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

AllScale

AllScale

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

Thomas Fahringer

University of Innsbruck, Austria

European HPC Summit week: EXDCI Workshop, Prague, May 10, 2016

universität innsbruck

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Queen's University Belfast

KTH VETENSKAP OCH KONST

NUMECA INTERNATIONAL

IBM Ireland

Background

© www.allscale.eu

- **Dominating HPC languages** are
  - tailored for **specific** architecture **designs**
  - largely **static** (e.g. fixed number of threads)

- Most languages promote **flat parallelism** like parallel loops, which imposes the need for **global synchronization**

- Accelerator languages and MPI:
  - Low-level style of programming – **much effort left to the developer**

- **Hybrid parallel programs** may suffer from
  - hard-coded problem decompositions schemas
  - lack of coordination among runtime systems

AllScale Vision

© www.allscale.eu

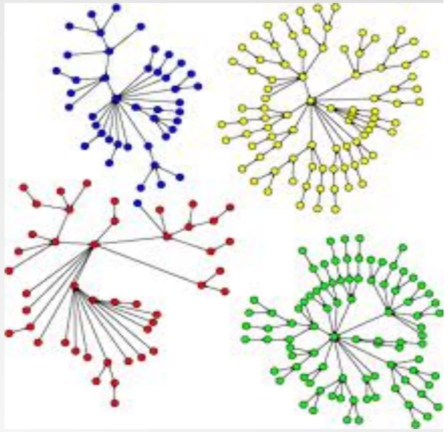
- **Single Source to Any Scale**
  - Write each algorithm only once
    - using a single model of parallelism
    - AllScale tool chain ports it to various architectures
  - scale up and down for any scale of parallel system

- **AllScale tool chain**
  - integrated dynamic load balancing and auto tuning
    - execution time, energy consumption, and power dissipation
  - hardware management (e.g. frequency scaling)
  - automated fault detection and recovery
  - monitoring and profiling tools

- Enable programmers to be **productive** on **any-scale of system**

Recursively Nested Parallelism

- **Requirements for Exascale:**
  - High degree of parallelism on multiple levels (node, socket, core, vector, pipeline)
  - Localized data access and communication

