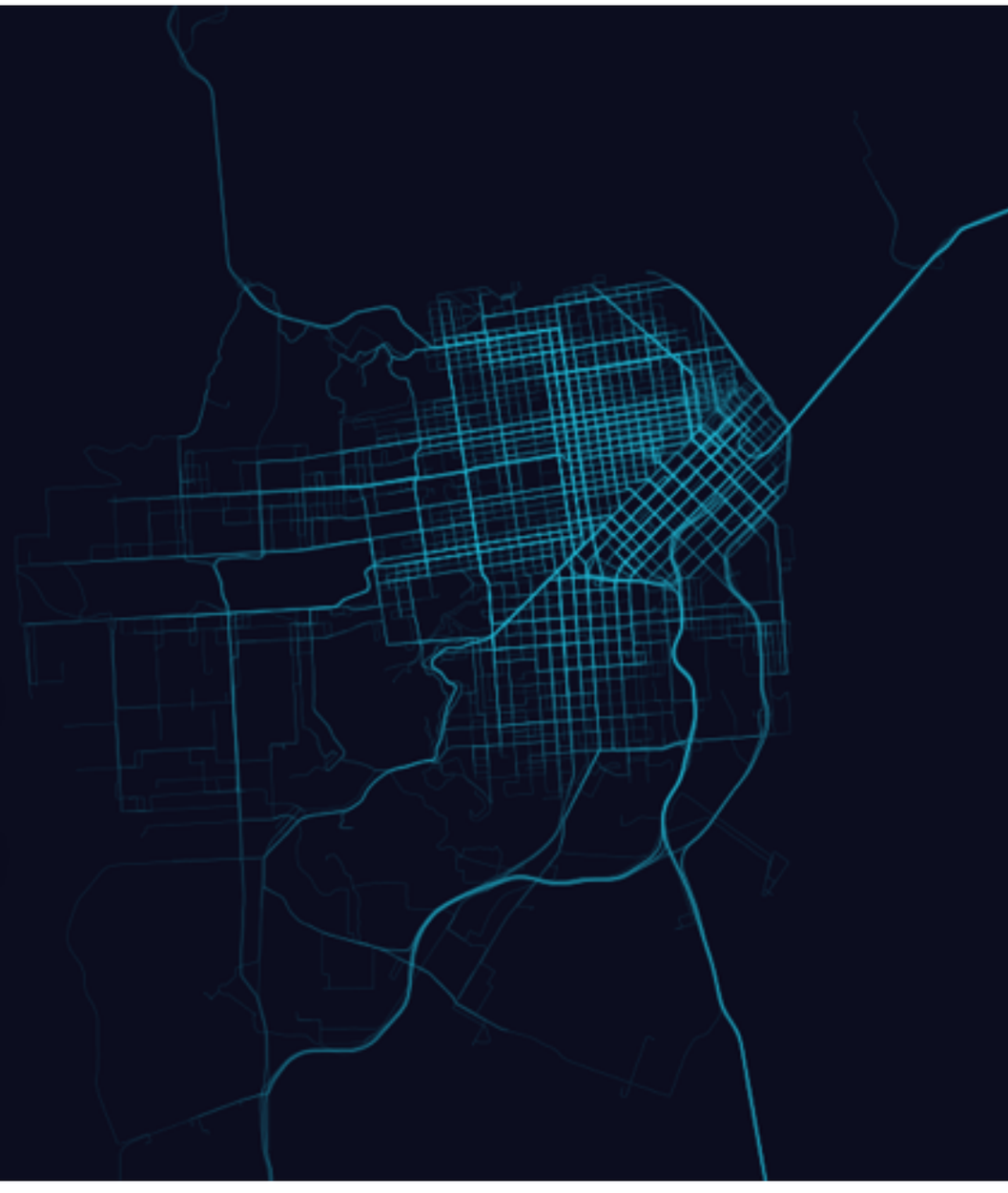
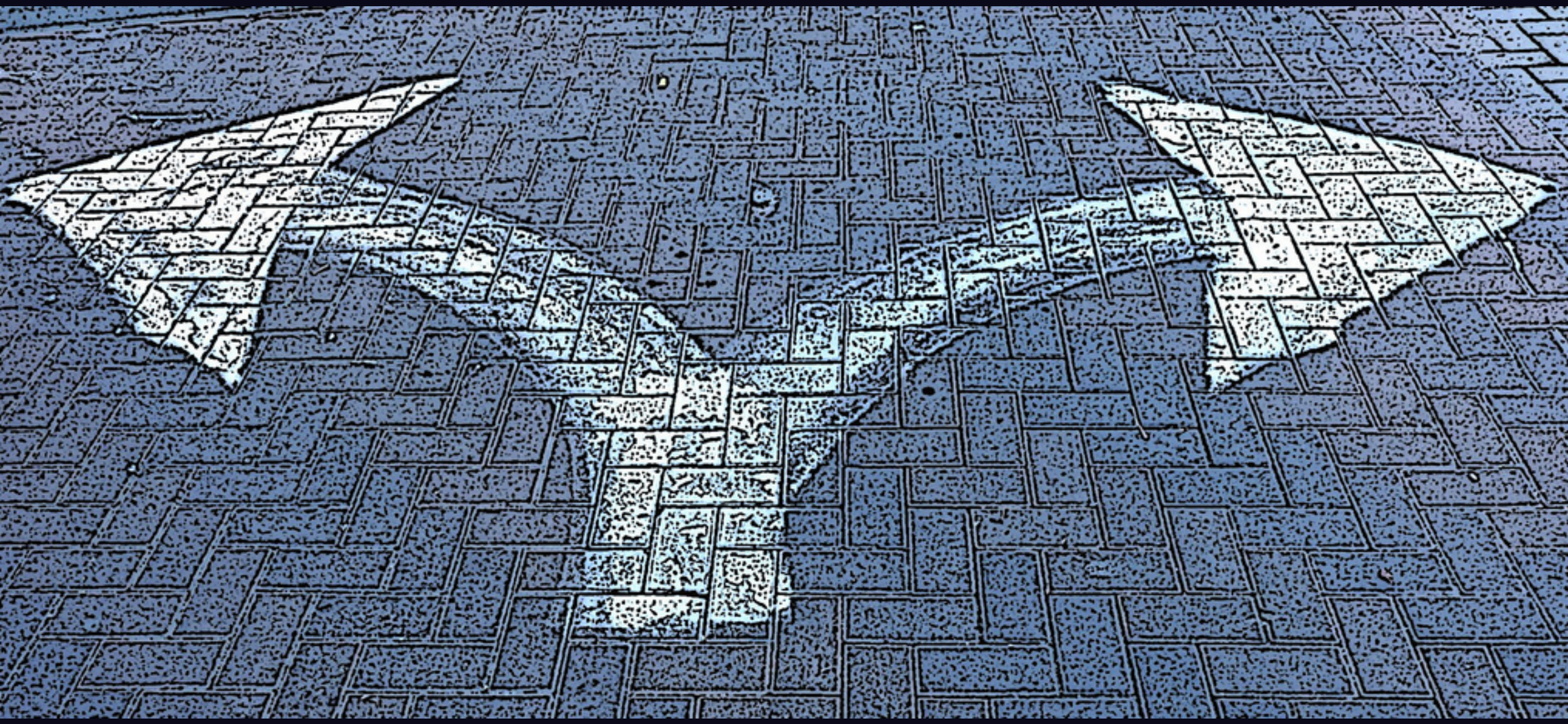


