Pregelix: Dataflow-Based Big Graph Analytics

Yingyi Bu
University of California, Irvine
yingyib@ics.uci.edu

1. PROBLEM

Recently, Google has proposed the Pregel programming model [2] for Big Graph analytics, where application programmers need no knowledge of parallel or distributed systems. Instead, they just need to "think like a vertex" and write a few functions that encapsulate the logic for what one graph vertex does. The vertex-oriented programming model has been found to ease the implementation of distributed graph algorithms to a great extent.

There are several efforts (e.g., [1, 4]) to build Pregel-like systems from scratch. However, building such a system is complicated because a complete implementation must consider network issues, memory management, message delivery, parallel task scheduling, fault-tolerance, vertex storage, and out-of-core support. Moreover, if the system developer wants to support different runtime execution choices, doing that in the built-from-scratch world would add still more complexity.

2. THE PREGELIX APPROACH

In this work, instead of building a Pregel system from scratch, we explore an architectural alternative — expressing Pregel's semantics as database-style dataflows and executing them on a general-purpose data-parallel engine using classical parallel query evaluation techniques. We have used this approach to build Pregelix — a dataflow-based Pregel implementation for Big Graph analytics. Pregelix examines the value of a "one-size-fits-a-bunch" philosophy, demonstrating that a general execution engine can also support Big Graph analytics well.

SoCC'13, 1–3 Oct. 2013, Santa Clara, California, USA.
ACM 978-1-4503-2428-1. http://dx.doi.org/10.1145/2523616.2525962

The design and implementation of Pregelix explore and answer two research questions: 1. In order to efficiently support Big Graph analytics, what the necessary extensibility does a data-parallel query execution engine need to provide? 2. Given an extensible data-parallel execution engine, how should one implement Pregel on top of it and what are the possible optimizations?

For the first research question, there are two extensibility requirements for the query execution engine:

- Support for user-defined operators, such as reading/writing from distributed file systems and setting up user-defined (Pregel-specific) job contexts,
- User-configurable task scheduling, with which one can implement sticky scheduling to leverage locally cached data on each cluster machine.

Based on these two requirements, we chose Hyracks [3] as the runtime execution engine for Pregelix. Regarding the second research question, we have been exploring several key design issues:

- Pregel as dataflows. Inside Pregelix, both vertices and messages are treated as tuples and Pregel's semantics are implemented by a query plan consisting of operators such as joins and grouped aggregations.
- Vertex storage. We store vertices in partitioned B-trees, either using standard B-trees or LSM-trees (log-structured merge-trees).
- Runtime choices. Pregelix offers several logically equivalent runtime execution alternatives and allows users to choose among them. Each alternative is implemented as data-parallel jobs.

3. KEY CONTRIBUTIONS

This work contains several key contributions. First, it demonstrates the feasibility of using a database-style approach to implement the Pregel model in a simple architecture. Second, we identified the extensibility requirements for a parallel query execution engine to support Big Graph analytics. Last but not least, we have released the Pregelix system (http://hyracks.org/projects/pregelix/) in open source form for use by the community.

4. REFERENCES

[1] Giraph. `http://giraph.apache.org/`.

[2] Grzegorz Malewicz et al. Pregel: a system for large-scale graph processing. In *SIGMOD*, 2010.

[3] Vinayak R. Borkar et al. Hyracks: A flexible and extensible foundation for data-intensive computing. In *ICDE*, 2011.

[4] Yucheng Low et al. Distributed GraphLab: A framework for machine learning in the cloud. *PVLDB*, 2012.