Campo

Cost-Aware Performance Optimization for Mixed-Precision Neural Network Training

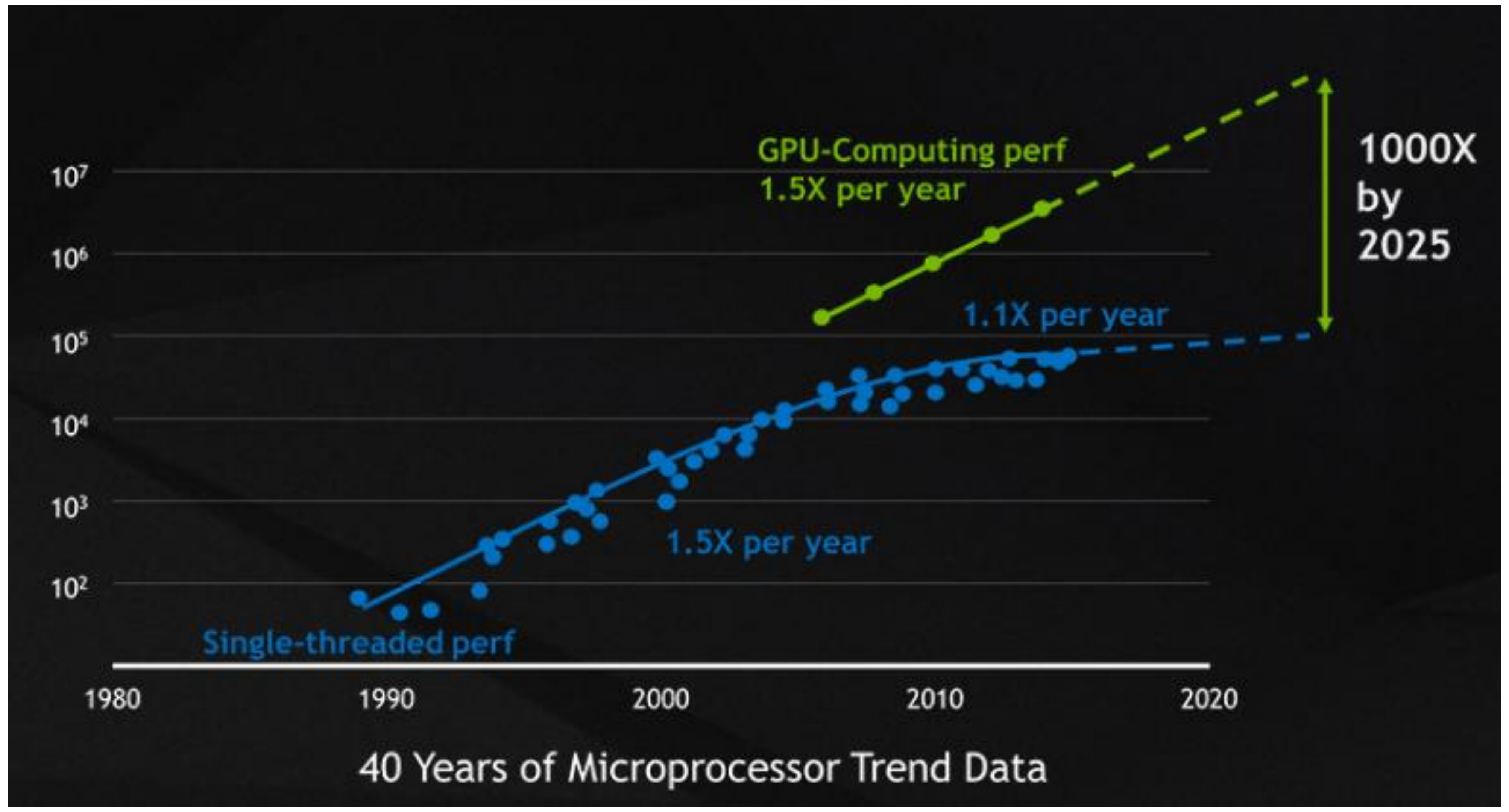
USENIX ATC'22

Xin He, CSEE, Hunan University & Xidian University; Jianhua Sun and Hao Chen,CSEE, Hunan University; Dong Li,
University of California, Merced

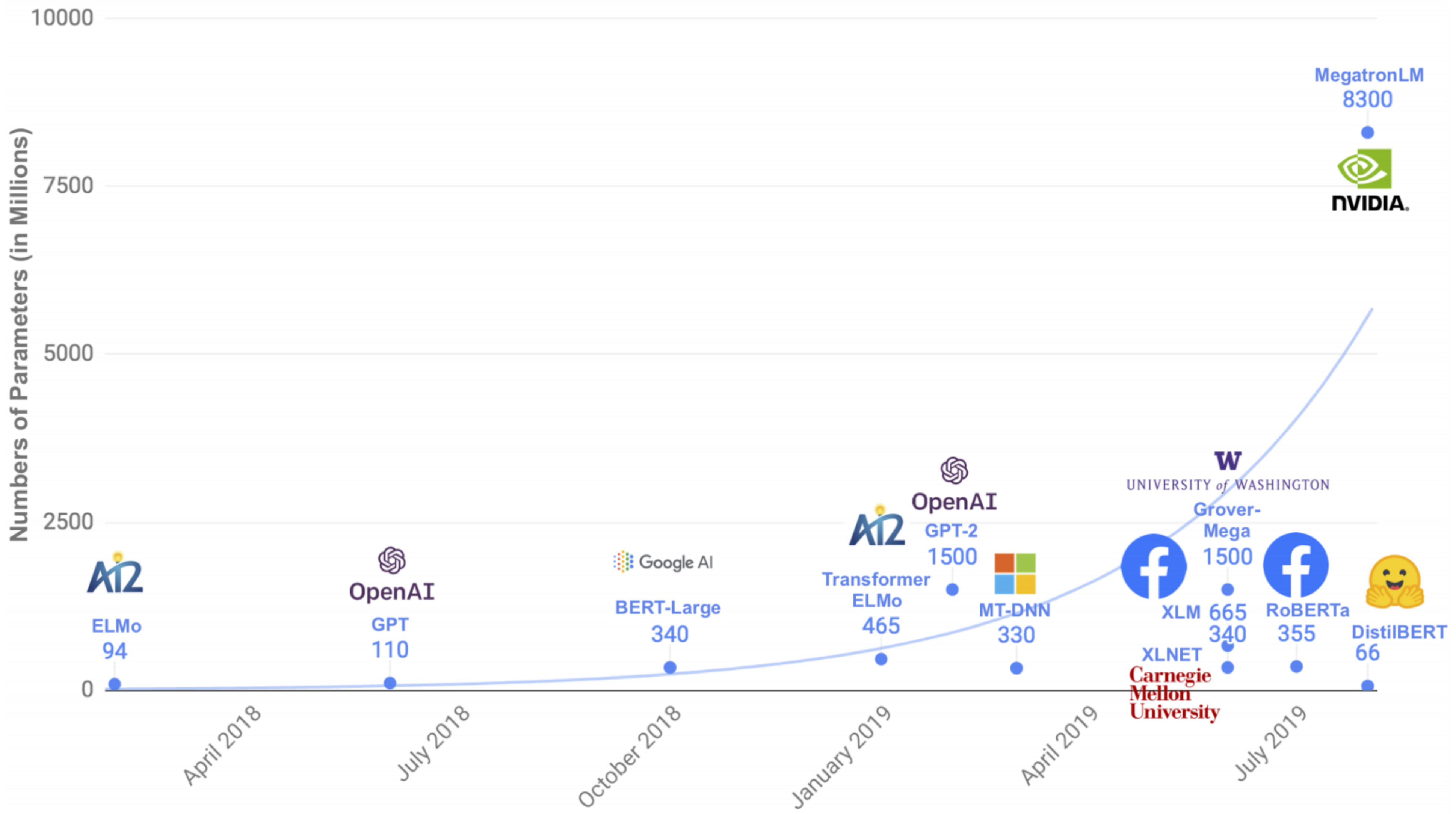
Introduction

Usage of **GPU** in AI field is getting more and more common

Model size is getting huge



GPU & CPU performance



Model size

Introduction

Present model require:

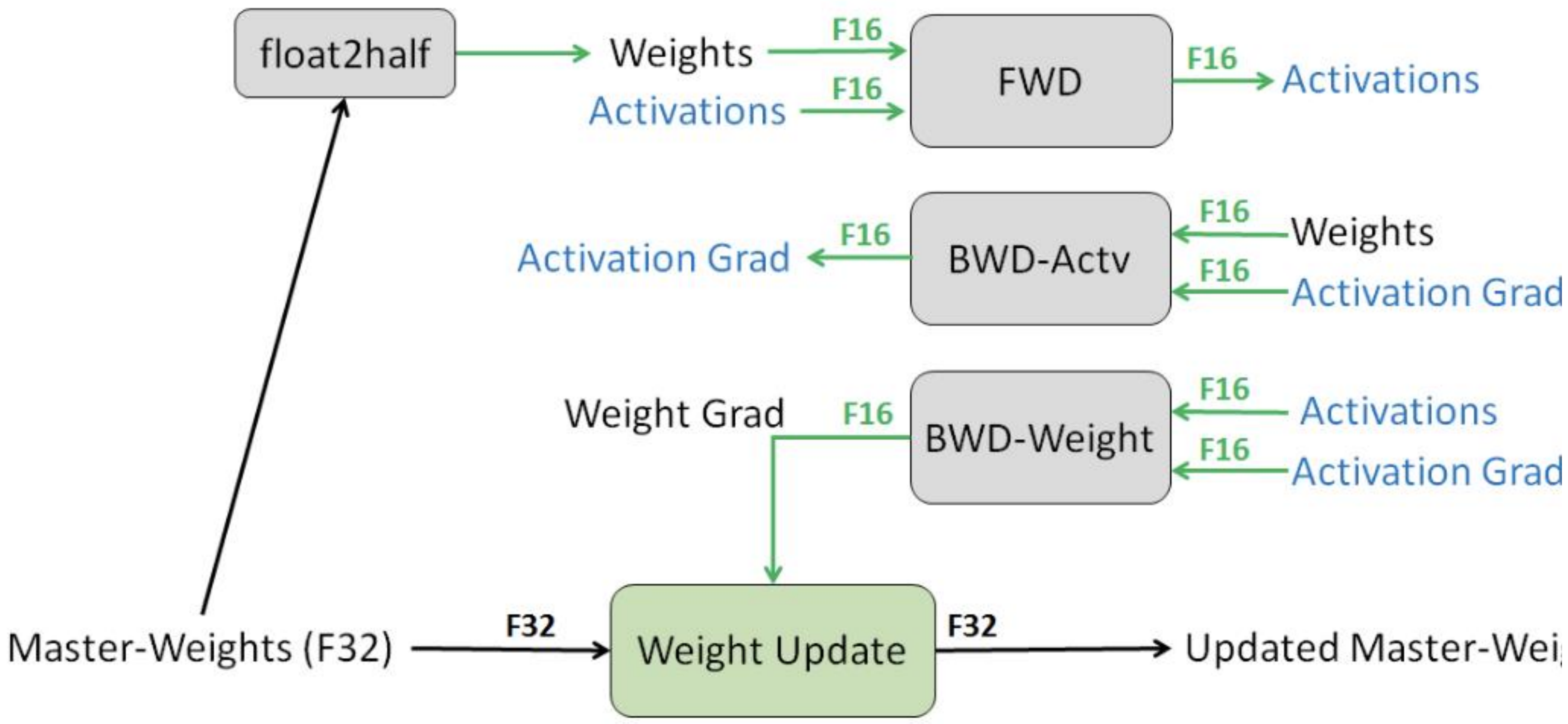
- high memory **bandwidth and capacity**
- High **calculation speed**
- Better **energy efficiency**

Mixed precision training

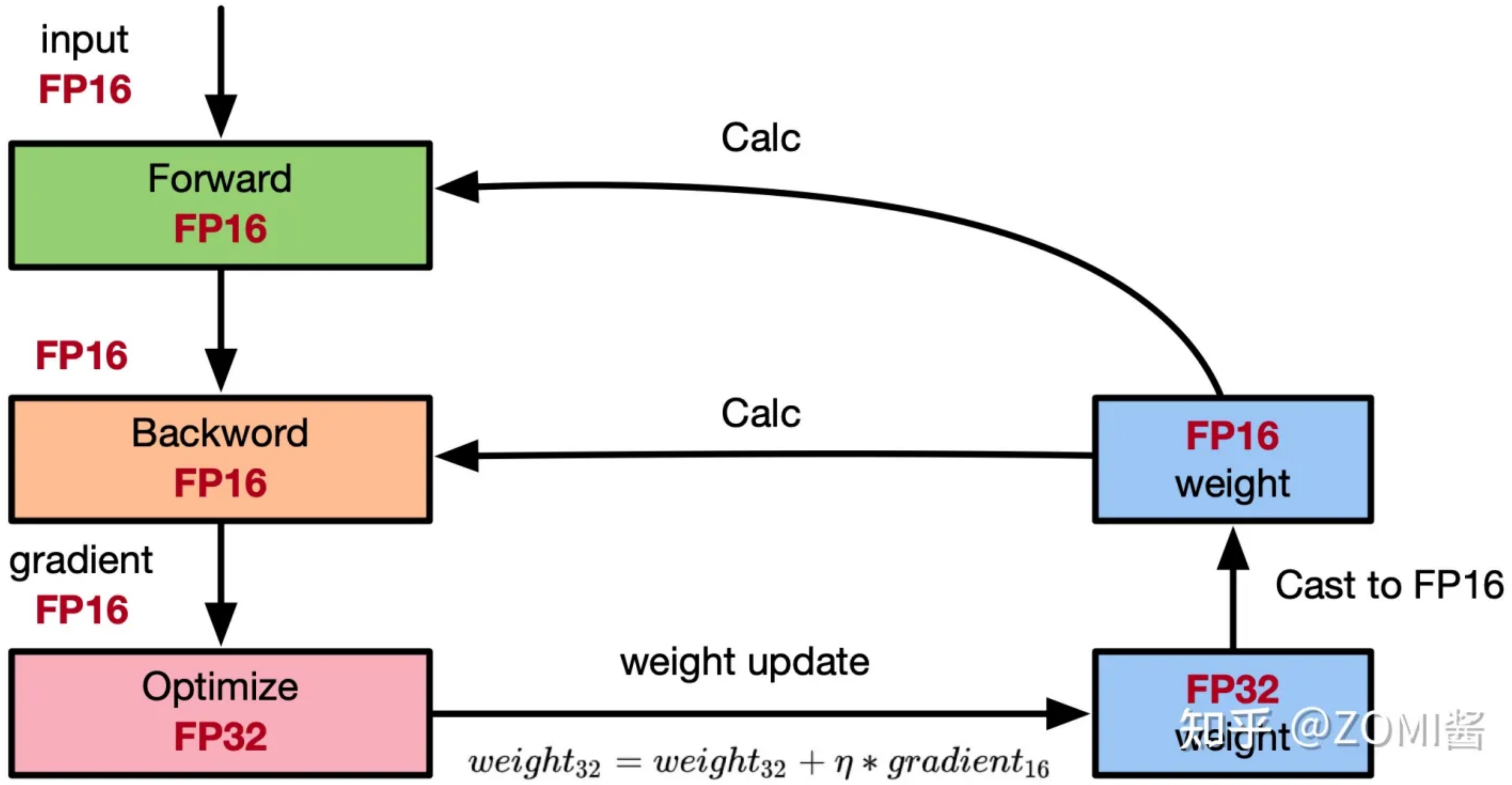
Mixed precesion training arXiv:1710.03740v3 [cs.AI] 15 Feb 2018

Introduction

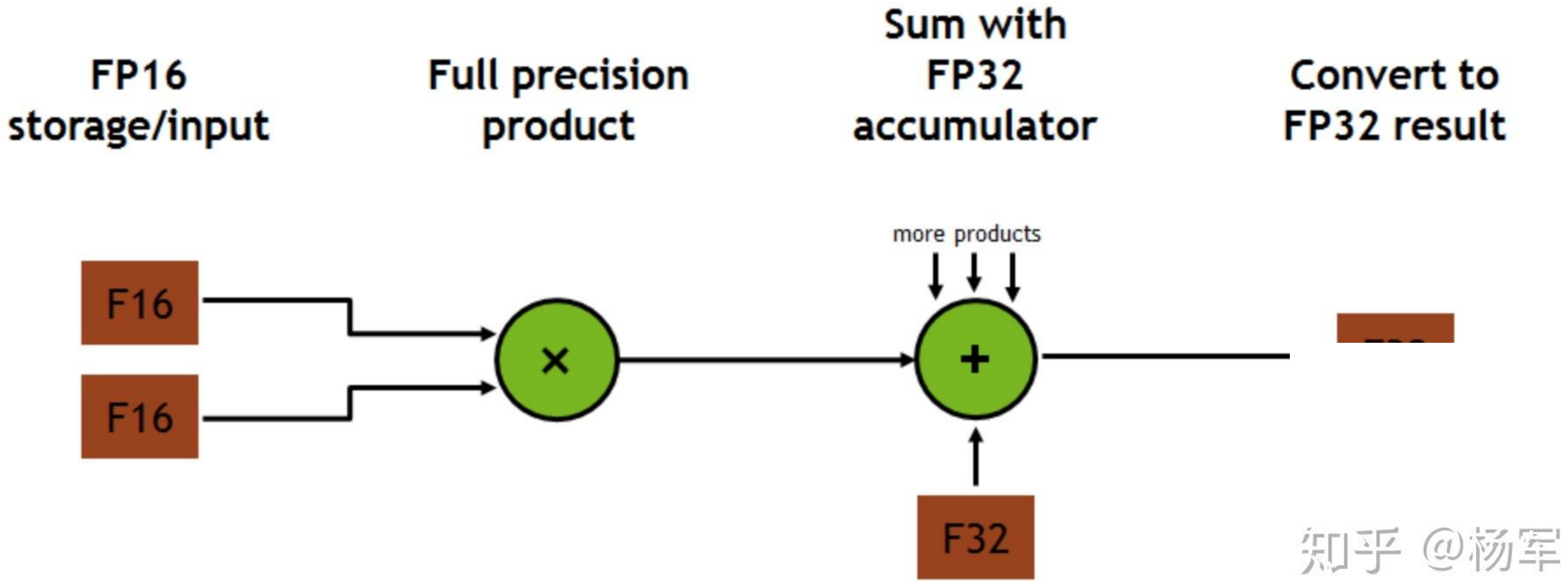
What is mixed precision training ?



