反復解法ライブラリ Lis
（反復解法研究グループ）

担当：小武守恒（JST CREST 研究員）

Lis (a Library of Iterative Solvers for linear systems)

- C言語で記述されたライブラリ
- AMG前処理ルーチンを含むLis-AMGでは、Fortran 90も使用
- 現バージョンではC言語からの呼び出しにのみ対応
- 並列計算には、OpenMP または MPI-1 を使用

特徴

- 様々な格納形式の入力を受け付ける
- 様々な解法と前処理を組み合わせて使える
- 逐次環境から共有メモリまたは分散メモリ環境への移行はプログラムの変更なし（あるいはごくわずかの変更）でよい
- 反復解法、前処理の選択は、コマンドラインからも選択可能

公式サイト

http://ssi.is.s.u-tokyo.ac.jp/lis/

実装されている解法・前処理・格納形式

解法

対称	CG
非対称	BiCG
	CGS
	BiCGSTAB
	BiCGSTAB(l)
	GPBiCG
	Orthomin(m)
	GMRES(m)
	QMR
	Jacobi
	Gauss-Seidel
	SOR

格納形式

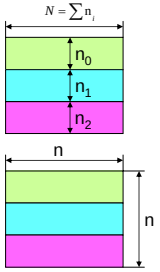
Point	Compressed Row Storage	(CRS)
	Compressed Column Storage	(CCS)
	Modified Compressed Sparse Row	(MSR)
	Diagonal	(DIA)
	Ellpack-Itpack generalized diagonal	(ELL)
	Jagged Diagonal	(JDS)
	Dense	(DNS)
	Coordinate	(COO)
Block	Block Sparse Row	(BSR)
	Block Sparse Column	(BSC)
	Variable Block Row	(VBR)

前処理

- Jacobi
- ILU(k)
- SSOR
- Hybrid　SORなどの反復法を用いる
- I+S　定常反復解法に有効
- SA-AMG　smoothed aggregationに基づく代数的マルチグリッド
- SAINV　近似逆行列

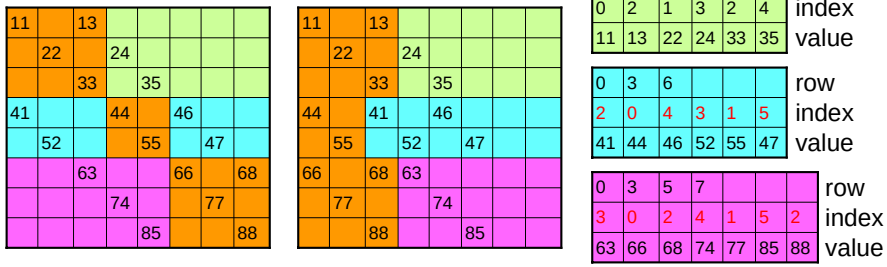
行列の作成

- lis_matrix_create(int local_n, int global_n, LIS_Comm comm, LIS_MATRIX *Amat)
- 2通りの作成方法
  - 部分行列の大きさを指定
    - lis_matrix_create(ni,0,comm,&A);
    - 各プロセッサにnp x Nの部分行列を作成
  - 全体行列の大きさを指定
    - lis_matrix_create(0,n,comm,&A);
    - 各プロセッサにmp x nの部分行列を作成



配列の関連付け

- lis_matrix_set_crs(int nnz, int *row, int *index, LIS_SCALAR *value, LIS_MATRIX A)
- 行列の列（行）番号は全体行列の番号



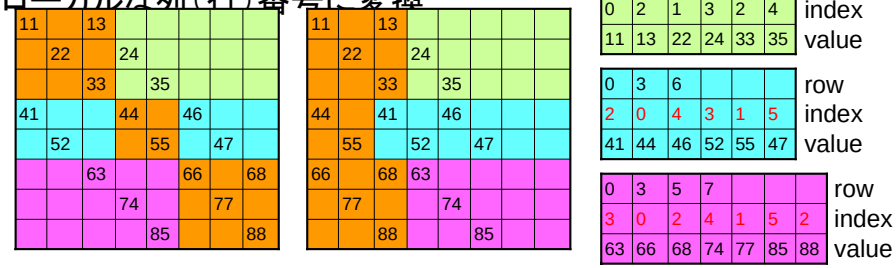
0	2	4	6			row
0	2	1	3	2	4	index
11	13	22	24	33	35	value

