ESPM2: 2022 ACM/IEEE 7th International Workshop on Extreme Scale Programming Models and Middleware

Panel: AI for HPC

Ali Jannesari,
Assistant Professor,
Iowa State University

Ali Jannesari is an Assistant Professor with the Computer Science Department at Iowa State University. He is the Director of the Software Analytics and Pervasive Parallelism Lab at ISU. His research primarily focuses on the intersection of high-performance computing (HPC) and data science. Prior to joining the faculty at ISU, he was a Senior Research Fellow at the University of California, Berkeley. He was in charge of the Multicore Programming Group at the Technical University of Darmstadt and a junior research group leader at RWTH Aachen University. He worked as a PostDoc fellow at Karlsruhe Institute of Technology and Bosch Research Center, Munich. Jannesari has published more than seventy refereed articles, several of which have received awards. He has received research funding from multiple European and US funding agencies. He holds a Habilitation degree from TU Darmstadt and received his Ph.D. degree in Computer Science from Karlsruhe Institute of Technology.

Panel: AI for HPC

ESPM2, Nov 14th 2022

Moderator: Ali Jannesari , Iowa State University

Panelists:

Mary Hall, University of Utah	Torsten Hoefler, ETH
Vipin Chaudhary, CRWU	Dong Li, UC Merced

SC22 Dallas, TX | hpc accelerates.

Agenda

3:30 pm - 3:35 pm CST	Panel opening and introduction of panelists
3:35 pm - 4:00 pm CST	Short presentation by each panelist (5-7 min): Research/experiences on using AI for HPC
4:00 pm - 4:50 pm CST	Interactive session (Q/A): Discussion on the challenges and opportunities
4:50 pm - 4:55 pm CST	Conclusion of major points of the panel

AI for HPC

Applying AI (DL) for the HPC domain including (but not limited to):

- Scheduling and resource management, memory and I/O,
- Profiling, runtime analysis, autotuning, device mapping,
- Code optimization, compilers, code generation, code translation,
- Verification and correctness analysis, debugging
- Power and energy efficiency,
- HPC artifacts,
- etc.

Challenges

- Unlabeled data and imbalance dataset (not enough data)
- Data augmentation and data generation
- Code representation (sequence of tokens, graph, AST, IR, MLIR), *universal* code representation
- Profiling and runtime Information (profiling & execution overhead)
- Characteristics of HPC code and data, ecosystems, and environments
- System heterogeneity and performance portability
- Data heterogeneity and transfer learning
- Efficient models, multimodal, transformers, etc.
- Unsupervised, self-supervised (Reinforcement Learning), and semi-supervised learning
- Zero-shot learning with a small dataset
- Models for scientific and HPC applications
- Domain-aware (physics-aware) models (hybrid models)
- Explainability (Interpratablity) of AI
- Meta-learning
- AI/HPC artifacts and FAIR principles (findability, accessibility, interoperability, and reusability)

Potential Questions...

- AI for HPC, Yes or No?
- Can AI be used for developing smart compilers? What about auto-tuners? Other tools?
- What are the challenges of using AI/ML to generate synthetic datasets? What criteria should be used to judge their usefulness?
- Should we trade off the correctness of the suggestion model in HPC over the running time?
- New programming models/DSLs?
- Synergic efforts of the HPC community?

Q/A

- Interactive session: Q/A
- Thank you!

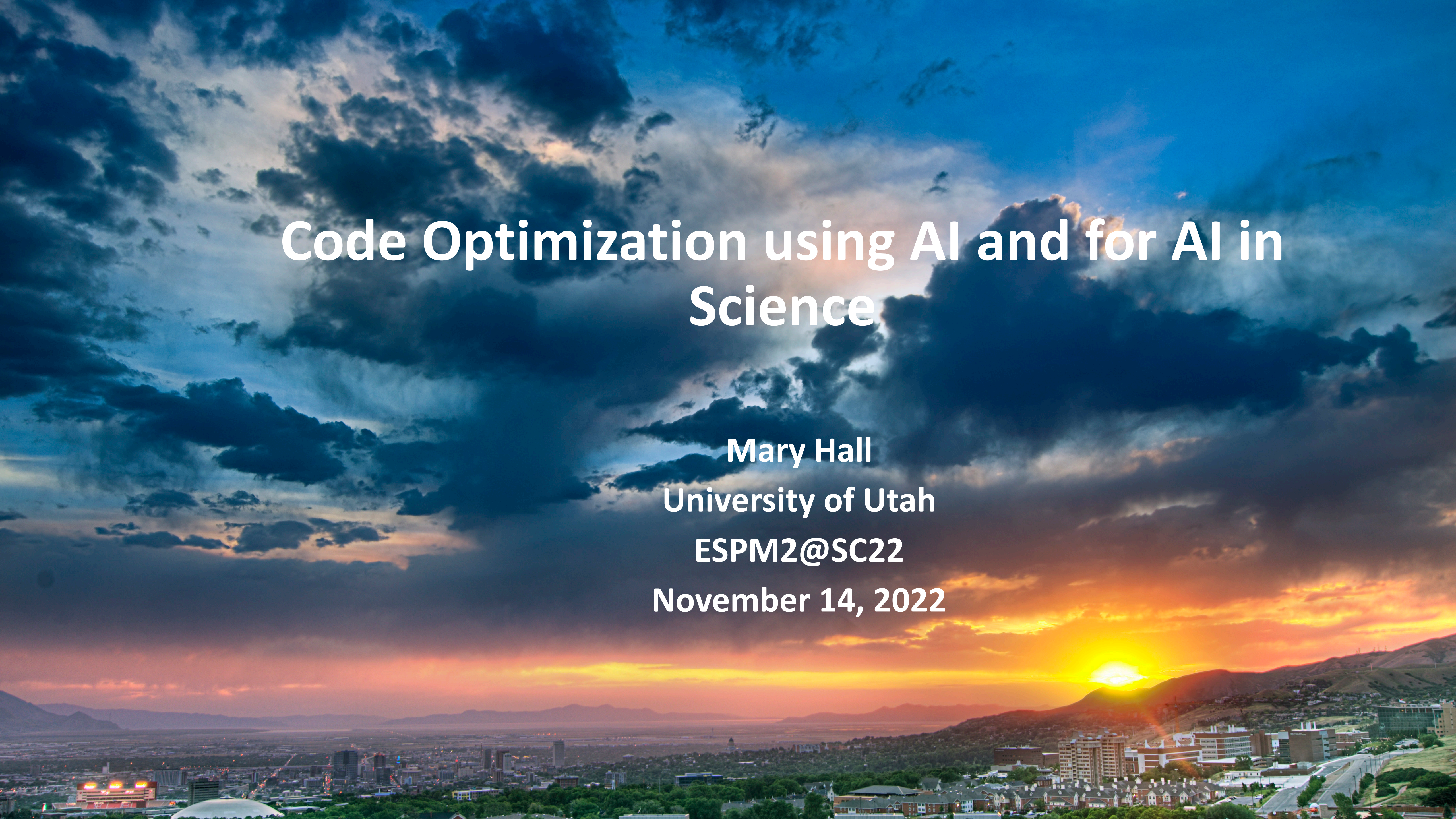
HPC for AI (AI-centric Challenges) – Accelerating AI…

High-performance machine/deep learning:

- Scalable Learning and large-scale AI models
- Accelerating Training time: Distributed training and parallelism (data, model, and hybrid)
- Network Architectural Search (NAS)
- Programming model supports
- Accelerating inference time and efficient inference
- Data heterogeneity and system heterogeneity in Learning
- Communication overhead in distributed training and decentralized learning
- Inefficient resource utilization
- Scheduling and resource management of AI Jobs (AI-Job/training-job scheduler)
- Performance of distributed physics-aware ML models
- Data privacy in decentralized learning and Federated Learning (FL)
- Cross-silo FL and cross-device FL
- Asynchronous FL/decentralized environment
- Limited computational resources on edges
- HPC centers usually don't support elastic distributed training

Potential Question on High-performance machine/deep learning

- What are some of the metrics other than FLOP/s and training time to evaluate distributed deep learning models?
- How can GPU (accelerator) trace activities be analyzed for distributed deep learning models?
- What are some of the common bottlenecks of model parallelism? How to identify and overcome them?
- What are the better metrics for evaluating the model accuracy in HPC?
- Besides the correctness of models, how should power/energy usage be considered?
- Can we convince HPC centers to provide additional support for efficient AI/ML training jobs?
- New programming models/DSLs?



Code Optimization using AI and for AI in Science

Mary Hall
University of Utah

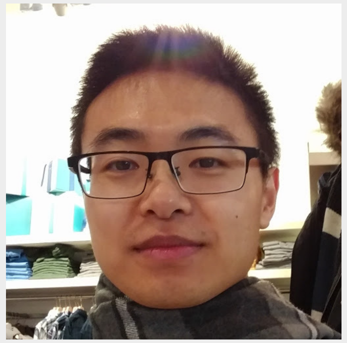
ESPM2@SC22

November 14, 2022

Acknowledgements

Tharinda Rusira Patabandi
Samsung

Anand Venkat
Nvidia



Tuowen Zhao
SambaNova

John Jolly
University of Utah

Mahesh Lakshminarasimhan
University of Utah

Oscar Antepara
LBNL

Additional collaborators:

Raj Barik, Justin Gottschlisch, Abhishek Kulkarni, Pushkar Ratnalikar, Leonard Truong, Sam Williams

This research was supported in part by the National Science Foundation under CCF-1564074, Intel Corporation, and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

1. Using ML to Derive Loop Transformation Schedules

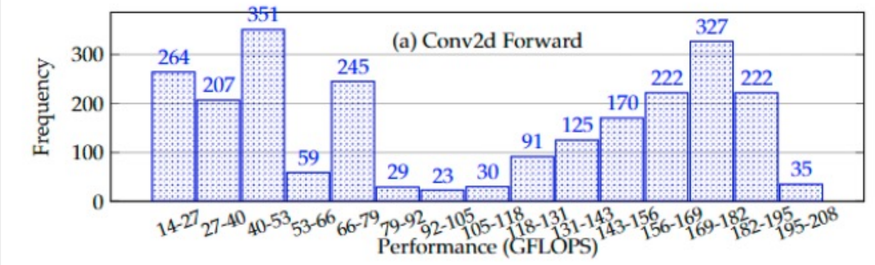
- Compilers perform code reordering transformations to, among other reasons, to
  - **change memory access order so that data is accessed in fast, nearby storage**
- Benefits and risks
  - Significant performance improvements possible, but best solutions may be difficult to derive, and are architecture-specific and input-dependent
- Consider: Convolutional Neural Networks (CNNs)
  - Similar optimization requirements to Matrix-Matrix multiply, but more degrees of freedom
  - 7-deep loop nests, 4D tensors, stencil pattern on one dimension
  - Architecture-specific solutions can achieve close to peak performance for large enough networks

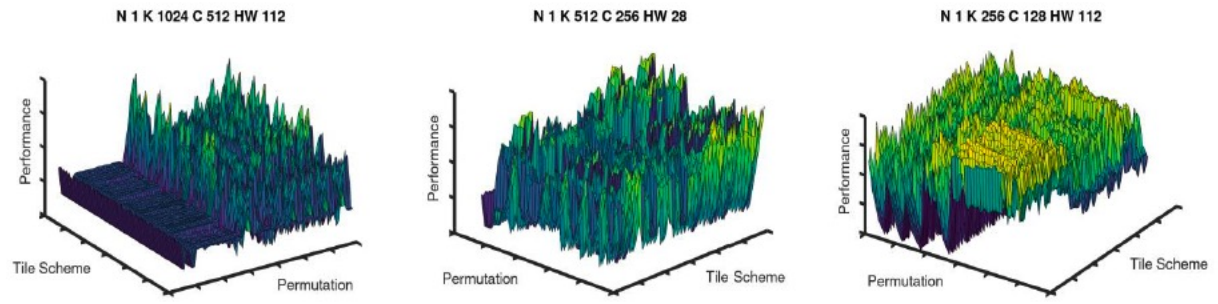
Loop Permutation:

Reorder loops in a nest <i,j> □<j,i>

Loop Tiling:

Partition iteration space to touch smaller amount of data to improve cache hit rate





Approaches to Transformation Selection: Example Loop Permutation

Apply Heuristics: Permute to innermost position the loop that carries the most reuse

Analytical Model: Evaluate data's cache footprint for different loop orders, minimize memory cost

Enabled by Exploration-Enabled Compiler Structure

Assisted Manual: Describe transformations in a scheduling language, compiler carries out

Autotuning: Encode search space for loop order, perform empirical search, guided by statistical model

Predictive Model: Encode loop order, train across a test set, predict best performance during inference

Big Picture: Composing Predictive Models for Sequence of Transformations

Typically multiplicative:

- Learning phase evaluates a collection of transformation schedules on an architecture with different problem sizes
- Training either very expensive, or exploration limited

