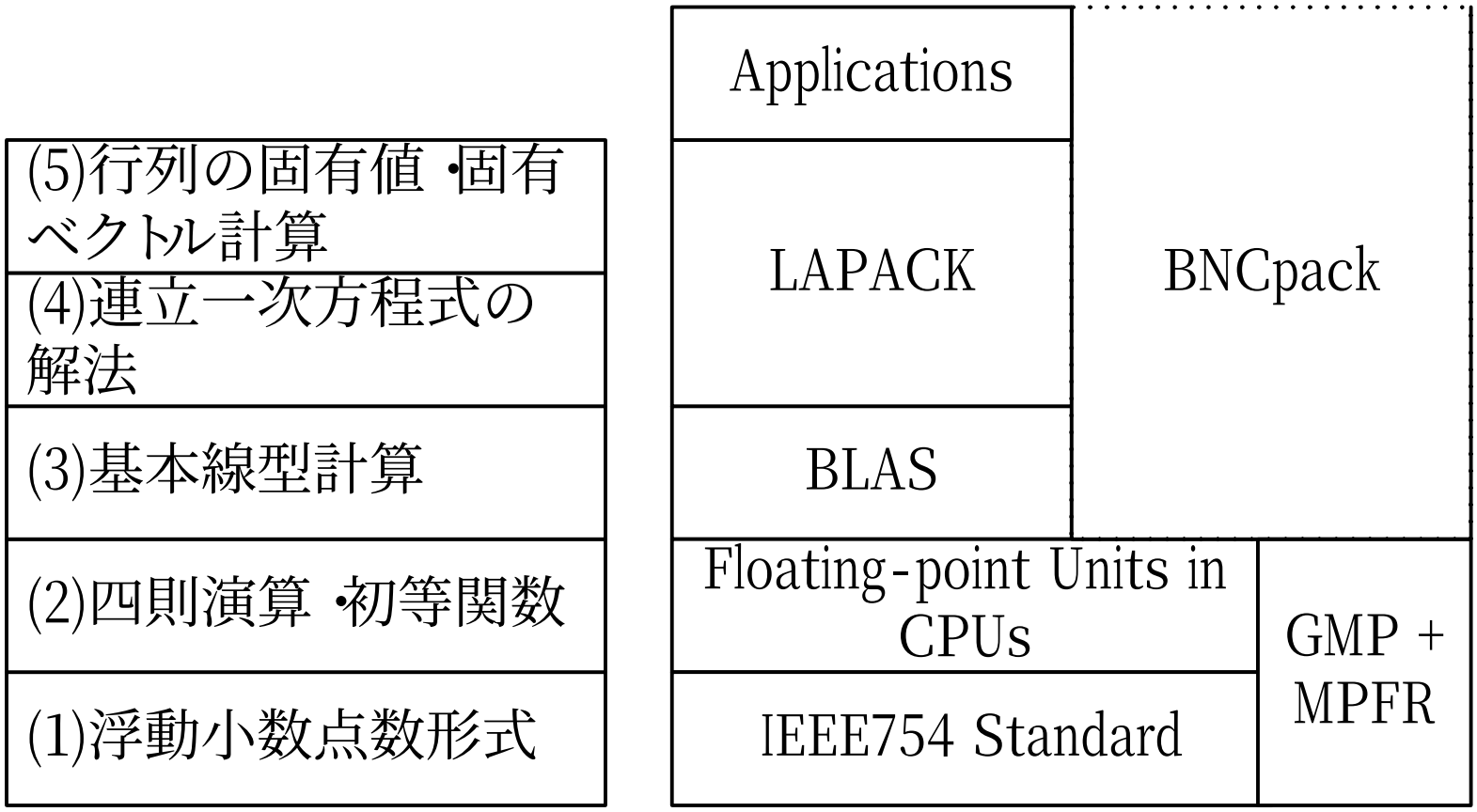
PC Cluster 上における多倍長数値計算ライブラリ BNCpack の並列分散化 ーMPIBNCpackについてー

幸谷智紀
静岡理工科大学
tkouya@na-net.ornl.gov

講演概要

1. 数値計算ソフトウェアの高速化
2. BNCpackとは？
3. BNCpack + MPI = MPIBNCpack
4. MPIBNCpackのプログラミングお作法
5. ベンチマークテスト
6. 今後の(アテにならない)予定
7. 蛇足
8. 参考文献

数値計算ソフトウェアの高速化 (1/2)



数値計算ソフトウェアの高速化 (2/2)

どのレベルでの高速化を図るか？

- 1CPU環境での高速化
  - 下部Layerの高速化は限界に近い
  - LAPACK/BLAS ・・・Cache Hit率の向上→ATLAS
- 複数CPU環境での高速化
  - SMP環境での高速化(共有メモリ)
  - PC Clusterでの高速化(分散メモリ)

BNCpackとは？

- 固定長・可変長浮動小数点演算両方のサポート
  - IEEE754 単精度・倍精度
  - GMP(GNU MP)+MPFRによる多倍長浮動小数点
- 主な機能
  - 基本線型計算
  - 連立一次方程式
  - 固有値問題
  - 非線型方程式
  - 代数方程式
  - 数値積分

・・詳細は, LC2002の資料参照。

BNCpack + MPI = MPIBNCpack (1/3)

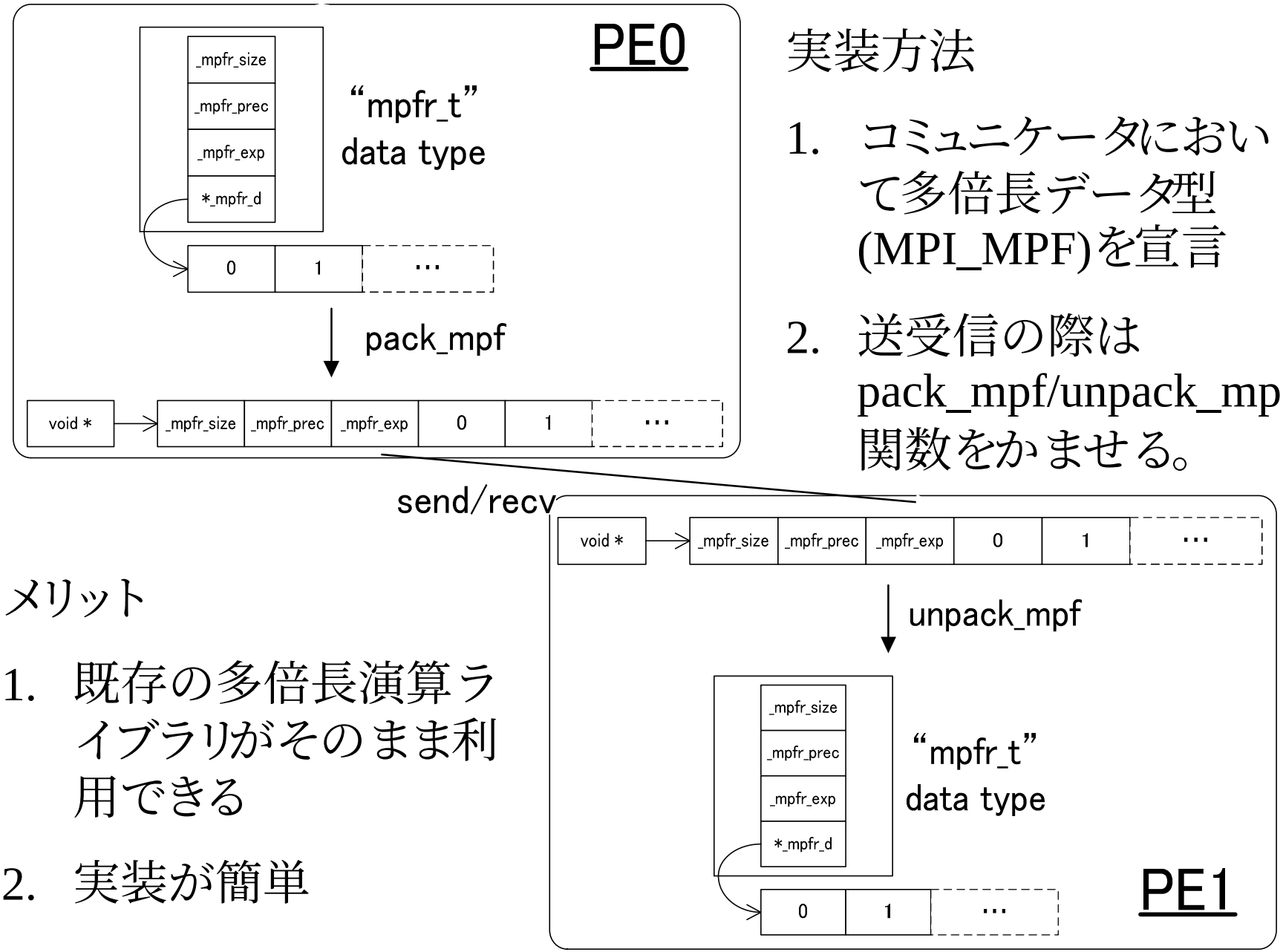
BNCpack ＋ MPI (mpich 1.2.5)

