Hierarchical Content Stores in High-Speed ICN Routers: Emulation and Prototype Implementation

Rodrigo B. Mansilha[1,2,6], Lorenzo Saino[3,6], Marinho P. Barcellos[2], Massimo Gallo[4,6], Emilio Leonardi[5], Diego Perino[4,6], Dario Rossi[1,6]

[1]**Telecom ParisTech, France**

[2]**Federal Univ. of Rio Grande do Sul, Brazil**

[3]University College London, UK

[4]Alcatel-Lucent, France

[5]Politecnico di Torino, Italy

[6]**Lincs, France**

ACM ICN'15, October, 1st, 2015, San Francisco, CA, USA

Context

- The success of the ICN paradigm depends on routers with large caches able to operate at line speed
- It's challenging to satisfy both requirements together
- Maximum size of *Content Store* (**CS**) that can sustain a data rate of 10 Gbps is estimated to be around 10 GB[1,2]

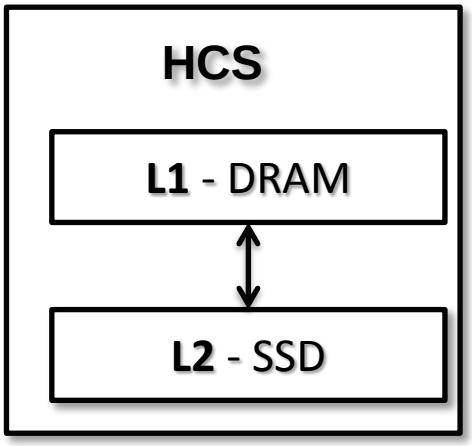
	Speed	Size	Cost
DRAM	O(10ns)	O(10GB)	O(10$/GB)
SSD	O(10us)	O(1TB)	O(1$/GB)

[1] D. Perino and M. Varvello. A reality check for content centric networking. In ACM SIGCOMM, ICN Workshop, 2011

[2] S. Arianfar and P. Nikander. Packet-level Caching for Information-centric Networking. In ACM SIGCOMM, ReArch Workshop, 2010

State of the Art

- *Hierarchical Content Stores* (**HCS**) have been proposed to bypass that limit by exploring arrival pattern in ICN[1]
  - Prefetching batch of chunks
  - To a faster but a smaller memory (L1)
  - From a larger but slow memory (L2)
- Micro-benchmarking of SSD technologies to assess their suitability for the HCS purpose[2]



[1] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi. Multi-Terabyte and multi-Gbps information centric routers. In IEEE INFOCOM, 2014

[2] W. So, T. Chung, H. Yuan, D. Oran, and M. Stapp. Toward terabyte-scale caching with ssd in a named data networking router. In ACM/IEEE ANCS, Poster session, 2014

Contribution

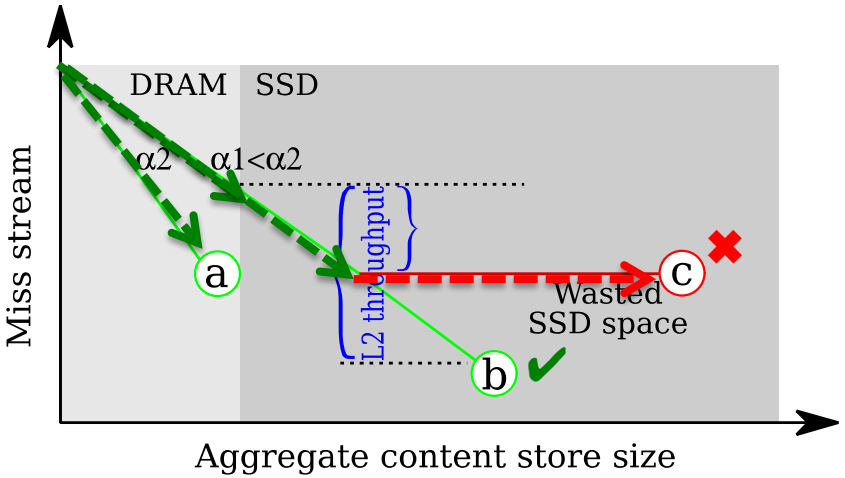
1. Investigate HCS employing two complementary methodologies, namely emulation and prototype
2. Carry out an extensive emulation of the design space using open-source software (NFD)
3. Present a complete system implementation (DPDK), in contrast with the benchmark of a specific component as in previous work

Outline

- Introduction
- HCS Overview
- Emulation investigation
- Prototype investigation
- Conclusion

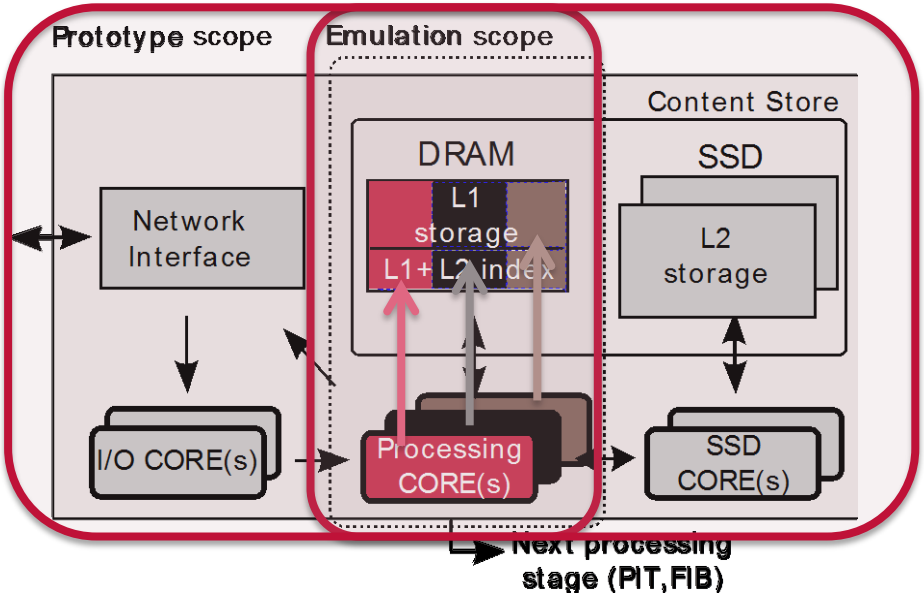
Performance Goal

- CS miss stream decreases as its size increases, depending on the popularity distribution
- In HCS, this holds up to a point at which the system is bottlenecked by L2 throughput
- Read throughput from L2 depends on hit at L2
  - Increasing L2 size also increases its demand
  - After the point, increasing SSDs brings no benefits
- We're targeting b, and avoiding c



