Quantifying Scalability with the USL

Baron Schwartz · DataEngConf NYC 2018

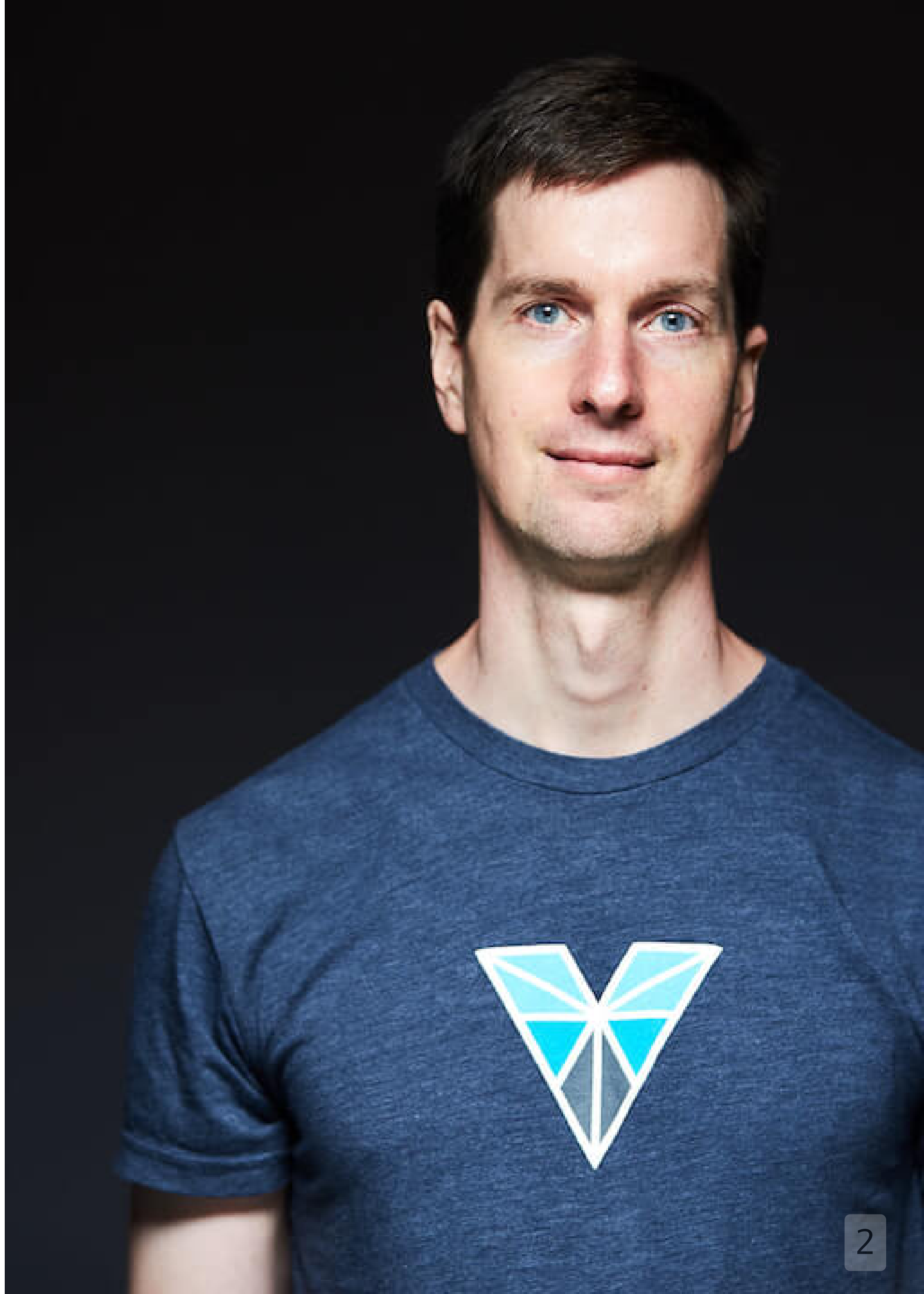
VividCortex

Introduction

I've been focused on databases for about two decades, first as a developer, then a consultant, and now a startup founder.

I've written High Performance MySQL and several other books, and created a lot of open source software, mostly focused around database monitoring, database operations, and database performance: innotop, Percona Toolkit, etc.

I welcome you to get in touch at @xaprb or baron@vividcortex.com.



Agenda

How can you quantify, forecast, and reason about scalability?

1. Queueing theory.
   In which we discover load

Agenda

How can you quantify, forecast, and reason about scalability?

1. Queueing theory.
   In which we discover load
2. Amdahl's Law.
   In which we define linearity

Agenda

How can you quantify, forecast, and reason about scalability?

1. Queueing theory.
   In which we discover load
2. Amdahl's Law.
   In which we define linearity
3. The Universal Scalability Law (USL).
   In which Frederick Brooks laughs last

Agenda

How can you quantify, forecast, and reason about scalability?

1. Queueing theory.
   In which we discover load
2. Amdahl's Law.
   In which we define linearity
3. The Universal Scalability Law (USL).
   In which Frederick Brooks laughs last
4. Application.
   In which things are even worse than we thought

Agenda

How can you quantify, forecast, and reason about scalability?

1. Queueing theory.
   In which we discover load
2. Amdahl's Law.
   In which we define linearity
3. The Universal Scalability Law (USL).
   In which Frederick Brooks laughs last
4. Application.
   In which things are even worse than we thought
5. Profit???
   In which we do the impossible

Queueing Theory

In Which We Discover Load

Queueing Theory

There's a branch of operations research called queueing theory.

It analyzes the **waiting** that happens when systems get busy.

What Causes Queueing?

Queueing happens even at low utilization:

1. Irregular arrival timings
2. Irregular job sizes
3. Lost time is lost forever

What Causes Queueing?

Queueing happens even at low utilization:

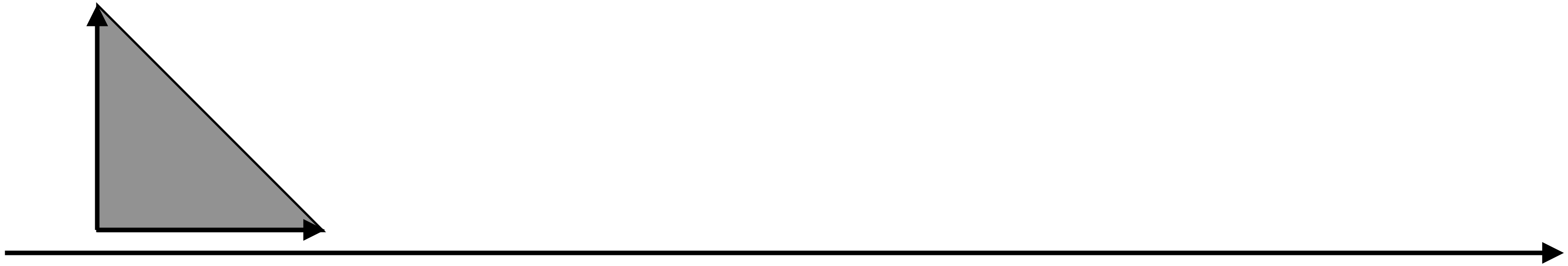
1. Irregular arrival timings
2. Irregular job sizes
3. Lost time is lost forever

A queue fundamentally changes how a system works:

- Increases availability and utilization
- Increases average residence time
- Increases cost/overhead

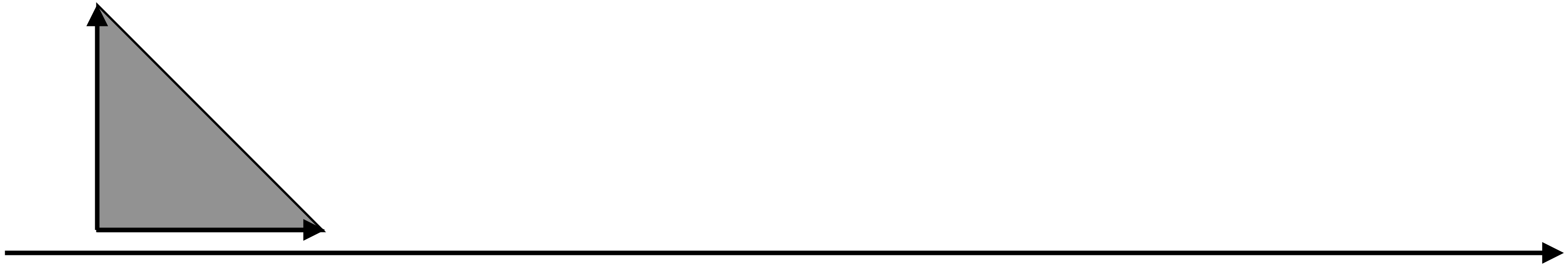
Arrival Rate and Queue Delay

Eben Freeman has a great visual that explains how arrival rate λ is related to queueing delay.



