Computer Sciences
SCHOOL OF COMPUTER, DATA & INFORMATION SCIENCES
UNIVERSITY OF WISCONSIN–MADISON

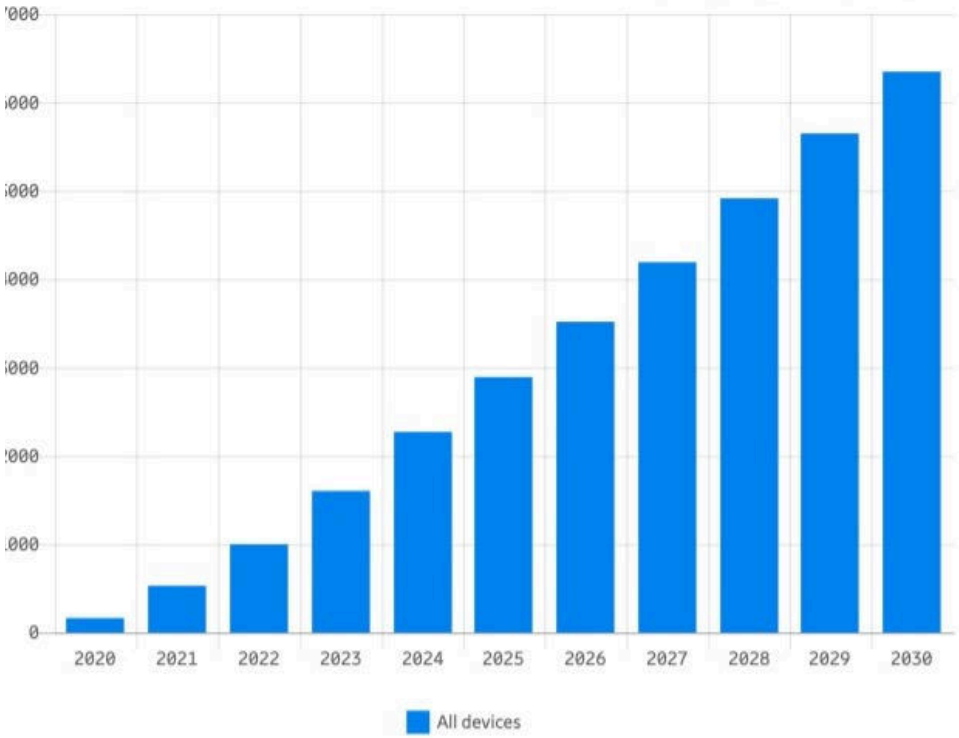
Building Massive MIMO Baseband Processing on a Single-Node Supercomputer

