S. Di Girolamo, A. Kurth, P. Xu, A. Calotoiu, T. Benz, T. Schneider, J. Beránek, L. Benini, M. Khalilov, … T. Hoefler

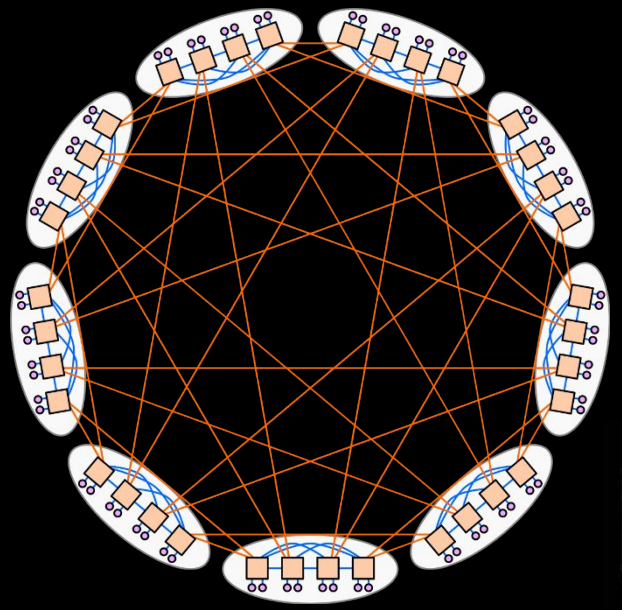
sPIN: in-network high-performance low-power packet processing

ISC 2023

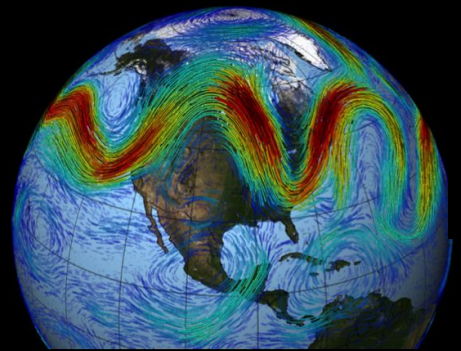
NETFLIX

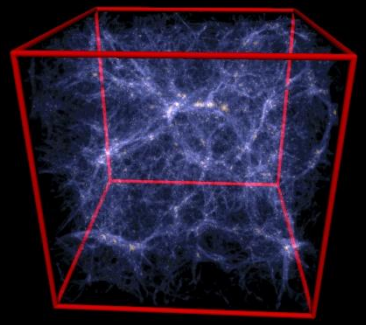
zoom

Low latency,
high throughput

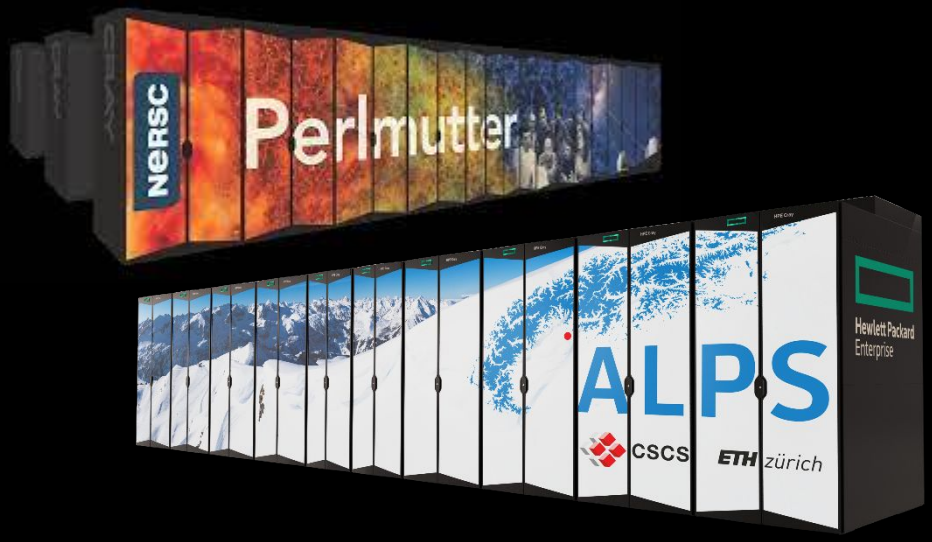


Workloads

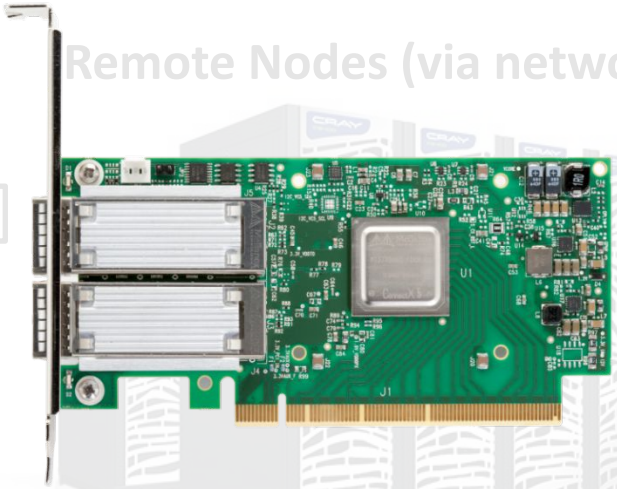




Systems



Data Processing in modern RDMA networks



Mellanox Connect-X5: 1 packet/5ns

Mellanox Connect-X7 (400G): 1 packet/1.2ns

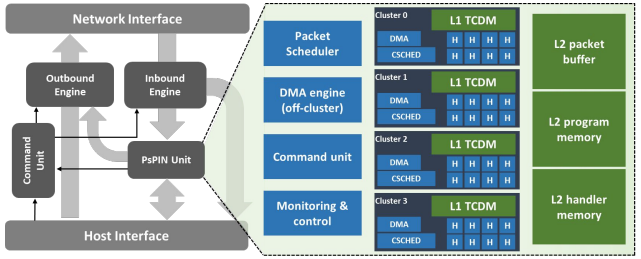
Define the principles an in-network compute solution like sPIN should follow