weights, activations and gradients are stored as FP16



weight backup



Multiply in low precision , sum with full precision

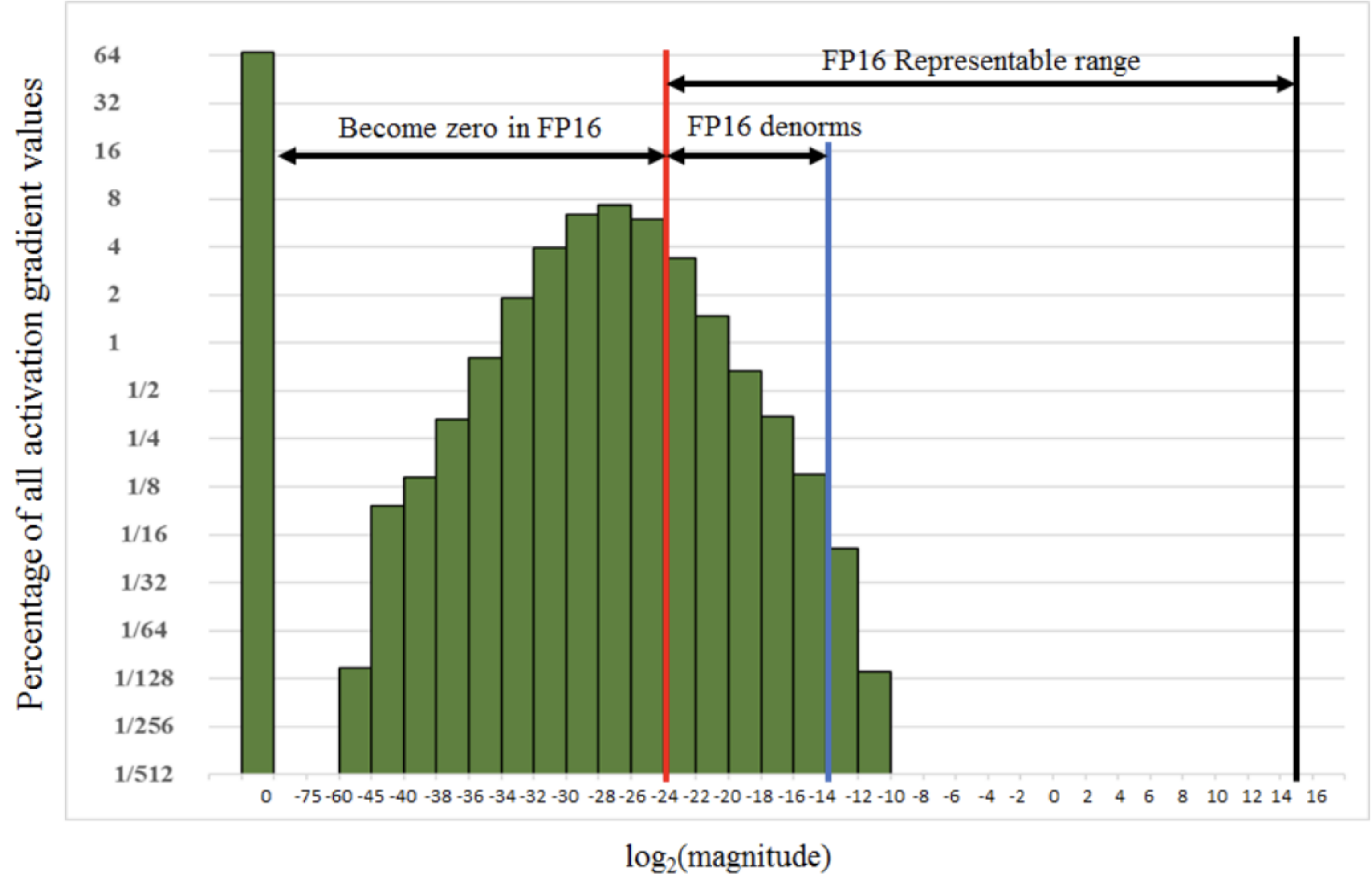
Introduction

What is mixed precision training ?

Balance

training speed ⟷ **accuracy loss**

- Training throughput
- memory bandwidth and capacity
- Algorithm power limit
- Fast execution



small gradient may be shrunk to zero

Introduction

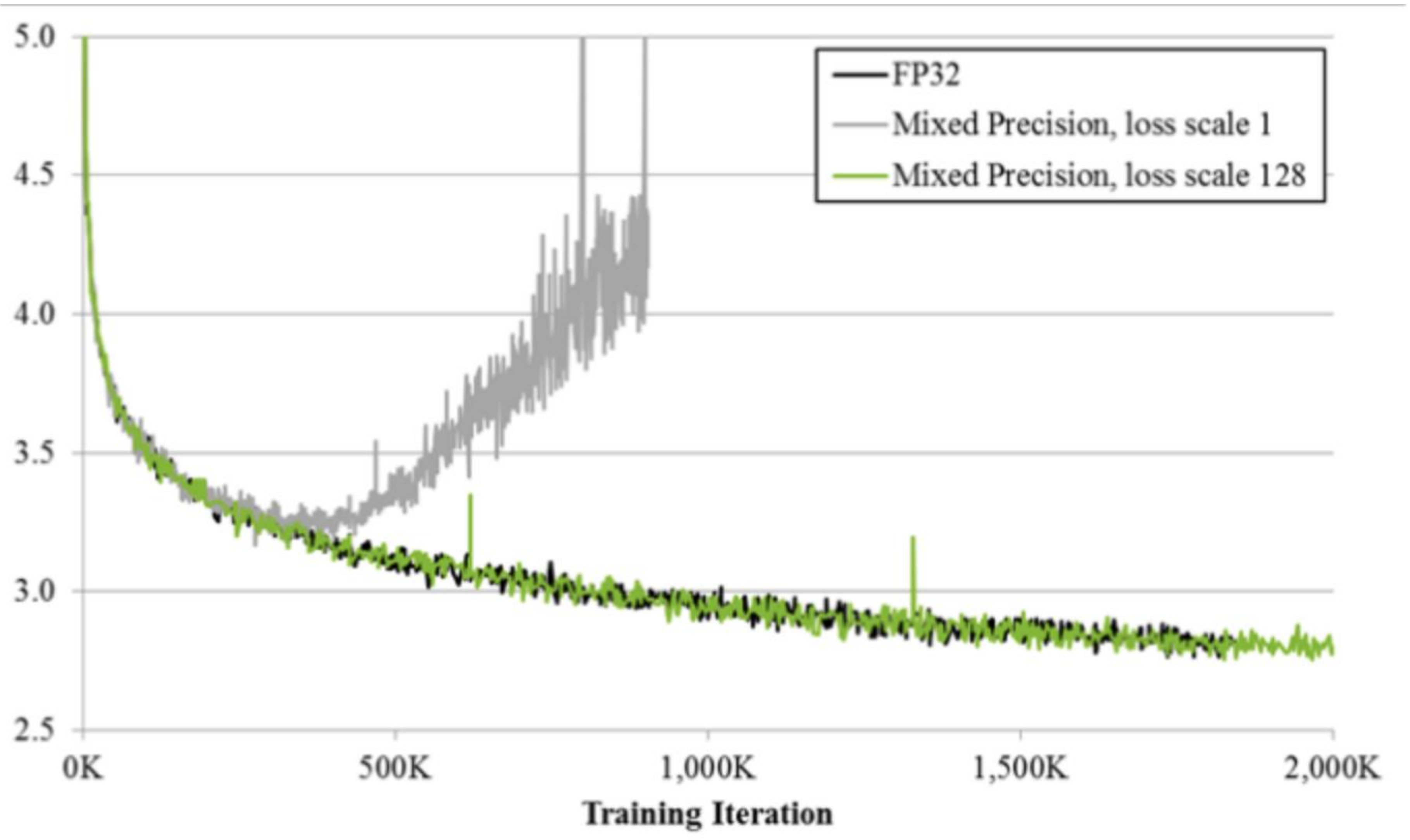
What is mixed precision training ?

Balance

training speed

accuracy loss

- Training throughput
- memory bandwidth and capacity
- Algorithm power limit
- Fast execution



Mixed precision

Could be solved by **loss scaling**

Introduction

How is mixed precision achieved by present framework ?

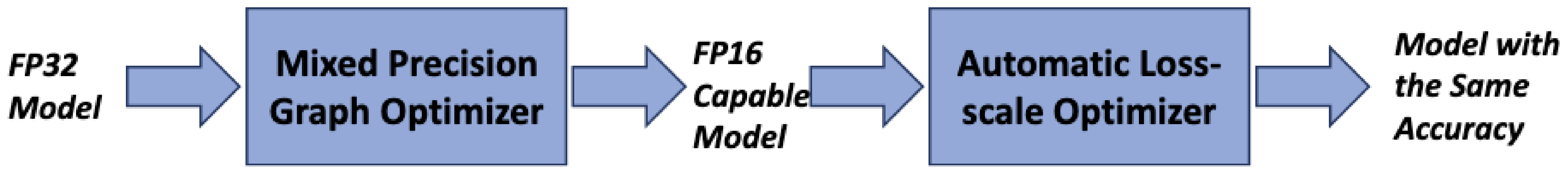


Figure 1: The workflow of mixed precision training.

PyTorch

TensorFlow

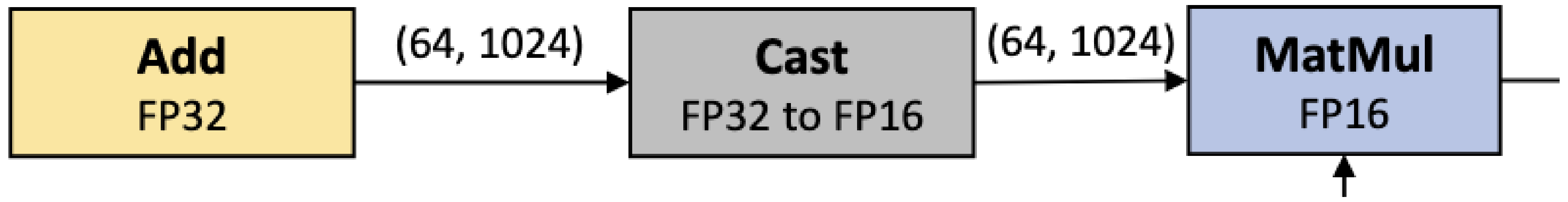
Only supported by **NVDIA** **in model-scale**

AMP, Automatic mixed precision

Introduction

How is mixed precision achieved by present framework ?

Step 1. identifying which nodes should be changed to FP16 and inserting casts between FP32 nodes and FP16 nodes by a mixed precision **graph optimizer (paper's focus)**



Step 2. adding loss scaling to preserve small gradient values by an **automatic loss-scale optimizer**

Introduction

How is mixed precision achieved by present framework ?

Step 1. Assign precision for each nodes and inserting casts by a mixed precision **graph optimizer (paper's focus)**

Ops are sorted into lists based on numerical safety

- Allowlist numerically-safe && performance-critical
- Denylist numerically-dangerous
- Inferlist numerically-safe on their own , affected by upstream denylist op
- Clearlist doesn't matter

Introduction

How is mixed precision achieved by present framework ?

Step 1. Assign precision for each nodes and inserting casts by a mixed precision **graph optimizer (paper's focus)**

The conditions to run op in low precision

1. in the allowlist
2. op in the clearlist , immediate ancestor(s) and descendent(s) are using low precision
3. in the inferlist and there is no upstream denylist op

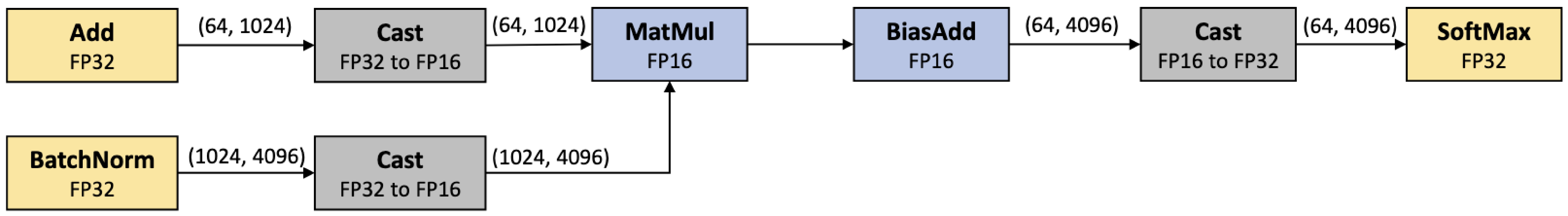
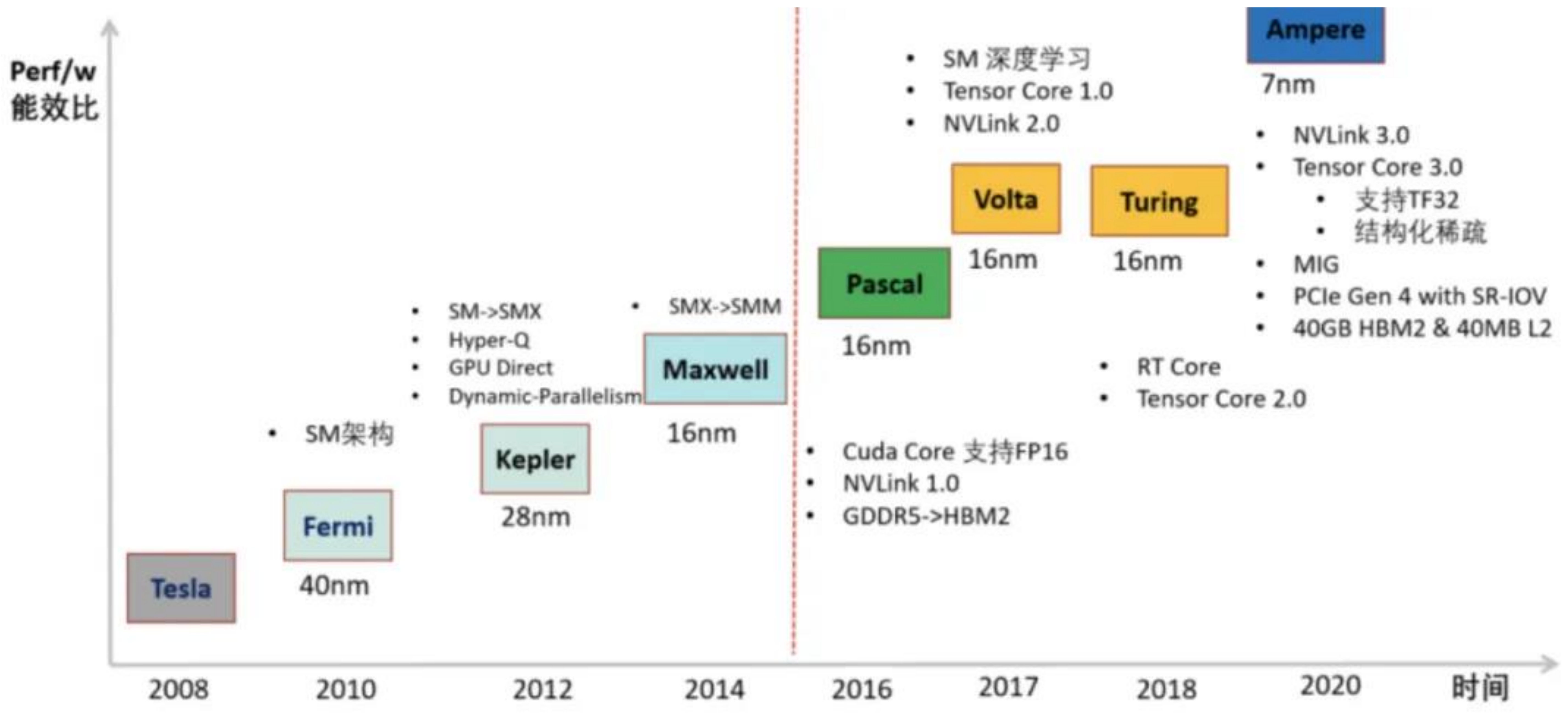


Figure 2: A snippet of the dataflow graph from BERT using mixed precision. The tuple on each edge represents a tensor with its shape (i.e., the size of each dimension).

