HPDC’22 Summary

June 27 – July 1 2022

Norbert Egi
Infrastructure Software

Agenda

Session 1: Data Centers and HPC Systems

- DAOS: Data Access-aware Operating System
- FPVM: Towards a Floating Point Virtual Machine
- Lifting and Dropping VMs to Dynamically Transition Between Time- and Space-sharing for Large-Scale HPC Systems

Session 2: HPC Memory, I/O, and Storage Systems

- Access Patterns and Performance Behaviors of Multi-layer Supercomputer I/O Subsystems under Production Load
- NVMe-oAF: Towards Adaptive NVMe-oF for IO-Intensive Workloads on HPC Cloud
- Capri: Compiler and Architecture Support for Whole-System Persistence

Session 3: Reliability and Scheduling

- Understanding Memory Failures on a Petascale Arm System
- **SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning**
- Holmes: SMT Interference Diagnosis and CPU Scheduling for Job Co-location
- TLPGNN: A Lightweight Two-Level Parallelism Paradigm for Graph Neural Network Computation on GPU

Agenda

Session 4: HPC Algorithms

- TAC: Optimizing Error-Bounded Lossy Compression for Three Dimensional Adaptive Mesh Refinement Simulations
- Communication-aware Sparse Patterns for the Factorized Approximate Inverse Preconditioner
- Ultra-fast Error-bounded Lossy Compression for Scientific Dataset
- Optimizing the Bruck Algorithm for Non-uniform All-to-all Communication

Session 5: HPC Toolchains, Traces, and More

- SciStream: Architecture and Toolkit for Data Streaming between Federated Science Instruments
- Machine Learning Assisted HPC Workload Trace Generation for Leadership Scale Storage Systems
- PROV-IO: An I/O-Centric Provenance Framework for Scientific Data on HPC Systems

Session 6: Cloud Computing and Machine Learning

- Locality-aware Load-Balancing For Serverless Clusters
- Practical Efficient Microservice Autoscaling with QoS Assurance
- **Hare: Exploiting Inter-job and Intra-job Parallelism of Distributed Machine Learning on Heterogeneous GPUs**
- **Efficient Design Space Exploration for Sparse Mixed Precision Neural Architectures**

Selected Papers of Key Interest

- SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning
- Efficient Design Space Exploration for Sparse Mixed Precision Neural Architectures

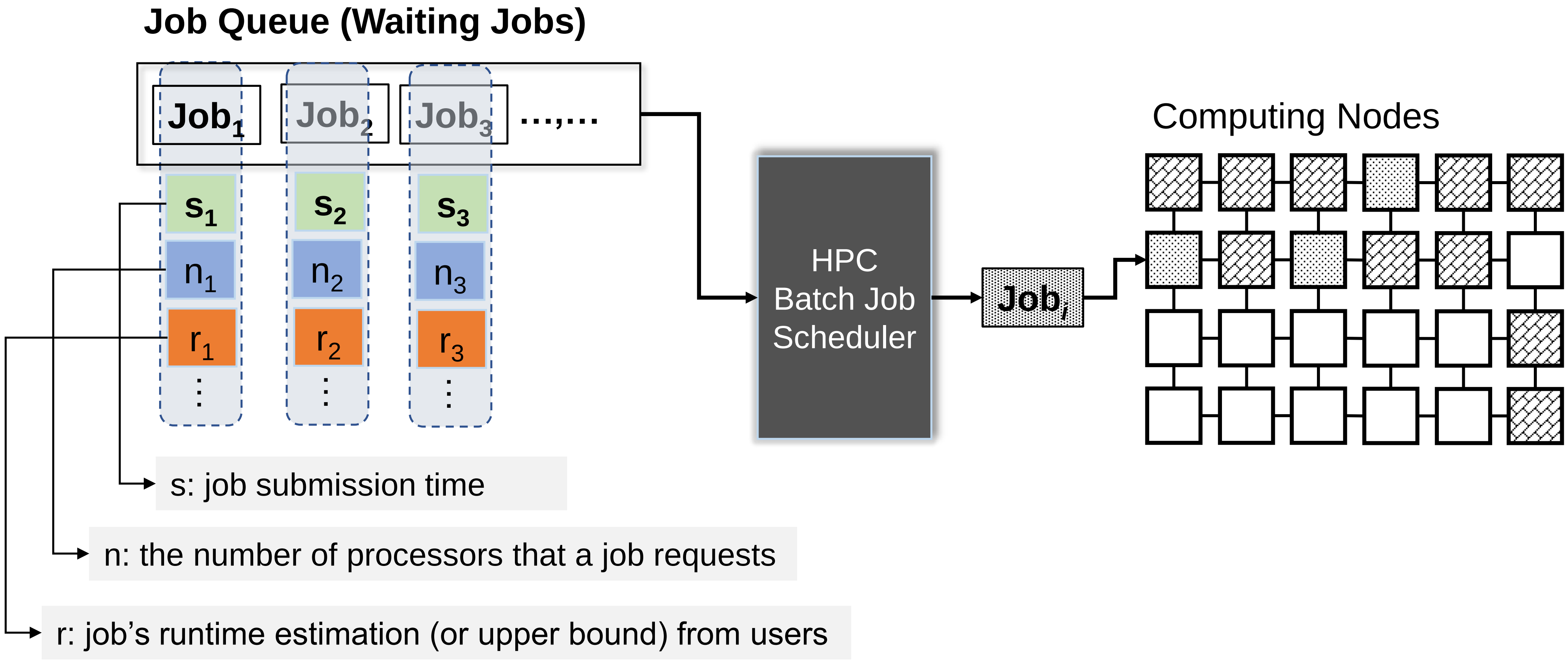
SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning

Di Zhang[1], Dong Dai[1], Bing Xie[2]

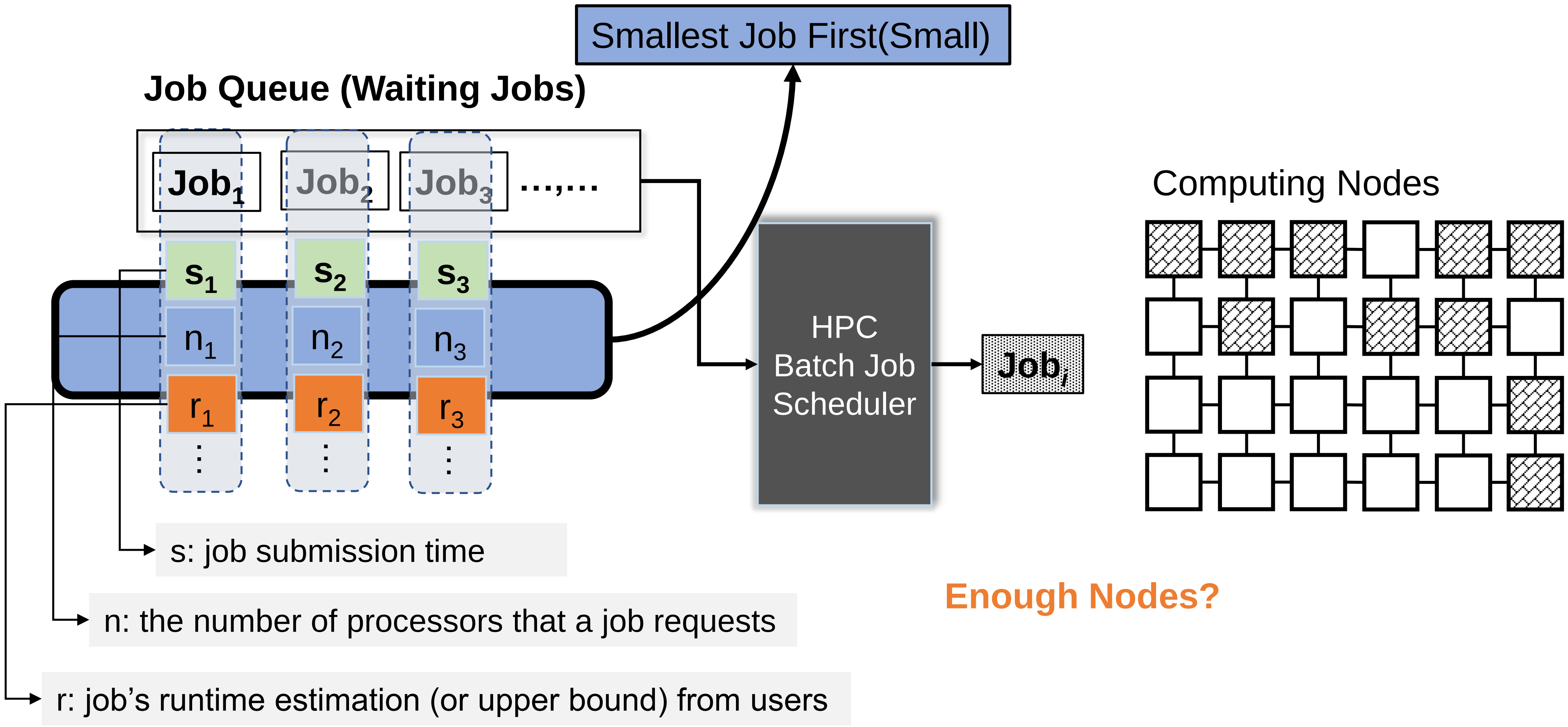
[1]University of North Carolina at Charlotte

[2]Oak Ridge National Laboratory

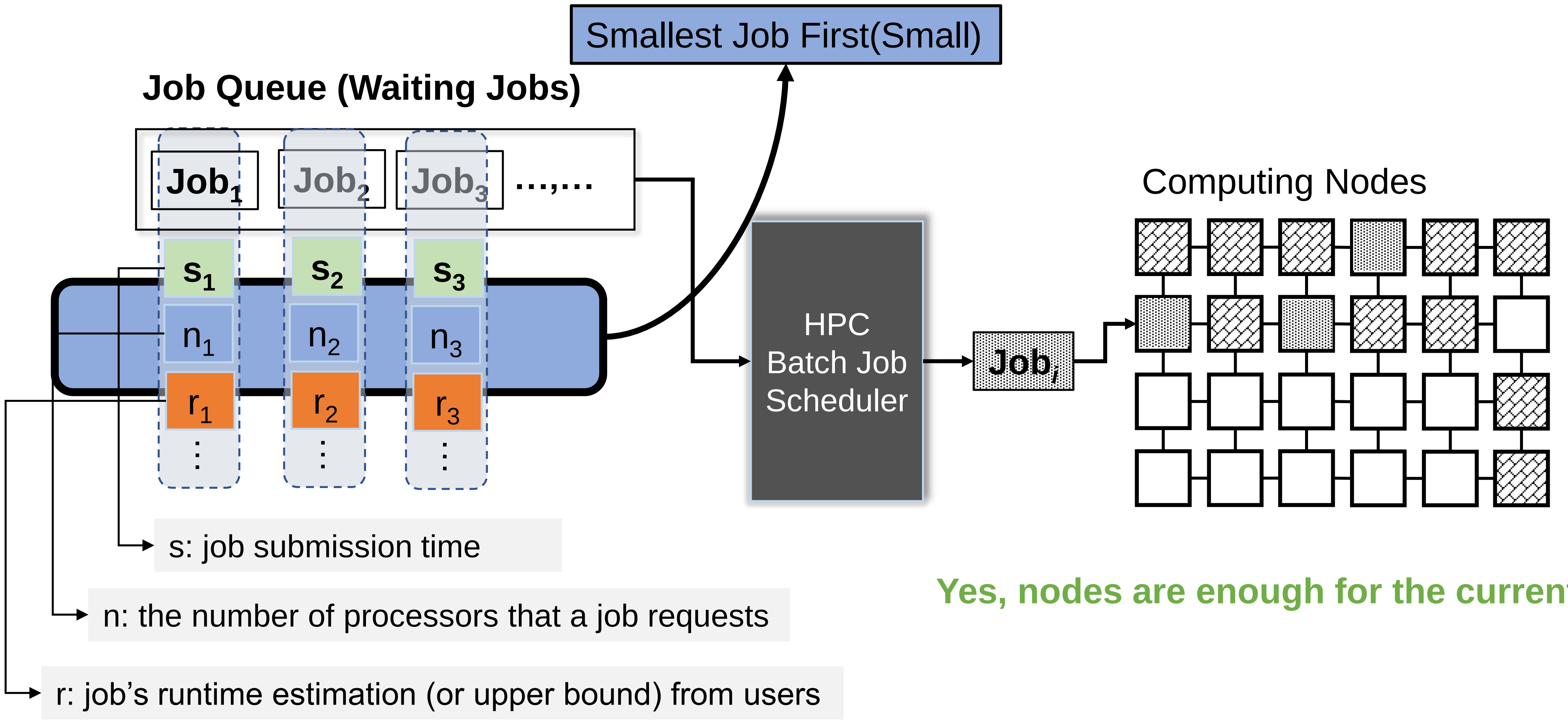
HPC Batch Job Scheduler



HPC Batch Job Scheduler

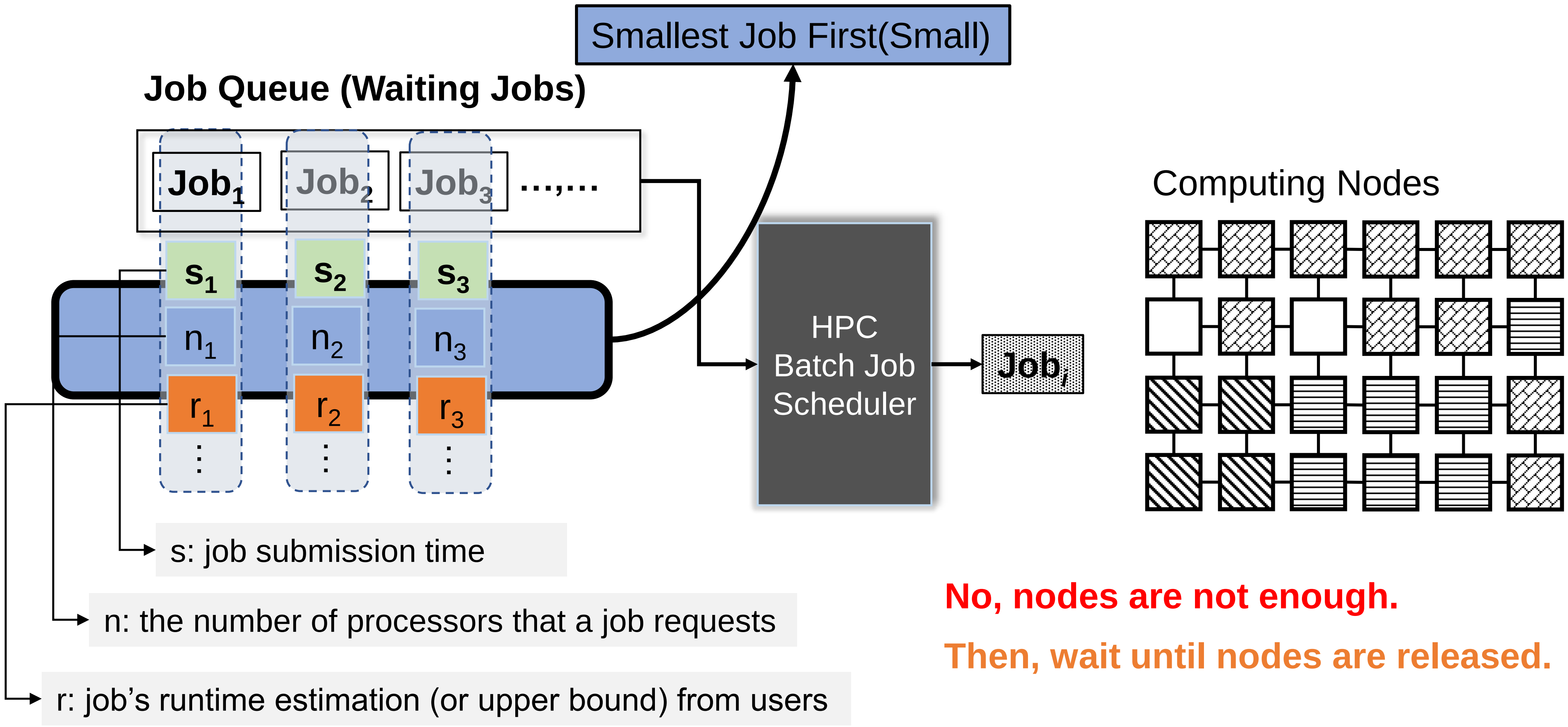


HPC Batch Job Scheduler



Yes, nodes are enough for the current job.

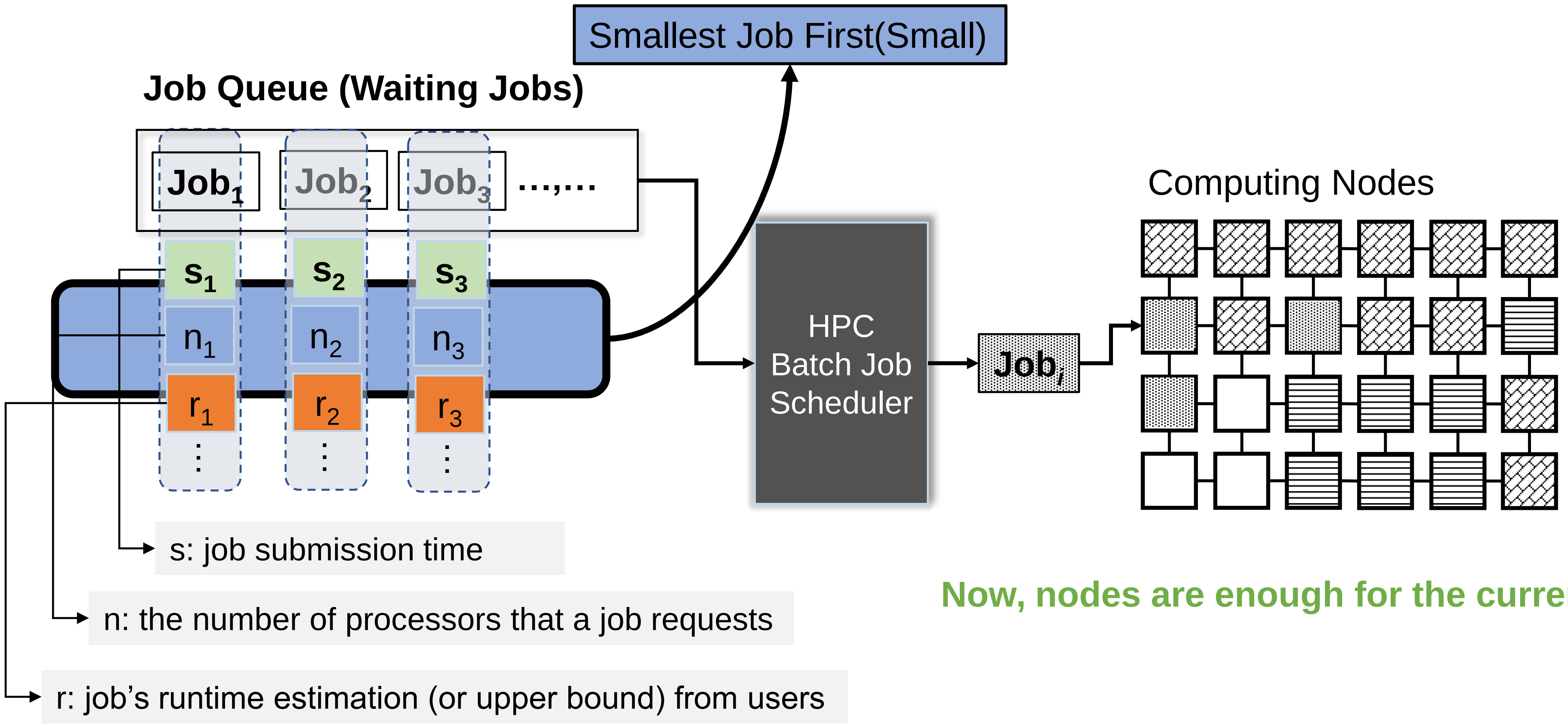
HPC Batch Job Scheduler



No, nodes are not enough.

Then, wait until nodes are released.

