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
Vipin Chaudhary, CRWU
Torsten Hoefler, ETH
Dong Li, UC Merced
### Agenda

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Permute to innermost position the loop that carries the most reuse</td>
<td>Evaluate data’s cache footprint for different loop orders, minimize memory cost</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enabled by Exploration-Enabled Compiler Structure
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 1.5X speedup 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: \( v() = A(i) \times B(i) \)

Physical Layout:

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

Physical to Logical Mapping:

\[
IS = \{ [pA, pB, i] | A.\text{idx}(pA) = i = B.\text{idx}(pB) \land 0 \leq pA < A.\text{len} \land 0 \leq pB < B.\text{len} \}\]

Co-Iteration Code:

```c
1  pB = 0;
2  for (int pA = 0; pA < A.len; ++pA) {
3      i = A.idx[pA];
4      while (pB < B.len && i > B.idx[pB]) ++pB;
5      if (pB < B.len && i == B.idx[pB])
6        { v += A.val[pA] \times B.val[pB]; ++pB; }
7  }
```

Iterate over nonzeros in A

if \((pb = \text{find}(i, B))\)
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

References


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


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


Backup Slides
Restructure Compiler for Exploration

Late 1990s-2000
- Autotuning Libraries automatically explore a search space (PhiPAC, ATLAS)

2005
- X Language exposes xforms to programmer (Donadio et al.)

2007 Chen, ChiLL Compiler (from LCPC 2009)

2007-present
- Halide
  ```
  vectorize(x_inner, factor), equivalent to
  gradient-split(x, x_out, x_inner, 4);
  gradient-vectorize(x_inner);
  gradient-parallel(tile_index);
  gradient-split(x, x_out, x_inner, 2);
  gradient-unroll(x_inner), equivalent to
  gradient-unroll(s, 2);
  gradient-tile(x, y, x_out, y_out, x_inner, y_inner, 4, 4);
  gradient-reorder(y, x); // similar to transpose
  gradient-split(x, x_out, x_inner, 2)
  fuse(x, y, fused)
  ```

Iterative Compilation compiler searches xforms (Bodin et al.)
Performance Comparisons

References:
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

<table>
<thead>
<tr>
<th></th>
<th>TACO</th>
<th>Eigen</th>
<th>SS:GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeqIter</td>
<td>1.63</td>
<td>2.43</td>
<td>1.50</td>
</tr>
<tr>
<td>HashMap</td>
<td>1.63</td>
<td>2.44</td>
<td>1.50</td>
</tr>
<tr>
<td>Auto</td>
<td>2.72</td>
<td>4.06</td>
<td>2.50</td>
</tr>
</tbody>
</table>

### SpMSpM (Comparison with TACO Compiler)

<table>
<thead>
<tr>
<th>Layout A</th>
<th>Layout B</th>
<th>COO</th>
<th>CSR</th>
<th>DCSR</th>
<th>BCSR*</th>
</tr>
</thead>
<tbody>
<tr>
<td>COO</td>
<td>1.21</td>
<td>1.00</td>
<td>0.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSR</td>
<td>0.99</td>
<td>0.99</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCSR</td>
<td>0.97</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCSR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1.01</td>
<td></td>
</tr>
</tbody>
</table>

### TTM (large # dims)

<table>
<thead>
<tr>
<th>Tensor</th>
<th>Collection</th>
<th>NNZ</th>
<th>TACO</th>
<th>Generated</th>
<th>Generated Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>delicious-3d</td>
<td>FROSTT</td>
<td>140M</td>
<td>2.07s</td>
<td>2.14s</td>
<td>0.44s</td>
</tr>
<tr>
<td>flickr-3d</td>
<td>FROSTT</td>
<td>113M</td>
<td>1.12s</td>
<td>1.20s</td>
<td>0.27s</td>
</tr>
<tr>
<td>freebase-music</td>
<td>HaTen2</td>
<td>100M</td>
<td>1.26s</td>
<td>1.30s</td>
<td>0.74s</td>
</tr>
</tbody>
</table>

### Social Network Analysis

- delicious-3d
- flickr-3d
- freebase-music

### Pattern Recognition

- vast-2015-mc1

### Natural Language Processing

- NELL1
- NELL2

### Anomaly Detection

- 1998darpa

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

<table>
<thead>
<tr>
<th>Source Code</th>
<th>Abstract Syntax Tree (AST)</th>
<th>Static Single Assignment (SSA)</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>code2vec [Alon et al. 2018]</td>
<td></td>
<td>IR2Vec (LLVM) [Keerthy et al. 2019]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CDFG (LLVM) [Brauckmann et al. 2020]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProGraML (LLVM, XLA) [Cummins et al. 2020]</td>
<td></td>
</tr>
</tbody>
</table>

---

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)

```c
double thres = 5.0;
if (x < thres)
    x = y * y;
else
    x = 2.0 * y;
```

```c
x += 1.0;
```

Ben-Nun et al.: Neural Code Comprehension: A Learnable Representation of Code Semantics, NIPS'18
Neural Code Comprehension – inst2vec (2018)

- Embedding space (using the Skip-gram model)
Neural Code Comprehension – inst2vec (2018)

Table 3: Algorithm classification test accuracy

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Accuracy [%]</td>
<td>88.2</td>
<td>84.8</td>
<td>94.0</td>
<td>94.83</td>
</tr>
</tbody>
</table>

