The 5 Stages of Scale

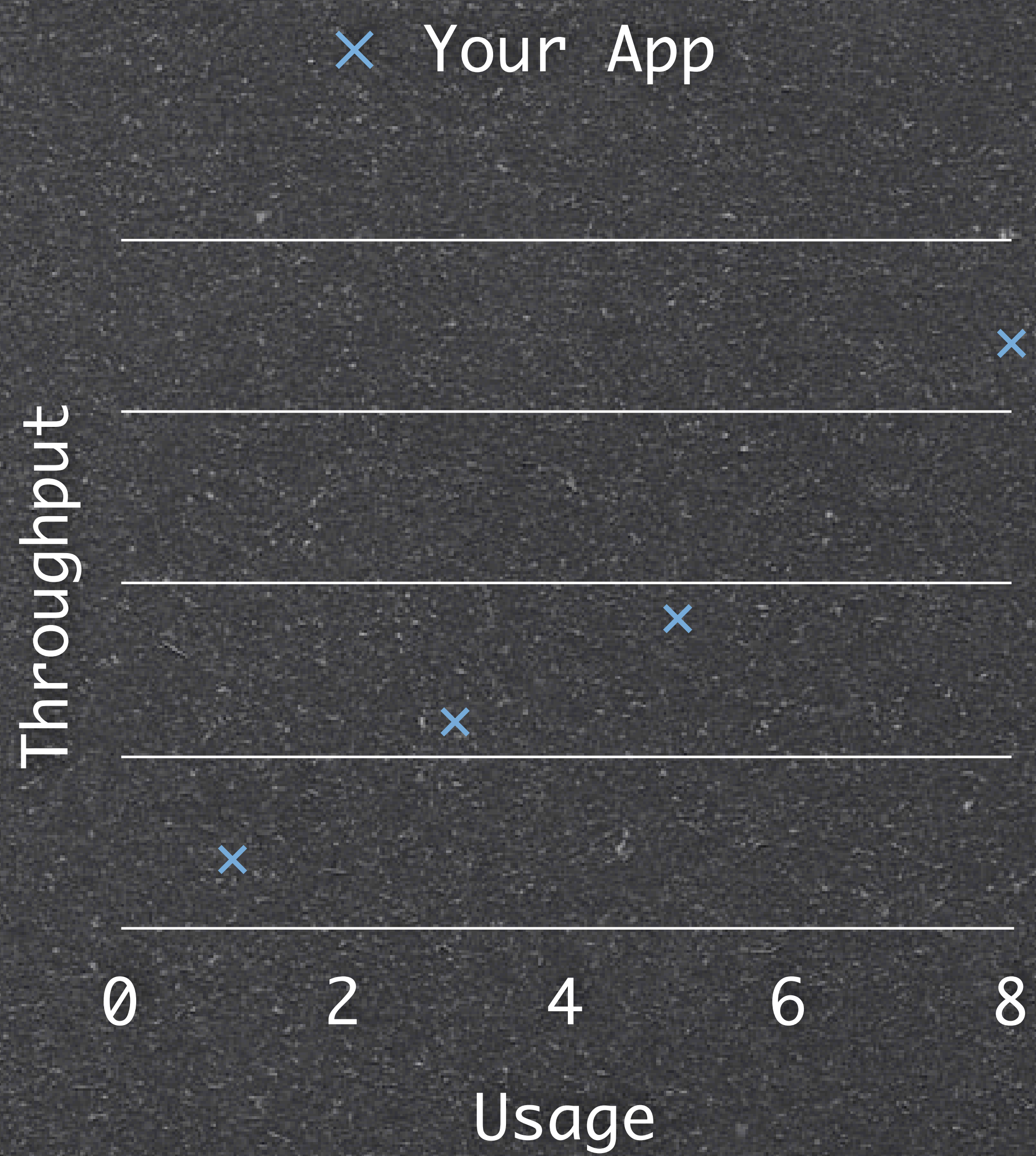
