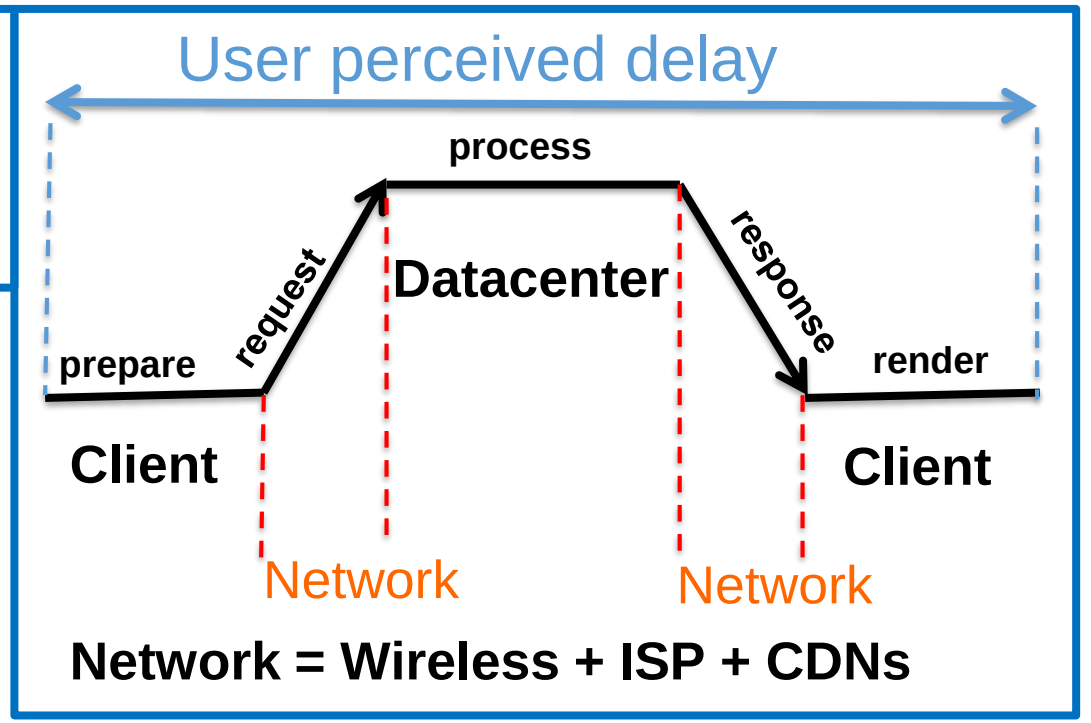
Georgia Institute of Technology

Fast, Flexible and Secure Onloading of Edge Functions using AirBox

Ketan Bhardwaj, Ming-Wei Shih, Pragya Agarwal, Ada Gavrilovska, Taesoo Kim, Karsten Schwan

Edge Computing Drivers

Latency matters: Better user experience, revenue



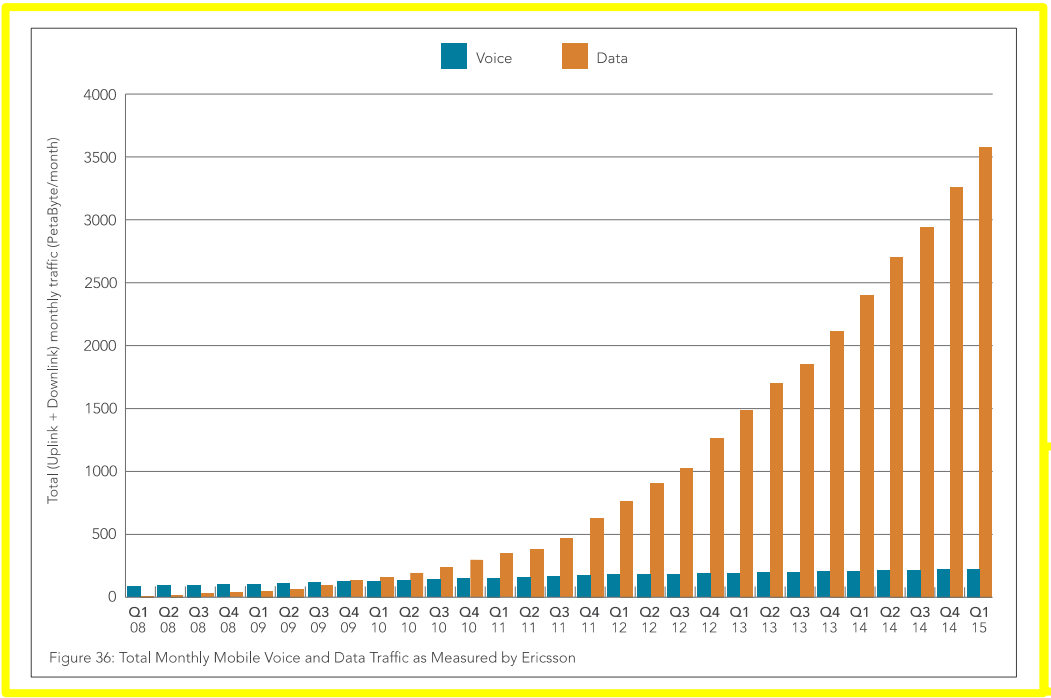


Figure 36: Total Monthly Mobile Voice and Data Traffic as Measured by Ericsson

Bandwidth Costs: For end users and service providers

Source: Akamai state of Internet report Q1'2015

Alternative: Build more, bigger data centers

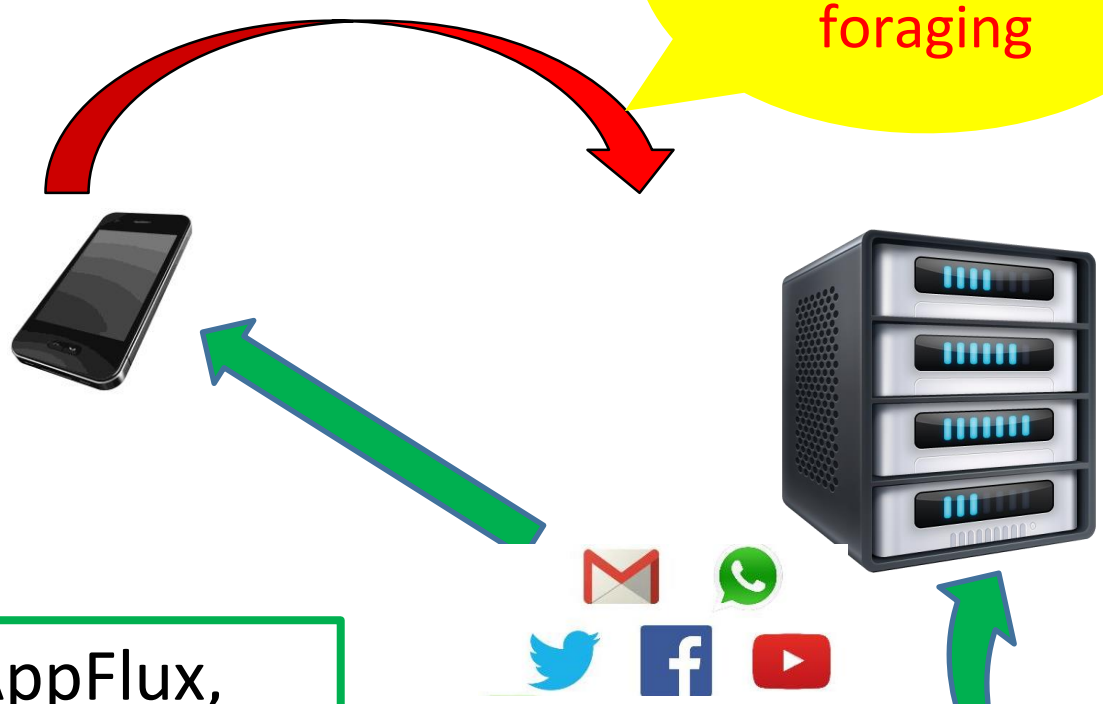
- Capital intensive to build.
- Expensive to operate.

