Does parallelism make lives easier? Intro to Oblivious Parallel RAM

Yue Chen, Yuxuan Wu

CS598DH

Course Advisor: David Heath

Today’s Objectives

- Oblivious RAM revisit
- Problem settings for OPRAM
- From ORAM to OPRAM* (BCP15)
- Improved Paradigm* (CLT15)
- Further improvements (NK16)

Oblivious RAM Revisit

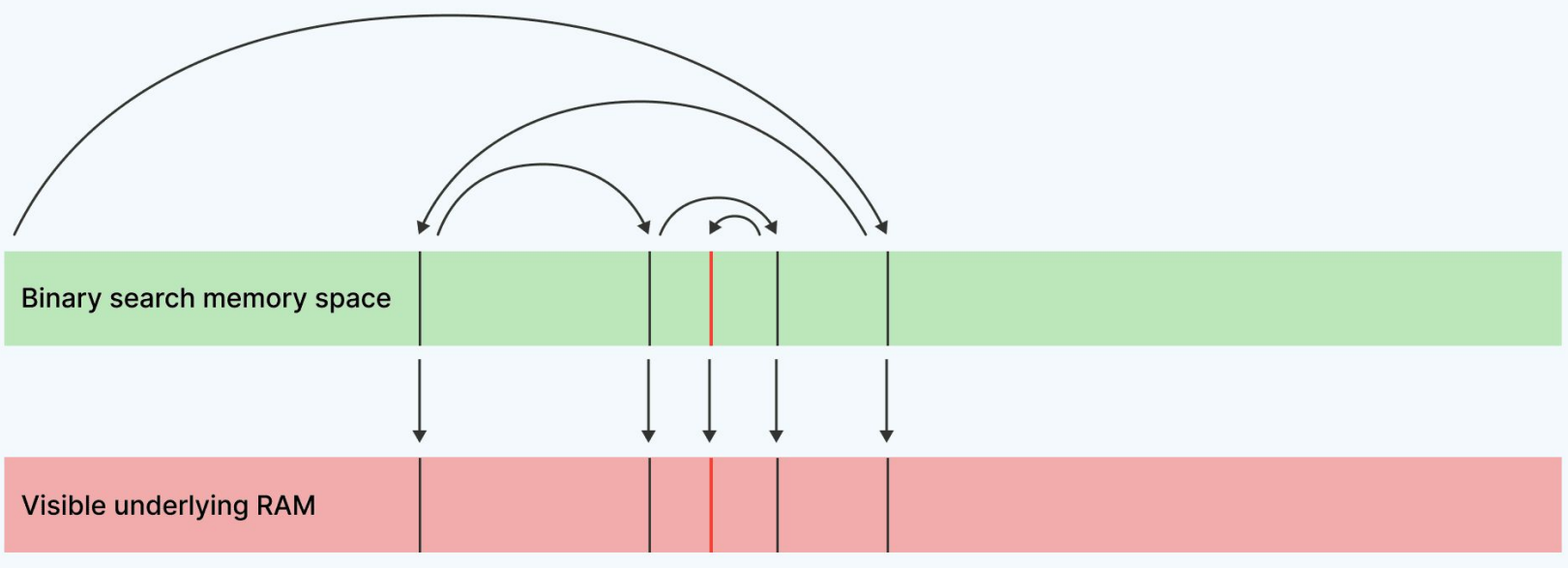
What is ORAM

- A Client(CPU) want to outsource has limited spaces storage
  - Securely to outsource data to an untrusted server.
- Hiding access pattern ⇒ obfuscation.
  - Access pattern leaks information.
- We need some obfuscation.
- Formally: for any two request sequences of **equal length**, the access patterns are **indistinguishable**.

Oblivious RAM Revisit

What is ORAM

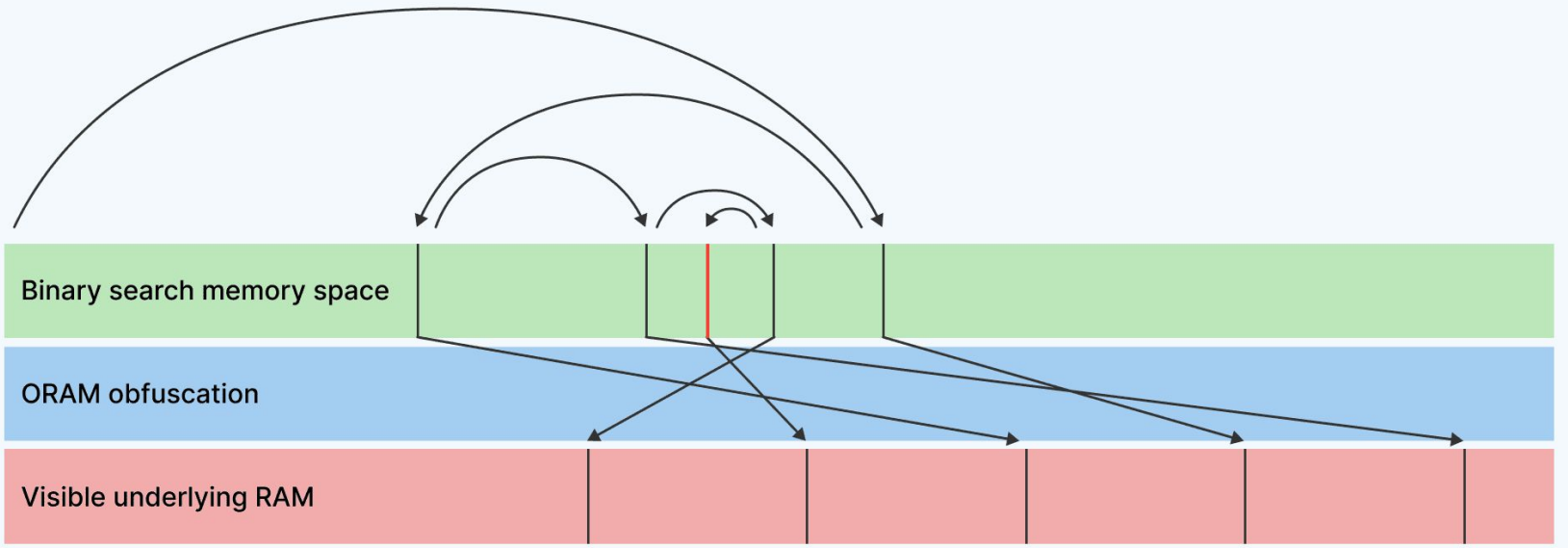
- Clients(CPUs) want to outsource has limited spaces storage
  - Securely to outsource data to an untrusted server.
- Hiding access pattern ⇒ obfuscation.
  - Access pattern leaks information.



Oblivious RAM Revisit

What is ORAM

- Clients(CPUs) want to outsource has limited spaces storage
  - Securely to outsource data to an untrusted server.
- Hiding access pattern ⇒ obfuscation.
  - Access pattern leaks information.



Oblivious RAM Revisit

What is ORAM

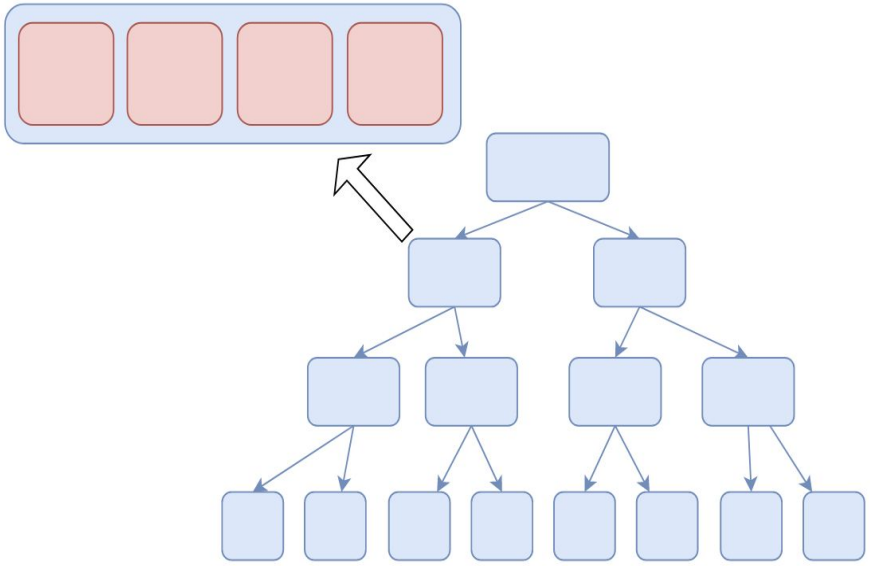
- Clients(CPUs) want to outsource has limited spaces storage.
  - Securely to outsource data to an untrusted server.
- Must hide access pattern.
  - Server guess: the data is sorted.
- We need some obfuscation.
- Formally: for any two request sequences of **equal length**, the access patterns are **indistinguishable**.

Oblivious RAM Revisit

Path ORAM

High-level idea:

- Get paths from position map.
- **Access** all buckets along path.
- Store everything in local stash.
- Erase the accessed cell.
- Randomly assign new path.
- **Flush** the whole stash.



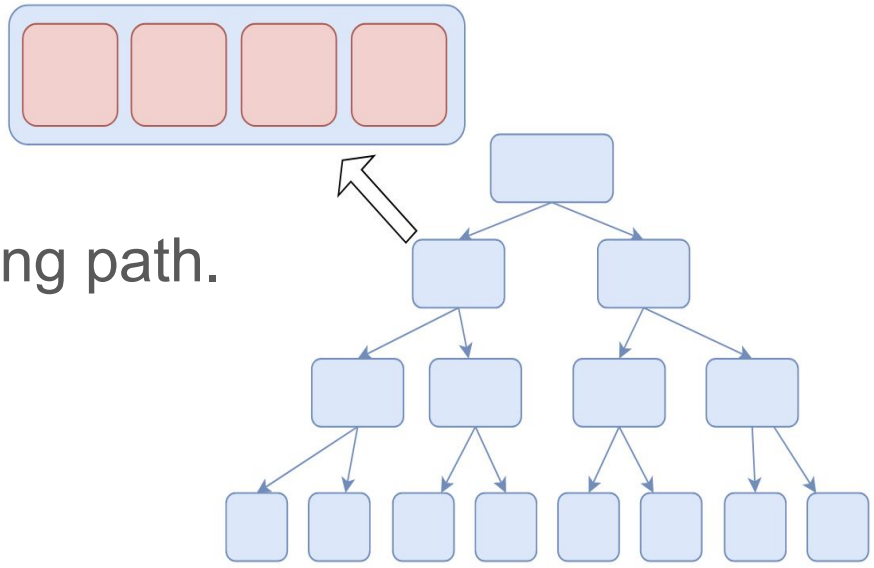
Pos(cell address) → Leaf

Oblivious RAM Revisit

Tree based ORAM

Different Eviction:

