News Tracker Application

Novelty:

The news tracker application is ideated in such a way that the users will be able to get only the relevant information and news of their chosen topic. The features are as follows:

1. Each user has their own profile which will contain information about their chosen or interested topics, and news will be fetched and displayed from only those topics. Furthermore the topics can later be changed as per the users wish.

2. Displaying various stats regarding the user's analytics based on the viewing of various new articles, which allows the user to know about his own preferences and interests in a deeper manner.

3. Option to save stories which can be seen in the users profile, this will allow the user to keep some news stories for quick reference in the future. This is a very useful feature which saves a lot of time for the user, in case they need to revisit something they read or saw before.

4. Allows the users to follow a particular news publication, so that the user can get to know about all the latest news updates provided by the selected news publication.

5. This application not only focuses on the new readers, it can also act as a portal for the news publication houses to understand their viewership and hence we can monetize this to make the app sustainable for the developers.

Feasibility:

1. The app uses a very simple, lightweight flask backend, optimized for production environments. This makes it easy to engineer, develop and deploy.

2. IBM Cloud provides the necessary infrastructure such as K8s clusters, databases etc. required for efficient deployment.

3. There are many open source news APIs with verified news articles, which may be used as preliminary data sources, for the serving of curated news articles to the users

Since the proposed architecture is simple and lightweight, and the necessary tools and infrastructure are readily available, it can be concluded that this solution is feasible.

Business Model:

We plan to propose the following business model to maximize the revenue:

1. Provide customized advertisements to the user through various ad providers: Google Ads, Amazon Ads, Facebook Ads, etc.

2. As the product is more customer focussed, we could generate a humongous amount of data which is related to user activity, screen time, favorite type of news, etc. This could help sell the data to many other companies with prior consent from the user.

3. Along with a B2C mindset, we also propose a B2B model, where we bring in news publishers which are ad - dependent. This could help get businesses and normal users as customers which could help us generate revenue from either side.

4. We also propose to sell branded content from the news publishers where the normal users could buy something like a subscription and follow their favorite news publishers.

5. A premium subscription model for the normal news consumers is proposed to be introduced so that the users can get rid of ads along with helping us generating more revenue through subscriptions.

Social Impact:

1. News is important for a number of reasons within a society. Mainly to inform the public about events that are around them and may affect them.

2. Often news is for entertainment purposes too; to provide a distraction of information about other places people are unable to get to or have little influence over. News can make people feel connected too.

3. News is important as a social gathering space too, hence newspapers either online or physical place an emphasis on news. Where there are a lot of people gathered there is opportunity to advertise. This advertising sometimes can cause a conflict of interest in the way news is reported.

Scalability:

1. A big part of developing a web app is its capacity to scale. We are aiming to launch a product that has to be ready for the influx of users and expect the system to handle it. Be extra vigilant because there might be a time when our system is not flexible enough and cannot support a heavy load. To prevent this, it is critical to get started on application scalability before the development step comes in.
2. While developing the application, as developers we kept in mind the scalability factor and made sure that our application would have large scalability
3. We have used Python Flask server as our backend which is one of the best when it comes to scalability.
4. Flask by itself is only limited in terms of scaling by your application code, the data store you want to use and the Python implementation and web server you are running on.
5. Moreover we have used IBM DB2 as our database. So now the question arises how does Flask server plus IBM DB2 increase the scalability of the application. The answer to this question is addressed below

Incremental growth
The Parallel Sysplex cluster can grow incrementally. You can add a new Db2 subsystem onto another central processor complex and access the same data through the new

Db2 subsystem. You no longer need to manage copies or distribute data. All Db2 subsystems in the data sharing group have concurrent read-write access, and all Db2 subsystems use a single Db2 catalog.

Workload balancing

Db2 data sharing provides flexibility for growth and workload balancing. With the partitioned data approach to parallelism (sometimes called the shared-nothing architecture), a one-to-one relationship exists between a particular DBMS and a segment of data. By contrast, data in a Db2 data sharing environment does not need to be redistributed when you add a new subsystem or when the workload becomes unbalanced. The new Db2 member has the same direct access to the data as all other existing members of the data sharing group.

Db2 works closely with the z/OS Workload Manager (WLM) to ensure that incoming work is optimally balanced across the systems in the cluster. WLM manages workloads that share system resources and have different priorities and resource-use characteristics.

