

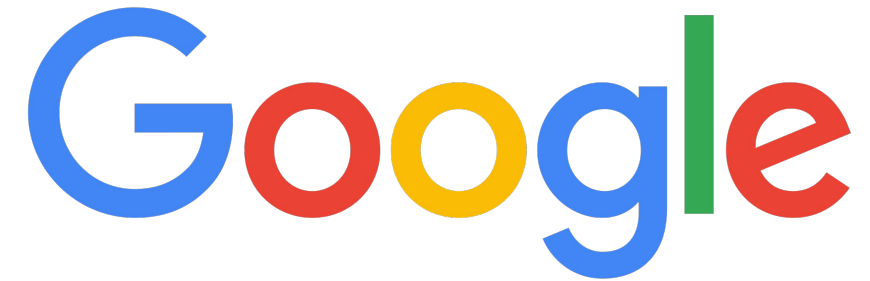
Insights from Data

Andy Velasquez | Account Manager

Sean Maday | Engineering Manager

Google Cloud





Organize the world's information and make
it universally accessible and useful.

– Our Mission

Government Cloud Use Cases



Government Accessibility

Solutions that make government more open to:

- Limited English Proficient residents
- Digital Natives
- Self-Service



Employee Productivity

Modernizing IT services and enhancing employee productivity



Data / Analytics

Leveraging machine learning and AI for data driven decision making, predictive analytics and more meaningful customer interactions.



Infrastructure Elasticity

Creating an elastic computing framework to service new and expanding IT needs

- Global Scale
- Google Grade Security
- Economies of Google Scale



1
BILLION
users



You Tube

Google in 1 minute



3M Searches

You Tube

500 Hours



**1000 new
devices**

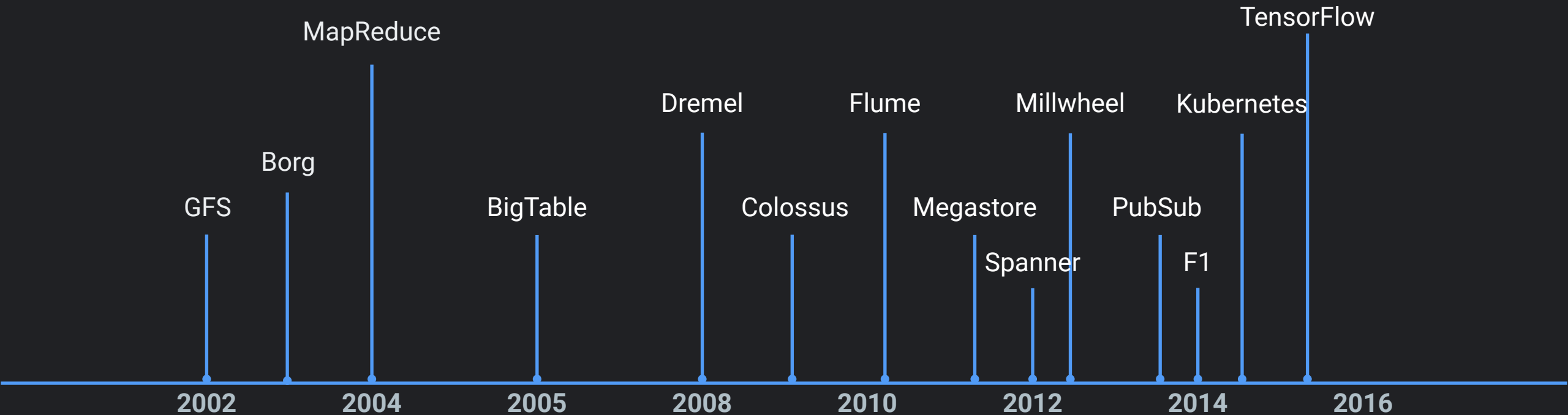


**1.2M Photos
Uploaded**



**450K Miles
Driven**

Google's Data Research



18 years of advancing the state of the art...



Google Cloud

Google File System

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.1.4: 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.ghemawat, 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, Boston, Massachusetts, 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, file sizes 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.

MapReduce

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

an associating large processes a key/value intermediate key. Many of, as shown

an automating the process of caring of the problem of scaling machine without any system. on a large ly scalable: s many teragrams Reduce pro of one thou-le's clusters

given day, etc. Most such computation ally straightforward. However, the input large and the computations have to be hundreds or thousands of machines in a reasonable amount of time. The issue allize the computation, distribute the failures conspire to obscure the original with large amounts of complex c these issues.

As a reaction to this complexity, we abstraction that allows us to express the abstraction we were trying to perform but hid tails of parallelization, fault-tolerance, and load balancing in a library. Our inspired by the *map* and *reduce* primitive and many other functional languages. most of our computations involved apperation to each logical "record" in our compute a set of intermediate key/value applying a *reduce* operation to all the v the same key, in order to combine the properly. Our use of a functional r specified map and reduce operations alize large computations easily and to as the primary mechanisms for fault tolerance

The major contributions of this work powerful interface that enables automaton and distribution of large-scale computation with an implementation of this interface high performance on large clusters of ex

Section 2 describes the basic program gives several examples. Section 3 describes the MapReduce interface our cluster-based computing environment describes several refinements of the program that we have found useful. Section 5 measurements of our implementation tasks. Section 6 explores the use of MapReduce including our experiences in using

BigTable

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandrasekaran, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilson,hert,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully provided a flexible, high-performance solution for all of these Google products. In this paper we describe the simple data model provided by Bigtable, which gives clients dynamic control over data layout and format, and we describe the design and implementation of Bigtable.

1 Introduction

Over the last two and a half years we have designed, implemented, and deployed a distributed storage system for managing structured data at Google called Bigtable. Bigtable is designed to reliably scale to petabytes of data and thousands of machines. Bigtable has achieved several goals: wide applicability, scalability, high performance, and high availability. Bigtable is used by more than sixty Google products and projects, including Google Analytics, Google Finance, Orkut, Personalized Search, WriteUp, and Google Earth. These products use Bigtable for a variety of demanding workloads, which range from throughput-oriented batch-processing jobs to latency-sensitive serving of data to end users. The Bigtable clusters used by these products span a wide range of configurations, from a handful to thousands of servers, and store up to several hundred terabytes of data. In many ways, Bigtable resembles a database: it shares many implementation strategies with databases. Parallel databases [14] and main-memory databases [13] have

achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage. Data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data as uninterpreted strings, although clients often serialize various forms of structured and semi-structured data into these strings. Clients can control the locality of their data through careful choices in their schemas. Finally, Bigtable schema parameters let clients dynamically control whether to serve data out of memory or from disk.

Section 2 describes the data model in more detail, and Section 3 provides an overview of the client API. Section 4 briefly describes the underlying Google infrastructure on which Bigtable depends. Section 5 describes the fundamentals of the Bigtable implementation, and Section 6 describes some of the refinements that we made to improve Bigtable's performance. Section 7 provides measurements of Bigtable's performance. We describe several examples of how Bigtable is used at Google in Section 8, and discuss some lessons we learned in designing and supporting Bigtable in Section 9. Finally, Section 10 describes related work, and Section 11 presents our conclusions.

2 Data Model

A Bigtable is a sparse, distributed, persistent multidimensional sorted map. The map is indexed by a row key, a column key, and a timestamp; each value in the map is an uninterpreted array of bytes.

(row:string, column:string, time:int64) → string

To appear in OSDI 2006

2002

2004

2005

FlumeJava: Easy, Efficient Data-F

Craig Chambers, Ashish Raniwala, Fran
Stephen Adams, Robert R. Henr
Robert Bradshaw, Nathan Weizenb
Google, Inc.
{chambers,raniwala,fjp,sra,rrh,robertwb,nweiz}@

Abstract

MapReduce and similar systems significantly ease the task of writing data-parallel code. However, many real-world computations require a pipeline of MapReduces, and programming and managing such pipelines can be difficult. We present FlumeJava, a Java library that makes it easy to develop, test, and run efficient data-parallel pipelines. At the core of the FlumeJava library are a couple of classes that represent immutable parallel collections, each supporting a modest number of operations for processing them in parallel. Parallel collections and their operations present a simple, high-level, uniform abstraction over different data representations and execution strategies. To enable parallel operations to run efficiently, FlumeJava defers their evaluation, instead internally constructing an execution plan dataflow graph. When the final results of the parallel operations are eventually needed, FlumeJava first optimizes the execution plan, and then executes the optimized operations on appropriate underlying primitives (e.g., MapReduces). The combination of high-level abstractions for parallel data and computation, deferred evaluation and optimization, and efficient parallel primitives yields an easy-to-use system that approaches the efficiency of hand-optimized pipelines. FlumeJava is in active use by hundreds of pipeline developers within Google.

Categories and Subject Descriptors D.1.3 [Concurrent Programming]: Parallel Programming

General Terms Algorithms, Languages, Performance

Keywords data-parallel programming, MapReduce, Java

1. Introduction

Building programs to process massive amounts of data in parallel can be very hard. MapReduce [6–8] greatly eased this task for data-parallel computations. It presented a simple abstraction to users for how to think about their computation, and it managed many of the difficult low-level tasks, such as distributing and coordinating the parallel work across many machines, and coping robustly with failures of machines, networks, and data. It has been used very successfully in practice by many developers. MapReduce’s success in this domain inspired the development of a number of related systems, including Hadoop [2], LINQ/Dryad [20], and Pig [3].

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.
PLDI’10, June 5–10, 2010, Toronto, Ontario, Canada
Copyright © 2010 ACM 978-1-4503-0019-3/10/06... \$10.00

MapReduce
down into a map
real-world comp
Such data-para
to chain togethe
tional work to r
mediate results
can become ob
making it diffi
records. Moreove
becomes “bake
logical computa

In this paper
support the dev
Java library cer
collections. Par
allel operations
computations. A
be implemented
stractions; there
separate progr

FlumeJava’s
how data is rep
as an in-memor
ternal storage s
Similarly, Flum
plementation str
as a local sequ
cation, or (in th
computation. T
tially developed
in a single proc
buggers, and th
data. They also
Java computati
services are dev

To achieve g
parallel operati
simply records
cution plan gra
computation ba
cution plan, fl
ment each oper
MapReduce, ba
places remote c

MillWheel: Fault-Tolerant Stream Processing at Internet Scale

Tyler Akidau, Alex Balikov, Kaya Bekiroğlu, Slava Chernyak, Josh Haberman, Reuven Lax, Sam McVeety, Daniel Mills, Paul Nordstrom, Sam Whittle

{takidau, alexgb, kayab, chernyak, haberman, relax, sgmc, millsd, pgn, samuelw}@google.com

ABSTRACT

MillWheel is a framework for building low-latency data-processing applications that is widely used at Google. Users specify a directed computation graph and application code for individual nodes, and the system manages persistent state and the continuous flow of records, all within the envelope of the framework’s fault-tolerance guarantees.

This paper describes MillWheel’s programming model as well as its implementation. The case study of a continuous anomaly detector in use at Google serves to motivate how many of MillWheel’s features are used. MillWheel’s programming model presents a notion of logical time, making it simple to write time-based aggregations. MillWheel was designed from the outset with fault tolerance and scalability in mind. In practice, we find that MillWheel’s unique combination of scalability, fault tolerance, and a versatile programming model lends itself to a wide variety of problems at Google.

1. INTRODUCTION

Stream processing systems are critical to providing content to users and allowing organizations to make faster and better decisions, particularly because of their ability to provide low latency results. Users want real-time news about the world around them. Businesses are likewise interested in the value provided by real-time intelligence sources such as spam filtering and intrusion detection. Similarly, scientists must cull noteworthy results from immense streams of raw data.

Streaming systems at Google require fault tolerance, persistent state, and scalability. Distributed systems run on thousands of shared machines, any of which can fail at any time. Model-based streaming systems, like anomaly detectors, depend on predictions that are generated from weeks of data, and their models must be updated on-the-fly as new data arrives. Scaling these systems by orders of magnitude should not cause a commensurate increase in the operational cost of building and maintaining the system.

Programming models for distributed systems, like MapReduce [11], hide the framework’s implementation details in the background,

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. Articles from this volume were invited to present their results at The 39th International Conference on Very Large Data Bases, August 26th - 30th 2013, Riva del Garda, Trento, Italy.
Proceedings of the VLDB Endowment, Vol. 6, No. 11
Copyright 2013 VLDB Endowment 2150-8097/13/09... \$10.00.

allowing users to create massive distributed systems expressed. By allowing users to focus solely on logic, this kind of programming model allows use the semantics of their system without being distracted. In particular, users are able to depend on correctness and fault-tolerance guarantees as axiomatic, restricting the surface area over which bugs and errors can creep. Supporting a variety of common programming models drives adoption, as users can leverage the utility of existing libraries in a familiar idiom, rather than to a domain-specific language.

MillWheel is such a programming model, tailored for streaming, low-latency systems. Users write about individual nodes in a directed compute graph, for example, to define an arbitrary, dynamic topology. Records are continuously along edges in the graph. MillWheel’s programming model at the framework level, where any node or topology can fail at any time without affecting the result. As part of this fault tolerance, every record is guaranteed to be delivered to its consumers.

API that MillWheel provides for record processing in an idempotent fashion, making record delivery exactly once from the user’s perspective. MillWheel progress at fine granularity, eliminating any need for data at external senders for long periods between updates.

Other streaming systems do not provide this correctness, versatility, and scalability. Spark Streaming [32] does excellent jobs of efficient checkpointing the space of operators that are available to use does not provide fully fault-tolerant persistent state [23] exactly-once mechanism for record delivery, requires strict transaction ordering to operate. Attention batch-processing model of MapReduce and Hadoop low-latency systems result in compromised flexibility operator-specific dependence on Replicated Distributed [33] in Spark Streaming. Streaming SQL system [21] [24] provide succinct and simple solutions to problems, but intuitive state abstractions and complex logic (e.g. matrix multiplication) are more natural in the operational flow of an imperative language declarative language like SQL.

Our contributions are a programming model for stream processing and an implementation of the MillWheel framework.

- We have designed a programming model that allows complex streaming systems to be created without complex state management expertise.
- We have built an efficient implementation of the MillWheel framework.

The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, Sam Whittle

{takidau, robertwb, chambers, chernyak, rfernand, relax, sgmc, millsd, fjp, cloude, samuelw}@google.com

ABSTRACT

Unbounded, unordered, global-scale datasets are increasingly common in day-to-day business (e.g. Web logs, mobile usage statistics, and sensor networks). At the same time, consumers of these datasets have evolved sophisticated requirements, such as event-time ordering and windowing by features of the data themselves, in addition to an insatiable hunger for faster answers. Meanwhile, practicality dictates that one can never fully optimize along all dimensions of correctness, latency, and cost for these types of input. As a result, data processing practitioners are left with the quandary of how to reconcile the tensions between these seemingly competing propositions, often resulting in disparate implementations and systems.

We propose that a fundamental shift of approach is necessary to deal with these evolved requirements in modern data processing. We as a field must stop trying to groom unbounded datasets into finite pools of information that eventually become complete, and instead live and breathe under the assumption that we will never know if or when we have seen all of our data, only that new data will arrive, old data may be retracted, and the only way to make this problem tractable is via principled abstractions that allow the practitioner the choice of appropriate tradeoffs along the axes of interest: correctness, latency, and cost.

In this paper, we present one such approach, the Dataflow Model¹, along with a detailed examination of the semantics it enables, an overview of the core principles that guided its design, and a validation of the model itself via the real-world experiences that led to its development.

¹We use the term “Dataflow Model” to describe the processing model of Google Cloud Dataflow [20], which is based upon technology from FlumeJava [12] and MillWheel [2].

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/3.0/>. Obtain permission prior to any use beyond those covered by the license. Contact copyright holder by emailing info@vldb.org. Articles from this volume were invited to present their results at the 41st International Conference on Very Large Data Bases, August 31st - September 4th 2015, Kohala Coast, Hawaii.