Low latency, full throughput

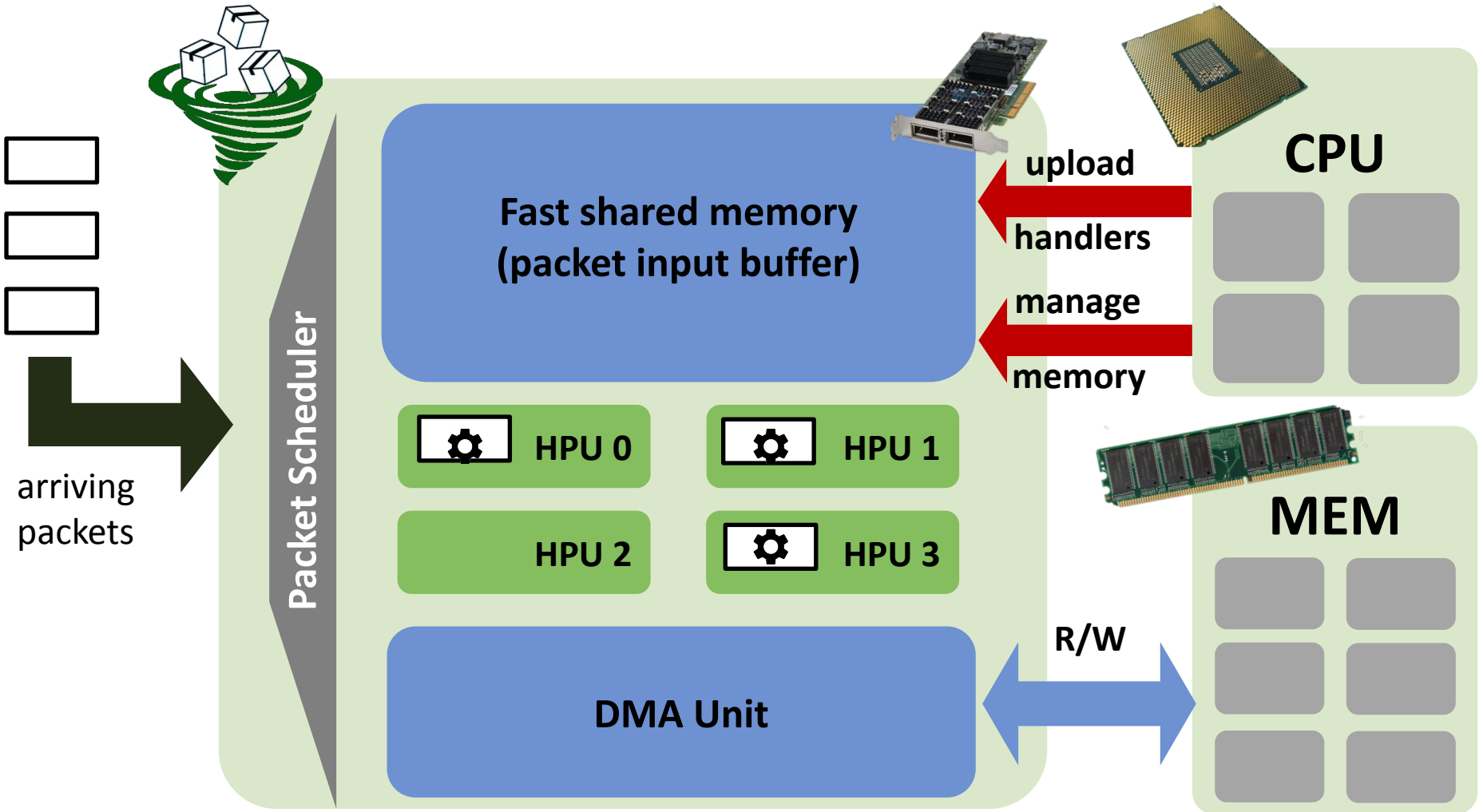
Support for wide range of use cases

Easy to integrate

Introduce an open-source in-NIC accelerator implementing the sPIN programming model

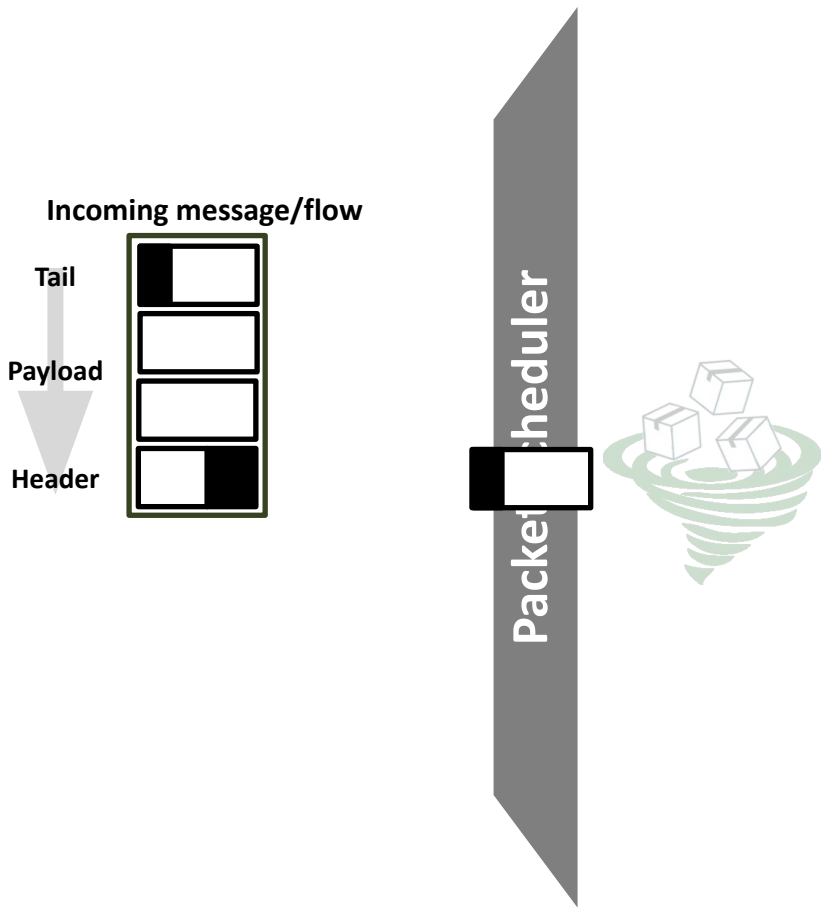


sPIN NIC - Abstract Machine Model



Hoefler, T., Di Girolamo, S., Taranov, K., Grant, R. E., & Brightwell, R. (2017, November). sPIN: High-performance streaming Processing in the Network. SC` 17

sPIN – Programming Interface



Header handler

```
int hh(const handler_args_t * args){
    return 0;
}
```

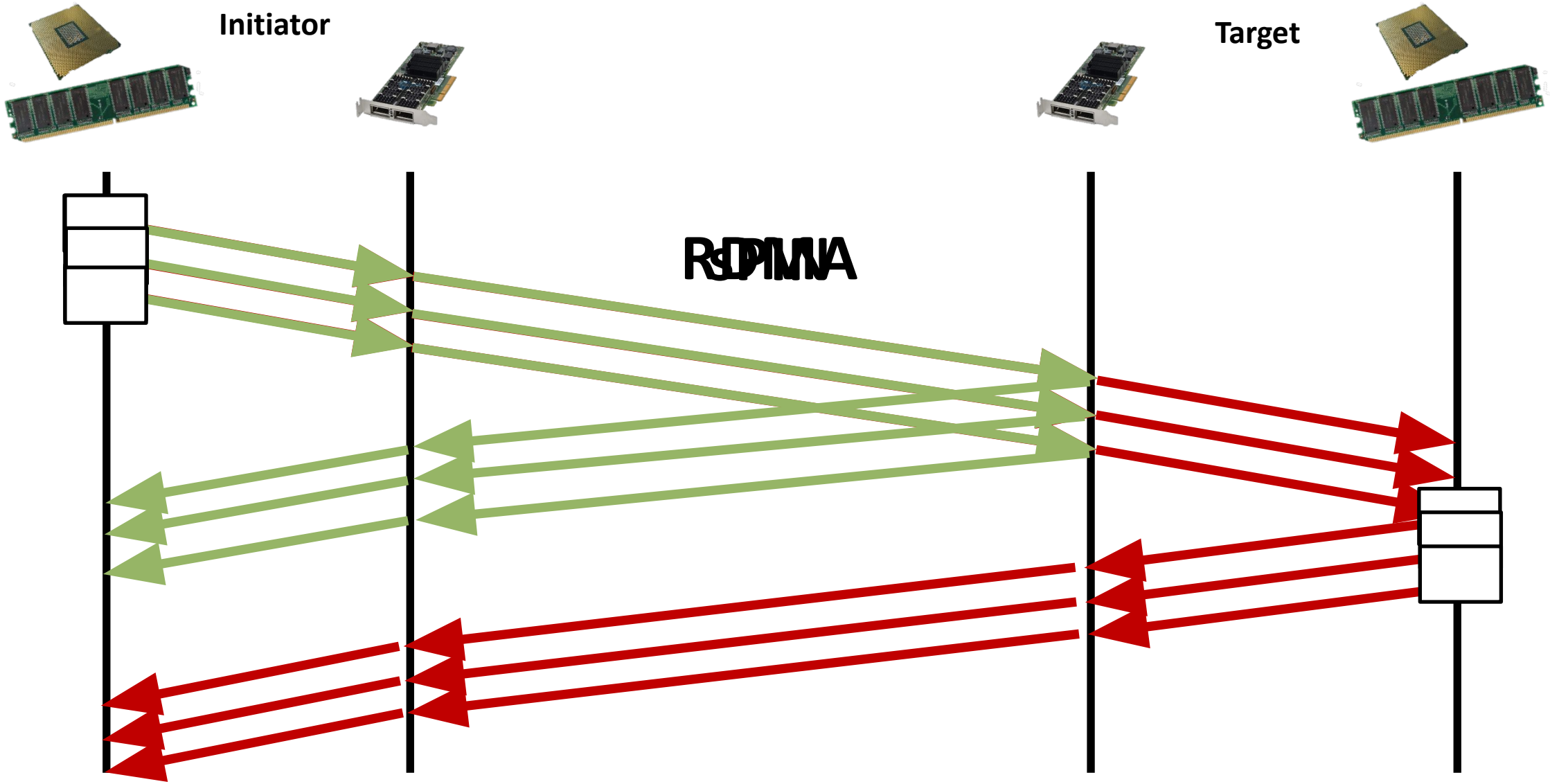
Payload handler

```
int ph (const handler_args_t * args){
   spin pkt send(args->pkt_pld, 0, args->source,
args->pkt_offset,
          args->rlength, args->pkt_pld_len);
    return 0;
}
```

