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

An Exascale Programming, Multi-objective Optimisation and Resilience Management Environment Based on Nested Recursive Parallelism

AllScale

Enable developers to be **productive**
and to **port** their **applications**
to **any scale of system**

Thomas Fahringer, Herbert Jordan, Peter Thoman
University of Innsbruck, Austria

universität innsbruck

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Queen's University Belfast

KTH VETENSKAP OCH KONST

IBM Ireland

Austrian HPC Meeting 2017

Supercomputing 2017, Denver

In Summary

All Scale

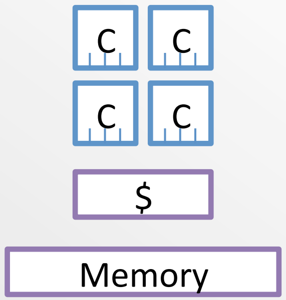
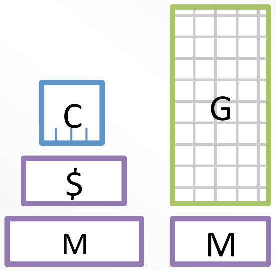
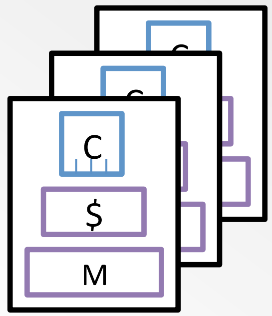
What people think of HPC programming

How computer scientists deal with the algorithm

How domain scientists construct their algorithm

Our future of HPC Programming

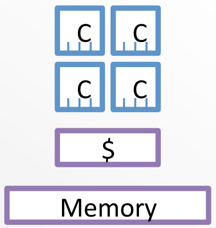
Parallel Architectures

Multicore:	Accelerators:	Clusters:
		
OpenMP/Cilk	OpenCL/CUDA	MPI/PGAS

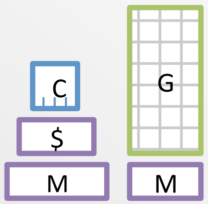
This project has received funding from the European Union's Horizon 2020 research
and innovation programme under grant agreement No. 671603

Real World Architectures

© www.allscale.eu

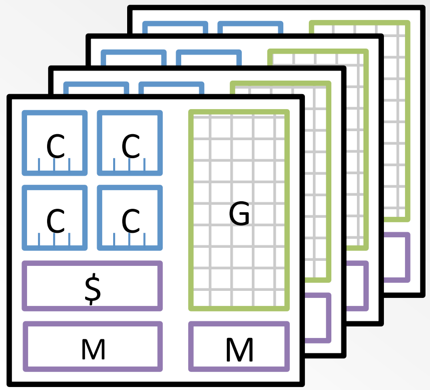


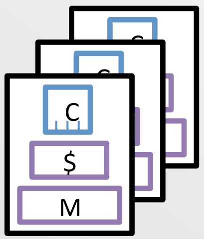
OpenMP/Cilk



OpenCL/CUDA

$\neq$



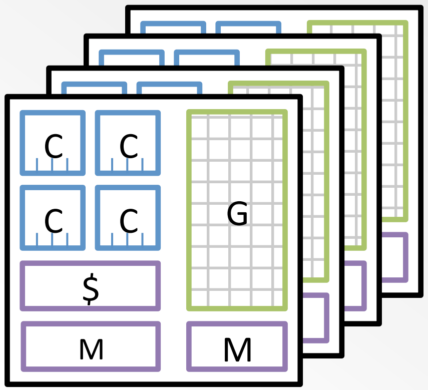


MPI/PGAS

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

Hybrid Codes

- *e.g. MPI+X+Y*
- ***Issues:***
  - **hard-coded** problem **decomposition**
  - **lack of coordination** among runtime systems
- *Limited built-in support for:*
  - portability, auto-tuning, load balancing, monitoring, or resilience



