An Extensible Computing Architecture Design for Connected Autonomous Vehicle System

Jacob Hochstetler
PhD Dissertation Defense
Dr. Song Fu
Dr. Rodney Nielsen
Dr. Armin Mikler
Dr. Ryan Garlick

Organization

1. Introduction
2. Background
3. Problems and challenges
4. Extensible Edge Computing Architecture
    - Data format/protocol
    - Application orchestration
5. Other PhD research
    - Cloud/Edge-related
    - CAV-related
6. Conclusion & future work
7. Publications

Introduction: Computing Hierarchy

Cloud

- Infinitely[a] scalable resources
- Largest data latency
- Multiple-hops to producer

Near Edge

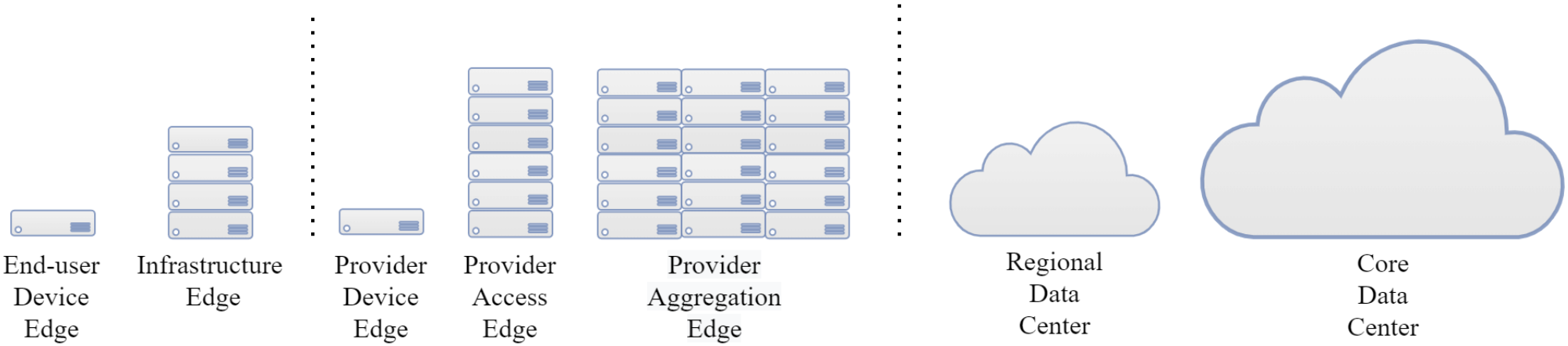
- Medium resources
- Medium data latency
- Single-hop to data producer

Far edge (Local)

- Limited resources
- Lowest data latency
- Directly connected producer
- Power throttling concerns

Connected computing hierarchy

Introduction: Edge



At every network layer, there exists Edge devices

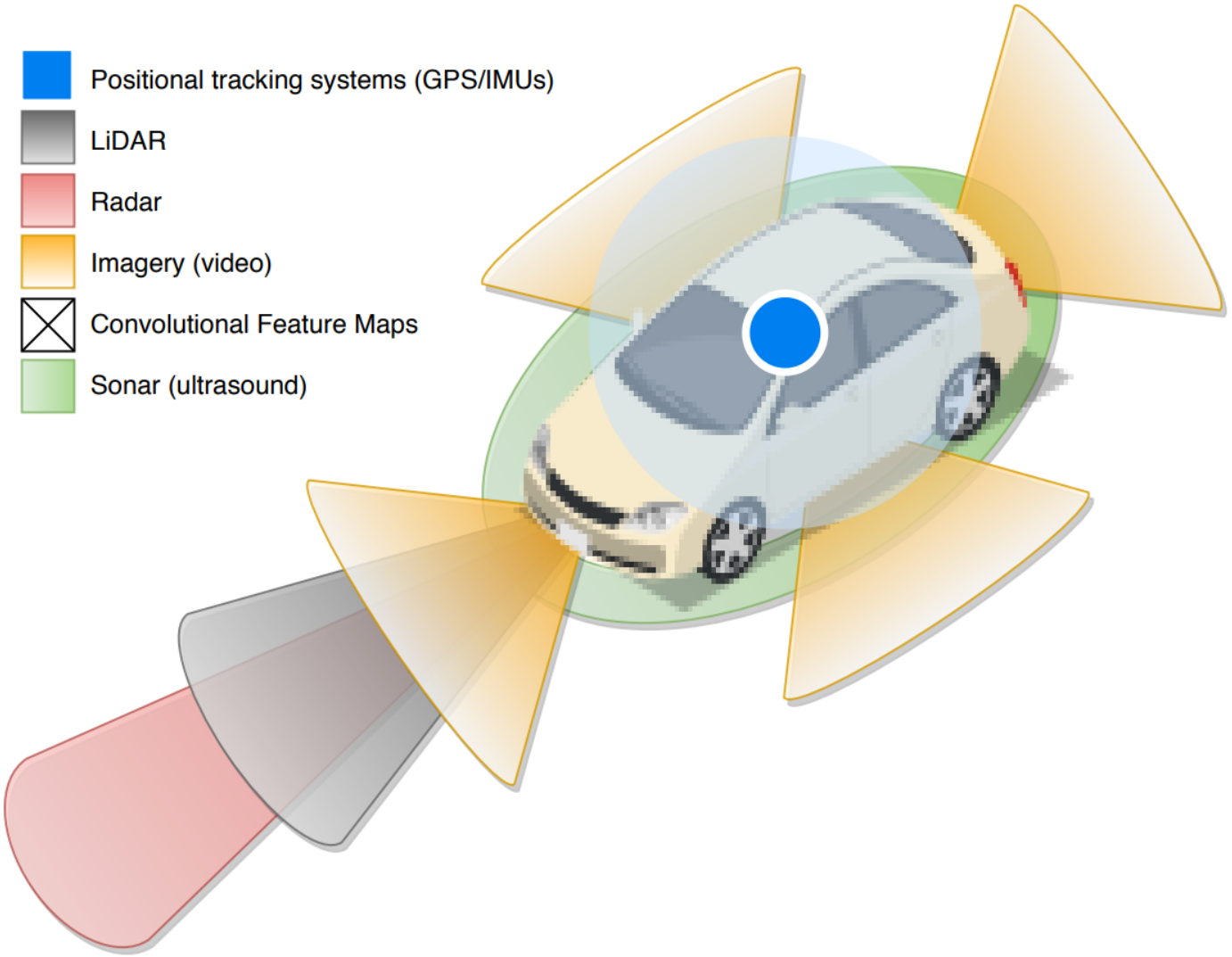
Background: Decentralized computing

- Cloud computing
  - Treating 'compute' like a utility
  - IaaS: **I**nfrastructure **a**s **a** **S**ervice (OS, application & data)
  - PaaS: **P**latform **a**s **a** **S**ervice (application & data)
  - SaaS: **S**oftware **a**s **a** **S**ervice (only data)
- Ubiquitous computing
  - Rise of embedded devices and IoT computing
  - Reducing the 'computer' interface, e.g. Apple Watch
- Connected vehicles
  - Connected and Autonomous Vehicles (CAV)
  - CAV ⇄ Edge / Edge ⇄ Cloud / CAV ⇄ Cloud

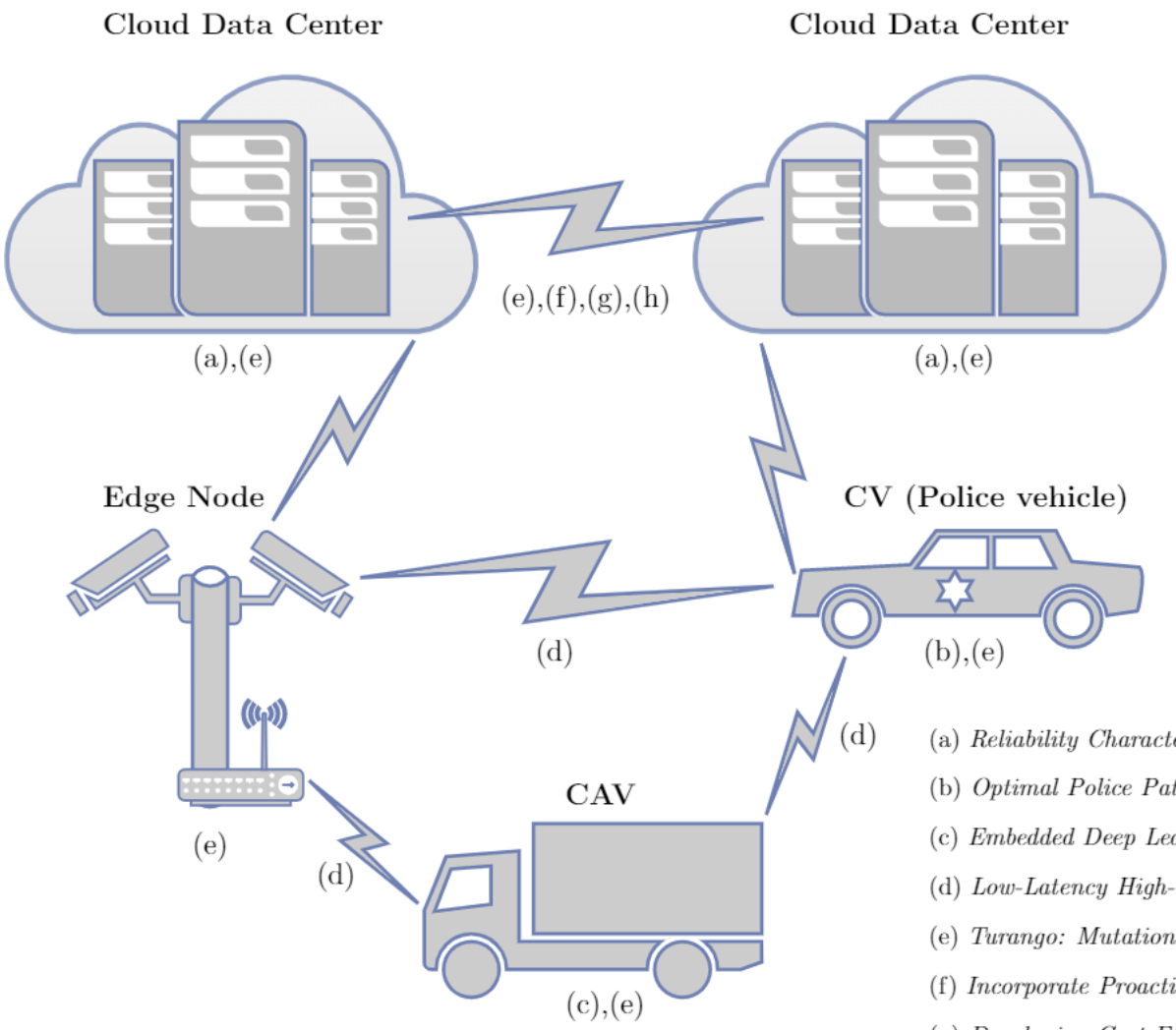
Background: In-Vehicle services and systems

