DAY 1: SUPERCOMPUTING
Supercomputer architecture and MPI primer

2021-02-01 | Stefan Kesselheim | Helmholtz AI @ JSC

JÜLICH
Forschungszentrum

OUTLINE

Parallelization Strategies

Parallelization with MPI

Tutorial Tasks

METRICS

Strong Scaling

Strong Scaling: How does the time to solve **the same** problem t_N on the number of compute units?

$$\text{Speedup} = \frac{t_1}{\tau_N}$$

Weak Scaling

Weak Scaling: How does the time to solve **a proportionally larger** T_N problem on the number of compute units?

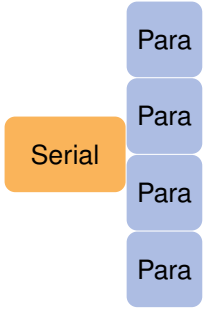
$$\text{Speedup} = N\frac{t_1}{T_N}$$

Ideally, Speedup $= N$.
Better look at efficiency: Speedup$/N$.

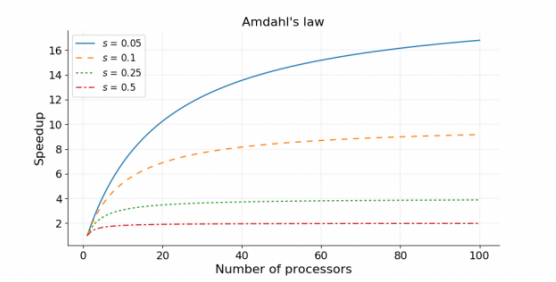
AHMDAHLS LAW OF STRONG SCALING

N=1

N=4



$$\text{speedup} = \frac{1}{s + p/N}$$

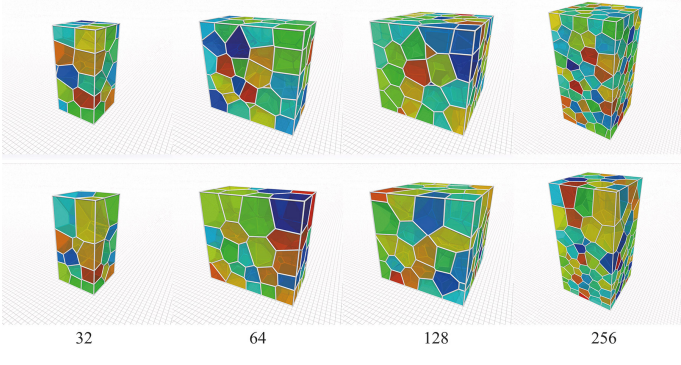


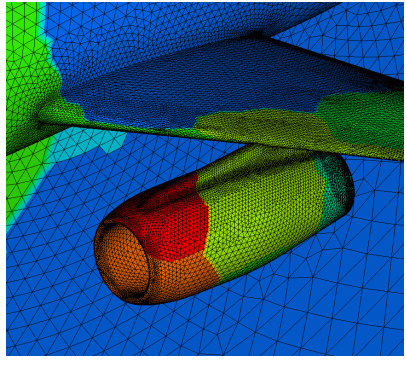
Attention

Strong scaling is difficult!

DOMAIN DECOMPOSITION

- The computational domain is composed into *N* spatial regions.
- Data (e. g. flow fields, particle forces) at the boundary is communicated
- Load Balancing requires adaptive decompositions.

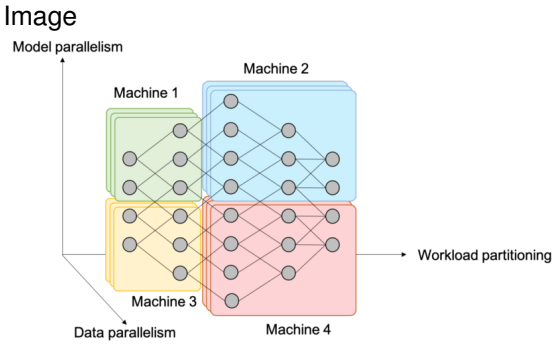


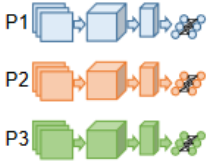


PARALLELISM IN ML

Model parallelism: Concurrent execution of different parts of the model
Data parallelism: Compute units perform calculation of different data
Layer pipelining: Different layers on different compute units.

