CS 425 / ECE 428 Distributed Systems Fall 2015

Indranil Gupta (Indy)

Measurement Studies

Lecture 23

Nov 10, 2015

Reading: See links on website

Motivation

- We design algorithms, implement and deploy them as systems
- But when you factor in the real world, unexpected characteristics may arise
- Important to understand these characteristics to build better distributed systems for the real world
- We'll look at three systems: Kazaa (P2P system), AWS EC2 (Elastic Compute Cloud), Hadoop

How do you find characteristics of these Systems in Real-life Settings?

- Write a crawler to crawl a real working system (e.g., p2p system), or a benchmark (e.g., Hadoop)
- Collect *traces* from the crawler/benchmark
- Tabulate the results

- Papers contain plenty of information on how data was collected, the caveats, ifs and buts of the interpretation, etc.
  - These are important, but we will ignore them for this lecture and concentrate on the raw data and conclusions

Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload

Gummadi et al
Department of Computer Science
University of Washington

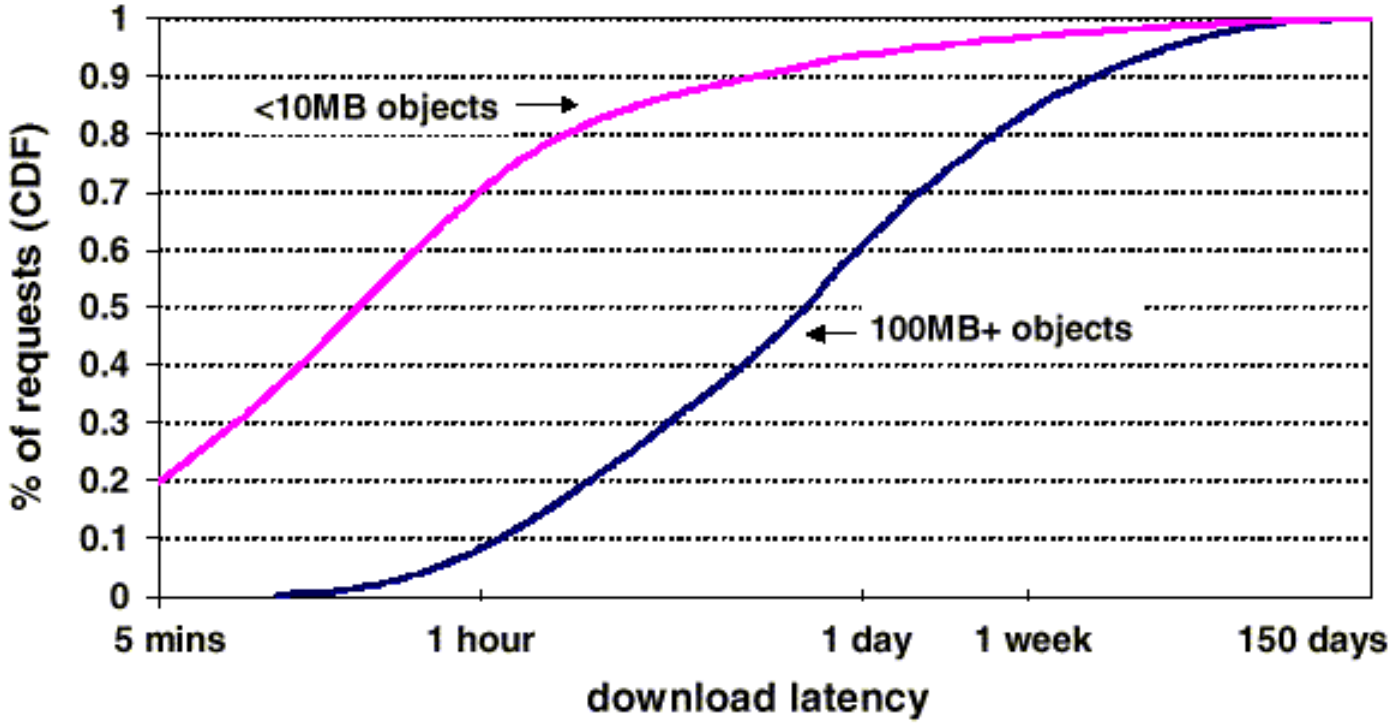
Acknowledgments: Jay Patel

What They Did

- Kazaa: popular p2p file sharing system in 2003
- 2003 paper analyzed 200-day trace of Kazaa traffic
- Considered only requests going from U. Washington to the outside
- Developed a model of multimedia workloads

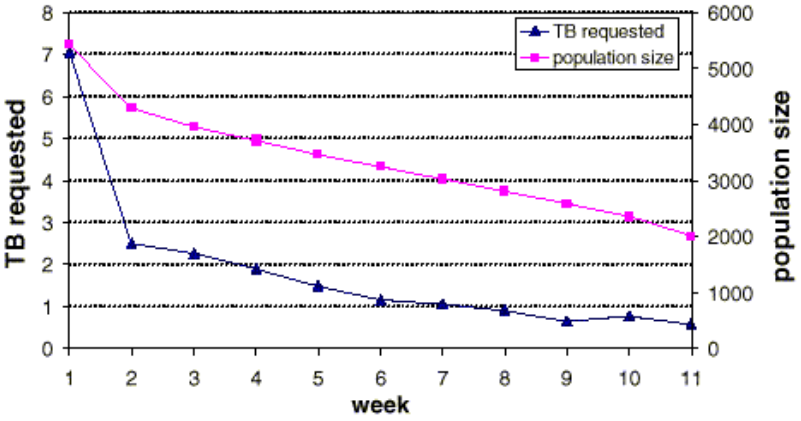
User characteristics (1)

- Users are patient



User characteristics (2)

- Users slow down as they age
  - clients "die"
  - older clients ask for less each time they use system

