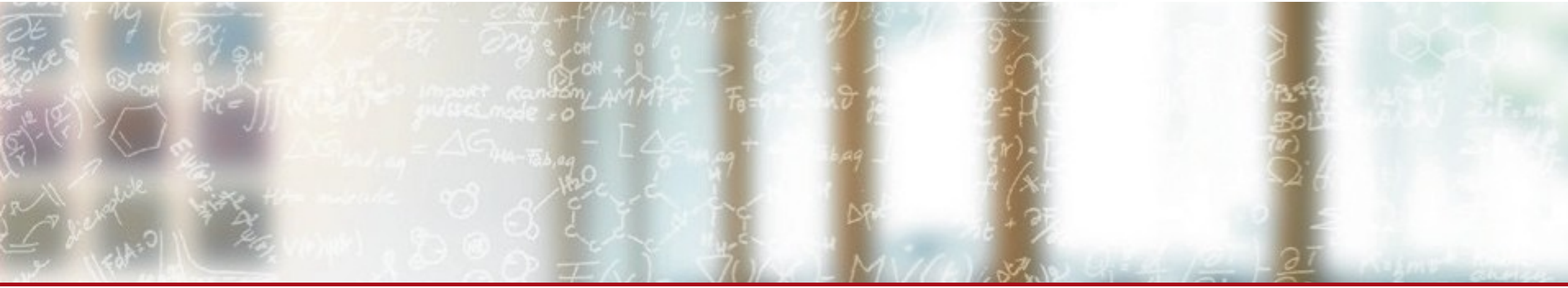


HPX at CSCS

CSCS User Lab Day

John Biddiscombe, Alberto Invernizzi, Auriane Reverdell, Alo Roosing, Mikael Simberg, Raffaele Solca, CSCS