For example, assume that large queries with a low priority are running on the same system as online transactions with a higher priority. WLM can ensure that the queries do not monopolize resources and do not prevent the online transactions from achieving acceptable response times. WLM works in both a single-system and a multisystem (data sharing) environment.

Capacity when you need it

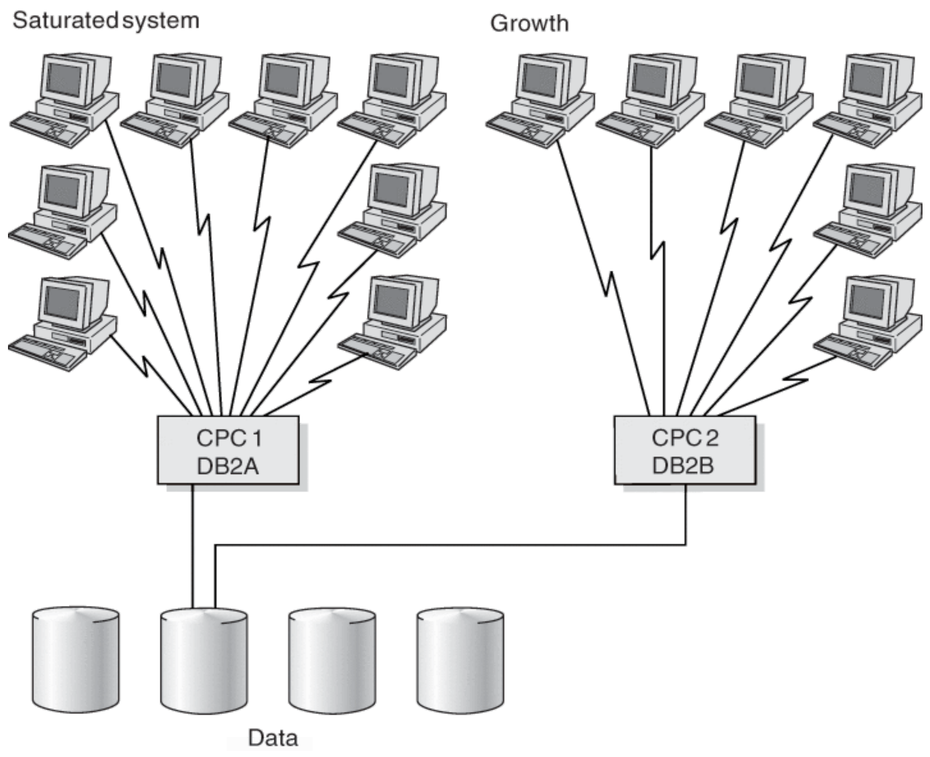
A data sharing configuration can handle your peak loads. You can start data sharing members to handle peak loads, such as end-of-quarter processing, and then stop them when the peak passes.

You can take advantage of all these benefits, whether your workloads are for online transaction processing (OLTP), or a mixture of OLTP, batch, and queries.

Higher transaction rates

Data sharing gives you opportunities to put more work through the system. As the following figure illustrates, you can run the same application on more than one Db2 subsystem to achieve transaction rates that are higher than are possible on a single subsystem.

Figure 1. How data sharing enables growth. You can move some of your existing Db2 workload onto another central processor complex (CPC).



We have deployed our server and the application in Kubernetes

Considerations for large clusters

A cluster is a set of nodes (physical or virtual machines) running Kubernetes agents, managed by the control plane. Kubernetes v1.25 supports clusters with up to 5000 nodes. More specifically, Kubernetes is designed to accommodate configurations that meet all of the following criteria:

- No more than 110 pods per node
- No more than 5000 nodes
- No more than 150000 total pods
- No more than 300000 total containers

You can scale your cluster by adding or removing nodes. The way you do this depends on how your cluster is deployed.

Cloud provider resource quotas

To avoid running into cloud provider quota issues, when creating a cluster with many nodes, consider:

- Requesting a quota increase for cloud resources such as:
    - Computer instances

    - CPUs
    - Storage volumes
    - In-use IP addresses
    - Packet filtering rule sets
    - Number of load balancers
    - Network subnets
    - Log streams
- Gating the cluster scaling actions to bring up new nodes in batches, with a pause between batches, because some cloud providers rate limit the creation of new instances.