FSA: Co-Designing Future High Performance Systems for Efficiency and Scalability

Shuaiwen Leon Song

Director, Future System Architecture Lab (FSA)

University of Sydney CS and UW Seattle ECE

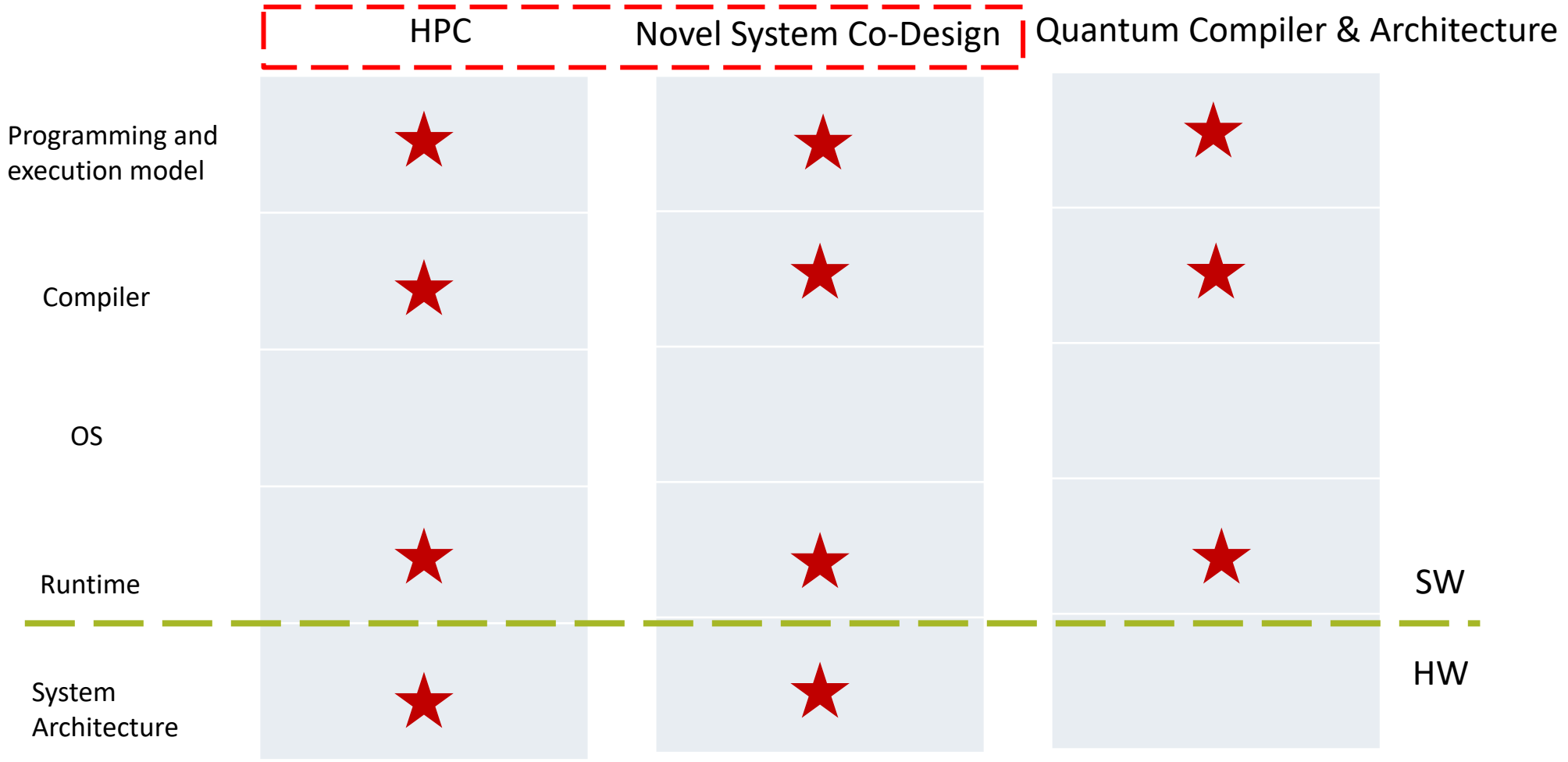
DSI presentation, August 18th, 2021

THE UNIVERSITY OF SYDNEY

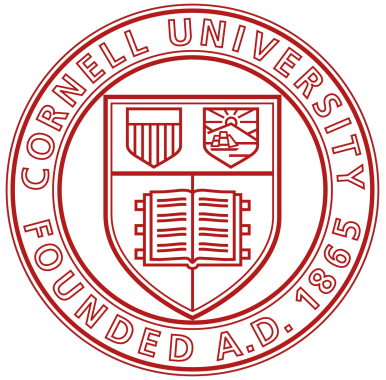
FSA

I acknowledge the tradition of custodianship and law of the Country on which the University of Sydney campuses stand. We pay our respects to those who have cared and continue to care for Country.

Our Research Overview



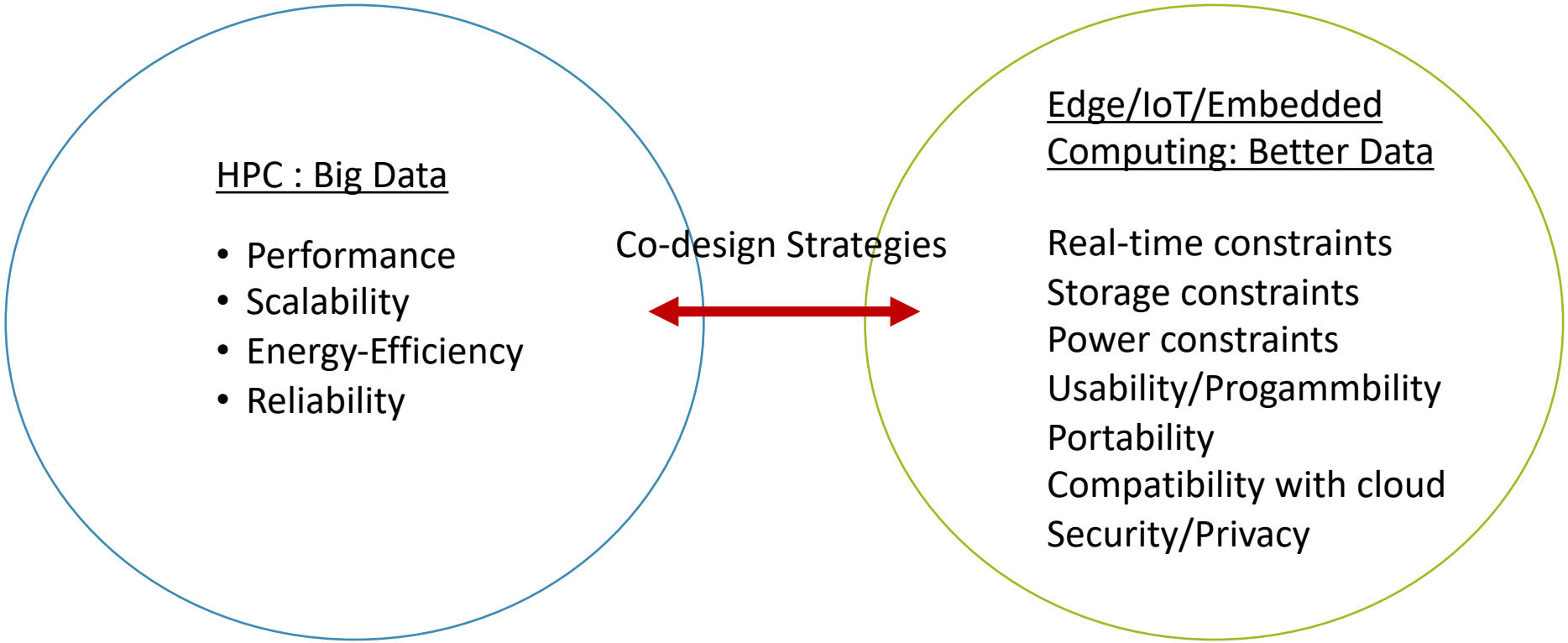
Our People and Collaborators



Data-Centric Bespoke Design

- We constantly struggle to identify the best design and optimization strategies to consider application features, data characteristics, and the system constraints from software and hardware. We always work under constraints unless we customize our own hardware and software for everything.

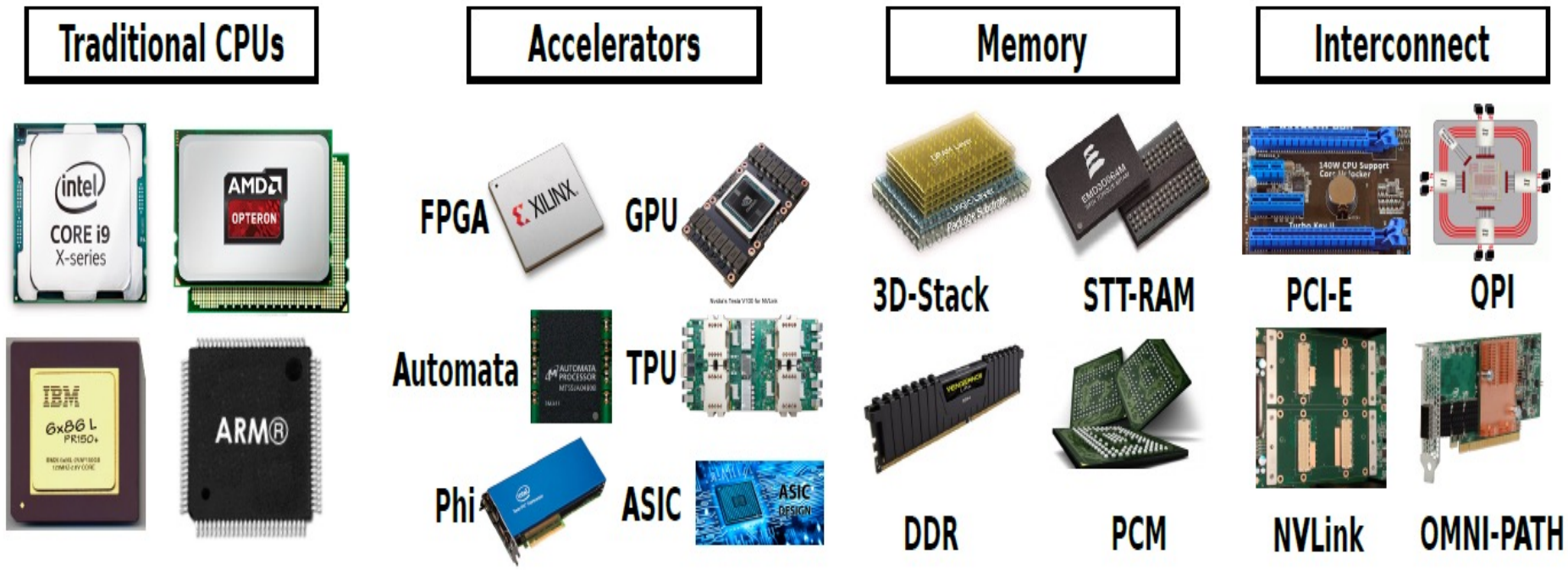
- Challenge: Hardware lottery between general-purpose design vs customized acceleration



