universität innsbruck

FZ HPC

The AllScale API

Philipp Gschwandtner, Herbert Jordan, Peter Thoman and **Thomas Fahringer**
University of Innsbruck, Institute of Computer Science, Austria

eScience 2019 Workshop on Platform-driven e-Infrastructure Innovations
September 24th 2019, San Diego

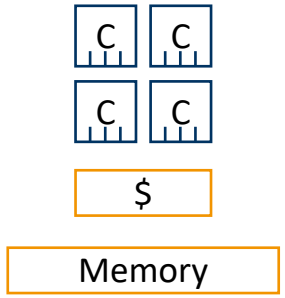
AllScale – A H2020 Success Story

- Horizon 2020 funded project and marked “Success Story”
- October 2015 – September 2018
- Coordinated by Thomas Fahringer, Distributed and Parallel Systems Group and Research Center HPC, Department of Computer Science
- 5 Partners
  - Friedrich-Alexander University Erlangen-Nürnberg
  - IBM Ireland
  - KTH Stockholm
  - Numeca, Brussels
  - Queen’s University Belfast

This project has received funding from the European Union‘s Horizon 2020 research and innovation programme under grant agreement No. 671603

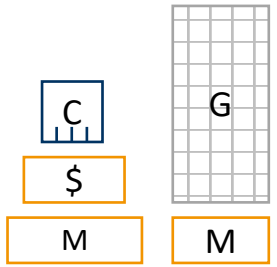
Parallel Architectures

Multicore



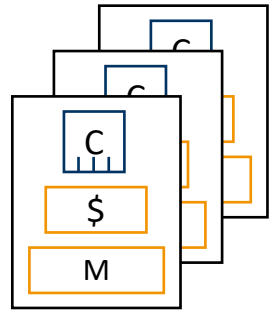
OpenMP/Cilk

Accelerators



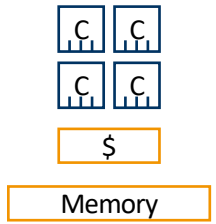
OpenCL/CUDA

Clusters

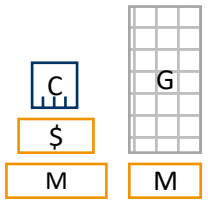


MPI/PGAS

Real World Architectures

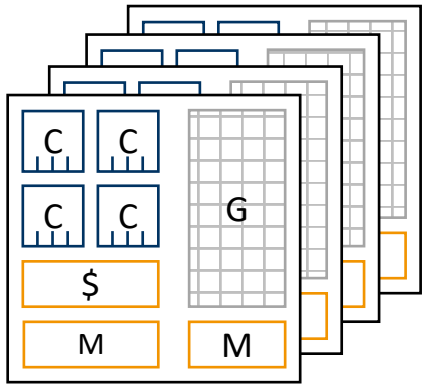


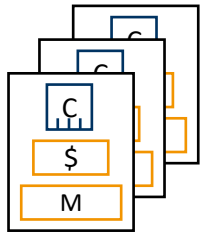
OpenMP/Cilk



OpenCL/CUDA

≠

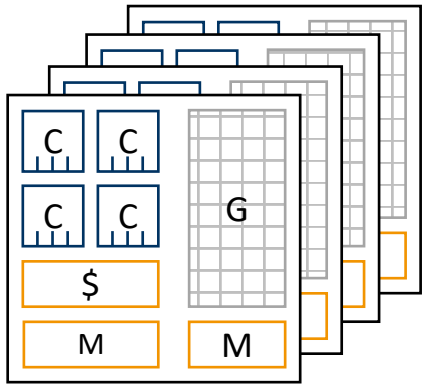




MPI/PGAS

Hybrid Codes

- Issues
  - Hard-coded problem decomposition
  - Lack of coordination among runtime systems
- No built-in support for
  - Portability
  - Auto-tuning
  - Load balancing
  - Monitoring
  - Resilience
  - …



AllScale Vision

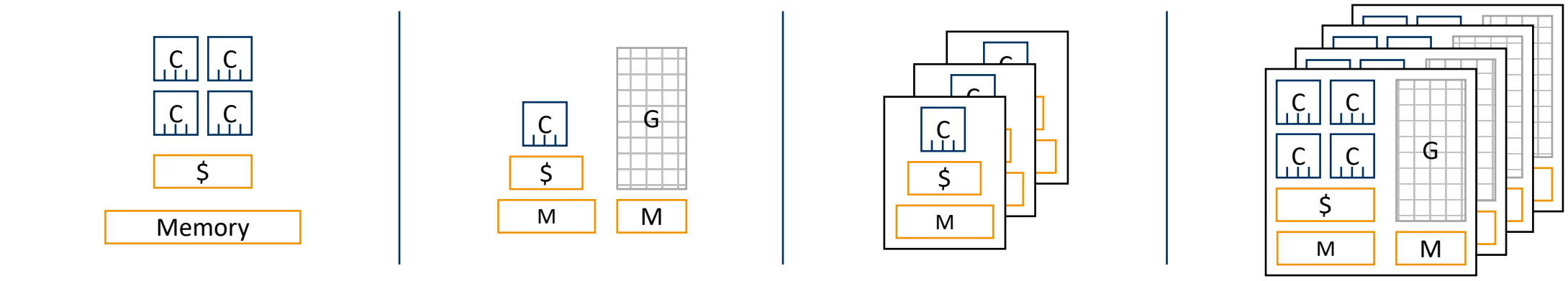
Application

Parallel algorithm

Unified parallel programming model

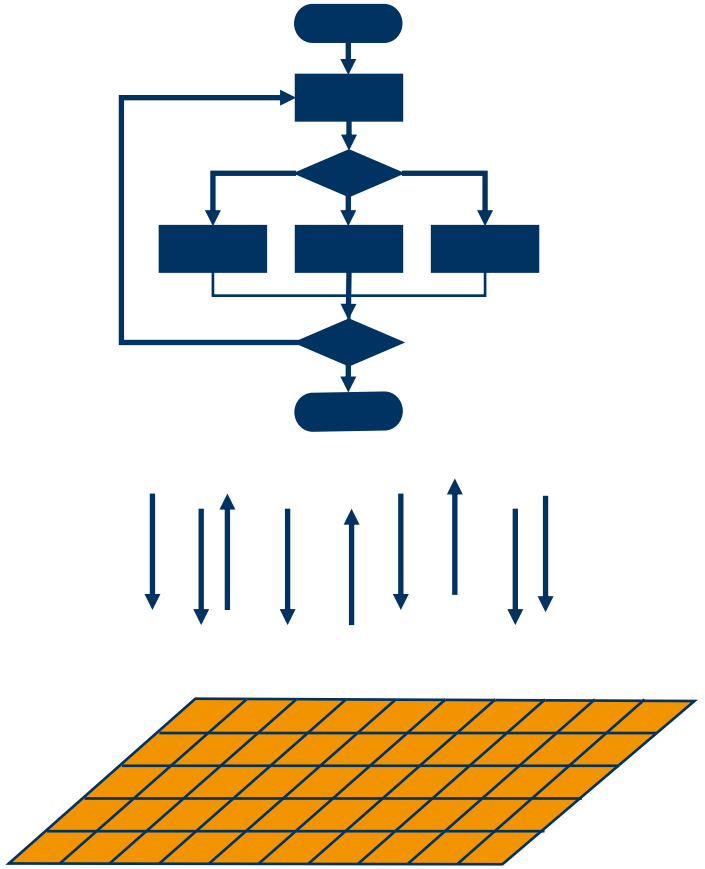
Toolchain

Portability, tuning, resilience, …



Serial Work and Data

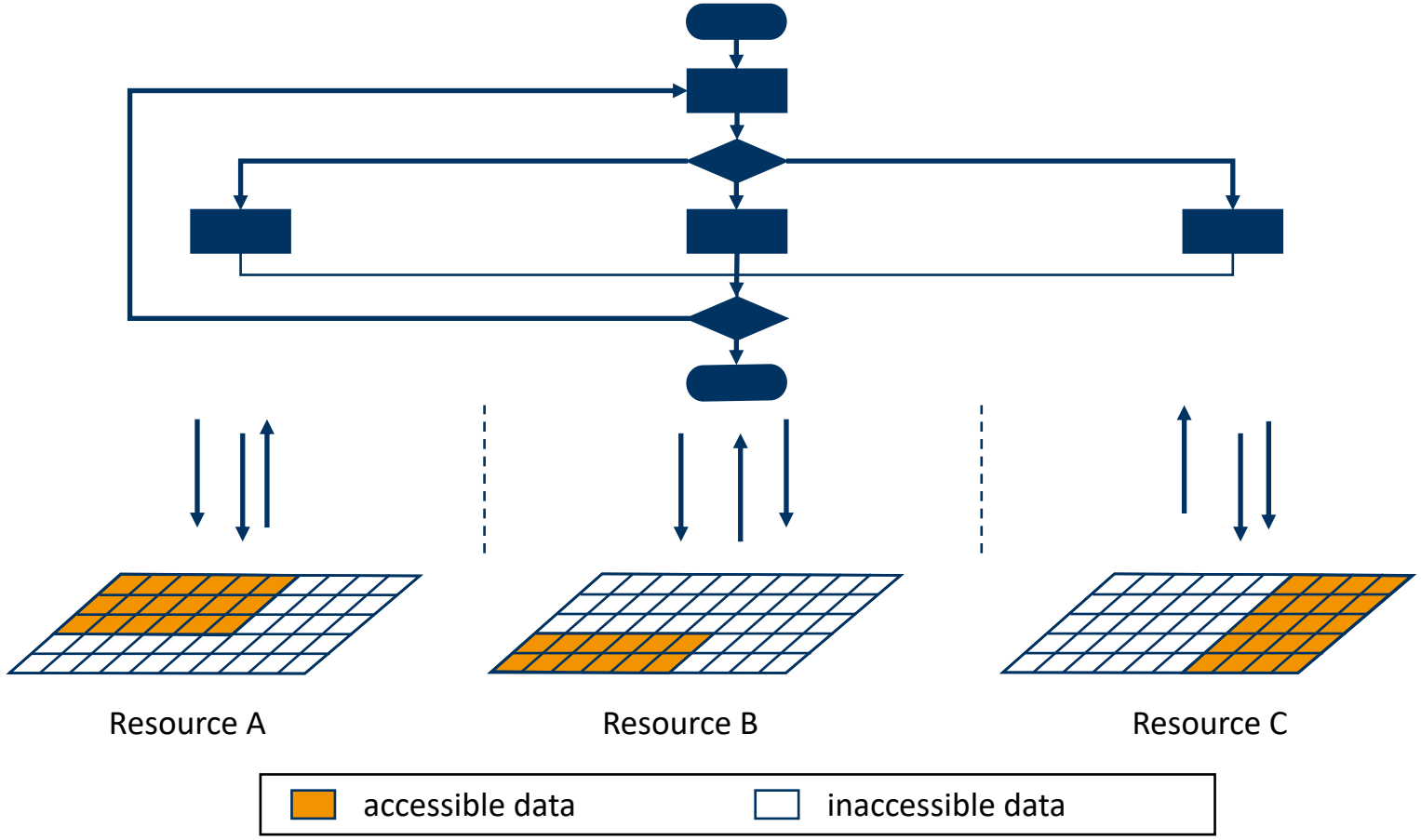
Algorithms
manipulate the
state of
data structures