Arrival Rate and Queue Delay

Eben Freeman has a great visual that explains how arrival rate λ is related to queueing delay.



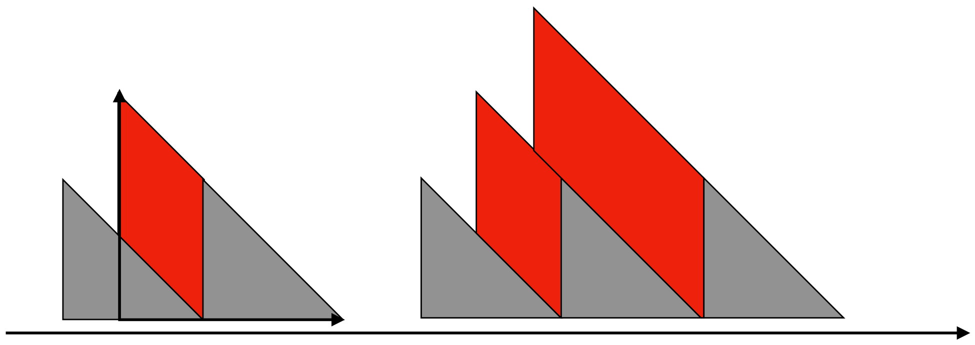
- A request arrives, and the server processes it until it's finished
- The height is the job size, and the width is the service time S
- The upper edge of the triangle is the amount of outstanding work to do

Another Request Arrives

- It has to wait W in the queue until the first is done
- Then it has S service time too
- Its total residence time $R = W + S$

An Equation For Queue Wait

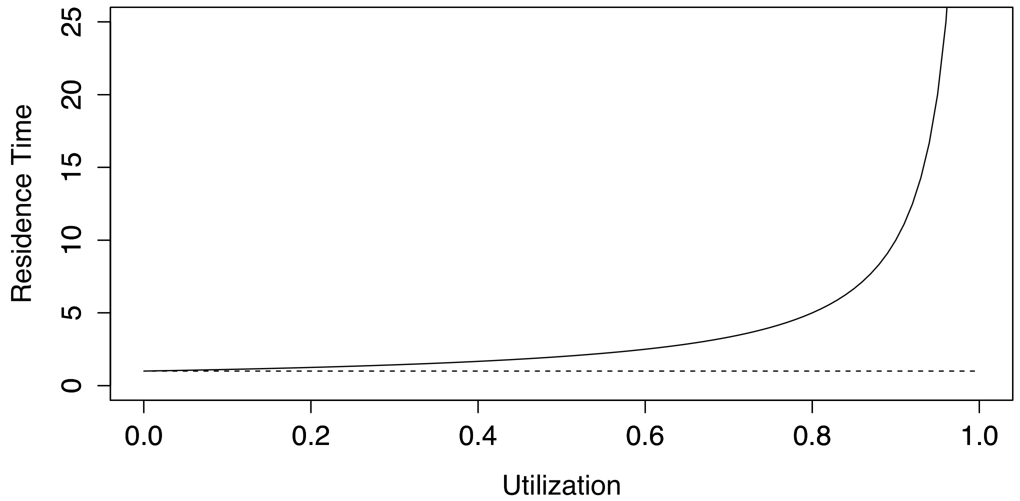
Eben uses the area under the graph to relate the height of the top edge to the width of the red wait parallelograms:



Solving this for W gives an equation for wait time:

$$W = \frac{\lambda S^2}{2(1 - \lambda S)}$$

This creates the familiar hockey stick curve, shown here in terms of utilization ρ.



Some Implications

One of the nice things about this form is that it lets you reason about service time and arrival rate easily:

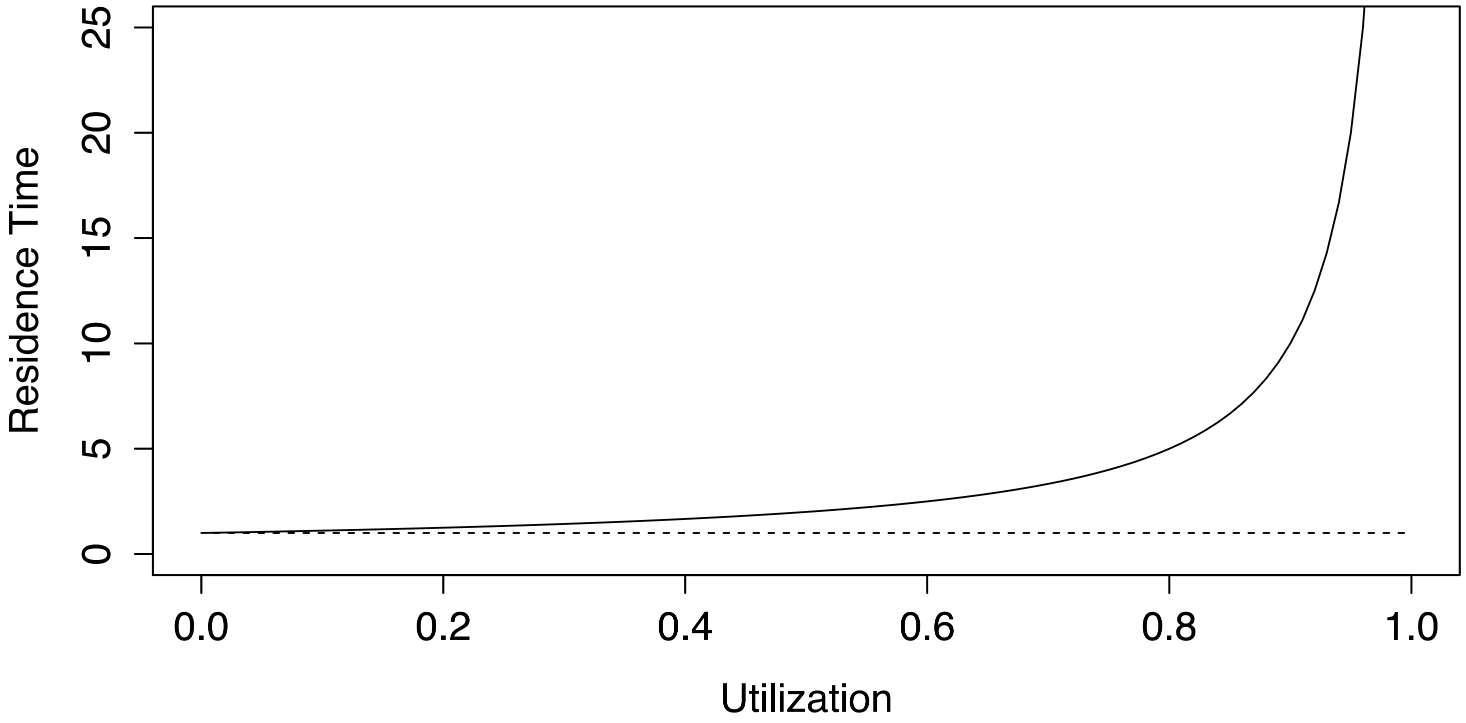
$$W = \frac{\lambda S^2}{2(1-\lambda S)}$$

What if you...

