9th Advanced School on PARALLEL COMPUTING

Advanced MPI

Giusy Muscianisi – g.muscianisi@cineca.itl
SuperComputing Applications and Innovation Department

CINECA

February 11 - 15, 2013

Outline

- Part 1
  - 5D Network Topology in a Blue Gene/Q
  - Mapping of MPI Tasks on a Blue Gene/Q
  - Blue Gene/Q Personality
- Part 2
  - Consideration of I/O on a Blue Gene/Q
  - I/O performance on FERMI

PART 1:

5D Network Topology in a Blue Gene/Q

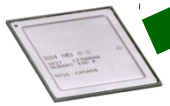
Mapping of MPI Tasks on a Blue Gene/Q

Blue Gene/Q Personality

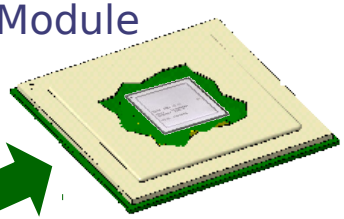
Some questions...

- Network topology on a BG/Q system :
  - How the compute cards are linked togheter?
  - How the midplanes are linked togheter?
- How the MPI Tasks are assigned into the BG/Q processors?
- How can I map my phisical domain into BG/Q hardware, to improve the performance of my application?

1. Chip:
16 cores



2. Single Chip Module



3. Compute card:
One chip module,
16 GB DDR3 Memory,

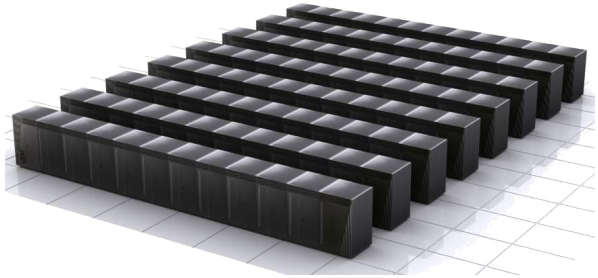
4. Node Card:
32 Compute Cards,
Optical Modules, Link Chips,
Torus

5a. Midplane:
16 Node Cards

5b. IO drawer:
8 IO cards w/16 GB
8 PCIe Gen2 x8 slots

6. Rack: 2 Midplanes

7. System:
20PF/s



A Rack in a BG/Q

1 Rack ==> 2 Midplanes

1 Midplanes ==> 16 Node Boards

1 Node Board ==> 32 Compute Cards

So ….

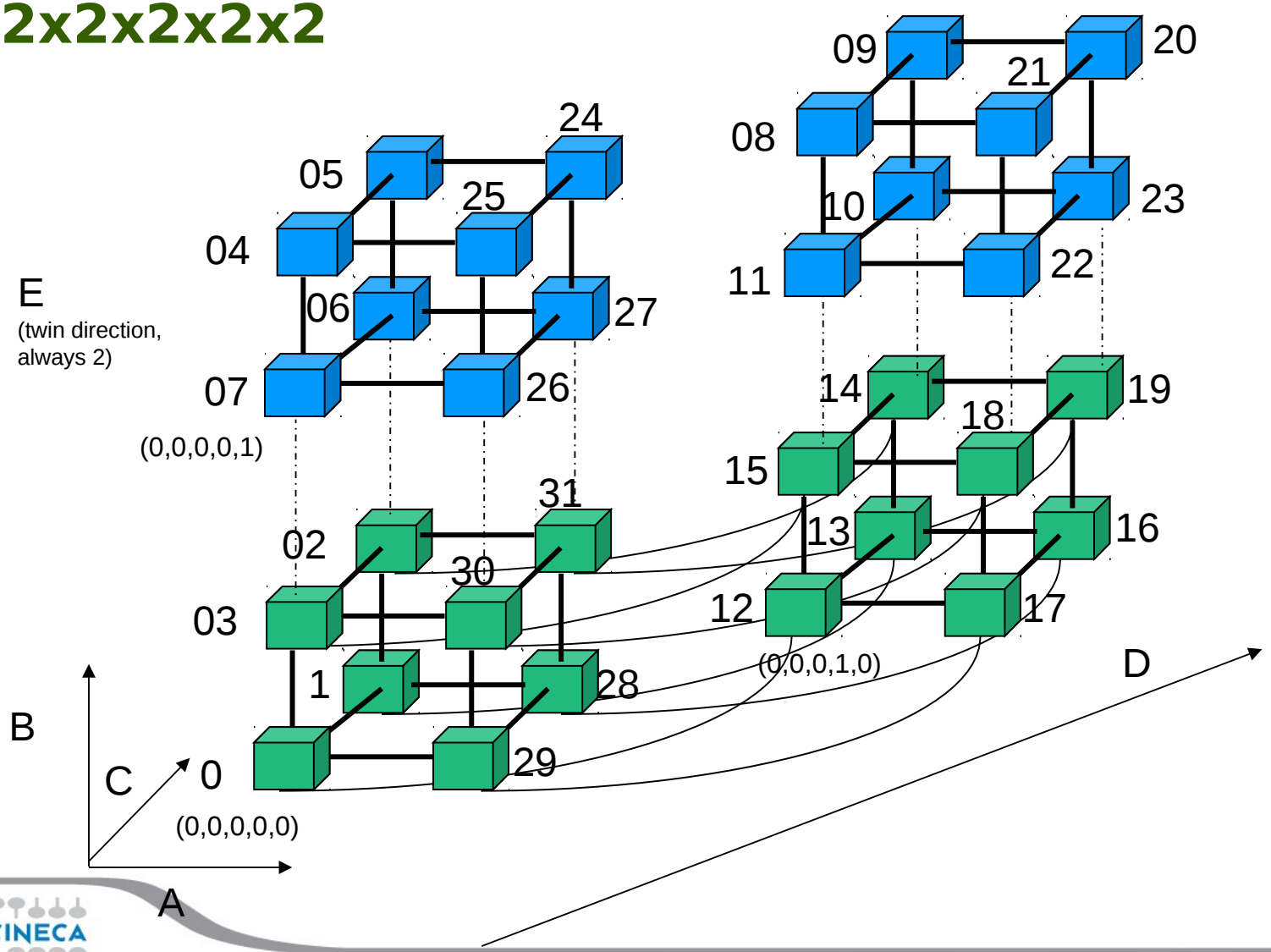
1 Rack ==> 32*16*2 = 1'024 Compute Cards
==> 16'384 Cores

Interconnection: 5D Torus

Compute blocks in a Blue Gene/Q :

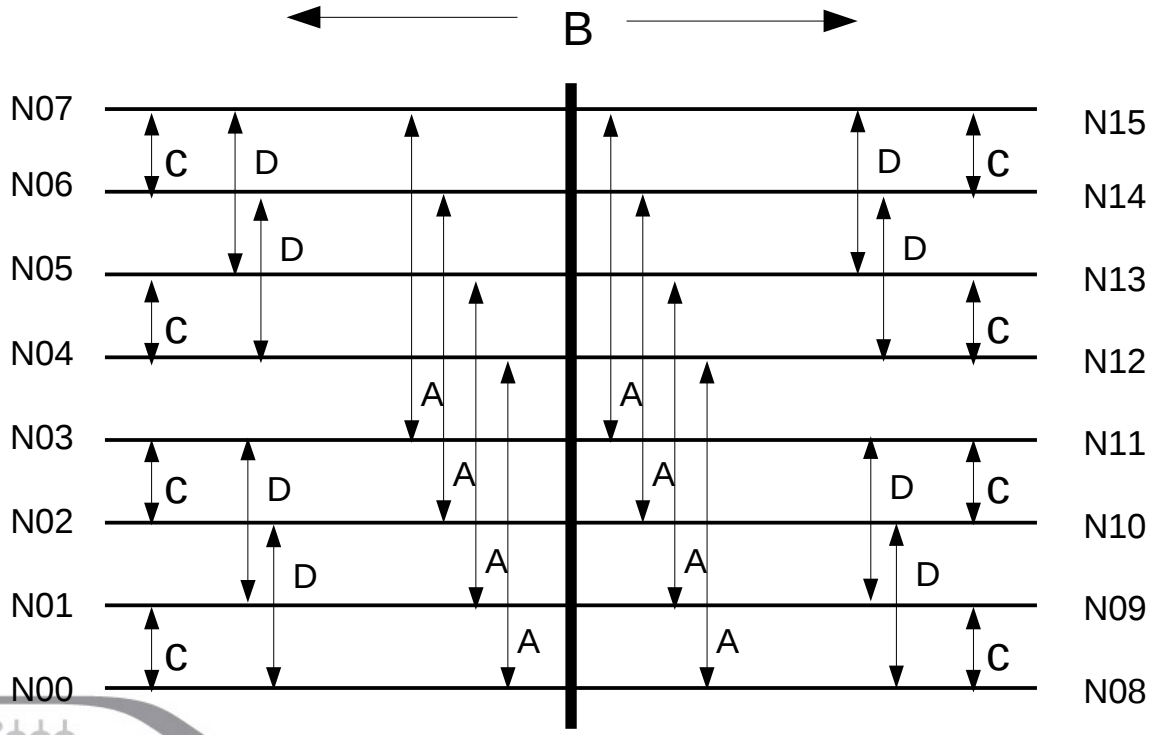
- The network topology for a Blue Gene/Q is a 5D torus or mesh, with direct links between the nearest neighbors in the ±A, ±B, ±C, ±D, and ±E directions.
- Blocks are described by dimensions ABCDET
  - ABCDE are the 5 dimensions of the 5-D torus
    - E is always of dimension 2
    - E is always entirely contained within a midplane
  - T is the extra dimension corresponding to the CPU cores (0-15)
- For any compute block, Compute Nodes (as well midplanes for large blocks) are combined in 4 dimensions
  - Only 4 dimensions need to be considered

Node Board (32 Compute Nodes): 2x2x2x2x2



5-D torus wiring in a Midplane

The 5 dimensions are denoted by the letters A, B, C, D, and E.
The latest dimension E is always 2, and is contained entirely within a midplane.
The nodeboards in a MP combine to form a 4x4x4x4x2 torus (8'192 CCs)



Side view of a midplane:
Each nodeboard is 2x2x2x2x2
Arrows show how dimensions A,B,C,D span across nodeboards
Dimension E does not extend across nodeboards
Note that nodeboards are paired in dimensions A,B,C and D as indicated by the arrows

BG/Q Nodes form a 5D torus

Nodecards: 2x2x2x2x2

Midplanes: 4x4x4x4x2

- 4D are cabled to other midplaness

5th dimension: extent 2 (stays within nodecard)

6th dimension is cpu # within the node

Dim. labels: ABCDE T

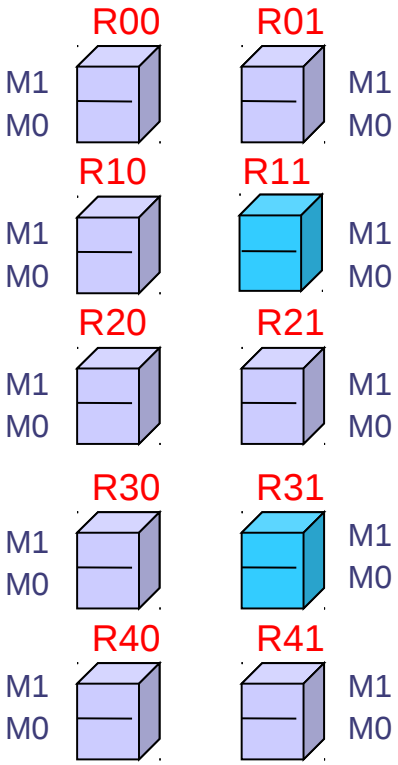
Different floor shapes (Rows x Cols) for a given number of racks may correspond to the same, or to different torus shapes