再造轮子之道

袁泳 @ UBER

UBER

不容忽视的两难选择



反方意见：造轮子 = 战略错误

It's important to remember that when you start from scratch there is absolutely no reason to believe that you are going to do a better job than you did the first time.

— Joel Spolsky, 2001

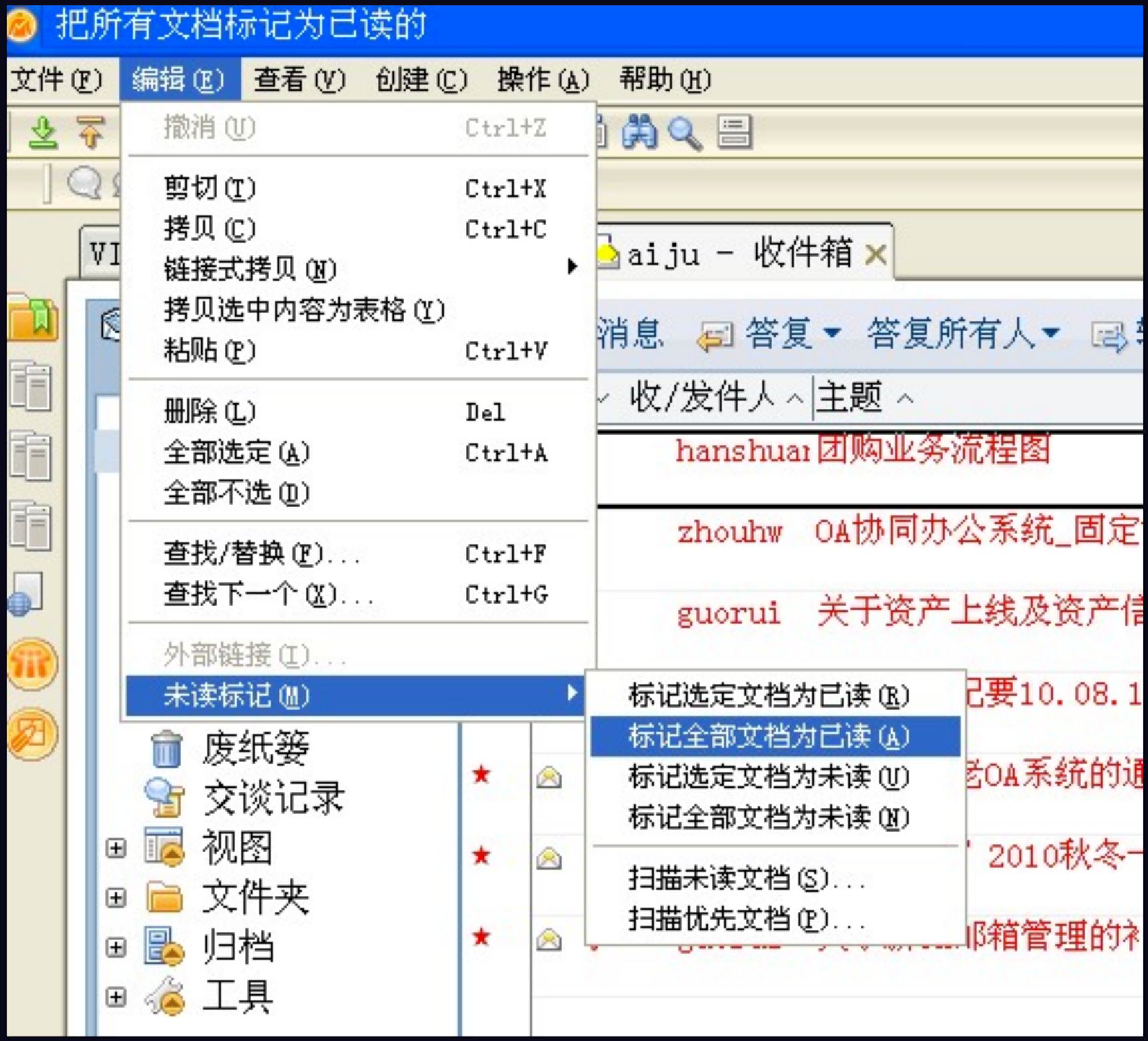
Netscape

Borland Quattro Pro

正方意见：再造轮子 = 学习+革新

案例：重写Lotus Notes 公式引擎

案例：重写Lotus Notes 公式引擎



改写前

- 超过10年的代码库
- 上百积压bugs
- 大量优化
- 资深工程师： “没人能动这套代码”
- Damien Katz: 多年UI经验，无C++经验

改写后

- 全新C++代码库
- 0 bug
- 提速300%
- 全新数据复制组件
- CouchDB

造轮子的唯一原因

核心竞争力

Google: 基础架构

Uber: Dispatch

Netflix: 高质廉价的影视作品

Undifferentiated Heavy Lifting

无关紧要的重活

常见工程师黑话

- 第三方设计太复杂
- 第三方代码太丑陋
- 第三方系统太不稳定

都不是理由

- 降低基建开销200%
- 系统可用时间从3个9提升到4个9
- 没有系统能够支持公司的战略功能

Auto Scaling

Auto Scaling
vs
Predictive Auto Scaling

造轮子的正确姿势

像科学研究一样

遍历文献

遍历文献 = 熟悉已知系统

熟悉历史沿革

熟悉历史沿革 = 了解技术发展中的取舍

案例： Elasticsearch

案例：Elasticsearch

- 自己的Leader Election算法

案例： Elasticsearch

- 自己的Leader Election算法
- 自己的集群管理算法 (zen discovery)

案例： Elasticsearch

- 自己的Leader Election算法
- 自己的集群管理算法 (zen discover)
- Call-me-maybe Test: > 50% 数据丢失

案例：Facebook Zeus

应用场景

- 服务发现

应用场景

- 服务发现
- 动态配置管理

应用场景

- 服务发现
- 动态配置管理
- 自动分片(shard)管理

应用场景

- 服务发现
- 动态配置管理
- 自动分片(shard)管理
- 分布式锁和分布协调

需求

- 跨区集群.
  - 西岸： 两个数据中心。
  - 东岸： 三个数据中心

需求

- 跨区集群.
