Stop Guessing, GET Requesting

Back of the envelope estimations to get started with.

Scheduled Sync:

 -> Request -> Transform -> Stash in DB

-> Request

-> Transform

-> Stash in DB

Questions to answer:

* Concurrent requests?

* Single process?

* Bottlenecks?

Info to collect and estimates to make

* How many requests?

* avg response time?

* origin limits?

* Sync frequency?

* DB limitations?

How?

* Service Docs

* Existing metrics

* Comparables

* Simulation

* “Guesstimates”

Request Considerations

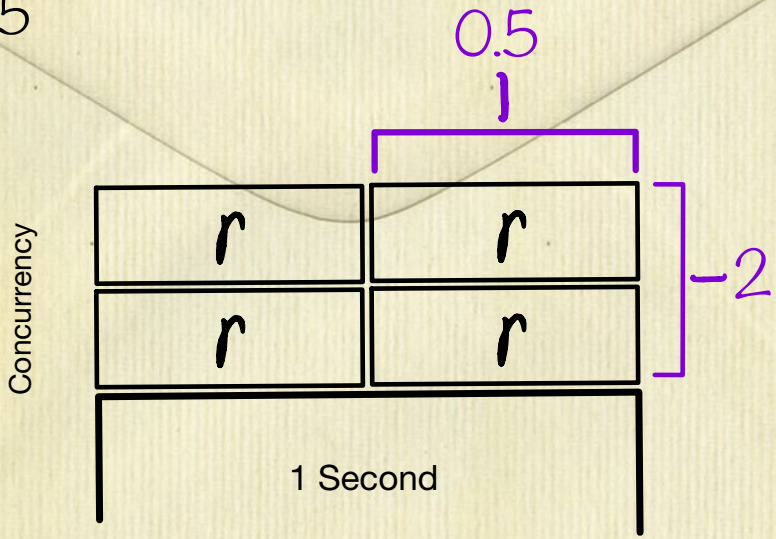
* Stay under rate limit

* Don’t kill the target service

Little's Law

C = RL

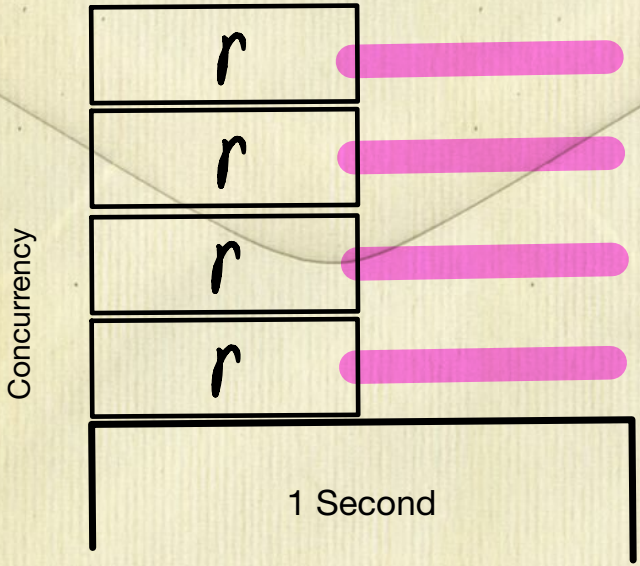