Christopher Smith

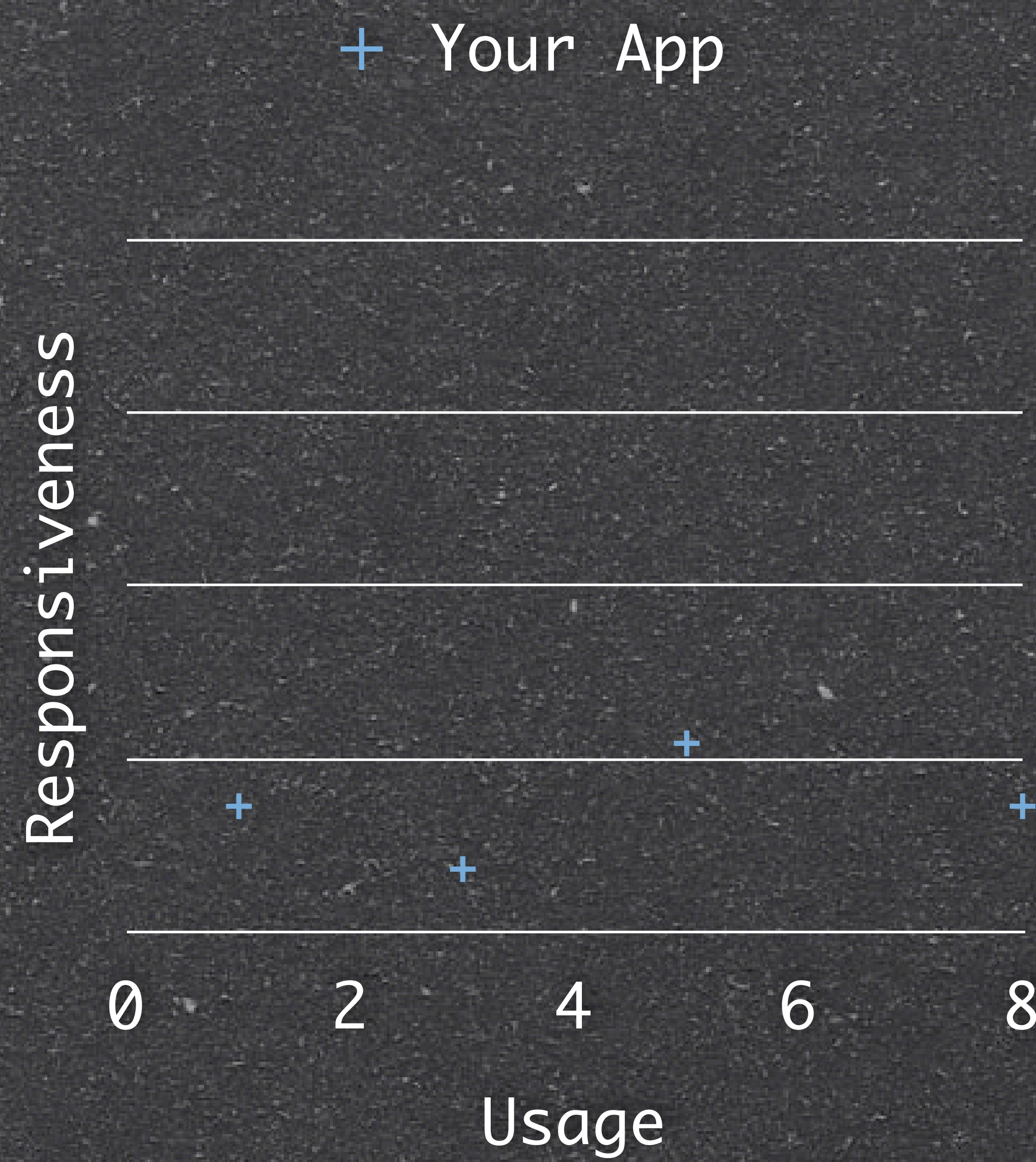
Who am I?

