Carnegie Mellon University

Team Incredi-codes Final Presentation

Aishwarya Agarwal, Arpit Sahni, Chantal D Gama Rose, Moulya Sridhar, Yujia Wang

Reflecting on Docker Swarm for Zero-Downtime Deployment

Features of Docker Swarm:

- Rolling updates
- Service discovery
- High Availability
- Load balancing

Reflections:

- Great choice for zero downtime deployments
- Ensured that our application stayed up and running even during updates
- Updated the containers one by one, ensuring that there is always a healthy version of the application running

Potential Issues and Alternative Approaches

1. **Limited scalability:** Docker Swarm was simple and easy to use, which made it a good choice for small to medium-sized clusters. However, if our application was to grow, we may run into scalability limitations.
2. **Limited flexibility:** Docker Swarm provides a limited set of features compared to more robust container orchestration platforms like Kubernetes.
3. **Limited community support:** Docker Swarm has a smaller community compared to Kubernetes, which made it harder to find support or resources.

Kubernetes, on the other hand, is a highly scalable, flexible, and popular container orchestration platform that can address these downsides.

Experimentation Consideration

Key decisions:

- how to split the users to different models
- which telemetry to use

Split users:

- decide user group with SHA-256 after the requests enter the flask application.
- Two models are running in the same container.

Telemetry Design:

- Average rating
- Total watching time by user

Experimentation Reflection

Split users:

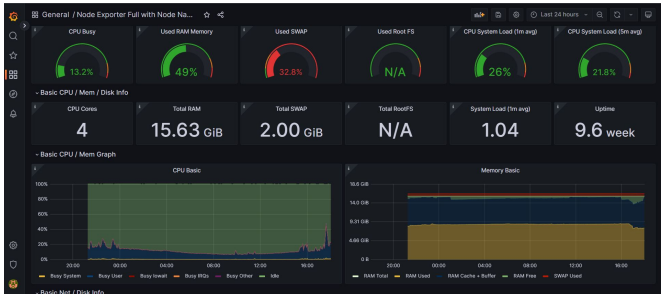
- Problem: Lack flexibility
- Solution: Use load balancers and deploy models in different containers

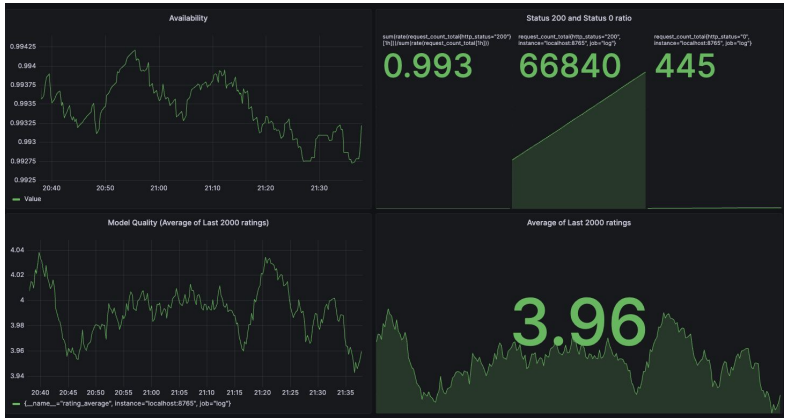
Telemetry Design:

- Problem: Need more direct effect
- Solution: Track the users that were just recommended with movies

Monitoring and Telemetry Design

- Prometheus for scraping and grafana for dashboards
- Simple average of ratings as an online **measure** of the quality of model predictions
- Counting status 200 requests and computing ratio with all other requests for gauging Service availability
- Monitor system vitals via node exporter to gauge the system health.





Carnegie Mellon University

Monitoring and Telemetry Reflection

Potential Issues

- Average rating not a reliable gauge of model performance because users have different preferences and behaviors
- Overall rating Average can be easily biased by majority

Alternative Approaches

- Use other metrics such as precision,recall, F-1 score and AUC-ROC curve.
- Compute metrics on various important subsets of the Data to monitor model performance

Reflecting on Fairness

Plus points of using the evaluation choices

- Visual representation of predicted rating distribution was helpful in detecting lack of representation for female users in the dataset
- Using multiple fairness metrics such as gender parity, Jaccard similarity, and entropy provided a comprehensive evaluation of the fairness of the recommendation system

How fairness was evaluated::

- Distributions of predicted ratings were compared visually for each gender group
- Statistical fairness metrics such as gender parity, Jaccard similarity, and entropy were computed
- Performance of the model was evaluated separately for each demographic group

Potential Issues and Alternative Approaches

Potential issues of using the approach

- Lack of representativeness of the dataset for underrepresented groups can lead to biased recommendations
- The fairness metrics used may not capture all aspects of fairness, such as intersectional fairness

Alternatives and better approaches

- Use data augmentation techniques to increase the size of the dataset for underrepresented groups
- Collect more data from underrepresented groups to improve representativeness
- Use intersectional fairness metrics to evaluate fairness for multiple demographic groups simultaneously

Reflecting on Drift Check

Evaluation and Metrics::

- Drift check is used to ensure that a model's accuracy is not affected by changes in user behavior or preferences.
- Mean and standard deviation are used to calculate the distribution of ratings in the initial training data and the incoming data from the Kafka stream
- A two-sample t-test is used to compare the distribution of ratings.
- A threshold is defined to detect substantial drift. (**5% significance level**)

Why these Metrics::

- Easy for detecting changes in the overall data distribution.
- Ensures a robust way to detect potential drift in the model due to changing user preferences.

Downsides and Alternative Approaches

Downsides of these approach::

- **Assumes a normal data distribution!**
- The t-test may not be suitable for detecting small changes in data distribution.
- The email alerts may generate false alarms, leading to unnecessary interruptions.

Alternative Approaches::

- **Time-based analysis**: Train and test splits w.r.t different time periods
- **Change Point Detection**: Identifies significant changes in data over time, like sudden increases or decreases in ratings. (preventive approach)
- **Machine Learning Approaches**: ensemble models, neural networks and other models can learn patterns in the data and detect changes in its distribution.

Reflecting on working as a team

- Breaking down project into smaller tasks and assign to team members based on strengths or interests.
- Detailed project log, assigning tasks to team members and setting due dates to facilitate tracking of responsibilities and deadlines for each milestone.
- Maintain mutual respect for each other's time and ensure frequent communication regarding progress and any arising issues.
- Our approach of having brainstorming sessions before beginning the work helped a lot in streamlining the process and making key design decisions.
- Regular check-ins with the TA mentor were helpful to set expectations for milestones, considering their open-endedness.

Thank you!