Lecture 12: Tail Tolerance

CSCI4180

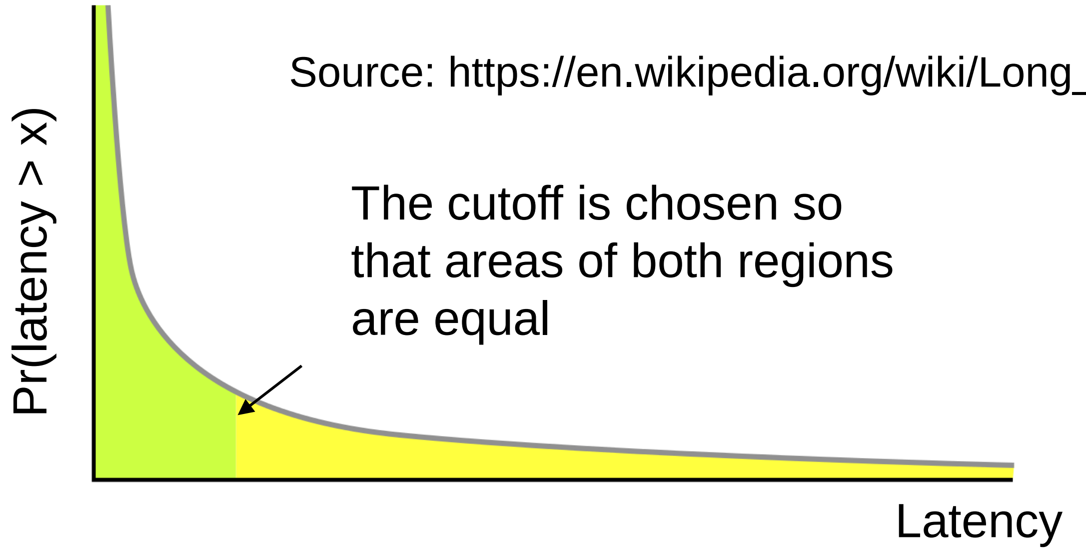
Patrick P. C. Lee

Goal of this Lecture

- Why is tail latency bad?
- How Google handles tail latency?
- What are the tail latency issues in NetApp's data center storage?

- References:
  - Dean and Barroso, "The Tail at Scale", CACM 2013
  - Hao et al., "The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments", FAST 2016

What is a Tail?



- **Tail** refers to the latency in high percentiles
  - e.g., 95-percentile or 99-percentile latencies
- A tail is significant (or a heavy tail) if the high-percentile latency is high
  - Mathematically, P(latency > x) is significant even for large x
- **Variability** of response time leads to high tail latency

Why Tails are Bad?

- Dominate the overall performance
  - e.g., stragglers in MapReduce determine the overall job performance
- Violation of service-level agreements
  - e.g., SLA may state 99.9% of requests are completed within 300 milliseconds
- Bad user experience
  - e.g., interactive queries

Why Tails are Bad?

- What should operators do with tails?

- Treat slow servers as hard failures?
  - Replace slow servers with new ones
  - Problem: there could be many false positives

Why Variability Exists?

- Resource sharing
  - CPUs / disks within machines
  - Switches or shared file systems
- Background jobs
  - Daemons
  - Maintenance
  - Garbage collection
- Queueing
- Energy management
  - Power limits of modern CPUs
  - Power saving modes

Variability Amplified at Scale

- Parallelization is often used to reduce latency, but also amplifies variability
- Scenario:
  - A request is broken into sub-operations that are issued to multiple servers in parallel
  - These sub-operations must all complete
  - Let Pr(latency > 1s) = 0.01
  - Sub-operations of a request issued to 100 servers
  - $\Pr(\text{request} > 1\text{s}) = 1 - (1 - 0.01)^{100} = 0.63$
- What happen if a request is issued to one server?

