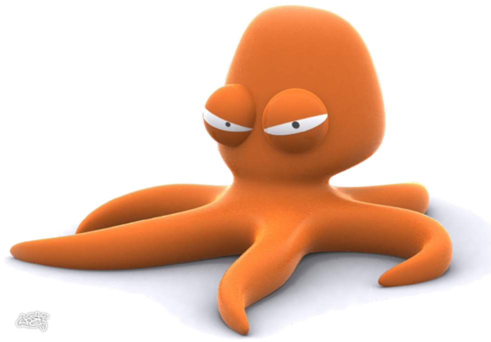
Octopus: Recent advances in GPU development

Sebastian Ohlmann
Max Planck Computing and Data Facility, Garching
In collaboration with H. Appel, M. Lüders, M. Oliveira, N. Tancogne-Dejean

APS March meeting, 2.3.2020

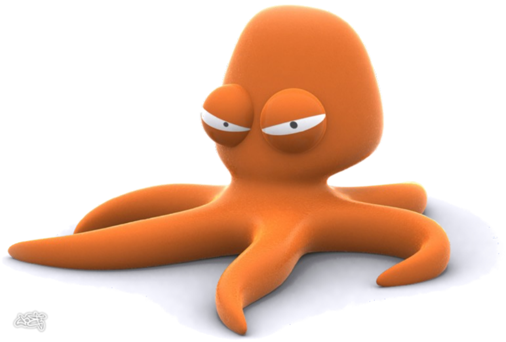


Why GPUs?

- Slower increase in CPU efficiency in last years
- Higher power efficiency of GPUs (FLOPS/Watt)
- All 3 US exascale machines will have GPUs

→ prepare now!

Octopus



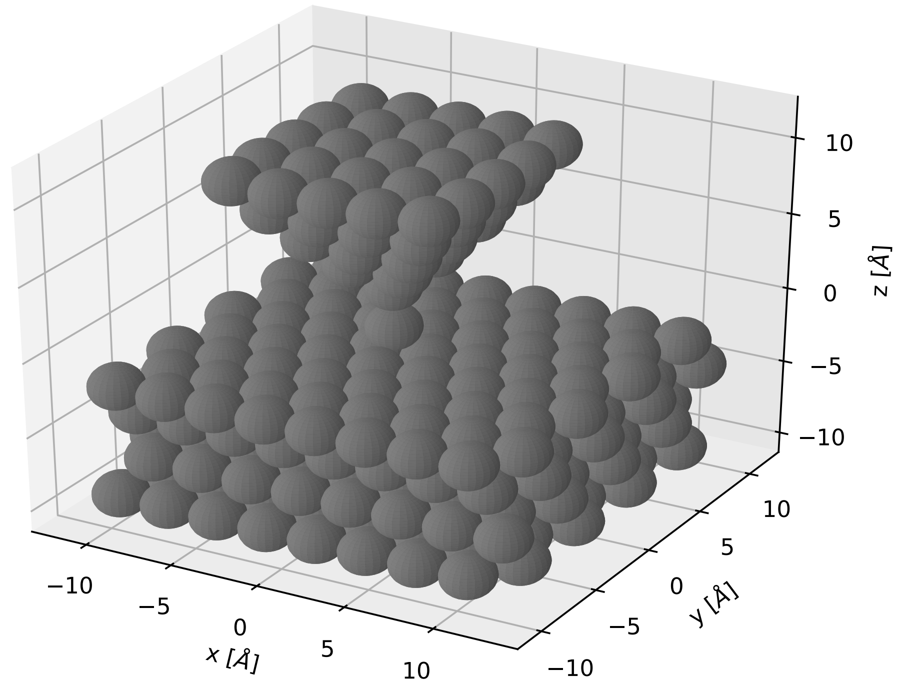
- Density functional theory code with pseudopotentials
- Real-space grid + finite differences
- Real-time time-dependent calculations
- Mainly Fortran, plus some C
- Open source: octopus-code.org

Octopus: GPU version

- Developed ~ 6 yr ago
- Written in OpenCL + wrapper for CUDA
- Interface Fortran $\leftrightarrow$ C
- Kernels compiled during runtime