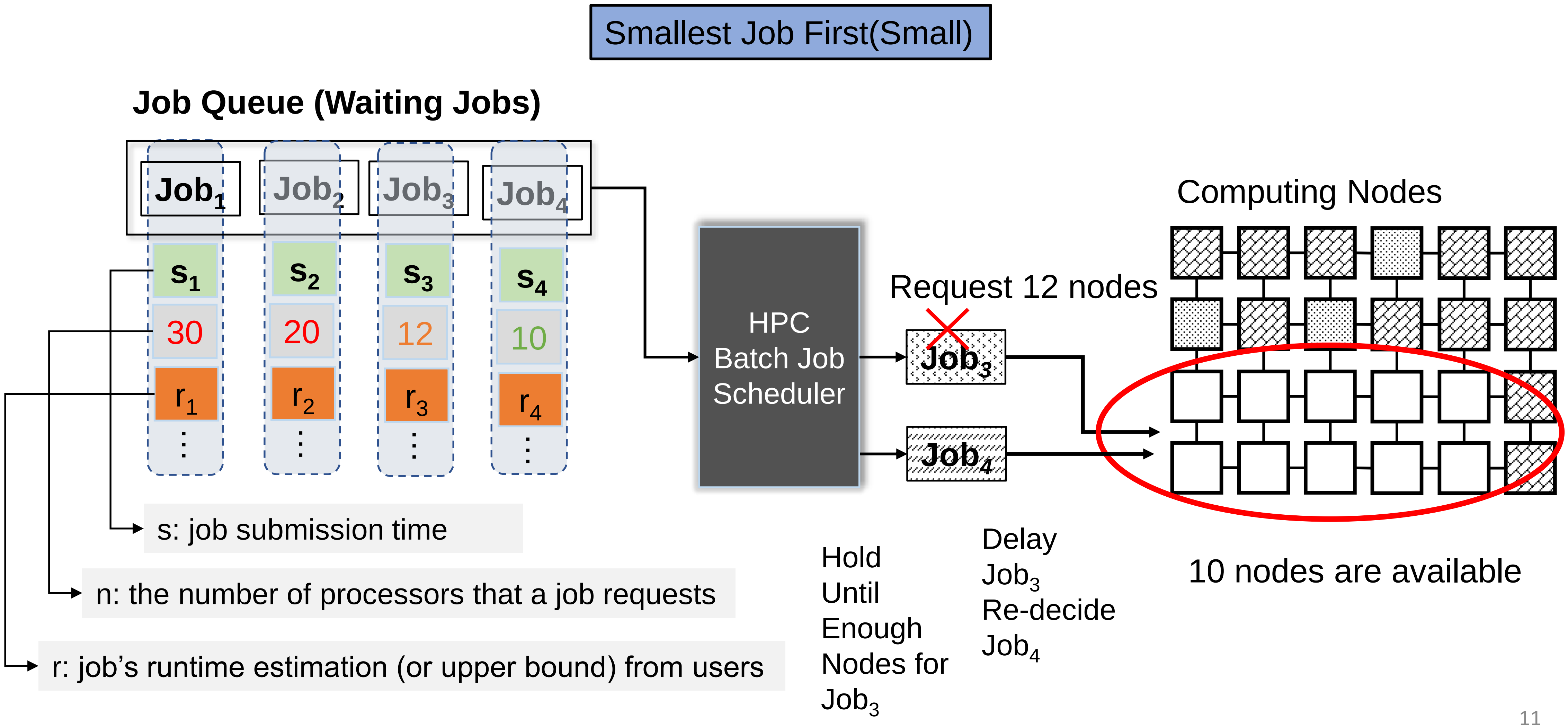
HPC Batch Job Scheduler



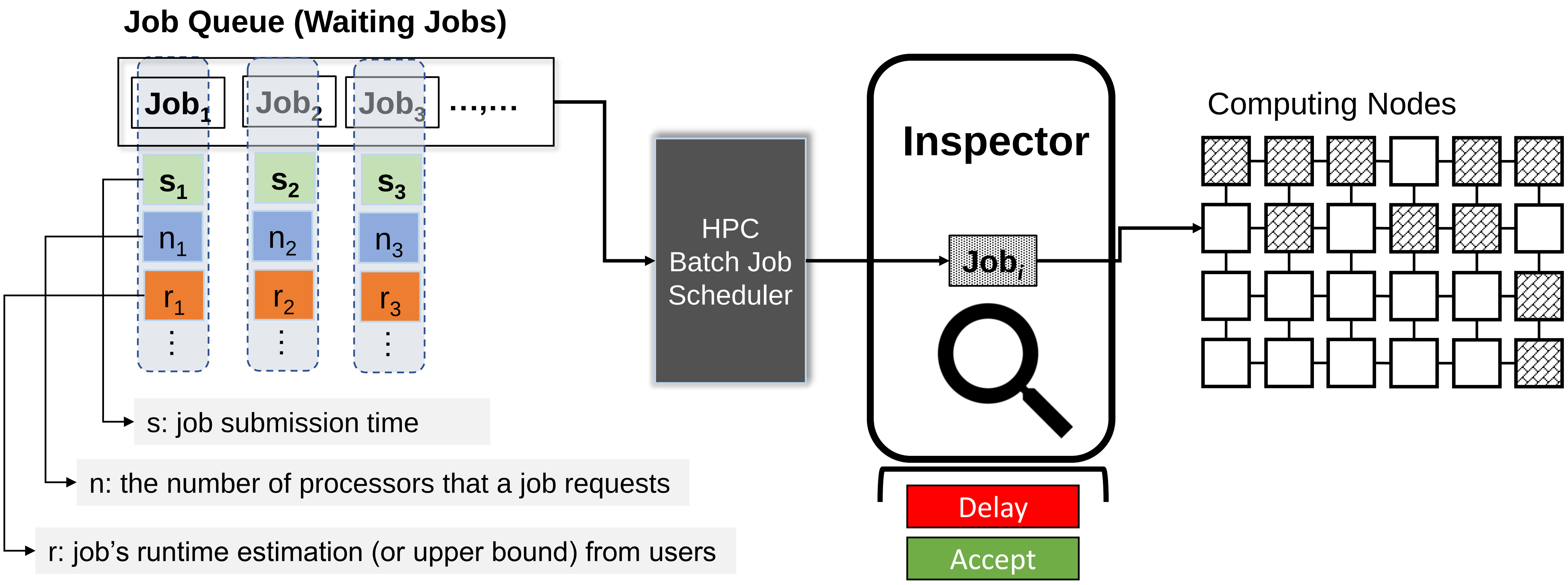
Now, nodes are enough for the current job.

DIRLAB

Motivation Example



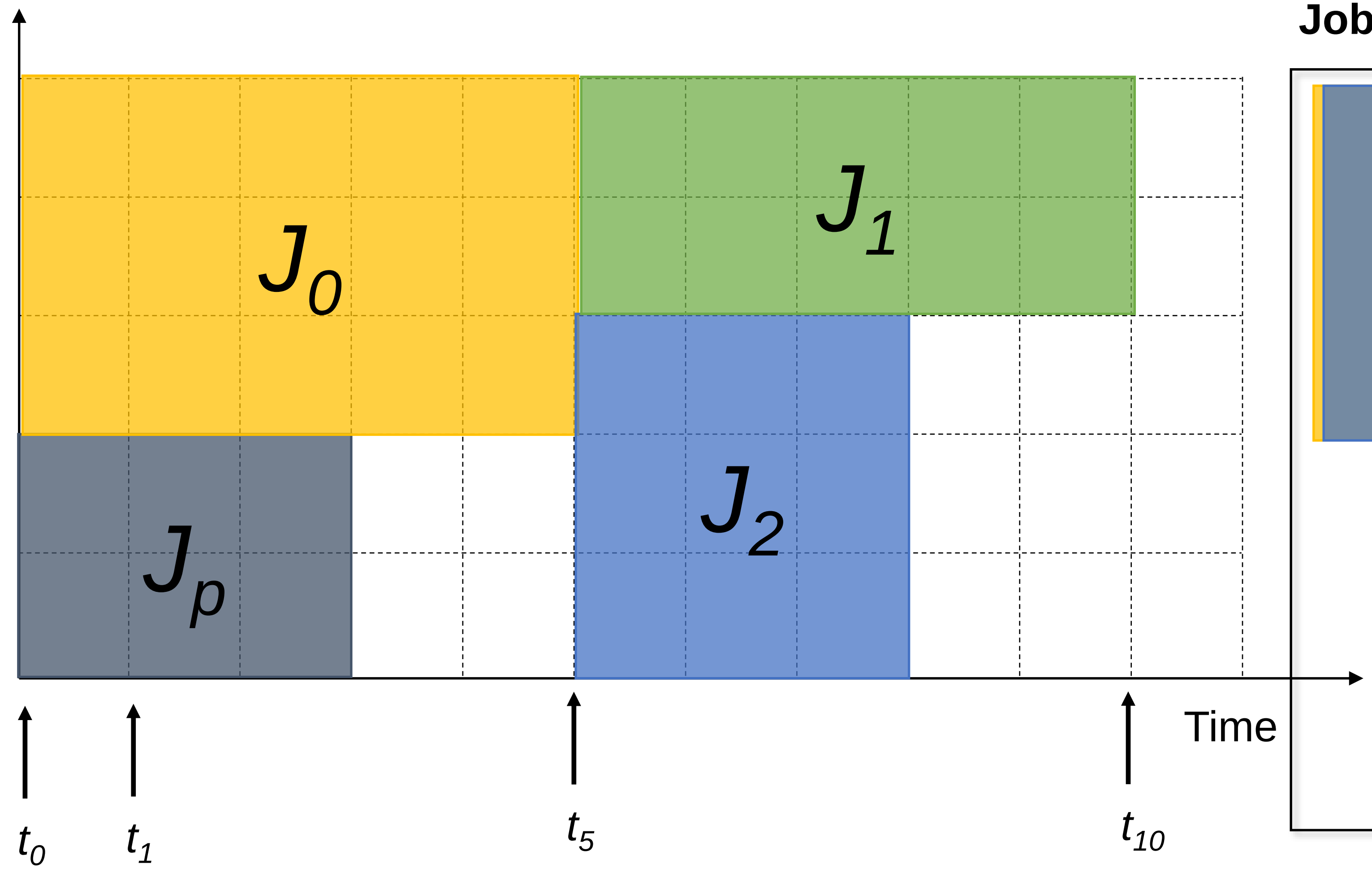
Motivation


Job Queue (Waiting Jobs)
Job_1
Job_2
Job_3
…,…
s_1
n_1
r_1
s_2
n_2
r_2
s_3
n_3
r_3
s: job submission time
n: the number of processors that a job requests
r: job's runtime estimation (or upper bound) from users
HPC
Batch Job
Scheduler
Inspector
Job_i
Delay
Accept
Computing Nodes

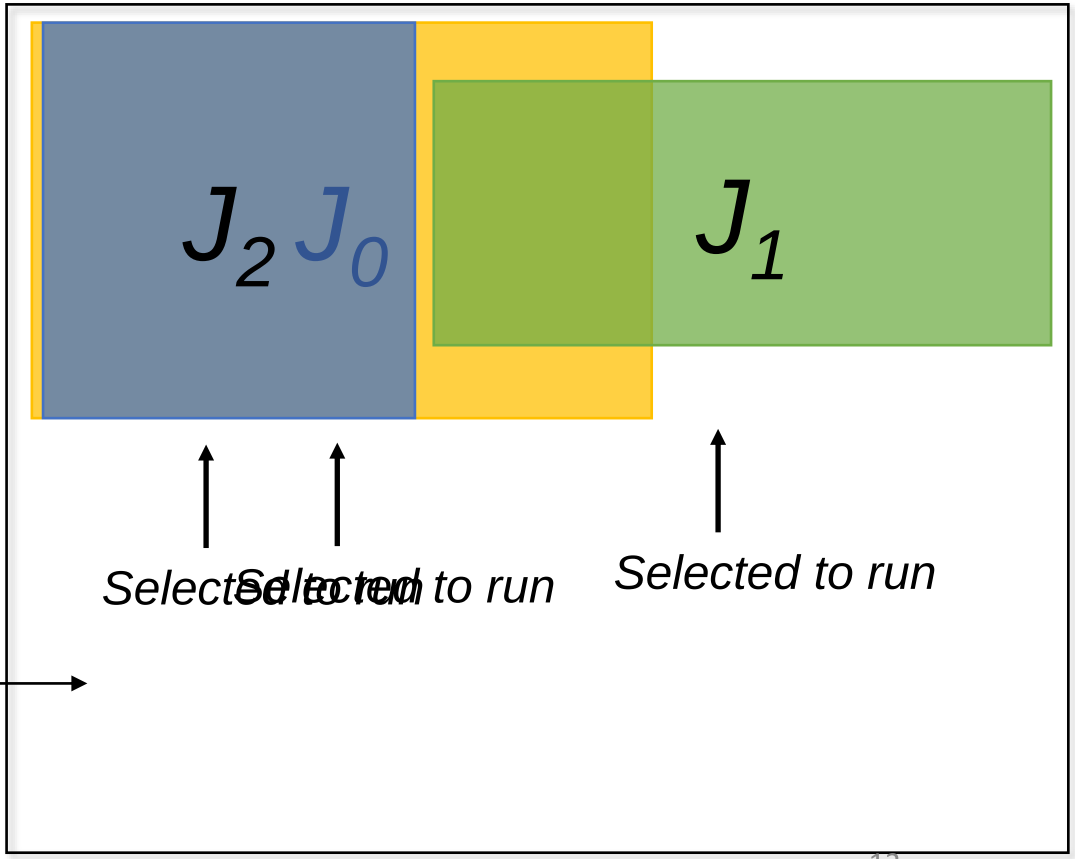
Shortest Job First(SJF)

Without Inspector

Completion Time : 10



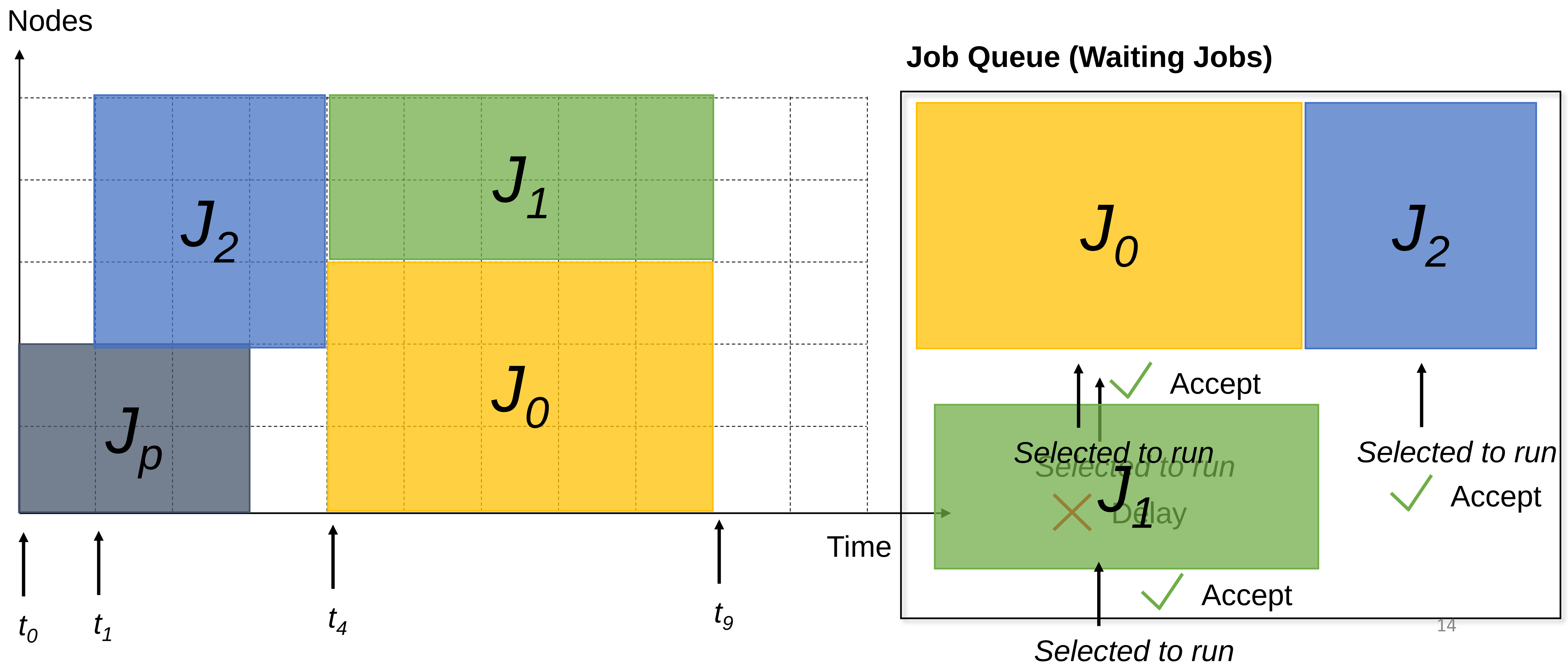
Job Queue (Waiting Jobs)



With Inspector

Shortest Job First(SJF)

Completion Time : 9



Challenges

Current Status

Future Job Arrival

Understanding of historical data

Attributes of the selected job

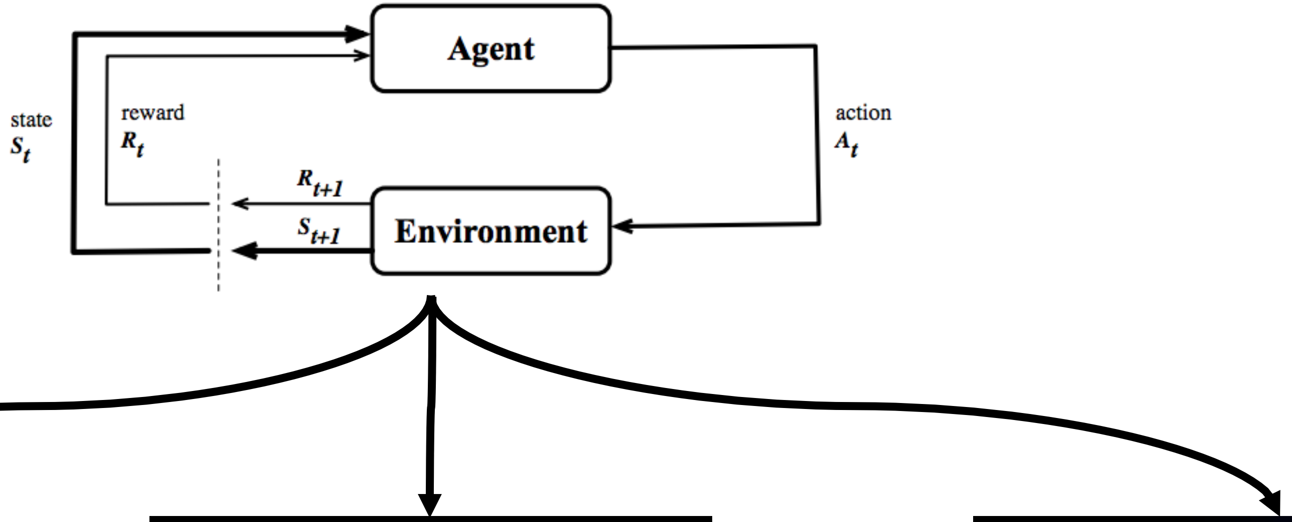
Impact of the rejection

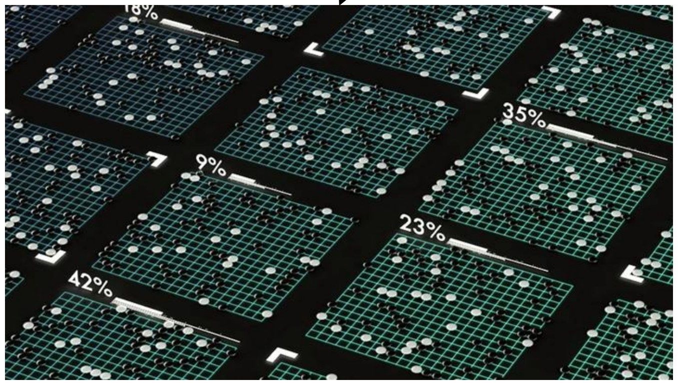
Number of rejected times

Whether the job is runnable

…

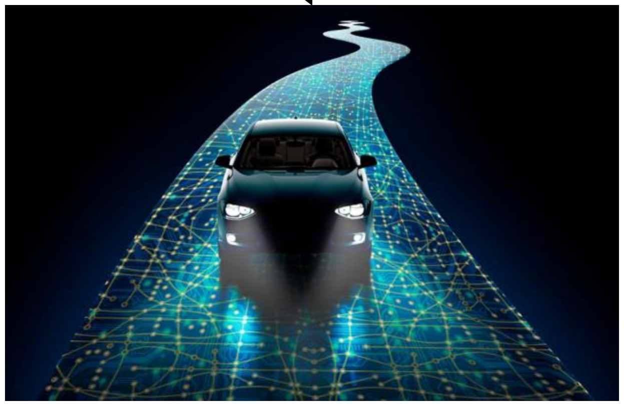
Reinforcement Learning





David Silver, et. al. Mastering the game of Go with deep neural networks and tree search, Nature vol. 529 (2016)

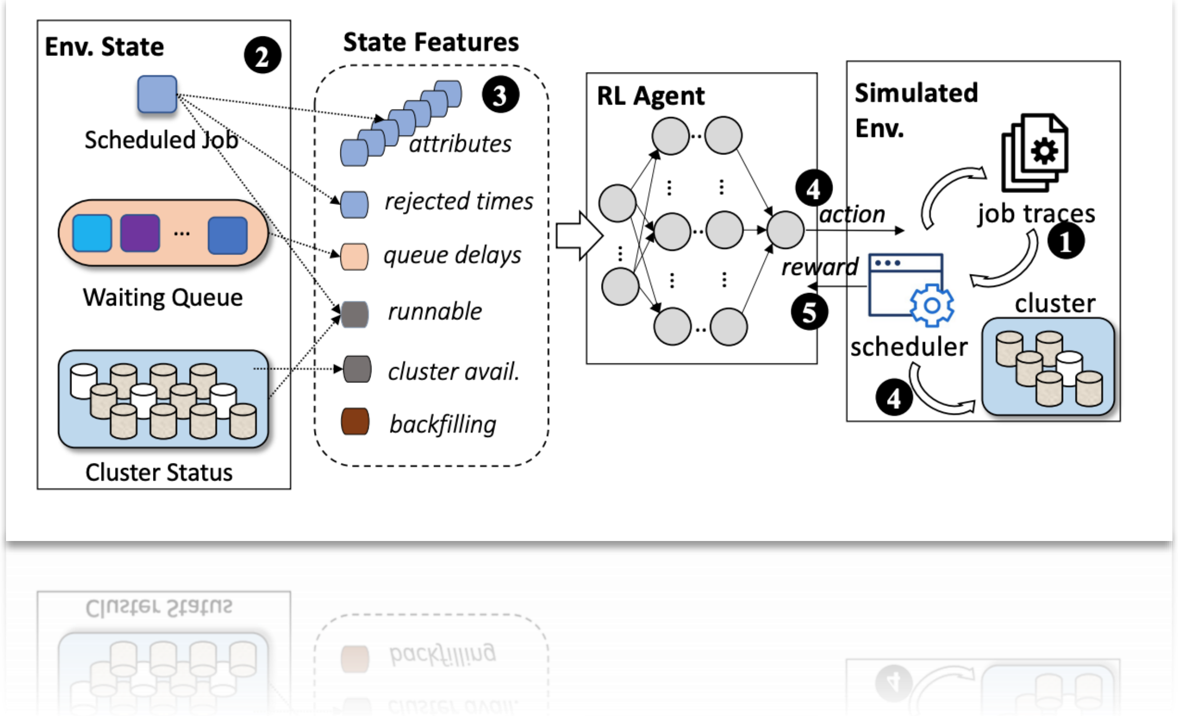
Volodymyr Mnih, et. al. Playing Atari with Deep Reinforcement Learning arXiv:1312.5602 (cs)



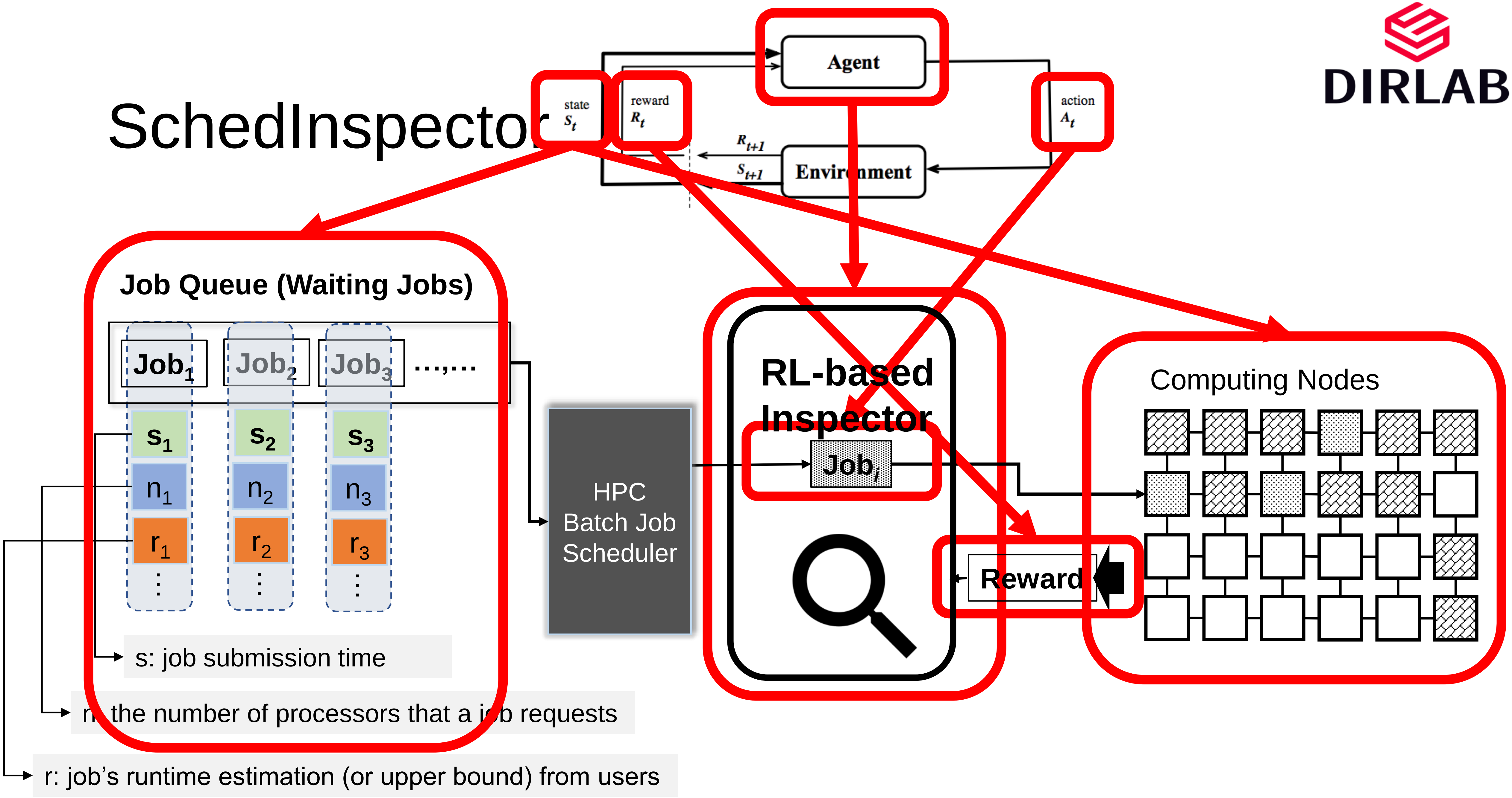
From https://www.selfdrivingcars360.com/how-autonomous-vehicles-fit-into-our-ai-enabled-future/

Our Contribution

- The first scheduling inspector for HPC systems.
- New optimizations of the state and reward to enable efficient RL training.
- Extensively evaluations on efficiency, stability and interpretability of SchedInspector.