- *double* the arrival rate λ
- *halve* the service time S

The Hockey Stick Curve

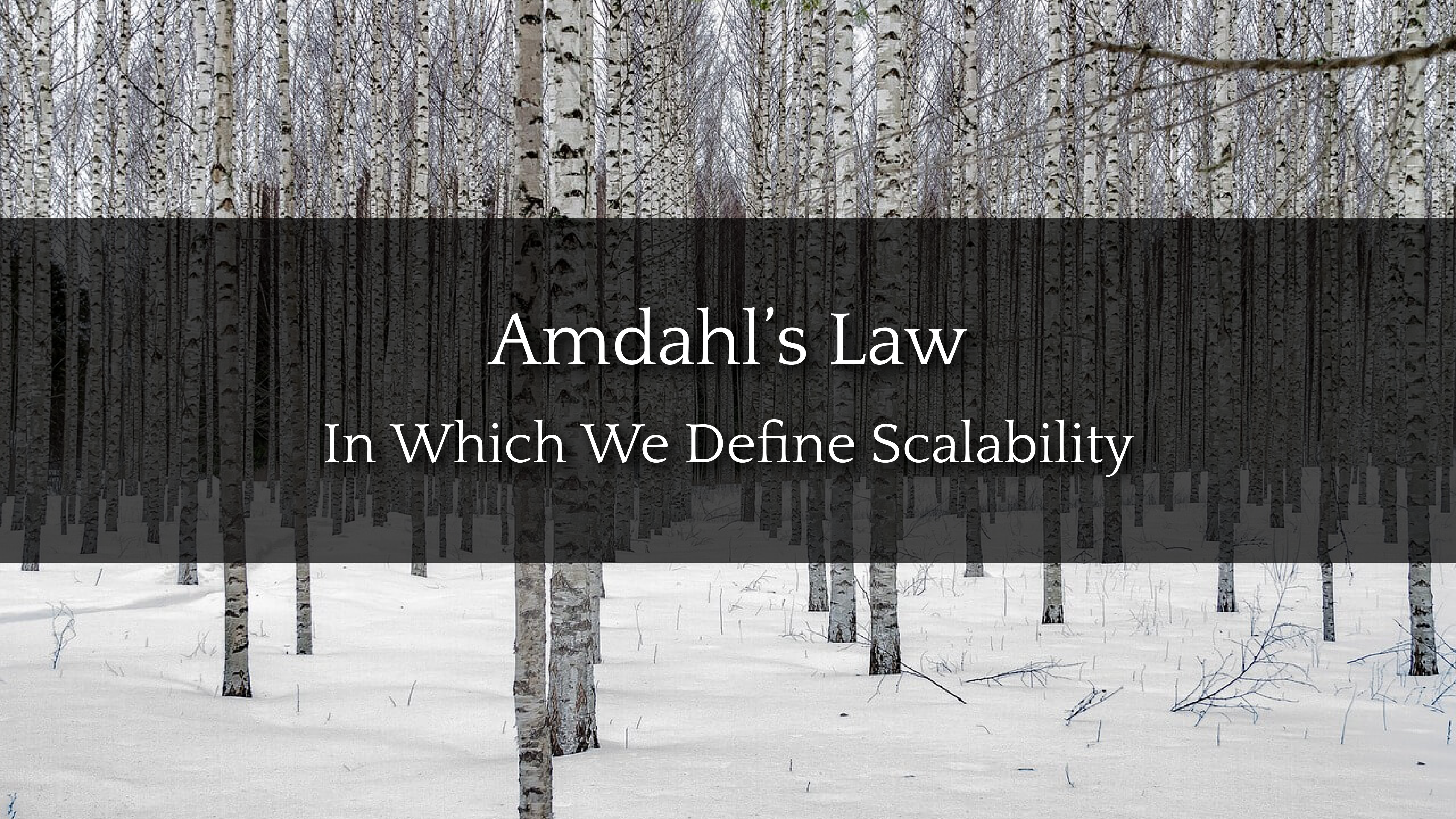
The “hockey stick” queueing curve is hard to use in practice. And the sharpness of the “knee” is nonlinear and very hard for humans to intuit.



Great Truths From Queueing Theory

1. Requests into ~any system have to queue and wait for service.
2. As the system gets busier, queueing escalates suddenly.
3. Queueing is very sensitive to service time and variability.
4. Contention over serialized resources causes nonlinear scaling.

The last point is quite a leap, but I'll explain.



Amdahl's Law

In Which We Define Scalability

What is Scalability?

There's a mathematical definition of scalability **as a function of concurrency**.

What is Scalability?

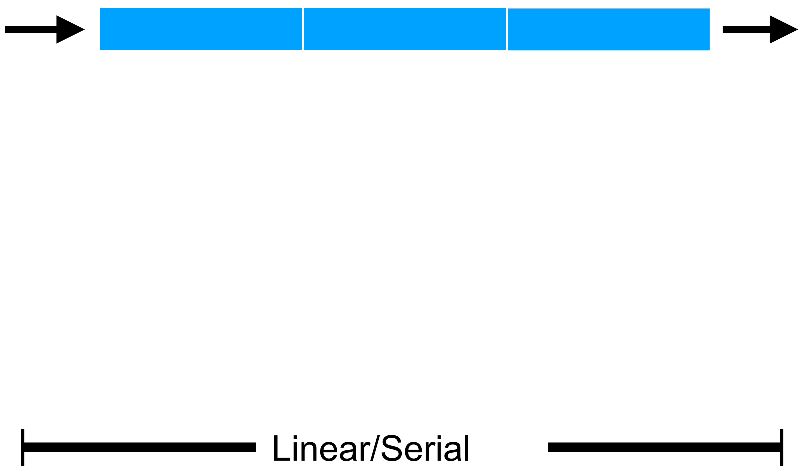
There's a mathematical definition of scalability **as a function of concurrency**.

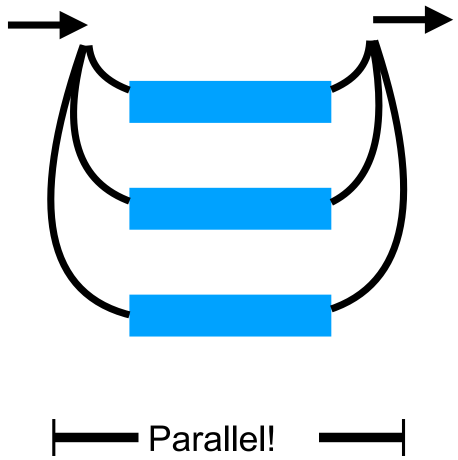
I'll illustrate it in terms of a **parallel processing system** that uses concurrency to achieve speedup.

Linear Scaling

Suppose a clustered system can complete X tasks per second with no parallelism.

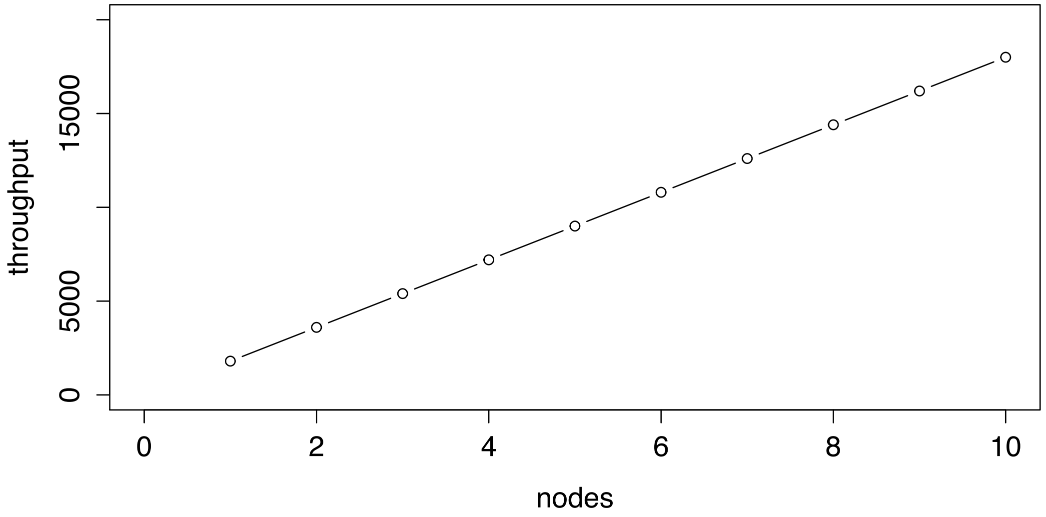
With parallelism, it completes tasks faster, e.g. higher throughput.





