Understanding I/O Performance Behaviors of Cloud Storage from a Client's Perspective

Binbing Hou, *Feng Chen*
Louisiana State University

Zhonghong Ou
Beijing Univ. of Posts & Telecomm.

Ren Wang, Michael Mesnier
Intel Labs

Storage in the Cloud Era

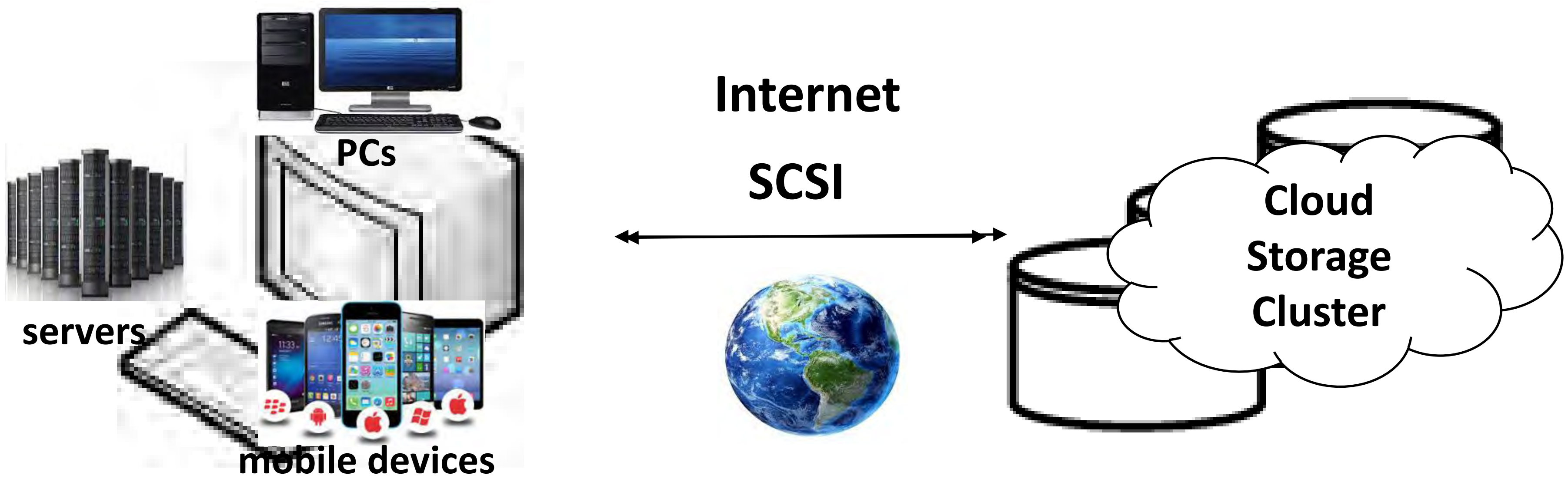
Enterprise Cloud Storage

Personal Cloud Storage

- Personal cloud storage subscriptions will reach **1.3 billion** by 2017[1]
- Public/private cloud storage market is predicted to be **$65.41 billion** by 2020[2]

[1] https://technology.ihs.com/410084/subscriptions-tocloud-storage-services-to-reach-half-billion-level-this-year.
[2] http://www.marketsandmarkets.com/Market-Reports/cloud-storage-market-902.html.

Cloud Storage vs. Conventional Storage



- **Clients**
  - Highly diverse
  - Different capabilities
- **Connection**
  - World-wide internet
  - HTTP-based protocol
- **Cloud storage cluster**
  - Massively parallelized
  - High throughput

