Computer Architecture Simulation Using Machine Learning

Adolfy Hoisie
Chair, Computing for National Security Department

Work with: Lingda Li, Thomas Flynn, Ray Ren, and many others

Challenges for System and Application Design

- Multiple constraints
  - Optimal performance
  - Power constraints
  - Fault tolerance
- Adaptivity: vast numbers of “knobs” to deal with
  - Applications–data driven
  - Systems–heterogeneous
- Complexity of the system software stack–dynamic behavior
  - Models in runtime
  - Actionable models
  - Guiding runtime optimizations and operation
- Complex architectures and associated technologies
  - Need to leverage marketplace
  - Extreme-scale systems are increasingly emerging as a synthesis of technologies
  - Leverage commoditization but adds specific smarts
- Modeling is called to capture multiple boundaries of the hardware-software (HW-SW) stack.
- Applications must cope with and help mitigate the increased complexity.
- Triggers the need for modeling now; wide-spread exploration of future applications and technologies

SMaSH: Smart Modeling and Simulation for HPC

Brookhaven National Laboratory

Performance Prediction Methods: Speed versus Accuracy

SMaSH is an intricate challenge because of the complexity of the design space.
Methodologies exist that lack either practicality or accuracy.

	Speed	Accuracy	Flexibility
Analytical Modeling	Fast	Low	Low
Emulation	Fast	High (?)	Very low
Discrete Event Simulation	Slow	High	High
ML-based Simulation	Medium; aiming high	High	Medium; aiming high

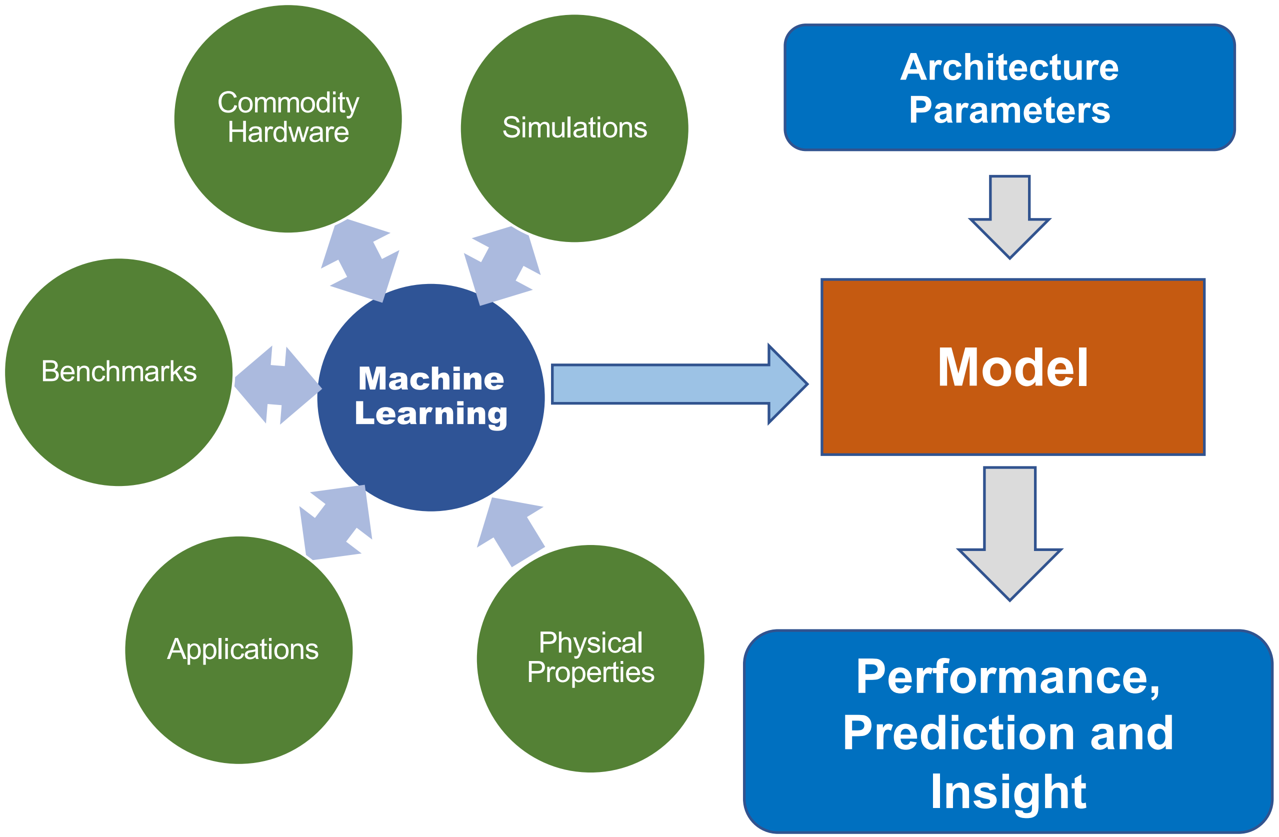
Discrete event simulation (DES) is slow:

- For example, gem5 simulates a modern microprocessor at several hundreds of KIPS.
- Not practical for realistic architectures and workloads.

GOAL: Accelerate accurate Architecture Simulation by two orders of magnitude.

Machine Learning as the Holy Grail?

- Recent progress in ML affords potential opportunities to address these problems
- Many questions need to be addressed:
  - Are the new methodologies applicable?
  - Are new uses possible?
  - Can ML's predictivity limits be conquered?
  - What is the accuracy vs. computational cost?

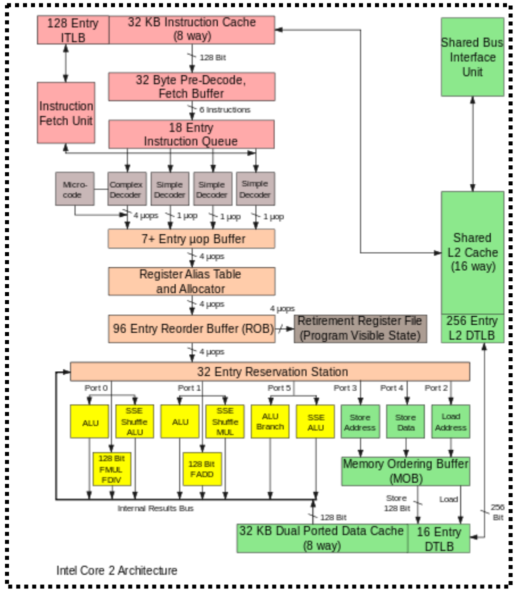


Brookhaven National Laboratory

Accelerate DES? Why Not Simulate the Entire Processor Instead?

