What We Got Wrong

Lessons from the Birth of Microservices at Google

November 6, 2018

LIGHTSTEP

Part One: The Setting

Still betting big on the
Google Search Appliance

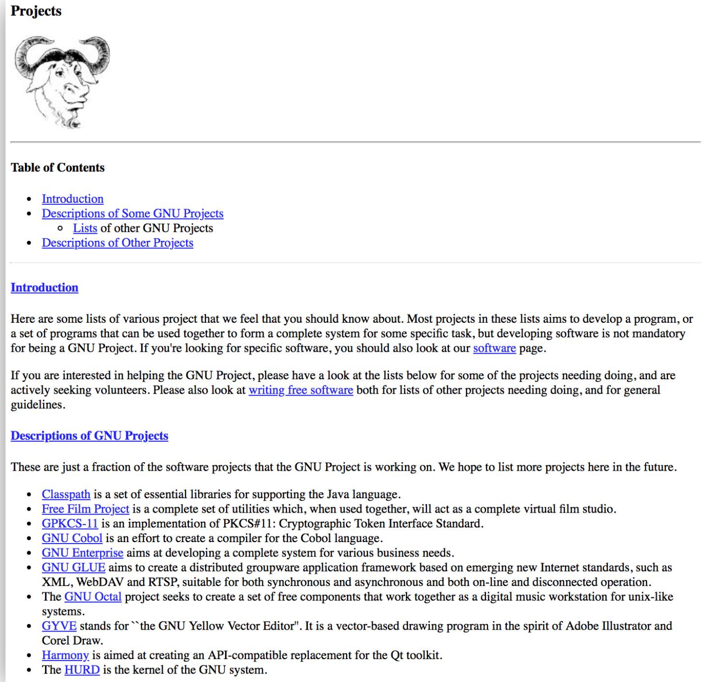
“Those Sun boxes are so expensive!”

“Those linux boxes are so unreliable!”

“Let’s see what’s on GitHub first…”

– literally nobody in 2001

“GitHub” circa 2001



Projects

Table of Contents

- Introduction
- Descriptions of Some GNU Projects
  - Lists of other GNU Projects
- Descriptions of Other Projects

Introduction

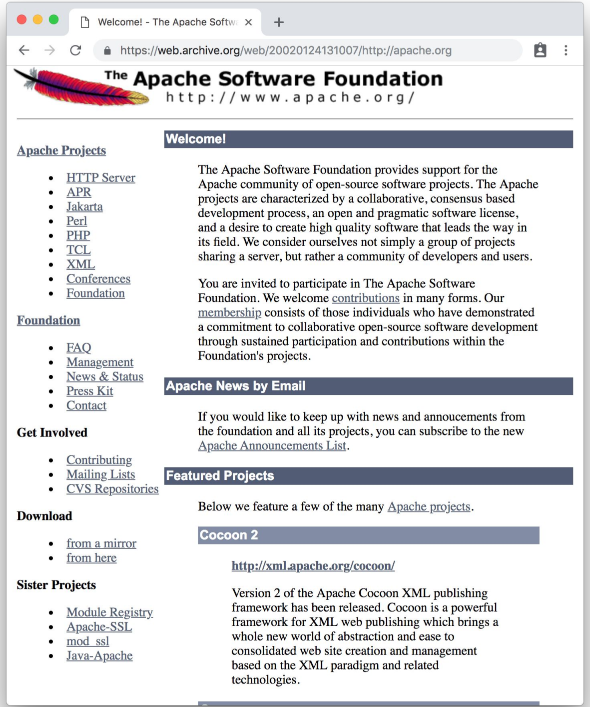
Here are some lists of various project that we feel that you should know about. Most projects in these lists aims to develop a program, or a set of programs that can be used together to form a complete system for some specific task, but developing software is not mandatory for being a GNU Project. If you're looking for specific software, you should also look at our software page.

If you are interested in helping the GNU Project, please have a look at the lists below for some of the projects needing doing, and are actively seeking volunteers. Please also look at writing free software both for lists of other projects needing doing, and for general guidelines.

Descriptions of GNU Projects

These are just a fraction of the software projects that the GNU Project is working on. We hope to list more projects here in the future.

