UNIVERSITY OF CAMBRIDGE

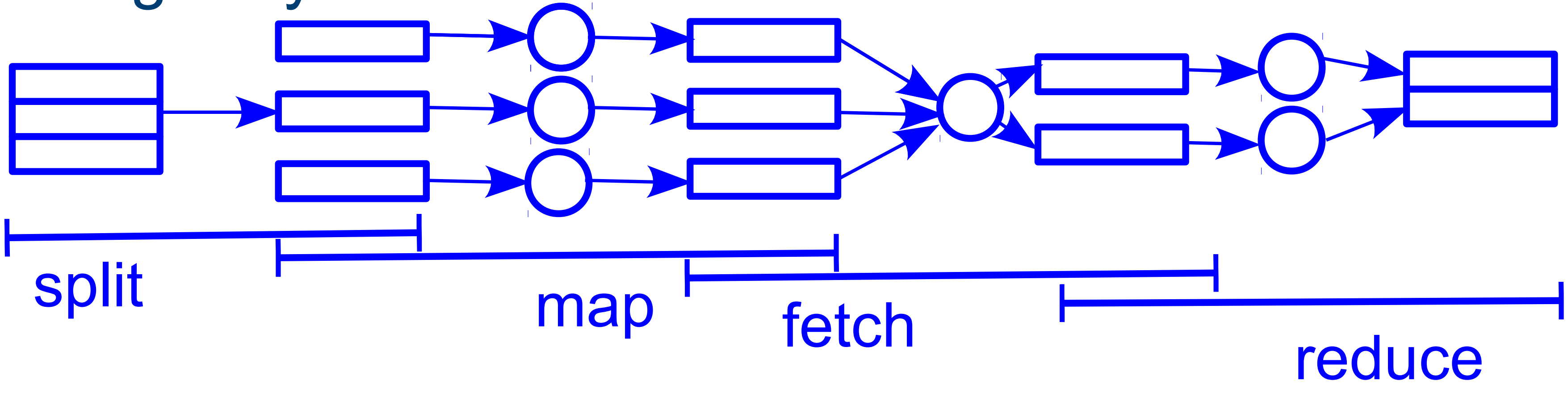
HadoopProv: Towards Provenance As A First Class Citizen in MapReduce

Sherif Akoush, Ripduman Sohan, Andy Hopper

Computer Laboratory

MapReduce: Huh?

- MapReduce: Express computation as:
  - map(key, val) → [(key1, val1)...]
  - reduce([(key1, val1)...]) → [(key, val)...]
- Logically:



HadoopProv: What?

- Provenance support in MapReduce (Hadoop)
  - Key-value tracking in map() and reduce()

- Premise: For any key-value record, what were the key-value pairs involved in its creation?

HadoopProv: Why?

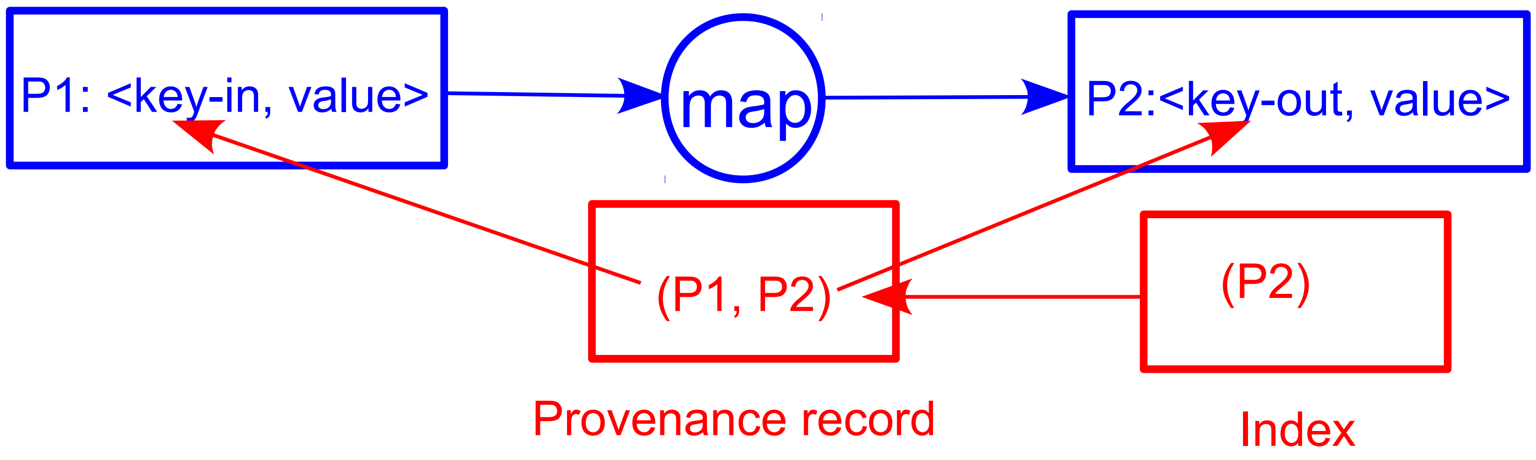
1. Verification, validation of key-pair values
2. Optimize subset processing:
   A) Incremental
   B) Additional
3. Self-tuning system

HadoopProv: What's Different?

