universität innsbruck

FZ HPC

On using modern C++ and nested recursive task parallelism for HPC applications with AllScale

Thomas Fahringer
University of Innsbruck, Institute of Computer Science, Austria

„APART 25th Anniversary Workshop, Obergurgl, Feb. 13, 2024

Funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671603, by the European HPC JU under grant agreement No 956137, by the Austrian Research Promotion Agency (FFG) under Grant Agreements 879201 (LIGATE).

From Performance Prediction to Compilers and Distributed Computing

- Superb and Vienna Fortran Compiler [SFB Aurora; Apart Working Group]
  - Parameter based Performance Prediction Tool (P^3T) for HPF
  - Scalea (instrumentation, monitoring, performance analysis) by Hong-Linh Truong (now Aalto University) for Vienna Fortran
  - Zenturio (automatic experiment generation for large scale analysis) by Radu Prodan (Klagenfurt University) for Vienna Fortran

Fortran

- Askalon (application development and runtime for the Grid and the Cloud) [8 EU projects]
  - performance predicton for scientific workflows (Farrukh Nadeem, King Abdulaziz University)
  - performance prediction for services and tasks for clouds (Thanh-Phuong Pham, Helsinky University)
  - scheduling and resource management

C, Java, Python

more Compilers and runtime system

- Insieme C/C++ compiler (Allscale EU project)
  - Cilk, OpenMP
  - auto-tuning for multiple parameters
  - runtime optimization
  - performance prediction (distance to finish task)

C/C++

- Allscale API, compiler and runtime system (Allscale EU project)
  - high-level C++ domain specific language
  - productivity (hide MPI)
  - sophisticated compiler analysis
  - recursive task parallelism

C++

- Celerity API and runtime system for GPU clusters based on SYCL (FETHPC LIGATE)
  - Productivity versus performance
  - API research to hide MPI
  - focus on asynchronous tasks
  - runtime optimization and scheduling

SYCL

more Distributed Systems

- Apollo (application development and runtime for the edge-cloud continuum)
  - scientific workflows
  - programming paradigms, scheduling, resource management
  - semi-automatic serverless function generation

YAML. serverless functions C, C++, Java, Python, …

- Apache Flink
  - data pipelines and big data
  - real-time latency intensive stream/batch applications
  - targeting edge-cloud
  - ensemble scheduling and resource management

Java, Python, Scala

Performance models and analysis:

- simple analytical models and real-time measurements
- more complex models based on ML

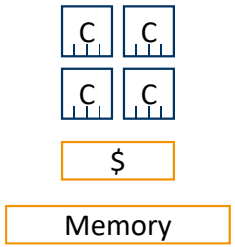
AllScale – A H2020 Success Story

- Horizon 2020 funded project and marked "Success Story"
- 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