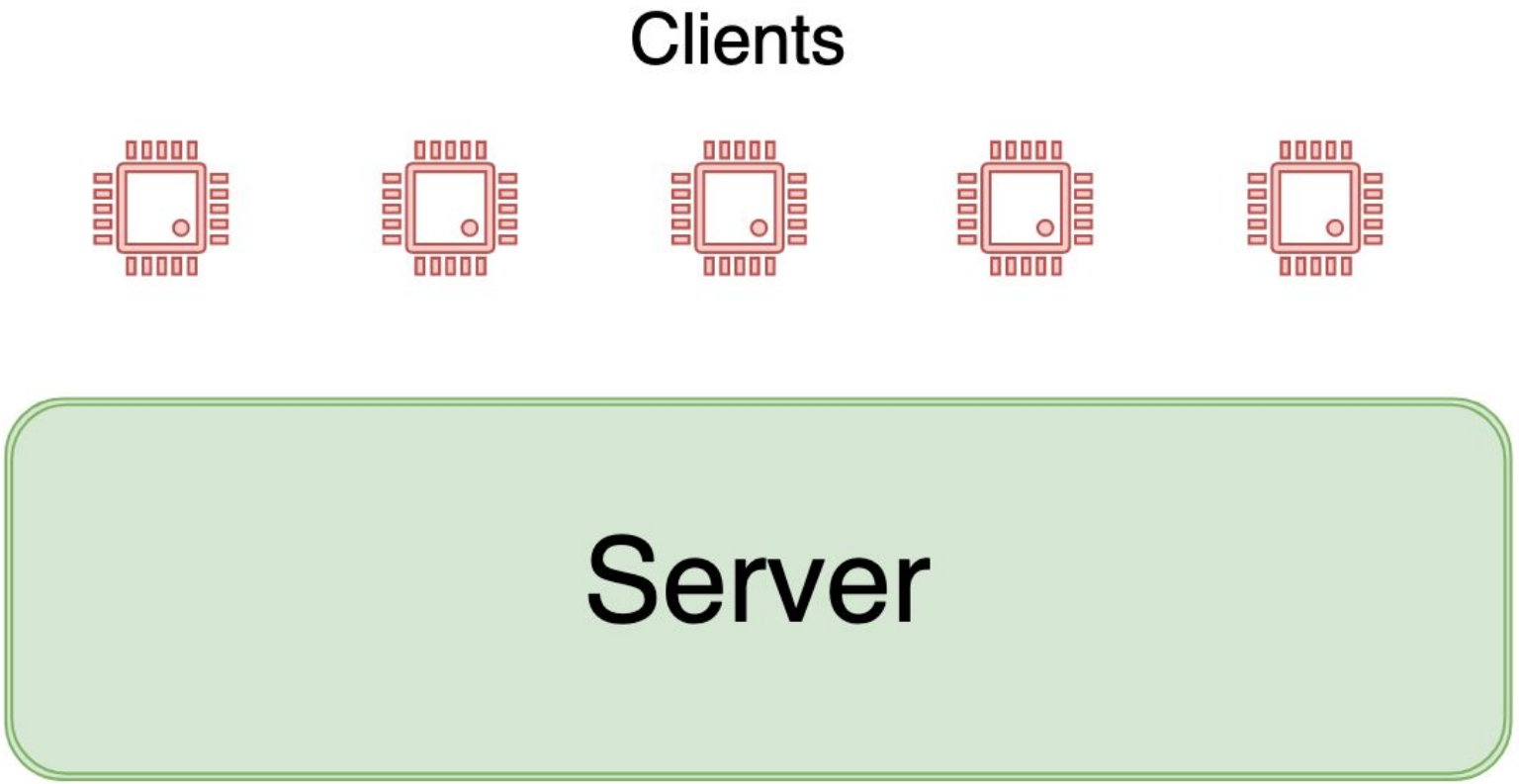
- Get paths from position map.
- **Access** buckets one by one along path.
- ~~Store everything in local stash.~~
- Erase the accessed cell.
- **Put back** data to root.
- Randomly assign new path.
- **Flush** to avoid overflow.



Pos(cell address) → leaf

Problem settings for OPRAM

- Now we have multiple clients who want to access the same Server…



Problem settings for OPRAM

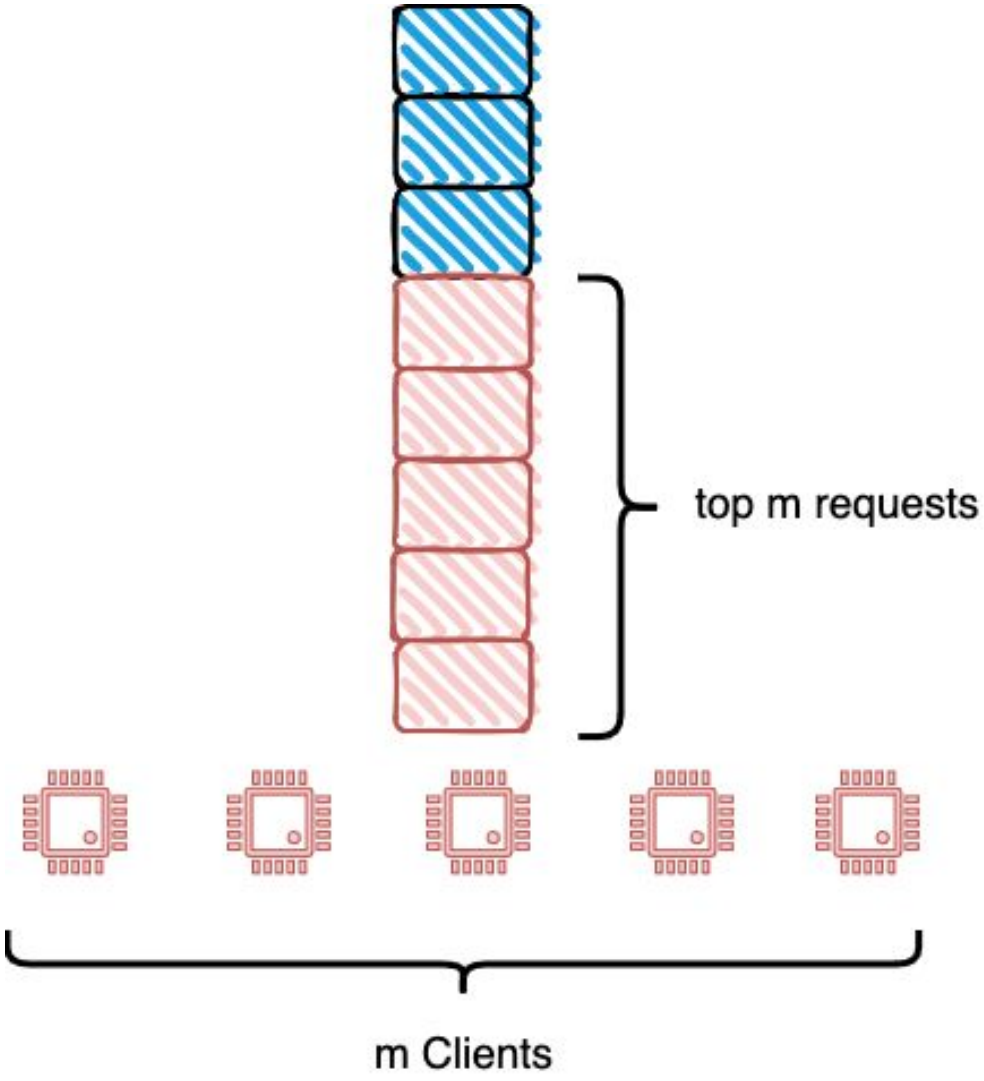
- Clients(CPUs) are mutually trusted
  - Clients are honest
  - Clients themselves don’t initiate requests, instead they coordinate to efficiently process a batch of requests
  - Every single client has access to all the data on the ORAM
  - Clients are able to communicate (in an oblivious manner)

- A good example would be a multi-core CPU trying to access a single untrusted RAM

