超並列計算への入門

並列効率の2つの概念:'strong scaling', 'weak scaling'

例え話:X人分の答案を、教員1人で採点すると、T 時間で終わる。

(1) strong scaling

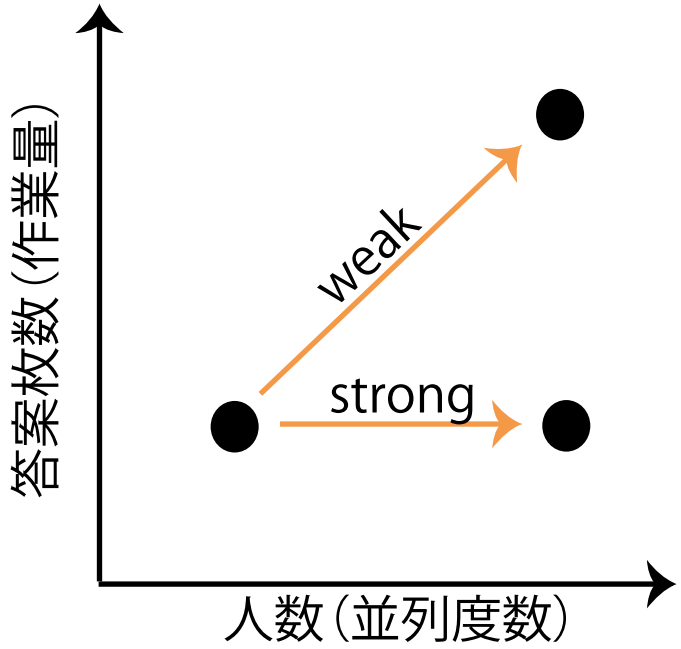
X人分の答案を、教員n名で採点したら、(T/n)時間で終わるか?

(作業量:固定、人数:n倍→作業時間は$(1/n)$倍?)

(2) weak scaling

nX人分の答案を、教員n名で採点したら、T 時間で終わるか?

(作業量:n倍、人数:n倍→ 作業時間は不変?)

