

the 5-minute rule for the cloud: caching in analytics systems

Kira Duwe, Angelos Anadiotis, Andrew Lamb,
Lucas Lersch, Boaz Leskes, Daniel Ritter, *Pınar Tözün*



Robust Query Processing in the Cloud

Dagstuhl Seminar 24101

when do we need caches
in the cloud?

can we eliminate them?



spoilers – caching is beneficial in the cloud ...

if an object needs to be accessed

→ ***7 times per second*** in a ***latency-insensitive workload***

→ ***2 times per hour*** in a ***latency-sensitive workload***

**if you care about latency in disaggregated architectures,
you are going to need an object store cache!**

THE 5 MINUTE RULE FOR TRADING MEMORY FOR DISC ACCESSES
and
THE 10 BYTE RULE FOR TRADING MEMORY FOR CPU TIME

Jim Gray
Franco Putzolu
Tandem Computers, Cupertino, CA, USA

ABSTRACT: If an item is accessed frequently enough, it should be main memory resident. For current technology, "frequently enough" means about every five minutes.

Along a similar vein, one can frequently trade memory space for cpu time. For example, bits can be packed in a byte at the expense of extra instructions to extract the bits. It makes economic sense to spend ten bytes of main memory to save one instruction per second.

These results depend on current price ratios of processors, memory and disc accesses. These ratios are changing and hence the constants in the rules are changing.

The derivation of the five minute rule goes as follows: A disc, and half a controller comfortably deliver 15 random accesses per second and are priced at about 15K\$ [Tandem]. So the price per disc access per second is about 1K\$/a/s. The extra CPU and channel cost for supporting a disc is 1K\$/a/s. So one disc access per second costs about 2K\$/a/s.

A megabyte of main memory costs about 5K\$, so a kilobyte costs 5\$.

If making a 1Kb data record main-memory resident saves 1a/s, then it saves about 2K\$ worth of disc accesses at a cost of 5\$, a good deal. If it saves .1a/s then it saves about 200\$, still a good deal. Continuing this, the break-even point is

when does it make economic sense to cache disk pages in DRAM? → if they are reused at least every 5mins.

The five-minute rule ten years later, and other computer storage rules of thumb

Authors:  [Jim Gray](#),  [Goetz Graefe](#) | [Authors Info & Claims](#)

[ACM SIGMOD Record, Volume 26, Issue 4](#) • Pages 63 - 68 • <https://doi.org/10.1145/271074.271094>

Published: 01 December 1997 [Publication History](#)



The Five-Minute Rule 20 Years Later: and How Flash Memory Changes the Rules: The old rule continues to evolve, while flash memory adds two new rules.


Author:  [Goetz Graefe](#) | [Authors Info & Claims](#)

[Queue, Volume 6, Issue 4](#) • Pages 40 - 52 • <https://doi.org/10.1145/1413254.1413264>

Published: 01 July 2008 [Publication History](#)



The five-minute rule 30 years later and its impact on the storage hierarchy

Authors:  [Raja Appuswamy](#),  [Goetz Graefe](#),  [Renata Borovica-Gajic](#),  [Anastasia Ailamaki](#) | [Authors Info & Claims](#)

[Communications of the ACM, Volume 62, Issue 11](#) • Pages 114 - 120 • <https://doi.org/10.1145/3318163>


Published: 24 October 2019 [Publication History](#)



Tracing the evolution of the five-minute rule to help identify imminent changes in the design of data management engines.

BY RAJA APPUSWAMY, GOETZ GRAEFE,
RENATA BOROVIKA-GAJIC, AND ANASTASIA AILAMAKI

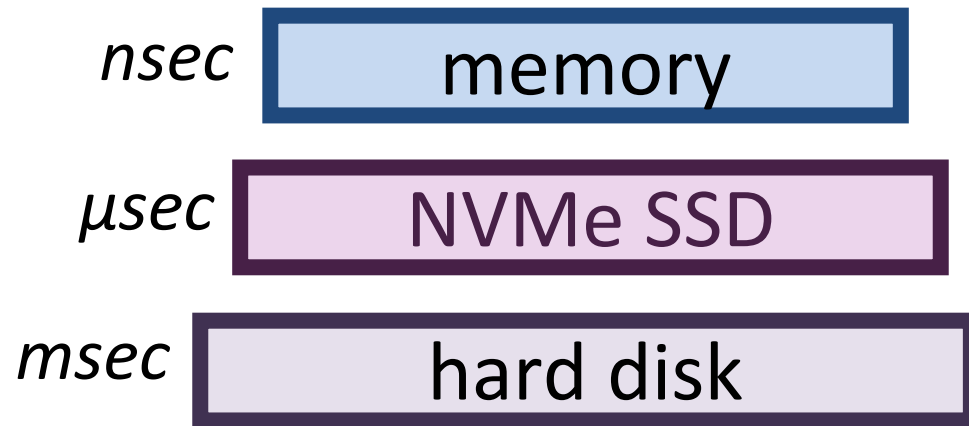
The Five-Minute Rule 30 Years Later and Its Impact on the Storage Hierarchy

Finally, with widespread adoption of cloud computing, the modern enterprise storage hierarchy not only spans several storage devices, but also different geographic locations from direct-attached low-latency devices, through network-attached storage servers, to cloud-hosted storage services. The price-performance characteristics of these storage configurations vary dramatically depending not only on the storage media used, but also on other factors like the total capacity of data stored, the frequency and granularity of I/O operations used to access the data, the read-write ratio, the duration of data storage, and the cloud service provider used, to name a few. Given the multitude of factors, determining the break-even interval for cloud storage is a complicated problem that we did not consider in this work. Thus, another interesting avenue of future work is extending the five-minute rule to such a distributed cloud storage setting. 