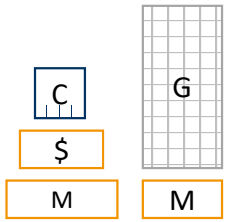
Parallel Architectures

Multicore



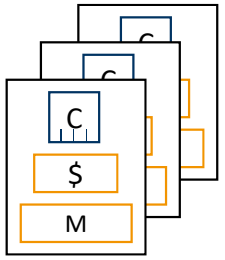
OpenMP/Cilk

Accelerators



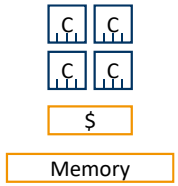
SYCL/OpenCL/CUDA

Clusters

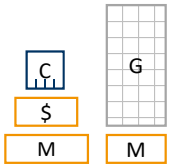


MPI/PGAS

Real World Architectures

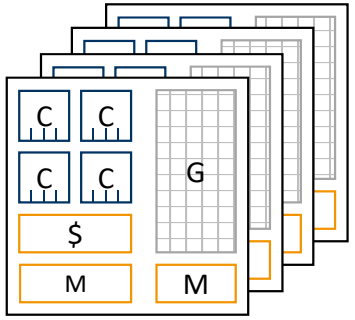


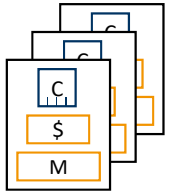
OpenMP/Cilk



SYCL/OpenCL/CUDA

≠

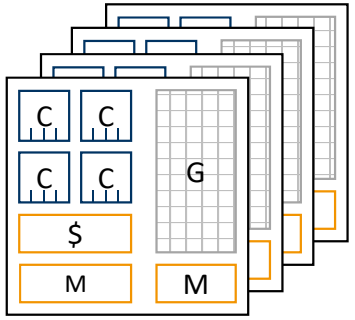




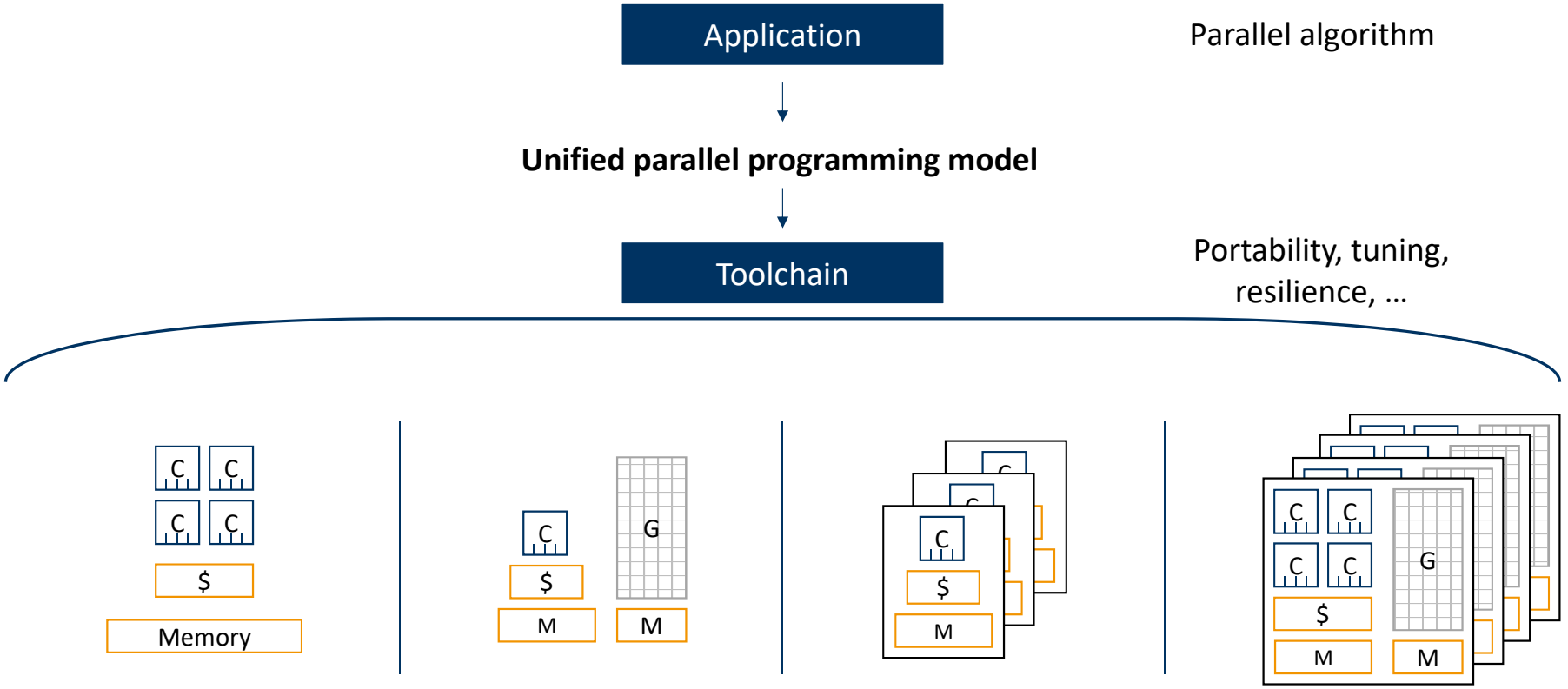
MPI/PGAS

Hybrid Programming Models for Heterogeneous Parallel Architectures

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

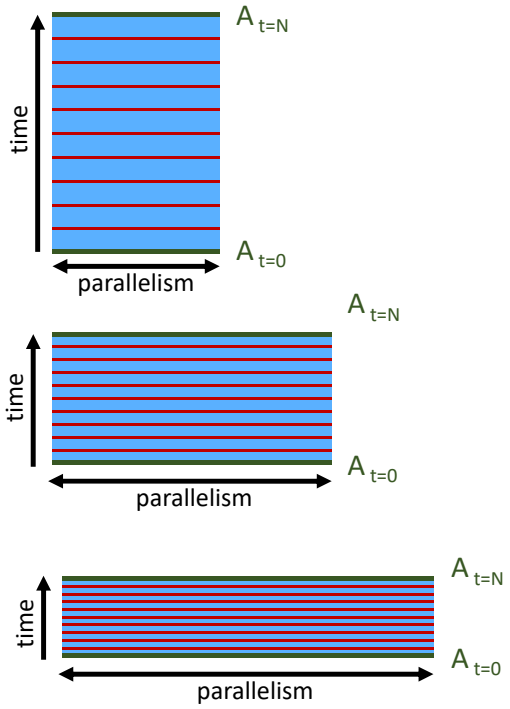


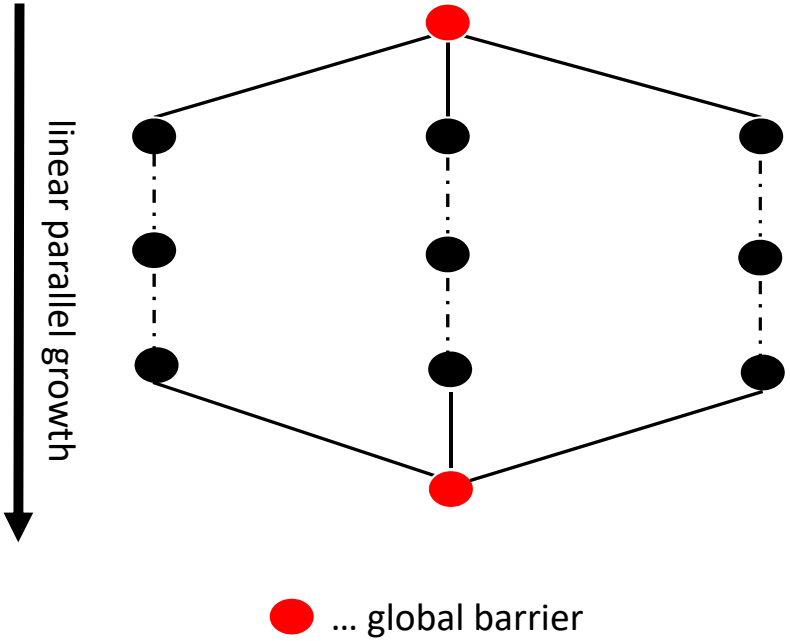
AllScale Vision



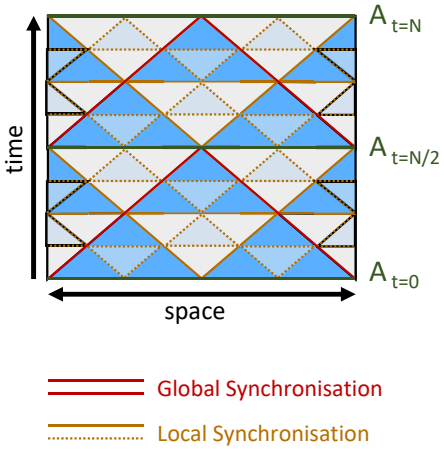
Conventional Flat Parallelism

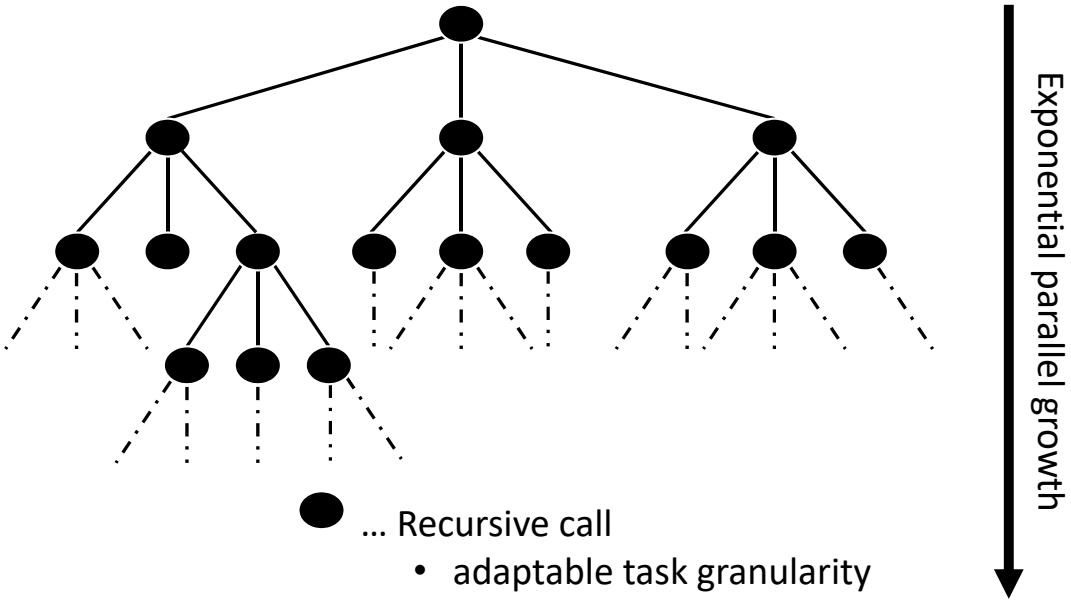
How to map flat parallelism to a hierarchical parallel architecture?
Complex handling of errors – global operations





Recursive Parallelism





… Recursive call

- adaptable task granularity
- multiversioning for tasks
- maps naturally to multiple levels of HW parallelism
- resilience

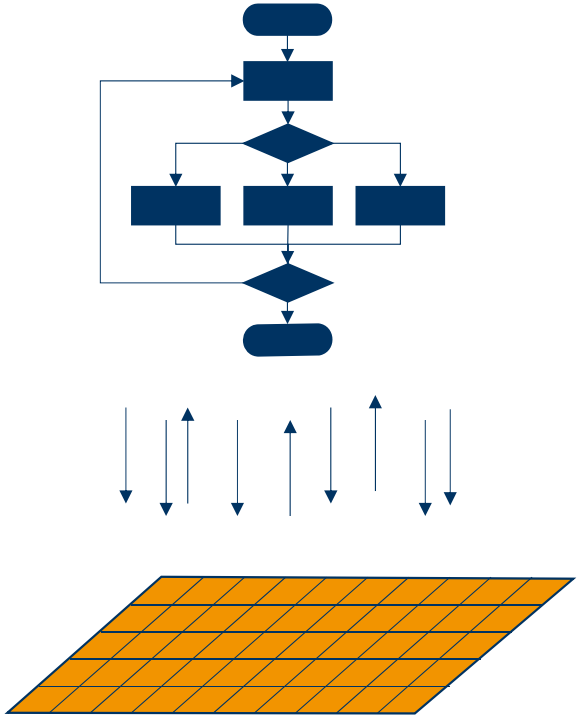
Recursive Task Parallelism Potential

- Branch and bound algorithms
- Place and Route algorithms
- SAT or SMT Solvers, Datalog Engines or similar reasoners
- Combinatorial optimization problems
- Stencil codes, in any number of dimensions
- Dense algebra operations
- N-body simulation
- Rendering

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

Serial Work and Data

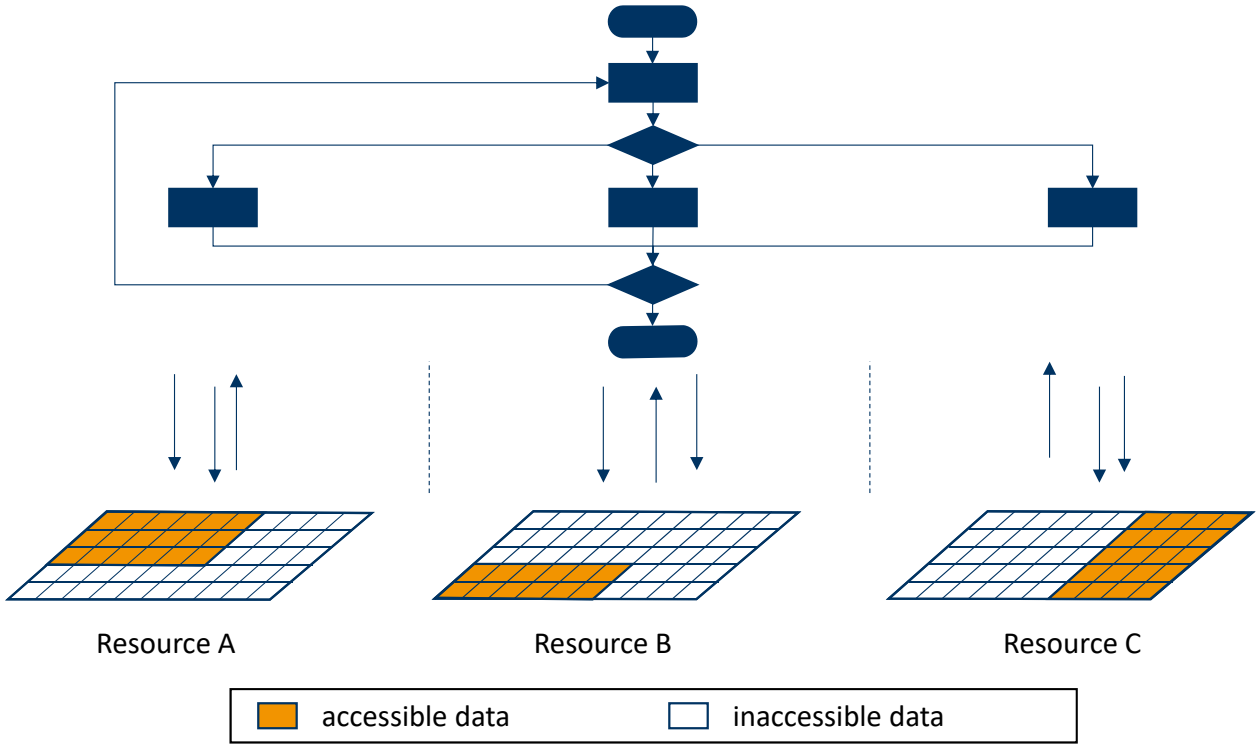
Algorithms
manipulate the
state of
data structures



