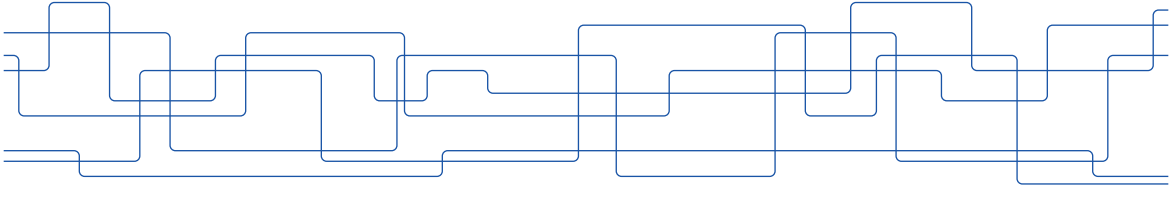
Using GPU-aware message passing to accelerate high-fidelity fluid simulations

Jacob Wahlgren
jacobwah@kth.se



Outline

1. Background: supercomputers, GPUs, message passing with MPI
2. Benchmarking GPU-aware MPI
3. Use case: computational fluid dynamics and Neko
4. Gather–scatter operation using GPU-aware MPI
5. Performance evaluation: speedup at scale
6. Conclusions

High-performance computing

Large computational problems

- Simulations in science and engineering
- Training large machine learning models

Supercomputers

- The fastest computers in the world
- Consist of many independent nodes
- Connected by high-speed network

Figure: Installation of Dardel at KTH in 2021.
Source: PDC.

GPU acceleration

A trend in supercomputing

- End of Moore's law
- Need for specialized hardware
- Graphics processing units (GPUs) provide high throughput
- Used in many of the fastest supercomputers

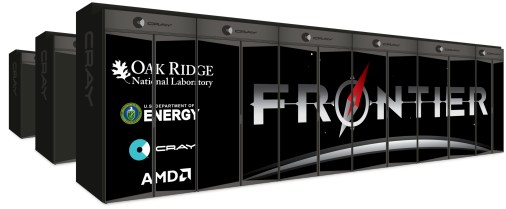


Figure: Frontier supercomputer at Oak Ridge National Laboratory. Source: ORNL.

GPU compute node

Host and device

- Initialization and program flow in CPU
- Compute power in GPU
- Separate memories
- Data movement is expensive
- Connected to high-speed network

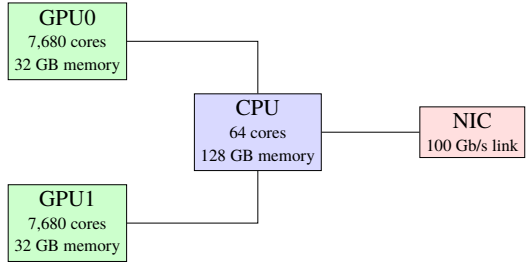


Figure: Example of a simple GPU compute node.

Inter-process communication

Distributed computation

- Distribute over several processes
- Larger problems (memory limit)
- Faster solution (compute limit)

Message passing between processes

- Message Passing Interface (MPI)
- Functions include `MPI_Send` and `MPI_Recv`
- One process per GPU

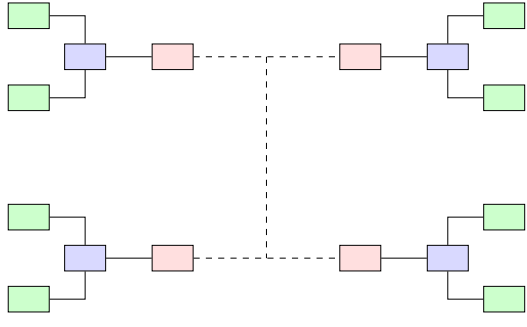


Figure: Four interconnected compute nodes.

GPU-aware MPI

- Normally MPI buffers in host memory
- GPU-aware MPI enables buffers in device memory
- Avoids data movement between host and device memory

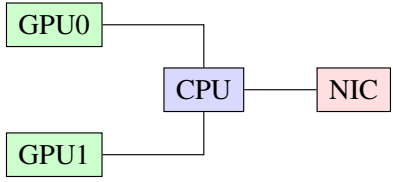


Figure: Example of a simple GPU compute node.

Example of GPU-aware MPI

Without GPU-aware MPI

```
for 1..N {
    d_y = gpu_compute(d_x)
    h_y = device_to_host(d_y)
    h_z = mpi_communicate(h_y)
    d_x = host_to_device(h_z)
}
```

With GPU-aware MPI

```
for 1..N {
    d_y = gpu_compute(d_x)
    d_z = mpi_communicate(d_y)
    d_x = d_z
}
```

Research question 1

How does GPU-aware message passing with device memory buffers perform compared to conventional message passing from host memory?

