Pragmatic Performance

@giltene

Gil Tene, CTO & co-Founder, Azul Systems

prag·mat·ic

/pragˈmadik/

adjective

dealing with things sensibly and realistically in a way that is based on *practical* rather than *theoretical* considerations.

Theoretical Performance

Question: How fast can this car go?

Theoretical Performance

Question: How fast can this car go?

Theoretical Performance answer: 189mph

Theoretical Performance

Faster?

Slower?

Pragmatic Performance

Faster?

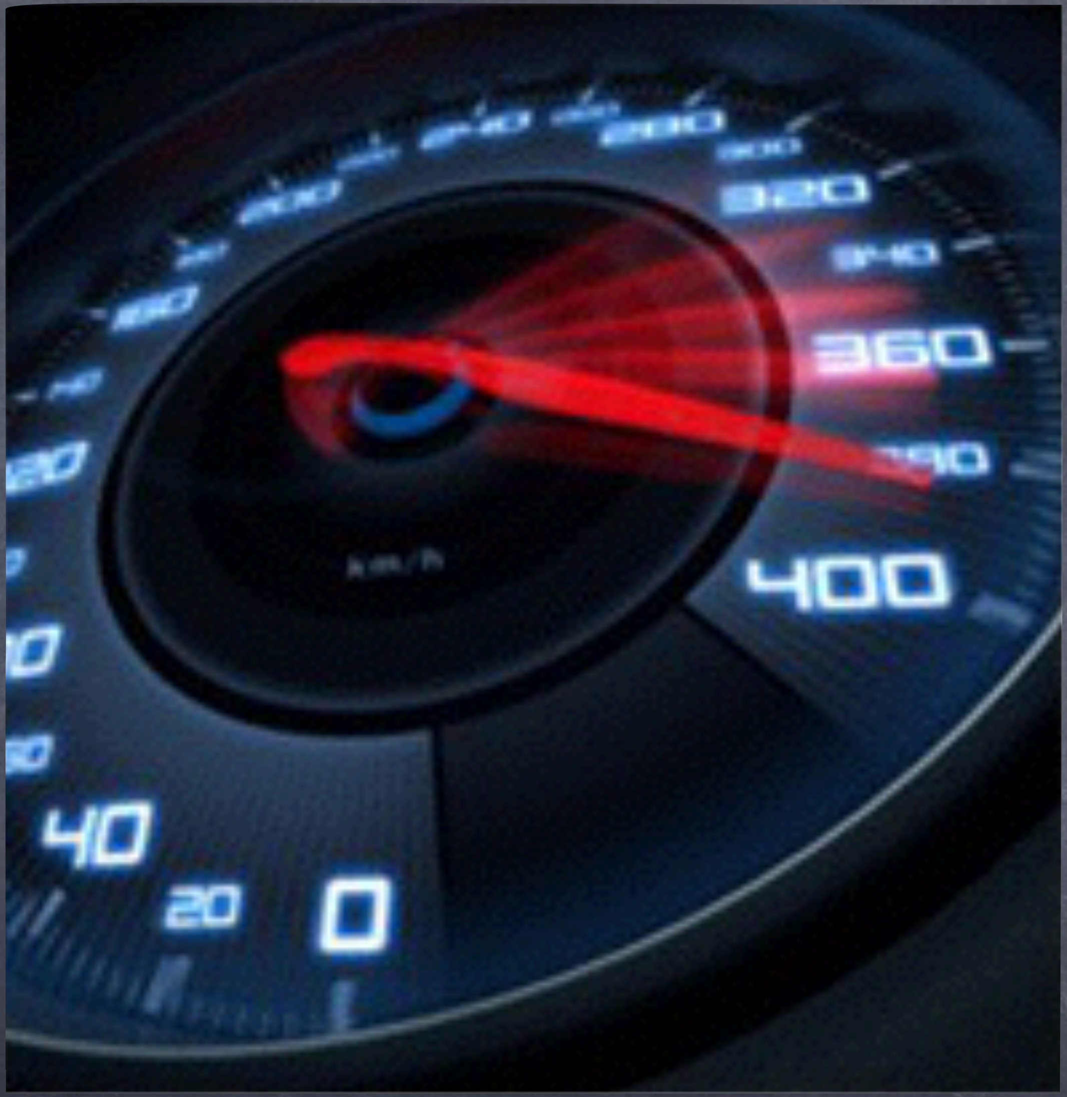
Slower?

How about now?

Pragmatic Performance

Pragmatic Question: How fast can this car go without crashing into things?

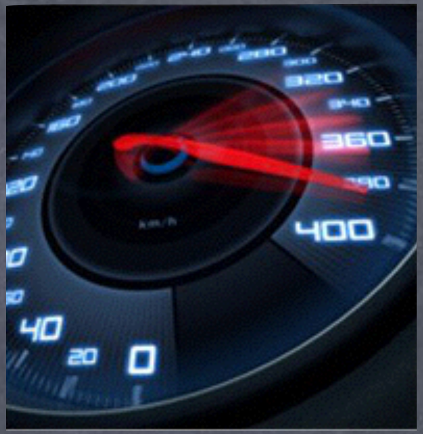
Theoretical Performance



How many queries per second can "cool tool X" answer?