*Is our **past wisdom** on storage still applicable to cloud storage?*

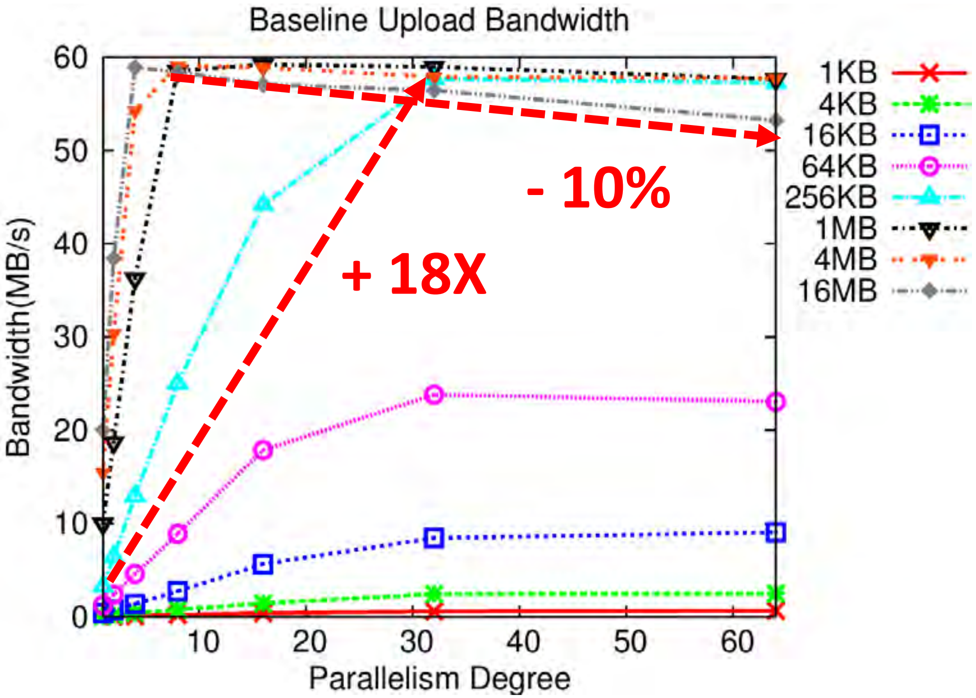
Measurement Methodology

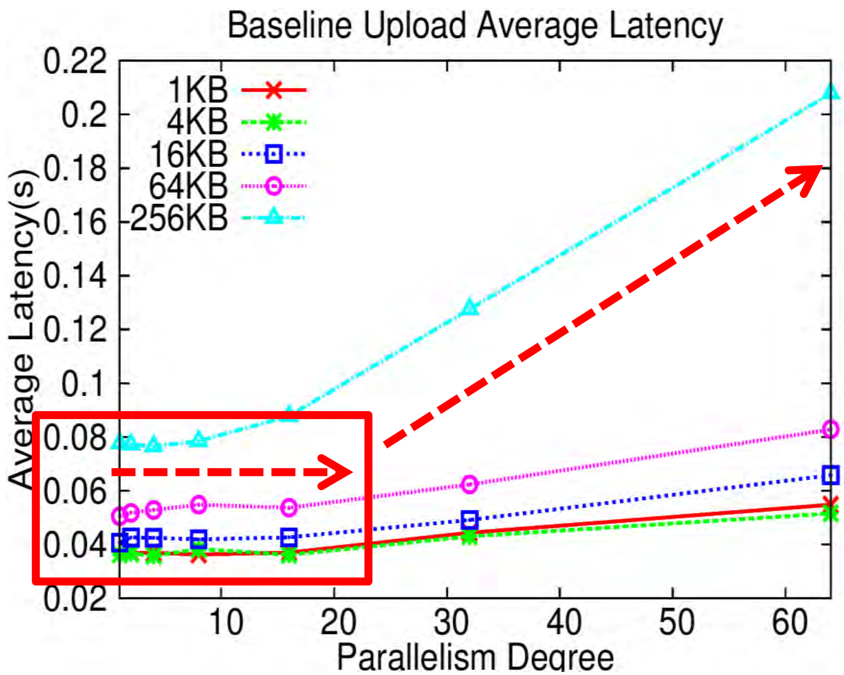
- **Investigating cloud storage as storage services**
  - “Black-box” testing
  - Adopting HTTP-based APIs rather than POSIX-like APIs
  - Purposely avoiding the client-side optimization techniques
- **Test Workloads**
  - Request Type: PUT (upload), GET (download)
  - Parallelism degree: 1 – 64
  - Request size: 1KB – 16MB
  - Metrics: **Bandwidth** and **Latency**
- **Platform to be tested:**
  - Cloud: select **Amazon S3** (the data center in Oregon)
  - Clients: customizing five **Amazon EC2** instances varying capabilities

Outline

- Basic Observations
- Effect of Client's Capability
- Effect of Geo-distance
- Case Study
- System Implications