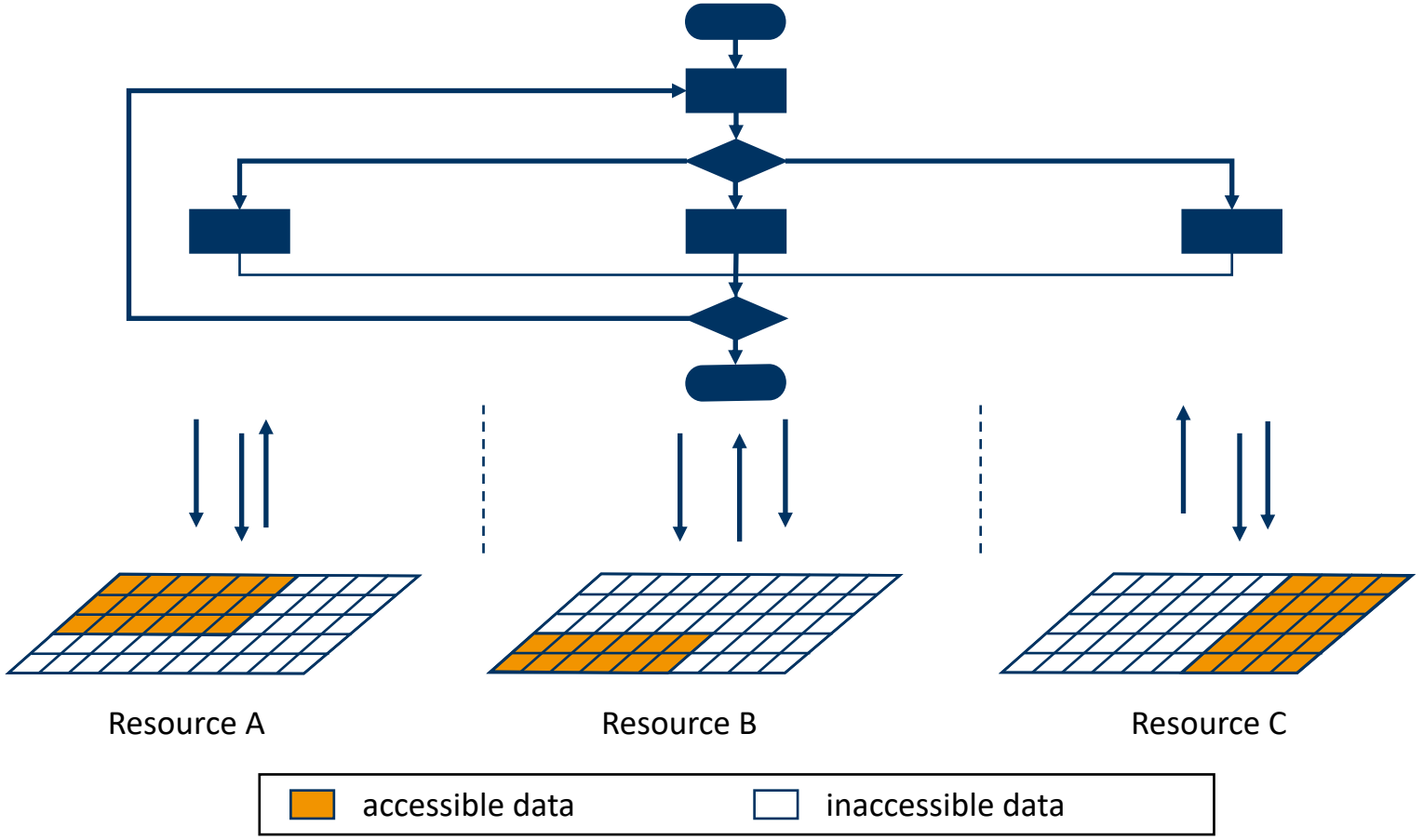
explicitly expressed
in program code

the **implicit** effect
of program code

Parallel Work and Data



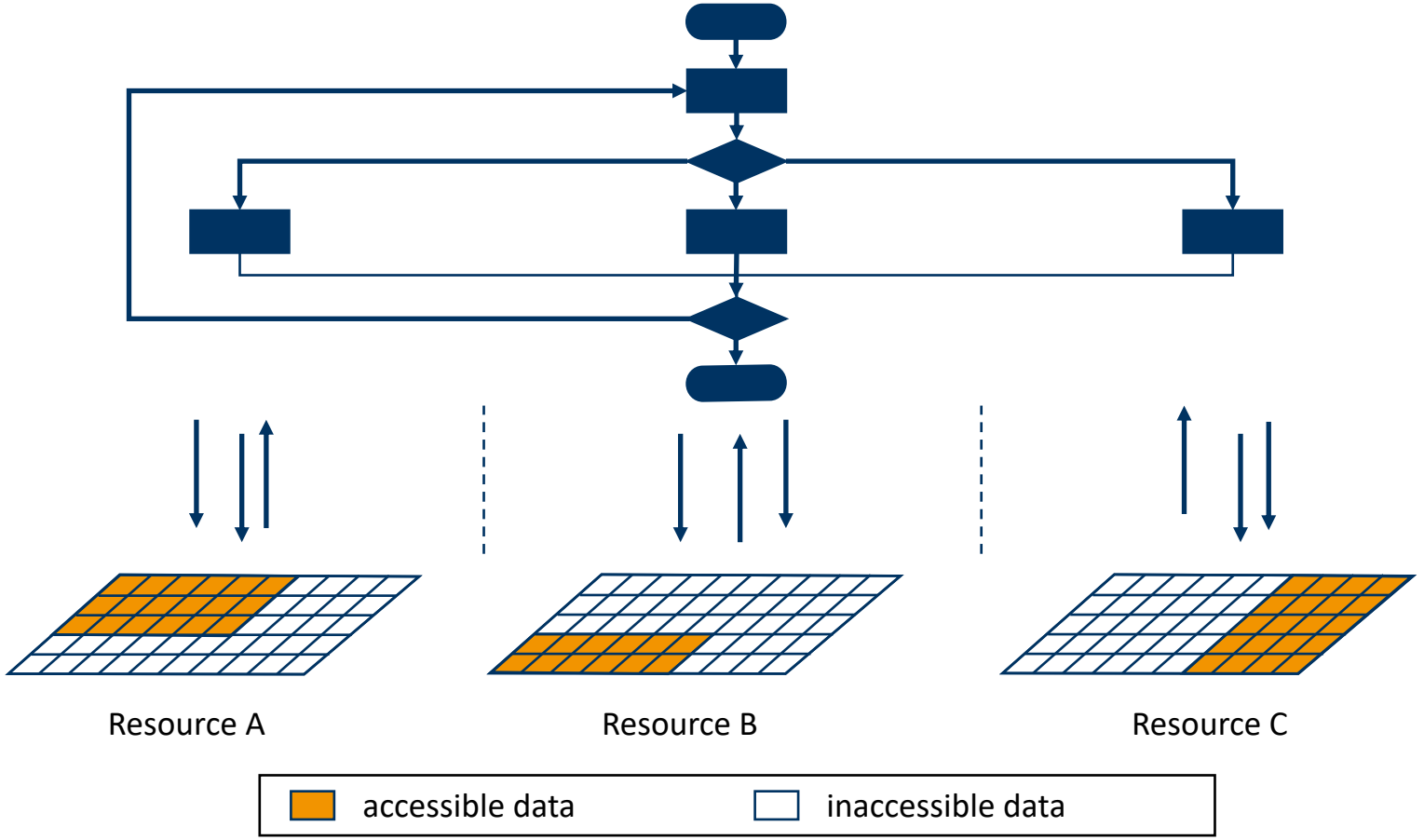
Parallel Work and Data



decomposition of **workload** is main focus of e.g. MPI, OpenMP, Cilk, OpenCL, CUDA,…

data structure decomposition and management **left to the user**

Parallel Work and Data



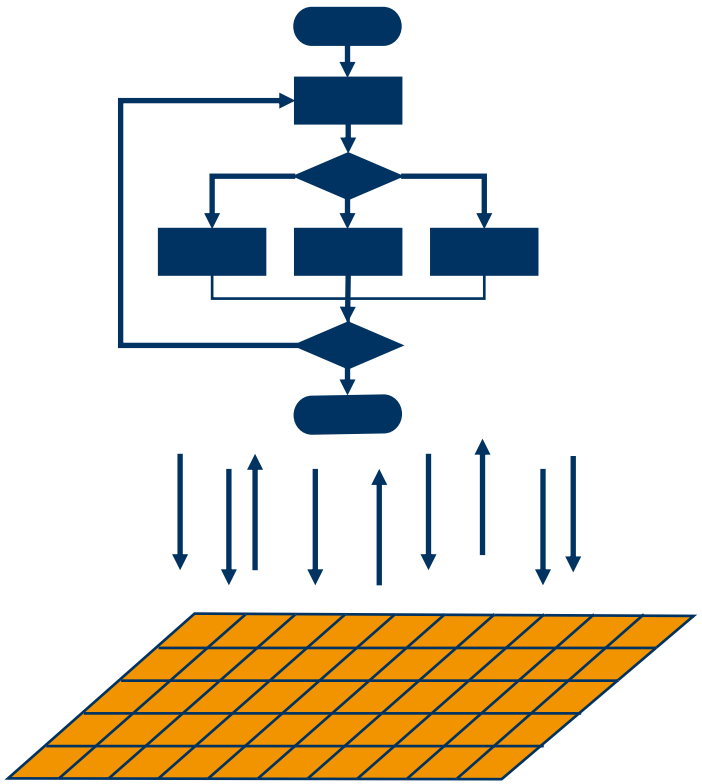
Desired:

advanced system features and services including

- performance **portability**
- inter-node **load balancing**
- **dynamic resource utilization**
- **resilience** to node failures