- Classpath is a set of essential libraries for supporting the Java language.
- Free Film Project is a complete set of utilities which, when used together, will act as a complete virtual film studio.
- GPKCS-11 is an implementation of PKCS#11: Cryptographic Token Interface Standard.
- GNU Cobol is an effort to create a compiler for the Cobol language.
- GNU Enterprise aims at developing a complete system for various business needs.
- GNU GLUE aims to create a distributed groupware application framework based on emerging new Internet standards, such as XML, WebDAV and RTSP, suitable for both synchronous and asynchronous and both on-line and disconnected operation.
- The GNU Octal project seeks to create a set of free components that work together as a digital music workstation for unix-like systems.
- GYVE stands for ``the GNU Yellow Vector Editor''. It is a vector-based drawing program in the spirit of Adobe Illustrator and Corel Draw.
- Harmony is aimed at creating an API-compatible replacement for the Qt toolkit.
- The HURD is the kernel of the GNU system.



Welcome! - The Apache Softw ×

https://web.archive.org/web/20020124131007/http://apache.org

The Apache Software Foundation
http://www.apache.org/

Apache Projects

- HTTP Server
- APR
- Jakarta
- Perl
- PHP
- TCL
- XML
- Conferences
- Foundation

Foundation

- FAQ
- Management
- News & Status
- Press Kit
- Contact

Get Involved

- Contributing
- Mailing Lists
- CVS Repositories

Download

- from a mirror
- from here

Sister Projects

- Module Registry
- Apache-SSL
- mod_ssl
- Java-Apache

Welcome!

The Apache Software Foundation provides support for the Apache community of open-source software projects. The Apache projects are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field. We consider ourselves not simply a group of projects sharing a server, but rather a community of developers and users.

You are invited to participate in The Apache Software Foundation. We welcome contributions in many forms. Our membership consists of those individuals who have demonstrated a commitment to collaborative open-source software development through sustained participation and contributions within the Foundation's projects.

Apache News by Email

If you would like to keep up with news and annoucements from the foundation and all its projects, you can subscribe to the new Apache Announcements List.

Featured Projects

Below we feature a few of the many Apache projects.

Cocoon 2

http://xml.apache.org/cocoon/

Version 2 of the Apache Cocoon XML publishing framework has been released. Cocoon is a powerful framework for XML web publishing which brings a whole new world of abstraction and ease to consolidated web site creation and management based on the XML paradigm and related technologies.

LIGHTSTEP

Engineering constraints

- Must DIY:
    - Wacky scaling requirements
    - Utter lack of alternatives
- Must scale horizontally
- Must build on commodity hardware that fails often

Google eng cultural hallmarks, early 2000s

- Intellectually rigorous
- Bottoms-up eng decision-making
- Aspirational
- … and maybe a little overconfident :-/

Part Two: What Happened

Cambrian Explosion of Infra Projects

Eng culture idolized epic infra projects (for good reason):

- GFS
- BigTable
- MapReduce
- Borg
- Mustang (web serving infra)
- SmartASS (ML-based ads ranking+serving)

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.

In this paper, we present file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-benchmarks and real world use.

Categories and Subject Descriptors

D [4]: 3—*Distributed file systems*

General Terms

Design, reliability, performance, measurement

Keywords

Fault tolerance, scalability, data storage, clustered storage

*The authors can be reached at the following addresses: {*sanjay,hgobioff,shuntak*}*@google.com*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA.
Copyright 2003 ACM 1-58113-757-5/03/0010 ...$5.00.

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their current failures. We have seen problems caused by application bugs, operating system bugs, human errors, and the failures of disks, memory, connectors, networking, and power supplies. Therefore, constant monitoring, error detection, fault tolerance, and automatic recovery must be integral to the system.

Second, files are huge by traditional standards. Multi-GB files are common. Each file typically contains many application objects such as web documents. When we are regularly working with fast growing data sets of many TBs comprising billions of objects, it is unwieldy to manage billions of approximately KB-sized files even when the file system could support it. As a result, design assumptions and parameters such as I/O operation and block sizes have to be revisited.

Third, most files are mutated by appending new data rather than overwriting existing data. Random writes within a file are practically non-existent. Once written, the files are only read, and often only sequentially. A variety of data share these characteristics. Some may constitute large repositories that data analysis programs scan through. Some may be data streams continuously generated by running applications. Some may be archival data. Some may be intermediate results produced on one machine and processed on another, whether simultaneously or later in time. Given this access pattern on huge files, appending becomes the focus of performance optimization and atomicity guarantees, while caching data blocks in the client loses its appeal.

Fourth, co-designing the applications and the file system API benefits the overall system by increasing our flexibility.

Convergent Evolution?

