BlueBerry Muffin: A Giant Scale File Storage

Eman Okyere and Abhiroop Reddy Nagireddygari

I. ABSTRACT

Blueberry Muffin is a Giant Scale Service for storing and accessing data from across the world. The system is designed to give users reliable and quick access to files online, regardless of where they are accessing from. Blueberry Muffin's primary goals are to reduce latency, increase data availability, and be easily scalable. The key to achieving all these goals are the clusters of geographically nearby machines called "Regions," which allow Blueberry Muffin to perform Region Based Smart Selection, Query Forwarding, and Optimistic Replication.

II. INTRODUCTION

With internet access increasing across the globe in recent years, there is a clear need for a large scale system capable of allowing users from across the globe to upload files without any notable differences in usage due to the location of the user. This paper describes our solution to the need: the worldwide file storage system Blueberry Muffin. Blueberry Muffin achieves its aforementioned goals by utilizing groups of nearby machines, the Regions, to act as homogeneous nodes that can service requests for nearby users.

A. What are Regions?

Understanding a Region is vital to understand Blueberry Muffin's design. A Region is a cluster of machines that work together to act as a node inside the greater system and are critical to the design of the system as a whole. Each region consists of 1 machine serving as a Front End server, with an arbitrary amount of extra machines serving as Database servers. Each region can service any add, lookup, or delete request from a client. Any request that would alter the database of Blueberry Muffin is optimistically replicated across the globe. Utilizing this homogeneous cluster design allows Blueberry Muffin to make the more expensive processes transparent to the user, providing an identical experience to anyone who contacts any Region.

III. SYSTEM REQUIREMENTS

Blueberry Muffin measures its performance using 3 metrics: availability, performance, and scalability. Each of these metrics, along with their solutions, is broken down further here:

A. Availability

Data availability is the most important of the metrics that Blueberry Muffin is measured on. In concept, there should never be a situation where a user is unable to retrieve a file from the file. Inside each Region, a file is replicated multiple times to add a first layer of fault-tolerance; for a Region to be unable to provide access to data, multiple Database servers must be down. In the case of a more severe failure where the Region-internal replications are all unavailable, Blueberry Muffin can forward the request to another Region to complete the request. In the event where the forwarded request fails as well, Blueberry Muffin is facing a catastrophic issue and requires a system checkup.

B. Performance

Blueberry Muffin is designed to service users from across the globe, not a specific geographic region in particular. Thus, we wanted to match a user with the Region that has the lowest latency respective to them. Using Region Smart Selection (RSS), a user that connects to Blueberry Muffin will perform a latency test to identify the two best Regions for that user. The best Region will be used to perform all requests, with the second serving as a backup in the event of a failure. RSS is performed every few minutes during a Client's session to ensure the best latency possible. RSS also serves as Blueberry Muffin's load balancer; if a Region is overwhelmed with requests, RSS will refrain from choosing that Region.

C. Scalability

Scalability in Blueberry Muffin is represented in two ways: adding Database servers to already existing Regions and adding new Regions to the system as a whole. For the former, the addition of new Database servers improves data availability as a faulty Database server would not affect as many replications of data. For the latter, the addition of Regions would serve to reduce latency for users. If a particular area is suffering due to large geographic distance to the nearest Region, or if a single Region is facing a large amount of load, the addition would help alleviate both issues.

As seen in the three different metrics, Blueberry Muffin relies extensively on the homogeneous nature of the Region groups to effectively maintain its performance across regions. As the system's user counts and locations fluctuate, updating the makeup of each Region is critical to optimize Blueberry Muffin's performance.

D. Related Work

Blueberry Muffin aims to improve on the main goals of a Giant Scale Service. Namely, our design is aimed to be more fault tolerant and have a higher degree of data availability then other similar services. Our primary differentiator is our Regions, which use their unique design to be more resilient and flexible than typical Front End layers or Load Balancers.

E. Organization of the Paper

This paper will be broken into 4 parts. First, we will be describing further design and architecture of Blueberry Muffin, then the implementation details of the system. We then will share the performance evaluations of our system and conclude with our plans for future improvements.

IV. Design and Architecture

Blueberry Muffin is a 3-tier application with a Client, Front End, and Database layer. Figure One shows the general design of Blueberry Muffin with multiple Regions, each with varying amounts of Database servers. As displayed in the diagram, Client machines only connect with a singular Front End Server. However, each Front End is capable of communicating with all other Front End machines within the system. In addition, Database servers can interact with other Database machines inside their own Region, but not machines outside the Region. Each of the 3 layers will be discussed further in depth within this section.