For those, **runtime systems** require **control** over **distribution** of **work and data**

AllScale's Approach



application

Operators:

- parallel loops
- parallel reduction
- map-reduce
- stencils
- V-cycles
- …

higher order parallel primitive

Data Structures:

- multi-dimensional grids
- adaptive grids
- maps
- trees
- hierarchical meshes
- …

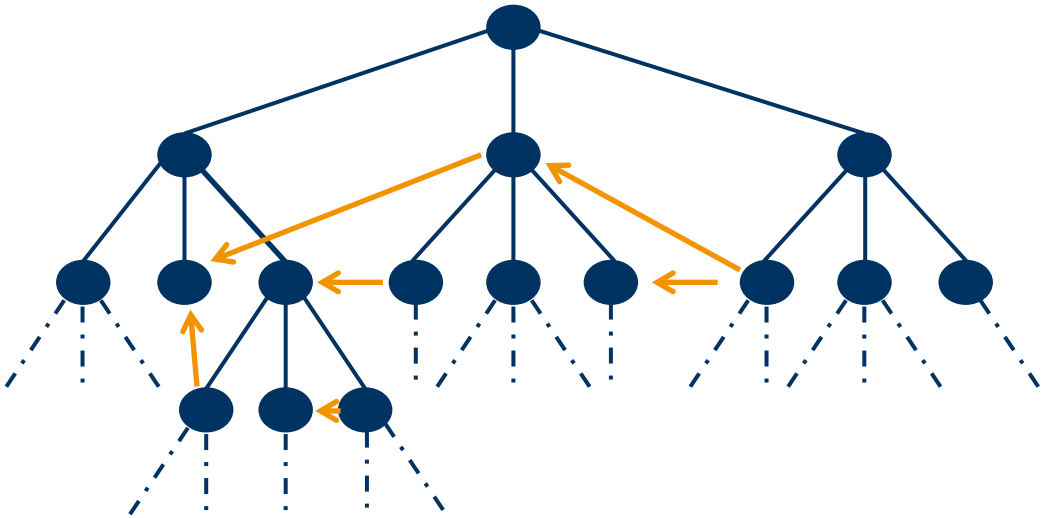
generalized abstract data structure

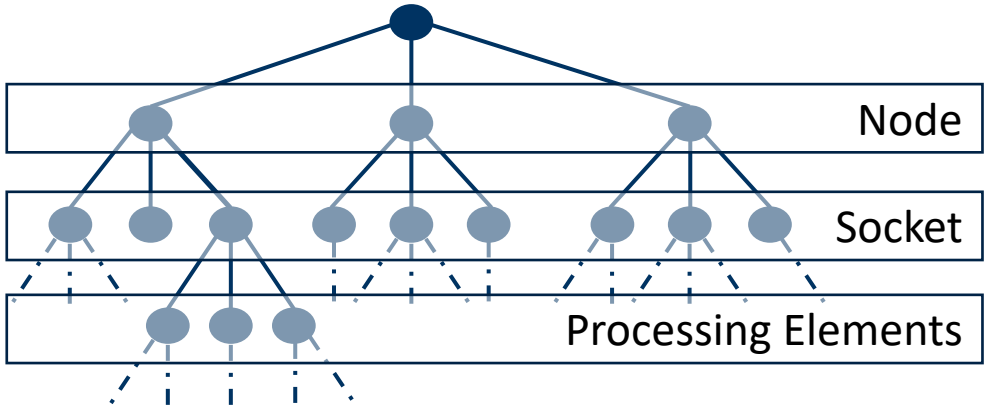
parallel constructs

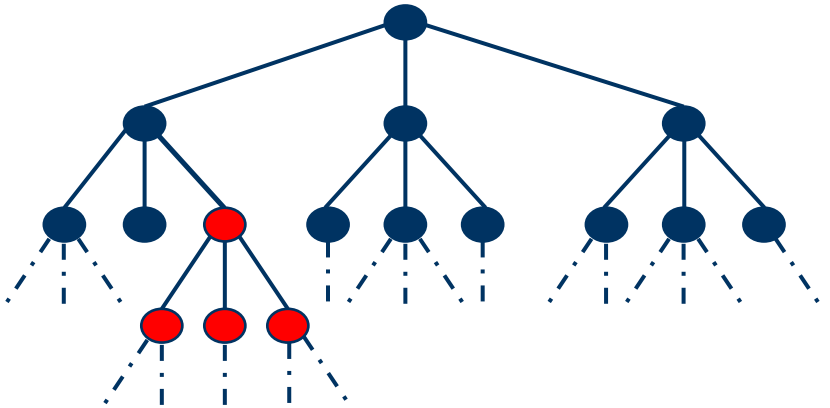
primitives for parallel construct

Work Flow Construct

- **prec** – a **higher order function** to
  - express recursive task-based parallelism
  - support fine-grained, hierarchical dependencies

