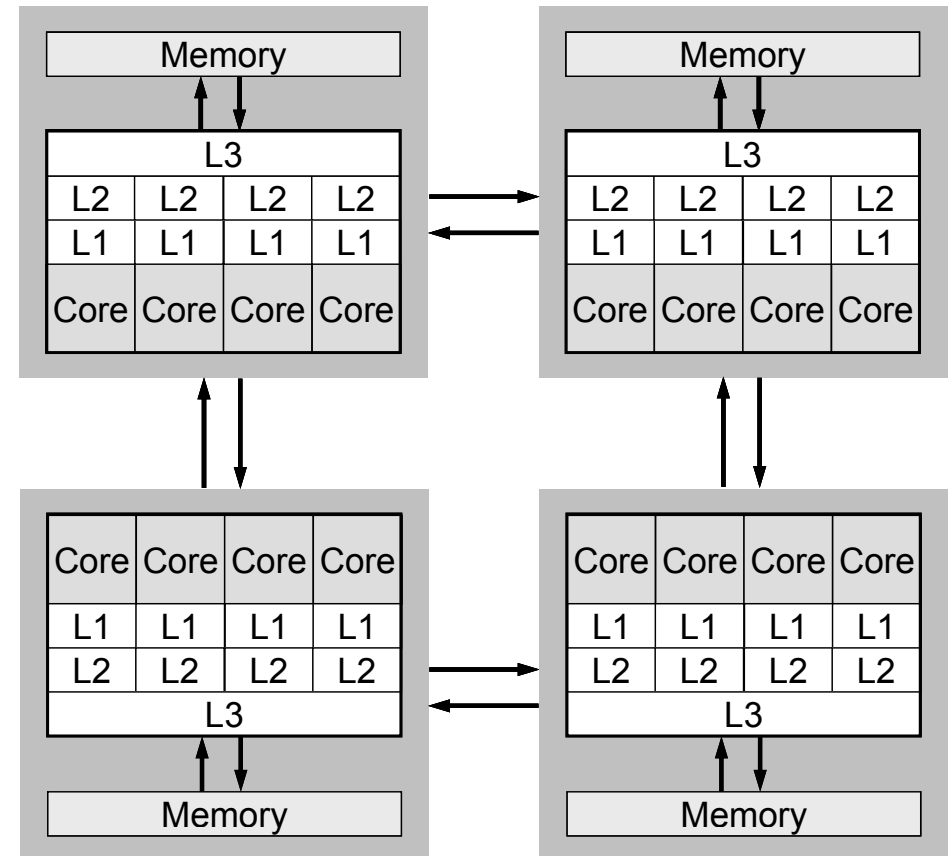
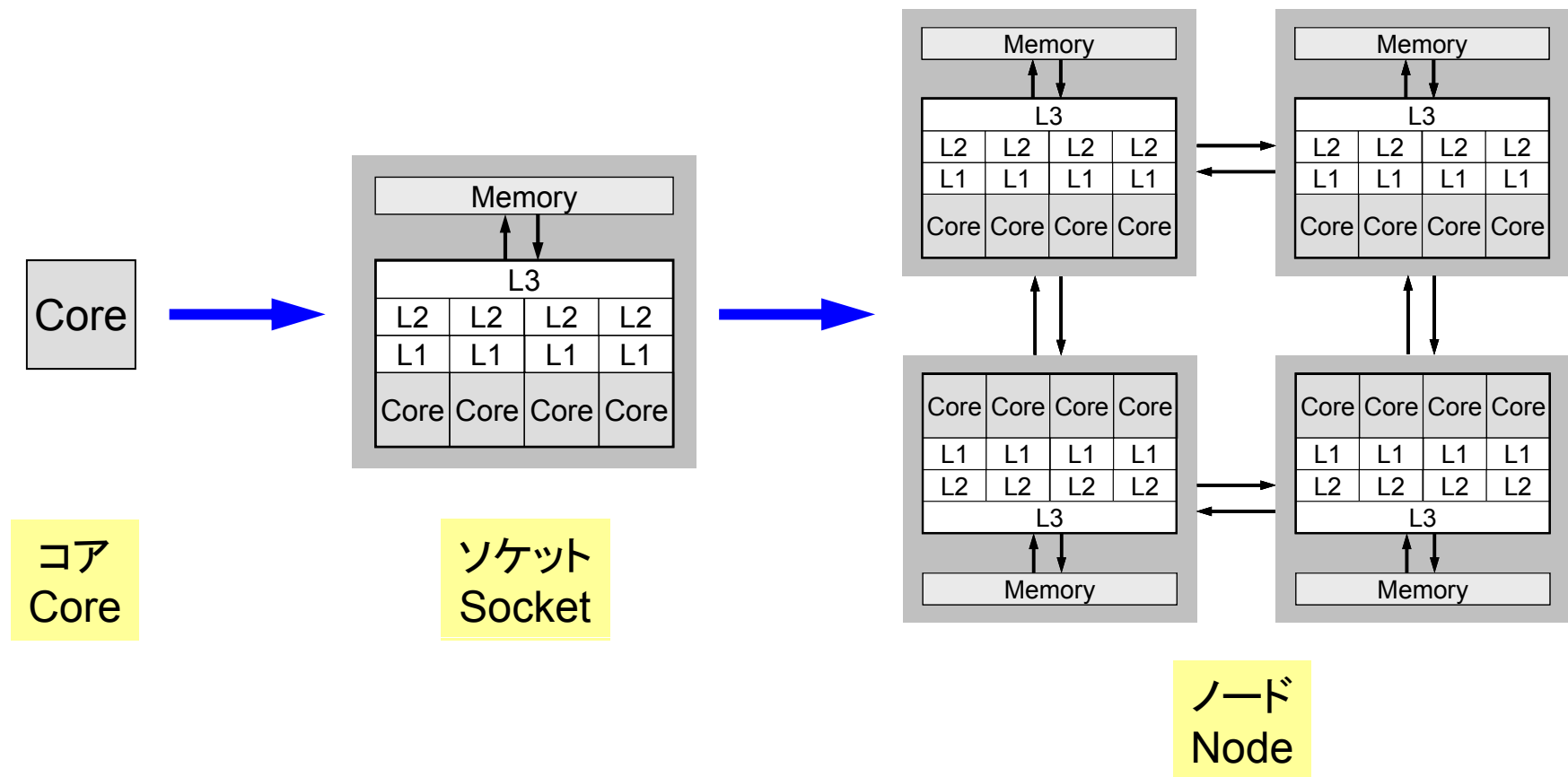


Quad Core Opteron: ccNUMA Arch.

