Pacific Northwest
NATIONAL LABORATORY

Gnosis Research Center

ILLINOIS INSTITUTE OF TECHNOLOGY

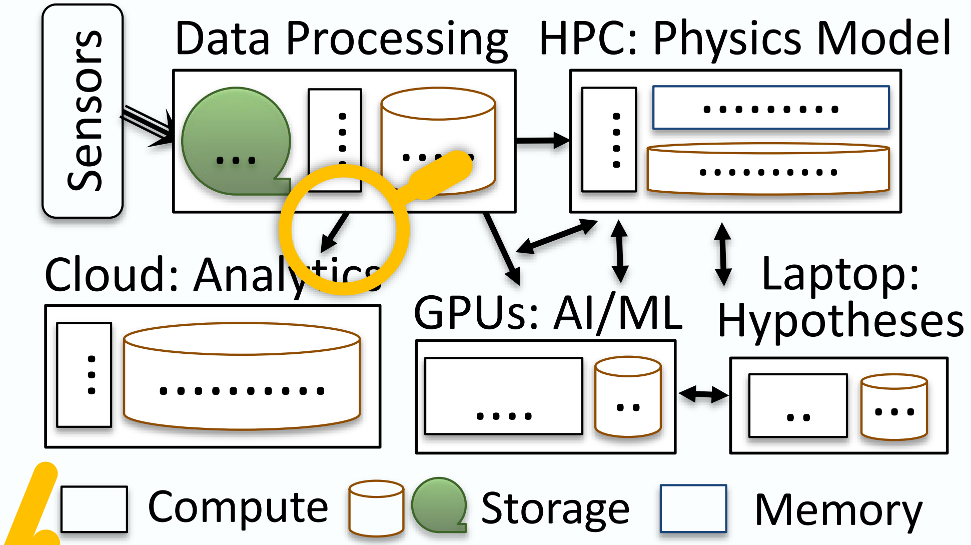
Quantifying AWS S3 I/O Performance Boundaries Using the Roofline Model

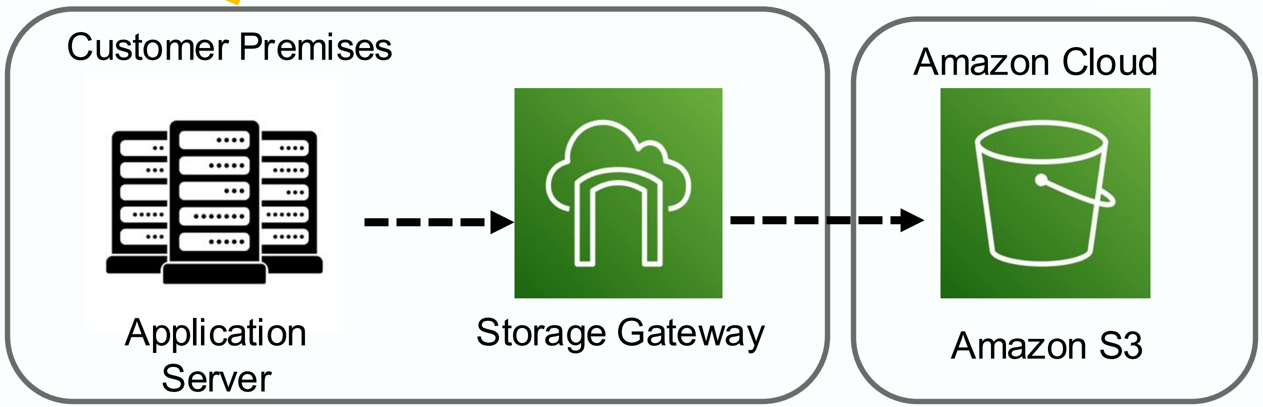
JG|U
JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Meng Tang, Zhaobin Zhu, Luanzheng Guo, James G. Bandy, Timothy Carlson,
Sarah Neuwirth, Anthony Kougkas, Xian-He Sun, Nathan R. Tallent
Illinois Institute of Technology, Johannes Gutenberg University Mainz,
Pacific Northwest National Laboratory

HPC-Cloud Integration and Scientific Workflows

- HPC facilities are increasingly integrating cloud services to leverage:
  - Virtually unlimited storage capacity
  - Improved data sharing across hybrid environments
- Storage Gateways emerge to bridge the gap:
  - Maps POSIX/NFS file storage to S3 object
- The fundamental challenge is I/O performance:
  - Quantifying trade-offs: Gateway access vs. Direct S3 API
- Our solution is the Extended I/O Roofline Model for cloud storage analysis.





Understanding and Quantifying AWS S3 Performance

- We extend the I/O roofline model to characterize cloud storage performance to compared:

  (1) POSIX I/O on NFS-mounted Storage Gateway

  (2) Data Migration NFS-mounted Storage Gateway

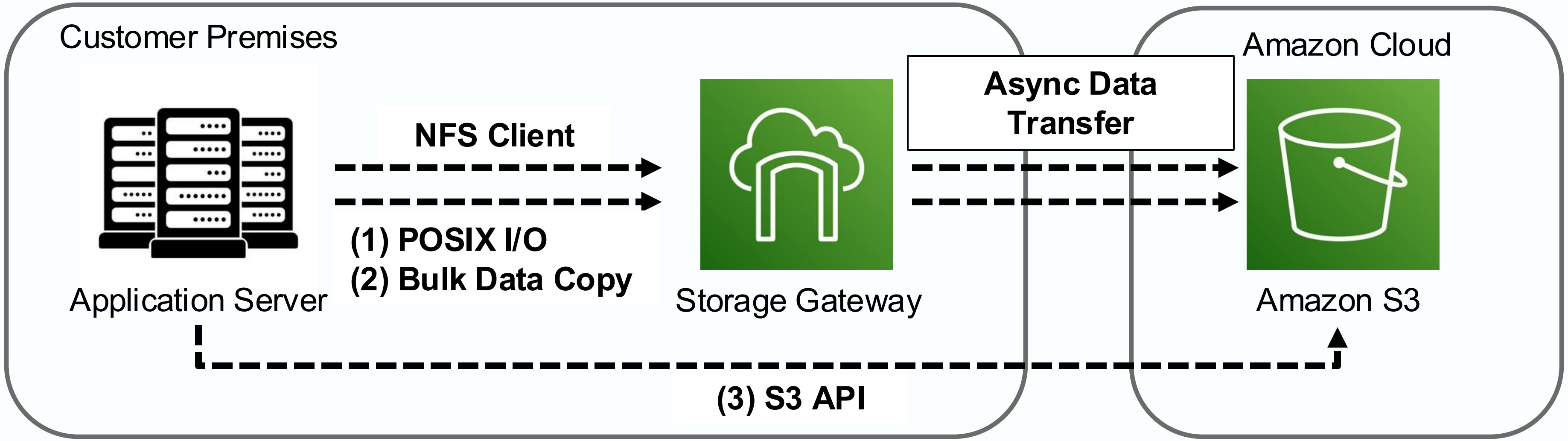
  (3) Direct S3 API transfers

Network-Bound vs. Protocol Bound?

Gateway Cache?

Latency?

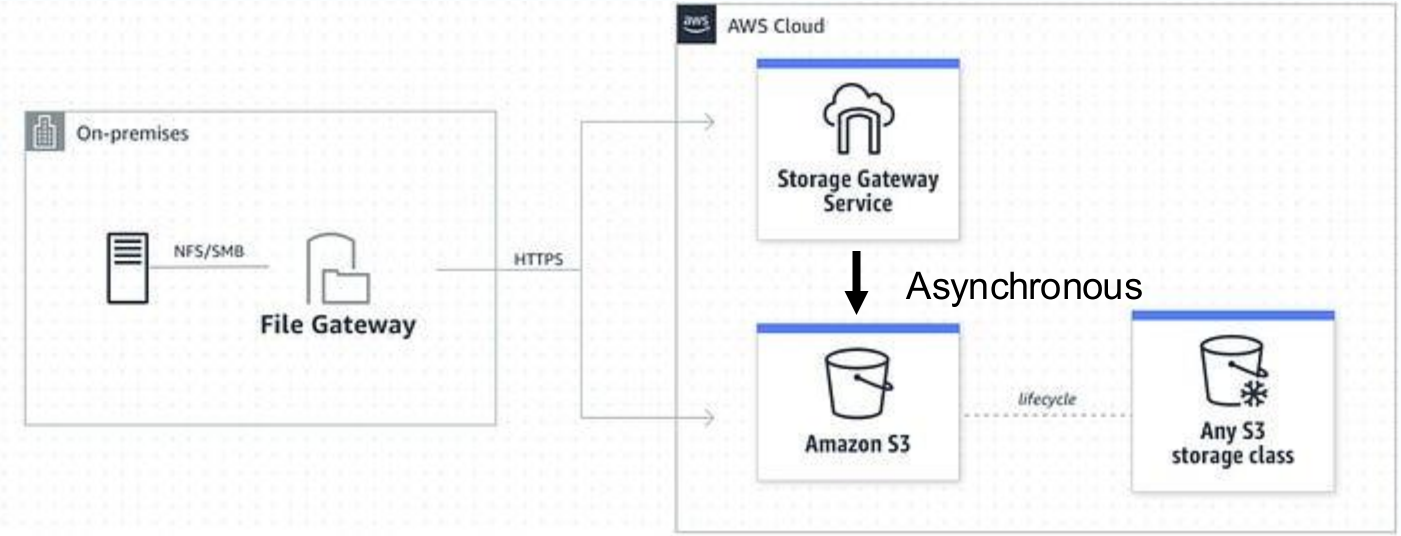
- We identify performance bottlenecks and provide guidance for practitioners.



Background

Background: AWS Storage Gateway

- **A hybrid cloud storage service.**
  - Connects on-premises environments to AWS cloud storage.
  - Provides standard storage protocols like NFS, SMB, etc.
- **File Gateway:**
  - Presents S3 objects as files in an NFS or SMB mount.
  - Provides a local cache for low-latency access to frequently used data.