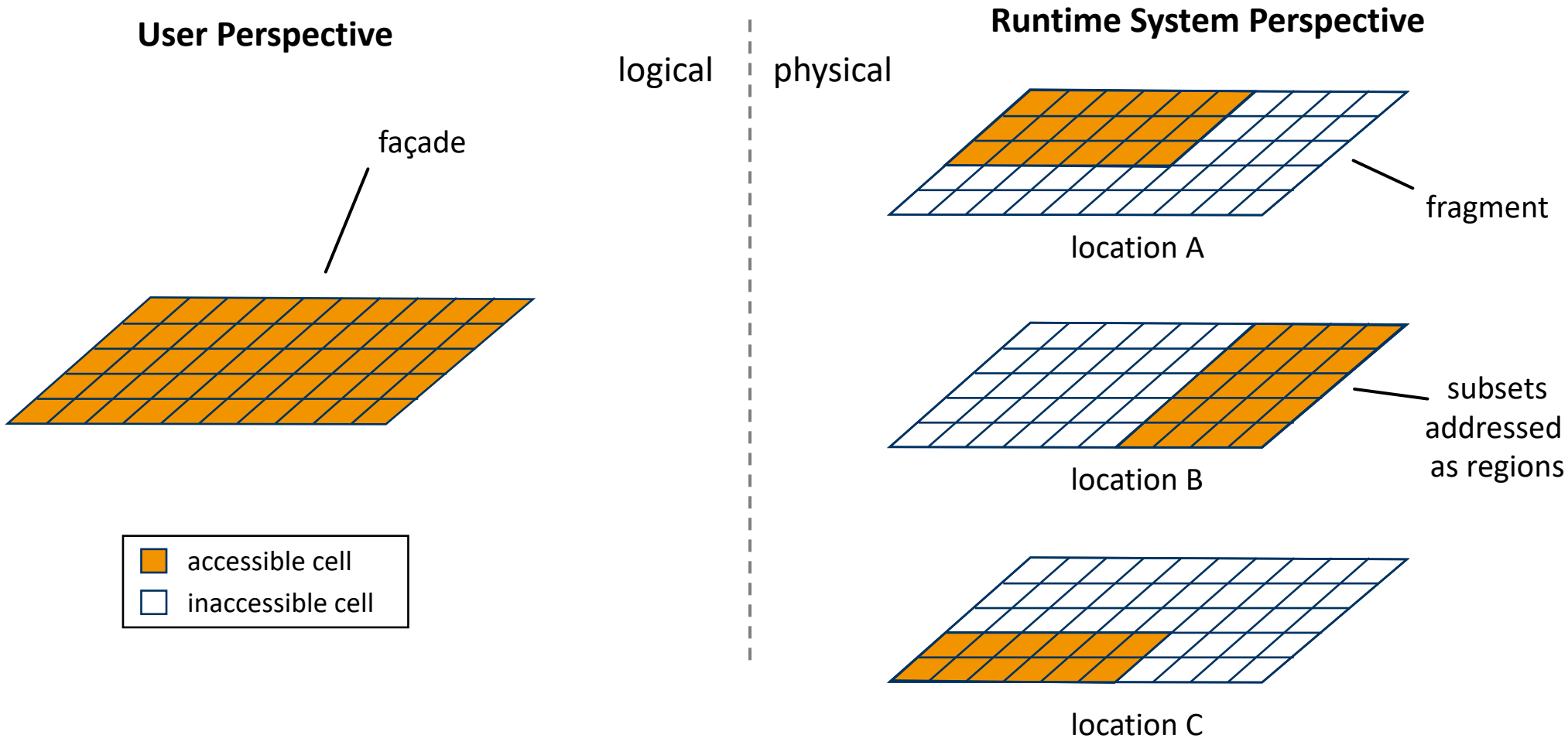




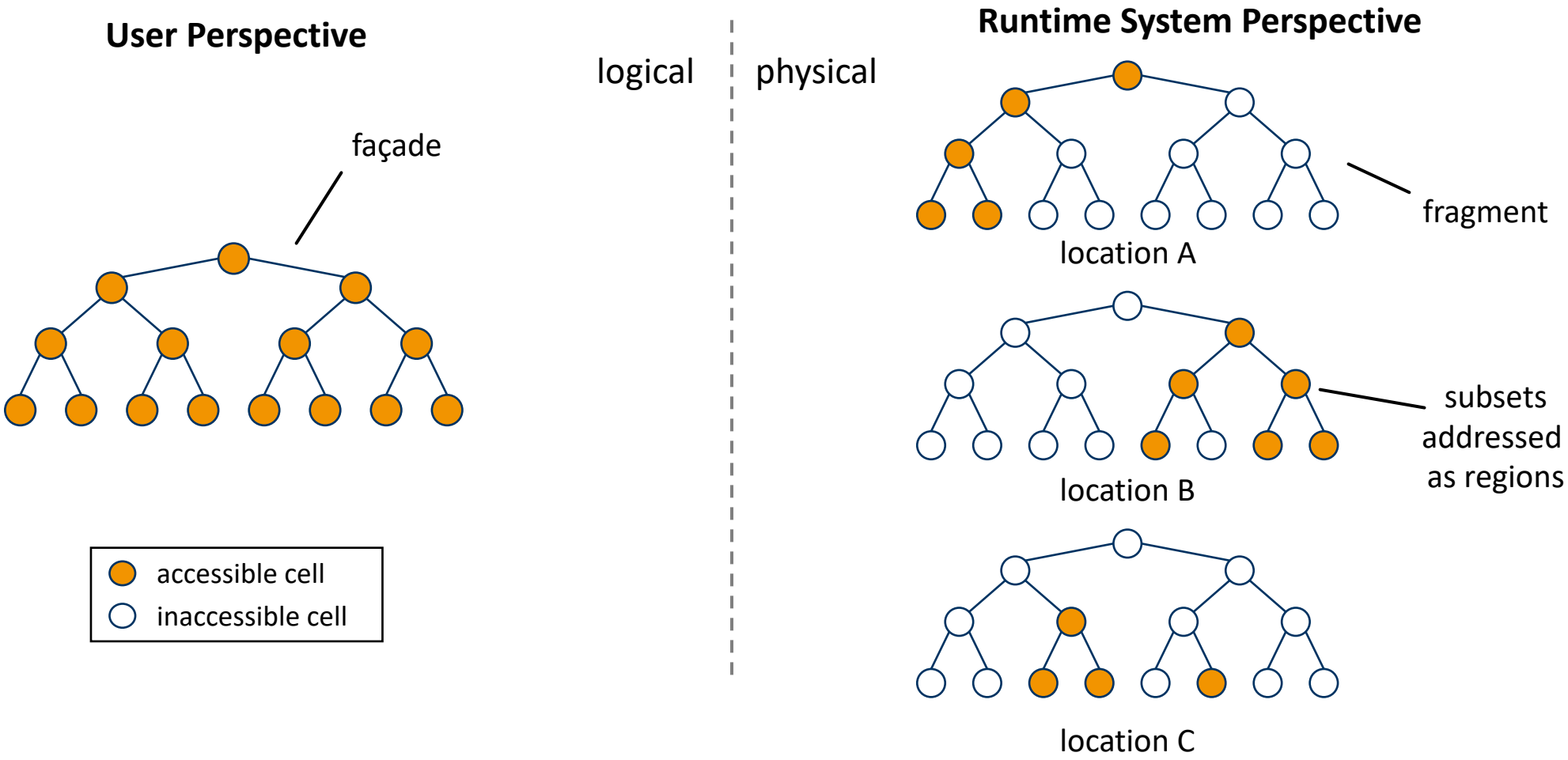


… recursive task — … task hierarchy ← … fine grained dependencies … failed task

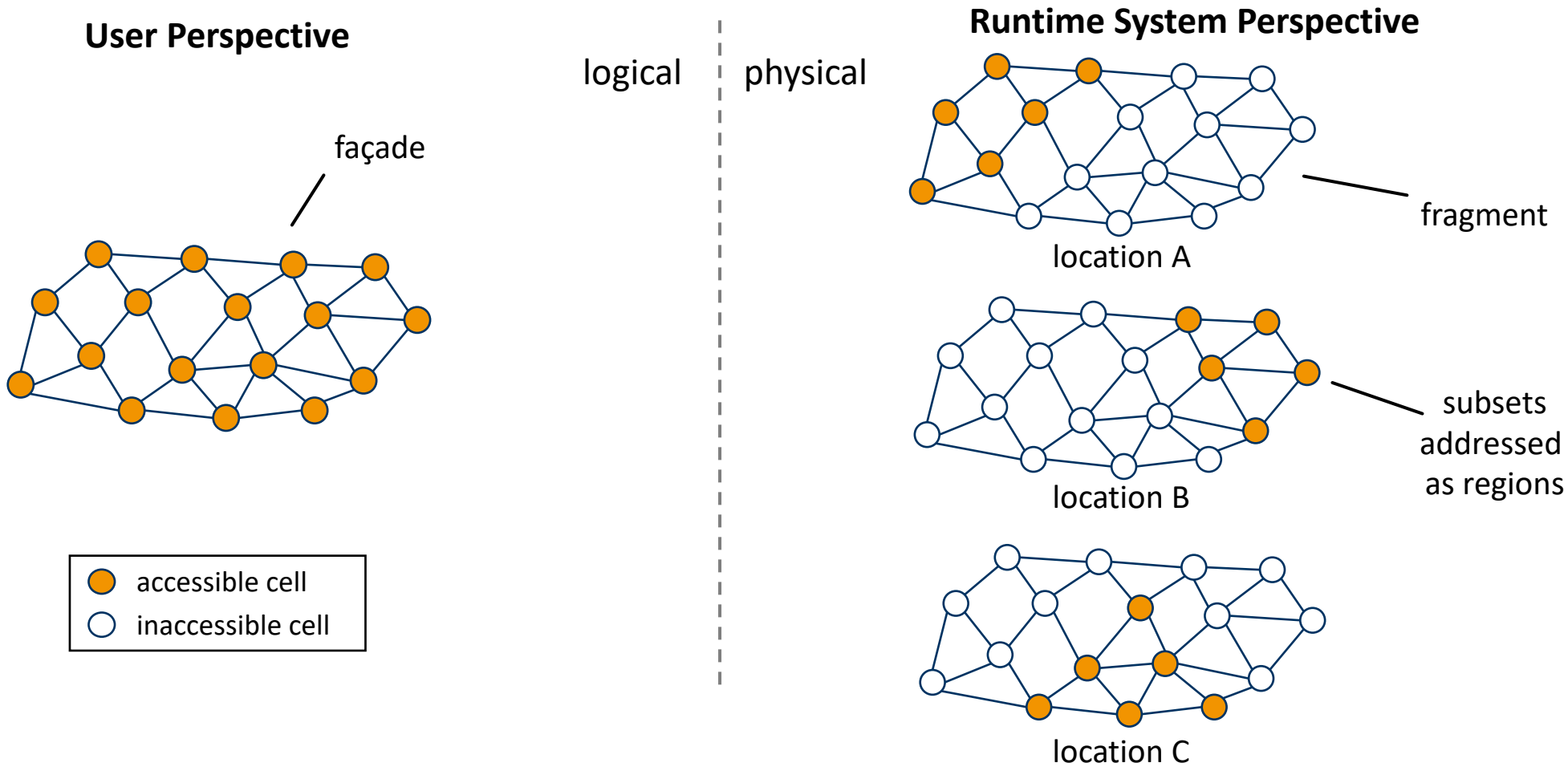
Data Items


User Perspective
logical
physical
Runtime System Perspective
façade
fragment
location A
subsets
addressed
as regions
location B
accessible cell
inaccessible cell
location C

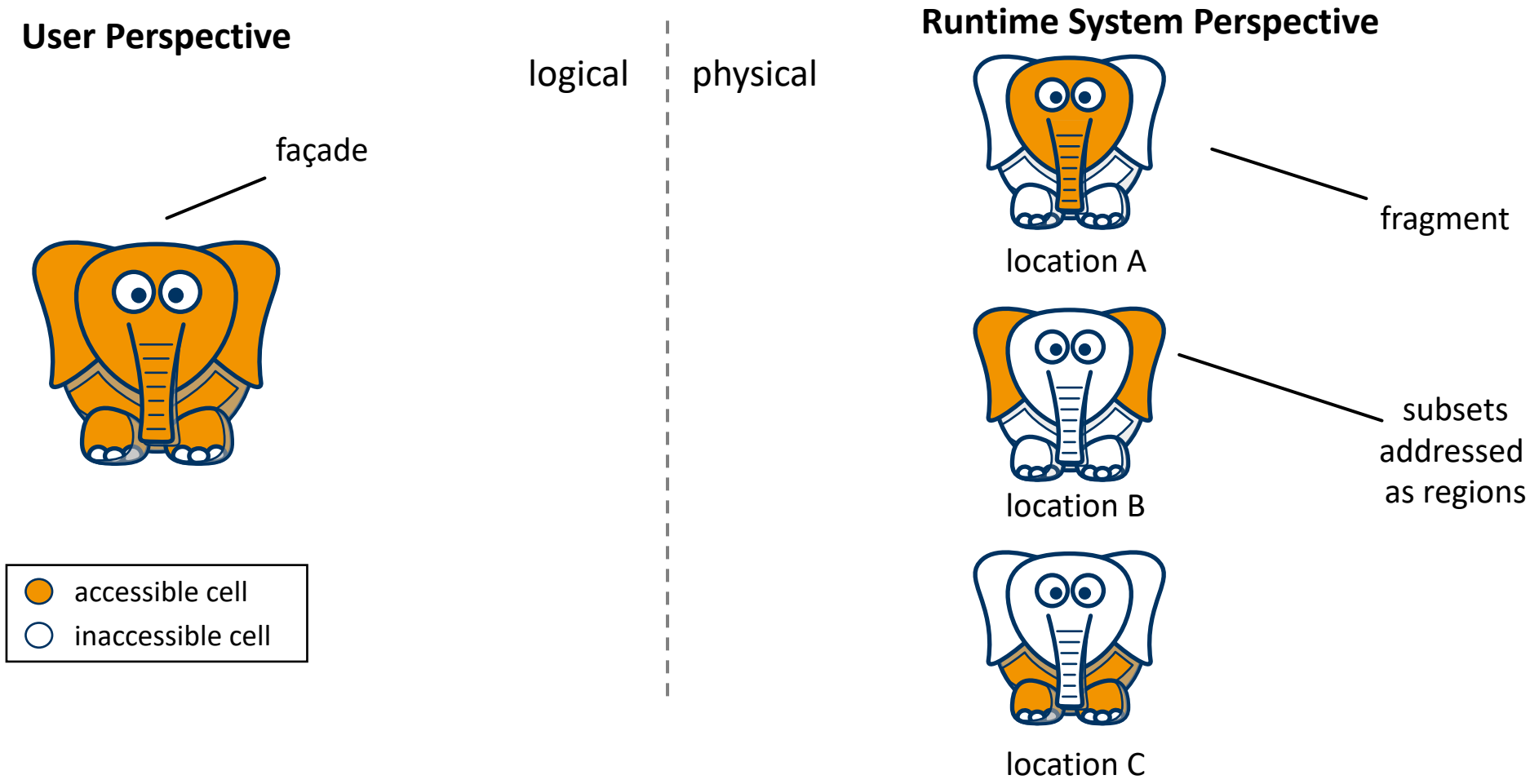
Data Items


User Perspective
façade
accessible cell
inaccessible cell
logical
physical
Runtime System Perspective
fragment
location A
location B
subsets
addressed
as regions
location C

