

AVAILABILITY AND PERFORMANCE

Module 5
Fall 2020

George Porter



ATTRIBUTION

- These slides are released under an Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) Creative Commons license
- These slides incorporate material from:
 - Jeffrey Dean and Luiz André Barroso. The tail at scale.

REQUIRED READING FOR THIS LECTURE



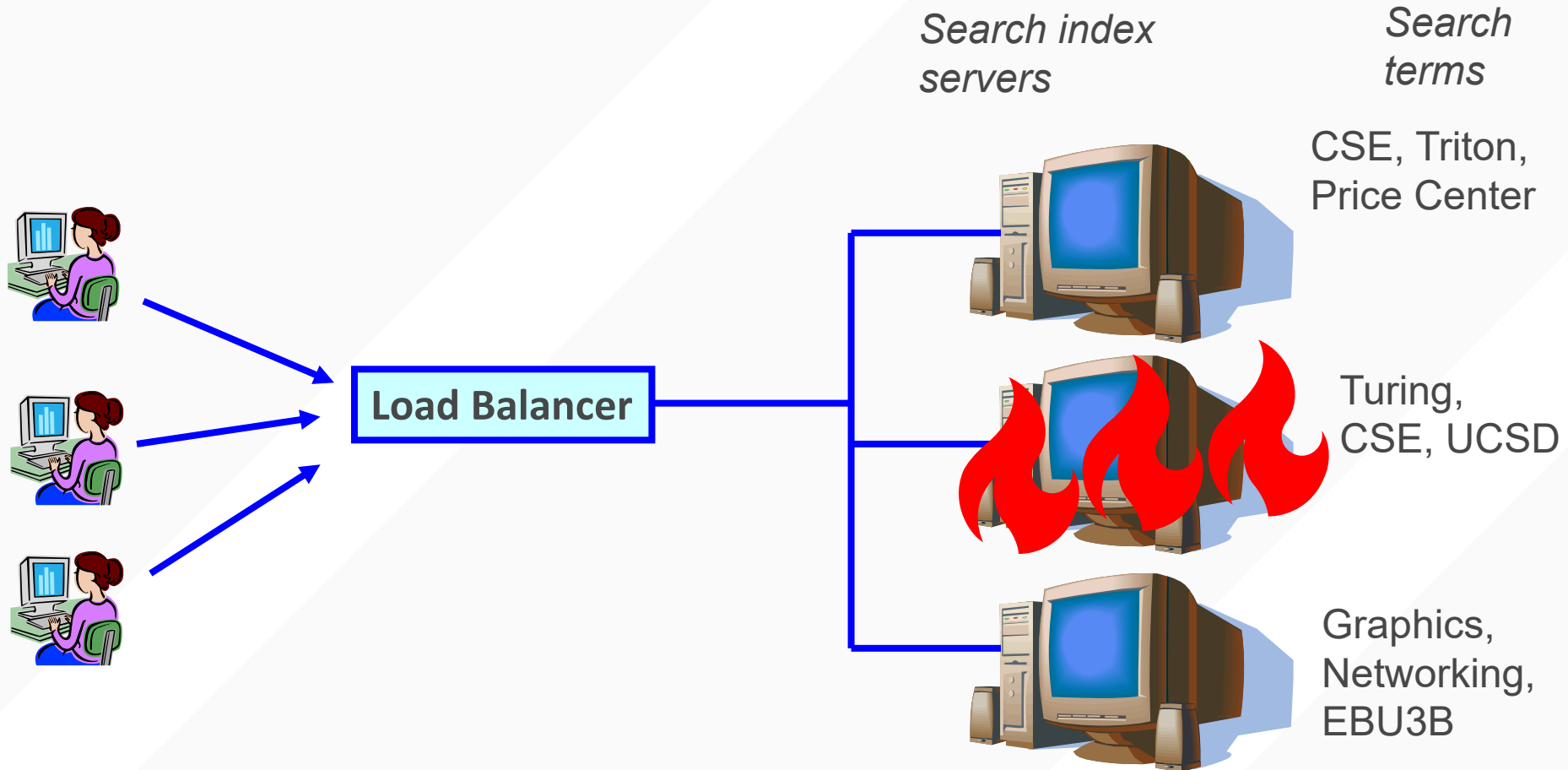
Head of Google ai; (Co-)designed Google's Ad engine, Web crawler, indexer, and query serving system. Created Spanner, BigTable, MapReduce, LevelDB, TensorFlow (AI/ML system), ...

Google Fellow, VP of Engineering, Technical lead of Google's infrastructure and datacenters



Jeffrey Dean and Luiz André Barroso. The tail at scale. Communication of the ACM 56, 2 (February 2013), 74-80. DOI: <https://doi.org/10.1145/2408776.2408794>

AVAILABILITY



AVAILABILITY METRICS

- Mean time between failures (MTBF)
- Mean time to repair (MTTR)
- $\text{Availability} = (\text{MTBF} - \text{MTTR}) / \text{MTBF}$
- Example:
 - MTBF = 10 minutes
 - MTTR = 1 minute
 - $A = (10 - 1) / 10 = 90\%$ availability
- Can improve availability by increasing MTBF or by reducing MTTR
 - Ideally, systems never fail but much easier to test reduction in MTTR than improvement in MTBF

