Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI

T. Hoefler[1,2], A. Lumsdaine[1] and W. Rehm[2]

[1]Open Systems Lab
Indiana University
Bloomington, IN 47405, USA

[2]Computer Architecture Group
Technical University of Chemnitz
Chemnitz, 09111 Germany

Supercomputing 2007
Reno, NV, USA
15th November 2007

Common Optimization Techniques

To decrease the time to solution!

Serial (CPU) Optimization

- Optimizing Compilers
- Code Tweaks (loop unrolling, manual vectorization)
- Optimized Routines (math libraries, BLAS, FFT)

Parallel (Communication) Optimization

- ⇒ adds a second layer on top of CPU optimization
- Schedule Communication (e.g., Alltoall, Collective Patterns)
- Hardware Collective Operations (e.g., Multicast)
- Overlap of Communication and Computation

Common Optimization Techniques

To decrease the time to solution!

Serial (CPU) Optimization

- Optimizing Compilers
- Code Tweaks (loop unrolling, manual vectorization)
- Optimized Routines (math libraries, BLAS, FFT)

Parallel (Communication) Optimization

- ⇒ adds a second layer on top of CPU optimization
- Schedule Communication (e.g., Alltoall, Collective Patterns)
- Hardware Collective Operations (e.g., Multicast)
- Overlap of Communication and Computation

Parallel Optimization Techniques

Collective Communication

and

Communication/Computation overlap

should be combined to achieve maximum performance!

⇒ non-blocking collective operations!

Non-blocking Collective Operations

Related Work

- Implemented in IBMs PE but no details/analysis available
- Specified for UPC (not in version 1.2)
- “Split barrier” has been used before (CAF, UPC)
- Danalis et al. replaced MPI_Alltoall with linear MPI_Isend/Irecv pattern
- MPI JoD defines Split Collectives

⇒ LibNBC

- Implementation on top of MPI
- Using MPI_Isend/Irecv with optimized collective algorithms
- Support for all MPI collectives
- Very low overhead

LibNBC Interface

- extension to MPI-2
- ”mixture” between non-blocking ptp and collectives
- uses MPI_Requests and MPI_Test/MPI_Wait

```
NBC_Handle req;
NBC_Ibcast(buf1, p, MPI_INT, 0, MPI_COMM_WORLD, &req);

/* do computation to overlap latency */

NBC_Wait(&req);
```

Proposal

Hoefler et. al. (2006): *”Non-Blocking Collective Operations for MPI-2”*

LibNBC Interface

- extension to MPI-2
- ”mixture” between non-blocking ptp and collectives
- uses MPI_Requests and MPI_Test/MPI_Wait

```
NBC_Handle req;
NBC_Ibcast(buf1, p, MPI_INT, 0, MPI_COMM_WORLD, &req);

/* do computation to overlap latency */

NBC_Wait(&req);
```

Proposal

Hoefler et. al. (2006): *”Non-Blocking Collective Operations for MPI-2”*

Implementation of LibNBC

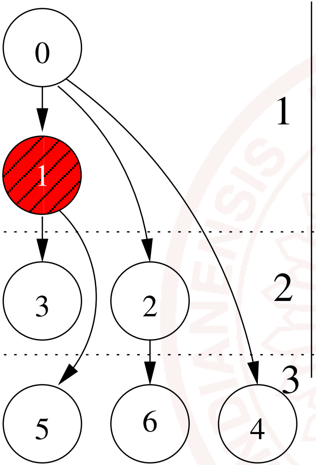
Collective Schedules

- Based on a round-based collective schedule.
- A round consists of non-blocking sends/recvs that run simultaneously.
- A round is finished if all operations are finished.
- Every algorithm can be expressed as such a schedule!

Interface

- NBC_Sched_recv/send, NBC_Sched_barr, NBC_Sched_copy, NBC_Sched_op, ...
- Addition of new algorithms is easy

LibNBC - Schedule Example



Pseudocode for schedule at rank 1:

```
NBC_Sched_recv(buf, count, dt, 0, schedule);
NBC_Sched_barr(schedule);
NBC_Sched_send(buf, count, dt, 3, schedule);
NBC_Sched_barr(schedule);
NBC_Sched_send(buf, count, dt, 5, schedule);
```

creating a broadcast schedule for rank 1 of 7
with a binomial tree algorithm

LibNBC - Schedule Example

- schedule is stored as a linear array
- all information is encoded in the elements
- 32-48 bytes per element

recv from 0	end	send to 3	end	send to 5

a broadcast schedule for rank 1 of 7
with a binomial tree algorithm

Benchmarking LibNBC

- microbenchmark methodology is different
- benchmark latency and overlap
- overlap more important than latency
- new microbenchmark ⇒ NBCBench
- takes the time on a single node
- prints the median of the maximum of N measurements