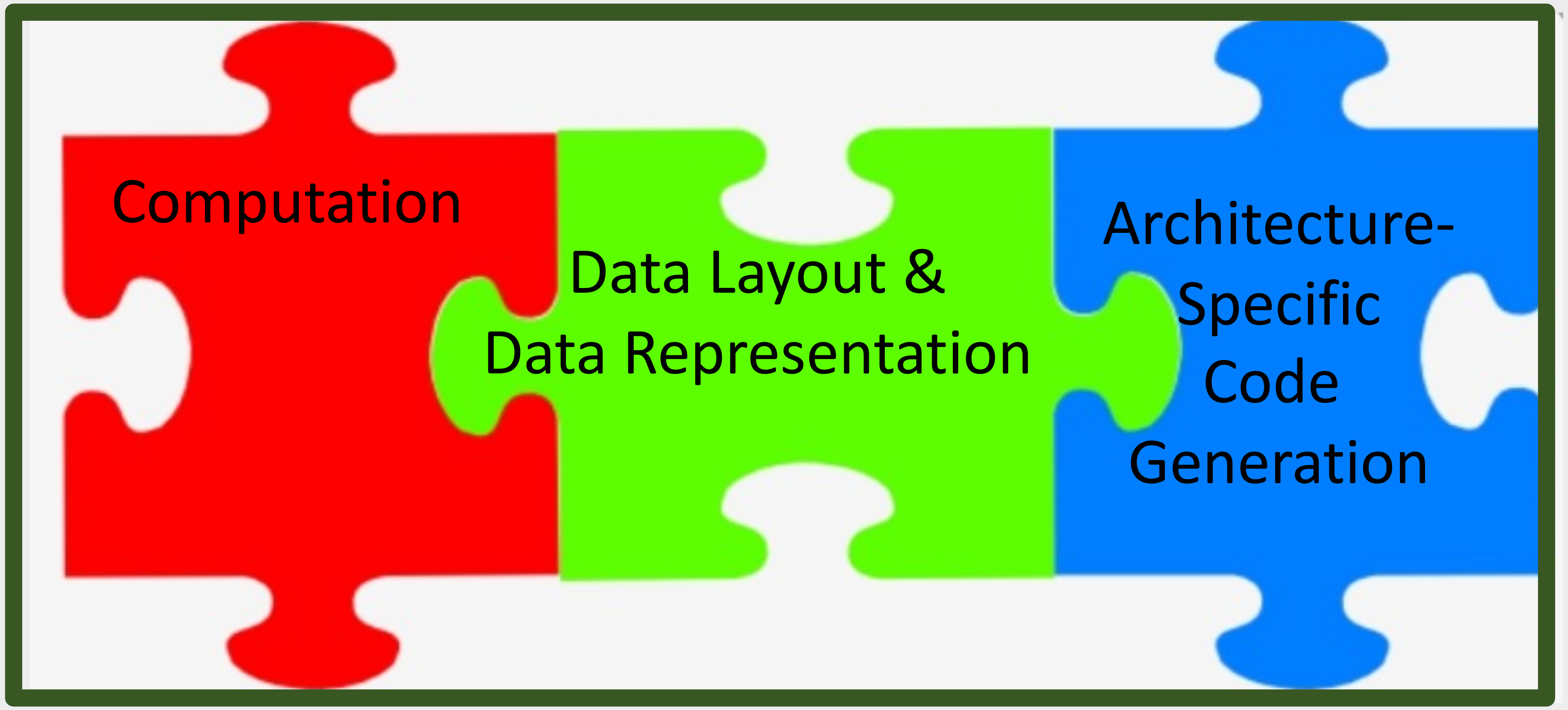
Can it be additive?

- Learning phase evaluates each transformation (somewhat) independently
- Dramatic reduction in training time

Result:

- Learning-derived schedule achieves average <u>1.5X speedup</u> relative to hand-tuned oneDNN library (see Rusira poster tomorrow!)

2. Data Layout Integrated into Compilation Flow



- Implicit or explicit data layout is exposed to optimization and code generation to reduce/optimize data movement (e.g., NCHW[x]C)
- Richer set of layouts than traditional compilers, tailored to architectures

Example: Sparse Tensor Co-Iteration

Goal: Dot product of two sparse tensors: v() = A(i)*B(i)

Physical Layout:

```c
// Data structure definition
struct SpVec { int len; int *idx; double *val; };
```

Physical to Logical Mapping:

$$IS = \{[pA, pB, i] \mid A.idx(pA) = i = B.idx(pB) \wedge 0 \leq pA < A.len \wedge 0 \leq pB < B.len\}$$

Co-Iteration Code:

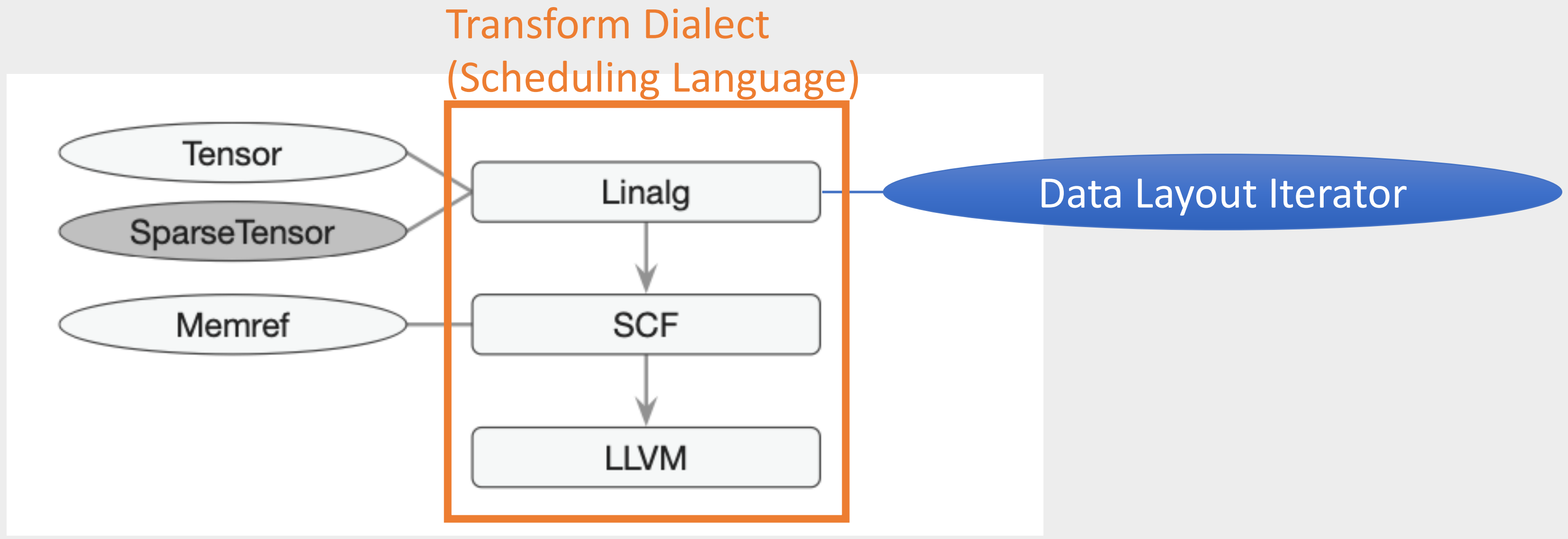
```c
pB = 0;
for (int pA = 0; pA < A.len; ++pA) {
 i = A.idx[pA];
 while (pB < B.len && i > B.idx[pB]) ++pB;
 if (pB < B.len && i == B.idx[pB])
   { v += A.val[pA] * B.val[pB]; ++pB; }
}
```

Iterate over nonzeros in A

if (pb = find(i, B))
    perform multiply

3. Open Source MLIR Compiler

- MLIR: Multi-Level Intermediate Representation
  - Google, with community engagement
  - Composable dialects at different levels of abstraction provide domain-specific building blocks
  - Part of LLVM ecosystem, lowers to LLVM



Compiler Support for Sparse Tensor Computations in MLIR, Aart J.C. Bik, Penporn Koanantakool, Tatiana Shpeisman, Nicolas Vasilache, Bixia Zheng, and Fredrik Kjolstad, arXiv:2202.04305, Feb. 2022.

References

T. Patabandi, A. Venkat, A. Kulkarni, P. Ratnalikar, M. Hall, J. Gottschlich, Predictive Data Locality Optimization for Higher-Order Tensor Computations, 5th ACM SIGPLAN International Symposium on Machine Programming (MAPS), June 2021.

T. Patabandi, Guiding Loop Optimizations for Higher-Order Tensor Computations, PhD Dissertation, University of Utah, August, 2022.

T. Zhao, T. Popoola, M. Hall, C. Olschanowsky, M. Strout, Polyhedral Specification and Code Generation of Sparse Tensor Contraction with Co-Iteration, ACM TACO, 2022.

J. Jolly, P. Goyal, V. Kumar, H. Johansen, M. Hall, Tensor Iterators for Flexible High-Performance Tensor Computation, Workshop on Languages and Compilers for Parallel Computing, Oct. 2022.

T. Zhao, S. Williams, M. Hall and H. Johansen, Delivering Performance-Portable Stencil Computations on CPUs and GPUs Using Bricks, P3HPC’18, 2018.

T. Zhao, P. Basu, S. Williams, M. Hall, and H. Johansen. Exploiting reuse and vectorization in blocked stencil computations on CPUs and GPUs, SC’19, Nov. 2019.

T. Zhao, M. Hall, H. Johansen, and S. Williams. Improving communication by optimizing on-node data movement with data layout, PPoPP’21, Feb. 2021.

Backup Slides

Restructure Compiler for Exploration

Late 1990s-2000

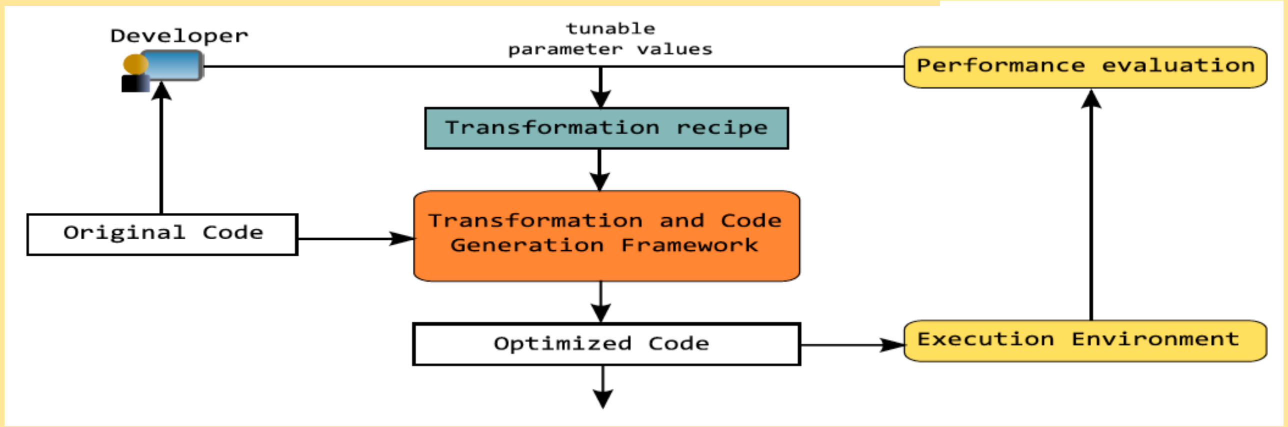
Autotuning Libraries automatically explore a search space (PhiPAC, ATLAS)

Iterative Compilation compiler searches xforms (Bodin et al.)

2005

X Language exposes xforms to programmer (Donadio et al.

2007 Chen, CHiLL Compiler (from LCPC 2009)



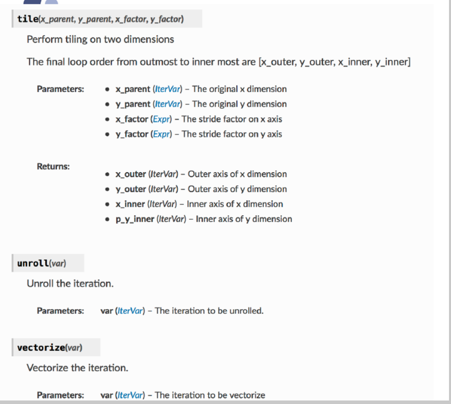
2013-present

Halide

a language for image processing and computational photography

```
vectorize(x_inner, factor), equivalent to
gradient.split(x, x, x_inner, 4);
gradient.vectorize(x_inner);
gradient.parallel(tile_index);
gradient.split(x, x_outer, x_inner, 2);
gradient.unroll(x_inner), equivalent to
gradient.unroll(x, 2);
gradient.tile(x, y, x_outer, y_outer, x_inner,
y_inner, 4, 4);
gradient.reorder(y, x); // similar to transpose
gradient.split(x, x_outer, x_inner, 2)
fuse(x, y, fused)
```

saml **TVM Stack**



tile(*x_parent, y_parent, x_factor, y_factor*)

Perform tiling on two dimensions

The final loop order from outmost to inner most are [x_outer, y_outer, x_inner, y_inner]

Parameters:
- **x_parent** (*IterVar*) – The original x dimension
- **y_parent** (*IterVar*) – The original y dimension
- **x_factor** (*Expr*) – The stride factor on x axis
- **y_factor** (*Expr*) – The stride factor on y axis

Returns:
- **x_outer** (*IterVar*) – Outer axis of x dimension
- **y_outer** (*IterVar*) – Outer axis of y dimension
- **x_inner** (*IterVar*) – Inner axis of x dimension
- **p_y_inner** (*IterVar*) – Inner axis of y dimension

unroll(*var*)

Unroll the iteration.