Completion handler

```
int th (const handler_args_t * args)
{
    return 0;
}
```

RDMA vs. sPIN in action: Streaming Ping Pong



Architectural principles for in-network compute

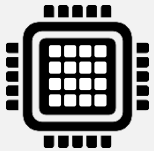
Low latency, full throughput

Support for wide range of use cases

Easy to integrate

Architectural principles for in-network compute

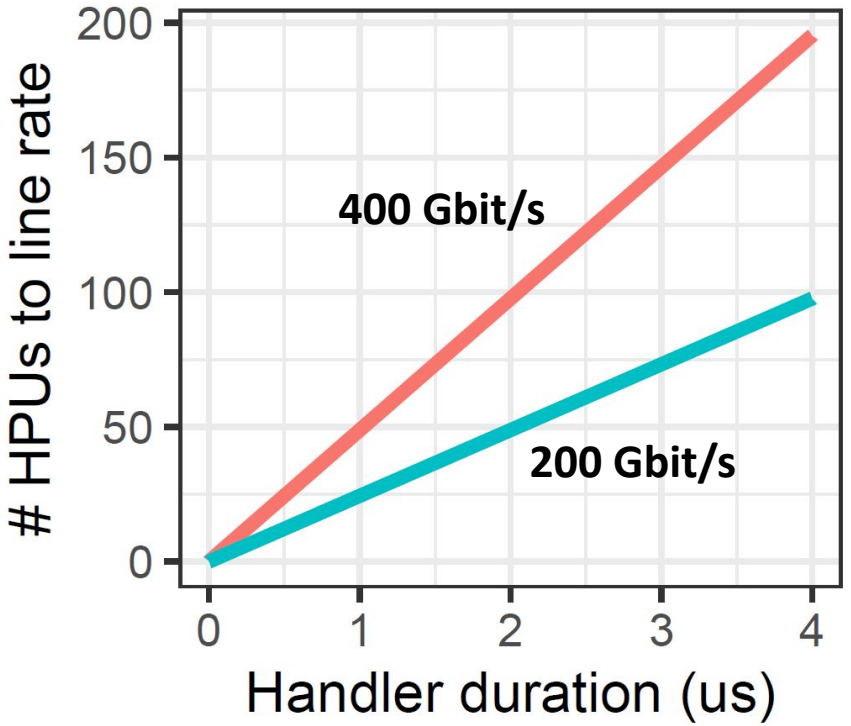
Low latency, full throughput



Highly parallel

Fast scheduling

Fast explicit
memory access

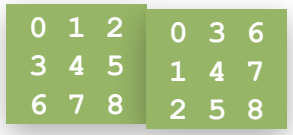


Architectural principles for in-network compute

Support for wide range of use cases

Stateful computation support

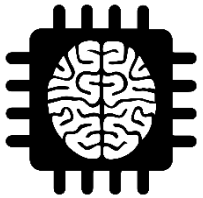
Handlers isolation



Network-accelerated datatypes [1]

sPIN-FS

Zoo-sPINNER
consensus protocols



Quantization
Allreduce and other collectives

Packet classification and pattern matching

Serverless

Erasure coding

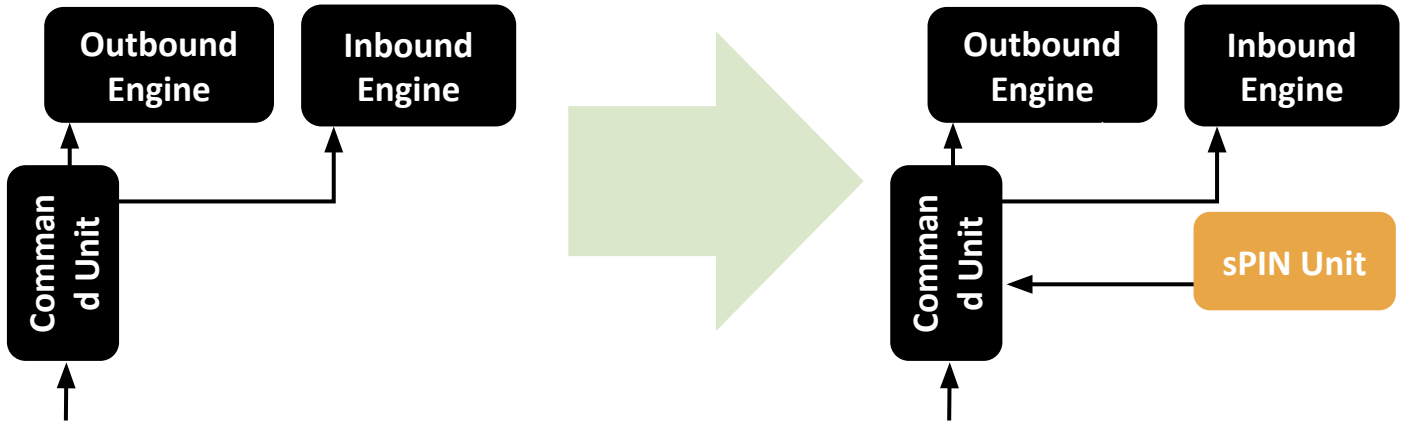
[1] Di Girolamo et al., "Network-Accelerated Non-Contiguous Memory Transfers", SC 2019, https://arxiv.org/abs/1908.08590

