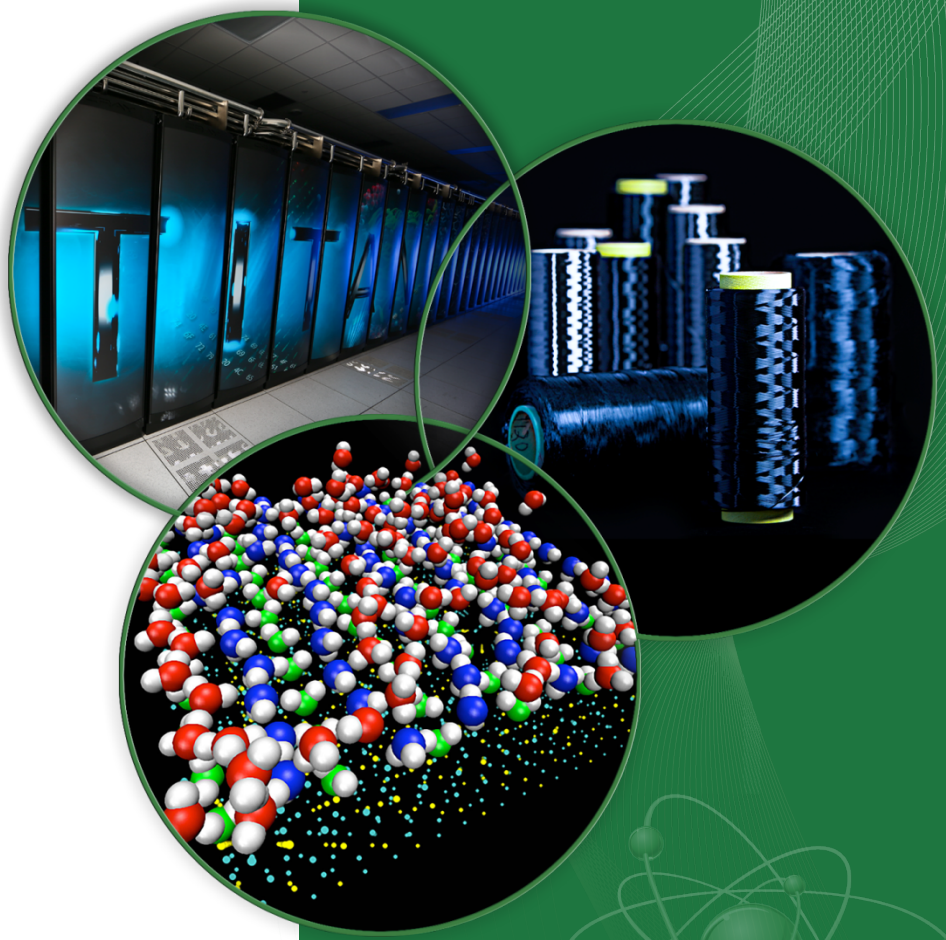
Portable Heterogeneous High-Performance Computing via Domain-Specific Virtualization

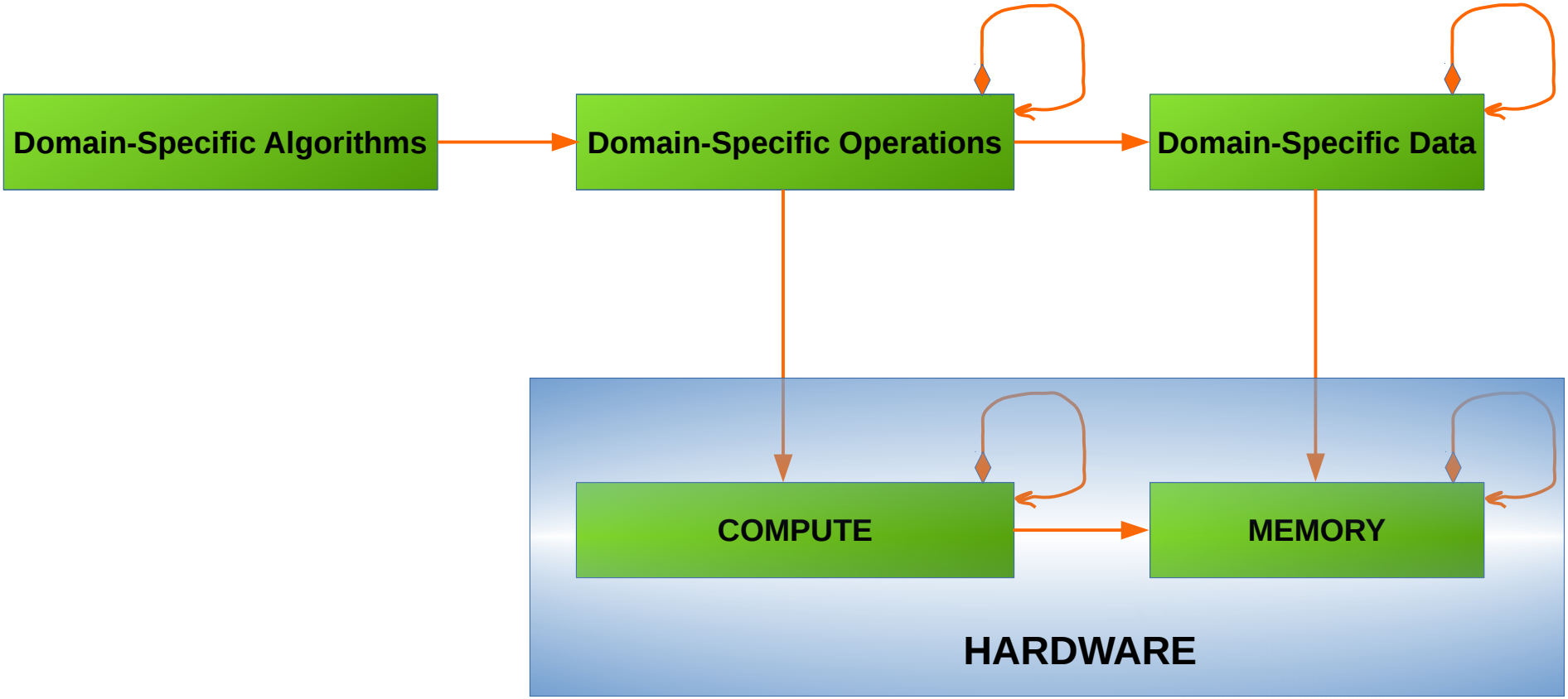


Dmitry I. Lyakh

liakhdi@ornl.gov

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract No. DE-AC05-00OR22725.