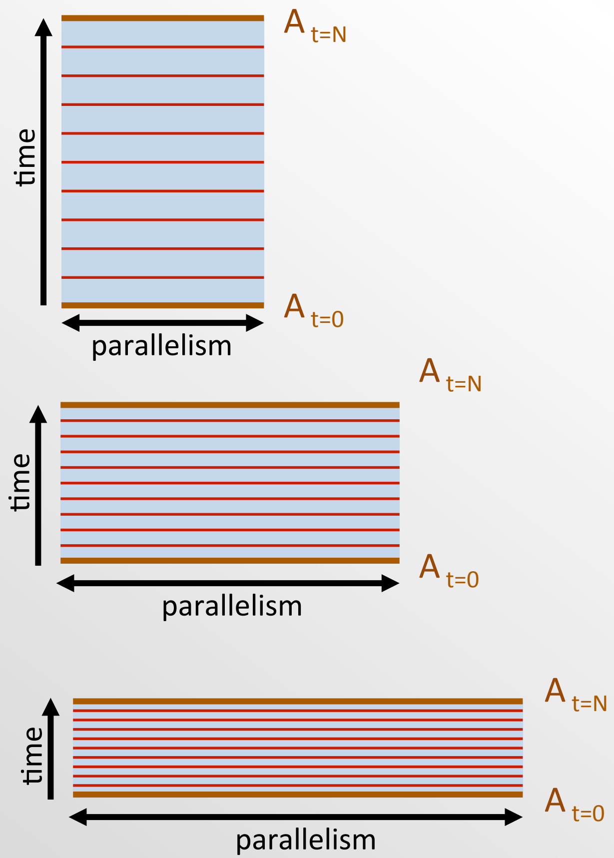


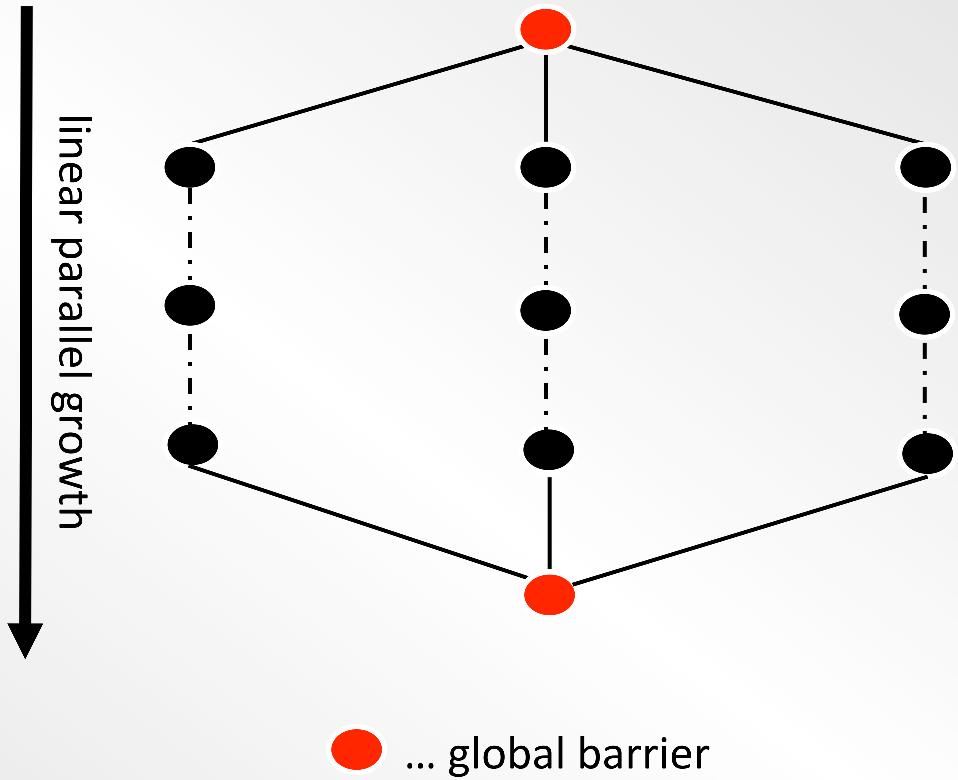
- Solution: **Recursively Nested Parallelism**
  - a hierarchical workload decomposition for a hierarchical hardware infrastructure
  - results in (mostly) locally synchronized parallelism
  - enables fine-grained resilience

