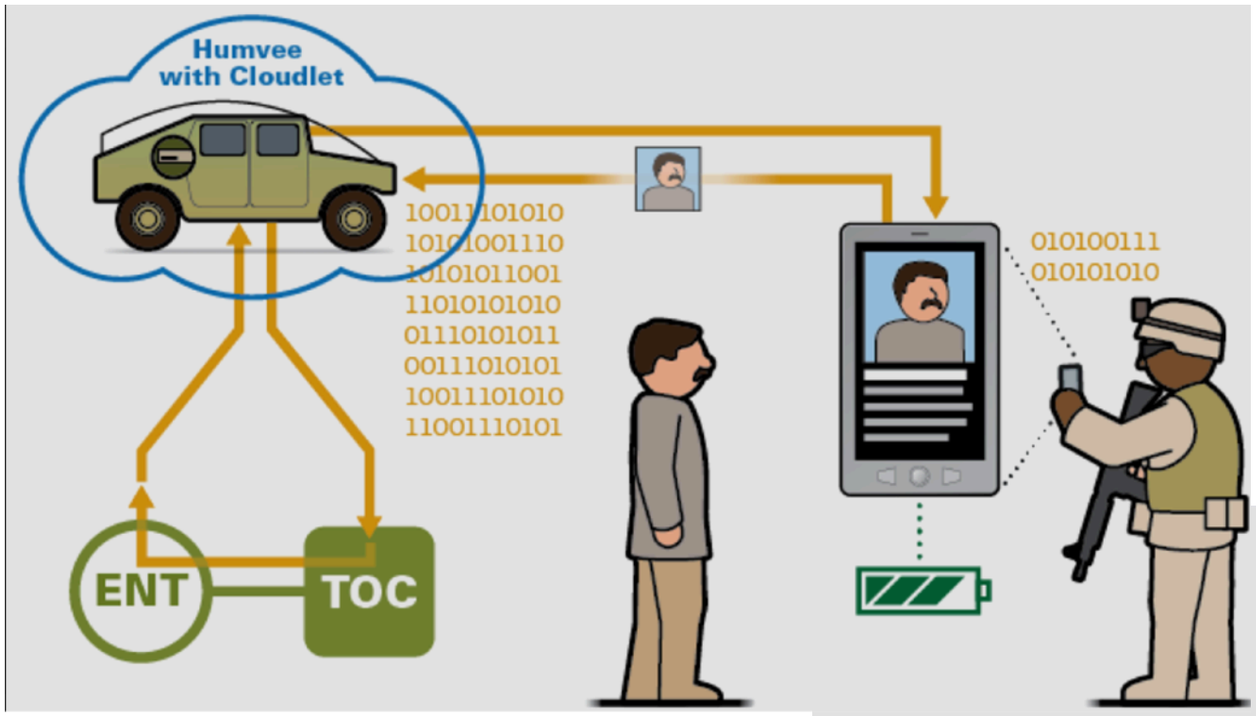
Agenda:

1. **Introduction: Mobile Edge Computing and Cloudlets**
2. Problems of VM-based Cloudlets
3. Docker-based Cloudlets
4. Challenges of Docker-based Cloudlets
5. Solutions
6. Evaluation
7. Immutability and Micro services.
8. Future works

1.Introduction: Mobile Edge Computing and Cloudlets



Work scenario:
-tasks for mobile devices.
-lack of rich resources.

Why need MEC?
WAN latency

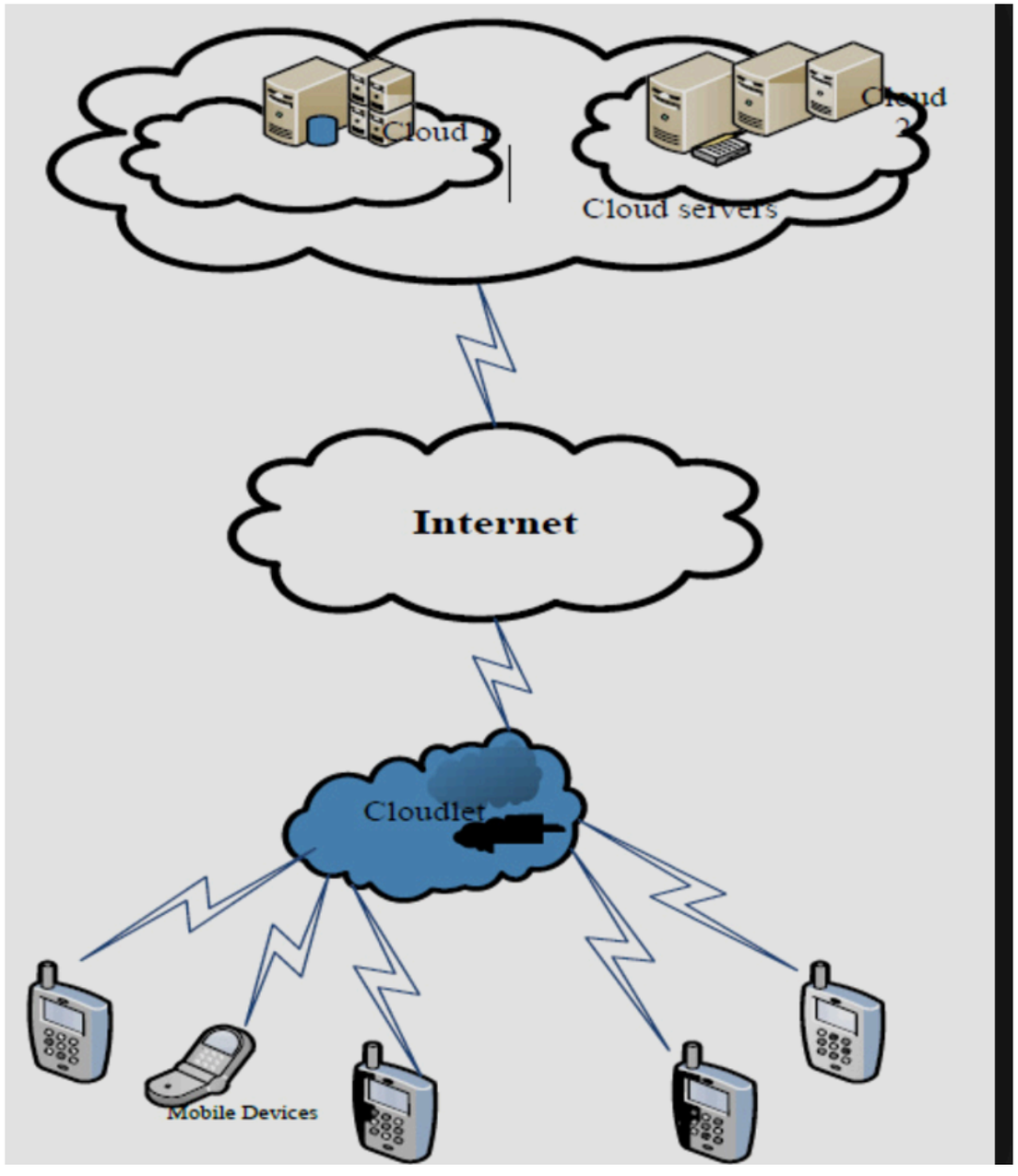
	Min	Mean	Max	Lower bound
Berkeley–Canberra	174.0	174.7	176.0	79.9
Berkeley–New York	85.0	85.0	85.0	27.4
Berkeley–Trondheim	197.0	197.0	197.0	55.6
Pittsburgh–Ottawa	44.0	44.1	62.0	4.3
Pittsburgh–Hong Kong	217.0	223.1	393.0	85.9
Pittsburgh–Dublin	115.0	115.7	116.0	42.0
Pittsburgh–Seattle	83.0	83.9	84.0	22.9

1.Introduction: Cloudlet

Cloudlet:

- a platform of MEC
- creates “server” for users.

