Technical Report of Distributed SF1

Zhongxia Li

April 16, 2012

Date	Author	Notes
2012-04-16	Zhongxia Li	Initial version.

Abstract

This report describes the initial design of the distributed SF1 system, including distributed topology and node management in association with ZooKeeper, distributed search and index, also mentions some further issues. Compared to distributed search we also have distributed recommendation, but we just mention distributed search in this report because they are very similar. The distributed SF1 will be gradually extended and optimized to adapt the requirements of our distributed application.

Contents

1 Distributed Topology

1.1 Overview

We firstly go through the initial design for distributed SF1R, and make clear about some terminologies.

- **Cluster** A Cluster is a group of distributed SF1 servers (nodes) coordinated to provide search service.

- **Node** A node, or SF1 node, refers to a running SF1 process in a cluster of distributed SF1 system. It's possible that more than one SF1 nodes run on one machine, though it's unusual. Typically, we deploy one SF1 node on one machine, and in this case each host or machine in network is a node.
- **Replica** For a cluster of nodes, we may have one or more replication(s) to support failover, we call each replicated cluster a Replica.
- **Replica Set** For each node, we may have one or more backup node(s) in different replications, we call each set of reduplicated nodes a Replica Set.
- **Master** A SF1 node can work as Master or Worker or both. A Master is the controller and router of a cluster of distributed search servers (nodes), it receives all the outside query requests, dispatch requests to sub servers and gather results.
- **Worker** Compared to Master, each search server (node) is a Worker, which provide partial search data and response for partial search result.
- **Data Shard** In a distributed SF1, search data will be partitioned to different Data Shards. Currently each data shard will reside on each one Worker, so we also use shard ids to identify workers.

As an example of distributed topology, see figure 1, we have three nodes in the cluster of distributed SF1, and there are two Replicas.

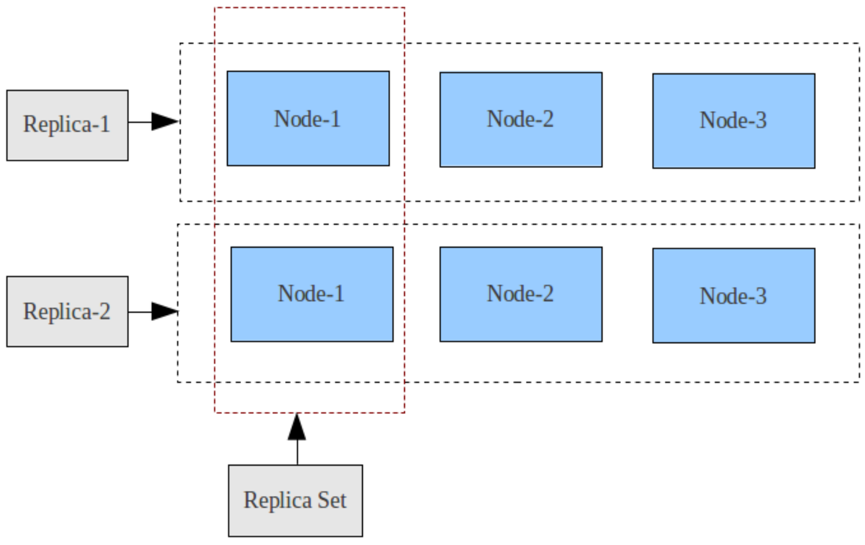

Figure 1: Example of Distributed Topology

1.2 Configuration of distributed SF1

The distributed SF1 is aimed at providing a flexible and configurable distributed solution, we need to configure each SF1 node (by editing the sf1 configuration file) to deploy a cluster of distributed SF1.

Commonly, we have to set *clusterid* for a SF1 node first to specify which cluster it will join in.

To configure current SF1 node as **Master**, we can enable *MasterServer*, notice that the distributed SF1 is collection related, which means different collections can have different distribution cases, we can see that a collection supported by a Master can be distributive or not. A undistributed collection will reside on local host only, while distributed collections can reside on different set of Workers.

To configure current SF1 node as **Worker**, we can enable *WorkerServer*, we have to specify the shard id which indicates which part of data the Worker will keep, and specify the collections it will support.

As an example, we'll give the configurations of an application scenario of distributed SF1 showed in figure 2. Considering that we have three SF1 server nodes, node1 and node2 work as Master respectively, and they all work as Workers.

