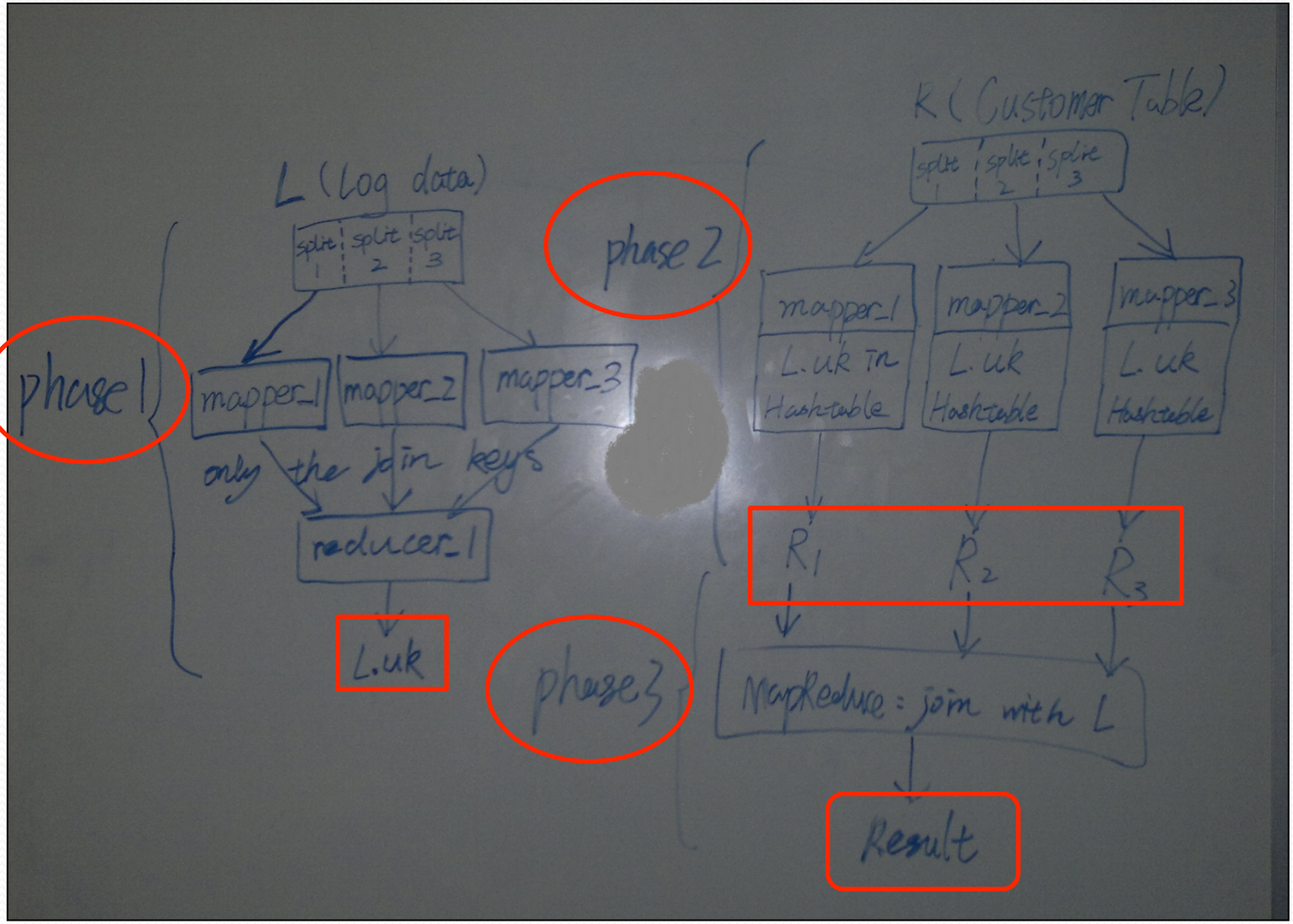
ReStore:
Reusing Results of MapReduce Jobs

Yang Zheng

Outline

- Introduction to MapReduce
- Overview of ReStore
- Implementation Details
- Experiments

Introduction to MapReduce



Introduction to MapReduce

What comes to your mind?

