Edge Computing: Where Mobile Meets the Cloud

Padmanabhan (Babu) Pillai

15-719
April 17, 2019

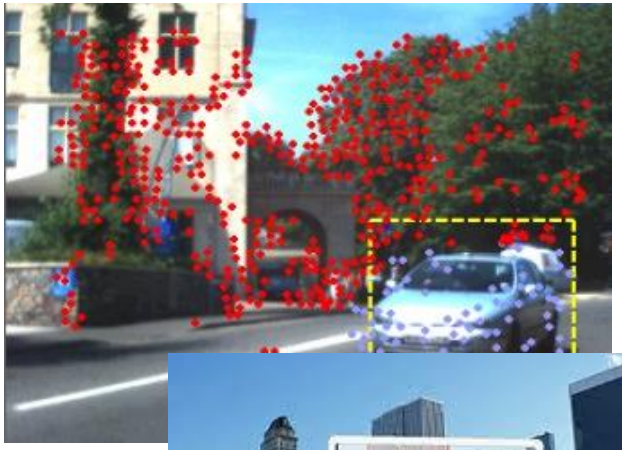
Mobile devices use the cloud

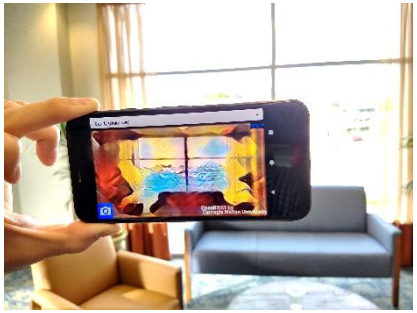
- Mostly as general Internet clients, video consumption
- Mobile (cellular) devices exceeded human population by Oct. 2014
- “smartphones consumed 92% of global mobile data traffic, despite only making up 18% of the handsets in use globally” – Cisco report by way of the Guardian
- “… the explosive growth of all of these devices connecting to the Internet is driving a $10 billion dollar server business.” – Stacey Smith, Intel
- Internet of Things – 13.4 billion connected things in 2015, will grow to 38 billion by 2020 – Juniper Research

Diverse Cloud-based Mobile Apps

- Many are small front-end applets:
  - News/weather tools, web data front ends, e-readers, simple online games, etc.
- Others push beyond capabilities of devices alone
  - Leverage Computing in Cloud, Big Data

Interactive Application Domains





Emerging Mobile apps

- Pushing beyond capabilities of devices
- Interactive, sensing applications using computer vision, inferencing, machine learning

- Need processing capacity of clouds, but with strict timing requirements

- May require rethinking cloud architecture

  → Cloudlets / Edge Computing

Pervasive 2009 paper [1]

- Position paper making the case for cloudlets
- Introduces Cognitive Assistance use case on mobile device
  - Presages a Google glass type device
  - Use computer vision to understand scene, recognize individuals
  - System names people, provides reminders

[1] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, "**The Case for VM-Based Cloudlets in Mobile Computing**," Pervasive Computing, October, 2009.

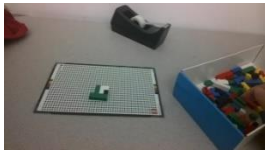
Interactive Assistive Applications

- Form of Augmented Reality
- High computational needs
- Tight latency requirements

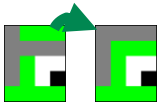
Chen et al., 2017 [3]

DNN face recognition → "Obama"

Face



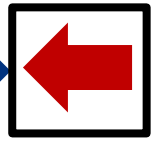
Contour/color detection →



"Find a piece of 1x3 brick and put on top"

Lego

Color/line detection →



Pool

3rd party "drawing assistant" software →

Draw

Activity recognition → "8"

Work-out

DNN object detector →

"Now put a piece of bread on the lettuce"

Sandwich

Do we really need to offload?

- Today's smartphone are getting more powerful
  - Multi-core CPUs
  - GPUs
- Advanced algorithms running on phones already
  - E.g., Built-in support for face detection in Android SDK
- But, computation varies dramatically depend on operational conditions
  - Face recognition
  1. Increasing number of possible faces,
  2. Reducing the constraints on the observation conditions

Persistent Mobile Performance Gap

Year	Typical Server Processor	Speed	Typical Handheld or Wearable Device	Processor Speed
1997	Pentium® II	266 MHz	Palm Pilot	16 MHz
2002	Itanium®	1 GHz	Blackberry 5810	133 MHz
2007	Intel® Core™ 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2011	Intel® Xeon® X5	32 GHz (2x6 cores)	Samsung Galaxy S2	2.4 GHz (2 cores)
2013	Intel® Xeon® E5-2697v2	64 GHz (2x12 cores)	Samsung Galaxy S4	6.4 GHz (4 cores)
			Google Glass	2.4 GHz (2 cores)
2016	Intel® Xeon® E5-2699v4	96.8 GHz (2x22 cores)	Samsung Galaxy S7	7.5 GHz (4 cores)
			HoloLens	4.16 GHz (4 cores)
2017	Intel® Xeon® E7-8894v4	115.2 GHz (2x24 cores)	Pixel 2	9.4 GHz (4 cores)

Adapted from J. Flinn, *Cyber Foraging: Bridging Mobile and Cloud Computing via Opportunistic Offload*. Morgan & Claypool Publishers, 2012.

Energy limits on Mobiles

- Even if mobile fast enough, may not want to spend the energy
- Battery technology advancing slowly compared to CPU, networks, storage
- Performance of mobile often limited by thermals

Empirical need for offload

High variability of execution time

1. *SPEECH recognition*

- Execution time increases with the number of recognized words

	no words recognized	1-5 words recognized	6-22 words recognized
Measured average time	0.057 s	1.04 s	4.08 s

2. *FACE recognition*

- Execution time varies depends on the contents of images

Image with single big face	Image with no face
0.30 s	3.92 s

* Tested with a netbook as a mobile device

More support for offload

Improvement from Cloud offloading

- Though variability still exists, the absolute response times are improved

	No Cloud		With Cloud	
	median	99%	median	99%
SPEECH (500 requests)	1.22 s	6.69 s	0.23 s	1.25 s
FACE (300 requests)	0.42 s	4.12 s	0.16 s	1.47 s

* Amazon EC2 for *SPEECH* and private Cloud for *FACE*.

** Netbook as a mobile device.

Isn’t Offload an old idea?