- Real-time diagnostics
  - Started with standardized engine emission collection in 1994 (OBD-II)
  - Independent RT diagnostics from separate manufacturers, mostly Satcom-based
  - Modern systems use 3G/4G/LTE to haul data to cloud providers
- Advanced Driver-Assistant Systems (ADAS)
  - Camera/radar/LiDAR/Sonar (ultrasonic)-based
  - From basic self-parking to highway lane "keeping" and 'smart cruise control'
- Third-party applications
  - *Now*: 'Infotainment' with integrated 1st and 2nd-party applications
  - *Future*: 3rd-party applications (and government-party)

Background: Vehicle Data Producers/Sensors



Our proposed CAV/Edge/Cloud architecture



(a) *Reliability Characterization of Solid State Drives in a Scalable Production Data Center*

(b) *Optimal Police Patrol Planning Strategy for Smart City Safety*

(c) *Embedded Deep Learning for Vehicular Edge Computing*

(d) *Low-Latency High-Level Data Sharing for Connected and Autonomous Vehicular Networks*

(e) *Turango: Mutation Testing a Language*

(f) *Incorporate Proactive Data Protection in ZFS Towards Reliable Storage Systems*

(g) *Developing Cost-Effective Data Rescue Schemes to Tackle Disk Failures in Data Centers*

(h) *An Empirical Study of Quad-Level Cell (QLC) NAND Flash SSDs for Big Data Application*

Problems and challenges

1. Extensibility and compatibility
    - Breaking changes in models/services
    - Ability to maintain service older clients
    - Interoperability between different manufacturers

---

2. Data sharing
    - End-user privacy
    - Origination/maintenance of trust
    - Legal and moral facets of data retention

Focus on extensibility and compatibility

- Data format
- Data protocol
- 1st through 3rd-party integration

"All problems in computer science can be solved by another level of indirection [abstraction]"

.

.

.

"but that usually will create another problem"

A favorite, misquoted quote from Butler Lampson

Challenge: Data format

- Compatibility
  - Backwards (new fields won't incur collisions)
  - Forwards (future clients accept 'older' versions)
  - No additional metadata (avoiding 'versioning' fields)
- Technical merits
  - Encoding size (literal size on the 'wire')
  - Speed (serialization / deserialization)
  - Efficient (lightweight CPU/memory requirements)
- Polyglot requirements
  - Wide variety of language support
  - Integration with various development platforms/pipelines

Challenge: Data protocol

- Operates at all levels of CAV ecosystem
  - **Edge/CAV**: Low-resource/low-bandwidth/low-latency environment
  - **Cloud**: infinite resources/high-latency (depending on data gravity)
- Machine-to-Machine focused
  - Human interfaces should not be a priority
  - Can transverse heterogenous networks
- Similar Polyglot requirements
  - Wide variety of language support
  - Integration with various development platforms/pipelines

Challenge: 1st through 3rd party integration

Massive difference between platforms

`CAV ⇄ Edge ⇄ Cloud`:

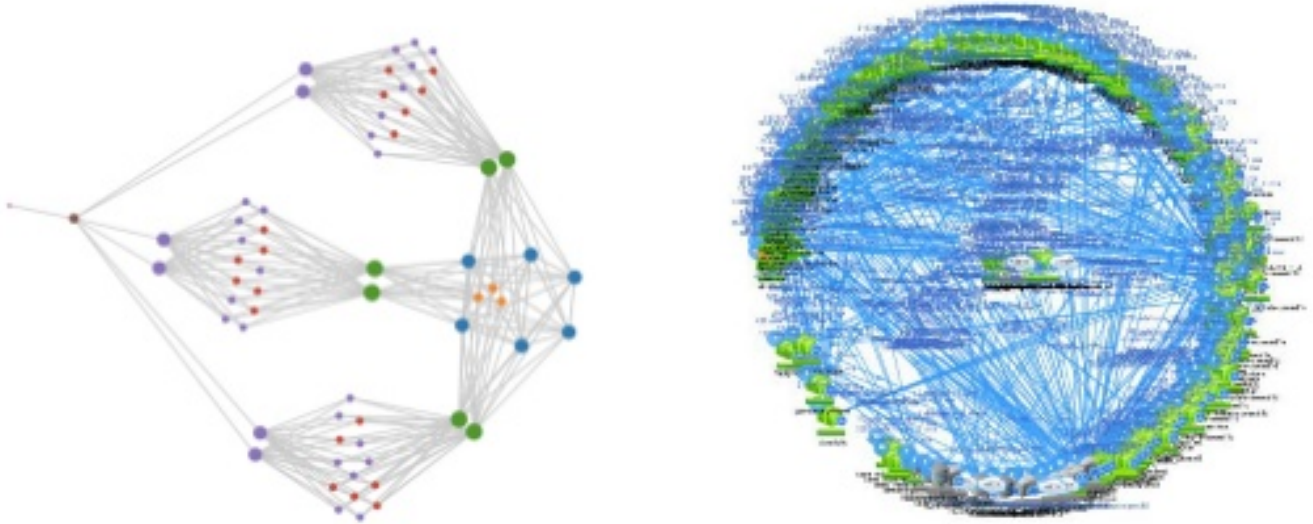
- Scale of Resources
- Variety of Services
- Varying Intervals of Delivery

Integrators nightmare, e.g. the same camera could be:

- In a different sensor package
- Sold from a different vendor/Valued Added Reseller (VAR) with "added" services
- Installed in a proprietary way by a different OEM

Extensible Edge Computing Architecture

Driven by the inevitability of microservices



Netflix's microservice Death Star

Contains three parts:

- Data format: `protobuf` (Protocol buffers)
- Data protocol: gRPC
- Application orchestration: *k3s* (lightweight kubernetes), *k3OS* (kubernetes OS)

Protocol buffers

Two kinds of problems solved:

- Technical
- "Cultural"

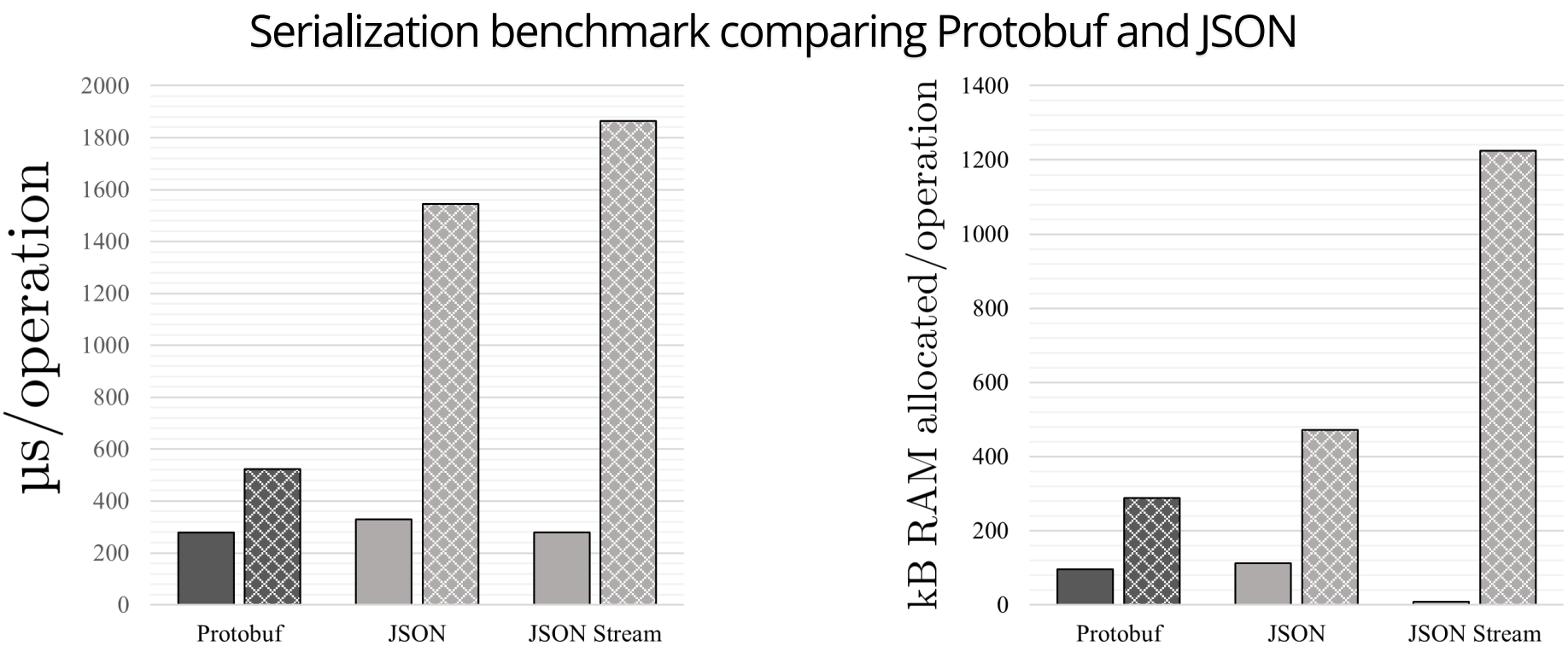
Comparisons:

- ASN.1
- XML
- CORBA
- JSON
- Thrift
- Avro

Protocol buffers: Background

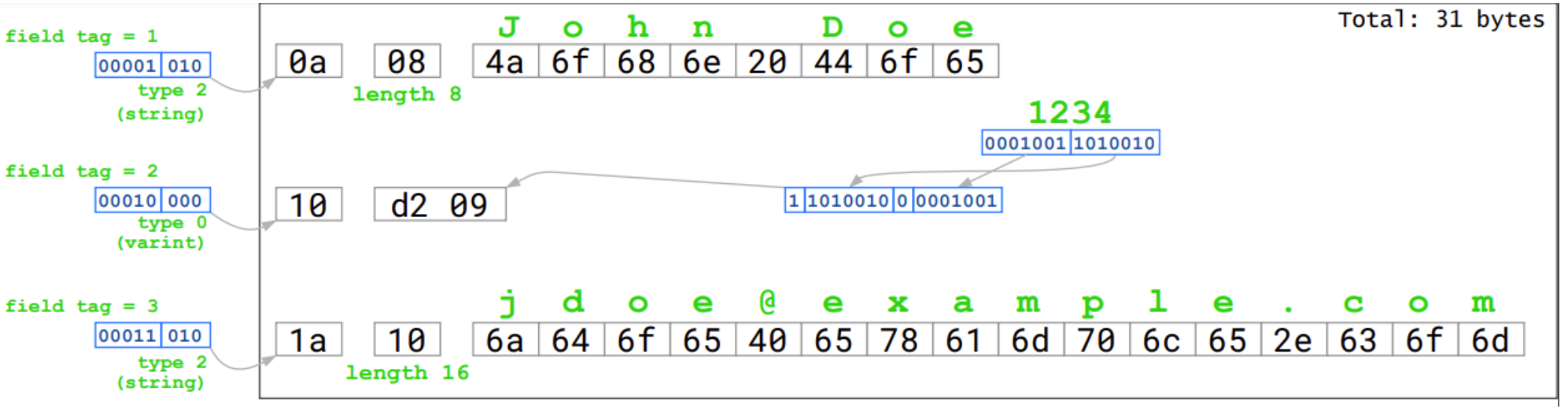
- Designed ~2001, evolved out of need for "web-scale" communication in data centers
- Compact, binary serialization
- Interface Definition Language to define models, compiles to interfaces for targets

---



Protobuf in dark gray, with JSON in light gray. Serialization is solid, while deserialization is hatched.