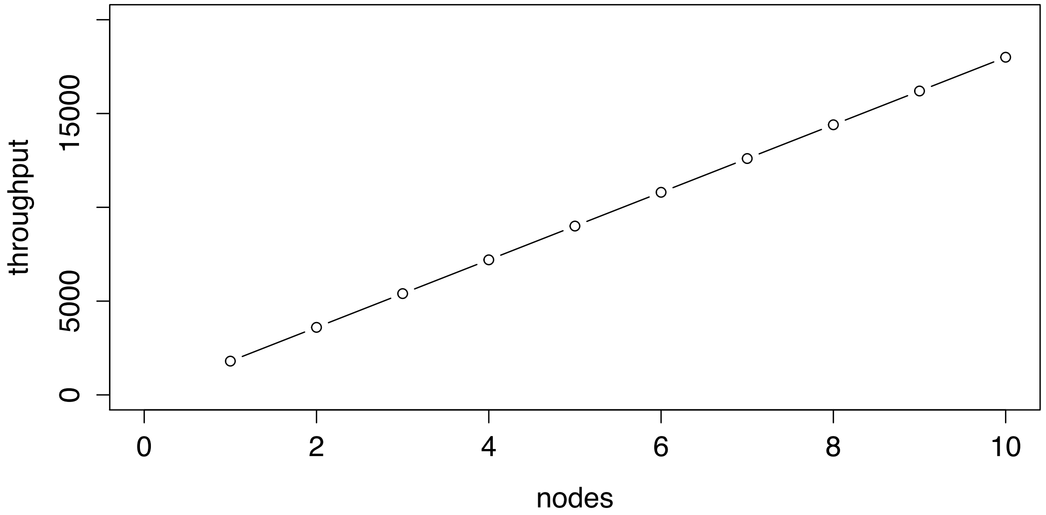
Ideal Linear Scalability

Ideally, **throughput increases linearly with parallelism**.



Ideal Linear Scalability

Ideally, **throughput increases linearly with parallelism**.



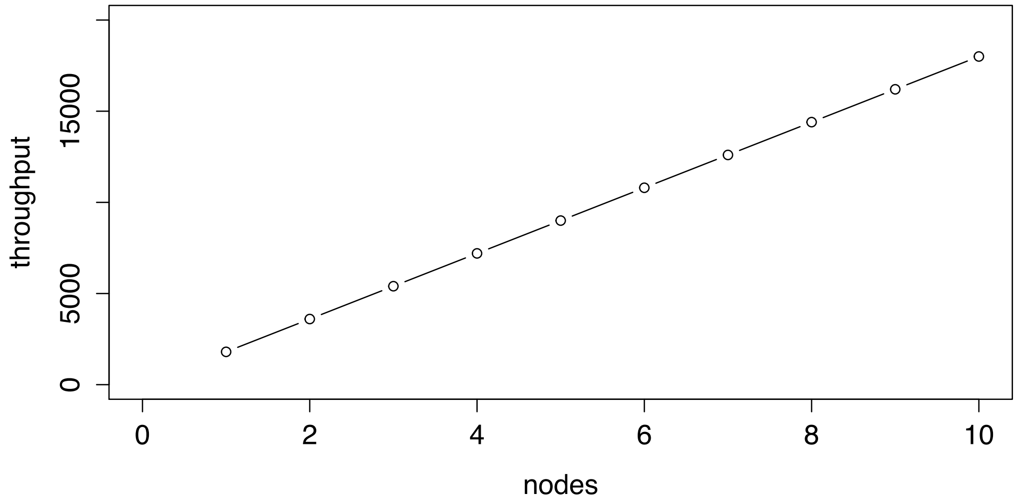
For example, triple the parallelism means $3X$ as much work completes.

The Linear Scalability Equation

The equation of ideal linear scaling:

$$X(N) = \frac{\gamma N}{1}$$

where the slope is $\gamma = X(1)$.



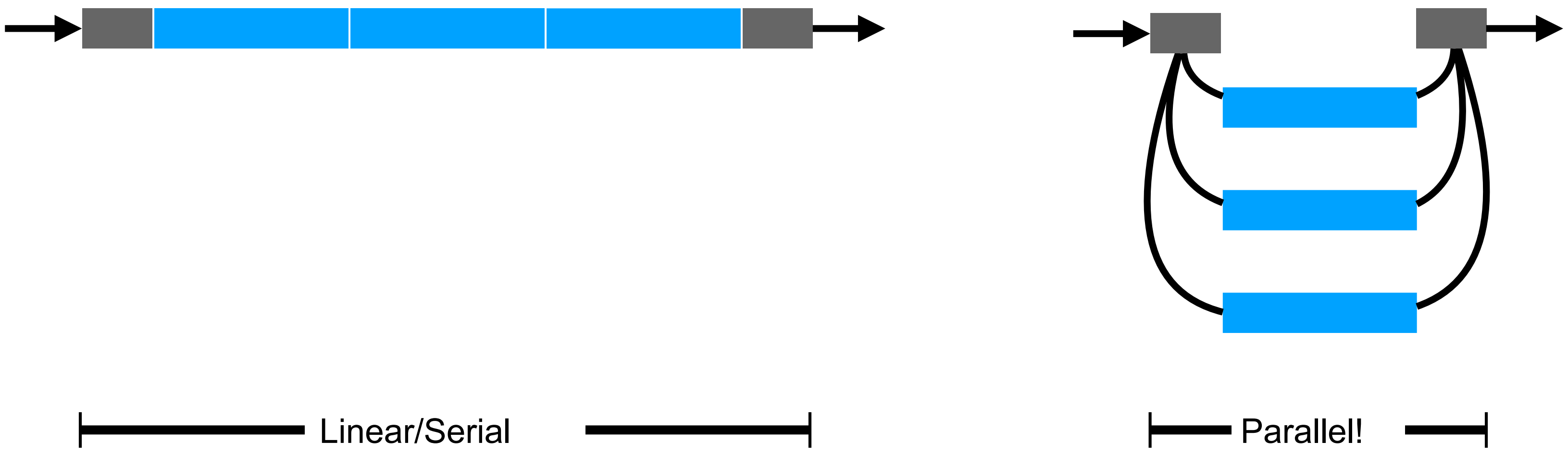
But Our Cluster Isn't Perfect

Linear scaling comes from subdividing tasks **perfectly**.

But Our Cluster Isn't Perfect

Linear scaling comes from subdividing tasks **perfectly**.

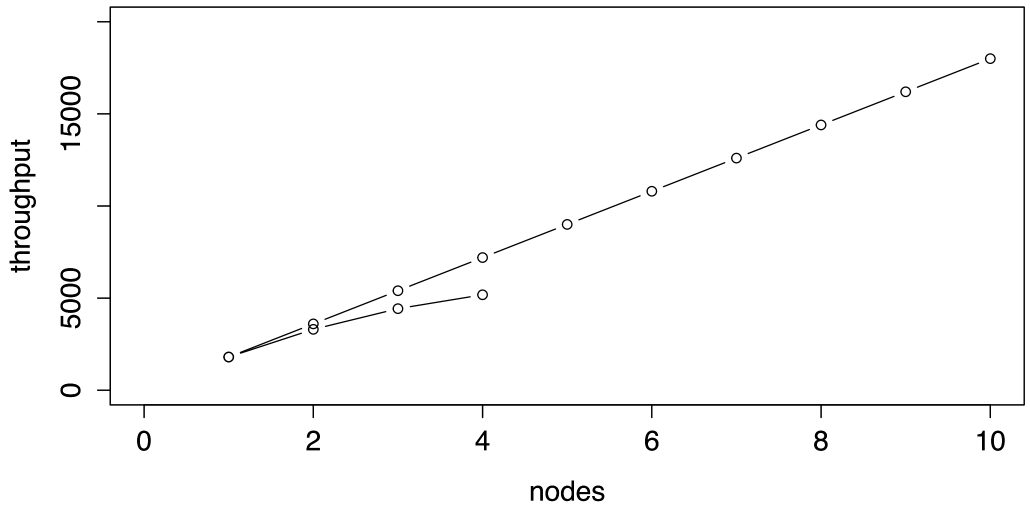
What if a portion isn't subdividable?