Experimental setup

Alvis supercomputer at C3SE

- Four NVIDIA A100 40GB per node
- NVLink between GPUs on the same node
- InfiniBand HDR network

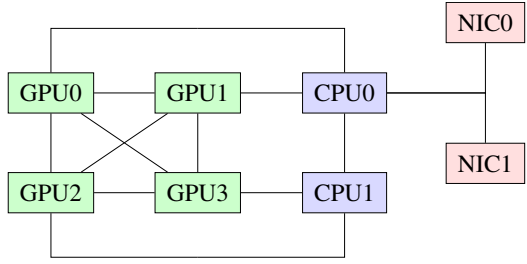
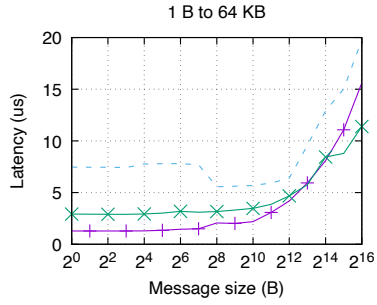


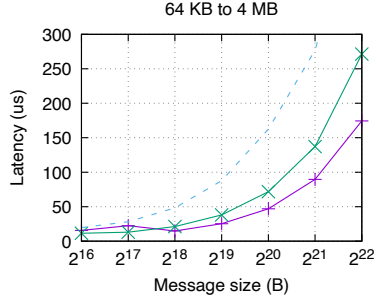
Figure: Alvis compute node.

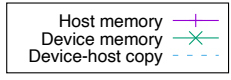
Benchmarking GPU-aware MPI between nodes

Inter-node latency

- Host memory faster than device memory for the most part
- Network communication faster than host-device copy
- Device memory faster than host-device copy + host memory MPI



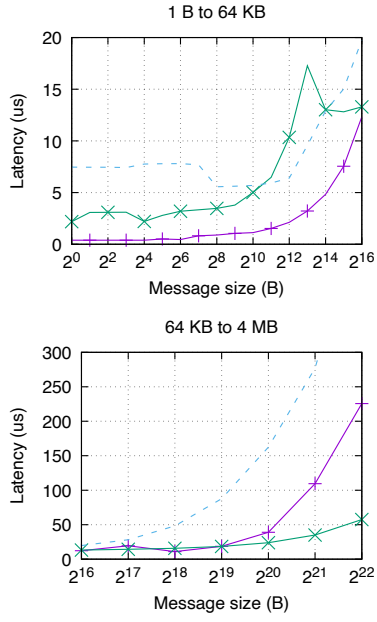




Benchmarking GPU-aware MPI on the same node

Intra-node latency

- Host memory is basically memcpy
- Device memory is sending between two GPUs
- GDRCopy or CUDA IPC depending on size
- Latency spike around 8 KB
- Host is bandwidth limited above 1 MB



Using it in practice

- Benchmarks of GPU-aware MPI show mixed, but overall good results
- What about in an application?

Computational fluid dynamics

Fluid flow

- Navier–Stokes equations
- Solved with numerical methods
- High-fidelity solutions require a lot of computation

Many use cases

- Aerodynamics of planes, cars
- Heat transport in nuclear reactors
- Efficiency of wind farms

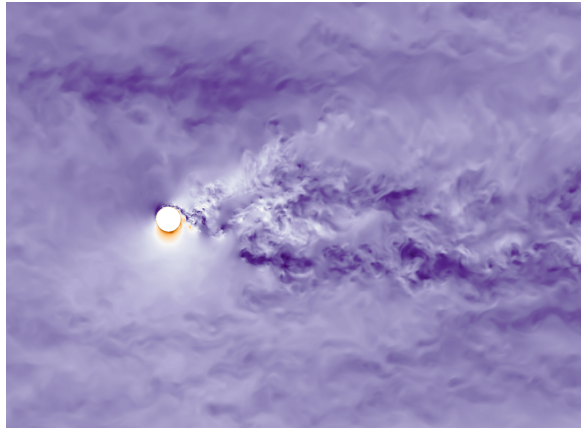


Figure: High-fidelity simulation of a Flettner rotor. Source: Karp et al. 2022.