  - 西岸： 两个数据中心。
  - 东岸： 三个数据中心
- 至少百万客户端

需求

- 跨区集群.
  - 西岸： 两个数据中心。
  - 东岸： 三个数据中心
- 至少百万客户端
- DoS 保护： 流畅处理大量客户端同时上线

需求

- 跨区集群.
  - 西岸：两个数据中心。
  - 东岸：三个数据中心
- 至少百万客户端
  - DoS 保护：流畅处理大量客户端同时上线
  - 100%上线 - 任何时候可读

怎么入手?

遍历文献 = 考察现有系统

- Google Chubby
- Apache Zookeeper

Apache Zookeeper

Zookeeper远非完美

Zookeeper远非完美

- 写入吞吐量太低

Zookeeper远非完美

- 写入吞吐量太低
- 保证写入顺序导致写入延迟

Zookeeper远非完美

- 写入吞吐量太低
- 保证写入顺序导致写入延迟
- 客户端短时大量接入导致服务器当掉

Zookeeper远非完美

- 写入吞吐量太低
- 保证写入顺序导致写入延迟
- 客户端短时大量接入导致服务器当掉
- 不能承受上亿监控节点

Zookeeper远非完美

- 写入吞吐量太低
- 保证写入顺序导致写入延迟
- 客户端短时大量接入导致服务器当掉
- 不能承受上亿监控节点
- Leader节点挂掉造成长时间服务中断

解决方案

- 确认性能瓶颈
  - Leader节点的锁争用
  - Java wait-notify 耗时太长

解决方案

- 确认性能瓶颈
  - Leader节点的锁争用
  - Java wait-notify 耗时太长
- 采纳精简数据结构
  - 位图表示监控节点 - 250GB to 300MB

解决方案

- 确认性能瓶颈
  - Leader节点的锁争用
  - Java wait-notify 耗时太长
- 采纳精简数据结构
  - 位图表示监控节点 - 250GB to 300MB
- 常用优化手段
  - 流水线
  - 客户端缓存
  - 适当放松协议限制

从“小”开始

从“小”开始 = 快速迭代

从“小”开始 = 累进改动

从“小”开始 = 最多考虑10倍增长

案例：Netflix Atlas

Netflix Atlas

- Netflix的监控系统

Netflix Atlas

- Netflix的监控系统
- 第一版：Perl写的RRD系统

Netflix Atlas

- Netflix的监控系统
- 第一版：Perl写的RRD系统
- 经验：OLAP是杀手功能

Netflix Atlas

- Netflix的监控系统
- 第一版：Perl写的RRD系统
- 经验：OLAP是杀手功能
- 经验：查询语言必须灵活完备

Netflix Atlas

- Netflix的监控系统
- 第一版：Perl写的RRD系统
- 经验：OLAP是杀手功能
- 经验：查询语言必须灵活完备
- 经验：系统瓶颈随着流量不断变化