This list is not complete; other configs are possible...

up to 16x16 = 256 racks

			Torus size, in nodes					Torus size, in midplanes						
Racks	Rows	Col.	A	B	C	D	E	A	B	C	D	Nodes	Cores	Threads
mid	1	1	4	4	4	4	2	1	1	1	1	512	8.192	32.768
1	1	1	4	4	4	8	2	1	1	1	2	1.024	16.384	65.536
2	1	2	4	4	8	8	2	1	1	2	2	2.048	32.768	131.072
3	1	3	4	4	12	8	2	1	1	3	2	3.072	49.152	196.608
4	1	4	8	4	8	8	2	2	1	2	2	4.096	65.536	262.144
4	2	2	4	8	8	8	2	1	2	2	2	4.096	65.536	262.144
6	1	6	12	4	8	8	2	3	1	2	2	6.144	98.304	393.216
6	2	3	4	8	12	8	2	1	2	3	2	6.144	98.304	393.216
6	3	2	4	12	8	8	2	1	3	2	2	6.144	98.304	393.216
8	1	8	8	8	8	8	2	2	2	2	2	8.192	131.072	524.288
8	2	4	8	8	8	8	2	2	2	2	2	8.192	131.072	524.288
8	4	2	8	8	8	8	2	2	2	2	2	8.192	131.072	524.288
10	5	2	4	20	8	8	2	1	5	2	2	10.240	163.840	655.360
12	3	4	8	12	8	8	2	2	3	2	2	12.288	196.608	786.432
12	2	6	12	8	8	8	2	3	2	2	2	12.288	196.608	786.432
16	2	8	16	8	8	8	2	4	2	2	2	16.384	262.144	1.048.576
16	4	4	8	16	8	8	2	2	4	2	2	16.384	262.144	1.048.576
20	5	4	8	20	8	8	2	2	5	2	2	20.480	327.680	1.310.720
24	6	4	8	12	16	8	2	2	3	4	2	24.576	393.216	1.572.864
24	2	12	8	8	12	16	2	2	2	3	4	24.576	393.216	1.572.864
28	7	4	8	28	8	8	2	2	7	2	2	28.672	458.752	1.835.008
32	8	4	8	16	16	8	2	2	4	4	2	32.768	524.288	2.097.152
32	4	8	8	16	16	8	2	2	4	4	2	32.768	524.288	2.097.152
40	5	8	8	20	16	8	2	2	5	4	2	40.960	655.360	2.621.440
48	6	8	16	12	16	8	2	4	3	4	2	49.152	786.432	3.145.728
48	4	12	8	16	12	16	2	2	4	3	4	49.152	786.432	3.145.728
48	3	16	8	12	16	16	2	2	3	4	4	49.152	786.432	3.145.728
56	7	8	8	28	16	8	2	2	7	4	2	57.344	917.504	3.670.016
64	8	8	8	16	16	16	2	2	4	4	4	65.536	1.048.576	4.194.304
64	4	16	8	16	16	16	2	2	4	4	4	65.536	1.048.576	4.194.304
72	9	8	12	12	16	16	2	3	3	4	4	73.728	1.179.648	4.718.592
80	10	8	8	20	16	16	2	2	5	4	4	81.920	1.310.720	5.242.880
96	12	8	16	12	16	16	2	4	3	4	4	98.304	1.572.864	6.291.456
96	8	12	16	16	12	16	2	4	4	3	4	98.304	1.572.864	6.291.456
96	6	16	16	12	16	16	2	4	3	4	4	98.304	1.572.864	6.291.456

FERMI Configuration

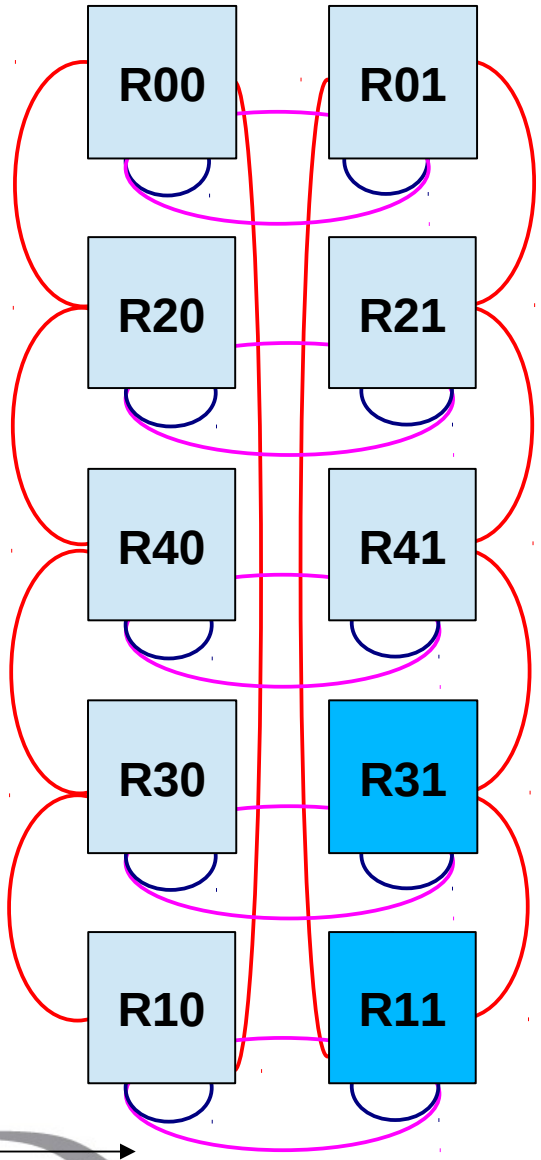


N° of MPs 20
N° of Compute Nodes 10'240

FERMI Size in Midplanes (Shape) 1x5x2x2
Compute Nodes in a Midplane 4x4x4x4x2

FERMI Size in Compute Nodes 4x20x8x8x2

Rack with 8 IO Nodes
Rack with 16 IO Nodes



Midplanes CABLING

B dimension

- connection among 2 midplains go down a column of racks

C dimension

- connection among 2 midplains go down a row of racks

D dimension

- connection among 2 midplains in the same rack

A dimension

- the remaining direction, which can go down a row or column (or both). When two sets of cables go down a row or column, the longest cables define the A dimension

Racks	MP	Row	Col	A	B	C	D
10	20	5	2	1	5	2	2

Partitions

Large blocks

- Partitions occupy one or more complete midplanes
- Always a multiple of 512 Compute Nodes
- Can be a torus in all 5 dimensions

Small blocks

- Partitions occupy less than a midplane
- Span on one or more node cards within a midplane
  - Strictly less than 512 Compute Nodes in total
- Always a multiple of 32 Compute Nodes: 32, 64, 128, 256
- Cannot be a torus in all 5 dimensions

Torus .vs. Mesh

- Torus: periodic boundary conditions (e.g. “close line”) in all the dimensions A, B, C, D and E.
- Mesh: almost one dimension is not like a “close line”

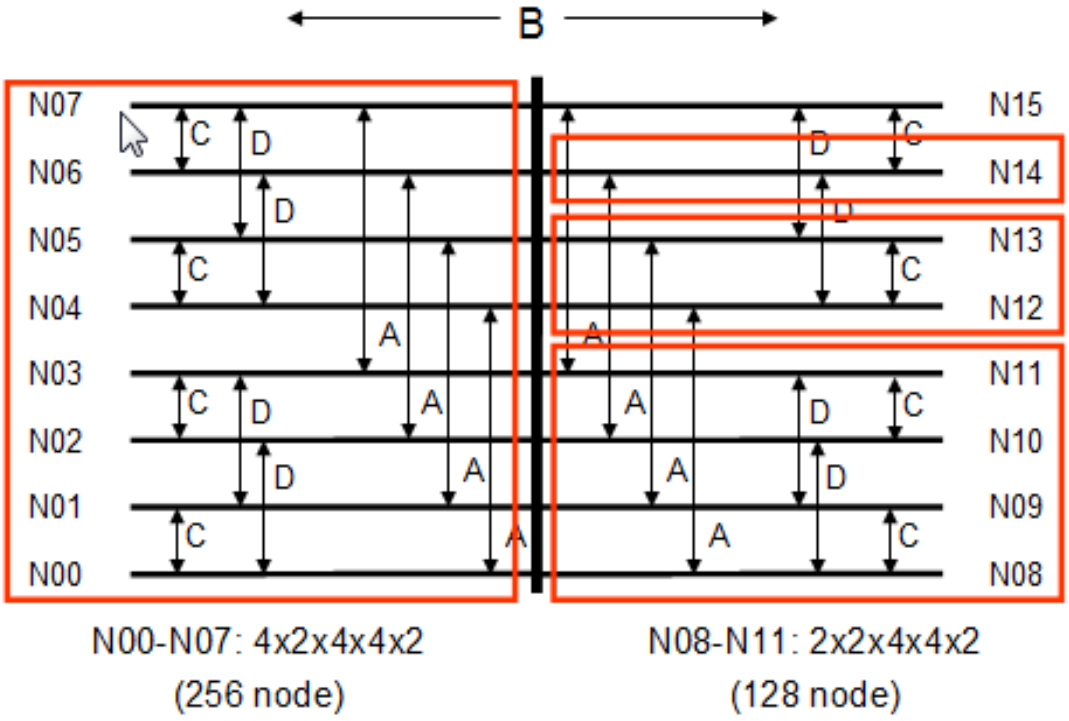
Small block

Torus vs. mesh on a per-dimension basis

- Partial torus for sub-midplane blocks
- Partial torus for multi-midplane blocks

# Node Boards	# Nodes	Torus Size, in Nodes	Is Torus (ABCDE)
1	32	2x2x2x2x2	00001
2 (adjacent pairs)	64	2x2x4x2x2	00101
4 (quadrants)	128	2x2x4x4x2	00111
8 (halves)	256	4x2x4x4x2	10111

Small block



N14: 2x2x2x2x2 (32 node)