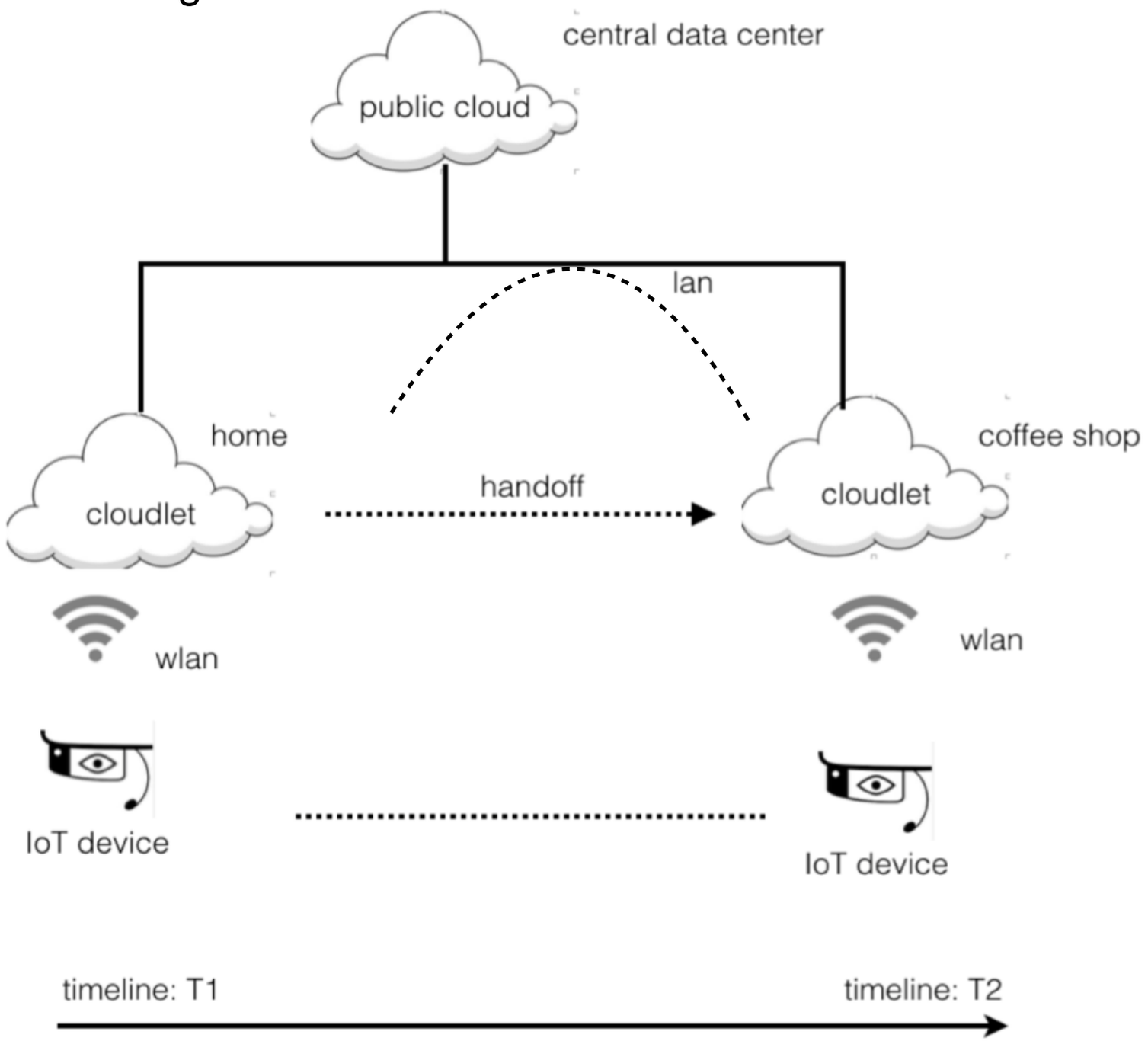
Traditional



1.Introduction: Cloudlet

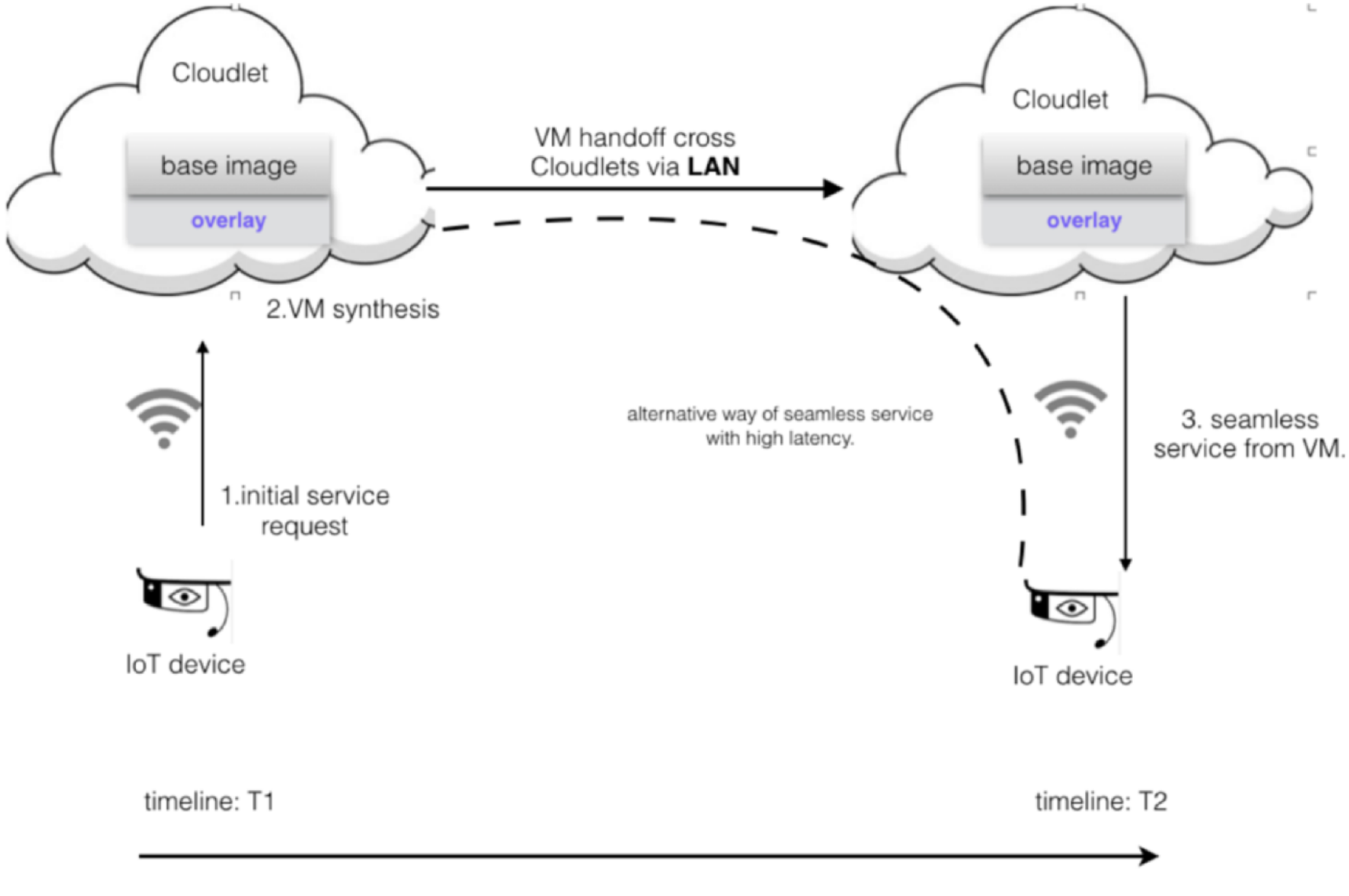
Cloudlet:

- provide seamless service during handoff
- migration with user



1.Cloudlet Implement: Virtual Machine

VM-based Cloudlets: http://elijah.cs.cmu.edu/
base image: ubuntu
overlay: applications



Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. **Problems of VM-based Cloudlets**
3. Docker-based Cloudlets
4. Challenges of Docker-based Cloudlets
5. Solutions
6. Evaluation
7. Immutability and Micro services.
8. Future works

2.Problems of VM-based Cloudlets

Image size is too large.

Image the usually size for virtual machine?

4GB?8GB?

Heavy task for Virtual Machine. OS level migration

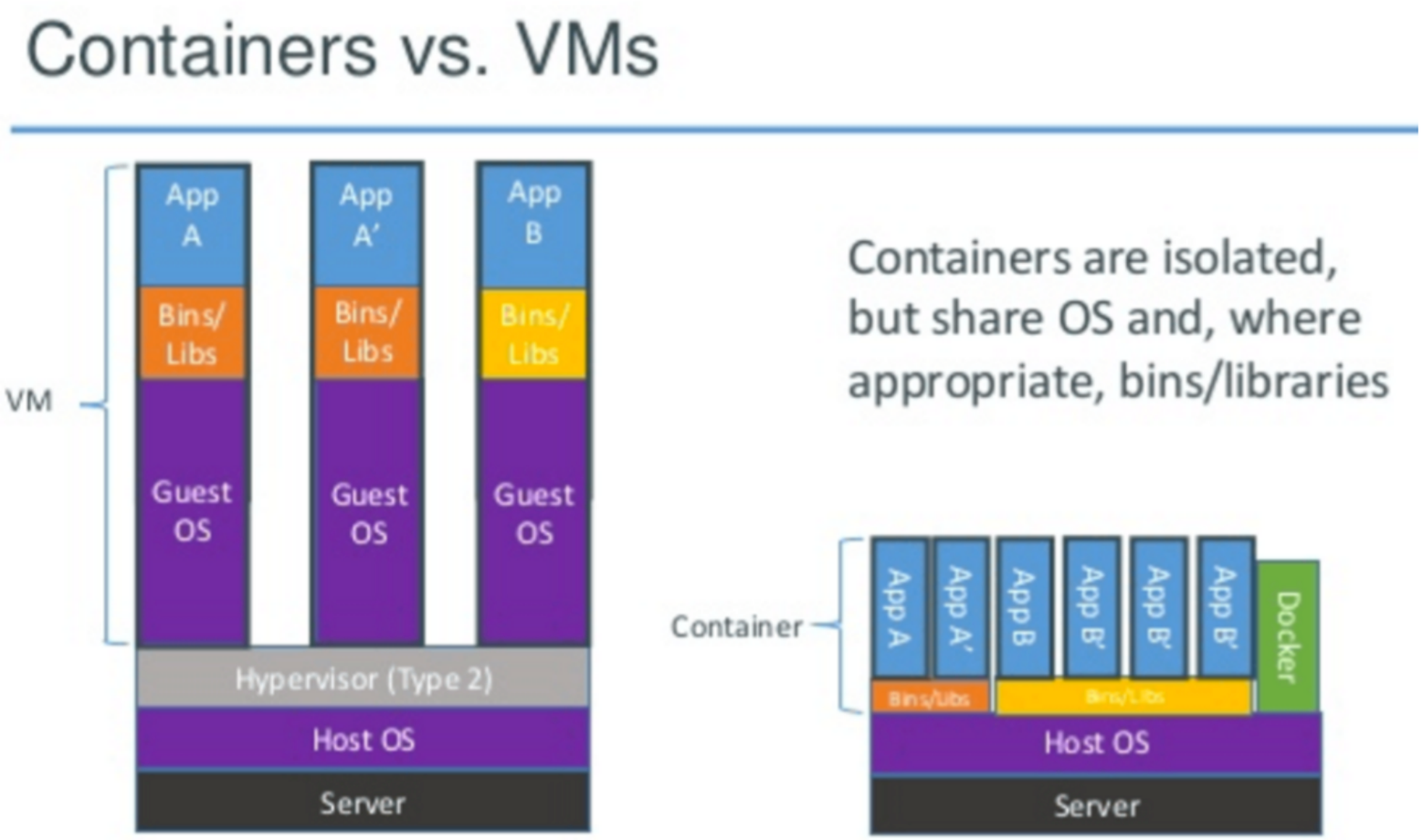
“stateful” : OS, disk, data, FS

2.Problems of VM-based Cloudlets

Replace VM? light, fit for cloud.