Introduction

Hardware basement of mixed precision—TC cores

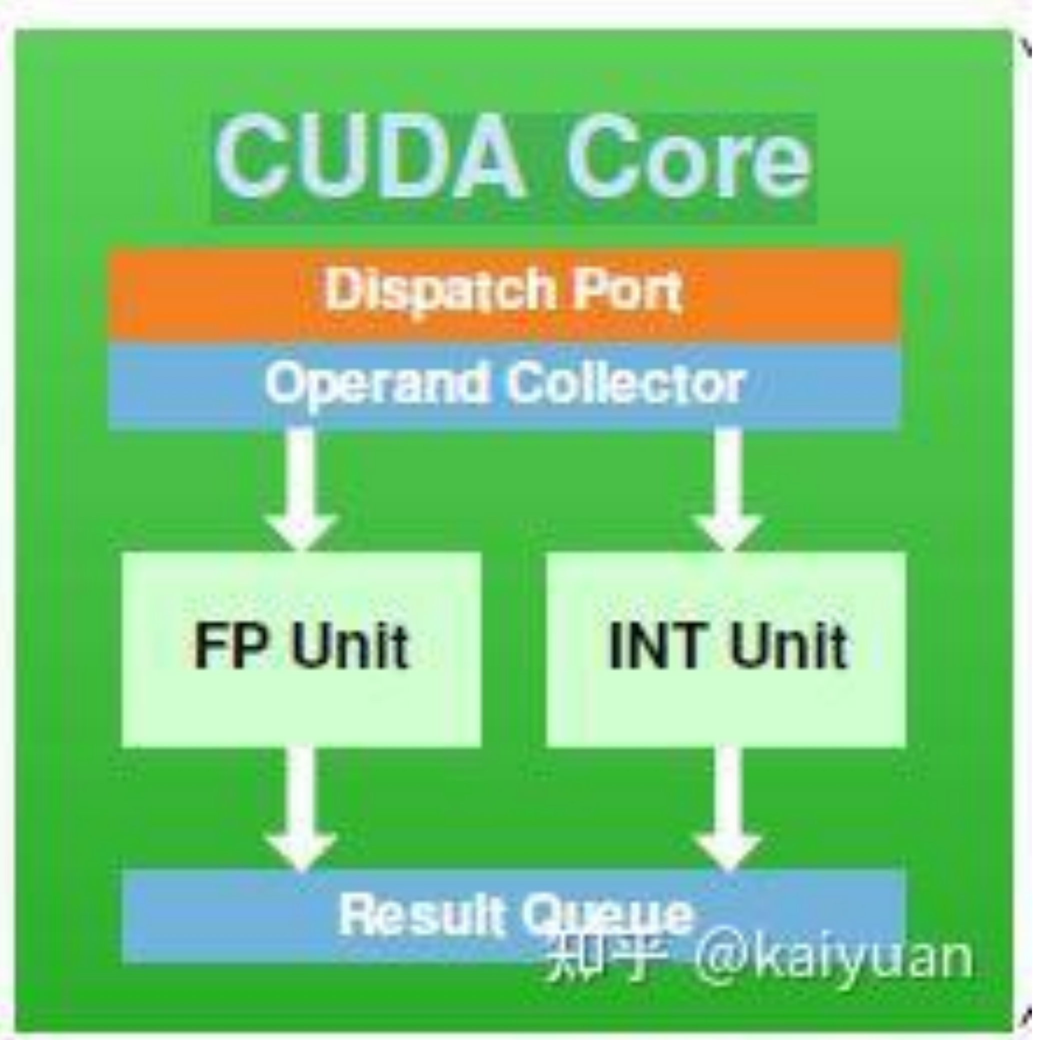


Development of NVIDIA GPU **architecture**

- Nano Process improvement
- CUDA core
- Tensor core (TC core)
- RT core

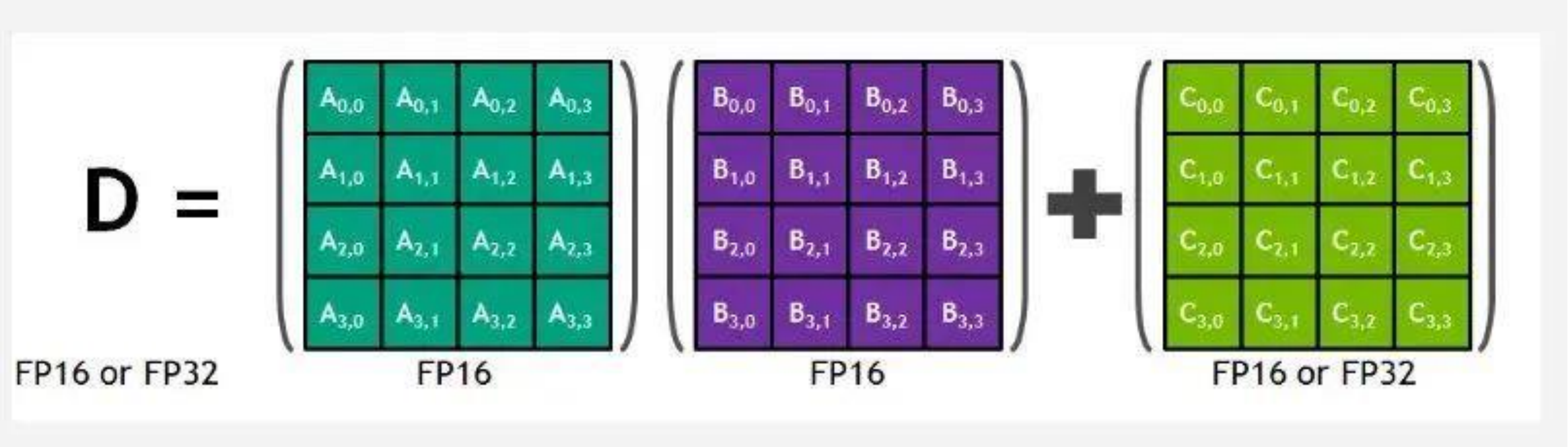
Introduction

Hardware basement of mixed precision—TC cores



- integer arithmetic logic unit (ALU)
- floating point unit (FPU)
- fused multiply-add (FMA)

General

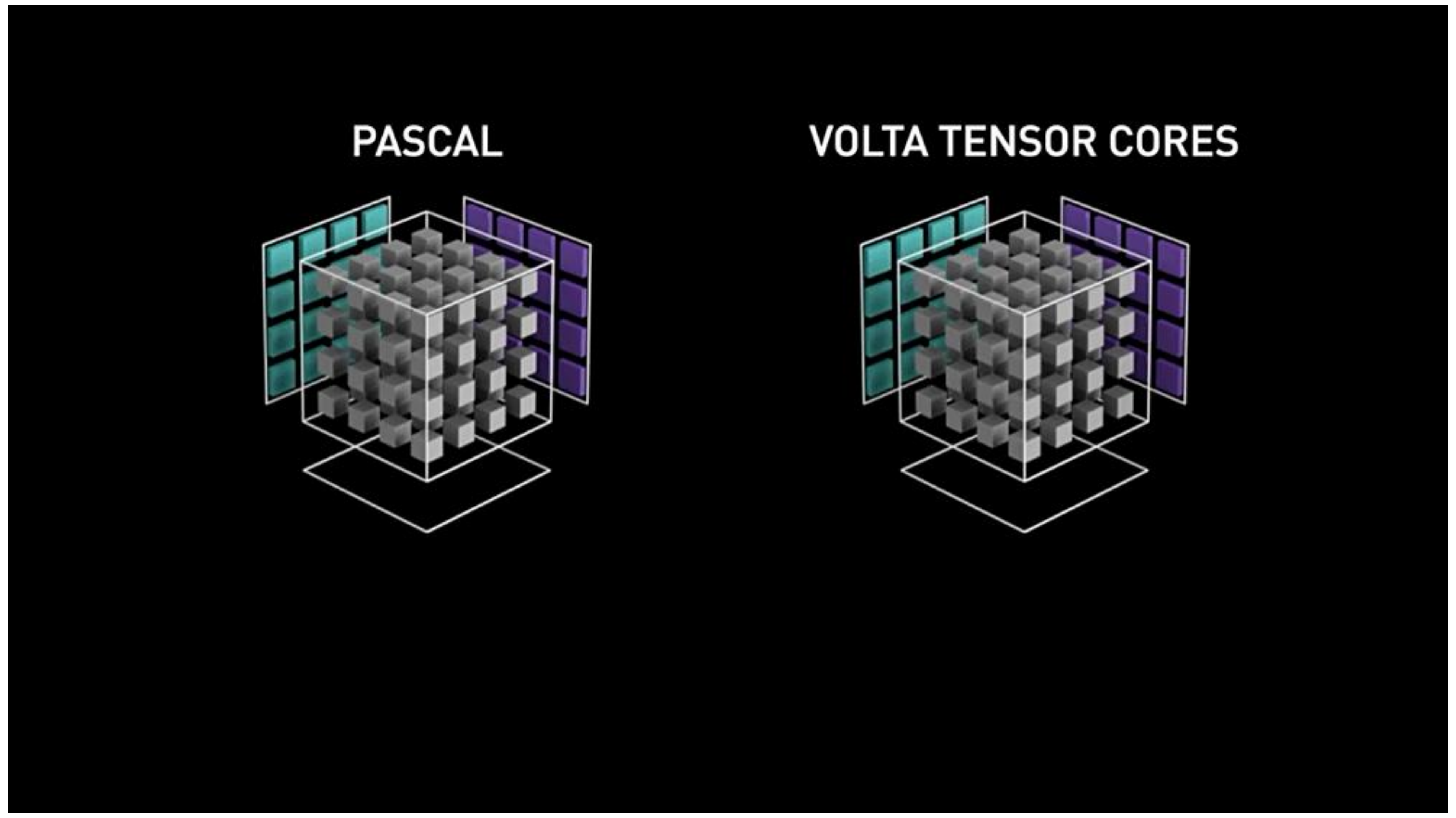


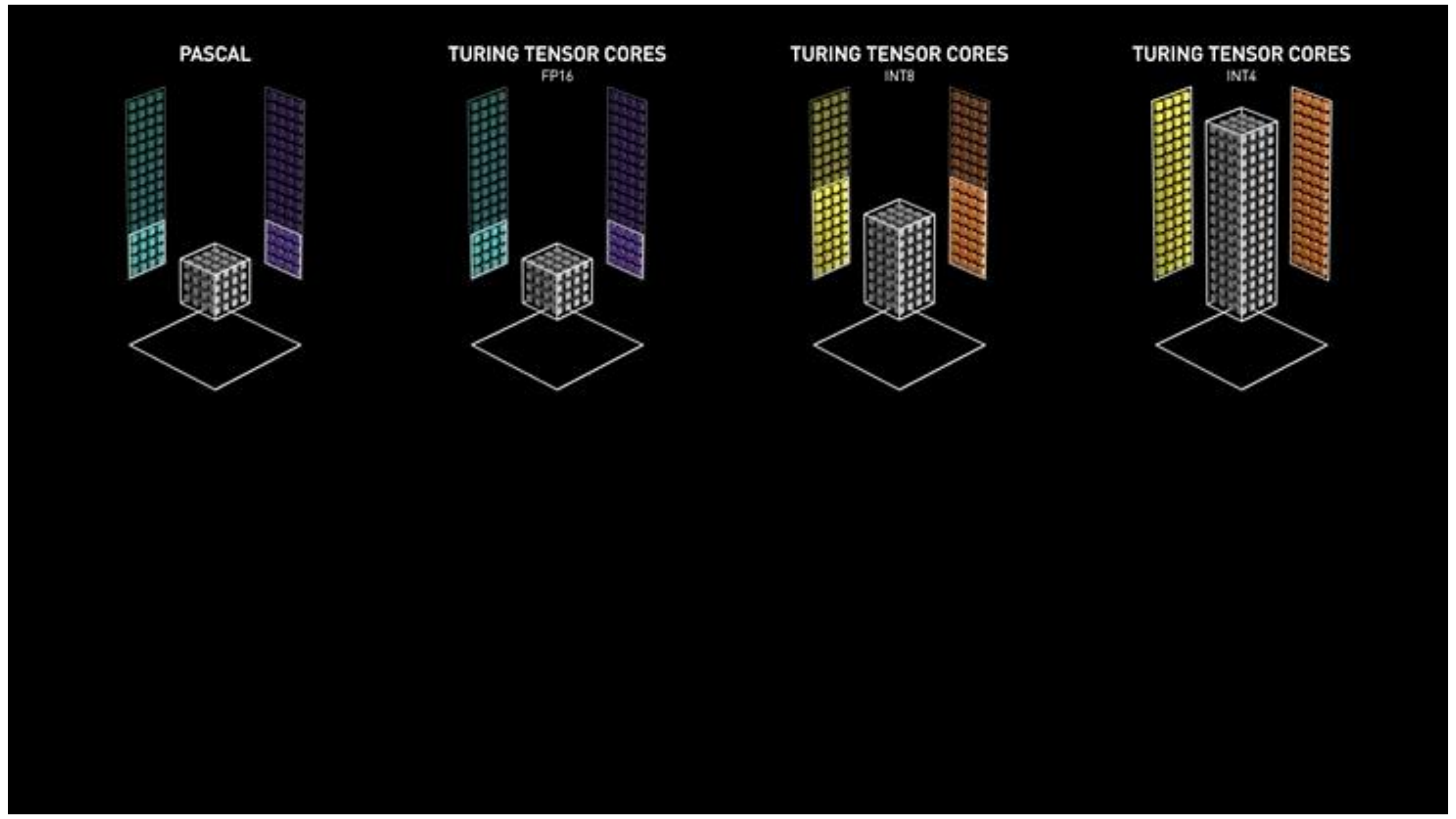
$$Z = W * X + b$$

Specific

Introduction

Hardware basement of mixed precision—TC cores





Difference between CUDA && TC more performant and energy-efficient

Introduction

Hardware basement of mixed precision—TC cores

TC candidate

- **the operation is either matrix multiplication or convolution using FP16**
- the input tensors of the operation satisfy the shape requirements.

	Hopper	Ampere	Turing	Volta
Supported Tensor core accuracy	FP64, TF32, bfloat16, FP16, FP8, INT8	FP64, TF32, bfloat16, FP16, INT8, INT4, INT1	FP16, INT8, INT4, INT1	FP16
Supported CUDA* core accuracy	FP64, FP32, FP16, bfloat16, INT8	FP64, FP32, FP16, bfloat16, INT8	FP64, FP32, FP16, INT8	FP64, FP32, FP16, INT8

Observation and Motivation

Impact of precision, TC, casting cost, input size on operation performance

Table 1: Performance comparison of some of the representative operations in NN training.

NN Operations	Input Data Size	FP16 Exe. Time (ms)	FP16+Cast Exe. Time (ms)	FP32 Exe. Time (ms)	Using TC
MatMul	(2048, 8, 8, 1024)	0.312	0.353	0.323	yes
	(64, 1001, 1001, 2048)	0.412	0.524	0.427	no
	(2048, 1024, 1024, 1024)	0.414	0.584	0.888	yes
Conv2D	(64, 35, 35, 48)	2.707	2.795	3.664	yes
	(64, 147, 147, 32)	28.965	29.249	29.487	no
	(64, 299, 299, 3)	57.879	58.944	60.098	no
Conv2DBackpropFilter	(64, 299, 299, 3)	8.690	9.773	10.246	no
	(64, 149, 149, 32)	6.013	7.988	7.011	no
	(64, 35, 35, 192)	0.786	0.948	0.871	yes
Conv2DBackpropInput	(64, 37, 37, 96)	3.954	4.049	6.943	yes
	(64, 149, 149, 32)	15.561	16.828	15.696	no
	(64, 35, 35, 192)	5.234	5.939	10.060	yes
BiasAdd	(64, 1001, 1001)	0.252	0.317	0.255	no
	(64, 4096, 4096)	0.294	0.323	0.298	no
	(64, 9216, 9216)	0.299	0.342	0.311	no
MaxPool	(64, 35, 35, 288)	1.849	2.072	1.793	no
	(64, 17, 17, 768)	1.399	1.542	1.402	no
	(64, 8, 8, 2048)	0.825	1.128	0.981	no

- Performance variance with different data precisions
- Impact of input data size on performance gains from FP16
- Impact of casting cost

Observation and Motivation

Observation :

- Using FP16 leads to **slightly better** performance than using FP32. **Using TC** for FP16 **magnifies the performance benefit** of FP16
- The performance gain of using FP16 varies largely across **input data sizes**
- **The cast operation introduces non-negligible overhead**. Considering the casting cost, it is **not always performance-profitable** to convert FP32 to FP16 regardless of using TC or not

Motivatoin :

Improve upon tree dimensions →

- Input data size
- **Casting cost**
- **Usage of TC**

Campo Overview

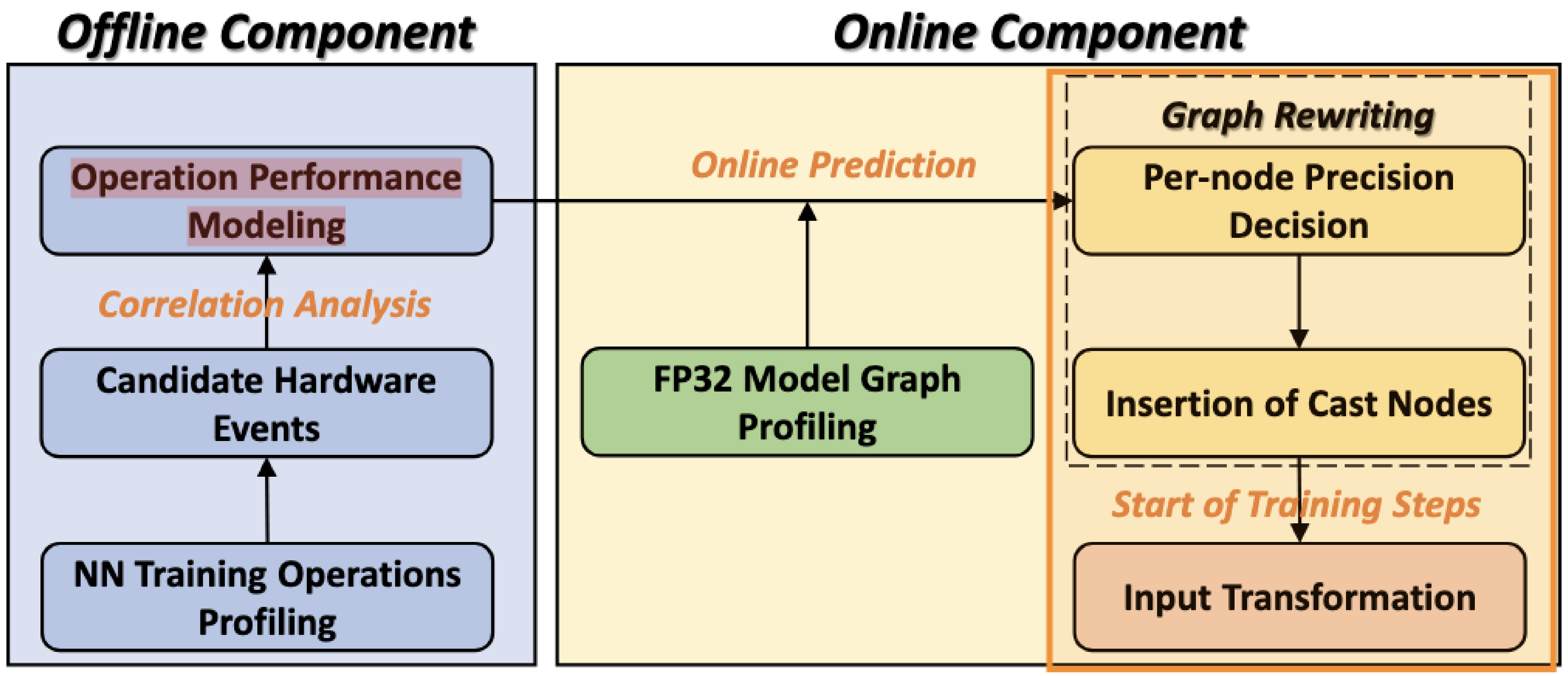


Figure 3: Overview of Campo.