Protocol buffers: Example serialized



Example message composed of three fields:

- hexadecimal wire values shown in *black*,
- with decoded values and field descriptions in *green*

Protocol buffers: Example model

```
1  message Person {
2      required string name = 1;
3      required int32 id = 2;
4      optional string email = 3;
5
6      enum TelType {
7        MOBILE = 0;
8        HOME = 1;
9        WORK = 2;
10      }
11
12      message TelNumber {
13        required string number = 1;
14        optional TelType type = 2 [default = HOME];
15      }
16
17      repeated TelNumber phone = 4;
18  }
```

'Person' model containing fields for Name, ID, Email and multiple phone numbers.

gRPC

Problems solved:

- New application paradigm ➔ Monolith vs Microservices
- Layered communications crossing network stacks, data centers and regions ("fleet")
- Built-in JSON↔gRPC gateway enables legacy clients, or where `.proto` files are inaccessible to consumers

---

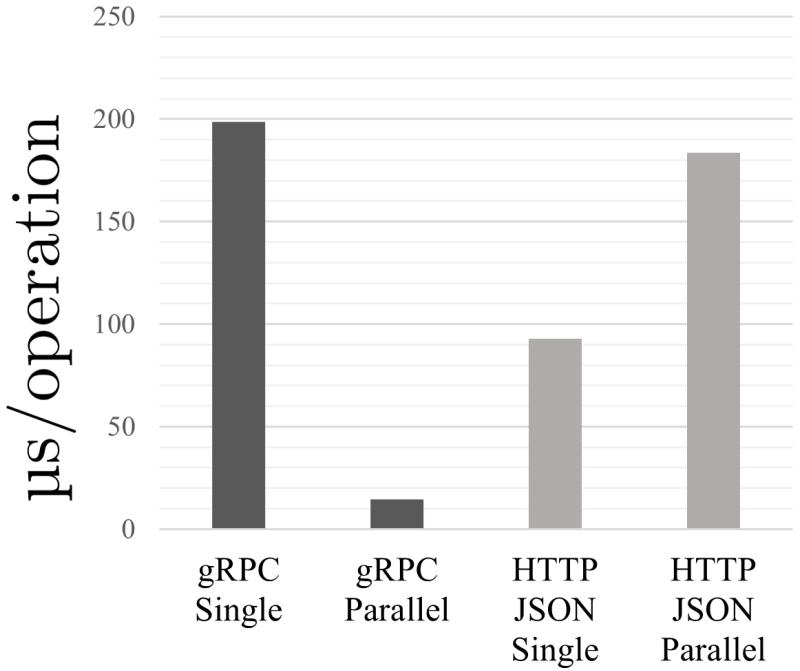
Comparisons:

- JSON over HTTP
- Thrift RPC
- CoAP

gRPC: Background

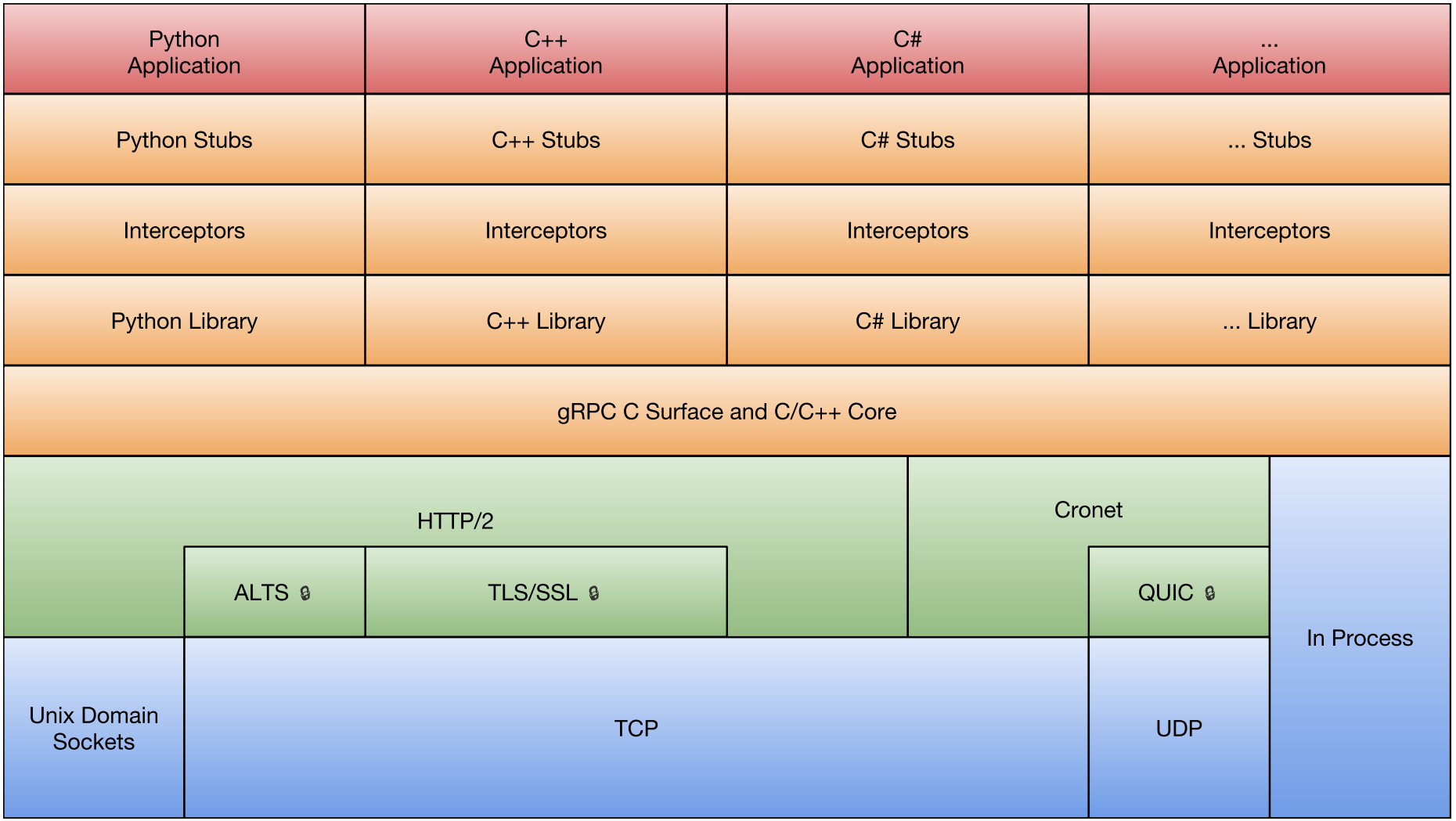
- Released ~2015, designed for massively distributed services spanning data centers
- While not tied directly to Protobufs, uses the same IDL
- "Rides" on HTTP/2, and already tested with HTTP/3