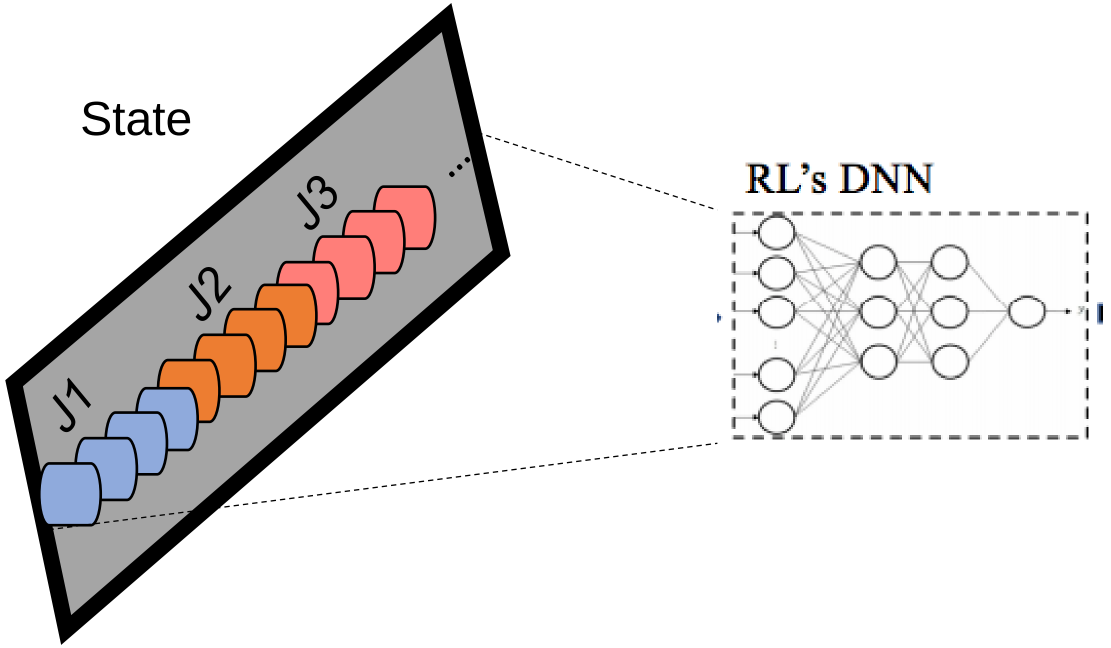


SchedInspector



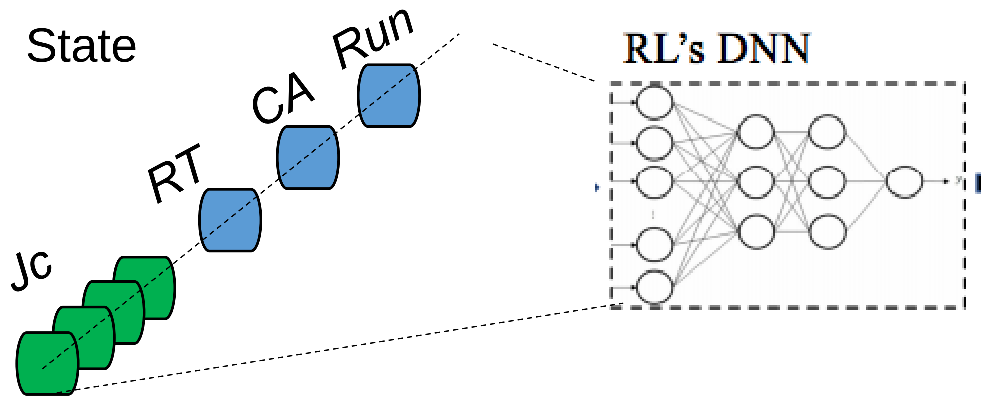
Design of State

Naïve Features



Compacted Features

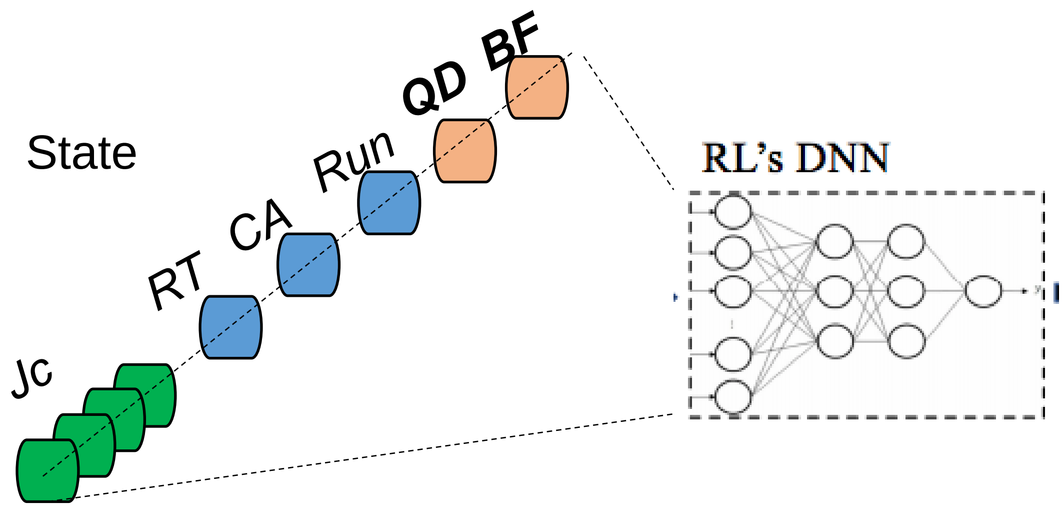
Jc: Scheduled Job
RT: Rejected Times
CA: Cluster Avail.
Run: Runnable

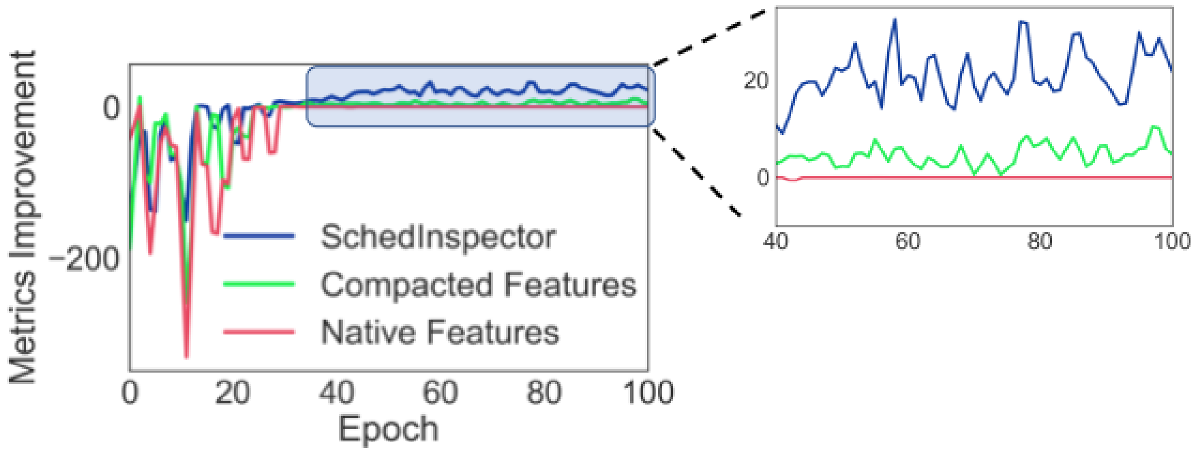


Design of State

SchedInspector

QD: Queue Delay
BF: Backfilling



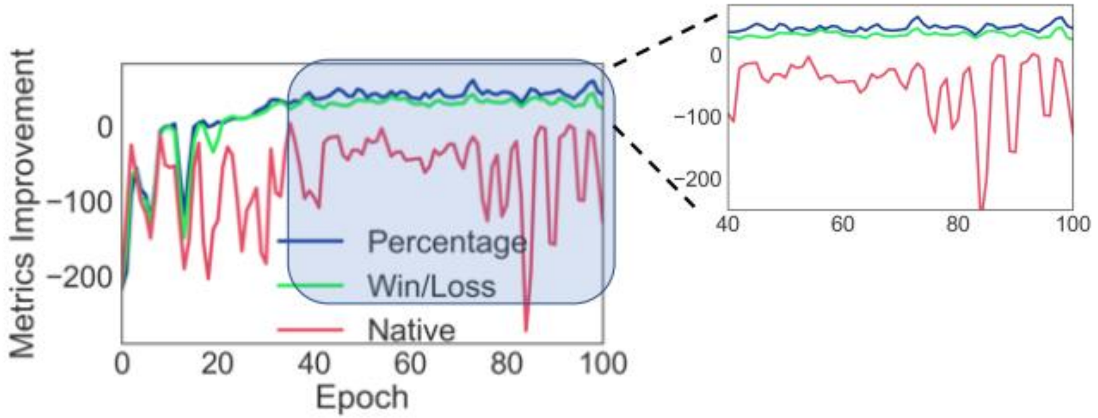


Design of Reward

Naïve: $Metric_{inspect}$ - $Metric_{orig}$

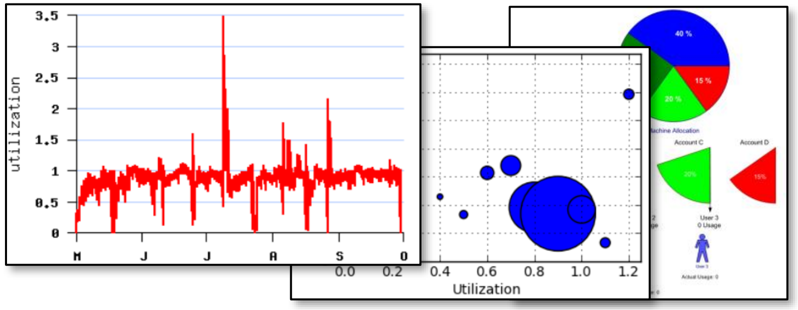
Win/Loss: $Integer(Metric_{inspect} > Metric_{orig})$

Percentage: $(Metric_{inspect} - Metric_{orig}) / Metric_{orig}$

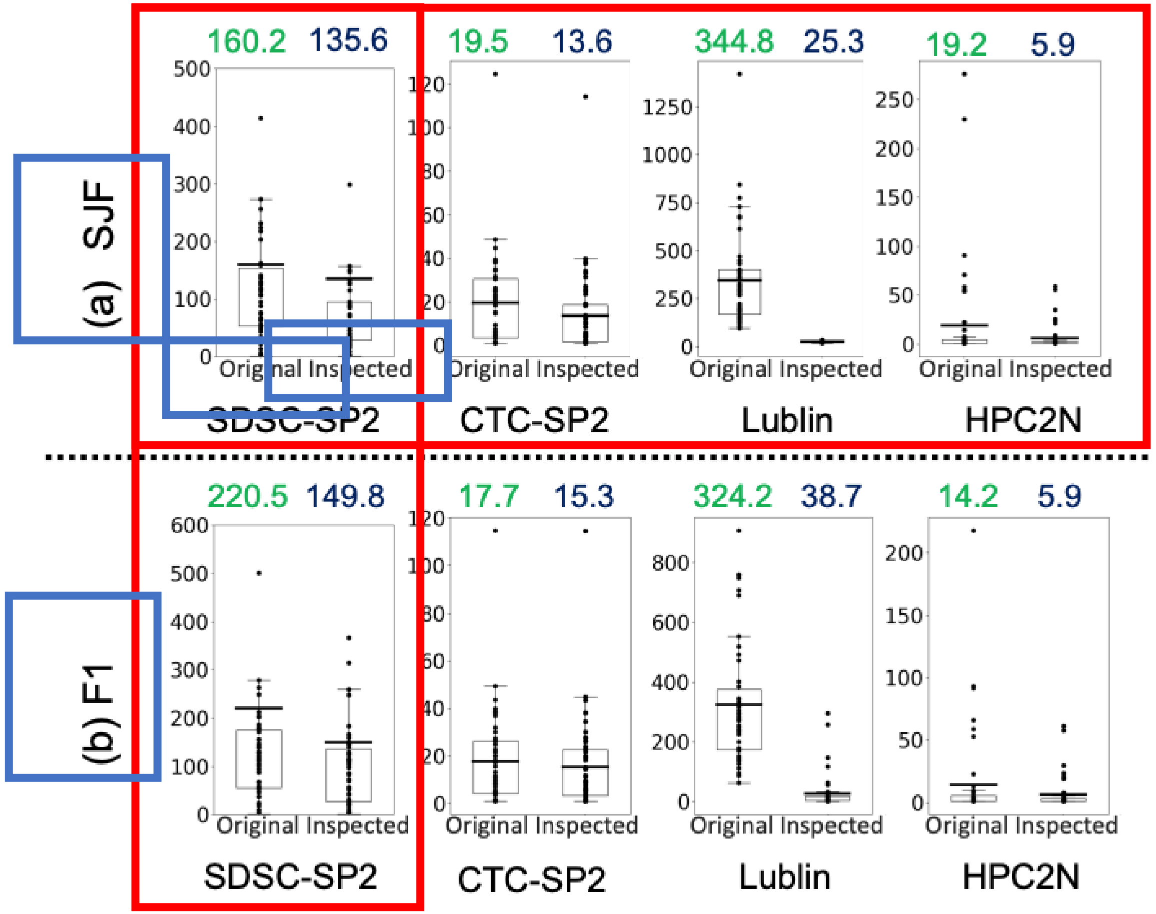


Evaluation Outline

- Usability
- Efficiency
- Interpretability



Testing for **Different Job Traces and Policies**



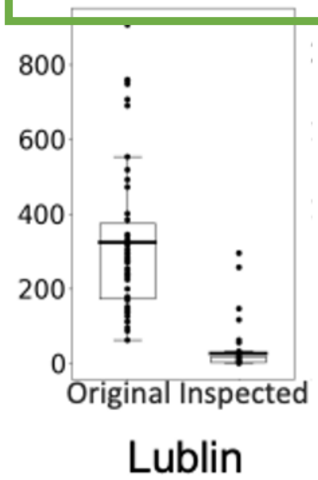
SchedInspector has significant improvement for the two scheduling policies on all job traces.

Impact on **System Utilization**

DIRLAB

	SJF			F1		
	BASE	INSP	Δ	BASE	INSP	Δ
*Scheduling **without** Backfilling*						
SDSC-SP2	59.64%	59.37%	**-0.27%**	60.18%	60.59%	**+0.41%**
CTC-SP2	51.35%	49.92%	**-1.43%**	54.40%	54.23%	**-0.17%**
Lublin	61.49%	61.06%	**-0.43%**	67.37%	63.04%	**-4.33%**
HPC2N	23.72%	23.47%	**-0.25%**	24.00%	23.79%	**-0.21%**
*Scheduling **with** Backfilling*						
SDSC-SP2	78.45%	78.37%	**-0.08%**	76.71%	76.93%	**+0.22%**
CTC-SP2	74.98%	74.89%	**-0.09%**	75.47%	76.05%	**+0.58%**
Lublin	79.38%	77.71%	**-1.67%**	80.38%	78.08%	**-2.30%**
HPC2N	56.81%	57.10%	**+0.29%**	57.11%	56.57%	**-0.54%**

324.2 38.7



88% Improvement

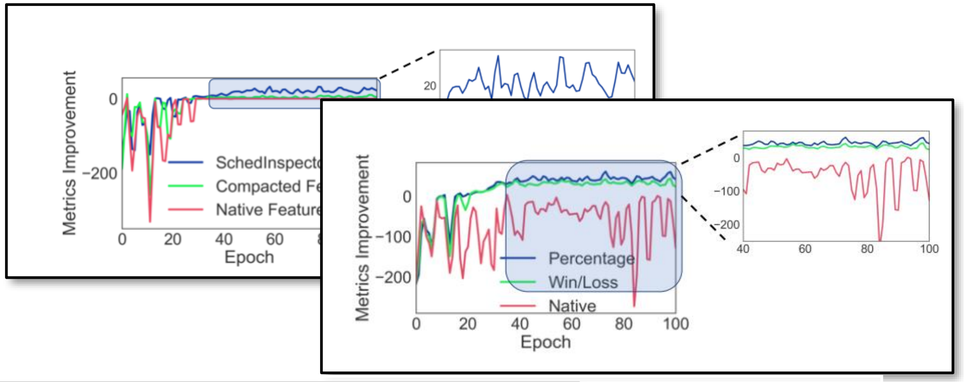
SchedInspector has barely noticeable reduction (1% difference) on system utilization

Evaluation Outline

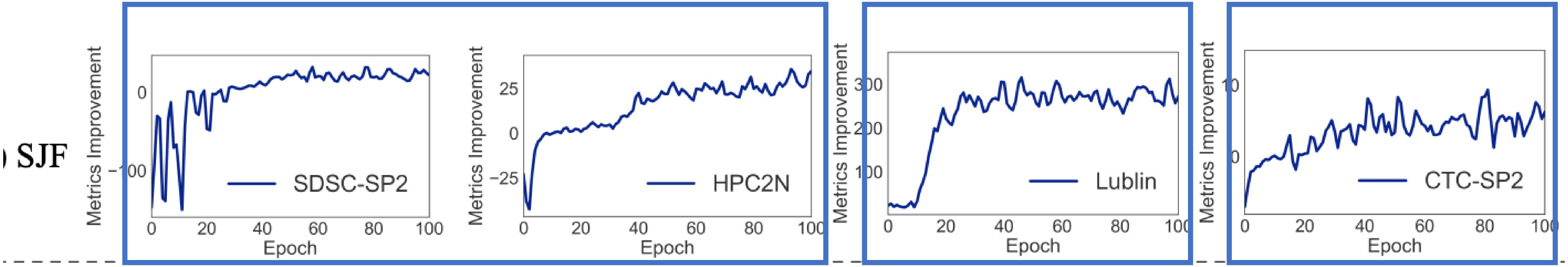
Usability

Efficiency

Interpretability

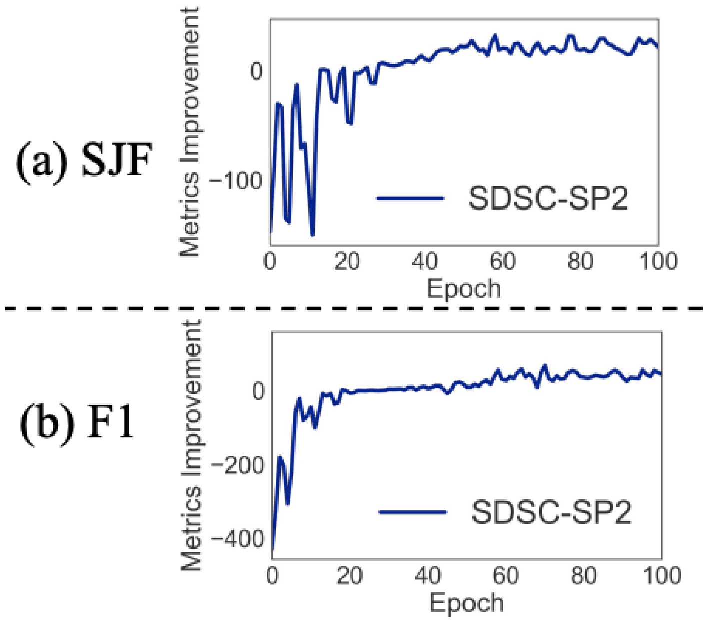


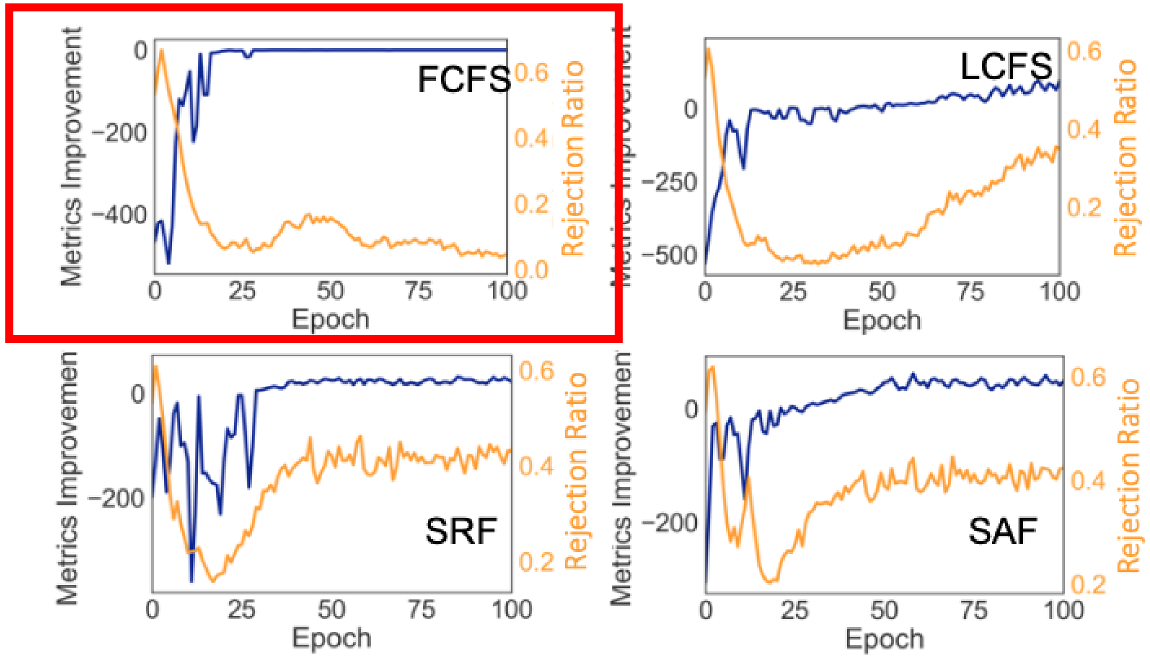
Training on **Different Job Traces**



SchedInspector converges in all of the workloads within 100 training epochs and different job traces have different converge pattern.