Container

- Lightweight VM
  Remove Duplicated OS.
- Not just Lightweight VM
  stateless
  separate data and computation



Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. Problems of VM-based Cloudlets
3. **Docker-based Cloudlets**
4. Challenges of Docker-based Cloudlets
5. Solutions
6. Evaluation
7. Immutability and Micro services.
8. Future works

3.Docker based Cloudlet

Concepts clarify:

Container: isolation technology based on Linux kernel Cgroups and Namespace.

Docker: a widely used implement of container technology.
Docker is the most popular way.

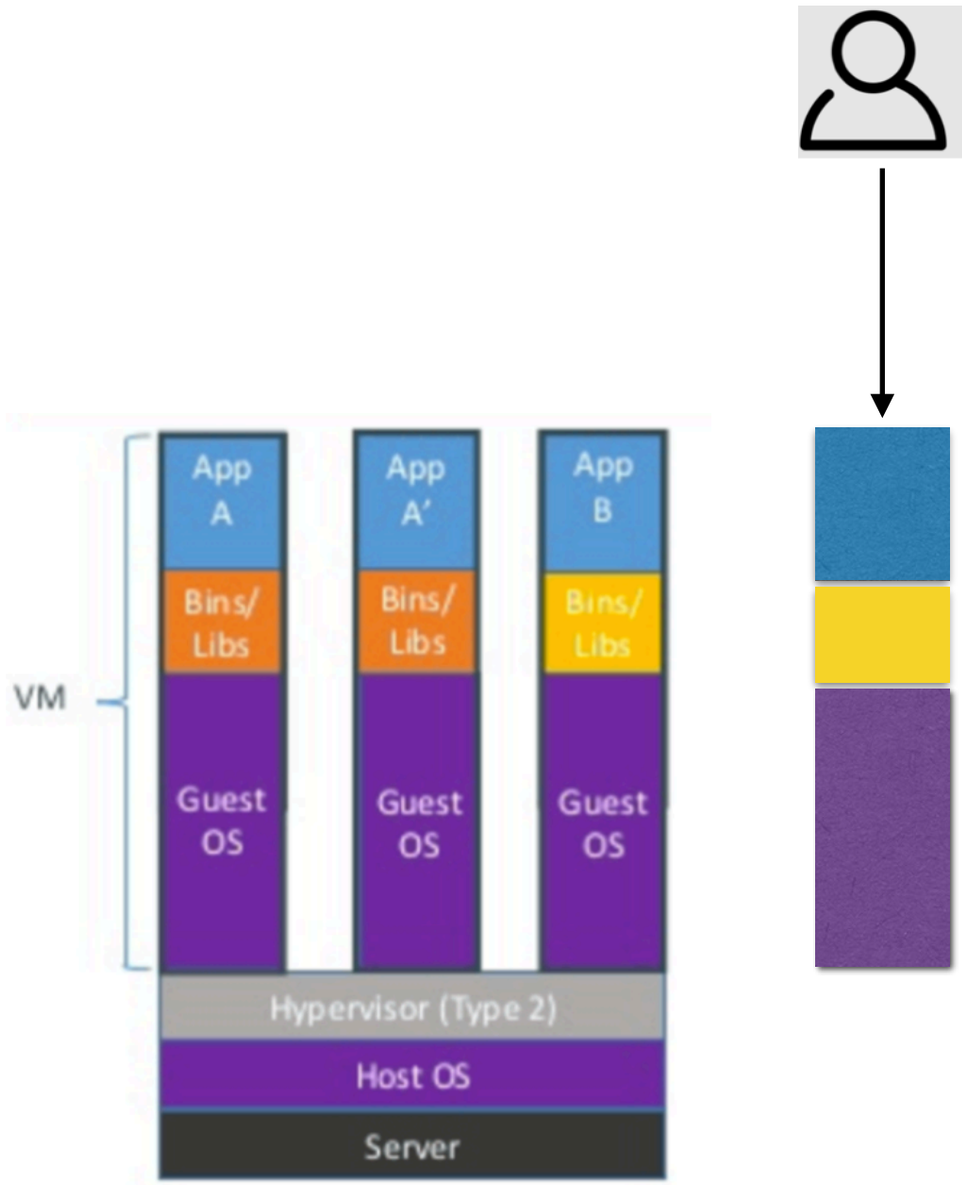
Docker Image: a snapshot of a container.
To be compared, images is like VM image.

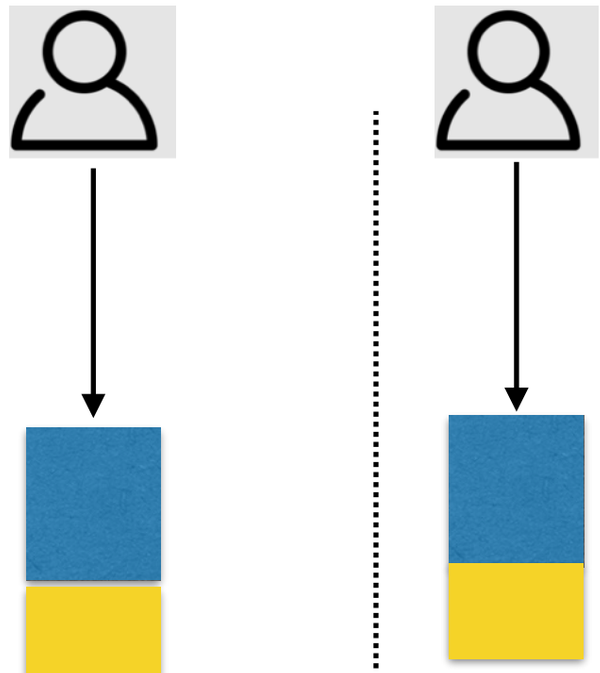
A Container: a running container based on an appoint base image.

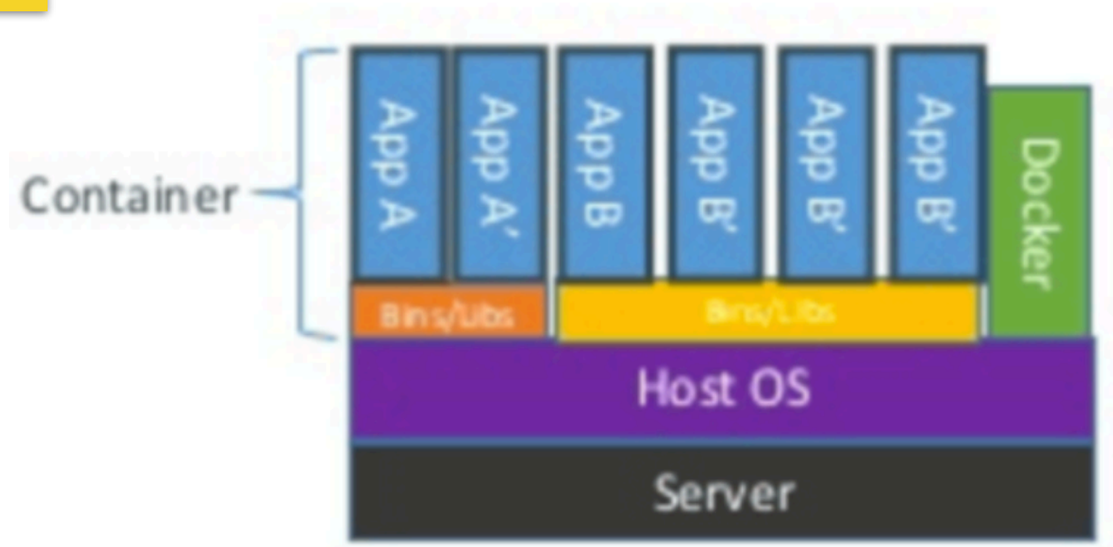
Overlay: delta part between base image and application image.