- Keep these intermediate results and final results and reuse them for future workflows submitted to the system.

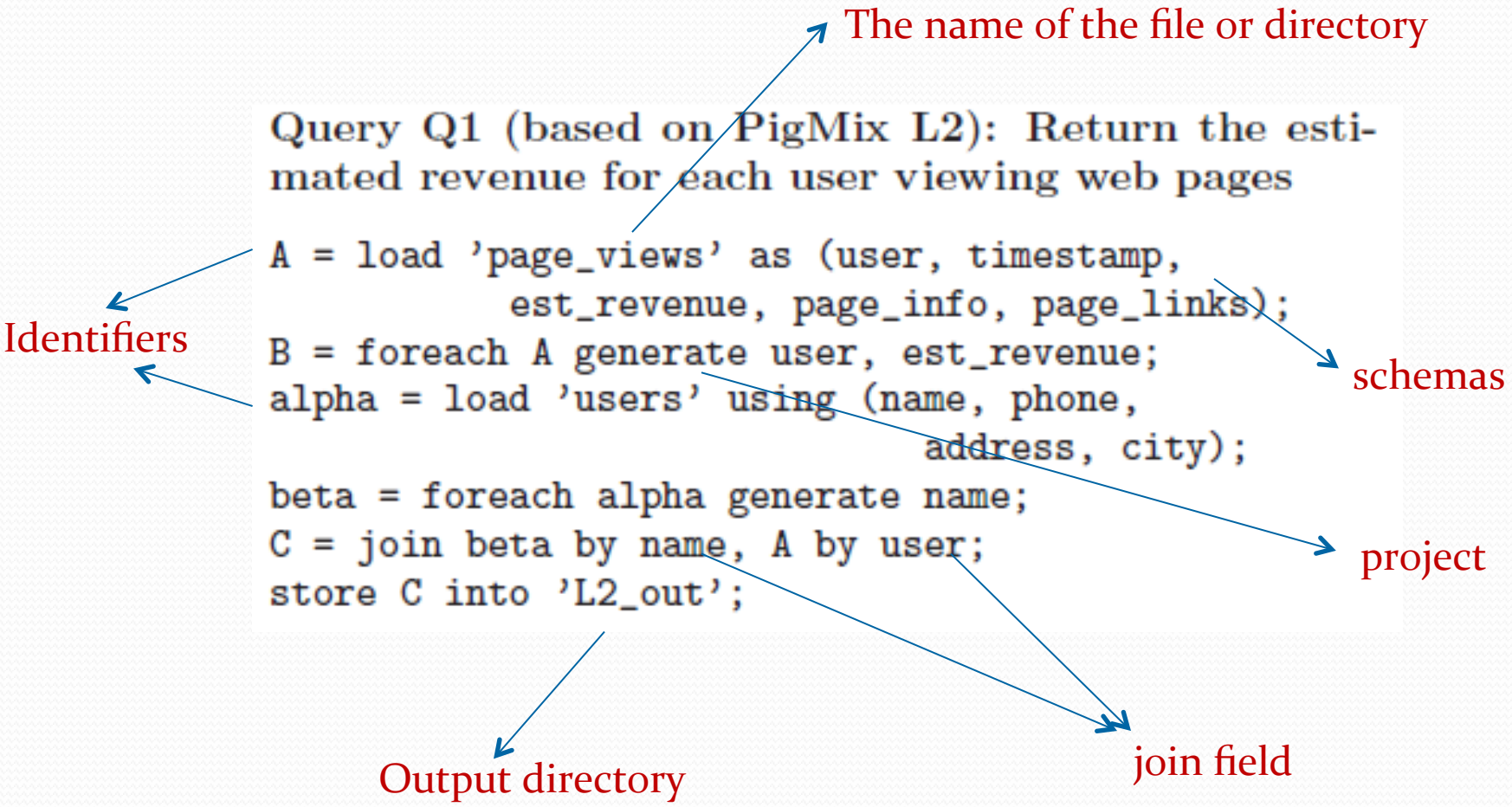
An example

- Facebook stores the result of any query in its MapReduce cluster for seven days so that it can be shared among users.

