Load Balanced large 3D FFTs on a Heterogeneous Cluster

Jakob Siegel

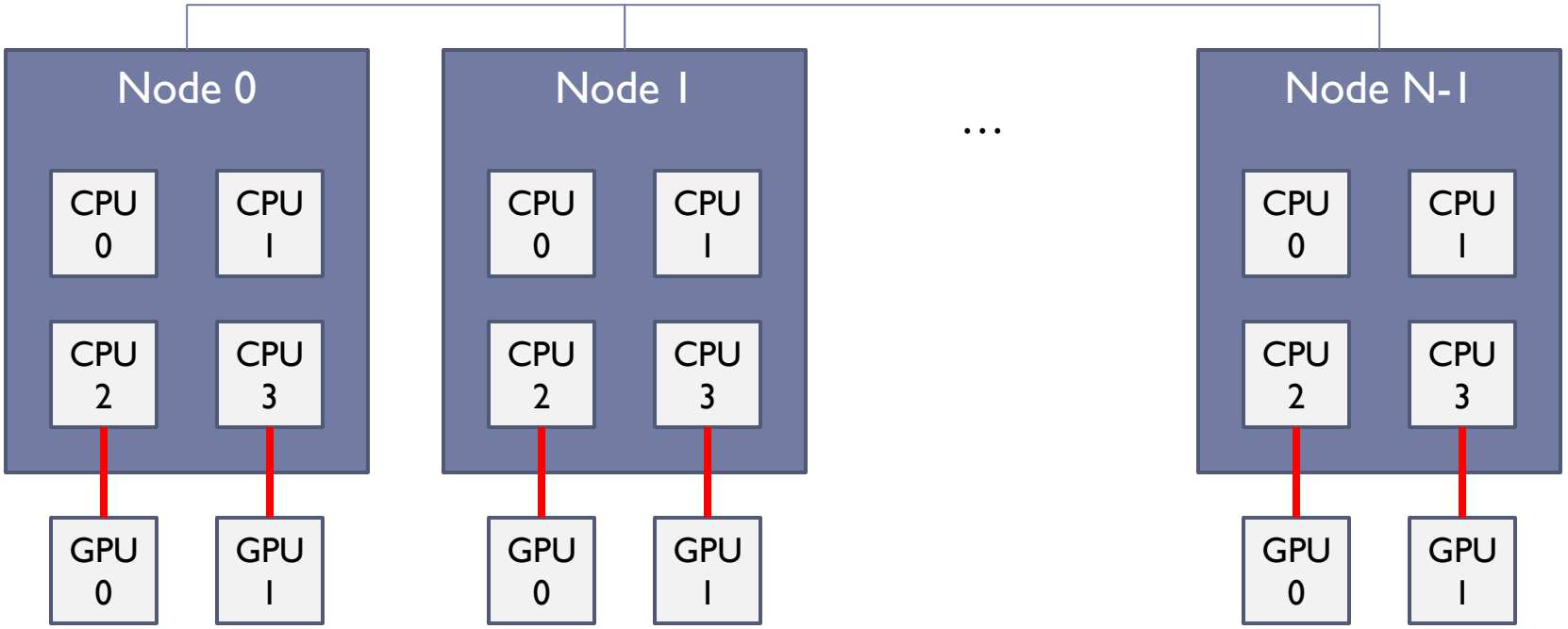
How can FFT be Imbalanced?

- Different communication/copy times across the cluster
- Using different compute elements CPU, GPU, Cell… available in the Cluster
- Allowing for any number of compute elements (some will have to do more or less work than others)
- Resource failure

- → Using a tasking approach could handle those issues and allows overlapping of different stages of the computation