Amdahl's Law Describes Serialization

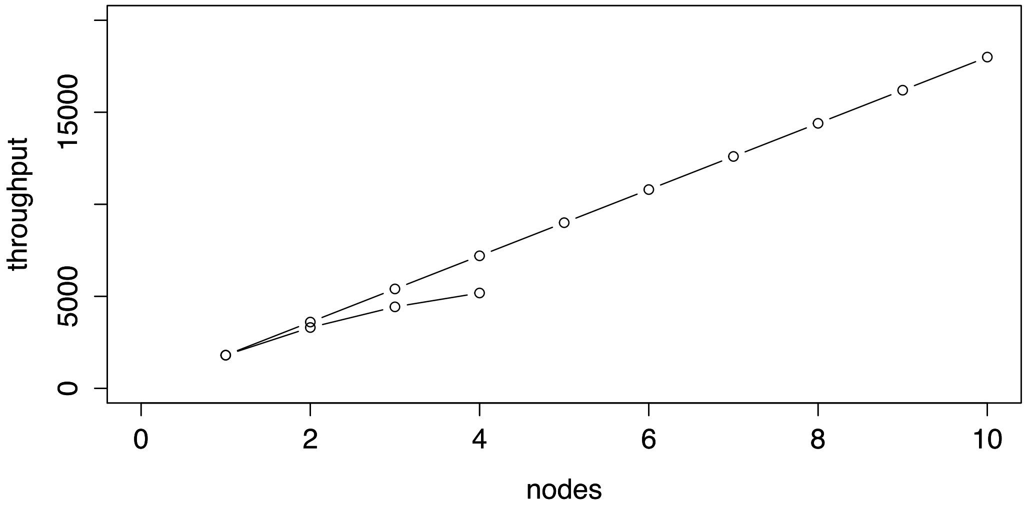
$$X(N) = \frac{\gamma N}{1 + \sigma(N-1)}$$

Amdahl's Law describes throughput when **a fraction σ can't be parallelized**.



Amdahl's Law Describes Serialization

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



Amdahl's Law describes throughput when **a fraction σ can't be parallelized**.

Serialization is queueing.

Amdahl's Law Has An Asymptote

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

Parallelism delivers speedup, but there's a limit:

$$\lim_{N \to \infty} X(N) = \frac{1}{\sigma}$$

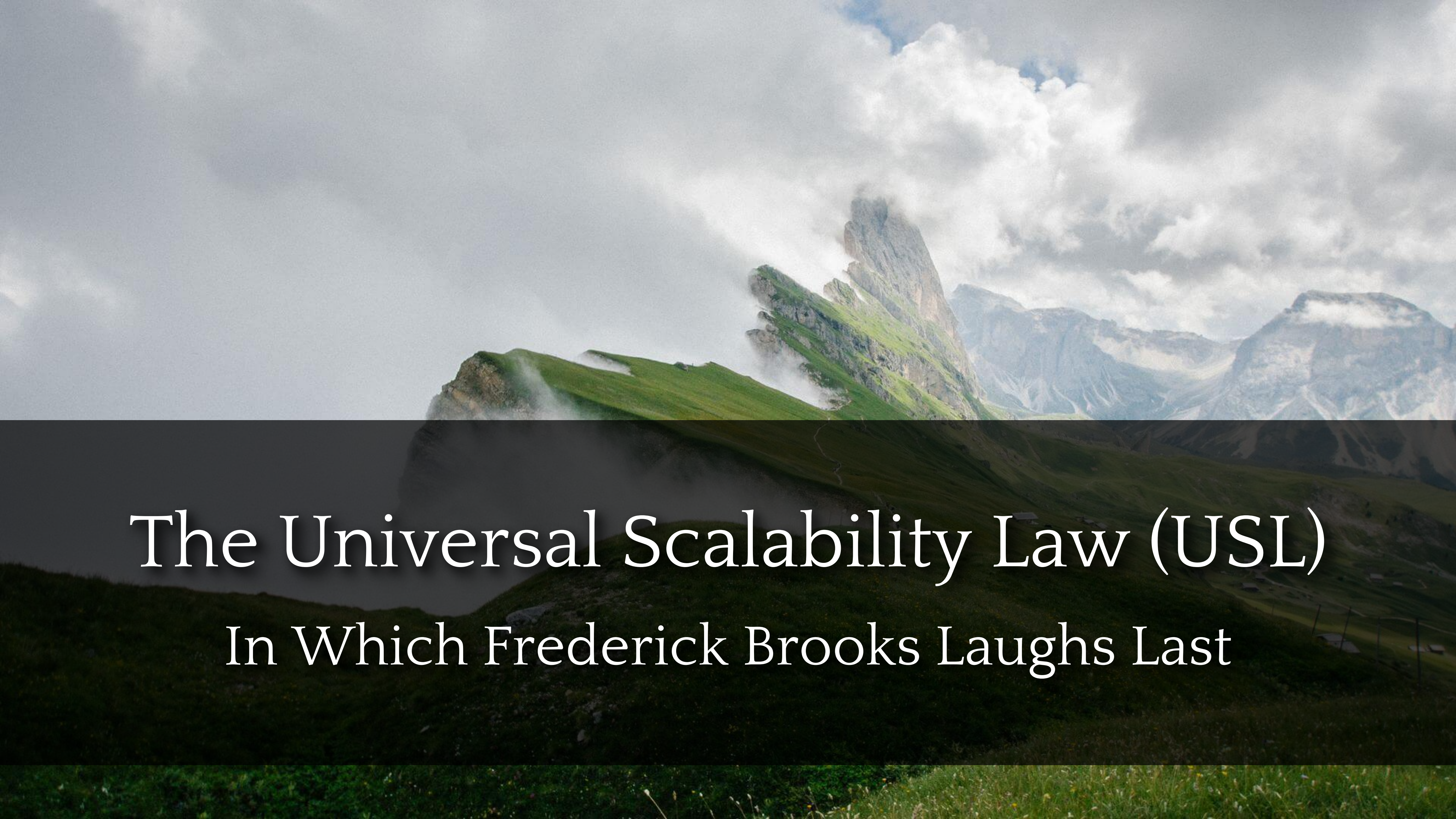
Amdahl's Law Has An Asymptote

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

Parallelism delivers speedup, but there's a limit:

$$\lim_{N \to \infty} X(N) = \frac{1}{\sigma}$$

e.g. a 5% serialized task can't be sped up more than 20-fold.

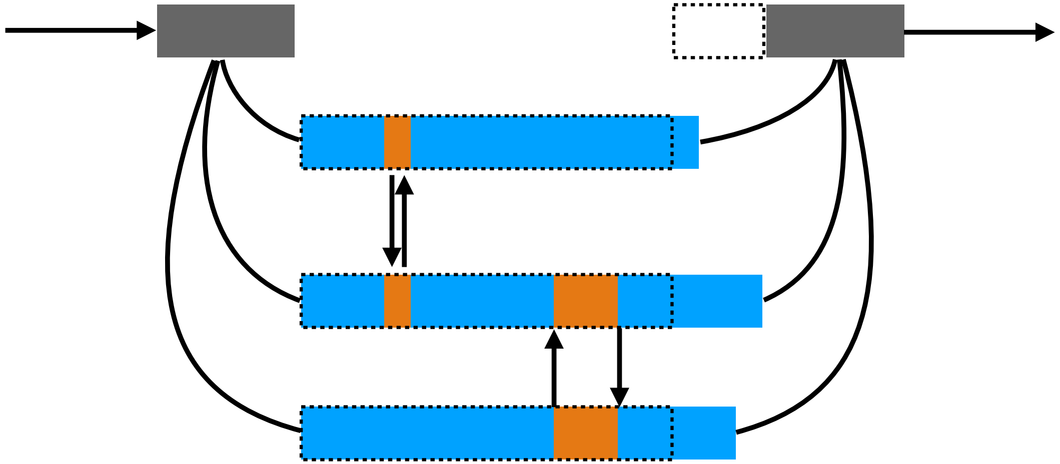


The Universal Scalability Law (USL)

In Which Frederick Brooks Laughs Last

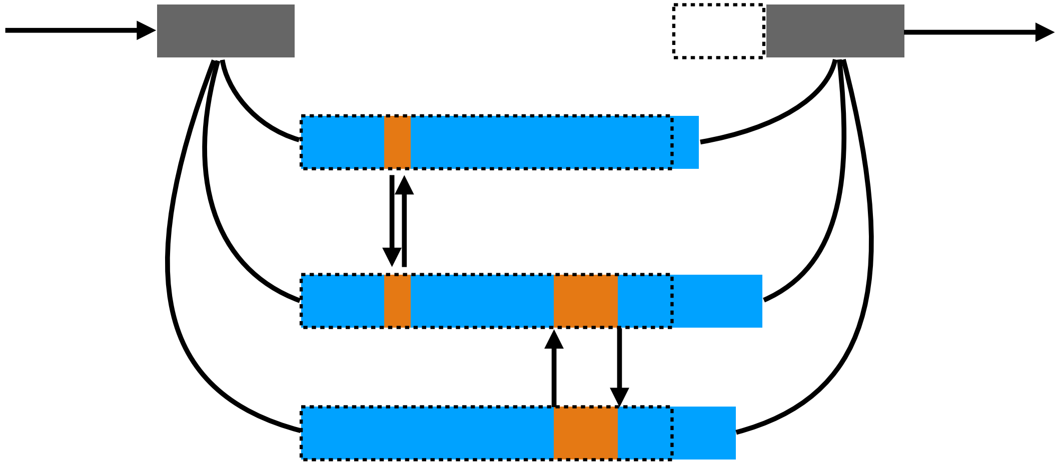
What If Workers Coordinate?