Architectural principles for in-network compute

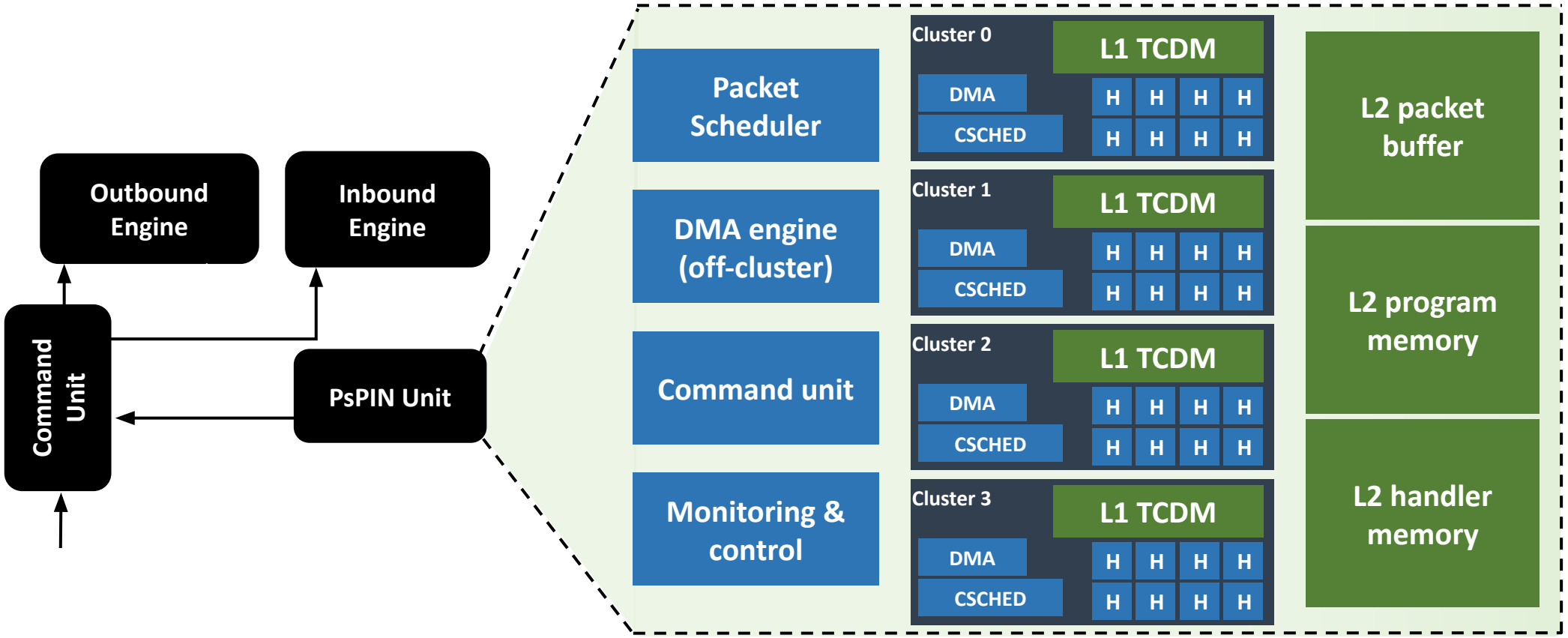
Easy to integrate

Area and power efficiency

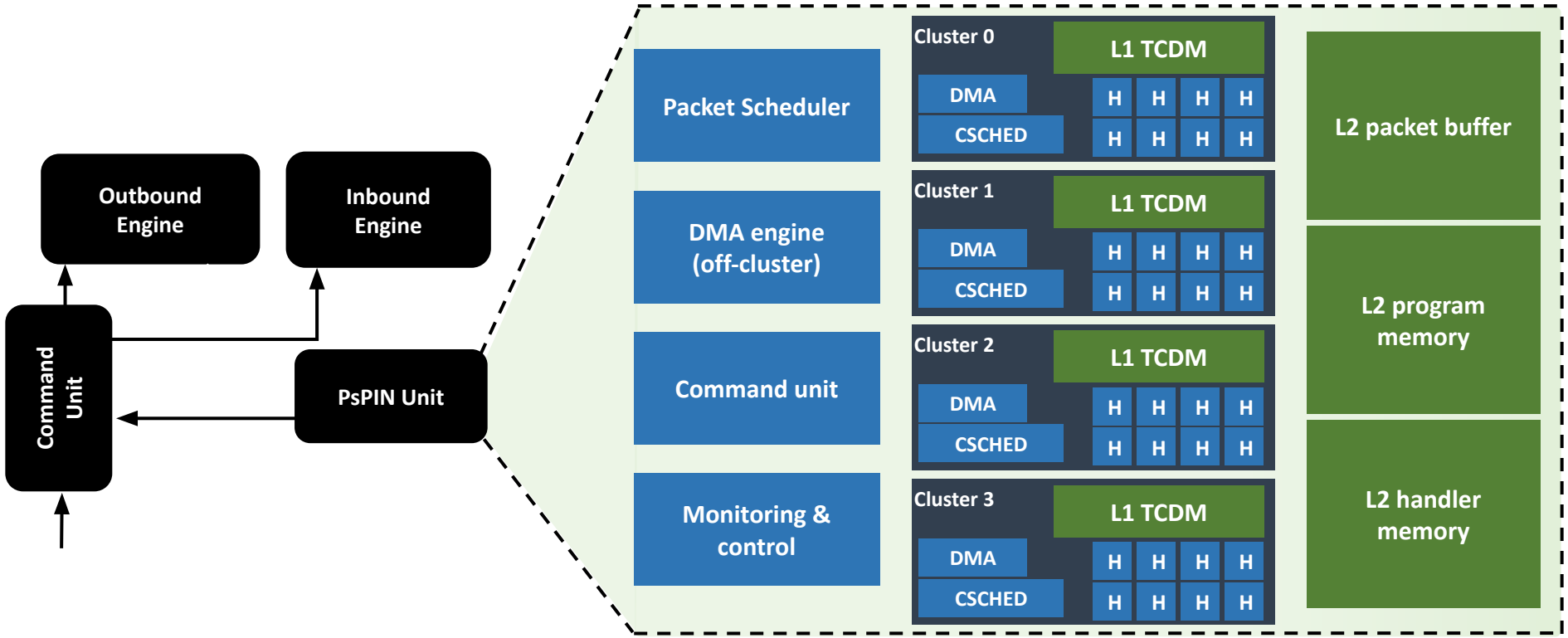
Configurability



PsPIN: A PULP-powered implementation of sPIN



PsPIN: A PULP-powered implementation of sPIN

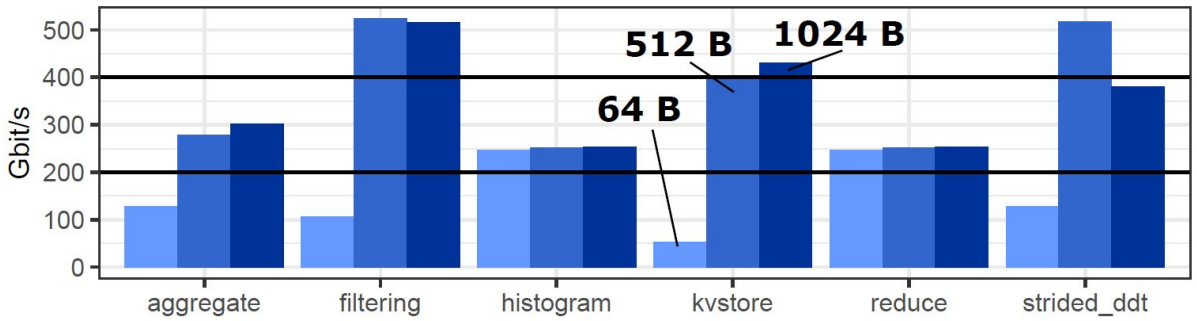


GF 22nm FDSOI @ 1GHz

Area:
95 MGE (18.5 mm2)

Power:
6.1 W (98% dynamic power)

Component	Area (mm^2) Unit	Total	Perc.	Power (W) Unit	Total	Perc.
PsPIN	18.47	18.47	100.0%	6.08	6.08	100.0%
↳ **L2 memories** (×1)	9.48	9.48	51.3%	1.09	1.10	18.1%
↳ **Interconnect** (×1)	0.57	0.57	3.0%	0.71	0.71	11.7%
↳ **Cluster** (×4)	1.99	7.95	43.0%	0.94	3.77	62.0%
↳ **L1** (×1)	1.65	1.65	82.9%	0.52	0.52	55.3%
↳ **Core** (×8)	0.01	0.08	4.0%	0.02	0.14	15.3%
↳ **Instr. cache** (×1)	0.08	0.08	4.0%	0.14	0.14	15.1%
↳ **Interconnect** (×1)	0.06	0.06	3.0%	0.11	0.11	11.3%



