2022 Argonne Training Program on Extreme-Scale Computing (ATPESC)

Introduction of AI-testbed and hands-on

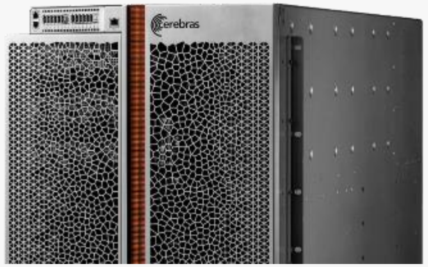
Zhen Xie, Siddhisanket Raskar, Kyle Felker, Sam Foreman, and Venkat Vishwanath

Argonne Leadership Computing Facility (ALCF)

Argonne National Laboratory, Lemont, IL 60439

ALCF AI Testbed

- The ALCF AI Testbed provides an infrastructure for the next-generation of AI-accelerator machines.
  - The AI Testbed aims to help evaluate the usability and performance of machine learning-based high-performance computing applications running on these accelerators. The goal is to better understand how to integrate with existing and upcoming supercomputers at the facility to accelerate science insights.



Cerebras CS-2 Wafer-Scale Deep Learning Accelerator

SambaNova Dataflow Accelerator

Graphcore MK1 Graphcore Intelligent Processing Unit (IPU)

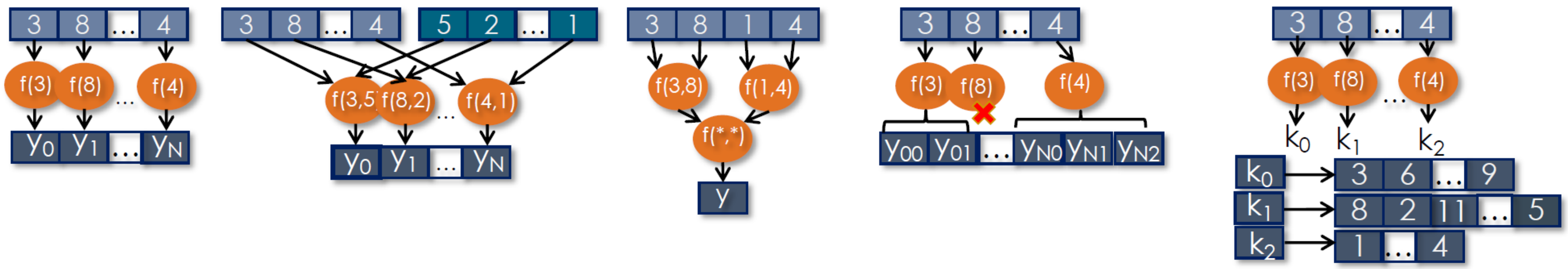
Groq Tensor Streaming Processor

Habana Gaudi Tensor Processing Cores

Hardware

Motivation of hardware design

- A Flexible Dataflow Substrate: Parallel Patterns
  - Looping abstractions with extra information on parallelism & access patterns



Map
element-wise
function f

Zip
element-wise
function f
(multi-collection)

Reduce
combine all
elements with f
(f is associative)

FlatMap
element-wise
function
≥0 values out
per element

GroupBy
group elements
into buckets
based on key

```
y = vector + 4
y = vector * 10
y = sigmoid(vector)
```

```
y = vecA + vecB
y = vecA / vecB
y = max(vecA,vecB)
```

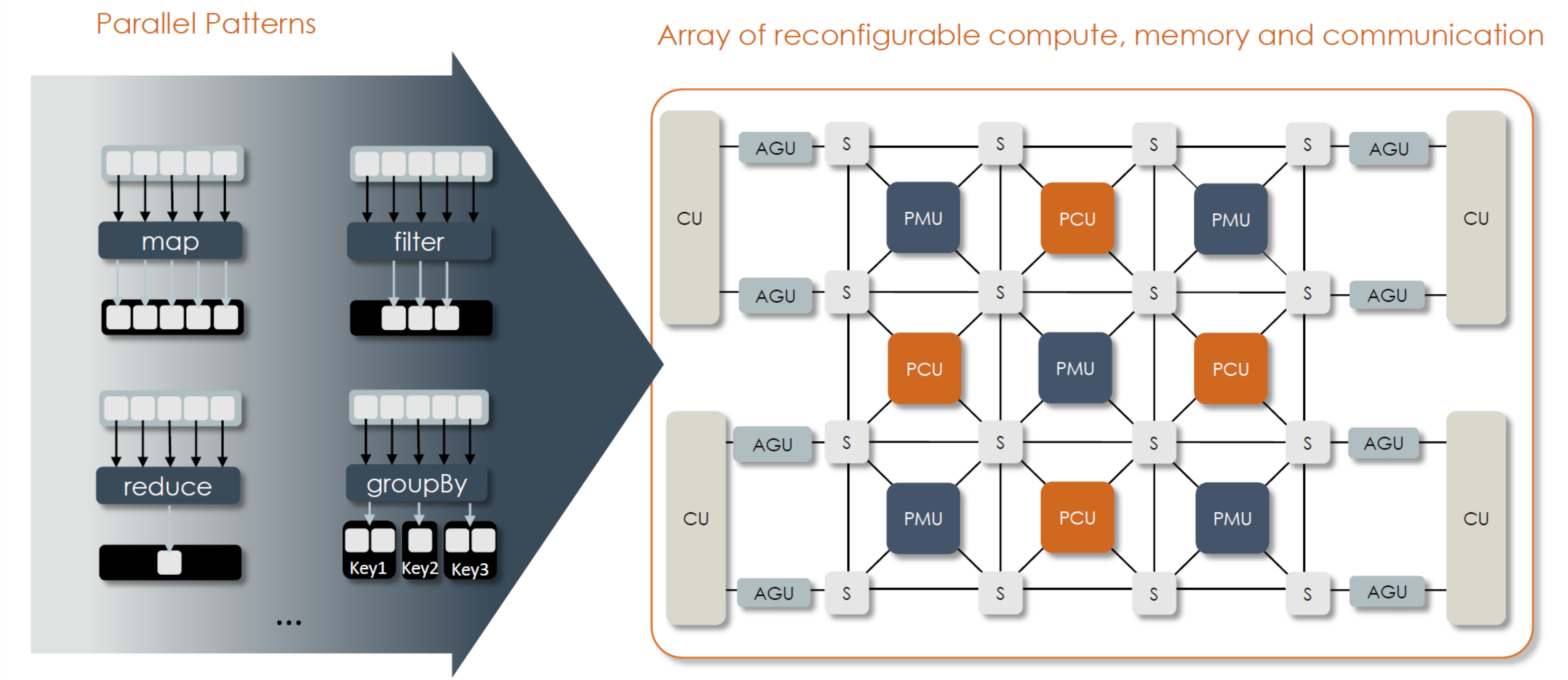
```
y = vector.sum
y = vector.product
y = max(vector)
```

```
SELECT * FROM vector
WHERE elem < 5
```

```
vector.groupBy{e => e % 3}
```

