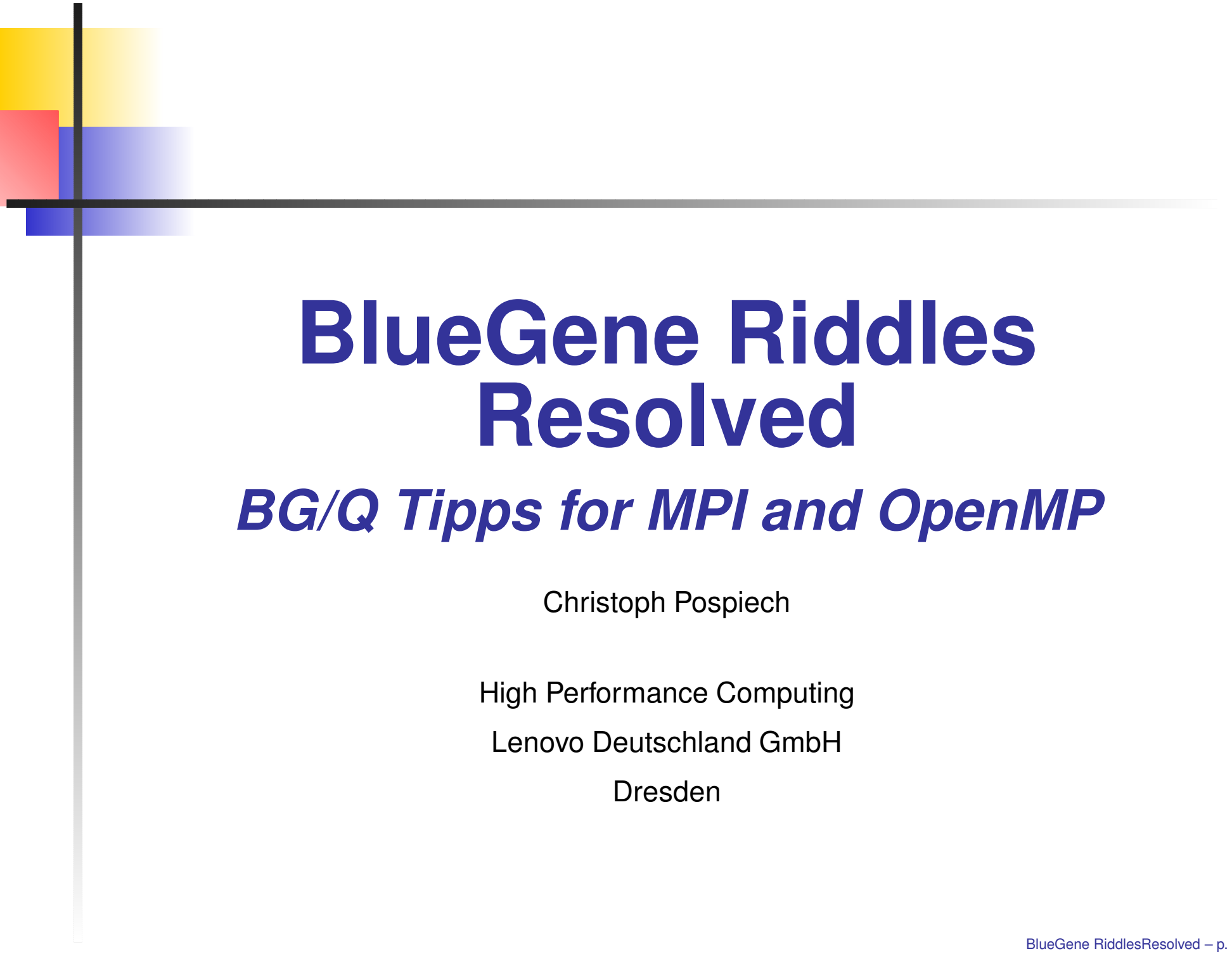


BlueGene Riddles Resolved

BG/Q Tipps for MPI and OpenMP

Christoph Pospiech

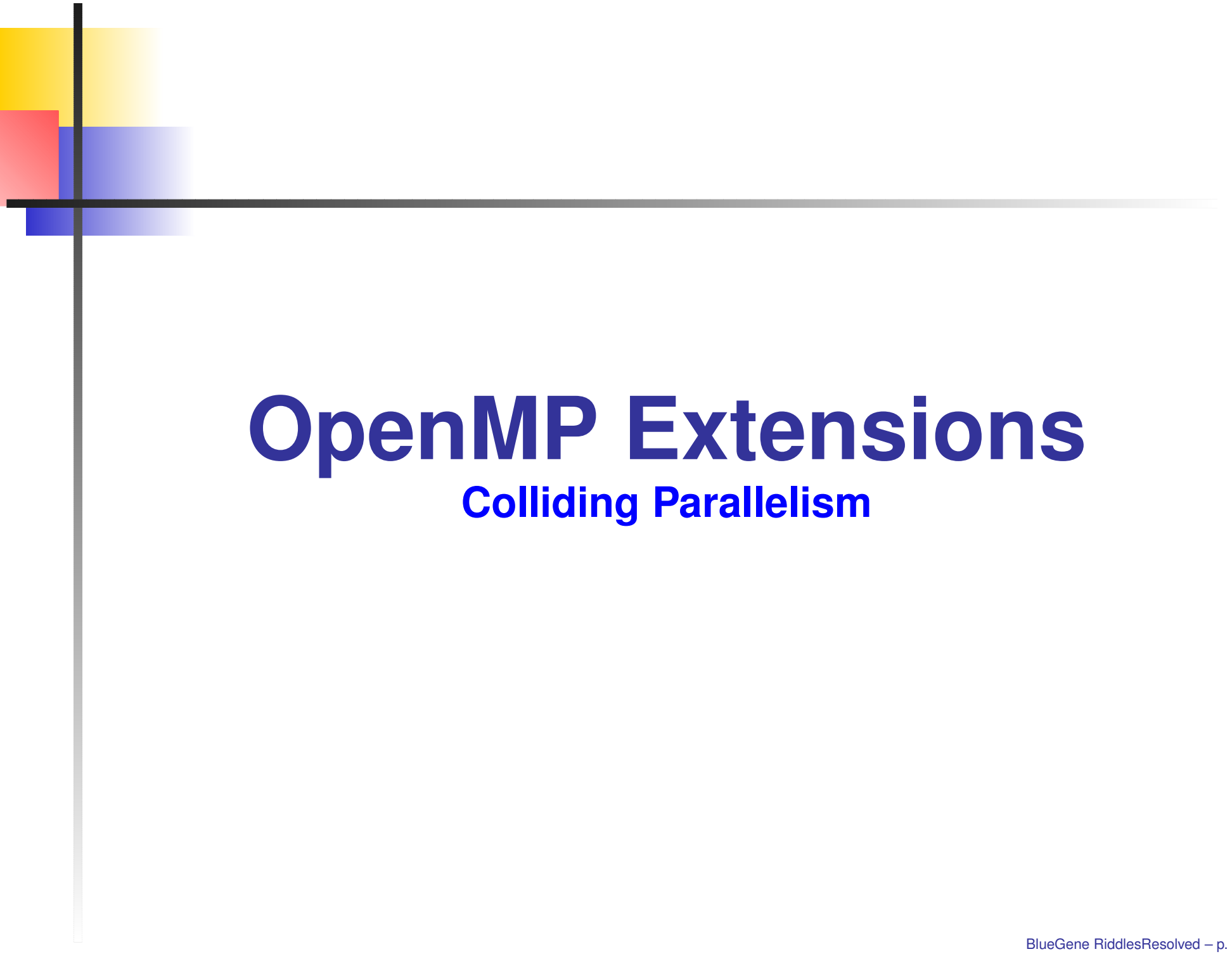
High Performance Computing

Lenovo Deutschland GmbH

Dresden

Contents

- OpenMP Extensions
  - Transactional Memory
  - Thread-level Speculation
- I/O Subsystem
- MPI Task Mapping
- Hybrid Parallelization



OpenMP Extensions

Colliding Parallelism

LOCKS and Deadlock

- Consider the following example

```
void move(T s, T d, Obj key){
  LOCK(s); LOCK(d);
  tmp = s.remove(key);
  d.insert(key, tmp);
  UNLOCK(d); UNLOCK(s);
}
```

- This can deadlock !