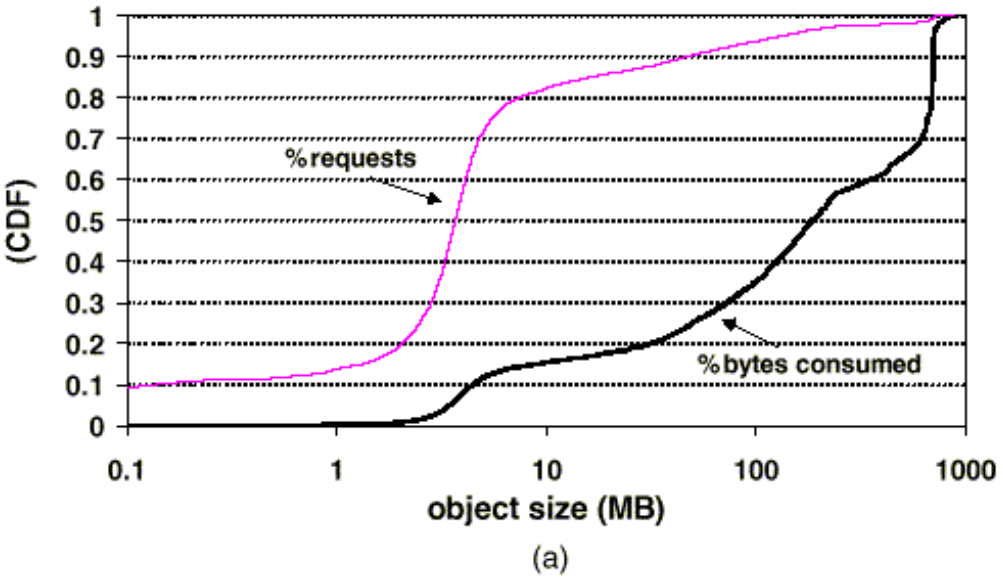


User characteristics (3)

- Client availability = time client present in system
  - Tracing used could only detect users when their clients transfer data
  - Thus, they only report statistics on client activity, which is a *lower bound* on availability
  - Avg session lengths are typically small (median: 2.4 mins)
    - Many transactions fail

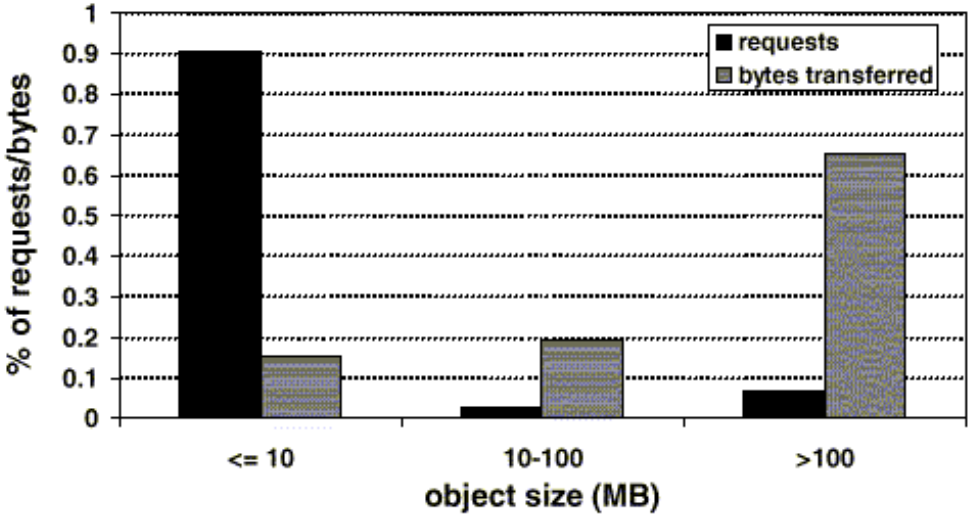
Object characteristics (1)

- Kazaa is not one workload



(a)

•Dichotomy: Small files vs. large files

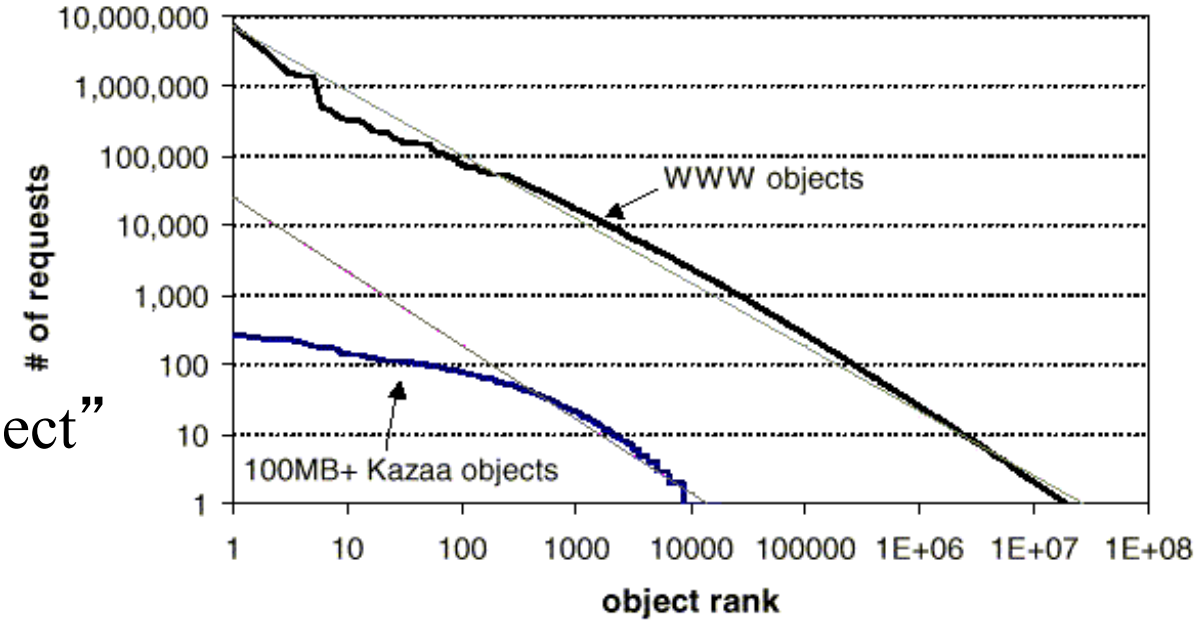


Object characteristics (2)

- Kazaa object dynamics
  - Kazaa clients fetch objects **at most once**
    - 94% time a Kazaa client requests a given object at most once
    - 99% of time a Kazaa client requests a given object at most twice
  - Popularity of objects is often short-lived
  - Most popular objects tend to be recently-born objects
  - Most requests are for old objects (> 1 month)
    - 72% old – 28% new for large objects
    - 52% old – 48% new for small objects

Object characteristics (3)

- Kazaa is not Zipf, but it is heavy-tailed
- Zipf's law: popularity of *i*th-most popular object is proportional to $i^{-\alpha}$, (α: Zipf coefficient)
- Web access patterns are Zipf
- Authors conclude that Kazaa is not Zipf because of the at-most-once fetch characteristics



Caveat: what is an "object" in Kazaa?