↓

MPIBNCpack

- IEEE754倍精度と多倍長精度をサポート
- 基本線型計算
  - 実ベクトル演算
  - 実正方行列演算
- 台形則(数値積分)
- CG法(連立一次方程式)
- DKA法(代数方程式)

http://na-inet.jp/na/bnc/



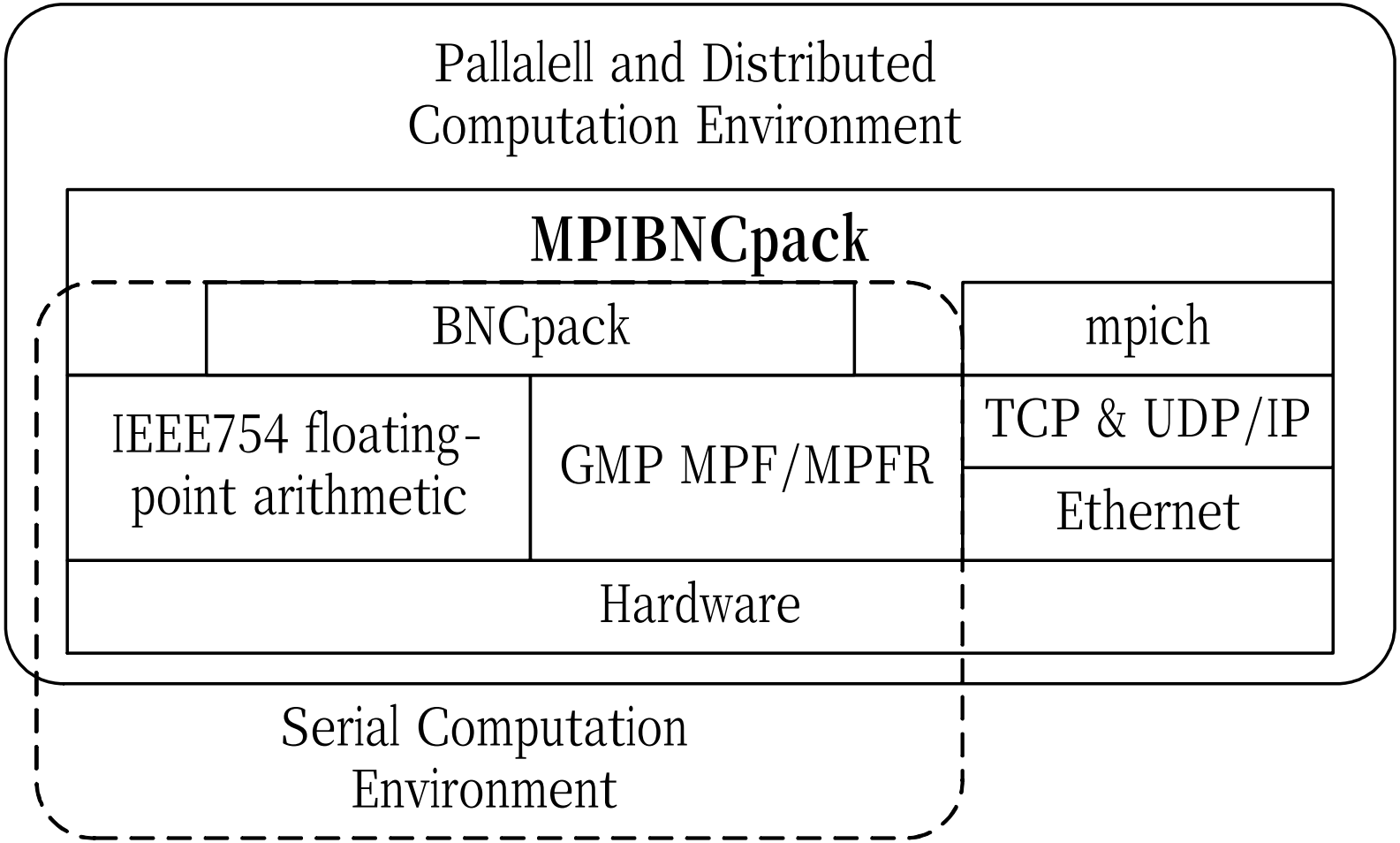
実装方法

1. コミュニケータにおいて多倍長データ型(MPI_MPF)を宣言
2. 送受信の際はpack_mpf/unpack_mpf関数をかませる。

メリット

1. 既存の多倍長演算ライブラリがそのまま利用できる
2. 実装が簡単

BNCpack + MPI = MPIBNCpack (3/3)



