Nomad: online migration for geo-distributed storage systems

Nguyen Tran[‡], Marcos K. Aguilera[†], and Mahesh Balakrishnan[†]

[†] Microsoft Research Silicon Valley

[‡] New York University

Internet applications are increasingly geo-distributed

- Large web apps no longer at a single site
- They are geo-distributed

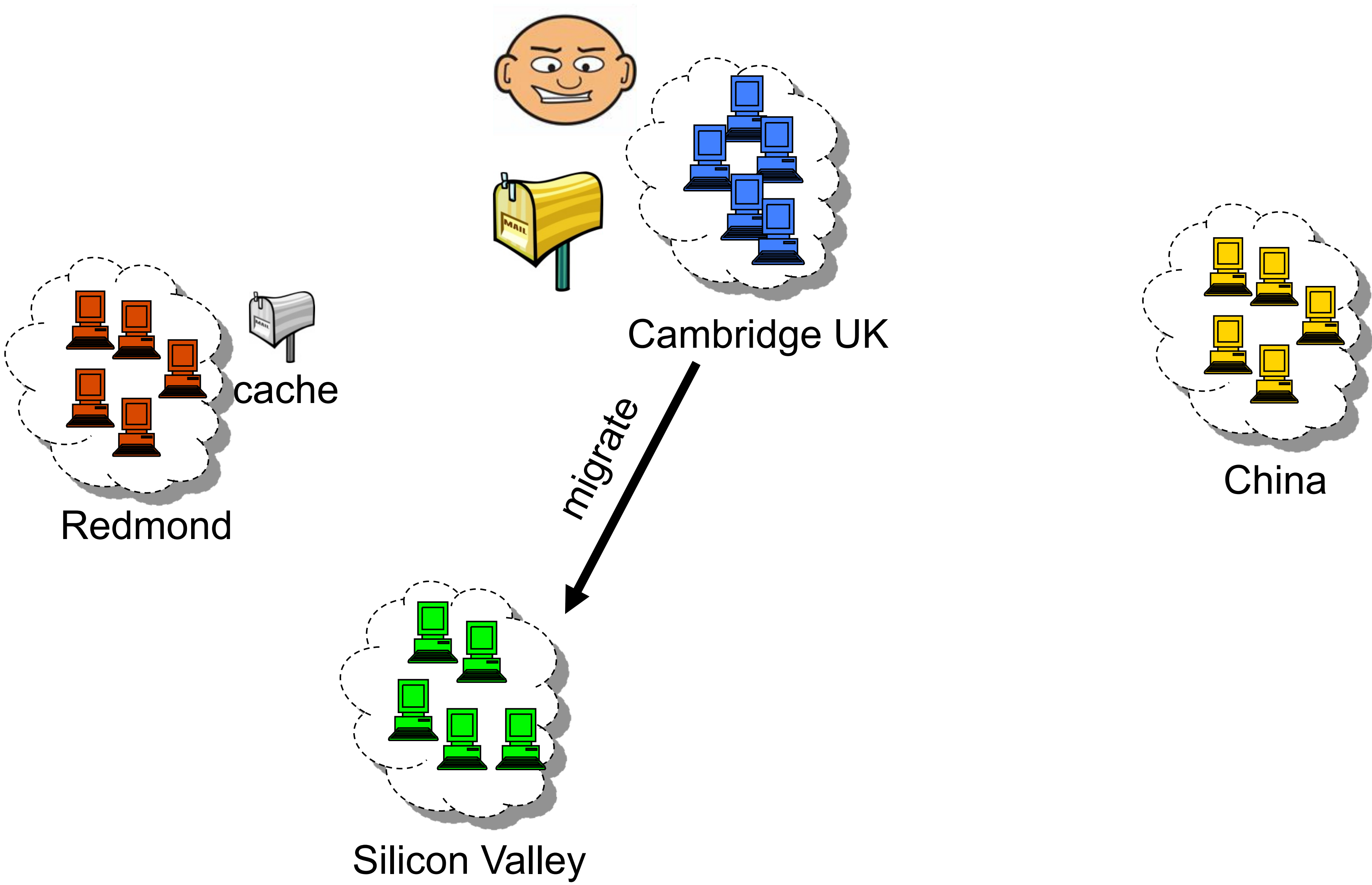
  Geo-distributed: distributed over multiple sites (site=data center)
- Reasons
  - Scalability
  - Reliability
  - Access locality: data close to its user

Geo-distributed storage systems needs to support migration

- Best site for data may change
  - Users relocate
  - Workload changes
  - New sites, new network links

- Migration mechanism
  - Online: data is available and consistent during migration
  - Support for canceling migration or changing target
  - Integrated with caching and replication

Sample use case



Existing approaches: locking, logging

- Locking [Ceph OSDI'06] [GFS SOSP'03] [Farsite OSDI'06]

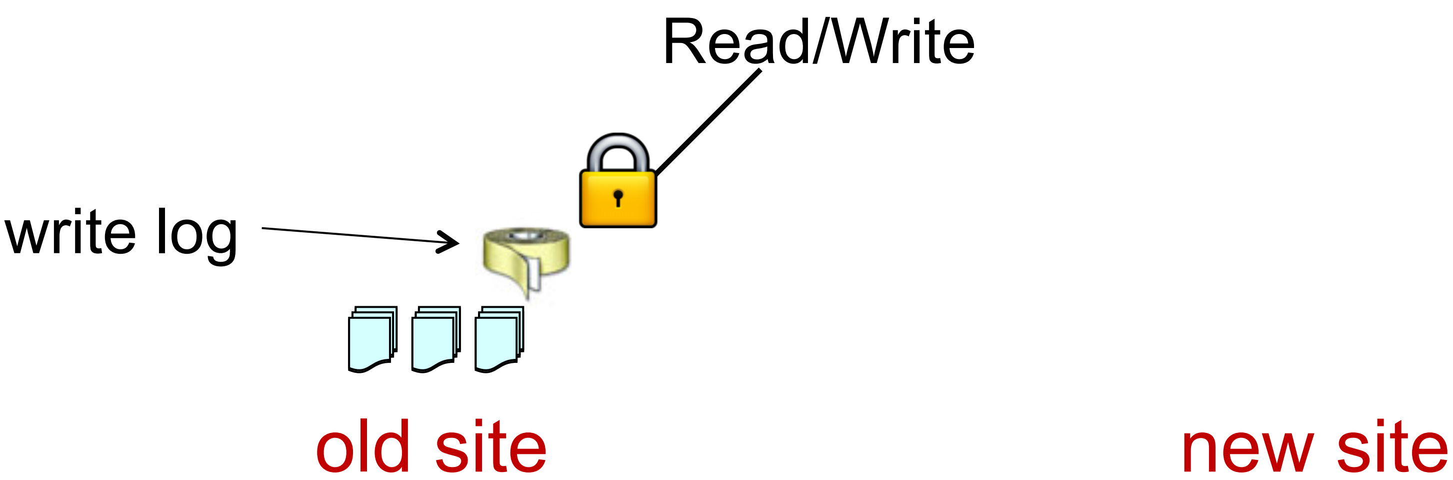
old site new site

Writes are blocked during migration

Migration may take a long time!!!