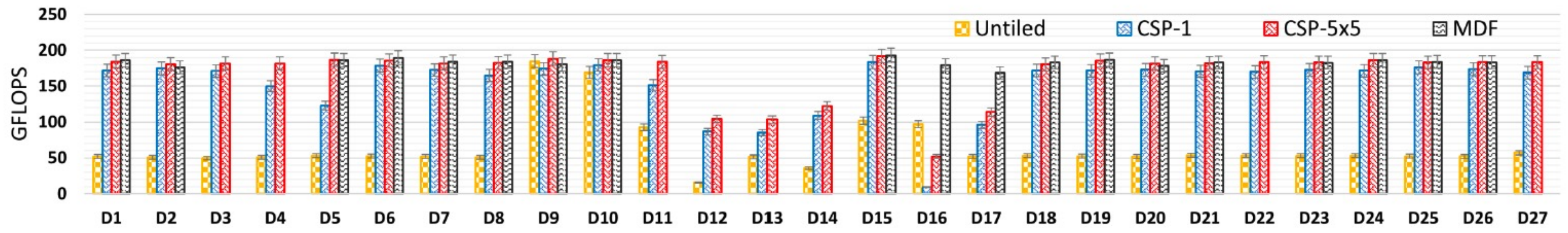
Parameters: **var** (*IterVar*) – The iteration to be unrolled.

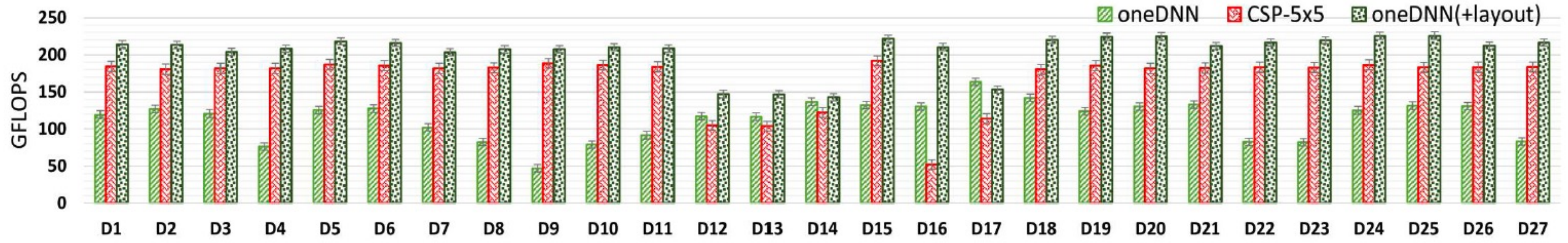
vectorize(*var*)

Vectorize the iteration.

Parameters: **var** (*IterVar*) – The iteration to be vectorize

Performance Comparisons





	Untiled		oneDNN	
	AVG	MAX	AVG	MAX
CSP-1	2.4×	5.5×	1.4×	3.7×
CSP-5x5	2.7×	6.6×	1.5×	4.0×

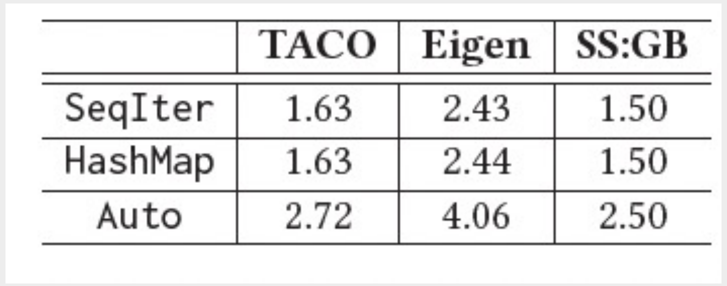
References:

T. Patabandi, A. Venkat, A. Kulkarni, P. Ratnalikar, M. Hall, J. Gottschlich, Predictive Data Locality Optimization for Higher-Order Tensor Computations, 5th ACM SIGPLAN International Symposium on Machine Programming (MAPS), June 2021.

T. Patabandi, Guiding Loop Optimizations for Higher-Order Tensor Computations, PhD Dissertation, University of Utah, August, 2022.

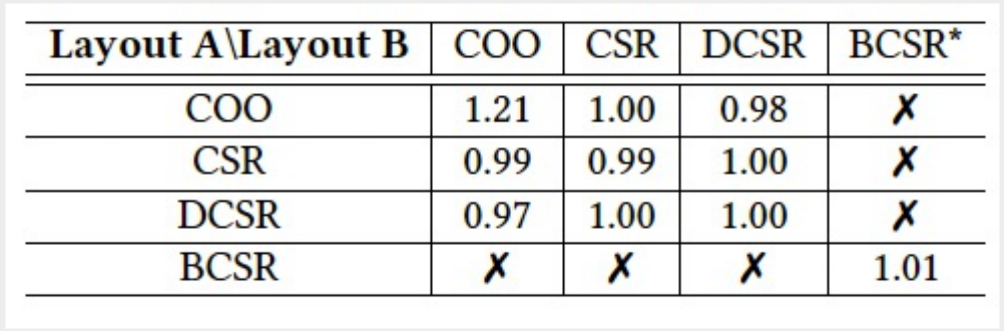
Sparse Tensors: Performance Comparison with State-of-the-Art

SpMSpV



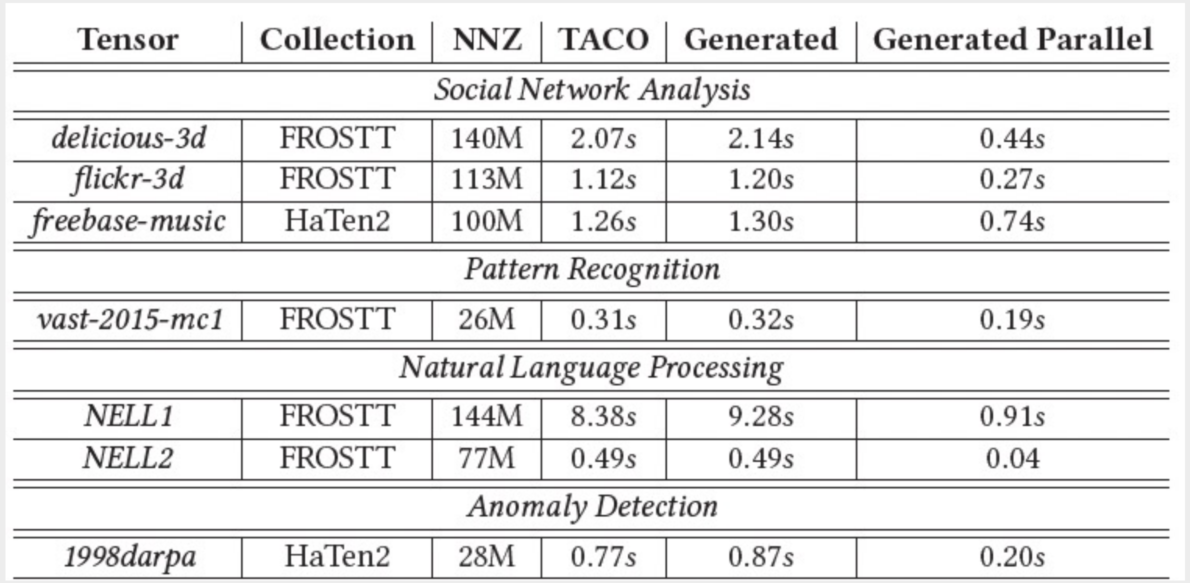
	TACO	Eigen	SS:GB
SeqIter	1.63	2.43	1.50
HashMap	1.63	2.44	1.50
Auto	2.72	4.06	2.50

SpMSpM (Comparison with TACO Compiler)



Layout A\Layout B	COO	CSR	DCSR	BCSR*
COO	1.21	1.00	0.98	✗
CSR	0.99	0.99	1.00	✗
DCSR	0.97	1.00	1.00	✗
BCSR	✗	✗	✗	1.01

TTM (large # dims)



Tensor	Collection	NNZ	TACO	Generated	Generated Parallel
Social Network Analysis					
delicious-3d	FROSTT	140M	2.07*s*	2.14*s*	0.44*s*
flickr-3d	FROSTT	113M	1.12*s*	1.20*s*	0.27*s*
freebase-music	HaTen2	100M	1.26*s*	1.30*s*	0.74*s*
Pattern Recognition					
vast-2015-mc1	FROSTT	26M	0.31*s*	0.32*s*	0.19*s*
Natural Language Processing					
NELL1	FROSTT	144M	8.38*s*	9.28*s*	0.91*s*
NELL2	FROSTT	77M	0.49*s*	0.49*s*	0.04
Anomaly Detection					
1998darpa	HaTen2	28M	0.77*s*	0.87*s*	0.20*s*

T. HOEFLER

AI for HPC panel

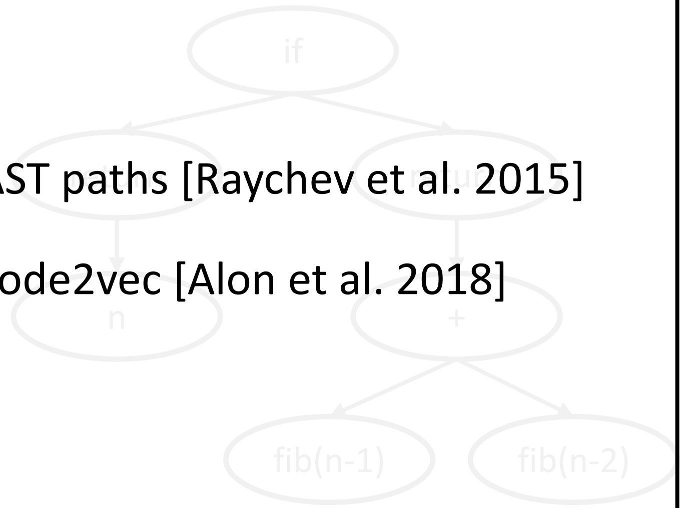
with contributions by the whole SPCL deep learning team (T. Ben-Nun, S. Li, K. Osawa, N. Dryden and many others)

AI for HPC – what does that mean?

- **Talked to several people – some confusion with**
  - AI for Science
  - AI for Performance
  - AI for Networking
  - AI for Programming
  - AI for Simulation
- **So what is this “HPC” really?**
  - Let’s not get into a discussion now – maybe on the panel
- **I decided to go with programming and program analysis**
  - Open for other interpretations

Representations of code

Source Code	Abstract Syntax Tree (AST)	Static Single Assignment (SSA)	Assembly
DeepTune [Cummins et al. 2017]	AST paths [Raychev et al. 2015] code2vec [Alon et al. 2018]	inst2vec (LLVM) [Ben-Nun et al. 2018] IR2Vec (LLVM) [Keerthy et al. 2019] CDFG (LLVM) [Brauckmann et al. 2020] ProGraML (LLVM, XLA) [Cummins et al. 2020]	[Le et al. 2018]



V. Raychev et al., "Predicting Program Properties from 'Big Code'" POPL 2015.
C. Cummins et al., "End-to-end Deep Learning of Optimization Heuristics", PACT 2017.
U. Alon et al., "code2vec: Learning Distributed Representations of Code", POPL 2018.
T. Ben-Nun et al., "Neural Code Comprehension: A Learnable Representation of Code Semantics", NeurIPS 2018.
Q. Le et al., "Deep learning at the shallow end: Malware classification for non-domain experts", DFRWS 2018.
V. Keerthy et al., "IR2Vec: A Flow Analysisbased Scalable Infrastructure for Program Encodings", arXiv 2019.
A. Brauckmann et al., "Compiler-Based Graph Representations for Deep Learning Models of Code", CC 2020.
C. Cummins et al., "ProGraML: Graph-based Deep Learning for Program Optimization and Analysis", arXiv 2020.

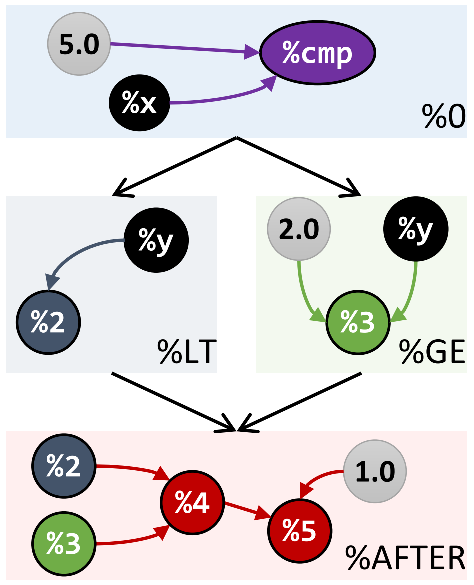
Neural Code Comprehension – inst2vec (2018)

- **In 2021, GitHub reports >1 billion git commits in 337 languages!**
- **Can DNNs *understand* code?**
- **Previous approaches read the code directly → suboptimal (loops, functions)**

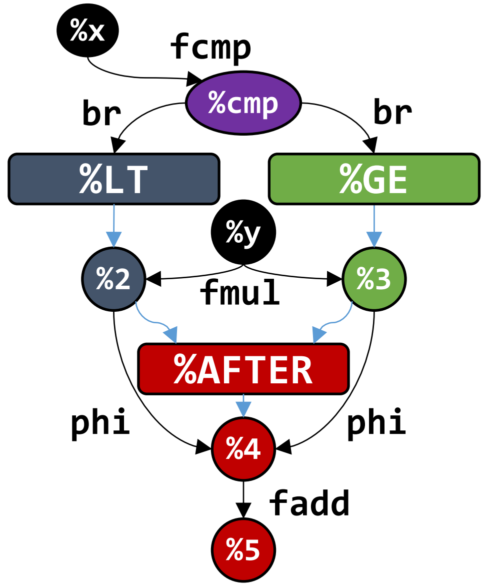
```
double thres = 5.0;

if (x < thres)
    x = y * y;
else
    x = 2.0 * y;


x += 1.0;
```

```
%cmp = fcmp olt double %x, 5.0

br i1 %cmp, label %LT, label %GE
LT:
  %2 = fmul double %y, %y
GE:
  %3 = fmul double 2.0, %y


AFTER:
  %4 = phi double [%2,%LT], [%3,%GE]
  %5 = fadd double %4, 1.0
```

C/C++ FORTRAN
Python Java
CUDA OpenCL
. . .