- AMD Quad Core Opteron 2.3GHz
 - Quad Coreのソケット × 4 ⇒ 1 ノード(16コア)
- 各ソケットがローカルにメモリを持っている
 - NUMA: Non-Uniform Memory Access
 - ローカルのメモリをアクセスして計算するようなプログラミング, データ配置, 実行時制御 (numactl) が必要
 - cc: cache-coherent
 - キャッシュ同期: Thread並列



ccNUMA Architecture



numactl

```
>$ numactl --show
```

```
policy: default
```

```
preferred node: current
```

```
physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

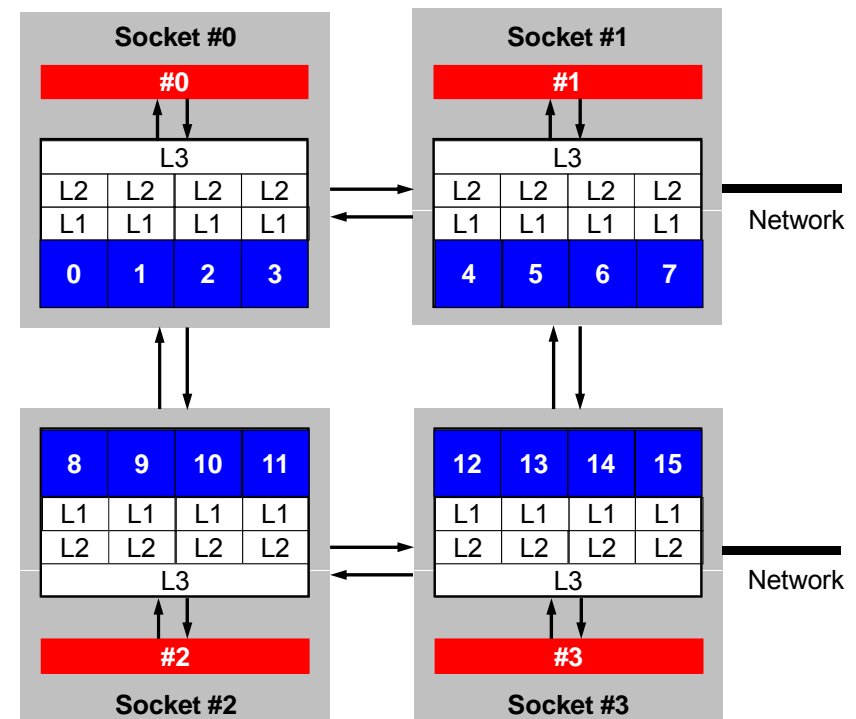
```
cpubind: 0 1 2 3
```

```
nodebind: 0 1 2 3
```

```
membind: 0 1 2 3
```

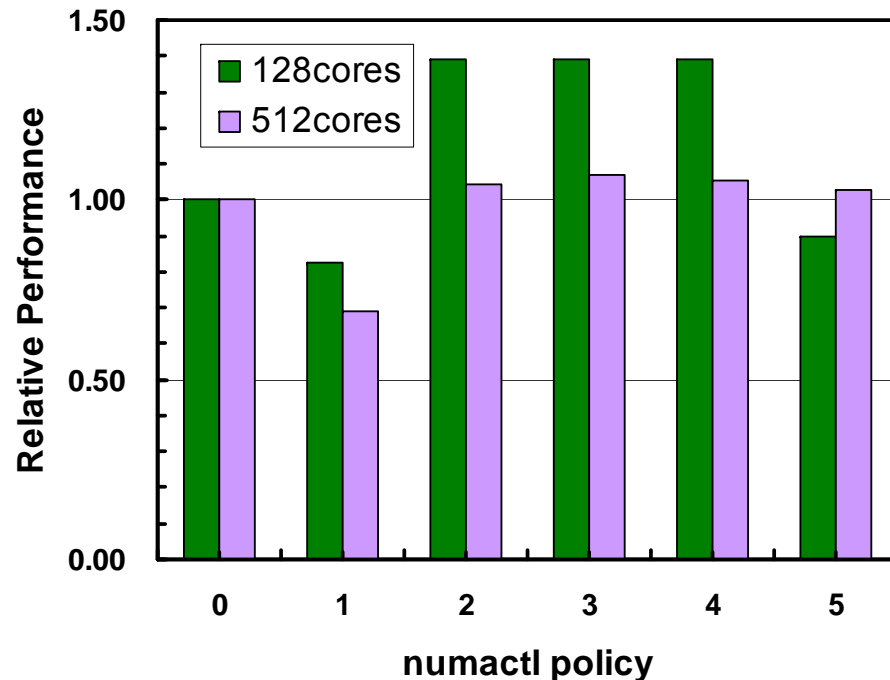
コア
ソケット
ソケット
メモリ

- NUMA(Non Uniform Memory Access) 向けのメモリ割付のためのコマンドライン: Linuxでサポート
- 使うコアとメモリの関係指定: ローカルなメモリ上のデータを使うのが得策



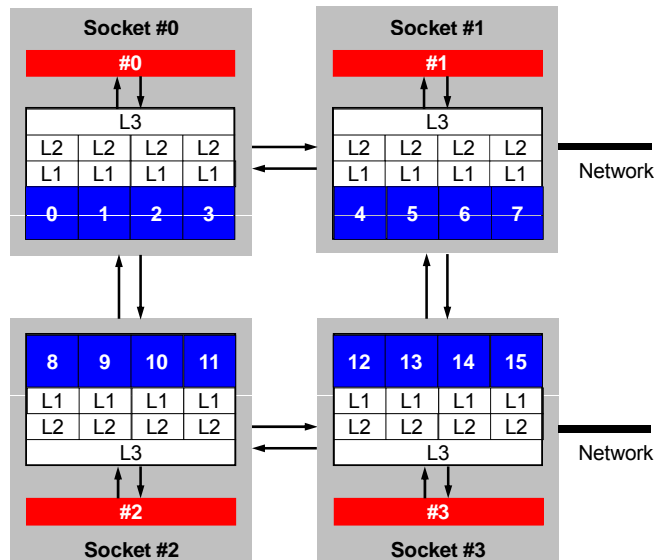
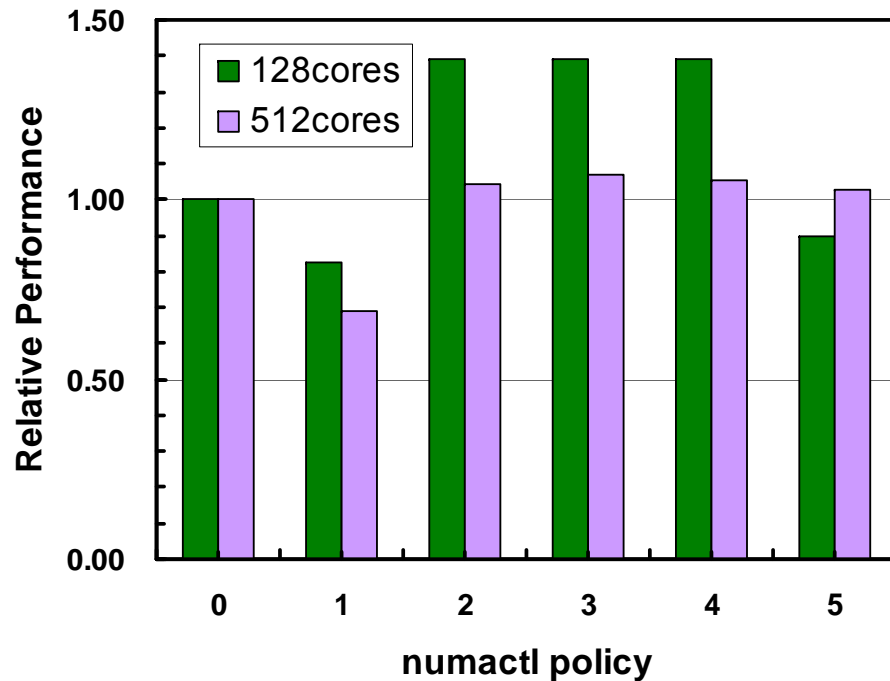
numactlの影響

- T2K, 有限要素法アプリケーション, Strong Scaling, Flat MPI
- 128コア: 1コアあたりの問題サイズは512コアの4倍
 - メモリへの負担大
- Policy=0: 何も指定しない場合
- 相対性能: 大きいほど良い—128ノードで影響大



