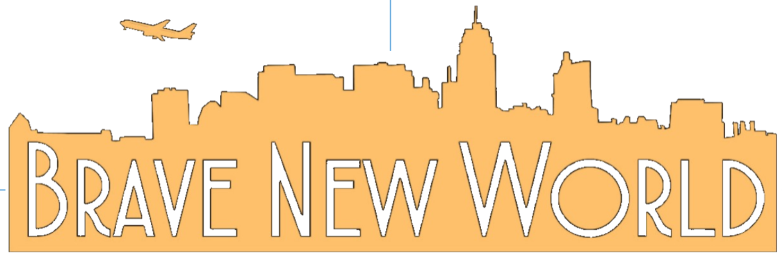
Efficiently Using Architectures in the Era of Customization and Specialization

Prof. Martin Schulz
Chair for Computer Architecture and Parallel Systems, TU Munich
Department of Computer Engineering

Efficiently Using Architectures in the Era of Customization and Specialization and Quantum

Prof. Martin Schulz
Chair for Computer Architecture and Parallel Systems, TU Munich
Department of Computer Engineering

The HPC World is Changing



Trends in HPC

Power and Energy as Hard Constraints

Power Limits

Cost are limiting

Power density is pushing limits

Societal pressure

BRAVE NEW WORLD

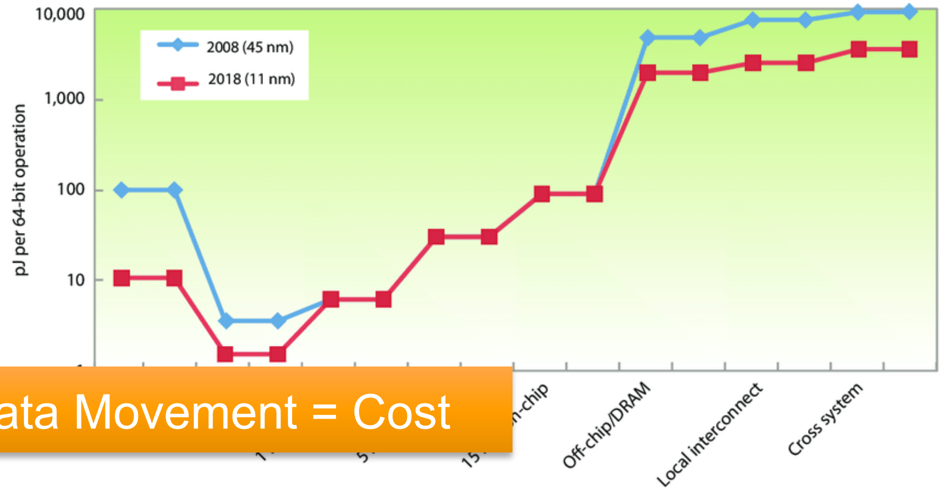
Cost of Data Movement is Becoming a Limiter

Power Limits

Cost are limiting

Power density is pushing limits

Societal pressure



Data Movement = Cost

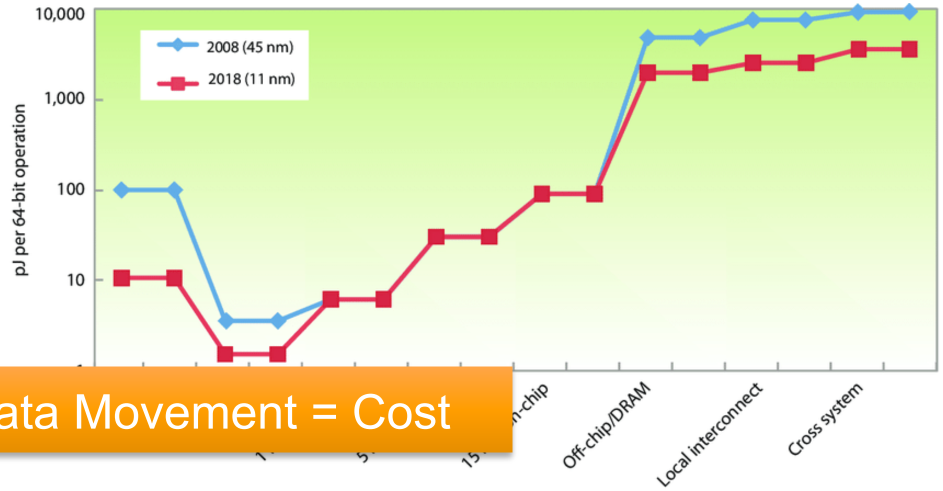
Workloads are Becoming More Complex and Diverse

Power Limits

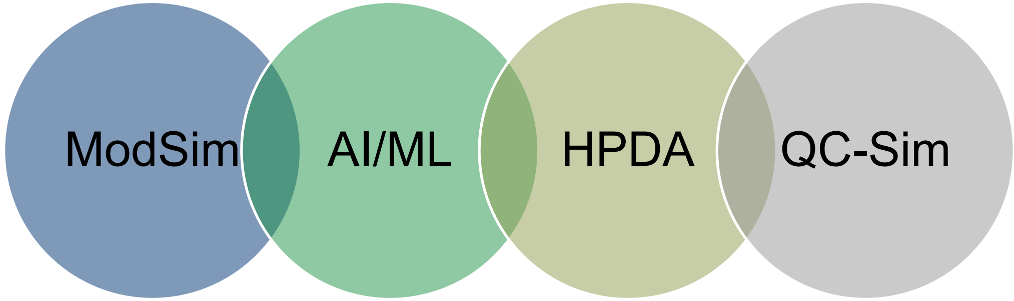
Cost are limiting

Power density is pushing limits

Societal pressure



Data Movement = Cost



More Diverse Workloads

Hybrid Workflows

Trends in HPC

Cambrian Explosion of Architectures

Power Limits

Cost are limiting

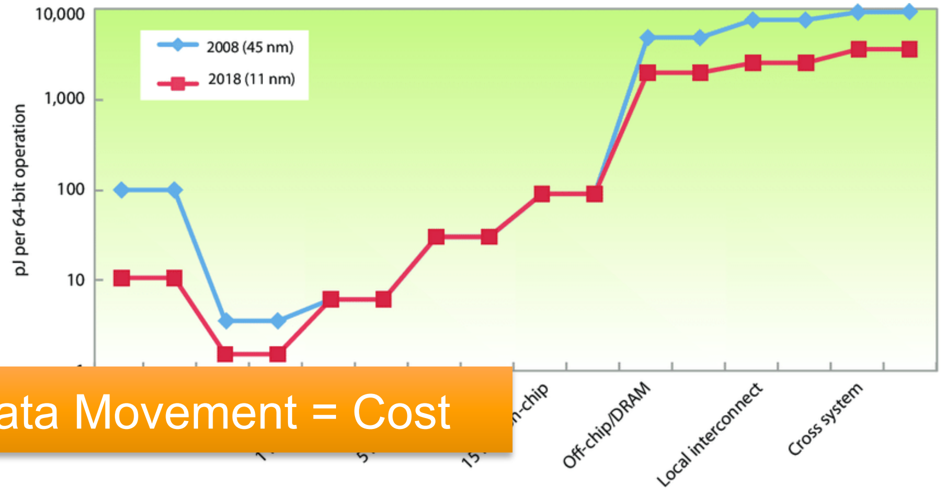
Power density is pushing limits

Societal pressure

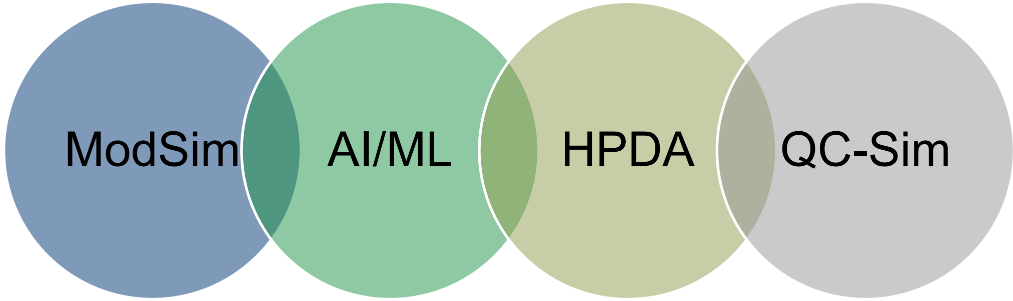
Specialization

Driven by end of Dennard Scaling and Pending end of Moore’s Law

Feature reduction is ending



Data Movement = Cost



ModSim

AI/ML

HPDA

QC-Sim

More Diverse Workloads

Hybrid Workflows

You Cannot Run Away From New Hardware

Customization

- Special purpose architectures
- New instructions/HW blocks
- Memory centric accelerators
- Hardware compression

Chiplets

- Enabling easy access to market
- Reduced engineering cost
- Decreased design times
- Heterogeneous integration

Better GPUs

- Open source GPU architecture
- Building on RISC-V vectors
- CuPBoP portability layer
- Complex ecosystem needed

Common theme:

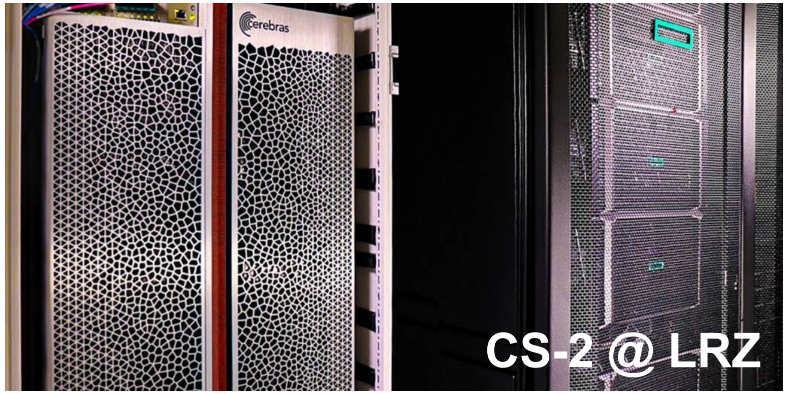
- New hardware, new ISAs, new system designs
- Option for deep integration relying on common memory
- New programming approaches, techniques and models
- New chances for power and energy steering

A Brave New World

Large Scale Accelerators

AI Accelerators

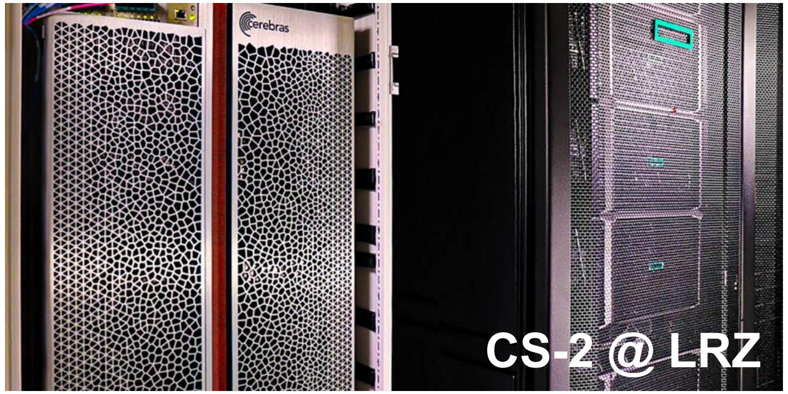
- Physically separated systems
- Tied into highspeed networks
- Data optimized environments
- Separate programming environments
- Focus on lower precision
- Themselves often clustered



Large Scale Accelerators

AI Accelerators

- Physically separated systems
- Tied into highspeed networks
- Data optimized environments
- Separate programming environments
- Focus on lower precision
- Themselves often clustered


CS-2 @ LRZ

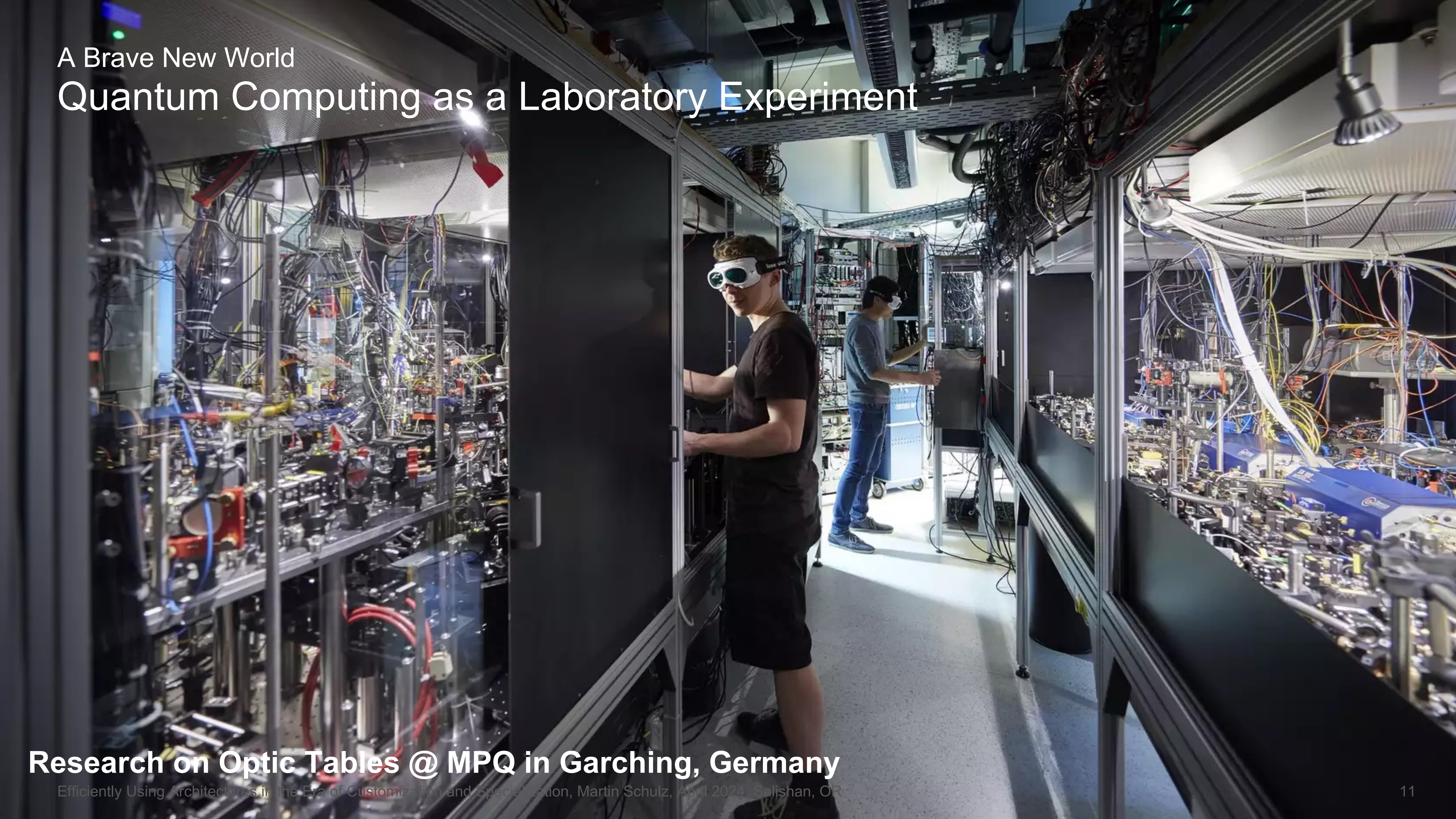
Quantum Systems

- Radically new technology
- High potential, high risk, low readiness (so far)
- Not a standalone technology
  Needs "classic" compute

