VCS: A VIRTUAL COMPUTING AND STORAGE APPROACH TO COMPUTING CONTINUUM

Agenda

- Background and problem statement
- Existing solutions and limitation
- Topology and concept
- VCS overview
- Client architecture and awareness
- Virtual storage
- Computing continuum workflow
- Optimization of workflow
- Cloud-edge-client collaboration
- Test bed evaluation
- Future direction
- A very early demo

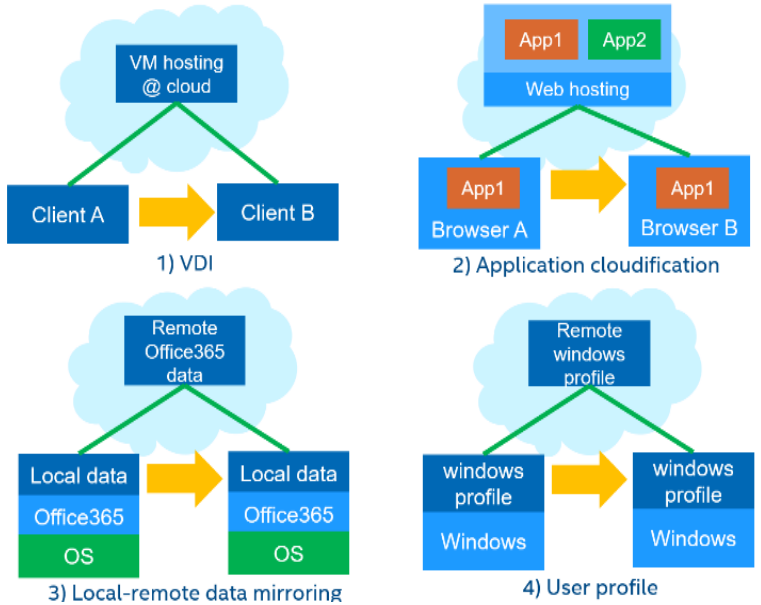
Background and problem statement

- Modern IT industry makes computer hardware cheaper
- 5G and broadband makes internet faster
- More users might have multiple computing devices like laptop, phone and special equipment

- New request comes out - access same copy of desktop from multiple devices located at geographically dispersed locations

