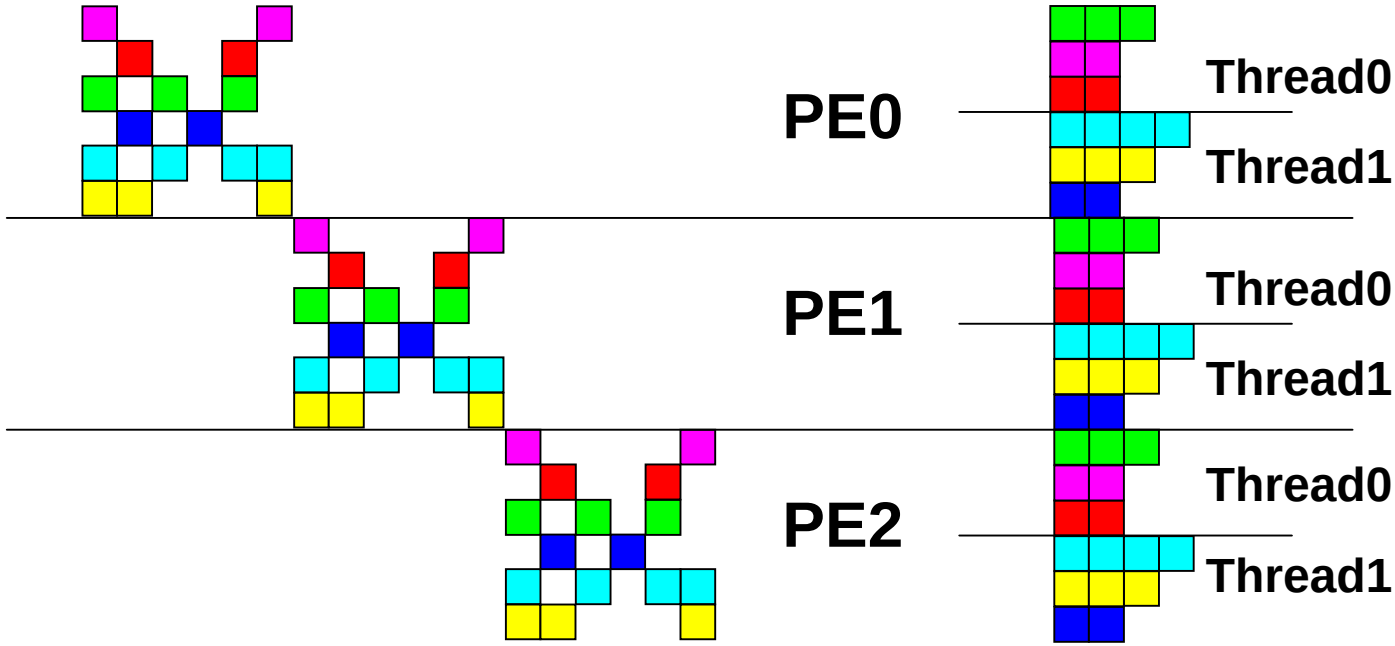
- 連立一次方程式
  - 科学技術計算では最もよく解かれる問題
  - 実問題では疎行列を扱うことが多い
    - 有限差分法，有限要素法，…
- 並列環境でのスケーラビリティを重視
  - 数千CPU規模の計算環境では切実な問題
  - 反復解法は直接解法に比べて並列化が容易
- 共役勾配法系の解法に重点
  - 研究成果(Krylov 部分空間アルゴリズム，前処理手法など)を早い段階で広く利用してもらう
  - 研究との相乗効果

反復解法(2)