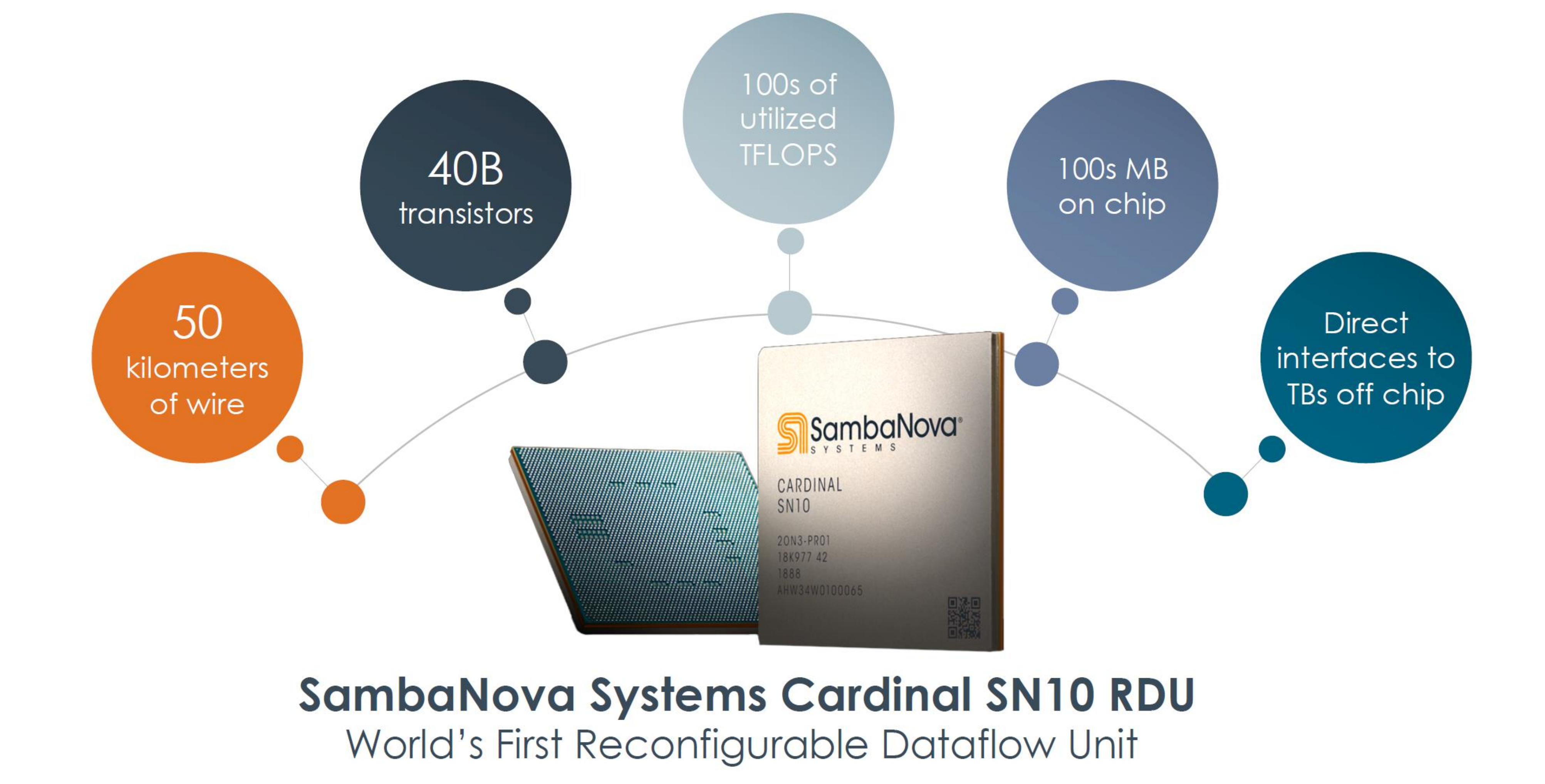
Motivation of hardware design

- Reconfigurable Dataflow Architecture (RDA)



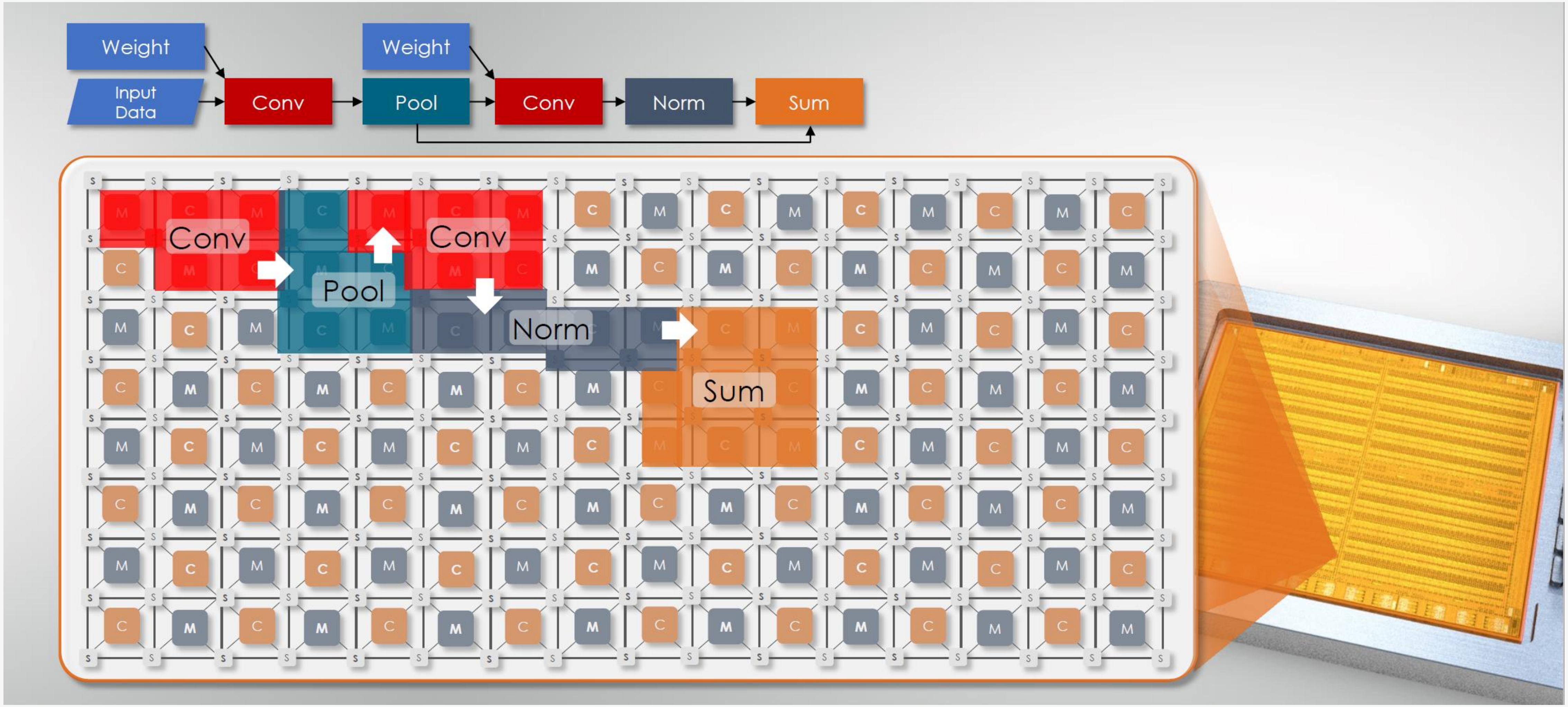
Hardware design

- Reconfigurable Dataflow Architecture (RDU)



