Universität Bremen

Movie Recommendation System

Mohamed Amine Kina
Prakunj Pratap Singh
Seyed Mohammad Taha Tabatabaei
Karanjaspreet Singh

Universität
Bremen

Table of Contents

➔ Goals
➔ Architecture
➔ Model

Goals

Scalability

Scalability Dimensions

Vertical Scalability:
Enhancing the capacity of existing servers or containers by adding more resources (e.g., CPU, memory).

Horizontal Scalability:
Expanding the system by adding more servers or containers to distribute the workload.

Key Focus

With technologies like **Kafka**, **Docker**, **Redis**, and a microservice architecture, the system is optimized for horizontal scalability, leveraging distributed systems to handle growth effectively.

Data Scalability

Handle increasing volumes of data, including:
- User profiles
- Movie datasets
- Interaction logs

Traffic Scalability

Efficiently process:
- Rising request rates
- High concurrent user loads

Processing Scalability

Support growing computational demands for:
- Real-time recommendations
- Complex data processing

Throughput (Requests Per Second)

The number of requests processed per second:
- Track requests via API gateways or load balancers.
- Monitor Kafka message throughput and service-specific metrics.

Load Distribution

How evenly traffic is distributed across nodes, containers, or services:
- Use logs from load balancers or traffic managers.
- Monitor Kafka partition metrics and Redis replication load.

Goals

Diversity

Delivering Diverse Recommendations

Provide users with recommendations that not only match their preferences but also encourage exploration of new and varied content.

Keeps user engagement high by avoiding repetitive or monotonous suggestions.

Key Focus

Recommendations should feel personal, yet surprising, offering users a mix of familiar and novel choices that resonate with their tastes.

Hybrid Recommendation Model

Combines multiple recommendation strategies to create a well-rounded, nuanced system

Collaborative Filtering

Leverages user behavior patterns (e.g., ratings) to suggest what others with similar preferences liked.

Content-Based Filtering

Uses item attributes (e.g., genres, actors, or directors) to recommend similar content.

Relevance vs. Diversity Ratio

Balances two key aspects of recommendation quality:

Relevance

The proportion of recommendations that align closely with user preferences or history.

Diversity

The variety of content offered, spanning genres, themes, or popularity levels.

Metrics

P@k, R@k

Goals

Latency

Minimizing Processing Time for Recommendations

Reduce the time required to process data and generate recommendations, ensuring users receive real-time or near-real-time suggestions.

Enhances user experience with instant and seamless interactions.

Key Focus

Achieving speed without sacrificing the quality or accuracy of recommendations.

Distributed Systems

Utilize technologies like Kafka, Redis, and microservices to handle large-scale data processing and communication efficiently.

Parallelized Computation

Computing recommendations in batch and spliting workloads across multiple servers or containers like Docker

Benefits

Reduces computational load on individual nodes.
Ensures faster response times even under heavy traffic or data loads.

Time to Generate Recommendations

The total time taken from receiving a request to generating and delivering recommendations.

Measurement

Track the end-to-end latency for generating recommendations.

Goal

Optimize processing to maintain a target time (e.g., under 200ms) regardless of system load.

