Implementing Load Balancer for ZooKeeper

Tian Guan, Yunlong Song, Zizhao Han

Index

- Motivation & Goals
- Related Work
- Prototype Design Specifics
- Testing & Evaluation Plan

Motivation & Goals

- In large distributed system, servers need to process large amount of requests.(For example in our project: ticket booking system, the amount of requests will increase rapidly during certain times.)
- In order to process requests more efficiently and handle node failure, we need to use some mechanism to ensure load-balancing. However Zookeeper doesn’t have a load-balancing algorithm.
- We propose a dynamic load-balancing algorithm based on Zookeeper to maintain good load-balancing to achieve better performance.

Related work

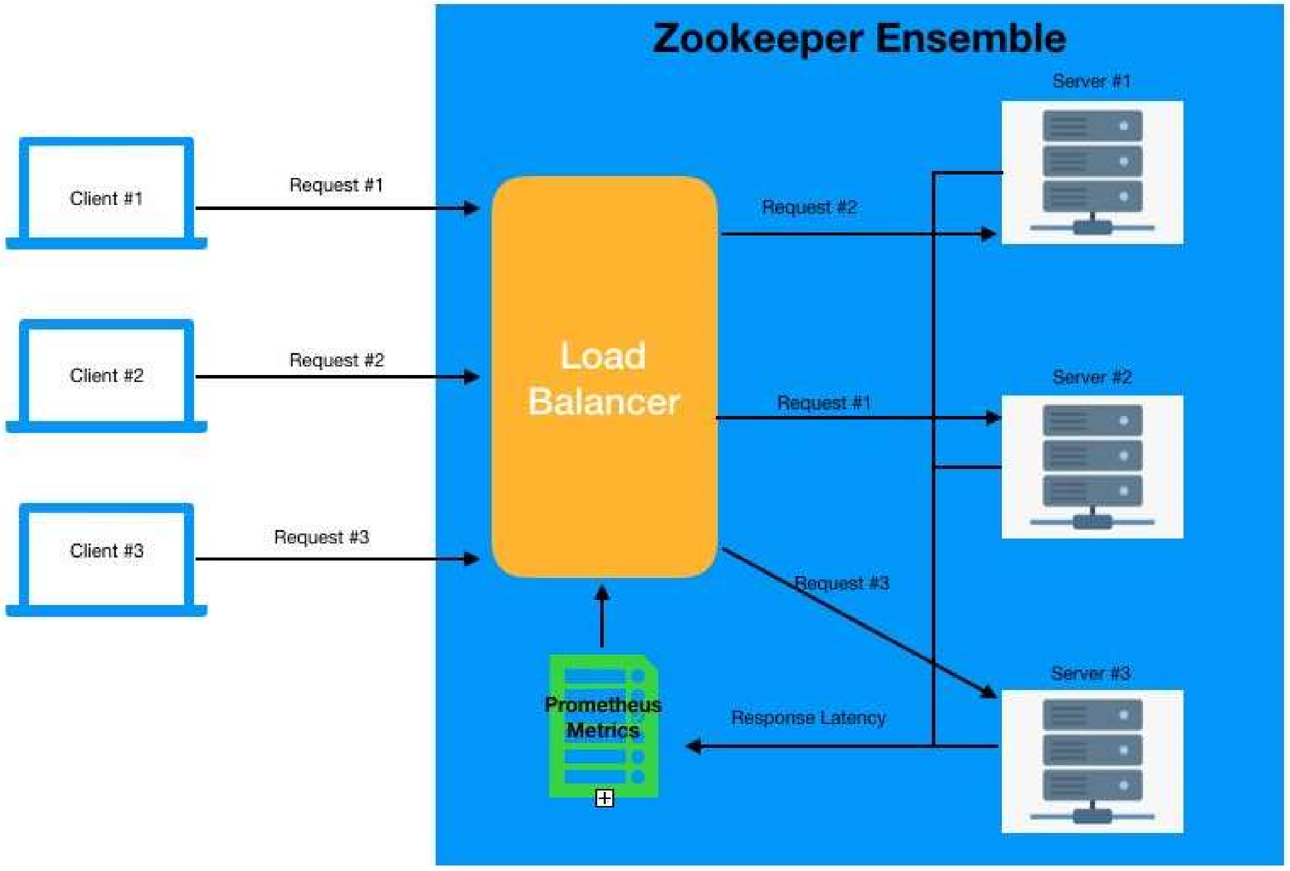
- S. Aslam and M. A. Shah, ”Load balancing algorithms in cloud computing: A survey of modern techniques,” 2015 National Software Engineering Conference (NSEC), Rawalpindi, 2015, pp. 30-35, doi: 10.1109/NSEC.2015.7396341.
- Yakhchi, M., Ghafari, S.M., Yakhchi, S., Fazeliy, M., Patooghi, A., 2015. Proposing a Load Balancing Method Based on Cuckoo Optimization Algorithm for Energy Management in Cloud Computing Infrastructures. Published In: Proceedings of the 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO).
- Westbrook, Jeffery. “Load Balancing for Response Time.” Journal of Algorithms 35.1 (2000): 1–16. Web.

