RDARuntime: An OS for AI accelerators

ROSS 2023: 11/12/2023

Benjamin Glick, Arjun Sabnis, Renate Kempf, Arnav Goel, Aarti Lalwani, Guoyao Feng, Kiran Ranganath

Copyright © 2023 SambaNova Systems

Contents

- Intro to RDU and RDA programming
- Related work
- RDARuntime architecture and features
- Experimental analysis

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

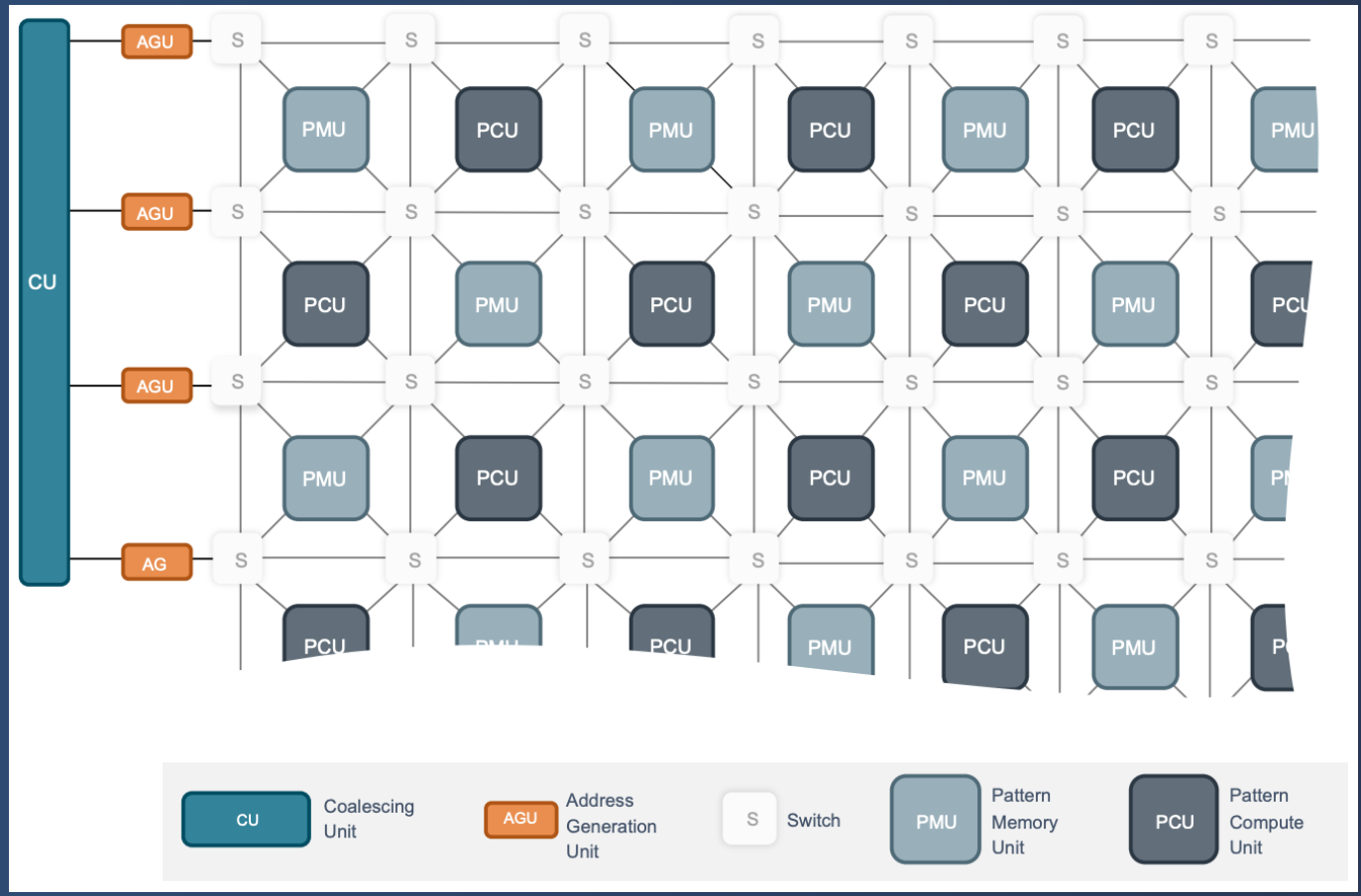
Why OS for RDU?
It's Different!

SambaNova® SYSTEMS

Copyright © 2023 SambaNova Systems

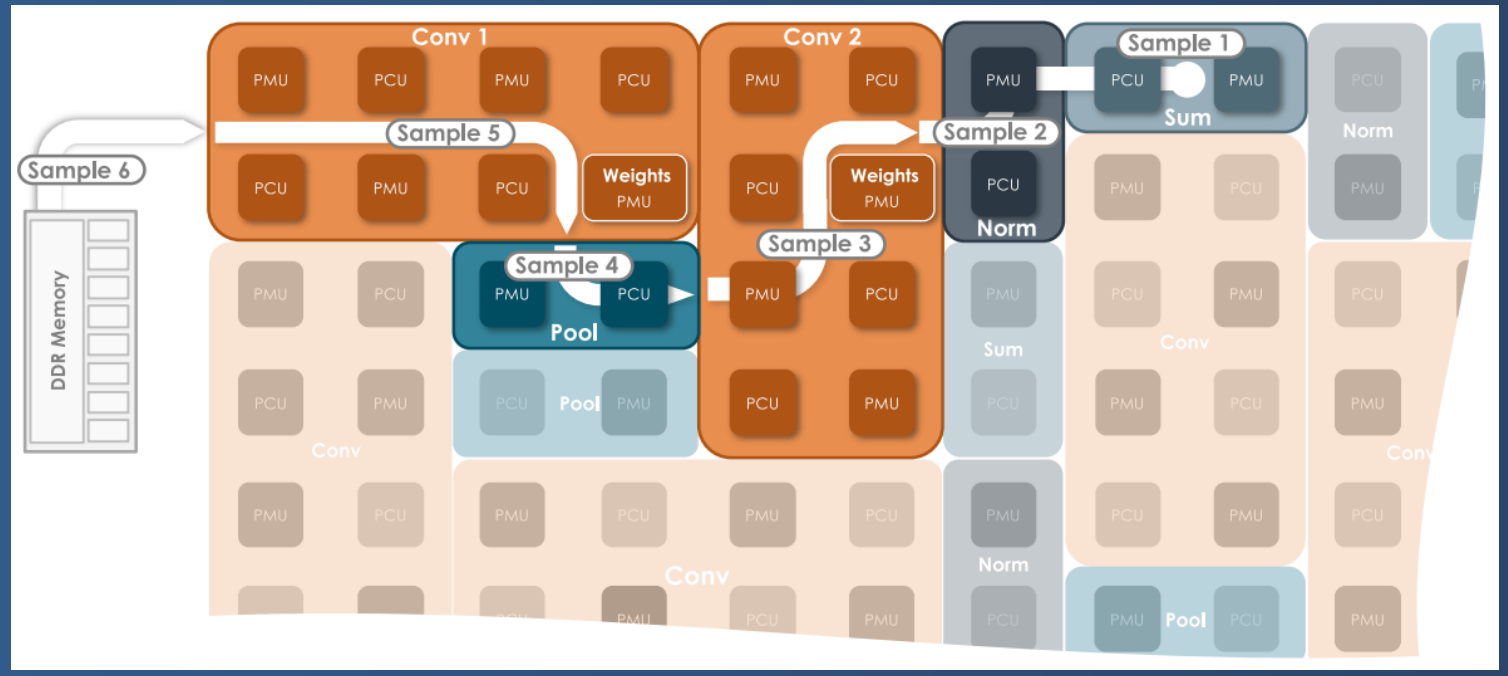
Reconfigurable Dataflow Architecture

- Each chip (RDU) is a checkerboard of memory and compute resources
  - **PCU** has vector and scalar compute operations
  - **PMU** is a multipurpose scratchpad
  - **Switches** forward packets across the chip
- The **AG/CU**s at the edge of the compute checkerboard mediate off-chip accesses to memory, host, and other RDUs



Copyright © 2023 SambaNova Systems

Introduction to Reconfigurable Dataflow Architecture



- At **compile time**, the ML model is mapped to a physical on-chip layout making use of compute/ mem/IO resources
- At **runtime**, the chip is statically programmed and data flows across the chip to enable computation

Copyright © 2023 SambaNova Systems

Units of Compute

- Each RDU contains *N* instances of the RDA, which we call a tile
- Tiles are connected to each other by an on-chip network
- Each system consists of 8 RDUs and an x86 based host
- Each chip is connected to its peers via an internal fabric
- Systems are connected to each other by 400G Ethernet