Common characteristics of the most-admired projects:

- Identification and leverage of horizontal scale-points
- Factored-out user-level infra (RPC, discovery, load-balancing, eventually tracing, auth, etc)
- Rolling upgrades and frequent (~weekly) releases

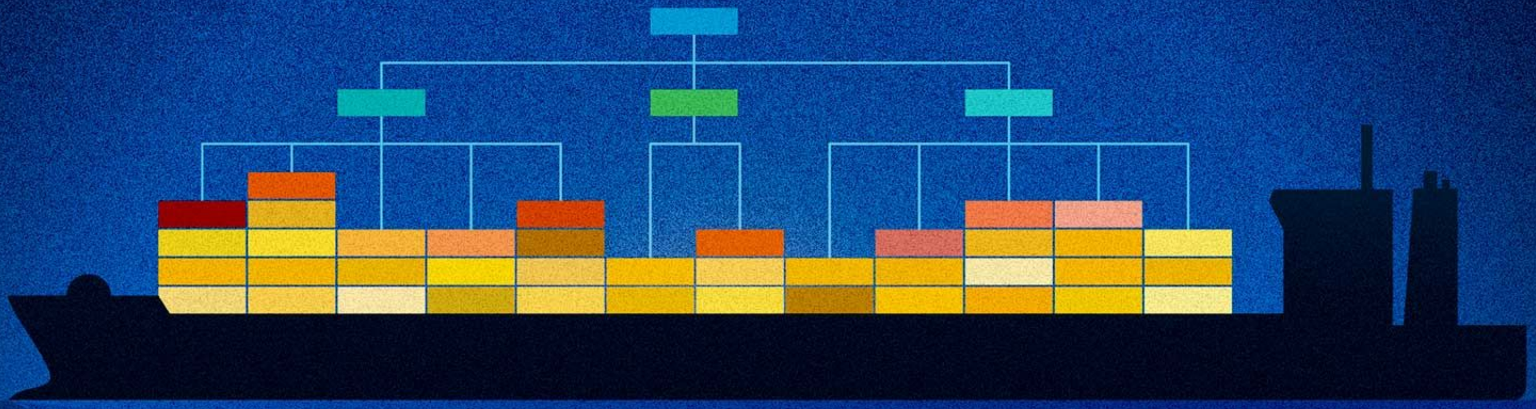
Sounds kinda familiar…

Part Three: Lessons

Lesson 1

Know Why

The only good reason to adopt microservices



You will inevitably ship your org chart

Accidental Microservices

- “Microservices due to Computer Science,” not org charts!
- Ended up with something similar to modern microservice architectures …
- … but for different reasons (and that eventually became a problem)

What’s best for Search+Ads is best for all!

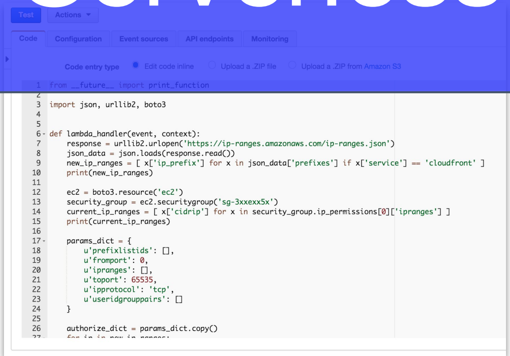
What's best for Search+Ads is best for ~~all!~~ just the massive, planet-scale services

“But I just want to serve 5TB!!”

Lesson 2

Serverless Still Runs on Servers

An aside: what do these things have in common?



All 100% Serverless!