```
#@$-r HID-org
#@$-q h08nk132
#@$-N 24
#@$-J T16
#@$-e err
#@$-o x384-40-1-a.lst
#@$-lM 27GB
#@$-lE 03:00:00
#@$-s /bin/sh
#@$
cd $PBS_O_WORKDIR
mpirun ./numarun.sh ./sol
exit
```

numarun.shの中身



Policy:1

```
#!/bin/bash
MYRANK=$MXMPI_ID MPIのプロセス番号
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --interleave=all $@
```

Policy:2

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --interleave=$SOCKET $@
```

Policy:3

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --membind=$SOCKET $@
```

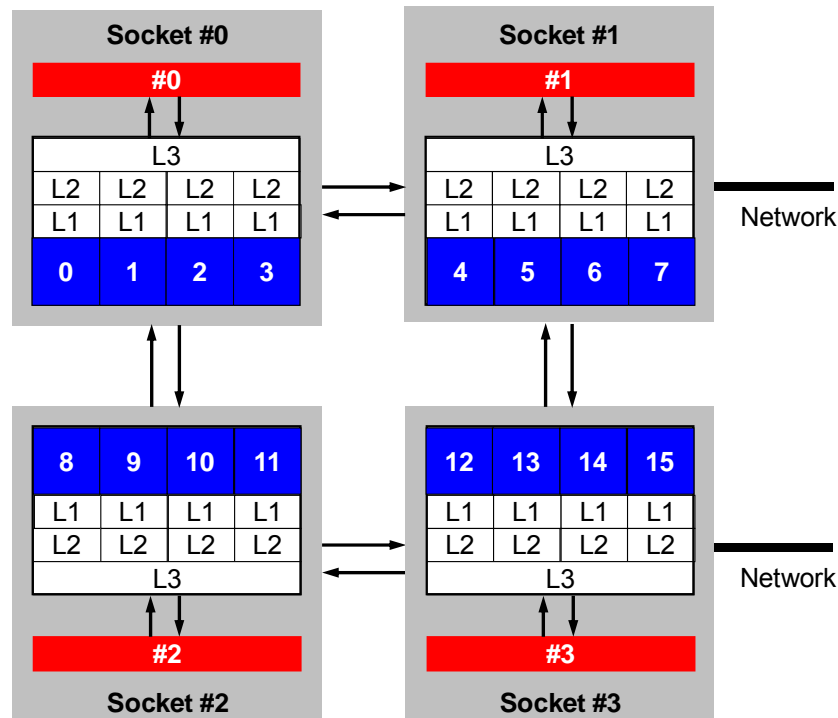
Policy:4

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --localalloc $@
```

Policy:5

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --localalloc $@
```

Policy-3の割り当て

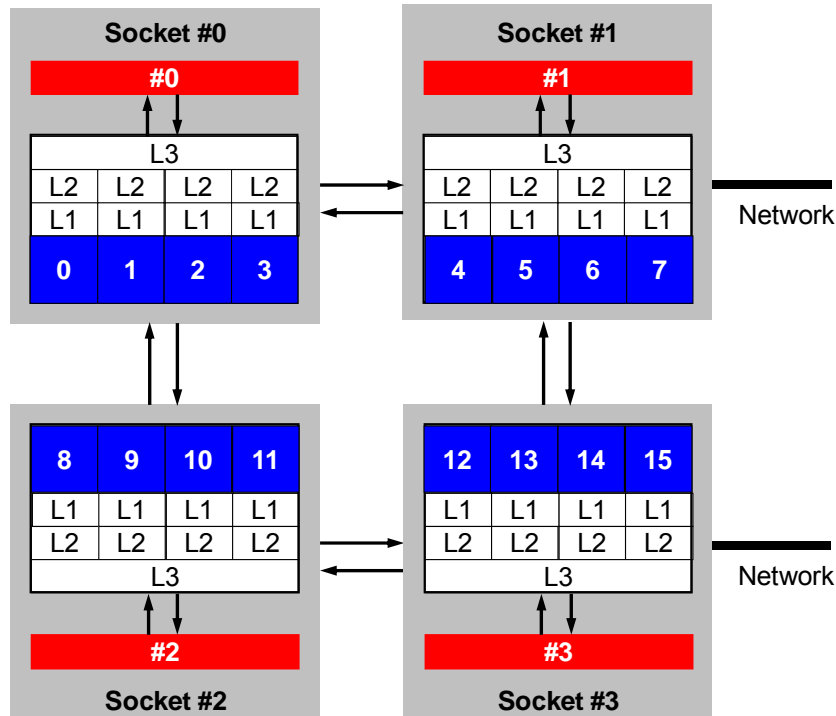


Policy:3

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --membind=$SOCKET $@
```

MPI process	Socket	Memory	Core
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	1	1	
5	1	1	
6	1	1	
7	1	1	
8	2	2	
9	2	2	
10	2	2	
11	2	2	
12	3	3	
13	3	3	
14	3	3	
15	3	3	

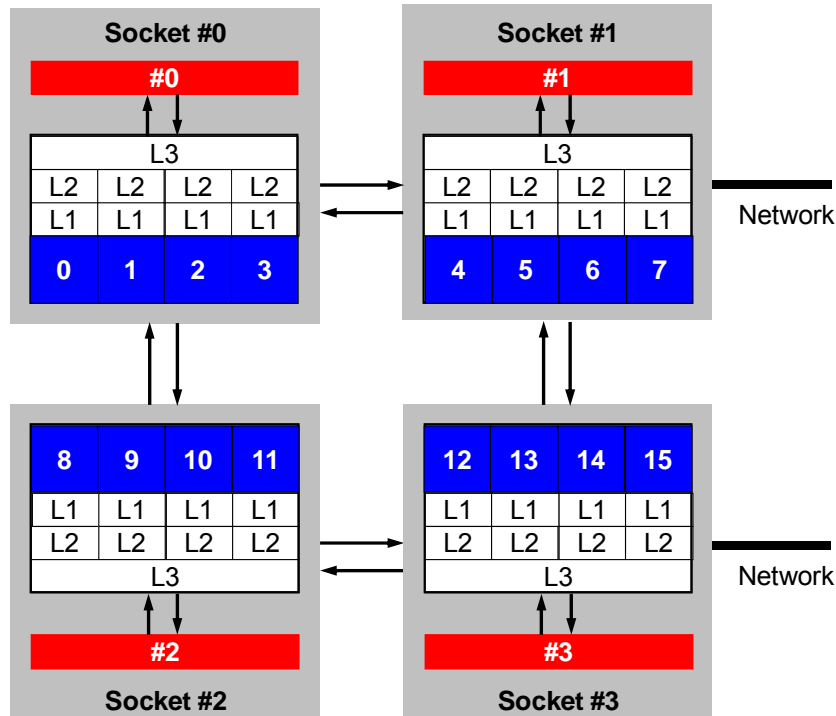
Cyclicな割り付けも可能



```
#!/bin/bash
MYRANK=$MXMPI_ID
SOCKET=$(expr $MYRANK % 4)
numactl --cpunodebind=$SOCKET --membind=$SOCKET $@
```

MPI process	Socket	Memory	Core
0	0	0	
1	1	1	
2	2	2	
3	3	3	
4	0	0	
5	1	1	
6	2	2	
7	3	3	
8	0	0	
9	1	1	
10	2	2	
11	3	3	
12	0	0	
13	1	1	
14	2	2	
15	3	3	