- To achieve 1 ms latency a data center every 300 KM

- Sheer number of bytes that need to travel over Internet
- No control over network

- 3.5 ZB per year by 2019 (Cisco)
- Would take 8 years on 800 Gbps connection

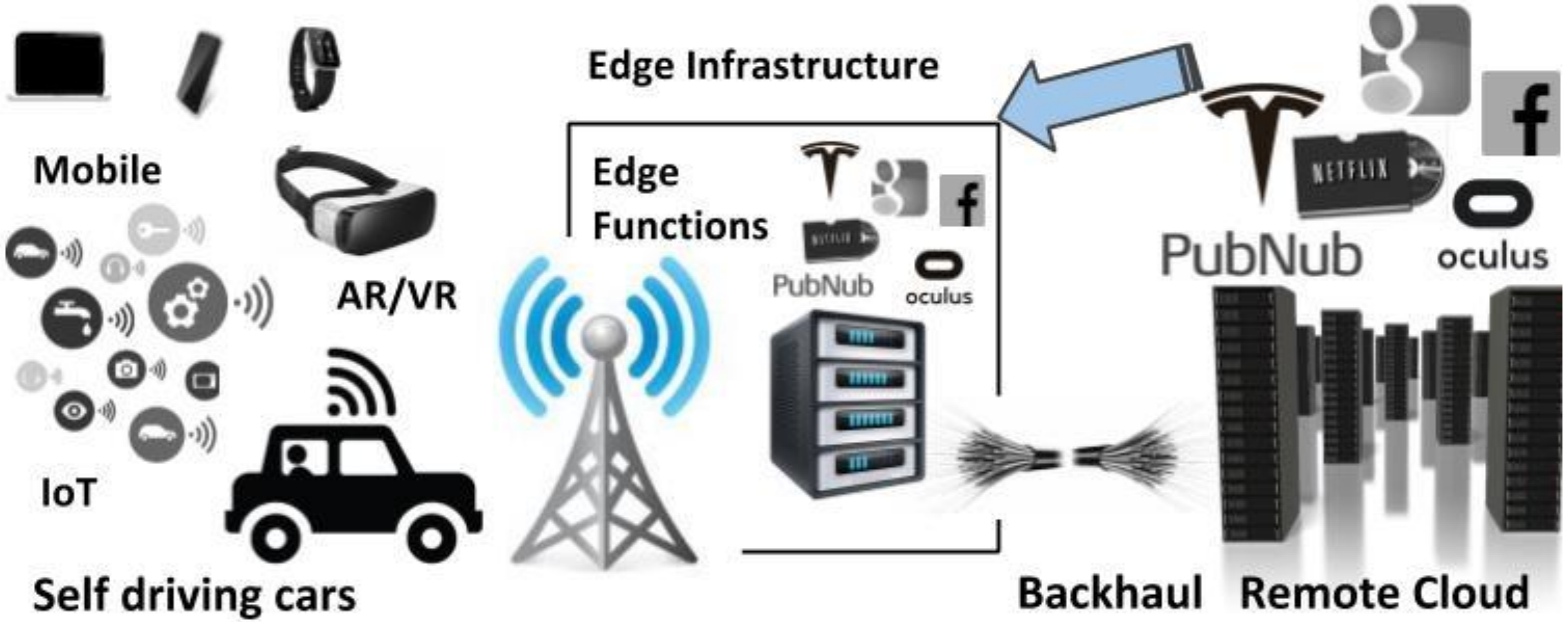
Approaches



Offloading: MAUI, CloneCloud, Comet, PCloud, ECC, COSMOS, …

Onloading: AppFlux, AppSachet, Cloudlet, FlyWheel, CDNs, micro-data centers, …

Onloading: Backend driven Edge Computing



Lets speak the same language …

- Beyond which users only have wireless access - ***Edge***
- Infrastructure – ***Edge cloud, Cloudlet, Fog server, …***
- Services running on edge infrastructure – ***Edge Functions***

Edge Function (EF)

- **Definition:** Any third party service deployed on edge infrastructure that interacts with end client requests on behalf of a backend service deployed in remote clouds.
  - *Typically implemented above layer 3*
  - *Employs application specific knowledge*

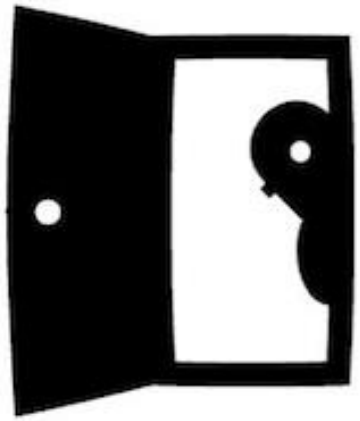
- **Edge function platform (EFP):** Software platform that enables Edge functions to be deployed at the edge

High Level Intuitive Choices

- Leverage cloud model for the edge computing
  - Use of virtualization to enable arbitrary edge functions
- Dynamic just-in-time deployment model
- Secure Edge nodes, Edge functions and their stored state

Challenges for an Edge function Platform

Developer Flexibility



Questions raised in this paper

- What type of virtualization to use for EFs?
  - *OS agnostic hypervisor – Virtual machines*
  - *OS level virtualization – OS Containers*
  - *Application level virtualization – Sandboxes*

- How to handle security concerns of edge functions?
  - Are they different from cloud security concerns?

Technology Space Exploration: Provisioning

- Chosen Technologies
  - Virtual machines - **Cloudlet**
  - Containers - **Docker**
  - Sandboxes - **Embassies**
- Constraints on developers
  - Cloudlet – None
  - Docker – OS
  - Embassies – Porting

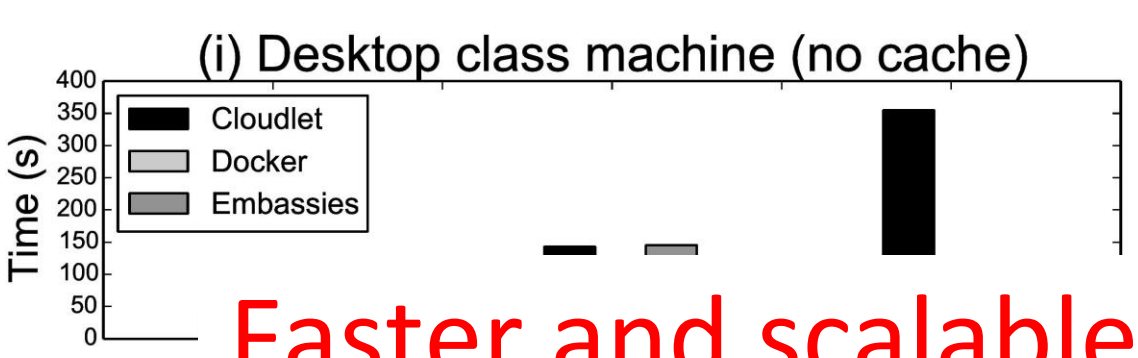
- Other Technologies
  - Java virtual machines
  - Native Sandboxes e.g., Chrome's NaCl
  - Runtimes: node.js
  - Unikernels: Jitsu
- Constraints on developers
  - Specific Toolchains
  - Lack of optimized libraries
  - Deployment packaging

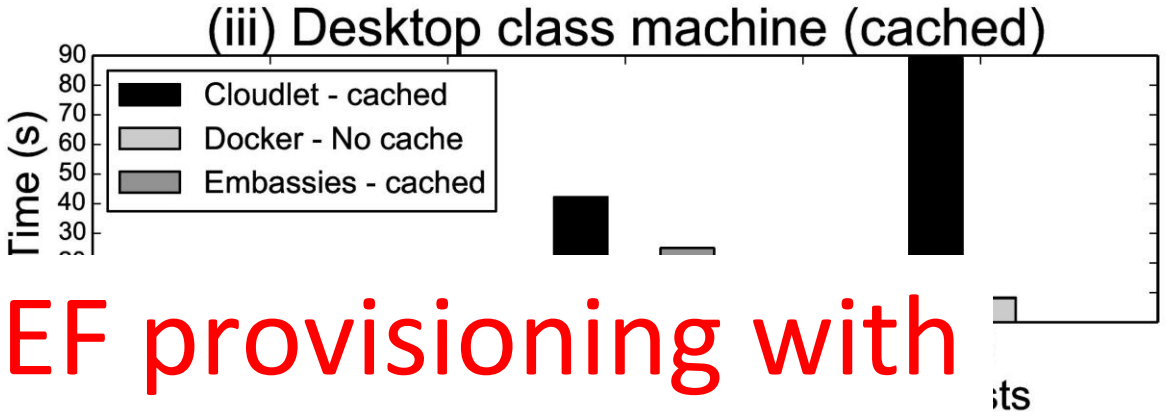
Experimental Setup

Type	Deployment scenario	Hardware configuration
Mini edge	Strategic placed server racks by mobile networks operators or enterprises - Server class machine	Intel x86-64, 24 CPUs, 1.6 GHz, 50 GB RAM, 4 NUMA nodes, 2 sockets, 6 cores per socket, 2 threads per core, VT-x, L1 (i+d): 64 KB, L2: 256 KB, L3: 12 MB
Micro edge	Randomly placed standalone servers by businesses or individuals - Desktop class machine.	Intel x86-64, 4 CPUs, 1.6 GHz, 4 GB RAM, VT-x, L1 (i+d): 64 KB, L2: 4096 KB