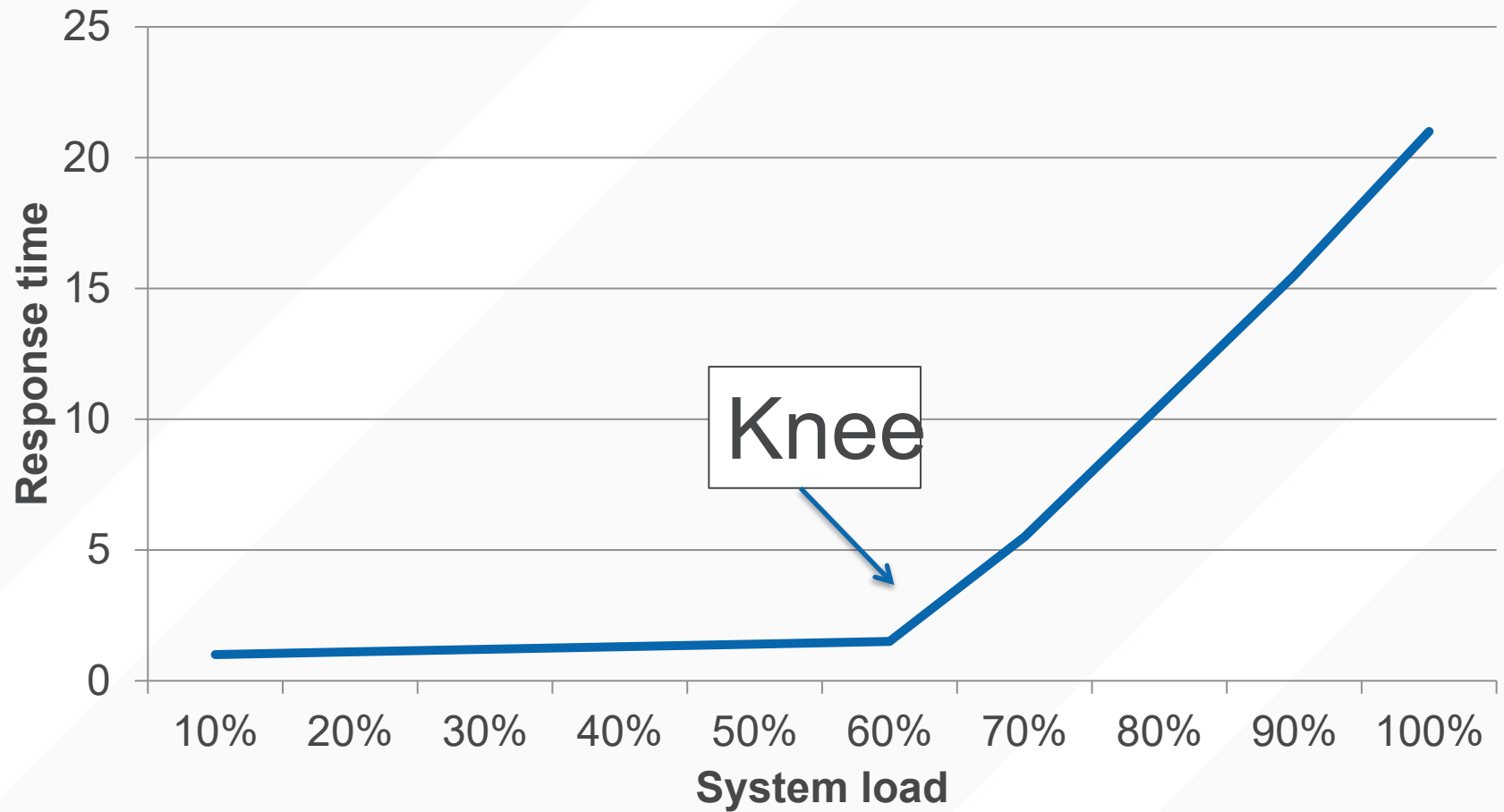
HARVEST AND YIELD

- *yield* = *queries completed/queries offered*
 - In some sense more interesting than availability because it focuses on client perceptions rather than server perceptions
 - If a service fails when no one was accessing it...
- *harvest* = *data available/complete data*
 - How much of the database is reflected in each query?
- Should faults affect yield, harvest or both?

DQ PRINCIPLE

- *Data per query * queries per second → constant*
- At high levels of utilization, can increase queries per second by reducing the amount of input for each response
- Adding nodes or software optimizations changes the constant