Proceedings of the VLDB Endowment, Vol. 8, No. 12
Copyright 2015 VLDB Endowment 2150-8097/15/08.

1. INTRODUCTION

Modern data processing is a complex and exciting field. From the scale enabled by MapReduce [16] and its successors (e.g. Hadoop [4], Pig [18], Hive [29], Spark [33]), to the vast body of work on streaming within the SQL community (e.g. query systems [1, 14, 15], windowing [22], data streams [24], time domains [28], semantic models [9]), to the more recent forays in low-latency processing such as Spark Streaming [34], MillWheel, and Storm [5], modern consumers of data yield remarkable amounts of power in shaping and taming massive-scale disorder into organized structures with far greater value. Yet, existing models and systems still fall short in a number of common use cases.

Consider an initial example: a streaming video provider wants to monetize their content by displaying video ads and billing advertisers for the amount of advertising watched. The platform supports online and offline views for content and ads. The video provider wants to know how much to bill each advertiser each day, as well as aggregate statistics about the videos and ads. In addition, they want to efficiently run offline experiments over large swaths of historical data.

Advertisers/content providers want to know how often and for how long their videos are being watched, with which content/ads, and by which demographic groups. They also want to know how much they are being charged/paid. They want all of this information as quickly as possible, so that they can adjust budgets and bids, change targeting, tweak campaigns, and plan future directions in as close to real time as possible. Since money is involved, correctness is paramount.

Though data processing systems are complex by nature, the video provider wants a programming model that is simple and flexible. And finally, since the Internet has so greatly expanded the reach of any business that can be parceled along its backbone, they also require a system that can handle the diaspora of global scale data.

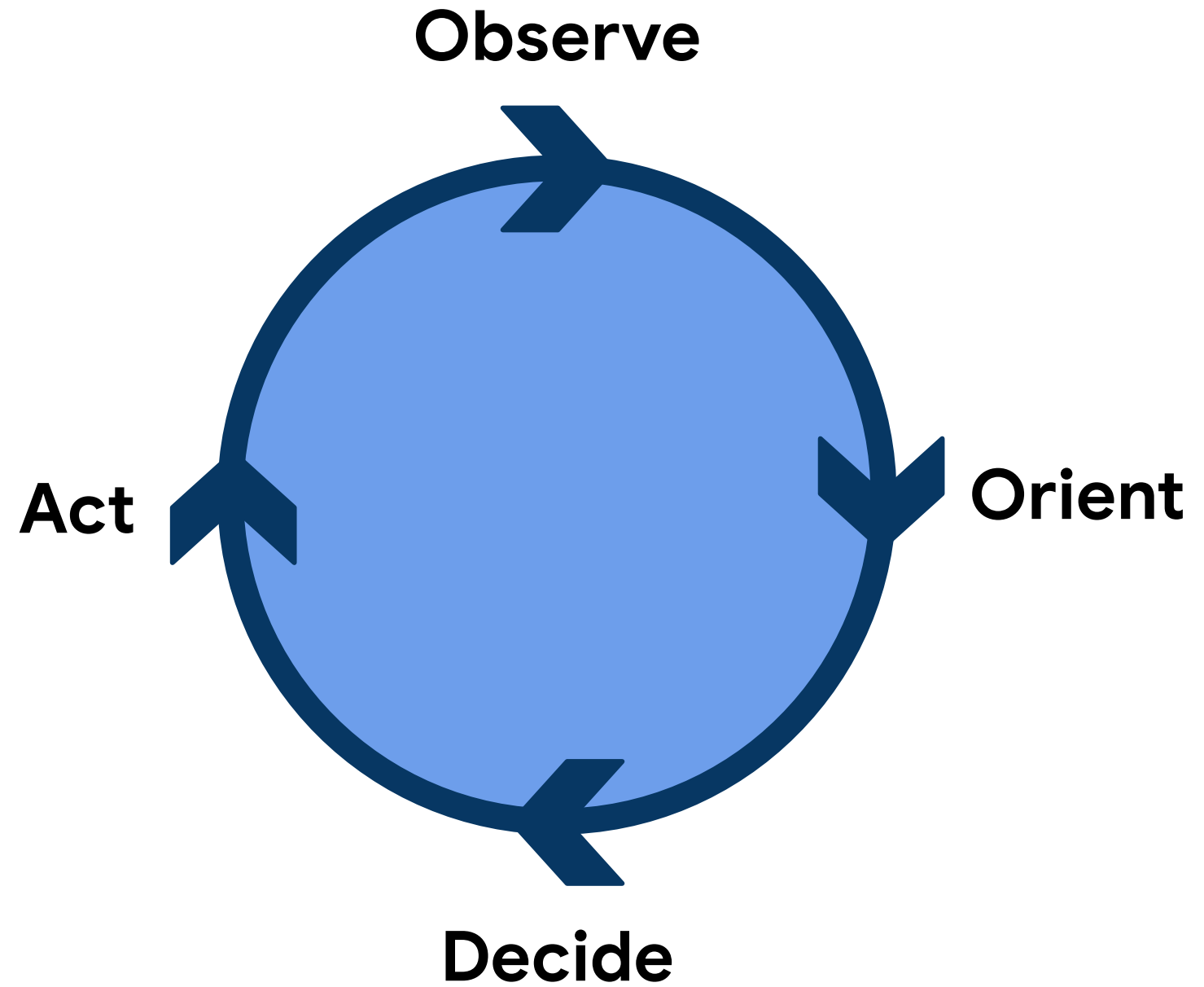
The information that must be calculated for such a use case is essentially the time and length of each video viewing, who viewed it, and with which ad or content it was paired (i.e. per-user, per-video viewing sessions). Conceptually this is straightforward, yet existing models and systems all fall short of meeting the stated requirements.

Batch systems such as MapReduce (and its Hadoop variants, including Pig and Hive), FlumeJava, and Spark suffer

Data is Everything

organizations win or lose based on how they use it

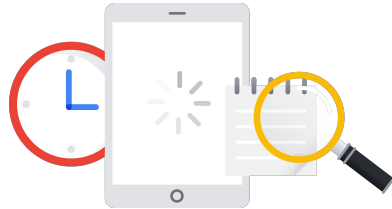
Boyd's OODA Loop



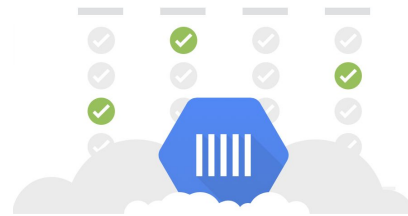
Our data analytics design principles



Focus on serverless
analytics not
infrastructure



Develop integrated
solutions



Operationalize
machine learning



Iterate and
measure

Government Leaders Face Challenges in Delivering Critical Insights

Enterprise demand
exceeds IT capacity
& funding



Increasing volumes
overrun capacity &
operating budgets



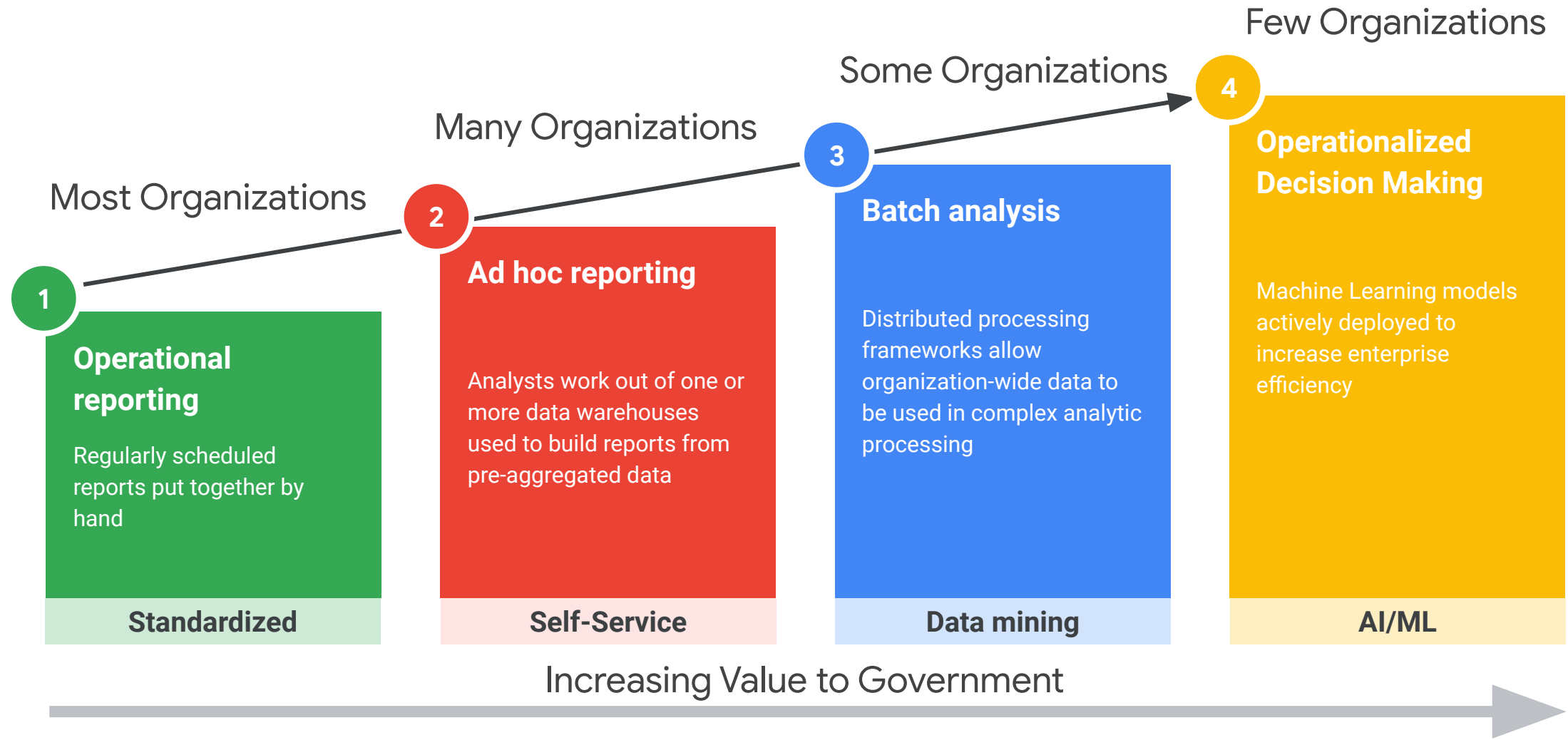
Data is scattered
across multiple
platforms



Security is inadequate to
protect the organization
and the mission



Government has valuable data to use



Opportunities

- Increase operational efficiencies
- Provide critical cyber security insights
- Foster civic engagement

Innovators in Transportation



Google Maps

Organizations need reliable, real-time data within user-friendly interfaces to empower their employees and keep up with their customers' expectations. Google Maps offers visualization, navigation, and analytics to help leading enterprises chart new territory and find transformative new opportunities.



Waze

Nobody knows what's happening in your city better than the people who live there. Waze exchanges publicly-available incidents and slow-down data, enabling our government partners to respond more immediately to accidents and analyze congestion on their roads.



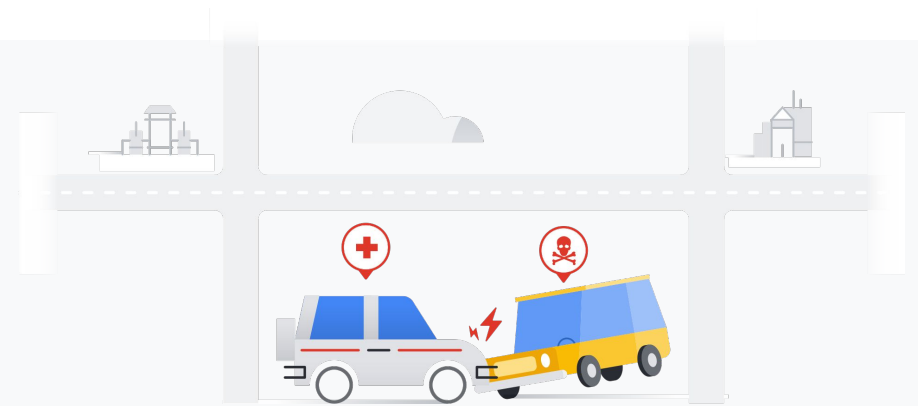
Waymo

Driving today is not as safe or enjoyable as it should be. Waymo's mission is to make it safe and easy for people and things to move around. They aim to bring fully self-driving technology to the world that can improve mobility by giving people the freedom to get around and help save thousands of lives now lost to traffic accidents.

Roads are busier, heavily congested...



The average commuter in metropolitan areas experience **4 hours of road congestion** every day.¹



...and dangerous

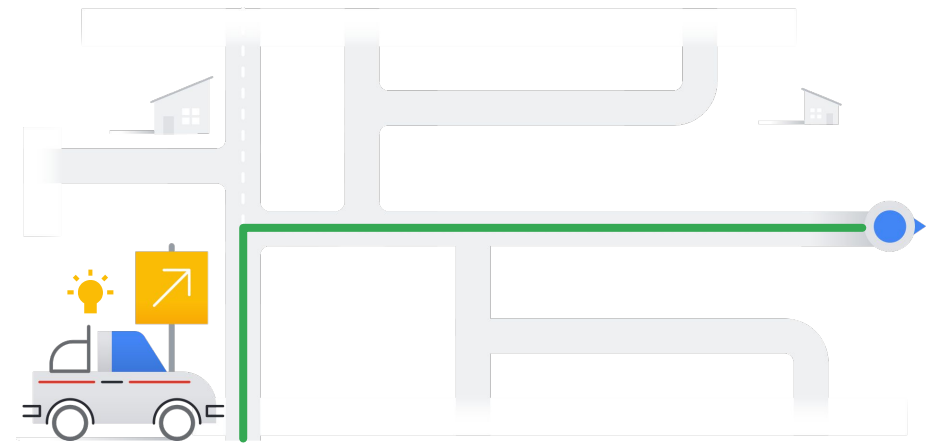
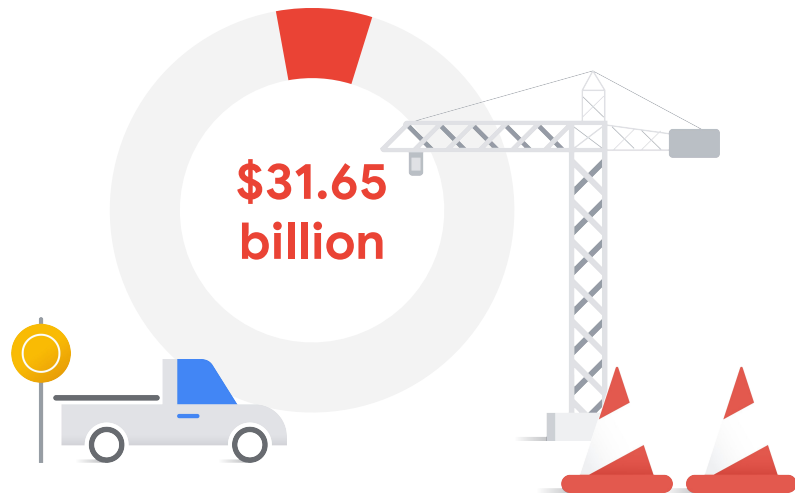
Current trends show that by **2030**, road traffic injuries will become the seventh leading cause of death globally.²

1. U.S. Department of Transportation, Federal Highway Administration Congestion Trends Report

2. CDC

The solution isn't building more roads

State and local government construction costs rose **13%** in the last five years¹



It's harnessing your data to **optimize those roads**



1. U.S. Census Bureau, Seasonally Adjusted Data

Traffic on 405 Fwy Is Worse 5 Years After \$1 Billion Widening Project in Sepulveda Pass: Study

POSTED 8:16 AM, MAY 7, 2019, BY [ERIC SPILLMAN](#), [TRACY BLOOM](#) AND [STEVE KUZJ](#), UPDATED AT 02:40PM, MAY 7, 2019

FACEBOOK

TWITTER

LINKEDIN

PINTEREST

EMAIL




Despite a \$1 billion widening project to improve traffic along a 10-mile stretch of the 405 Freeway that connects the San Fernando Valley and West Los Angeles, traffic is worse now in the Sepulveda Pass than it was when construction was completed several years ago.



