Partitioning

Partition Function

In Hama BSP computing framework, the Partition function is used for obtaining scalability of a Bulk Synchronous Parallel processing, and determining how to distribute the slices of input data among BSP processors. Unlike Map/Reduce data processing model, many scientific algorithms based on Message-Passing Bulk Synchronous Parallel model often requires that a processor obtain “nearby or related” data from other processors in order to complete the computation. In this case, you can create your own Partition function for determining processor inter-communication and how to distribute the data.

Internals

Internally, File format input data partitioning works as following sequence:

- If user specified partition function, internally, "partitioning job" is ran as a pre-processing step.
  - Each task of "partitioning job" reads its assigned data block and rewrite them to particular partition files.
  - Job Scheduler assigns partitions to proper task based on partition ID.
- After pre-partitioning done, launch the BSP job.

In NoSQLs table input case (which supports range or random access by sorted key), pre-processing step will be skipped because they supports range scan.

- Job Scheduler assigns Scanner or tablet with its partition ID to proper task, launch the BSP job.

Partitioning internals in Graph module

The internals of the Graph module implemented on top of BSP framework, are pretty simple. Input data partitioning will be done at BSP framework level. Each GraphJobRunner processors just reads assigned splits, and loads parsed vertices into vertices storage at loadVertices() method. If you want to learn details and internals about Graph job, Please see also Design of Graph Module.

Create your own Partitioner

Tutorial

....

```
BSPJob job = new BSPJob(conf);
...
job.setPartitioner(HashPartitioner.class);
...
```

Specify the partition files and directories

If the input is already partitioned, you can skip pre-partitioning step as following configuration:

```
...
```