Variability Amplified at Scale

	50%ile latency	95%ile latency	99%ile latency
One random leaf finishes	1ms	5ms	10ms
95% of all leaf requests finish	12ms	32ms	70ms
100% of all leaf requests finish	40ms	87ms	140ms

Measurements from a real Google service

- ➢ The slowest 5% of the requests to complete is responsible for half of the total 99%-percentile latency

Reducing Variability

- Differentiating service classes and higher-level queuing
- Reducing head-of-line blocking
  - HOL blocking: blocked by large requests at the front of a queue
  - Solution: break a long-running request into smaller slices and allow them interleave with others
- Managing background activities and synchronized disruption
  - Synchronizing all background activities reduces variability

Reducing Variability

- What about caching?
  - Cache the data that resides in slow nodes
- Caching is useful only when the entire working set can reside in a cache

Living with Variability

- Eliminating all latency variability is infeasible
  - Too large scale
  - Too complex

- Google develops tail-tolerant techniques that work around temporary latency hiccups

Within Request Short-Term Adaptations

- Replication of data items across servers
  - Provide both fault tolerance and tail tolerance

- **Hedged requests**:
  - Send a request to the primary server
  - Send the same request (i.e., hedged request) to a different server after some brief delay
  - Cancel any outstanding request if the response is received

Within Request Short-Term Adaptations

- **Tied requests**
  - Enqueue copies of a request in multiple servers simultaneously
  - Allow servers to communicate updates on the status of these copies to each other (e.g., send a cancellation message when the request is processed)

- In Google's implementation of tied requests, medium latency is reduced by 16%, and 99.9-percentile latency is reduced by 40%

Cross-Request Long-Term Adaptations

- **Micro-partitions**:
  - Generate many more partitions than servers
  - Perform load balancing at the (more fine-grained) partition level
  - Example: tablets in BigTable
- **Selective replication**:
  - Detect or predict certain items (or micro-partitions) that are likely to cause load imbalance and create more replicas of them
  - Spread the load of these “hot” items

Cross-Request Long-Term Adaptations

- **Latency-inducted probations:**
  - If a server is detected to respond slowly, put it on probation
  - Exclude requests to the slow servers
  - Issue shadow requests to the slow servers and collect statistics from them; bring the servers back to normal if they are ready

Special Cases

- **Take “good enough” responses**:
  - Once a sufficient fraction of all servers responded, the user accepts the slightly incomplete (“good-enough”) results for better end-to-end latency
  - Example: PageRank
- **Canary requests**:
  - Rather than initially send a request to thousands of servers, send it first to one or two servers
  - The remaining servers are only queried if there is a successful response from the canary in a reasonable period of time
  - Rationale: don’t waste efforts

Summary

- Seven tail-tolerant techniques:
  - Reactive or proactive
- Rationale: optimize for the common case while providing resilience against uncommon cases

Tails in Storage

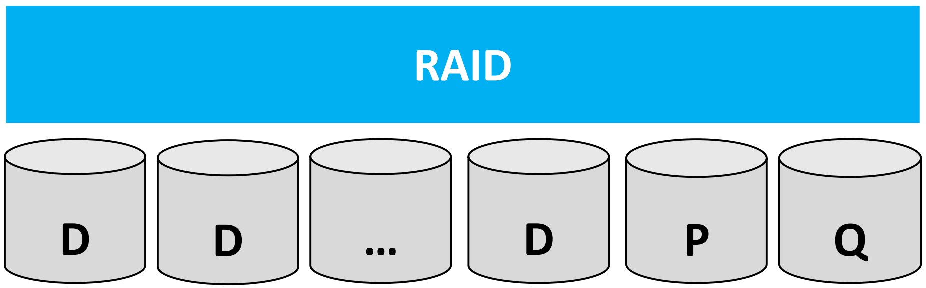
- What about the tail latency problem in storage environments?
- In reality, performance variability also exists in hard disks and solid-state drives (SSDs)
  - Hard disks
    - Weak disk head, bad packaging, missing screws, broken/old fans, too many disks/box, firmware bugs, bad sector remapping
    - Bandwidth drops by 80%, and introduces seconds of delay
  - SSDs:
    - Firmware bug, GC, …
    - 4 – 100x slowdown