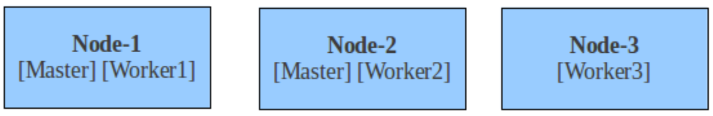

Figure 2: An exmaple cluster of Distributed SF1

- Configuration for Node1

```
<DistributedCommon clusterid="sf1-example" username="lscm"
   localhost="172.16.0.36" workerport="18151" masterport↩
       ="18131" datarecvport="18121" />

<DistributedTopology enable="y" type="search" nodenum="3">
  <CurrentSf1rNode nodeid="1" replicaid="1">
    <MasterServer enable="y" name="undefined" shardnum="3">
      <Collection name="web" distributive="y" shardids↩
          ="1,2,3" />
      <Collection name="b5mo" distributive="n" />
      <Collection name="b5mp" distributive="y" shardids="1,2"↩
           />
      <Collection name="b5mc" distributive="y" shardids="2,3"↩
           />
    </MasterServer>

    <WorkerServer enable="y" shardid="1">
      <Collection name="web" />
      <Collection name="b5mp" />
      <Collection name="b5mc" />
    </WorkerServer>
  </CurrentSf1rNode>
</DistributedTopology>
```

- Configuration for Node2

```
<DistributedCommon clusterid="sf1-example" username="lscm"
    localhost="172.16.0.161" workerport="18151" masterport↩
        ="18131" datarecvport="18121" />

<DistributedTopology enable="y" type="search" nodenum="3">
  <CurrentSf1rNode nodeid="2" replicaid="1">
    <MasterServer enable="y" name="undefined" shardnum="3">
      <Collection name="web" distributive="y" shardids↩
          ="1,2,3" />
      <Collection name="qa" distributive="n" />
      <Collection name="b5mp" distributive="y" shardids="1,2"↩
           />
      <Collection name="b5mc" distributive="y" shardids="2,3"↩
           />
    </MasterServer>

    <WorkerServer enable="y" shardid="2">
      <Collection name="web" />
      <Collection name="b5mp" />
      <Collection name="b5mc" />
    </WorkerServer>
  </CurrentSf1rNode>
</DistributedTopology>
```

- Configuration for Node3

```
<DistributedCommon clusterid="sf1-example" username="lscm"
    localhost="172.16.0.162" workerport="18151" masterport↩
        ="18131" datarecvport="18121" />

<DistributedTopology enable="y" type="search" nodenum="3">
  <CurrentSf1rNode nodeid="3" replicaid="1">
    <WorkerServer enable="y" shardid="3">
      <Collection name="web" />
      <Collection name="b5mp" />
      <Collection name="b5mc" />
    </WorkerServer>
  </CurrentSf1rNode>
</DistributedTopology>
```

2 Distributed Node Management

2.1 ZooKeeper

Apache ZooKeeper is a high-performance coordination service for distributed applications, and it was employed in the distributed SF1 as distributed coordination service. More detailed information about ZooKeeper can be referred at the Apache web site [ZooKeeper].

The key data structure that used in ZooKeeper for coordination tasks is the *ZooKeeper Namespace*, the ZooKeeper Namespace defined for distributed SF1 is showed below.

The namespace kept in ZooKeeper service provides a consistent view of the topology of the whole distributed system, e.g., the *znode* identified by path */SF1R-[clusterid]/SearchTopology/Replica1/Node1* reflects the status of *Node1* in the cluster, and we can set data to znodes to share information or status among all the distributed nodes.

- *ZooKeeper Namespace for Distributed SF1*

```
/
|--- SF1R-[clusterid]
      |--- SearchTopology
            |--- Replica1
                  |--- Node1  (Master/Worker)
                  |--- Node2  (Master/Worker)
                  |--- Node3  (Worker)
            |--- Replica2
            ...
      |--- SearchServers       # A search server is a master
            |--- Server00000000
            |--- Server00000001

      |--- RecommendTopology
            |--- Replica1
            |--- Replica2
      |--- RecommendServers
            |--- Server00000000
            |--- ...
```

