**Book store Part2**

Distributed systems is best approach for big application which has a heavy load and use, book store application were updated to support distribution concepts such as: replication, consistency and caching to enhance the performance and reduced response time. These approaches were applied to application by replicate the database and servers, synchronizing the updates and caching the request data

To dive deep in application, the front server keep handling the requests but in different way, a load balancing algorithm were applied to server so it can distribute the request on multi servers, the used algorithm is Round Robin algorithm, by using this algorithm the requests passed to servers by a simple operation which is the number of request module the number of servers.

In addition the front server were redesigned to use in-memory cache, this step is useful to enhance performance and decrease response time, when request is arrived the front server check the cache for previous saved record, this step saves the time required to fetch data from database, if the record is not available it fetched from database then save it to cache to be used in future requests.

In backend side the structure has changed to support replication and consistency, the replication were applied to database and servers by creating a new instance of database and another on of servers( catalog and order servers). To support consistency between servers, databases and cached data the update operation were handled in the front server by checking a returned flag from the backend servers, when the data is updated in one database the changes reflects to the other database, and removed from cache

The new book store design is good and provide a good enhancement for the performance, but there would be a problem with the load balancer algorithm which is not adaptive, the problem appears when a lot of request came to front server so if one of the backend servers is already processing a large request the load balancer algorithm will still forward the next request to it, in the same time the other server may be idle and completed the request passed to it, so this would take more time and reduced the performance. Another problem is that bottleneck may happen to the front server when there is a huge number of request needed to be distributed, this problem can be solved by increasing the number of backend servers to reduce the load on the other ones

But on the other hand there is the in-memory cache which may compensate the time consumed for distributed requests if the requested data was cached.

In overall the performance for the application is better than the first version, because it reduced the time needed for fetching data, but it may face some problems with update and a bottleneck may happen.