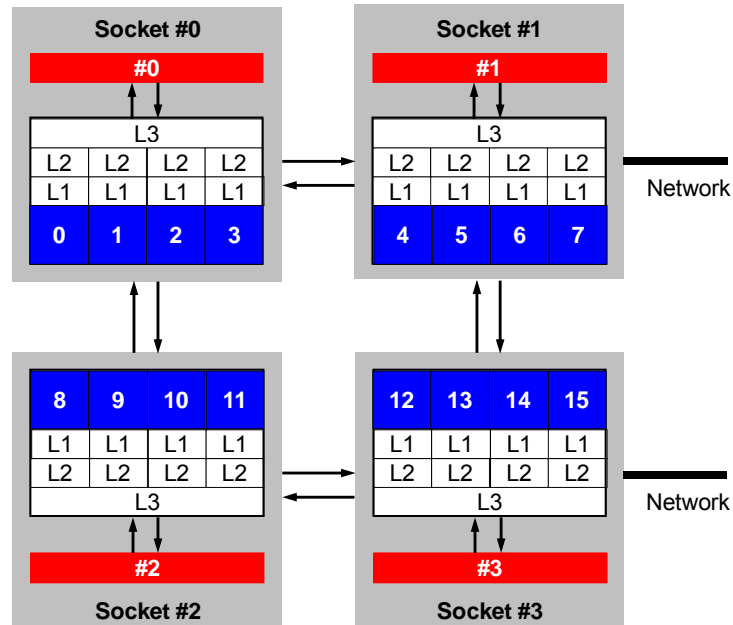
コアを指定することもできる



```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$((expr $MYRANK / 4))
SOCKET=$((expr $MYVAL % 4))
CORE=$((expr $MYRANK % 16))
numactl --physcpubind=$CORE --membind=$SOCKET $@
```

MPI process	Socket	Memory	Core
0		0	0
1		0	1
2		0	2
3		0	3
4		1	4
5		1	5
6		1	6
7		1	7
8		2	8
9		2	9
10		2	10
11		2	11
12		3	12
13		3	13
14		3	14
15		3	15

8コア使う場合はどうする？



MPI process	Socket	Memory	Core
0		0	0
1		0	2
2		1	4
3		1	6
4		2	8
5		2	10
6		3	12
7		3	14

MPI process	Socket	Memory	Core
0		0	0
1		0	1
2		0	2
3		0	3
4		1	4
5		1	5
6		1	6
7		1	7

MPI process	Socket	Memory	Core
0		0	0
1		0	1
2		1	4
3		1	5
4		2	8
5		2	9
6		3	12
7		3	13

STREAM ベンチマーク

<http://www.streambench.org/>

- メモリバンド幅を測定するベンチマーク
 - Copy, Scale, Add, Triad (DAXPYと同じ)

```
-----  
Double precision appears to have 16 digits of accuracy  
Assuming 8 bytes per DOUBLE PRECISION word  
-----
```

```
Number of processors =          16  
Array size =      2000000  
Offset      =          0  
The total memory requirement is    732.4 MB (    45.8MB/task)  
You are running each test  10 times  
--
```

```
The *best* time for each test is used  
*EXCLUDING* the first and last iterations  
-----
```

Function	Rate (MB/s)	Avg time	Min time	Max time
Copy:	18334.1898	0.0280	0.0279	0.0280
Scale:	18035.1690	0.0284	0.0284	0.0285
Add:	18649.4455	0.0412	0.0412	0.0413
Triad:	19603.8455	0.0394	0.0392	0.0398

ファイル:今回はFORTRANのみ

```
>$ cd <$FVM>/stream  
>$ mpif90 -Oss -noparallel stream.f -o stream
```

今回はMPI版だが, OpenMP版もある

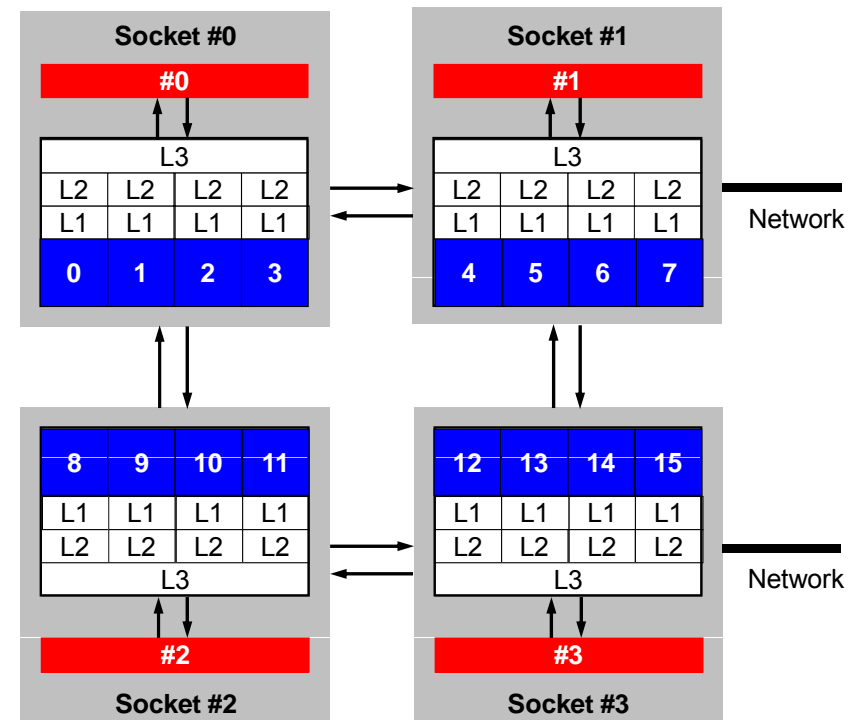
set-a0.sh: Policy-5相当

- プロセスが割り当てられたソケットのローカルメモリを使う
- ソケット番号は基本的にサイクリック(round-robin的)に割り当てられるが, 異なる場合もある(実行ごとに性能が変動)

```
#@$-r stream
#@$-q lecture
#@$-N 1
#@$-J T16
#@$-e err
#@$-o a0-16-1.1st
#@$-lM 28GB
#@$-lT 00:05:00
#@$
```

```
cd $PBS_O_WORKDIR
```

```
mpirun numactl --localalloc ./stream
exit
```



set-b1/b2.sh: Policy-3,4相当

- 各プロセスをソケット0番から順番に埋まるように割り当て, 各ソケットのローカルメモリを使う
- b1とb2ではあまり差は無い

set-b1.sh

```
#@$-r stream
#@$-q lecture
#@$-N 1
#@$-J T8
#@$-e err
#@$-o b1-08-1.lst
#@$-lM 28GB
#@$-lT 00:05:00
#@$

cd $PBS_O_WORKDIR

mpirun ./b1.sh ./stream
exit
```

b1.sh : Policy-3相当

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --membind=$SOCKET $@
```

b2.sh : Policy-4相当

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET --localalloc $@
```

set-x.sh

- 各プロセスが割り当てられたソケットのメモリを使う
- T2:0,1, T4:0,1,2,3
- T8:0,0,1,1,2,2,3,3
- T16:0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3