Traditional Simulation

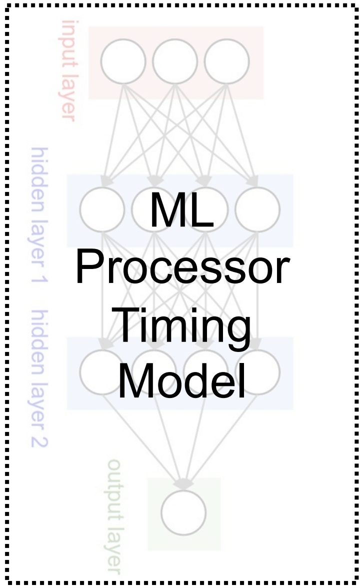
Instruction



Latency

ML-based Simulation

Instruction



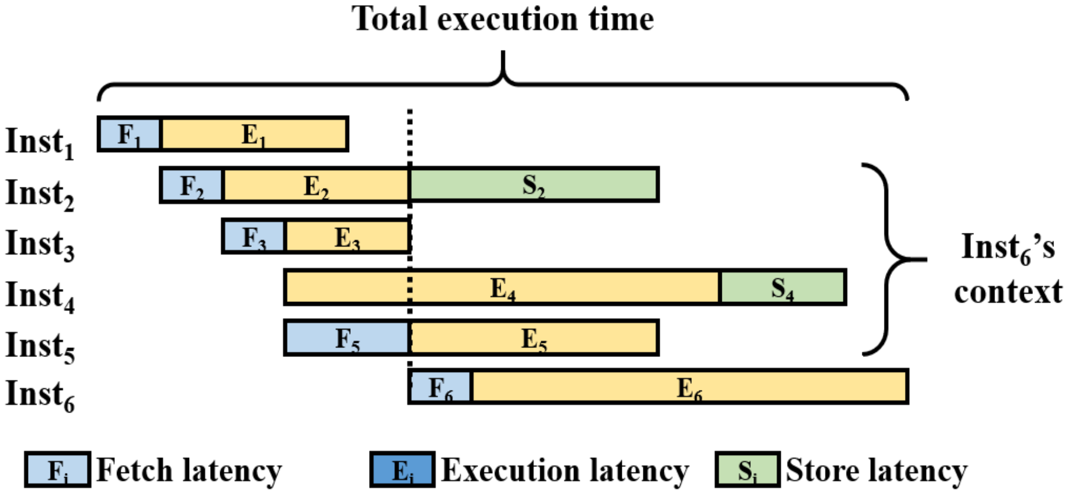
Latency

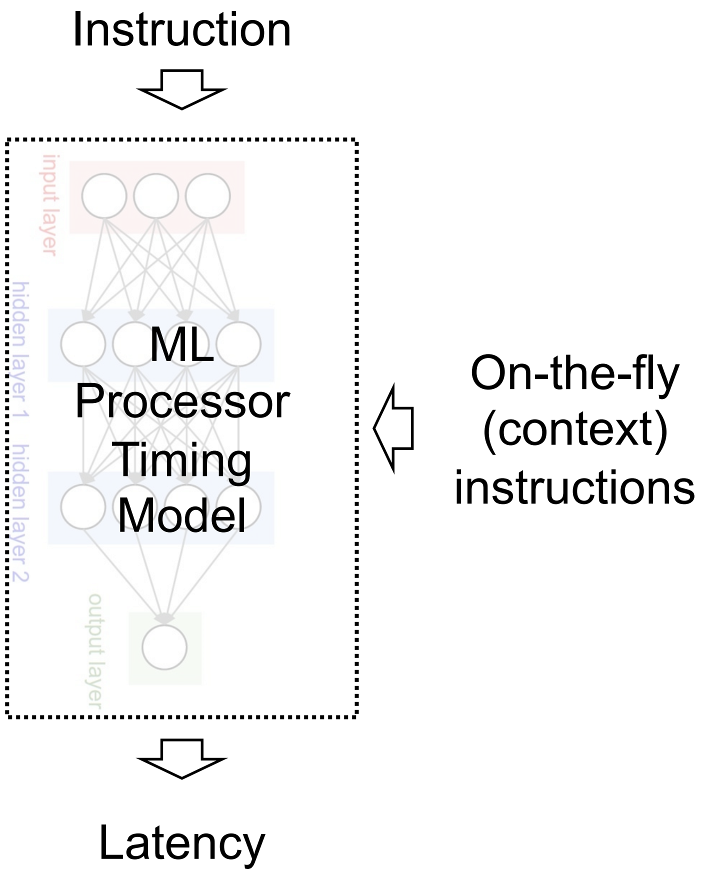
On-the-fly (context) instructions

- Traditional approach simulates all processor behavior.
- ML-based approach incorporates timing-related details into a mathematical model and ignores timing-irrelevant details.
- Use context instructions as part of input to capture dependencies/ hazards.