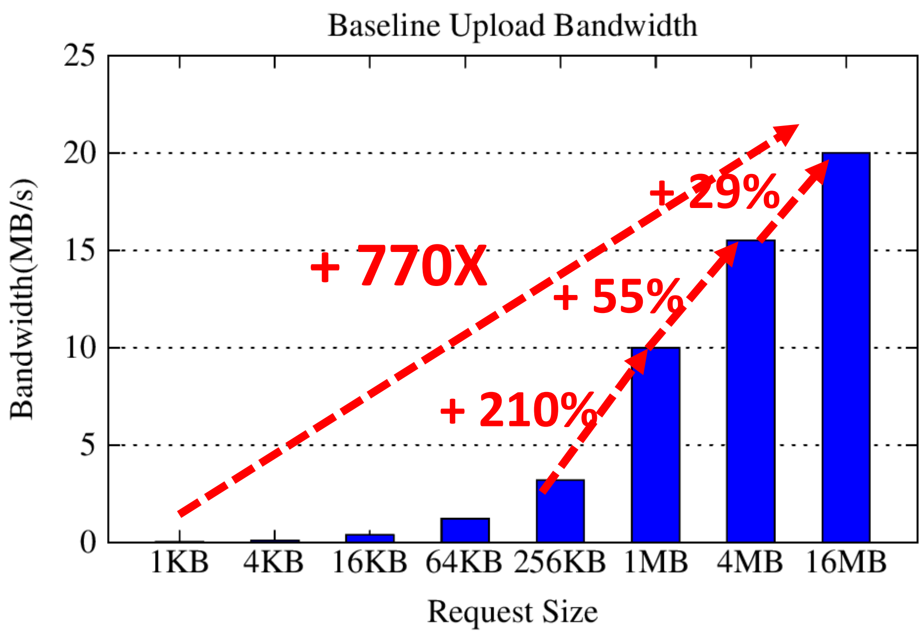
Effect of Parallelism

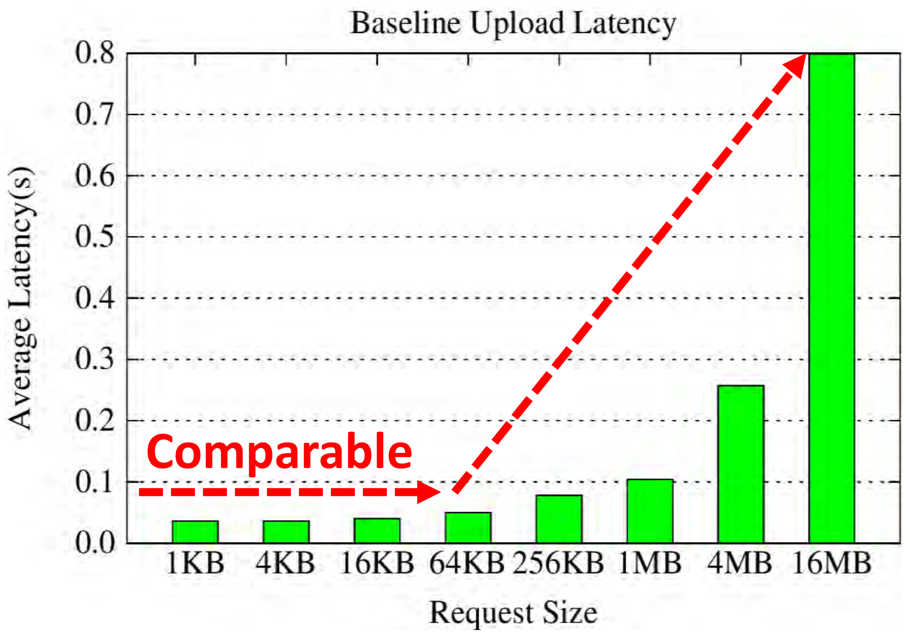




- **Effect of parallelism on bandwidth**
  - Proper parallelization dramatically improves the bandwidth (e.g., 18x speedup).
  - Over-parallelization may cause performance degradation (e.g., 10% degradation).
- **Effect of parallelism on request latency**
  - Proper parallelization does not significantly affect the latency.
  - Over-parallelization may lead to the latency increasing linearly.

Effect of Request Size

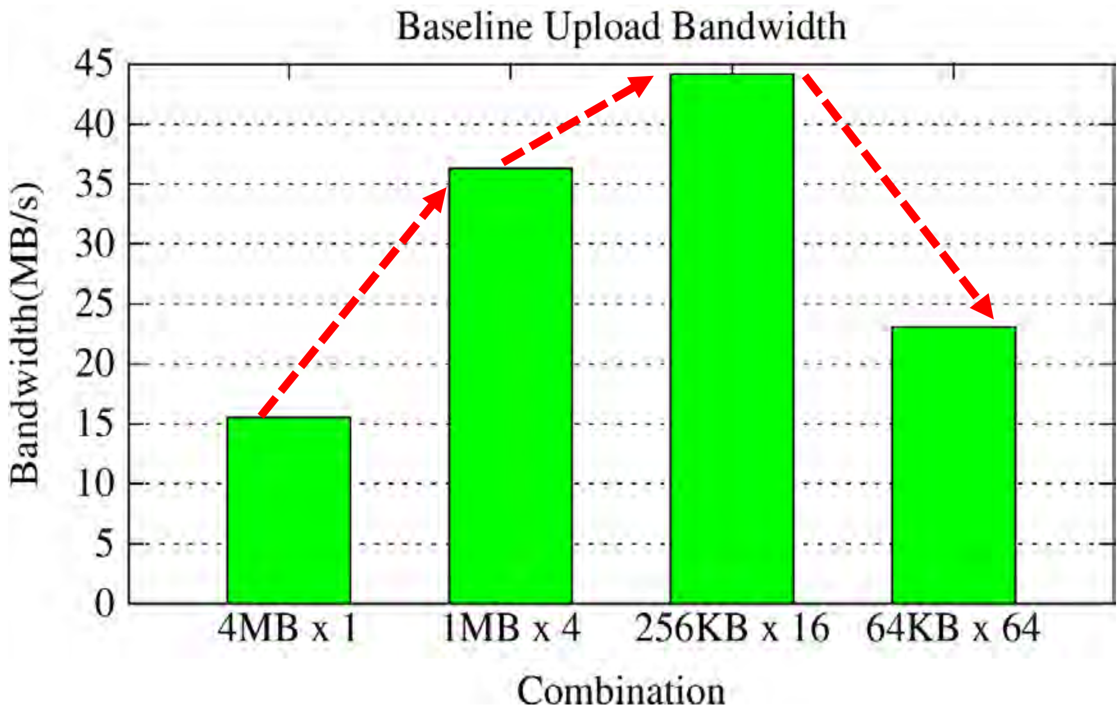




- **Effect of request size on bandwidth**
  - Increasing request size can significantly increase the bandwidth (e.g., 770x speedup).
  - The benefit brought by increasing request size is not unlimited (i.e., diminishing improvement).
- **Effect of request size on request latency**
  - Larger requests generally have higher request latencies.
  - The latency does not necessarily increase when request size increases (e.g., 1KB-16KB).