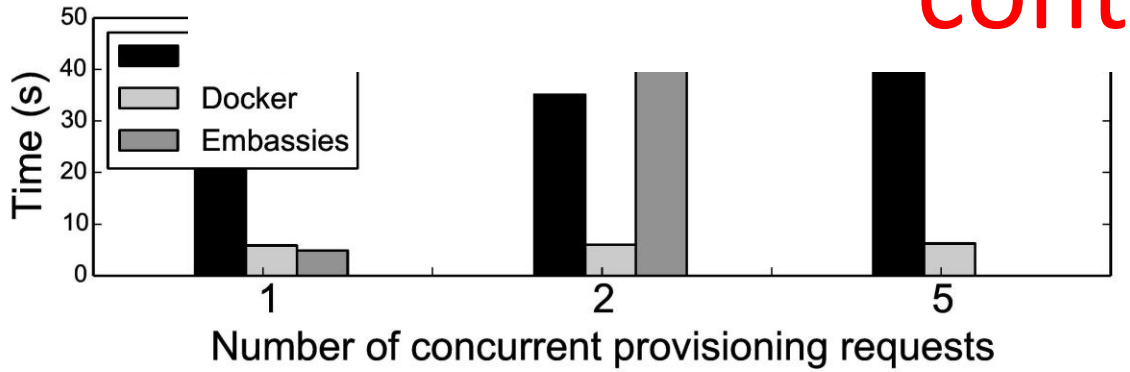
OS: Ubuntu 14.04 LTS, Ported all chosen solutions to it.
Edge function: Image processing using exactimage library, Think Instagram filters.

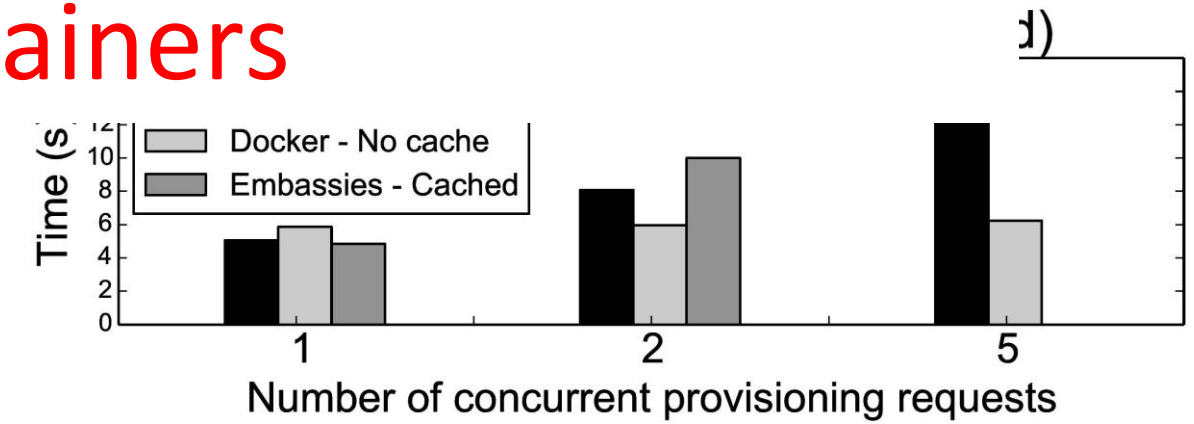
Provisioning Speed and Scalability



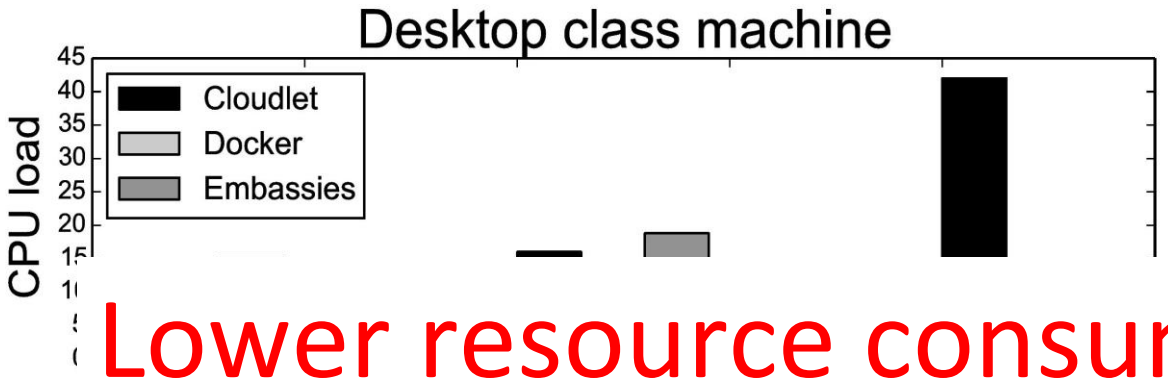


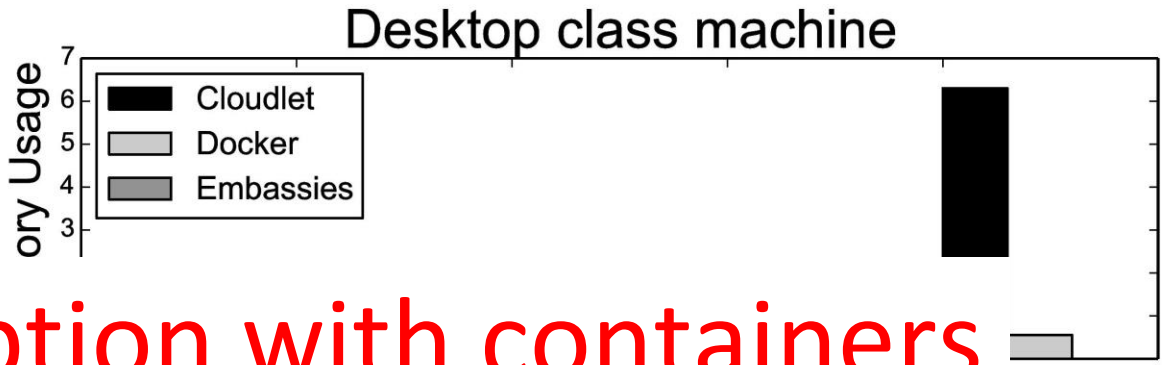
Faster and scalable EF provisioning with containers