- Yes. Many papers from 10-20 years ago on mobile offload.
- Assumed nonexistant local infrastructure, nonstandard software frameworks

- Now, we do have public cloud infrastructure
- IaaS, VMs can be used to run any offload framework
- Cloud is not local

Effects of Cloud location

Network measurement

- **Average RTT from 260 global vantage points to an "optimal" Amazon EC2 instance is 73.68 ms [1]**
- [CMU – Amazon East] has amazingly good connection, but becomes similar with Amazon West case if it is measured off-campus

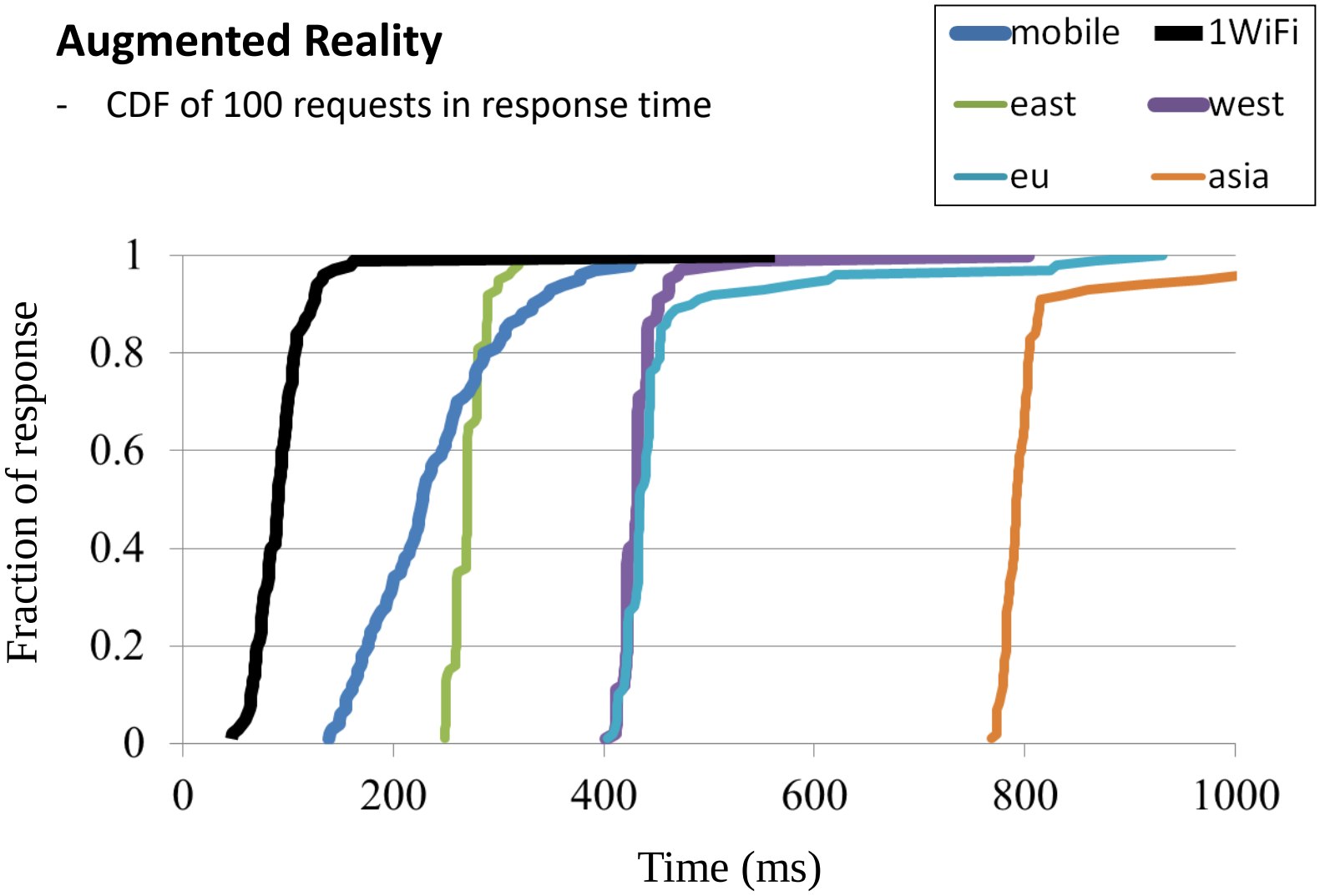
EC2	Throughput (Mbps)		Latency (RTT, ms)
	To	From	median
East	28	34	9.2
West	**12**	**14**	**92**
EU	3.6	0.9	99
Asia	10	0.5	265