PERFORMANCE “HOCKEY STICK” GRAPH



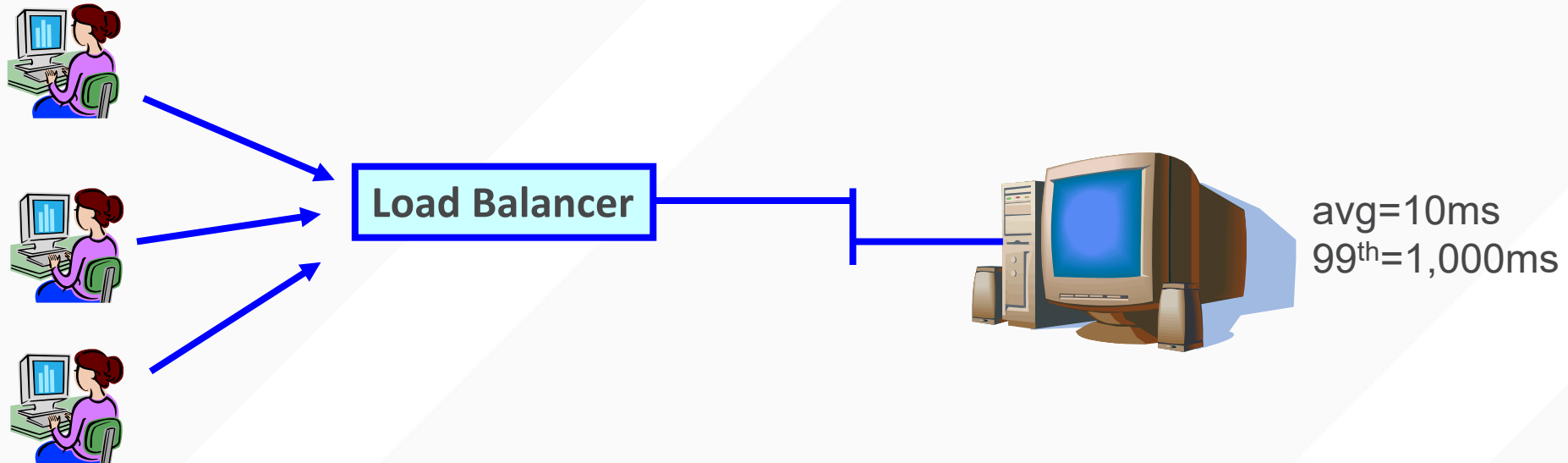
EFFECT OF LATENCY VARIATION

service to feel responsive.

Variability in the latency distribution of individual components is magnified at the service level; for example, consider a system where each server typically responds in 10ms but with a 99th-percentile latency of one second. If a user request is handled on just one such server, one user request in 100 will be slow (one second). The figure here outlines how service-level latency in this hypothetical scenario is affected by very

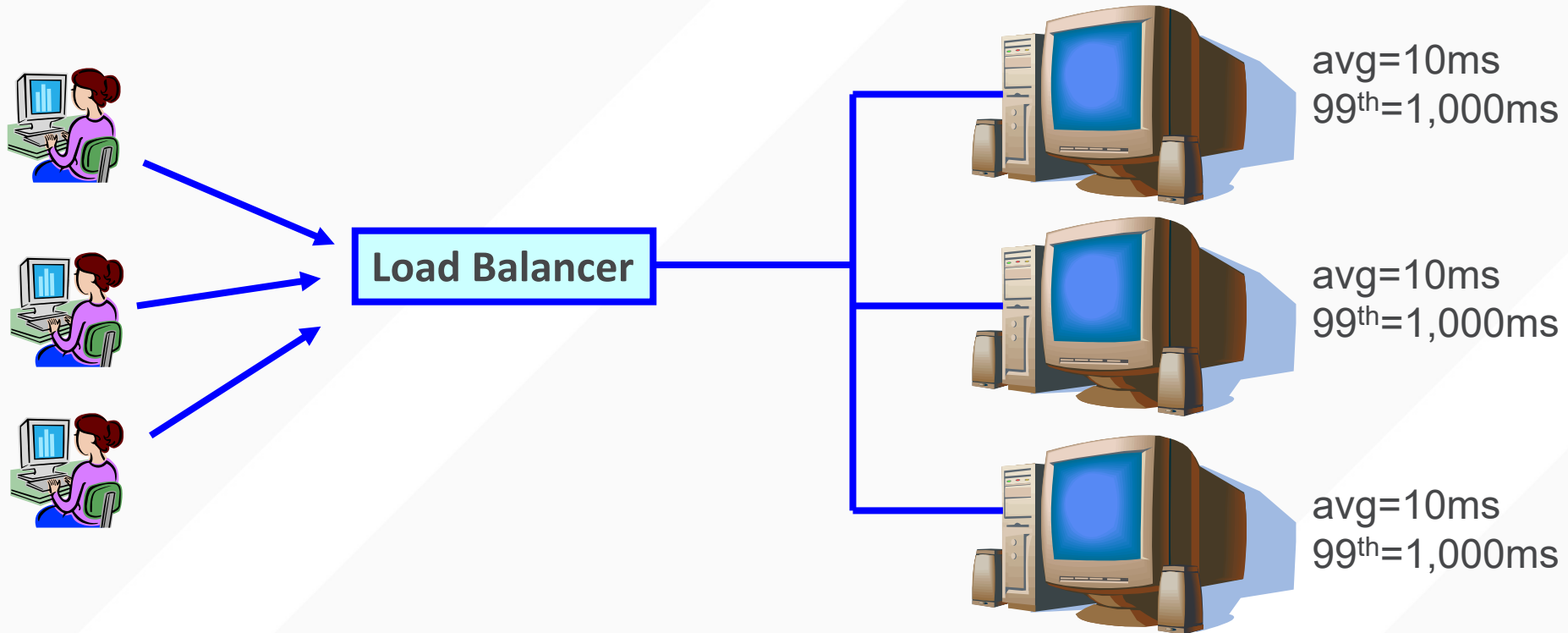
higher-level queuing. Different service classes can be used for prioritizing requests for which different policies take effect more prominently. For example, the storage server's cluster-level file-system operations may take a few operations outstarvation of the operating system's disk scheduling policy, maintaining their own queue of pending disk requests.

PERFORMANCE NOT AT SCALE



- What is the expected time to service one request to one server?
 - 10ms? more? less?