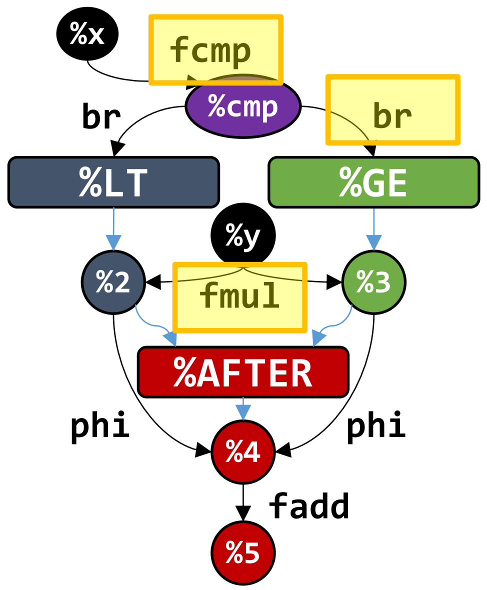
Dataflow (basic blocks)

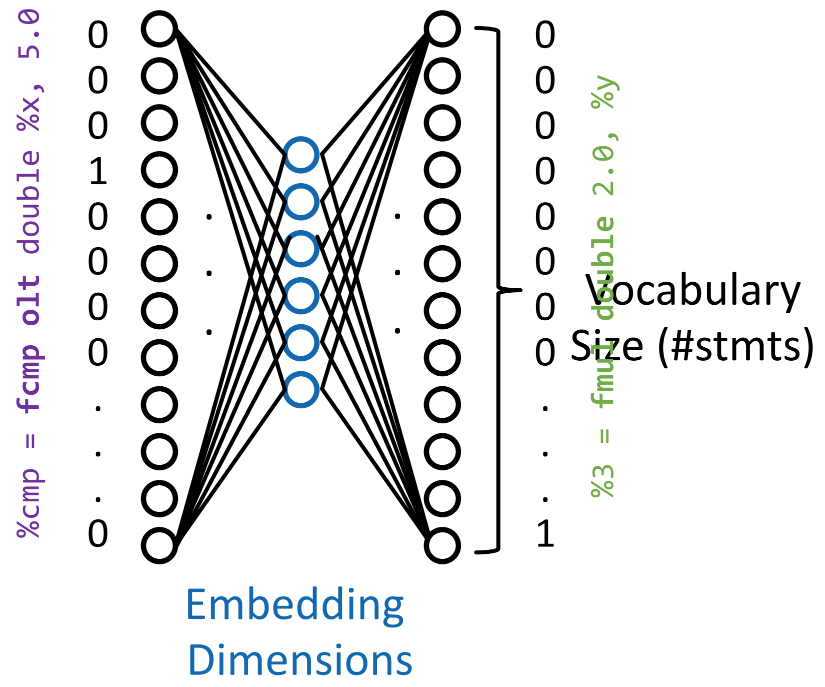


conteXtual Flow Graph

Neural Code Comprehension – inst2vec (2018)

- Embedding space (using the Skip-gram model)





Neural Code Comprehension – inst2vec (2018)

Table 3: Algorithm classification test accuracy

Metric	Surface Features [46] (RBF SVM + Bag-of-Trees)	RNN [46]	TBCNN [46]	inst2vec
Test Accuracy [%]	88.2	84.8	94.0	**94.83**

Predicts which device is faster (CPU or GPU)

Table 4: Heterogeneous device mapping results

Architecture	Prediction Accuracy [%]			Speedup		
	Grewe et al. [27]	DeepTune [17]	inst2vec	Grewe et al.	DeepTune	inst2vec
AMD Tahiti 7970	73.38	**83.68**	82.79	2.91	3.34	**3.42**
NVIDIA GTX 970	72.94	80.29	**81.76**	1.26	**1.41**	1.39

Optimal tiling

Table 5: Speedups achieved by coarsening threads

Computing Platform	Magni et al. [43]	DeepTune [17]	DeepTune-TL [17]	inst2vec
AMD Radeon HD 5900	1.21	1.10	1.17	**1.25**
AMD Tahiti 7970	1.01	1.05	**1.23**	1.07
NVIDIA GTX 480	0.86	1.10	**1.14**	1.02
NVIDIA Tesla K20c	0.94	0.99	0.93	**1.03**

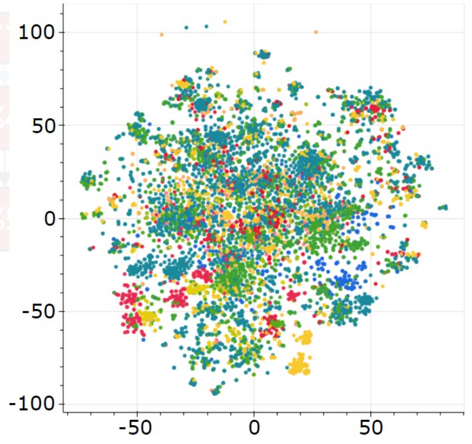
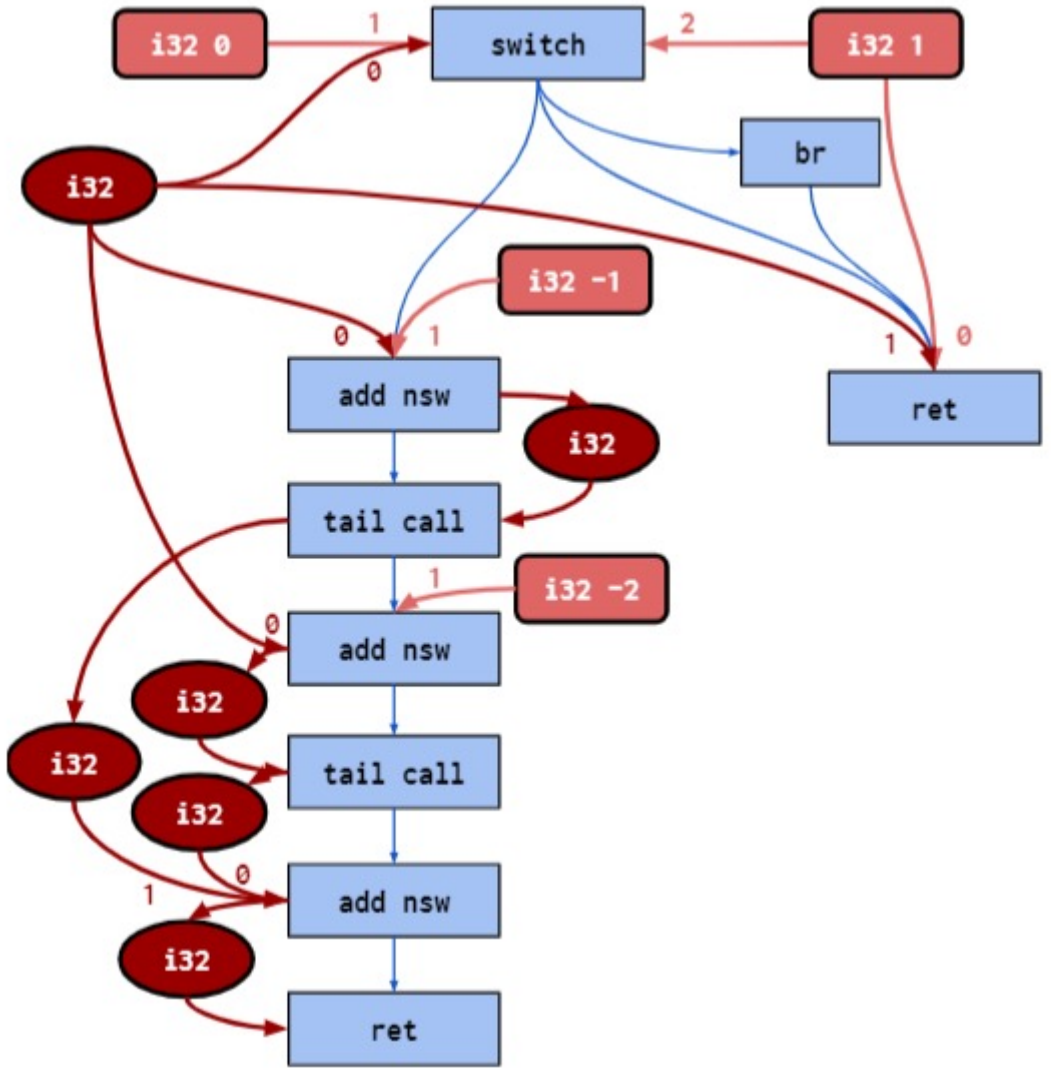
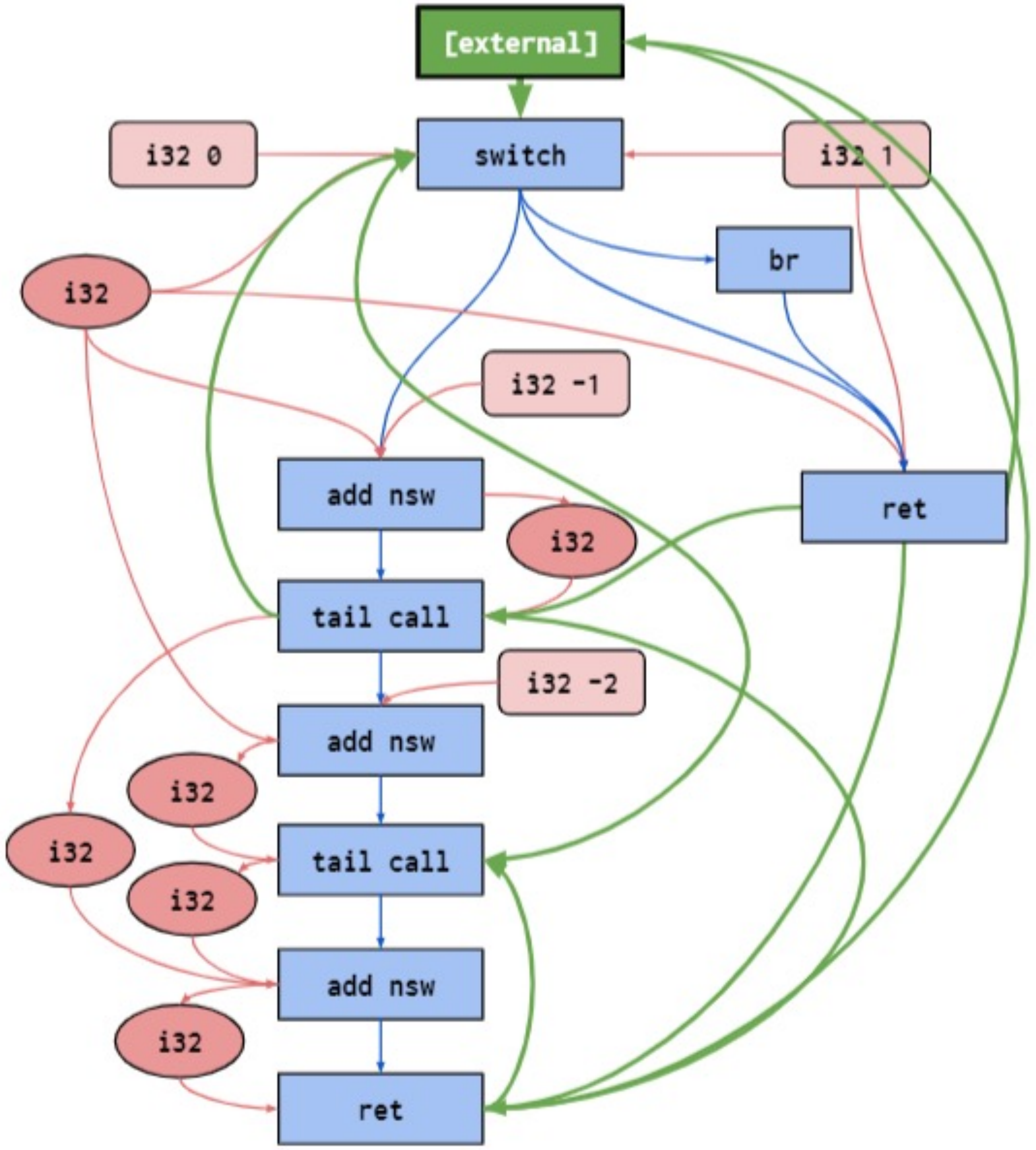


Table 2: Analogy and test scores for `inst2vec`

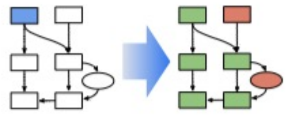
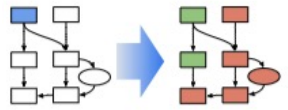
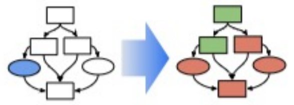
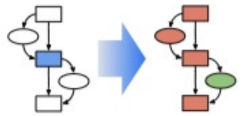
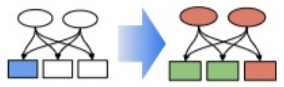
Context Size	Syntactic Analogies		Semantic Analogies		Semantic Distance Test
	Types	Options	Conversions	Data Structures	
1	101 (18.04%)	13 (24.53%)	100 (6.63%)	3 (37.50%)	60.98%
2	**226 (40.36%)**	**45 (84.91%)**	**134 (8.89%)**	**7 (87.50%)**	**79.12%**
3	125 (22.32%)	24 (45.28%)	48 (3.18%)	**7 (87.50%)**	62.56%

