Hybrid Skiplist: Combining the Best of Near-Data-Processing and Lock-Free Algorithms

Jiwon Choe* — Brown University, jiwon_choe@brown.edu
Tali Moreshet — Boston University, talim@bu.edu
Maurice Herlihy — Brown University, mph@cs.brown.edu
R. Iris Bahar — Brown University, iris_bahar@brown.edu

BROWN

BOSTON UNIVERSITY

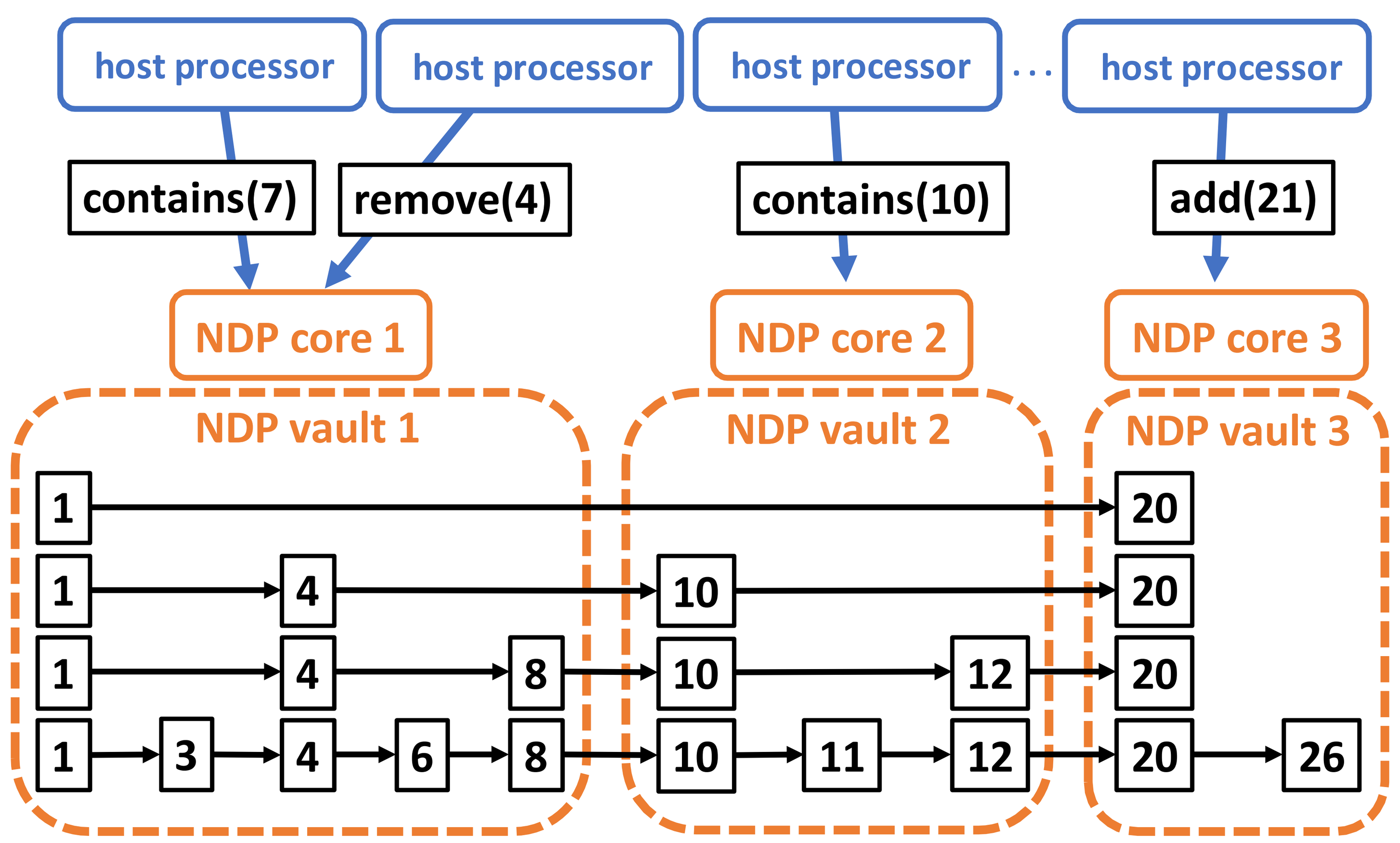
Skiplist

- **pointer-chasing data structure** that probabilistically achieves $\log_2 N$ access time
- better for concurrency than binary search trees
- used for key-value stores, DB indexes
- pointer-chasing incurs **irregular memory access**
  – expected to benefit from near-data-processing (NDP)