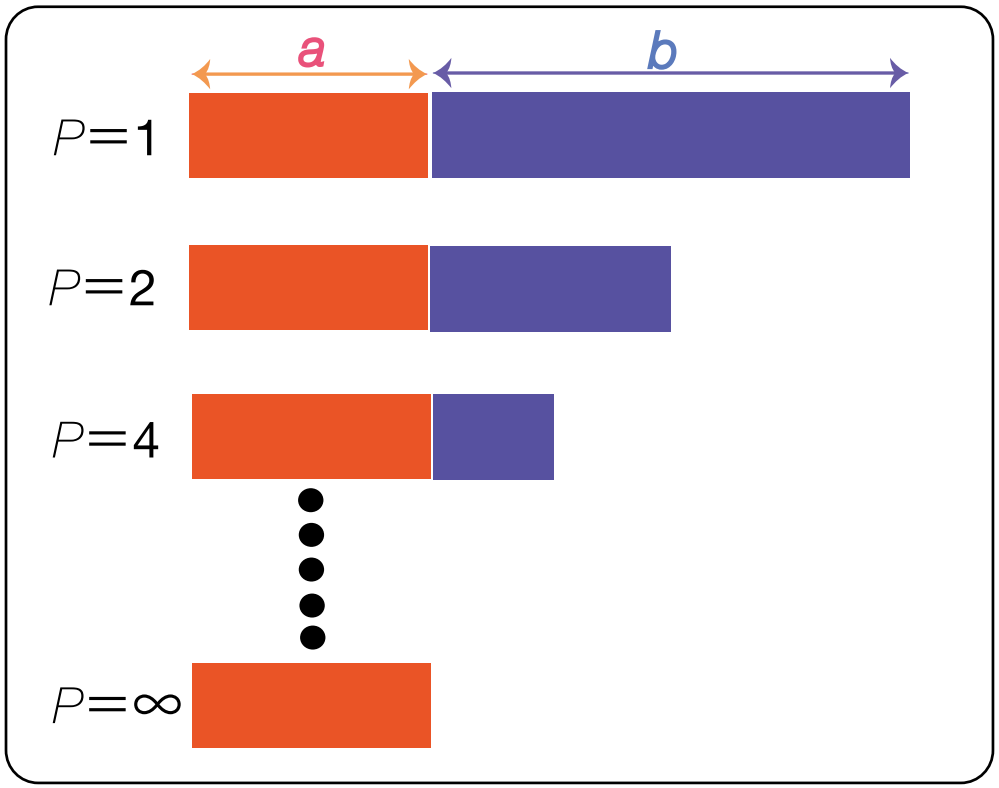


Amdahl's relation (アムダールの関係式)(*)

G. Amdahl, AFIPS Conference Proceedings 30, 483–485 (1967)

→非並列部分があると、並列度数*P*を増やしても加速に限界がある

a :非並列部分の経過時間 *b* :並列部分の経過時間



*P*並列したときの経過時間

$$T(P) = a + \frac{b}{P} \qquad (1)$$

*P*並列したときの時間短縮率

$$\frac{T(P)}{T(1)} = \frac{a + \frac{b}{P}}{a + b}$$

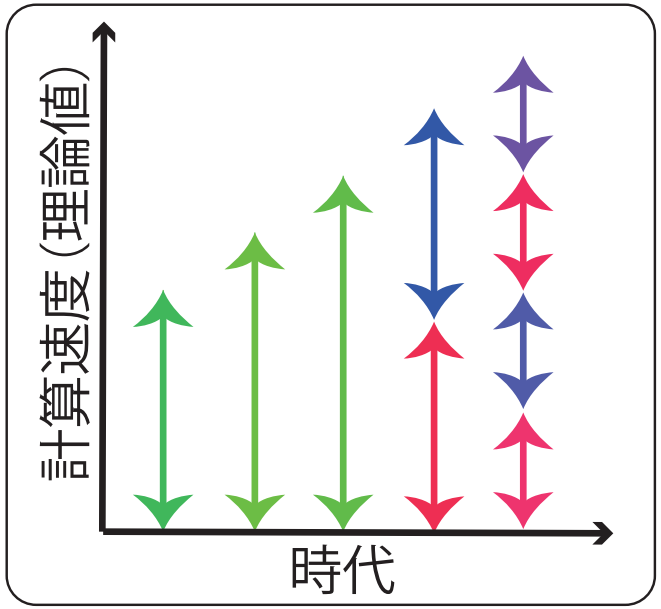
$$= (1 - x) + \frac{x}{P} \qquad (2)$$

ただし, $x \equiv \frac{b}{a + b}$ (3)

(*) Amdahl's law, Amdahl's argumentと呼ばれることもある

並列計算へのパラダイムチェンジ(2000年代後半~)