Heterogeneous Program Execution

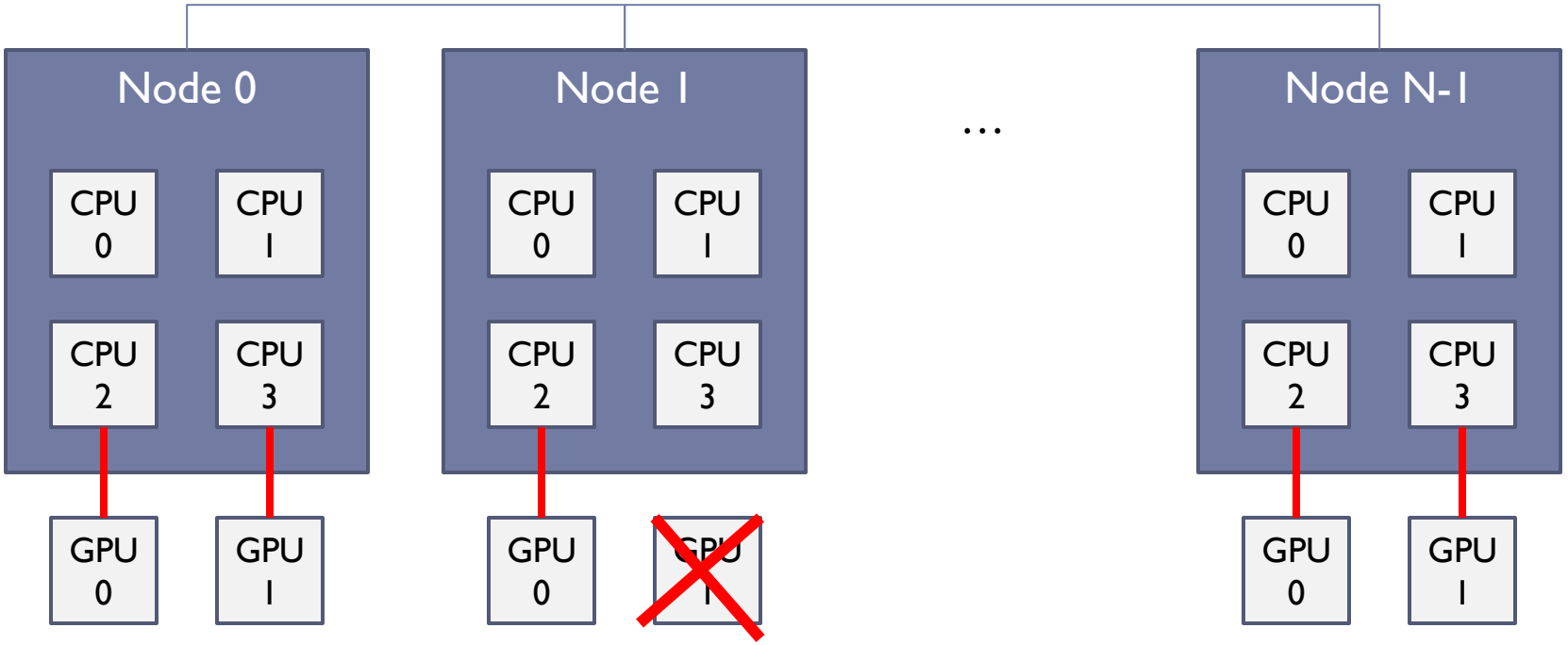
Heterogeneous Cluster: Usually some cores in each node drive an accelerator e.g. GPU or Cell



If a program that on a homogeneous system is balance gets executed on a cluster using the accelerators, the execution inside the nodes becomes imbalanced. Accelerator in general complete assigned work much faster.

Heterogeneous Program Execution

Hard to predict factors like outages or resource failures can introduce a load imbalance across the nodes



Because of their complexity large HPC clusters tend to have outages of some resources, therefore some nodes might be more efficient than others

What is Task based execution

- Computation is grouped into small tasks, where each task depends on a set of independent inputs and generates defined outputs
- Tasks are not "assigned" to compute elements in advance, the thread driving the compute element polls a task when computational power is available.
- A task has to be small enough to be executed on any given compute element and, if practical, to allow double buffering
- New tasks/follow up tasks can be generated as soon as the inputs are available (see next slides)