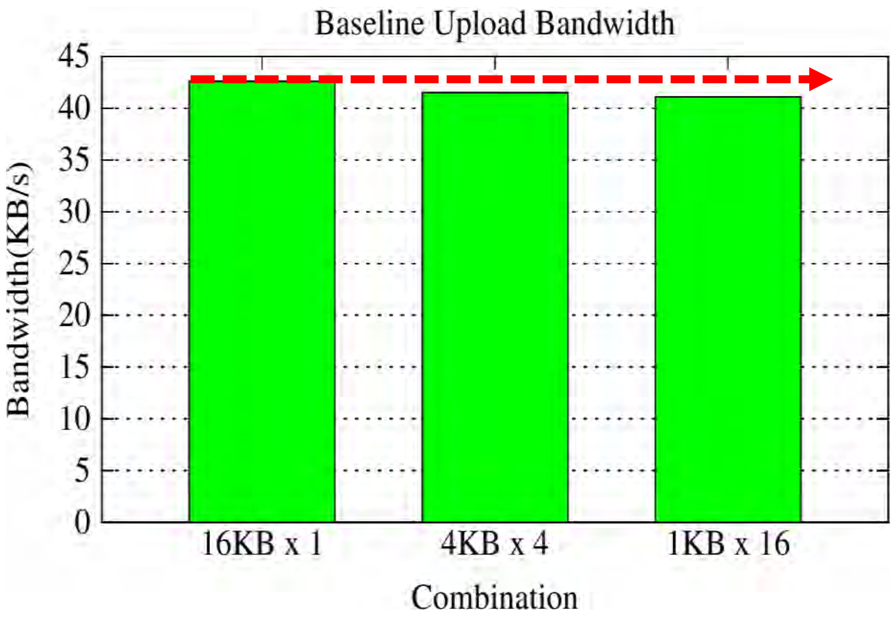
Parallelism vs. Request Size

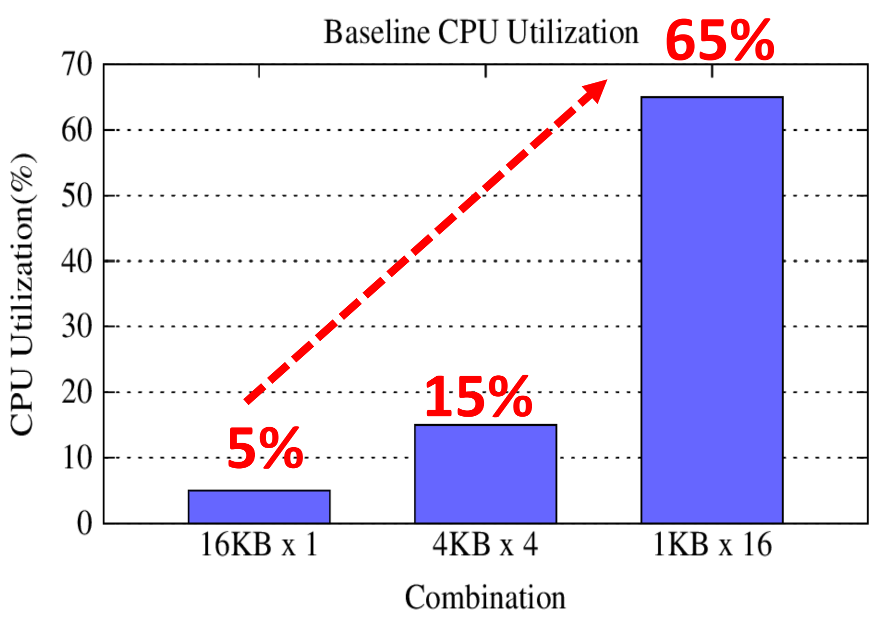
- **Both are helpful to improve bandwidth but have limitations.**
- **Does there exist any optimal combination?**
  - e.g., Upload a 4MB object
  - Reasonable combinations: 4MBx1, 1MBx4, 256KBx16, 64KBx64



Parallelism vs. Request Size (cont.)

- **What if comparable bandwidth with different combinations?**
  - e.g., Upload 16 objects of 1KB
  - Bandwidth: 16KB X 1 = 4KB X 4 = 1KB X 16





In such cases, larger requests consume less CPU resources.

Effect of Client's Capability

- **Experimental comparisons**
  - Comparing the performance of Baseline client with other four clients separately
  - Client CPU: Baseline (2 CPUs) vs. CPU-plus (4 CPUs)
  - Client memory: Baseline(7.5GB) vs. MEM-minus(3.5GB)
  - Client storage: Baseline (Magnetic) vs. STOR-ssd (SSD)