Suppose the parallel workers also **ask each other for things**?



What If Workers Coordinate?

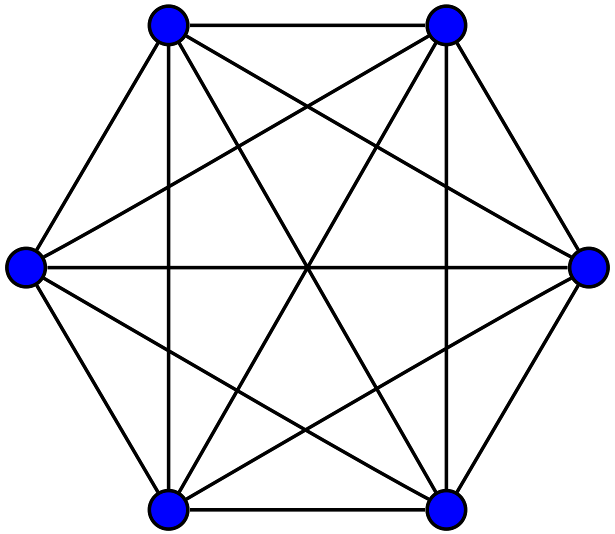
Suppose the parallel workers also **ask each other for things**?

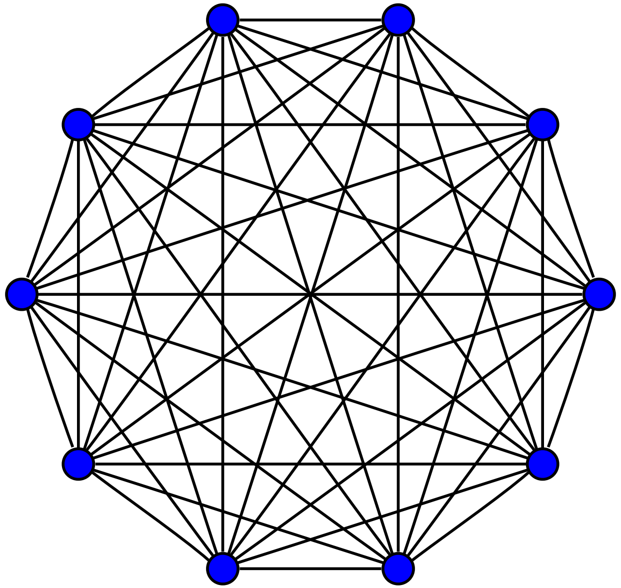


They're making each other do extra work. As load increases, **each task's job gets harder**.

How Bad Is Coordination?

N workers = $N(N-1)$ pairs of interactions, which grows fast: $\mathcal{O}(n^2)$ in N.



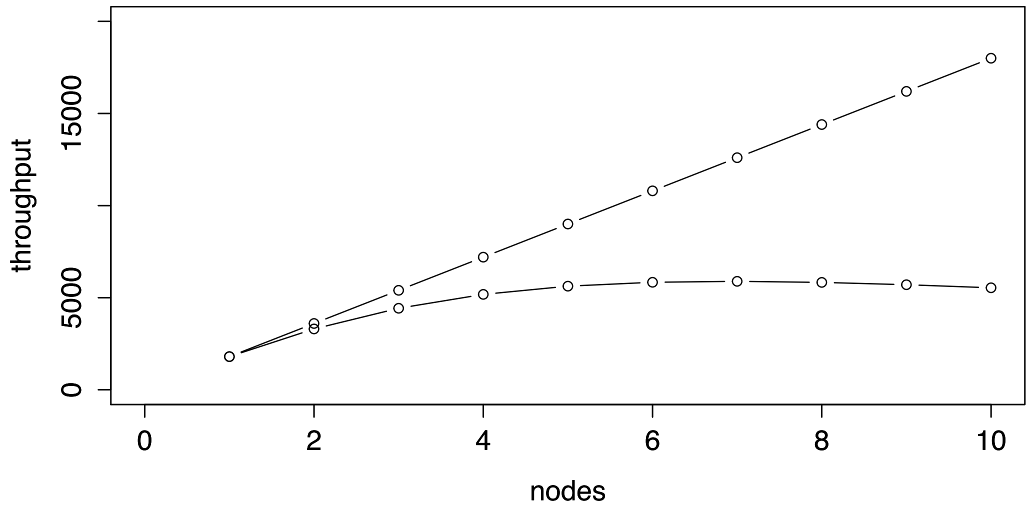


The Universal Scalability Law

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

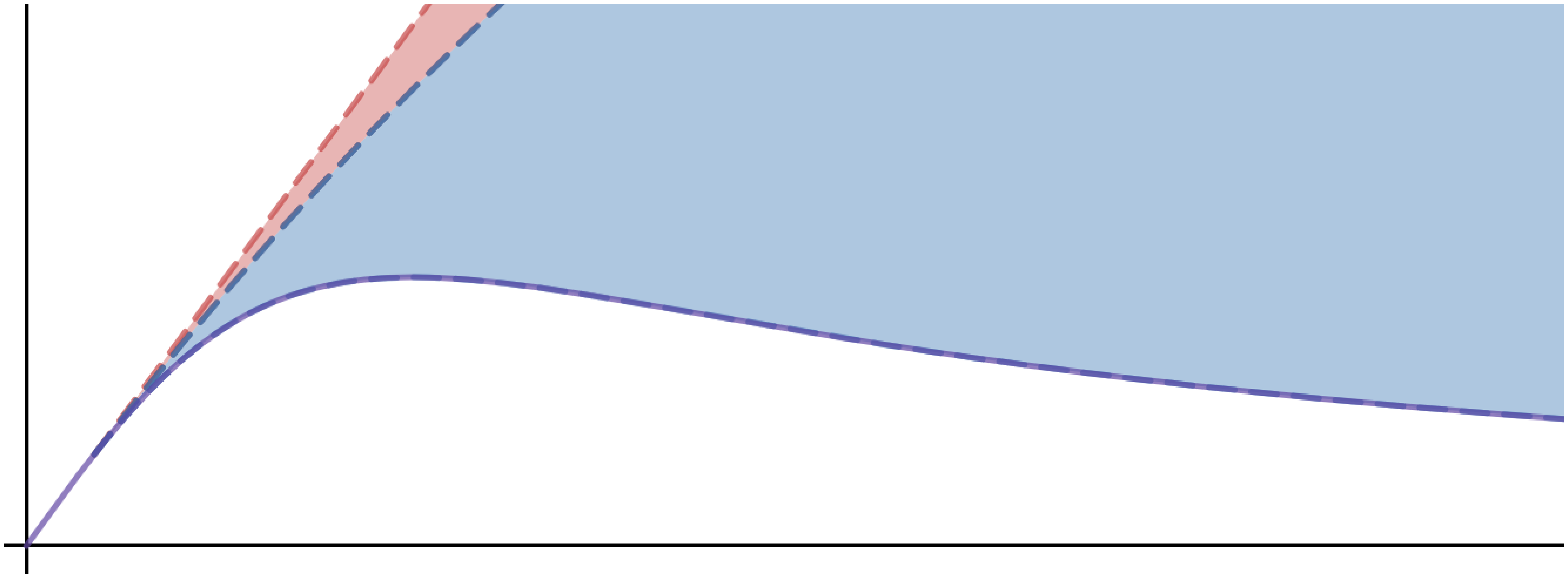
The USL adds a term for crosstalk, multiplied by the κ coefficient. Crosstalk is also called coordination or coherence penalty.

Now there's a **point of diminishing returns**!