MPIBNCpackのプログラミングお作法

- 入出力はPE0で行う(そうしておくと楽)。
- データの分割・集約処理は関数にお任せ。

vec1.c with BNCpack

1. ベクトル初期化
2. ベクトル要素の代入
3. 計算
4. 表示
5. ベクトル消去

mpi_vec1.c with MPIBNCpack

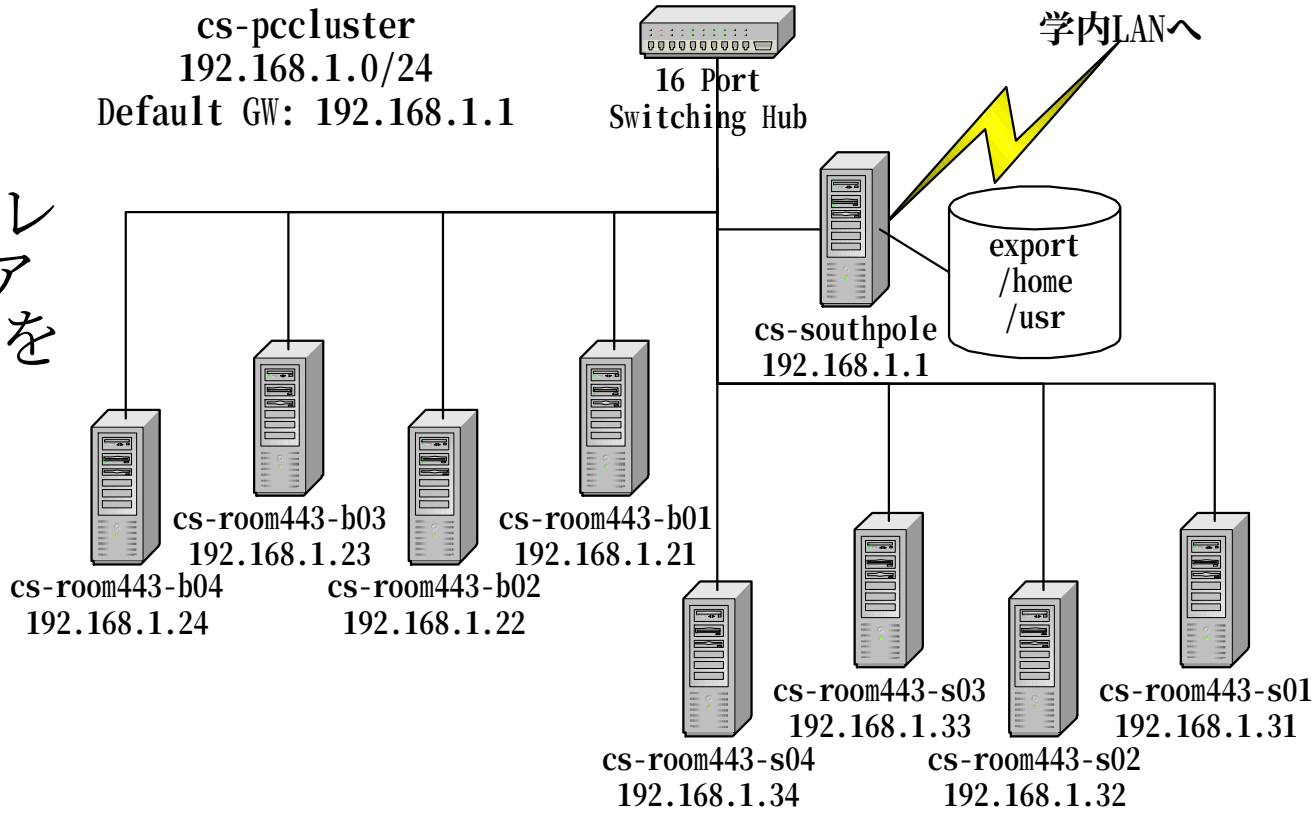
1. @PE0
   1. 初期化
   2. ベクトル要素の代入
2. 各PEで使用するベクトルの初期化
3. ベクトル要素の分散配置
4. 計算
5. ベクトル要素をPE0へ集約
6. ベクトル消去
7. @PE0
   1. ベクトル表示
   2. ベクトル消去

ベンチマークテスト

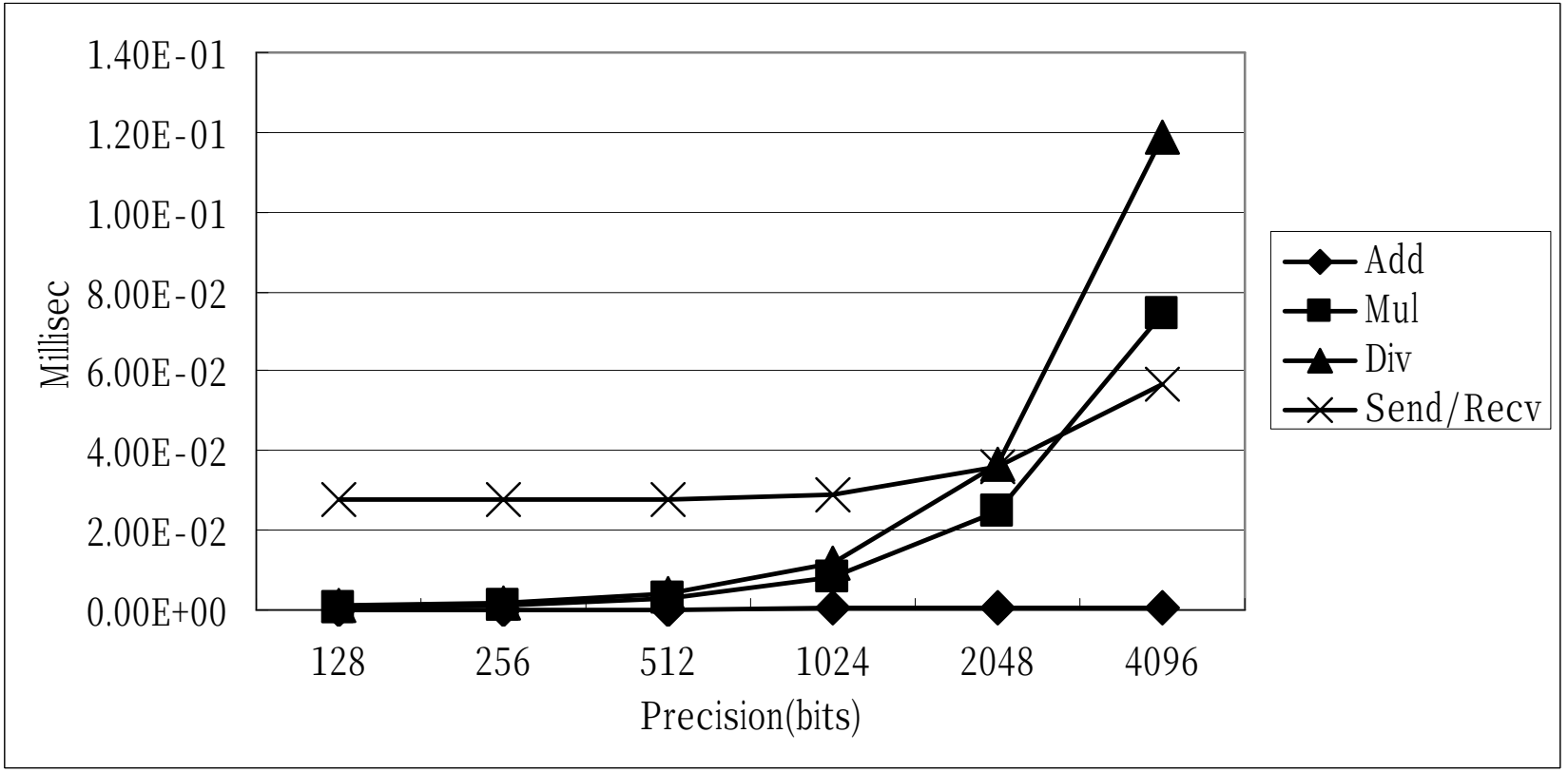
1. 実験環境
2. 1対1通信, 集団通信
3. 行列積(_mpi_mul_mpfmatrix関数)
4. CG法(_mpi_MPFCG関数)
5. DKA法(_mpi_mpf_dka関数)

実験環境

- OSは全てVine Linux 2.6r1
- NFS/NISでディレクトリとユーザアカウントの共有を行う
- PCはIntel Pentium III/Celeronの1GHzマシン