2 = 4 * 0.5



Little's Law

C = RL

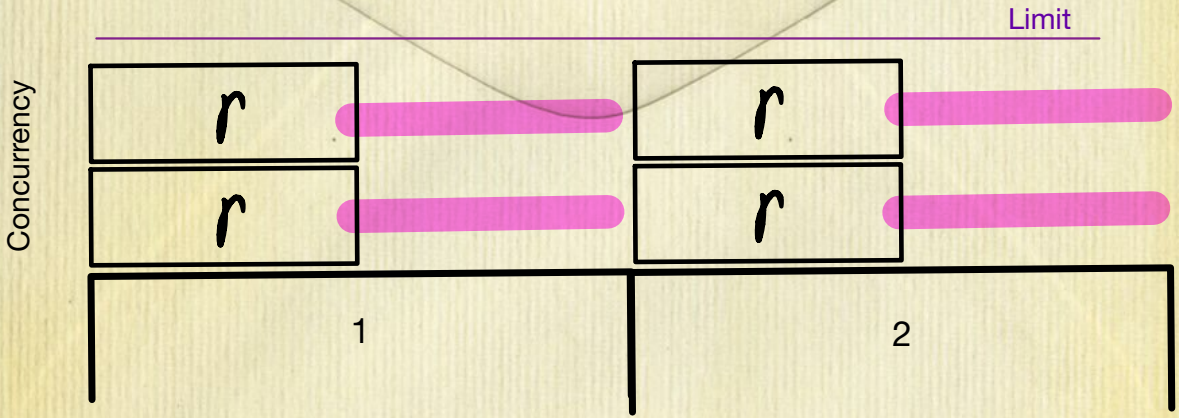
4↑ = 4 * 1↑

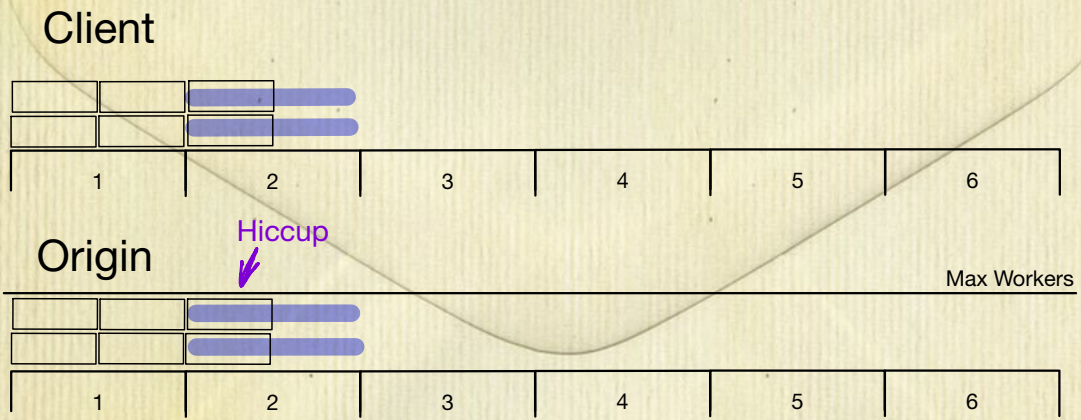


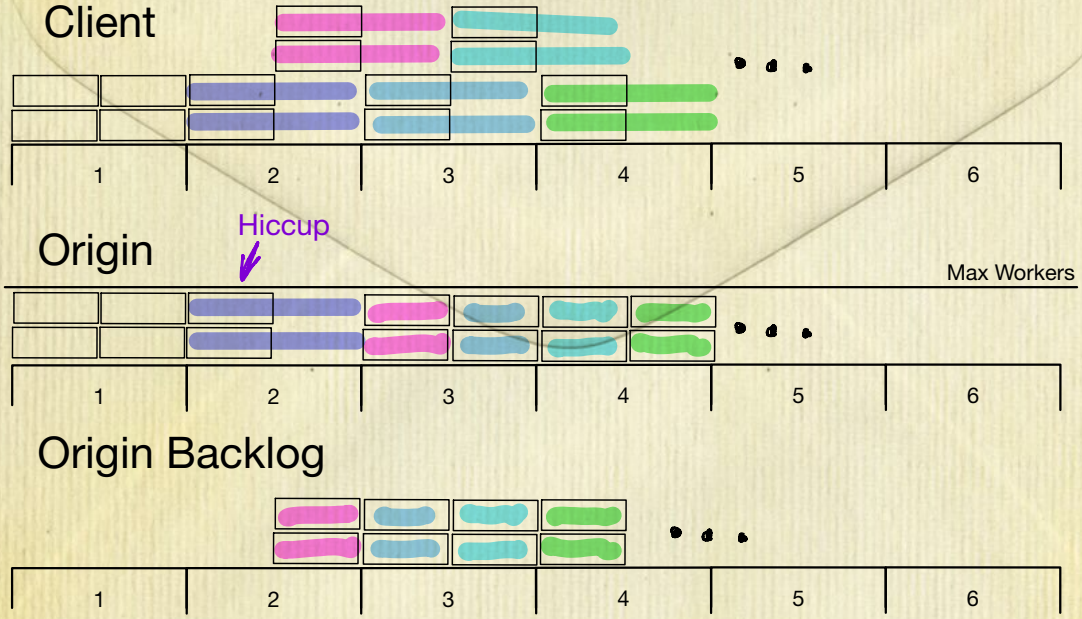
Little's Law

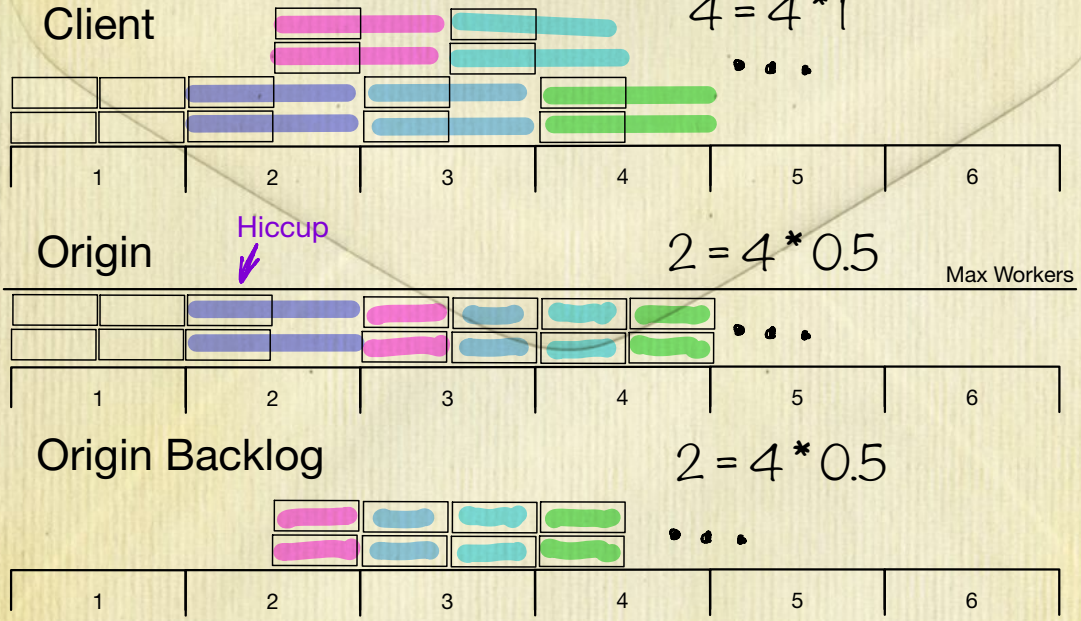
C = RL

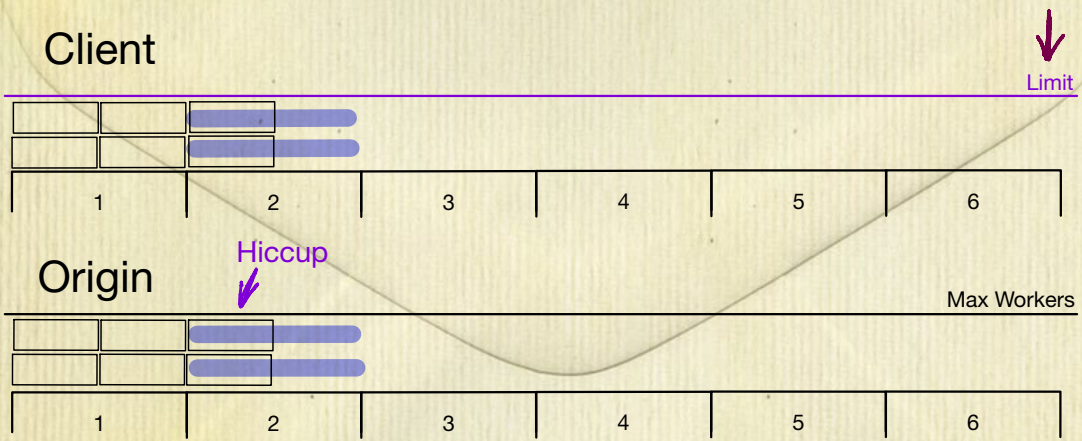
2 = 2↓ * 1↑

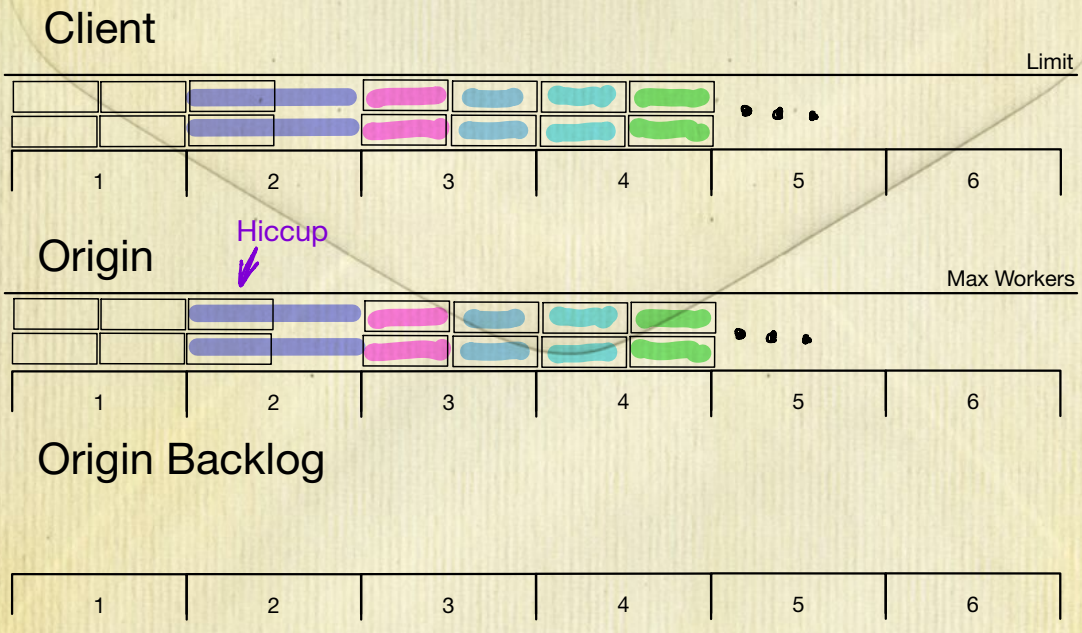