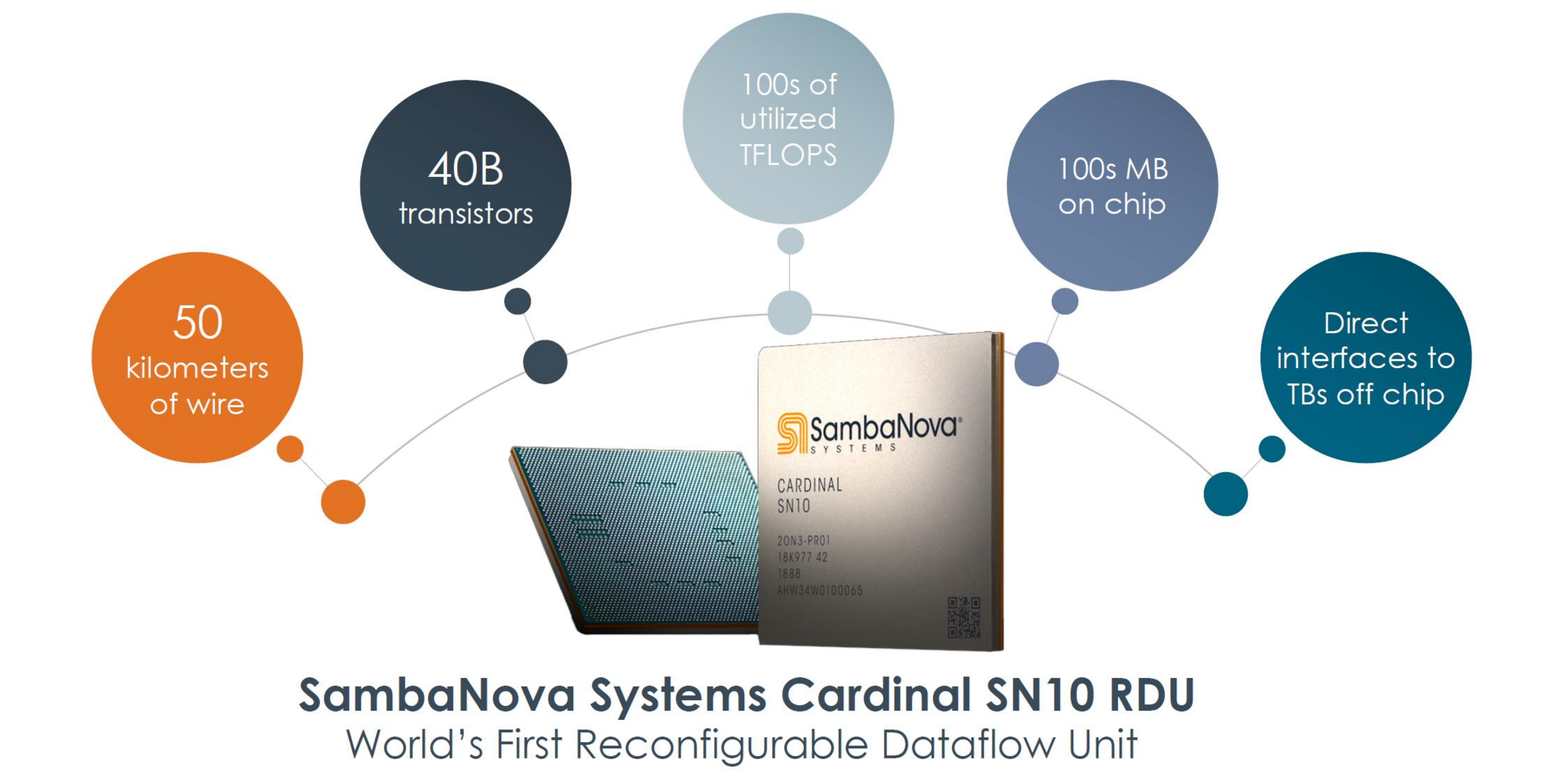
Hardware design

- Rapid Dataflow Compilation to RDU



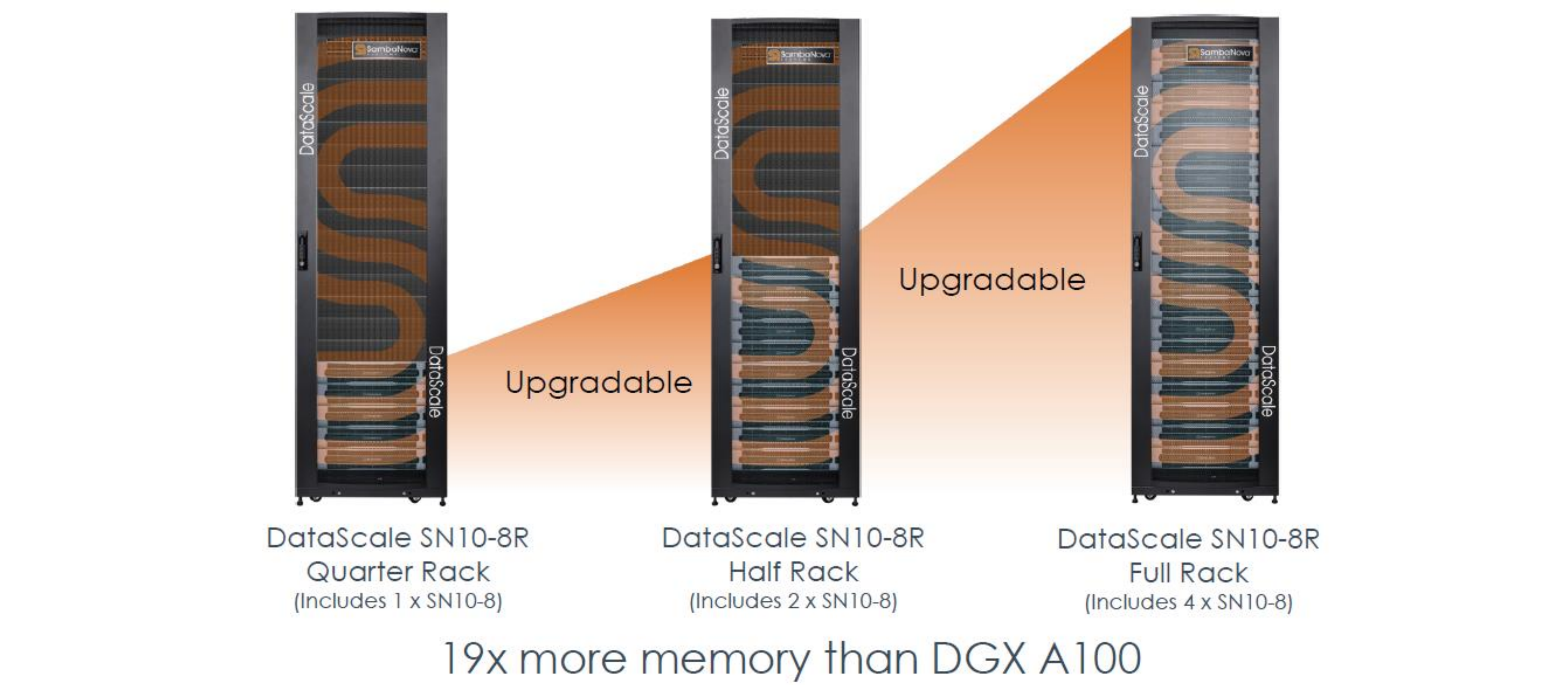
Hardware design

- Reconfigurable Dataflow Architecture (RDU)