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

AllScale Vision

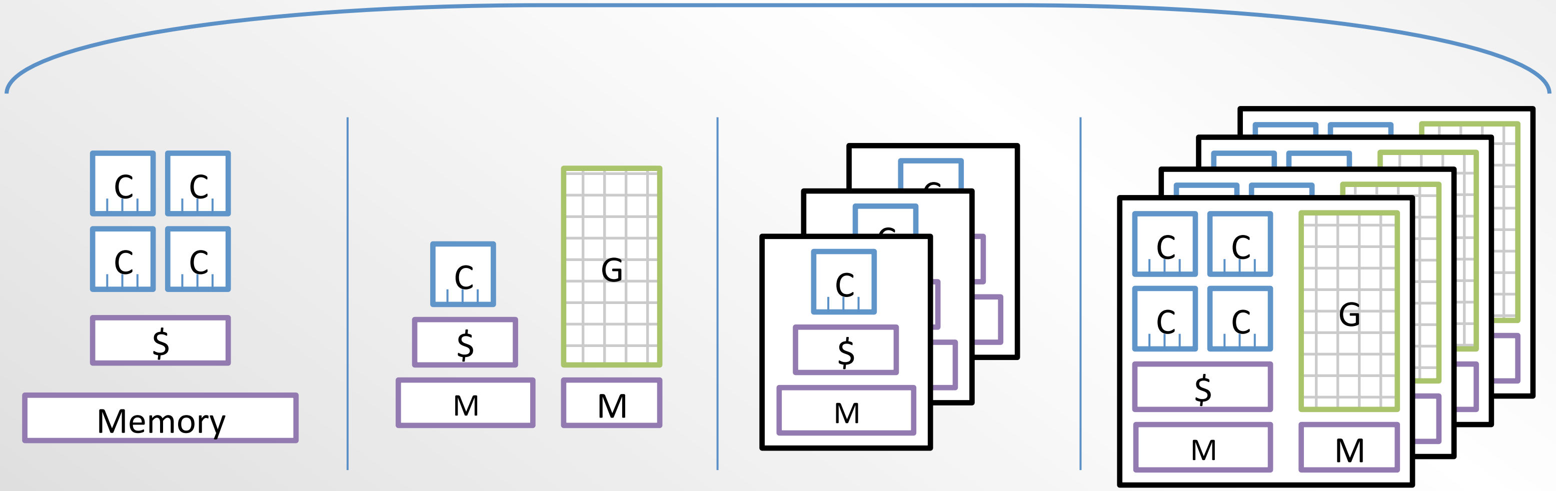
Application

Parallel Algorithm

Unified Parallel Programming Model

Toolchain