Overview of ReStore

- Rewrites the MapReduce jobs in a submitted workflow to reuse job outputs previously stored in the system
- Stores the outputs of executed jobs for future reuse
- Creates more reuse opportunities by storing the outputs of sub-jobs in addition to whole MapReduce jobs
- Selects the outputs of jobs to keep in the distributed file system and those to delete

Overview of ReStore



Overview of ReStore

Query Q2 (based on PigMix L3): Return the total estimated revenue for each user viewing web pages, grouped by user name

```
A = load 'page_views' as (user, timestamp,
          est_revenue, page_info, page_links);
B = foreach A generate user, est_revenue;
alpha = load 'users' using (name, phone,
                            address, city);
beta = foreach alpha generate name;
C = join beta by name, A by user;
D = group C by $0;
E = foreach D generate group, SUM(C.est_revenue);
store E into 'L3_out';
```

Positional notation

Overview of ReStore

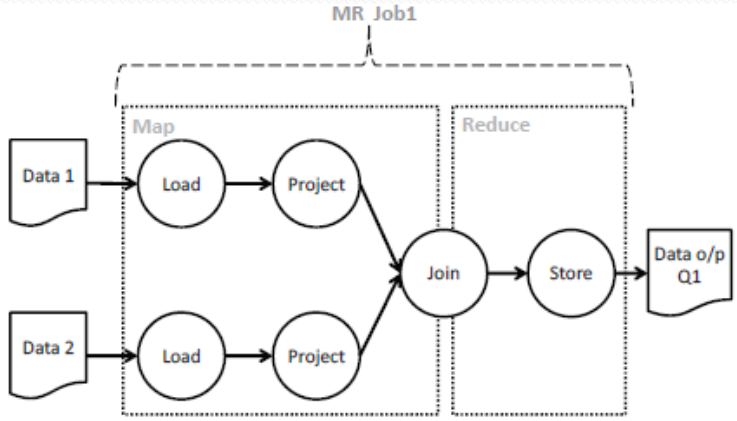


Figure 2: The MapReduce workflow for query Q1.

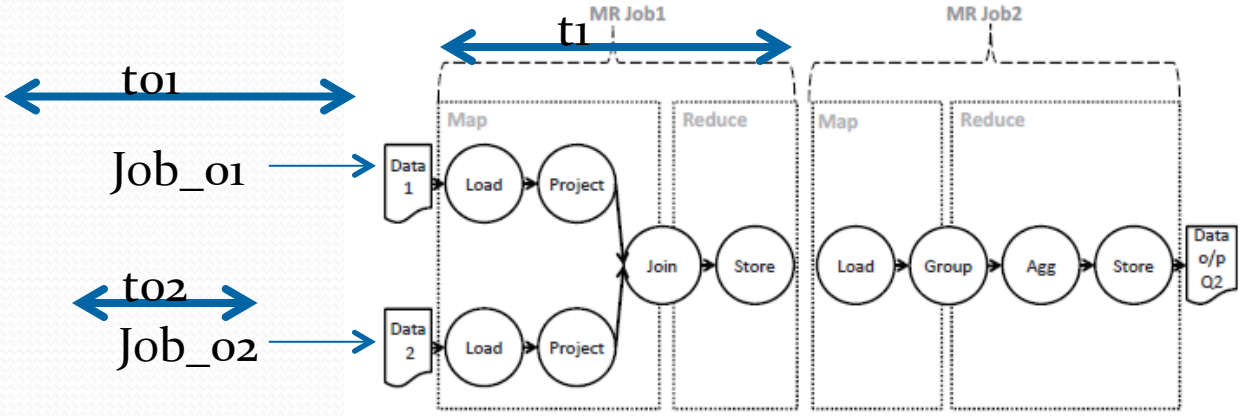


Figure 3: The MapReduce workflow for query Q2.