Theoretical Performance

How many queries per second can “cool tool X” answer?



Examples of important questions to ask:

- Do all the answers have to be correct?
- Is it OK to only answer easy questions?
- Is it OK to take 1,000,000,000,000 questions now and answer them all next week?

Comparing Performance

- System A does X things per second. But fails some key requirements

- System B does 0.9x things per second, and meets all key requirements

- “System B is slower but more reliable” — WRONG

- How fast can system A go while meeting requirements?

Performance does not live in a vacuum

“Performance” is (usually) meaningful only when practical considerations, requirements and constraints are met

performance metric examples

- Operations per second
- Latency or Response Time
- Failure rate
- Recovery time (e.g. to “normal”) after disruption
- Each of these is best measured when all the others are held to required levels

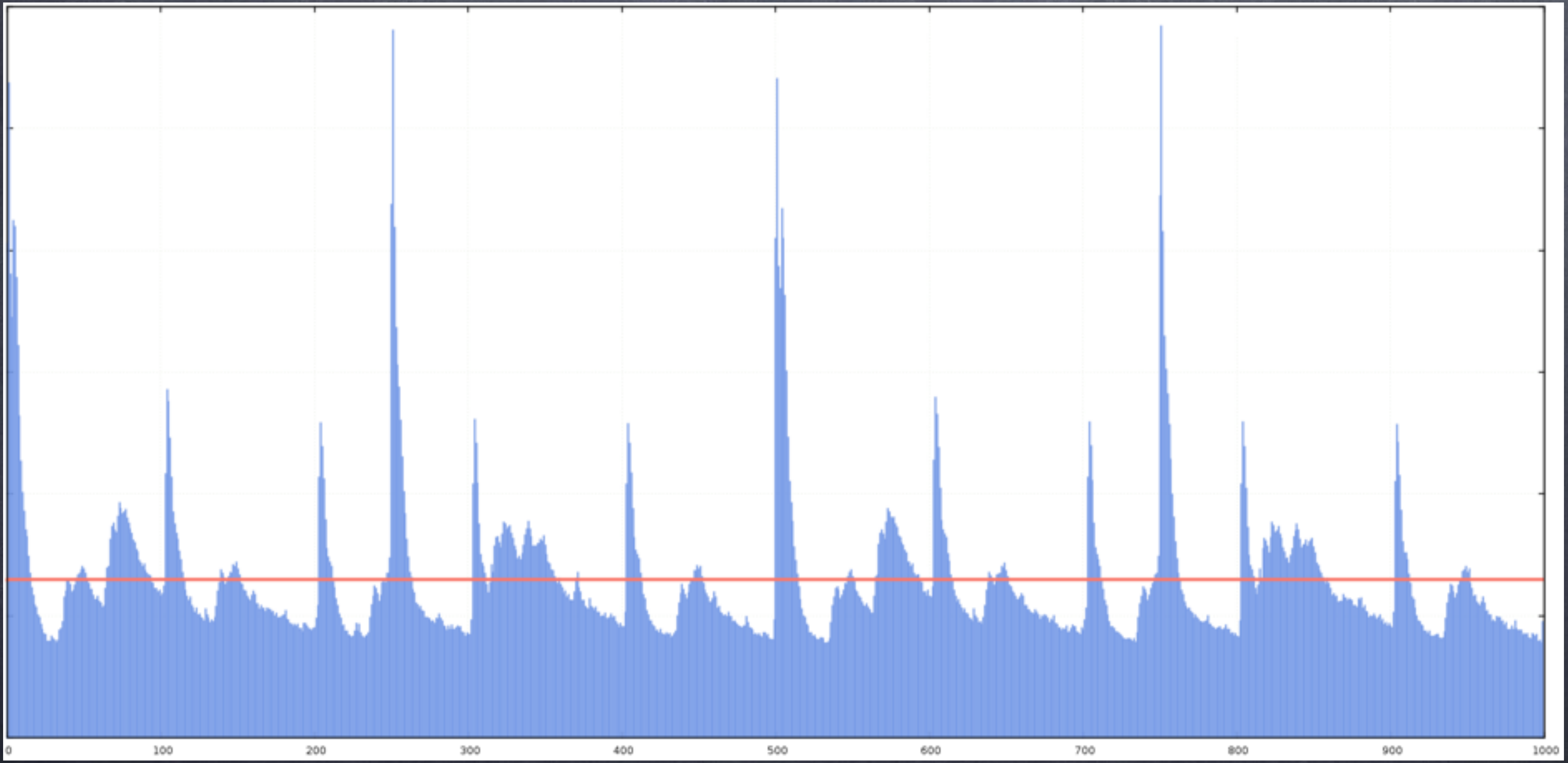
performance metric examples

- Operations per second (“speed”, “throughput”)
- Latency or Response Time (“quickness”)
- Failure rate (“reliability”, “availability”)
- Recovery time (e.g. to “normal”) after disruption
- Each of these is best measured when all the others are held to required levels

Example: How much "throughput" do we need?