Results Summary

1. Users are patient
2. Users slow down as they age
3. Kazaa is not one workload
4. Kazaa clients fetch objects at-most-once
5. Popularity of objects is often short-lived
6. Kazaa is not Zipf

An Evaluation of Amazon's Grid Computing Services: EC2, S3, and SQS

Simson L. Garfinkel

SEAS, Harvard University

What they Did

- Did bandwidth measurements
  - From various sites to S3 (Simple Storage Service)
  - Between S3, EC2 (Elastic Compute Cloud) and SQS (Simple Queuing Service)

Host	Location	N	Read Avg	Read top 1%	Read Stdev	Write Avg	Write top 1%	Write Stdev
Netherlands	Netherlands	1,572	212	294	34	382	493	142
Harvard	Cambridge, MA	914	412	796	121	620	844	95
ISP PIT	Pittsburgh, PA	852	530	1,005	183	1,546	2,048	404
MIT	Cambridge, MA	864	651	1,033	231	2,200	2,741	464
EC2	Amazon	5,483	799	1,314	320	5,279	10,229	2,209

Units are in bytes per second

Table 2: Measurements of S3 read and write performance in KBytes/sec from different locations on the Internet, between 2007-03-29 and 2007-05-03.

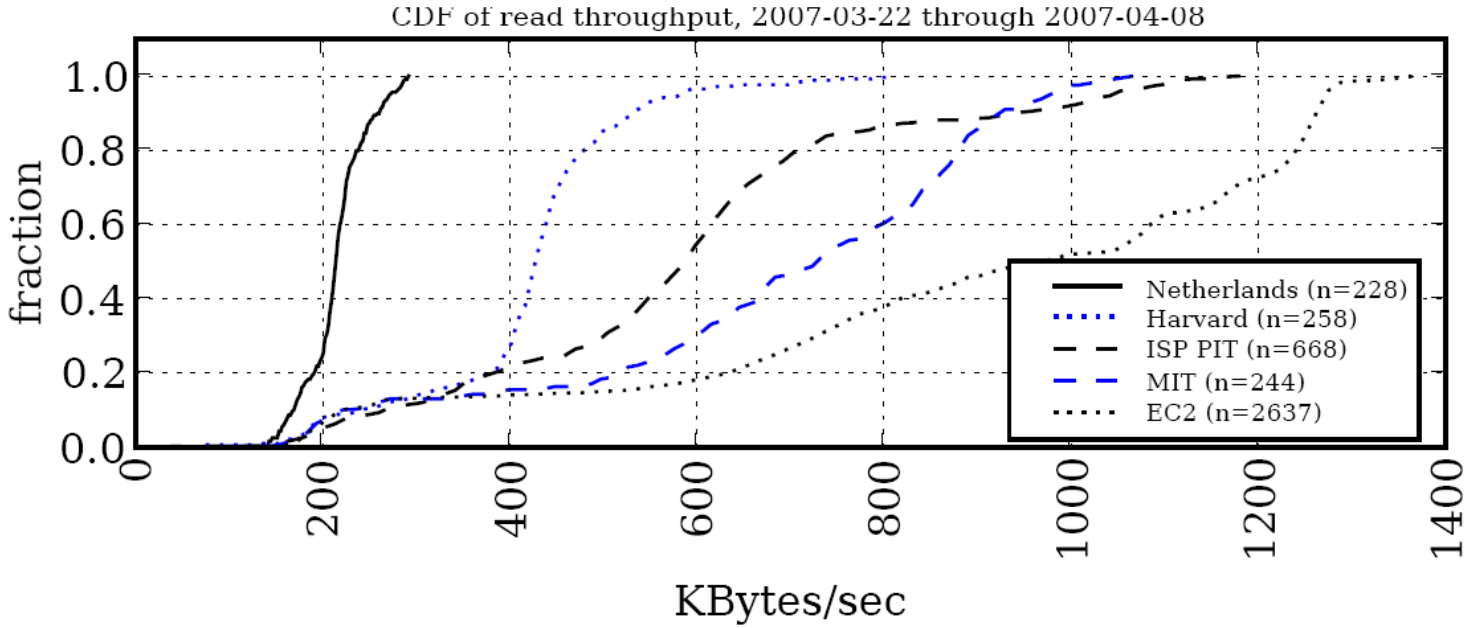


Figure 9: Cumulative Distribution Function (CDF) plots for 1MB GET transactions from four locations on the Internet and from EC2.

Effective Bandwidth varies heavily based on (network) geography!

100 MB Get Ops from EC2 to S3

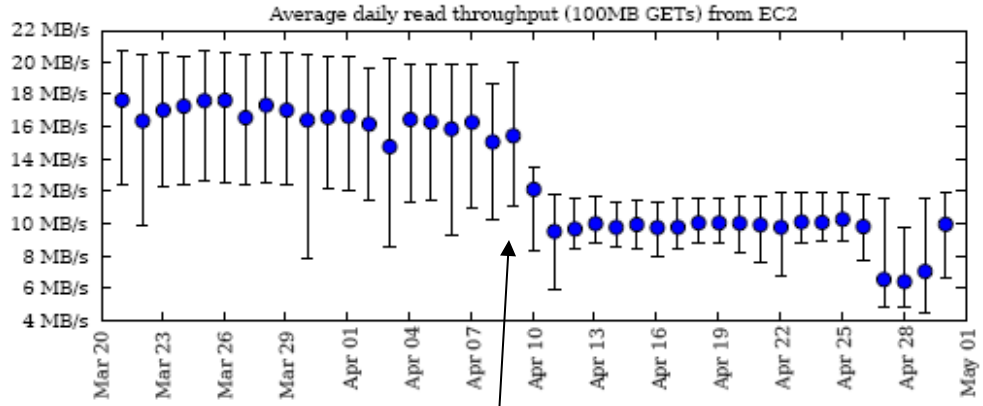


Figure 1: Average daily throughput as measured by 100MB GET operations from EC2. Error bars show the 5th and 95th percentile for each day's throughput measurement.