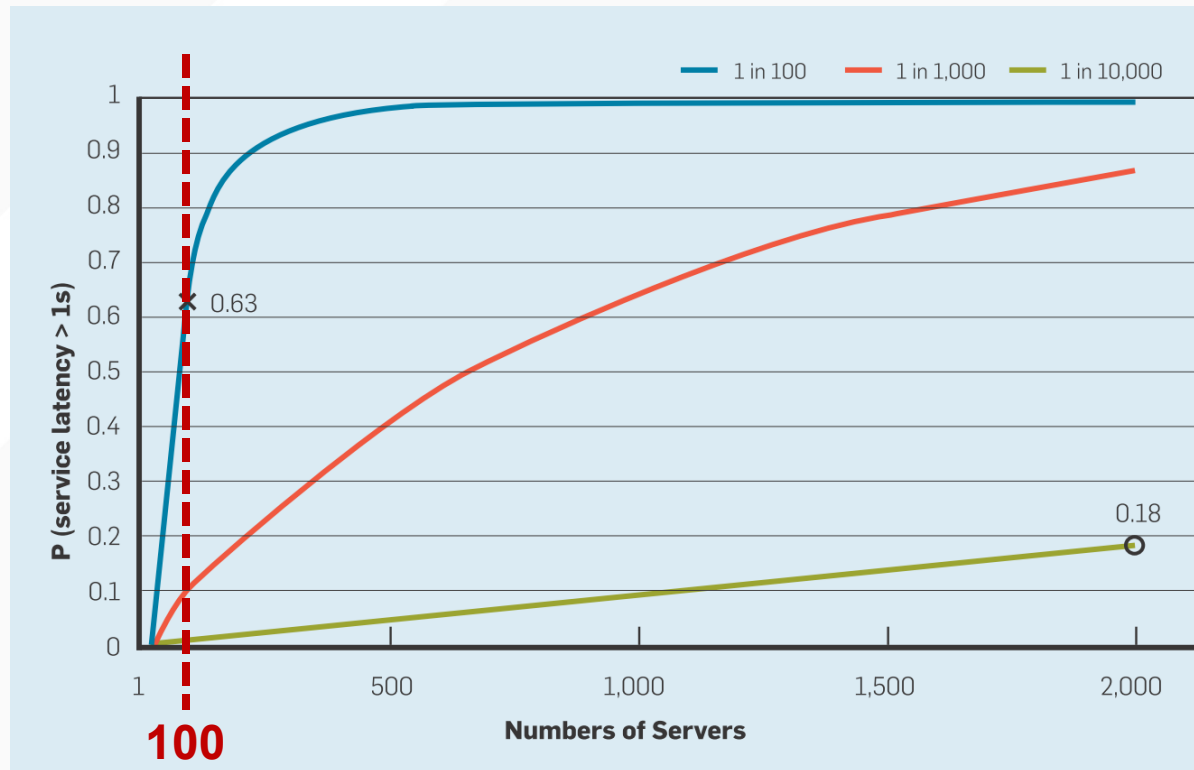
PERFORMANCE AT SCALE



- What is the expected time to service three correlated requests to three servers?
 - Must wait until all complete before the load balancer can return a result to the user
 - 10ms? more? less?

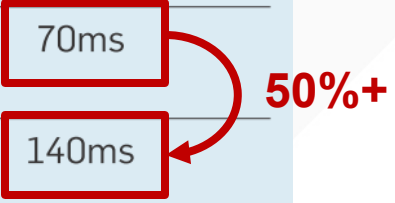
COMPONENT VARIABILITY AMPLIFIED BY SCALE

- Latency variability is magnified at the service level.



REQUEST LATENCY MEASUREMENT

	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



- Key Observation:
 - 5% servers contribute nearly 50% latency.

KEY QUESTION

- What would you do with that 5% servers which contribute 50% latency?

~~Eliminate~~ all interactive request latencies.

Live with

But it's infeasible to eliminate all variations!

FACTORS OF VARIABLE RESPONSE TIME

- Shared Resources (Local)
 - CPU cores
 - Processors caches
 - Memory bandwidth
- Global Resource Sharing
 - Network switches
 - Shared file systems
- Daemons
 - Scheduled Procedures



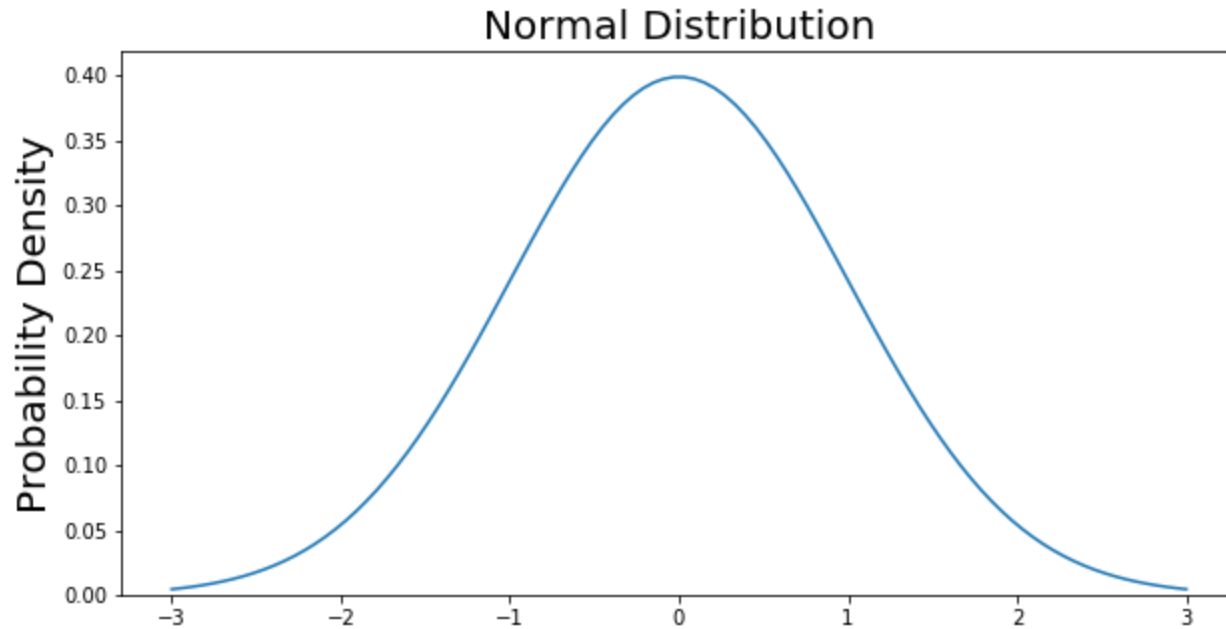
FACTORS OF VARIABLE RESPONSE TIME

- Maintenance Activities
 - Data reconstruction in distributed file systems
 - Periodic log compactions in storage systems
 - Periodic garbage collection in garbage-collected languages
- Queueing
 - Queueing in intermediate servers and network switches