Composition

- Offline component

  performance predict model
- Online component

  model graph profiling

  graph rewriting

Campo Overview

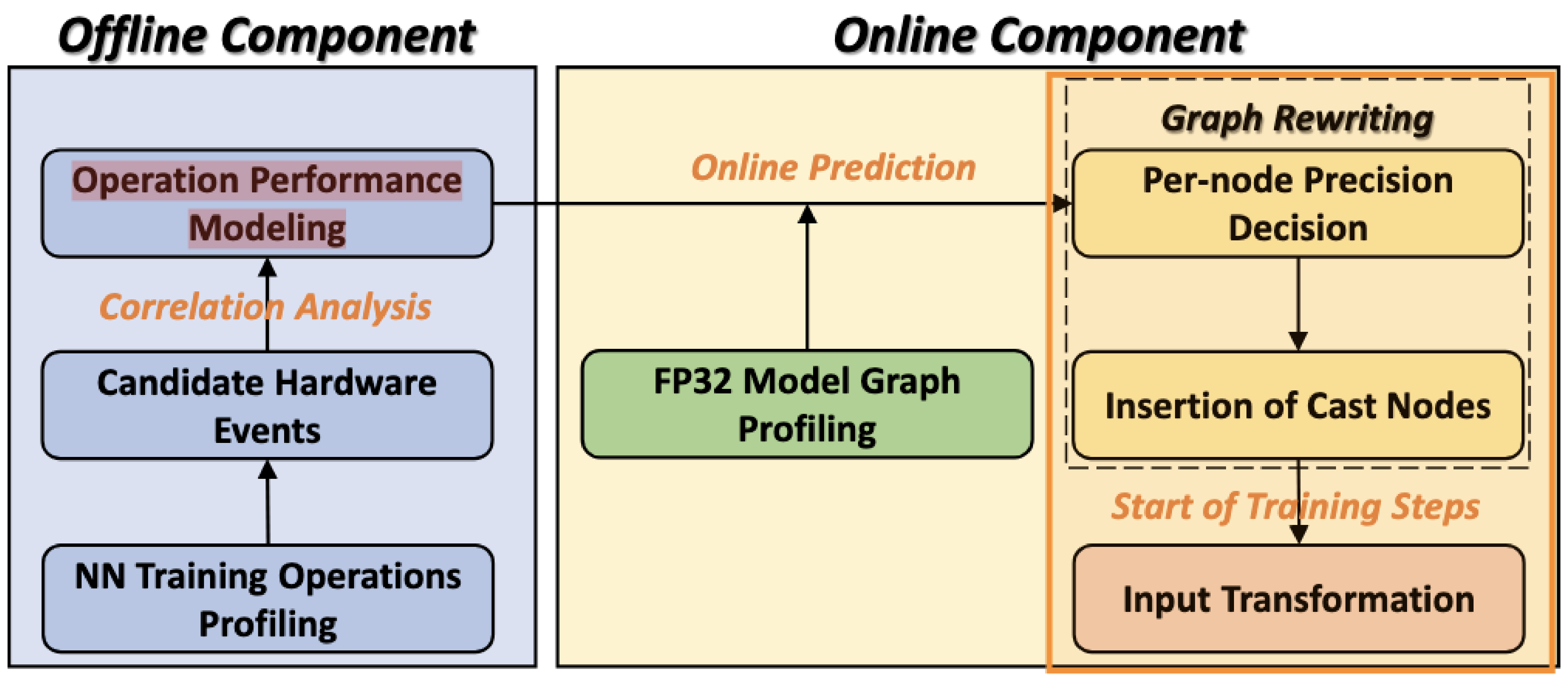


Figure 3: Overview of Campo.

Composition

- **Offline component**

  performance predict model

  decide precision for a given **operation with input data size**
- Online component

  model graph profiling

  graph rewriting

Campo Overview

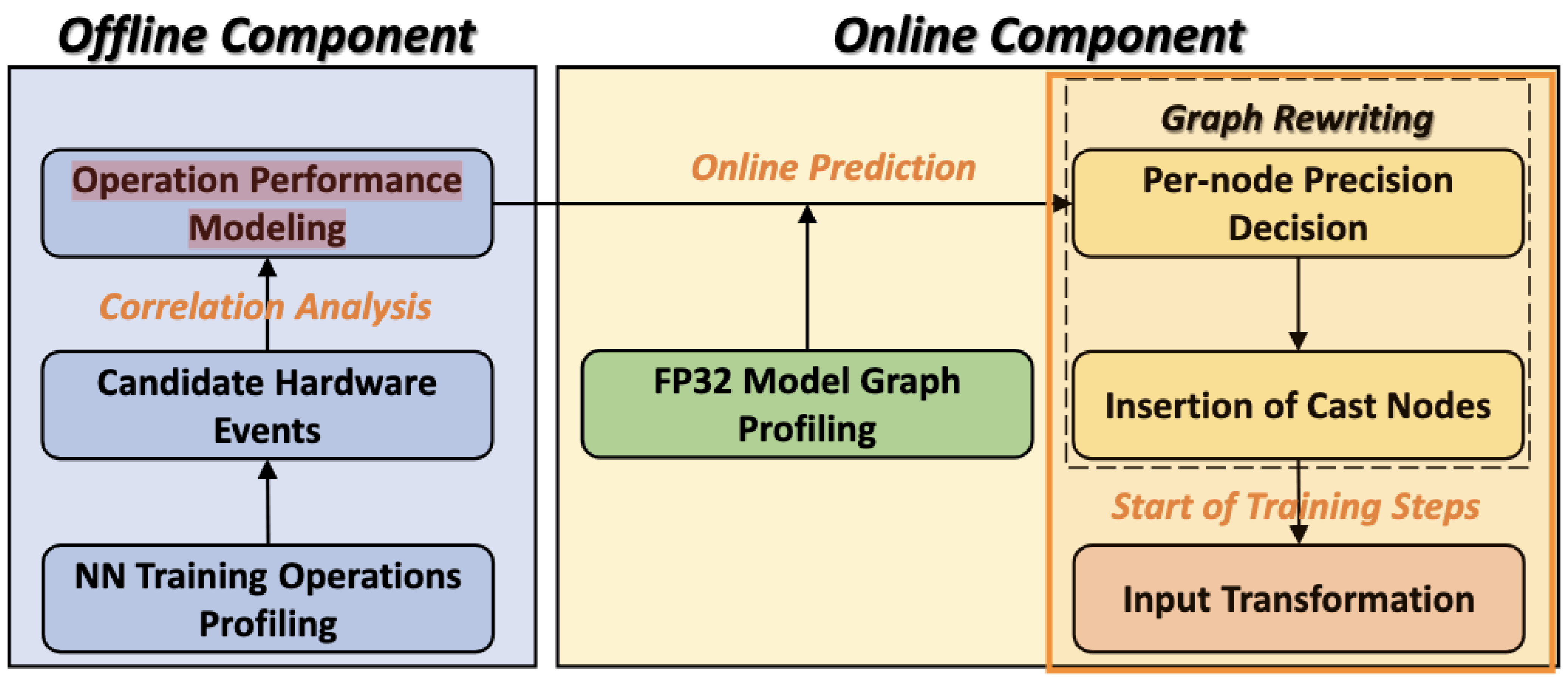


Figure 3: Overview of Campo.

Composition

- Offline component

  performance predict model
- **Online component**

  model graph profiling

  profile operation **in FP32** to **provide Information for performance model**

  graph rewriting

Campo Overview

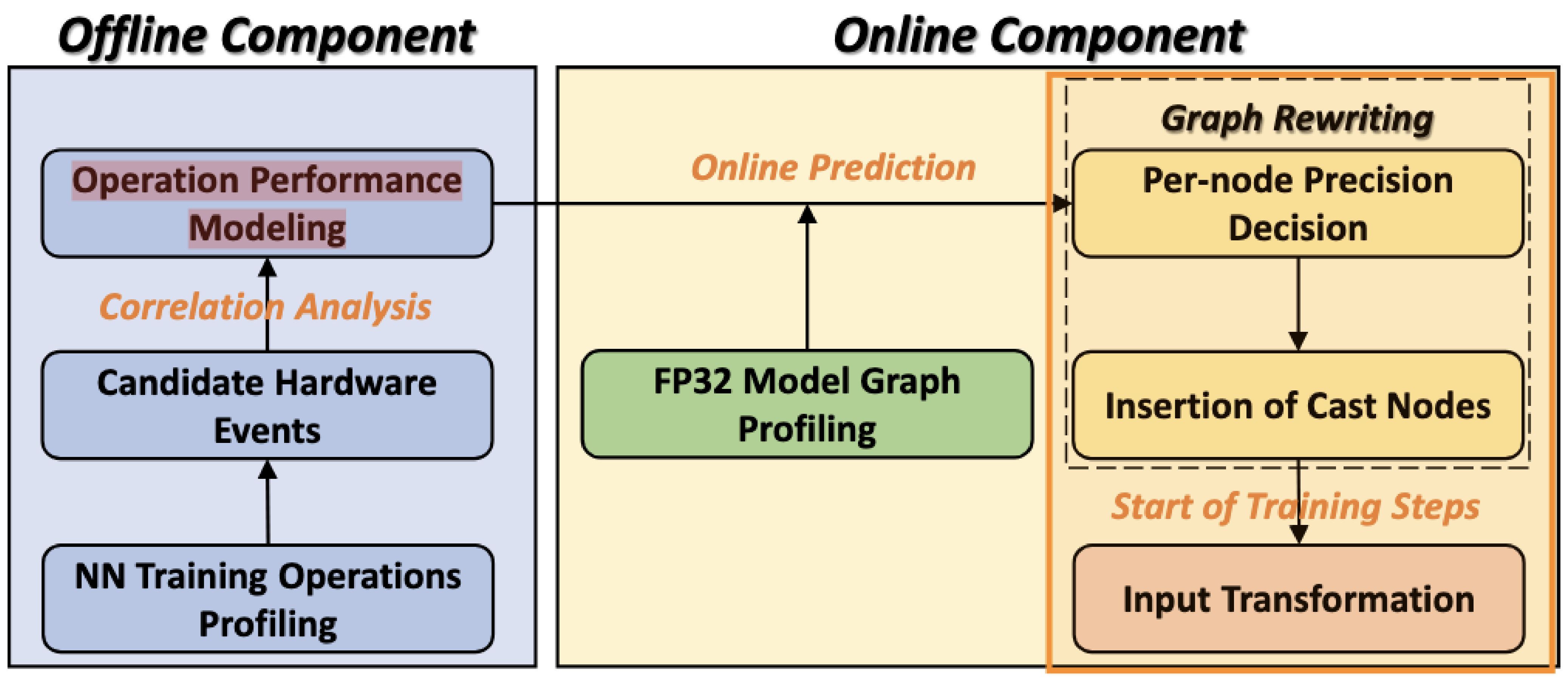


Figure 3: Overview of Campo.

Composition

- Offline component

  performance predict model
- **Online component**

  model graph profiling

  graph rewriting
  - **graph traverse**
  - **input transformation**

Campo Overview

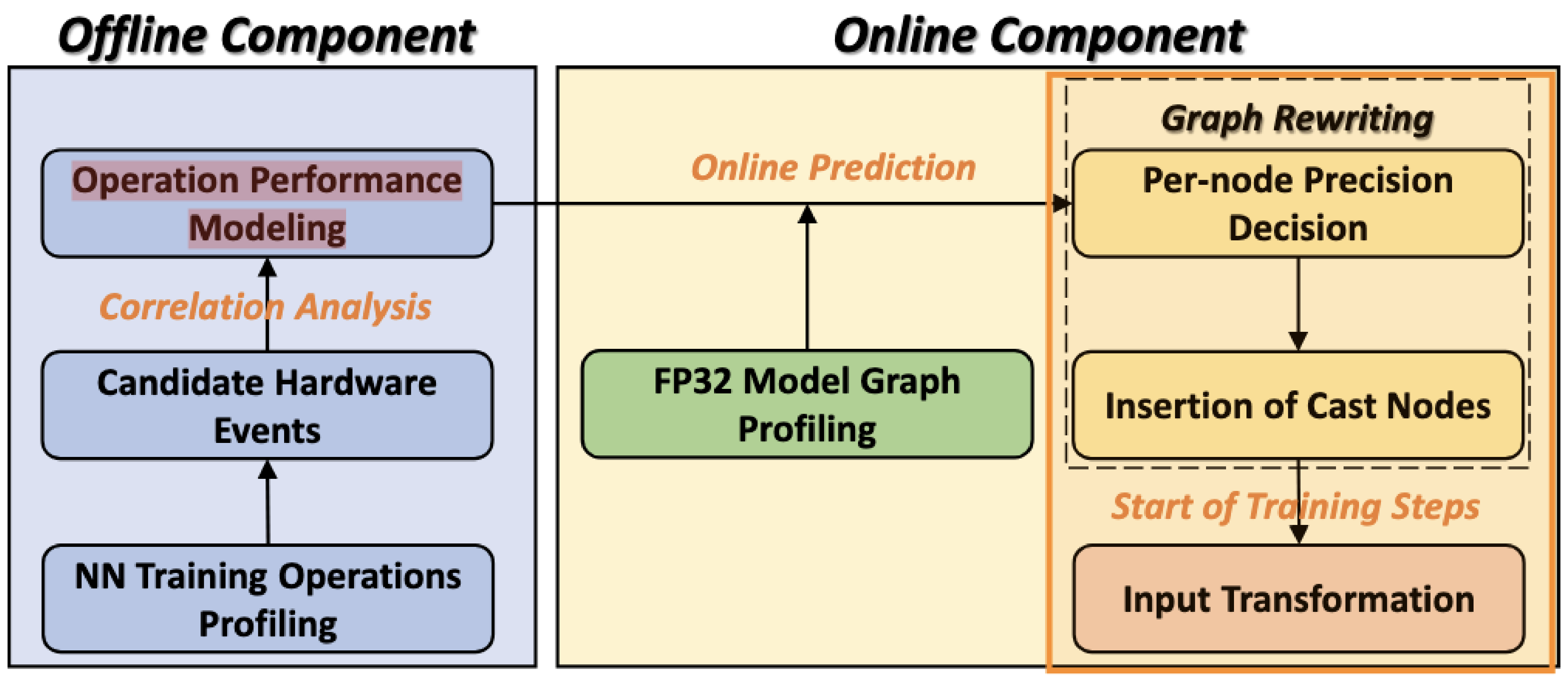


Figure 3: Overview of Campo.

Composition

- Offline component
- **Online component**

  model graph profiling

  graph rewriting

  - **graph traverse**

    1. make **four passes on dataflow graph**

    2. make **final decision** on **precision assignment of each operation node**

  - input transformation

Campo Overview

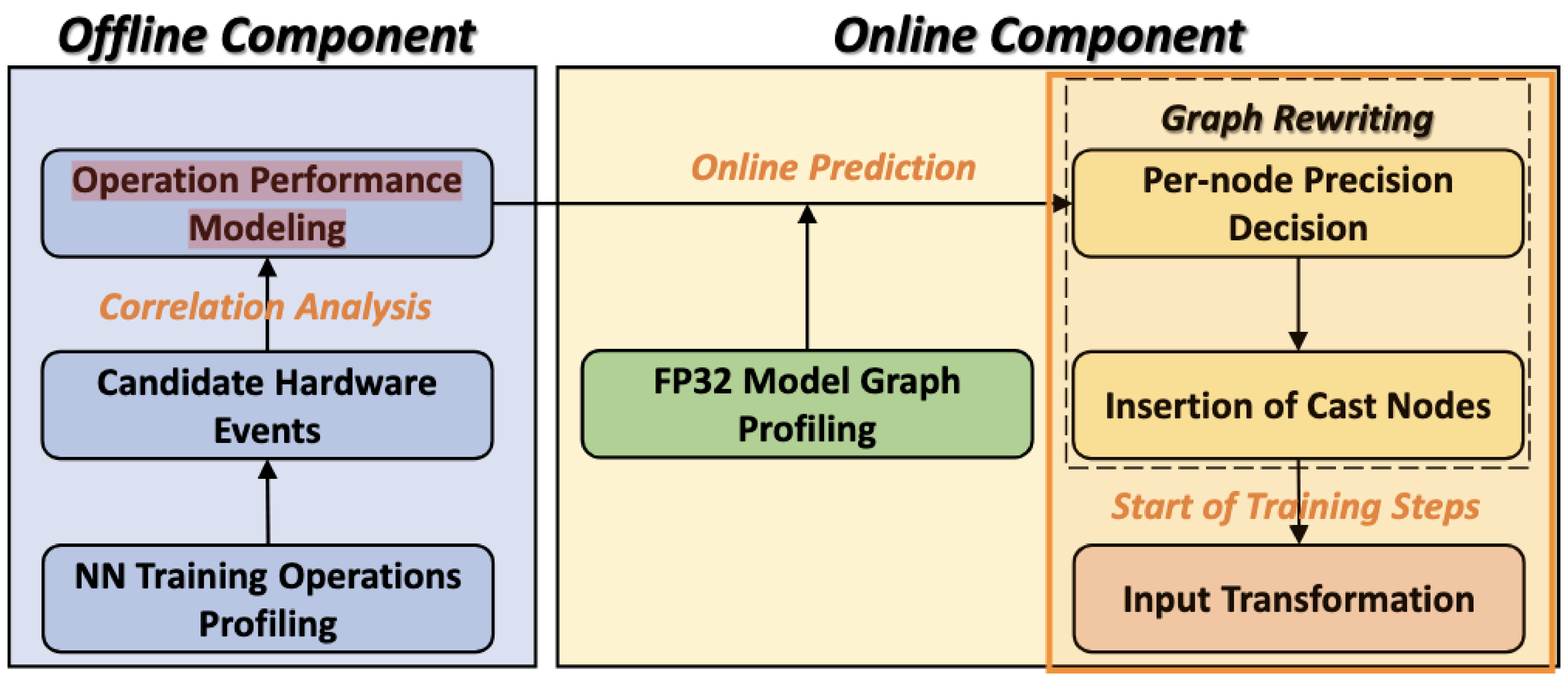


Figure 3: Overview of Campo.

Composition

- Offline component
- **Online component**

  model graph profiling

  graph rewriting

  - graph traverse
  - **input transformation**

    pad the input tensors to make TC candidates **meet the requirement of TC** on input shape

Campo Overview

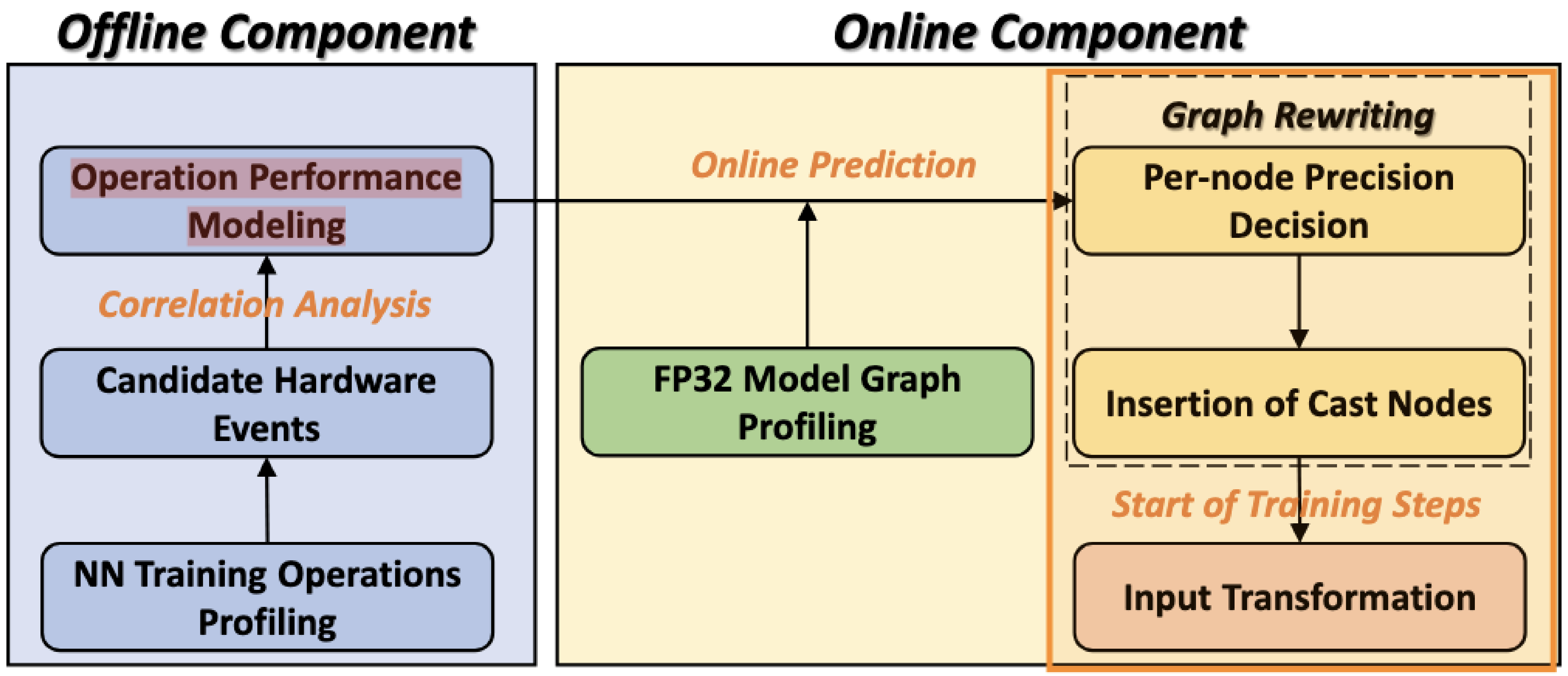


Figure 3: Overview of Campo.