Focus of HPC Research

- Scalability, efficiency and optimization of scientific and non-scientific applications under current hardware constraints;
- Effective big data reduction for scientific simulation and modelling;
- High-performance System Designs for emerging ML services and applications (SystemML)

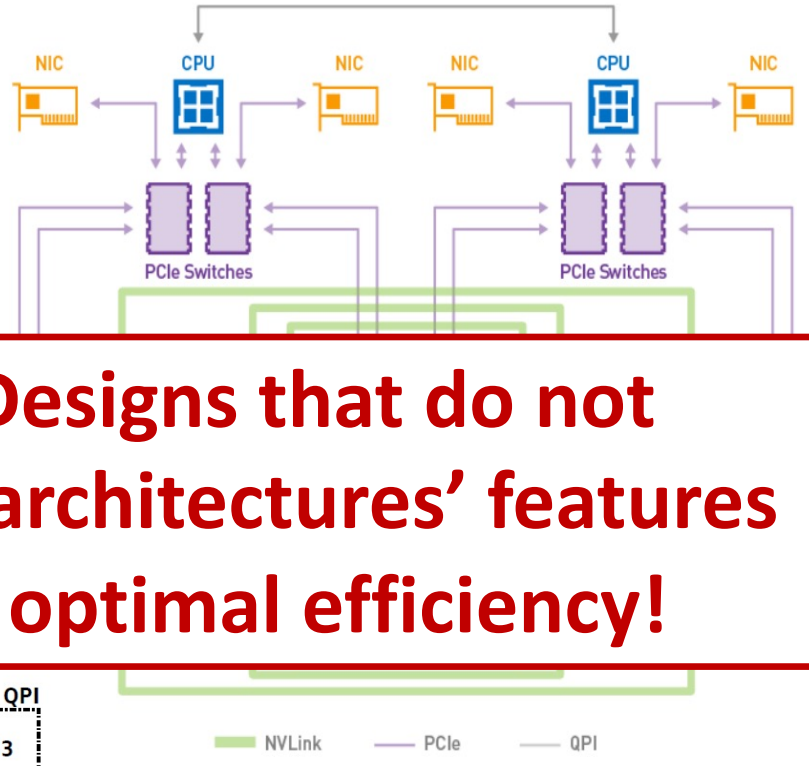
HPC Architectures: Extreme Heterogeneity



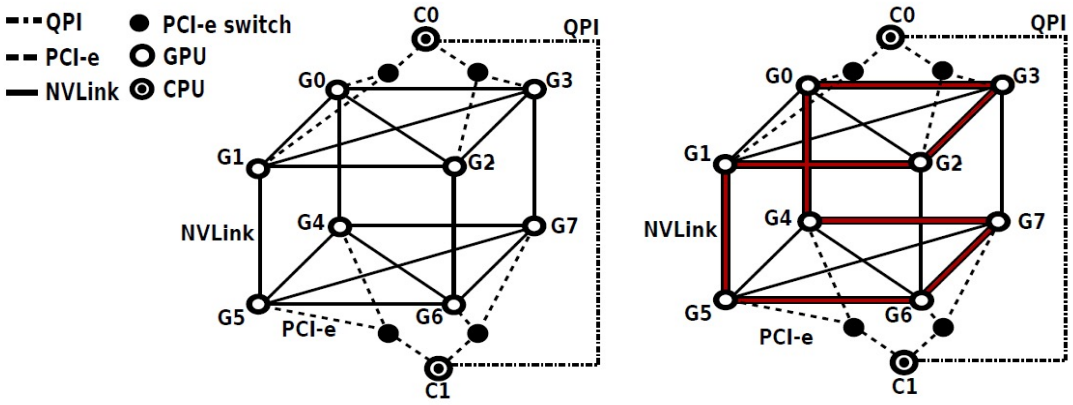
- ECP RFI feedback from vendors suggests that exascale platforms and beyond will become increasingly heterogeneous.
- Reports also suggest a variety of novel technologies are likely to be integrated into a single extreme heterogeneous system.
- We already see sings of this in the planned system architectures for **CORAL and APEX** exascale supercomputer acquisition.

Co-Design Dilemma: Architecture Becomes More Complex

Fig. Hybrid Cube Mesh NVLink Topology as used in DGX-1 with V100 (courtesy of NVIDIA)



Traditional Software Designs that do not consider future complex architectures' features will no longer provide optimal efficiency!



(A) P100-based DGX-1 with NVLink-V1

(B) V100-based DGX-1 with NVLink-V2

✓ Even the NVLink design itself exhibits heterogeneity: a fast backbone ring is formed for V2.

Showcase I: Efficiency and Scalability

S-BLAS: High performance parse linear algebra kernels for complex multi-accelerator based HPC architectures

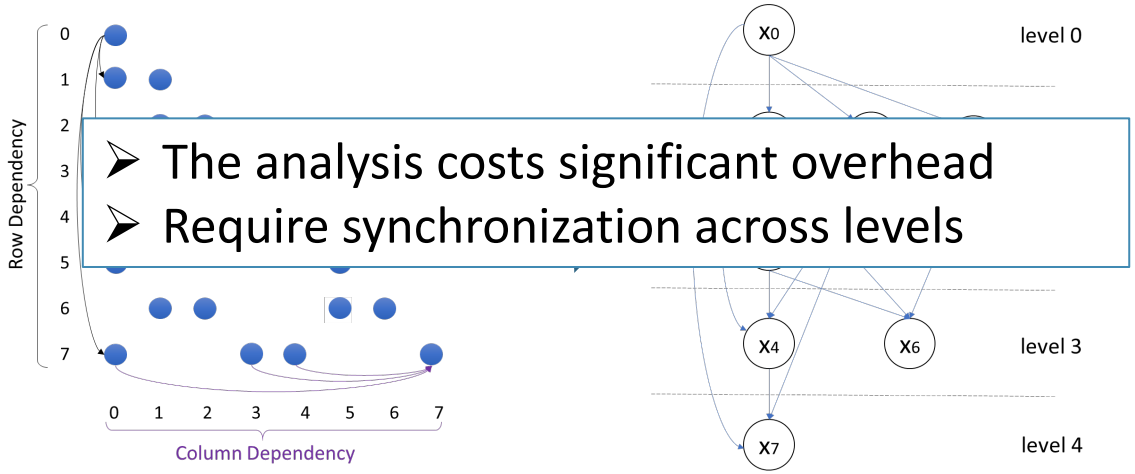
https://github.com/pnnl/s-blas

- Sparse-Matrix-Vector-Multiplication (SpMV)
- Sparse-Triangular-Solve (SpTRSV)
- Sparse-Matrix-Transposition (SpTrans)
- Sparse-Matrix-Matrix-Multiplication (SpMM)

Fast and Scalable Sparse Triangular Solver for Multi-GPU Based HPC Architectures, IEEE ICPP 2021 (A)

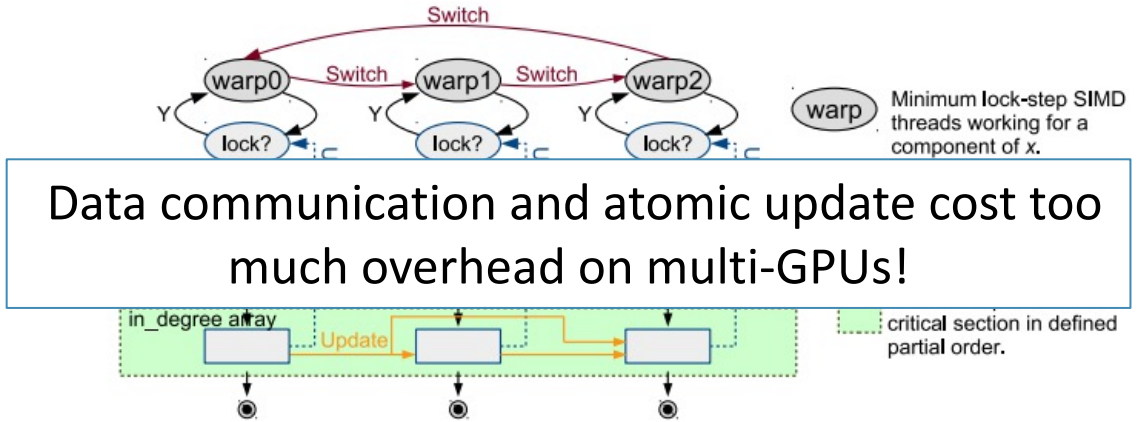
The SOTA SpTrsv on Multi-GPU Based HPC Architectures

- cuSparse lib: ***csrsv2()***
  - Basic idea: some components are independent and can be processed simultaneously.



- The analysis costs significant overhead
- Require synchronization across levels

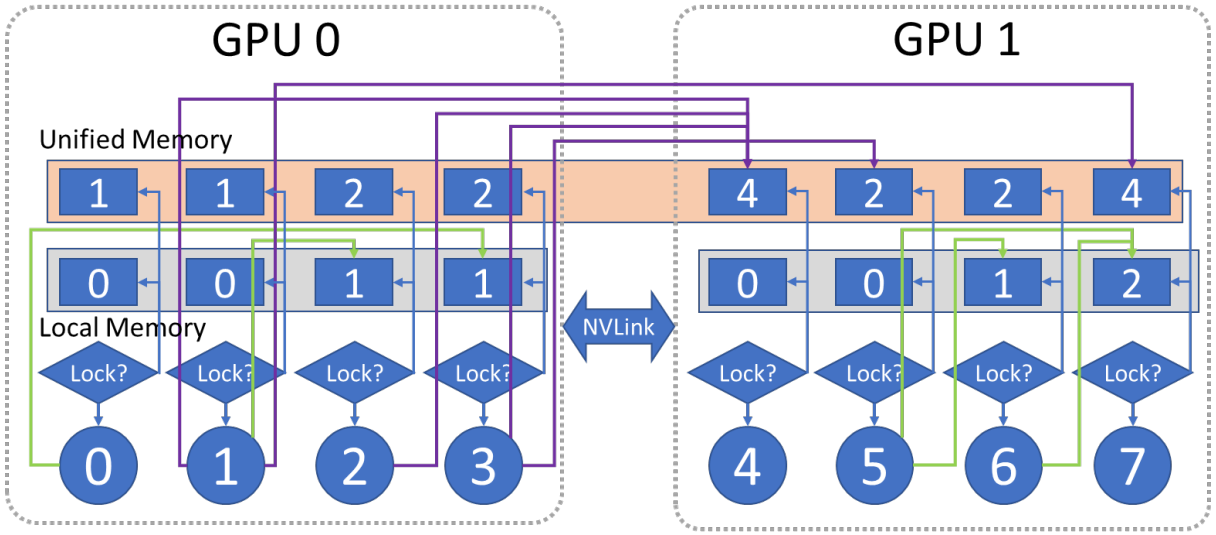
- A Synchronization-Free SpTrsv
  - Basic idea: Migrating the level-sets analysis at runtime
  - Components are scheduled by the hardware warp-switch of the GPU
  - Update the intermediate value (in_degree and left_sum) using GPU atomic operations



Data communication and atomic update cost too much overhead on multi-GPUs!