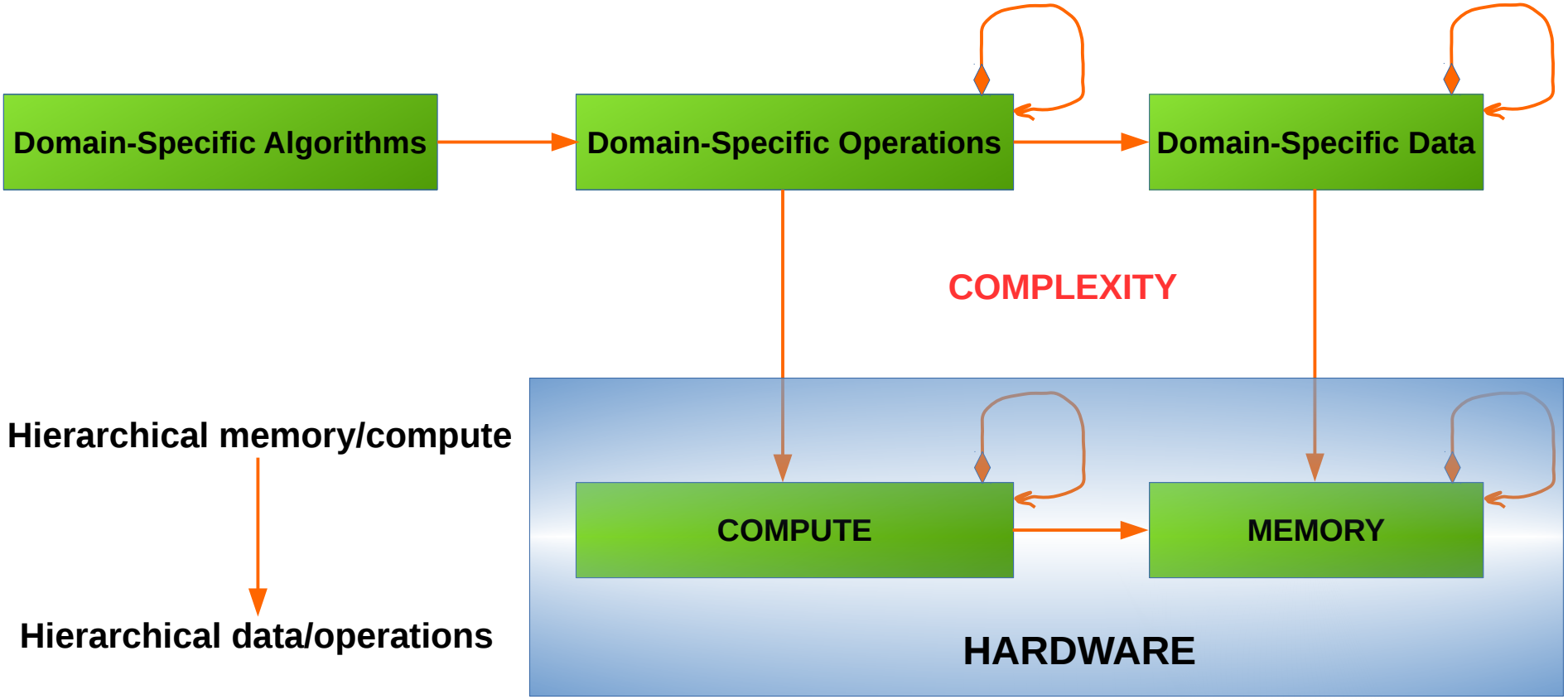
Performance Portability



PORTABILITY: Multiple targets, one code, maybe minor extension (not modification)

PERFORMANCE: Minimization/optimization of data movement to keep compute busy:
Optimal mapping of data and operations

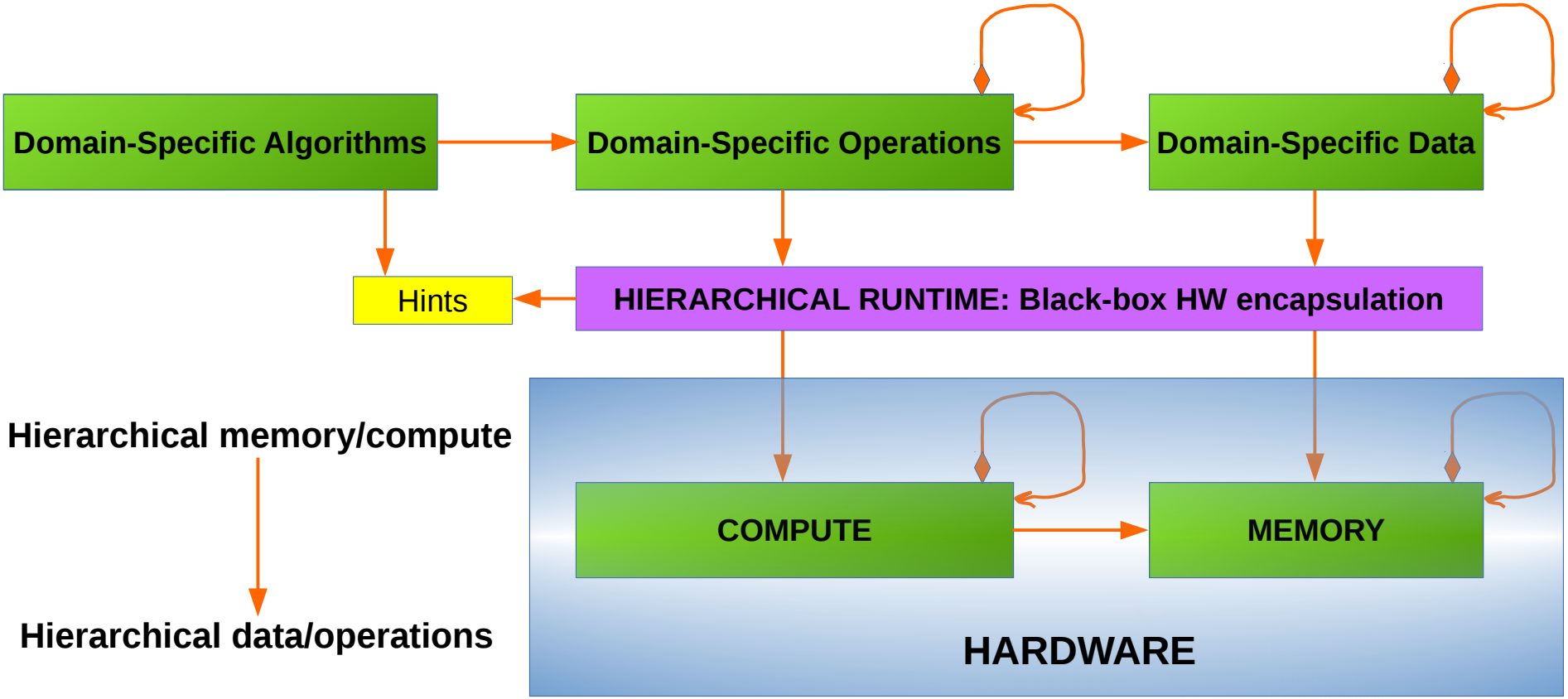
Performance Portability



PORTABILITY: Multiple targets, one code, maybe minor extension (not modification)

PERFORMANCE: Minimization/optimization of data movement to keep compute busy:
Optimal mapping of data and operations