Predicts which device is faster (CPU or GPU)

Table 4: Heterogeneous device mapping results

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Grewe et al. [27]</th>
<th>DeepTune [17]</th>
<th>inst2vec</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Tahiti 7970</td>
<td>73.38</td>
<td>82.79</td>
<td>82.79</td>
<td>2.91</td>
</tr>
<tr>
<td>NVIDIA GTX 970</td>
<td>72.94</td>
<td>80.29</td>
<td>81.76</td>
<td>1.26</td>
</tr>
</tbody>
</table>

Optimal tiling

Table 5: Speedups achieved by coarsening threads

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Radeon HD 5900</td>
<td>1.21</td>
<td>1.10</td>
<td>1.17</td>
<td>1.25</td>
</tr>
<tr>
<td>AMD Tahiti 7970</td>
<td>1.01</td>
<td>1.05</td>
<td>1.23</td>
<td>1.07</td>
</tr>
<tr>
<td>NVIDIA GTX 480</td>
<td>0.86</td>
<td>1.10</td>
<td>1.14</td>
<td>1.02</td>
</tr>
<tr>
<td>NVIDIA Tesla K20c</td>
<td>0.94</td>
<td>0.99</td>
<td>0.93</td>
<td>1.03</td>
</tr>
</tbody>
</table>

Table 2: Analogy and test scores for inst2vec

<table>
<thead>
<tr>
<th>Context Size</th>
<th>Syntactic Analogies</th>
<th>Semantic Analogies</th>
<th>Semantic Distance Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Types</td>
<td>Options</td>
<td>Conversions</td>
</tr>
<tr>
<td>1</td>
<td>101 (18.04%)</td>
<td>13 (24.53%)</td>
<td>100 (6.63%)</td>
</tr>
<tr>
<td>2</td>
<td>226 (40.36%)</td>
<td>45 (84.91%)</td>
<td>134 (8.89%)</td>
</tr>
<tr>
<td>3</td>
<td>125 (22.32%)</td>
<td>24 (45.28%)</td>
<td>48 (3.18%)</td>
</tr>
</tbody>
</table>

Ben-Nun et al.: Neural Code Comprehension: A Learnable Representation of Code Semantics, NIPS’18
ProGraML – using the full graph structure with Graph Neural Networks

Cummins et al.: ProGraML: A Graph-based Program Representation for Data Flow Analysis and Compiler Optimizations, ICML'21
# ProGraML on compiler tasks

<table>
<thead>
<tr>
<th>Problem</th>
<th>Analysis type</th>
<th>Example optimization</th>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability</td>
<td></td>
<td>Dead code elimination</td>
<td>DeepTuneIR</td>
<td>0.520</td>
<td>0.497</td>
<td>0.504</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ProGraML</td>
<td>0.997</td>
<td>0.995</td>
<td>0.996</td>
</tr>
<tr>
<td>DomTree</td>
<td></td>
<td>Global Code Motion</td>
<td>DeepTuneIR</td>
<td>0.721</td>
<td>0.081</td>
<td>0.138</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ProGraML</td>
<td>0.985</td>
<td>0.693</td>
<td>0.781</td>
</tr>
<tr>
<td>DataDep</td>
<td></td>
<td>Instruction scheduling</td>
<td>DeepTuneIR</td>
<td>0.999</td>
<td>0.136</td>
<td>0.236</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ProGraML</td>
<td>1.000</td>
<td>0.988</td>
<td>0.993</td>
</tr>
<tr>
<td>Liveness</td>
<td></td>
<td>Register allocation</td>
<td>DeepTuneIR</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ProGraML</td>
<td>1.000</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>Subexpressions</td>
<td></td>
<td>Global Common Subexpression</td>
<td>DeepTuneIR</td>
<td>1.000</td>
<td>0.123</td>
<td>0.214</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Elimination</td>
<td>ProGraML</td>
<td>0.965</td>
<td>0.925</td>
<td>0.930</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>—</td>
<td>DeepTuneIR</td>
<td>0.810</td>
<td>0.209</td>
<td>0.273</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>ProGraML</td>
<td>0.989</td>
<td>0.920</td>
<td>0.940</td>
</tr>
</tbody>
</table>

Cummins et al.: ProGraML: A Graph-based Program Representation for Data Flow Analysis and Compiler Optimizations, ICML’21
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

Traditional numerical simulation

Surrogate model (machine learning)

Query

Approx

Surrogate model (machine learning)

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
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)

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

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

ESPM2 2022: Seventh International Workshop on Extreme Scale Programming Models and Middleware
ICICLE: NSF AI Research Institute

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.

ICICLE: Intelligent CyberInfrastructure with Computational Learning in the Environment

Use Inspired Science Domains

Smart Foodsheds  Animal Ecology  Digital Agriculture

Emerging Computing Continuum

On Field Sensors  Edge & Near Edge  Clouds  HPC Systems & Data Centers

ICICLE: NSF AI Research Institute

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.
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
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
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
AI4CI: Cross Stack Layers

- Efficient plug-n-play: Constantly adapt and optimize the heterogeneous (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
**AI for CI**

- **Efficient plug-and-play:** Constantly adapt and optimize the heterogeneous 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
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.
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](https://submissions.supercomputing.org/eval.html)

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