Design and Implementation of Knowledge Base for Runtime Management of Software Defined Hardware

Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan and Viktor Prasanna

University of Southern California

September 26, 2019

HPEC' 19, Waltham, MA USA

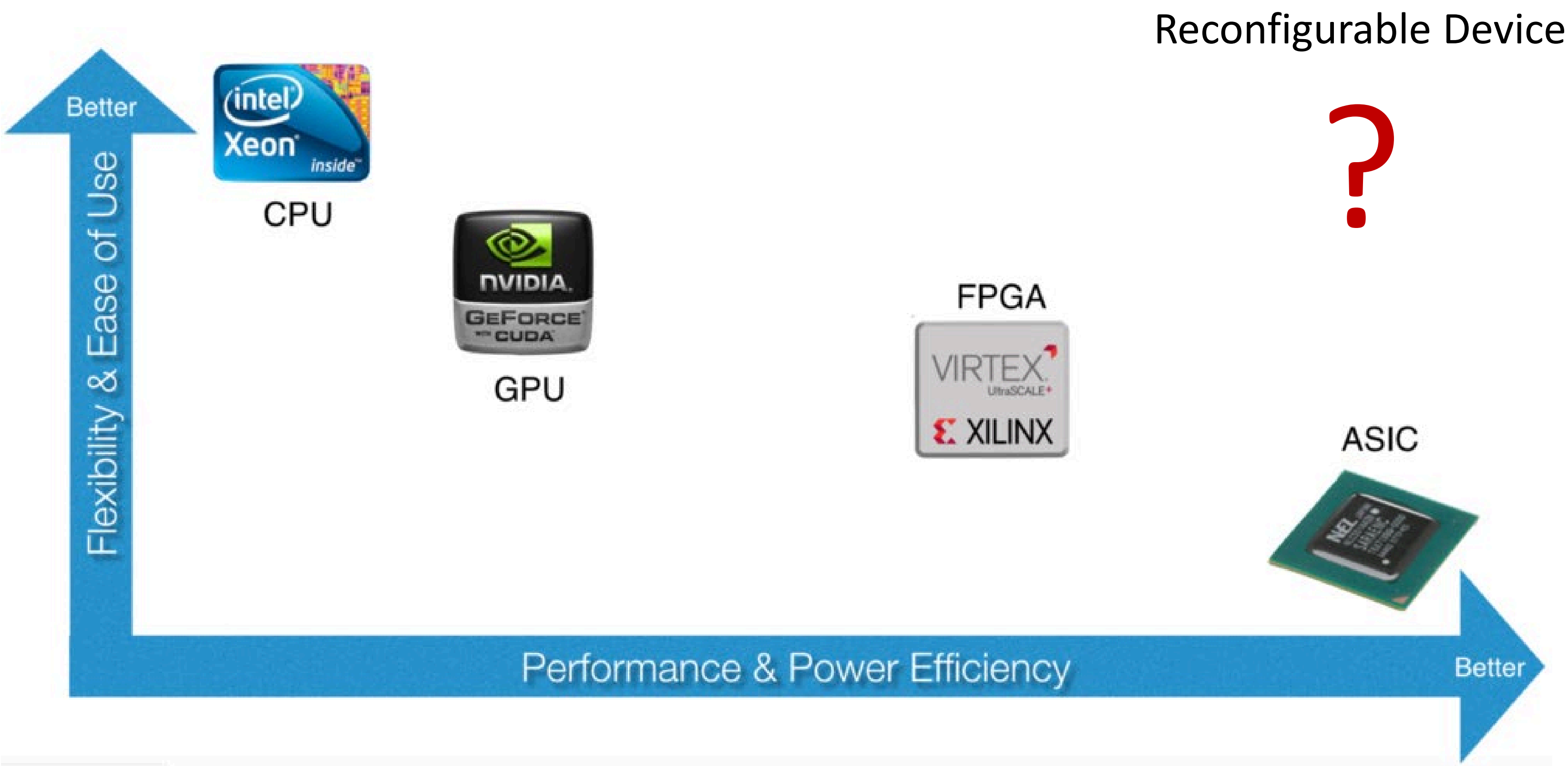
Outline

- Motivation and Background
- The Knowledge Base
  - Tripartite Representation
  - Creation
  - Interaction with Other Components
- Performance of the Knowledge Base
- Conclusion