Source: Bluefors

A Brave New World

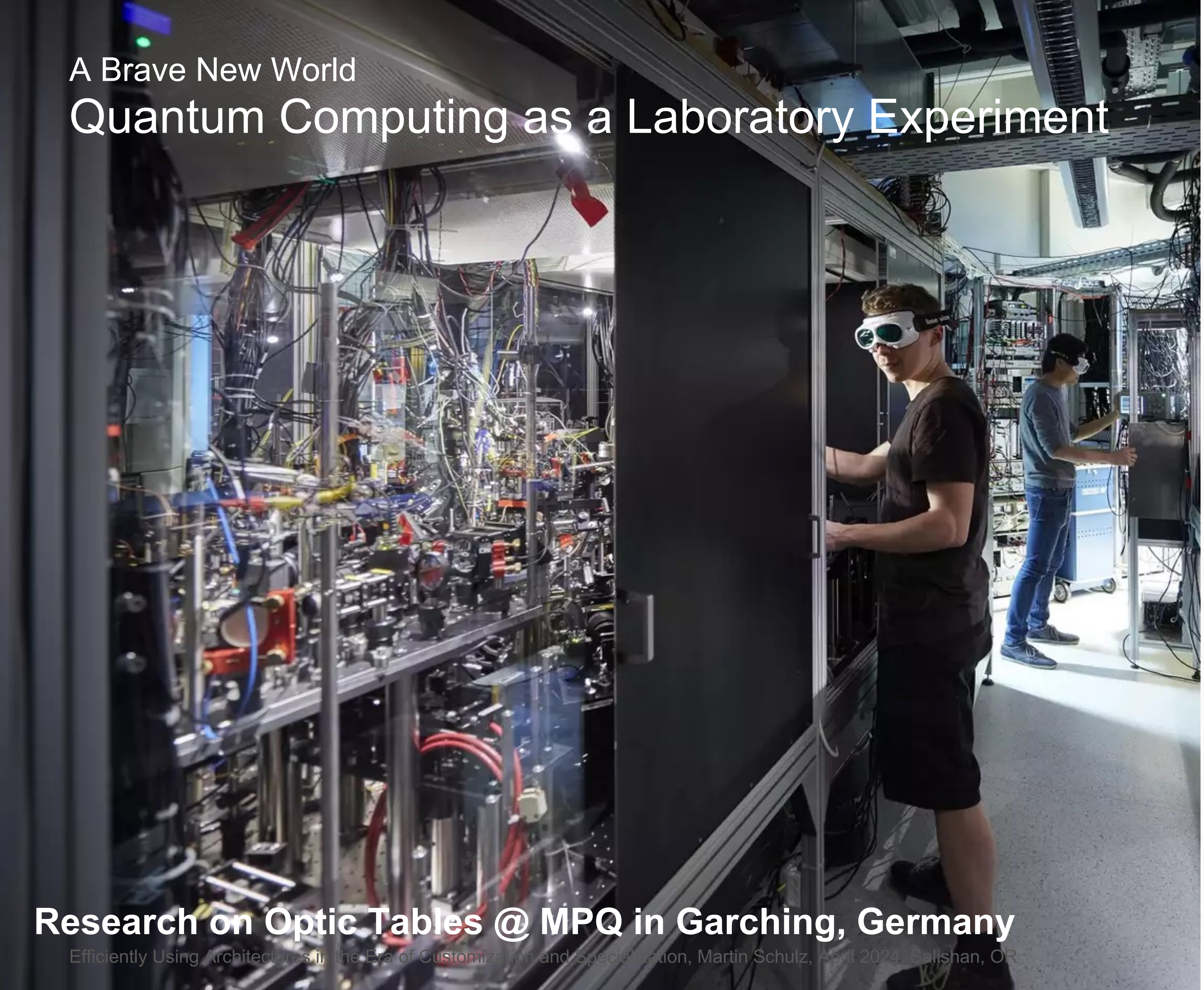
Quantum Computing as a Laboratory Experiment



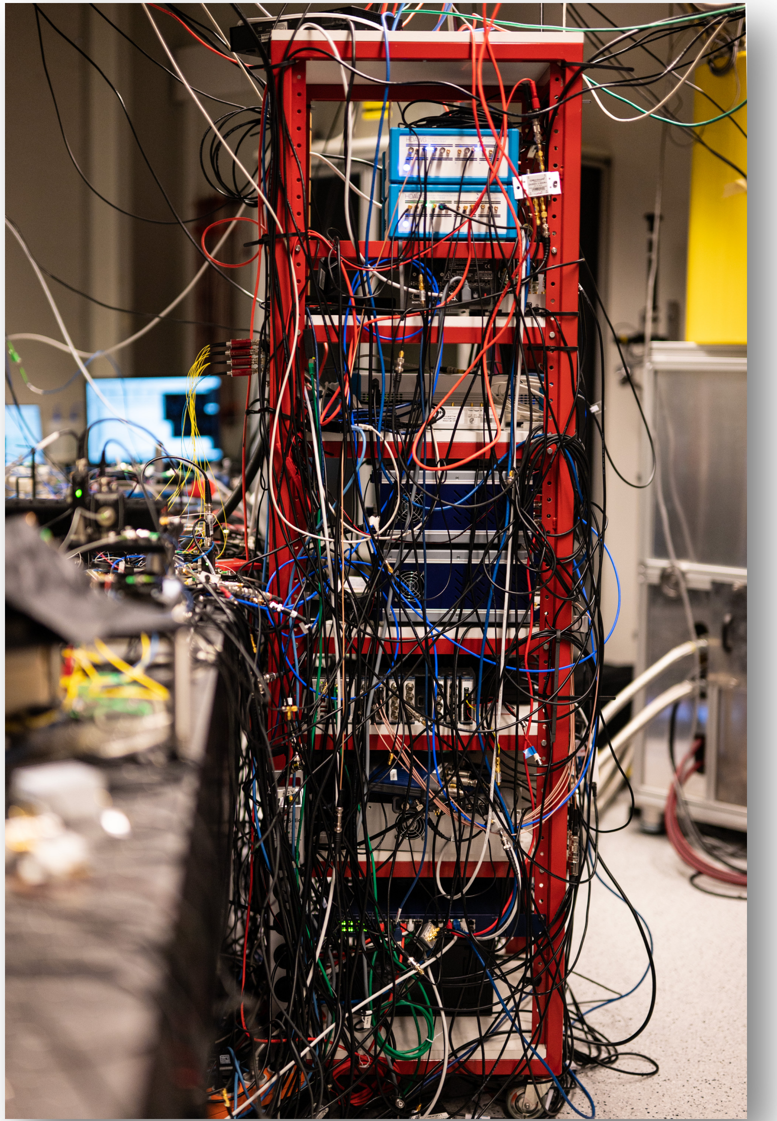
Research on Optic Tables @ MPQ in Garching, Germany

A Brave New World

Quantum Computing as a Laboratory Experiment



Research on Optic Tables @ MPQ in Garching, Germany

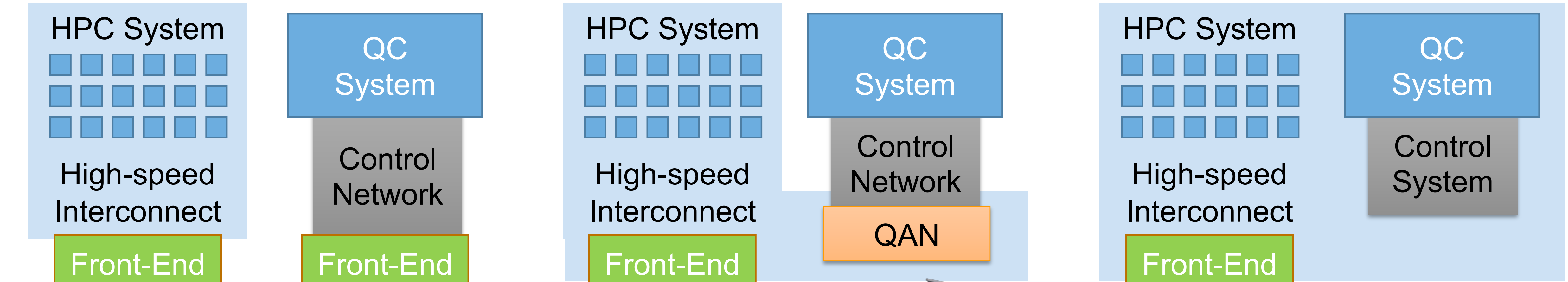


Source: MPQ

Quantum Integration Centre (QIC) at LRZ

HPCQC Integration: Hardware Integration

Reducing the Gap between Host and Accelerator

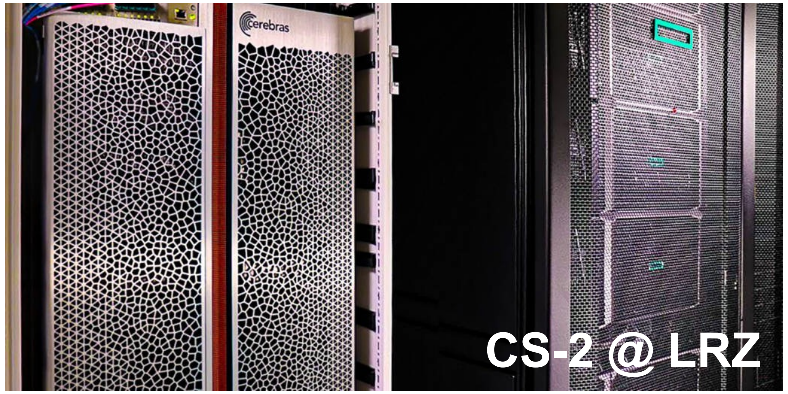


Evolution from network integration to system image i

Large Scale Accelerators

AI Accelerators

- Physically separated systems
- Tied into highspeed networks
- Data optimized environments
- Separate programming environments
- Focus on lower precision
- Themselves often clustered



CS-2 @ LRZ

Quantum Systems

- Radically new technology
- High potential, high risk, low readiness (so far)
- Not a standalone technology
  Needs “classic” compute

Quantum Accelerators

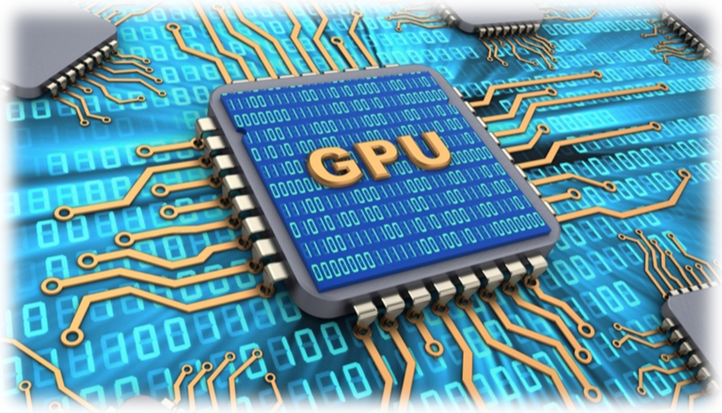
- Heterogeneity to the extreme
- New paradigm with new comp. models
- Many common challenges compared to existing HPC accelerators

Source: Bluefors

Where is the Software?

Accelerated Systems

- Used to be disruptive
- New ways to program
- Abstraction layers
- Today: integrated software and management stack

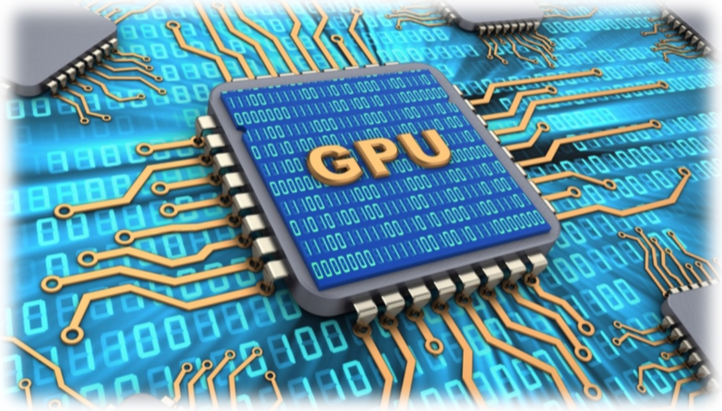


Making use of this new architectural world

Where is the Software?

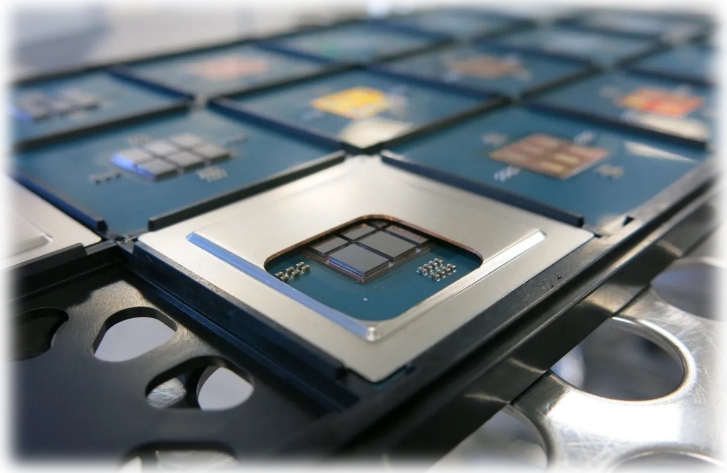
Accelerated Systems

- Used to be disruptive
- New ways to program
- Abstraction layers
- Today: integrated software and management stack



Integrated Systems

- New opportunities
  - Quick load shifts
  - Power shifts
- Need for node sharing
- Impact on Scheduling



Making use of this new architectural world

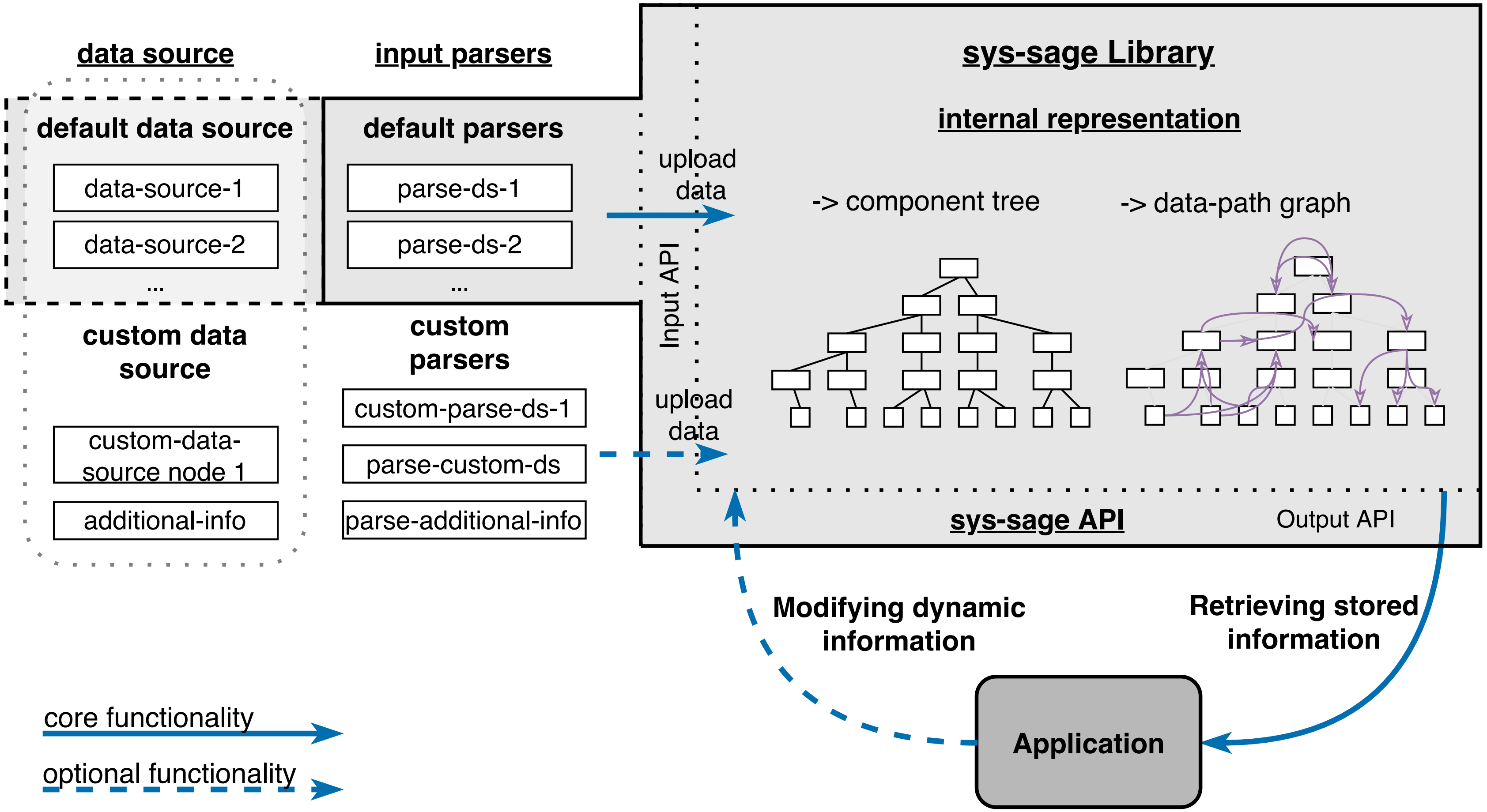
Topology Detection with *sys-sage*

Need to capture topology

- Dynamic
- With attributes
- Expandable

New project: *sys-sage*

- Capture topology
- Dual representation
  - Component tree
  - Data-path graph
- Express dynamic behavior
- Capture system changes



A Unified Representation of Dynamic Topologies & Attributes on HPC Systems, Stepan Vanecek, Martin Schulz, to appear in ICS 2024

Making use of this new architectural world

Where is the Software?