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

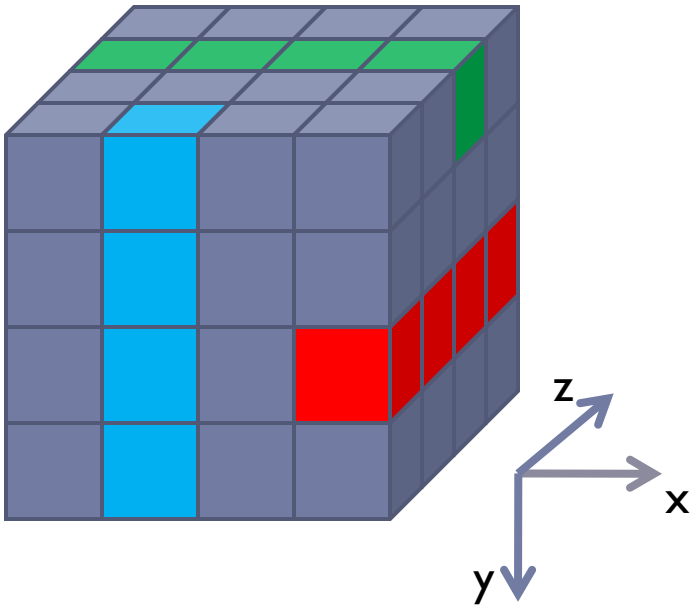
Input data serialized in x,y,z order.

Data is access for 3x 1D FFT

Z direction: stride of width X * height Y

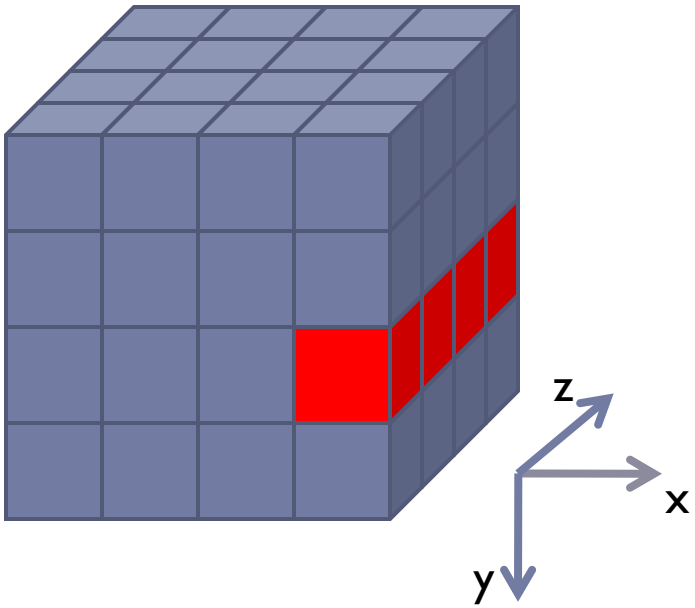
Y direction: stride of width X

X direction: no stride



Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

The most expensive access is the strided single element access in Z-direction, where the elements are the furthest spaced out and most probably stored in different nodes all over the cluster.



In this simple example we have 3x16 tasks.

Assume the system has 7 processor cores working on the problem

The task execution starts with the most expensive tasks: calculating the Z-direction FFT, followed by the Y-direction and last the X-direction which might even be calculated locally without communication.