The USL Describes Behavior Under Load

The USL explains the **highly nonlinear behavior** we know systems exhibit near their saturation point. desmos.com/calculator/3cycsgdl0b



Application

In Which Things Are Even Worse Than We Thought

Applying the USL to the Real World

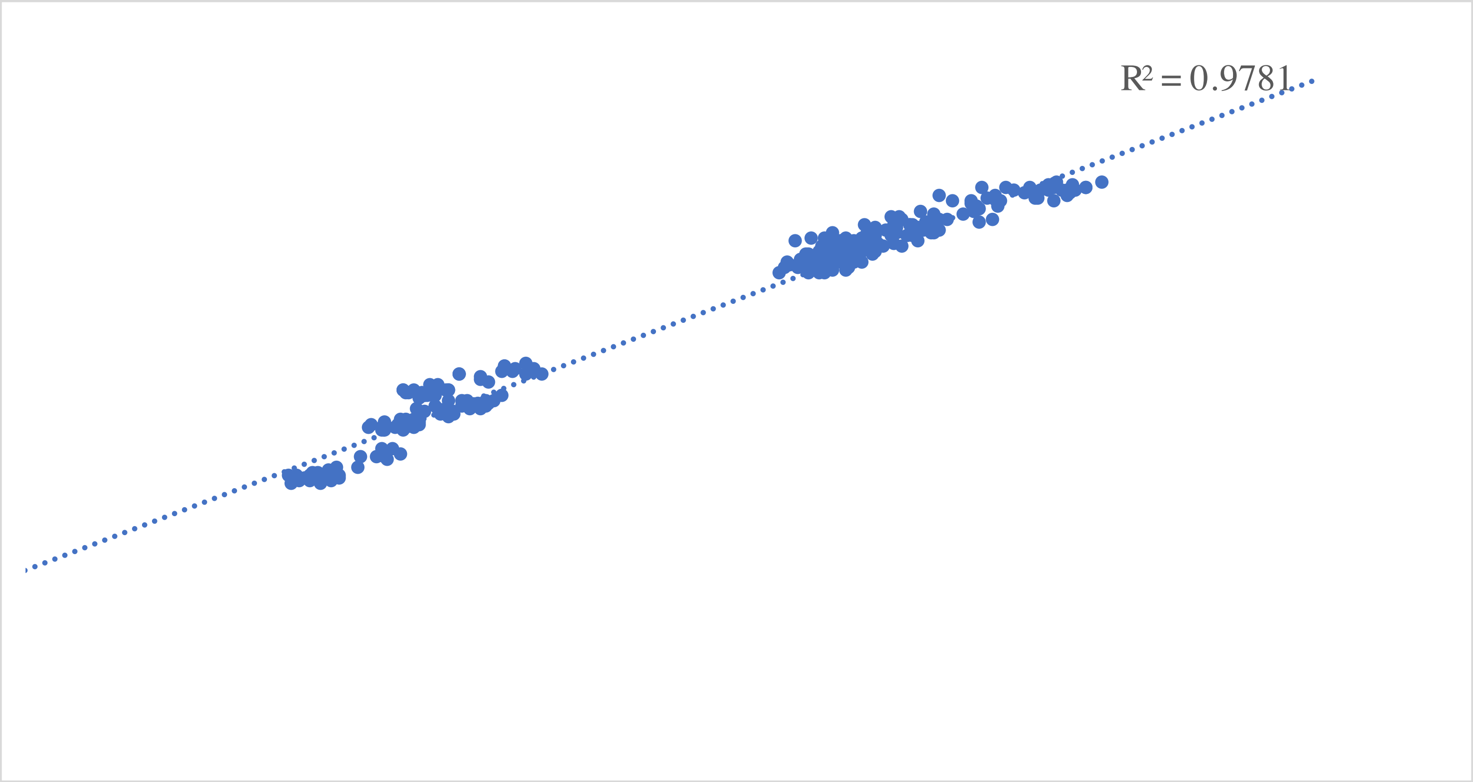
Behold, I give you two metrics of concurrency and throughput.

What do they mean?

Let's Scatterplot Concurrency vs Throughput

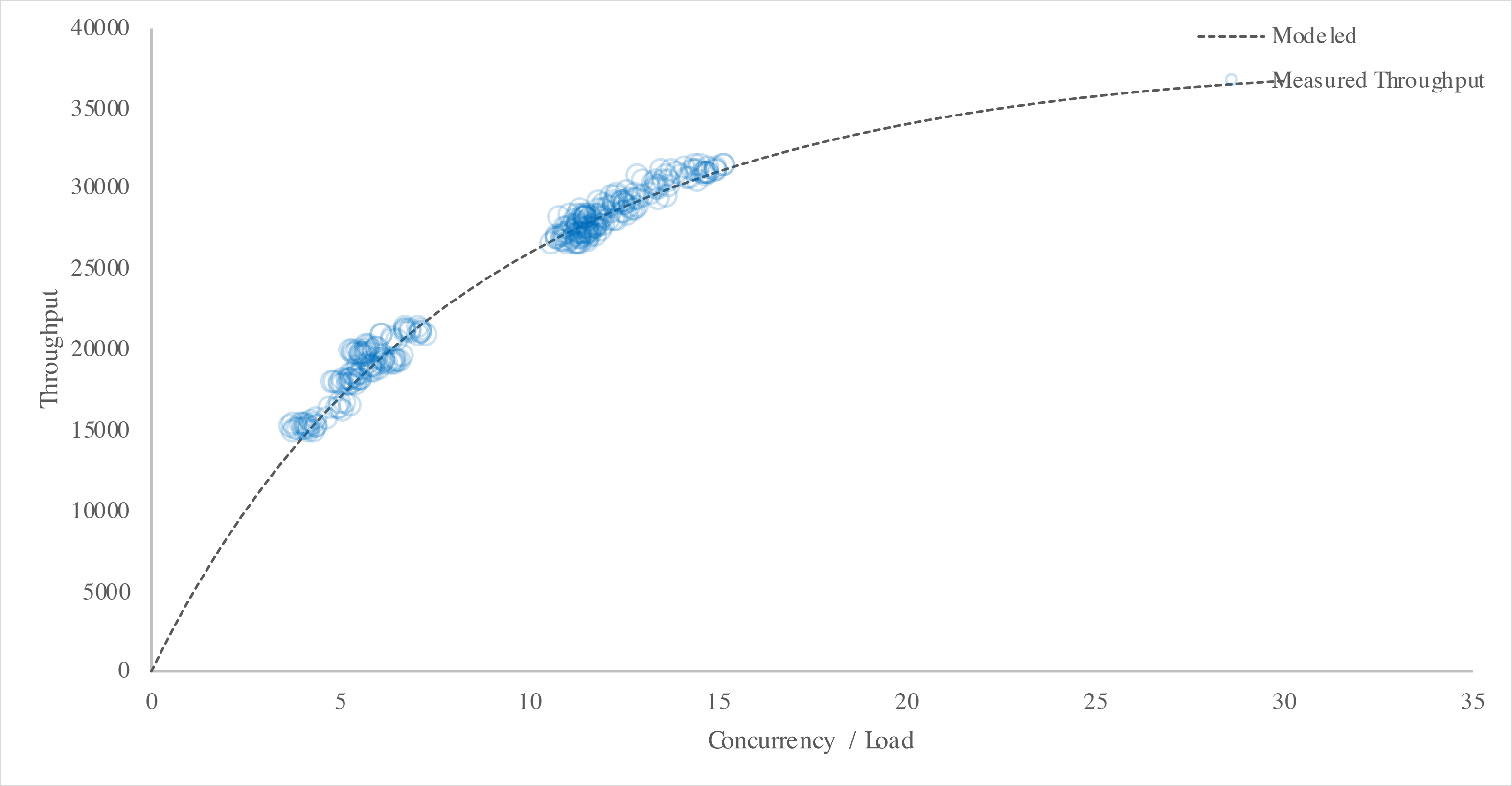
This is the USL's input and output. Is it linear?

It Looks Highly Linear, Doesn't It?