0	3	6				row
2	0	4	3	1	5	index
41	44	46	52	55	47	value

0	3	5	7				row
3	0	2	4	1	5	2	index
63	66	68	74	77	85	88	value

行列の組み立て

- lis_matrix_assemble(LIS_MATRIX *A, int matrix_type, ...)
- 要素の並べ替え
- ローカルな列（行）番号に変換

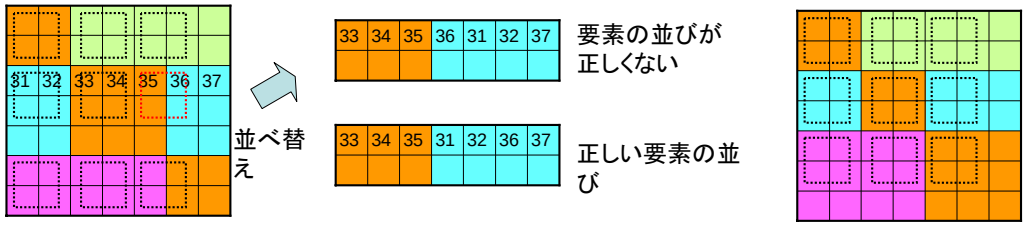


0	2	4	6			row
0	2	1	3	2	4	index
11	13	22	24	33	35	value

0	3	6				row
2	0	4	3	1	5	index
41	44	46	52	55	47	value

0	3	5	7				row
3	0	2	4	1	5	2	index
63	66	68	74	77	85	88	value

- 対角ブロックをまたぐ場合
  - p-1までの対角ブロックが列ブロック数で割り切れるように再分散



プログラム例

- 以下のプログラムは、CRS形式の行列 A を作成し、線型方程式 Ax=b を指定の解法、前処理で解くプログラムである。ただし、右辺ベクトル b は解ベクトルの値がすべて1となるように設定。

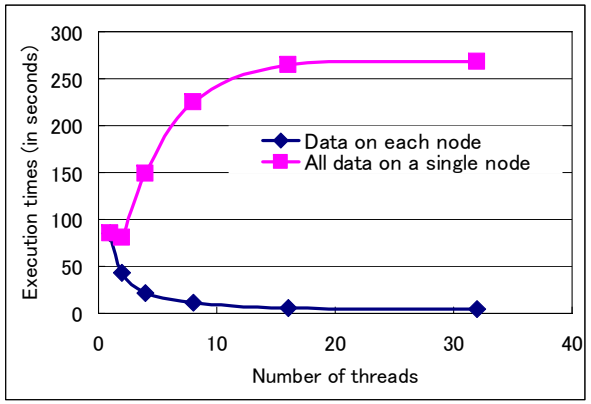
```
int main(int argc, char* argv[]) {
  LIS_MATRIX    A;
  LIS_VECTOR    b,x,u;
  int           n,gn,options[LIS_OPTIONS_LEN];
  LIS_SCALAR    params[LIS_PARAMS_LEN];
  LIS_SCALAR    status[LIS_STATUS_LEN];

  lis_initialize(argc, argv, NULL, options, params);   nn = n*m;
  lis_matrix_create(0,nn,LIS_COMM_WORLD,&A);
  /* malloc ptr,index and value */
  lis_matrix_get_range(A,&is,&ie);
  /* fill ptr,index and value */
  lis_matrix_set_crs(ptr[ie-is],ptr,index,value,A);
  lis_matrix_assemble(&A,LIS_MATRIX_CRS);

  n  = A->n; gn = A->gn;
  lis_vector_create(0,gn,LIS_COMM_WORLD,&u);
  lis_vector_duplicate(u,&b); lis_vector_set_all(1.0,u);
  lis_matvec(A,u,b);
  lis_vector_duplicate(u,&x); lis_vector_set_all(0.0,x);
  lis_solve(A,b,x,params,options,status);
  lis_finalize();
}
```