FACTORS OF VARIABLE RESPONSE TIME

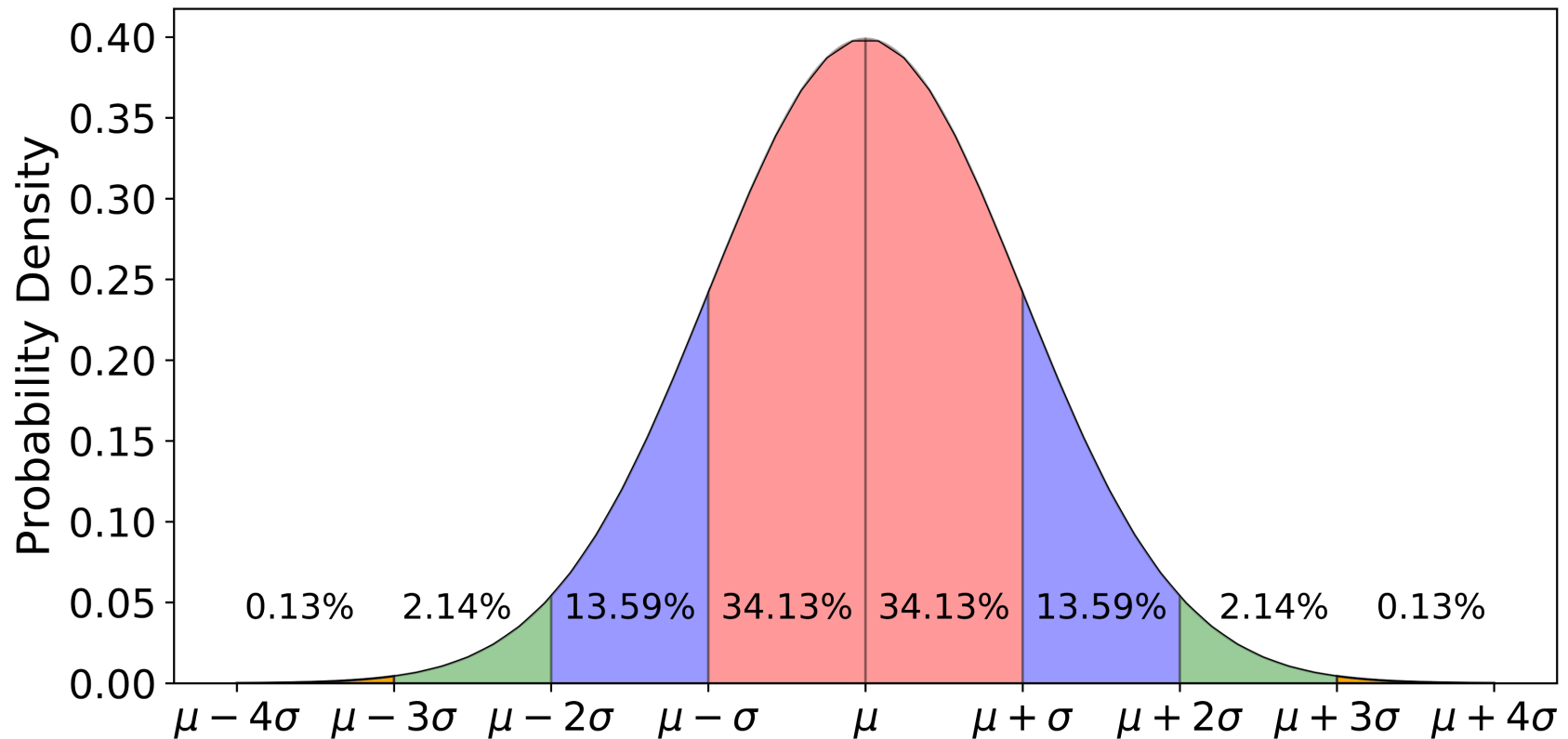
- Power Limits
 - Throttling due to thermal effects on CPUs
- Garbage Collection
 - Random access in solid-state storage devices
- Energy Management
 - Power saving modes
 - Switching from inactive to active modes

RANDOM VARIABLES: NORM(0,1)