Neko

Solver for incompressible fluids

- Supports GPU-acceleration
- Uses MPI communication
- Can scale to hundreds of GPUs
- Spectral element method for solving Navier–Stokes

Spectral element method

Spectral element method

- Domain decomposed into many small parts called elements
- Polynomial basis of order N on each element
- Computations done independently for every element
- Results are merged using the *gather–scatter operation*

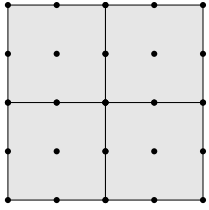


Figure: Four elements, $N = 2$.

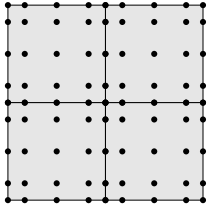


Figure: Four elements, $N = 4$.

Gather–scatter operation

Main communication kernel

- Sum values at element boundaries
- Use MPI if elements owned by different process
- In Neko executed in host memory

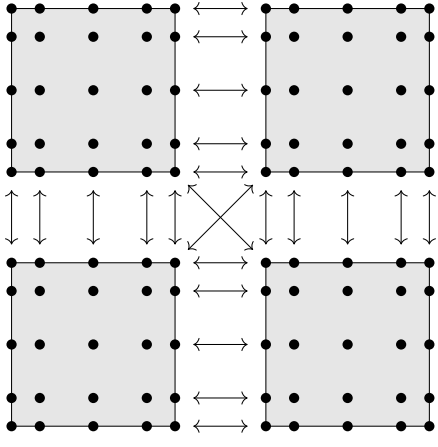
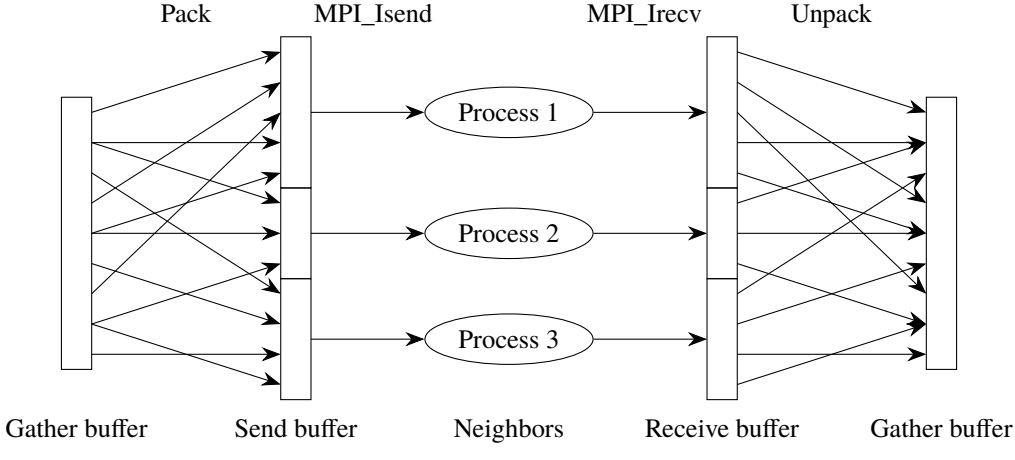


Figure: Arrows indicate shared points.

Research question 2

How can GPU-aware message passing be leveraged to optimize large-scale fluid simulations in the spectral element method?

Gather–scatter using GPU-aware MPI



Experimental setup

Alvis supercomputer at C3SE

- Four NVIDIA A100 40GB per node
- NVLink between GPUs on the same node
- InfiniBand HDR network
- One Neko process per GPU

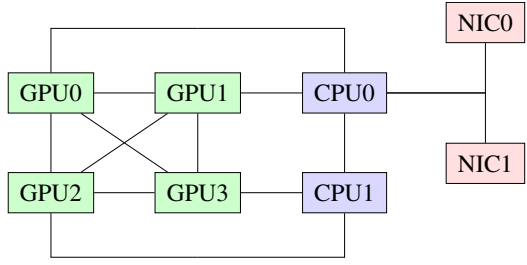
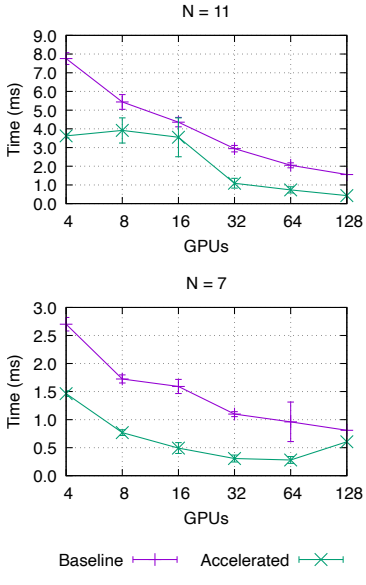


Figure: Alvis compute node.