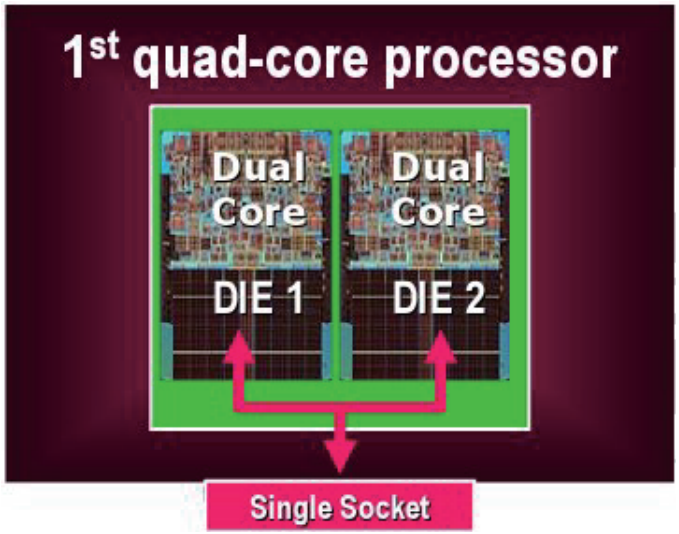
並列計算機の登場



注：単体性能は高くない

→計算手法にパラダイムチェンジを迫ることに

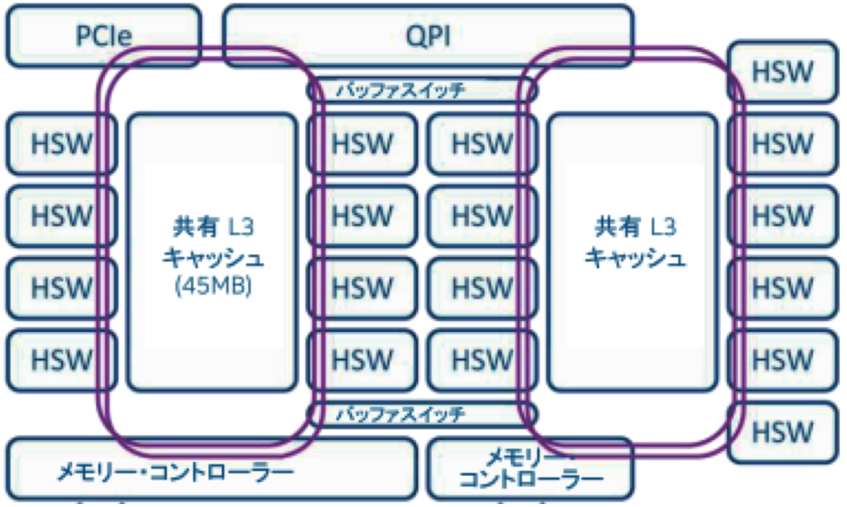
Herb Sutter
"The Free Lunch Is Over:
A Fundamental Turn
Toward Concurrency in Software",
Dr. Dobb's Journal, 30(3), March (2005)



インテル社のCPU

←Core 2 Quad(2006)

