Evaluation of Apache Kafka & Redis as a potential back-end for Ascoltatori & Ponte to enable highly resilient & fault tolerant pub-sub mechanism

Nithin Vommi(87203572), Ashish Pedaballi(73222322), Sripad K S(73524050)

University of California, Irvine

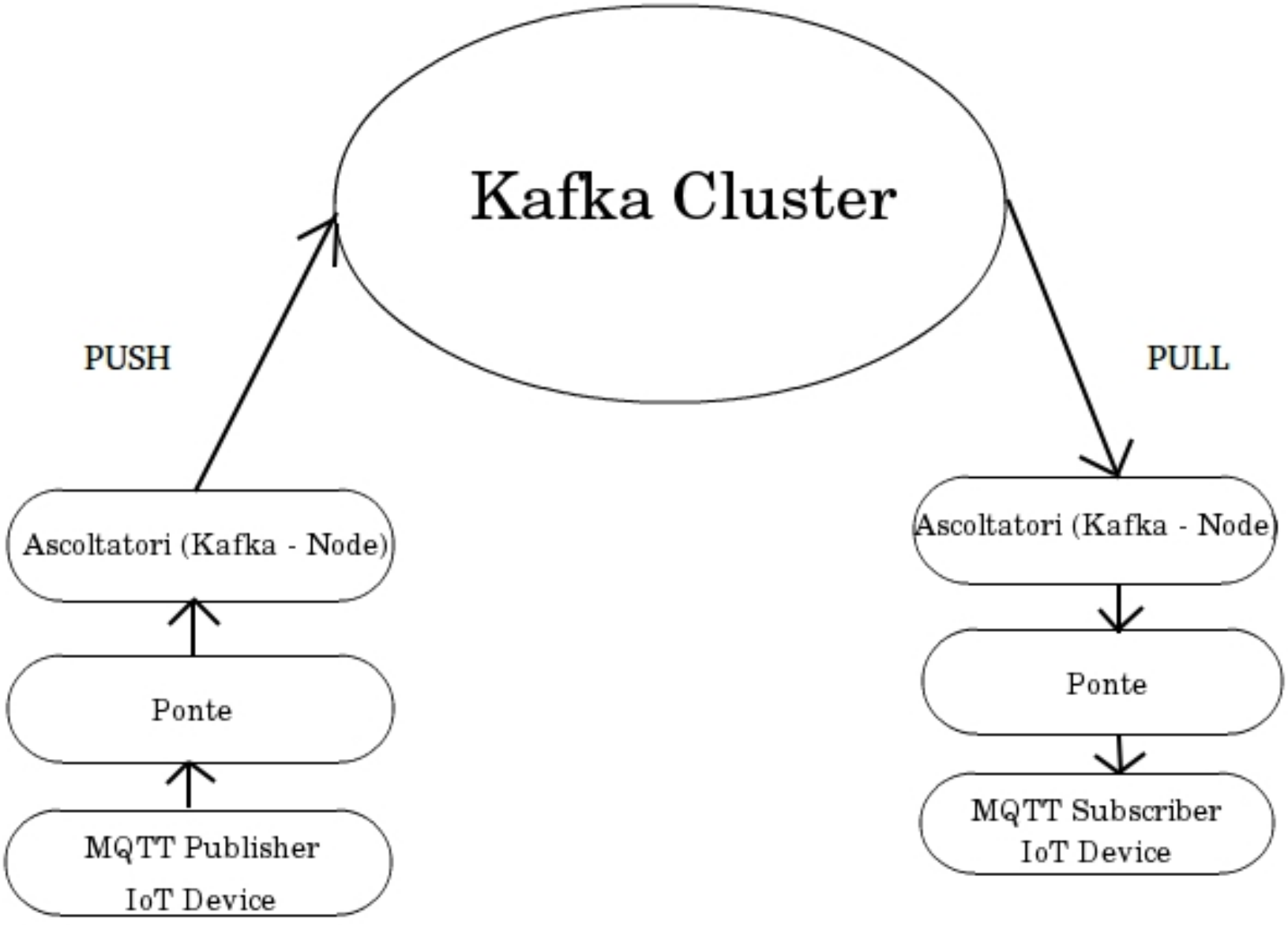
June 9, 2016

Objectives

Evaluate the potential of Apache Kafka and Redis in terms of the ability to provide a distributed broker network for the IoT devices along with the multi-protocol benefits of Ponte.
These two systems were tested against the following metrics -

- Average End to End Latency
- Communication Losses
- Ability to equip the system to favor storage of messages at the production site
- Auto creation of topics when a publisher starts publishing to a topic that is non existing
- Fault tolerance in-case of broker failure
- Ability to support at-least 500 open clients (publishers and subscribers) simultaneously
- Ease of integration with Ascoltatori and Ponte