- A system takes 100 usec to respond to a request
- The system will receive 1000 ops/sec during peeks
- Requirement: 99.9% of operations must complete in 20 msec of less, even during busiest second
- Does the system have the capacity to handle the the load with the required behavior?
- Maybe...
- E.g. what if 50 requests arrived in 1 msec?

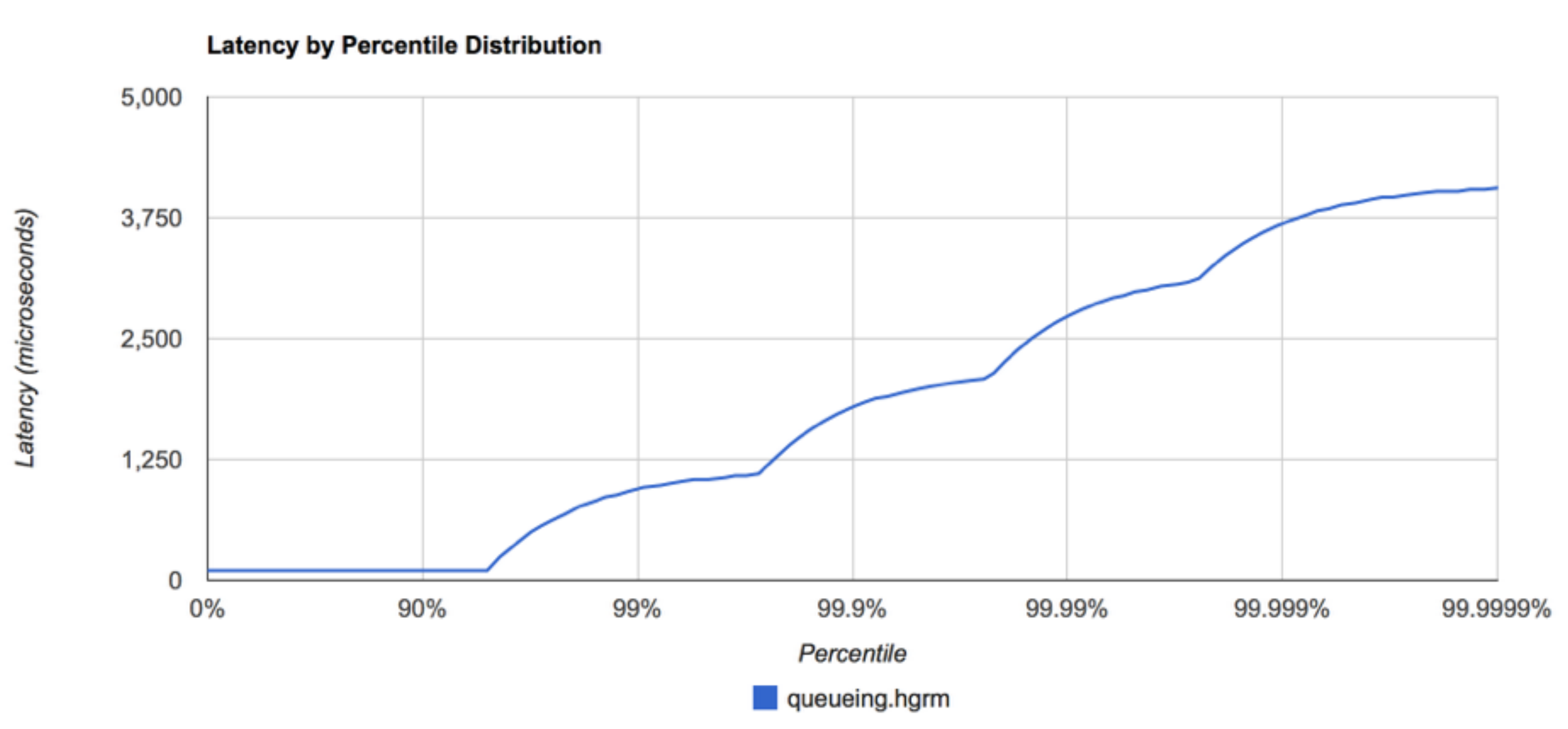
Arrival times



Example: arrival rate within a second (averaged over entire day)