Simulating Application Performance

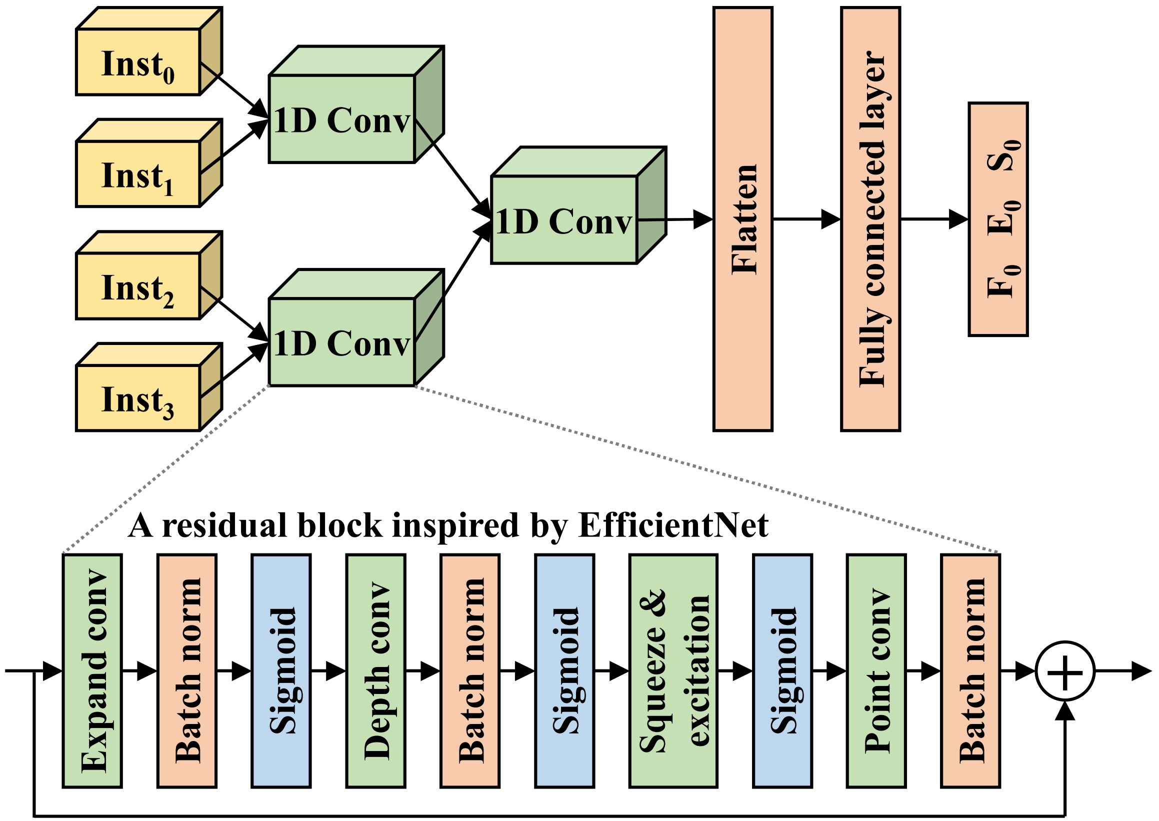
- Instructions are fed into the ML model in execution order.
- For each instruction, the fetch, execution, and store latencies are predicted.
- On-the-fly instructions are updated based on the results then move on to predict for the next instruction.
- Application performance is determined after all instructions have been simulated.





Neural Network Architectures

- Explored a spectrum of state-of-the-art ML models for computer architecture simulation.
  - Fully connected layers
  - Convolution layers: capture the timing relationship between instructions
  - Improved the transformer encoder model [NIPS'17], a vision transformer (ViT)-like model [arXiv'20]
  - Implemented a long short-term memory (LSTM)-based model [ICML'19]
- Designed specific layers for simulation
  - Use a neural network to study the relationship between the current instruction and one context instruction and do so for all context instructions.



Brookhaven National Laboratory

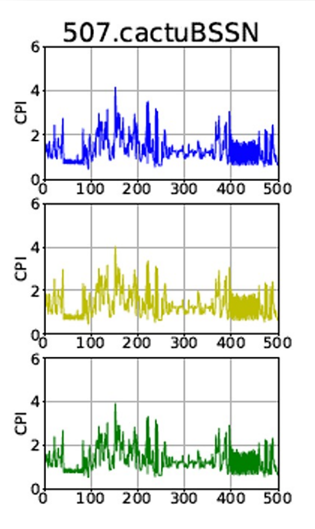
Machine Learning Works for Architecture Simulation!

