

I ILLINOIS

CSL | Coordinated
Science Lab

COLLEGE OF ENGINEERING

Subho S. Banerjee*, Saurabh Jha*, Zbigniew T.
Kalbarczyk, Ravishankar K. Iyer^{+*}

*Computer Science

+Electrical and Computer Engineering

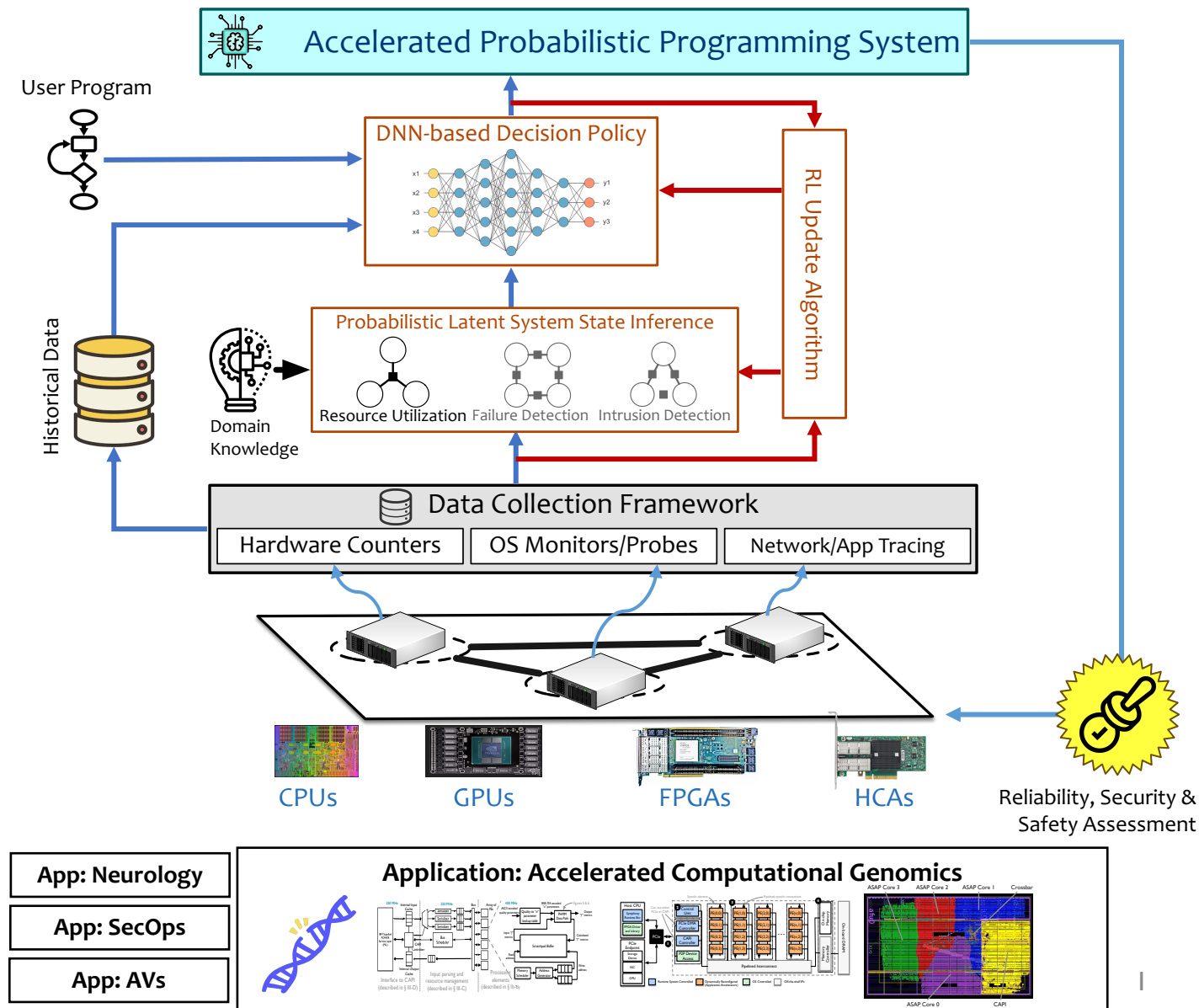
Inductive-bias-based Reinforcement Learning for Efficient Schedules in Heterogeneous Clusters

csl.illinois.edu



Symphony: A Framework for ML in Systems

- A new framework for management of large-scale computers using ML
 - Encode “systems knowledge” as inductive bias
 - Combine Bayesian model with Deep Learning
- Broader goals:
 - Performance (*Scheduling, SLO*)
 - Resilience (*Errors, Failures, Attacks*)
- This paper
 - **Model:** Scheduling accelerated workloads
 - **Uncertainty:** Dealing with errored telemetry data
 - **Training Algorithm:** Sampling based approximations for backprop in Bayesian Models

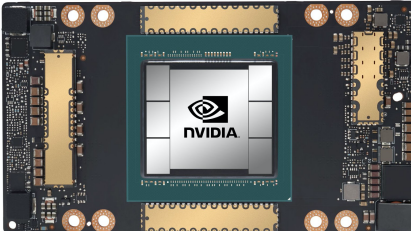

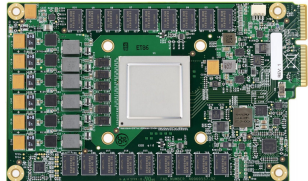


Changing Landscape of Computing

Emerging applications drive need for more compute

 Autonomous Agents  Personalized Medicine  SecOps Analytics ...