Netflix Atlas

- 200亿数据点
- 极度灵活的查询语言
- 高效强大时序数据OLAP系统

聚焦关键问题

聚焦关键问题 = 专注公司发展

公司发展 = 工程师关注的焦点

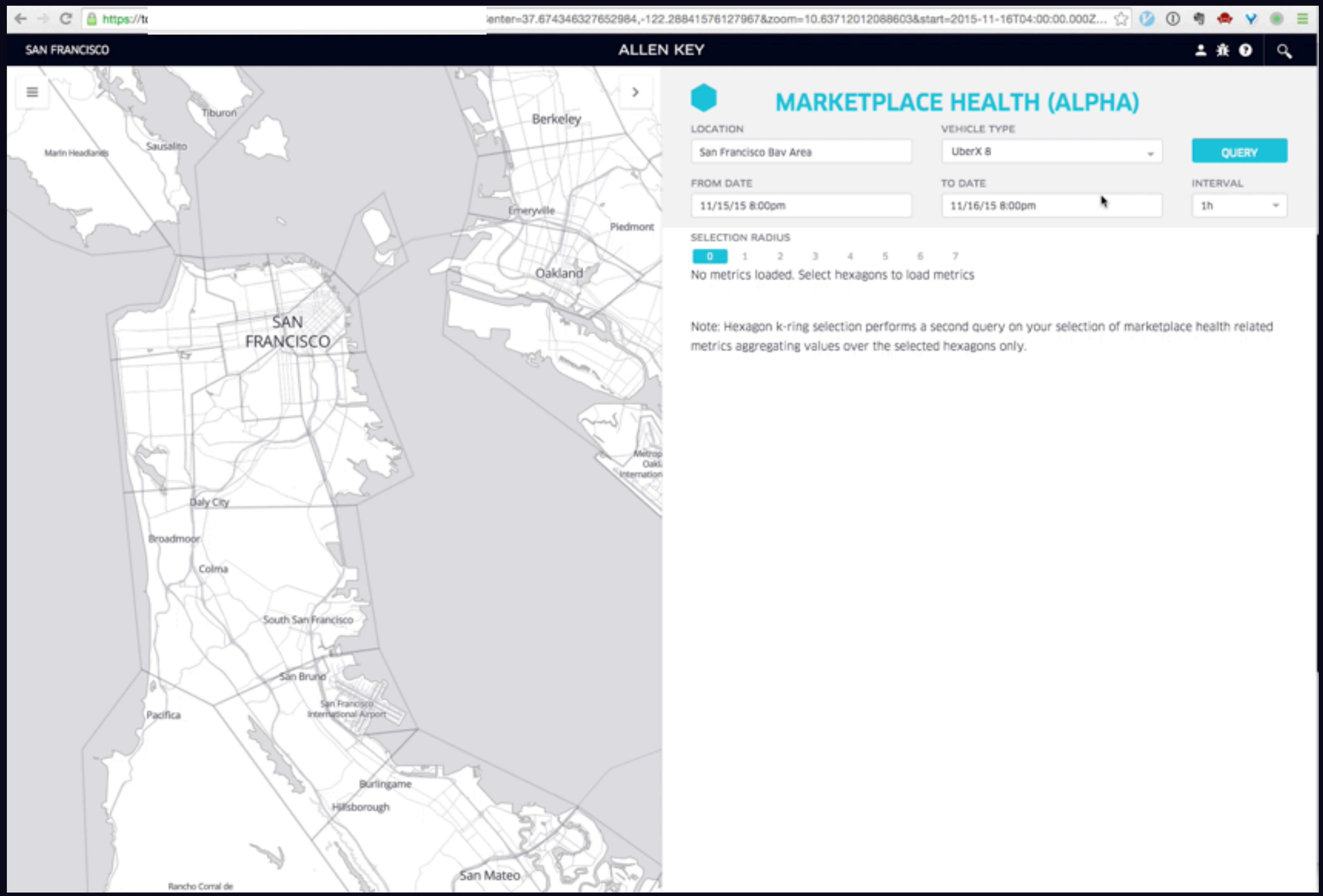
公司发展 => 高强度应用场景

工程师的顾虑：程序员不是开源软件的组装师

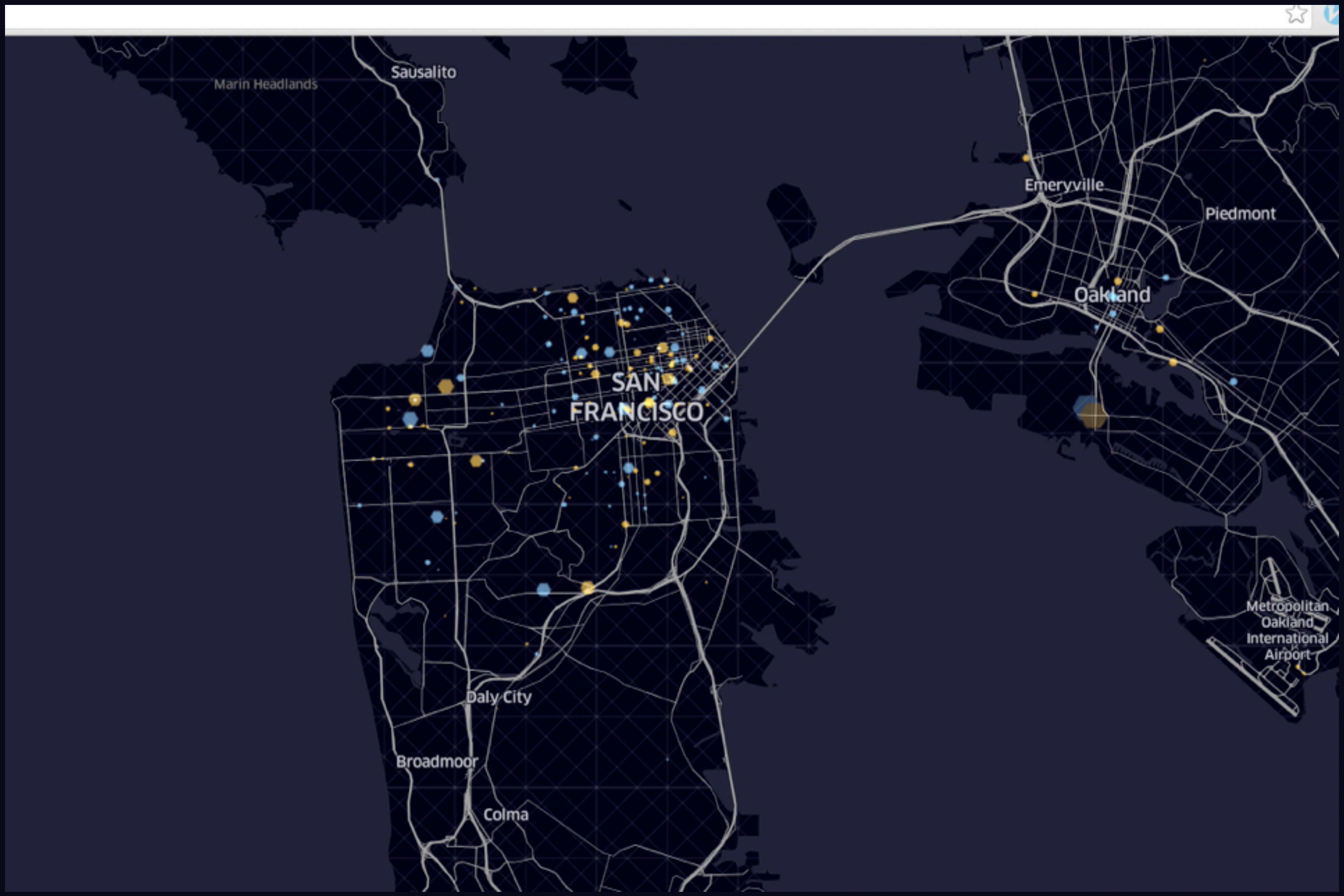
案例：阿里JStorm

案例：Uber Marketplace Streaming

历史变化

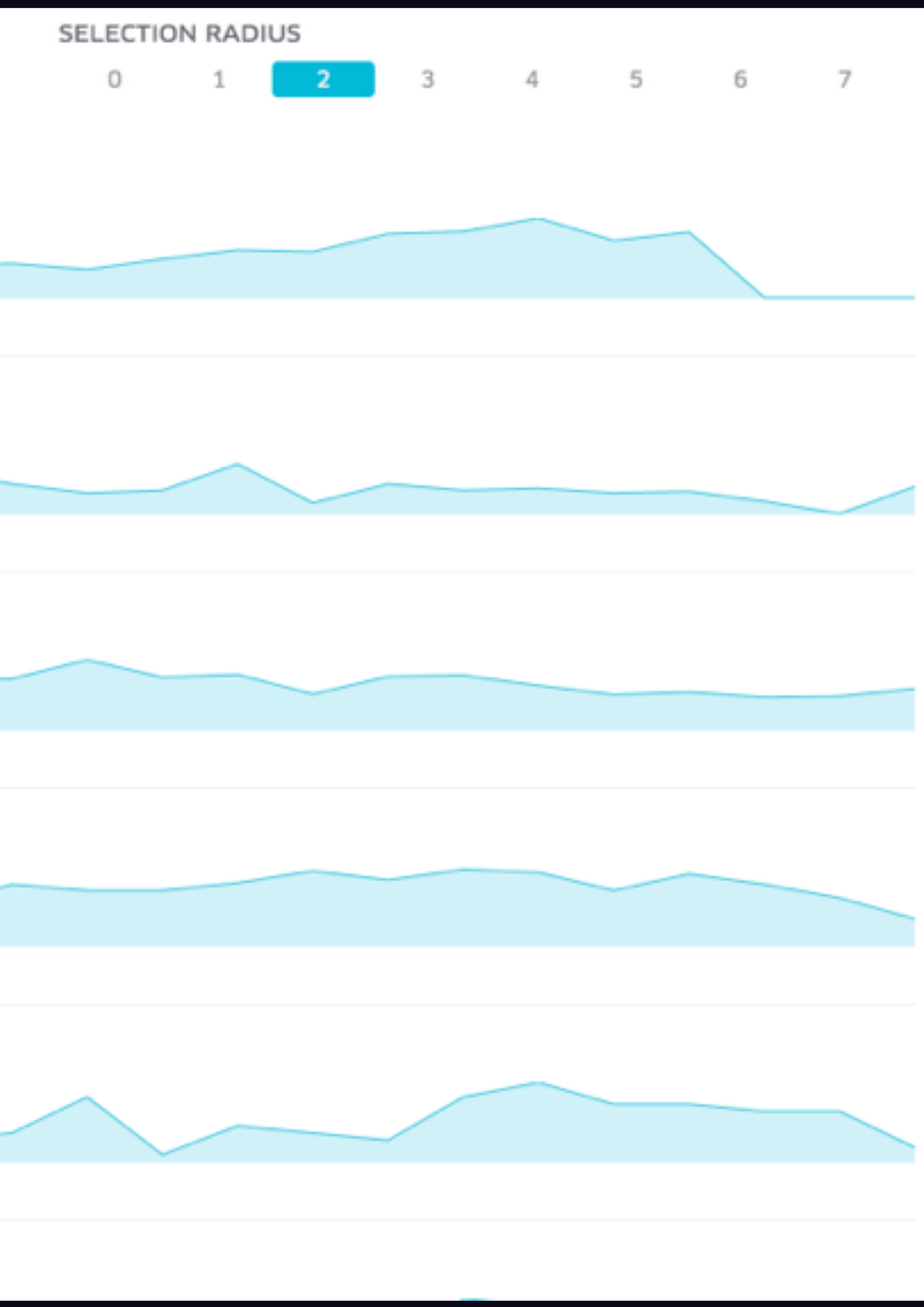

SAN FRANCISCO
ALLEN KEY
MARKETPLACE HEALTH (ALPHA)
LOCATION
San Francisco Bay Area
VEHICLE TYPE
UberX 8
QUERY
FROM DATE
11/15/15 8:00pm
TO DATE
11/16/15 8:00pm
INTERVAL
1h
SELECTION RADIUS
0 1 2 3 4 5 6 7
No metrics loaded. Select hexagons to load metrics
Note: Hexagon k-ring selection performs a second query on your selection of marketplace health related metrics aggregating values over the selected hexagons only.
Tiburon
Sausalito
Marin Headlands
Berkeley
Emeryville
Piedmont
Oakland
SAN FRANCISCO
Daly City
Broadmoor
Colma
South San Francisco
San Bruno
Pacifica
Burlingame
Hillsborough
San Mateo

