The Usual Way of Parallelization
The algorithm is sprinkled with parallelization blocks

- MPI
- Threading/Tasking
- ILP/Unrolling
- Vectorization
- GPU-Offloading
Layered Approach
Parallelization is hidden in different layers

- MPI
- Threading/Tasking
- ILP/Unrolling
- Vectorization
- GPU-Offloading

Software Stack

- Application I: Gromacs
- Coulomb Library: FMSolv
- Tasking Library: Eventify
- Comm. Library: MPI
- Laptop
- Desktop
- Cluster
- Exascale HW
### CPU Tasking Framework

**Task life-cycle per thread**

![Diagram showing the CPU Tasking Framework](image)

- Tasks can be computation or communication tasks
- Tasks can be prioritized by task type
- Only ready-to-execute tasks are stored in queue
- Workstealing from other threads is possible

### Adding Inter-node Communication via MPI

![Diagram showing inter-node communication via MPI](image)

Rationale: writing to data structure should not be concurrent → avoid critical sections
MPI Details

Code

```c
while (notFinished()){
    executeTask();
    Status = Communicator.Iprobe();
    if (Status.MessageNeeded()){
        Communicator.Irecv();
    } else {
        Communicator.Discard();
    }
    /*do something else */
    Communicator.Wait();
    /*use received data */
}
```

MPI Calls

- `Isend` to all ranks
- `Iprobe` busy waiting for messages
- Call `Irecv` for messages in any case
- If message not needed, write data to dummy buffer
- Call `Wait` before using data

Distinguishing Incoming Messages

**MPI Calls**

```c
MPI_I[send|recv](buf, count, datatype, [dest|source], tag, comm, request)
```

**It's in the Tag**

- MPI send / receive operations need a tag (integer)
- Information can be encoded in this tag
- Type of sent data (multipole, local moment, particle)
- Level of the corresponding box ($2^{d\cdot\text{level}}$ boxes)
- ID of box on this level
- This essentially mimics a matching probe / receive operation

```
rest (usually 27 Bits) | 3 Bits | 2 Bits
-----------------------|--------|--------
0 1 0 ... 0 1 1 | 0 1 1 | 0 1
ID | depth | type
```
Results from JURECA

\( N = 1000, p = 3, d = 3 \)

\( N = 103680, p = 10, d = 4 \)
Outlook

- Race condition in multithreaded MPI, no multithreading + MPI yet
- Handle message information more generally not via tag
- Restrict send operations to just some ranks
- Communicate data in larger chunks (defined by communication algorithm)

Questions?

💬 Please feel free to contact us via email if you have any questions.

👤 Ivo Kabadshow

✉️ i.kabadshow@fz-juelich.de