Zanzibar

Overview

- System for determining whether users are authorized to access digital objects
- Allows clients to create, modify, and evaluate access control lists (ACLs)
  - Of the form user U has relation R to object O
  - The user U can also be a set of users (i.e. another relationship)
  - Examples:
    - doc:readme#owner@10 - User 10 is an owner of doc:readme
    - group:eng#member@11 - User 11 is a member of group:eng
    - doc:readme@viewer@group:eng#member = members of group:eng are viewers of doc:readme
- ACL checks are just of the form:
  - check(object, user, action) -> yes/no
  - In ACL database, Zanzibar looks in its DB for all ACL entries starting with object, action
    - I.e. doc:readme#owner
  - Due to groups, it prob won't find the specific user, but it will find group names
    - It then does a search down these groups
    - An ACL check can take a lot of work depending on these groups
- Data is stored in a Spanner deployment specifically for Zanzibar
  - Uses ~1500 Spanner servers but to get 200M reads per second to get the throughput they want, they would need 15,000
    - This assumes perfect load balance and that hot spots aren't a problem
- We also split up work among multiple aclservers to process client requests
  - Using consistent hash of the object we are searching for, we shard among the aclservers
  - Searches down the group tree require communication among the aclservers
  - Helps with hot-spots by bringing together simultaneous accesses for a given object
    - Cache for RPC results will then hit more often
      - 10% for RPC results, which is low due to random access but still helps with hotspots
    - We can also prefetch objects that are being commonly used
    - The aclserver also has a cache of Spanner data that is prob higher hit rate
  - Using this we can go from 10M client requests requiring 20M instead of 200M spanner reads

Consistency

- Zanzibar is **not** linearizable
  - Spanner replicas may have not seen latest writes
  - aclserver caches would have to be kept up to date which can be hard
- What we want:
  - ACL checks must respect order in which users modify ACLs
  - ACLs update order should be maintained
    - If someone is removed from the ACL of a folder, then they should not be able to see any new contents of that folder (but may still be able to access the old contents depending on the ACL)
      - If the ordering of the changes is neglected, then the ACL check may say that person can access those new contents
  - ACLs should not see stale data
    - If someone is removed from the ACL of a folder, it should not be possible to get old info that says that person can still access said folder
  - Two key consistency properties:
    - External consistency:

        - We can assign timestamps to each ACL / content update so that two causally related updates will have timestamps that respect said ordering
        - If a read observes an update x, then it must observe all updates that happen causally before x
    - Snapshot reads with bounded staleness:
        - Whenever our content is updated at some timestamp, we need to make sure that whatever we read is at least up to date with that timestamp so we know how much this person had permission to see at that point in time
- This means that if a person is removed from a document, they will still be able to see it for some time, but must not be allowed to see any content updates after they are removed

---

- ACLs are stored in Spanner which has a TrueTime mechanism that assigns each ACL with a timestamp
    - We evaluate each ACL check at a single snapshot timestamp
        - Only writes with timestamps up to that query timestamp are visible
    - If read time is too recent, then we are forced to wait (but Zanzibar would prefer to avoid this)
- We could use always the latest snapshot but this would yield high latency
    - Instead, we use a protocol that allows most checks to be evaluated on replicated data
    - A Zanzibar client requests an opaque consistency token called a zookie for each content version via a content-change ACL check when the content modification is about to be saved
        - Zanzibar gives this zookie a global timestamp and ensures that all ACL writes prior to this content change have lower timestamps
        - Client stores the zookie in an atomic write to client storage
    - The client sends this zookie in subsequent ACL check requests to ensure that the check snapshot is at least as fresh as the timestamp for the content version
    - This allows us to only use this timestamp and not have to wait for Spanner to catch up