Overview of ReStore

The time needed to finish executing job_n

the time needed to finish all jobs on which Job_n depends

$$T_{total}(Job_n) = ET(Job_n) + max_{i \in Y}\{T_{total}(Job_i)\} \qquad (1)$$

The total time needed to execute job_n

Overview of ReStore

ReStore generates two types of reuse opportunities by materializing the output of:
(1) whole jobs, which reduces
$\max_{i \in Y}\{T_{total}(Job_i)\}$ in future workflows,
(2) operators in jobs (sub-jobs), which reduces $ET(Job_n)$ in future workflows.

Overview of ReStore

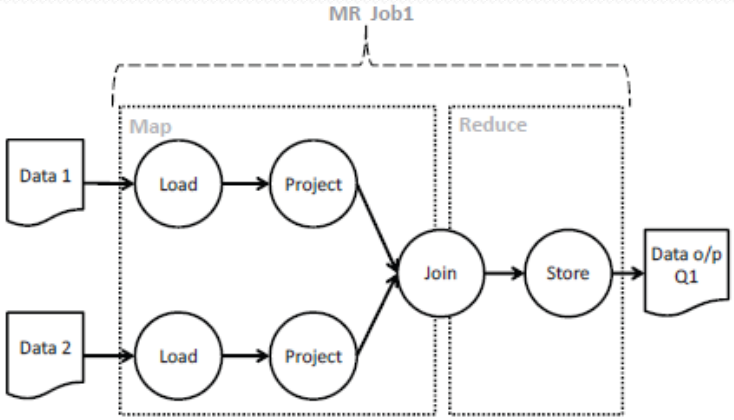


Figure 2: The MapReduce workflow for query Q1.

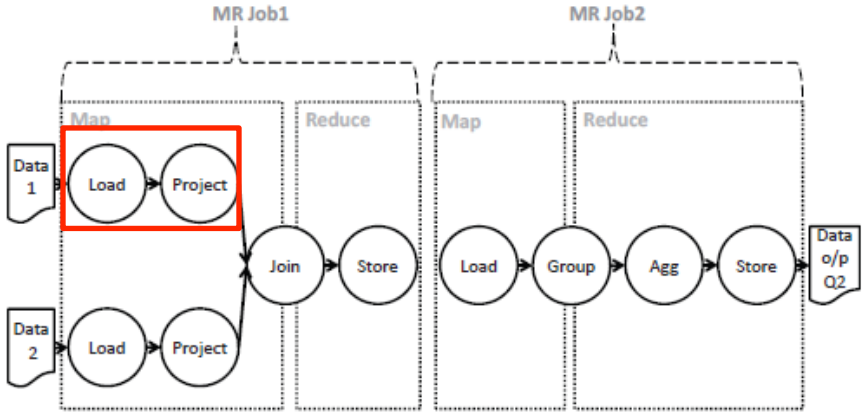


Figure 3: The MapReduce workflow for query Q2.

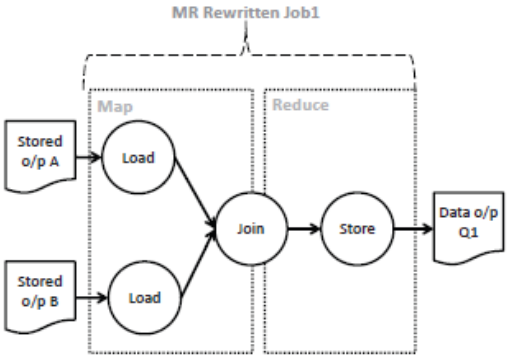


Figure 6: The MapReduce workflow for query Q1 after rewriting it to reuse the outputs of sub-jobs.

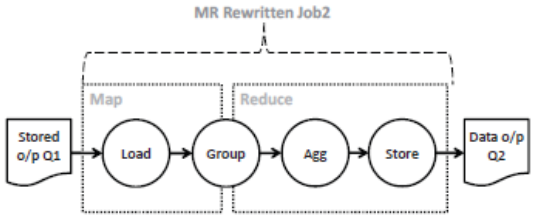


Figure 4: The MapReduce workflow for query Q2 after rewriting it to reuse the output of query Q1.