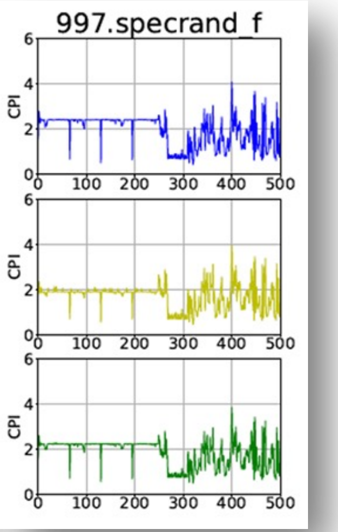
Accuracy: Quantitatively

Neural network architecture	Computation demand (million multiplications)	Instruction latency prediction accuracy (# cycles)			Average absolute application simulation error
		Fetch latency	Execution latency	Store latency	
7RB+2F, best CNN model	93	0.15	0.96	0.52	0.96%
Transformer encoder	88	0.49	2.06	0.88	2.4%
ViT, small	118	0.34	6.99	1.69	20%
ViT, large	351	0.26	4.19	1.35	14%
LSTM	119	0.57	3.27	1.29	2.4%

Observation: CNN models achieve the best accuracy with less computation demand.

…and Qualitatively





[Paper under publication at:](https://arxiv.org/abs/2105.05821)