Accelerated Systems

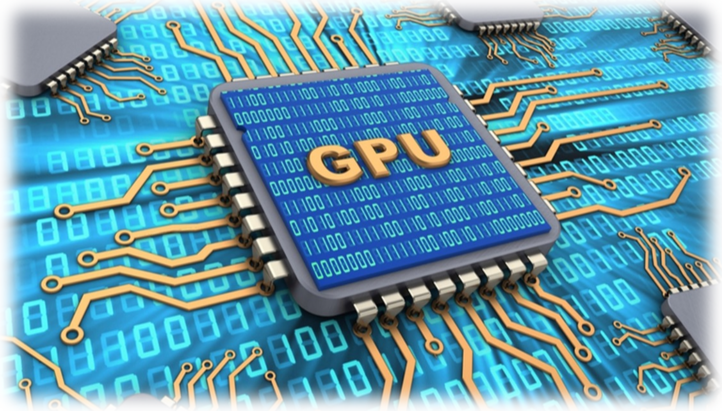
- Used to be disruptive
- New ways to program
- Abstraction layers
- Today: integrated software and management stack

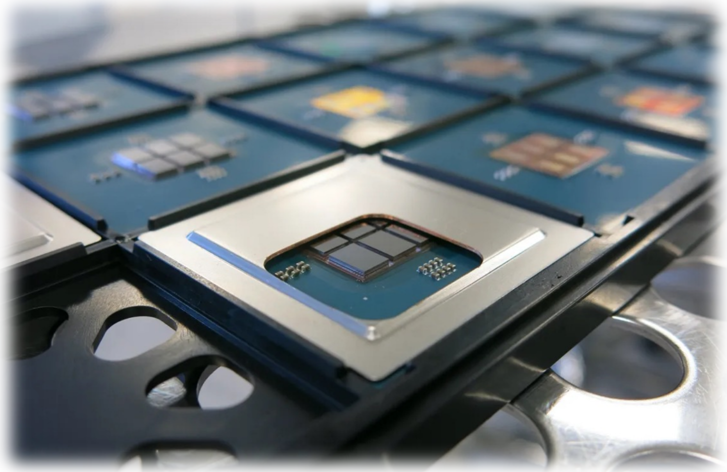
Integrated Systems

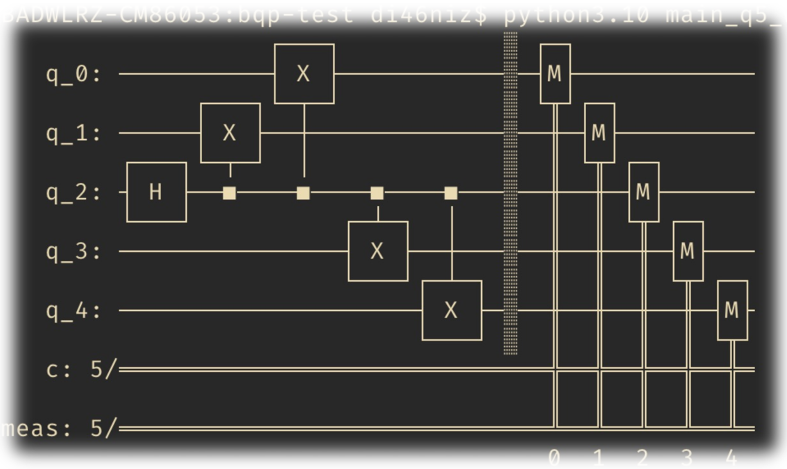
- New opportunities
  - Quick load shifts
  - Power shifts
- Need for node sharing
- Impact on Scheduling

Quantum Paradigm

- Again disruptive
- For now: circuit-based
- New software stack needed





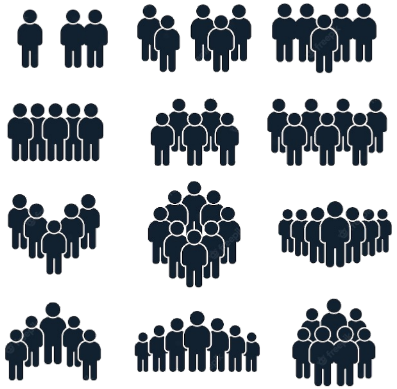


```
BADWLRZ-CM86053:bqp-test di46niz$ python3.10 main_q5_
```

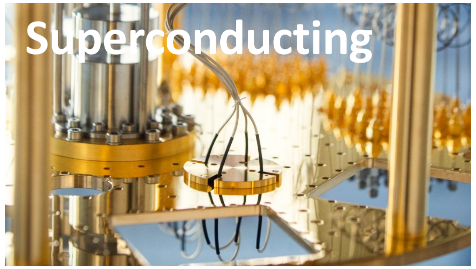
The Munich Quantum Software Stack (MQSS)

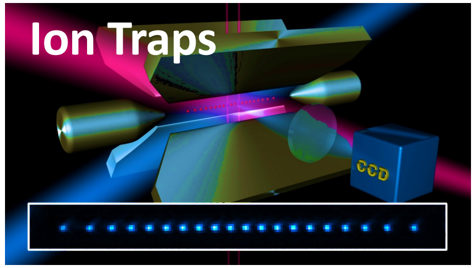
Front-End / Languages

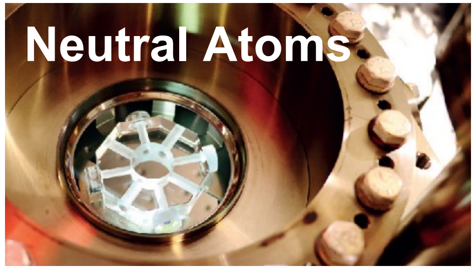
Wide HPC User Communities



Hybrid/Quantum Programming Toolkits / Libraries







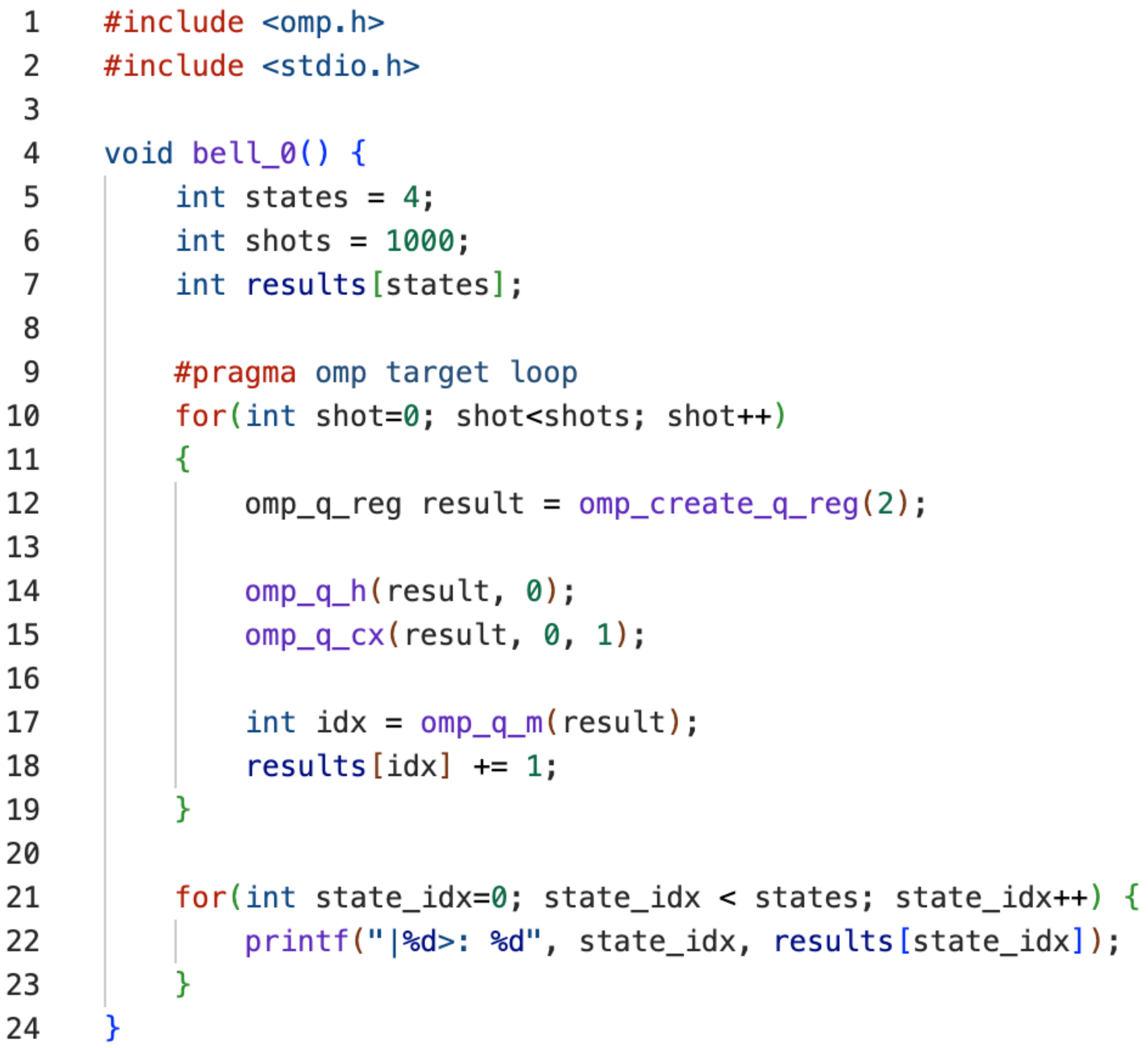
Enabling Domain User Communities to Compute on Quantum Devices

Quantum Systems

OpenMP Quantum Tasks Integration

- Lower learning curve for HPC users
- Benefits from compiler level information instead of a library level
- Possibility to include offloading classical task to “nearby compute”

Quantum Task Offloading with the OpenMP API
Joseph KL Lee, Oliver T Brown, Mark Bull, Martin Ruefenacht, Johannes Doerfert, Michael Klemm, Martin Schulz
Posters at SC23



```
#include <omp.h>
#include <stdio.h>

void bell_0() {
    int states = 4;
    int shots = 1000;
    int results[states];

    #pragma omp target loop
    for(int shot=0; shot<shots; shot++)
    {
        omp_q_reg result = omp_create_q_reg(2);

        omp_q_h(result, 0);
        omp_q_cx(result, 0, 1);

        int idx = omp_q_m(result);
        results[idx] += 1;
    }

    for(int state_idx=0; state_idx < states; state_idx++) {
        printf("|%d>: %d", state_idx, results[state_idx]);
    }
}
```

The Munich Quantum Software Stack (MQSS)

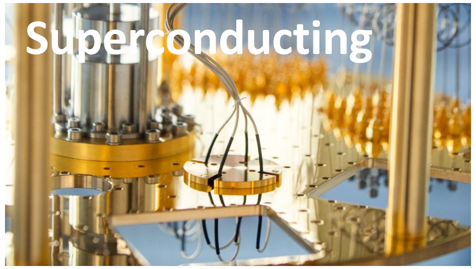
HPC Access

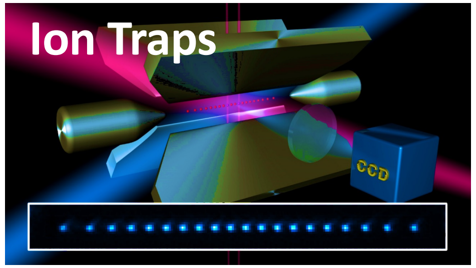
Wide HPC User Communities

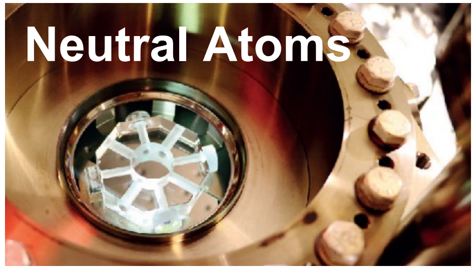
Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator





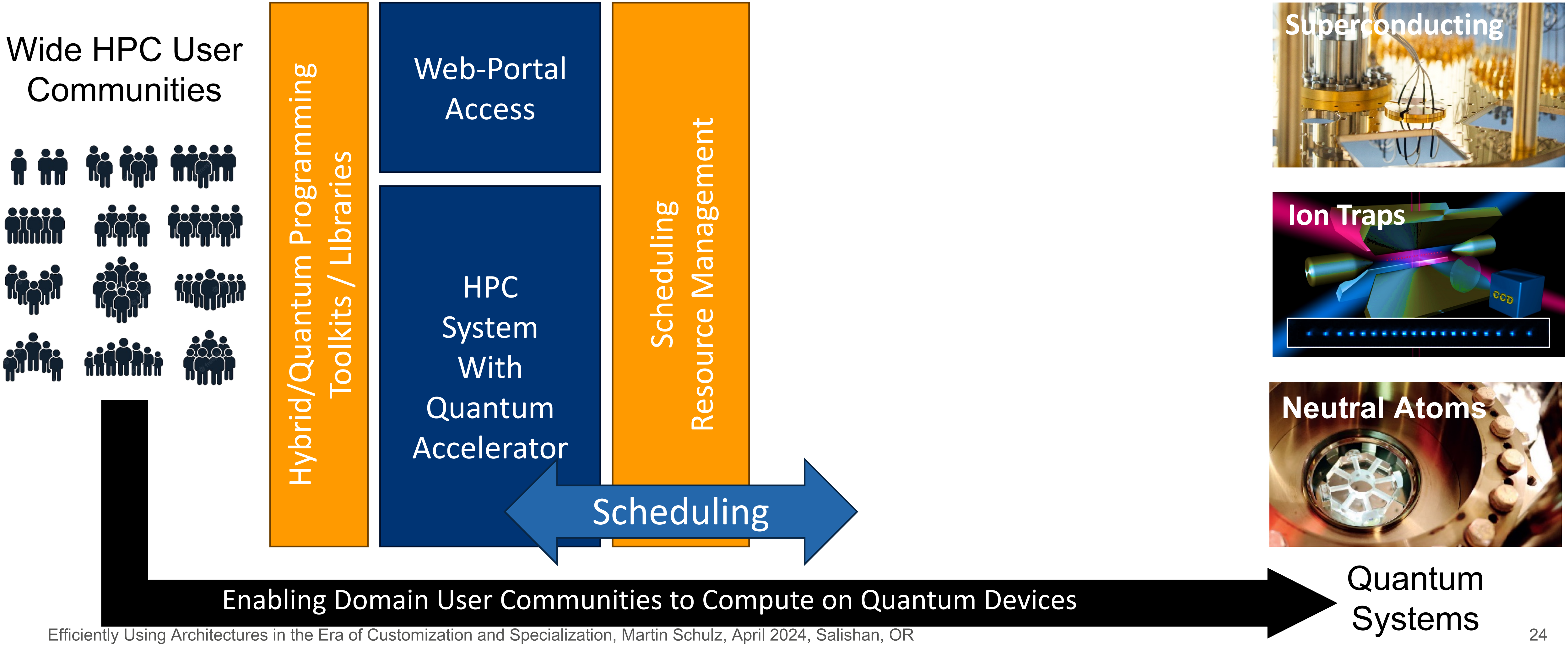


Quantum Systems

Enabling Domain User Communities to Compute on Quantum Devices

The Munich Quantum Software Stack (MQSS)

Scheduling & Resource Management



Wide HPC User Communities

Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

Scheduling Resource Management

Scheduling

Superconducting

Ion Traps

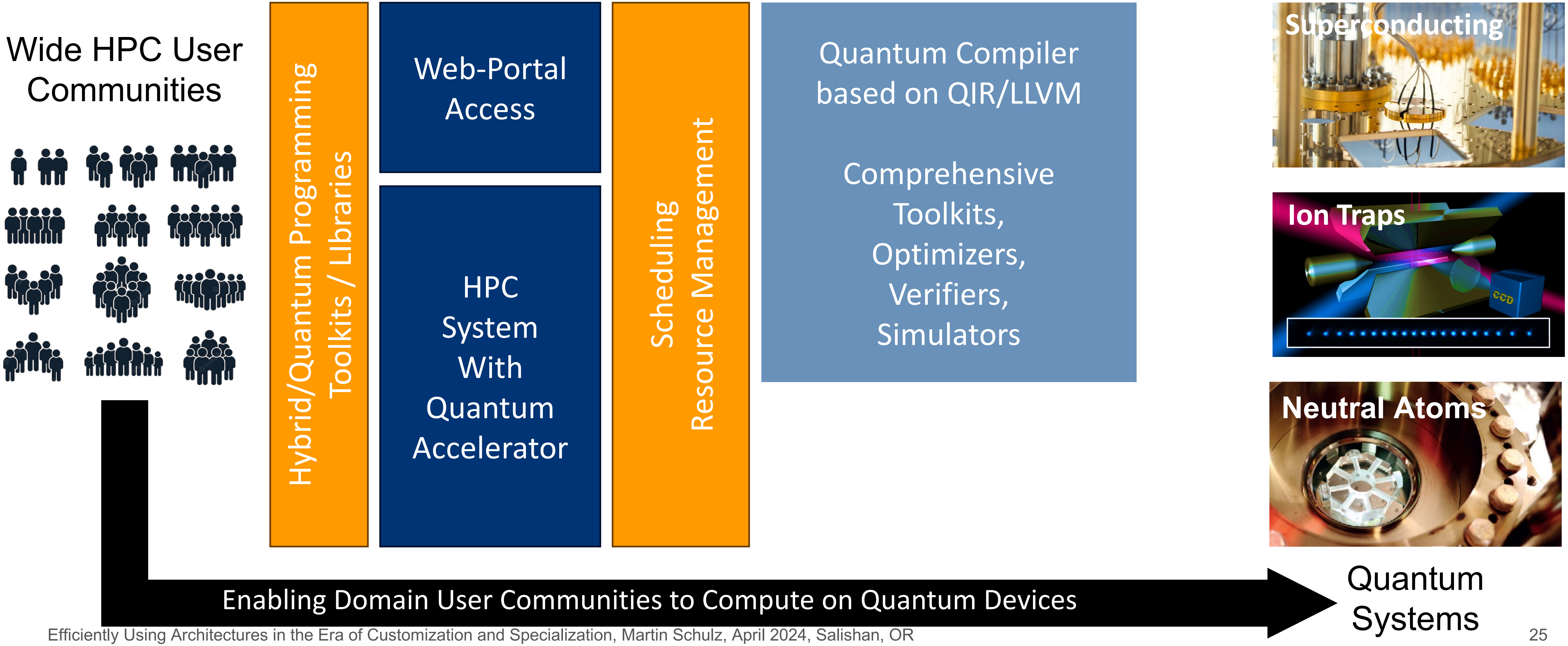
Neutral Atoms

Enabling Domain User Communities to Compute on Quantum Devices

Quantum Systems

The Munich Quantum Software Stack (MQSS)

Quantum Compiler



Wide HPC User Communities

Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

Scheduling Resource Management

Quantum Compiler based on QIR/LLVM

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

Superconducting

Ion Traps

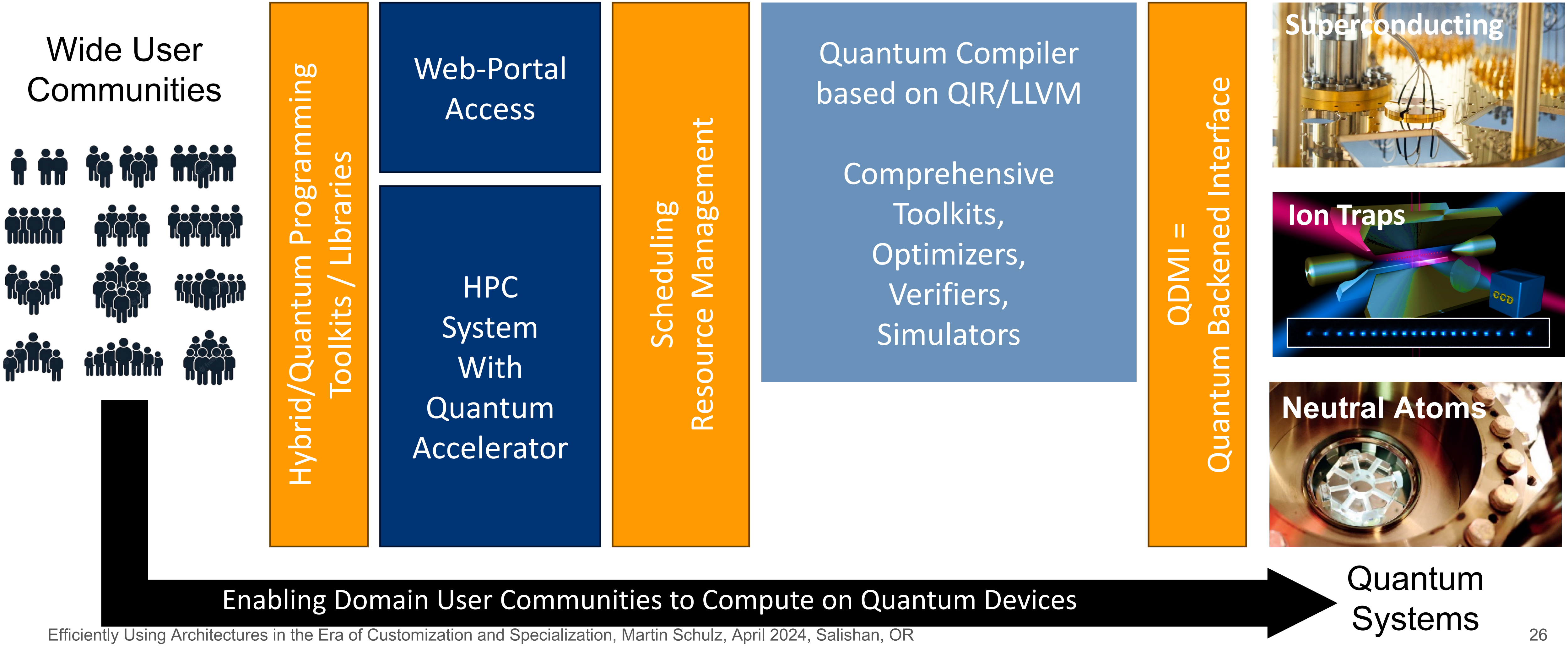
Neutral Atoms

Quantum Systems

Enabling Domain User Communities to Compute on Quantum Devices

The Munich Quantum Software Stack (MQSS)

The QDMI Backend



Wide User Communities

Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

Scheduling Resource Management

Quantum Compiler based on QIR/LLVM

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

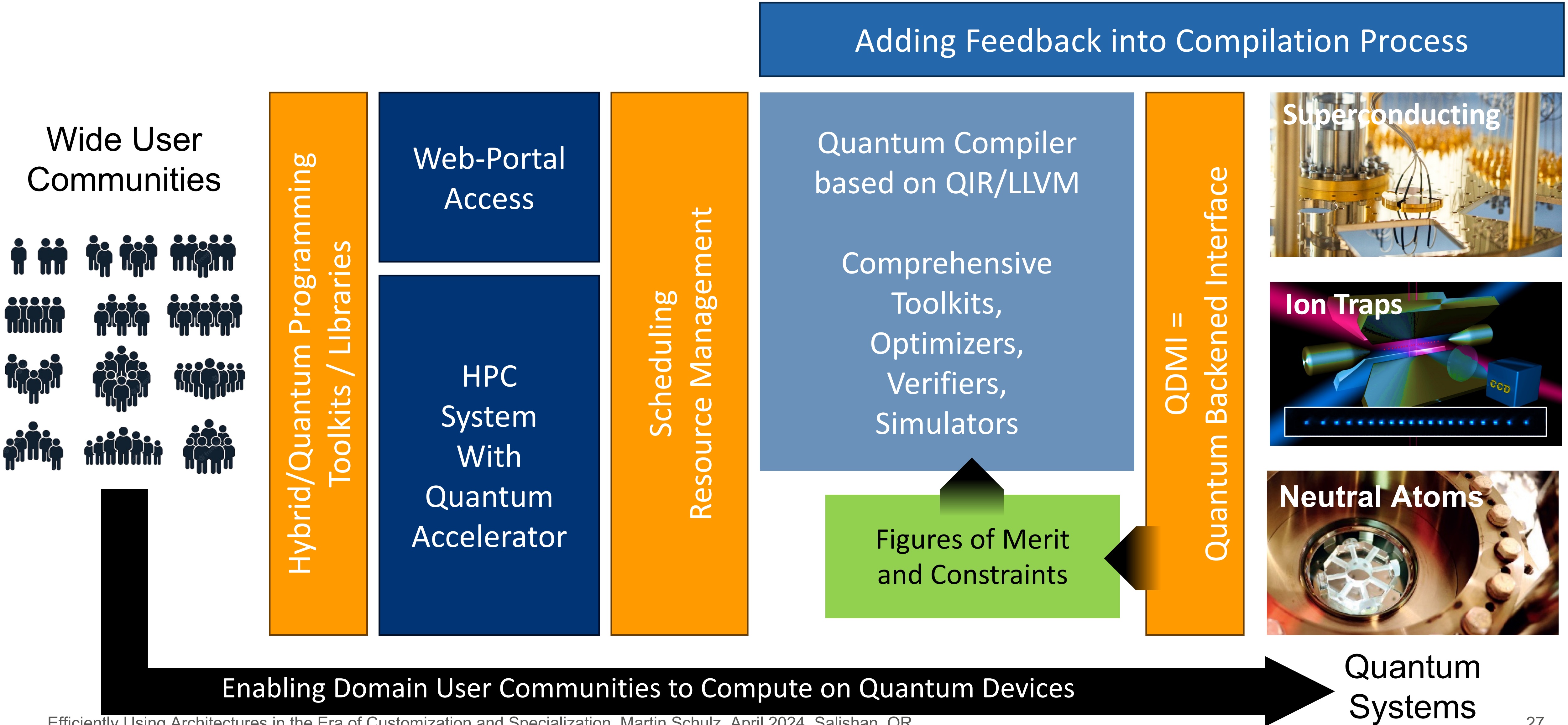
QDMI = Quantum Backened Interface

Enabling Domain User Communities to Compute on Quantum Devices

Quantum Systems

The Munich Quantum Software Stack (MQSS)

Feedback from the Target System



Adding Feedback into Compilation Process

Wide User Communities

Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

Scheduling Resource Management

Quantum Compiler based on QIR/LLVM

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

Figures of Merit and Constraints

QDMI = Quantum Backened Interface

Superconducting

Ion Traps

Neutral Atoms

Quantum Systems

Enabling Domain User Communities to Compute on Quantum Devices

Making use of this new architectural world

Where is the Software?

Accelerated Systems

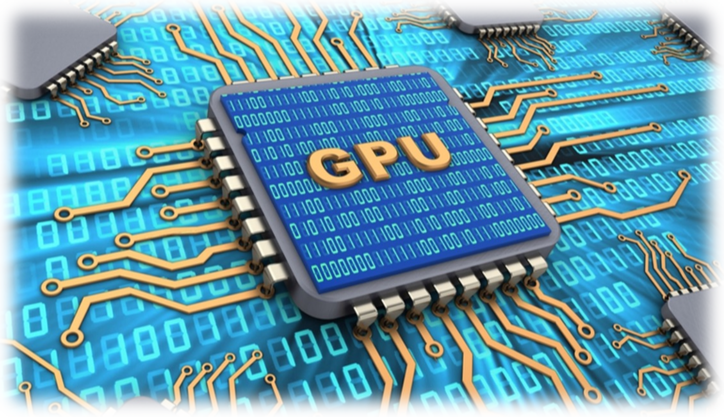
- Used to be disruptive
- New ways to program
- Abstraction layers
- Today: integrated software and management stack

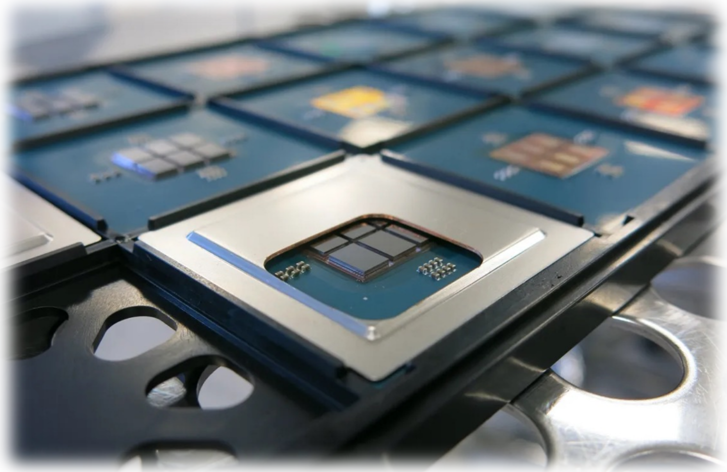
Integrated Systems

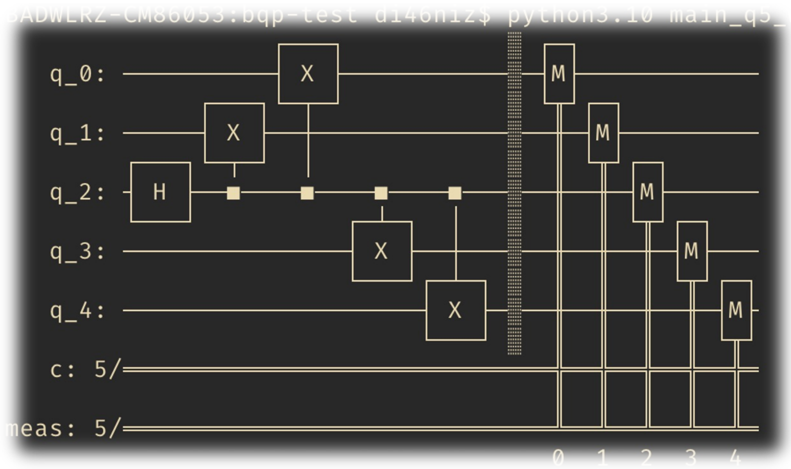
- New opportunities
  - Quick load shifts
  - Power shifts
- Need for node sharing
- Impact on scheduling

Quantum Paradigm

- Again disruptive
- For now: circuit-based
- New software stack needed
- Key: HPC integration
- Impact on scheduling







```
ADWLRZ-CM86053:bqp-test di46niz$ python3.10 main_q5_
```

How to Get to One Software Stack?