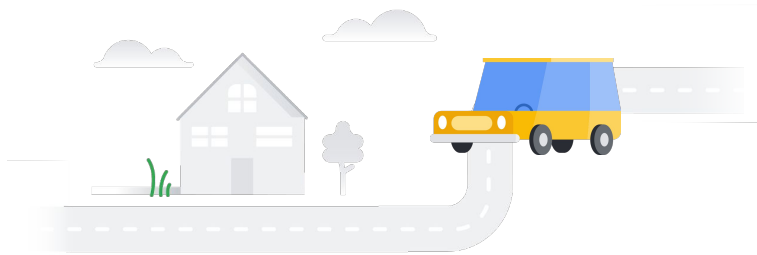
But that data exists in **silos**,
making it difficult to use



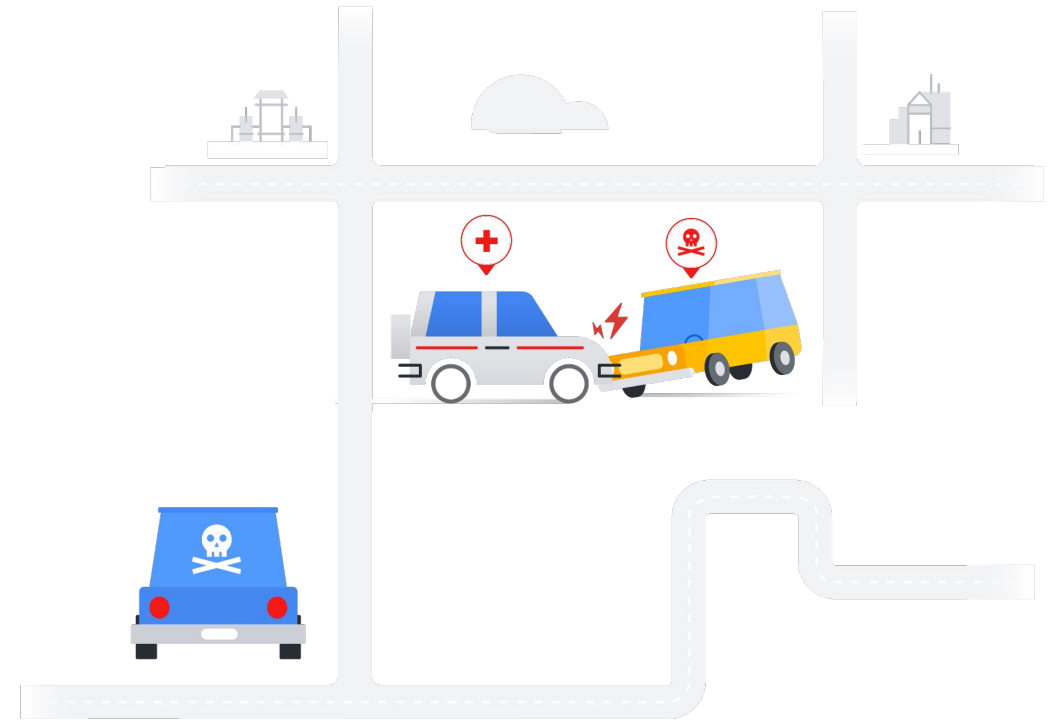
The image shows a collection of overlapping gray rectangular windows, each containing a different blue icon representing various data sources and services. The icons include a puzzle piece, a list, a Waze logo, a document, a cloud download, a Google Maps logo, a magnifying glass, a target, a folder, a share symbol, a line graph, a bar chart, and a pie chart.

 Google Cloud

Disconnected data is costing us too much



NSC estimates the cost of motor-vehicle deaths, injuries, and property damage in 2016 was **\$416.2 billion**.¹

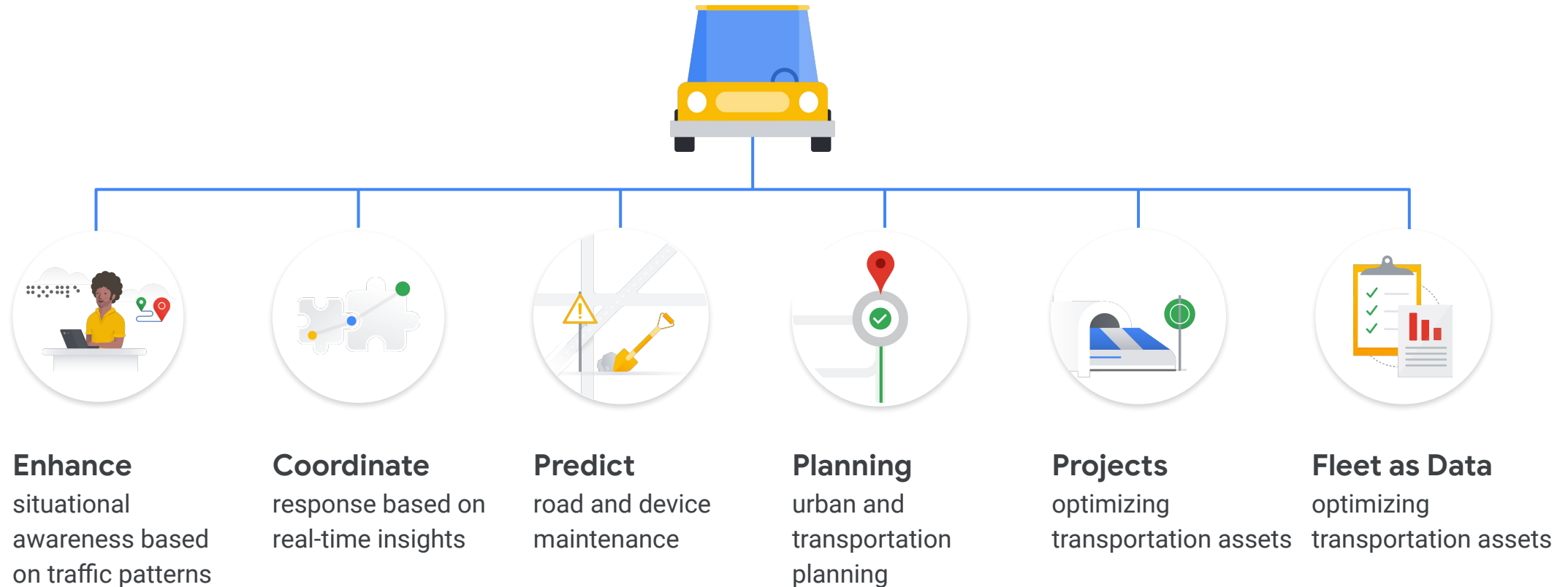


More than **40,000** people died in motor vehicle crashes in 2017.²

1. National Safety Council

2. US Department of Transportation's National Highway Traffic Safety Administration

What if your data could be the **driver** of traffic management?





DAISy is a cloud-based data analytics platform that brings intelligence, efficiency, and interoperability to CDOT's existing transportation network while **enabling world-leading roadway operations for a safer, more reliable, connected, and autonomous future.**



Ingest

Get **petabytes** of data in
from a **variety of
formats**.



Transform

Prepare, clean, and
transform data **quickly
and easily**.



Store

Create, save, and
store datasets
inexpensively.



Analyze

Derive data insights
**at scale without
managing servers**.



Interact

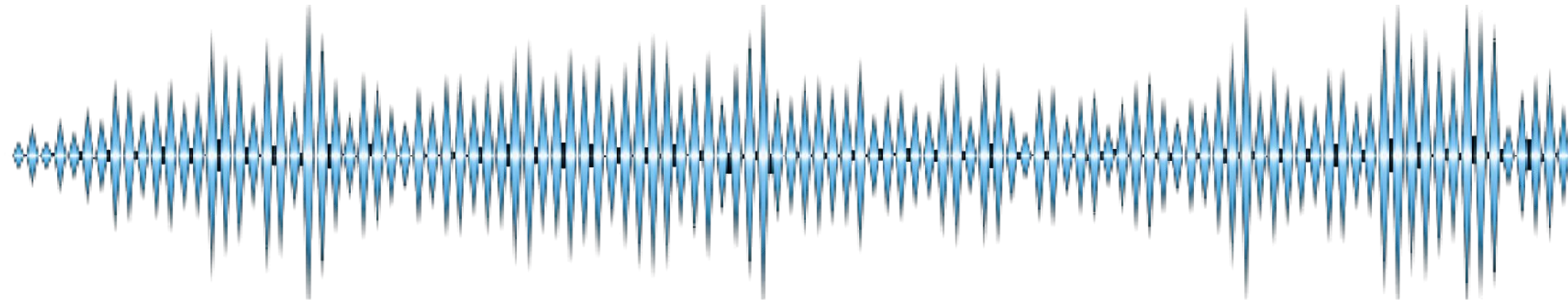
Explore and
present **interactive and
impactful** data insights.

Conversational Interfaces

Speech to text queries powered by machine learning

What is the average age of the water mains in town?

How many homes are on Santa Paula Avenue?



How many bikes per day use the bridge over Stevens Creek Blvd?

How many properties added pools last year?



Google

Thank You



Boulder AI

January 2020

Agenda

What we do

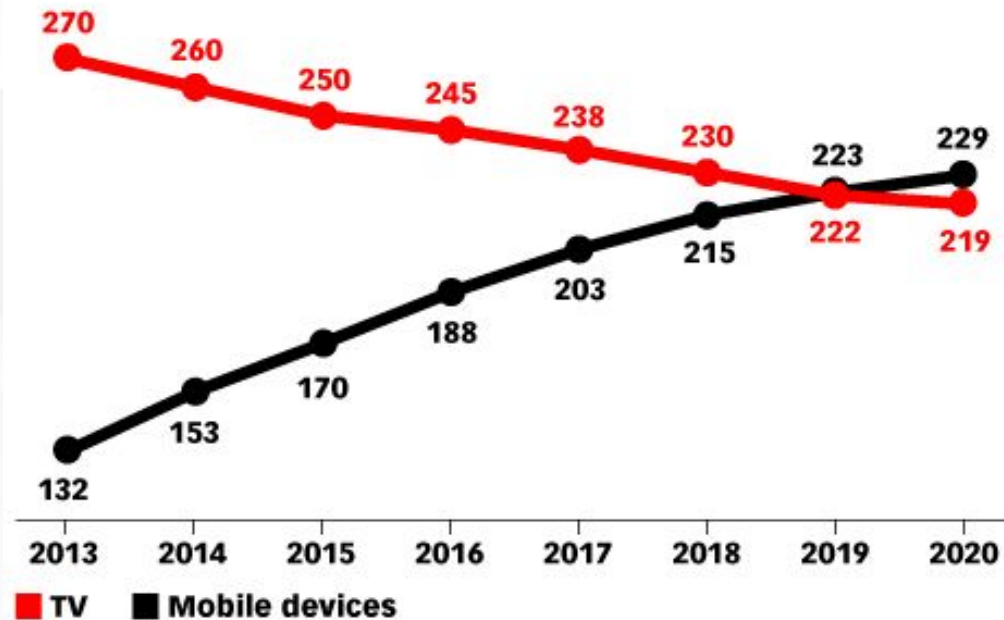
Problems we're solving with Cupertino's peers

What is Cupertino trying to solve?

How can city ops and capital keep up with technology?

Average Time Spent per Day with TV and Mobile Devices by US Adults, 2013-2020

minutes



Note: ages 18+; time spent with each medium includes all time spent with that medium, regardless of multitasking; for example, 1 hour of multitasking on a mobile device while watching TV is counted as 1 hour for TV and 1 hour for mobile device
Source: eMarketer, April 2018

238500

www.eMarketer.com



IoT can reduce fatalities, but existing options don't scale



~80%

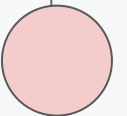
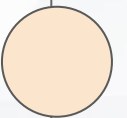
Low detection accuracy

\$10k-\$20k

High price per device

Single Purpose

Siloed object detection capabilities

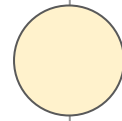


Boulder AI is changing what cities expect of the IoT.



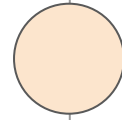
Boulder AI DNN Cameras:

- Edge to cloud; no additional hardware.
- Privacy first sensors. Metadata anonymized at the source.
- High compute. General purpose deep learning. Continuously improving detection



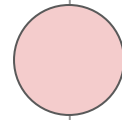
90%+

High detection and tracking accuracy



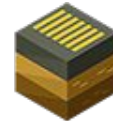
Priced to scale

Volume pricing for PoC and scaled deployments



Many objects

Detect, track and count many city objects with one device



1

Sensors acquire raw video



2

On-camera analytics extract valuable metadata from video

Pedestrians

Cars

Bikes

Freight

Public Transport

Carts

Scooters

Strollers

- Boulder AI standard hardware platform (DNN Cameras)