NDP-based Skiplist

Liu *et al.* SPAA '17, Choe *et al.* SPAA '19

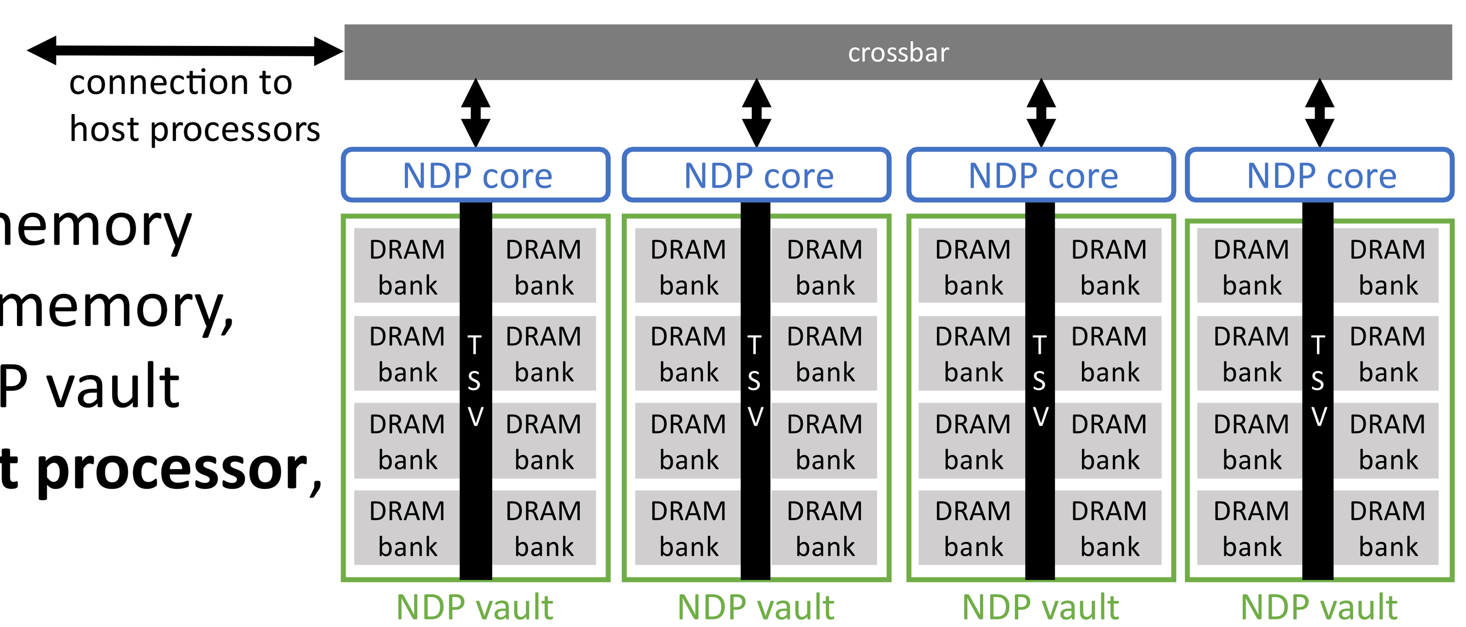
- **preserves concurrency** by:
  - **partitioning** skiplist across NDP vaults
  - **flat-combining** concurrent ops at NDP cores



- **little performance gains compared to state-of-the-art host-based lock-free skiplist:**
  - failed to consider cache-friendly access pattern of skiplist traversal – **higher-level nodes** are accessed repeatedly, **likely to remain in cache**

Near-Data-Processing (NDP)

