# Multi-Gpu Parallelization of Scientific Computing Applications: the Case of the NAS Multi-Zone Parallel Benchmarks

Compilers For High Performance Computing

Computer Architecture Department

Technical University of Catalonia

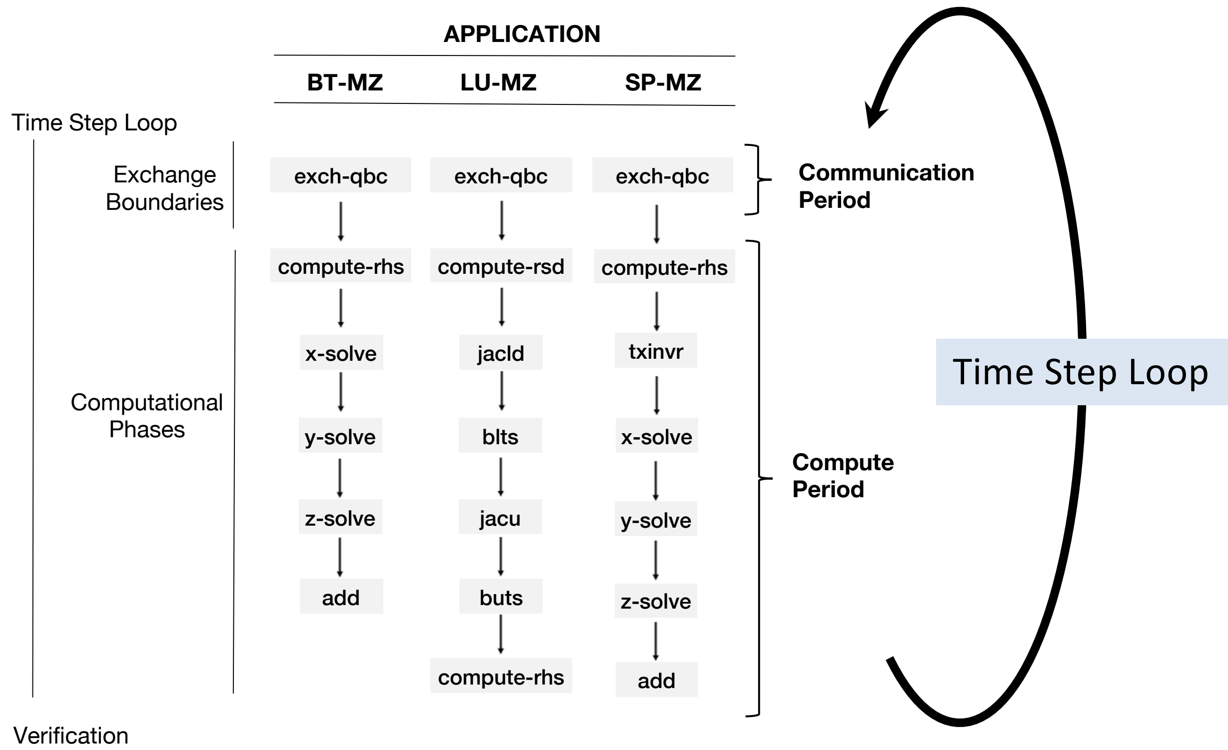
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

# Outline

- **Multi-Zone** NAS Parallel Benchmarks (NPB-MZ)
- NPB-MZ Multi-Gpu Parallelization
- Performance Analysis
- Conclusions

# NPB-MZ: Applications, Computation and Data Structures

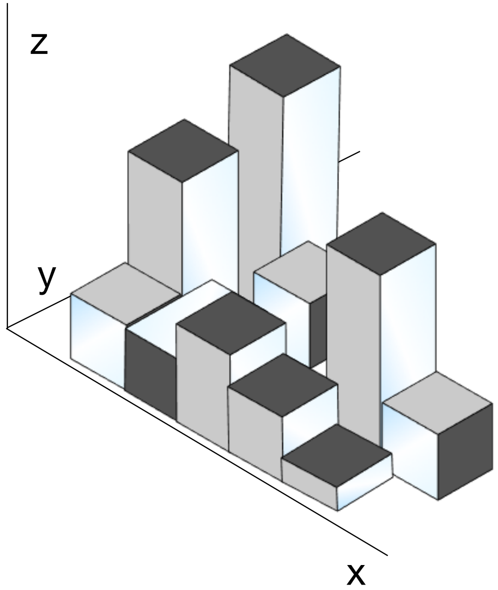
- Applications
  - BT-MZ, LU-MZ, SP-MZ
- Computational phases
  - **Communication** Period
  - **Computation** Period
  - **Time Step Loop**
    - Both periods are repeated several times according to input parameters of the application



# NPB-MZ: Data Structures and Input Sizes

- Applications operate over a mesh of ***zones***
  - **ZONE** = Set of multidimensional floating point matrices
  - Zones are in contact (e.g.: have borders, 2D surfaces)
  - Input size is identified as a CLASS
    - S, W, A, **B, C, D,** E and F

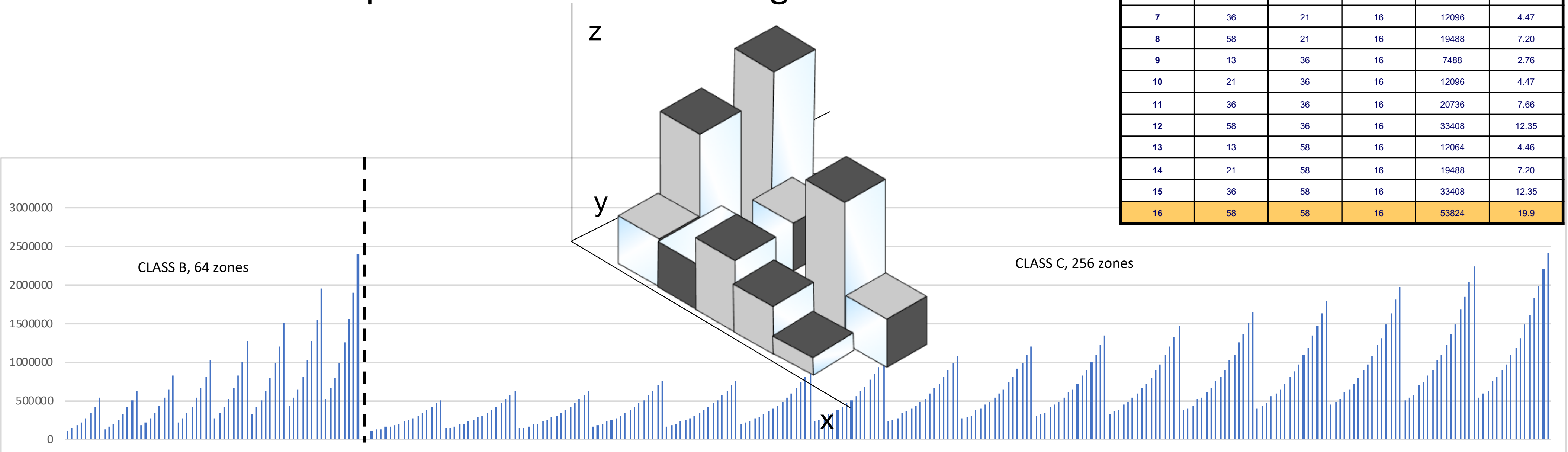
| APPLICATION | BT-MZ | | LU-MZ | | SP-MZ | | Aggregated Grid Size | Memory Requirement (approx.) |
|---|---|---|---|---|---|---|---|---|
| Problem Class | no. zones | no. iters | no. zones | no. iters | no. zones | no. iters | | |
| Class S | 2 x 2 | 60 | 4 x 4 | 50 | 2 x 2 | 100 | 24 x 24 x 6 | 1 MB |
| Class W | 4 x 4 | 200 | 4 x 4 | 300 | 4 x 4 | 400 | 64 x 64 x 8 | 6 MB |
| Class A | 4 x 4 | 200 | 4 x 4 | 250 | 4 x 4 | 400 | 128 x 128 x 16 | 50 MB |
| Class B | 8 x 8 | 200 | 4 x 4 | 250 | 8 x 8 | 400 | 304 x 208 x 17 | 200 MB |
| Class C | 16 x 16 | 200 | 4 x 4 | 250 | 16 x 16 | 400 | 480 x 320 x 28 | 0.8 GB |
| Class D | 32 x 32 | 250 | 4 x 4 | 300 | 32 x 32 | 500 | 1632 x 1216 x 34 | 12.8 GB |
| Class E | 64 x 64 | 250 | 4 x 4 | 300 | 64 x 64 | 500 | 4224 x 3456 x 92 | 250 GB |
| Class F | 128 x 128 | 250 | 4 x 4 | 300 | 128 x 128 | 500 | 12032 x 8960 x 250 | 5.0 TB |