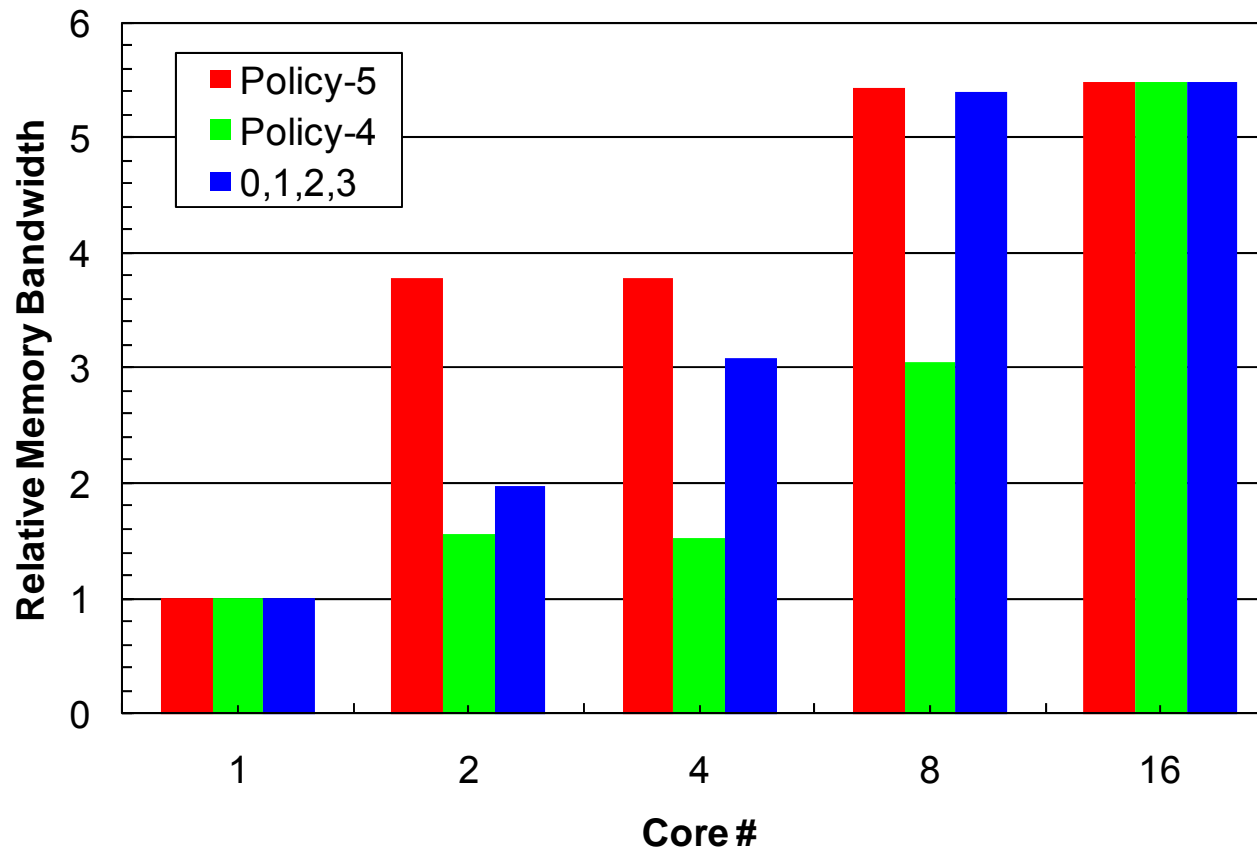
```
#@$-r stream
#@$-q lecture
#@$-N 1
#@$-J T16
#@$-e err
#@$-o x1-16-1.lst
#@$-lM 28GB
#@$-lT 00:05:00
#@$
```

```
cd $PBS_O_WORKDIR
```

```
mpirun numactl --cpunodebind=0,1,2,3 --localalloc ./stream
exit
```

Triadの結果: Policy-5@T1の性能を1.00

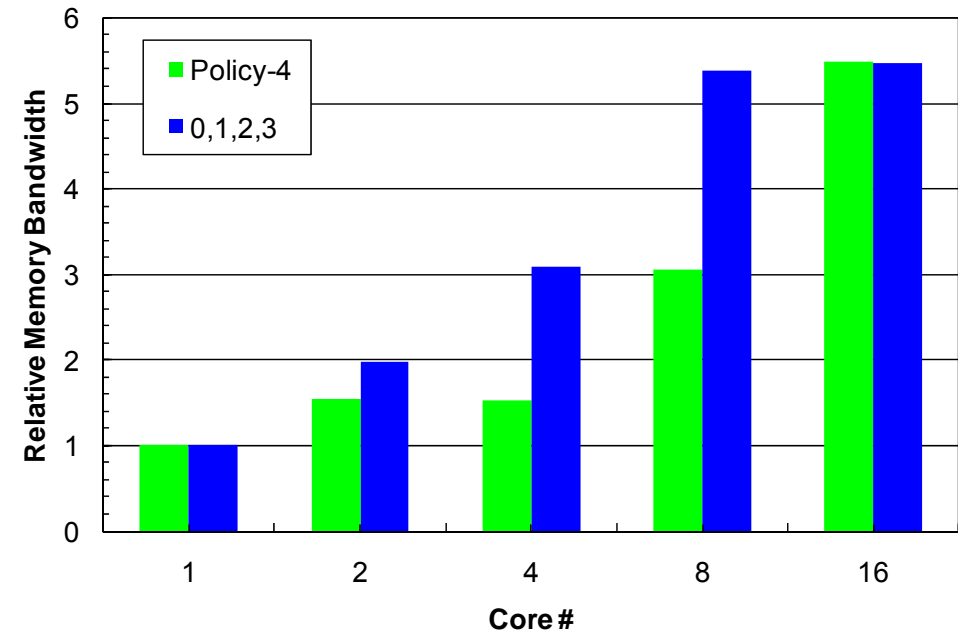
- Policy-5: set-a0, Policy-4: set-b2
- 0,1,2,3: set-x
- 16コア使った場合, メモリの性能は5倍強にしかっていない