September 9, 2019

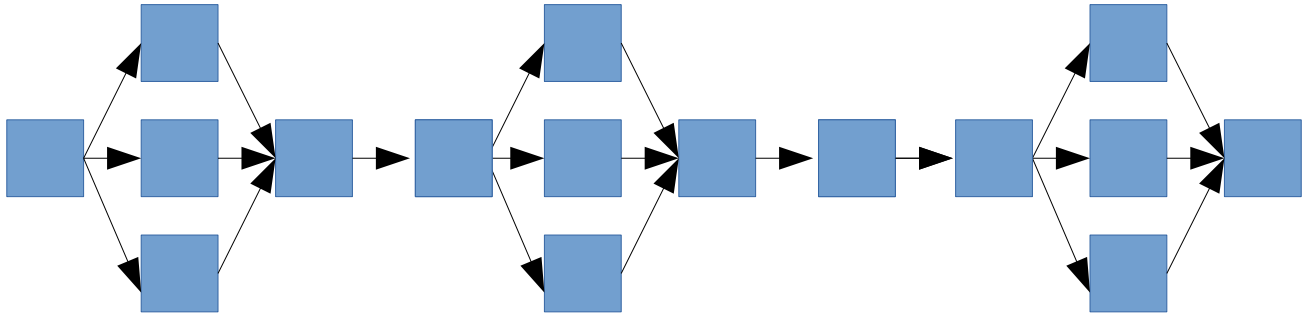
Challenges in HPC

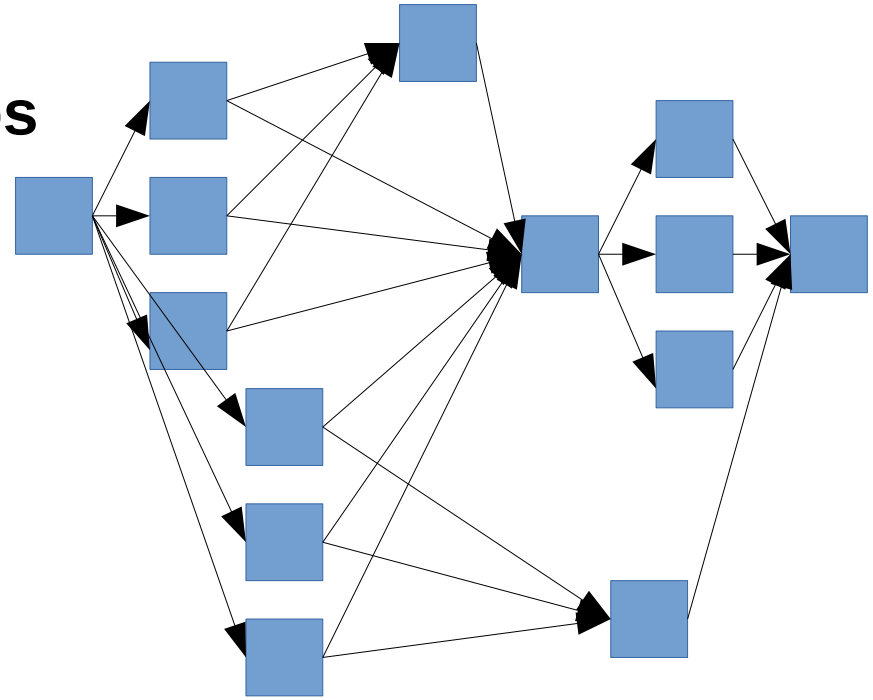
Challenges

- Hardware increasingly complex
  - Multicore, hybrid, multi-GPU
- It‘s getting more difficult to
  - utilize modern nodes with traditional programming models
  - manage work across a full node
  - keep up with different architectures
- Fork-join
  - Parallel regions and global barriers (OpenMP and MPI)
  - Amdahl‘s law
- Asynchronous task-based programming, and C++ to the rescue?

Task-based programming

- Thinking of computation in terms of tasks rather than threads
- Tasks take their minimal dependencies as inputs, produce outputs
- Removing artificial restrictions from applications
- **Giving the scheduler a chance to fill in the gaps**





The future of C++

- Why C++?
  - Increasingly popular even in HPC
  - Better tools for abstraction (templates, function overloading)
- Strong effort to standardize
  - Data-parallel algorithms (C++17 parallel algorithms)
  - Task-based programming (C++11: async, futures; C++23: executors)
  - Heterogenous systems (C++23: executors)
- Gives everyone a common foundation
- But… C++23? Really?

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

HPX

HPX

- Developed by LSU, CSCS, and many volunteer contributors
- A general purpose runtime system for parallelism and concurrency
- Exposes a C++ standards-conformant asynchronous programming model
  - `std → hpx` and `hpx → std`
- Lightweight tasking
  - Thread pools with work-stealing, load balancing on the node
  - Can spawn millions of tasks
- Best suited for dynamic, irregular parallelism
  - But regular data-parallelism maps just as well on to HPX with minor overheads
- Extends the standard syntax to the distributed setting

Tasks

```
std::future<T> f = std::async(fun, 1, 2.3);
```

Tasks

```
hpx::future<T> f = hpx::async(fun, 1, 2.3);
```

Remote tasks

```
hpx::future<T> f = hpx::async(act, loc, 1, 2.3);
```

Parallel algorithms

```
std::for_each(std::execution::par,
    x.begin(), x.end(), fun);
```

Parallel algorithms

```
hpx::for_each(hpx::execution::par,
    x.begin(), x.end(), fun);
```

Parallel algorithms

```
reduce(par, x.begin(), x.end(), fun);
```

Parallel algorithms as tasks

```
future<void> f =
    reduce(par(task)), x.begin(), x.end(), fun);
```

Parallel algorithms as tasks on GPUs

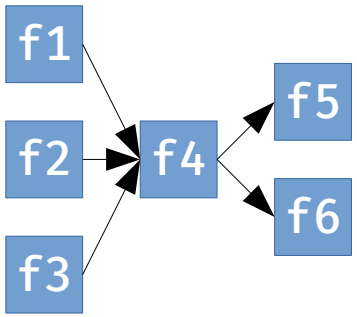
```
future<void> f =
    reduce(par(task).on(cuda::default_executor),
        x.begin(), x.end(), fun);
```

Putting it all together

```
future<double> f1 = async(…);
future<void> f2 = reduce(…);
future<int> f3 = my_algorithm(…);

shared_future<void> f4 =
    dataflow(…, f1, f2, f3);

future<void> f5 = dataflow(…, f4);
future<void> f6 = dataflow(…, f4);
```