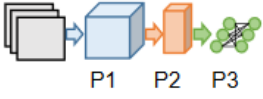
Image





(a) Data Parallelism

(b) Model Parallelism



(c) Layer Pipelining

Ben-Nun et al., ACM Computing Surveys 52, 2019.

MESSAGE PASSING INTERFACE

- MPI is a communication protocol, including API definition.
- Different MPI implementations are available, most notable MPICH and OpenMPI.
- Defined on C/Fortran level.
- Supercomputers have custom MPI installations where communication strategies are optimized. Use it.
- MPI knows about numerical datatypes, arrays and structs, but no classes.

Envelope

PARALLELIZATION WITH MPI

- In practice, almost all MPI programs are SPMD
- SPMD: Single Program Multiple Data
- Communicator: Abstraction of processes and topology. `MPI_COMM_WORLD` is the default global communicator.
- Rank: Linear number in Communicator.
- Point-to-Point communication: One-to-one communication
- Collective communication: Many-to-many communication
- Blocking- and non-blocking versions.

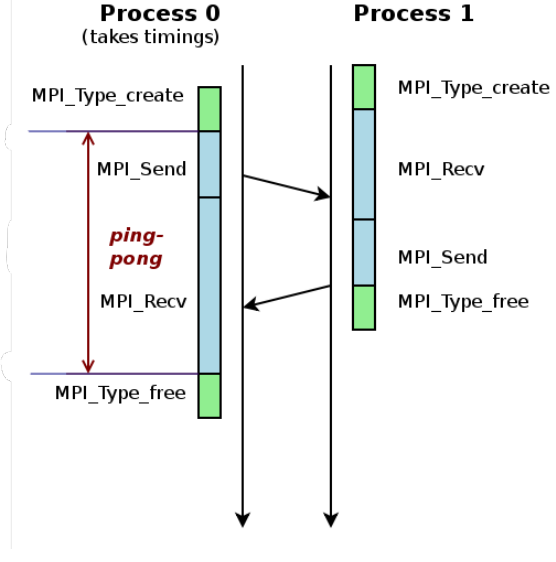
```c
#include "mpi.h"
#include <stdio.h>

int main( int argc, char *argv[] )
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    printf( "I␣am␣%d␣of␣%d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

COLLABORATIVE BLOCKING COMMUNICATION

Code

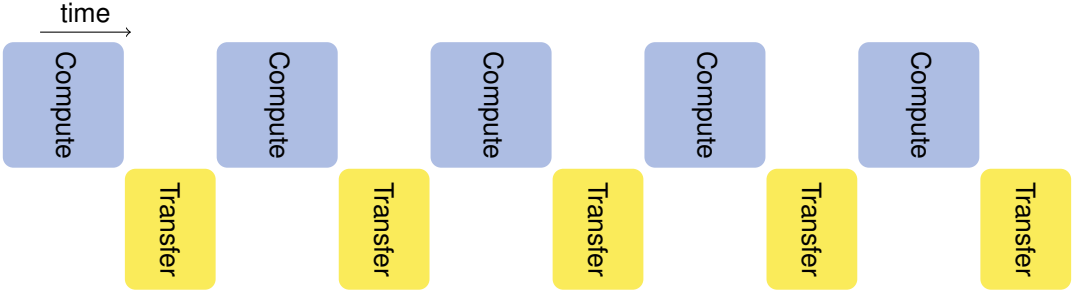
```c
#include "mpi.h"
#include <stdio.h>