< From mobile to Cloud over WiFi >

Impact on response time

Augmented Reality

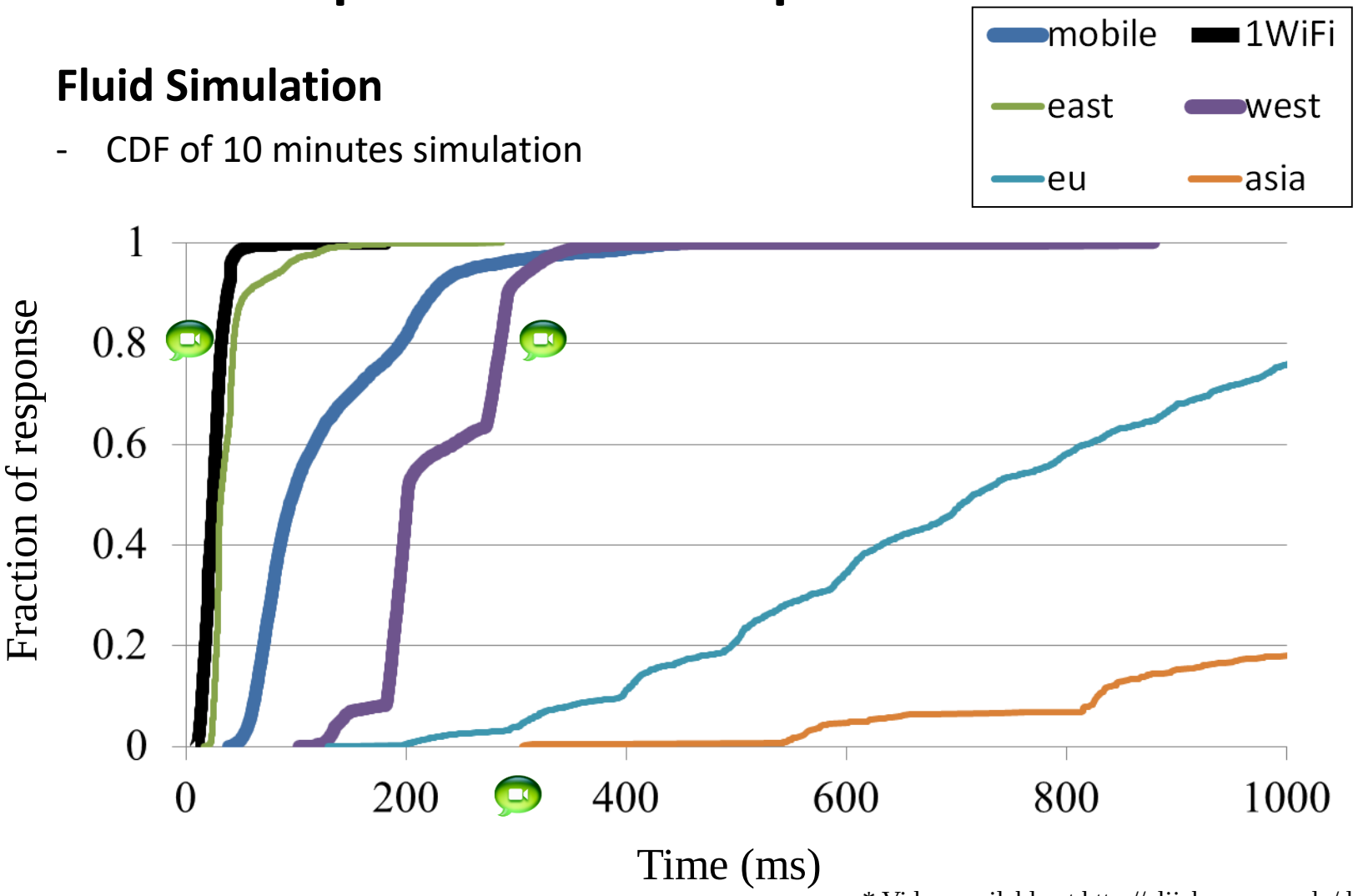
- CDF of 100 requests in response time



Impact on response time

Fluid Simulation

- CDF of 10 minutes simulation

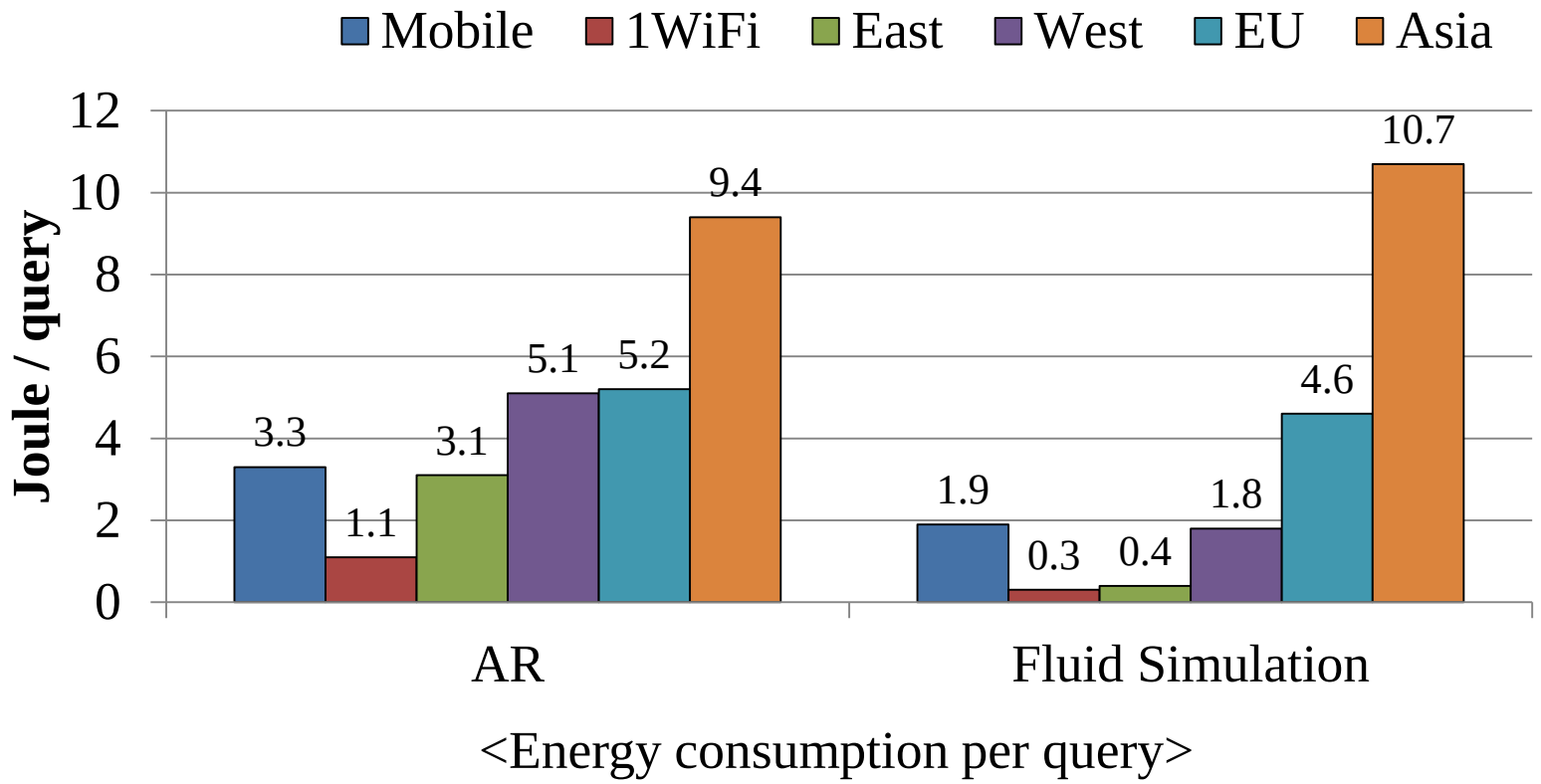


* Video available at http://elijah.cs.cmu.edu/demo.html

Impact on Energy Usage

Energy consumption on mobile device

- Measured during response time experiment



<Energy consumption per query>

* netbook's baseline idle power dissipation is 10W

Cloud location really matters!

- Proximity to the data center is essential
  - Response time & Energy consumption
- Local Cloud infrastructure on LAN / WiFi
  - Best attainable proximity
  - Enables interactive, compute-intensive perception applications on mobile devices
- Contradictory requirements
  - **Local clouds are valuable** for mobile computing
  - However, it needs **many data centers at the edges the Internet**
  - Contradicts consolidation, economies of scale of central DCs

→ How can we reconcile this conflict?

Cloudlet Concept [1]

- Local cloud infrastructure
  - On LAN / WLAN
  - high BW, low latency relative to cloud
- Provides IaaS – like a mini AWS datacenter
- Amenable to decentralized, incremental deployment – like WiFi

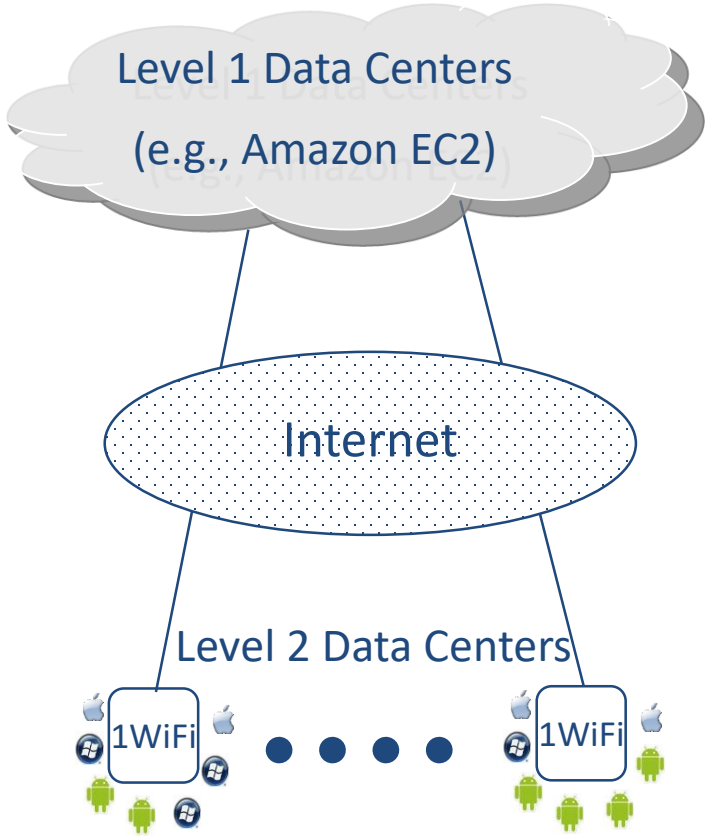
Cloudlet vs. Cloud

	Cloudlet	Cloud
State	Only soft state	Hard and soft state
Management	Self-managed; little to no professional attention	Professionally administered, 24/7 operator
Environment	"Datacenter in a box" at business premises	Machine room with power conditioning and cooling
Ownership	Decentralized ownership by local business	Centralized ownership by Amazon, Yahoo, etc.
Network	LAN latency/ bandwidth	Internet latency/ bandwidth
Sharing	Few users at a time	Hundreds to thousands of users at a time