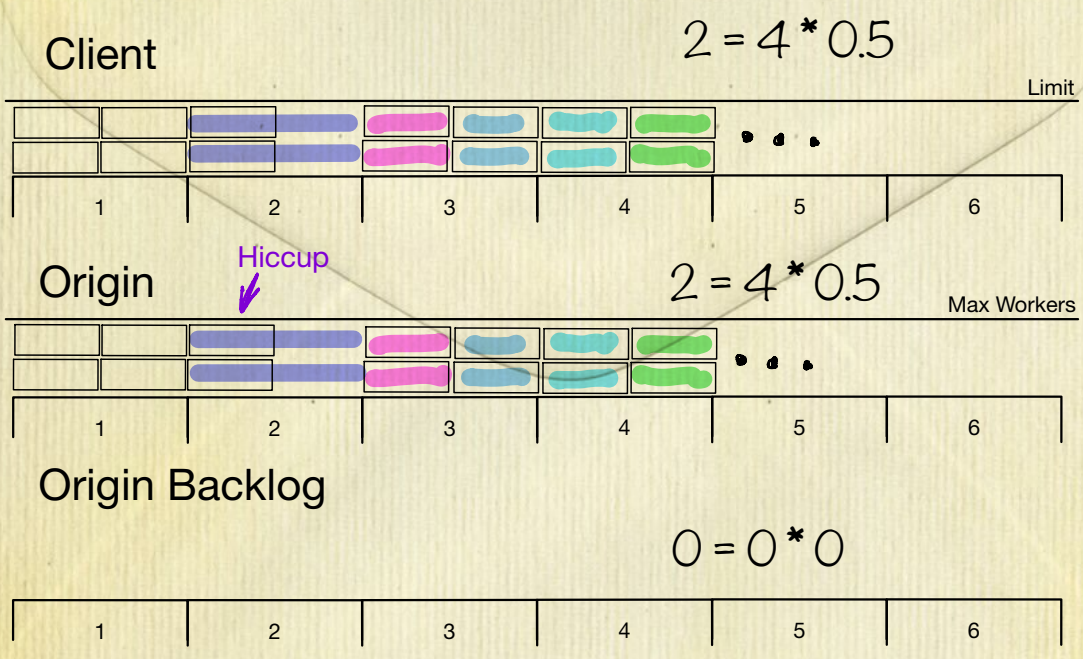

Client
1
2
3
4
5
6
Origin
Hiccup
Max Workers
1
2
3
4
5
6


Client
1
2
3
4
5
6
Origin
Hiccup
Max Workers
1
2
3
4
5
6
Origin Backlog
1
2
3
4
5
6

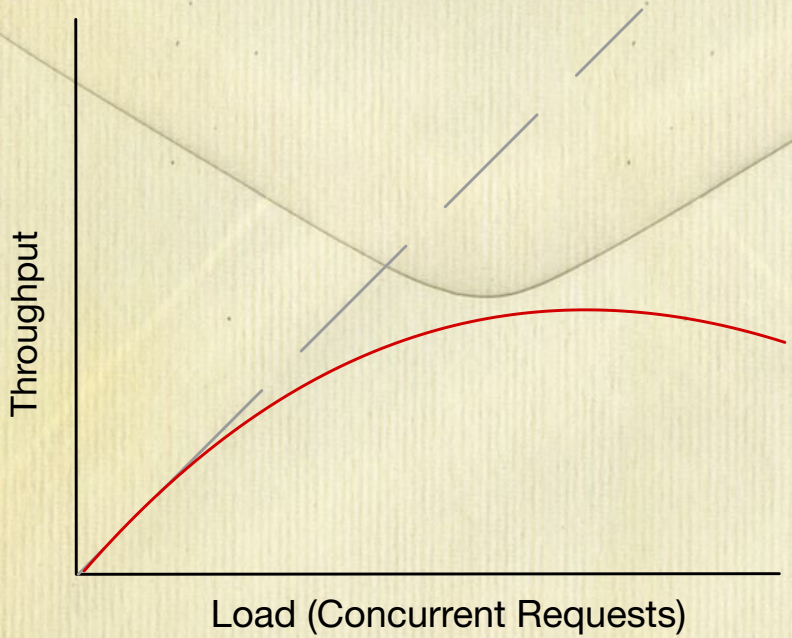

Client
4 = 4 * 1
1
2
3
4
5
6
Origin
Hiccup
2 = 4 * 0.5
Max Workers
1
2
3
4
5
6
Origin Backlog
2 = 4 * 0.5
1
2
3
4
5
6

