Cluster, Nodes and Allocations

Cluster and nodes

As said in the Using BigGrph document, BigGrph clusters are made of one master node and a set of slave nodes. The master node executes the main program and send computations to the slave nodes.

In a BigGrph cluster, a slave is actually a java process that executes as its main program the function `bigobject.BigObjectNodeMain.main(List<String>)` located in project `ldjo`. This main program starts two TCP servers listening on dedicated ports :

1. The first port is used for an instance of `octojus.TCPRequestServer` which is a remote computation server. Computations to be executed are instance of a subclass of `octojus.ComputationRequest<ReturnType>`. They are serialized in order to be transmitted over the TCP stream, deserialized and executed on the receiving node. The port that this server use is the first command line argument of the slave java process. This port is named the *Computation Request Port*.

2. The second port is used by another TCP server, instance of `bigobject.BigObjectServer`. Purpose of this server is to process requests sent between local instances of distributed objects (see the document on BigObjects for details). This port is called the *Bigobject Port*.

A BigGrph cluster node is uniquely defined by the hostname on which the java process is run, plus the two port numbers above. This means that on a host, it is possible to run several BigGrph slaves, as soon as they use different TCP port numbers for both of two servers. In BigGrph logs, cluster nodes are printed using this three information separated by the `:` character.

Example:

- `localhost:43567:57312` is a slave node running on the localhost and listening on two TCP ports, 43567 where it receives remote computations to execute, and 57312 where it receives service requests between instances of distributed objects. If another slave node was to be run on the same local host, it should use different port numbers.

Distribution of data and Allocations

A large part of data handled by BigGrph involve association between graph vertices and other relevant information. For example, the base graph representation is a distributed adjacency table, where each vertex, represented by its identifier, a long (64 bits) number, is associated to the list of neighbors of the vertex.

The basis of the distribution of data in BigGrph is to provide a function that given a long number, outputs the cluster node on which the data associated with the number is stored.

The class Hierarchy starting at `bigobject.Allocation` is where such a function is implemented. All `Allocation` and subclasses instances are created from the list of slave nodes of a cluster.

For BigGrph associative maps where the key is a long number, the class of interest is a subclass of `dht.DHTAllocation` (in project `ldjo`). This class introduces the function `OctojusNode getOwnerNode(long e)` as an abstract function that must be implemented in subclasses.

The default `Allocation` object in BigGrph is an instance of `dht.allocation.BasicHashDHTAllocation` where the computation of the storage node is based on a simple modulus operation from the long number representation of the vertex. This crude mechanism is arbitrary and deterministic.

If you want to design your own algorithm for the distribution of data, you have to create a subclass of `dht.DHTAllocation` and implement the `getOwnerNode()` function. Take care that `Allocation` objects are serializable and are transfered between cluster nodes when distributed objects are created (see document on BigObjects for details).