Portability, Tuning, and Resilience

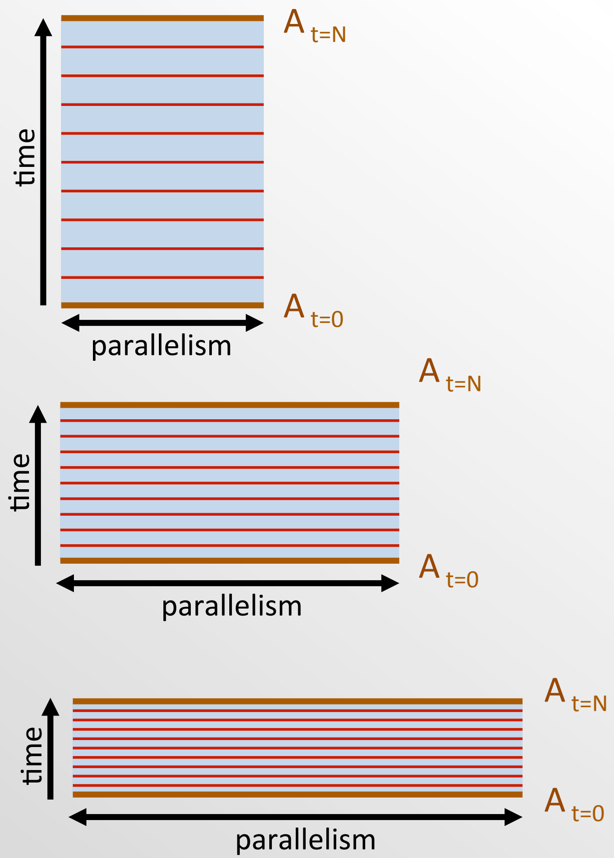


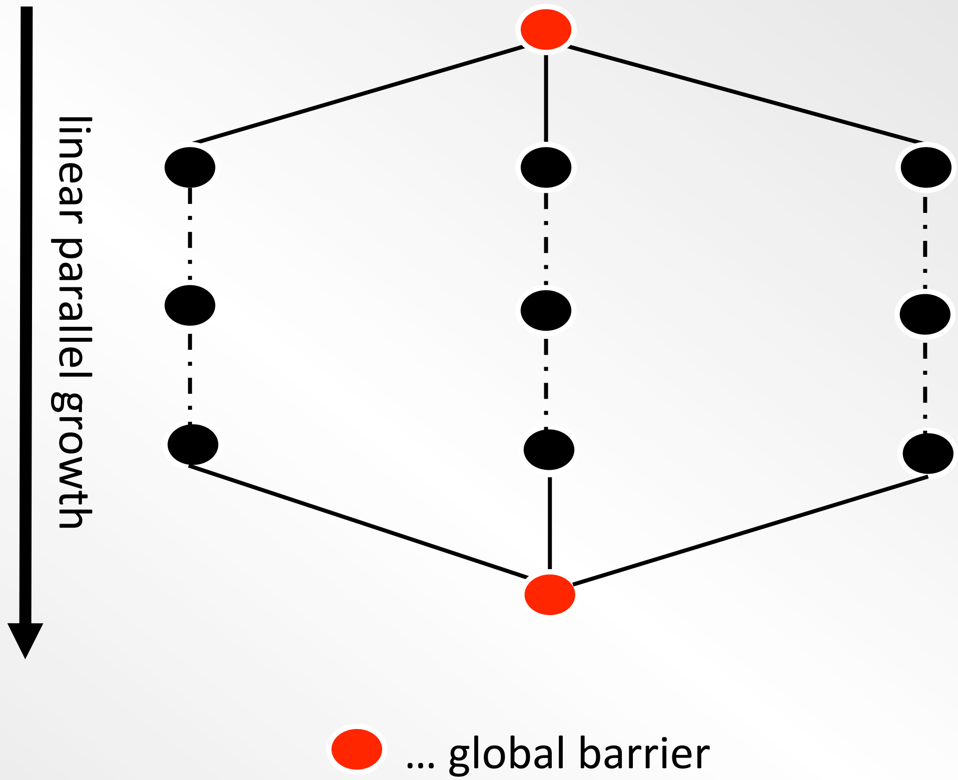
Multicore | Accelerators | Clusters | Heterogeneous Clusters

