Reedbush-U/H

Integrated Supercomputer System for Data Analyses & Scientific Simulations

- SGI (HPW) was awarded (Mar. 22, 2016)
- Compute Nodes (CPU only): Reedbush-U
  - Intel Xeon E5-2695v4 (Broadwell-EP, 2.1GHz 18core) x 2socket (1.210 TF), 256 GiB (153.6GB/sec)
  - InfiniBand EDR, Full bisection Fat-tree
  - Total System: 420 nodes, 508.0 TF
- Compute Nodes (with Accelerators): Reedbush-H
  - Intel Xeon E5-2695v4 (Broadwell-EP, 2.1GHz 18core) x 2socket, 256 GiB (153.6GB/sec)
  - NVIDIA Pascal GPU (Tesla P100)
    - (5.3TF, 720GB/sec, 16GiB) x 2 / node
  - InfiniBand FDR x 2ch (for ea. GPU), Full bisection Fat-tree
  - 120 nodes, 145.2 TF(CPU)+ 1.27 PF(GPU)= 1.42 PF

Now operating 3 Systems !!

2,600+ users (55+% from outside of U.Tokyo)

- Reedbush (HPE, Intel BDW + NVIDIA P100 (Pascal))
  - Integrated Supercomputer System for Data Analyses & Scientific Simulations
    - Jul.2016-Mar.2021
  - Our first GPU System, DDN IME (Burst Buffer)
  - Reedbush-U: CPU only, 420 nodes, 508 TF (Jul.2016-June 2020)
  - Reedbush-H: 120 nodes, 2 GPUs/node: 1.42 PF (Mar.2017)
  - Reedbush-L: 64 nodes, 4 GPUs/node: 1.43 PF (Oct.2017)

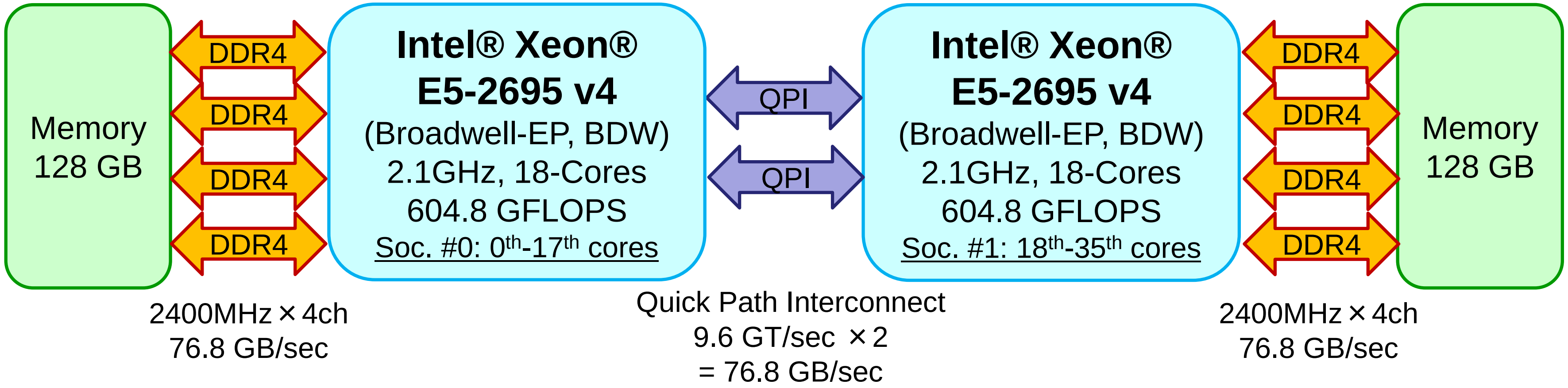
- Oakforest-PACS (OFP) (Fujitsu, Intel Xeon Phi (KNL))
  - JCAHPC (U.Tsukuba & U.Tokyo)
  - 25 PF, #18 in 55th TOP 500 (June 2020) (#3 in Japan)
  - Omni-Path Architecture, DDN IME (Burst Buffer)

- Oakbridge-CX (OBCX) (Fujitsu, Intel Xeon Platinum 8280, CLX)
  - Massively Parallel Supercomputer System
  - 6.61 PF, #60 in 55th TOP 500, July 2019-June 2023
  - SSD's are installed to 128 nodes (out of 1,368)

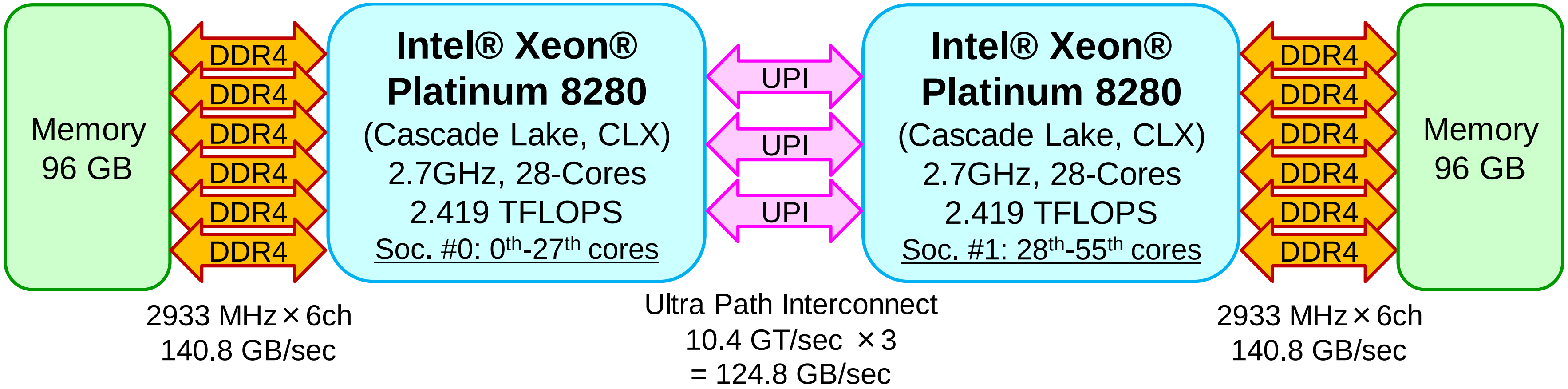
JPY (=Watt)/GFLOPS Rate
GFLOPS/W（Green 500）

System	JPY/GFLOPS Small is Good	GFLOPS/W Large is Good
Oakleaf-FX/Oakbridge-FX (Fujitsu) (Fujitsu SPARC64 IXfx)	125	0.8663
Reedbush-U (HPE) (Intel BDW)	61.9	2.310
Reedbush-H (HPE) (Intel BDW+NVIDIA P100x2/node)	15.9	8.575
Reedbush-L (HPE) (Intel BDW+NVIDIA P100x4/node)	13.4	10.167
Oakforest-PACS (Fujitsu) (Intel Xeon Phi/Knights Landing)	16.5	4.986
Oakbridge-CX (Fujitsu) (Intel Cascade Lake (CLX))	20.7	5.076