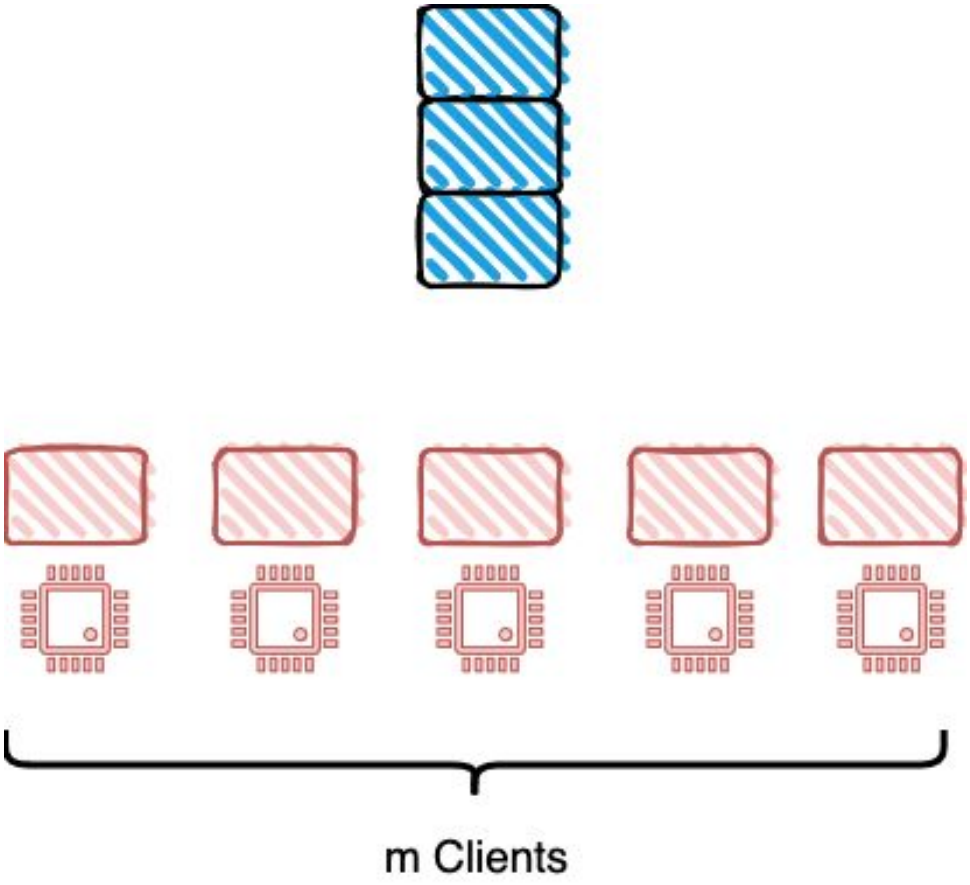
Problem settings for OPRAM

A queue of incoming requests

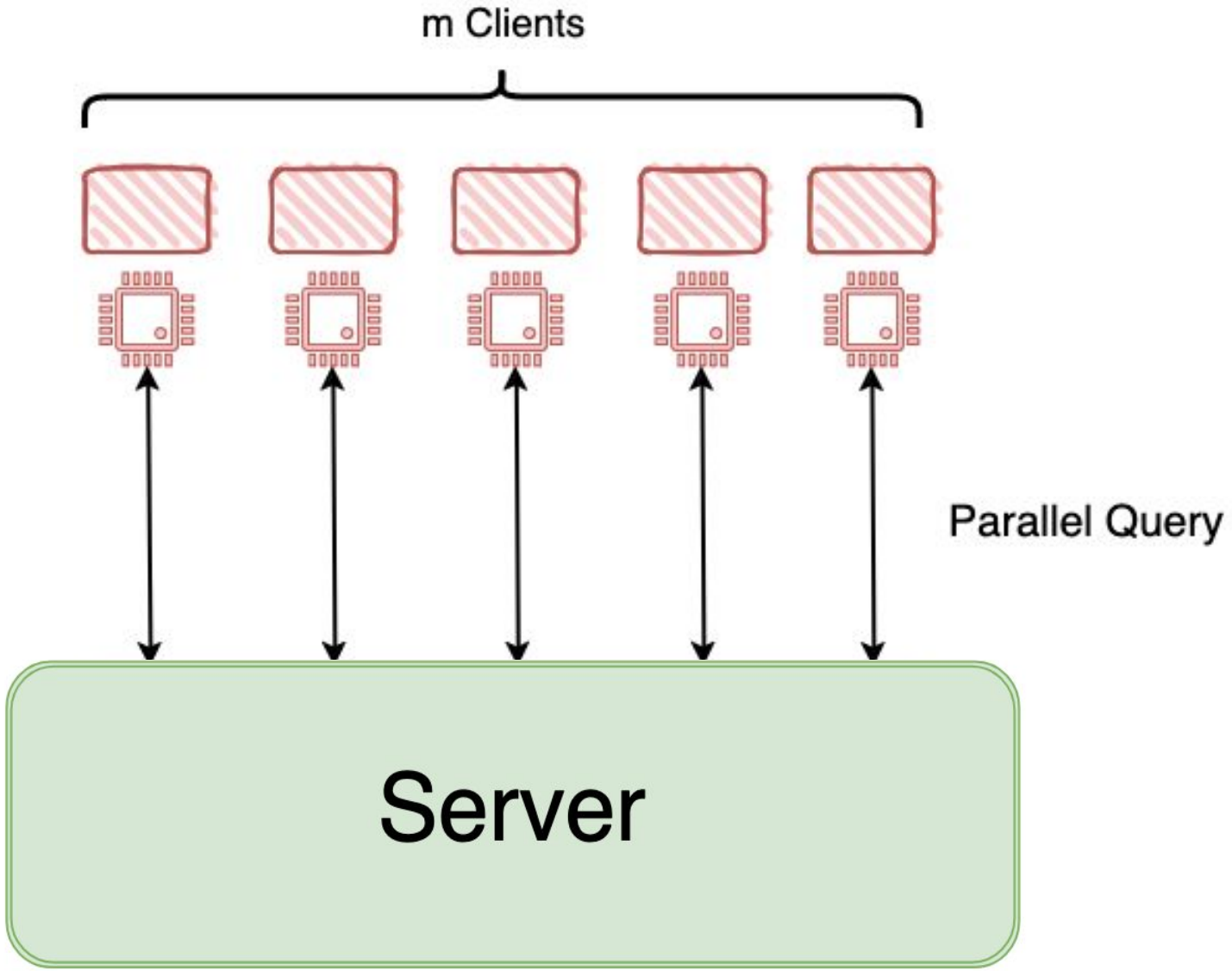
Problem settings for OPRAM



Problem settings for OPRAM

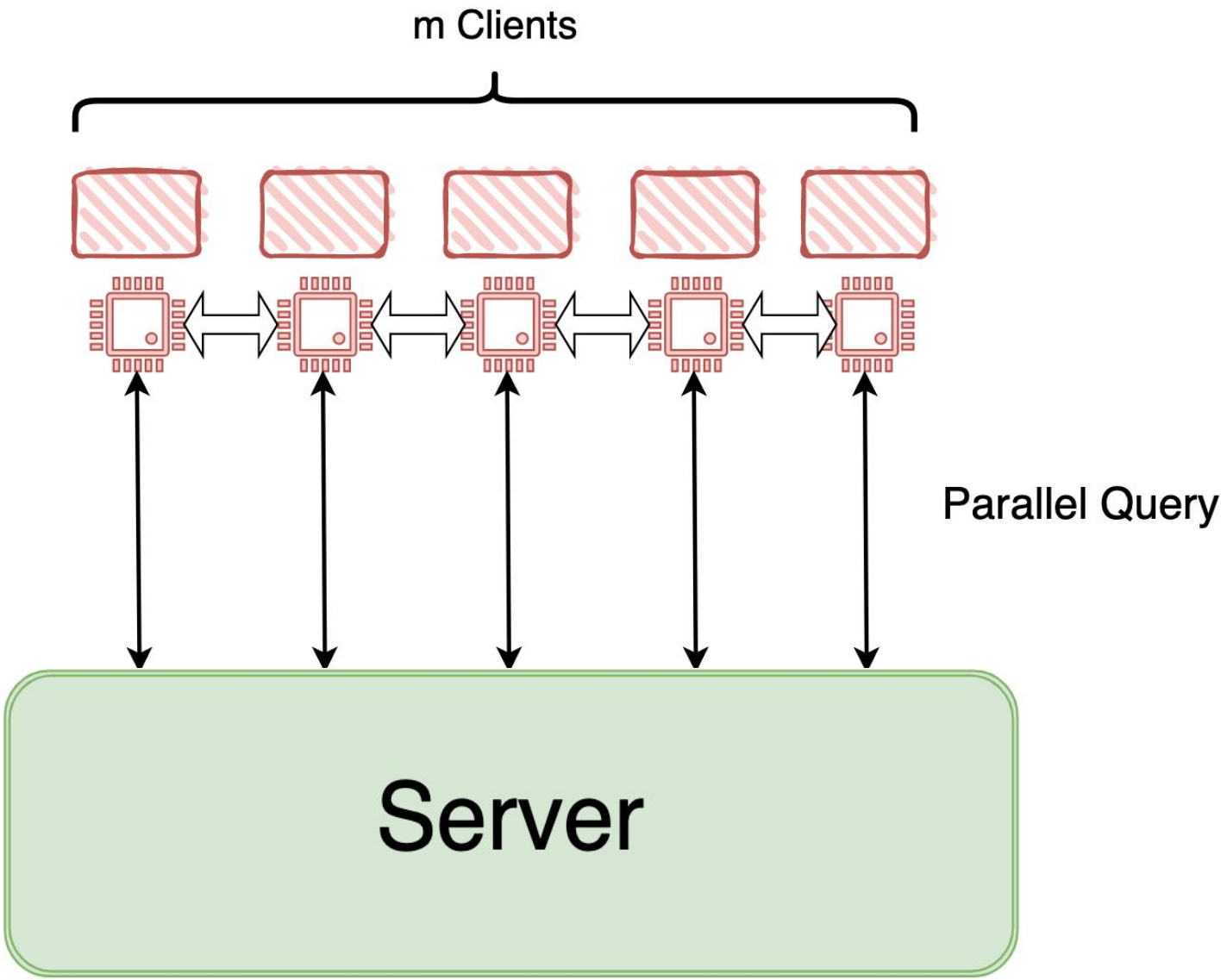


Problem settings for OPRAM



Problem settings for OPRAM

Clients are allowed to communicate



Problem settings for OPRAM

- Clients(CPUs) are mutually trusted
  - Clients are honest
  - Clients are able to communicate (in an oblivious manner)
  - Clients themselves don’t initiate requests, instead they coordinate efficiently to process a batch of requests
  - Every single client has access to all the data on the ORAM
- Same to ORAM, data is hosted on a single, untrusted server
  - Data is encrypted
  - Data accesses should appear oblivious and can be simulated
  - Any data retrieved will be encrypted differently when put back, so server won’t know whether it remains untouched or it has been moved to another place.

From ORAM to OPRAM (BCP15)

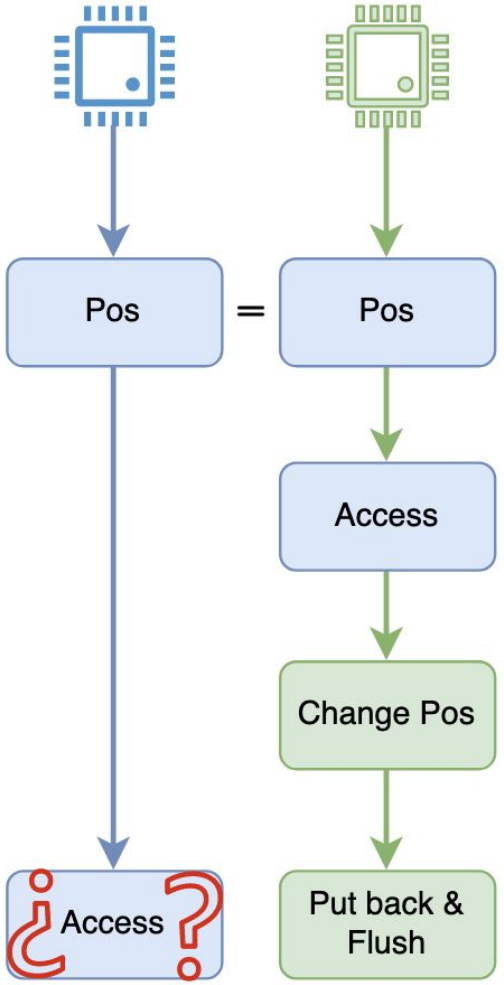
Elette Boyle, Kai-Min Chung, Rafael Pass

- Accessing on the same server ⇔ collision
  - Time collision
  - Data collision
- Need to be fast.
  - Faster than use ORAM sequentially.
  - BCP15: $O\left(\log(m)\log^3(n)\right)$

From ORAM to OPRAM (BCP15)

Time Collision

- What if we allow clients to access data freely?
- They should always stay in the same phase.
- Do things step by step (foreshadow)

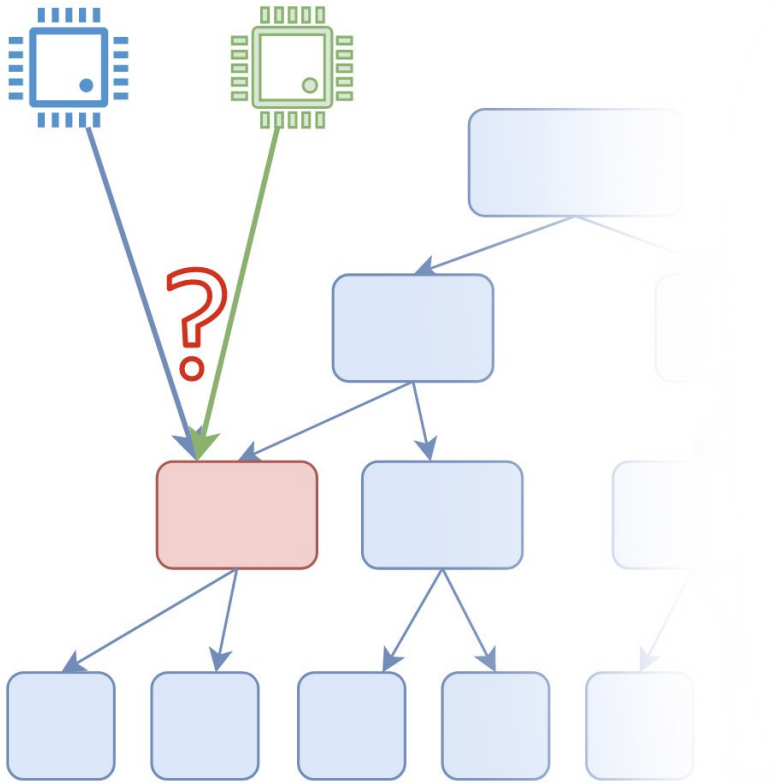


From ORAM to OPRAM (BCP15)

Data Collision

High-level idea (Non-recursive):

- **Resolve** tasks conflicts on buckets.
- Read/update position map.
- Access paths in parallel: read only
- Update Buckets.
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.



Choose representative (**Election**)

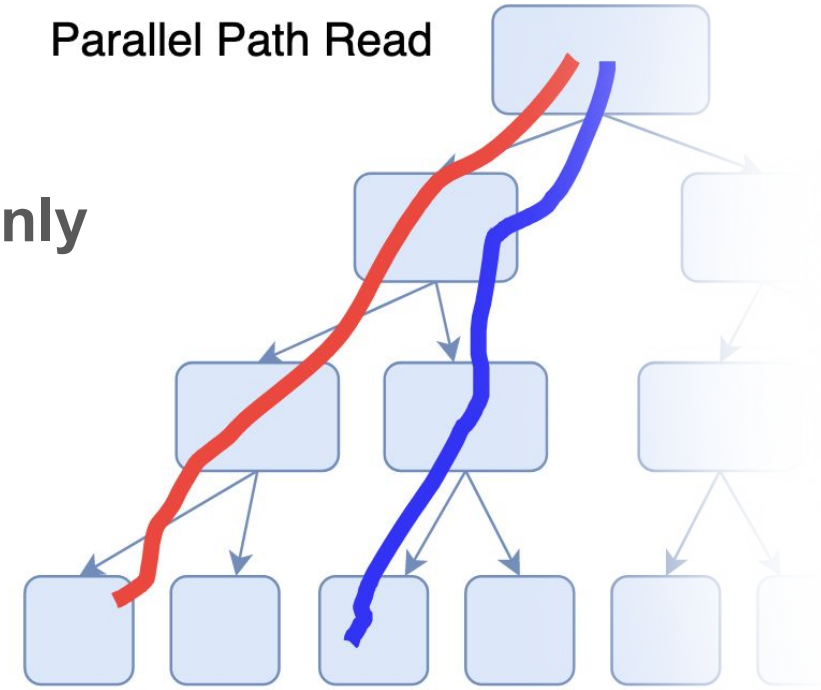
Broadcast & combine data (**Aggregation**)

From ORAM to OPRAM (BCP15)

Data Collision

High-level idea (Non-recursive):

- **Resolve** tasks conflicts.
- Read/update position map.
- **Access paths in parallel: read only**
- Update Buckets.
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.