Heartbeat benchmarks comparing Protobuf and JSON



gRPC is in *dark gray* and JSON over HTTP is in *light gray.*

gRPC: Architecture stack

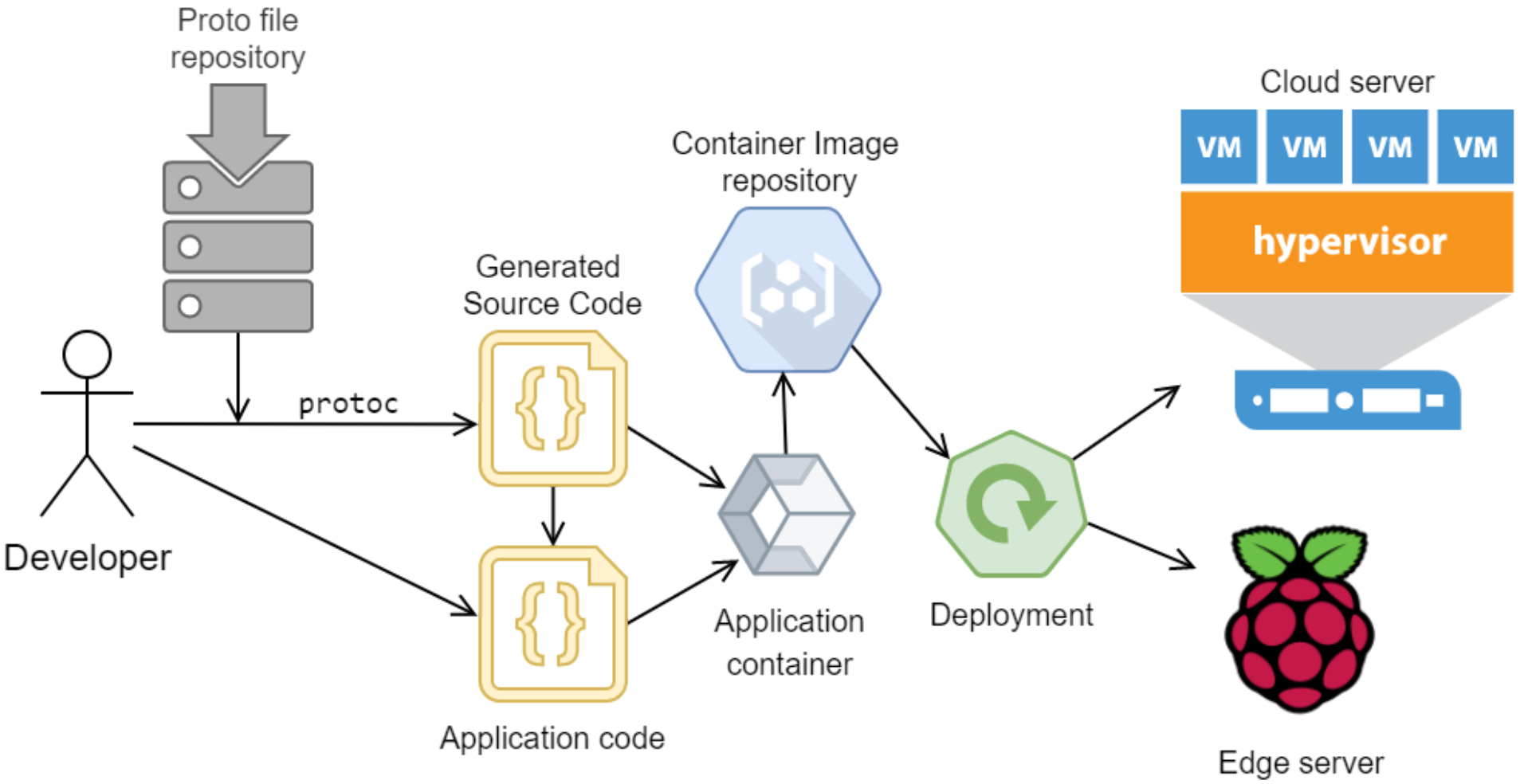


gRPC: Example service

```
1  service Contacts {
2    rpc CreateContact(CreateContactRequest) returns (Empty) { // Adds a Person to the Contacts
3      options (google.api.http) = {
4        post: "/v1/contacts", // Create maps to HTTP POST.
5        body: "person",
6      }
7    }
8    rpc GetContact(Person) returns (Person) { // Get returns person for ID
9      options (google.api.http) = { // Get maps to HTTP GET. No body.
10        get: "/v1/contacts/{id}" // 'id' field mapped from Person definition
11       }
12    }
13    rpc ListContacts(Name) returns (People) { // Searches Contacts for People matching 'name'
14      options (google.api.http) = { // Get maps to HTTP GET. No body.
15        get: "/v1/contacts"  // URL query params are automatically mapped, e.g. ?name=$name
16      }
17    }
18 }
19 message CreateContactRequest {
20     Person person = 1; // The field name should match the Noun in the method name.
21 }
22 message Name { string name = 1; }
23 message Empty {} // empty message to represent an empty response
```

'Contacts' service model with an endpoint for Creation, Getting, and Listing.

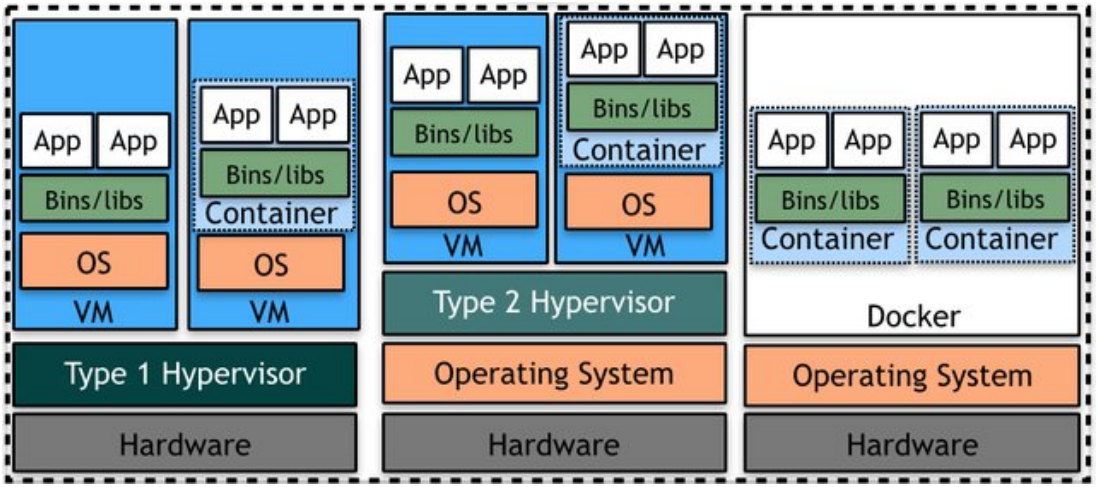
Protocol buffers/gRPC development lifecycle



Protocol buffer and gRPC developer workflow

Application orchestration

- Portable application deployment
- Network topology is "unstable" ... clusters should form to do "work" dynamically
- Multi-cloud capable (from edge clusters to public cloud providers)



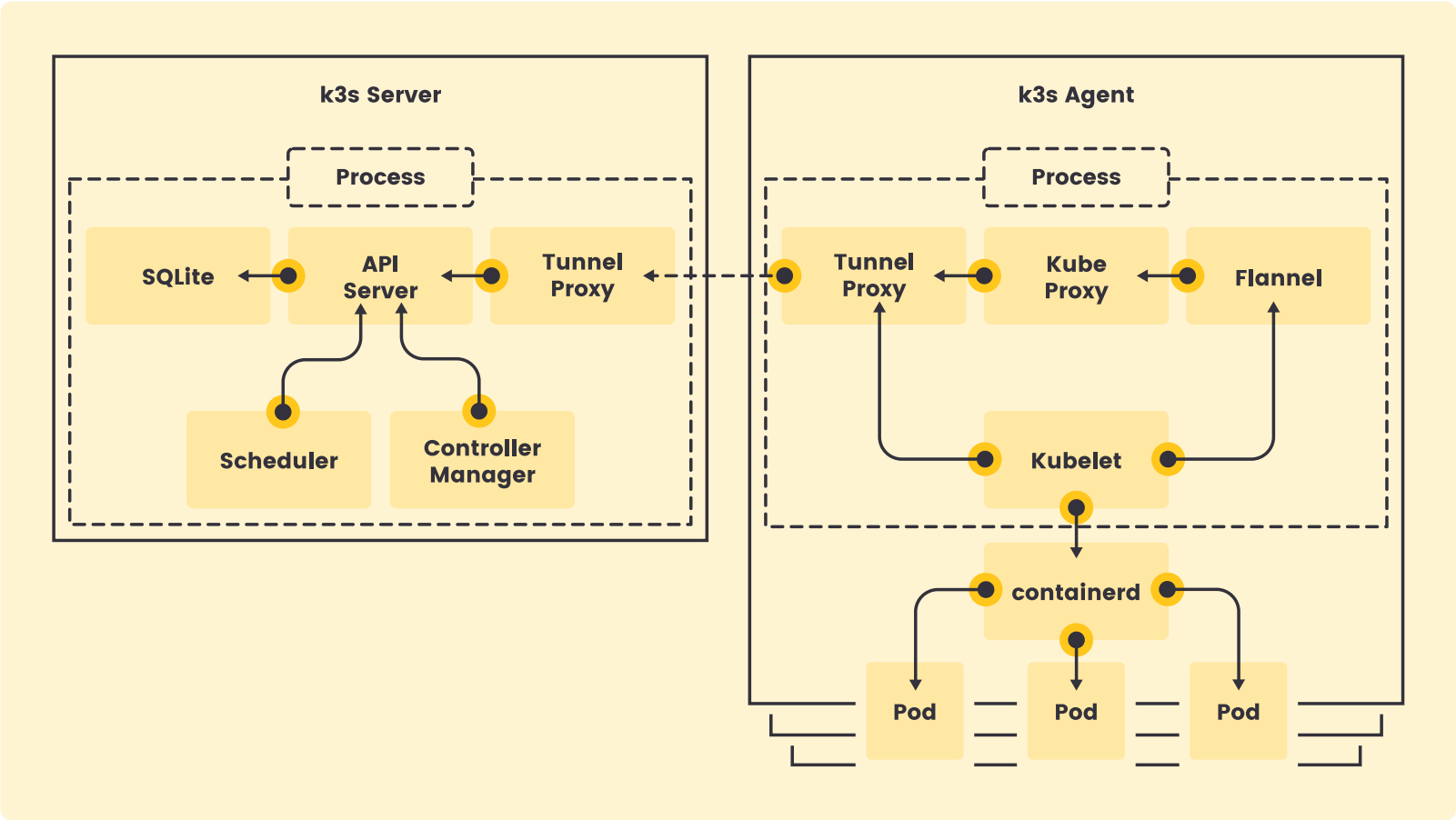
Type 1 and Type 2 virtualization compared to containerization