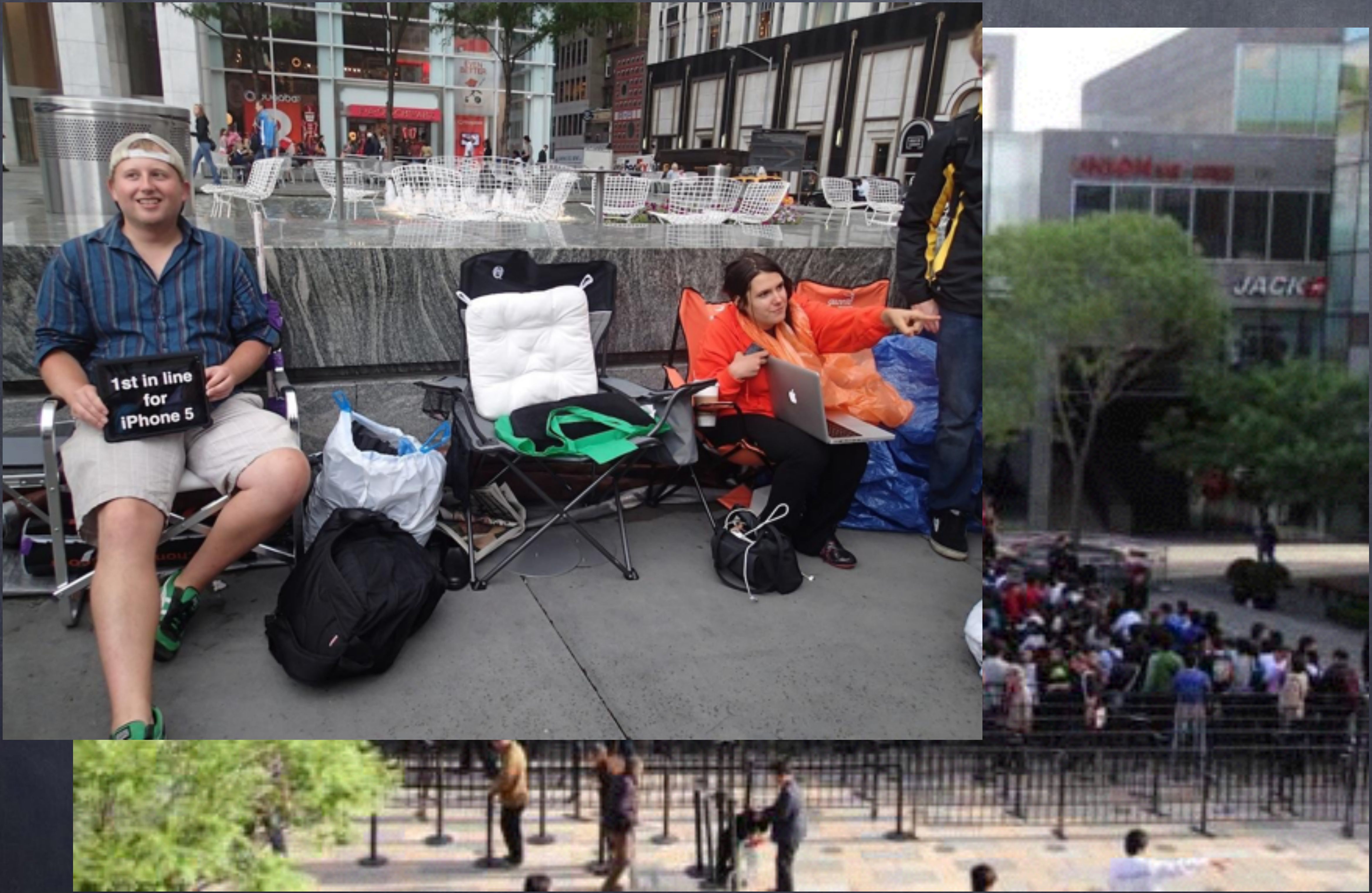
Service Time vs. Response Time

Latency behavior when bursts occur

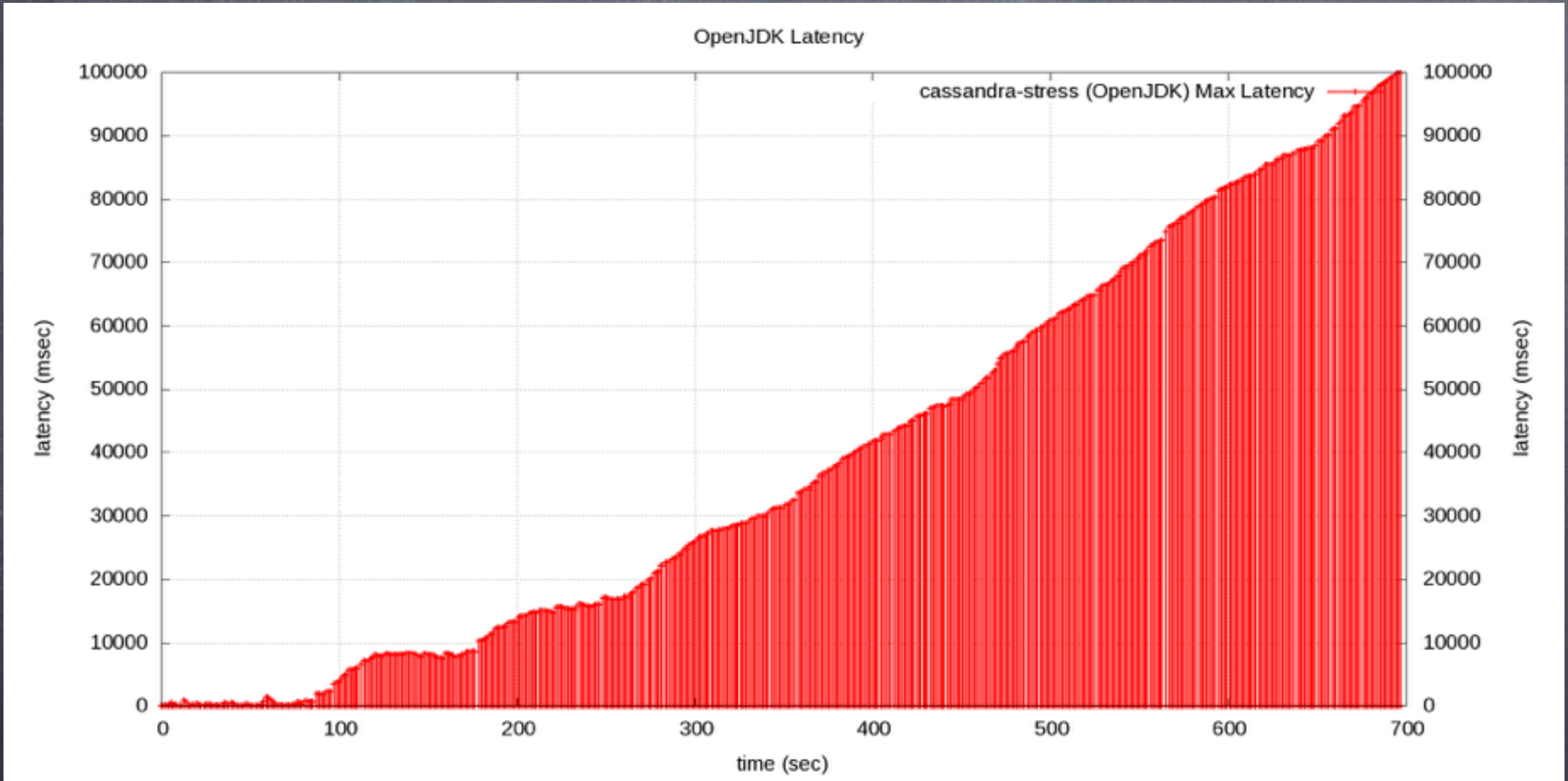


100 usec base latency, 5K msg/sec capacity
occasional bursts of 50 msgs
burst likelihood = 0.001