3. Docker based Cloudlet

First time request / Handoff





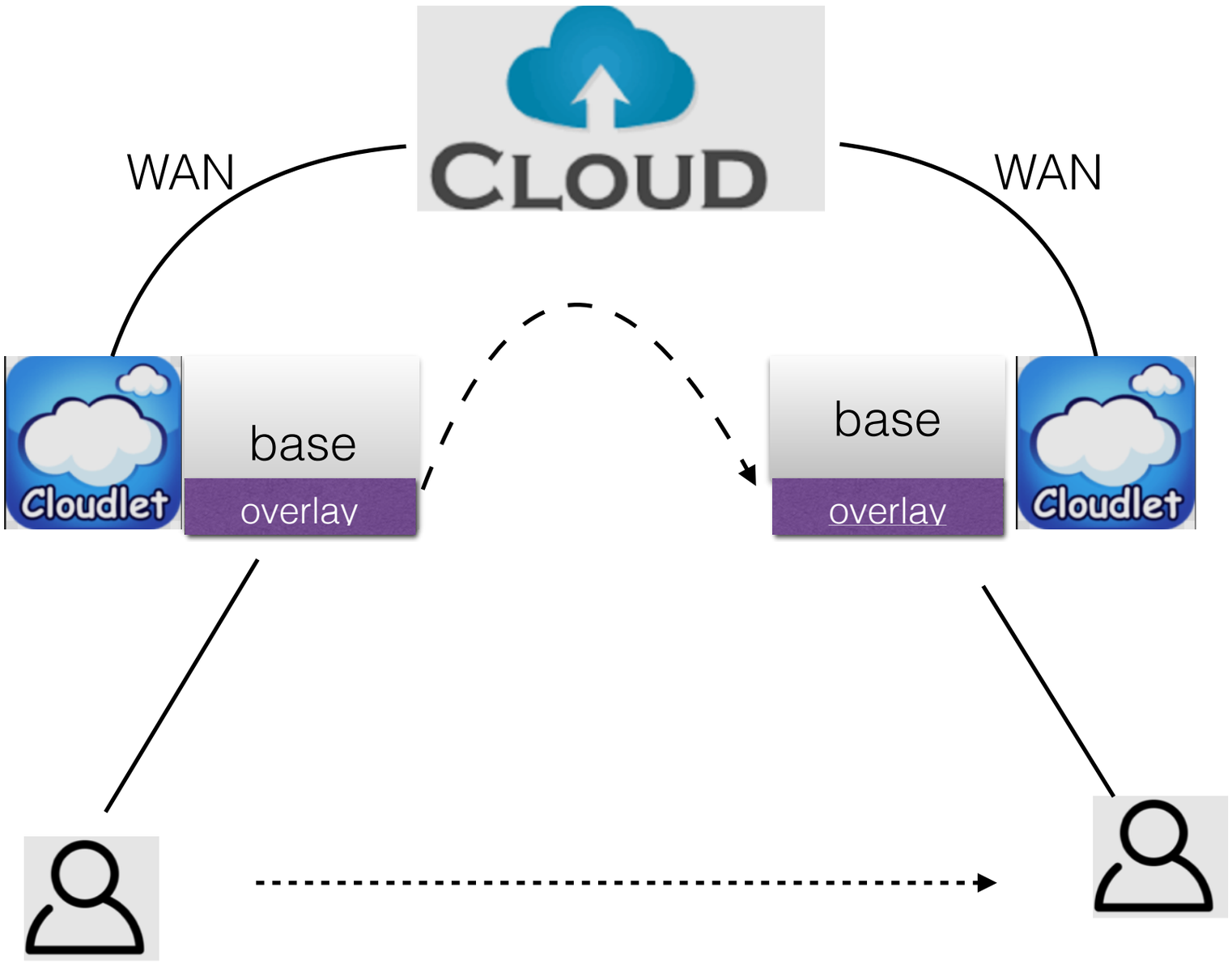


Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. Problems of VM-based Cloudlets
3. Docker-based Cloudlets
4. **Challenges of Docker-based Cloudlets**
5. Solutions
6. Evaluation
7. Immutability and Micro services.
8. Future works

4. Challenges

Images and Migration



4. Challenges

- Manage the base image
- Migrated the changed files in a container
- Seamless network connection
- Live migration: minimal total time and “down” time.

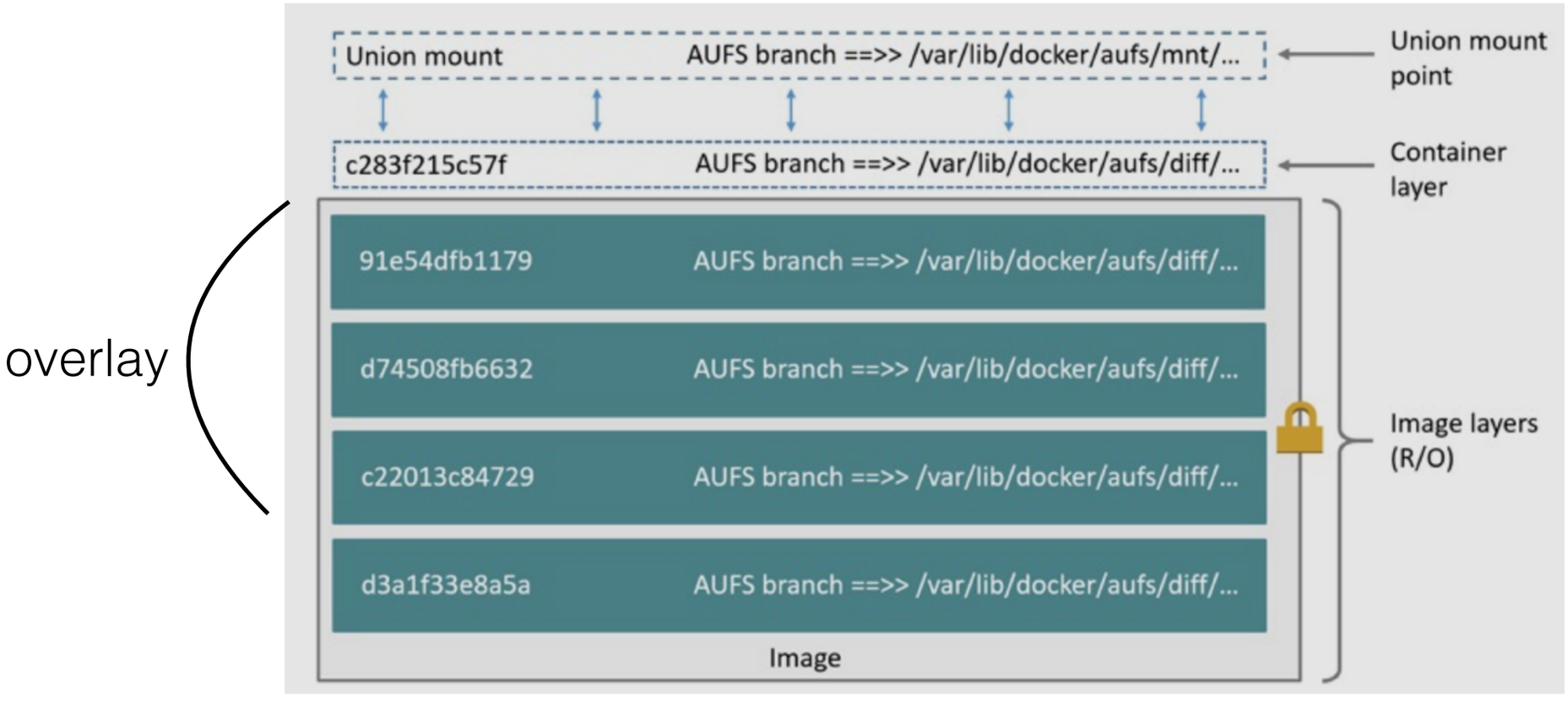
Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. Problems of VM-based Cloudlets
3. Docker-based Cloudlets
4. Challenges of Docker-based Cloudlets
5. **Solutions**
6. Evaluation
7. Immutability and Micro services.
8. Future works

5. Solutions

AUFS: advanced multi-layered unification filesystem

Keep the packages images as base image and application image extract the R/W layer for changed file.



Our case: directly extract container layer.

/var/lib/docker/aufs/diff/

Note: From Docker 1.10, the uuid is not longer direct mapping to the directory.

5. Solutions: CRIU

C/R: checkpoint, transfer and restore.

CRIU: checkpoint/restore in userspace.

a project to implement checkpoint/restore functionality for Linux in users pace

The information we need to dump.

- Filesystem: container R/W layer
- Memory: track process memory information
- Network: tcp connection, socket

5. Solutions: CRIU and Docker

- external: treat container as normal process

  !Docker daemon doesn't know that the container is running

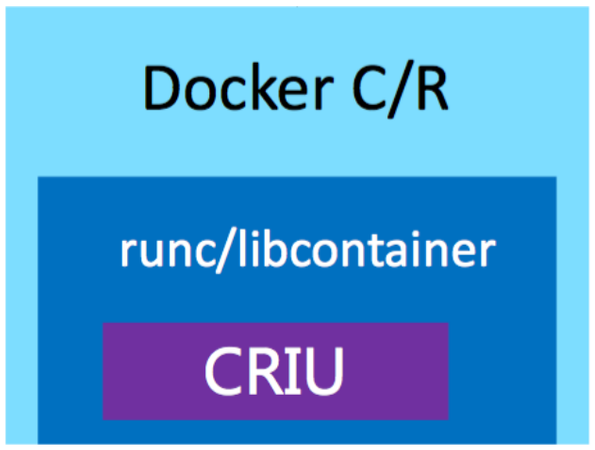
  process trees do not belong to docker daemon.

- native: integrate with Docker library

  docker checkpoint

  docker restore

Docker native way:



Note: runC and containerd are developing very fast in the community .
In the future , the sequence could be”contianerd -> runC —> CRIU”.

5. Solutions: CRIU and pre-dump and iterative dump

Current situation:

criu pre-dump *<pre_img>*

docker checkpoint *<whole mm img>*

pre-dump option should be:

docker -> libcontainer -> CRIU

solution:

1. xdelta3: generate the delta part

   time depends on base size: [40,50], [200,220]

2.hijacked CRIU code: docker checkpoint <delta mm img>

code: https://github.com/hixichen/criu_xichen_modify/tree/hixichen-patch

5. Solutions:Track dirty pages and iterative dump

method: soft-dirty bit in the pagemap
kernel patch: mm: Ability to monitor task memory changes (v3)

Then read the /proc/$pid/pagemap2 and check the soft-dirty bit reported there (the 55'th one). If set, the respective PTE was written to since last call to clear refs.