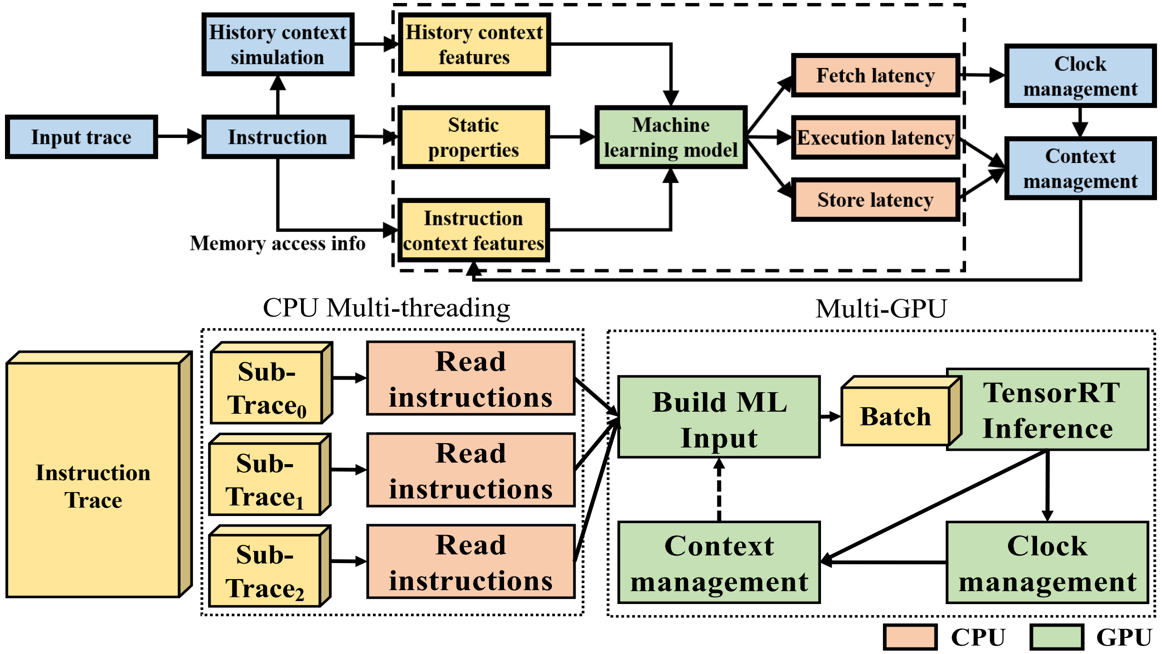
https://arxiv.org/abs/2105.05821

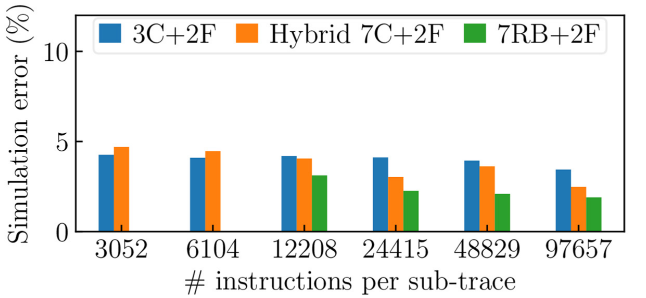
Brookhaven National Laboratory

SMaSH to Date: Significant, Meaningful Progress

- ML-based ModSim methodology developed.
- "SIMNET" Simulator infrastructure and research prototype implemented.
- SIMNET optimized algorithmically and through software engineering.
- Validation using realistic benchmarks and architectures.

Model	Benchmark Simulation Error	
	Range	Abs. Average
2F	[-0.97%, 28%]	18%
3C+2F	[-6.6%, 5.2%]	1.9%
5C+2F	[-4.6%, 5.5%]	2.0%
7C+2F	[-8.7%, 6.3%]	2.5%
7C+2F	[-6.3%, 10.6%]	2.3%
7RB+2F	[-2.7%, 0.78%]	0.96%





Extremely promising results…quantitatively and qualitatively

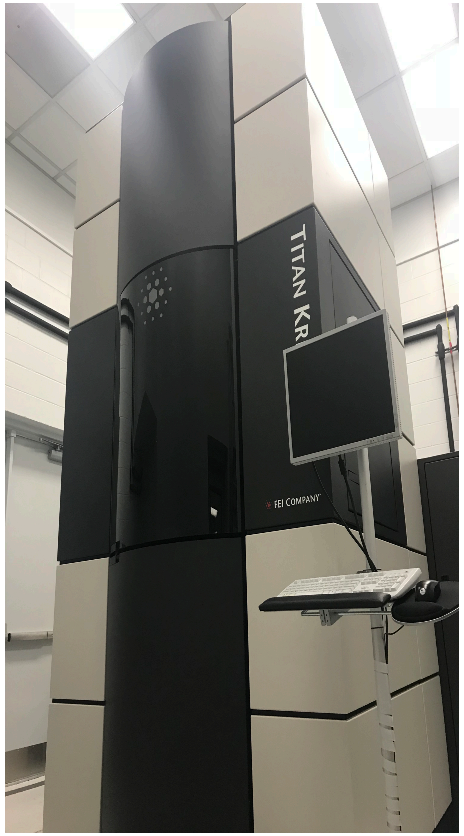
https://arxiv.org/abs/2105.05821

Brookhaven National Laboratory

Dynamic Codesign of HW-SW for Fast Analysis of High-throughput Scientific Experiments

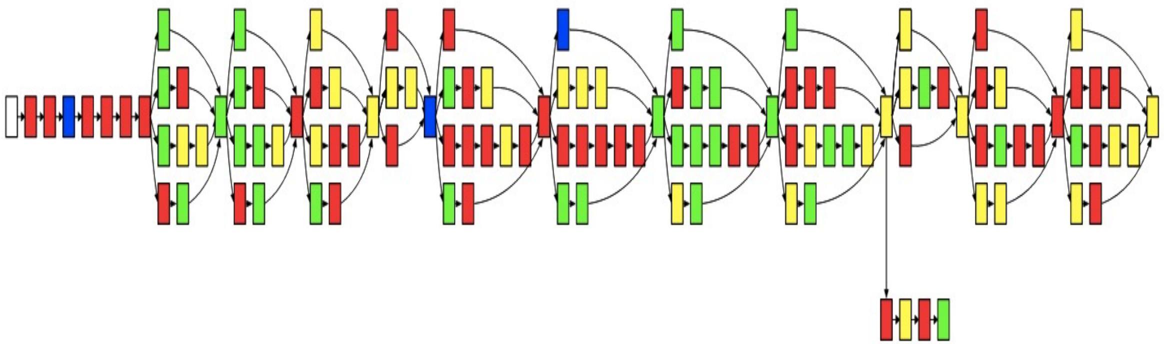
Next-generation Detectors: High-throughput, High-velocity