Xincheng Xie Wentao Hou Zerui Guo Ming Liu

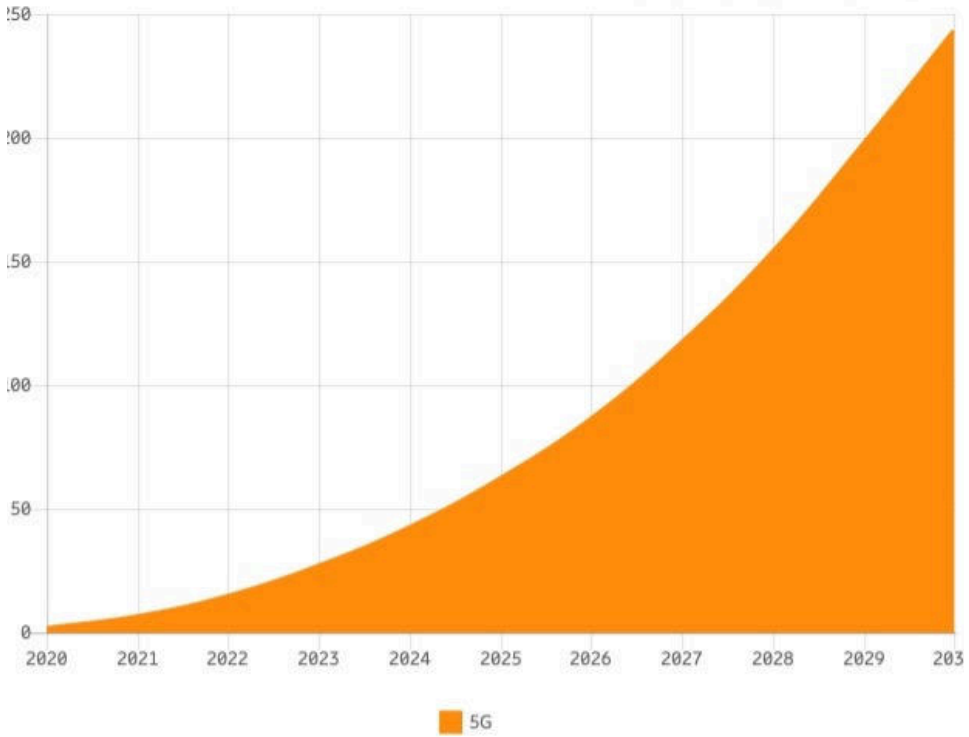
University of Wisconsin-Madison

Rising mobile traffic

Mobile subscriptions



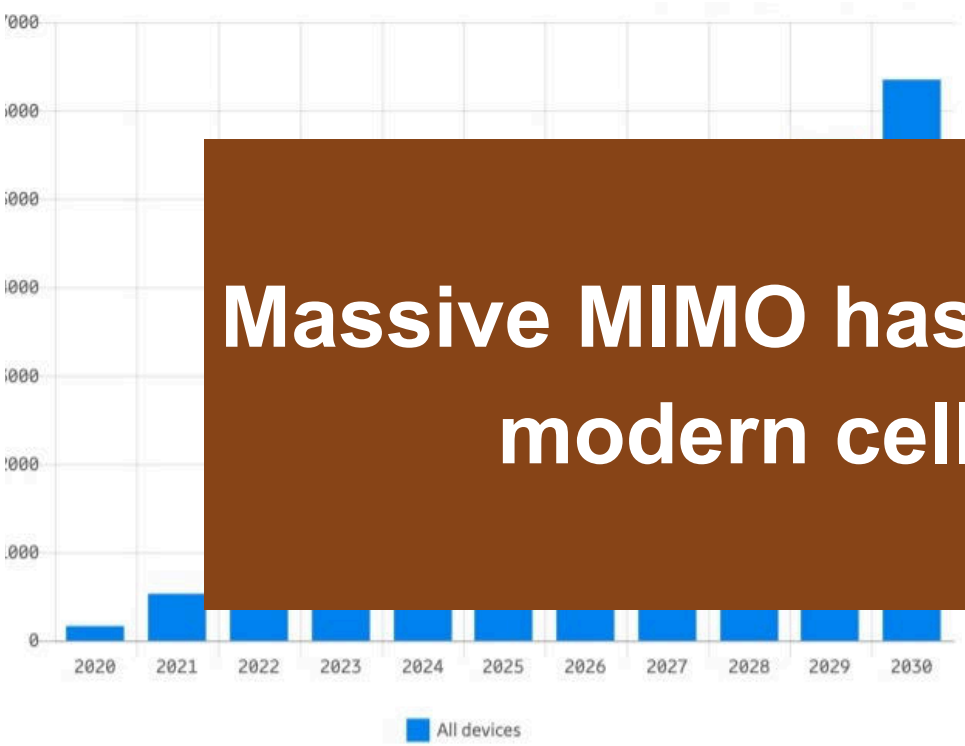
Mobile data traffic



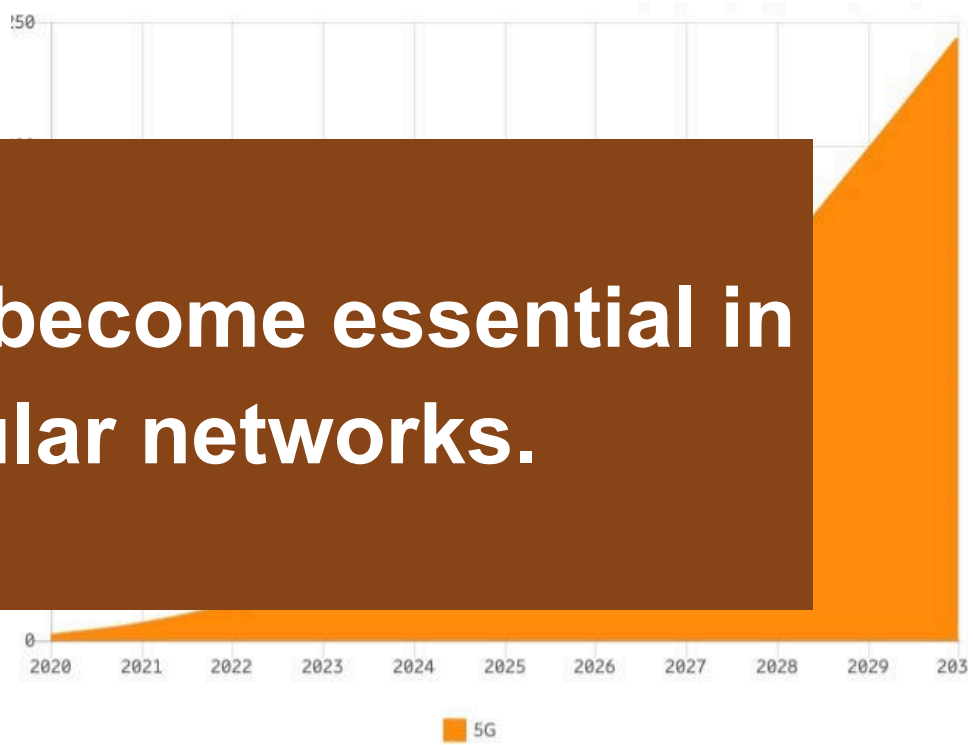
Ericsson, 2024 Mobile Traffic Forecast

Rising mobile traffic

Mobile subscriptions



Mobile data traffic

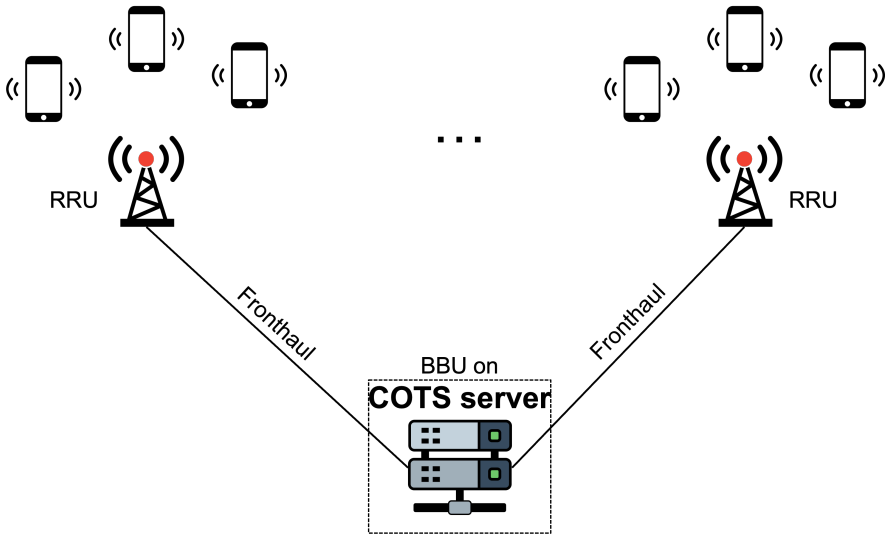


Massive MIMO has become essential in modern cellular networks.

Ericsson, 2024 Mobile Traffic Forecast

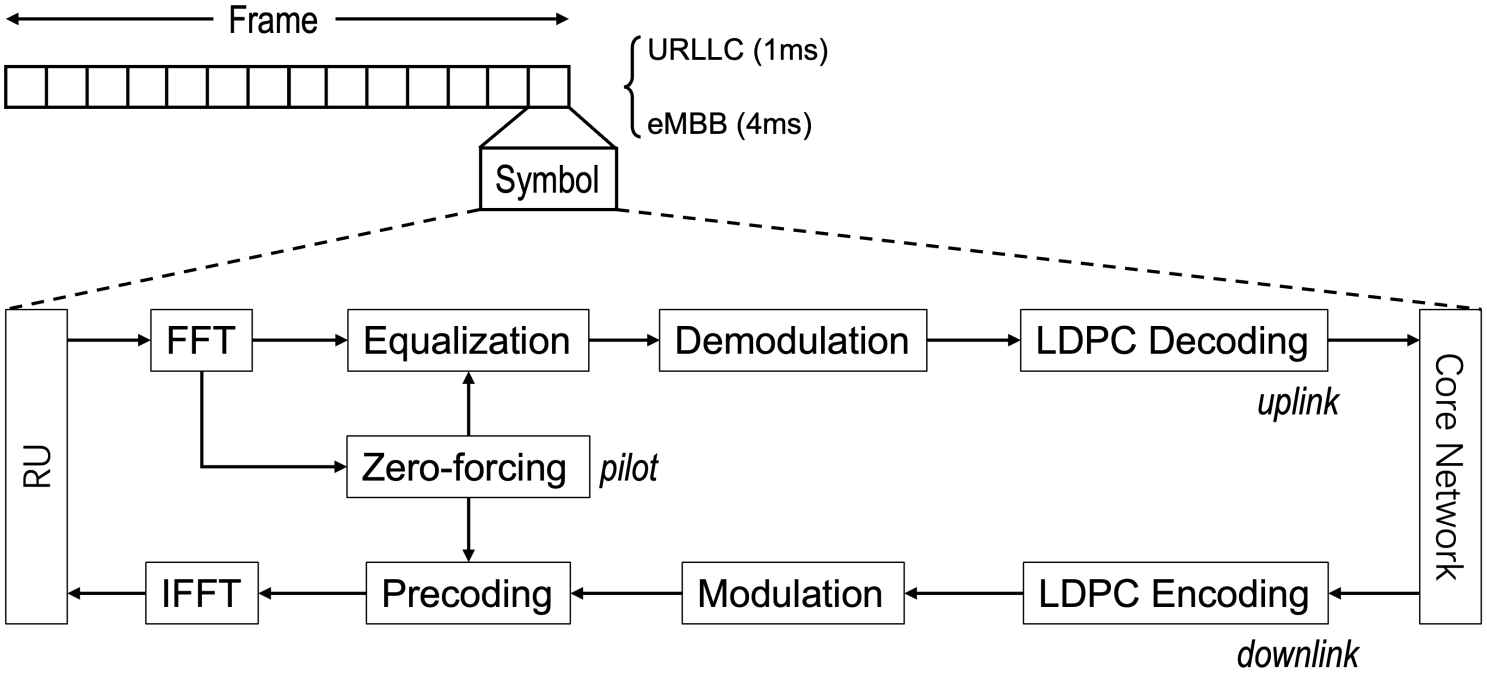
vRAN is becoming widely adopted

- Use commodity hardware for baseband processing
  - Mitigate vendor lock-in
  - Enable flexible deployment
  - Offer better operational visibility
  - Reduce the TCO