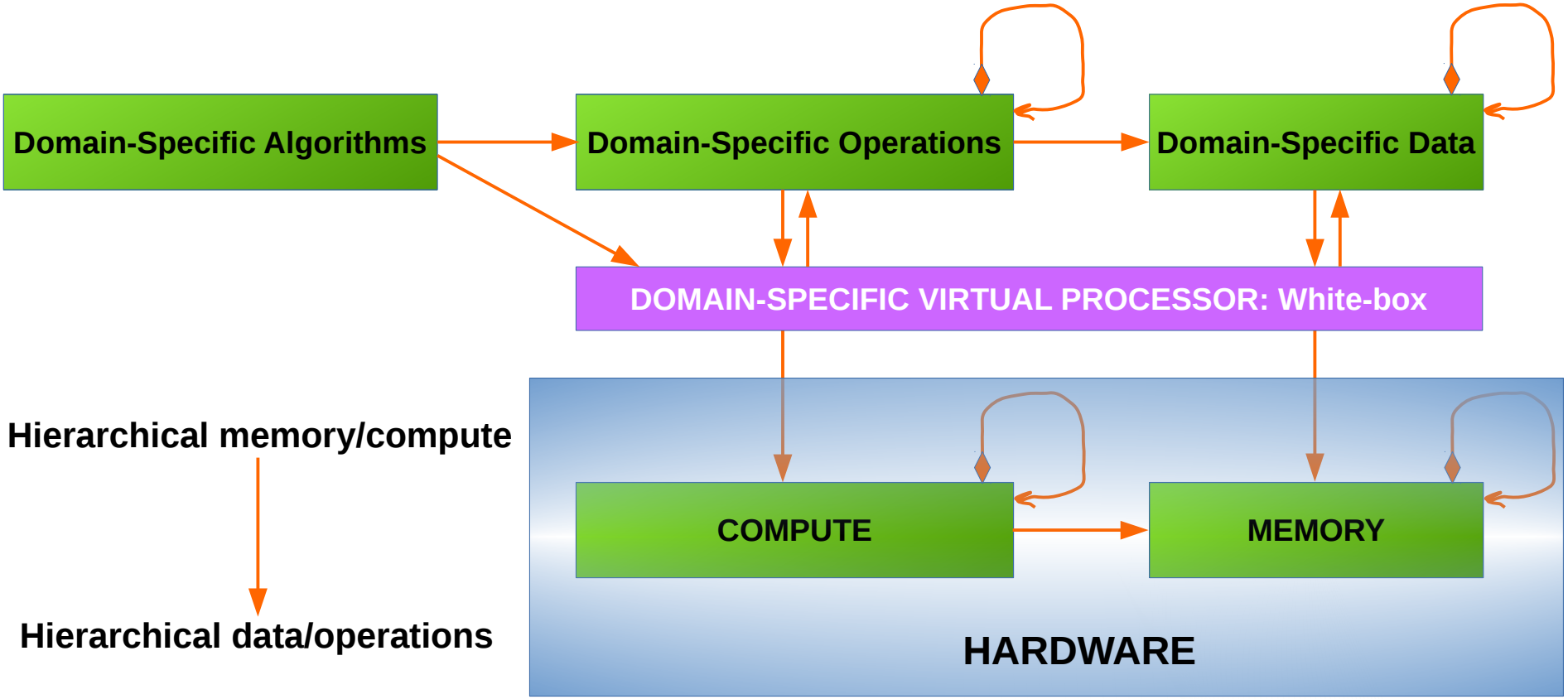
Performance Portability



PORTABILITY: Multiple targets, one code, maybe minor extension (not modification)

PERFORMANCE: Minimization/optimization of data movement to keep compute busy:
Optimal mapping of data and operations

Performance Portability



PORTABILITY: Multiple targets, one code, maybe minor extension (not modification)

PERFORMANCE: Minimization/optimization of data movement to keep compute busy:
Optimal mapping of data and operations

Domain-Specific Virtual Processor

- Generalization, elaboration, and formalization of previous efforts
- Explicitly structured runtime system that formally resembles a **processor**, but specialized to **domain-specific** workloads
- Virtualizes physical hardware by encapsulating it with a **fixed** virtual processing architecture **best suited** for domain algorithms
- Encapsulates a wide class of HPC architectures via a virtual node architecture **template**: Opportunities for **co-design**
- Understands the specificity of domain data, operations and algorithms: Better opportunity for optimization (**performance**)
- Domain algorithms are expressed **once**, either via a standalone or an embedded DSL, then compiled and executed (or interpreted)
- Debugging/profiling in terms of domain-specific abstractions

Relevant Past/Present Efforts

- 2002-2005: CLUSTER: Automated code generation: CAS-CCSD method = Millions of generated SLOC
- 2006-2008: CLUSTER moved to a **direct interpretation** of many-body CC equations: High-level specs → Bytecode
- 2008+: ACES III and ACES IV: Domain-specific super-instruction language and runtime (SIAL/SIP): SIAL (med.level) → SIP bytecode → Interpretation by SIP
- 2014+: ExaTENSOR framework: Direct interpretation of **hierarchical tensor algebra** workloads on heterogeneous HPC architectures with cross-domain applications
- Other domains of scientific computing?

Math Framework: Basic Tensor Algebra

- Formal tensor: $T^{pq\ldots}_{rs\ldots} \mapsto T(p,q,r,s,\ldots)$: n-D Array

  Full tensor: $T(p,q,r,s): p \in P, q \in Q, r \in R, s \in S$

  Tensor slice: $T(p,q,r,s): p \in P' \subseteq P, q \in Q' \subseteq Q,$
  $r \in R' \subseteq R, s \in S' \subseteq S$

Few primitive operations:

- Tensor addition:

$$\forall p,q,r,s: \; T^{pq}_{rs} = L^{pq}_{rs} + R^{pq}_{rs}$$

Parallelism!

- Tensor product:

$$\forall p,q,r,s: \; T^{pq}_{rs} = L^{p}_{r} R^{q}_{s}$$

Parallelism!

- Tensor contraction:

$$\forall p,q,r,s: \; T^{pq}_{rs} = L^{pai}_{bcd} R^{qbcd}_{rsai}$$

Compute intensive (potentially)!

Math Framework: Tensor Decompositions

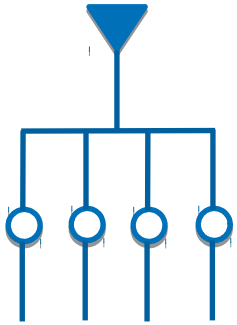
- Graphical (diagrammatic) representation:

Matrix:  Matrix*Matrix: 

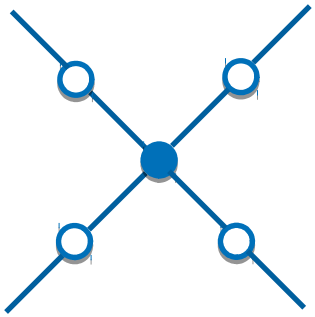
- Linear algebra: SVD is optimal in the 2-norm:

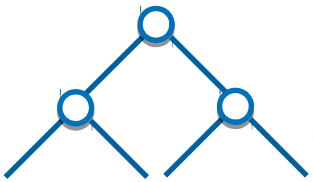
- Tensor (multi-linear) algebra: Many choices:



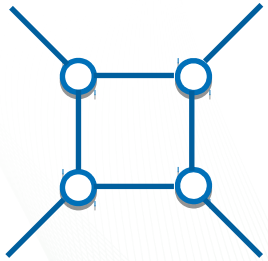
Canonical polyadic



Tucker



Tensor tree

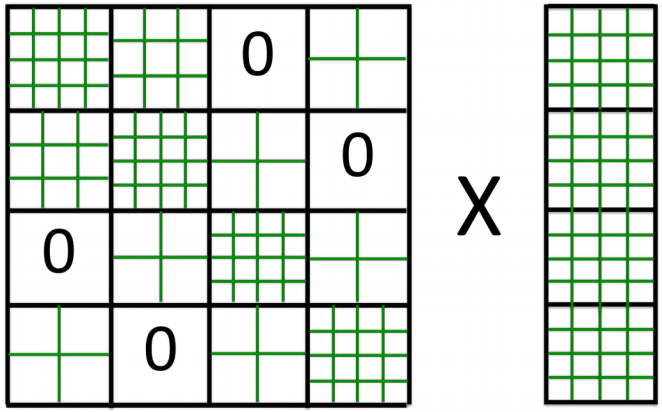


Tensor train

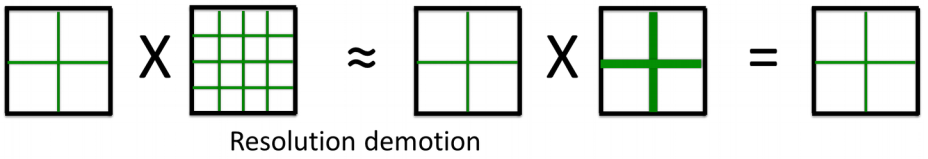
Adaptive (+Hierarchical) Tensor Algebra

Inspired by Multiresolution Analysis

Adaptive dynamic block-sparse representation of many-body tensors

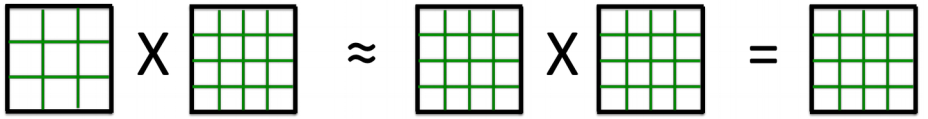


X

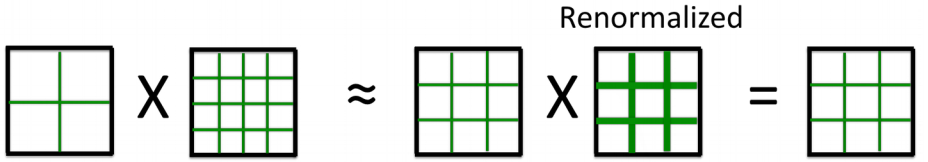


Renormalized

Resolution demotion



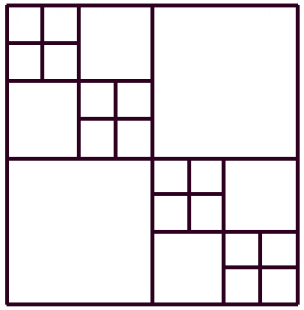
Resolution promotion

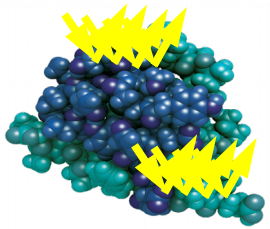


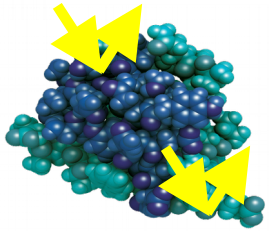
Renormalized

Resolution intermediate match

The resolution of each tensor block can be dynamically adjusted if needed in a specific tensor operation (computational cost reduction)





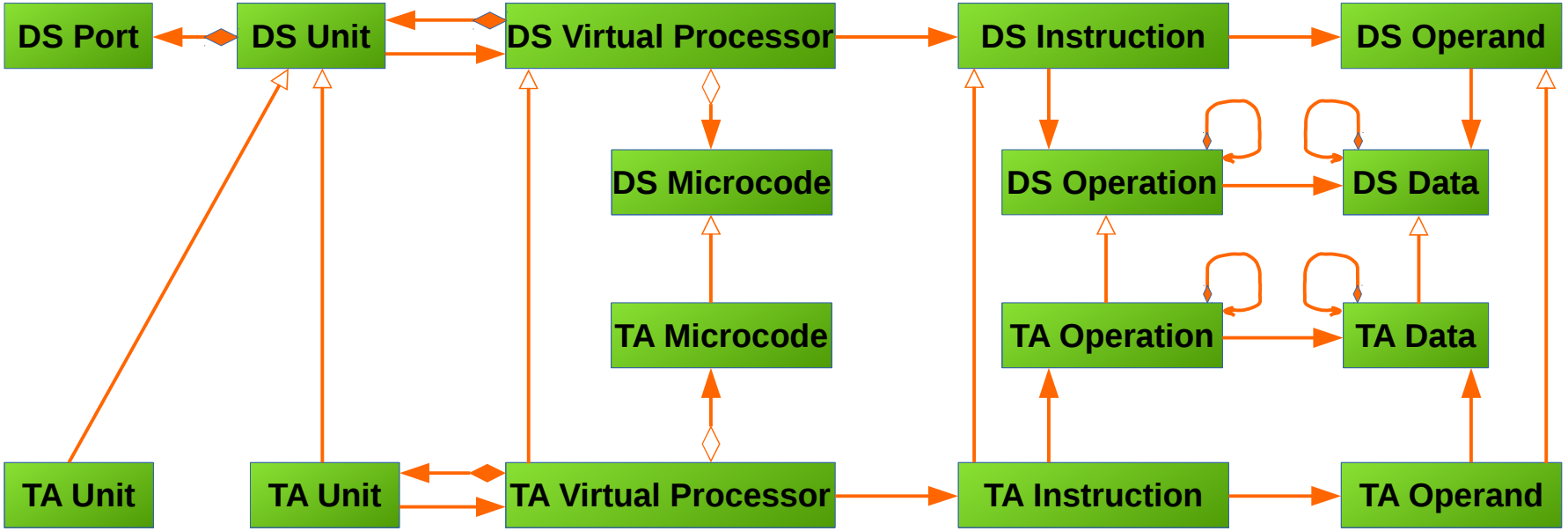


Extrapolation of H/H2-matrix algebra to TENSORS

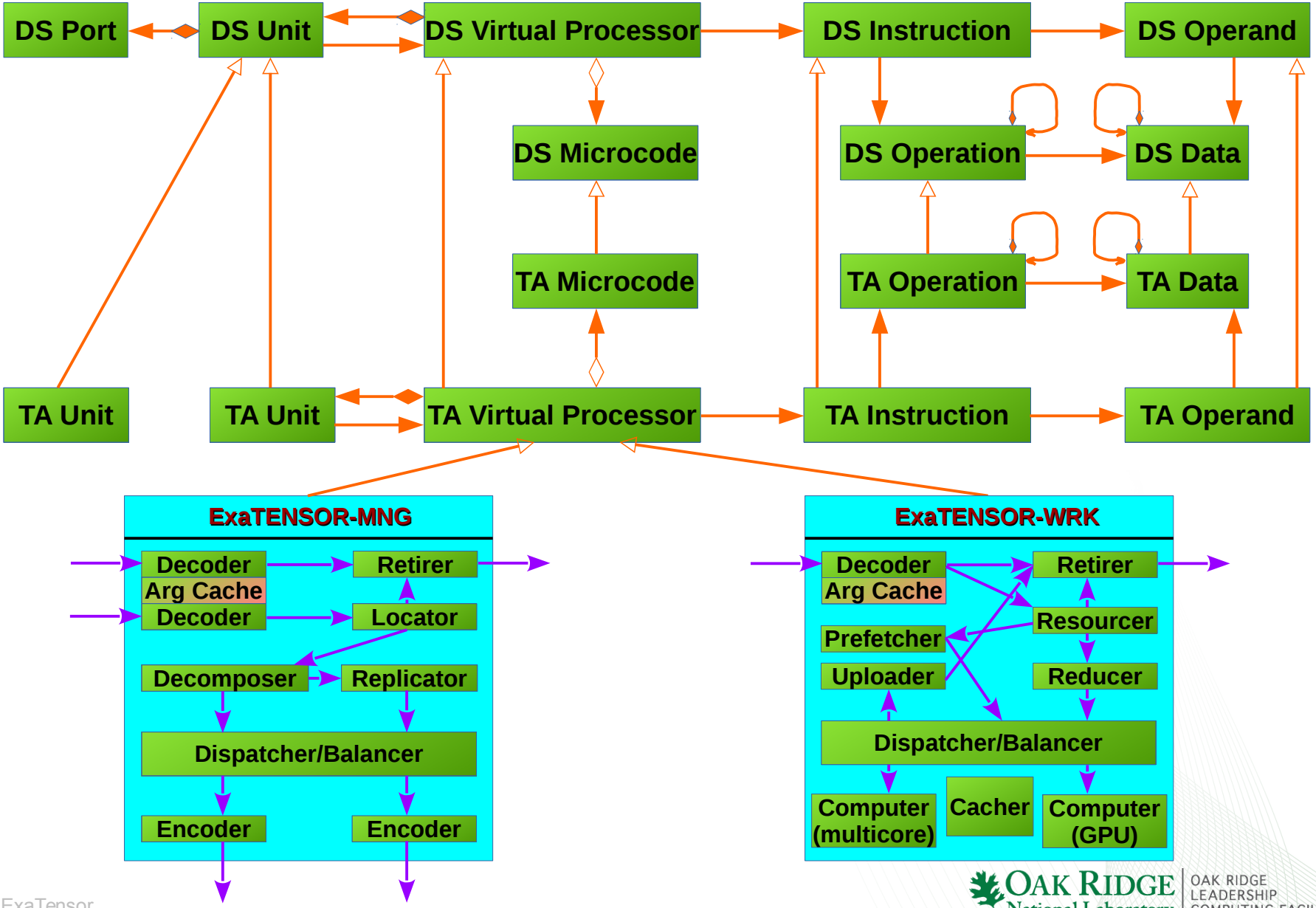
- Large tensor elements can become tensors themselves (higher resolution);
- Weak tensor slices can be compressed by lowering the resolution, up to a single (complex) number;
- Adapt to the calculated electronic state and available HPC resources;
- Should be better than just black-and-white discarding.