LIGHTSTEP

About “Serverless” / FaaS

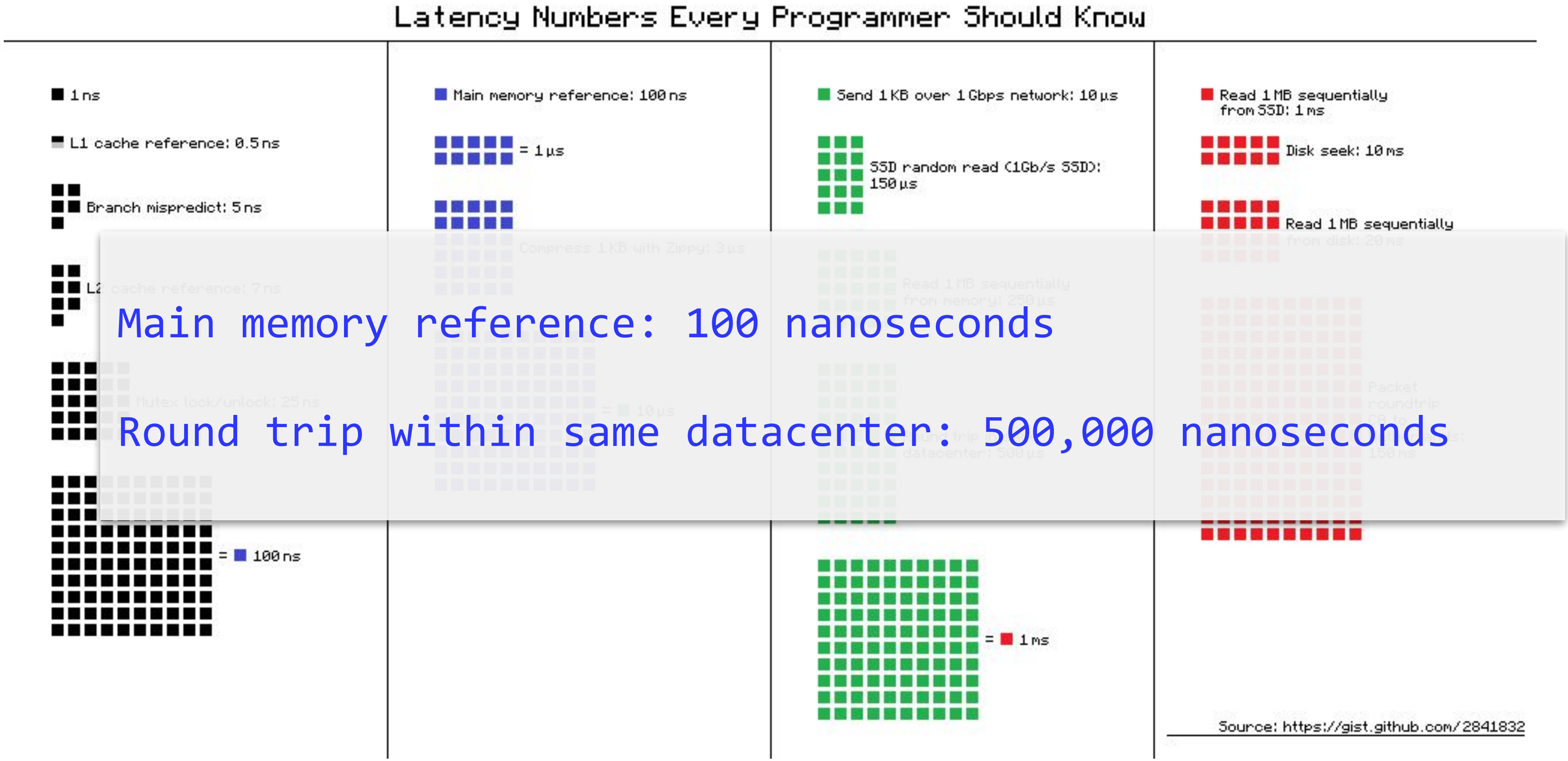
```
Numbers every engineer should know

Latency Comparison Numbers (~2012)
----------------------------------
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        3 us
Send 1K bytes over 1 Gbps network       10,000   ns       10 us
Read 4K randomly from SSD*             150,000   ns      150 us          ~1GB/sec SSD
Read 1 MB sequentially from memory     250,000   ns      250 us
Round trip within same datacenter      500,000   ns      500 us
Read 1 MB sequentially from SSD*     1,000,000   ns    1,000 us    1 ms  ~1GB/sec SSD, 4X memory
Disk seek                           10,000,000   ns   10,000 us   10 ms  20x datacenter roundtrip
Read 1 MB sequentially from disk    20,000,000   ns   20,000 us   20 ms  80x memory, 20X SSD
Send packet CA->Netherlands->CA    150,000,000   ns  150,000 us  150 ms

Notes
-----
1 ns = 10^-9 seconds
1 us = 10^-6 seconds = 1,000 ns
1 ms = 10^-3 seconds = 1,000 us = 1,000,000 ns

Credit
------
By Jeff Dean:               http://research.google.com/people/jeff/
Originally by Peter Norvig: http://norvig.com/21-days.html#answers
```