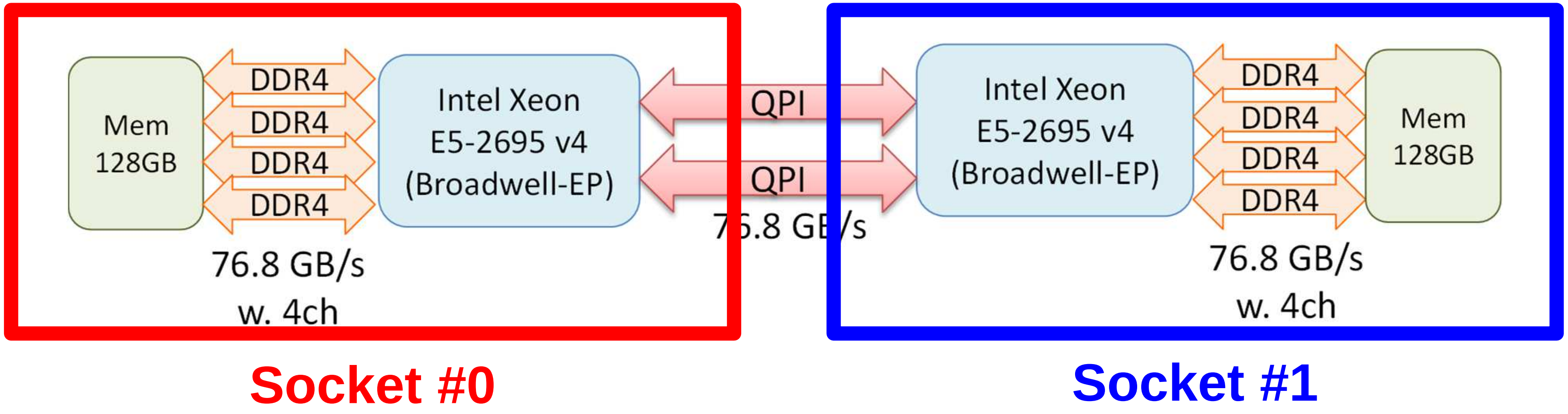
Reedbush-U: intel/18.1.163 (default)



Oakbridge-CX (OBCX): intel/2019.5.281 (default)



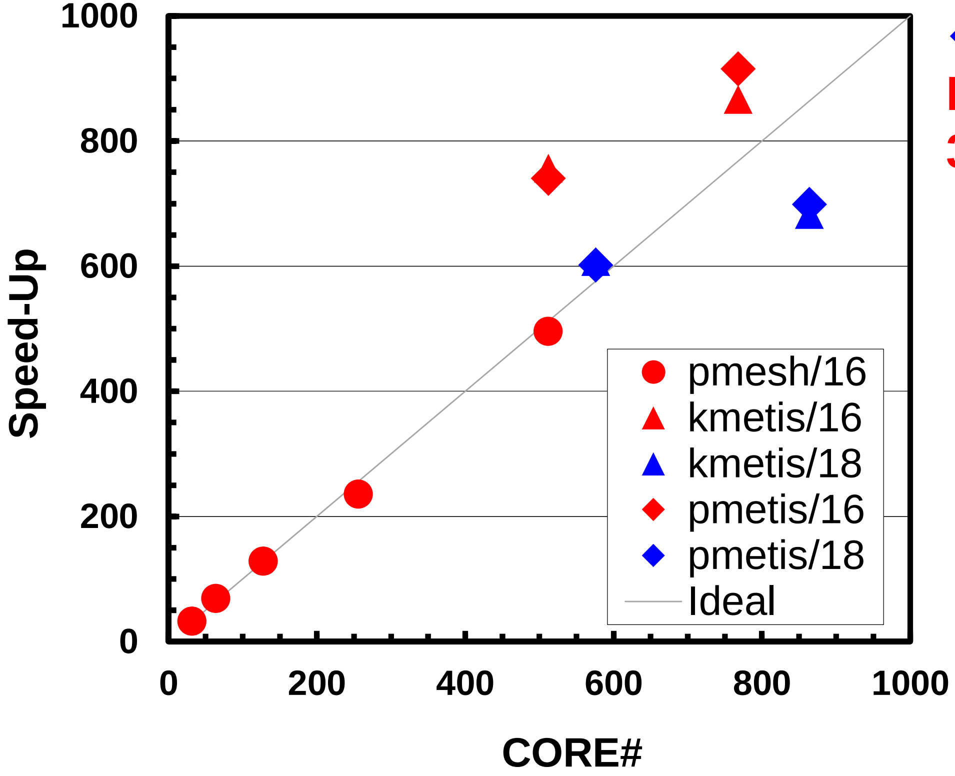
1 Node = 2 Sockets (CPU's)



- Each Node of Reedbush-U
  - 2 Sockets (CPU's) of Intel Broadwell-EP
  - Each socket has 18 cores
- Each core of a socket can access to the memory on the other socket : NUMA (Non-Uniform Memory Access)
  - I_MPI_PIN_DOMAIN=socket, impimap.sh: local memory to be used

Example: Strong Scaling: Fortran

- 256×128×128 nodes
  - 4,194,304 nodes, 4,112,895 elements
- 32～864 cores
- Linear Solver



pmesh/16: pmesh 16 cores/socket
kmetis/16: kmetis 16 cores/socket
kmetis/18: kmetis 18 cores/socket
pmetis/16: pmetis 16 cores/sockets
pmetis/18: pmetis 18 cores/sockets

Performance of pmesh/16 w/32 cores= 32.0

```
256 128 128
  8   2   2
pcube
```

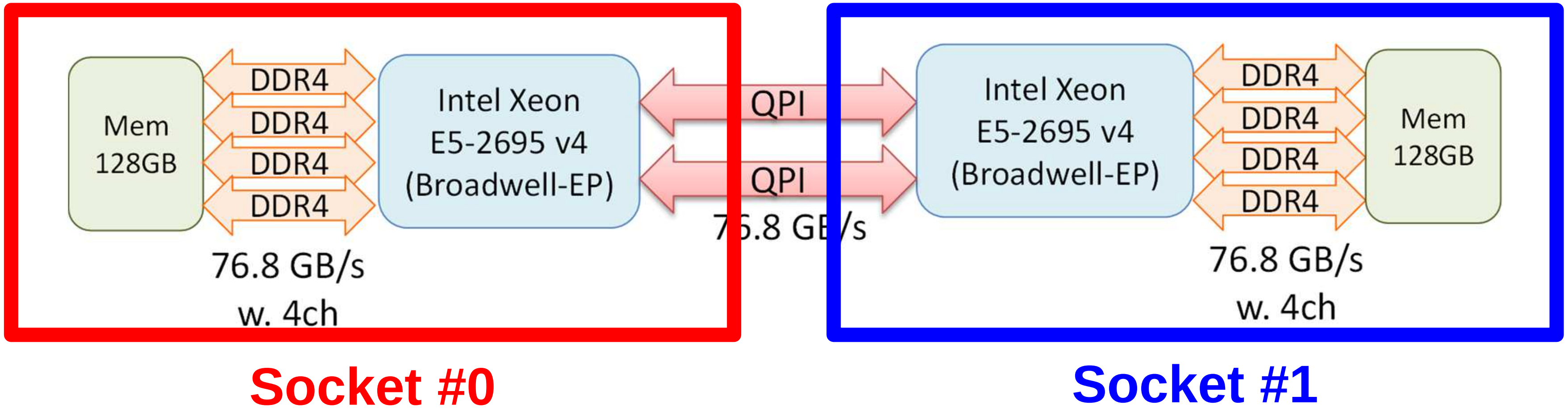
```
select=1:
mpiprocs=32
```

```
256 128 128
  8   4   2
pcube
```

```
select=2:
mpiprocs=32
```

```
256 128 128
 16   4   4
pcube
```

```
select=8:
mpiprocs=32
```