Existing approaches: locking, logging

- Locking [Ceph OSDI’06] [GFS SOSP’03] [Farsite OSDI’06]
  – Disallow writes during migration
- Logging [AFS TOCS’88] [Farsite OSDI’06]



Existing approaches: locking, logging

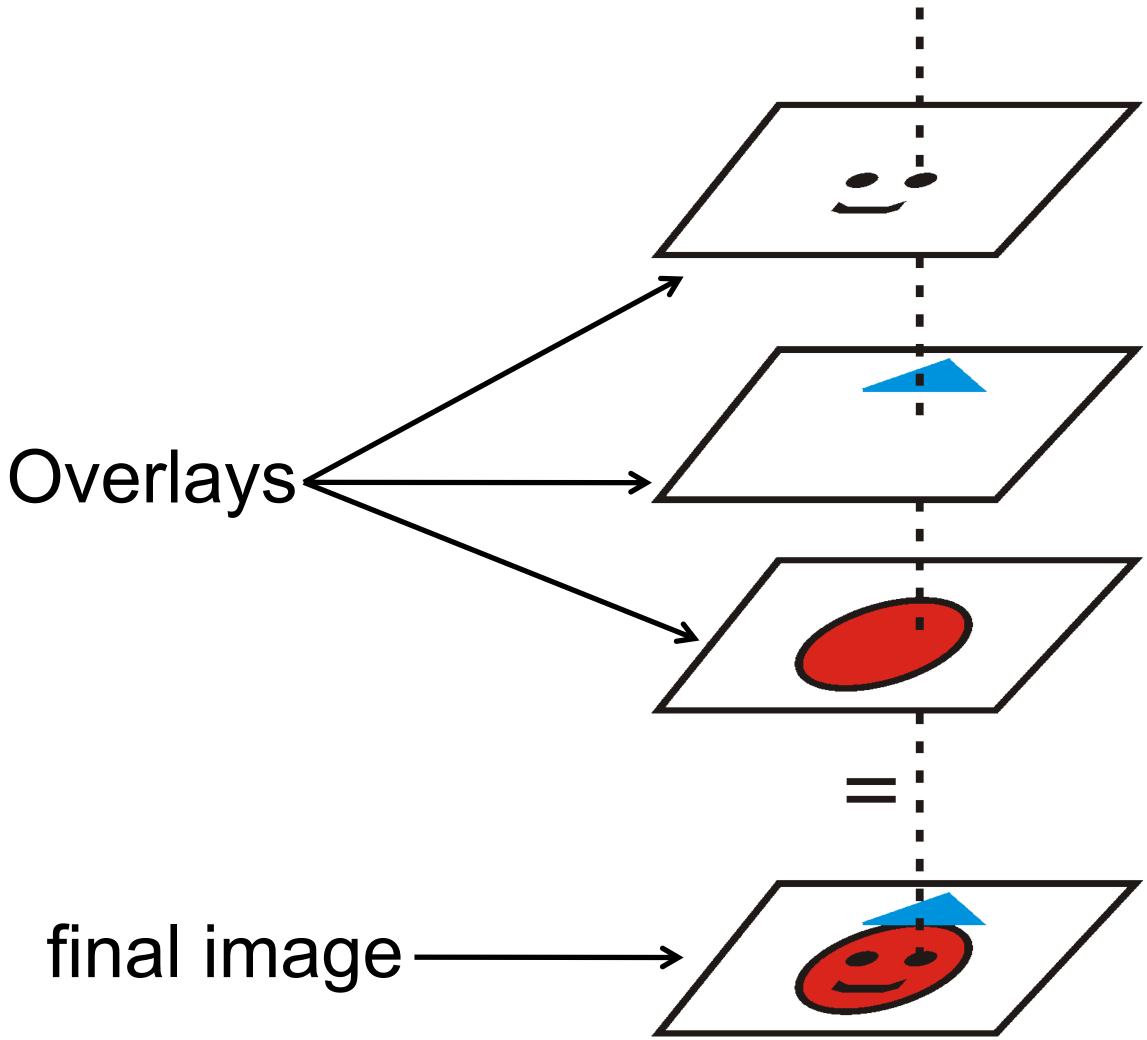
- Locking [Ceph OSDI'06] [GFS SOSP'03] [Farsite OSDI'06]
  - Disallow writes during migration
- Logging [AFS TOCS'88] [Farsite OSDI'06]
  - Disallow writes while transferring log
  - Reads and writes go to old site during migration

    We want reads and writes on new site, otherwise
    - Wastes bandwidth
    - Delays migration benefit
- Both require additional complexity to support caching and replication consistently
  - E.g., cache coherence protocol