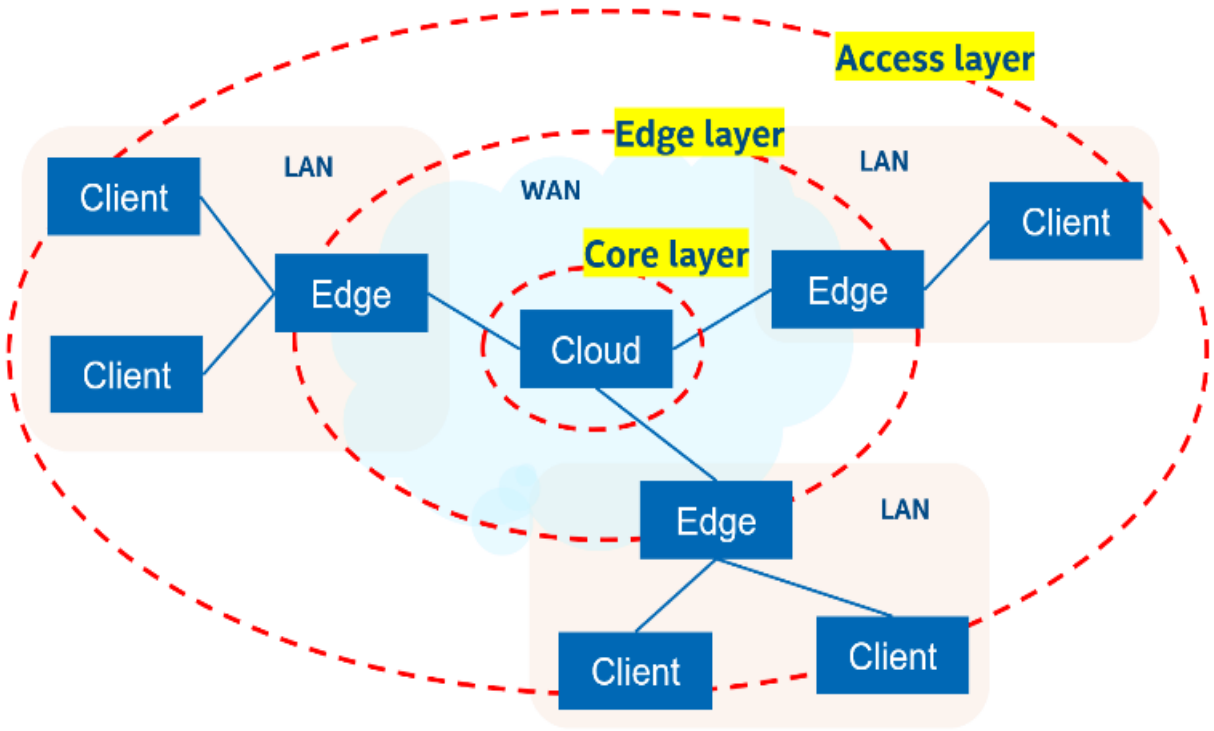
Existing solutions and limitation

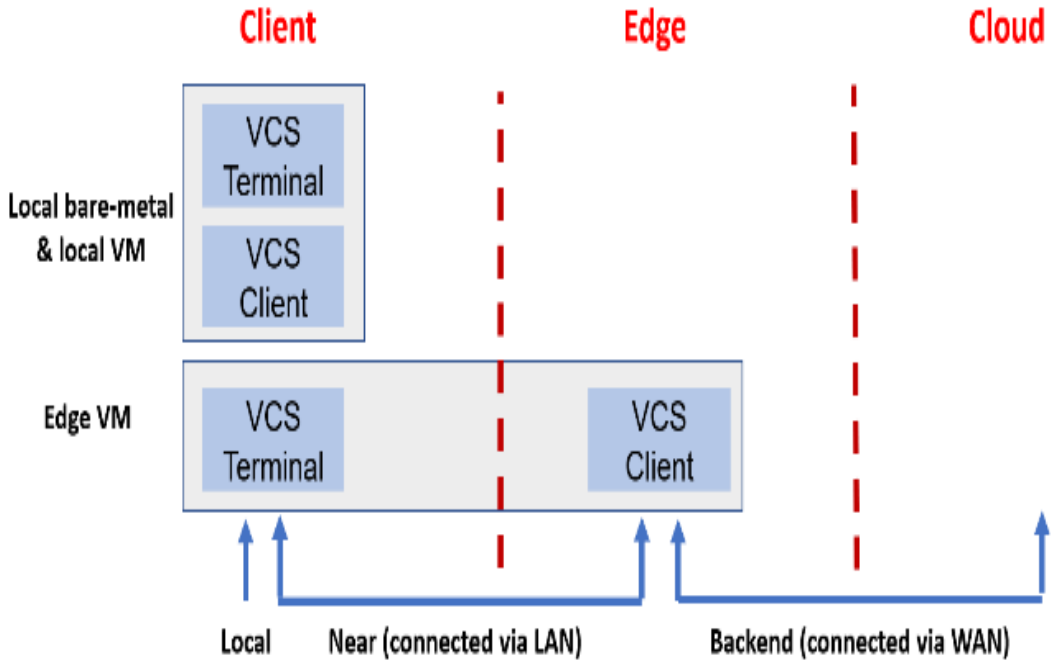
	ZAP, NomadBIOS	CR&TR, LS	JIT, Sledge	VDI	Application cloudification	Local-remote data mirroring	User profile
What is migrated	process	VM	container	desktop	application	Data	profile
Support all OS	×	×	×	√	√	×	×
Desktop continuum	×	√	×	√	×	×	√
No legacy application porting	√	√	√	√	×	×	√
No strong network dependency	×	×	×	×	×	√	√
Computing performance near to native	√	√	√	×	×	√	√



- Steven et al proposed Zap, a pod or group of process migration solution based on an OS level thin virtualization layer.
- Jacob et al proposed an on-the-fly entire OS migration solution based on NomadBIOS, a Xen-based hypervisor with pre-copy migration to keep the OS running while migrating, tracking the changes of its address space and sending updates of image over a couple of iterations.
- Haikun et al provided a CR & TR (checkpointing/recovery and trace/replay) mechanism for fast and transparent VM migration design to reduce the migration time and save network bandwidth.
- Fei and Xiaoming et al proposed a three-layer image management mechanism and correlated layer moving system to improve the migration performance.
- Takahiro and Hirotaka et al suggested a very good live storage migration design with on-demand fetching and background copying strategy to keep IO virtual disk consistency while minimal IO performance impact.