ProGraML – using the full graph structure with Graph Neural Networks





ProGraML on compiler tasks

Problem	Analysis type	Example optimization	Model	Precision	Recall	F_1
REACHABILITY		Dead code elimination	$\text{DeepTune}_{\text{IR}}$	0.520	0.497	0.504
			ProGraML	**0.997**	**0.995**	**0.996**
DOMTREE		Global Code Motion	$\text{DeepTune}_{\text{IR}}$	0.721	0.081	0.138
			ProGraML	**0.985**	**0.693**	**0.781**
DATADEP		Instruction scheduling	$\text{DeepTune}_{\text{IR}}$	0.999	0.136	0.236
			ProGraML	**1.000**	**0.988**	**0.993**
LIVENESS		Register allocation	$\text{DeepTune}_{\text{IR}}$	—	—	—
			ProGraML	**1.000**	**0.999**	**0.999**
SUBEXPRESSIONS		Global Common Subexpression Elimination	$\text{DeepTune}_{\text{IR}}$	**1.000**	0.123	0.214
			ProGraML	0.965	**0.925**	**0.930**
Average	—	—	$\text{DeepTune}_{\text{IR}}$	0.810	0.209	0.273
			ProGraML	**0.989**	**0.920**	**0.940**

SC22

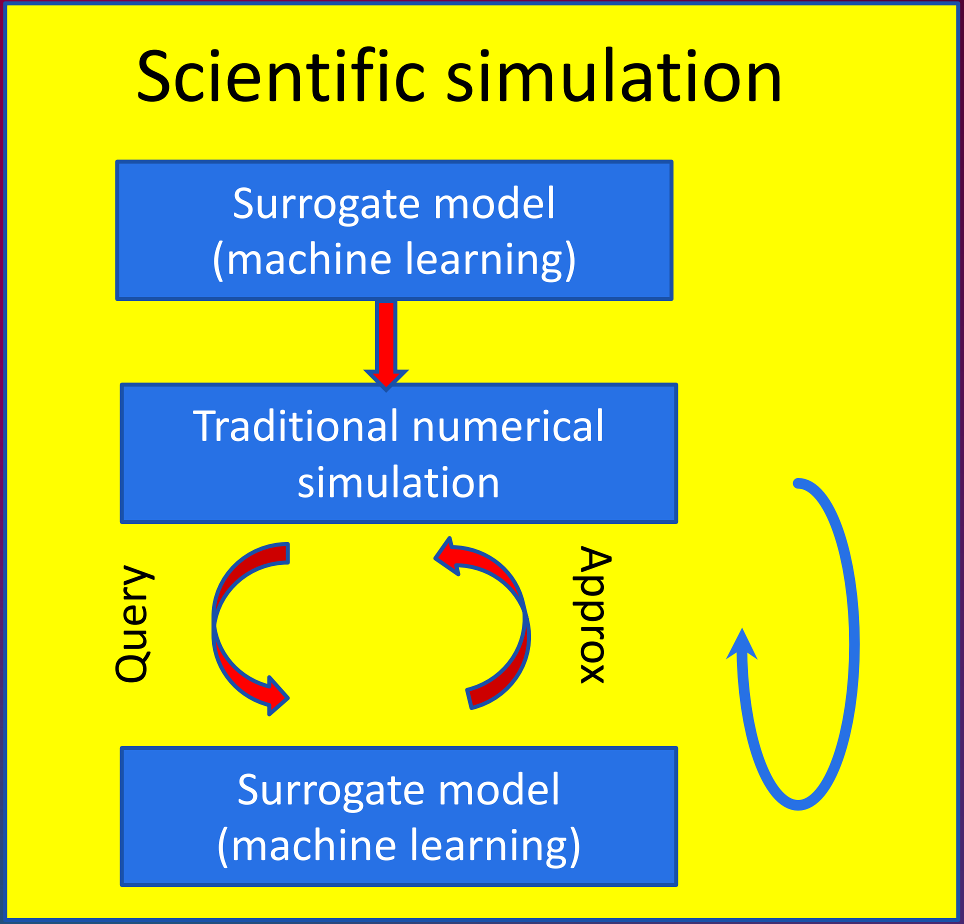
Dallas, TX | hpc accelerates.

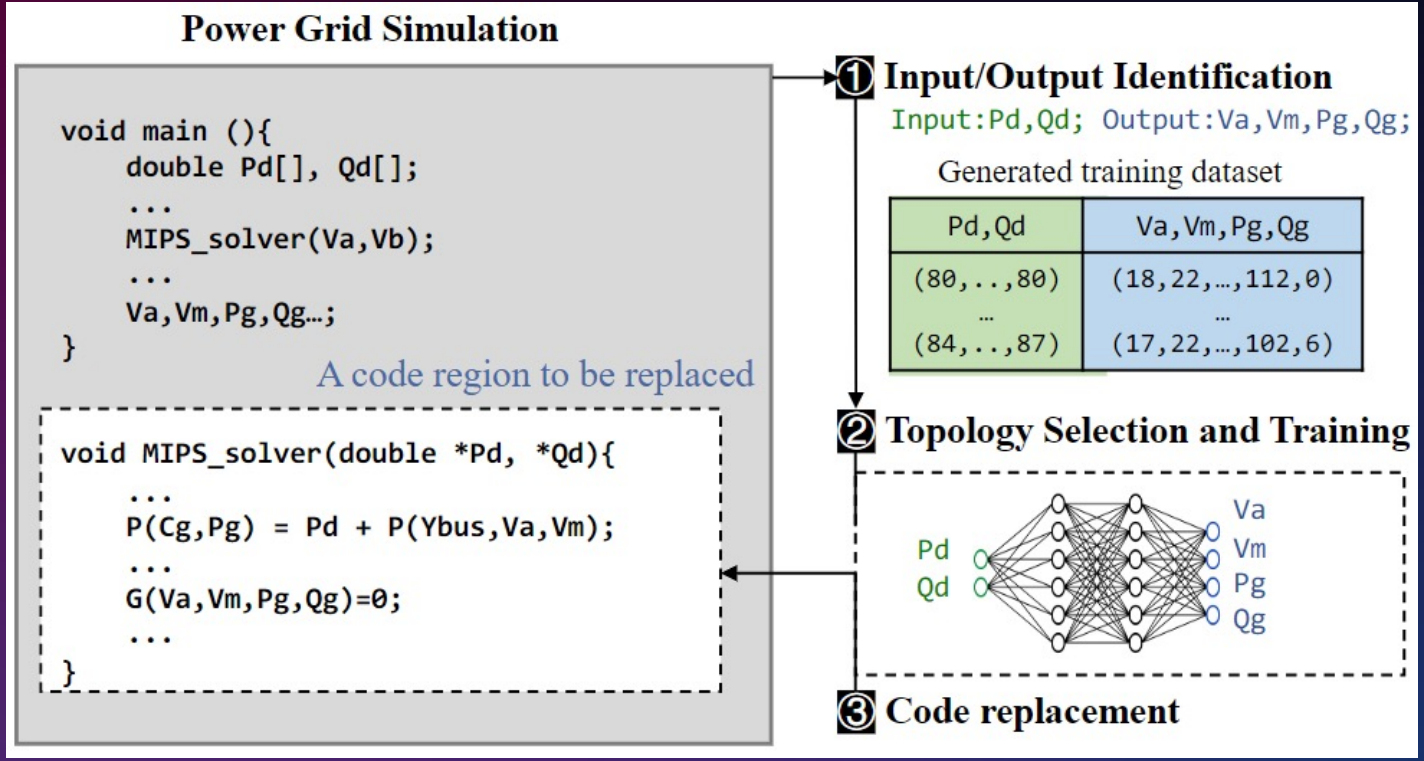
AI for HPC

Seventh International Workshop on Extreme Scale Programming Models and Middleware

Panelist: Dong Li
University of California, Merced

Using machine learning-based surrogate models to accelerate HPC applications


Scientific simulation
Surrogate model (machine learning)
Traditional numerical simulation
Query
Approx
Surrogate model (machine learning)


Power Grid Simulation
A code region to be replaced
① Input/Output Identification
Generated training dataset
② Topology Selection and Training
Pd
Qd
Va
Vm
Pg
Qg
③ Code replacement

```
void main (){
    double Pd[], Qd[];
    ...
    MIPS_solver(Va,Vb);
    ...
    Va,Vm,Pg,Qg…;
}
```

```
void MIPS_solver(double *Pd, *Qd){
    ...
    P(Cg,Pg) = Pd + P(Ybus,Va,Vm);
    ...
    G(Va,Vm,Pg,Qg)=0;
    ...
}
```

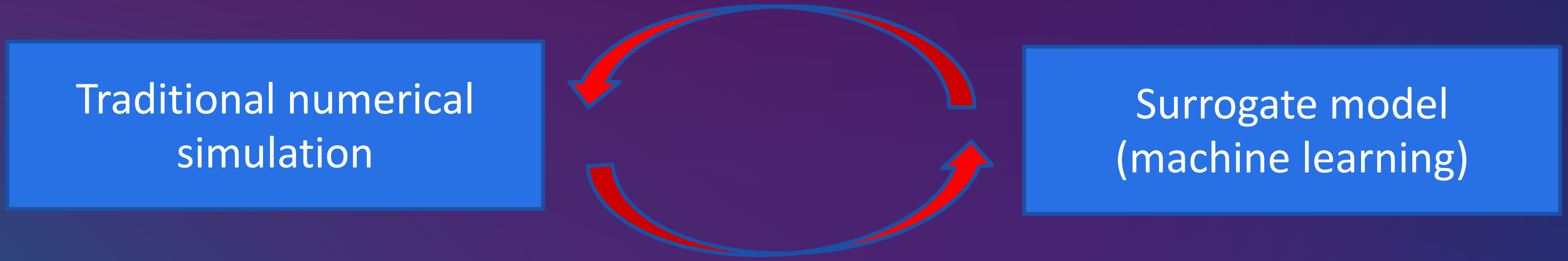
```
Input:Pd,Qd; Output:Va,Vm,Pg,Qg;
```

Pd,Qd	Va,Vm,Pg,Qg
(80,..,80)	(18,22,…,112,0)
…	…
(84,..,87)	(17,22,…,102,6)

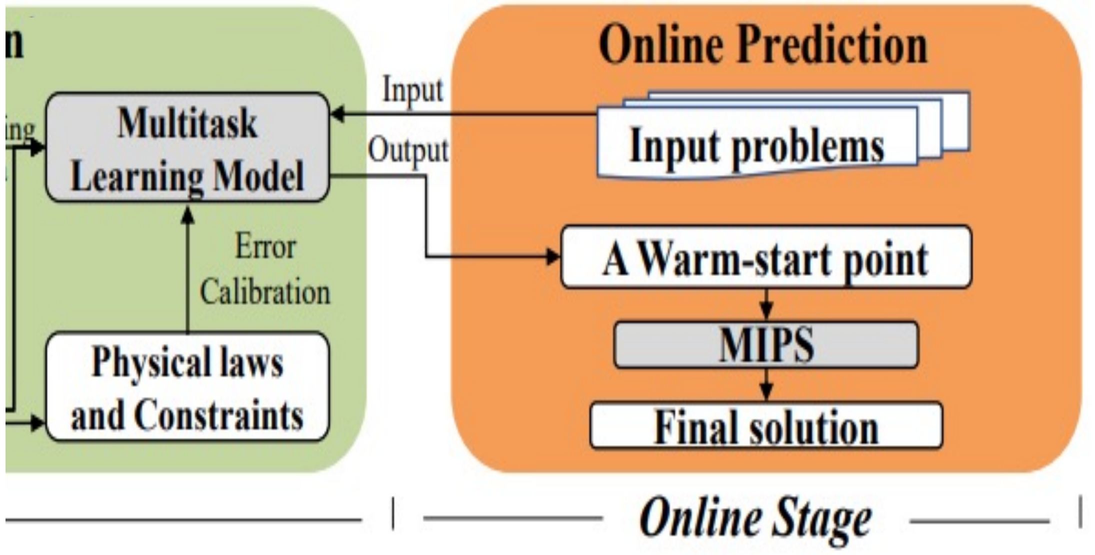
An example of applying the surrogate model

Benefits of using machine learning-based surrogate models

- Increase code portability
- Remove some performance problems in the original code
- Leverage AI-specific accelerators
- Potentially huge performance improvement