- *Communication with ZooKeeper*

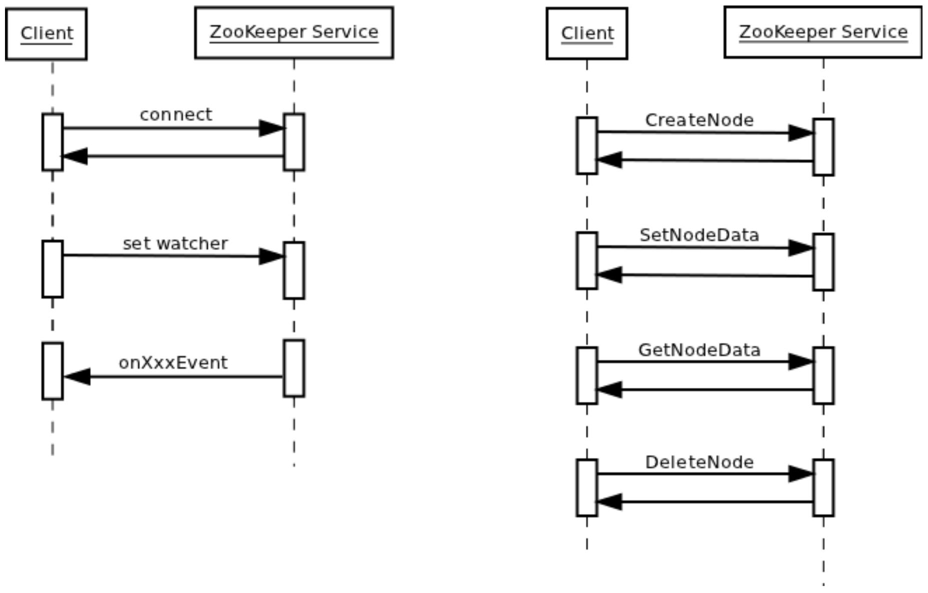

Figure 3: Basic communication operations with Zookeeper

- *Distributed coordination using ZooKeeper*

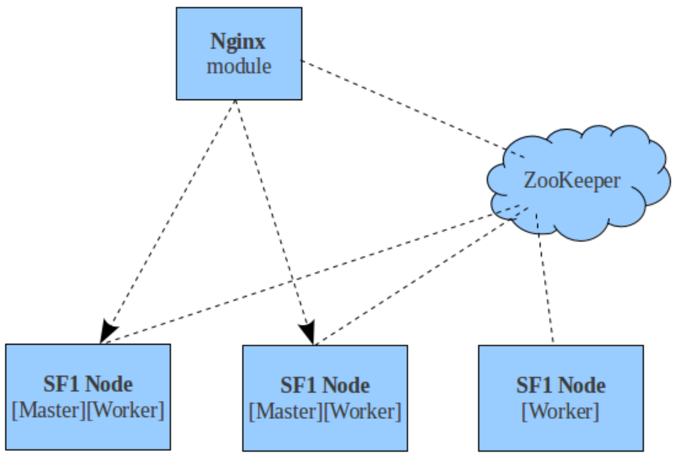

Figure 4: Coordination with Zookeeper

2.2 Node Manager

In distributed scenario, the node manager component of SF1 server performs node management, it registers current SF1 node to the namespace of ZooKeeper service to share info and detects other nodes of the cluster. The following shows the structure and sequences of node manager.

- *Node Manager*

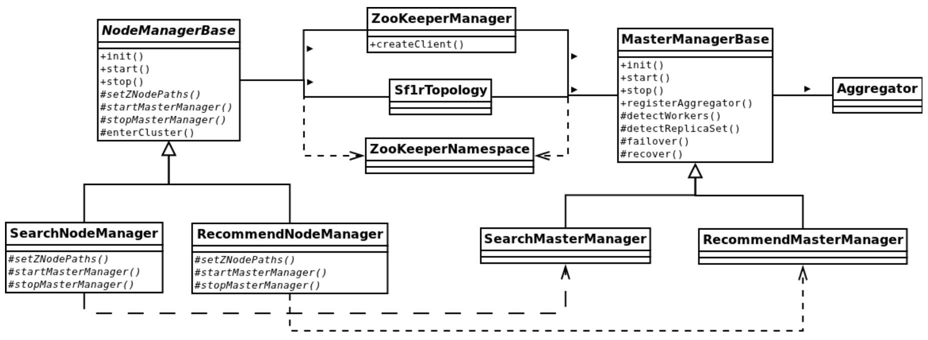

Figure 5: Class diagram of node manager

- *Startup of Node Manager*

  If SF1 server was configured as distributed SF1 node, it will start node manager to register itself into cluster. If it was also configured as Master server, it will start master manager to detect Workers in cluster and perform some related tasks, there are also some *further* tasks such as *failover* and *recover*.