18-core Xeon (*)
(E5-2600-v3) (2014)→



(*)http://pc.watch.impress.co.jp/docs/column/kaigai/20140909_665735.html

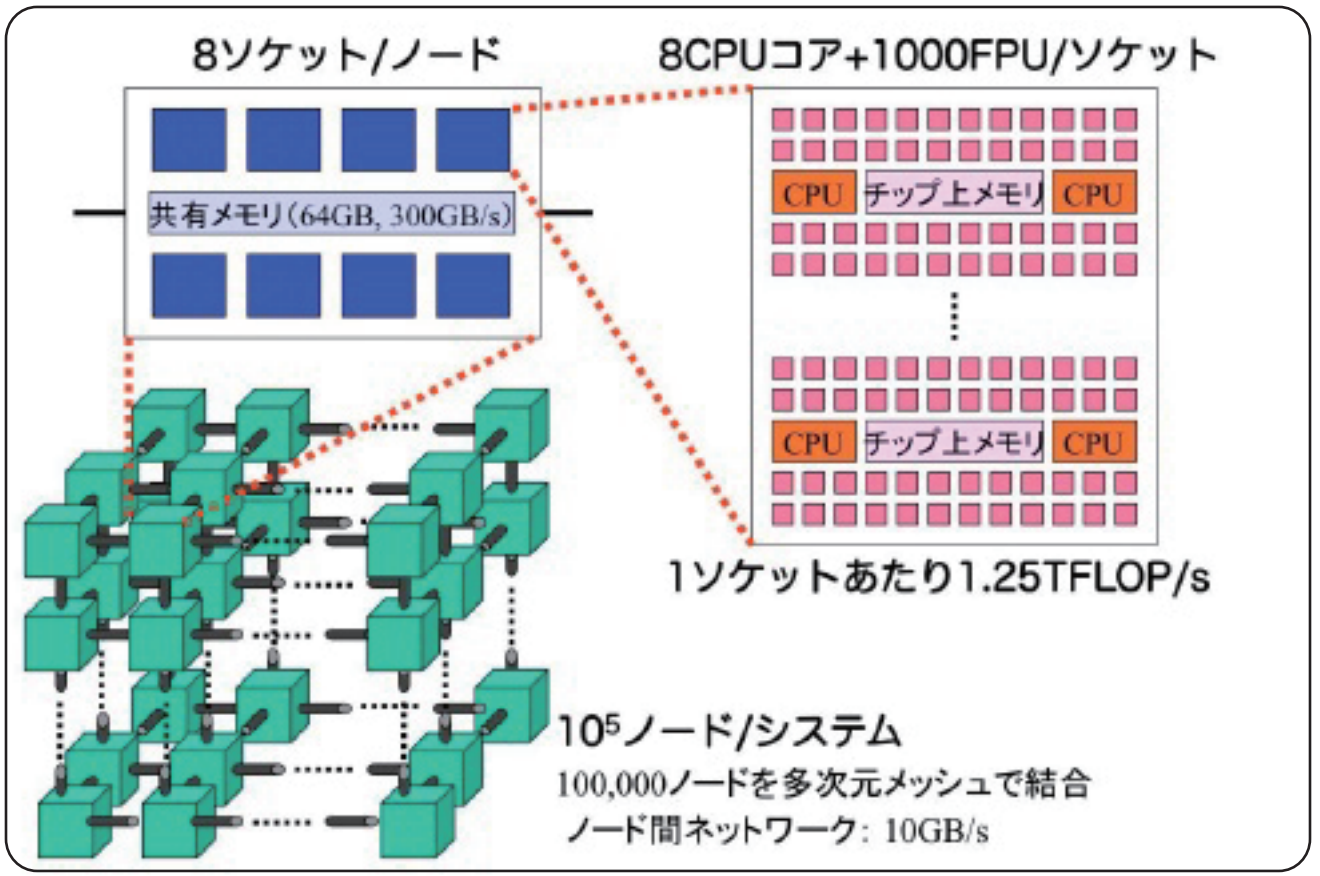
超並列計算への入門

例:「京」スーパーコンピュータ

- 1 node に、1 CPU = 8 CPU cores
- 計算部分全体:82, 944 node = 663,552 CPU cores
- 1 node に、16 GBのメモリ

例え話:
- 82,744部屋あるマンション
- 1部屋あたり8人居住