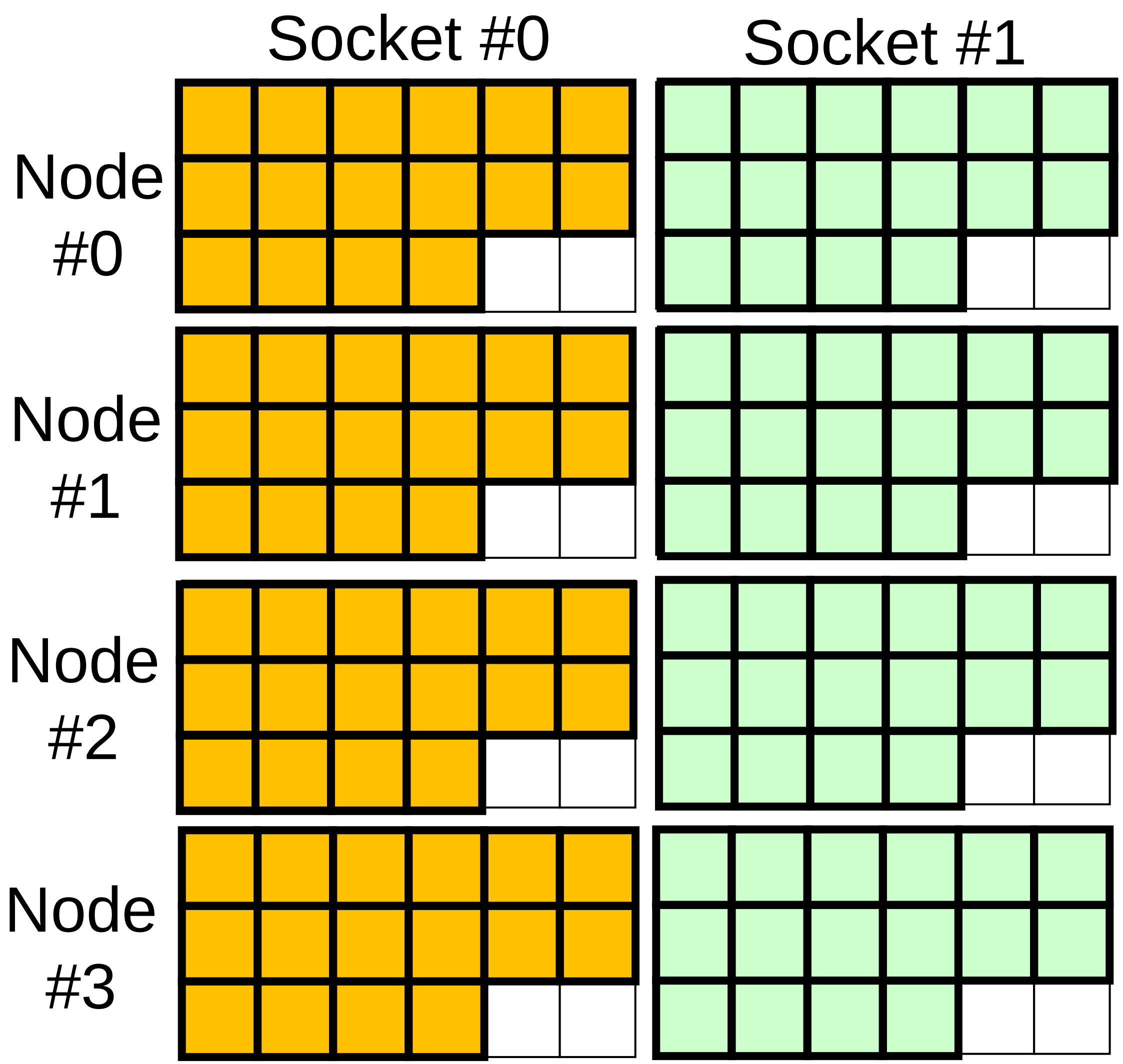


HB M x N

Number of OpenMP threads per a single MPI process

Number of MPI process per a single “socket”