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

Conventional Flat Parallelism

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

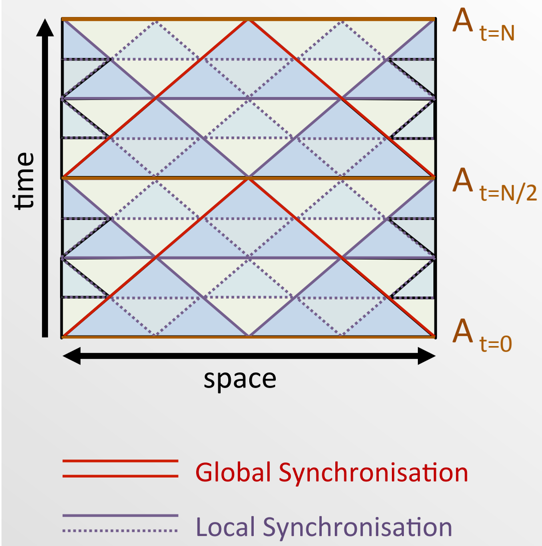


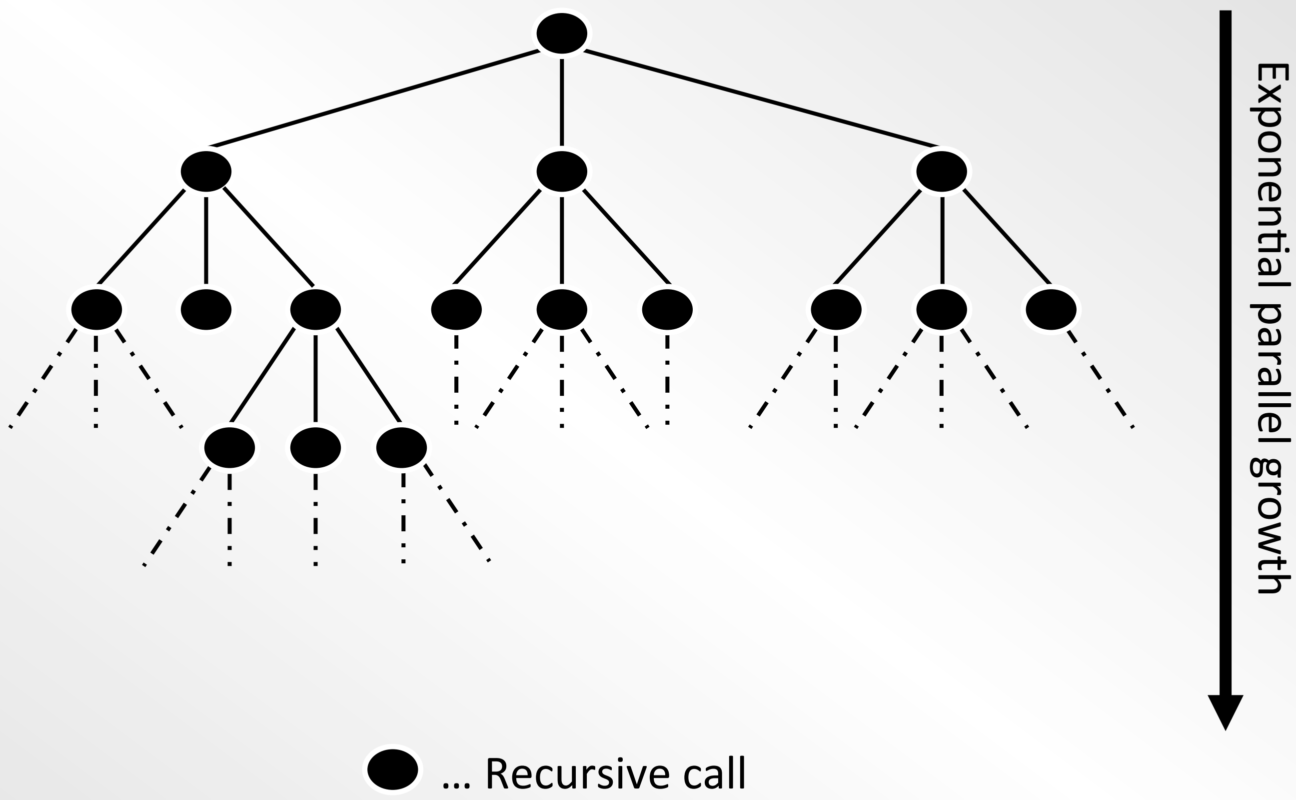


AllScale Core Programming Model

- Try to provide an automatic solution:
  - Performance portability, load balancing, resilience, autotuning
- Our answer:
  Recursive Nested Task Parallelism
  - *Why?*

Recursively Nested Parallelism





Objective

- ***Developers:***
  - focus on application
  - expose maximum amount of parallelism
- ***Toolchain:***
  - utilize parallelism
  - handle data management and portability
  - load balancing, resilience, and tuning

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

API

- Based on C++ templates
  - Widely used industry standard

- Two Layers:

User-Level API

- High-level abstractions (e.g. grids, meshes, stencils, channels)
- Familiar interfaces (e.g. parallel for loops, map-reduce)

implemented based on

Core API

- Generic function template for recursive parallelism
- Set of recursive data structure templates
- Synchronization, control- and data-flow primitives

How does the code look like?

AllScale


```
auto allscale_fib = allscale::prec(
  [](int n) { return n<2; },
  [](int n) { return n; },
  [](int n, const auto& fib) {
    auto x = fib(n-1);
    auto y = fib(n-2);
    return x.get() + y.get();
  } );
```

Base Case Condition

Base Case

Step Case

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

No Recursion Required

AllScale
© www.allscale.eu

- Previous code directly uses core API and is one of the smallest possible examples
- You probably have (at least) two questions:
  - *What about data?*
  - *How am I supposed to write a recursively task parallel version of my HPC code?*

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

What about data?

- The AllScale environment manages data for you
  - Whether to distribute it, keep it up to date, move it to an accelerator, make a backup for resilience, ...
- What it needs for that is a data item type ***T***, which specifies the following types:
  - a type ***F*** for fragments of the data storage
  - a type ***R*** for addressing sub-ranges of the data structure

Domain scientists are **not** expected to write these!
They are part of the user API.

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

How to write a recursively task parallel version for an HPC code?

- The short answer: you don‘t need to.
- There are three options:
  - Directly use allscale::prec.
  - Use mid-level primitives provided by the user API. (e.g. allscale::pfor)
  - Use high-level algorithmic skeletons which fit your application domain (also part of the user API).