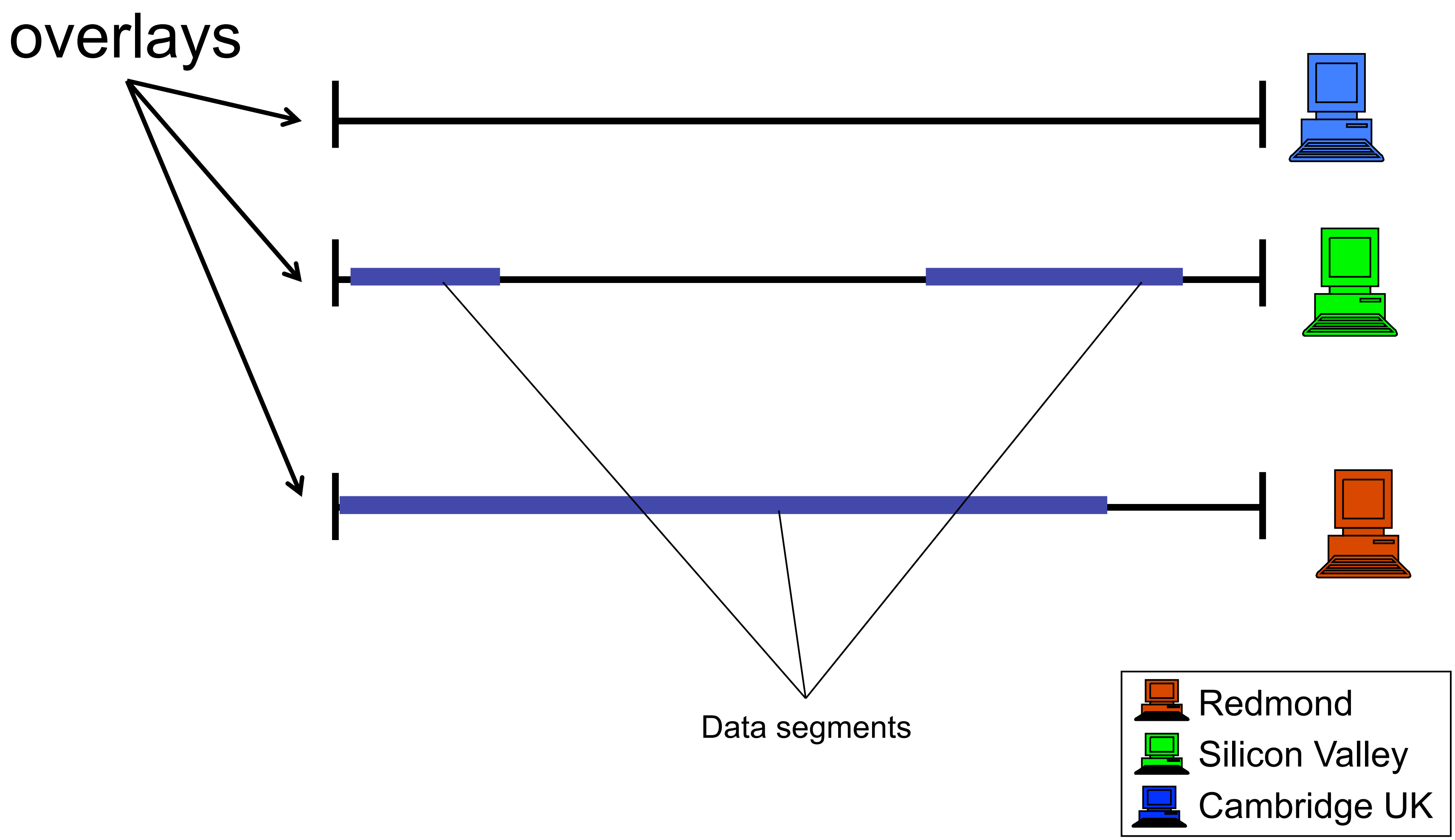
New approach: distributed data overlay

- Data is accessible at all times
- Migration benefit is realized quickly
    - Writes go to new site instantly
    - Reads are served at new site as soon as possible
    - Intuition: read first at new site
      and redirect if data not there
- Seamlessly support caching and replication