Virtualization and containerization is not enough

k3s

containerization = abstracting applications' runtime environment

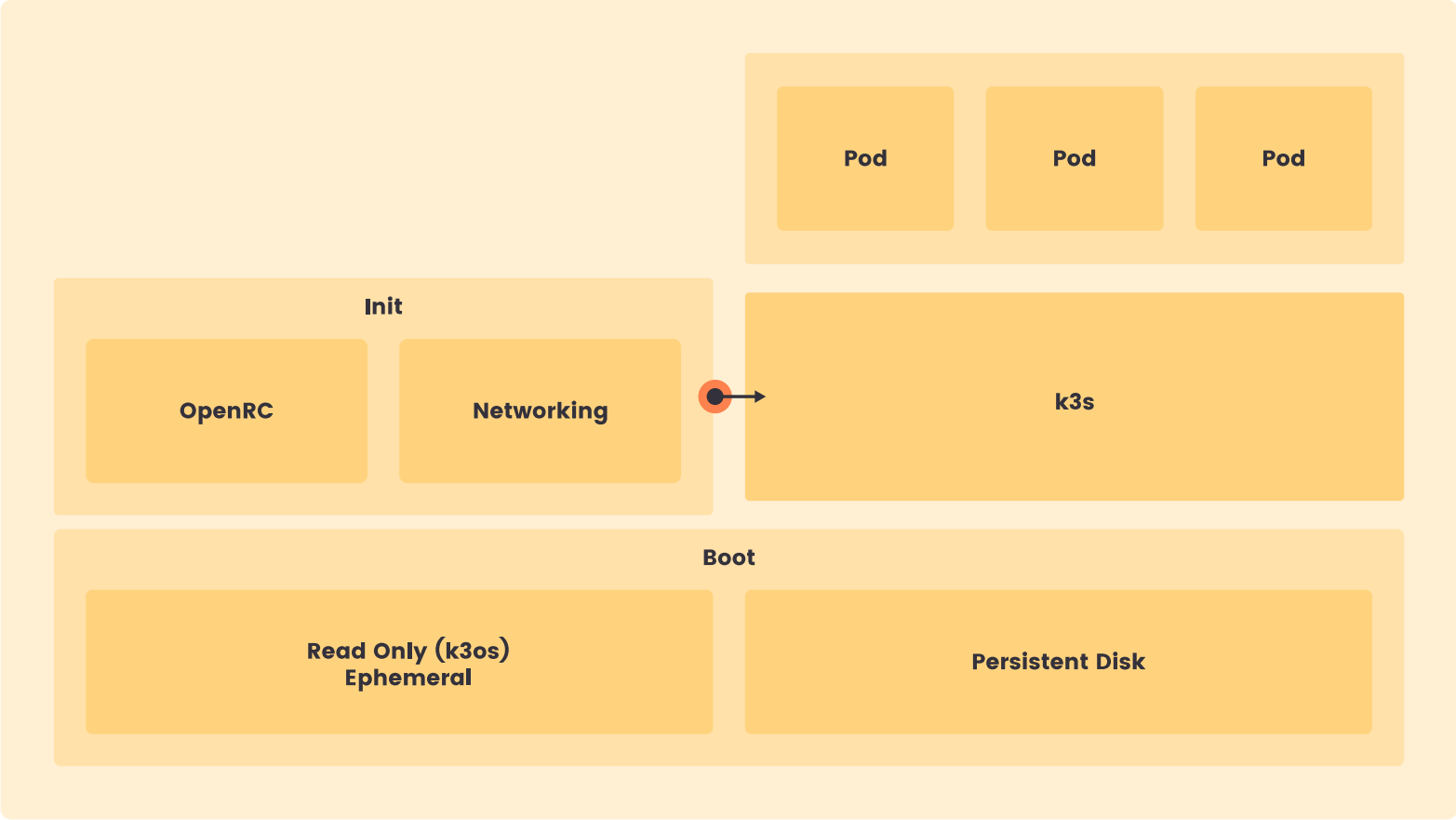
kubernetes (k8s) = abstracting applications' runtime scheduling



k3s overview of a Server and Agent

k3os

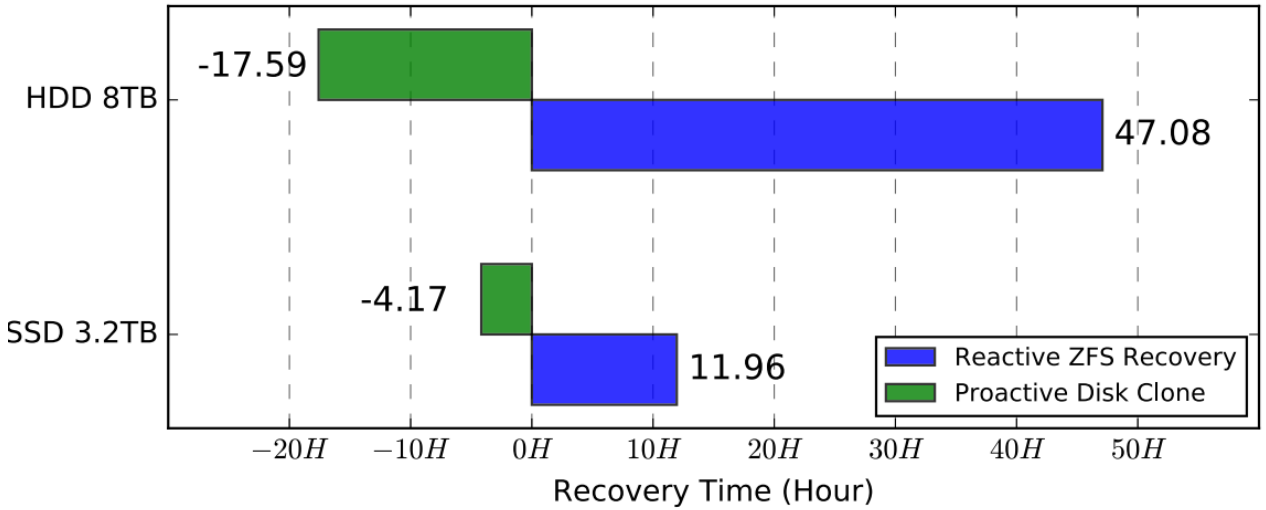
Just like the kernel schedules processes in an operating system;
k8s schedules applications in a data center



k3os overview of a VM

My other research: Cloud related

- Reliability Characterization of Solid State Drives in a Scalable Production Data Center
- Incorporate Proactive Data Protection in ZFS Towards Reliable Storage Systems
- Developing Cost-Effective Data Rescue Schemes to Tackle Disk Failures in Data Centers
- An Empirical Study of Quad-Level Cell (QLC) NAND Flash SSDs for Big Data Applications



Proactive disk cloning and post-failure disk recovery for a fully-utilized *zpool*. **0H** denotes the disk failure occurrence.

My other research: Proactive data protection

Disk access speed has been outpaced by the increasing capacity.

The stripe width of RAID arrays have grown to fill the gap between disk speed and capacity.

Unfortunately, the probability of having double and even triple failures also increases as the disk recovery time is significantly prolonged

---

In this work a Proactive Data Protection (**PDP**) framework is proposed to reduce the recovery time for RAID systems in ZFS.

PDP chooses among one of three strategies based upon real-time characteristics of the RAID pools to provide the quickest, most cost-effective recovery for data center storage.

My other research: Proactive data protection

Three strategies:

1. Proactive Disk Cloning (P-DISCO)

   Migrates data on a predicted, failing drive to a hot spare using disk cloning.

---

2. Proactive Active-data Recovery (P-ADAR)

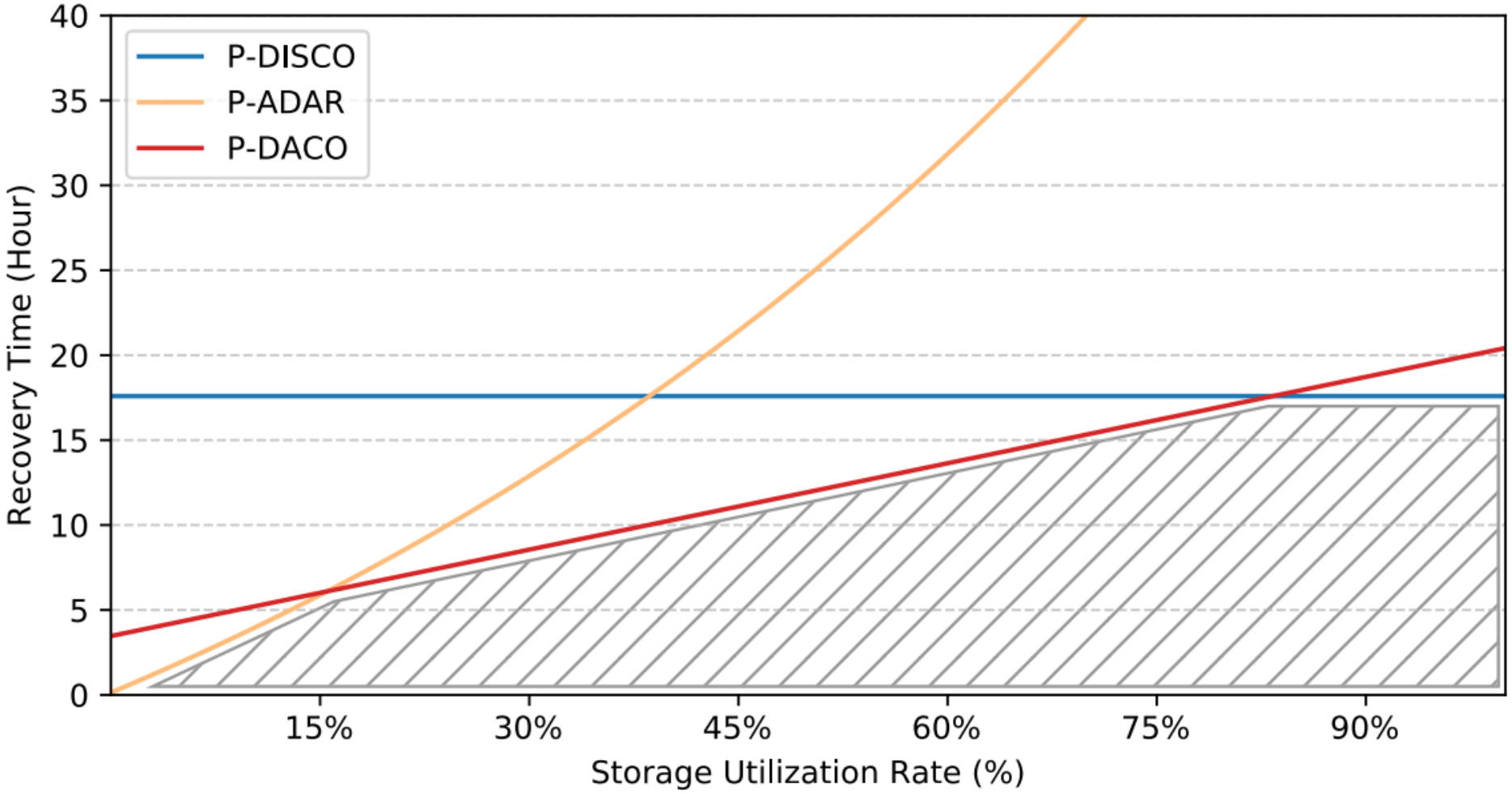
   Proactively rescues only active-data to the spares, although computationally intensive, recovering only the minimum amount of necessary data can save time.

---

3. Proactive Active-data Cloning (P-DACO)

   Only clones the active data during rescue, and eliminates the need to compute parity (combination of #1 and #2 strategies).