Throughput is relatively stable, except when internal network was reconfigured.

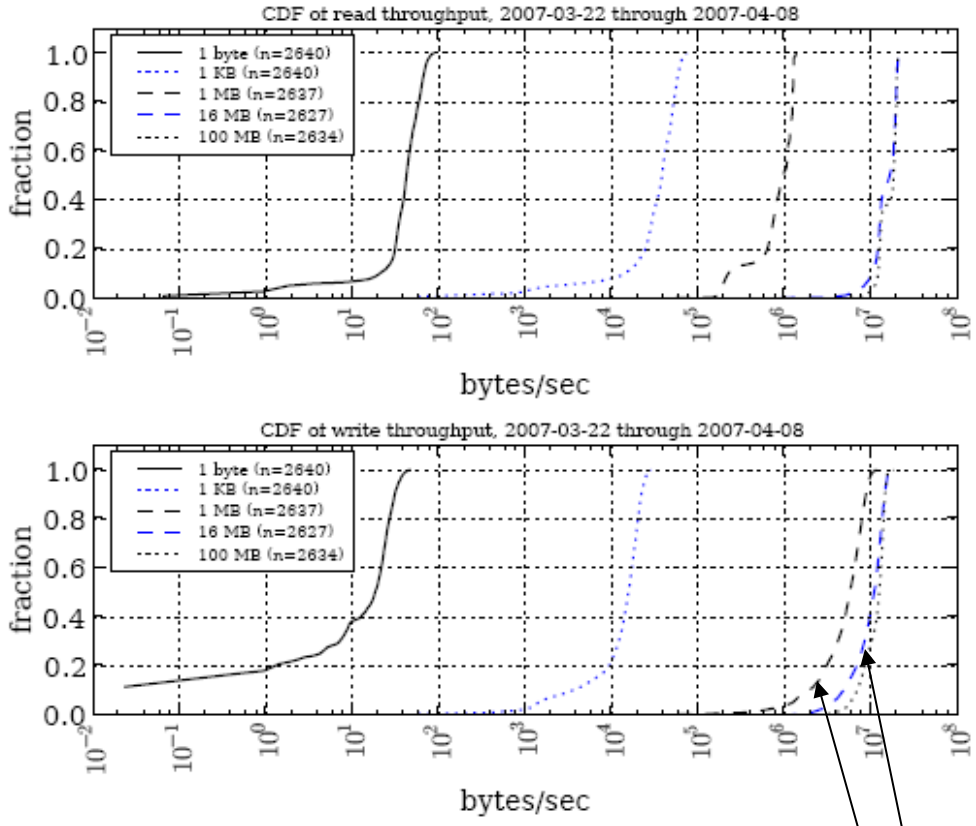


Figure 4: Cumulative Distribution Function (CDF) plots for transactions from EC2 to S3 for transactions of various sizes.

Read and Write throughputs: larger block size is better (but beyond some block size, it makes little difference).

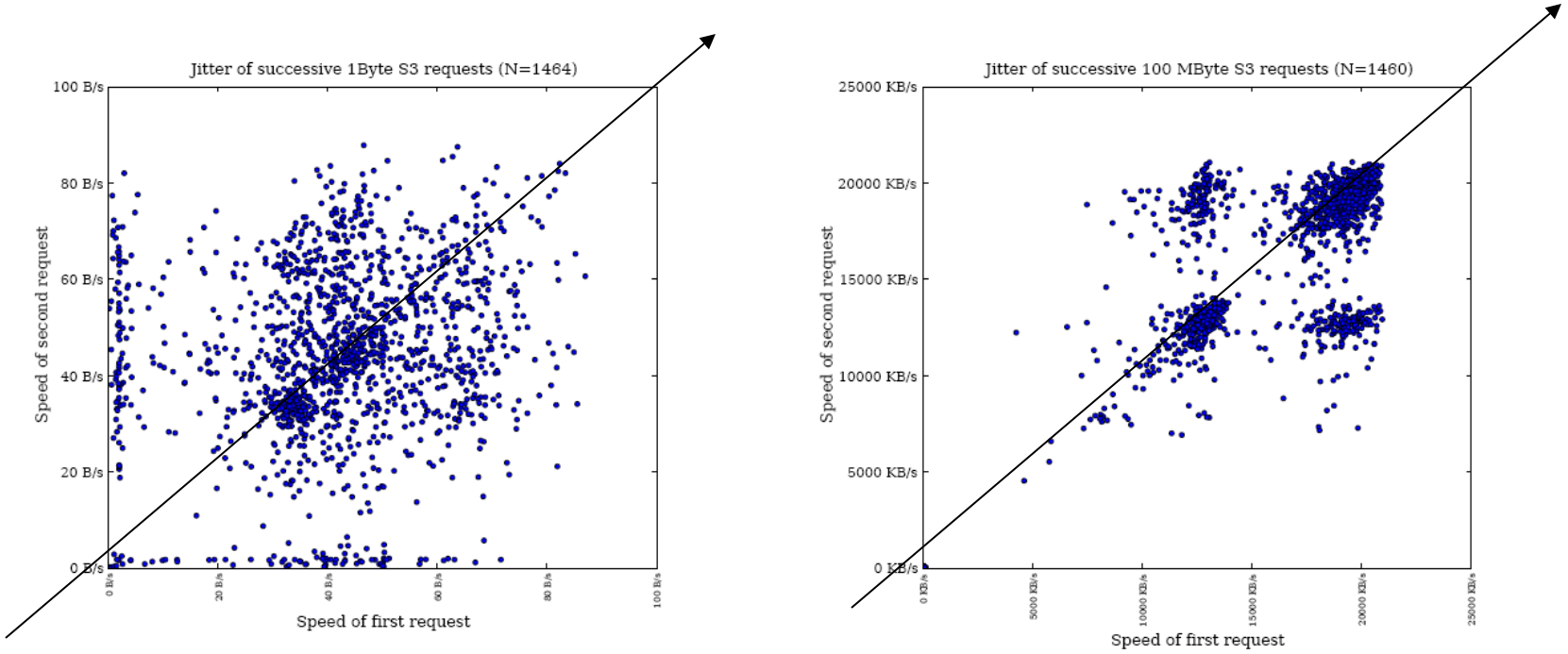


Figure 7: Scatter plots of bandwidth successive S3 GET requests for 1 Byte (left) and 100 Megabyte (right) transactions. The X axis indicates the speed of the first request, while the Y axis indicates the speed of the second.