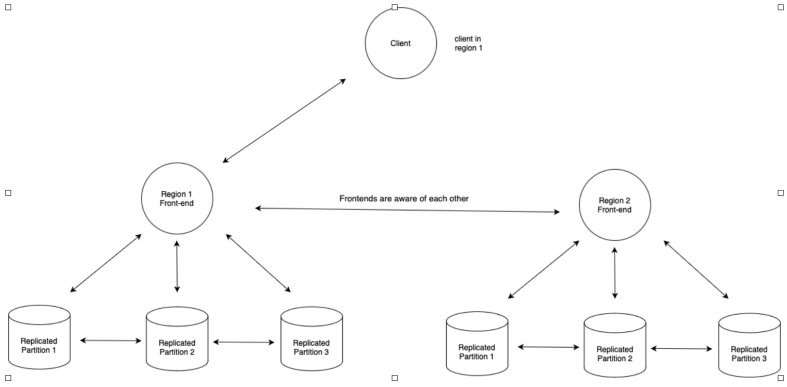

Fig. 1. A diagram of Blueberry Muffin showing how each machine interacts within the system. Clients communicate with Front Ends. Front Ends communicate with Clients, other Front Ends, and Databases inside their Region. Databases communicate with their Front End and other Databases inside the Region.

A. Client Tier

The client tier is the part of Blueberry Muffin that a user will be interacting with. Namely, this is the layer where we get the request from the user and return the information back to them. To start the application, a Client does require the IP Address of a currently online Front End machine connected to Blueberry Muffin, but this parameter will be omitted once Blueberry Muffin is fully online and the IP Address can be sourced from a DNS lookup.

1) Key Data and Data Structures: The Client servers depend on choosing the optimal Region's Front End Server's IP address to ensure the lowest latency for each user. For fault tolerance and graceful degradation, RSS returns the two best Region's Front End server's IP addresses. If a singular category of data is unavailable for the optimal region, such as in a Database server failure, the system does not discard the IP for that region as it could still be used to successfully query other categories. However, requests for a faulty data will be sent to the second optimal region, which will cause a higher latency but prevent the user from being unable to get their data. In a Front End Server failure, the Client will recognize that the Region is down and

now fully discard the IP address and work off the backup it has stored. RSS is run at initial startup and then will run every 5 minutes a Client is connected to the System. This allows RSS to act as a psuedo-load balancer as changes in latency from a Region will be detected and accounted for. In addition, the Client also stores its initial entry point IP so that if both optimal Front End IPs are down, RSS can rerun using the entry point.

2) Region Smart Selection: RSS works by completing an iterative ping check to each Front End machine within Blueberry Muffin. After getting a list of the IPs to each Front End machine from the entry point, RSS keeps track of both the best times and their respective IP addresses during the iteration. At the end, RSS stores the addresses it found for optimal access to Blueberry Muffin.

B. Front End Tier

The Front End Servers are the most complex and hardworking parts of the system. Besides acting as a layer to convert user queries into Database requests and to return Database confirmations, the Front End servers also manage adding new Regions into Blueberry Muffin, partitioning and re-partitioning of data, and performing Optimistic Replication.

1) Key Data and Data Structures: For the complexity of the Front End machines, there are a large number of key components needed. The most critical of them is a list of categories. This list is both necessary to provide access to the correct Database and for the re-partitioning process. Another important list contains the IP addresses of all other Front End machines currently in Blueberry Muffin. By tracking these IP addresses, a Front End machine enables the Client's RSS process and Optimistic Replication. Lastly, there is a list of the IP addresses currently within the Region, which are used for creating calls to access those machines.

2) New Region Addition: A benefit of a Region's homogeneous nature is that any Region's Front End machine can be used as an entry point into the system, even for new Regions. A new region is given the IP address of a currently online Front End machine, similar to a Client when starting up, and gives the entry point its own IP address. The entry point then adds the new Regions IP address to its internal list of Front Ends, and also broadcasts the addition to all other Front Ends so that the new Region is accessible from any Front End machine in Blueberry Muffin. Figure 2 visualizes this process.