About “Serverless” / FaaS



Main memory reference: 100 nanoseconds

Round trip within same datacenter: 500,000 nanoseconds

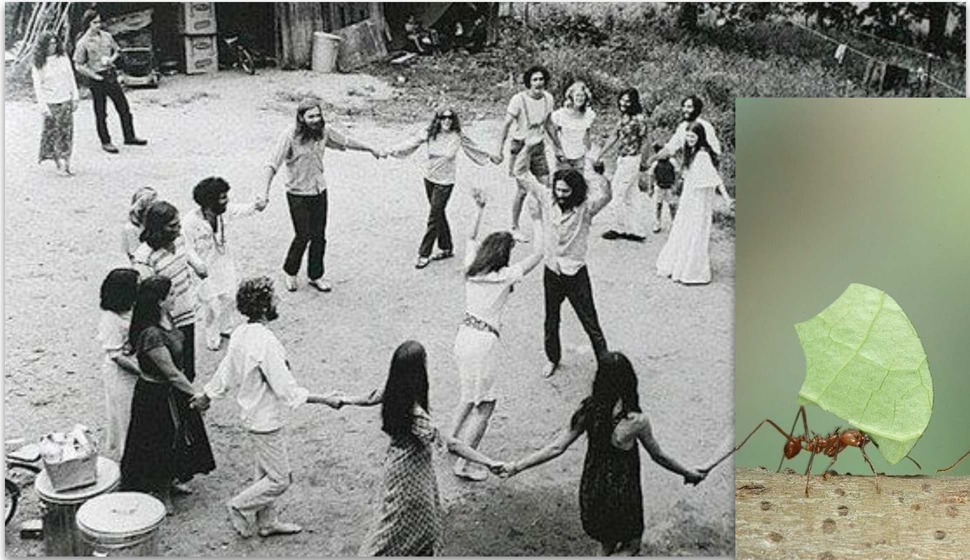
About “Serverless” / FaaS

- Embarrassingly parallel / stateless situations do exist
- FaaS are great for them
- … but *caching*