Thread 0 `move(a, b, key1);`

Thread 1 `move(b, a, key2);`

Transactional Memory (TM)

- Programmer says — “I want this atomic”

```
void move(T s, T d, Obj key){
  #pragma TM_ATOMIC SAFE_MODE {
    tmp = s.remove(key);
    d.insert(key, tmp);
  }
}
```

- TM system — “I’ll make it so”
  - Avoids deadlock
  - Replaces fine grain locking

Unleashing TM

- Compiler flag to enable TM pragmas: −qtm

Unleashing TM

- Compiler flag to enable TM pragmas: −qtm
- User needs to add directives

Unleashing TM

- Compiler flag to enable TM pragmas: −qtm
- User needs to add directives
- Use SAFE_MODE if no access to I/O

Unleashing TM

- Compiler flag to enable TM pragmas: −qtm
- User needs to add directives
- Use SAFE_MODE if no access to I/O
- Performance governed by tradeoff
  - Execution time of atomic region
    (entry and exit overhead)
  - conflict probability
    (rollback overhead)

Unleashing TM

- Compiler flag to enable TM pragmas: −qtm
- User needs to add directives
- Use SAFE_MODE if no access to I/O
- Performance governed by tradeoff
  - Execution time of atomic region
    (entry and exit overhead)
  - conflict probability
    (rollback overhead)
- http://spscicomp.org/wordpress/wp-content/...
  uploads/2013/03/maurer-*.pdf

Thread-level Speculation

- Similar to Transactional Memory

  (But different usage model)

Thread-level Speculation

- Similar to Transactional Memory

  (But different usage model)

- Leverages existing OpenMP parallelization

  (However less assumptions for the compiler)

Thread-level Speculation

- Similar to Transactional Memory

  (But different usage model)

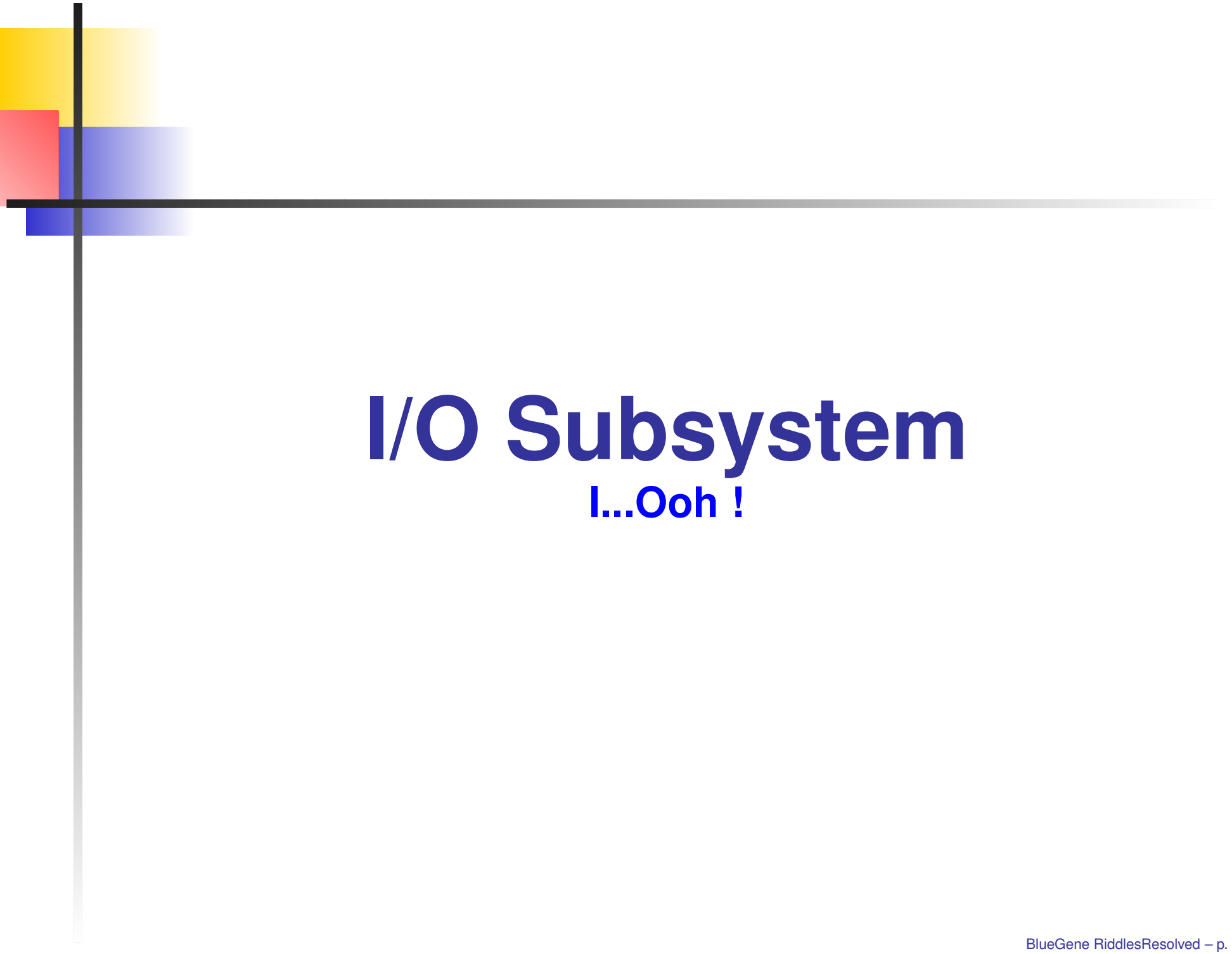
- Leverages existing OpenMP parallelization

  (However less assumptions for the compiler)

- Subdivision into work units without locking

Thread-level Speculation

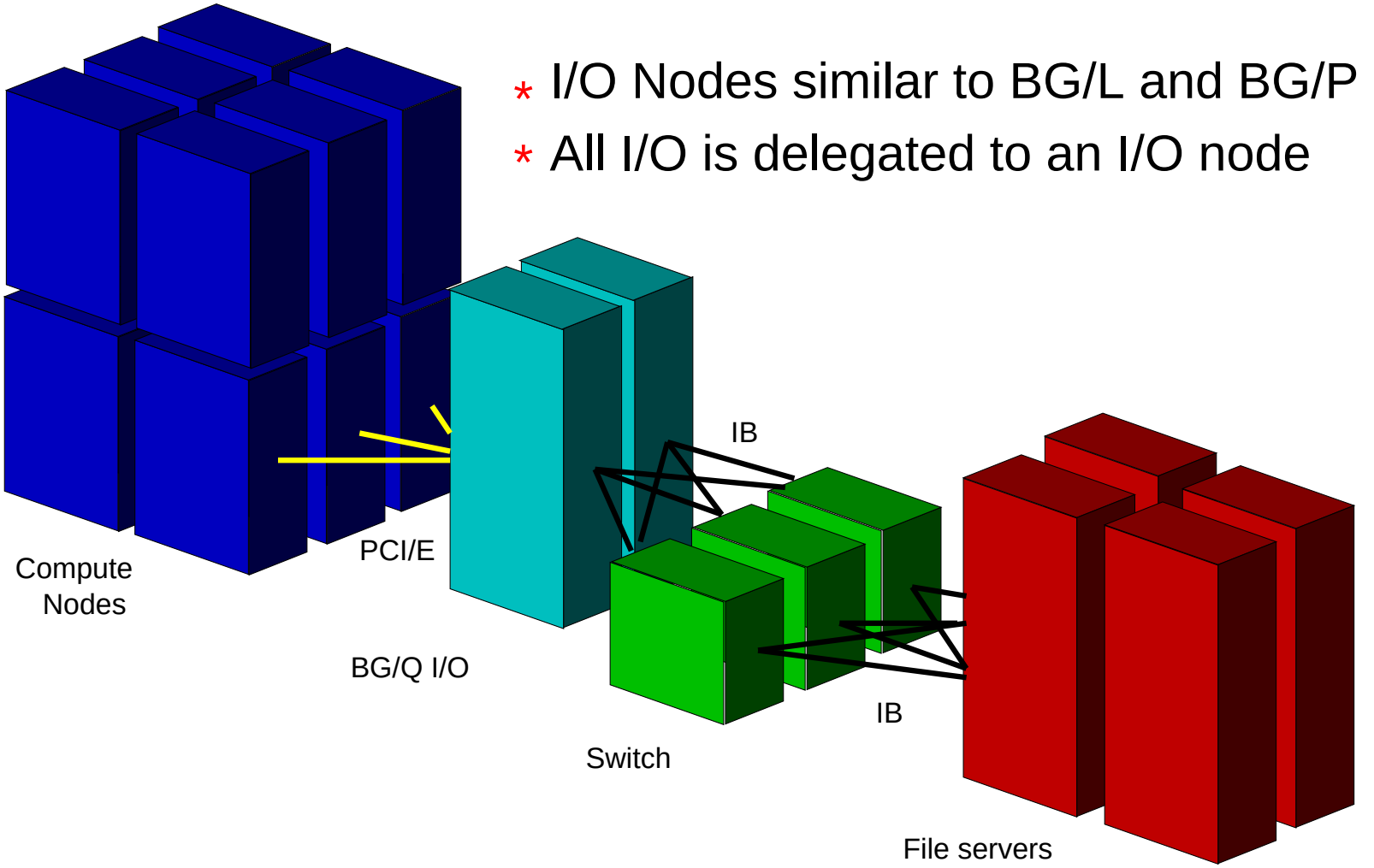
- Similar to Transactional Memory
  (But different usage model)
- Leverages existing OpenMP parallelization
  (However less assumptions for the compiler)
- Subdivision into work units without locking
- If work units collide in memory
  - Hardware detects collision
  - Kernel rolls back transaction
  - Runtime decides whether to retry or serialize



I/O Subsystem

I...Ooh !

BlueGene/Q I/O



* I/O Nodes similar to BG/L and BG/P
* All I/O is delegated to an I/O node

I/O Considerations

- I/O is delegated to the I/O nodes
  - Compute nodes free from I/O load
  - Bandwidth to I/O nodes limited by PCI/E
- Beware of I/O only from MPI rank 0
  - Uses only one link to I/O nodes
  - Doesn’t scale
- Consider MPI I/O

MPI Task Mapping

Nail it down !