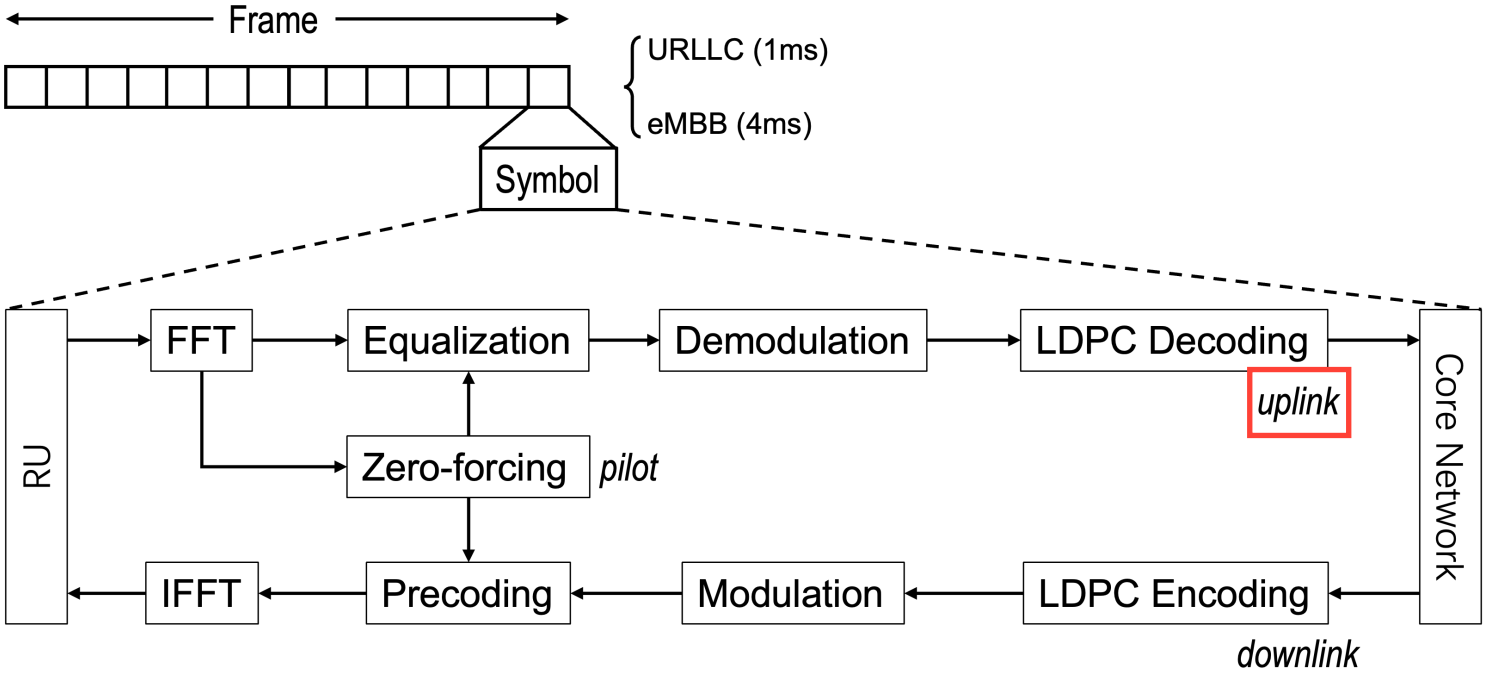
MIMO baseband processing

- Require high computing power
- Has stringent latency requirements



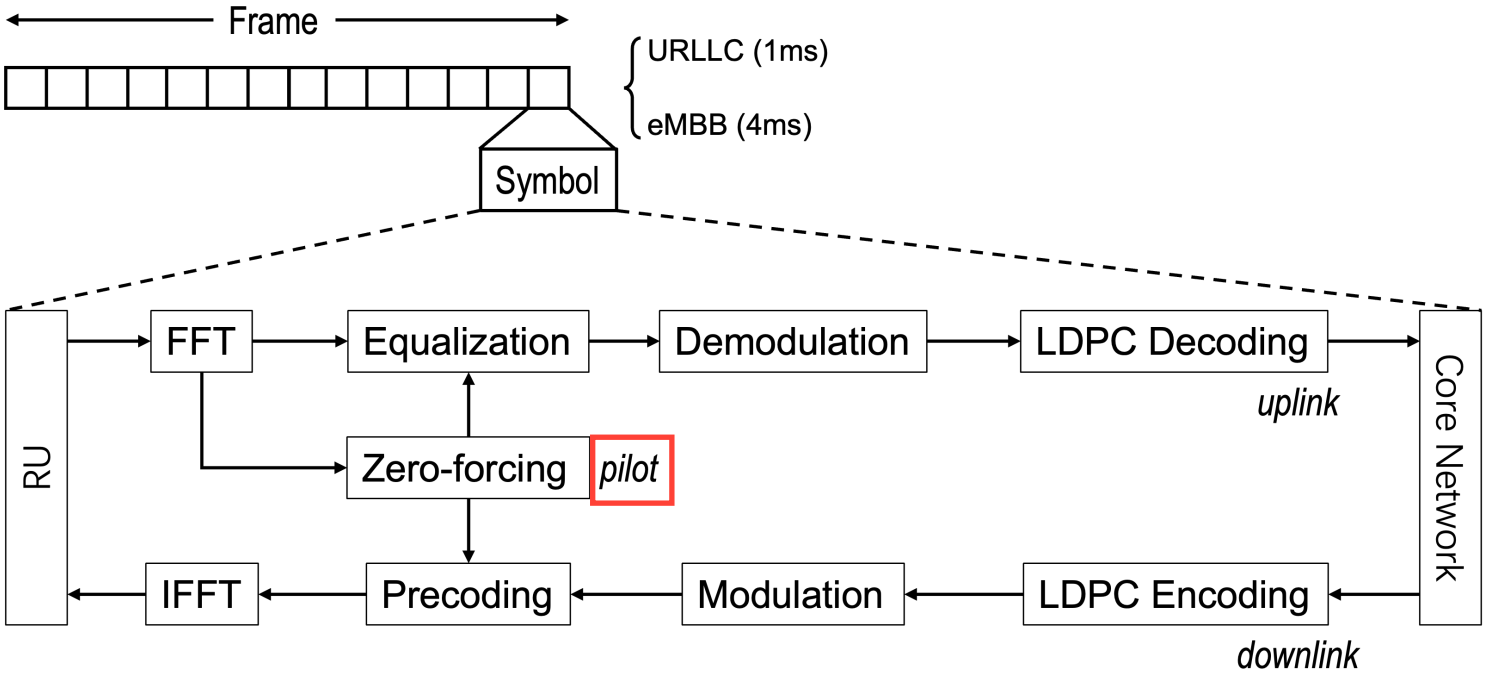
MIMO baseband processing

- Require high computing power
- Has stringent latency requirements



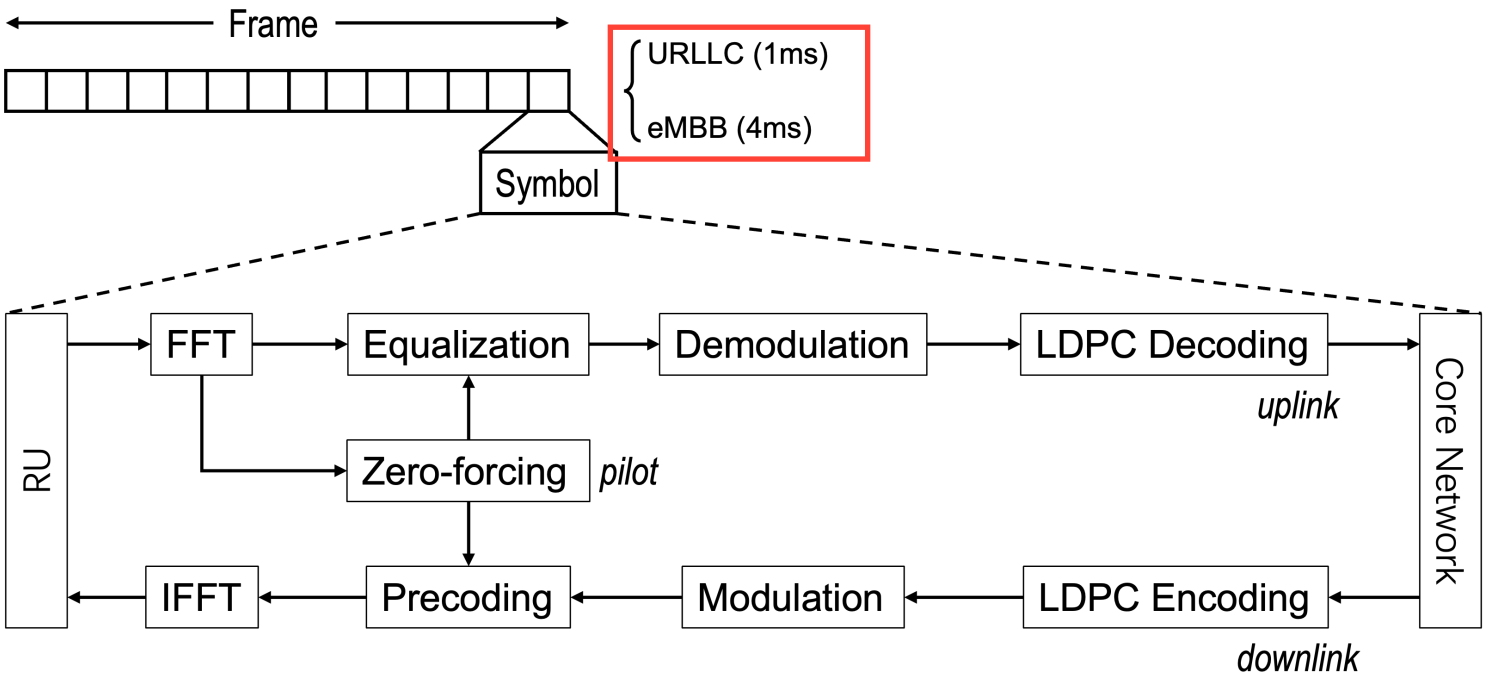
MIMO baseband processing

- Require high computing power
- Has stringent latency requirements



MIMO baseband processing

- Require high computing power
- Has stringent latency requirements



Existing vRAN hardware substrates

- **General-purpose x86 servers**
  - BigStation [SIGCOMM'10], Hydra [NSDI'23]
  - Cons: high cost and larger physical space under scale
- **Domain-specific accelerators**
  - LuMaMi: a bespoke testbed using 50 Xilinx FPGAs
  - Cons: management inflexibility and limited reconfigurability

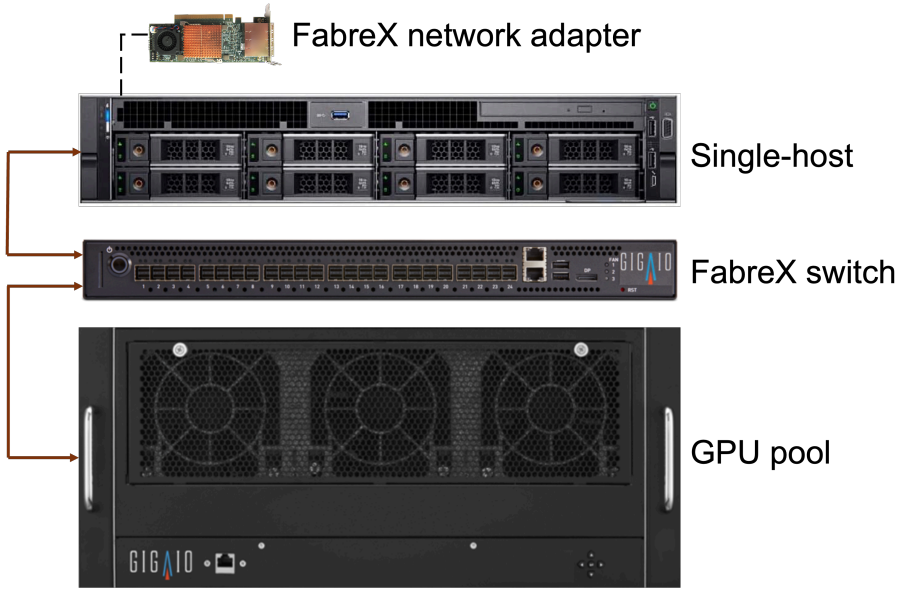
Existing vRAN hardware substrates

Requirements:

- **Scalable computational power**
- **High programmability**
- **Flexible deployment**

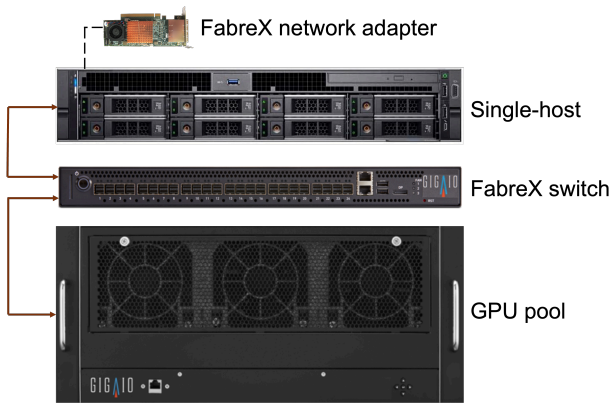
Single-Node Supercomputer (SNC)

- A single server with a pool of GPUs via load-store interconnects
  - Host adapter: provide pool access connectivity
  - External switch: enable fanout scalability
  - GPU box: consolidate bunch of GPUs in one or several standalone chassis