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

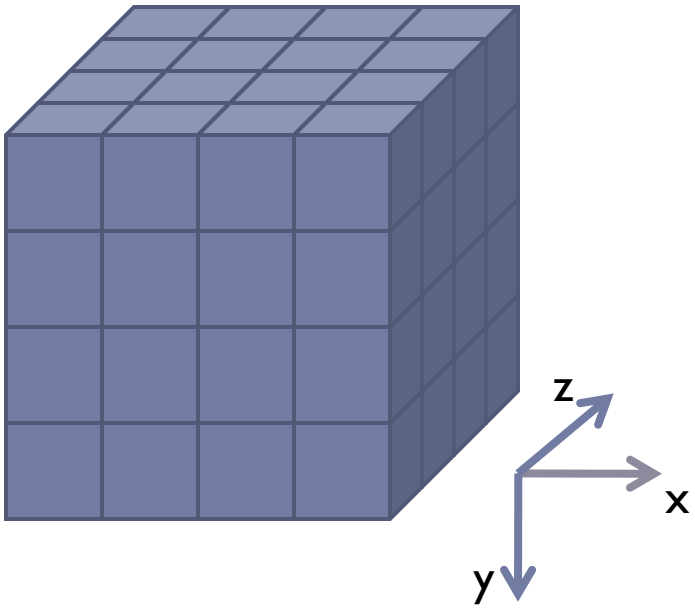
Global Task Pool

z0 z1 z2 z3 z4 z5

z6 z7 z8 z9 z10 z11

z12 z13 z14 z15

Before the first step the Global Task Pool is filled only with Z-direction tasks.
When the execution starts each core will poll a task.



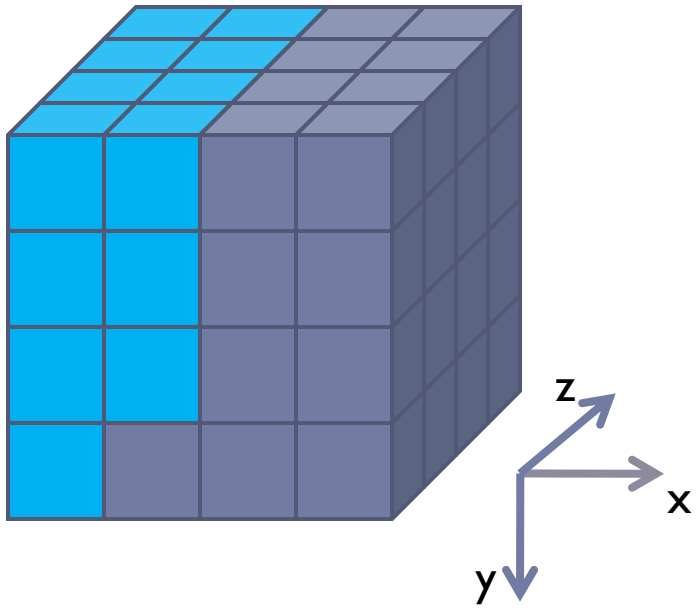
Input

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

Global Task Pool

z7 z8 z9 z10 z11
z12 z13 z14 z15 y0 y1
y2 y3

When the first 7 tasks are completed, the first 4 Y-direction tasks are ready to be executed.



Input Z complete

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

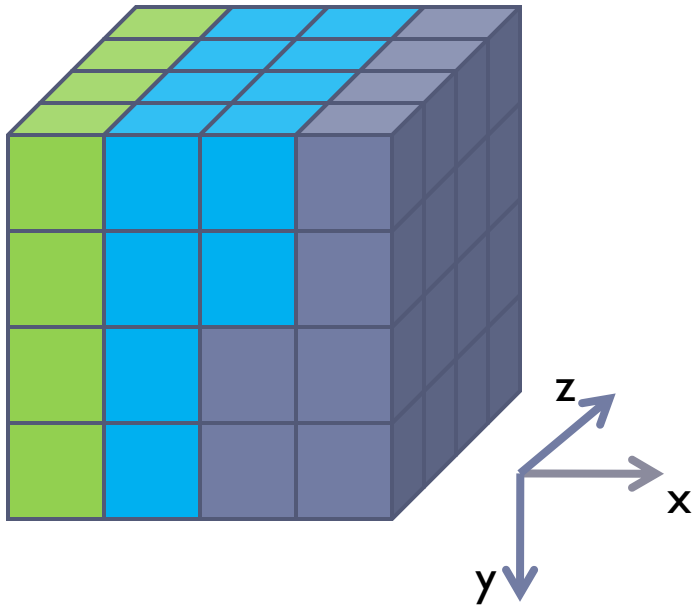
Global Task Pool

z10 z11

z12 z13 z14 z15

y4 y5 y6 y7

Less communication intense tasks will have a slightly higher probability to be executed when available. Therefore Y-direction tasks will be executed even if there are still Z-direction tasks in the pool.