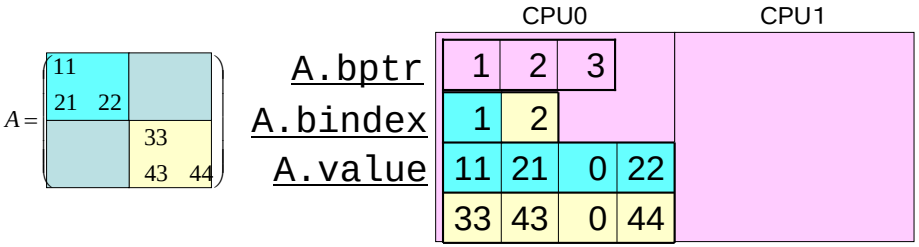
共有メモリ型計算機に対する並列化

- OpenMPによる並列化
- First-touchに対応
  - データは最初にアクセスしたプロセスのローカルメモリに配置される
  - First-touch未対応だと性能が大幅に低下

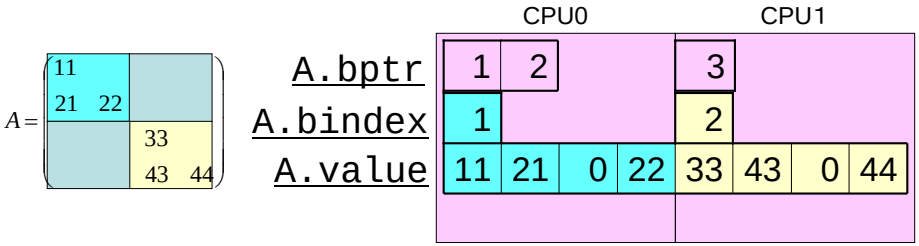


  - 格納形式変換ルーチンを並列化することで対応

逐次変換ではCPU0にすべてデータが配置されてしまう



並列変換では各スレッドに必要なデータが配置される

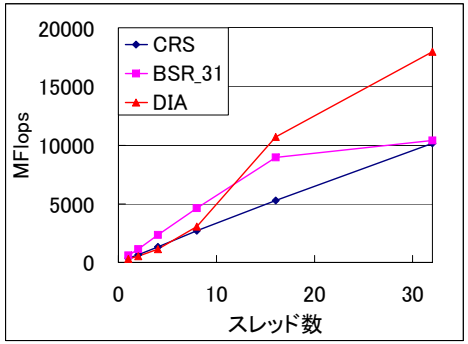


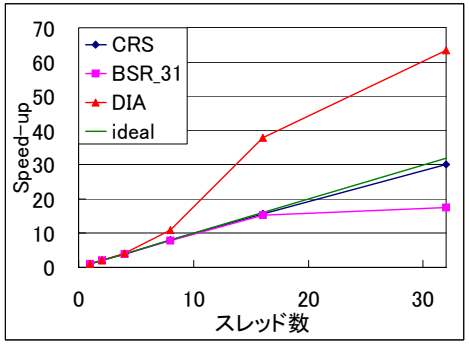
数値実験

- 行列ベクトル積の性能を評価する
  - 反復解法の性能に最も影響を与えるため
- 行列： 3次元ポアソン方程式を有限要素法で離散化

次数	1，000，000
非零要素数	26，463，592
平均非零要素数	26．46
メモリ量（CRS）	306.7MB
メモリ量（BSR_31）	394.9MB
メモリ量（DIA）	206.0MB
メモリ量（BSR_31）	394.9MB

- 環境： 分散共有メモリ型計算機 SGI Altix3700
  Itanium2 1.3GHz, 32GB, SGI Advanced Linux2.4
  Intel Compiler C/C++8.1
- 結果： 十分なスケーラビリティが得られた