Conventional Flat Parallelism

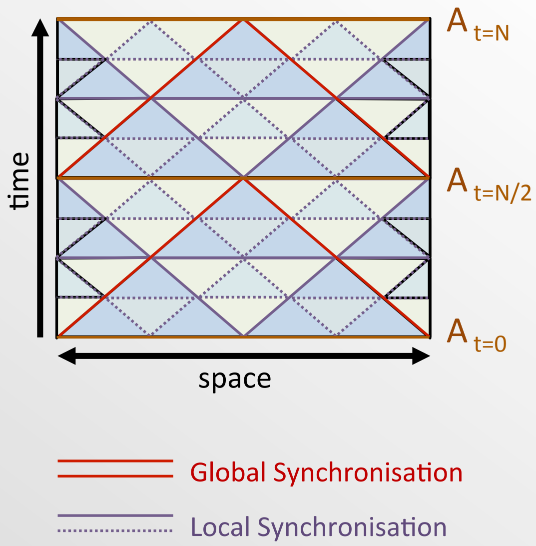
© www.allscale.eu

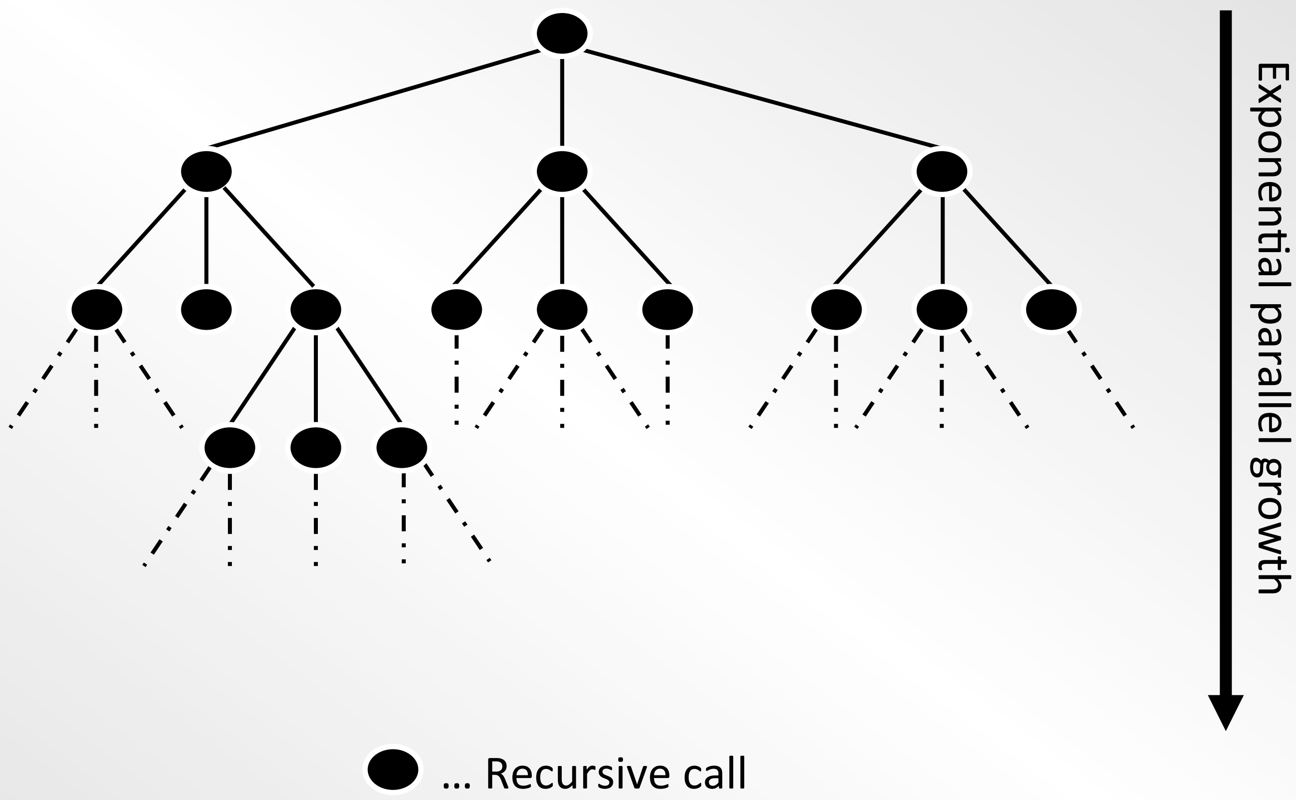
How to map flat parallelism to a hierarchical parallel architecture?
Complex handling of errors – global operations