Input Z complete Z&Y complete

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

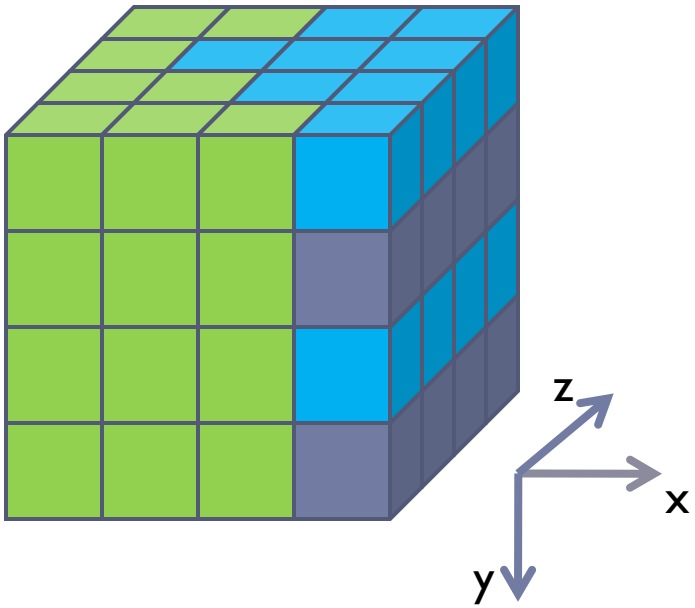
Global Task Pool

z13 z15

y6

y9 y10 y11

This might reduce the pressure on the interconnect in some regions since the Y- tasks are not communicating "as far" across the cluster as the Z-tasks. The more tasks are executed, the more random the order of completion will look.



Input Z complete Z&Y complete

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

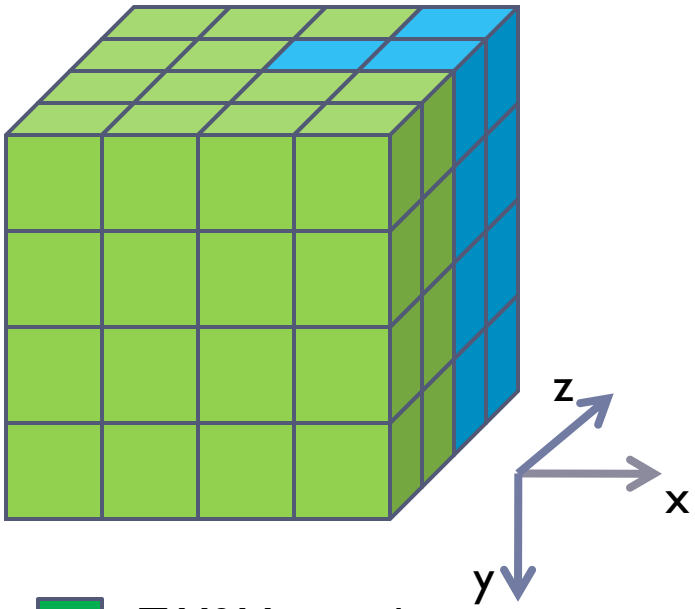
Global Task Pool

y10

y14 y15 x1 x2 x3 x4

x5 x6 x7 x8

When the first 2 FFTs are completed for any XY plane, the X-direction tasks will be available and can be executed with the least or in some cases no interconnect communication at all.