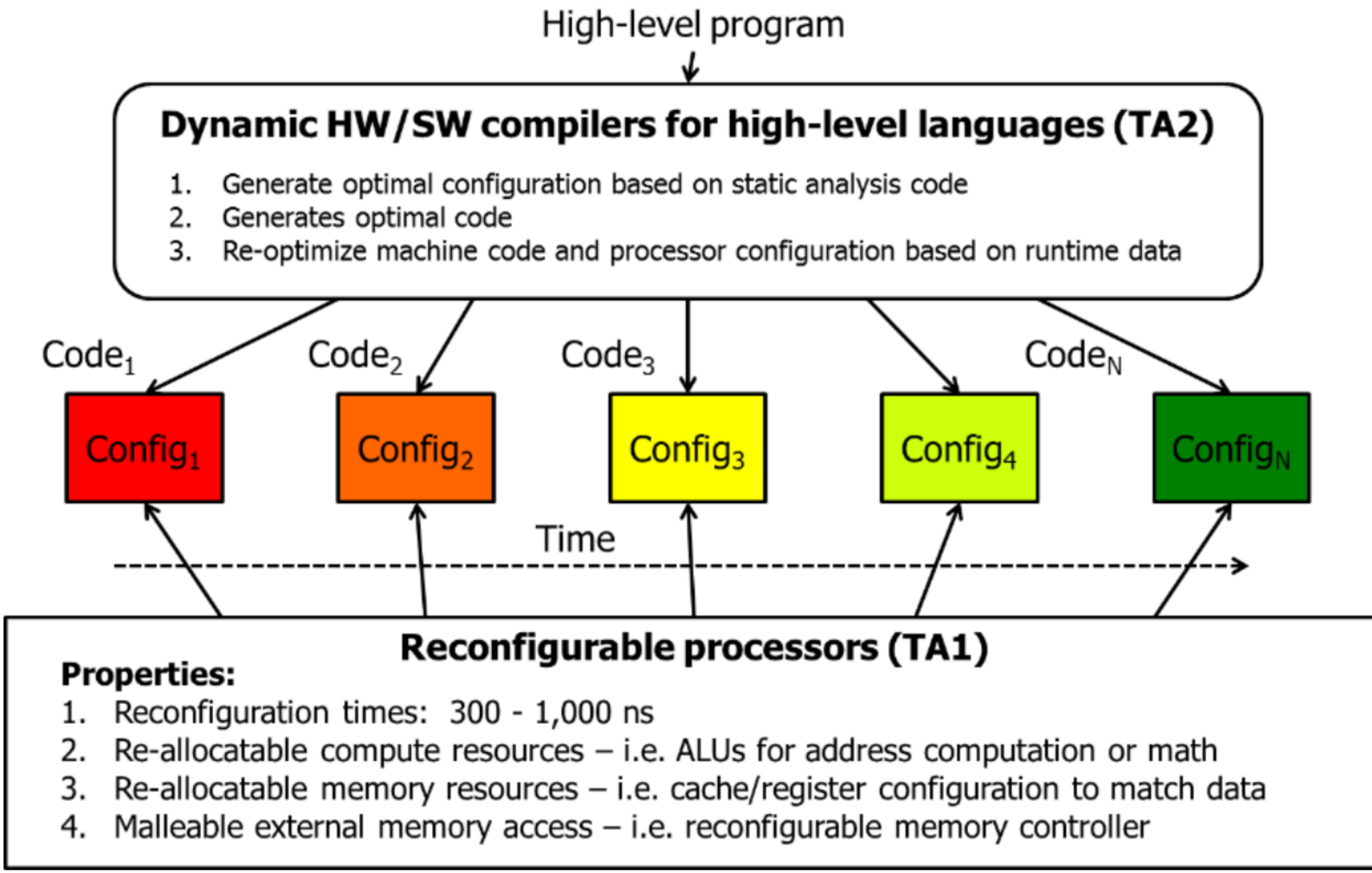
Outline

- Motivation and Background
- The Knowledge Base
  - Tripartite Representation
  - Creation
  - Interaction with Other Components
- Performance of the Knowledge Base
- Conclusion