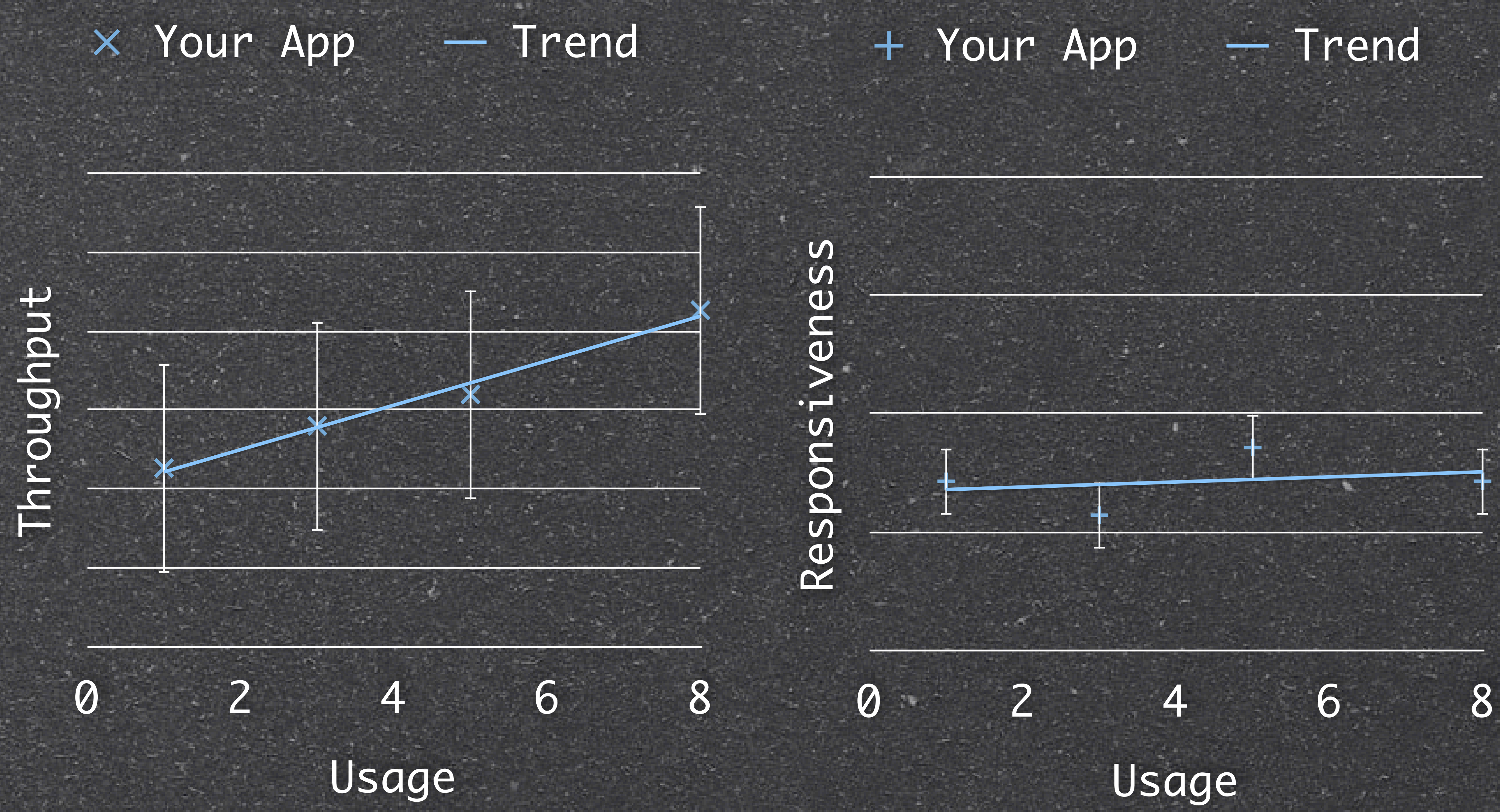
- Two decades experience
- Half of that in online advertising
- Internet systems engineering
- Scaling web serving, data collection & analysis
- Places big & small.