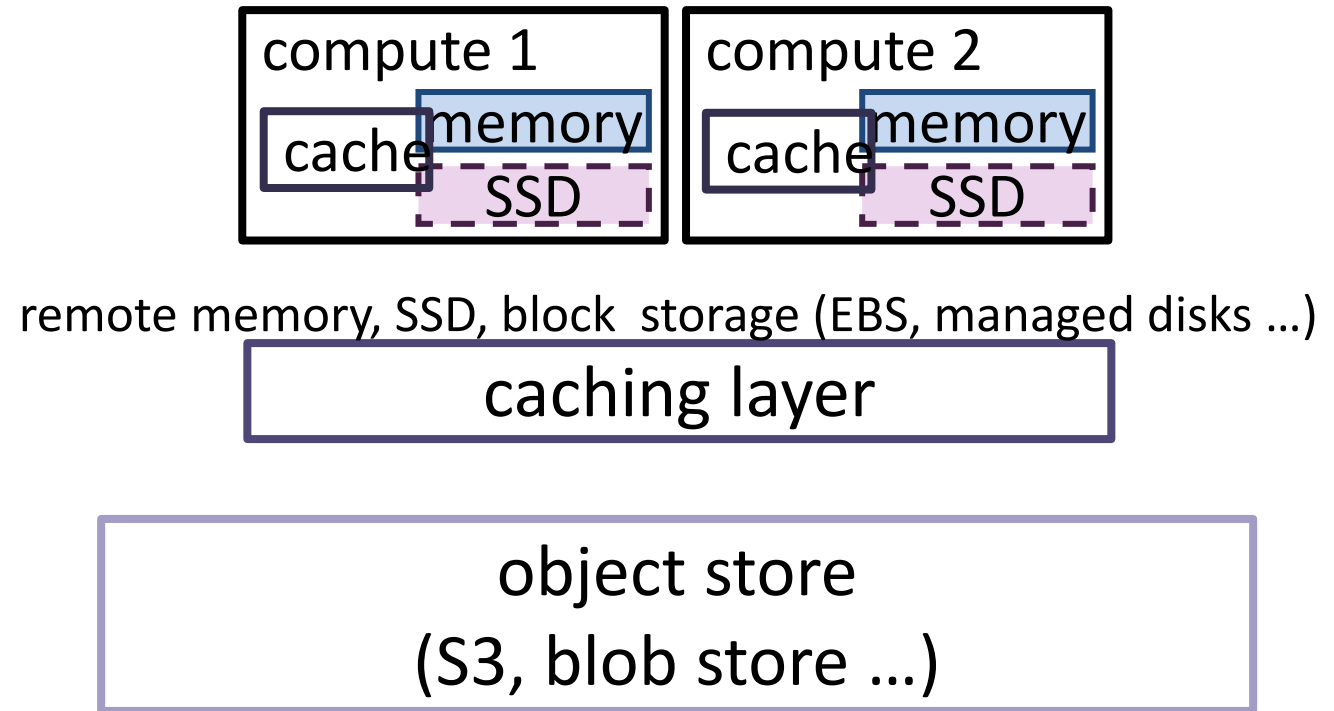
what changes for *the cloud*?

what changes for *the cloud*? – storage hierarchy

conventional 5-min rule



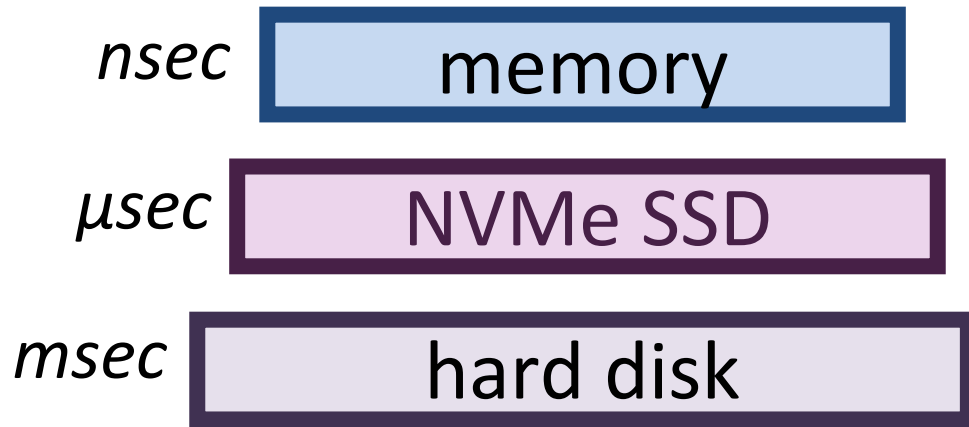
cloud



- **compute & storage disaggregation**
- **object store has the ground-truth for data**