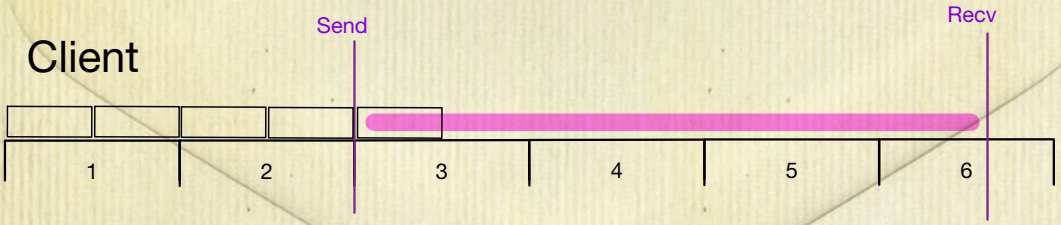

Client
Limit
1
2
3
4
5
6
Origin
Hiccup
Max Workers
1
2
3
4
5
6

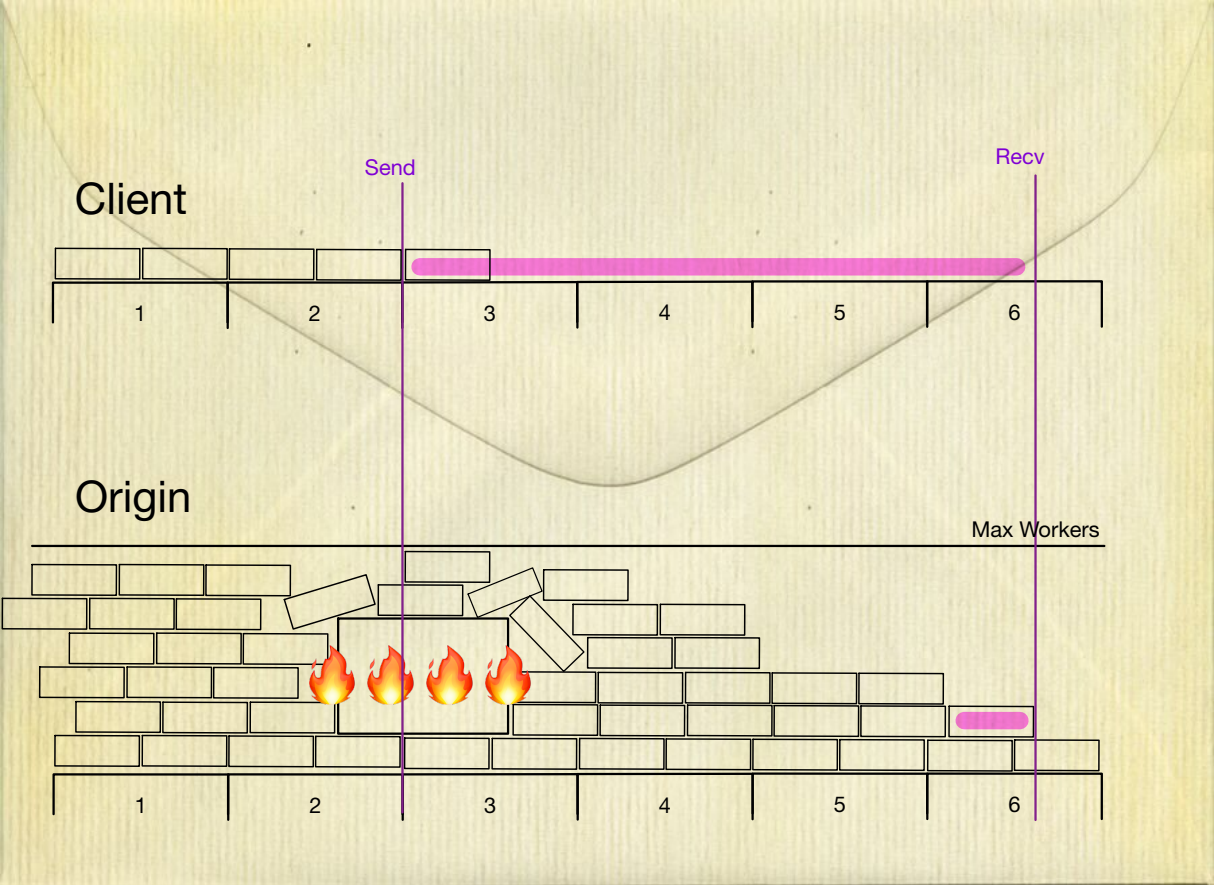

Client
Limit
1
2
3
4
5
6
Origin
Hiccup
Max Workers
1
2
3
4
5
6
Origin Backlog
1
2
3
4
5
6


Client
2 = 4 * 0.5
Limit
1
2
3
4
5
6
Origin
Hiccup
2 = 4 * 0.5
Max Workers
1
2
3
4
5
6
Origin Backlog
0 = 0 * 0
1
2
3
4
5
6