Motivation

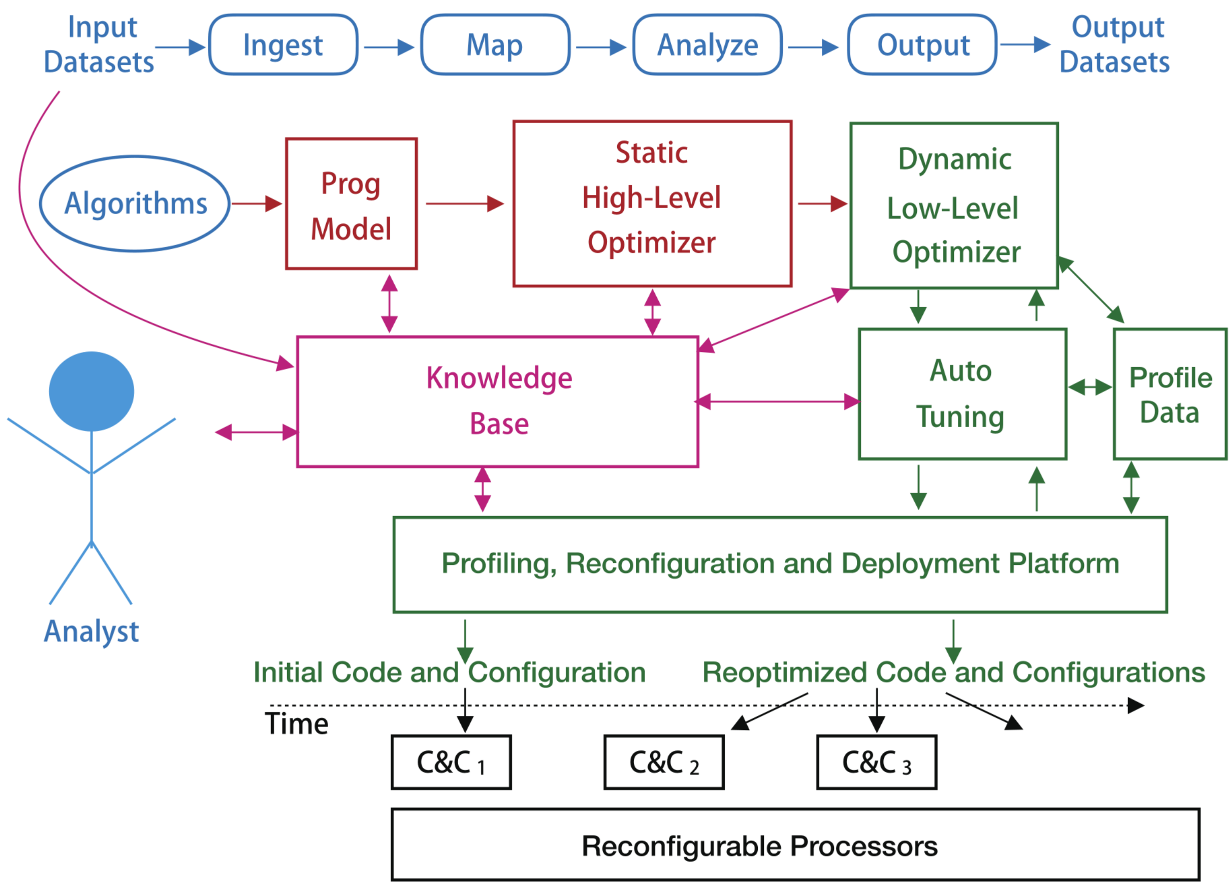

Reconfigurable Device
?
Better
Flexibility & Ease of Use
intel
Xeon
inside
CPU
NVIDIA
GEFORCE
CUDA
GPU
FPGA
VIRTEX
UltraSCALE+
XILINX
ASIC
Performance & Power Efficiency
Better