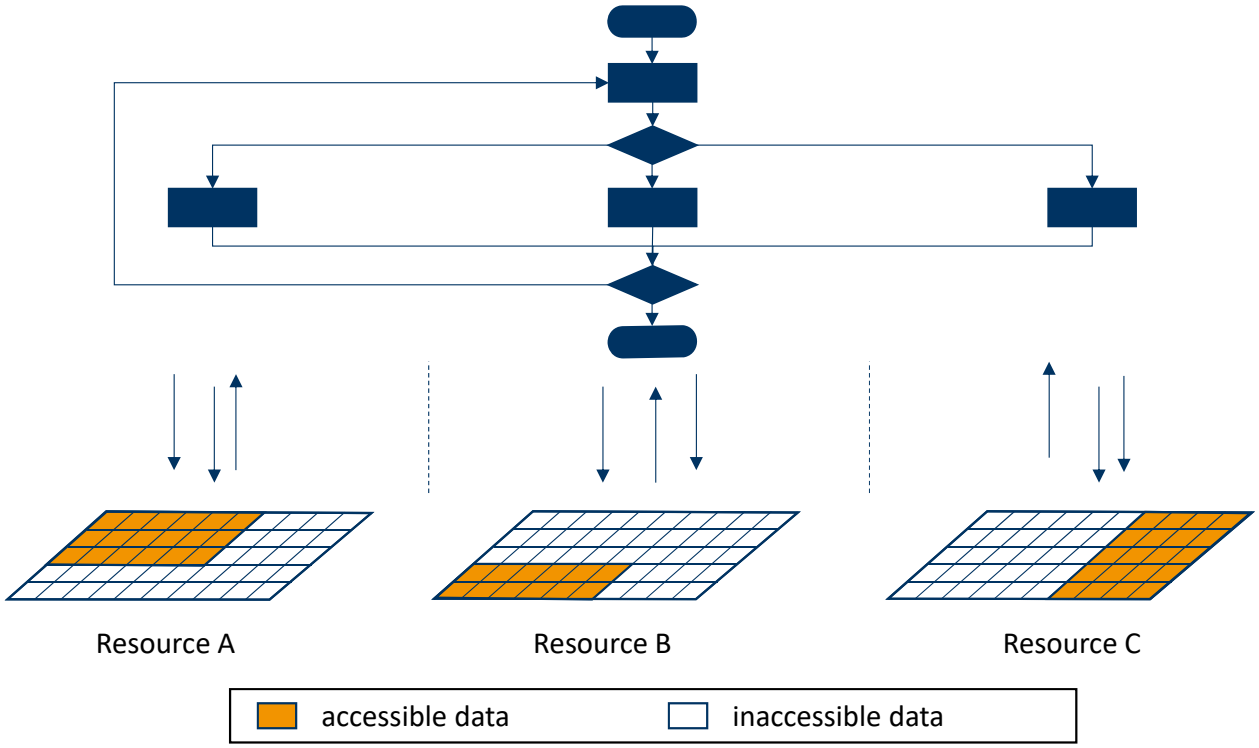
explicitly expressed
in program code

the **implicit** effect
of program code

Parallel Work and Data



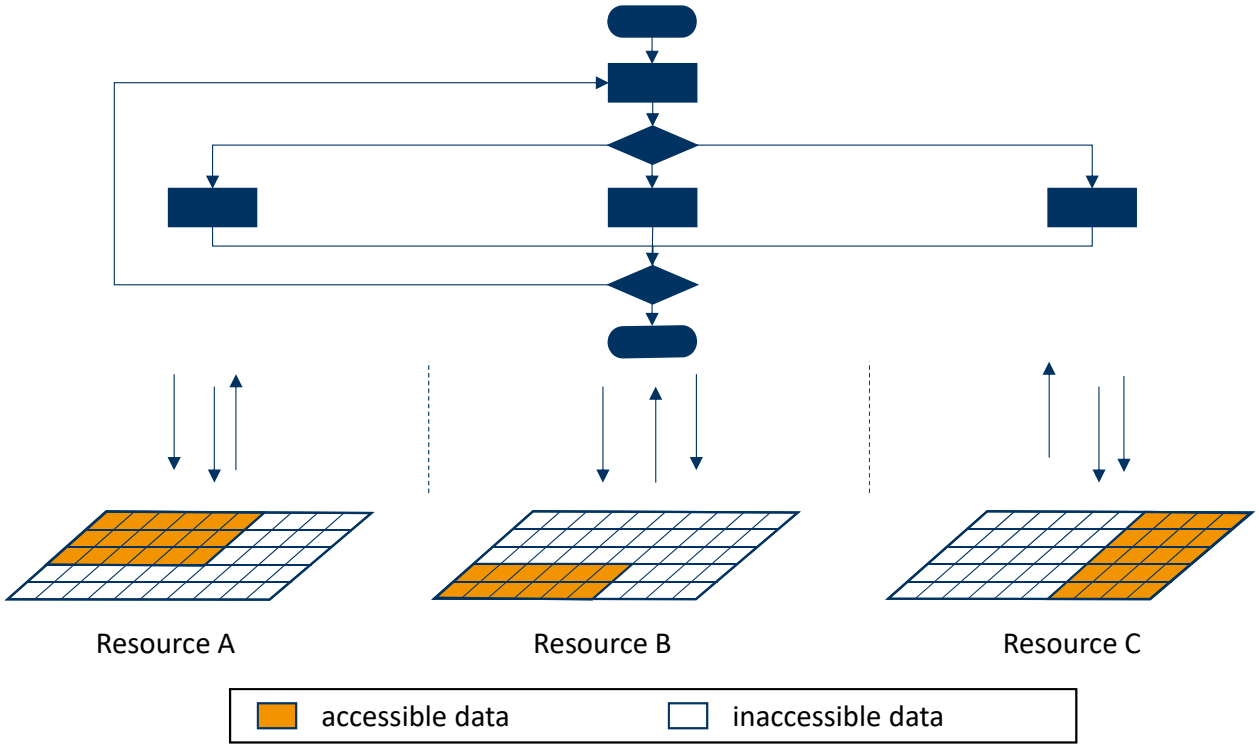
Parallel Work and Data



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

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

Parallel Work and Data

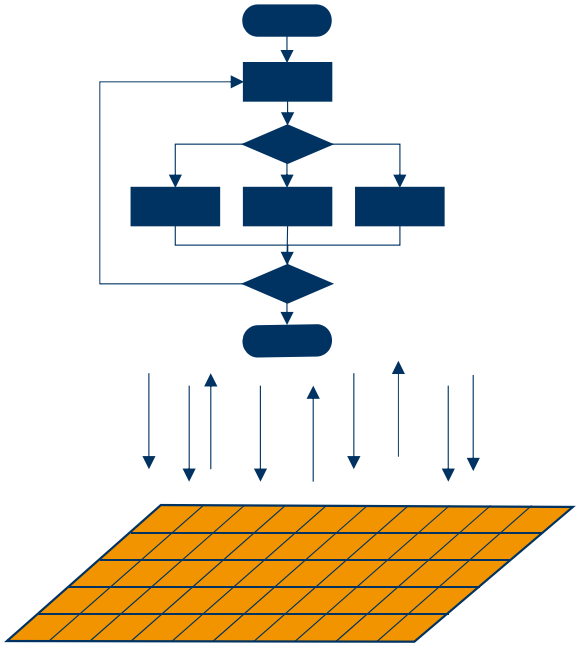


Desired:
advanced system features and services including

- 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 User DSL/API Design



application

Operators:

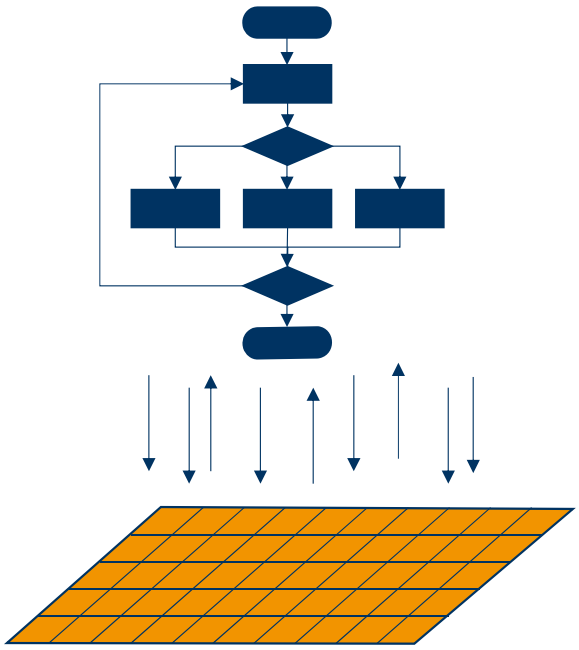
- parallel loops
- parallel reduction
- map-reduce
- stencils
- V-cycles
- brand and bound, linear algebra, …

Data Structures:

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

parallel constructs

AllScale's Approach



application

Operators:

- parallel loops (pfor)
- parallel reduction
- map-reduce
- stencils
- V-cycles
- branch and bound, …

higher order parallel primitive

Data Structures:

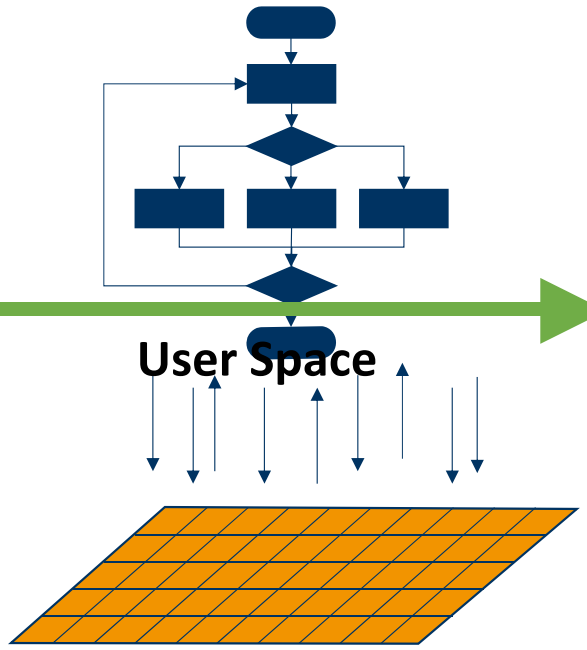
- multi-dimensional grids
- adaptive grids
- maps
- trees
- hierarchical meshes
- …

generalized abstract data structure

parallel constructs

primitives for parallel construct

AllScale's Approach



User Space

application

Operators:

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

higher order parallel primitive

Toolchain

Data Structures:

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

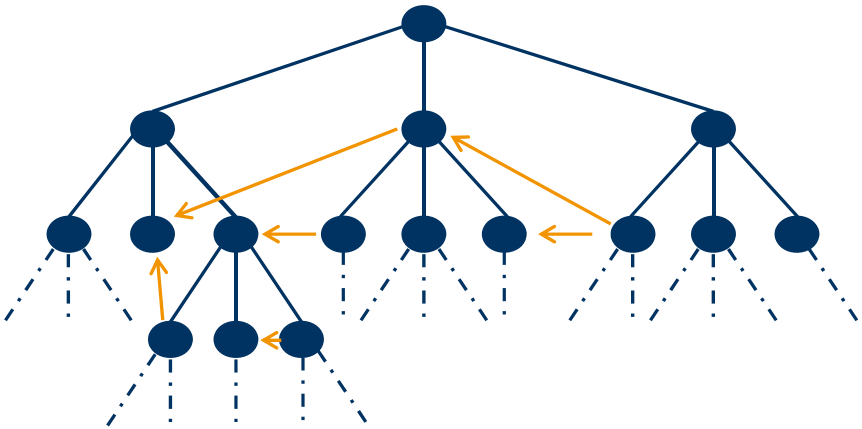
generalized abstract data structure

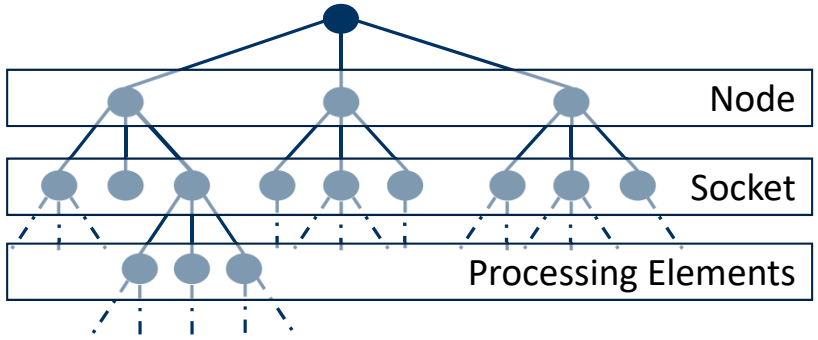
parallel constructs

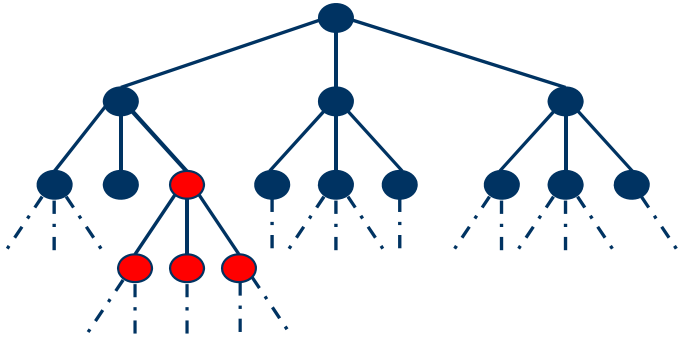
primitives for parallel construct

Work Flow Construct (Implicit Task Parallelism): AllScale Core API

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

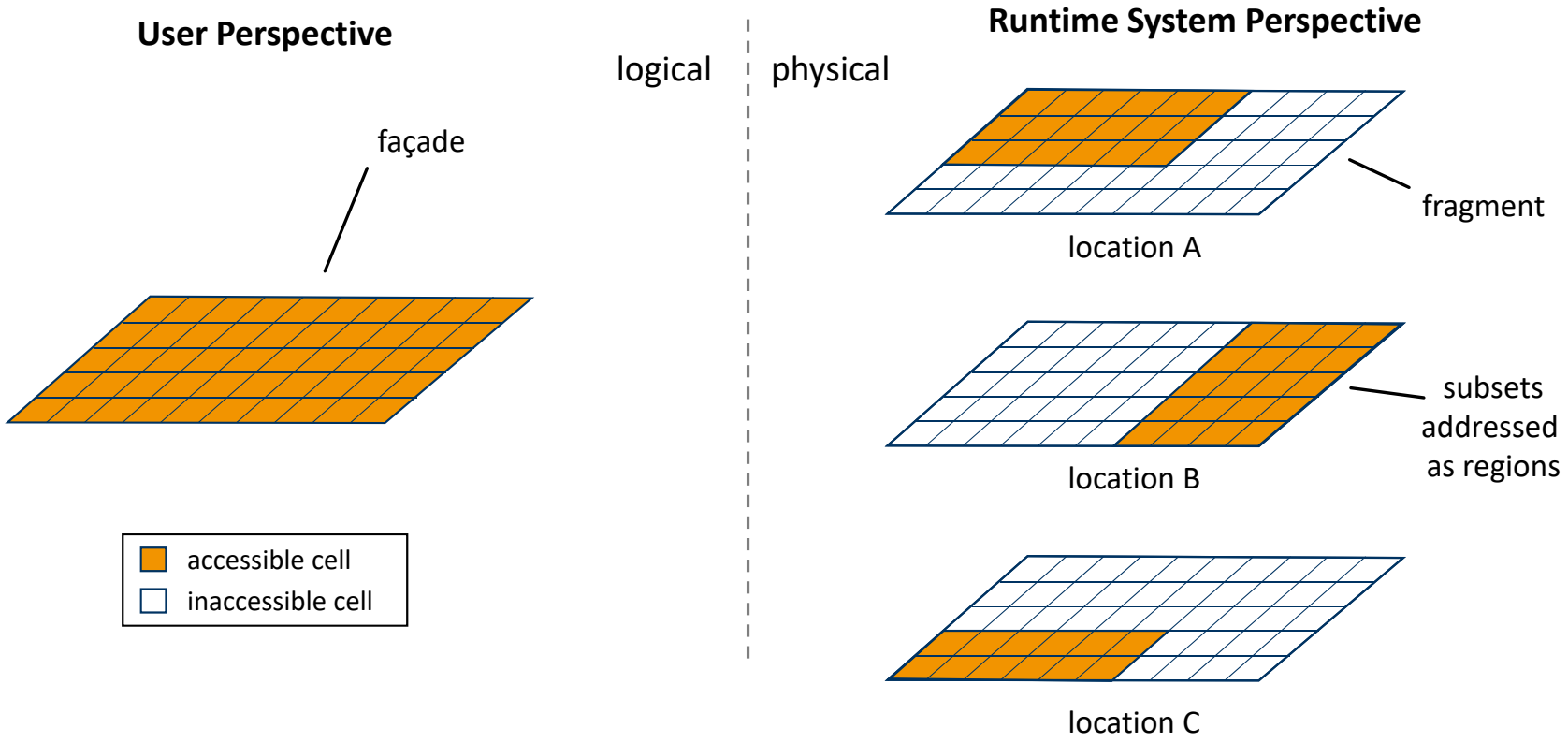




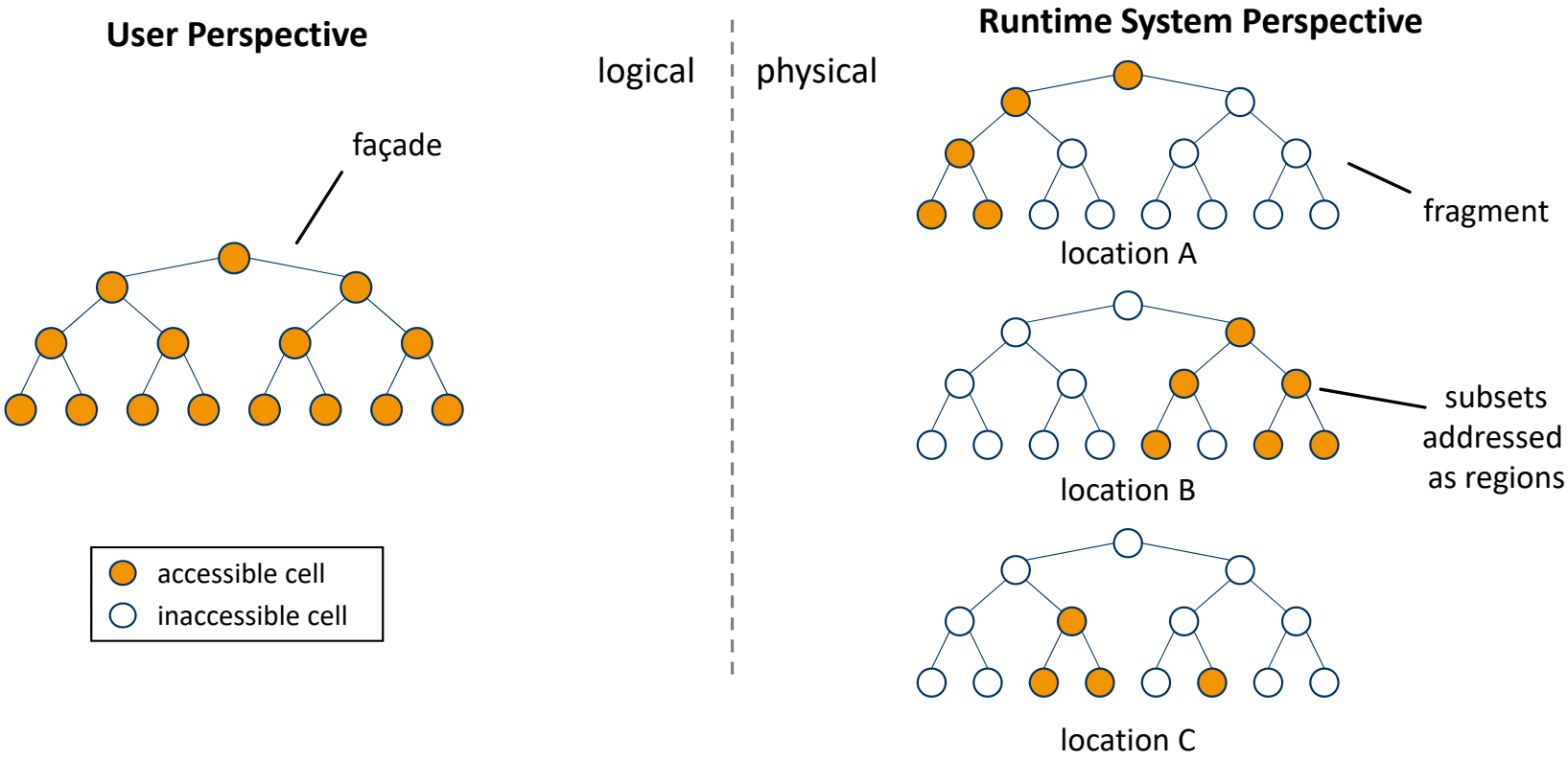


… recursive task — … task hierarchy ← … fine grained dependencies … failed task

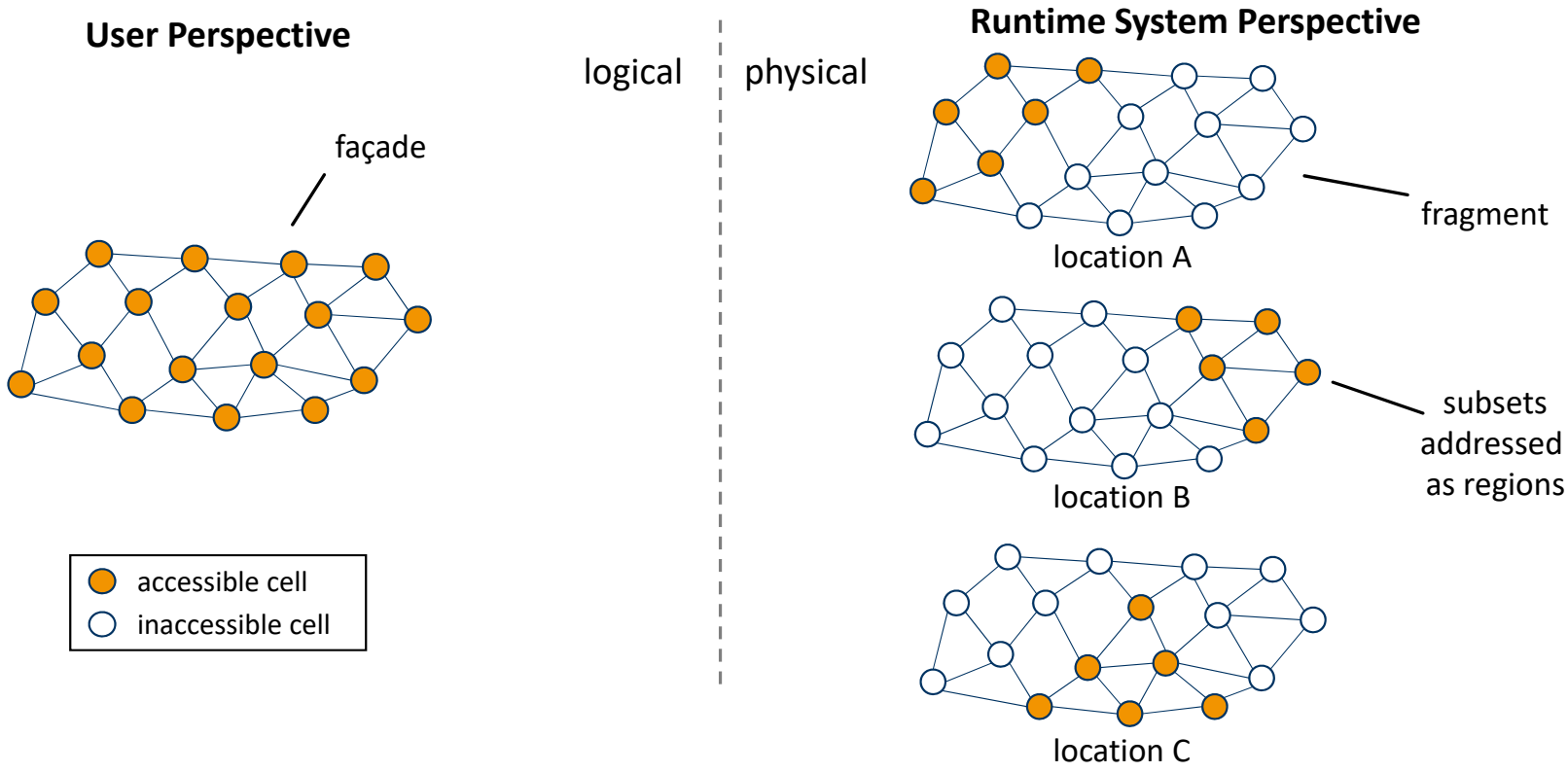
Data Items


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

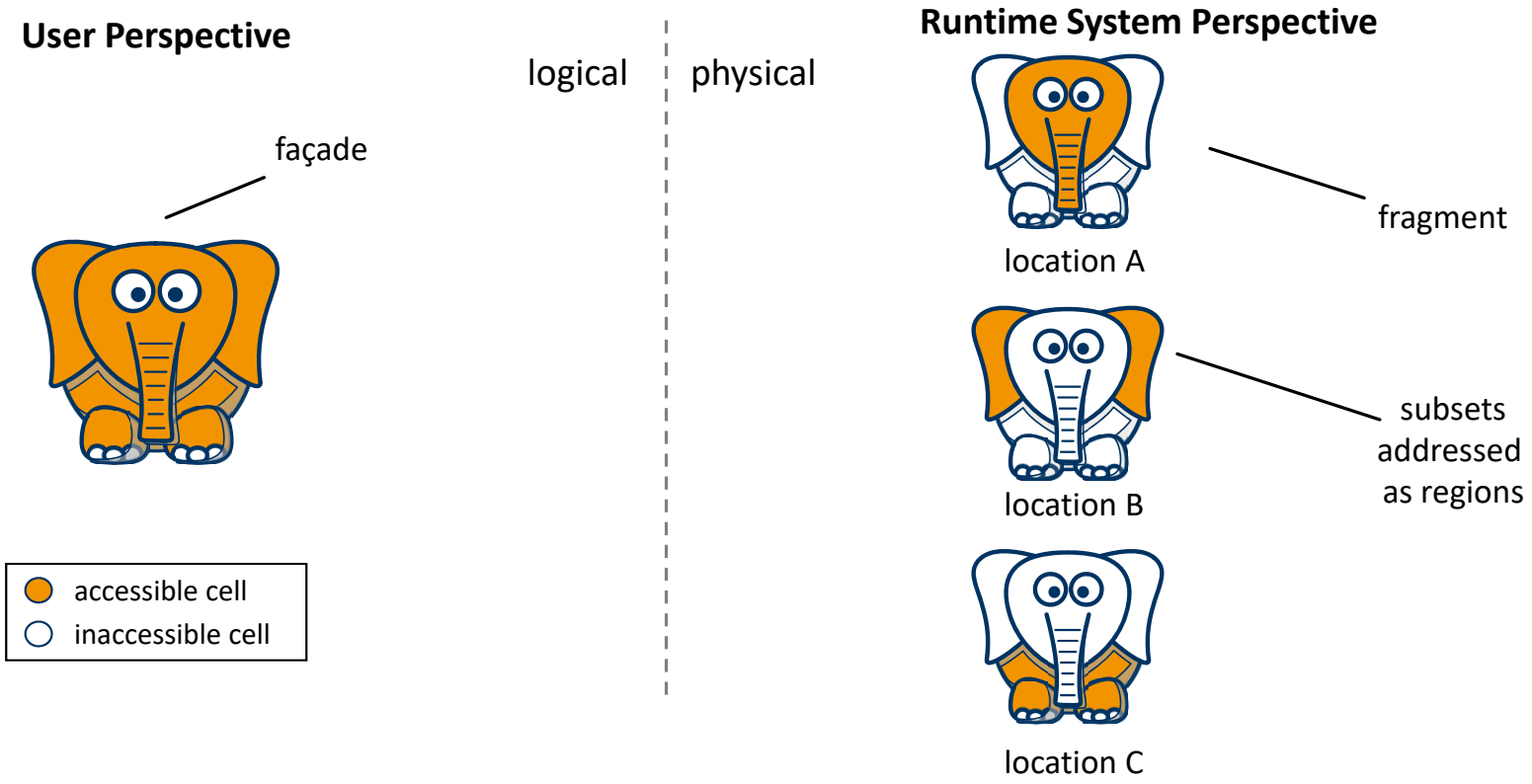