Composition

- Offline component

  performance predict model
- Online component

  model graph profiling

  graph rewriting

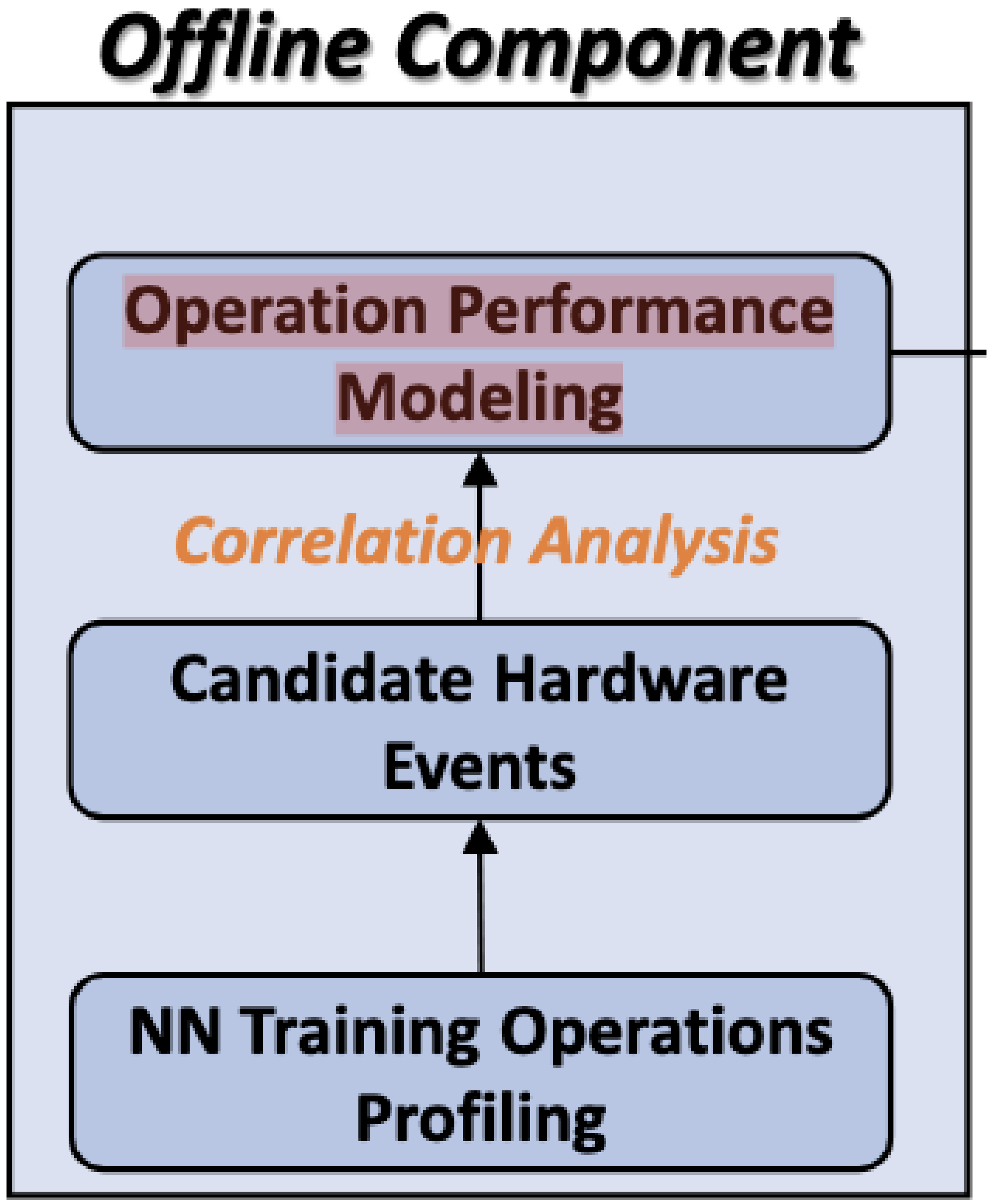
Operation-specific performance modeling

Performance model’s work:

Determine whether **low or full precision** should be employed for **a single given operation with a given input data size**

Benefit:

More **efficient and accurate** method for **offline prediction**

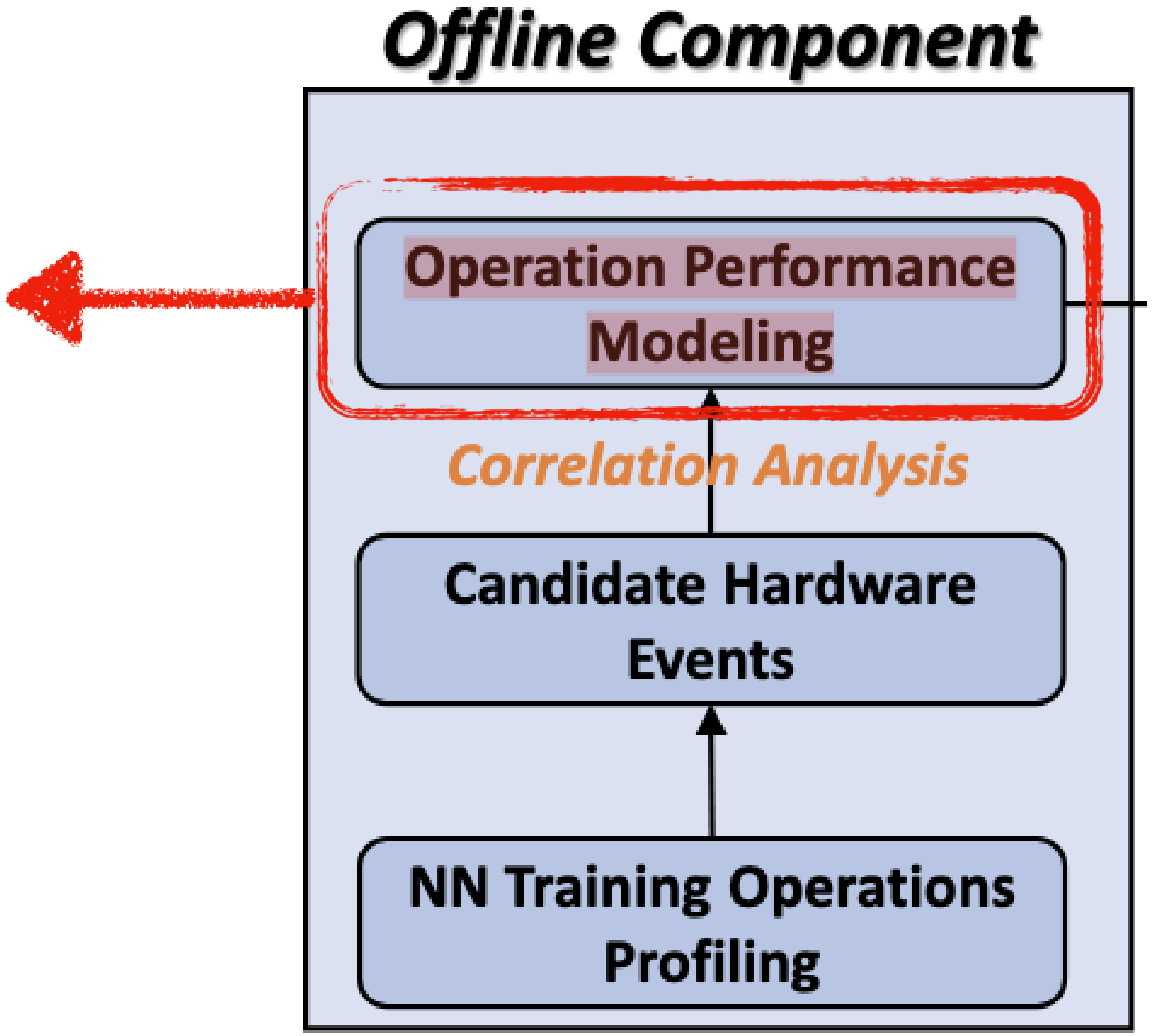


Operation-specific performance modeling

Predictions

1. **casting cost** based on the input data size

2. **execution time** of the operation with low precision



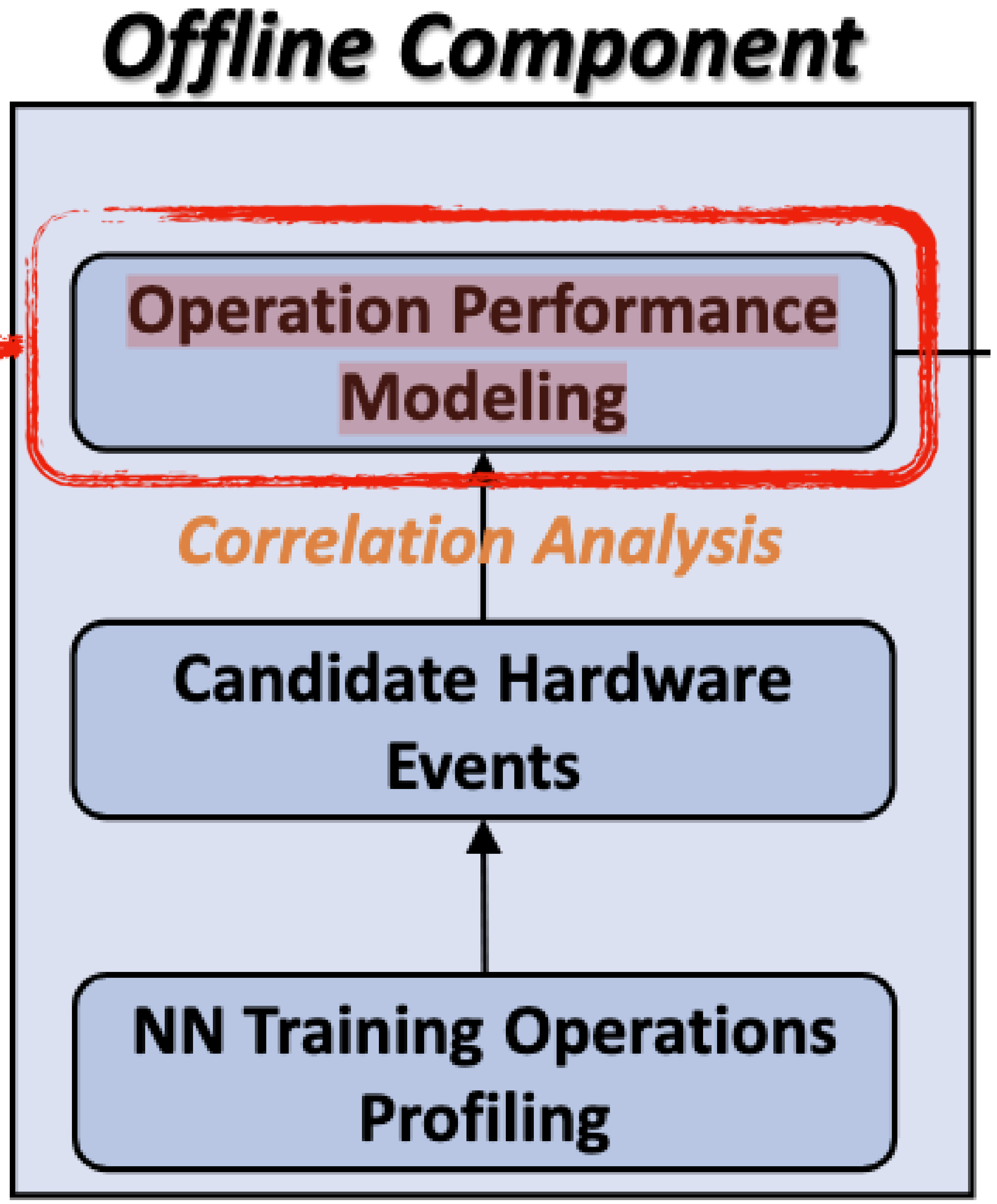
Operation-specific performance modeling

Casting cost based on the input data size

1. **casting cost** based on the input data size

1. time to initialize the cast operation node

2. time to do the conversion



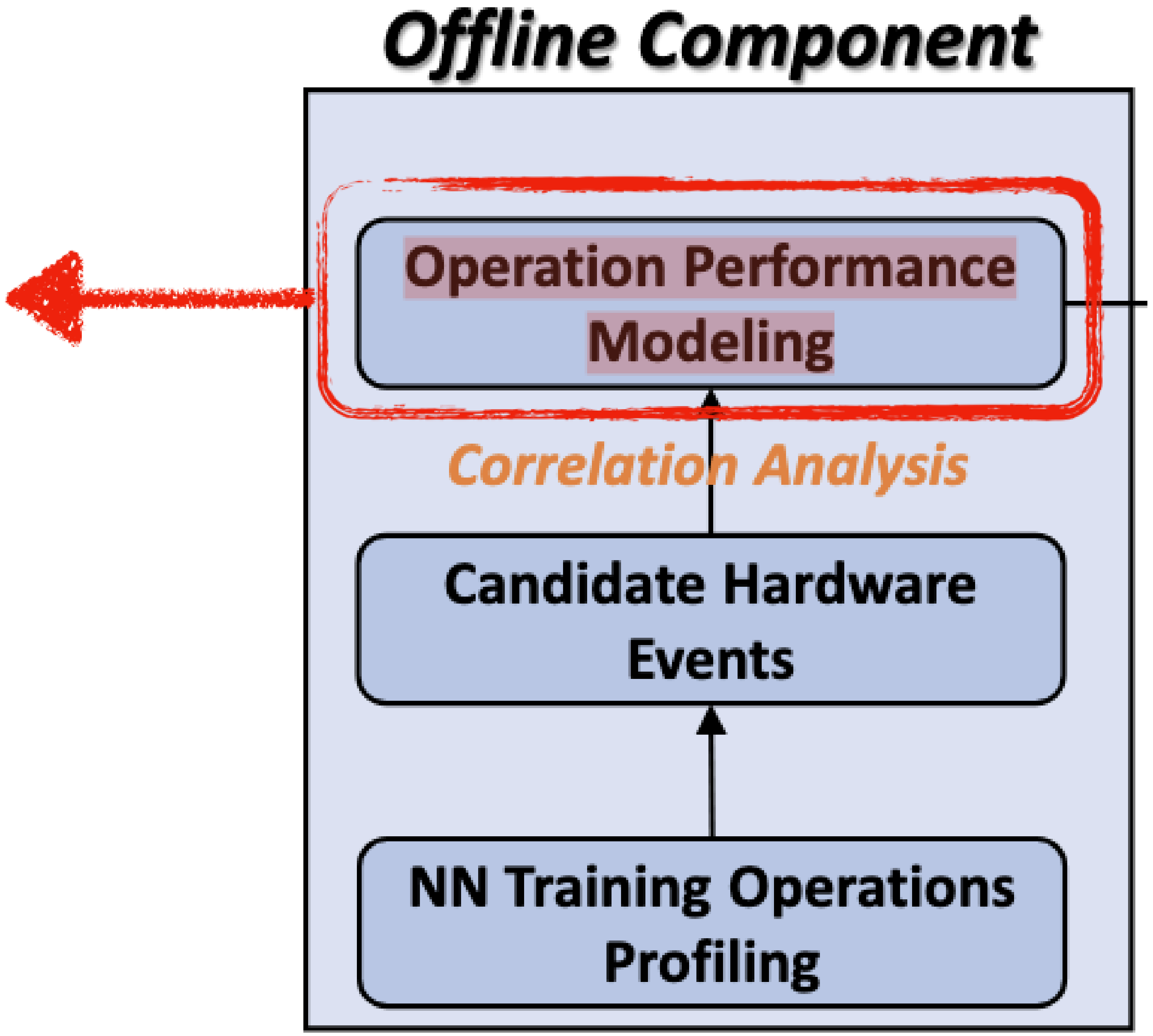
Operation-specific performance modeling

Casting cost based on the input data size

1. initialize the cast operation node
2. do the conversion

$$casting\_cost = \boxed{r * tensor\_size} + C_I$$

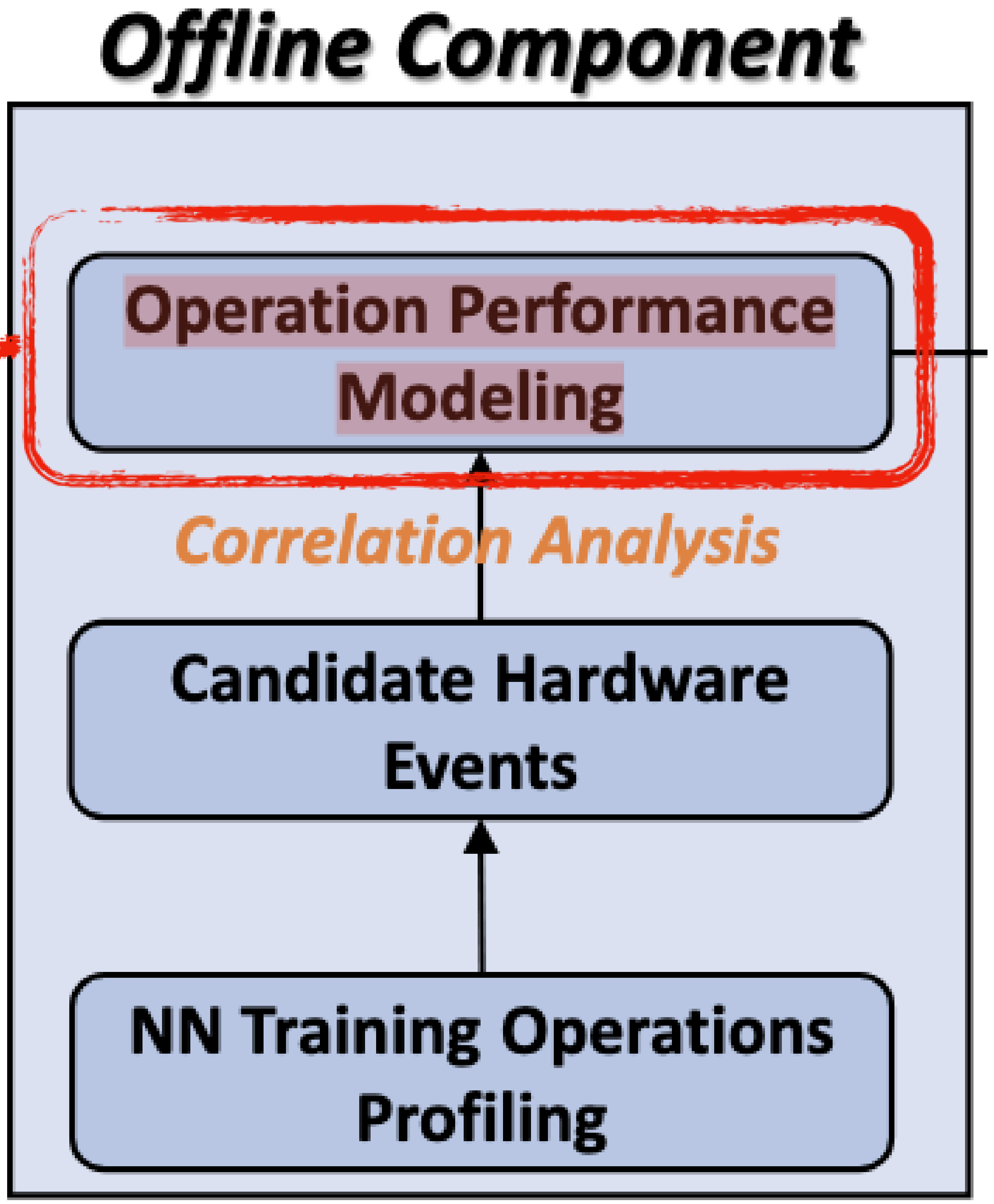
r & CI get by linear regression



Operation-specific performance modeling

Execution time of the operation with low precision

2. **execution time** of the operation with low precision



1. ~~**general** model~~ for all the operations to predict performance ← ~~1~~. Low prediction **accuracy** & long **execution time**(on average)