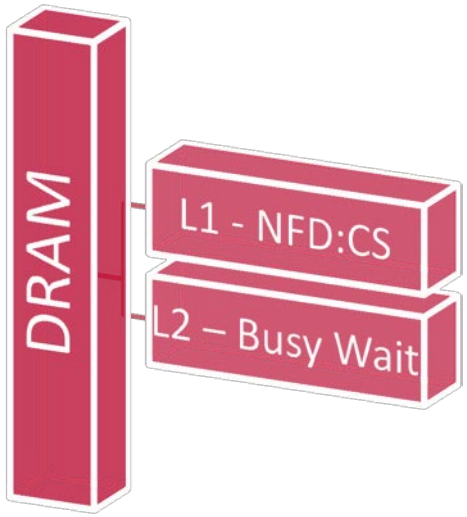
System Design

- Parallelism avoiding contention
  - Each thread manages an isolated HCS
  - Requests are classified among threads according to a given hash function
  - Chunks of a specific batch are always handled by the same thread
- Two instantiations
  1. Emulation (NFD-HCS)
  2. Prototype (DPDK-HCS)



Emulation Design

- Layer 1 instantiates NFD::CS
- Layer 2 emulates SSD
  - Delay = Batch size / emulated throughput
  - Busy wait more reliable than timers
- Serial read algorithm
  - In case of L1 Hit
    - Return chunk
  - In case of L1 Miss
    - Read batch of chunks of L2
    - Insert batch of chunks on L1
    - Return chunk



✔ Functional with real code

✔ Explore design space

✖ Limits |L1|+|L2| size to DRAM size

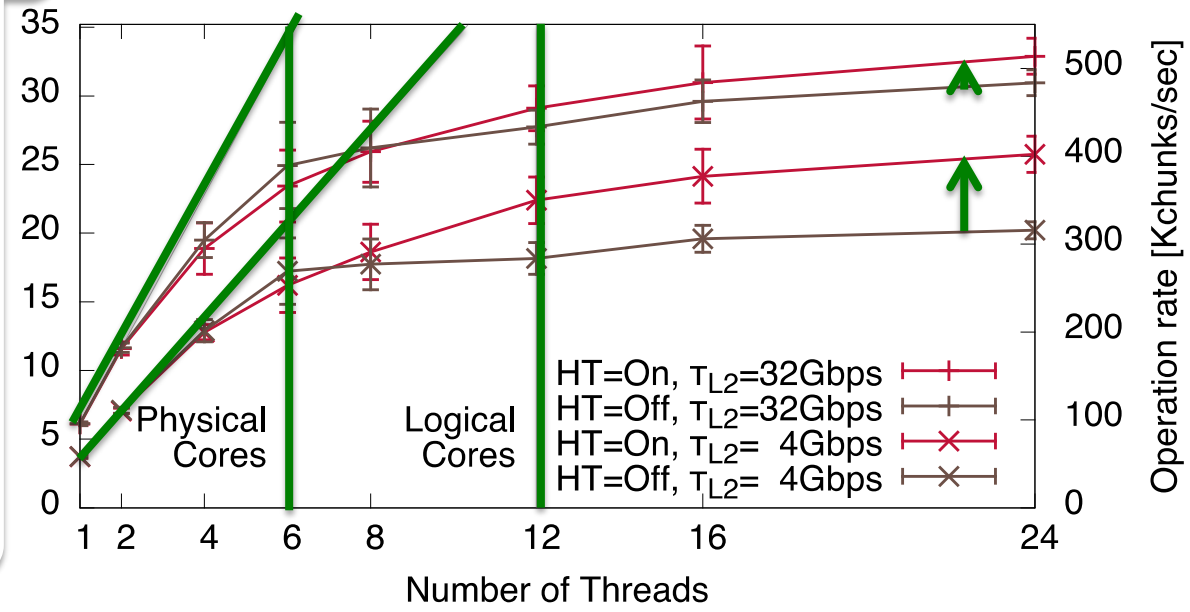
Emulation Evaluation

- Baseline NFD Performance
- NFD-HCS Performance
  - Validating Emulation via Analytical Modeling
  - Inferring software bottlenecks
  - Multi-threaded HCS Performance
- Sensitivity Analysis
  - Software design: serial vs parallel
  - Hardware: L2 throughput
  - Hardware: Off-the-shelf PC

Parameter	Range
Workload	[real,seq,unif]
L1 Size	[1-10] GB
Hyperthreading	[on, off]
# Threads	[1-24]
L2 throughput	[1-32] Gbps
System	[local, cloud]