階層的並列構造



「のべ時間」での利用

単位:ノード時間(NH)

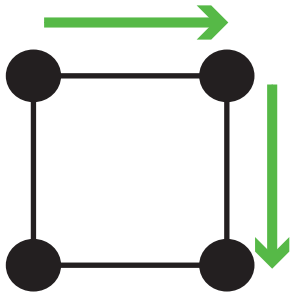
例:100ノードを使って、
2時間計算すると、
100 x 2 = 200 NH

多次元状のノード配置における、通信の加速可能性(1/2)

→多次元状にノードを配置することで、通信が加速する（可能性がある）

例：1次元状に4ノードを配置
→最大3hopでデータ到達

例：2次元状に4ノードを配置
→最大2hopでデータ到達



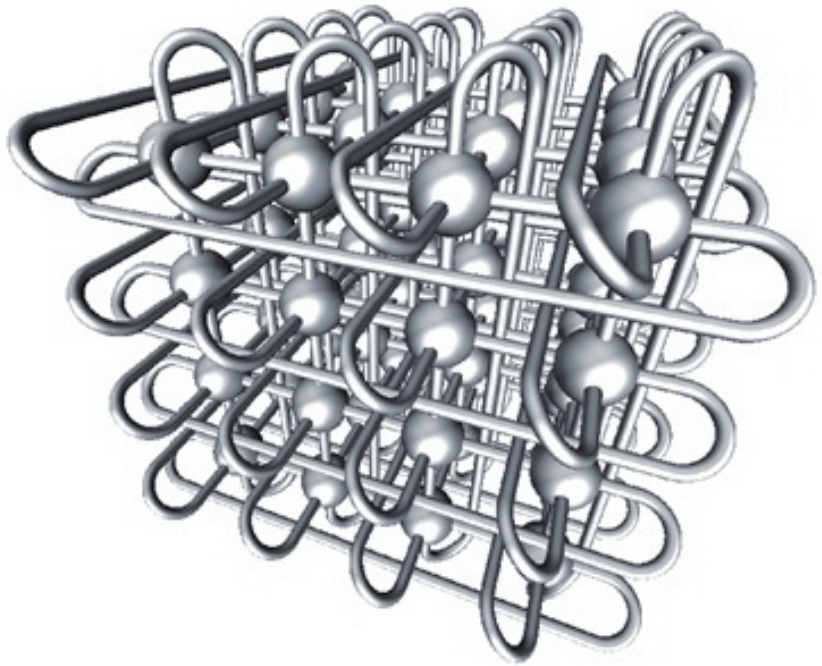
多次元状のノード配置における、通信の加速可能性(2/2)