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

pfor

AllScale
© www.allscale.eu

start of iterator range (inclusive)

end of iterator range (exclusive)

pfor operator
(generic function)

body function (C++11 lambda)

```
pfor(0,10,[&](int i) {
        A[i] = i;
});
```

iterator variable

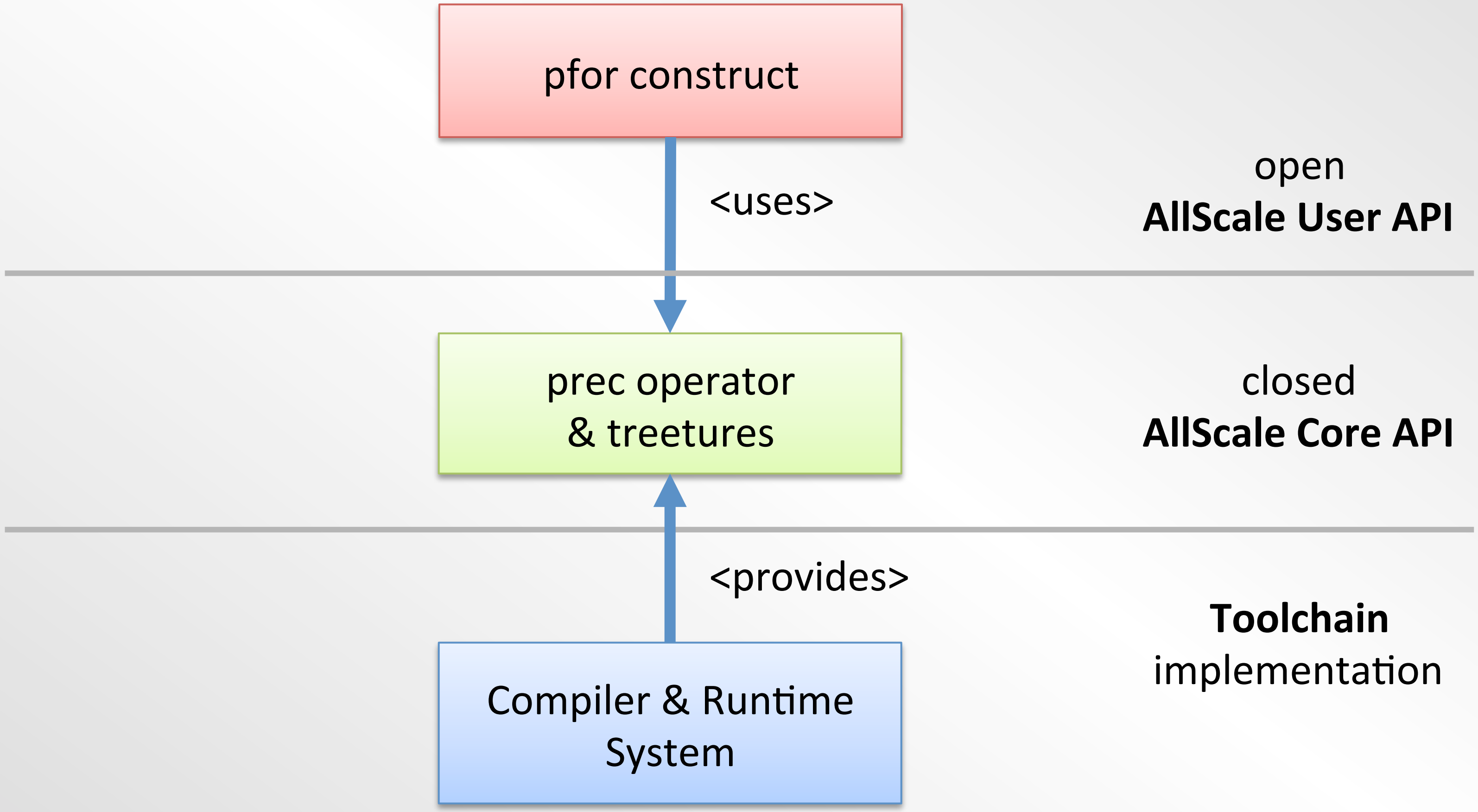
loop body

array A captured by reference

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

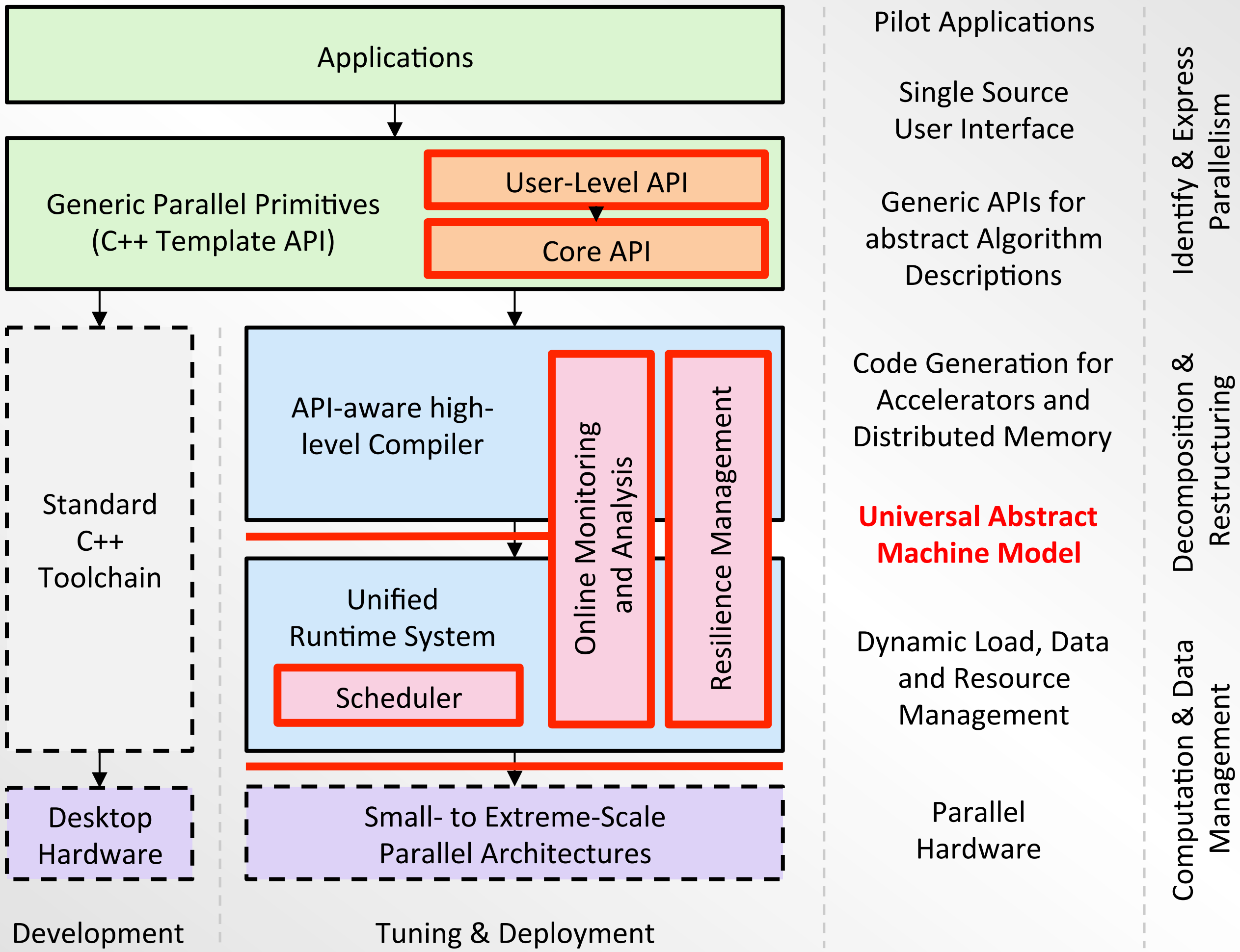
This project has received funding from the European Union‘s Horizon 2020 research
and innovation programme under grant agreement No. 671603