- Object detection, counts, position and speed data.
- Live and recorded video

3

Boulder AI structures software services on metadata from video



Near misses

Emergency response

Intersection timing

Pedestrian safety

Sanitation demand

Lane occupancy

Traffic mix

HAZMAT Vehicles

Rideshare compliance

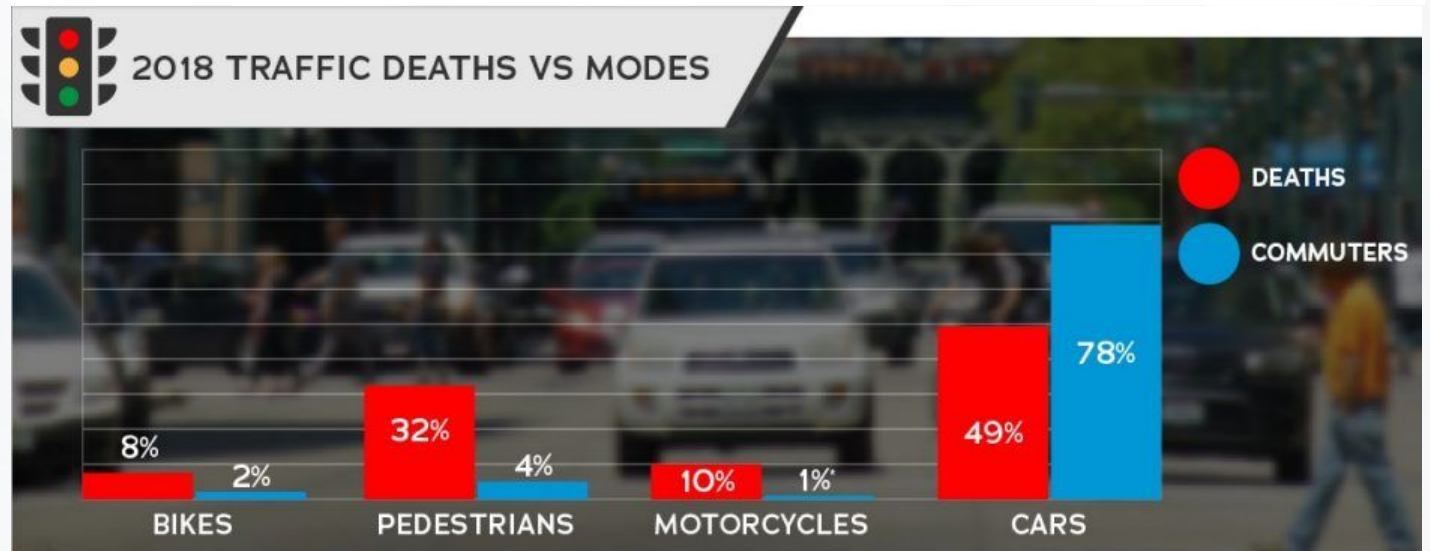
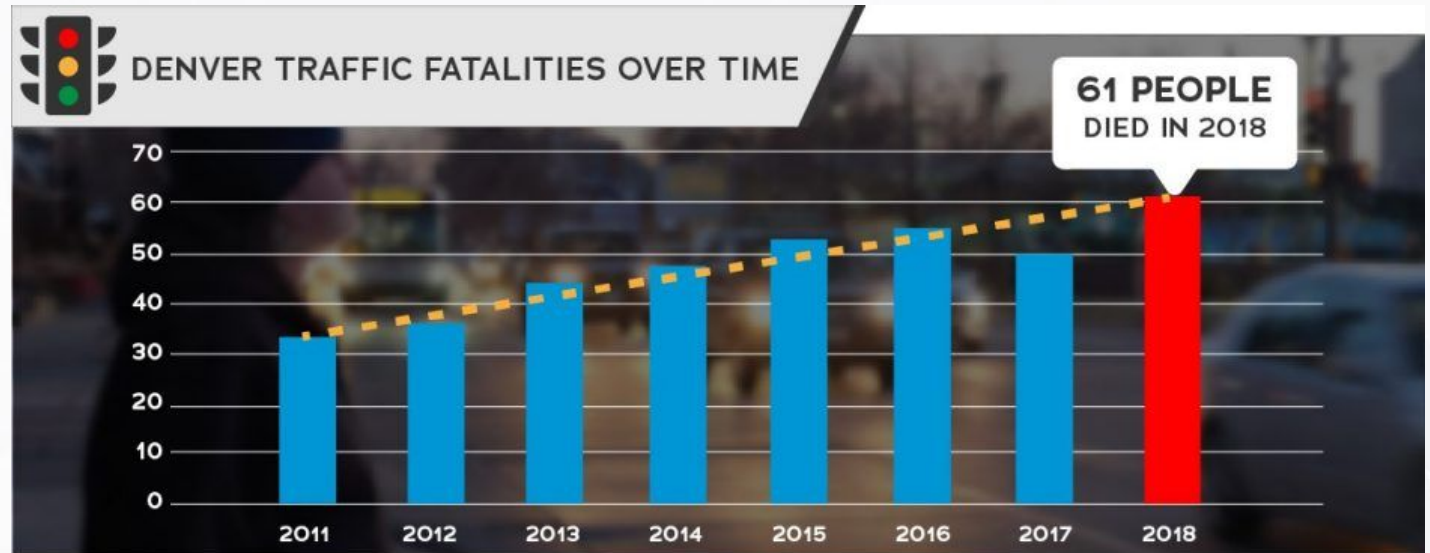
Parking occupancy

Cyclist/scooter safety

- Software (edge or cloud) built on metadata or additional analytics

Making Denver safer is a complicated challenge.

- **Growth.**
2x population increase since 1990 (Denver Metro Area)
- **Multimodal.**
Massive rideshare adoption, changing multimodal fleet.
- **Technology.**
Distracted drivers on busier intersections.

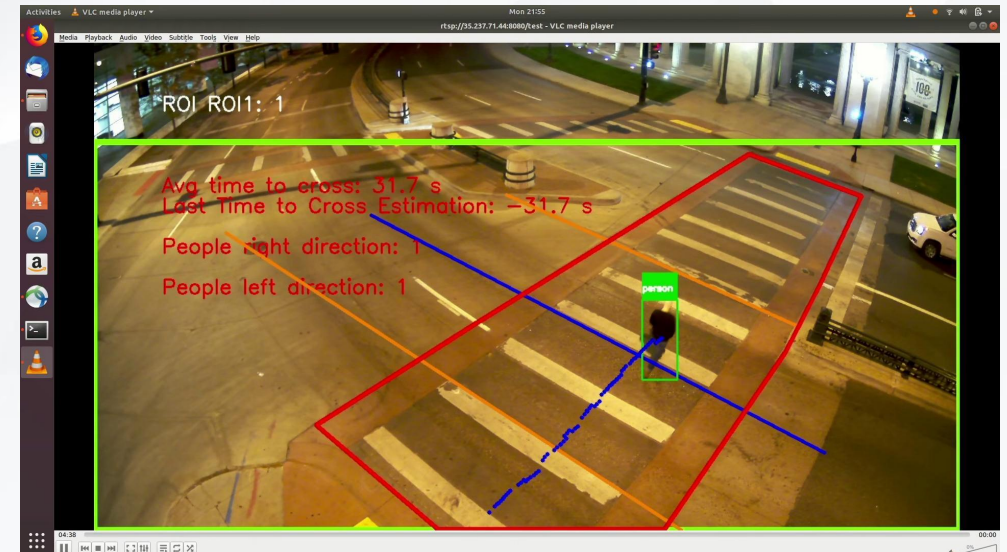
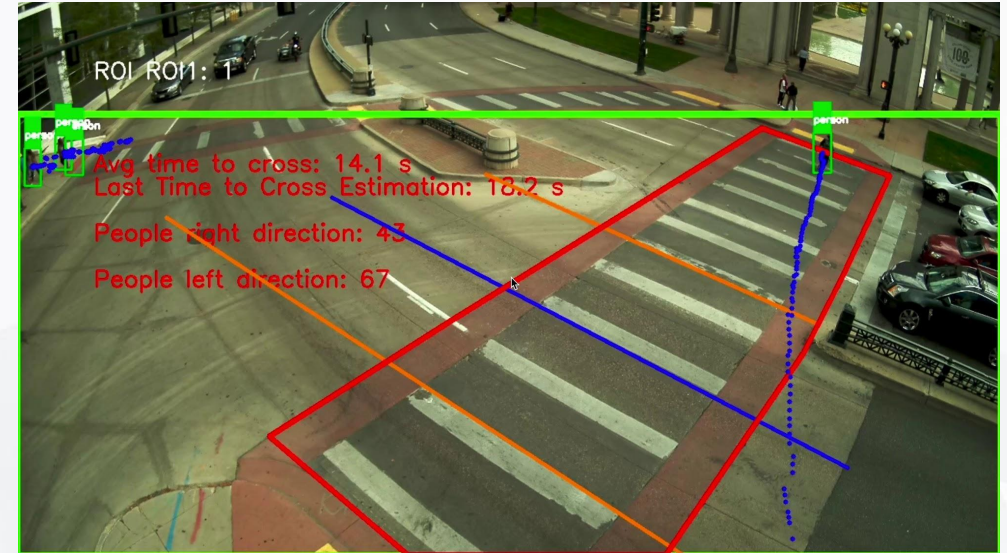
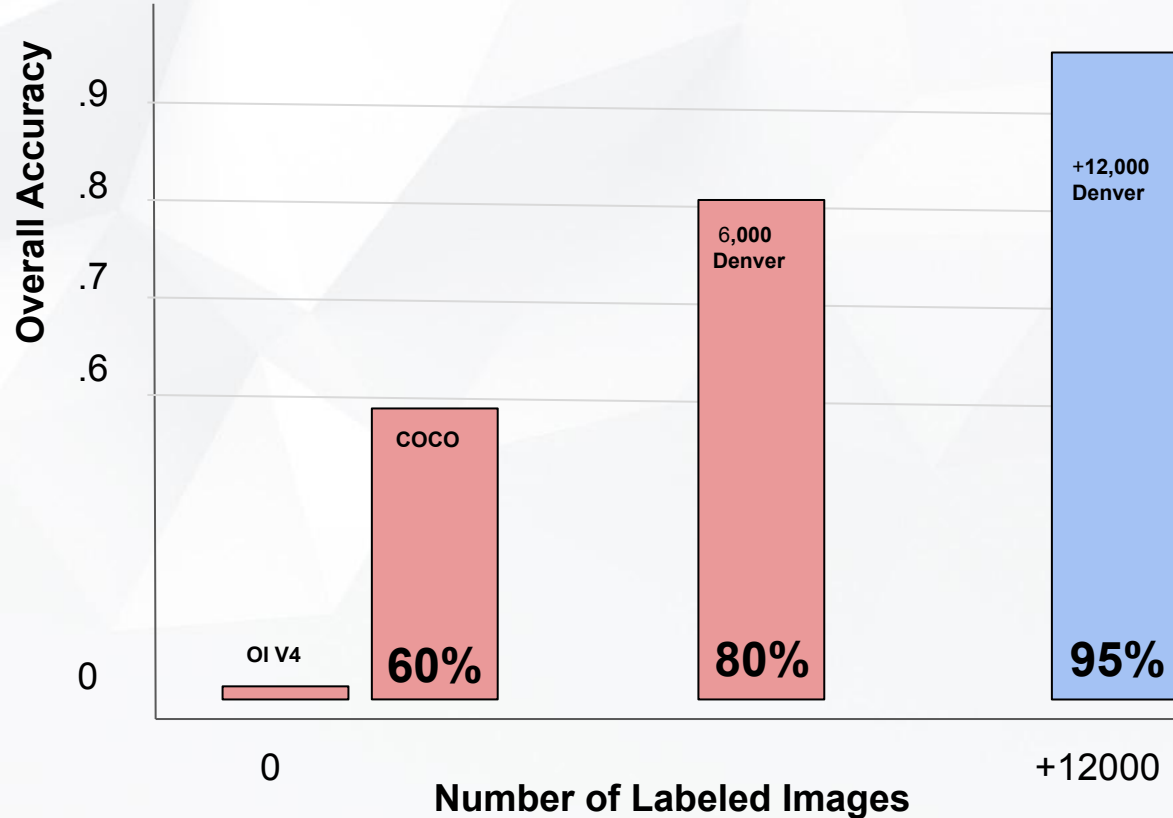


*Motorcycles are included in a larger percentage of "Other" modes of transportation.

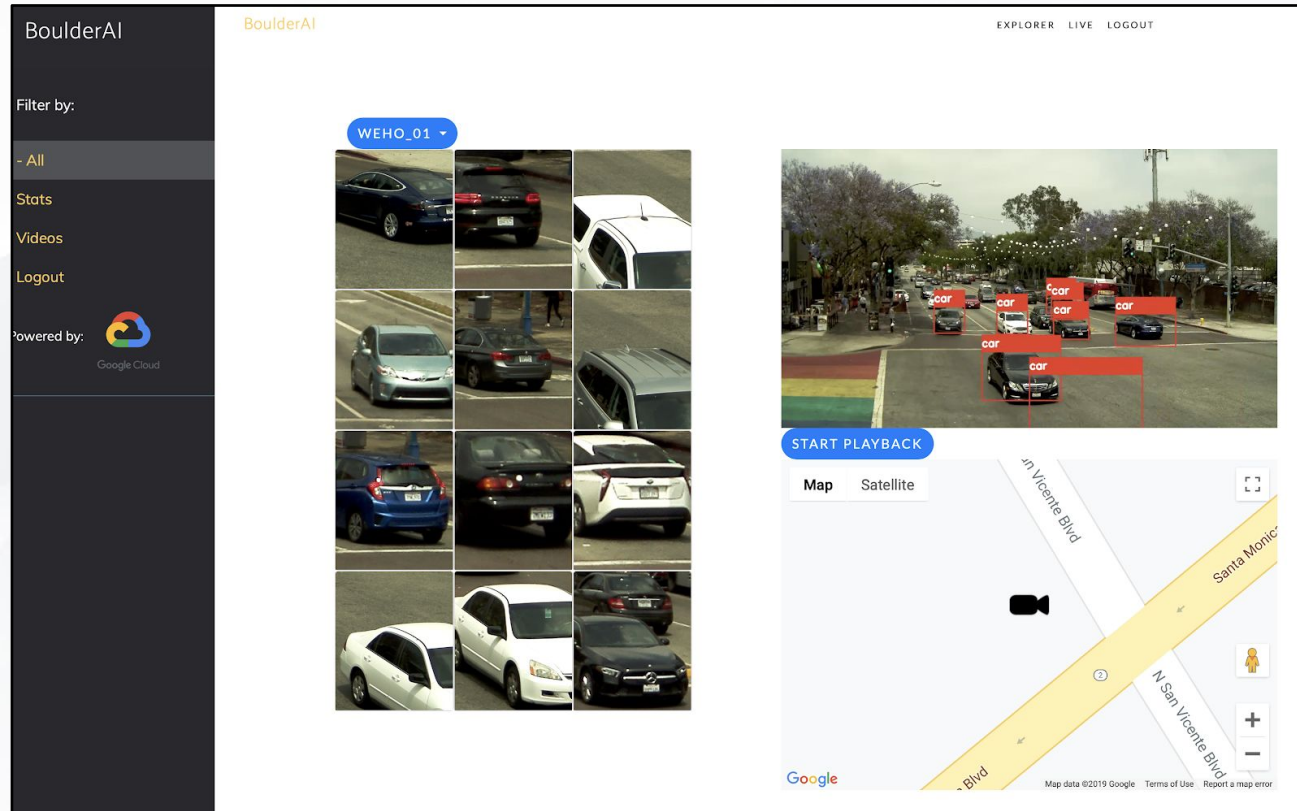
Denver PoC: connected pedestrians and signaling

3 month milestones:

- Determine ped crosswalk occupancy
- Count and track peds (anonymously)
- Make live data available to signal controller