Field Study (NetApp)

- Study of over 450,000 disks and 4000 SSDs
  - Deployed as RAID groups
  - 87 days on average
  - Total: 857 million disk hours and 7 million SSD hours

	Disk	SSD
#RAID groups	38,029	572
#Data drives per group	3-26	3-22
#Data drives	458,482	4,069
Total drive hours	857,183,442	7,481,055
Total RAID hours	72,046,373	1,072,690

Metric

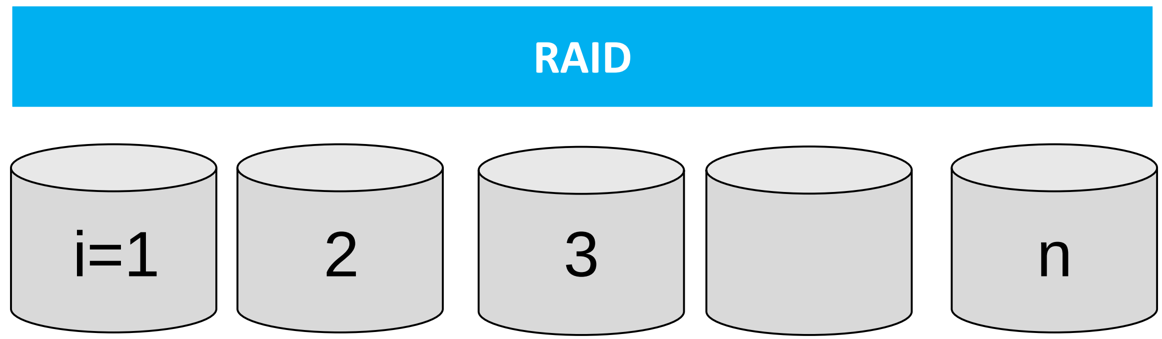


RAID group: P and Q drives are parity drives that provide redundancy. Parities are non-rotational

Primary metric:

- L_i = average I/O latency per drive
  - L_{median} = median $(L_1 \ldots L_N)$
- **Slowdown** → $S_i = L_i / L_{median}$
  - **Slow drive** hour → **$S_i \geq 2x$**

Full-Stripe Workloads



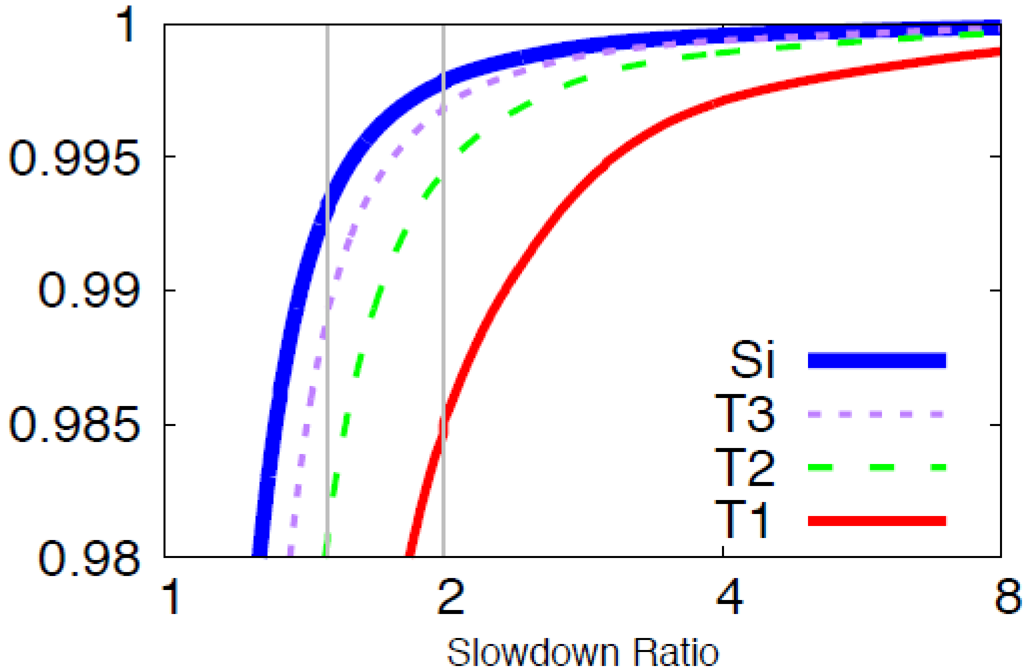
Hourly average latency (ms)	10	9	22	8	11
Slowdown	1.0x	0.9x	2.2x	0.8x	1.1x
	L_{med}		T1		T2