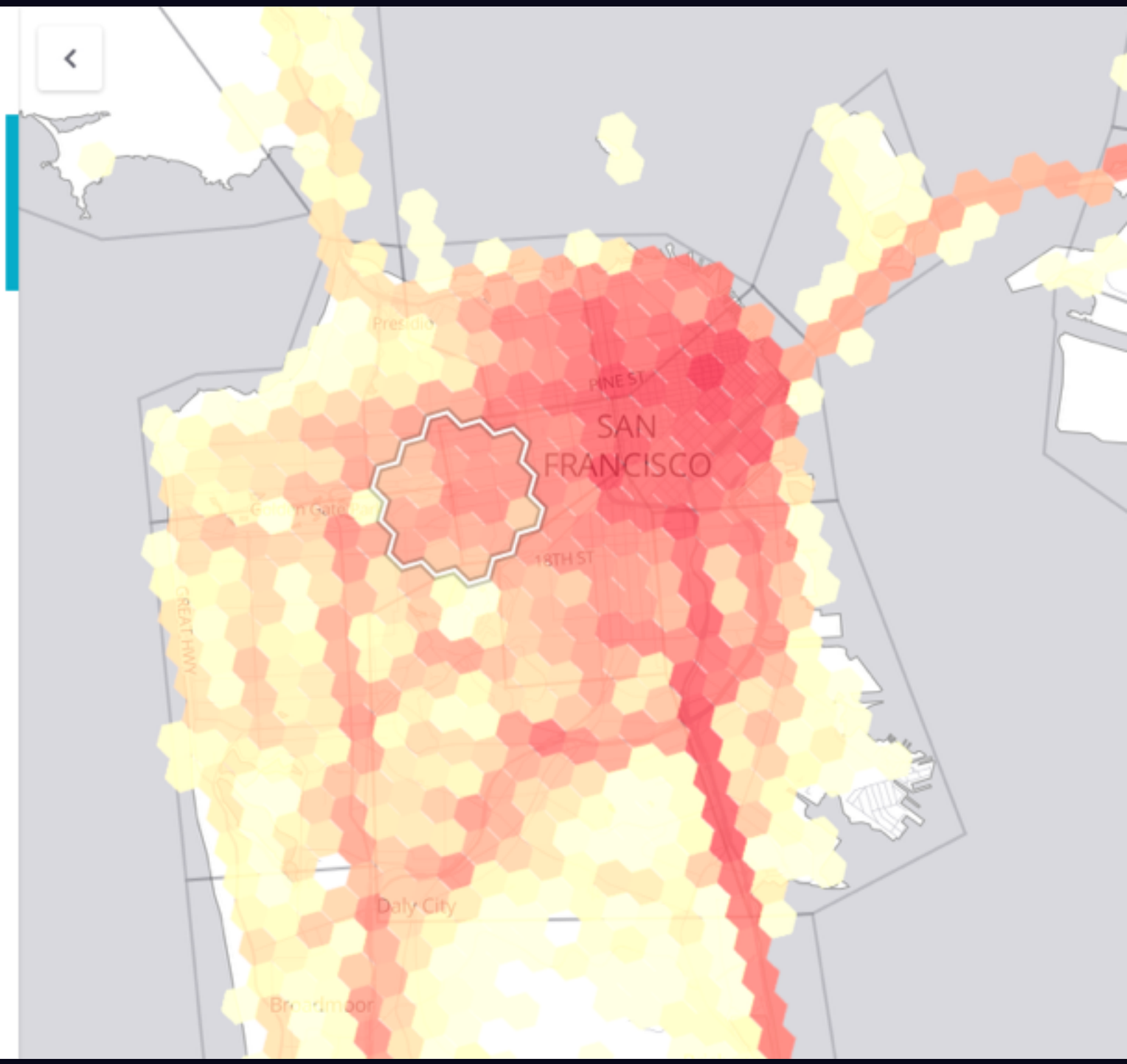
供求分布


Marin Headlands
Sausalito
Emeryville
Piedmont
Oakland
SAN
FRANCISCO
Metropolitan
Oakland
International
Airport
Daly City
Broadmoor
Colma