Example system

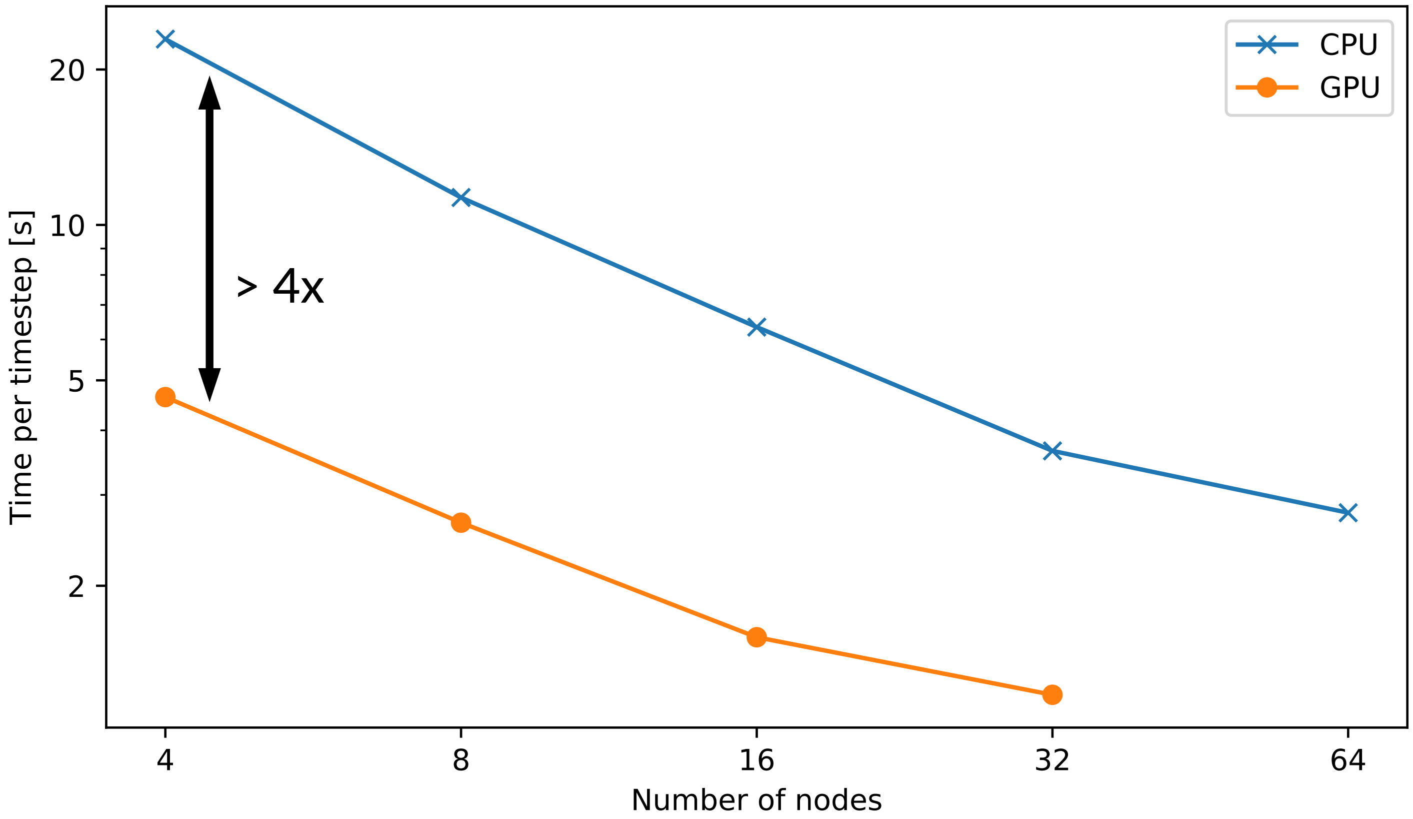
- Silver tip over crystal
- Periodic in x and y
- 312 Ag atoms
- 3200 orbitals
- 2.4 M grid points
- Compare TD runs



Benchmark clusters

- **Cobra** @ MPCDF
  - CPU nodes: 40 cores/node (2x 20-core sockets)
  - GPU nodes: CPU nodes + 2 Nvidia V100 GPUs
  - Interconnect: Omnipath (100 Gbit/s)
- GPU machines @ MPSD
  - 2x 8-core sockets
  - 8 Nvidia V100 GPUs + NVLink