- 平成17年9月に Lis (A Library of Iterative Solvers for linear systems) 1.0.0 を公開
  - 多数の反復解法アルゴリズムと前処理，格納形式に対応
    - 計算機環境に合わせて最適な格納形式，解法を選択可能
  - OpenMP(共有並列環境向け)，MPI(分散並列環境向け)を用いた並列処理に対応
- 平成18年11月には Lis 1.1.0 beta1 を公開
  - MPI 版は IBM Watson 研究所の BGW (Blue Gene 20-rack system, LINPACK 性能 91.2TFLOPS) にて動作検証を完了 (平成18年3月30日．16384CPUまで線形な性能向上を達成)
  - AMG 前処理 (1.1.0 alpha)
    - 2次元Poisson問題(10000X10000)、Blue Gene 512ノード上
      - 0.083秒(反復回数17回)　前処理付きCG
      - 2.36秒(反復回数228回)　前処理なしCG

反復解法(3)
MPI＋OpenMPのハイブリッド化

- JDS(Jagged Diagonal Storage)形式
  - ベクトル計算機向けの格納形式



反復解法(4)
格納形式変換

- CRS形式からJDS形式への格納方式変換のベクトル化
  - ディレクティブの挿入
  - ループの入れ替え

```
for(i=is;i<ie;i++)  {
  js = Ain->ptr[perm[i]];
  je = Ain->ptr[perm[i]+1];
  for(j=js;j<je;j++) {
    l = ptr[my_rank*(maxnzr+1) + j-js]
        +i-is;
    value[l]   = Ain->value[j];
    index[l]   = Ain->index[j];
  }
}
```

```
for(j=0;j<maxnzr;j++) {
  js = ptr[my_rank*(maxnzr+1) + j];
  je = ptr[my_rank*(maxnzr+1) + j+1];
  #pragma cdir nodep
  for(i=js;i<je;i++) {
    l = Ain->ptr[perm[i-js]] + j;
    value[i]   = Ain->value[l];
    index[i]   = Ain->index[l];
    }
}
```

反復解法(5)
通信のオーバーラップ

- 行列AをA=D-L-Uに分離
  - DはAの対角部分
  - -L,-UはAの下三角部分と上三角部分
- 行列ベクトル積y=Axと通信をオーバーラップ
  - (D-L)xに必要なxの通信
  - Uxに必要なxの通信
  - y = (D-L)x
  - y += Ux

オーバーラップ

- 実装途中

高速フーリエ変換(1)

- 平成17年9月に FFT ライブラリ FFTSS 1.0 を公開.
  - スーパースカラ型プロセッサを対象
  - キャッシュメモリに格納可能なサイズの最適化 FFT と多次元 FFT との組み合わせで構成
  - 移植性を重視
    - プログラムによる最適化とコンパイラによる最適化機能の双方を活用
    - アーキテクチャに応じてカーネルを切り替え
    - プロセッサ拡張機能(SIMD(Single Instruction Multiple Data)演算命令など)の積極的な利用
    - ひねり係数テーブルの最適化
    - MIT FFTW とのインタフェース互換性. パフォーマンスで独自性を追求.
- 平成18年11月に FFTSS 2.0 を公開
  - 多次元並列 FFT に対応
- 産総研(生命情報科学研究センター)との共同研究

高速フーリエ変換(2)
高性能2次元FFTの実現

FFTの地球シミュレータへの最適化としては

- 演算の完全なベクトル化
- ノード内並列化
- ノード間通信

の3点が必要となる。
これらの条件を全て整えることによって高い
スケーラビリティが得られた。
512ノードを用い、1ノードの約370倍を達成。

高速フーリエ変換（3）
ベクトル化（1）

N×Nの2D-FFTをPノードで計算する時、

各ノード内で行う計算：

N／P組のN点FFT

Multirow FFTにより容易にベクトル化可能。

ベクトル長はN／P以上であるが、

ノード数が大きくなるとベクトル長が短くなり

効率が低下

⇒　ループ交換を用いることで緩和。

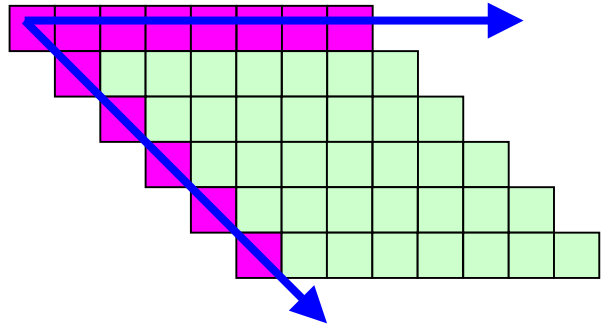
高速フーリエ変換(4)
ベクトル化(2)