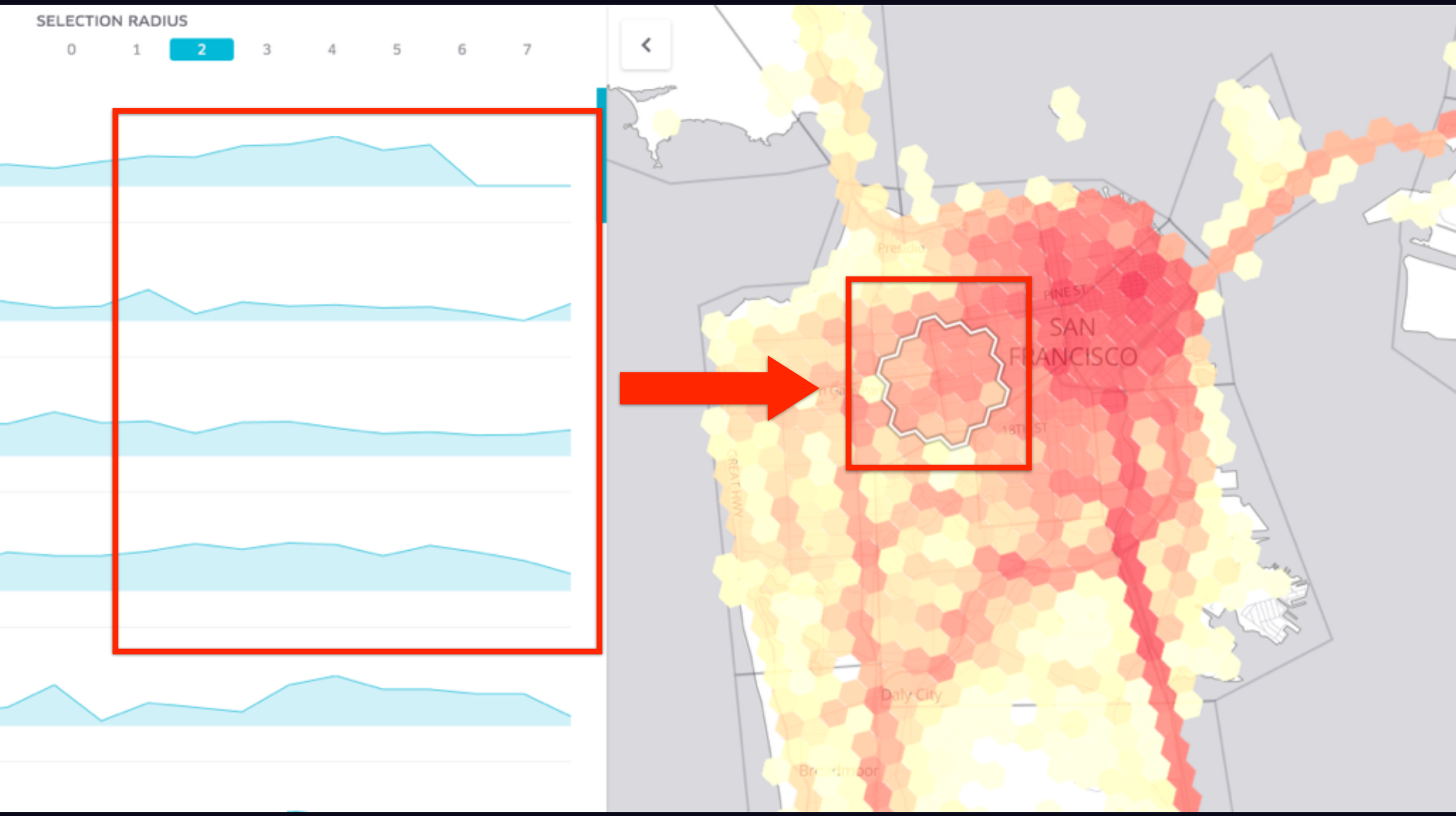
黑话: Clustering & Pr(D, S, E)

