Trackr High Availability Research 2023

1. Introduction

As Trackr is intended to support many organizations and projects all submitting various data, high availability is a large area of concern. There are multiple areas of Trackr that will require improvement and modification to effectively handle this problem. This research will highlight these areas of improvement and will also discuss recommended techniques that should be implemented throughout the next term(s) of development. The following are the primary areas of focus for high availability; note that there may be other improvements that can be made which are not being considered here, hence further research should be conducted to achieve the best high availability results possible from Trackr.

- How to handle more than one database.
    - Prioritizing availability and partitioning and ensuring users are routed to the correct partitions to ensure optimal consistency.
- How to handle multiple servers automatically scaling.
    - Trackr already has rate limiting and a load balancer which can be used to help accomplish this. The current load balancer setup assumes that the additional servers are within the same network.
- How to handle server failures/crashes.
    - This includes hardware issues such as running out of disk space.
- Safeguards and prevention mechanisms against common attacks against availability such as DDoS attacks.
- Implementing additional testing methodologies
    - This will help reduce the possibility of introducing new bugs when updating and deploying Trackr which will reduce the possibility of server crashes.

The primary goal is to ensure the application remains available for long periods of time, and to allow many concurrent users, both systems submitting data to projects such as Arduinos and users accessing the web interface to view the data from projects.

2. Multiple Servers

Currently, Trackr is using load balancing to handle multiple servers, however, there is the current assumption that all servers are running on the same network/virtual machine. Additional modifications may be required to support Trackr running on multiple virtual machines, however, this should not be a large change as it is expected that all Trackr instances will be running on the University of Manitoba network, and hence cross network requests may not be required. If multiple databases are being used (discussed in section 3), then these servers will also require varying environment variables and setups to request the correct database.

3. Multiple Databases

Making use of multiple databases will be required for Trackr in the long term as more organizations and projects are submitting data to Trackr. Currently, all servers are making use of the same database, however, eventually, this will be infeasible. Availability and partition tolerance should be prioritized for

Trackr. Then, requests to specific projects will have to be routed to the correct set of servers designated for the database partition that contains the requested project.

Additionally, to reduce the overall database load, data caching should be implemented on Trackr. As projects are sending time-based streaming data it is expected that many projects will be receiving data on a continuous basis. Instead of starting a database connecting and updating the project's data each time a new field is submitted, Trackr can cache many values before offloading them to the database. This can reduce the amount of database requests significantly.

The current database that is being used in Trackr is a MySQL Docker Image. By just changing the image that is being used, other MySQL-based databases can be used in place, which allows switching to a distributed MySQL database easily. One such example is MariaDB Xpand, a distributed database based on MySQL which supports high availability and scaling. Further research should be conducted to determine the optimal distributed database to use for Trackr. Note that Docker may not be optimal for using a distributed database, however, it can be used for initial testing and development.

4. Handling Server Failures

There are many different server failure issues that must be considered; note that this is not a comprehensive list and there are other issues that may have to be considered down the line in addition to the following failures and potential problems that Trackr may encounter.

4.1 Disk Space Issues

To prevent issues with running out of disk space, multiple methods can be used. Firstly, monitoring can be implemented which can send alerts whenever disk space is running low, and then manual intervention can be taken to prevent issues with running out of disk space. Multiple databases can then be used to help offload the amount of data that is stored on one database. Additionally, projects can be monitored for their last updated/accessed time and any projects that have not been accessed/updated past a certain threshold of time can be moved to a different database; this will minimize the number of projects that are held on the primary Trackr databases, by moving unused projects off and storing them on different servers.

Log files can also take up a considerable amount of disk space and should be monitored and removed when too many logs have been generated.

4.2 Server Redundancy

To help mitigate other hardware failure issues, redundancy should be employed to ensure that if one Trackr instance fails other instances can take over. This will require client modifications as if the main backend API that is being queried fails, the entire Trackr application will no longer work. Hence additional secondary backend API domains can be added to the Trackr frontend client such that if the first API does not respond others can be queried instead.

4.3 Data Backups

To prevent data loss from server failures routine backups should be performed and stored off-site from the main machines that Trackr is being deployed on. Then backups can restore the majority of the data in case of a server failure that results in data loss. However, some data loss may still be inevitable as backups cannot be made constantly, and only during routine times. It is recommended that the best practices for data backups be investigated and implemented in Trackr to reduce potential data loss from the users of Trackr, as a large amount of data loss can be a large problem for organizations and projects that are hosted on Trackr.

4.4 Automatically Restarting Failed Containers

Containers should be set to automatically restart in case of failures by making use of Docker. Additionally, the implementation of tools such as Kubernetes to help manage containers may be beneficial and can aid in this task.

4.5 Failover to Previous Trackr Versions

In case issues occur in production, having previous versions of Trackr deployed which are known to work can assist with faulty deployments. Instead of having Trackr unavailable until a fix is deployed, the previously known working version of Trackr will be used in place. This will require additional research to implement as modifications to the frontend, backend, database, and other parts of the architecture will all affect how failing over to previous versions will work.

5. Protections Against Availability Attacks

To protect against various availability attacks such as DDoS attacks and UDP flooding attacks the following techniques and improvements should be implemented.

5.1 Rate Limiting Improvements

The current rate limit is on a user-only basis, which can allow attackers to use many accounts to perform attacks. These additional rate-limiting techniques can help with this issue:

- **IP address rate limiting**
    - Use IP addresses to prevent additional accounts from being used by the same IP address to overcome user-based rate limiting.
- **API key-based rate limiting**
    - Use API key-based rate limiting to prevent too many requests being submitted to a single project.
- **Daily rate limits**
    - It may be worthwhile to implement a daily rate limit, such that users cannot max out the hourly rate limit constantly. However, this may not need to be implemented.
- **Per-second rate limiting**
    - Prevent too many requests from being submitted in rapid succession by implementing a finer granularity rate limit, such as per-second rate limiting.

5.2 IP Blocking

IP blocking should be implemented to prevent known IP addresses from launching repeated attacks. Currently, no such mechanism exists in Trackr. Further research may be required to investigate techniques to automatically detect malicious requests coming from IP addresses which can then be added to the IP blocked list. Otherwise, manual analysis of network logs during attacks can be performed and malicious IP addresses which are found can be added to the IP block list.

5.3 Traffic Shaping

Traffic shaping is a technique that can be used to help ensure the bandwidth of Trackr does not exceed the maximum supported bandwidth. This is done by delaying requests that are submitted to Trackr by some time and by setting priorities to specific types of network traffic that Trackr will receive. For example, traffic from authenticated users may be prioritized over traffic from non-authenticated requests. There are numerous different traffic shaping techniques, such as deep packet inspection and quality of service. Additional research should be performed to see how traffic shaping should be implemented on Trackr and which techniques (or combination of techniques) would be most beneficial.

It should be noted that implementing traffic shaping in Trackr may require a significant amount of research and time to be done correctly and it is recommended that other techniques to support high availability be investigated and implemented first.

5.4 DDoS Protection Services

There are different services that help with the prevention of DDoS attacks, such as Cloudflare. These services may be beneficial to Trackr, however, additional research is required in order to see the full benefits of implementing these services as well as determining the cost-benefit analysis of the protection offered.

6. Conclusion

In conclusion, these recommendations provide a baseline of high-availability modifications that can benefit Trackr. A range of redundancy, architectural design, failover techniques, and testing quality and usage of various methodologies will be important to ensuring high availability for Trackr.