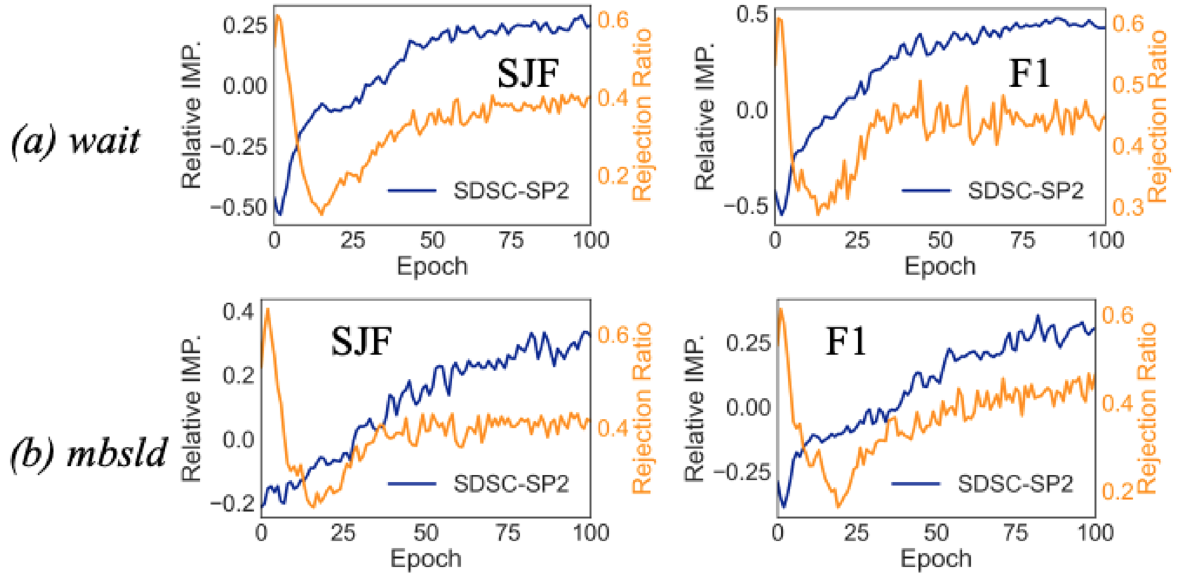
Training on **Different Scheduling Policies**





SchedInspector converges in all scheduling policies.
For some scheduling policies, the converged value is near 0 and the rejection ratio is low.

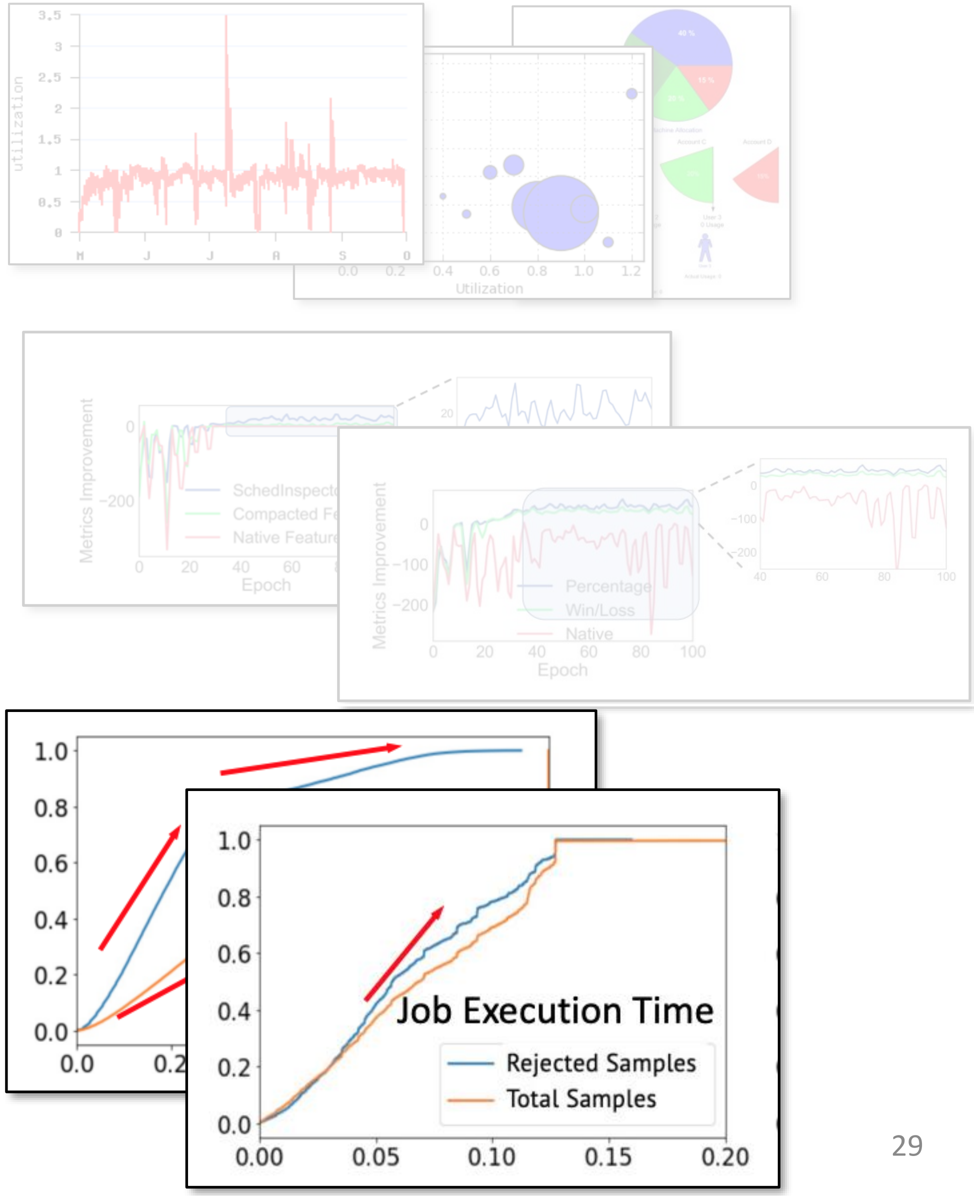
Training for **Different Metrics**



SchedInspector converges towards two new metrics but with different patterns.

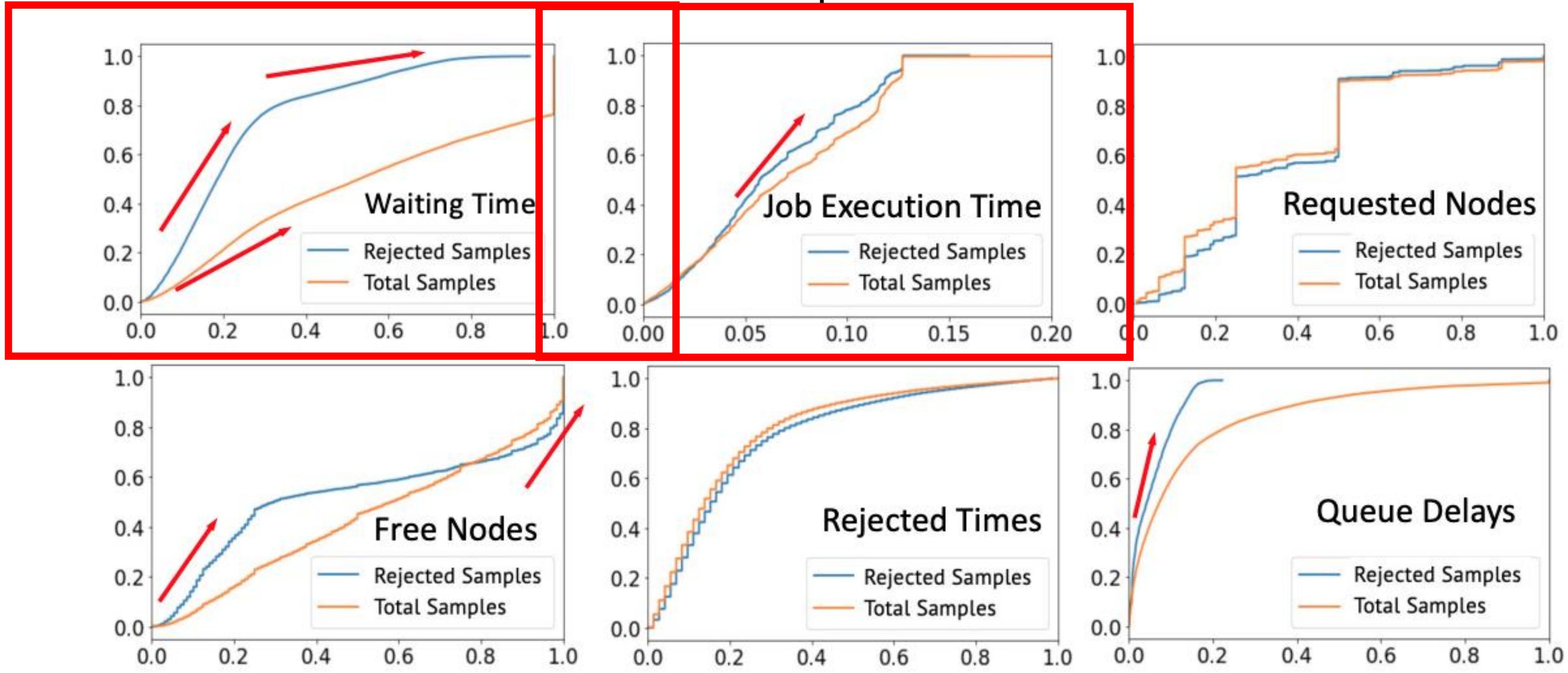
Evaluation Outline

- Usability
- Efficiency
- Interpretability



What SchedInspector Learns

CDF of input features.



SchedInspector has obvious patterns for different features which indicates the effectiveness of feature selection

Hare: Exploiting Inter-job and Intra-job Parallelism of Distributed Machine Learning on Heterogeneous GPUs

Fahao Chen[1], Peng Li[1], Celimuge Wu[2], Song Guo[3]

[1]The University of Aizu

[2]University of Electro-Communications

[3]The Hong Kong Polytechnic University & The Hong Kong Polytechnic University Shenzhen Research Institute

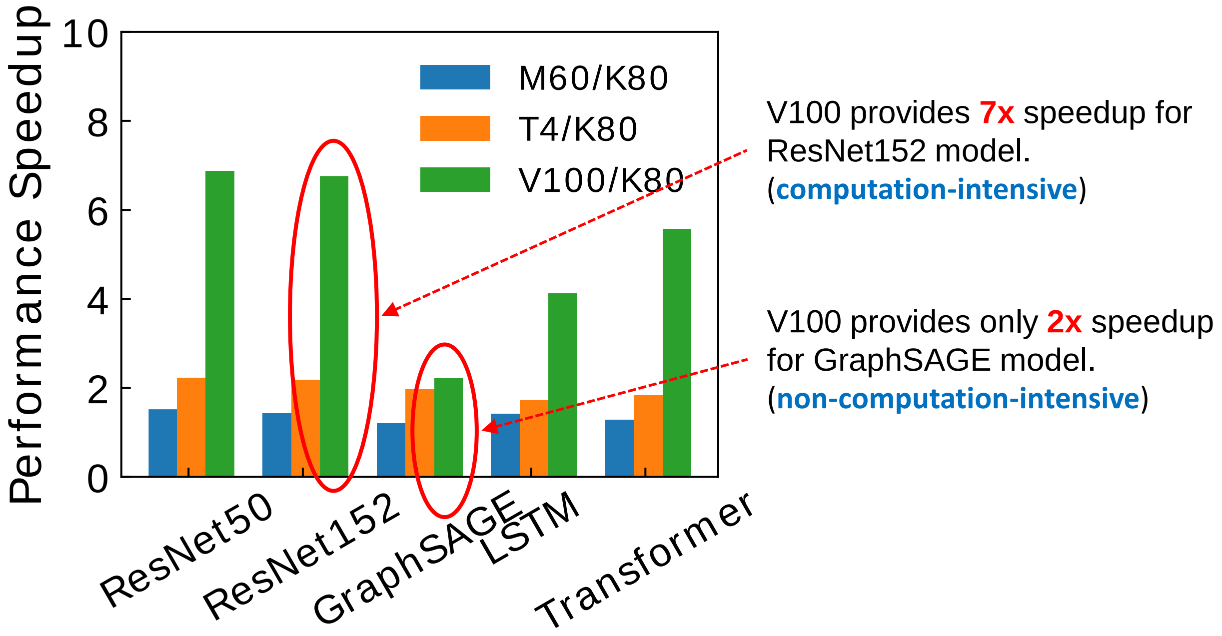
ACM HPDC 2022
Minneapolis, Minnesota, United States

THE UNIVERSITY OF AIZU

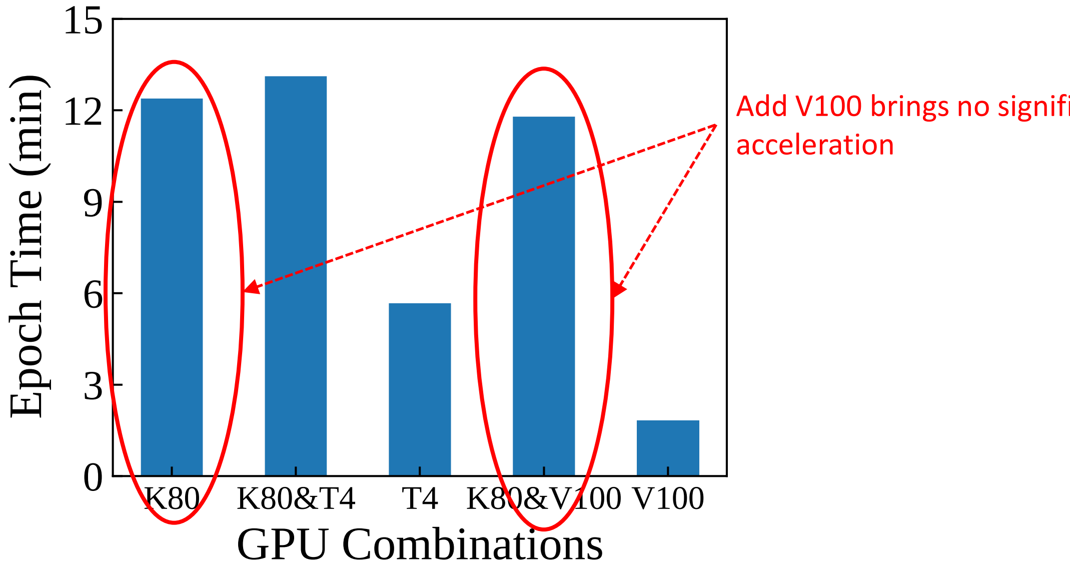
The University of Electro-Communications UEC

THE HONG KONG POLYTECHNIC UNIVERSITY 香港理工大學

Why Do We Need to Consider GPU Heterogeneity?



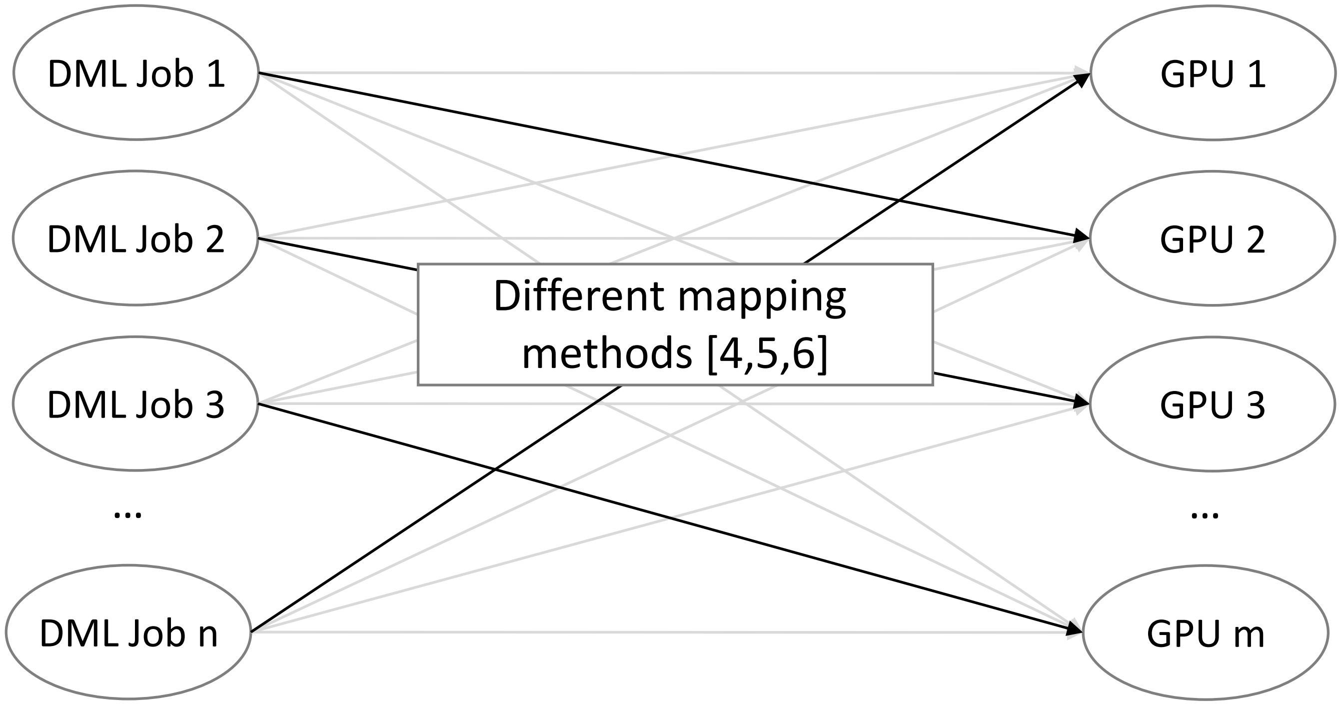
(a) Training speedup of different jobs on different GPUs.



(b) Round time of ResNet152 under different GPU combinations.

- Different GPUs provide different performance speedups for different jobs. (**inter-job parallelism**)
- Different GPU combinations provide different performance speedups. (**intra-job parallelism**)

Existing Works on Heterogeneous GPUs

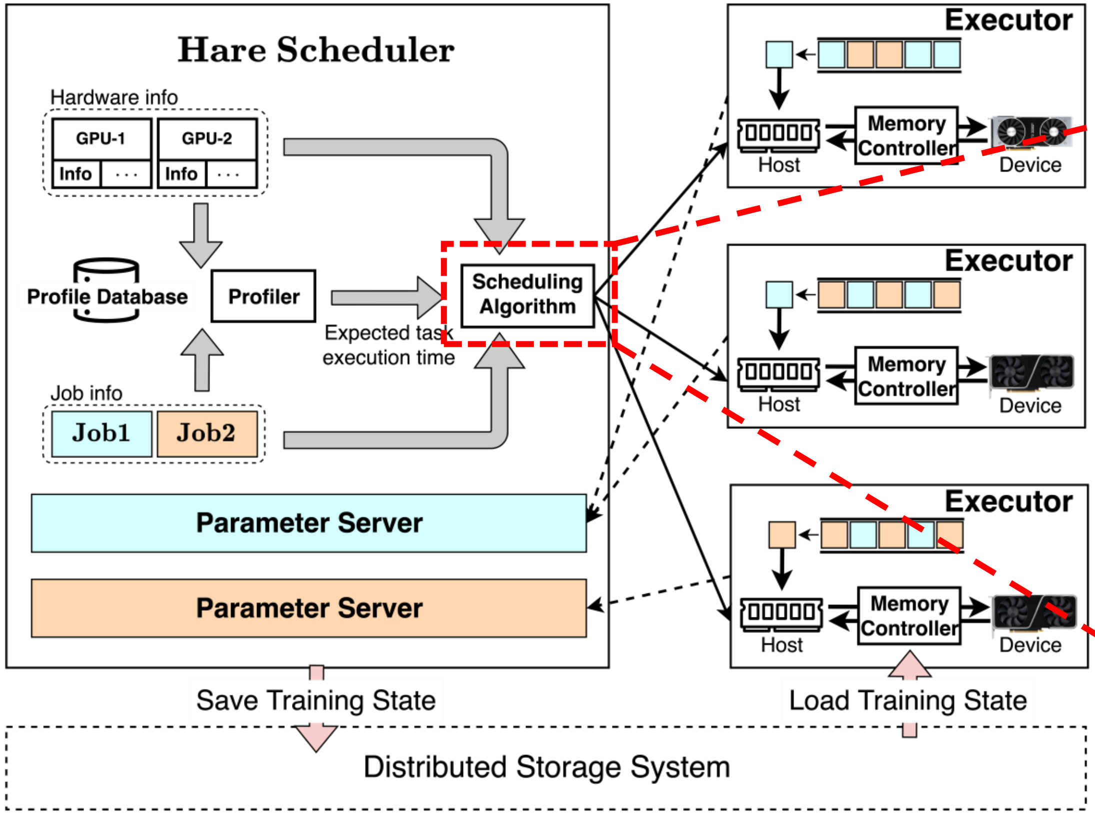


They treat DML jobs as unsplittable units and ignore the intra-job parallelism.

[6] Chaudhary, Shubham, et al. "Balancing efficiency and fairness in heterogeneous GPU clusters for deep learning." Proceedings of the Thirteenth EuroSys Conference. 2020.
[7] Le, Tan N., et al. "Allox: compute allocation in hybrid clusters." Proceedings of the Thirteenth EuroSys Conference. 2020.
[8] Narayanan, Deepak, et al. "Heterogeneity-Aware Cluster Scheduling Policies for Deep Learning Workloads." 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20).