Prototype Design Specifics

Architecture

Algorithm

Architecture

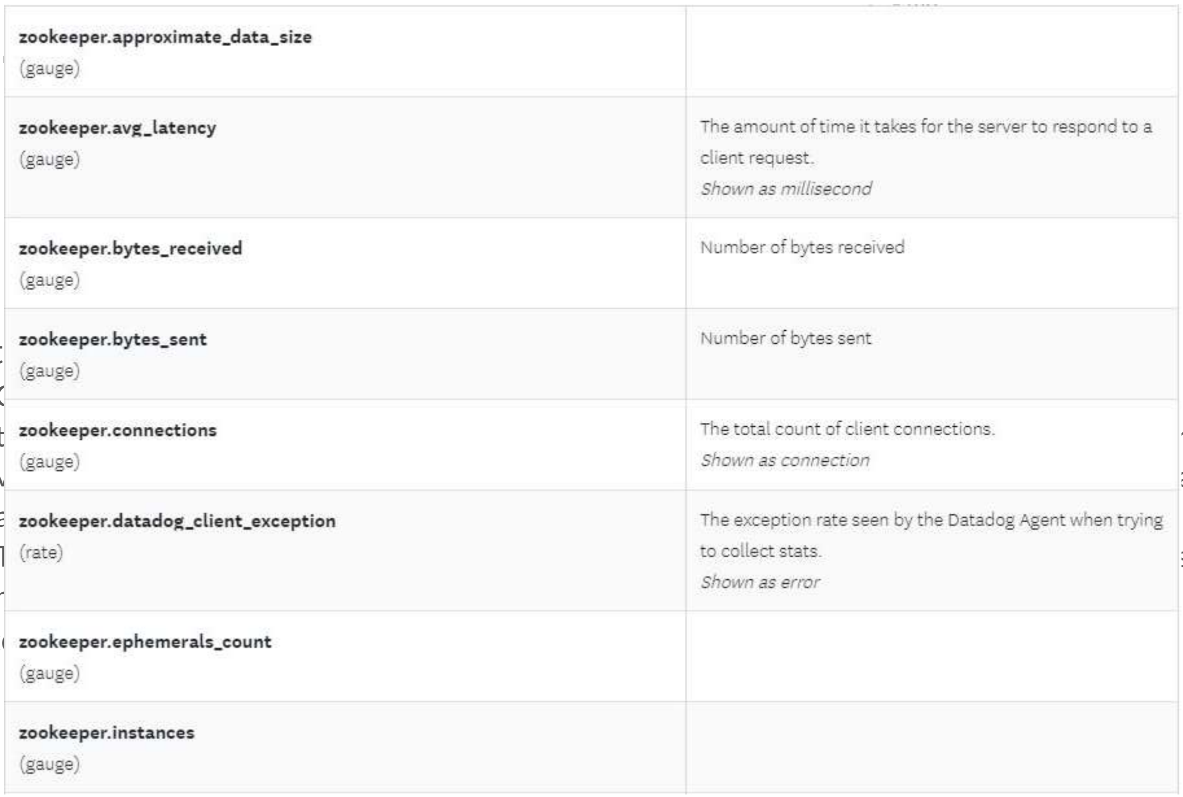


Algorithm design

- Detect overutilized hosts
    - Generally, an over-utilized server cannot service to all the requests which be allocated to it. In this situation, the **response times of the requests** will be increased. It would also cause SLA (service level agreement) violation, as the overutilized server could not satisfy the needs of the requests that been allocated to it.   In any cloud service provider, it is essential to deal with the over-utilized servers.
    - The ZooKeeper metrics will provide the request response latency, and we will use prometheus to capture the metrics and use this information to balance the load of servers.
- Balance load

Algori

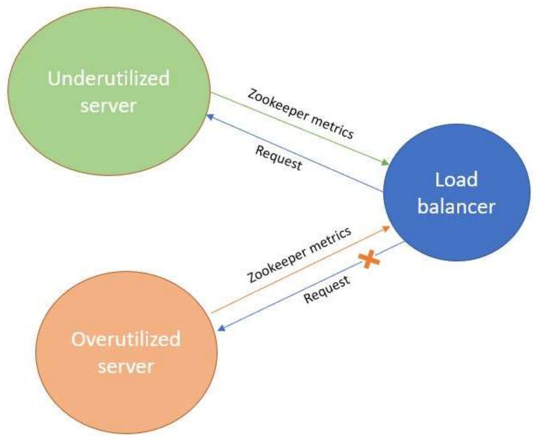
- Detect
  - C this situation,
    t reement)
    v ated to it. In
    a
  - T s to capture the
    r
- Balanc



zookeeper.approximate_data_size (gauge)	
zookeeper.avg_latency (gauge)	The amount of time it takes for the server to respond to a client request. *Shown as millisecond*
zookeeper.bytes_received (gauge)	Number of bytes received
zookeeper.bytes_sent (gauge)	Number of bytes sent
zookeeper.connections (gauge)	The total count of client connections. *Shown as connection*
zookeeper.datadog_client_exception (rate)	The exception rate seen by the Datadog Agent when trying to collect stats. *Shown as error*
zookeeper.ephemerals_count (gauge)	
zookeeper.instances (gauge)	

Part of ZooKeeper metrics

Algorithm design



- Detect overutilized servers
- Balance load
    - After we are aware of the overutilized servers, we will not send requests to these servers until the response latency of them drops below the boundary.

Testing & Evaluation Plan

1. Deploy the system on computers (emulating multiple servers in computers)
    a. Set-up:
        i. 1000 requests
        ii. 4 servers
2. Send sets of requests from multiple clients to the servers (e.g. using Postman)
3. Check if the metrics of each server are collected
4. Compare the results of using our algorithm with using other ones
5. Check if overloaded servers are be detected
6. Check if client requests are delivered to least-loaded servers