SNC benefits

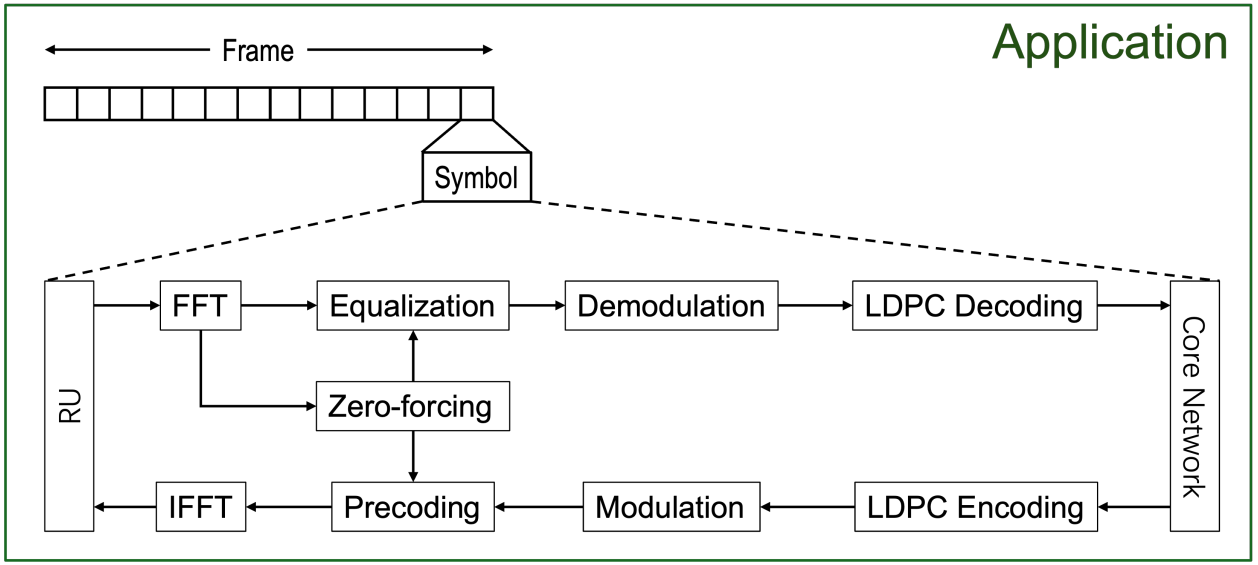
- Host-native system stack
  - Apply CUDA programming model and use existing SDKs
- On-demand GPU scaling
  - Add/remove GPUs without host interrpution
- Consistent inter-GPU communication bandwidth
  - Cross-GPU data movements traverse external fabric



How can we use the SNC to efficiently run massive MIMO baseband processing?

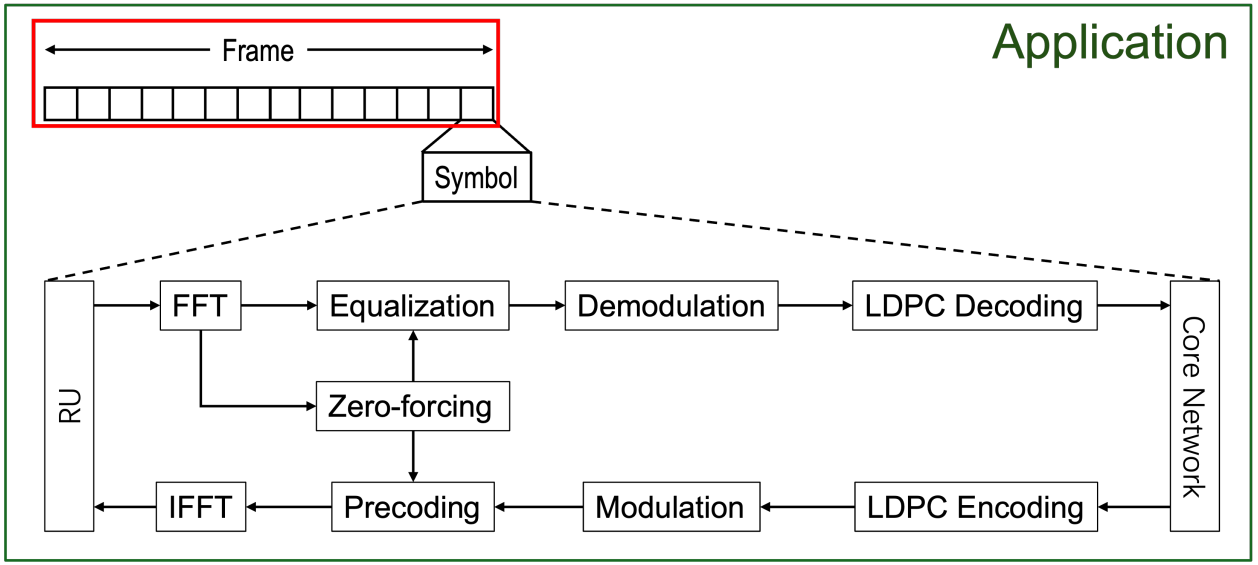
Challenges: diverse SW parallelism

- Software parallelism



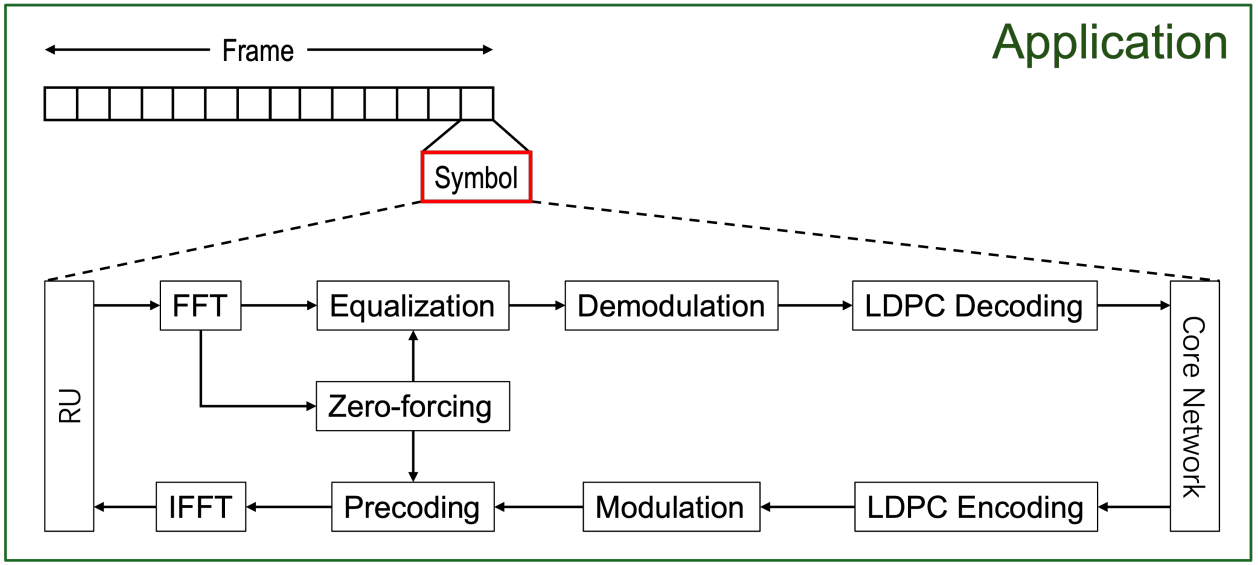
Challenges: diverse SW parallelism

- Software parallelism
  - Frame level (*LuMaMi*)



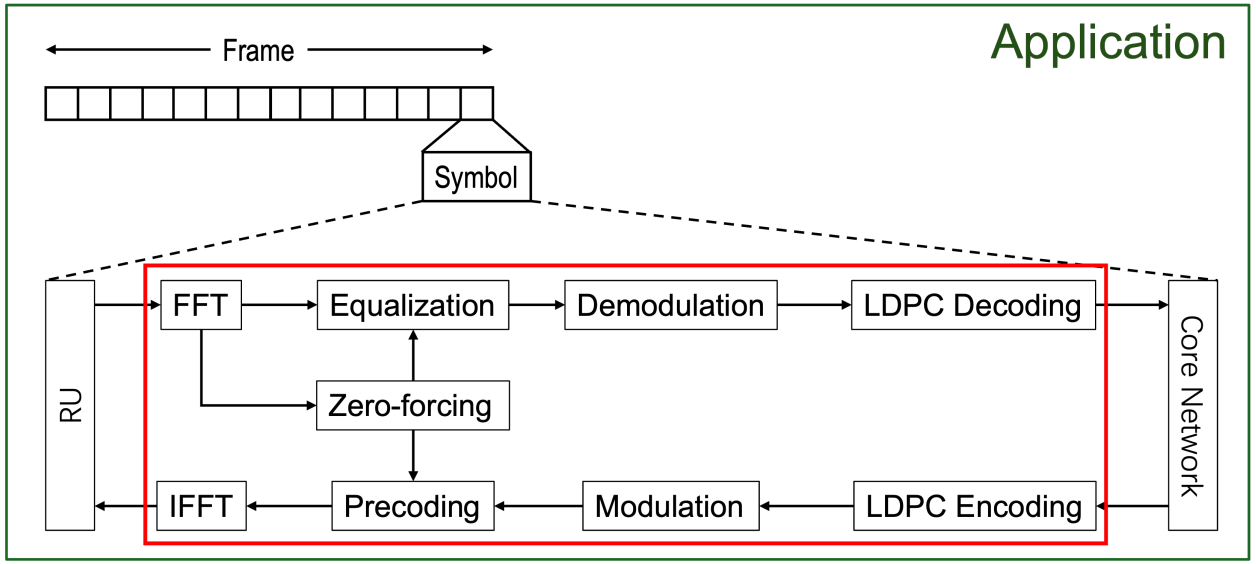
Challenges: diverse SW parallelism

- Software parallelism
  - Frame level (*LuMaMi*)
  - Symbol level (*Hydra*)