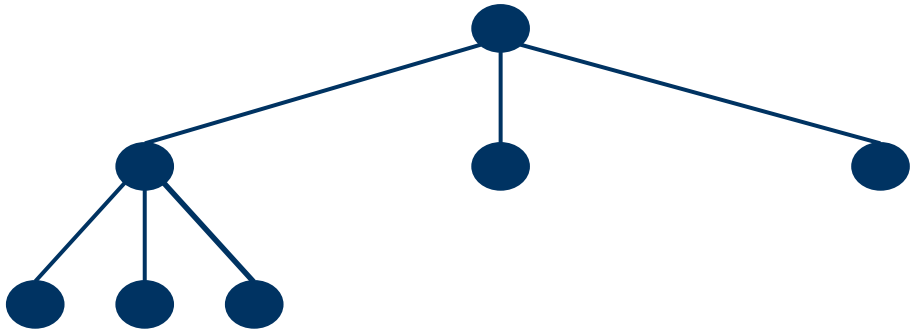
Data Items


User Perspective
logical
physical
Runtime System Perspective
façade
fragment
location A
location B
subsets
addressed
as regions
location C
accessible cell
inaccessible cell

Data Items


User Perspective
logical
physical
Runtime System Perspective
façade
fragment
location A
location B
subsets
addressed
as regions
location C
accessible cell
inaccessible cell

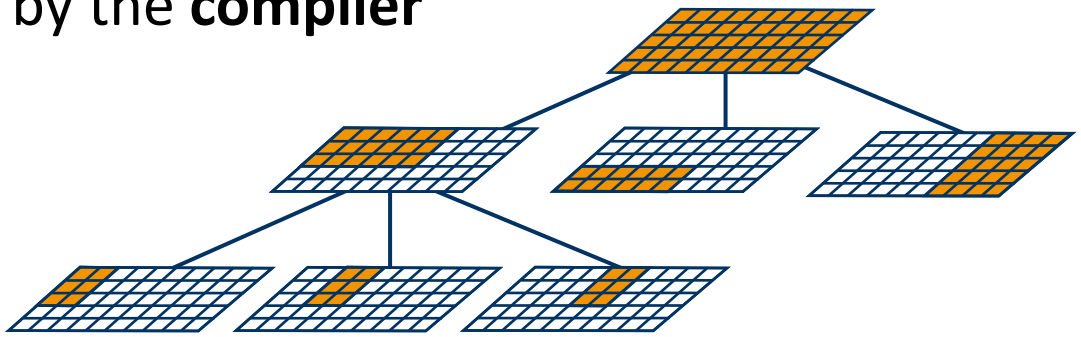
Work and Data Link



recursive work load
decomposition defined
by **algorithm**

automatically deduced
by the **compiler**

data dependencies
of tasks to be managed
by the **runtime system**



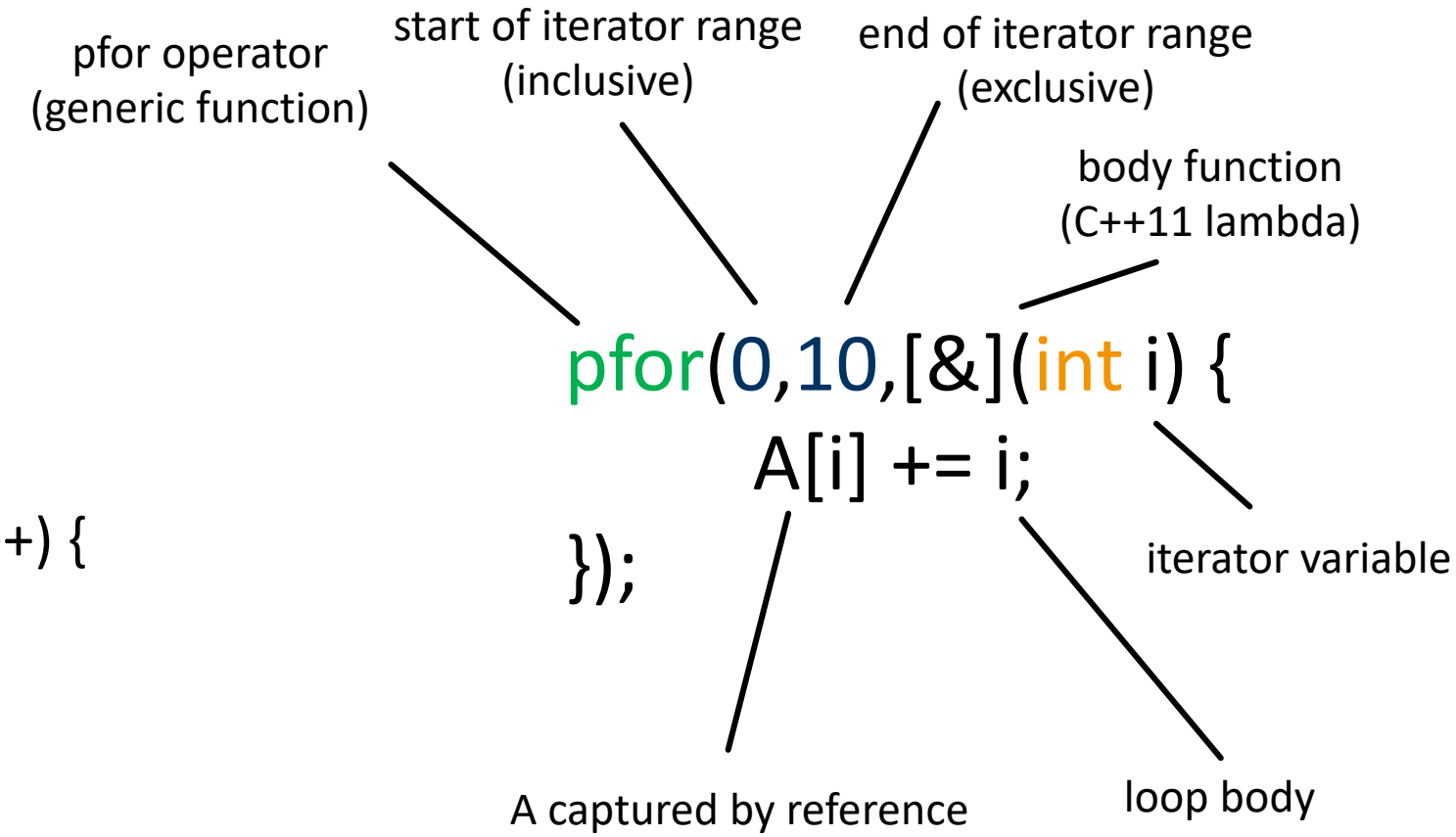
A Simple Loop

```
for (int i = 0; i < 10; i++) {
        A[i] += i;
}
```

Increments the first 10 elements of array A with values 0-9.

A Simple Parallel Loop

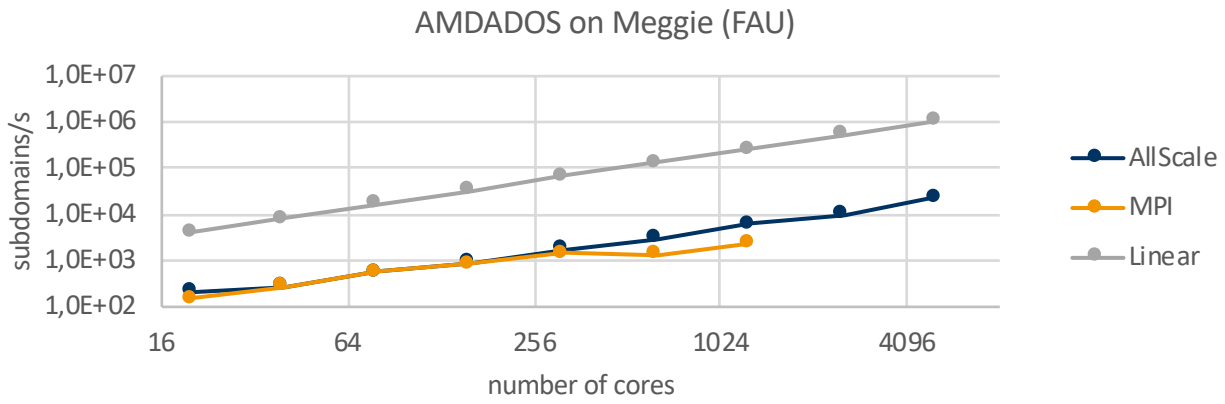
```
int rank, size;
MPI_Com_rank(COM,&rank)
MPI_Com_size(COM,&size)
int p = 10/size;
MPI_Scatter(A,…)
for (int i = p*rank; i < p*rank+p; i++) {
        A[i] += i;
}
MPI_Gather(A,…)
```

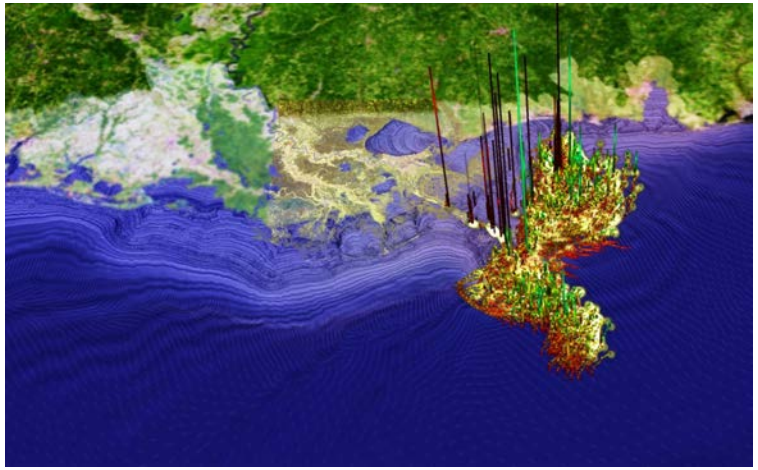


Increments the first 10 elements of array A with values 0-9 in parallel.