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

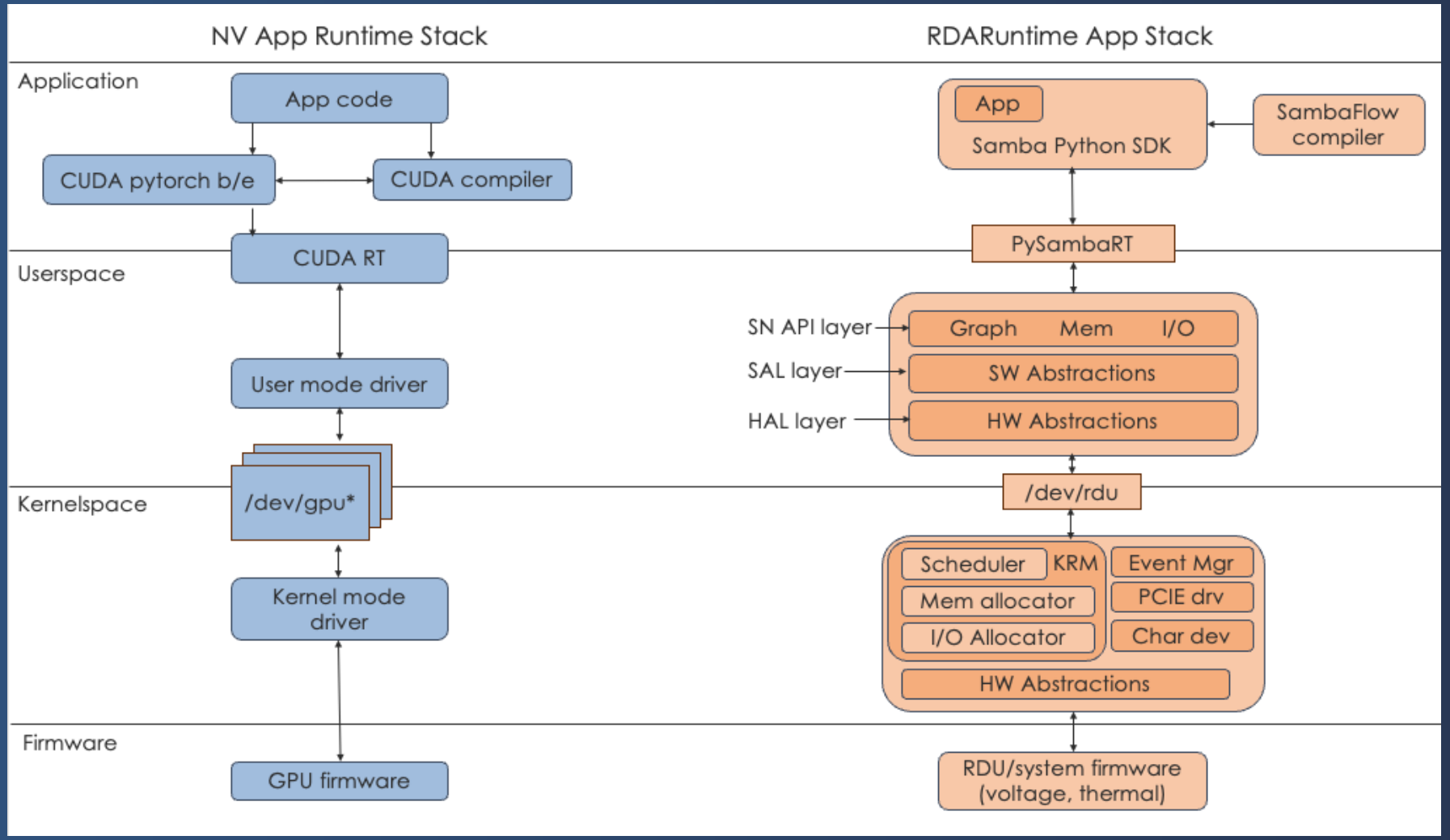
Related Work

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

CUDA Runtime vs RDARuntime

- One device file vs many device files
- Device firmware vs kernel driver
- High level APIs

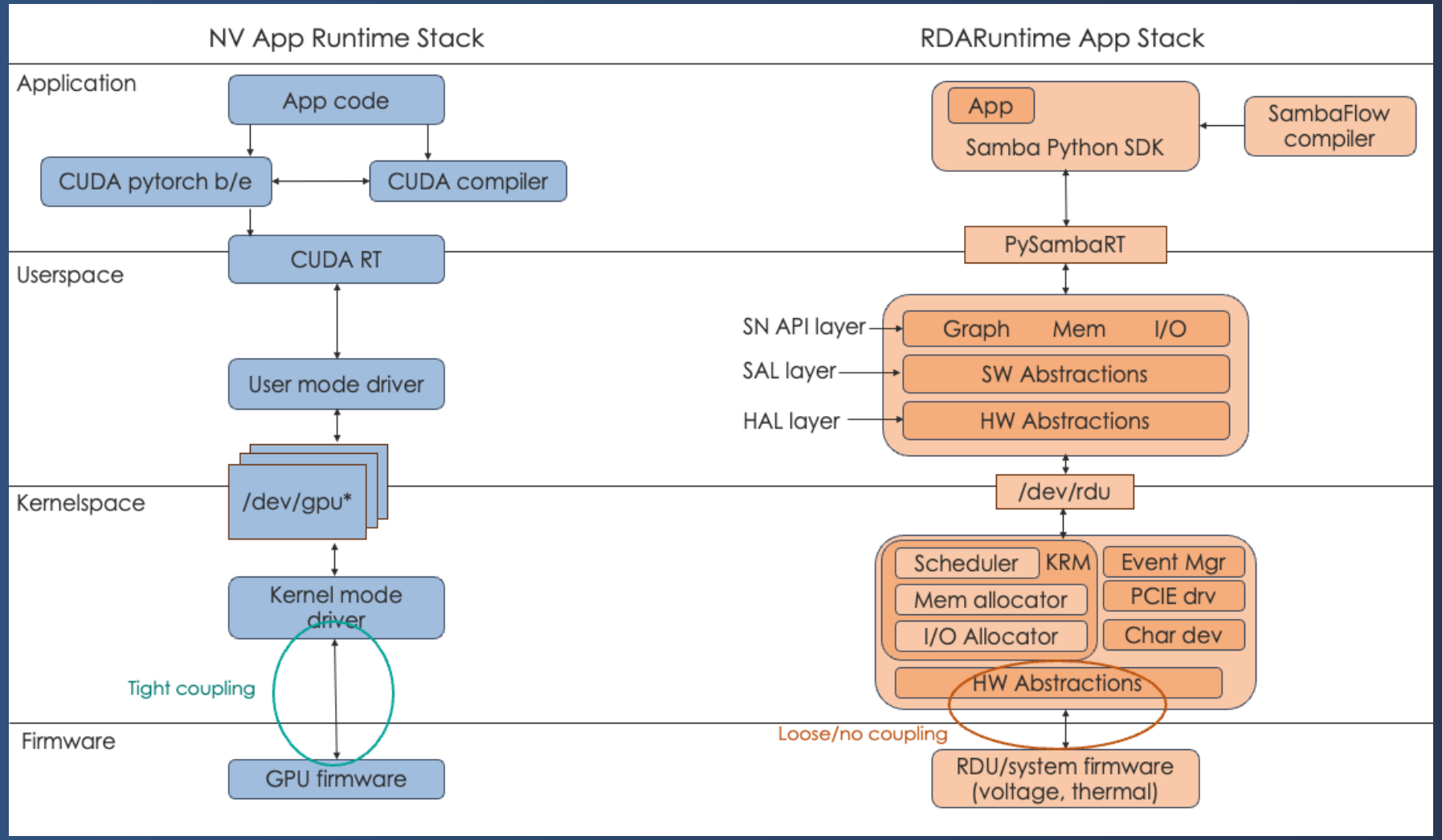


SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

CUDA Runtime vs RDARuntime

- One device file vs many device files
- Device firmware vs kernel driver
- High level APIs

