Distributed Order Dependency

Sebastian Schmidl, Juliane Waack[1]

Abstract: Abstract goes here.

Keywords: Actor Model; Akka; Distributed Computing; Parallelization

1 Introduction

1.1 Order Dependencies

- What are ODs?
- Why are they useful?
  - Query optimization
  - Data quality (find constraints, help understand semantics)
  - support index selection

1.2 Order Dependency Discovery

What are the challenges?

- Defining Minimality (Unhelpful definition of minimality in Naumann paper, incorrect one in Consonni paper)
- Large searchspace (Long runtime and large memory-needs)
- current agorithms slow on large datasetz
- Therefore need scalability -> distribution

1.3 Distribution

- What are distributed Systems? Why are they useful?

[1] Hasso-Plattner-Institut, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, {sebastian.schmidl, juliane.waack}@student.hpi.de

- What is the actor model? Why use it here?

2 Related Work

The concept of order dependencies was first introduced in the context of database systems by Ginsburg; Hull [GH83] as *point-wise ordering*. Ginsburg; Hull's definition specified that a set of columns orders another set of columns.

Szlichta et al. [SGG12] later introduced another definition for order dependencies, which was used by the following research in this area [Co19; LN16; Sz17] and is also the basis of this work. It differentiates from the point-wise ordering by considering list of columns instead of sets. This leads to a lexicographical ordering of tuples, as by the `order by` operator in SQL.

- Data profiling includes dependency discovery
- FDs, etc. but order dependency more complex through a larger search space than for FDs
- Former solutions by [LN16] (incomplete), [Sz17] and [Co19].
- Our project based on [Co19] (already parallelized with multi-threading). (Explained in 3)

3 Approach

3.1 OCDDISCOVER

- Idea: OD = OCD + FD
- Pruning rules used to build search tree
- Where was their mistake?

3.2 Our architecture

- Whole system
  - Peer-to-peer after seednode shared data
  - Every node holds all data
  - Workstealing
  - State Replication

- Within every Node
  - Master Worker
  - Master holds state (Candidate queue) + does Workstealing
  - Pull-based work distribution
  - State Replicator for sharing and recovering states (via streaming)
  - Data Holder for holding and sharing data (via streaming)
  - ClusterListener to support node to node communication
  - ResultCollector saves results to file (still need to be merged and reduced to unique results in the end)
  - Reaper

3.3 Communication Protocols

3.3.1 Work Stealing Protocol

3.3.2 Downing Protocol

3.3.3 State Replication Protocol

4 Evaluation

- What to test? (see testing-strategy)
- Testing setup
- Tests and results

References

[Co19] Consonni, C.; Sottovia, P.; Montresor, A.; Velegrakis, Y.: Discoverying order dependencies through order compatibility. In: International Conference on Extending Database Technology. 2019.

[GH83] Ginsburg, S.; Hull, R.: Order dependency in the relational model. Theoretical computer science 26/1-2, pp. 149–195, 1983.

[LN16] Langer, P.; Naumann, F.: Efficient order dependency detection. The VLDB Journal—The International Journal on Very Large Data Bases 25/2, pp. 223–241, 2016.

[SGG12] Szlichta, J.; Godfrey, P.; Gryz, J.: Fundamentals of order dependencies. Proceedings of the VLDB Endowment 5/11, pp. 1220–1231, 2012.

[Sz17] Szlichta, J.; Godfrey, P.; Golab, L.; Kargar, M.; Srivastava, D.: Effective and complete discovery of order dependencies via set-based axiomatization. Proceedings of the VLDB Endowment 10/7, pp. 721–732, 2017.