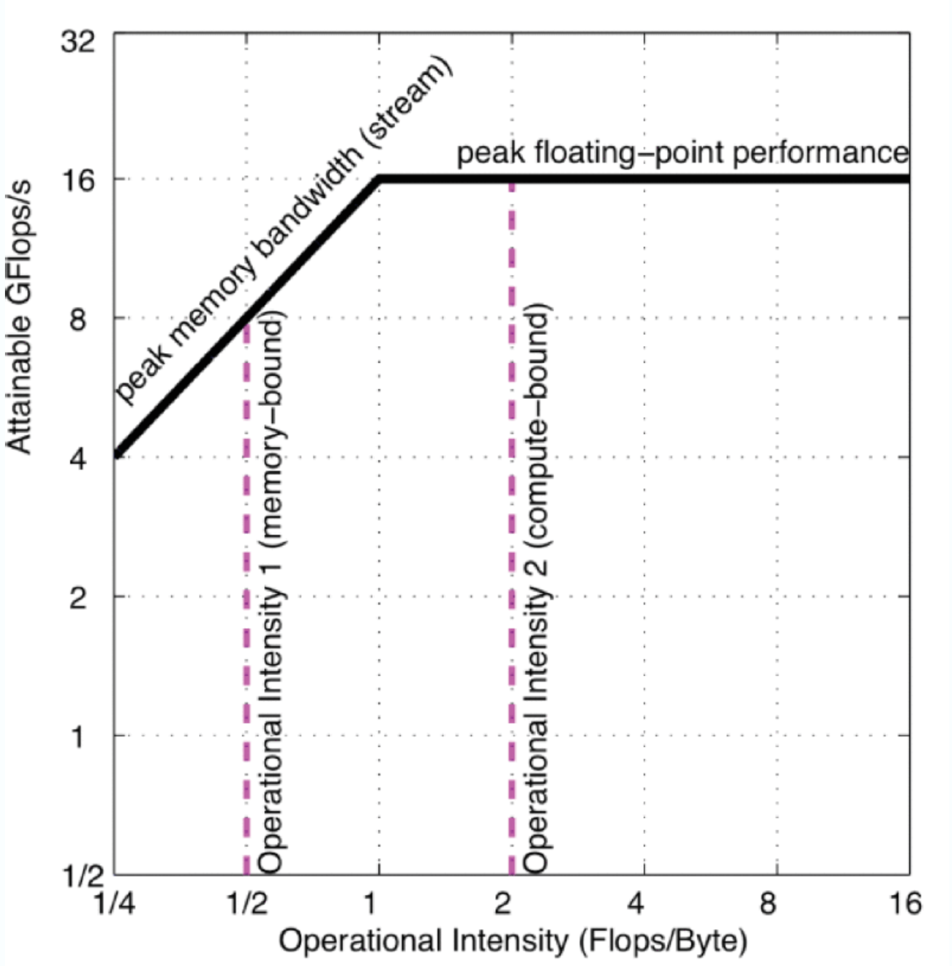


Background: The Roofline Model

- A visual model to understand the performance of computing systems.
- Relates computational performance (GFLOPs/s) to operational intensity (FLOPs/byte).
- Helps identify if a program is compute-bound or memory-bound.

$$\text{AP (GFlops/s)} = \min\big(\text{Peak Floating Point Performance}, \\ \text{Peak Memory Bandwidth} \times \text{OI}\big) \qquad (1)$$

$$\text{Operational intensity} = \frac{\text{Floating point operations}}{\text{Memory bytes transferred}} \qquad (2)$$



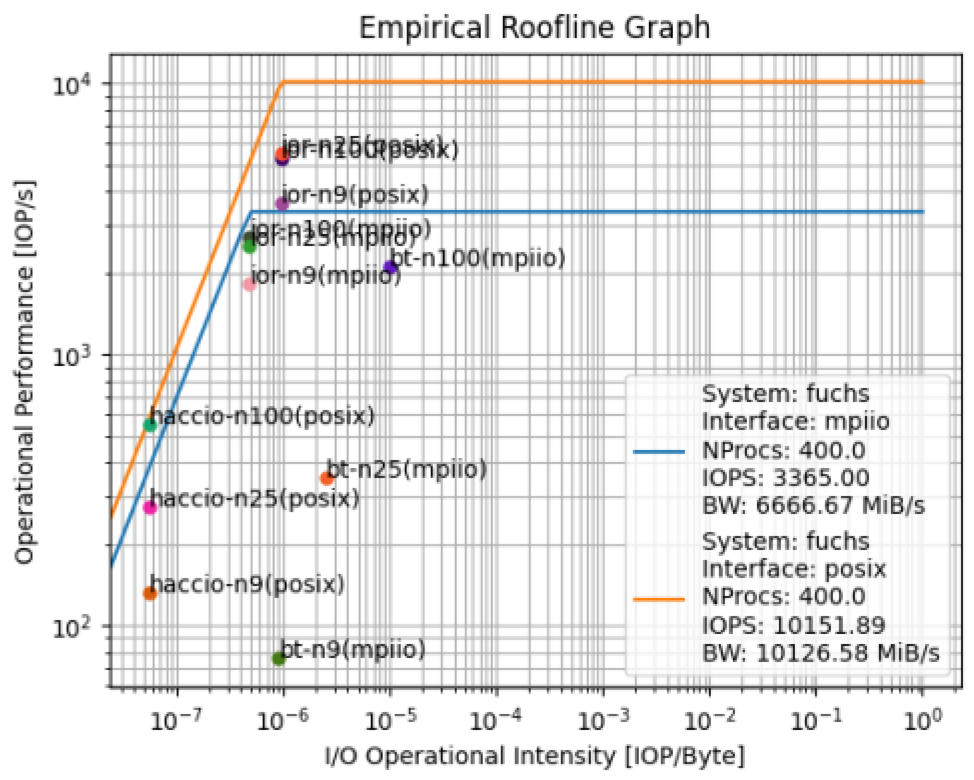
Example of a naïve Roofline Model

Background: The I/O Roofline Model

- A visual model to understand computing system performance
- Relates computational performance (GFLOPs/s) to operational intensity (FLOPs/byte).
- Helps identify if a program is compute-bound or memory-bound.
- We adapt this model for cloud storage, defining work as an abstract I/O operation (read, write, put, get, etc.).

$$\begin{aligned} \text{Attainable Performance} = Min(&\text{Peak IOPS},\\ &\text{Peak I/O Bandwidth} \times \ \text{I/O Intensity}) \end{aligned} \tag{5}$$

$$\text{I/O Intensity} = \frac{\text{Total I/O Operations}}{(\text{Read Bytes} + \text{Write Bytes})} \tag{6}$$



Methodology

Methodology: Three S3 Access Methods

Method (1): NFS-mounted Storage Gateway (Parallel I/O)