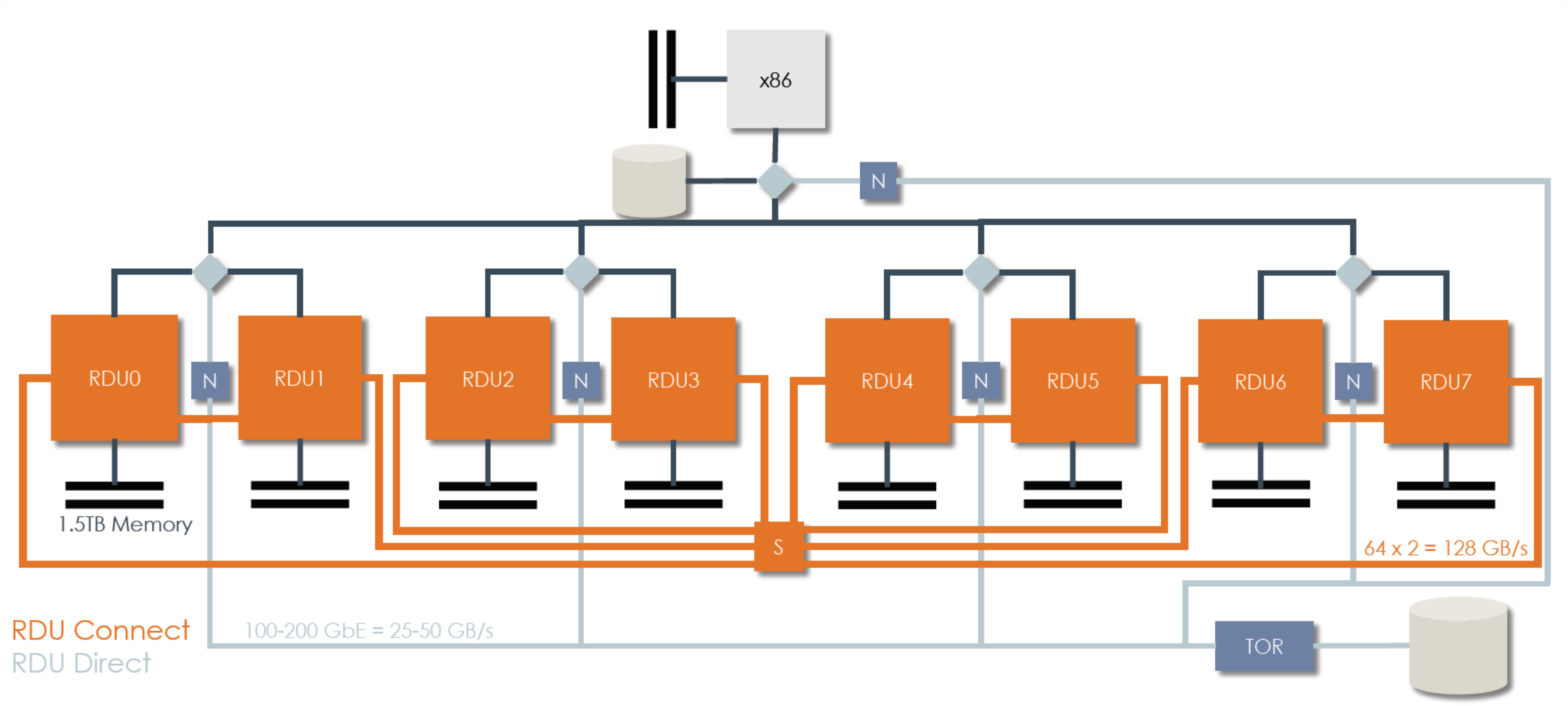
Hardware design

- DataScale SN10-8R: Scalable performance for training and inference



Hardware design

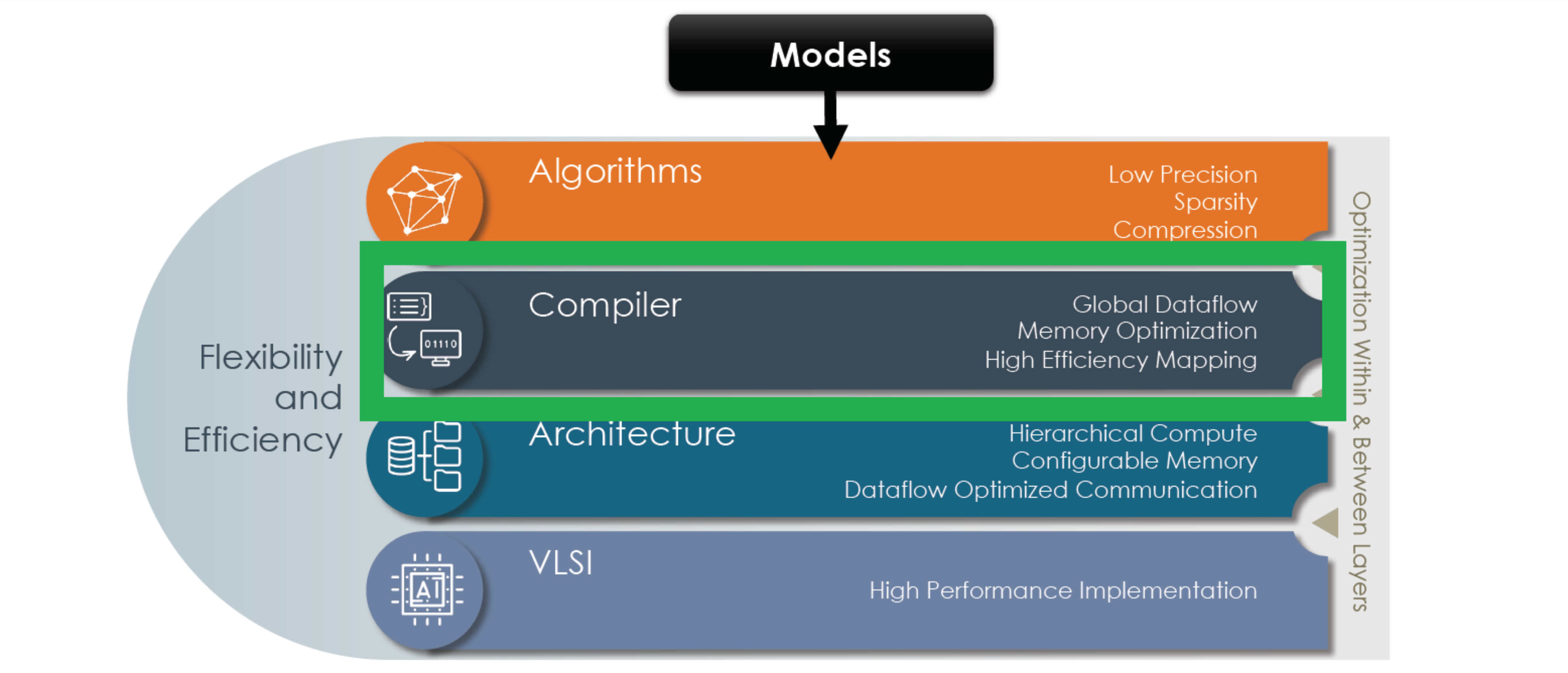
- Excelling at Model and Data Parallel Execution Models



Software

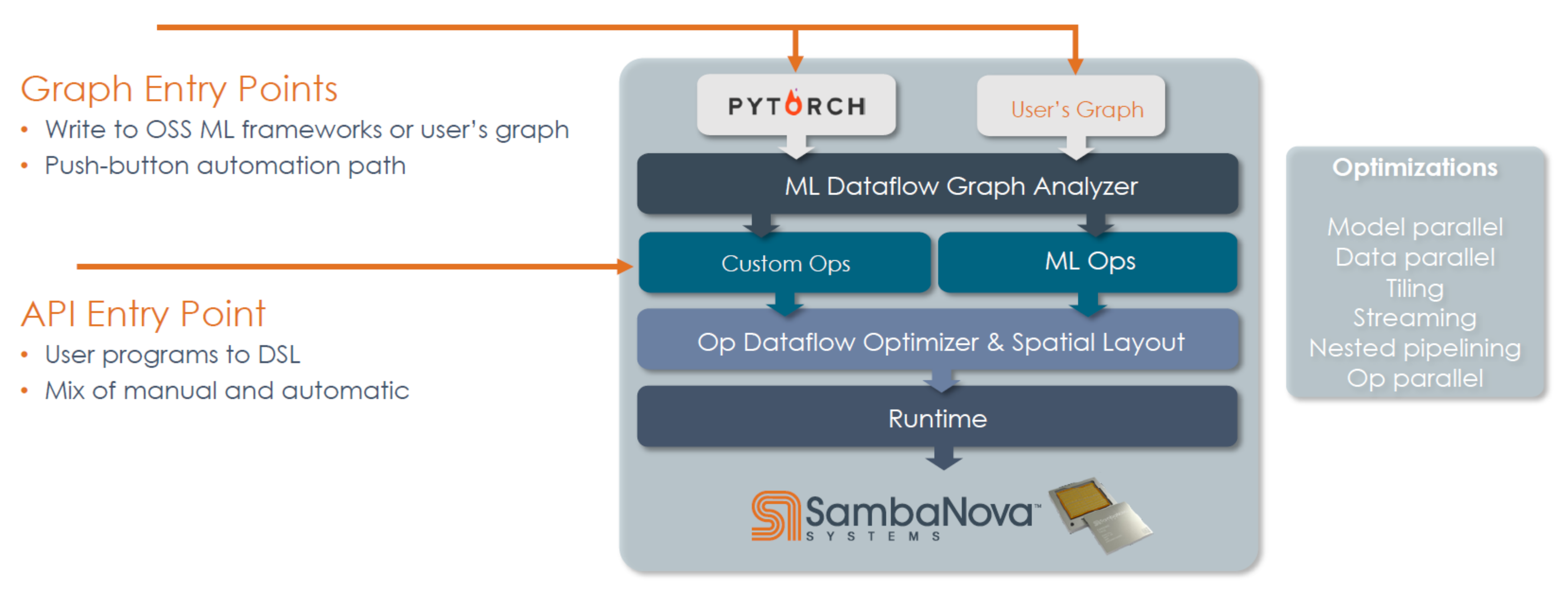
Software design

- Full stack co-engineering yields optimizations where best delivered with the highest impact