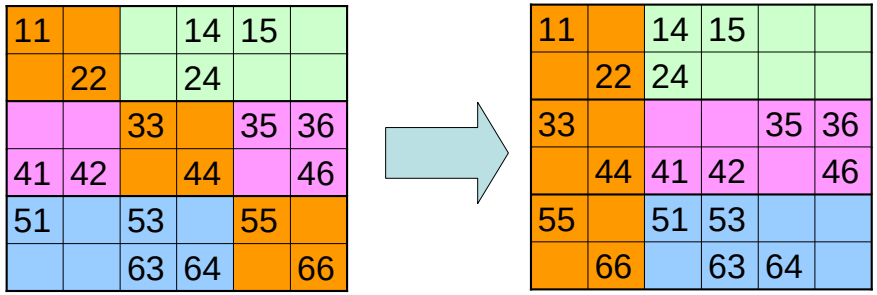


研究成果

- Hisashi Kotakemori, Hidehiko Hasegawa, Tamito Kajiyama, Akira Nukada, Reiji Suda and Akira Nishida
  Performance Evaluation of Parallel Sparse Matrix--Vector Products on SGI Altix3700, IWOMP2005
- Hisashi Kotakemori, Hidehiko Hasegawa and Akira Nishida
  Performance Evaluation of a Parallel Iterative Method Library using OpenMP, HPC Asia2005

分散メモリ型計算機に対する並列化

- MPIによる並列化
- 行ブロック分割（CCS,BSCは列ブロック分割）
- 通信テーブルの作成
  - 通信テーブルはGeoFEMを参考
  - 行列作成時にライブラリ側で自動生成



変数		PE0	PE1	PE2
neibpetot	通信を行うPE数	2	2	2
beibpe	PE番号	1,2	0,2	0,1
export_index	export_nodeの開始位置	0,2,3	0,1,3	0.1.3
export_node	送信する要素の位置	0,1,0	1,0,1	0,0,1
import_index	import_nodeの開始位置	0,1,2	0,2,4	0,1,2
import_node	受信する要素の位置	2,3	2,3,4,5	2,3,4

- 行列ベクトル積 y=Ax
  - 事前にベクトル x を通信テーブルにしたがって送受信
  - CCS,BSCは事後にベクトル y を通信テーブルの受信と送信を入れ替えて送受信

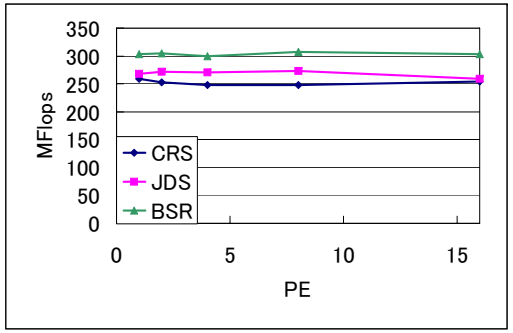
数値実験

- 行列ベクトル積の性能を評価する
- 行列： 2次元ポアソン方程式を5点中心差分で離散化
  - 各PEの行数一定（1,000,000）
- 環境：

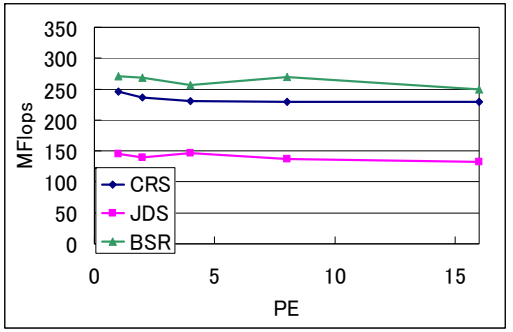
	SGI Altix3700	Cray XT3	Xeon Cluster (1000Base)	Opteron Cluster (InfiniBand)
CPU	Itenium2 1.3GHz	Opteron 2.0GHz	Xeon 2.8GHz	Opteron 2.0GHz
メモリ /node	2GB	2GB	1GB	2GB
コンパイラ	Intel C/C++8.1	PGI 6.0	Intel C/C++8.0	GCC 3.3.3

- 結果： 十分なスケーラビリティが得られた

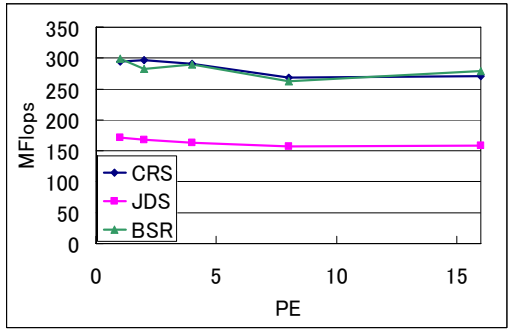
Altix



XT3



Xeon Cluster



Opteron Cluster