SpTrsv with Unified Memory

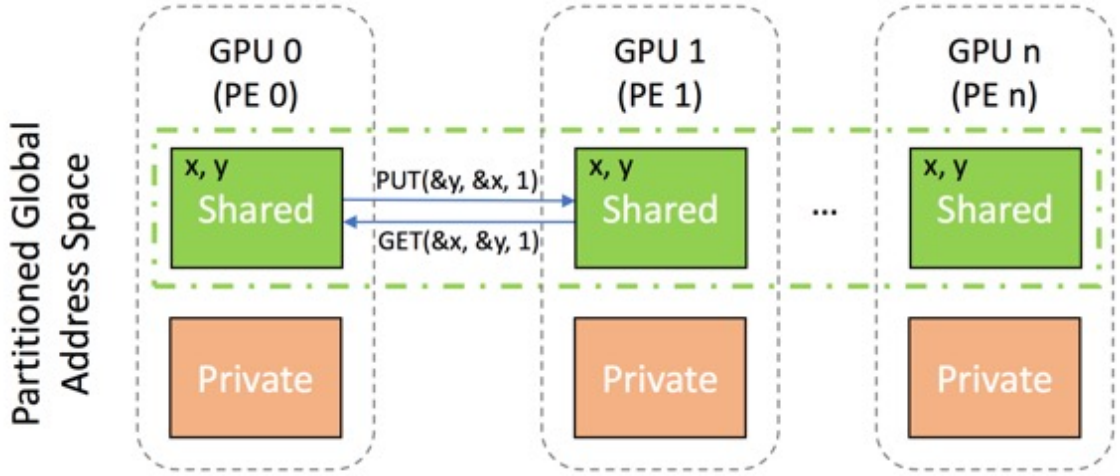
- Convenient
  - Hide complexity from users
- Page fault mechanism
  - Coarse-grained data copy
- Data contentions
  - System-wide atomic update will access the data simultaneously
- Workload unbalance
  - Dependency are unidirectional



Require Low Overhead Fine-Grained Communications!!

GPU CUDA NVSHMEM

- OpenSHMEM-based PGAS programming interface for multi-GPUs
- GPU-side interface allows GPU threads to
  1. Access distributed memory via data movement API
  2. Direct load/store (LD/ST) where GPUs are P2P-accessible
  3. **Highly-concurrent fine-grained messaging**
  4. **Asynchronously one-side data communication**

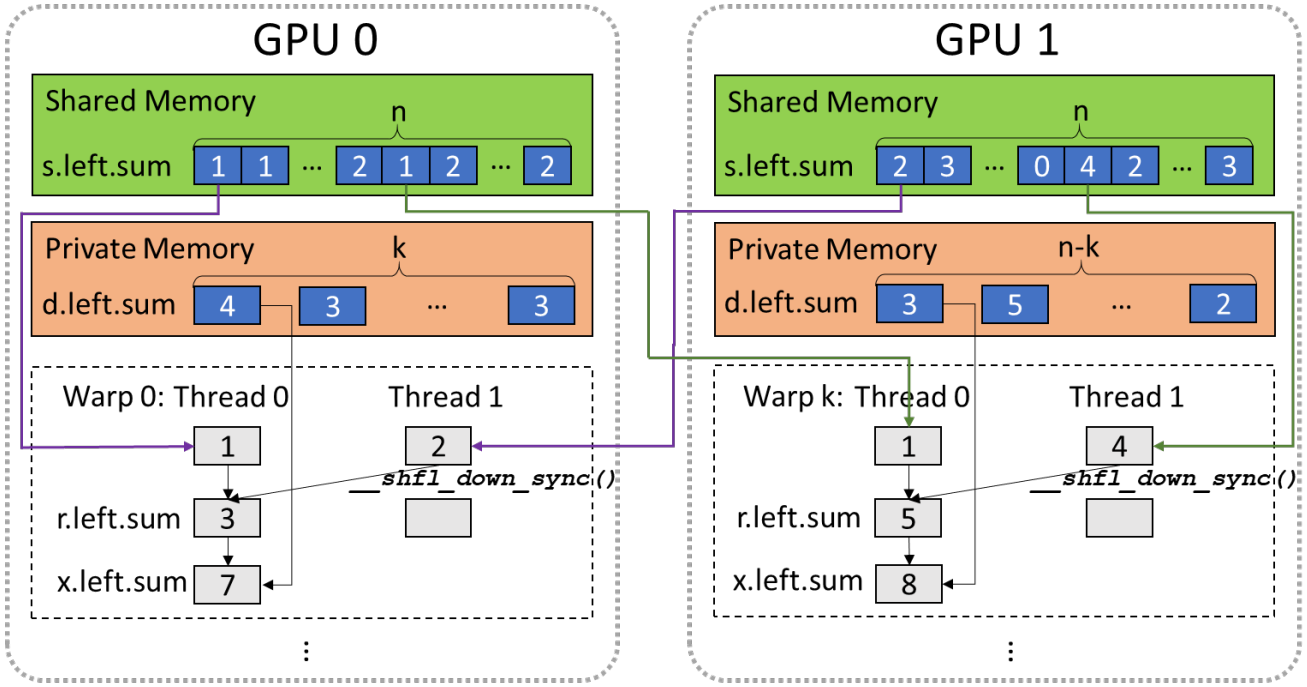


Peer Direct LD/ST and Global Remote Access

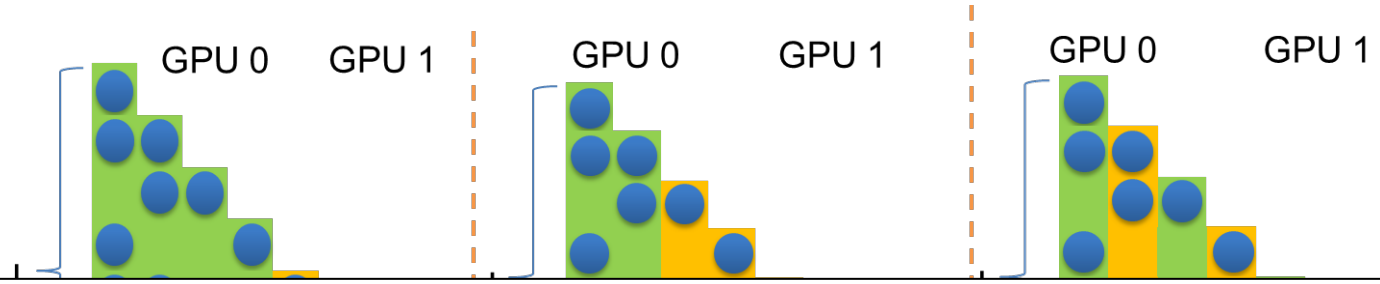
Using NVSHMEM for Resolving Dependency

- For SpTrsv, we convert the system-wide atomic update from unified memory to NVSHMEM shared memory space

```
Solve device x:
for all i ∈ dev.x do                                  ▷ parallel in warps
    x.in.degree[i] ← 0
    while d.in.degree[i]+1 ≠ x.in.degree[i] do
        for all d ∈ ngpu do                           ▷ parallel in threads
            if r.in.degree[i][d] ≠ 0 then
                r.in.degree[i][d] ← get(s.in.degree[i],d)
        x.in.degree[i] ← reduction(*r.in.degree[i])
    for all d ∈ ngpu do                               ▷ parallel in threads
        r.left.sum[i][d] ← get(s.left.sum[i],d)
    x.left.sum[i] ← reduction(*r.left.sum[i])
    x[i] ← b[i] - d.left.sum[i] - x.left.sum[i]
    x[i] ← x[i]/val[col.ptr[i]]
    for all j ∈ nnz_i do                              ▷ parallel in threads
        rid ← row.idx[j]
        if rid ∈ dev.x then
            d.atomic.add(&d.left.sum[rid], val[j]x[i])
            d.atomic.incr(&d.in.degree[rid])
        else
            d.atomic.add(&s.left.sum[rid], val[j]x[i]))
            d.atomic.decr(&s.in.degree[rid])
```



Workload Balance among GPUs

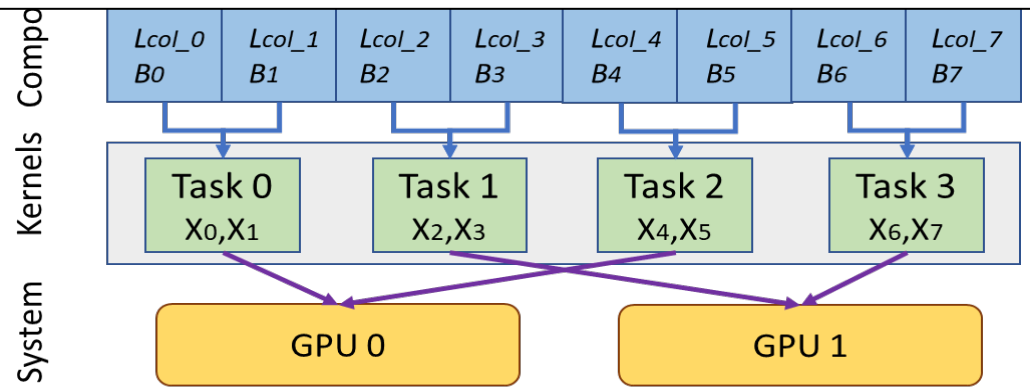


Three Approaches

- Balance components

Our design can achieve an average of 3.53× speedup on a DGX-1 system and 3.66 × speedup on a DGX-2 system over the Unified-Memory design, respectively.

- Task Based Workload Distribution
  - More tasks per GPU: workload becomes more balanced among GPUs
  - Less task per GPU: can exploit in-task data locality for better performance



Showcase II: Big Data Reduction via Lossy Compression

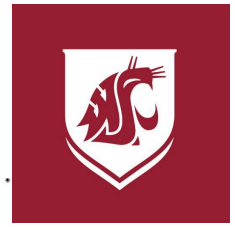
Trend of Supercomputing Systems
Gap Between Compute and I/O

The compute capability is ever growing while storage capacity and bandwidth are developing **more slowly** and not matching the pace.

supercomputer	year	class	PF	MS	SB	MS/SB	PF/SB
Cray Jaguar	2008	1 PFLOPS	1.75 PFLOPS	360 TB	240 GB/S	1.5k	7.3k
Cray Blue Waters	2012	10 PFLOPS	13.3 PFLOPS	1.5 PB	1.1 TB/S	1.3k	13k
Cray CORI	2017	10 PFLOPS	30 PFLOPS	1.4 PB	1.7 TB/S•	0.8k	17k
IBM Summit	2018	100 PFLOPS	200 PFLOPS	>10 PB••	2.5 TB/S	>4k	80k

PF: peak FLOPS MS: memory size SB: storage bandwidth
• when using burst buffer •• counting only DDR4 *Source: F. Cappello (ANL)*

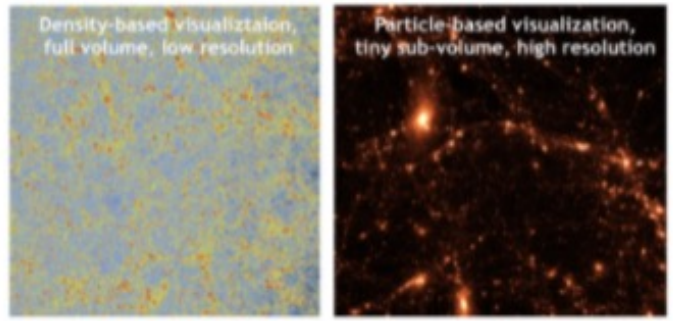
Table 1: Three classes of supercomputers showing their performance, **MS** and **SB**.