Programming

- Multi-accelerator programming is hard and manual
- Abstractions can help
- Single source option / for quantum?

System software is equally critical

- Scheduling and resource management
- Efficient and uniform device usage

We must revisit old HPC dogmas!

Consequences on the software side

Breaking HPC Dogmas

One Node
=
On Job

On-Node Co-Scheduling

Effective use
of complementary
accelerators

Mitigation of Inefficient GPU Utilization

Issues with GPU utilization

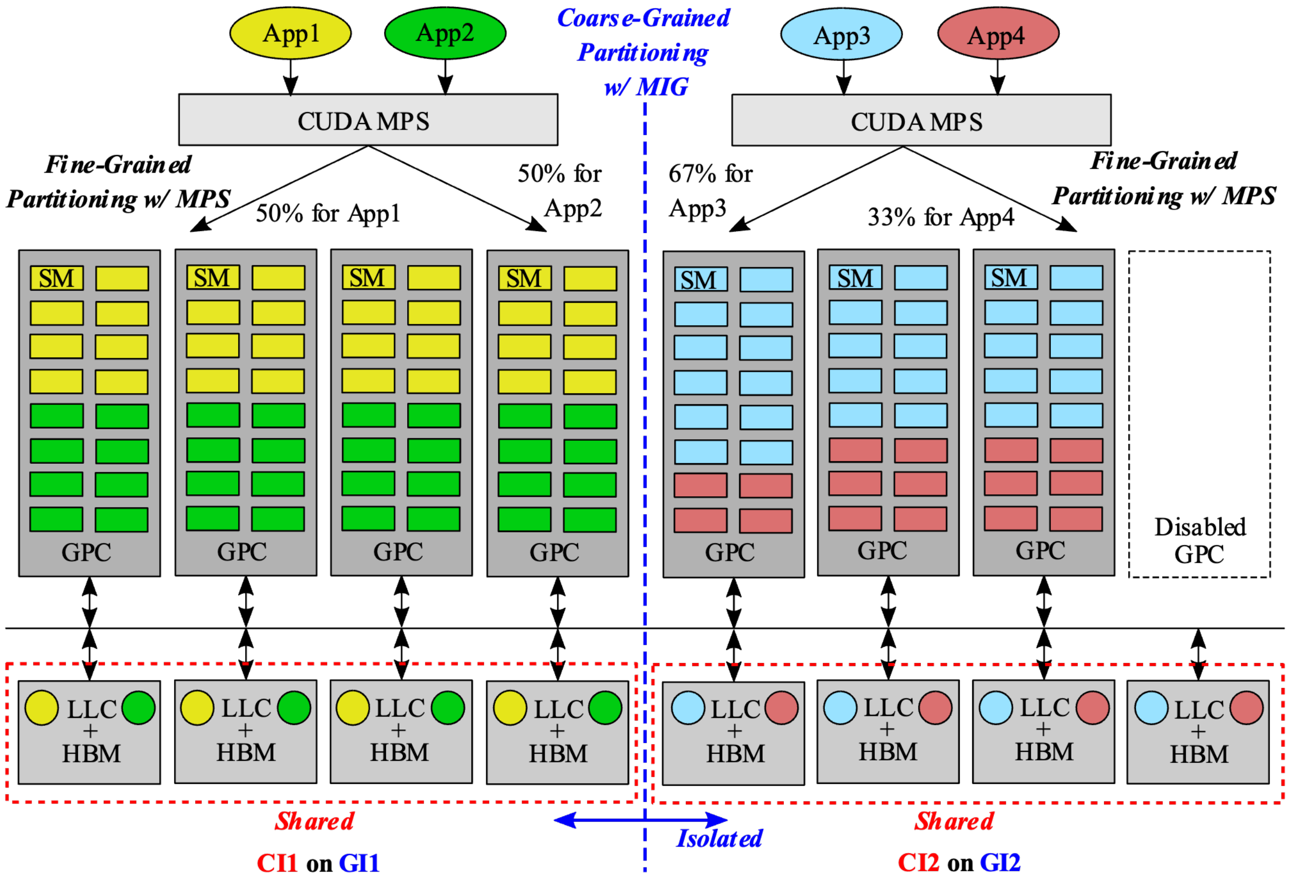
- Not all workloads use the entire GPU
- Multiple processes per node

Co-scheduling as an option

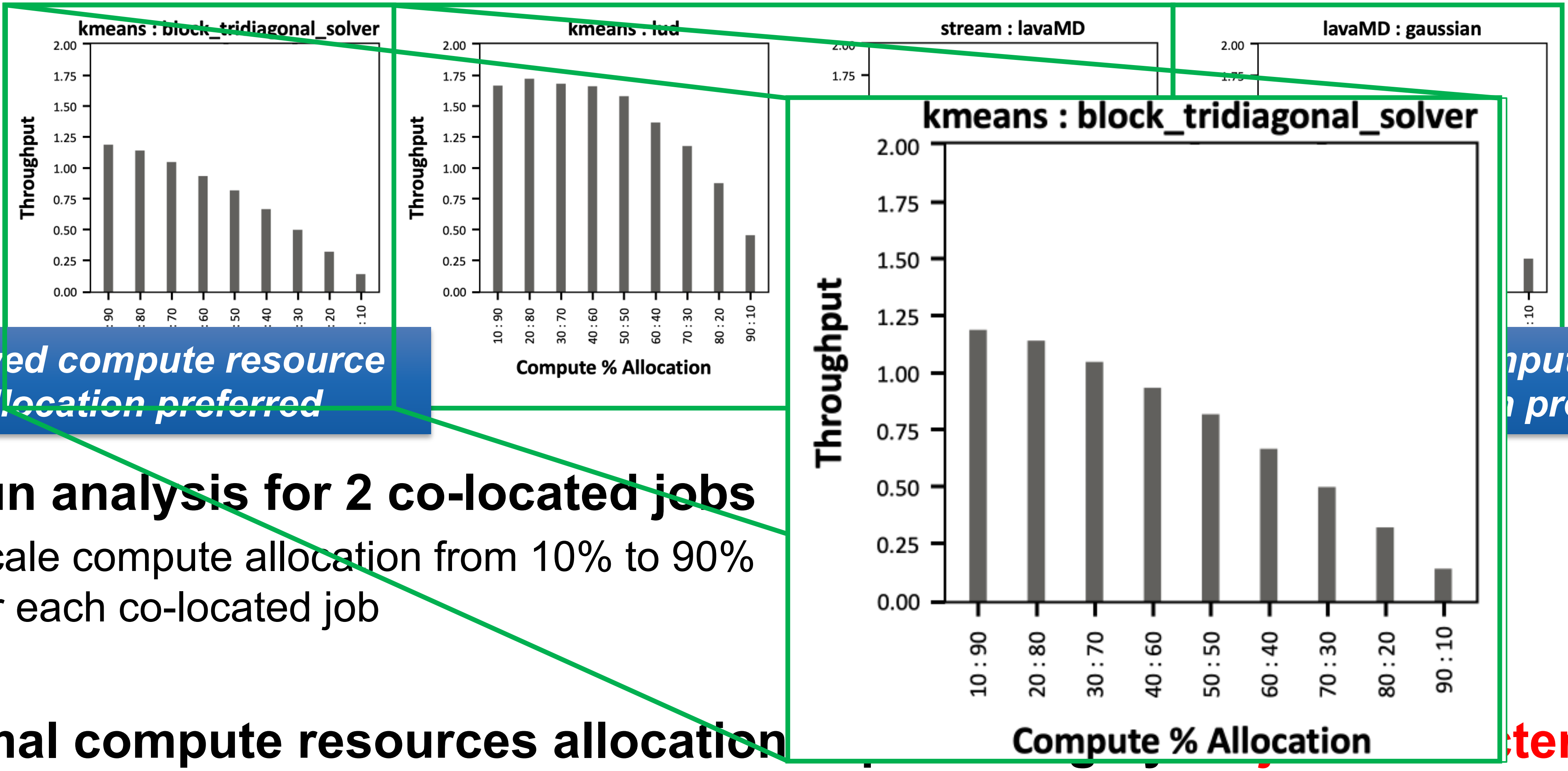
- Multiple applications share the node
- … share the GPU

Example: NVIDIA features

- MIG (Multi-Instance GPU)
- MPS (Multi-Process Service)



Benefit of Flexible Partitioning by MPS



Skewed compute resource allocation preferred

pute resource … preferred

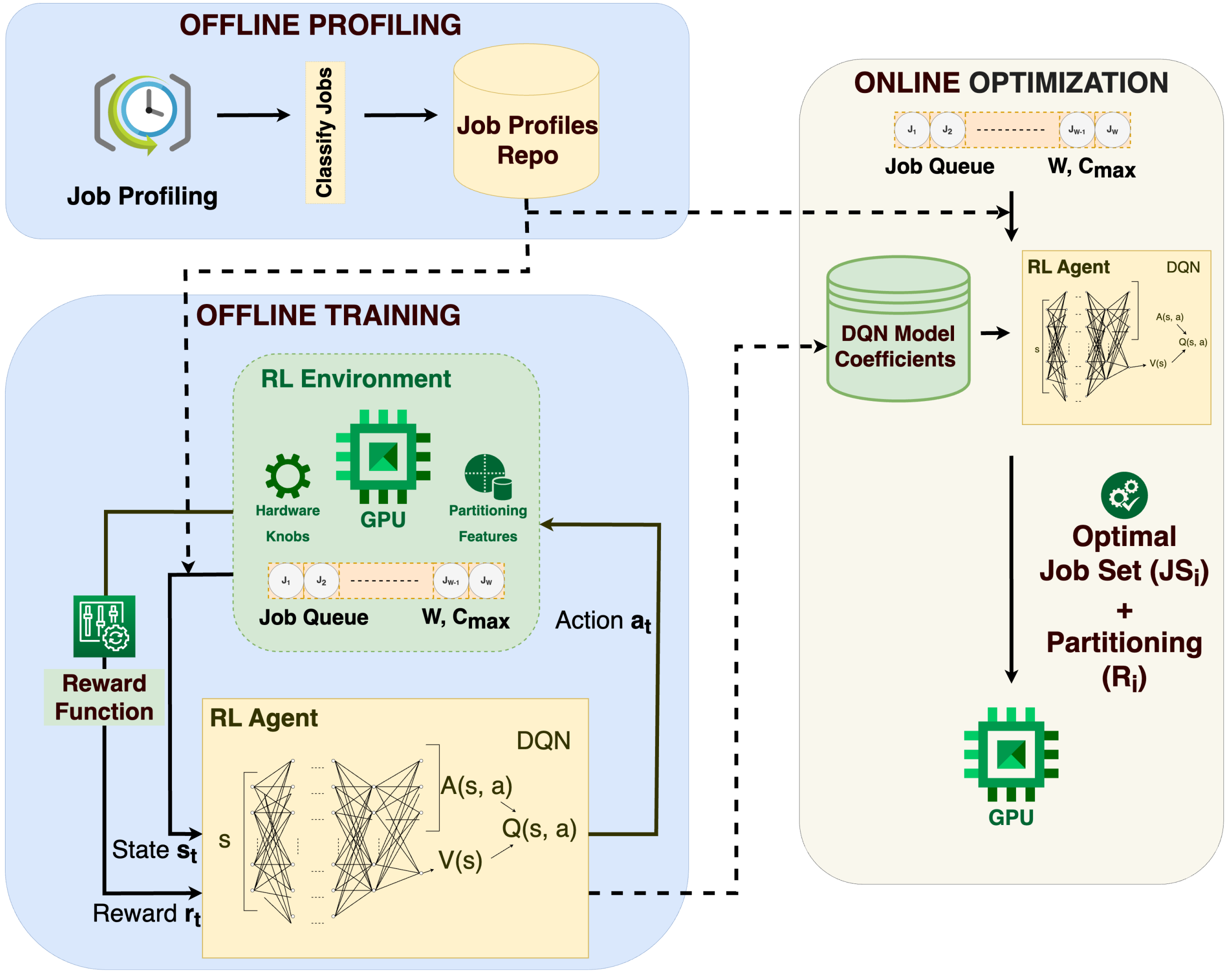
Co-run analysis for 2 co-located jobs

- Scale compute allocation from 10% to 90% for each co-located job

Optimal compute resources allocation … teristics

On-Node Co-Scheduling

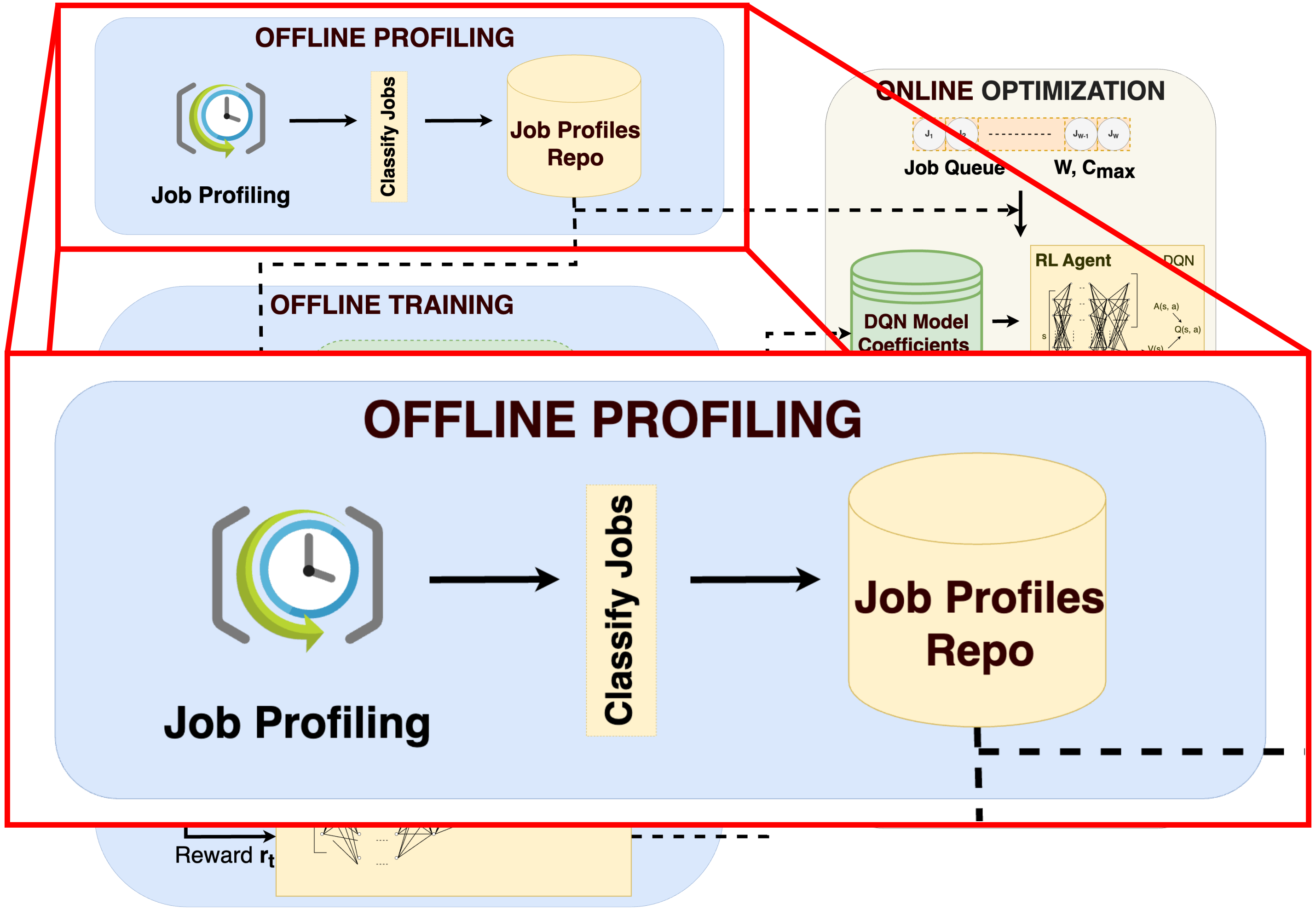
Scheduling



Scheduling

Stage 1: Profiling

- Classify jobs into categories:
  - Compute-Intensive
  - Memory-Intensive
  - UnScalable



