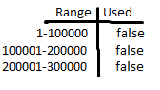
**Distributed Unique ID generator**

**Abstract:**

Creating a unique id across cluster using Zookeeper. There are several techniques to achieve this feature. In this approach, we store the numeric ranges in zookeeper and each cluster node will get unused range from the zookeeper and generate unique id within that range. Zookeeper is highly available and well scalable component that will best fit for this configuration management.

**Design:**

Zookeeper



**API**

getUnusedRange()

generateRange()

R

N3

N1

N2

**Cluster**

* Find the optimal range buffer and limit and build the range table. Write operation in zookeeper is slow operation. Hence choosing the optimal range buffer is significant while considering performance
* Each node in the cluster will request zookeeper and get unused range and locally create the unique sequence id within the available range. When the range exhausted, node will request the next available unused range from the zookeeper and continue to generate unique id.
* This approach ensures each node in the cluster works with the available range so that there won’t be any clash with id across the cluster

**Implementation:**

* Zookeeper distributed map:
* Zookeeper can store data in znodes. Znodes are actually stored in tree-like folder structure
* Each znode path name represents a map key and its content represent map value
* This map is implemented in generic way. It can currently accept all java primitive classes as key and any class with serialization implemented as value.

i.e, Map<K, V> where K – can be java primitive classes , V – any primitive or user defined class with serialization

**API:**

CMap<K, V> *rangemap* = **new** CMap<K, V>(Class K.**class**, Class V.**class**, ZooKeeper *zookeeper\_instance*, String *znode*, CreateMode createMode, **boolean toSort**)

* Application logic:

range = getUnusedRange()

assert range != null : generateRange(start, buffer, limit)

( **!**isExhausted ) ? getUID(range) : getUnusedRange()