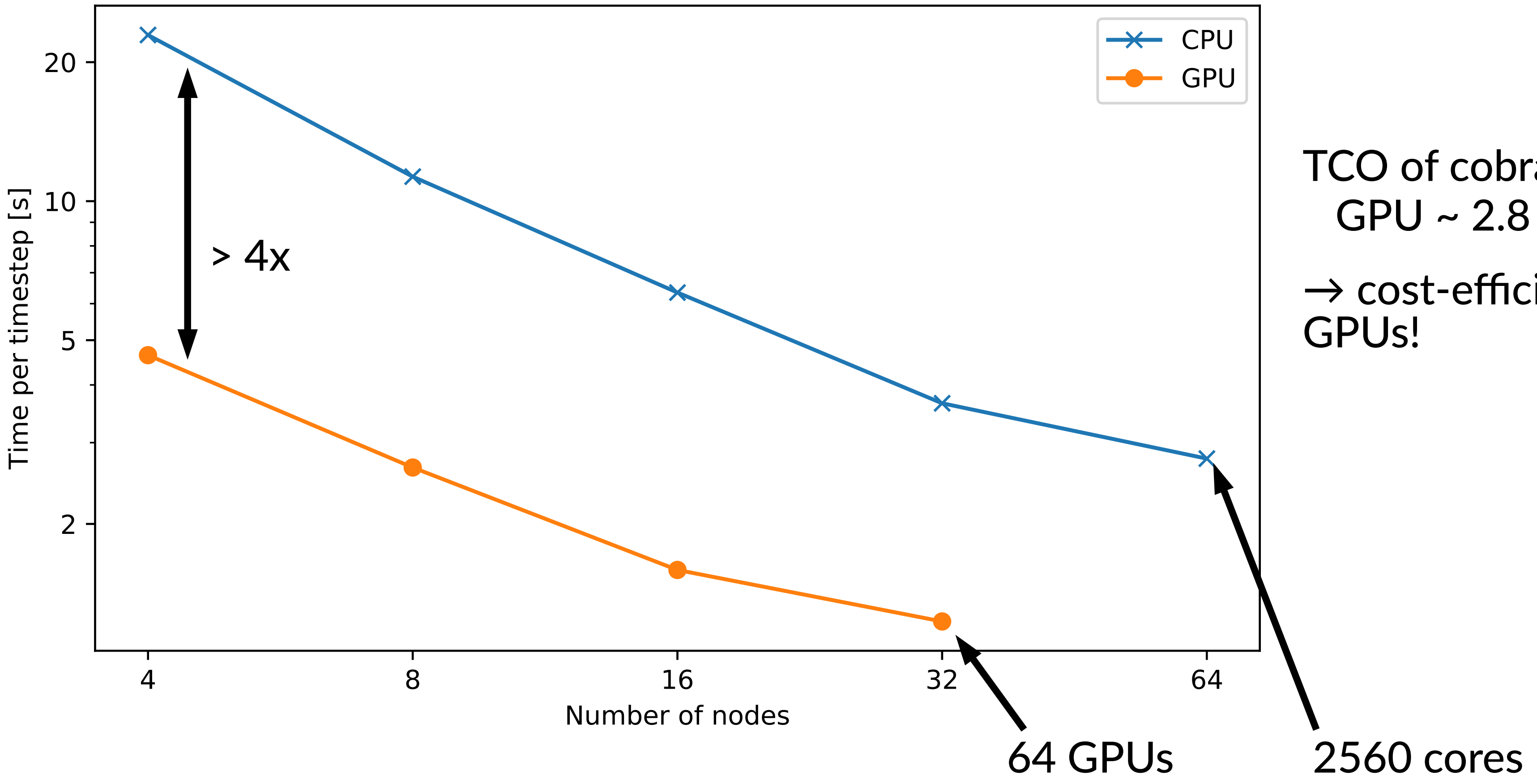
Comparison on cobra: CPU vs. GPU



TCO of cobra nodes:
GPU ~ 2.8 CPU

→ cost-efficient on GPUs!