- Most RAID I/O requests are **full-stripe**
- Three largest slowdowns (tails): T1, T2, T3
  - T1 = 2.2, T2 = 1.1, T3= 1

Measurement Questions

- How often does slowdown happen?
- Does slowdown depend on the workloads?
- What are the possible root causes of the slowdown?
- What happens after slowdown occurs?

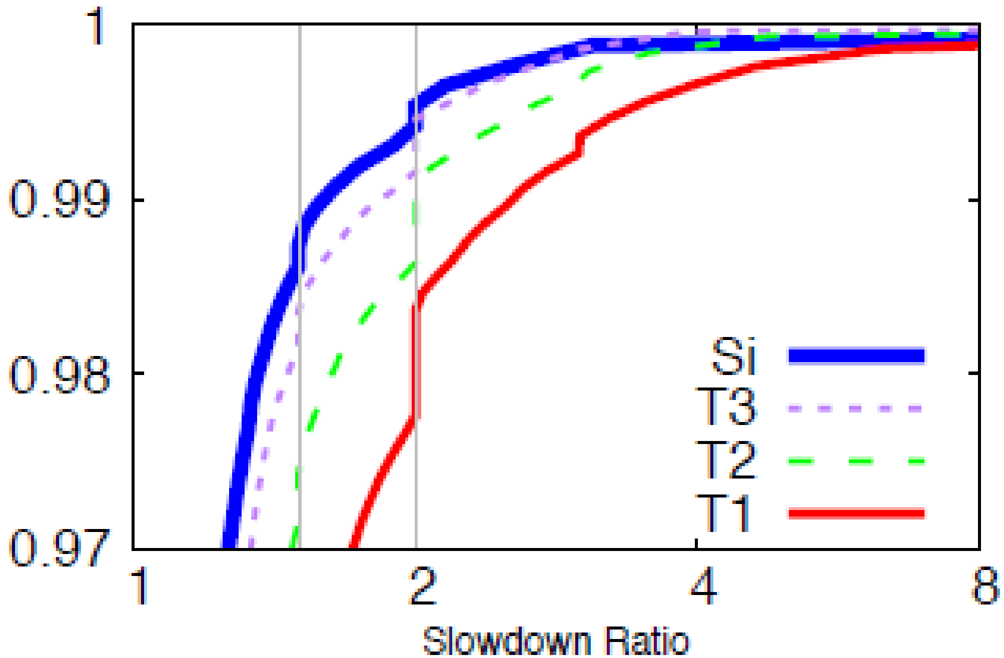
Slowdown Distribution (Disks)



Cumulative distributions of disk hours versus slowdown

- Slowdowns are significant:
  - Some disks have ≥ 2x slowdown (i.e., Si ≥ 2) in 0.22% of time
    - In 1000 disk hours, 2 hours are slow
- Longest tails (T1) appear 1.5% of time

Slowdown Distribution (SSDs)