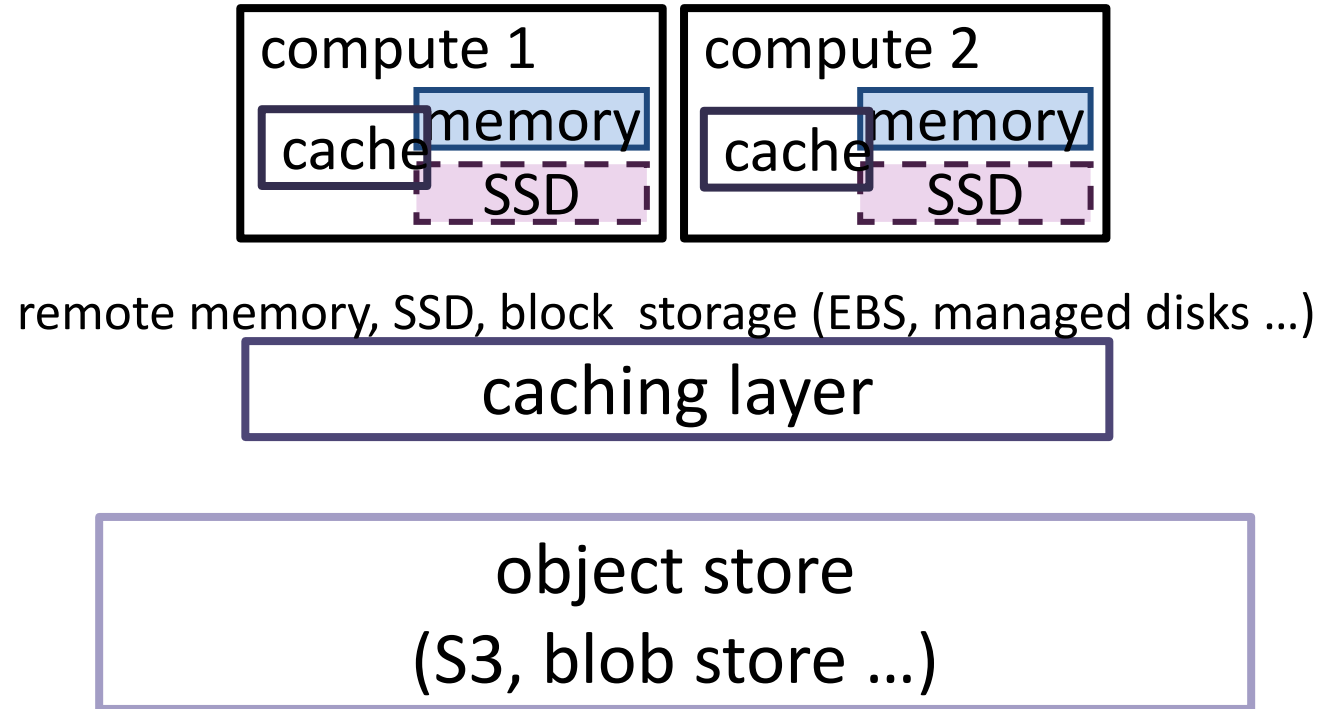
what changes for *the cloud*? – costs

conventional 5-min rule



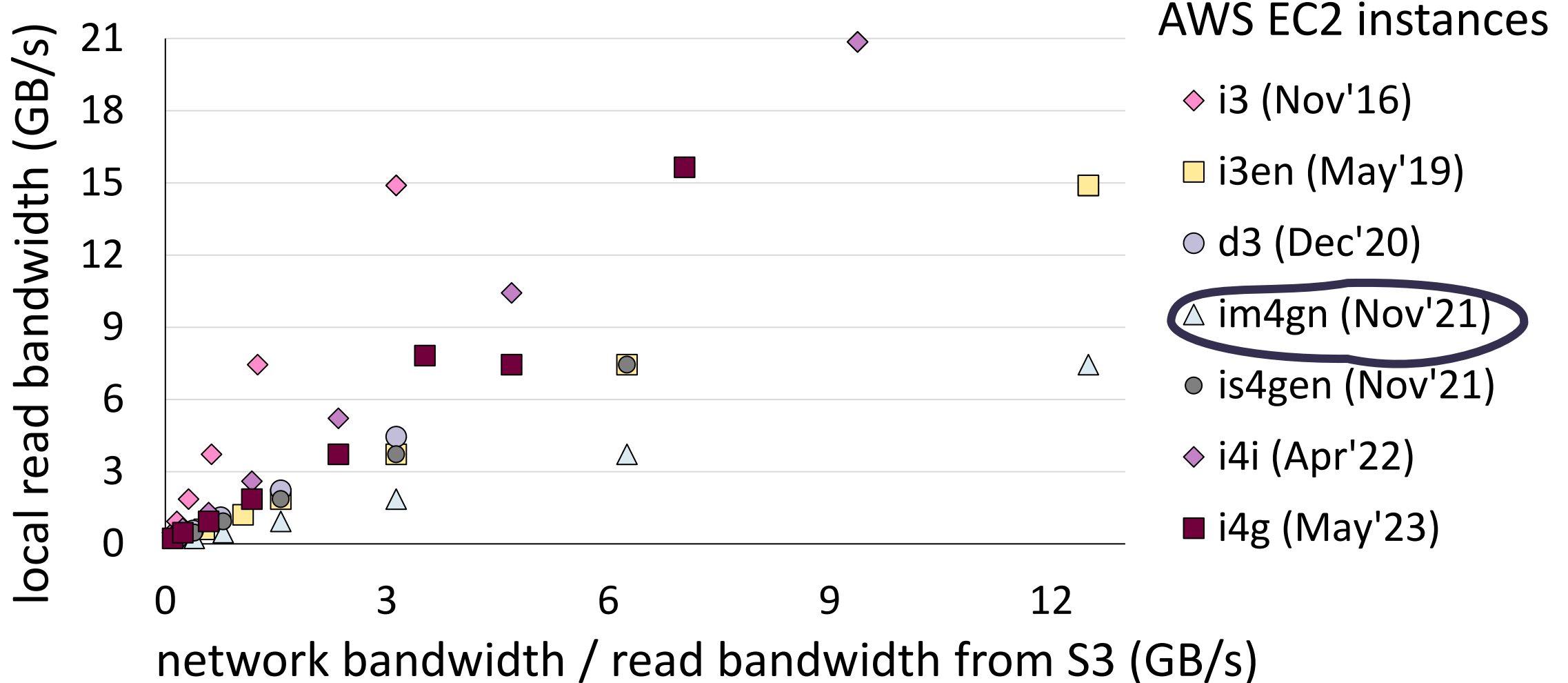
storage devices

cloud



- **compute instances**
- **data storage**
- **object store accesses**

what changes for *the cloud*? – network



some compute instances have higher bandwidth than local storage, but this doesn't guarantee stable or low latency!