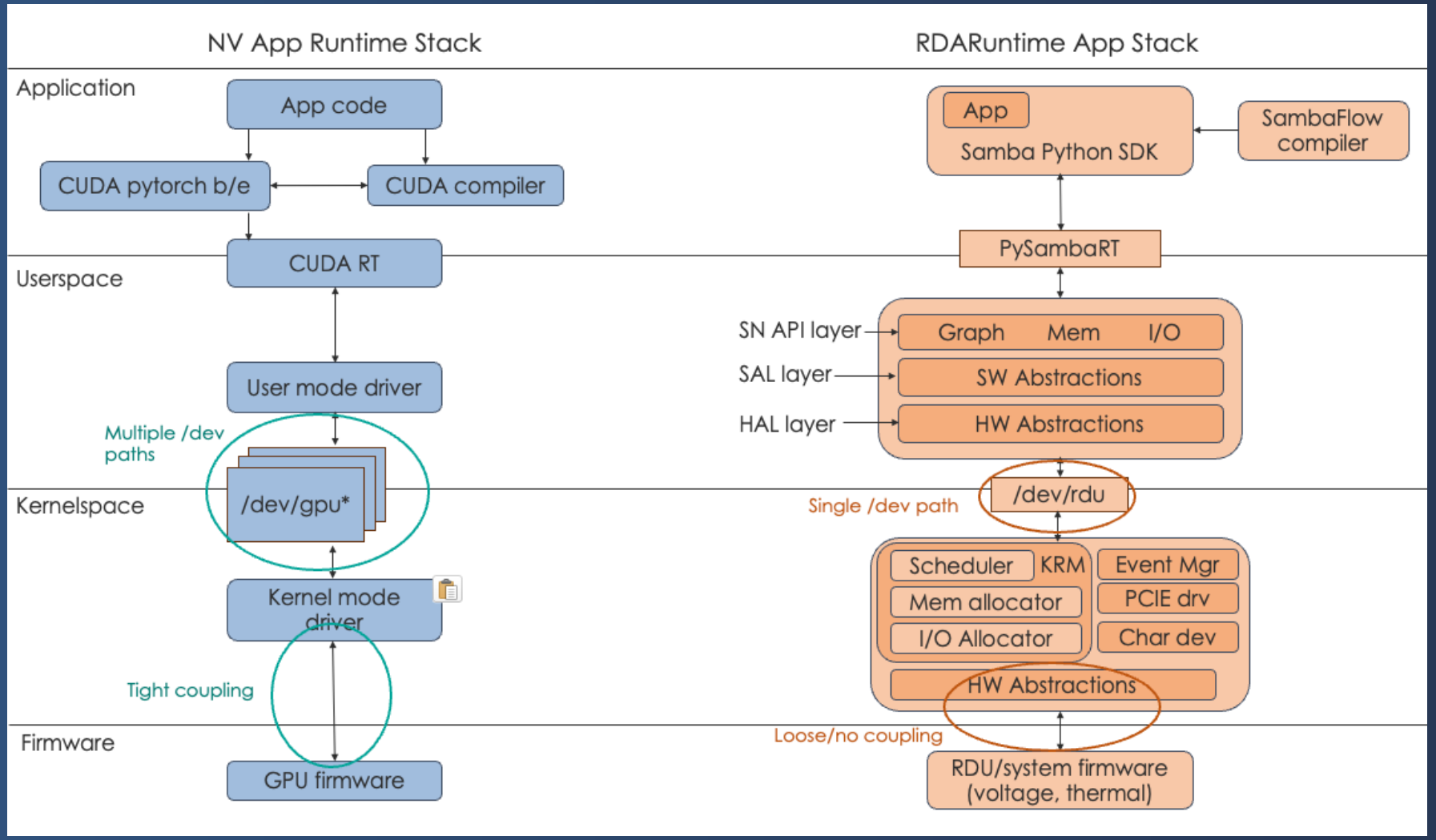


SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

CUDA Runtime vs RDARuntime

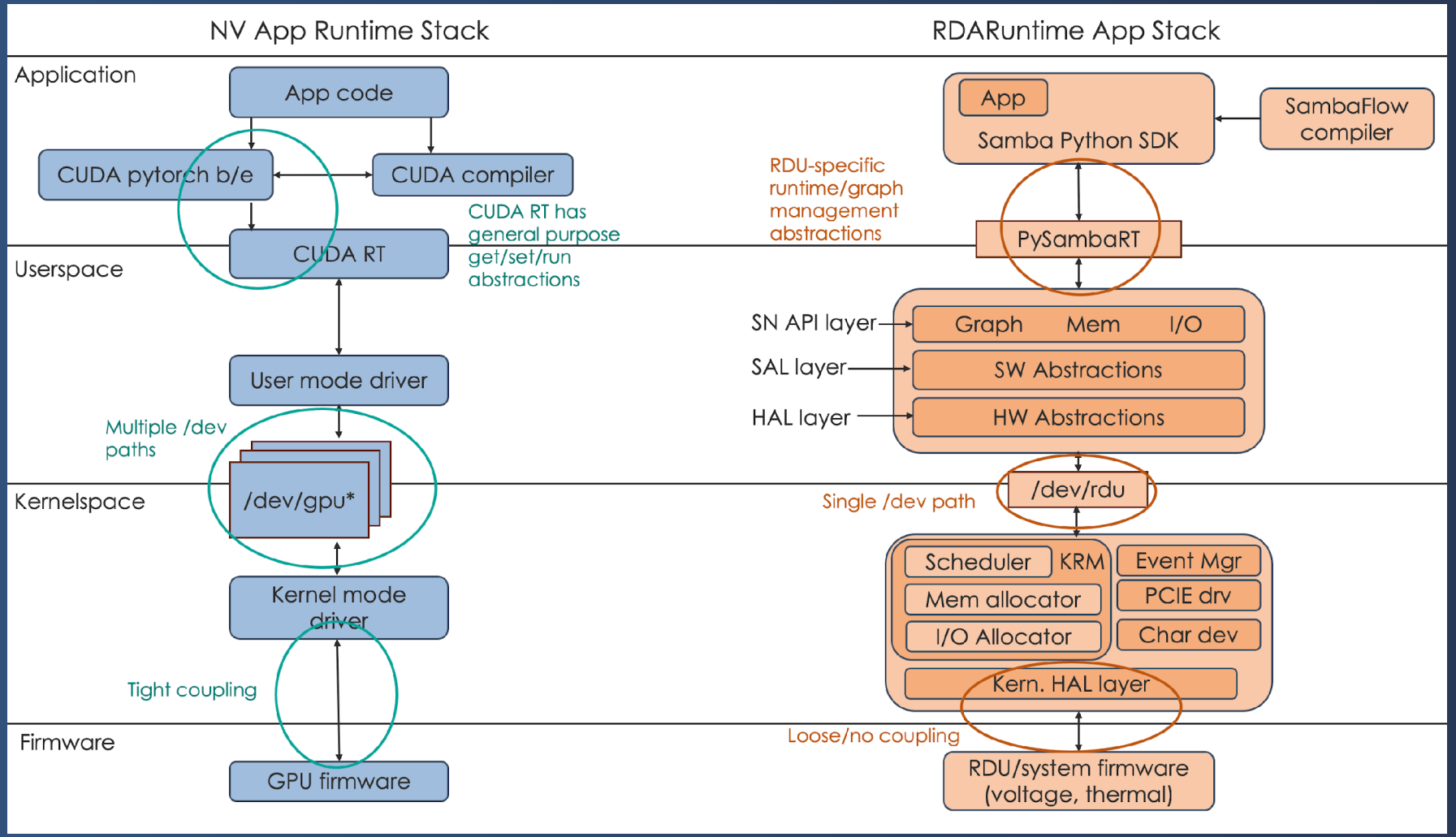
- One device file vs many device files
- Device firmware vs kernel driver
- High level APIs



Copyright © 2023 SambaNova Systems

CUDA Runtime vs RDARuntime

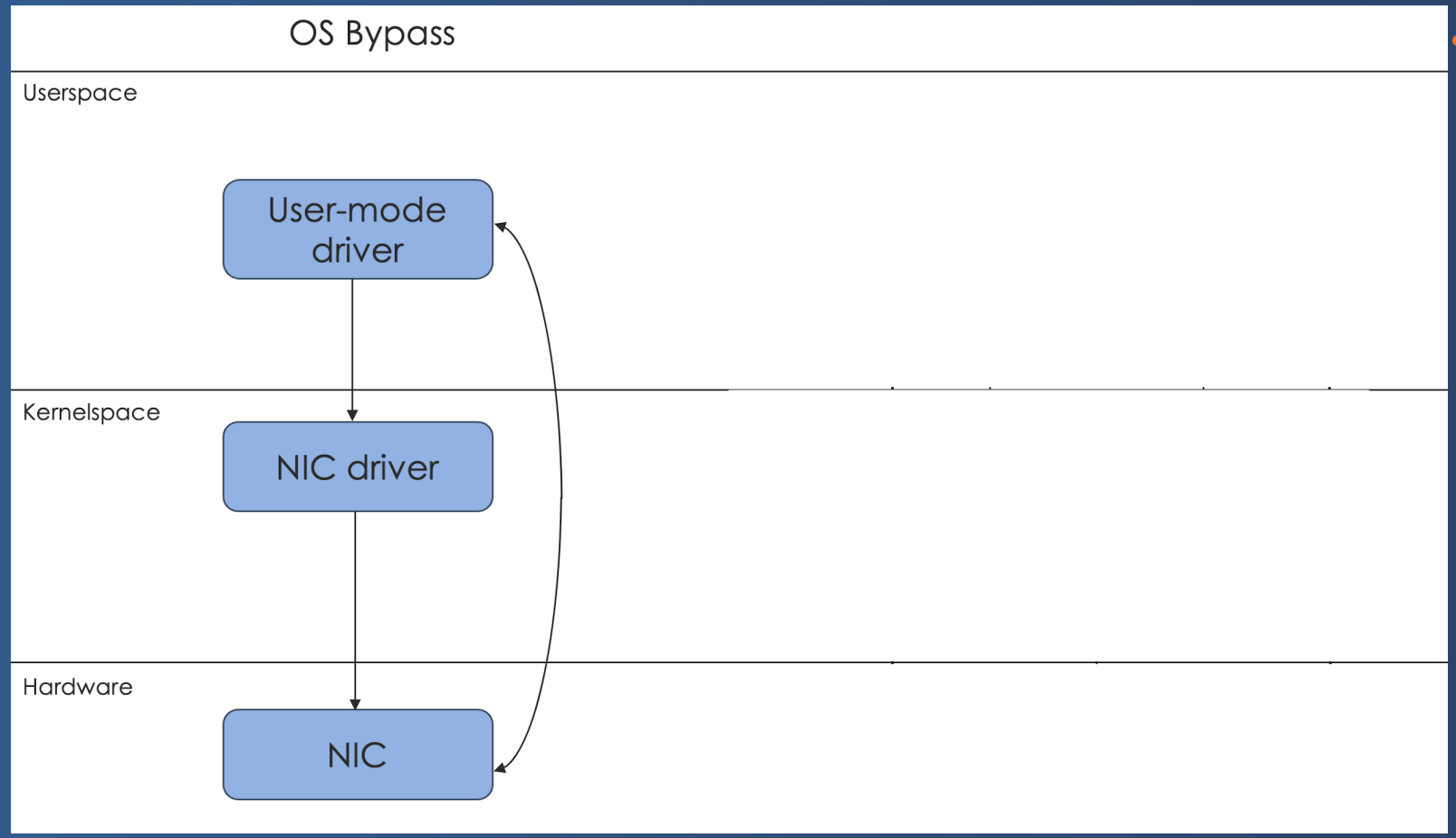
- One device file vs many device files
- Device firmware vs kernel driver
- High level APIs



SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Kernel bypass network drivers (DPDK, etc.)

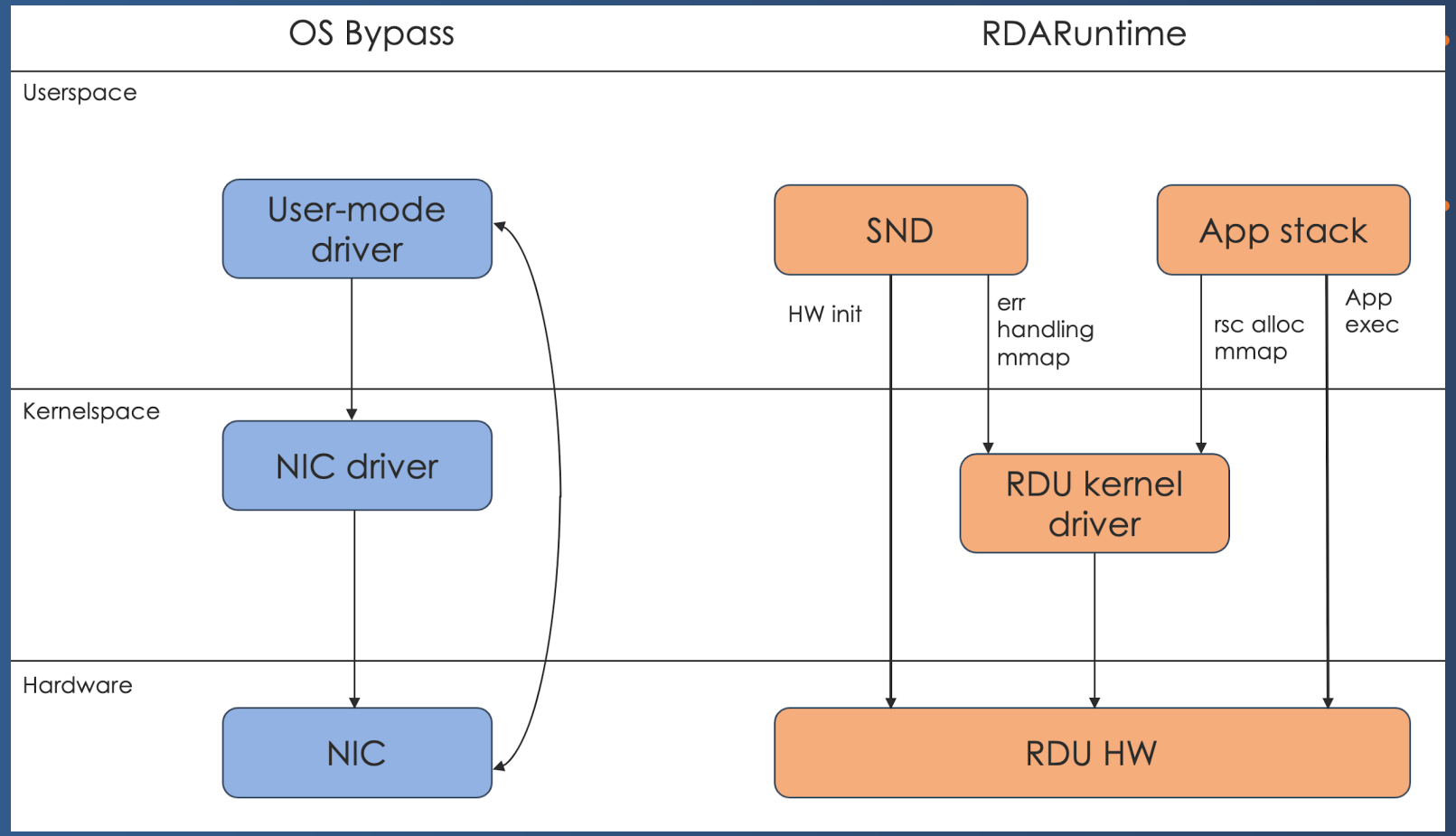


- Take the slower kernel out of the performance path

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Kernel bypass network drivers (DPDK, etc.)