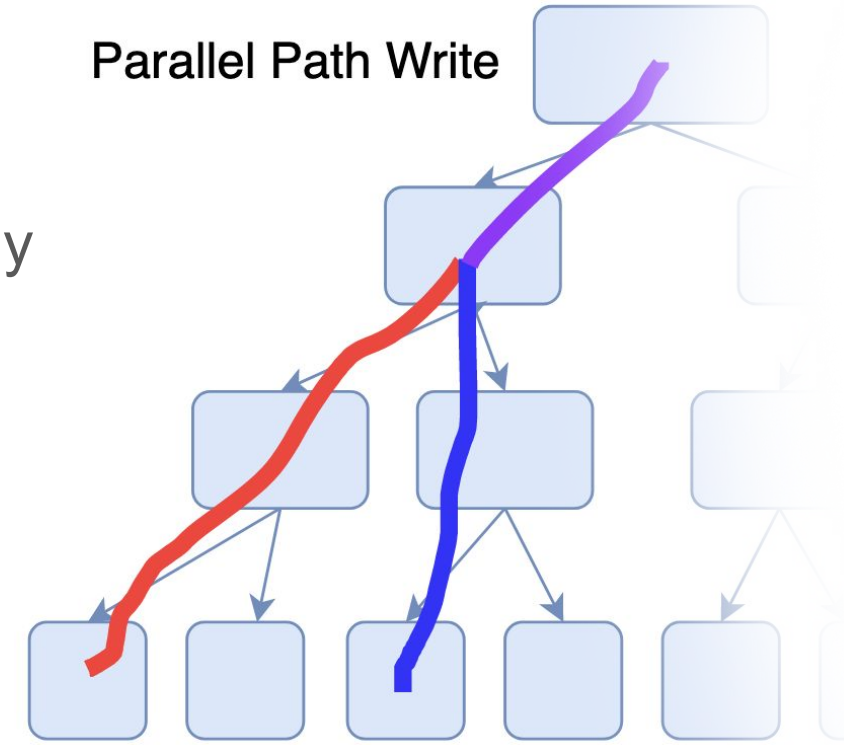


From ORAM to OPRAM (BCP15)

Data Collision

High-level idea (Non-recursive):

- **Resolve** tasks conflicts.
- Read/update position map.
- Access paths in parallel: read only
- **Update Buckets.**
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.

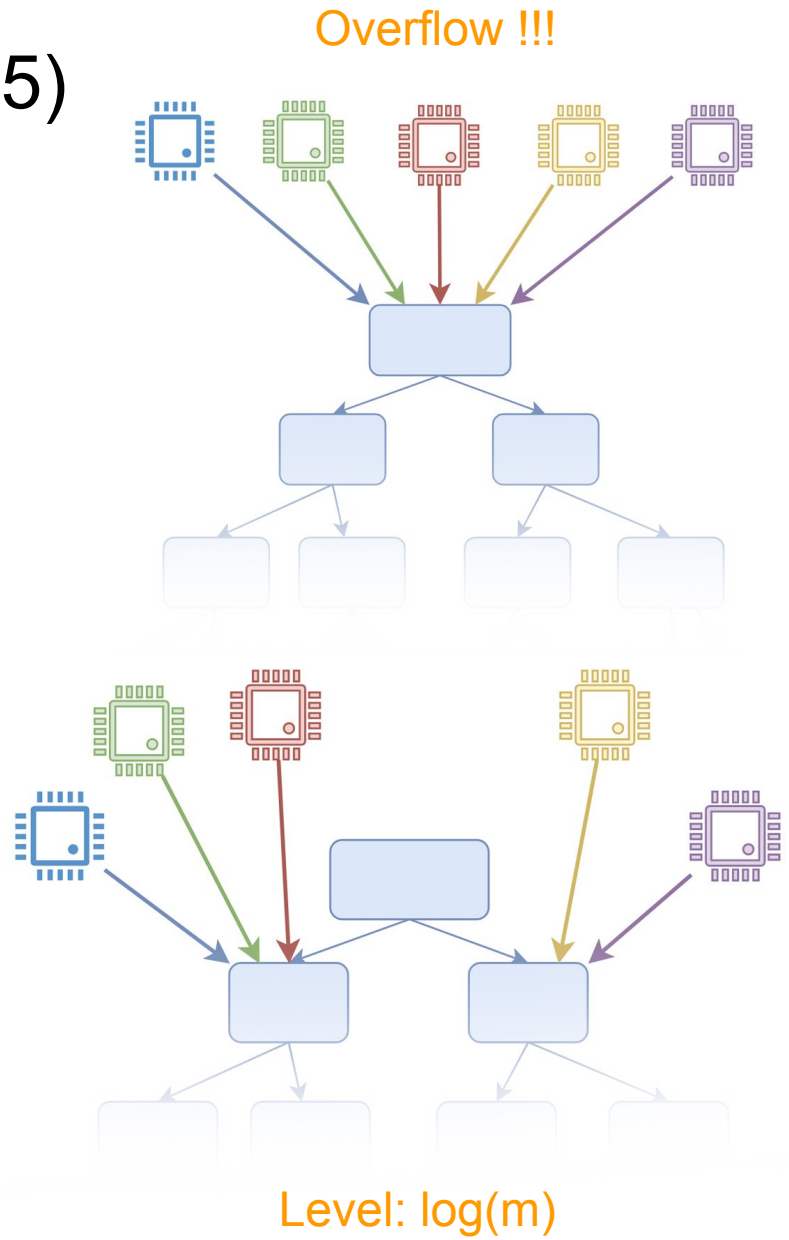


From ORAM to OPRAM (BCP15)

Data Collision

High-level idea (Non-recursive):

- **Resolve** tasks conflicts.
- Read/update position map.
- Access paths in parallel: read only
- Update Buckets.
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.

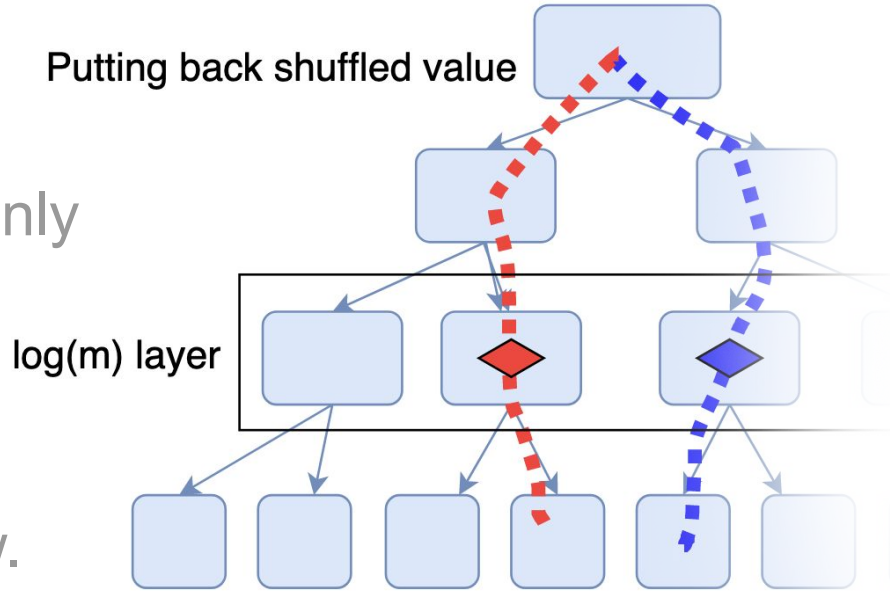


From ORAM to OPRAM (BCP15)

Data Collision

High-level idea (Non-recursive):

- **Resolve** tasks conflicts.
- Read/update position map.
- Access paths in parallel: read only
- Update Buckets.
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.



From ORAM to OPRAM (BCP15)

Data Collision

High-level idea (Non-recursive):

- **Resolve** tasks conflicts.
- Read/update position map.
- Access paths in parallel, Read.
- Update Buckets.
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.

Flush may also conflict.

Communication required.

Independent ⇒ naturally oblivious.

From ORAM to OPRAM (BCP15)

Data Collision

High-level idea:

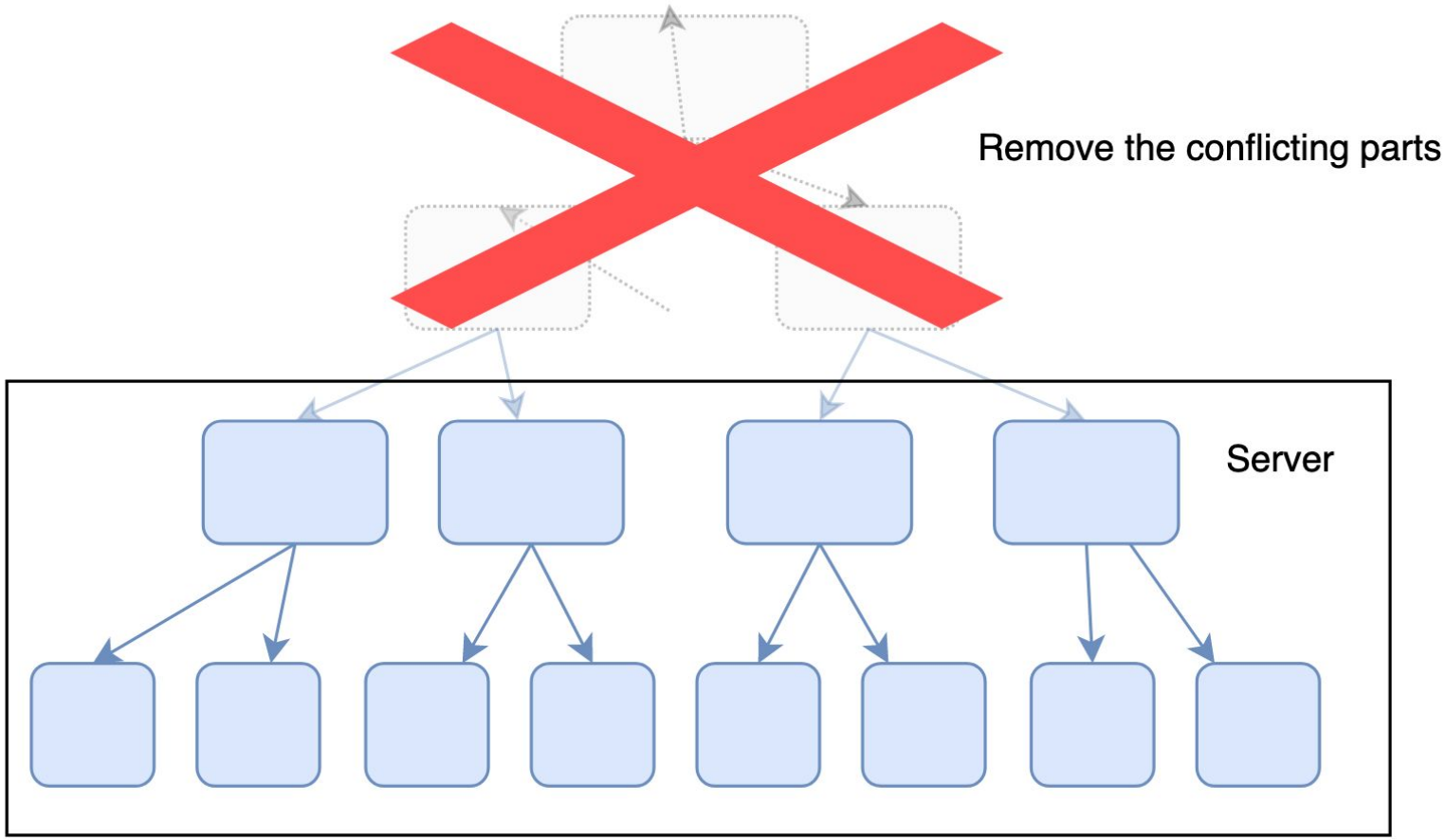
- **Resolve** tasks conflicts.
- Read/update position map ⇐ actually a recursion.
- Access paths in parallel, Read.
- **Update Buckets.**
- **Parallel insert** to lower level.
- **Parallel flush** to avoid overflow.

Many conflicts, many communications !!!