4-nodes/8-sockets: 128 MPI process's Flat MPI, 32 MPI process's/Node



mesh.inp
```
256 128  64
 16   8   1
pcube
```

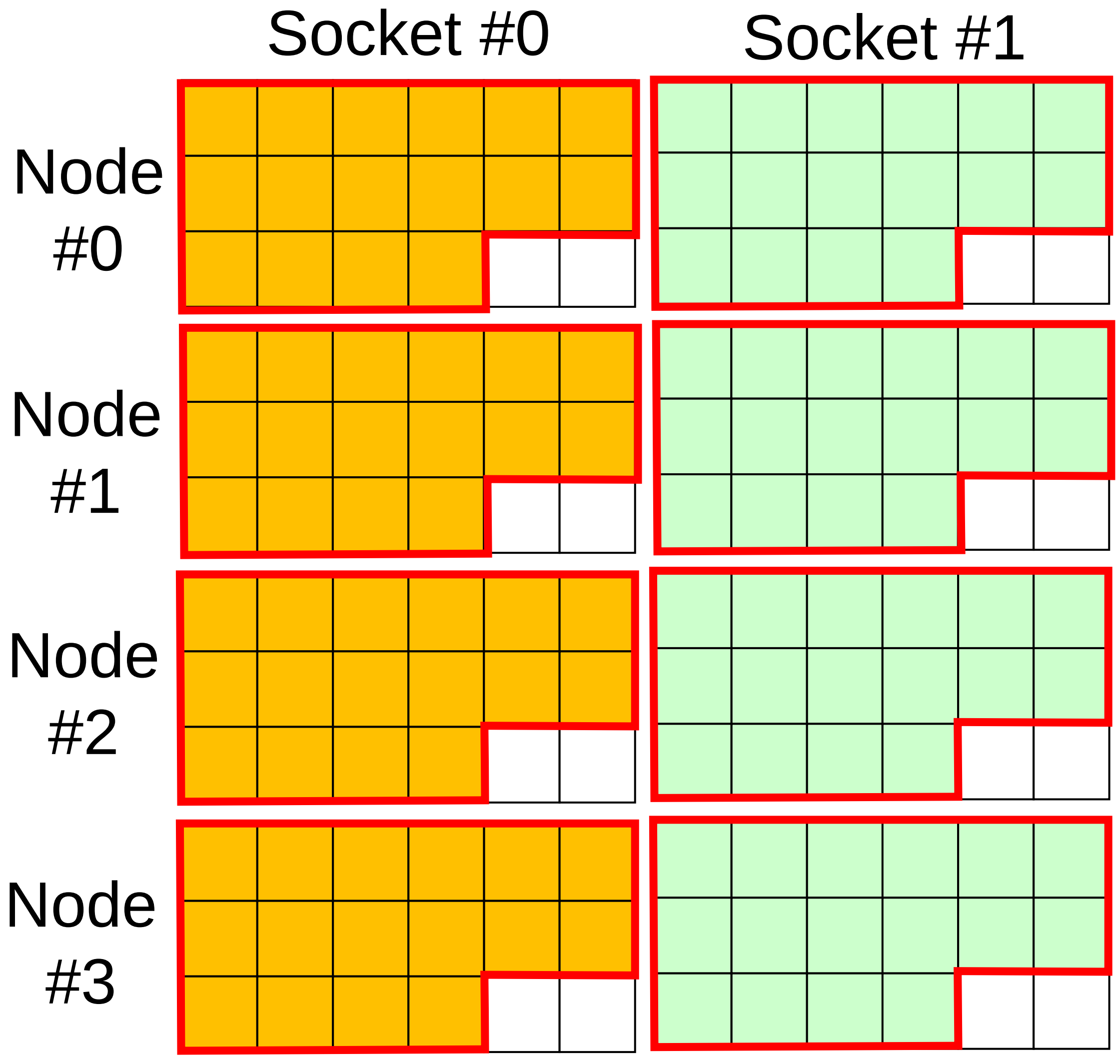
inp_mg
```
256 128  64
```

inp_kmetis
```
cube.0
2
128
pcube
```

```
select=4:
mpiprocs=32

I_MPI_PERHOST=32
```

16/18 cores/socket

4-nodes: 16-threads x 8 MPI process's HB 16x1, 2 MPI process's/Node




```
mesh.inp
256 128  64
  4   2   1
pcube
```

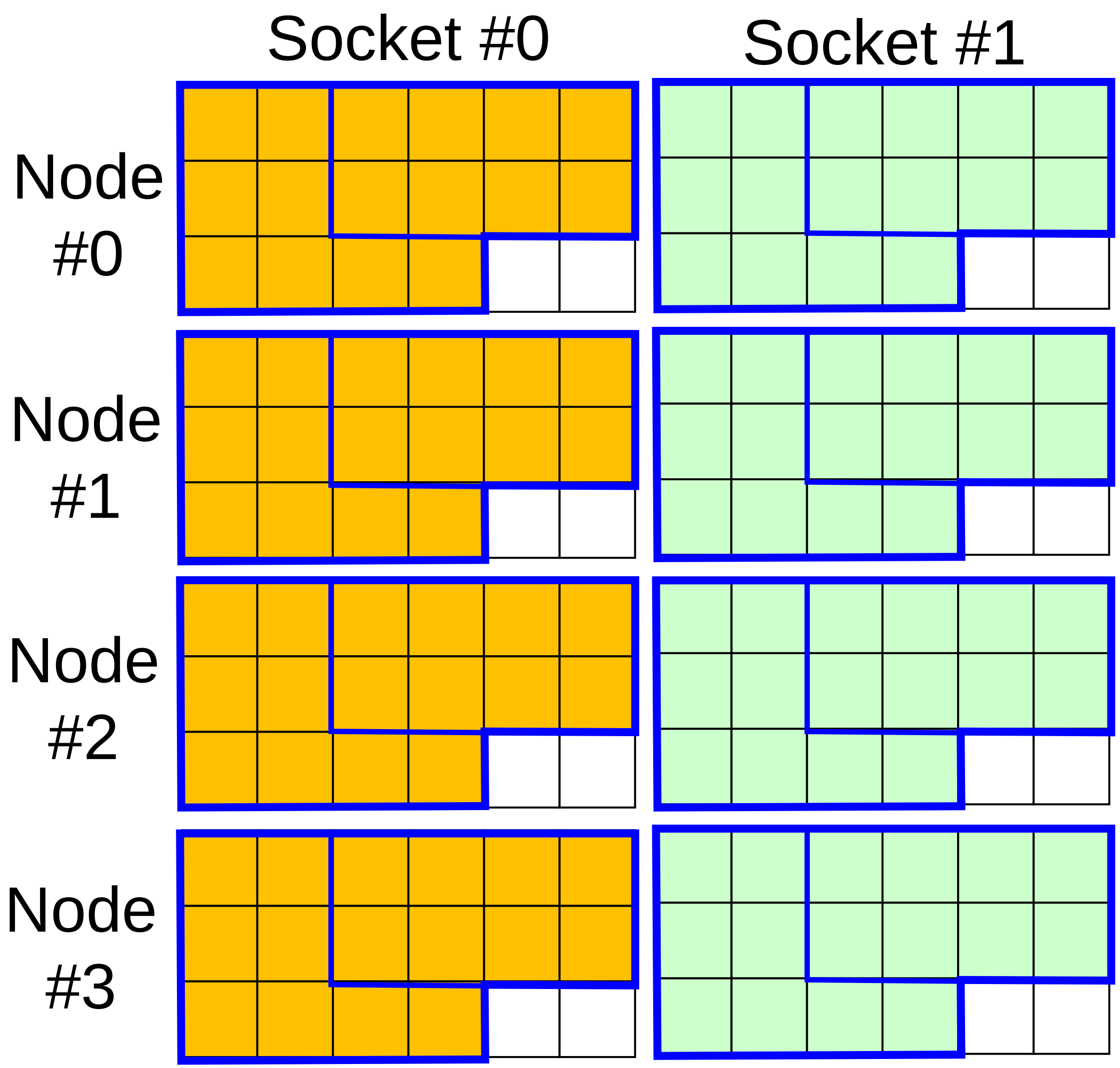
```
inp_mg
256 128  64
```

```
inp_kmetis
cube.0
2
8
pcube
```

```
select=4:
mpiprocs=2

I_MPI_PERHOST=2
OMP_NUM_THREADS=16
```

16/18 cores/socket

4-nodes: 8-threads x 16 MPI process's
HB 8x2, 4 MPI process's/Node




```
mesh.inp
256 128  64
  4   4   1
pcube
```