5-min rule in the cloud – for cloud analytics

**Given a latency target,
how often must an application access
an object to justify caching instead of
directly fetching from object storage?**

the model – latency-insensitive cases

cost of not caching

■ cost per object store request ✕ (number of requests ■ 1)

cost of caching

■ (hourly storage cost per GB + hourly instance cost per GB)

✕ cache size in GB

✕ lifetime of cache in hours

+ cache miss rate ✕ number of requests ✕ cost per object store request

the model – latency-insensitive cases

cost of not caching

■ cost per object store request ✖ (number of requests ■ 1)

cost of caching – separate cache instances

■ (hourly storage cost per GB + hourly instance cost per GB)

✖ cache size in GB

✖ lifetime of cache in hours

+ cache miss rate ✖ number of requests ✖ cost per object store request

cost of caching – same instance

■ cost of caching at separate instance ■ instance cost without cache

the model – latency-sensitive cases

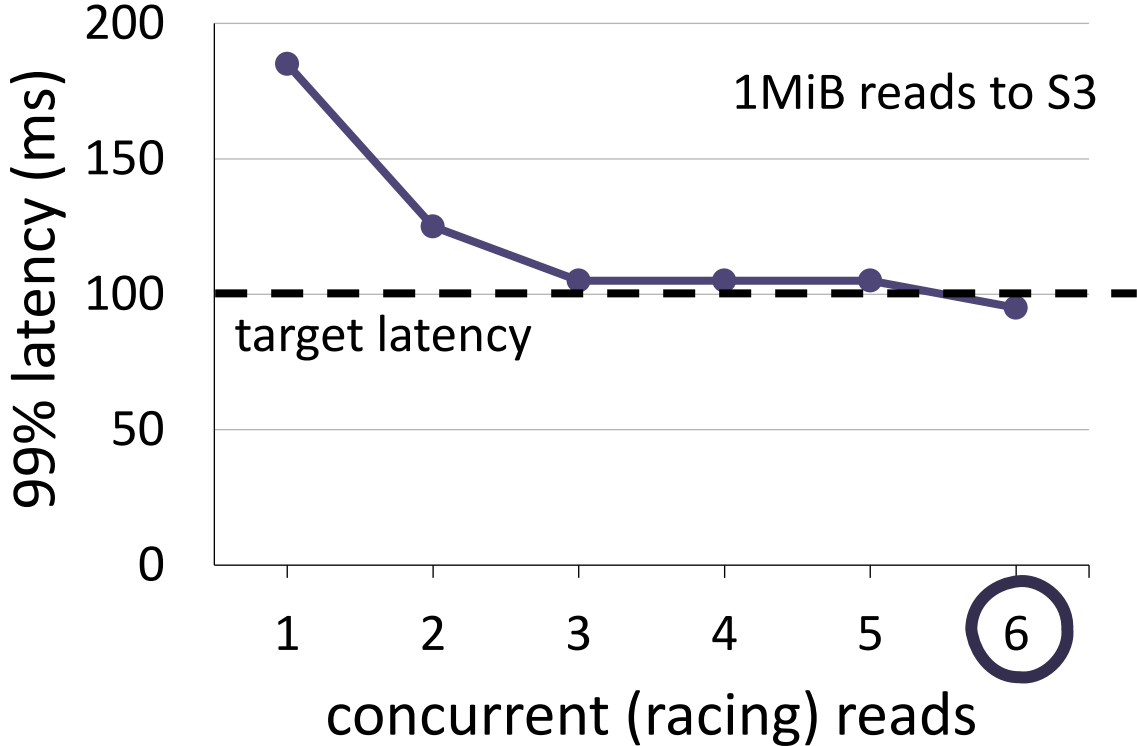
cost of not caching

- ▬ cost per object store request
 - ✘ (number of requests - 1)
 - ✘ concurrent reads to guarantee latency
- only term that requires measurement!

cost of caching

→ same as previous

racing reads to reach the latency target
→ concurrent requests for the object
→ proceed with the first response, ignore the rest






preliminary evaluation

on AWS m7 instances
with ARM Graviton processors


S3 cost per request	0.0000004\$
EBS monthly storage cost per GB	0.08\$
hourly on-demand compute instance costs	
m7g with EBS	0.0408\$
m7gf with 59GiB NVMe SSD	0.0534\$