Improved Paradigm (CLT15)

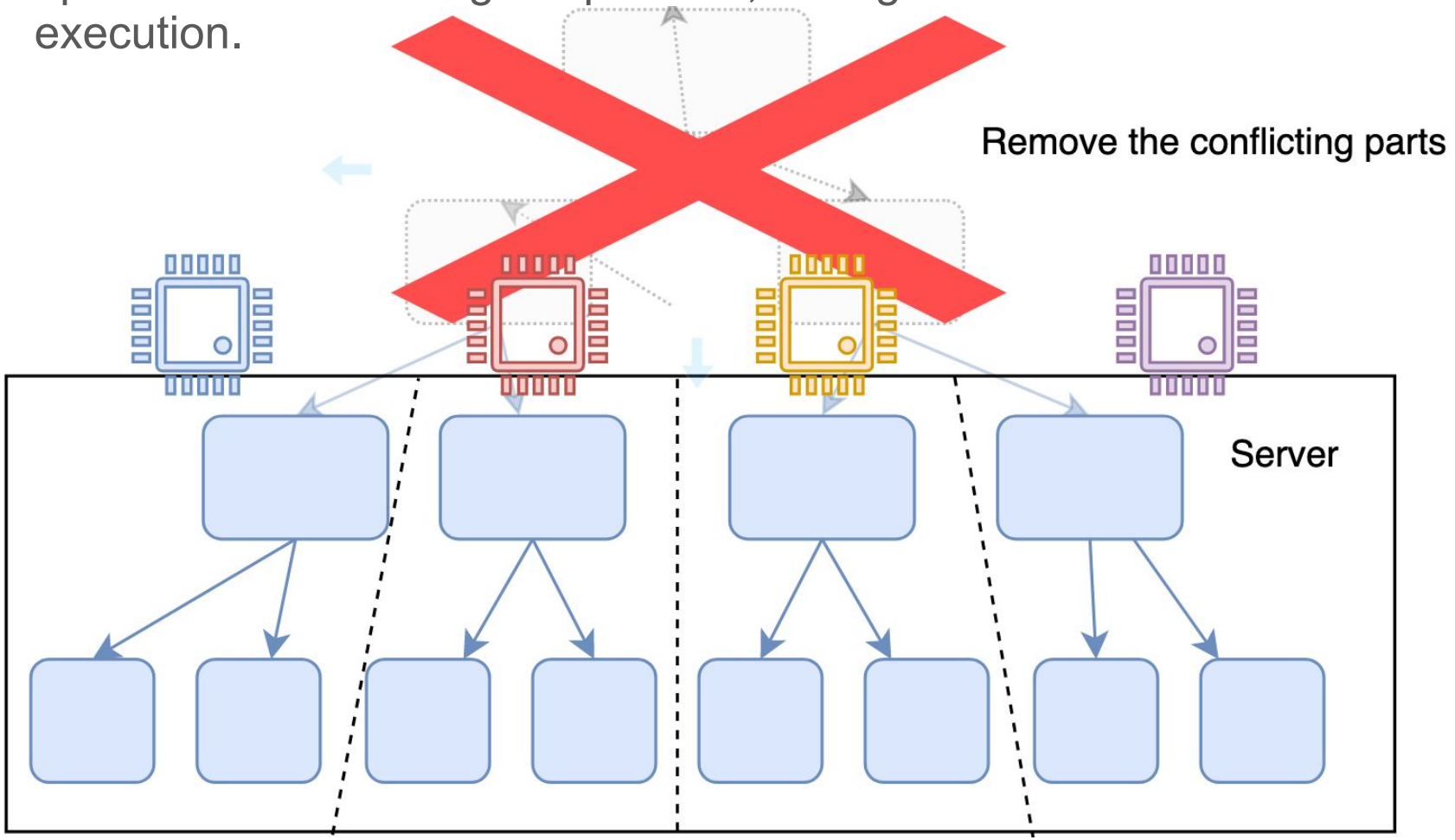
Binyi Chen, Huijia Lin, Stefano Tessaro

- Improved upon BCP scheme
- Noticed that clients only put data below log(m) level
- The upper levels are untouched and can actually be removed!

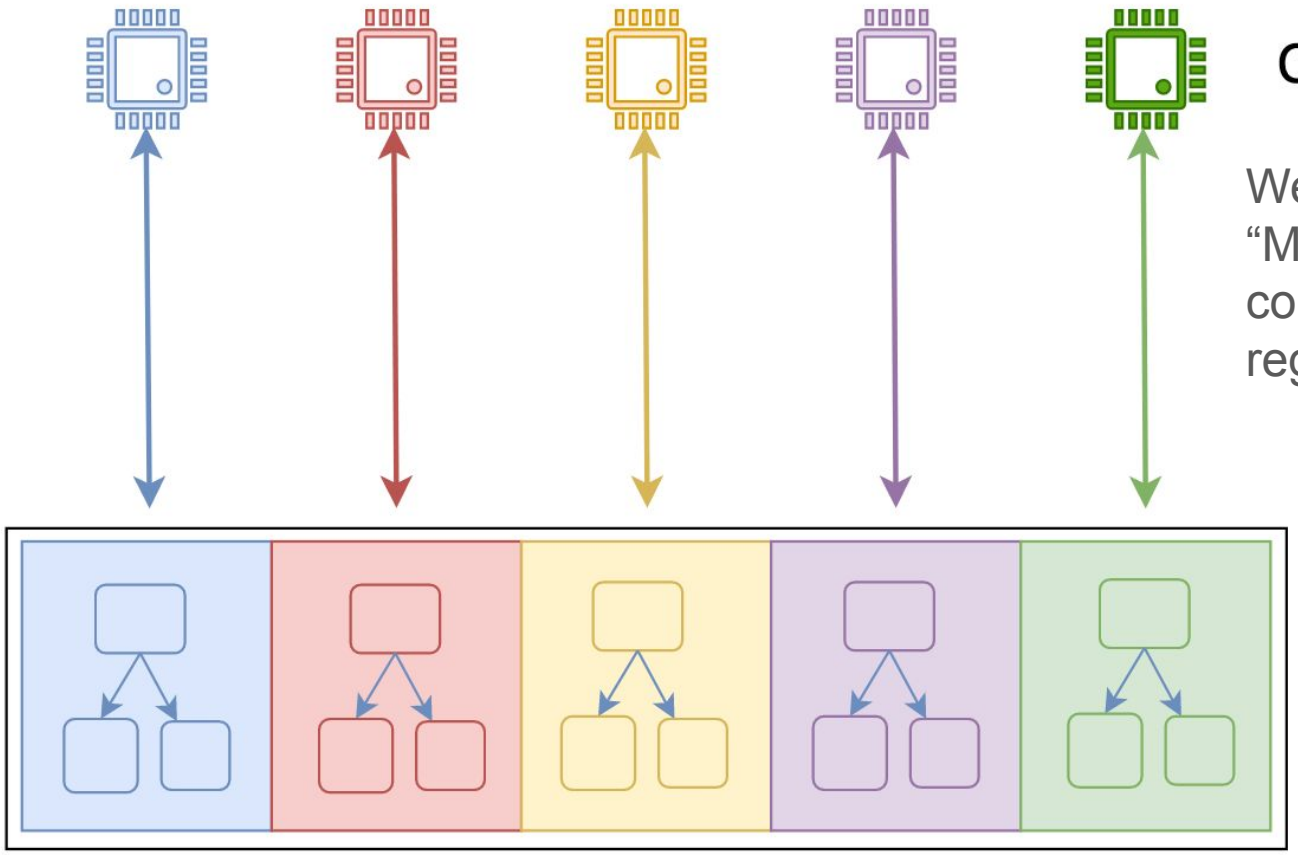


Improved Paradigm (CLT15): Ownership Partition

- Access control - each client is responsible for all the w/r operations to the assigned partition, throughout the entire execution.

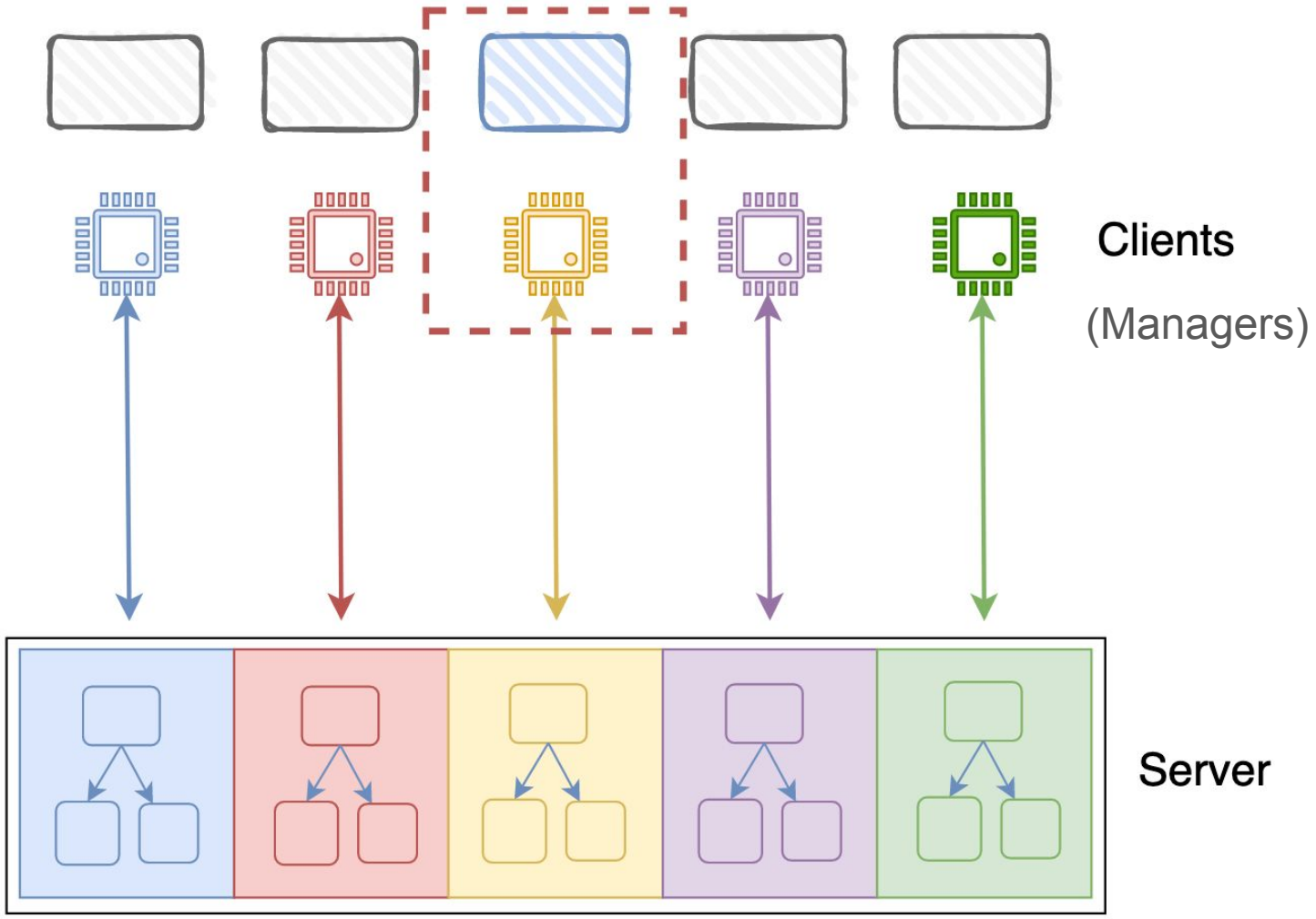


Improved Paradigm (CLT15): Ownership Partition

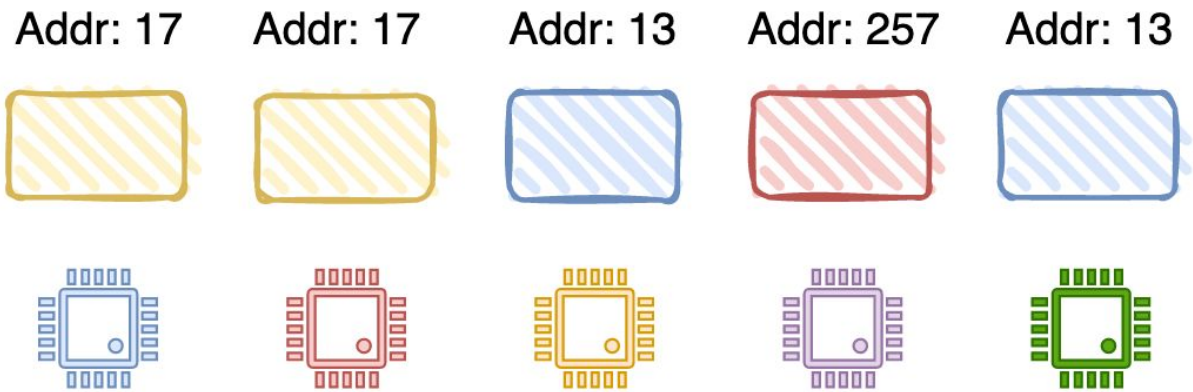


Improved Paradigm (CLT15)