Why we care




Client
Send
Recv
1
2
3
4
5
6

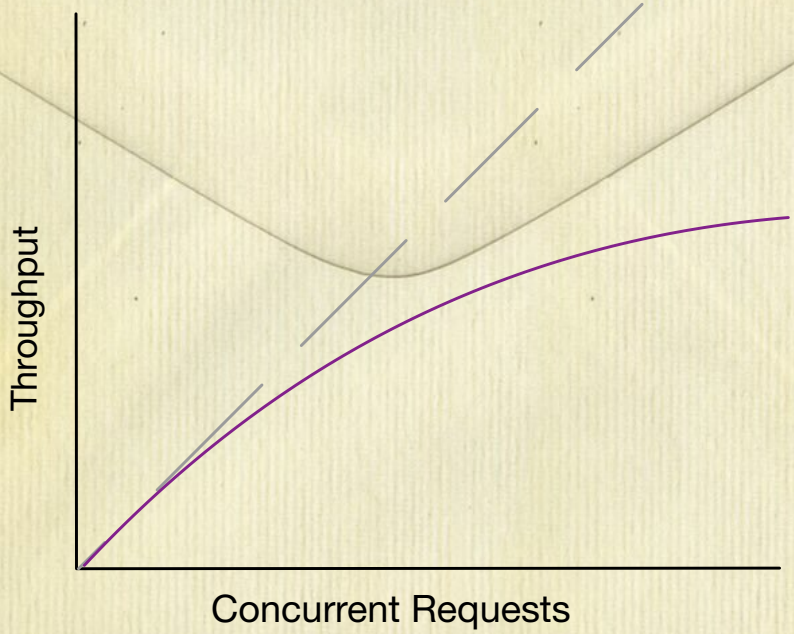

Send
Recv
Client
1
2
3
4
5
6
Origin
Max Workers
1
2
3
4
5
6

Back to our Scheduled Sync Task

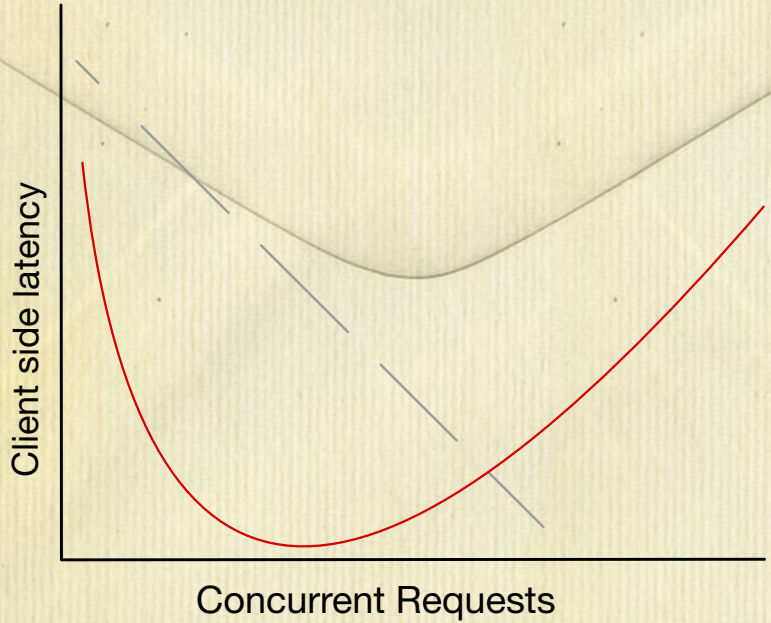
1,800,000 requests to be made.