Scalability

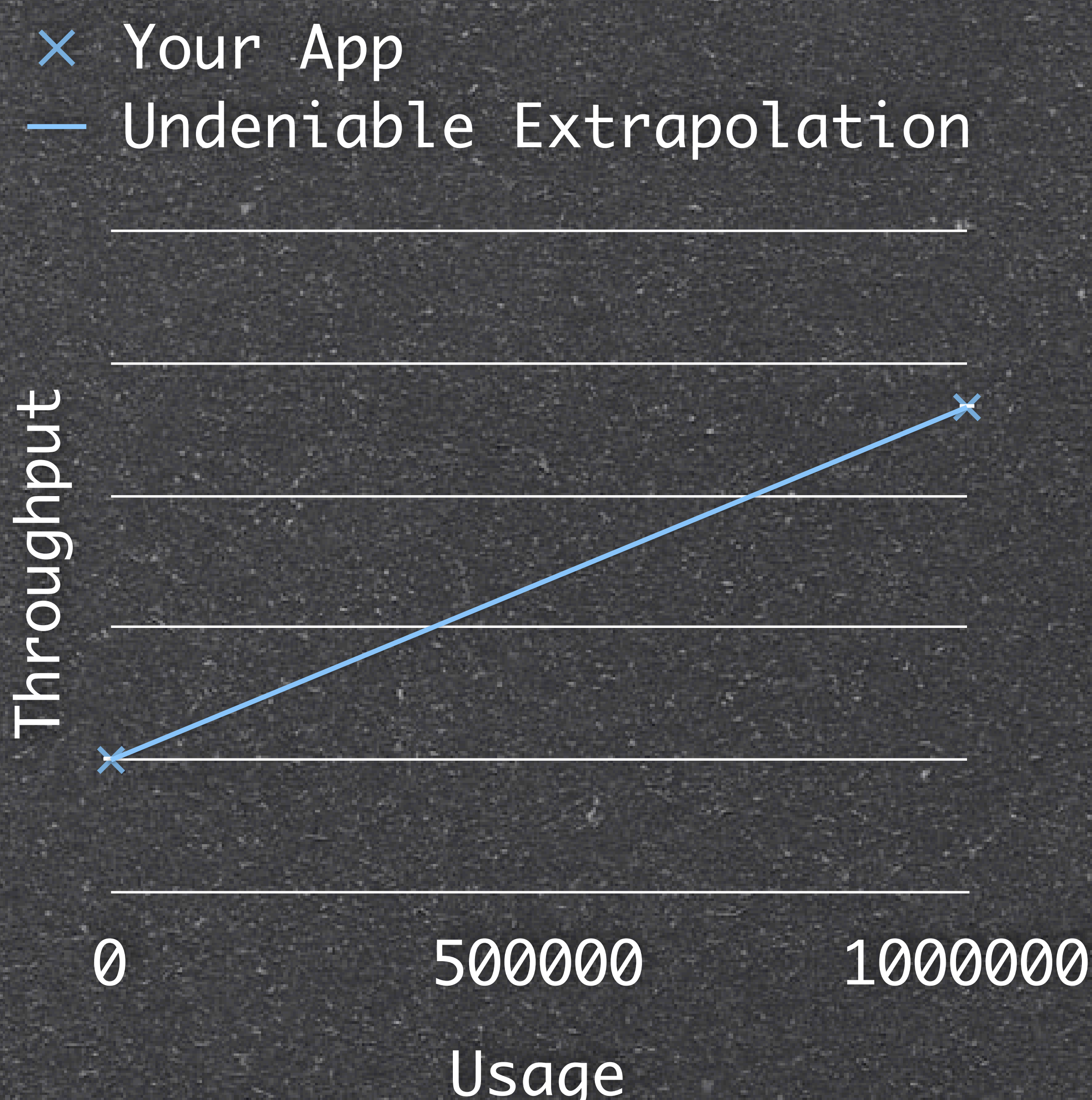
- scale: *v.tr.*
  - 1. To clear or strip of scale or scales.
  - 2. Weigh a specified weight.
  - 3. Climb up or over (something steep)