Accelerators are becoming first class citizens in datacenter deployments

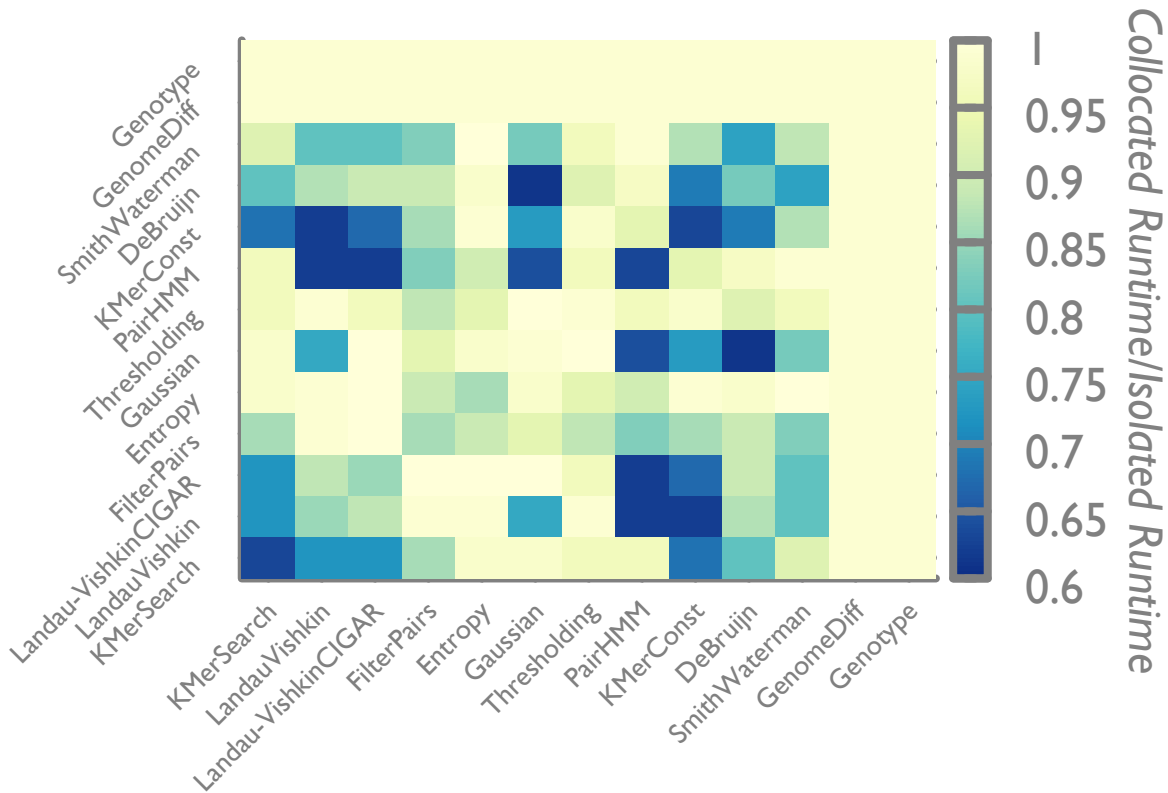
 GPGPUs  FPGAs [EC2, Azure]  ASICs [TPU, IPU etc.] ...

What's missing? How & When do you take advantage of heterogeneous hardware without painstakingly deriving new heuristics? – ML to the rescue

- Today done largely by static policies (heuristics)
- How to make use of **dynamic** and **contextual information** about Apps/Systems?

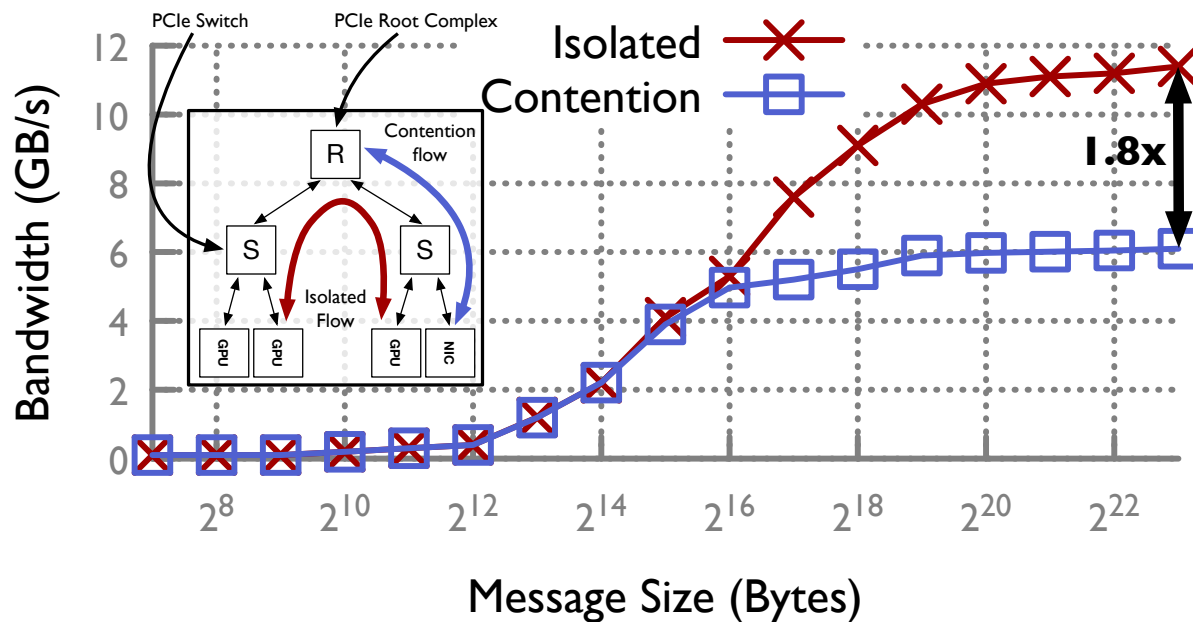
Dynamic Information is Key to Resource Management