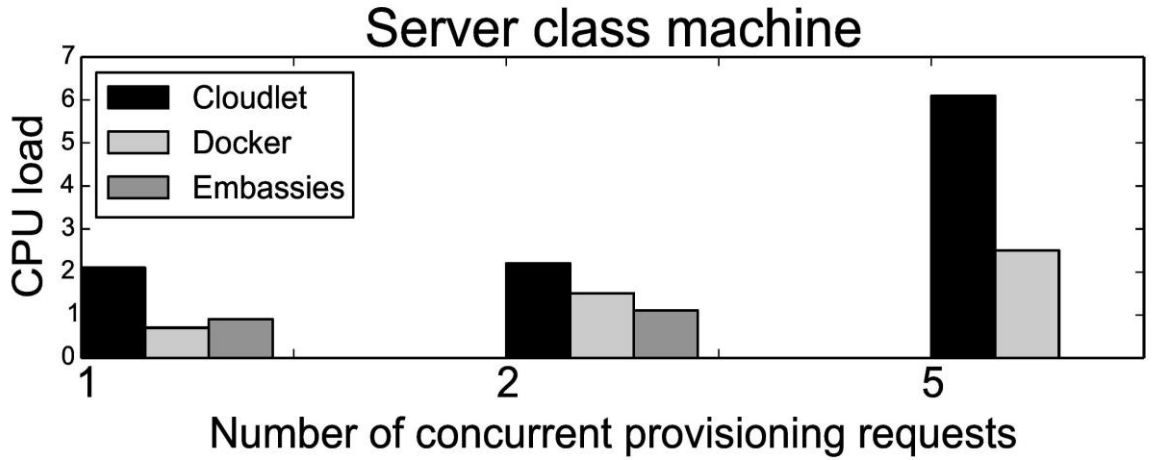


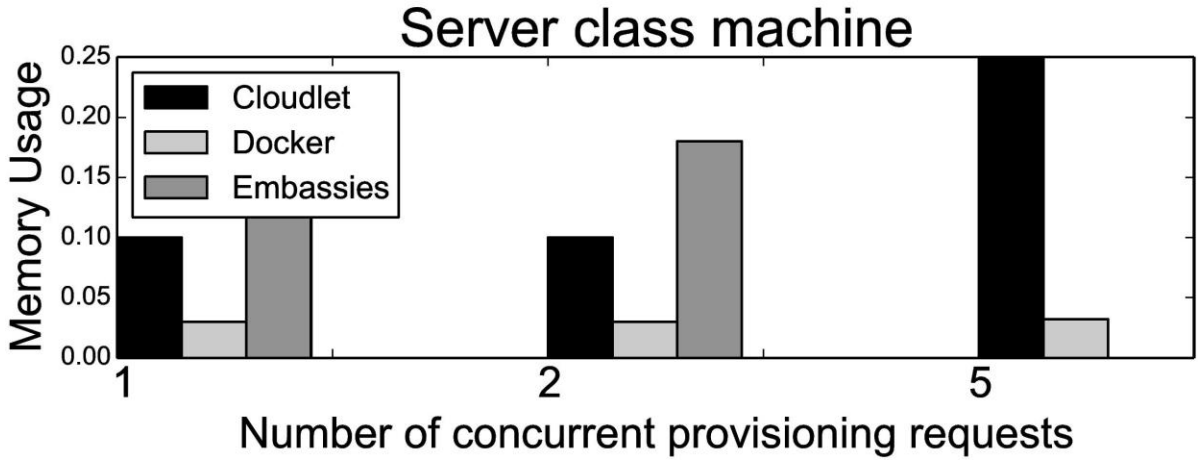
Provisioning Resource Consumption





Lower resource consumption with containers





Questions raised in this paper

- What type of virtualization to use for EFs?
  - *OS agnostic hypervisor – Virtual machines*
  - *OS level virtualization – OS Containers*
  - *Application level virtualization – Sandboxes*

- How to handle security concerns of edge functions?
  - Are they different from cloud security concerns?

EF Security Concerns

1. Integrity verifica cation of d EF code
2. Execution security: g EF execution
3. State confidentiality: ed by EF on edge cloud
4. End user priva hat e ests remain private

Approaches to Security

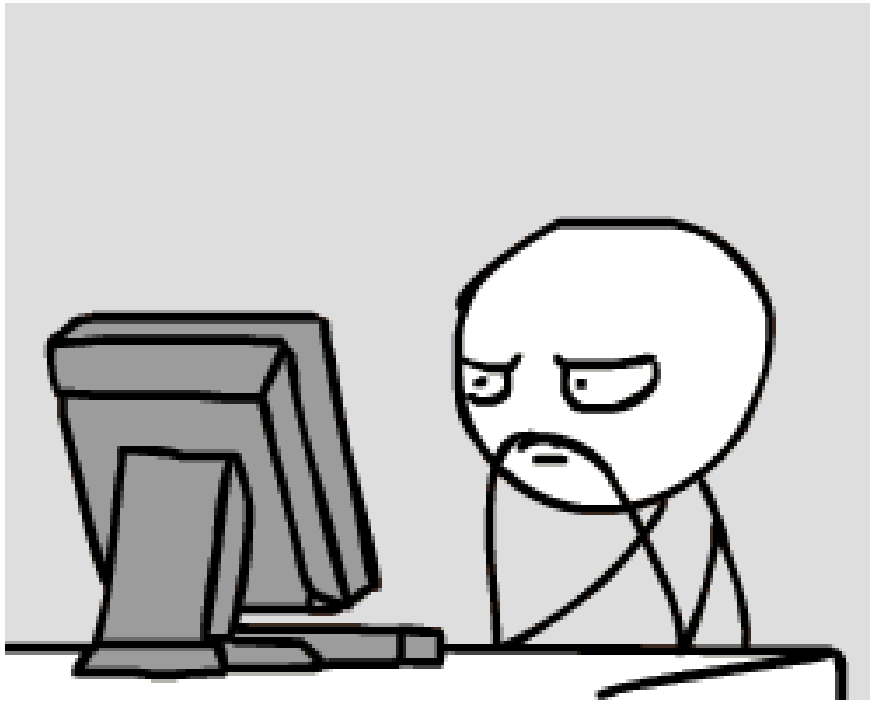
- Cloudlet VMs
  - TrustVisor – formally verified VM, InkTag – verification
- Embassies Sandboxes
  - Cryptographic attestation
- Docker Containers
  - Docker registry, namespaces, SELinux, AppArmour
- Haven
  - Narrow system call interface, use of libOS and Intel SGX
- VC3 – solution to secure map reduce via SGX based verification of results

AirBox: An Edge Function Platform

- Using Docker containers with SGX based integrity verification boot block for deployment vehicles for EFs
- Designed as a console for EF managers and provisioner module to be deployed for edge cloud nodes
- Prescribes a secure EF anatomy using Intel SGX for security concerns
  - Implemented on top of OpenSGX

Intel SGX

- New secure instructions in Intel processors
  - Loading, entry and exit to SGX enclave
  - For OS to allocate PRM page/eviction
- Processor reserved memory hashed after enclave load till its exit
- New processor mode & related HW structures
- Unforgeable attestation qoute generation
- Remote attestation
- No I/O or interrupts in/from enclave
- Difficult to setup SGX enclave debugging



AirBox Benefits

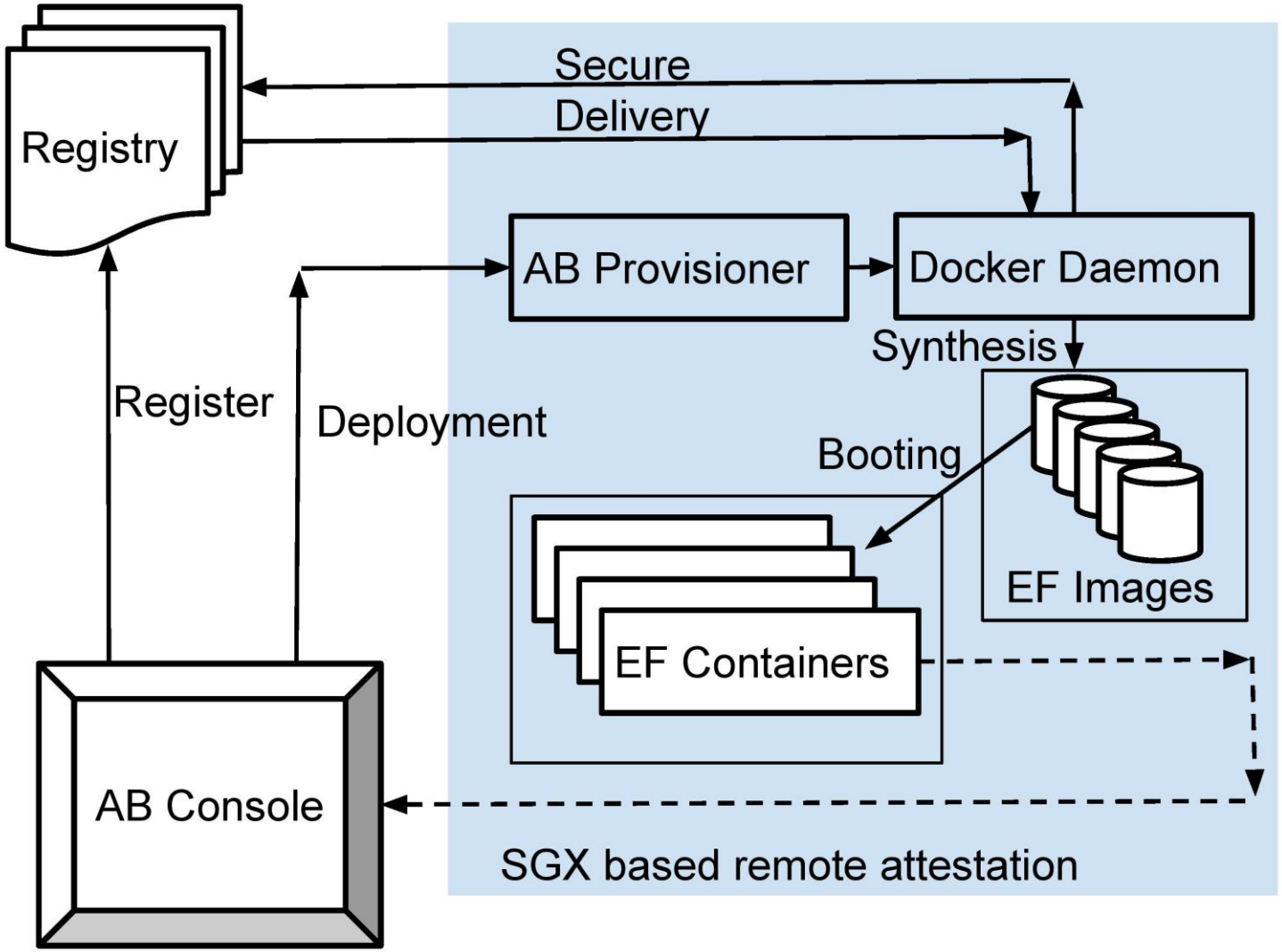
- Integrates seamlessly with Docker ecosystem making it easy to create, package and deliver EFs
- A single interface to deploy EFs, a single module for edge nodes and easy remote attestation of enclave in an EF
- Abstracts Intel SGX provided features to provide intuitive API to an EF developer or EF manager