Triadの性能:T2

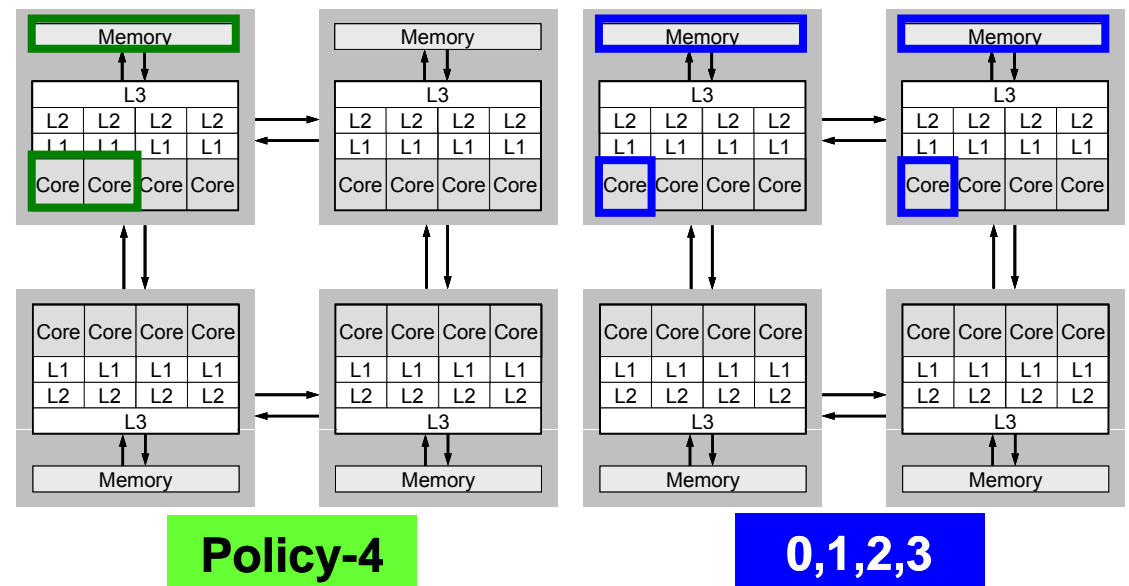
- Policy-4

- 1コアのときの2倍までは行かないが性能は向上
- 2コアで1つのメモリを共有するため、多少の競合が生じている



- 0,1,2,3

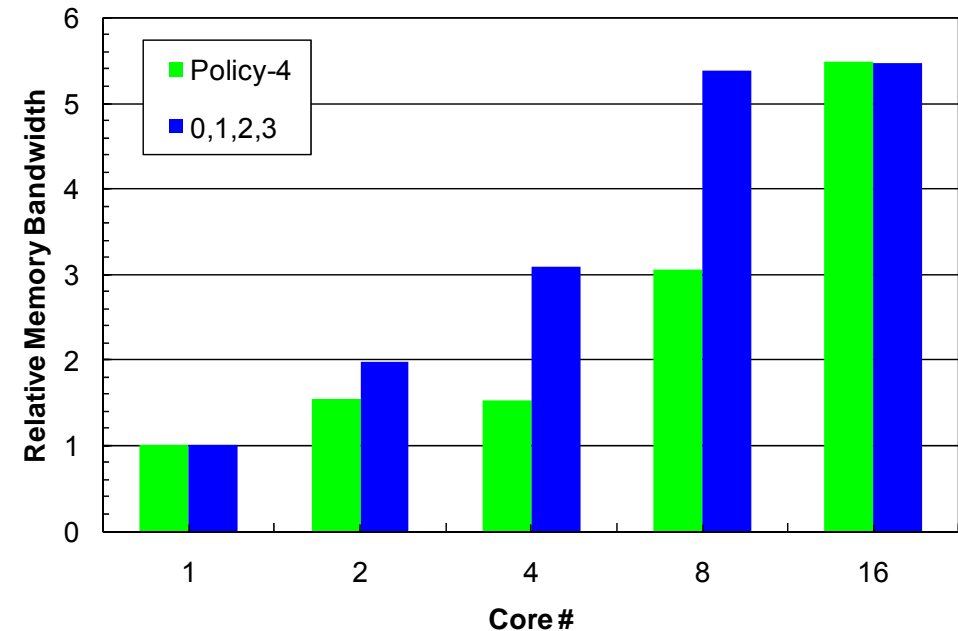
- ほぼ100%近い性能向上
- 各コアでローカルメモリを占有



Triadの性能:T4

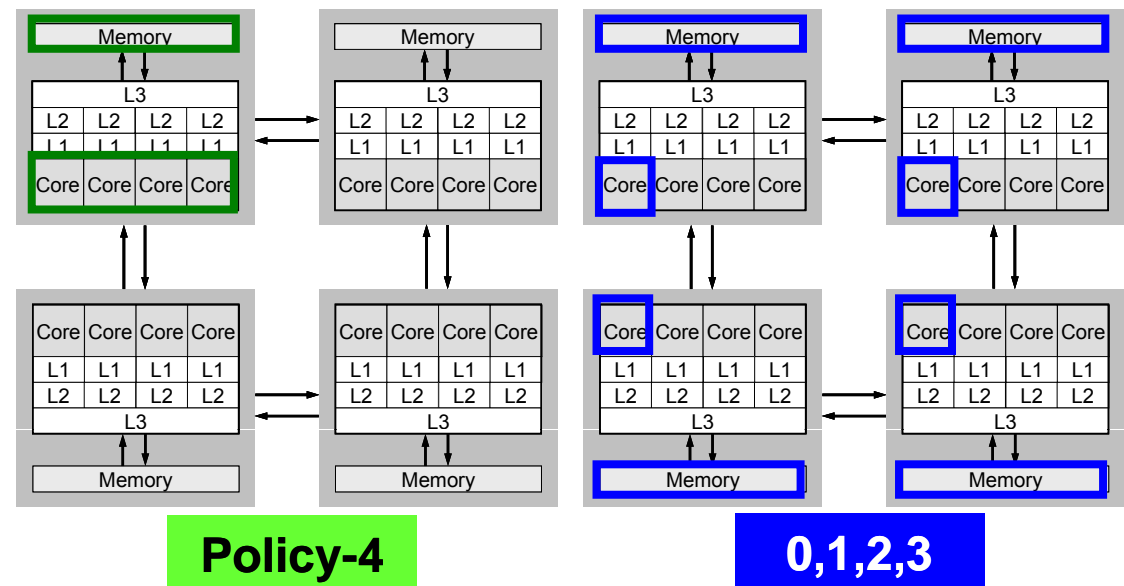
- Policy-4

- T2のときとほとんど変わらない, つまり1コア当たりのメモリ性能は半分に落ちている
- 飽和状態



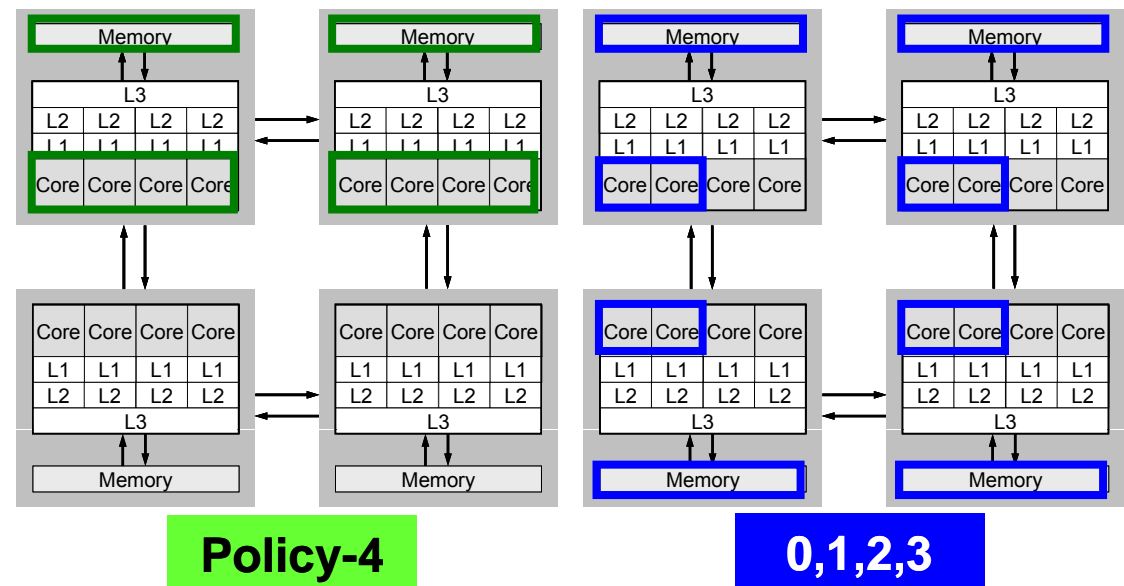
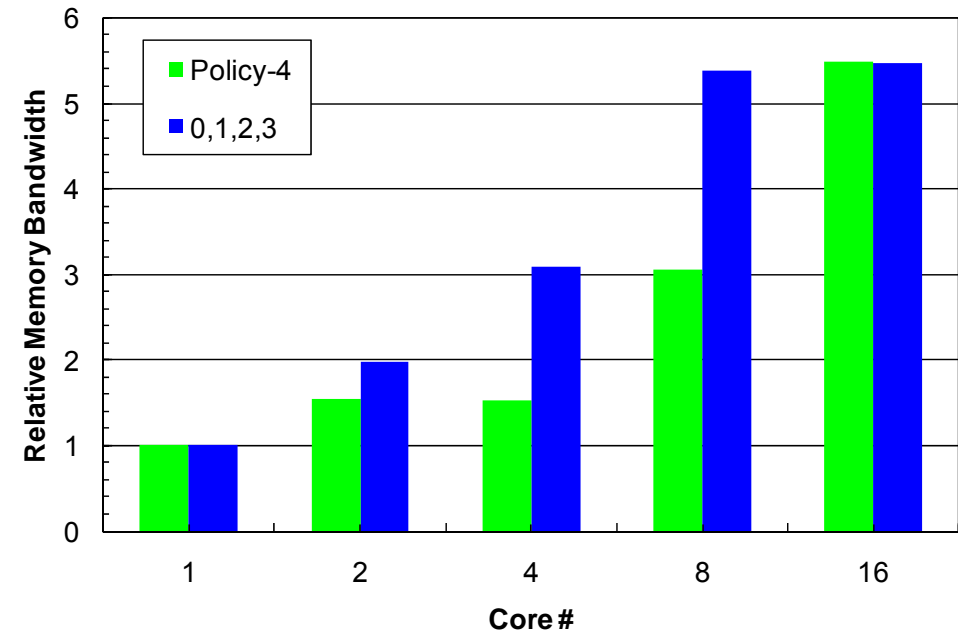
- 0,1,2,3

