Distributed Order Dependency

Sebastian Schmidl, Juliane Waack[1]

Abstract: Abstract goes here.

Keywords: Actor Model; Akka; Distributed Computing; Parallelization

1 Introduction

1.1 Motivation

Uses for ODs [SGG12]:

- Query Optimization
- Data quality (integrity constraints)
- Index selection

Challenges:

- Inference co-NP-complete [GH83]
- best known algorithms between $O(n!)$ [Co19] and $O(2^n)$ [Sz17] with n being the number of attributes

Use distribution to improve runtime and memory usage, because SotA-algorithms take too long or exceed memory limits (>5h, >110GB for FLIGHT_1K-dataset)

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.

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

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] (unknown programming error) and [Co19].
- Our project based on [Co19] (already parallelized with multi-threading).

3 Approach

- collect and persist partial results
- master worker pattern
- at the beginning permutation tree stored only on master (split if it gets too big)
- parallelize additional parts of the algorithm (initial pruning)
- reaper pattern for clean shutdown
- three options for holding the input table:
  - **full replication**
  - split column wise
  - split row wise
- for the architecture see Figure 1

4 Evaluation

Use same datasets as previous works for comparison (see table 6 from [Co19], was taken from HPI homepage). The link in the paper was broken, this is the updated one: `https://hpi.de/naumann/projects/repeatability/data-profiling/fds.html`.

1. Find same number of ODs than related approaches (quality metric). If we have a gold standard: compare results to gold standard.

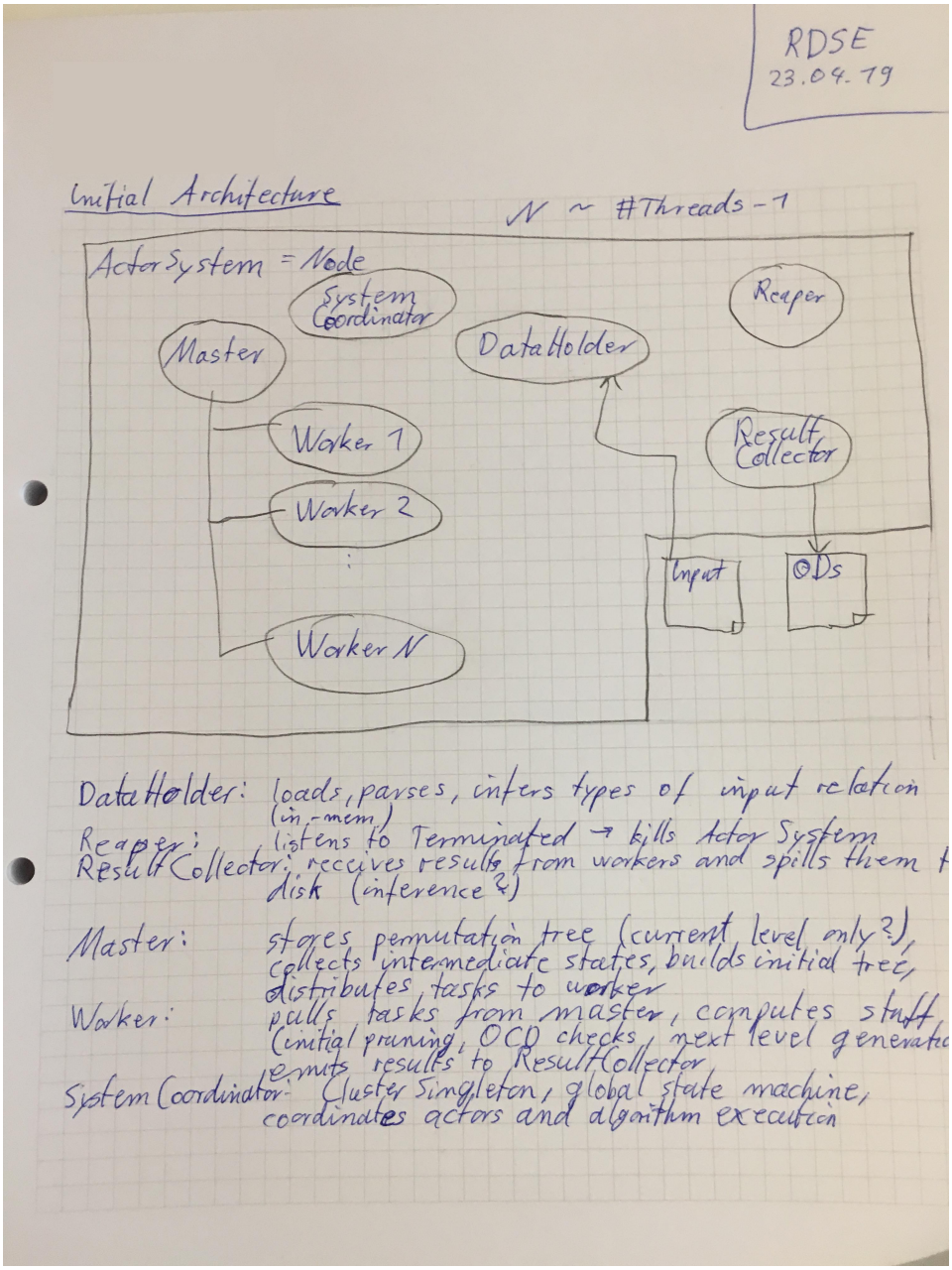

Fig. 1: Actor architecture

2. Measure single node time (slightly higher than OCDDISCOVER)
3. Test robustness via killing nodes or jamming network.
4. Measure scaling across columns, rows.
5. Measure scaling across nodes (cores).

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.