Latent Resource Contention in CPUs



Shared resource contention **reduces performance by as much as 40%**

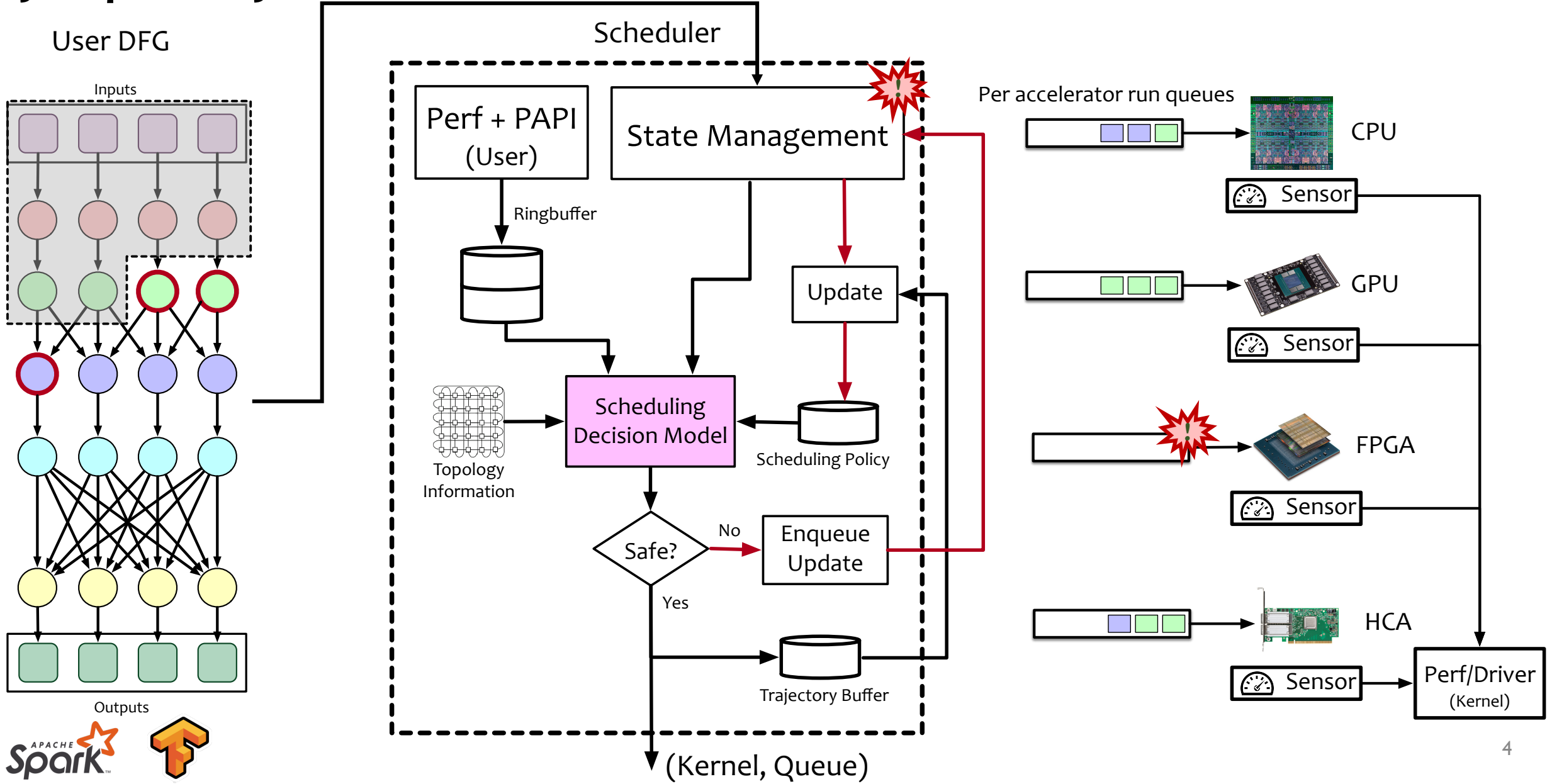
Latent Resource Contention in Accelerators