Scientific Simulations and Experiments: Flood of Data!

Today's scientific research using simulations or instruments produces extremely large datasets

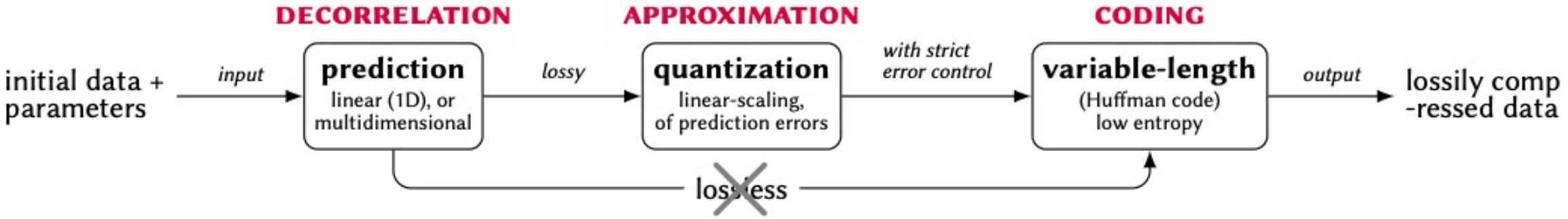
- Many datasets are in petabyte (PB = 10^{18} Bytes)!



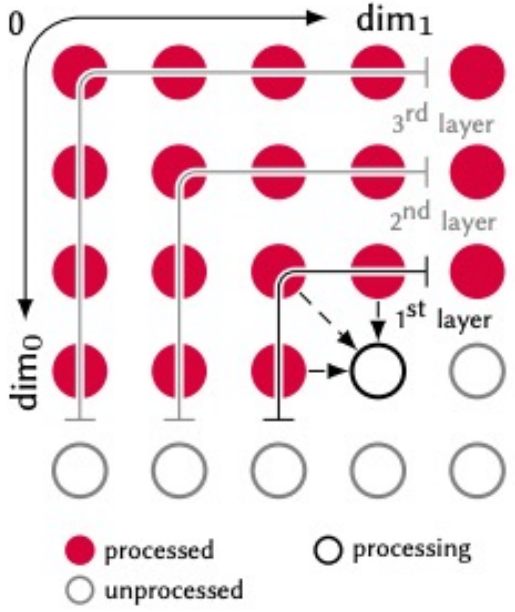
Cosmology Simulation

- HACC: hardware/hybrid accelerated cosmology code (twice IEEE/ACM Gordon Bell Prize Finalists)
- A total 20 PB data when simulating one trillion of particles
- Peta-scale system's file system ~ 20 PB
- Mira supercomputer has 26 PB FS, 20 PB / 26 PB ~ 80%
- NSF Blue Waters (1TB/s I/O bandwidth), 20 x 10^{15} / 10^{12} seconds (5h30m) to store the data
- Data reduction of about a factor of 10 is needed currently drop 9 snapshots over 10 (called decimation in time)

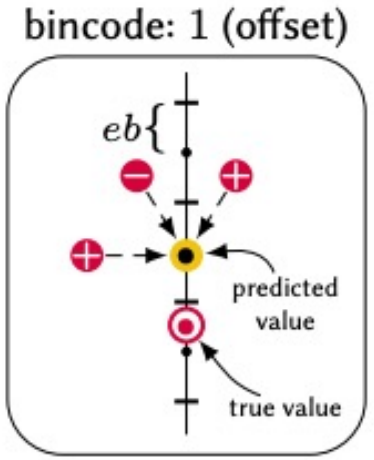
How SZ works?



- Lorenzo predictor allows arbitrary-dimensional prediction.
  - $\sum_{0\le k_{1\ldots d}\le n}^{k_{1\ldots d}\neq \mathbf{0}} \left\langle \prod_{j=1}^{d} (-1)^{k_j+1} \binom{n}{k_j} \right\rangle \cdot D_{x_1-k_1,\cdots,x_d-k_d}.$
  - Single-layer form Lorenzo predictor works the best generally [Tao et al. 2017]. 2D form:
    $\ell(D_{0,0}) = \mathrm{dot}\left(\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} D_{-1,-1} & D_{0,-1} \\ d_{-1,0} & D_{0,0} \end{bmatrix} \right).$
- Customized Huffman encoding
  - `sizeof(T)`-byte long symbol to Huffman code
  - high quantization quality (aggregated in center) makes Huffman coded bitstream more possible to further `gzip`



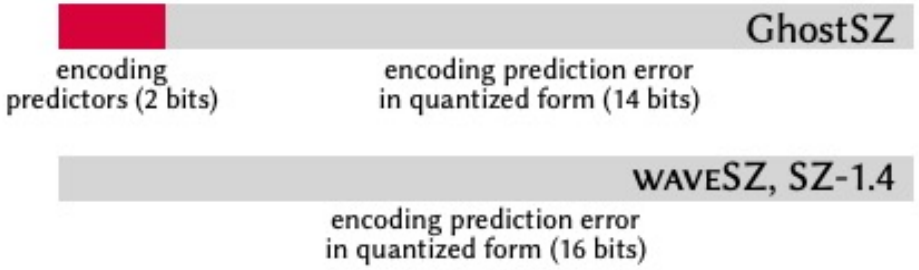
Lorenzo (ℓ) prediction



linear-scaling quantization

Issues with SZ and Its Current FPGA Implementation

- Low throughput of SZ
  - lack of parallelism: SIMD and SIMT cannot apply
- Limitations in FPGA GhostSZ
  - totally performance-driven design
  - 3 predictors in use, need extra bits to encode
  - **more** "workflow pipelines" (more resource)
  - **low** compression ratio



- New use scenarios of adopting FPGA
  - real-time processing; "inline processing" (Intel, 2018)
  - *ExaNet—an FPGA-based direct network architecture of the European exascale systems* [Ammendola et al. 2018]

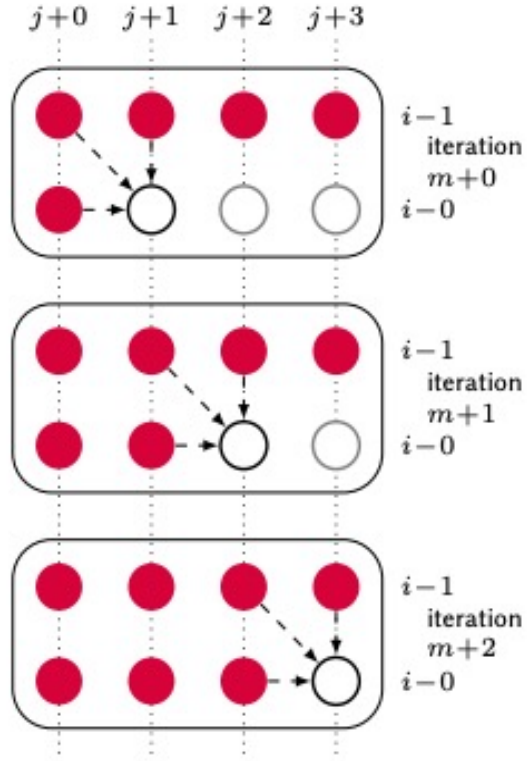


Figure 1: Loop-carried dependencies due to writeback.

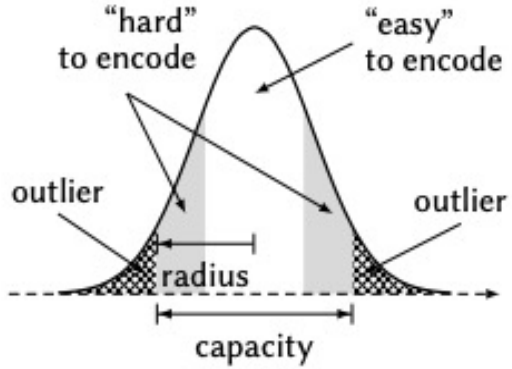


Figure 2: General distribution pattern of quantization code.

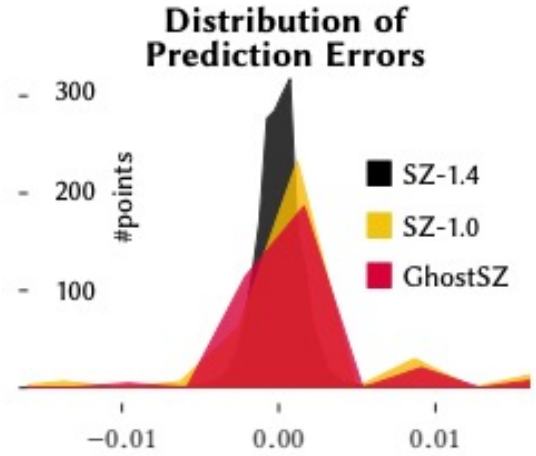
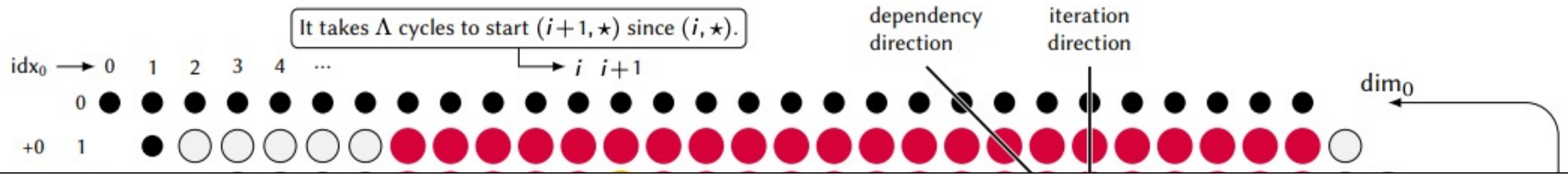


Figure 3: CESM-ATM CLDLOW.