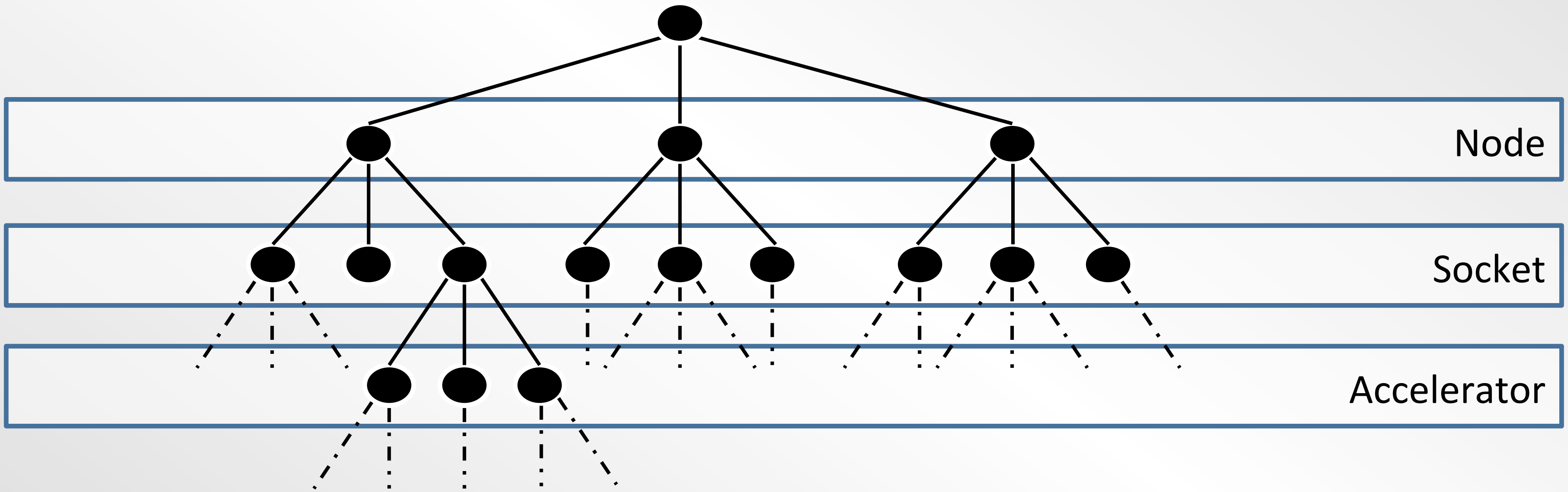
Recursively Nested Parallelism





Recursively Nested Parallelism

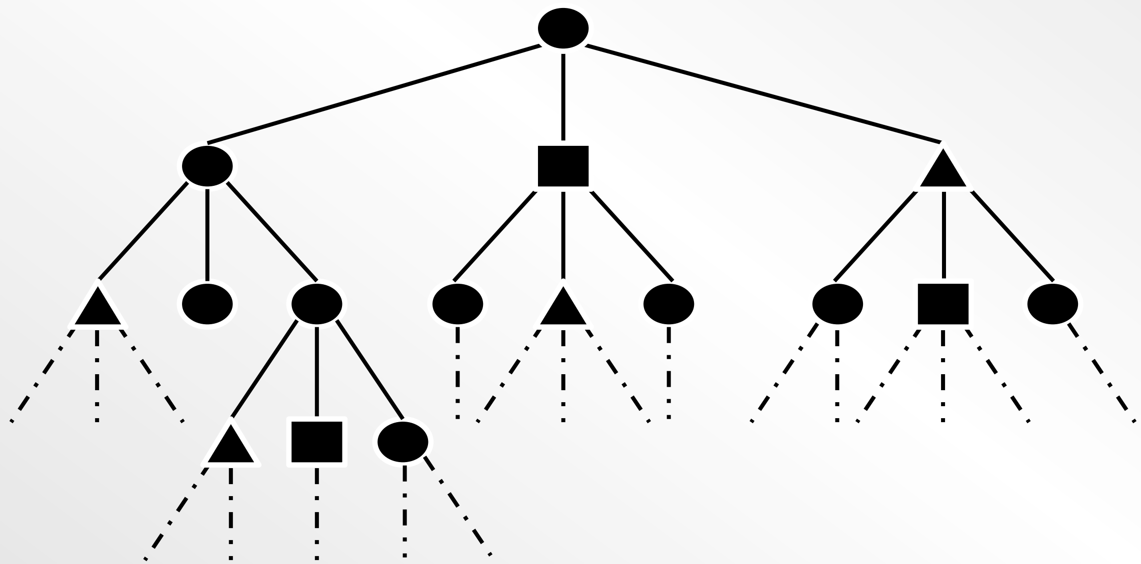
© www.allscale.eu



Maps naturally to multiple levels of HW parallelism

Recursively Nested Parallelism

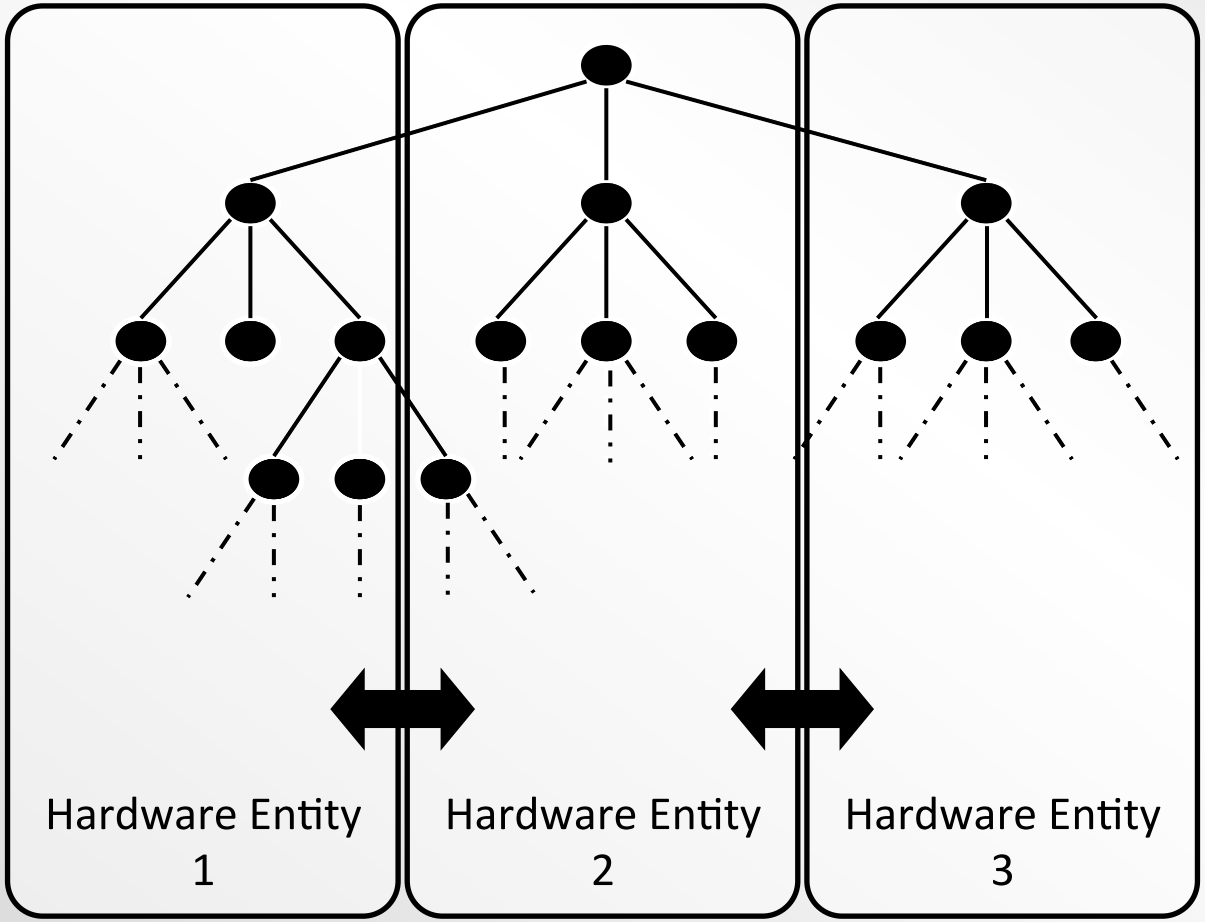
All Scale



Multiversioning allows adaption to hardware & system state

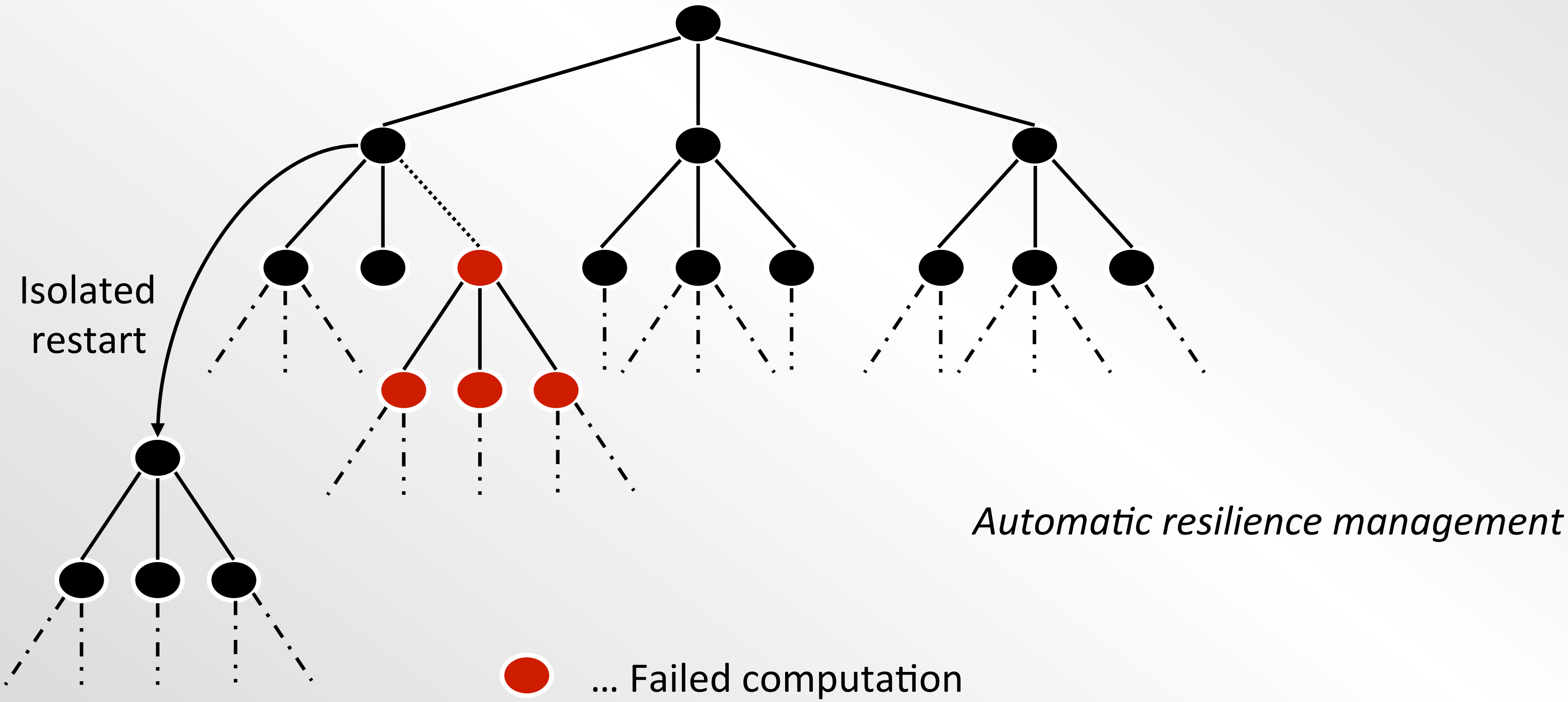
 … Code Versions

Recursively Nested Parallelism

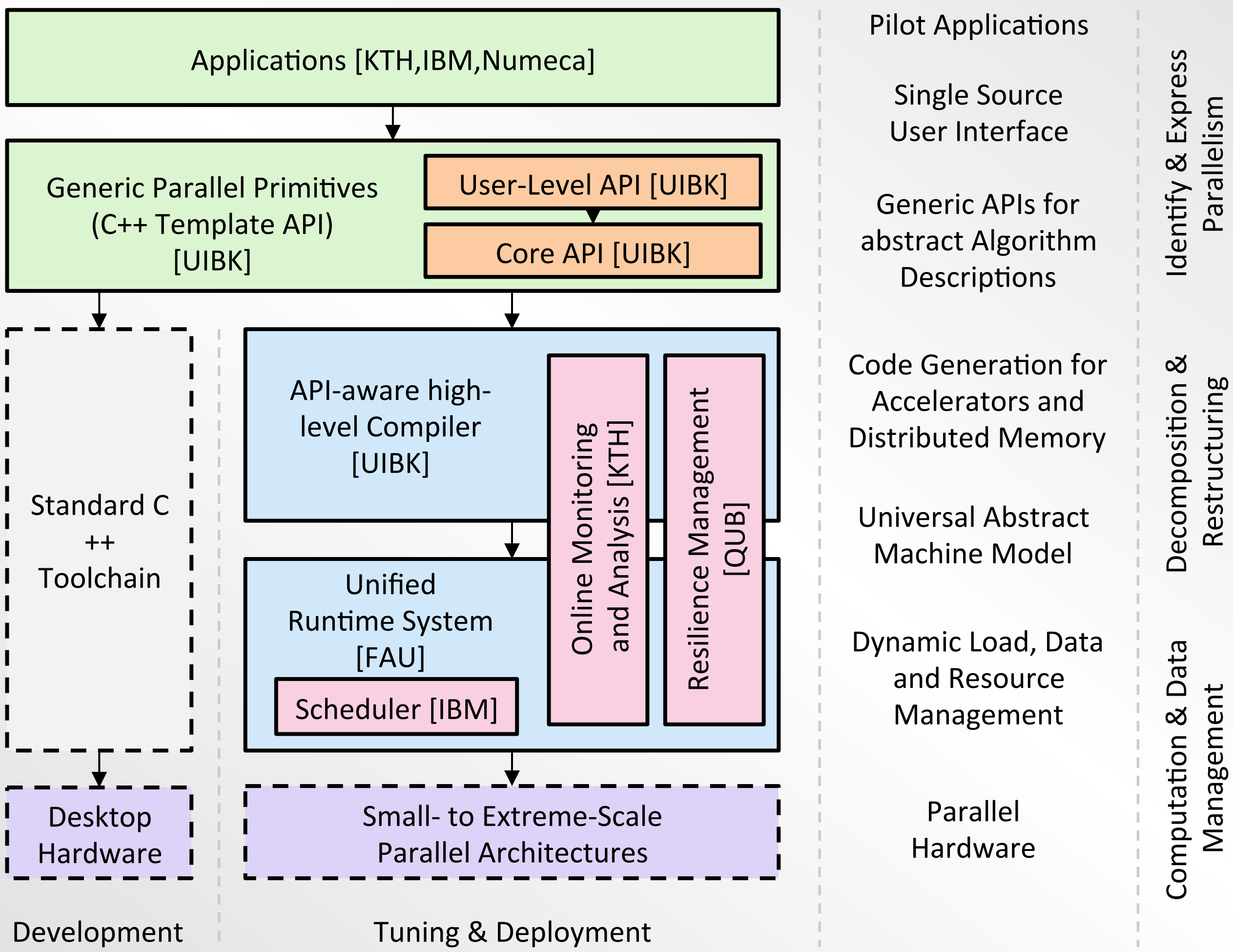