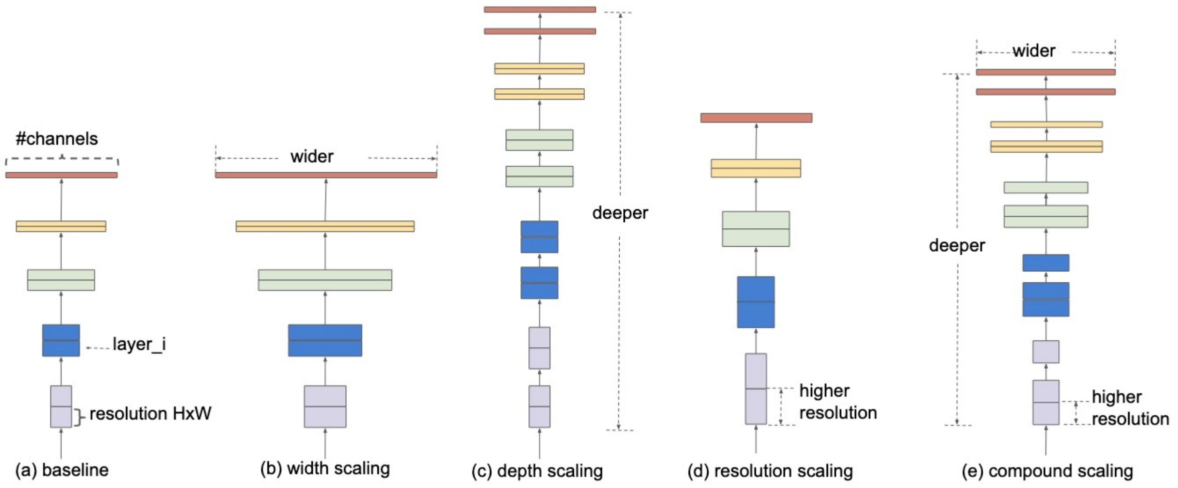
- Increasing spatial and temporal resolutions leads to high volumes and velocities of data.
  - Example: Recent scanning electron microscopes can produce 50 GB/s of data.
- Need to process images quickly, extract insights and rapidly incorporate insights into new settings or experiments.
- Diverse operating modes (streams, bursts of data) and heterogenous detectors.



Beyond Static Codesign Approaches

- Promising work on optimizing ML workloads
  - Device placement: how to place elements of a computation graph onto available accelerator cores
  - Resizing neural nets: How to trade off accuracy for model size.
- However, dynamic approaches are needed:
  - Different parts of an experiment call for disparate imaging settings (impacting data resolution/rate)
  - Algorithm settings change (e.g., required accuracy)
  - Shifting demands may require different HW-SW mappings for optimal performance





Top: Device Placement Optimization with Reinforcement Learning. Mirhoseini et al. (2017)
Bottom: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Tan and Le (2019)

Wanted: Model-driven for Dynamic Modeling/Codesign

Ability to find new placements or mappings on the fly is needed.

From this vantage point, codesign is not merely static mapping of HW onto SW, but a dynamic data-driven process.

