In essence, these platforms combine

1. global authentication,
2. selective essential/critical data replication to different region and
3. intelligent routing to nearest region to get essential info/detail.
4. intelligent routing to **primary region until data is replicate/copied in current region. ( we can cache for some time like 5 day ,1 month etc based on business need to make sure that user is goinmg to stay in this region long.)**

## how user login work from different region in Facebook linked in or Instagram in GEO based partition?

**Summary**

In essence, these platforms combine

1. global authentication,
2. selective essential/critical data replication on **geo-distributed databases** and
3. intelligent routing to user' GEO near DB /.

to enable a smooth login experience, regardless of region. By caching essential data locally, relying on CDNs for static resources, and managing sessions globally, they deliver low-latency performance even when users switch regions. This approach optimizes both performance and cost while respecting geo-partitioning constraints.

When a user logs into geo-partitioned services like Facebook, LinkedIn, or Instagram from a different region, the systems employ several sophisticated techniques/algorithms to ensure quick and seamless access. These platforms rely on **geo-distributed databases, caching, multi-region replication, and consistent login services** to manage the login process efficiently. Here’s a closer look at how login works in this scenario:

**1. Global Authentication Services**

* The first step in any login process, regardless of location, is user authentication. Facebook, LinkedIn, and Instagram have globally distributed authentication servers that handle logins independently of the user’s data location.
* These platforms use identity servers that replicate user credentials and login states across regions, allowing them to authenticate users locally. This way, when a user logs in from a different region, their credentials can be validated without needing to access their primary data location.

**2. Intelligent Routing and Load Balancing**

* Once authentication succeeds, global load balancers determine the nearest or most appropriate data center to handle the user’s session. These load balancers look at factors such as:
  + **User location** (nearest available region)
  + **Server load and latency**
  + **Regional data presence** (availability of user data in the region)
* The system then routes the login session and subsequent requests to this optimal data center, ensuring the user experiences minimal delay.

**3. Session State Replication and Global Session Management**

* To make login sessions accessible across different regions, these platforms use session state replication. When a user logs in, session information (like tokens, cookies, and user context) is distributed across relevant regions.
* This allows the user to access their session quickly, even if they move across regions or log in from multiple devices in different locations.

**4. Dynamic Caching and Temporary Data Replication**

* After login, a temporary copy of essential user data (such as profile information, friend lists, recent posts, and notifications) is often cached in the new region. This reduces the need for the system to repeatedly fetch data from the user’s primary data partition.
* Caching is dynamic, meaning as the user continues to interact with the app, more data is cached in the local region as needed, ensuring a faster response.

**5. Partitioned Data Retrieval and Eventual Consistency**

* In geo-partitioned architectures, each user’s primary data (DK not essential) (including personal content, activity logs, and relationships) is often stored in a specific “home” region to avoid global data replication.
* When a user logs in from a different region, essential data is either fetched directly from the primary region or retrieved from a cached version if available (**DK- we can maintain flag if all data is replicated to nearest region, till that forward user request to primary region and cached that**). For dynamic content (like live comments or recent notifications), platforms often use **eventual consistency** models, allowing this data to update gradually across regions as necessary.

**6. Edge Caching and CDNs for Static Content**

* For multimedia and static assets (e.g., photos, videos), Facebook, LinkedIn, and Instagram rely on Content Delivery Networks (CDNs) and edge caching to store copies of these assets closer to users globally.
* During login, if the user requests media files (like their profile photo or timeline images), these are delivered from the nearest CDN node, minimizing latency.

**7. Session Failover and Availability Zones**

* To ensure high availability during the login process, these platforms use multiple availability zones and failover mechanisms. If one data center experiences high latency or becomes unreachable, the system can seamlessly route the session to another data center within the same region or a nearby region without interrupting the user experience.

**8. Read/Write Optimization and Replication Control**

* For less critical data requests, like viewing historical posts or notifications, the system may allow temporary read-only access while data is still being fetched or replicated in the new region.
* Any updates or writes to user data (like adding new connections or posting updates) are propagated back to the primary region for consistency. This allows the system to prioritize local reads while maintaining data integrity.