Dynamic load balancing and data migration

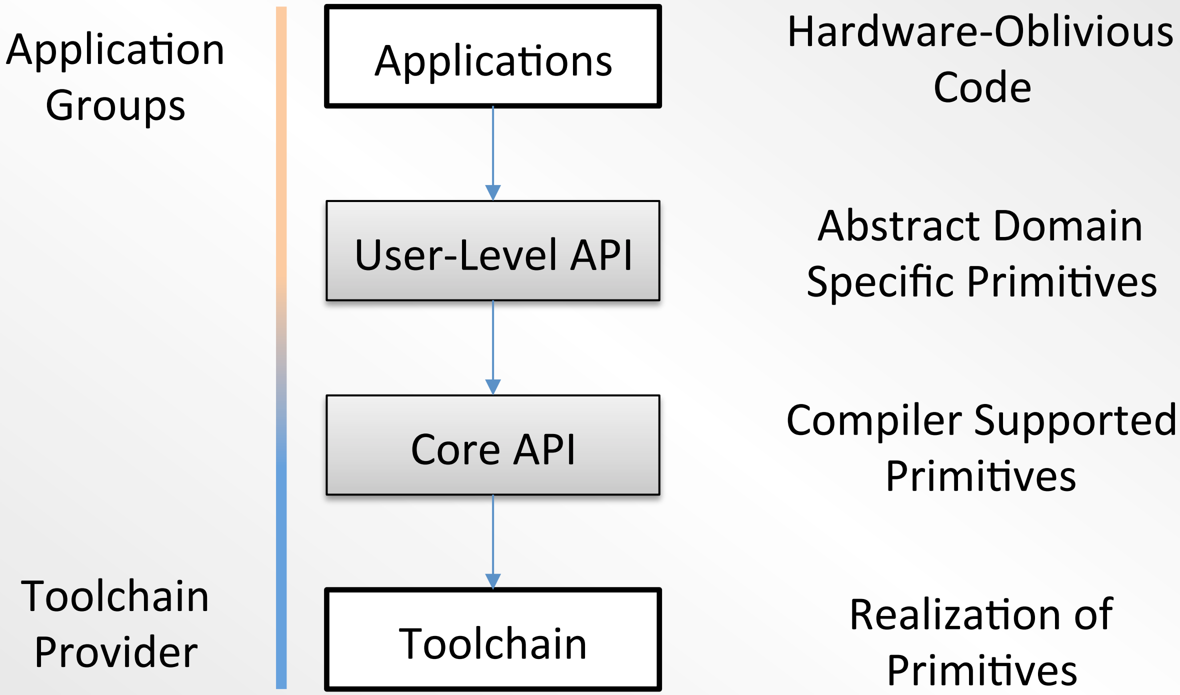
Recursively Nested Parallelism



Architecture



AllScale API



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

- Objectives:
  - Standard C++ tool chains can be used to exploit shared memory parallelism of AllScale generated code.
  - Division of responsibilities among:
    - **Domain**, **HPC**, and **System Level Expert**

Core API

- Main Primitive: rec

  rec (base_test, base, step)