Scheduling overhead:
- 64 B packets: 12 ns
- 1 KiB packets: 26 ns

Application perspective

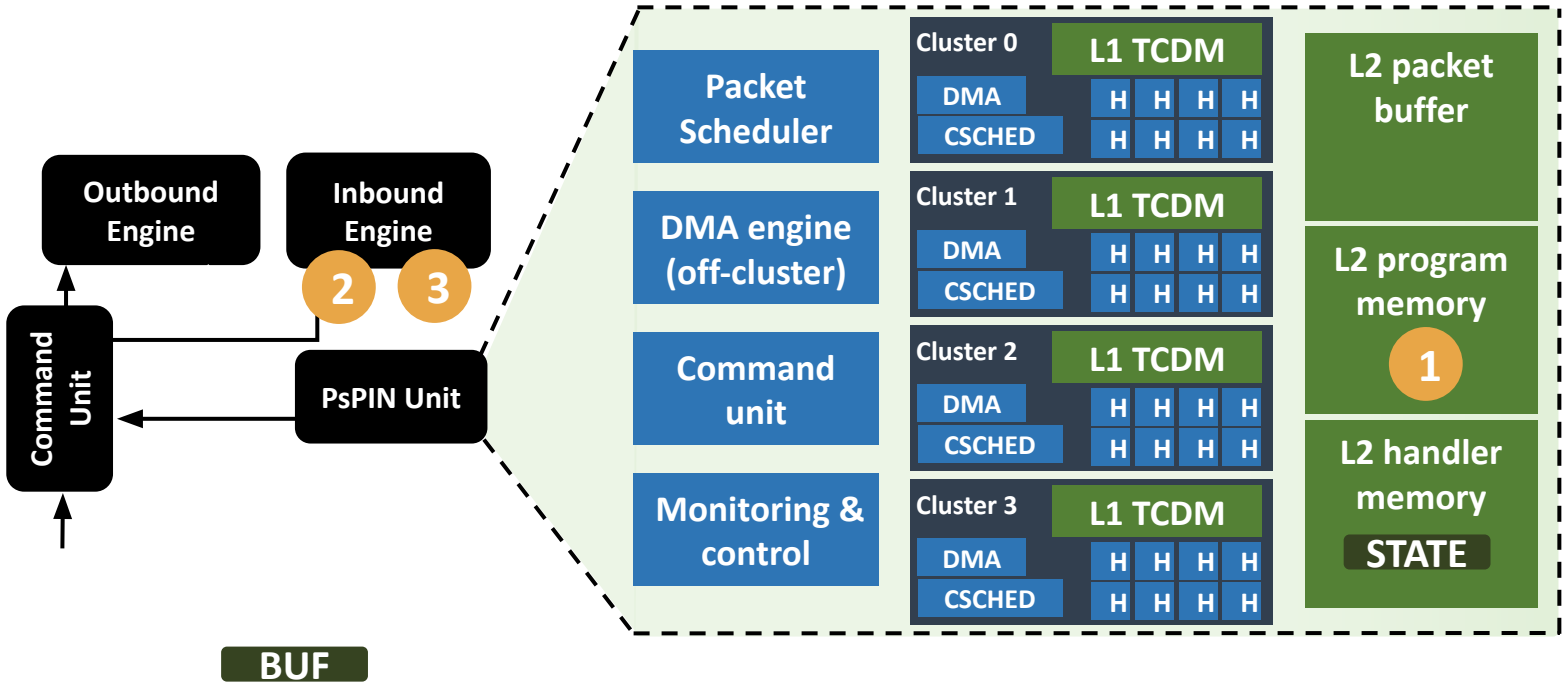
1 Define and offload handlers

Telemetry:
telemetry_hh(), telemetry_ph(), telemetry_th();
Filtering:
filter_hh(), filter_ph, filter_th();

2 Define an execution context

Execution context:
EC_filter
filter_hh(), filter_ph(),
filter_th();
NIC memory: STATE
Host buffer: BUF

3 Define matching rule:
e.g., (IP packets) -> EC_filter



Network perspective

1. Match packet to execution context
   e.g., (IP packets) -> EC_filter

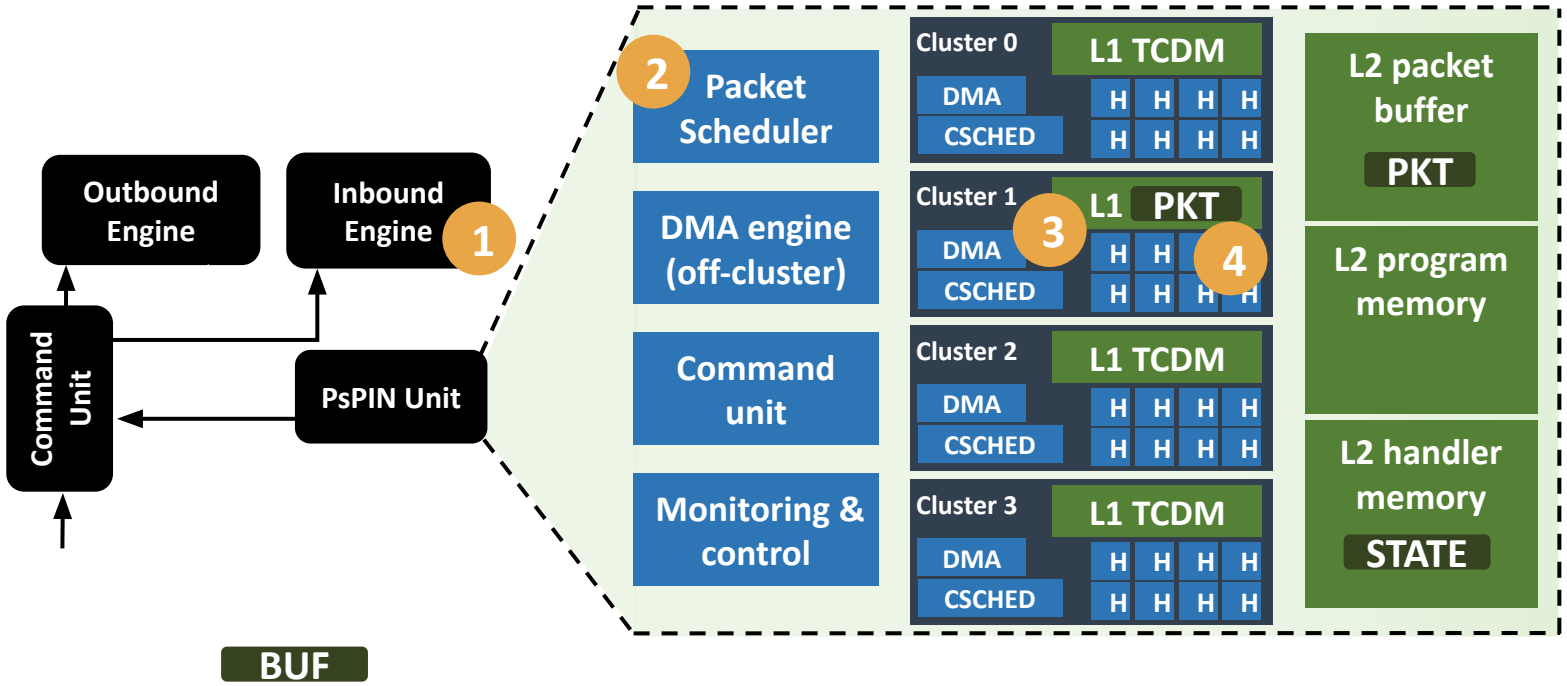
2. Write PKT to L2 pkt buffer and inform PsPIN of the new packet to process