Related Work - Kafka



Related Work - Redis

- Deployed Redis in Master-Slave configuration with the two instances located geographically distant from each other
- Configured Redis to work as a cluster with 3 Master Servers
- Pub - Sub across all the servers
- Tested Ponte over Redis

Testing & Evaluation Plan

Avg End - End Latency Comparison for Pub-Sub in MQTT

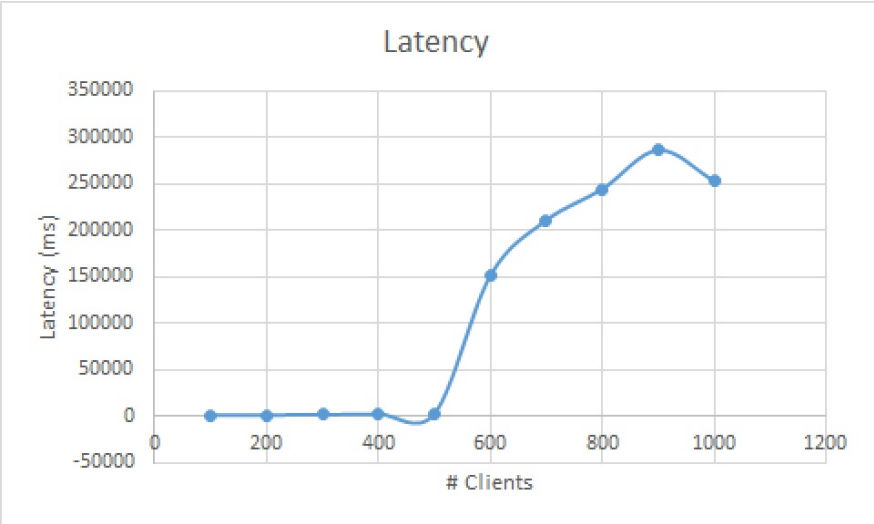
Distributed Network

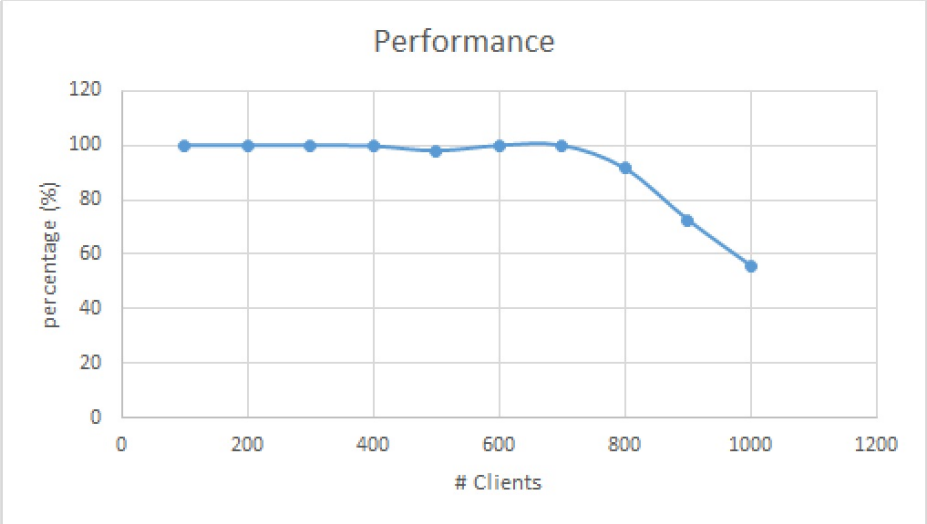
Metric - " Maximum Number of Concurrent Clients "

Experimental Setup -
Experiments were conducted by firing a MQTT Publisher for each topic and a MQTT subscriber subscribing to that topic.

Publisher Location - North California
Logs Location - North California
Subscriber Location - Sydney
Publisher : Subscriber = 1
messages per topic - 100

Distributed Broker Network Performance - Kafka





Publisher Location - North California
Logs Location - North California
Subscriber Location - Sydney
Publisher : Subscriber = 1
messages per topic - 100

Conclusion

- The performance of Kafka and Redis in a non - distributed network are comparable.
- Ponte is compatible with Kafka in single and distributed broker networks but with Redis, it only supports a single broker network
- Storage of messages at the production site can be achieved by passing JSON objects that store the reassignment information.
- Kafka can handle upto (n-1) failures where 'n' is the replication factor
- Kafka Distributed Network is best suitable for 500 clients because the performace degrades as we increase the number of clients beyond 500.

The End