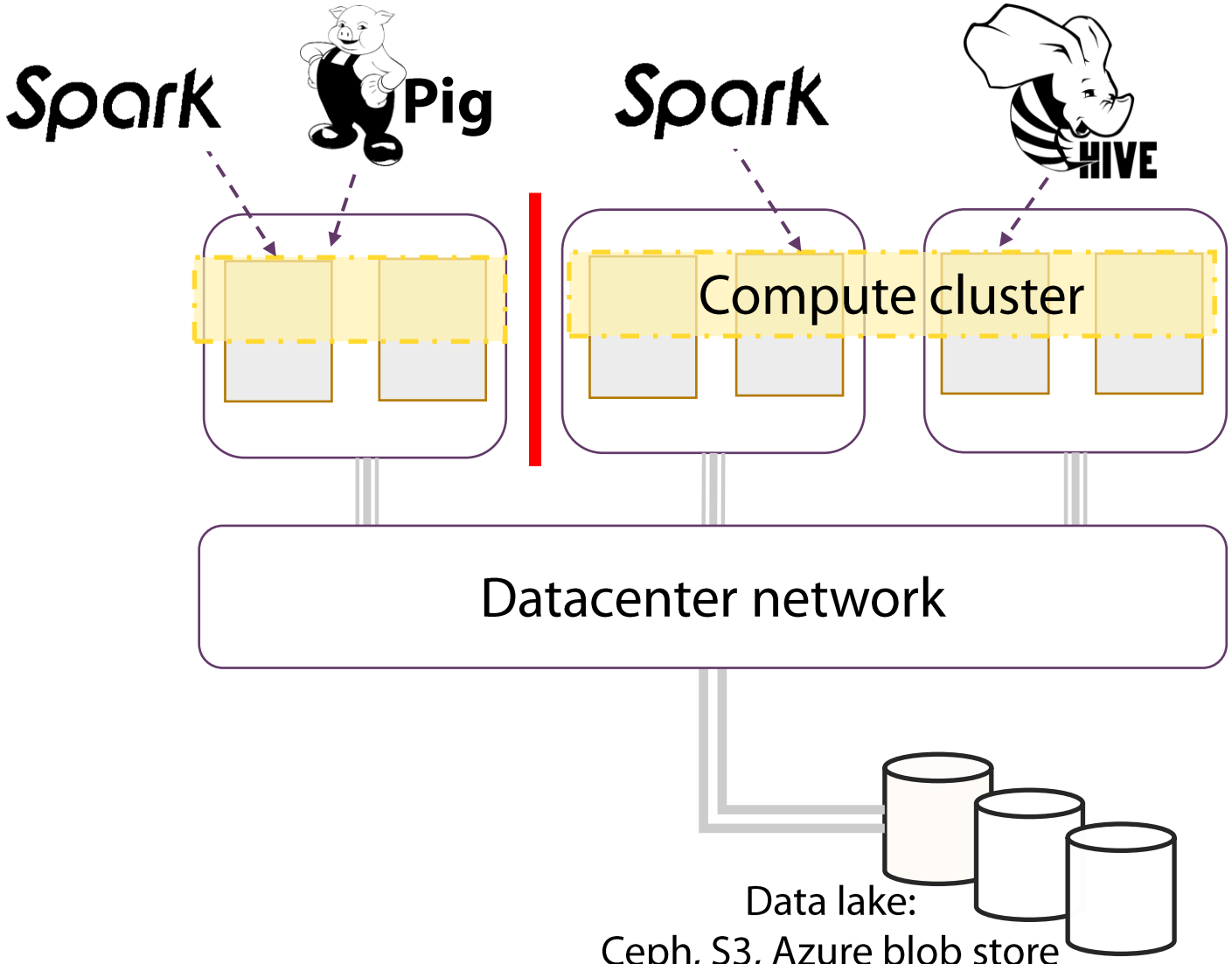
# A community cache with complete information

**Mania Abdi**, Amin Mosayyebzadeh, Mohammad Hossein Hajkazemi, Emine Ugur Keynar, Ata Turk, Larry Rudolph, Orran Krieger, Peter Desnoyers

Usenix FAST, 2/25/21.

