Scalability is Quantifiable

Universal Scalability Law
Baron Schwartz - November 2017
Logistics & Stuff

Slides will be posted :)  
Ask questions anytime!

Founder of VividCortex  
Wrote High Performance MySQL

Love to hear from you: @xaprb and baron@vividcortex.com
How Systems Fail Under Load

You’ve seen systems become sluggish under high load

How can we describe and reason about what’s happening?
Failure Boundaries

Cook and Rasmussen describe failure boundaries around the operating domain. One such is the unacceptable workload boundary.
Workload Failure Isn’t Crisp

Unacceptable workload is not sharply defined, it’s a gradient

Cook lists 18 precepts of system failure in “How Complex Systems Fail”

#5: Complex systems run in degraded mode
Workload Failure Isn’t Crisp

Cook introduces *error margin*. What’s the *workload margin*? What if you drift into it?
The Failure Boundary Is Nonlinear

This region is *highly* nonlinear and unintuitive. It’s analogous to post-elastic material behavior.
Capacity

Systems can, and do, function beyond their capacity limits.

Capacity limits are scalability limits.

How can we define and reason about system capacity?

Ditto, for scalability?
Queueing Theory

There’s a branch of operations research called queueing theory

It analyzes what happens to customers when systems get busy

Difficult to apply in “the real world” of capacity & ops
The hockey stick curve is difficult to use in practice. It is very nonlinear and hard for humans to intuit.
Scaling A System: Ideal

Suppose a clustered system can do X work per unit of time. Ideally, if you double the cluster size, it can do 2X work.
Equation

The linear scalability equation:

\[ X(N) = \frac{\lambda N}{1} \]

where \( \lambda \) is the slope of the line
But Our Cluster Isn’t Perfect

Speedup by executing tasks in parallel, e.g. ~ scatter-gather

What happens to performance if some portion isn’t parallelizable?
Amdahl’s Law

Amdahl’s Law describes the fraction $\sigma$ that can’t be done in parallel.

$$X(N) = \frac{\lambda N}{1 + \sigma(N-1)}$$

Adding nodes provides some speedup, but there’s a ceiling.
But What If Workers Coordinate?

Suppose the parallel workers have dependencies on each other?
N Workers = N(N-1) Pairs
Universal Scalability Law

Represent crosstalk (coherence) penalty by coefficient $\kappa$

$$X(N) = \frac{\lambda N}{1 + \sigma (N - 1) + \kappa N (N - 1)}$$

The system gets less work done as it gets more load!
Crosstalk Penalty Grows Fast

when we reach saturation, \( \kappa \) is growing very rapidly, again creating very nonlinear behavior
More About Crosstalk

Q: Isn’t crosstalk just a design flaw?
   
   A: Yes and no. Real-life: consensus, 2-phase commit, NUMA, etc…

Q: Doesn’t it seem odd to assume that crosstalk is a constant?
   
   A: It’s not, the amount of crosstalk-related work is a function of N
How Do You Measure Parameters?

How can you measure how much serialization/crosstalk you have? You don’t — USL is black-box. Measure the things on the axes… (cont’d)
How Do You Measure Parameters?

...Then use regression (least-squares curve fitting) to estimate the parameters of the equation. This lets you figure out $\sigma$ and $\kappa$ without needing to be able to measure.
Experiment Interactively
desmos.com/calculator/3cycsgdl0b
What is Scalability?

The USL is a mathematical definition of scalability

It’s a function that turns workload into throughput

It’s formally derived and has real physical meaning

\[ X(N) = \frac{\lambda N}{1 + \sigma(N - 1) + \kappa N(N - 1)} \]
But What Is Load?

In most circumstances we care about, load is concurrency.

Concurrency is the number of requests in progress.

It’s surprisingly easy to measure: \( \text{sum(latency)}/\text{interval} \)

Many systems emit it as telemetry:

- MySQL: `SHOW STATUS LIKE ‘Threads_running’`
- Apache: active worker count
Four Great Uses Of The USL
1. Forecast Workload Failure Boundary

The USL can reveal the workload failure boundary approaching
1. Forecast Workload Failure Boundary

You can use regression to extract the coefficients, then plot. Or pot and eyeball to see if you’re getting near the edge.
1. Forecast Workload Failure Boundary

Coda Hale wrote a thing about the USL
https://codahale.com/usl4j-and-you/
1. Forecast Workload Failure Boundary

- By estimating the parameters, you can forecast what you can’t see
- This means you can “load test” under load you don’t yet experience
- The USL is a pessimistic model, so you should expect better
- The USL is pessimistic, but you should be more pessimistic
2. Characterize Non-Scalability

Why doesn’t your system scale perfectly?

The USL reveals amount of serialization vs crosstalk
2. Characterize Non-Scalability

Paypal’s NodeJS vs Java benchmarks are a good example!

3. How Scalable SHOULD It Be?

The USL is a framework for making systems look really bad.

Many 10+ node MPP databases barely do anything per-node.

Calculate per-node a) clients b) data size c) throughput.

One 18-node database: 4000 QPS ~220 QPS/node, 5ms latency.
3. How Scalable SHOULD It Be?

This is an animation of how Citus’s distributed database works
For the record: Citus isn’t one of the terribly unscalable DB’s
4. See Your Teams As Systems

Rick Branson @rbranson · 17 Aug 2016
The hope of the open office is that people will coordinate more regularly, but coordination is the wolf in sheep's clothing.

Tim Kellogg @kellogg
Replies to @rbranson

yep, if you need to scale, then you need to *reduce* communication. Find ways to avoid coordination and still be successful
4. See Your Teams As Systems

“To go fast, go alone. To go far, go together.”

Adrian Colyer wrote a good blog post about teams-as-systems and USL

4. See Your Teams As Systems
The USL isn’t novel in that sense...
What Else Can The USL Illuminate?

Open-plan offices: My work takes more work when others are nearby

Map-Reduce: That’s a whole lotta overhead, but it sure is scalable

Mutexes: Theoretically just serialize, but those damn OS schedulers
What’s NOT Scalability?

I commonly see throughput-vs-latency charts. This seems legit till you get systems under high load.
Scalability Isn’t Throughput-vs-Latency

The throughput-vs-latency equation has two solutions
Concurrency-vs-Latency is OK

It’s a simple quadratic per Little’s Law, and is quite useful
Some Resources

I wrote a book.
I created an Excel sheet.
Conclusions

Scalability is formally definable, and black-box observable

Scalability is nonlinear; this region is the failure boundary

Scalability is a function with parameters you can estimate
Further Reading/References

- https://www.vividcortex.com/resources/ for ebook, Excel worksheet
- http://www.perfdynamics.com/Manifesto/USLscalability.html for the original source