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

Binbing Hou, Feng Chen
Division of Computer Science and Engineering, LSU

Zhonghong Ou
Beijing Univ. of Posts & Telecomm.

Ren Wang, Michael Mesnier
Intel Labs

Storage in the Cloud Era

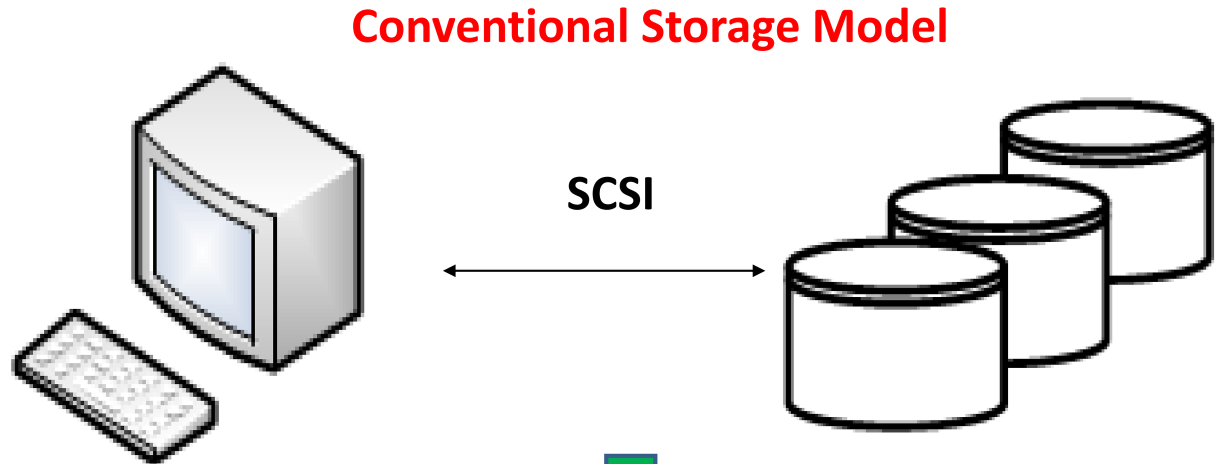
Enterprise Cloud Storage **Personal Cloud Storage**

- Cloud storage market is predicted to be $74.96 billion by 2021

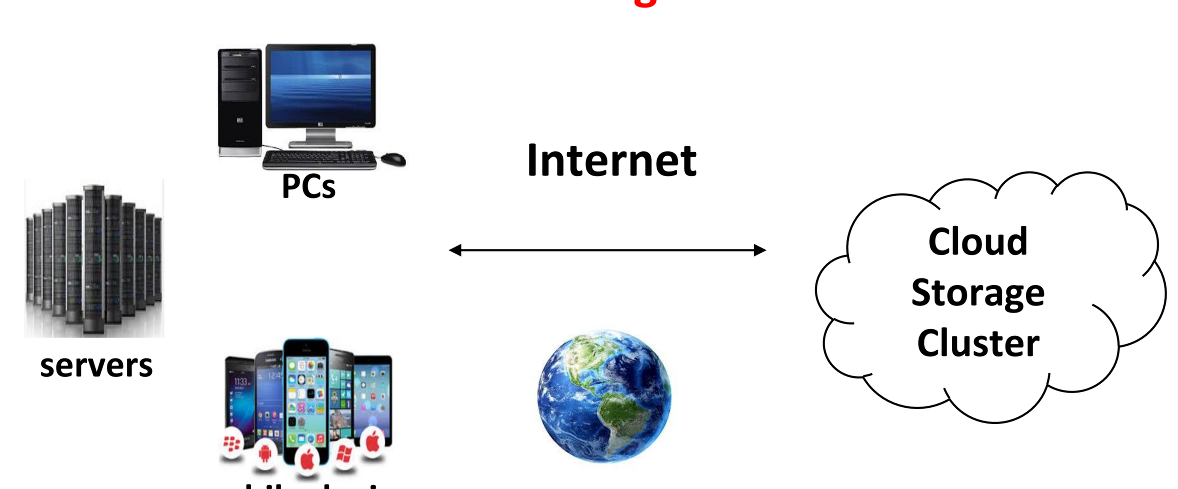
http://www.marketsandmarkets.com/Market-Reports/cloud-storage-market-902.html.