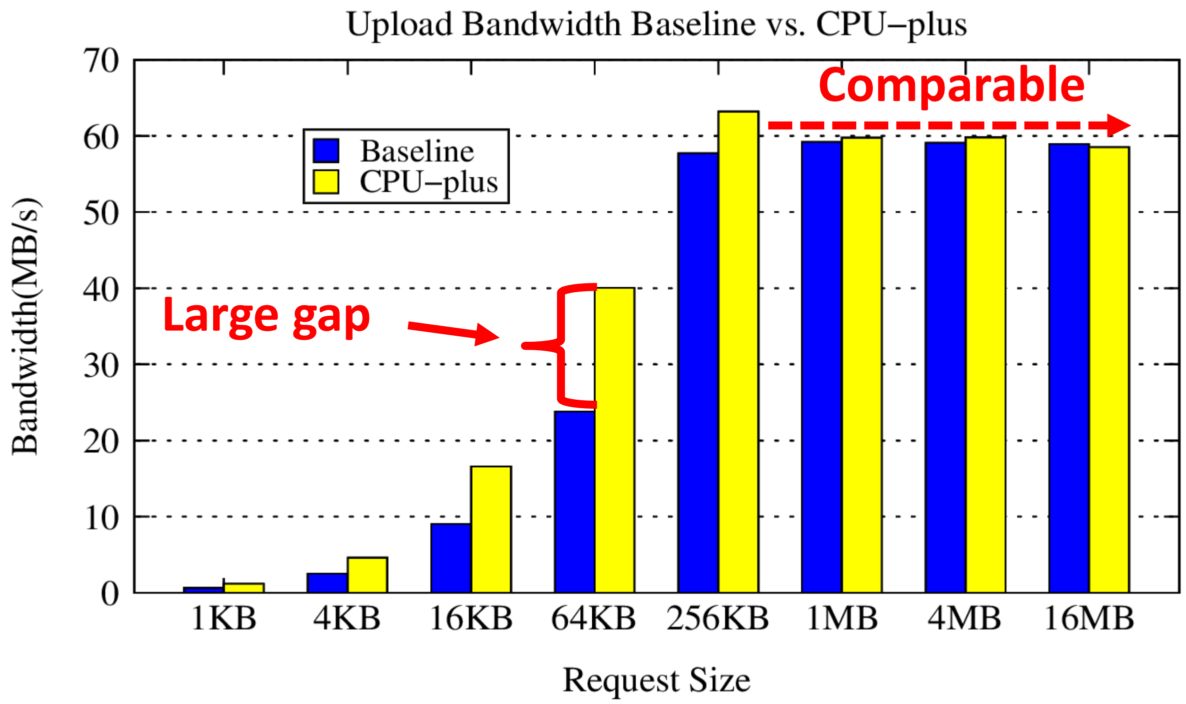
Client	Instance	Location	vCPU	Memory	Storage
Baseline	m1.large	Oregon	2	7.5 GB	Magnetic (410 GB)
CPU-plus	c3.xlarge	Oregon	**4**	7.5 GB	Magnetic (410 GB)
MEM-minus	m1.large	Oregon	2	**3.5 GB**	Magnetic (410 GB)
STOR-ssd	m1.large	Oregon	2	7.5 GB	**SSD (410 GB)**
GEO-Sydney	m1.large	**Sydney**	2	7.5 GB	Magnetic (410 GB)

Effect of Client CPU

- **Client CPU**

  - CPU is responsible for both data packets sending/receiving and client I/O.

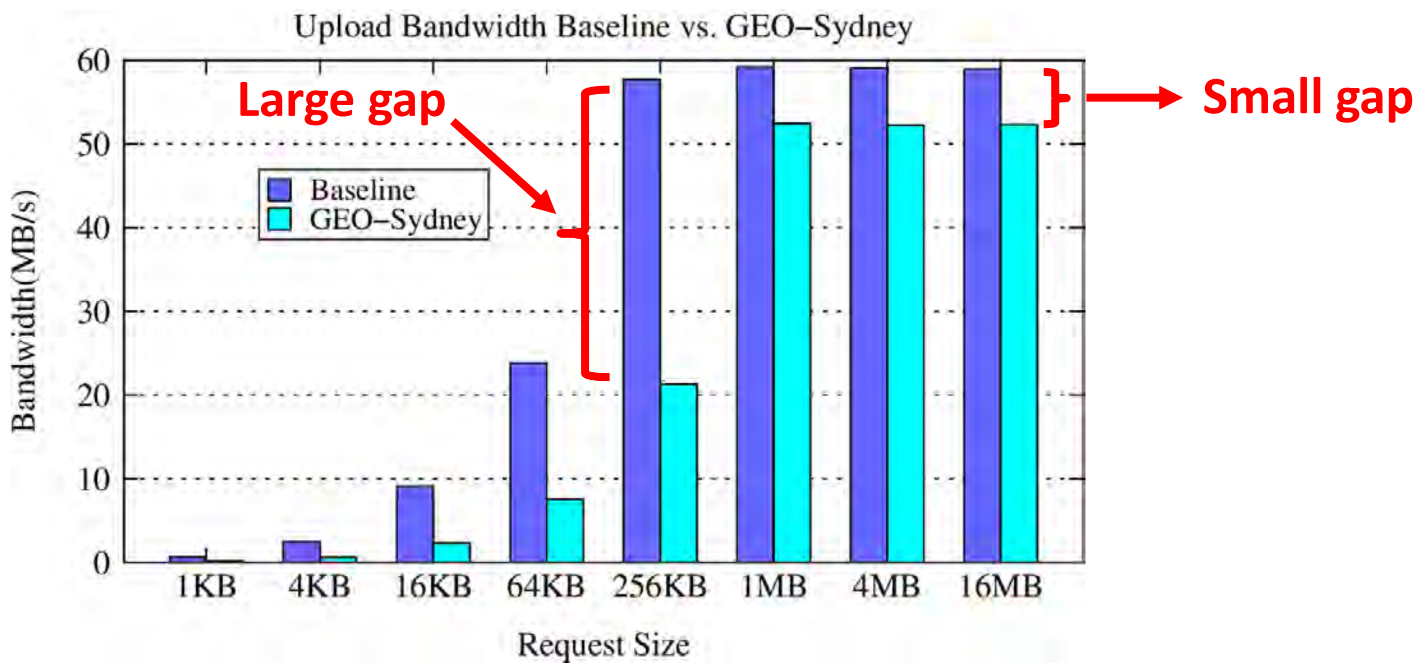
  - Comparison: Baseline (2 CPUs) vs. CPU-plus (4 CPUs)



- Client CPU has significant effects on small requests.
- Client CPU **does not** have significant effects on large requests.