Concurrency: Consecutive requests receive performance that are highly correlated, especially for large requests

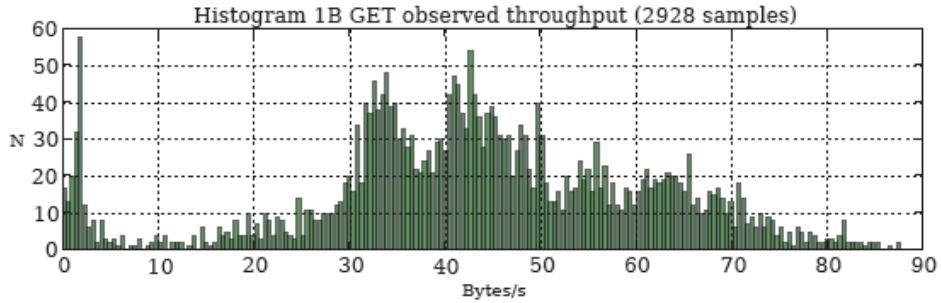


Figure 5: Histogram of 1 byte GET throughput, March 20 through April 7.

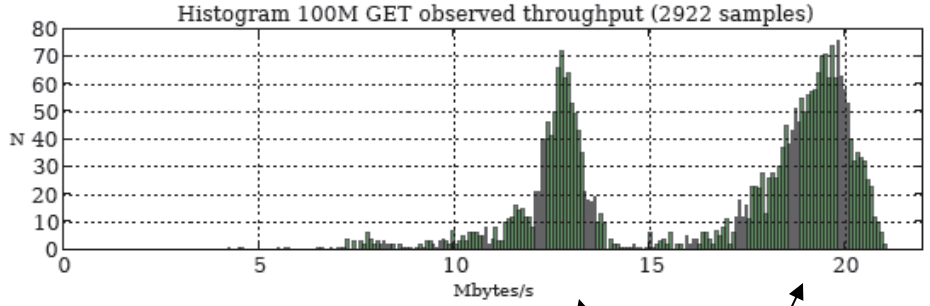


Figure 6: Histogram of 100Mbyte GET throughput, March 20 through April 7.

QoS received by requests fall into multiple “classes”
- 100 MB xfers fall into 2 classes.

Results Summary

1. Effective Bandwidth varies heavily based on geography!
2. Throughput is relatively stable, except when internal network was reconfigured.
3. Read and Write throughputs: larger requests are better, but throughput plateaus with file size
   - Decreases overhead
4. Consecutive requests receive performance that are highly correlated
5. QoS received by requests fall into multiple "classes", but need to give clients more control (e.g., via SLAs = Service Level Agreements)

What Do Real-Life Hadoop Workloads Look Like? (Cloudera)

(Slide Acknowledgments: Brian Cho)

What They Did

- Hadoop workloads from 5 Cloudera customers
  - Diverse industries: "in e-commerce, telecommunications, media, and retail"
  - 2011
- Hadoop workloads from Facebook
  - 2009, 2010 across same cluster

The Workloads

Trace	Machines	Length	Date	Jobs	Bytes moved	TB/ Day	Jobs/ Day	GB/ Job
CC-a	<100	1 month	2011	5759	80 TB	3	190	14
CC-b	300	9 days	2011	22974	600 TB	67	2550	26
CC-c	700	1 month	2011	21030	18 PB	600	700	856
CC-d	400-500	2+ months	2011	13283	8 PB	133	220	602
CC-e	100	9 days	2011	10790	590 TB	66	1200	55
FB-2009	600	6 months	2009	1129193	9.4 PB	52	6270	8
FB-2010	3000	1.5 months	2010	1169184	1.5 EB	33333	25980	1283
Total	>5000	≈ 1 year	-	2372213	1.6 EB			

Data access patterns (1/2)

- Skew in access frequency across (HDFS) files

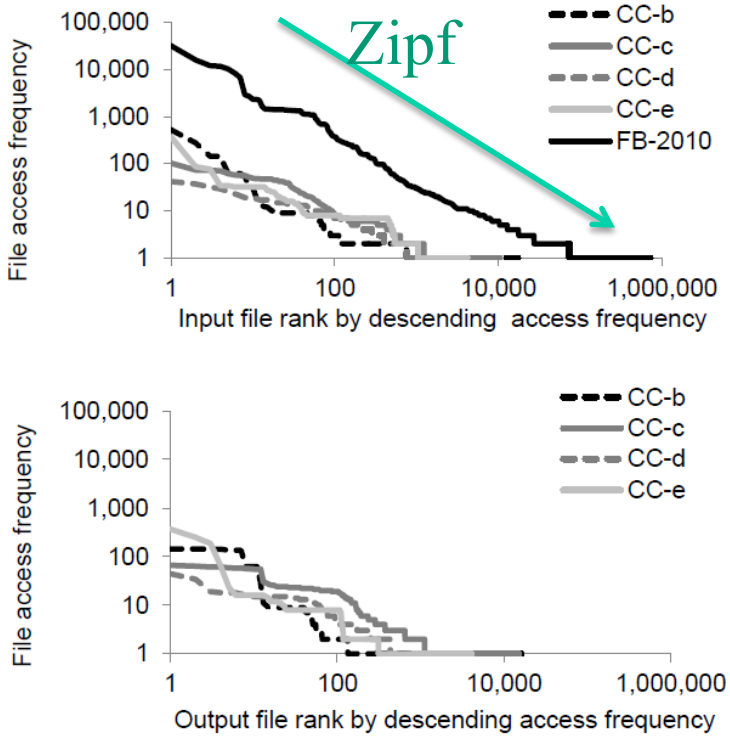


Figure 2: Log-log file access frequency vs. rank. Showing Zipf distribution of same shape (slope) for all workloads.