5-dimensional Torus

- Given by 5 coordinates ABCDE

# Node Cards	# Nodes	Torus size	IsTorus
1	32	2x2x2x2x2	00001
2(adjacent)	64	2x2x4x2x2	00101
4(quadrants)	128	2x2x4x4x2	00111
8(halves)	256	4x2x4x4x2	10111

- Maximal 10 direct neighbours
- On a midplane ≤ 2 hops in one direction
- Different message travel times

To Map or Not To Map

- MPI communication should have
  - a big chunk of the execution time
  - a well understood communication pattern

To Map or Not To Map

- MPI communication should have
  - a big chunk of the execution time
  - a well understood communication pattern
- Set up MPI task mapping via
  - MPI topology (your mileage may vary)
  - runjob option –mapping
  - Set reorder=false if using both

To Map or Not To Map

- MPI communication should have
  - a big chunk of the execution time
  - a well understood communication pattern
- Set up MPI task mapping via
  - MPI topology (your mileage may vary)
  - runjob option –mapping
  - Set reorder=false if using both
- Necessary LoadLeveler keywords
  - #bg_shape
  - #bg_rotate = FALSE

Understanding MPI

Basic features of MPI (ordered by importance)

- Global Communication

Understanding MPI

Basic features of MPI (ordered by importance)

- Global Communication
- MPI Derived Datatypes

Understanding MPI

Basic features of MPI (ordered by importance)

- Global Communication
- MPI Derived Datatypes
- Communicators

Understanding MPI

Basic features of MPI (ordered by importance)

- Global Communication
- MPI Derived Datatypes
- Communicators
- Point to Point Communication

Understanding MPI

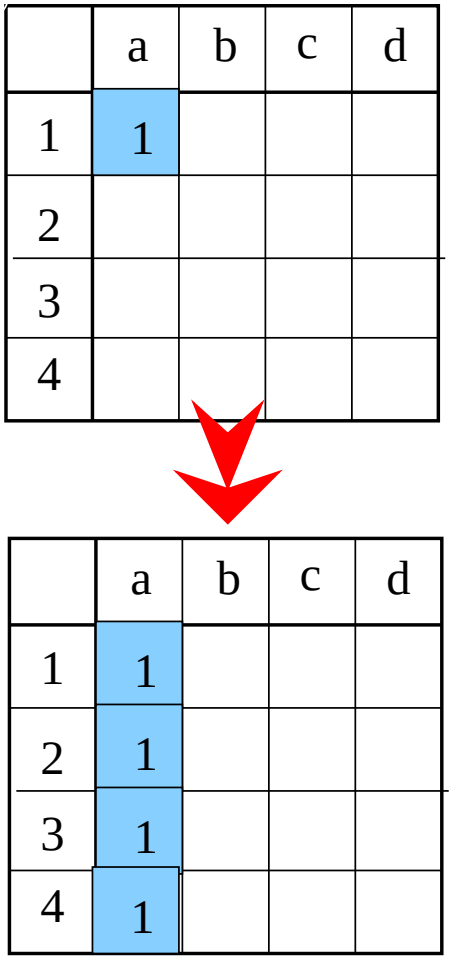
Basic features of MPI (ordered by importance)

- Global Communication
- MPI Derived Datatypes
- Communicators
- Point to Point Communication

You should view MPI Global Communication as the "BLAS" routines of Distributed Programming. They offer a "High Level" approach to parallel programming.

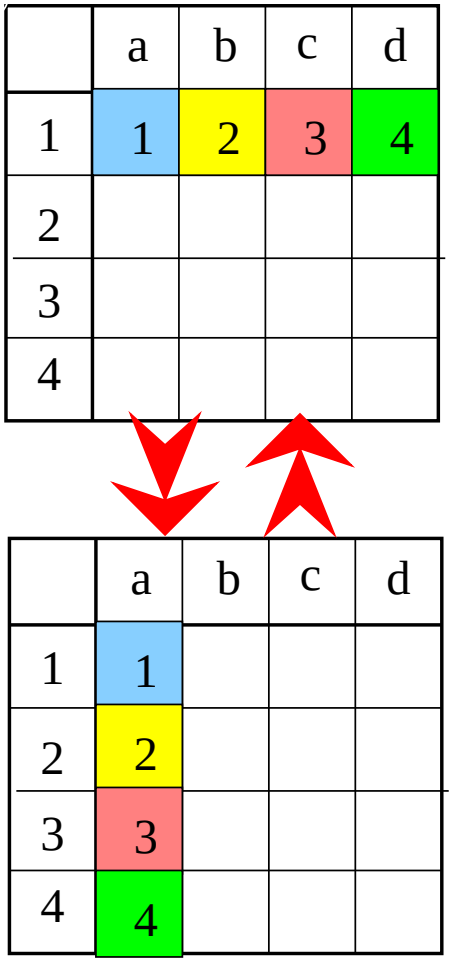
Routines for Global Communication

- broadcast



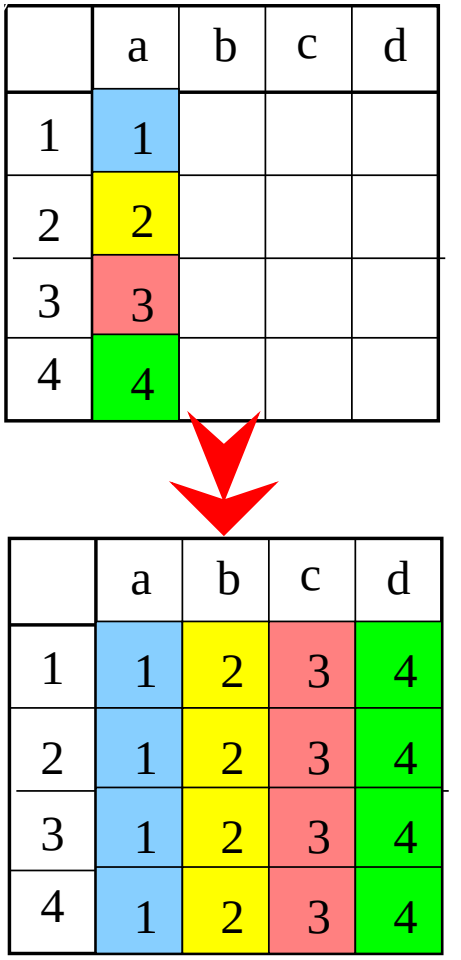
Routines for Global Communication

- broadcast
- gather/scatter



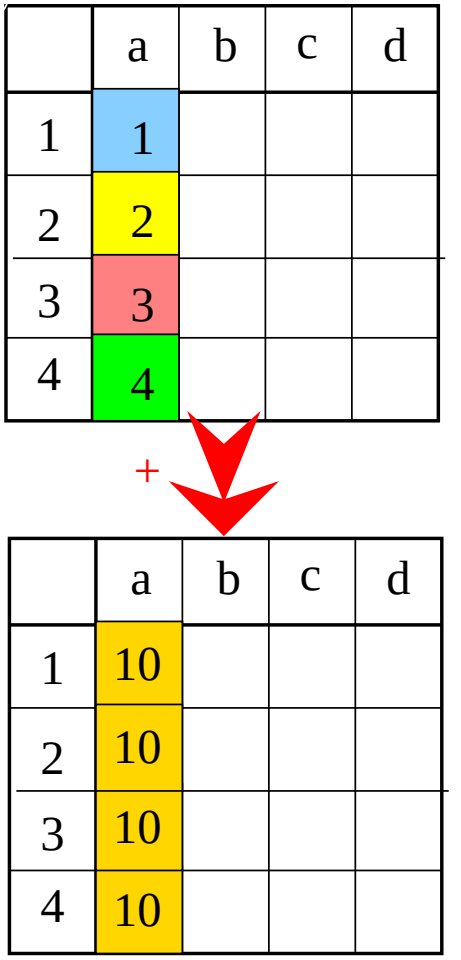
Routines for Global Communication

- broadcast
- gather/scatter
- allgather



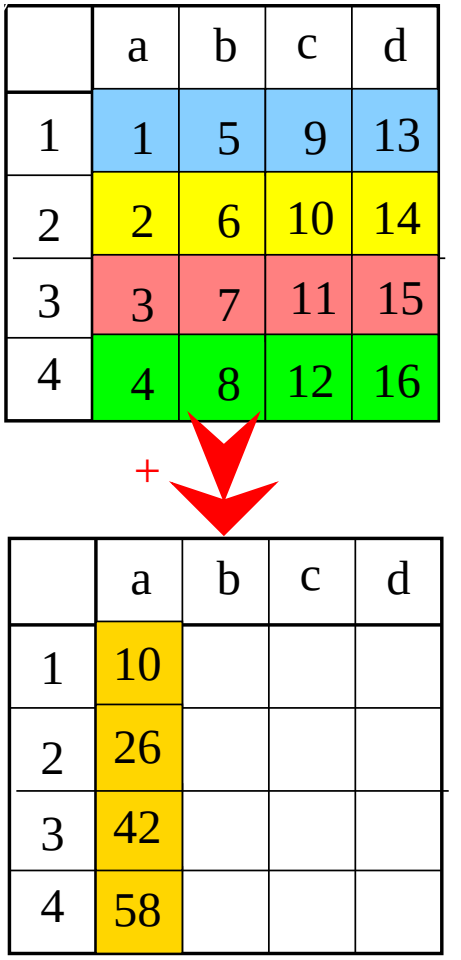
Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce



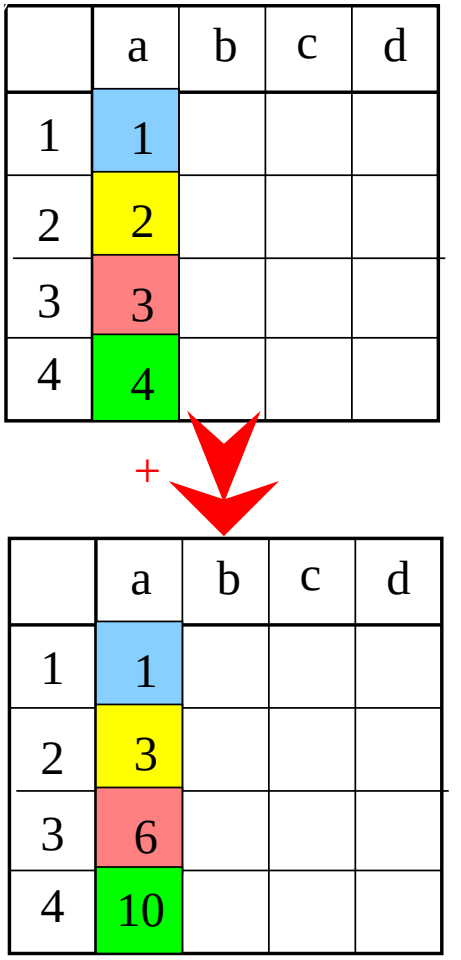
Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter



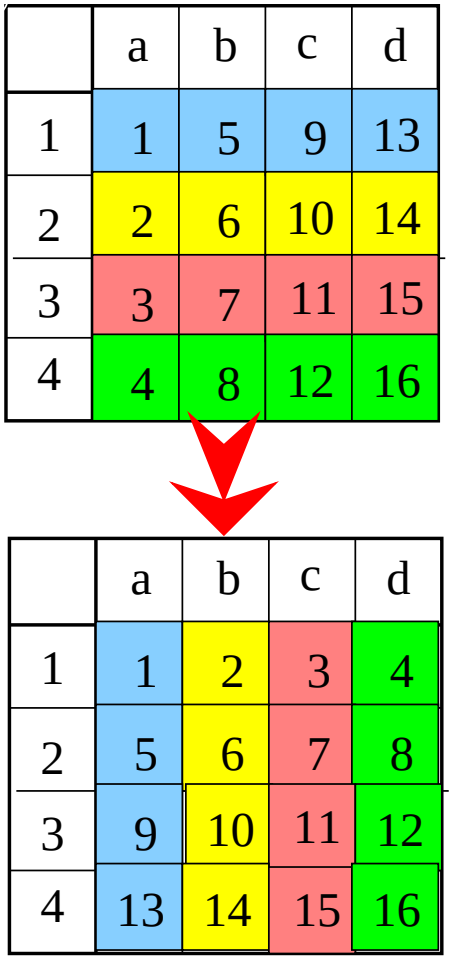
Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan



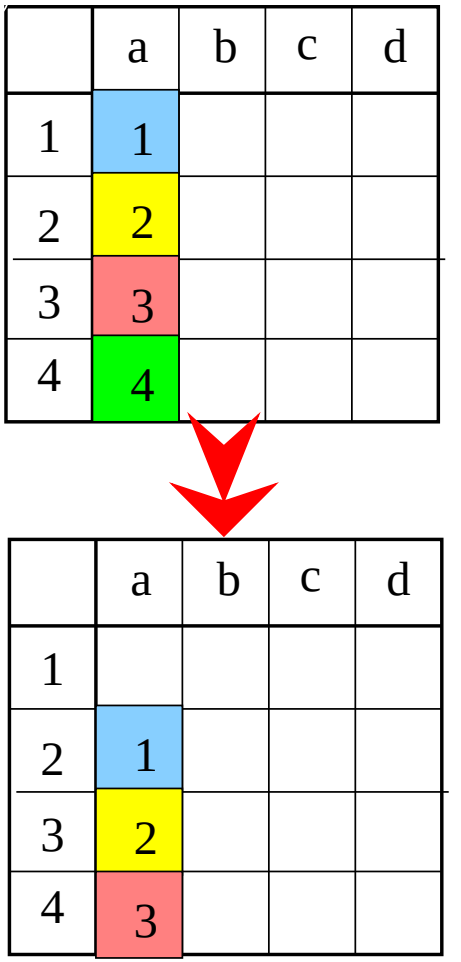
Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan
- all2all



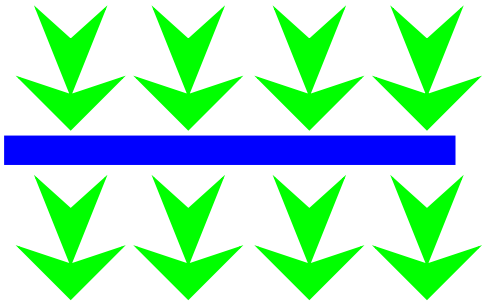
Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan
- all2all
- sendrecv



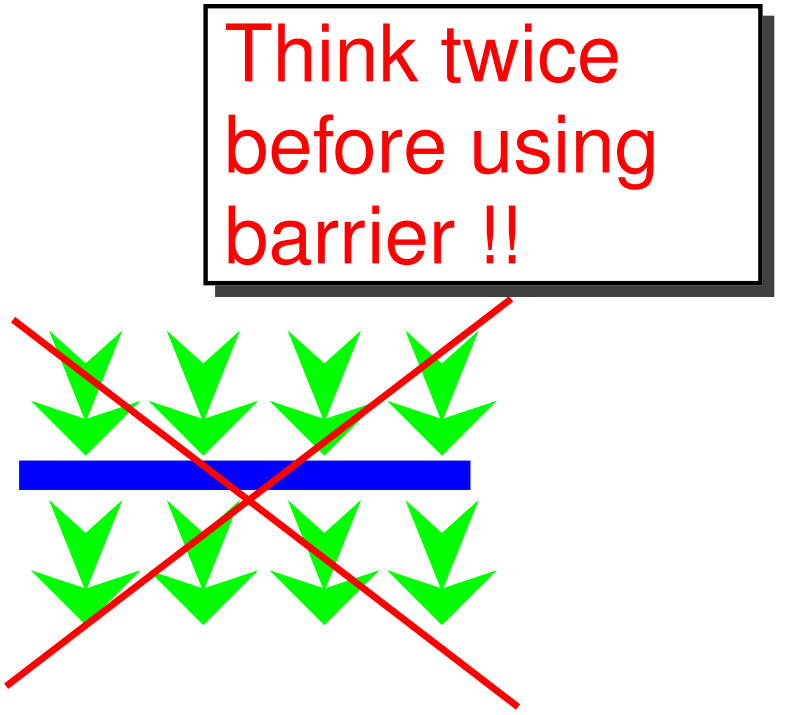
Routines for Global Communication

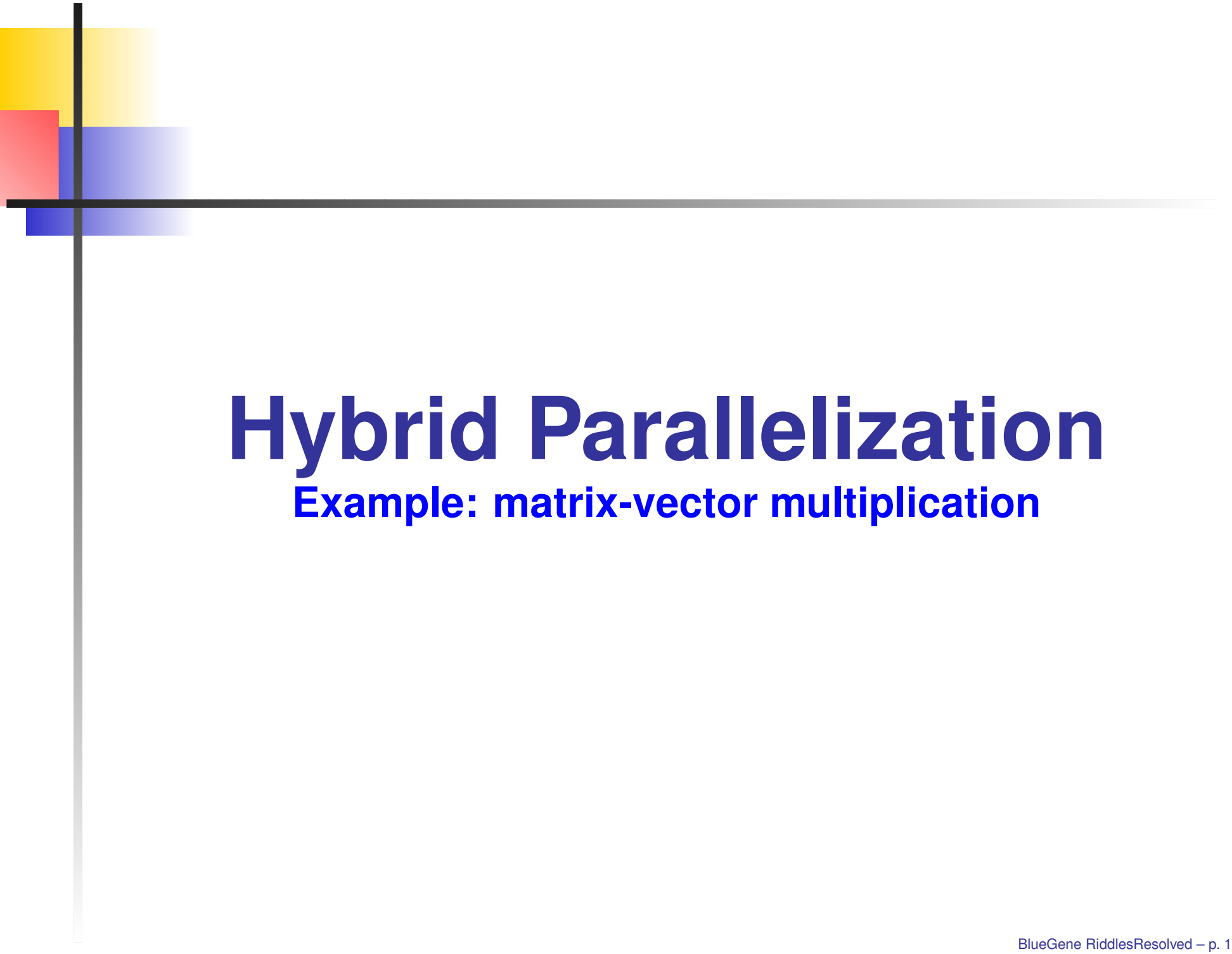
- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan
- all2all
- sendrecv
- barrier



Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan
- all2all
- sendrecv
- barrier

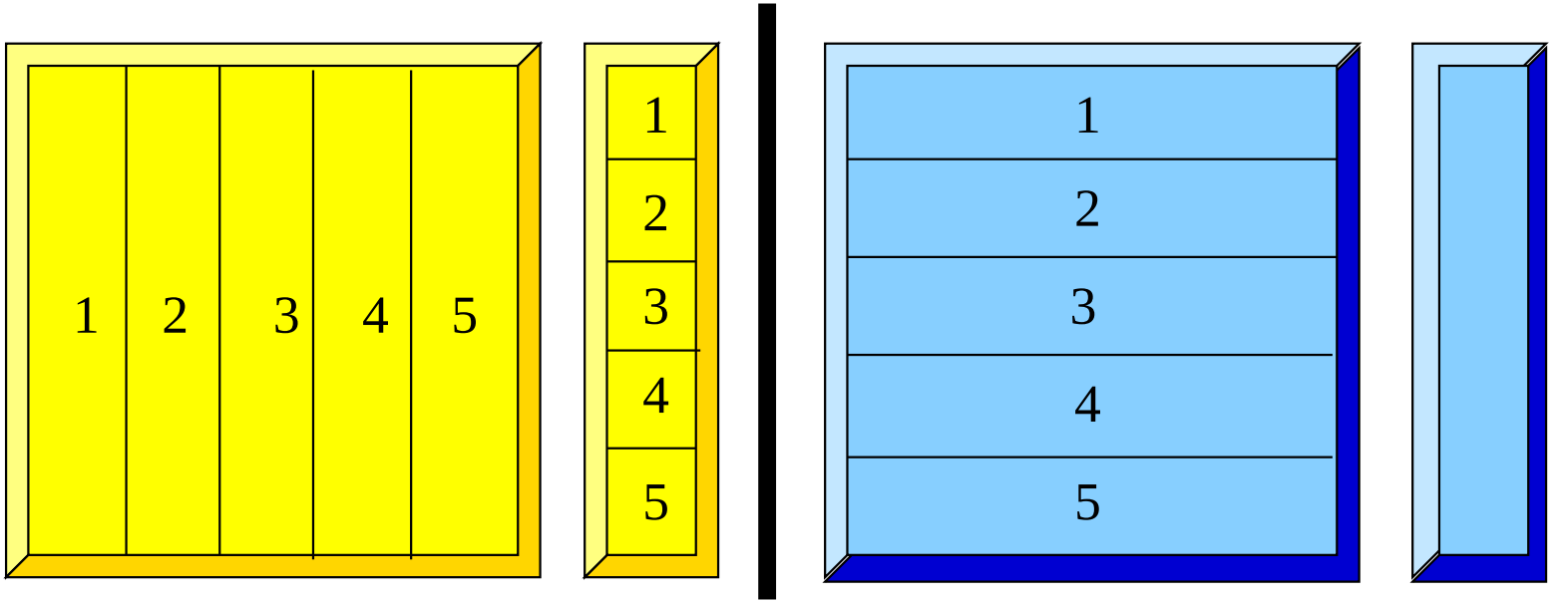




Hybrid Parallelization

Example: matrix-vector multiplication

Choose Data Distribution



Stride 1 access

Access with large stride

Set Up MPI Communication

Do $j = 1$,n_loc

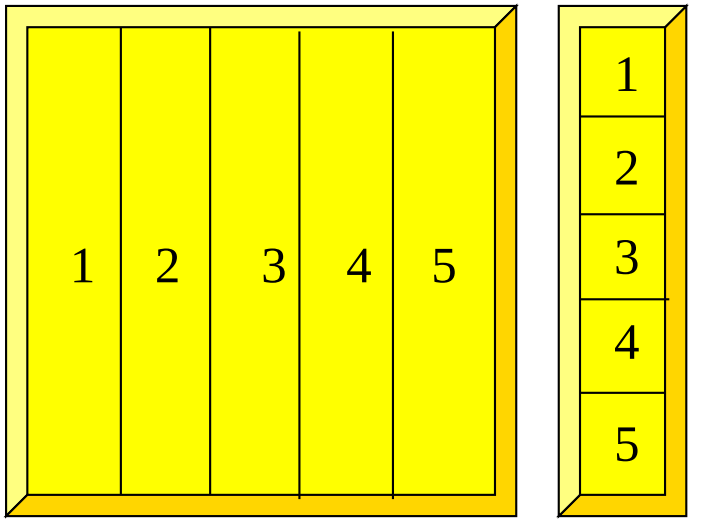
  Do $i = 1$,n

    $c(i) = c(i) + a(i,j) * b(j)$