Multi-threaded HCS Performance

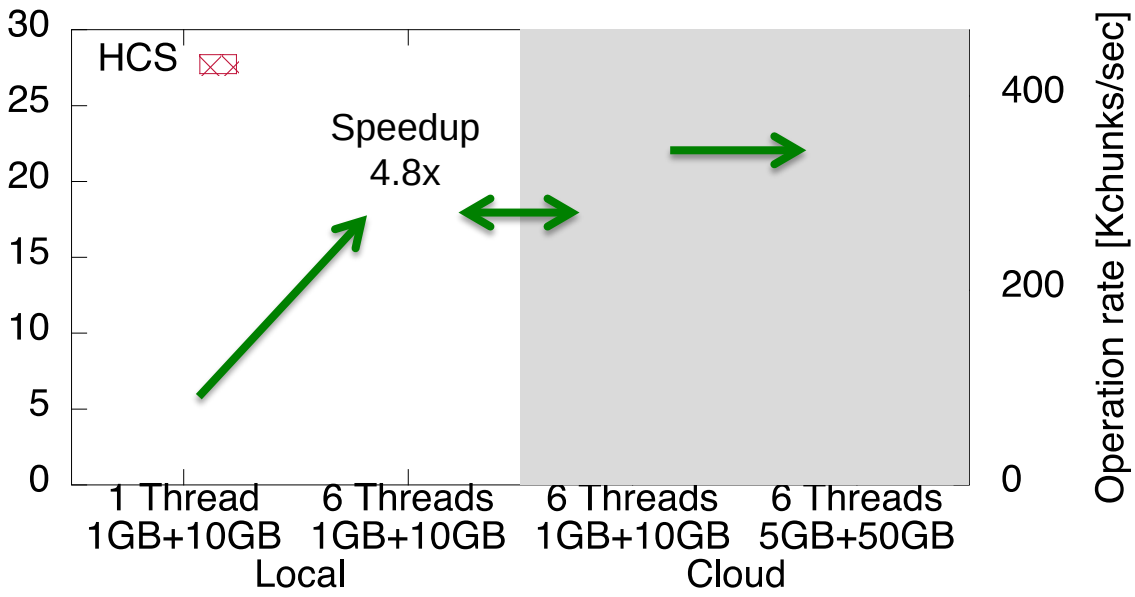
Logarithmic gains with number of threads



- Linear returns up to 2 threads
- Knee in the curve where # threads = # cores
- Hyper-threading is advantageous where # threads >> # cores

Sensitivity Analysis: Off-the-shelf PC

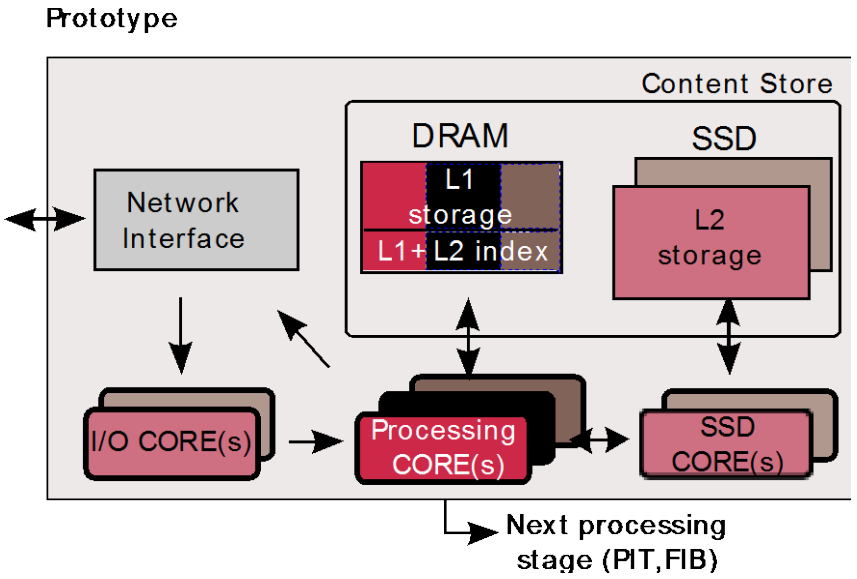
HCS exceeds 10Gbps by exploiting parallelism



- Multithread needed to achieve 10Gbps
- Emulation results are not biased
- Confirms memory scalability of HCS

Prototype Implementation

- **NIC.** DPDK enables zero-copy packet processing
- **Batching**. Performs all I/O operations over batches instead of single chunks
- **SSD I/O.** Set parameters such as Queue depth (i.e., the number of access operations executed in parallel by the SSD controller)
- Also, multi-threading, load balancing, lookup, etc.



Experimental Evaluation

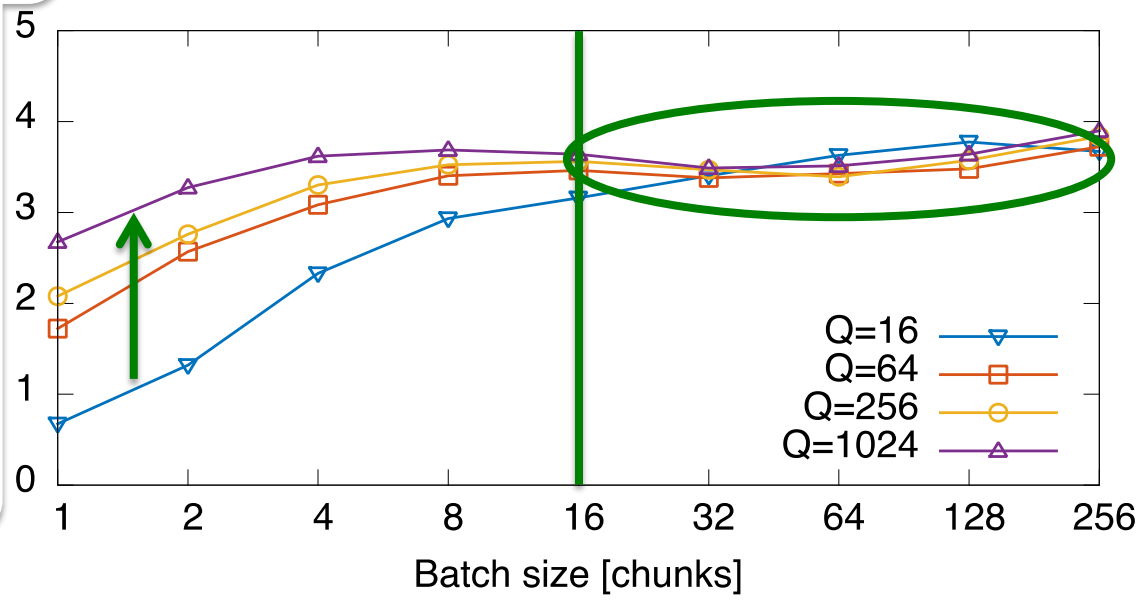
- Baseline SSD performance[1]
  - Throughput vs read/write mix
  - Throughput vs queue depth
- DPDK-HCS performance
  - Number of SSDs and L1 Size