3. Schedule the packet to a cluster (task: pkt pointer, handler fun)

4. Copy packet to L1 and run the handler

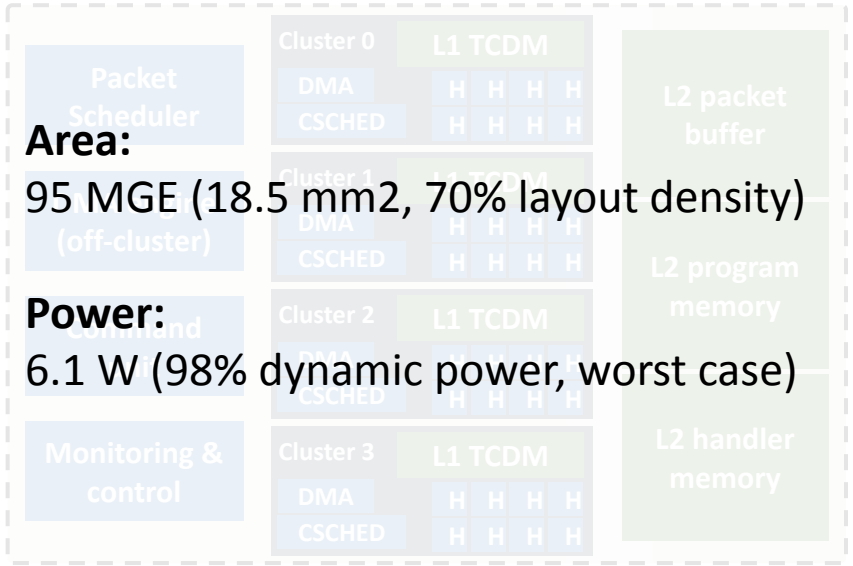
Execution context:
EC_filter
filter_hh(), filter_ph(), filter_th();
NIC memory: STATE BUF
Host buffer:

Scheduling overhead:
- 64 B packets: 12 ns
- 1 KiB packets: 26 ns



Circuit Complexity and Power Estimations

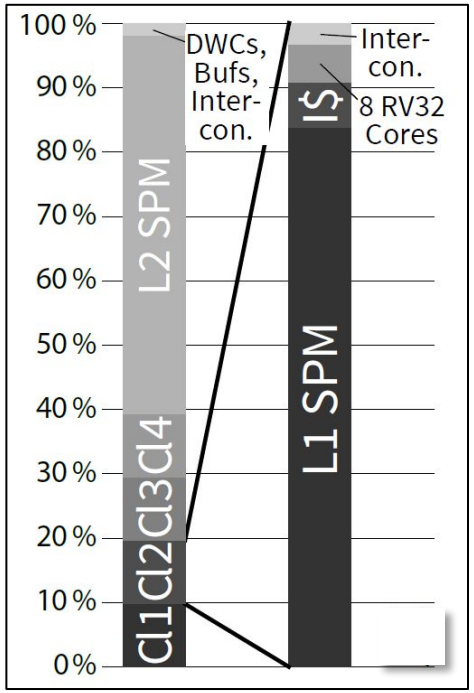
GlobalFoundries 22nm FDSOI @ 1GHz

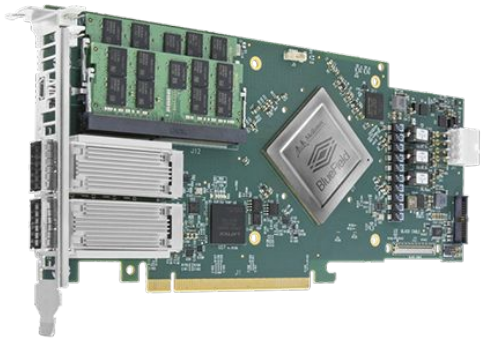


Area:
95 MGE (18.5 mm2, 70% layout density)

Power:
6.1 W (98% dynamic power, worst case)

Component	Area (mm^2) Unit	Total	Perc.	Power (W) Unit	Total	Perc.
PsPIN	18.47	18.47	100.0%	6.08	6.08	100.0%
↳ **L2 memories** (×1)	9.48	9.48	51.3%	1.09	1.10	18.1%
↳ **Interconnect** (×1)	0.57	0.57	3.0%	0.71	0.71	11.7%
↳ **Cluster** (×4)	1.99	7.95	43.0%	0.94	3.77	62.0%
↳ **L1** (×1)	1.65	1.65	82.9%	0.52	0.52	55.3%
↳ **Core** (×8)	0.01	0.08	4.0%	0.02	0.14	15.3%
↳ **Instr. cache** (×1)	0.08	0.08	4.0%	0.14	0.14	15.1%
↳ **Interconnect** (×1)	0.06	0.06	3.0%	0.11	0.11	11.3%

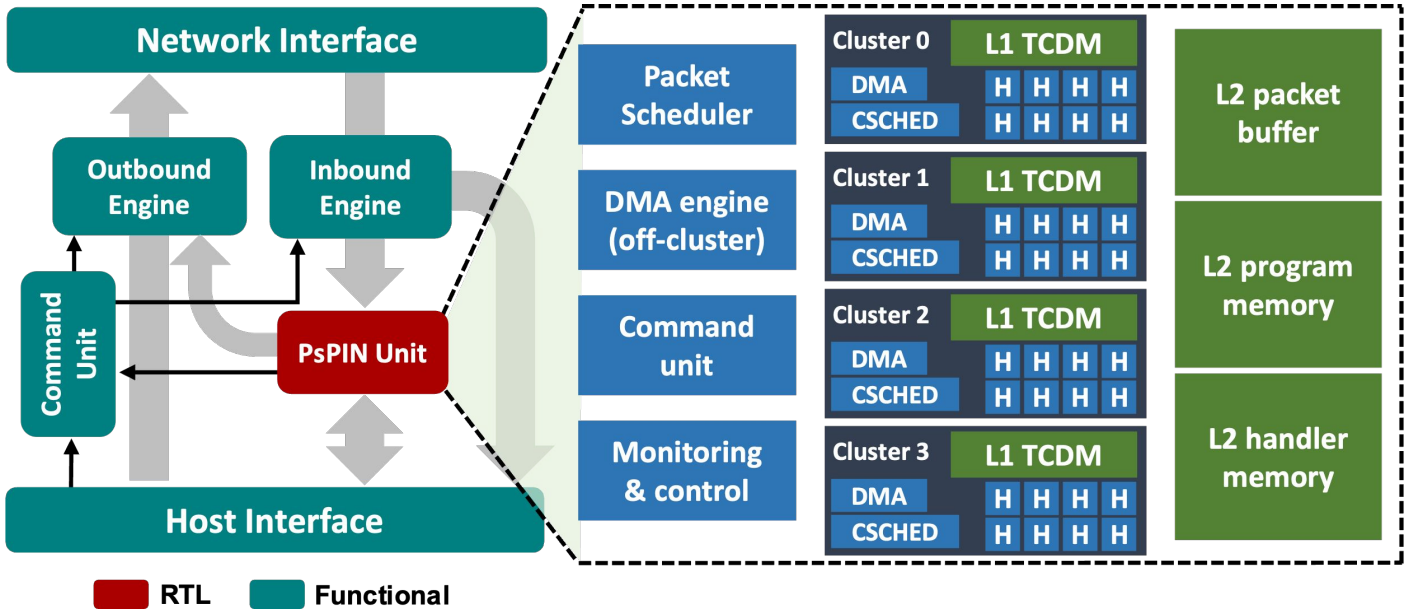




Mellanox BlueField: 16 A72 64bit cores
Estimated area: 51 mm2

Low latency, full throughput	Highly parallel	✓	**32 cores, higher core-count configurations are possible with more clusters**
	Fast scheduling	✓	**Tens of nanoseconds to get handlers started**
	Fast explicit memory access	✓	**Single-cycle L1 memory**
Support for wide range of use cases	Stateful computation support	✓	**Implicit in the sPIN programming model**
	Handlers isolation	✓	**HW-configured (1 cycle) RISC-V PMP**
Easy to integrate	Area and power efficiency	✓	**18.5 mm2, 6.1 W**
	Configurability	✓	**Configurable number of clusters and cores/cluster**