RANDOM VARIABLES: $\text{NORM}(\mu, \sigma)$

Normal Distribution



EXPLORING NORMAL RANDOM VARIABLES WITH GOOGLE SHEETS

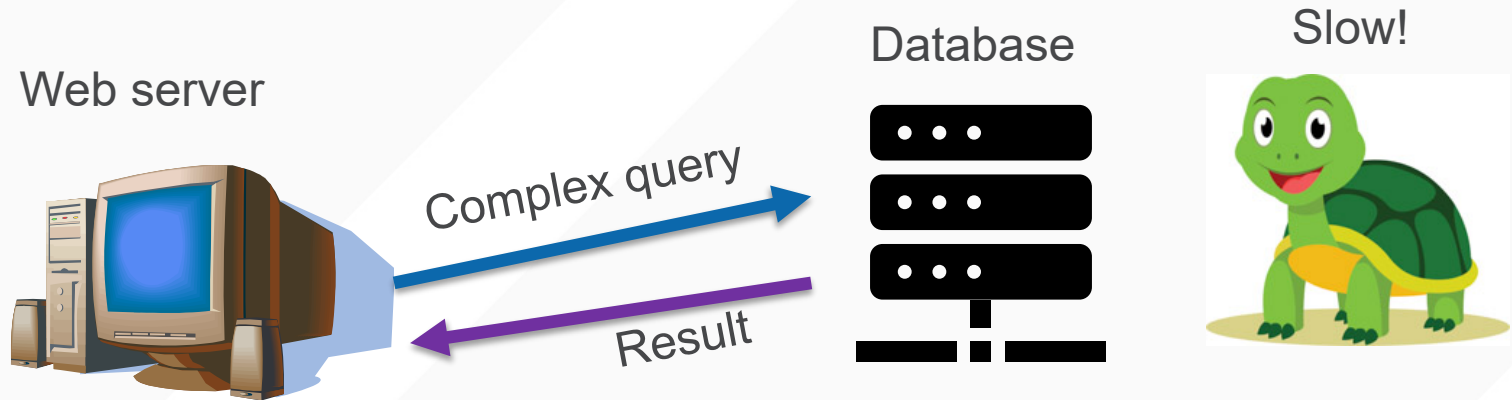
- You too can generate observations of a normal random variable by adding this to a google sheets (or excel, numbers, etc) document:
 - `=NORMINV (rand () , 0 , 1)`

CASE STUDY: MEMCACHED

- Popular in-memory cache
- Simple `get()` and `put()` interface
- Useful for caching popular or expensive requests