Lines can get pretty long…



Latency behavior when incoming rate is consistently faster than what we can handle



Cutting corners in performance testing

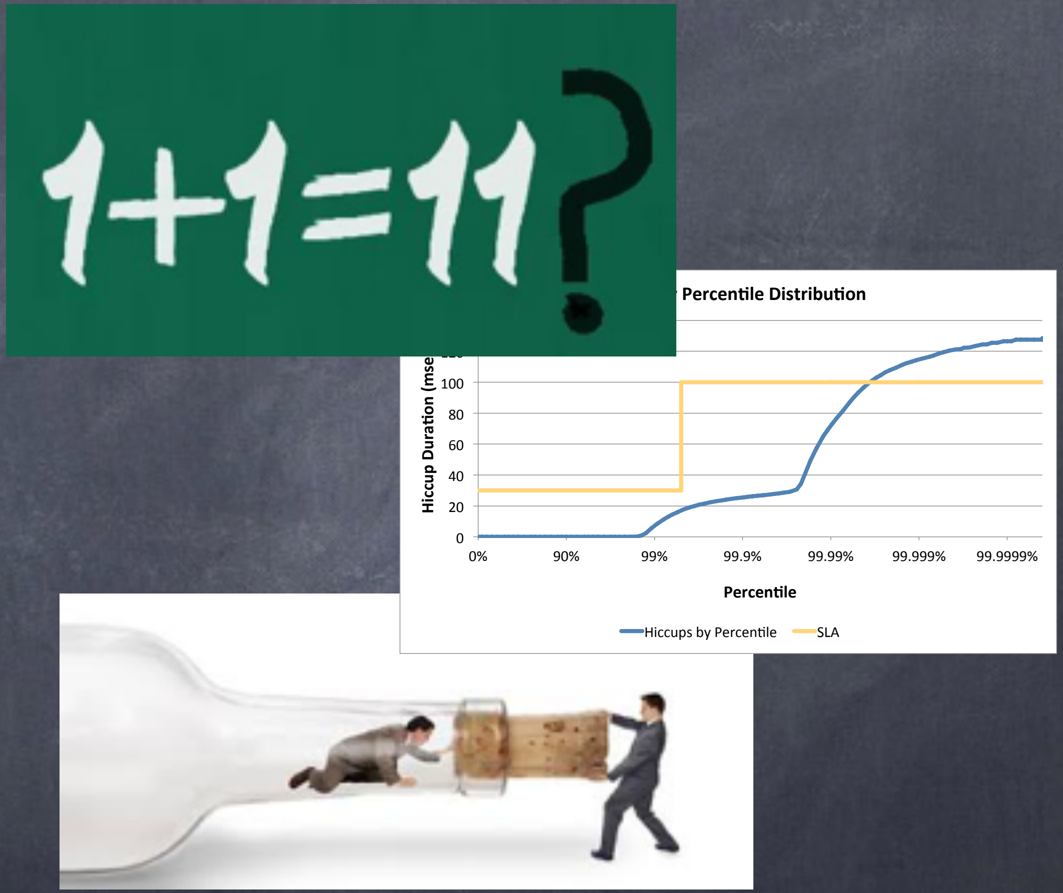
Typical Reaction

And most commonly

Repeat. Repeat. Repeat.