# NPB-MZ: Zones Sizes

- For BT-MZ application
  - Zones are of different size
  - *Saw-based* pattern for zone size assignment

| Block | I-dimension | J-dimension | K-dimension | Size | Proportions |
|---|---|---|---|---|---|
| 1 | 13 | 13 | 16 | 2704 | 1 |
| 2 | 21 | 13 | 16 | 4368 | 1.61 |
| 3 | 36 | 13 | 16 | 7488 | 2.76 |
| 4 | 58 | 13 | 16 | 12064 | 4.46 |
| 5 | 13 | 21 | 16 | 4368 | 1.61 |
| 6 | 21 | 21 | 16 | 7056 | 2.61 |
| 7 | 36 | 21 | 16 | 12096 | 4.47 |
| 8 | 58 | 21 | 16 | 19488 | 7.20 |
| 9 | 13 | 36 | 16 | 7488 | 2.76 |
| 10 | 21 | 36 | 16 | 12096 | 4.47 |
| 11 | 36 | 36 | 16 | 20736 | 7.66 |
| 12 | 58 | 36 | 16 | 33408 | 12.35 |
| 13 | 13 | 58 | 16 | 12064 | 4.46 |
| 14 | 21 | 58 | 16 | 19488 | 7.20 |
| 15 | 36 | 58 | 16 | 33408 | 12.35 |
| 16 | 58 | 58 | 16 | 53824 | 19.9 |



# Outline

- Multi-Zone NAS Parallel Benchmarks (NPB-MZ)
- **NPB-MZ Multi-Gpu Parallelization**
- Performance Analysis
- Conclusions

# NPB-MZ Sources of parallelism

- Compute Period
  - **INTER** zone parallelism
    - Coarse grain parallelism
    - Parallelization of phase-zone loops

```
for (zone=1; zone<=NUM_ZONES; zone++)
   add(u[zone], rhs[zone], nx[zone], ny[zone], nz[zone]);
```

**Apply a phase to each zone**

  - **INTRA** zone parallelism
    - Fine grain parallelism
    - Parallelization of **one phase** computation over **one zone**

**Phase code**

```
void add(double *u,
         const double *rhs,
         const int nx,
         const int ny,
         const int nz) {
  int i, j, k, m;

  for (k=0; k<nz; k++)
   for (j=0; j<ny; j++)
    for (i=0; i<nx; i++)
     for (m=0; m<5; k++)
        u(m,i,j,k) += rhs(m,i,j,k);
}
```

# NPB-MZ Inter Zone Parallelization

- Inter Zone Parallelism
  - OpenMP parallel region executed by as many threads as gpus in the system
    - **One CPU controls one GPU**
  - Zones are assigned to gpus according to the scheduler

- *SCHEDULING::schedTasks*
- *RUNTIME::commit-task*
- *RUNTIME::get-task*

**INTER-ZONE PARALLELISM**

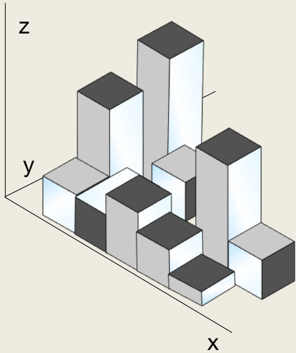
**Phase Code**

```
void MAIN-APPL ()
{
  for time = 1, num-time-steps

    for zone = 1, num-zones
      phase-zone(arguments[zone]…)

    for zone = 1, num-zones
      phase-zone(arguments[zone]…)

    ......

  verify();
}
```

**Code Transformation**

```
SCHEDULING::schedTasks(num-zones,
                       CUDA::num-gpus);

#pragma omp parallel
{
  task = RUNTIME::get-task();
  while (task!=NO-TASK) {
    zone = task;
    phase-zone( arguments[zone] … );
    RUNTIME::commit-task(task);

    task = RUNTIME::get-task();
  }
}
CUDA::synchDevices();
```

**Data Structures**



# NPB-MZ Work Scheduling and Zone Placement

- Inter Zone Parallelism
  - **Schedulers** determine **work balance** at the INTER zone parallelism
  - **Schedulers** determine the zone **placement** and **memory allocation** for zones

- *SCHEDULING::schedTasks*
- *RUNTIME::commit-task*
- *RUNTIME::get-task*

**INTER-ZONE PARALLELISM**

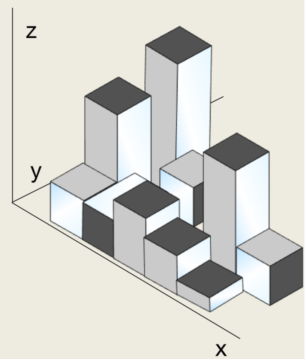
**Phase Code**

```
void MAIN-APPL ()
{
  for time = 1, num-time-steps

    for zone = 1, num-zones
      phase-zone(arguments[zone]…)


    for zone = 1, num-zones
      phase-zone(arguments[zone]…)

    ......

  verify();
}
```

**Code Transformation**

```
SCHEDULING::schedTasks(num-zones,
                       CUDA::num-gpus);

#pragma omp parallel
{
  task = RUNTIME::get-task();
  while (task!=NO-TASK) {
    zone = task;
    phase-zone( arguments[zone] … );
    RUNTIME::commit-task(task);

    task = RUNTIME::get-task();
  }
}
CUDA::synchDevices();
```

**Data Structures**



# NPB-MZ Work Scheduling: STATIC and DYNAMIC

- Zones are assigned to gpus according to the schedulers:
  - STATIC
    - Zones are evenly distributed across the gpus
  - DYNAMIC
    - Zones are dynamically assigned to gpus in chunks
      - Non-memorizing
        - Scheduler does not recall the zone assignment in the latest instance of the time step loop
      - Memorizing
        - Scheduler captures first assignment occurred in the first instance of the time step loop
        - Scheduler keeps constant the zone distribution
      - CHUNK = 1, 2, 4, 8 and 16