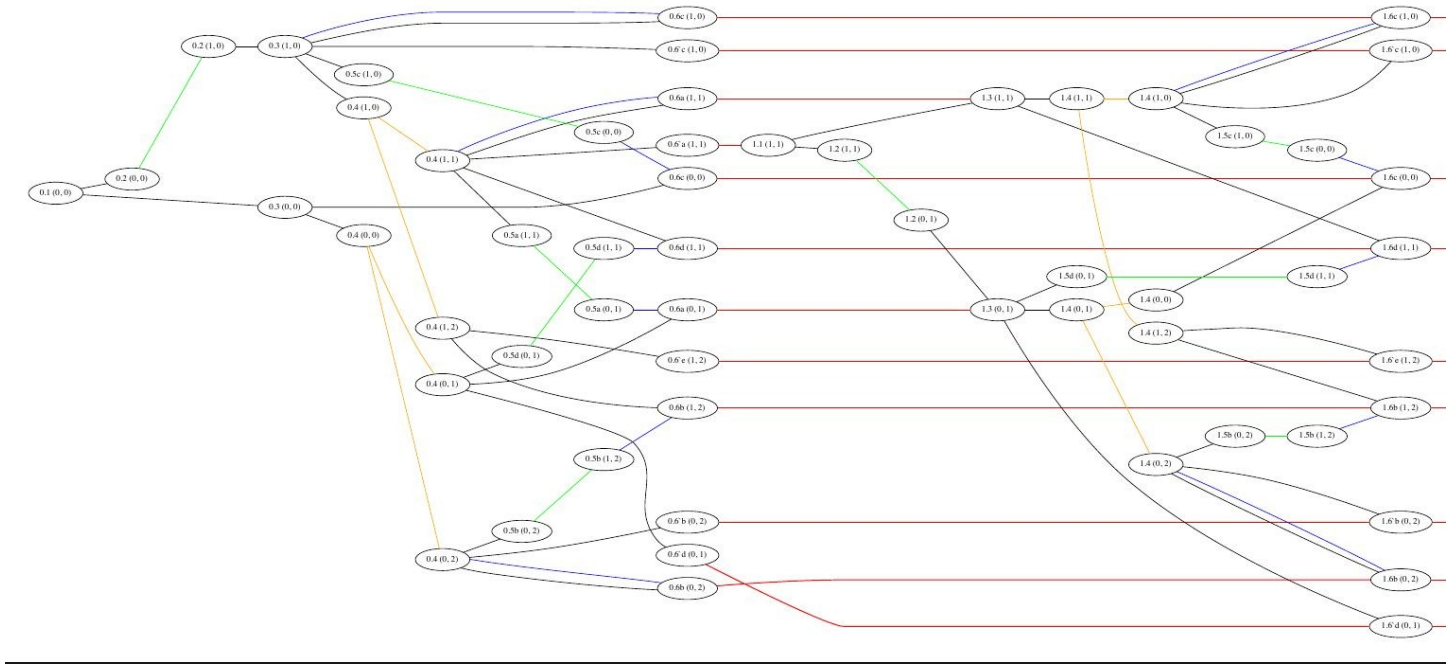
HPX

- The **whole program** becomes a parallel region
- High performance runtime which dynamically schedules tasks
- Interoperability
  - HPX runtime can be suspended and resumed like OpenMP for „parallel regions“
  - hpxMP is an OpenMP implementation based on HPX
  - Modularization effort ongoing to provide only thread pools and a single-node runtime
  - MPI can be used explicitly and overlapped with computation