聚焦：实时计算


SELECTION RADIUS
0
1
2
3
4
5
6
7


PINE ST
SAN
FRANCISCO
18TH ST
GREAT HWY
Daly City

聚焦：实时计算



计算规模

一个城市至少10,000六边形

计算规模

每个六边形331个邻居需要处理

计算规模

一次计算：331 x 10,000 = 310万六边形

计算规模

99%-ile 处理时间: 70ms

计算规模

六边形变长缩短一倍＝计算量变为4倍

应用场景

- 市场供求

应用场景

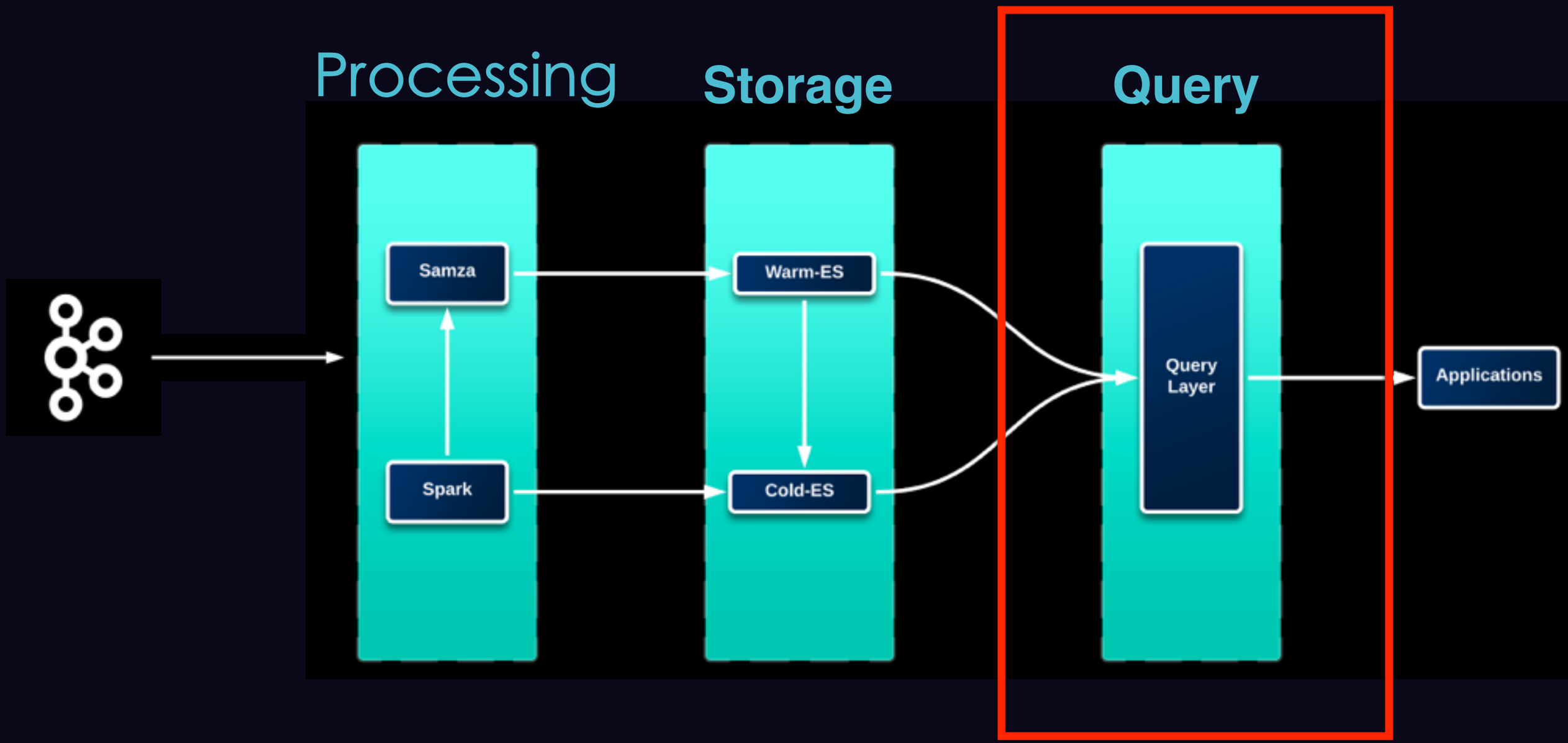
- 市场供求
- 市场价格

应用场景

- 市场供求
- 市场价格
- 预测

应用场景

- 市场供求
- 市场价格
- 预测
- 用户行为

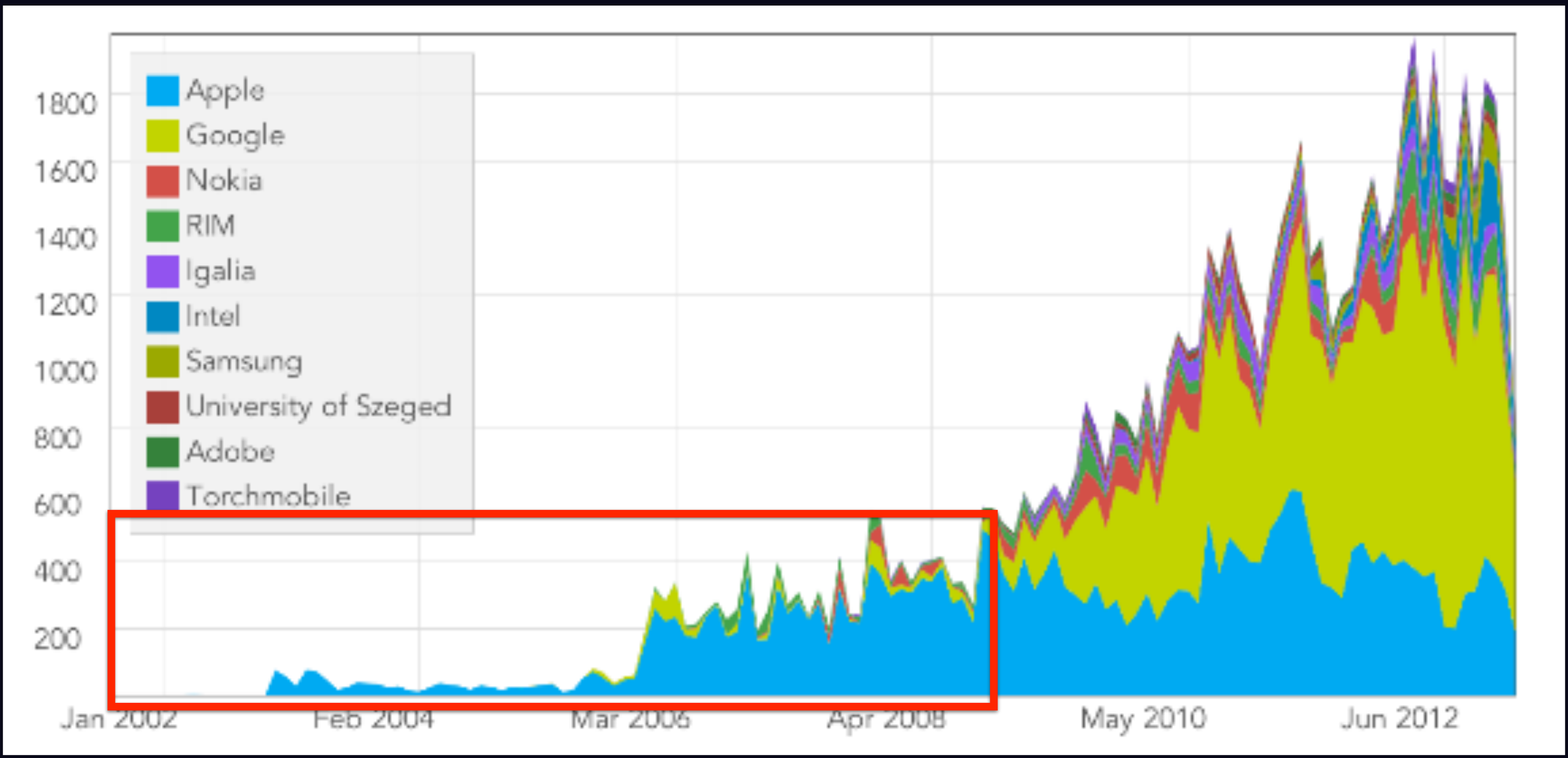

Processing
Storage
Query
Samza
Warm-ES
Query Layer
Applications
Spark
Cold-ES

老板的顾虑：如何控制开源软件

Publish or Perish

Code Hard or Go Home

案例：WebKit vs KHTML



案例：Netflix Open Source

案例：Netflix Open Source

壮大社区，而不是疏离社区

总结

- 关注公司核心问题
- 寻求高度挑战的应用场景
- 像科学家一样再造轮子

谢谢！