PCIe bandwidth **impacts performance by as much as 50%**



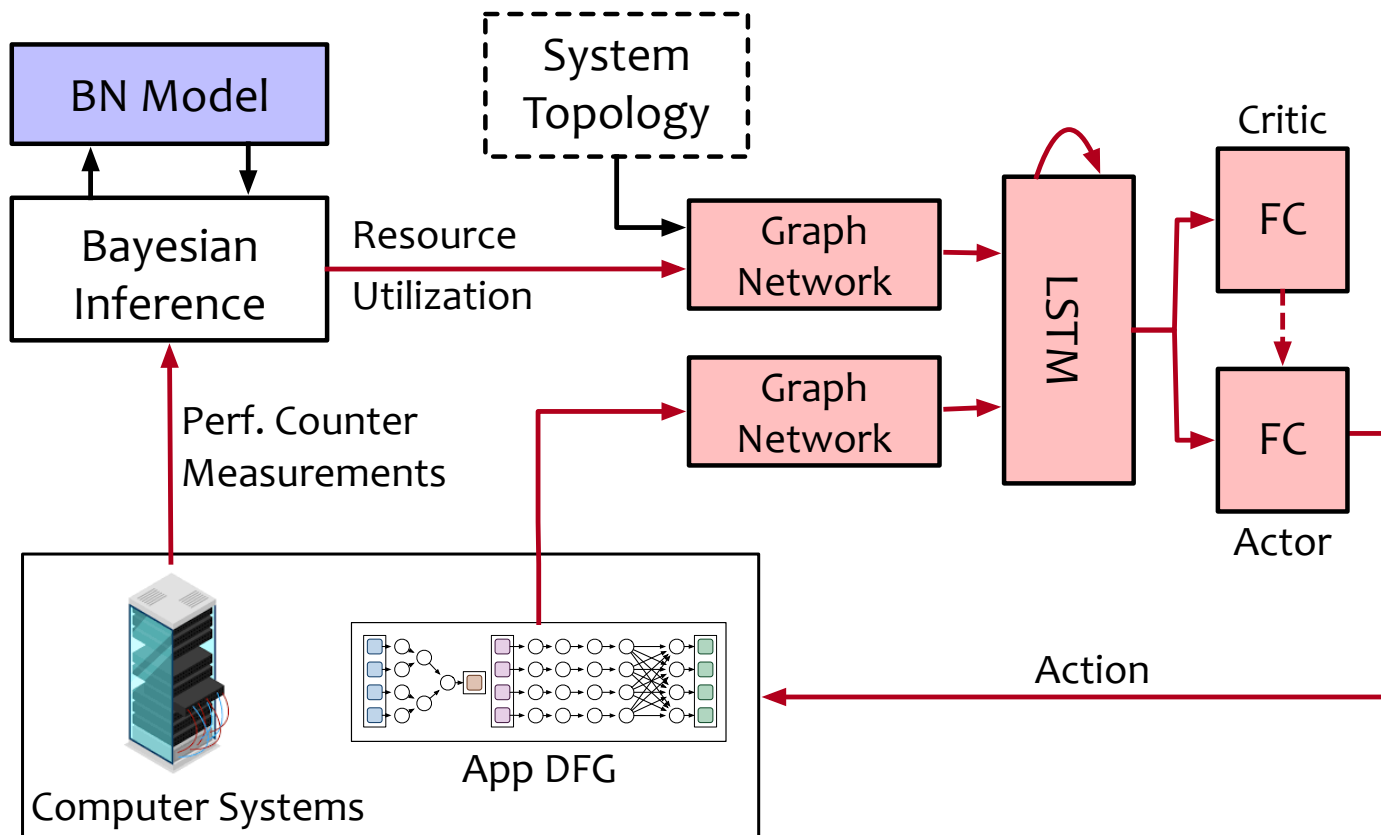
Symphony: Execution Overview



Symphony: Bayesian + Deep Learning Models

Resource Management Problem:

- **State Estimation:** Find the important resources and current utilizations
- **Decision:** Find a packing that optimally utilizes resources



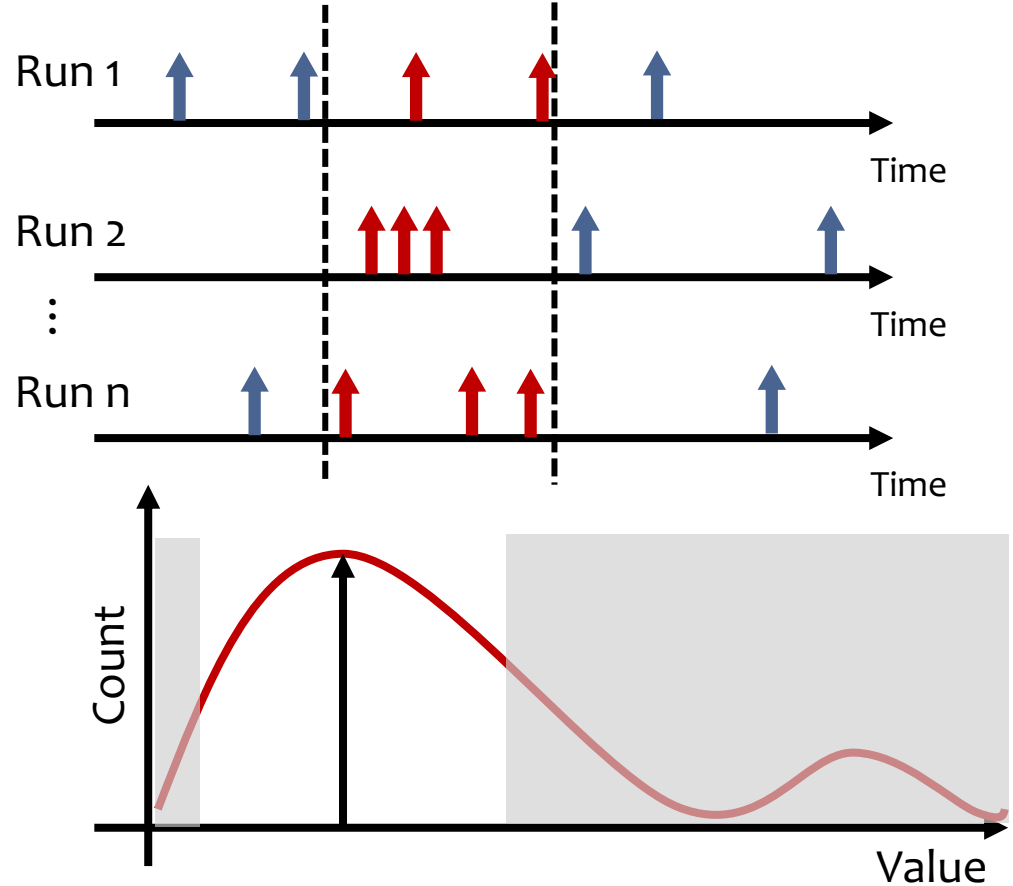
- BN Model: Capture **uncertainty** in state estimation
 - Nondeterministic sampling + delays
 - Measurement Error
- NN Model: Capture aspects of optimization
 - Graph Network: Capture DFG & System Topology embeddings
 - LSTM + AC: Capture time varying information
- Trained by RL to minimize makespan

Modeling Uncertainty in Sampled Performance Data

Problem: Measurements of performance counters are noisy (as much as 40%)

Traditional Solution (Offline Variance Reduction)

- Used primarily for offline analysis like profiling—**completely unusable!**

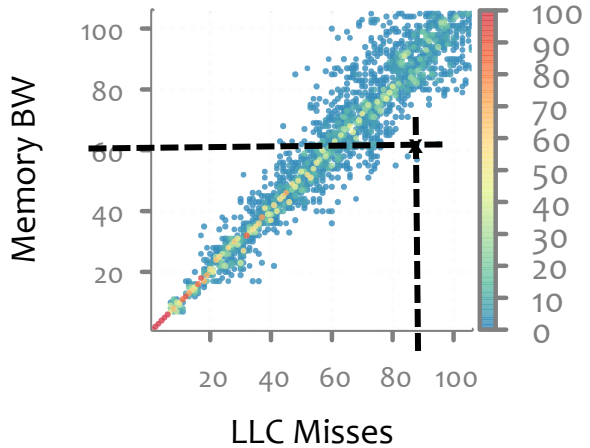
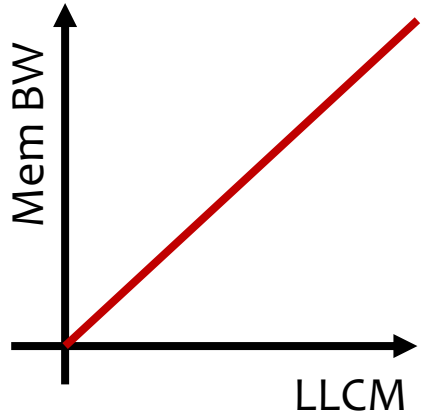


