A comparison of the incremental computation platforms Naiad and TBD

Walter Tichy, Umut Acar, Emanuel Jöbstl
Karlsruhe Institute of Technology, Faculty of Computer Science
Carnegie Mellon University, Computer Science Department

***Abstract*—Abstract - What is this about?**

***Keywords*—Keywords.**

I. Introduction

This section should describe the purpose of incremental computation and also explain for which purpose the concept of incremental computation is useful.

A. Approaches to incremental computation

This subsection should describe various historic approaches to incremental computation. This includes, but is not limited to:

B. Naiad and TBD

This subsection should shortly outline Naiad and TBD. Additionally, the choice of comparing Naiad to TBD should be explained.

II. NAIAD

This section should outline the Naiad framework and it's purpose.

A. Approach

This subsection should shortly explain the approach Naiad uses, especially in regard of incremental computation. In other words, parallel and distributed features of Naiad should not be described in depth, because they are not a matter of interest for this theseis.

B. Programming model

This subsection should describe the programming model of Naiad. The description should be from the view of a developer who creates software atop of the platform. The concepts and used programming patterns should be explained clearly, also an example should be provided.

Frameworks build on top of Naiad, for example the Liandi framework should be mentioned.

III. TBD

This section should outline the TBD framework.

A. Approach

This subsection should describe the approach TBD uses, especially on how memoization and the tracing of dependency graphs can be used to achieve incremental computation.

B. Programming model

This subsection should describe the TBD core API. Since TBD relies on the use of patterns known from functional languages, these patterns should be described. Also, the constraints for and the responsibilites of the developer have to be specified. The section should be concluded with an example.

C. Prerequisites for implementing performant algorithms on top of TBD

This subsection should provide a very short outline to the term *stable algorithm*.

IV. Conducted experiments

This section should outline important properties of algorithms used for this benchmark, and why these features are important.

Note: The testcases listed here are subject to change.

Each subsection should describe the differences and challenges of the implementations for Naiad and TBD.

A. Wordcount

Wordcount was choosen as a benchmark because many machine-learning approaches rely on wordcount on big data. Furthermore, wordcount serves as a prime example of an algorithm implemented on top of map-reduce.

B. Sorting numbers

A sorting algorithm (quicksort or mergesort, has yet to be decided) was chosen as benchmark because it introduces a special form of difficulties for incremental computation: If the input of a sorting operation changes, the dependency graph is expected to change.

C. Pagerank

Pagerank is an algorithm highly relevant for real-world applications. Futhermore, it is also prone to cascading updates, which makes it an interesting benchmark for incremental computation platforms.

V. Benchmark results

This section should clearly describe the testing environment. Also, the input data to the tests should be characterized.

A. Speed

This section should describe the results in regard to processing time. If there are strong deviations during testing, the cause should be explained.

B. Memory

This section should describe the results in regard to memory usage. This section should also expand on anomalies.

C. Code overhead

Additionally to processing speed and memory usage, the difference of code overhead between the platform dependend implementations should be measured.

VI. Conclusion

This section should conclude with the findings of the executed benchmarks. Furthermore, it should provide a guideline for choosing one of the two frameworks, depending on which type of algorithm is to be implemented.

A. Future work

The final section should shortly outline problems encountered but not solved during the writing of this theses, as well as encourage future research on interesting issues of incremental computation.