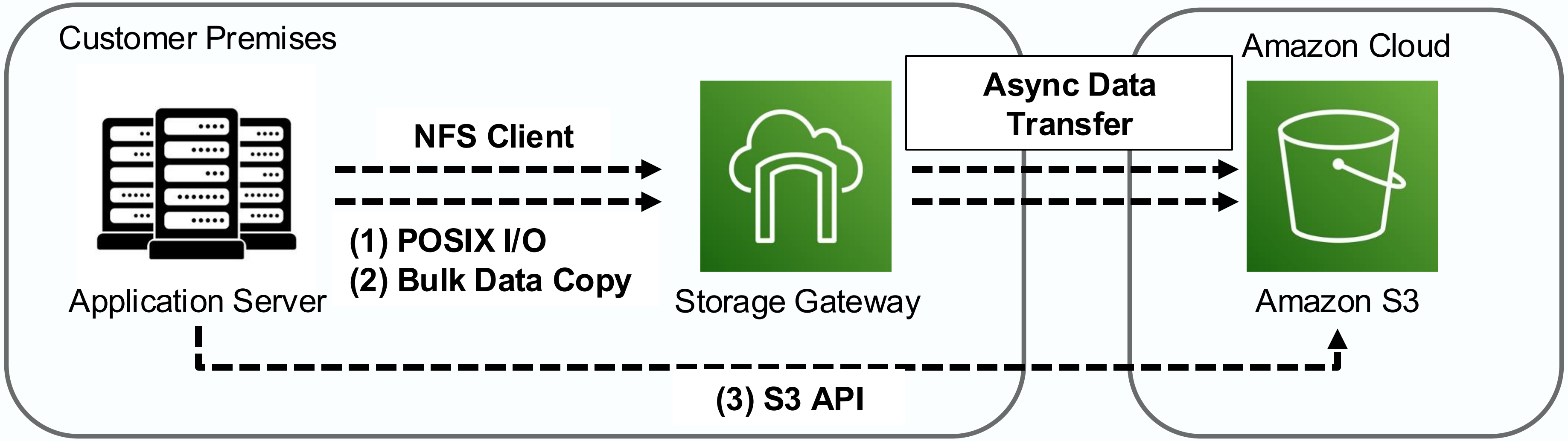
- IOR benchmark with file-per-task POSIX I/O.

Method (2): Data Migration via Storage Gateway

- Bulk data transfer using `cp` commands.

Method (3): Direct S3 API

- Custom C++ benchmark utilizing the AWS SDK's /Native API Object Put/Get operations.



Methodology: Three S3 Access Methods

Method (1): NFS-mounted Storage Gateway (Parallel I/O)

- Measures performance of HPC applications using a familiar I/O interface.
- Enables **zero-code-change** access for legacy HPC applications.

Method (2): Data Migration via Storage Gateway

- Measures efficiency of large, **explicit data transfers**.
- Represents data staging scenarios.

Method (3): Direct S3 API

- Measures **End-to-End Performance**, bypassing POSIX filesystem and local caching layers.
- Represents the standard programming interface for cloud-native scripts.

Methodology: Experimental Environment

- **PNNL HPC Cluster:**
    - CPU: 2x AMD EPYC 7502 (64 cores)
    - Memory: 264 GB DDR4
    - Network: 10 Gigabit Ethernet
    - Shared Storage: BeeGFS
- **AWS Storage Gateway:**
    - File Gateway mode
    - 300 GB local cache
    - Mounted via NFSv3
- **Experiments**
    - Varied number of nodes, tasks per node, file sizes, and transfer sizes.
    - Averaged over 3 runs.