Software design

- SambaFlow Open Software for DataScale Systems



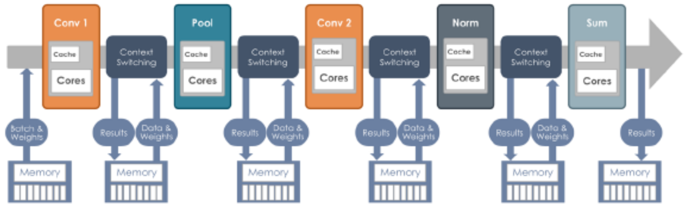
Software design

- SambaFlow Open Software for DataScale Systems

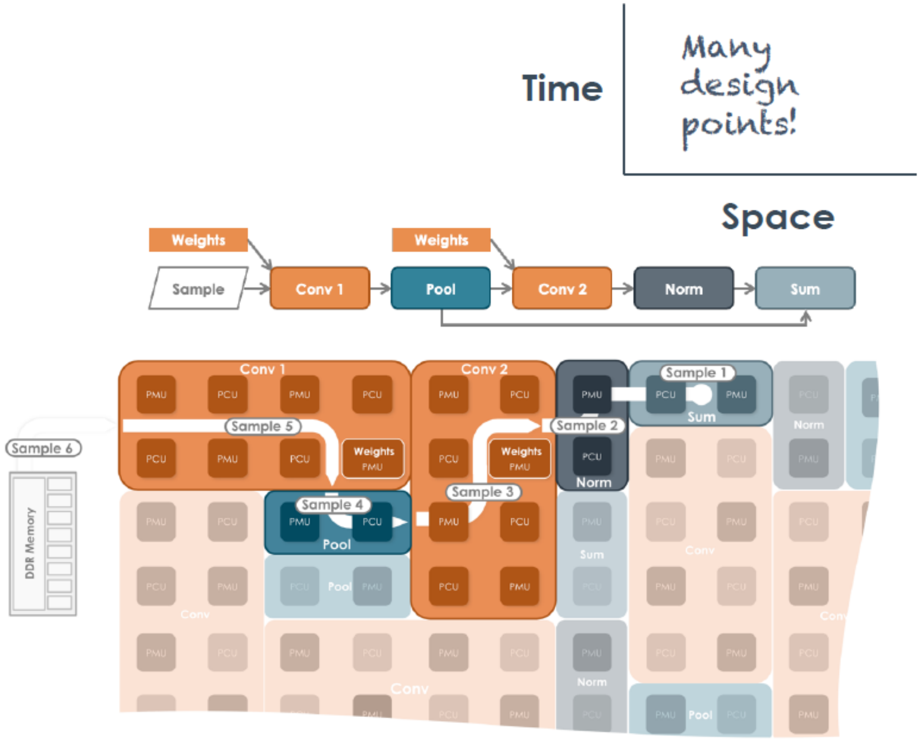
Spatial Dataflow within RDU

```
t1 = conv(in)
t2 = pool(t1)
t3 = conv(t2)
t4 = norm(t3)
t5 = sum(t4)
```

Time: Kernel by Kernel Execution



Traditional compilers map operations to processor instructions in time
Communication through the memory hierarchy is implicit and handled by hardware

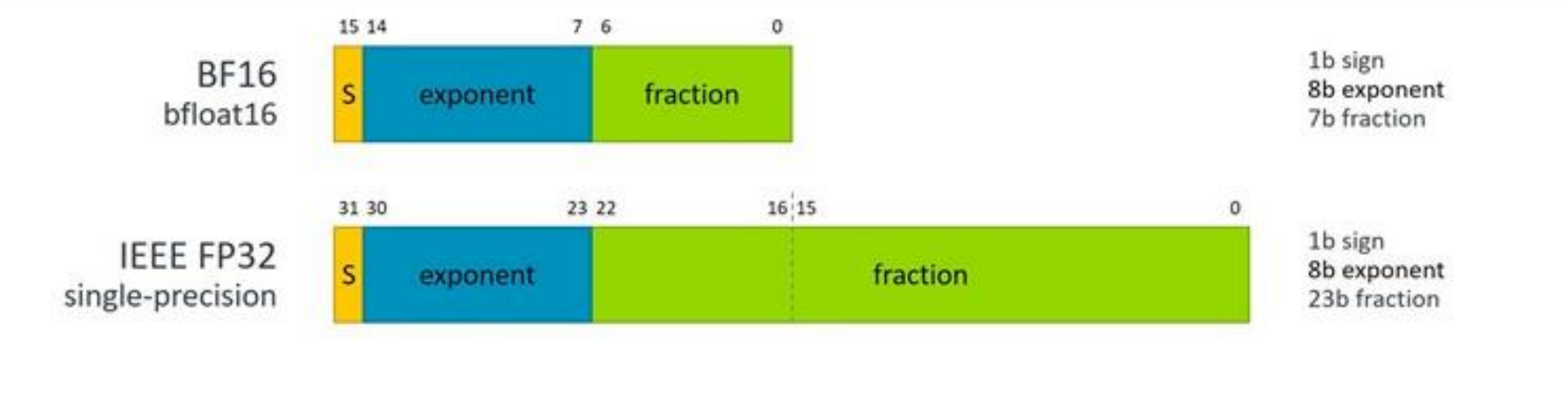


Dataflow compilers map operations to instructions in time and in space and program the communication between them
SambaFlow eliminates overhead and maximizes utilization

Argonne NATIONAL LABORATORY

Software design

- A bit about precision
  - The main idea of bfloat16 to provide a 16-bit floating point format that has the same dynamic range as a standard IEEE-FP32, but with less accuracy. That amounted to matching the size of the FP32 exponent field at 8 bits and shrinking the size of the FP32 fraction field down to 7 bits. With bfloat16, SambaNova can provide better training throughput.