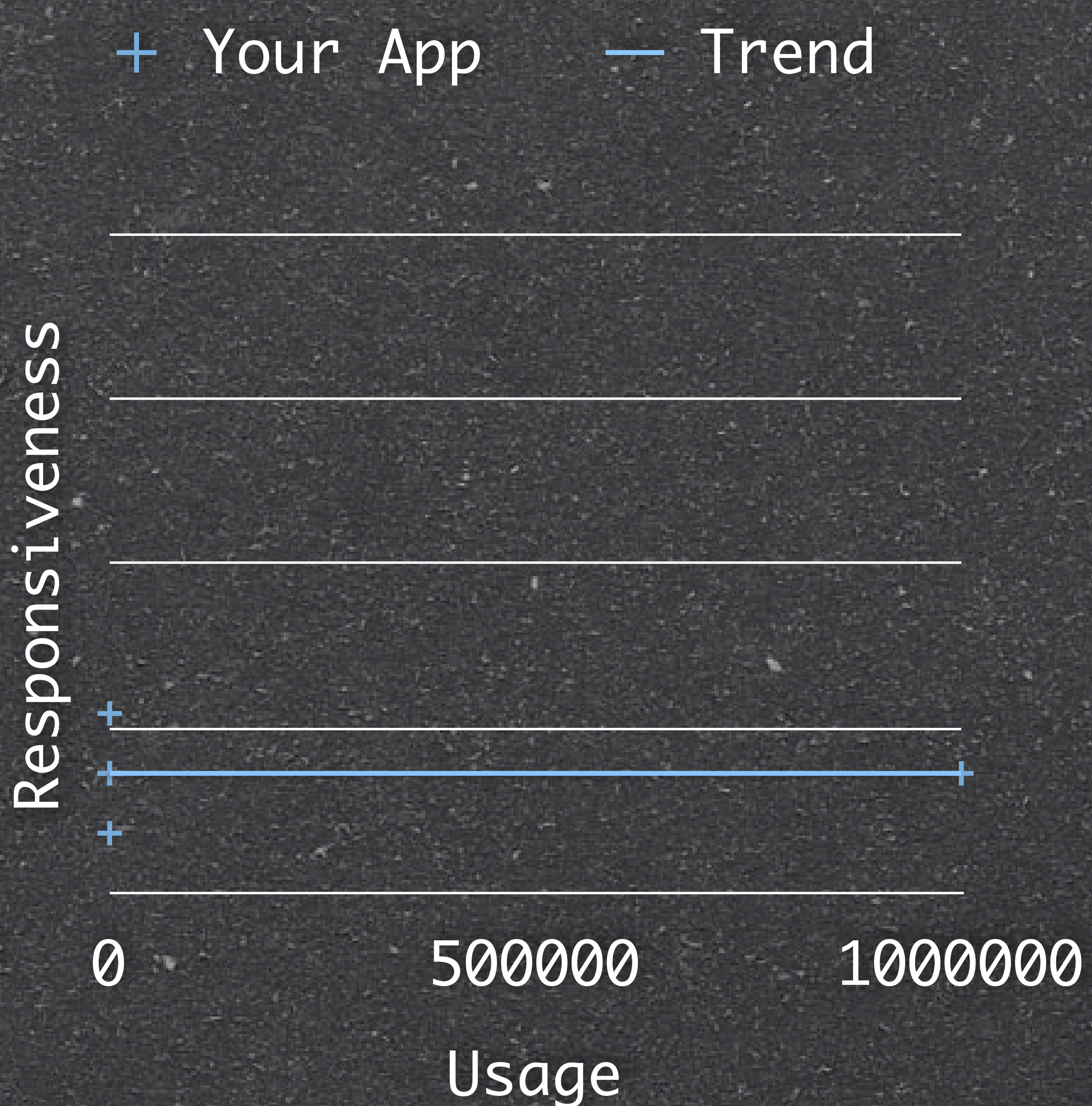

Your App
Throughput
0
2
4
6
8
Usage


Your App
Responsiveness
0
2
4
6
8
Usage