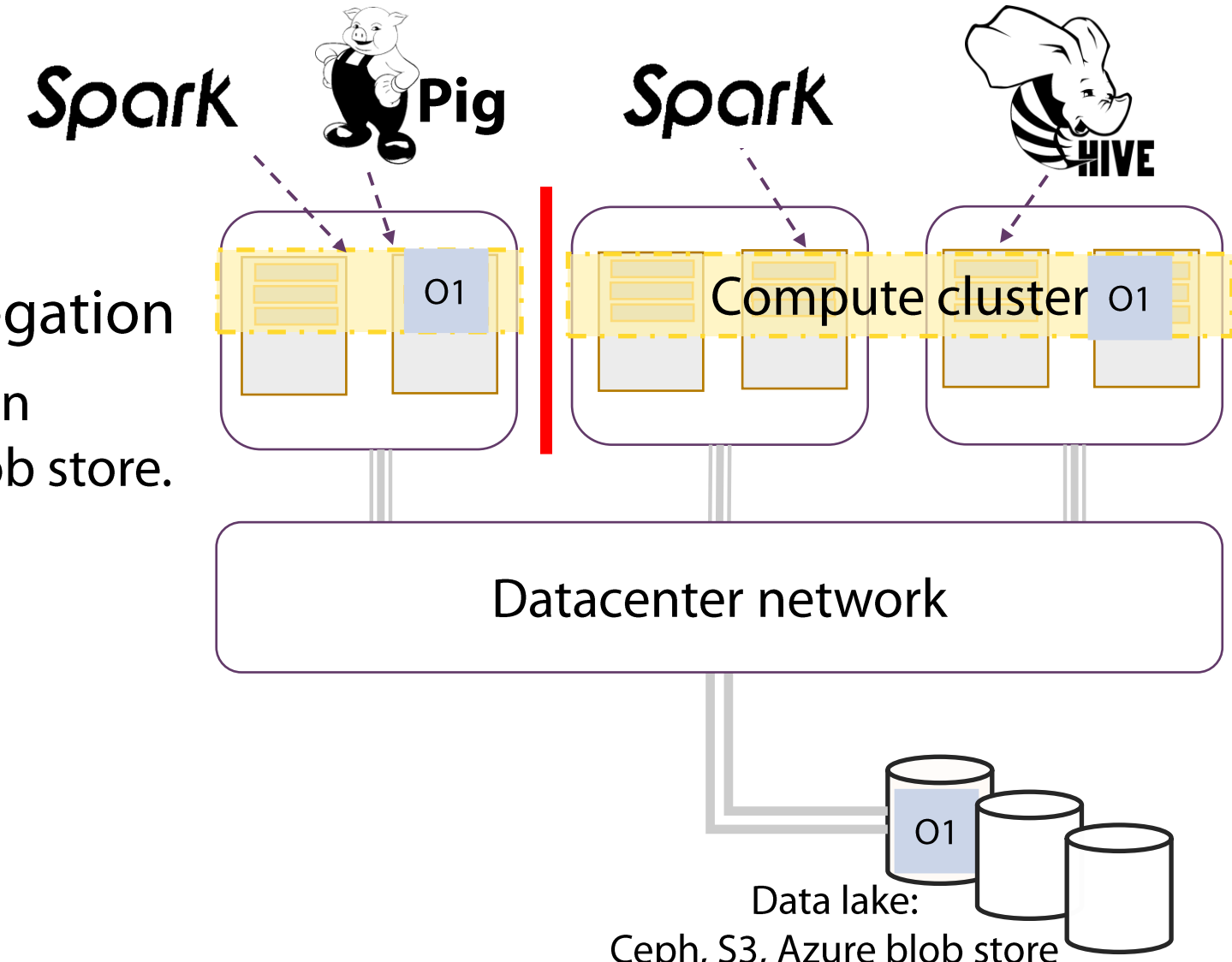



# Deployment of analytic applications



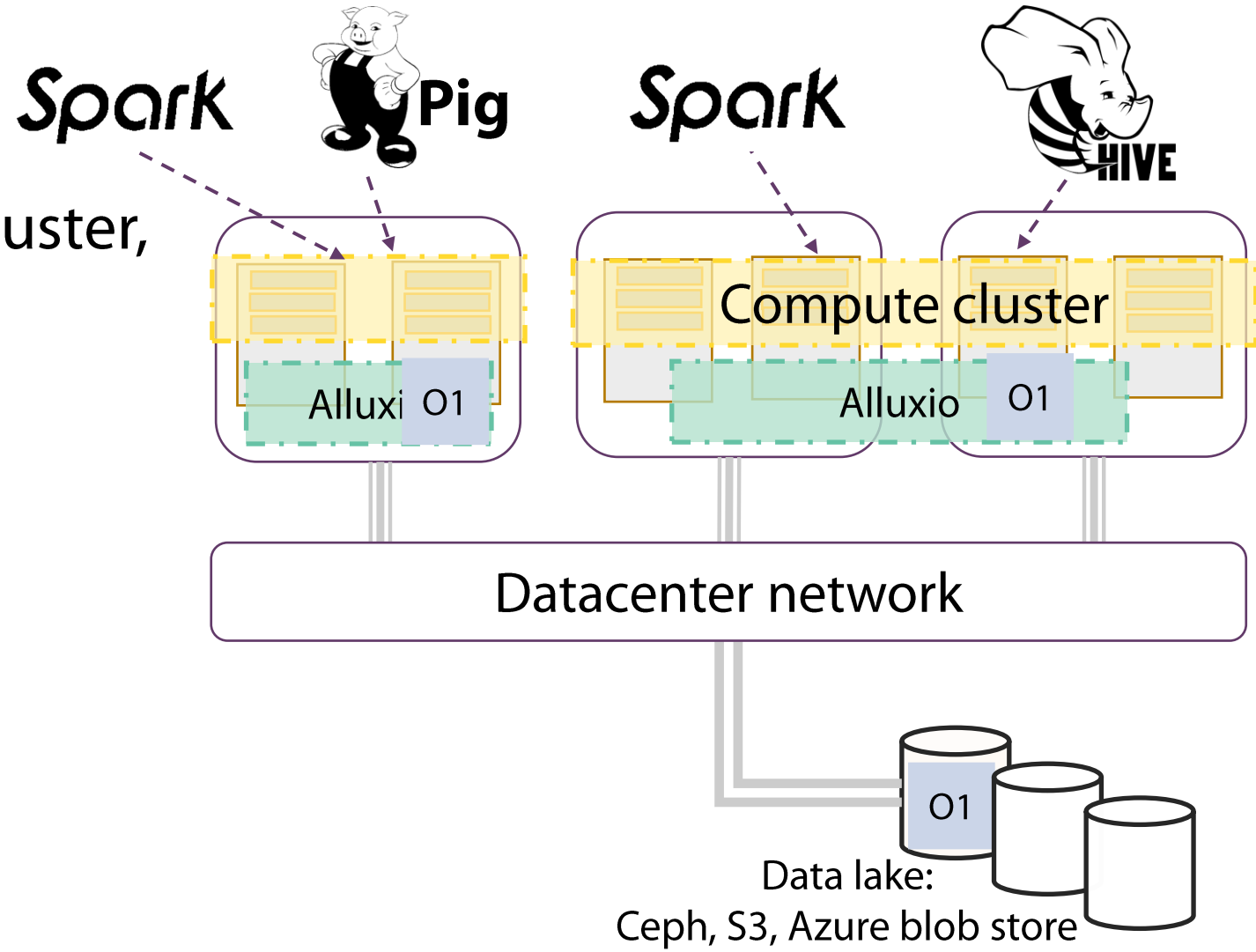
# Deployment of analytic applications

- Storage-compute disaggregation
  - Data is stored and shared in datalake, e.g. S3, Azure blob store.



# Caching for analytic frameworks

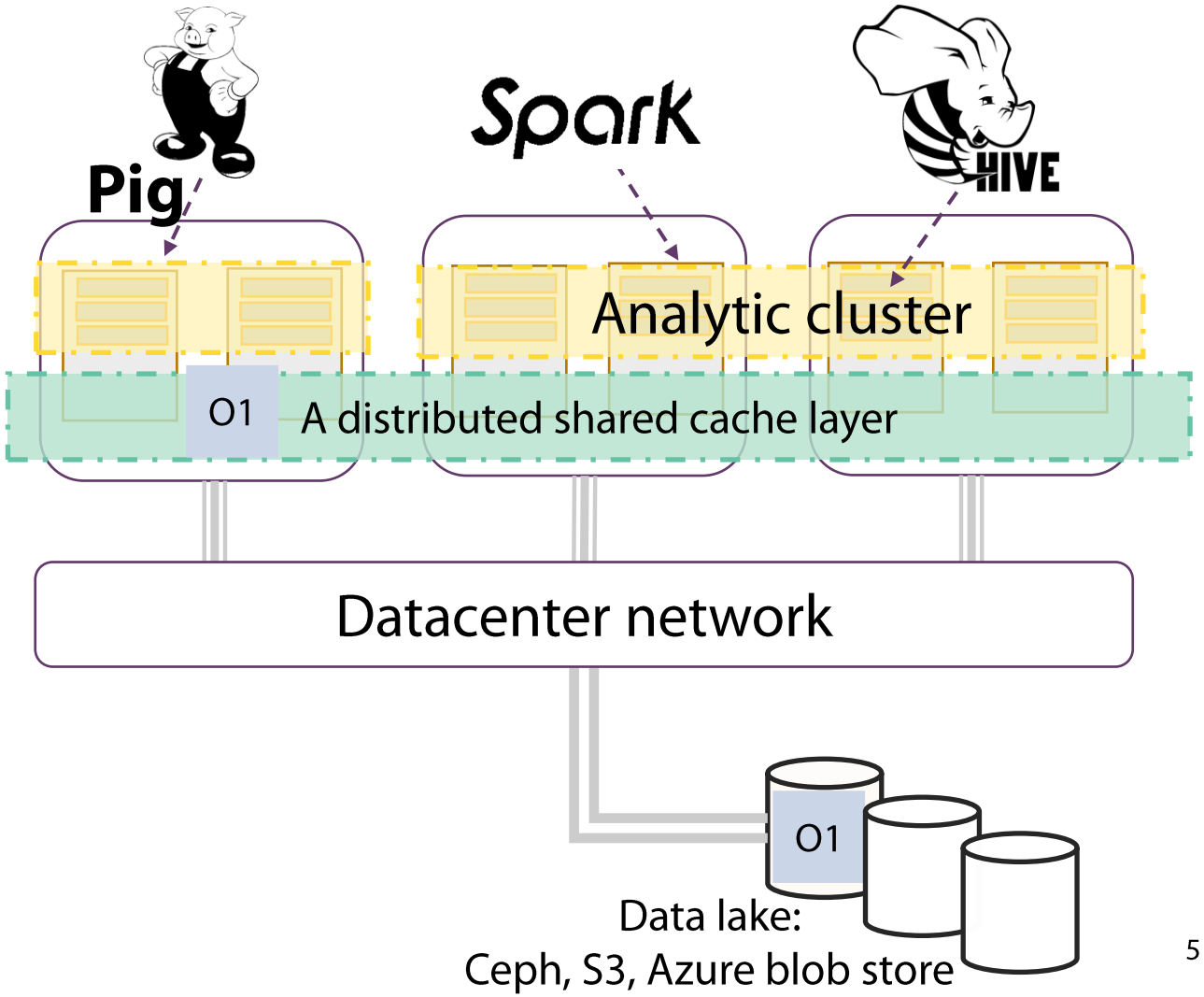
- Caching within compute cluster, e.g. Alluxio,
  - Poor cache allocation
  - Poor network utilization