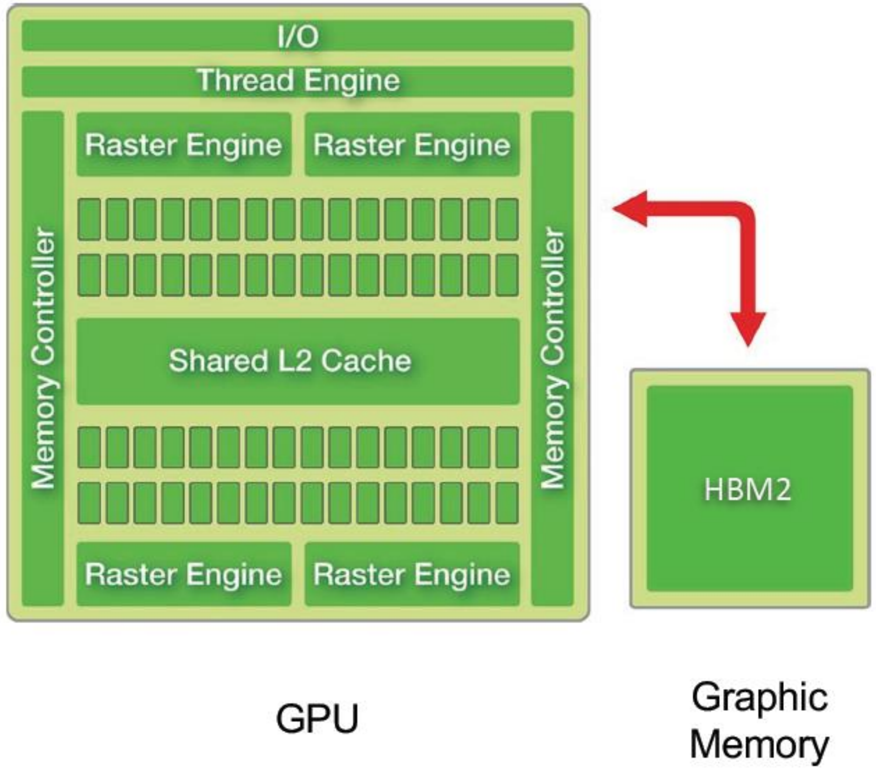


Hardware

Motivation of hardware design

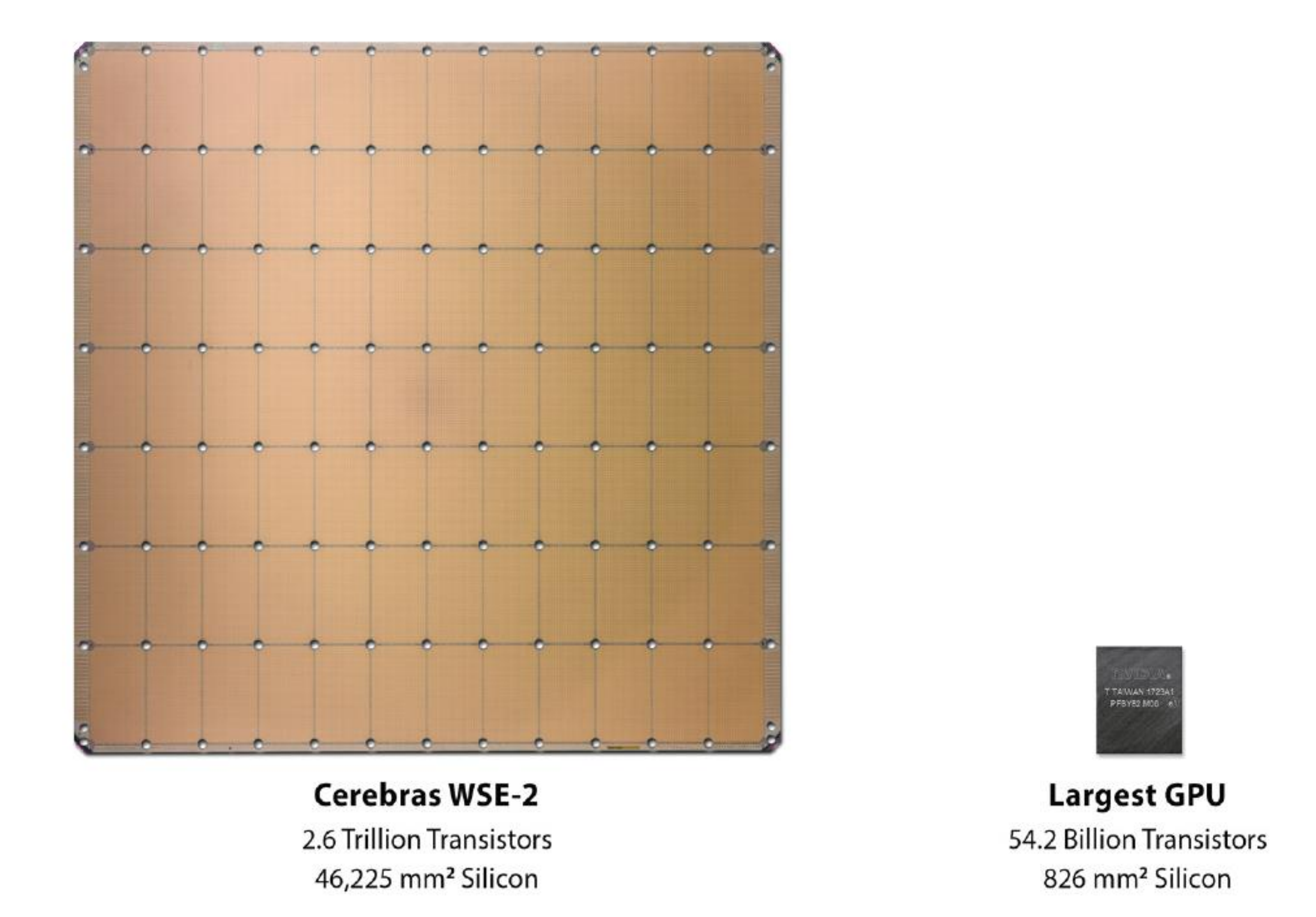
- GPU approach

- < 10% silicon area used for Deep Learning
  - > 90% used for graphics: Raster Engines, Shaders, Texture Maps, Thread and Instruction Control
- Memory is far from graphics core
  - Little on-chip memory
  - Cache memory hierarchy
- Graphics cores not built for communication
  - On chip: low bandwidth, through memory
  - Off chip: even lower bandwidth, PCIe/NVLink
- Designed for dense-matrix operations
  - Sparsity devastates performance
  - Implemented as CPU co-processor



Hardware design

- Cerebras CS-2: The world's only purpose-built Deep Learning solution



Cerebras Wafer Scale Engine (WSE)

