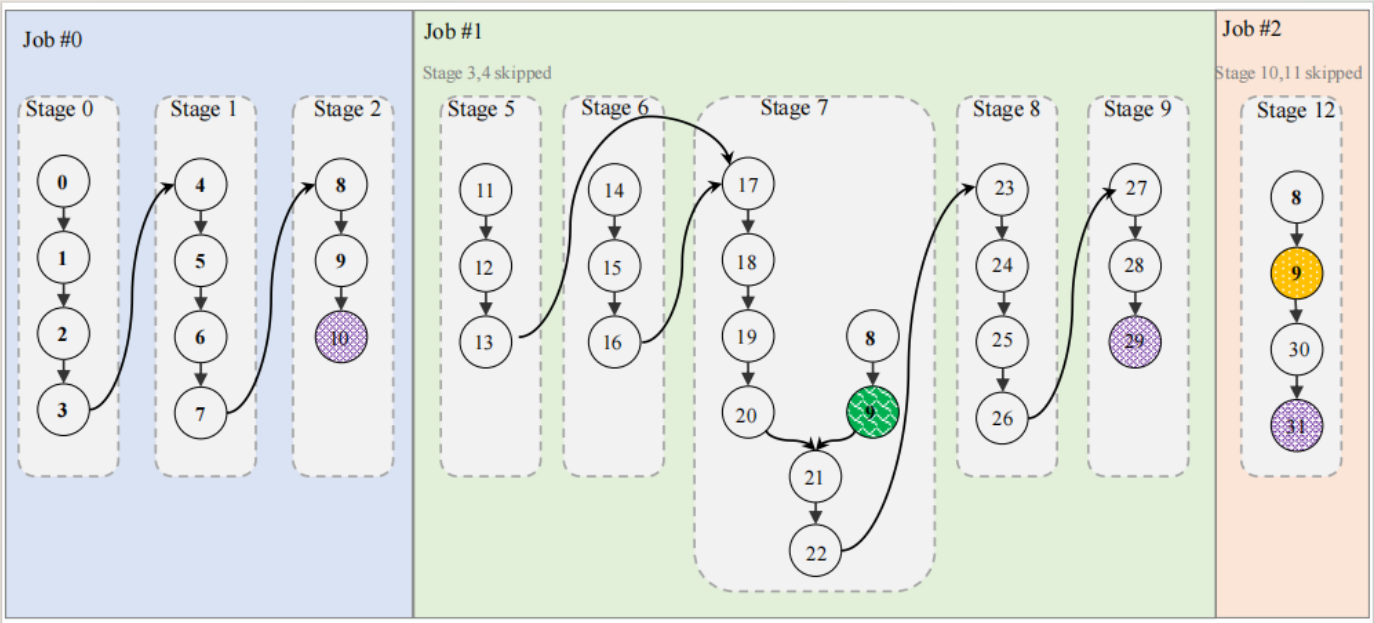
AutoCache: An Online and Automatic Caching Solution for Spark

Hui LI, ShuPing JI, Yang LI, Yujie QIAO, HuaYi SUI, Zhen TANG, Wei CHEN, Zheng QIN, Wei WANG, Hua ZHONG, Tao HUANG

Frontiers of Computer Science, DOI: 10.1007/s11704-025-40776-9

Problems & Ideas

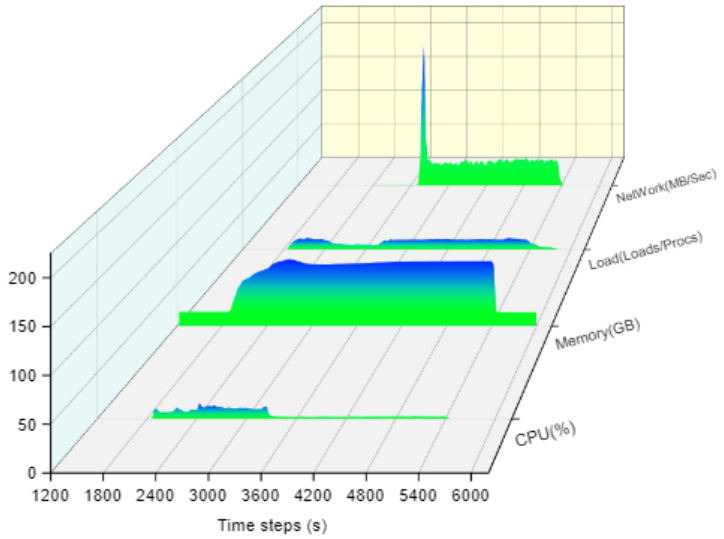
- Problems of conventional Spark Caching:
  - Developers must manually explicitly call the cache API.
  - Existing automatic caching approaches focus on iterative jobs.
- Ideas: A joint execution path profiling and dynamic caching optimization framework that takes both execution time and memory utilization efficiency into account.

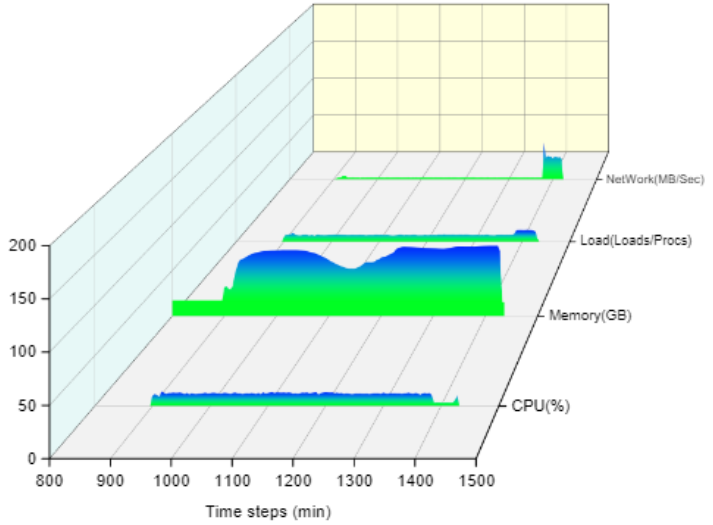


RDD 9 should persist, owing to the max reference count and the largest ID during the execution of Job 1. By reviewing the log, RDD 9 was indeed added in memory (green ripple background) in stage[7] and was hit (yellow background) in memory in stage[12] when executing Job 1.

Main Contributions

- Contributions:
  - AutoCache identifies hot datasets that are likely to be reused at a finer granularity at the stage level, can dynamically evict data in time based on proposed ReuseDistance strategy;
  - AutoCache are evaluated based on real-world applications, nine unknown caching issues is detected by AutoCache in popular third-party libraries are confirmed.
  - AutoCache and the experimental datasets publicly online, which can facilitate further researches in this area.





Results(such as execution time, memory useage etc) yielded by each method under the same application.. Left: Execution of KMeans with K=3 and iteration=1000 with AutoCache; Right: Execution of KMeans with K=3 and Iteration=1000 under NoCache.