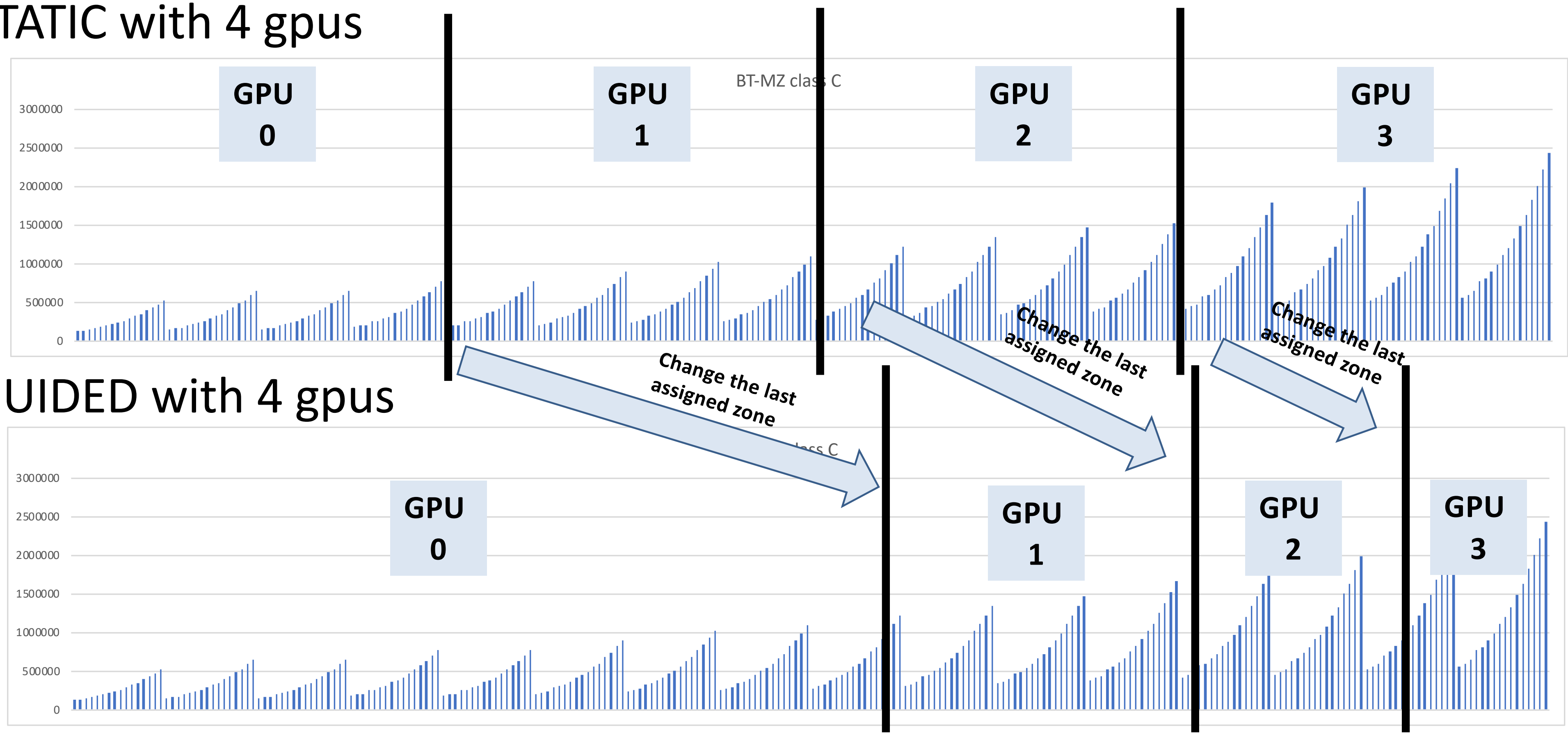
# NPB-MZ Work Scheduling: GUIDED

- GUIDED: Tries to balance the work distribution given some information that describes the work unbalance
  - Starts with a STATIC distribution
  - Monitors what is the computational cost per each zone
    - Estimates what is the total amount of computation (TOTAL = add computational cost of all zones)
    - Estimates what is the work per gpus
      - WORK-GPU = TOTAL / NGPUS
  - Reassigns consecutive zones but the number of zones is different per each gpu according to the
  - Two variants
    - Guided by the sizes of zones (**GUIDED-SIZES**)
    - Guided by the runtime execution times of zones (**GUIDED-RUNTIME**)

```
INPUT: WorkDonePerGpu. WorkPerZone, IndexFirstZone, IndexLastZone
OUTPUT: IndexFirstZone, IndexLastZone

for gpu=0, num-gpus-1
    TotalWork += WorkDonePerGpu[gpu];

WorkPerGpu = TotalWork/num-gpus;

for gpu =0, num-gpus-2

        Work = WorkDonePerGpu[gpu]
        First = IndexFirstZone[gpu]
        Last = IndexLastZone[gpu]
        if (Work<WorkPerGpu)
            while (Work<WorkPerGpu)
                diff1 = WorkPerGpu-Work;
                WorkZone = WorkPerZone[Last+1];
                diff2 = abs(WorkPerGpu-(Work+WorkZone));
                if (diff2<diff1)
                    Last++;
                    Work += WorkZone;

                else break;
                if (Last==num-zones-2) break;

        else if (Work>WorkPerGPU && First!=Last)
            while (Work>WorkPerGPU && First!=Last)
                diff1 = abs(WorkPerGpu-Work);
                WorkZone = WorkPerZone[Last];
                diff2 = abs(WorkPerGpu-(Work-WorkZone));
                if (diff2<diff1)
                    Work-=WorkTask;
                    Last--;

                else break;
                if (Last==0) break;

        IndexLastZone[gpu] = Last;
        IndexFirstZone[gpu+1] = Last+1;
        if (IndexFirstZone[gpu+1] > IndexLastZone[gpu+1])
            IndexLastZone[gpu+1] = IndexFirstZone[gpu+1];
```

# NPB-MZ Work Scheduling: GUIDED

- STATIC with 4 gpus
- GUIDED with 4 gpus



# NPB-MZ Sources of parallelism

- Intra zone parallelism

```
void add(double *u,
         const double *rhs,
         const int nx,
         const int ny,
         const int nz) {
  int i, j, k, m;

  for (k=0; k<nz; k++)
   for (j=0; j<ny; j++)
    for (i=0; i<nx; i++)
     for (m=0; m<5; k++)
       u(m,i,j,k) += rhs(m,i,j,k);
}
```

**Kernel Definition**

```
__global__ static void add_kernel (double *u,
                                   const double *rhs,
                                   const int nx,
                                   const int ny,
                                   const int nz) {
         int i, j, k, m;

         k = blockIdx.y+1;
         j = blockIdx.x+1;
         i = threadIdx.x+1;
         m = threadIdx.y;

         u(m,i,j,k) += rhs(m,i,j,k);
}
```

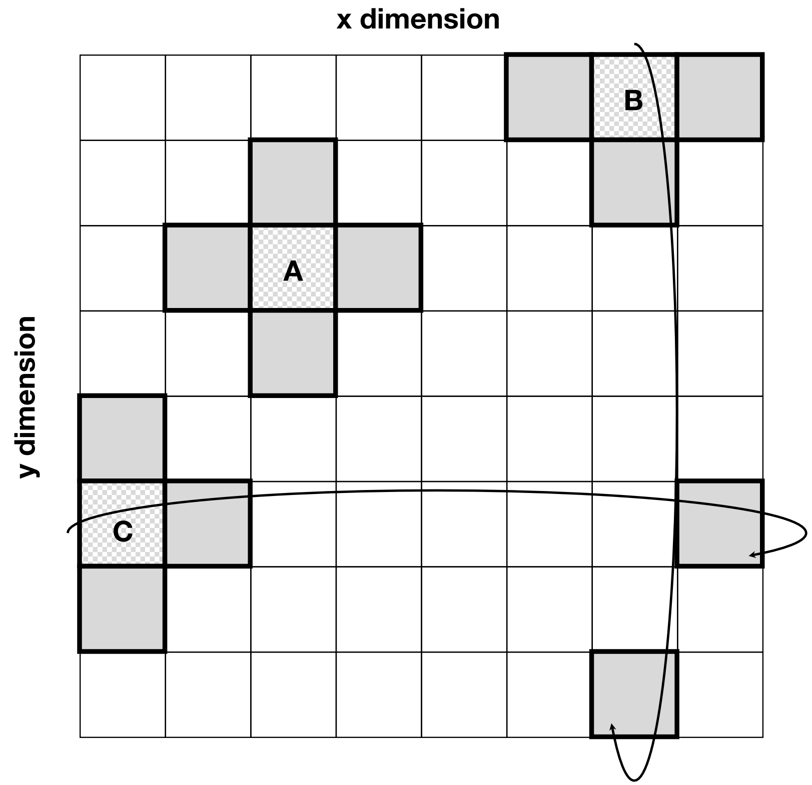
**Kernel Invocation**

```
void add(double *u,
         const double *rhs,
         const int nx,
         const int ny,
         const int nz) {

  add_kernel<<<grid,block>>>(u, rhs, nx, ny, nz);

}
```

# NPB-MZ Zone Adjacency

- Zones have contact with 4 bordering zones
  - 4 x 2D surfaces
    - north
    - south
    - east
    - west

- Zones at the edges are adjacent from one edge to the other



# NPB-MZ Sources of parallelism

- Communication Period
  - There is **not INTER** zone parallelism
    - Exchange boundary values is a sequential computation over zones
  - **INTRA** zone parallelism
    - Fine grain parallelism
    - Parallelization of **one phase** computation over **one zone**

CPU BOUNDARY-EXCHANGE CODE

```
for zone =1, num-zones

  east-zone = adjacency-east[zone] // east adjancent zone
  north-zone = adjacency-north[zone] // north adjacent zone
  copy-face(tmpEast, mesh[east-zone])
  copy-face(tmpNorth, mesh[north-zone])

  compute-border(mesh[zone], tmpEast, tmpNorth)

  copy-face(mesh[east-zone, tmpEast)
  copy-face(mesh[north-zone], tmpNorth)
```

SINGLE GPU BOUNDARY-EXCHANGE CODE

```
for zone =1, num-zones

  east-zone = adjacency-east[zone] // east adjancent zone
  north-zone = adjacency-north[zone] // north adjacent zone

  CUDA::compute-border<<< grid, block, shared >>>(mesh[zone],
                                                  mesh[east-zone],
                                                  mesh[north-zone])
```

MULTI-GPU BOUNDARY-EXCHANGE CODE

```
for zone =1, num-zones

  east-zone = adjacency-east[zone] // east adjancent zone
  north-zone = adjacency-north[zone] // north adjacent zone
  east-gpu = zone-gpu-mapping[east-zone] // GPU where the east zone is placed
  north-gpu = zone-gpu-mapping[north-zone] // GPU where the north zone is placed
  zone-gpu = zone-gpu-mapping[zone] // GPU where current zone is placed

  CUDA::copy-zone(east-gpu, mesh[east-zone], zone-gpu, tmp1)
  CUDA::copy-zone(north-gpu, mesh[east-zone], zone-gpu, tmp2)

  CUDA::compute-border<<< grid, block, shared >>>(mesh[zone],
                                                  tmp1,
                                                  tmp2)

  CUDA::copy-zone(zone-gpu, tmp1, east-gpu, mesh[east-zone])
  CUDA::copy-zone(zone-gpu, tmp2, east-gpu, mesh[north-zone])
```

# Outline

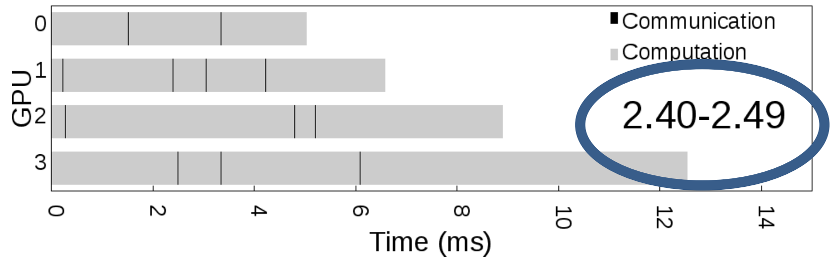
- Multi-Zone NAS Parallel Benchmarks (NPB-MZ)
- NPB-MZ Multi-Gpu Parallelization
- **Performance Analysis**
- Conclusions