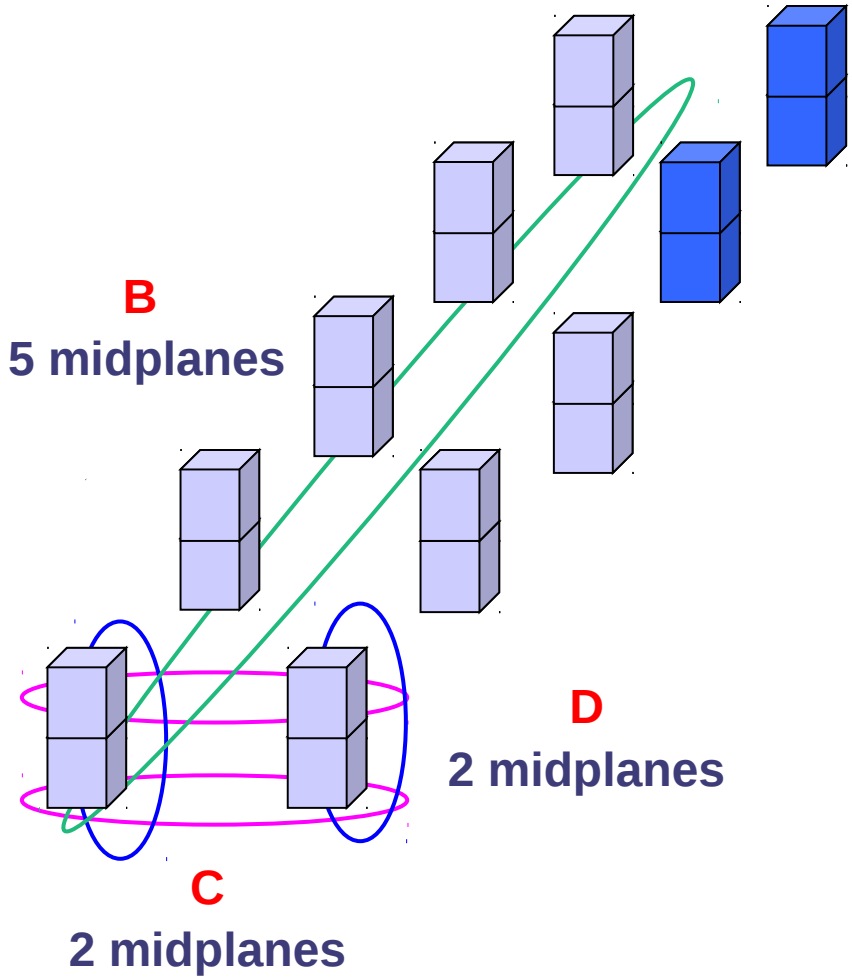
N12-N13: 2x2x4x2x2 (64 node)

N00-N07: 4x2x4x4x2 (256 node)

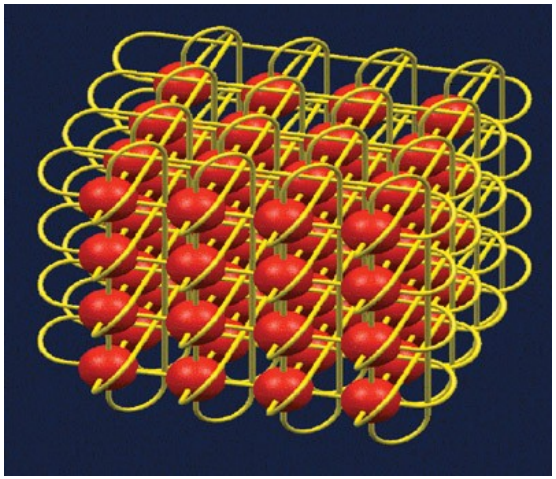
N08-N11: 2x2x4x4x2 (128 node)

- Partitioning a midplane is the same concept as in BG/P
- 5th dimension always wrapped
- Other dimensions wrapped whenever they are 4 nodes wide

TORUS

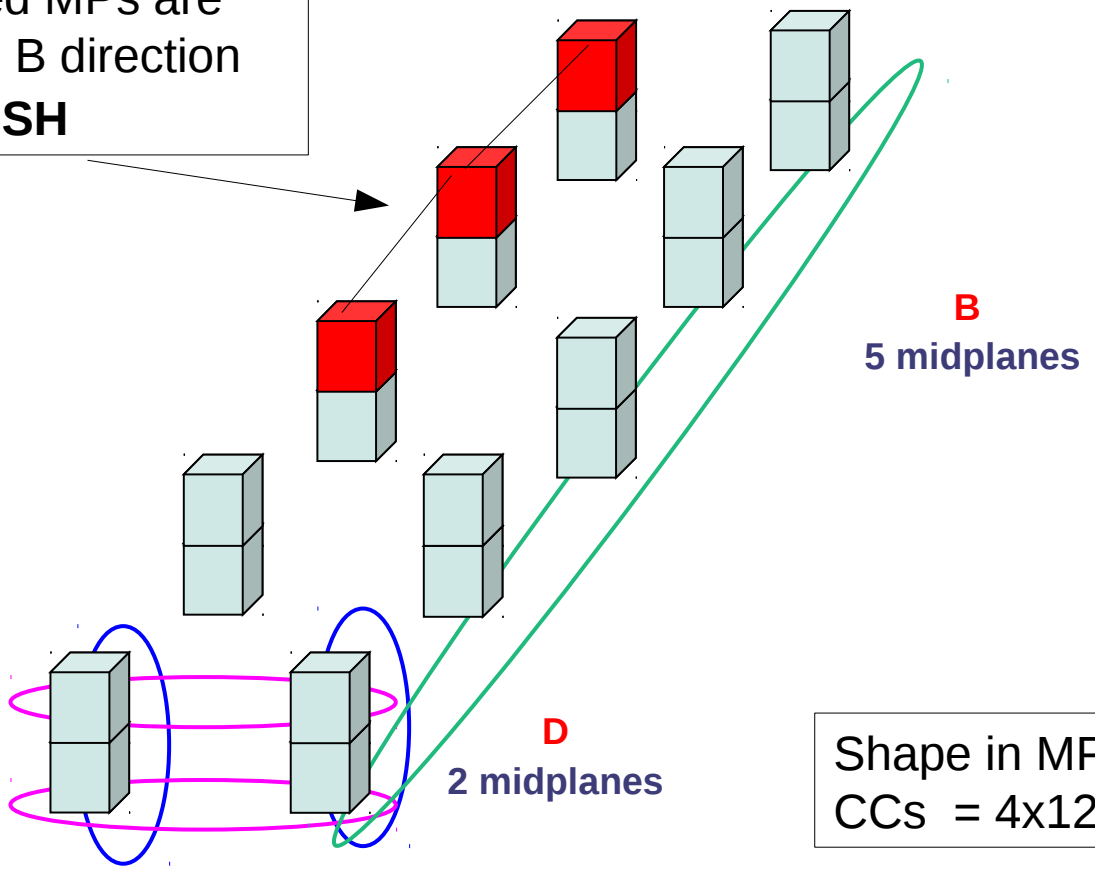


1 Midplane is the minum TORUS available on a BlueGene/Q system: 4x4x4x4x2



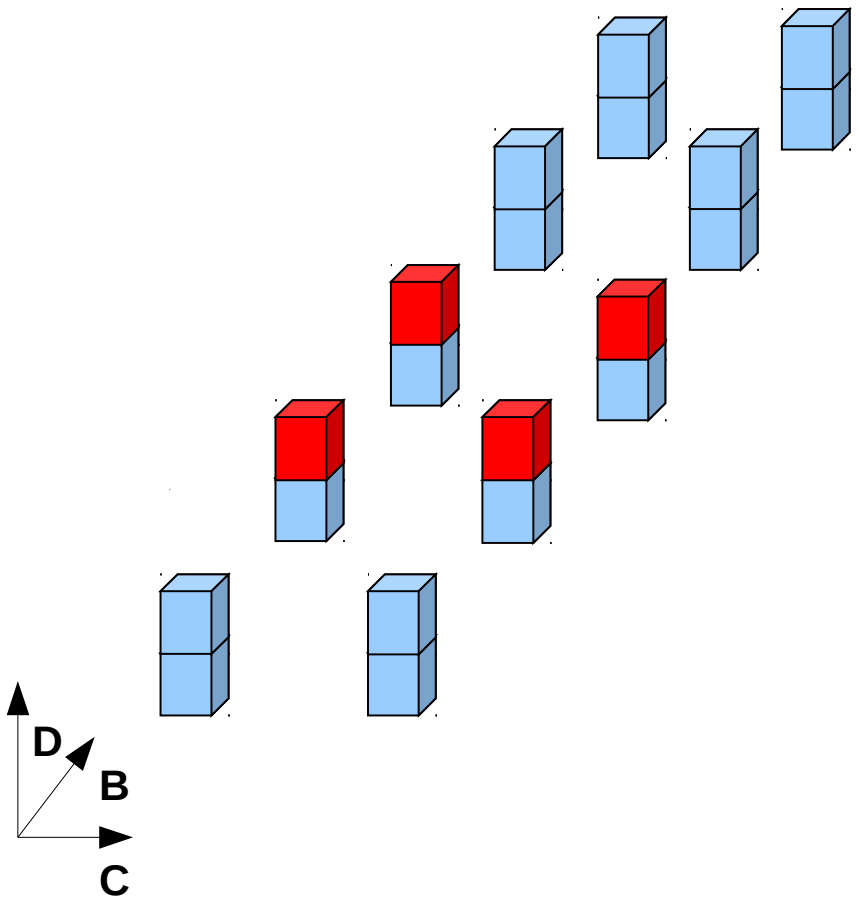
TORUS vs. MESH

The 3 red MPs are linked in B direction as a **MESH**

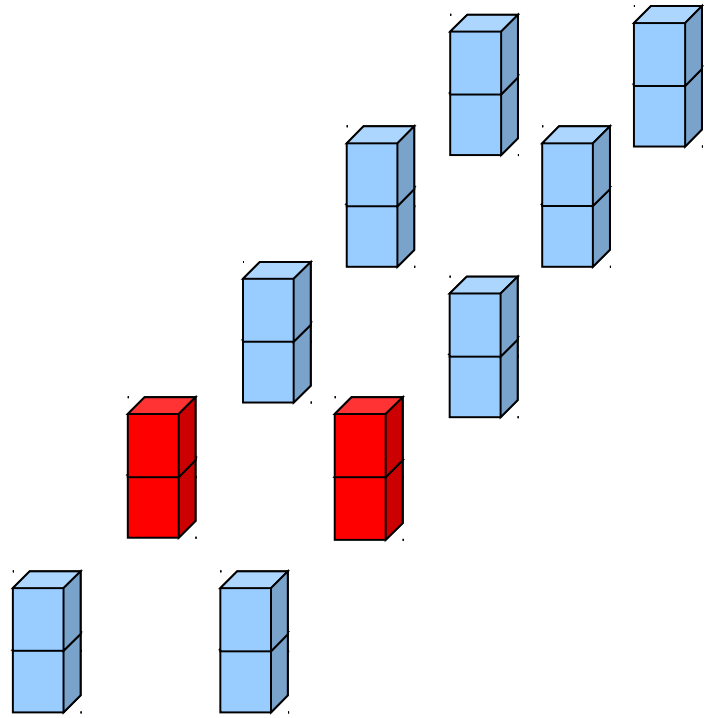


Shape in MPs = 1x3x1x1
CCs = 4x12x4x4x2

TORUS .vs. MESH



A	B	C	D
Torus	Mesh	Torus	Torus



A	B	C	D
Torus	Torus	Torus	Torus

PART 1:

5D Network Topology in a Blue Gene/Q

Mapping of MPI Tasks on a Blue Gene/Q

Blue Gene/Q Personality

Mapping overview

- **Mapping, definition:**
  - assignment of processes (as an MPI task) to Blue Gene processors.

Mapping overview

- **Why to take care of mapping.....**
  - When communication involves the nearest neighbors on the torus network, a **large fraction of the theoretical peak bandwitdh can be obtained**.

    In a number of cases, it is possible to control the placement of MPI ranks so that communication remains local. This placement can significantly improve scaling for a number of applications, particularly at large processor counts.