  end Do

end Do

call mpi_reduce_scatter



SMP Autoparallelization

Do $j = 1$,n_loc

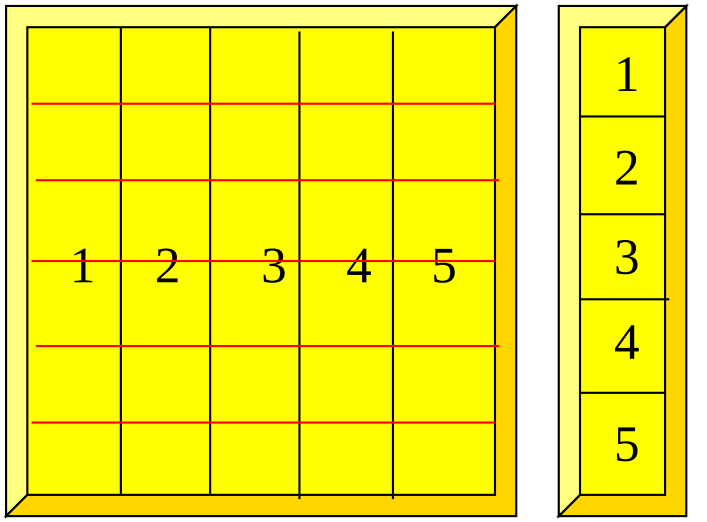
  Do $i = 1$,n

    $c(i) = c(i) + a(i,j) * b(j)$

  end Do

end Do

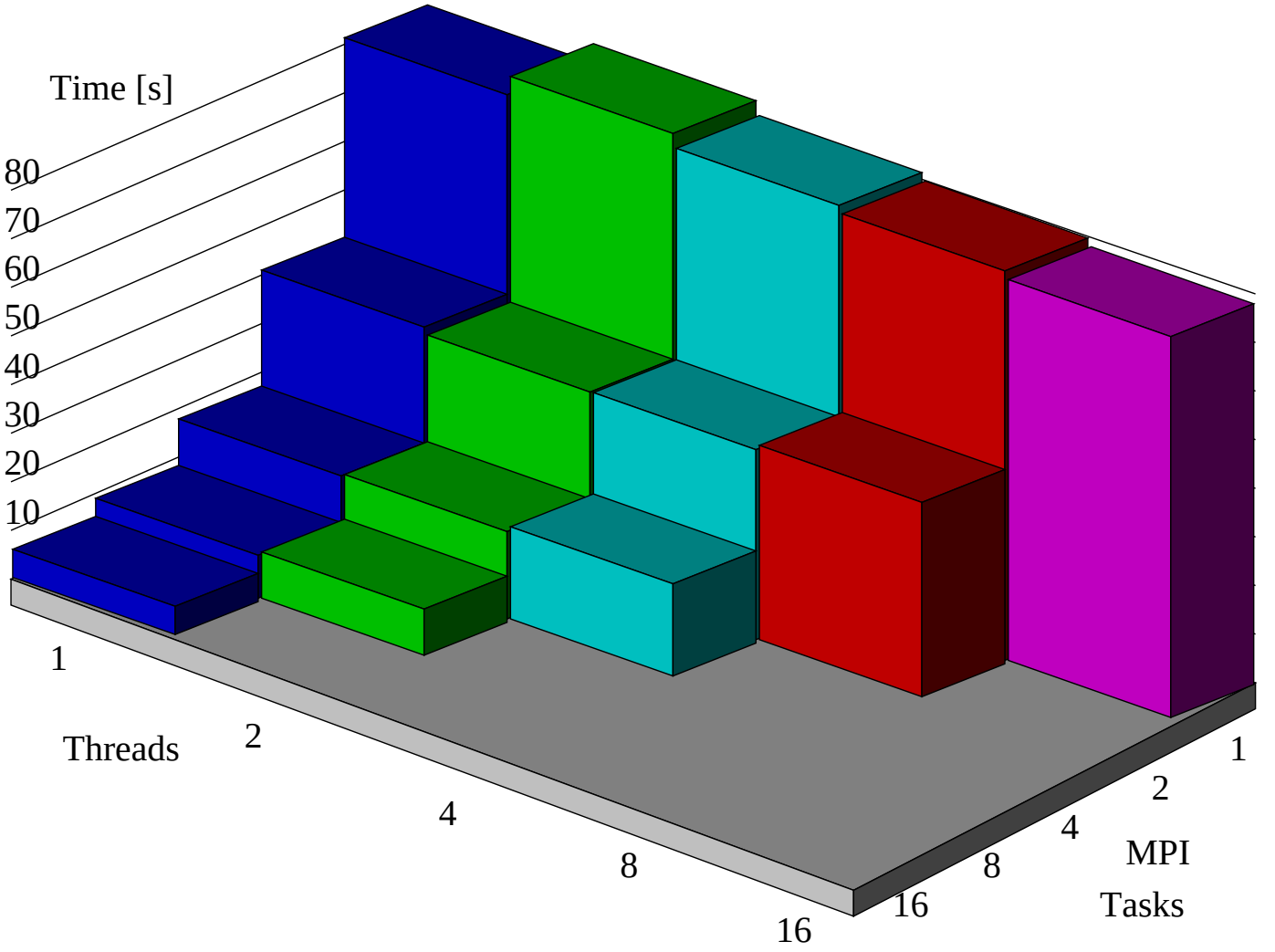
call mpi_reduce_scatter



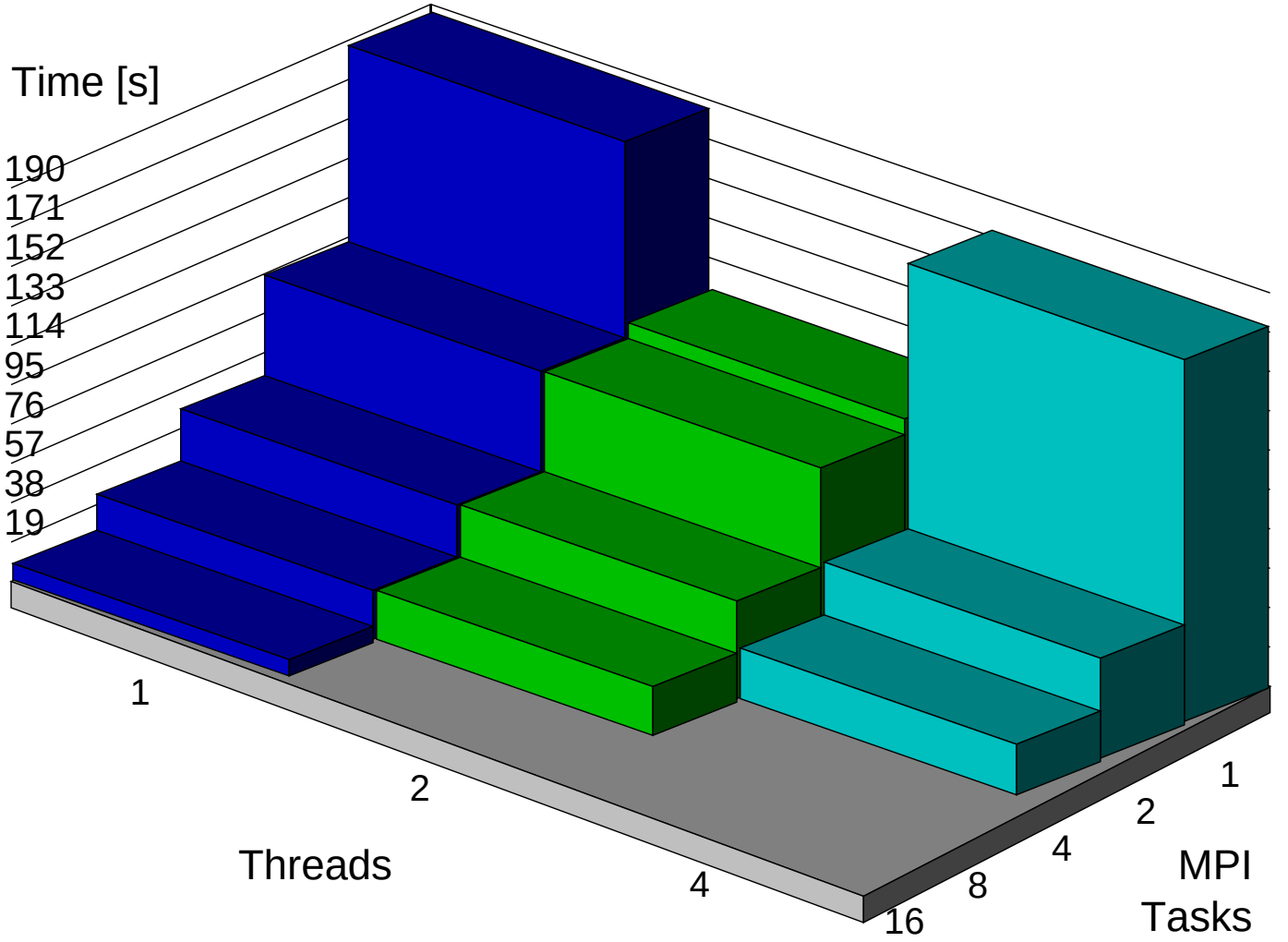
Add -qsmp=auto

If this handles the inner loop, then c is shared !!

