Zanzibar: Google’s Consistent, Global Authorization System

Ruoming Pang, Ramón Cáceres, Mike Burrows, Zhifeng Chen, Pratik Dave, Nathan Germer, Alexander Golynski, Kevin Graney, and Nina Kang, Google; Lea Kissner, Humu; Jeffrey L. Korn, Google; Abhishek Parmar, Carbon; Christopher D. Richards and Mengzhi Wang, Google

{rpang,caceres}@google.com
Authorization checks are central to preserving privacy.
Zanzibar stores access control lists (ACLs)

"Video X is public"

"Group Y manages Cloud project Z"
...and performs authorization checks based on stored ACLs

"Can Alice view video X?"

"Can Bob create a VM under Cloud project Z?"
Zanzibar is...

- **Consistent**: Respects causal ordering of updates to ACLs and objects
- **Flexible**: Supports a rich variety of access control policies
- **Scalable**: Trillions of ACL entries, millions of checks/second
- **Fast**: Less than 10ms @ 95%, less than 100ms @ 99.9%
- **Available**: 99.999% over the past 3 years
Namespaces, relations, usersets, and tuples

**Namespace**: videos

<table>
<thead>
<tr>
<th>Object</th>
<th>Relation</th>
<th>Userset</th>
</tr>
</thead>
<tbody>
<tr>
<td>video X</td>
<td>viewer</td>
<td>user A</td>
</tr>
<tr>
<td>video Y</td>
<td>viewer</td>
<td>All Users</td>
</tr>
</tbody>
</table>

System-defined value
Namespaces, relations, usersets, and tuples

**Namespace:** videos

<table>
<thead>
<tr>
<th>Object</th>
<th>Relation</th>
<th>Userset</th>
</tr>
</thead>
<tbody>
<tr>
<td>video X</td>
<td>viewer</td>
<td>user A</td>
</tr>
<tr>
<td>video Y</td>
<td>viewer</td>
<td>All Users</td>
</tr>
</tbody>
</table>

**Check results:**

- video X, viewer, user A? Yes
- video X, viewer, user B? No
- video Y, viewer, user A? Yes
- video Y, viewer, user B? Yes
Userset indirection can create deep/wide hierarchies

**Namespace:** videos

<table>
<thead>
<tr>
<th>Object</th>
<th>Relation</th>
<th>Userset</th>
</tr>
</thead>
<tbody>
<tr>
<td>video X</td>
<td>viewer</td>
<td>user A</td>
</tr>
<tr>
<td>video X</td>
<td>viewer</td>
<td>(group 1, member)</td>
</tr>
</tbody>
</table>

**Namespace:** groups

<table>
<thead>
<tr>
<th>Object</th>
<th>Relation</th>
<th>Userset</th>
</tr>
</thead>
<tbody>
<tr>
<td>group 1</td>
<td>member</td>
<td>user B</td>
</tr>
<tr>
<td>group 1</td>
<td>member</td>
<td>user C</td>
</tr>
</tbody>
</table>
Userset indirection can create deep/wide hierarchies

Namespace: videos

<table>
<thead>
<tr>
<th>Object</th>
<th>Relation</th>
<th>Userset</th>
</tr>
</thead>
<tbody>
<tr>
<td>video X</td>
<td>viewer</td>
<td>user A</td>
</tr>
<tr>
<td>video X</td>
<td>viewer</td>
<td>(group 1, member)</td>
</tr>
</tbody>
</table>

Namespace: groups

<table>
<thead>
<tr>
<th>Object</th>
<th>Relation</th>
<th>Userset</th>
</tr>
</thead>
<tbody>
<tr>
<td>group 1</td>
<td>member</td>
<td>user B</td>
</tr>
<tr>
<td>group 1</td>
<td>member</td>
<td>user C</td>
</tr>
</tbody>
</table>

Check results:
- video X, viewer, user B? Yes
- video X, viewer, user D? No
“New enemy” protection

Alice ➔ Google Drive ➔ Bob

Bob ➔ Google Drive ➔ Charlie
“New enemy” protection
Consistency protocol

ACL update by Alice:
UpdateACL(doc X, viewer, remove Bob)

timestamp T0

Leverages Spanner’s TrueTime mechanism [Corbett et al. 2012]
Consistency protocol

ACL update by Alice:
UpdateACL(doc X, viewer, remove Bob)

Content update by Charlie:
CheckContentUpdate(doc X, writer, Charlie)

Yes, timestamp T1 [T1 > T0]
Consistency protocol

ACL update by Alice:
UpdateACL(doc X, viewer, remove Bob) [timestamp T0]

Content update by Charlie:
CheckContentUpdate(doc X, writer, Charlie) Yes, timestamp T1 [T1 > T0]

ACL check for Bob:
CheckACL(doc X, viewer, Bob, T1) No [at T2 ≥ T1 > T0]
Architecture

Zanzibar serving cluster

Write, Check, Read, Expand

aclserver

client

watchserver

Watch

Spanner global database system

read/write

namespace configs

read/write

namespace 1 relation tuples

append

namespace N relation tuples

tail

changelog

Read, Expand

client

write/append

namespace 1

read/write

namespace N
Implementation techniques

- Timestamps chosen to reduce latency
- Hot-spot mitigation to increase availability
- Request hedging to reduce tail latency
- Isolation to protect against misbehaving clients
- Optimized processing of large and deeply nested sets
Deployment

- Zanzibar has been in production use for > 5 years
- > 1,500 namespaces defined by hundreds of clients
- > 2 trillion relation tuples replicated in several dozen locations worldwide
- > 10 million client queries per second, mostly read-only
- > 10,000 servers in several dozen clusters worldwide
Check queries per second

Checks peak at 4.2M QPS, Reads at 8.2M, Expands at 760K, Writes at 25K
Check Safe latency

95th-percentile latency is below 10 ms, 99.9th-percentile below 100 ms
Availability over the last 3 years has remained above 99.999%
Summary

- Robust authorization checks are central to preserving privacy
- Zanzibar is a unified authorization system for Google services
  - Respects causal ordering of user actions
  - Supports a rich variety of access control policies
  - Offers low latency and high availability
  - Scales to trillions of ACL entries and millions of checks per second
  - Supports hundreds of services used by billions of people

Come visit us at our poster tonight during 6:00-7:30pm PDT!