「京」は3次元状にノードを配置している
→jobファイルで、3次元状ノード指定により、通信が加速する（の可能性がある）

例：512ノードのjob

(a) 1Dノード構造の指定
"node=512"
(b) 3Dノード構造の指定
"node=8x8x8"

「京」の「Tofuインターコネクト」*



*Tofu = Torus fusion
http://www.aics.riken.jp/jp/k/system.html

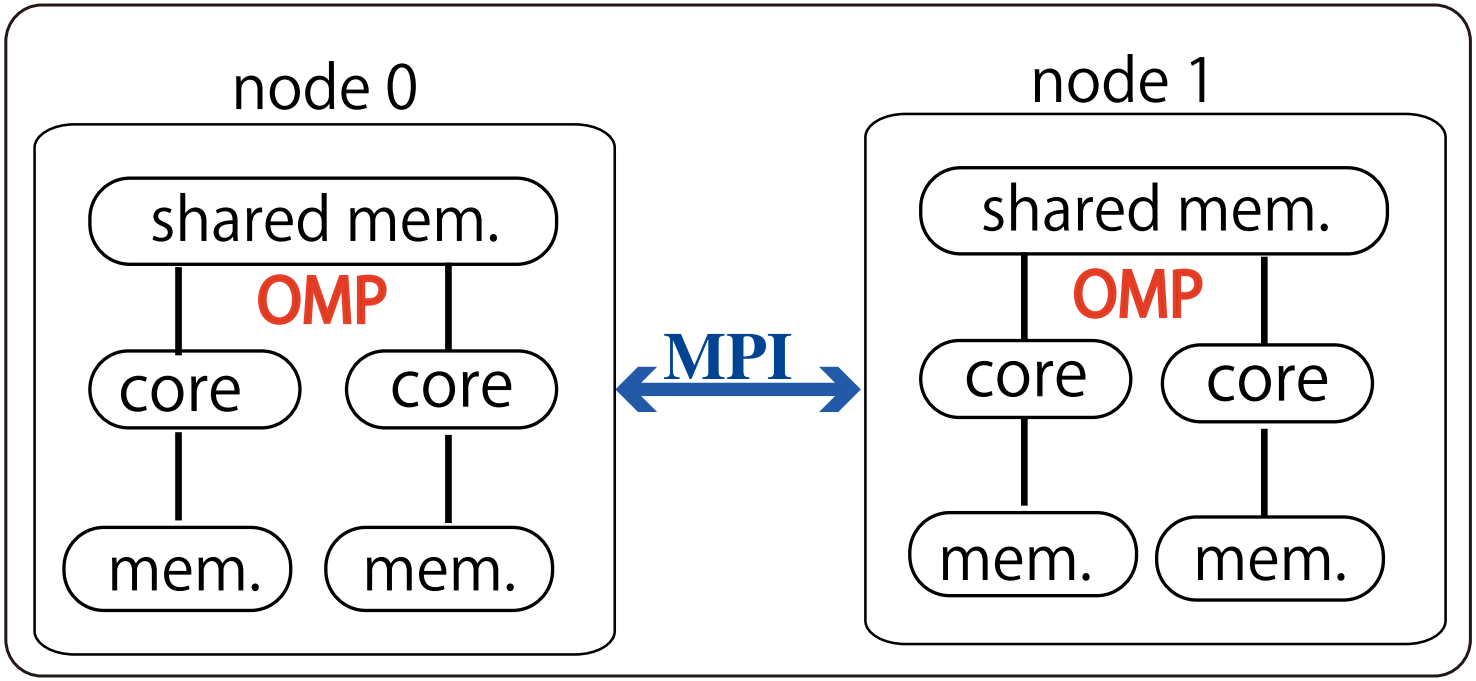
並列計算手法の基礎(MPI, OpenMP)