Overview of ReStore

ReStore Sysem Archiecture

- Input
  - Workflow of MapReduce jobs generated by a dataflow system for an input query
- Output
  - a modified MapReduce workflow that exploits prior jobs executed in the MapReduce system and stored by ReStore
  - a new set of job outputs to store in the distributed file system

Overview of ReStore

- Repository
  - the physical query execution plan of the MapReduce job that was executed to produce this output
  - the filename of the output in the distributed file system
  - statistics about the MapReduce job that produced the output and the frequency of use of this output by different workflows

Overview of ReStore

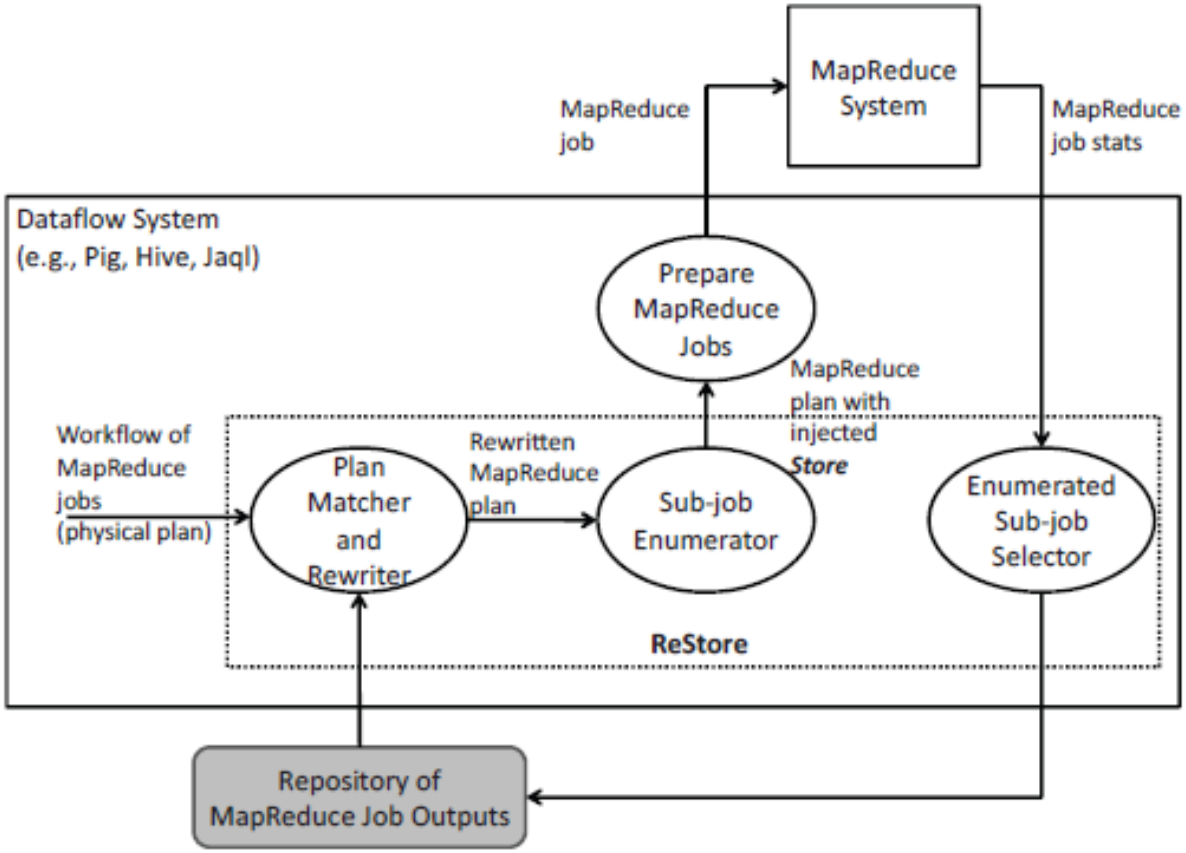


Figure 7: ReStore system architecture.