Background: SDH Program

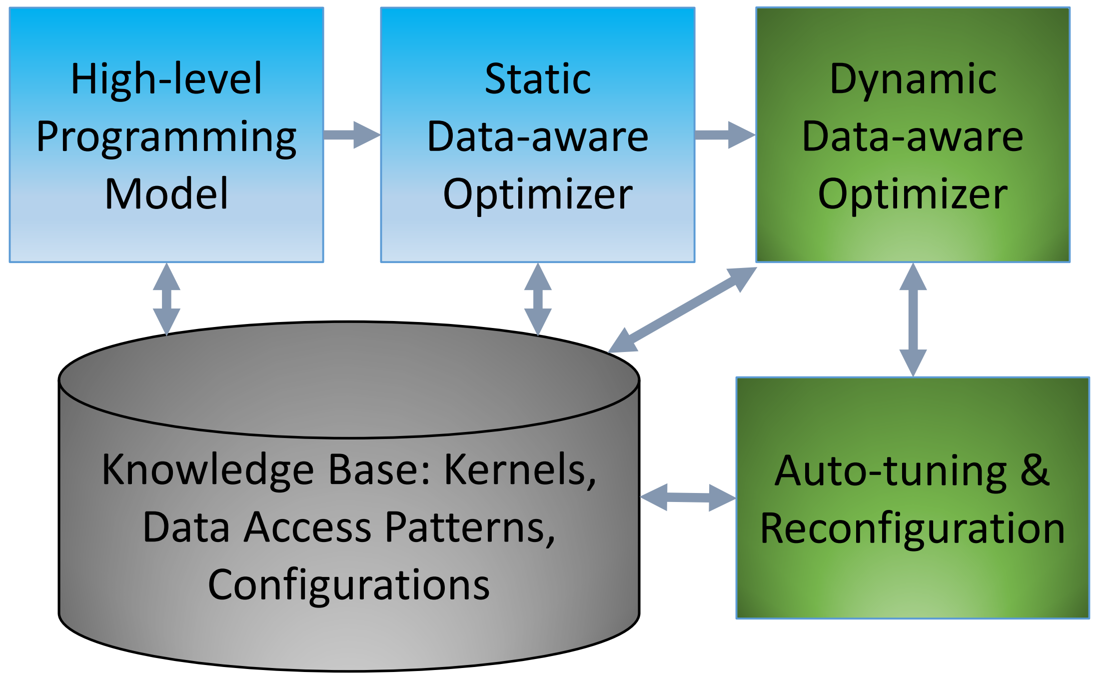


Background: DDARING Project

Aim to accelerate data-intensive workflows to achieve **near-ASIC performance** with **high-level programming language**



Background: DDARING Project



The knowledge base, a center component in the DDARING Project

Outline

- Motivation and Background
- The Knowledge Base
  - Tripartite Representation
  - Creation
  - Interaction with Other Components
- Performance of the Knowledge Base
- Conclusion