→2種類の手法を階層的に併用する

(A) Message Passing Interface (MPI)
- 分散メモリ型。
- 通信を明示的に指定する。
- 以下、MPIでの並列単位を「node」と呼ぶ

(B) Open Multi-Processing (OpenMP, OMP)
- 共有メモリ型。
- 各変数を「shared(共有)」か「private(非共有)」に指定。(通信を明示的には制御しない)。
- 以下、OMPでの並列単位を「thread」と呼ぶ
- 通常、適用できる範囲が限られる。

階層的並列計算の概念図

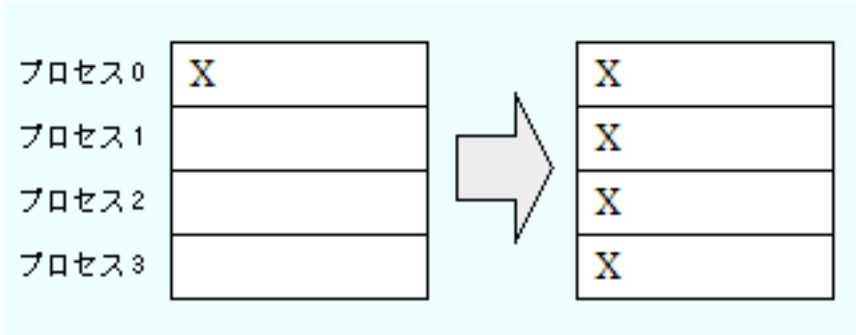


(MPI並列度数)=2,
(OMP並列度数)=2

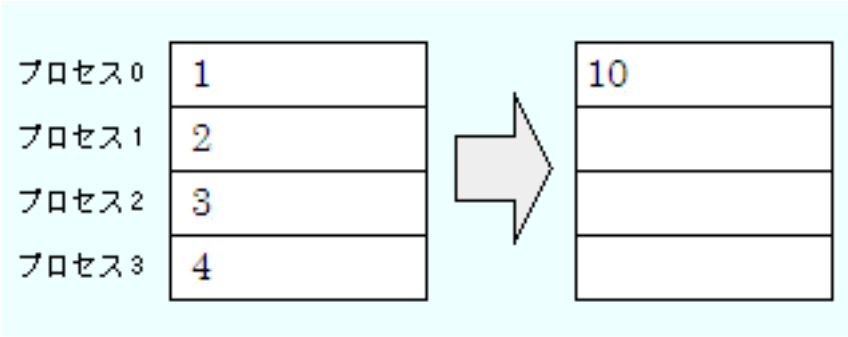
注:左図は、ハードウェア構成とは、必ずしも対応しない。

MPIルーチンの例

MPI_Bcast - ブロードキャスト

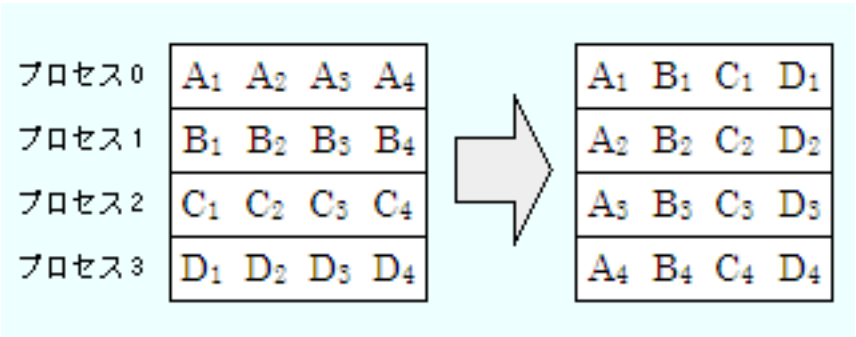


MPI_Reduce - レデュース

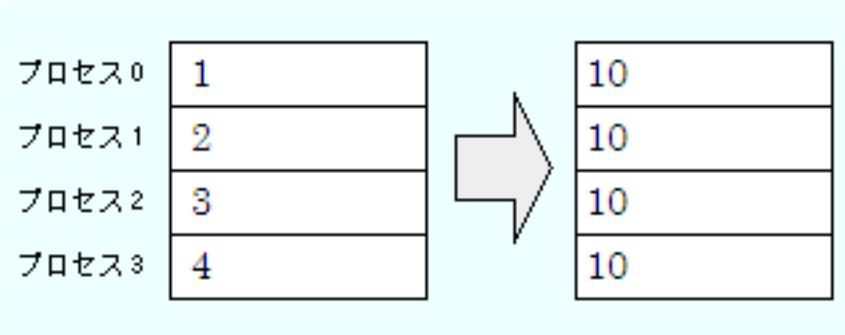


(この例は総和計算：1+2+3+4=10)

MPI_Alltoall - 全交換



MPI_Allreduce - オールレデュース



(この例は総和計算：1+2+3+4=10)

Ref. (図) http://www.cv.titech.ac.jp/~hiro-lab/study/mpi_reference/chapter3.html

情報処理学会論文誌　コンピューティングシステム　Vol.5　No.5　152–162 (Oct. 2012)

MPI_Allreduceの「京」上での実装と評価

松本 幸[1,a)]　安達 知也[1]　住元 真司[1]　南里 豪志[2]　曽我 武史[2]　宇野 篤也[3]
黒川 原佳[3]　庄司 文由[3]　横川 三津夫[3]

受付日 2012年3月28日, 採録日 2012年6月9日

概要：本論文では，82,944 台の計算ノードを Tofu インターコネクトと呼ばれる 6 次元の直接網で結合した「京」における MPI 集団通信の高速化について述べる．従来の MPI ライブラリには，トポロジを考慮したアルゴリズムが存在しないため，「京」のような直接網において性能を出すことができない．そのため，Trinaryx3 と呼ばれる Allreduce を設計し，「京」向けの MPI ライブラリに実装した．Trinaryx3 アルゴリズムは，トーラス向けに最適化されており，「京」の特長の 1 つである複数 RDMA エンジンを活用することができる．実装を評価した結果，既存のトポロジを考慮していないアルゴリズムと比較して，5 倍のバンド幅の向上を確認した．

キーワード：京コンピュータ，MPI 集団通信，MPI_Allreduce，トーラスネットワーク

行列データを分散して格納するには？

代表例：Block-Cyclic Mapping(*)

The block-cyclic distribution scheme is a mapping of a set of blocks onto the processes.

→ (block size) NB : 入力パラメーター (例：NB=128)