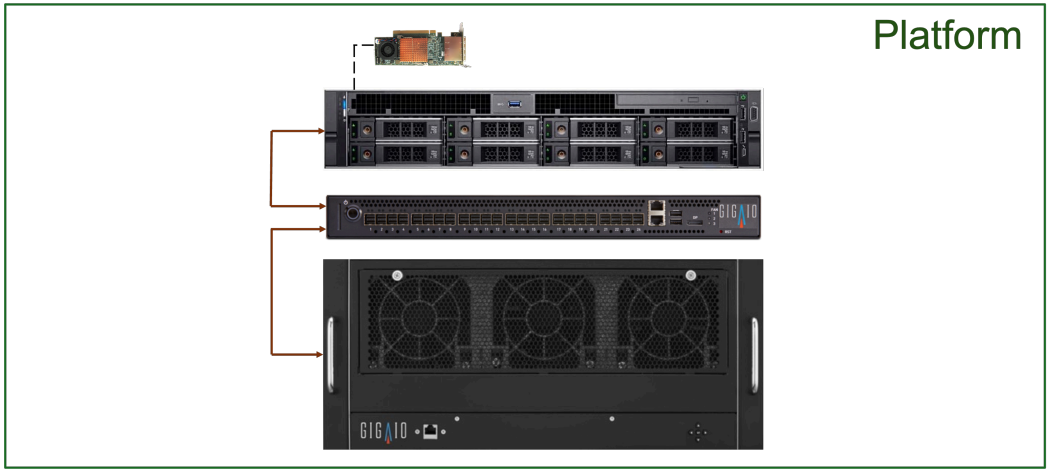
Challenges: diverse SW parallelism

- Software parallelism
  - Frame level (*LuMaMi*)
  - Symbol level (*Hydra*)
  - Task level (*BigStation*)



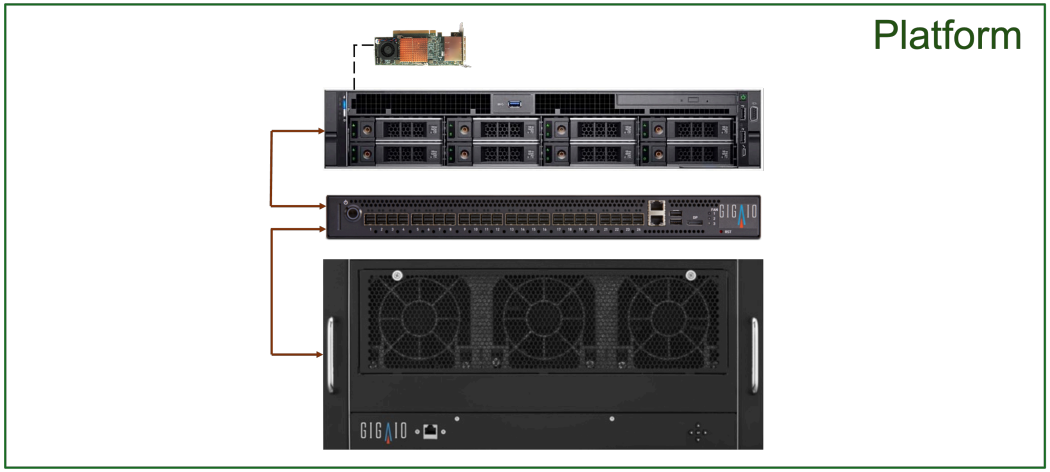
Challenges: irregular HW parallelism

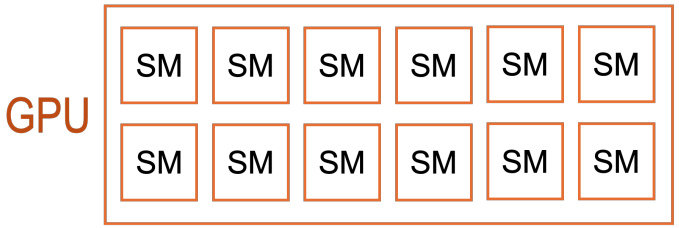
- Hardware parallel degree
  - P: required parallelism (SM #)
  - Q: available parallelism (SM #)



Challenges: irregular HW parallelism

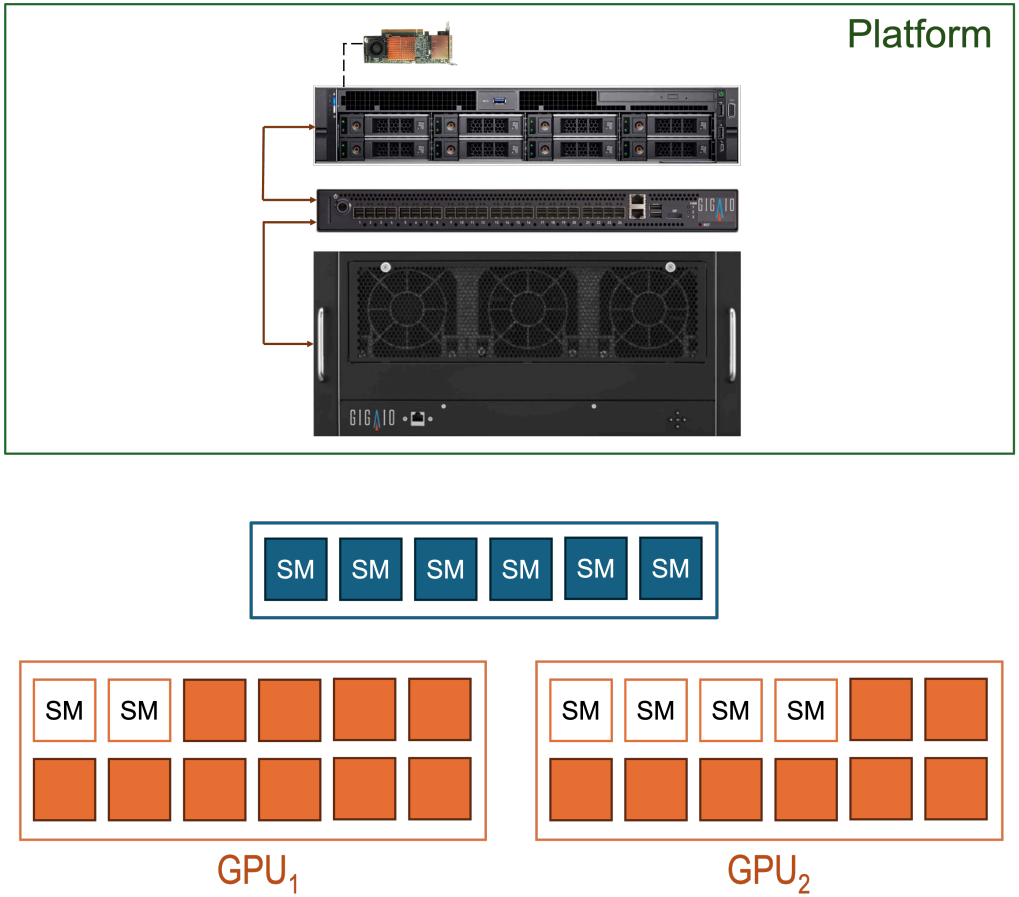
- Hardware parallel degree
  - P: required parallelism (SM #)
  - Q: available parallelism (SM #)
- Full parallelism: $P \leq Q$





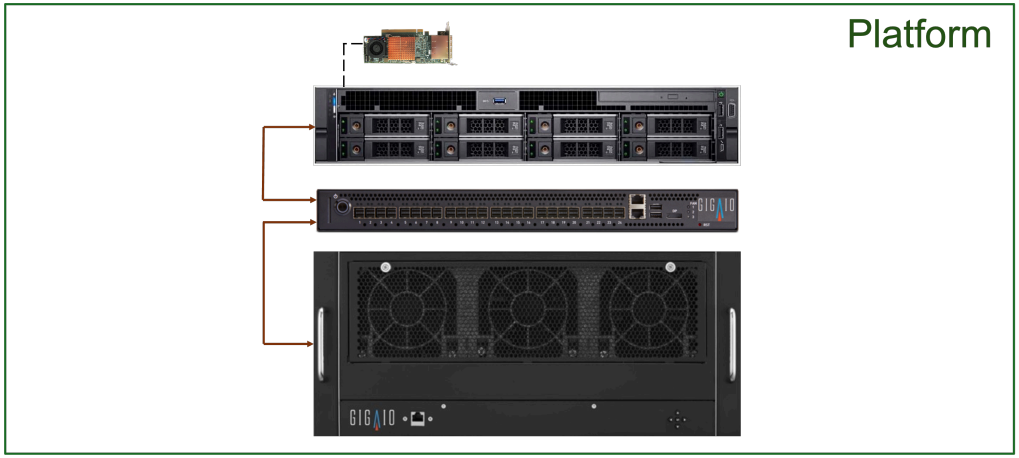
Challenges

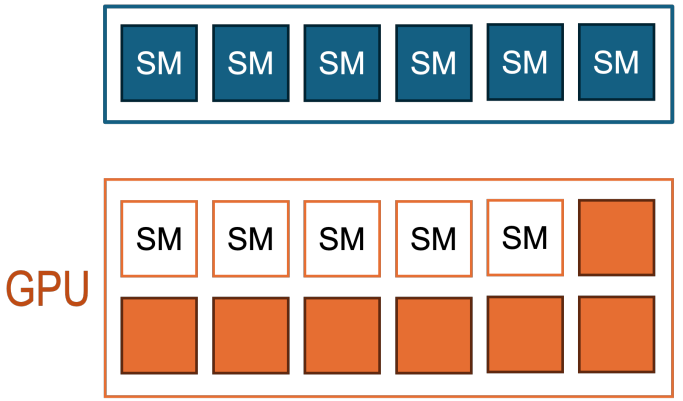
- Hardware parallel degree
  - Assume P required, Q available
  - Full parallelism:
    $P \leq Q$
  - Fragmented parallelism:
    $P \leq Q$ but Q distributed