Alternative Cloud Architecture [2]

Hierarchical organization data centers

- Level 1
  - Today's unmodified Cloud
- Level 2
  - ***Stateless*** data centers at the edge
  - "Cloudlet" Satyanarayanan2009 "1WiFi" Ha2013
  - Appliance-like deployment model
    1. Only soft state
       - Cached virtual machine images
       - Cached files from DFS
    2. No hard state keeps management overhead low.



Alternative Cloud Architecture

Operating Environment

- Common requirements between Levels 1 and 2
  1. Strong **isolation** between untrusted user-level computations
  2. Mechanisms for authentications, access control and metering
  3. **Dynamic resource** allocation
  4. The ability to **support a wide range** of user-level computations

→ **Virtual machine** as Cloud of today like EC2

Alternative Cloud Architecture

Physical realizations

- Hardware technology is already out today
- Repurpose as cloudlet / Level 2 data center by removing hard state and adding self-provisioning

Myoonet's Outdoor micro data center

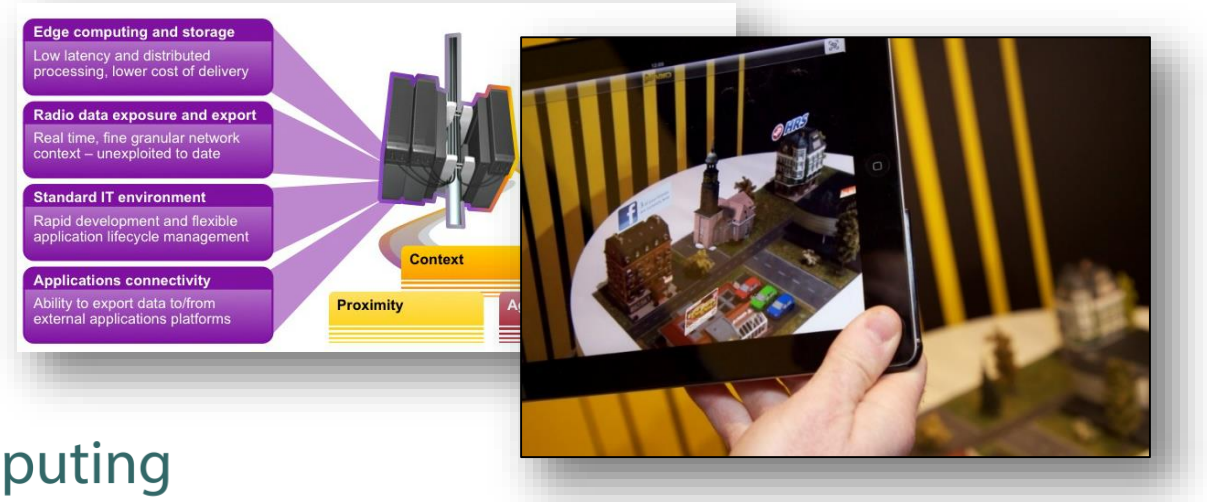
AOL's micro data center

Industrial efforts towards Cloudlets

- IBM, Nokia announced Liquid Applications at MWC 2013
- ETSI Mobile Edge Computing (MEC) standardization for cell-tower-based clouds [now Multi-Access Edge Computing]
- OpenFog consortium announced in Nov. 2015, focusing on IoT
- Open Edge Computing (OEC) -- creating common APIs across diverse deployments
- Amazon AWS Lambda@Edge – run functions on CloudFront CDN nodes
- Google Stadia – streaming games from cloud + edge [announced March 2019]

STADIA

OpenFog



openEDGEcomputing

What makes Edge Computing different from Cloud?

- Need for discovery
  - Cloud services are ubiquitous
  - Edge is best for local use, so need to find the best cloudlet for a particular user
- Cloud vs. Cloudlet provisioning
  - One time cost in Cloud
  - Can’t have all applications running in all cloudlets all the time!
  - Dynamic, on-use provisioning is needed at edge
- User mobility
  - Not critical at granularity of large data center service area
  - May need to migrate services between cloudlets as user moves

Deep Dive: Dynamic VM Synthesis

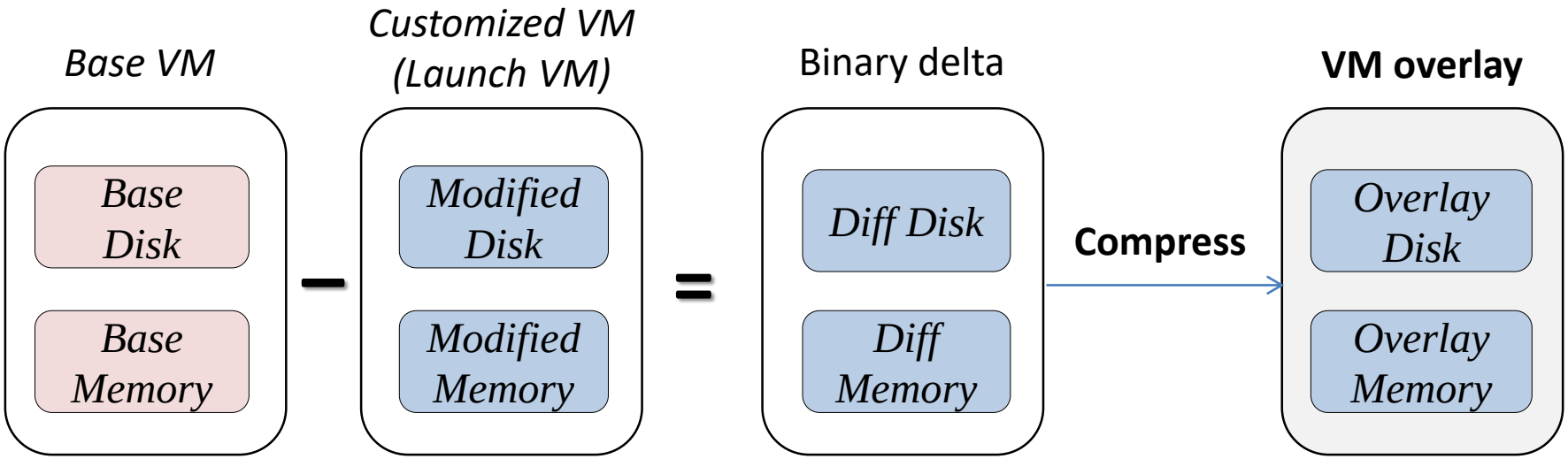
- Transient provisioning of Cloudlets
- Issue: VMs are large – slow transfer, launch