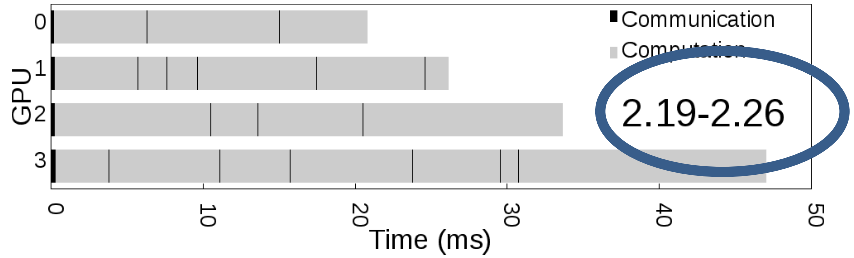
# Multi-Gpu Computing System

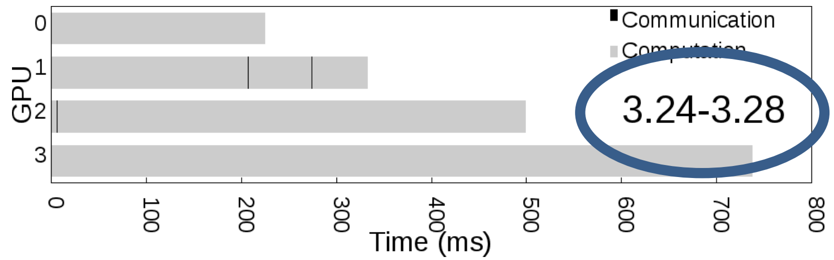
- Architecture
  - 2 x IBM Power9 8335-GTH @ 2.4GHz
    - 20 cores and 4 threads/core, total 160 threads
  - 512GB of main memory (16 x 32GB DIMMS @ 2666MHz)
  - **4 x GPU NVIDIA V100 (Volta) with 16GB HBM2**
- Software Environment
  - Red Hat Enterprise Linux Server 7.5
  - **CUDA 10.1 compiler, CUDA 418.39 driver**
  - GCC 4.8.5.

# BT-MZ Multi-Gpu Computation Period

- Computation Period
  - STATIC, Class B
    - Work unbalance ranges 2,40
  - STATIC, Class C
    - Work unbalance ranges 2,19
  - STATIC, Class D
    - Work unbalance ranges 3,24

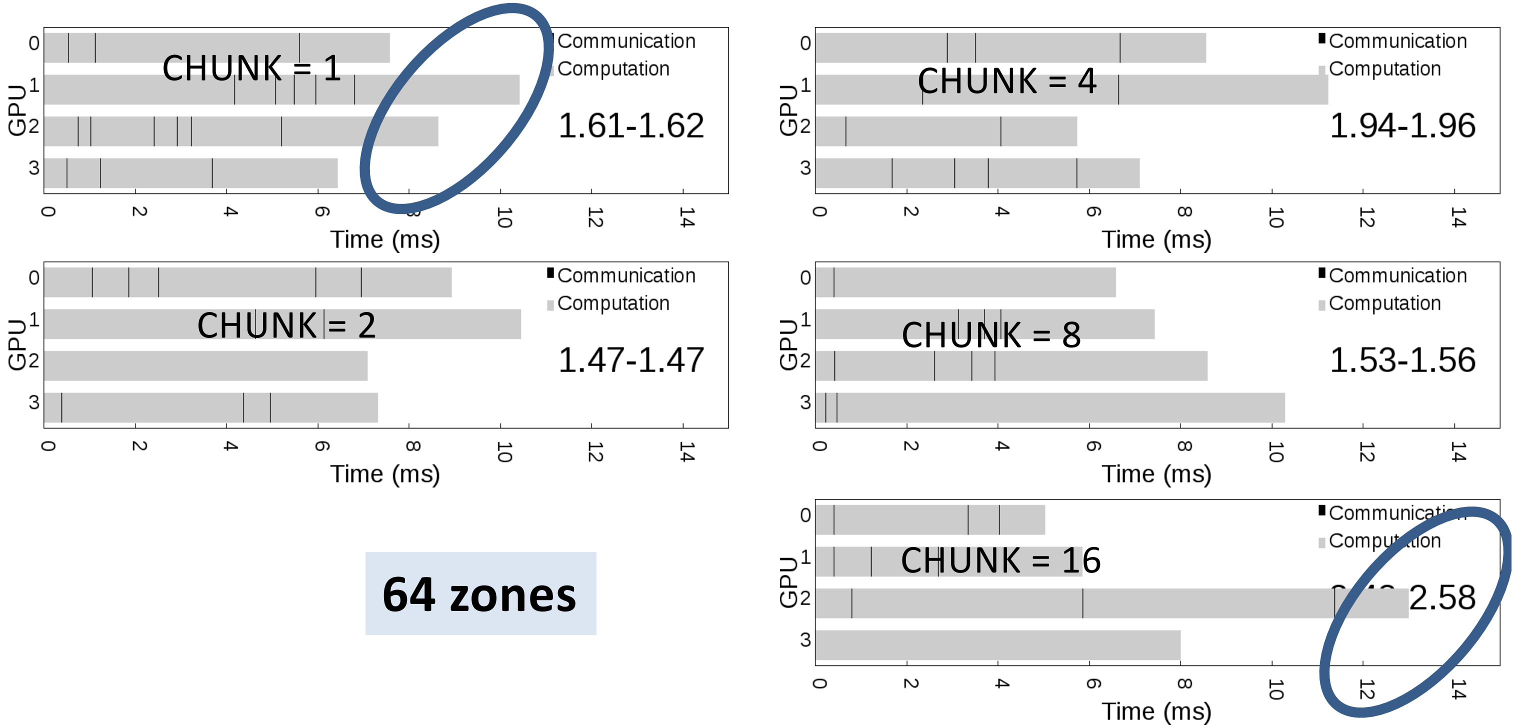


# BT-MZ Multi-Gpu Computation Period

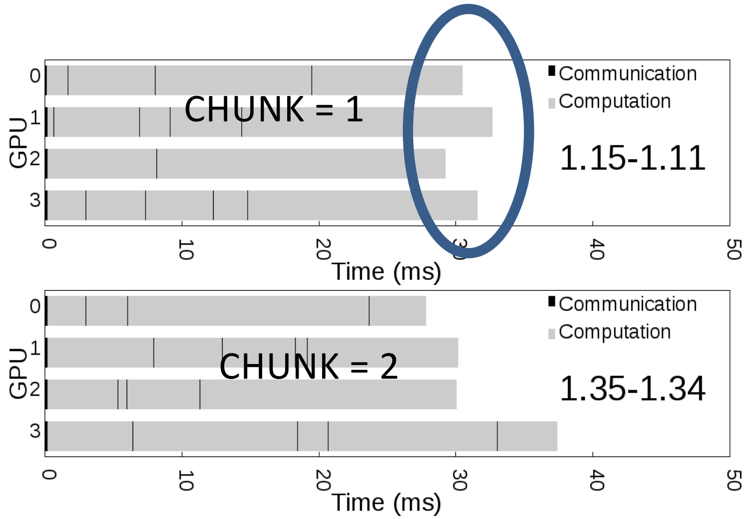
- Computation Period
  - DYNAMIC
    - Chunk = 1, 2, 4, 8, 16
    - CLASS B
    - Bigger chunk, worst work balance

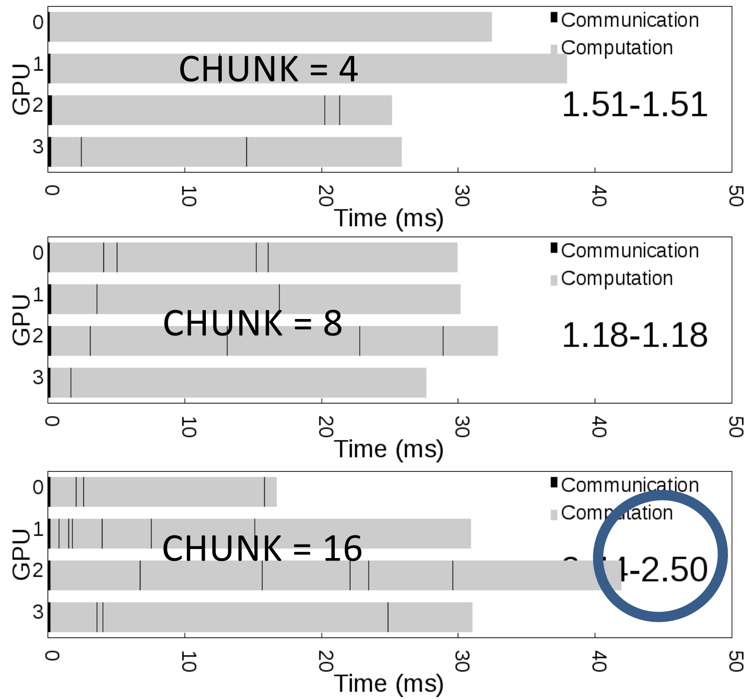



**64 zones**

# BT-MZ Multi-Gpu Computation Period

- Computation Period
  - DYNAMIC
    - Chunk = 1, 2, 4, 8, 16
    - CLASS C
    - Bigger chunk, worst work balance