Overlay: a simple abstraction

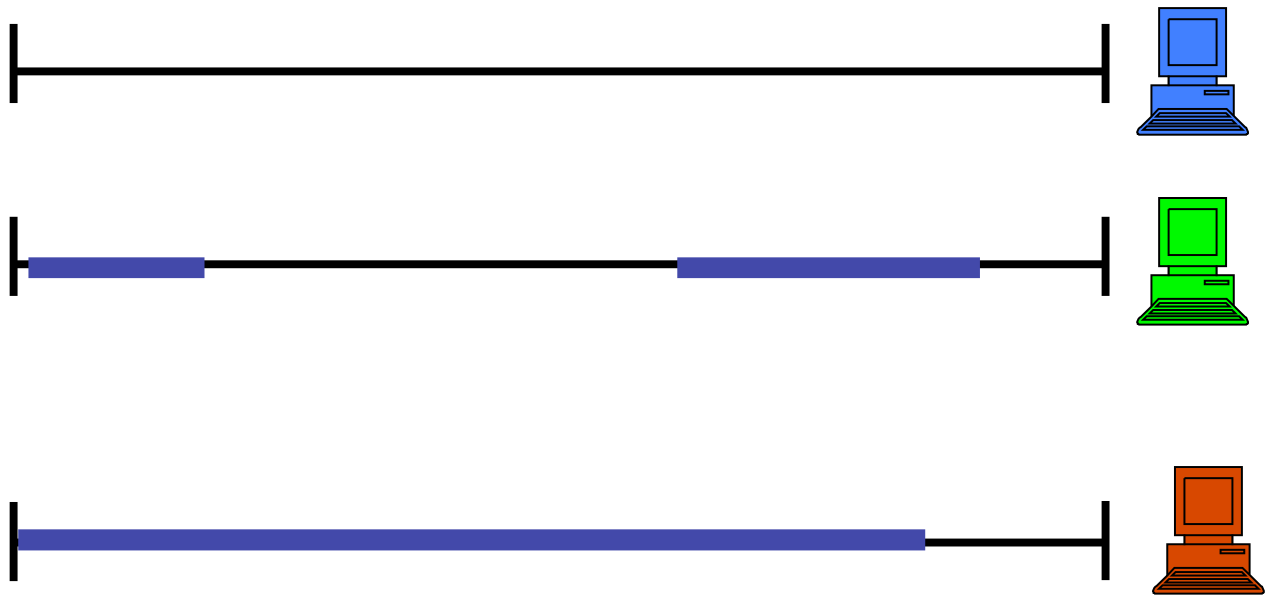


Overlay stack structure for an object

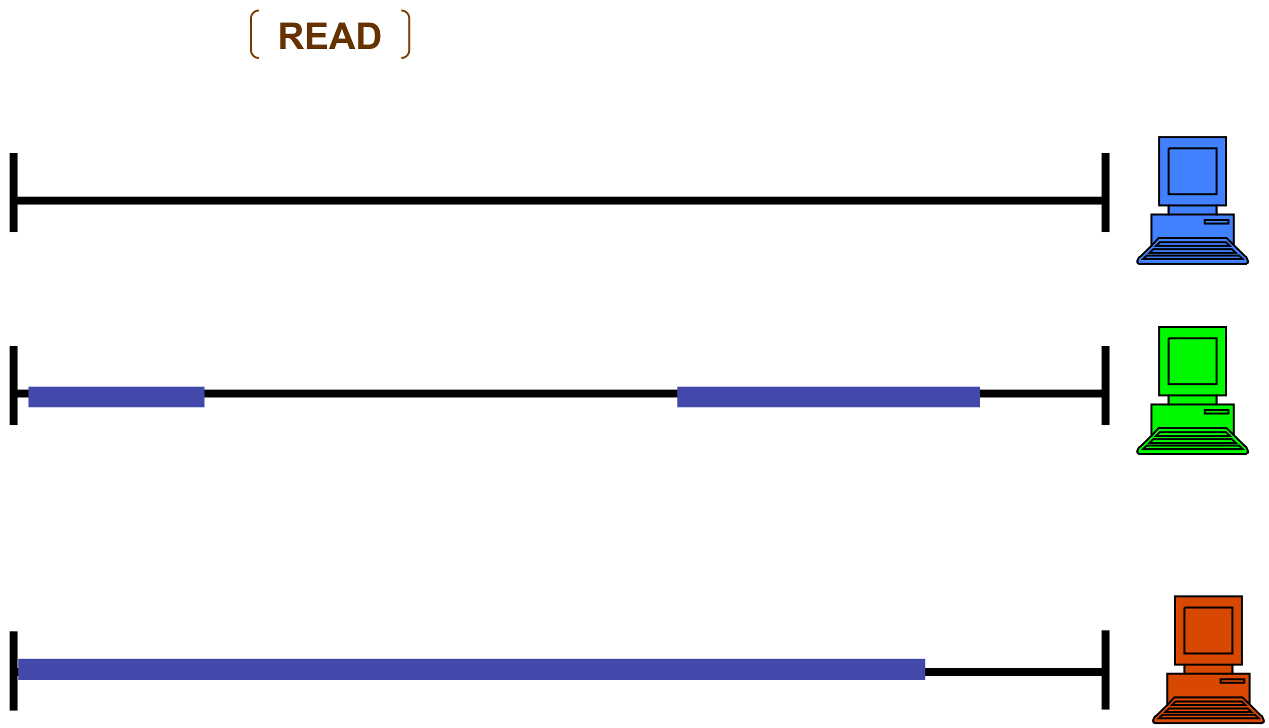


Semantics of overlay operations:
Create, Read, Write, Migrate

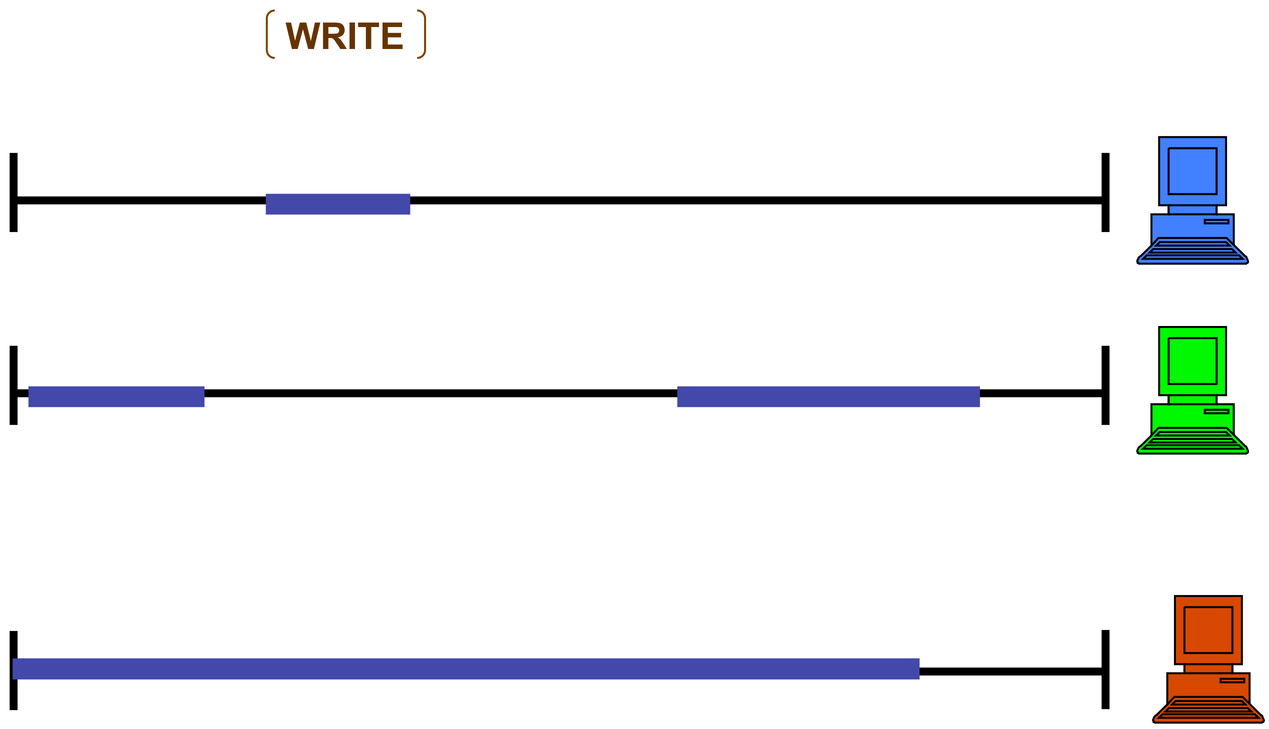
Overlay operation: CREATE

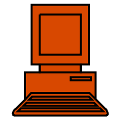
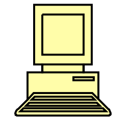