Cloud Storage vs. Conventional Storage

Conventional Storage Model



Cloud Storage Model

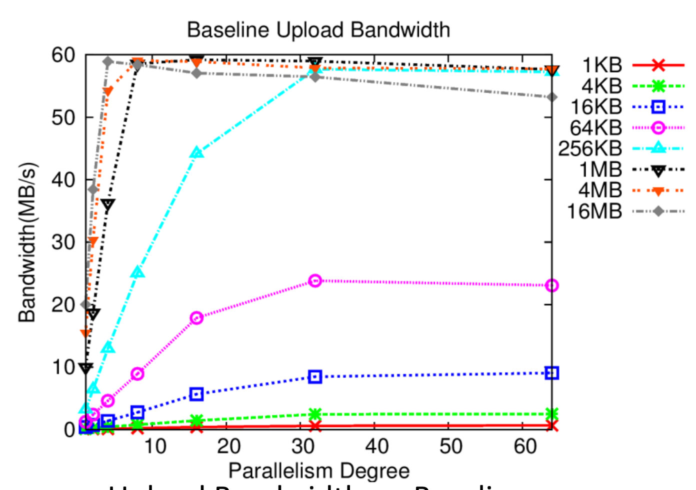


- **Characteristics of cloud storage model**
  - The clients are highly diverse and have different capabilities e.g., PCs, servers, and mobile devices
  - The connection is world-wide Internet with HTTP-based protocol
  - The storage medium is massively parallelized storage cluster
- **Is our past wisdom on storage still applicable?**
  - What are the effects of parallelism and request size?
  - What are the effects of client's capabilities?
  - What are the effects of geographic distance?
  - Can we obtain useful system implications based on the findings?

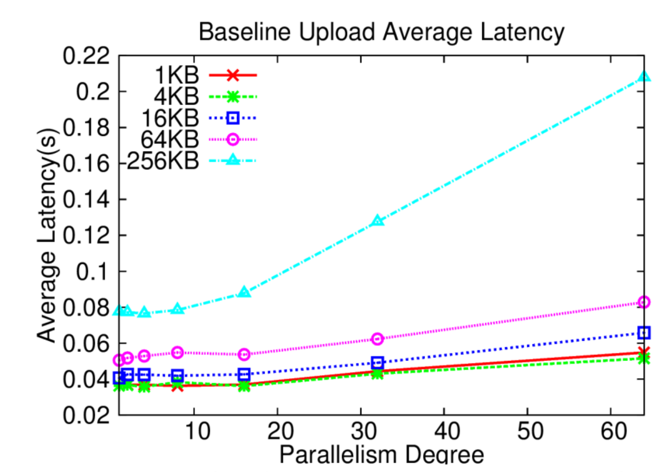
Methodology

- **Investigating cloud storage as storage services**
  - Focusing on object-based cloud storage
  - Treating cloud storage as a "black-box"
  - 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
- **Test platform**
  - Cloud: Amazon S3 (the data center in Oregon)
  - Clients: customizing five Amazon EC2 instances as clients
  - *Baseline* client: 2 CPUs, 7.5 GB memory, 410 HDD, in Oregon
  - Four comparison clients, each has a different factor as *Baseline*
- **Measurement steps**
  - Characterizing effects of parallelism and request size
  - Investigating client-related factors by controlled comparisons