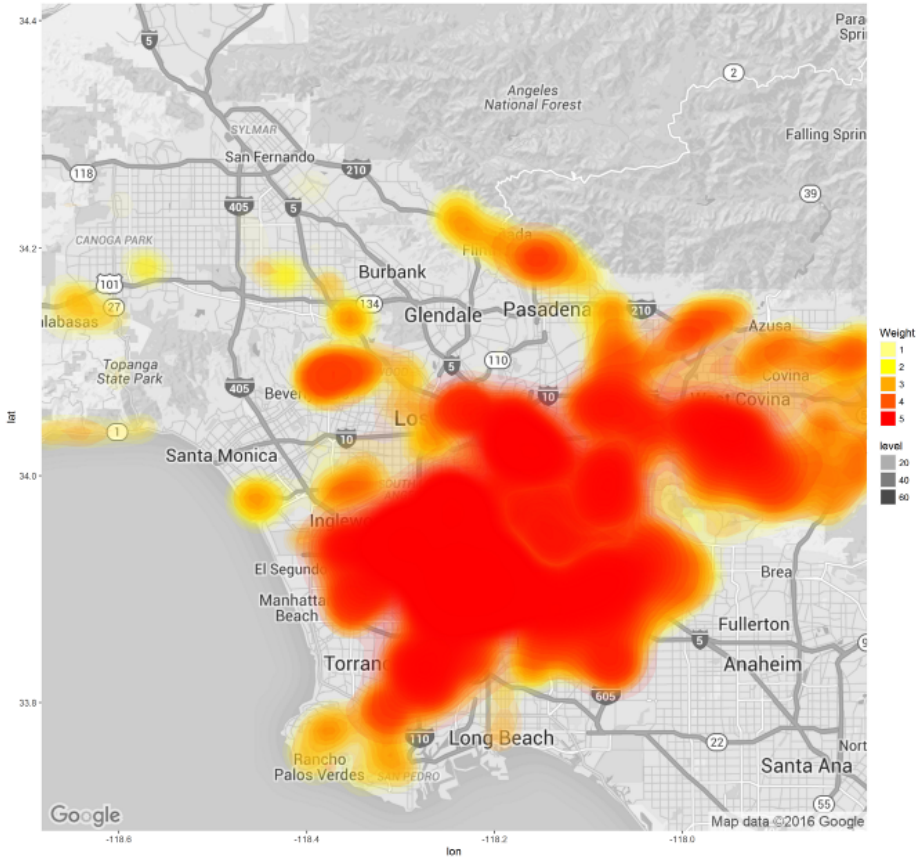
My other research: Proactive data protection



Time used by each proactive strategies for data rescue.

My other research: Edge related

- TuranGo: Mutation Testing a Language
- Optimal Police Patrol Planning Strategy for Smart City Safety



LA County crime *stat_density2d* plot with size=1, bins=128.

My other research: Police Patrol Planning

Police are not an infinite resource and must be managed effectively to reduce overall crime.

This work uses a novel approach to police patrol placement by using the following workflow:

1. Process crime data set: statistical analysis, remove "bad" data, assign crime weights
2. Cluster crime centroids by quantity and weight
3. Use historical traffic information for distance calculation between these centroids
4. Maximize entropy by placement of police patrols on the cluster centroids

My other research: Police Patrol Planning

LA County was chosen for three reasons:

1. Lower external variables (namely weather stability)

   LA County: low of 8.6°C to a high of 23.5°C with 380mm rain
   Cook (Chicago): low of -10.8°C to a high of 28.6°C with 940mm rain

---

2. Long drive distances to take advantage of the entropy calculation

   LA County: 12,300 km^2

   Cook (Chicago): 4,235 km^2

---

3. Open, consistent data

   The Los Angeles Open Data project provides crime data for LA County, and in total comprised 2,130,504 crimes from 2005 to 2015.

My other research: Police Patrol Planning

Number of clusters chosen from hexbin plots.

50 clusters plotted using K-means (weighted crimes replicated).

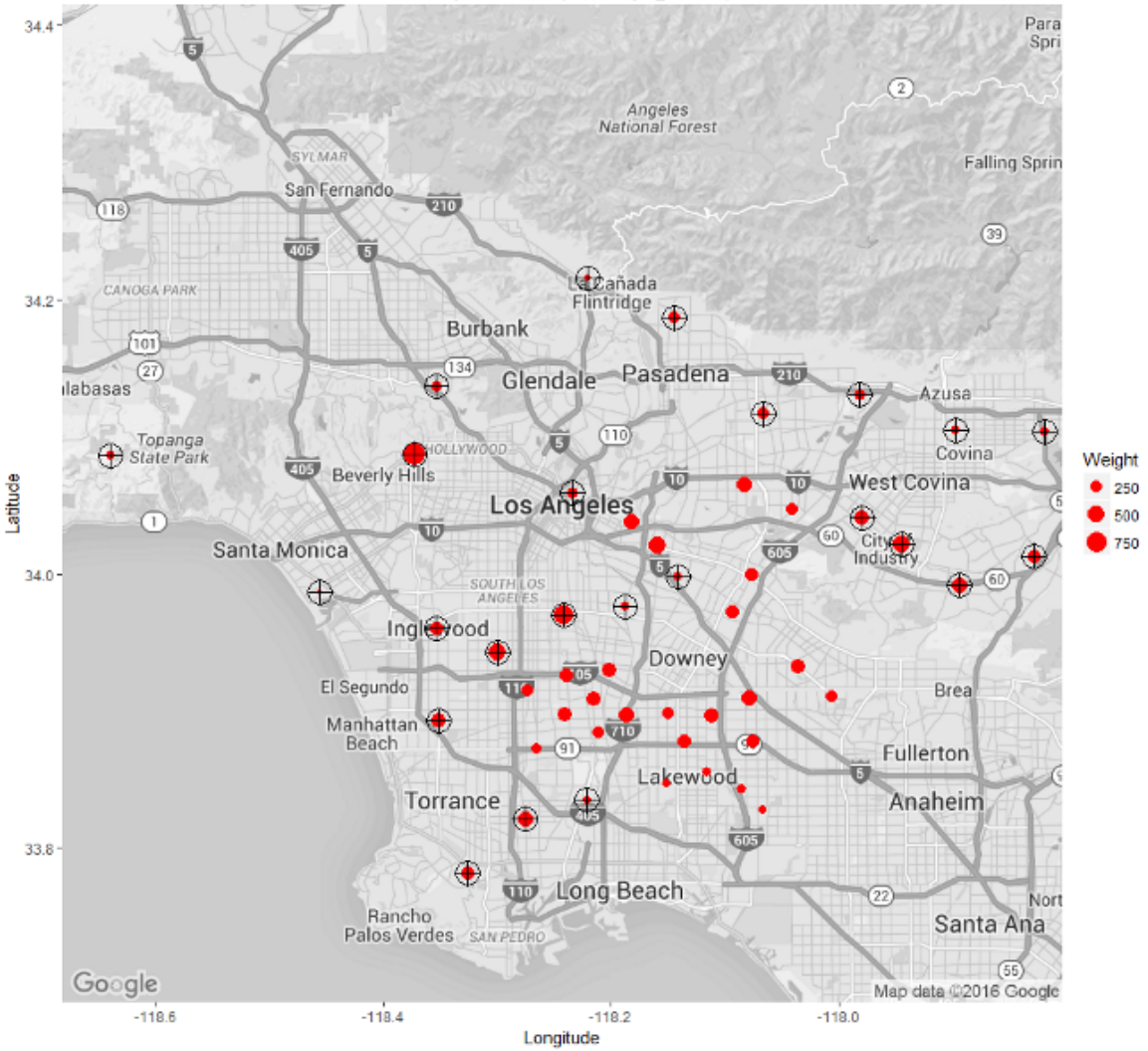
Drive distances between all the centroids retrieved using historical traffic data.

Patrols placed to maximize entropy using:

$$H_{c1} = -p(c_1)\ln p(c_1)\,, \quad p(c_1) = \left(\frac{r_{c1}}{r_{sys}} + \frac{w_{c1}}{w_{sys}}\right) \Big/ 2$$

The entropy is a combination of the weight of a crime centroid (w_{c1}), generated from crime clustering, over the total system weight (w_{sys}), added to the quickest path from the centroid to any other centroid (r_{c1}), over the quickest path in the entire system (r_{sys})

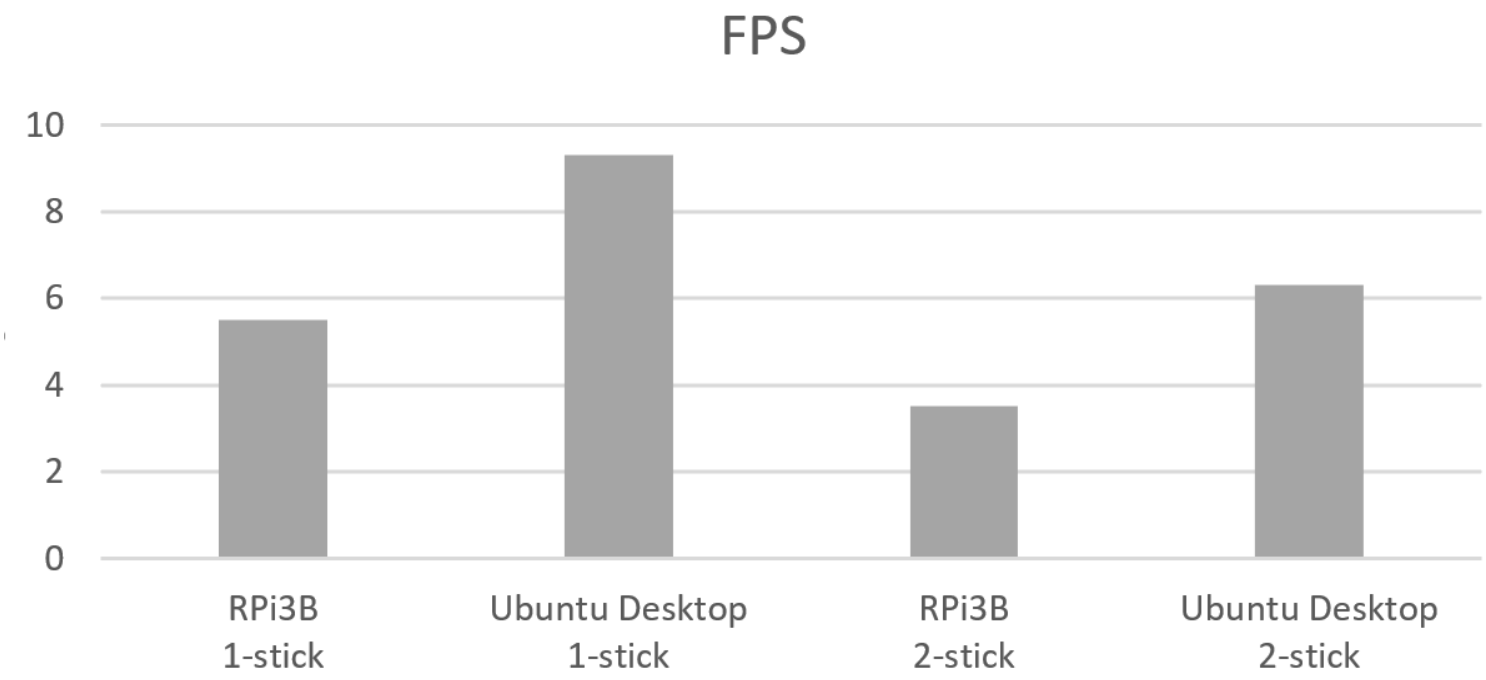
My other research: Police Patrol Planning



50 crime clusters with 25 patrols placed using entropy algorithm. Black cross-hairs show placed patrols.