Challenges

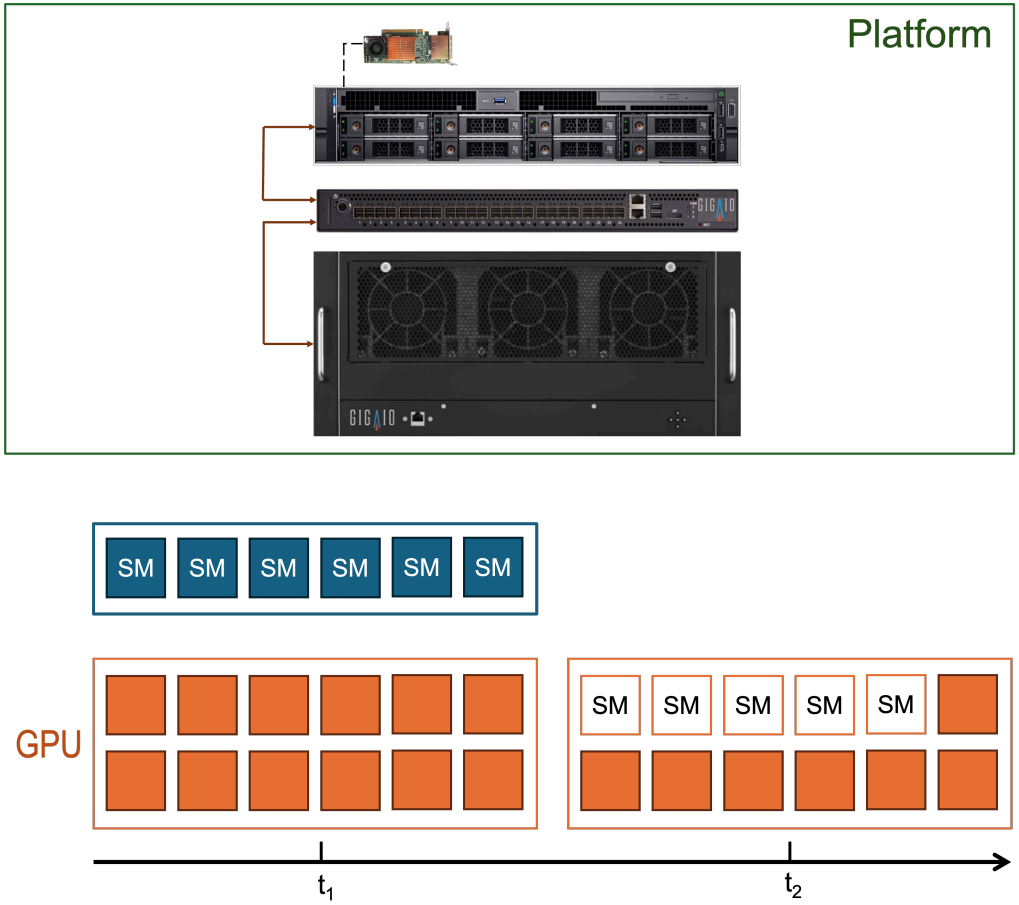
- Hardware parallel degree
  - Assume P required, Q available
  - Full parallelism:
    $P \leq Q$
  - Fragmented parallelism:
    $P \leq Q$ but Q distributed
  - Partial parallelism:
    $P > Q > 0$





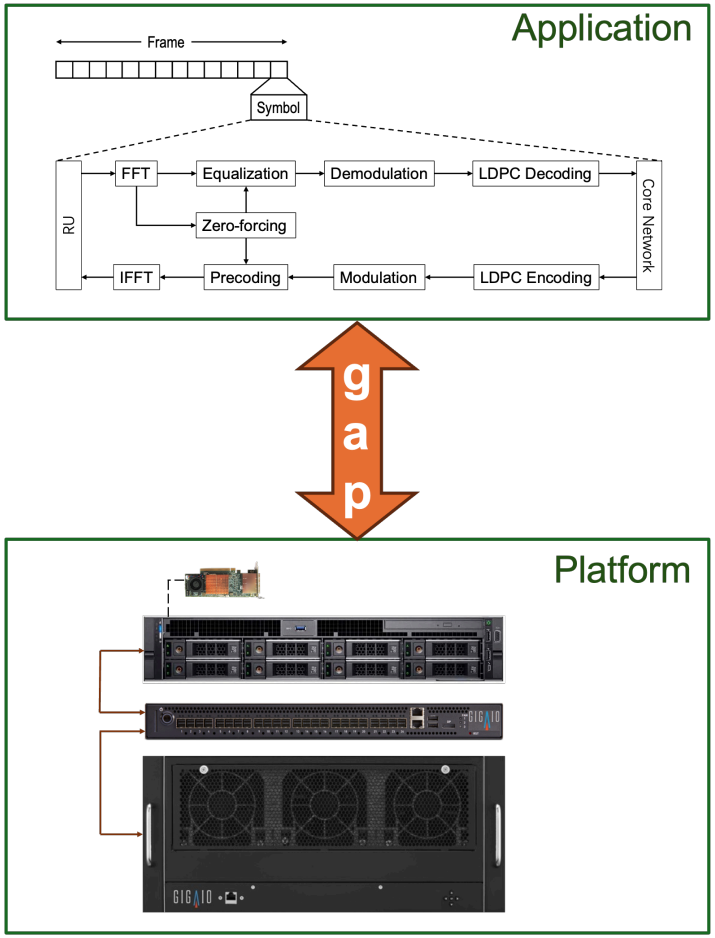
Challenges

- Hardware parallel degree
  - Assume P required, Q available
  - Full parallelism:
    $P \leq Q$
  - Fragmented parallelism:
    $P \leq Q$ but Q distributed
  - Partial parallelism:
    $P > Q > 0$
  - Delayed parallelism:
    $Q = 0$



Challenges

Software-parallelism
- Frame-level
- Symbol-level
- Task-level



Hardware-parallelism
- Full parallelism
- Fragmented parallelism
- Partial parallelism
- Delay parallelism

Challenges

How to effectively minimize the gap between software and hardware parallelism?

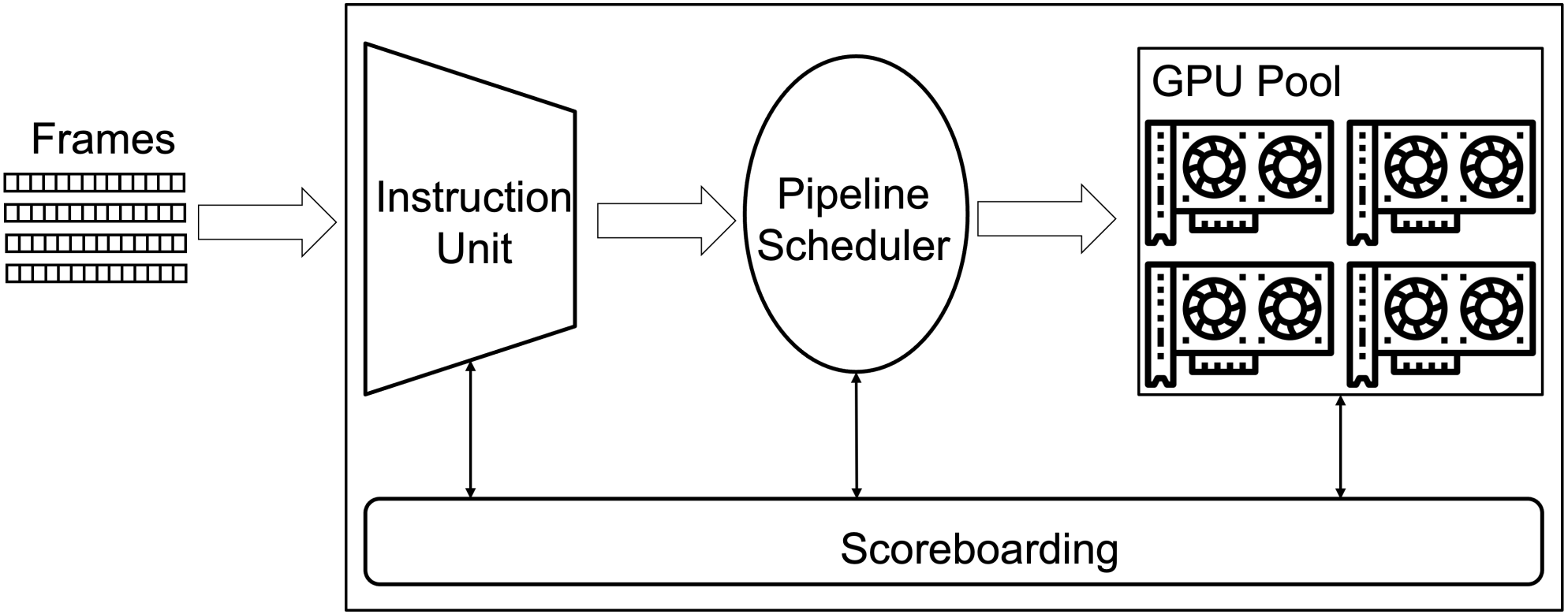
Key idea: dynamically adjust software granularity based on the hardware parallelism availability

MegaStation overview

- Inspired by dynamic instruction scheduling[1]
- Model SNC as a tightly coupled microprocessor
- System components
  - Instruction unit: parse and analyze instructions
  - Scoreboarding: track GPU status and instruction execution
  - Pipeline scheduler: schedule instructions to GPUs