Implementation Details

- Plan Matcher and Rewriter
  - Scans sequentially through the physical plans in the repository and tests whether each plan matches the input MapReduce job.
  - As soon as a match is found, the input MapReduce job is rewritten to use the matched physical plan in the repository.
  - After rewriting, a new sequential scan through the repository is started to look for more matches to the rewritten MapReduce job.
  - If a scan through the repository does not find any matches, ReStore proceeds to matching the next MapReduce job in the workflow.

Implementation Details

- Sub-job Enumerator & Enumerated Sub-job Selector
  - Two types of reuse opportunaties
    - Store the output of whole MapReduce jobs
    - Store the output of subjobs
  - Which subjob should also be considered as candidates?
    - Some physical operators such as Project and Filter are known to reduce the size of their input
    - Other physical operators such as Join and Group are known to be expensive, so their outputs are also good sub-job candidates because replacing them with stored output reduces the time to execute these operators

Implementation Details

- Manage the Restore Repository
  - Keep a candidate job in the repository only if the size of its output data is smaller than the size of its input data.
  - Keep a candidate job in the repository only if Equation 1 tells us that there will be a reduction in execution time for workflows reusing this job.
  - Evict a job from the repository if it has not been reused within a window of time.
  - Evict a job from the repository if one or more of its inputs is deleted or modified.

Experiments

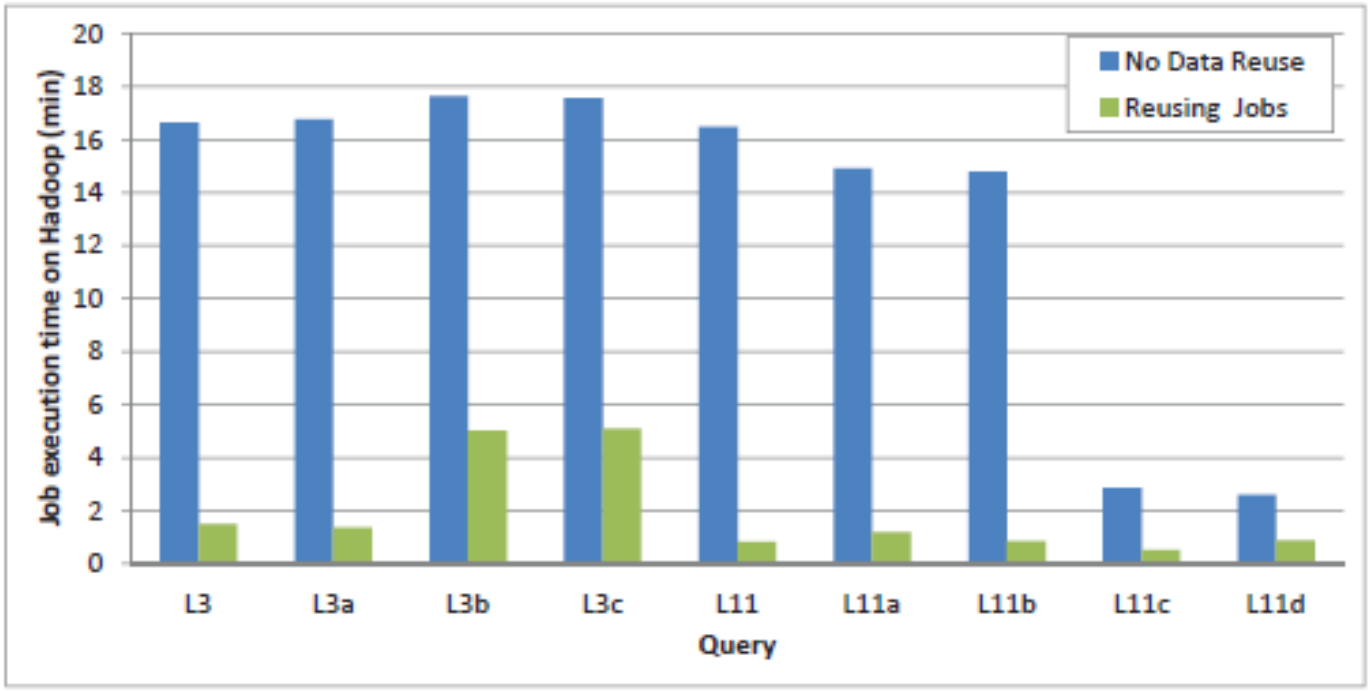


Figure 9: The effect of reusing whole job outputs for data size 150GB.