The Most Powerful Processor for AI

400,000 AI-optimized cores

46,225 mm^2 silicon

1.2 trillion transistors

18 Gigabytes of On-chip Memory

9 PByte/s memory bandwidth

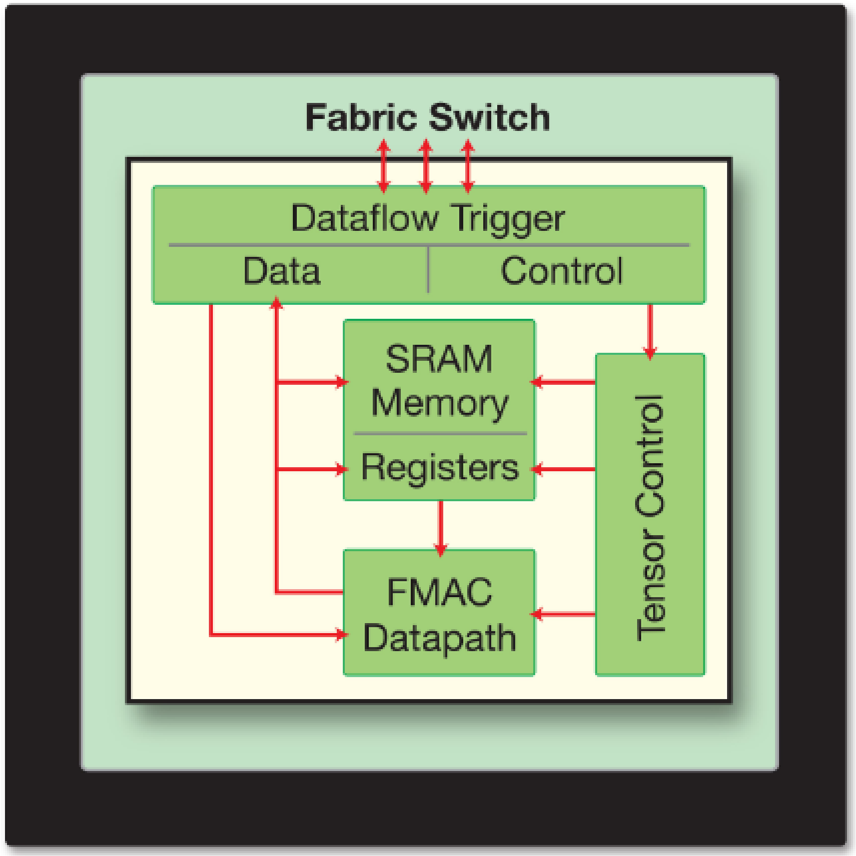
100 Pbit/s fabric bandwidth

TSMC 16nm process

Hardware design

- The CS WSE architecture is built for deep learning

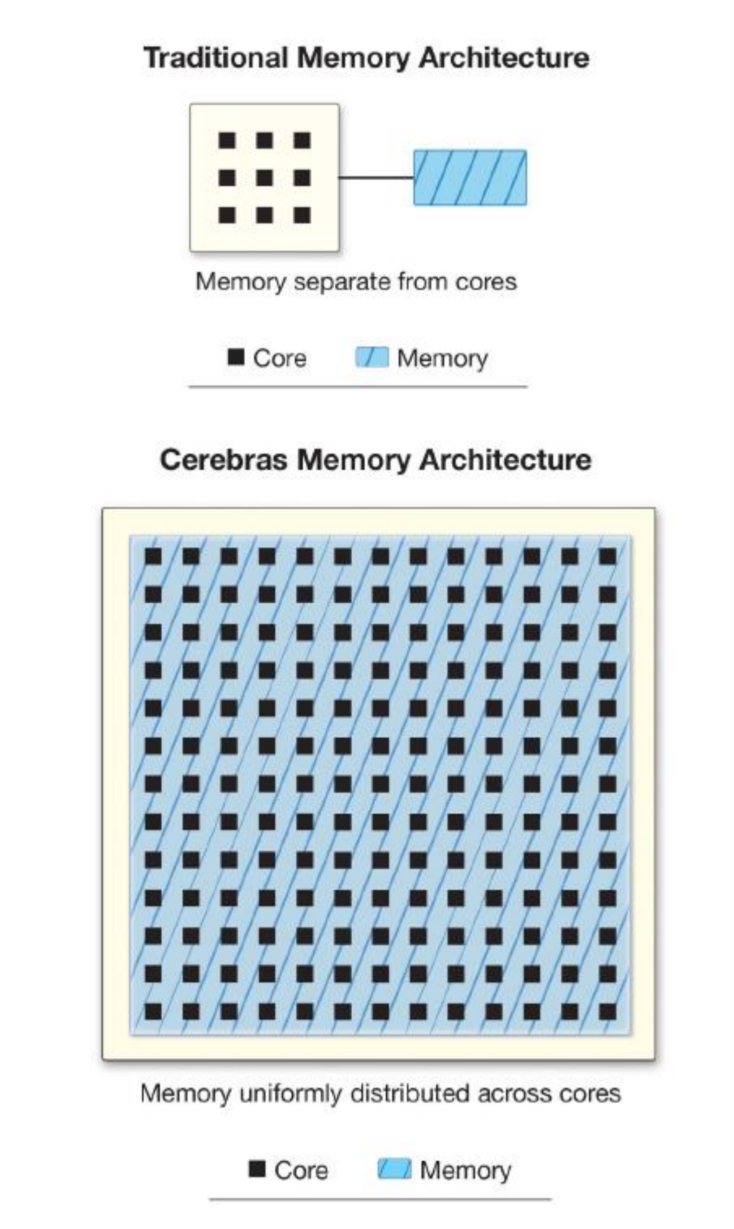
- AI-optimized **compute**
  - Fully-programmable core, ML-optimized extensions
    - e.g. arithmetic, logical, load/store, branch
  - Dataflow architecture optimized for sparse, dynamic workloads
    - Higher performance and efficiency for sparse NN



Hardware design

- The CS WSE architecture is built for deep learning

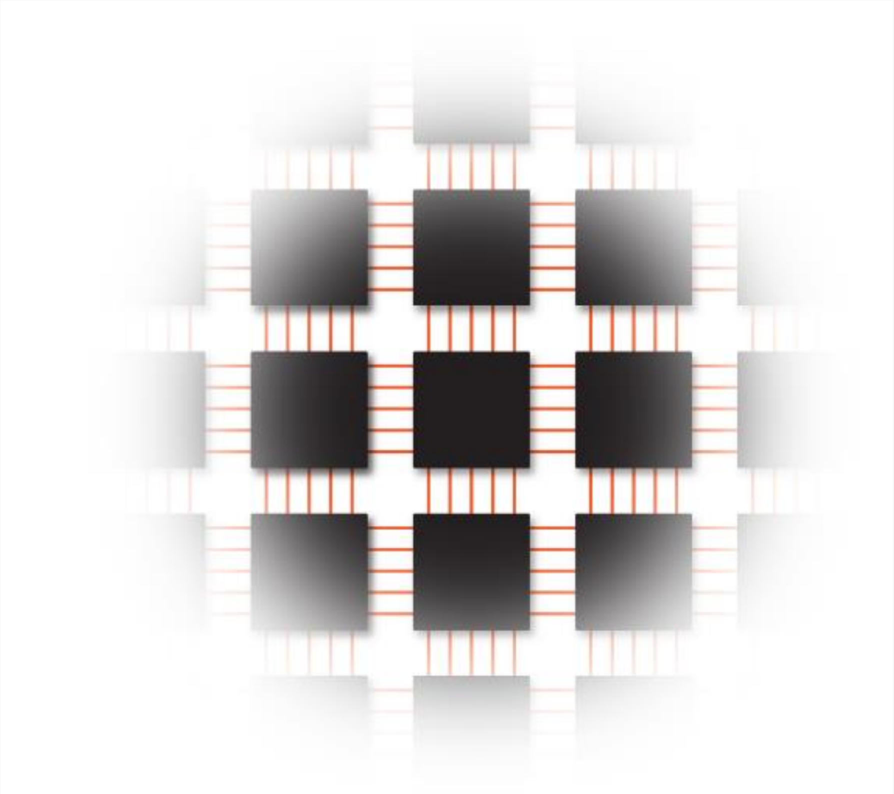
- AI-optimized **memory**
  - Traditional memory architectures shared memory far from compute
  - The right answer is distributed, high performance, on-chip memory



Hardware design

- The CS WSE architecture is built for deep learning

- AI-optimized **communication**
  - High bandwidth, low latency cluster-scale networking on chip
  - Fully-configurable to user-specified topology



Hardware design

- Cerebras CS-2: Comparison with NVIDIA A100 GPU

	Cerebras WSE-2	A100	Cerebras Advantage
Chip size	46,225 mm^2	826 mm^2	**56 X**
Cores	850,000	6,912 + 432	**123 X**
On chip memory	40 Gigabytes	40 Megabytes	**1,000 X**
Memory bandwidth	20 Petabytes/sec	1,555 Gigabytes/sec	**12,862 X**
Fabric bandwidth	220 Petabits/sec	600 Gigabytes/sec	**45,833 X**

Table 1. Overview of the magnitude of advancement made by the Cerebras WSE-2.

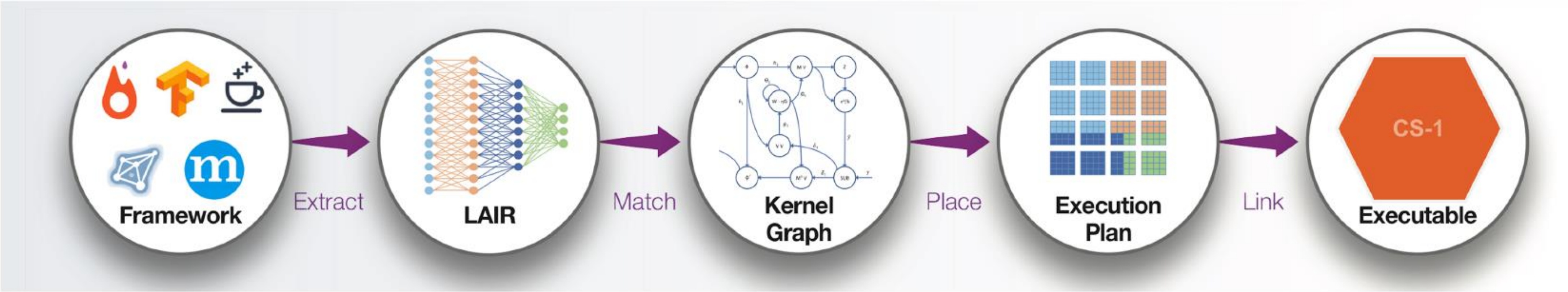
Argonne NATIONAL LABORATORY

cerebras

Software

Software design

- Cerebras Software Stack handles graph compilation



- **Extract** – Obtain graph representation of model from framework and express it in our intermediate form.
- **Match** – Consult kernel library for kernels that implement portions of model.
- **Place & Route** – Assign kernels to regions of fabric guided by graph connectivity and kernel performance functions.
- **Link** – Create executable output that can be loaded and run by CS-1.