Applications

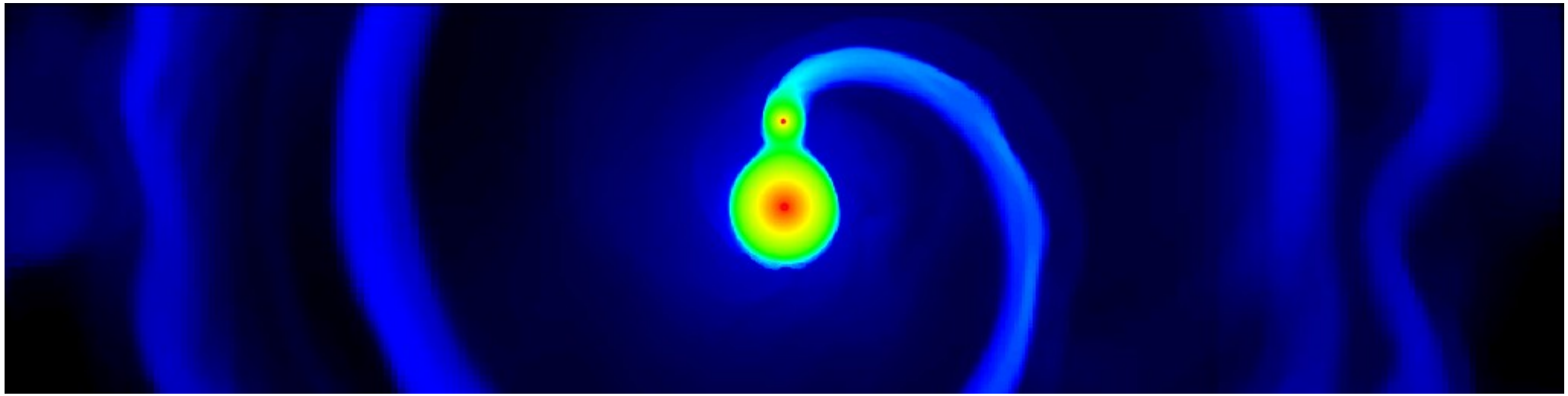
Linear algebra

- Cholesky decomposition developed at CSCS (John Biddiscombe, Alberto Invernizzi, Alo Roosing, Raffaele Solca)
- Complex dependencies expressed easily in HPX
- MPI explicitly used for communication, wrapped into asynchronous tasks



Adaptive mesh refinement

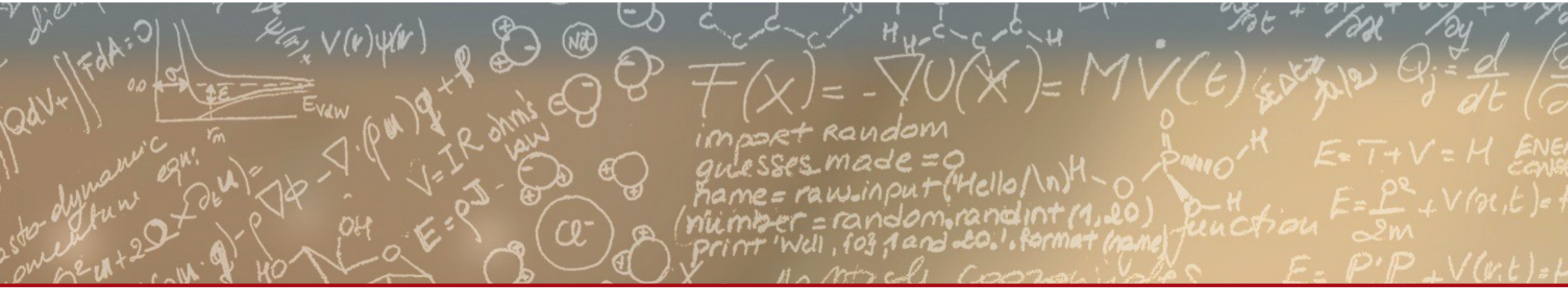
- Octotiger: simulating binary star mergers
- Collaboration between LSU, University of Stuttgart, and CSCS (John Biddiscombe)
- SC19 paper: „From Piz Daint to the Stars: Simulation of Stellar Mergers using High-Level Abstractions“



More information

More information

- HPX course: October 17-18, 2019
  - Registration closes October 9, 2019
- HPX course videos from 2016 (search Youtube for „task based programming with HPX“)
- HPX available on Piz Daint: `module load HPX`
- Documentation (https://stellar-group.github.io/hpx/)
- Contact:
  - Mailing list: hpx-users@stellar.cct.lsu.edu
  - IRC: #ste||ar on freenode
  - Slack: #hpx cpplang.slack.com



Thank you for your attention.