Experiments

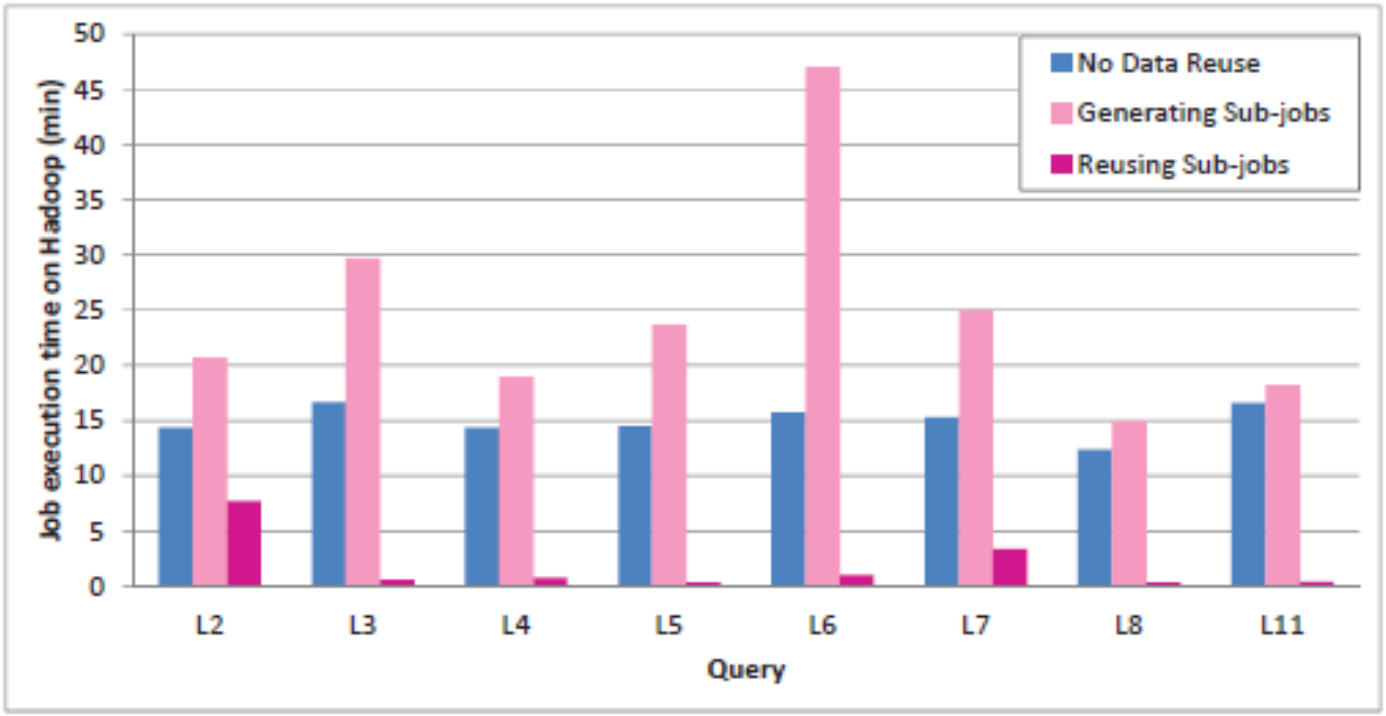


Figure 10: The effect of reusing sub-job outputs for data size 150GB.

Conclusion

- MapReduce mode has become widely accepted for analyzing large data sets
- In many cases, users express complex analysis tasks not directly with MapReduce but rather with higher-level SQL-like query languages
- Important to make full use of the intermediate outputs of jobs to gain performance improvement

Any Questions?