Topology and concept



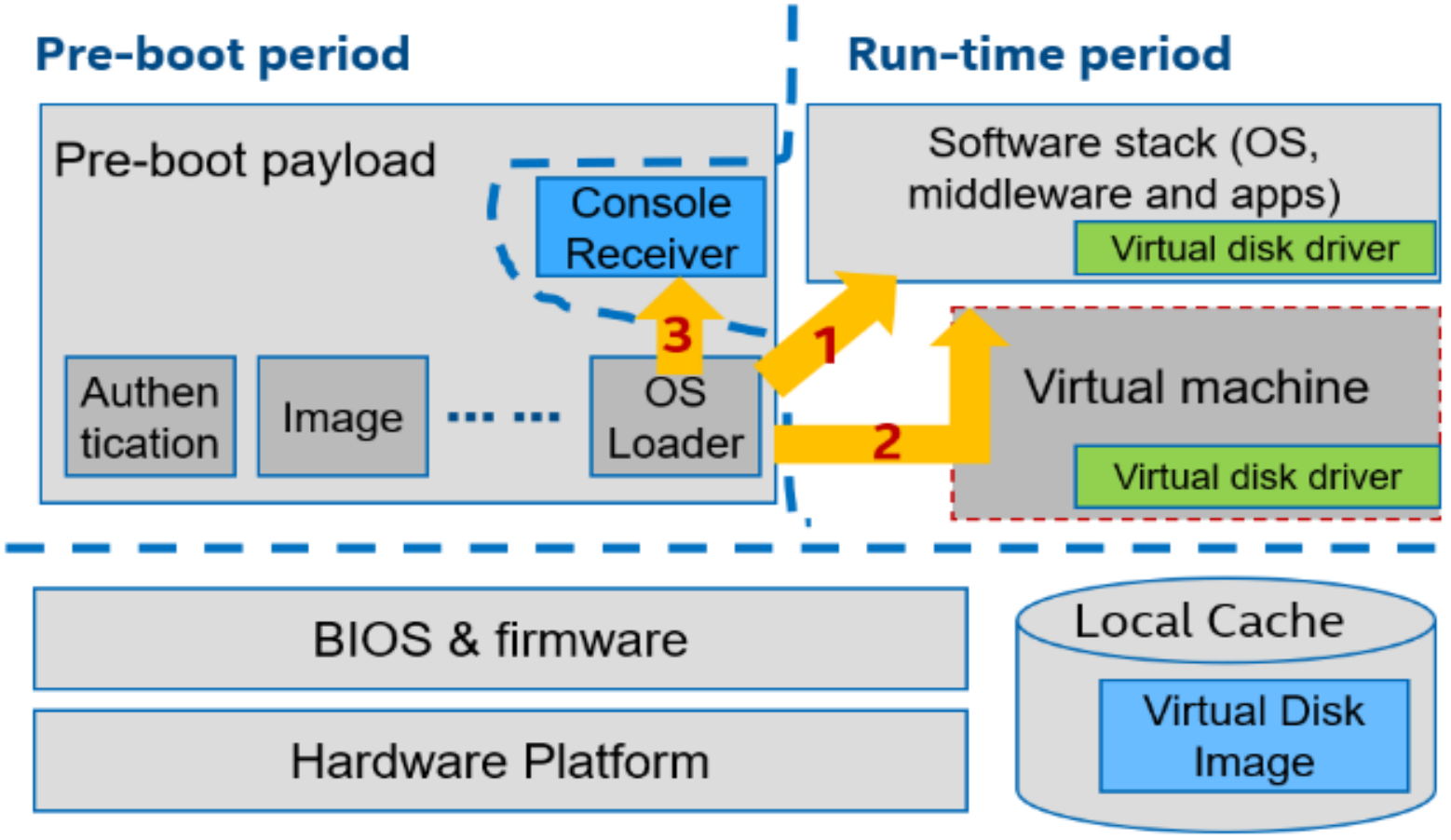


- **Software stack, Desktop**
- **Computing tasks,**
- **Terminal and Client,**
- **Edge and cloud, Local,**
- **near and backend,**
- **Client awareness**

VCS overview – problem statement and scope

- Problem statement:
  - User could **pick up any VCS terminal** within network scope to access her private desktop environment
  - VCS terminal is aware of software stack. It will **designate the most suitable hardware** nearby to provide the best user experience as native as PC.
  - The software stack of each individual user is manageable, updatable, migratable and **propagatable**
- Scope:
  - VCS terminal is mostly **PC-like**, Different users will share similar software stack as much as possible
  - VCS works under **diversified network** environments
  - VCS would support all **mainstream desktop OS**
  - does **not require instancy** or live

Client architecture and awareness



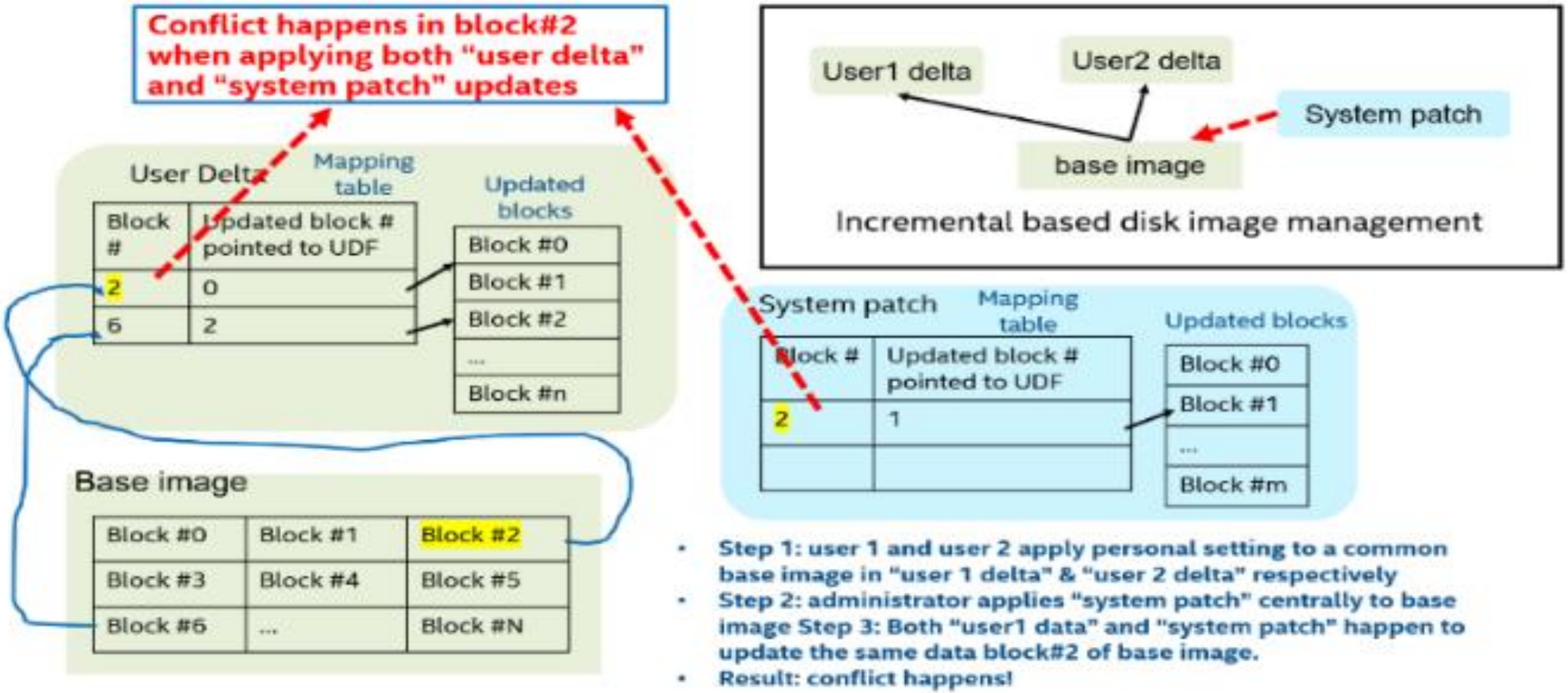
Awareness considerations:

- ISA/CPU arch
- Performance
- Commercial reasons

Result:

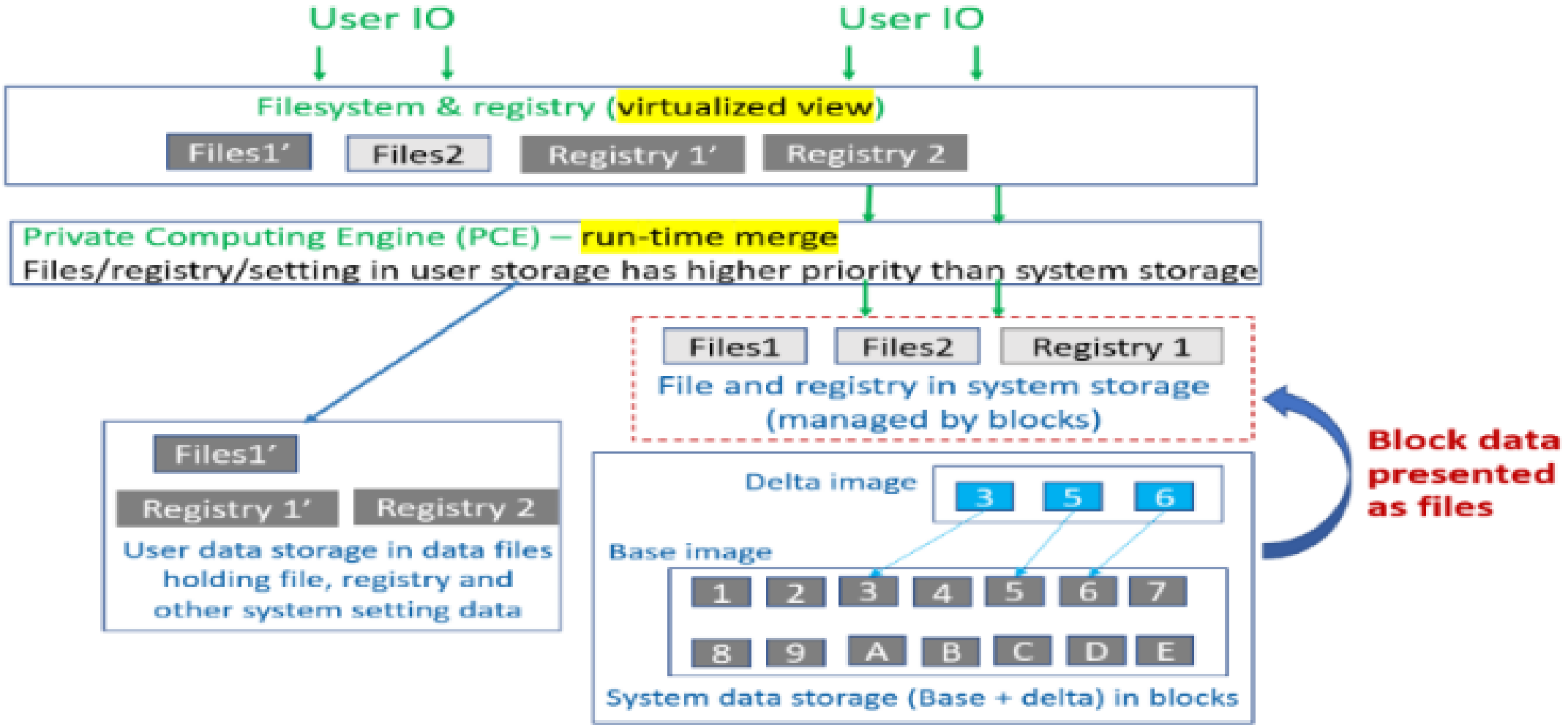
- Local bare-metal
- Local VM
- Adjacent edge