So before you start to measure something like widgets/sec:

- Establish requirements
  - Correctness
  - Timeliness
  - Availability, etc.



- Understand expected environmental limitations
  - Governing bottlenecks and realities

And most importantly

DO NOT consider “performance results” from non-passing tests

Managed Runtimes

What’s so special about them?

?

? Scala Ruby ?

Java Go F#

? Python

C# PHP Clojure

JavaScript Erlang

? Objective-C ?

C C++

? ? ?

Why?

Garbage Collection is… Good

- Productivity, stability
  - Programmers not responsible for freeing and destroying objects
  - Eliminates entire (common) areas of instability, delay, maintenance
- Guaranteed interoperability
  - No "memory management contract" needed across APIs
  - Uncoordinated libraries, frameworks, utilities seamlessly interoperate
- Facilitates practical use of large amounts of memory
  - Allows for complex and intertwined data structures
  - Within and across unrelated components
- Interesting concurrent algorithms become practical…

But most importantly

Garbage Collection makes things go fast faster

Time to market

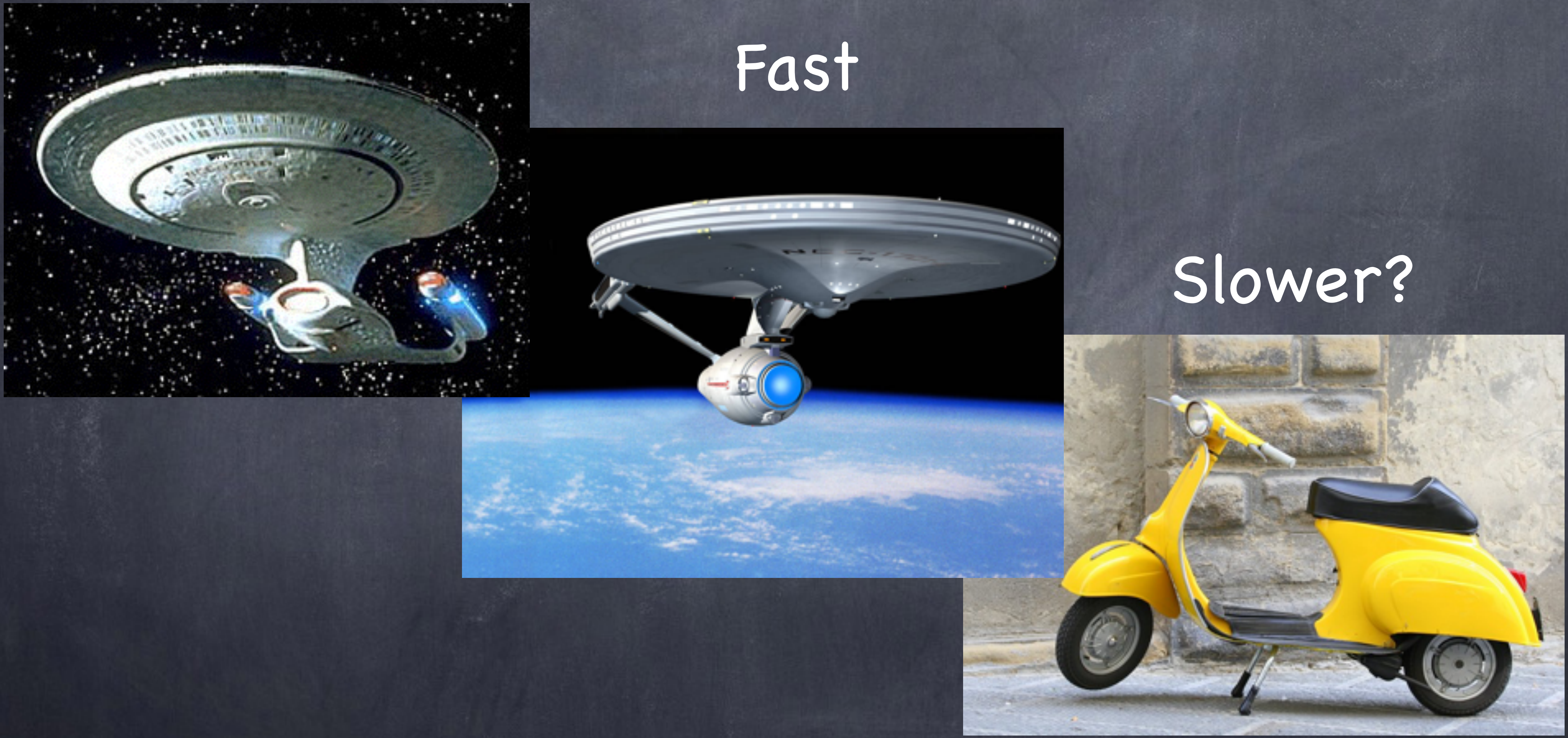
Time to performance

Theoretical Performance

Fastest?