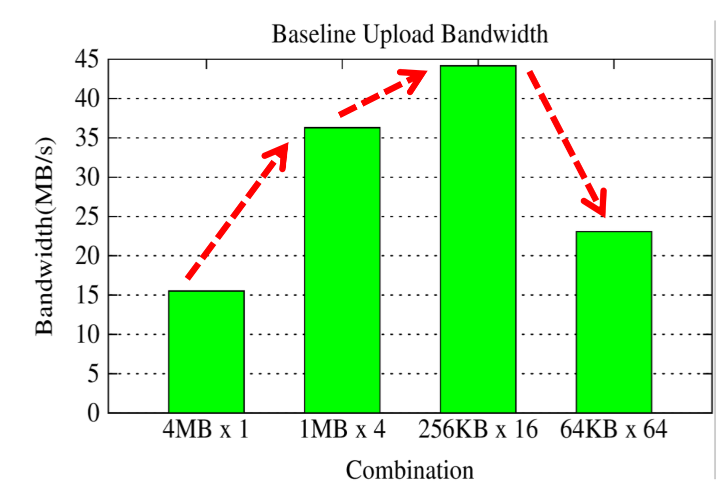
Critical Findings and Implications



Upload Bandwidth on Baseline

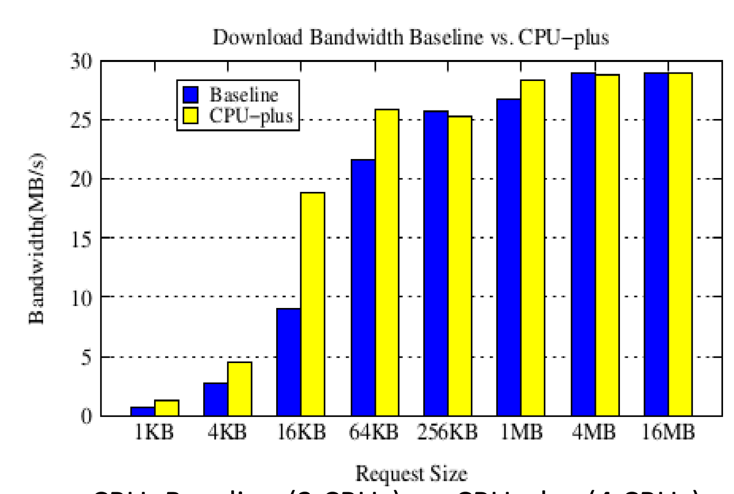


Average Upload Average Latency on Baseline

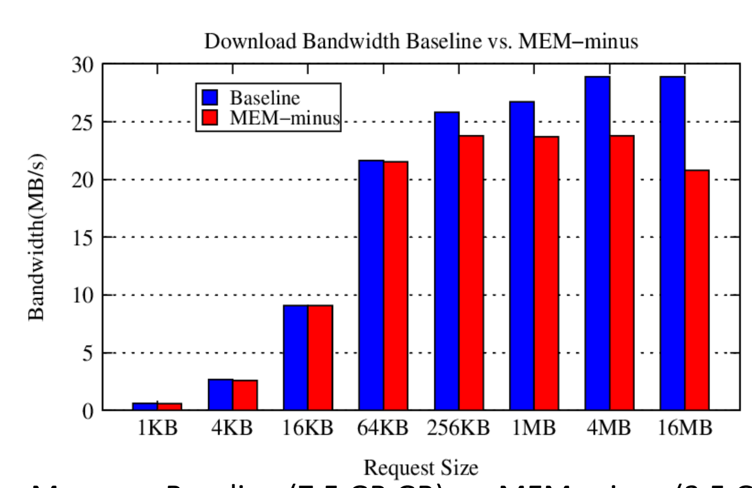


Different Combinations for Uploading a 4MB Object

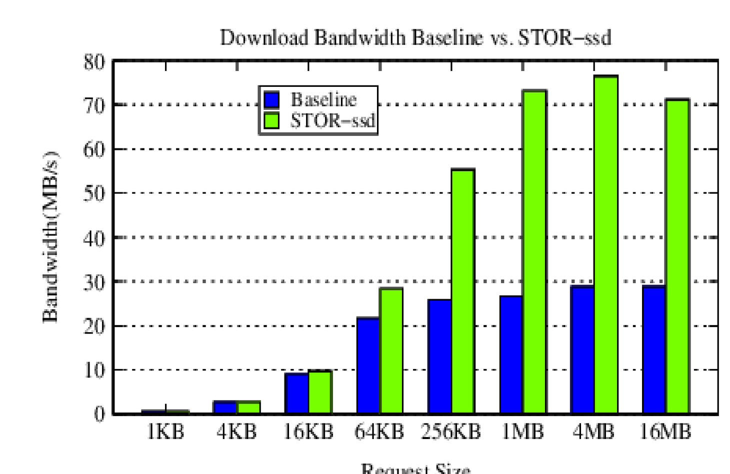
Finding #1: Either increasing *parallelism degree* or enlarging *request size* can improve performance, and optimal performance can be achieved by properly combining these two factors (e.g., uploading a 4MB object with 16 parallel 256KB requests).
Implications: Reshaping the workloads is effective to improving performance (e.g., chunking large requests to create parallelization opportunities, merging small requests to increase request size, and properly combining these two techniques).