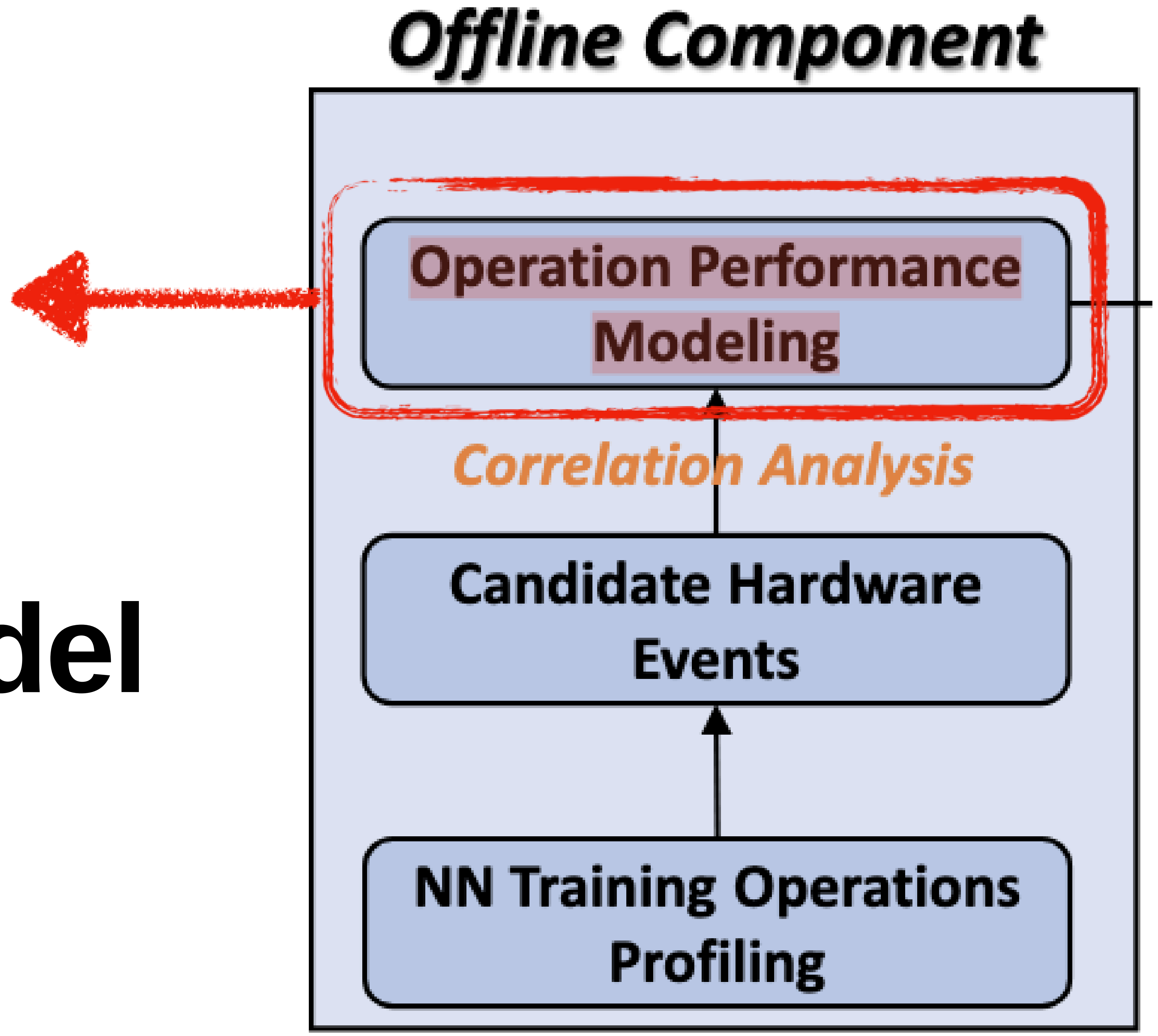
2. ~~**dynamic** profiling~~ via three training iterations of the NN model ← ~~2~~. increase in **training time**

Operation-specific performance modeling

Execution time of the operation with low precision

1. ~~general model~~ for all the operations to predict performance

2. ~~dynamic profiling~~ via three training iterations of the NN model



Each op -> specific model

Offline strategy

Architecture-specific

Operation-specific performance modeling

Model intuition

Intuition

The performance of an operation in FP32 and FP16 has **correlation**

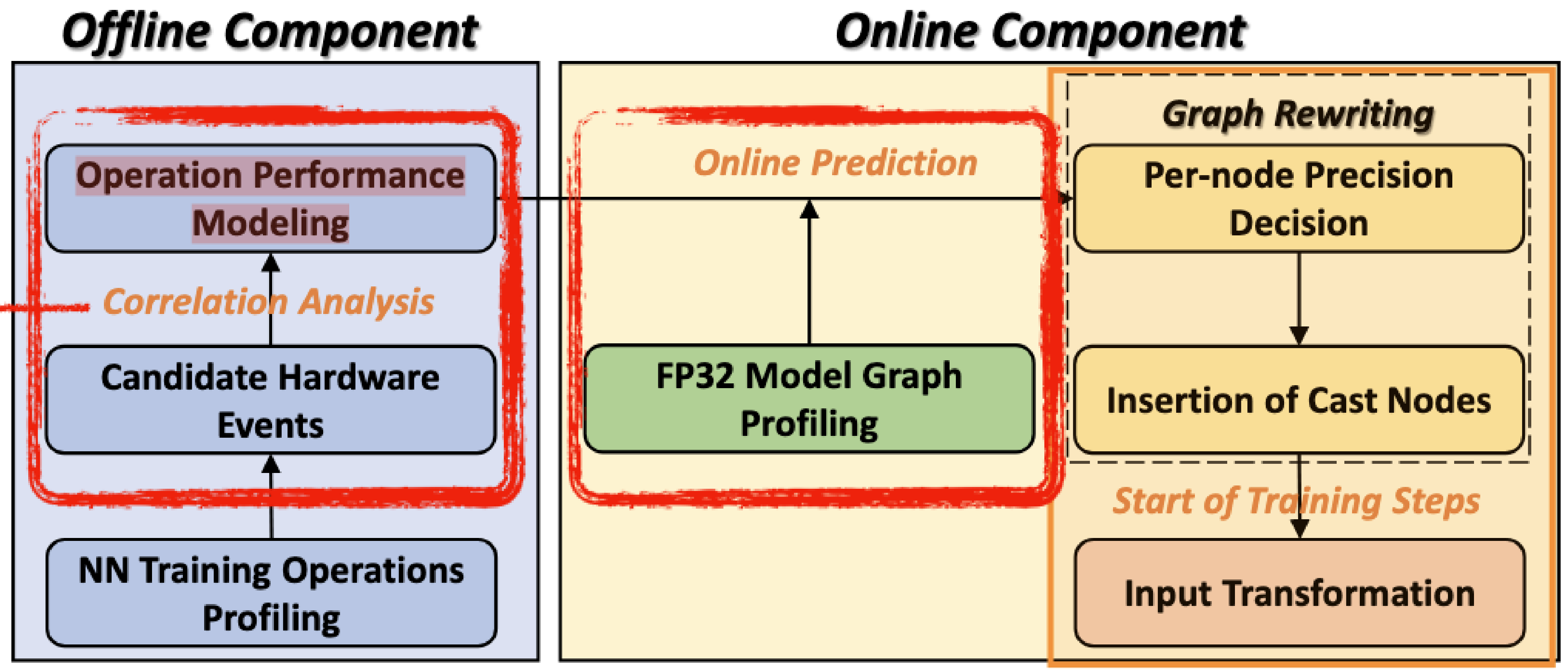


Figure 3: Overview of Campo.

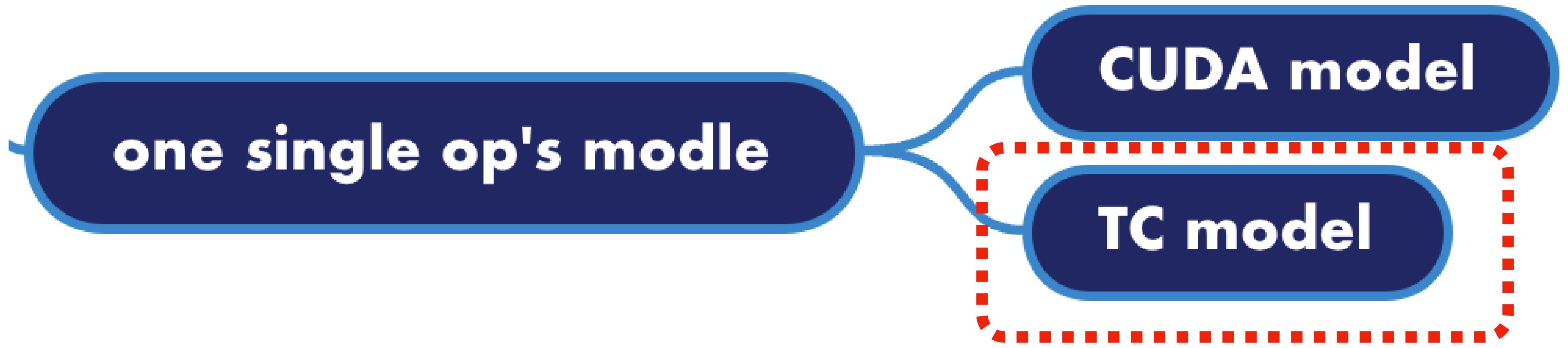
using
execution time in FP32
handful of performance-critical events measured in FP32

1.
2. a

Operation-specific performance modeling

Performance modeling

Number of models



Only when it’s TC candidate

$$OP_{LP} = OP_{FP32} \cdot (\sum_{i=1}^{N} w_i \cdot PC_i) + \sigma$$

Collected by graph profiling

Events with correlation

Step 1. Correlation analysis

Operation-specific performance modeling

Performance modeling

$$OP_{LP} = OP_{FP32} \cdot \left(\sum_{i=1}^{N} w_i \cdot PC_i\right) + \sigma$$

Spearman's rank correlation coefficient

$$\rho = 1 - \frac{n \sum d_i^2}{n(n^2 - 1)}$$

$$d_i = rank_1 - rank_2$$

Threshold

$|\rho| > 0.75$

Events with correlation

Step 1. Correlation analysis

Operation-specific performance modeling

Performance modeling

$$OP_{LP} = OP_{FP32} \cdot \left(\sum_{i=1}^{N} w_i \cdot PC_i\right) + \sigma$$

Spearman's rank correlation coefficient

$$\rho = 1 - \frac{n \sum \boxed{d_i^2}}{n(n^2 - 1)}$$

$$d_i = rank_1 - rank_2$$

Threshold

$|\rho| > 0.75$

Events with correlation

Step 1. Correlation analysis

Train to find the values of coefficients (offline)

Step 2. Linear regression

Building performance models for 143 operations takes about **112.5 hours**

Runtime graph rewriting

Graph rewriting's work:

1. **assign precision for each operation**

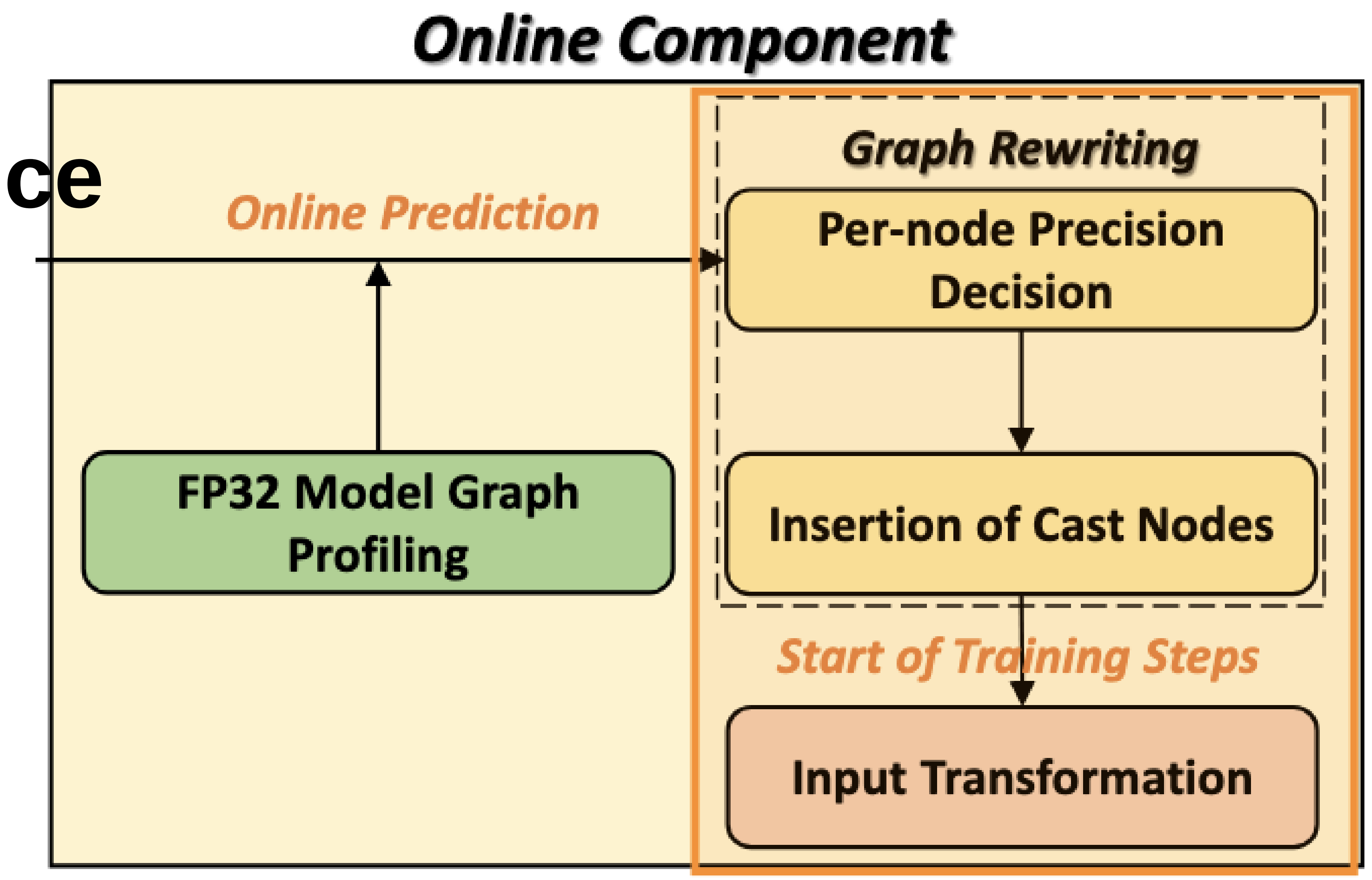
2. which operations to be **converted together** to **reduce** the number of **cast operation nodes**

Benefit & Goal:

1. minimizing the **training time**

2. minimizing the **casting cost**

3. no adverse impact on the **numerical safety** (compared with the traditional mixed-precision training)



Runtime graph rewriting—graph profiling

Graph profiling run on CUDA cores in FP32

Works:

1. Provide basic information for **performance modeling**

2. Triggered right after the first few training steps

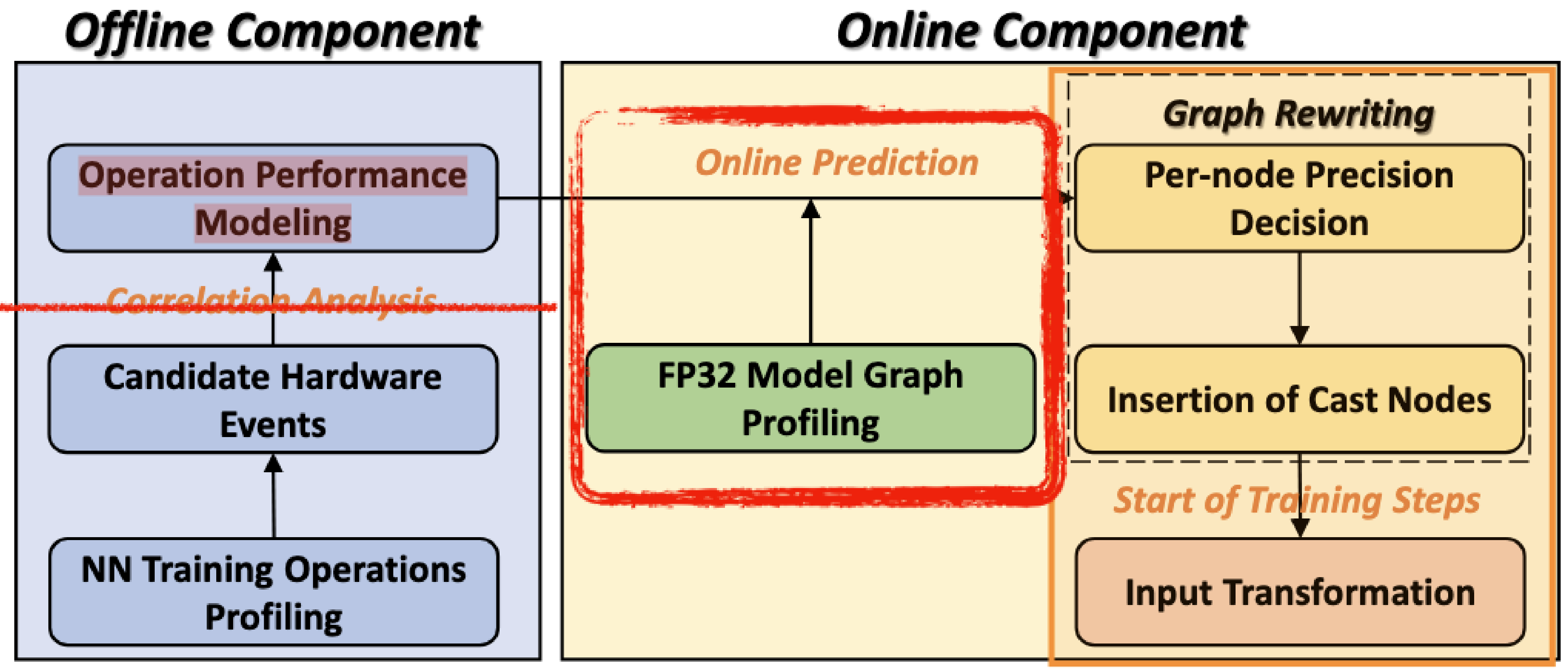


Figure 3: Overview of Campo.