1. Tight, transparent framework integration
2. Eager provenance logging
3. No shuffling of provenance metadata
4. Lazy provenance graph construction

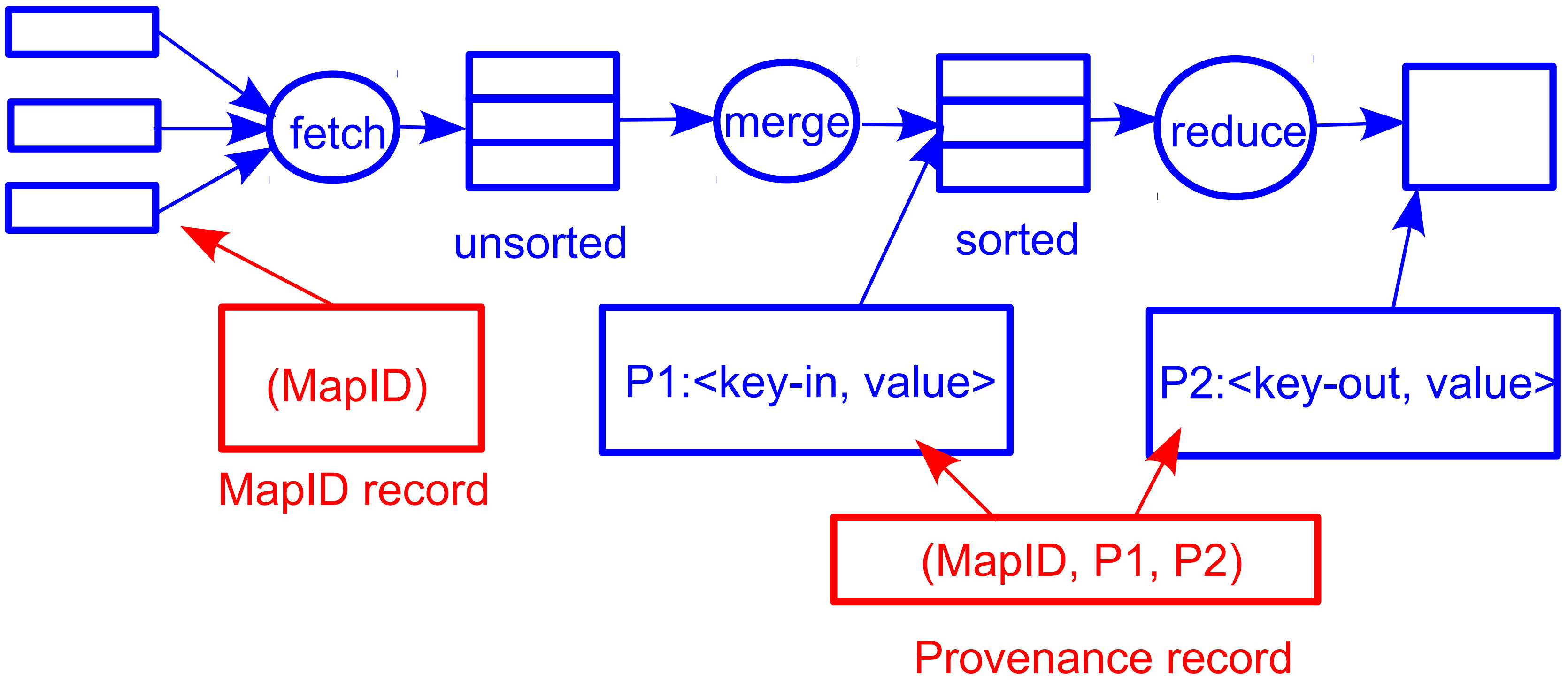
HadoopProv: How?

- map()