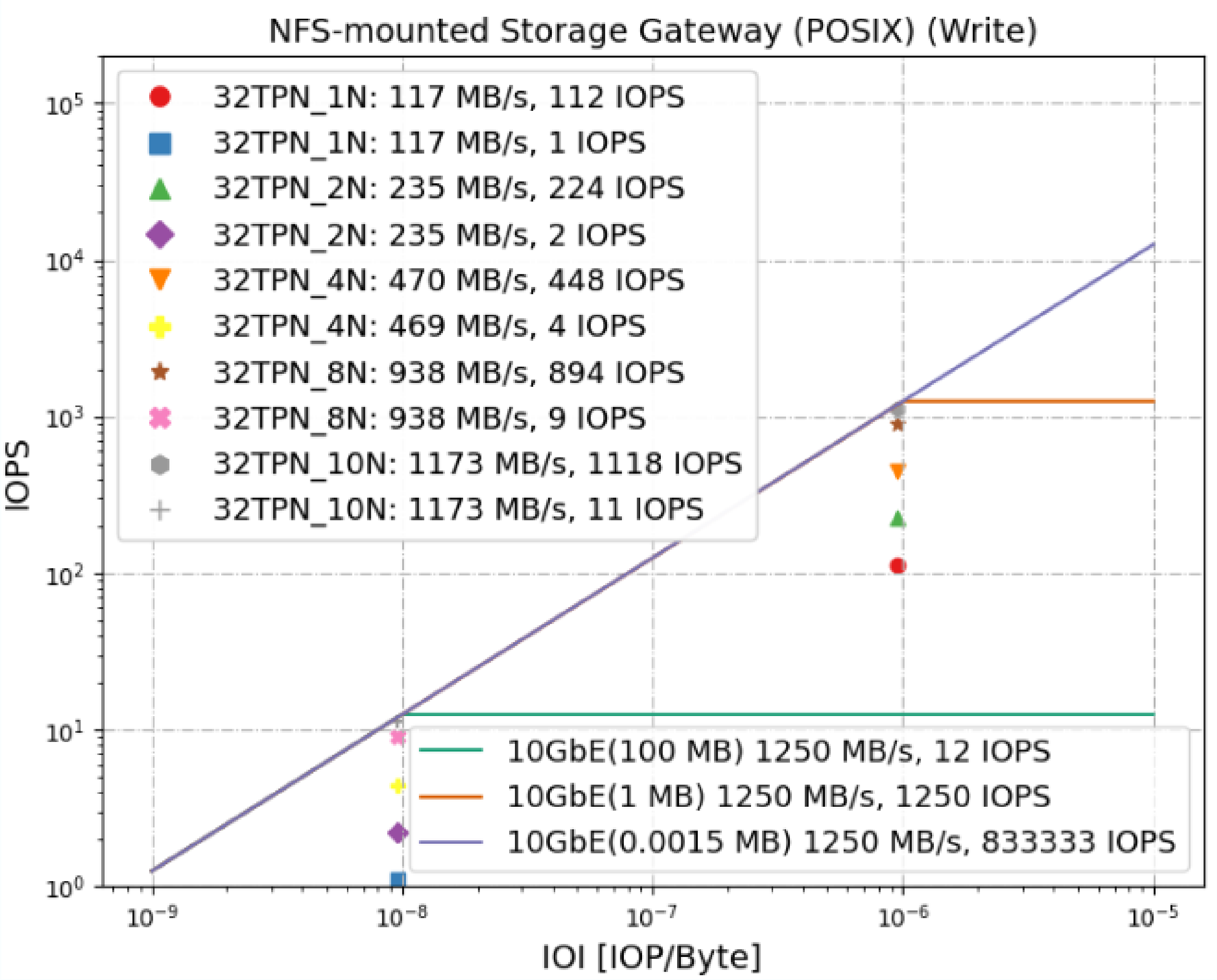
Evaluation Results

- Roofline Analysis

Roofline Analysis: Write Performance

Method 1 - POSIX (NFS Gateway):

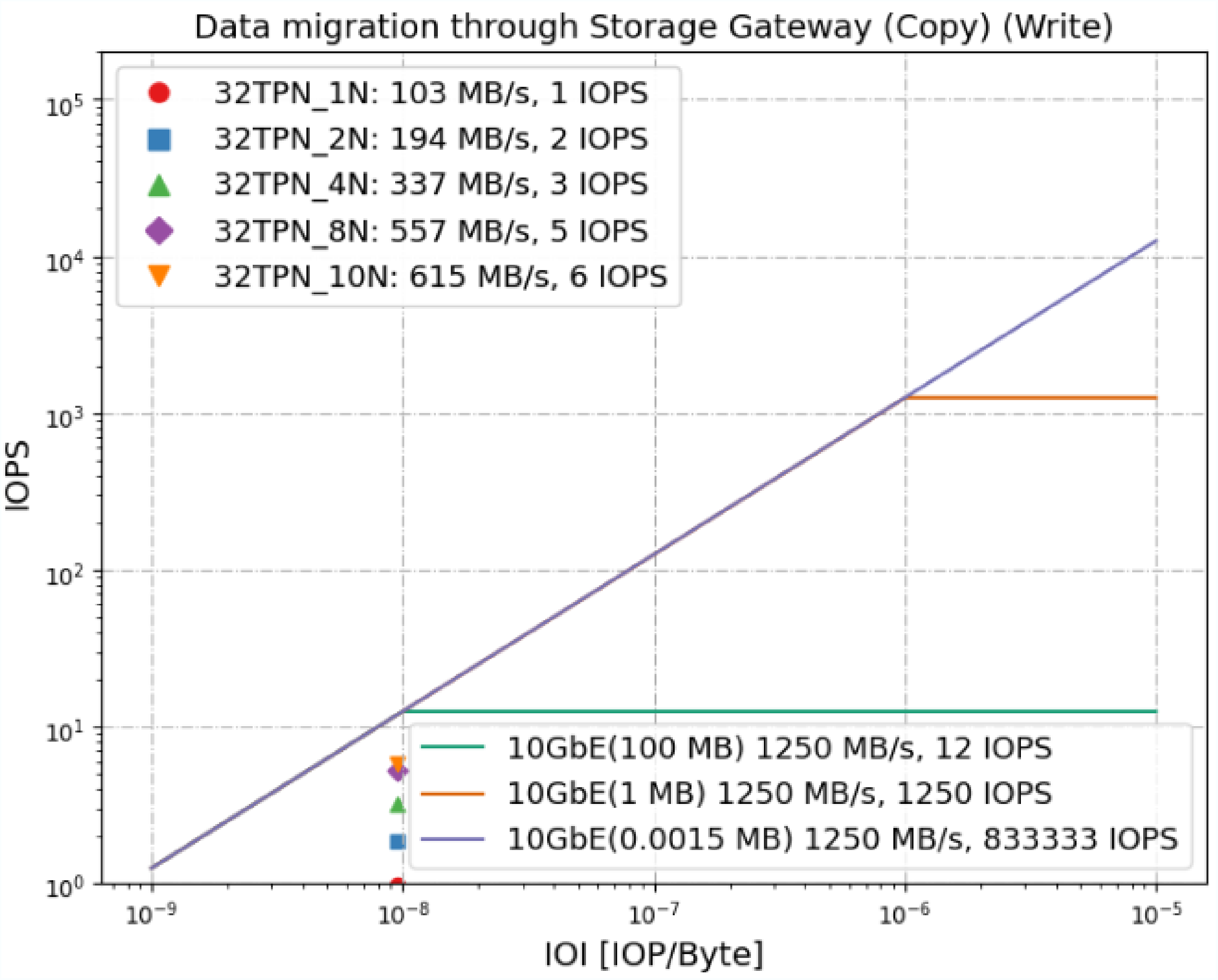
- Performance limited by the 10GbE network.
- Effectively utilizes available bandwidth.
- Performance scales with task and data volume.