- Semantically equivalent to a parallel version of:

```
auto fun( data ) {
    // check for the base case
    if ( base_test(data) ) return base(data);
    // compute the step case
    return step(data, fun);
}
```

Example fib()


```
rec(
        [](int x) { return x < 2; },
        [](int x) { return x; },
        [](int x, const auto& f) {
                return f(x-1) + f(x-2);
        }
) ;
```

Base Case Condition

Base Case

Step Case

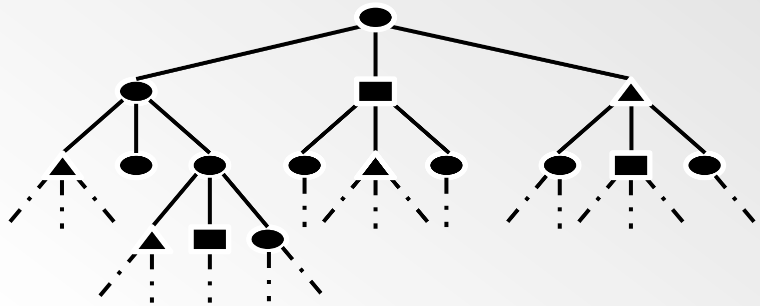
Input Data

User-Level API

- **Based on C++ templates**
  - Widely used industry standard
- **High-level Abstractions**
  - N-body, Stencil, Branch-and-Bound, Linear Algebra, Monte Carlo, Dynamic Programming, …
  - Recursive data structures and algorithms developed by parallelism experts for domain experts.
- **Familiar Primitives**
  - Pfor, Map-reduce, Async, Containers, …
  - Provided to enable upgrade path for legacy applications
- **Standard C++ tool chains** can be used **to exploit shared memory parallelism** of AllScale generated code.