Virtual storage – image overlap problem



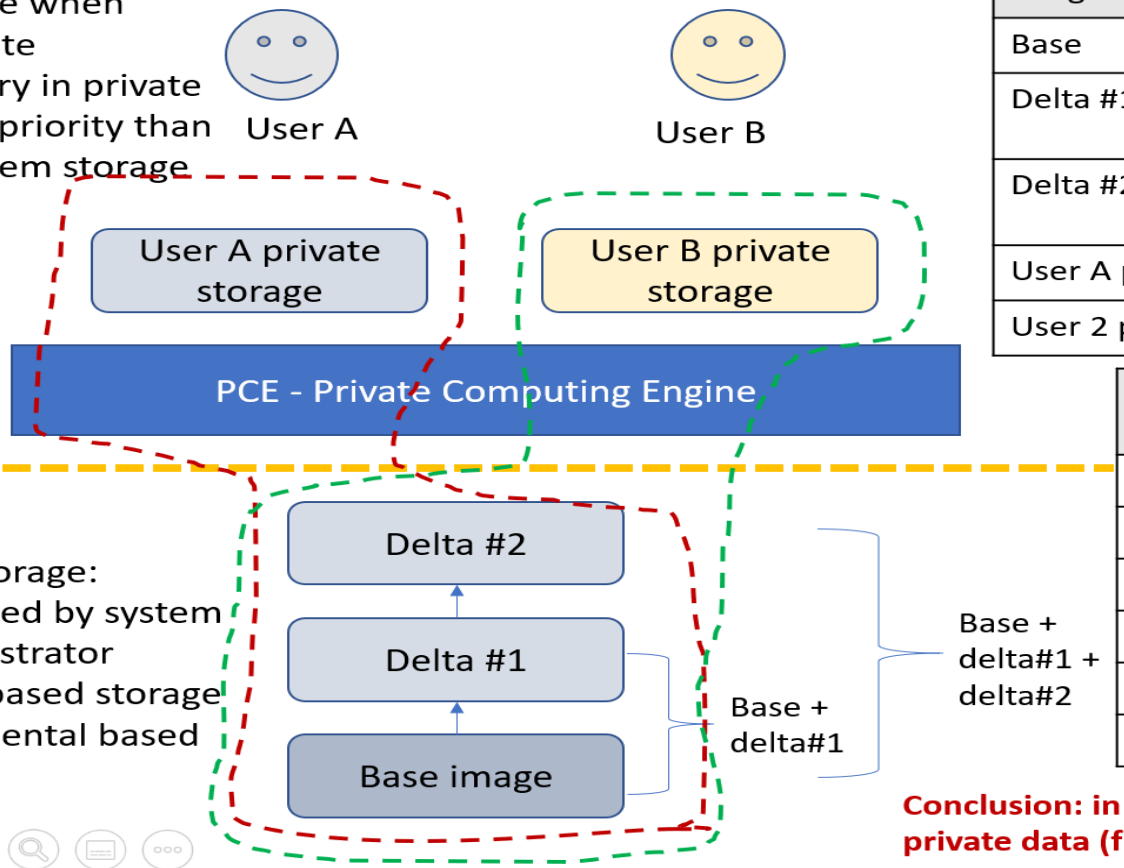
- **Step 1: user 1 and user 2 apply personal setting to a common base image in "user 1 delta" & "user 2 delta" respectively**
- **Step 2: administrator applies "system patch" centrally to base image Step 3: Both "user1 data" and "system patch" happen to update the same data block#2 of base image.**
- **Result: conflict happens!**

Virtual storage – personal computing engine



Virtual storage – an example

Private storage:
- Managed by each user
- File and registry- based
- Only update when write/update
- Files/registry in private has higher priority than that of system storage



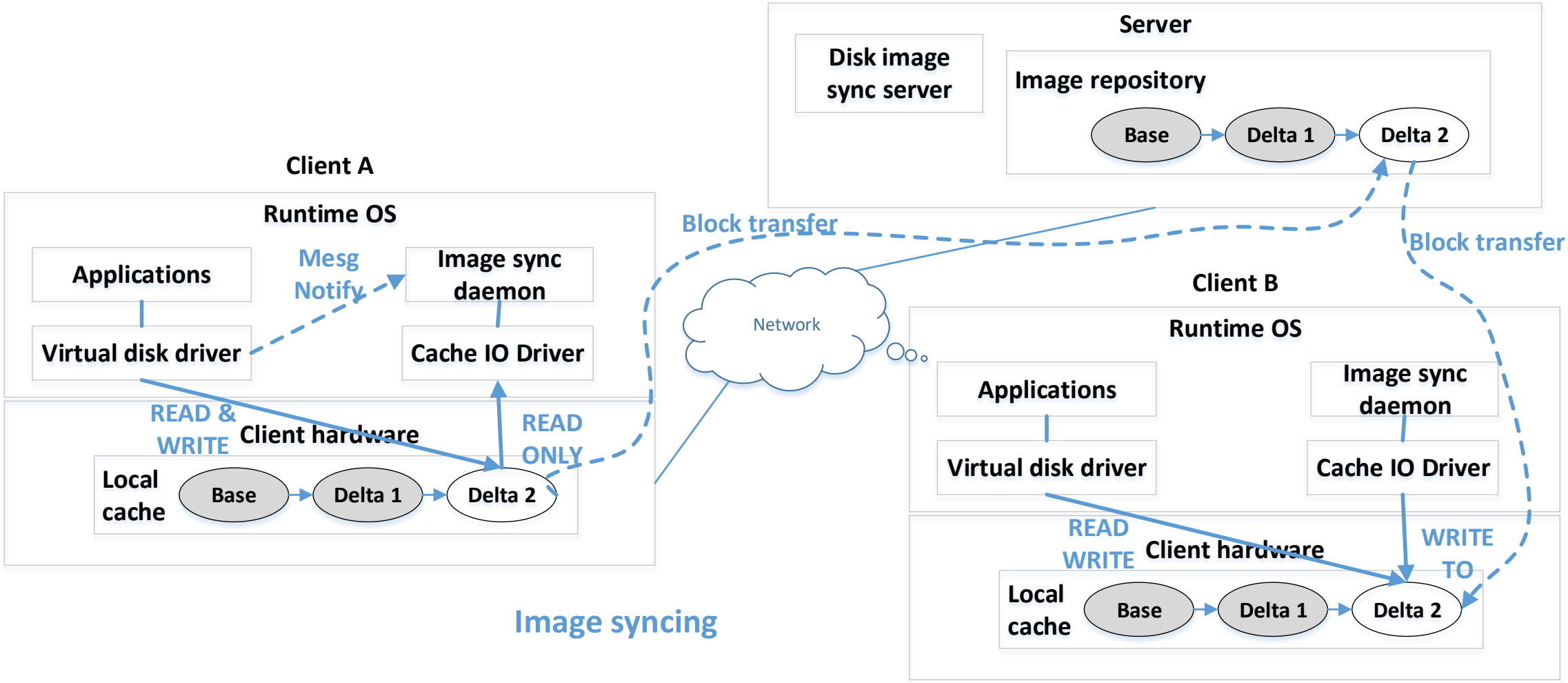
System storage:
- Managed by system administrator
- Block based storage
- Incremental based