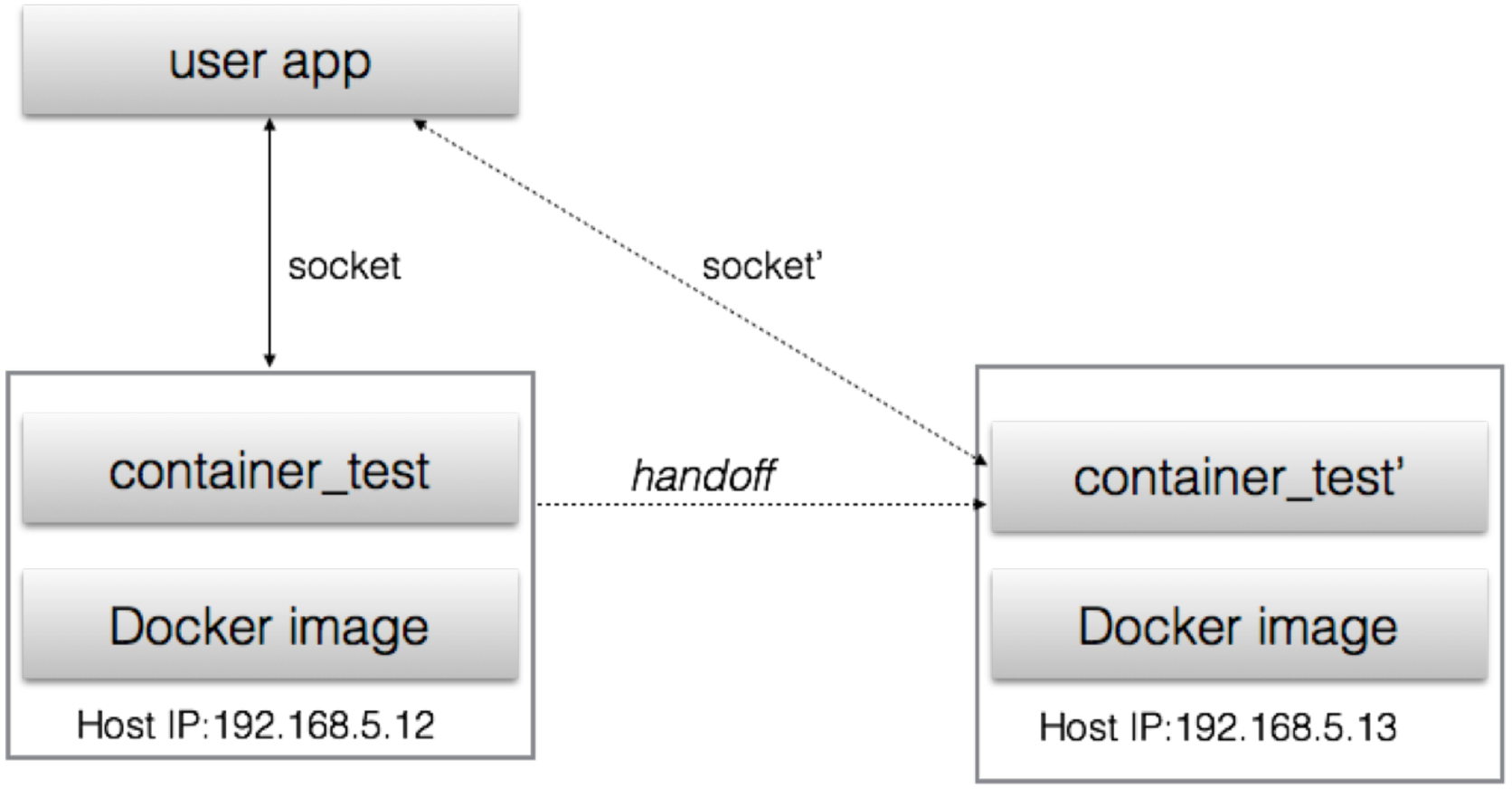
Example:
First dump: 166M
Second: 37M

```
root@vm1:/var/lib/docker/tmp/1pjrps# du -h *
52K     1pjrps-fs.tar.gz
15M     1pjrps-mm.tar.gz
58M     1pjrps-pre.tar.gz
37M     mm
166M    pre
```

5. Solutions: Network migration

Keep the tcp connection between user app and container (server).

Dump socket information.



Technical Challenges:

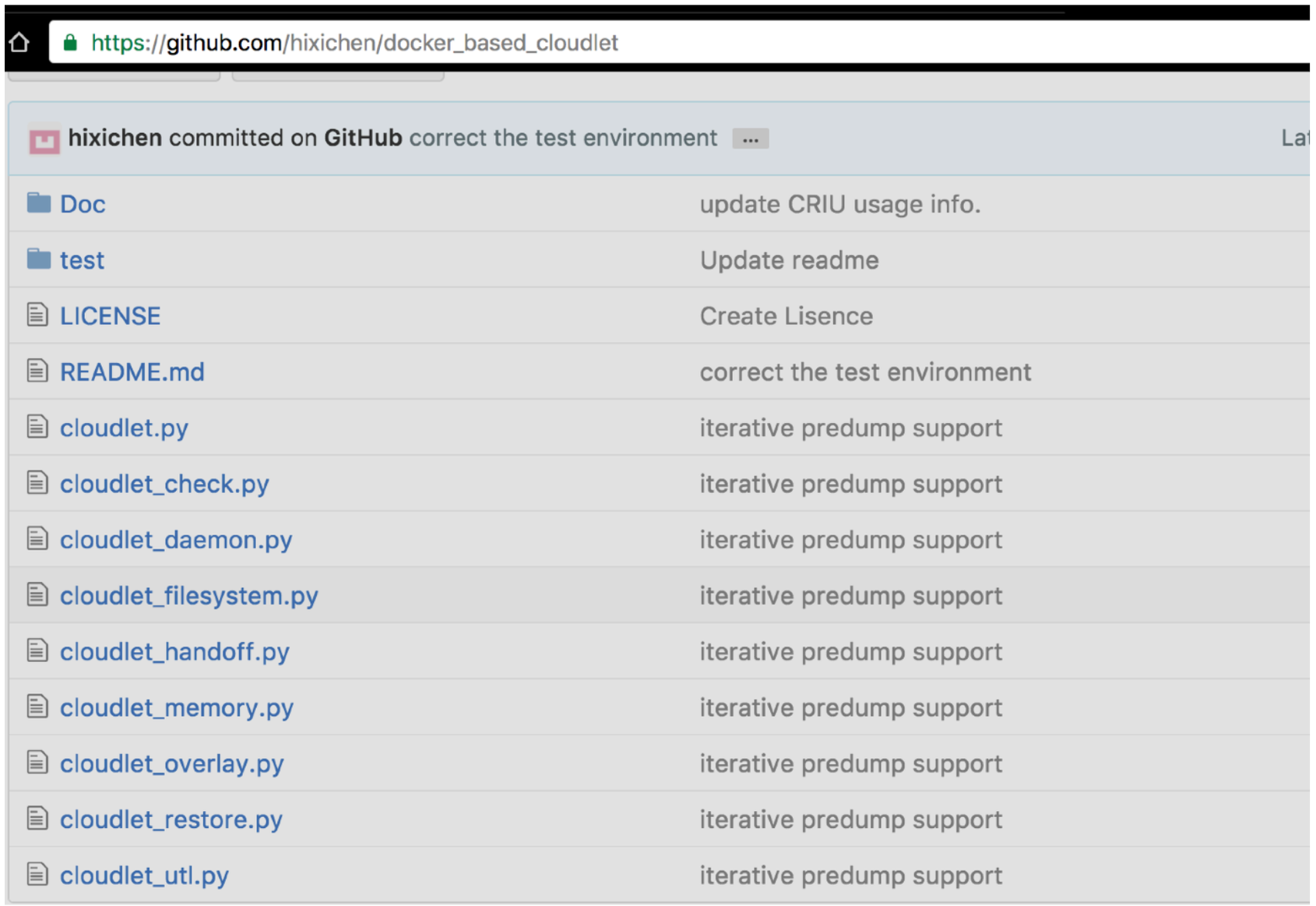
- Namespace
- Mountpoint
- Iptables: drop unused packages
- Network Mode: default bridge, host mode.

My Code

- the total platform to Docker-based Cloudlets
- CRIU code patch for pre-dump

Language: python, C

https://github.com/hixichen/docker_based_cloudlet



Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. Problems of VM-based Cloudlets
3. Docker-based Cloudlets
4. Challenges of Docker-based Cloudlets
5. Solutions
6. **Evaluation**
7. Immutability and Micro services.
8. Future works

6. Evaluation: Image synthesis

Overlay:

	base image	base image size	application image	size different	overlay size(.tar.gz)
moped	ubuntu:precise	103.6MB	302.6MB	199MB	**69MB**
graphics	ubuntu:lastest	129MB	131.1MB	2.1MB	**0.69MB**

Synthesis:

	uncompress time	load time	synthesis time
moped	9s~10s	8-9s	**~18s**
graphics	~2.5s	~1.5s	**~4s**