On-Node Co-Scheduling

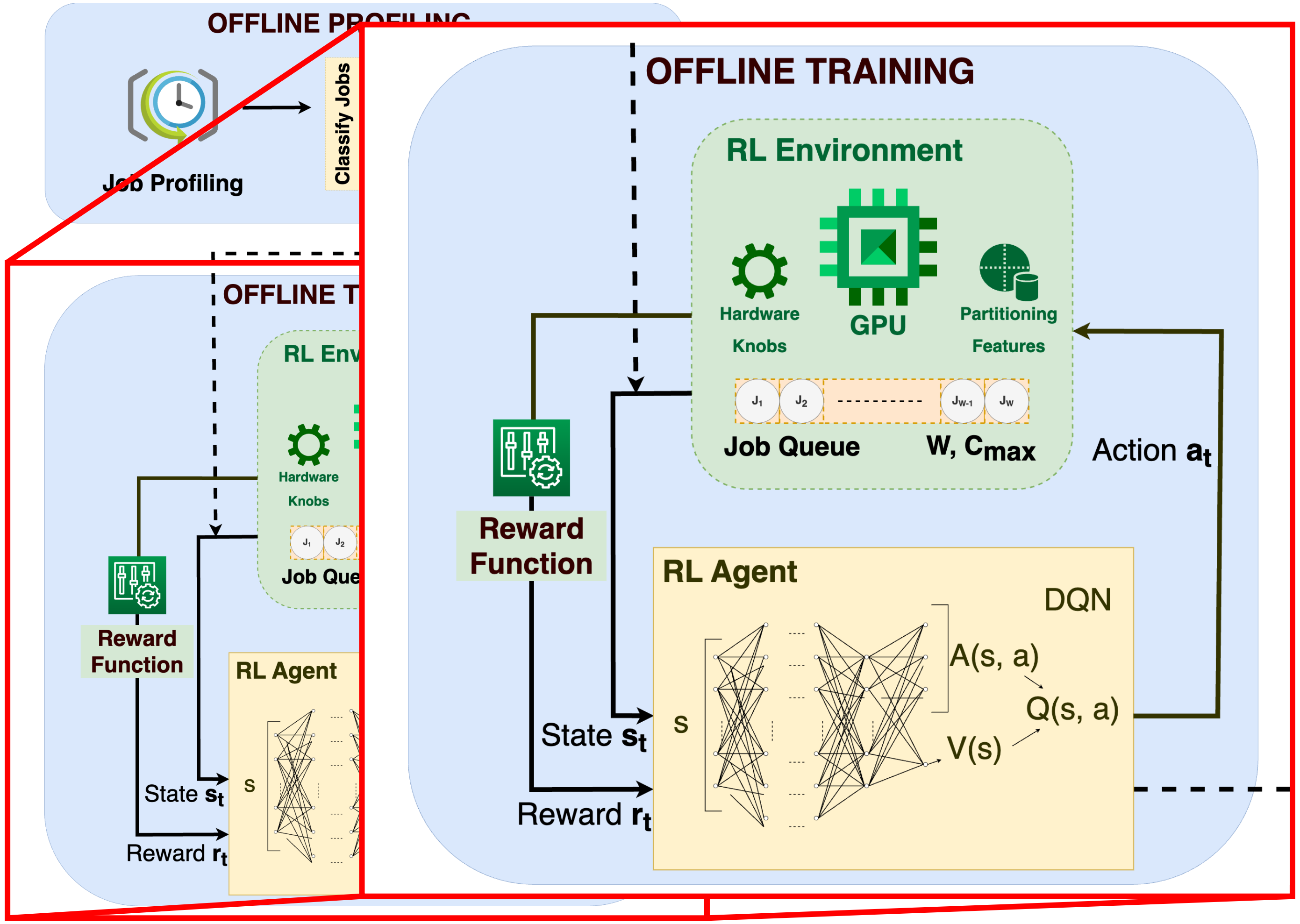
Scheduling

Stage 1: Profiling

- Classify jobs into categories:
    - Compute-Intensive
    - Memory-Intensive
    - UnScalable

Stage 2: Offline Training

- Sweep over partitioning options
    - Explore parameter space
    - Train model
    - Store for online use



On-Node Co-Scheduling

Scheduling

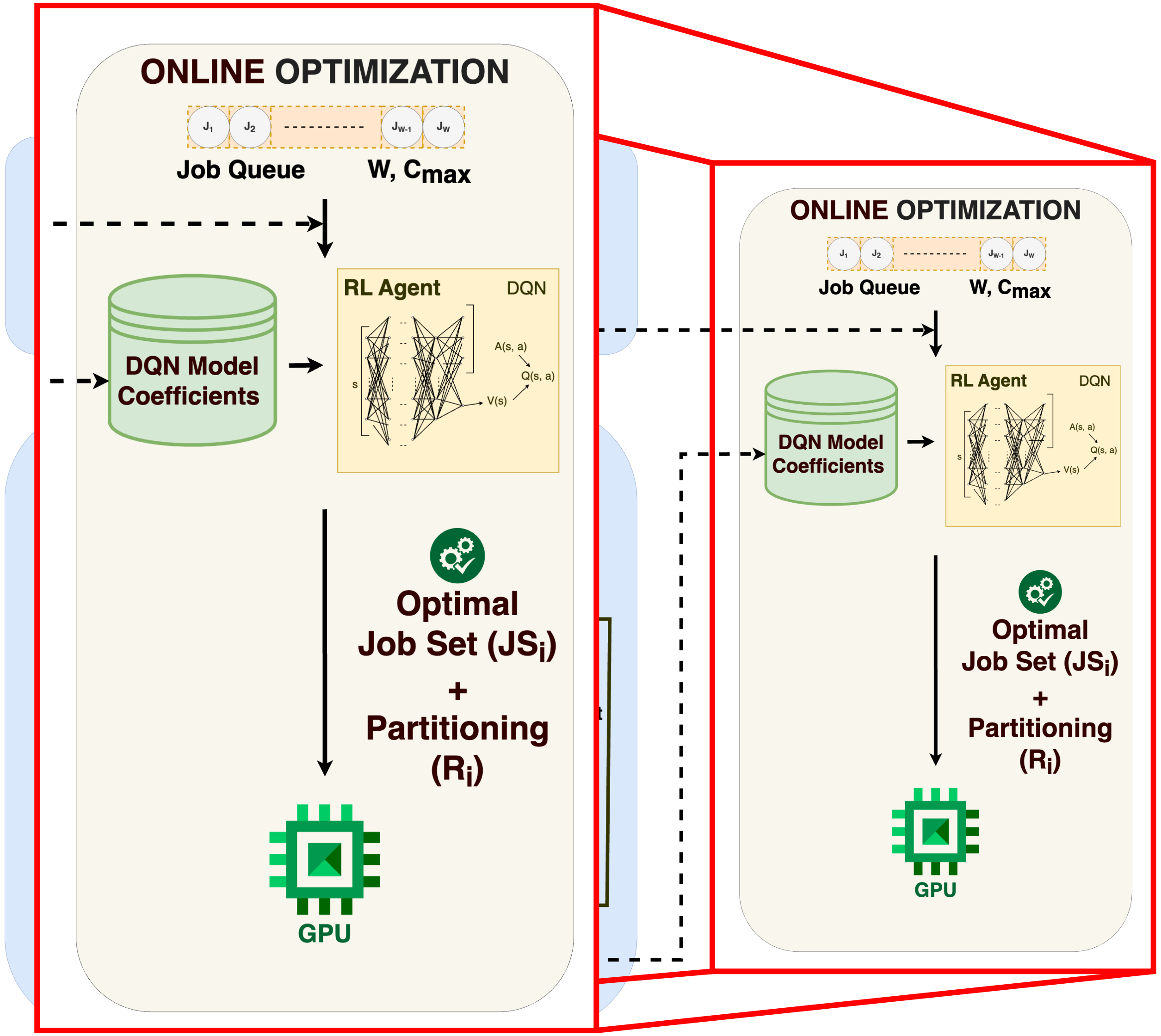
Stage 1: Profiling

- Classify jobs into categories:
  - Compute-Intensive
  - Memory-Intensive
  - UnScalable

Stage 2: Offline Training

- Sweep over policies
  - Explore parameter space
  - Train model
  - Store for online use

Stage 3: Query job assignments



Scheduling

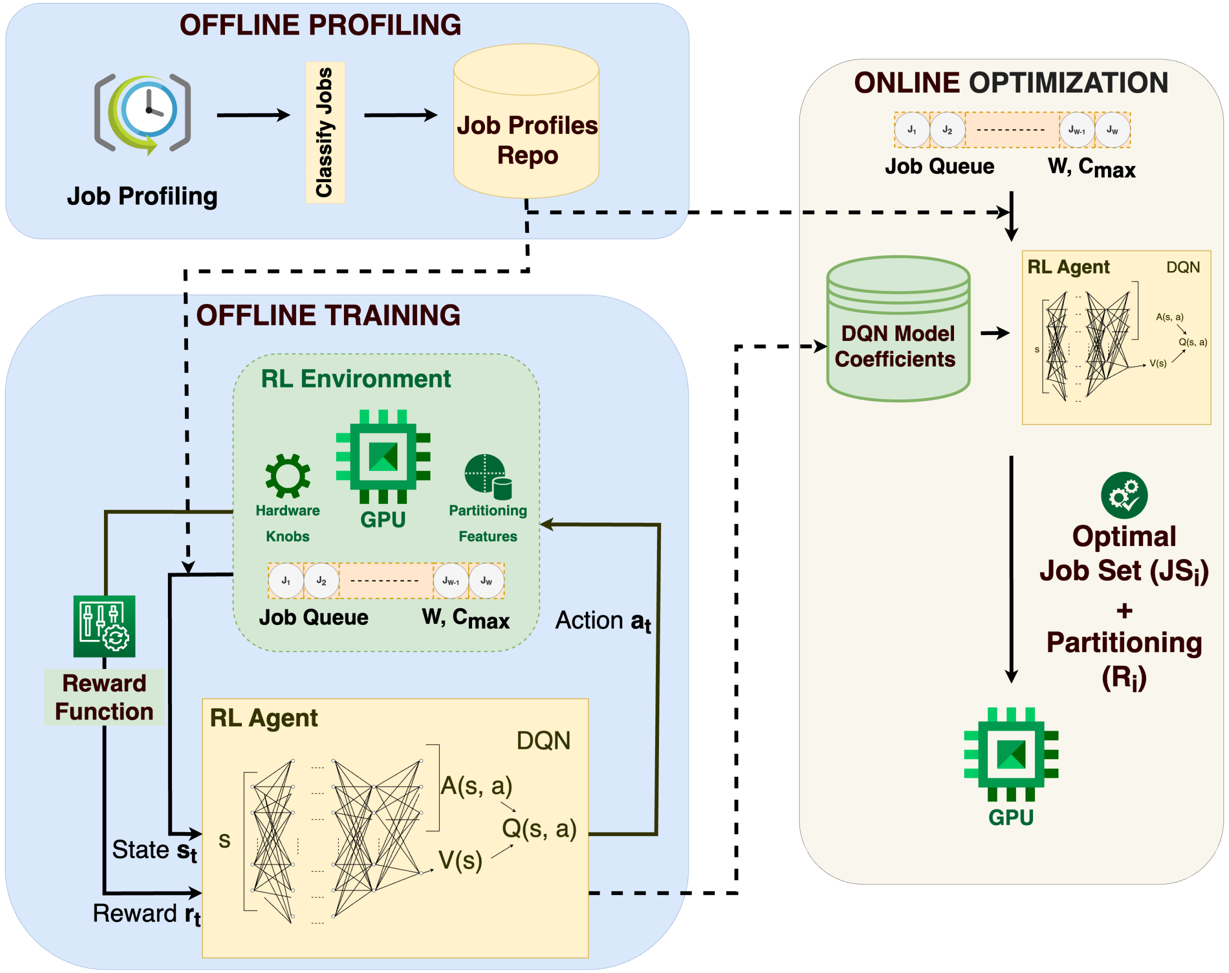
Stage 1: Profiling

- Classify jobs into categories:
  - Compute-Intensive
  - Memory-Intensive
  - UnScalable