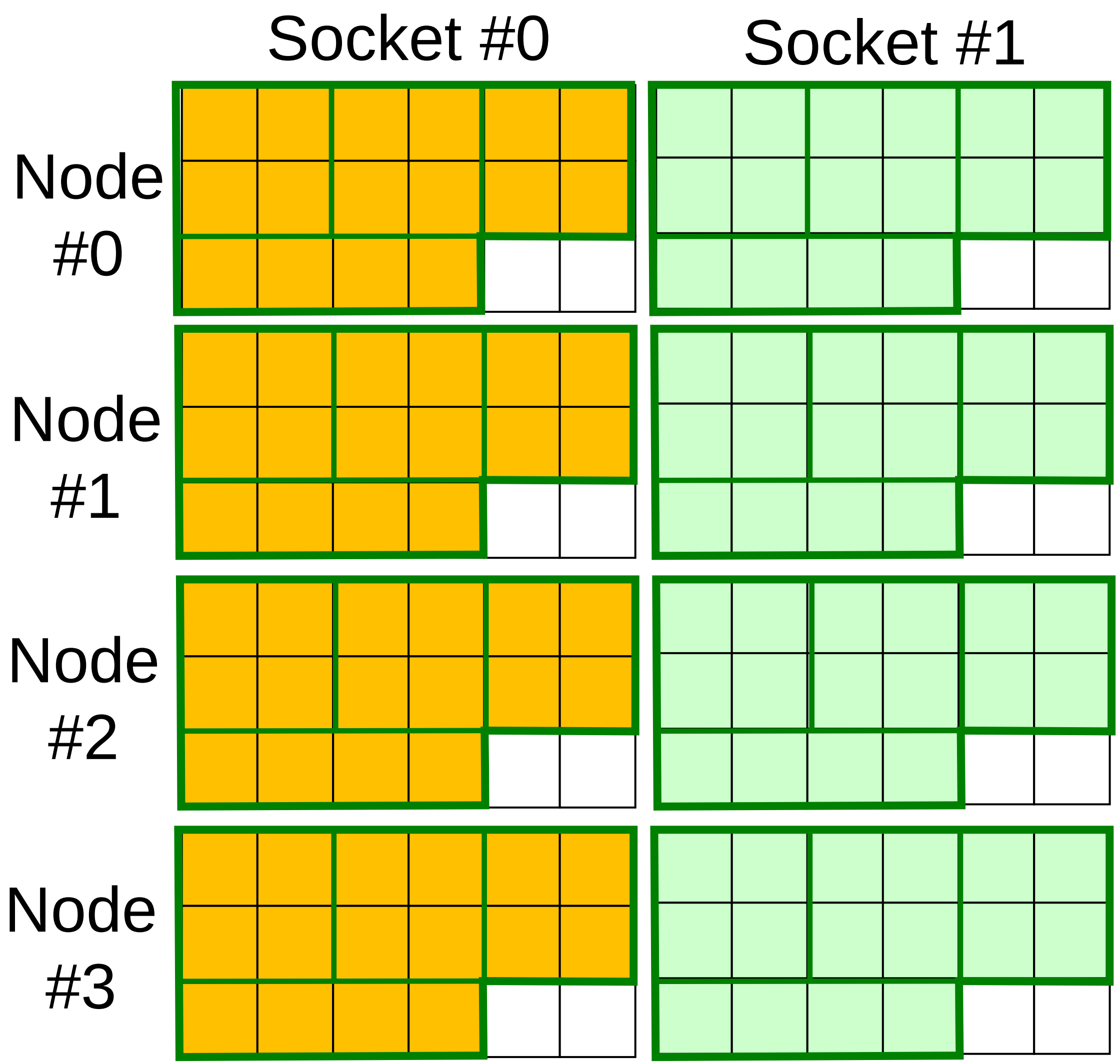
```
inp_mg
256 128  64
```

```
inp_kmetis
cube.0
2
16
pcube
```

```
select=4:
mpiprocs=4

I_MPI_PERHOST=4
OMP_NUM_THREADS=8
```

16/18 cores/socket

4-nodes: 4-threads x 32 MPI process's
HB 4x4, 8 MPI process's/Node



mesh.inp
```
256 128  64
  8   4   1
pcube
```

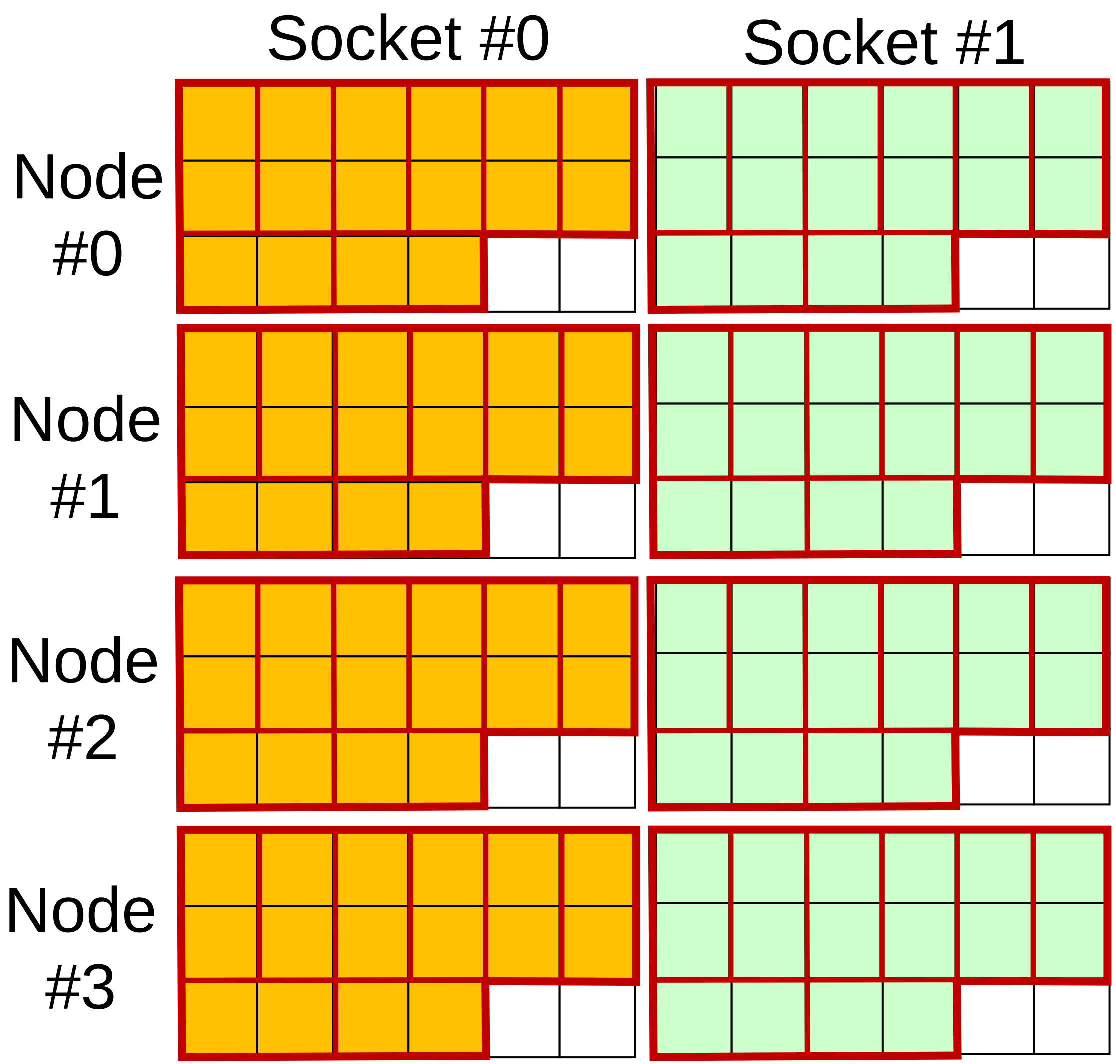
inp_mg
```
256 128  64
```

inp_kmetis
```
cube.0
2
32
pcube
```

```
select=4:
mpiprocs=8

I_MPI_PERHOST=8
OMP_NUM_THREADS=4
```