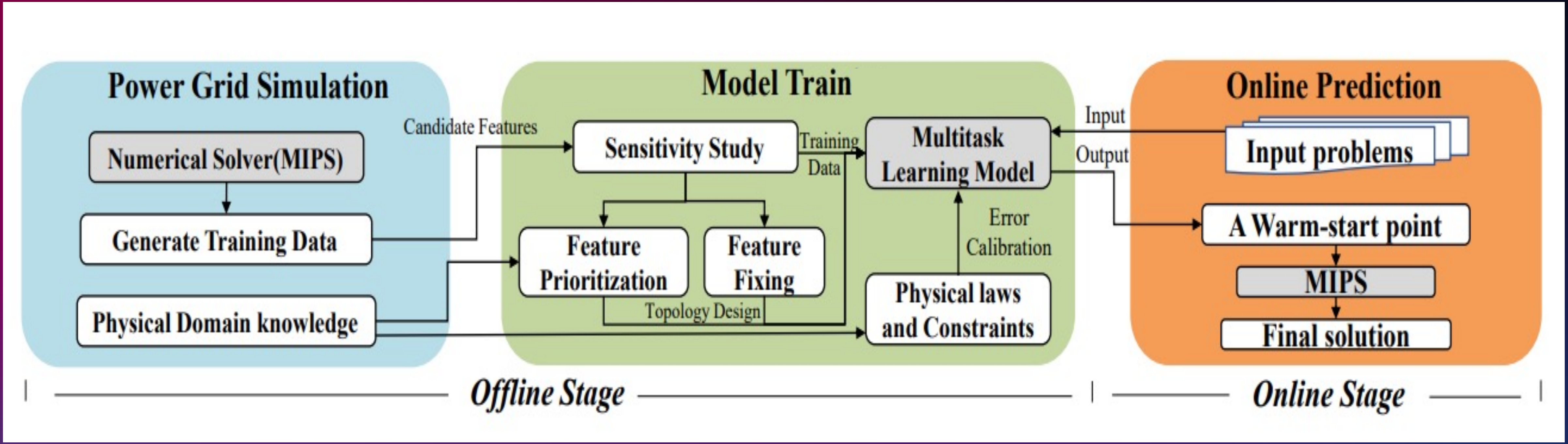

Traditional numerical simulation
Surrogate model (machine learning)

Smart-PGSim: using neural network to accelerate AC-OPF power grid simulation (SC'20)



Workflow of Smart-PGSim

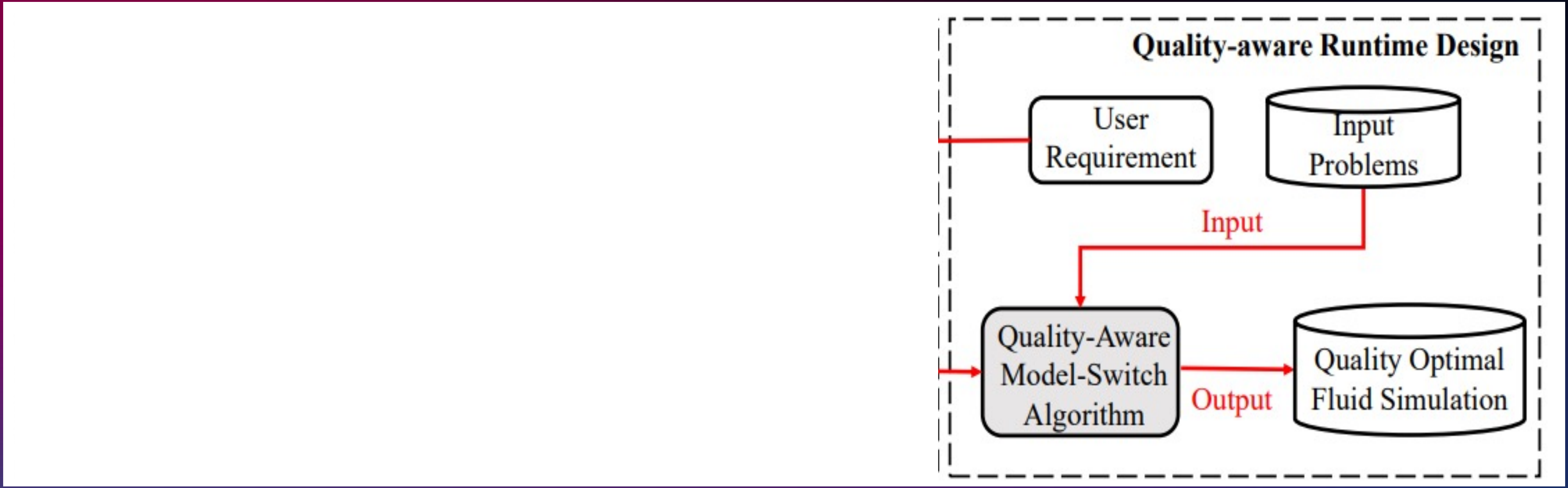
Smart-PGSim: using neural network to accelerate AC-OPF power grid simulation (SC'20)



Workflow of Smart-PGSim

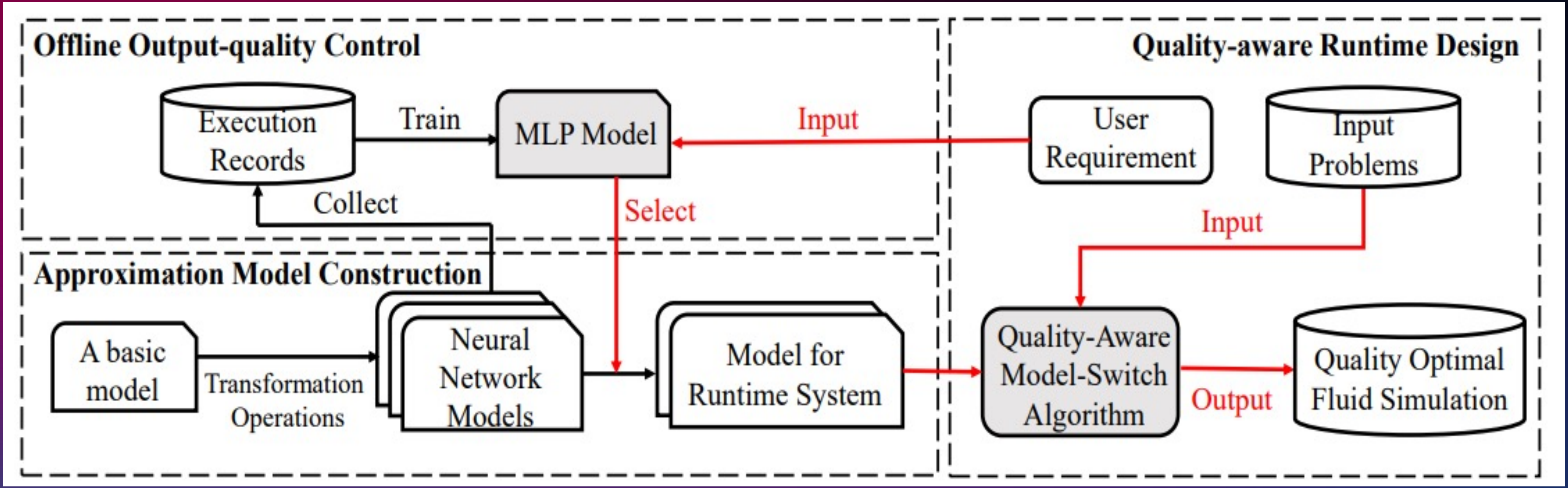
2.60× speedup on average (up to 3.28×) computed over 10,000 problems, without losing solution optimality.

Smart-fluidnet: adaptive neural network-based approximation to accelerate Eulerian fluid simulation (SC'19)


Quality-aware Runtime Design
User Requirement
Input Problems
Input
Quality-Aware Model-Switch Algorithm
Output
Quality Optimal Fluid Simulation

Workflow of Smart-fluidnet

Smart-fluidnet: adaptive neural network-based approximation to accelerate Eulerian fluid simulation (SC'19)


Offline Output-quality Control
Execution Records
Train
MLP Model
Input
Collect
Select
Quality-aware Runtime Design
User Requirement
Input Problems
Input
Approximation Model Construction
A basic model
Transformation Operations
Neural Network Models
Model for Runtime System
Quality-Aware Model-Switch Algorithm
Output
Quality Optimal Fluid Simulation

Workflow of Smart-fluidnet

590x speedup over the original fluid simulation without losing simulation quality

Challenge: Automate the deployment and construction of surrogate models

- Automatically extract input features out of the original code
  - Compiler analysis
- Automatically decide mode topology
  - Based upon AutoML tools (e.g., autokeras)
- Model quality control
  - Need a user-specified metric

AI for HPC

Vipin Chaudhary

Case Western Reserve University

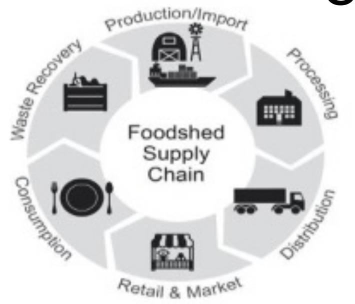
vipin@case.edu

ICICLE

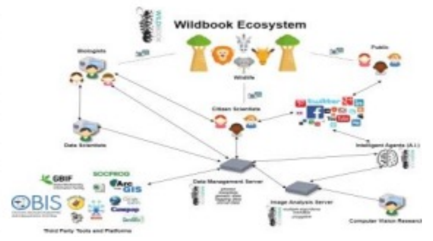
ESPM2 2022: Seventh International Workshop on Extreme Scale Programming Models and Middleware

ICICLE: NSF AI Research Institute

Use Inspired Science Domains



Smart Foodsheds



Animal Ecology

Digital Agriculture

ICICLE: Intelligent CyberInfrastructure with Computational Learning in the Environment

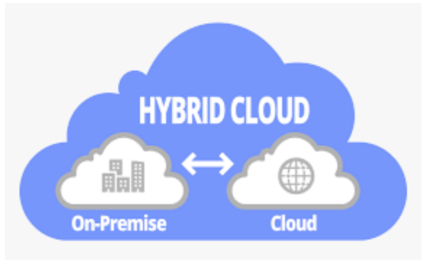
Systems AI Foundational Research for CI

Intelligent Cyber Infrastructure

CI for AI | AI for "CI for AI"

On Field Sensors

Edge & Near Edge



Clouds

HPC Systems & Data Centers

Emerging Computing Continuum

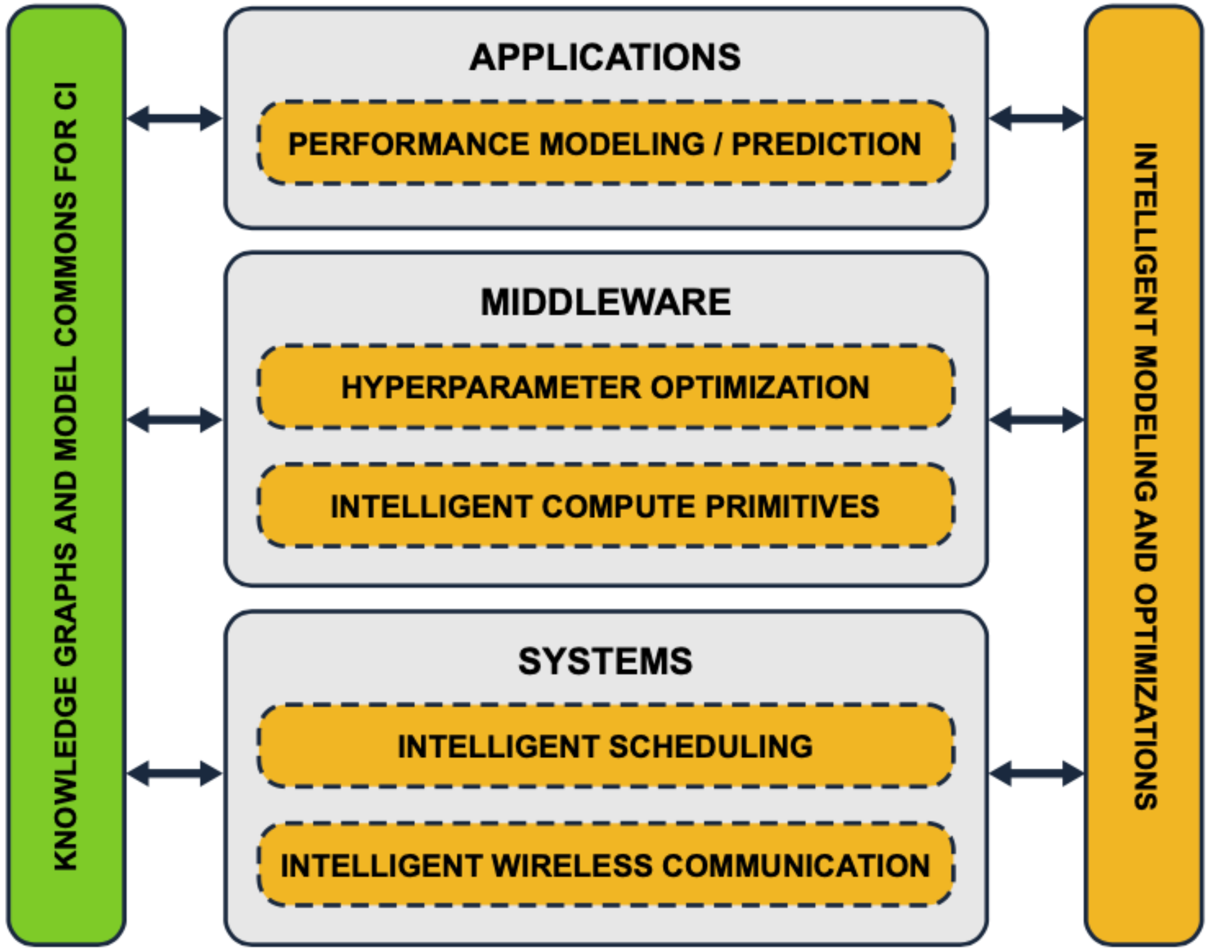
Integrating a broad range of

- Scientists-in-the-field
- Engineers
- Educators
- Collaborative partners
- Institutions

under one roof enables **democratized, adaptable, plug-and-play AI** and **long-tail science.**