D.I.L. *Int. J. Quantum Chem.* 2014; D.I.L. ArXiv 2017

Domain-Specific Virtual Processor Architecture



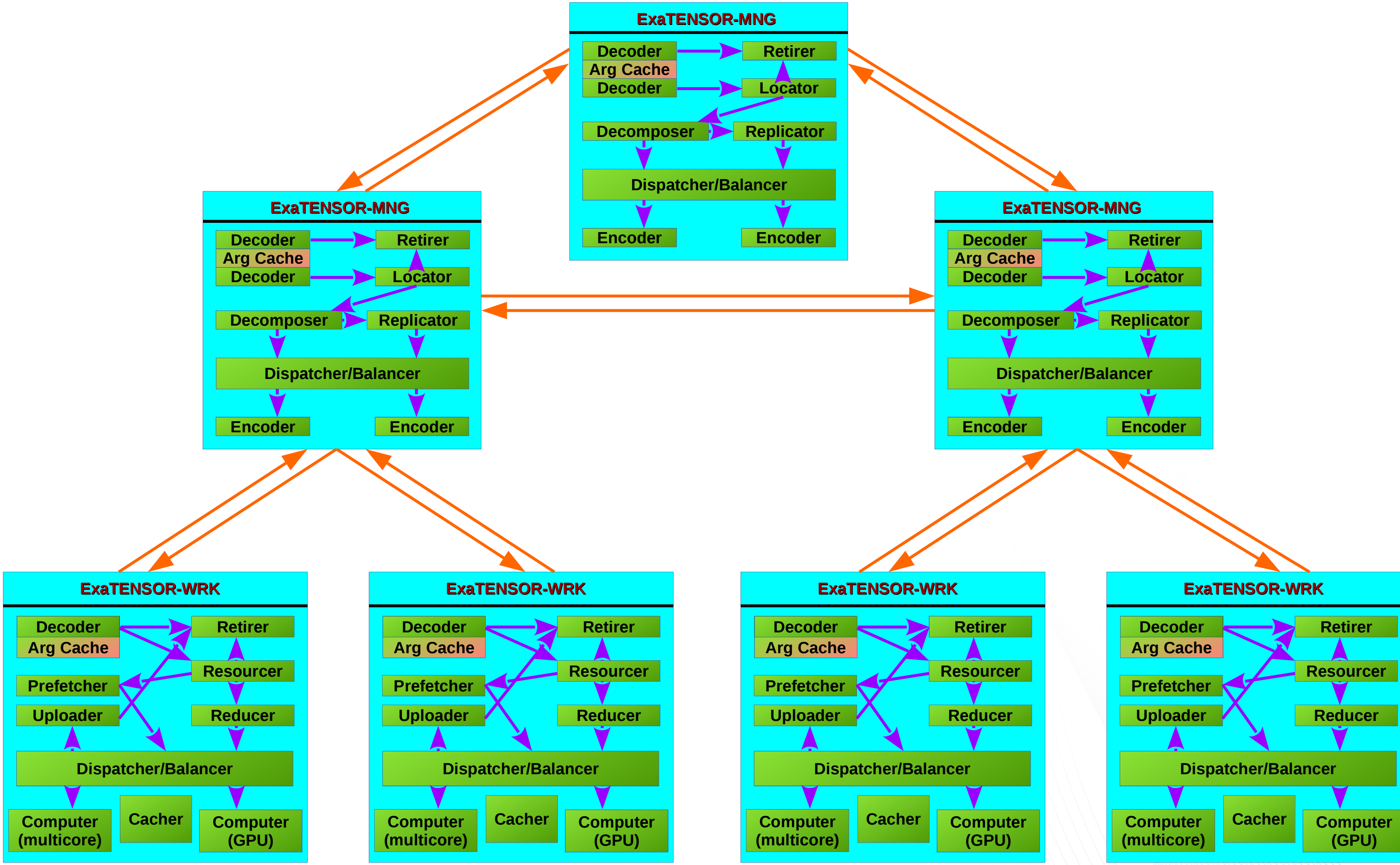
Domain-Specific Virtual Processor Architecture



Performance Portability Strategy

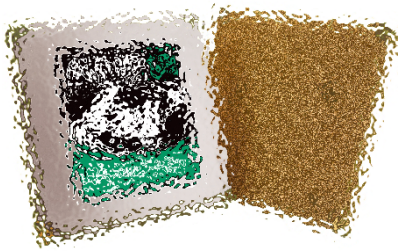
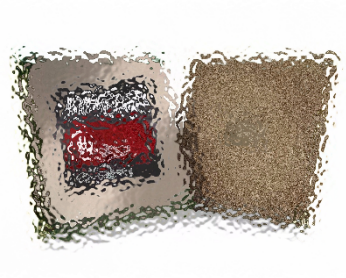
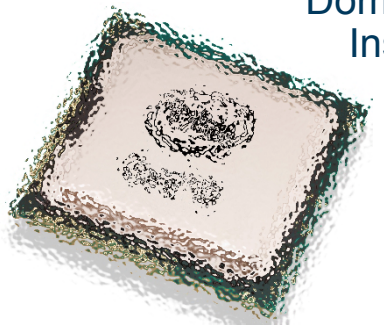
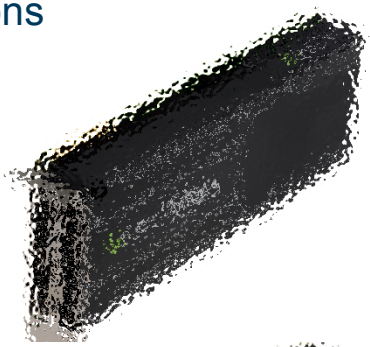
- **Abstract computing system template**:
  - Distributed (weakly-coupled) level: The computing system is composed of compute nodes interconnected via network interfaces in some topology
  - (Semi-)shared (strongly-coupled) level: Each node is composed of multiple compute devices of the same or different kinds, possibly sharing the same (hierarchical) memory
- Algorithms are formulated for this abstract computer
- The hardware specificity is masked by driver libraries that provide a device-unified API interface for a set of necessary domain-specific primitives (DS-ISA)
- New hardware = New driver library

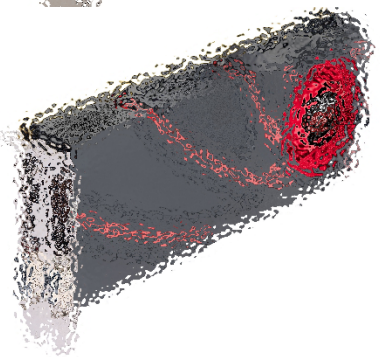
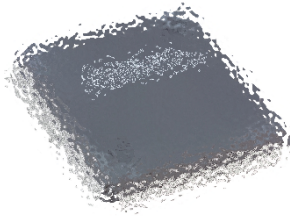
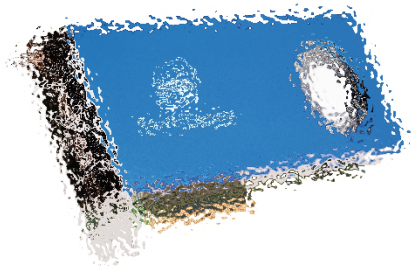
Hierarchical Virtualized HPC Platform

