Dashboard Design

I will design the dashboard as a separate service and gather user and app data then send it to the dashboard service using Kafka of RabbitMQ message broker, I do this for several reasons:

1- Data Transfer

- Asynchronous Processing: Prevent the Appstore service from blocking the requests, and Appstore service does not need to wait for the Dashboard to process.
- I don't want the large number of analytical data to impact the Appstore service.
- I also can rely on message brokers because they will store the data in the queue in case the dashboard is down.

2- Data Aggregation

- Because I need to aggregate, and query data with many complex conditions, for example, app download counts, user analytics, or purchased apps belonging to specific uses at specific times I would choose a relational database for example Postgres, but in case of real-time statistics like current active users I would use Redis beside Postgres, because it’s fast (in-memory caching)

3- Scalability

- In terms of scalability, I try to use concurrent consumers in dashboard service, and maybe in multiple queues and multiple routing keys to separate all kinds of messages, I also choose Kafka rather than RabbitMQ, because it is more sufficient for high-volume processes.
- Also, I use read replicas in case of high traffic and partitioning for example I would partition based on Category, or time series.
- I would use a load balancer for deploying multiple instances, and use CDN for loading media files.