1対1通信

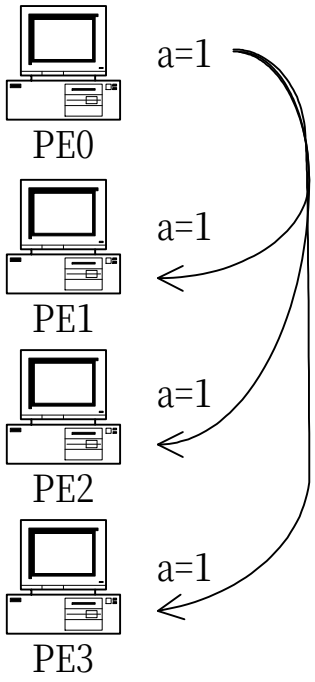


Bit数が長くなるほど, 粒度が上がり, 並列分散処理向き。

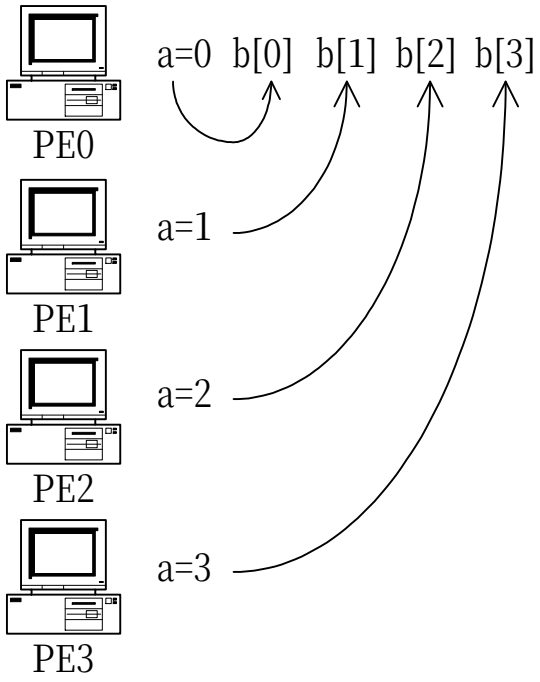
通信時間はBit数に比例(但し, 集団通信は別)。

集団通信(1/3)

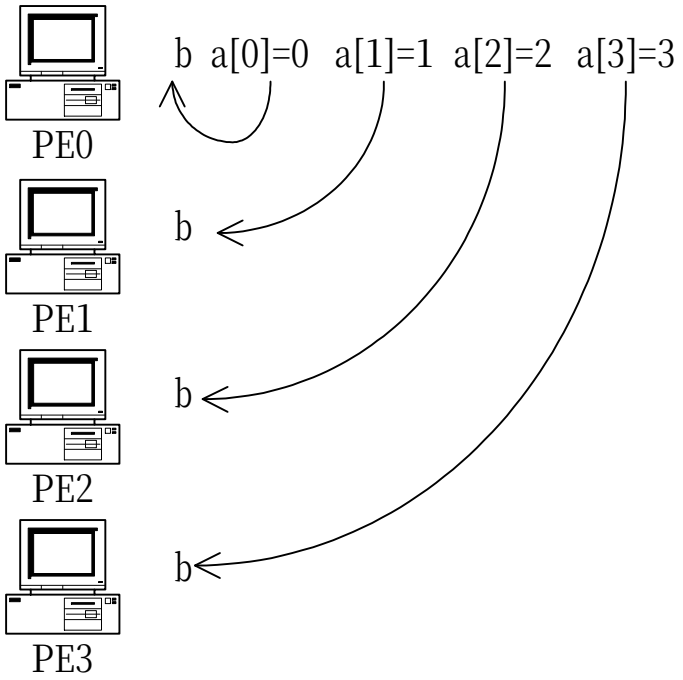
- Bcast8



Gather8

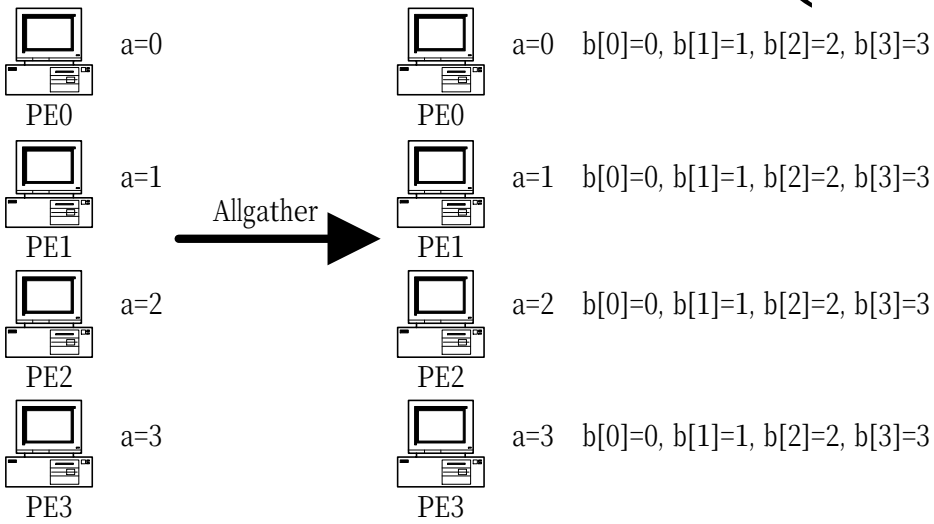


Scatter8

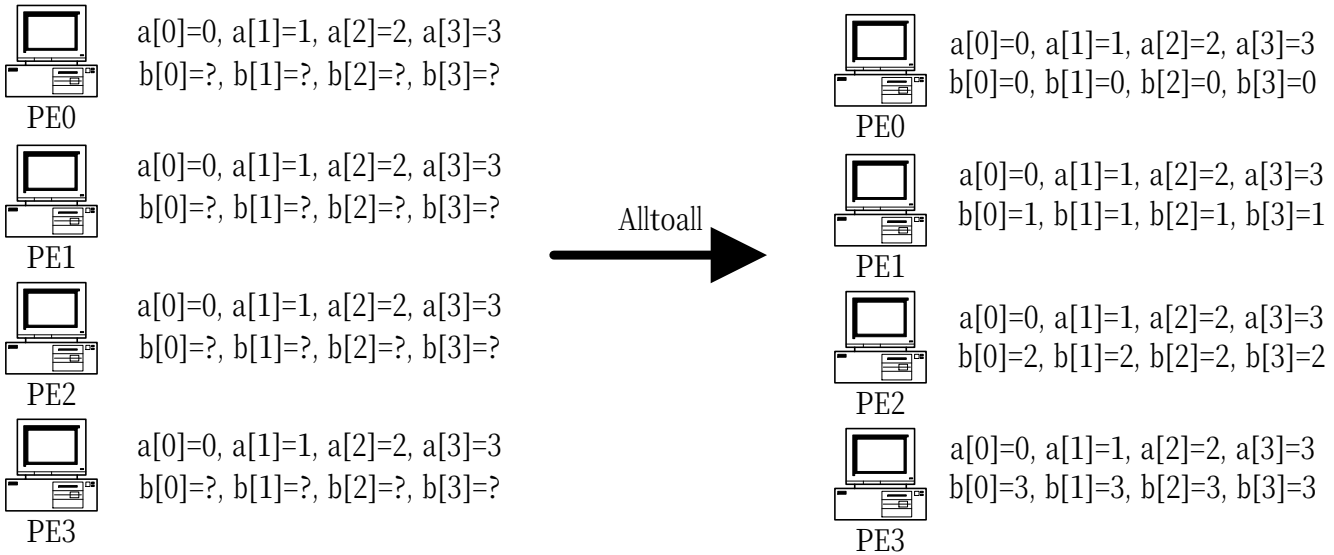


集団通信 (2/3)