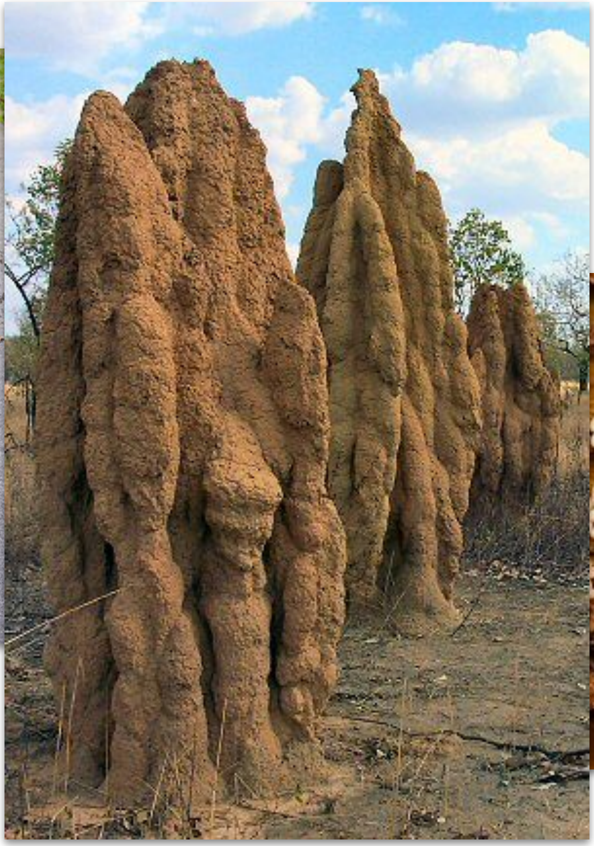
Lesson 3

“Independence” is
not an Absolute

Hippies vs Ants



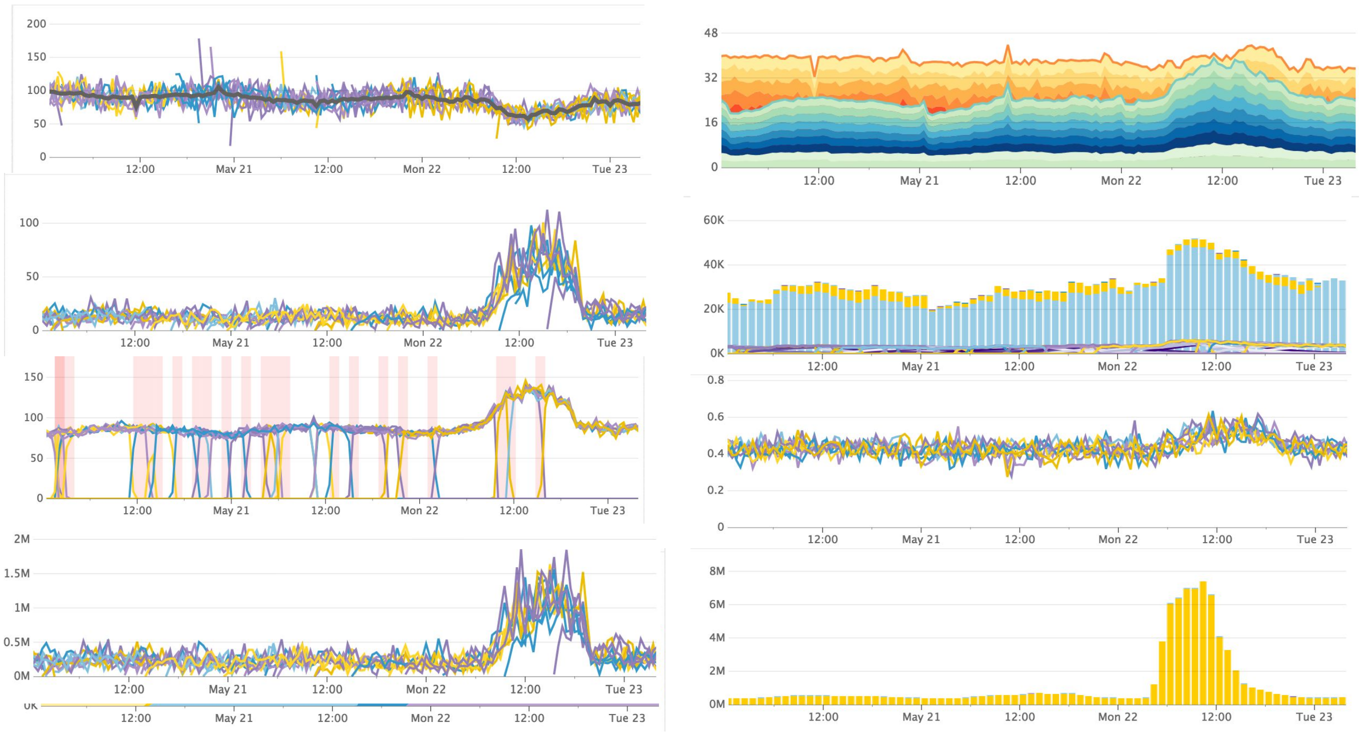
More Ants!