  - When MPI ranks communicate with many hops between the neighbors, **the effective bandwidth is reduced** by a factor that is equal to the average number of hops that messages take on the torus network.

Mapping overview

- **When mapping is important:**
  - **MAY BE NOT IMPORTANT:**

    for jobs that use 1 MP (512 nodes) or less of the BG/Q system due to the compact shape, <A,B,C,D,E> = <4,4,4,4,2> for a MP, and the high degree of connectivity

  - **CAN BE USEFUL:**

    for applications that have a
    - **regular Cartesian topology**,
    - are dominated by **nearest-neighbor boundary exchange**, particularly for large configurations.

Default mapping

The MPI ranks are placed on the system in **ABCDET** order where :

- the rightmost letter increments first
- <A,B,C,D,E> are torus coordinates
- T is the processor ID in each node (T = 0 to N -1, where N is the number of processes per node being used)

Example: default mapping

bg_size=128; --np 512 --ranks-per-node 4
block shape: 2x2x4x4x2; torus: 00111

```
MPI rank   0 of 512, coordinates: (0,0,0,0,0,0)
MPI rank   1 of 512, coordinates: (0,0,0,0,0,1)
MPI rank   2 of 512, coordinates: (0,0,0,0,0,2)
MPI rank   3 of 512, coordinates: (0,0,0,0,0,3)
MPI rank   4 of 512, coordinates: (0,0,0,0,1,0)
MPI rank   5 of 512, coordinates: (0,0,0,0,1,1)
MPI rank   6 of 512, coordinates: (0,0,0,0,1,2)
MPI rank   7 of 512, coordinates: (0,0,0,0,1,3)
MPI rank   8 of 512, coordinates: (0,0,0,1,0,0)
MPI rank   9 of 512, coordinates: (0,0,0,1,0,1)
MPI rank  10 of 512, coordinates: (0,0,0,1,0,2)
MPI rank  11 of 512, coordinates: (0,0,0,1,0,3)
… … …
MPI rank 506 of 512, coordinates: (1,1,3,3,0,2)
MPI rank 507 of 512, coordinates: (1,1,3,3,0,3)
MPI rank 508 of 512, coordinates: (1,1,3,3,1,0)
MPI rank 509 of 512, coordinates: (1,1,3,3,1,1)
MPI rank 510 of 512, coordinates: (1,1,3,3,1,2)
MPI rank 511 of 512, coordinates: (1,1,3,3,1,3)
```

How manage the mapping

- The mapping can be controlled:
  - by passing arguments to the runjob command
    - **`runjob --mapping TEDCBA`**
    - **`runjob --mapping my.map`**
  - by constructing a specially ordered communicator in the application

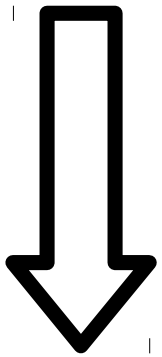
Example: TEDCBA mapping

bg_size=128; --np 512 --ranks-per-node 4
block shape: 2x2x4x4x2; torus: 00111

```
MPI rank   0 of 512, coordinates: (0,0,0,0,0,0)
MPI rank   1 of 512, coordinates: (1,0,0,0,0,0)
MPI rank   2 of 512, coordinates: (0,1,0,0,0,0)
MPI rank   3 of 512, coordinates: (1,1,0,0,0,0)
MPI rank   4 of 512, coordinates: (0,0,0,0,0,0)
MPI rank   5 of 512, coordinates: (0,0,1,0,0,0)
MPI rank   6 of 512, coordinates: (1,0,1,0,0,0)
MPI rank   7 of 512, coordinates: (0,1,1,0,0,0)
MPI rank   8 of 512, coordinates: (1,1,1,0,0,0)
MPI rank   9 of 512, coordinates: (0,0,2,0,0,0)
MPI rank  10 of 512, coordinates: (1,0,2,0,0,0)
MPI rank  11 of 512, coordinates: (0,1,2,0,0,0)
… … …
MPI rank 506 of 512, coordinates: (0,1,2,3,1,3)
MPI rank 507 of 512, coordinates: (1,1,2,3,1,3)
MPI rank 508 of 512, coordinates: (0,0,3,3,1,3)
MPI rank 509 of 512, coordinates: (1,0,3,3,1,3)
MPI rank 510 of 512, coordinates: (0,1,3,3,1,3)
MPI rank 511 of 512, coordinates: (1,1,3,3,1,3)
```

Mapping: general guidance

For application that use a **regular Cartesian topology**



to map MPI ranks onto the BG/Q torus network in a way that preserves locality for nearest-neighbor communication

Mapping: general guidance

1 DIM

- each MPI rank communicates with its rank ±1,
- the default ABCDET mapping can be good for blocks large enough to use torus wrap-around
  - “T” coordinate increments first
  - E-dimension is always torus-enabled
  - other dimensions are torus is multiple of 4
- only one extra hop at the torus edges
  - to eliminate the extra hop, use a **snake-like pattern** through the torus by reversing direction

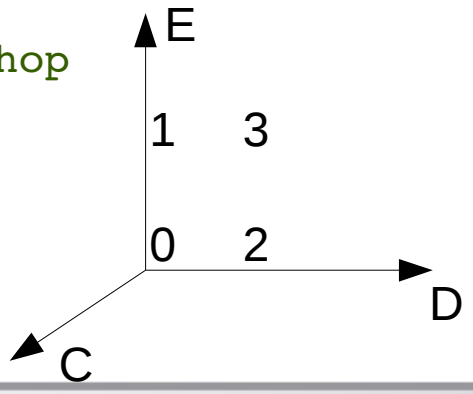
Example of 1D topology

0	1	2	3	4	5	… … … …	n-2	n-1

bg_size=64
shape=2x2x4x2x2
ranks-per-node=16
MPI tasks=1024

```
                                A B C D E
0 | MPI ranks 0-15              (0,0,0,0,0)
1 | MPI ranks 16-31             (0,0,0,0,1)
2 | MPI ranks 32-47             (0,0,0,1,0)   extra hop
3 | MPI ranks 48-63             (0,0,0,1,1)
4 | MPI ranks 64-79             (0,0,1,0,0)   extra hops
5 | MPI ranks 80-95             (0,0,1,0,1)
6 | MPI ranks 96-111            (0,0,1,1,0)   extra hop
7 | MPI ranks 112-127           (0,0,1,1,1)
8 | MPI ranks 128-143           (0,0,2,0,0)   extra hops
9 | MPI ranks 144-159           (0,0,2,0,1)
10| MPI ranks 160-175           (0,0,2,1,0)   extra hop
11| MPI ranks 176-191           (0,0,2,1,1)

and so on...
```



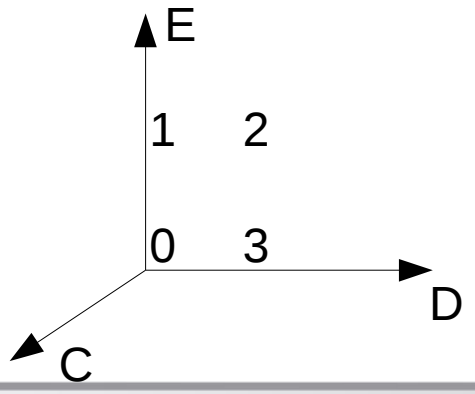
Example of 1D topology

0	1	2	3	4	5	… … … …	n-2	n-1

bg_size=64
shape=2x2x4x2x2
ranks-per-node=16
MPI tasks=1024

```
                                A B C D E
0   |MPI ranks 0-15             (0,0,0,0,0)
1   |MPI ranks 16-31            (0,0,0,0,1)
2   |MPI ranks 32-47            (0,0,0,1,1)
3   |MPI ranks 48-63            (0,0,0,1,0)
4   |MPI ranks 64-79            (0,0,1,1,0)
5   |MPI ranks 80-95            (0,0,1,1,1)
6   |MPI ranks 96-111           (0,0,1,0,1)
7   |MPI ranks 112-127          (0,0,1,0,0)
8   |MPI ranks 128-143          (0,0,2,0,0)
9   |MPI ranks 144-159          (0,0,2,0,1)
10  |MPI ranks 160-175          (0,0,2,1,1)
11  |MPI ranks 176-191          (0,0,2,1,0)
and so on...
```

Snake-like pattern...
to avoid the hops



Mapping: general guidance

2 DIM

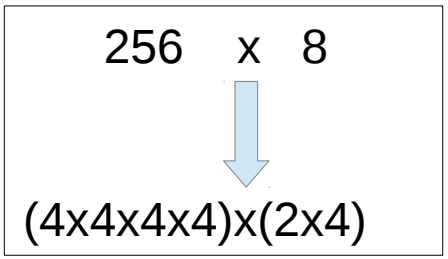
- **Example**: 1 MP , 16 ranks-per-node = <4,4,4,4,2,16> = 8192 tasks
  - When two or three of these dimensions are grouped together, this ordering can be naturally a good fit for
    - 256x32 & 64x128
    - Instead, 128x64 fit better with TEDCBA
  - these map have one extra hop at torus boundaries. Avoid it using a customized map file:
    - With 16 MPI ranks-per-node, arrange the 16 procs in a logical 4x4 box ==> most of the boundary exchange can be kept inside each node.
  - group 2 or 3 of the torus dimensions into logical X or Y dimension
  - grouping creates a snake-like pattern

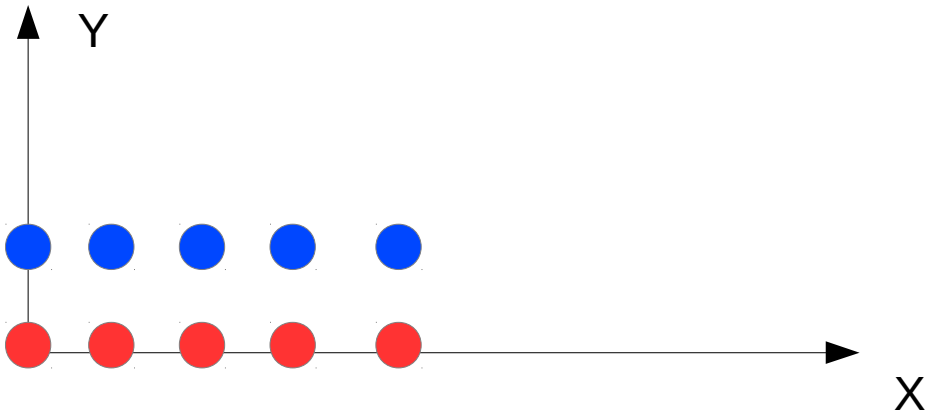
Example of 2D

```
# A B C D E T #2d : <X, Y>
  0 0 0 0 0 0 #2d : <0, 0>
  0 0 0 0 0 1 #2d : <0, 1>
  0 0 0 0 0 2 #2d : <0, 2>
  0 0 0 0 0 3 #2d : <0, 3>
  0 0 0 0 1 0 #2d : <0, 4>
  0 0 0 0 1 1 #2d : <0, 5>
  0 0 0 0 1 2 #2d : <0, 6>
  0 0 0 0 1 3 #2d : <0, 7>

  0 0 0 1 0 0 #2d : <1, 0>
  0 0 0 1 0 1 #2d : <1, 1>
  0 0 0 1 0 2 #2d : <1, 2>
  0 0 0 1 0 3 #2d : <1, 3>
  0 0 0 1 1 0 #2d : <1, 4>
  0 0 0 1 1 1 #2d : <1, 5>
  0 0 0 1 1 2 #2d : <1, 6>
  0 0 0 1 1 3 #2d : <1, 7>

  0 0 0 2 0 0 #2d : <2, 0>
  0 0 0 2 0 1 #2d : <2, 1>
  0 0 0 2 0 2 #2d : <2, 2>
  0 0 0 2 0 3 #2d : <2, 3>
  0 0 0 2 1 0 #2d : <2, 4>
  0 0 0 2 1 1 #2d : <2, 5>
  0 0 0 2 1 2 #2d : <2, 6>
  0 0 0 2 1 3 #2d : <2, 7>
  ......
```





Mapping: general guidance

2 DIM

Global approach, processes in the node are arranged in 2D.