Our Solution (Generative Model of Error)

Key Insight: Different perf counters are interrelated based on system architecture

Perf Counters: *Memory BW, LLC Misses*

$$\text{Memory BW} = \frac{\text{LLC Misses} \times \text{Cacheline Size}}{\delta T}$$



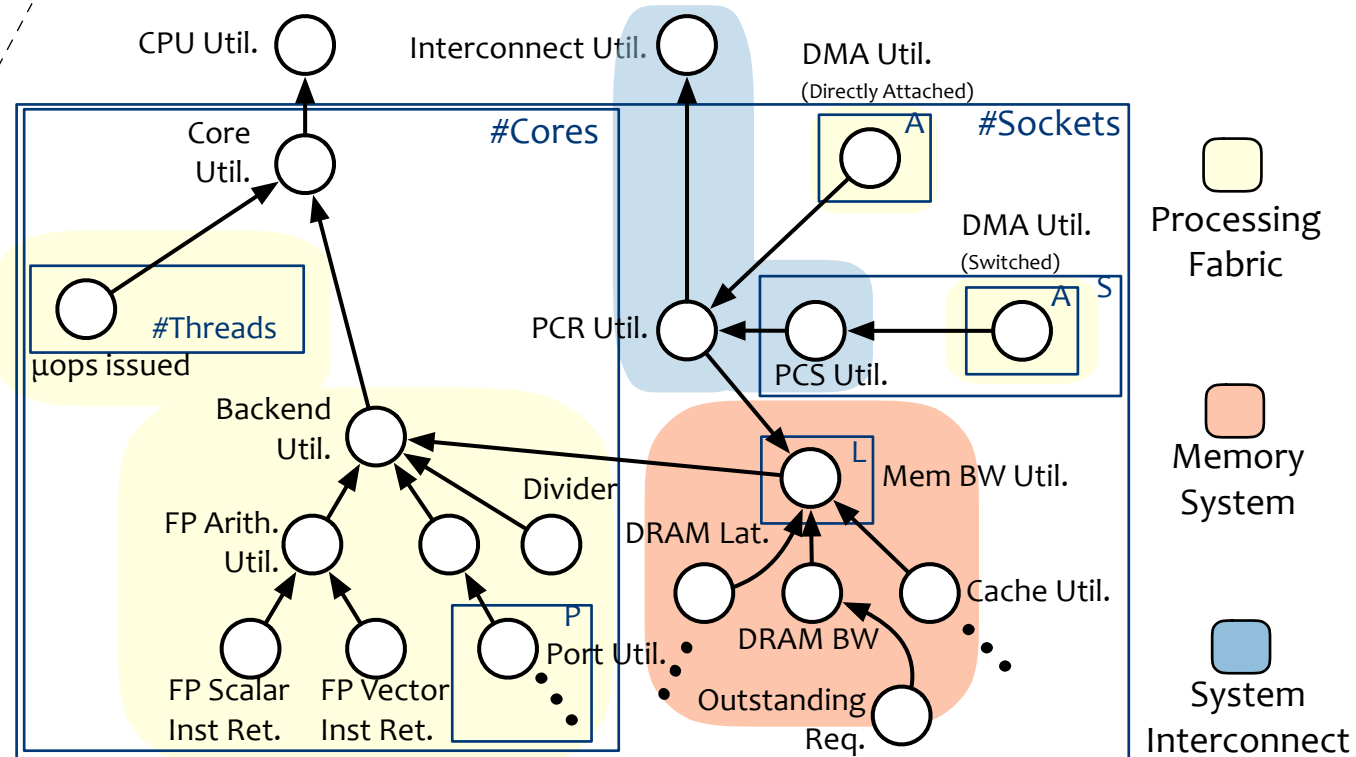
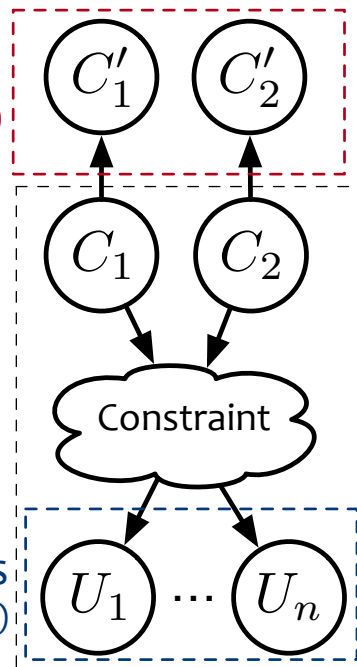
Bayesian Network Model

$$C'_i \sim C_i + \frac{\alpha\sigma(\bar{C}_i)}{\sqrt{N}} \text{Student}(\nu = N - 1)$$

Measured Values
(noisy)

True Values

Utilizations
(inferred)