- Observation: most of custom VM is standard OS, libraries
  - Assume a set of standard OS images
  - VM Overlay: compressed diff between custom, standard VM – much smaller
  - Possibility to carry VM Overlay, provision from mobile

VM Synthesis

VM Synthesis: dividing a custom VM into two pieces

1) *Base VM*: Vanilla OS that contains kernel and basic libraries
2) *VM overlay*: A binary patch that contains customized parts



VM Synthesis – Baseline Performance

- Performance measurement with rich, interactive applications
- *Base VM*: *Windows 7* and *Ubuntu 12.04*
  - **8GB** *base disk* and **1GB** *base memory*

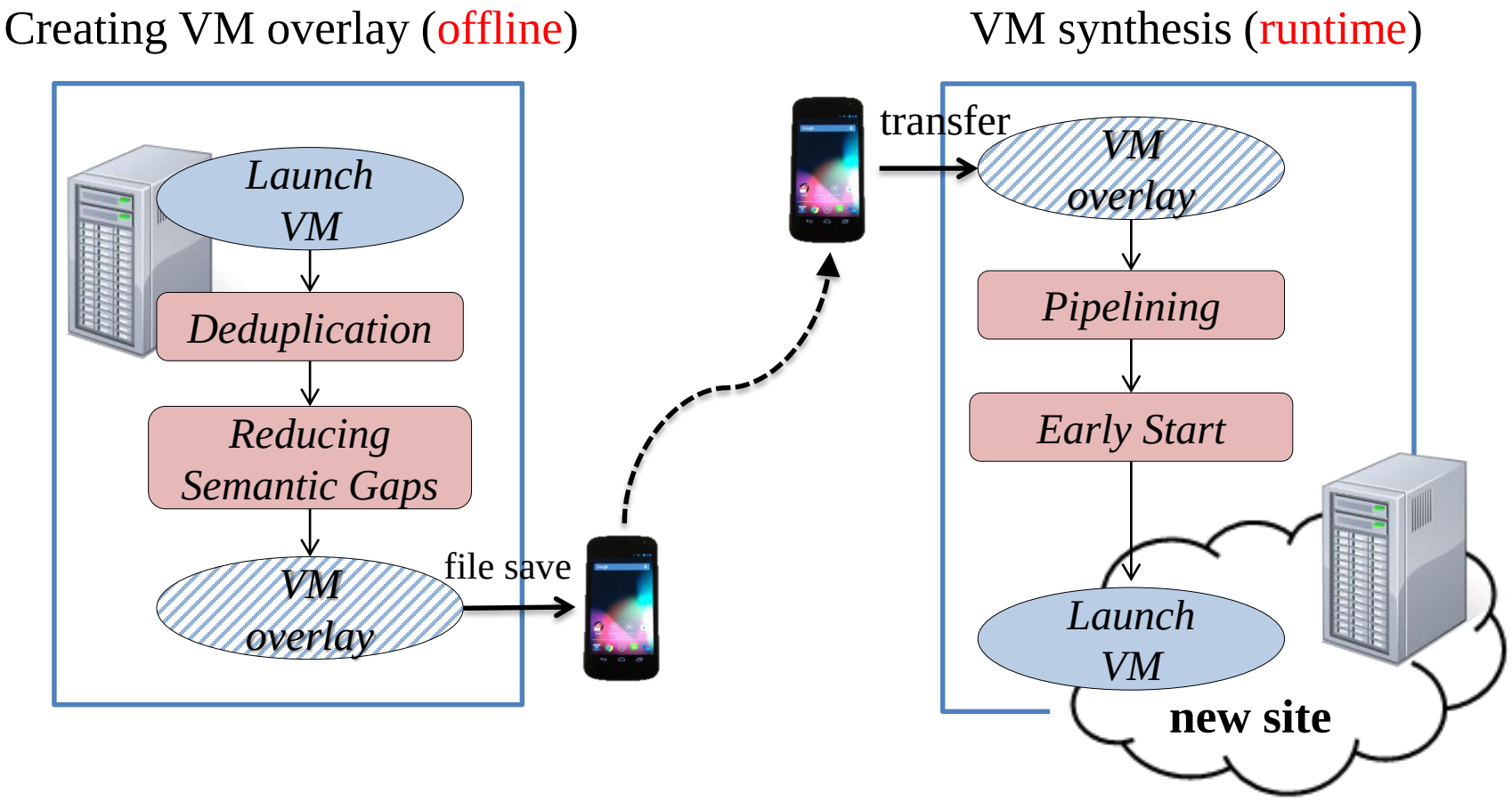
Application	Install size (MB)	Overlay Size		Synthesis time (s)
		Disk (MB)	Memory (MB)	
OBJECT	39.5	92.8	113.3	**62.8**
FACE	8.3	21.8	99.2	**37.0**
SPEECH	64.8	106.2	111.5	**63.0**
AR	97.5	192.3	287.9	**140.2**
FLUID	0.5	1.8	14.1	**7.3**

802.11n WiFi (average 38 Mbps)

→ Still too slow!! Want to get this down to a few seconds

Optimized, Just-in-time VM Provisioning [4]

1. Minimize *VM overlay* size
2. Accelerate VM synthesis



1.1 Deduplication

- Remove redundancy in the VM overlay
- Sources of redundancy
  1) Between *base VM* and *VM overlay*
     - Shared library copied from base disk
     - Loaded executable binary from base disk
  2) Between *VM overlay's memory* and *disk*
     - Page cache, disk I/O buffer
- Compare each block in VM overlay against all other blocks in overlay and base image
- Expensive, but one-time, offline cost