Parameter	Range
Batch size	[1-256]
Read/Write mix	[0-100]%
Queue depth	[16-1024]
L1 Size	[5-20] GB
# SSDs (200 GB)	[1, 2]

[1] Similarly to: W. So, T. Chung, H. Yuan, D. Oran, and M. Stapp. Toward terabyte-scale caching with ssd in a named data networking router. In ACM/IEEE ANCS, Poster session, 2014

SSD: Throughput vs Queue Depth

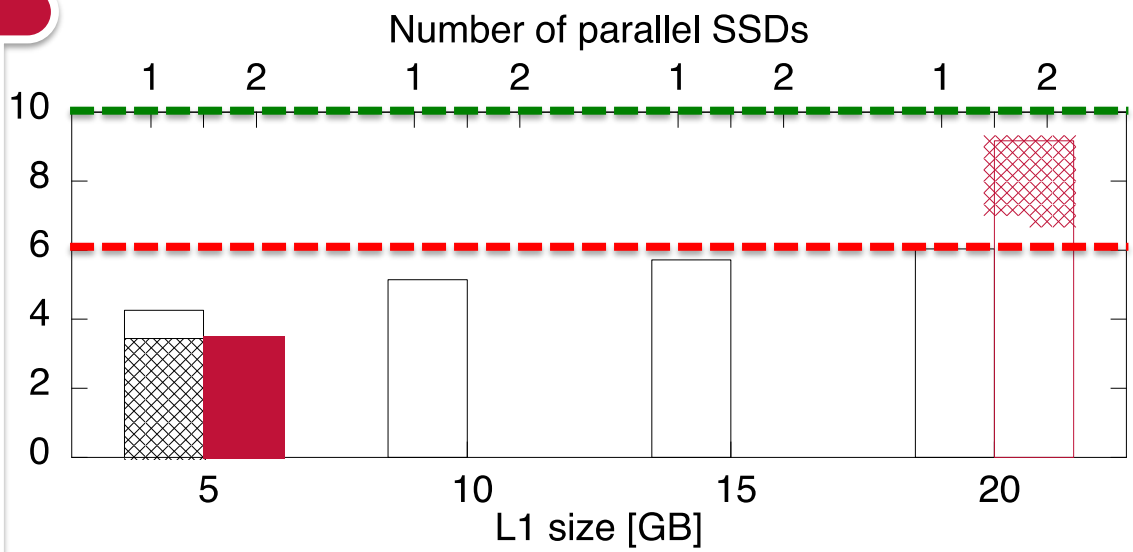
B=16 & Q=16 are good values considering our settings



- Workload = synthetical, 50% read/write mix
- For small batches, a large SSD queue is beneficial as it improves throughput by increasing the number of parallel SSD operations
- If the batch size is large enough (B=16),
  - Increasing the Q>16 does not provide significant throughput benefits
  - Yields a latency penalty

DPDK-HCS Performance

Thanks to the parallel design, we were able to achieve 10Gbps



- *B*=16 chunks, *Q*=16 batches, Workload=Real trace
- 1 SSD cannot sustain line speed
- 2 SSD drives can sustain a line rate of 10Gbps

Conclusion

- Summary
  - We explore the issue of designing Large Caches for High-speed ICN routers
  - We advance the state of the art by providing emulation- and prototype-based studies about HCS
- Take away message
  - *Line-rate O(10 Gbps) operation of HCS equipped with O(10GB) L1-DRAM and O(1TB) L2-SSD memory technologies can be achieved in practice*

On Going Work

- Emulation investigation
  - Expand workload scenarios by advancing the emulation techniques
- Experimental investigation
  - Increase the DPDK-HCS performance by, for example, reducing stress on SSD by requiring multiple L1 hits before writing to L2

The End

- Questions?
- Thanks!