## How Facebook loading page when user scrolling the page down

When you scroll down on Facebook, the platform loads additional content dynamically using a process called **infinite scrolling**. This approach allows Facebook to display new posts, photos, videos, and other updates seamlessly without requiring a page refresh. Here’s how it works:

**1. Lazy Loading via API Requests**

* Facebook uses **lazy loading** to fetch additional content as you scroll down. Instead of loading all content at once, which would slow down the page, the app only loads a certain number of posts initially.
* When you reach near the bottom of the currently loaded content, the app automatically triggers an API request to the server, fetching the next set of posts or stories in the background.

**2. AJAX Requests and Asynchronous Loading**

* Infinite scrolling on Facebook relies heavily on **AJAX (Asynchronous JavaScript and XML)**. When you scroll near the end of the content displayed, the frontend code sends an AJAX request to Facebook’s servers to request the next batch of posts.
* This AJAX request is asynchronous, meaning it doesn’t block other actions on the page. While new content is fetched in the background, you can continue interacting with loaded posts.

**3. Batch Processing and Data Pagination**

* Facebook’s backend delivers content in **batches** or “pages.” Each batch includes a certain number of posts or stories, optimized to balance speed and data transfer efficiency.
* Pagination allows Facebook to control how much data is sent with each request, keeping data transfer minimal and reducing strain on both the user’s device and Facebook’s servers.

**4. Pre-Fetching and Predictive Loading**

* As you scroll, Facebook may **pre-fetch** some of the next content in anticipation of you reaching it. This predictive loading improves the experience by having content ready when you reach it, reducing the appearance of loading delays. <https://stackblitz.com/edit/angular-scroll-to-bottom-load-more-items?file=src%2Fapp%2Fscroll-tracker.directive.ts>
* Pre-fetching is often done for content just outside the visible area, ensuring that if you scroll faster, the new content appears almost instantly.

**5. Content Ranking and Prioritization**

* Facebook uses algorithms to prioritize and rank content based on relevancy, so the posts loaded are those the system predicts you’d find most engaging.
* This ranking affects which posts appear in each batch as you scroll, meaning that each subsequent load may prioritize newer or more relevant posts to your interests and interactions.

**6. Caching for Faster Access**

* Facebook’s front end caches certain elements and static resources (like profile images, reactions icons) in the browser. This caching minimizes data transfer and accelerates the loading of repeat elements across posts.
* For dynamic content, Facebook may also use short-term caching, reducing the need to re-fetch the same post content if you scroll back up or revisit the area briefly.

**7. Database Query Optimization**

* On the backend, Facebook uses optimized database queries to retrieve user-specific content, often leveraging **distributed databases** like TAO (a Facebook-developed database optimized for social media graph data).
* These queries are designed to be highly efficient, retrieving only the relevant, unseen posts while balancing server load and data transfer costs.

**8. User Interaction Tracking and Feedback Loop**

* Facebook tracks user interaction (e.g., likes, shares, comments) and uses this feedback to adjust the content it loads. For example, if you engage more with video content, it might prioritize video posts in future batches as you scroll.
* This feedback loop also helps in ad placement, where ads are loaded as part of the scrolling content based on relevance and targeted ad campaigns.

**9. Smooth User Experience with Virtual Scrolling**

* To reduce memory usage, Facebook implements **virtual scrolling**, where only a portion of the loaded content remains in memory. As you scroll further, older content is unloaded from memory, while the newer content is fetched.
* This technique allows the page to run smoothly without overwhelming device memory, as only a subset of posts is displayed and stored in the DOM at any given time.

In short, Facebook’s scrolling experience is powered by AJAX, lazy loading, and caching, which together deliver content in manageable chunks without overloading your device or bandwidth. This design ensures a fast, seamless experience that’s both resource-efficient and responsive.