主記憶へのベクトルアクセス

ストライド間隔:

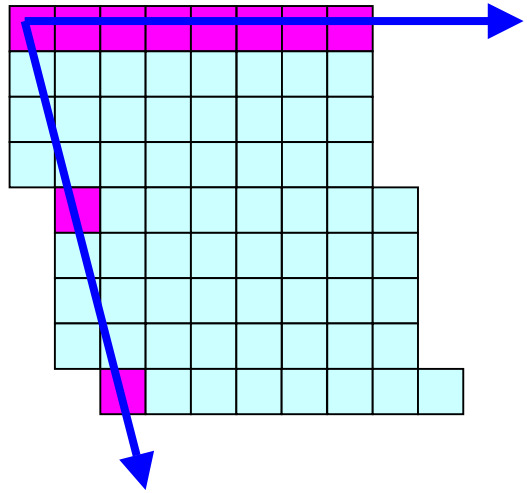
　奇数×8byte が望ましい

⇒パディングを挿入して調整。



ループ交換も考慮し、両方の次元方向のストライドを調整。

(パディングを含めた状態でノード間通信も行う)



高速フーリエ変換（５）
ノード内並列化

ノード内の並列化にはOpenMPを使用。
（フラットMPIの場合）
ノード内の余分な通信が必要　×
ノード数が増えてベクトル長が短くなる　×

N／P組のN点FFTを8PEで分担。
N／8P組づつ　⇒　ベクトル長が短く・・・　×
一つ内側のループを並列化
必要なPE間の同期は高々O(log(N))回。

高速フーリエ変換（6）
ノード間通信

ノード間通信のバンド幅は12．3GB/sと高い。

ノードの演算性能も64GFLOPSと高い。

演算：通信の比は$N^2 \log N : N^2$。

Nが大きくなる

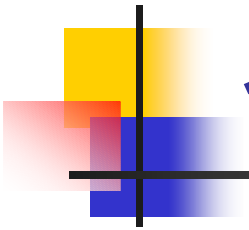
⇒通信に要する時間の割合が相対的に低下。

全対全通信ではK番目に（ID+K）%Pのノードへ送信するシンプルな方式を採用。

MPI_Isend()/MPI_Irecv()ではうまくスケジューリングされず、MPI_Put()を用いた。

現状と課題

- 使ってもらうことの重要性
  - 全ソフトウェアについて和文・英文マニュアルを用意
  - インターネット（web，ML，…）を最大限に活用
    - 国内外の主要なメーリングリストを通じてアナウンス
  - 公開以来，2006年12月24日現在で513件のダウンロード
- 応用分野との連携
  - フィードバックを得るためには
    - ワークショップ，学会等を積極的に活用すべき



今後のスケジュール（1）

- 実装手法
  - Fortran インタフェースへの完全対応
    - ライブラリは C 言語で記述
    - Lis , SILC は対応完了
  - SILC
    - スクリプト言語としての機能の充実
    - MPI環境への完全対応（現在は ScaLAPACK, Lis など一部のライブラリに対応）
    - 特殊な計算環境への対応
      - 演算コプロセッサのサポート（GRAPE, ClearSpeed, GPU など）
  - 並列言語への対応
    - 現状は OpenMP, MPI で記述
    - Co-Array Fortran, Unified Parallel C にも対応予定

今後のスケジュール（2）

- 反復解法
  - 地球シミュレータの完全サポート
  - 理論分野，アプリケーション分野との連携強化
  - 直接解法のサポート
  - 演算コプロセッサへの対応（単精度演算を積極的に活用）
- 高速フーリエ変換
  - 分散メモリ環境のライブラリとしてのサポート
  - 機能の充実（現在は2冪のみ）