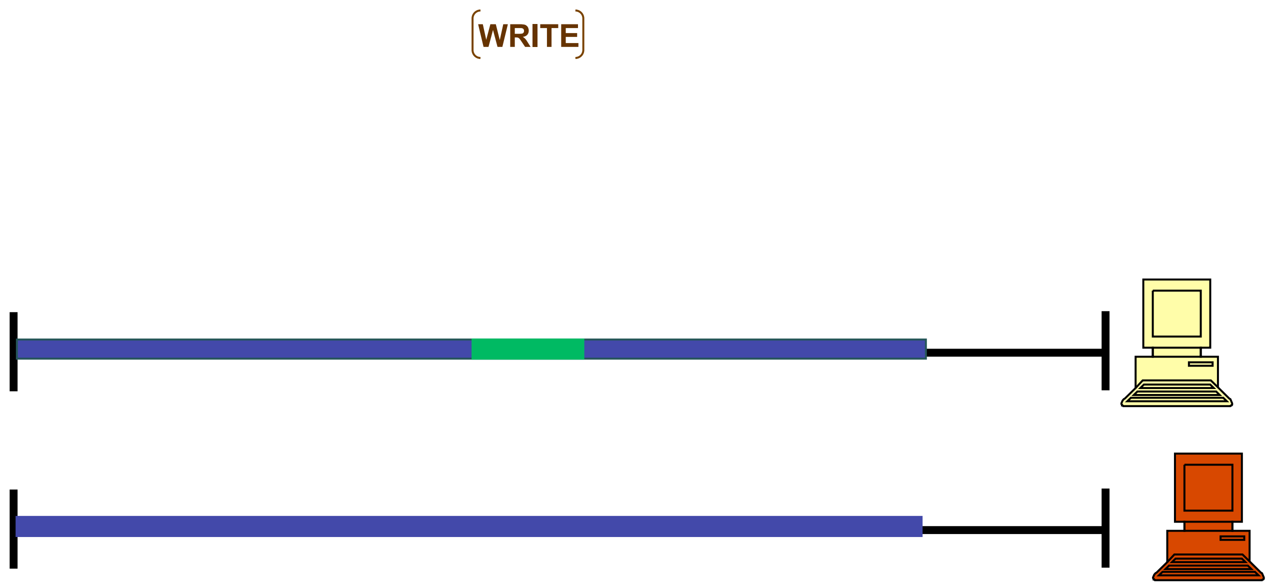
Overlay operation: READ



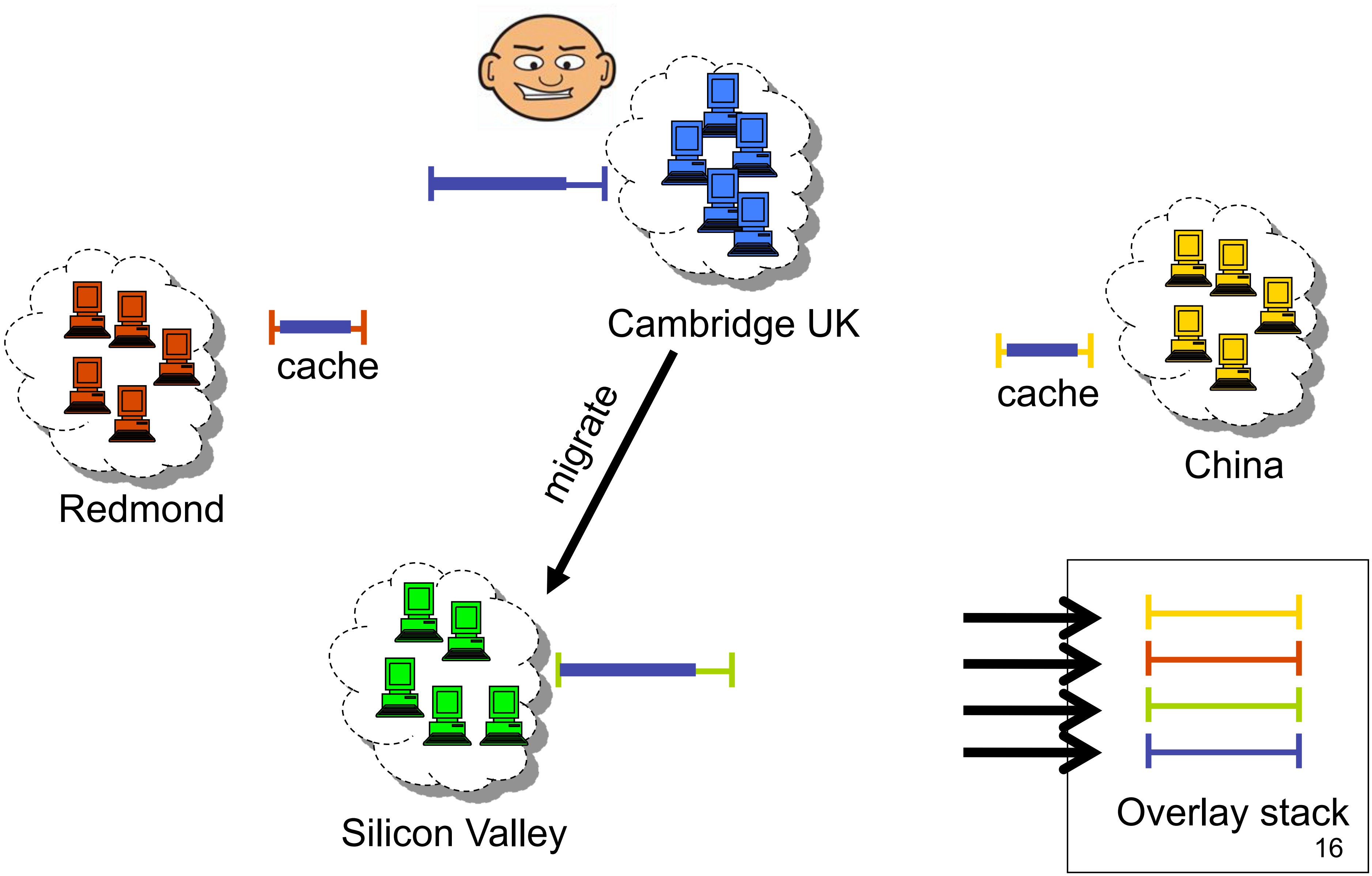
Overlay operation: WRITE



Overlay operation: MIGRATE from  to 

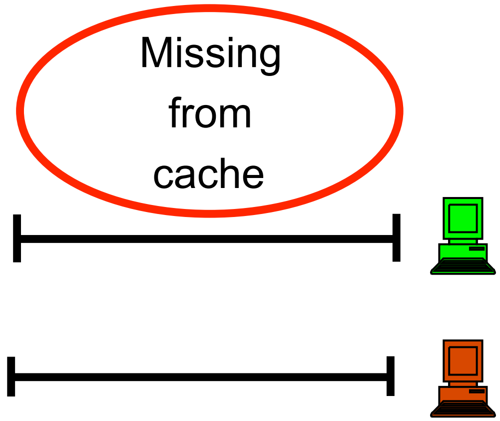


Using overlays



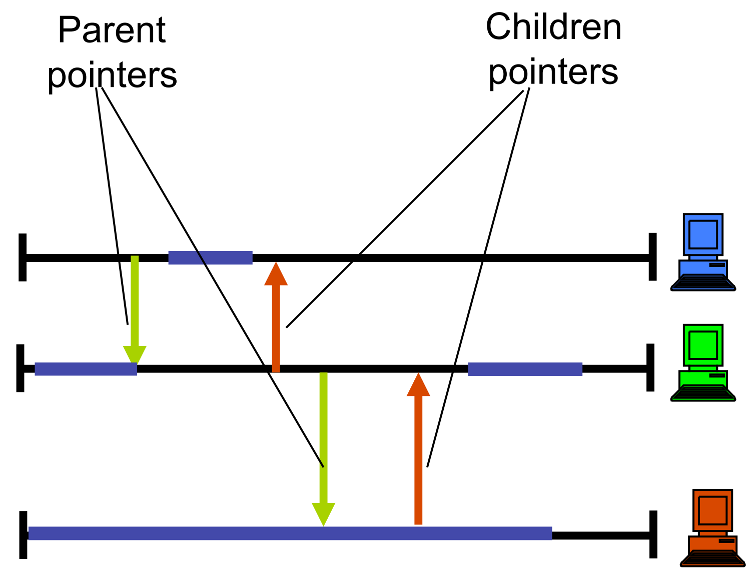
Overlay implementation

Overlay internals