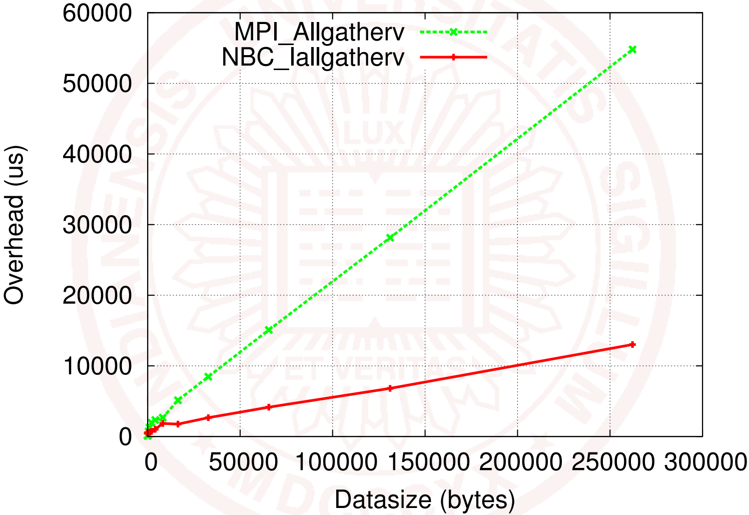
NBCBench Methodology

- measures the time for a “blocking execution” as t_{bl}
- execute `do_compute()` runs for time t_{bl}

```
MPI_Barrier(comm) or Internal_Barrier(comm);

NBC_Ibcast(buf, count, type, root, comm, handle);

do_computation_test(duration);

NBC_Wait(handle);
```

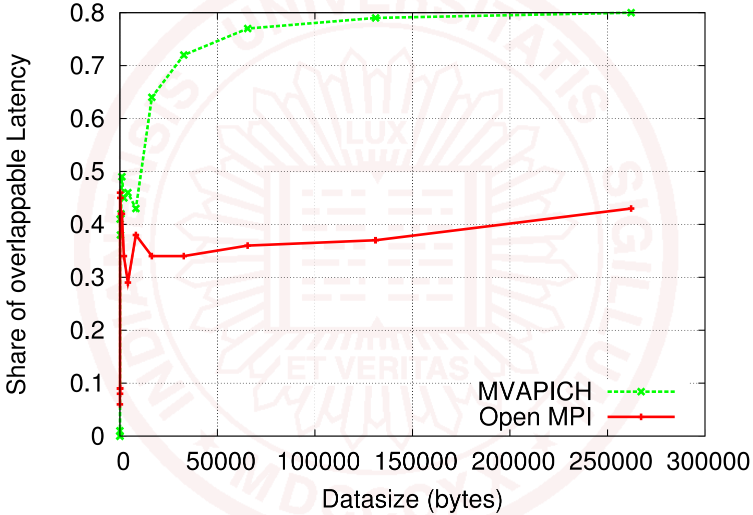
overhead = t(NBC_Ibcast) + t(NBC_Test) + t(NBC_Wait)

NBCBench Results



NBC_Iallgather vs. MPI_Allgather on 64 InfiniBand nodes

NBCBench Results



NBC_Iallgather vs. MPI_Allgather on 64 InfiniBand nodes

Analysis of Overhead

Sources of Overhead

- schedule creation
- copy overhead (some operations)
- MPI_Isend/Irecv overhead
- MPI_Testall/Waitall overhead

Our Findings

- schedule creation and copy overhead are negligible
- MPI overheads are dominating
- MPIs don't support independent progress

Application Kernel Benchmarks

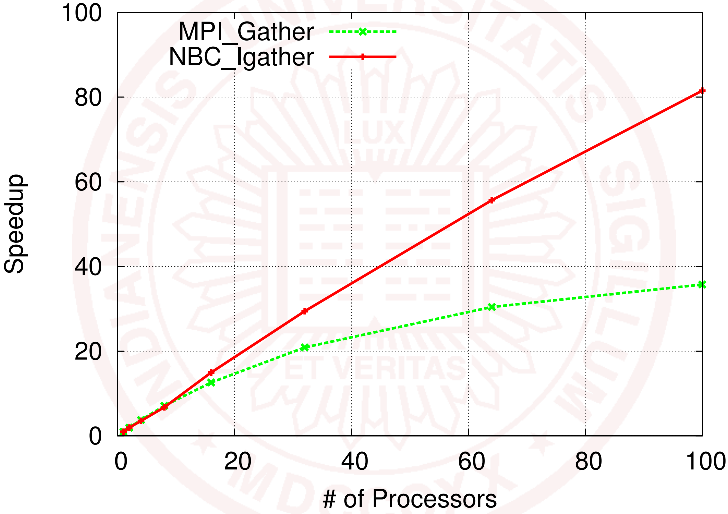
Parallel Compression

- Used in scientific applications
- Optimized gathering to a single process
- NBC_Igather vs. MPI_Gather
- pipelining of blocks