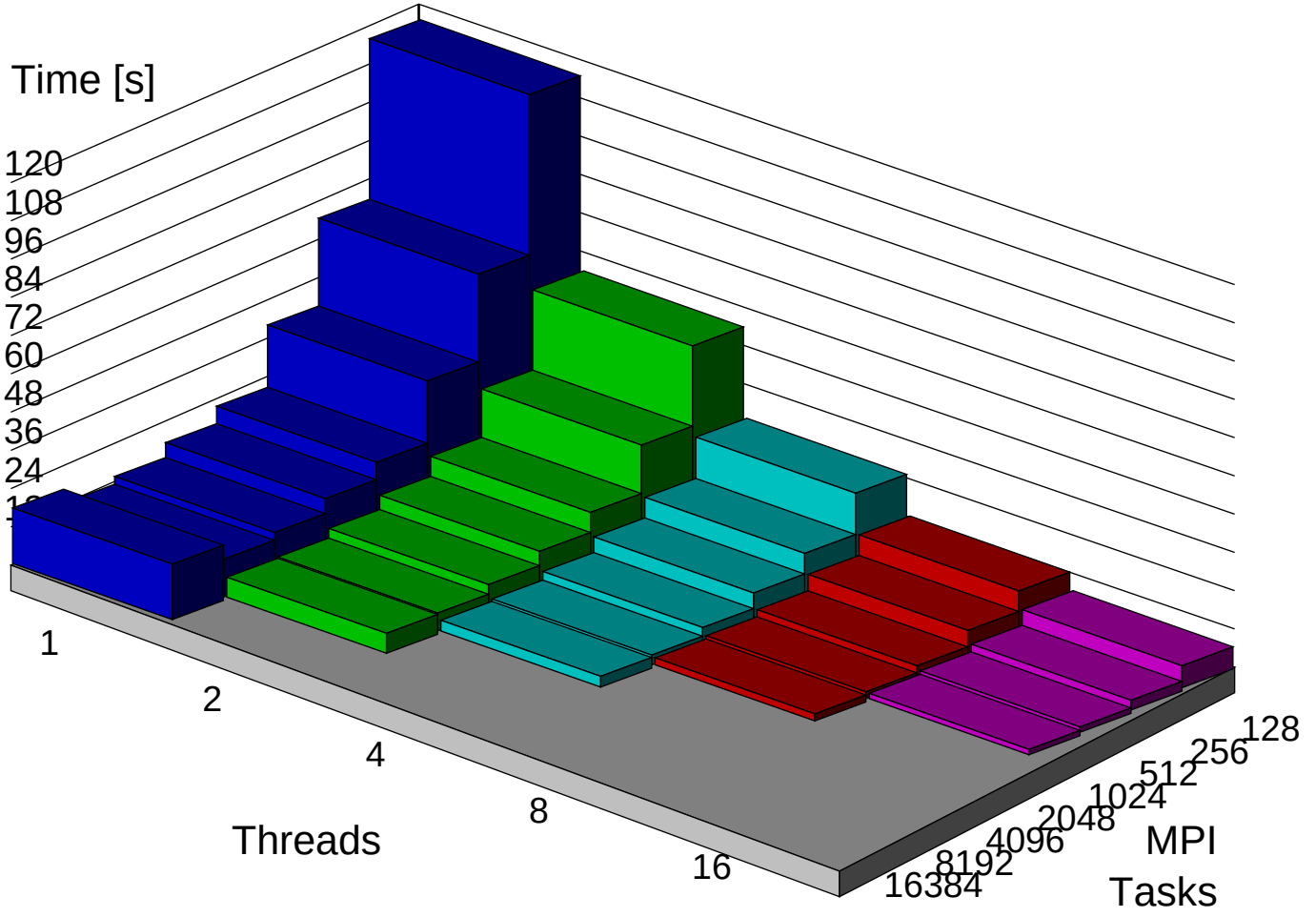
Timings on p690



Timings on BG/P

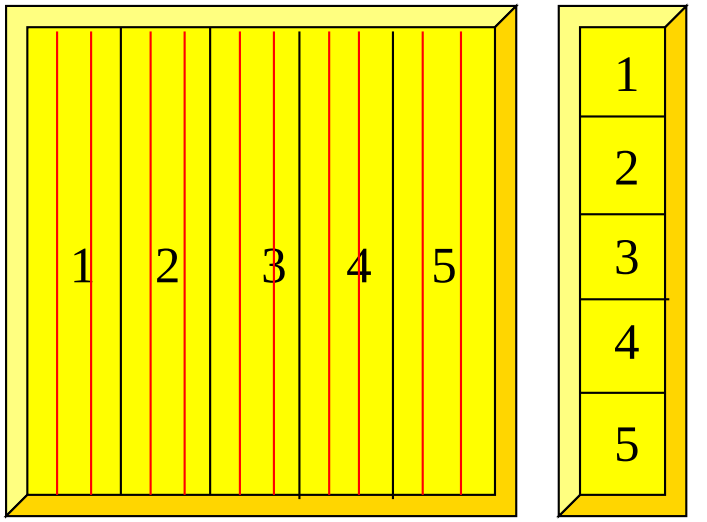


BG/Q Timings (SMT)



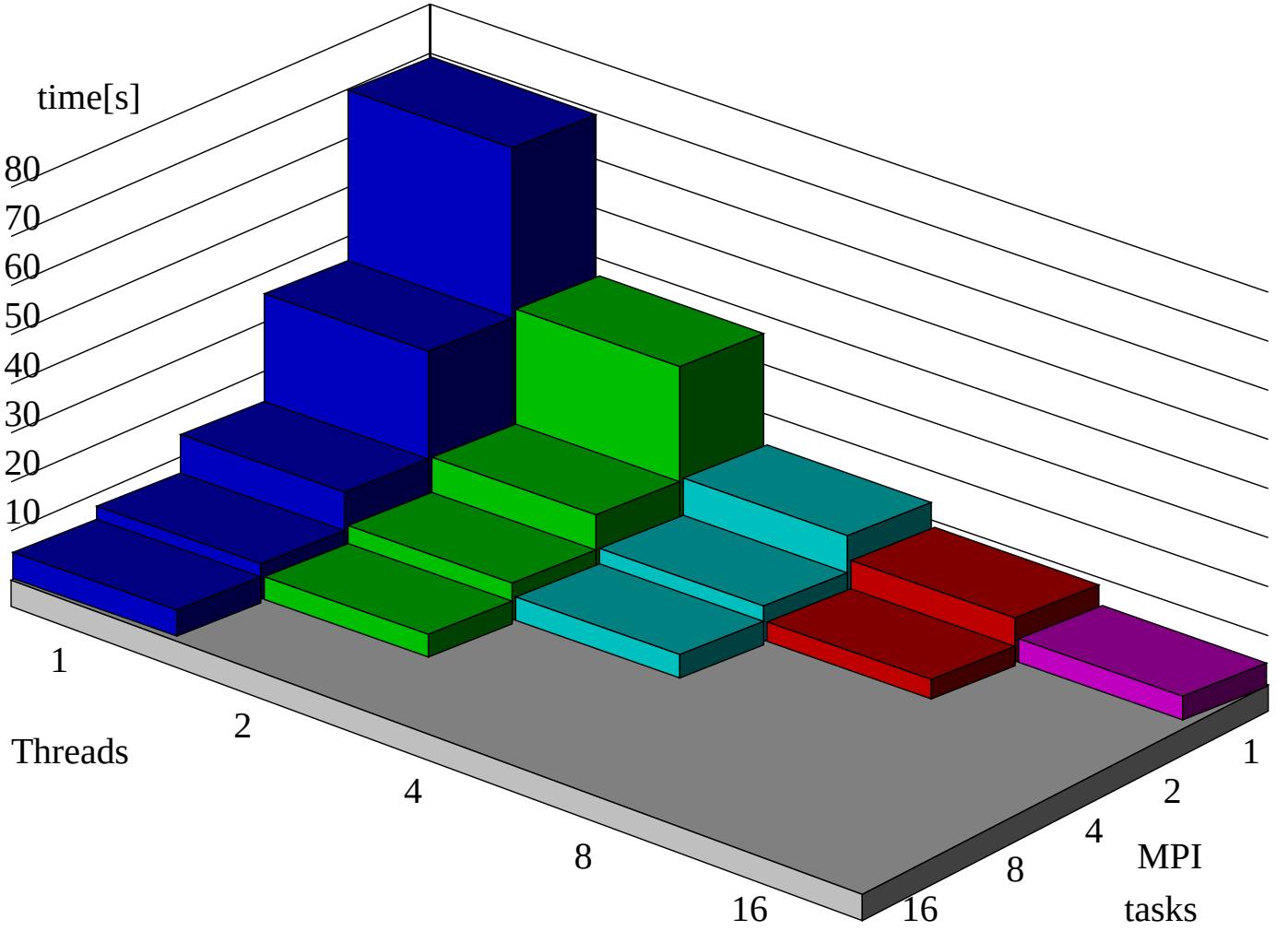
Redistribute SMP Work

$OMP do parallel
$OMP reduction(+:c)
Do $j = 1,n\_loc$
  Do $i = 1,n$
    $c(i) = c(i) + a(i,j) * b(j)$
  end Do
end Do
$OMP end do parallel
call mpi_reduce_scatter

