Albert Wen

Cloudflare Internship Application: Product Management

Executive Summary

Video game developers require large data infrastructure to capture metrics from their players. This data is often crucial towards the function of the game or used for analytics to maximize user retention and create monetization opportunities[1]. As a result, video game developers would prioritize consistency more than Cloudflare Worker's KV does currently. Therefore, **Cloudflare Workers KV** as a data store can be leveraged to expand into this vertical by **changing the data system to a session consistent level and building capabilities to store dynamic data**.

Plans for Further Market Research

A significant portion of the market research I conducted was focused on late-stage development of larger, established video game franchises because that is the most easily accessible data given limited resources. As a result, the recommended features below are most optimized for games that have large consumer bases and high volumes of data. For the future, further research should focus on learning about the video game development cycle and challenges faced by video game developers at every stage of the process. That will allow Cloudflare to tailor features specifically for indie game developers or games where scale is still small in order to ensure higher customer retention. Methods for conducting such research could involve interviews of game developers, or company case studies. Throughout this research process, the chief goal would be to learn more about various teams' priorities in game development, differentiating factors between game genres, and specific requirements that must be met for developing generalizable game features.

Product Changes and Additions

As it stands, Workers KV is a low-latency key-value data store that is incredibly fast when used for high-read applications and optimized for persistent data. However, video game users generate a high volume of new data and have large quantities of changing data. As such, I recommend that Workers KV adds a feature that allows users to change the consistency of their data store to session consistency, as opposed to the current eventual consistency. Alongside that, Workers KV should develop capabilities to handle dynamic data for the video game industry. There are a few reasons for these changes.

1. Changing Current Information

With eventual consistency, users may read outdated information because changes in KV pairs take a long time to propagate. With session consistency, users in the same session see updates much quicker and creates a better user experience, and is therefore a preferable consistency level for video game developers. Especially when adjusting scores, item inventories, or similar features in video games, updates in real time are important.

2. In-order Reads

In a system with eventual consistency, writes are not always read in order. In video games, sequences of events are often important and valuable information can be lost if only evaluating

[1] https://dzone.com/articles/how-important-is-data-in-game-development

aggregates. The consistency prefix level for session consistency guarantees that read never sees out of order writes.[2]

Methods for Quality Improvement

To iterate on this feature before release, it would be best to user-test what levels of consistency are most desirable for the largest genre of game developers. With each iteration, we should user-test to gather data from specific segments of game developers to reduce confounding variables; we can segment based on development stage and game genre. If other consistency levels are more desirable for a larger segment of game developers, then we can adjust accordingly. As well, time and cost of development for Cloudflare's engineering team should be taken into consideration; for instance, the consistency level and data needs for single player games and multiplayer games may be drastically different, and the ease of implementation and size of market should both be taken into consideration when iterating to improve quality before release.

Goals to Measure Success

Below is a brief list of metrics to measure the success of the previously mentioned features and their justifications.

Metric	Justification
% of New Customers Acquired	To measure success of new features in attracting new customers onto Cloudflare's platform (higher is better).
Reported Number of Inconsistencies	To measure number of instances in which clients report an issue with the feature (lower is better)
Requests by user and type	Since Cloudflare's unlimited plan charges by requests, revenue generated by Workers KV is driven by requests (higher is better).
Market share	To measure the number of game developers that have chosen to use Cloudflare Workers for Gaming as their tool (higher is better).
Margin per request	To measure how many cents Cloudflare earns per million requests (higher is better).

Future Risks

There are a few risks associated with implementing these new features. Foremost is the **increased cost associated with a greater burden on data centers** derived from game developers storing dynamic data instead of persistent data. Because there will be an influx of data and requests on Cloudflare's data centers, the energy, maintenance, and new construction costs may be high and thus potentially reduce margins. However, given the large size of Cloudflare's current network and the effect of economies of scale as operations increase, these costs should be somewhat mitigated.

Second, other competitors in the industry have implemented similar features for their distributed databases, so it **may be hard for Cloudflare to gain a competitive edge**. For instance, Microsoft's Azure DB also allows users to change consistency levels.[3] But, **Cloudflare's CDN is larger and the gaming industry is growing incredibly fast and is relatively fragmented,** so Cloudflare should have large enough market share when expanding into the video game vertical.

---

[2] https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels
[3] https://docs.microsoft.com/en-us/azure/cosmos-db/