Experimental results

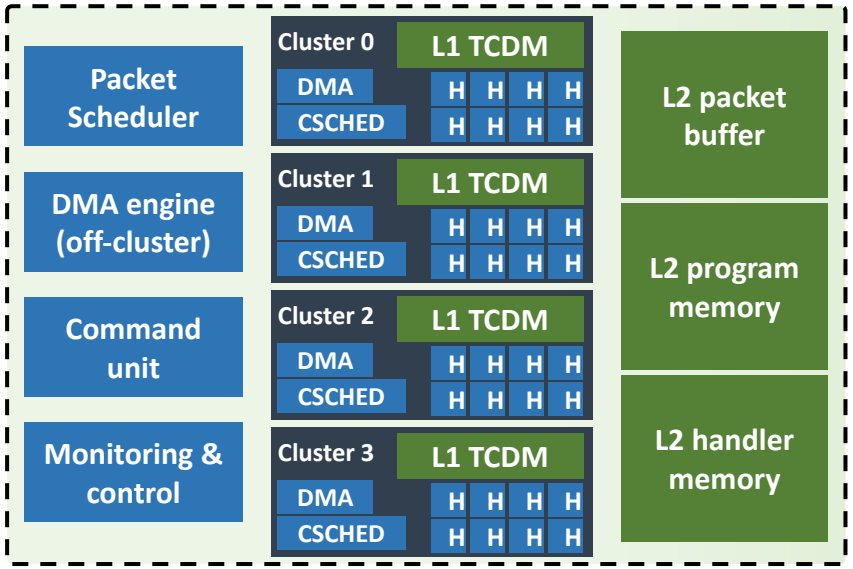


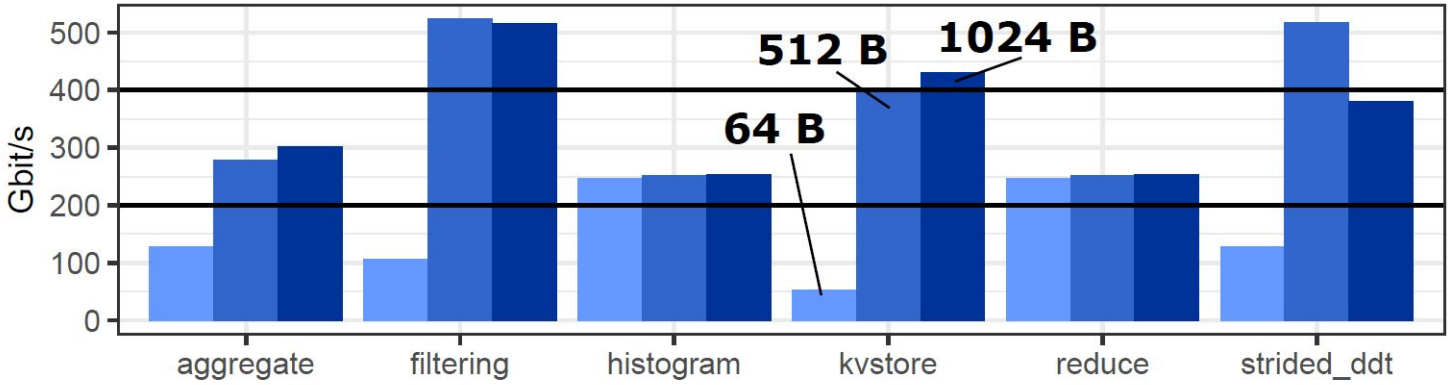
Handlers Characterization

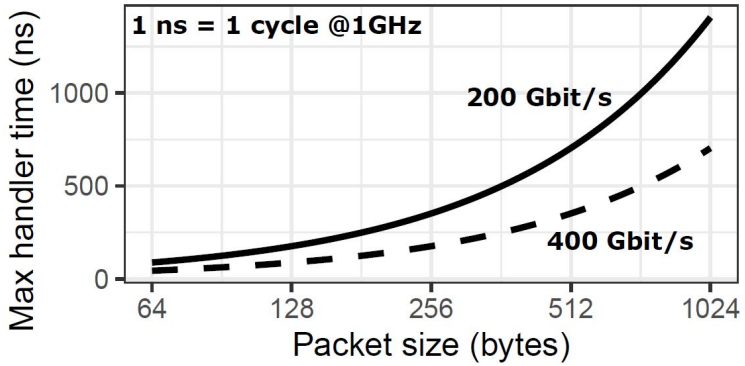
Packet steering
filtering, strided datatypes

Data movement
key-value store

Full packet processing
aggregate, histogram, reduce







How about other architectures?

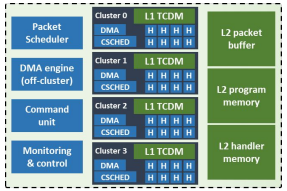
ault @ CSCS

Xeon Gold @ 3 GHz
(18-core, 4-way superscalar, OOO, 64-bit)

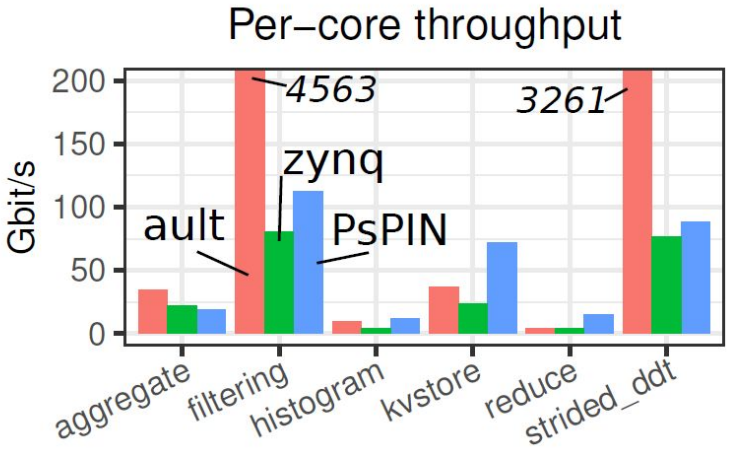
zynq

ARM Cortex-A53 @ 1.2 GHz
(4 cores, 2-way superscalar, 64-bit)

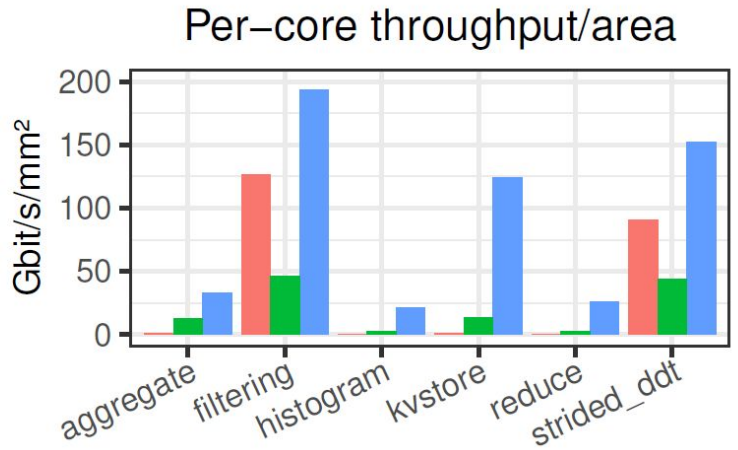
PsPIN



RI5CY (RISC-V) @ 1 GHz
(32 cores, single-issue, in-order, 32-bit)