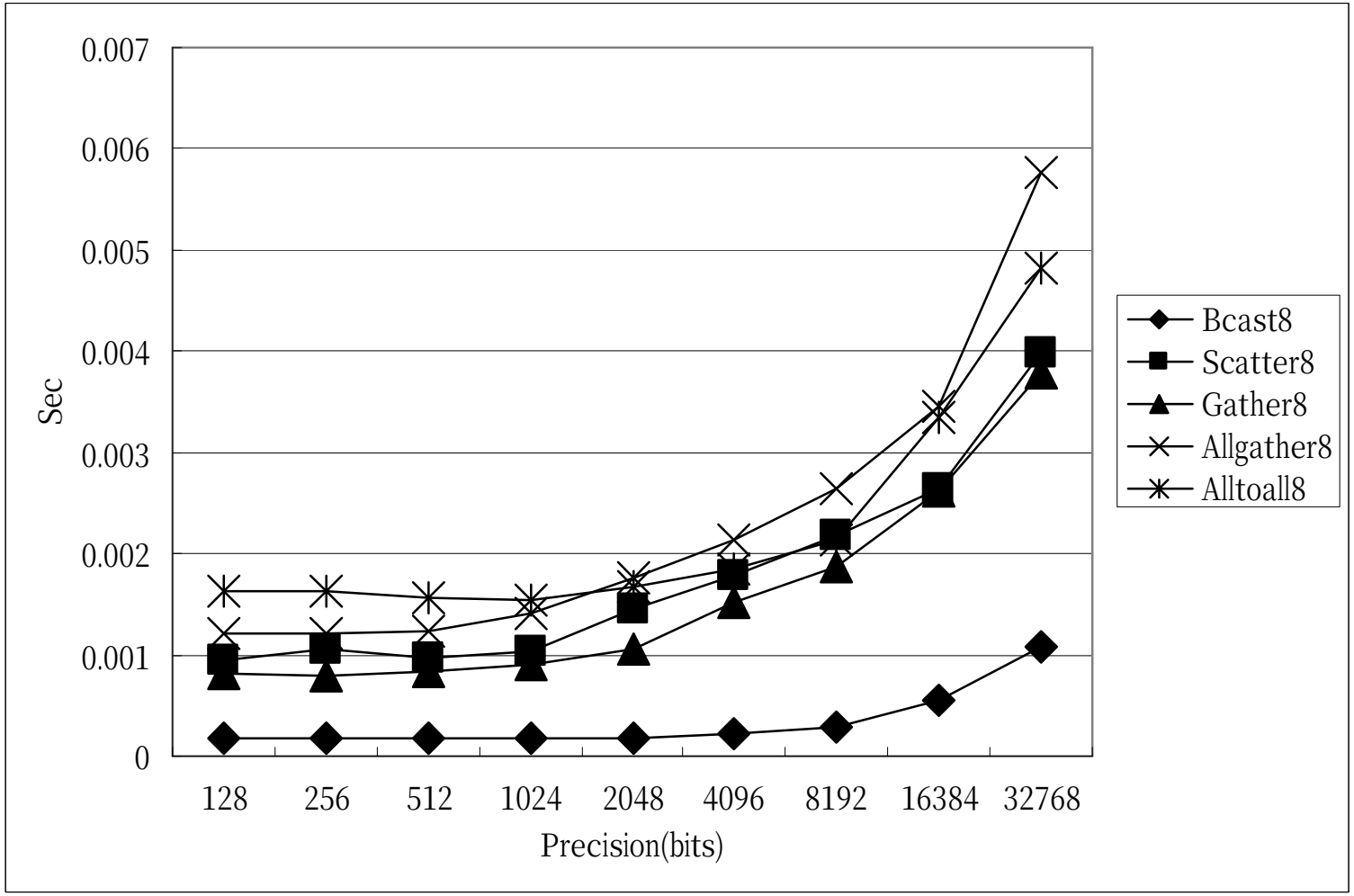
- Allgather8



- Alltoall8



集団通信(3/3)