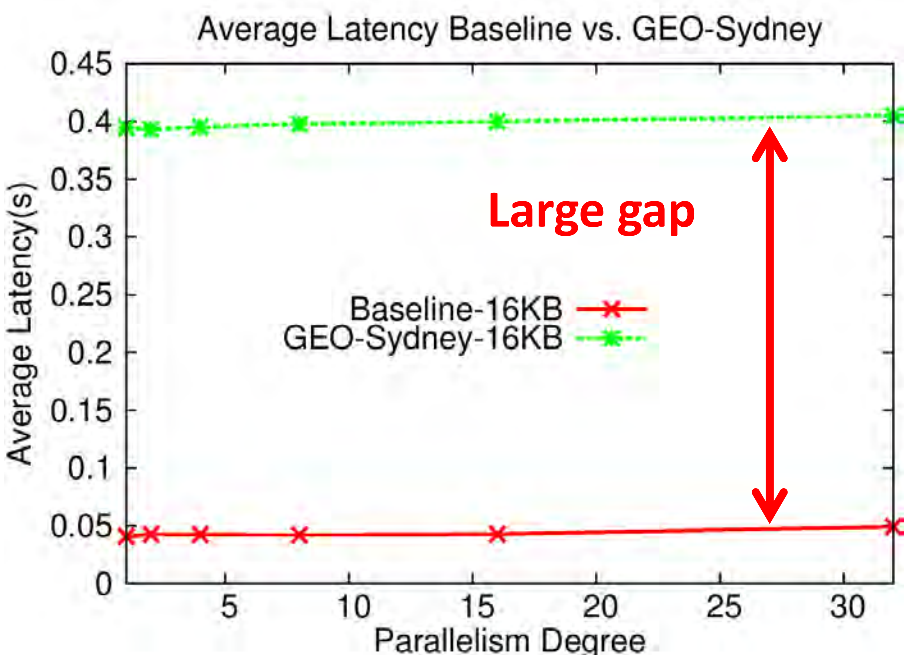
Effect of Geo-distance

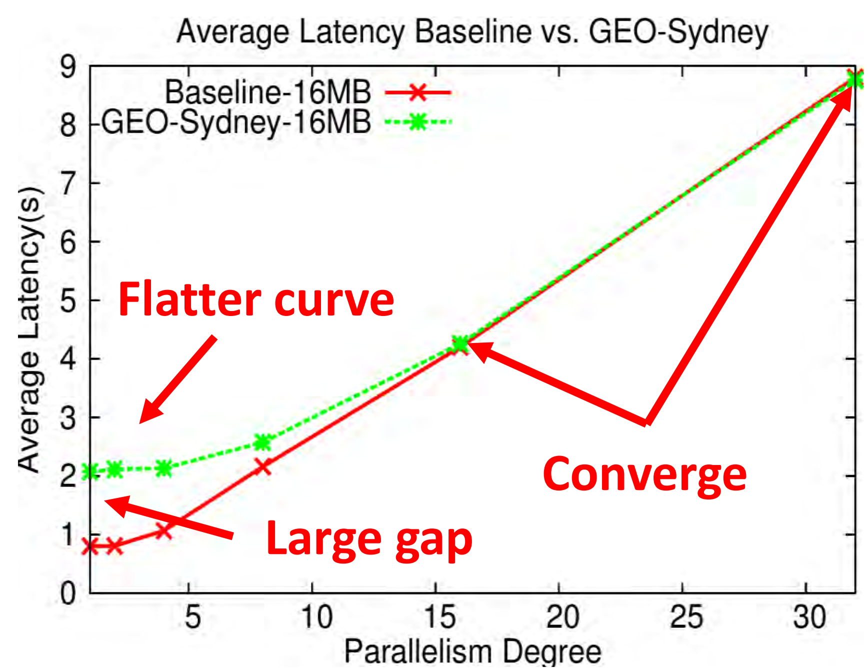
- **Geo-distance**
  - Comparison: Baseline (in Oregon) vs. GEO-Sydney (in Sydney)
  - RTT: 0.28ms (same data center in Oregon) vs. 176ms (from Sydney to Oregon)
- **Effect of geo-distance on bandwidth**
  - Geo-distance has significant effect on peak bandwidth of small requests
  - Geo-distance **does not** significantly affect peak bandwidth of large requests



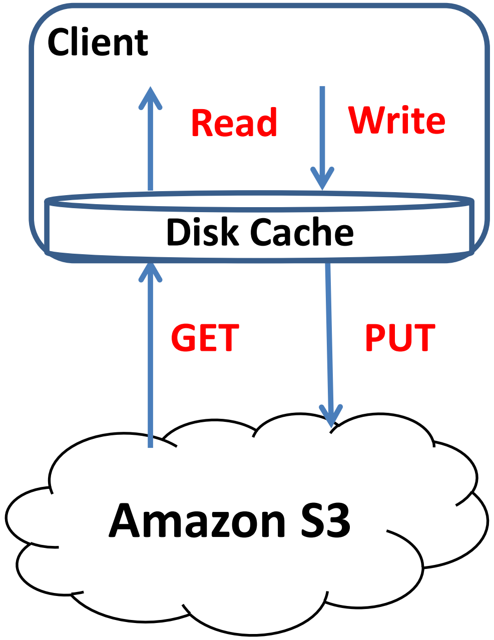
Effect of Geo-distance (cont.)