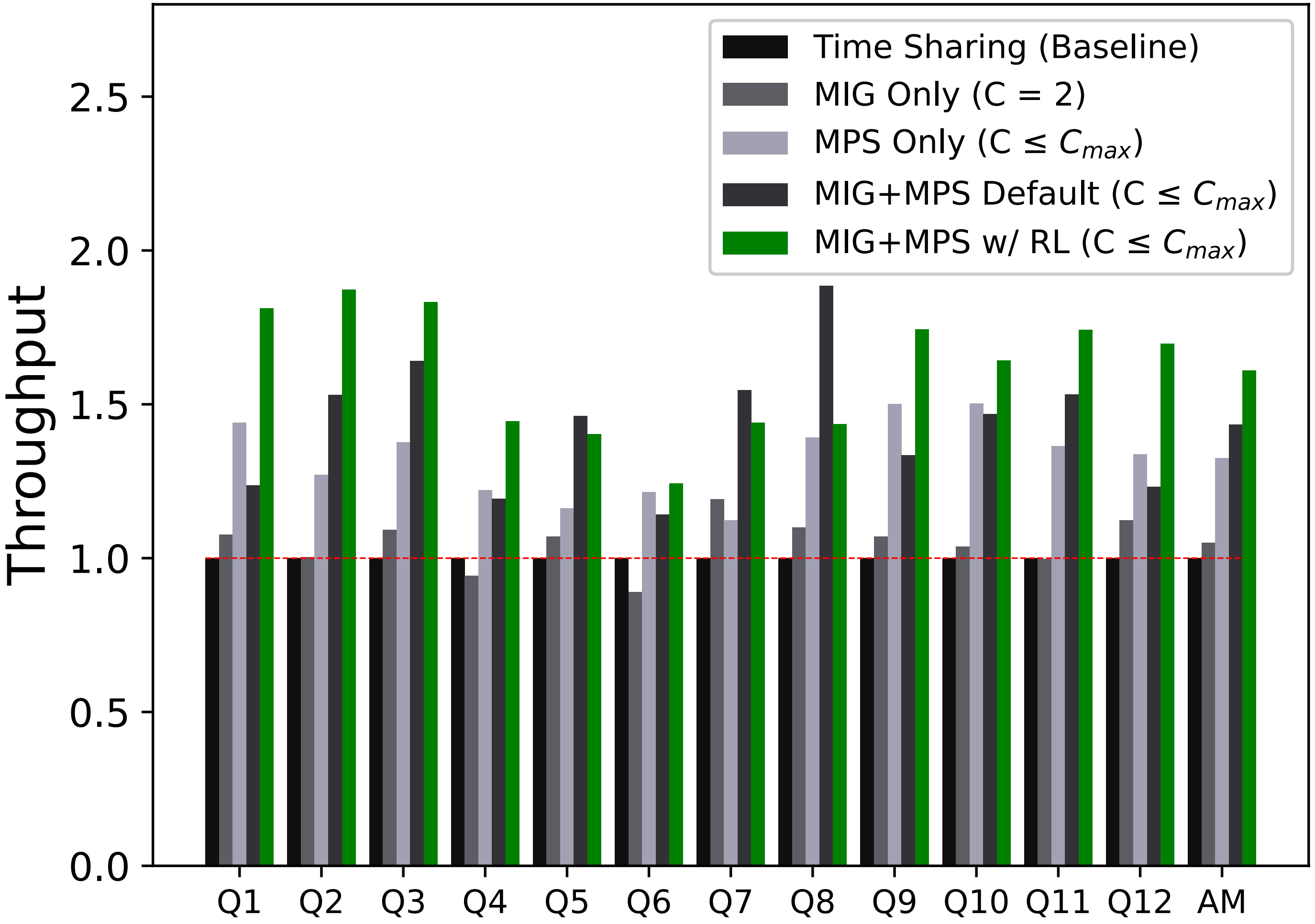
Stage 2: Offline Training

- Sweep over policies
  - Explore parameter space
  - Train model
  - Store for online use

Stage 3: Query job assignments



Experimental Results: Throughput



Name	Remarks
GPU	NVIDIA A100 40GB PCIe 250W TDP
Operating System	Ubuntu 20.04.4 LTS, Kernel Version: 5.4.0-137-generic
Software	CUDA Version: 11.6, Driver Version: 510.108.03, Python Version: 2.7.18

Achieves improvement of

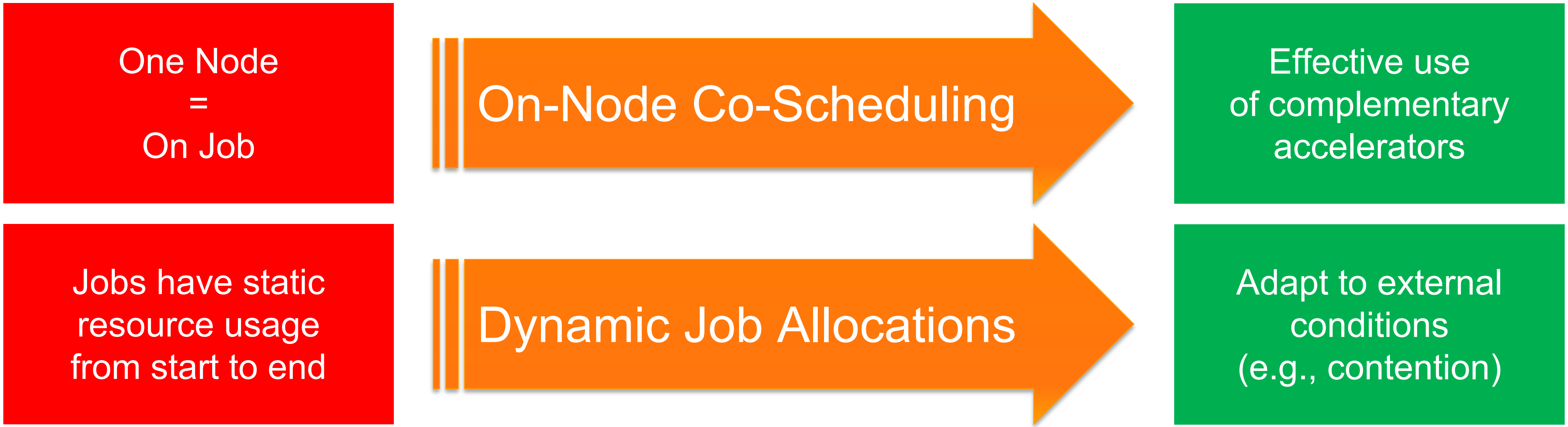
x 1.516 on average

x 1.873 max

Hierarchical Resource Partitioning on Modern GPUs: A Reinforcement Learning Approach,
Urvij Saroliya, Eishi Arima, Dai Liu, Martin Schulz, Stepan Vanecek, Martin Schulz, IEEE Cluster 2023

Consequences on the software side

Breaking HPC Dogmas



One Node = On Job

On-Node Co-Scheduling

Effective use of complementary accelerators

Jobs have static resource usage from start to end

Dynamic Job Allocations

Adapt to external conditions (e.g., contention)

Malleable Programming Models

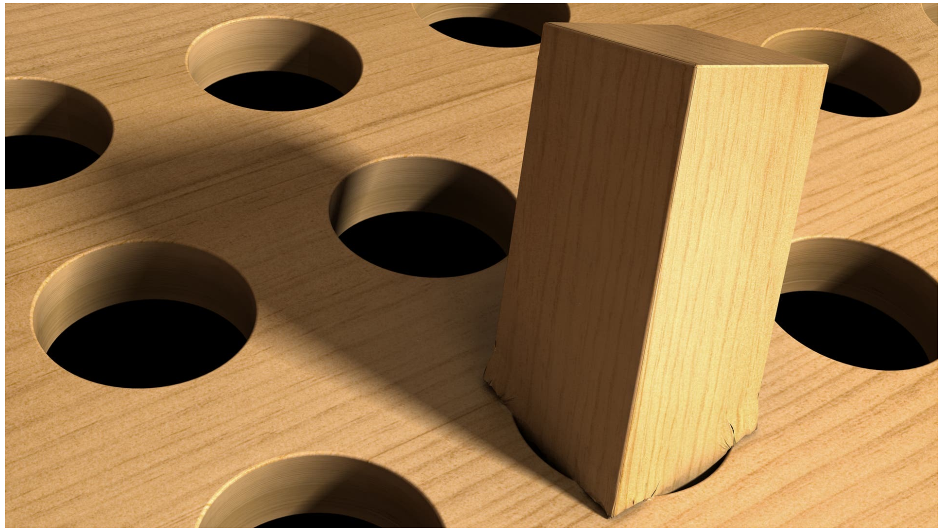
EU Grant #955606
BMBF #16HPC014

Applications need to support malleability

- Overdecomposition/Virtualization
- Via dedicated programming abstractions (e.g., tasking)
- Explicit APIs embedded within the programming model

Most used HPC programming model/abstraction: MPI, the Message Passing Interface

- Static view on resources (MPI_COMM_WORLD)
- Moldability present in standard, but not in practice
- Dynamic adaptation impractical and rarely used



How to change MPI to support malleability

- Maintain basic “look & feel” of MPI
- MPI Process Sets and MPI Sessions

Building on top of MPI Sessions

Option 1: MPI Session form ”MPI Bubbles”

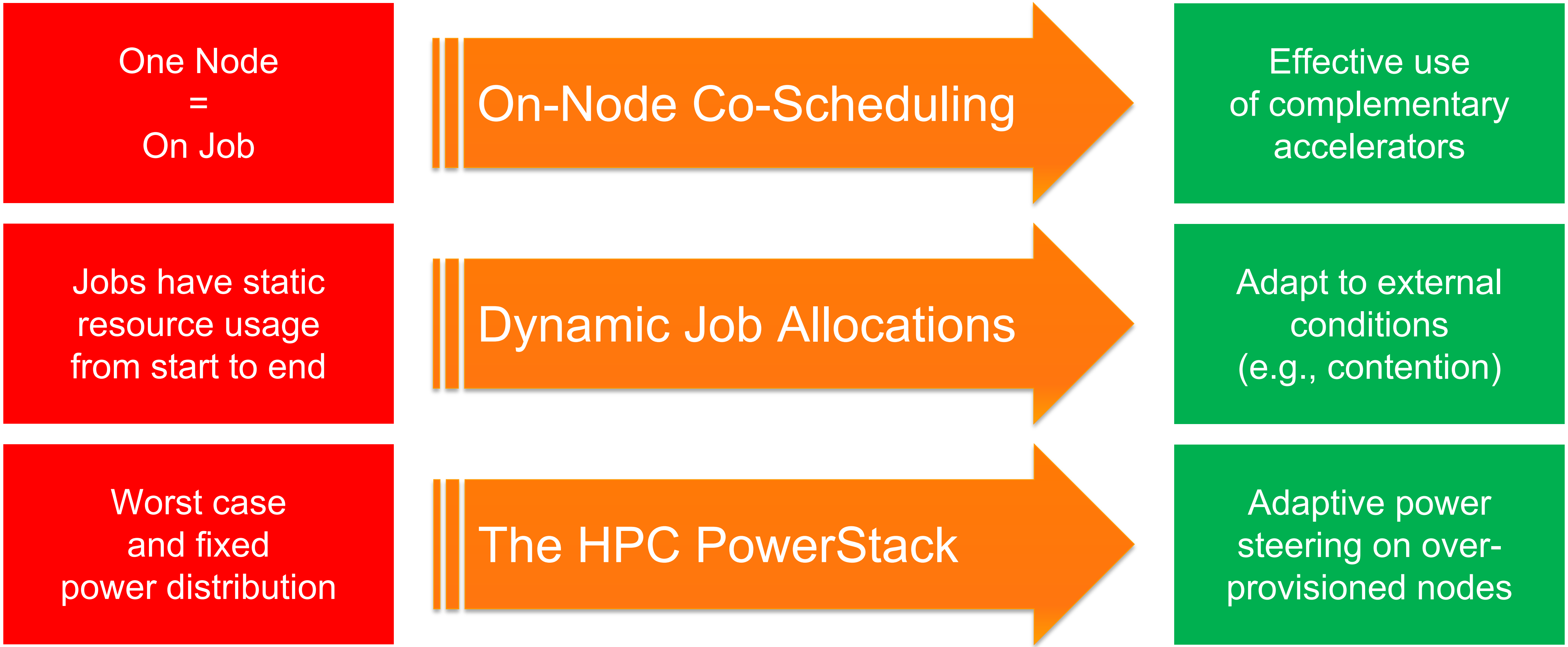
- “All resources that are derived from a set of resources across a set of MPI processes”
- Implicitly derived from MPI application using sessions
- Within an MPI bubble, normal MPI
- Can invalidate and recreate new bubble, while maintaining state

Option 2: Process sets can change

- Ability for the runtime to “tell” something to the application
- Enable process sets ot grow or shrink
- Names are local to MPI Sessions
- Agreement protocol/versioning to agree on new set

Consequences on the software side

Breaking HPC Dogmas



One Node
=
On Job

On-Node Co-Scheduling

Effective use
of complementary
accelerators

Jobs have static
resource usage
from start to end

Dynamic Job Allocations

Adapt to external
conditions
(e.g., contention)

Worst case
and fixed
power distribution

The HPC PowerStack

Adaptive power
steering on over-provisioned nodes

Active Power Steering

The PowerStack Initiative

Active international initiative to identify

… Common terminology

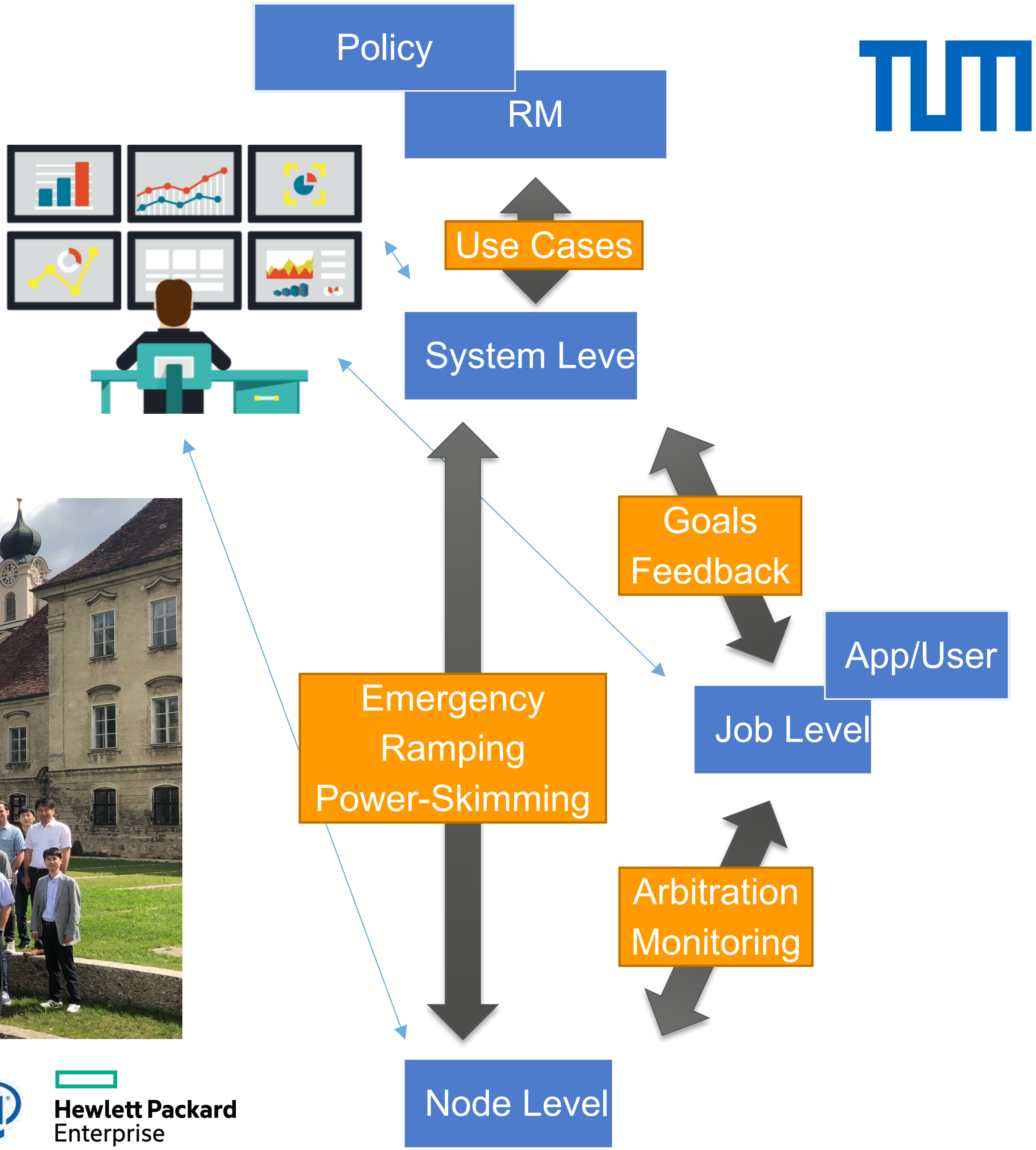
… Compatible components

… Site-specific policies

Starting Point:
June 2018 Seminar at the
TUM Science & Study Center
Raitenhaslach