16/18 cores/socket

4-nodes: 2-threads x 64 MPI process's HB 2x8, 16 MPI process's/Node




```
mesh.inp
256 128  64
  8   8   1
pcube
```

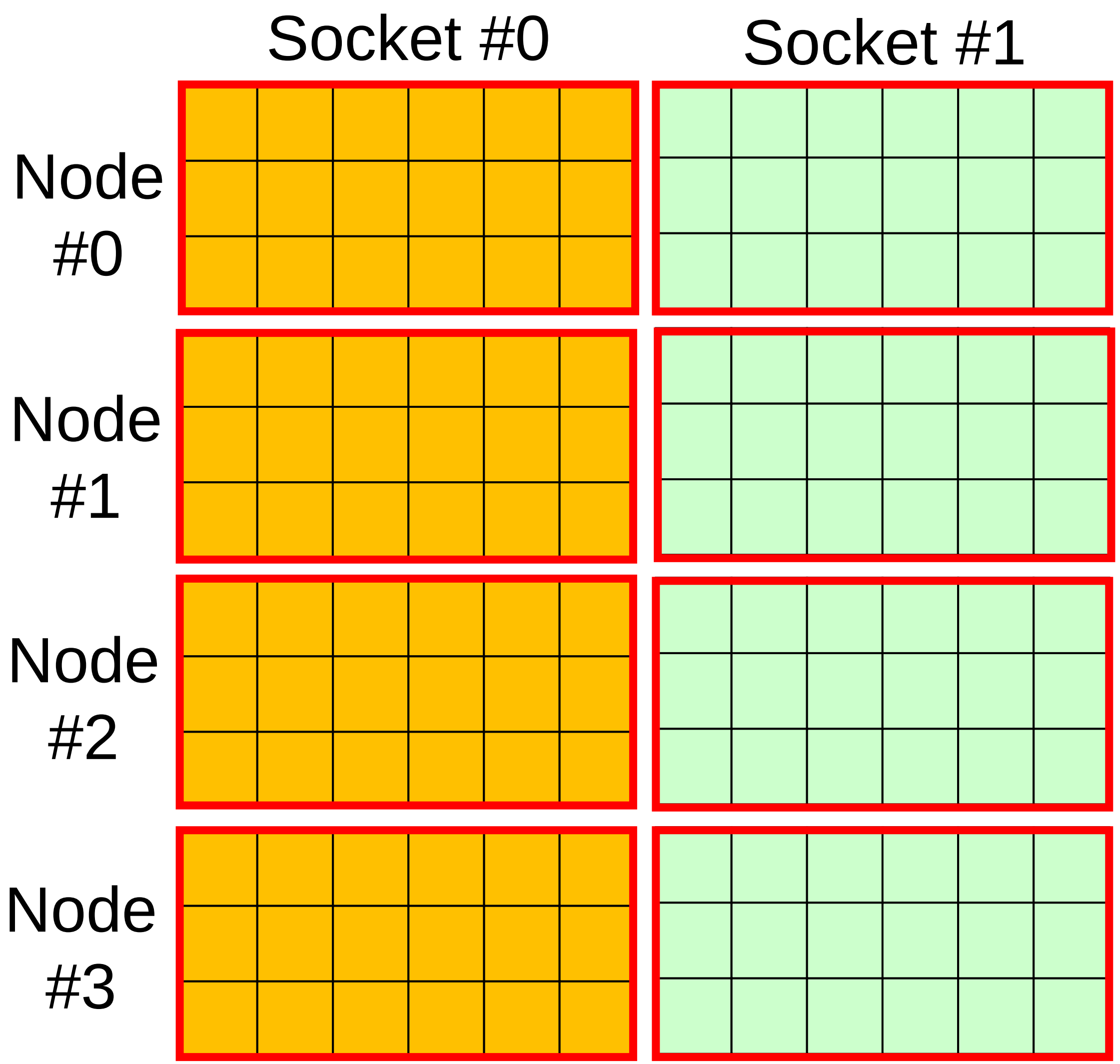
```
inp_mg
256 128  64
```

```
inp_kmetis
cube.0
2
64
pcube
```

```
select=4:
mpiprocs=16

I_MPI_PERHOST=16
OMP_NUM_THREADS=2
```

16/18 cores/socket

4-nodes: 18-threads x 8 MPI process's
HB 18x1, 2 MPI process's/Node




```
mesh.inp
256 128  64
  4   2   1
pcube
```

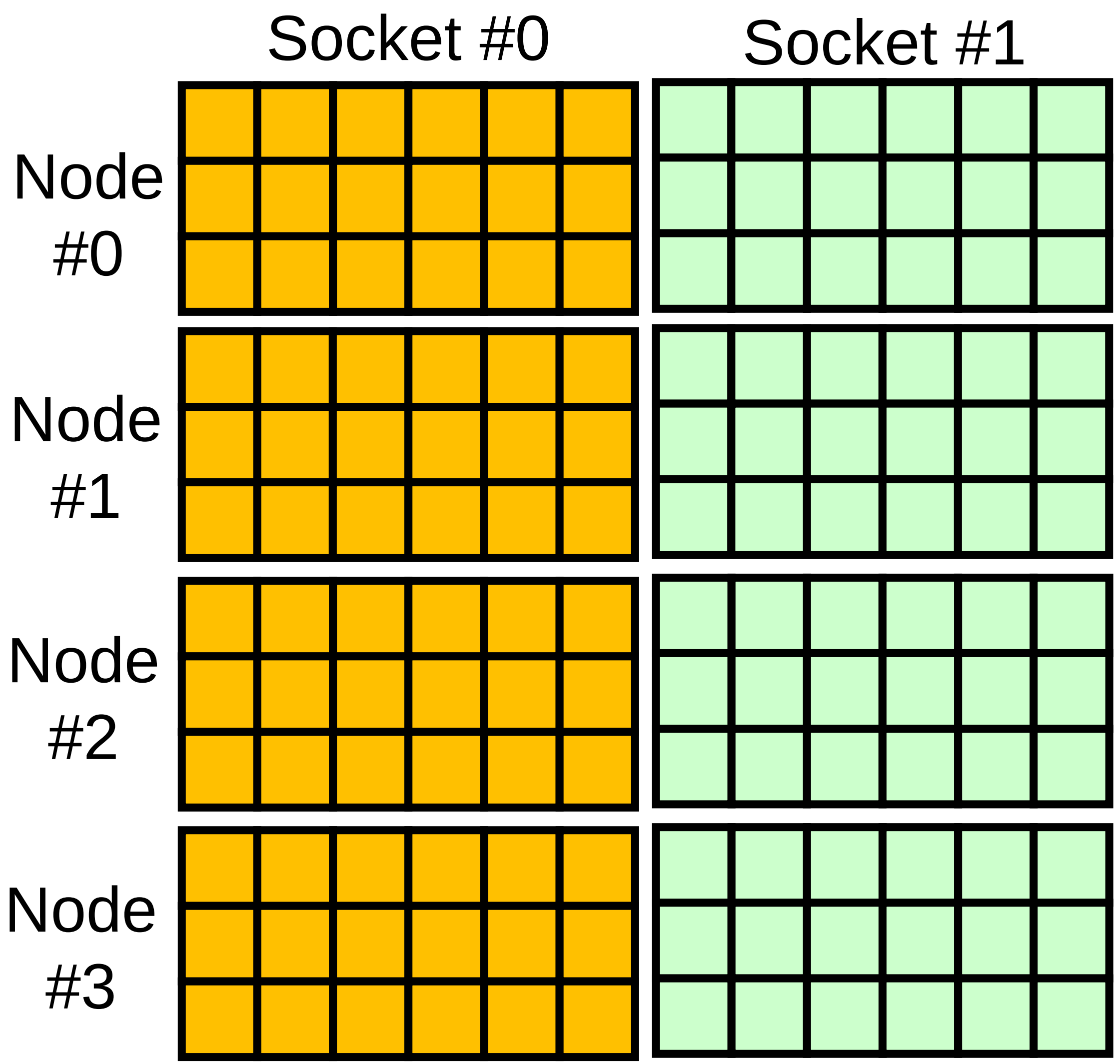
```
inp_mg
256 128  64
```

```
inp_kmetis
cube.0
2
8
pcube
```

```
select=4:
mpiprocs=2

I_MPI_PERHOST=2
OMP_NUM_THREADS=18
```