- **Geo-distance**
  - Comparison: Baseline (in Oregon) vs. GEO-Sydney (in Sydney)
  - RTT: 0.28ms (same data center in Oregon) vs. 176ms (from Sydney to Oregon)
- **Effect of geo-distance on latency**
  - RTT plays a critical role but is **not the only** determining factor





Case Study: Client-side Caching

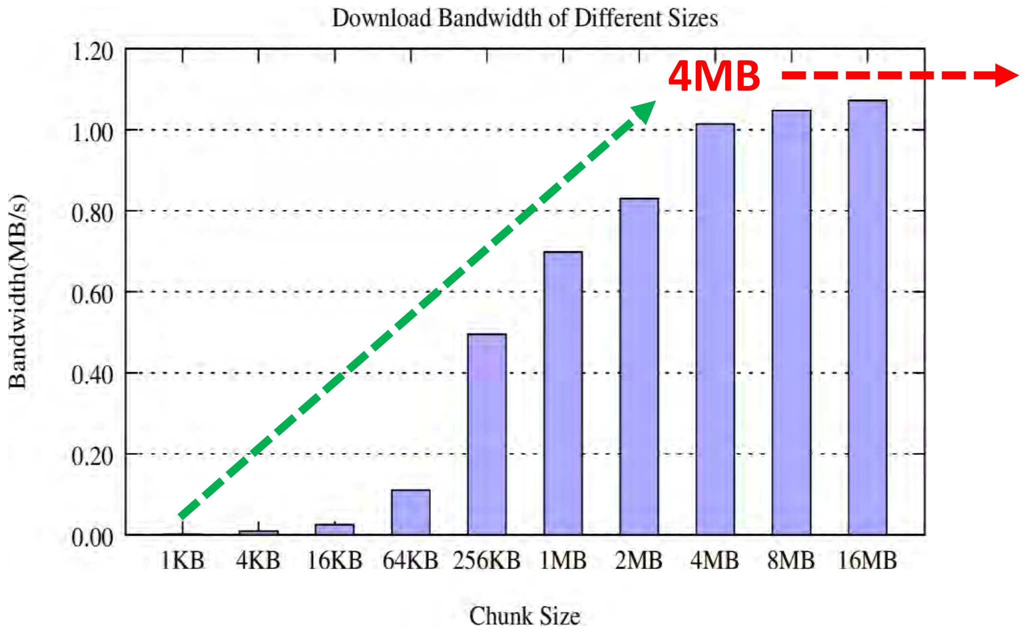


- **Client-side caching and chunking**
  - Chunking is a key technique used in cloud storage
  - Chunking will affect the caching efficiency
    - Small chunk size may lead to high cache miss ratio
    - Large chunk size may be risky of loading unwanted data
- **Experimental platform**
  - Cloud storage services: Amazon S3 in Oregon
  - Client: a workstation in Louisiana
  - Emulator: converting POSIX operations to HTTP requests; disk cache support
- **Trace**
  - Converted from a segment of NFS trace of Harvard SOS project[3]
  - Workload size: 4.8 GB
  - Average file size: 12.9 MB

[3] https://www.eecs.harvard.edu/sos/traces.html

Case Study: Proper Chunk Size for Caching

- **How can we determine a proper chunk size?**
  - Bandwidth-based method can give some hints
  - Select a relatively small size that can approximately reach the peak bandwidth

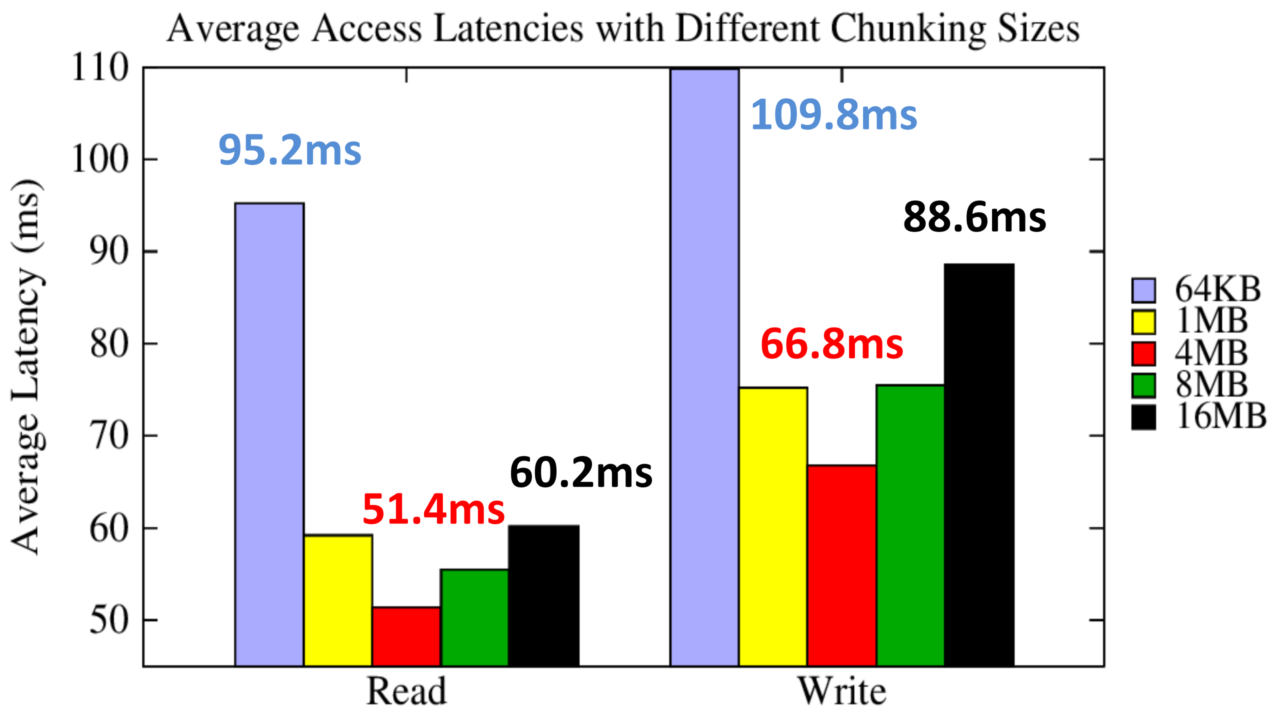


When chunk size exceeds 4MB:
- **cannot bring significant benefit**
- **high risk of loading unwanted data**