Input Z complete Z&Y complete Z,Y&X complete

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

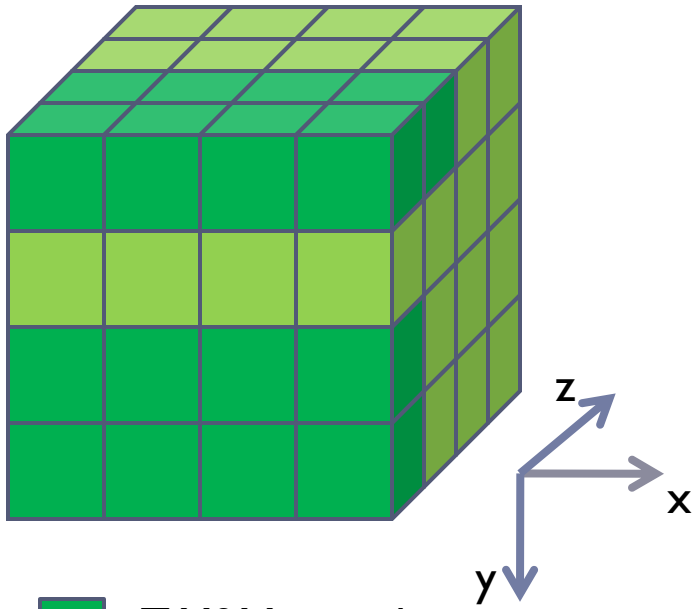
Global Task Pool

x1

x5 x6 x7 x8 x9

x10 x11 x12 x13 x14 x15

In the end there will be only low communication tasks left which will reduce the overall imbalance to the cost of one of those lightest tasks.

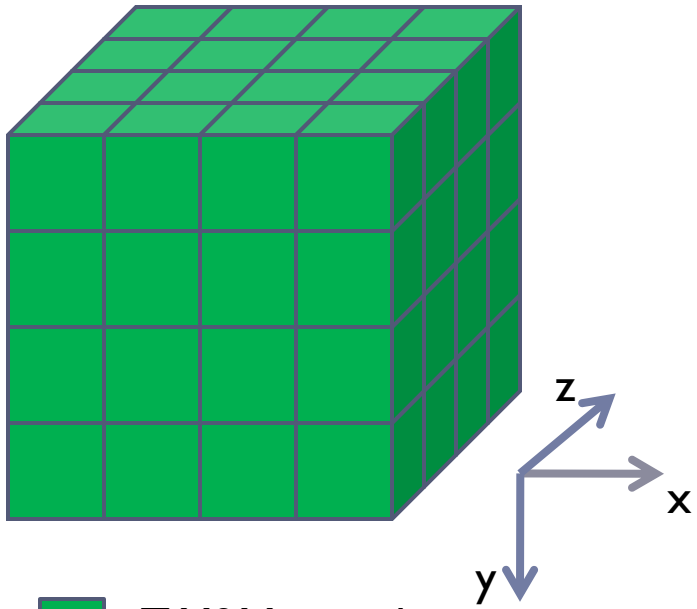


Input Z complete Z&Y complete Z,Y&X complete

Large FFT with many fine-grain tasks to overlap the execution of the 3 1D FFTs

Global Task Pool

When the task pool is empty and all calculations are completed only a minimum amount of communication is needed since the final tasks are mostly working on local data and if not the results can be communicated with a single copy.



Input Z complete Z&Y complete Z,Y&X complete

Conclusion

- By applying a fine grain tasking approach to the FFT calculation it might be possible to:
  - Prevent communication bottlenecks when in an evenly distributed computation all threads perform the same kind of data transfers simultaneously
  - Introduce a more random execution pattern, where low cost tasks can be overlapped with the execution/communication of higher cost tasks
  - Limit the maximum possible imbalance is the execution time of the task that is polled last, which is one of the lowest cost tasks

Conclusion (cont.)

- Imbalances that might be introduced by the topology of the cluster are automatically handled
  - Faulty resources
  - Non exclusive usage of the cluster
    (other applications occupying resources during the execution)
  - Uneven interconnect speeds and local bottlenecks

- heterogeneous execution when using accelerators automatically gets balanced even that the number of tasks per core/accelerator might differ greatly

Next Steps

- Implementation of the task based FFT calculation on a cluster
- comparison of the static to the tasking approach
- Heterogeneous execution using CPUs and GPUs simultaneously
- Collecting empiric data to define most efficient task granularities