3d Poisson Solver

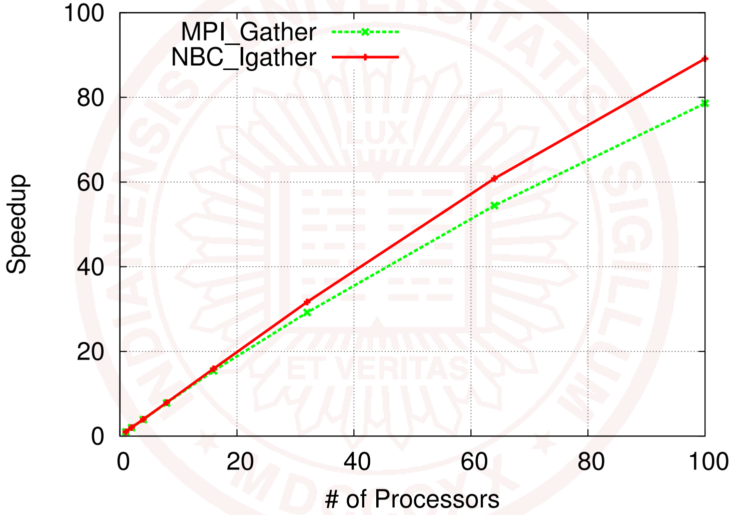
- → parallel CG
- NBC_Ialltoallv vs. MPI_Alltoallv
- overlap of halo zone communication with calculation

Parallel Compression - Gigabit Ethernet



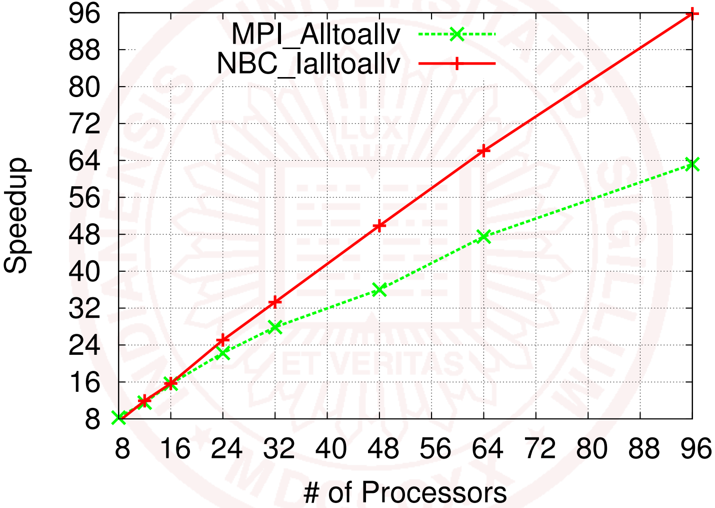
compressing 15.25 MiB on Dual Opteron Nodes

Parallel Compression - InfiniBand



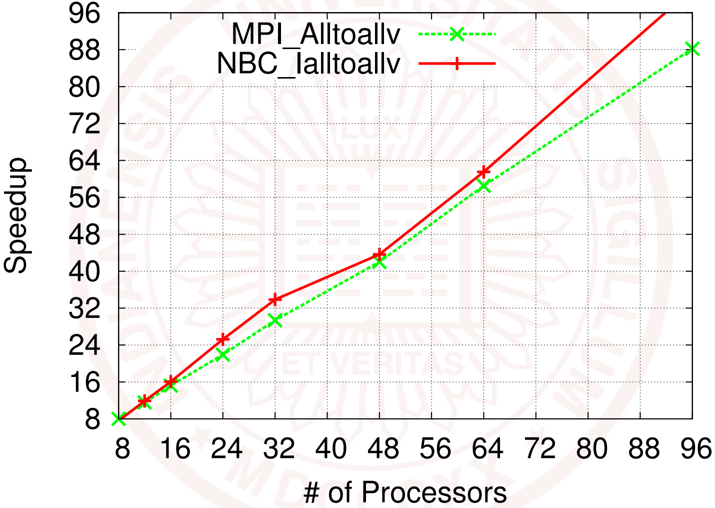
compressing 15.25 MiB on Dual Opteron Nodes

Parallel Conjugate Gradient - Gigabit Ethernet



800^3 system on Dual Opteron Nodes

Parallel Conjugate Gradient - InfiniBand



800^3 system on Dual Opteron Nodes

Conclusions and Future Work

LibNBC Download/Further Information

http://www.unixer.de/NBC/

Conclusions

- low-overhead implementaion of non-blocking collectives
- new benchmark to assess overhead
- main problem due to MPI overhead

Future Work:

- LibNBC will be shipped with Open MPI 1.3!
- hardware-optimize LibNBC
- optimize applications
- ⇒ We would like to collaborate with scientists!

Conclusions and Future Work

LibNBC Download/Further Information

http://www.unixer.de/NBC/

Conclusions

- low-overhead implementaion of non-blocking collectives
- new benchmark to assess overhead
- main problem due to MPI overhead

Future Work:

- LibNBC will be shipped with Open MPI 1.3!
- hardware-optimize LibNBC
- optimize applications
- ⇒ We would like to collaborate with scientists!

Conclusions and Future Work

LibNBC Download/Further Information

http://www.unixer.de/NBC/

Conclusions

- low-overhead implementaion of non-blocking collectives
- new benchmark to assess overhead
- main problem due to MPI overhead

Future Work:

- LibNBC will be shipped with Open MPI 1.3!
- hardware-optimize LibNBC
- optimize applications
- ⇒ We would like to collaborate with scientists!