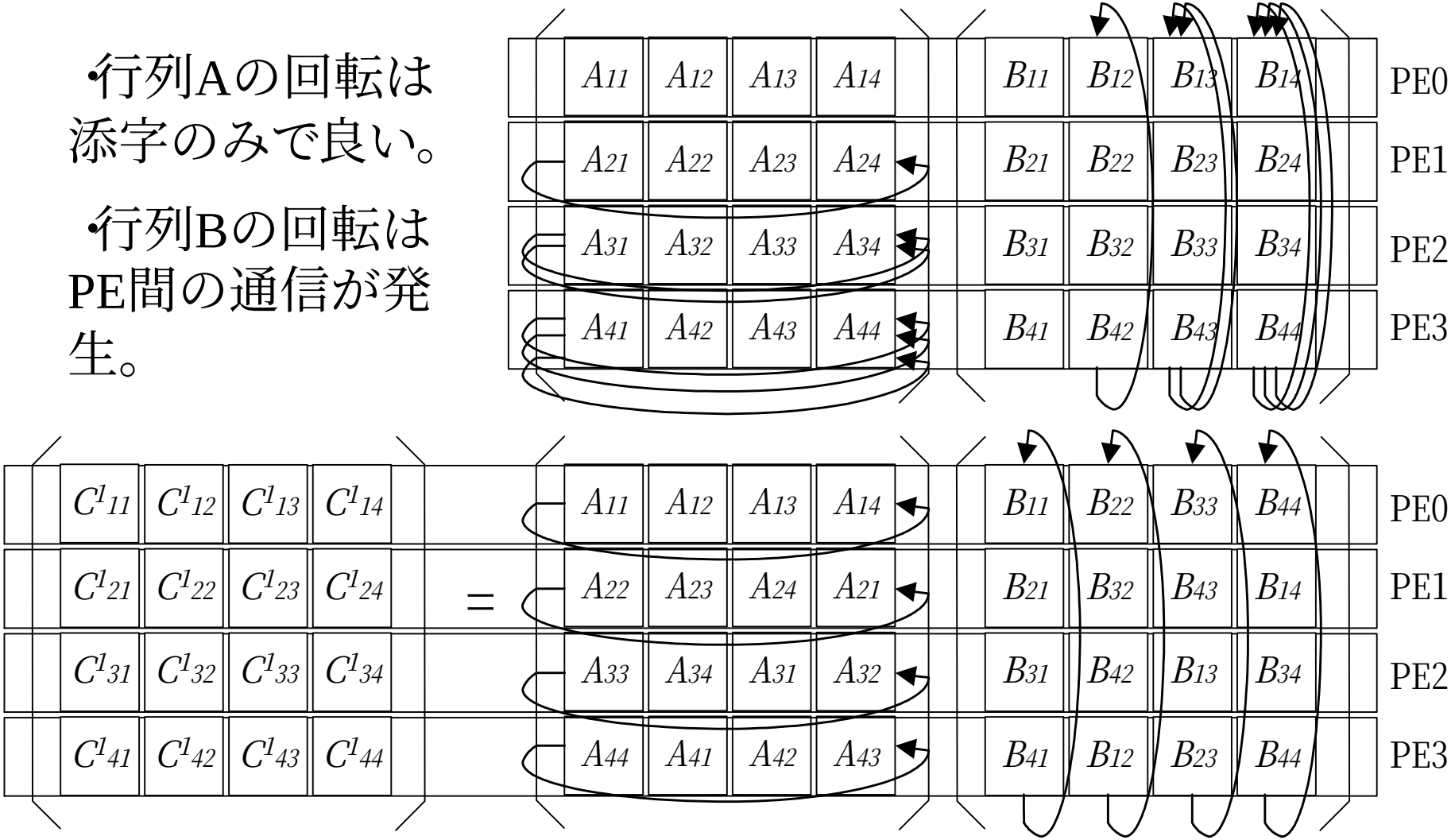
Bcastを除けば，同じ程度で通信時間が増加する。

この程度の長さのデータでは大差が出ない？

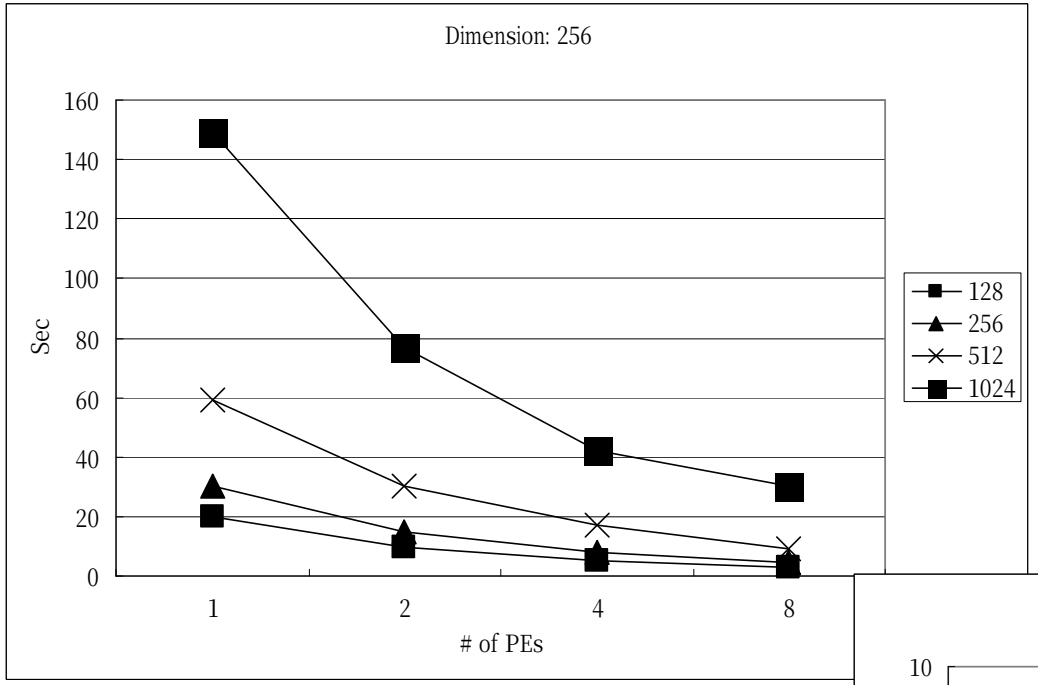
行列積(1/2)

$$C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + A_{i3}B_{3j} + A_{i4}B_{4j}$$

- 行列Aの回転は添字のみで良い。
- 行列Bの回転はPE間の通信が発生。



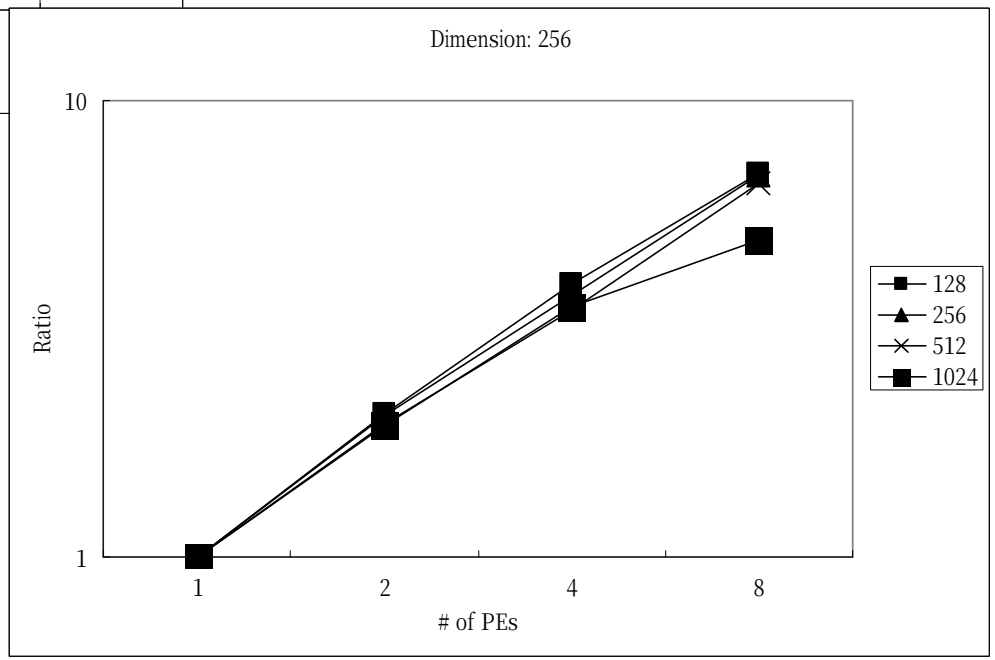
行列積(2/2)



- 256次実正方行列の行列積を計算

精度が大きくなると通信のオーバーヘッドが大きい？

MPI_Send/Recvによる実装がよろしくない？



連立一次方程式

CG法(1/3)

$A\mathbf{x} = \mathbf{b}$

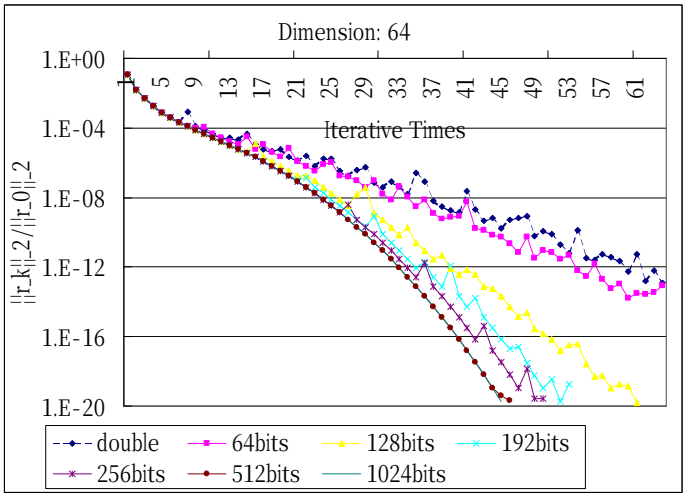
内積, スカラー倍, ノルム, 行列 •ベクトル積を並列分散化

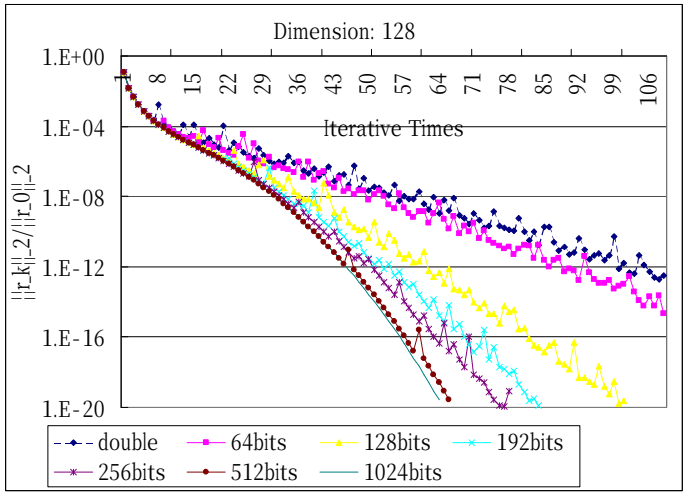
$$A = \begin{bmatrix} n & n-1 & \cdots & 1 \\ n-1 & n-1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ n-1 \end{bmatrix}$$