Input for performance model

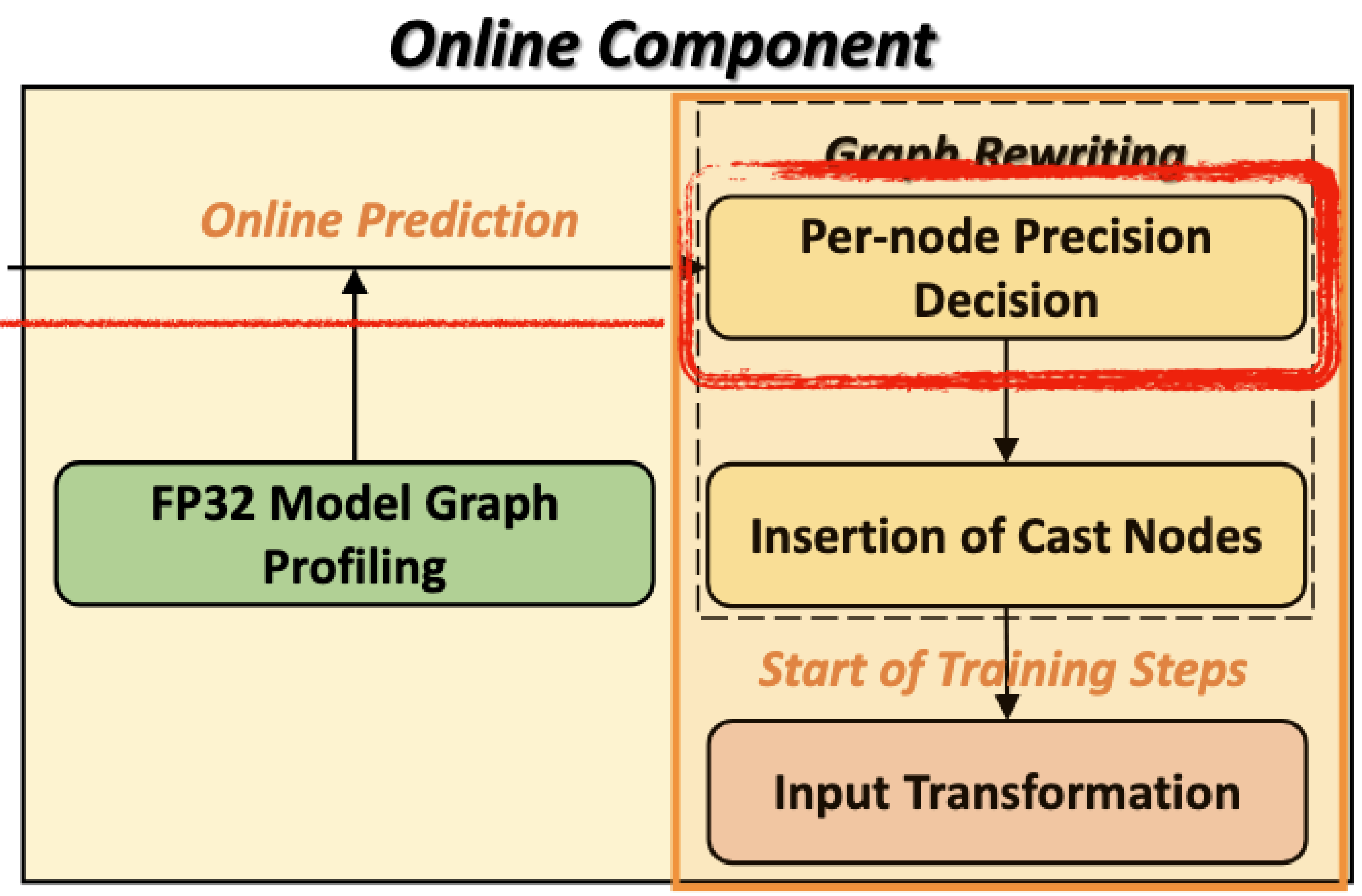
Runtime graph rewriting—graph traverse

Four-round traverse on dataflow graph

Works:

1. Make **final decisions on precision assignments** via **four-round traverse**

2. Sort nodes into **allow_nodes, deny_nodes**

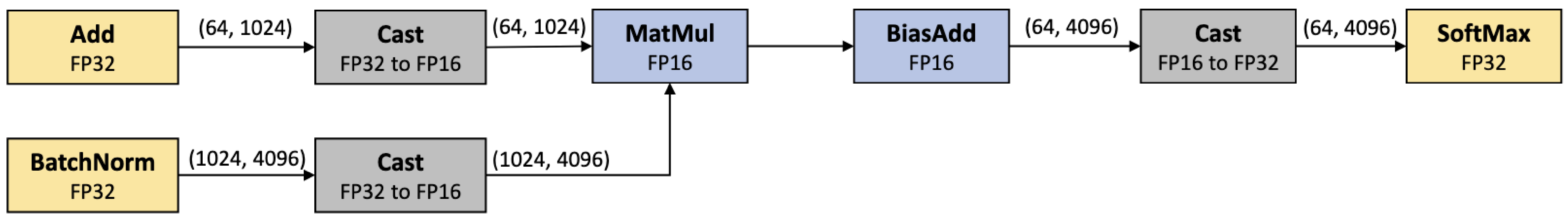


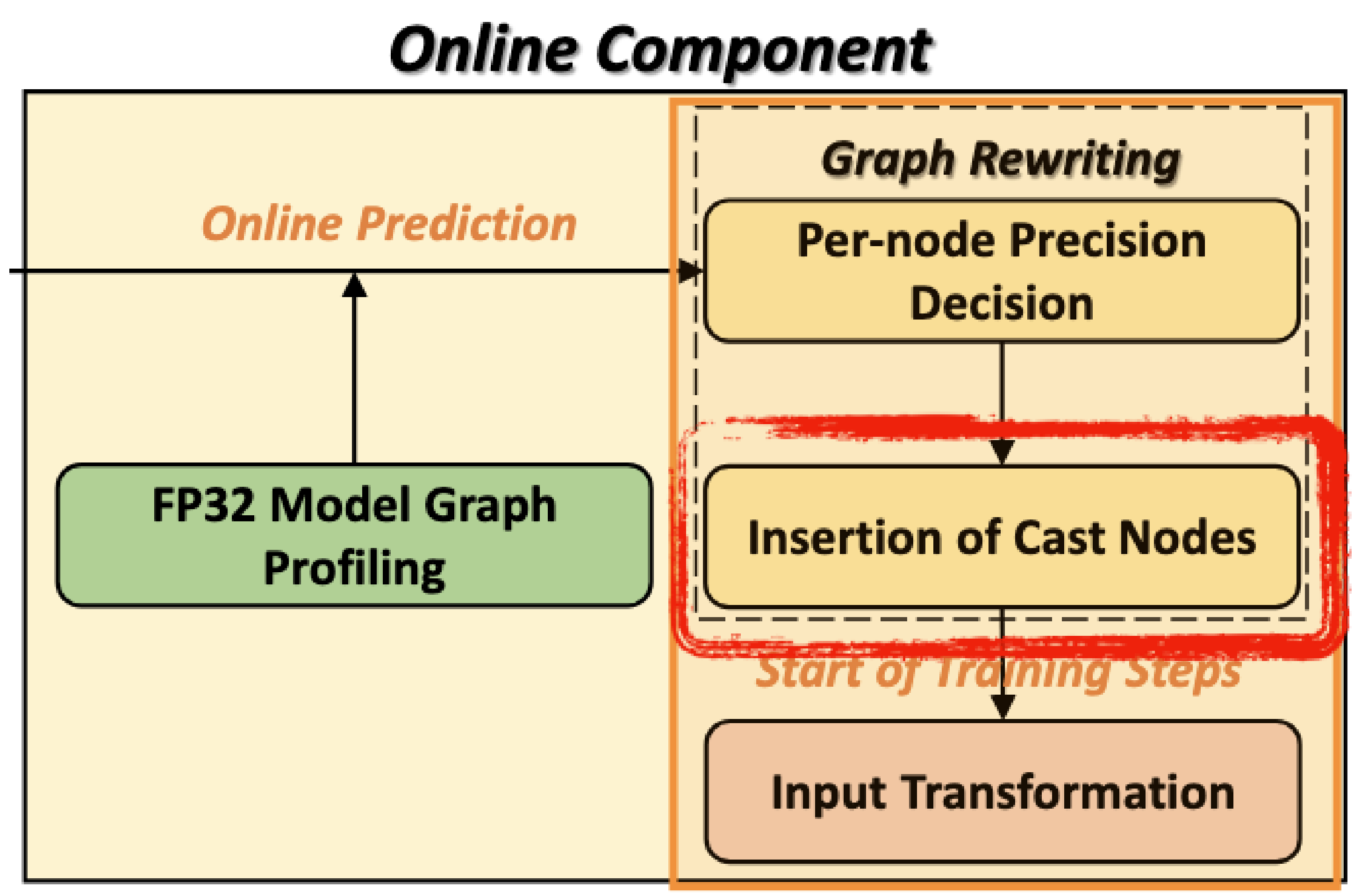
Runtime graph rewriting—graph traverse

Four-round traverse on dataflow graph

Traverse

Runtime graph rewriting—insert casting node





using the API ChangeTypeAttrsAndAddCasts **provided by TensorFlow**

FP16 -> FP32 , vice versa

Usage of tensor cores

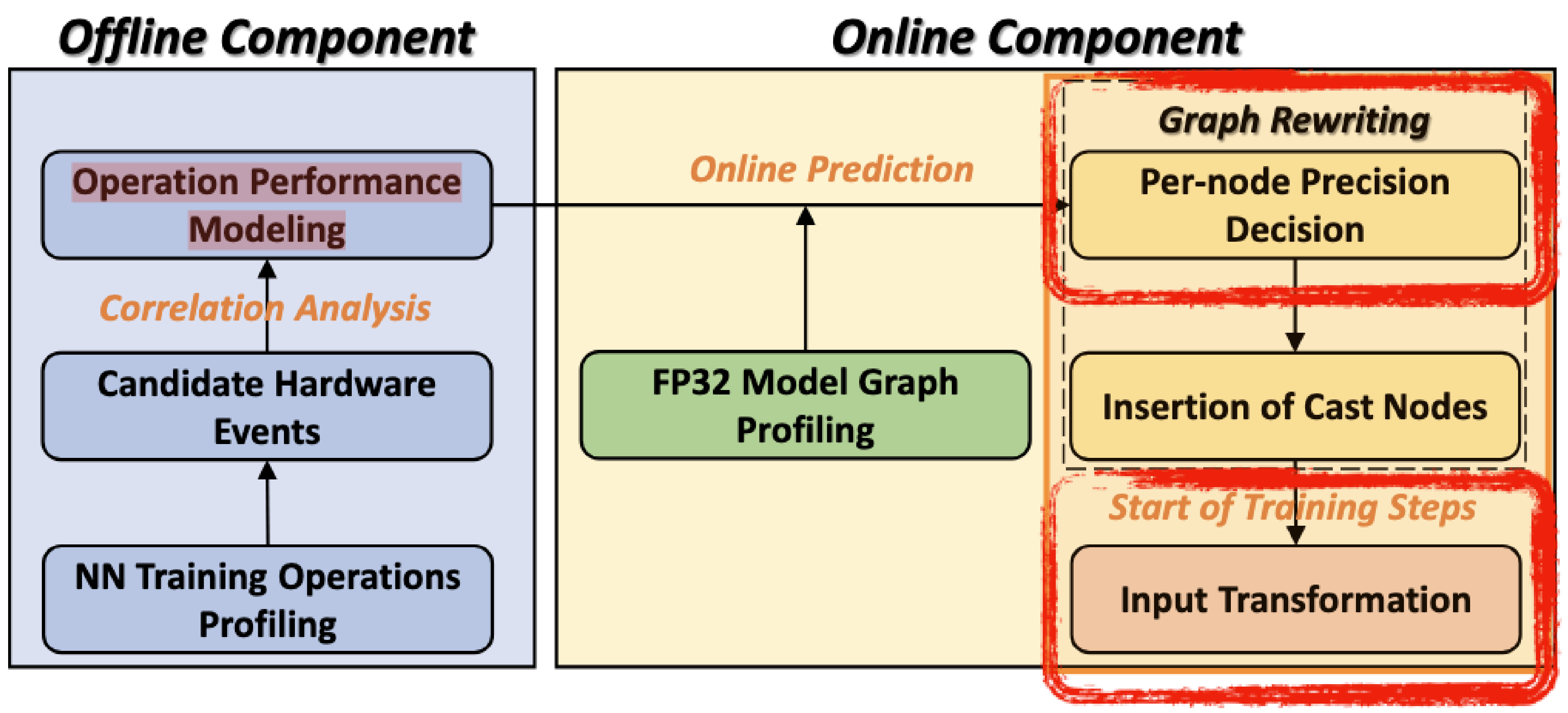


Figure 3: Overview of Campo.

Efforts to make full use of TC:

1.Take **TC&CUDA's difference** into account when **assignment**

2.Pad the unfit input to **suit TC's input requirement**

Benefit:

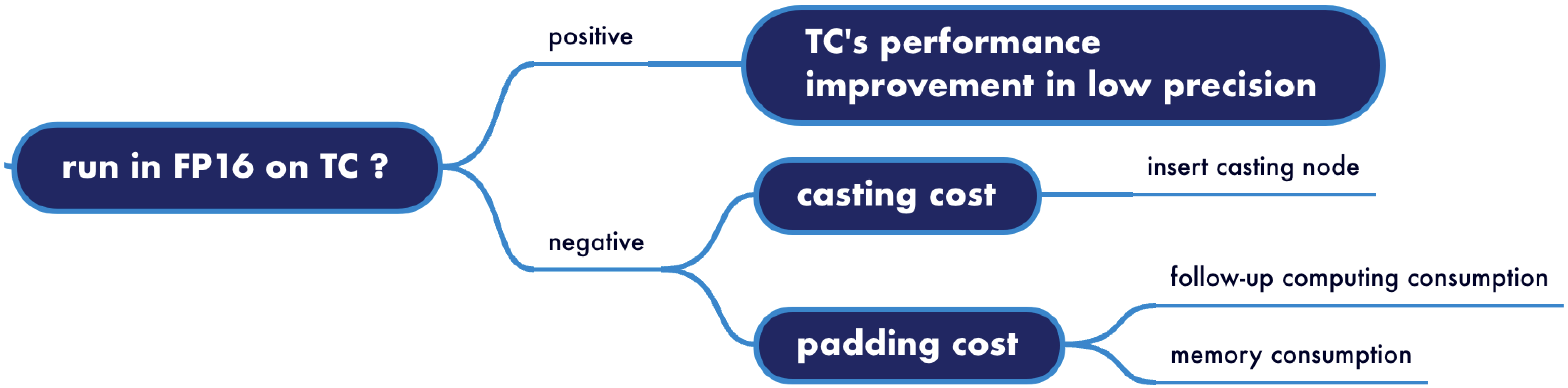
1.Take advantage of **TC's superiority in low precision**

2.Make full use of **heterogeneous computing resources**

Usage of tensor cores

Overhead analysis

The usage of tensor cores is a **multi-dimensional** problem



Campo Overview

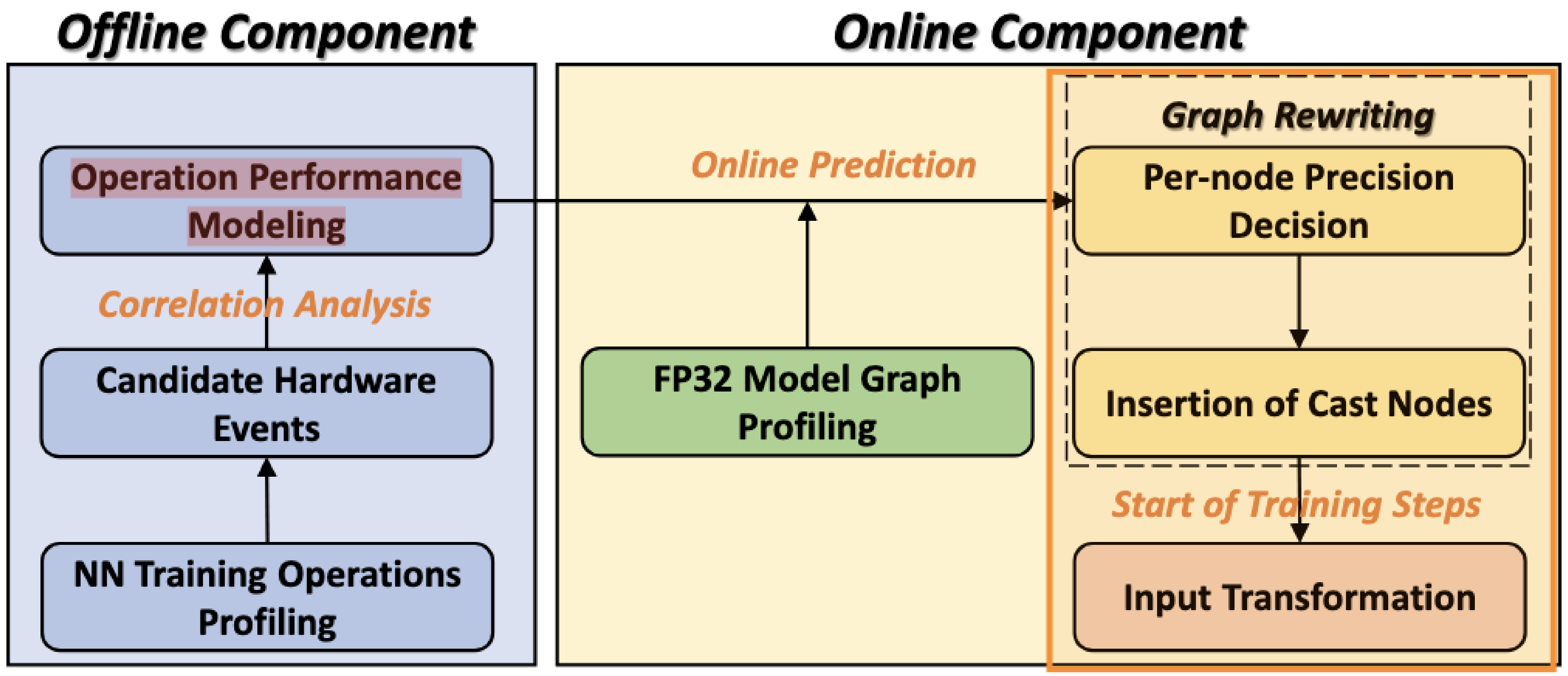


Figure 3: Overview of Campo.