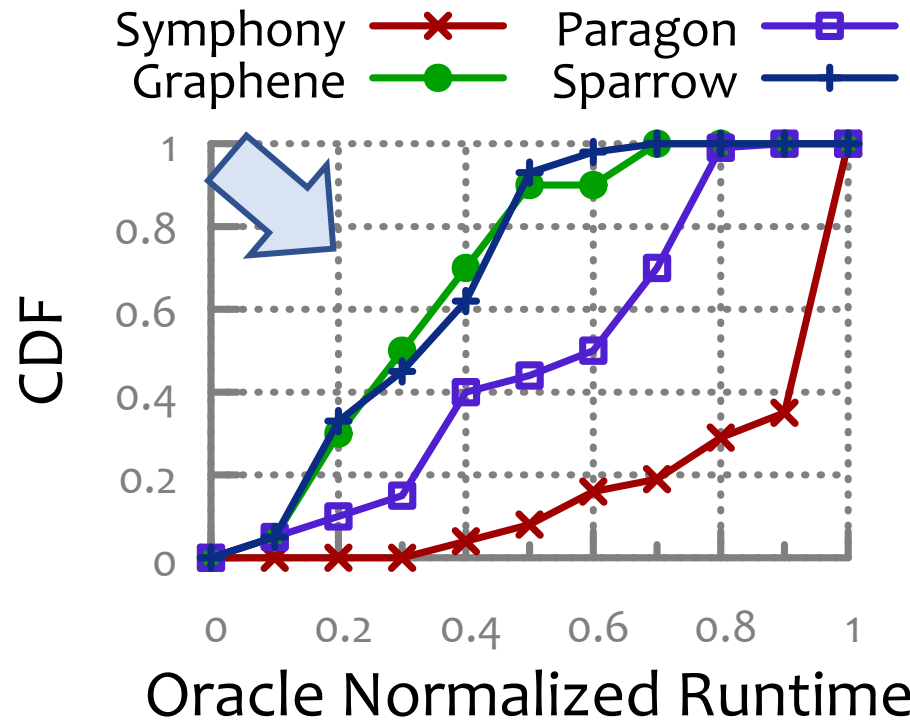


- Scalable, general & works for real processors
 - x86 (Intel, AMD Zen), ppc64 (IBM), ARM (aarch64 – subset)
- BN automatically automatically from per μ -arch listing in Linux Source Tree
 - Contributed by vendors to Linux

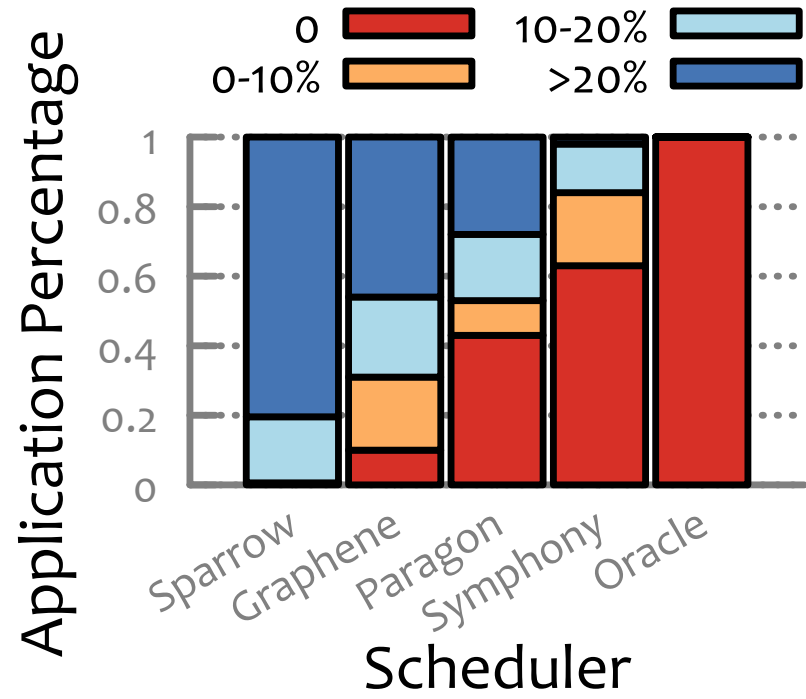
Results: Keep the Abstractions and Get Performance Too

Outperforms Paragon **by 32%** (at 99thile)

Performance **within 6% of oracle schedule** (at 99thile)



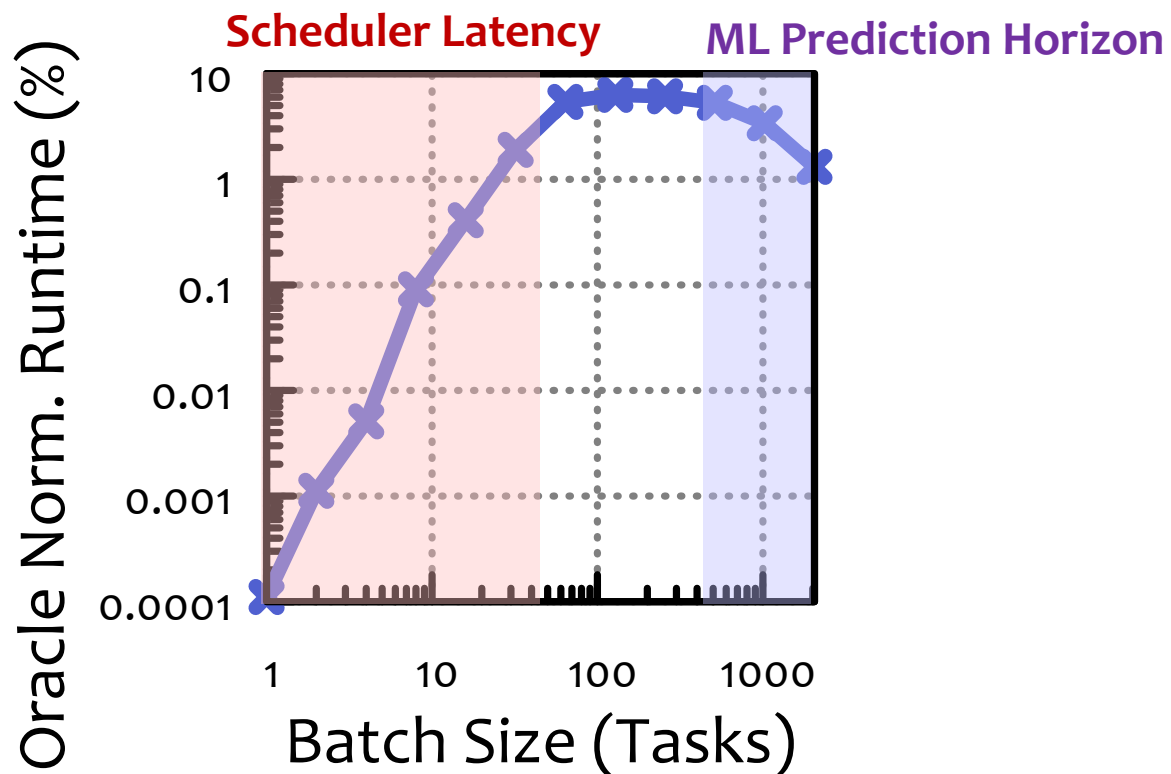
63% of kernels execute with isolated performance, remaining with **<20% performance loss**