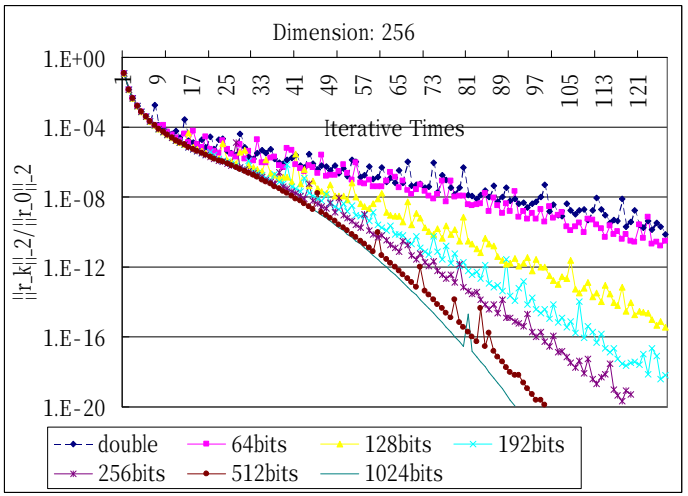
1. 初期値 $\mathbf{x}_0$ を決める。
2. $\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$, $\mathbf{p}_0 := \mathbf{r}_0$ とする。
3. $k = 0, 1, 2, \cdots$ に対して以下を計算する。
   - (a) $\alpha_k := \dfrac{(\mathbf{r}_k, \mathbf{p}_k)}{(\mathbf{p}_k, A\mathbf{p}_k)}$
   - (b) $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
   - (c) $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k A\mathbf{p}_k$ (又は $:= \mathbf{b} - A\mathbf{x}_{k+1}$)
   - (d) $\beta_k := \dfrac{\|\mathbf{r}_{k+1}\|_2^2}{\|\mathbf{r}_k\|_2^2}$
   - (e) 収束判定
   - (f) $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

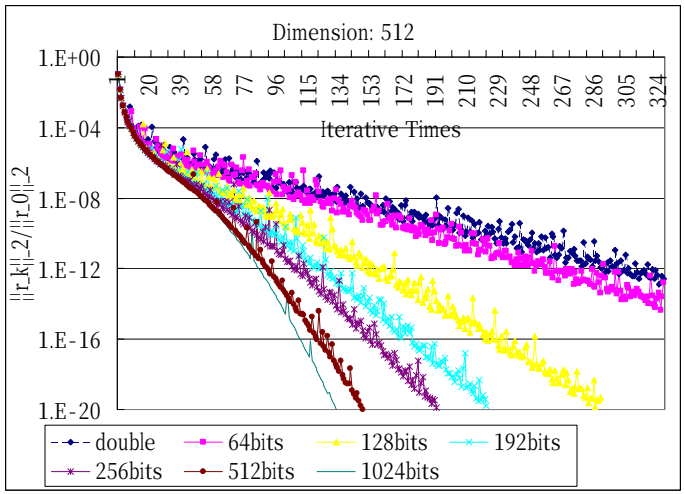
CG法(2/3)

- 64次元～512次元の収束履歴









多倍長で計算すると，反復回数が少なくなる。

↓

最短計算時間は？

CG法(3/3)

次元数	64	128	256	512
倍精度 (PE 数)	0.02(1)	0.083(2)	0.31(4)	1.46(4)
多倍長 (bit 数, PE 数)	0.073 (128,4)	0.306 (128,4)	1.06 (128,8)	4.89 (128,8)
比率	**3.67**	**3.71**	**3.44**	**3.35**

- IEEE754倍精度の3倍強が限界。
- もっと次元数を上げると？
- PE数がもっと多ければ？

DKA法(1/3)

ここで，DKA 法は $P_n(z)=0$ という n 次代数方程式に対しては

$$z_l^{(k+1)} := z_l^{(k)} - \frac{P_n(z_l^{(k)})}{\prod_{j=1,j\neq l}^{n}(z_l^{(k)} - z_j^{(k)})} \quad (l = 1, 2, ..., n)$$