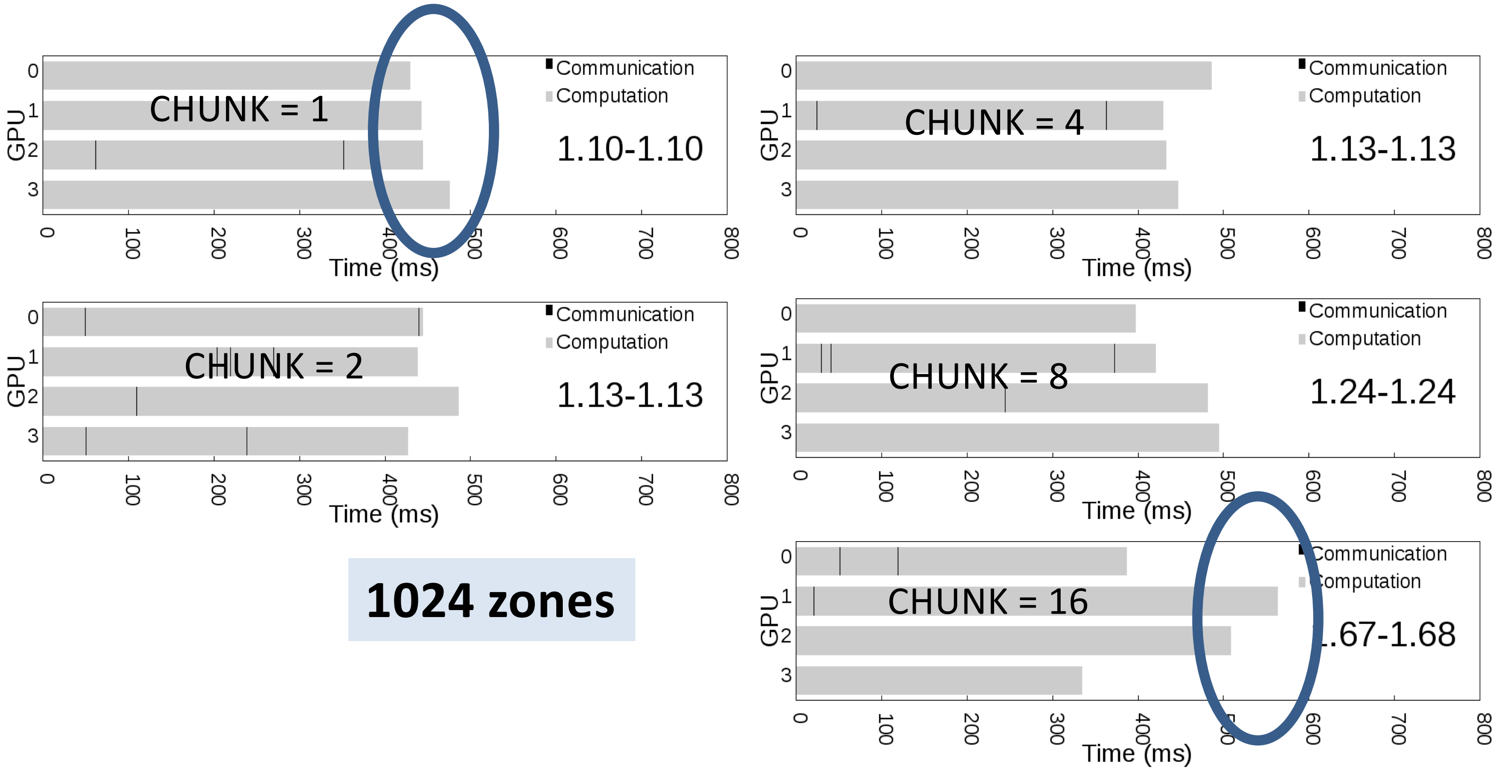



**256 zones**

# BT-MZ Multi-Gpu Computation Period

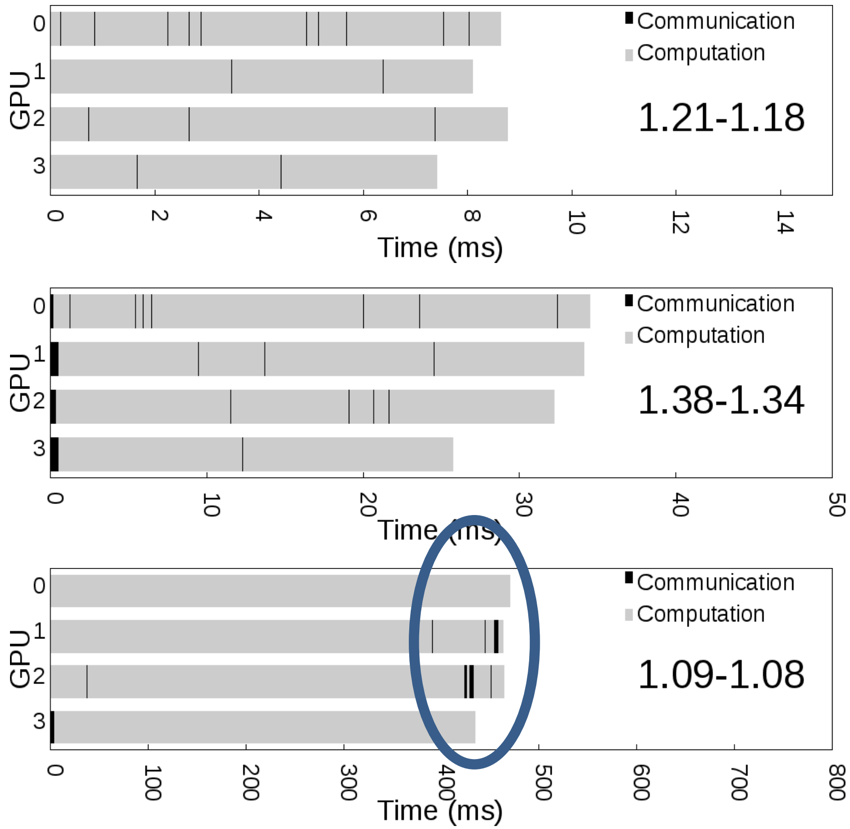
- Computation Period
  - DYNAMIC
    - Chunk = 1, 2, 4, 8, 16
    - CLASS D
    - Bigger chunk, worst work balance




**1024 zones**

# BT-MZ Multi-Gpu Computation Period

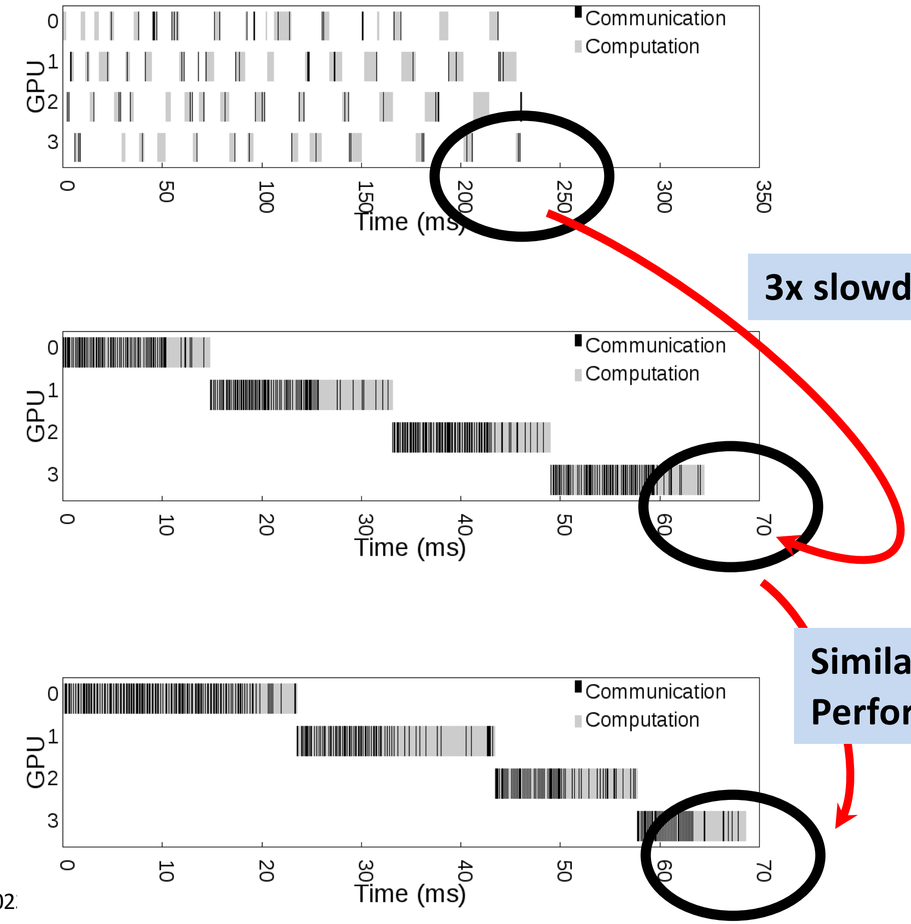
- Computation Period, GUIDED-RUNTIME
  - CLASS B
    - Work unbalance ranges 1,21
  - CLASS C
    - Work unbalance ranges 1,38
  - CLASS D
    - Work unbalance ranges 1,09



# BT-MZ Multi-Gpu Communication Period

- DYNAMIC
  - Sequential bursts of computation (compute-borders) are dynamically distributed across the gpus
- STATIC
  - Sequential bursts of computation (compute-borders) are evenly distributed across the gpus
- GUIDED-RUNTIME
  - Sequential bursts of computation (compute-borders) are NOT evenly distributed across the gpus.