1.2 Reducing Semantic Gap

VM is a black box

- VMM cannot interpret high-level information of memory and disk

Example: Download 100 MB file over network and delete it

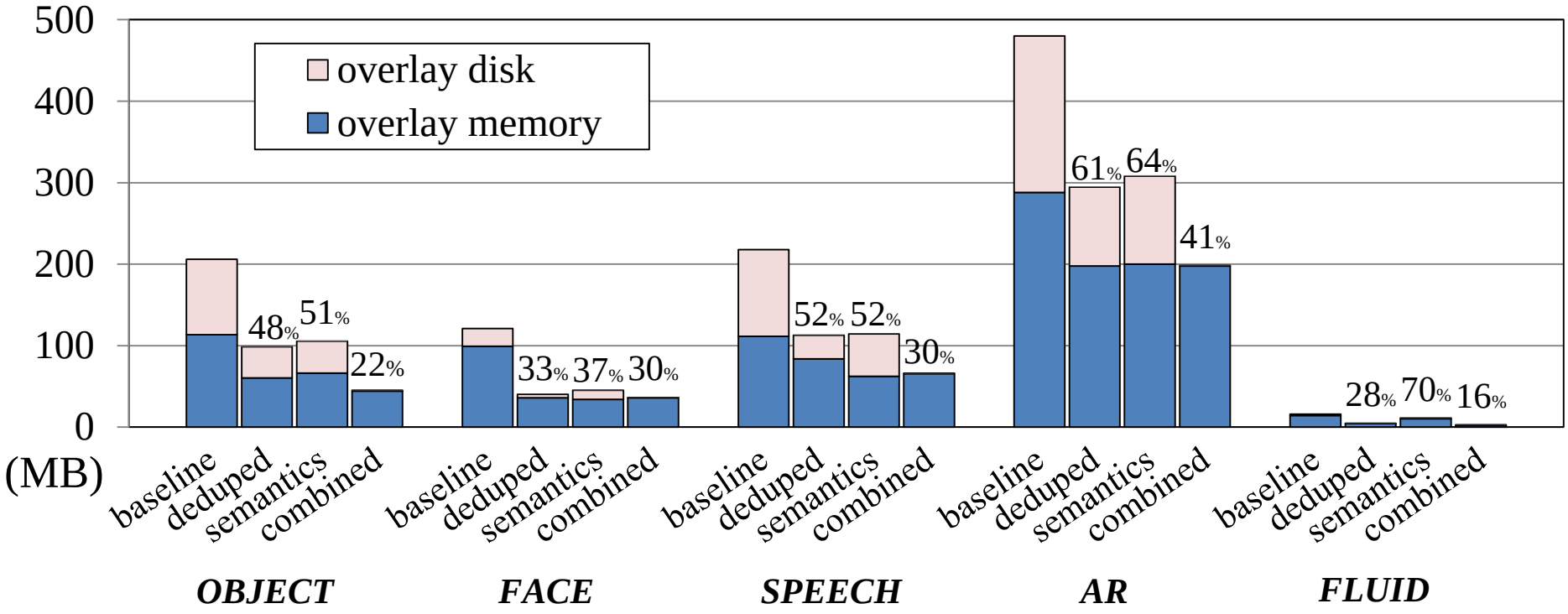
- Ideally, it should result in no increase in VM overlay size
- However, VMM will see **200 MB of modifications**: changed disk blocks + changed memory pages (mainly disk cache)

→ Include only the state that actually matters to the guest OS

How? For disk – use TRIM support!

For memory – inspect free page lists (for open source OS)

VM Overlay Size



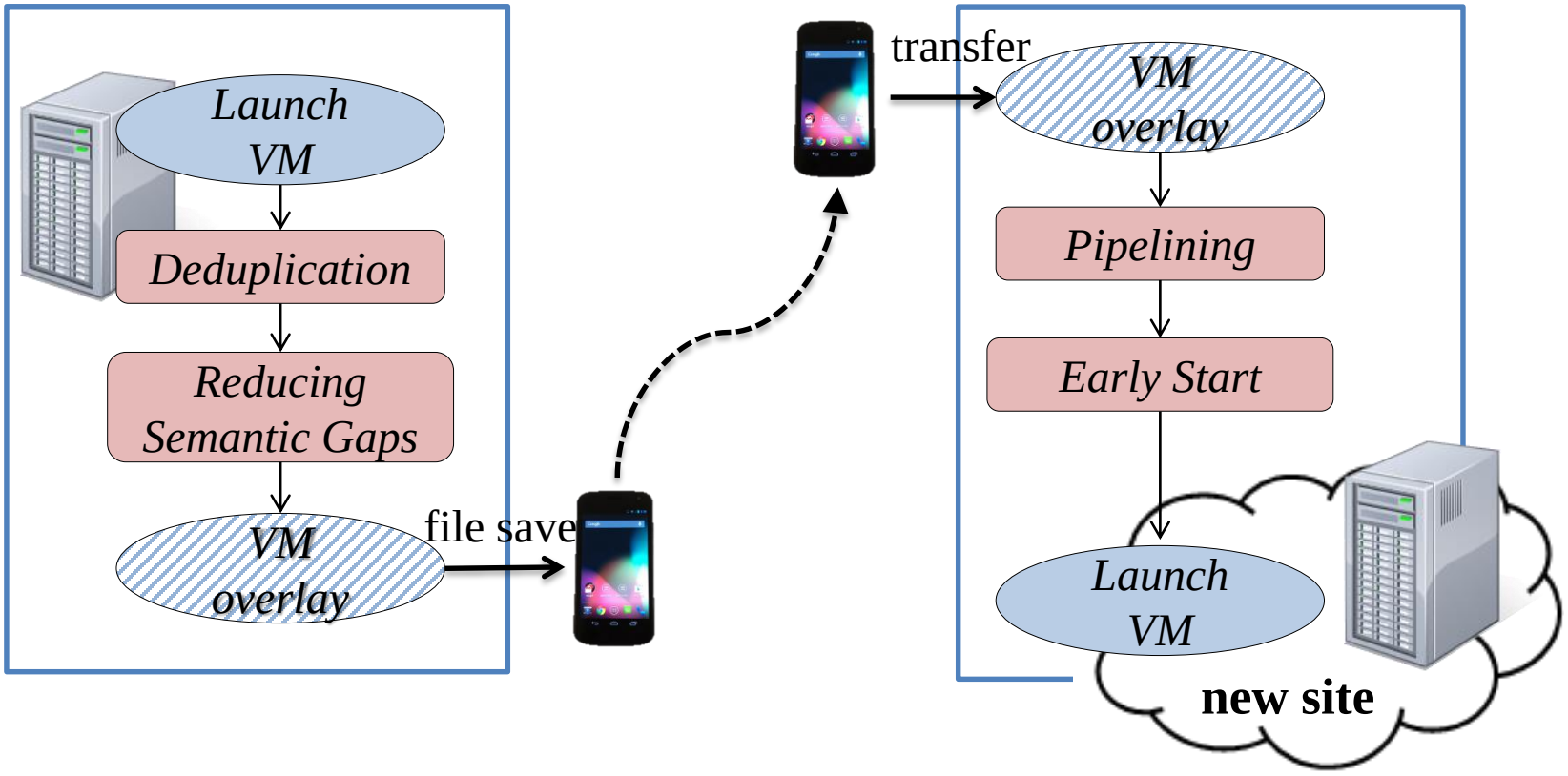
- Deduplication optimization reduces the VM overlay size to 44%
- Using semantic knowledge reduces the VM overlay size to 55%
- Both applied together, overlay size is reduced to **28% of *baseline***

Overview of Optimizations

1. ~~Minimize *VM overlay* size~~
2. **Accelerate VM synthesis**

Creating VM overlay (offline)

VM synthesis (runtime)