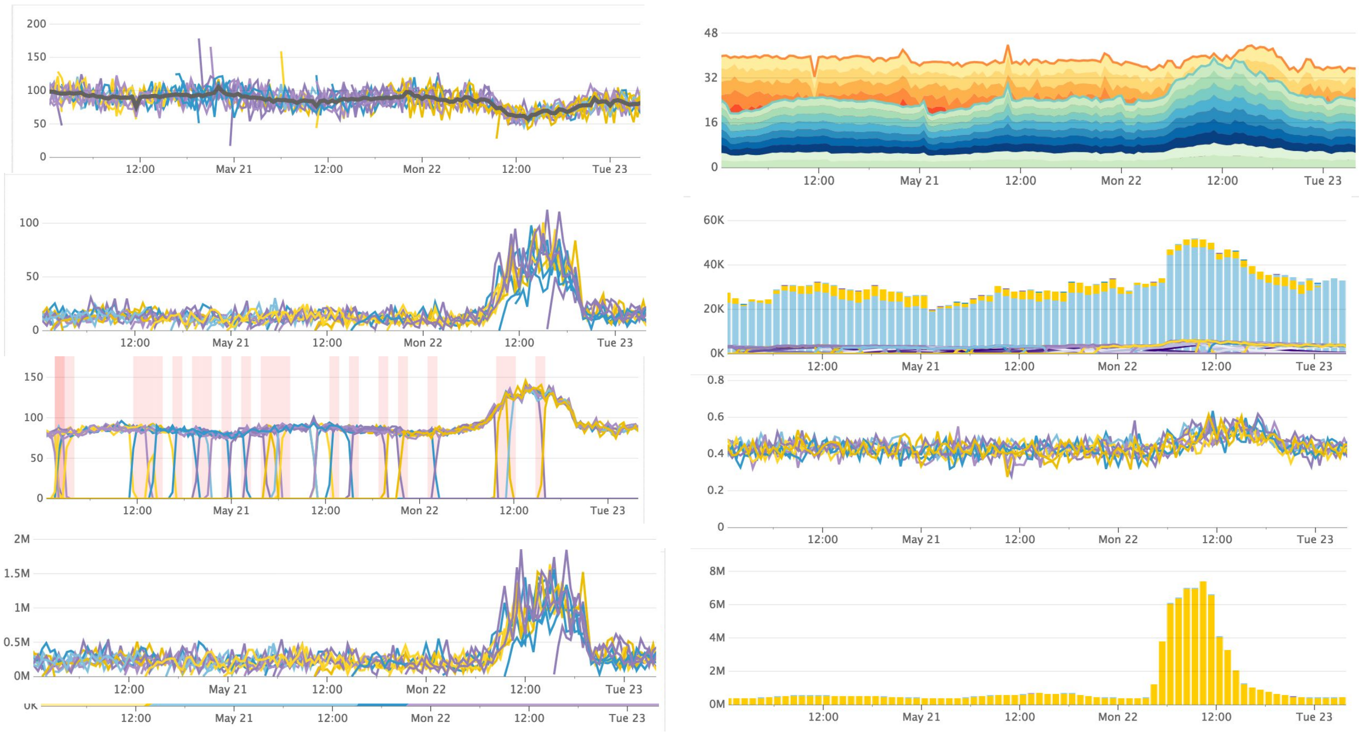
Lesson 4

Beware Giant Dashboards

We caught the regression!



… but which is the culprit?



Observability boils down to two activities

1. Detection of critical signals
2. Refining the search space

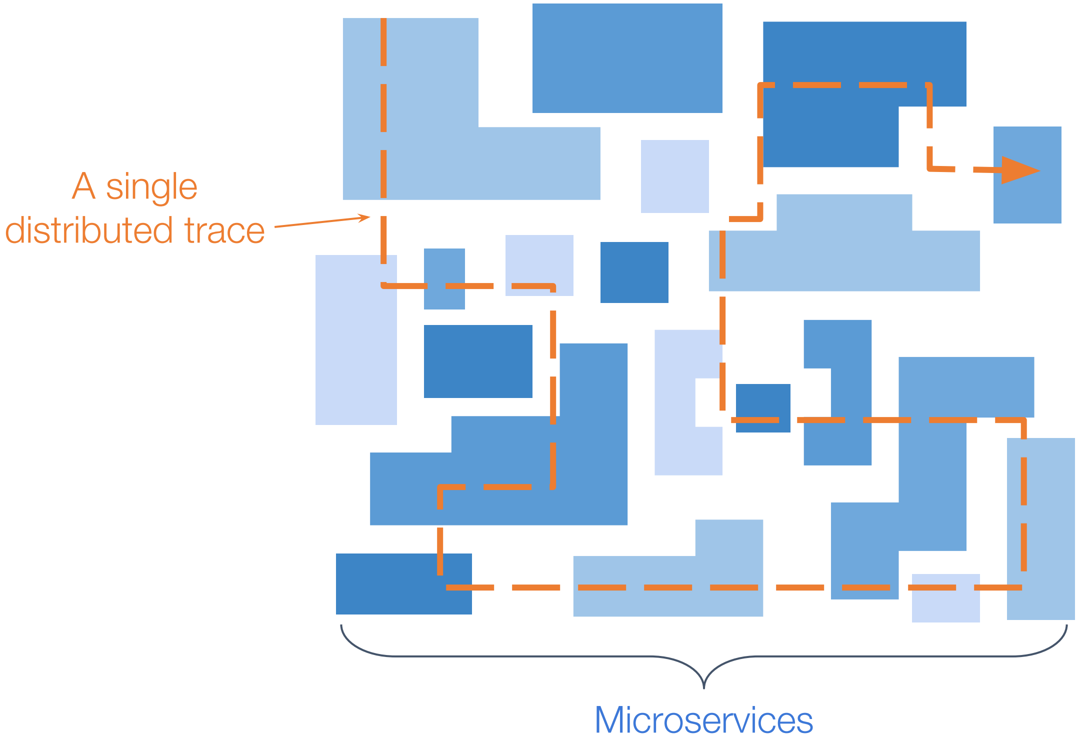
Don't confuse "visualizing the *entire* search space" with "*refining* the search space"

