Parallelize Operations on a Durable Queue by Using Log Structured Storage with Memory Interleaving

Avinash D'Silva
15028534
MSc in Cloud Computing

26th July 2016

Abstract

In the recent years, Queues have become the de-facto standard for various purposes such as asynchronous processing, buffering and load balancing. Queues basically have two operations, enqueue and dequeue. These operations are parallel in theory but in practice when implemented as a durable queue, they tend to become more akin to an ACID compliant database system and hence loose their ability to perform operations in parallel. Durability of a queue cannot be compromised and yet at the same time, the need for performance is ever more increasing as queues have become the backbone of most distributed systems. This study will focus on Log Structured Storage along with Memory Interleaving and how a combination of these can make it possible for a queue to be durable as well as be able to perform parallel operations at the same time.

Contents

1 Introduction

Queues are not a modern invention related to the field of Computer Science. They have existed well over the centuries. People queue up instinctively when there is a crowd for any given situation (Spieser and Davison; 2008). Queues particularly in the field of Computer science are known for their FIFO(First in First Out) data flow (MacLaren; 1969). Queues were initially used for simple well known tasks such as enqueueing a print task on a printer, for breadth first search and have applications in graph theory.

The structure of a queue very simple to understand and implement and queues in the recent years have become the backbone of distributed systems (Lamport; 1978). They

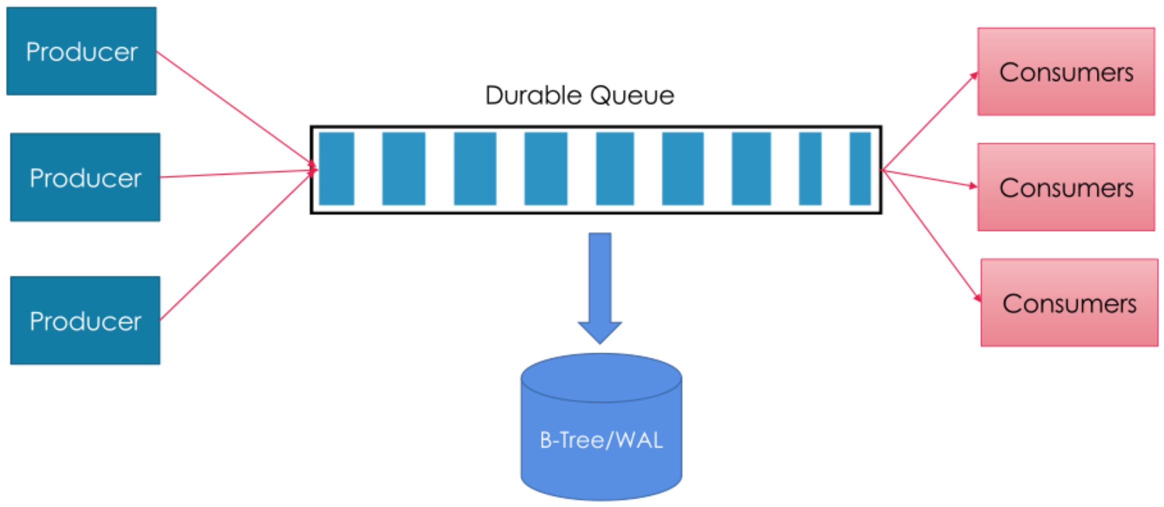


Figure 1: Overall architecture of the current queueing systems

are used for asynchronous processing, message passing, buffering and load balancing (Lu et al.; 2011).

The size of data that most public facing applications have to deal with has been growing. There is ever more data from different IoT devices, Cell phones and Communication satellites and hence this age is called the Age of Big Data (Metz; 2015) (Lohr; 2012).

The distributed systems have come under heavy load due to this growth in the size of data. One of the important part of the distributed system, the queues are the most affected as these are the ones that carry and distribute the data throughout the system (Lamport; 1978). Queueing servers are becoming a bottleneck due to their fundamental shortcoming of not supporting parallel operations. The Figure 1 shows how queues are implemented currently and shows how producers and consumers both use a single Write Ahead Log for durability. The Write Ahead Log only allows one operation at a time to be logged and hence queues aren't parallel due to this persistance layer.