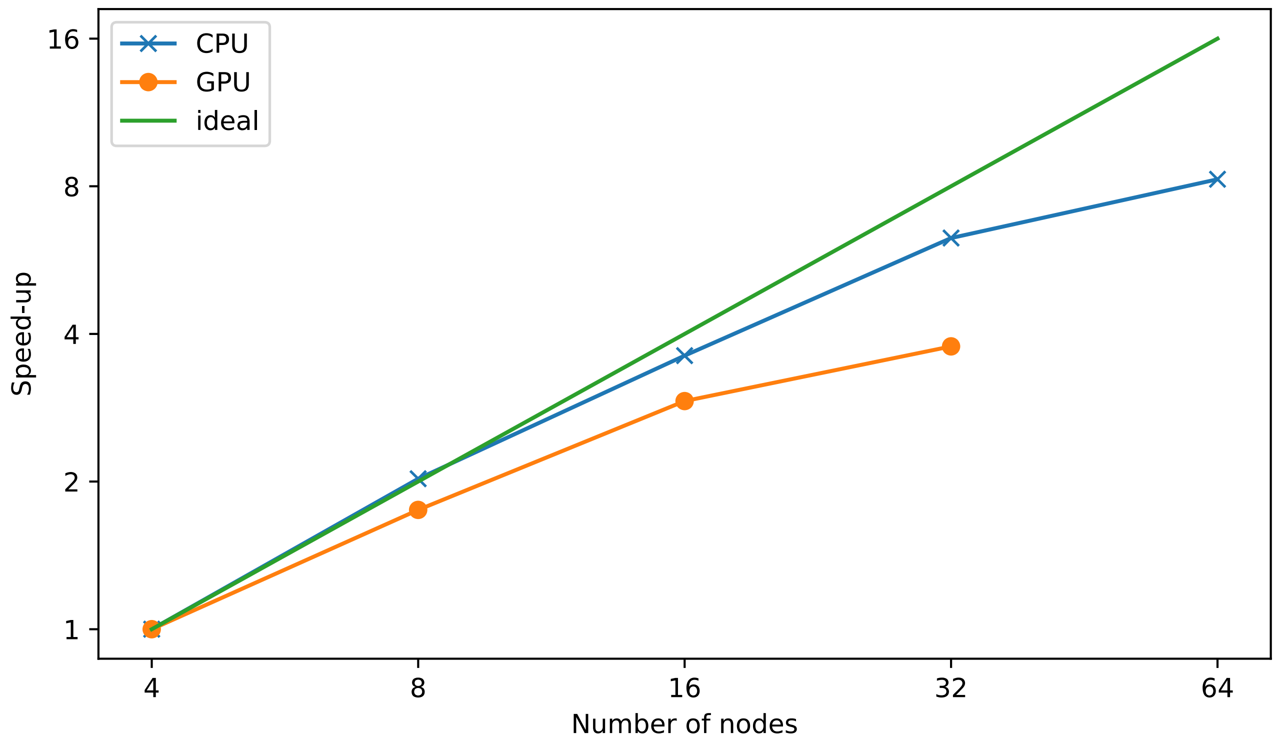
Comparison on cobra: CPU vs. GPU



TCO of cobra nodes:
GPU ~ 2.8 CPU

→ cost-efficient on GPUs!

Scaling on cobra

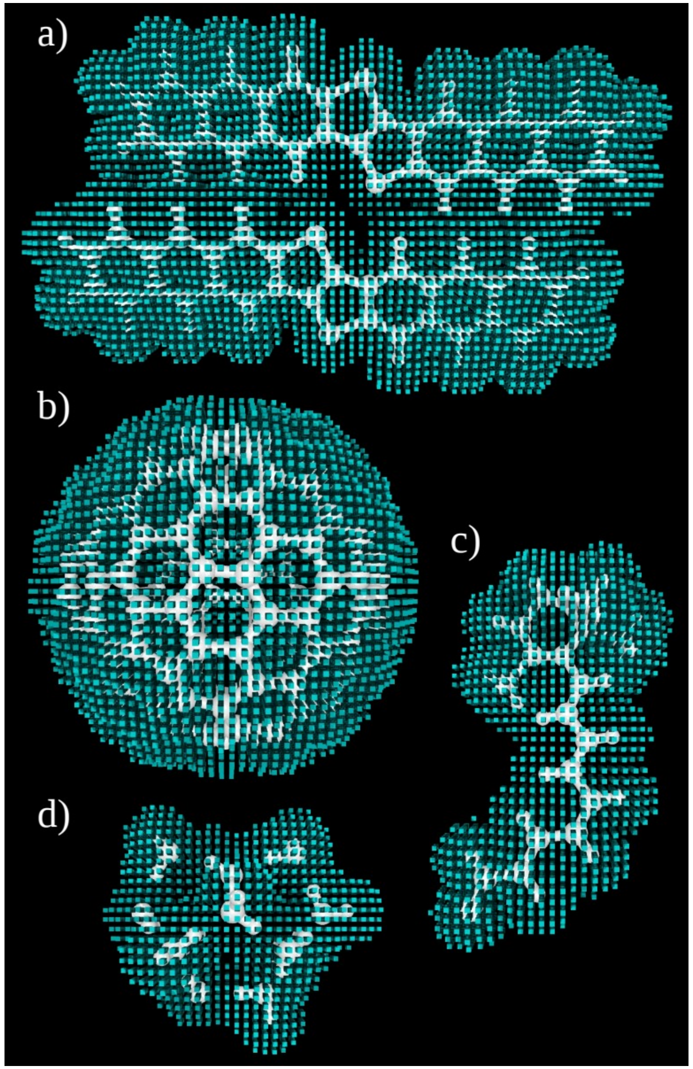


Implementation

- Pinned memory → faster transfer speed
- Streams → asynchronous operations
- CUDA-aware MPI → GPU-GPU communication
- Prefetching → overlap communication & computation

Data layout

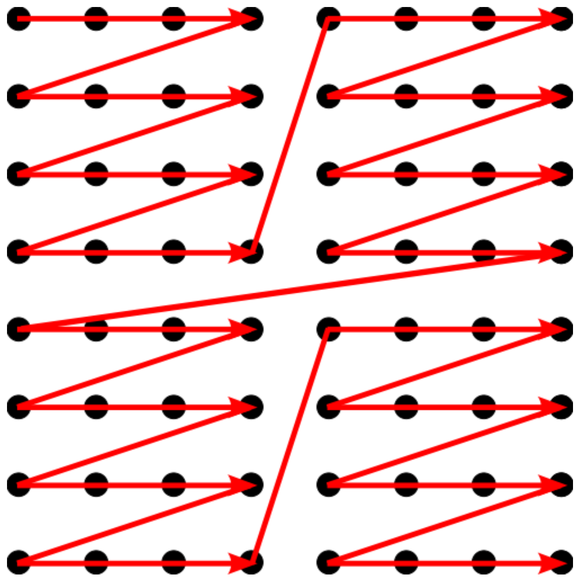
- Real-space grid for FD
- Complicated shape possible, e.g. molecules