(*) ScaLAPACK(標準的な分散メモリ型線形計算ライブラリ)などで利用されている

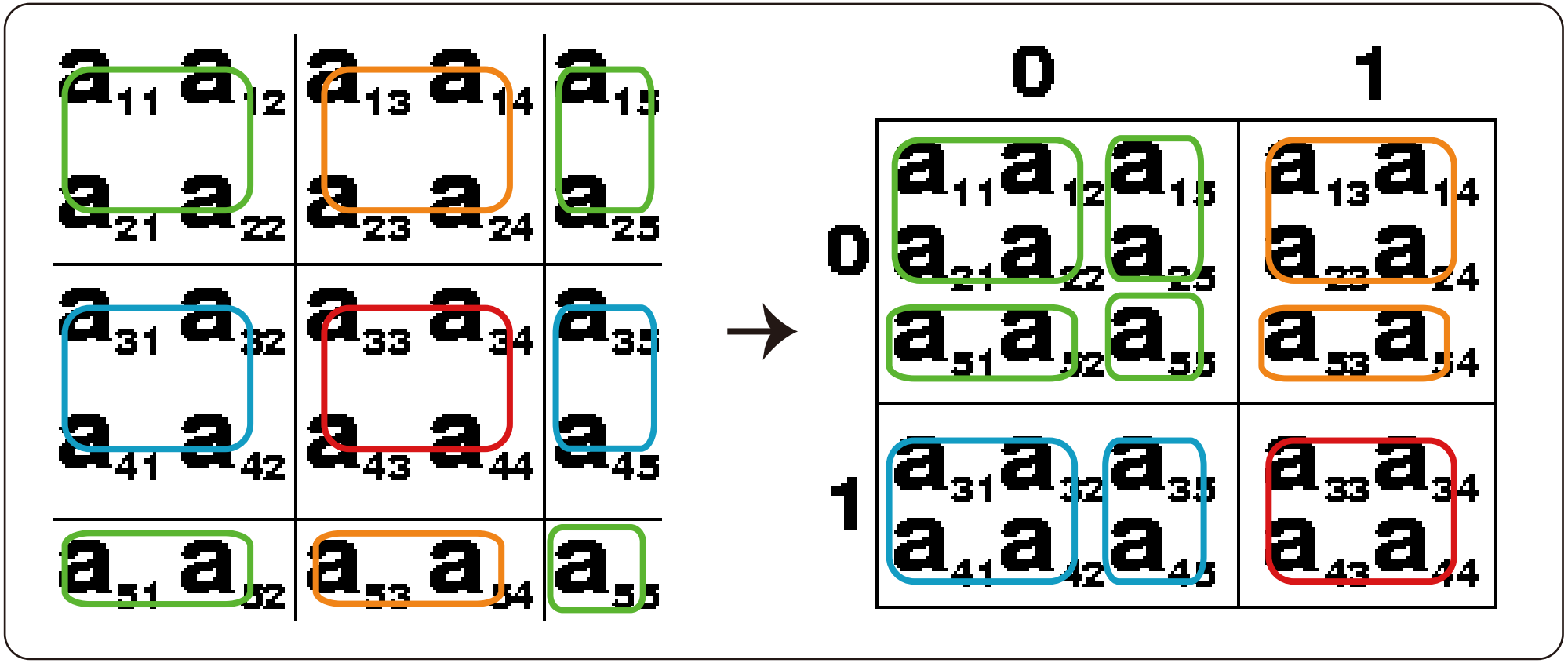


Fig. (5x5) matrix decomposed into (2x2) blocks
mapped onto a (2x2) process grid

http://netlib.org/scalapack/slug/node76.html

並列(大規模)計算むけコード開発で注意することの例

(1) 並列計算では、和の順序が保証されない→丸め誤差に注意

例: $((a+b)+c)+d \underset{?}{=} (a+b)+(c+d)$

(2) 標準(4 byte)整数規格の破綻可能性

△ 4 byte integerの最大数: 2^{31} - 1 = 2,147,483,647 (2.1×10^9, 約21億)

○ 8 byte integerの最大数: 2^{63} - 1 = 9,223,372,036,854,775,807 (約 9×10^{18})

例: 1億 x 100 ÷ 100 = 1億

100億

(3) 倍精度実数規格の破綻可能性

→4倍精度実数規格の利用

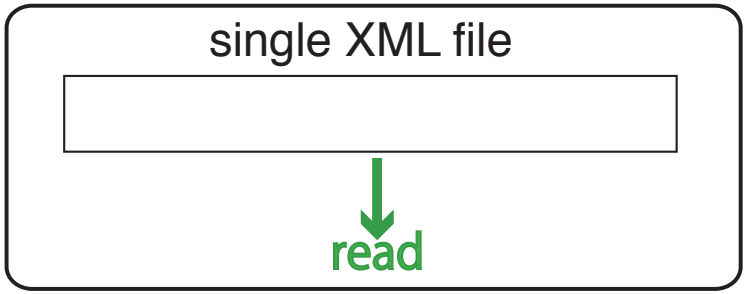
並列(大規模)計算むけコード開発で注意することの例

(4) ファイルI/O並列化の検討
→例：京での「rank directory機能」

Parallel file IO with split XML file

ex. Acceralation in the initial procedure with 10M atoms

(a) with non-parallel file reading
→ T =1426.6 sec



(b) with parallel file reading
→ T = 69.6 sec