Although queues are parallel in theory, they loose their parallel nature due to reliance on external disk based storage structures such as B-Trees and WAL(Write Ahead Log) for durability. Queueing servers use various workarounds such as fan out queues and clustering as shown in Figure 2 (Albrecht et al.; 2013).

The method of fanning out or clustering involves using multiple queues instead of a single queue to better handle the load. These multiple queues are abstracted as a single queue. One queueing server acts as a master while the rest act as slaves. These queues can be either mirror replicated or sharded (Albrecht et al.; 2013). This type of setup is widely used in the industry today and almost all major queuing servers support the feature of clustering (Kermarrec and Triantafillou; 2013). But this doesn't solve the fundamental problem of durable queues not being parallel. This proposal describes the ways and means to parallelize operations on a durable queue using a combination of well known technique of memory interleaving and Log Structured Storage.

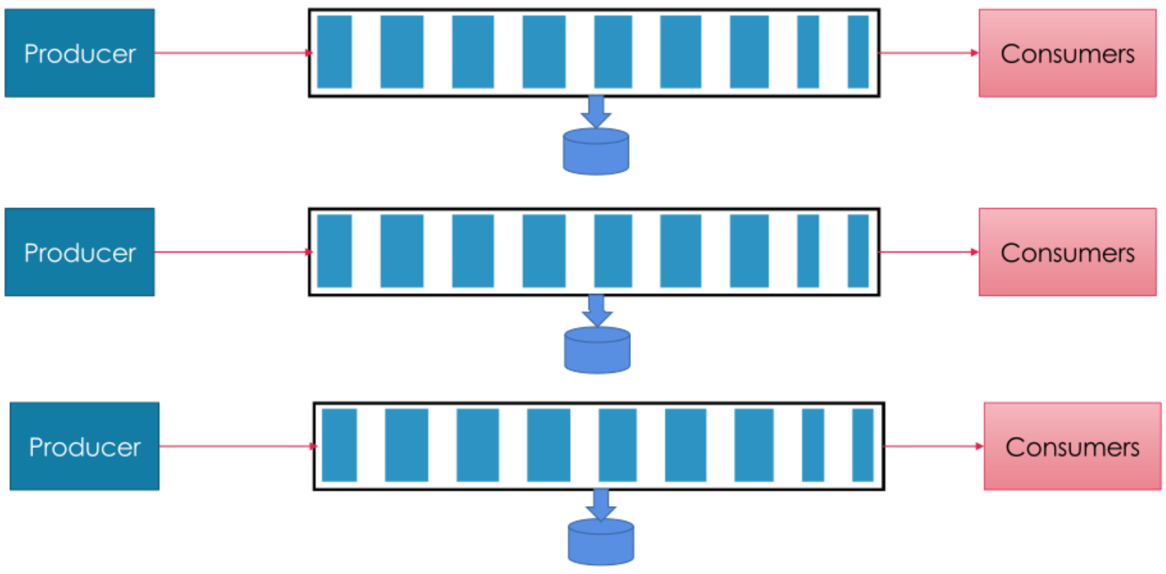


Figure 2: Achitecture of a clustered queueing systems

2 Literature Review

Various file formats have been defined, one of the oldest and the most widely used is the distributed systems have come under heavy load due to this growth in the size of data. One of the important part of the distributed system, the queues are the most affected as these are the ones that carry and distribute the data throughout the system (Lamport; 1978). Queueing servers are becoming a bottleneck due to their fundamental shortcoming of not supporting parallel operations. Queueing servers use various workarounds such as fan out queues and clustering (Albrecht et al.; 2013).