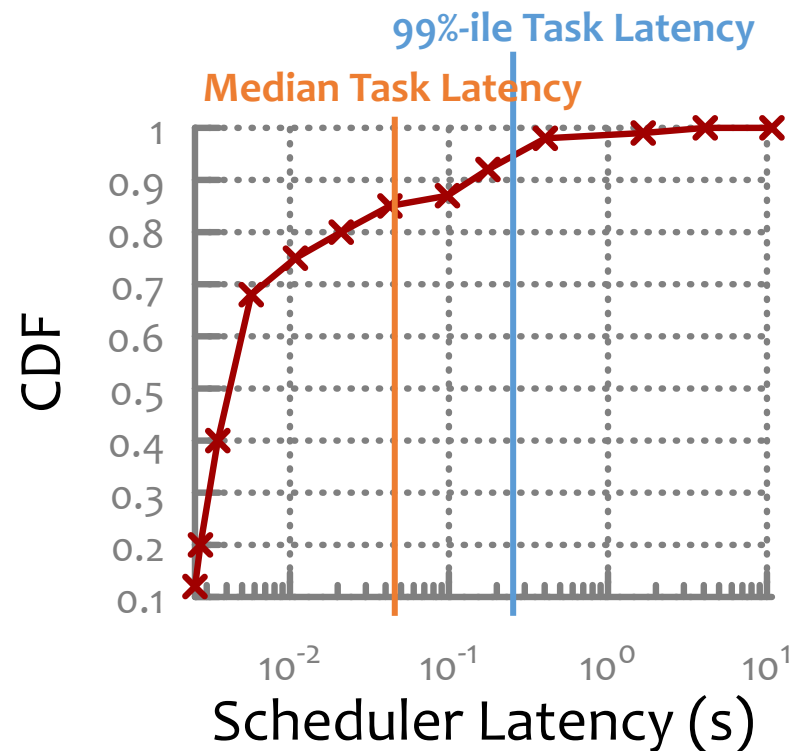
No Free Lunch: Scheduler Latency Can Negate Gains

Hack: Batching

Insight: Amortize cost of scheduling over batches

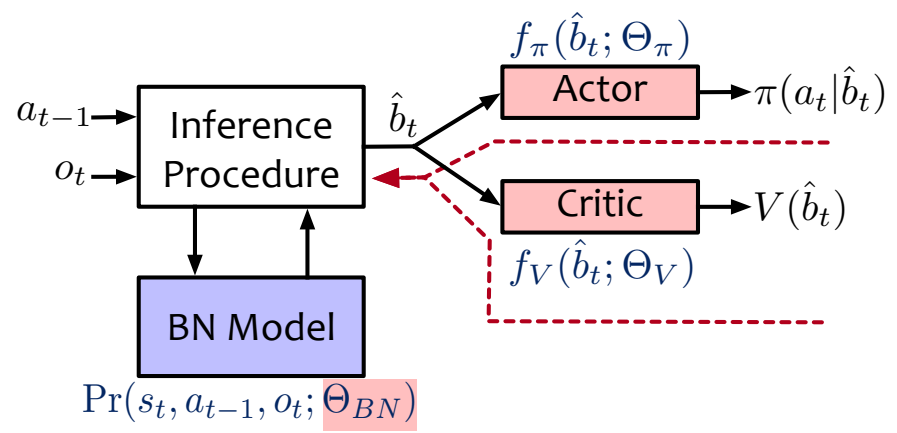


Real Solution: Reduce Latency to a point where latency becomes irrelevant



- Outer loop: Can be prohibitively expensive to calculate the ideal batch size
- BN training time dominates in cases in the tail

Sampling-based Back Propagation for Bayesian Networks



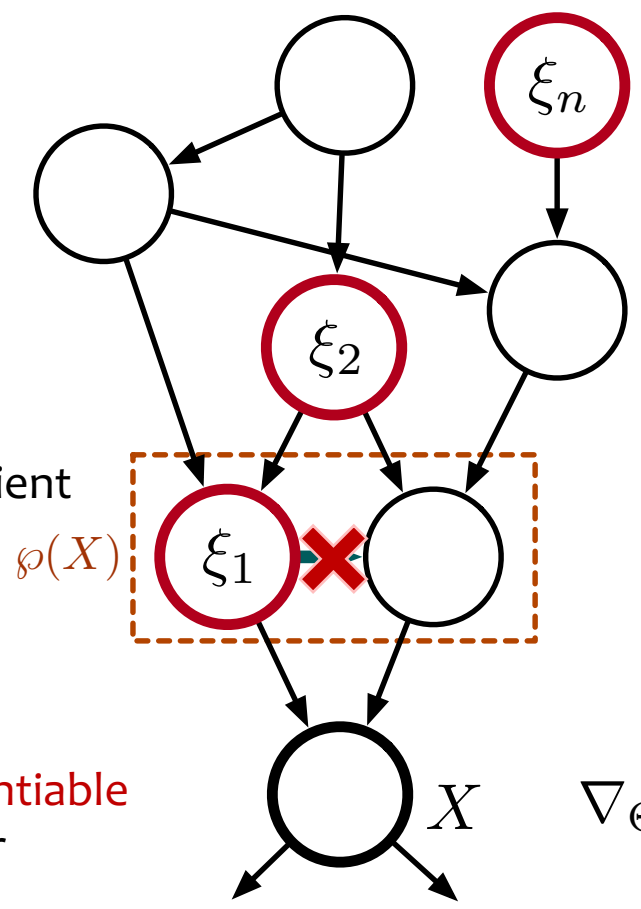
Backpropagation requires calculating its gradient