Client side

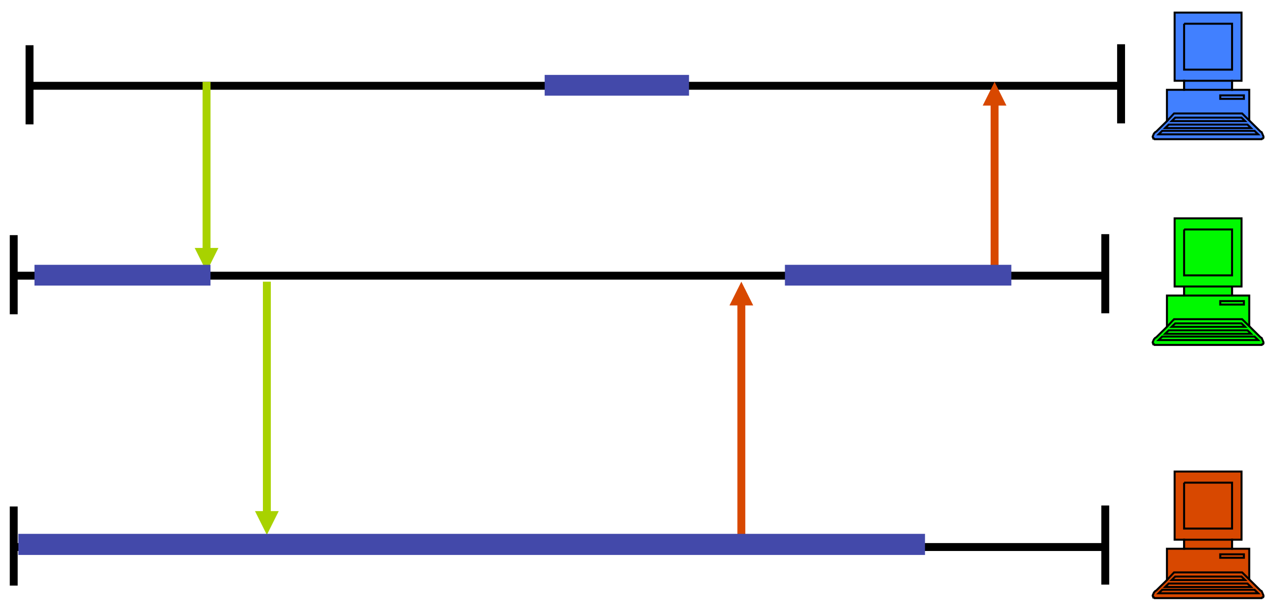
Cache the overlay stack structure



Server side

At each overlay, maintain local pointers to the above and below overlays

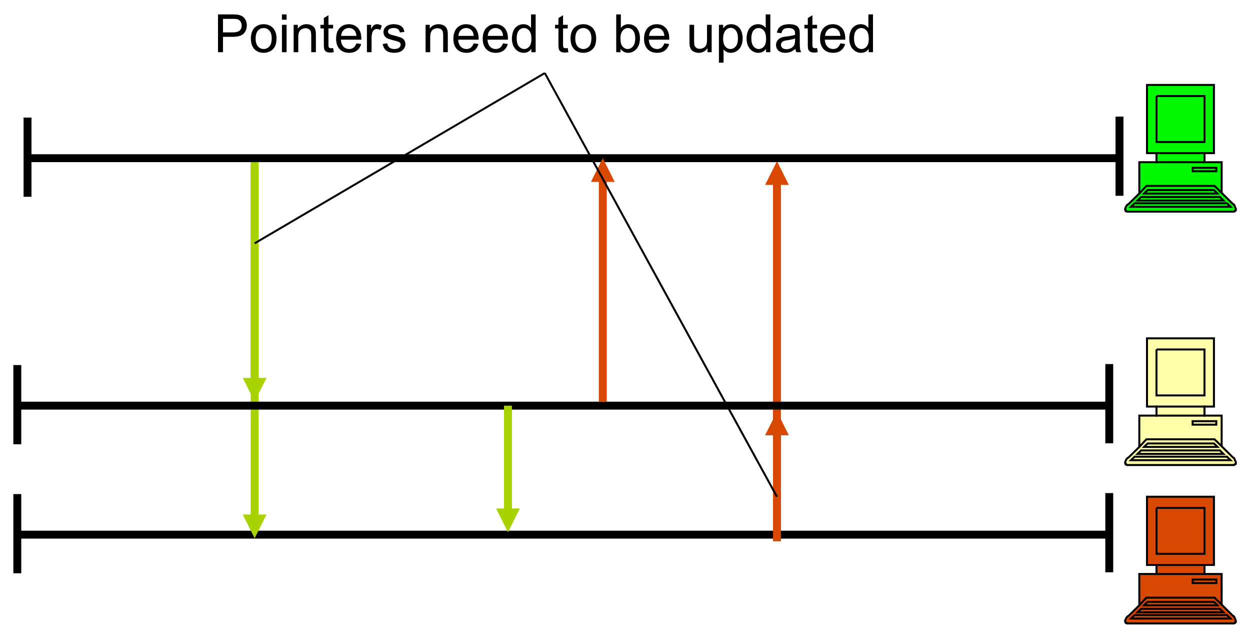
Local pointers are used to redirect R/W



WRITE

Challenges of concurrent overlay operations