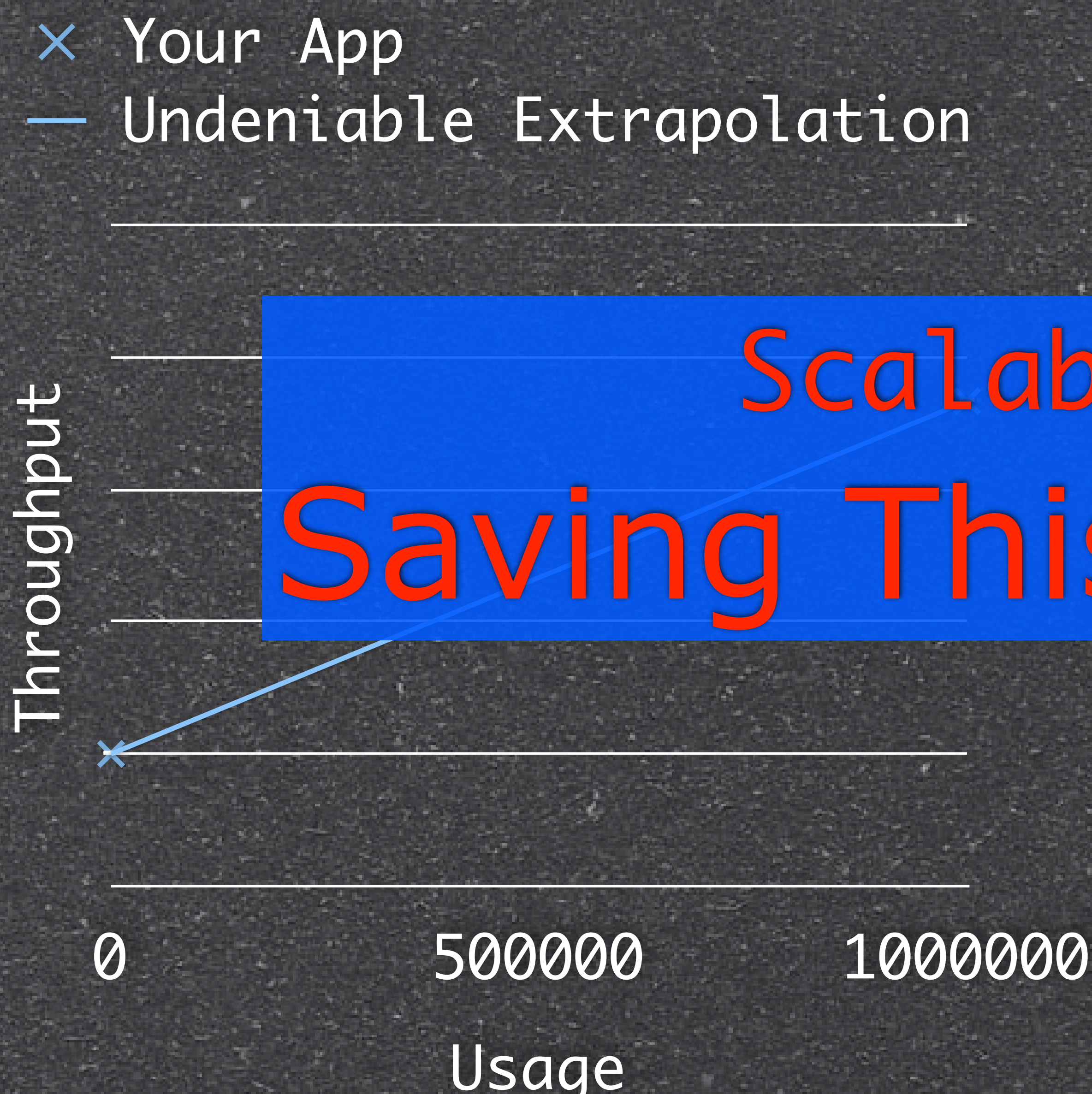
Your App
Trend
Throughput
0
2
4
6
8
Usage
Your App
Trend
Responsiveness
0
2
4
6
8
Usage