AI for HPC

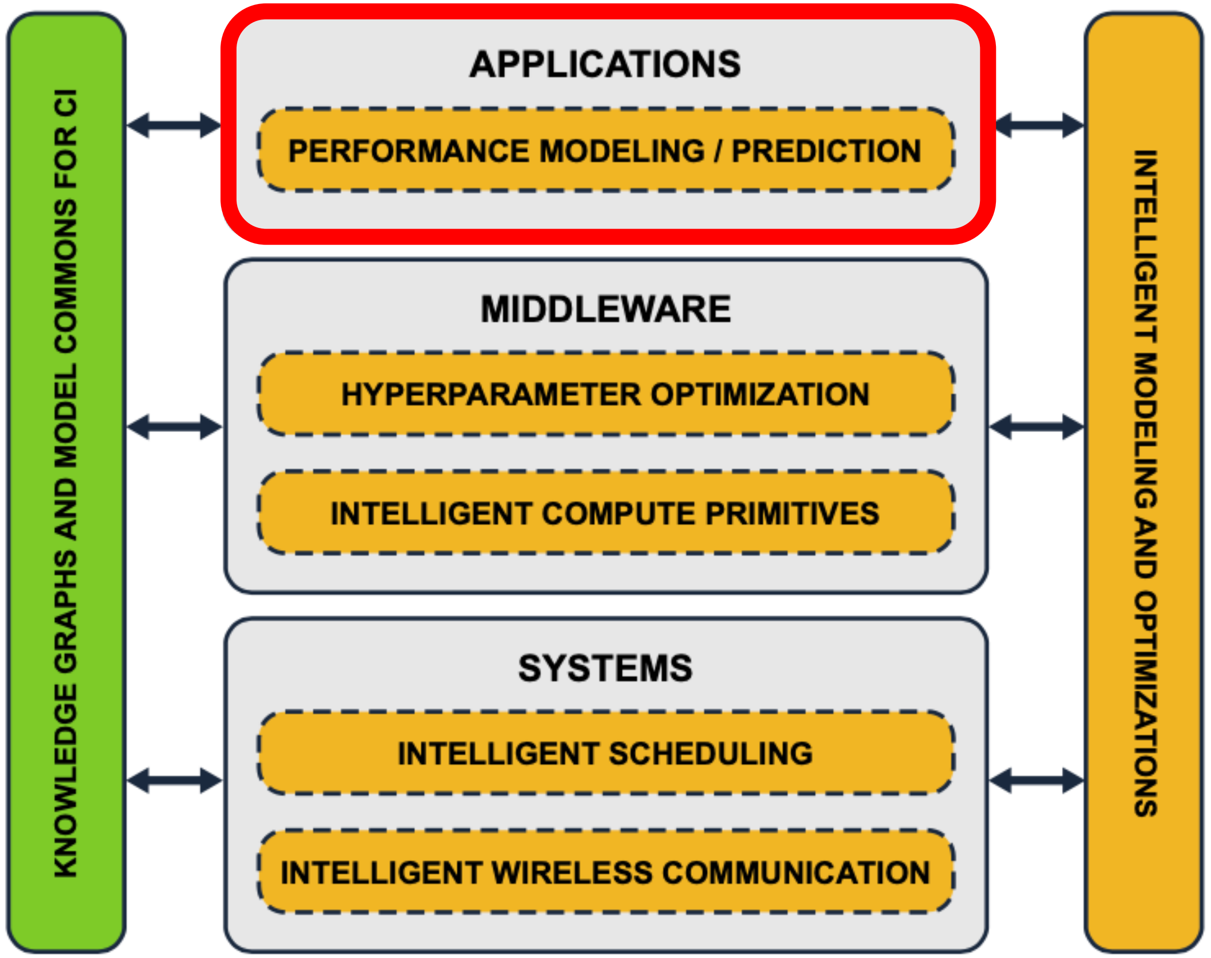
- ***Efficient*** **plug-n-play:** Constantly adapt and optimize the heterogenous (cloud, HPC, and edge) CI to meet requirements of ICICLE applications including digital agriculture and wildlife detection


KNOWLEDGE GRAPHS AND MODEL COMMONS FOR CI
APPLICATIONS
PERFORMANCE MODELING / PREDICTION
MIDDLEWARE
HYPERPARAMETER OPTIMIZATION
INTELLIGENT COMPUTE PRIMITIVES
SYSTEMS
INTELLIGENT SCHEDULING
INTELLIGENT WIRELESS COMMUNICATION
INTELLIGENT MODELING AND OPTIMIZATIONS

AI4CI: Applications

- ***Efficient* plug-n-play:** Constantly adapt and optimize the heterogenous (cloud, HPC, and edge) CI to meet requirements of ICICLE applications including digital agriculture and wildlife detection

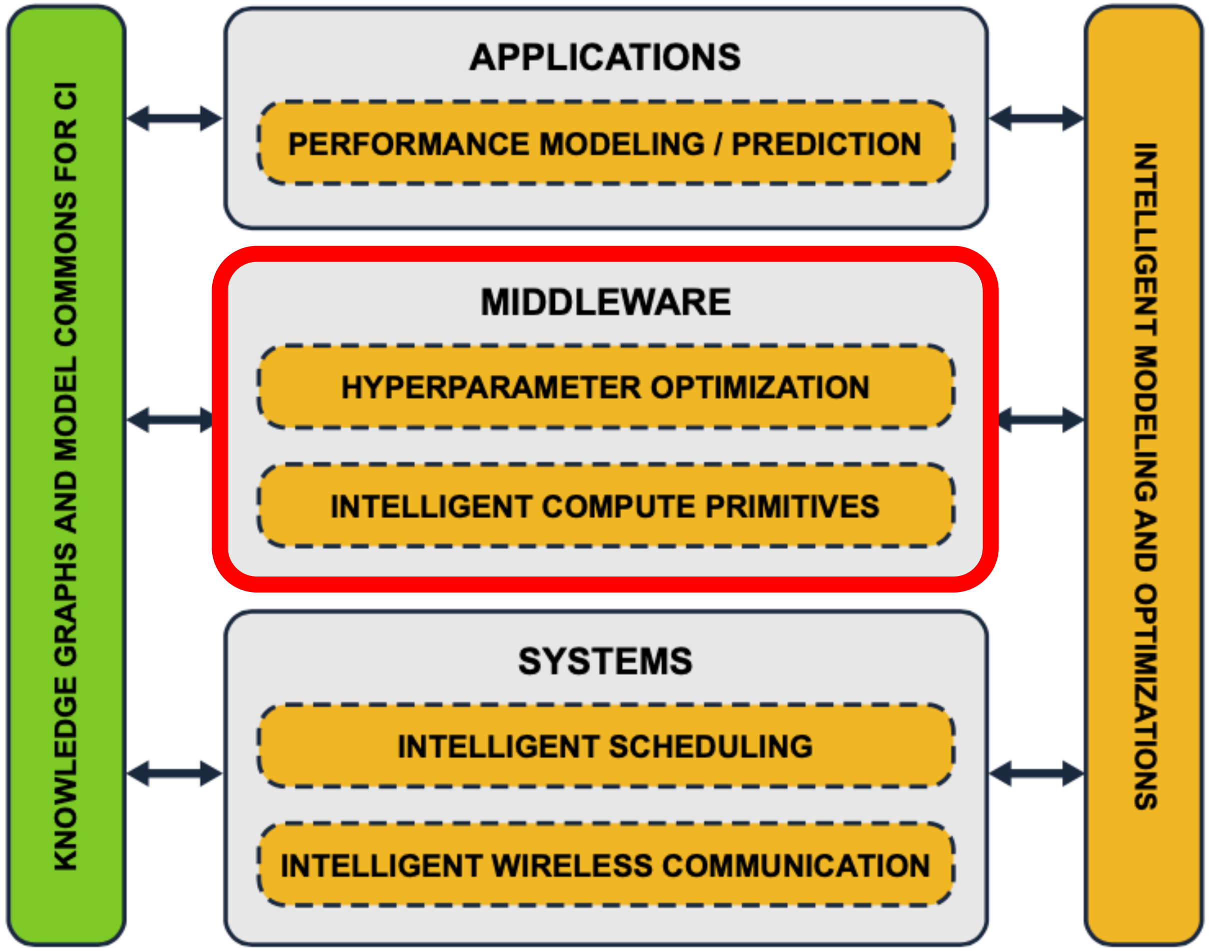
- Develop data-driven approaches to model application performance:
  - Agnostic of the execution environment
  - Guide modeling of other CI components


KNOWLEDGE GRAPHS AND MODEL COMMONS FOR CI
APPLICATIONS
PERFORMANCE MODELING / PREDICTION
MIDDLEWARE
HYPERPARAMETER OPTIMIZATION
INTELLIGENT COMPUTE PRIMITIVES
SYSTEMS
INTELLIGENT SCHEDULING
INTELLIGENT WIRELESS COMMUNICATION
INTELLIGENT MODELING AND OPTIMIZATIONS

AI4CI: Middleware

- ***Efficient* plug-n-play:** Constantly adapt and optimize the heterogenous (cloud, HPC, and edge) CI to meet requirements of ICICLE applications including digital agriculture and wildlife detection

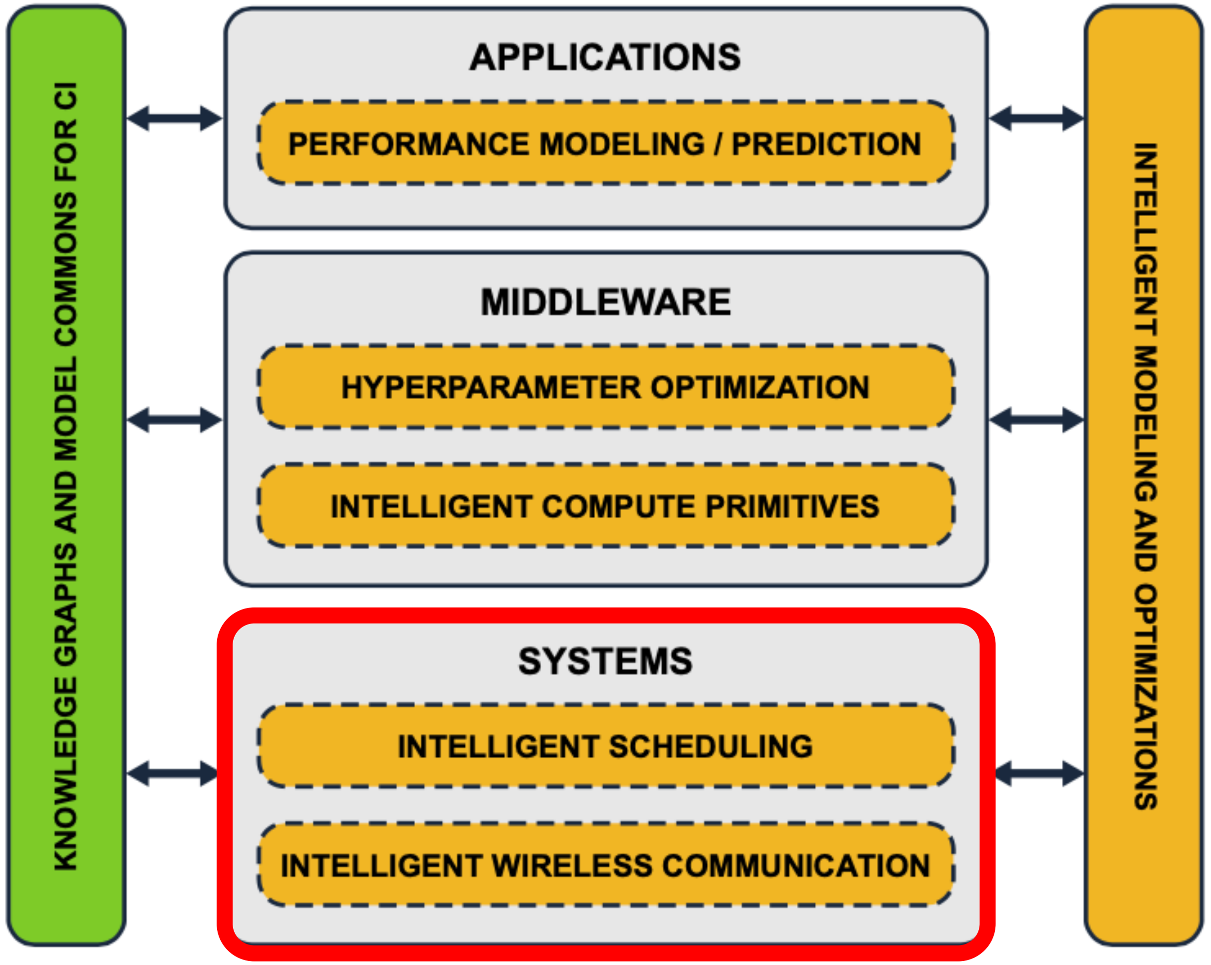
- Self-driving middleware for AI frameworks:
  - Optimize end-to-end AI workloads
  - Hyperparameter optimization
  - Compiler optimizations
  - Matrix operations


KNOWLEDGE GRAPHS AND MODEL COMMONS FOR CI
APPLICATIONS
PERFORMANCE MODELING / PREDICTION
MIDDLEWARE
HYPERPARAMETER OPTIMIZATION
INTELLIGENT COMPUTE PRIMITIVES
SYSTEMS
INTELLIGENT SCHEDULING
INTELLIGENT WIRELESS COMMUNICATION
INTELLIGENT MODELING AND OPTIMIZATIONS