Update pointers in CREATE operation

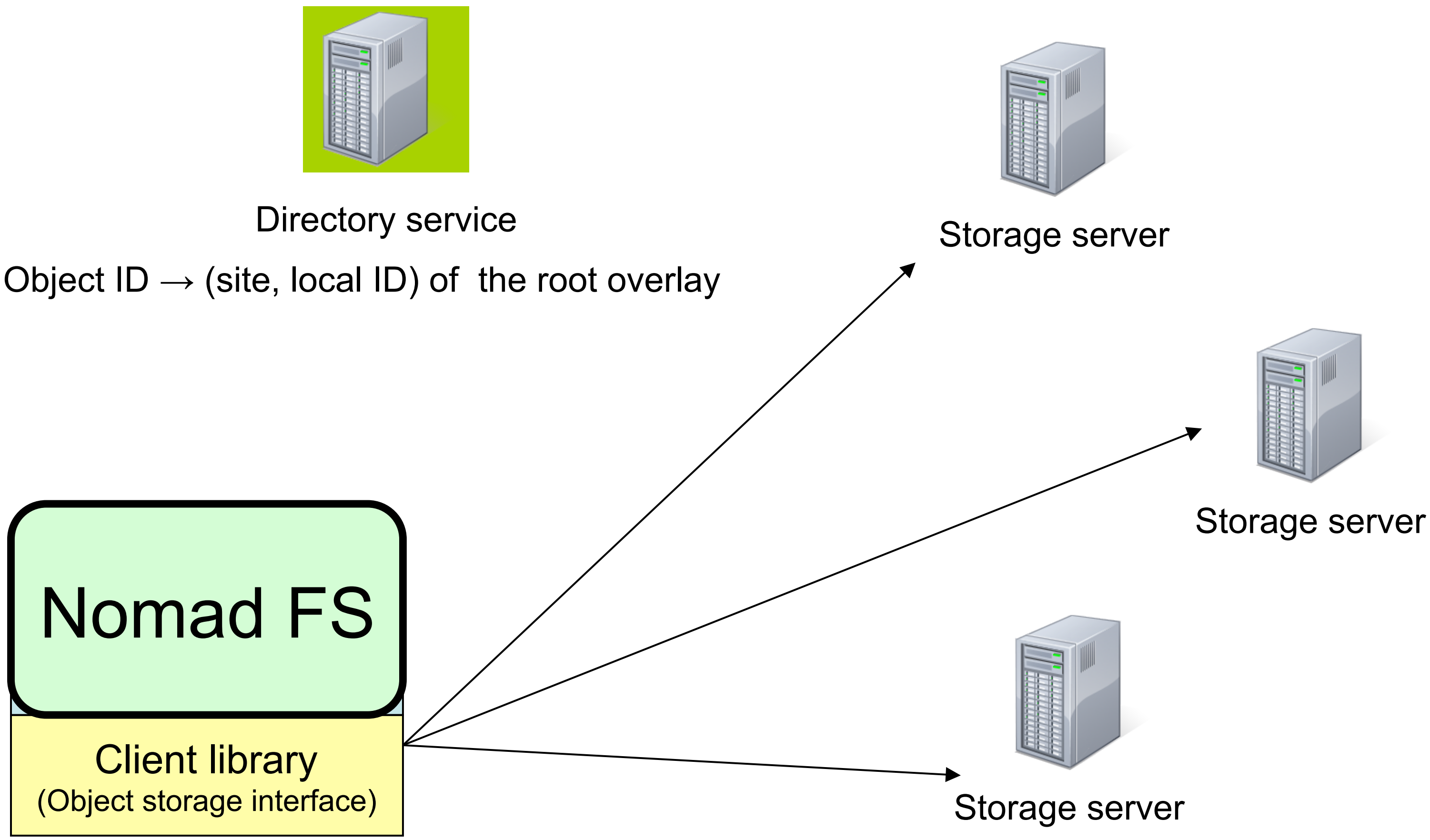


Challenges: pointers are at different machines.
Do we need 2PC? Answer: NO

⇩

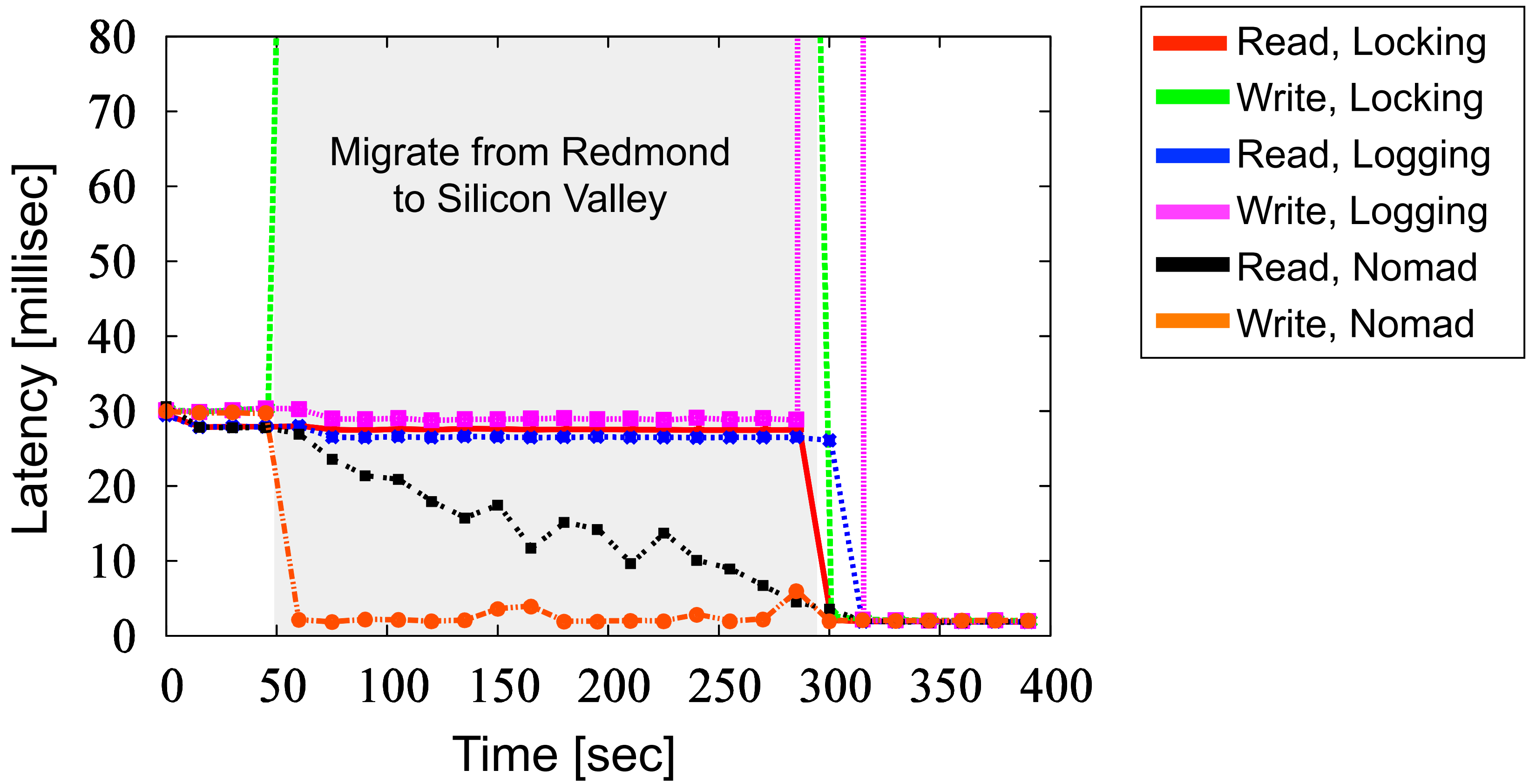
Update order: create pointers at the new overlay before pointers at its parent, before pointer at its child