If a client wants to access a cell in a partition that it doesn't own, forward the request to partition owner



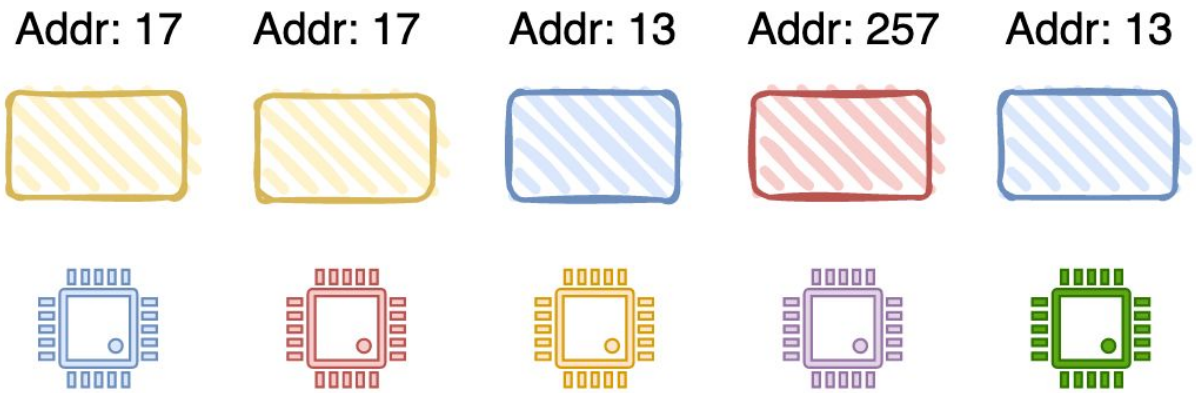
Improved Paradigm (CLT15)



Addr 9	Addr 13	Addr 17	Addr 257
leaf 106	leaf 292	leaf 461	leaf 313

Position Map is shared globally

Improved Paradigm (CLT15)



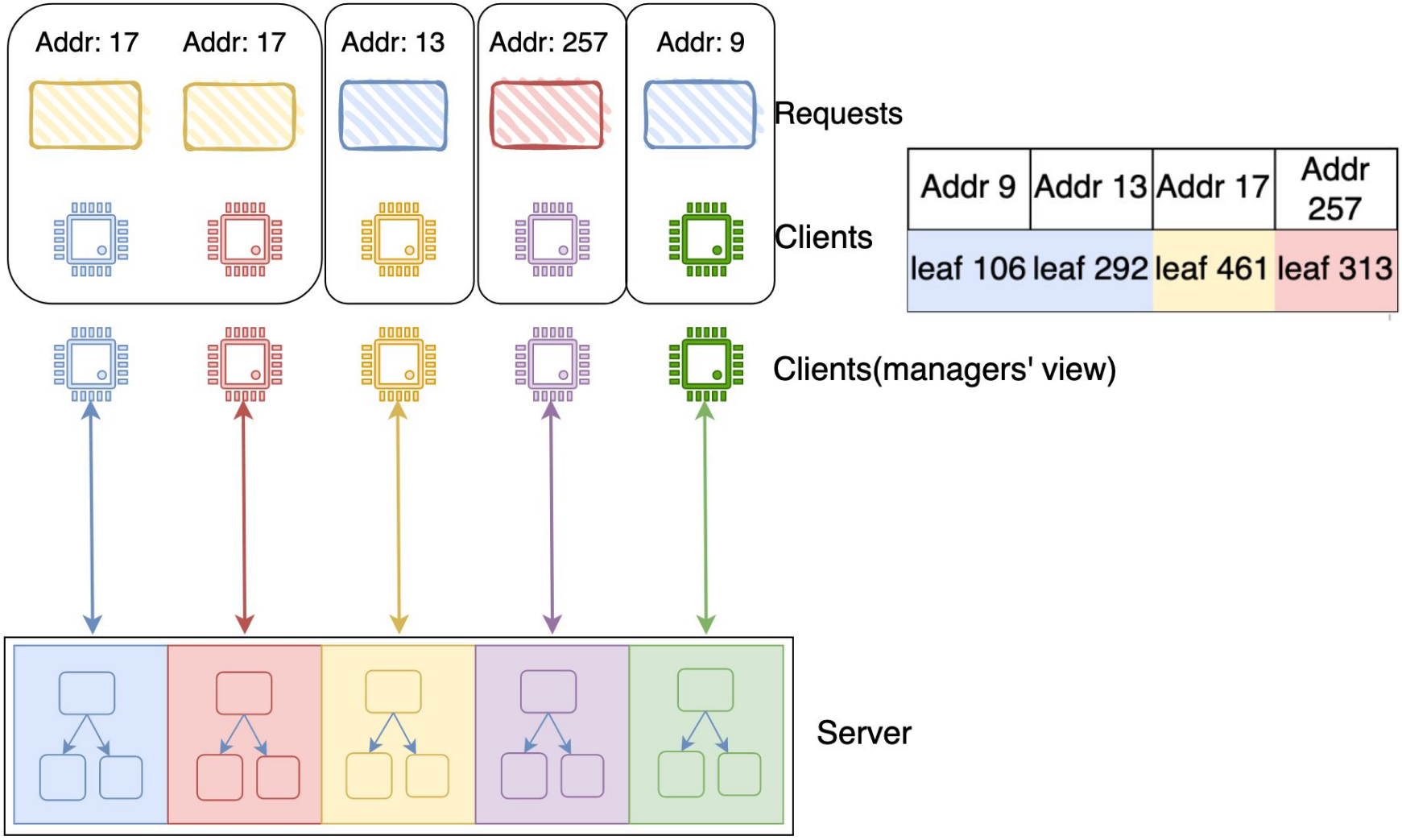
Addr 9	Addr 13	Addr 17	Addr 257
leaf 106	leaf 292	leaf 461	leaf 313

Position Map is shared among clients

Every client knows where to forward the request

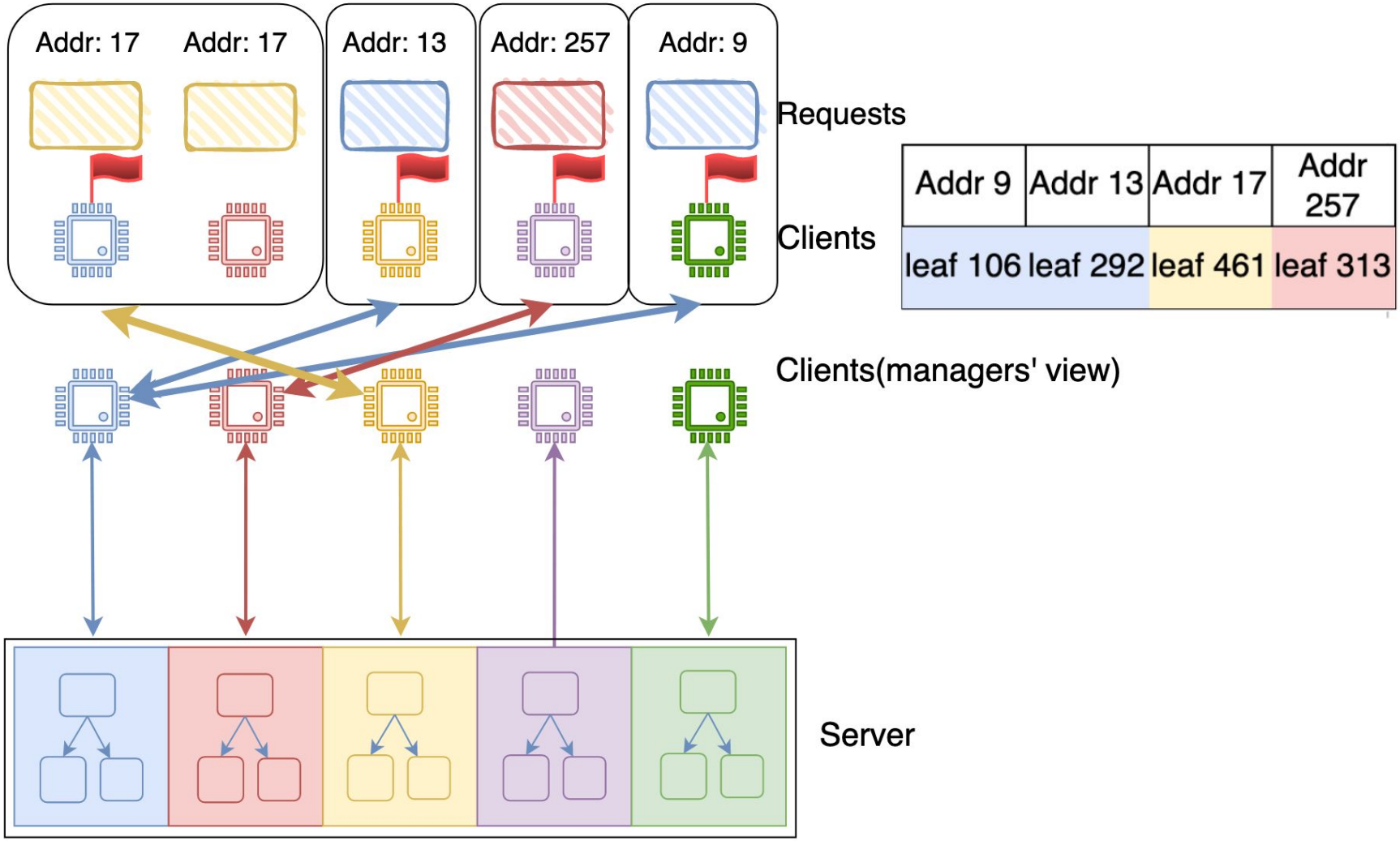
Improved Paradigm (CLT15)

ObliviousElection:
Clients accessing the same address will use the address as the key, to find a representative that does the query



Improved Paradigm (CLT15)

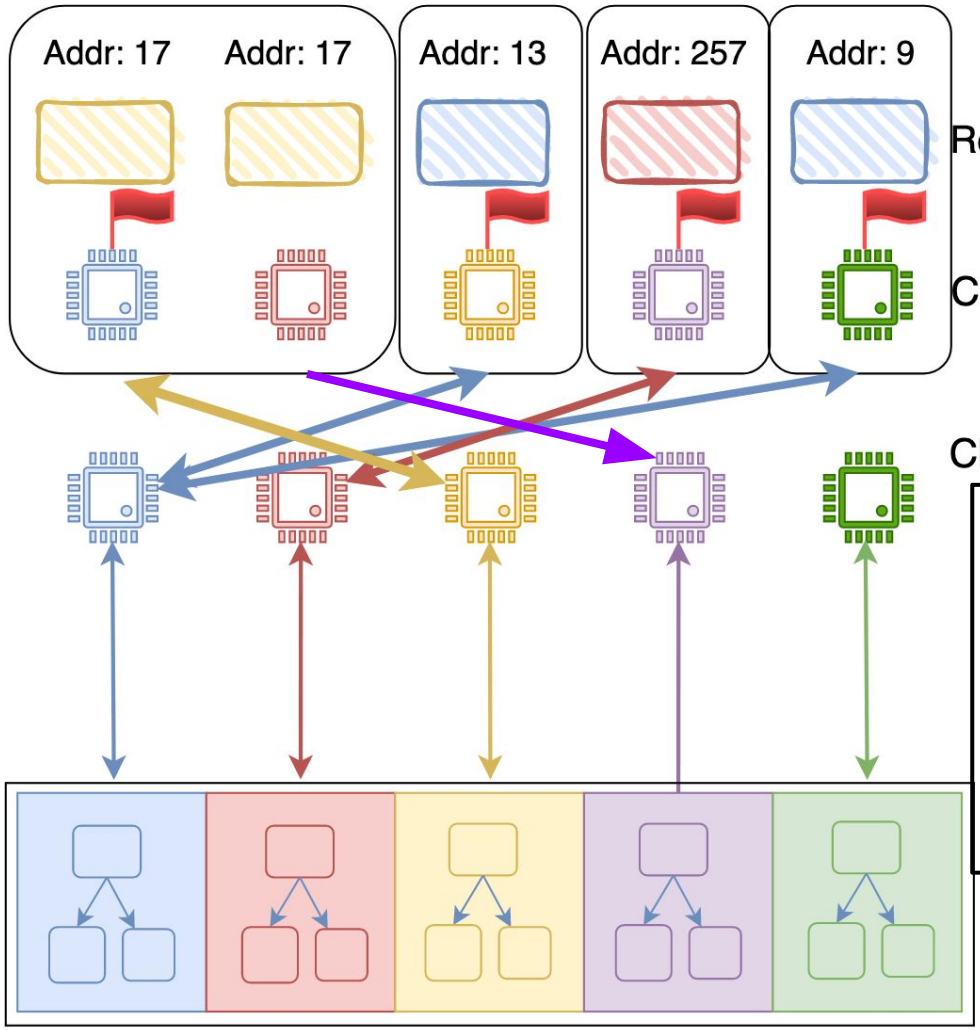
ObliviousElection:
Clients accessing the same address will use the address as the key, to find a representative that does the query