- 90% of jobs access files of less than a few GBs; such files account for only 16% of bytes stored

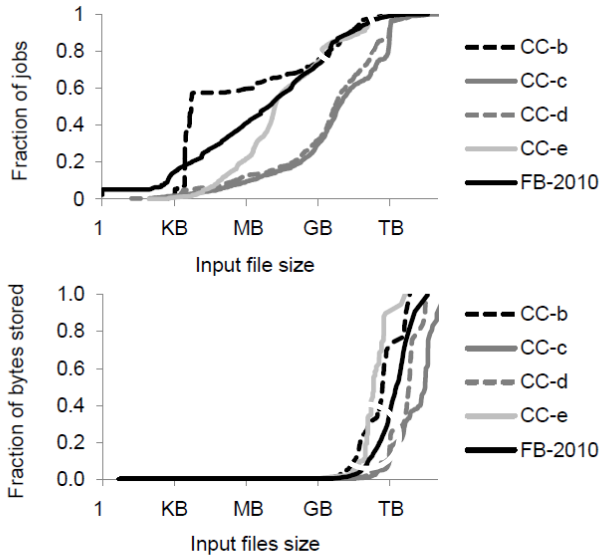


Figure 3: Access patterns vs. input file size. Showing cummulative fraction of jobs with input files of a certain size (top) and cummulative fraction of all stored bytes from input files of a certain size (bottom).

Data access patterns (2/2)

- Temporal Locality in data accesses

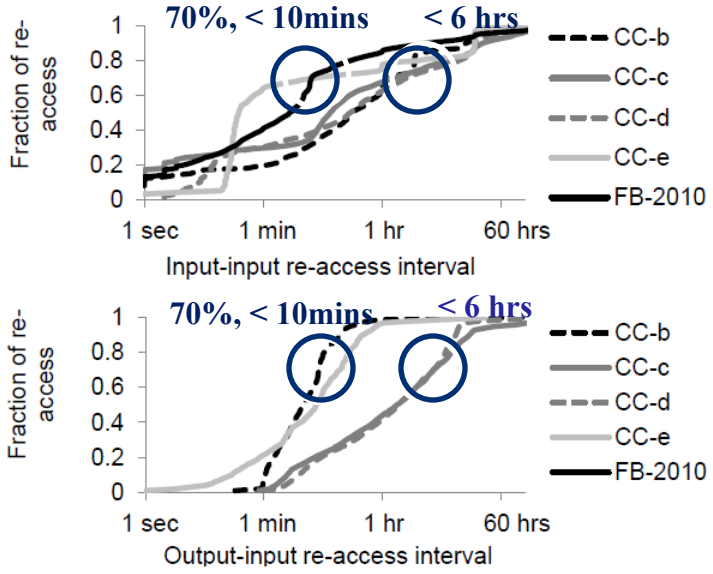


Figure 5: Data re-accesses intervals. Showing interval between when an input file is re-read (top), and when an output is re-used as the input for another job (bottom).

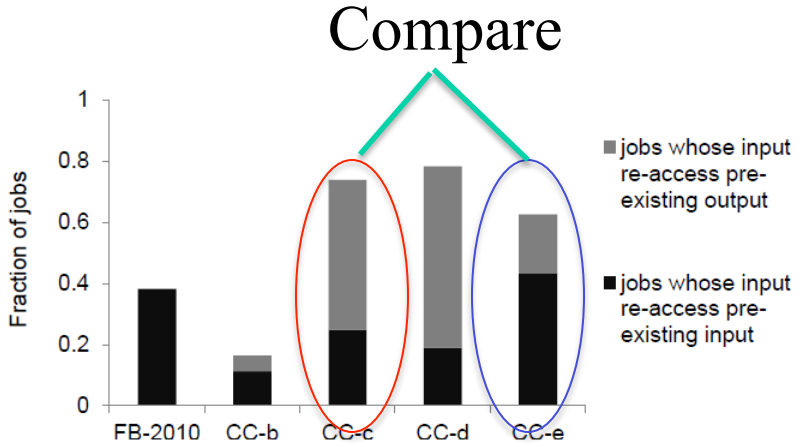


Figure 6: Fraction of jobs that reads pre-existing input path. Note that output path information is missing from FB-2010.

Can you make Hadoop/HDFS better, now that you know these characteristics?

Burstiness

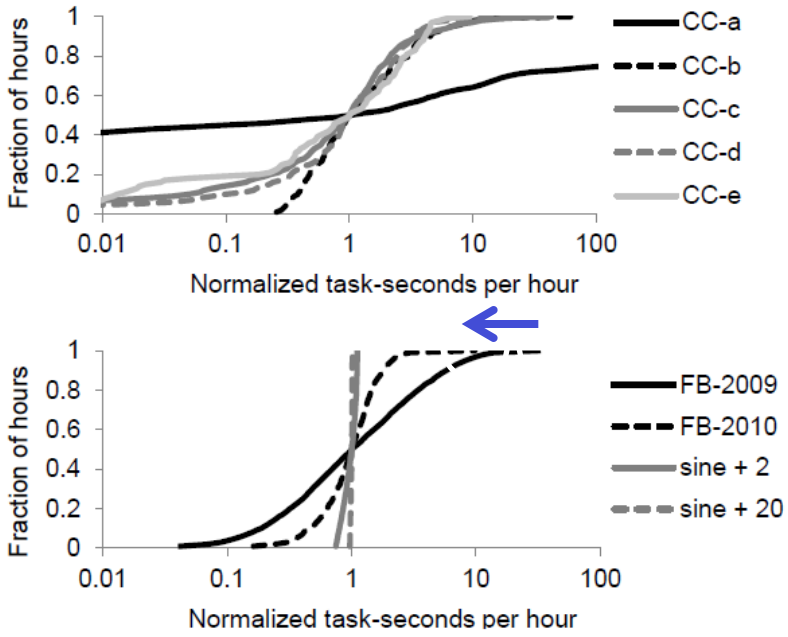


Figure 8: Workload burstiness. Showing cummulative distribution of task-time (sum of map time and reduce time) per hour. To allow comparison between workloads, all values have been normalized by the median task-time per hour for each workload. For comparison, we also show burstiness for artificial sine submit patterns, scaled with min-max range the same as mean (sine + 2) and 10% of mean (sine + 20).