Temporal-Spatial Mapping



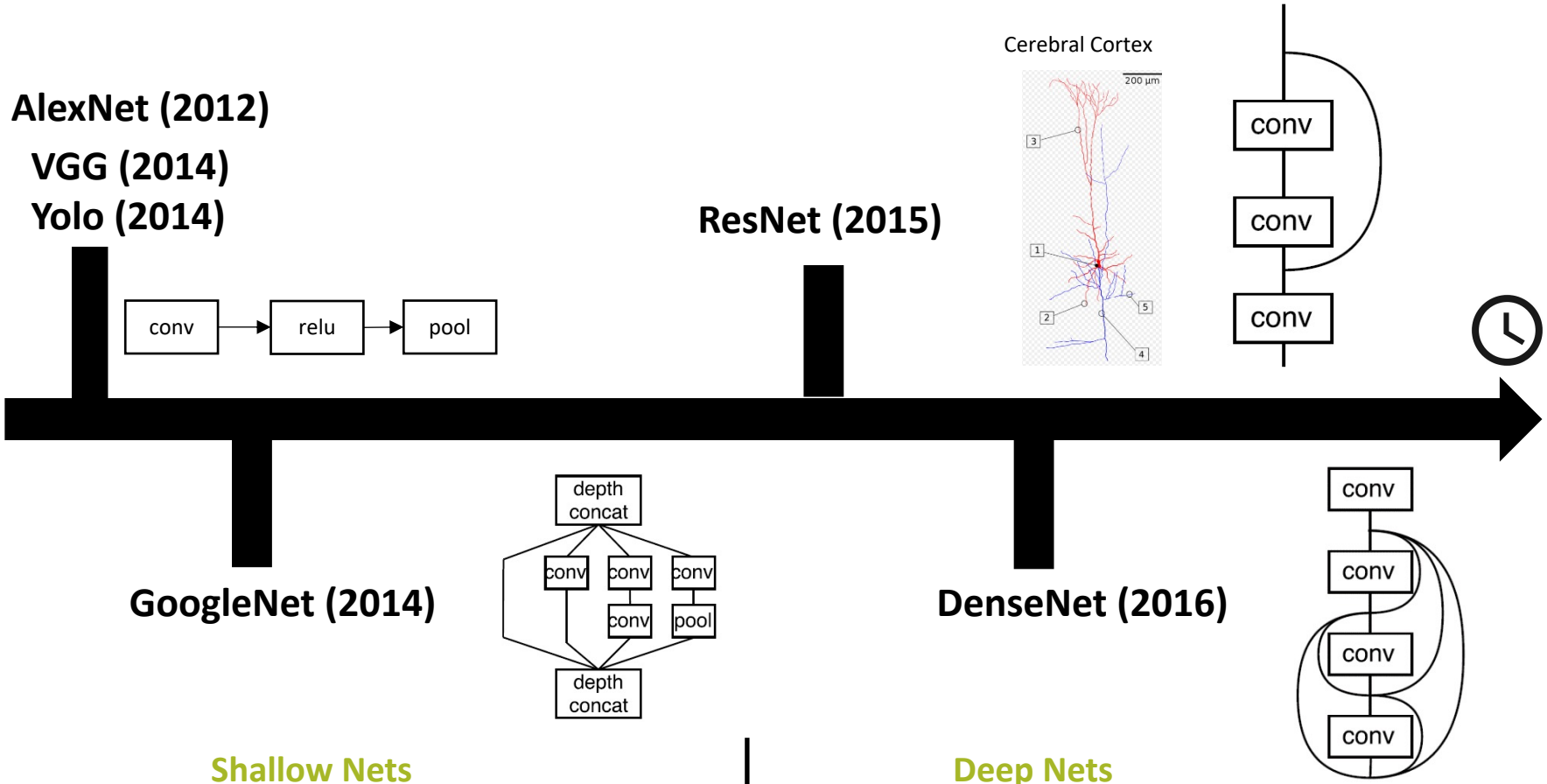
We evaluate on three real-world datasets from SDRB suite, showing 5.8× throughput on average, compared with the current FPGA implementation.

- FPGA + wavefront memory layout = more pipelining control
- Ideally, suppose Λ cycles to finish (prediction + quantization), no stall if ❶ II = 1, and ❷ (vertically) iterating over Λ points from (r, c) to $(r, c+1)$
- BODY ("perfect loop") unrolled with factor Λ (= vertical dimension) and II = 1

Showcase III: System ML for Accelerated Scientific Discovery

- Innovative software-hardware co-design strategies for emerging ML model architectures (*with UW, MIT*)
- Large-scale data-center and HPC ML services design and optimization (*with Alibaba Research*)
- Noise and randomness control in ML training (*with Google*)
- Resource-constrained training and inference (*with UW*)
- Neural Architecture Search and Tiny ML (*with Brown*)

CNN Architecture Evolution



Shallow Nets

AlexNet: (8 layers, top-5: 83.6%)
VGG: (19 layers, top-5: 92.7%)
Yolo: (26 layers, real time: 45 fps)

GoogleNet: (22 layers, top-5: 93.3%)

Deep Nets

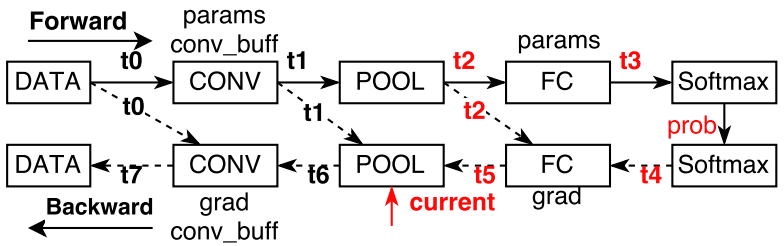
ResNet: (152 layers, top-5: 96.43%)

DenseNet: (103 layers, top-5: 94.7%)