AMDADOS

- FetHPC H2020 project AllScale (coordinator: Dept. of Computer Science, UIBK)
- Oil spill simulation, developed with IBM Ireland
- Stencil and kalman filter for assimilating sensor data
- AllScale exceeds performance of MPI reference implementation

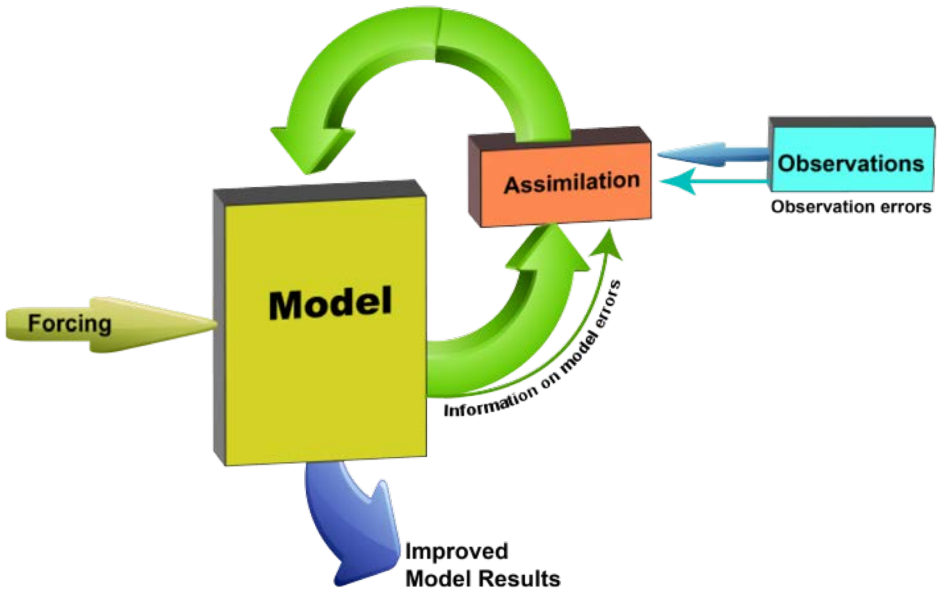




Sources: https://www.chemistryworld.com/features/oil-spill-cleanup/3008990.article, Marcel Ritter (UIBK)

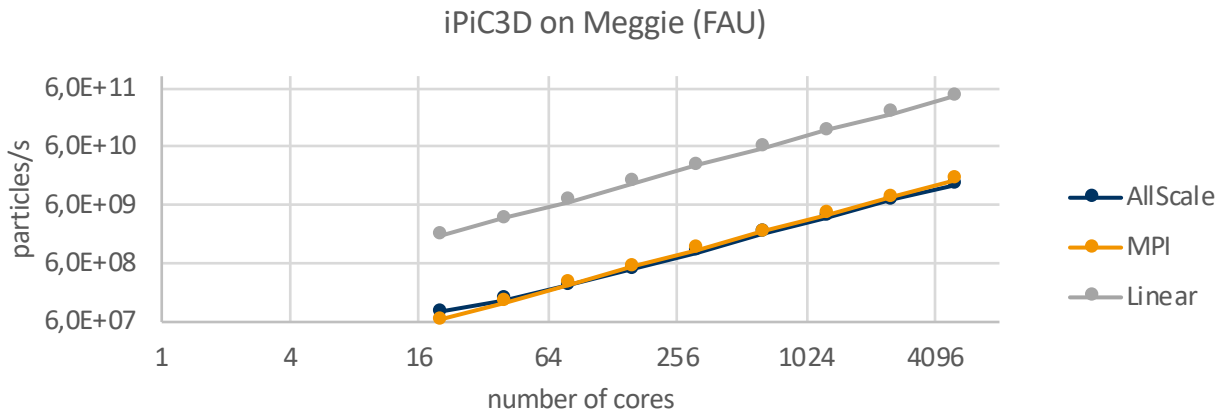
AMDADOS

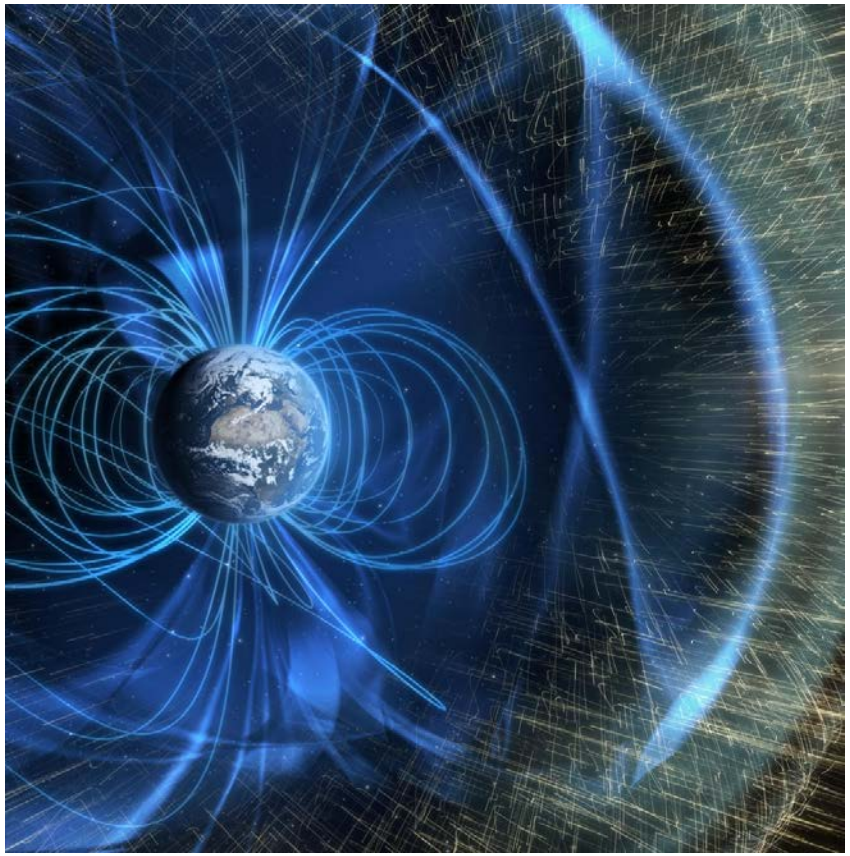
- Goal: simulate oil spill in real time using advection diffusion model with data assimilation
- Data assimilation introduces large load imbalance
  - Assimilation increases computational load by a factor in the order of 10^2
- Stencil operator + adaptive mesh data item
  - Spatio-temporal parallelization
  - Refinements at observations with 3 levels of refinement



iPiC3D

- FetHPC H2020 project AllScale (coordinator: Dept. of Computer Science, UIBK)
- Space weather prediction, developed with KTH Stockholm
- Particle-in-Cell simulation
- Productivity improvement compared to MPI reference implementation





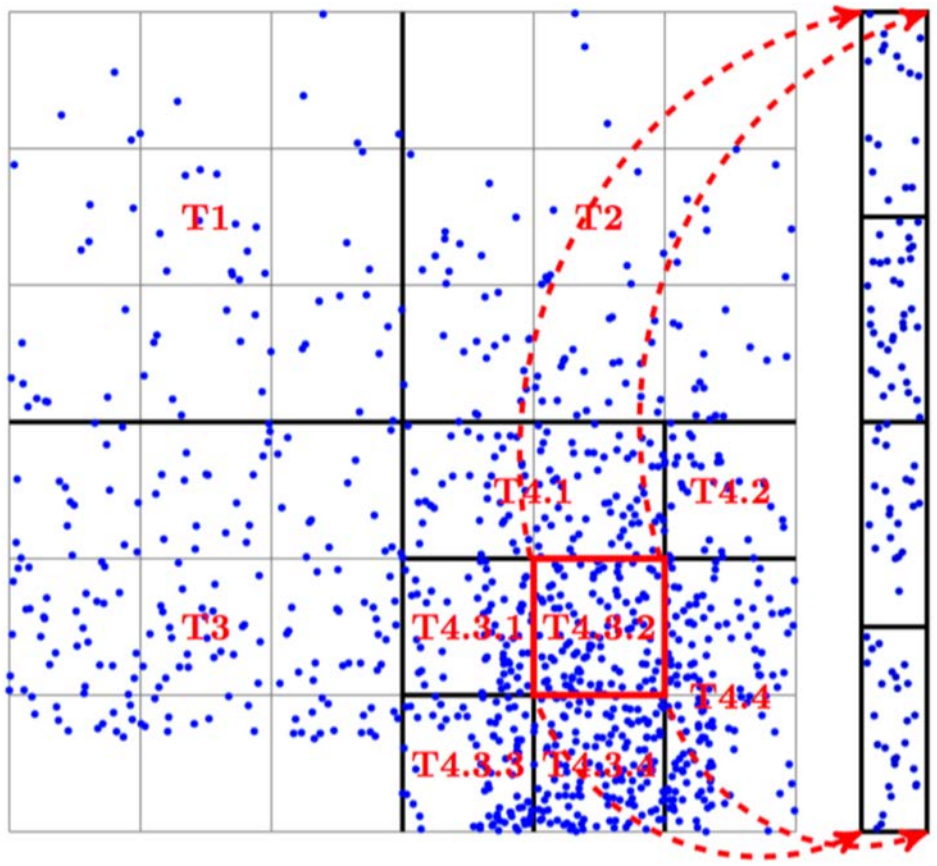
Source: https://twitter.com/maven2mars/status/984440044659159040

iPiC3D

- Goal: Collision-less plasma simulation

$$\frac{\partial f_s}{\partial \boldsymbol{t}} + \boldsymbol{v} \cdot \frac{\partial f_s}{\partial \boldsymbol{x}} + \frac{q_s}{m_s}\left(\boldsymbol{E} + \frac{\boldsymbol{v} \times \boldsymbol{B}}{c}\right) \cdot \frac{\partial f_s}{\partial \boldsymbol{v}} = 0$$