# Key insight: Community cache

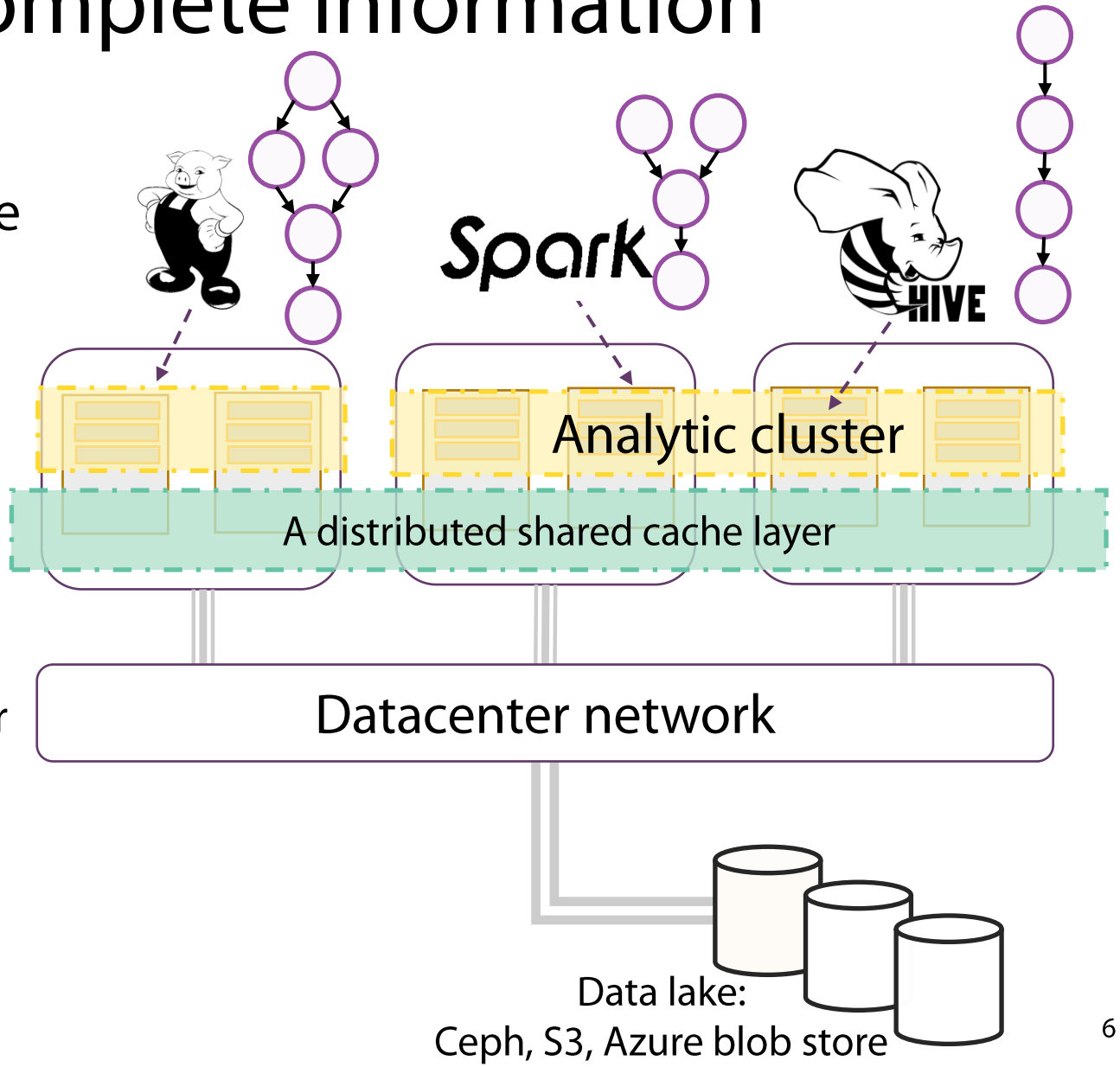
Multiple clusters, potentially running different analytic platforms share a single instance of cache:

- Data is not replicated,
- Elastic compute cluster.



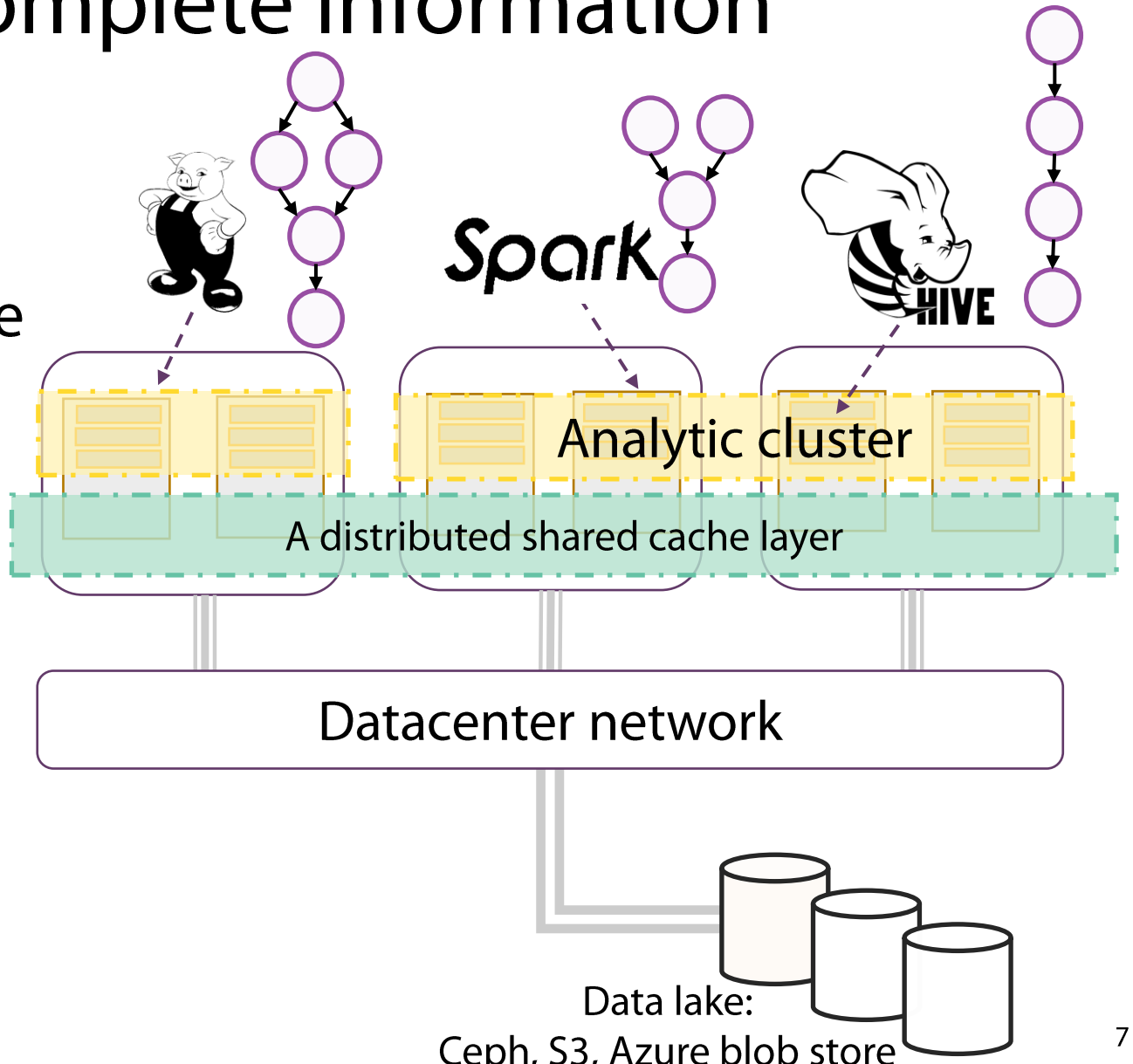
# Community cache with complete information

- Compute cluster caches can cooperate with the scheduler to receives complete scheduling information.
  - The data-flow DAG
    - The operation
    - Path to input objects
    - Ranges within input object
    - The output objects
  - Example of data-flow DAG aware cluster caching is “Caching in the multiverse” [HotStorage 2019]

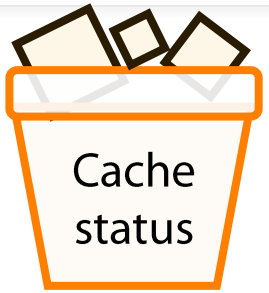


# Community cache with complete information

- Scheduling information and I/O access pattern becomes available through analytic frameworks,
- Datalake infrastructure provides visibility to network and storage usage.