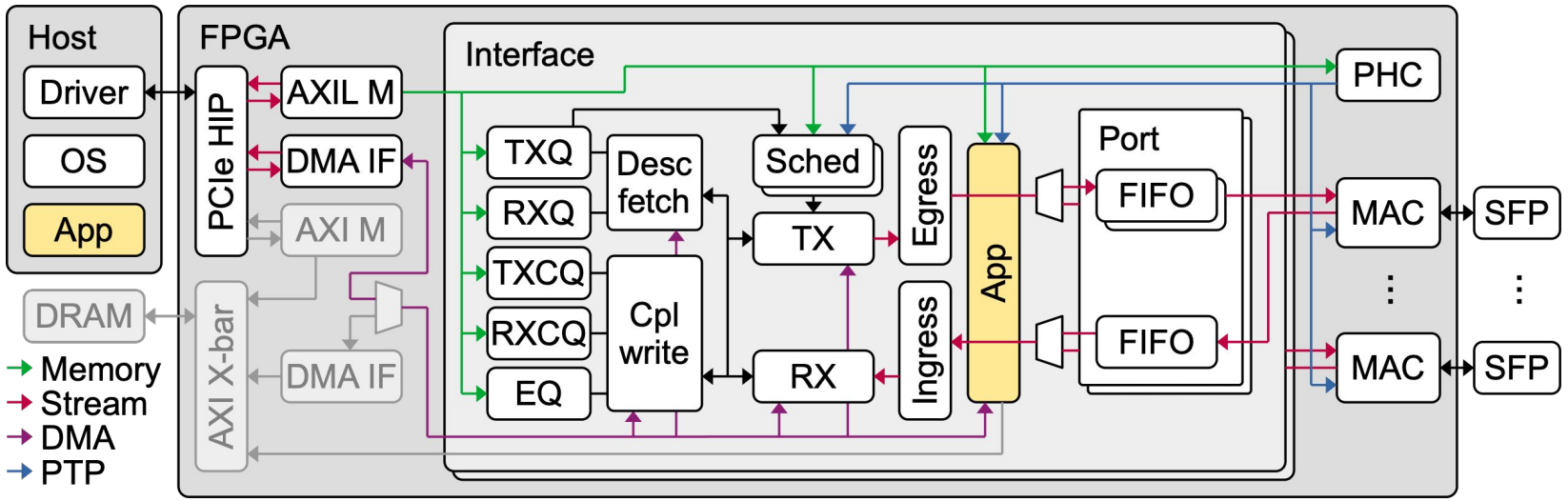


Arch.	Tech.	Die area	PEs	Memory	Area/PE	Area/PE (scaled)
ault	14 nm	485 mm^2 [4]	18	43.3 MiB	17.978 mm^2	35.956 mm^2
zynq	16 nm	3.27 mm^2 [3]	4	1.125 MiB	0.876 mm^2	1.752 mm^2
PsPIN	22 nm	18.5 mm^2	32	12 MiB	0.578 mm^2	0.578 mm^2



FPGA Prototype Implementation

- **Corundum [1]: open-source 100Gbps NIC for FPGAs from UCSD**



UC San Diego

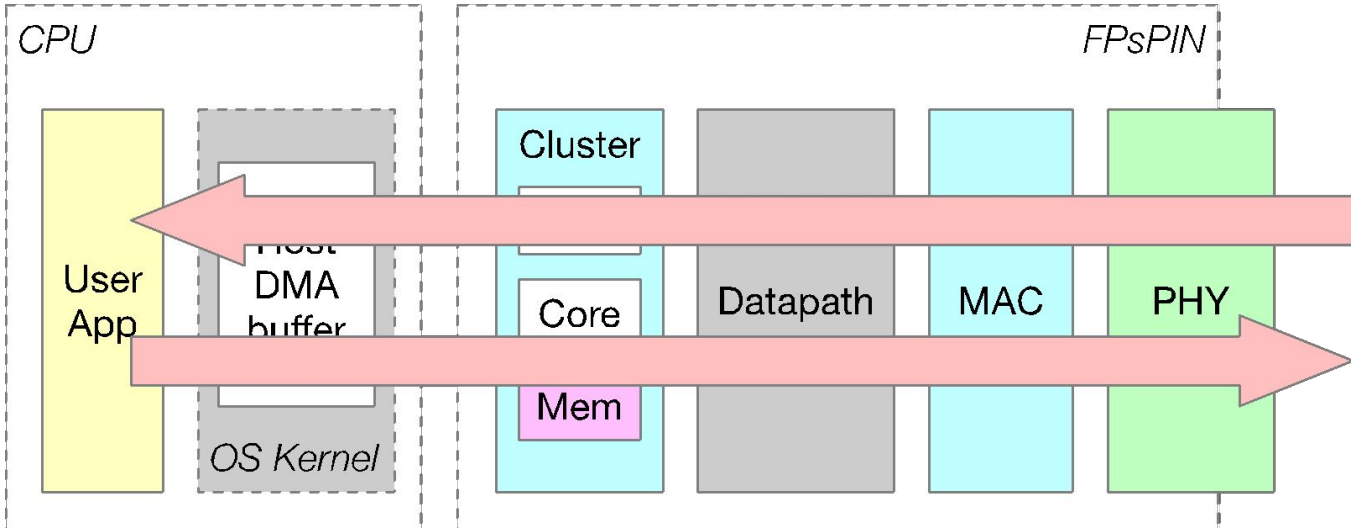
VIVADO™

- **Interface IPs from Corundum & Xilinx Vivado: AXI/AXI Stream infrastructure**
- **Linux kernel module + user-space scripts and library**

[1]: Corundum: An Open-Source 100-Gbps NIC, A. Forencich, A. Snoeren, G. Porter, G. Papen, FCCM'20.

Current progress and plans

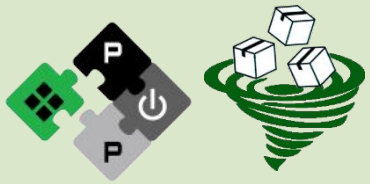
- **Basic implementation finished: the extended ping-pong demo**



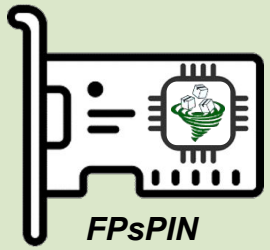
- **Upcoming: real-world complex workload (MPI Datatypes on sPIN [2])**
- **Upcoming: Performance analysis, instrumentation & optimisation**

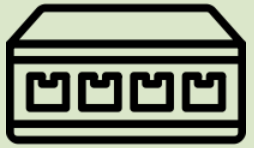
[2]: Accelerating Data Serialization/Deserialization Protocols with In-Network Compute, S. Cao, S. Di Girolamo, T. Hoefler, ExaMPI'22.

A sPINning ecosystem

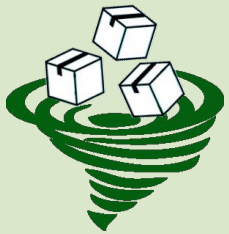


PsPIN (ISCA '21)
Power-efficient sPIN accelerator





Flare (SC '21)
Offloading all-reduce to sPIN switches



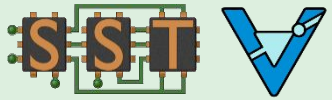
sPIN
(SC '17, best paper fin.)
programming model

architecture

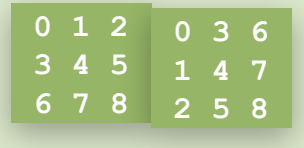
use cases

Simulations

Verilator support (Github)
Running PsPIN in an open-source cycle-accurate simulator



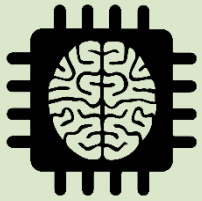
SST support (in progress)
Large scale network simulations mixed with cycle accurate ones



Network-accelerated datatypes (SC '19)

Zoo-sPINNER (MSc)
consensus on sPIN

sPIN-FS (SC '21)



Quantization
Allreduce and other collectives

Erasure coding

Serverless sPIN

Packet classification and pattern matching