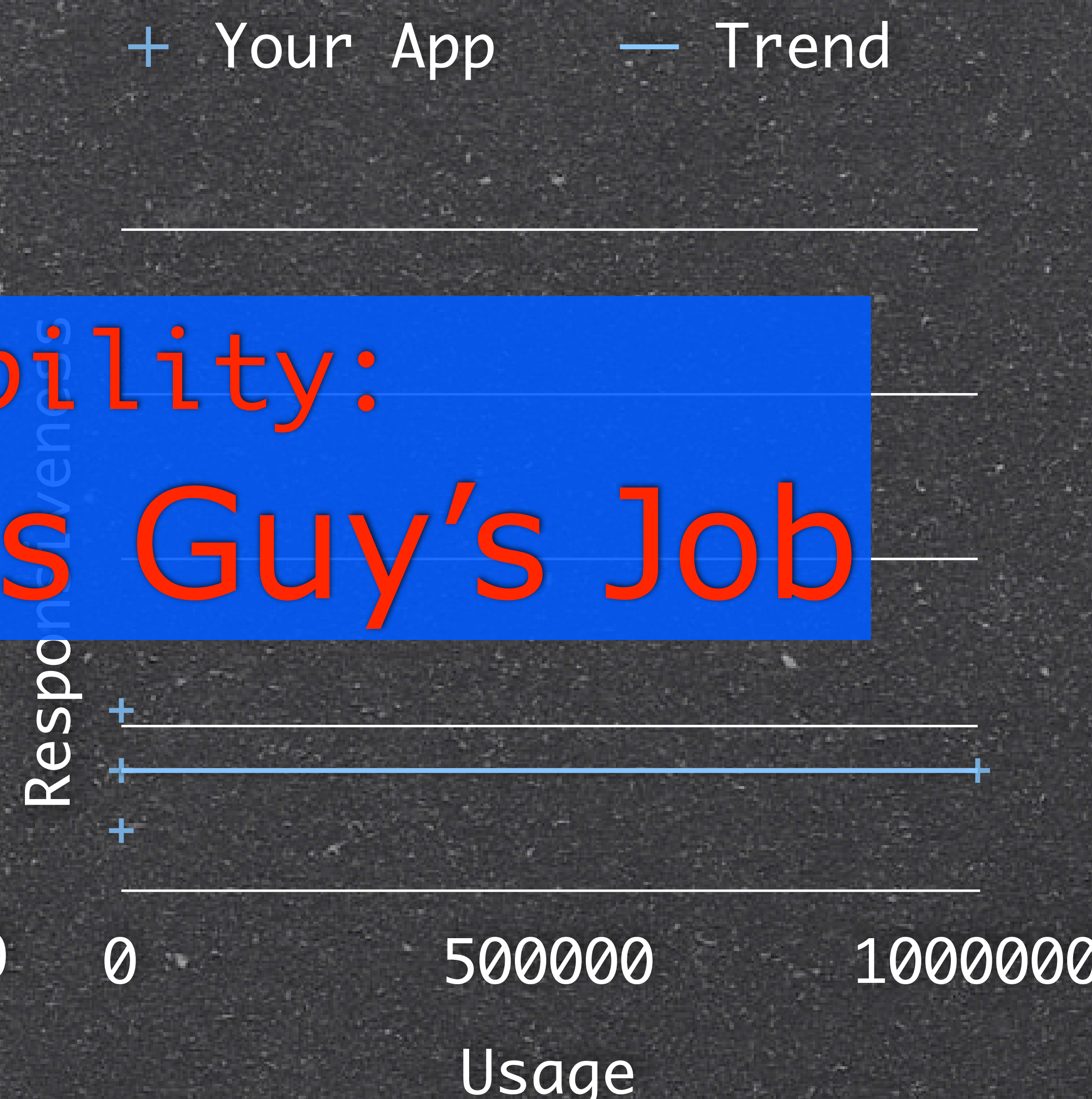

Your App
Undeniable Extrapolation
Throughput
0
500000
1000000
Usage


Your App
Trend
Responsiveness
0
500000
1000000
Usage

Scalability:

Saving This Guy’s Job





Scalability Envelopes

- There is always a “next” bottleneck.
- In case of scalability problem...
- 6 envelopes

Envelope 0

- Session partitioning
- Commodity: load balancer, multi-*
- Linear scale for CPU
- Limit: C10K?

