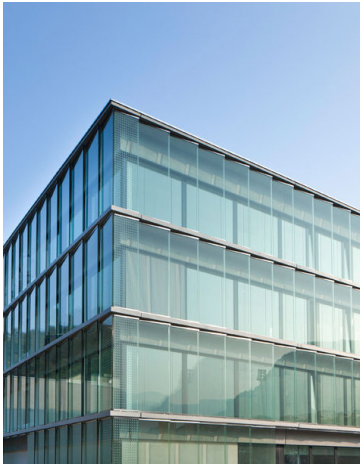
Optimised all-to-all communication on multicore architectures applied to FFTs with pencil decomposition

CUG 2018, Stockholm

Andreas Jocksch, Matthias Kraushaar (CSCS), David Daverio (University of Cambridge, Centre for Theoretical Cosmology)

24th May, 2018

Optimised all-to-all communication on multicore architectures applied to FFTs with pencil decomposition



- Basic algorithm
- Generalisation $\rightarrow$ Bruck
- Blocking / non-blocking communication
- GPU-direct
- Applications programming interface
- Benchmarks
- Applications

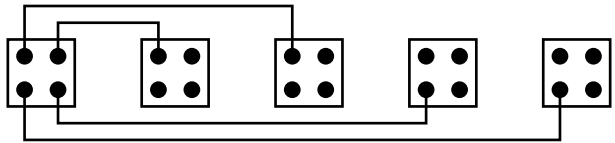
Motivation

- Modern supercomputers equipped with multicore CPUs
- Hybridly — OpenMP + MPI — or pure MPI programmed, focus on MPI only
- If GPUs are present often MPI + CUDA / OpenACC
- Collective communication operations provide fast message transfers for certain communication patterns
- All-to-all personalised communication critical for many applications
- Assumption of a fully connected network, described by latency and bandwidth

Motivation

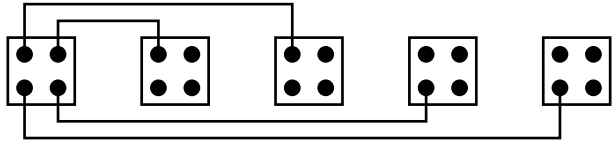
- Scheduling scheme performance critical
- For small messages store and forward algorithms, e.g., Bruck's algorithm
- MPICH, OpenMPI and MVAPICH apply the algorithm with all cores equally connected, the reason should be the current MPI standard
- For allreduce, reduce, broadcast, scatter and gather shared memory parallelism
- Generalised case of multiple all-to-all communication for pencil decomposed FFTs
- Library which exploits shared memory, Li and Laizet CUG 2010
- More efficient communication algorithm for all-to-all / all-to-allv / FFTs introduced here
- "plan" routines, GPU direct supported

Basic all-to-all algorithm



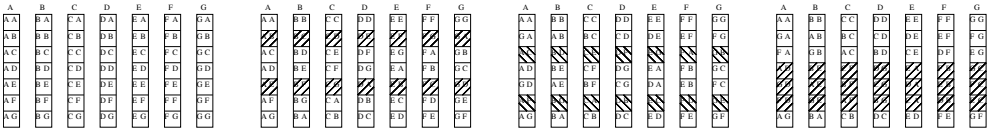
- Communication on the node using shared memory
- Communication between nodes done by multiple (e.g. all) cores

Basic all-to-all algorithm



1. MPI_Irecv
2. memcpy sendbuffer to shared sendbuffer
3. node_barrier
4. MPI_Isend
5. memcpy shared sendbuffer to shared receivebuffer, for data which remains on the node
6. node_barrier
7. MPI_Waitall
8. memcpy shared receivebuffer to receivebuffer

Generalisation → Bruck's algorithm



(a) Initial data

(b) After local rotation

(c) After substep 1 of communication step 1

(d) After substep 2 of communication step 1

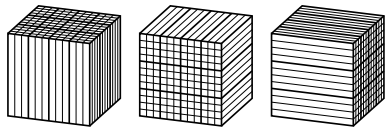
(e) After substep 1 of communication step 2

(f) After substep 2 of communication step 2

(g) After local inverse rotation

- Bruck et al. 1997

Multiple all-to-all algorithm



- For FFTs with pencil decomposition tasks communicate in multiple groups independent from each other
- Our routine includes this case

Blocking communication

- Communication algorithm message size dependent
- Very small messages forwarded between nodes
- Small messages sent directly using the shared memory segment
- Large messages are sent with the original MPI implementation
- Cyclic scheduling is applied
- Throtteling is applied as done in the MPICH reference implementation

Blocking communication

- A “plan” routine is used and a code generator encodes the algorithm
- The execution routine decodes the data
- MPI_Isend, MPI_Irecv, MPI_Wait are used
- Enrichment of the set of opcodes possible
- All-to-allv is implemented without any performance penalty

Non-blocking communication

- For overlapping of communication and computation as done for FFTs
- For no message forwarding and no throtteling one MPI_Waitall present
- Split of the execution in two parts, before and after the waitall, for start and end of the communication