object store access latency values

Exploiting Cloud Object Storage for High-Performance Analytics

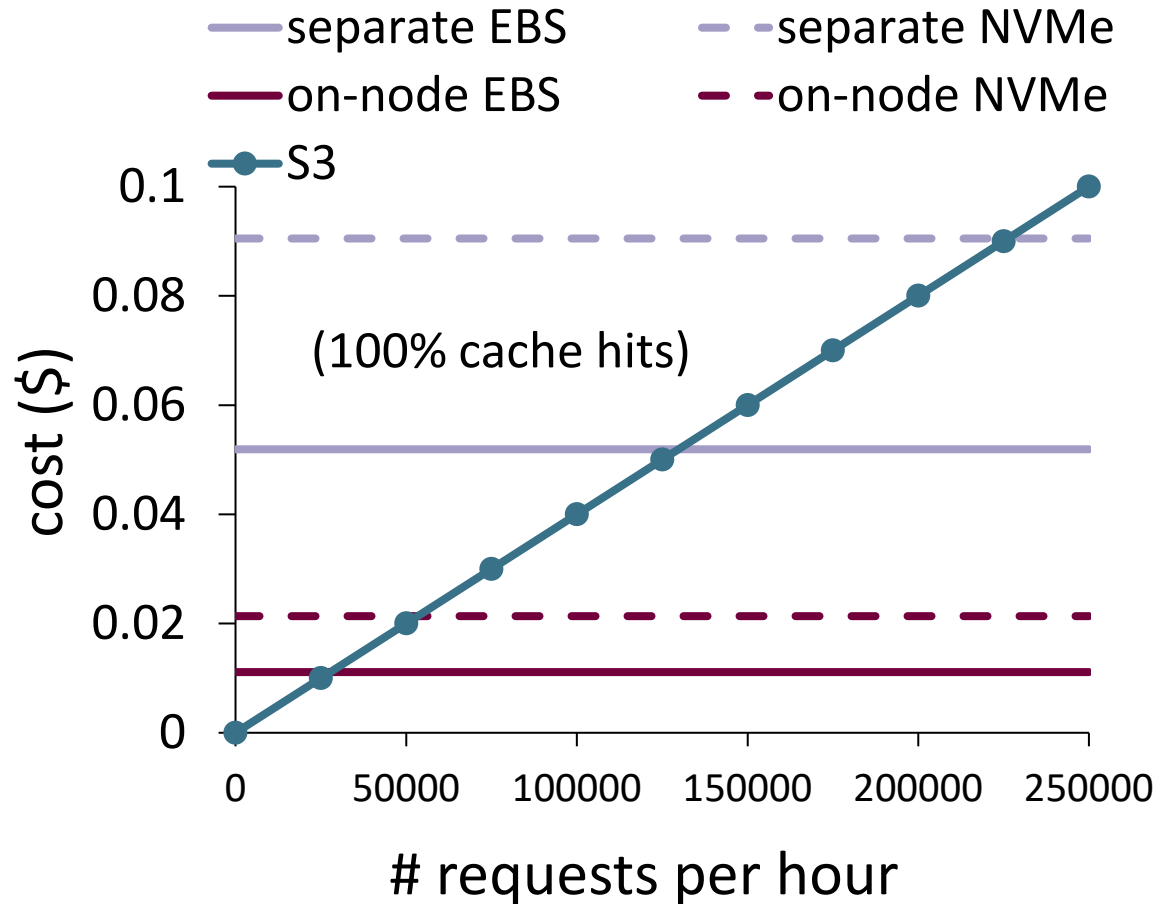
Authors:  [Dominik Durner](#),  [Viktor Leis](#),  [Thomas Neumann](#) | [Authors Info & Claims](#)

[Proceedings of the VLDB Endowment, Volume 16, Issue 11](#) • Pages 2769 - 2782 • <https://doi.org/10.14778/3611479.3611486>

Published: 01 July 2023 [Publication History](#) 

results – when does it make sense to cache?

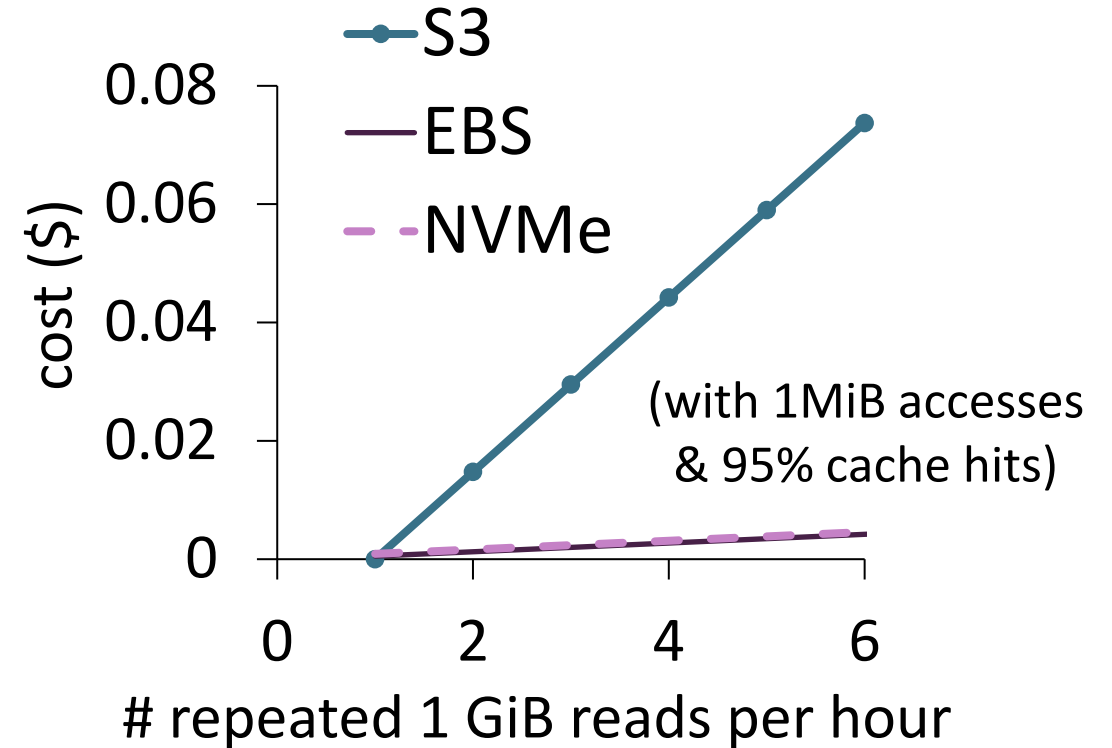
latency insensitive



**even with cheapest option →
7 requests per sec for an object**

latency sensitive

99% target → 100ms latency per 1MiB access



**2 requests per hour for an object
is enough to justify caching**

summary

need a separate framework to reason about caching in the cloud

- storage hierarchy relies on object stores
- cost of storage access isn't **0** after deployment
- **hence, the 5-min rule needs revisiting**

thank you!