Envelope 1

- Read Caching
  - Reverse-proxy
  - memcached
  - CDN
- log(n) scale: thank you Zipf
- Limit: ~200 w/sec

Envelope 2

- Get a real persistence framework
  - Data structures FTW!
  - DB: concurrent read/write
  - MOM: queuing/event IO/TP monitors
  - Cheat on ACID (particularly C & D)
- log(n) scale?
- 1000-10000 w/sec

Tipping over

Scaling Catamaran's

- RAM caching I/O
- RAID
- Threads (sometimes)
- Packet loss (UR DUING IT WRONG)
- SSD's?

Jeff Dean's Numbers

- Latency Comparison Numbers
- ---------------------------

L1 cache reference	0.5	ns			
Branch mispredict	5	ns			
L2 cache reference	7	ns			14x L1 cache
Mutex lock/unlock	25	ns			
Main memory reference	100	ns			20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy	3,000	ns			
Send 1K bytes over 1 Gbps network	10,000	ns	0.01	ms	
Read 4K randomly from SSD*	150,000	ns	0.15	ms	
Read 1 MB sequentially from memory	250,000	ns	0.25	ms	
Round trip within same datacenter	500,000	ns	0.5	ms	
Read 1 MB sequentially from SSD*	1,000,000	ns	1	ms	4X memory
Disk seek	10,000,000	ns	10	ms	20x datacenter roundtrip

Problem: IO latency

- Throughput: 2x every 18 months
- Latency:
  - CPU: <2x every 18 months
  - LAN network: 2x every 2-3 years
  - Memory: 2x every 3-5 years
  - Disk: 2x every decade? (SSD?)
  - WAN Network: 1x every...

Problem IO Latency

- Traditional indexes on the wrong side
  - Turns a scan in to a seek
  - Index lookup: scan 0.1% of records + 1 random seek
  - Scan: scan 100% of records, 0 random seek
  - Seek is 10ms & Scan is 100Hz -> 10x win
  - Seek is 1ms & Scan is 1GHz -> 1000x loss

Envelope 3

- Real partitioning of IO
- Move code, not data
- Commodities: Map/Reduce (Hadoop), DHT (Cassandra, HBase, Riak)
- CAP Theory limiting sync'ing

Envelope 4

- Route new data through data partitions
  - Using MOM/EventIO “the right way”
  - ESP/CEP: Eigen, Storm, Esper, StreamBase, 0mq, etc.

Envelope 5

- Cheat more on reliability.
  - UDP w/o reliability > TCP
  - Measure loss vs. prevent loss
  - Horseshoes, hand grenades, features...?

Integrated Systems

- Combined IO management solutions:
  - real-time memory key/value lookup
  - LSM + bitmap indexes + etc.
  - eventual consistency
  - mobile code for batch processing
- Cassandra, HBase, etc.

Efficient Logging

- Events in efficient machine parseable form: (protobuf, thrift, etc.)
- Event source writes only to NIC
- UDP Multicast
- Redundant listeners

```
message LogEvent {
    required uint64 pid = 1;
    optional uint64 tid = 2;
    optional uint64 sid = 4;
    required uint64 sequence = 5;
    required uint64 timestamp = 6;
    enum Level { PANIC = 0, ERROR = 1..}
    required Level level = 7;
    required bytes payload = 8;
}
```

Announcements

- Dedicated channel.
- Payload: channel IP, channel port, last seq, pid, tid, sid + stats
- All announcers listen and self-throttle.
- Directory service accumulates

Consolidation

- Redundant journalers (RAID)
- ESP: detect loss in real time window
- If necessary, Map/Reduce processing to try to resolve partial loss.

Efficiency

- Hundreds of nodes
- >50MB/sec
- >50,000 pps
- 3-4 “journalers” resolving data
- >5TB reconciled data a day
- <0.1% data loss

Envelope 6

- Take out 6 envelopes...