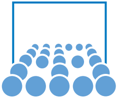
Technische Universität München

Assignment 3: MPI Point-to-Point and One-Sided Communication

Programming of Super Computers

Friedrich Menhorn, Benjamin Rüth, Erik Wannerberg
Team 12

December 22, 2015



Contents

1. **Provided Implementation and Baseline**
   - 1.1 Cannon’s algorithm
   - 1.2 Baseline
   - 1.3 Scaling

2. **MPI Point-to-Point Communication**
   - 2.1 MPI Non-Blocking Operations
   - 2.2 Optimizations
   - 2.3 Scaling

3. **MPI One-Sided Communication**
   - 3.1 MPI One-Sided Operations
   - 3.2 Optimizations
   - 3.3 Scaling

4. **Overall Comparison**

1. Provided Implementation and Baseline

1.1 Cannon’s algorithm

1.2 Baseline

1.3 Scaling

2. MPI Point-to-Point Communication

2.1 MPI Non-Blocking Operations

2.2 Optimizations

2.3 Scaling

3. MPI One-Sided Communication

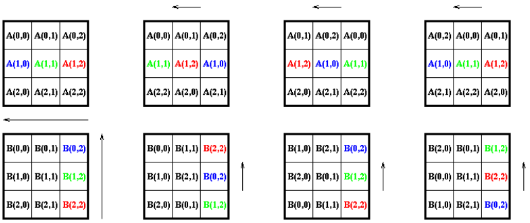
3.1 MPI One-Sided Operations

3.2 Optimizations

3.3 Scaling

4. Overall Comparison

Cannon’s algorithm



Initial A, B

A(0,0)	A(0,1)	A(0,2)
A(1,0)	A(1,1)	A(1,2)
A(2,0)	A(2,1)	A(2,2)

B(0,0)	B(0,1)	B(0,2)
B(1,0)	B(1,1)	B(1,2)
B(2,0)	B(2,1)	B(2,2)

A, B initial alignment

A(0,0)	A(0,1)	A(0,2)
A(1,1)	A(1,2)	A(1,0)
A(2,2)	A(2,0)	A(2,1)

B(0,0)	B(1,1)	B(2,2)
B(1,0)	B(2,1)	B(0,2)
B(2,0)	B(0,1)	B(1,2)

A, B after shift step 1

A(0,1)	A(0,2)	A(0,0)
A(1,2)	A(1,0)	A(1,1)
A(2,0)	A(2,1)	A(2,2)

B(1,0)	B(2,1)	B(0,2)
B(2,0)	B(0,1)	B(1,2)
B(0,0)	B(1,1)	B(2,2)

A, B after shift step 2

A(0,2)	A(0,0)	A(0,1)
A(1,0)	A(1,1)	A(1,2)
A(2,1)	A(2,2)	A(2,0)

B(2,0)	B(0,1)	B(1,2)
B(0,0)	B(1,1)	B(2,2)
B(1,0)	B(2,1)	B(0,2)

Figure: Cannons’s algorithm. Figure from[1].

- good and simple on homogeneous 2D grid topology
- provided implementation with blocking `MPI_Send/MPI_Recv`
- no initial alignment in provided implementation

[1] Zhiliang Xu. “Lecture Notes: Advanced Scientific Computing (Lecture 5 part 3)”. In: (2014). URL: http://www3.nd.edu/%7B~%7Dzxu2/acms60212-40212/Lec-06-3.pdf.

Baseline

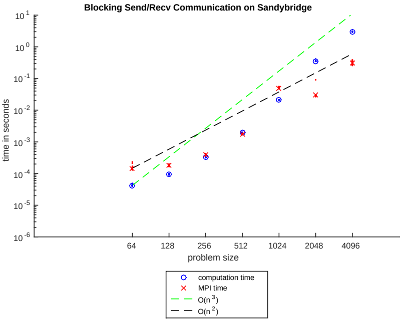
Challenges:

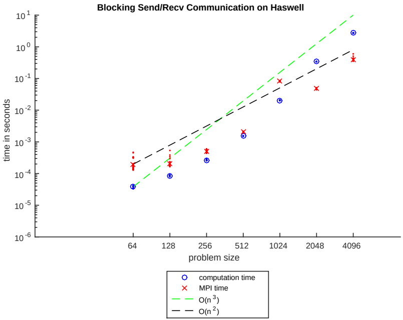
- variance of test runs
- any more???