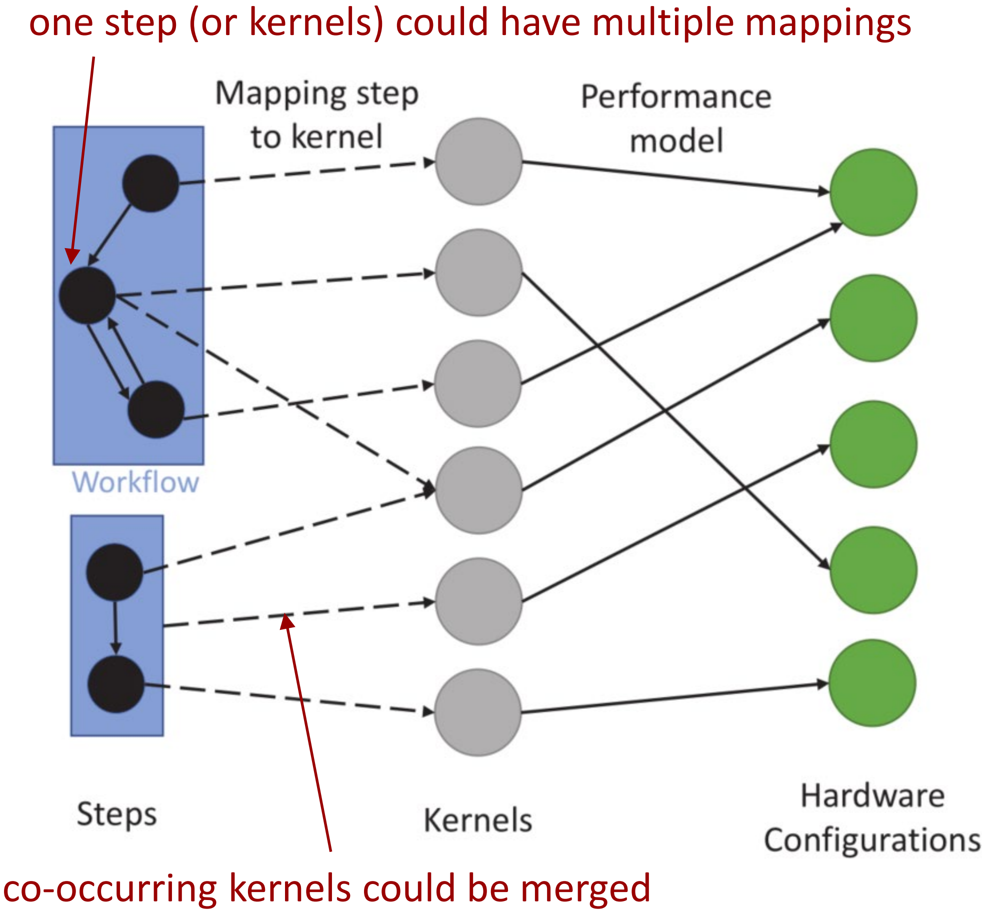
The Knowledge Base

- Challenge:
  - To store program metadata and status on previous executions
  - To analyze the gathered information
  - To answer queries from other component rapidly

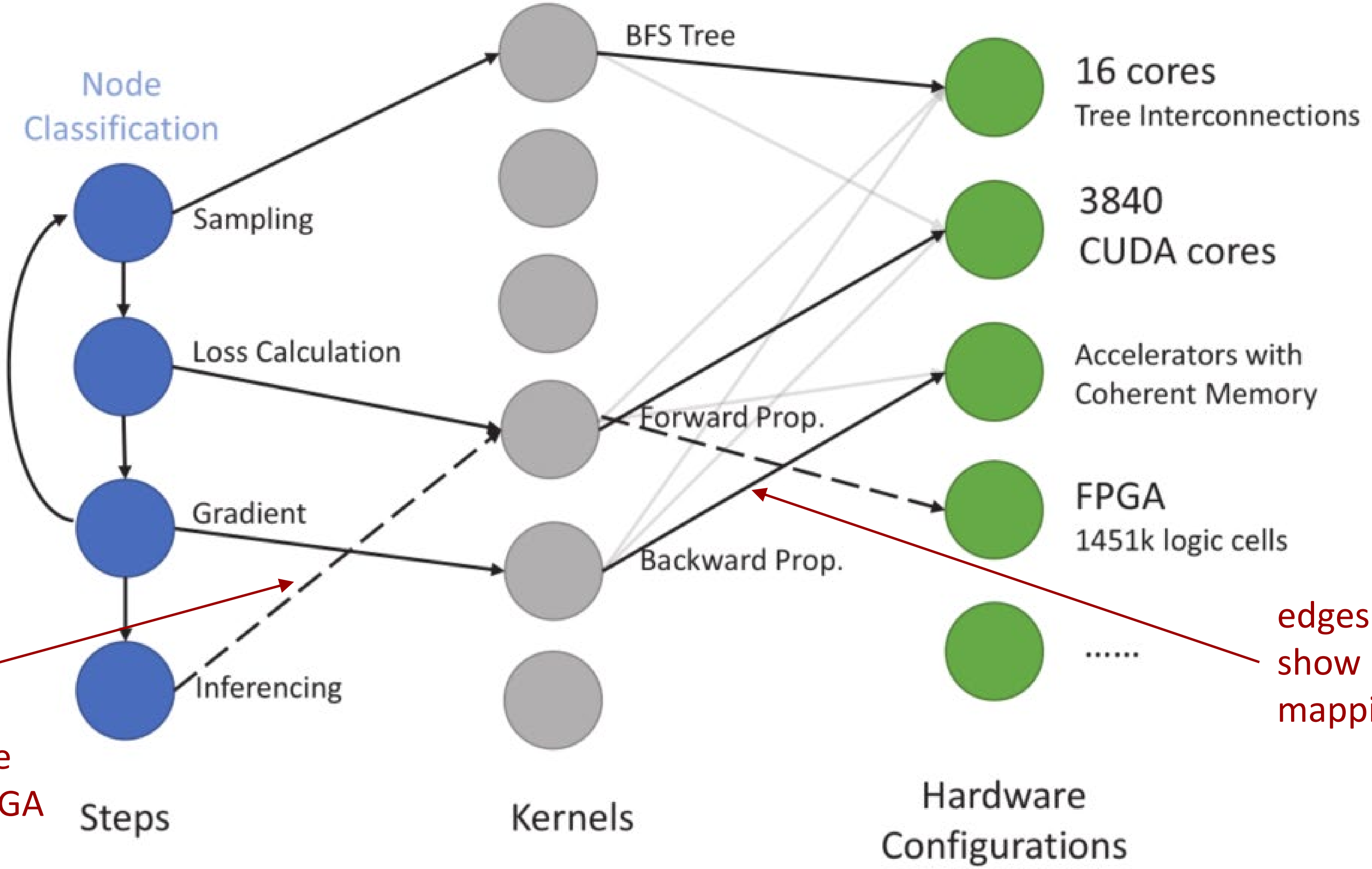
➢ Our approach: the knowledge base
  - ✓ **Dynamic** and **expendable**
  - ✓ Continuously gather knowledge during execution of workflow
  - ✓ Identify optimal implementations of workflows on optimal hardware configurations
  - ✓ Answer to compile- and run-time queries in real time