**Work distribution in Computation Period determines zone placement for Communication Period!!**

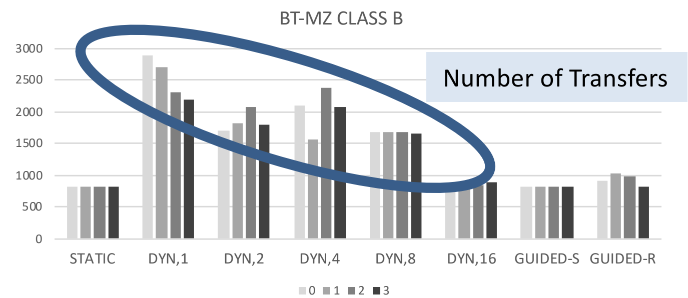


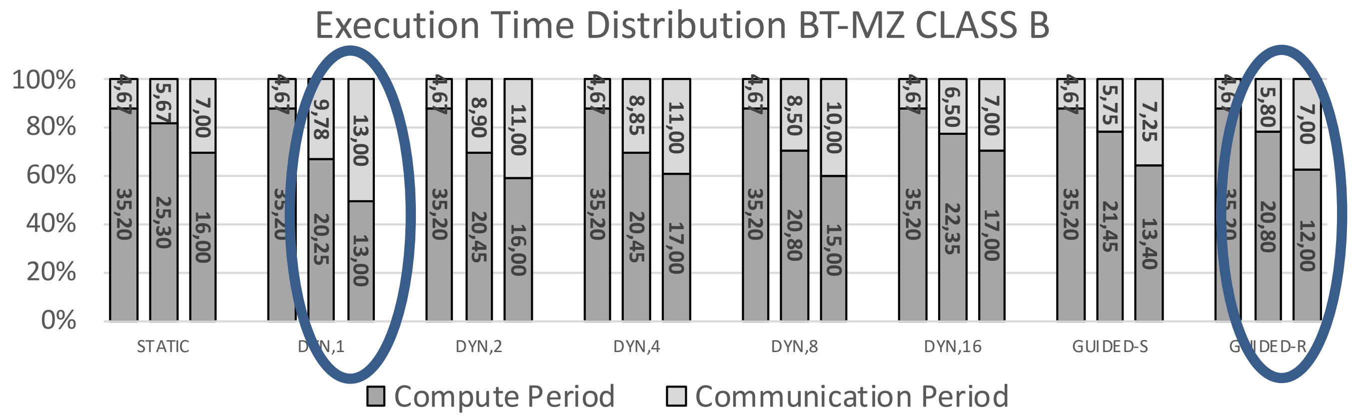

**3x slowdown!!**

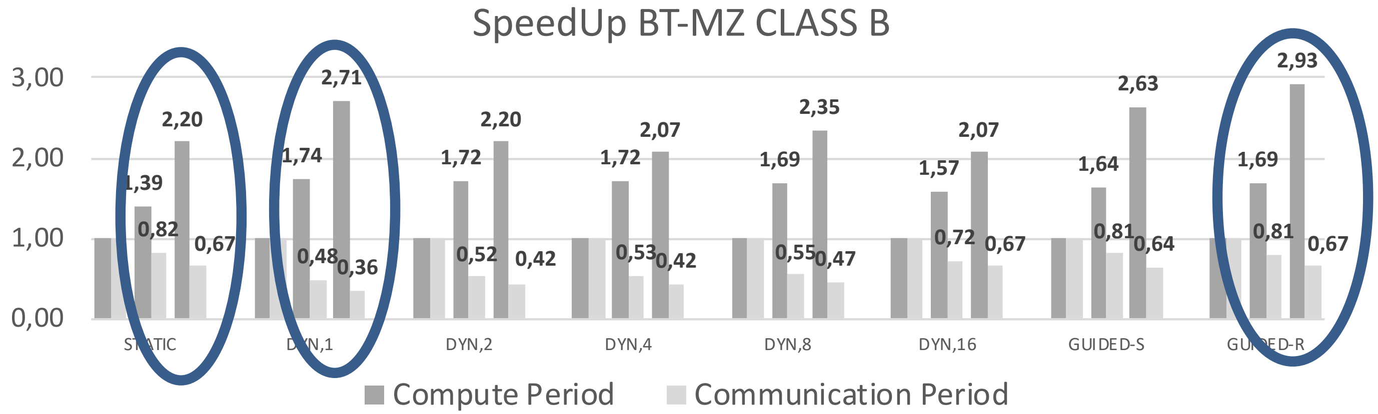
**Similar Performance**

# Results: BT-MZ, Class B, Computation vs Communication Periods

- STATIC
  - Comp Period **does not scale**
  - Comm Period is **not drastically slowed down**
- DYNAMIC
  - Comp Period **scales well**
  - Comm Period is **drastically slowed down**
- GUIDED
  - Comp Period **scales well**
  - Comm Period is **not drastically slowed down**

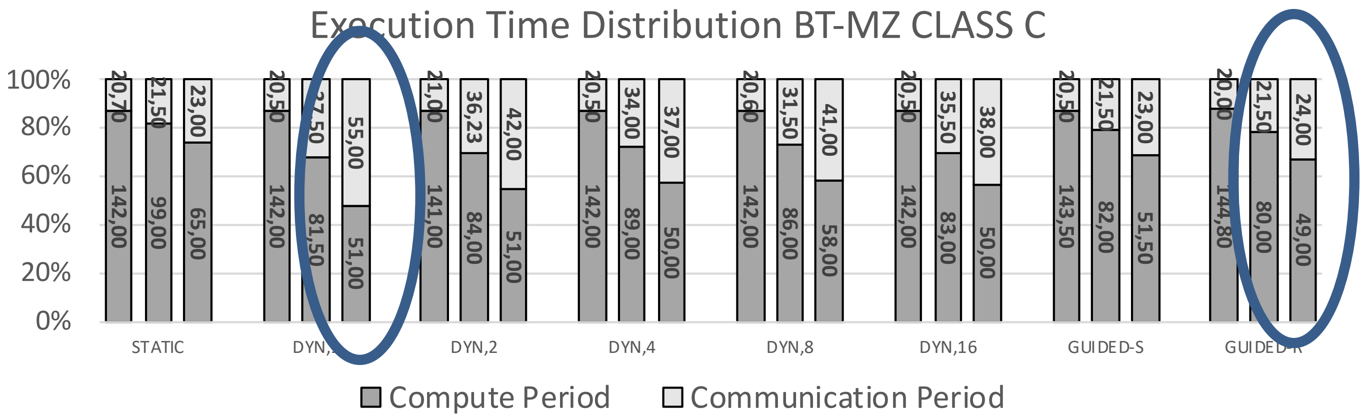


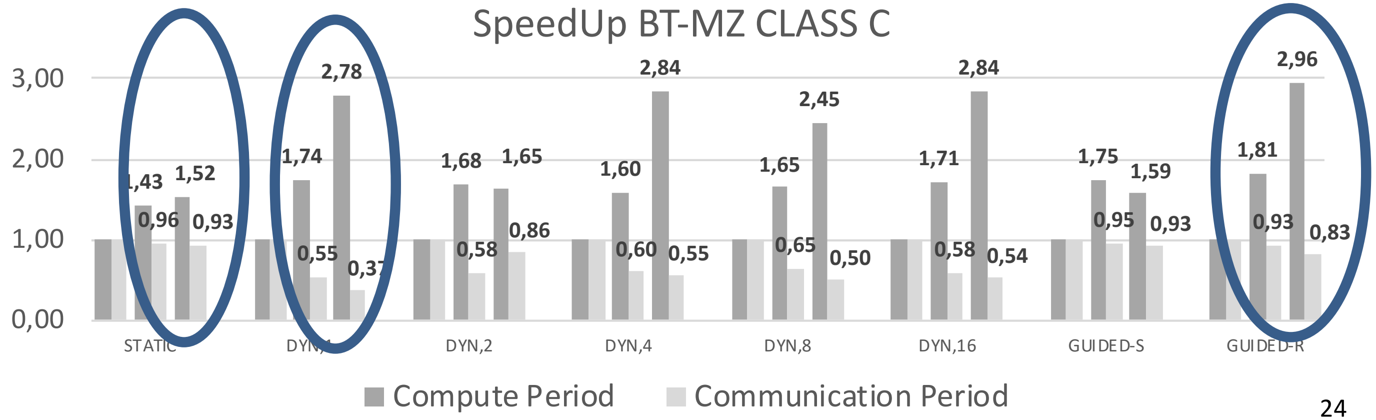



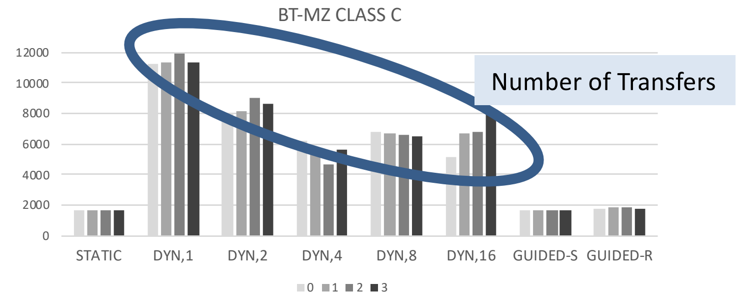



# Results: BT-MZ, Class C, Computation vs Communication Periods

- STATIC
  - Comp Period **does not scale**
  - Comm Period is **not drastically slowed down**
- DYNAMIC
  - Comp Period **scales well**
  - Comm Period is **drastically slowed down**
- GUIDED
  - Comp Period **scales well**
  - Comm Period is **not drastically slowed down**