Roofline Analysis: Write Performance

Method 2 - Copy (to Gateway):

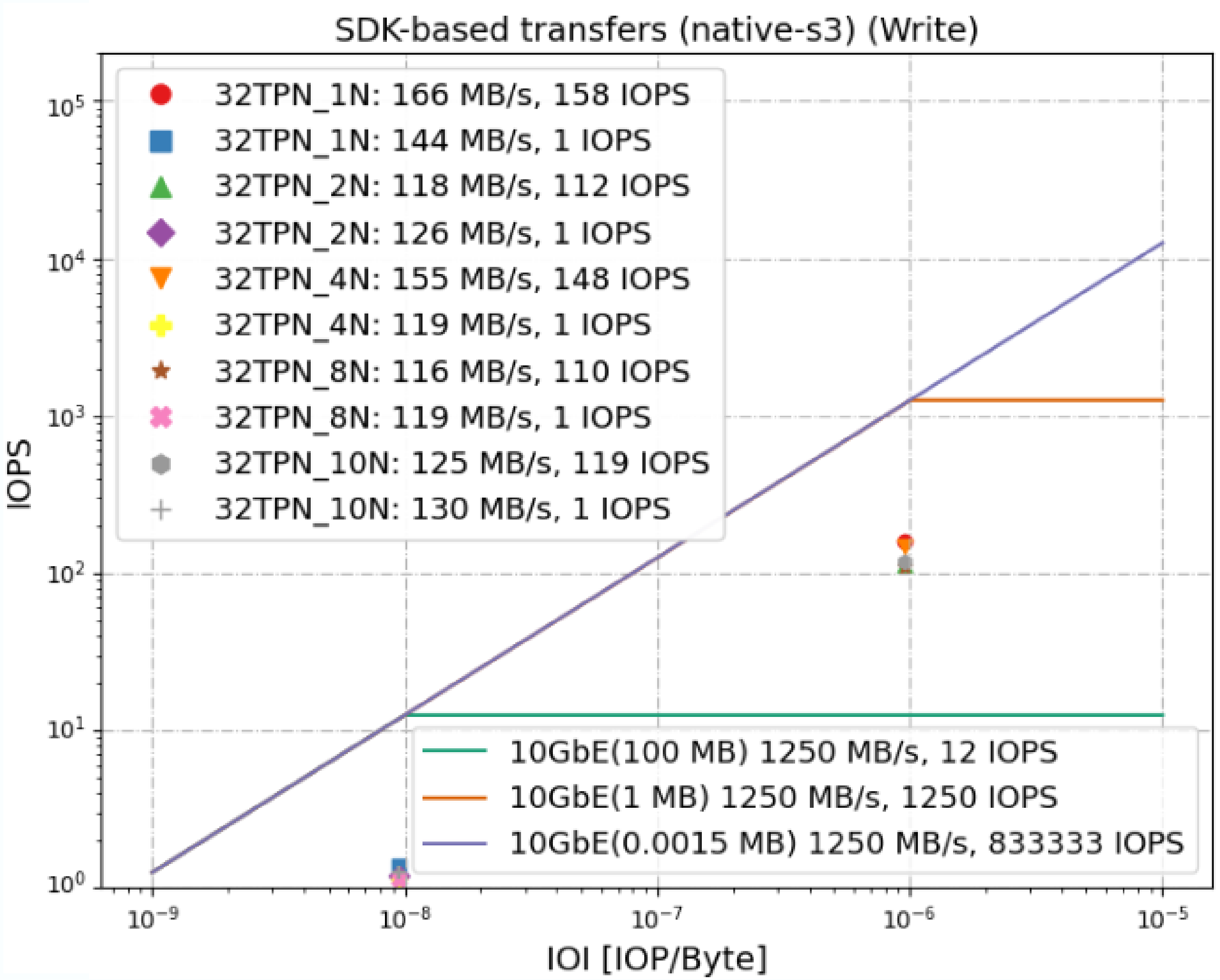
- Performance scales with parallelism, but not as well as POSIX.



Roofline Analysis: Write Performance

Method 3 - S3 API:

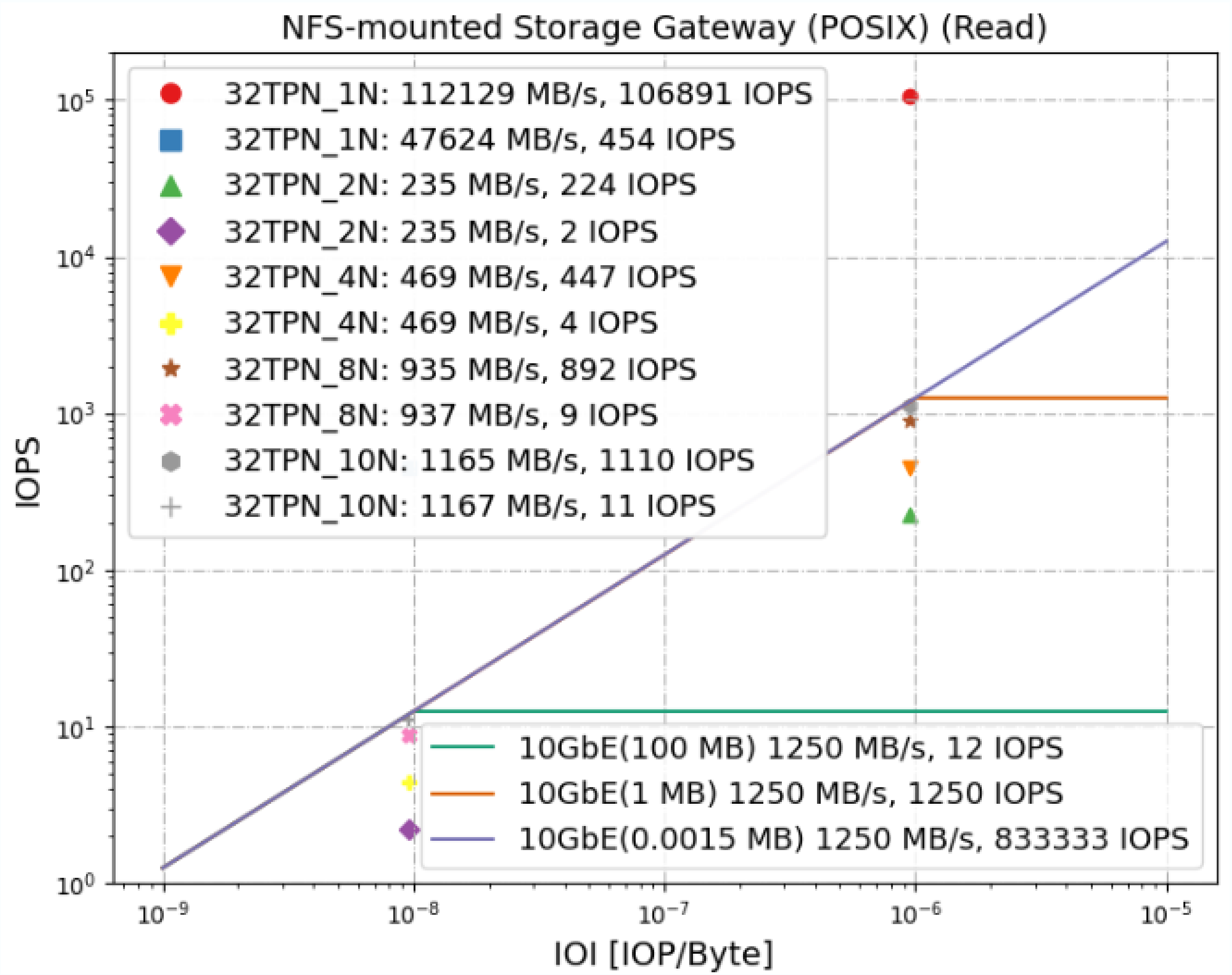
- Significantly lower performance.
- Does not scale well with parallelism.
- Likely limited by protocol overhead.