Hare: An Efficient DML Job Training System

System Goals: High training efficiency, High GPU utilization, and ***Starvation-free***



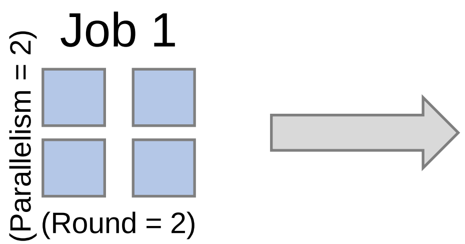
Overview of Hare

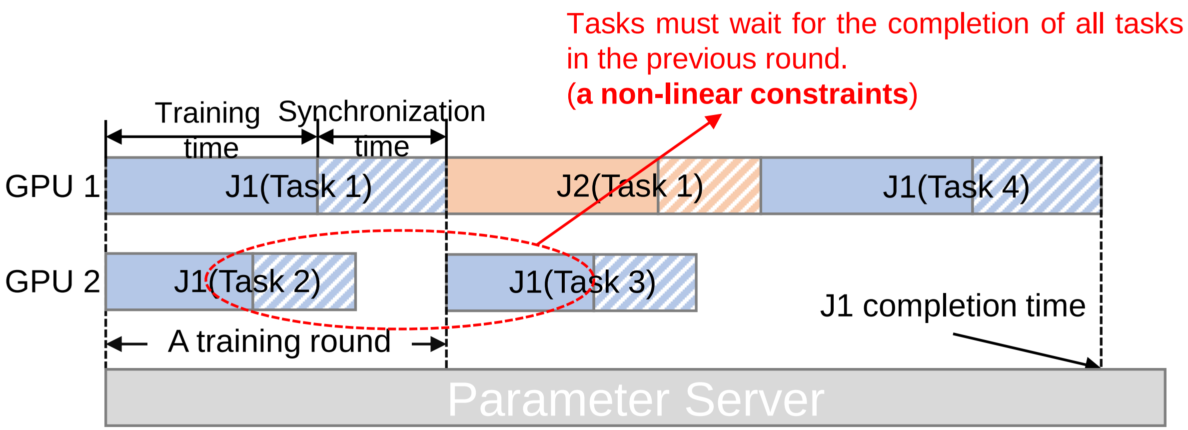
Scheduling Algorithm	Intra-job Parallelism	Inter-job Parallelism	Theoretical bound
Gandiva Tiresias Optimus Themis Antman	✓	✗	✗
Allox	✗	✓	✗
Hare	✓	✓	✓

Compared with other scheduling algorithms

Scheduling Algorithm: Problem Statement

1. Model:





2. Task scheduling problem

- **Objective:** minimize the total weighted job completion time
- **Solution:** the start running time of each task

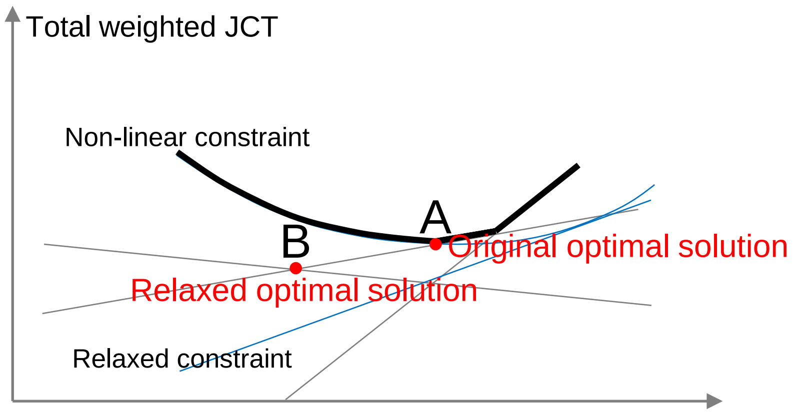
Theorem: The above problem is NP-hard, which cannot be solved within a polynomial time.

Proof: Reduce the well-known SS13 problem [11].

[11] Garey, Michael R., and David S. Johnson. Computers and intractability. Vol. 174. San Francisco: freeman, 1979.

Scheduling Algorithm: Algorithm Design

Step 1: Problem relaxation [12]



Step 2: Task scheduling

- Decide the scheduling ordering of tasks according to the solution of relaxed problem
- Greedily assign tasks to GPUs with the earliest available time

Theorem: Our scheduling algorithm is $\alpha(2+\alpha)$-approximation.

Proof: Please refer to our paper.

$\alpha = \max_{task}\{\frac{T^{c,max}}{T^{c,min}}, \frac{T^{s,max}}{T^{s,min}}\}$
T^c: task training time
T^s: task synchronization time

[12] Queyranne, Maurice. "Structure of a simple scheduling polyhedron." Mathematical Programming 58.1 (1993): 263-285.

Evaluation: Experimental settings

Testbed: 15 heterogeneous GPUs (V100 + T4 + K80 + M60)

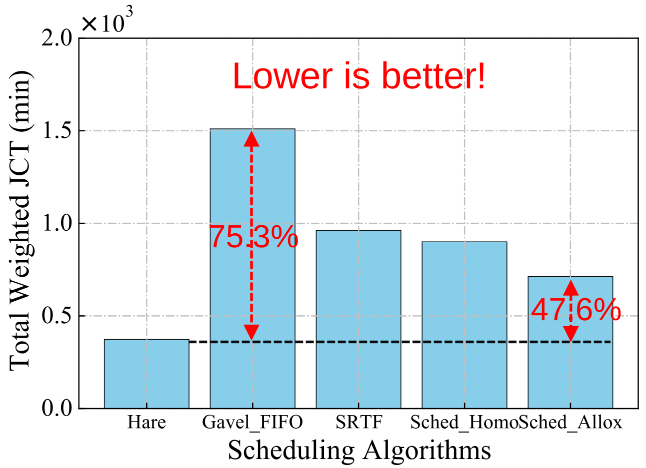
Simulator: up to 200 GPUs and 300 DML jobs

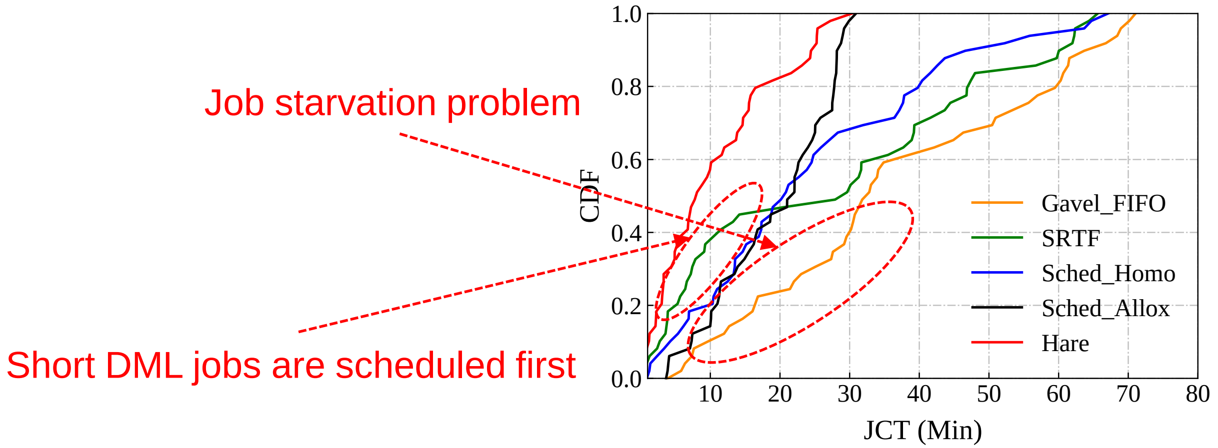
Workload: 8 popular models across domains of CV, NLP, Speech, and Recognition

Baseline: 4 popular scheduling schemes.

(*Please refer to our paper for more details on experimental settings.)

Evaluation: Testbed

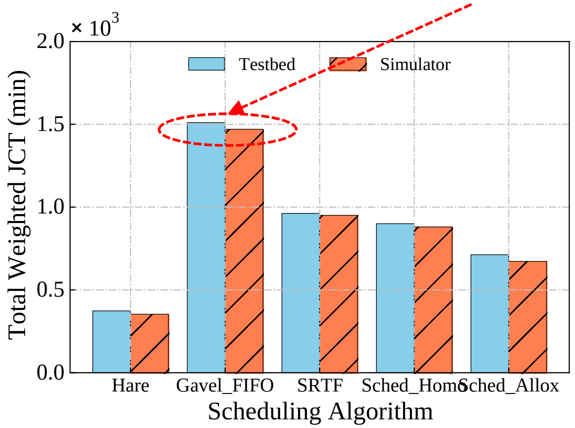


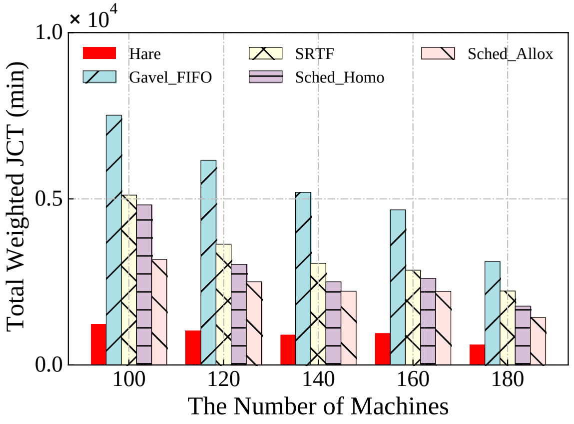


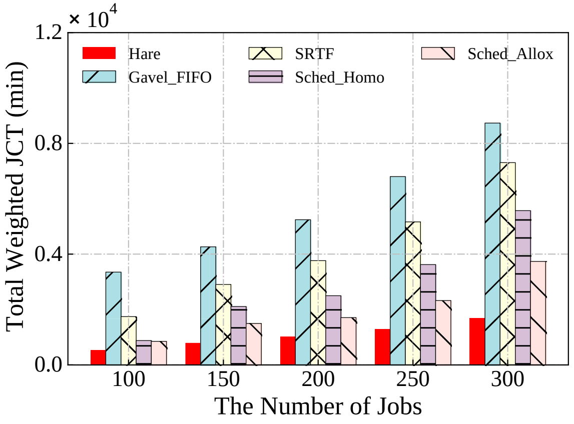
- Our scheduling algorithm reduces the total weighted JCT by 47.6% to 75.3% by others.
- About 90.5% of jobs can complete within 25 minutes based on our scheduling algorithm.

Evaluation: Simulation

Error in prediction of training time

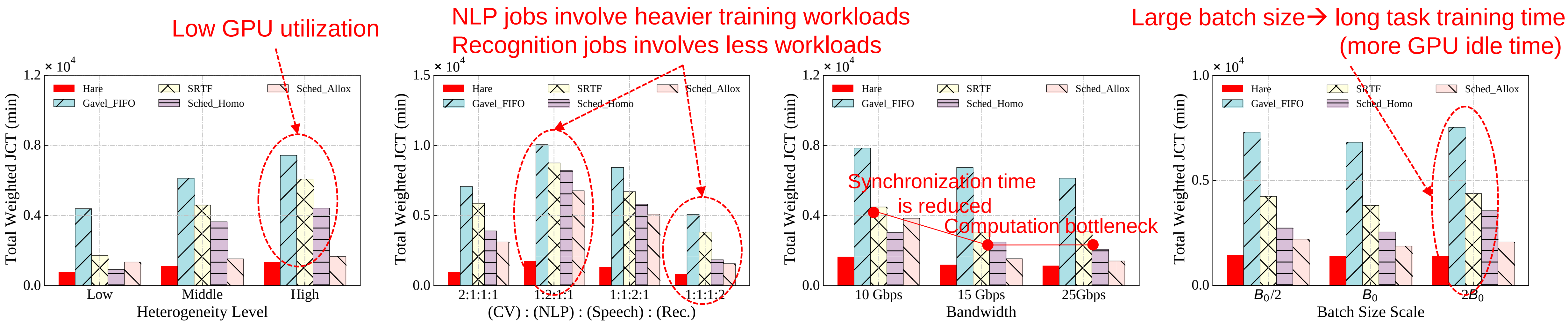






- The maximum performance gap between the testbed and simulator is only 5%.
- Our scheduling algorithm outperforms others under a scaling of GPUs and jobs.

Evaluation: Simulation



- Our scheduling algorithm outperforms others under various settings.

Conclusion

- **Hare:** An DML jobs training system with the objectives of ***high training efficiency***, ***high GPU utilization***, and ***starvation-free***

- **Task Scheduling Algorithm:**
  - Minimize total weighted JCT
  - Reduce total weighted JCT by **47.6% to 75.3%** over other schemes

- **Constraint:**
  - Jobs with changed setting may incur high overhead when scheduling
  - Hare is short in handling dynamic jobs

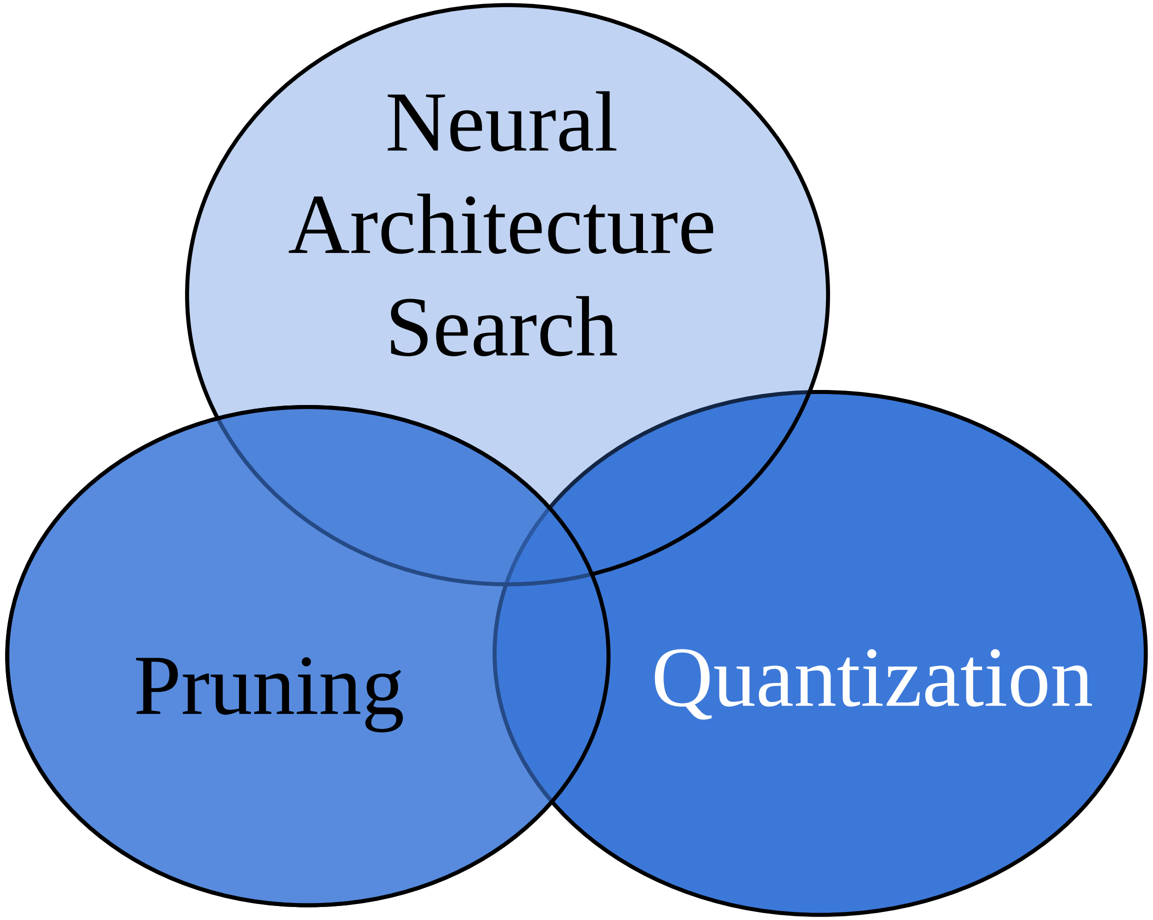
Efficient Design Space Exploration for Sparse Mixed Precision Neural Architectures

Krishna Teja Chitty-Venkata[1], Arun Somani[1]
Murali Emani[2], Venkatram Vishwanath[2]

[1]Iowa State University, Ames, IA, USA
[2]Argonne National Laboratory, Lemont, IL, USA

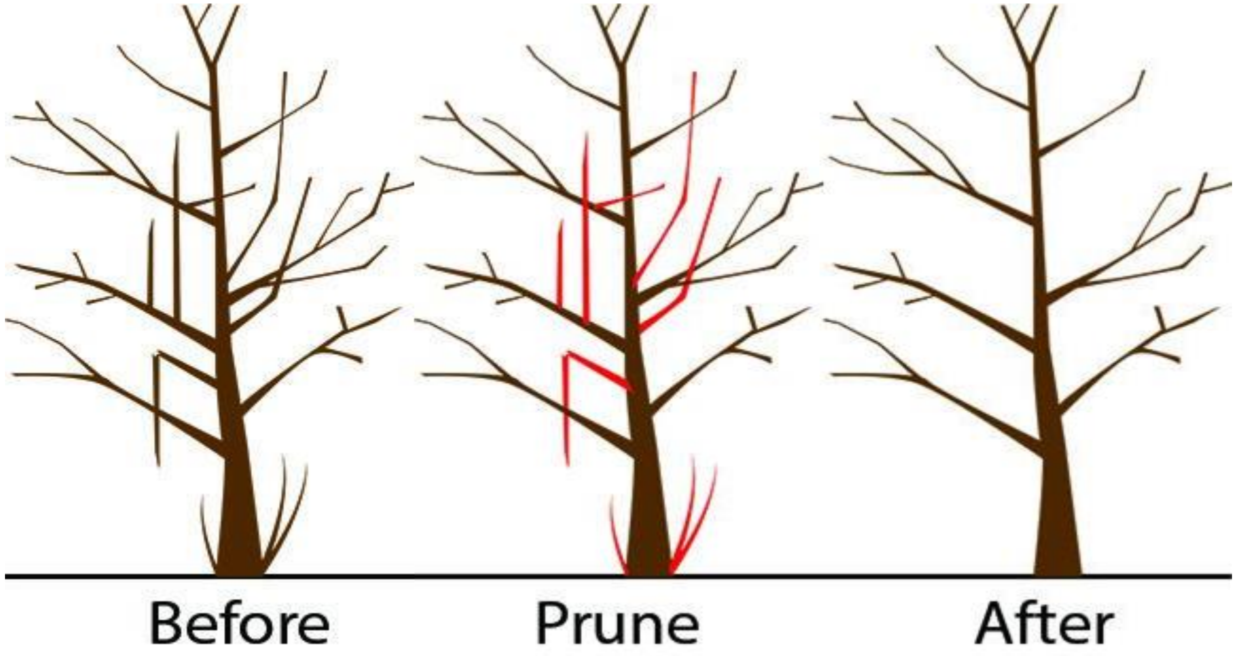
ACM HPDC 2022
The 31st International Symposium on High-Performance Parallel and Distributed Computing

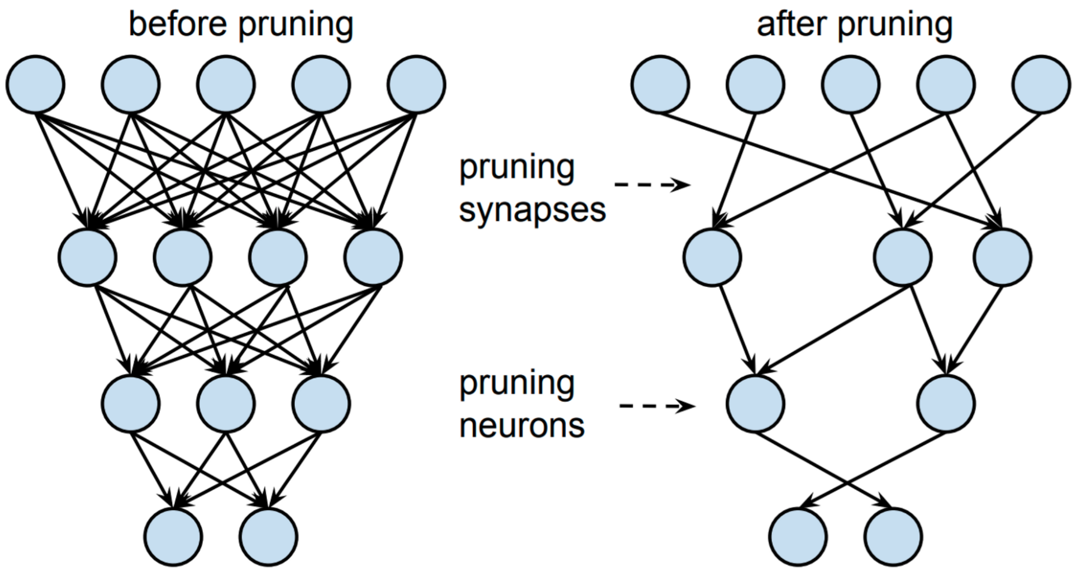
Deep Neural Network Optimization



Pruning

- Pruning in general refers to cutting down branches/leaves
- Neural Network Pruning refers to removing weights/connection without compromising the accuracy
- The parameters which do not contribute to the final accuracy are removed to save memory
- Example: Magnitude-based

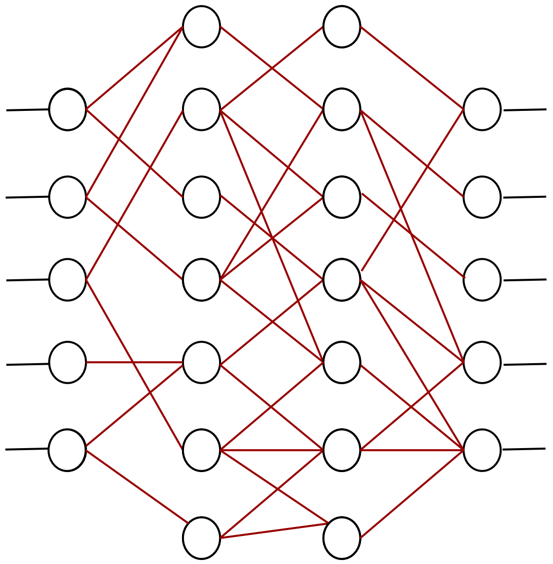




Song Han et.al. Deep Compression

Neural Network Pruning

Weight/Irregular Pruning

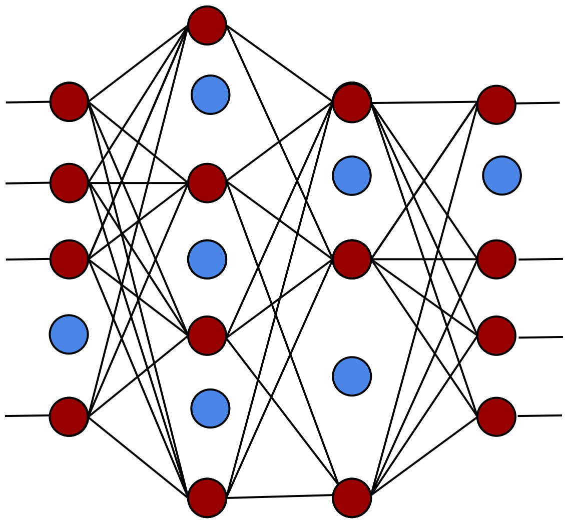


Weight Matrix

2	0	0	1	0	6
0	0	2	0	0	0
0	0	0	6	0	0
1	0	0	0	7	0
0	0	3	0	0	0
7	0	0	8	0	9

- Creates Sparse Matrices
- Requires sparse decoding
- Irregular memory accesses if not stored contigously