Microservice Architecture

model_service

endpoints
/train
/recommend

ports
7070
6060

database_service

endpoints
/add_user
/add_movie
/delete_user
/close

ports
3303

auth_service

endpoints
/authenticate

ports
3300

analytics_service

endpoints
/history
/preferences

ports
3040

telemetry_service

endpoints
-

ports
-

feedback_service

endpoints
/feedback

ports
6090

ui_service

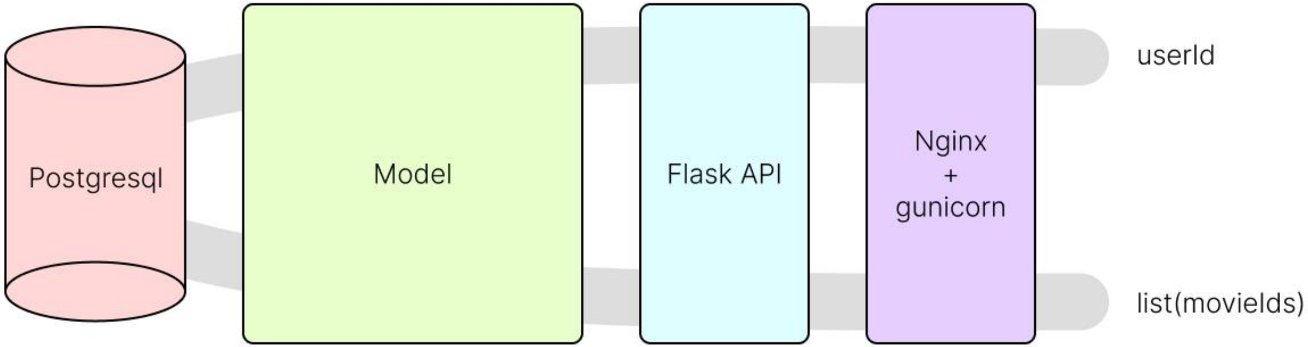
endpoints
/api/movies

ports
5500
3030

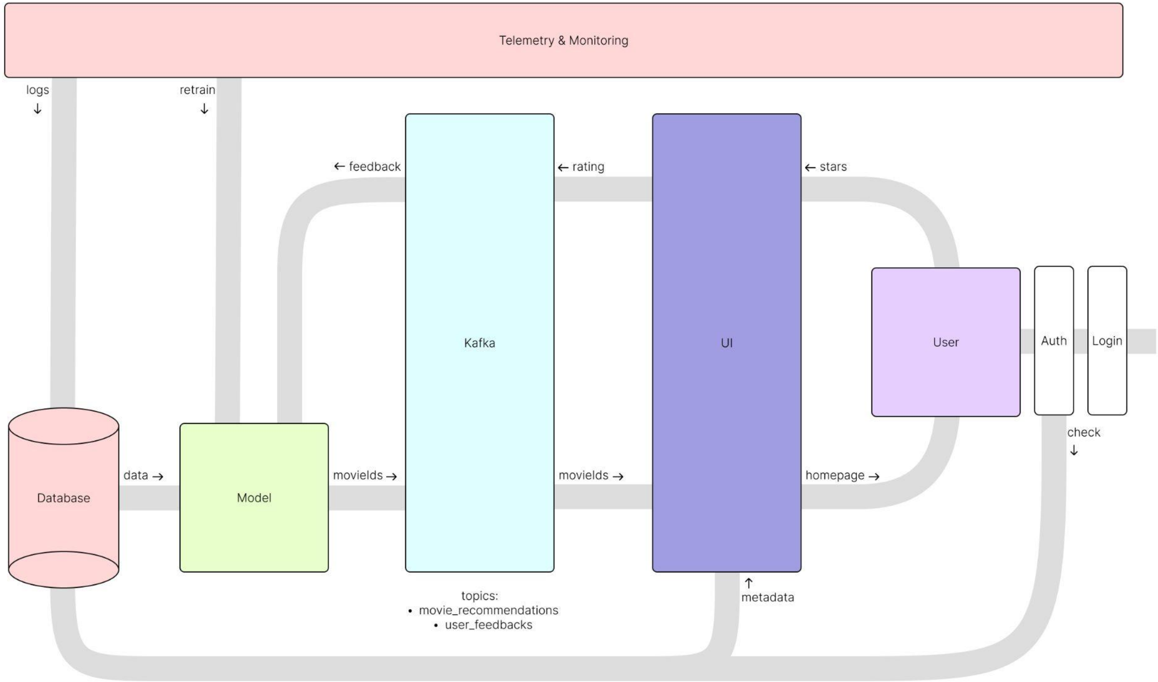
kafka_service

role
Message broker

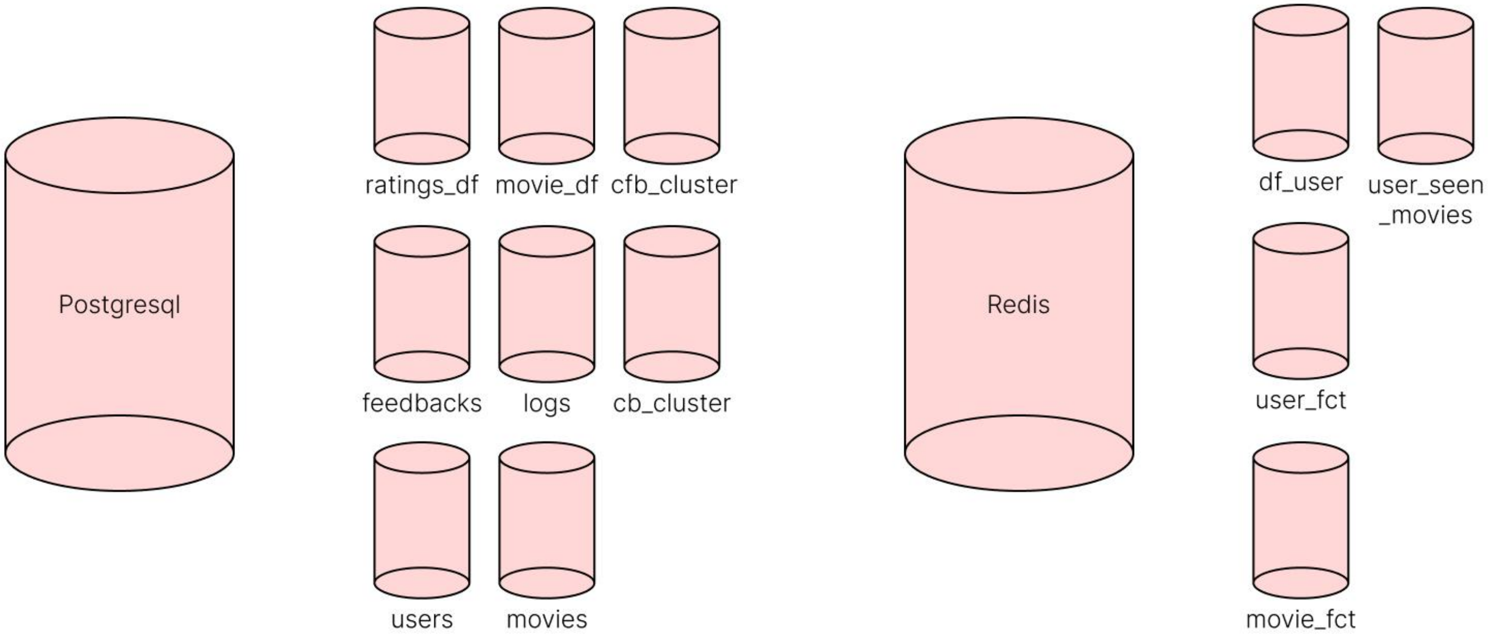
API


Postgresql
Model
Flask API
Nginx
+
gunicorn
userId
list(movieIds)

Data Flow


Telemetry & Monitoring
logs ↓
retrain ↓
← feedback
← rating
← stars
Kafka
UI
User
Auth
Login
check ↓
Database
data →
Model
movieIds →
movieIds →
homepage →
topics:
• movie_recommendations
• user_feedbacks
↑ metadata

Databases


Postgresql
ratings_df
movie_df
cfb_cluster
feedbacks
logs
cb_cluster
users
movies
Redis
df_user
user_seen
_movies
user_fct
movie_fct

Model Architecture

Collaborative-Based Model	Content-Based Model
data	**data**
MovieLens rating.csv	metadata from tmdb
method	**method**
Alternating Least Squares	log scaling by rating

Training Model

```
new_entry = {
        'movieId' : 99999,
