Characterizing and Improving the General Performance of Apache Zookeeper

Chandan Bagai / Kenneth B. Kent
University of New Brunswick
Faculty of Computer Science
cbagai@unb.ca ken@unb.ca

Outline

- **Introduction to Hadoop and sub project Zookeeper built on Top of it.**
- **Visualization of logical components of Apache Zookeeper.**
- **Focusing on Performance Metrics and some design aspects which may lead to a performance enhancement.**

Motivation

The Apache Hadoop project is an open source project which supports distributed applications involving Big Data. It is an umbrella for a number of sub-projects like Zookeeper. Zookeeper is a fault tolerant distributed co-ordination service used to provide functionalities required by cloud based applications that are bootstrapping, storing configuration data etc. It uses wait free shared data objects rather than using a conventional approach of using locks in order to guarantee the order of operations being applied on the above objects.

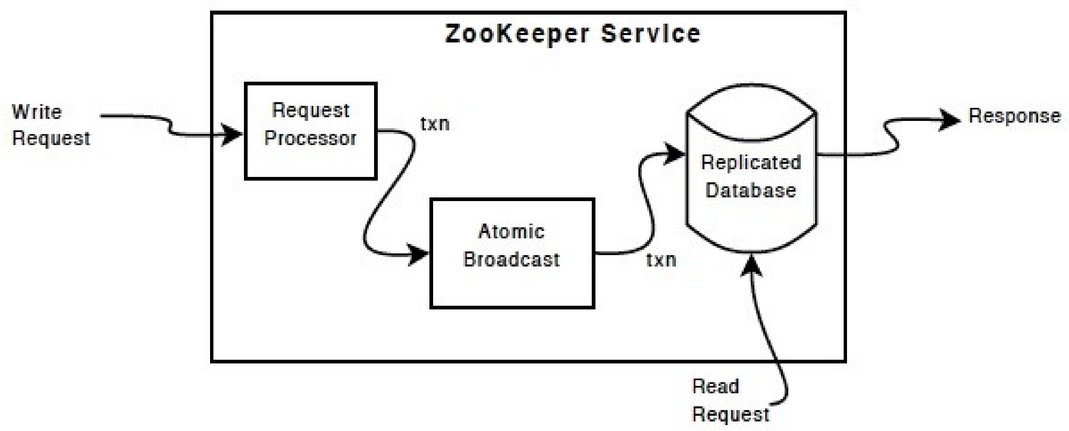
In this contribution we provide the insights of this coordination service. Working flow of this service and metrics used for its performance. Some tests and optimizations that may be helpful in boosting the performance of Apache Zookeeper.

Background

Below is the figure showing the logical makeup of the Zookeeper service. Read requests are serviced from the local database. On the contrary write requests are transformed into idempotent transactions. These transactions are communicated through the protocol used by Zookeeper i.e. Zookeeper Atomic Broadcast Protocol (ZAB) . ZAB satisfies all the requirements of Zookeeper i.e. Reliable Delivery, Total Order, Casual Order etc.

The processing of the transaction where ZAB comes into scenario are as follows :-

- **Leader Election among quorum (majority) of servers.**
- **After leader is elected it synchronizes with its followers.**
- **Next to synchronization is broadcasting of messages and recovery (if required) .**



Prior to ZAB is the request processing pipeline which is the core area in regard to performance. The request is passed through a chain of request processors .

Methodology

- **As above discussed that request pipeline is a major part of Zookeeper. When the write transaction enters the request pipeline it is first put into PrepRequestProcessor and the request is converted to a transaction and for read requests only session verification takes place. In case of multi-op the request is serialized here itself but, it should be serialized at the next processor.**
- **In the next processor to optimize read throughput if there are no pending writes it is directly passed to the Final processor. But, for write requests the whole queue is blocked. We have seen that the take() method of the blocked queue takes most of time and is closely related to performance. We can use ArrayBlockingQueue in place of that and measure performance. It will give a much nicer garbage collection profile and is more cache friendly than the LinkedBlockingQueue.**
- **As of now, various load tests which test the performance of Apache Zookeeper have been fired onto it to bring some insights about Zookeeper. We came to know that both the metrics i.e. Throughput and Latency vary depending upon the type of Znodes.**
- **We found that Persistent Znodes have higher throughput and less latency as compared to Persistent Sequential nodes because these involve extra processing done at one of the request processors we talked above. Moreover especially in Ephemeral node where the server has to clean them up after the session expired so it has to maintain some kind of data structure for these nodes.**
- **Also we tried to obtain a lower bound on the speed of on-disk Zookeeper and compared the result with the conventional File Write Disk Mechanism thus collecting results depicting that there is not much of difference for writing a 10000000 bytes the difference was merely 0.042 seconds.**
- **The goal of this project is to do the same tests and more after carrying out some optimizations and measure performance.**