pfor Implementation



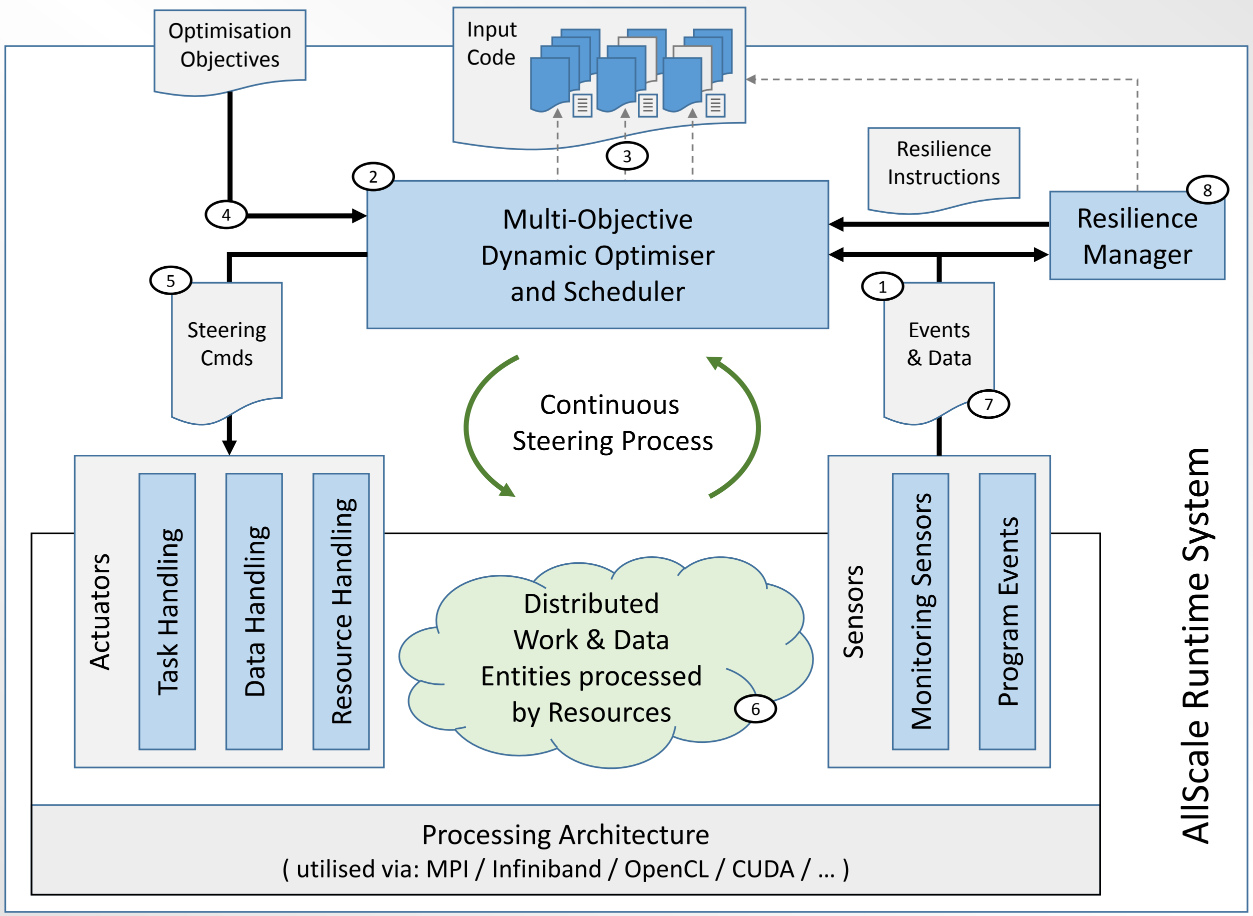
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671603

Interfaces



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

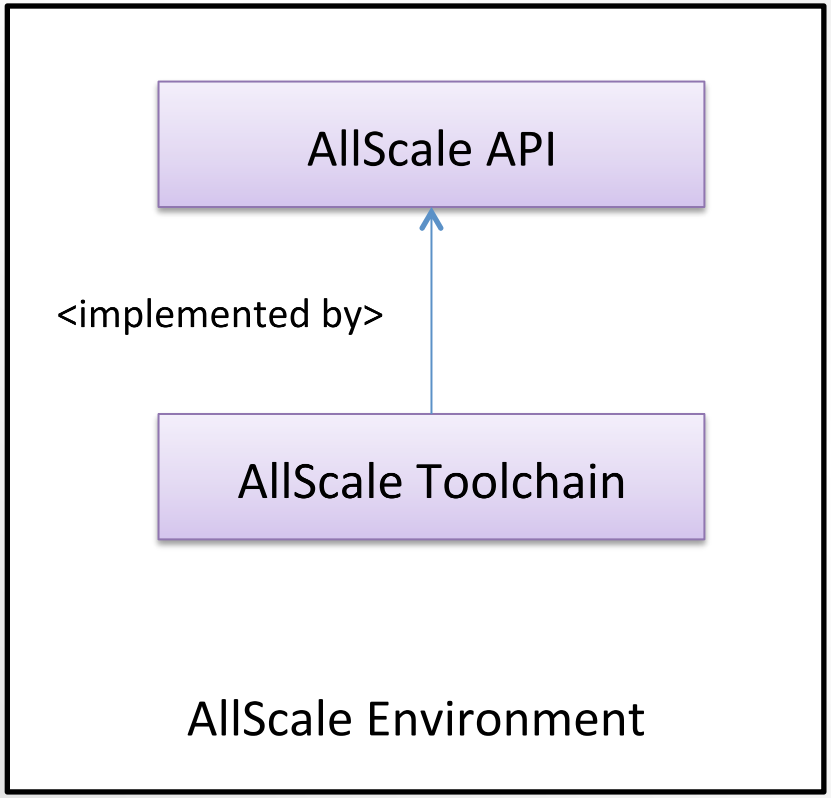
Execution



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

AllScale Products

© www.allscale.eu



Parallel C++
Data Structures
and **Algorithms**

Compiler and
Runtime System
providing
Portability, Tuning,
and Resilience

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

Objective

- ***Developers:***
  - focus on application
  - expose maximum of parallelism
- ***Toolchain:***
  - utilize parallelism
  - handle data management and portability
  - load balancing, resilience, and tuning