Fast

Slower?



Pragmatic Performance

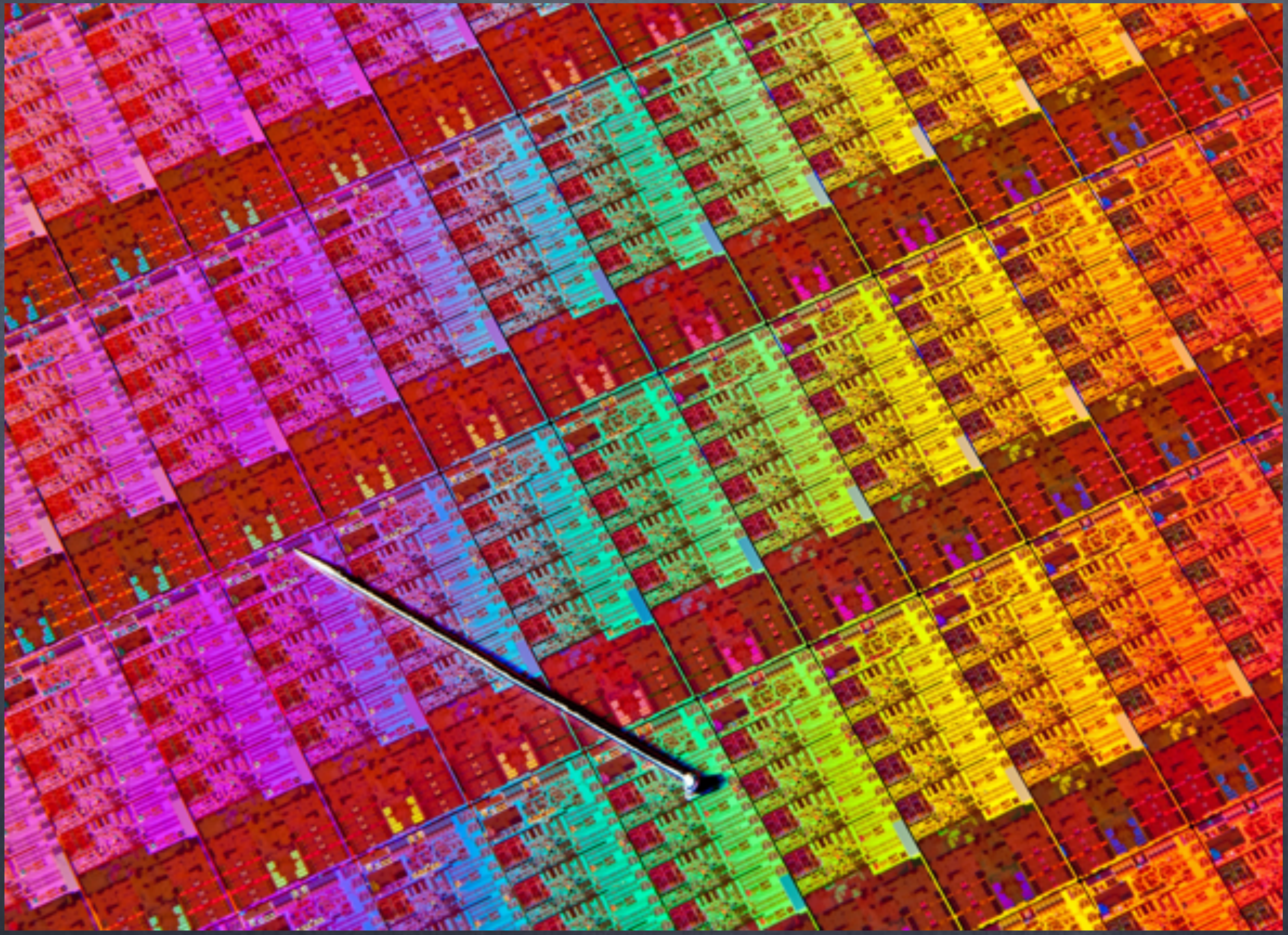
Which of these is fast enough
to get to work in 15 minutes or less?