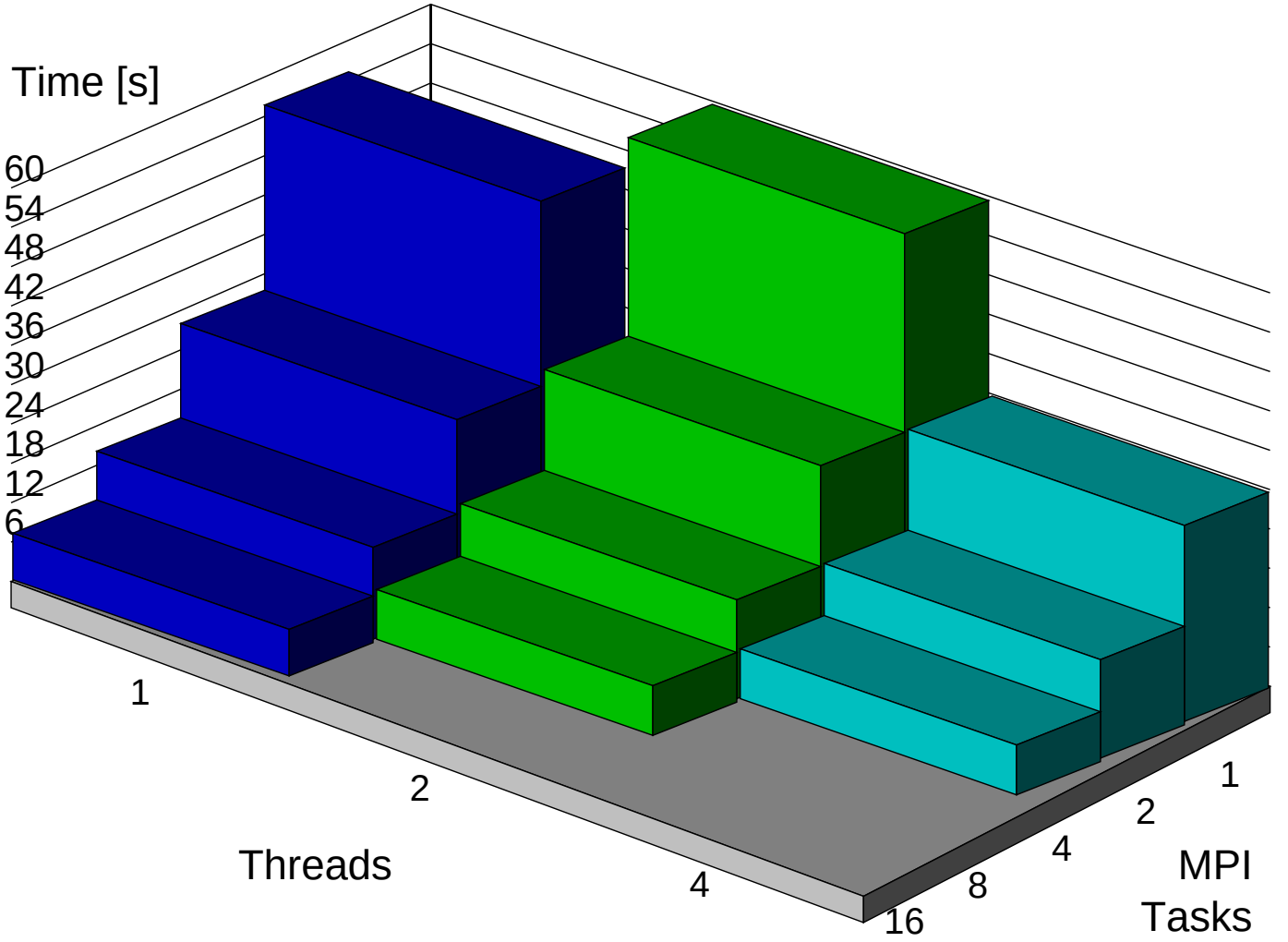


Choose the same loop for SMP and MPI

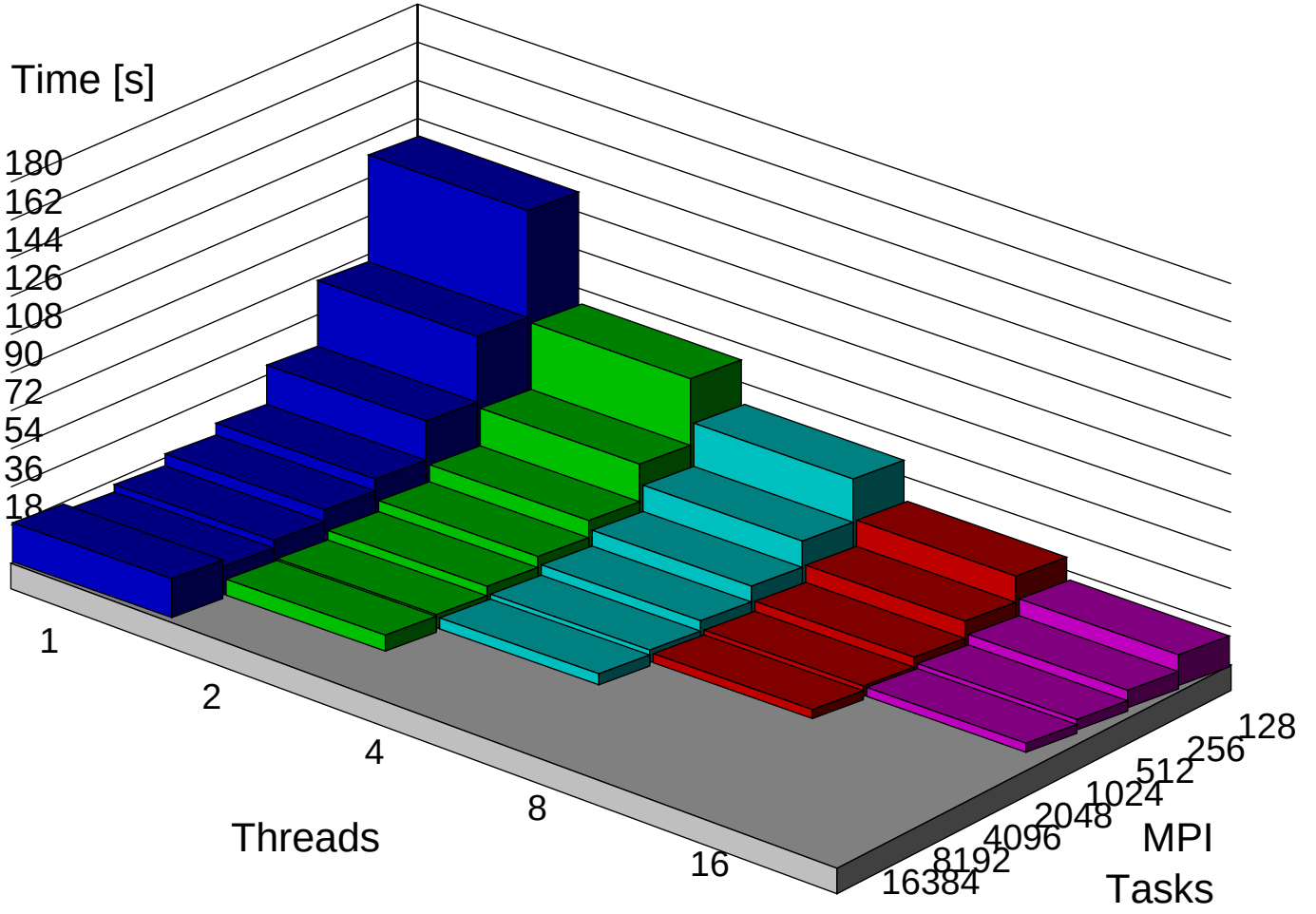
Timings on p690



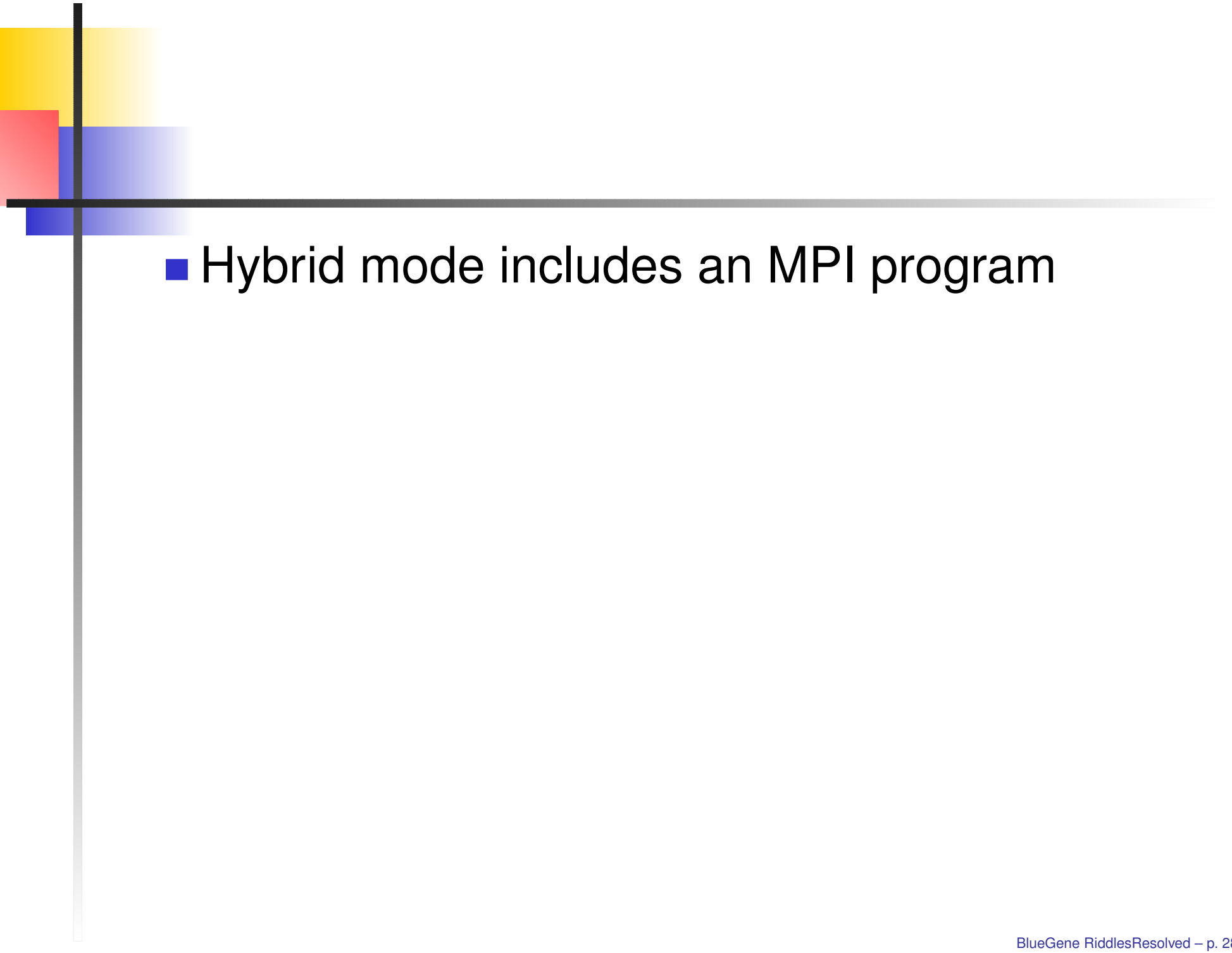
Timings on jugene



BG/Q Timings (SMT)



Conclusion



- Hybrid mode includes an MPI program

Conditions for Hybrid Parallelization

- Hybrid mode includes an MPI program
- SMP on each node has to pay off
  - Easy and effective SMP parallelization
  - MPI communication overhead explodes
  - Escape memory contention
  - Local dynamical load balancing

Conditions for Hybrid Parallelization

- Hybrid mode includes an MPI program
- SMP on each node has to pay off
  - Easy and effective SMP parallelization
  - MPI communication overhead explodes
  - Escape memory contention
  - Local dynamical load balancing

Mixed mode does not invalidate or escape from Amdahl's Law !