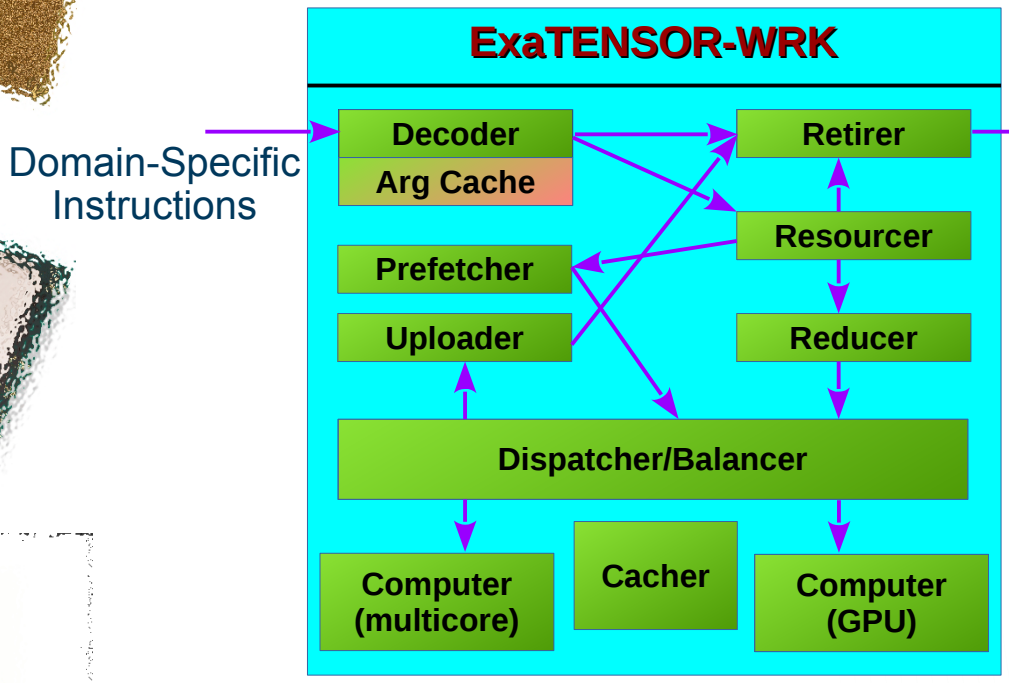


Node-Level Virtualization: Hiding Hardware

Domain-Specific Virtual Processor (DSVP)