Multiple meetings
since then

Sessions at major conferences



TECHNISCHE UNIVERSITÄT MÜNCHEN · Lawrence Livermore National Laboratory · 東京大学 THE UNIVERSITY OF TOKYO · intel · Hewlett Packard Enterprise

https://powerstack.caps.in.tum.de/

Active Power Steering

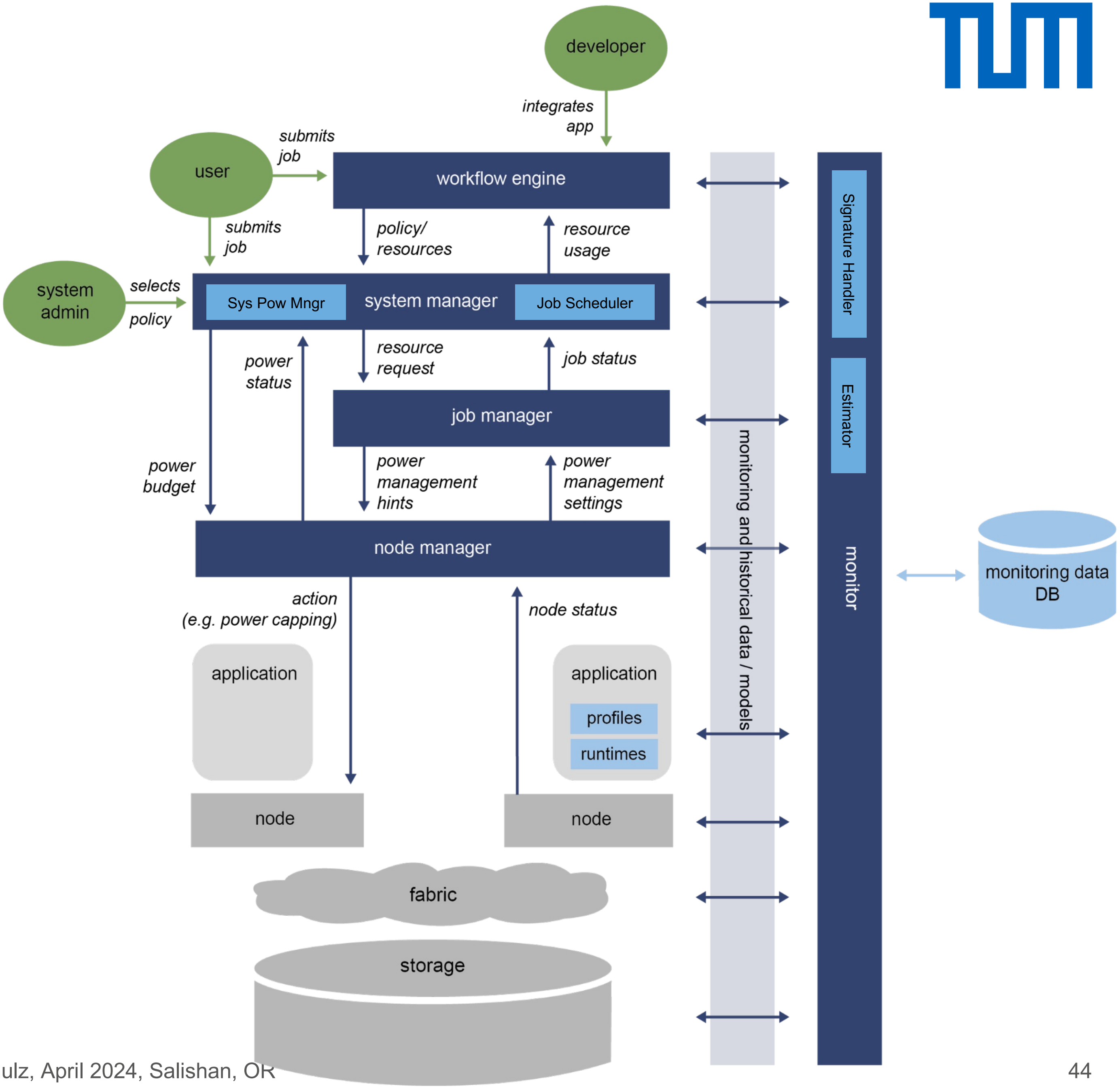
REGALE Architecture

Three levels of management

- System-level
- Job-level
- Node-level

Integration of existing software

- Elastic MPI
- Standardization (PMIx)
- Workflow systems



EU Grant #956560
BMBF #16HPC039K
REGALE

Active Power Steering

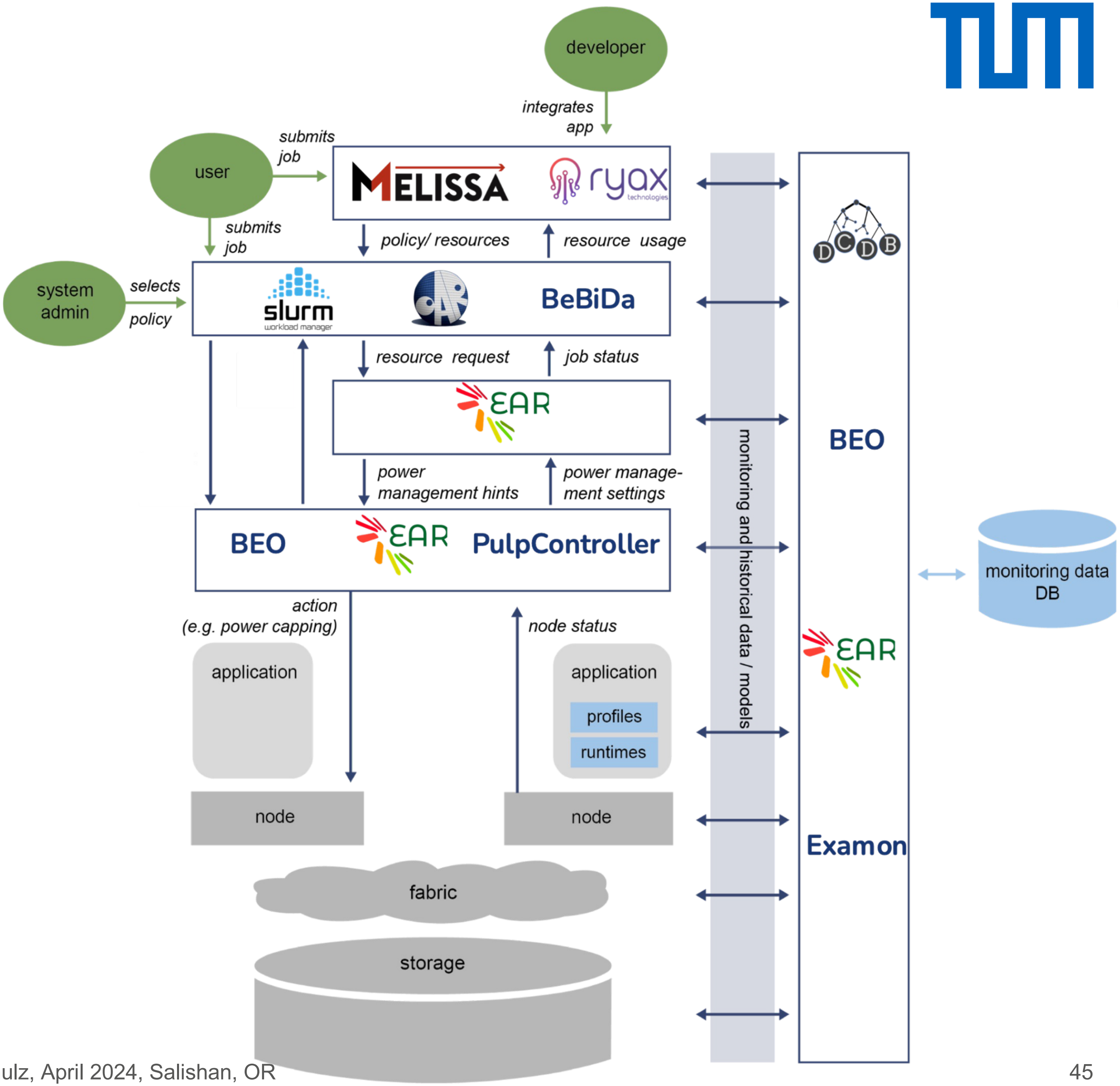
REGALE Architecture

Three levels of management

- System-level
- Job-level
- Node-level

Integration of existing software

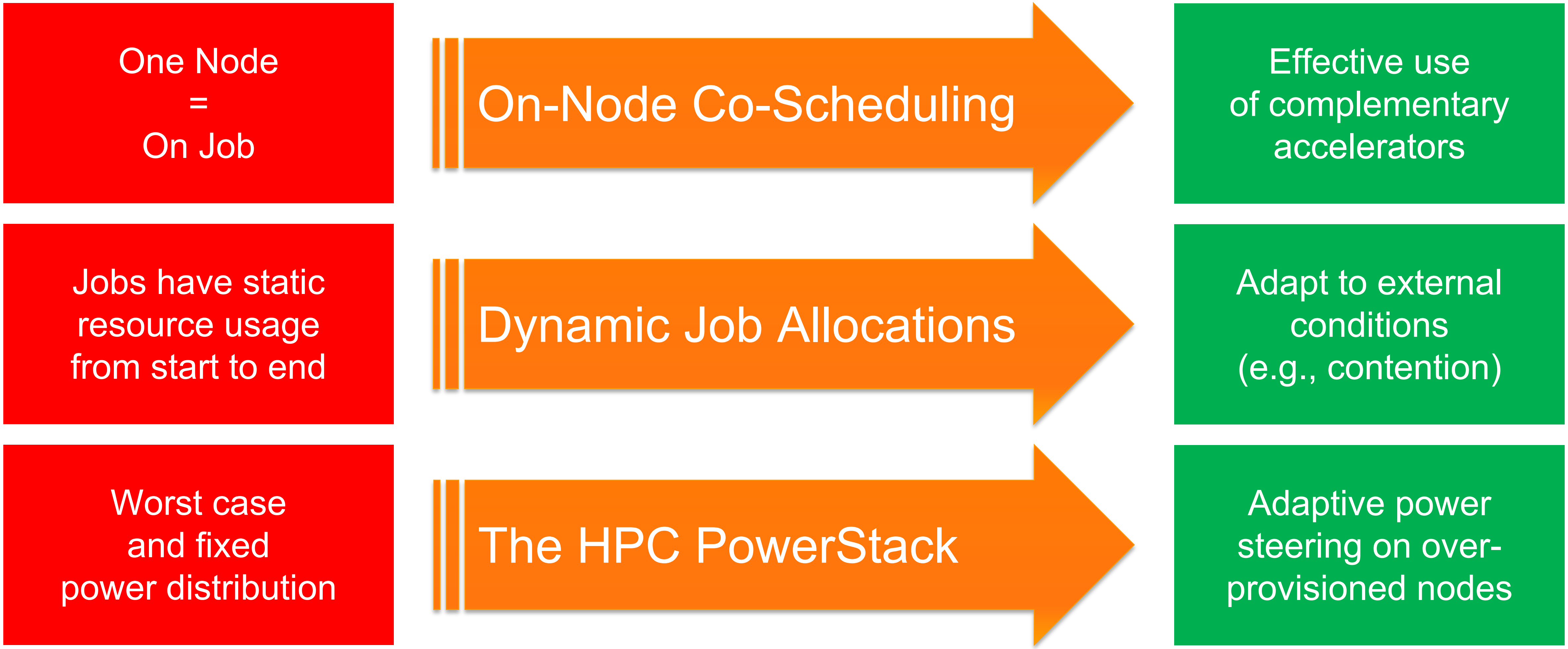
- Elastic MPI
- Standardization (PMIx)
- Workflow systems



EU Grant #956560
BMBF #16HPC039K
REGALE

Consequences on the software side

Breaking HPC Dogmas



One Node = On Job → **On-Node Co-Scheduling** → Effective use of complementary accelerators

Jobs have static resource usage from start to end → **Dynamic Job Allocations** → Adapt to external conditions (e.g., contention)

Worst case and fixed power distribution → **The HPC PowerStack** → Adaptive power steering on over-provisioned nodes

We Will See Specialized/Customized Hardware, But How Can We Use if Efficiently?

Heterogeneous Architectures

- More specialized “crazy” architectures
- Integrated processors with accelerators
- Tight integration of large-scale accelerators

Software Stacks

- Programming models aside …
- Flexible and dynamic scheduling will be key
- Importance of (fine grained) workflows
- Impact across the entire stack

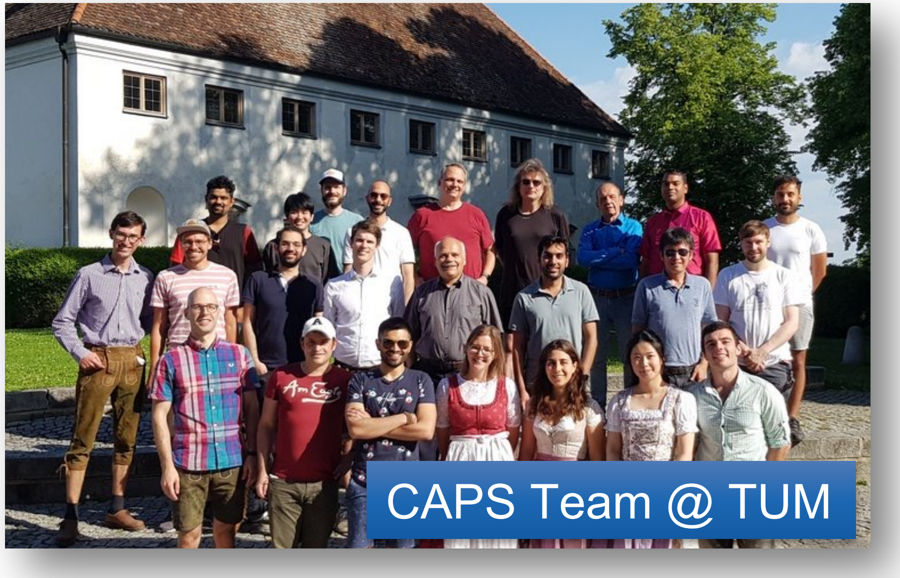
Breaking HPC dogmas

- Single node scheduling → Co-Scheduling
- Rigid resource allocation → Dynamic Resources
- Worst case power → Dynamic Power Shifting

Acknowledgements

It takes a team, or rather many teams!

Munich Quantum Valley


CAPS Team @ TUM

QCT Team @ LRZ

QC@LR

HPCQC

From the MQV Review Meeting 2022

Bayerisches Staatsministerium für
Wissenschaft und Kunst

Federal Ministry
of Education
and Research

Federal Ministry
for Economic Affairs
and Climate Action

European
Commission

EuroHPC
Joint Undertaking

Conclusions

We Will See Specialized/Customized Hardware, But How Can We Use if Efficiently?

Heterogeneous Architectures

- More specialized “crazy” architectures
- Integrated processors with accelerators
- Tight integration of large-scale accelerators

Software Stacks

- Programming models aside …
- Flexible and dynamic scheduling will be key
- Importance of (fine grained) workflows
- Impact across the entire stack

Breaking HPC dogmas

- Single node scheduling → Co-Scheduling
- Rigid resource allocation → Dynamic Resources
- Worst case power → Dynamic Power Shifting

https://www.ce.cit.tum.de/caps/