GPU-direct

- Multiple MPI tasks might use one GPU
- Two implementations: small message copied between GPU and CPU and forwarded between nodes
- Large messages are bundled on the GPU with “interprocess communication”
- CUDA kernel which copies from many sourche buffers to many destination buffers

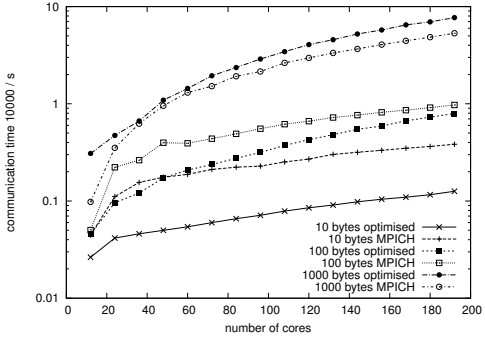
Application programming interface

- Routines are written in ANSI C with different interfaces
- Native one which calles our algorithm with all its parameters
- Automatic one which choses the algorithm and its parameters
- Both options are available in C and Fortran

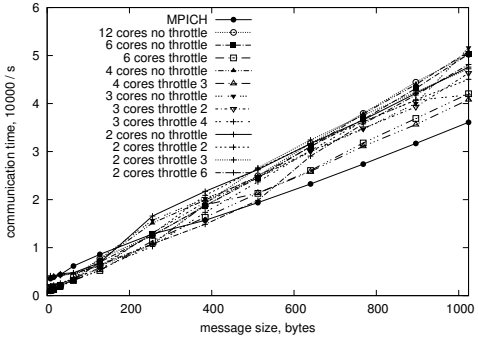
C/C++ : C native interface

```
int EXT_MPI_Alltoall_init_native (void *sendbuf, int sendcount,
    MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype
    recvtype, MPI_Comm comm_row, int cores_per_node_row, MPI_Comm
    comm_column, int cores_per_node_column, int num_ports, int
    num_active_ports, int chunks_throttle);
int EXT_MPI_Alltoall_exec_native (int handle);
int EXT_MPI_Alltoall_done_native (int handle);
```

Benchmarks



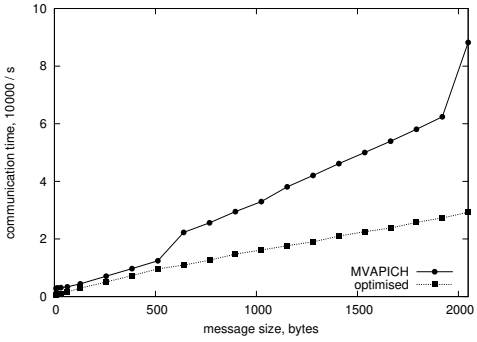
standard all-to-all, 12 nodes, 12 cores per node



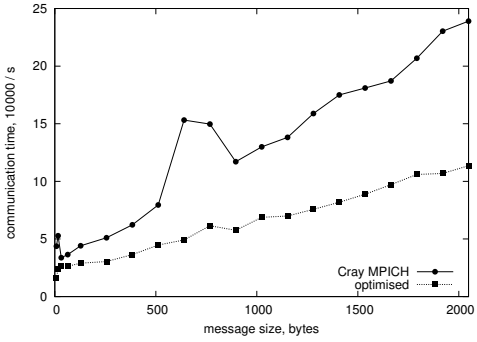
standard all-to-all, variable tile size, 12 nodes, 12 cores per node

- Cray XC50, 12 CPU cores per node, 1 NVIDIA Tesla P100
- Cray XC 40 KNL, 12 cores used per KNL

Benchmarks



all-to-all on Piz Kesch, 5 nodes, 12 cores per node

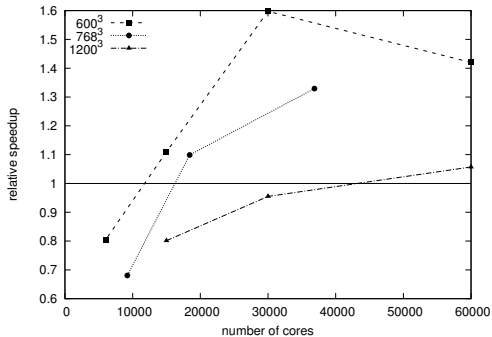


GPU-direct, 12 nodes, 12 cores per node

- Infiniband cluster Piz Kesch (Meteoswiss)
- GPU-direct on Piz Daint

Applications: LATfield2

- Spectral code with real-to-complex 3D FFTs and complex-to-real 3D FFTs
- High memory requirement $\rightarrow$ small messages originating from FFTs