という反復式に基づくものを用いる。

↑複素数演算！

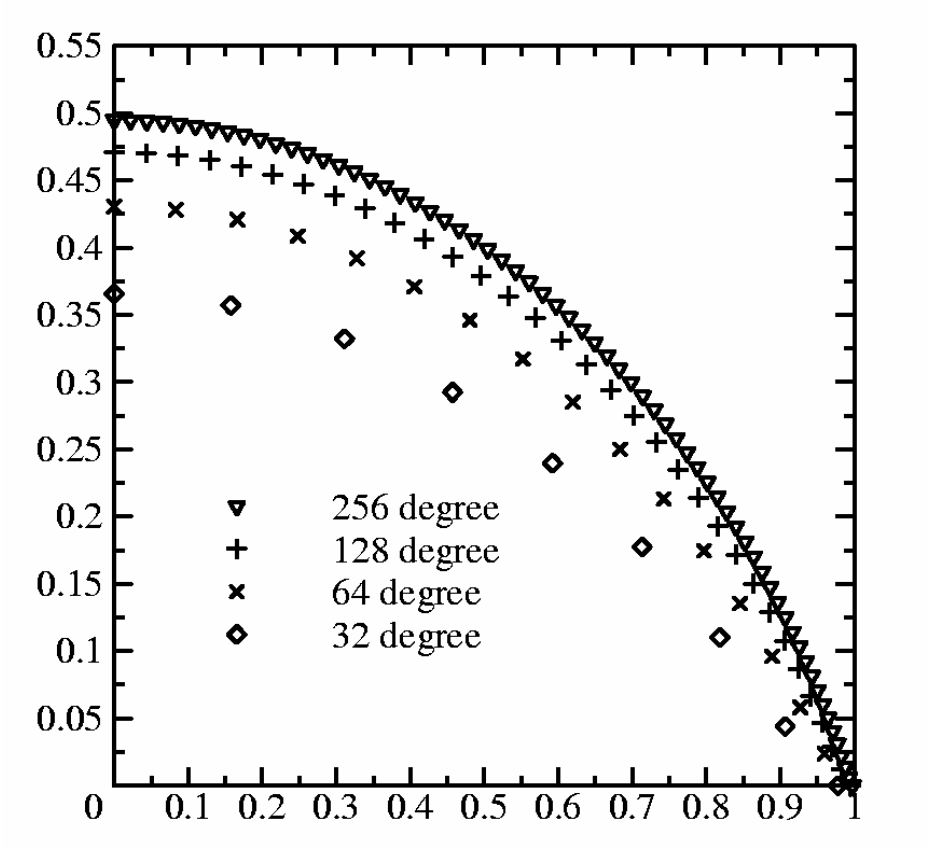
今回は次の代数方程式を解いてみる。

$$a_0 = 1,\ a_{2k} = -\frac{n}{2k}\sum_{j=1}^{k}\frac{1}{2j+1}a_{2(k-j)}$$

$$P_n(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

但し, 奇数次の係数は0。

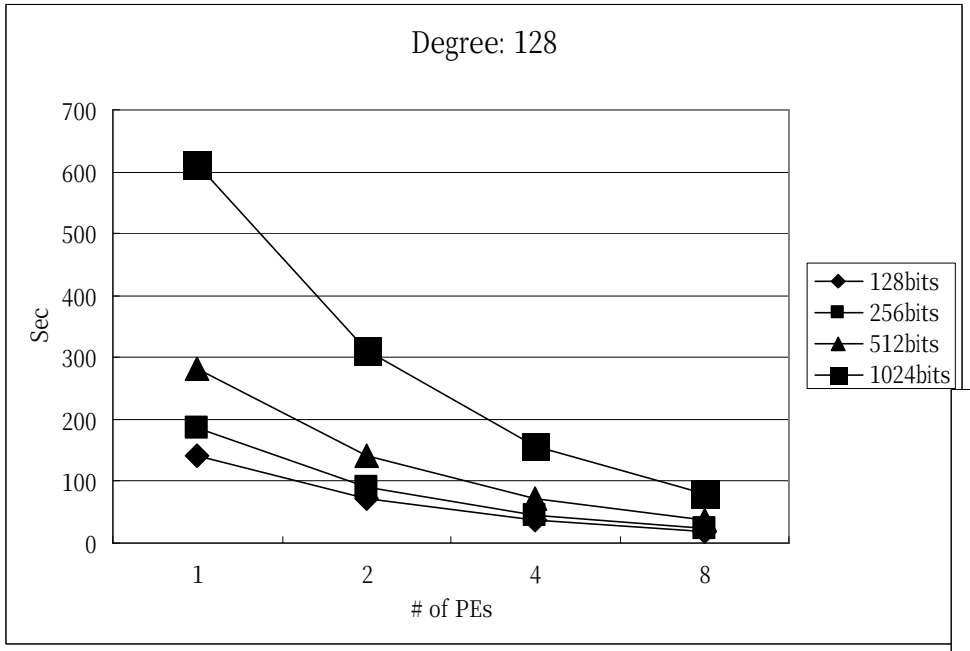
DKA法(2/3)

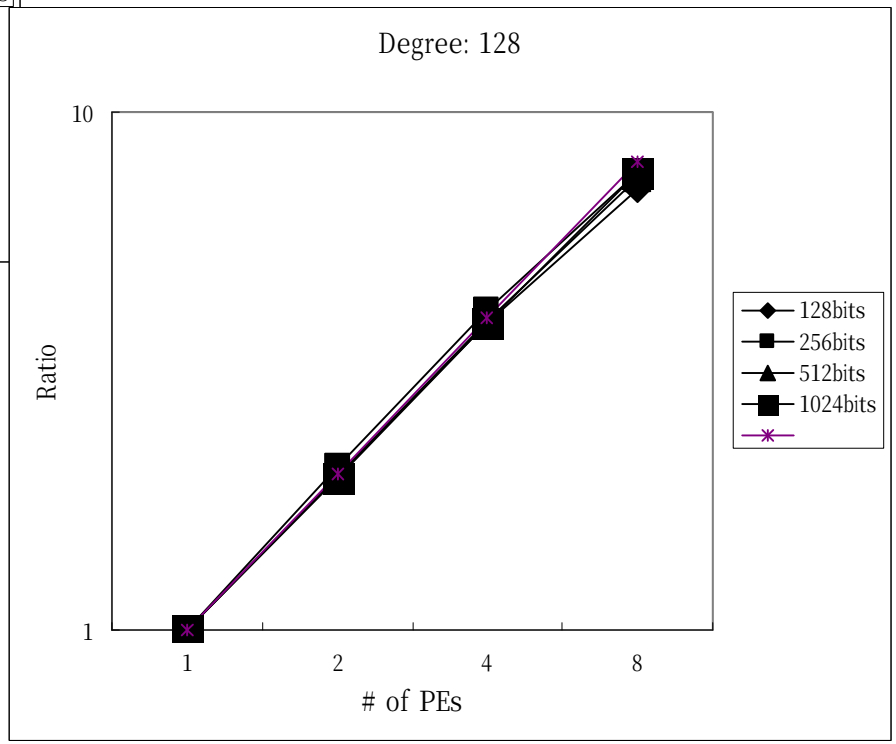


- Gauss平面の第１象限のみ，会を表示している。
- 大部分が複素数解となる。

DKA法(3/3)

- 128次元の場合





並列化効率はほぼMAX値。

今後の(アテにならない)予定

- 多倍長計算を生かす数値計算アルゴリズムの研究
- ライバルとの比較検討・・・計算効率 ・使い勝手
  - [1CPU] FMLIB, ARPREC(MPFUN) v.s. GMP+MPFR
  - [MPI] PETSc v.s. MPIBNCpack with ScaLAPACK
- MPIBNCpackの機能強化
  - 精度のスコープの明確化(必須！)
  - SMP/MTへの対応・・・local scalability
  - 超多倍長への対応・・・global scalability
  - 可視化ツールへの対応(Patch?)

蛇足(1/2)

- 大学院「分散処理」(前半)を担当
  - MPIBNCpackを使った並列数値アルゴリズムを教える

↓

- MPIは主要な機能のみ解説
- 数値計算に集中できる！！（分散化を自動化したおかげ）

蛇足(2/2)

A Tutorial of BNCpack and MPIBNCpack

並列分散多倍長数値計算に向けて

Last Update: October 28, 2003

---

1. 表紙・初めに・目次
2. PC Clusterとは？
3. UNIXにおけるプログラミング初歩
4. 初歩のBNCpackプログラミング
   - hellow.c, integer.c, float.c
   - gmp-mpz.c, gmp-mpf.c, mpfr.c
   - bnc1.c, bnc2.c, bnc3.c, bnc4.c
5. 初歩のMPIプログラミング
   - mpi1.c, mpi2.c, mpi3.c, mpi4.c
   - mpi-sr.c, mpi-sr-gmp.c
6. MPIの集団通信
   - mpi-bcast.c, mpi-bcast-gmp.c
   - mpi-gather.c, mpi-gather-gmp.c
   - mpi-scatter.c, mpi-scatter-gmp.c
   - mpi-reduce.c, mpi-reduce-gmp.c
   - mpi-allgather.c, mpi-allgather-gmp.c
   - mpi-allreduce.c, mpi-allreduce-gmp.c
   - mpi-alltoall.c, mpi-alltoall-gmp.c, mpi-alltoall2.c, mpi-alltoall-gmp2.c
7. 最初のMPIBNCpackプログラミング
   - vec1.c, mpi-vec1.c, mpi-vec1-gmp.c

http://na-inet.jp/tutorial/にて公開。

参考文献

- BNCpack/MPIBNCpack, http://na-inet.jp/na/bnc/
- Try! MPFR, http://www.jpsearch.net/try_mpfr.html
- A Tutorial of BNCpack and MPIBNCpack, http://na-inet.jp/tutorial/

- G.H.Golub/C.F. Van Loan, Matrix Computations 3rd ed. , Johns Hopkins Univ. Press, 1996.
- P.Paccheco/秋葉博 •訳, MPI並列プログラミング, 培風館, 2001.

- mpich, http://www-unix.mcs.anl.gov/mpi/mpich/
- GNU MP, http://swox.com/gmp/
- MPFR Project , http://www.mpfr.org/