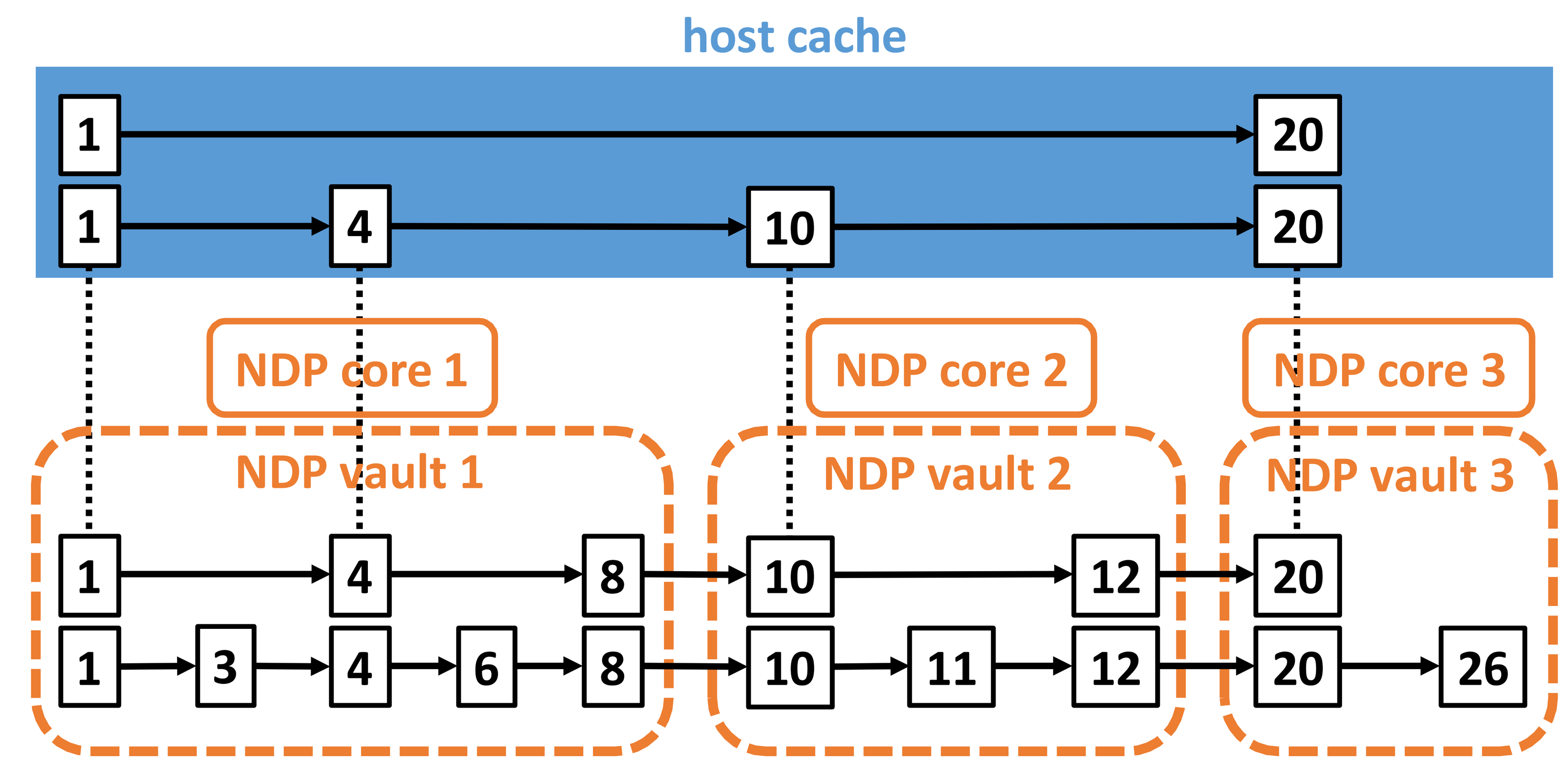
- based on 3D die-stacked memory
- **NDP vaults:** vertically divided sections of memory
- **NDP cores:** simple processors placed near memory, has exclusive access to data in coupled NDP vault
- **faster access to memory compared to host processor**, DRAM-inherent delays remain



Hybrid Skiplist

Combines lock-free and NDP-based implementations to account for data access patterns and underlying architecture.