Linear vs. Non-Linear Networks

LINEAR

AlexNet, VGG16 and VGG19

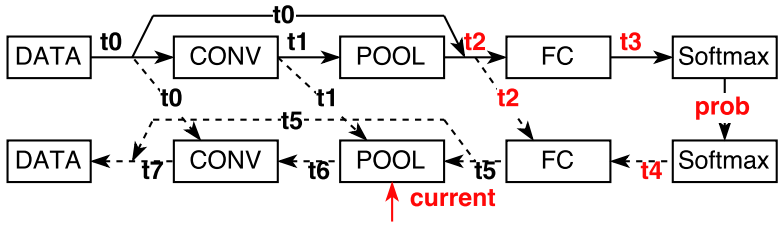


- Static Dependency
- Predictable

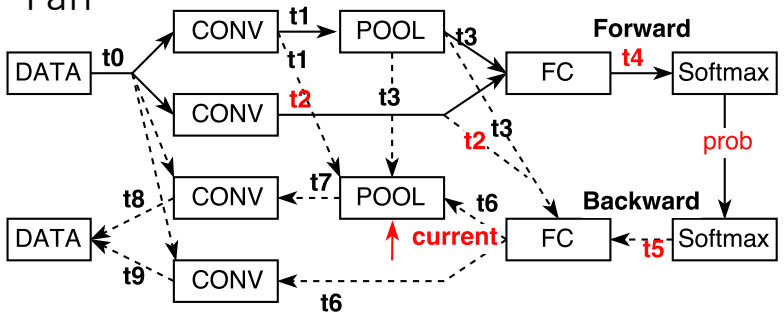
Non-LINEAR

ResNet, Inception and DenseNet

Join

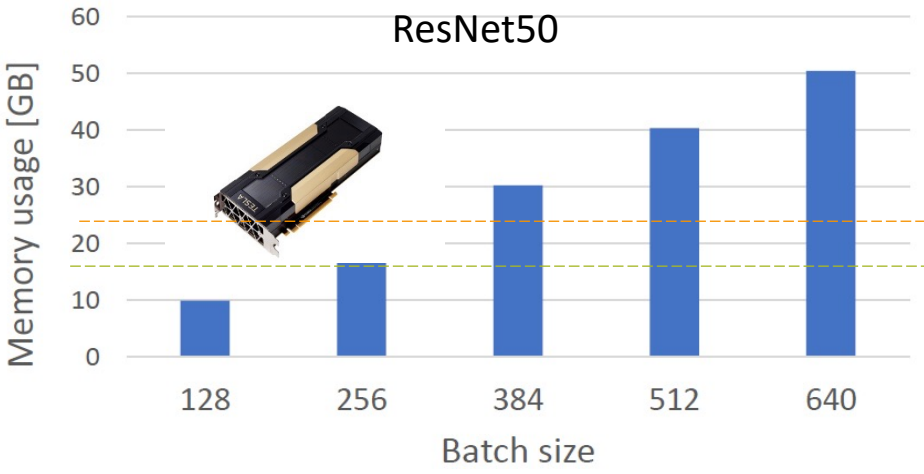


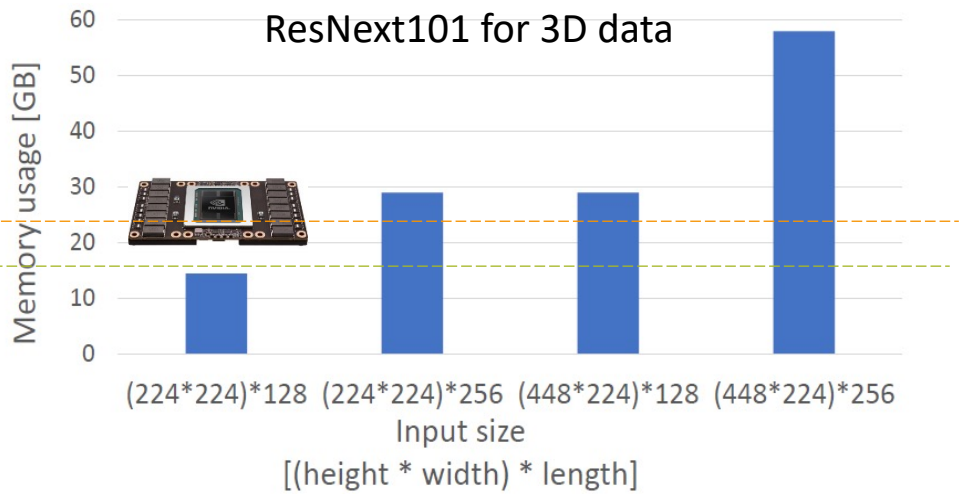
Fan

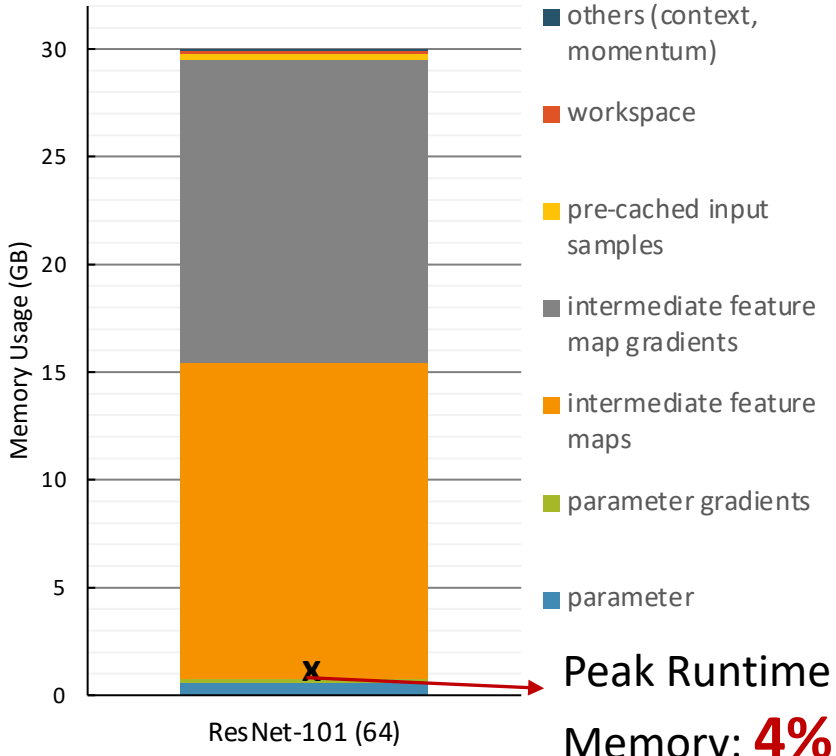


- Dynamic Dependency
- Unpredictable

Challenge: Memory Shortage





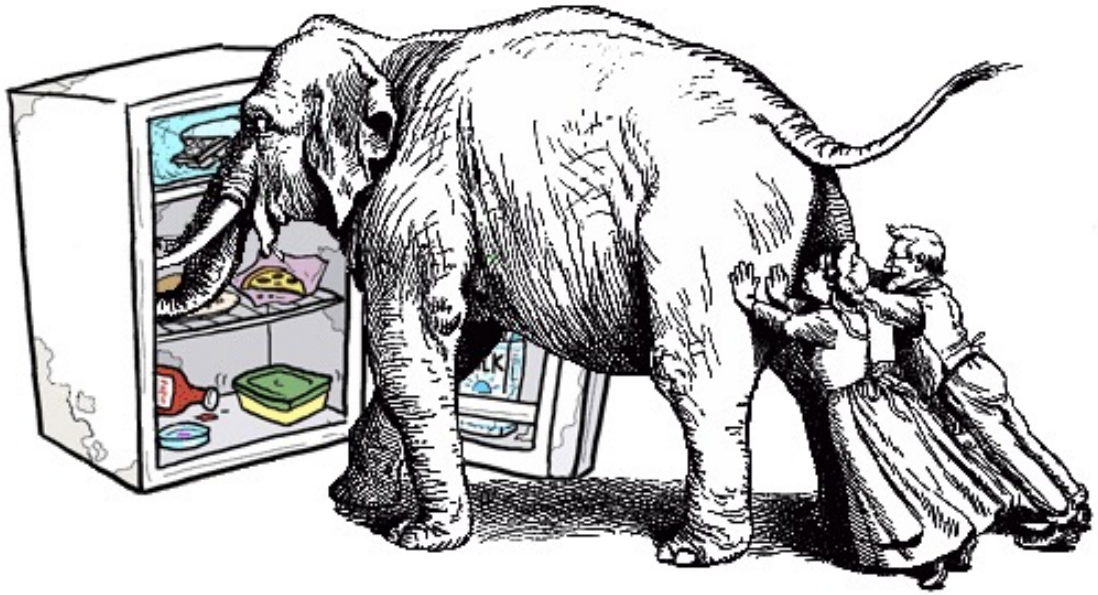


Peak Runtime Memory: **4%**

Negative Impact

- ❑ Ultra Deep Net is untrainable
- ❑ Network-wide memory allocation is extremely wasteful
- ❑ Long training time affects hyper-paramaters tuning
- ❑ Significantly restrict minibatch size tuning, affecting parallelism, scalability and energy efficiency

Our Goal



Largest Accelerator Memory:
Tens of GBs

training
high speed

Training Memory Request:
Hundreds of GBs

State-of-the-art Frameworks

Caffe

torch

TensorFlow

mxnet

(1) No Liveness Analysis,
(2) Limited Support in Non-linear Networks

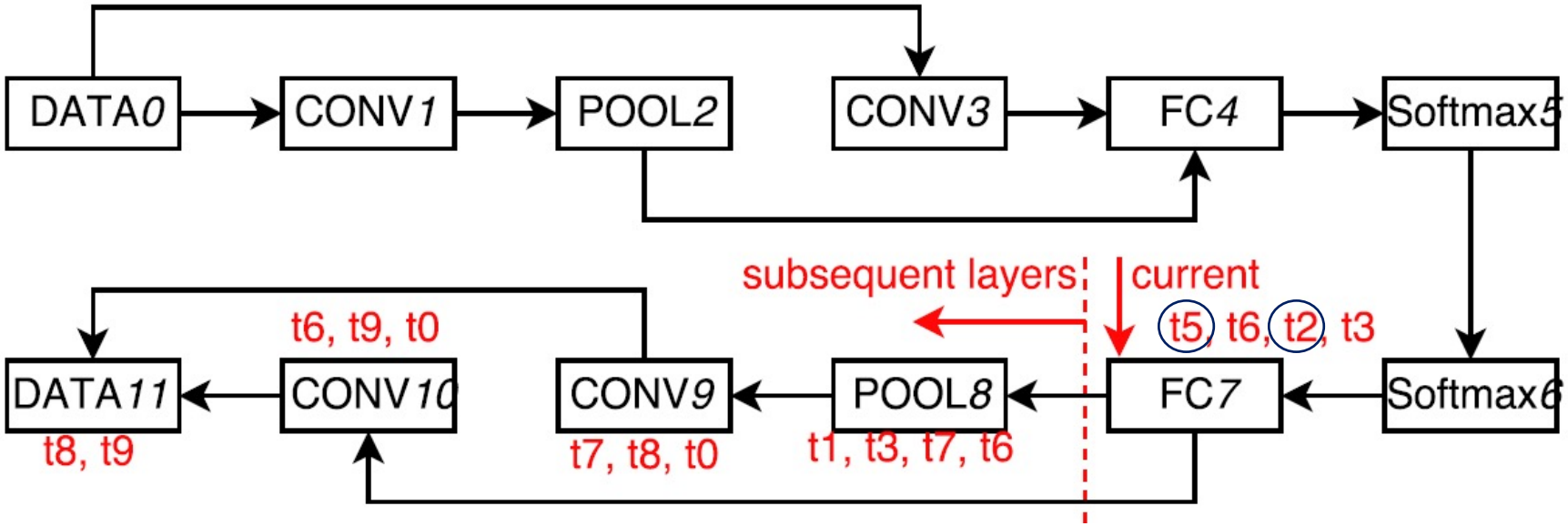
(1) Inefficient Memory Offloading
(2) No performance-memory trade-off
(3) No minibatch-LR tuning

(1) Inefficient Memory Offloading
(2) Speed-Centric
(3) No minibatch-LR tuning

SuperNeurons Framework

- Dynamic memory provisioning framework for deep non-linear networks
- Recognizing non-uniform memory distribution across layers
- Runtime bound the minimal peak memory usage ($peak_m = \max(l_i)$), at layer-wise granularity

Opportunity 1: Physical Memory Reuse



Core Idea

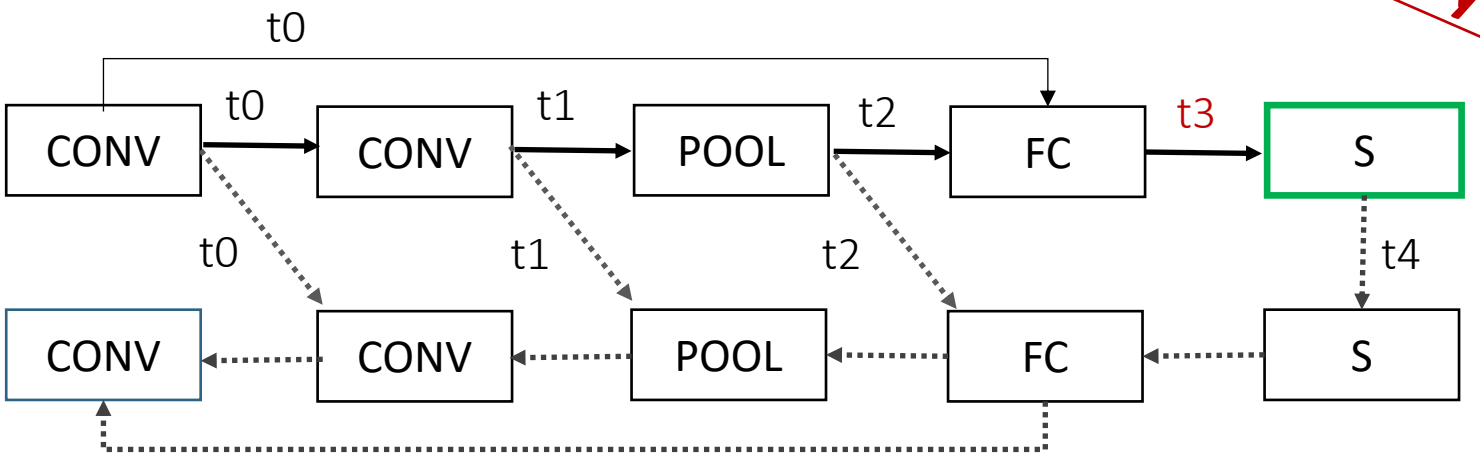
Reuse the same physical memory at different time partitions

Liveness Analysis

Runtime Peak Memory (l_{peak})

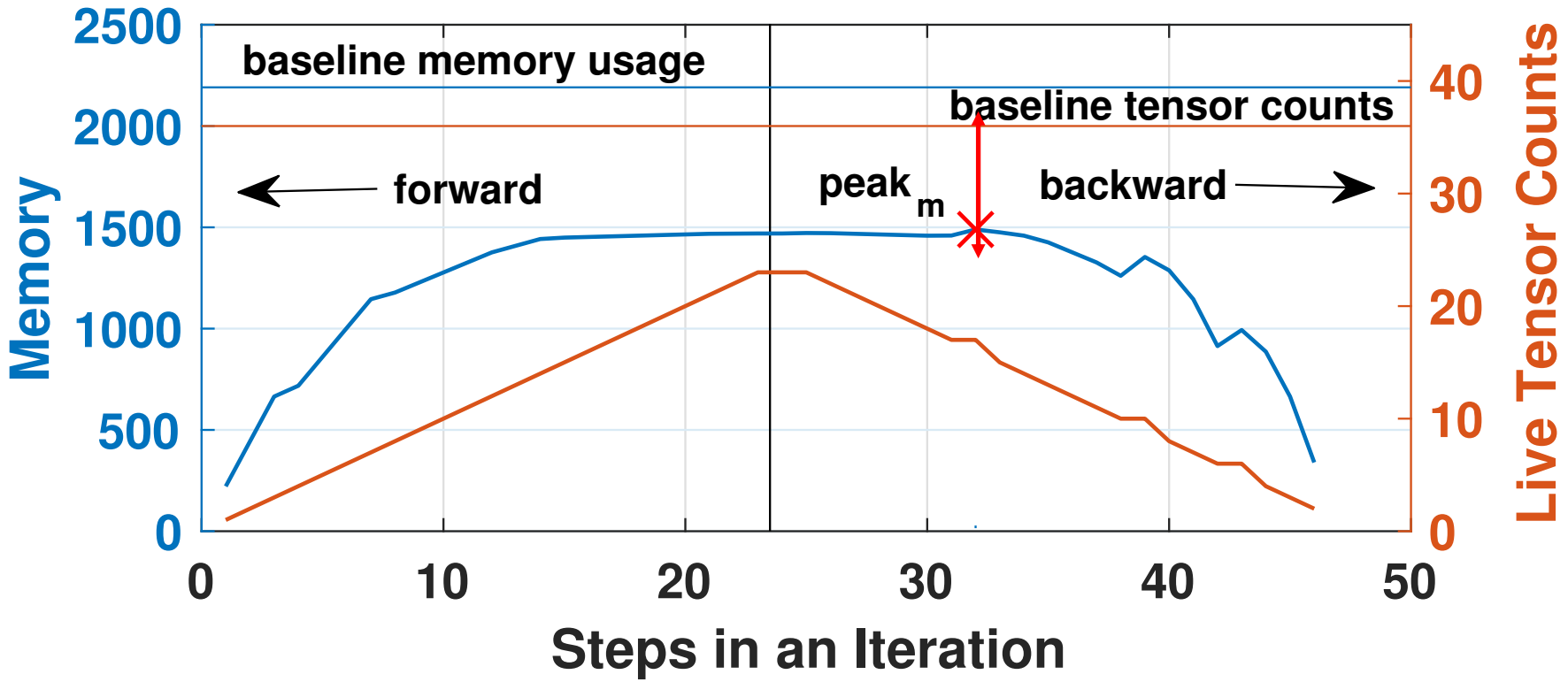
Peak Runtime Memory !