18/18 cores/socket

4-nodes/8-sockets: 144 MPI process's Flat MPI, 36 MPI process's/Node




```
inp_mg
256 128   64
```

```
inp_kmetis
cube.0
2
144
pcube
```

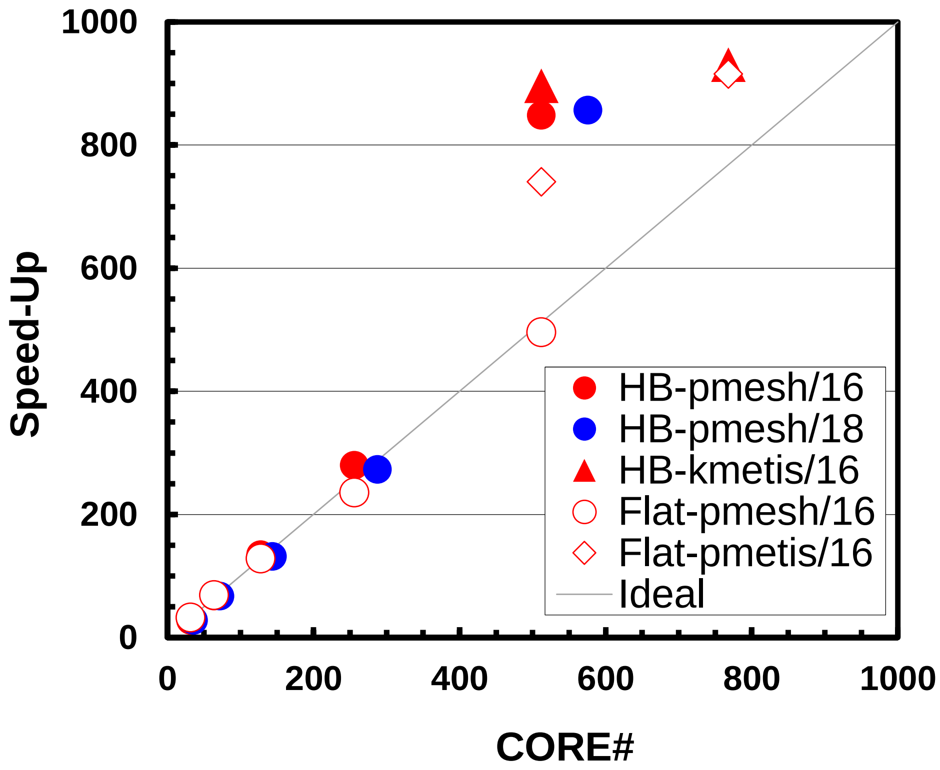
```
select=4:
mpiprocs=36

I_MPI_PERHOST=36
```

18/18 cores/socket

Example: Strong Scaling: Fortran

- 256×128×128 nodes
  - 4,194,304 nodes, 4,112,895 elements
- 32～864 cores, HB 16x1, HB 18x1, Flat MPI
- Linear Solver

Performance of Flat-pmesh/16 w/32 cores= 32.0



HB 16x1, HB 18x1

```
256 128 128
  2   1   1
pcube
```

```
select=1:
mpiprocs=2
```

```
256 128 128
  2   1   2
pcube
```

```
select=2:
mpiprocs=4
```