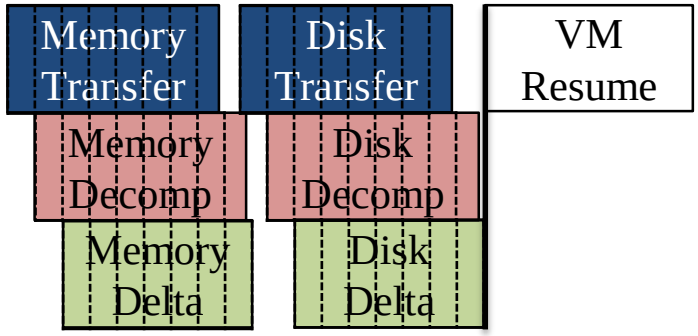
2.1 Pipelining

- Steps for VM synthesis

  ① Transfer VM overlay ② Decompress ③ Apply delta

<Sequential>

<Pipelined>



- Complexities in removing inter-dependencies among segments

2.2 Early Start

Approach

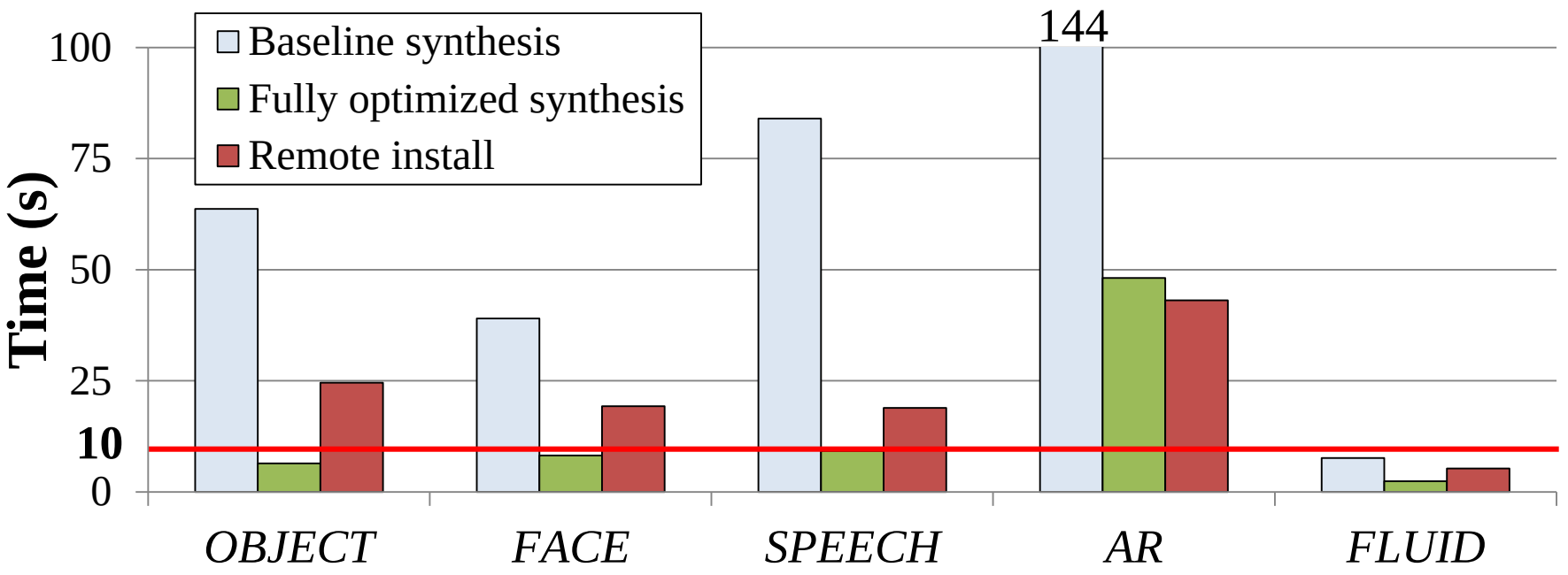
- Start executing VM before synthesis completes!

Implementation

1) **Reorder the chunks in estimated access-order**
2) Break the ordered overlay into **smaller segments for demand fetching**
3) Start the VM and begin streaming the segments in order
4) Allow out-of-order demand fetches to preempt stream

→ Complex run time implementation, with careful monitoring of VMM block accesses

First-response time compared to *baseline*



* Chunks are ordered with segment size of 1 MB

Time between starting VM synthesis and receiving the first offload result

- Faster than remote installation, while maintaining strong guarantees
- Except *AR*, first-response within 10 seconds (up to 8x improvement)

VM Synthesis Conclusions

- Shown to be effective for rapid provisioning of cloudlet resources
- Overcomes one of the key problems with a VM-based offload solution

- Effective way to deploy service near end user

User Mobility and the Edge

- Physical distance ≠ Network distance!

Live VM Migration in Data Centers

- "Downtime" is the ultimate performance metric
  - This is the duration which the VM instance is suspended
  - Ideally, VM migration should not cause any service interruption
- Conventional live migration is optimized for datacenter environment
  - The source and destination share storage; only the memory state migrated
  - Migration is performed over fast, low-latency LAN
- Iterative state transfer
  - Identify, transfer modified blocks as VM runs
  - Repeat until delta is small
  - Briefly stop VM, transfer remaining delta, resume at desitnation

- Problem: Often works poorly across WANs!

VM Handoff for the WAN [5]

- Conceptually similar to live migration
- However, focus more on total migration time
  - Users will see degraded performance until done
  - Resources used at source and destination
- Aggressive use of deduplication, compression to handle the WAN BW bottleneck
- Need to adapt due to computational bottlenecks
- Stopping condition: tradeoff small increase in downtime for large wins in total time

Results

- Comparison with QEMU/KVM
  - Network: 10Mbps, CPU: 1 CPU
  - Assumes VM's disk exists at destination for KVM

	Method	Handoff Time	VM downtime
OBJECT (Linux)	VM Handoff	**1 min**	4.5 s
	KVM live migration	**12 min**	1.5 s
MAR (Windows)	VM Handoff	**4 min**	11 s
	KVM live migration	**166 min**	5 s

- At least on order of magnitude speedup in handoff time
- A couple of seconds more downtime

What about Containers?

- Containers are glorified processes
  - Private file system, with custom environment
  - Shared kernel, drivers with host environment
  - Some resource isolation through cgroups, namespaces in Linux kernel
- Docker:
  - Adds layer to efficiently store, distribute, and deploy containers
  - Uses layered filesystem model (a bit like overlays)

Containers

- Pros:
  - Relatively small compared to VMs – easier to transfer, provision
  - No booting – just run the process
- Cons:
  - Not as general (OS, kernel dependency)
  - Less isolation than VMs
  - Migration is hard – process state within kernel, etc.

Containers are not always faster

- Migration (Docker+CRIU, checkpoint, restore)

VM	Approach	Total time	Down time	Transfer size
App1	VM handoff	15.7s	2.5s	7.0 MB
	Docker	6.9s	6.9s	6.5 MB
App4	VM handoff	58.6s	5.5s	61 MB
	Docker	118s	118s	98 MB

10 Mbps network

Bandwidth Issues

- Discussion so far mostly about latency
- However, WAN BW to remote datacenter can be limiting factor

- Can Edge Computing help with BW issues?