- Plotted
  - Sum of task-time (map + reduce) over an hour interval
  - n-th percentile / median
- Facebook
  - From 2009 to 2010, peak-to-median ratio dropped from 31:1 to 9:1
  - Claim: consolidation decreases burstiness

High-level Processing Frameworks

Each cluster prefers 1-2 data processing frameworks

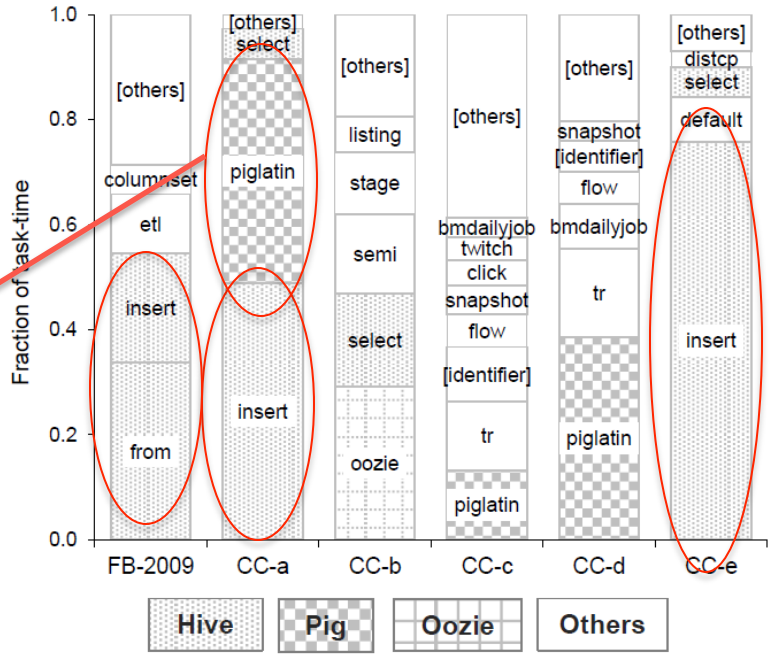


Figure 10: The first word of job names for each workload, weighted by the number of jobs beginning with each word (top), total I/O in bytes (middle), and map/reduce task-time (bottom). For example, 44% of jobs in the FB-2009 workload have a name beginning with "ad", a further 12% begin with "insert"; 27% of all I/O and 34% of total task-time comes from jobs with names that begin with "from" (middle and bottom). The FB-2010 trace did not contain job names.

Classification by multi-dimensional clustering

	# Jobs	Input	Shuffle	Output	Duration	Map time	Reduce time	Label
CC-e	10243	8.1 MB	0	970 KB	18 sec	15	0	Small jobs
	452	166 GB	180 GB	118 GB	31 min	35,606	38,194	Transform, large
	68	543 GB	502 GB	166 GB	2 hrs	115,077	108,745	Transform, very large
	20	3.0 TB	0	200 B	5 min	137,077	0	Map only summary
	7	6.7 TB	2.3 GB	6.7 TB	3 hrs 47 min	335,807	0	Map only transform
FB-2009	1081918	21 KB	0	871 KB	32 s	20	0	Small jobs
	37038	381 KB	0	1.9 GB	21 min	6,079	0	Load data, fast
	2070	10 KB	0	4.2 GB	1 hr 50 min	26,321	0	Load data, slow
	602	405 KB	0	447 GB	1 hr 10 min	66,657	0	Load data, large
	180	446 KB	0	1.1 TB	5 hrs 5 min	125,662	0	Load data, huge
	6035	230 GB	8.8 GB	491 MB	15 min	104,338	66,760	Aggregate, fast
	379	1.9 TB	502 MB	2.6 GB	30 min	348,942	76,736	Aggregate and expand
	159	418 GB	2.5 TB	45 GB	1 hr 25 min	1,076,089	974,395	Expand and aggregate
	793	255 GB	788 GB	1.6 GB	35 min	384,562	338,050	Data transform
	19	7.6 TB	51 GB	104 KB	55 min	4,843,452	853,911	Data summary
FB-2010	1145663	6.9 MB	600 B	60 KB	1 min	48	34	Small jobs
	7911	50 GB	0	61 GB	8 hrs	60,664	0	Map only transform, 8 hrs
	779	3.6 TB	0	4.4 TB	45 min	3,081,710	0	Map only transform, 45 min
	670	2.1 TB	0	2.7 GB	1 hr 20 min	9,457,592	0	Map only aggregate
	104	35 GB	0	3.5 GB	3 days	198,436	0	Map only transform, 3 days
	11491	1.5 TB	30 GB	2.2 GB	30 min	1,112,765	387,191	Aggregate
	1876	711 GB	2.6 TB	860 GB	2 hrs	1,618,792	2,056,439	Transform, 2 hrs
	454	9.0 TB	1.5 TB	1.2 TB	1 hr	1,795,682	818,344	Aggregate and transform
	169	2.7 TB	12 TB	260 GB	2 hrs 7 min	2,862,726	3,091,678	Expand and aggregate
	67	630 GB	1.2 TB	140 GB	18 hrs	1,545,220	18,144,174	Transform, 18 hrs

Results Summary

- Workloads different across industries
- Yet commonalities
  - Zipf distribution for access file access frequency
  - Slope same across all industries
- 90% of all jobs access small files, while the other 10% account for 84% of the file accesses
  - Parallels p2p systems (mp3-mpeg split)
- A few frameworks popular for each cluster
- Lots of small jobs

Summary

- We design algorithms, implement and deploy them
- But when you factor in the real world, unexpected characteristics may arise
- Important to understand these characteristics to build better distributed systems for the real world

Optional Slides - Understanding Availability (P2P Systems)