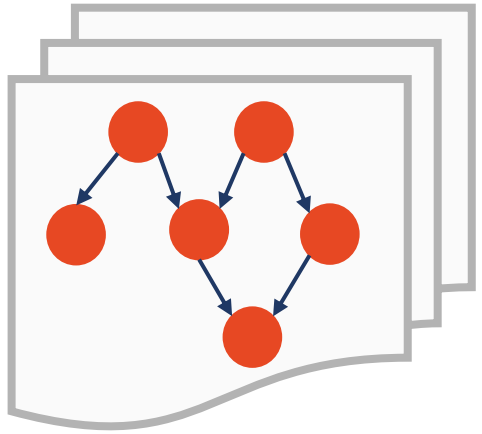


# Community cache with complete information




Execution time prediction

Storage bandwidth monitoring



**Calculate optimize cache plan to improve the runtime of dataflow DAGs** by deciding:

- ❑ What to prefetch
- ❑ What to admit
- ❑ What to evict

# CMR: cache for minimizing runtime

User queries -> Query optimizer -> Dataflow DAG

$S_1$ { $O_1$ = filter (load $I_1$)

$S_2$ { $O_2$ = filter (load $I_2$)

$S_3$ { $p$ = filter (load $I_{3-1}$)
$O_3$ = join $p$ &load $I_{3-2}$

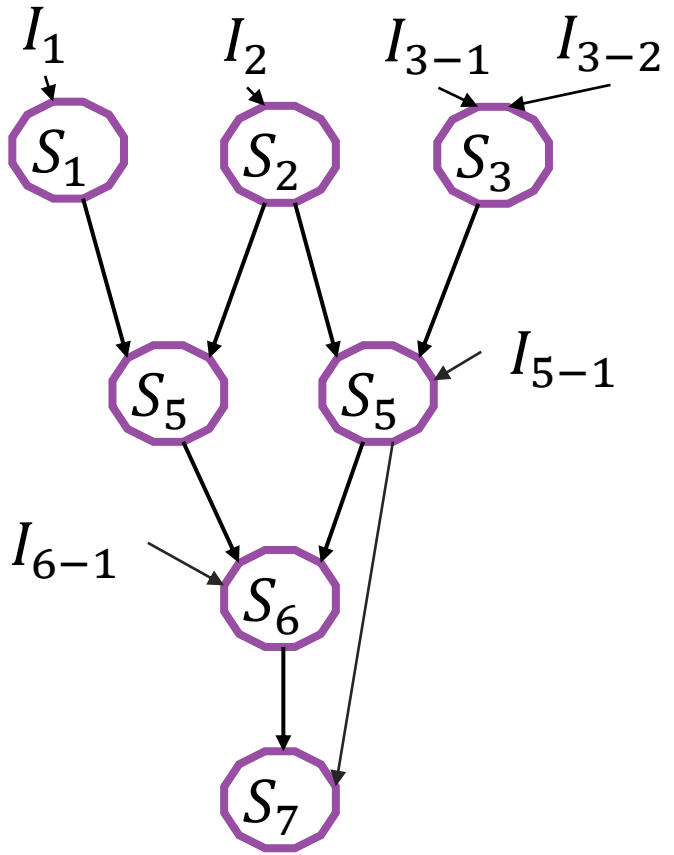
$S_4$ { $O_4$ = join $O_1$& $O_2$

$S_5$ { $O_5$ = join $O_2$& $O_3$& (load $I_5$)

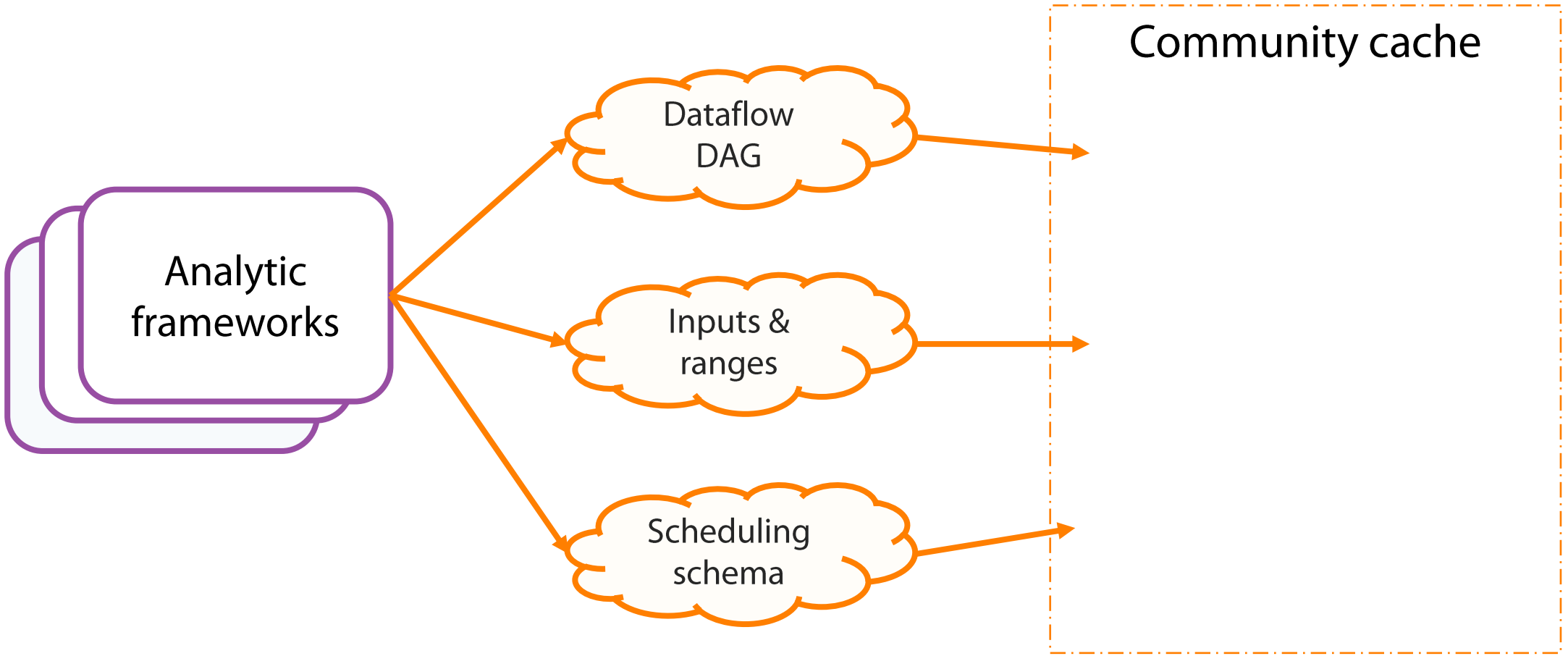
$S_6$ { $O_6$ = join $O_4$& $O_6$& (load $I_6$)

$S_7$ { $(out)$ = store (groupby $O_6$&$O_5$)