- T1の4倍までは行かないが順調にスケール
- Cache-Coherency
- 通信の影響もあり



Triadの性能: T8, T16

- Policy-4
 - T8
 - T4の約2倍弱
 - T16
 - T8の約2倍弱
- 0,1,2,3
 - T8
 - T4の約2倍弱
 - T16
 - T8とほぼ同じ
 - Policy-4におけるT2→T4と同じ



T2Kの性能

- 実は余りノード内の性能について細かく測定したことは無い
 - 1ノードが最低単位
- メモリの性能(の悪さ)に充分注意する必要がある
 - Strong Scaling(同じ問題をコア数を変えて解く)の場合, 基準を1ノードあるいは1ソケット(Policy-2,3,4のようにする)にする

疎行列ソルバーの性能: 三次元弾性問題

ICCG法, T2K・SR11000 1ノード: メモリバンド幅が効く

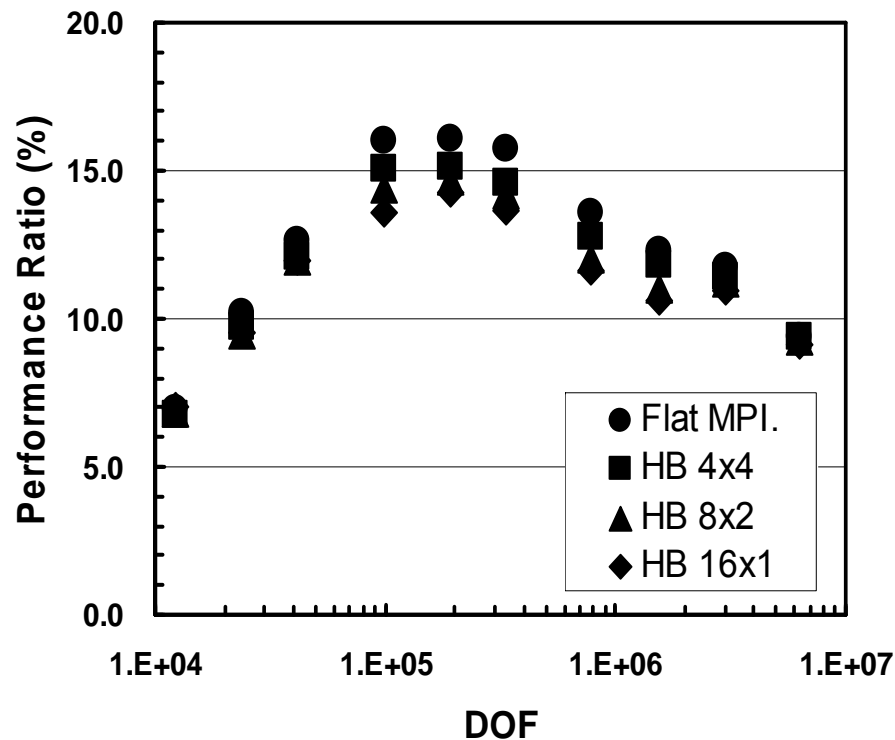
Hitachi SR11000/J2

Power 5+ 2.3GHz x 16

147.2 GFLOPS/node

100 GB/s for STREAM/Triad

L3 cache: 18MB/core



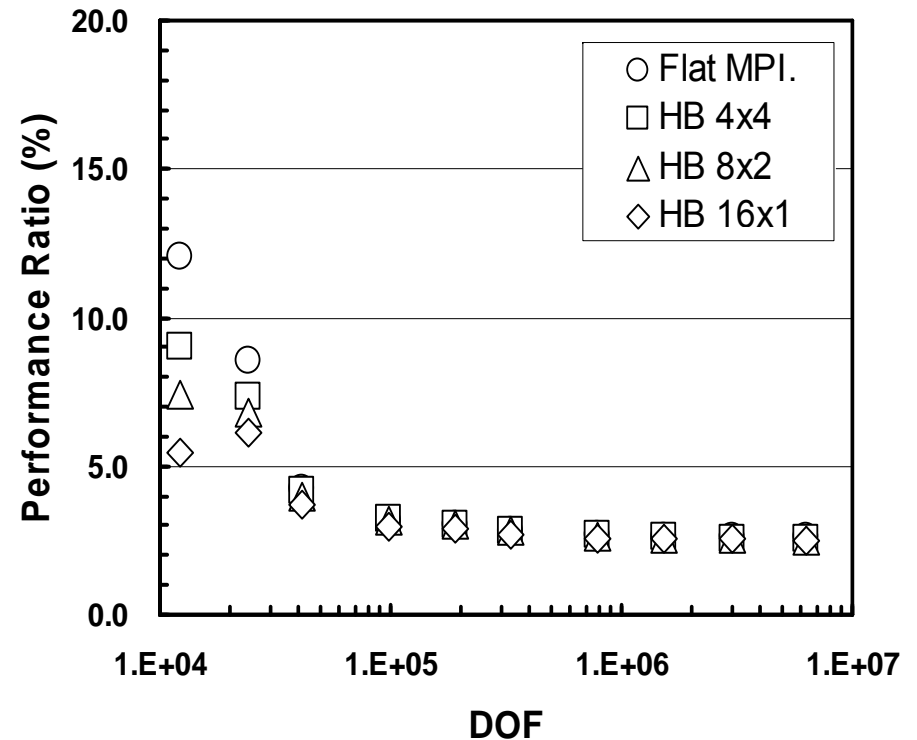
T2K/Tokyo

Opteron 2.3GHz x 16

147.2 GFLOPS/node

20 GB/s for STREAM/Triad

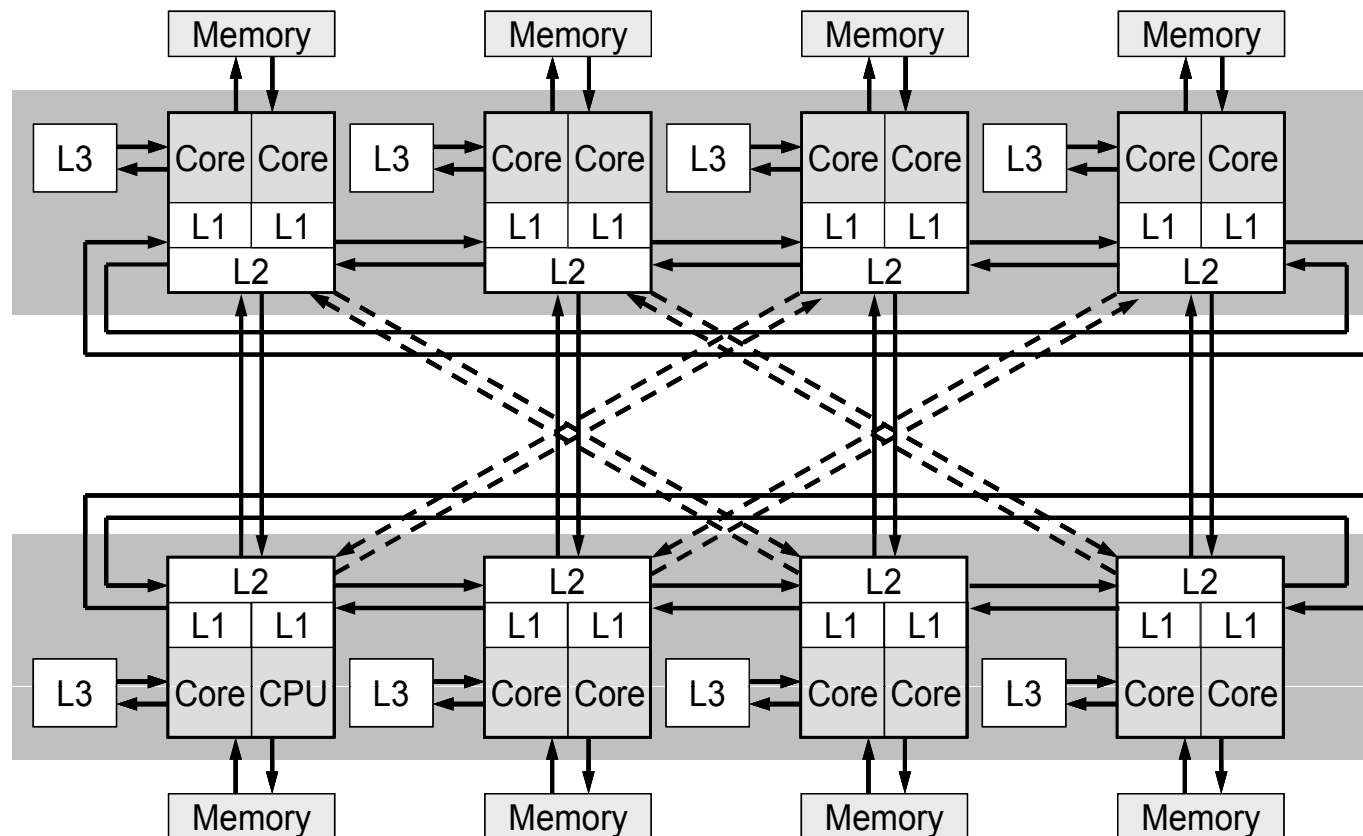