The Knowledge Base: Tripartite Representation

- Rich tripartite graph representation $G(V_1, V_2, V_3, E)$
  - V_1: Domain Specific Language Level steps
  - V_2: Bare bone tasks or kernels
  - V_3: Hardware configurations
  - $E(V_1, V_2)$: Mapping of steps to kernels, captures the algorithmic realizations of steps
  - $E(V_2, V_3)$: Mapping of kernels to hardware configurations, captures the performance models



The Knowledge Base: Tripartite Representation



dashed line means less precision needed, hence mapped to FPGA

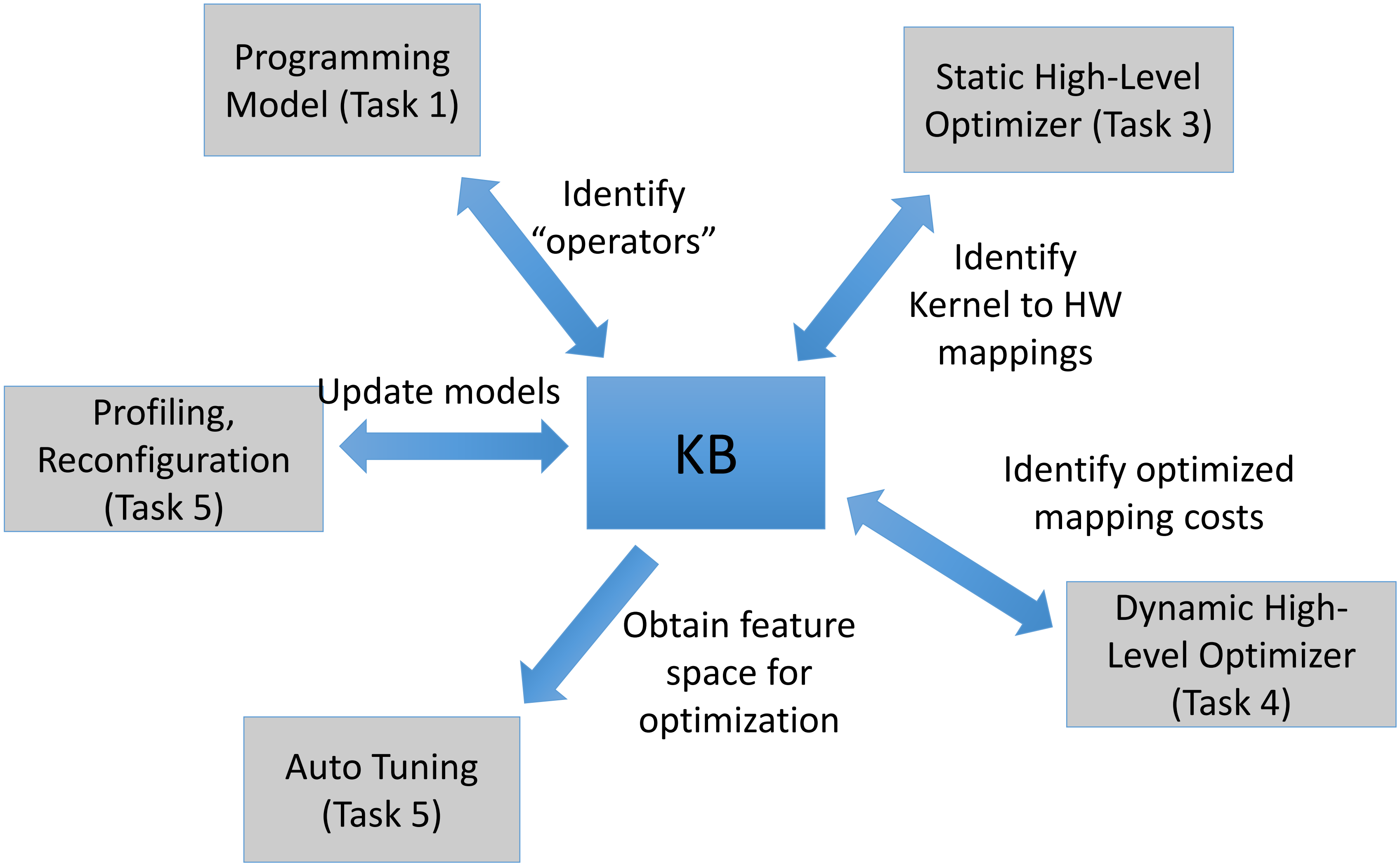
edges in black show optimal mapping

Example of the knowledge base capturing the node classification workflow
(demonstrated on heterogeneous architecture)