R. Bhagwan, S. Savage, G. Voelker
University of California, San Diego

What They Did

- Measurement study of peer-to-peer (P2P) file sharing application
  - Overnet (January 2003)
  - Based on Kademlia, a DHT based on xor routing metric
    - Each node uses a random self-generated ID
    - The ID remains constant (unlike IP address)
    - Used to collect availability traces
  - Closed-source
- Analyze collected data to analyze availability
- Availability = % of time a node is online (node=user, or machine)

What They Did

- Crawler:
  - Takes a snapshot of all the active hosts by repeatedly requesting 50 randomly generated IDs.
  - The requests lead to discovery of some hosts (through routing requests), which are sent the same 50 IDs, and the process is repeated.
  - Run once every 4 hours to minimize impact
- Prober:
  - Probe the list of available IDs to check for availability
    - By sending a request to ID *I*; request succeeds only if *I* replies
    - Does not use TCP, avoids problems with NAT and DHCP
  - Used on only randomly selected 2400 hosts from the initial list
  - Run every 20 minutes

Scale of Data

- Ran for 15 days from January 14 to January 28 (with problems on January 21) 2003
- Each pass of crawler yielded 40,000 hosts.
- In a single day (6 crawls) yielded between 70,000 and 90,000 unique hosts.
- 1468 of the 2400 randomly selected hosts probes responded at least once

Multiple IP Hosts

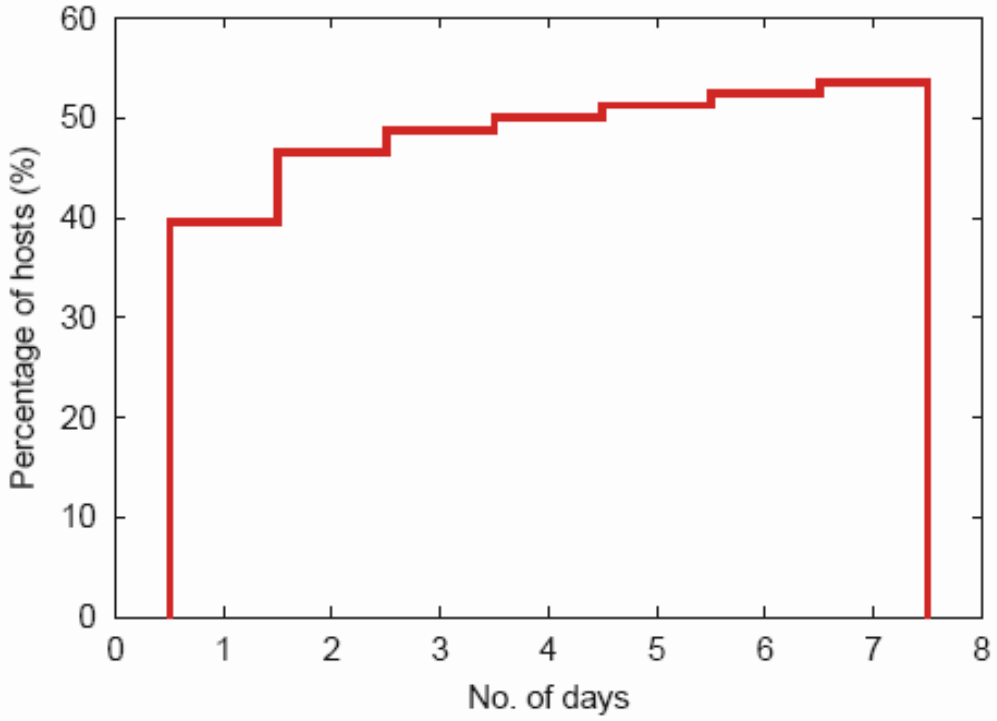


Figure 1: Percentage of hosts that have more than one IP address across different periods of time.

Availability

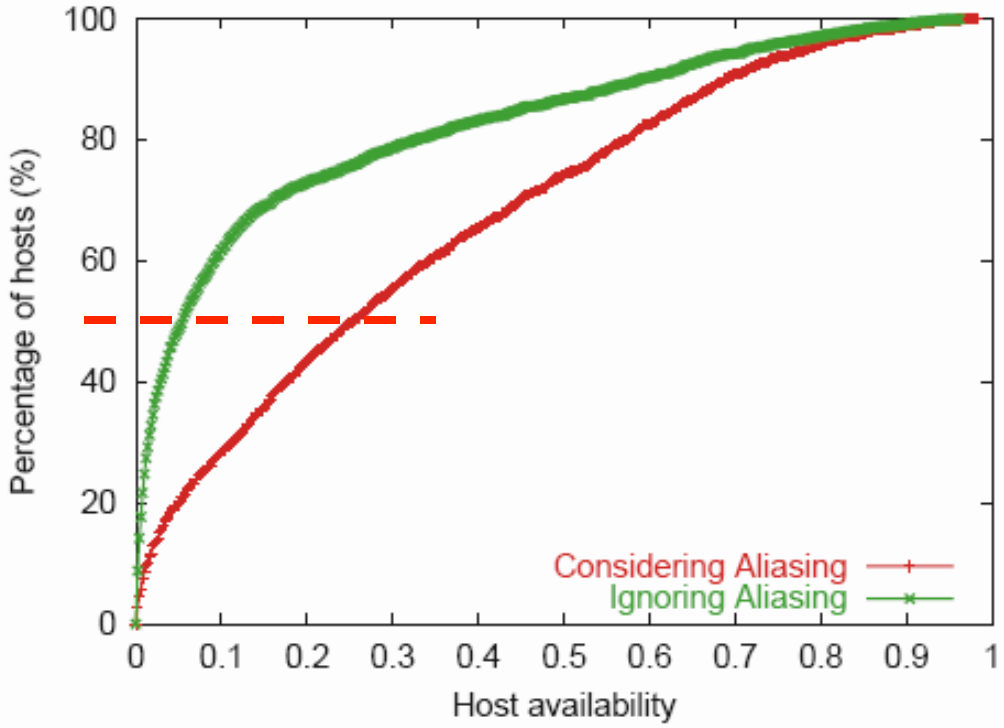


Figure 2: Host availability derived using unique host ID probes vs. IP address probes.

Results Summary

1. Overall availability is low
2. Diurnal patterns existing in availability
3. Availabilities are uncorrelated across nodes
4. High Churn exists