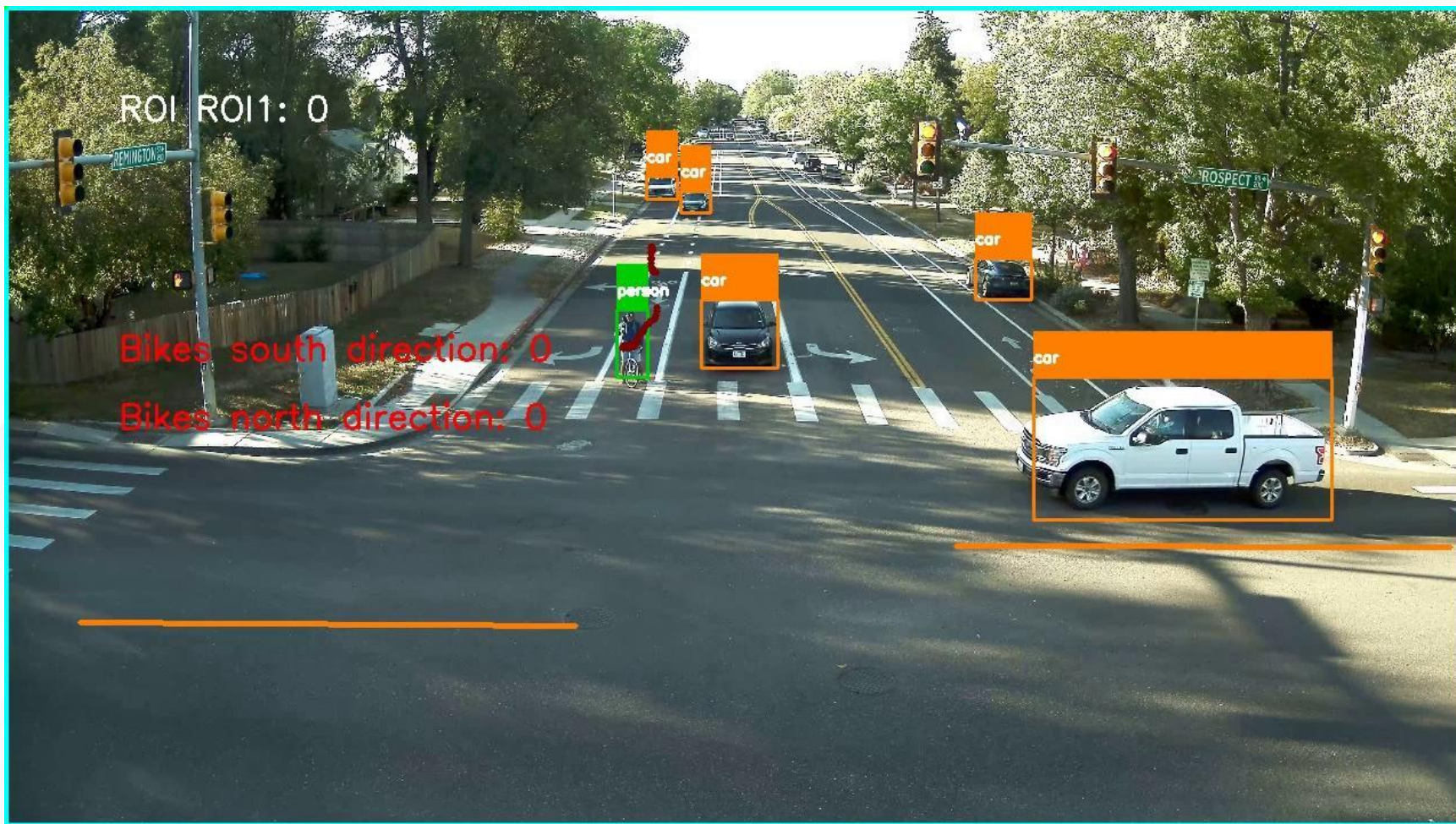
West Hollywood: Searchable Objects



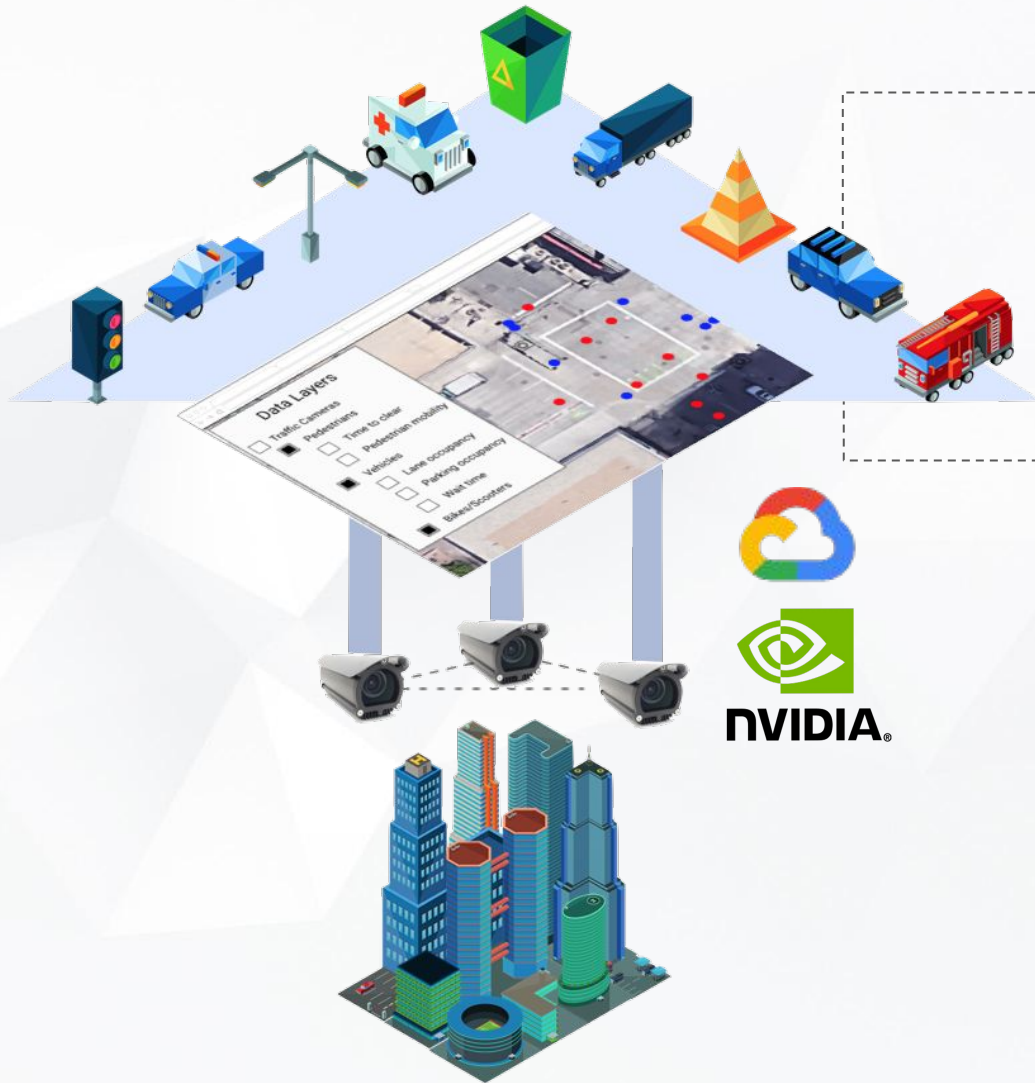
Combining the best of edge and cloud:

- Search objects, not video.
- Explore your data in a rich dashboard with real time video, real-time GIS object view.
- Download video clips of important events for context.

Fort Collins Cyclist Safety



Our vision for cities: one device, many services



1

Camera networks provide metadata

Networks of Boulder AI cameras provide private data about objects, vehicles, speed, and position.

2

Layer services on top of metadata

Rideshare cars and scooters

Pedestrian needs

Intersection danger

Emergency response

Sanitation service demand

Bicycle safety

Lane occupancy

Traffic mix

Hazards in road

Parking occupancy and violations

Intersection timing

HAZMAT vehicles

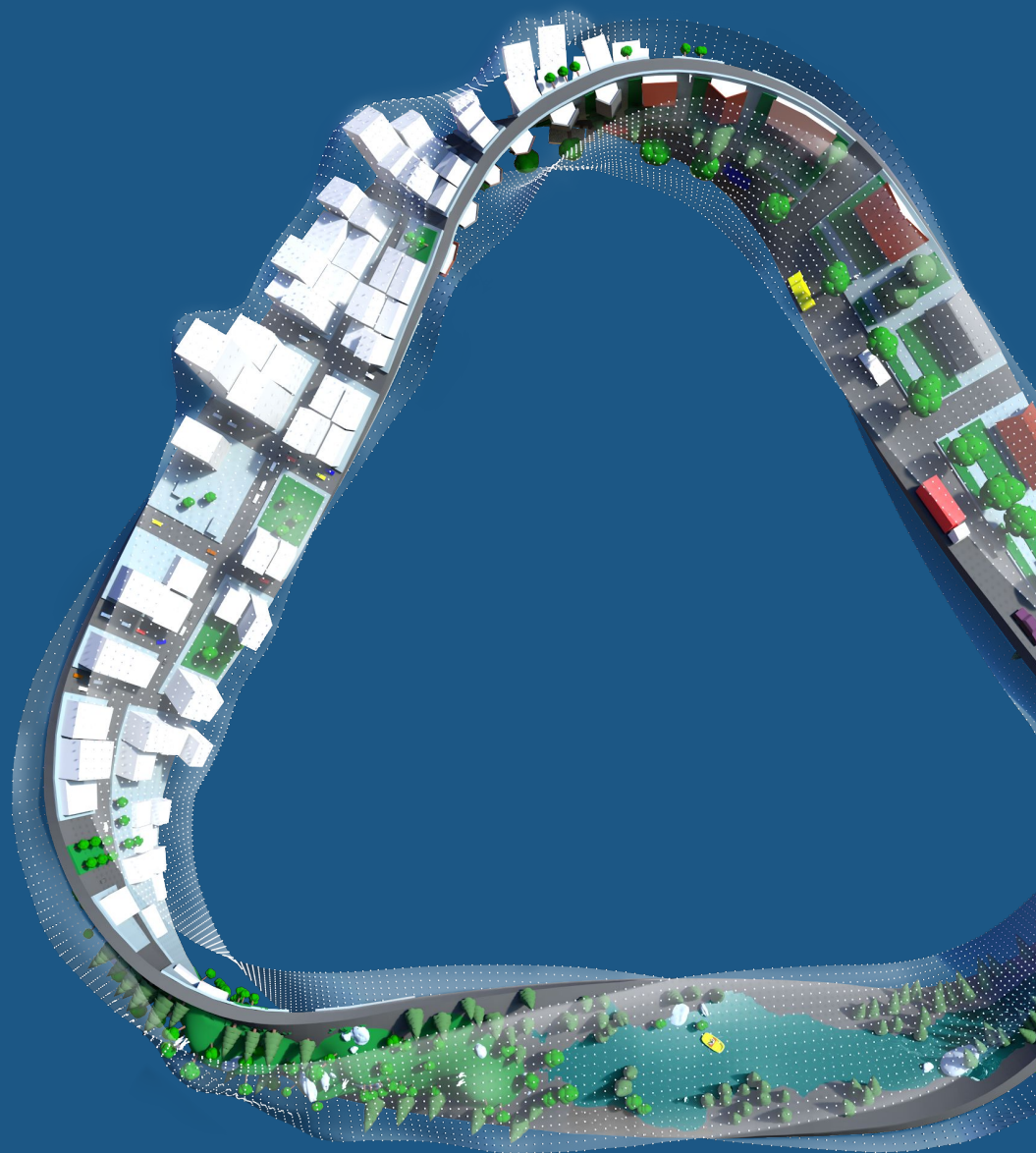
3

Help Cupertino define a digital infrastructure for operations and growth

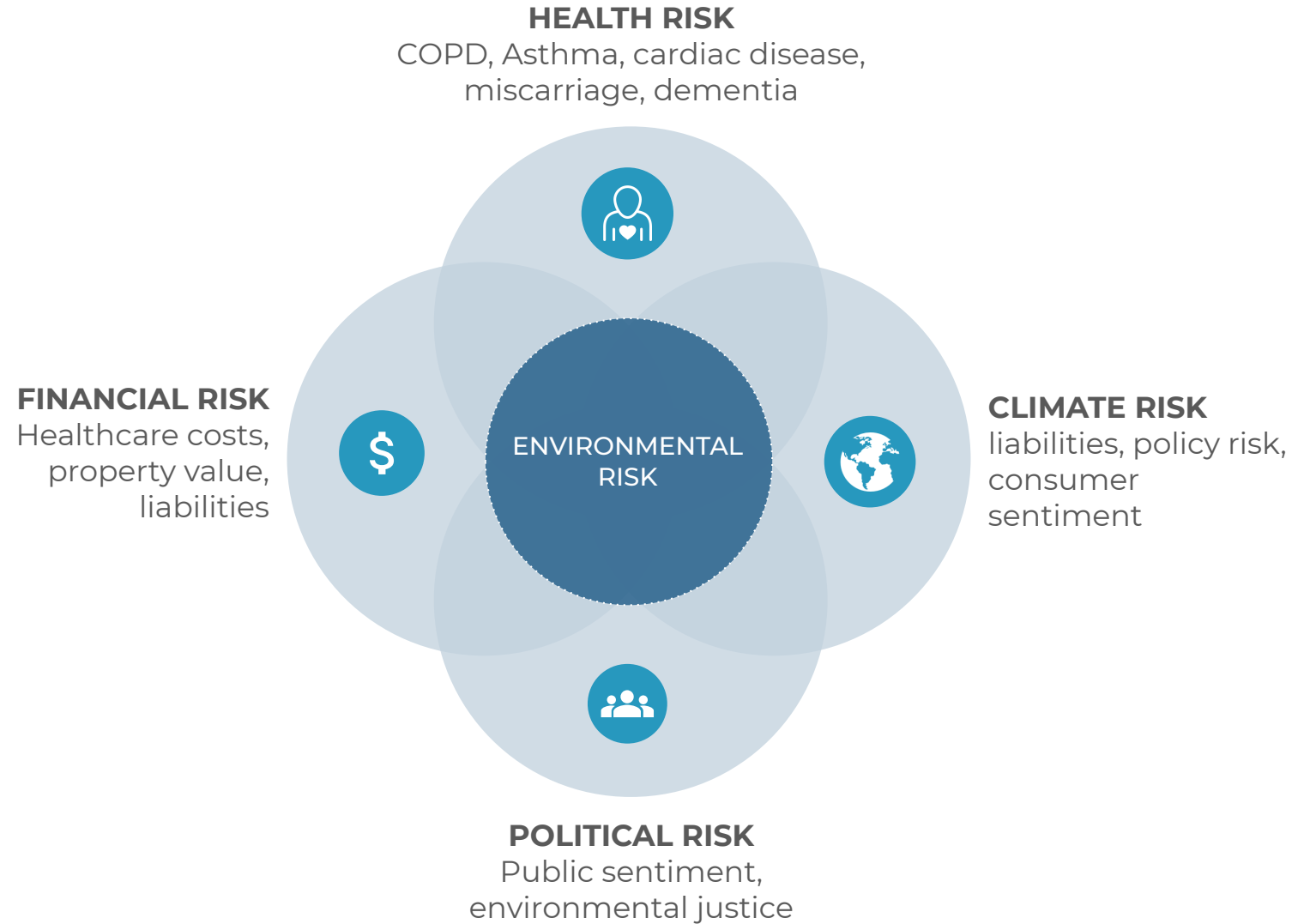


Environmental intelligence for people and the planet

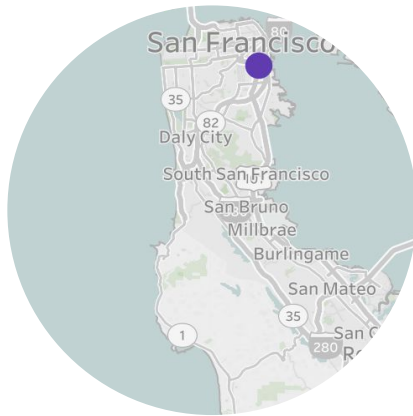
Aclima Introduction - Cupertino City Council
January 2020



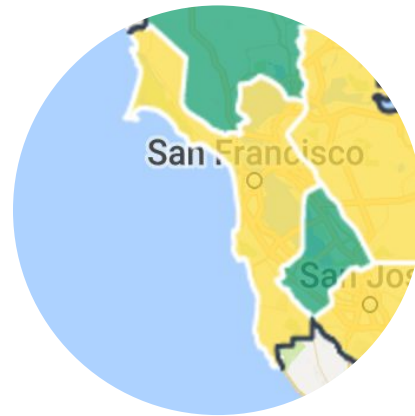
**92% of the
global
population
breathes
unhealthy air,
each day**



We can't manage what we don't measure



Limited coverage



Low resolution

Regulatory data today



Hyper-local is the future

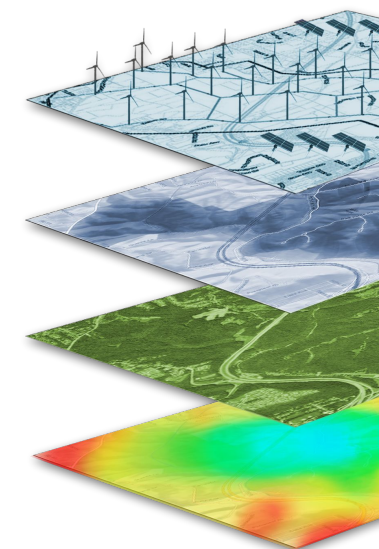
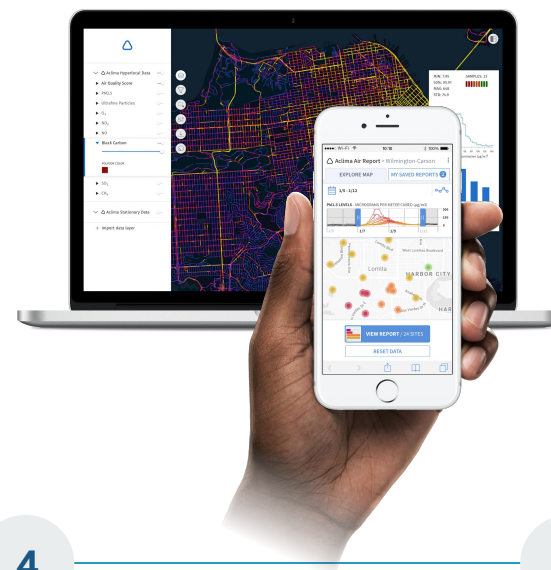
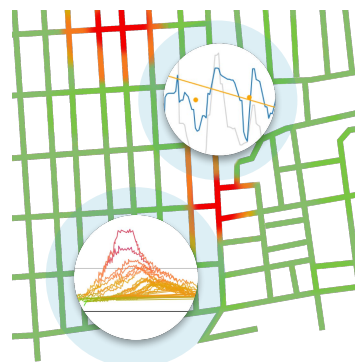
NO₂ concentration (ppb)