int main( int argc, char *argv[] )
{
    int rank, size, data;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    if (rank==0) {
        data=7;
        mpi_send(&data, 1, 1, MPI_INT,
            MPI_COMM_WORLD);
    } else {
        mpi_recv(&data, 0, 1, MPI_INT,
            MPI_COMM_WORLD);
    } // Now both are synched.
    if (rank==0) {
        mpi_recv(&data, 1, 1, MPI_INT,
            MPI_COMM_WORLD);
    } else {
        data+=3;
        mpi_send(&data, 0, 1, MPI_INT,
            MPI_COMM_WORLD);
    }
    MPI_Finalize();
    return 0;
}
```

Process



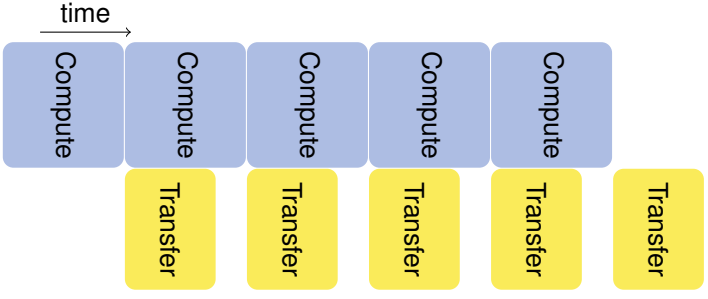
ASYNCHRONOUS COMMUNICATION

- Asynchronous Communication: Continue computation while transfer is being processed.
- Can lead to ideal latency hiding: No time is lost by transfer.
- Can that work ideally for Data-parallel training?

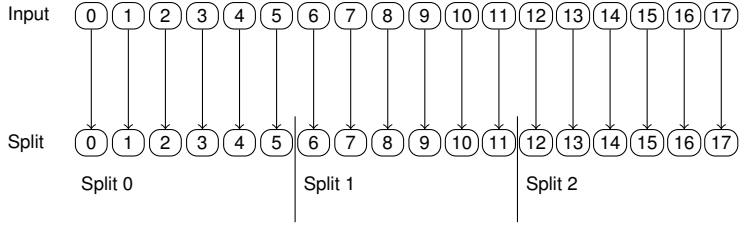


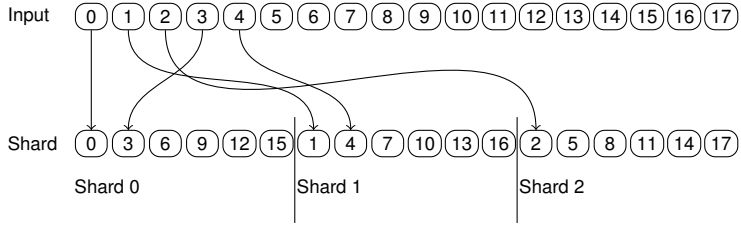
ASYNCHRONOUS COMMUNICATION

- Asynchronous Communication: Continue computation while transfer is being processed.
- Can lead to ideal latency hiding: No time is lost by transfer.
- Can that work ideally for Data-parallel training?

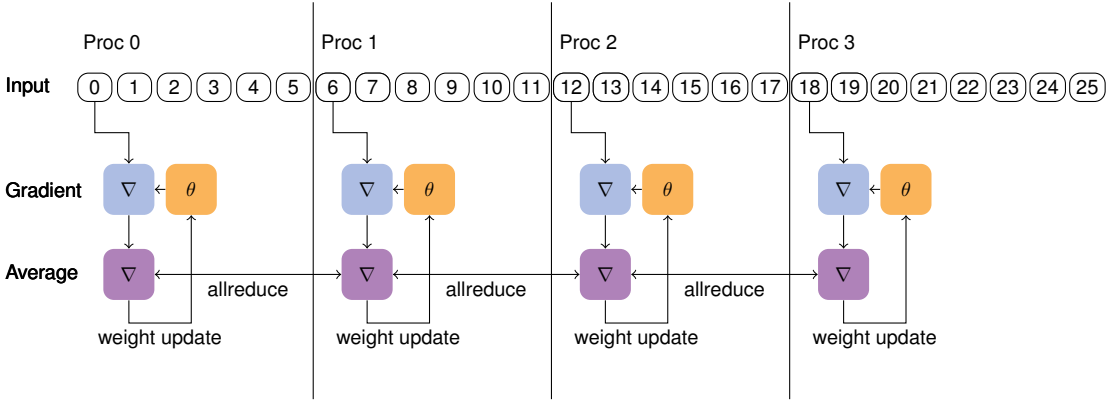


DATA SHARDING





DATA-PARALLEL GRADIENT DESCENT



TUTORIAL TASKS

1. Parallel Hello World
2. Parallel computation of π
3. Dataset sharding.