Addr 9	Addr 13	Addr 17	Addr 257
leaf 106	leaf 292	leaf 461	leaf 313

Improved Paradigm (CLT15)

ObliviousElection:
Clients accessing the same address will use the address as the key, to find a representative that does the query



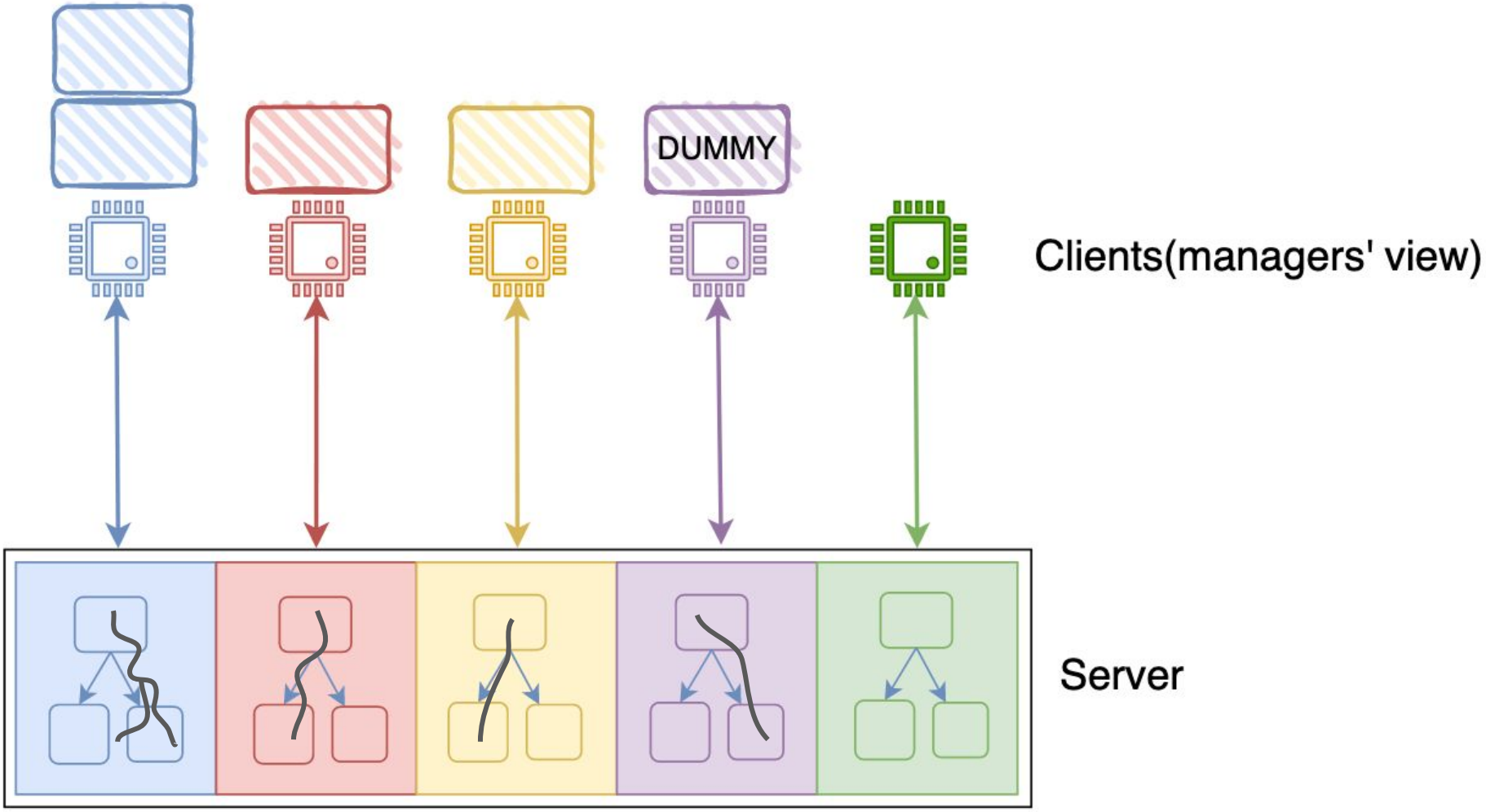
Addr 9	Addr 13	Addr 17	Addr 257
leaf 106	leaf 292	leaf 461	leaf 313

The server can observe a missing query(expecting 5)

Thus, non-representatives send out a **dummy request** to a random manager(could be itself)

Improved Paradigm (CLT15)

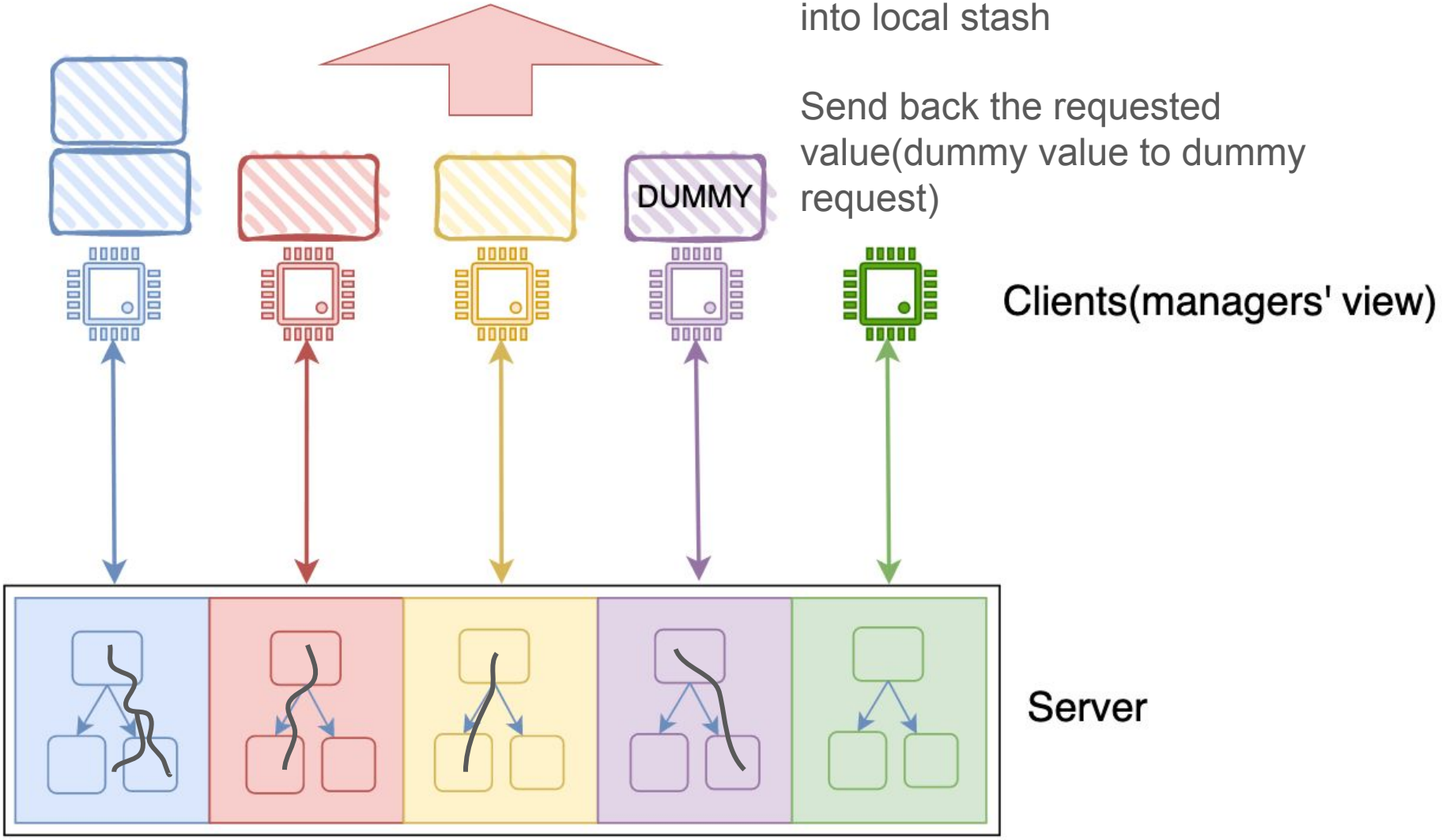
Read requested paths and dump data into local stash



Improved Paradigm (CLT15)

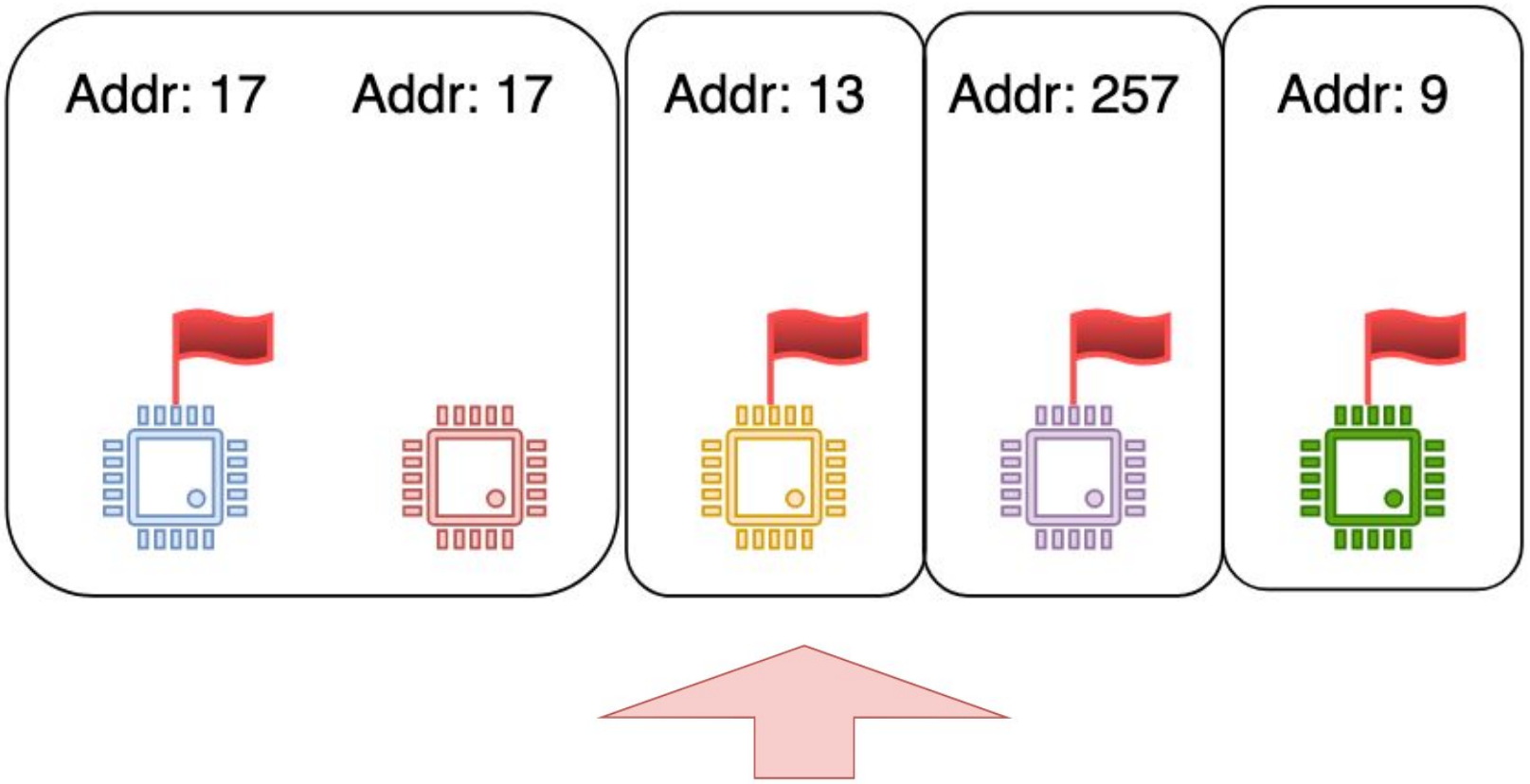
Read requested paths and dump data into local stash