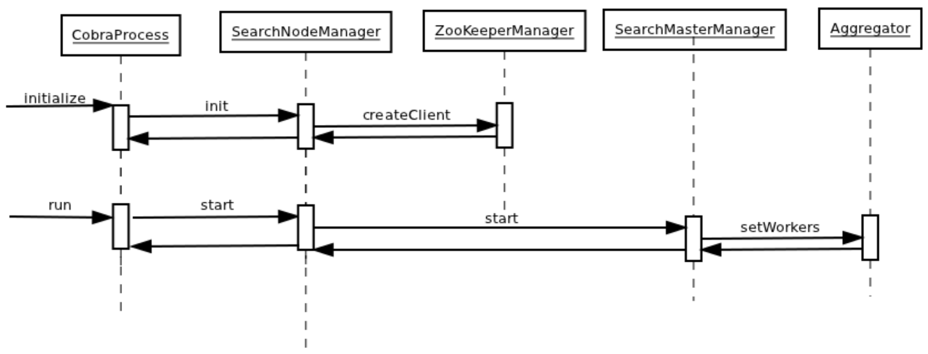

Figure 6: Sequence of startup of node manager

3 Distributed Search

In distributed SF1, all search requests will be sent to Masters in the cluster. When Master received a search request, basically it will dispatch the request to all the Workers and then aggregate (merge) results returned from different Workers, these works are performed through a *Aggregator* framework.

As an exmaple, in figure 7, we showed how *keyword search* is processed as a distributed search. For other kinds of requests, the basic processes are more or less the same.

Note that if a Worker works on the same SF1 node as Master, it will perform local function call to the local Worker instead of sending rpc request for efficiency.

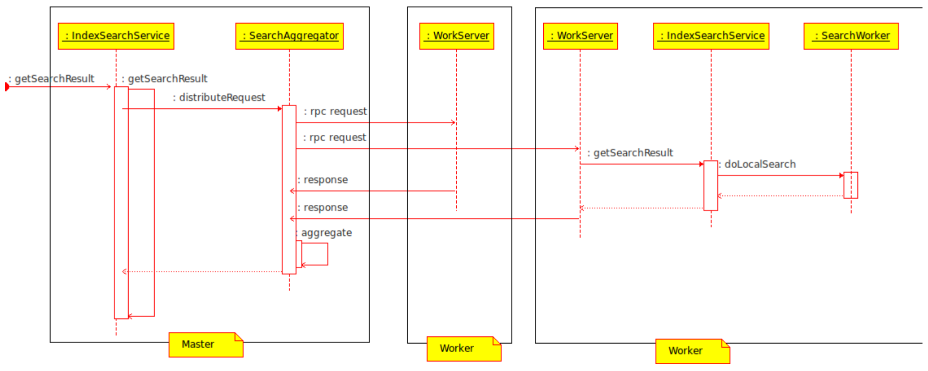

Figure 7: Distributed keyword search

4 Distributed Index

In distributed SF1, search data will be partitioned and dispatched to multiple Workers (each data shard resides on each Worker) and the key problem is *Data Sharding* or *Data Partitioning*, concretely we'll partition SCDs which will be indexed by SF1. Each document in SCD is recorded with properties in key-value pairs, also we can treat the whole document as a key-value pair by using a specified key. SCDs for distributed SF1 will be partitioned by Master, and dispatched to related Workers, as show in figure 8.

Generally, there are some strategies for this hashing like partitioning, such as *consistent hashing* which has the essential property that removal or addition of one node changes only the set of keys owned by the nodes with adjacent IDs, and leaves all other nodes unaffected. *Locality-preserving hashing* ensures that similar keys are assigned to similar nodes, this can enable a more efficient execution of range queries. We may want to consider sharding strategy with issues of node change, range query, and relation between different collections. But it's hard to satisfy all the needs in a strategy, so we may have to weigh the advantages and disadvantages. Currently we just provided simple hashing based strategy for data sharding.

Further, we may need to add *Replication* strategy for data redundancy.

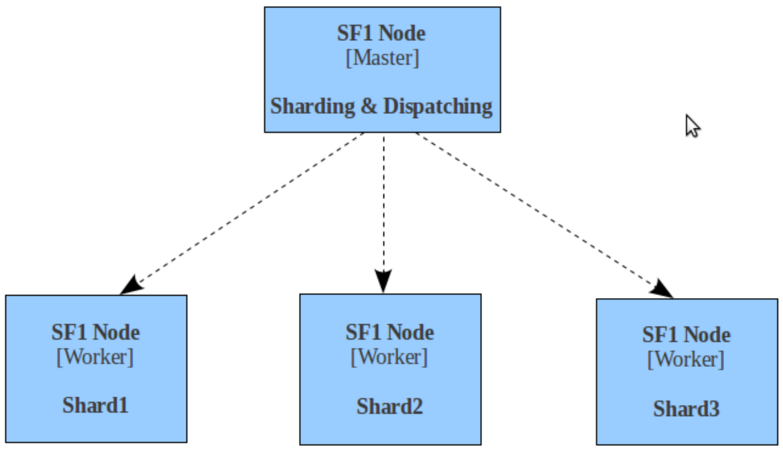

Figure 8: Distributed index

References

[ZooKeeper] http://zookeeper.apache.org/