Live Video as a Service

- One focus of Intel Science and Technology (ISTC) on Visual Cloud Systems here at CMU
- Potential value of live video analysis

Liquid Spill!
Lost Child!
Icy Patches!
Checkout line too long!
Unattractive Window Display

Upload bandwidth

Pittsburgh: 300k users

x 2.5 Mbps (480p) = 750 Gbps

x 8.5 Mbps (1080p) = 2550 Gbps

recommended YouTube upload rates

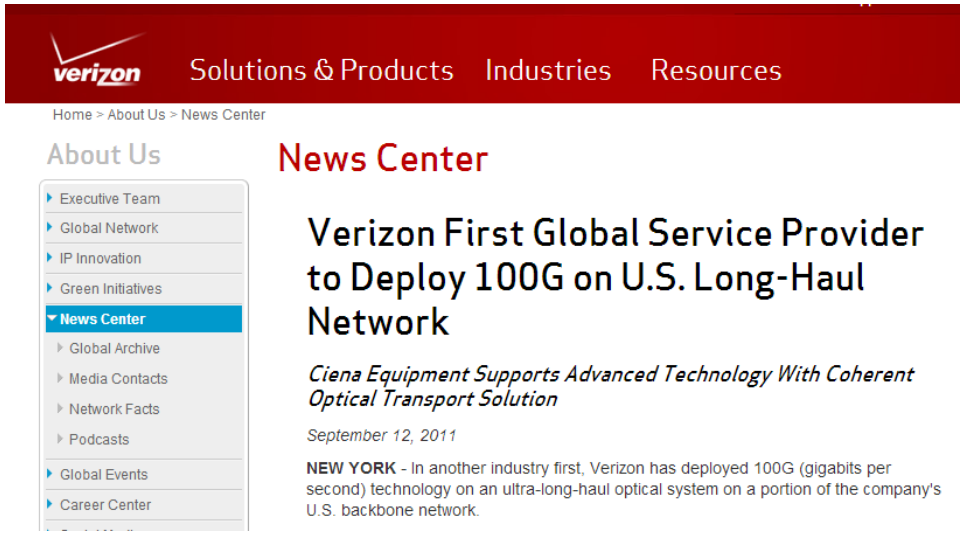
YouTube Upload Rate:

300 hours of video / minute

= 45 Gbps

= 18000 std. def. uploads

Verizon Press Release, Dec 2011



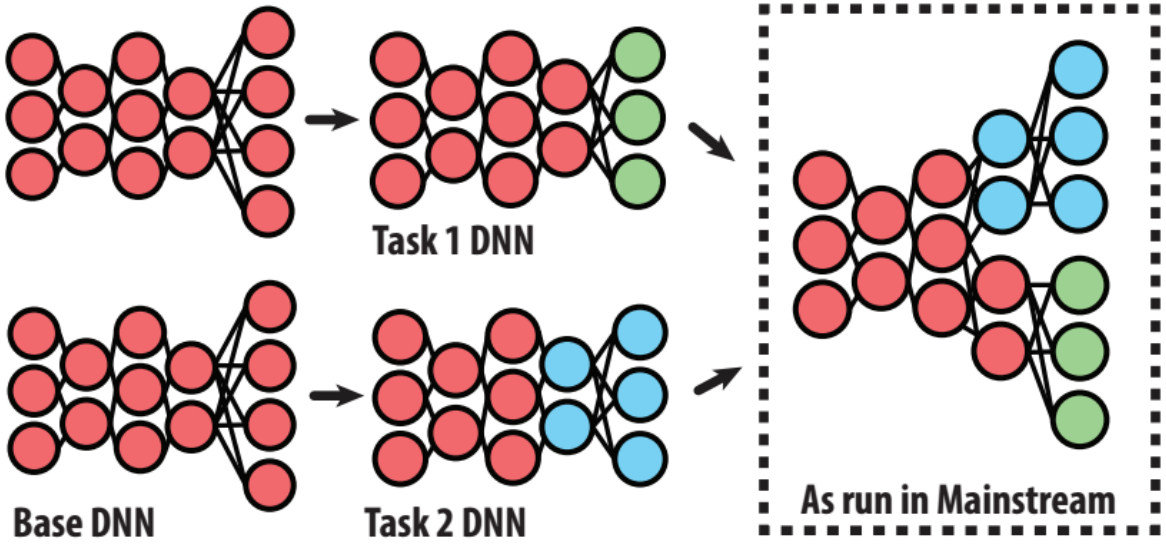
One solution: Edge computing!

- "Streamer" project
- Shared edge computing resources near cameras

- Process live frames
  - E.g., object detection, activity recognition
  - Focus on DNN, computer vision algorithms
  - Store at edge, forward only as needed to cloud

Getting the most out of the Edge

- Compute resources limited at Edge
- Video analytics, DNNs are expensive
- “Mainstream” project – exploit common practice of fine-tuning pretrained DNNs
- Share compute when DNNs use same base model, leave some layers unmodified
- Build training, runtime system to maximize sharing, subject to application accuracy



Getting the most out of the Edge

- “FilterForward” project
- Perform expensive analysis in Cloud
- Edge processing – filter out all but most relevant frames for particular task
  - First find generic “interesting” frames, using DNN
  - Use intermediate DNN activations as features
  - Train cheap, task-specific interest detectors
  - Research Question: how many parallel tasks can we handle? 10? 100? 1000?

What we have covered

- Aren't mobiles getting more capable? Why do we need offload?
  - Yes, but persistent gap in mobile performance, limited energy
- Isn't offload an old idea?
  - Yes, but it was an idea ahead of its time – nonexistant local infrastructure, nonstandard frameworks; now we have cloud model
- Can't we just use Amazon or other conventional cloud?
  - No, distance to datacenter matters for latency-sensitive apps
- How can Clouds better support these new mobile applications?
  - Stateless cloudlets distributed at edge of network, act as second tier in hierarchical cloud architecture
- How does Edge computing differ from Cloud?
  - Need for discover, dynamic provisioning, migration to handle mobility
- Do all apps need to run everywhere, all the time?
  - No, not possible; instead, dynamically provision cloudlets as needed
- VMs are big and slow, what can we do about this?
  - Highly optimized VM synthesis, handoff can greatly reduce provisioning, migration times
- Can containers help?
  - Maybe. Generally smaller, faster to deploy than VMs, but less general, and not always faster (e.g., for migration)
- Are cloudlets for latency only? What about BW?
  - No, they can help with BW; e.g., live analytics at edge. Ongoing research at ISTC-VCS.

Open Issues

- Can edge computing be cost effective?
- Business model for cloudlets
  - Centralized or distributed model of deployment
  - Chicken and egg problem: apps and infrastructure
- How about apps that have a Big Data component?
- Trust and security

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, "**The Case for VM-Based Cloudlets in Mobile Computing**," Pervasive Computing, October, 2009.

[2] K. Ha, P. Pillai, G. Lewis, S. Simanta, S. Clinch, N. Davies, M. Satyanarayanan, "**The Impact of Mobile Multimedia Applications on Data Center Consolidation**," IEEE International Conference on Cloud Engineering, 2013.

[3] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. W, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, D. Siewiorek, M. Satyanarayanan, "**An Empirical Study of Latency in an Emerging Class of Edge Computing Applications for Wearable Cognitive Assistance**," ACM/IEEE Symposium on Edge Computing, 2017.

[4] K. Ha, P. Pillai, W. Richter, Y. Abe, M. Satyanarayanan, "**Just-in-Time Provisioning for Cyber Foraging**," ACM MobiSys 2013.

[5] K. Ha, Y. Abe, T. Eiszler, Z. Chen, W. Hu, B. Amos, R. Upadhyaya, P. Pillai, M. Satyanarayanan, "**You Can Teach Elephants to Dance: Agile VM Handoff for Edge Computing**," ACM/IEEE Symposium on Edge Computing, 2017.