The Knowledge Base: Creation

- The knowledge base is created by **offline discovery** and **dynamic updates**.
  - Offline discovery:
    - Choose some typical workflows
    - Manually profile the selected workflows
    - Decompose the workflows into steps and kernels
    - Construct the performance model
  - Dynamic Updates:
    - Get updates from the profiling and reconfiguration component
    - Modify the tripartite graph accordingly

The Knowledge Base: Interaction with Other Components



Outline

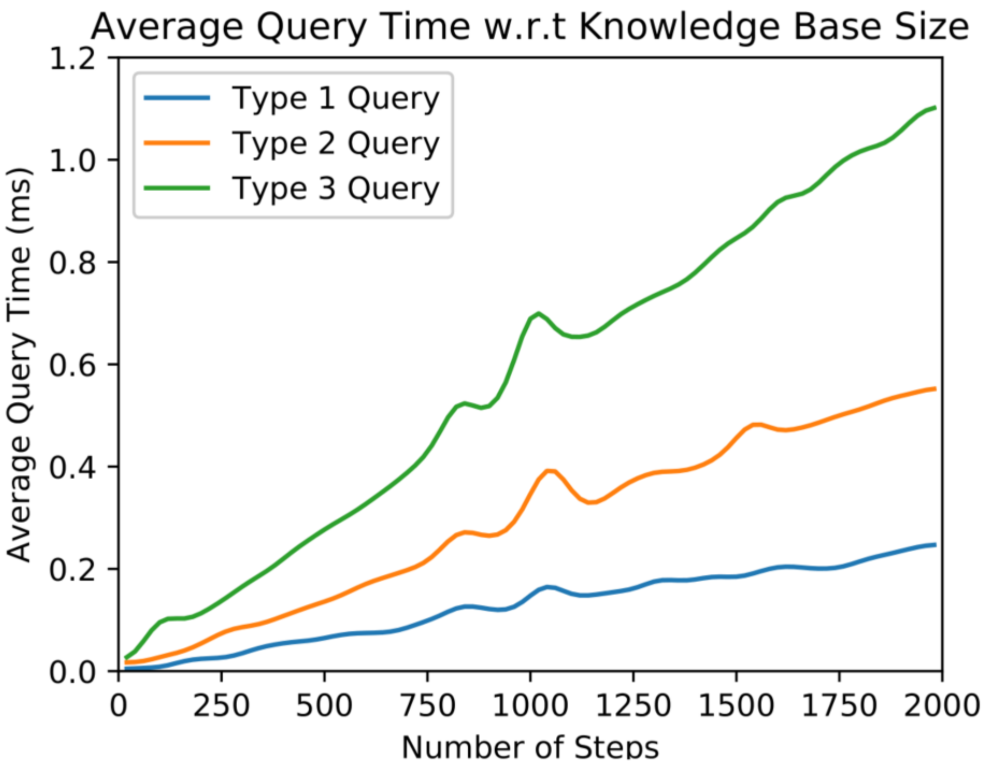
- Motivation and Background
- The Knowledge Base
  - Tripartite Representation
  - Creation
  - Interaction with Other Components
- Performance of the Knowledge Base
- Conclusion