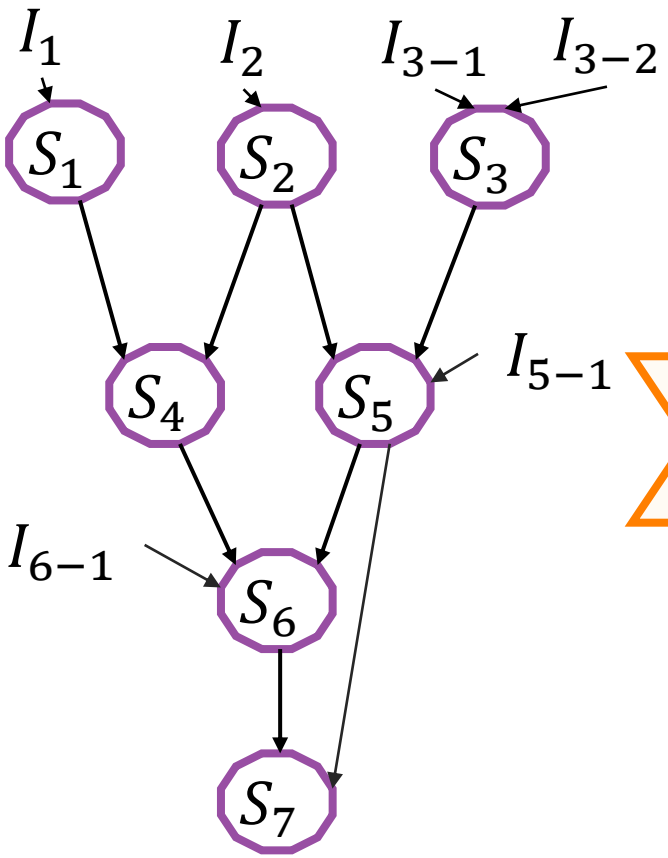
Query #7 from TPC-H benchmark

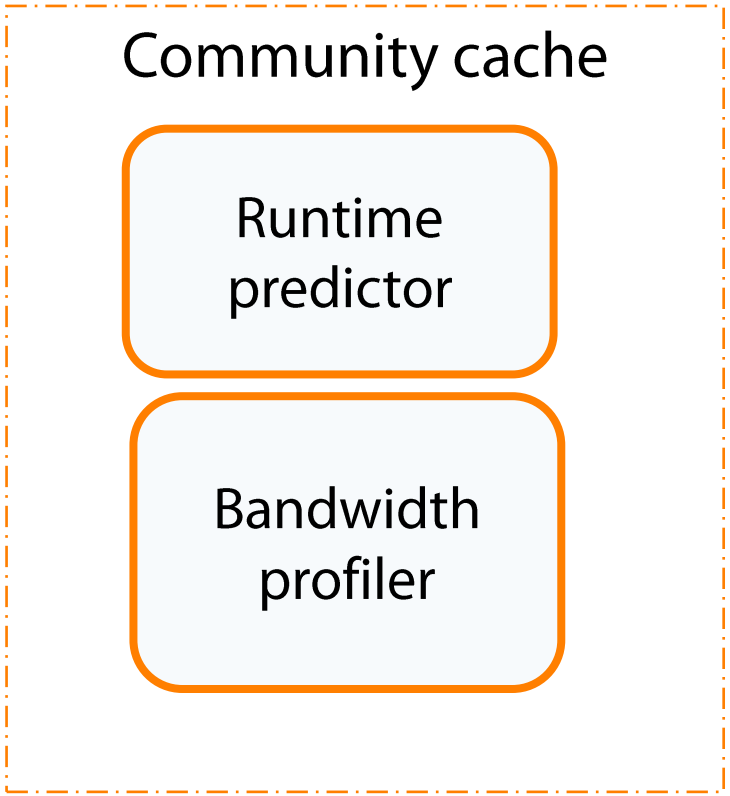


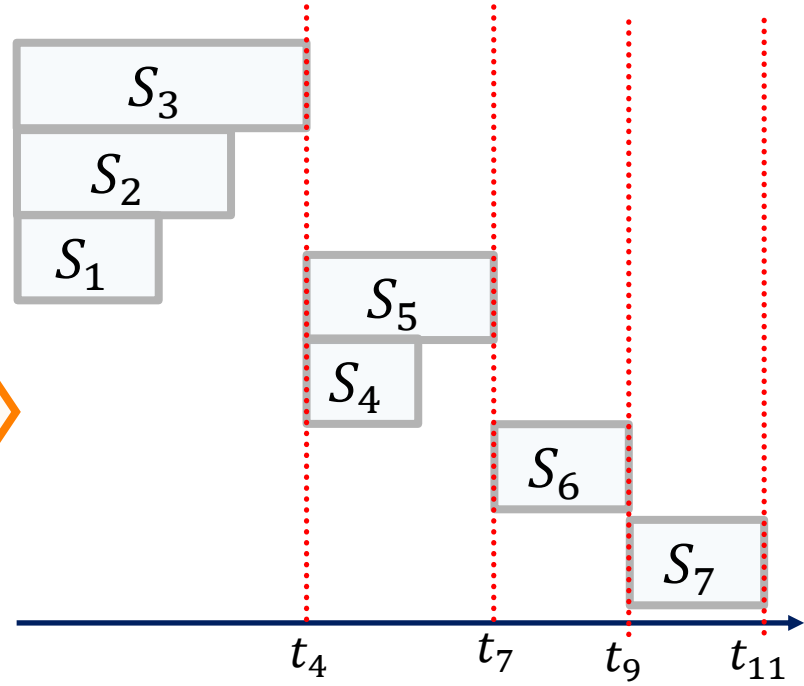
# CMR: cache for minimizing runtime



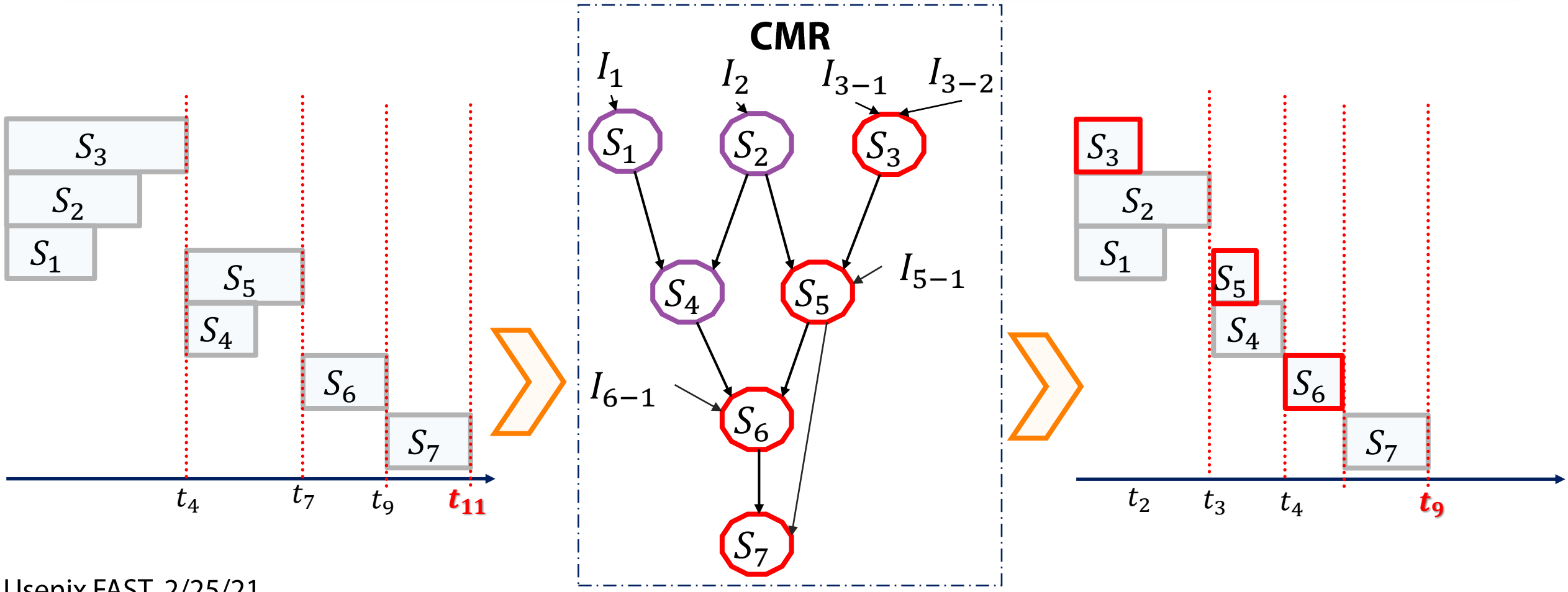
# CMR: cache for minimizing runtime



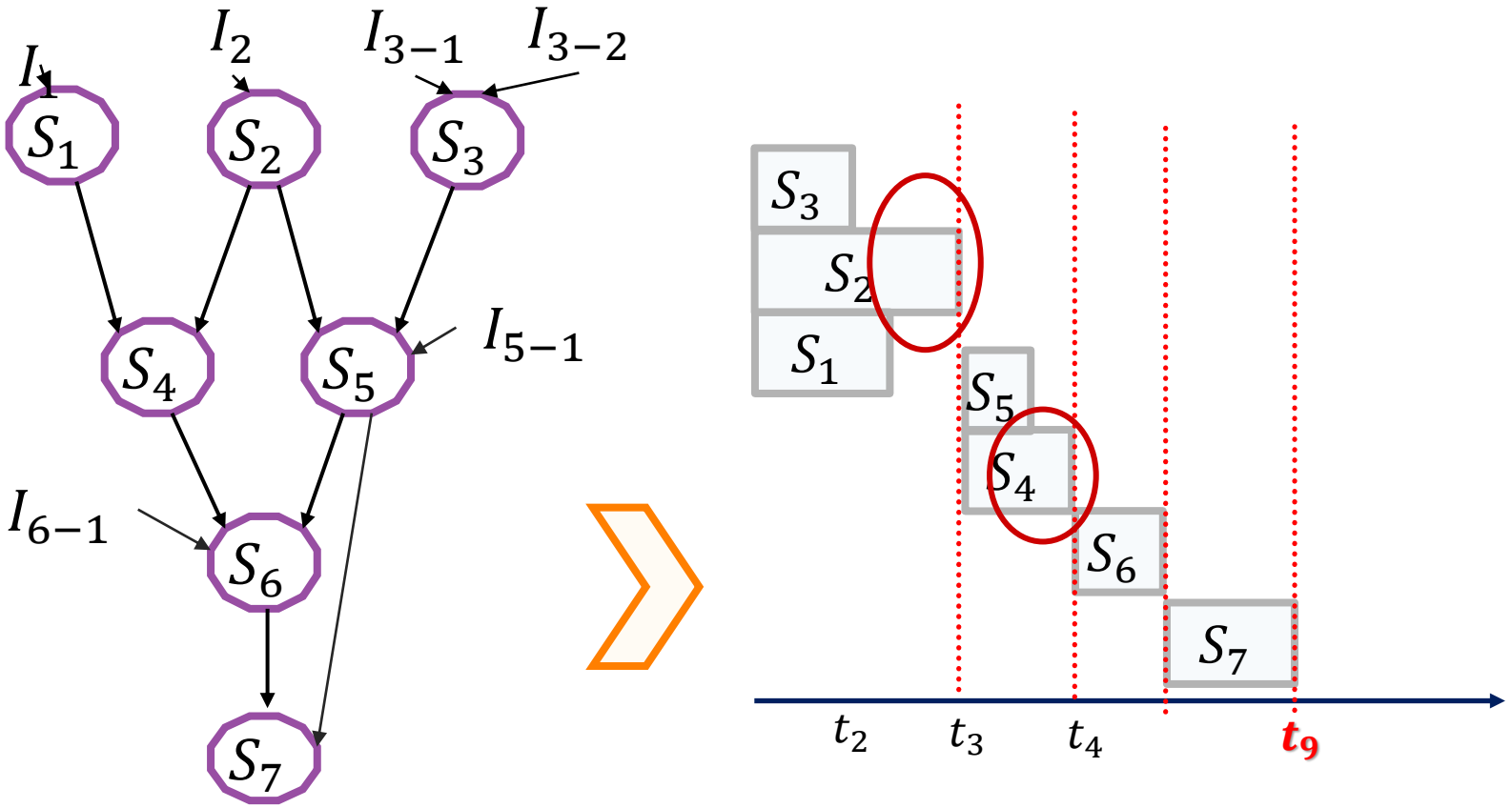



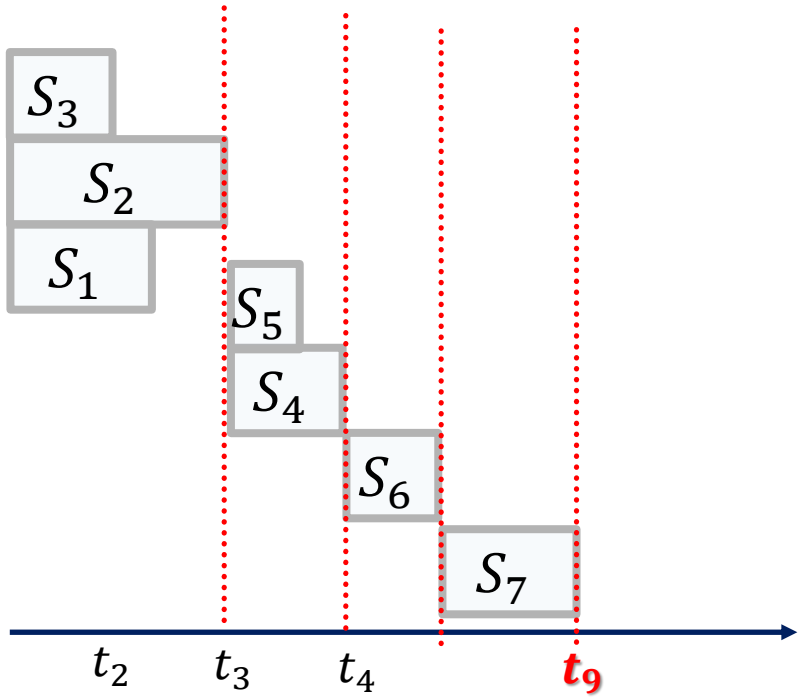
# CMR: cache for minimizing runtime

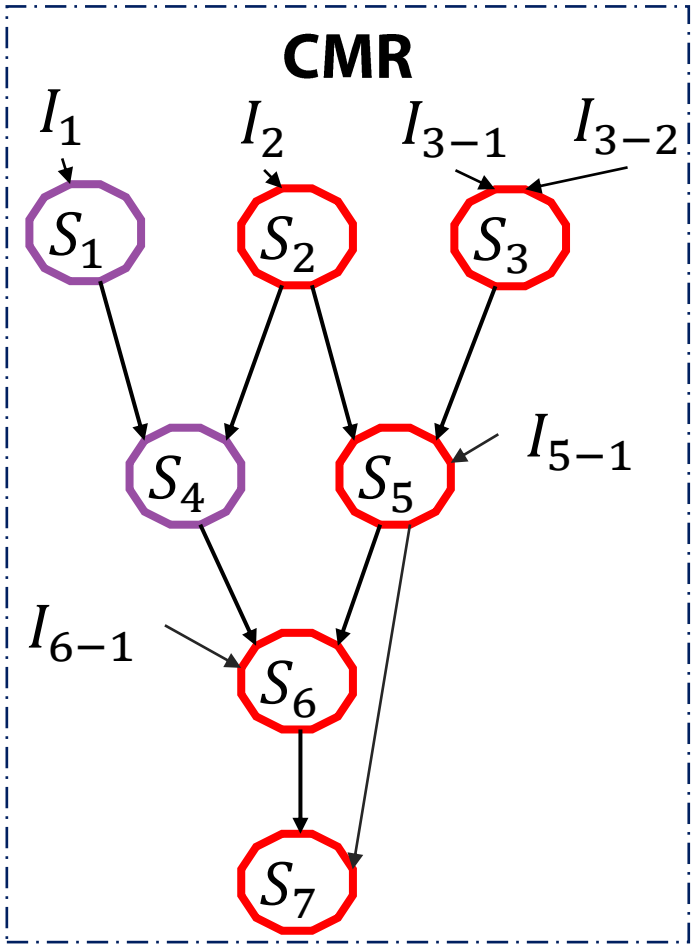


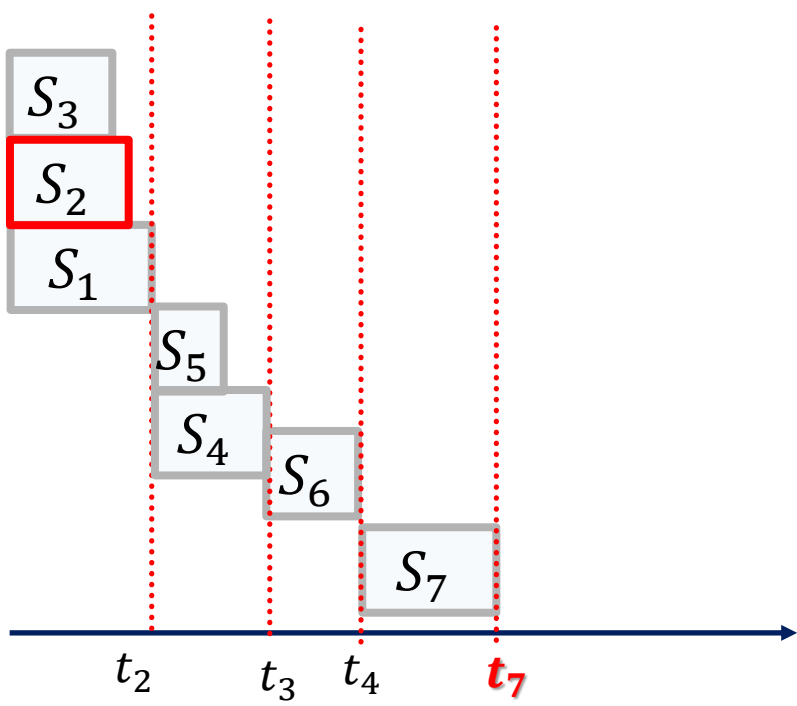