Software design

- Program using familiar ML Frameworks

The user starts as usual by developing their ML model.

Cerebras integrates with popular ML Frameworks so researchers can write their models using familiar tools.

TensorFlow PyTorch

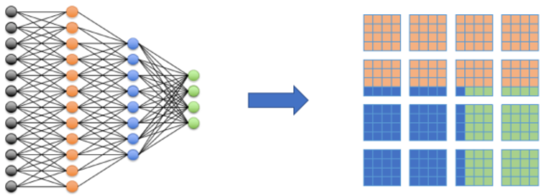
We have TensorFlow support today and are working on Pytorch

Software design

- Model extraction from ML Framework -> LAIR

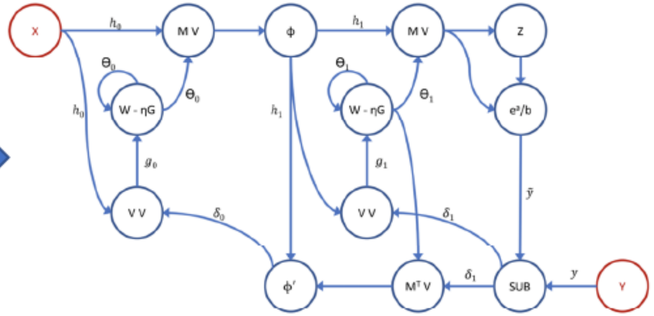
Cerebras LAIR (**L**inear **A**lgebra **I**ntermediate **R**epresentation) is the standard input into the Cerebras software stack.

We extract the explicit linear algebra graph representation of the model from the ML Framework and translate it into LAIR.

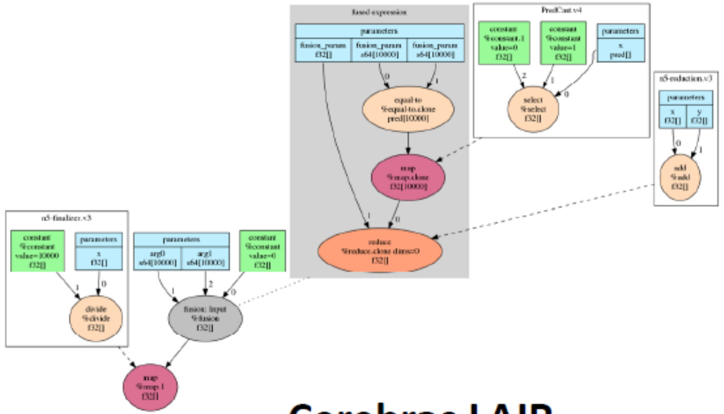


TensorFlow

PyTorch



Framework IR



Cerebras LAIR

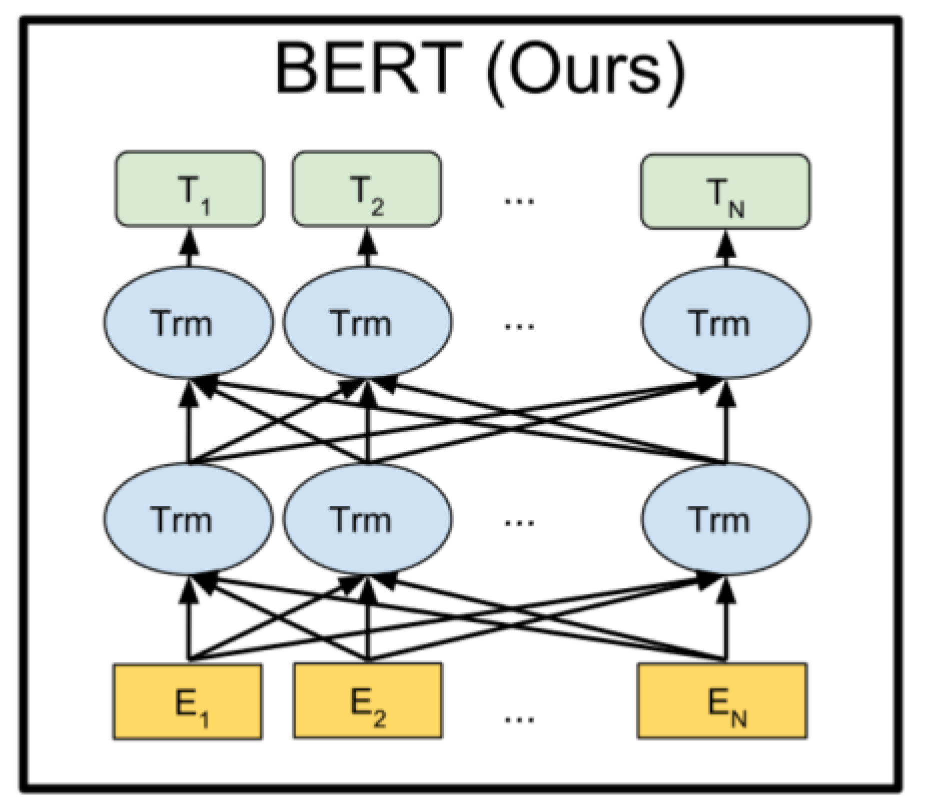
Argonne NATIONAL LABORATORY

Hands-on Section on SambaNova and Cerebras

BERT (language model) on hands-on section

- Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google.

- The original English-language BERT has two models:
  - (1) BERT_BASE: 12 encoders with 12 bidirectional self-attention heads;
  - (2) BERT_LARGE: 24 encoders with 16 bidirectional self-attention heads.



SambaNova

- 1. Login to sn:

```
ssh ALCFUserID@sambanova.alcf.anl.gov
ssh sm-01
```

- 2. SDK setup:

```
source /software/sambanova/envs/sn_env.sh
```

- 3. Copy scripts:

```
cp
/var/tmp/Additional/slurm/Models/ANL_Acceptance_RC1_11_5
/bert_train-inf.sh ~/
```

- 4. Run scripts:

```
cd ~; ./bert_train-inf.sh;
```

Cerebras

- 1. Login to CS-2:

```
ssh ALCFUserID@cerebras.alcf.anl.gov
ssh cs2-01-med1
```

- 2. SDK setup:

```
source /software/sambanova/envs/sn_env.sh
```

- 3. Copy scripts:

```
cp -r /software/cerebras/model_zoo ~/
cd modelzoo/transformers/tf/bert
```

modify data_dir to "/software/cerebras/dataset/bert_large/msl128/" in configs/params_bert_large_msl128.yaml

Cerebras

4. Run scripts:

```
MODELDIR=model_dir_bert_large_msl128_$(hostname)

rm -r $MODELDIR

time -p csrun_cpu python run.py --mode=train --compile_only --params configs/params_bert_large_msl128.yaml --model_dir $MODELDIR --cs_ip $CS_IP

time -p csrun_wse python run.py --mode=train --params configs/params_bert_large_msl128.yaml --model_dir $MODELDIR --cs_ip $CS_IP
```

Thanks!

Zhen Xie, Siddhisanket Raskar, Kyle Felker, Sam Foreman, and Venkat Vishwanath

Argonne Leadership Computing Facility

2022 Argonne Training Program on Extreme-Scale Computing (ATPESC)