My other research: CAV related

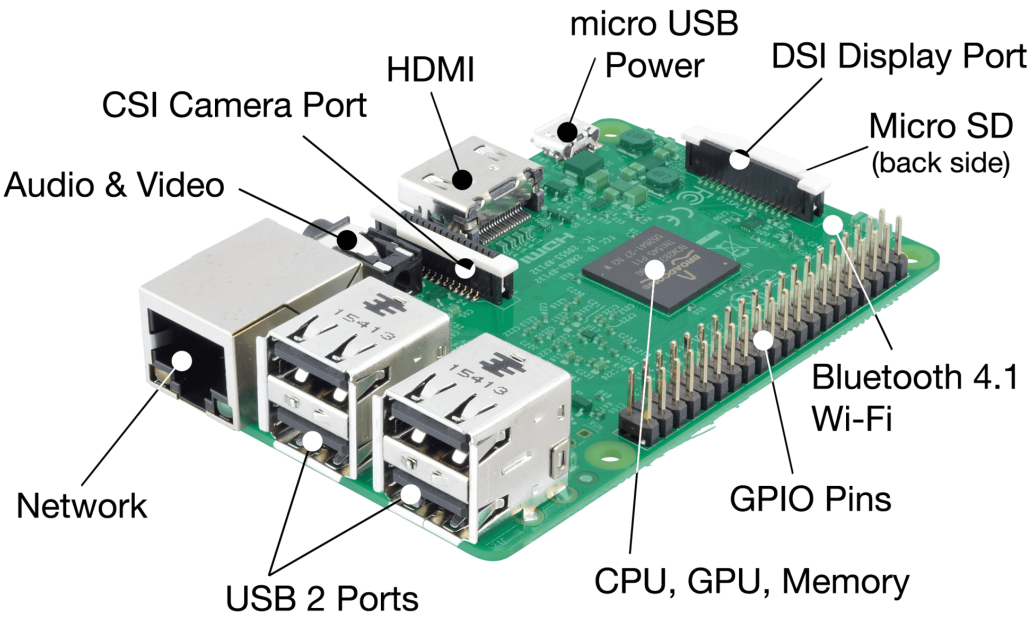
- Embedded Deep Learning for Vehicular Edge Computing
- Low-Latency High-Level Data Sharing for Connected and Autonomous Vehicular Networks



GoogLeNet CNN classifier: RPi3B + VCS versus Desktop + VCS

My other research: Embedded Deep Learning

The proliferation of resource-constrained edge devices has pushed development of supplementary AI-specific devices for inference on already trained models.

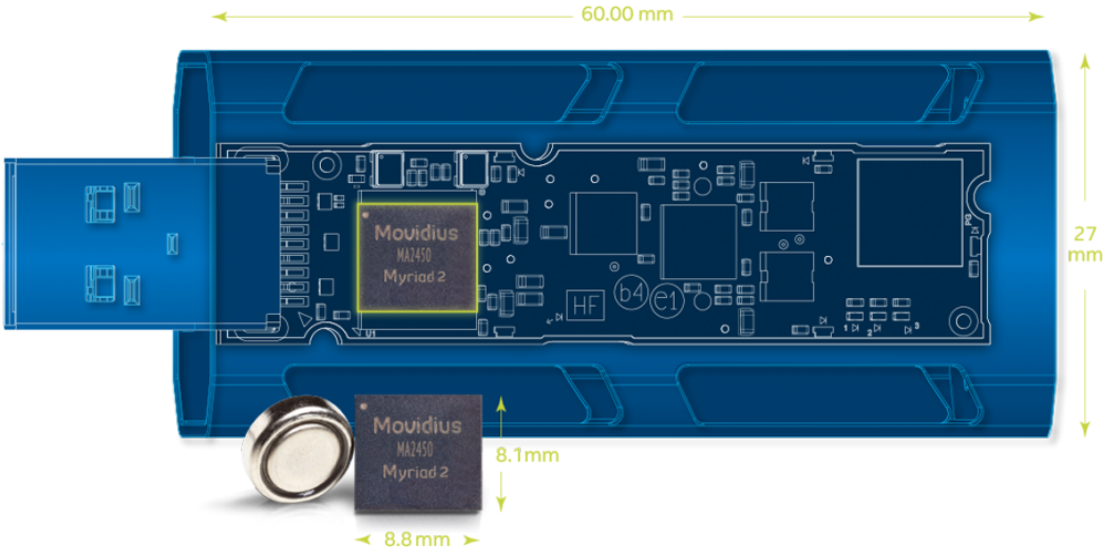


The Raspberry Pi 3 Model B used in my experiments contains a 1.2gHz 64-bit quad-core Cortex-A53 (ARMv8) CPU, 1GB of low-power DDR2 SDRAM and four USB 2.0 ports via on-board 5-port USB hub, drawing 6.7W at peak load.

My other research: Embedded Deep Learning

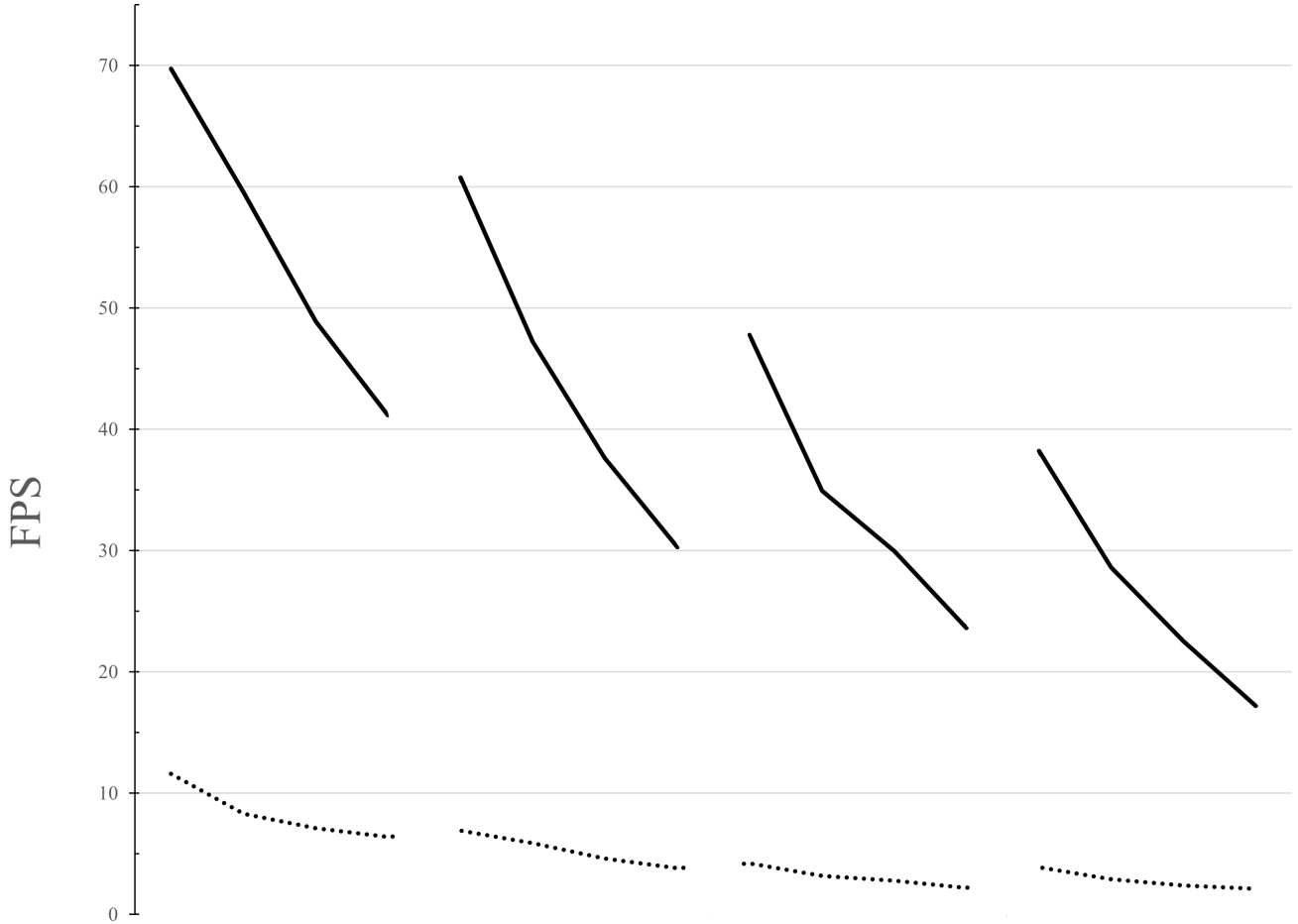
The AI stick is a USB device containing a specialized processing unit, in this case a Vision Processing Unit (VPU).

This VPU is optimized to execute models in TensorFlow and Caffe frameworks.



Intel™ Movidius® Neural Compute Stick (NCS) is a tiny fanless, USB 3.0 Type-A deep learning device containing a Myriad 2 Vision Processing Unit producing almost 100 GFLOPS while only using 1W of power.