L3 cache: 0.5MB/core



Hitachi SR11000/J2

- IBM POWER5+ 2.3GHz (9.2GFLOPS)
- Dual Core × 4 ⇒ 「MCM (Multi Core Module)」 × 2 ⇒ 1node (16cores)
- based on NUMA architectures, but small latency of memory, huge cache

青木, 中村, 助川, 齋藤, 深川, 中川, 五百木 (2005)
スーパーテクニカルサーバーSR11000 モデルJ1のノードアーキテクチャと性能評価
情報処理学会論文誌: コンピューティングシステム 45-SIG12 (ACS11), 27-36 より作成

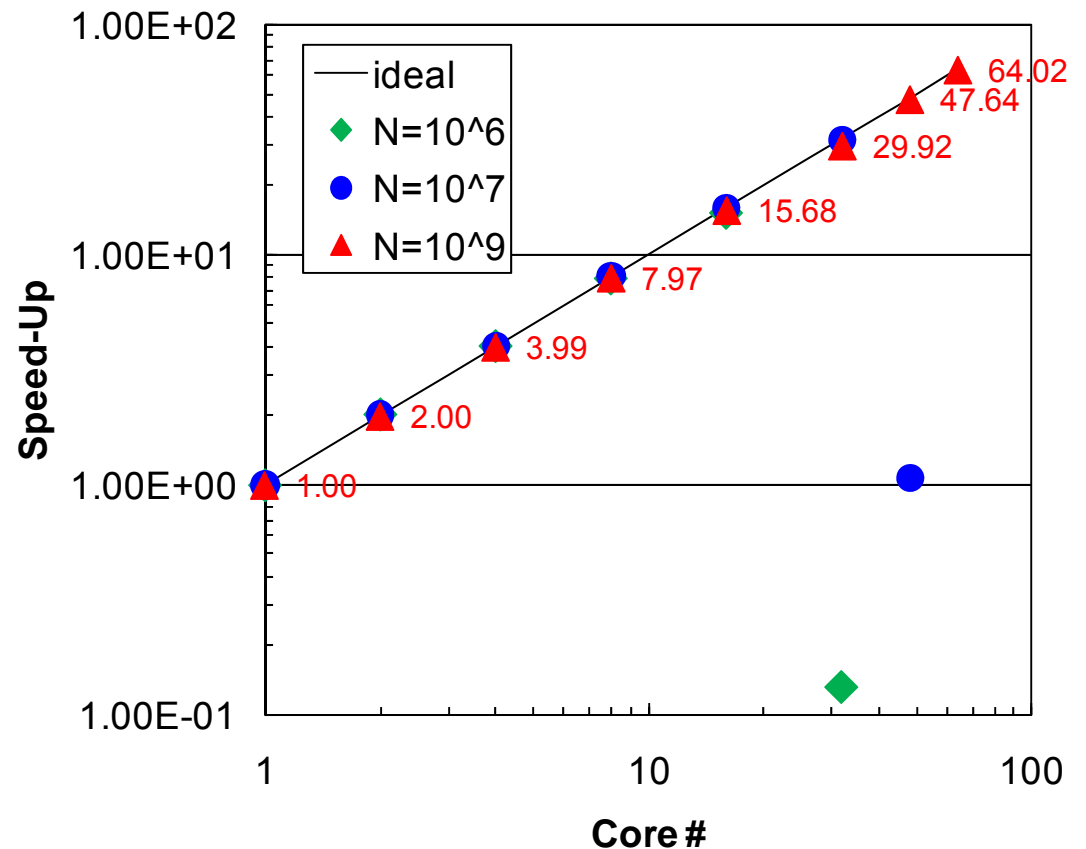


台形積分:T2K(東大)における並列効果

- ◆ : $N=10^6$, ● : 10^7 , ▲ : 10^9
- : 理想値
- 1コアにおける計測結果(sec.)からそれぞれ算出

```
#@$-r test
#@$-q lecture
#@$-N 1
#@$-J T4
#@$-e err
#@$-o hello.lst
#@$-lM 28GB
#@$-lT 00:05:00
#@$

cd $PBS_O_WORKDIR
mpirun numactl --localalloc ./a.out
```



- 赤字部分の影響
- 1ノードより多いと変動あり
- メモリに負担のかからない計算