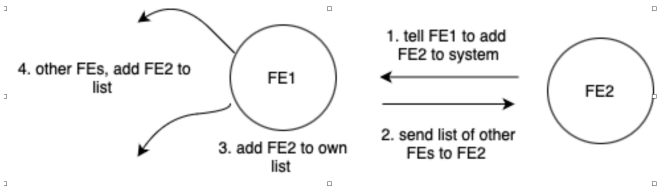

Fig. 2. This image depicts the flow of processes that allow a Region to be added to Blueberry Muffin. The incoming Region's Front End machine is given an entry point. The incoming machine sends over its own IP address and receives a list of all the Front End IP addresses currently in the system, while the entry point adds the incoming IP to its own list. Lastly, the entry point forwards the IP to all other Front Ends as well

3) Hashing and Partitioning: Blueberry Muffin utilizes a system of partitioning the data by categories to optimize database usage and improve data availability. Incoming data is hashed using an internal hash function that uses the data category and maps it to two Database Machines for storage. The benefit of hashing to two machines is a higher degree of data availability; if only one replication goes down, all data is still available. Subsequent requests for a file is also put through the hash function to see what Database machines can serve the query. Front End machines also have the ability to re-partition the data when necessary. This occurs whenever a new Database machine joins the Region or when an existing one is deemed faulty and must be removed. A faulty machine is recognized during an add or remove call that notices that both replications are down. This is not done during a lookup call because there is a chance that the file is available on the client's second optimal front. We also do not re-partition when only one replica is down, since the data is still available in the Region. The reason we do not re-partition in these situations is due to the high cost of the process due to our hash function. In either situation, the internal Database list is modified and the hash function remaps each category to two new Database machines. If there is a disparity between where a category is currently held and where it should be held, the Front End server prompts a transmission of data from the current to new Database machine. Typically, both replications in a Region will be prompted to send their data in the interest of being fault tolerant in case one replication became faulty.

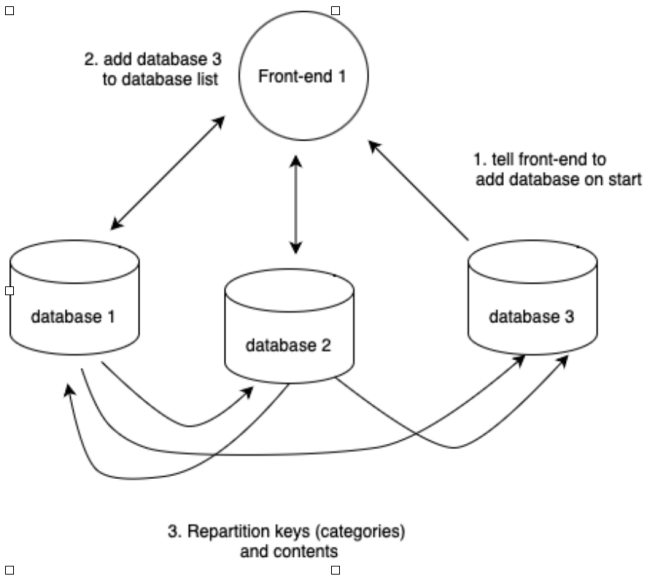

Fig. 3. A diagram of how a Database machine joins a Region. First the incoming machine tells its entry point to add the Database's IP to the internal list of Database IPs. Then, re-partitioning occurs and the new Database is given categories of data.

4) Optimistic Replication: The chosen method of Database synchronization in Blueberry Muffin is Optimistic Replication. Anytime that a user adds a new file or requests to delete an existing one, the Front End machine broadcasts that request to every single Region inside the system. Since we want each user to be able to have the same experience regardless of where they are accessing from, Optimistic Replication is necessary to meet that goal.

C. Database Tier

As the tier's name mentions, the Database machines are used to store the contents of Blueberry Muffin. The current implementation makes uses of the machines file system, creating a new folder for each category and adding text files from users to the category specified.

1) Key Data and Data Structures: Besides the file storage, Database machines store very little data. The only notable object inside the machine is the object lock we use to prevent data corruption during file addition or deletion.