Proper chunk size ≈ 4MB ?

Case Study: Proper Chunk Size for Caching(cont.)

- **Experimental comparison**
  - Standard LRU, 200 MB disk cache, write back every 30s
  - Comparison: 64KB, 1MB, **4MB**, 8MB, 16MB

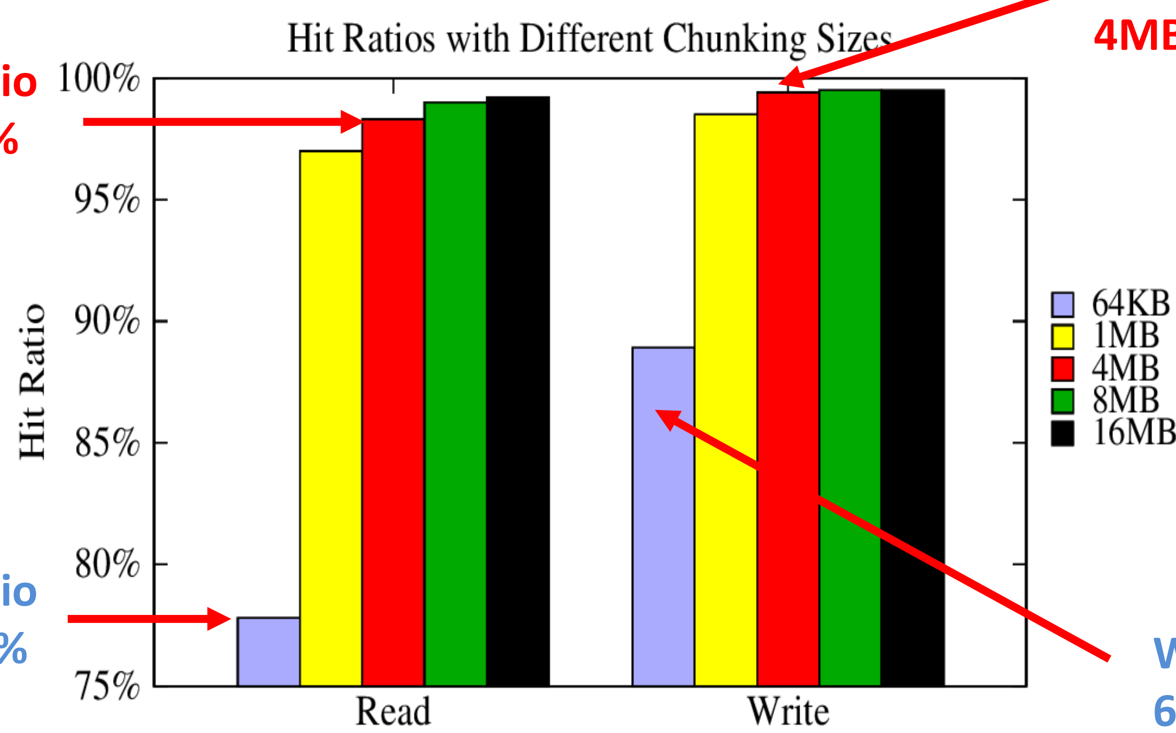


- 4MB leads to the lowest read/write latencies.

Case Study: Proper Chunk Size for Caching(cont.)

Write hit ratio
4MB: 99.4%

Read hit ratio
4MB: 98.4%



Read hit ratio
64KB: 77.8%

Write hit ratio
64KB: 88.9%

- **How does chunking policy affect caching efficiency?**
  - Increasing request size significantly improves the performance (i.e., hit ratio).
  - Excessively large request size causes performance degradation.
    - high cache miss penalty=high download latency (4s to 4MB, 14.2s to 16MB).

System Implications

- **Properly combining parallelism and request size**
  - Reshaping the workloads: chunking/bundling, parallelizing
  - Optimal bandwidth can be achieved by proper combination
- **Client-aware optimization**
  - Special attentions should be paid to exploiting client's capability
  - Small requests (CPU) vs. large requests(Memory, Storage)
  - Client aware optimization: e.g., smartphone (weak CPU)
- **Geographical distance plays an important role**
  - The negative effects can be offset by proper optimization
  - Latency-sensitive applications: e.g., file system, database
  - Bandwidth-sensitive applications: e.g., backup, video services

Conclusion

- **We present a comprehensive measurement of cloud storage from the perspective of the client side.**
- **Our case studies demonstrate that user experiences can be better optimized by understanding cloud I/O behaviors.**
- **Based on our findings, we present a series of system implications.**

Thank you!

{bhou, fchen}@csc.lsu.edu
zhonghong.ou@bupt.edu.cn
{ren.wang, michael.mesnier}@intel.com