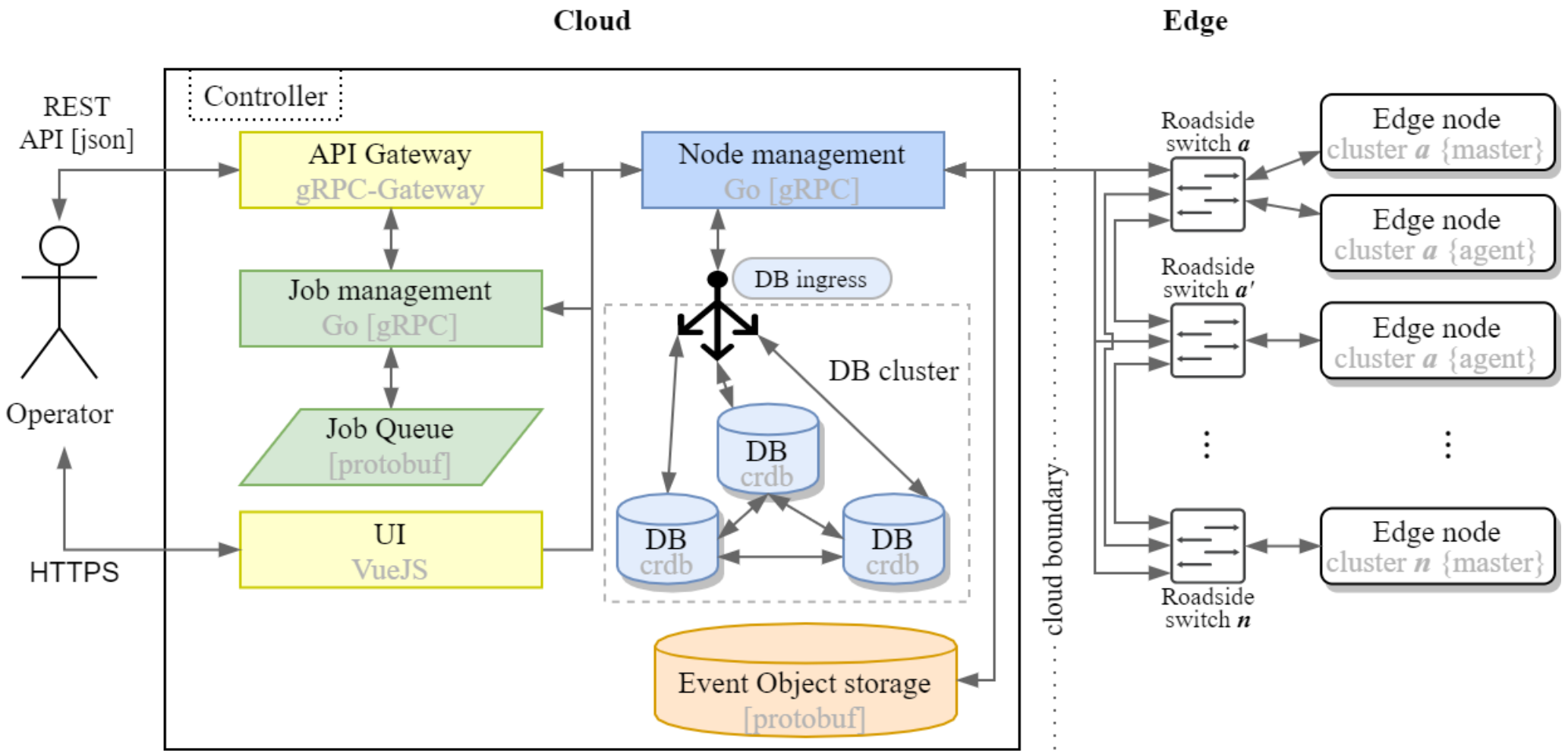
My other research: Embedded Deep Learning



Size	128	160	192	224	128	160	192	224	128	160	192	224	128	160	192	224
Depth	0.25	0.25	0.25	0.25	0.50	0.50	0.50	0.50	0.75	0.75	0.75	0.75	1.00	1.00	1.00	1.00
Pi3	11.60	8.30	7.10	6.40	6.90	5.90	4.60	3.80	4.20	3.20	2.80	2.20	3.90	2.90	2.40	2.10
Pi3 w/NCS	69.71	59.52	48.89	41.15	60.77	47.22	37.61	30.27	47.77	34.94	29.95	23.61	38.20	28.62	22.53	17.21
NCS Speedup	601%	717%	689%	643%	881%	800%	818%	797%	1137%	1092%	1070%	1073%	979%	987%	939%	820%

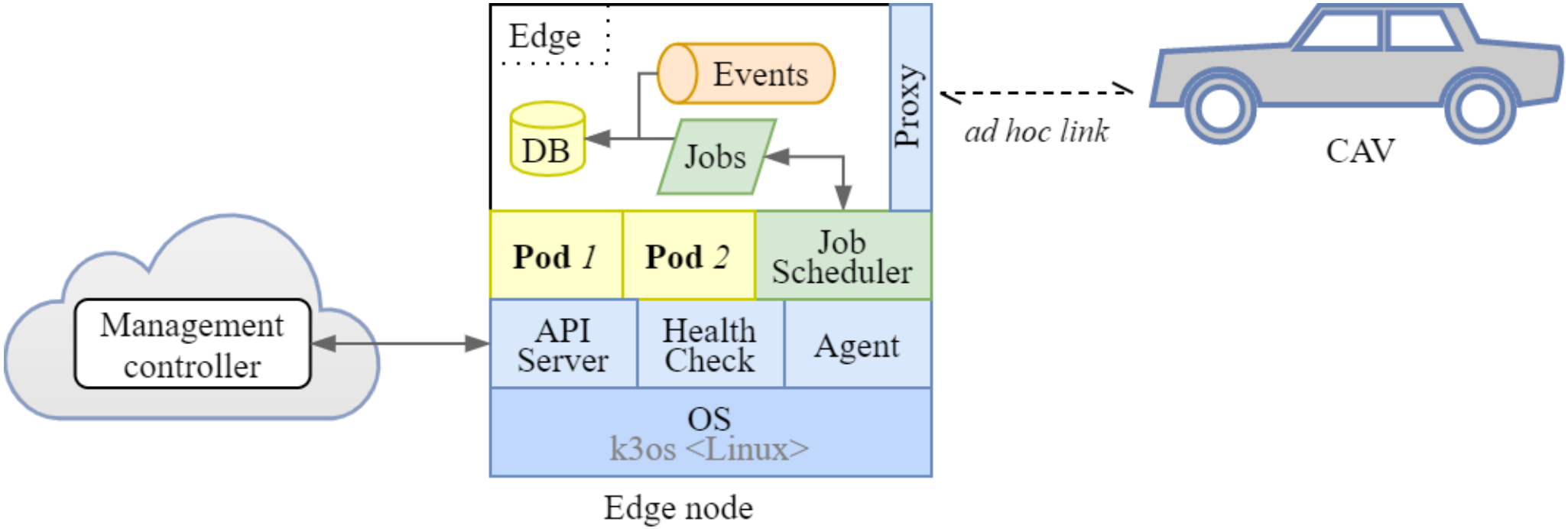
MobileNets RPi3B (FPS): CPU only vs. 1 x NCS

Conclusion



Example Node management platform using Protobufs as data format for serialization and storage, along with gRPC for remote service calls.

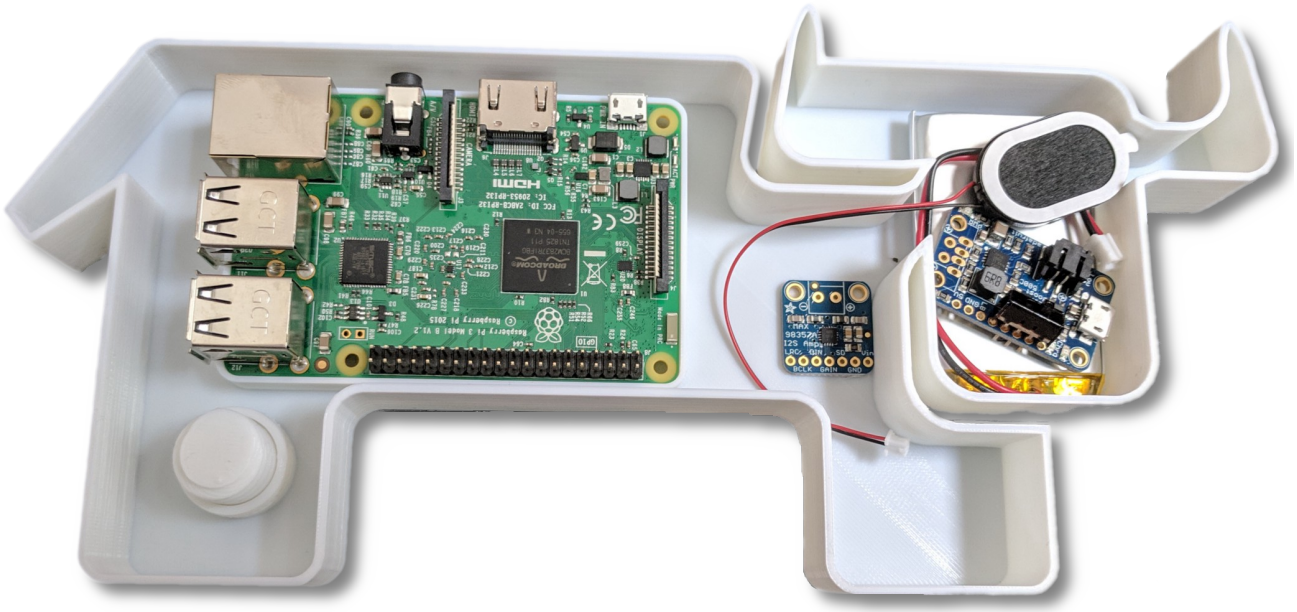
Conclusion



Example Edge node using application orchestration to deploy two Pods for microservices.

Future work

- Large scale testing of Edge clusters using kubernetes OS, as miles of road can be simulated with a few small-board computers
- Creation of Edge-focused Trust Management Platform
- Negotiation and sharing of data between Edge and CAV systems



k3s Raspberry Pi cattle case for Rancher cluster (Mark Abrams 2019)

Publications

- *Embedded deep learning for vehicular edge computing*,
  2018 IEEE/ACM Symposium on Edge Computing (SEC)
- *An optimal police patrol planning strategy for smart city safety*,
  2016 IEEE 14th Intl. Conf. on Smart City
- *Low-latency high-level data sharing for connected and autonomous vehicular networks*,
  2019 IEEE Intl. Conf. on Industrial Internet (ICII)
- *Reliability characterization of solid state drives in a scalable production data center*,
  2018 IEEE Intl. Conf. on Big Data (Big Data)
- *An empirical study of quad-level cell (QLC) NAND flash SSDs for big data applications*,
  2019 IEEE Intl. Conf. on Big Data (Big Data)
- *Incorporate proactive data protection in ZFS towards reliable storage systems*,
  2018 IEEE Intl. Conf. on Big Data Intelligence and Computing(DataCom)
- *Developing cost-effective data rescue schemes to tackle disk failures in data centers*,
  2018 Intl. Conf. on Big Data

Thank you

Jacob Hochstetler
PhD Dissertation Defense
Dr. Song Fu
Dr. Rodney Nielsen
Dr. Armin Mikler
Dr. Ryan Garlick
jacobhochstetler@my.unt.edu (mailto:jacobhochstetler@my.unt.edu)
http://github.com/jh125486 (http://github.com/jh125486)