1,000 * 0.5 = 500 <- Little's Law

…but, diminishing returns



…because



We need more GIL's

1 / 0.01 = 100 p/s

100 < 1,000

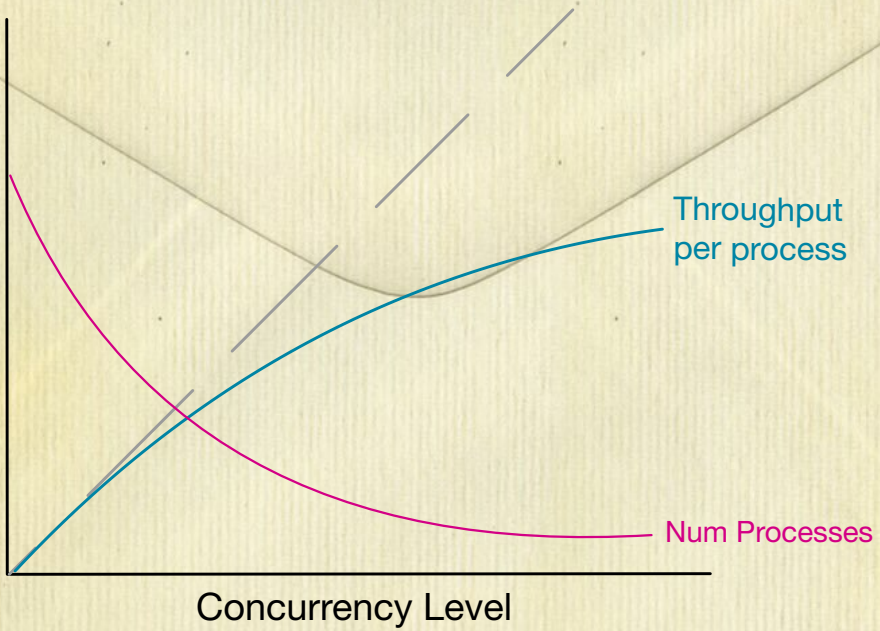
Amdahl's Law

G: processing time averaged per request
T: target rate

$R(C) = 1 / (G + (L / C))$

Calculating number of processes needed.
$P(C, T) = T / R(C)$

Amdahl's Law



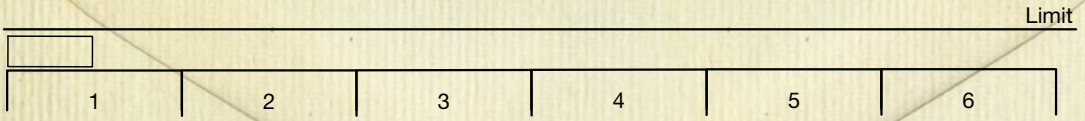
R(50) -> 50 requests/s

1,000 / 50 = 20

What about the DB?

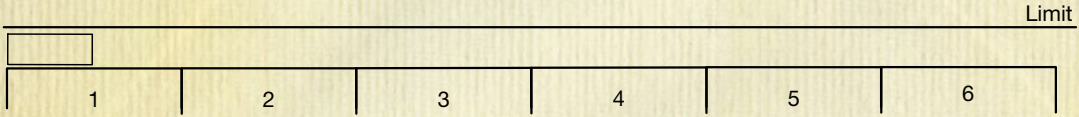
100 rows per request
0.5 response time.