Roofline Analysis: Read Performance

POSIX (NFS Gateway):

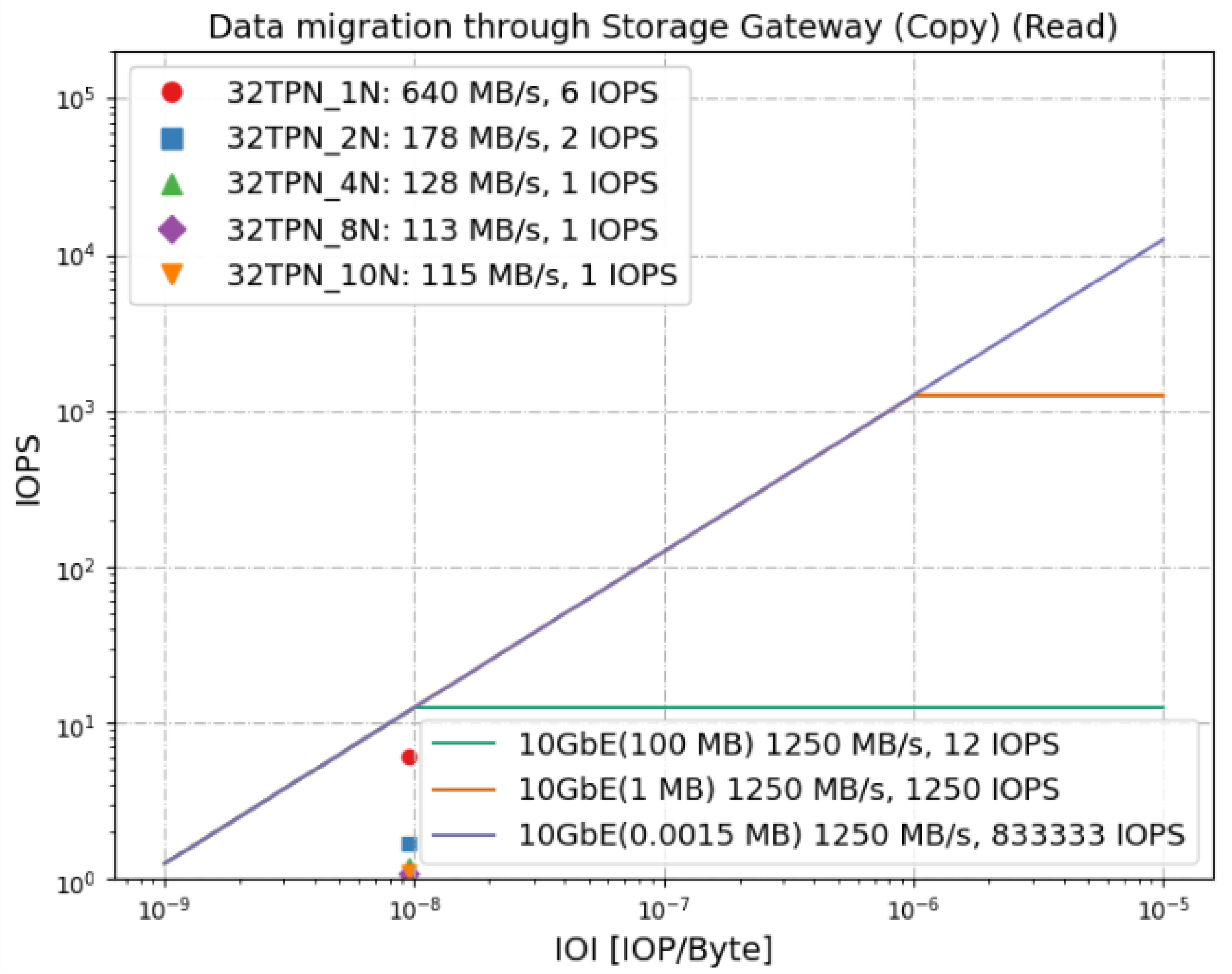
- Very high performance, especially with caching.
- Outliers show performance far beyond network limits, indicating strong cache effects.