Send back the requested value(dummy value to dummy request)



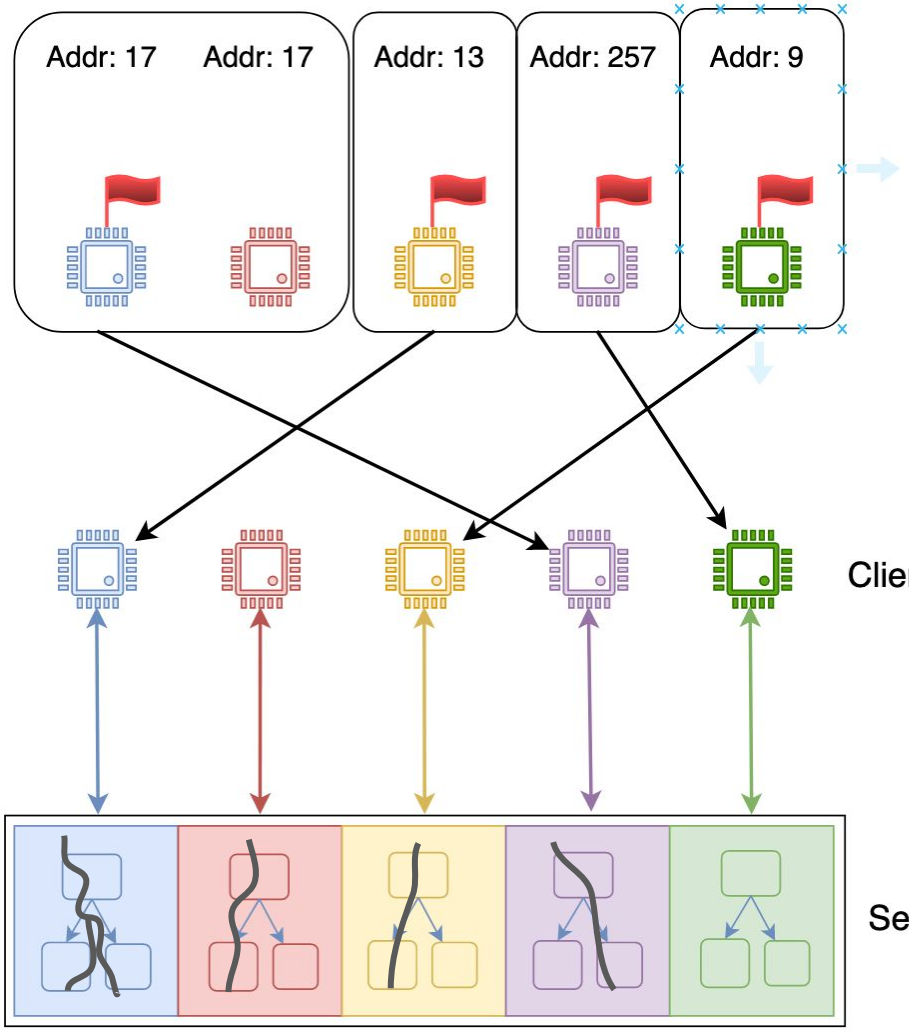
Improved Paradigm (CLT15)

Upon receiving the value, the representative propagates it to the group



Improved Paradigm (CLT15)

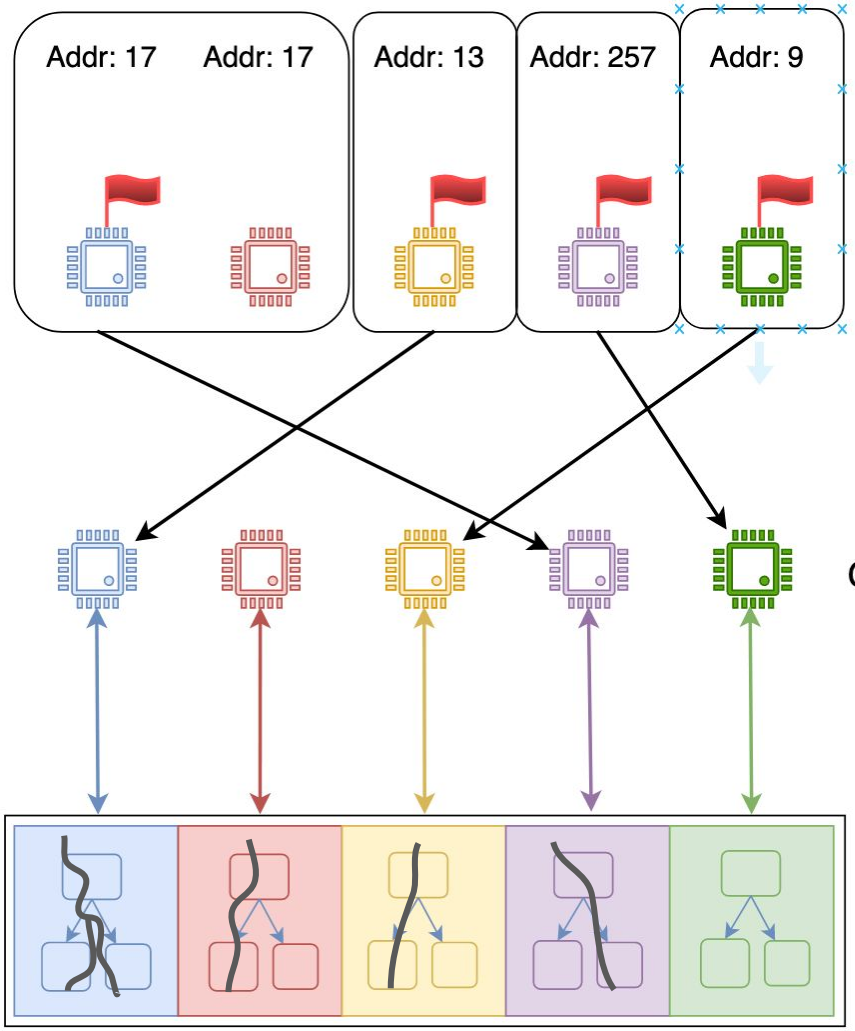
To relocate a value that has been read, representative randomly select a path, update position map and pass the value to the corresponding manager.



Addr 9	Addr 13	Addr 17	Addr 257
leaf xxx	leaf yyy	leaf zzz	leaf aaa

Improved Paradigm (CLT15)

To relocate a value that has been read, representative randomly select a path, update position map and pass the value to the corresponding manager.



Addr 9	Addr 13	Addr 17	Addr 257
leaf xxx	leaf yyy	leaf zzz	leaf aaa

Managers try their best to flush the value down the paths that have been read in this round, if this is not possible, store it temporarily in stash.

With negligible probability that the stashs will overflow!

Improved Paradigm (CLT15)

- Can guarantee client-server communication bandwidth to be $O\left(f(m)\log(m)\log^2(n)\right)$ under recursive construction
- Why does ownership partition still appear random?

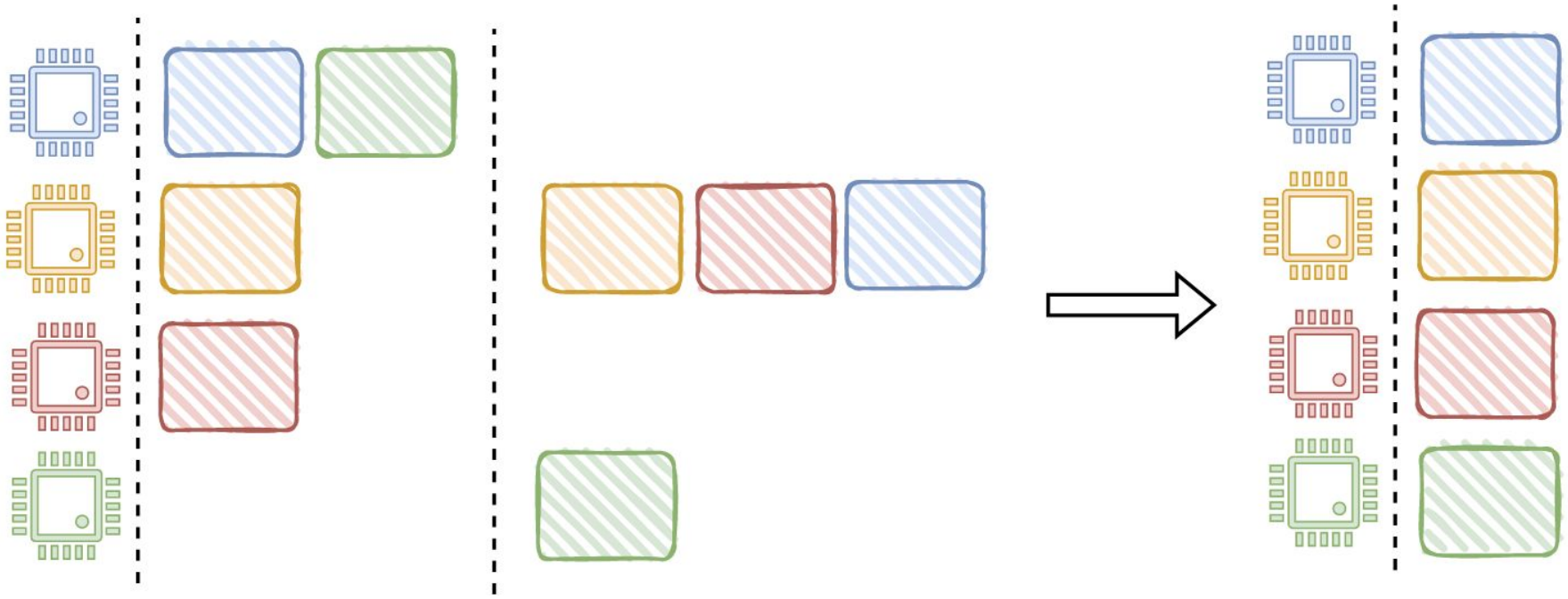
Improved Paradigm (CLT15)

- Can guarantee client-server communication bandwidth to be O(f(m)log(m)log^2(n)) under recursion construction
- Why does ownership partition still appear random?
  - The server knows the ownership, but no other information beyond this.

Further improvements (NK16)

Kartik Nayak, Jonathan Katz

- Under recursive construction, idle managers would need wait for busy managers to proceed to the next round addressing.
- Is there a way to balance manager's tasks?



Further improvements (NK16)

Kartik Nayak, Jonathan Katz

- Managers only need to handle evictions.
  - Clients can directly read any addresses they want
    - Reads won't incur conflicts
  - By adopting additive homomorphic encryption, clients **can independently mark** the cell they read
  - The read values are assigned to a random leaf, will then be merged into a subtree by the corresponding manager

Wrap Up! ˚ν˚

- We can use OPRAM to help us securely access RAM in parallel.
- Parallelism increase throughput at expense of latency.
- Ownership Partition: reduce conflicts.

References

- ORAM images: https://signal.org/blog/building-faster-oram/
- Kartik Nayak and Jonathan Katz. An oblivious parallel RAM with O(log^2 N) parallel runtime blowup. IACR Cryptology ePrint Archive, 2016:1141, 2016.
- Elette Boyle, Kai-Min Chung, and Rafael Pass. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In CRYPTO, 2015.

- Kartik Nayak and Jonathan Katz. An oblivious parallel RAM with O(log^2 N) parallel runtime blowup. IACR Cryptology ePrint Archive, 2016:1141, 2016.