AirBox – SGX interface

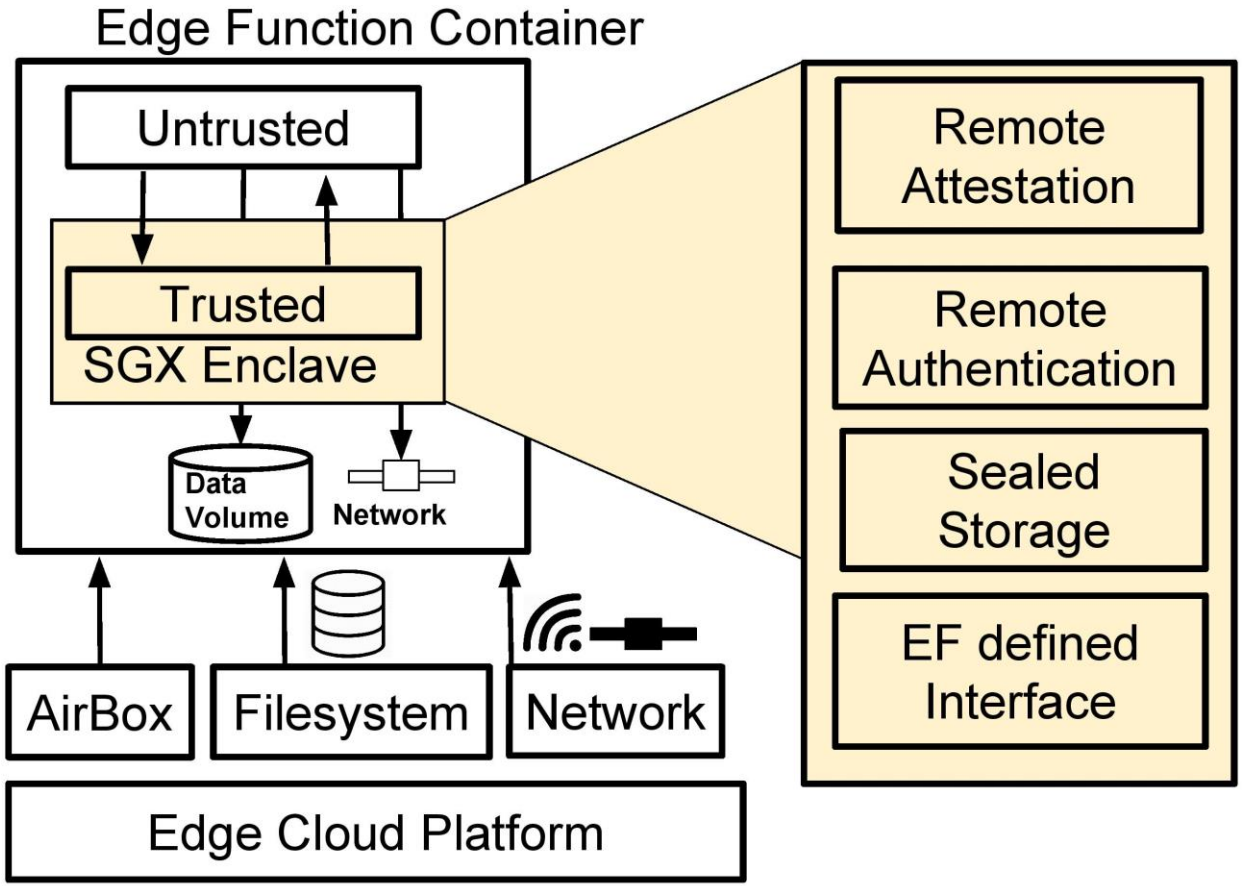
- Remote Attestation: `airbox_sgx_attest(quote)`
- Remote Authentication: `airbox_sgx_auth(quote)`
- Sealed Storage:
  - `airbox_sgx_get(key, len)`
  - `airbox_sgx_put(key, klen,*value, *vlen)`
  - `airbox_sgx_getkeys(*keys, len)`
- EF defined:
  - `airbox_sgx_run(module, conf)`

- **Ease of use:** Focus on the functionalities
- **Performance:** minimize SGX overhead – enclave page cache, TLB thrashing, …
- **Security:** Enclave can be compromised by incorrectly using system calls in host part of SGX application

AirBox EF provisioning



AirBox Secure EF anatomy

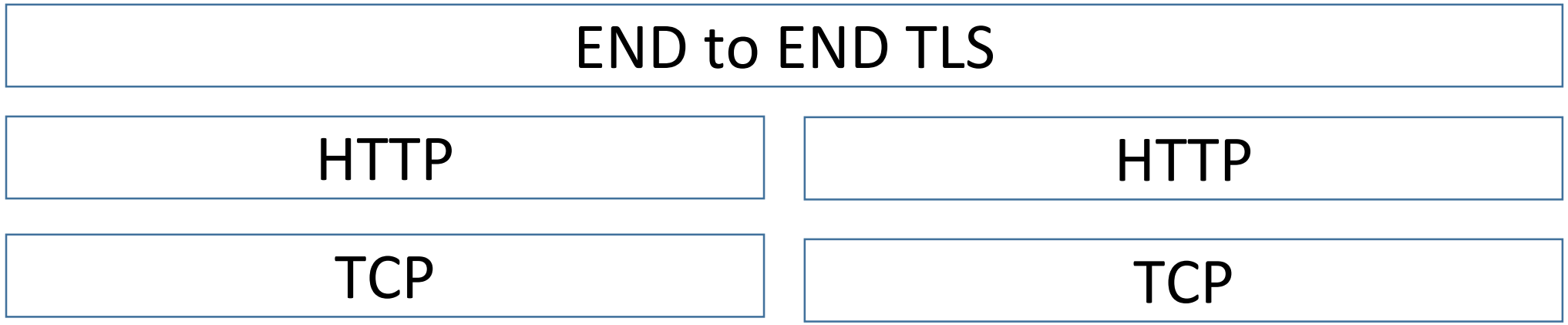


SGX features in AirBox

- Remote Attestation
- Sealing

Preserving end user privacy at Edge

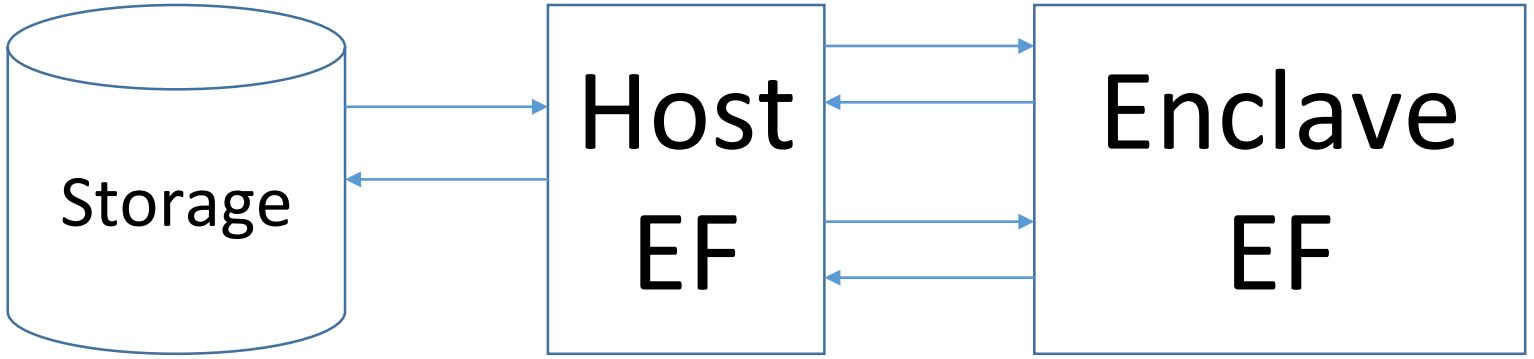
- EF acquires TLS session key in enclave
- EF saves it using sealing for session duration
- EF Decrypt user requests and their responses inside enclave
- Even AirBox just sees encrypted blobs going network