Backup Slides

Emulation Settings

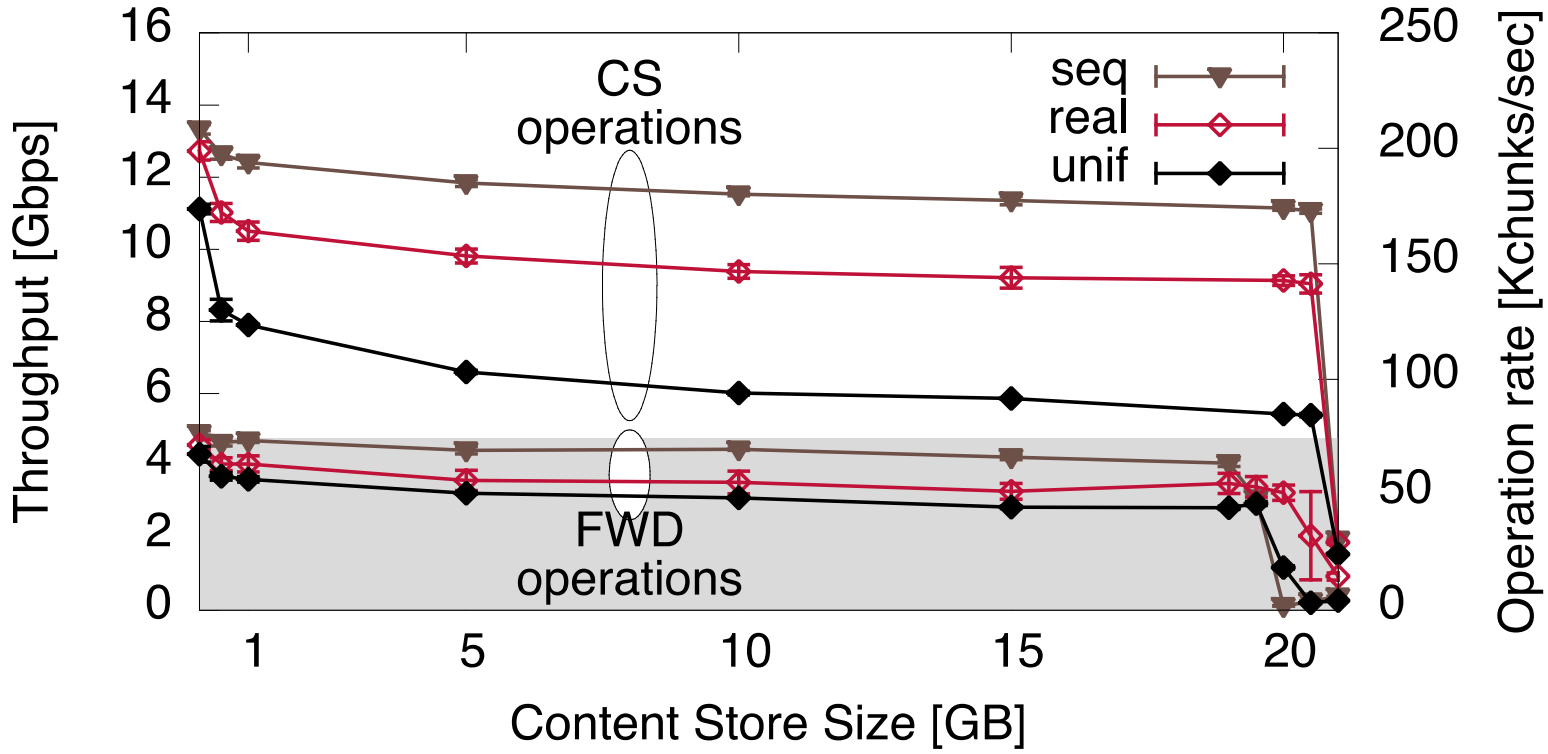
Table 1: Emulation settings

	Meaning	Param	Range
Software	Batch Size	B	10 chunks
	L1 Size	$\|L1\|$	[100MB-10GB]
	L2 Size	$\|L2\|$	10 GB
	L2 Throughput	τ_{L2}	[1,32] Gbps
	Chunk size	$\|c\|$	8KB

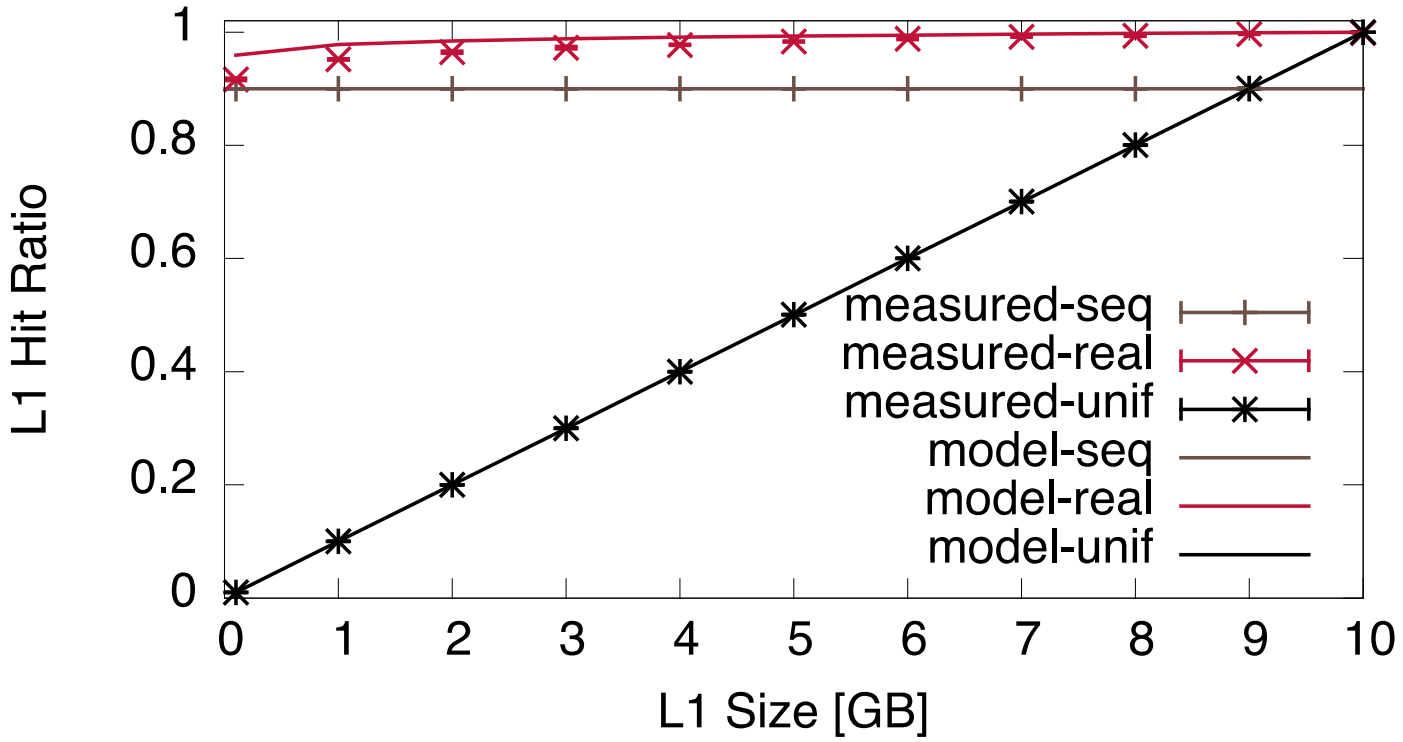
	Label	Param	Value
Hardware	Local	CPU	1.90 GHz Intel E52420
		NUMA	1 node, 6 cores
		RAM	32GB - 1333 MHz (0.8 ns)
		Opts.	CPU Gov., Hyper-threading
	Cloud (Microsoft Azure G3)	CPU	2.00 GHz Intel E52698B
		NUMA	1 node, 8 cores
		RAM	112GB (speed unknown)
		Opts.	None