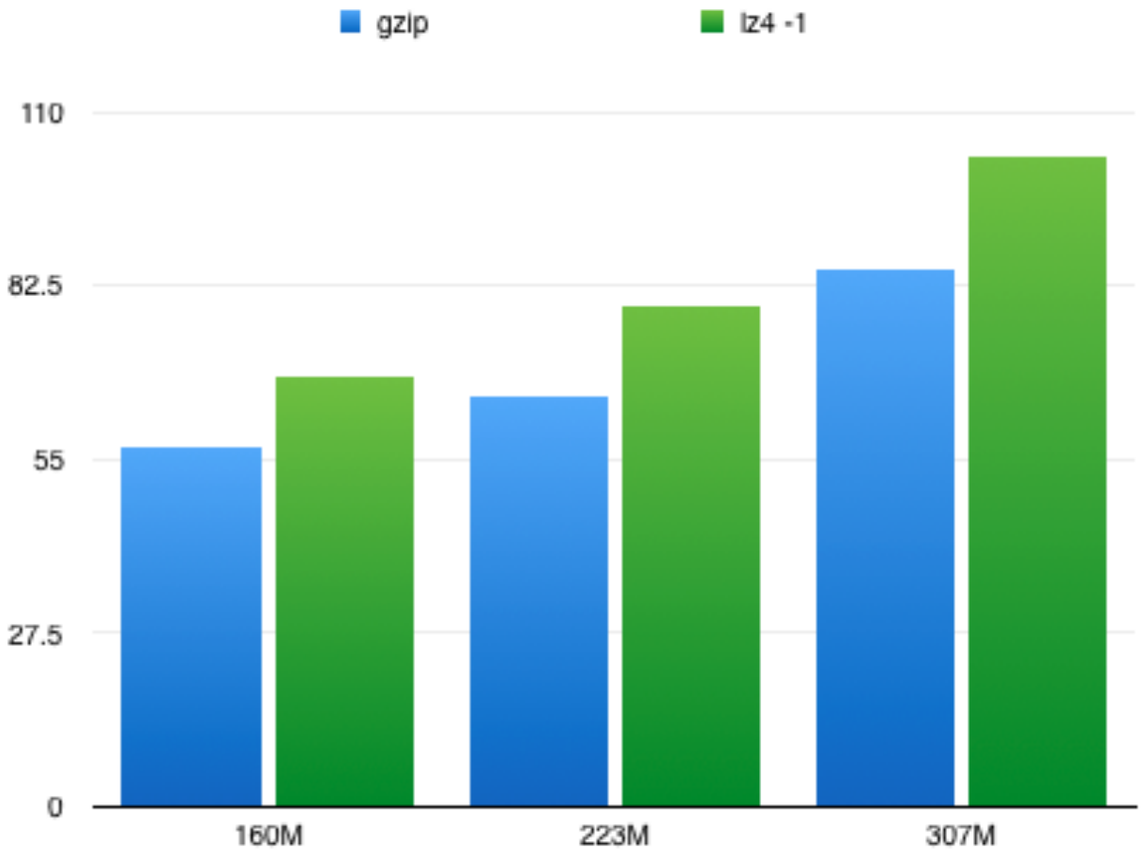
6. Evaluation: optimize total time

specific scenario:
The memory information constructs more than 99% of the total size.

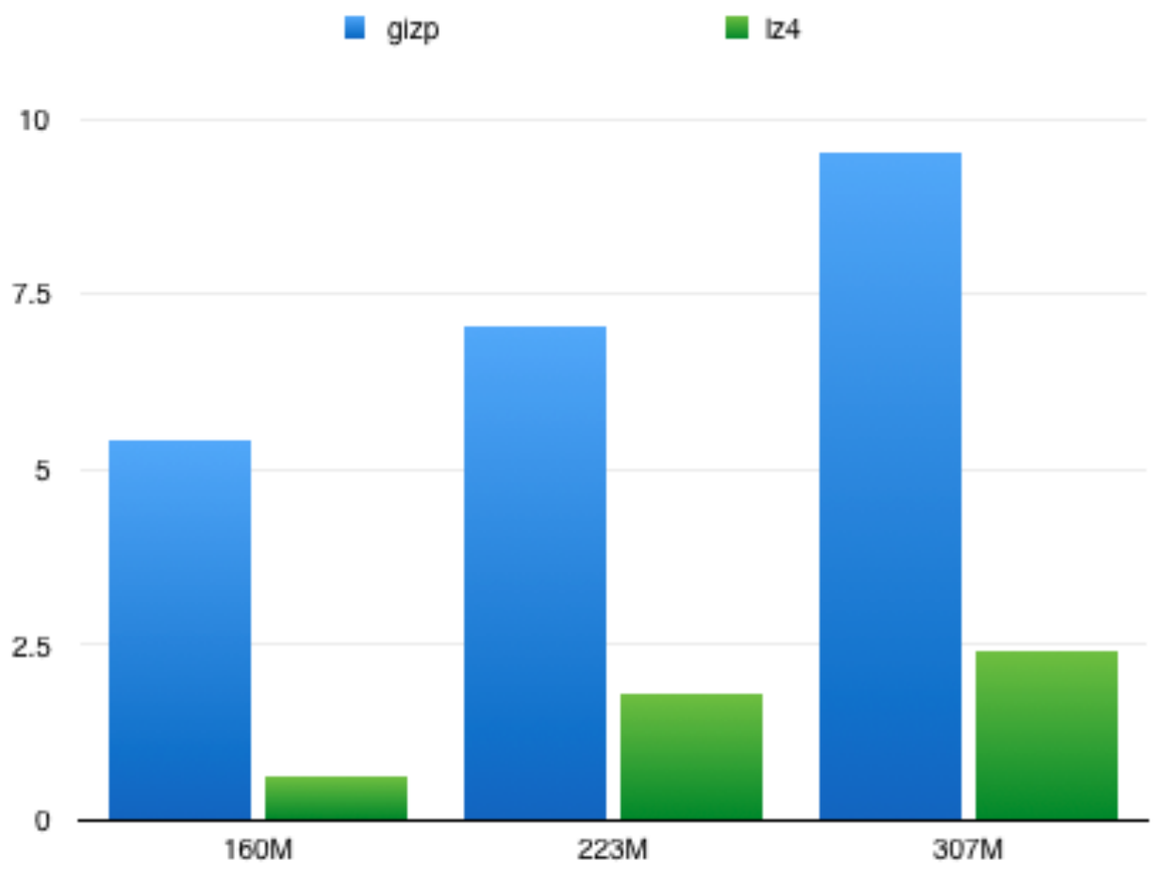
method:
- compress the dump memory image.
- then tar the directory to get a tarball.

Algorithm: LZ4 vs gzip

Live migration: optimize total time



size difference



time difference

compress time is large

compress gain: ~3 - 6M/s

compress speed vs network bandwidth (100Mb/s)

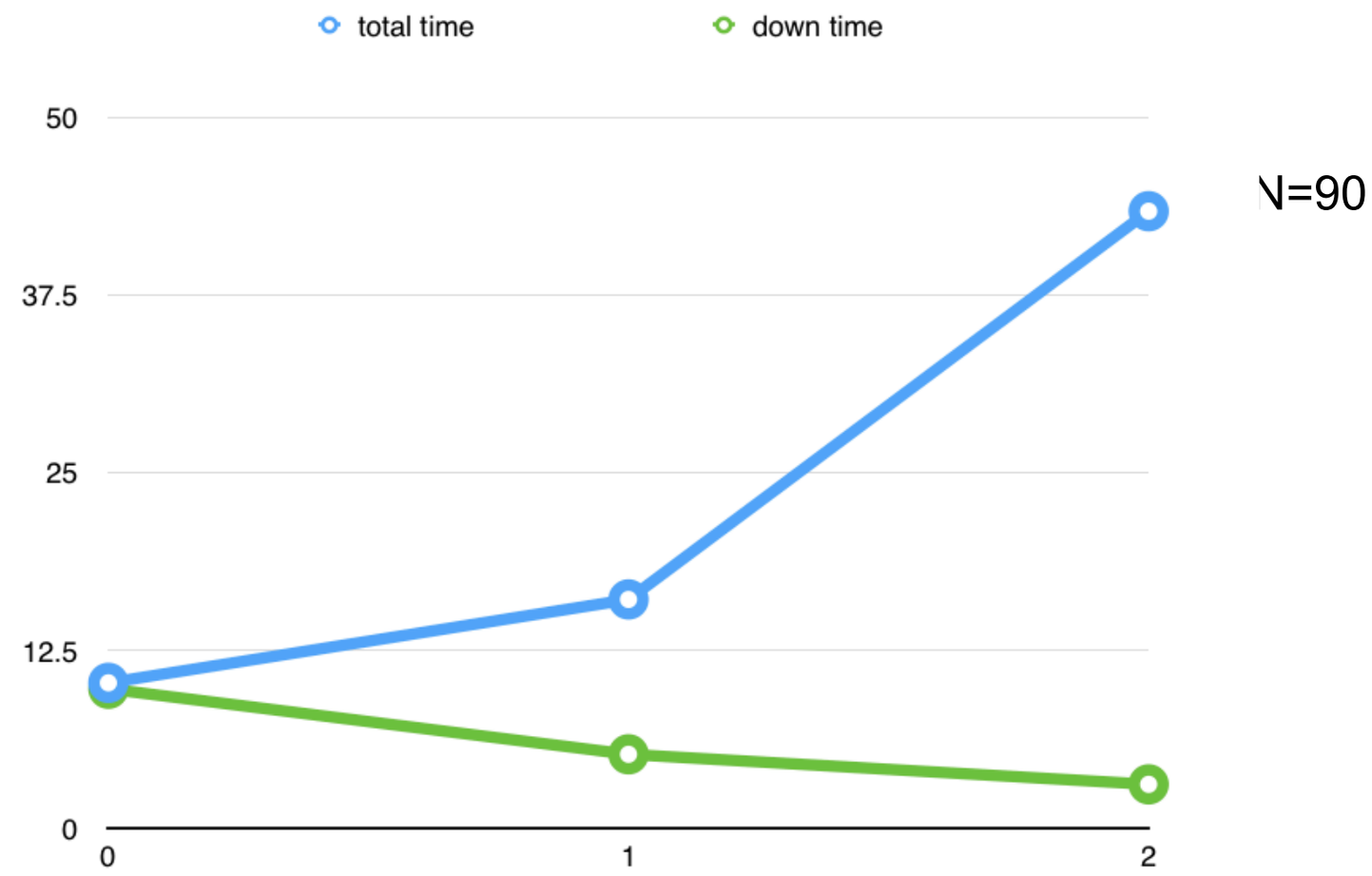
6. Evaluation: optimize down time

iterative dump.