preliminary evaluation shows that

- for ***latency-insensitive*** cases, you can live without a cache
 - considering the cost of maintaining caches
- for ***latency-sensitive*** cases, you need an object store cache

backup

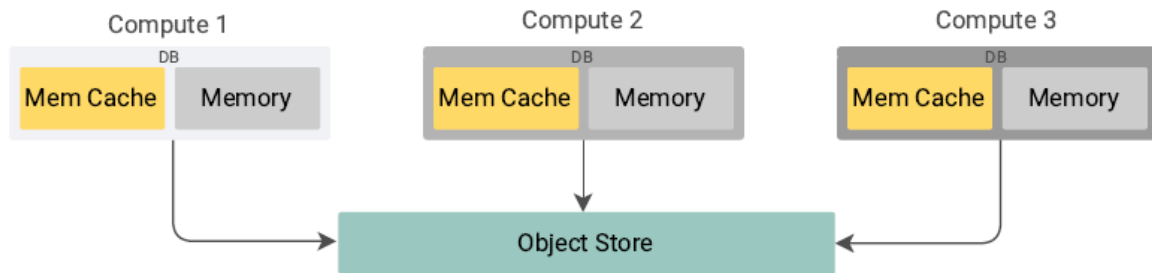
going forward ...

- evaluation of other vendors
 - ... given access latency measurements
- cases that need larger cloud instances
 - e.g., to get more vCPUs – necessary for throughput
 - you get the storage space on the side, which you can use for caching – does it make caching less costly, overall?
- separate considerations for meta-data and data
 - or do we assume that meta-data will likely be cached, regardless?
- modeling advanced caching
 - involving data transformations / push-down on the way
- transaction processing - impact of updates

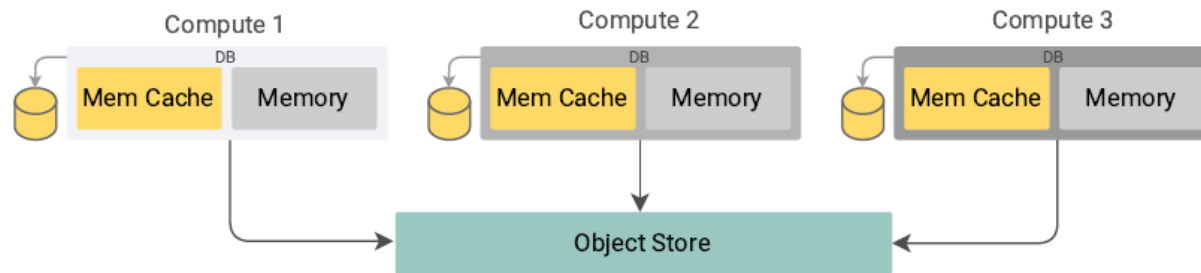
conventional 5-min rule

Metric	DRAM					HDD					NVMe SSD	
	1987	1997	2007	2018	2024	1987	1997	2007	2018	2024	2018	2024
Unit price (\$)	5k	15k	48	80	42	30k	2k	80	49	343	589	180
Unit capacity	1MB	1GB	1GB	16GB	32GB	180MB	9GB	250GB	2TB	20TB	800GB	2TB
\$/MB	5k	14.6	0.05	0.005	0.0014	83.33	0.22	0.0003	0.00002	0.00001	0.0007	0.00009
Random IOPS (r/w)	-	-	-	-	-	5	64	83	200	168 / 550	460 k	1,400k/1,550k
Seq bandwidth (MB/s) (r/w)	-	-	-	-	-	1	10	300	200	285	2500	7,450/6,900

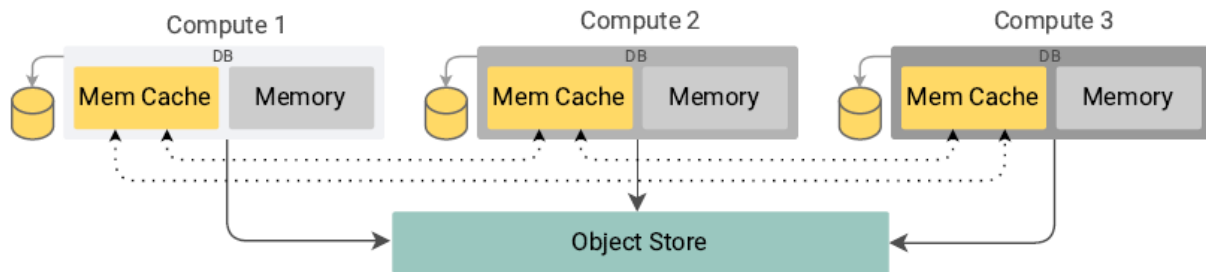
$$\frac{1000 \text{ kB} \times (\text{page size in kB})^{-1}}{\text{read IOPS}} \times \frac{\$ \text{ per Disk}}{\$ \text{ per MB of DRAM}}$$



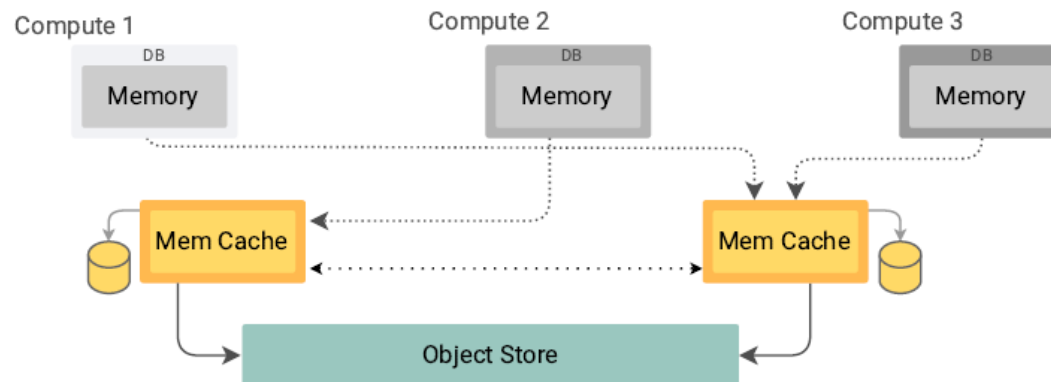
(a) Compute local (mem): Separate in-memory cache on each compute node.