Step 5



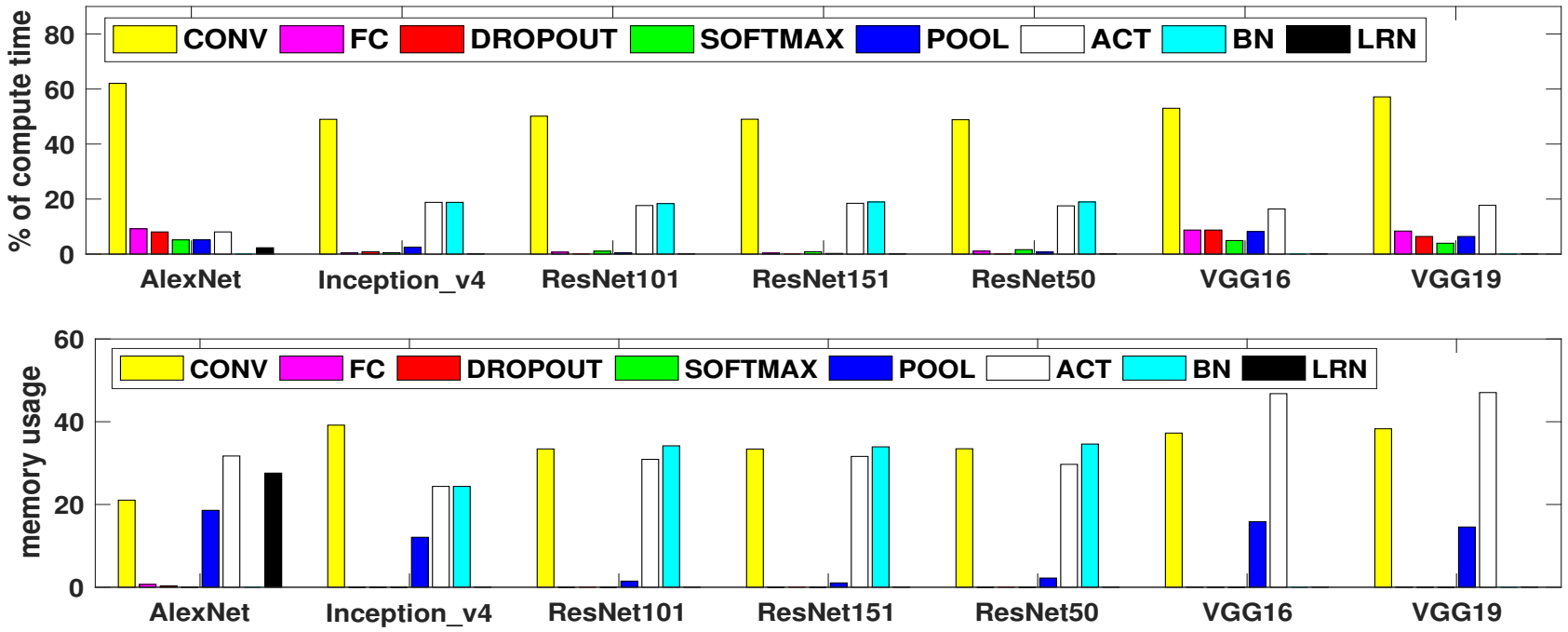
Liveness Analysis	Baseline
t0, t1, t2, t4,	t0, t1, t2, t3, t4, t5, t6, t7, t8
4 Tensors	9 Tensors

Example: Liveness analysis on AlexNet



It reduces more than 1/3 of memory requirement for AlexNet

Opportunity 2: Computation Pattern



Computation and Memory intensity differ across the layers:

Layers	Comp (%)	Memory (%)	Checkpoint? (Y/N)
CONV	~50%	~50%	Y
POOL, ACT, BN, LRN	~30%	~50%	Y, but requires opt
DROPOUT, SOFTMAX, FC	~20%	~1%	N

Unified Tensor POOL (UTP)

Key Operations:

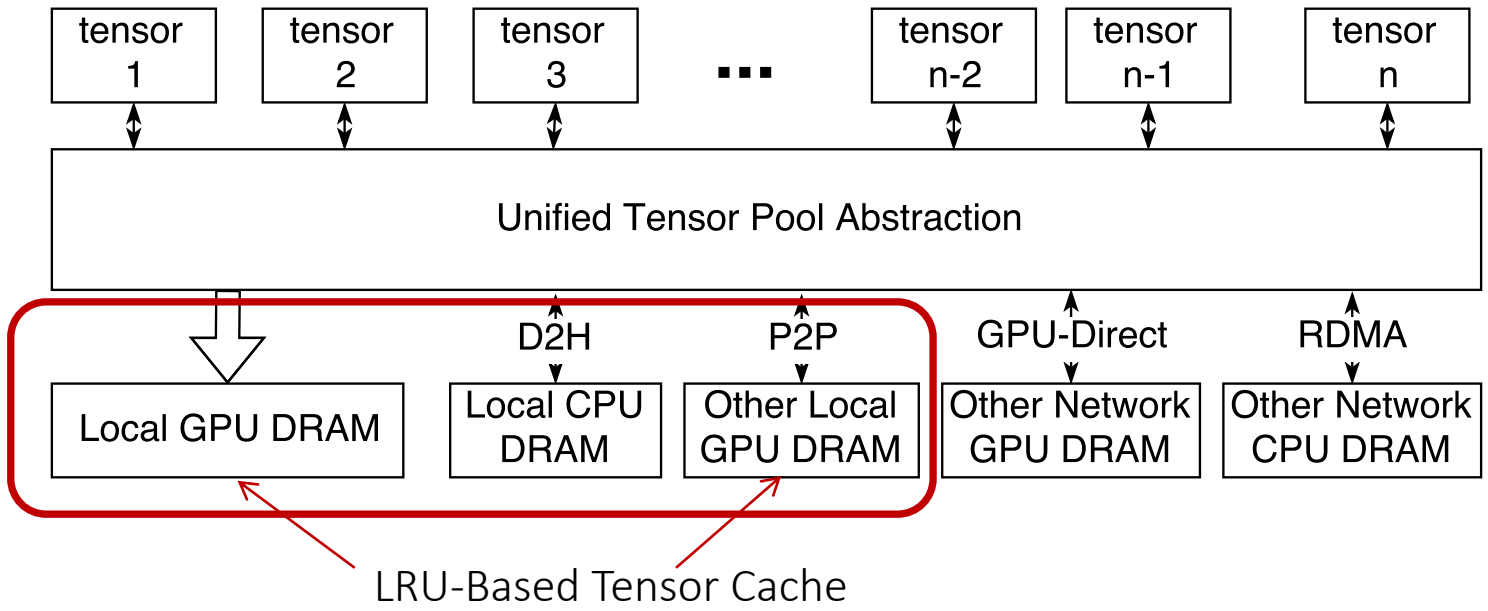
Checkpointing

- Offload outputs of “Y” layers to external memory buffers

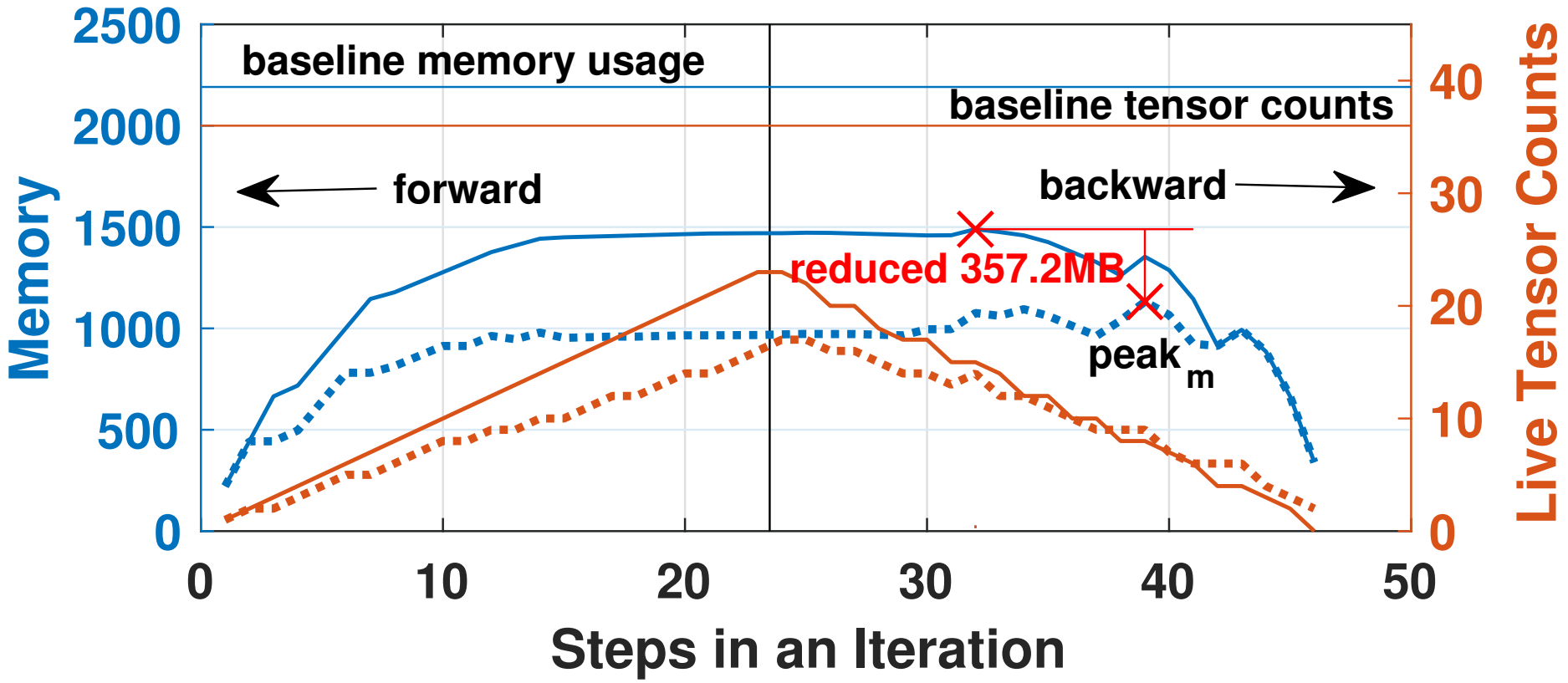
Pre-fetch

- Retrieve outputs of “Y” layers back to accelerator memory

Extensible to Various Physical Memory Pools

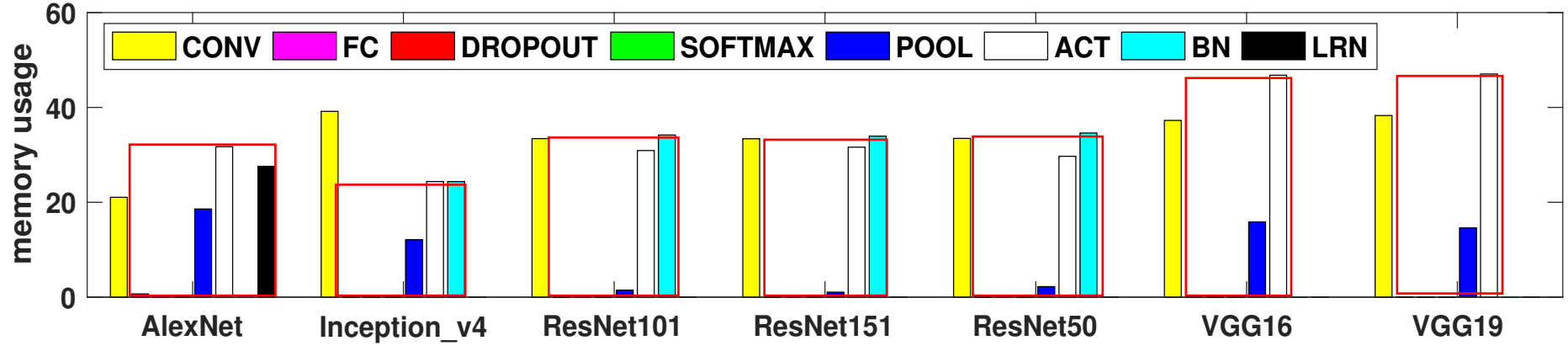


Example: Liveness Analysis + UTP on AlexNet

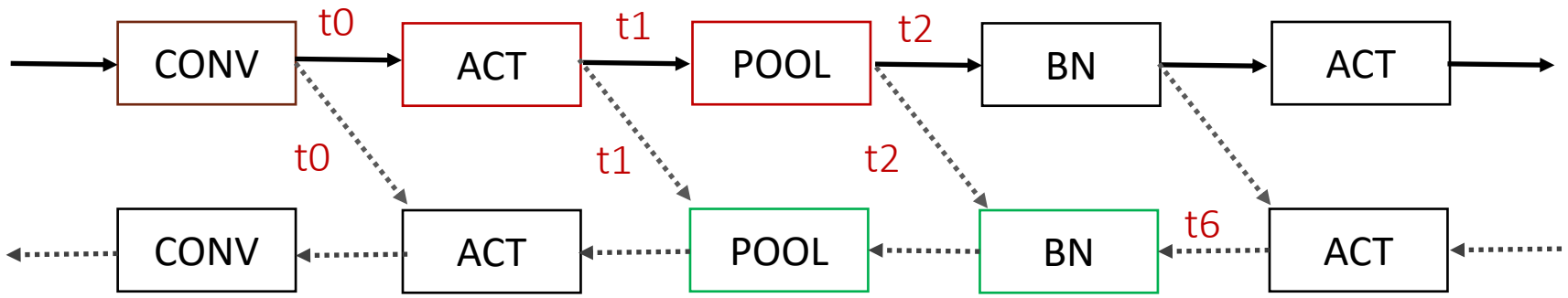


It further reduces another 1/6 of memory requirement for AlexNet

Opportunity 3: Store vs. Recompute

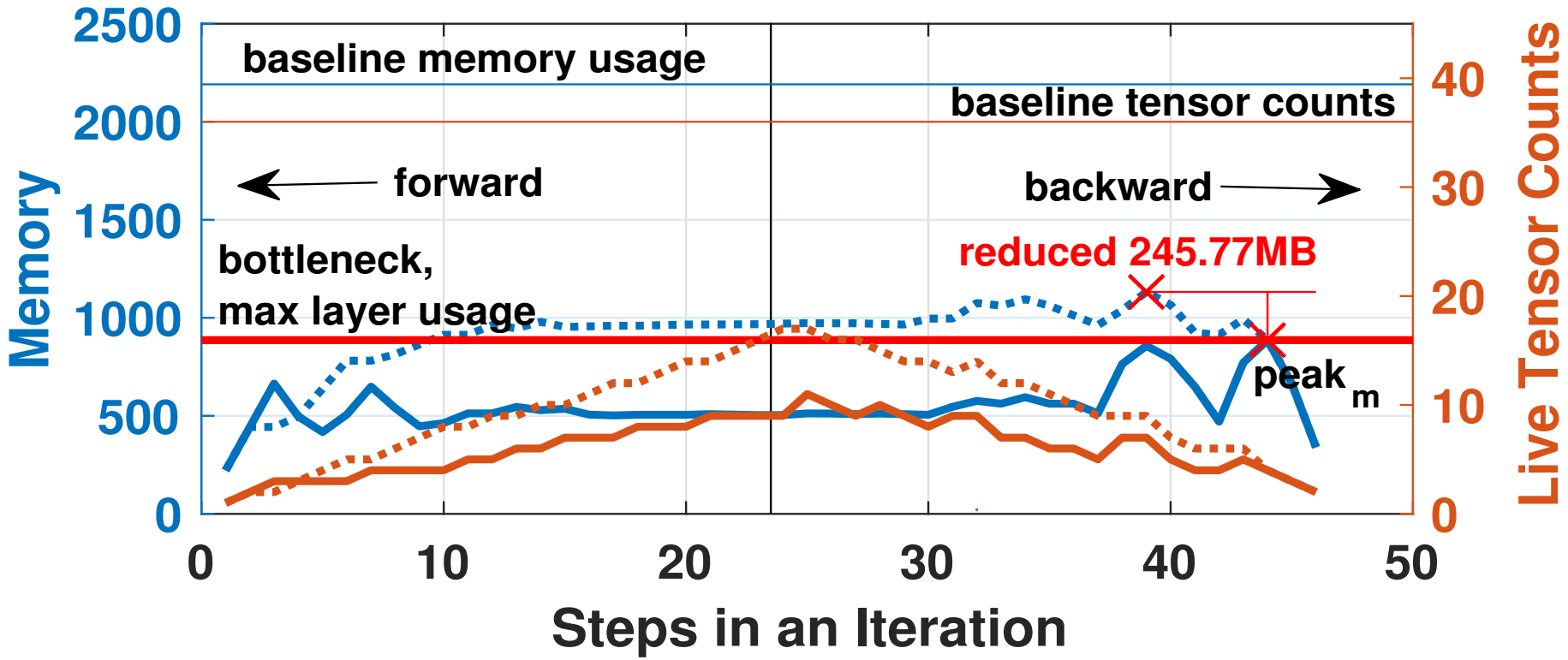


Observation: POOL, ACT, LRN and BN forward computation only accounts for less than 10% of total time, but with 50% memory consumption.



Memory-Centric (recompute dependencies) vs. Speed-Centric(reuse)

Example: Liveness Analysis + UTP on AlexNet + Cost-Aware



Memory is bounded by the layer peak and peak is further reduced for AlexNet

Many other works in this direction

- Machine Learning Optimizing Compiler vs. XLA and TVM (ASPLOS'22, under review, A*)
- Bayesian accelerator design for cloud and autonomous driving (MICRO'20, MICRO'21, A*)
- Large-scale Bayesian inference system for Argonne's genome analysis application and cosmic tagger application (Supercomputing'21, A*)
- LSTM and Transformer's accelerator design (ISCA'21, HPCA'21, A*)
- Capsule network acceleration design (HPCA'2020, TPDS architecture research highlight 2020, A*)
- Reinforcement learner for interactive virtual reality system design (ASPLOS'21, A*)

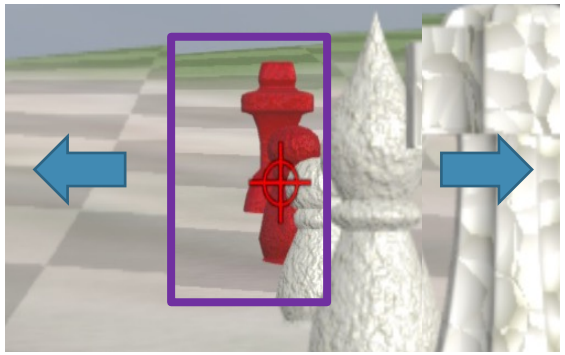
Edge/IoT/Embedded Scenarios

- Real time constraint, low power , low storage require better quality of data;
- We do not have luxury of even powerful chips; everything is working under a tighter constraints;
- We have to overperform our design to match users' demand;
- We have to squeeze every bit of efficiency out of our design;
- Better optimization, better hardware utilization, better compatibility with cloud, better security mechanisms, etc.

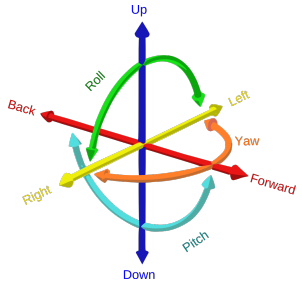
Data-centric Design is very important !

Showcase IV: Lightweight Reinforcement Learning Based VR SoC

- Strong relationship between users' input and rendering workload

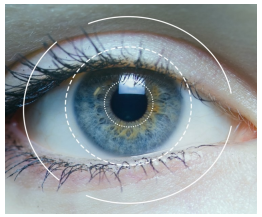


- Q-learning Based Eccentricity Controller using User Input



6DoF of Head

+



Fovea Movement

Index

6 bits for head movement
2 bits for fovea position

- Index the user input to simplify the design complexity for mobile devices

Software-Only Prototyping on Unity

Collaborative Foveal Rendering

Local-only Rendering

FSA

Contact me if you are interested in our projects!

HPC
Memory and Data Centric Accelerator Design
Quantum Compiler and Architecture Design
…

Shuaiwen.song@Sydney.edu.au
https://shuaiwen-leon-song.github.io/

Australian Government
Australian Research Council

Google™

Alibaba Cloud