Data Items


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

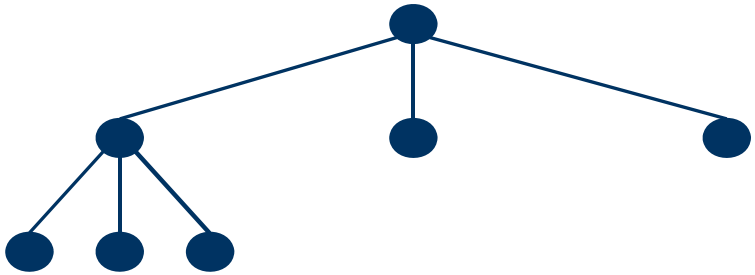
Data Items


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

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

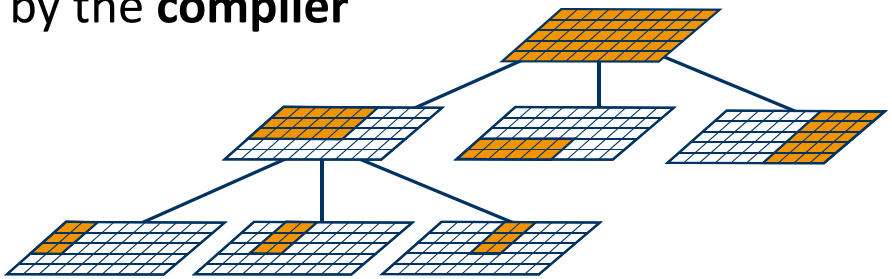
Work and Data Link



recursive work load
decomposition defined
by **algorithm**

automatically deduced
by the **compiler**

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



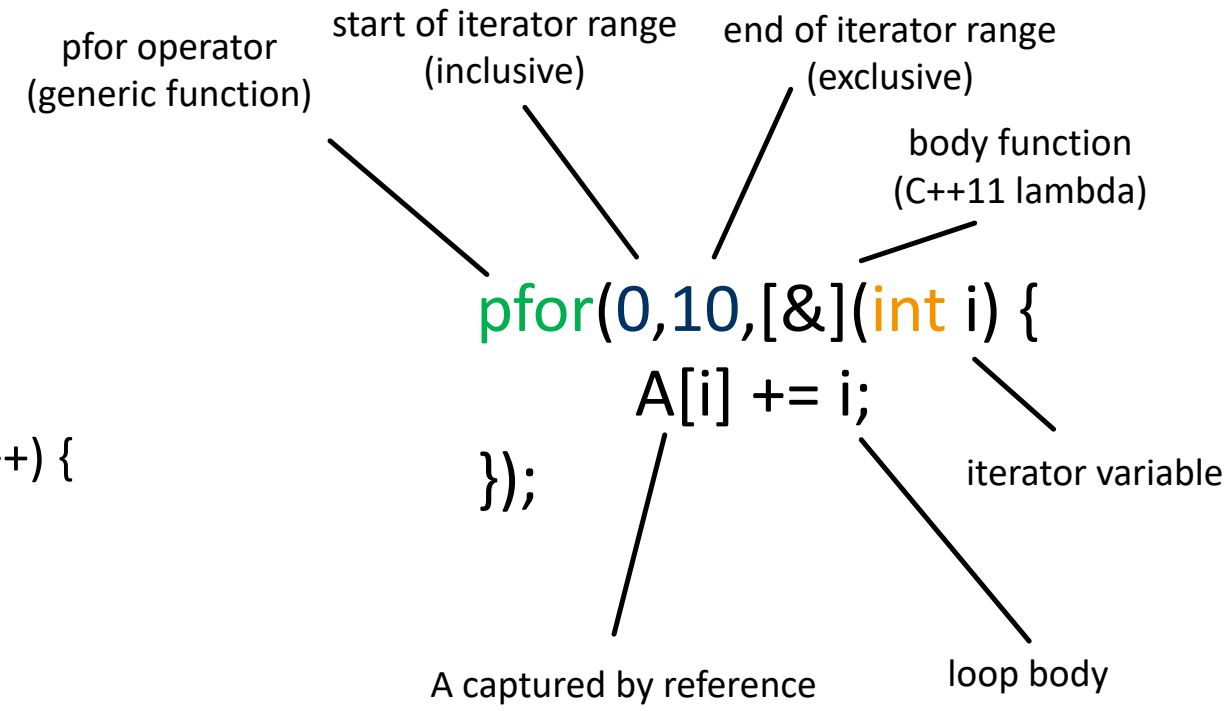
A Simple Loop

```
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.

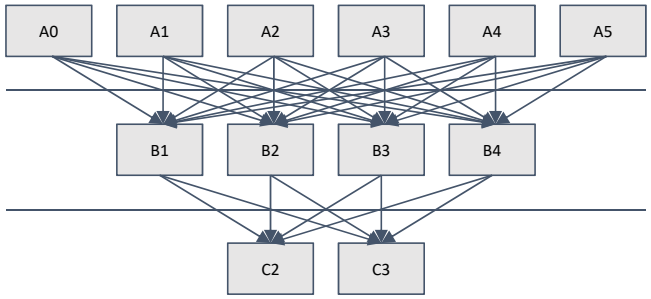
Pfor in Allscale

- Use it just like regular parallel loops:

```
pfor(0…6) a[i] = f(...);

pfor(1…5) b[i] = f(a[i-1],a[i],a[i+1])

pfor(2…4) c[i] = f(b[i-1],b[i],b[i+1])
```

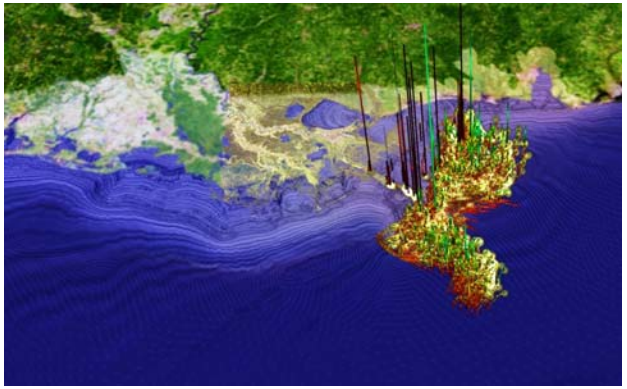


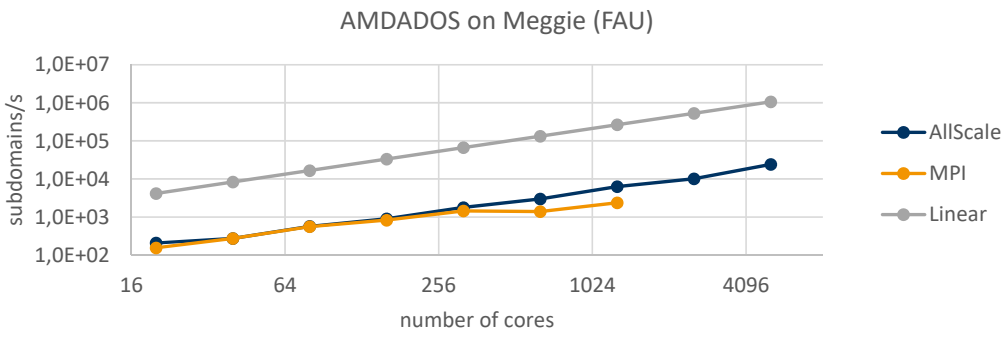
- Each of these spawns a decomposable recursive task processing the given range

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

AMDADOS

- FetHPC H2020 project AllScale
- Oil spill simulation, developed with IBM Ireland
- Stencil and kalman filter for assimilating sensor data
- AllScale exceeds performance of MPI reference implementation

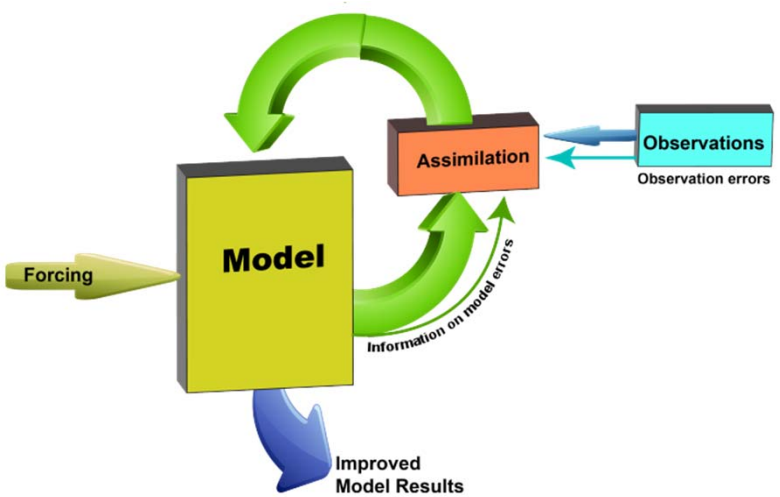




Sources: https://www.chemistryworld.com/features/oil-spill-cleanup/3008990.article, Marcel Ritter (UIBK)

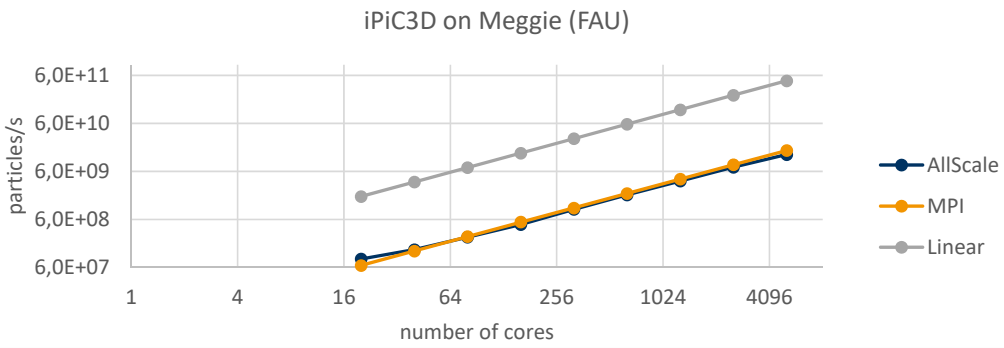
AMDADOS

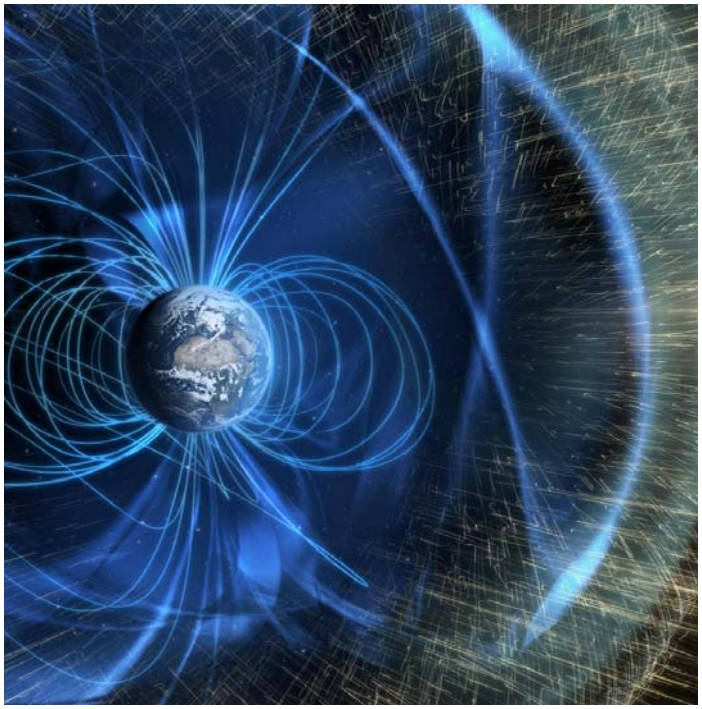
- Goal: simulate oil spill in real time using advection diffusion model with data assimilation
- Data assimilation introduces large load imbalance
  - Assimilation increases computational load by a factor in the order of 10^2
- Stencil operator + adaptive mesh data item