Pragmatic Performance

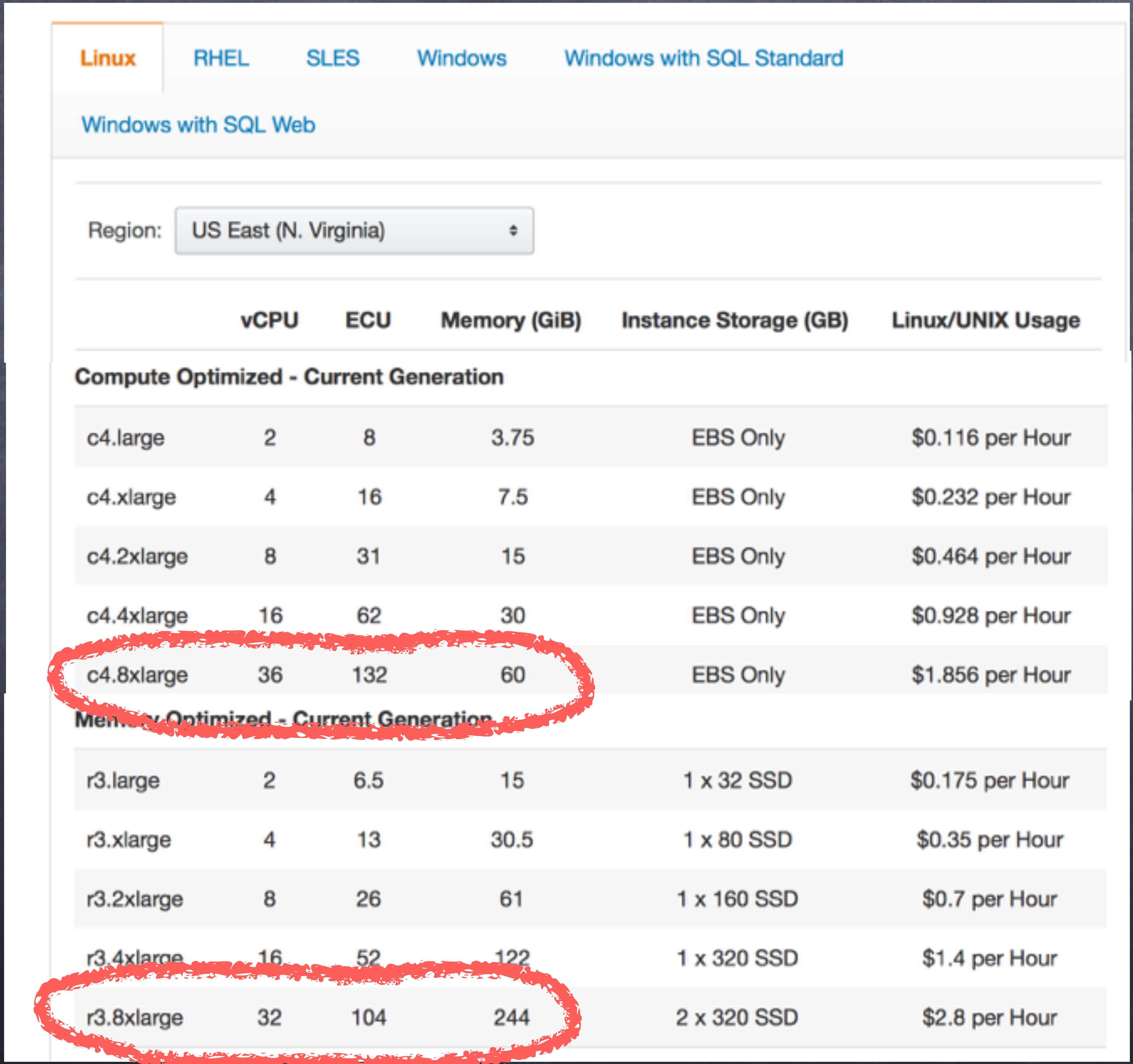
Which will provide the needed speed by [now + 6 months] ?

Tomorrow is here

Multicore is so 2012...



Current (2015) cloud stuff



Region: US East (N. Virginia)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
Compute Optimized - Current Generation					
c4.large	2	8	3.75	EBS Only	$0.116 per Hour
c4.xlarge	4	16	7.5	EBS Only	$0.232 per Hour
c4.2xlarge	8	31	15	EBS Only	$0.464 per Hour
c4.4xlarge	16	62	30	EBS Only	$0.928 per Hour
c4.8xlarge	36	132	60	EBS Only	$1.856 per Hour
Memory Optimized - Current Generation					
r3.large	2	6.5	15	1 x 32 SSD	$0.175 per Hour
r3.xlarge	4	13	30.5	1 x 80 SSD	$0.35 per Hour
r3.2xlarge	8	26	61	1 x 160 SSD	$0.7 per Hour
r3.4xlarge	16	52	122	1 x 320 SSD	$1.4 per Hour
r3.8xlarge	32	104	244	2 x 320 SSD	$2.8 per Hour

“lots” of cores

“lots” of memory

“Waste” and Performance

T-Mobile
12:31 PM
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Notes
Reminders
Stocks
Game Center
Newsstand
iTunes Store
App Store
iBooks
Health
Passbook
Settings
Phone
Mail
Safari
Music

“Waste”

Summary

- Pragmatic Performance
  - What actually needs to get done?
  - How much. How quickly. How fast.
  - What needs to remain true while we do the stuff
- Performance is (usually) not about efficiency
  - unless efficiency is your performance metric
- Do not be afraid to come up with creative "waste"

Q & A

@giltene

http://www.azulsystems.com

AZUL SYSTEMS