- Take the slower kernel out of the performance path
- RDARuntime uses a hybrid approach
  - + Privileged tasks go in kernel
  - + Low-latency app create/destroy in userspace
    - One-time expensive setup/teardown of resource control structures the control path
    - Kernel-bypass path to execute applications directly via memory mapped control structures
  - + Interrupt steering, configuration, and system-wide structures live in kernelspace for convenience

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

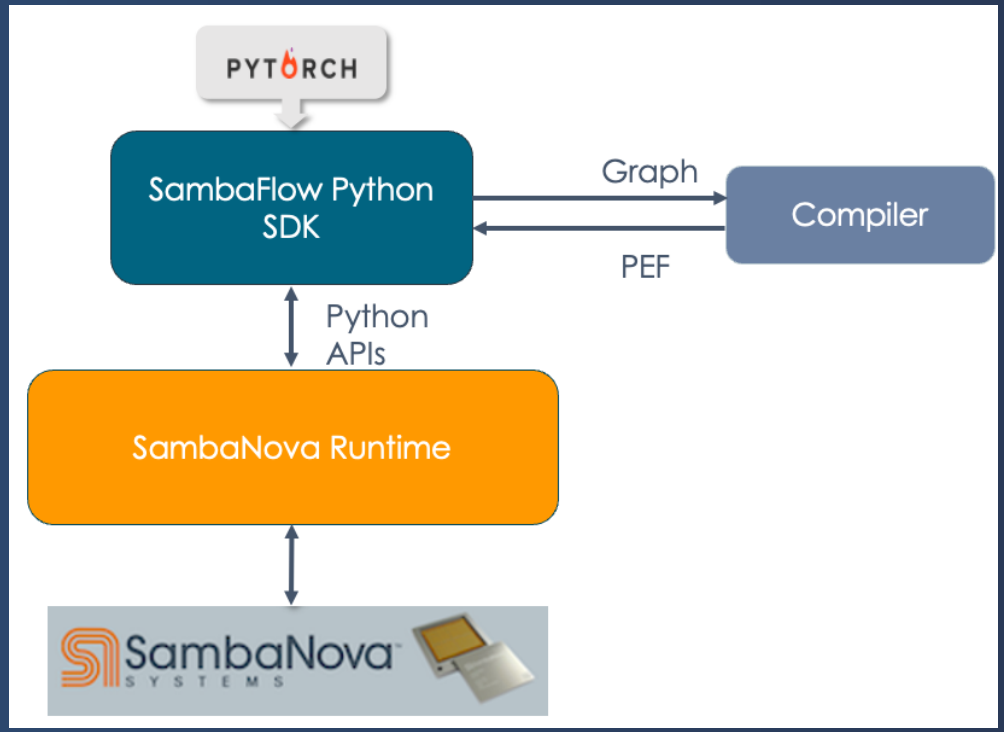
RDARuntime Architecture

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

User's Perspective

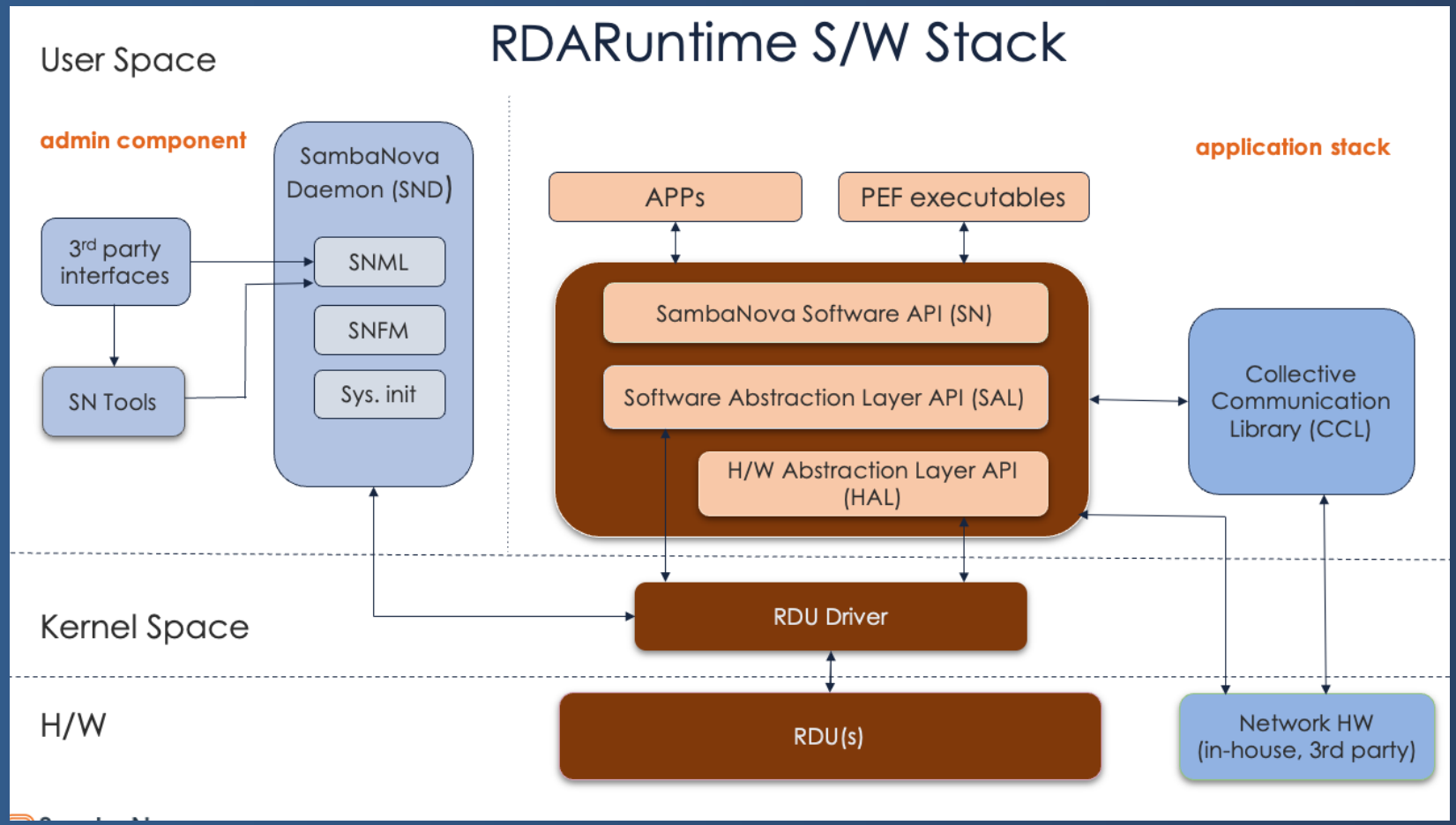
- User provides PyTorch graph
- Compiler processes the graph and creates PEF (Plasticine Executable Format)
- Application looks like:
  - PEF specifies RDU exec
  - Python code specifies CPU exec
- Runtime & Samba SDK orchestrates
  - Data movement
  - Graph exec
  - HW setup/destroy
  - rdu<->rdu<->host communication



SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

High level components

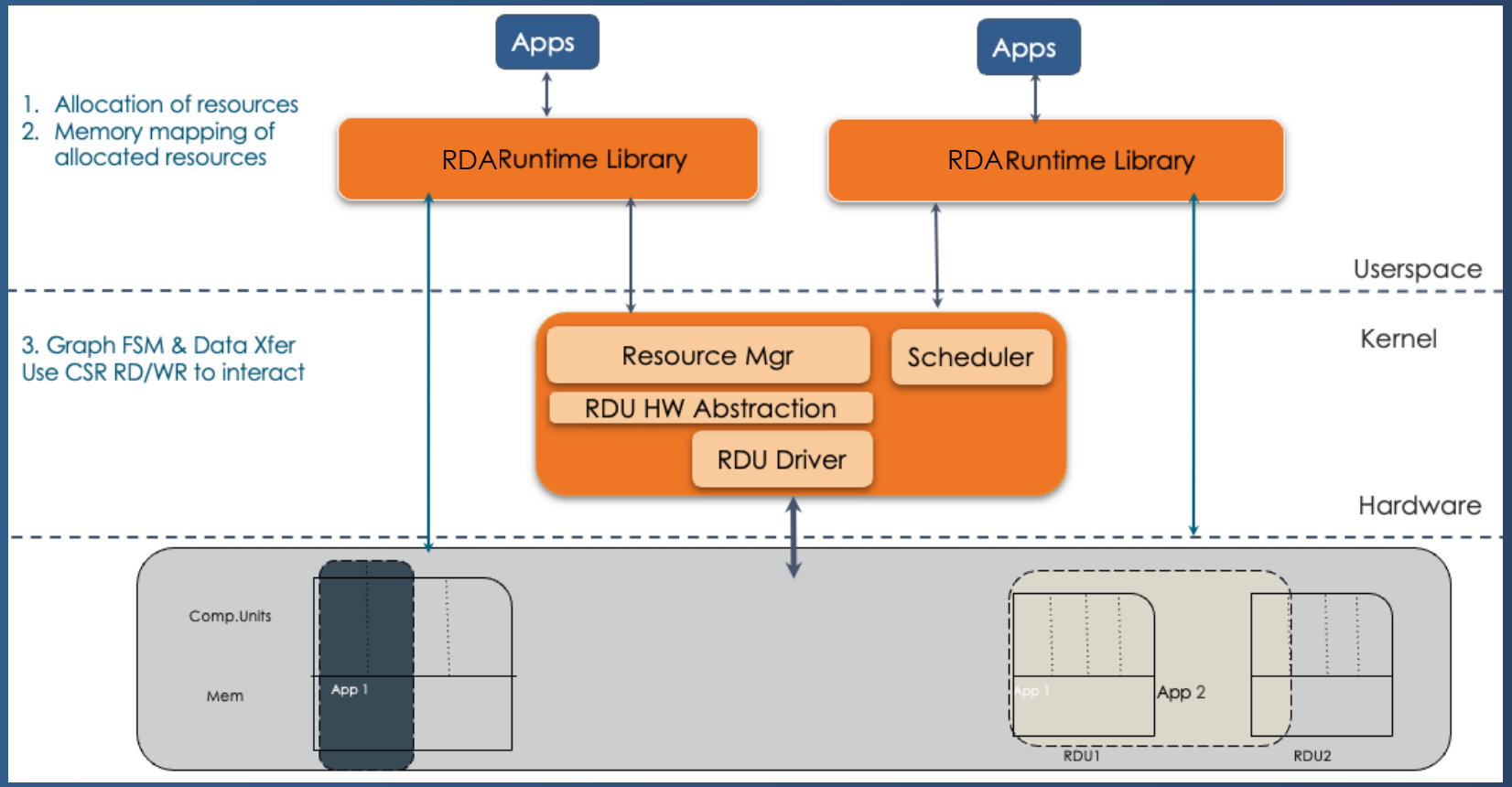


- Graph stack
  - HW and SW abstractions
  - Graph execution and data movement engine
  - Frontend APIs
  - CCL
- Admin section
  - SNML
  - System init
  - Fault management
- Kernelspace
  - Interrupt steering
  - Resource allocation
  - HW abstraction

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Multi-Tenancy



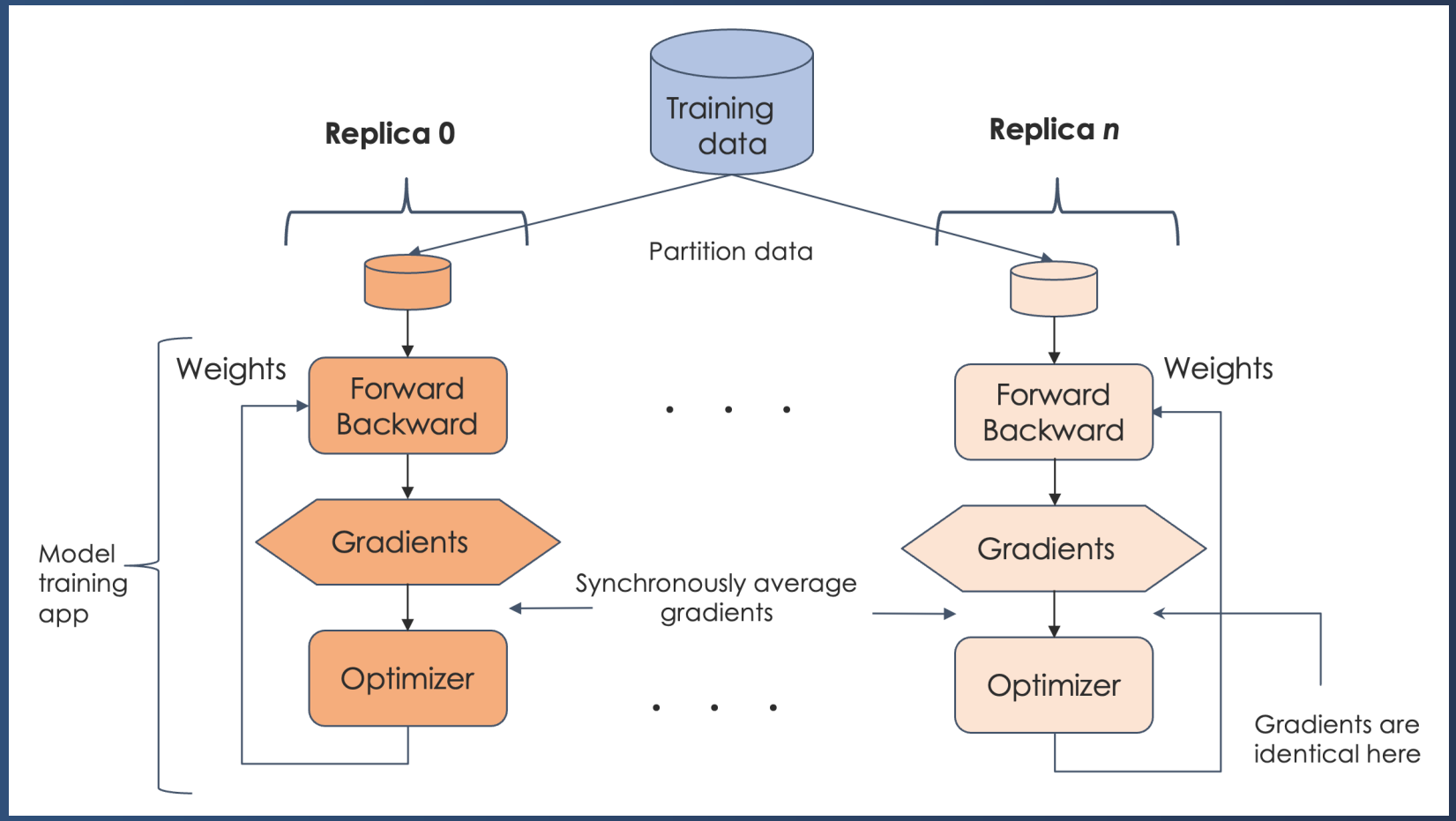
- Different users open the library
- All ring0 privileged tasks are handled in kernel
- App setup provisions resources
- At setup/teardown time, DMA map/unmap HW resources into userspace

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Scale-out: Data parallel

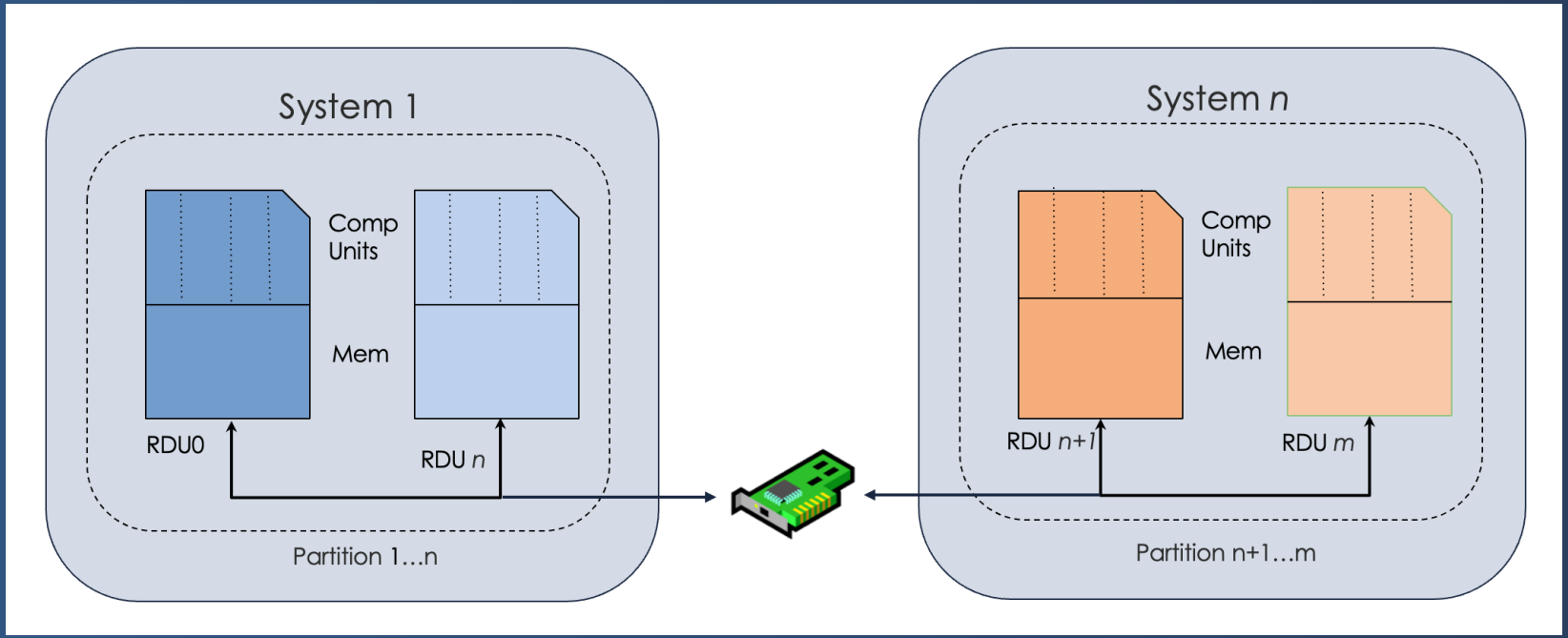
- Run replicas of a model in parallel
- Exchange progress after FW/ BW, before optimizer
- Parallel across minibatches
- Training only
- Communication is SW initiated and limited to gradient sync



SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Scale-out: Model parallel

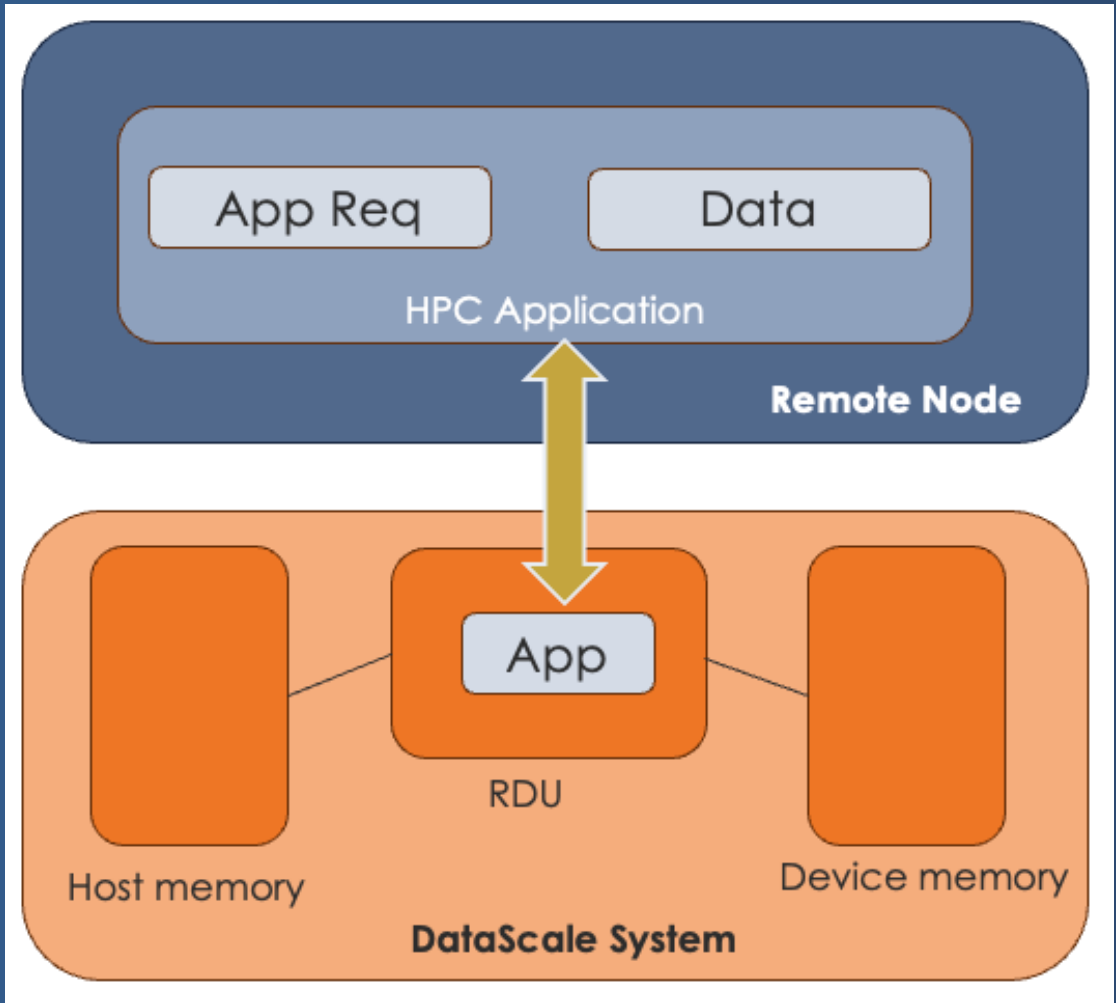


- Divide a model into synchronized pieces
- HW initiated data transfers are not limited to any particular step
- More flexible
- More difficult to map with a compiler
- More dependent on physical compute HW than DP

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Disaggregated RDUs: Remote Execution



- HPC app runs ML inference or periodic training throughout its execution
  - Common for "in the loop" AI guided simulations
- CPU/GPU parts of the app run on a separate HPC node
- Target RDUs via a preconfigured app server running in RDARuntime

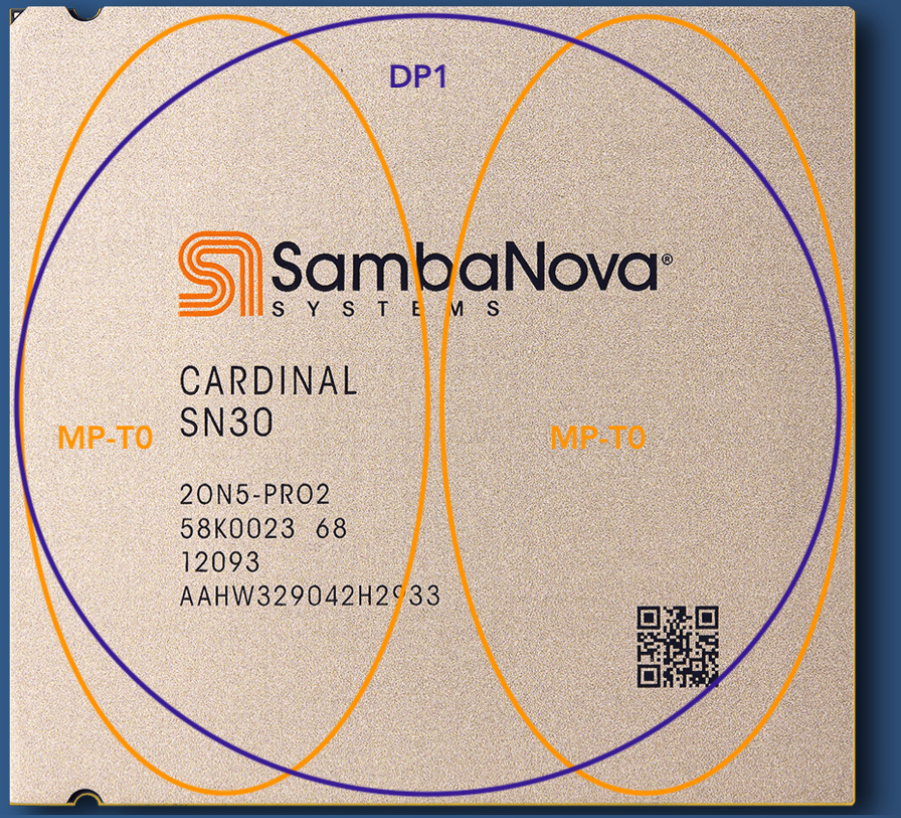
SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Experimental Analysis

Copyright © 2023 SambaNova Systems

Scaling Study Experimental Setup - Chip Level



- DP + MP training
- Per chip:
  - 2 threads on 4 tiles each
  - MP mode
- Each 2-way MP group acts as a DP replica
- Most efficient way to map due to I/O hierarchy
  - per chip, 2 groups of 4 tiles with higher internal bandwidth

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

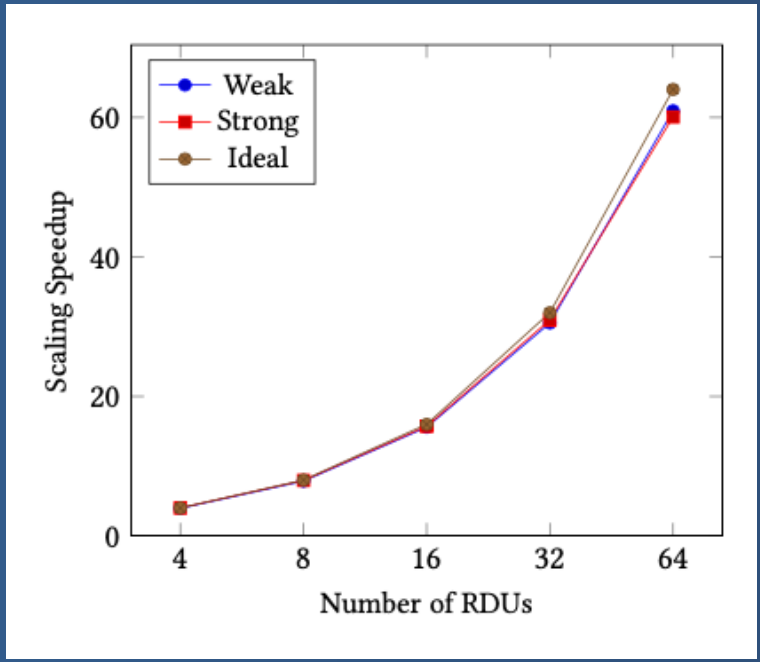
Scaling Study Experimental Setup - Rack Level

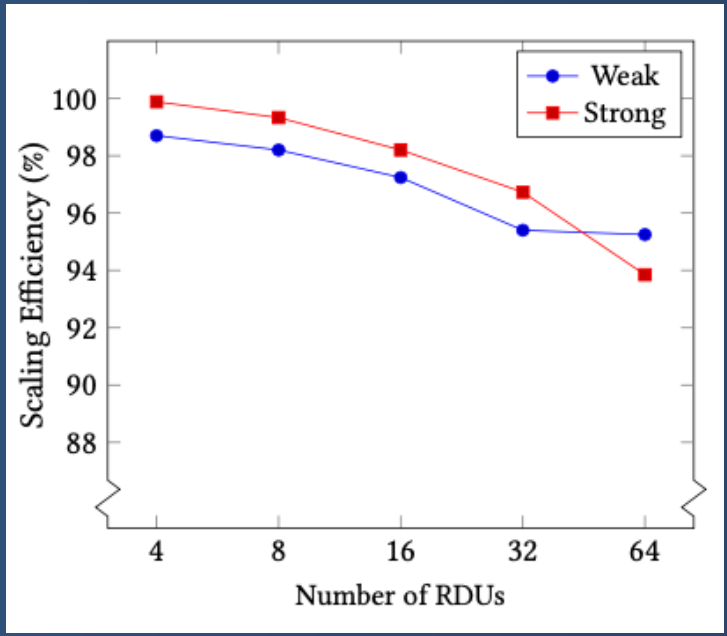
- Per rack: 2 systems
- Per system: 8 RDUs, 1 x86 host
  - Connected by internal fabric
- 1-64 way DP training
  - Using 2-way MP per replica
  - 400G Ethernet + RDMA (RoCE) across hosts
- 4 racks

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Scaling Study Results





- GPT 13.5B parameter model
- Training
- Used global and per-chip batch size to vary amount of work
  - Global & per worker range: 128-8192
- Efficiency:
  - weak: 95.3% at 64 RDU
  - strong: 93.8% at 64 RDU
- Full numerical results in paper appendix and on extra slide

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Latency Profiling

Summary:

- Logistic regression
- 10,000 iterations
- Batch size 1
- Measured with SW timers

Python + Python-to-C:

- Pytorch app setup
- Data prep
- CPU part of app
- Time in Pybind

Data Xfer and Conv

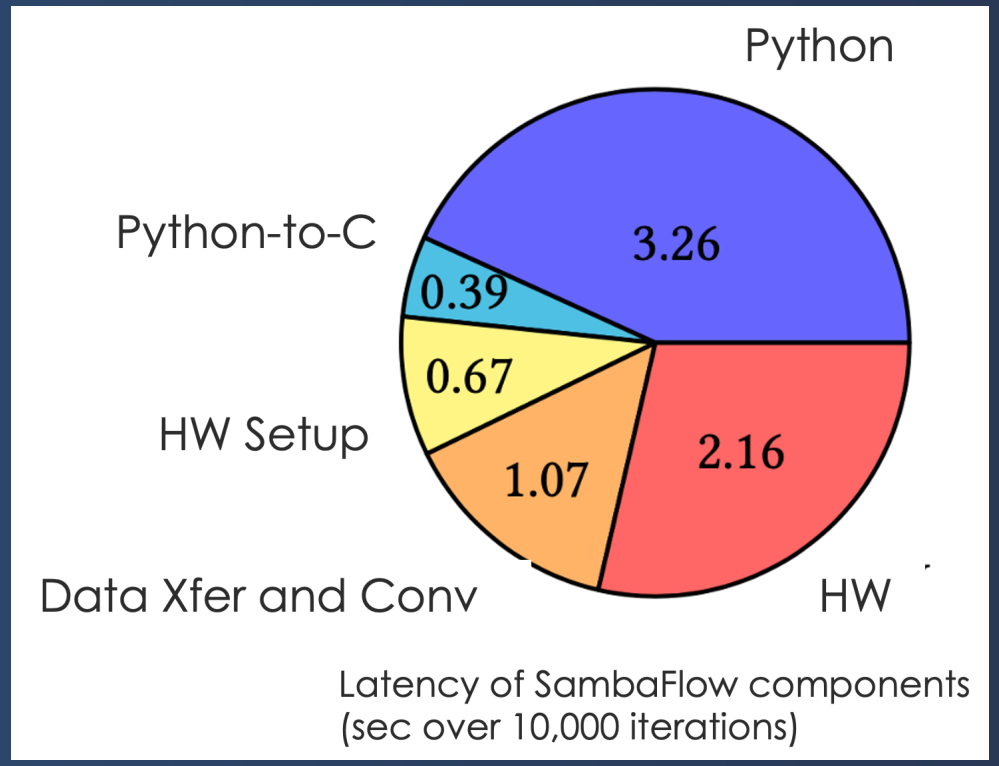
- Layout, data type, and MOrder changes
- Transfer to RDU via PCIe

HW setup

- Register programming
- Section execution/swaps

HW

- Time with RDU compute running



Latency of SambaFlow components (sec over 10,000 iterations)

This logreg model is extremely small (doesn't require a lot of FLOPS) and was run only for the purpose of collecting runtime latency

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Future work

- More details about distributed learning
- More comprehensive scaling and latency studies
- "Deep dive" into some of the components mentioned here
- SN40 poses new challenges:
  - Memory and I/O hierarchy
  - More flexible scaling

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Acknowledgments

- RDARuntime team (https://www.glick.cloud/rdaruntime-contributors)
- Blaine Rister, Fansheng Cheng, Greg Dykema (editors)
- Neal Sanghvi (figure assistance)

SambaNova SYSTEMS

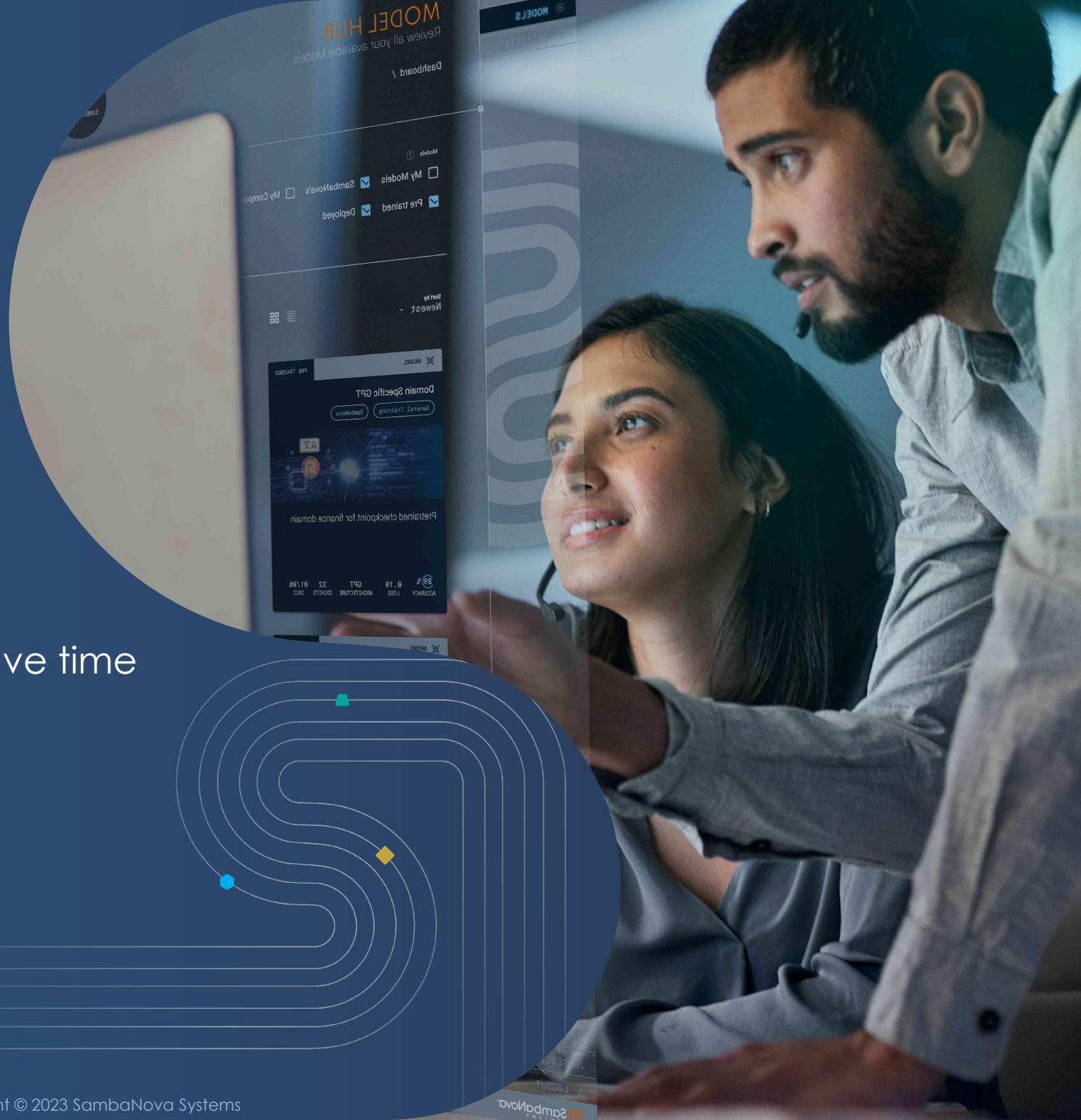
Copyright © 2023 SambaNova Systems

SambaNova SYSTEMS

Thank you

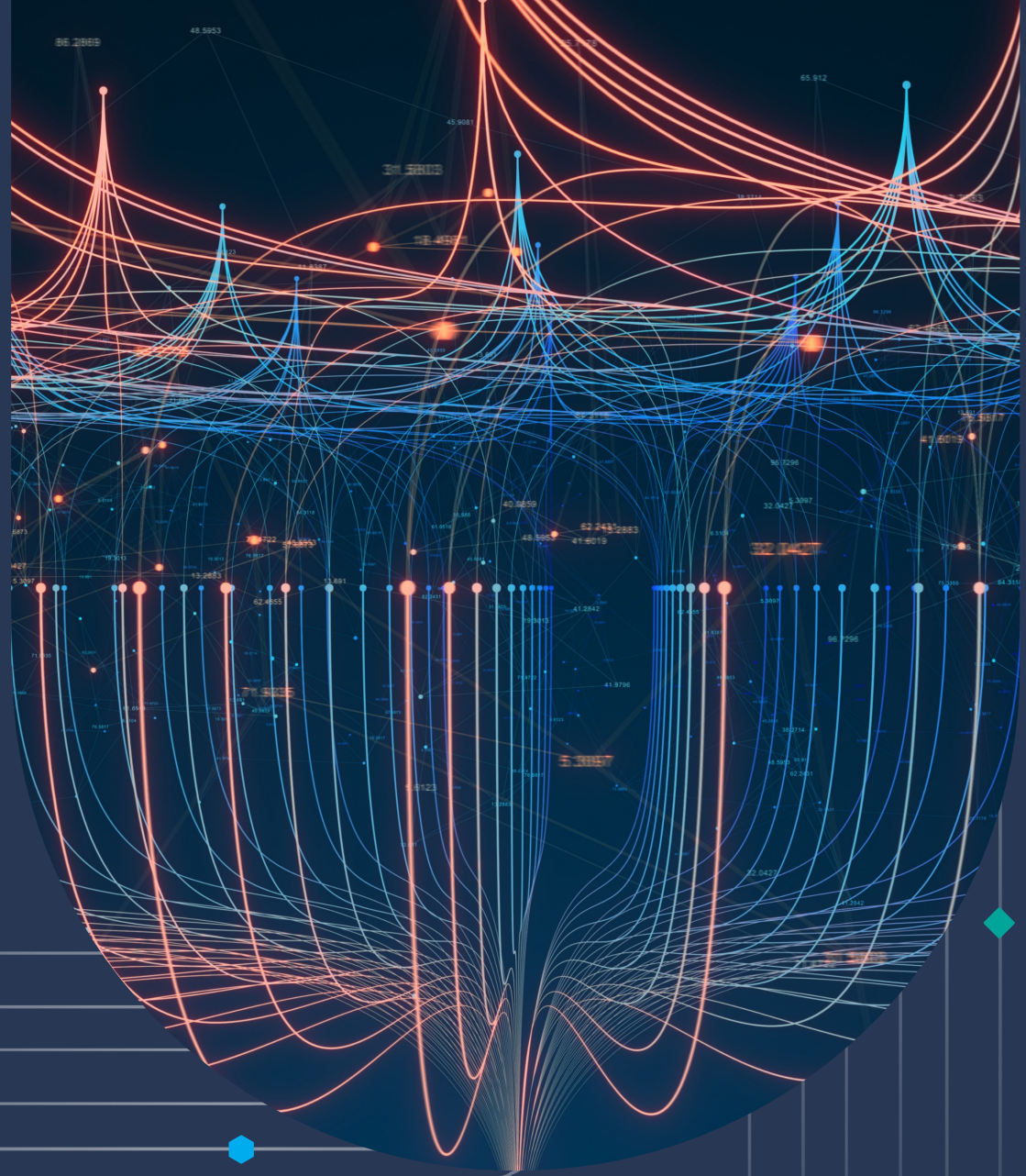
benjamin.glick@sambanova.ai

Come to booth #681 for questions we don't have time to answer here :)