Composition

- Offline component

  performance predict model
- Online component

  model graph profiling

  graph rewriting

Evaluation

Experimental setup

Table 2: Hardware configurations

CPU	Intel Xeon CPU E5-2648L v4@ 1.80GHz
Main Memory	64 GB DDR4
CPU Cores	2 sockets, 14 cores per socket
GPU	NVIDIA GeForce RTX 2080 Ti (Turing)
CUDA Cores	4352 CUDA cores (68 SMs, 1.54GHz)
Tensor Cores	544 tensor cores
L1 Cache	64 KB (per SM)
L2 Cache	5.767 MB
GPU Device Memory	11 GB GDDR6
GPU	NVIDIA Tesla V100 (Volta)
CUDA Cores	5376 CUDA cores (84 SMs, 1.53GHz)
Tensor Cores	672 tensor cores
L1 Cache	128 KB (per SM)
L2 Cache	6.144 MB
GPU Device Memory	32 GB HBM2

TensorFlow	**v1.15**
CUDA	**9.0**
cuDNN	**8.0**
Ubuntu	**18.04**

Hardware & Software

Evaluation

Training throughput

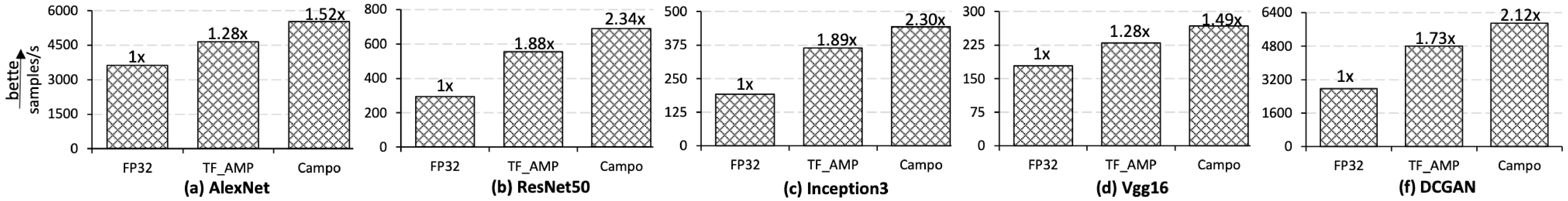


Figure 4: Training throughput with FP32, TF_AMP and Campo on RTX 2080 Ti.

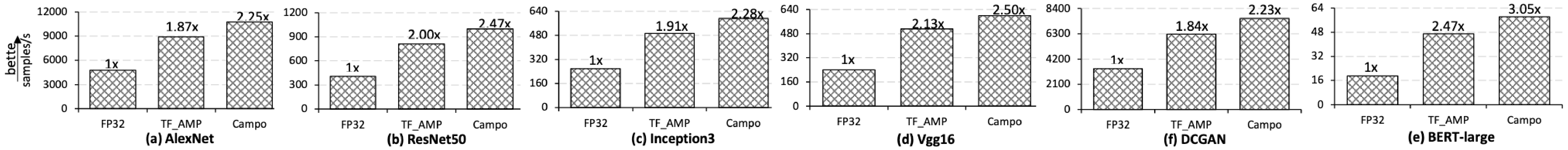


Figure 5: Training throughput with FP32, TF_AMP and Campo on V100.

Speedup reach 1.28x - 3.05x

Evaluation

Performance breakdown——Number of cast operation nodes

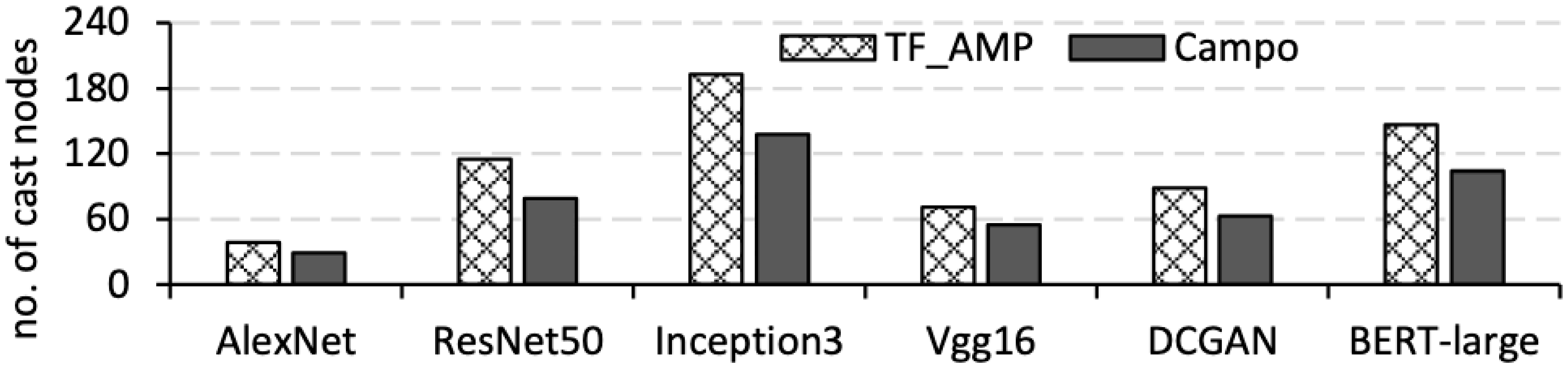


Figure 6: The number of cast nodes of NN models trained with TF_AMP and Campo, respectively.

27.7% less cast operation nodes on average **(up to 31.3%)**

compared with TF_AMP

Evaluation

Performance breakdown——TC utilization

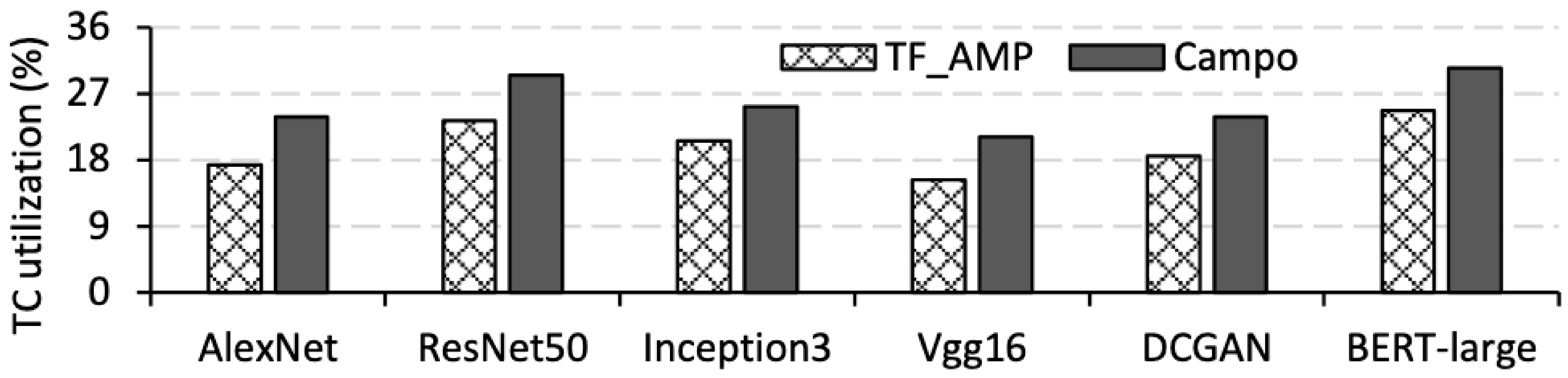


Figure 7: TC utilization of NN models trained with TF_AMP and Campo, respectively.

increases the utilization of TC by **29.4%** on average **(up to 37.9%)**

compared with TF_AMP

Evaluation

Performance breakdown——Contribution quantification of the graph rewriting and TC utilization

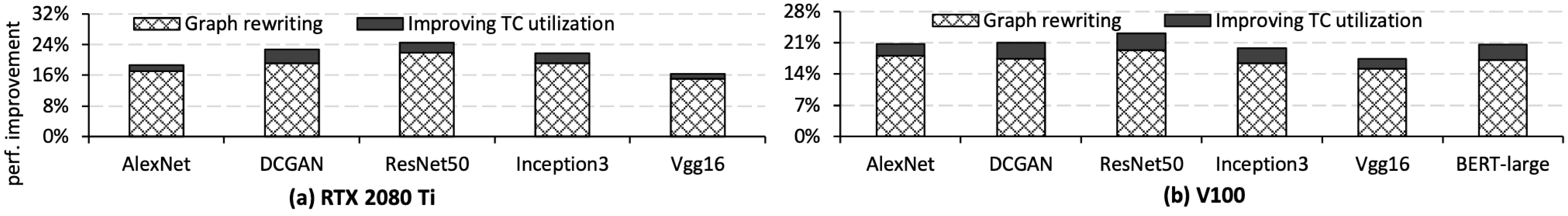


Figure 8: Breakdown of the overall performance improvement from graph rewriting and improving TC utilization.

84.5% of the overall performance improvement (on average)

comes from the graph rewriting

Evaluation

Training accuracy

Table 3: Model accuracy of NN models trained with FP32, TF_AMP and Campo, respectively.

NN models	Top-1 Accuracy (%)			Top-5 Accuracy (%)		
	FP32	TF_AMP	Campo	FP32	TF_AMP	Campo
AlexNet	63.39	64.41	64.38	81.24	81.21	81.19
ResNet50	78.77	78.74	78.75	94.86	94.82	94.85
Inception3	78.42	78.45	78.43	90.15	90.16	90.15
Vgg16	71.58	71.6	71.57	88.28	88.25	88.27
DCGAN	80.12	80.16	80.13	92.47	92.46	92.44
BERT-large	91.35	91.36	91.33	N/A		

Evaluation

Prediction accuracy of performance models

$$M\_A = 1 - \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

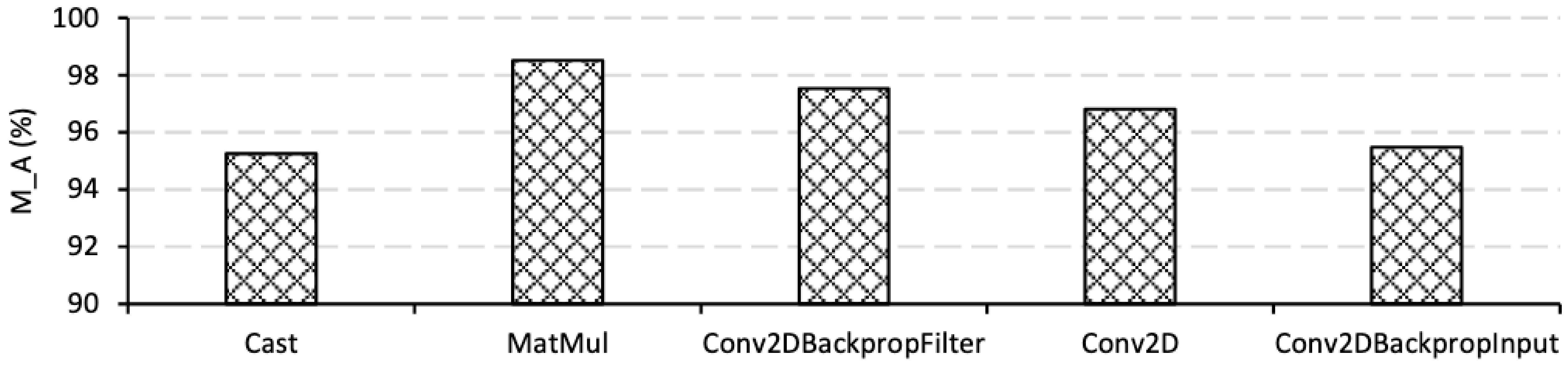


Figure 9: Performance prediction accuracy for five operations based on the operation-specific performance models

prediction error for 143 operations is **5.8%** on average

Evaluation

Power consumption and energy efficiency

Table 4: Average system power consumption of NN models trained with FP32, TF_AMP and Campo on RTX 2080 Ti and V100, respectively.

NN Models	Average System Power (W)					
	RTX 2080 Ti			V100		
	FP32	TF_AMP	Campo	FP32	TF_AMP	Campo
AlexNet	274	268	267	325	319	316
ResNet50	272	265	263	324	313	311
Inception3	273	264	263	326	316	315
Vgg16	273	267	267	324	316	316
DCGAN	275	268	267	327	320	319
BERT-large	N/A	N/A	N/A	332	320	318

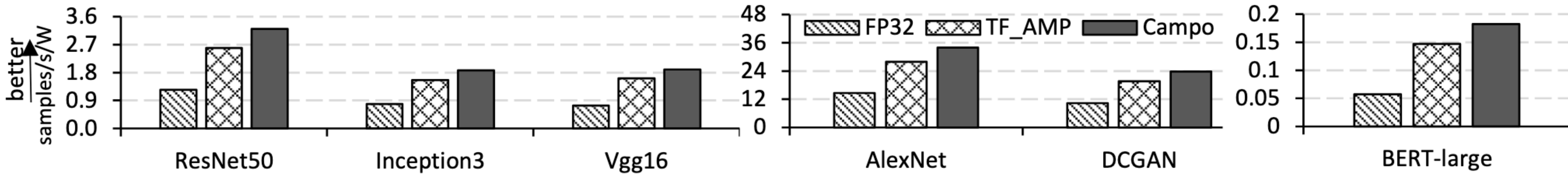


Figure 10: Energy efficiency of NN models trained with FP32, TF_AMP and Campo.

Campo Review

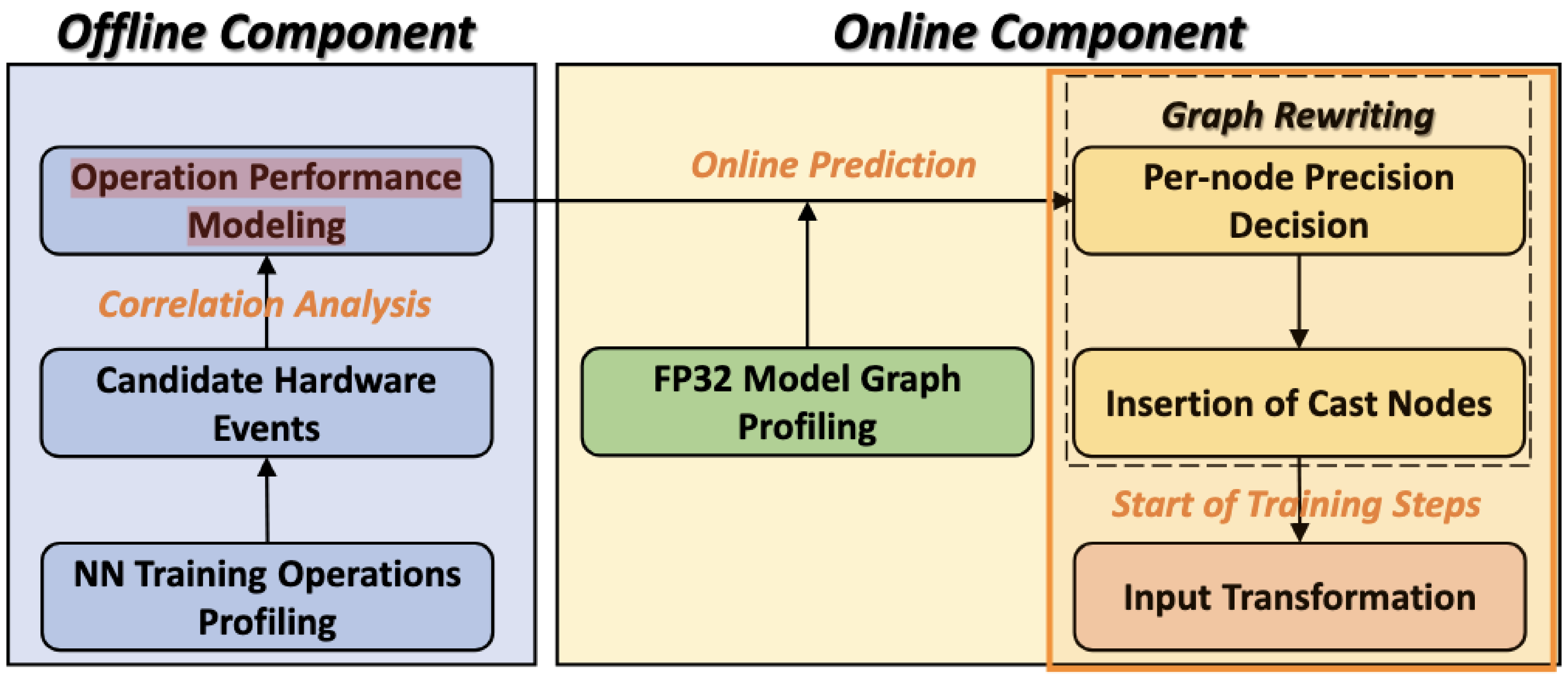


Figure 3: Overview of Campo.

Composition

- Offline component

  performance predict model
- Online component

  model graph profiling

  graph rewriting

Conclusion

Advantage :

- **Reveal the rule that the casting cost can outweigh the performance benefit of using low precision**
- Consider many **dimensions to complete performance model** : Input data size, casting cost, usage of TC
- Reach perfect **balance on training speed and accuracy loss**
- Offline operation-specific performance model can train **faster than dynamic profiling**

Disadvantage :

- Only consider **FP16** as lower precision in this paper
- Campo can only be applied to those **NN models whose dataflow graphs are static**
- Performance model is architecture-specific, which means **poor reusability on different devices**

Thanks

2023-1-9

Presented by Guangtong Li