PegasusN

Developer guide

January 11, 2018

Contents

1 Overview

1.1 General Information

- PegasusN: Peta-Scala Graph Mining System
- Version: 3.0
- Date: Jan. 11st, 2018
- Authors: Chiwan Park, Ha-Myung Park, U Kang

PegasusN is a Peta-scale graph mining system on hadoop and spark. It computes the degree distribution, PageRank, RWR(Random Walk with Restart) scores, radii/diameter, connected components, subgraphs that match a query graph including a triangle from very large graphs with more than billions of nodes and edges.

2 Building PegasusN

2.1 Building with SBT

PegasusN uses SBT (Simple Build Tool) for managing depedencies and builds. Building PegasusN requires SBT v0.13+ and Java 8.

PegasusN comes packaged with a self-contained SBT installation to ease buliding and deployment of PegasusN from source located under `tools` directory. This script will download and setup all necessary depedencies and build requirements (SBT and Scala) locally within the home directory.

`tools/sbt` execution acts as a pass through to the `sbt` call allowing easy transition from previous build methods. As an example, the user can build PegasusN as follows:

```
./tools/sbt assembly
```

2.2 Specifying the Hadoop Version

You can specify the exact version of Hadoop to compile against through `hadoop` variable in `build.sbt` file. In default, PegasusN uses Hadoop 2.7.3 as a default Hadoop version.

2.3 Testing with SBT

The following is an example of a command to run the tests:

```
./tools/sbt test
```

3 Adding a New Algorithm on Matrix-Vector Multiplication Framework

3.1 Overview

The user can create a new algorithm based on the matrix-vector multiplication. PegasusN provides a generalized matrix-vector multiplication framework that needs only few user-defined methods to describe an iterative graph mining algorithm.

3.2 Types of Data

To create a new algorithm based on matrix-vector multiplication, you have to declare the type of matrix block, vector block, and partial vector block. For example, in PageRank algorithm, the type of block of matrix, vector and partial vector is `DoubleWritable` that is provided by Hadoop. The types of data can be different each other. For example, in label propagation algorithm, the type of element of matrix is `UnitWritable`, but the type of element of vector and partial vector is `LongWritable` that represents a community.

The user can define new type class for the data. Note that the user-defined type should implement the `Writable` interface which is provided by Hadoop.

3.3 Describing Algorithm on Matrix-Vector Multiplication

To describe an algorithm on matrix-vector multiplication in PegasusN, the user needs to implement the following methods:

- `createVectorBlock` - initializes a vector block with initial values. For example, in PageRank algorithm, this methods sets a value of each vector element to $1/|V|$.
- `createIntermediateVectorCache` - initializes a partial vector block with initial values. For example, in shortest path computation algorithm, this method sets a value of each vector element to ∞.
- `combine2Op` - given a matrix block and a vector block, combines them and produces a partial vector block.
- `combineAllOp` - given two partial vector blocks, combines them and produces a partial vector block.

- `applyOp` - given a vector block, and a combined partial vector block, computes a new vector block.
- `setupComputation` (optional) - initializes shared variables between workers.
- `cleanupComputation` (optional) - finalizes the iterative computation. For example, in diameter estimation algorithm, this methods computes the estimated diameter given the computed bit-vector blocks from the iterative matrix-vector multiplication.

4 Contributing to PegasusN

Follow these steps to contribute code changes to PegasusN:

- Download and extract the source code of PegasusN.
- Make the changes necessary for your particular issue. Try to avoid unnecessary changes such as extra whitespace or formatting changes. Include unit test, or be ready to justify why one is not necessary.
- Verify the new and existing tests continue to pass via `tools/sbt test checkstyle`. This SBT command will check the code style and run unit tests.
- Generate patch with `git diff --no-prefix master > PATCH_NAME.patch`. For subsequent patches, if necessary, number each version to make it easier for reviewers to track their progress.
- Send the patches with brief explanation what changes it contains and what testing was done to our main contributors via email.
- The one of main contributors should review the patch shortly and either provide feedback for a new version, or commit it to the PegasusN source.