Roofline Analysis: Read Performance

Copy (to Gateway):

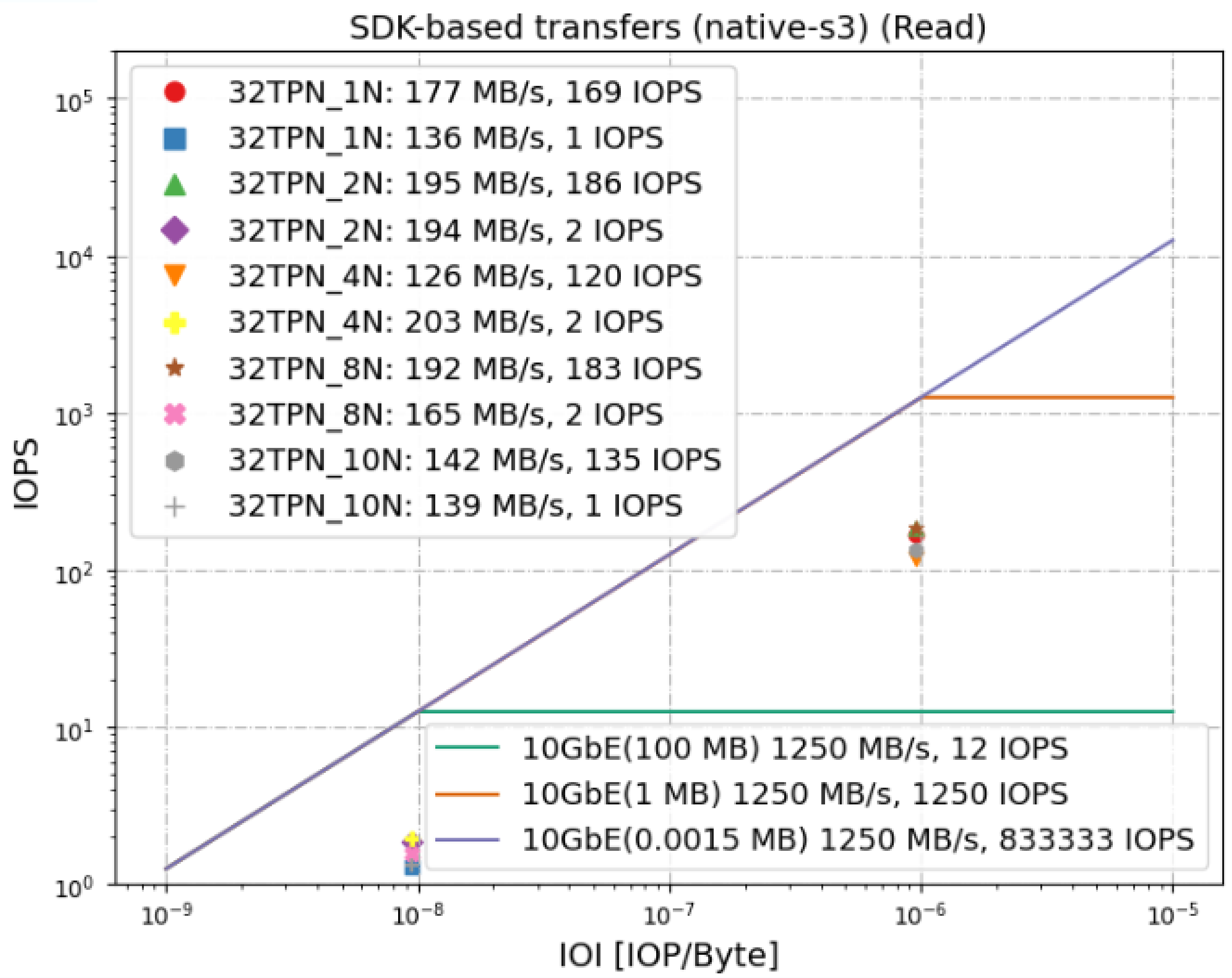
- Best performance with a single node, decreases with more nodes.
- Does not reach network limits.



Roofline Analysis: Read Performance

Native S3 API:

- Limited performance, similar to write.



The Hidden Cost: Asynchronous Latency

Storage Gateway writes are **asynchronous**—data is not immediately available in S3 upon local completion.

Measurement Method: Latency is the time difference between local write completion and S3 object availability, monitored via **AWS API calls**.

The Penalty: Latency is high for bulk operations:

- Method 1 - IOR POSIX Write: Average S3 availability latency **7 minutes (421.17 s)**.
- Method 2 - Data Copy Write: Average S3 availability latency **80 minutes (4833.5 s)**.

Conclusion: This latency can significantly degrade the perceived end-to-end performance for workflows needing immediate data access.

Take Aways: Performance vs. Latency Trade-off

POSIX on NFS Gateway

- **Pros:**
  - Highest throughput, achieves **network saturation** (~1.17 GB/s), scalable
  - Familiar interface for HPC apps.
- **Cons:** High latency for data to appear in S3.

Data Migration via NFS Gateway

- **Pros:**
  - Good for quick staging of small working sets (cache-hit downloads peak ~600 MB/s).
  - Familiar interface for HPC apps.
- **Cons:** High latency for data to appear in S3, impractical for timely transfers.

Direct S3 API

- **Pros:** Low latency, immediate data availability.
- **Cons:** Poor scalability, lower throughput.

Conclusions

- We presented a **roofline-based analysis** of three AWS S3 access methods.
- NFS-mounted Storage Gateway offers the highest throughput but with high latency.
- Direct S3 API provides low latency but with limited scalability.
- Our work provides quantitative guidance for choosing the right S3 access method.
- The I/O roofline model is a valuable tool for analyzing cloud storage performance.

Future Work

- Investigate other cloud providers and storage services.
- Explore different network configurations (e.g., AWS Direct Connect).
- Develop more sophisticated models for predicting cloud storage performance.
- Optimize S3 access patterns for specific HPC workloads.

Thank you! Questions?

- Meng Tang
- mtang11@hawk.illinoistech.edu

- Zhaobin Zhu
- zhu@uni-mainz.de