CS554 Project Ideas

CloudKon:DQS – Implementing a Scalable Distributed Queue Service

Overview

Distributed Task Queues can be used in many systems and play different roles such as content delivery, notification system and message delivery tools. It is important for the queue services to be able to deliver messages in larger scales and provide parallel access at the same time. An example of a commercial distributed task queue is Amazon SQS. Amazon SQS is a distributed message delivery fabric that is highly scalable. It can queue unlimited number of short messages (maximum size: 256 KB) and deliver them to multiple users in parallel. In order to be able to provide such high throughput at large scales, SQS confines some of the features that are provided by traditional queues. SQS does not guarantee the order of the messages. It guarantees the delivery of each message. But it does not guarantee the exactly once delivery. That means there could be multiple copies of the same message delivered to the users. The goal of this project is to implement a Distributed Message Queue similar to the Amazon SQS. The user has to be able to upload messages in string format up to 256KB sizes. The queue has to provide simultaneous upload and download from multiple users at the same time. It is also important to provide SQS like APIs in order to provide compatibility with current systems. There also other features that needs to be implemented. An important feature is queue monitoring. The user has to be able to check the queue size and different time stamps for each message (e.g. upload time, download time). The DQS should be compared to SQS and Rabbit MQ and Celery. This is a project for 2 students. If a 3rd student is on the team, there would be the expectation that DQS would be integrated into CloudKon, and the CloudKon+DQS would be compared to CloudKon+SQS.

Methodology

One way to implement a distributed queue is to use a no-SQL Key Value Store as the message container. The Key-Value store can save to message and deliver it in a distributed fashion. ZHT is a good example for an open source distributed key value store that can be used to implement such system. Other methodologies to implement the system can be used.

Relevant Systems and Reading Material

Amazon SQS:

- http://aws.amazon.com/sqs/
- http://sqs-public-images.s3.amazonaws.com/Building_Scalabale_EC2_applications_with_SQS2.pdf
- http://awsdocs.s3.amazonaws.com/SQS/latest/sqs-gsg.pdf

ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table; **http://datasys.cs.iit.edu/publications/2013_IPDPS13_ZHT.pdf**

Rabbit MQ: http://www.rabbitmq.com/documentation.html

Apache ActiveMQ:

http://activemq.apache.org/articles.html

http://cometdaily.com/2008/10/08/scalable-real-time-web-architecture-part-1-stomp-comet-and-message-queues/

http://www.christianposta.com/blog/?p=273

Comparison of RabbitMQ with ActiveMQ:

http://bhavin.directi.com/rabbitmq-vs-apache-activemq-vs-apache-qpid/

KAFKA:

http://kafka.apache.org/

http://kafka.apache.org/documentation.html#introduction

Hedwig:

http://wiki.apache.org/hadoop/HedWig

http://zookeeper.apache.org/bookkeeper/docs/r4.0.0/hedwigUser.html

Couch RQS: https://code.google.com/p/couch-rqs/wiki/HowRQSWorks

Preferred/Required Skills

Programming language choice: any low level high performance programming language (e.g. C/C++, Python).
Skills/knowledge: REST API, Amazon EC2, Amazon SQS, Linux Bash Scripting, distributed Hash Tables.

Performance Metrics

Throughput, Latency

Project Mentor

Iman Sadooghi: isadoogh@iit.edu, http://datasys.cs.iit.edu/~isadooghi/