Performance of the Knowledge Base

- The knowledge base implementation:
  - Object oriented, with each node or edge as a class
  - Adjacency list graph in the C++ Boost Library
  - Single thread

- Query types:
  - Type 1 query: For a given step in the knowledge base, return all kernels linked with that step.
  - Type 2 query: For a given kernel and a given hardware configuration in the knowledge base, return the performance model for executing the kernel on the hardware configuration.
  - Type 3 query: For a given kernel in the knowledge base, return the performance models for executing the kernels on all hardware configurations.

Performance of the Knowledge Base



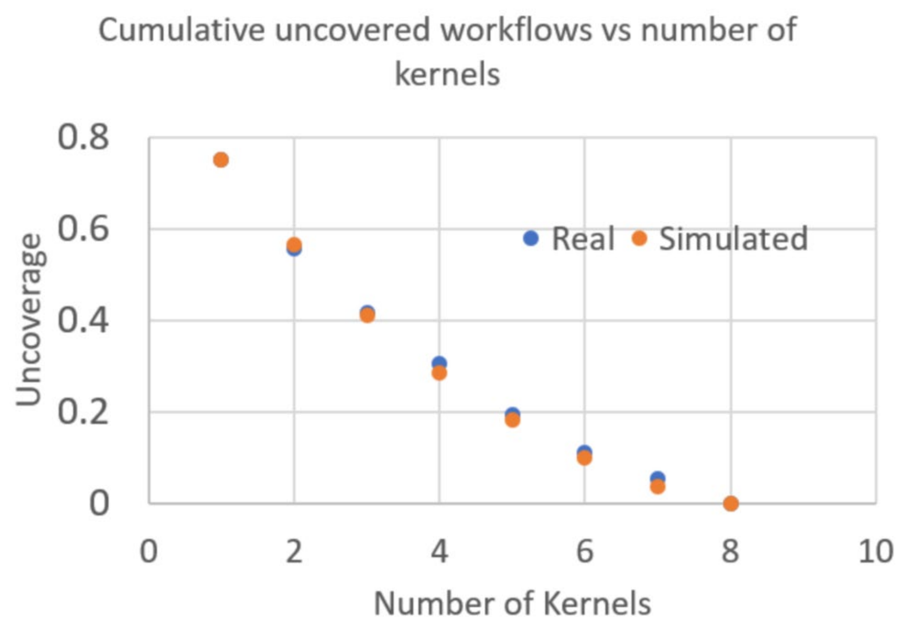
Number of kernels = 1.5 times than number of steps
Number of hardware configurations = 10

The feature of the nodes and edges are filled with random content

Kernel Coverage

- Query time ∝ size of the knowledge base
- However, we **do not need a large knowledge base**!
- Kernel coverage: percentage of stored kernels included a collection of workflows

➢ We model the coverage by setting the possibility that a kernel is included in one workflow follows **preferential attachment rule** with a fixed parameter λ



$\lambda = 2.5$ with real kernel coverage

Only 8 kernels needed to cover 60+ workflows!

Outline

- Motivation and Background
- The Knowledge Base
    - Tripartite Representation
    - Creation
    - Interaction with Other Components
- Performance of the Knowledge Base
- Conclusion

Conclusion

- We proposed a expendable and dynamic rich labeled tripartite network representation of the knowledge base.
- The knowledge base gathers the knowledge of optimized implementations of key algorithmic steps and kernels on various parameterized hardware.
- The knowledge base is implemented using the C++ Boost Library and is capable of answering different types of queries rapidly.

- On going work: *Portable lightweight deep learning models for kernel performance [submitted]*

Thank You!