Copyright © 2023 SambaNova Systems

Appendices



SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Full scaling data

RDUs	Per-worker Batch Size	Global Batch Size	E2E Time (sec)	Throughput (samples/sec)	Strong-scaling Speedup	Strong-scaling Efficiency
1	8192	8192	291505.40	2877.68	1.00	100.00%
4	2048	8192	72962.69	11497.12	3.99	99.88%
8	1024	8192	36685.74	22866.13	7.95	99.33%
16	512	8192	18553.87	45212.17	15.71	98.20%
32	256	8192	9417.76	89072.26	30.95	96.73%
64	128	8192	4853.36	172841.15	60.06	93.85%

Table 1: Strong scaling study raw data

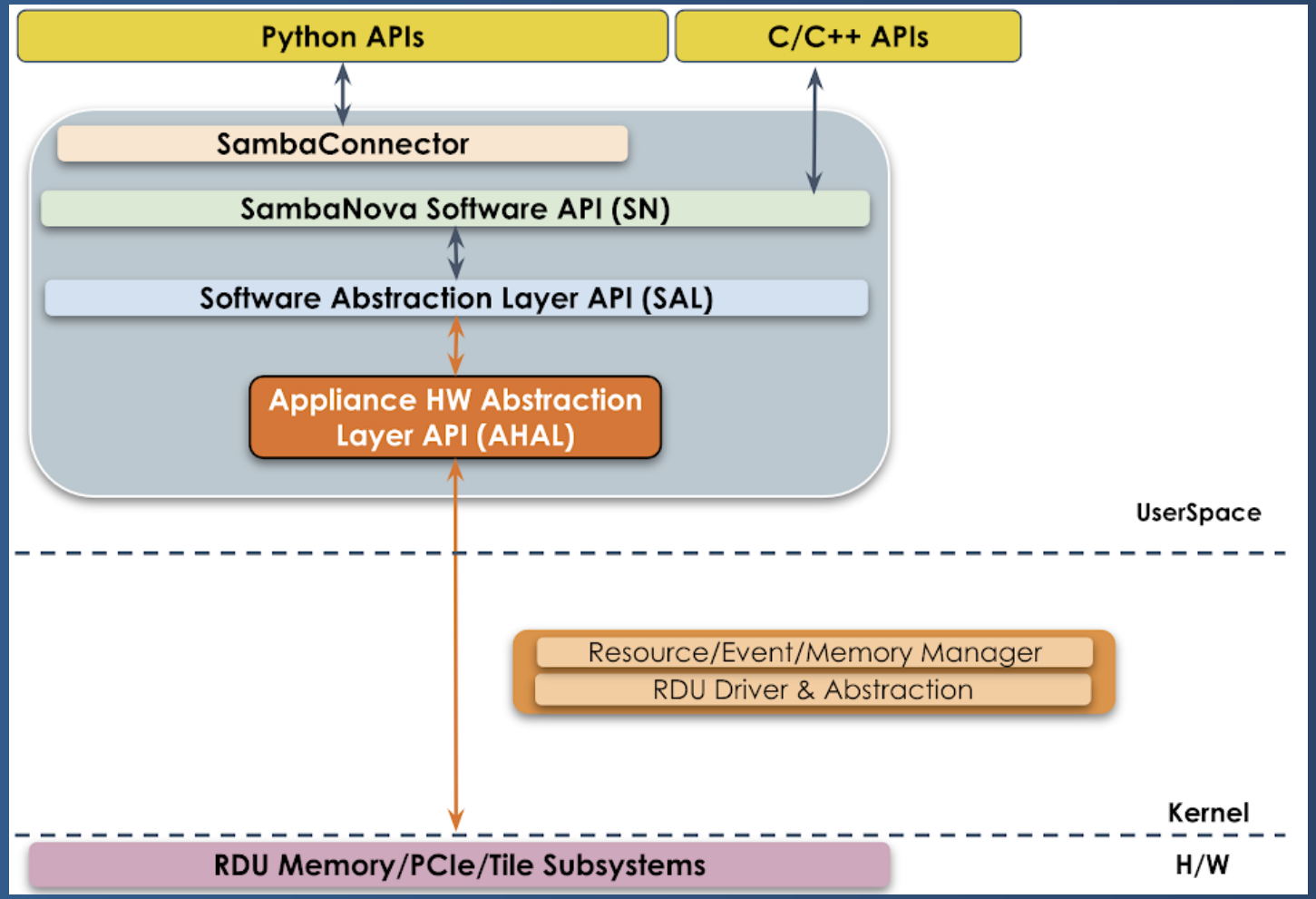
RDUs	Per-worker Batch Size	Global Batch Size	E2E Time (sec)	Throughput	Weak-scaling Speedup	Weak-scaling Efficiency
1	128	128	4649.97	2818.77	1	100.00%
4	128	512	4709.86	11131.70	3.95	98.73%
8	128	1024	4733.56	22151.95	7.86	98.23%
16	128	2048	4781.85	43856.46	15.56	97.24%
32	128	4096	4874.28	86049.74	30.53	95.40%
64	128	8192	4881.88	171831.50	60.95	95.25%

Table 2: Weak Scaling study raw data

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Cloud vs Appliance: Appliance



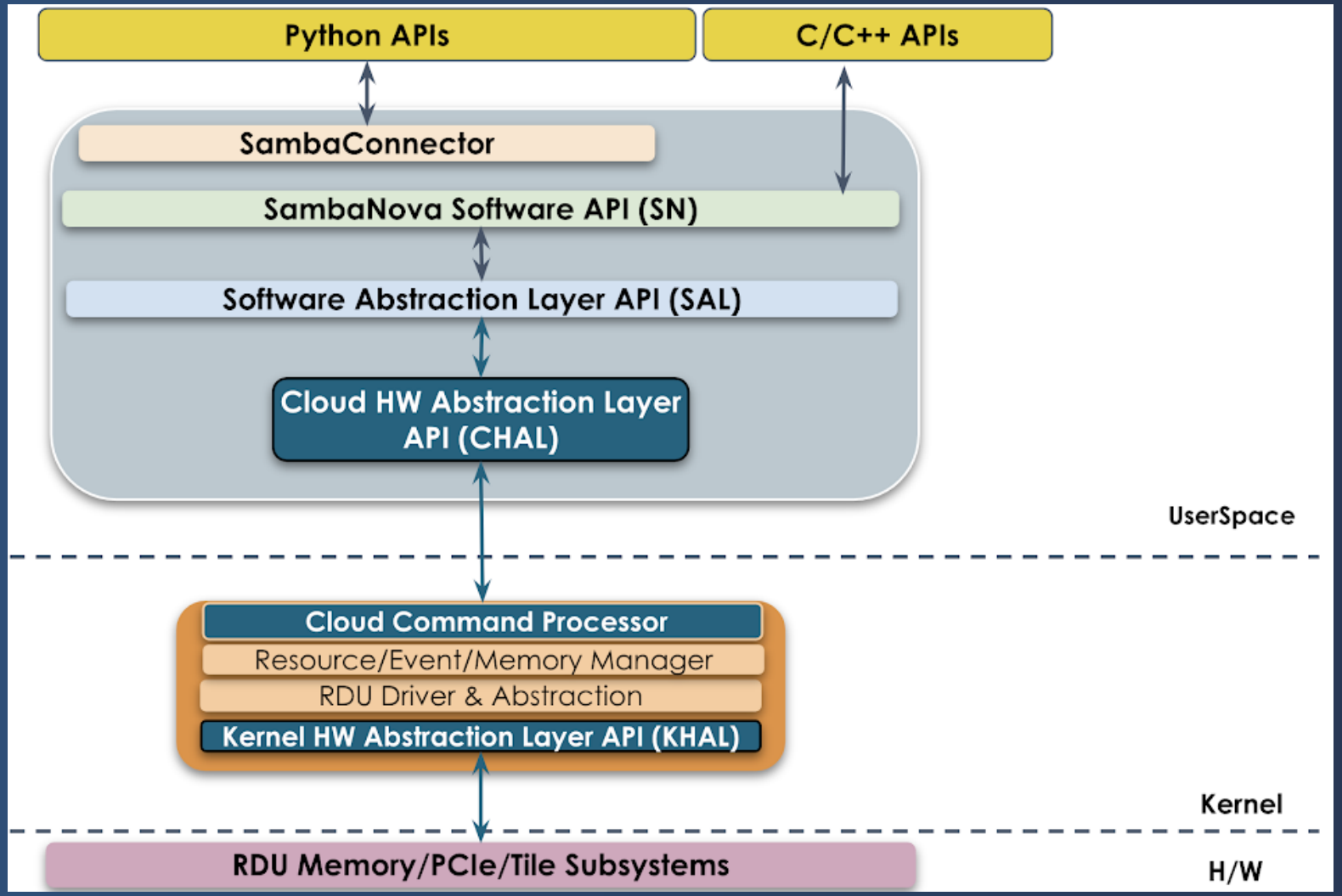
- Low latency mode
- App create: map HW resources into userspace
- Selectively map resources based on allocated resources
- HAL layer directly accesses RDU resources from user process after map finishes
- After resources are mapped, security checking doesn't happen until context destroy

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

Cloud vs Appliance: Cloud

- Elevated isolation mode
- Direct I/O to device is disabled in cloud mode
- All app control/HW access moves to kernelspace
- Cloud command processor implements a work queue
- Client/server model - server in kernelspace validates each request for security



SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems

SambaNova SYSTEMS

Copyright © 2023 SambaNova Systems