	Meaning	Param	Value
Real workload	Catalog size	$\|C\|$	(up to) 10^3 objects
	Request arrival rate	λ	1 Hz
	Zipf skew	α	1
	Streaming rate		512 Kbps (8 chunks/s)
	Streaming duration		160 seconds
	Stream size		10.25 MB (1,280 chunks)

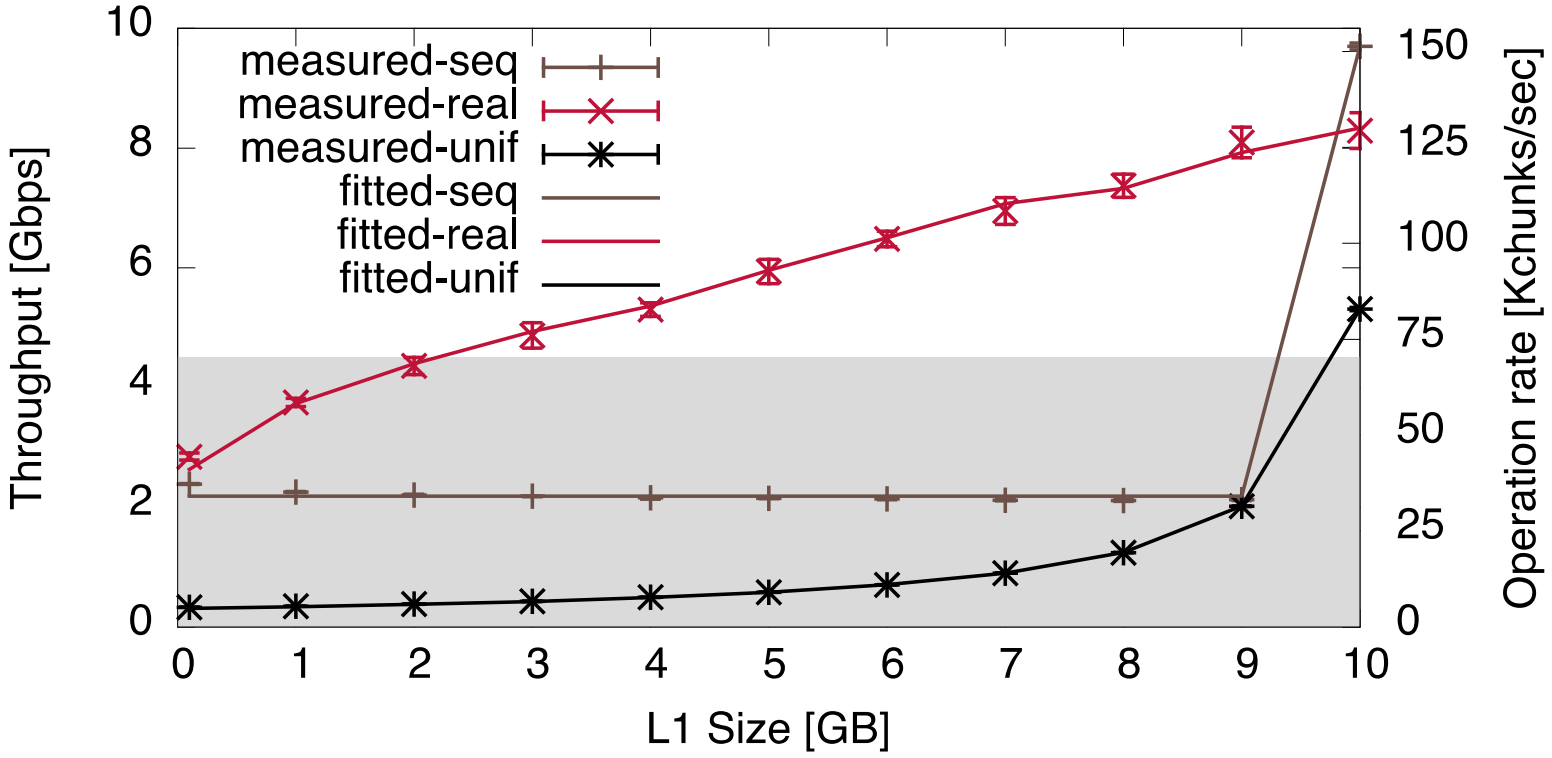
Baseline NFD Performance



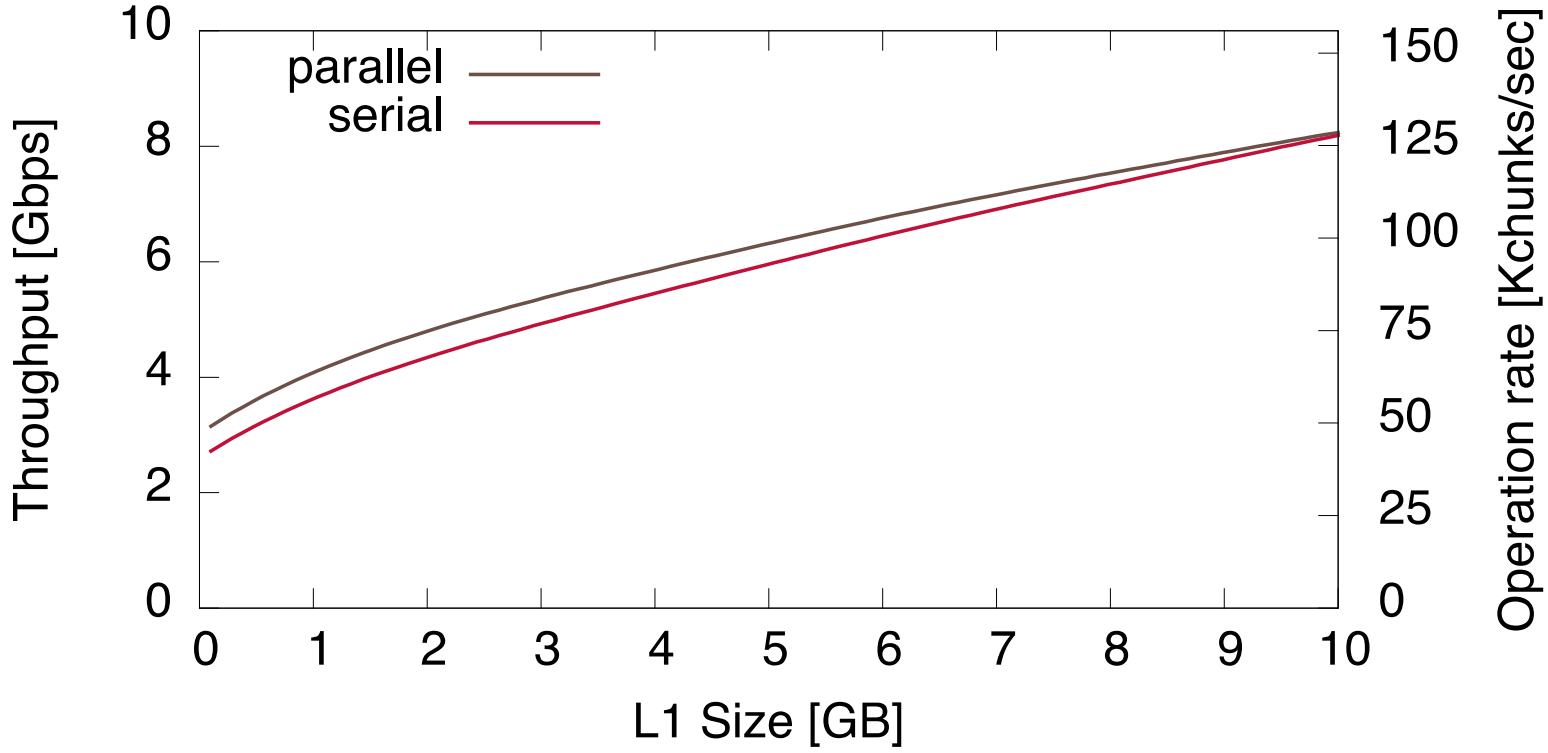
Validating Emulation



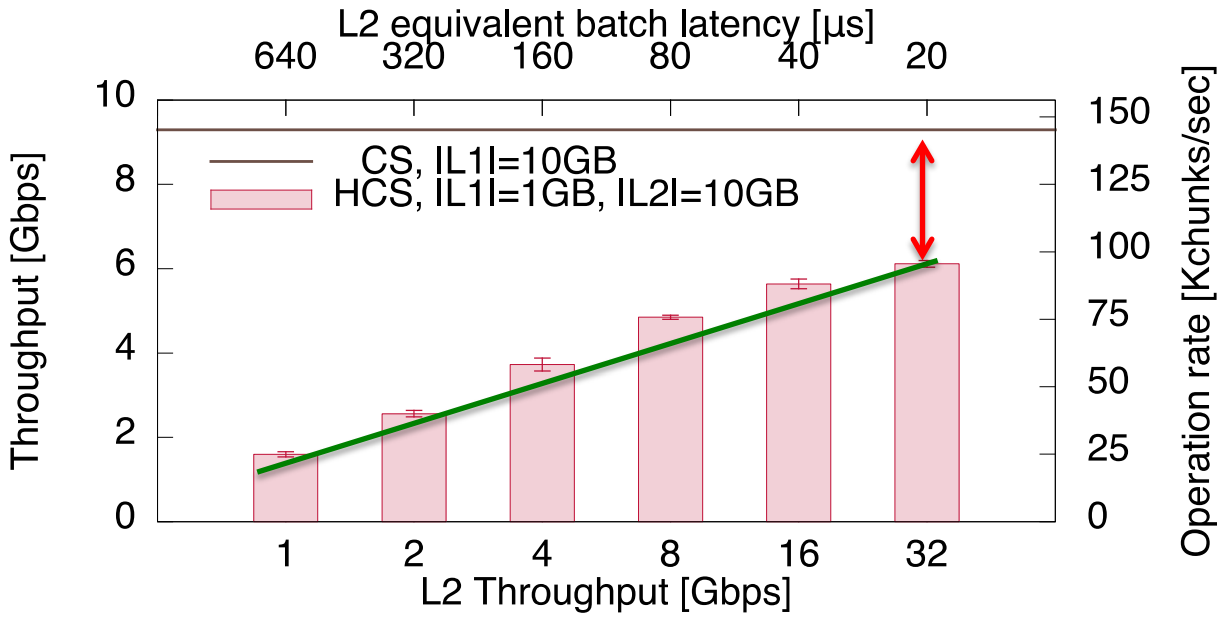
Inferring Software Bottlenecks



Software: Design Space

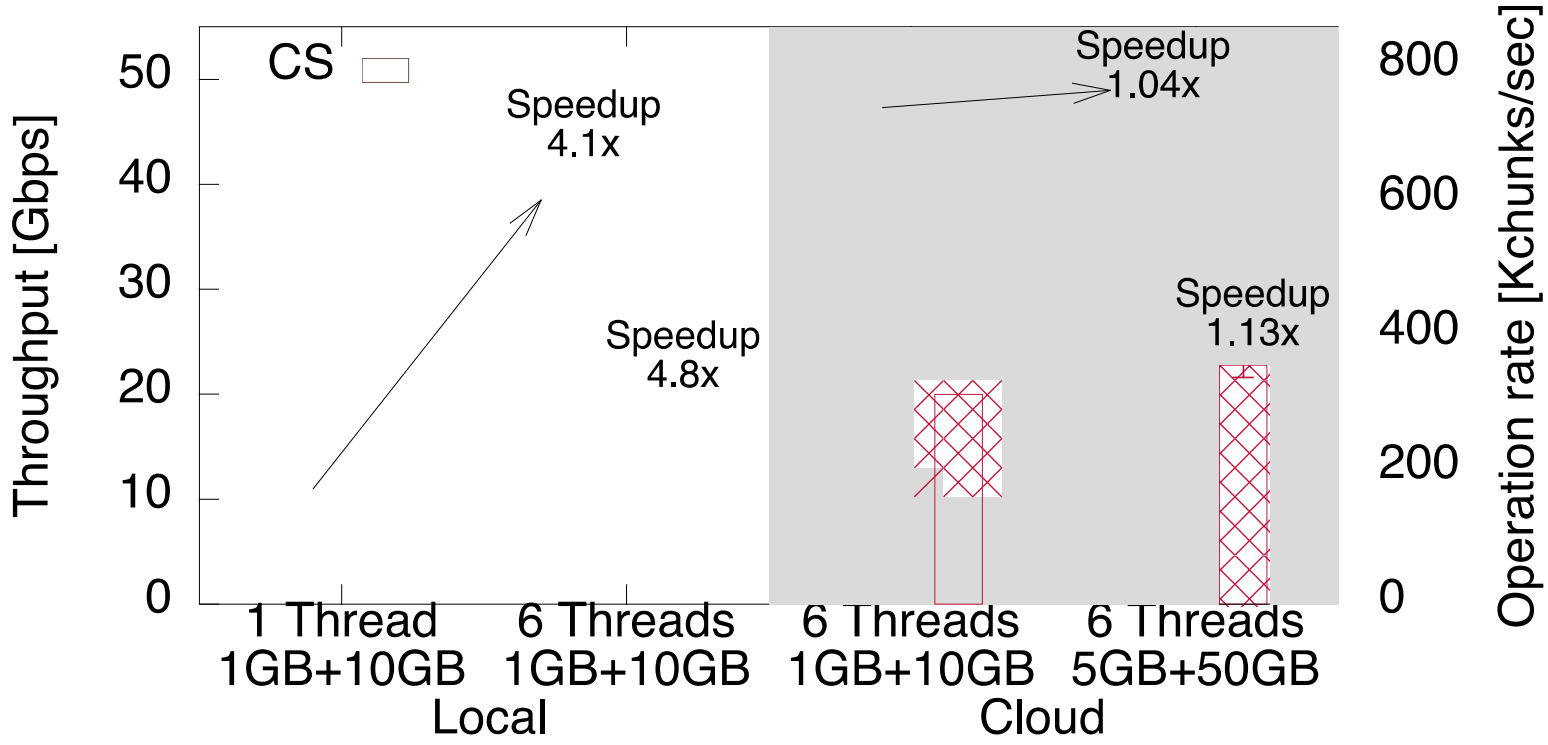


Sensitivity Analysis: L2 throughput