Secure split connection at edge using SGX

Securing Storage at Edge

- EF cannot direct read/write from inside of enclave
- 2 phase disk I/O: Phase 1: meta-data, Phase 2: data
- AirBox reads and writes only encrypted blobs from/to disk



Secure storage at the Edge using SGX

Implementation Details

- Using stable Docker release on Ubuntu 14.04
- SGX functionality prototyped using OpenSGX
- Generic edge functions for SGX impact

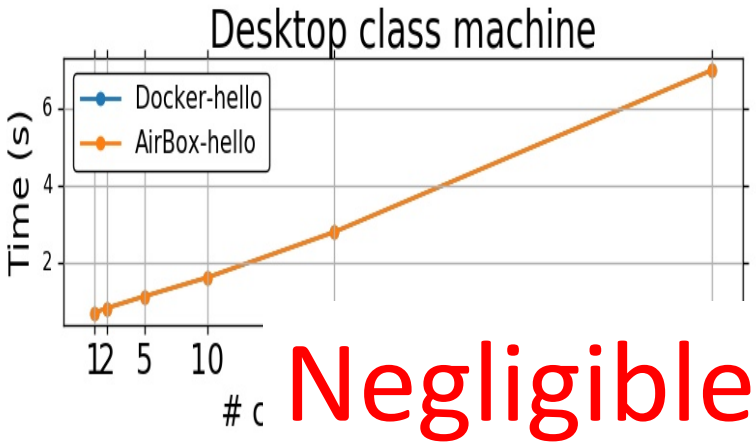
- **OpenSGX**: A qemu based software platform that provides necessary support for SGX application programmers to readily implement and evaluate their applications that leverage

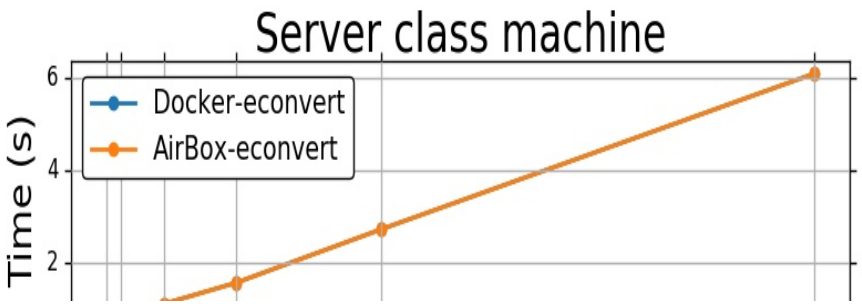
Experimental Setup

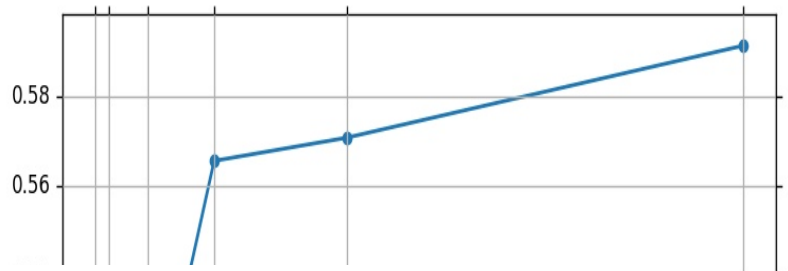
Type	Deployment scenario	Hardware configuration
Mini edge	Strategic placed server racks by mobile networks operators or enterprises - Server class machine	Intel x86-64, 24 CPUs, 1.6 GHz, 50 GB RAM, 4 NUMA nodes, 2 sockets, 6 cores per socket, 2 threads per core, VT-x, L1 (i+d): 64 KB, L2: 256 KB, L3: 12 MB
Micro edge	Randomly placed standalone servers by businesses or individuals - Desktop class machine.	Intel x86-64, 4 CPUs, 1.6 GHz, 4 GB RAM, VT-x, L1 (i+d): 64 KB, L2: 4096 KB

OS: Ubuntu 14.04 LTS, Ported all chosen solutions to it.
Edge function: Image processing using exactimage library, Think Instagram filters.

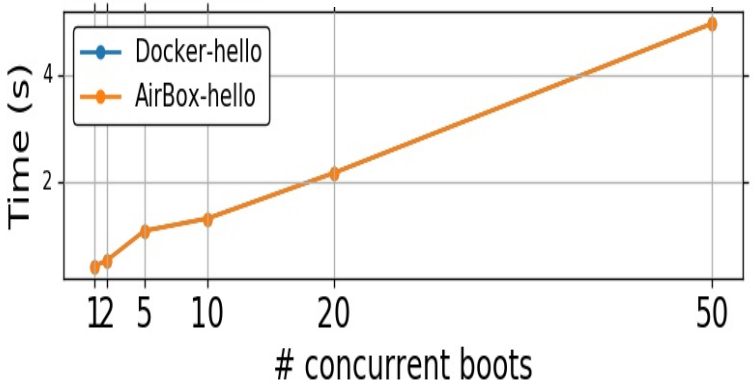
AirBox provisioning performance

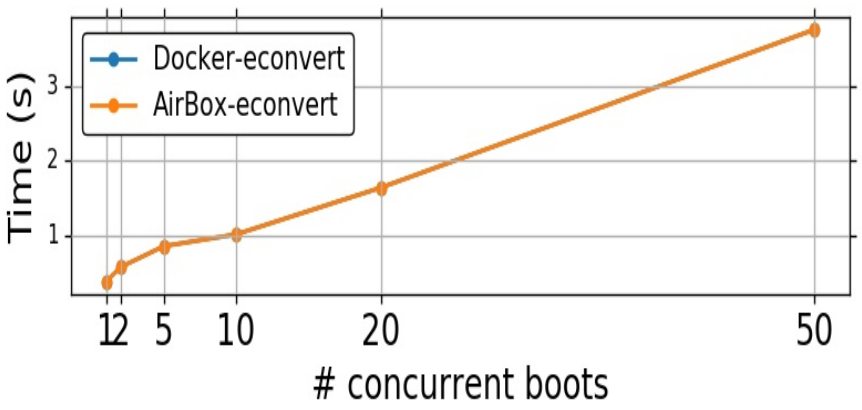


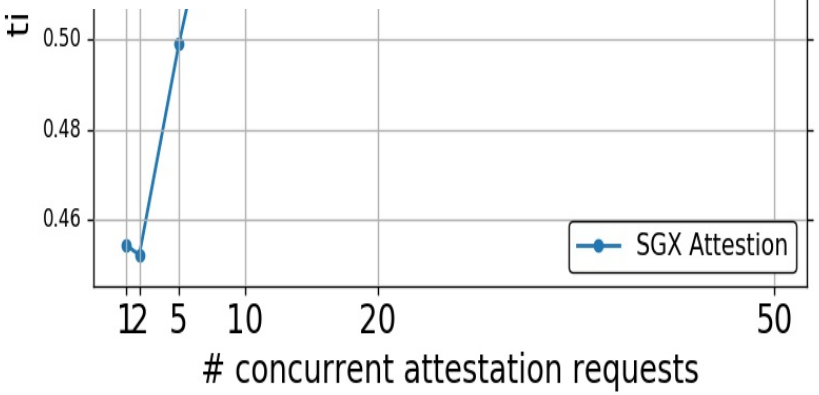




Negligible attestation overhead (< 1ms)