  - Spatio-temporal parallelization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



iPiC3D

- FetHPC H2020 project AllScale
- Space weather prediction, developed with KTH Stockholm
- Particle-in-Cell simulation
- Productivity improvement compared to MPI reference implementation





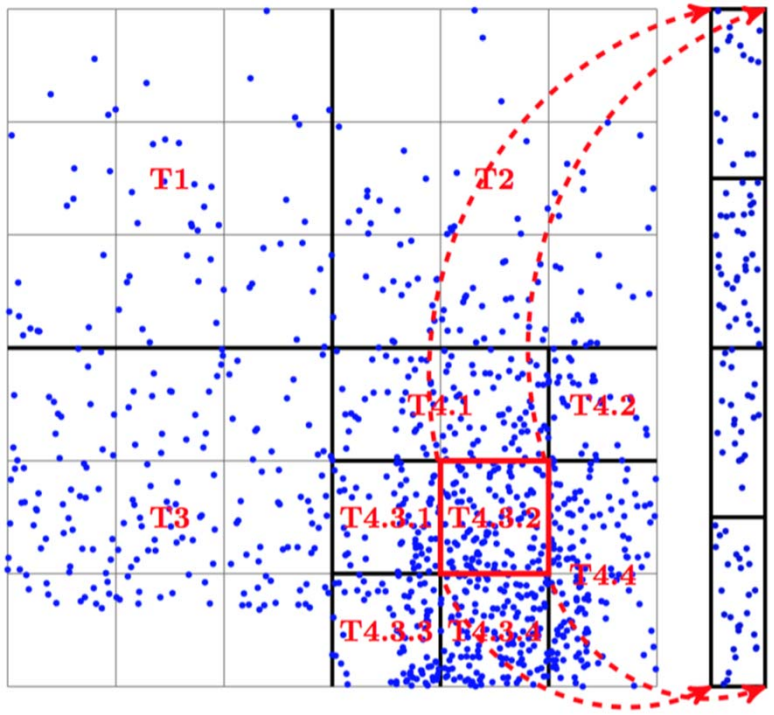
Source:
https://twitter.com/maven2mars/status/984440044659159040

iPiC3D

- Goal: Collision-less plasma simulation

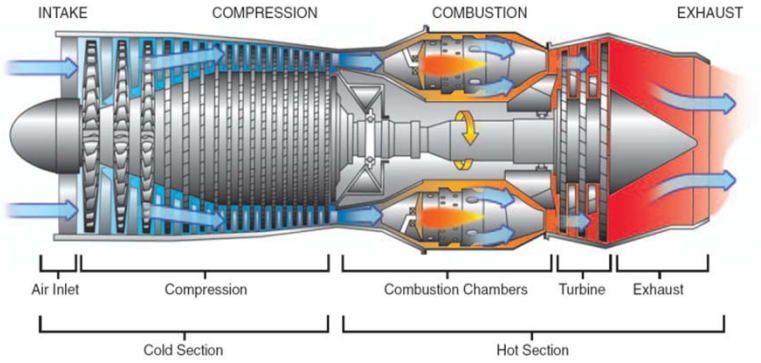
$$\frac{\partial f_s}{\partial \boldsymbol{t}} + \boldsymbol{v} \cdot \frac{\partial f_s}{\partial \boldsymbol{x}} + \frac{q_s}{m_s}\left(\boldsymbol{E} + \frac{\boldsymbol{v} \times \boldsymbol{B}}{c}\right) \cdot \frac{\partial f_s}{\partial \boldsymbol{v}} = 0$$

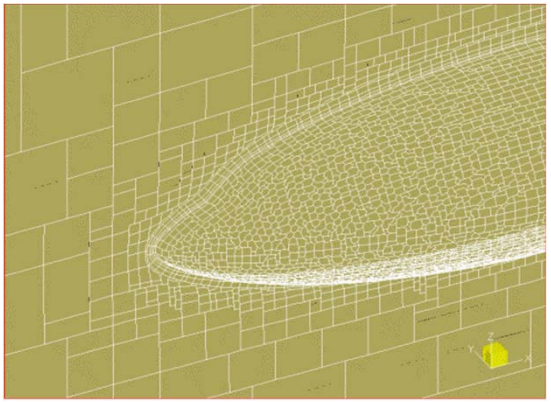
- Compute equation of motion for charged particles and solve Maxwell's equations in turns
- Highly dynamic load imbalance due to varying particle densities
- Nested pfor operators + multiple grids



FINE/Open

- Goal: perform scalable CFD simulations for various use cases (e.g. aircraft)
- Complex configurations with several hundreds of million of points.