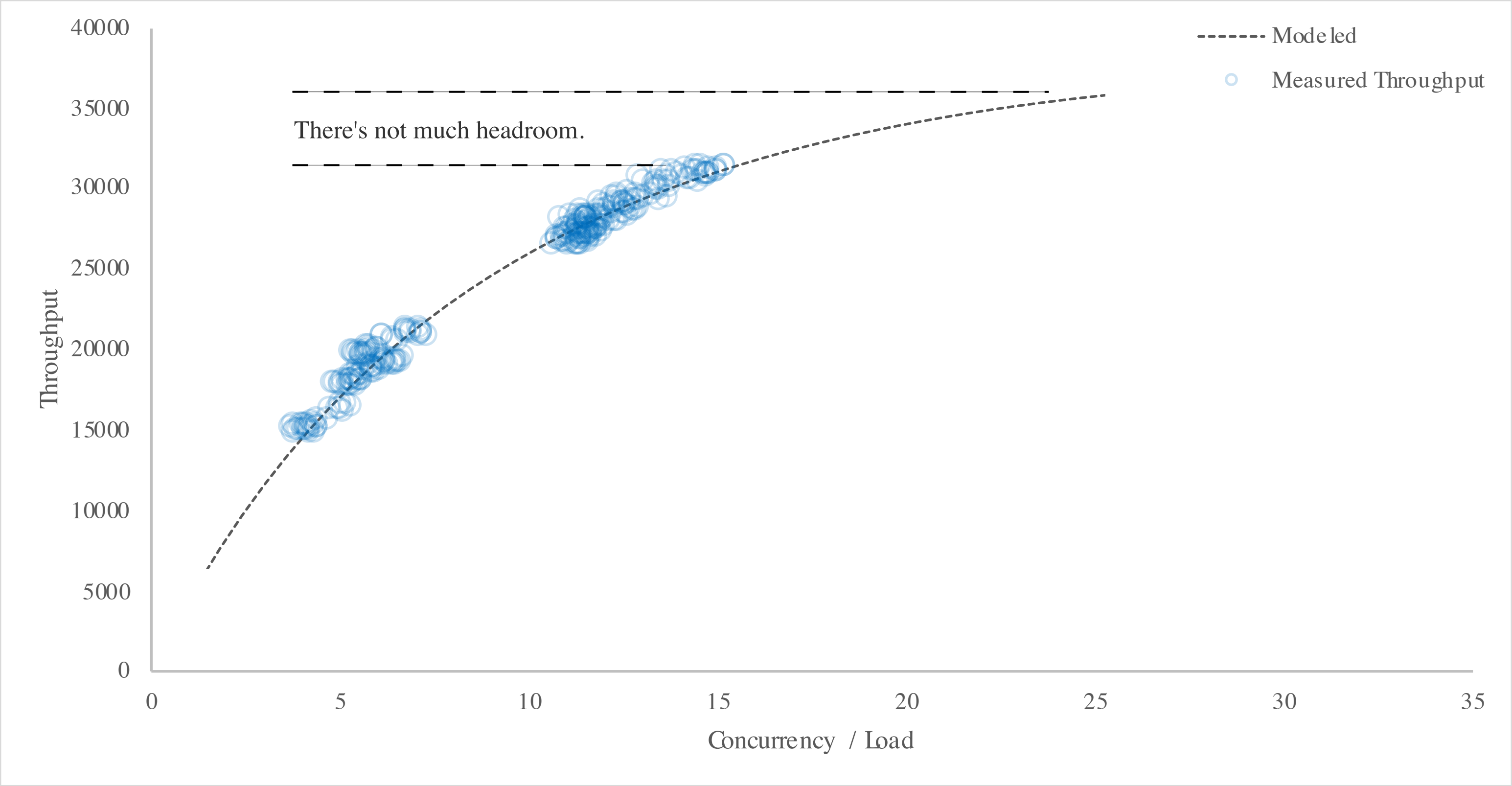
Don't celebrate yet.

Fit the USL Equation with Regression



Now the picture looks totally different!

How Much Headroom Does This System Have?



Just by looking, you can tell this system has maybe 10–15% more to give.

Profit???

In Which We Do The Impossible

What is the System's Primary Bottleneck?

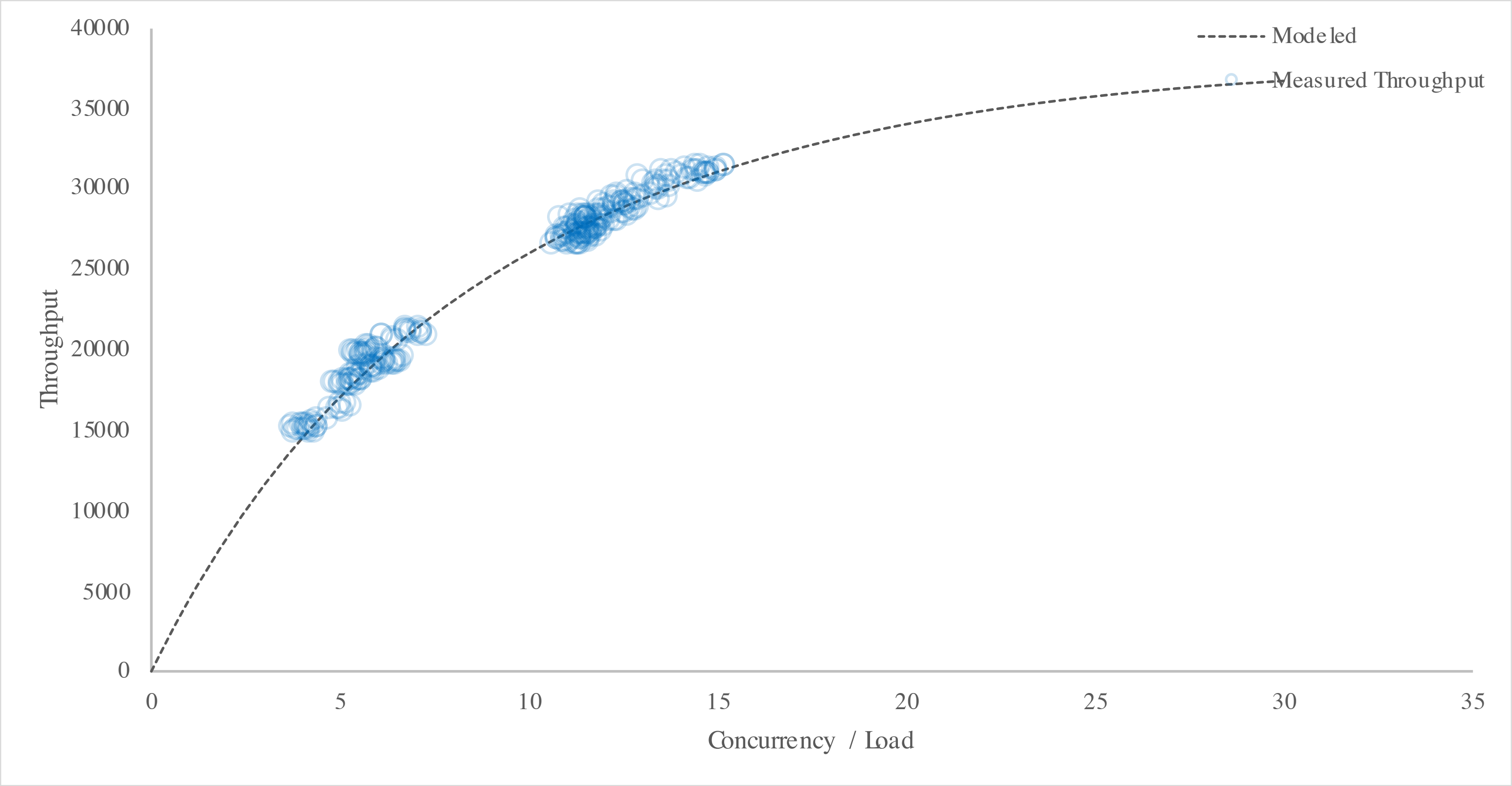
The regression gives estimates of the USL parameters.

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

The parameters have **physical meaning**.

- γ is the throughput of single-threadedness.
- σ is the fraction that's serialized/queued.
- κ is the fraction that's crosstalk/coherency.

This System Is Sublinear Because Of Queueing



σ = 7.4%, κ = 0.1%


WH
.GOV
PRESIDENT OF THE UNITED

Slides and Contact Information

Slides are at https://www.xaprb.com/talks/ or you can scan the QR code.

Contact: baron@vividcortex.com, @xaprb

Further Reading & References

- Neil Gunther, author of the USL.
- My USL book.
- My USL Excel workbook.
- Eben Freeman's LISA17 talk and slides
- Kavya Joshi's QCon talk
- There are lots of good books on queueing theory and scalability from Neil Gunther, Mor Harchol-Balter, Gross & Harris, etc