Image	Operation
Base	Have 3 files: a, b, c
Delta #1	Update a to a', remove b. so base + delta#1 = a', c
Delta #2	Add file d, so base + delta # + delta#2 = a', c, d
User A private	Private file: a'', d''
User 2 private	Private file: e

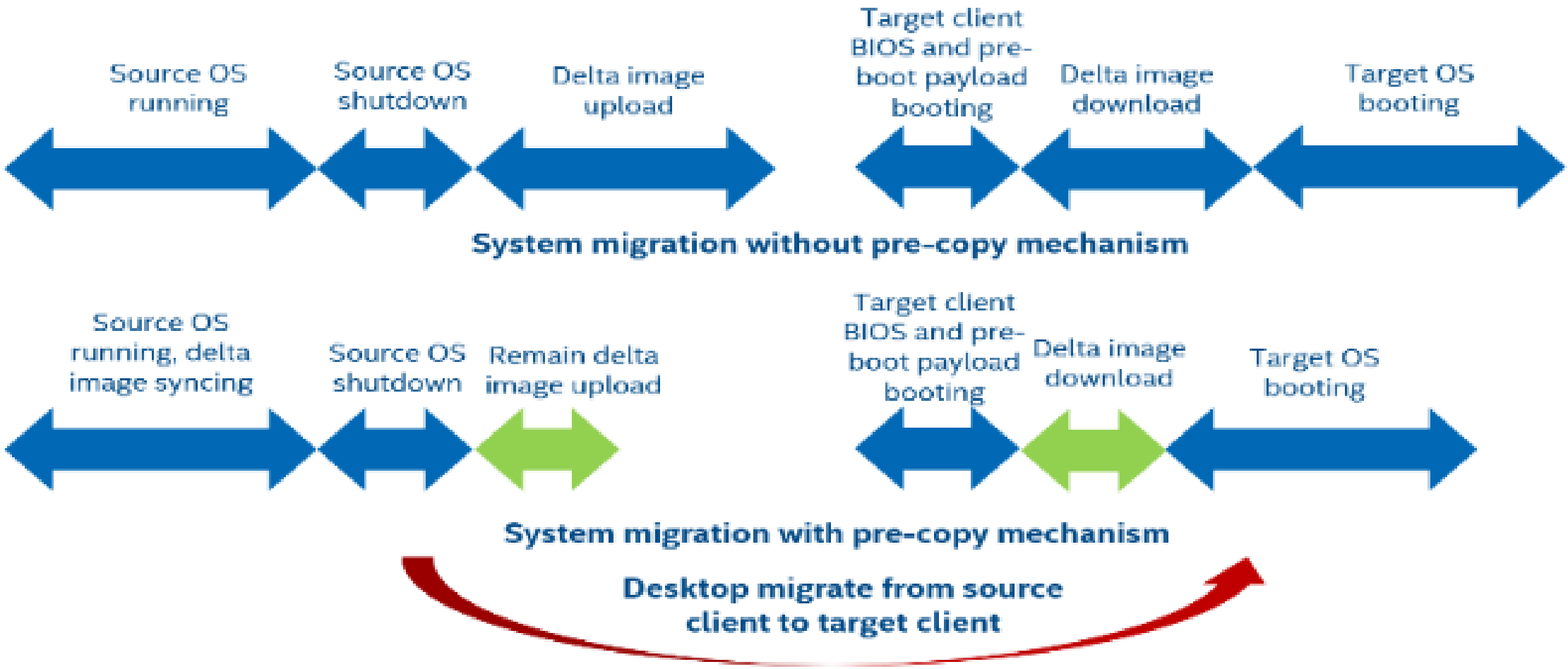
Combination	Result of files
Base + User A	a'', b, c, d''
Base + delta #1 + user A	a'', c, d''
Base + delta #1 + delta #2 + User A	a'', c, d''
Base + User A	a, b, c, e
Base + delta #1 + user A	a', c, e
Base + delta #1+ delta #2 + User A	a', c, d, e

Conclusion: in any combination of system storage & private storage, private data (file) is still there no matter how system data is changed.