Air quality data from Google / Aclima

Google Earth

A unified platform



1

2

3

4

5

Mobile & Stationary Sensors

Best-in-class data quality

End-to-end network management

Unprecedented scale + block-by-block resolution, via multi-pass driving

Data Management + Analytics

Synthesis of Aclima and integrated third party data to derive insights

Software Tools

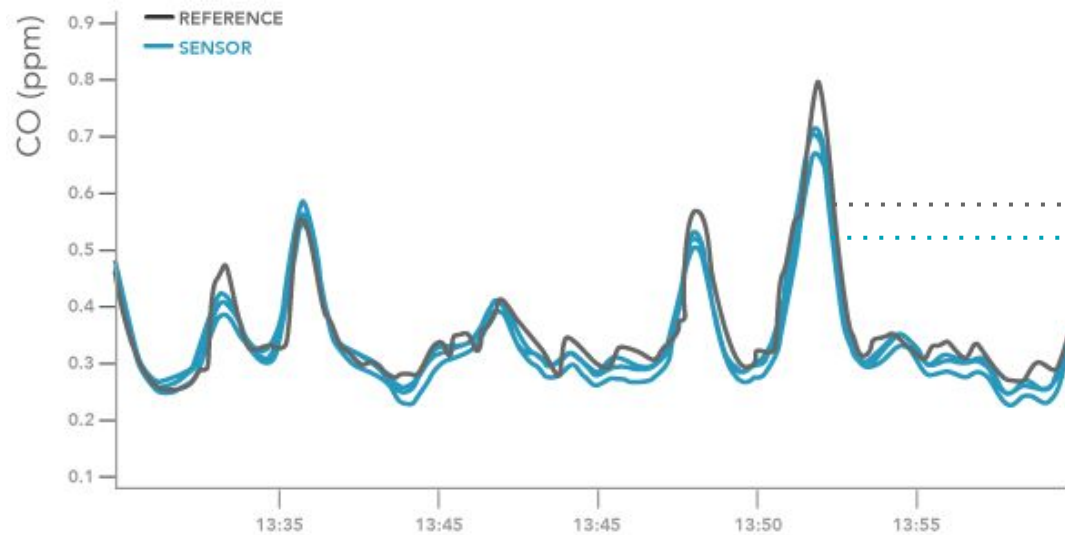
Intuitive software tools for experts and citizens to drive action

Data integration for diagnosis + action

Wind, land-use, health data and more

High quality data, lower cost sensors

Comparable Performance



REFERENCE
High-Cost, Low-Density



SENSOR
Low-Cost, High-Density



Backed by scientific rigor at every step



Carnegie Mellon University



Article

pubs.acs.org/est

Environmental Science & Technology

High-Resolution Air Pollution Mapping with Google Street View Cars: Exploiting Big Data

Joshua S. Apte,^{*,†,§} Kyle P. Messier,^{†,‡} Shahzad Gani,[†] Michael Brauer,[§] Thomas W. Kirchstetter,^{||} Melissa M. Lunden,[‡] Julian D. Marshall,[#] Christopher J. Portier,[‡] Roel C.H. Vermeulen,[∇] and Steven P. Hamburg[‡]

[†]Department of Civil, Architectural and Environmental Engineering, University of Texas at Austin, Austin, Texas 78712 United States

[‡]Environmental Defense Fund, New York, New York 10010 United States

[§]School of Population and Public Health, University of British Columbia, Vancouver V6T 1Z3 Canada

^{||}Energy Technologies Area, Lawrence Berkeley National Laboratory, Berkeley, California 94720 United States

[#]Aclima, Inc., 10 Lombard St., San Francisco, California 94111 United States

[∇]Department of Civil and Environmental Engineering, University of Washington, Seattle, Washington 98195 United States

[∇]Institute for Risk Assessment Science, Utrecht University, Utrecht 3584 CM Netherlands

Supporting Information

ABSTRACT: Air pollution affects billions of people worldwide, yet ambient pollution measurements are limited for much of the world. Urban air pollution concentrations vary sharply over short distances (≤ 1 km) owing to unevenly distributed emission sources, dilution, and physicochemical transformations. Accordingly, even where present, conventional fixed-site pollution monitoring methods lack the spatial resolution needed to characterize heterogeneous human exposures and localized pollution hotspots. Here, we demonstrate a measurement approach to reveal urban air pollution patterns at 4–5 orders of magnitude greater spatial precision than possible with current central-site ambient monitoring. We equipped Google Street View vehicles with a fast-response pollution measurement platform and repeatedly sampled every street in a 30-km² area of Oakland, CA, developing the largest urban air quality data set of its type. Resulting maps of annual daytime NO, NO₂, and black carbon at 30 m-scale reveal stable, persistent pollution patterns with surprisingly sharp small-scale variability attributable to local sources, up to 5–8 \times within individual city blocks. Since local variation in air quality profoundly impacts public health and environmental equity, our results have important implications for how air pollution is measured and managed. If validated elsewhere, this readily scalable measurement approach could address major air quality data gaps worldwide.



Diverse expertise in every domain

Diverse team comprised of domain experts from world-class organizations, with a history of building impactful solutions at scale.

Bound by a shared mission to build a global, multi-billion dollar business that drives humanity forward.

Diversity

40% women

50% women in leadership

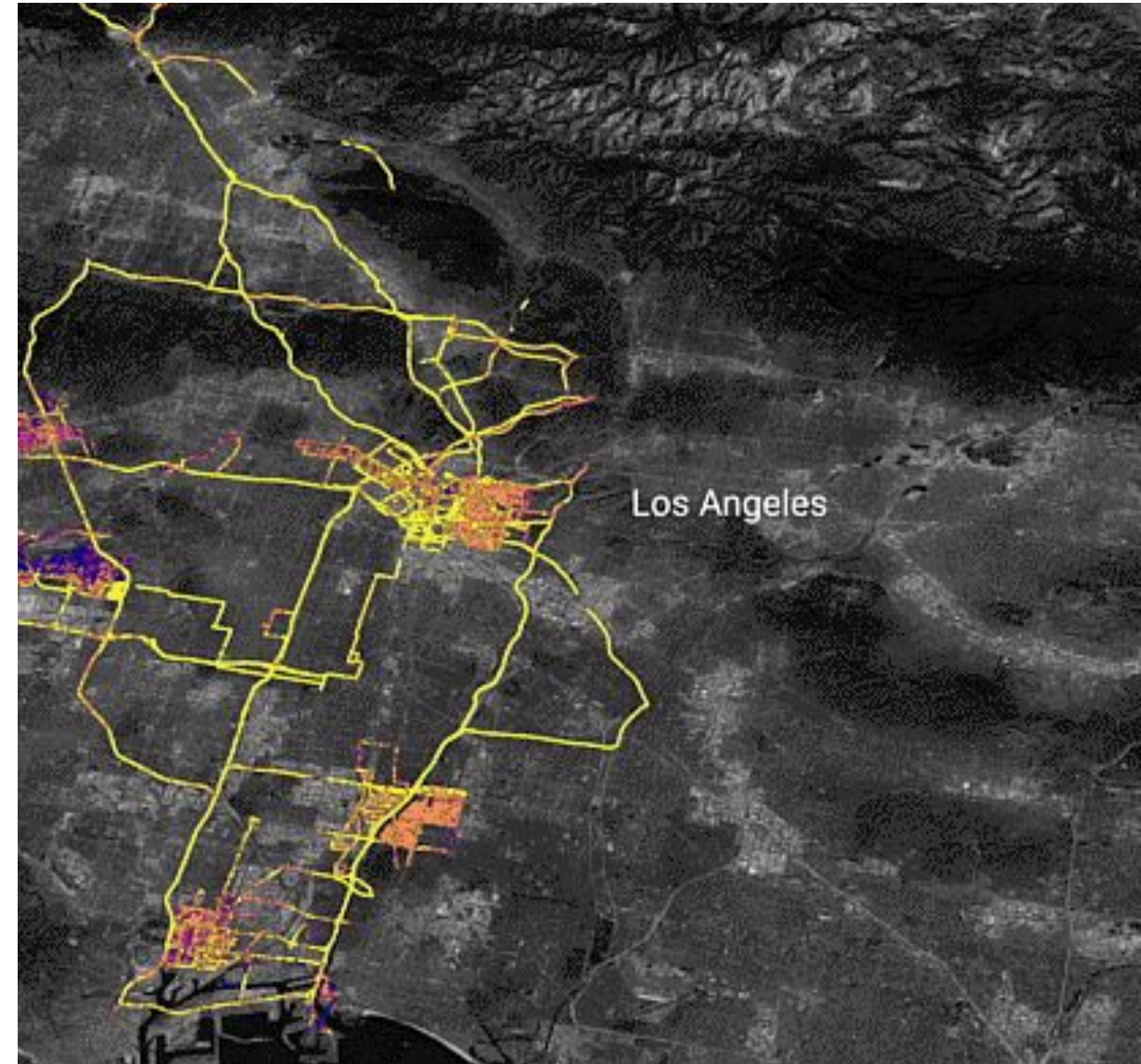


Experience



Measuring what matters

- Carbon Dioxide (CO₂)
 - Carbon Monoxide (CO)
 - Ozone (O₃)
 - Nitrogen Dioxide (NO₂)
 - Nitric Oxide (NO)
 - Particulate Matter (PM_{2.5})
-
- Black Carbon
 - Methane (CH₄)
 - Total Volatile Organic Compounds (TVOC)



Aclima advantage, from the user perspective

Customer problem: *“I need to understand hyper-local air quality across my city.”*

EXISTING TECH

STEPS (~months to years)

1. Determine what to measure, at appropriate levels of detection to answer user questions
2. Source and purchase suitable hardware from multiple vendors
3. Locate appropriate site(s) for continuous monitoring
4. Establish QA/QC plan
5. Permit and install hardware, protecting from the elements
6. Establish cloud based systems and scientific data structures to ingest, manage, calibrate, harmonize and analyze data
7. Hire trained atmospheric scientists to manage hardware
8. Hire data scientists to interpret results
9. Hire UI/UX designers to visualize large-scale scientific data sets
10. Hire field team to maintain, replace and calibrate sensors
11. Build atmospheric models to translate stationary data into high-spatial resolution outputs

ACLIMA

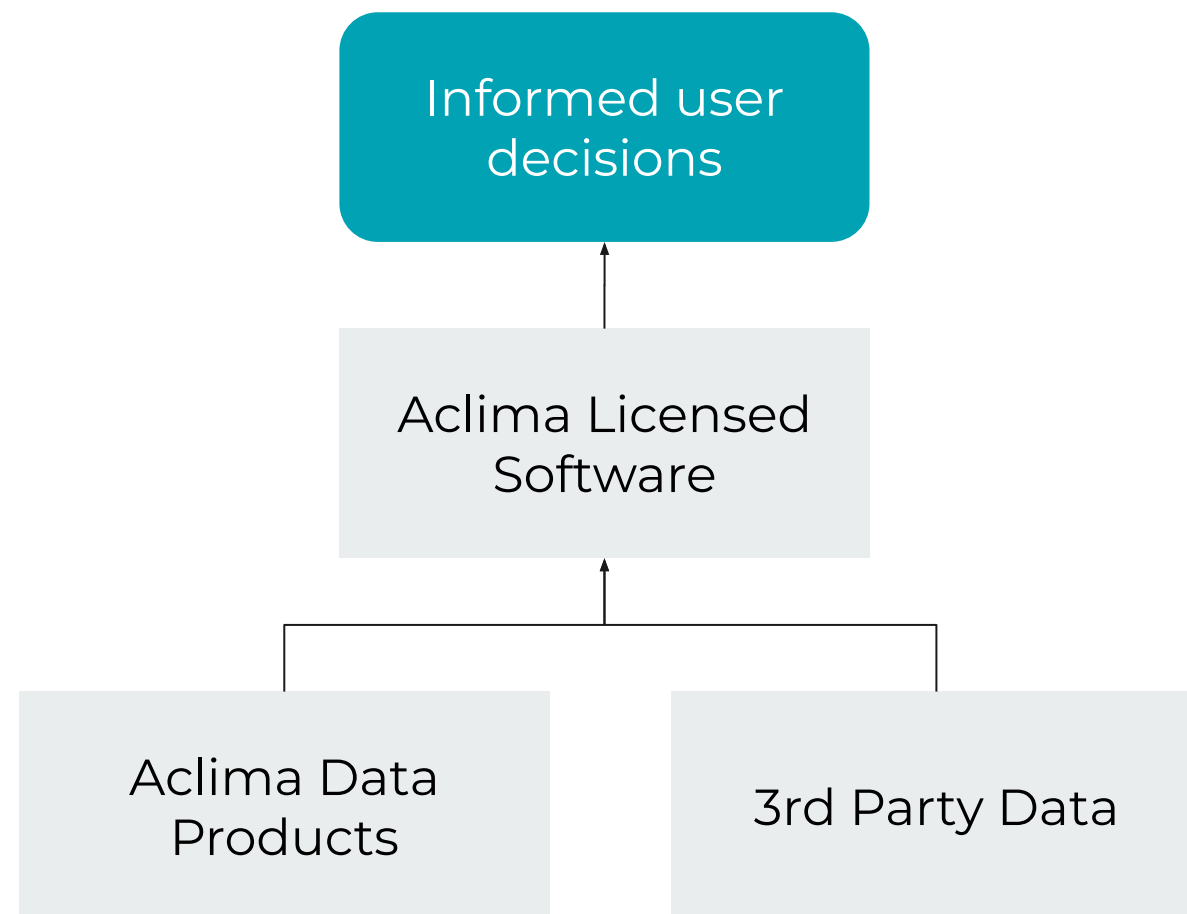
STEPS (~minutes)

1. Subscribe to Aclima Pro

Environmental Intelligence tools

Aclima SaaS products empower users with best-in-class environmental data and analysis tools

