Hamburg

Rationale

Motivation

The MapReduce (M/R) programming model is inappropriate to problems based on data where each portion depends on many other potions and their relations are very complicated. It is because these problems cause as follows:

- limit to assigning one reducer
    - In case that the relations of data are very complex, assigning intermediate data to appropriate reducers by considering their dependency of partitioned graphs may be very hard. Assigning only one reducer is a straightway to solve complexity dependency, but it is apparent to cause deterioration of scalability.
- many M/R iterations
- or make an M/R program more complicated
    - To avoid above two inefficient methods, the M/R program will be complicated with code to communicate data among data nodes.

These problems are very common in many areas; especially, many graph problems are exemplary.

Therefore, we try to propose a new programming model, named Hamburg. The main objective of Hamburg is to support well the problems based on data having complexity dependency one another. This page is an initial work of our proposal.

Goal

- Follow scalability concept of shared-nothing architecture
- Support a simple programming model to compute complex relations such as, graph data.

Hamburg

Hambrug is an alternative to M/R programming model. It is based on bulk synchronization parallel (BSP) model. Like M/R, Hambrug takes advantages from shared-nothing architecture (SN), so I expect that it will also show scalablity without almost degradation of performance as the number of participant nodes increases. A Hamburg based on BSP computation step consists of three sub steps:

- Computation on data that reside in local storage; it is similar to map operation in M/R.
- Each node communicates its necessary data into one another.
- All processors synchronize which waits for all of the communications actions to complete.

Let's see more detail in the diagram of computing method of Hamburg based on BSP model.

http://lh4.ggpht.com/_DBxyBGtfa3g/SmQUYTHWooI/AAAAAAAABmk/cFVlLCdLVHE/s800/figure1.PNG

Each worker will process the data fragments stored locally. And then, We can do bulk synchronization using collected communication data. The 'Computation' and 'Bulk synchronization' can be performed iteratively, Data for synchronization can be compressed to reduce network usage. The main difference between Hamburg and M/R is that Hamburg does not make intermediate data aggregate into reducer. Instead, each computation node communicates only necessary data into one another. It will be efficient if total communicated data is smaller then intermediate data to be aggregated into reducers. Plainly, It aims to improve the performance of traverse operations in Graph computing.

For example, to explores all the neighboring nodes from the root node using Map/Reduce (FYI, Breadth-First Search (BFS) & MapReduce), We need a lot of iterations to get next vertex per-hop time.

Let's assume the graph looks like presented below:

http://lh5.ggpht.com/_DBxyBGtfa3g/SmQTwhOSGwI/AAAAAAAABmY/ERiJ2BUFxI0/s800/figure2.PNG

The root node is 1. Then, we need only one 'Bulk synchronization' between server2 and server3 with Hamburg. Rests will be calculated on local machine.

http://lh6.ggpht.com/_DBxyBGtfa3g/SmQTwvxT2zI/AAAAAAAABmc/BRrv7plzPtc/s800/figure3.PNG

Initial contributors

- Edward J. (edwardyoon AT apache.org)
- Hyunsik Choi (hyunsik.choi AT gmail.com)

Any volunteers are welcome.

Implementation