# CMR: going beyond critical path



# CMR: cache for minimizing runtime



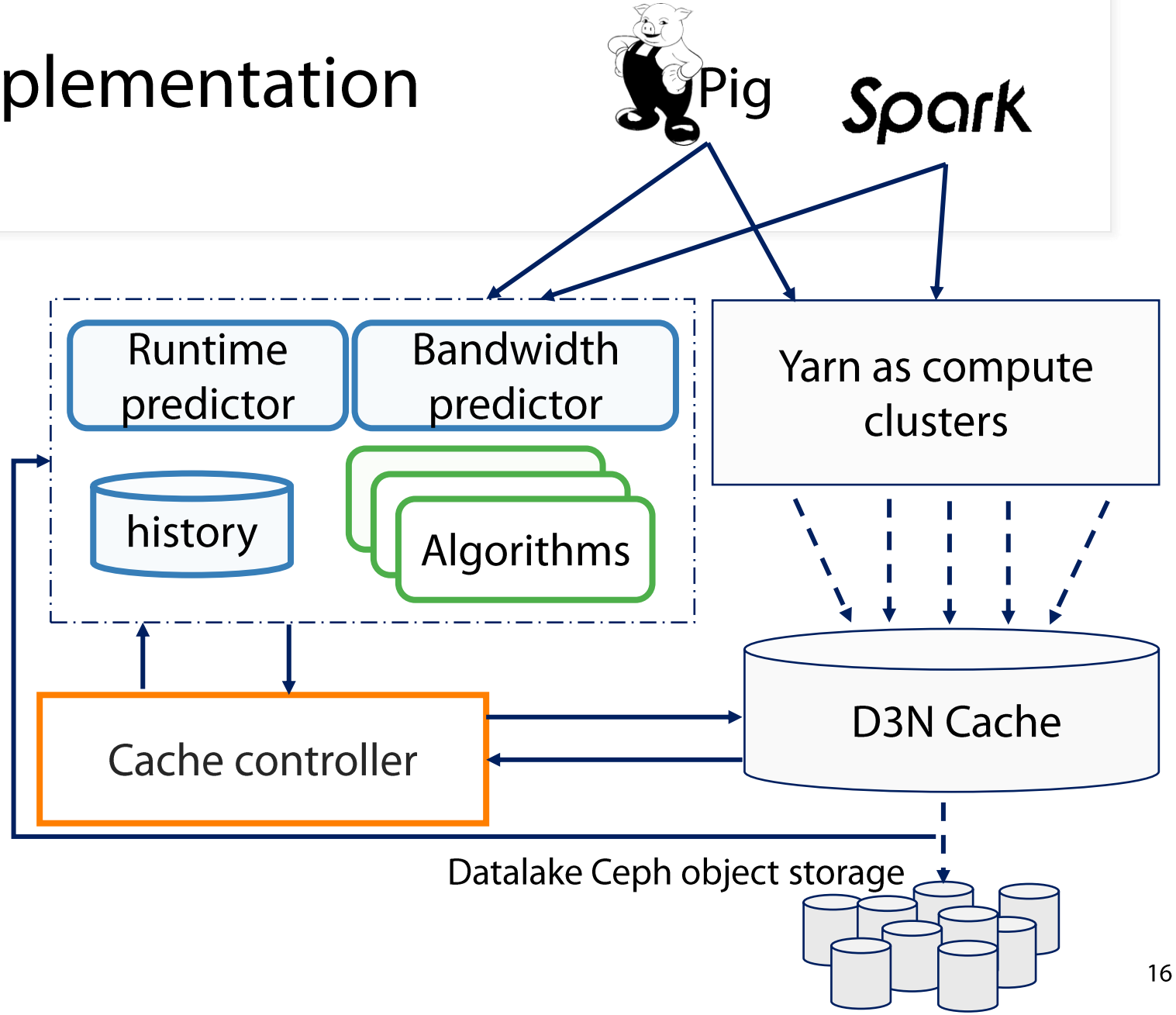



# CMR-M: concurrent queries

- Prioritize prefetching and caching for DAGs with maximum data sharing among them to minimize the backend storage load and maximize the runtime reduction.

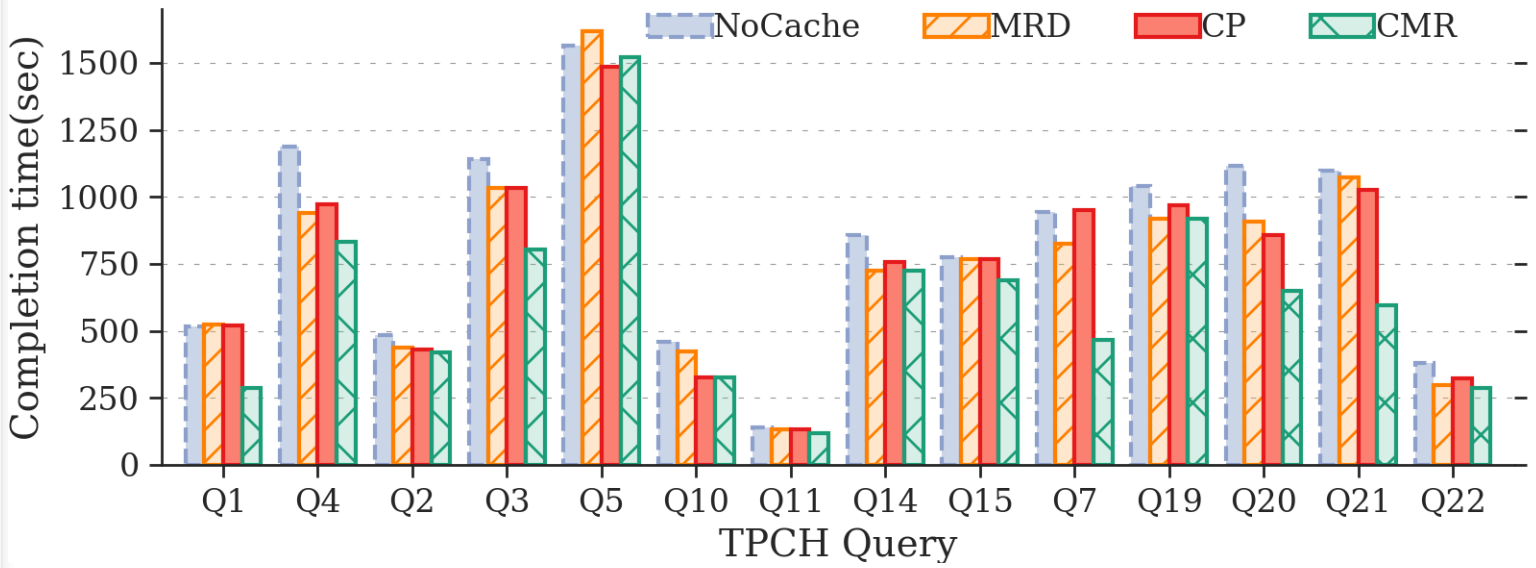
# Community cache implementation

- Instrumenting Spark and Pig to extract dataflow DAG