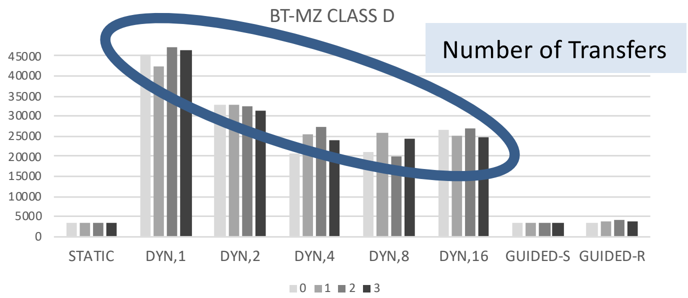


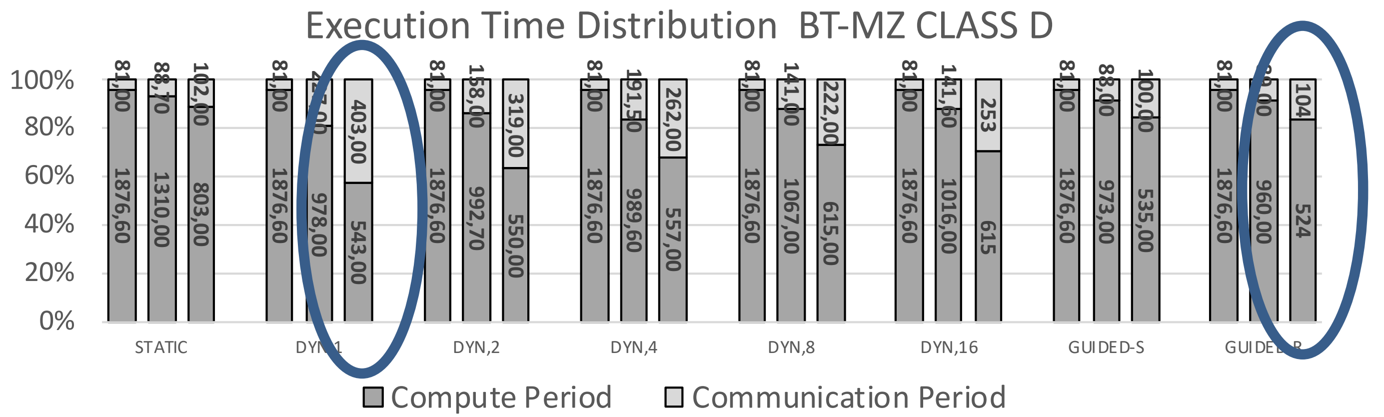



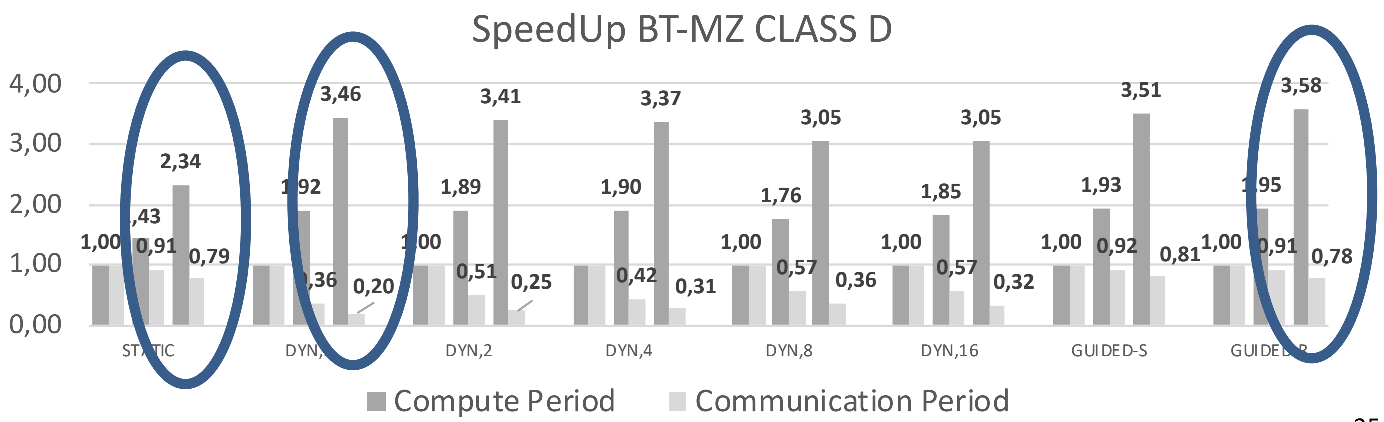



# Results: BT-MZ, Class D, Computation vs Communication Periods

- STATIC
  - Comp Period **does not scale**
  - Comm Period is **not drastically slowed down**
- DYNAMIC
  - Comp Period **scales well**
  - Comm Period is **drastically slowed down**
- GUIDED
  - Comp Period **scales well**
  - Comm Period is **not drastically slowed down**

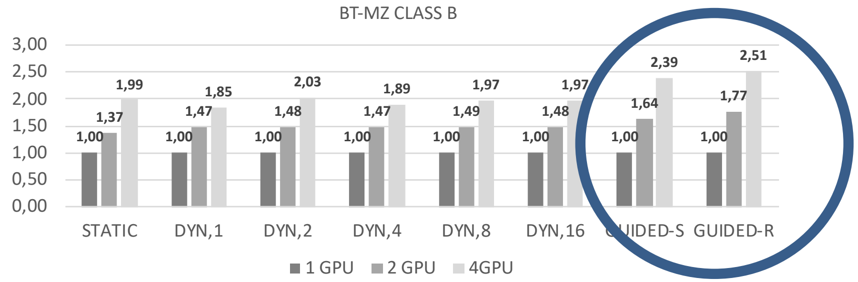


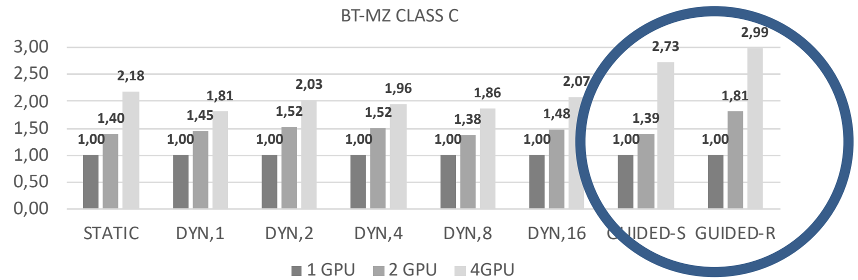



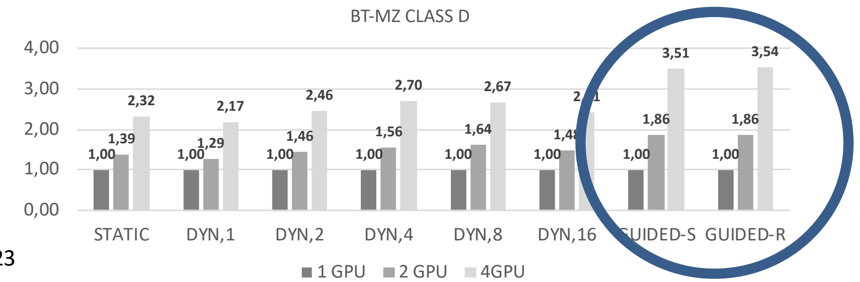



# Results: BT-MZ, Overall Speedup

- STATIC
  - Speedup range between 1,37-2,32 with 2 and 4 GPUS
- DYNAMIC
  - Speedup range between 1,37-2,67 with 2 and 4 GPUS
- GUIDED
  - Speedup range between 1,39-3,54 with 2 and 4 GPUS