X. Andrade & A. Aspuru-Guzik, J. Chem. Theory Comput. (2013), 9, 10, 4360-4373

Data layout

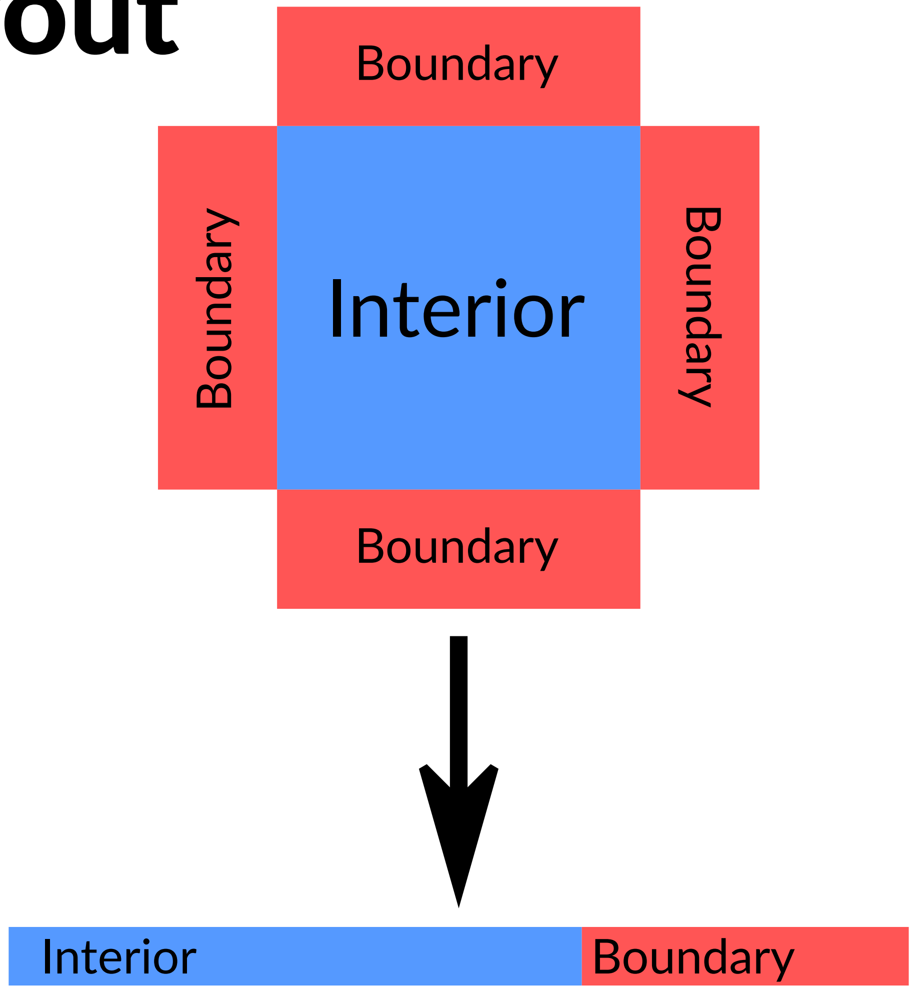
- Real-space grid for FD
- Complicated shape possible, e.g. molecules
- Cache-aware mapping to 1D array



X. Andrade & A. Aspuru-Guzik, J. Chem. Theory Comput. (2013), 9, 10, 4360-4373

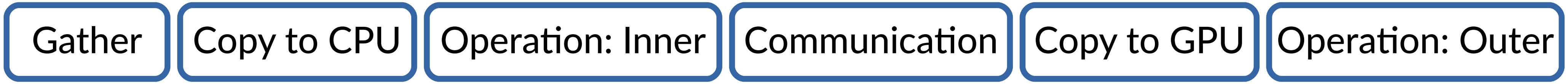
Data layout

- Real-space grid for FD
- Complicated shape possible, e.g. molecules
- Cache-aware mapping to 1D array
- 1D data layout: 2 blocks
  - Interior points
  - Boundary/ghost points



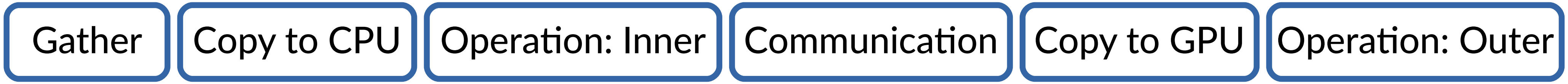
CUDA-aware MPI in octopus

- Timeline before (distributed mesh):

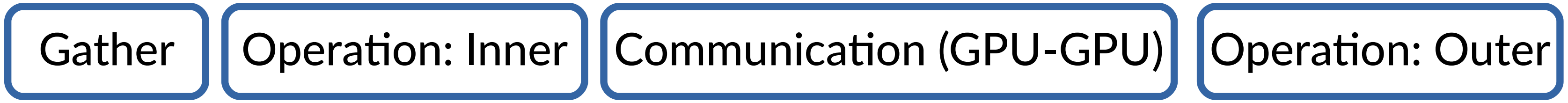


CUDA-aware MPI in octopus

- Timeline before (distributed mesh):