- pre-dump and keep container alive .
- next pre-dump, just get the dirty memory.
- the dirty memory become less and less.
- send time decrease.
- do the last dump, down time begin
- restore.

Applied iterative dump

Note: we run the clients for n times to get dirty memory.
The larger n, the more dirty pages.



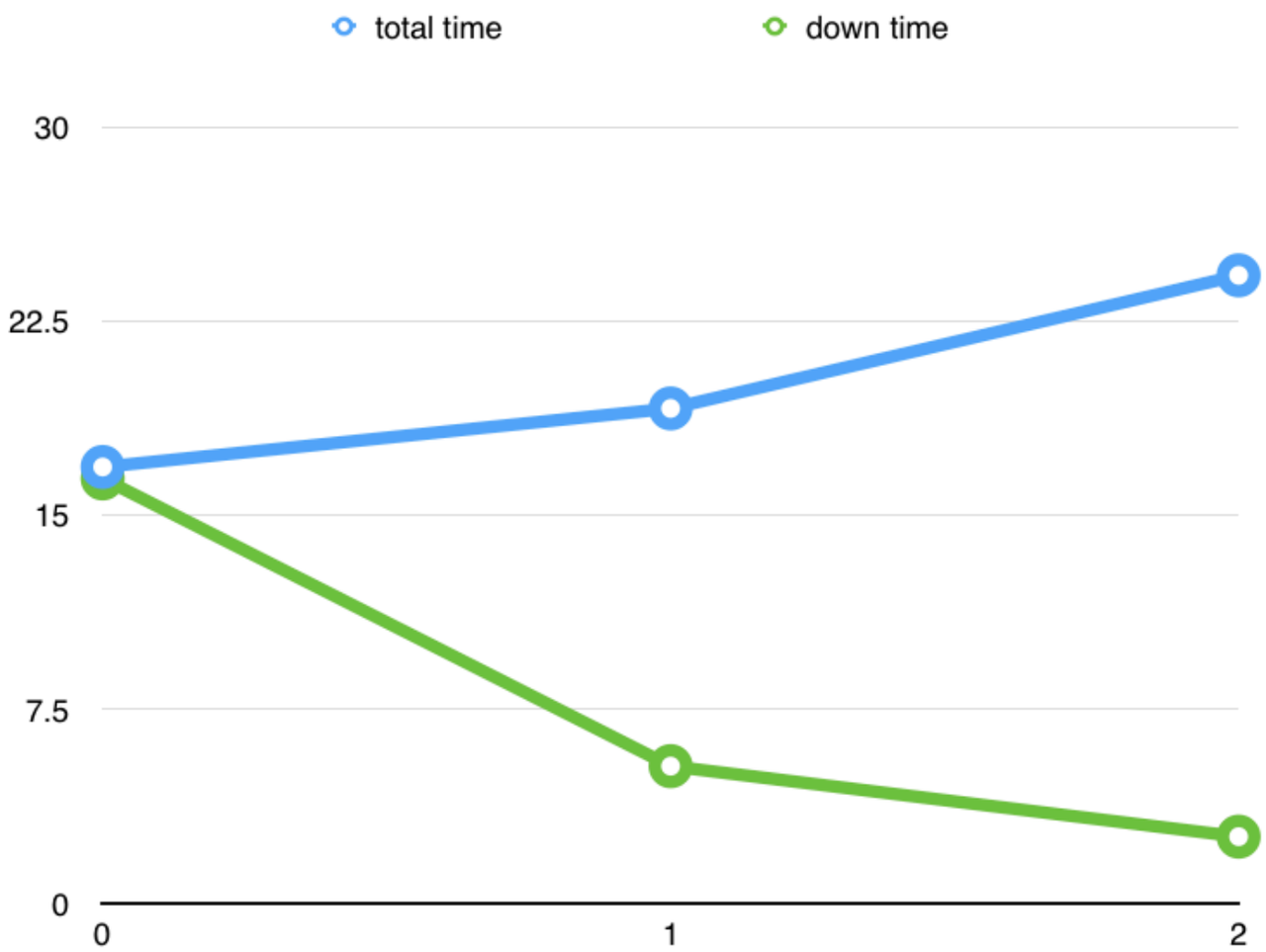
N=90

0,1, 2 means the pre-dump times.

bandwidth: 100 Mbit/s

Applied iterative dump

Note: we run the clients for n times to get dirty memory.
The larger n, the more dirty pages.



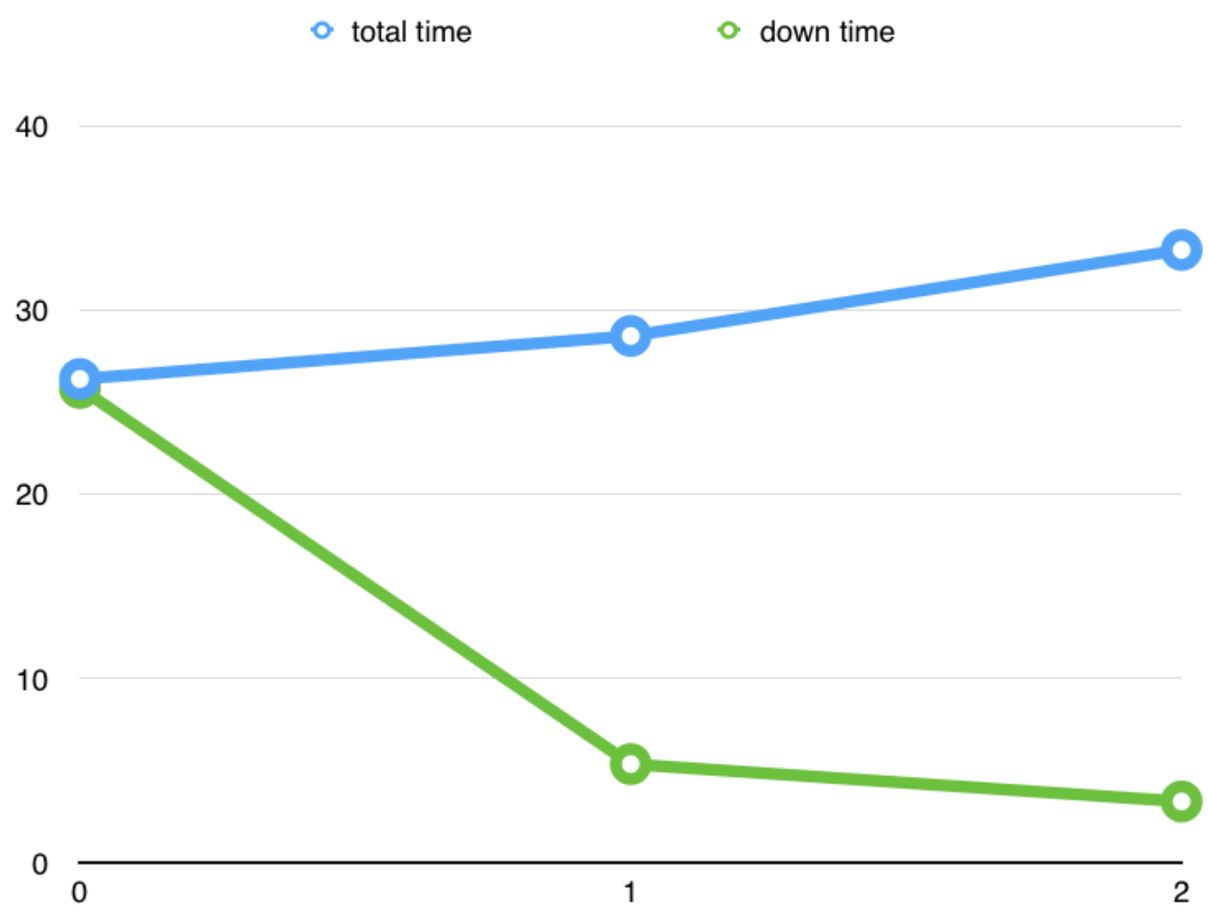
N=180

0,1, 2 means the pre-dump times.

bandwidth: 100 Mbit/s

Applied iterative dump

Note: we run the clients for n times to get dirty memory.
The larger n, the more dirty pages.



N=270

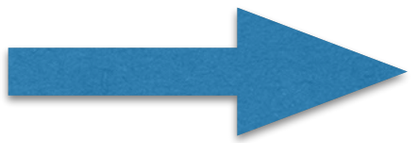
0,1, 2 means the pre-dump times.

bandwidth: 100 Mbit/s

6. Evaluation: Decision model for iterative dump

First pre-dump:

- Get image: depends on the dirty memory
- transfer: image size, compress, bandwidth



Second pre-dump:

- get images: dirty memory during first pre-dump
- transfer

should we do pre dump again?

if not: downtime could be large, but total time could be decrease.
if yes: downtime could be decrease, but total time increases