- Vital for now-dominant, data-driven workloads and heterogenous architectures

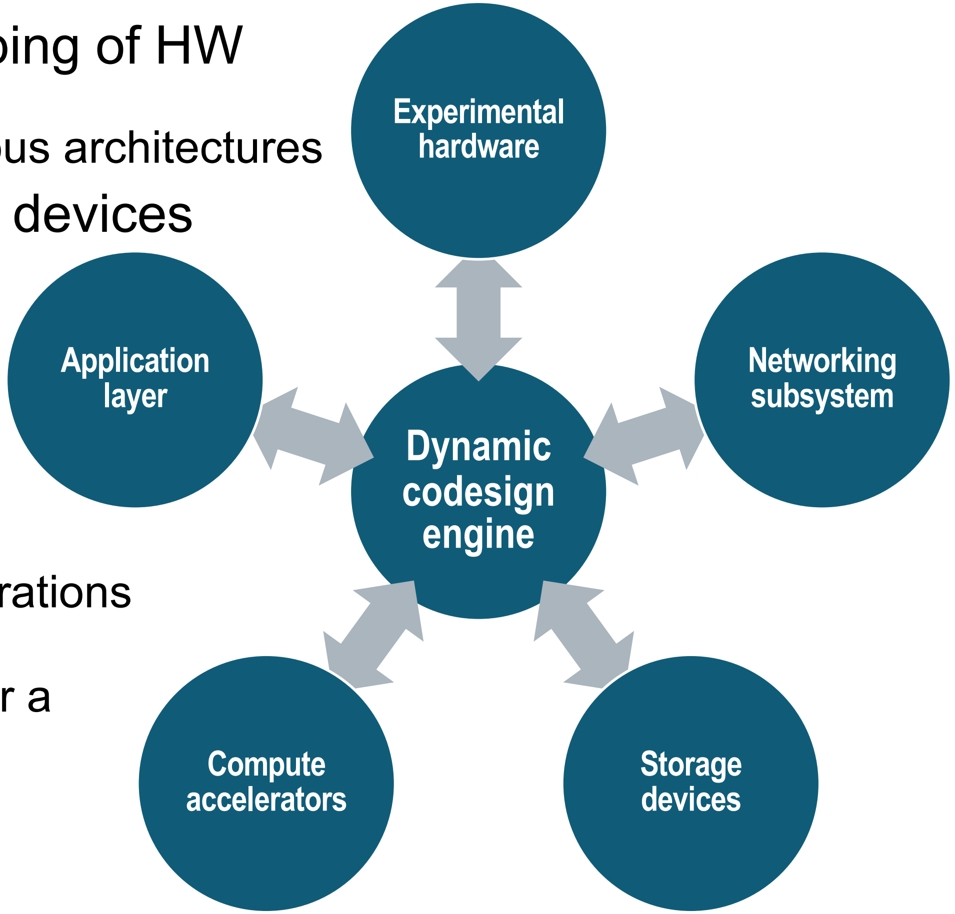
In this regime, experimental HW, application SW, and other devices (e.g., storage) are all coupled through feedback loops.

- Rational, quantitative ways to reconfigure components while experiments are conducted

Feedback loops for performance and scientific criteria.

This model-based codesign approach can benefit from ML tools and frameworks:

- Gather training data from actual experimental and SW configurations
- Observe performance data
- Use ML models to predict optimal actions and knob settings for a dynamic codesign engine
- Train the intelligent runtime using reinforcement learning



Summary and Conclusions

ModSim is at a crossroads due to system heterogeneity and data-driven workloads.

- Solution *may* be in sight when dealing with complexity seems unbearable.

Workload characterization is on a new path.

ML is the dominant application on clouds and extreme-scale systems.

- ML is a promising modeling tool!

For performance modeling, simulator development remains a significant challenge.

- SMaSH is a new frontier in ML for system ModSim.

Dynamic modeling is key.

- Static approaches cannot account for dominant runtime effects in a data environment.
- Dynamic models, including those based on ML, show significant promise for complex data workflow management and optimization.

Center for Advanced Technology for Artificial Intelligence (CAT-AI) at Brookhaven Lab: nexus for these and other related technologies.