Nomad's architecture



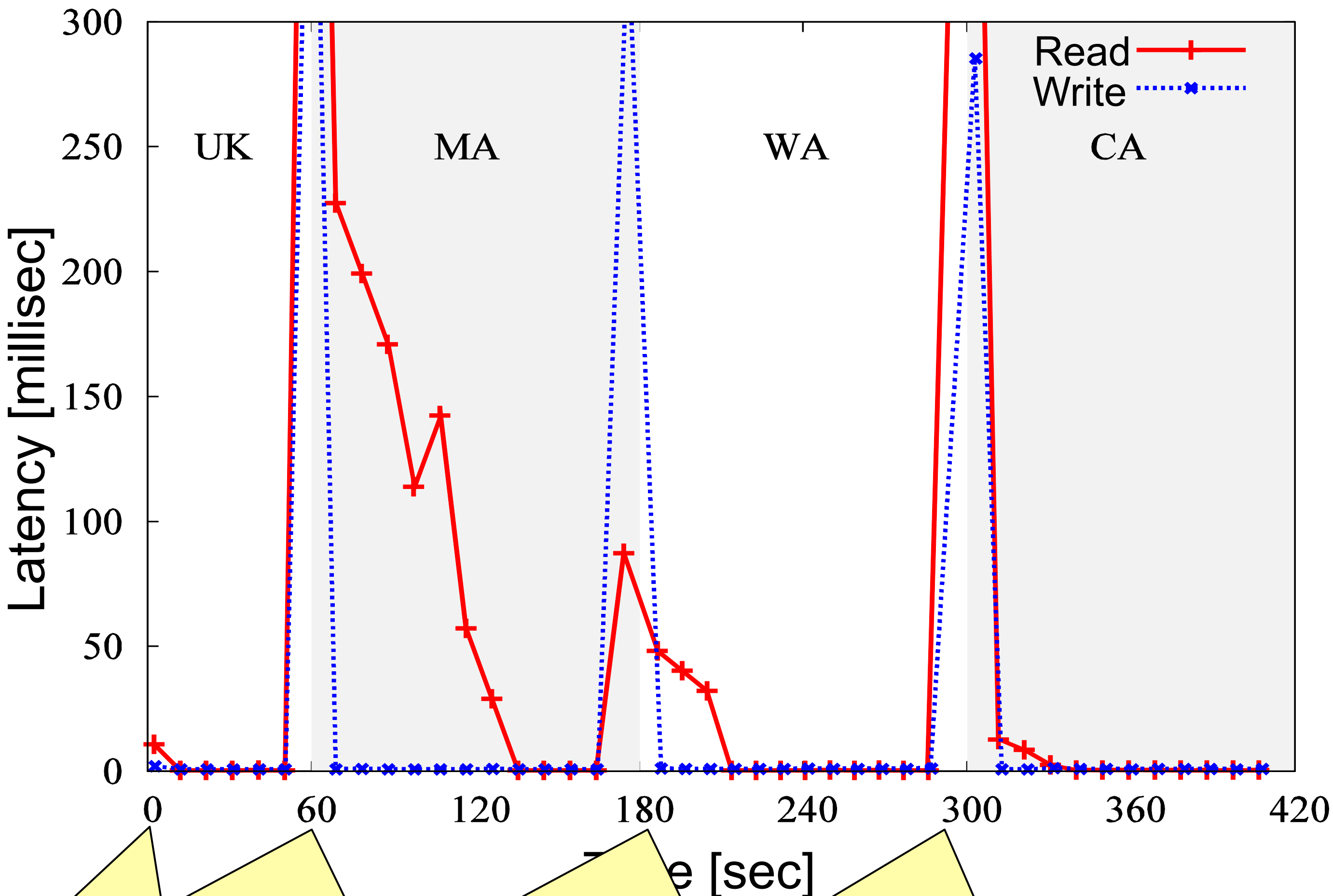
Evaluation of overlays

Comparison with locking, logging



- Object of 50 MB is initially at Redmond
- Client at Silicon Valley issues 1 Read and 1 Write every 200 msec
- Migrate object from Redmond to Silicon Valley after 50 sec

Nomad provides flexible migration



• She moves to
• A cache is crea
• Her data is mig

• She moves to
• Her cache is m
• Her data is stil

• She moves to Mountain View (CA)
• Her cache is migrated from WA to CA
• Her data is still migrated from UK to MA

Policies to drive migration

Migration policies

- Goal
  - Study when and what to migrate based on cost and predicted benefits
  - Web mail application

- Suppose user travels and closest site changes. When should we trigger migration of her data?
- Two simple policies
  - Count policy: # of accesses at new site is $>T_{count}$
  - Time policy: user stays at new site for longer than T_{time}

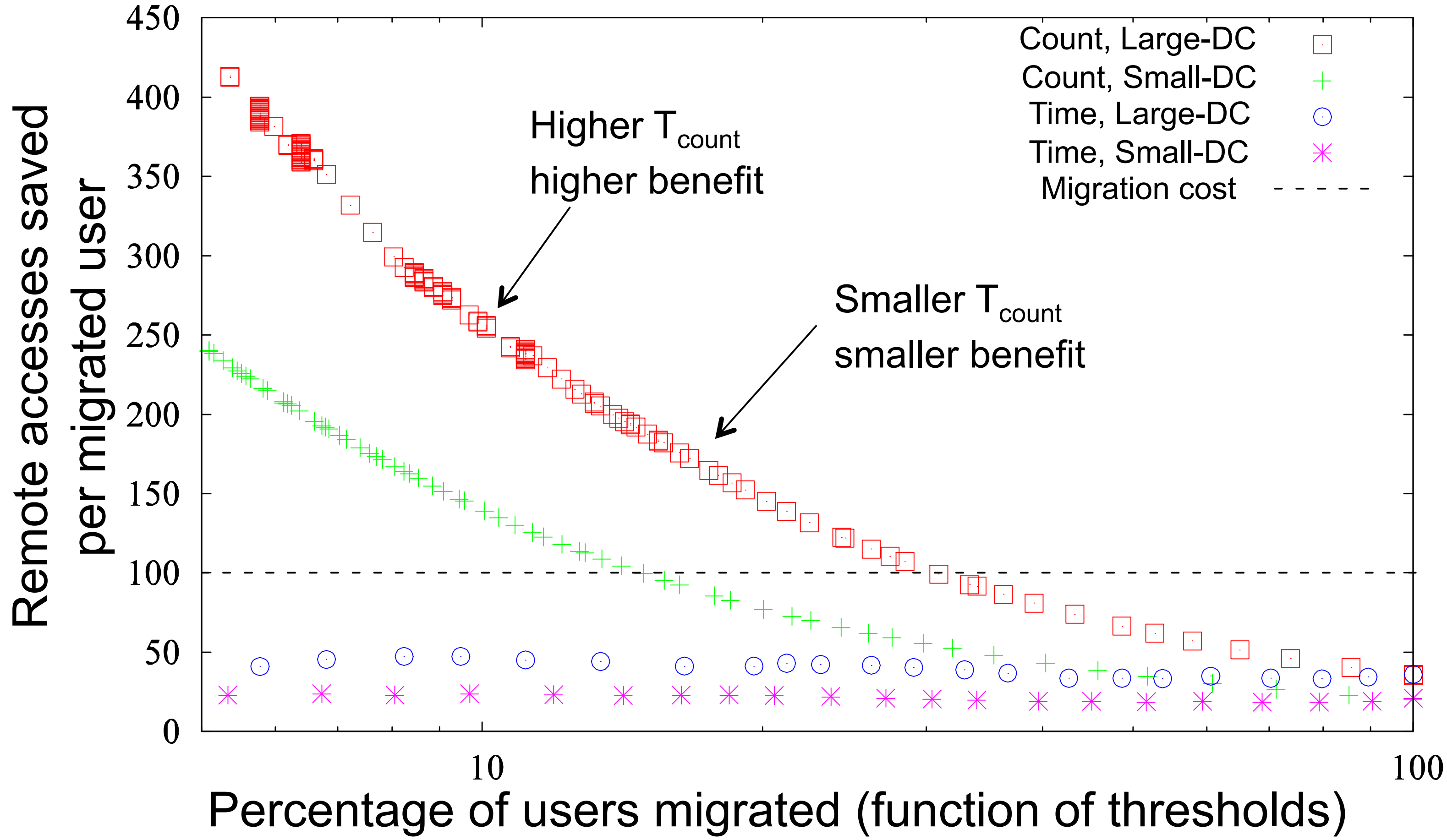
Evaluation of migration policies

- Based on trace of Hotmail usage
  - 50,000 random users (Aug-Sept 2009)
  - Each user: login time, IP address
  - Convert IP addresses to locations

Hotmail users' movement

- User changes sites if she moves more than t miles
- Data center granularity
  - Large-DC: t = 2000 miles
  - Small-DC: t = 450 miles

Count policy vs time policy



- Count policy is better than time policy
- At a given time, migrating data for users with more remote accesses yields more benefits

Summary

- Overlays: mechanism for online migration, caching, and replication
  - More flexible and efficient than prior methods
- Nomad: object storage system with overlays
- Study policies to drive migration for Hotmail application