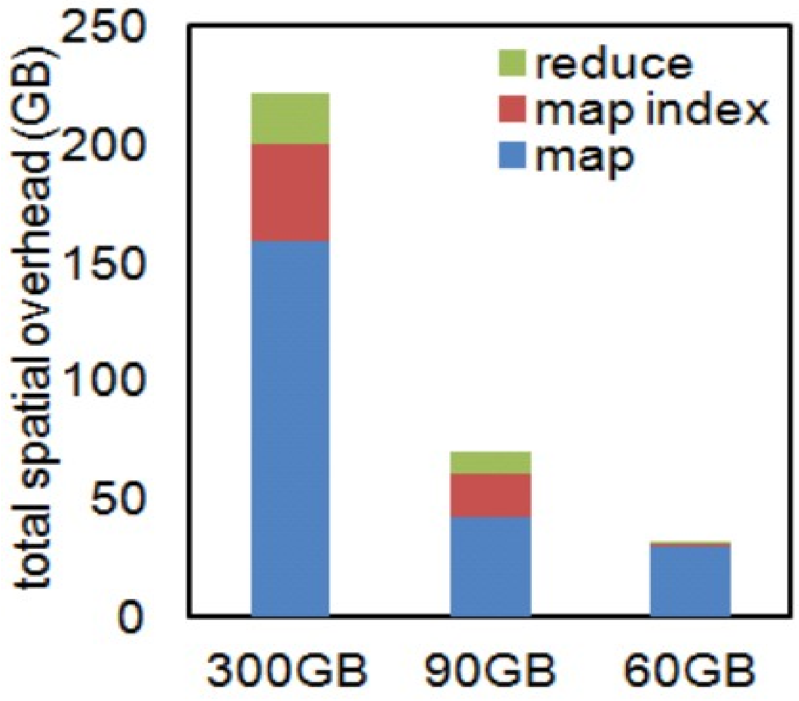
HadoopProv: How?

- reduce()



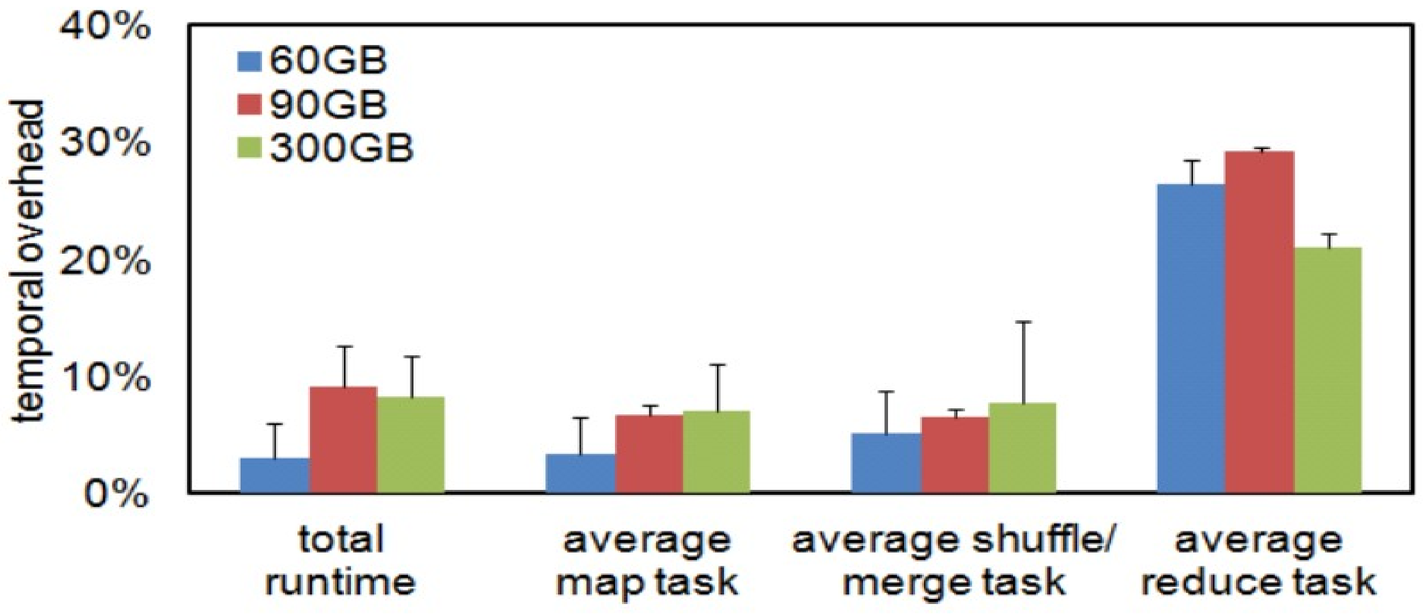
HadoopProv: To What Extent?

- Wordcount: 60, 90, 300 GB Wikipedia subset
- Spatial Overhead



HadoopProv: To What Extent?

- Wordcount: 60, 90, 300 GB Wikipedia subset
- Temporal Overhead



UNIVERSITY OF CAMBRIDGE

HadoopProv: What Next?

- Optimize implementation: Spatial, temporal overhead
- Feedback between provenance and MapReduce phases
- Prove usefulness:
    - Real-world use-cases
    - Trade-off: Re-computation vs Provenance Reconstruction

HadoopProv: Take-Aways

1. Key-value lineage logging (MapReduce) feasible
2. Delaying provenance reconstruction until *absolutely* needed feasible
3. Delayed provenance reconstruction *could* have tangible performance benefits
4. FRESCO @ Cambridge developing these ideas (google “FRESCO + Computer Lab Cambridge”)