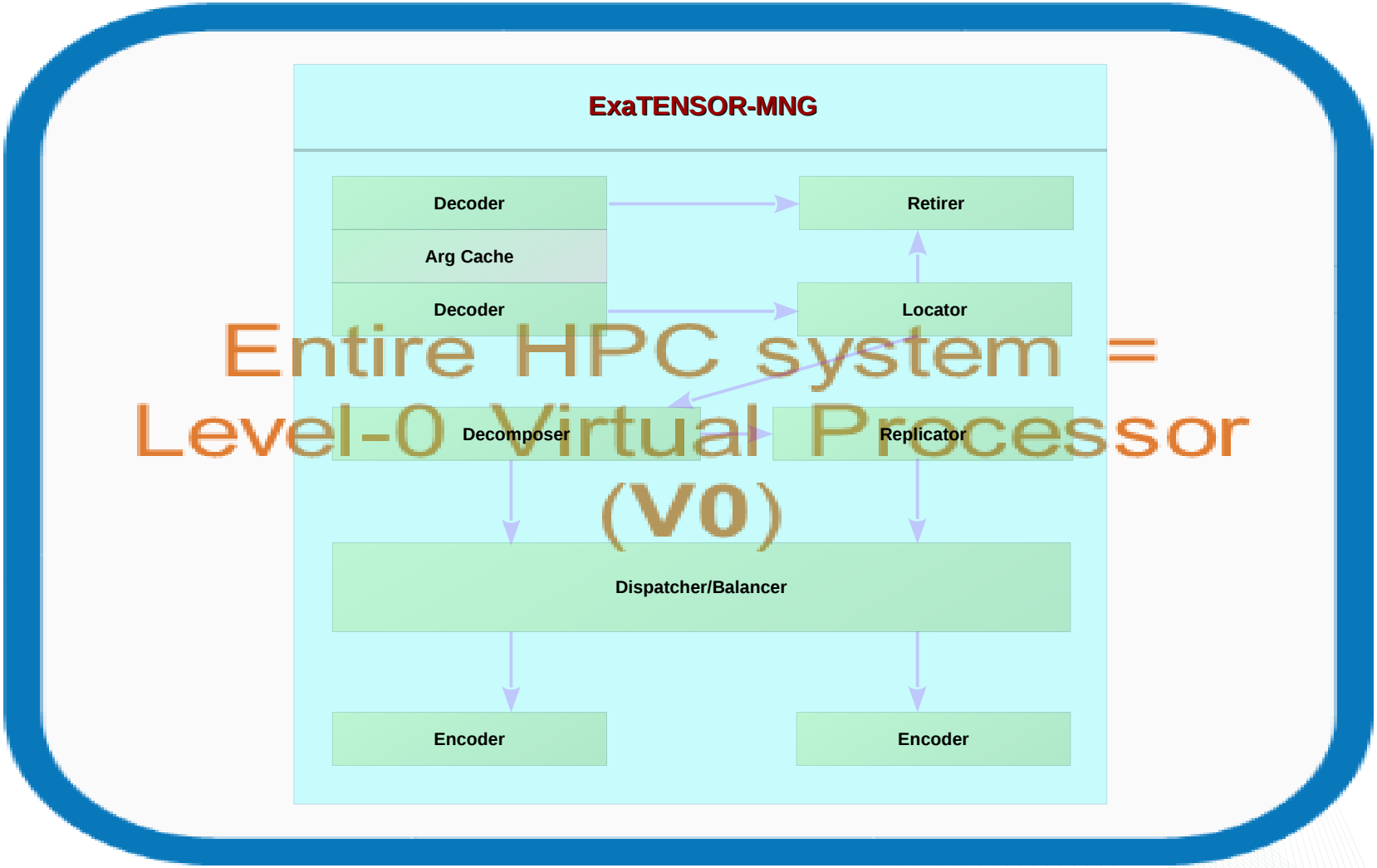





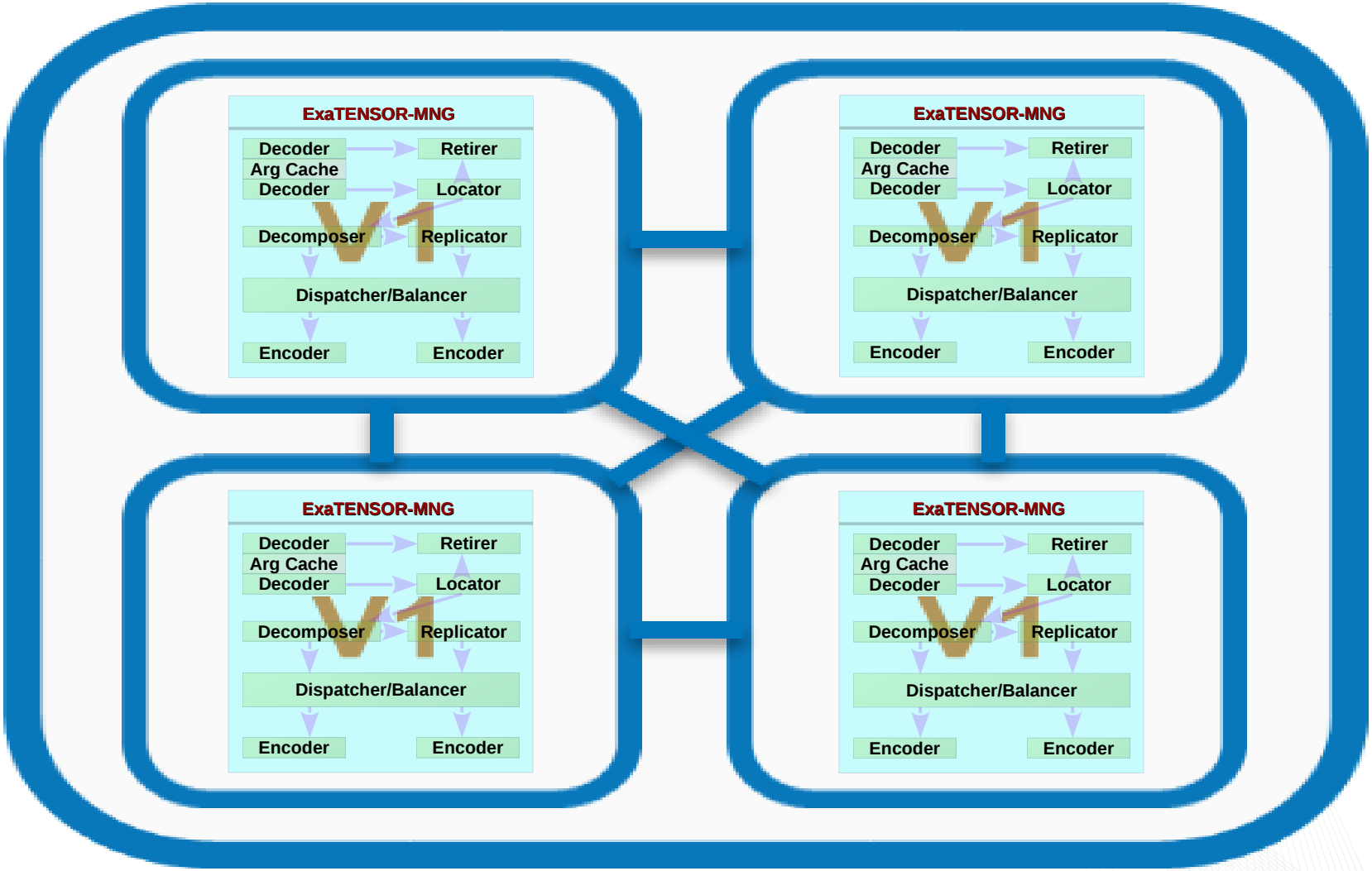


Global Virtualization: Hiding HPC Scale

Global Virtualization: Hiding HPC Scale

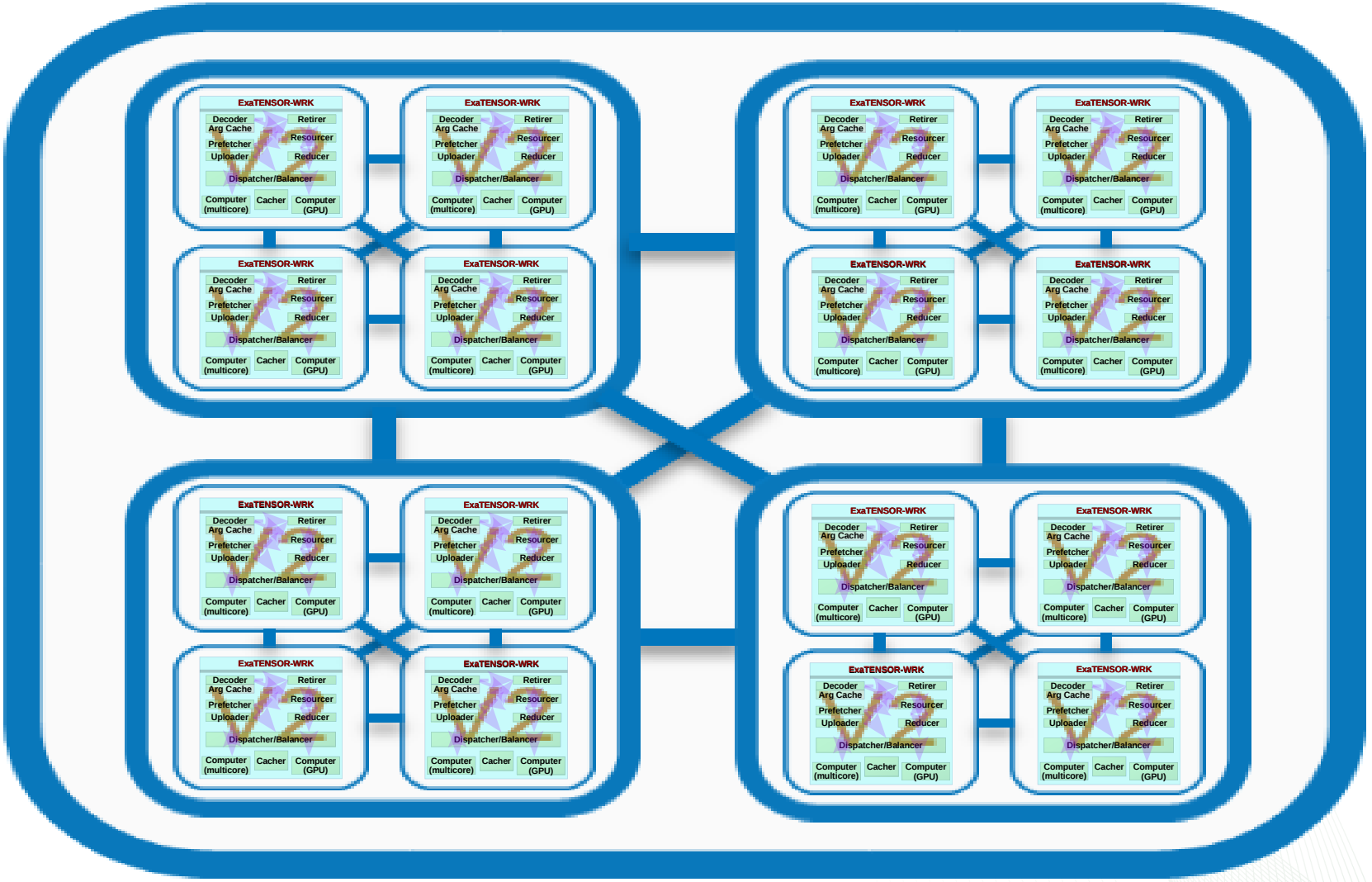


Global Virtualization: Hiding HPC Scale

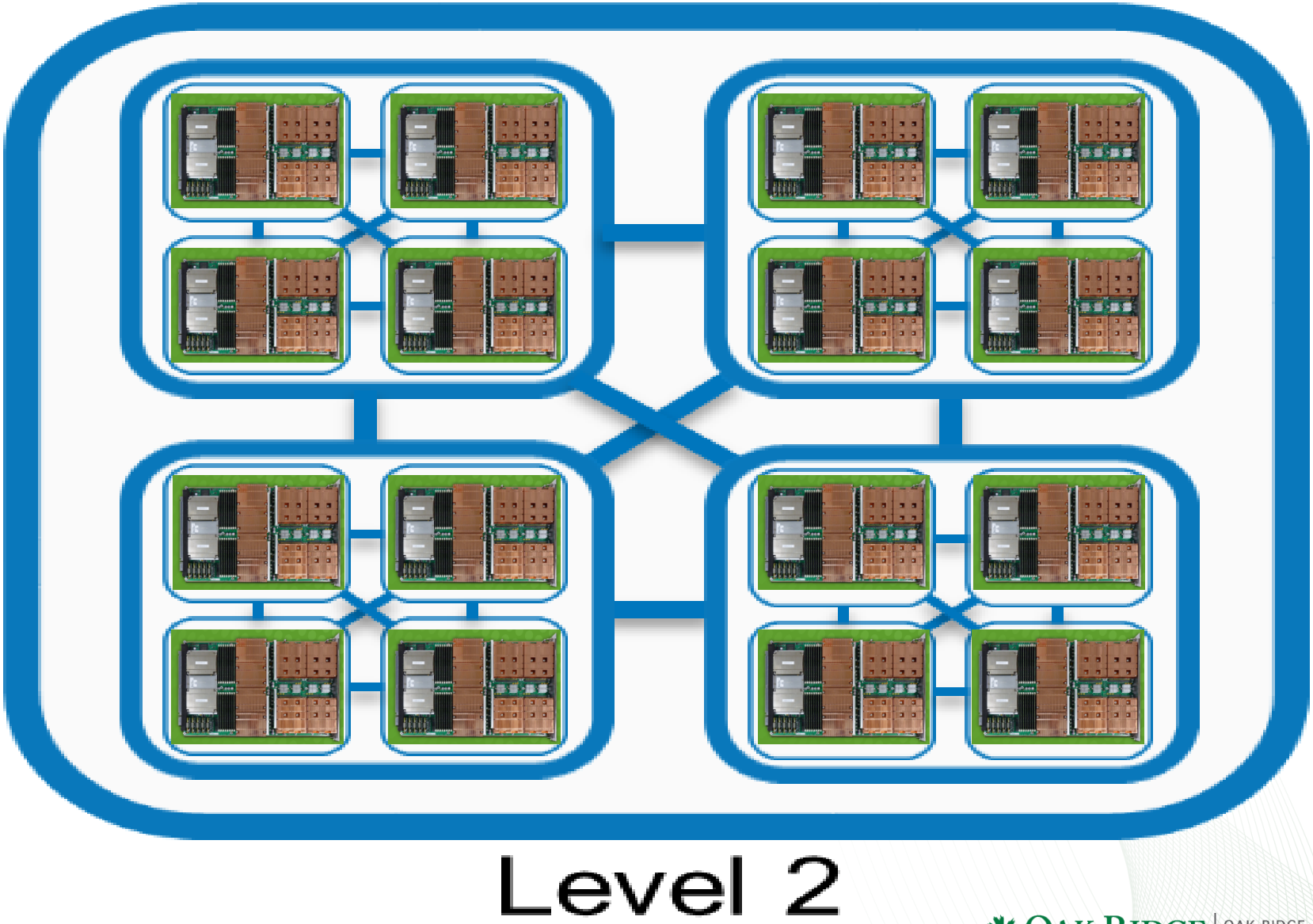


Level 1

Global Virtualization: Hiding HPC Scale

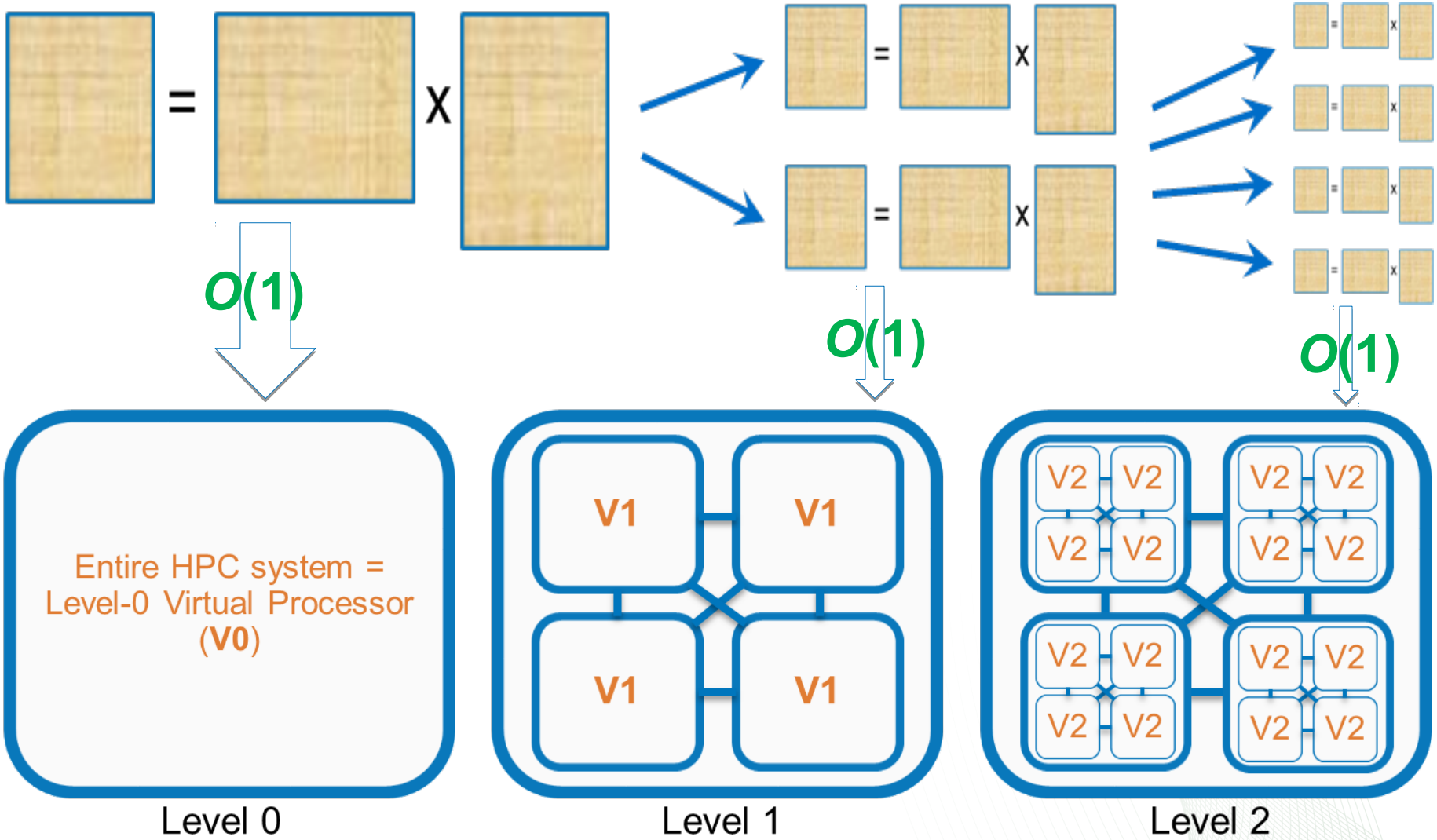


Global Virtualization: Hiding HPC Scale



Recursive Dynamic Task Scheduling

Data storage granularity is decoupled from the task granularity



How to Build DSVP

- **Domain-Specific Microcode**: Library-based implementation of domain-specific primitives, plus auxiliary operations: Manual or generated code
- **Resource Allocation Primitives**: Building blocks for hierarchical memory management: Target-agnostic (HiHAT?)
- **Data Transfer Primitives**: Building blocks for data transfer between devices in the same node as well as between nodes: Target-agnostic (HiHAT?)
- **Data Decomposition/Aggregation methods**: User-provided
- **Virtual Architecture Specification**: Composition of the domain-specific virtual processor in terms of domain-specific virtual units with well-defined functionality: Domain-provided