Batch Script:

- writing output to file `mpiexec -n 64 ./cannon$arch_ending $cannon_matrices_path/64x64-1.in $cannon_matrices_path/64x64-2.in | tee 64_$JOB_ID.out`
- postprocessing `.out` files into `.csv` files
- automation of job submission for multiple test runs

Scaling


Blocking Send/Recv Communication on Sandybridge


Blocking Send/Recv Communication on Haswell

- Communication and computation time match well with theoretical complexity.
- Sandybride and Haswell similar, but Haswell has higher variance

1. Provided Implementation and Baseline
   - 1.1 Cannon's algorithm
   - 1.2 Baseline
   - 1.3 Scaling
2. **MPI Point-to-Point Communication**
   - 2.1 MPI Non-Blocking Operations
   - 2.2 Optimizations
   - 2.3 Scaling
3. MPI One-Sided Communication
   - 3.1 MPI One-Sided Operations
   - 3.2 Optimizations
   - 3.3 Scaling
4. Overall Comparison

MPI Non-Blocking Operations

- Send/Receive
  - **MPI_Isend**
  - **MPI_Irecv**
- Synchronization
  - **MPI_Wait**
  - **MPI_Probe**

Optimizations

What is overlap?

We do not wait for either task to be completed, but try to do communication and computation at the same time. Therefore blocking communication cannot result in any overlap.

What is the theoretical maximum overlap that can be achieved?

Bounds for pure communication time:

$$\max\left(0, T_{\text{MPI}}^{\text{blocking}} - T_{\text{computation}}\right) \leq T_{\text{MPI}}^{\text{non-blocking}} \leq T_{\text{MPI}}^{\text{blocking}}$$

As soon as $T_{\text{computation}} > T_{\text{MPI}}^{\text{blocking}}$, we can theoretically achieve 100% overlap.

Overheads

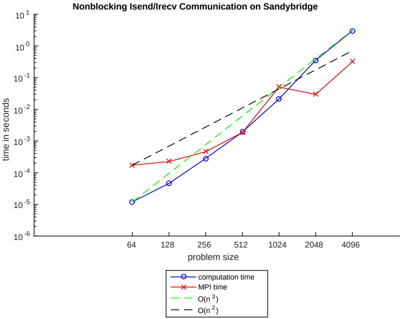
- Copying into and from buffers
- Initialization

Optimizations (contd.)

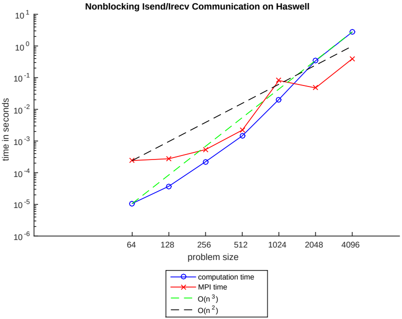
Was communication and computation overlap achieved?

```
// cannon's algorithm
 ...
MPI_Request send_row_request;
MPI_Request send_column_request;
MPI_Request recv_row_request;
MPI_Request recv_column_request;
 ...
for(cannon_block_cycle = 0; cannon_block_cycle < sqrt_size; cannon_block_cycle++){
     ...
    // Horizontal communication
    MPI_Isend (...,row_communicator, &send_row_request);
    MPI_Irecv (..., row_communicator, &recv_row_request);
    // Vertical communication
    MPI_Isend (...,column_communicator, &send_column_request);
    MPI_Irecv (..., column_communicator, &recv_column_request);
     ...
    // computation heavy part
     ...
    MPI_Wait(&send_row_request, &status);
    MPI_Wait(&send_column_request, &status);
    MPI_Wait(&recv_row_request, &status);
    MPI_Wait(&recv_column_request, &status);
     ...
}
```

Scaling



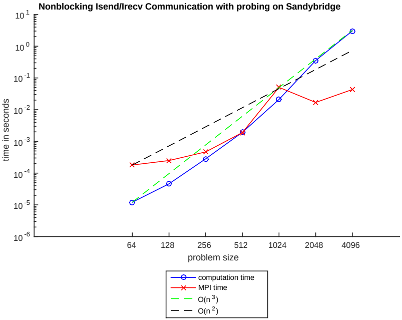
Nonblocking Isend/Irecv Communication on Sandybridge

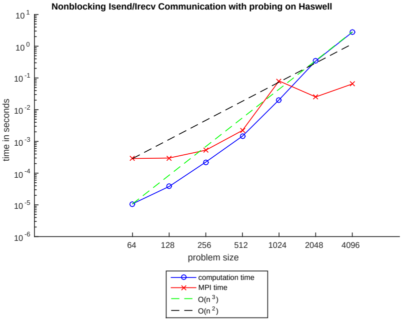


Nonblocking Isend/Irecv Communication on Haswell

- Only Little speedup.
- No big differences between Haswell and Sandybridge.