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

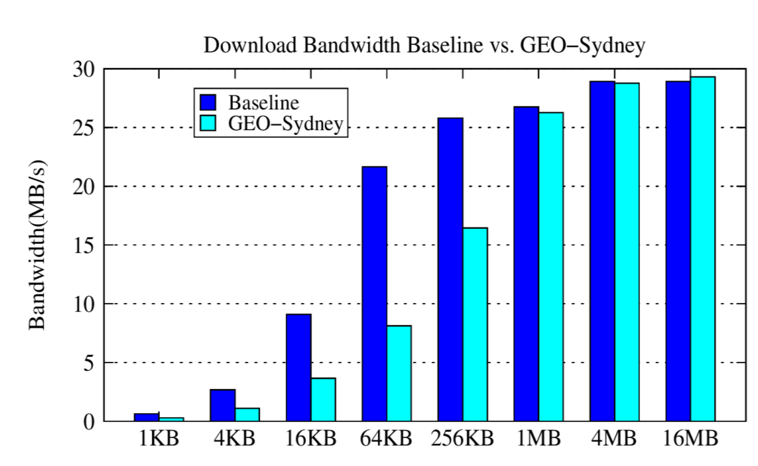


Memory: Baseline (7.5 GB GB) vs. MEM-minus (3.5 GB)

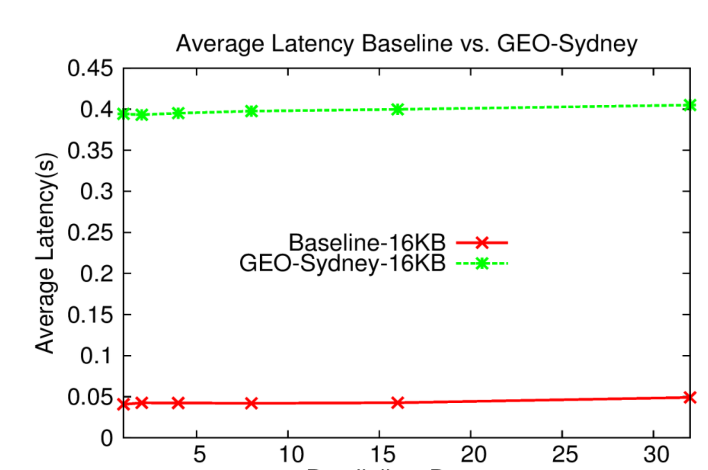


Storage: Baseline (HDD) vs. MEM-minus (SSD)

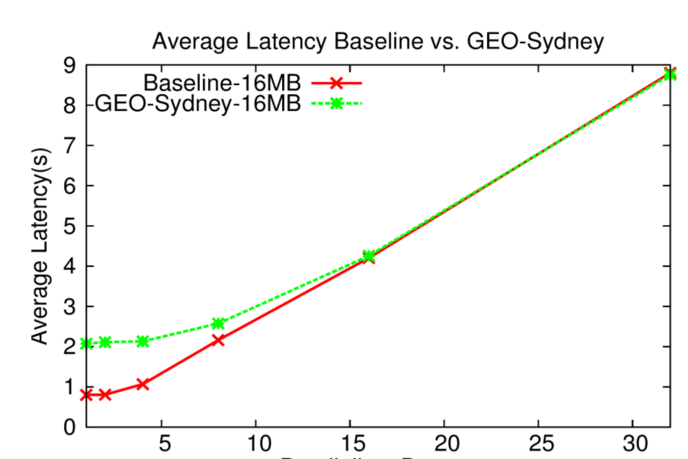
Finding #2: Client's capabilities play an important role in determining user-perceived performance (i.e., CPU is key to parallelizing small requests, and memory and storage are critical to serving large requests).
Implications: Client-aware optimization is necessary (e.g., smartphones generally have weaker CPUs than workstations, thus merging small requests can bring more benefits in this case).



Distance: Baseline (Oregon) vs. GEO-Sydney(Sydney)



Download latency comparison: 16KB

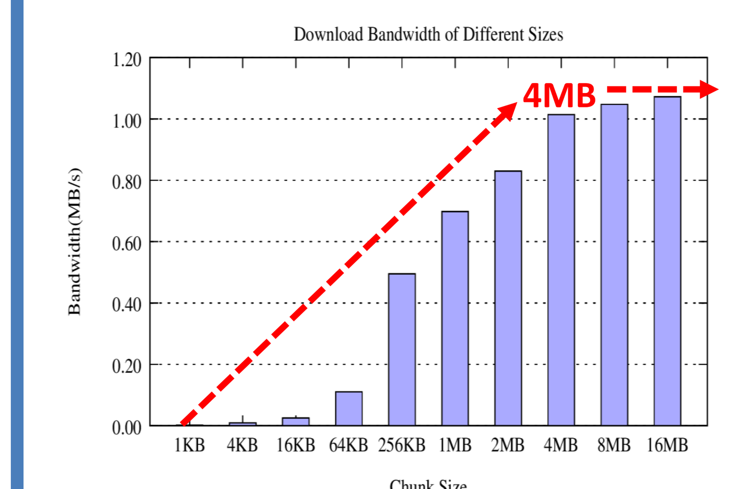


Download latency comparison: 16MB

Finding #3: A long *geographic distance* has negative effects but does not necessarily lead to worse performance (e.g., the peak bandwidth is not significantly affected, the impact of over-parallelization may offset the advantage of a short distance on latency).
Implications: We do not have to always choose closer but maybe more expensive cloud storage services (e.g., for bandwidth-sensitive applications, properly parallelizing large requests can achieve desirable performance even being far away from the cloud).