$$\nabla_{\Theta} \mathcal{L}_t^{RL} = \nabla_{\Theta} \mathcal{L}_t^A(\rho) + \nabla_{\Theta} \mathcal{L}_t^V(\eta)$$

$$\nabla_{\Theta_{BN}} \hat{b}_t \text{ ???}$$

Inference procedure might not be differentiable

- Requires calculation of an integral over domain of variable



Can be recursively expanded

$$\nabla_{\Theta_{BN} \setminus \theta_X} \Pr(\varphi(X) = \mathbf{y}_i | \xi = \mathbf{a})$$

$$\nabla_{\Theta_{BN} \setminus \theta_X} \Pr(X = x | \xi = \mathbf{a})$$

$$\approx \sum_{i=1}^S \frac{n_S(\mathbf{y}_i)}{S} f_X(x, \mathbf{y}_i; \theta_X) \nabla_{\Theta_{BN} \setminus \theta_X} \Pr(\varphi(X) = \mathbf{y}_i | \xi = \mathbf{a})$$

$$\frac{\partial \Pr(X = x | \xi = \mathbf{a})}{\partial \theta_X} \approx \sum_{i=1}^S \frac{n_S(\mathbf{a}, \mathbf{y}_i)}{n_S(\mathbf{a})} \frac{\partial f_X(x, \mathbf{y}_i; \theta_X)}{\partial \theta_X}$$

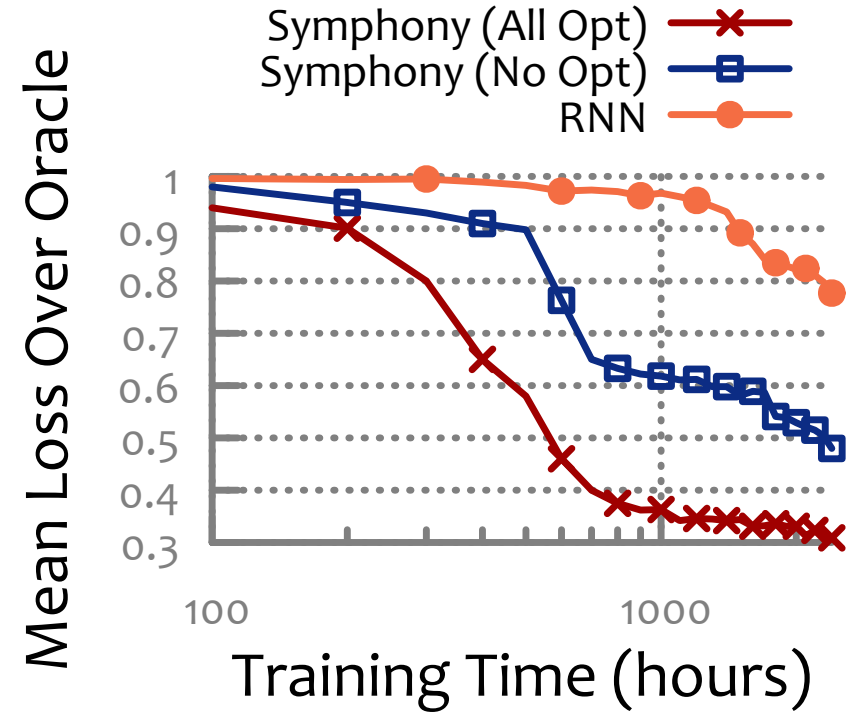
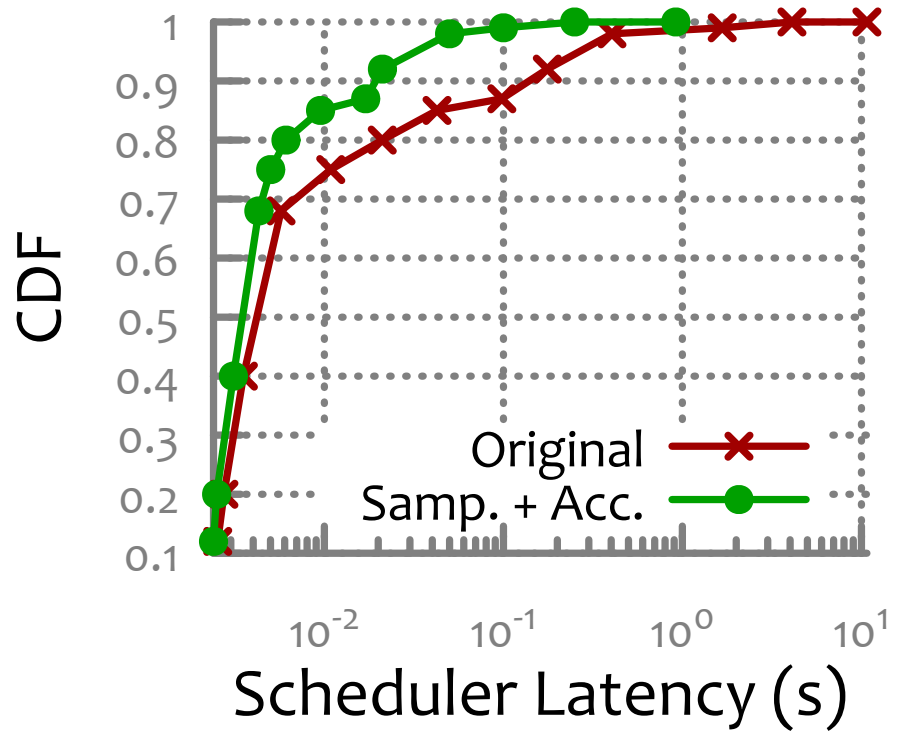
$$\nabla_{\Theta_{BN}} \Pr(X = x | \xi = \mathbf{a})$$

This is algorithmically expensive (exponentially large)

Our approximation can do this in polynomial time

Symphony: Model + Approximations + Acceleration

Approximations reduce latencies (**12x in tail**) and improve training times



- Summary:
- Model enables **capturing system state** and **aleatoric uncertainty**
 - Approximations enable **low-latency operation**