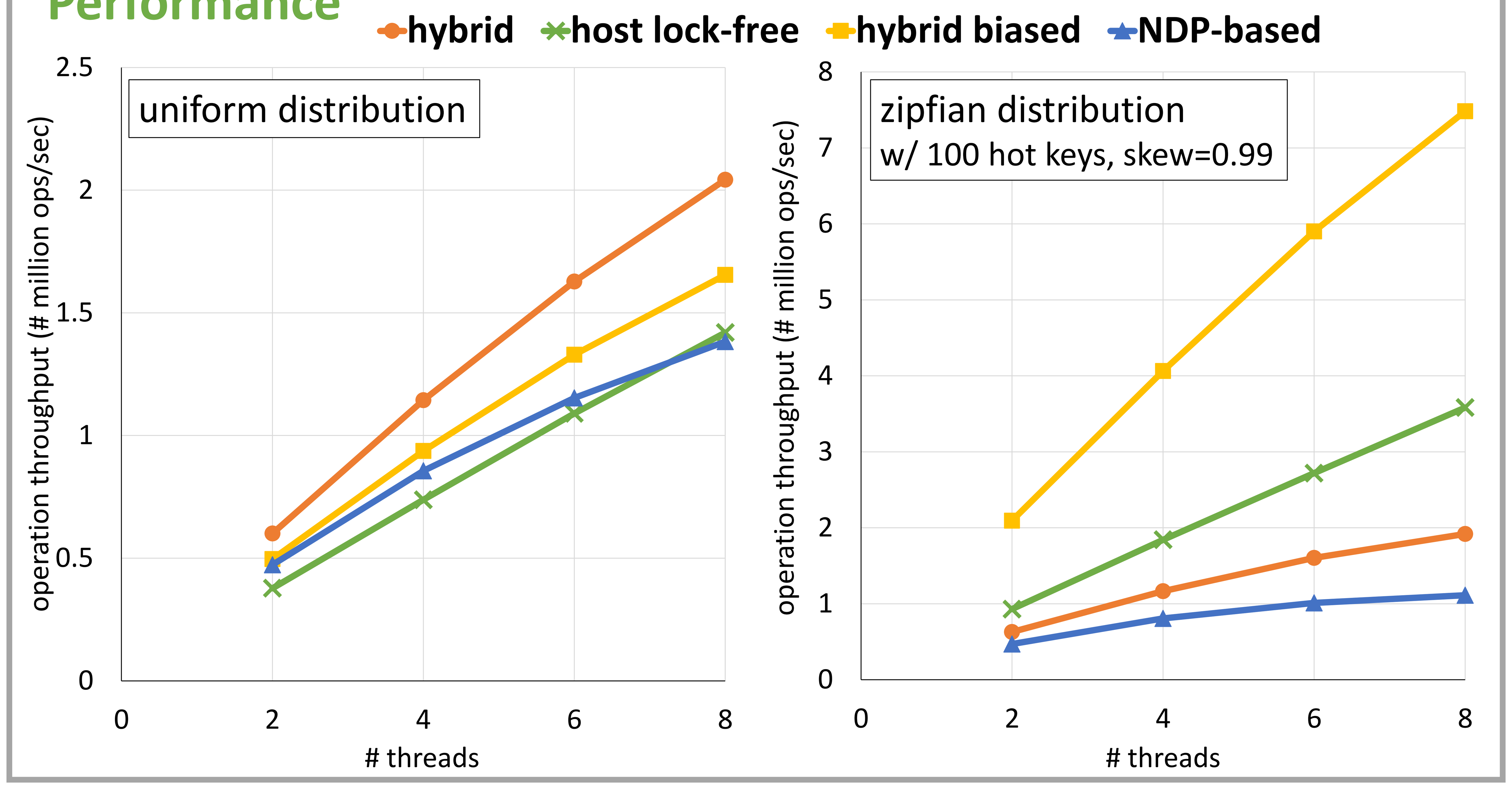
- **higher levels** as **lock-free skiplist** – "pinned" to host cache
- **lower levels** as **NDP-based skiplist** – prevents cache pollution from infrequently-accessed nodes while providing fast access
- **linearizable**: linearization points of add/remove in NDP portion



Hybrid Biased Skiplist

- **dynamically promotes/demotes nodes** between lock-free and NDP portions **based on node popularity**
- yields higher throughput in realistic workloads that have skew in data access

Performance



Future Work

- **HW/SW co-design** for **efficiently identifying node popularity**:
  - **overhead in determining node popularity** causes hybrid biased skiplist performance to suffer in uniform distribution workload
- **evaluation w/ realistic workloads** – not only performance but also **energy consumption**

Evaluation Setup

- initial skiplist size: 0.5GB (2^{22} nodes)
- operations divided as:
  90% contains, 9% add, 1% remove

Evaluation Framework

Brown-SMCSim: gem5 full-system simulator

- https://github.com/jiwon-choe/Brown-SMCSim
- 8 host processors
  - ARMv7 Cortex A15, 1 thread/core
- 64kB L1 dcache/core
  - 256B/block, 2-way set associative, 0.8ns access latency
- 2MB shared L2 cache
  - 256B/block, 8-way set associative, 1.8ns access latency
- 1GB host-accessible main memory
  - close-page row-buffer-management policy
- 8 NDP core-vault pairs
  - NDP core: ARMv7 Cortex A15, 40kB scratchpad memory, 8kB reserved for memory-map (communication w/ host)
  - NDP vault: 128MB/vault, open-page policy
- DRAM timing:
  - t_{RP}=13.75ns, t_{RCD}=13.75ns, t_{CL}=13.75ns, t_{BURST}=3.2ns