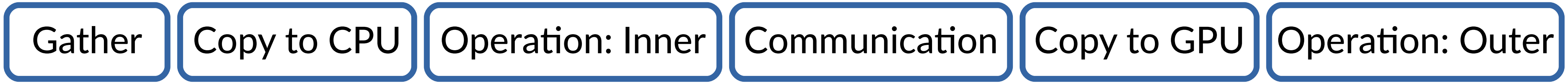


- With CUDA-aware MPI: communication between GPUs → no copies to/from GPU

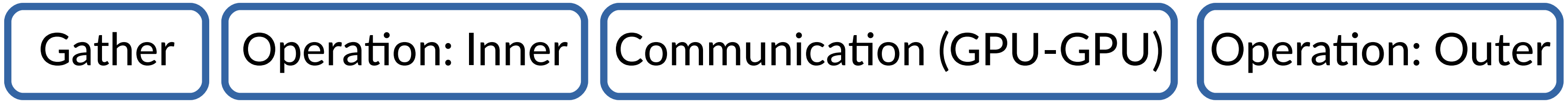


CUDA-aware MPI in octopus

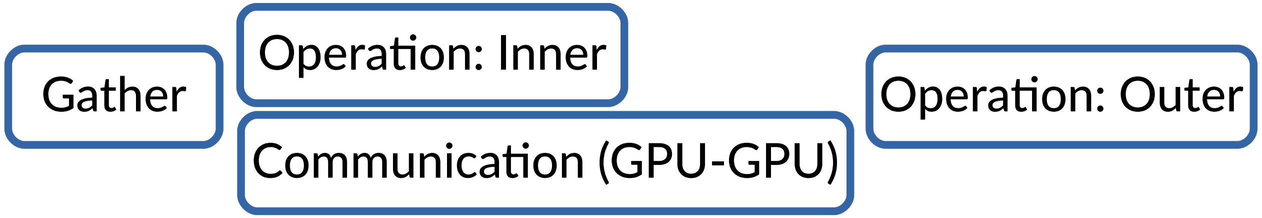
- Timeline before (distributed mesh):



- With CUDA-aware MPI: communication between GPUs → no copies to/from GPU



- CUDA-aware MPI + streams: overlap communication & computation

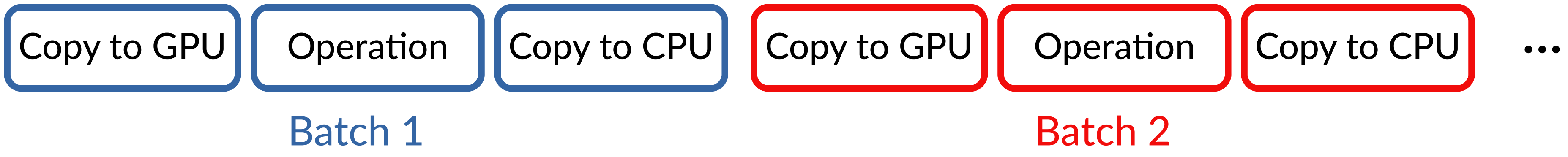


CUDA-aware MPI in octopus

- Implementation:
  - Get pointers to GPU memory from C
  - Use c_f_pointer in Fortran to get a Fortran pointer to this memory
  - Use this Fortran pointer in the MPI calls
- On 8 GPUs with NVLink (machine @ MPSD)
  - Peer-to-peer transfer speed: ~24 GB/s
  - Speed-up of ~ 2.4x vs. normal MPI

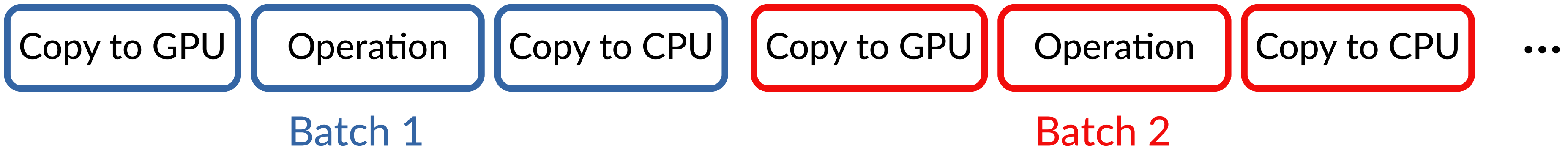
Prefetching batches

- Timeline without prefetching:

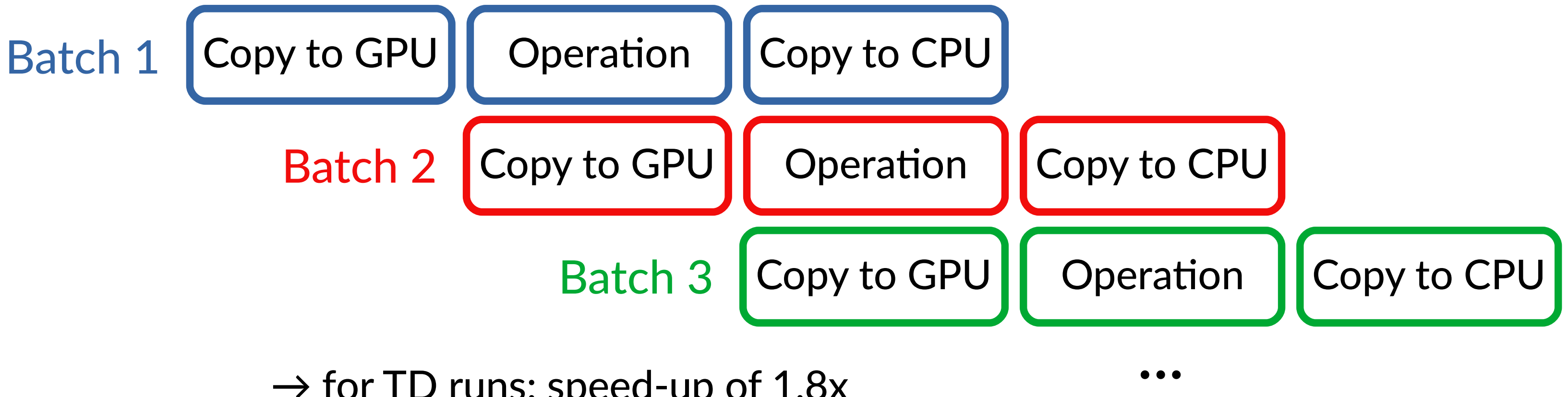


Prefetching batches

- Timeline without prefetching:



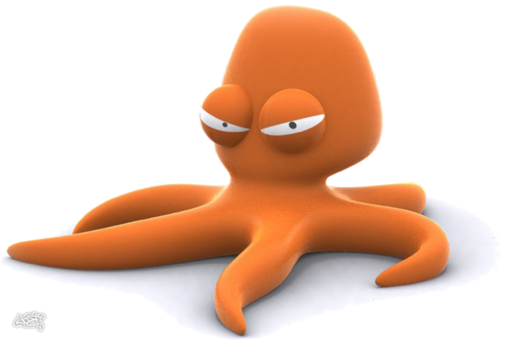
- Timeline with prefetching:



→ for TD runs: speed-up of 1.8x
(only used if states do not fit in GPU memory)

Summary

- Octopus more efficient on GPUs
- Improvements based on
  - Pinned memory
  - Streams
  - CUDA-aware MPI
  - Prefetching batches
- Outlook:
  - Port more parts of the code
  - Improve scaling

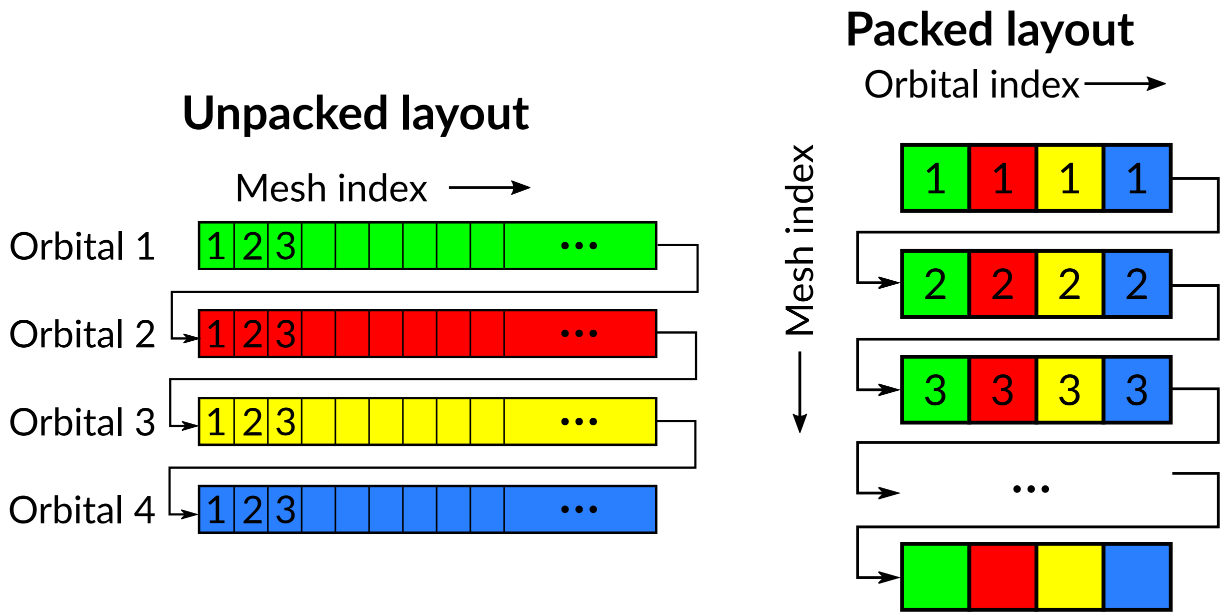


sebastian.ohlmann@mpcdf.mpg.de

Backup slides

Data layout II: batches

- Aggregate several orbitals into one batch
- Operations done over batches
- 2 layouts:
  - Unpacked
  - Packed → vectorization, GPUs