  - Irregular data structure
  - Many elements: cells, faces, nodes, ...
  - Even more properties: velocity, pressure, …
- V-cycle and pfor operators + unstructured multi-level mesh

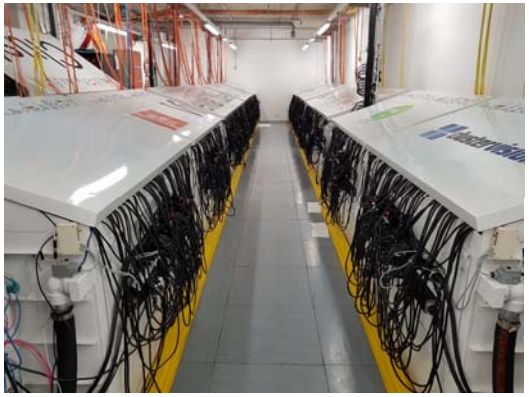




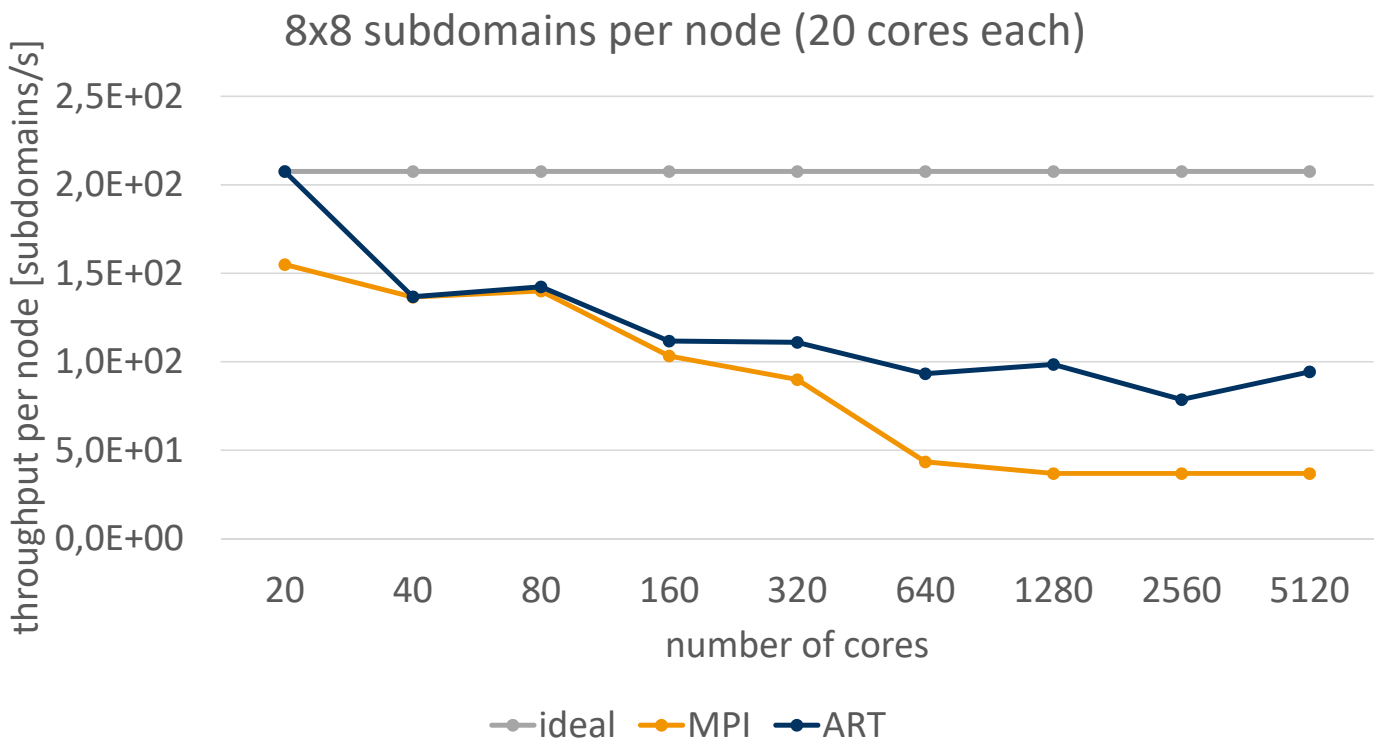
Experimental Target Systems

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

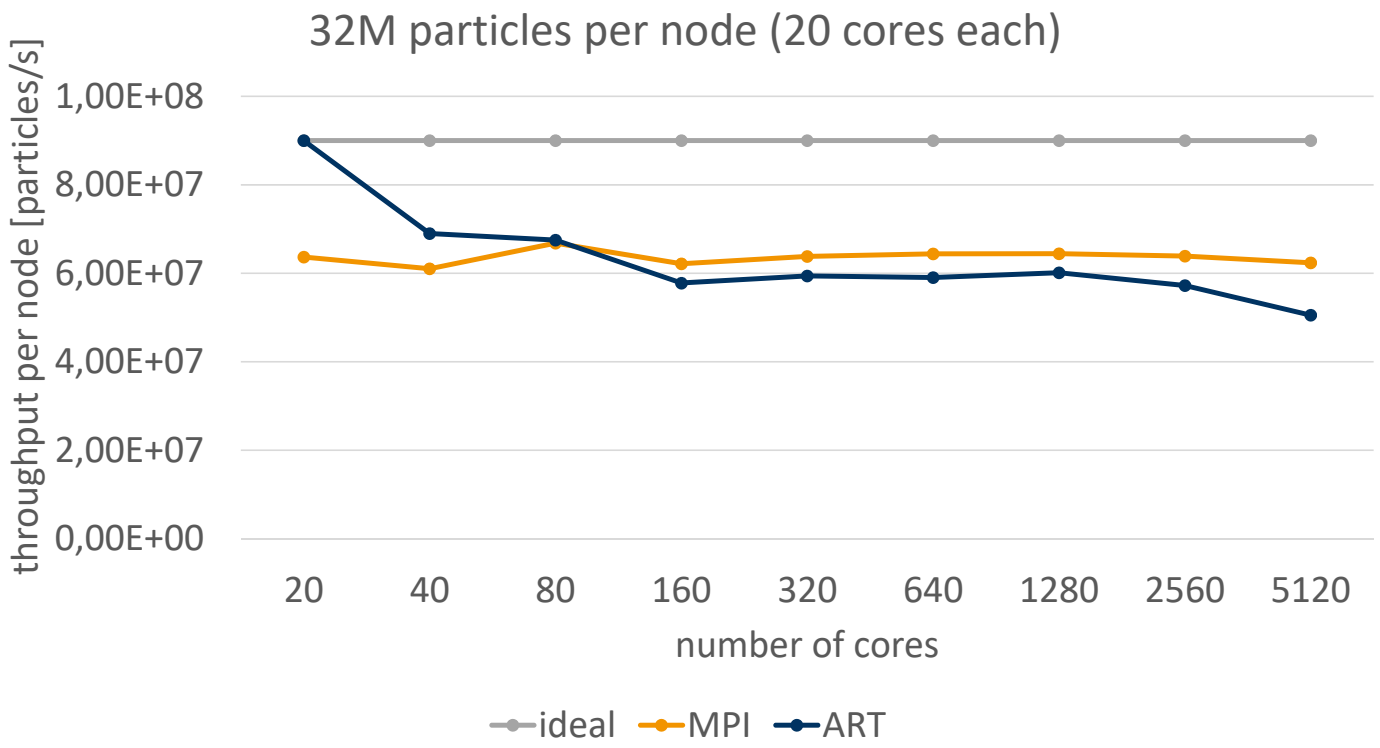
- VSC-3 (UIBK)
  - 2020 nodes
  - 2x Intel Xeon E5-2650 v2 (8 cores) each
  - 32,320 cores in total
  - QDR-80 dual-link InfiniBand



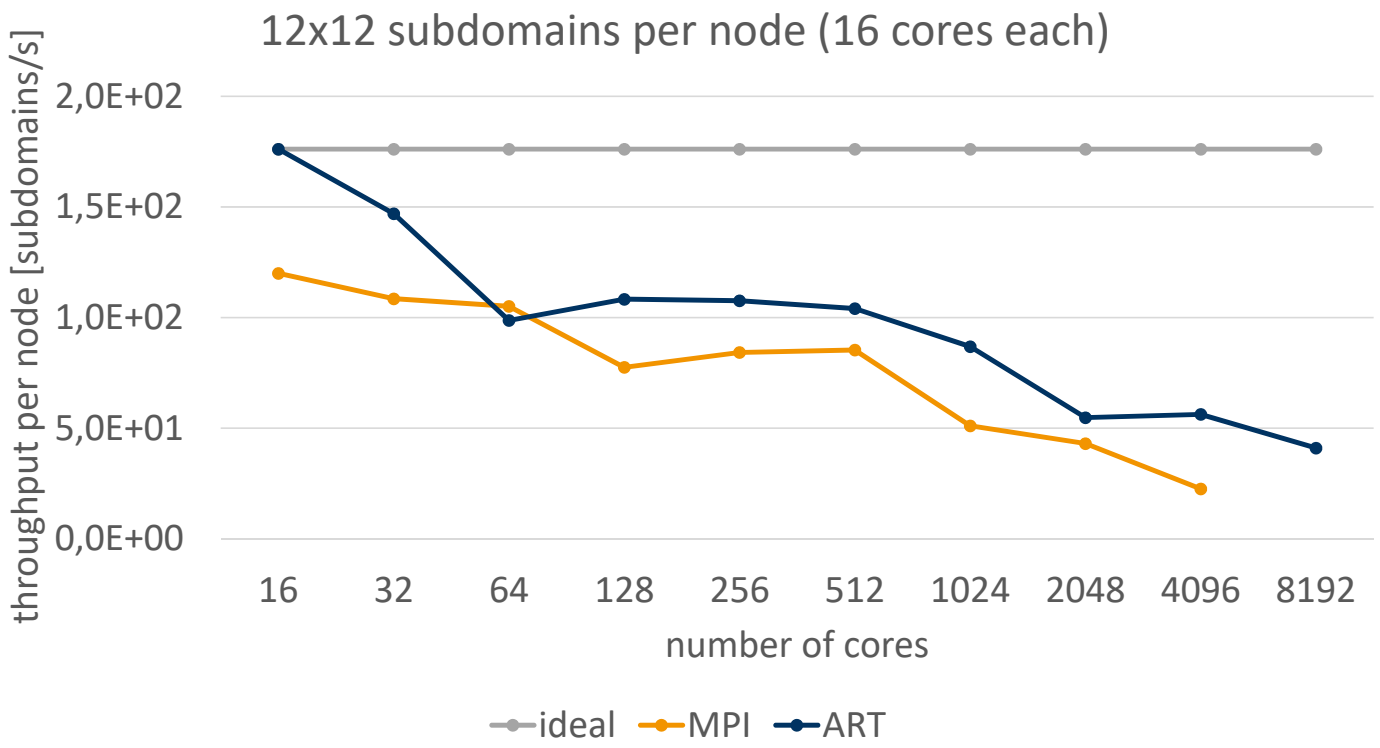
AMDADOS on Meggie



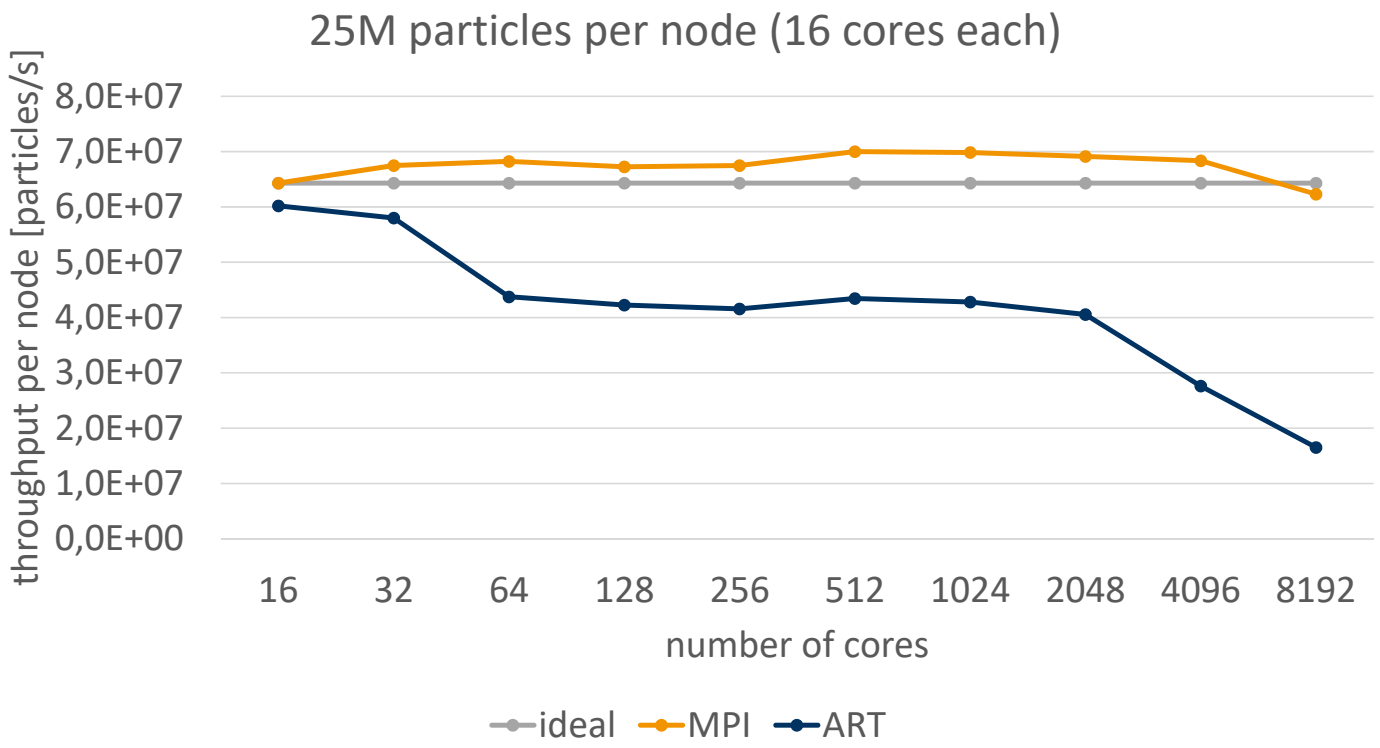
iPiC3D on Meggie



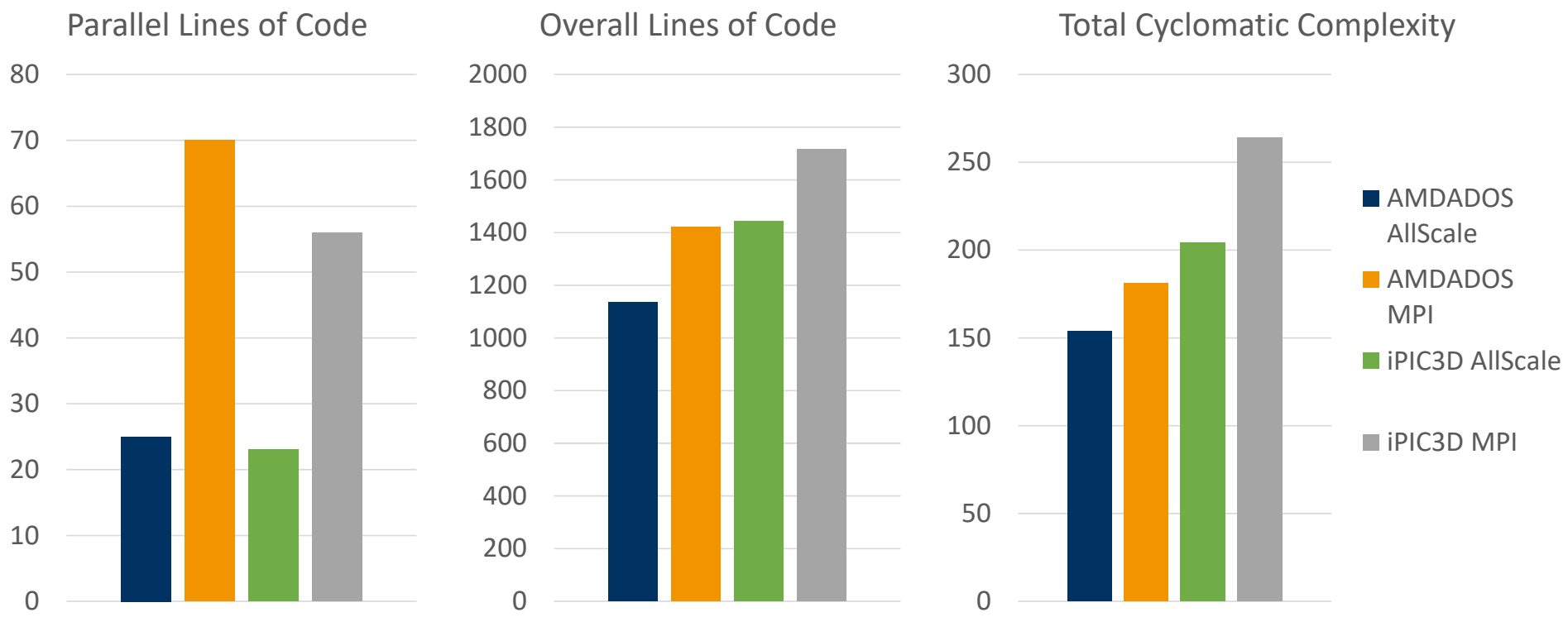
AMDADOS on VSC-3



iPiC3D on VSC-3



Productivity Evaluation



Practical Experience: Benefits and Challenges of AllScale

- Higher-order primites for parallelism and data structures to improve productivity and performance.
- Implicit task parallelism
- 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 in the application

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