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

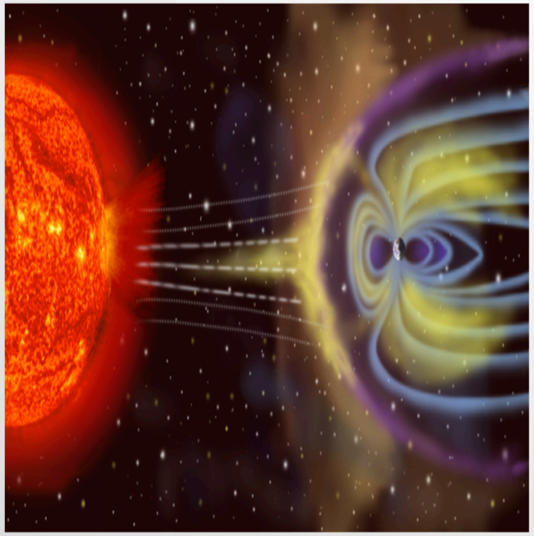
Pilot Applications

© www.allscale.eu

iPIC3D

Implicit particle in-cell code for space weather applications

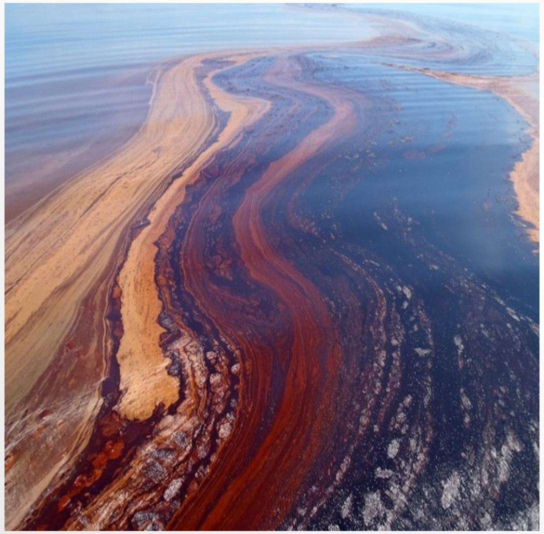
KTH



AmDaDos

Adaptive meshing, data assimilation for dispersion of oils spills

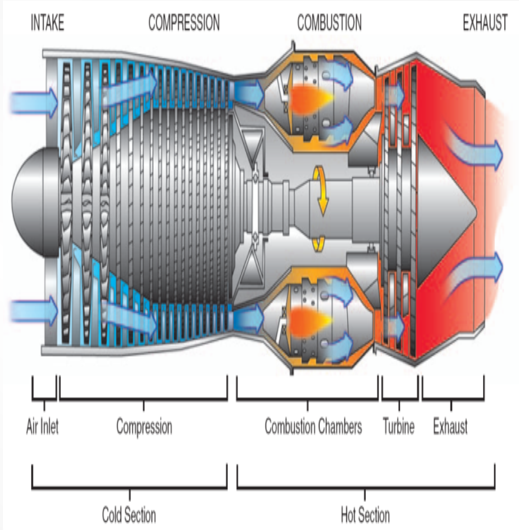
IBM Research



Fine/Open

Large Industrial unsteady CFD simulations

NUMECA



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

Summary

© www.allscale.eu

- **Challenge**
  - Explore recursive task parallelism for extreme scale HPC
- **AllScale**
  - single programming model based on C++ templates
  - main source of parallelism: recursive parallelism
  - single compiler/single runtime system
  - auto-tuning, code-versioning, fault tolerance, on-line monitoring
- **First prototype released with tutorial**

  https://github.com/allscale
- **More information**
  - www.allscale.eu

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

Partners

AllScale

universität innsbruck

KTH VETENSKAP OCH KONST

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

NUMECA INTERNATIONAL

Queen's University Belfast

IBM Ireland

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