- Implement the community cache by extending D3N caches
- Profiling datalake bandwidth
- Profiling job execution
- CMR algorithms for multi-job and single job



# CMR evaluation: Pig

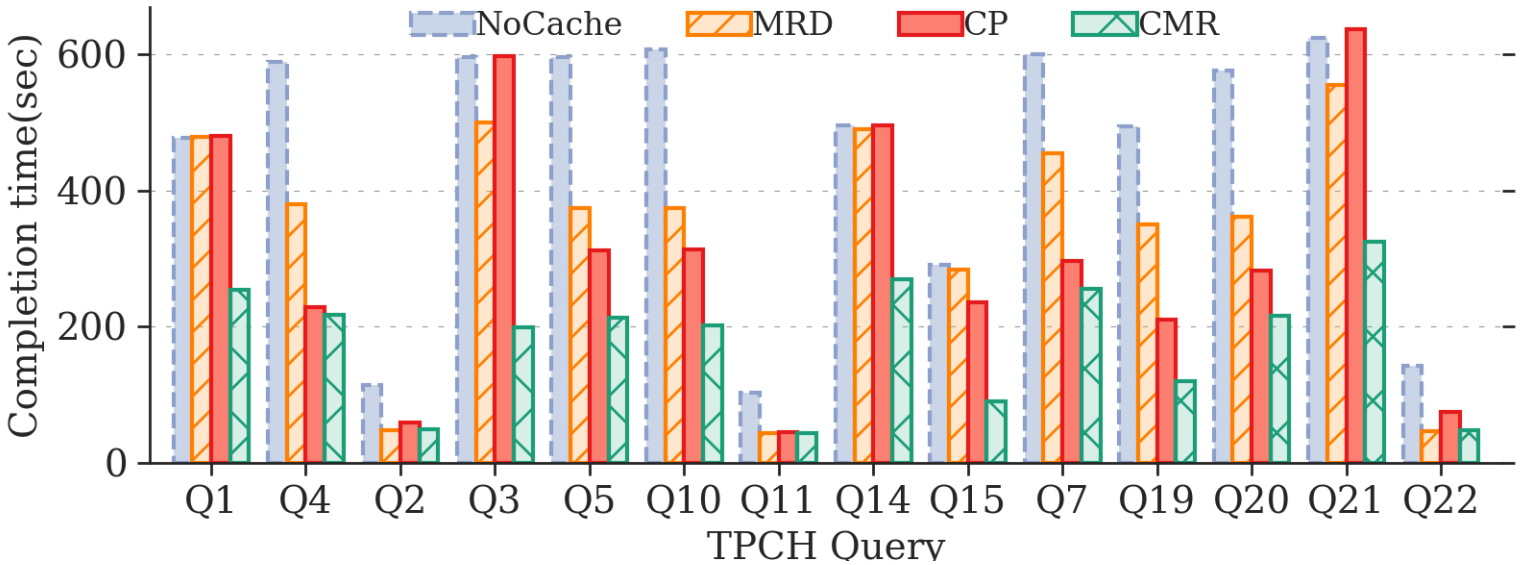
- TPC-H benchmark queries
- Dataset size: 64GB
- Backend bandwidth: 10Gbps
- Plug different algorithms:
  - No cache/LRU
  - Minimum reference distance (MRD)
  - Critical path (CP)
  - CMR
- Hadoop Cluster:
  - 16 nodes
    - 1.4TB memory,
    - 128 core