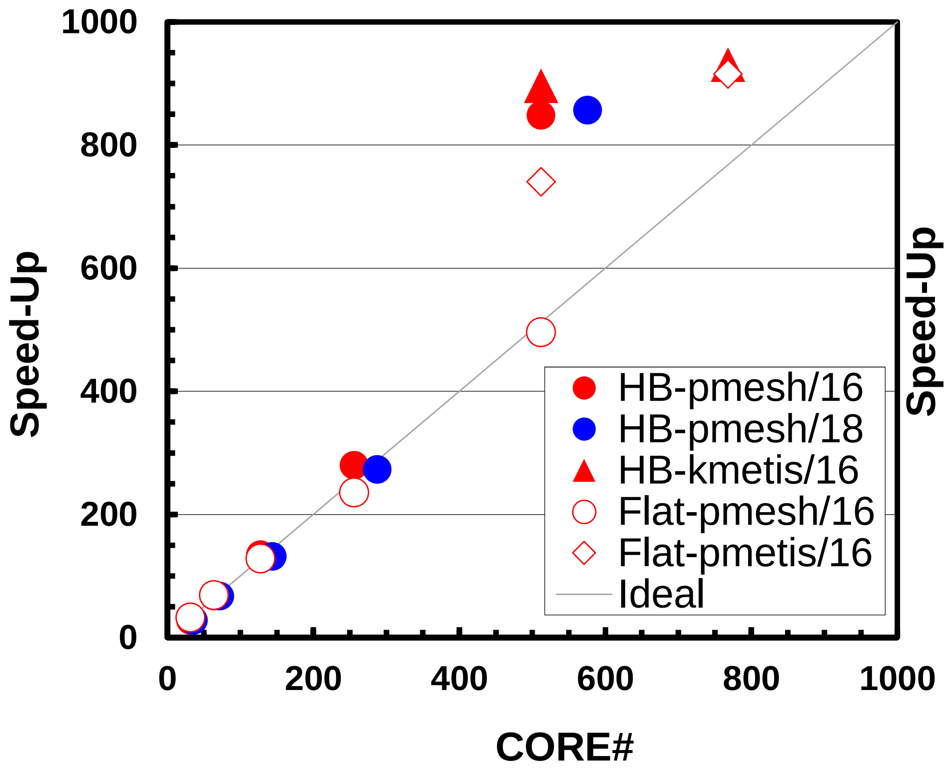
```
256 128 128
  4   2   2
pcube
```

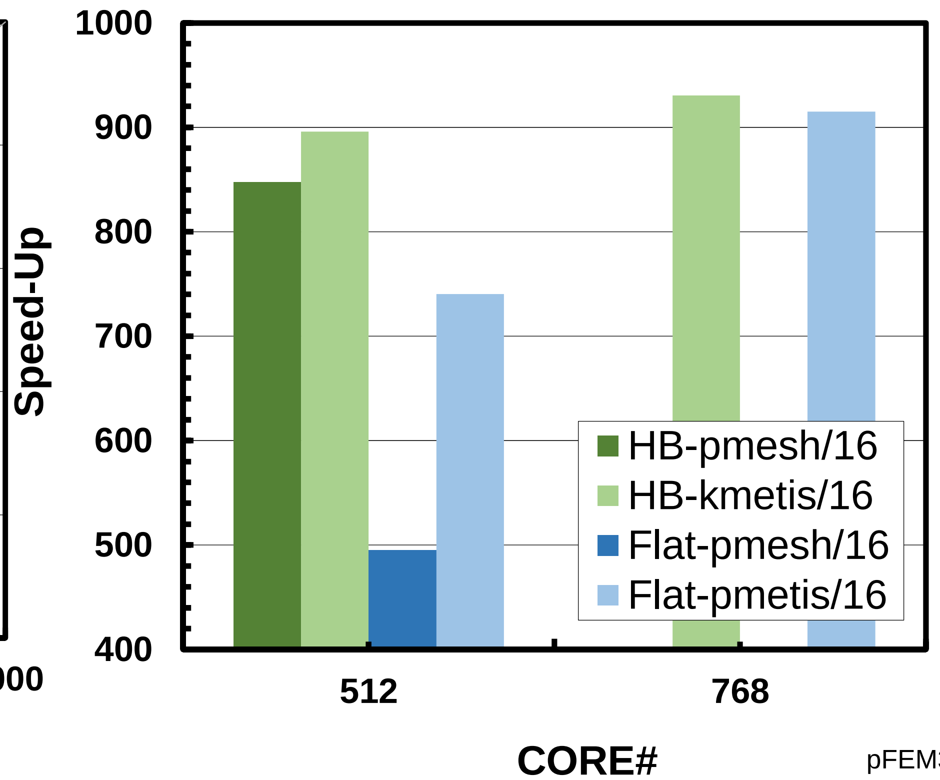
```
select=8:
mpiprocs=16
```

Example: Strong Scaling: Fortran

- 256×128×128 nodes
  - 4,194,304 nodes, 4,112,895 elements
- 32～864 cores, HB 16x1, HB 18x1, Flat MPI
- Linear Solver

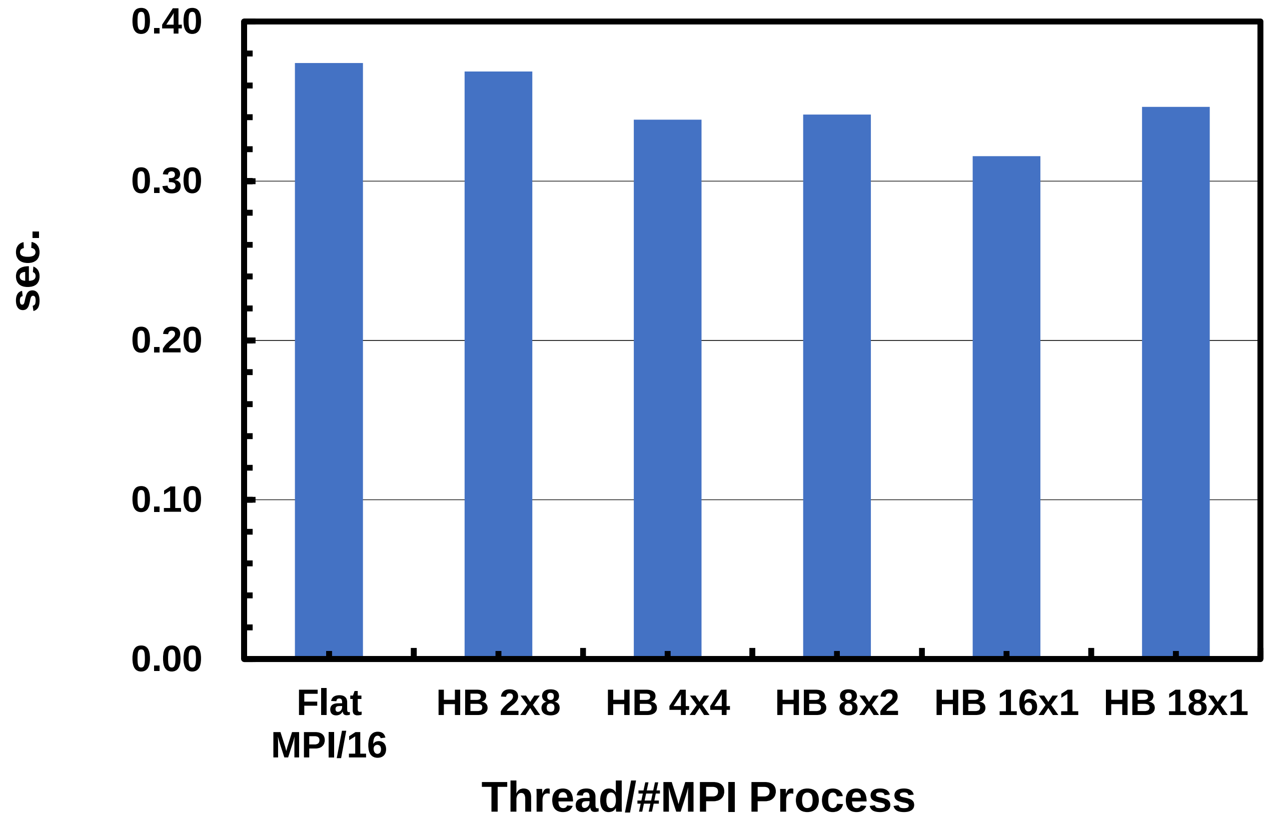
Performance of Flat-pmesh/16 w/32 cores= 32.0





Computation Time using 16 nodes

- kmetis
- Linear Solver



Flat MPI vs. Hybrid

- Depends on applications, problem size, HW etc.
- Flat MPI is generally better for sparse linear solvers, if number of computing nodes is not so large.
  – Memory contention
- Hybrid becomes better, if number of computing nodes is larger.
  – Fewer number of MPI processes.
- 1 MPI Process/Node is possible: NUMA (S1/S2)