Compiler

- **Analyzes** rec primitive **usage** and **data accesses**
- Generates **multiple code versions** for each step
  – Sequential
  – Shared memory parallel
  – Distributed memory parallel
  – Accelerator



- **Reports** potential **issues** to programmer
  – Data dependencies, race conditions, …
- Provides **additional information to runtime**
  – E.g. type of recursion and data dependencies
  – Improves dynamic optimization potential

Runtime System

- Provides an **abstract parallel machine** as target for compiler-generated code
- **Manages distributed resources**
  - Data locality
  - Communication & synchronization
  - Accelerators
  - Dynamic load balancing
- **Selects** from compiler-generated **code versions**
  - Depending on hardware and execution context

Multi-Objective Optimization

AllScale

- Runtime Scheduler decides:
  - **where** to place data
  - **where** to run **which version** of tasks
  - how to **configure** hardware (e.g. frequency)

- Can be utilized to steer execution towards
  - low **execution time**
  - low **energy consumption**
  - capped **power dissipation**

or a tradeoff of those

Scalable Resilience & Online Performance Analysis

© www.allscale.eu

- **Scalable online performance analysis**
  - instruments, measures, and analyses time, events, energy, power, and faults
  - integrated with runtime system as basis of dynamic optimization decisions
  - integrated with compiler in order to provide profiling data to developers
  - closing the feedback loop

- **Scalable resilience support**
  - directives and/or compiler analysis to guide fault tolerance
  - monitors distributed execution
  - support localized, fine-grained restarting on failures

Queen's University Belfast
Prof. Dimitrios Nikolopoulos

Prof. Erwin Laure
KTH VETENSKAP OCH KONST

AllScale pilot applications

© www.allscale.eu

- **AMDADOS** (IBM Ireland)
  - Adaptive Meshing and Data Assimilation for the Deep water Horizon Oil Spill
- **iPIC3D** (KTH, Sweden)
  - Implicit Particle-in-Cell code for Space Weather Applications
- **Fine/Open** (Numeca, Belgium)
  - Large Industrial unsteady CFD simulations

- **Objective is to understand the achieved gain in their performance improvements.**
  - How => Data Management?
- **Concerns the data and statistics about the result from the project activities (WP5 and WP6):**
  - monitoring data (WP5)
  - output data generated by the pilot applications (WP6)

AllScale Offer to HPC Ecosystem

© www.allscale.eu

- Programming environment for a range of parallel computers including HPC and extreme scale supercomputing.
  - Compiler, runtime system, online performance analysis, resilience management
  - Programming API
- Tutorials and training for our environment.
- Open source HPC applications

AllScale Intl. Cooperations

- Joint development of AllScale runtime system based on HPX – Stellar Group – Lousiana State University

Relations with cPPP, SRA and FETHPC/ CoE projects, PRACE

- Plans for cooperation with CoE POP
  - Performance analysis and optimization
- Access to Prace infrastructure

Role for EsD 2018-2020

All Scale

- Compare AllScale API against other APIs
  - Productivity
  - Performance and scalability
  - Energy/runtime trade-off
- Combine all Auto-tune projects to a single EsD
- Tests to be done on variety of HPC hardware with different benchmarks and applications

AllScale Summary

- **Single high level API close to the user problem**
  - based on existing language and familiar C++ tool chain
  - in contrast to low level and mixed programming paradigms
- **Aggressively exploits flexible and scalable parallelism**
  - nested recursive parallelism
  - supports small scale to extreme scale parallel architectures
  - in contrast to conventional, flat parallelism
- **Holistic compiler and runtime system**
  - no information hiding/encapsulation between different SW layers
  - maintains maximum information across SW stack
- **Resilience and online performance analysis across all SW layers**
- **Multi-objective optimization for runtime, resilience, power, and energy**
  - based on sound theory: pareto front
  - in contrast to ad-hoc approaches

AllScale Consortium

All Scale
© www.allscale.eu