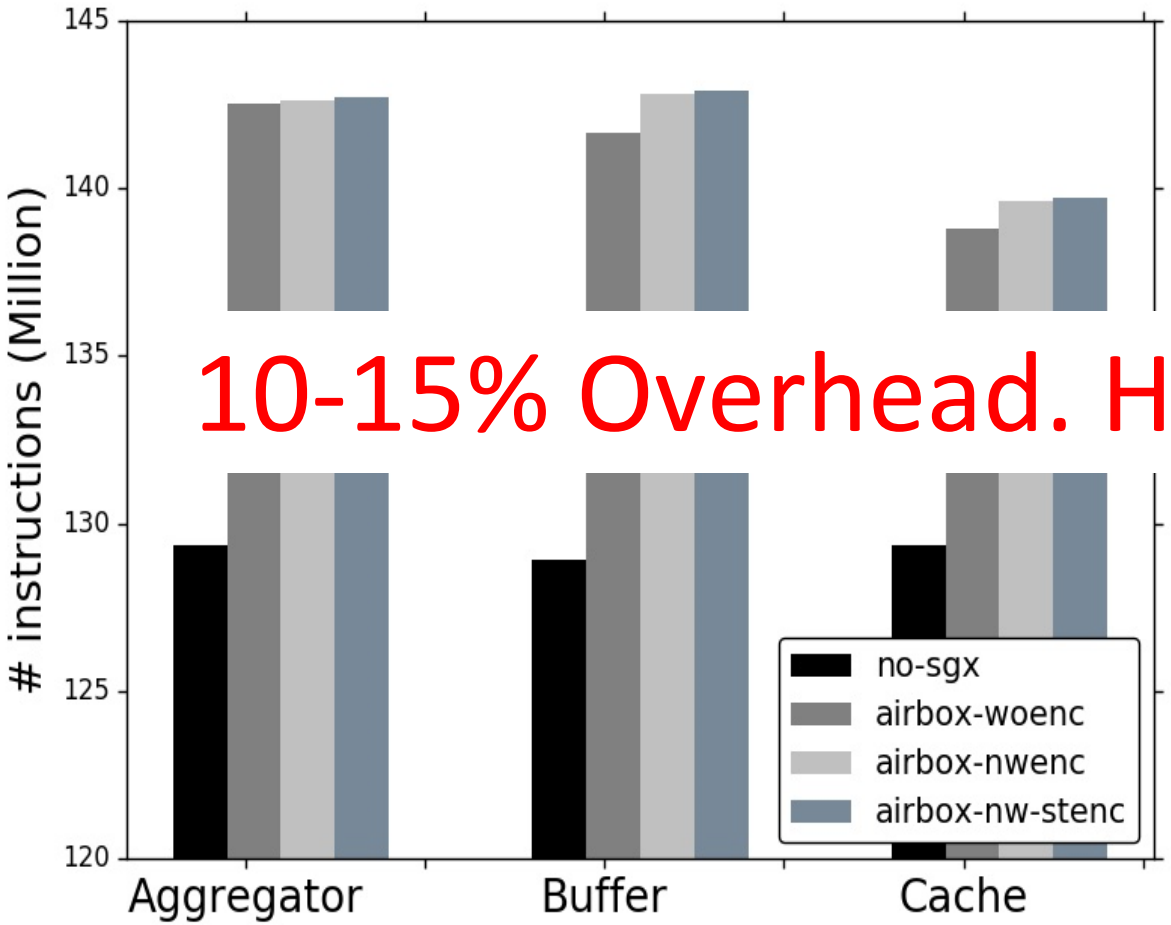


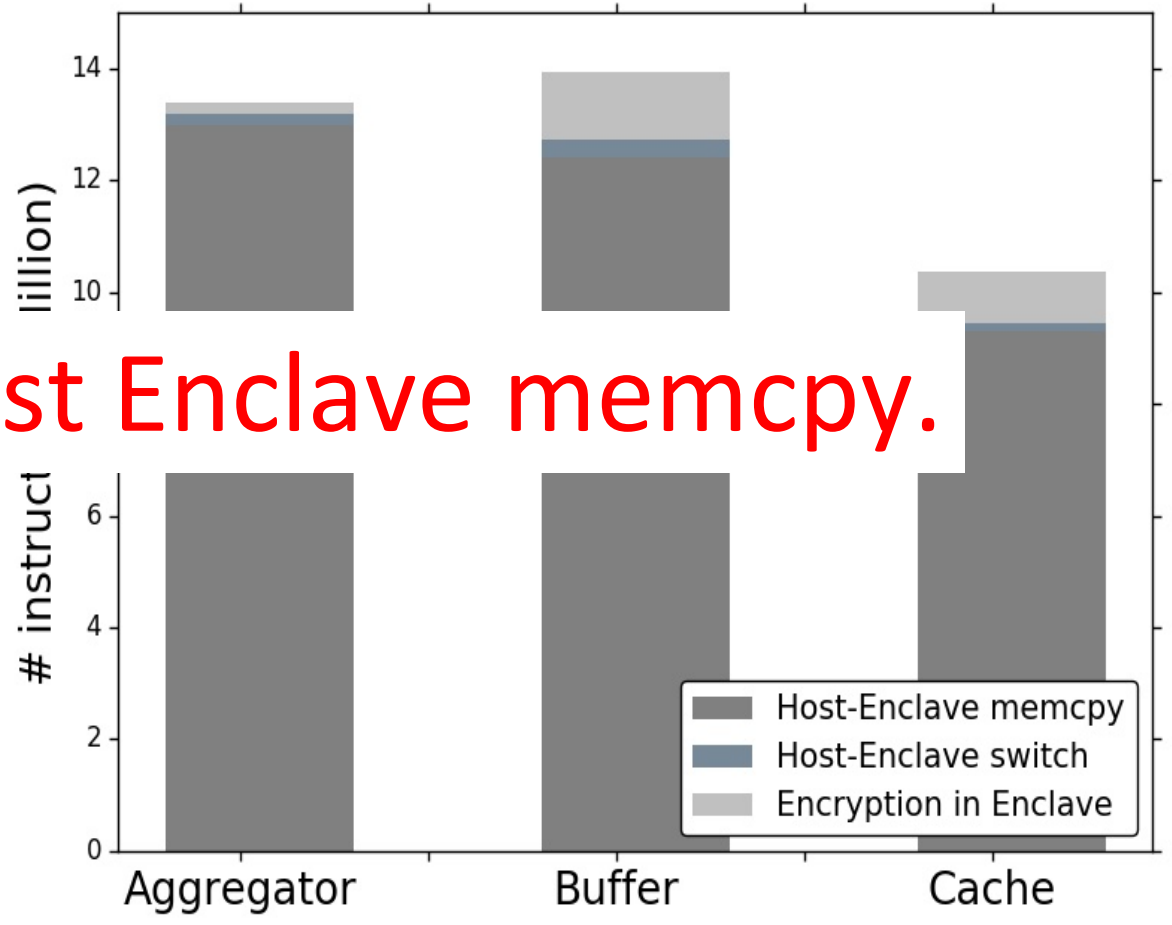


SGX overhead: Generic Edge Functions

EF Benefit	**Aggregation**	**Buffering**	**Caching**
Latency	x		x
Bandwidth	x	x	x
Edge resource			
Compute	x	x	
Storage	x	x	x