- group 2 or 3 dimensions into logical X and Y dimensions.
- grouping creates a snake-like pattern
- ensure that shared-memory is used for much of the communication, and that the maximum number of hops on the torus network is one

Use such approach to create map file or to reorder ranks in a new communicator

Mapping: general guidance

3 DIM

- The default mapping can be effective in a number of cases
  - 1 midplane, 16 ranks-per-node = <4,4,4,4,2,16> = 8192 tasks
    - When one or more dimensions are grouped together, the default mapping can be a good match for 3D domain such as 16x32x16, and 16x16x32, where the last dimension increments first

Mapping: general guidance

4 DIM

- Quantum chromo dynamics (QCD) applications typically use a 4D process topology.
- This topology can often fit perfectly onto the BG/Q system. However, a map file or an ordered communicator might be required.

Mapping: general guidance

4 DIM – Example

- 1MP, 32 ranks per node : ABCDET dims are <4,4,4,4,2,32>=16384 tasks.
- If you want to use a roughly balanced 4D decomposition, such as (X,Y,Z,T)=(8,8,16,16), there is no natural grouping of ABCDET that fit.
- Possible solution :
  - to consider the 32 MPI ranks within each node as being arranged in a 4D box : bx=2 ; by=2 ; bz=4 ; bt=2.
  - logical 4D mapping can be (bx*A, by*B, bz*C, bt*DE), where DE is treated as a 1D line that snakes throught the DE plane to avoid the extra hop at torus boundary.
  - Other 4D decompositions that map naturally with ABCDET mappig are: (16,16,2,32) ; (4,16,8,32) ; and permutations.

Unstructured grid

- The default ordering is frequently a good choice
- This appens because neighboring cells tend to end up in the same part of the machine, and ABCDET order keeps many contiguous ranks in close proximity.
- MPI performance tools can be used to check for locality, but optimization of the layout in the general case of unstructured grids is a challenging problem

Mapping: summary

- Application that use a regular Cartesian topology might benefit from a careful mapping of processes onto processors.
- Default mapping is frequently a good choice.
- It might be possible to optimize performance by using a map file passed to the runjob command .
- It might be also possible to optimize performance by constructiong a specially ordered communicator for use within the application.

PART 1:

5D Network Topology in a Blue Gene/Q

Mapping of MPI Tasks on a Blue Gene/Q

Blue Gene/Q Personality

Blue Gene/Q Personality

- **Definition:**
  - Static data given to every compute node and IO node at boot time by the Control System. This data contains information that is specific to the node, with respect to the block that is being booted.
  - Set of C language structures that contains such items as the node coordinates on the torus network.

- **This informations can be useful:**
  - If the programmer wants to determine, at run time, where the tasks of the application are running
  - To tune certain aspects of the application at run time (as optimize the network traffic from the compute nodes to the IO node, for tasks sharing the same IO node

Personality : usage

- Include file
  - #include <spi/include/kernel/location.h>
- Structure
  - Personality_t pers
  - Query function
    - Kernel_GetPersonality(&pers, sizeof(pers));
  - Properties
    - pers.Network_Config.[A-E]nodes : Nb nodes in each torus dimension
    - pers.Network_Config.[A-E]coord : Coordinates of the nodes in the torus
    - pers.Network_Config.[A-E]bridge : Coordinates of the IO bridges in the torus
- Other routines
  - Kernel_ProcessorID() : Processor ID (0-63)
  - Kernel_ProcessorCoreID() : Processor core ID (0-15)
  - Kernel_ProcessorThreadID() : Processor thread ID (0-3)

Personality: output example 1

128-node block, 2 ranks per node, 256 total MPI tasks, mapping ABCDET

```
block shape : <2,2,4,4,2>
torus links enabled : <0,0,1,1,1>

rank 0 has processor name Task 0 of 256 (0,0,0,0,0,0) R02-M1-N00-J00
rank 0 location <0,0,0,0,0> core 0 hwthread 0 procid = 0

rank 1 has processor name Task 1 of 256 (0,0,0,0,0,1) R02-M1-N00-J00
rank 1 location <0,0,0,0,0> core 8 hwthread 0 procid = 32

rank 2 has processor name Task 2 of 256 (0,0,0,0,1,0) R02-M1-N00-J07
rank 2 location <0,0,0,0,1> core 0 hwthread 0 procid = 0

rank 3 has processor name Task 3 of 256 (0,0,0,0,1,1) R02-M1-N00-J07
rank 3 location <0,0,0,0,1> core 8 hwthread 0 procid = 32

rank 4 has processor name Task 4 of 256 (0,0,0,1,0,0) R02-M1-N00-J12
rank 4 location <0,0,0,1,0> core 0 hwthread 0 procid = 0
```

Personality: output example 2

128-node block, 2 ranks per node, 256 total MPI tasks, mapping TEDCBA

```
block shape : <2,2,4,4,2>
torus links enabled : <0,0,1,1,1>

rank 0 has processor name Task 0 of 256 (0,0,0,0,0,0) R02-M1-N00-J00
rank 0 location <0,0,0,0,0> core 0 hwthread 0 procid = 0

rank 1 has processor name Task 1 of 256 (1,0,0,0,0,0) R02-M1-N00-J29
rank 1 location <1,0,0,0,0> core 0 hwthread 0 procid = 0

rank 2 has processor name Task 2 of 256 (0,1,0,0,0,0) R02-M1-N00-J03
rank 2 location <0,1,0,0,0> core 0 hwthread 0 procid = 0

rank 3 has processor name Task 3 of 256 (1,1,0,0,0,0) R02-M1-N00-J30
rank 3 location <1,1,0,0,0> core 0 hwthread 0 procid = 0

rank 4 has processor name Task 4 of 256 (0,0,1,0,0,0) R02-M1-N00-J01
rank 4 location <0,0,1,0,0> core 0 hwthread 0 procid = 0
…
```

PART 2:

Consideration of I/O on a Blue Gene/Q

I/O performance on FERMI

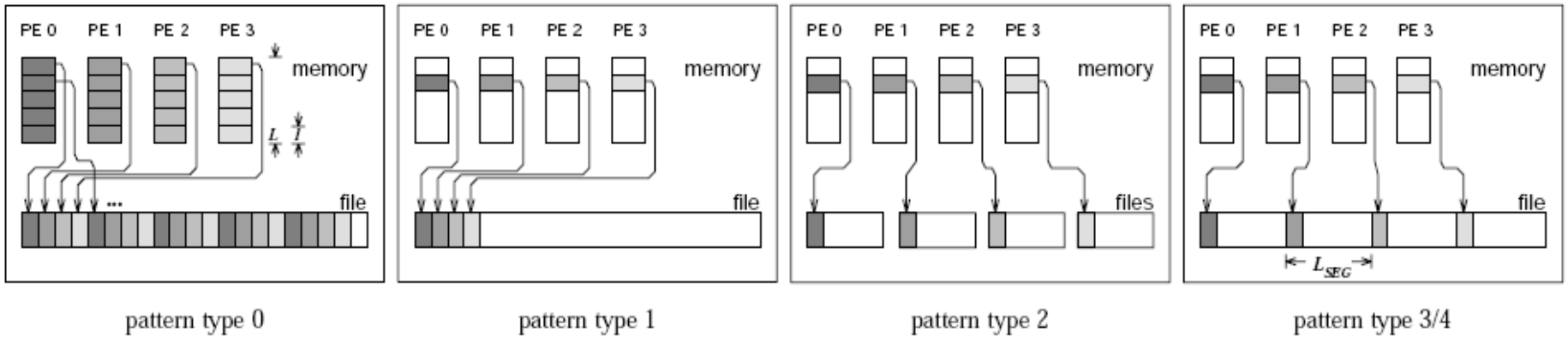
b_eff_io benchmark

- **Effective I/O bandwidth benchmark :**
- The effective I/O bandwidth benchmark (b_eff_io) covers two goals: (1) to achieve a characteristic average number for the I/O bandwidth achievable with parallel MPI-I/O applications, and

  (2) to get detailed information about several access patterns and buffer lengths.

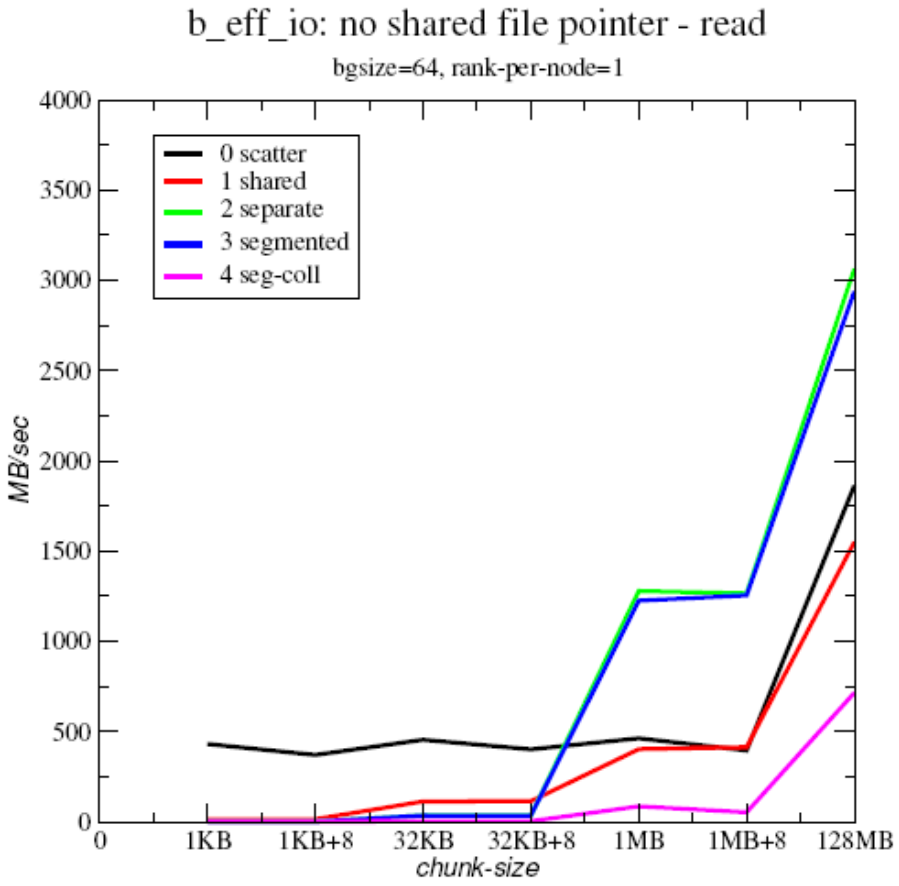
  The benchmark examines "first write", "rewrite" and "read" access, strided (individual and shared pointers) and segmented collective patterns on one file per application and non-collective access to one file per process. The number of parallel accessing processes is also varied and wellformed I/O is compared with non-wellformed.