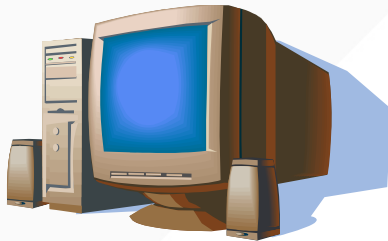


BASELINE: DATABASE-DRIVEN WEB QUERY

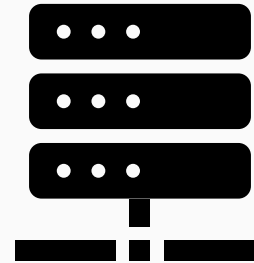


MEMCACHED EXAMPLE: CACHE HIT

Web server



Database



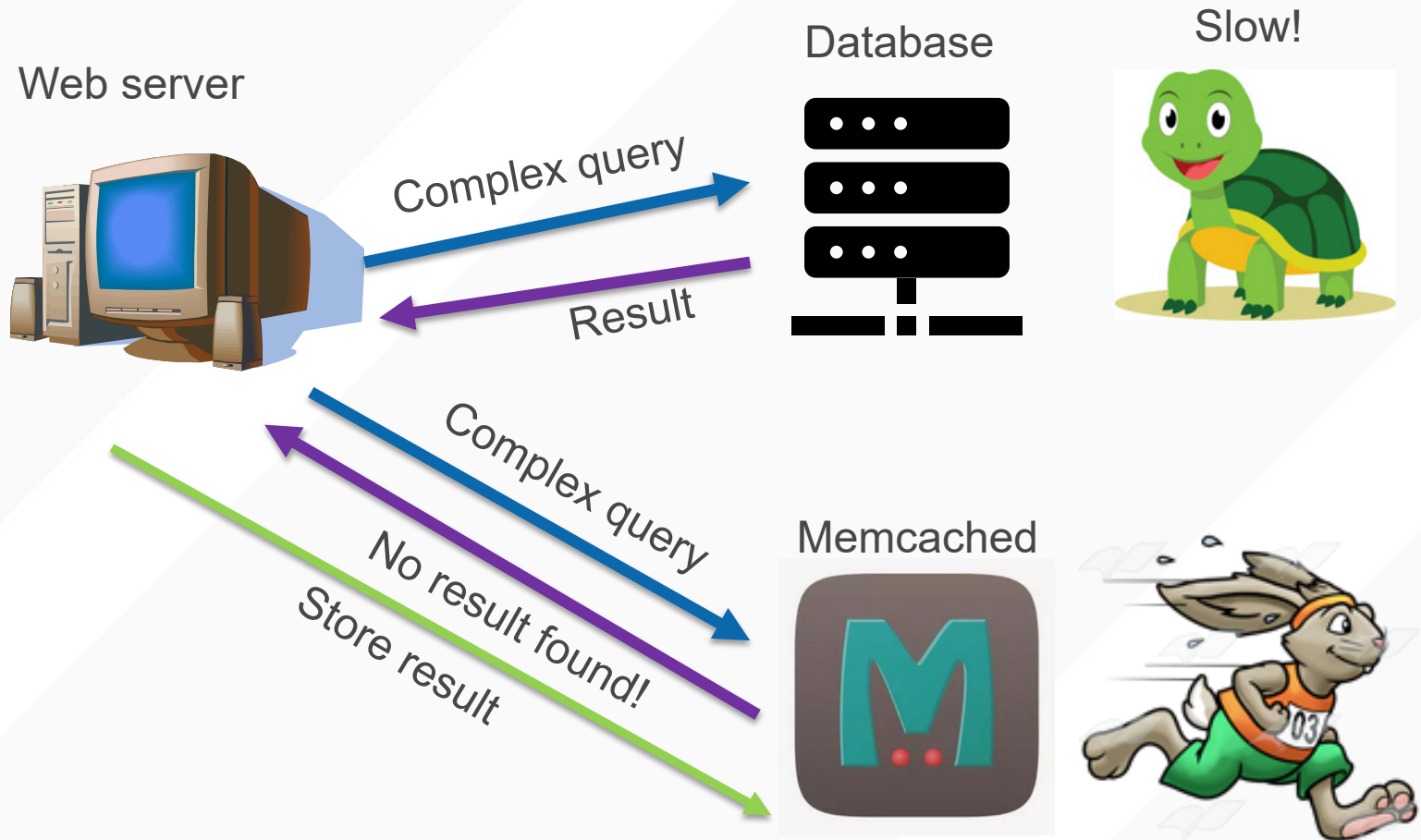
Memcached



Complex query

Result

MEMCACHED EXAMPLE: CACHE MISS



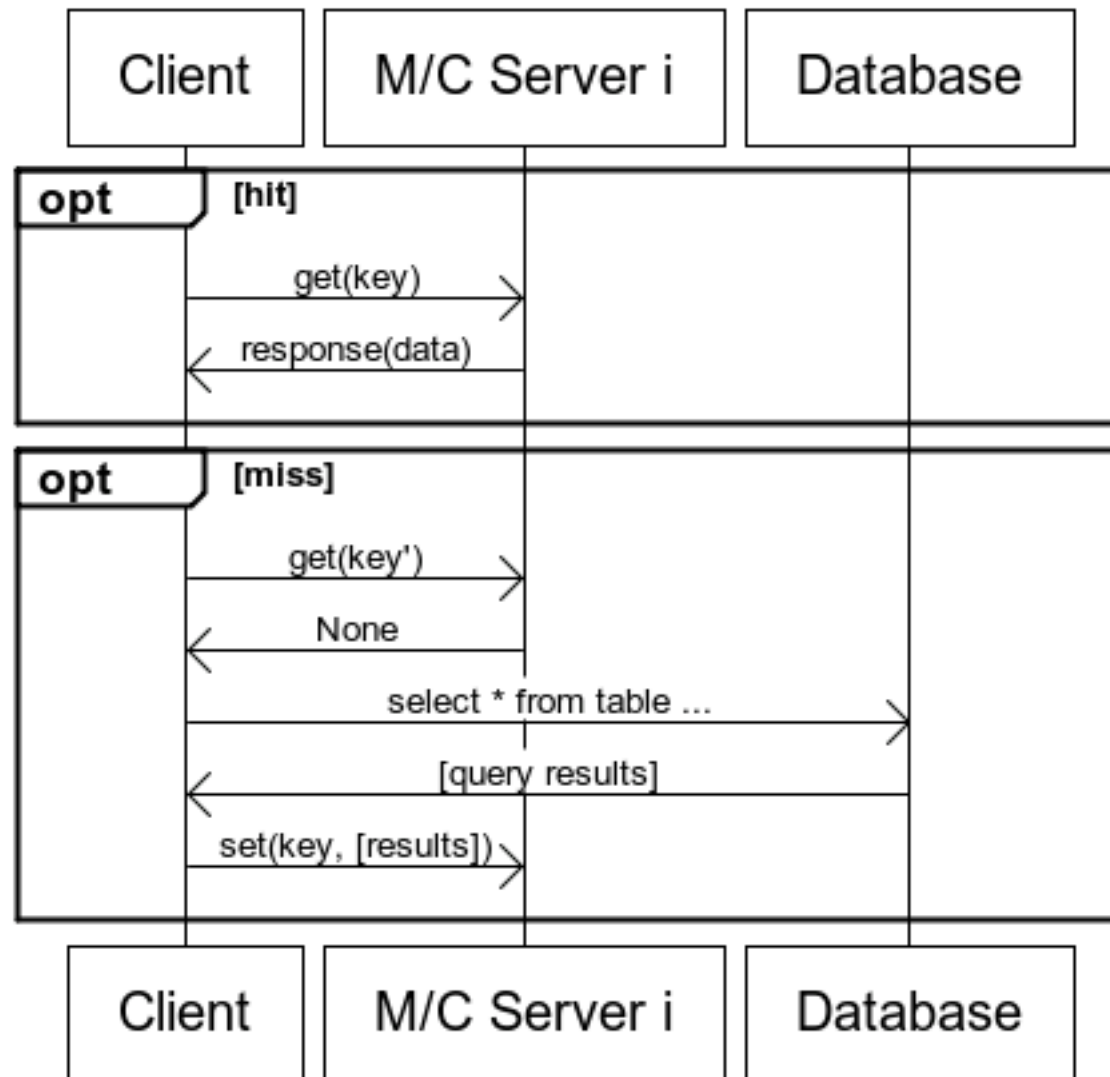
CASE STUDY: MEMCACHED

- Popular in-memory cache
- Simple get() and put() interface
- Useful for caching popular or expensive requests
- LRU replacement policy

```
function get_foo(foo_id)
  foo = memcached_get("foo:" . foo_id)
  return foo if defined foo

  foo = fetch_foo_from_database(foo_id)
  memcached_set("foo:" . foo_id, foo)
  return foo
end
```

MEMCACHED DATA FLOW



EXPERIMENT: GET/SET WITH MEMCACHED

```
from pymemcache.client import base

client = base.Client(('localhost', 11211))

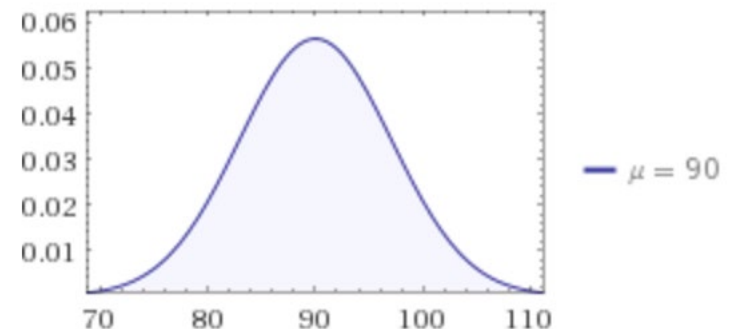
client.set('some_key', 'some value')

print(client.get('some_key'))
```