How small could downtime be?
Gain vs loss

Decision model for iterative dump

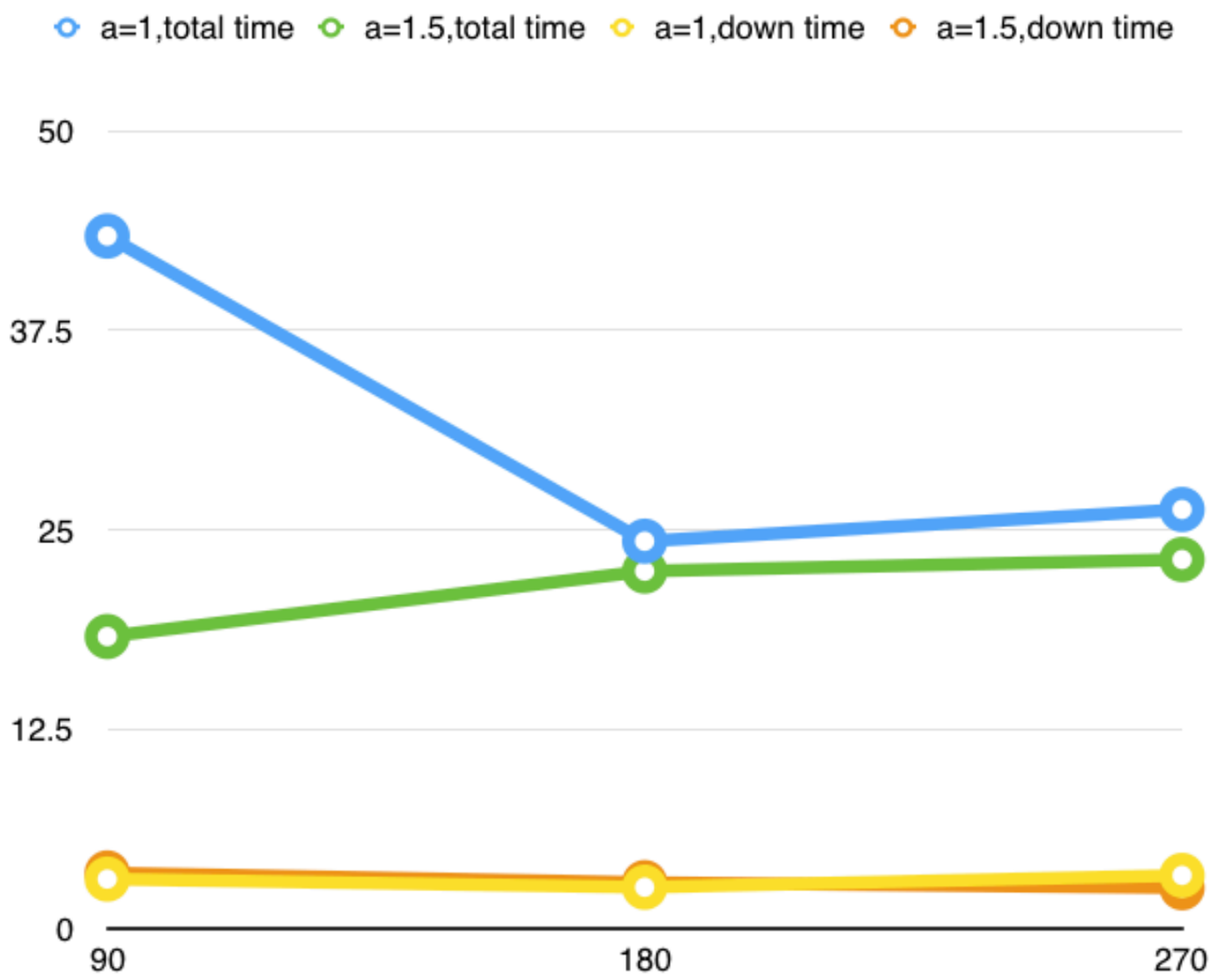
bandwidth = (*send pre dump size*) / (*send_time*)

if: *send time* < ***α***

stop pre dump

In this case, dump memory will be send less than **α** second.

α=1　vs α=1.5



run the clients for n times to get dirty memory. The larger n, the more dirty pages. n=90,180, 270

bandwidth:100Mbit/s

α=1　vs α=1.5

Conclusion:
when α=1.5,The total time decreased significantly.
while the down time just has slight difference.

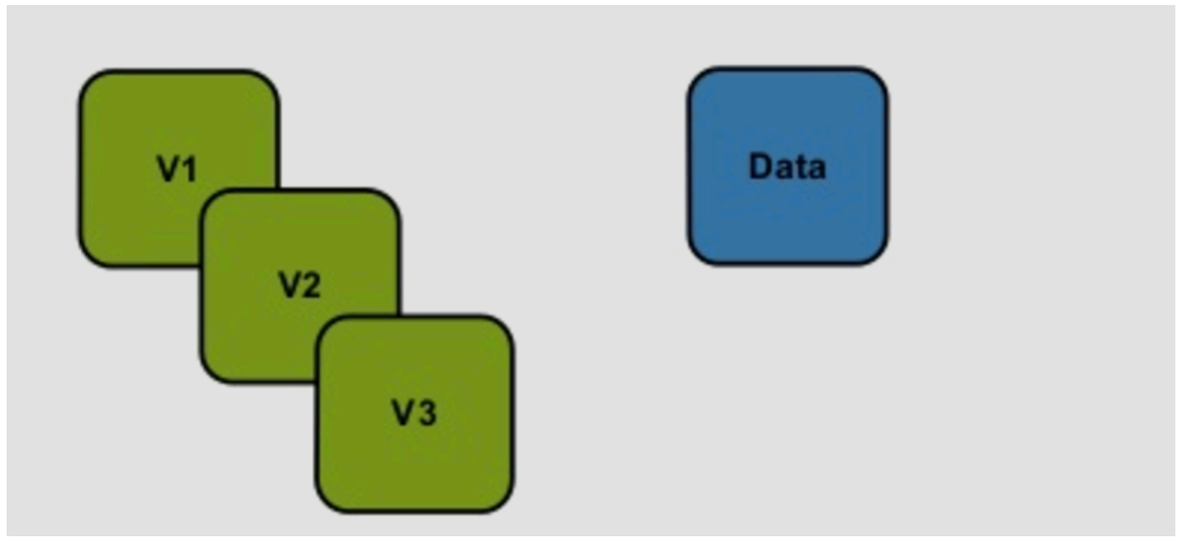
A trade-off game.

Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. Problems of VM-based Cloudlets
3. Docker-based Cloudlets
4. Challenges of Docker-based Cloudlets
5. Solutions
6. Evaluation
7. **Immutability and Micro services.**
8. Future works

7.Immutable Infra and micro services

- Container is not only a lightweight VM.



- Separate data and computation
- No more maintenance of the servers: replacement
- Static pre-built images.
- Easy rollback.

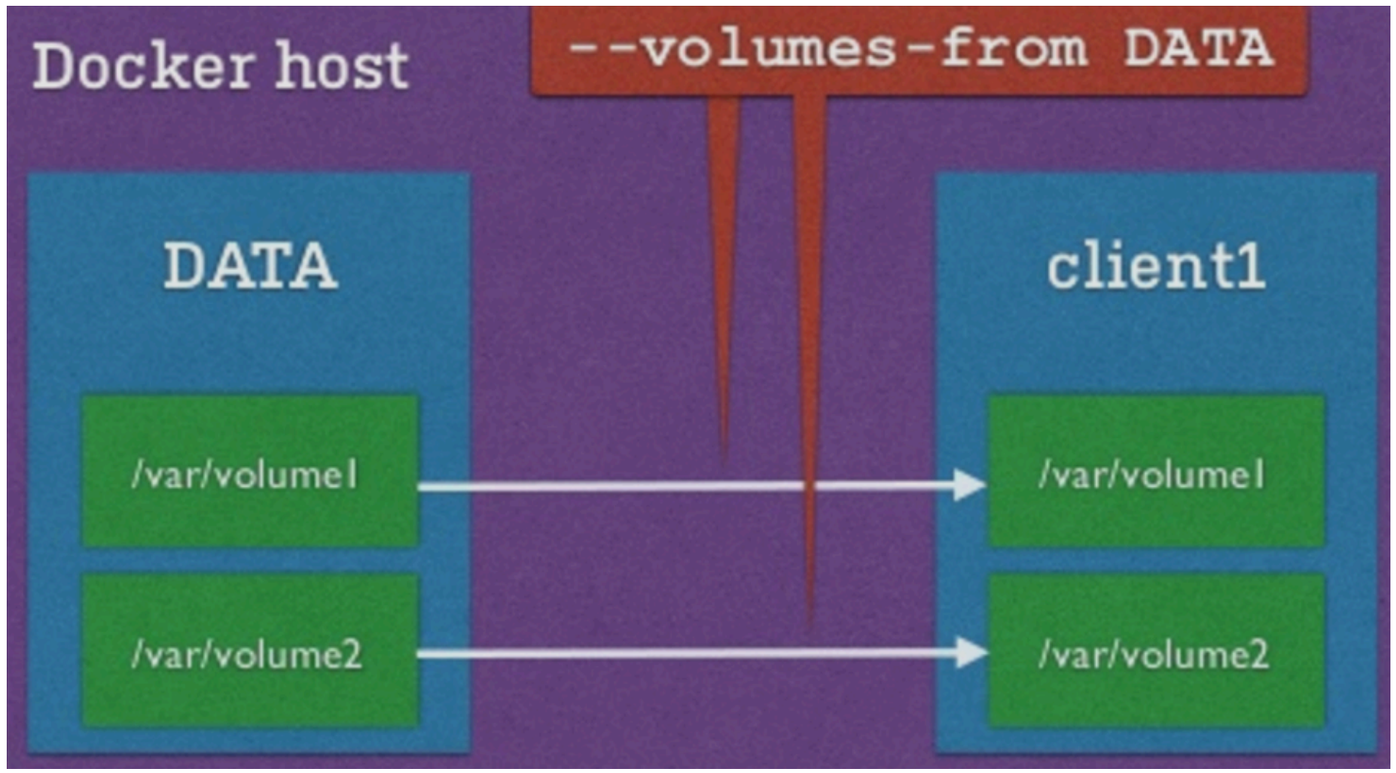
- Migrate the old one or Create new one?

Agenda:

1. Introduction: Mobile Edge Computing and Cloudlets
2. Problems of VM-based Cloudlets
3. Docker-based Cloudlets
4. Challenges of Docker-based Cloudlets
5. Solutions
6. Evaluation
7. Immutability and Micro services.
8. **Future works**

8.Future work

- More test work.
- Accurate filesystem syntonization: volume persistent data storage



Demo Time

Thank you.
Q&A?