Node/Structured Pruning



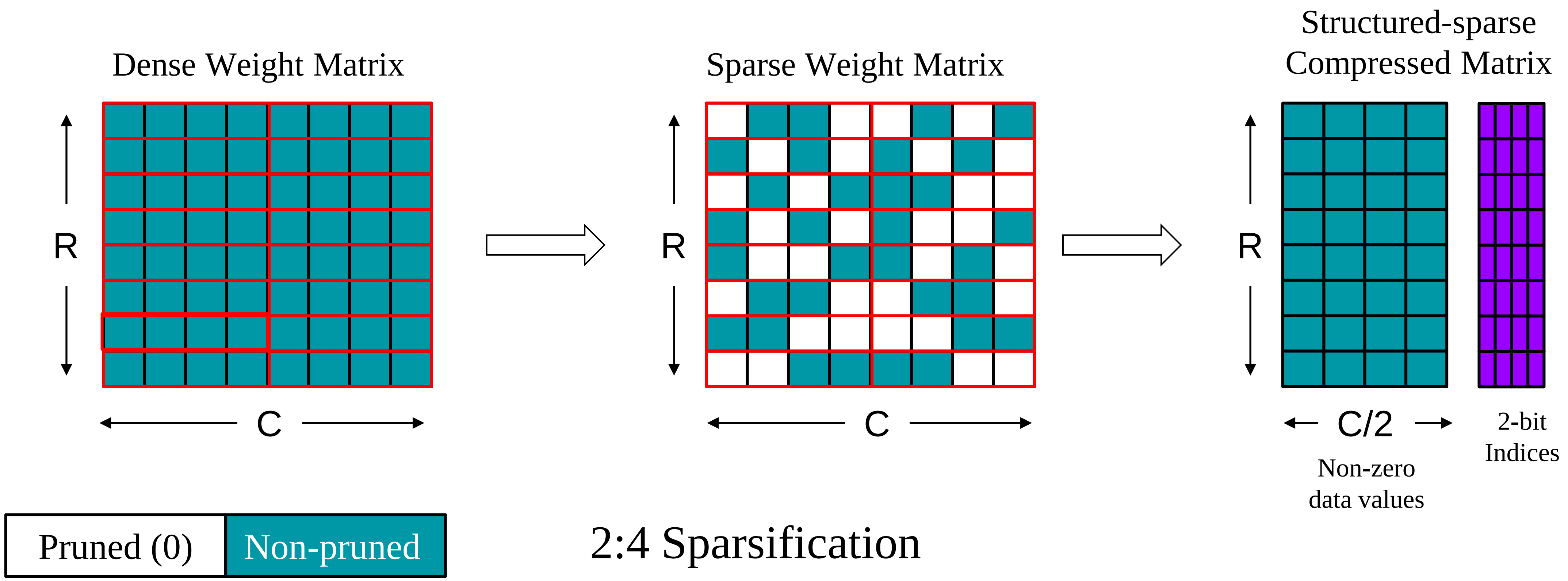
Weight Matrix

0	1	9	0	2	0
0	8	2	0	9	0
0	1	5	0	1	0
0	3	8	0	7	0
0	5	3	0	7	0
0	2	7	0	1	0

- Regular Dense Matrices
- Regular DNN implementation
- Regular memory accesses as easily stored in contiguous memory

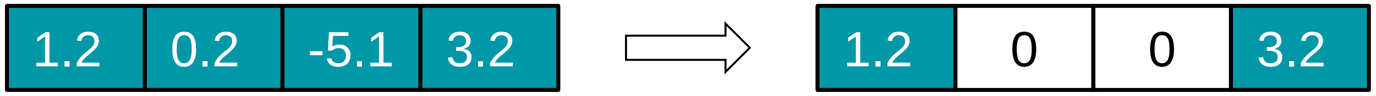
2:4 (two-to-four) Sparsity Pattern by Nvidia A100 GPU Tensor Cores:

- The 2:4 pattern mandates that for each group of 4 values, at least 2 must be zero
- The pattern leads to 50% sparsity while maintaining accuracy



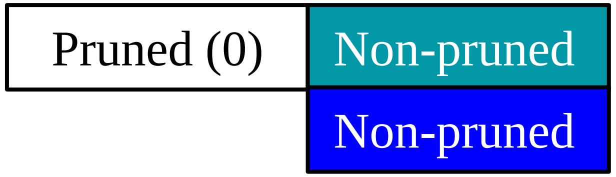
2:4 Sparsification

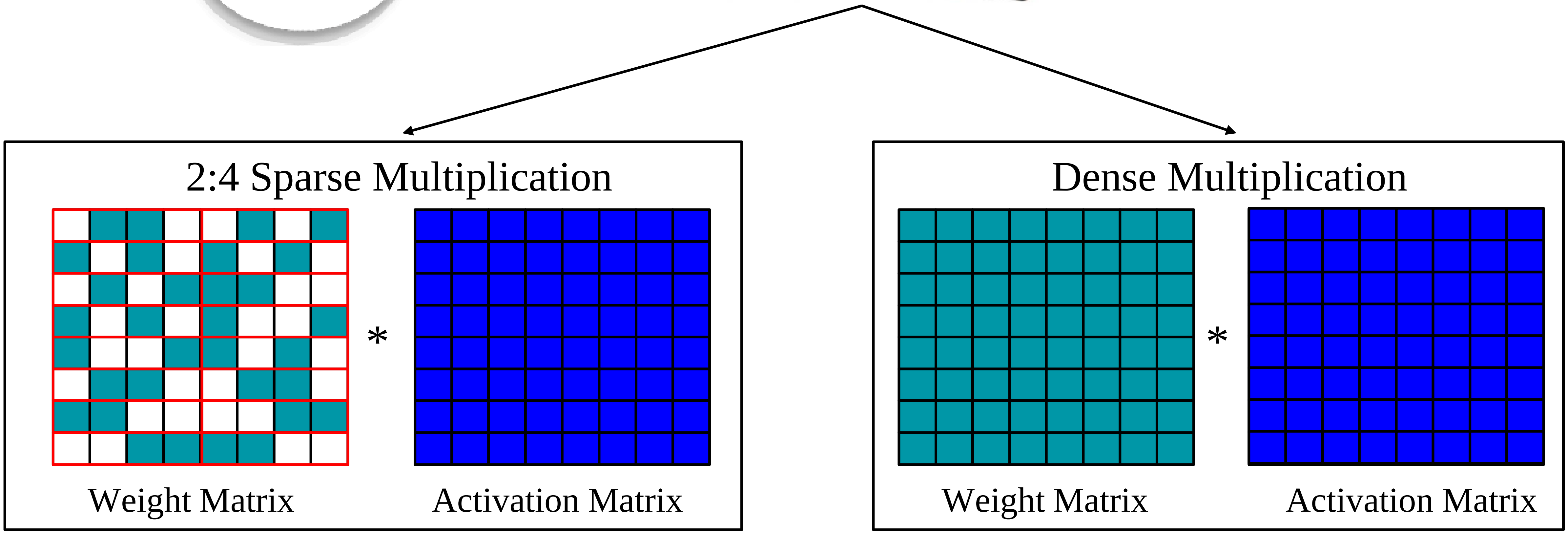
Example Pruning Strategy:



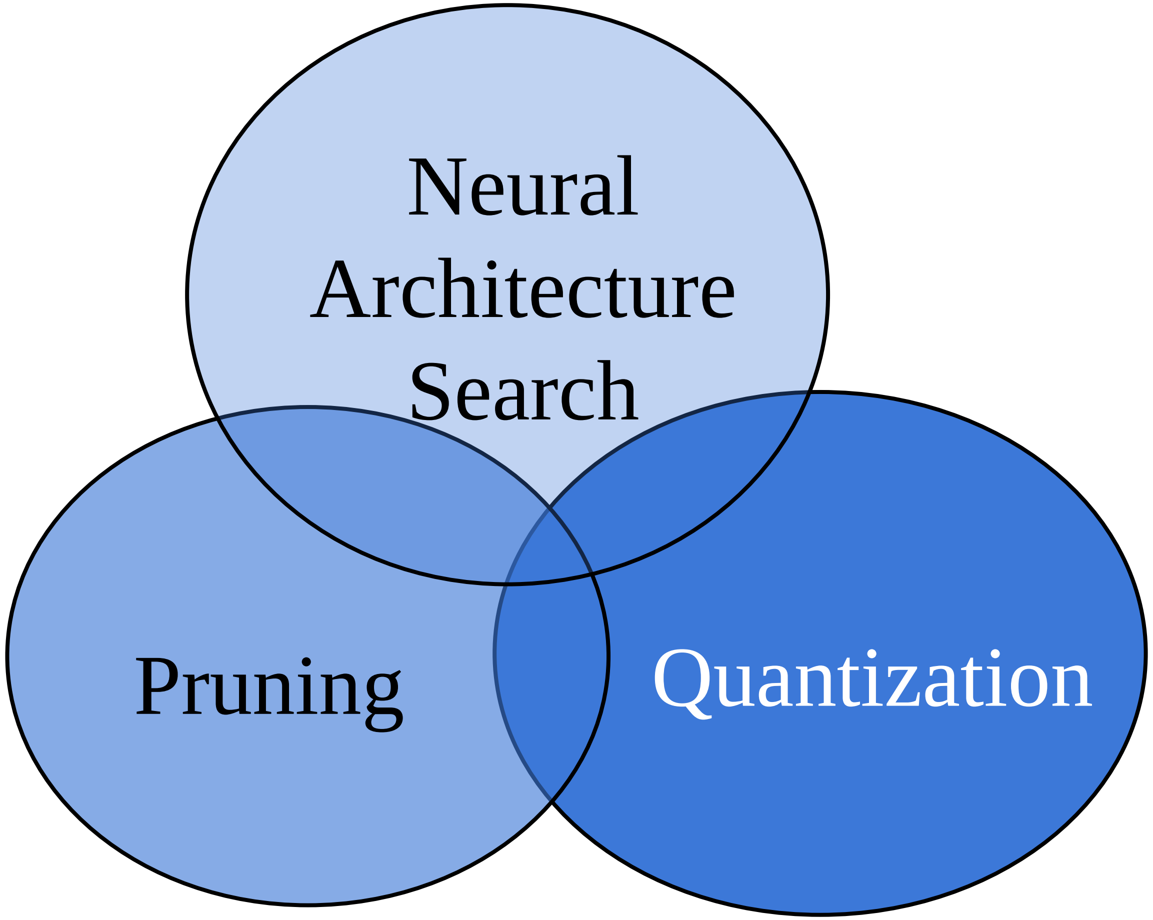
Types of Multiplications Supported by Nvidia A100 GPU

Theoretical
Speedup

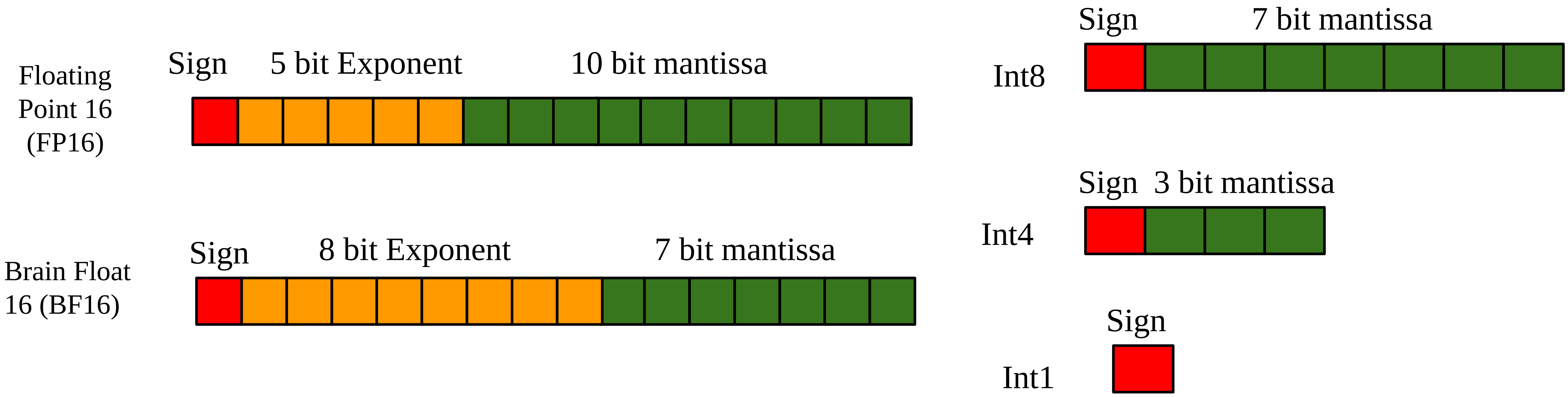




Deep Neural Network Optimization



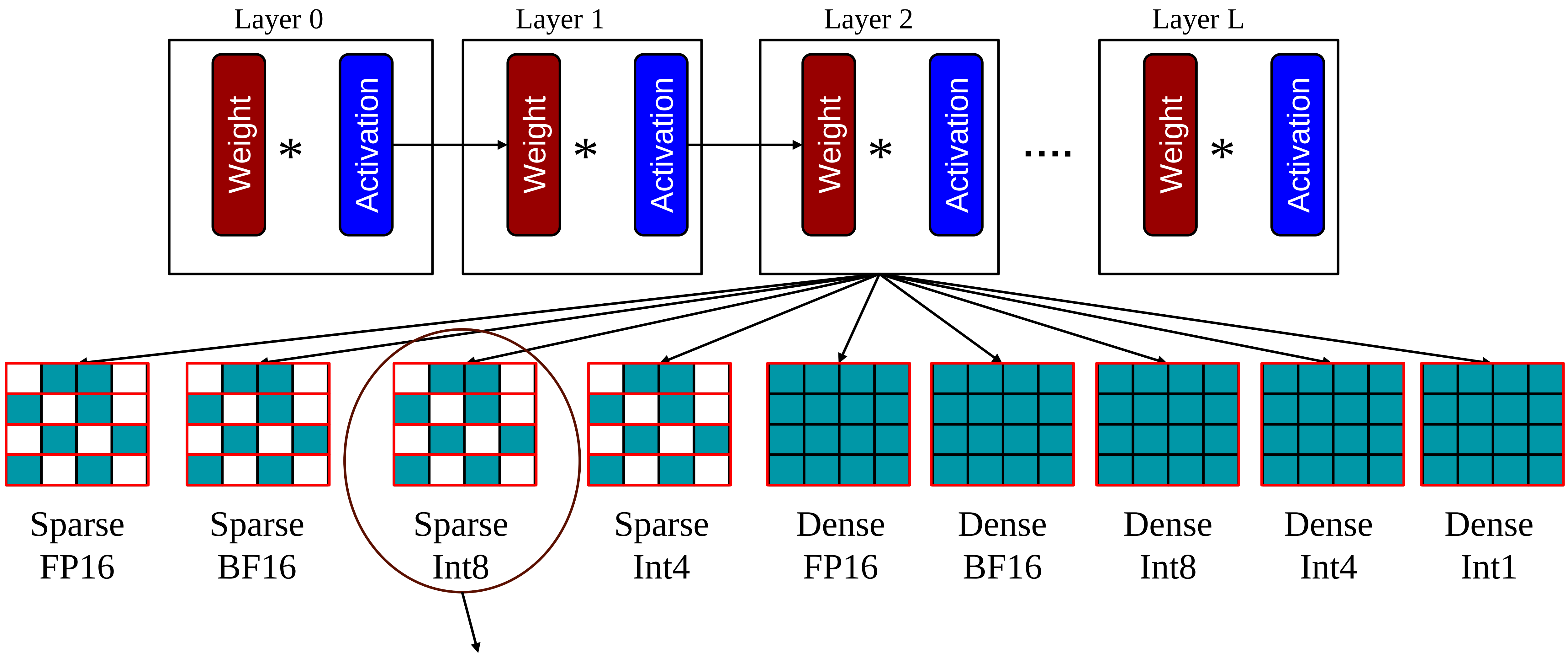
Types of Precisions Supported by Nvidia A100 Tensor Cores



Summary of Sparse-Precisions Supported by Nvidia A100 GPU

Multiplication	FP16	BF16	Int8	Int4	Int1
2:4 Sparse	Yes	Yes	Yes	Yes	No
Dense	Yes	Yes	Yes	Yes	Yes

Summary of Sparse-Precisions Supported by Nvidia A100 GPU Tensor Cores



Nvidia State-of-the-art

- 4x Latency
- ~ Accuracy

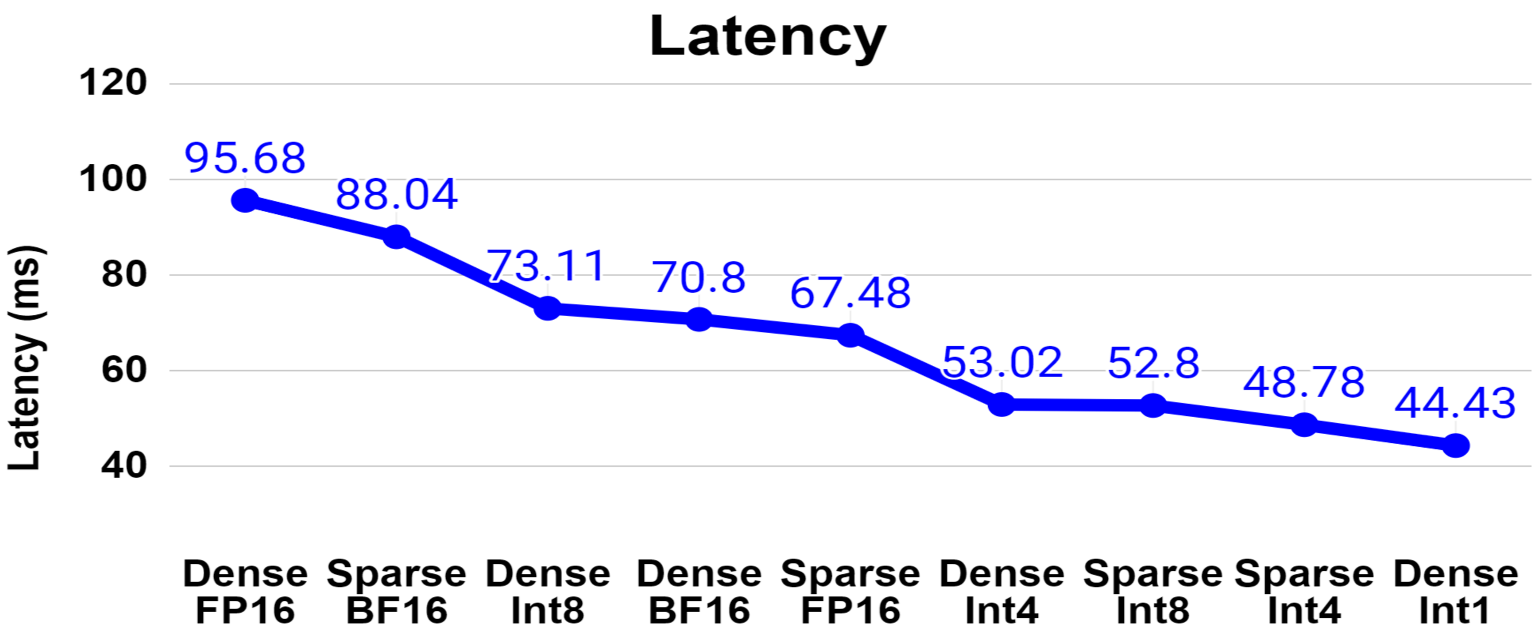
Mishra et al. "Accelerating Sparse Deep Neural Networks"

Latency vs Accuracy

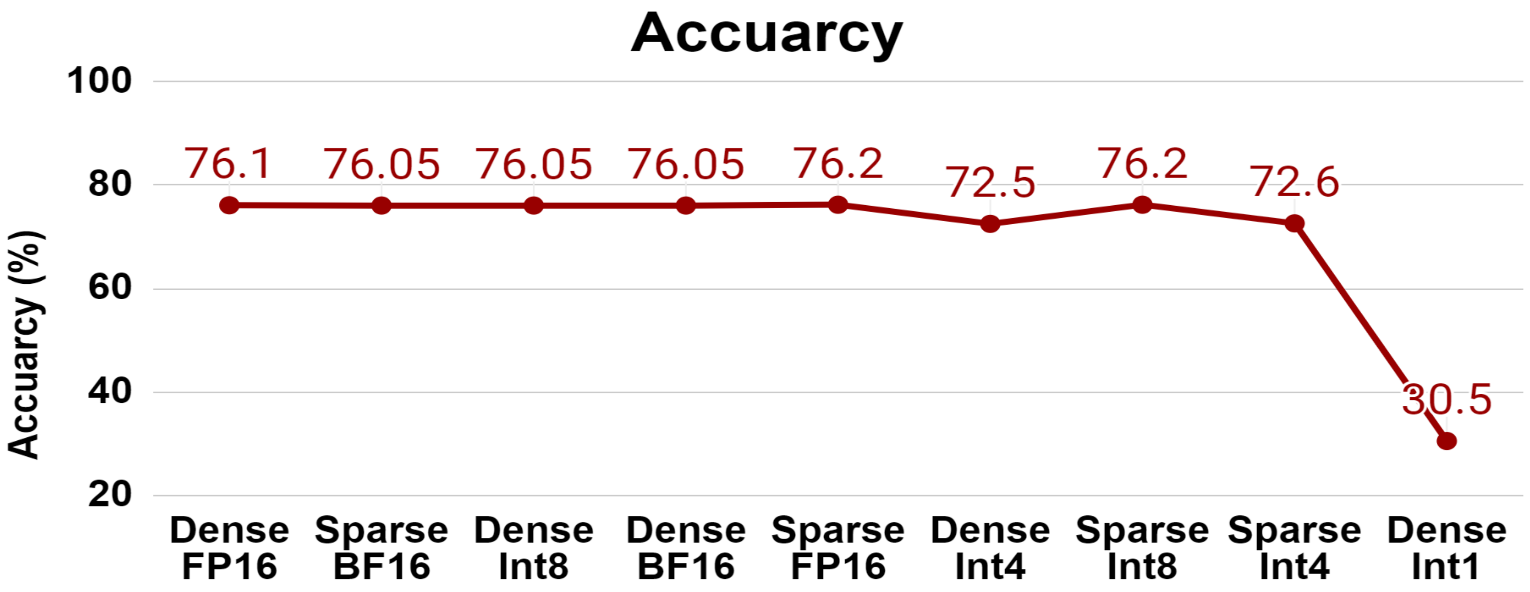
- Example:

ResNet50 on **Imagenet Dataset**

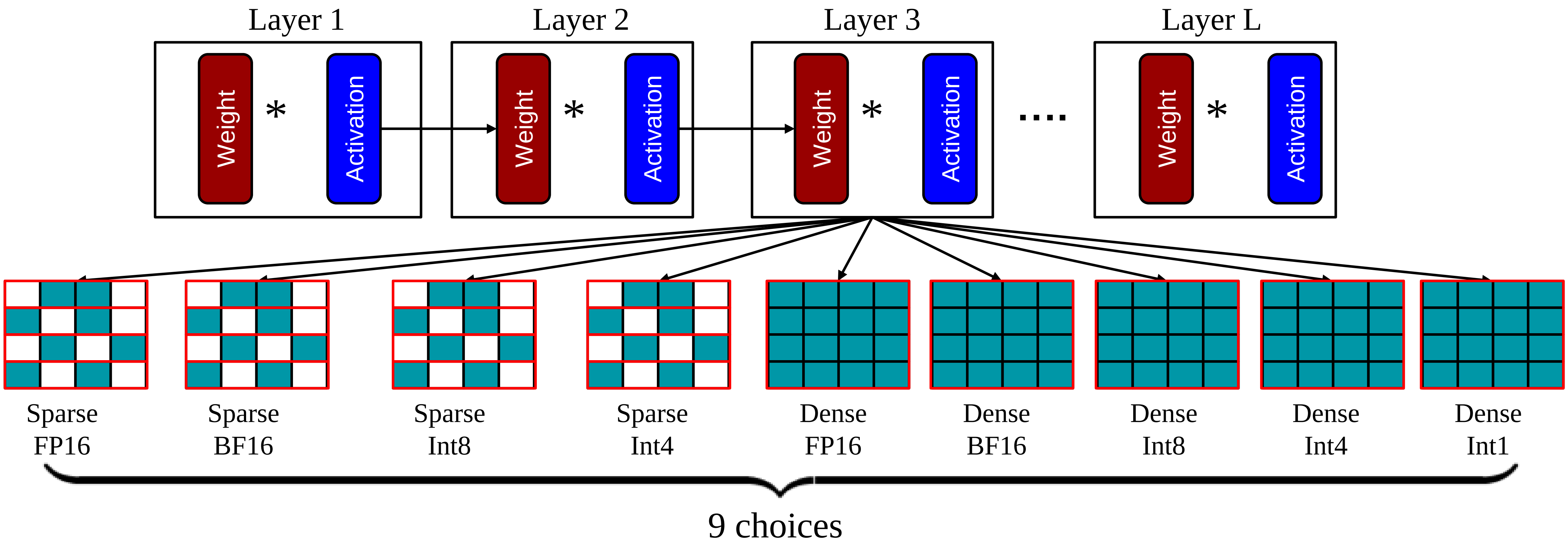
- Latency: Time taken for all the matrix multiplications in the Network



- Accuracy: Percentage of images correctly classified

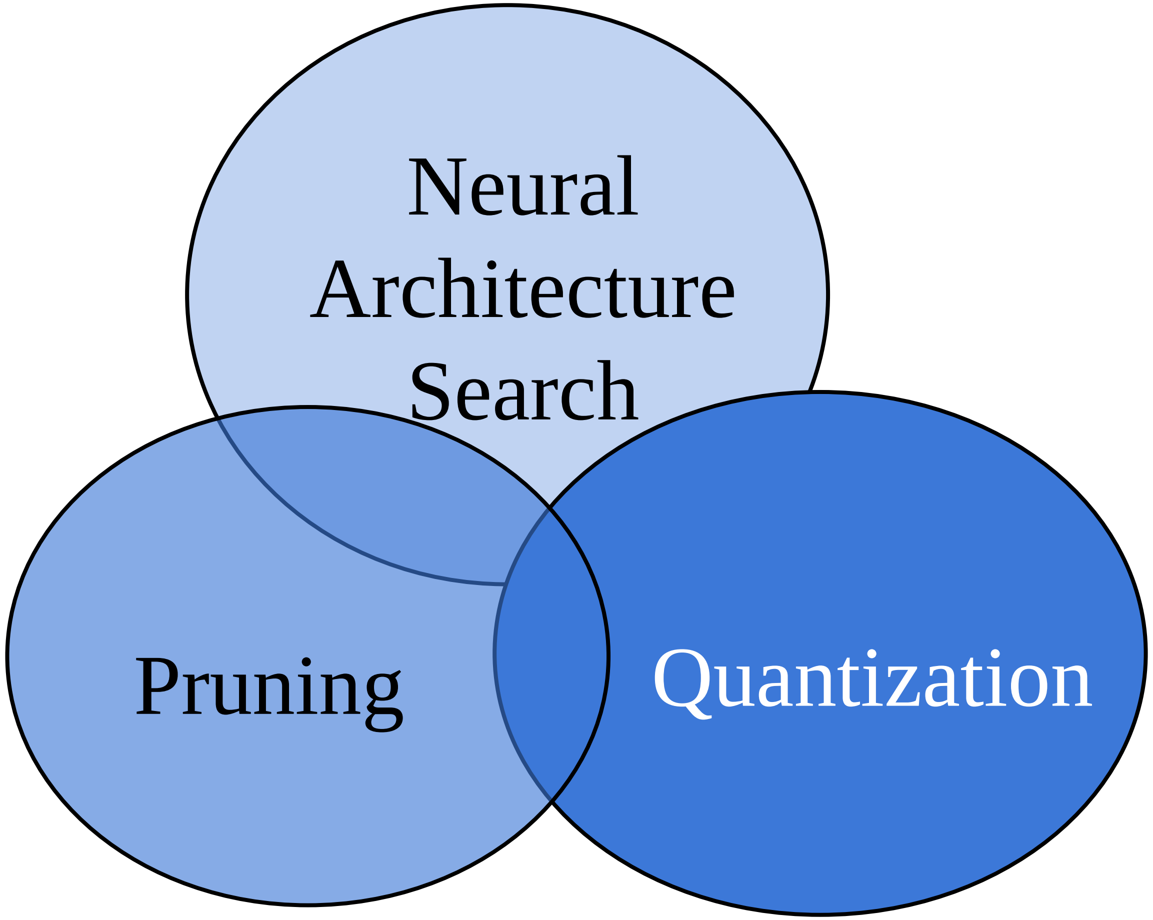


Problem: Need for Automated Search Method



- Total Number of distinct combinations for "N" layer = 9^N
- Example: For Resnet50, number of combinations = 9^{50} (practically impossible to go through each combination)
- Hence, automated Neural Architecture Search method is required to find optimal combination

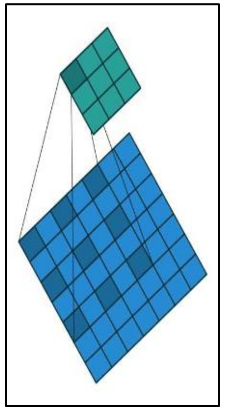
Deep Neural Network Optimization



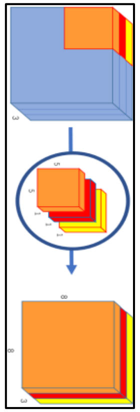
Neural Architecture Search (NAS)

Search Space

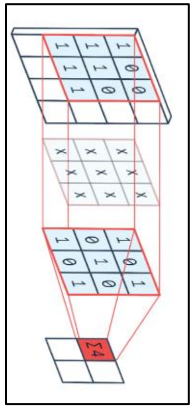
Dilation Convolution

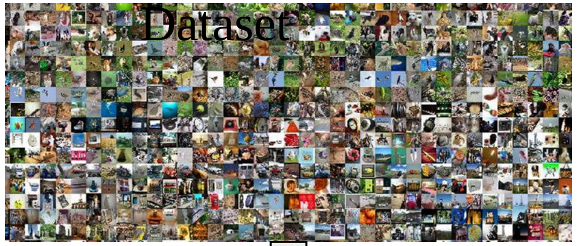


Depthwise Convolution



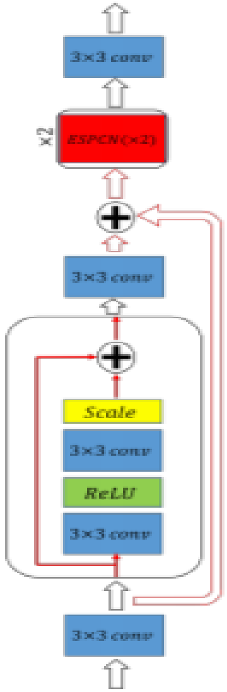
Spatial Convolution

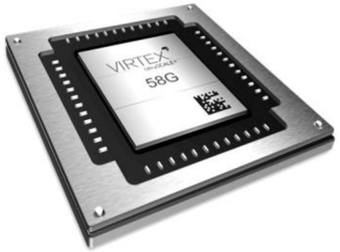




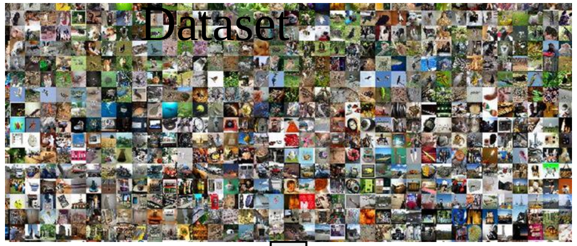
Hardware-aware

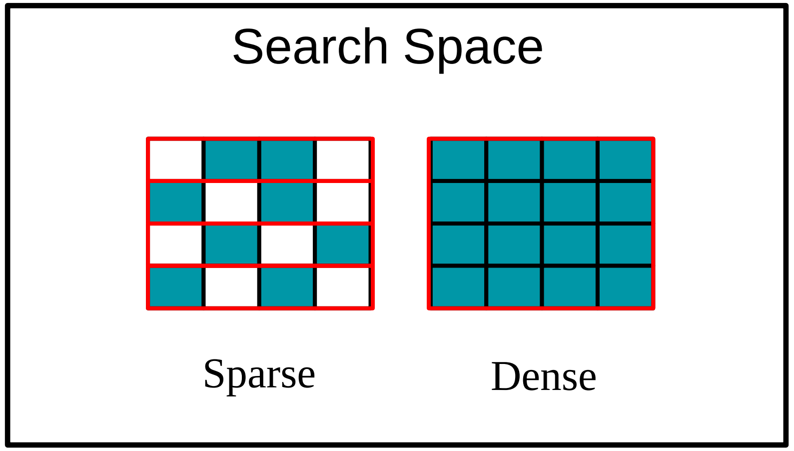
Neural Architecture Search





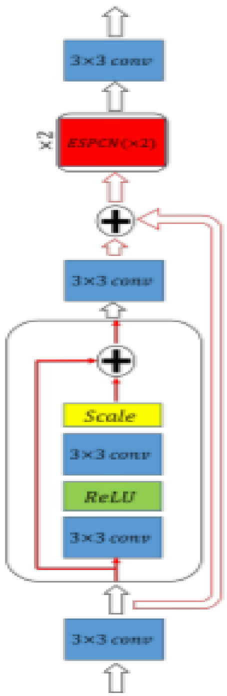
Neural Architecture Search (NAS)



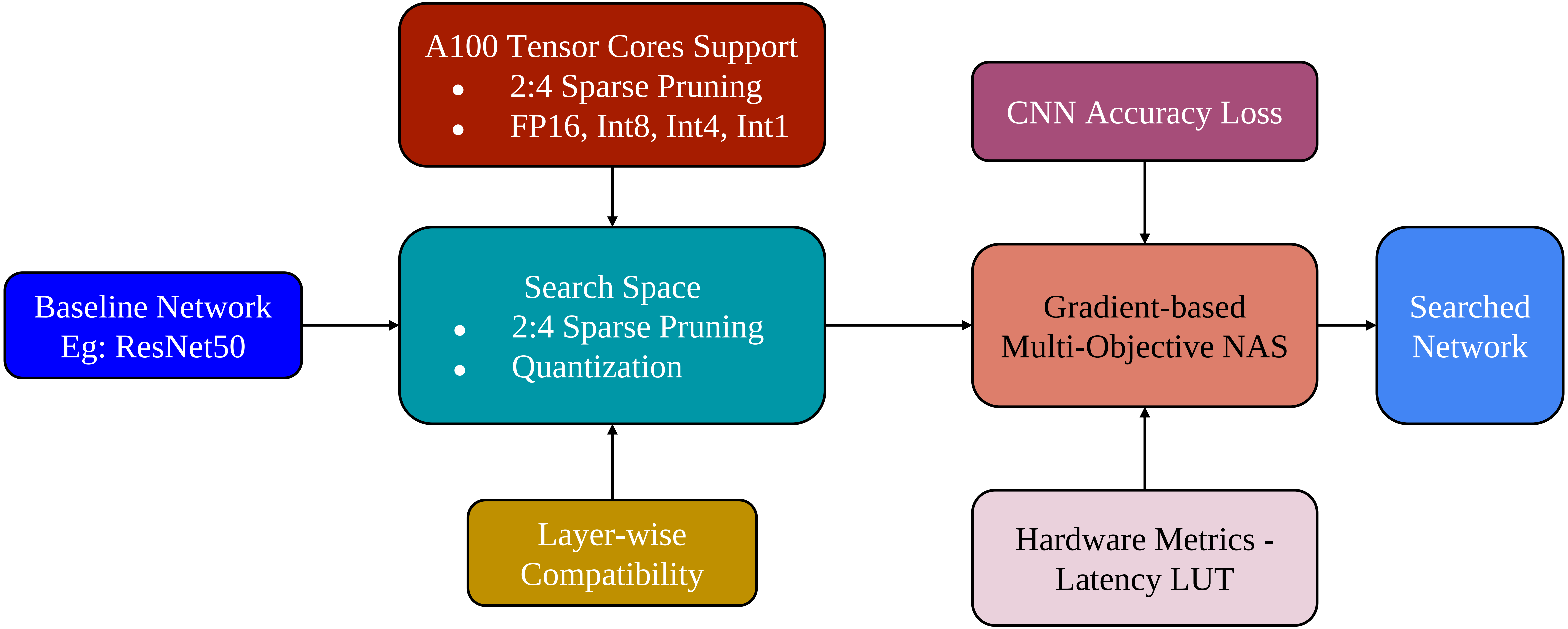


A100 Tensor Core-aware

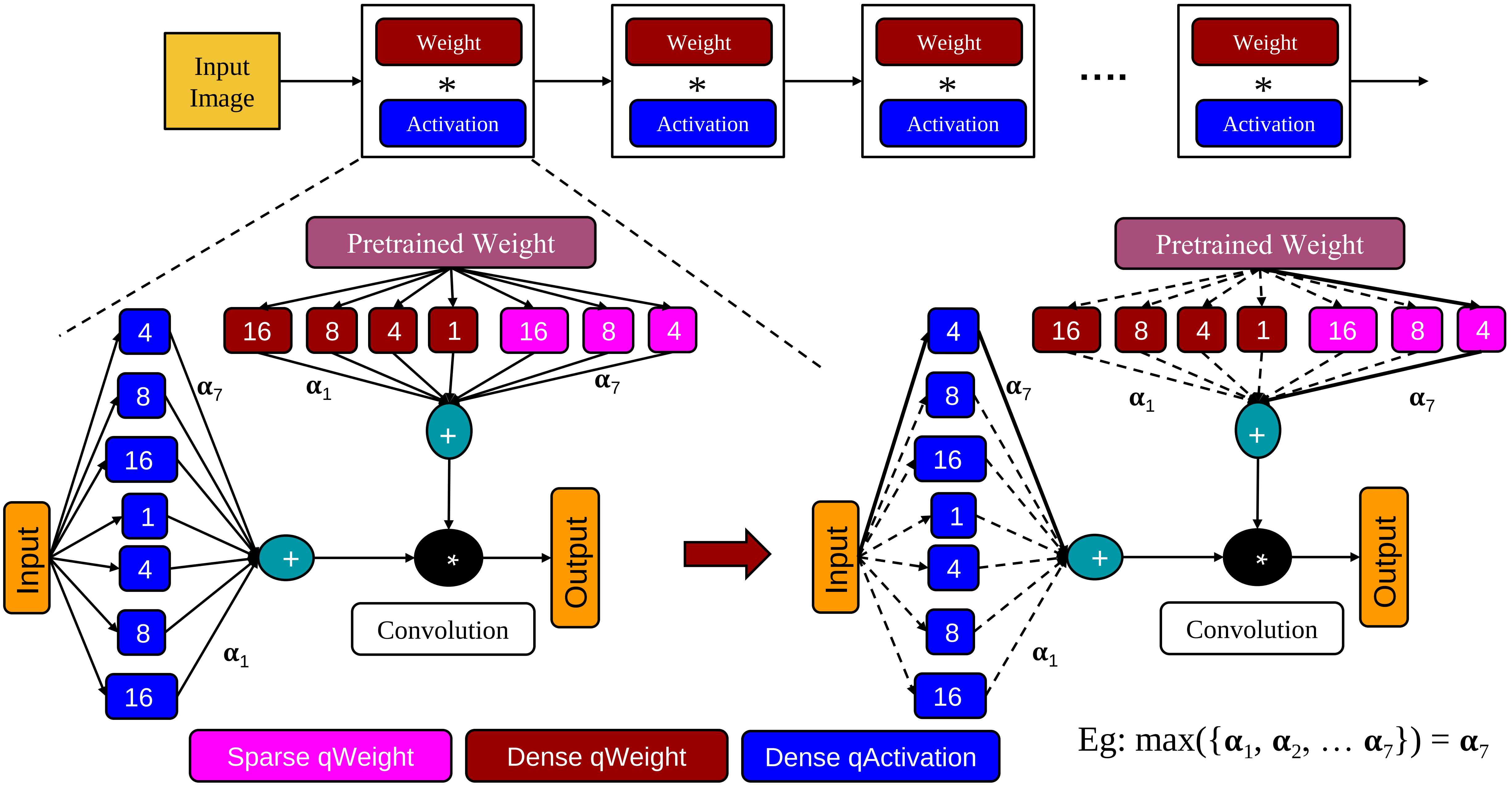
Neural Architecture Search



Neural Architecture Search Schematic for A100 GPUs

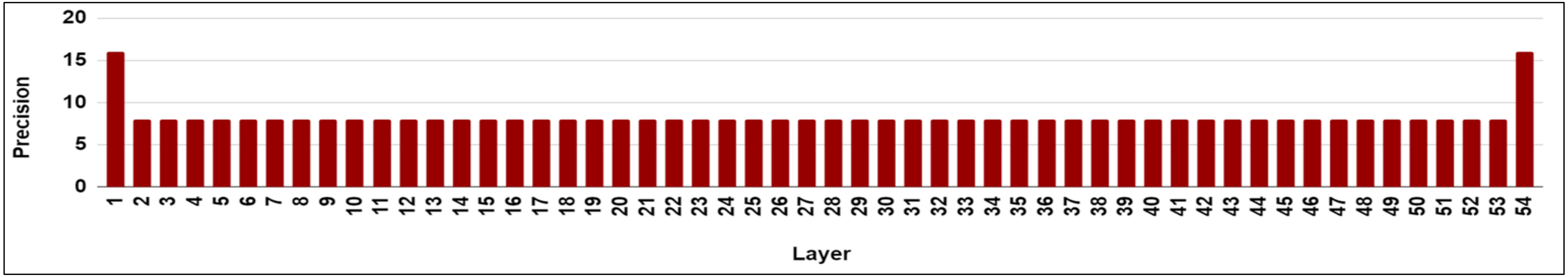


Mixed Sparse & Precision Search (MSPS) - *Sparse and Mixed Precision Quantized Supernetwork*



Eg: $\max(\{\alpha_1, \alpha_2, \ldots \alpha_7\}) = \alpha_7$

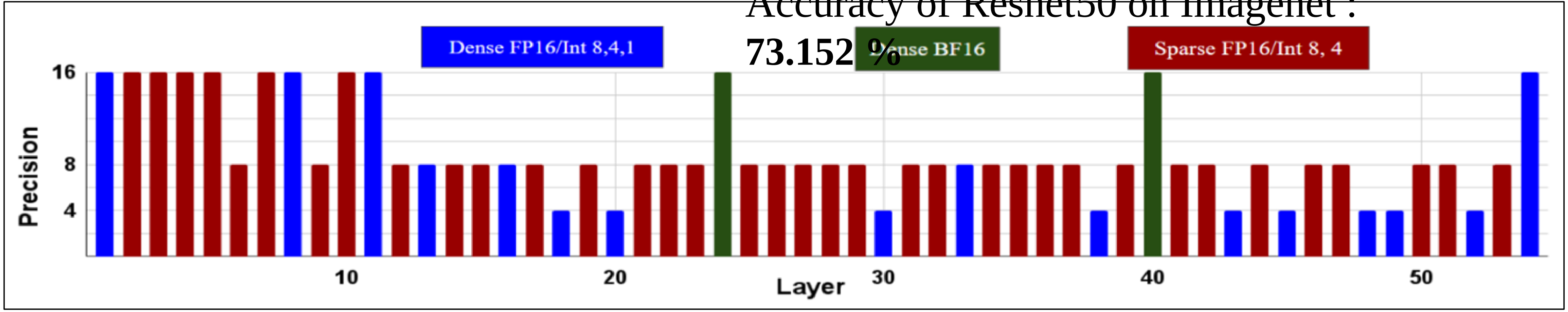
*Results: **ResNet50** on **Imagenet Dataset***