- Compute equation of motion for charged particles and solve Maxwell's equations in turns
- Highly dynamic load imbalance due to varying particle densities
- Nested pfor operators + multiple grids

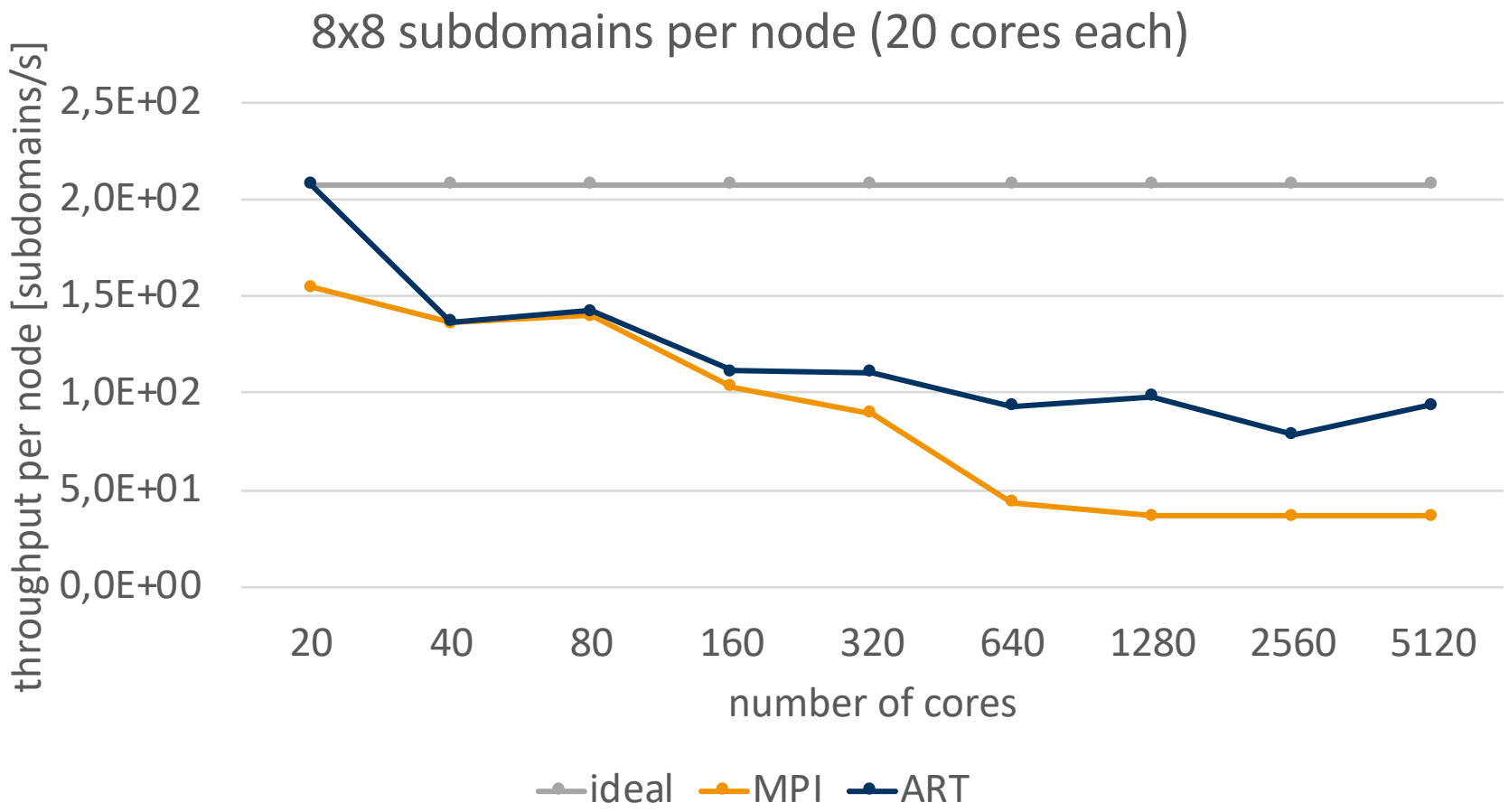


Experimental Target Systems

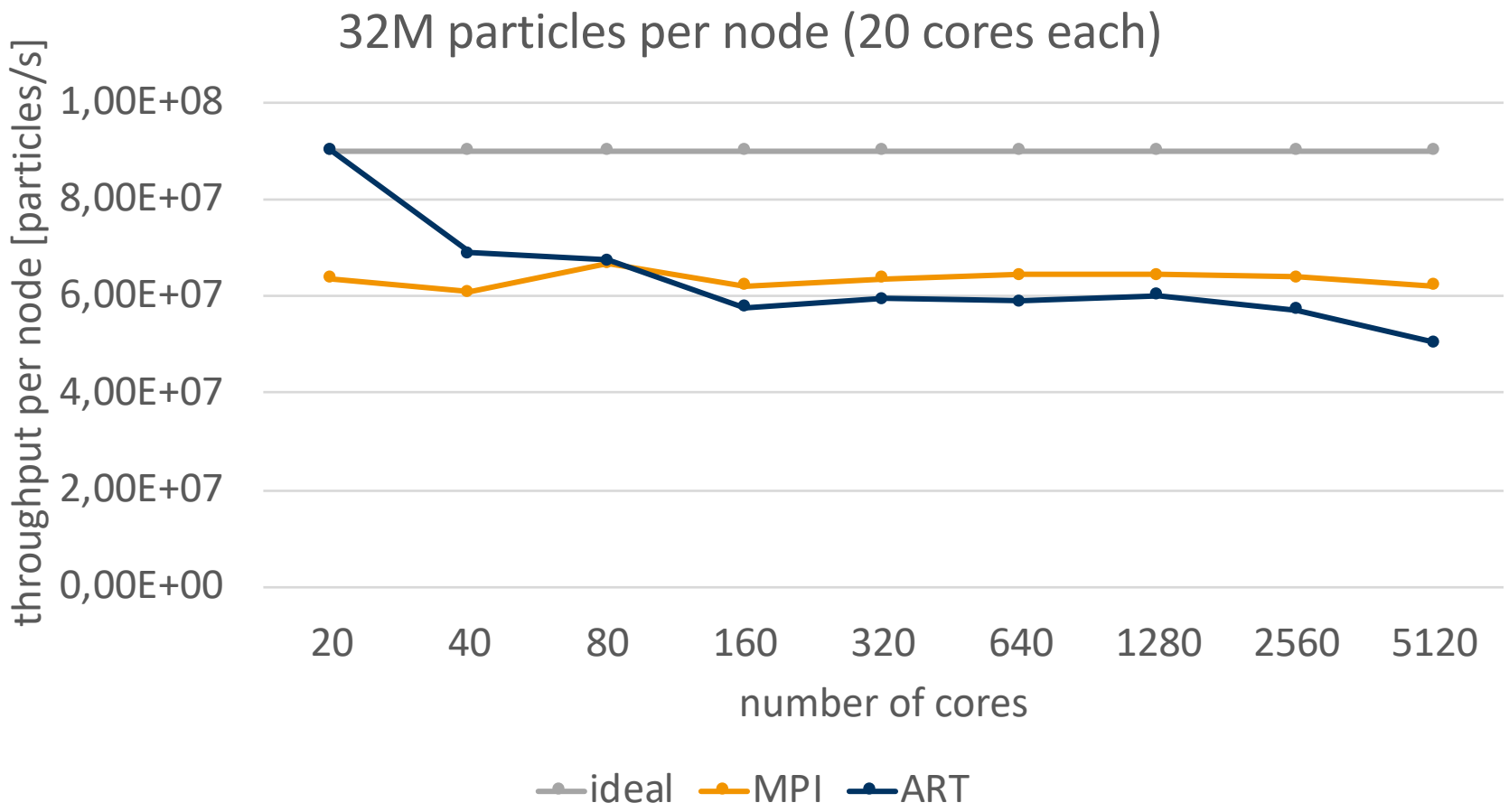
- Meggie (FAU)
  - 728 nodes
  - 2x Intel Xeon E5-2630 v4 (10 cores) each
  - 14,560 cores in total
  - Intel OmniPath

- VSC-3 (UIBK)
  - 2020 nodes
  - 2x Intel Xeon E5-2650 v2 (8 cores) each
  - 32,320 cores in total
  - QDR-80 dual-link InfiniBand