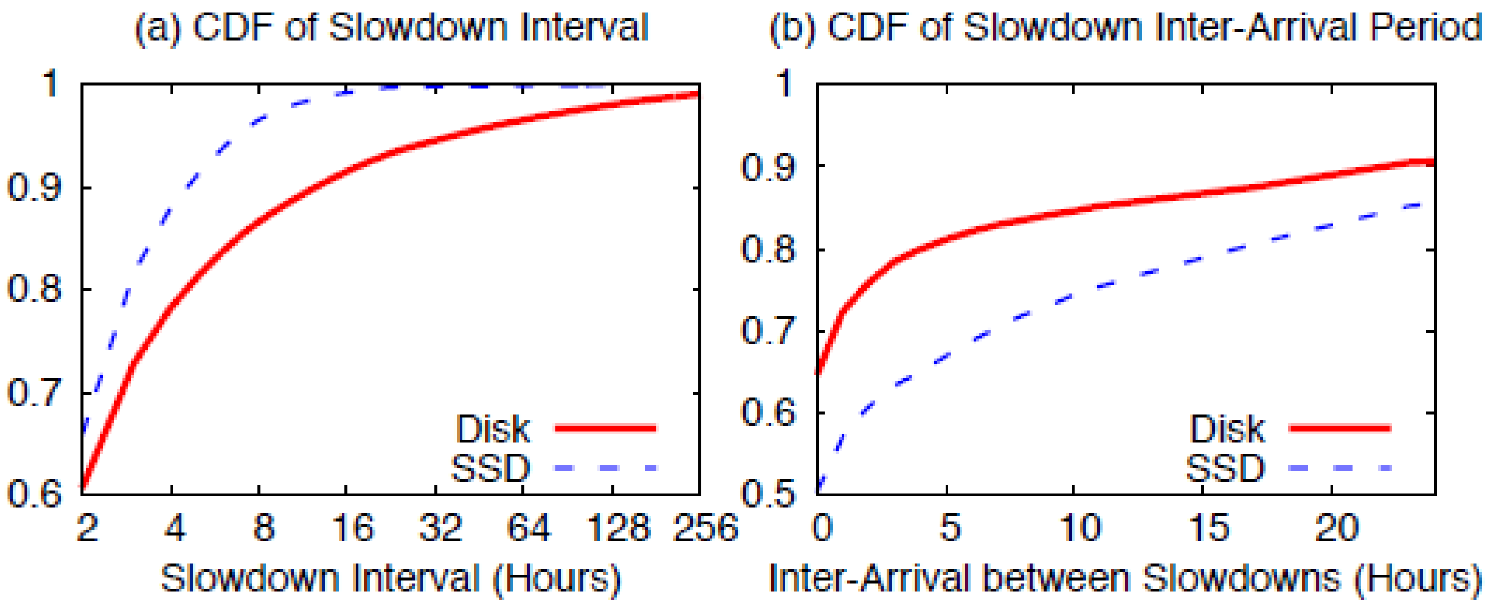
Cumulative distributions of disk hours versus slowdown

- SSDs experience even more slowdown than disks (0.58% of time)
  - Faster devices don’t imply fewer slowdowns
- Longest tails appear at 2.23% of time

Observations

- Storage performance instability is not uncommon
- Storage tails appear at a significant rate
- Tail-tolerant RAID has a significant potential to increase performance stability
  - Move the tail from T1 to T3