(b) Compute local: Separate local storage and memory-based cache on each compute node.



(c) Shared nothing: Shared cache collaboratively managed and accessed by compute nodes.



(d) Cache Service: Cache is managed by a separate set of nodes.

Design	Latency Variability	Implementation Complexity	Operational Complexity	Object Store Request Count	Cache Capacity Elasticity
No cache	↑↑↑↑↑	None	None	↑↑↑↑↑	None
Compute local (memory only)	↑↑↑	↑	None	↑↑↑↑	↑
Compute local	↑↑↑	↑↑	↑	↑↑↑	↑
Shared nothing	↑↑	↑↑↑	↑↑	↑↑	↑↑↑
Cache Service	↑	↑↑↑	↑↑↑↑	↑	↑↑↑↑↑

calculating required repeated reads

knowing the latency distribution of requests to read an object of a certain size from the object store, we know

- $P \rightarrow$ probability of reading an object within the target latency

if there are n independent requests for this object

- $(1 - P)^n \rightarrow$ probably of all of them taking longer than target latency
- $P' = 1 - (1 - P)^n \rightarrow$ probability that at least one of the n reaches target

as the user, you can pick your desired P' (e.g., 99%)

then, given a P' , solving n yields

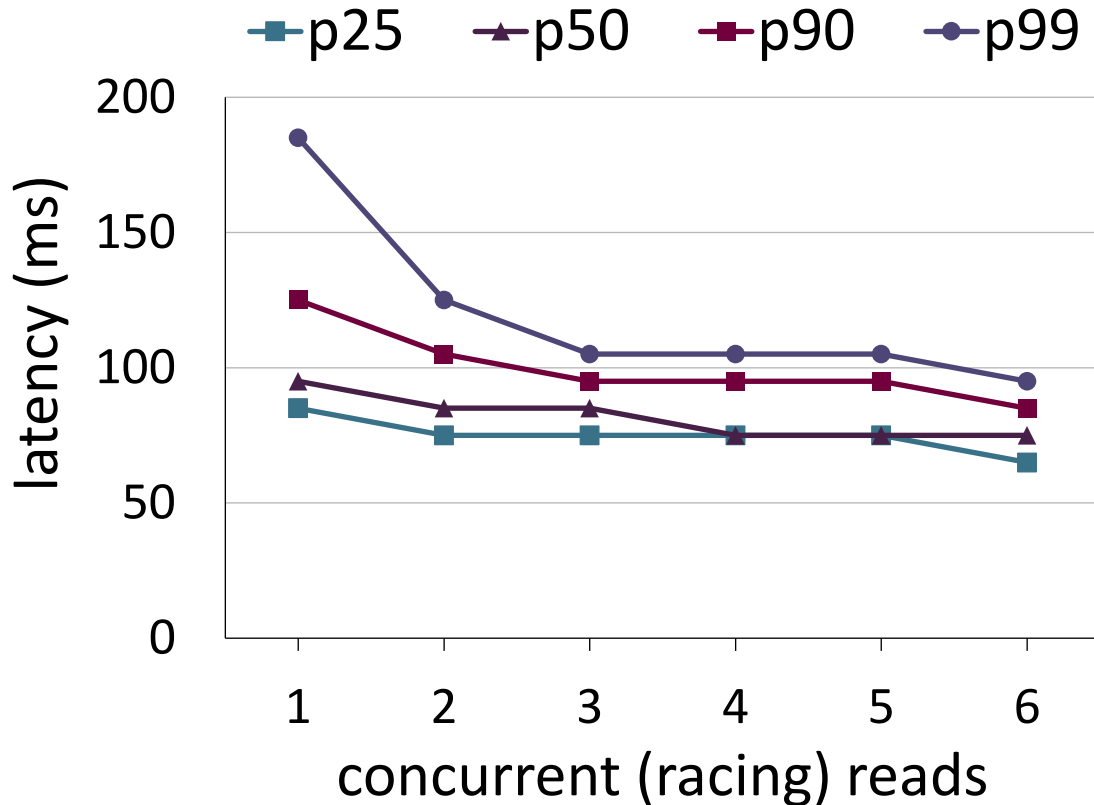
$$n = \text{RepeatsToGuaranteeLatency} = \log_{(1-P)}(1 - P')$$

racing reads

latency distribution for racing 1MiB reads to S3.

pN represents the Nth percentile:

N% of the requests completed within this time.