Uniform Sparse Int8 Network on Resnet50

Latency on Nvidia A100 GPU : **52.8 ms**

Accuracy of Resnet50 on Imagenet : **73.152 %**



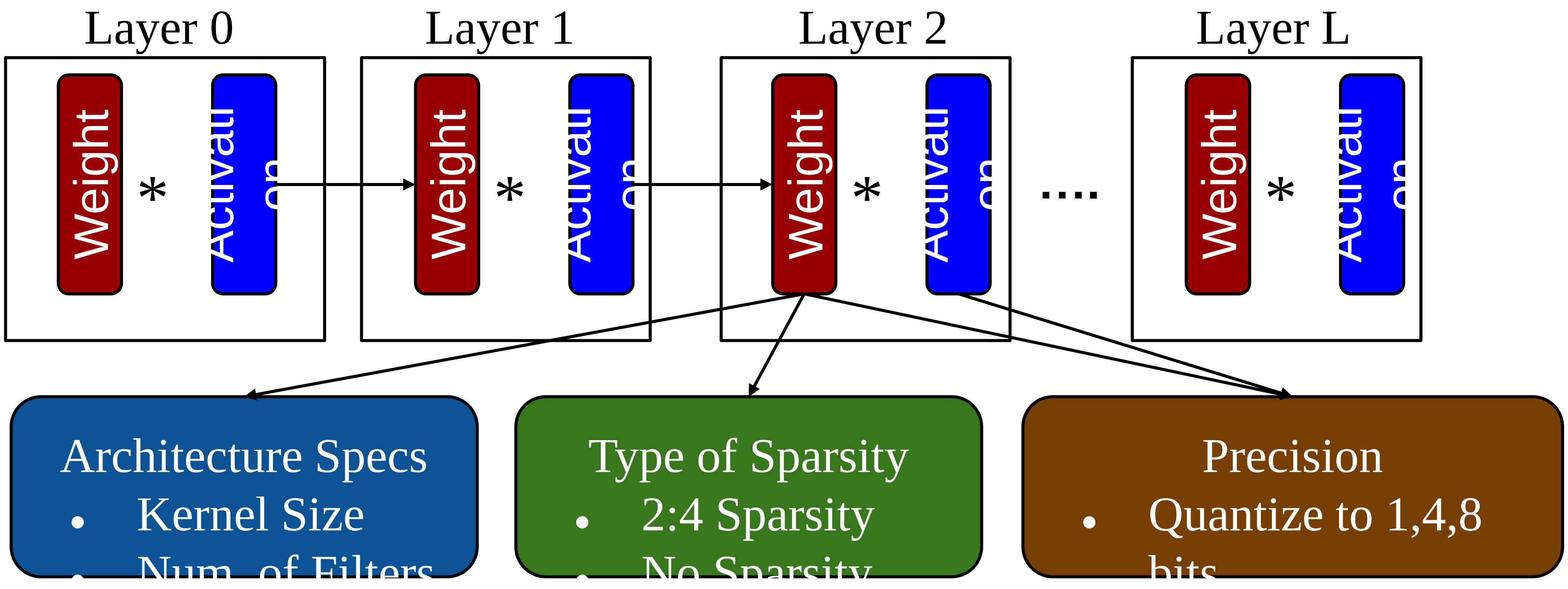
Our Mixed Searched Network on Resnet50

Latency on Nvidia A100 GPU : **52.37 ms**

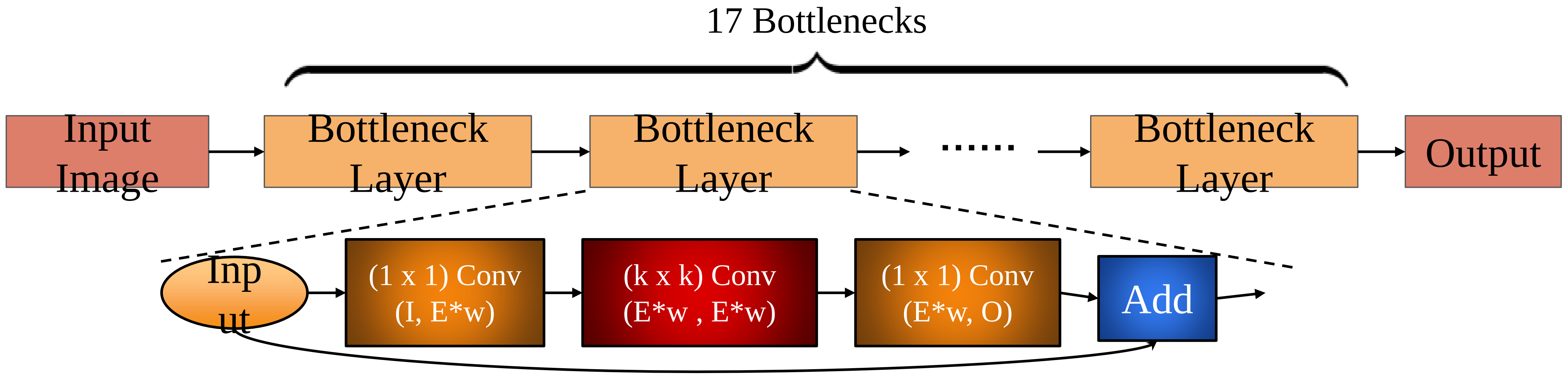
Accuracy of Resnet50 on Imagenet : **73.32 %**

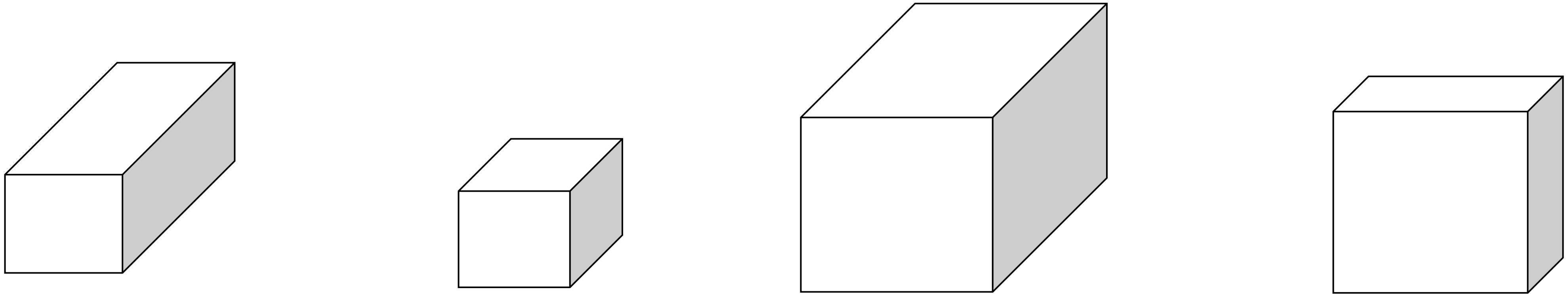
Neural Architecture, Sparsity and Precision Search (ASPS)

- Problem: Simultaneously search for the following dimensions for Ampere 100 Tensor Cores:
  - **Architecture Choices:** Kernel Size and Number of Filters
  - **Optimization:** Mixed Sparse and Precision Combination



Benchmark: ResNet50 Network on Imagenet dataset





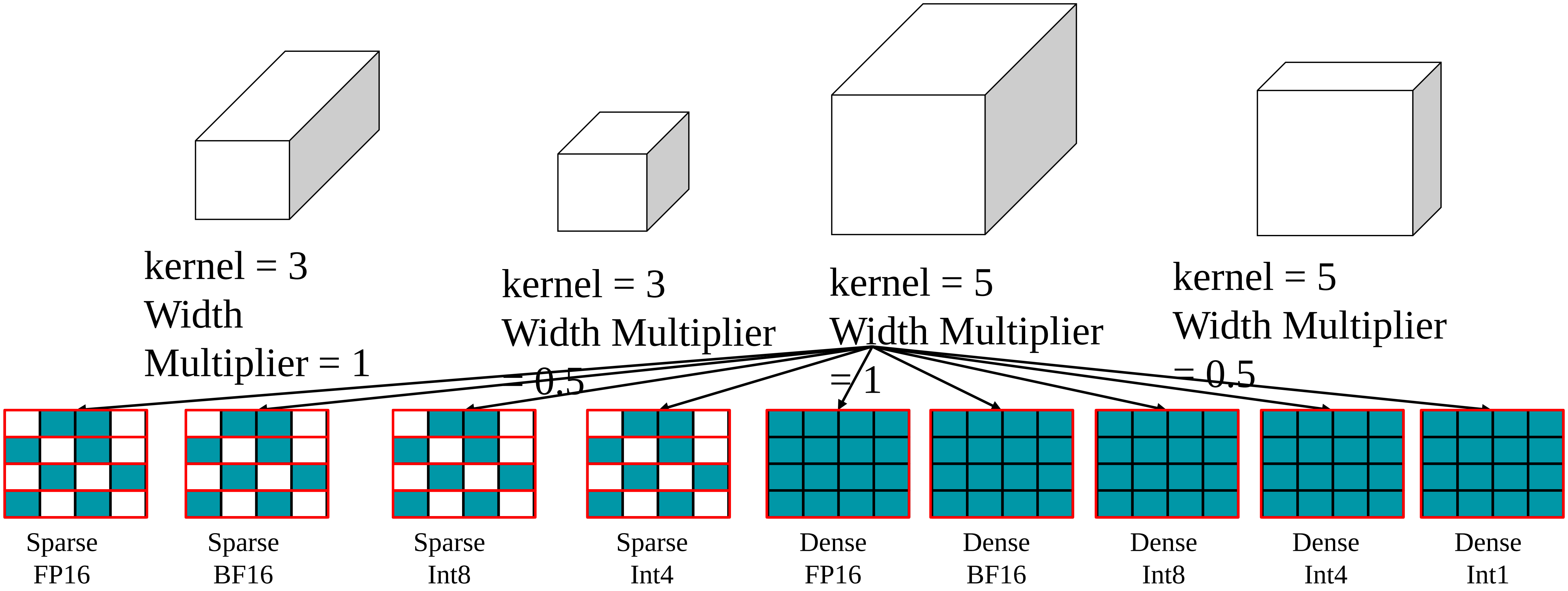
kernel = 3
Width Multiplier = 1

kernel = 3
Width Multiplier = 0.5

kernel = 5
Width Multiplier = 1

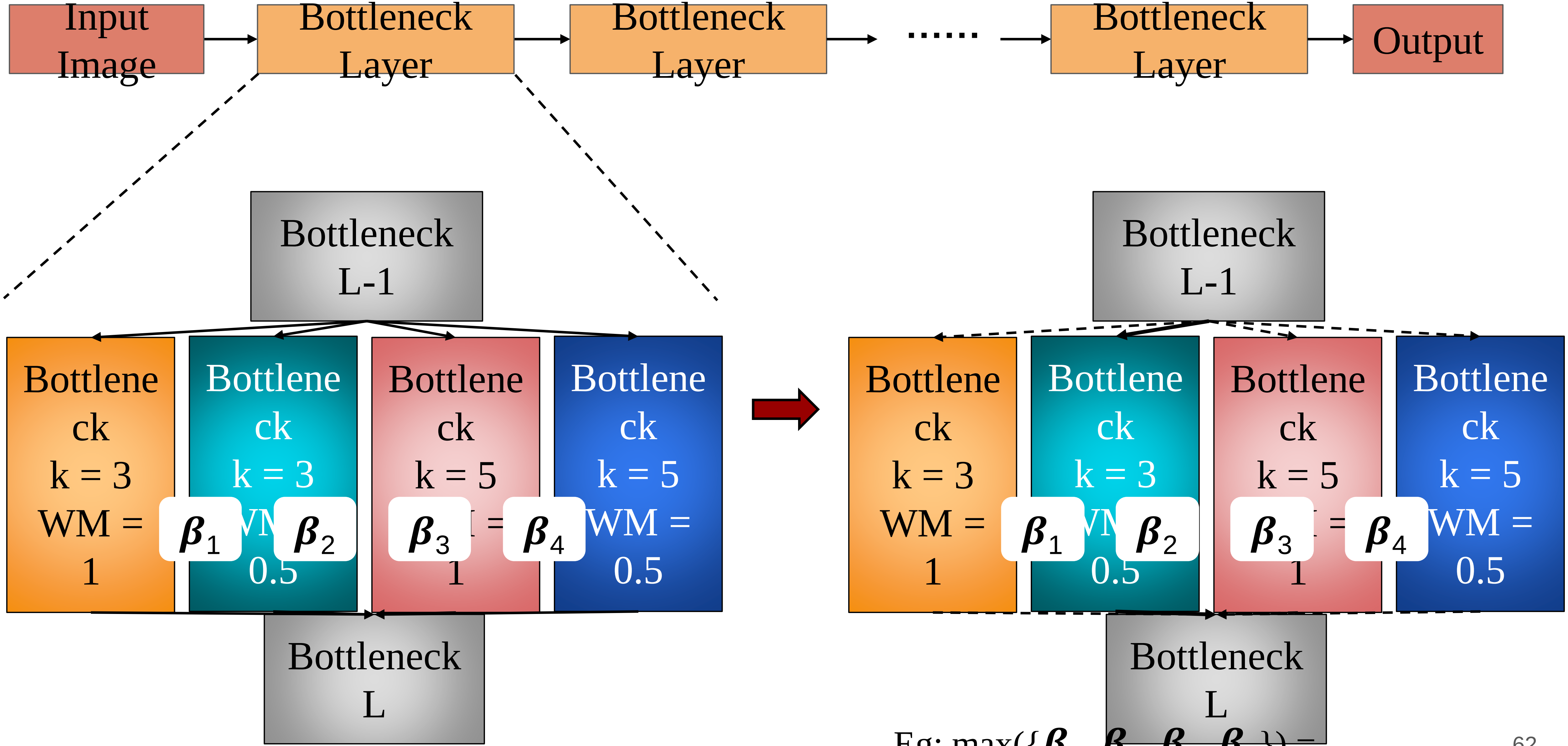
kernel = 5
Width Multiplier = 0.5

Architecture and Sparse-Precision Search Space on ResNet50 Network



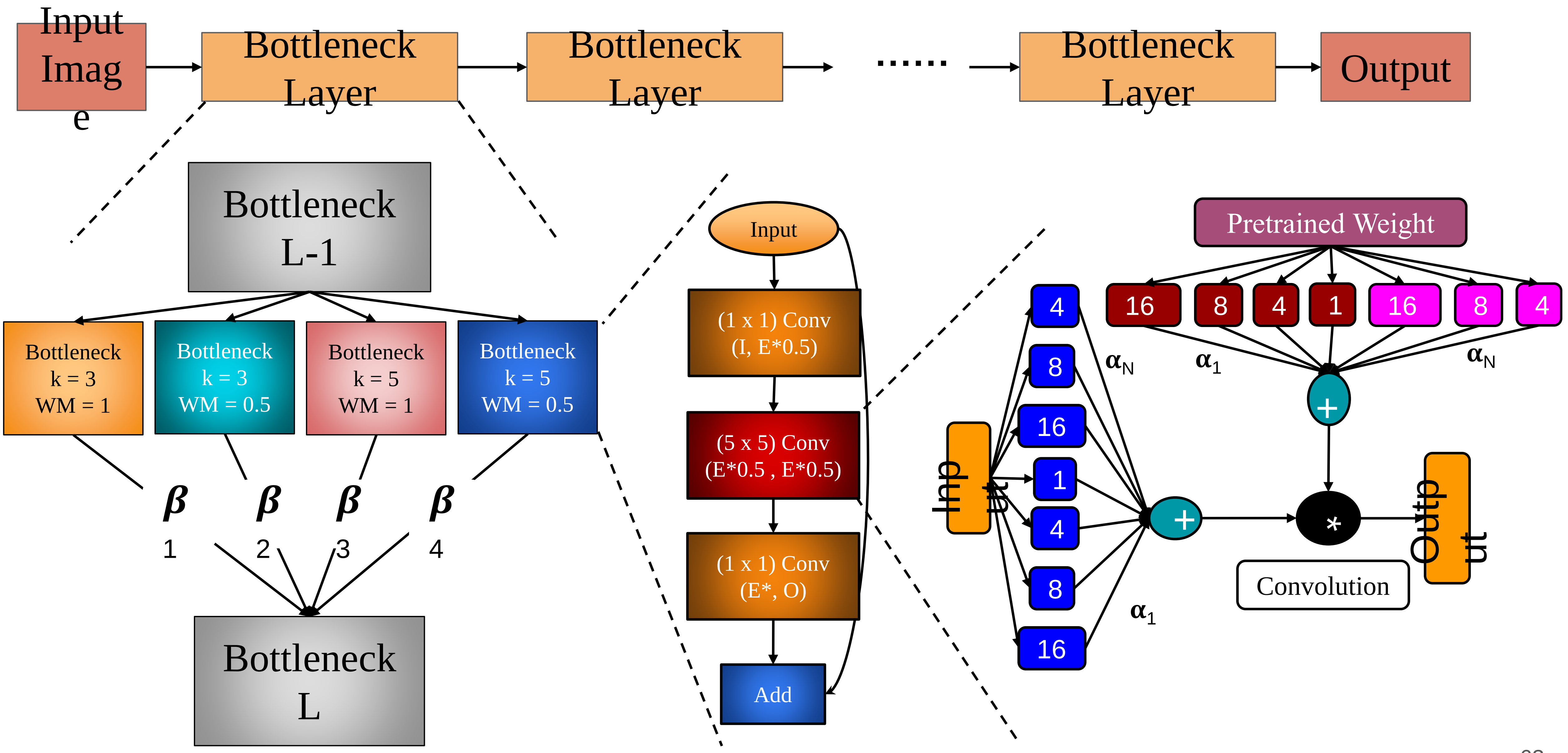
Number of Distinct Combinations: $17^{36} * 9^{35}$

Architecture Search Supernetwork

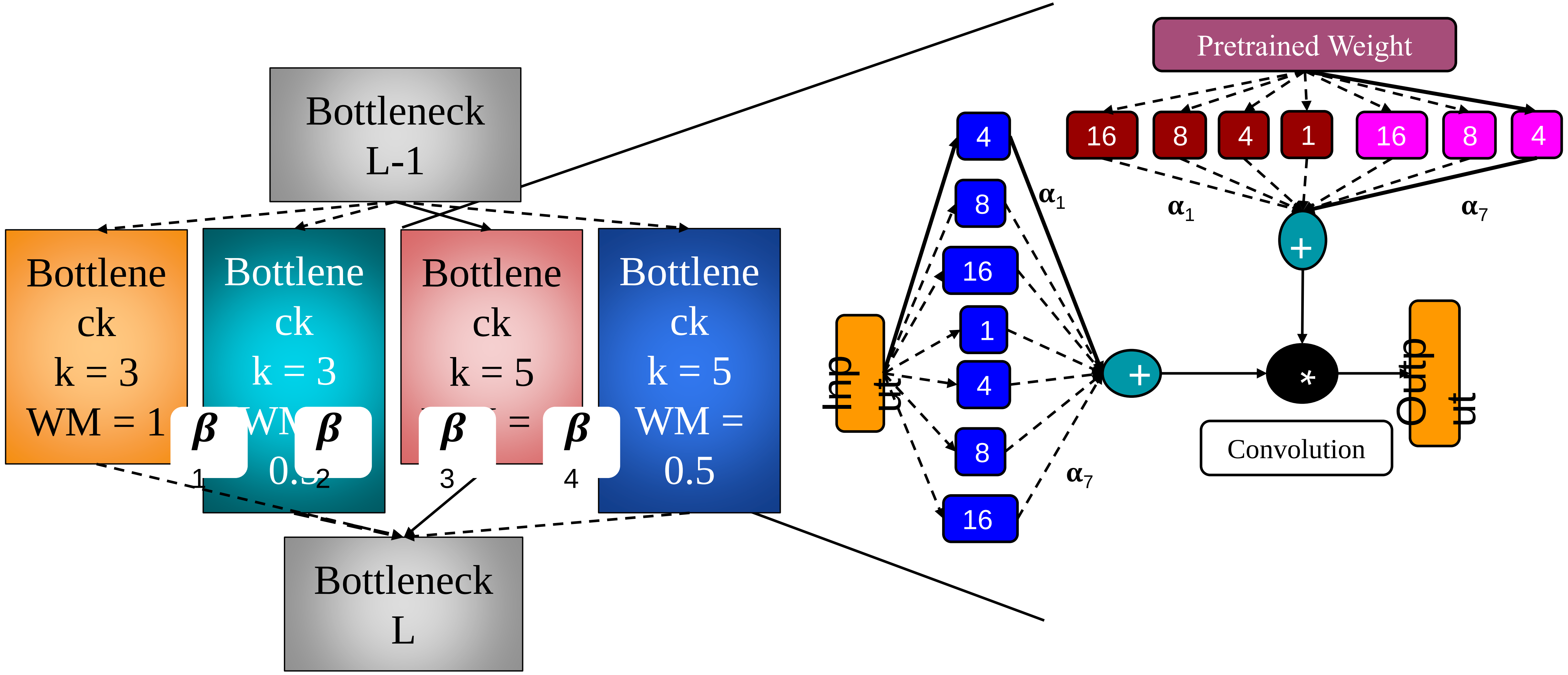


Eg: $\max(\{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3, \boldsymbol{\beta}_4\}) = \boldsymbol{\beta}$

Architecture, Sparse & Mixed Precision Search Supernetwork



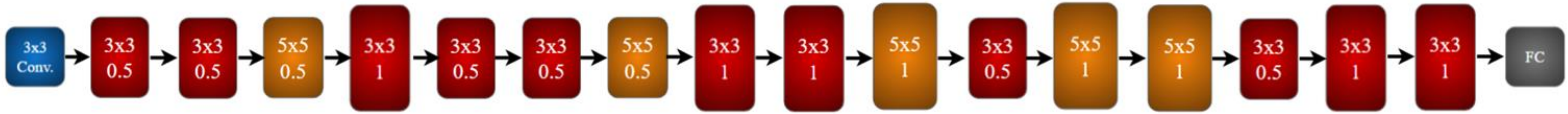
Sampling the best Architecture, Sparse and Precision Combination



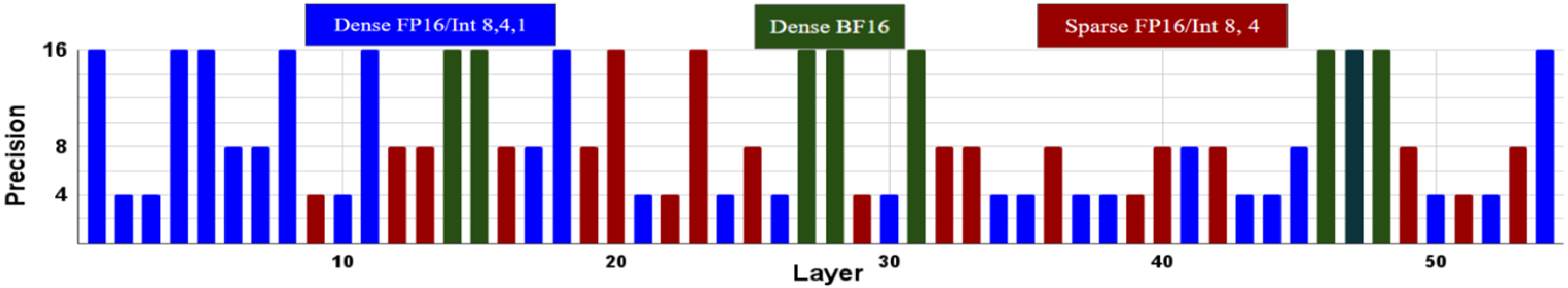
$\max(\{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3, \boldsymbol{\beta}_4\}) = \boldsymbol{\beta}_3$

$\max(\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots \boldsymbol{\alpha}_7\}) = \boldsymbol{\alpha}_7$

Searched Resnet50 Network on Imagenet dataset



End-to-end Microarchitecture on Resnet50 Search Space



End-to-end Sparse and Mixed Precision Quantization Combination of the corresponding Searched Network

Architecture, Mixed Sparse & Precision Search (ASPS) - Performance Comparison between Uniform Sparse Int8 and our Searched Networks

CIFAR dataset

Configuration	Lat. (ms)	CIFAR10 Acc. (%)	CIFAR100 Acc. (%)
Uniform Sparse Int8	5.4	94.32	74.29
MSPS Most Efficient Model	5.27	**94.76**	74.54
ASPS best Model	**5.19**	94.38	**74.7**

Imagenet dataset

Configuration	Lat. (ms)	Acc. (%)
Uniform Sparse Int8	52.8	73.15
MSPS Most Efficient Model	49.44	72.86
ASPS Model ($\lambda = 0$)	52.8	73.42
ASPS best Model	**49.36**	**73.72**

Conclusion

We developed the following methods:

- ***Mixed Sparse and Precision Search (MSPS):*** to search for the optimal weight matrix type (sparse or dense) and precision (FP16, BF16, Int8, Int4, Int1) combination for every layer

- ***Architecture, Sparsity, and Precision Search (ASPS):*** a method to search for better hyperparameters (kernel and filter sizes) along with the matrix type and bit-width in the same loop

- Our searched models on Resnet50 Search Space outperforms the manually designed models on the ImageNet dataset

- For example, our best ASPS model is ~1.1x faster and 0.57% more accurate than the baseline sparse-only Integer 8 ResNet50 network