b_eff_io benchmark

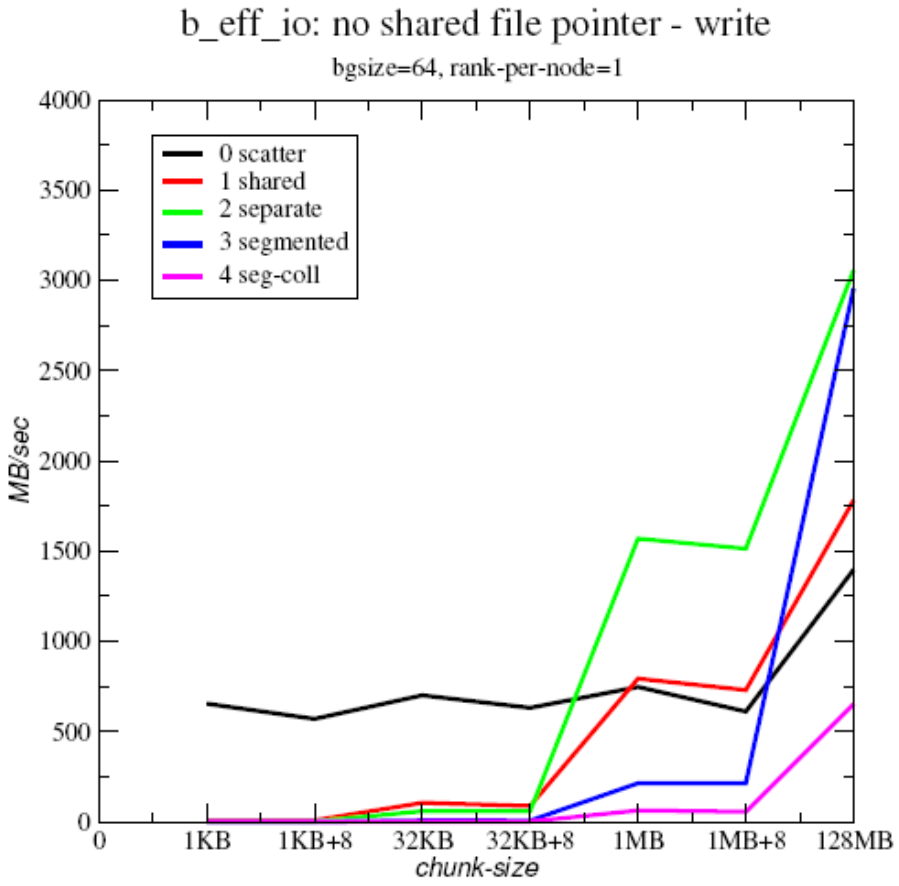


pattern type 0 | pattern type 1 | pattern type 2 | pattern type 3/4

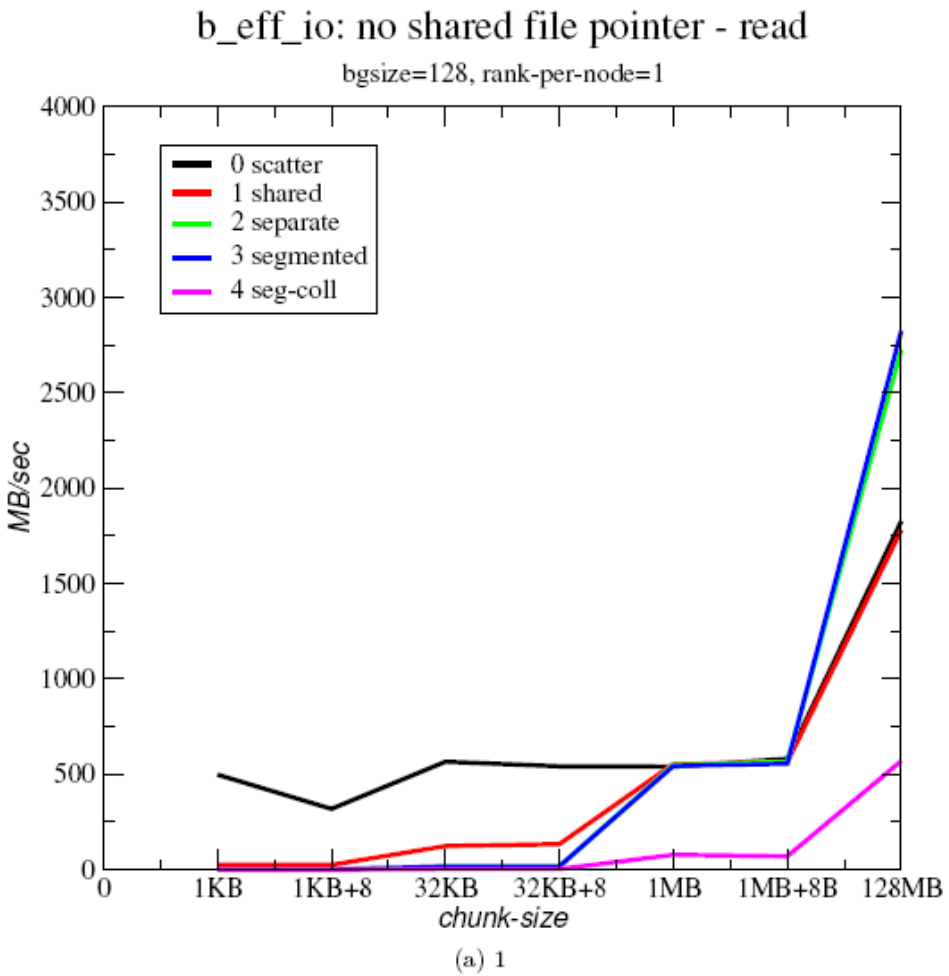
(0) strided collective access, scattering large chunks in memory with size L each with one MPI-I/O call to/from disk chunks with size l,
(1) strided collective access, but one read or write call per disk chunk,
(2) noncollective access to one file per MPI process, i.e., on separated files,
(3) same as (2), but the individual files are assembled to one segmented file, and
(4) same as (3), but the access to the segmented file is done with collective routines;


b_eff_io: no shared file pointer - read
bgsize=64, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
4000
3500
3000
2500
2000
1500
1000
500
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

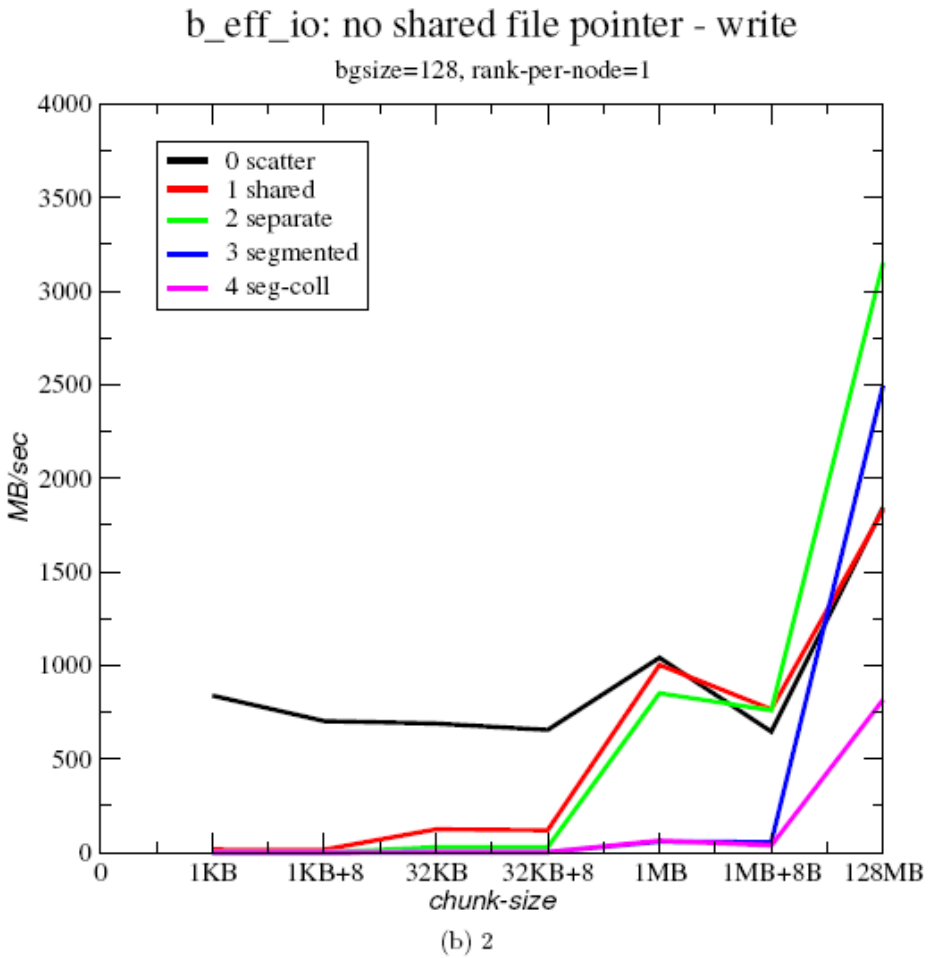
(a) 1


b_eff_io: no shared file pointer - write
bgsize=64, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
4000
3500
3000
2500
2000
1500
1000
500
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

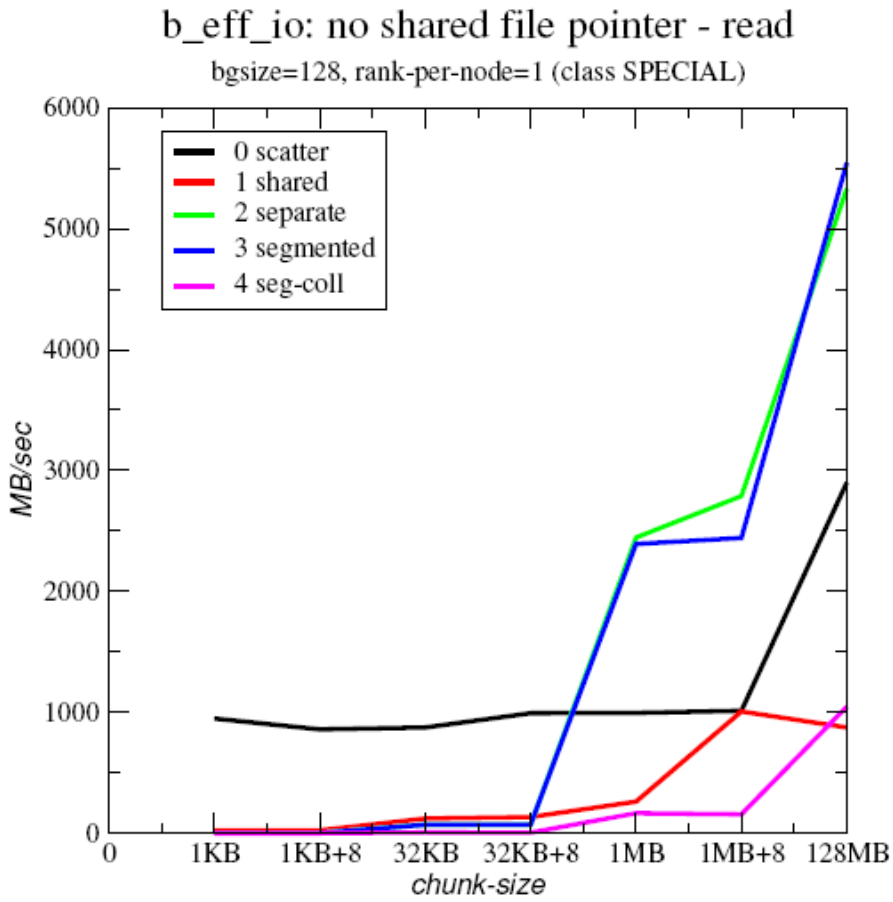
(b) 2


b_eff_io: no shared file pointer - read
bgsize=128, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
chunk-size
0
500
1000
1500
2000
2500
3000
3500
4000
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8B
128MB