Client A

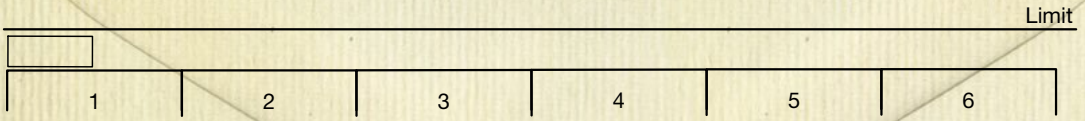


Client B

Client C

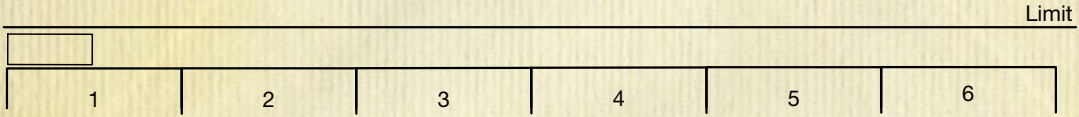


Client A

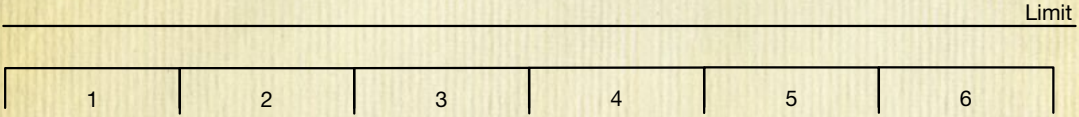


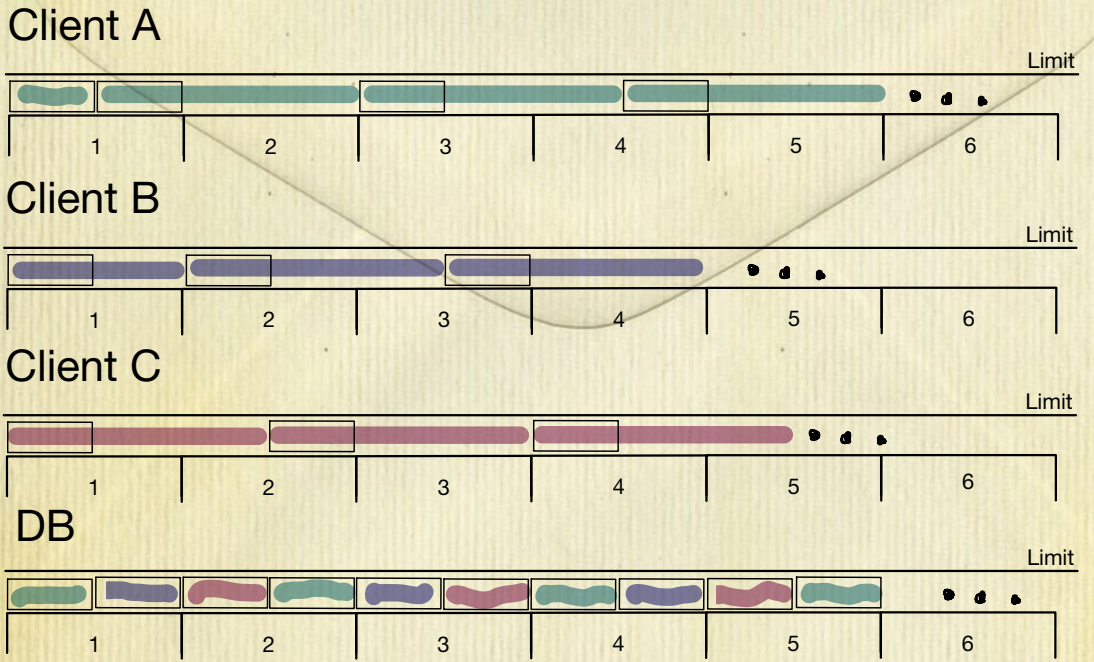
Client B

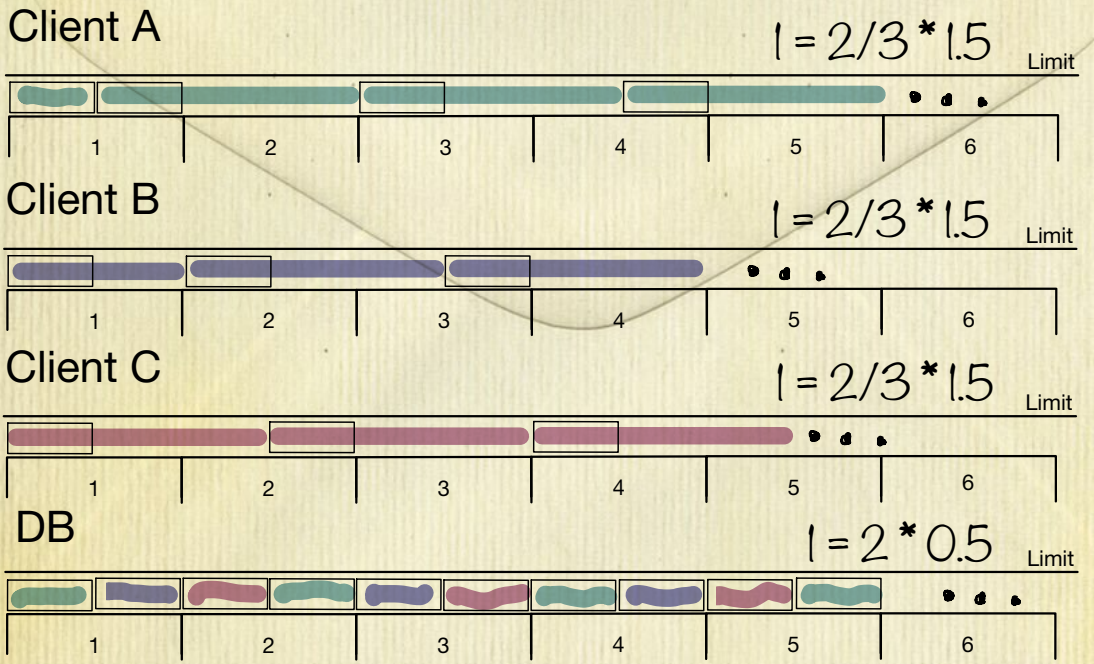
Client C



DB




Client A
Limit
1
2
3
4
5
6
Client B
Limit
1
2
3
4
5
6
Client C
Limit
1
2
3
4
5
6
DB
Limit
1
2
3
4
5
6


Client A
1 = 2/3 * 1.5
Limit
1
2
3
4
5
6
Client B
1 = 2/3 * 1.5
Limit
1
2
3
4
5
6
Client C
1 = 2/3 * 1.5
Limit
1
2
3
4
5
6
DB
1 = 2 * 0.5
Limit
1
2
3
4
5
6

1 = 2 * 0.5

2 * 100 = 200 rows/second

200 < 1,000

The DB is a bottleneck

Do we care? Enough?

1,800,000 / 200 / 60 / 60 = ~2.5 hrs

Other considerations

* DB connection limits (e.g. PostgreSQL sans PGBouncer)

What if tasks overlap?

Things to note

* Only as good as the inputs

* Model a distribution of possible inputs

* Just one example.

Key Takeaway

1. Resource contention -> queuing -> Queuing theory

2. Metrics and models

GitHub: steven-cutting

References:

* "Stop Rate Limiting! Capacity Management Done Right" by Jon Moore, StrangeLoop 2017
* Amdahl's Law, Jakob Jenkov, jenkov.com
* Probability and Statistics with Reliability, Queuing, and Computer Science Applications, Kishor S. Trivedi