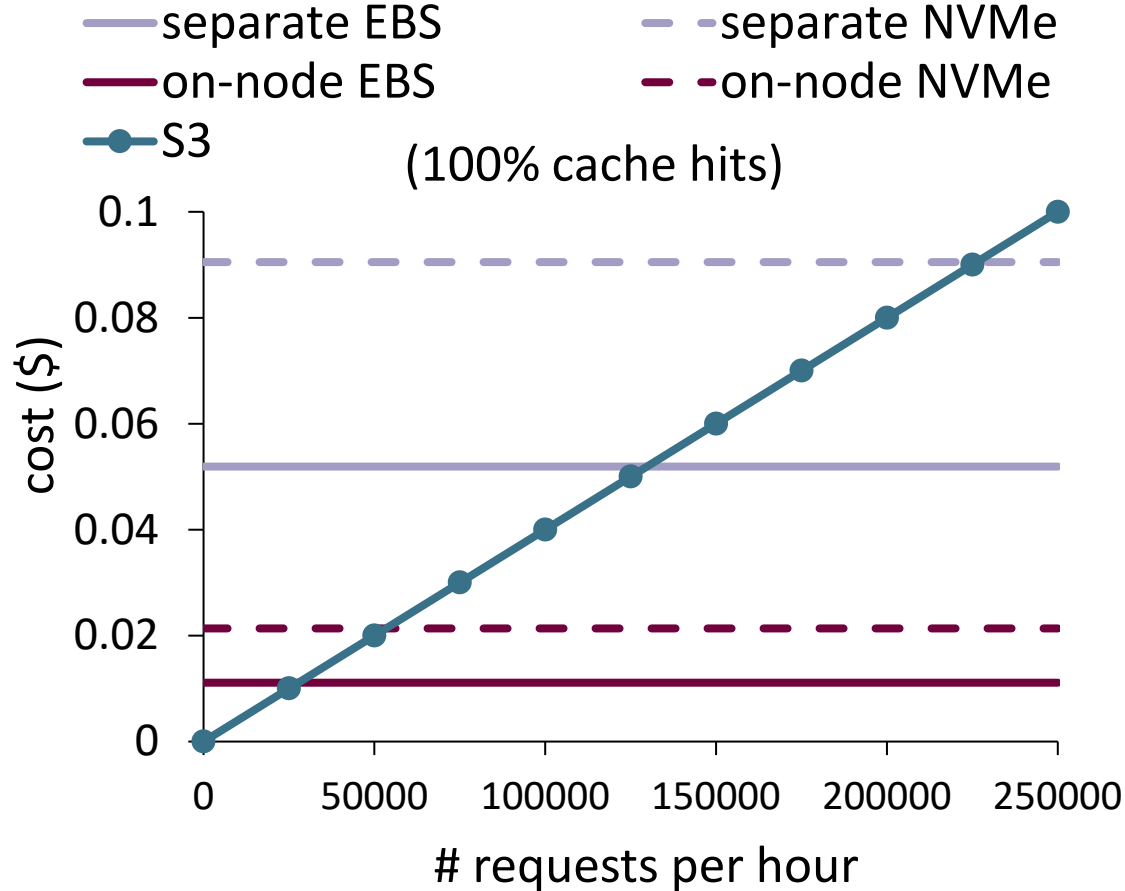


requests needed to download 10GiB from S3, where each request finishes in 150ms with 99% probability.

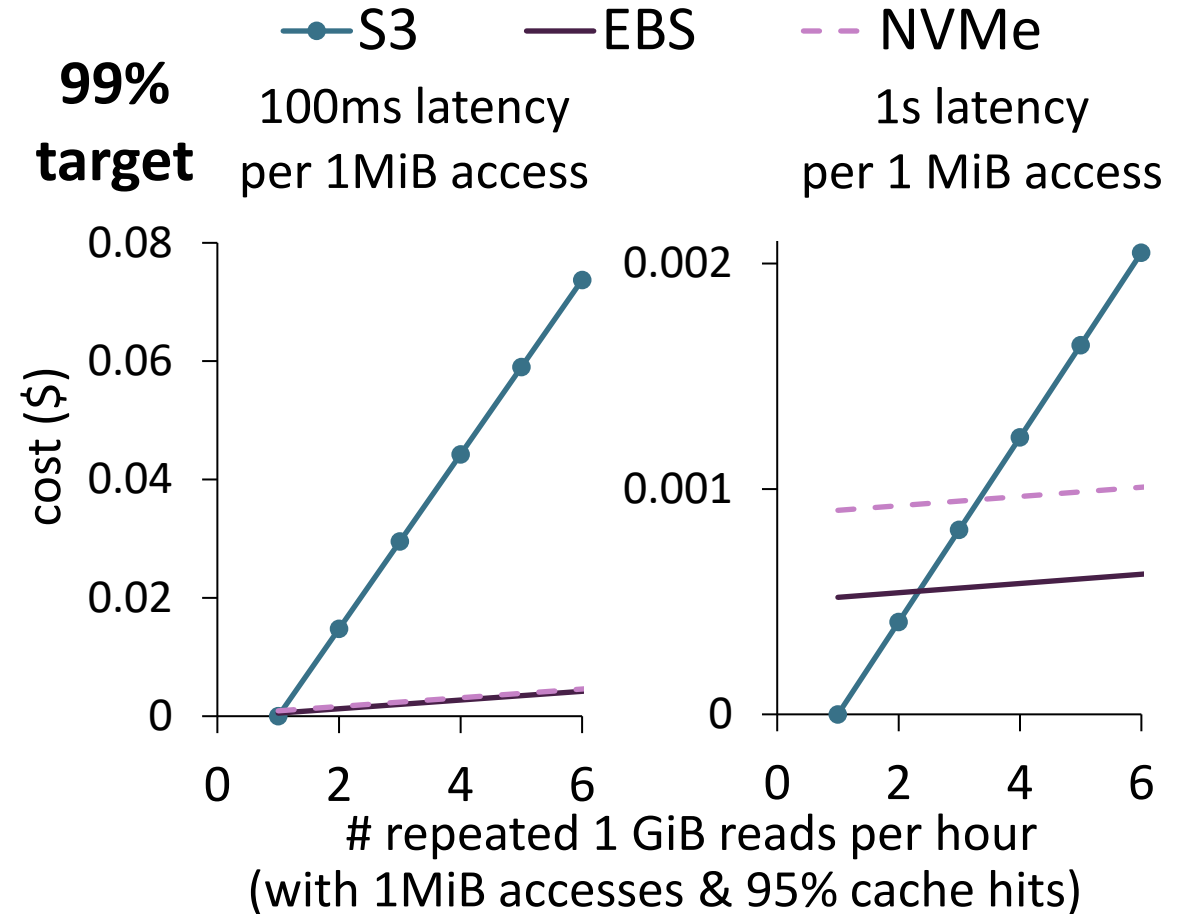
request size (MiB)	#requests
1	20480
4	94720
8	57600

results – when does it make sense to cache?

latency insensitive



latency sensitive



with cheapest option →

7 requests per sec for an object

with most sensitive case →

2 requests per hour for an object