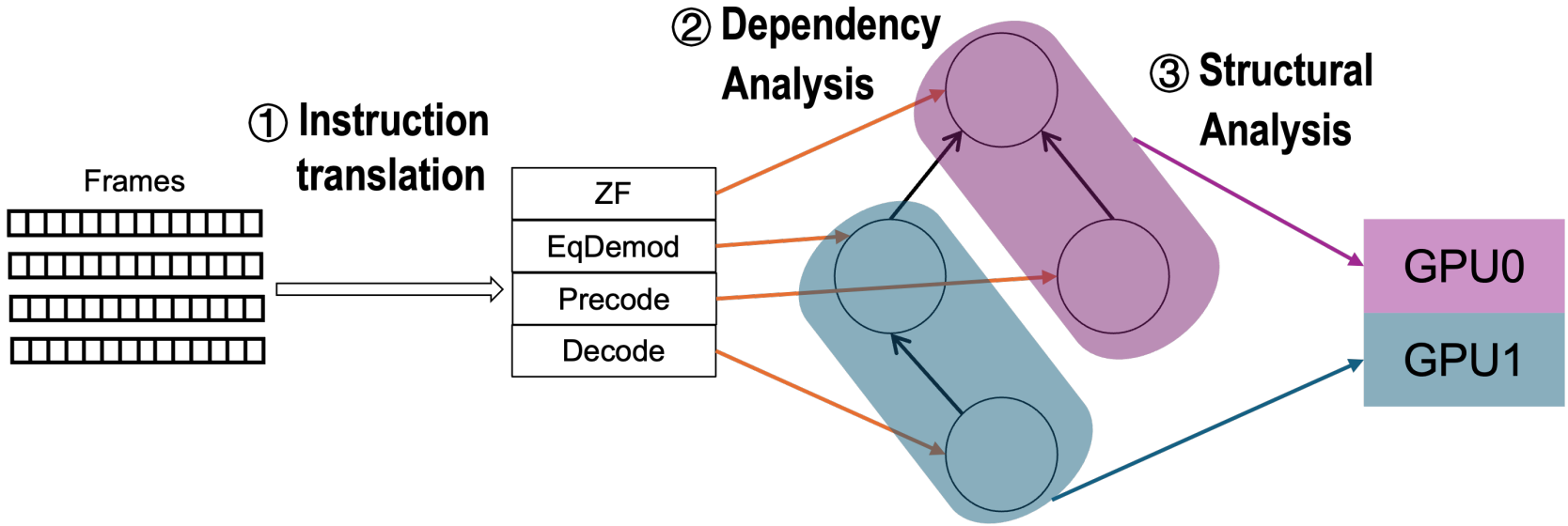
[1] R. M. Tomasulo, “An Efficient Algorithm for Exploiting Multiple Arithmetic Units,” in IBM Journal of Research and Development, vol. 11, no. 1, pp. 25-33, Jan. 1967.

MegaStation overview



MegaStation: instruction unit

- Parse frames to 11 types of task instructions
- Analyze instruction dependency during frame processing
- Group instructions based on locality

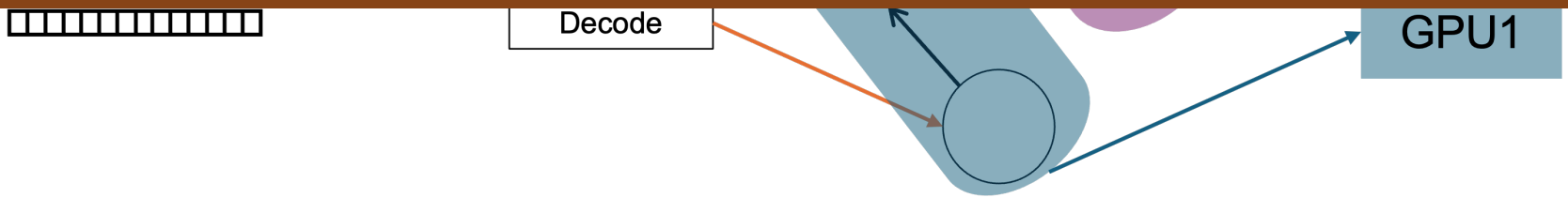


MegaStation: instruction unit

- Parse frames to 11 types of task instructions
- Analyze instruction dependency during frame processing

Structural analysis is based on the characteristics experiment.

(More details in the paper)



MegaStation: scoreboarding

- Track instruction execution status and GPU availability condition

Instruction Status Table

inst index	required resources	execution time
inst j	SM# j, SMem j, …	time j
inst j+1	SM# j+1, SMem j+1, …	time j+1
inst j+2	SM# j+2 , SMem j+2, …	time j+2

GPU Status Table

GPU index	planned instructions			completion flag		
GPU k	inst 1	…	inst n_k	flag 1	…	flag n_k
GPU k+1	inst 1	…	inst n_{k+1}	flag 1	…	flag n_{k+1}
GPU k+2	inst 1	…	inst n_{k+2}	flag 1	…	flag n_{k+2}

MegaStation: pipeline scheduler

- Apply least slack time scheduling displine
- Optimization
  - Reordering: to fill idle SMs
  - Coalescing: to maximize locality
  - Over-commitment: to avoid scheduling delay

Algorithm 2 LROC Algorithm.

```
1:  procedure LROC_SCHEDULER
2:      min_inst = instQ.min();
3:      while min_inst.TS_slack − TS_cur < T_thr do                ▷ LSTF
4:          SUBMIT(min_inst, preced_stream(min_inst))
5:          min_inst = instQ.next_min()
6:      for inst ∈ instQ do                                        ▷ Reorder
7:          if GPU.avail_SM ≤ 0 then break
8:          if pred_insts.finish & inst.SM ≤ GPU.avail_SM then
9:              SUBMIT(inst, pred_insts.stream)
10:     for inst ∈ instQ do                                        ▷ Over-commit
11:         if inst.label = delayed then
12:             SUBMIT(inst, low_prior_stream)
13:             break
14:     for inst ∈ instQ do                                        ▷ Coalesce
15:         if pred_insts.running & inst.SM ≤ pred_insts.SM then
16:             SUBMIT(inst, pred_insts.stream)
```

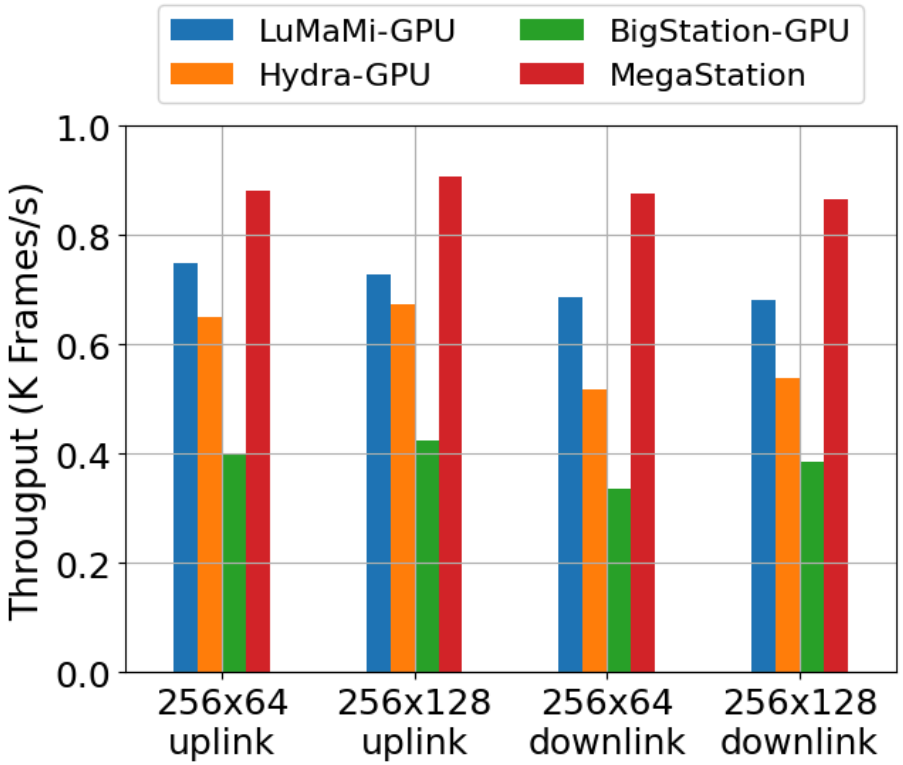
Evaluation

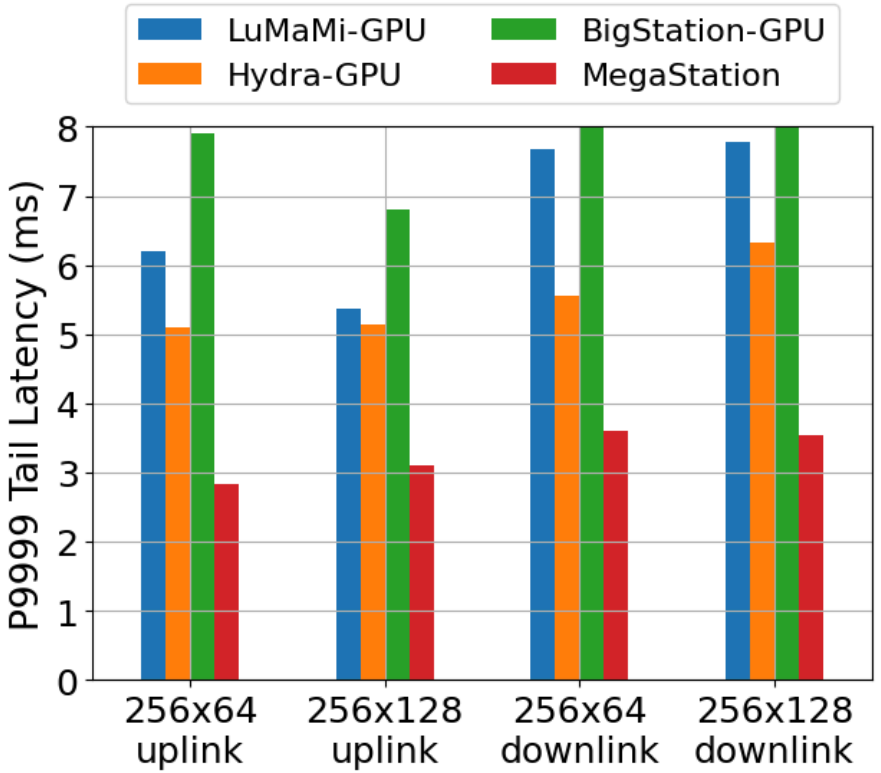
- Hadware setup
  - NVIDIA A100/V100 GPUs
  - Intel Xeon Gold 6248 processors
  - GigaIO FabreX platform

- Comparison baselines
  - LuMaMi-GPU: frame-level parallelism
  - Hydra-GPU: symbol-level parallelism
  - BigStation-GPU: task-level parallelsim