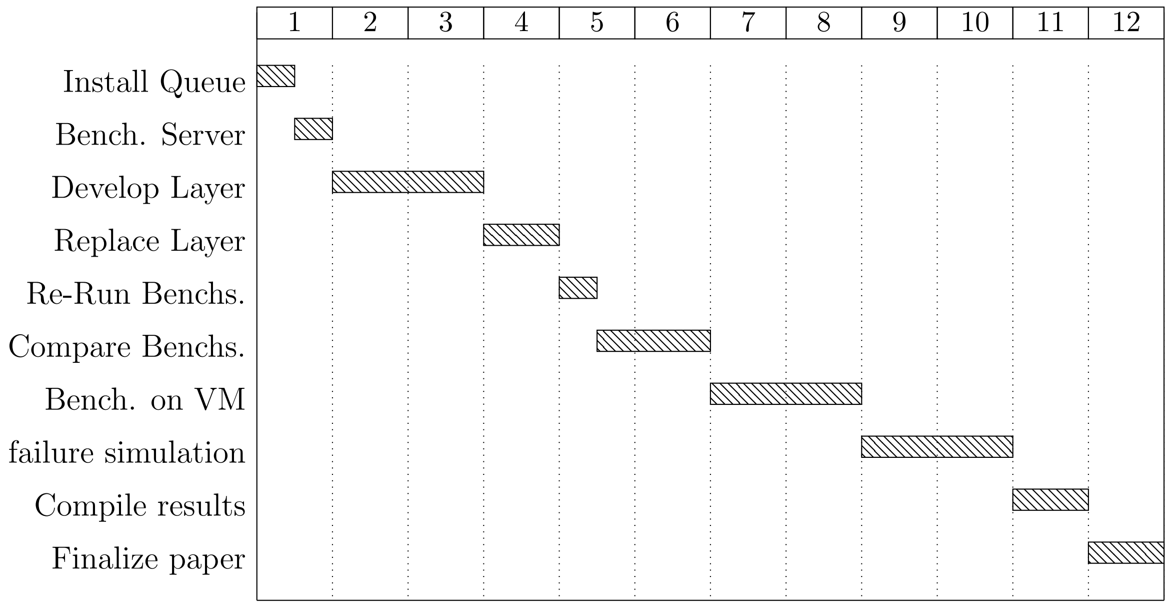
The method of fanning out or clustering involves using multiple queues instead of a single queue to better handle the load. These multiple queues are abstracted as a single queue. One queueing server acts as a master while the rest act as slaves. These queues can be either mirror replicated or sharded (Albrecht et al.; 2013).

3 Research Method and Specification

The distributed systems have come under heavy load due to this growth in the size of data. One of the important part of the distributed system, the queues are the most affected as these are the ones that carry and distribute the data throughout the system (Lamport; 1978). Queueing servers are becoming a bottleneck due to their fundamental shortcoming of not supporting parallel operations. Queueing servers use various workarounds such as fan out queues and clustering (Albrecht et al.; 2013).

3.1 Gantt Chart

The following gantt chart shows the number of weeks on the X-Axis and the tasks that are to be the executed on the Y-Axis.



The first phase involves installing and configuring an open source queueing server. After the installation, several benchmarks are run to see how the server performs in different conditions. The conditions will include different processor speeds, different number of processor cores and varying size of memory. Then the proposed solution is developed and this solution will replace the previously benchmarked queueing servers' storage layer. After the replacement, The benchmark tests are performed again and then compared. Then both the versions are compared based on how they perform on a virtualized environment. To test the effectiveness of memory interleaving, hardware failure is simualated. These tests along with the benchmarks results are compiled and the paper is finalized based on these results.

References

Albrecht, M., Rajan, D. and Thain, D. (2013). Making work queue cluster-friendly for data intensive scientific applications, *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, IEEE, pp. 1–8.

Kermarrec, A.-M. and Triantafillou, P. (2013). Xl peer-to-peer pub/sub systems, *ACM Computing Surveys (CSUR)* **46**(2): 16.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system, *Communications of the ACM* **21**(7): 558–565.

Lohr, S. (2012). The age of big data, *New York Times* **11**.

Lu, Y., Xie, Q., Kliot, G., Geller, A., Larus, J. R. and Greenberg, A. (2011). Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services, *Performance Evaluation* **68**(11): 1056–1071.

MacLaren, M. D. (1969). The art of computer programming-volume 1: Fundamental algorithms (donald e. knuth), *SIAM Review* **11**(1): 89–91.

Metz, S. (2015). Big data., *Science Teacher* **82**(5): 6.

Spieser, K. and Davison, D. (2008). Stabilizing the psychological dynamics of people in a queue, *2008 American Control Conference*, IEEE, pp. 4173–4178.