Temporal Behavior

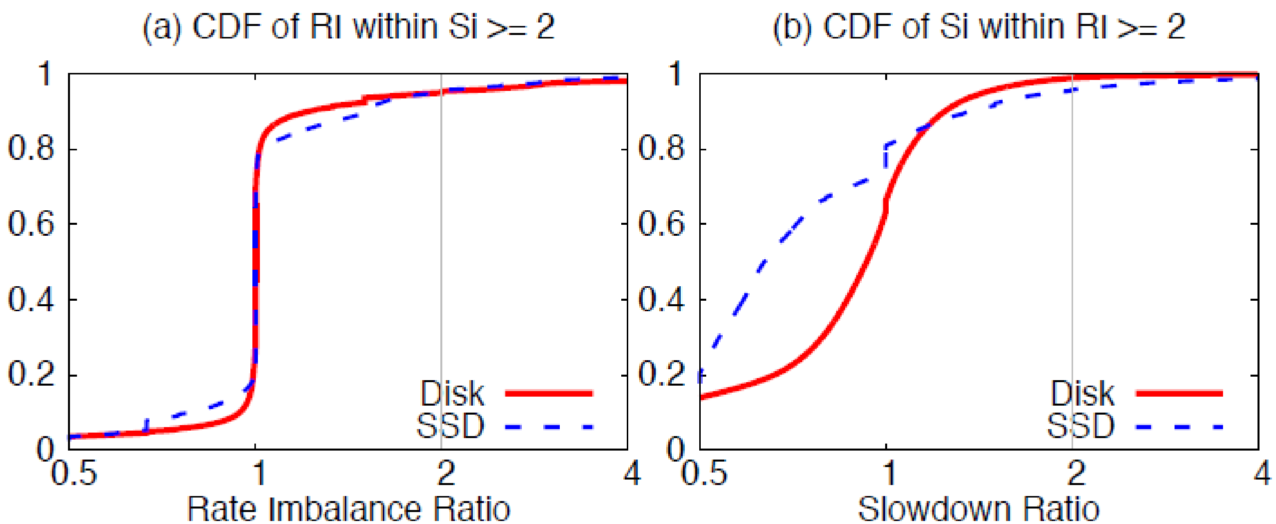


- Slowdown can persist over several hours
- Slowdown has a high temporal locality
  - 90% and 85% of disk and SSD slowdown occurrences from the same drive happen within the same day of the previous occurrence, respectively

Slowdown Population

- How many drives have experienced at least a slowdown in their lifetimes?
  - 25% of disks have seen ≥ 2x slowdown
  - 29% of SSDs have seen ≥ 2x slowdown
- A large proportion of drives experience slowdown

Workload Dependence



Slowdown vs I/O rate imbalance

- R_i = I/O rate of drive i
- Rate imbalance: $RI = \max(R_i) / R_{med}$
- (Left) Only 5% of slow drives have high RI
- (Right) Only 5% of imbalances occur in slow drives

Workload Dependence

- Similar observations for I/O size

- Conclusion: Slowdown is independent of I/O rate and size imbalance

- Further analysis shows that slowdown is a “silent” performance problem
  - Slowdowns are not accompanied with any explicit drive events

More Analysis

- Older disks tend to have more slowdowns, but no age correlation for SSDs
- Single-level-cell (SLC) SSDs slightly outperform multi-level-cell (MLC) SSDs in terms of performance stability
- Slowdown depends on SSD vendors
- No correlation with time of day
- Slow drive replacement rate is low
  - Unplug: 4-8% (within 24 hours after a slowdown)
  - Replug: 89-100% of unplugged drives are replugged

How to be Tail-Tolerant?

- **Reactive**: if a drive didn’t return data after a timeout, perform an extra read to parity drives

- **Proactive**: perform extra reads to parity drives concurrently with original I/Os

- **Adaptive**: if the reactive policy is triggered repeatedly on the same drive, use the proactive policy

How to be Tail-Tolerant?

- Trade-offs:
  - Reactive: slow response time
  - Proactive: extra read traffic

- What about writes?
  - Can be mitigated by caching

Potential Directions

- More fine-grained analysis is needed
  - Traces are hourly averages
  - Absence of other metrics
- Correlations to failure events are important
- Tail mitigation techniques
  - What about distributed file systems (e.g., HDFS)?
- Can we *predict* tails?