relative speedup for FFTs

Applications: ORB5

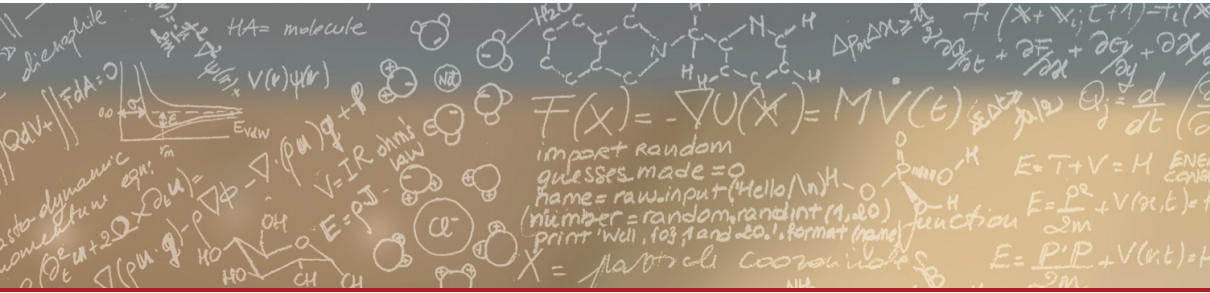
- Particle in cell code with toroidal domain
- MPI parallelisation in subdomains and clones
- Random variable message sizes for particle exchange between all subdomains (all-to-all communication)
- Typically all clones of a specific subdomain on a node $\rightarrow$ application for our algorithm

Discussion

- Implementation of the single all-to-all algorithm in the current MPI standard possible but would slowdown the communicator creation
- Multiple all-to-all possible with a graph communicator, also computationally expensive
- Suggestion: extension of the MPI standard with ”plan” routines

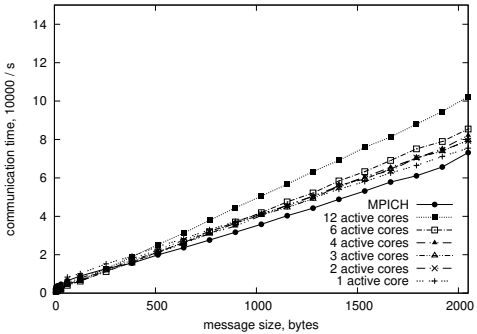
Conclusions

- Message passing algorithm for all-to-all communication for networks with shared memory nodes
- Cray MPICH outperformed by a factor of 2.9 for 10 bytes messages
- On infiniband cluster standard MVAPICH outperformed for small messages
- GPU implemenation advantegous for small and medium message size
- Application to FFTs (LATfield2) demonstrates its strength

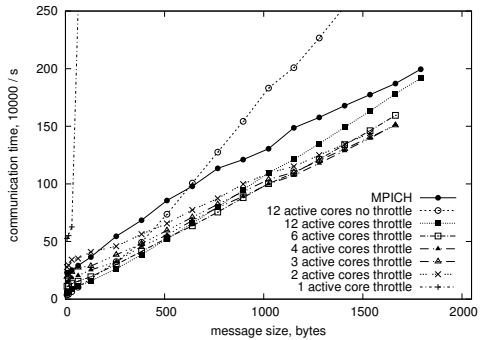


Thank you for your attention.

Benchmarks

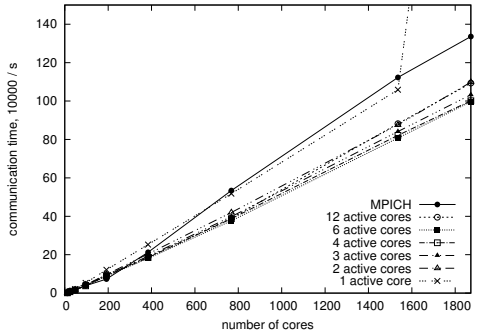


standard all-to-all, variation of active number of cores, 12 nodes, 12 cores per node



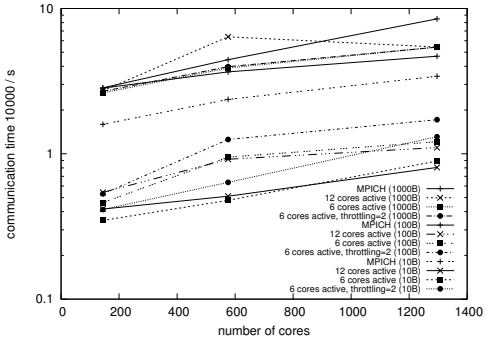
standard all-to-all, 156 nodes, 12 cores per node

Benchmarks

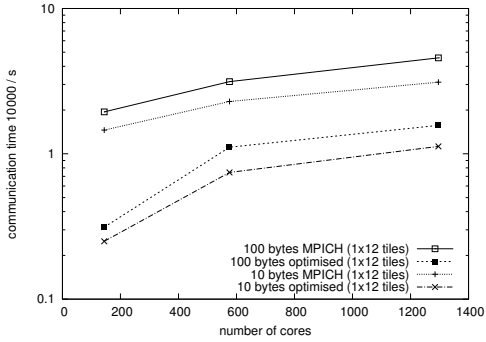


standard all-to-all, 1024 bytes, 12 cores per node, throttle

Benchmarks



multiple all-to-all 3 × 4 cores



multiple all-to-all 1 × 12 cores