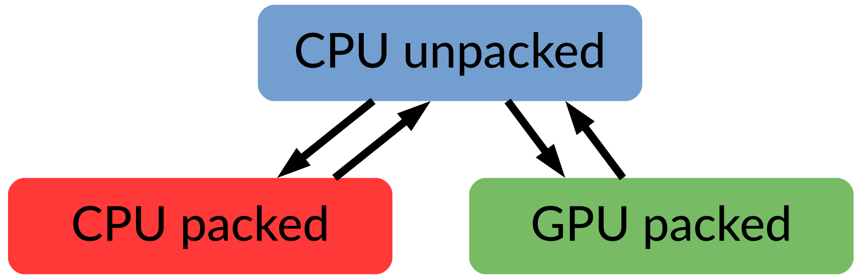


Batch handling

- Batch can have 3 states:

CPU unpacked   CPU packed   GPU packed

- Transitions before:



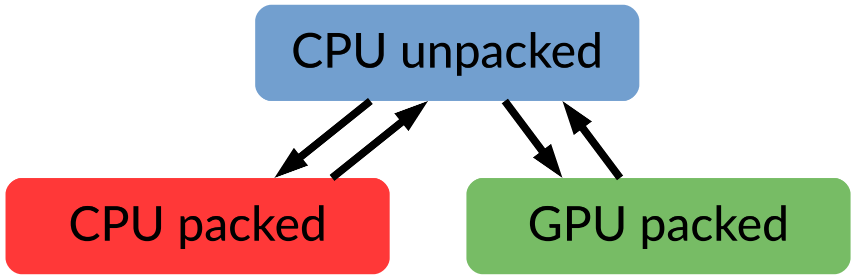
→ always involves transposition

Batch handling

- Batch can have 3 states:

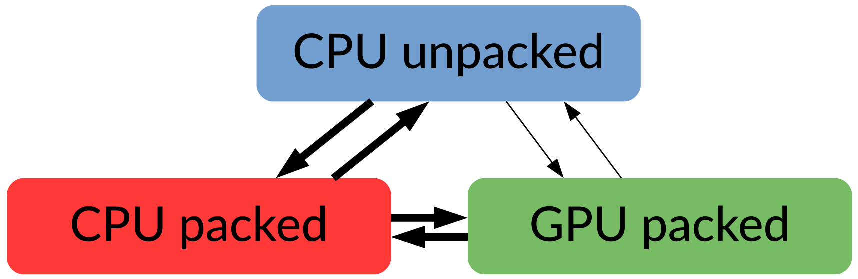
CPU unpacked
CPU packed
GPU packed

- Transitions before:



→ always involves transposition

- Transitions now:



→ simple copy to GPU

Pinned memory

- Normal allocations: pageable memory
- Transfers to GPU: pinned memory needed → faster transfer
- Solution:
  - Allocate pinned memory in C (CUDA call)
  - Use c_f_pointer in Fortran to use this memory
- Transfer speed on PCIe 3: ~12 GB/s vs. ~5 GB/s

Streams

- Default: CUDA operations are blocking
- Streams needed to overlap operations
- Also needed for CUDA-aware MPI
- 32 Streams are initialized in the C layer
- Selection from Fortran layer
- Usage example: asynchronously launch norm kernels with strides

CUDA-aware MPI

- Extension of MPI, available for some flavours (OpenMPI, MPICH, MVAPICH, …)
- Requires compatible low-level drivers
- Usage:
  - Pass GPU pointers to MPI calls
  - MPI library can directly access the GPU memory
- Advantages:
  - Peer-to-peer copies on the same node (even better with NVLink)
  - Less latency for inter-node communication

Overlap communication & computation

- 2 ways of running octopus on GPUs:
  - If enough GPU memory → store all batches on GPU
  - Otherwise → copy batch to GPU, operate, copy back
- For second way:
  - Overlap of communication & computation possible
  - Use asynchronous prefetching on different stream

Prefetching batches

- Advantage:
  - Hide copy latency, except for first & last copy
- Disadvantages:
  - Needs memory for 3 batches
  - Does not overlap completely if operation involves copies to/from the GPU
- For TD runs: speed-up of 1.8x

Timing data (Ag tip on cobra)

Nodes		4	8	16	32	64
CPU	Time [s]	22.9	11.3	6.34	3.65	2.77
	Speedup	1	2.0	3.6	6.3	8.3
GPU	Time [s]	4.64	2.65	1.59	1.23	
	Speedup	1	1.8	2.9	3.8	
GPU ParDomain=2	Time [s]	12.76	6.55	3.45	1.86	
	Speedup	1	2.0	3.7	6.9	
GPU StatesPack=no	Time [s]	5.26				

Timing data (Ag tip on MPSD machine)

Number of GPUs		1	2	4	8
GPU	Time [s]				4.44
GPU ParDomain=2	Time [s]				7.42
GPU ParDomain=2 normal MPI	Time [s]				17.9
GPU StatesPack=no	Time [s]	32.3	27.9	14.7	7.95
	Speedup	1	1.16	2.2	4.1
GPU StatesPack=no ParDomain=2	Time [s]		28	24	14
	Speedup		1	1.16	2