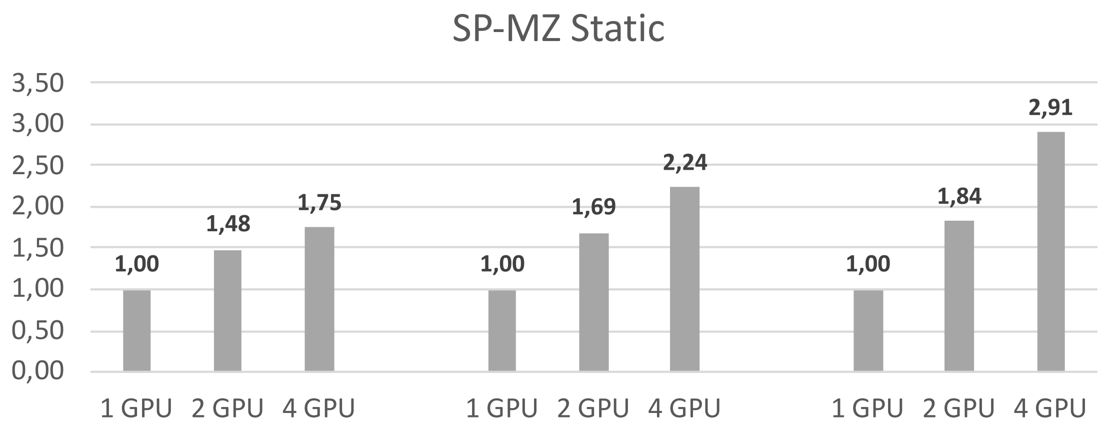


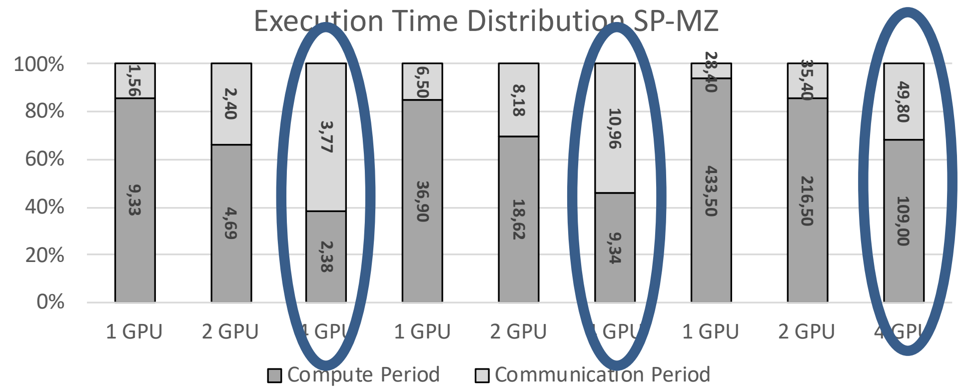



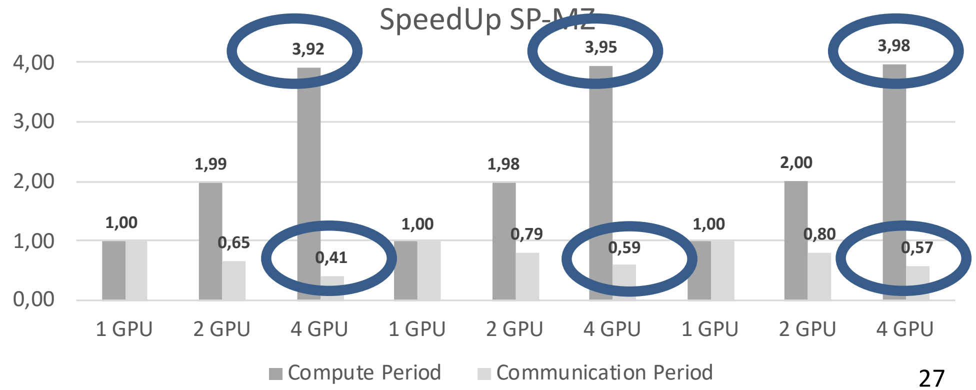



# Results: SP-MZ

- STATIC
  - Relation between Comp./Comm. Periods
    - Slow down in Communication Perdiod
    - Communication Period becomes a bottleneck
  - Speedup rages between 1,48-2,91

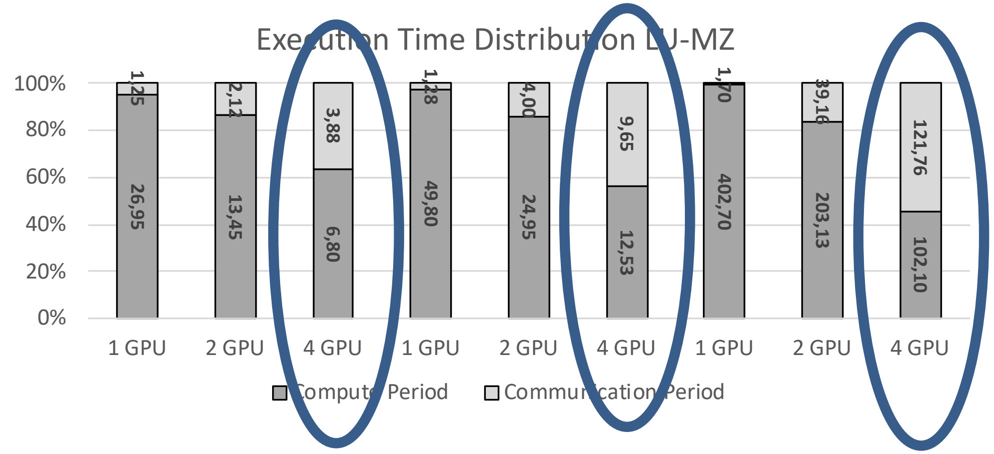


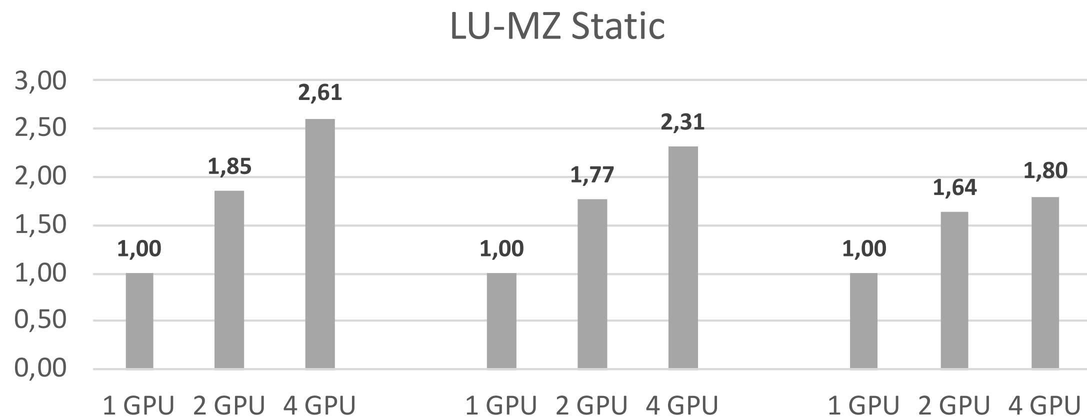



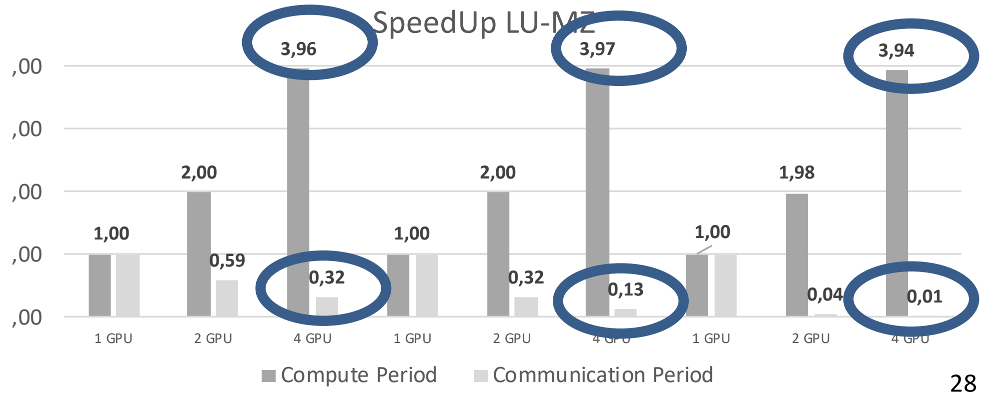



# Results: LU-MZ

- STATIC
  - Relation between Comp./Comm. Periods
    - Slow down in Communication Perdiod
    - Communication Period becomes a bottleneck
  - Speedup rages between 1,64-2,61



# Outline

- Multi-Zone NAS Parallel Benchmarks (NPB-MZ)
- NPB-MZ Multi-Gpu Parallelization
- Performance Analysis
- **Conclusions**

# Conclusions For Multi-GPU Execution

- **Work distribution** schemes are essential for multi-gpu parallelization
  - They affect **work unbalance**
  - They determine **data placement**
- **Communication** phases are heavily affected by the **data placement**
  - Multiple GPUs eventually exchange data, in contrast to single GPU execution where data is stored in a single device.
  - These **additional communication** can slowdown the overall execution becoming a bottleneck for performance
- **Tradeoff** between **computation** and **communication** phases
  - Speedup in computation phases can be totally irrelevant if communication phases suffer critical performance degradation

# Multi-Gpu Parallelization of Scientific Computing Applications: the Case of the NAS Multi-Zone Parallel Benchmarks

Marc Gonzalez Tallada

Associate Professor

Computer Architecture Department

Technical University of Catalonia