        'title' : "Back in Action (2025)",
        'genres' : "Action|Comedy",
        'imdbId' : 99874,
        'tmdbId' : 993710,
        'most_popular_cast' : "Jamie Foxx",
        'director' : "Seth Gordon",
        'original_language' : "en",
        'overview' : "Fifteen years after...",
        'popularity' : 28.184,
        'release_date' : "2025-01-17",
        'poster_path' : "/mLxIlIf2Gopht23v5VFNpQZ2Rue.jpg",
        'vote_average' : 7.6
}
```

Concatenate title and overview

Extract release_year

Popluate missing values

Generate embeddings

Standardize numerical values

One-hot encoding for genres

One-hot encoding for original_language

Perform PCA

```
movie_vector = {
        'movieId' : 99999,
        'movie_feature_1':6.640252897434398,
        'movie_feature_2': 2.6575203909070644,
        'movie_feature_3': -1.8489170305625435,
        'movie_feature_4': 3.953802668802876,
        'movie_feature_5': 2.5796162473539592,
        'movie_feature_6': -0.3405246915045055,
            . . .                   . . .
        'movie_feature_46': -5.432615283071777,
        'movie_feature_47': -4.238303541721341,
        'movie_feature_48': 0.5828429710791725,
        'movie_feature_49': -1.2335654630823163,
        'movie_feature_50': -1.5658623561554261
}
```