2) Re-partitioning Data: Databases also have the ability to communicate with other Database machines within the same Region to send and receive data during the re-partitoning of the Region. After the Front End machine takes care of assigning each

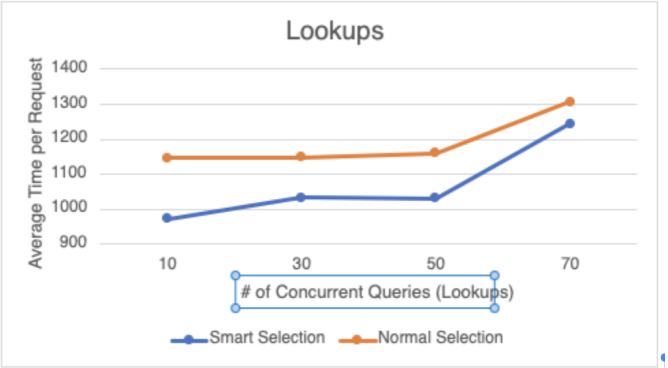

Fig. 4. A diagram plotting the Average Time per Query(ms) based on the number of Concurrent Look-ups occurring for both RSS and Normal Selection.

category to their Databases, the Database machines will transmit any data they are no longer in charge of to their new machine. During the transmission, any outdated or stale data is automatically trimmed, ensuring that each Database contains only data for the categories it has been assigned.

V. Implementation Details

We decided to implement Blueberry Muffin in Java using XML-RPC. Java was chosen as our language of choice simply due to us having more experience working with Distributed System with the language. Furthermore, using Java allowed us to use XML-RPC. The framework simplified the transport protocol for us as implementing fully functional HTTP transportation seemed unnecessary for text file transfers.

VI. Performance Evaluation

We tested Blueberry Muffin for four things: the scalability of the system, the benefit of RSS, the fault tolerance of the System, and the time for re-partitioning. We used Amazon VMs to create a standardized environment, and all queries were performed using a 300KB text file to make file size a non-factor during our tests.

A. Scalability

To test Blueberry Muffin's scalability, we put the system under a stress test. A Region with two Database machines was given concurrent lookup and add calls, plotted in Figure 4 and 5 respectively. For the former, we saw the system maintained a constant Average Time per Request, before starting

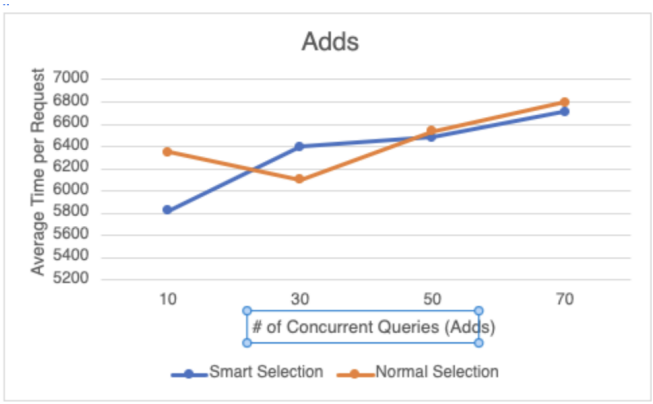

Fig. 5. A diagram plotting the Average Time per Query(ms) based on the number of Concurrent Adds occurring for both RSS and Normal Selection.

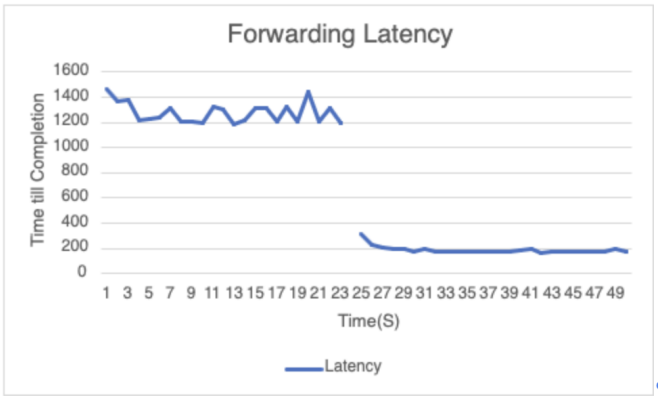

Fig. 6. A diagram plotting the how long a query took to complete, Time till Completion in ms, against time in seconds before and after knocking out a Clients optimal Front End

Fig. 7. A diagram plotting the how long a query took to complete, Time till Completion in ms, against time in seconds in a time-frame where re-partitioning occurred

to increase at 70 concurrent calls. The former saw a linear growth of Average Time per Request as a trend. Overall, these results do align with our expectations and goals for Blueberry Muffin. Lookups, being the cheaper of the two, can be handled to a much higher degree than adds. Though adds did see a linear growth, this bodes well for the System's scalability as there was only a marginal growth. Adding extra Database or Front End machines, as mentioned in Section II, would remedy this growth.