AI4CI: Systems

- ***Efficient* plug-n-play:** Constantly adapt and optimize the heterogenous (cloud, HPC, and edge) CI to meet requirements of ICICLE applications including digital agriculture and wildlife detection

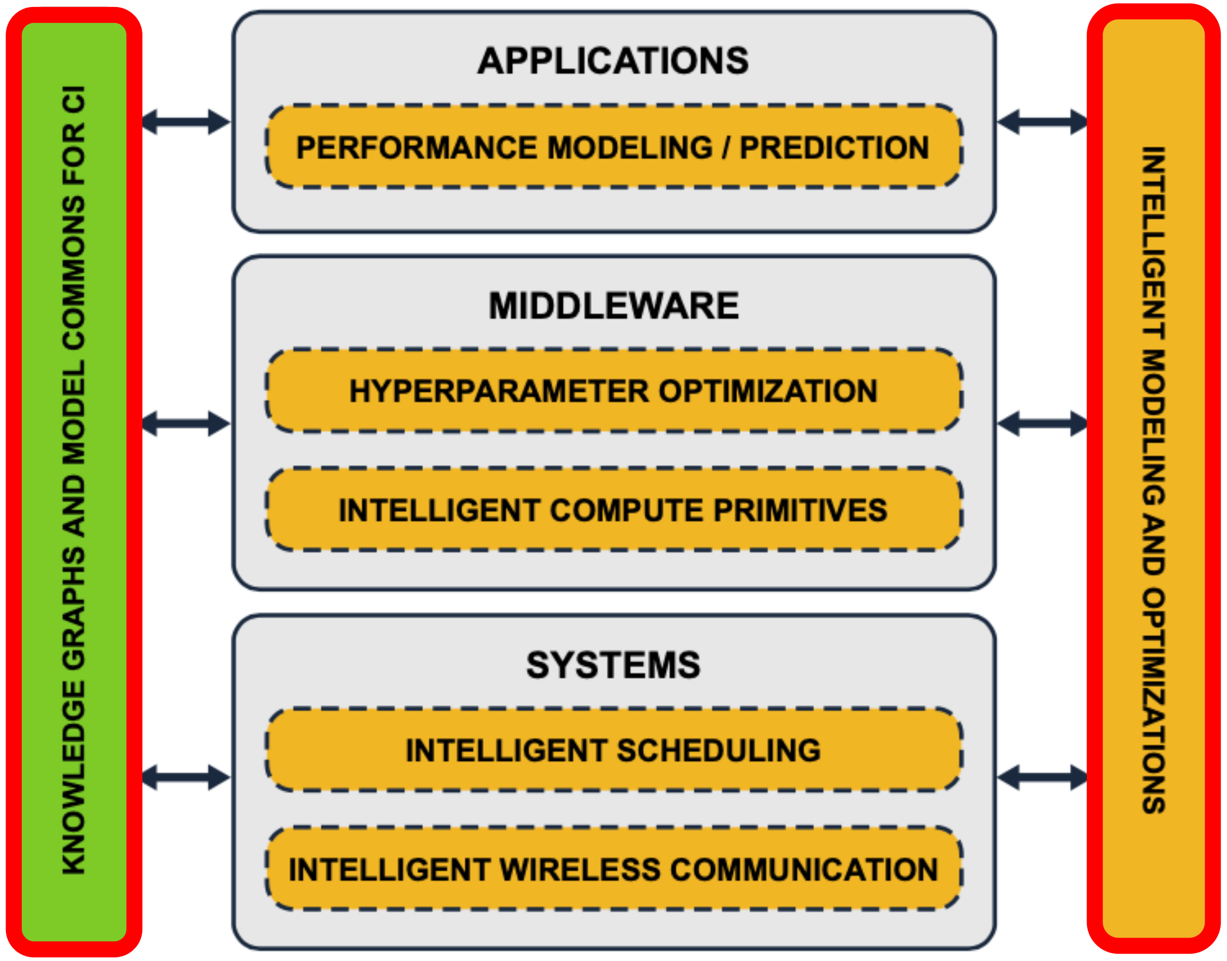
- Resource management and scheduling for heterogenous CI:
  - Reinforcement learning
- Transient computing:
  - Reduce cost and democratize computing resources
- Intelligent wireless communication


KNOWLEDGE GRAPHS AND MODEL COMMONS FOR CI
APPLICATIONS
PERFORMANCE MODELING / PREDICTION
MIDDLEWARE
HYPERPARAMETER OPTIMIZATION
INTELLIGENT COMPUTE PRIMITIVES
SYSTEMS
INTELLIGENT SCHEDULING
INTELLIGENT WIRELESS COMMUNICATION
INTELLIGENT MODELING AND OPTIMIZATIONS

AI4CI: Cross Stack Layers

- ***Efficient* plug-n-play:** Constantly adapt and optimize the heterogenous (cloud, HPC, and edge) CI to meet requirements of ICICLE applications including digital agriculture and wildlife detection

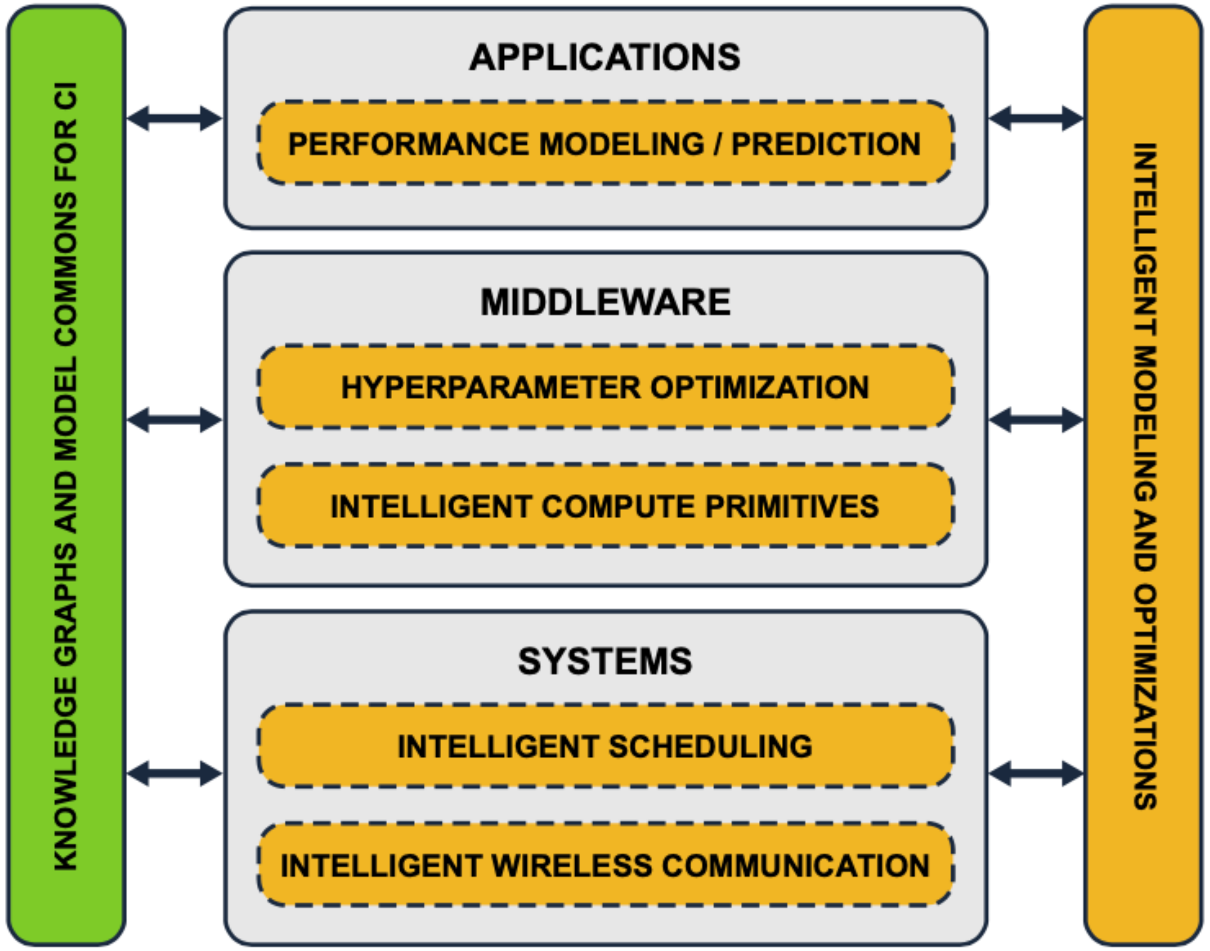
- Two cross-cutting components
- New mechanisms needed for collecting and distributing data to the AI4CI stack:
  - KGs and Model Commons for CI
- ML models for predicting performance throughout the AI4CI stack


KNOWLEDGE GRAPHS AND MODEL COMMONS FOR CI
APPLICATIONS
PERFORMANCE MODELING / PREDICTION
MIDDLEWARE
HYPERPARAMETER OPTIMIZATION
INTELLIGENT COMPUTE PRIMITIVES
SYSTEMS
INTELLIGENT SCHEDULING
INTELLIGENT WIRELESS COMMUNICATION
INTELLIGENT MODELING AND OPTIMIZATIONS

AI for CI

- ***Efficient* plug-and-play:** Constantly adapt and optimize the heterogenous CI using AI techniques
- Enhance performance, scalability, and manageability *synergistically* across the CI stack

- KGs and model commons for CI:
  - Build KGs from CI data from existing infrastructure (XD-MoD, TACC Stats, OSU INAM, application logs)
- Applications:
  - Application profiles are used to create KGs that enable performance prediction by modeling
- Middleware:
  - Self-driving middleware: Jointly optimize configuration knobs of AI frameworks, hyperparameter optimization, resource allocation
  - ML-based compiler, matrix operation optimization
- Systems:
  - Reinforcement Learning (RL) based schedulers for heterogeneous applications and CI
  - Joint transmission and scheduling for IoT devices
- Explore optimizations to the edge CI


KNOWLEDGE GRAPHS AND MODEL COMMONS FOR CI
APPLICATIONS
PERFORMANCE MODELING / PREDICTION
MIDDLEWARE
HYPERPARAMETER OPTIMIZATION
INTELLIGENT COMPUTE PRIMITIVES
SYSTEMS
INTELLIGENT SCHEDULING
INTELLIGENT WIRELESS COMMUNICATION
INTELLIGENT MODELING AND OPTIMIZATIONS

Concluding Remarks

Concluding Remarks

- Thanks again to Keynote Speaker, Invited Speakers, Panelists, Panel Moderator, Authors, PC Members, and Attendees!!
- Presentations are being linked to the Website
  - Speakers: please send us your pdfs
- Plan to continue this workshop in conjunction with SC ’23.
- Plan to submit a proposal to SC ‘23 Workshop Committee.

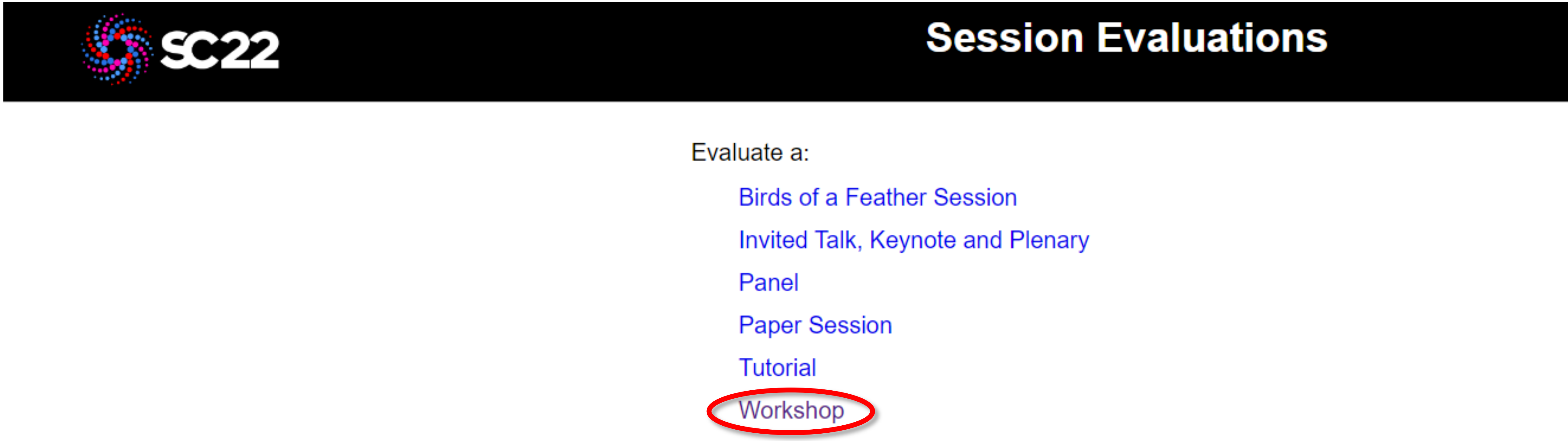
Concluding Remarks (Cont'd)

- Looking forward to feedback and comments
- Let us know if you would like to be involved in this workshop for future years
- Send us an e-mail:

panda@cse.ohio-state.edu
shafi.16@osu.edu
subramon@cse.ohio-state.edu
karl.schulz@amd.com
abduljabbar.1@osu.edu

Workshop evaluation

- Please submit your surveys at the following link
- https://submissions.supercomputing.org/eval.html



SC22 Session Evaluations

Evaluate a:

- Birds of a Feather Session
- Invited Talk, Keynote and Plenary
- Panel
- Paper Session
- Tutorial
- Workshop