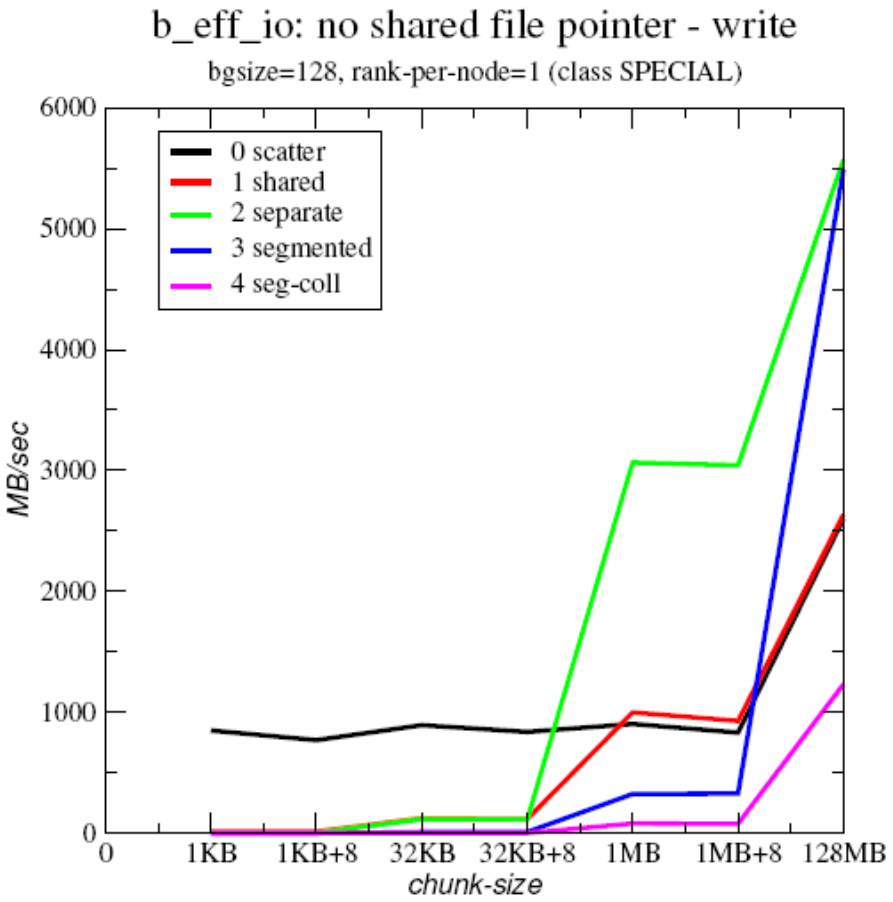
(a) 1


b_eff_io: no shared file pointer - write
bgsize=128, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
chunk-size
0
500
1000
1500
2000
2500
3000
3500
4000
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8B
128MB

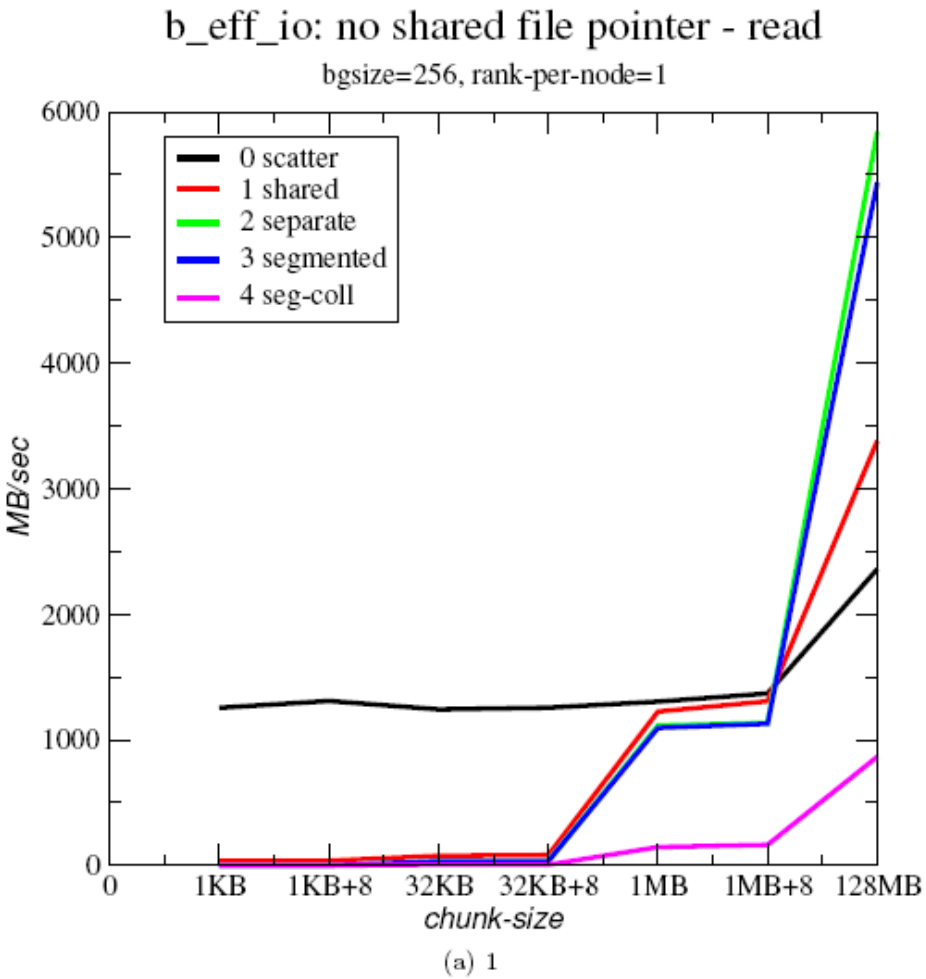
(b) 2


b_eff_io: no shared file pointer - read
bgsize=128, rank-per-node=1 (class SPECIAL)
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
6000
5000
4000
3000
2000
1000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

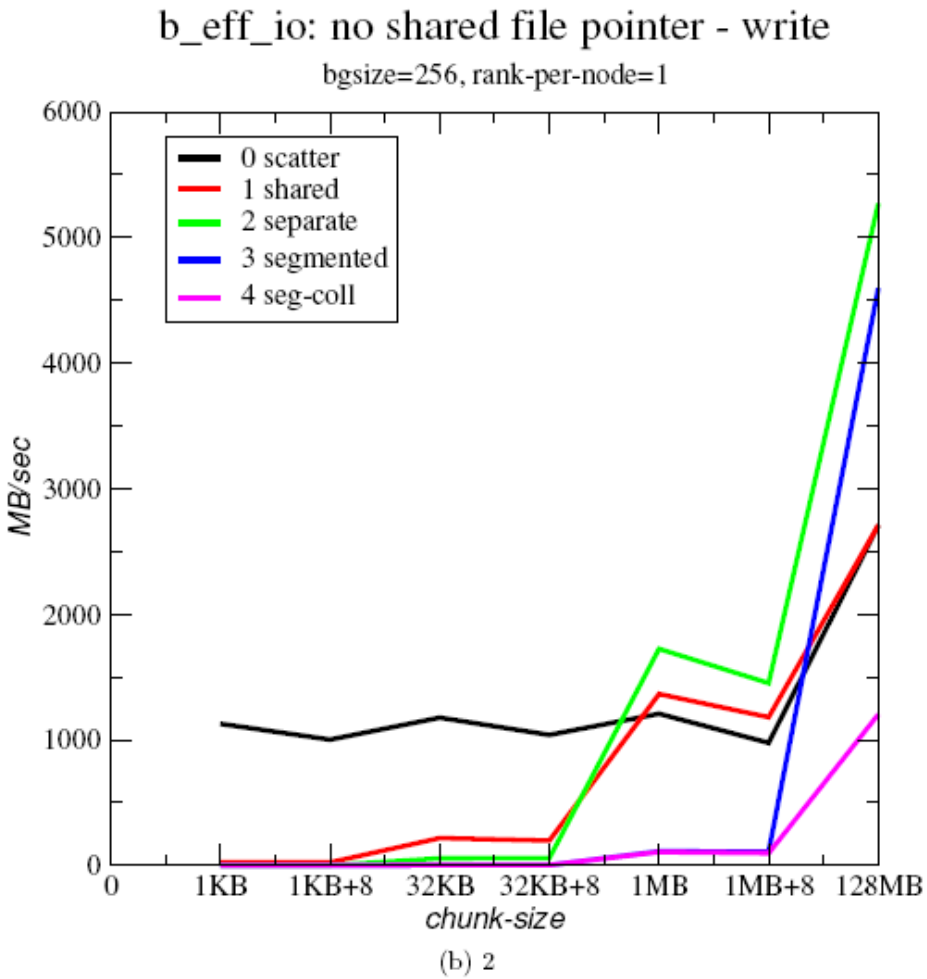
(a) 1


b_eff_io: no shared file pointer - write
bgsize=128, rank-per-node=1 (class SPECIAL)
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
6000
5000
4000
3000
2000
1000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