B. RSS

While measuring Scalability, we also measured RSS by creating a Client that had randomly picked a Region to use, denoted as Normal Selection. In Figure 4, we see that RSS provided a constant benefit to our overall Average Time, but did not provide much, if any, benefit for Adds as displayed in Figure 5. This pattern holds true with our prior beliefs as lookups navigate to only one Region, choosing the best should have caused a decrease in our Time till Completion. However, since adds must optimistically replicate to each Region, we did not expect to see any noticeable improvements. We did not test the delete function, but we expect the trend to be similar to adds due to how deletes are also optimistically replicated.

C. Fault Tolerance

Our fault tolerance test, while proving that our system is highly fault tolerant, we noticed some results that did not line up with our expectations. For this test, we ran sequential lookups on a Region with 1 Database machine. The latency for these requests can be seen in the left half of Figure 6. At 24 seconds in, we forcibly shut down the optimal Front End machine for the client, denoted by the dis-junction in the graph. The client then forwards its current and all requests to the backup IP. During this test, we succeeded at every request. However, we did not expect the Time till Completion value to drop after swapping Regions, we expected for the opposite to occur. One reason we thought caused this trend is that the optimal Region incurred load due to our prior requests, causing the latency to go up.

D. Re-partitioning

Similar to the fault tolerance test, the re-partitioning test performed recurrent add calls on a Region and recorded how long it took to complete, the Time till Completion. According to Figure 7, it took around 3000 milliseconds to complete on average. The spike seen towards the 12th second is when both replications of a category went down and re-partitioning occurred. Immediately after, the latency went back to the previous average. The trend shown here further adds to Blueberry Muffin's fault tolerance since we complete the query by gracefully degrading, allowing other Database machines within the region to complete the request.

VII. Future Improvements

While we are happy with the current functionality of Blueberry Muffin, we have also identified multiple improvements that could make the system even better. Some of these improvements include implementing HTTP transfer, consistent hashing, and a ping to database machines.

A. HTTP Transfer

Currently Blueberry Muffin does have a limit on the file size. A user can upload up to 20MB before causing the system to fail. In addition, the system is designed with only text files in mind. Switching to a HTTP transfer system will allow us to increase the maximum file size and allow for the transfer of different types of files.

B. Consistent Hashing

Currently, the Front End's hash function can cause data to be transmitted from an already existing Database machine to another previously existing machine during re-partitioning. In the future, we hope to implement consistent hashing so that previously connected Database machines only send data, and that the sent data is received only by the new machine joining the region. This addition would greatly decrease the time needed to finish the re-partitioning process.

C. Database Pinging

To further decrease the time to complete a request, a Front End machine can ping its Database machines to identify which ones have the best latency. These latency differences could be due to a geographic distance or a imbalance of load. Regardless, choosing the best machine with respect to latency would decrease time to completion while also acting as a load balancer.

VIII. Conclusion

In this paper, we have described the architecture, implementation details, performance evaluation, and the future improvements for the giant scale file storage system Blueberry Muffin. We have shown that Blueberry Muffin meets its system criteria:

1) Availability: All data inside a Blueberry Muffin Region is replicated twice within the region. This creates high data availability even during a system fault. In the event both replications are taken down, Blueberry Muffin can forward the request to another Region thanks to the backup Region IP stored during RSS.

2) Scalability: Blueberry Muffin has the ability to scale to need and user location density by utilizing the ease of adding new Database machines and/or new Regions. By managing the number of machines servicing an area, Blueberry Muffin has no problem being flexible to meet the unique demands of different areas of the world.

3) Performance: A high level of performance is maintained within Blueberry Muffin by RSS. A client will always be serviced by the Region that has the best latency to the client, which takes into account not only geographic distance, but Region load as well. This ensures that a client is getting the best connection possible.

Blueberry Muffin's ability to meet these three goals comes directly from the homogeneous cluster node design of Regions, which allow sections of the greater system to be modular; capable of being appended and removed when necessary. In the future, we hope to implement the desired updates and see Blueberry Muffin continue to provide equal access to data across the world.