Case Study: Proper Chunking for Caching

- **Client-side caching and chunking**
  - Chunking is a key technology for cloud-based applications
  - Small chunk size may lead to high cache miss ratio
  - Large chunk size may be risky of loading unwanted data
- **How can we determine a proper chunk size?**
  - Approximation: sampling and inferring based method
  - Select a small size that closely reaches peak bandwidth
- **Experimental platform**
  - Emulator: a cloud-based file system with disk cache
  - Cloud: Amazon S3 in Oregon
  - Client: a workstation on campus
  - Trace: concerted from a piece of NFS trace
  - Caching replacement policy: standard LRU
- **Sampling and inferring based method**

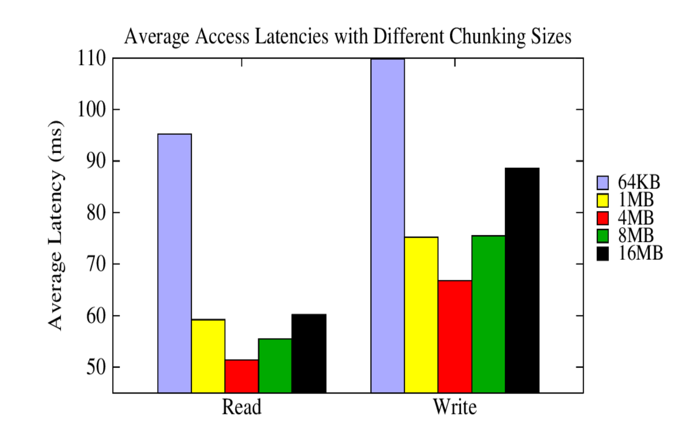


Sampling
When chunk size exceeds 4MB:
- cannot bring significant benefit
- may load unwanted data

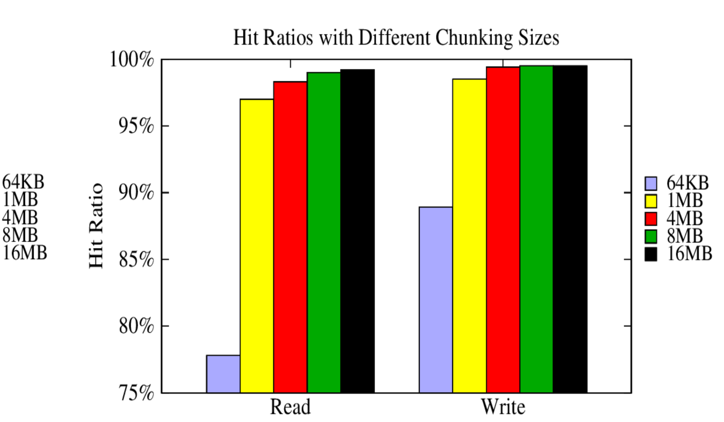
Inferring: Proper chunk size ≈ 4MB

Bandwidth of different chunk sizes on test client

- **Evaluation and Analysis**



Read/write latencies with different chunk sizes



Hit ratios with different chunk sizes

- **4MB chunking leads to lowest read/write latencies**
- **How do chunk sizes affect caching efficiency?**
  - Increasing request sizes significantly improves hit ratios
  - Excessively large request sizes cause performance loss
  - Long download latency = high cache miss penalty

Conclusions

- Unlike prior work that focuses on cloud storage providers and specific cloud storage clients, we present a comprehensive measurement of cloud storage from a client's perspective.
- Through controlled comparisons and quantitative analysis, we obtain several interesting and useful findings and system implications for optimizing user experiences.
- We further present a case study to illustrate how to improve user experiences by understanding cloud I/O performance behaviors of cloud storage in practical working scenarios.

*This work is published in Proceedings of the 32nd International Conference on Massive Storage Systems and Technology (MSST'16), Santa Clara, CA, May 2-6, 2016