Scaling (contd.)


Nonblocking Isend/Irecv Communication with probing on Sandybridge


Nonblocking Isend/Irecv Communication with probing on Haswell

- using `MPI_Probe`
- big speedup
- no big differences

1. Provided Implementation and Baseline

1.1 Cannon’s algorithm

1.2 Baseline

1.3 Scaling

2. MPI Point-to-Point Communication

2.1 MPI Non-Blocking Operations

2.2 Optimizations

2.3 Scaling

3. MPI One-Sided Communication

3.1 MPI One-Sided Operations

3.2 Optimizations

3.3 Scaling

4. Overall Comparison

MPI One-Sided Operations

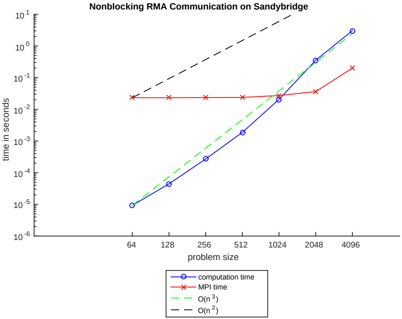
- Initialization
  - **MPI_Win_create**
  - **MPI_Win_free**
- Remote Memory Access
  - **MPI_Put**
  - MPI_Get
  - MPI_Accumulate
- Synchronization
  - **MPI_Win_fence**
  - MPI_Win_post / MPI_Win_start / MPI_Win_complete / MPI_Win_wait
  - MPI_Win_lock / MPI_Win_unlock

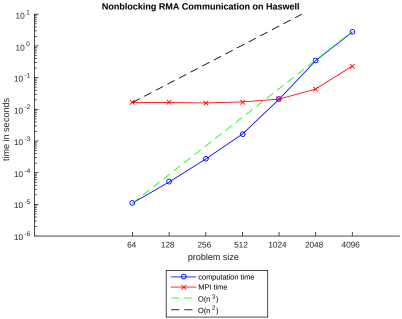
Optimizations

Was communication and computation overlap achieved?

```
// cannon's algorithm
 ...
MPI_Win_fence(0,win_A_even);
 ...
for(cannon_block_cycle = 0; cannon_block_cycle < sqrt_size; cannon_block_cycle++){
    ...
   if(cannon_block_cycle%2==0){
      MPI_Win_fence(0,win_A_even);
      A_local_block  = A_local_block_even;
      MPI_Win_fence(0,win_B_even);
      B_local_block  = B_local_block_even;
      //Horizontal communication
      MPI_Put(...,  win_A_odd);
      //Vertical communication
      MPI_Put(...,  win_B_odd);
   }else{
       ... // odd and even are exchanged
   }
    ...
   // computation heavy part
}
```

Scaling

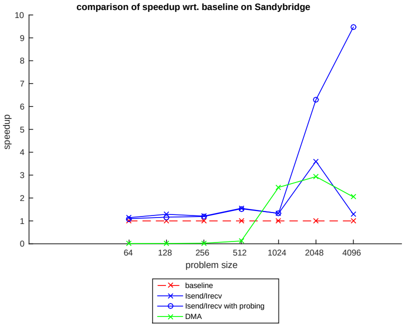


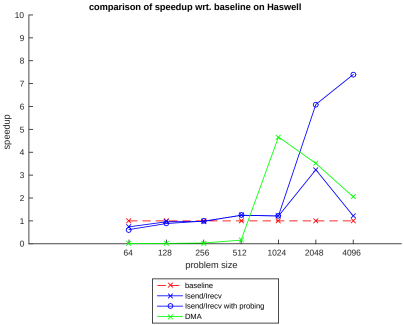


- Good speedup for big size, very high overhead for small size.
- No big differences between Haswell and Sandybridge.

1. **Provided Implementation and Baseline**
   - 1.1 Cannon’s algorithm
   - 1.2 Baseline
   - 1.3 Scaling
2. **MPI Point-to-Point Communication**
   - 2.1 MPI Non-Blocking Operations
   - 2.2 Optimizations
   - 2.3 Scaling
3. **MPI One-Sided Communication**
   - 3.1 MPI One-Sided Operations
   - 3.2 Optimizations
   - 3.3 Scaling
4. **Overall Comparison**

Speedup of communication time


comparison of speedup wrt. baseline on Sandybridge


comparison of speedup wrt. baseline on Haswell

- **Nonblocking:** Usually little speedup, but big speedup for 2048 × 2048.
- **Nonblocking with probing:** High speedup for big problems.
- **DMA:** High overhead, but good speedup for big problems.