Lesson 5

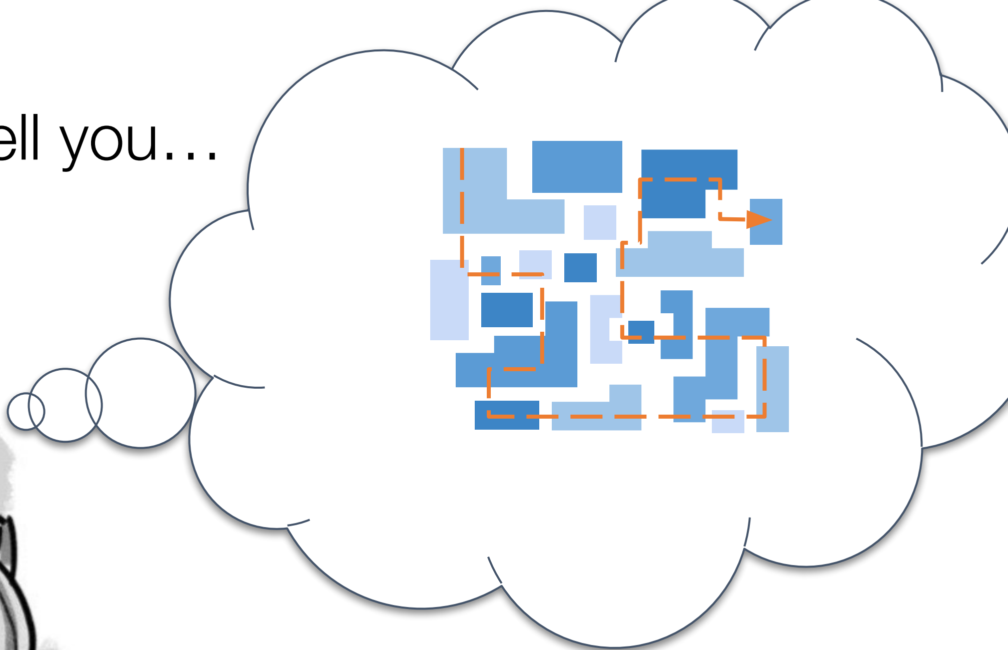
You Can’t Trace Everything

(or can you?)

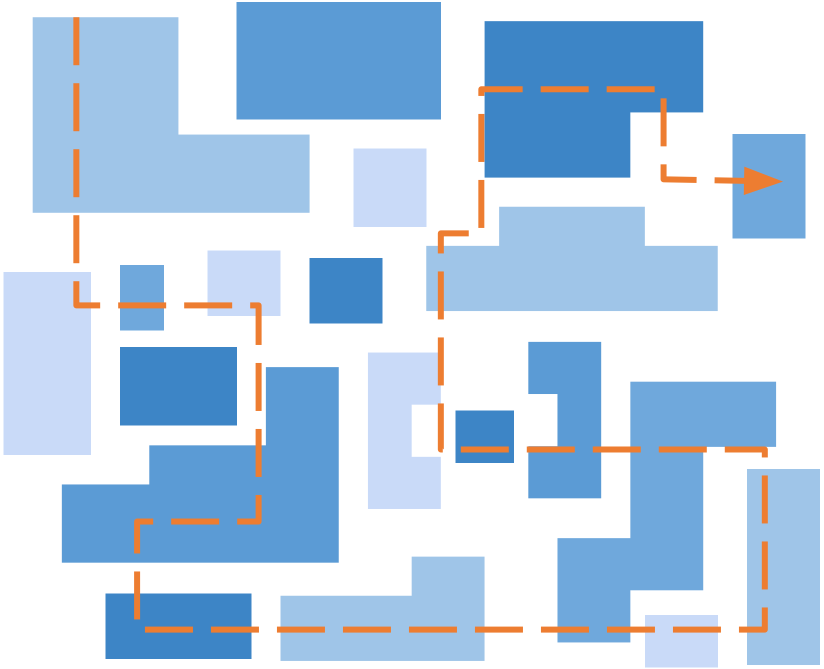
Distributed Tracing 101



There's something I need to tell you…



Trace Data Volume: a reality check



```
  app transaction rate
x    # of microservices
x   cost of net+storage
x    weeks of retention
------------------------
     way too much $$$$
```

LIGHTSTEP

The Life of Trace Data: Dapper

Stage	**Overhead affects…**	**Retained**
Instrumentation Executed	App	100.00%
Buffered within app process	App	000.10%
Flushed out of process	App	000.10%
Centralized regionally	Regional network + storage	000.10%
Centralized globally	WAN + storage	000.01%

LIGHTSTEP

The Life of Trace Data: ~~Dapper~~ Other Approaches

Stage	Overhead affects…	Retained
Instrumentation Executed	App	100.00%
Buffered within app process	App	100.00%
Flushed out of process	App	100.00%
Centralized regionally	Regional network + storage	100.00%
Centralized globally	WAN + storage	on-demand

LIGHTSTEP

Almost Done…

Let’s review…

- Two drivers for microservices: what are you solving for?
    - Team independence
    - “Computer Science”
- Understand the appropriate scale for any solution
- Services can be too small (i.e., “the network isn’t free”)
- Hippies vs Ants
- Observability is about *Detection* and *Refinement*
- We *can* trace everything

LIGHTSTEP

Thank you

Ben Sigelman, Co-founder and CEO
twitter: @el_bhs
email: bhs@lightstep.com

PS: Working at LightStep is fun!
lightstep.com/culture

LIGHTSTEP