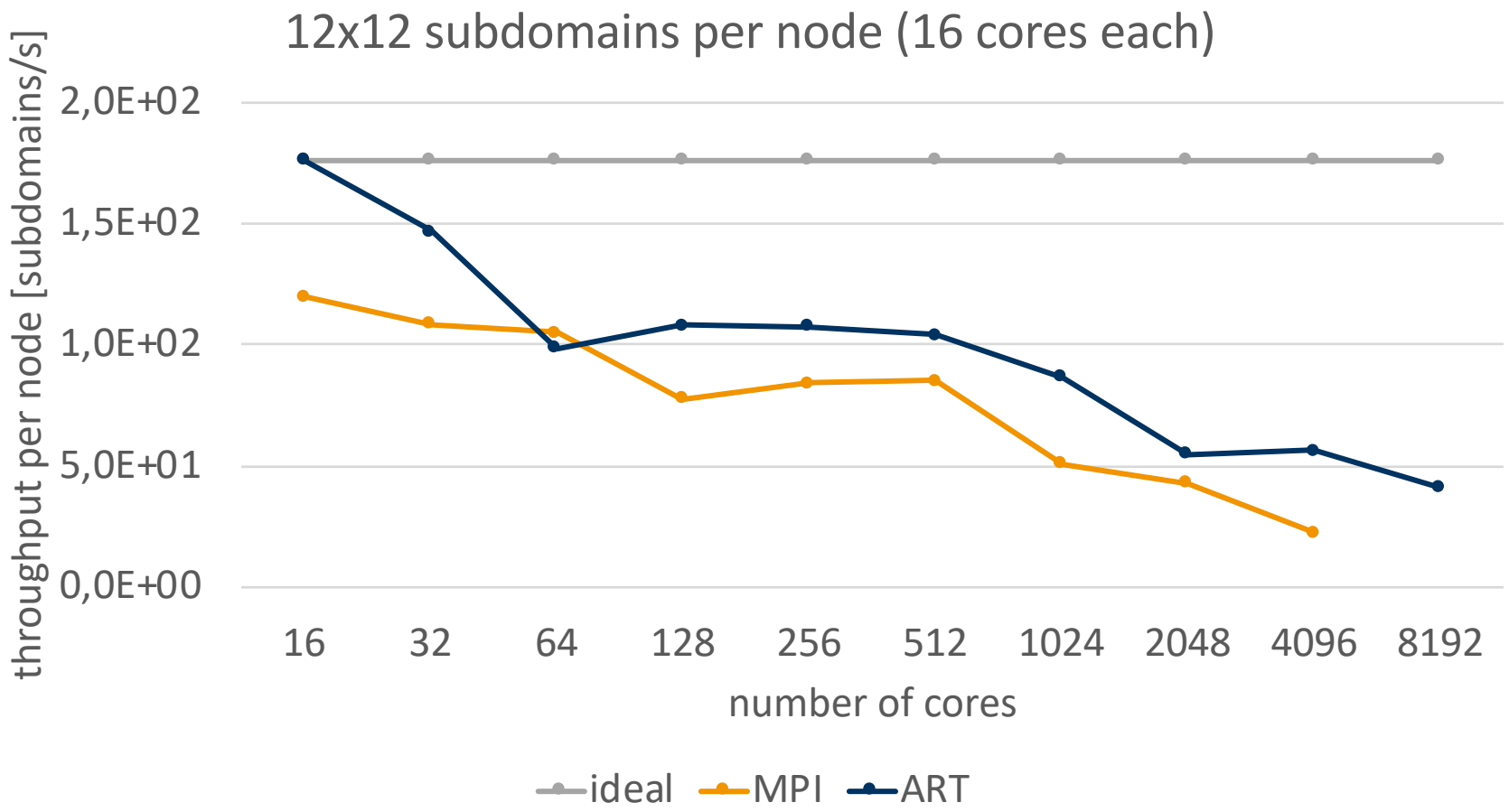
AMDADOS on Meggie



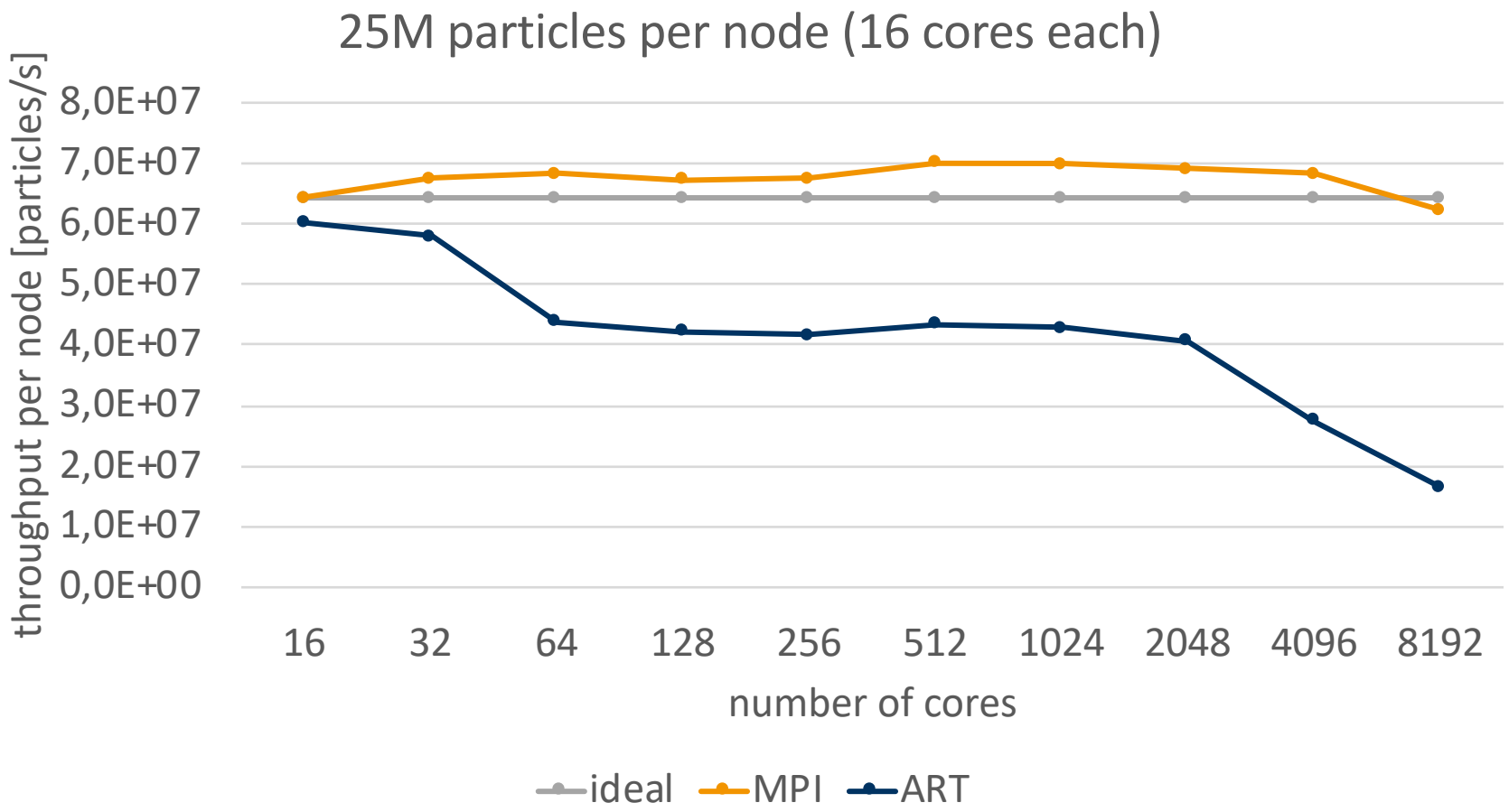
iPiC3D on Meggie



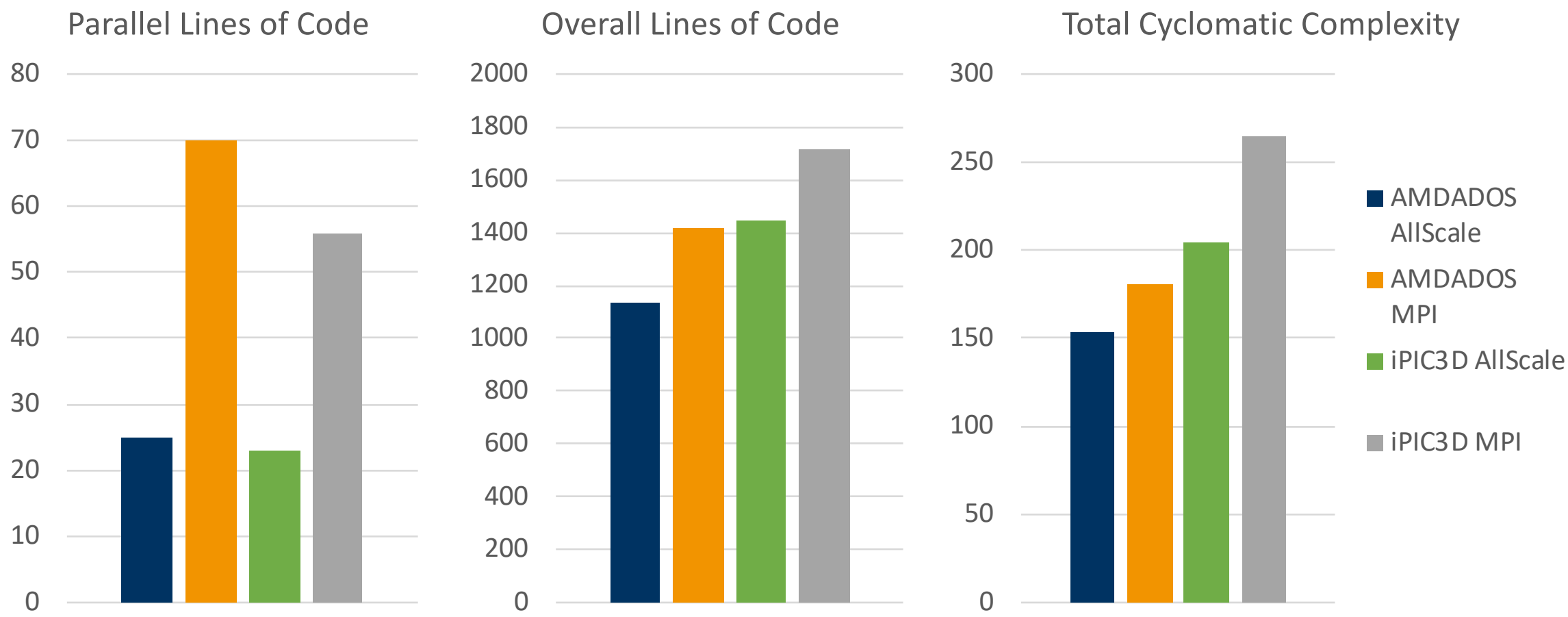
AMDADOS on VSC-3



iPiC3D on VSC-3



Productivity Evaluation



Practical Experience: Benefits and Challenges

- Data management out of the box
- No explicit communication and data management
- Automated object serialization
- Implicit intra- and inter-node load balancing
- No need to deal with MPI+X
- Preserve structure of mathematical equations
- **Separation of concerns**

- Requires advanced C++ knowledge
- Longer compile times (but once for multiple runs)
- No support for external libraries that employ non-AllScale parallelism

Thanks

- Visit our AllScale website

  http://www.allscale.eu

- Visit our AllScale repositories

  https://github.com/allscale

- Visit our Research Center HPC website

  www.uibk.ac.at/fz-hpc

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671603