TAIL TOLERANCE: PARTITION/AGGREGATE

- Consider distributed memcached cluster
 - Single client issues request to S memcached servers
 - Waits until all S are returned
 - Service time of a memcached server is normal w/ $\mu = 90\mu s$, $\sigma = 7\mu s$
 - Roughly based on measurements from my former student

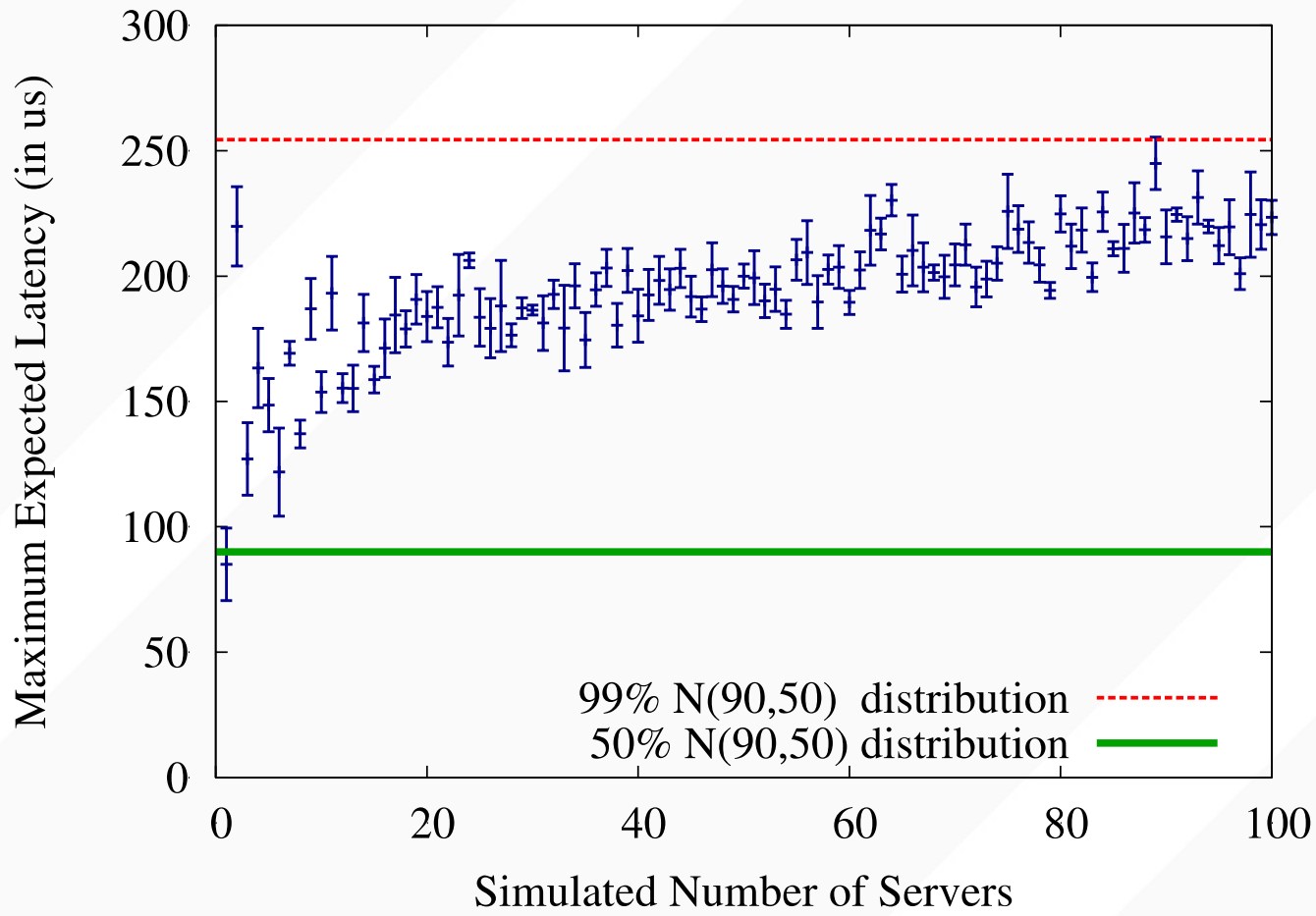
Plot of PDF:



EXPLORING NORMAL RANDOM VARIABLES WITH GOOGLE SHEETS

- You too can generate observations of a normal random variable by adding this to a google sheets (or excel, numbers, etc) document:
 - Based on Memcached:
 - `=NORMINV (rand () , 90 , 7)`

MATLAB SIMULATION



WITHIN REQUEST SHORT-TERM ADAPTATIONS

- Tied Requests
 - Hedged requests with cancellation mechanism.

	Mostly idle cluster			With concurrent terasort		
	No hedge	Tied request after 1ms		No hedge	Tied request after 1ms	
50%ile	19ms	16ms	(-16%)	24ms	19ms	(-21%)
90%ile	38ms	29ms	(-24%)	56ms	38ms	(-32%)
99%ile	67ms	42ms	(-37%)	108ms	67ms	(-38%)
99.9%ile	98ms	61ms	(-38%)	159ms	108ms	(-32%)

UC San Diego