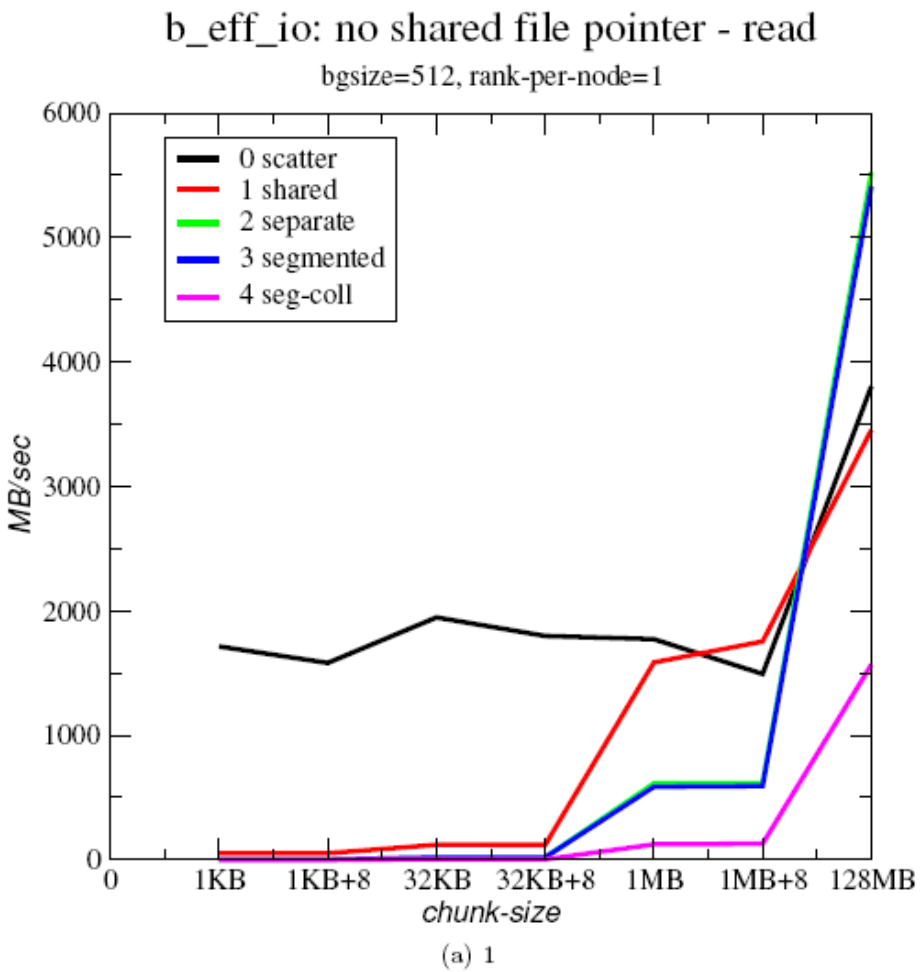
(b) 2


b_eff_io: no shared file pointer - read
bgsize=256, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
6000
5000
4000
3000
2000
1000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

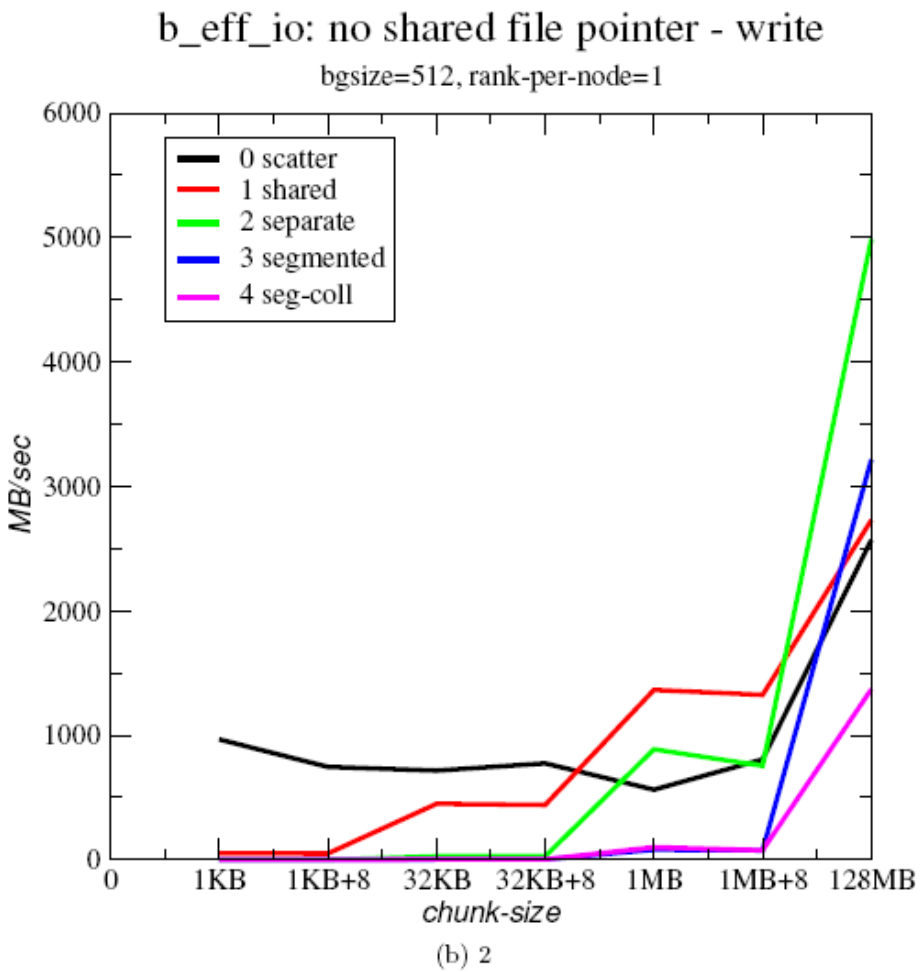
(a) 1


b_eff_io: no shared file pointer - write
bgsize=256, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
6000
5000
4000
3000
2000
1000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

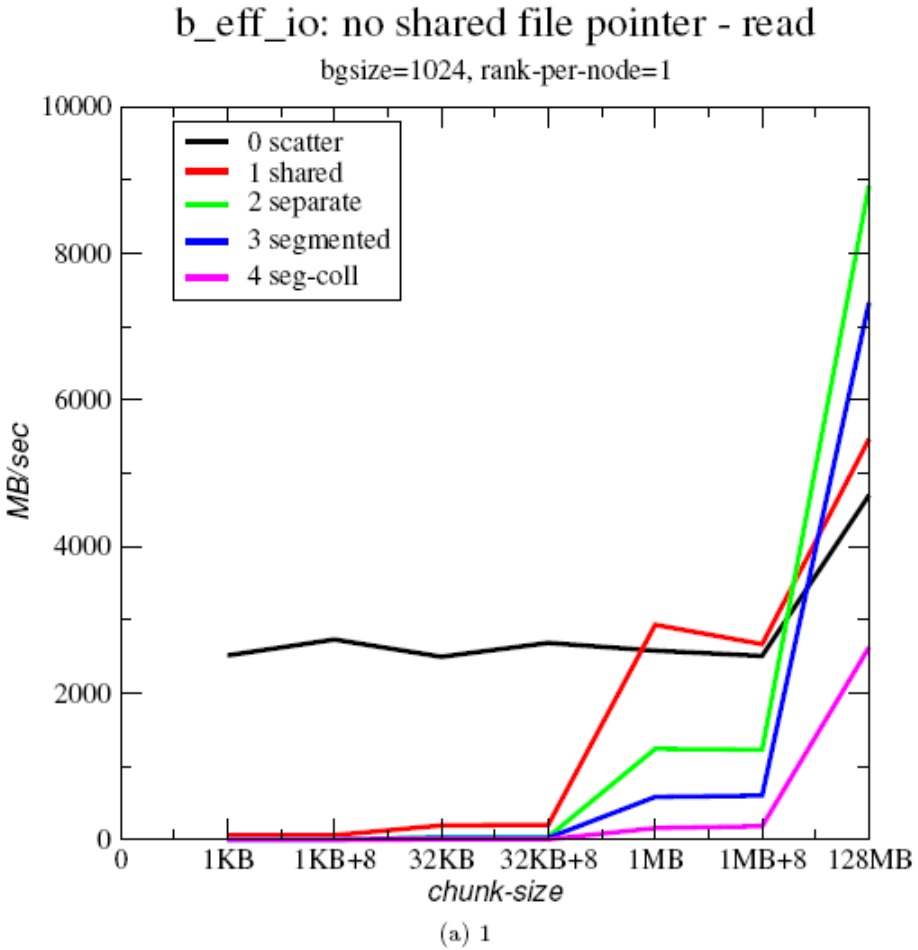
(b) 2


b_eff_io: no shared file pointer - read
bgsize=512, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
6000
5000
4000
3000
2000
1000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

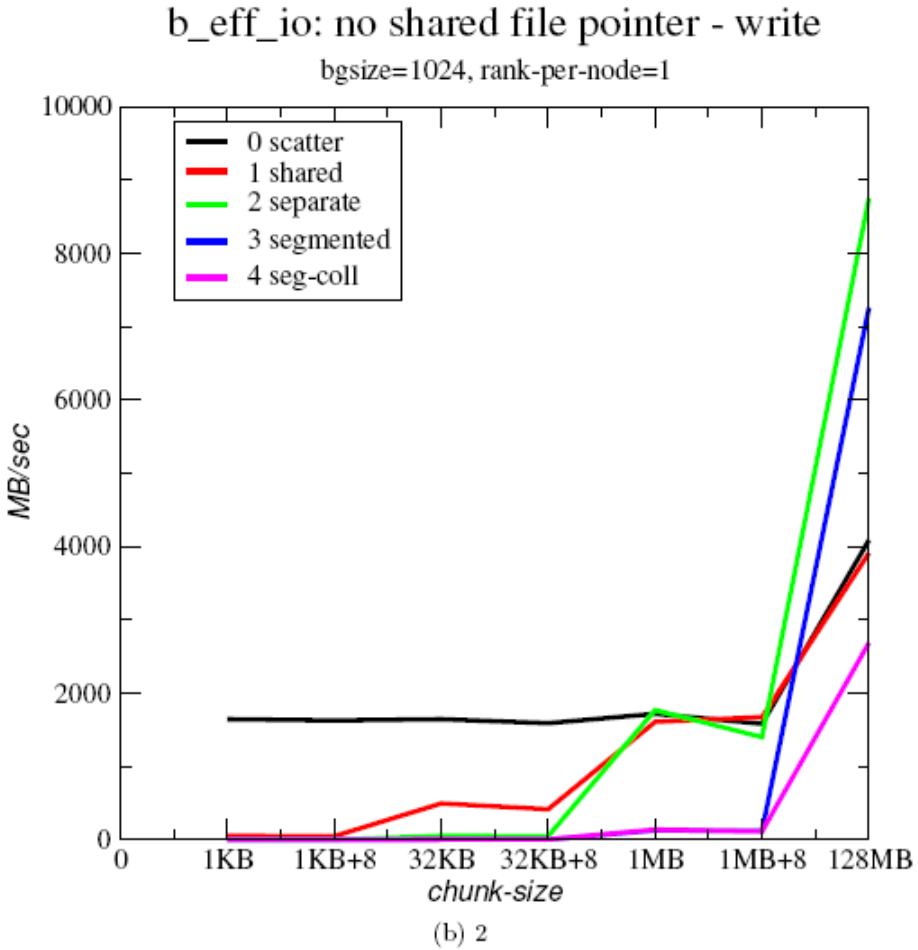
(a) 1


b_eff_io: no shared file pointer - write
bgsize=512, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
6000
5000
4000
3000
2000
1000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

(b) 2


b_eff_io: no shared file pointer - read
bgsize=1024, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
10000
8000
6000
4000
2000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

(a) 1


b_eff_io: no shared file pointer - write
bgsize=1024, rank-per-node=1
0 scatter
1 shared
2 separate
3 segmented
4 seg-coll
MB/sec
10000
8000
6000
4000
2000
0
0
1KB
1KB+8
32KB
32KB+8
1MB
1MB+8
128MB
chunk-size

(b) 2

Considerations

- Use chunk size greater than 4MB
- IO triavially parallel show good performance only if the chunk size is greater than 4MB
- Depending on the view of the file and on the chunk size, MPI-IO presents good performance.
- The performance of strided MPI-IO are more or less the same, independently on the chunk size.
- Asynchronous MPI-IO can improve the performance of the application.


CINECA
KEEP
CALM
AND
USE THE
DEFAULT

THANKS FOR ATTENTION !!!

QUESTIONS ???