- Single thread
- Logarithmic return for the system as a function of L2
- HCS approaches but not reaches CS performance
- Likely due to software bottlenecks tied to the additional overhead of handling a second memory layer

Sensitivity Analysis: Off-the-shelf PC



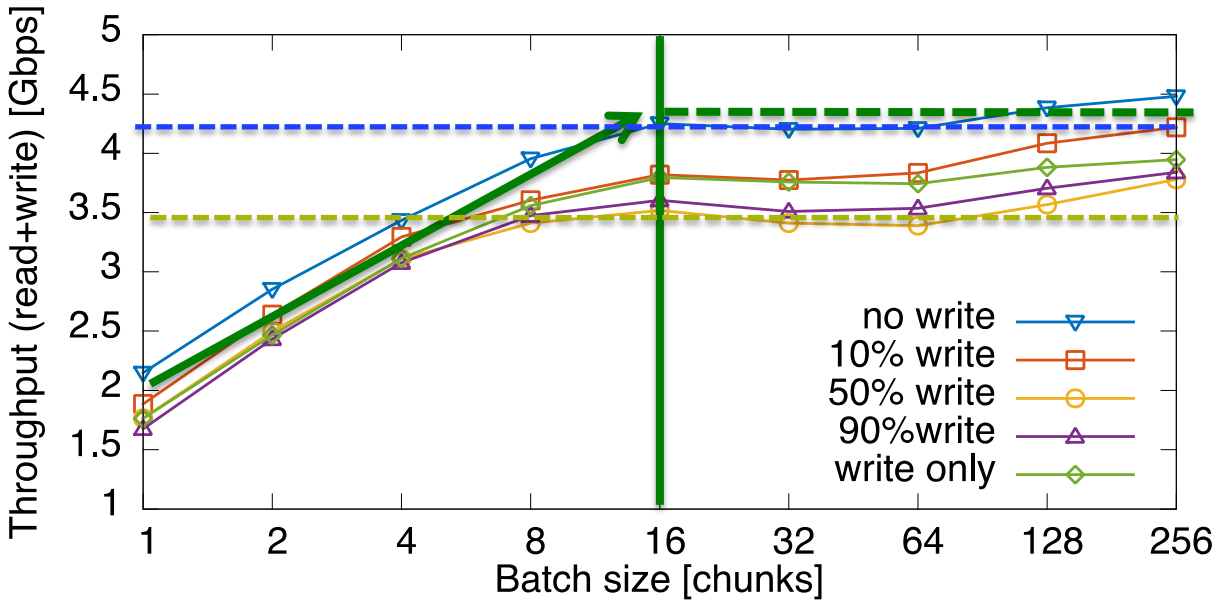
- Multithread needed to achieve 10Gbps
- Emulation results are not biased
- Confirms memory scalability of HCS

Experimental Settings

Table 2: Experimental settings

Label	Param	Value
Hardware	CPU	(2 ×) Intel Xeon E5540, 2.53 GHz
	NUMA	2 nodes, 4 cores/node
	RAM	32GB - 1.3GHz (0.8 ns)
	SSD	(2 ×) 200 GB HP enterprise SAS
	NIC	Dual-port 10GbE Intel 82599EB
Software	OS	Ubuntu 12.04 LTS
Workload	Catalog size	1.3M items
	Item size	10 MB
	Chunk size	8 KB
	Zipf skew	1

SSD: Throghput vs Read/write Mix



- Queue size *Q*=64 batches
- Without writing, performance near to declared external data rate
- With 50% write, throughput decreases to ~3.5 Gbps
- Threshold at Batch=16 chunks
  - Near-maximum SSD throughput for all read/write mixes.
  - Yields a latency penalty