Universität
Bremen

Training Model

2.5		4.0	2.0	2.5	
	3.5			4.5	0.5
		1.0		4.0	
5.0			0.5		2.5

train_data

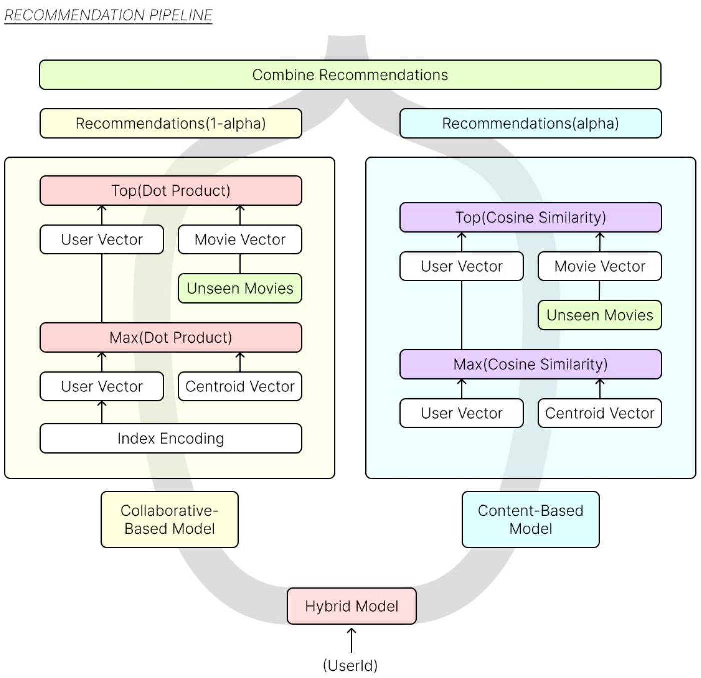
Training

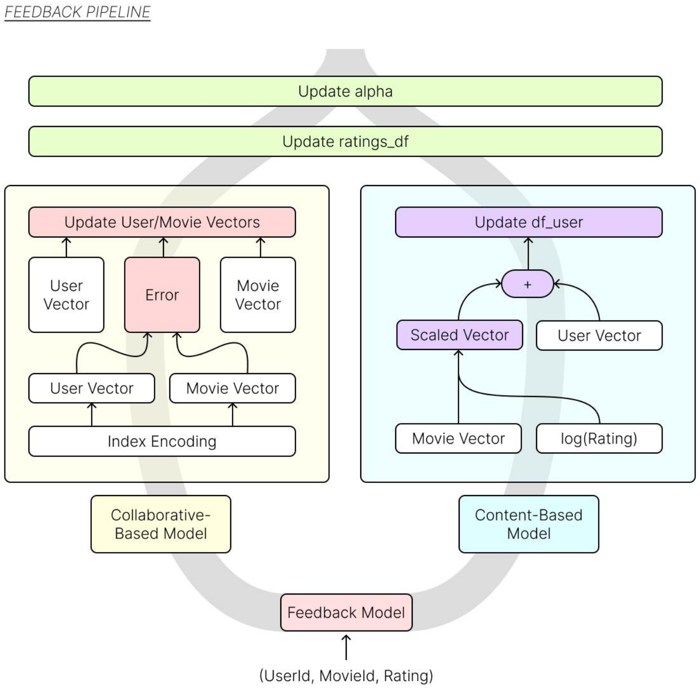
movie_fct

2.5	1.0	4.0	2.0	2.5	5.0
1.5	3.5	3.0	6.0	4.5	0.5
4.0	3.5	1.0	1.5	4.0	7.5
5.0	5.5	2.0	0.5	3.5	2.5

user_fct

Inference & Feedback





Universität
Bremen

Model Evaluation

userId	movieId	rating
22	260	2.0
4885	9887	4.0
957	4751	5.0
76524	21	1.0
22	9875	3.5

ratings_df

userId	movieId	rating
22	997	4.0
4885	1244	4.5
957	4511	5.0
76524	9645	5.0
22	8775	4.0

test_data

Universität
Bremen

Evaluation Results

Precision Evaluation for content-based model*

feature_method	**P@10**	**P@20**	**P@50**
log_scaling	**0.0352**	0.0318	0.0266
subtracting_mean	0.0251	0.0226	0.0188
normalized (-1,1)	0.0320	0.0285	0.0234
baseline_model	x	x	x
random_model	0.0010	0.0011	0.0011

Recall Evaluation for content-based model*

feature_method	**R@10**	**R@20**	**R@50**
log_scaling	0.0021	0.0038	**0.0074**
subtracting_mean	0.0015	0.0028	0.0054
normalized (-1,1)	0.0019	0.0035	0.0066
baseline_model	x	x	x
random_model	0.0000	0.0001	0.0002

**for users with seen_movies > 20*

Evaluation Results

RMSE Evaluation for collaborative-based model

bias	maxIter		rank		regParam			train_data	test_data
	10	15	10	20	0.1	0.2	0.4		
	x		x		x			1.5780	1.7971
x	x		x		x			1.4539	1.7304
x	x		x			x		1.3806	**1.6297**
x	x		x				x	1.3716	1.6394
x	x			x	x			1.4643	1.7531
x	x			x		x		1.3815	1.6328
x		x	x			x		1.3840	1.6338
x		x	x				x	**1.3705**	1.6372