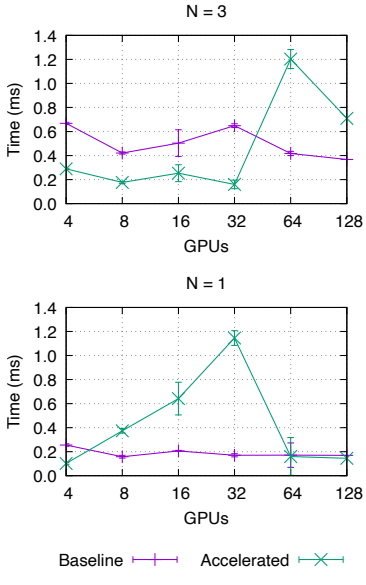
Benchmarking the accelerated gather–scatter

- Strong scaling with 262K elements
- Main solver orders
- Slower at higher N
- Speedup across whole range



Benchmarking the accelerated gather–scatter

- Coarse grid orders
- Faster than higher orders, but called more often
- Spikes in execution time correlate with MPI latency spikes
- Run time auto-tuning?



Taylor–Green vortex

Smaller version ($Re = 1600$)

- 32K elements, main order $N = 7$
- Reference solution for verification

Larger version ($Re = 5000$)

- 262K elements, main order $N = 9$
- For evaluating the scaling

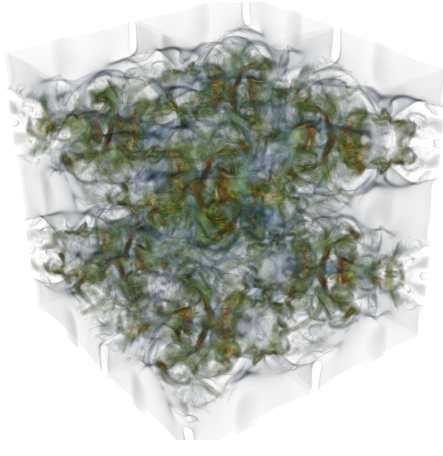


Figure: Velocity magnitude of the Taylor–Green vortex at $Re = 5000$. Source: Jansson et al. 2021.

Small Taylor–Green vortex

Using accelerated gather–scatter

- With few processes, communication is small fraction
- With more processes, communication is larger fraction
- Speedup of 1.66 at 12 GPUs

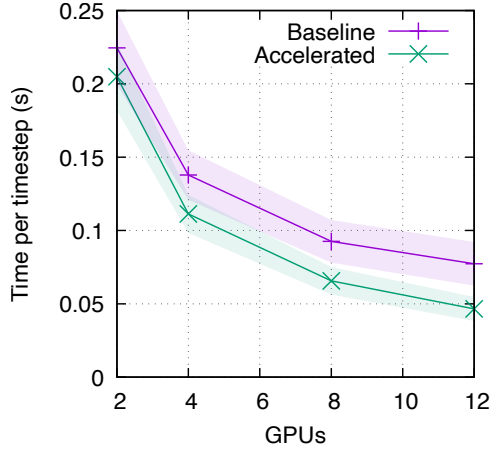


Figure: Strong scaling results.

Large Taylor–Green vortex

Accelerated performance at scale

- Up to 128 GPUs
- Speedup of 2.59 at 64 GPUs
- Speedup of 1.58 at 128 GPUs

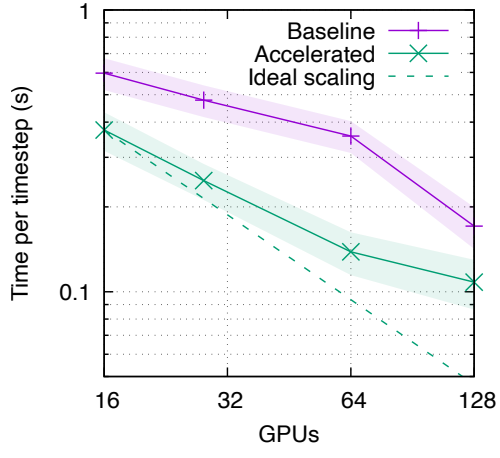


Figure: Strong scaling results (logarithmic).

Conclusions

GPU-aware MPI

- Overall good performance
- Bad intra-node at 1-64 KB

Accelerated gather–scatter in Neko

- Avoiding data movement enables speedup
- Will be included in v0.4

Future work

- Investigate intra-node performance issues
- Test on other systems
- Usage of GPU-aware MPI in other applications

Thanks for listening!

Supervisors: Niclas Jansson, Martin Karp
Examiner: Stefano Markidis