- Upto 2x improvement compare to no-cache baseline.
- Upto 1.3x improvement compare to other schemas.

# CMR evaluation: Spark

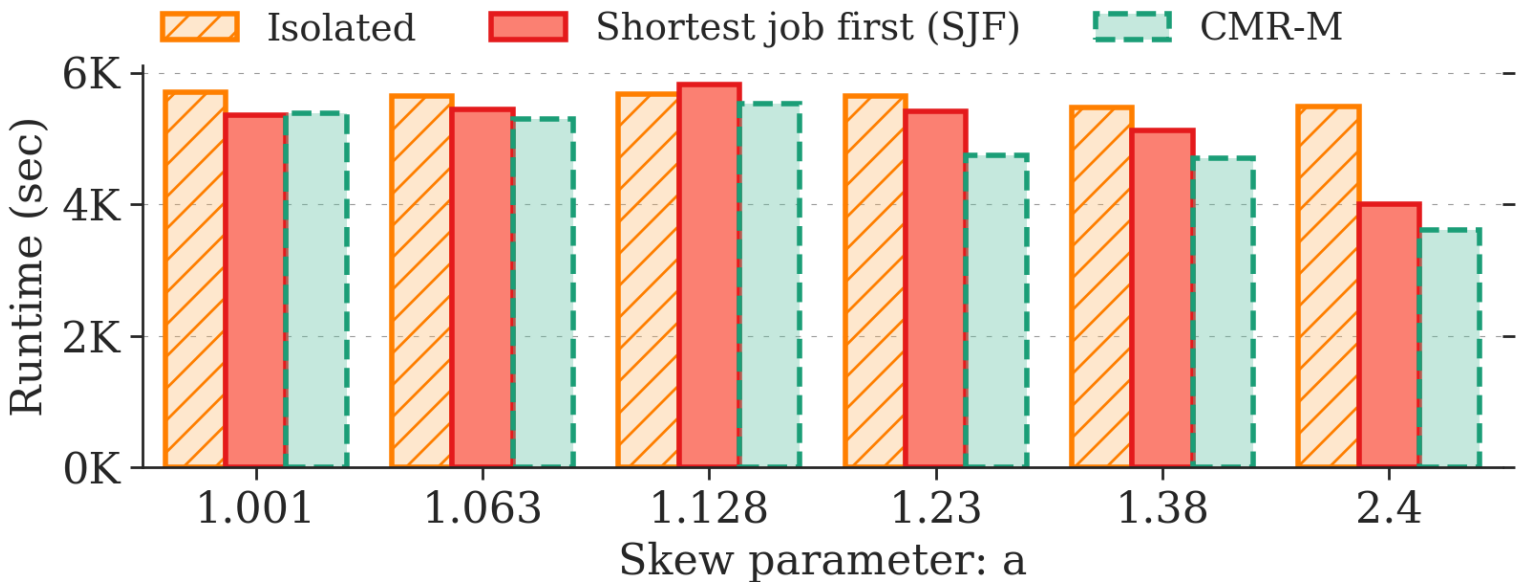
- TPC-H benchmark queries
- Dataset size: 64GB
- Backend bandwidth: 10Gbps
- Plug different algorithms:
  - No cache/LRU
  - Minimum reference distance (CMR)
  - Critical path (CP)
  - CMR
- Hadoop Cluster:
  - 16 nodes:
    - 1.4TB memory,
    - 128 core




- Up to 3x improvement compare to no-cache baseline.
- Up to 2x improvement compare to other schemas.

# CMR-M evaluation: compare with alternatives

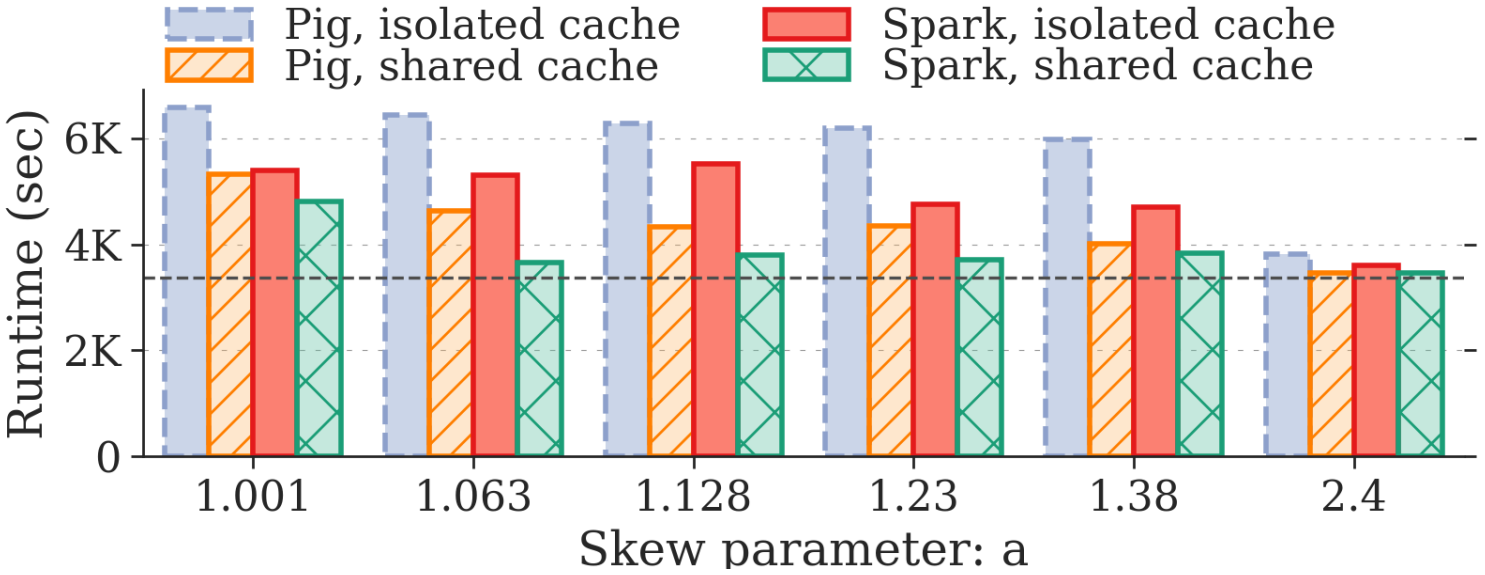
- Synthetic workload with different reuse access pattern
- Plug different algorithms:
  - CMR-M
  - Shortest job first (SJF)
  - Isolated cache




- CMR-M outperforms its alternatives as the data sharing increases among queries.

# Community cache vs. cluster cache

- Synthetic workload with different reuse access pattern,
- Two compute clusters running different analytic frameworks.




- Community cache leads to performance improvement by 1.5x compare to cluster caching, e.g. Alluxio.

# Conclusion

- Community cache enables data sharing across analytic clusters, potentially running different analytic frameworks.
- By being part of datalake, instead of compute cluster, a community cache has visibility to unique information such as competing demands to datalake. This enables:
  - Intelligent caching mechanisms such as CMR (cache for minimizing runtime),
  - Joint scheduling of backend bandwidth and cache capacity,
  - Cache scheduling for concurrently running jobs.