Evaluation: end-to-end performance

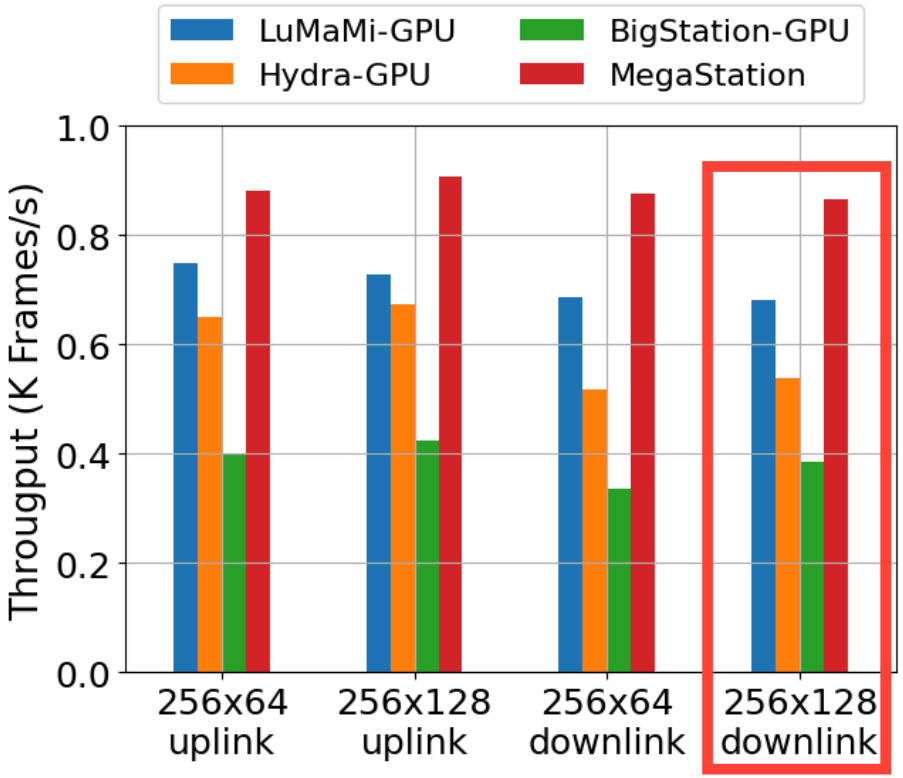
- MegaStation acheives up to $3\times$ throughput improvement without violating the latency requirement

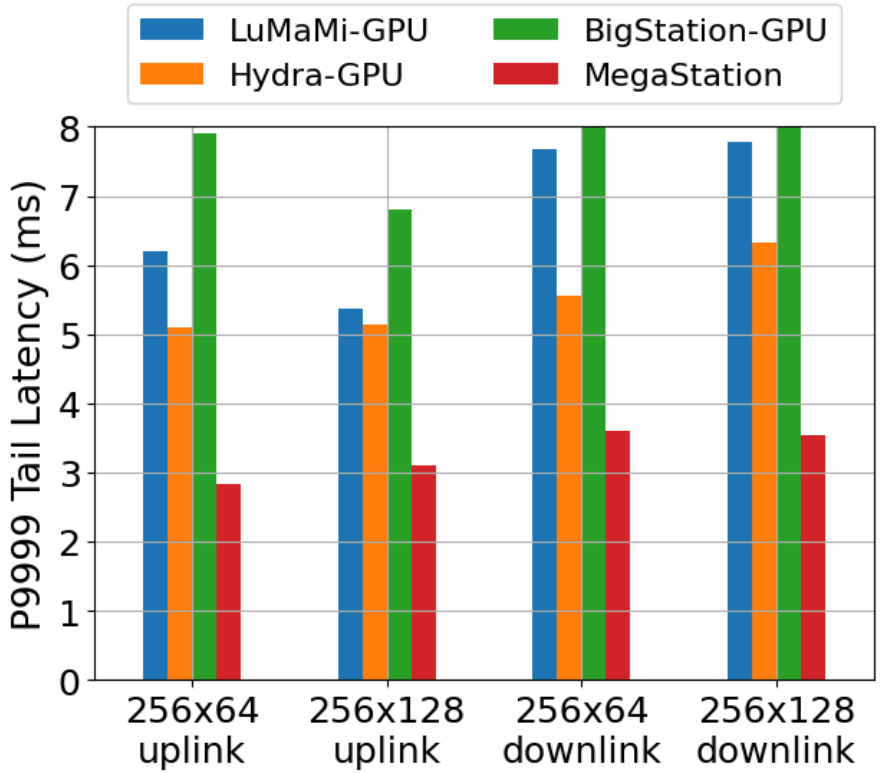




Evaluation: end-to-end performance

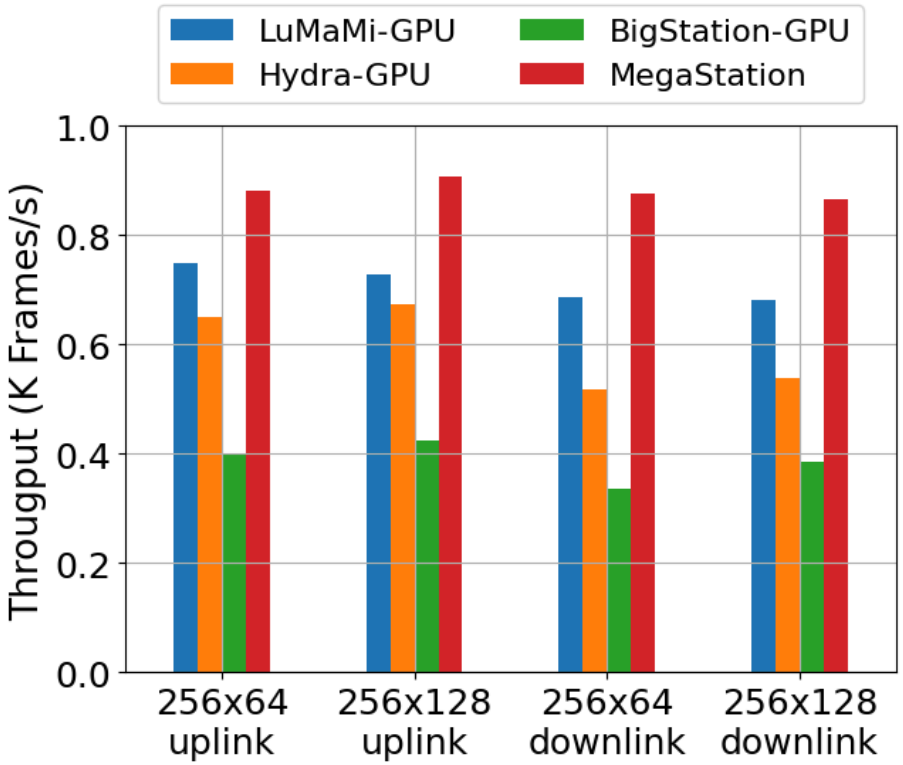
- MegaStation acheives up to $3\times$ throughput improvement without violating the latency requirement

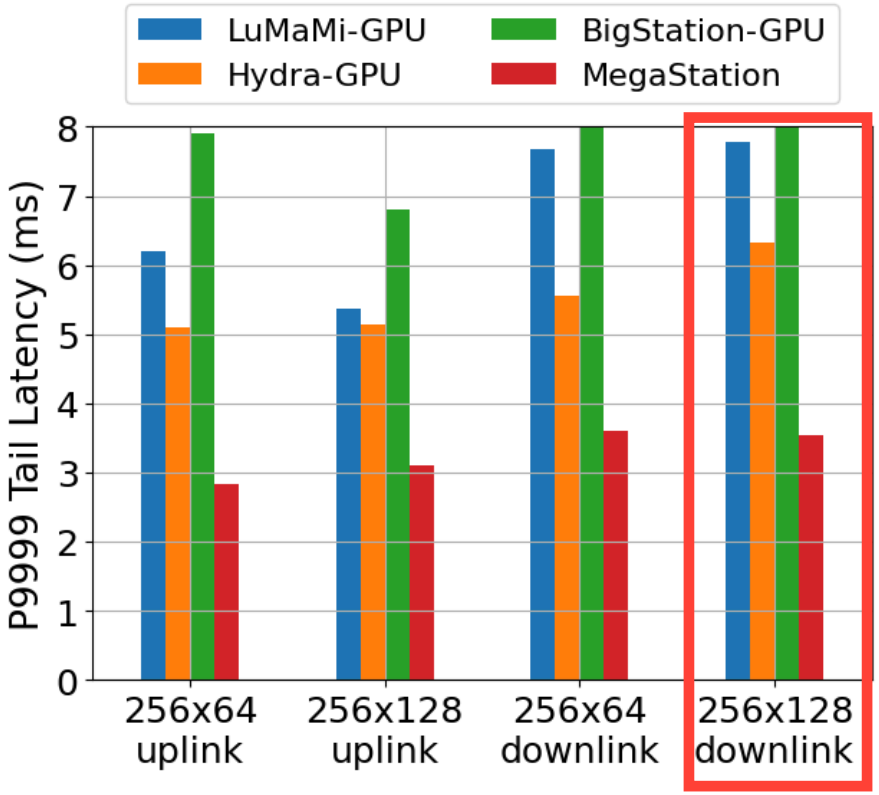




Evaluation: end-to-end performance

- MegaStation acheives up to $3\times$ throughput improvement without violating the latency requirement





Other results

- RU/GPU scalability
- Heterogeneous GPUs
- MegaStation Drill-down
- …

Summary

- An emerging computing platform: Single-Node Supercomputer
  - Scalable, programmable, and physically compact
- Problem: the dilemma between SW and HW parallelism
- Key idea: adjust SW granularity based on the HW parallelism
- MegaStation: model a SNC as a tightly coupled microprocessor
  - Instruction unit, scoreboarding, pipeline scheduler
- Evaluation results
  - 66.2% lower tail latency and $3\times$ higher throughput
- Repo: https://github.com/netlab-wisconsin/MegaStation

Thanks!