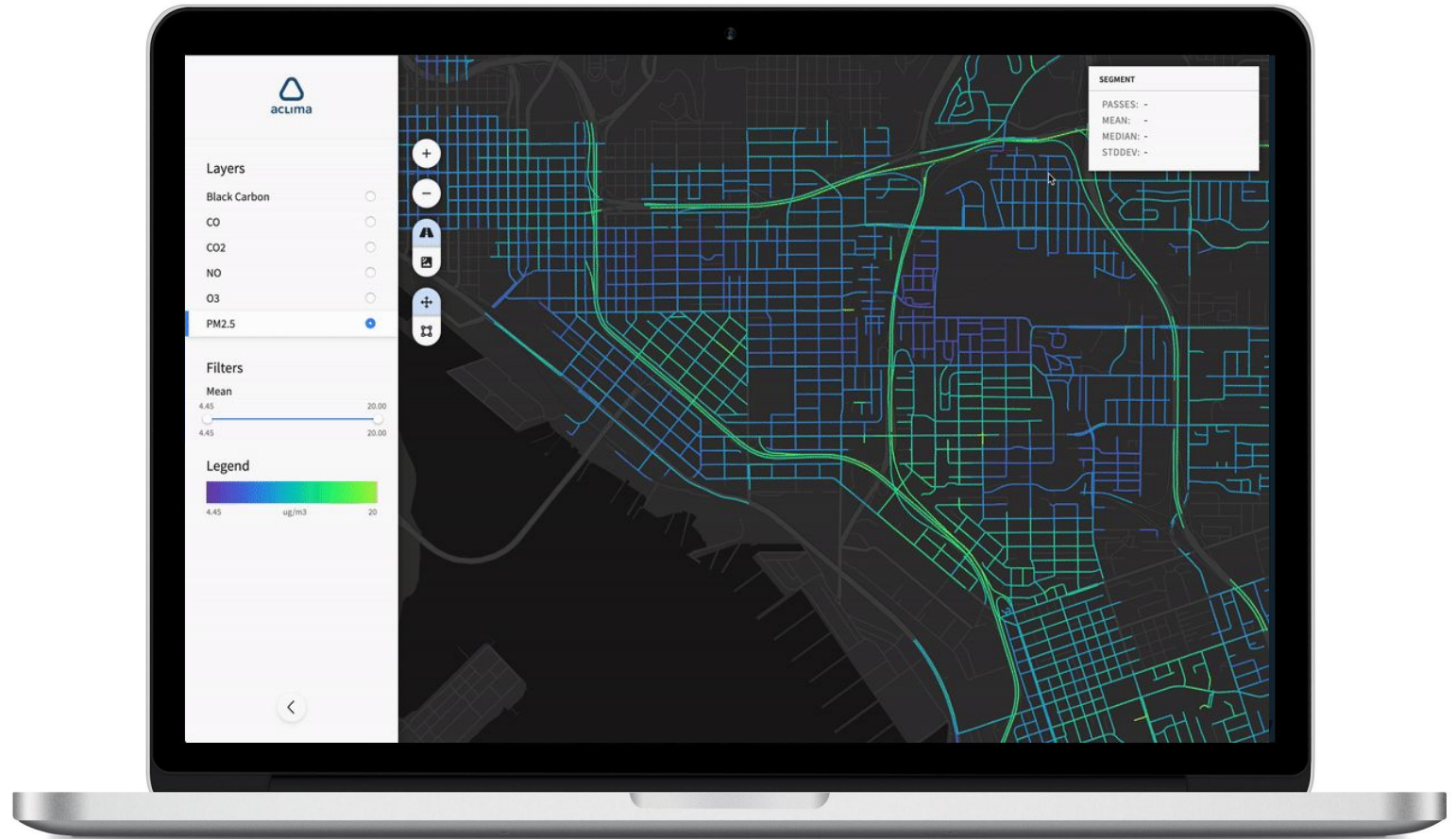
Licensed Aclima software applications are populated with Aclima hyper-local data, in addition to stationary air quality monitoring data from Aclima and 3rd party sources



Aclima Pro

Executives and technical staff at cities and regulatory agencies ask:

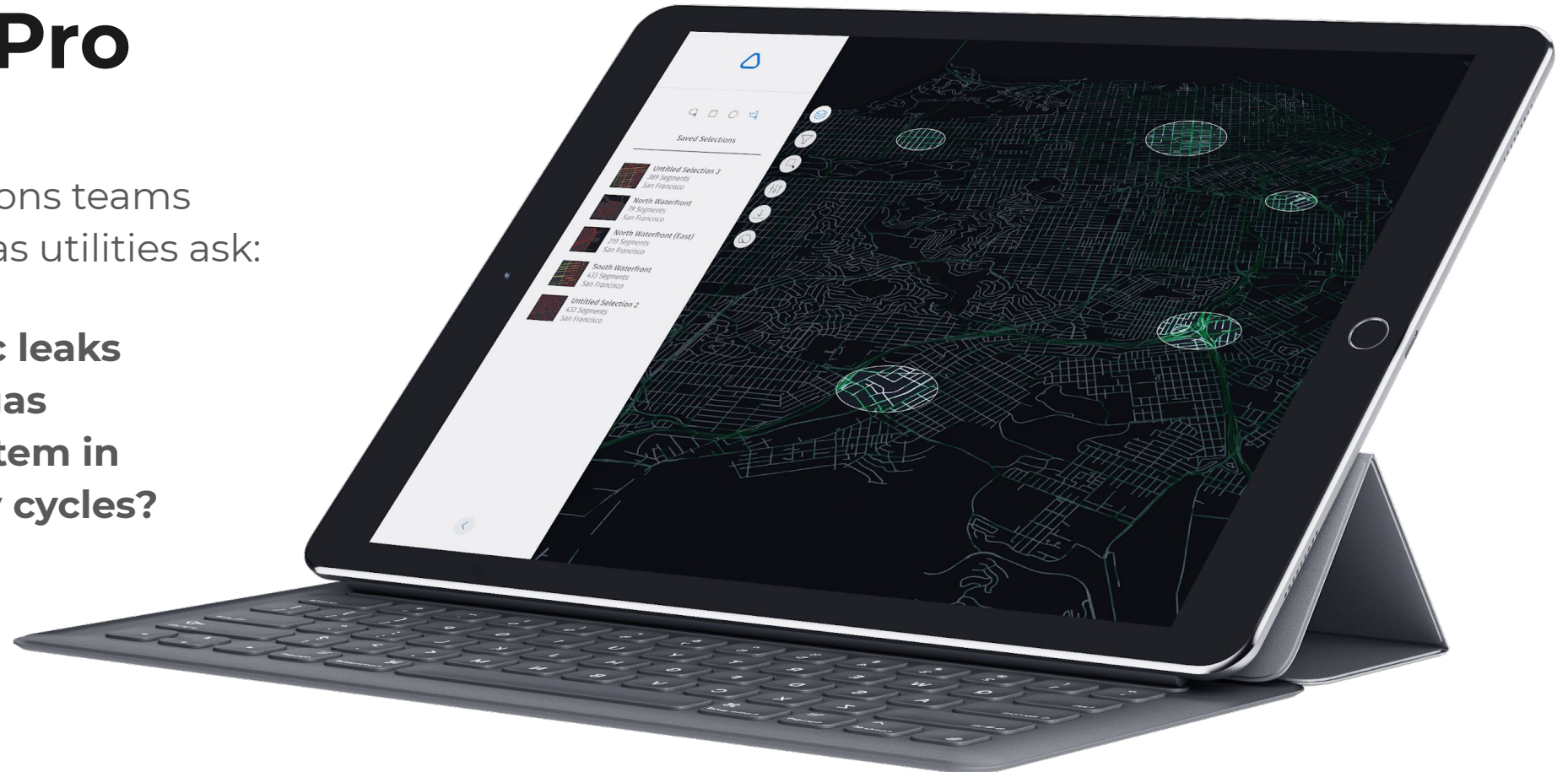
What is happening in the vast majority of my jurisdiction that I cannot see?



Aclima Pro

Network operations teams
within natural gas utilities ask:

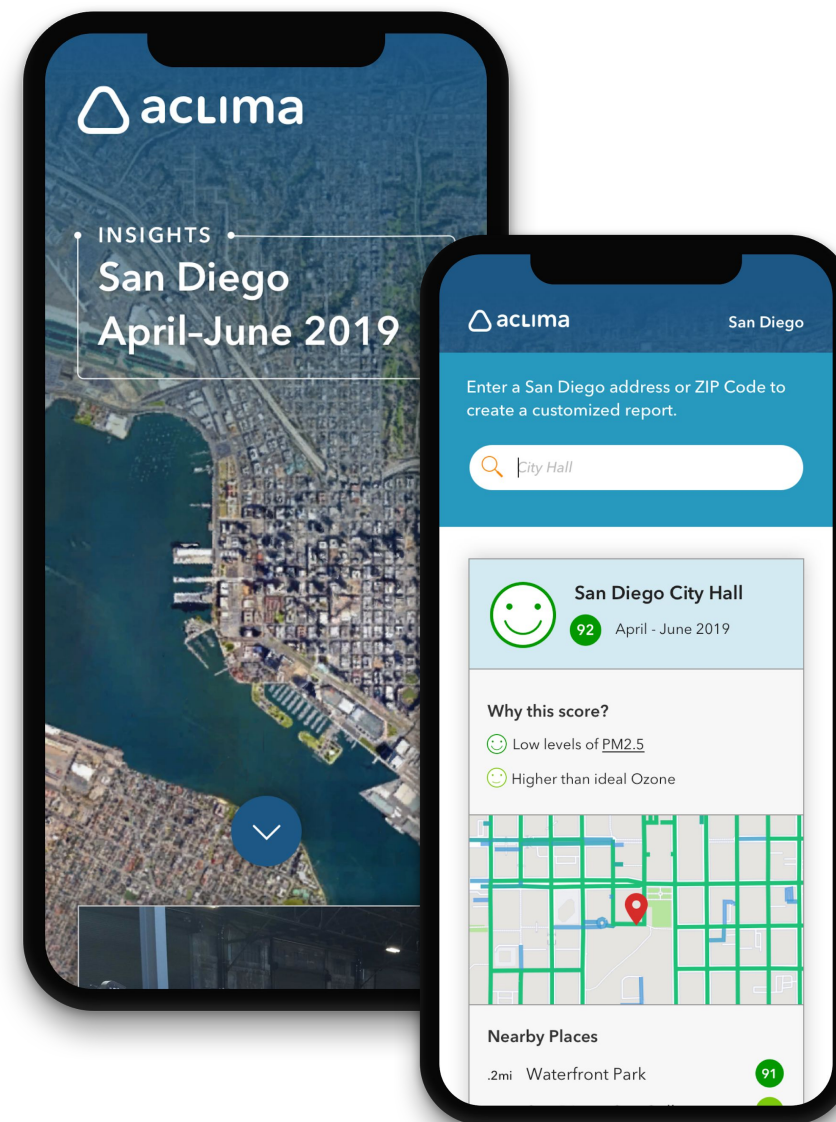
**Are problematic leaks
forming in my gas
distribution system in
between survey cycles?**



Aclima for Communities

Free app for allows the general public to ask:

How does air quality impact me and my family at our home, and in our community? How can I act to make a difference?



Data Products

Annual & Quarterly Baseline

Collection results in stable median baseline values for each road segment during the collection period

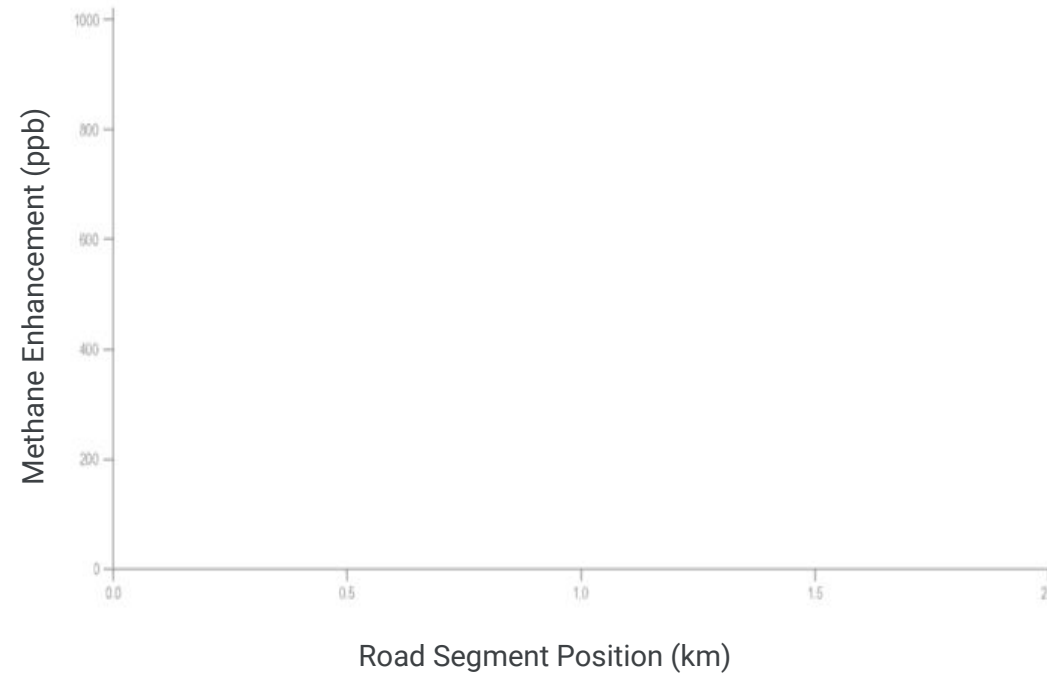
Multi-pass Approach

Achieved through repeated driving of target geography

Rigorous Statistical Sampling Design

Passes distributed across time of day, day of week, and collection period

Aclima methodology reveals persistent elevated pollution levels at hyper-local resolution.



Google Cloud: Stable, scalable & resilient

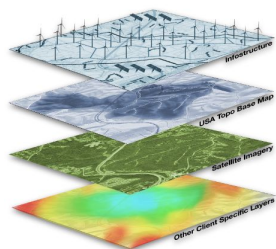
DATA COLLECTION



Aclima Mobile Mapping Data



Aclima & 3rd party stationary sensor data



Customer & 3rd party data

BACKEND PLATFORM

Ingestion & Pre-Processing



Go HTTP



Python Worker



Pub/Sub



Dataflow



Cloud Storage



BigQuery

Data Lake



Cloud Functions



Go



Python



DataProc/
Hadoop

Product Store, Servers & Interfaces



Elasticsearch



NodeJS/React



TileServer

API



Python 3

Device Data



MySQL/ Postgres
Cloud SQL

Data Analysis Pipeline



Kubernetes



Container
Registry

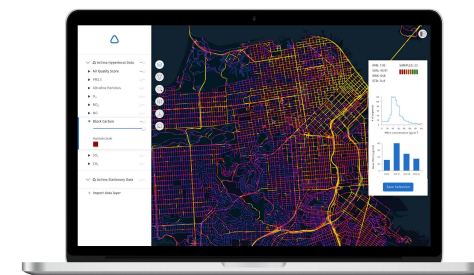


CircleCI



Docker

PRODUCTS



Aclima Pro



Aclima Citizen App

Diagnose



IDENTIFY
HOTSPOTS &
SOURCES



UNDERSTAND
GREENHOUSE
GASES



REVEAL
EXPOSURE



INTEGRATE
LAND USE

Act



TARGET
INTERVENTIONS



ENGAGE
COMMUNITIES



TRACK
PROGRESS



OPTIMIZE
INVESTMENT



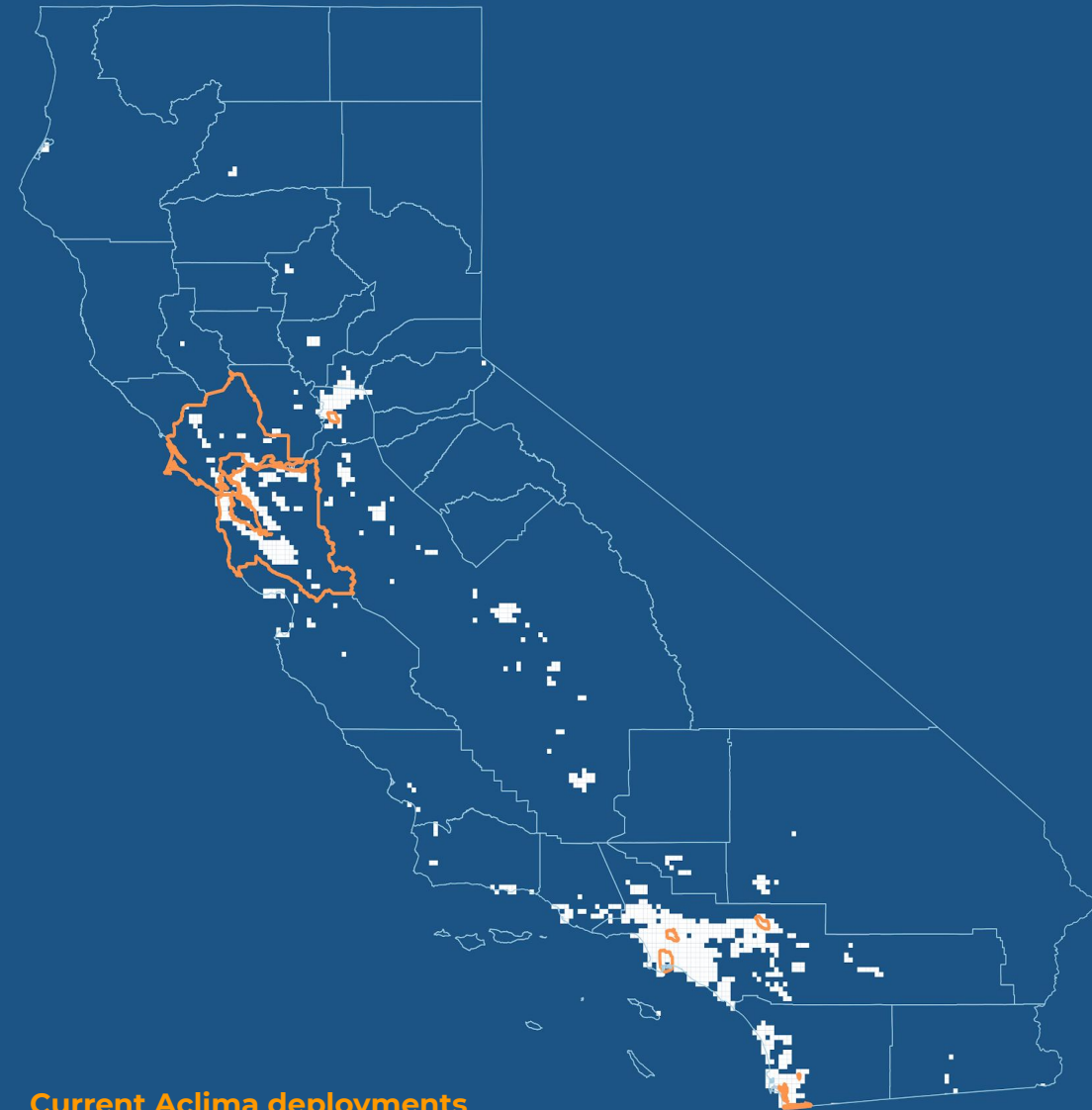
Google

Rapidly Expanding Coverage

Aclima is contracted with growing number of California public sector users:

- Bay Area (BAAQMD)
- County of San Mateo
- LA Metro (SCAQMD)
- San Diego County (SDAPCD)
- California Air Resources Board (CARB)

In the Bay Area, the Aclima fleet is currently deploying region-wide — will map indefinitely



Current Aclima deployments
High population density areas



Cupertino Presentation

▼ Aclima Hyperlocal

PM_{2.5}

O₃



NO₂

NO

CO

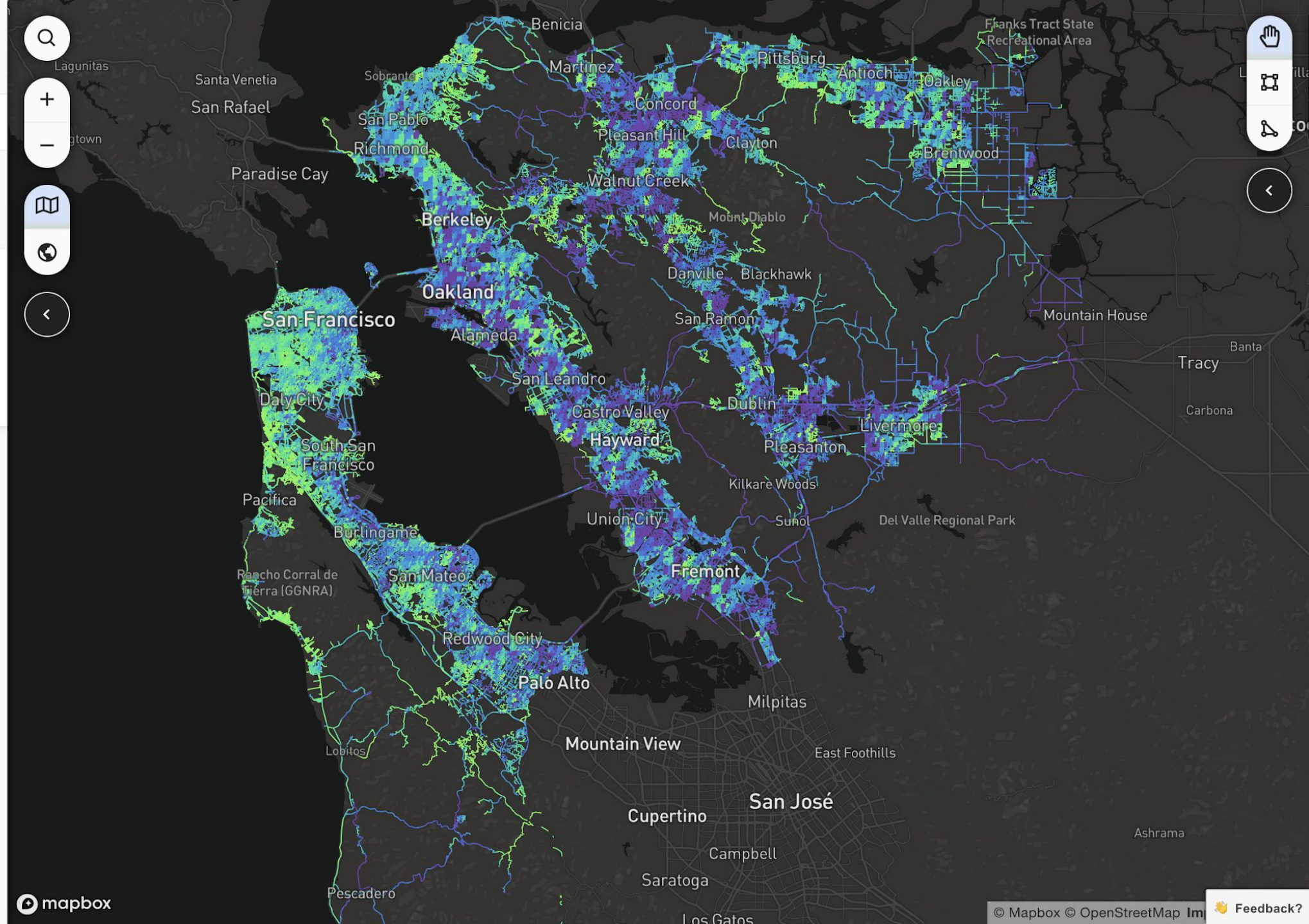
CO₂

Black Carbon

Methane

► AQMIS Stationary

+ Import Data



Serving Cupertino

Aclima is excited to deliver innovative services to enhance community wellbeing and support Cupertino's Climate Action Plan goals through:

- Access to hyper-local air quality data and Aclima Pro analytical tool
- Working with city staff to fully understand needs and align product roadmap to meet them
- Release of Cupertino-specific public experience
- Support from Aclima scientific staff to assess implications of hyper-local data and insights





Thank you.



Replica

**Change the way you
see your city move.**

At the center of many important transportation and land use decisions is a travel-demand model.

The development of a travel-demand model can take upwards of ten years, rendering it nearly useless by the time it has been certified.

THE TRAVEL-DEMAND MODEL

01**SEND A PAPER SURVEY**

Send a paper survey to approximately 0.5% of the region's population

**02****HIRE CONSULTANTS**

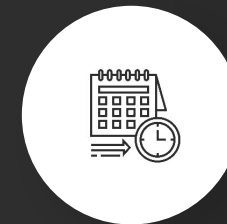
Hire consultants, who will spend the next 3-5 years building the model

**03****COLLECT THE DATA**

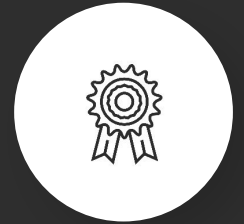
Spend 1-2 years collecting traffic and transit counts to calibrate the model

**04****CALIBRATE THE MODEL**

Spend 1-2 years calibrating the model

**05****VALIDATE THE MODEL**

Certify or validate the model **7-10 years after** the primary data was collected



This creates a significant opportunity to improve the places we live by equipping public and private organizations with better data and tools.



No single source of data is a silver bullet for understanding movement.

01

MOBILE
LOCATION
DATA

02

CONSUMER
/ RESIDENT
DATA

03

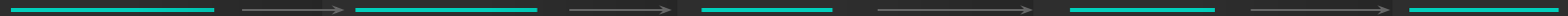
LAND USE /
REAL ESTATE
DATA

04

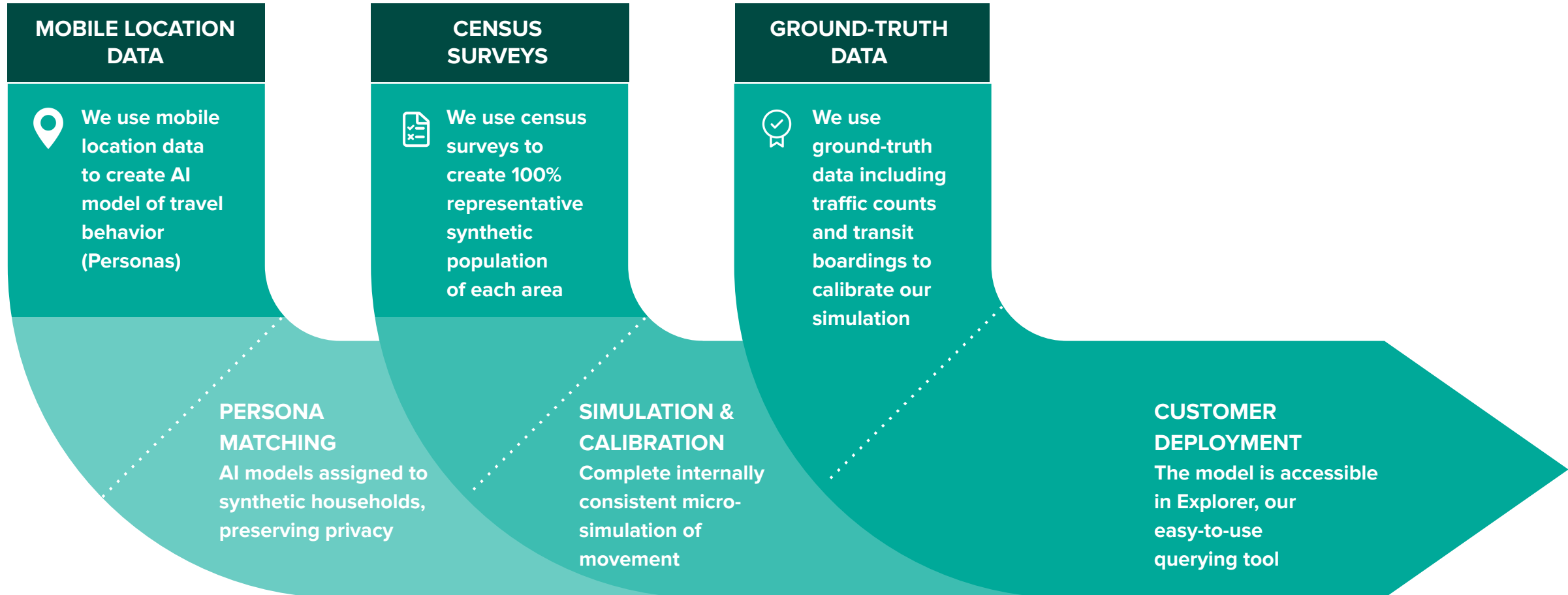
CREDIT
TRANSACTION

05

GROUND-TRUTH
DATA



Complete, accurate and representative turn-key solution.



Meet Replica.

A shared, fully-calibrated platform for monitoring the movement of people and goods in a privacy-sensitive way.

FRESH We deliver 4 models a year with data for every day of a typical week.

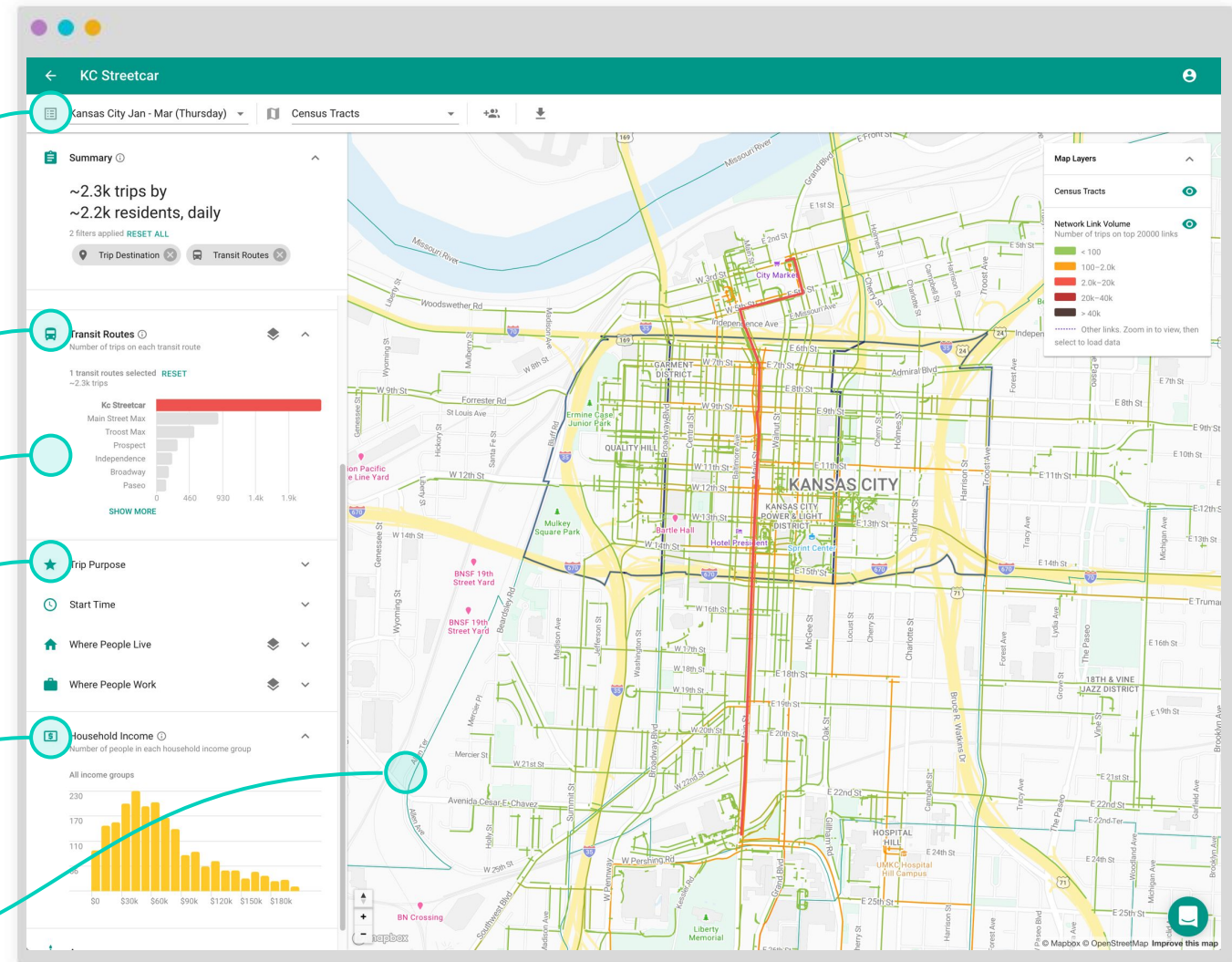
HOW Understand which modes, transit lines, roads and streets people are using.

WHY Explore why people are traveling.

WHEN Activities are modeled second-by-second with parcel level precision.

WHO Demographic info including income and age while protecting privacy

VISUAL High-fidelity visualizations of data on the map with choropleths, line maps and more.





Replica **QUESTIONS**

QUESTIONS?