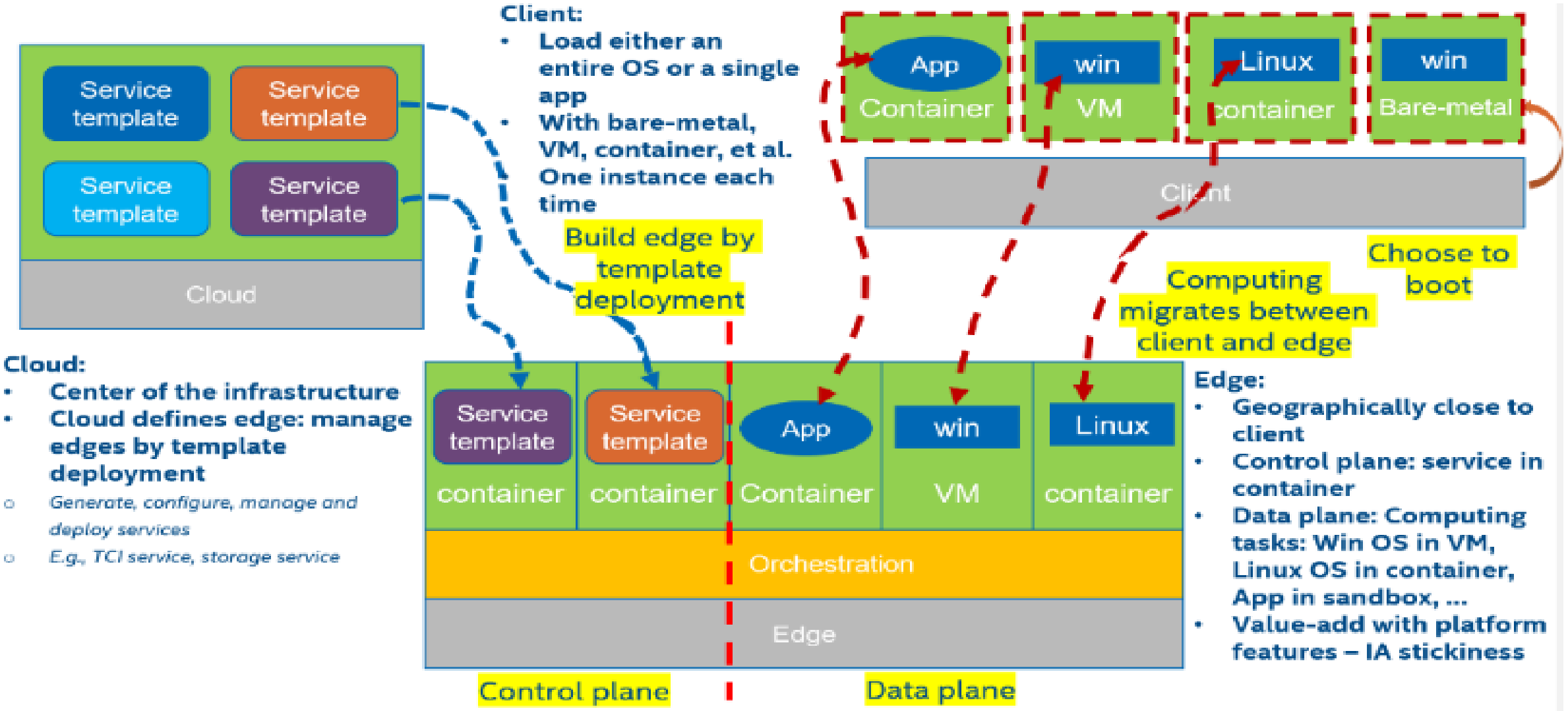
Computing continuum workflow



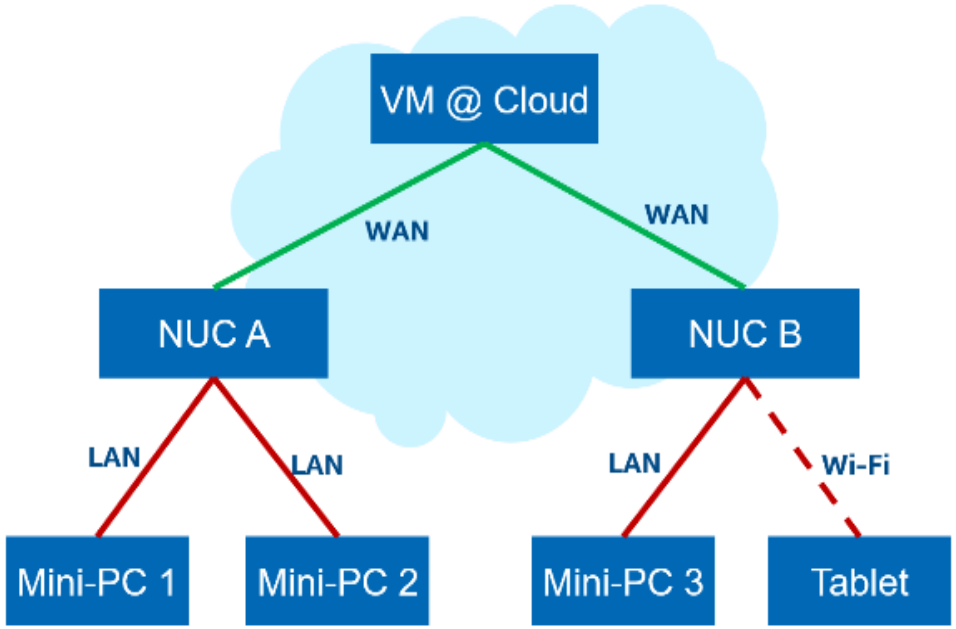
Optimization of workflow: the pre-copy mechanism