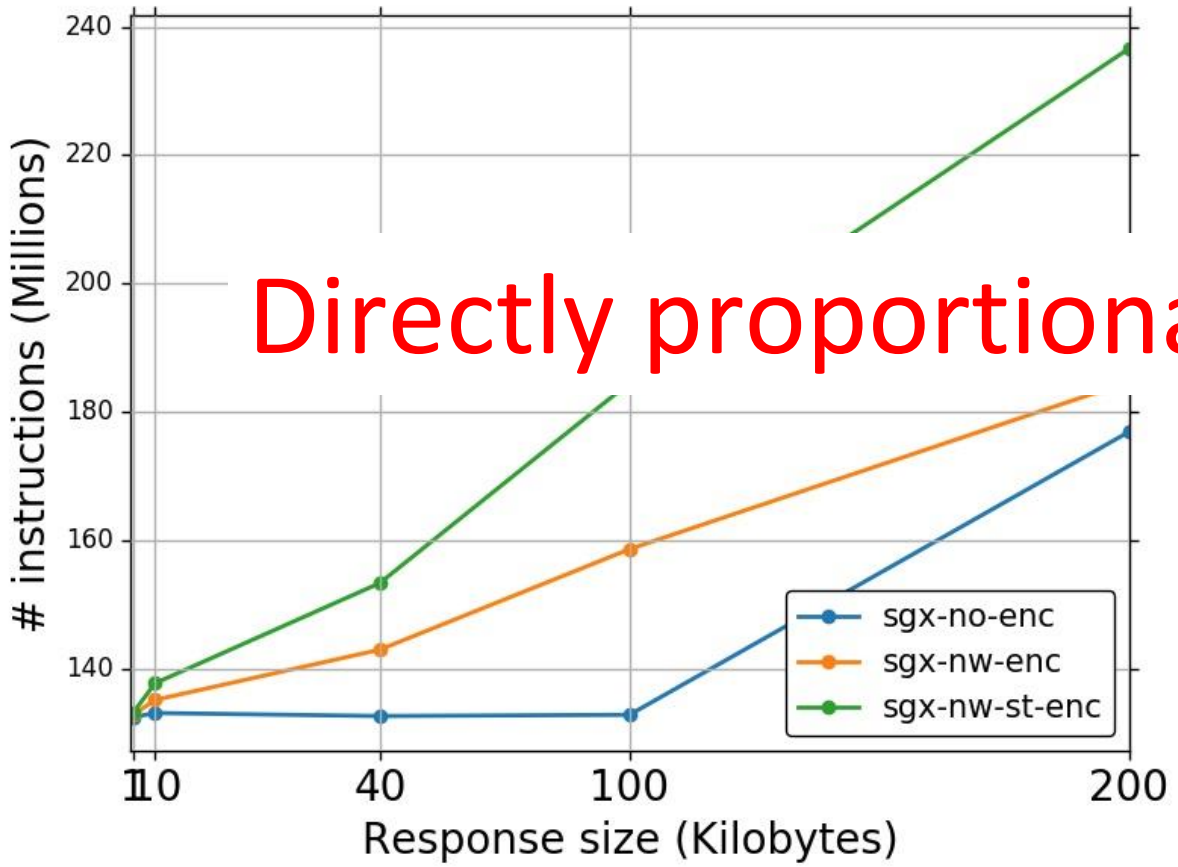
SGX Overhead: ABC usecases

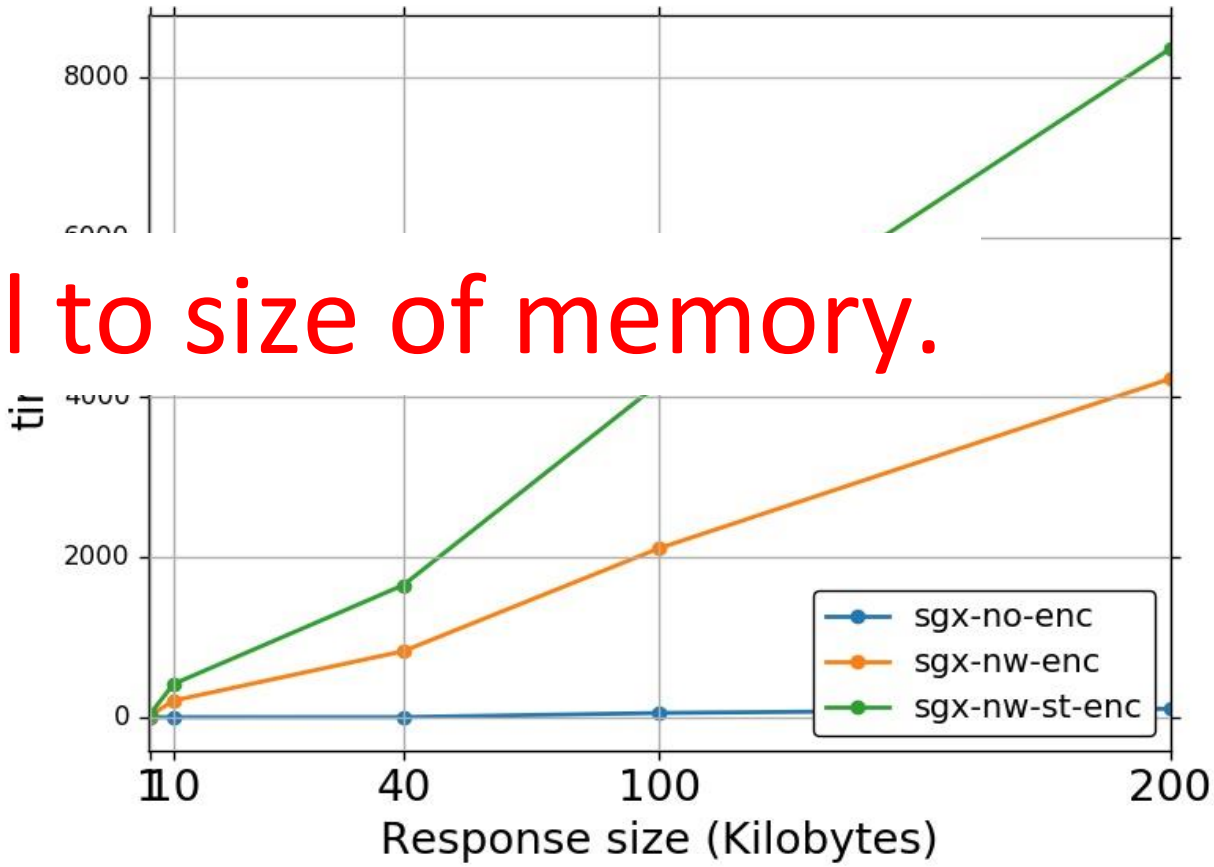




10-15% Overhead. Host Enclave memcpy.

memcpy: OpenSGX vs. Real Hardware





Directly proportional to size of memory.

Deployment Scenarios

- In mobile networks to enable performant and secure edge computing
- In enterprise networks, as part of vCPE equipment for better price to performance ration and securing valuable services
- In military tactical edge, where security concerns are paramount in case of a compromise

Next steps

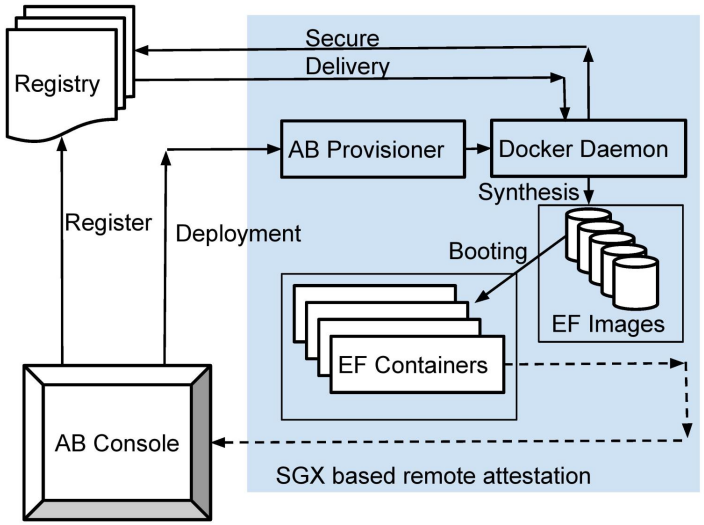
- Already ported in real SGX hardware
  - After SGX linux SDK was released (mid 2016)
- Protocol level solutions for handling traffic over secure protocols
- Evaluate on real mobile infrastructure – 4G LTE, …
- Formal model of Edge Functions and their benefits

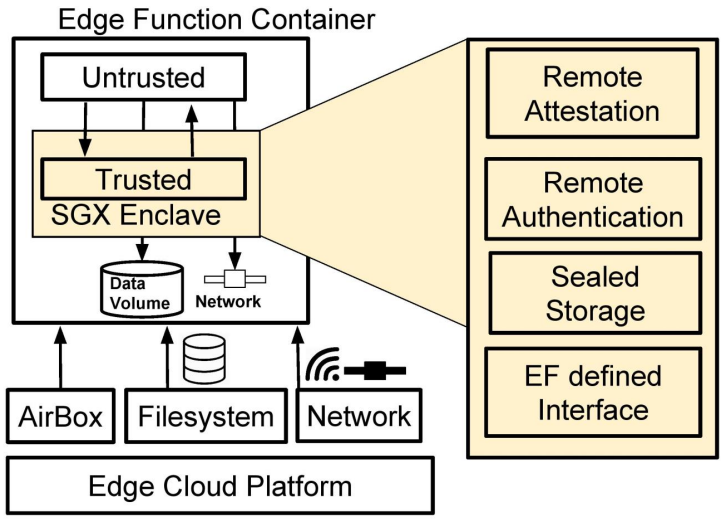
Summary

- Introduced the notion of Edge Functions
- Design of AirBox based on empirical analysis
  - Integrated with Docker eco-system
- Simplify use of Intel SGX for EF security
  - AirBox secure Interface
- Experimental demonstration
  - AirBox delivers competing benefits in terms of deployment
  - Speed, costs (in terms of resource consumption) and developers constraints
  - EF can be secured with ~10% runtime impact.

Questions?

AirBox





Contact:
Ketan Bhardwaj (ketanbj@gatech.edu)