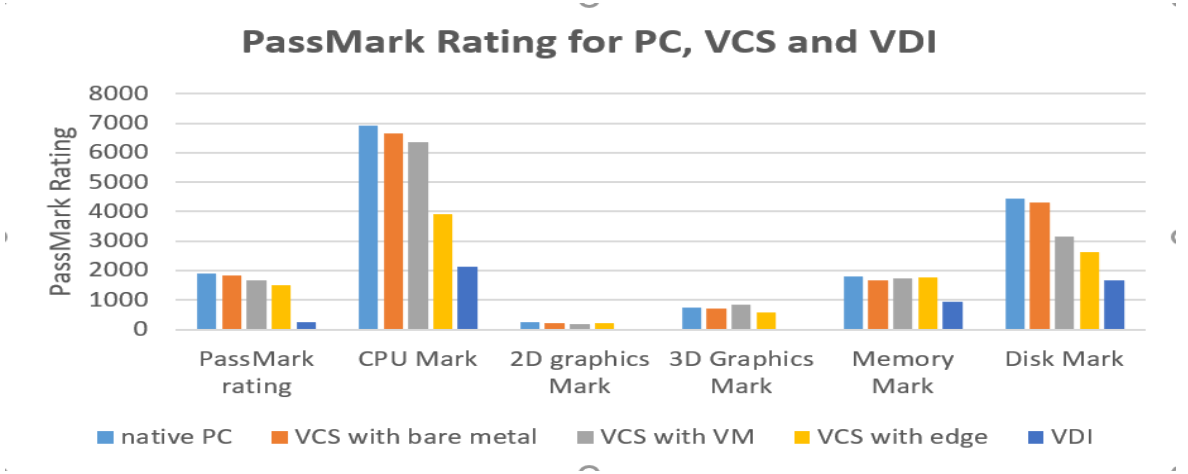


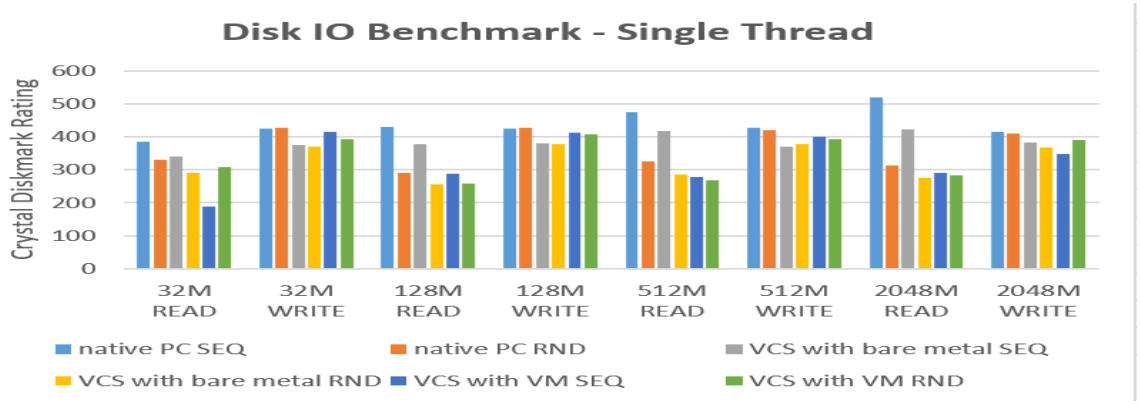
Cloud-edge-client collaboration

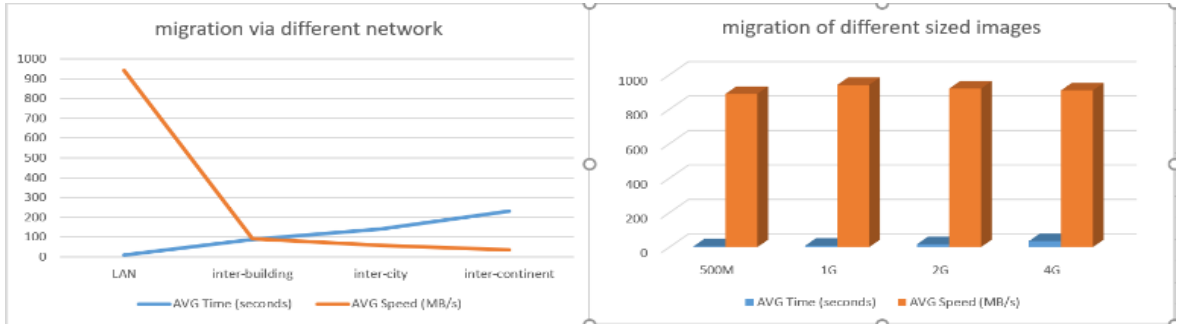


Test bed and evaluation









Future directions

- Application continuum
- Partial continuum?
- Non-network continuum
- Scalable cloud-edge-client hierarchy

A very early demo to show the VCS approach

THANK YOU