Caching

Google Cloud

Why Use Caching ?

Why Use Caching ?

- Performance

Why Use Caching ?

- Performance
- Stability

apigee

Why Use Caching ?

- Performance
- Stability
- Scalability

apigee

Why Use Caching ?

- Performance
- Stability
- Scalability
- Persistence

apigee

Why Use Caching ?

- Performance
- Stability
- Scalability
- Persistence
- Security

apigee

Caching

Caching

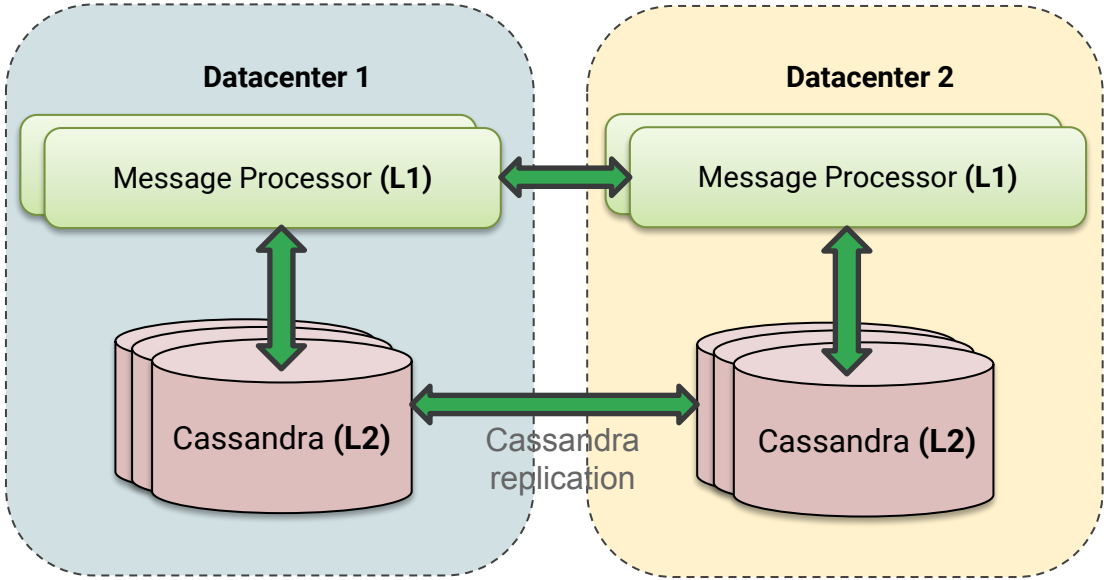
- **In-Memory Cache (L1)**
  - fast access
  - certain percentage of memory per message processor
  - entries are removed in the order of time since last access

Caching

- **In-Memory Cache (L1)**
  - fast access
  - certain percentage of memory per message processor
  - entries are removed in the order of time since last access
- **Persistent Cache (L2)**
  - cache datastore per message processor
  - persisted even if removed from L1
  - no limit on the number of cache entries
  - expired only on the basis of expiration settings

apigee

Distributed Caching



Caching Policies

- Response Cache
    - caches the entire HTTP response (headers, payload, etc.)
    - configure time-to-live
    - honor HTTP cache headers
    - caching of multiple formats
    - attached at request and response segments

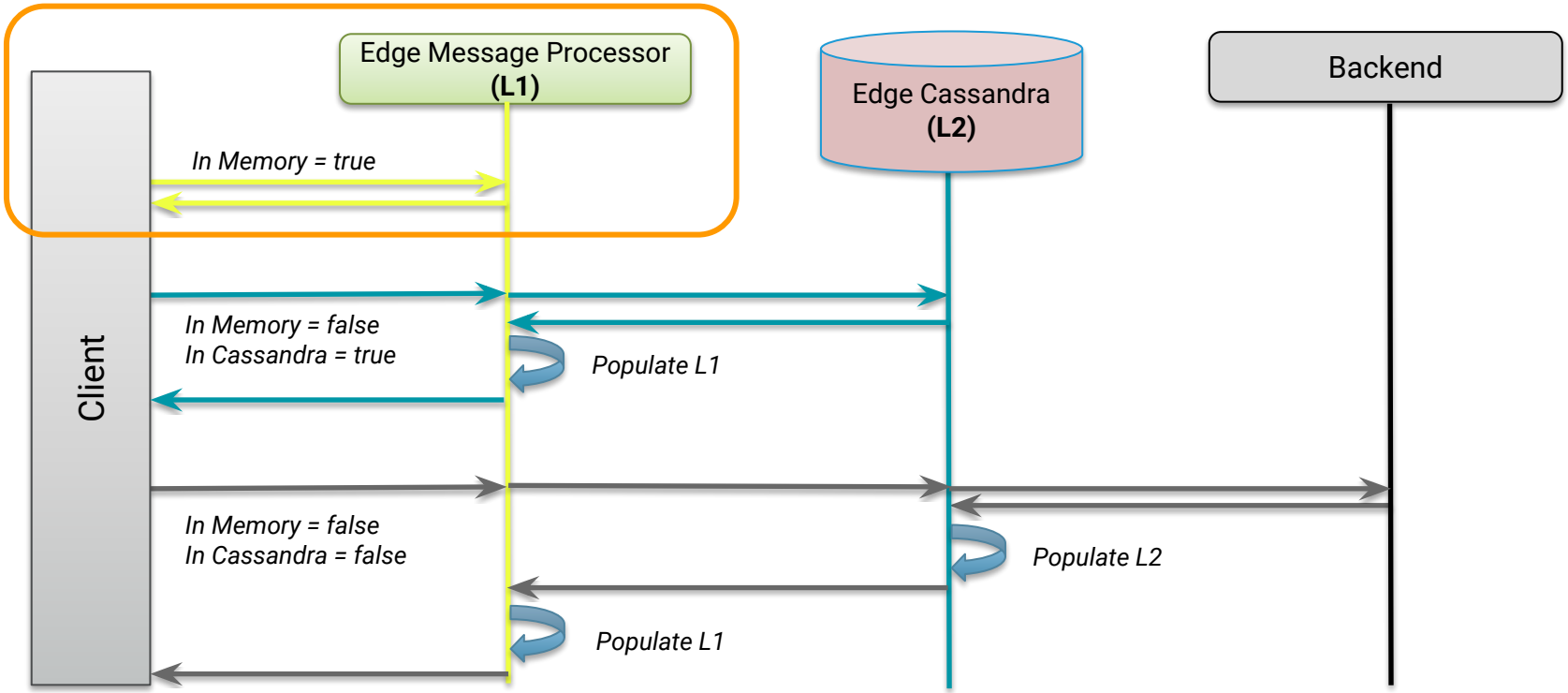
apigee

Caching Policies

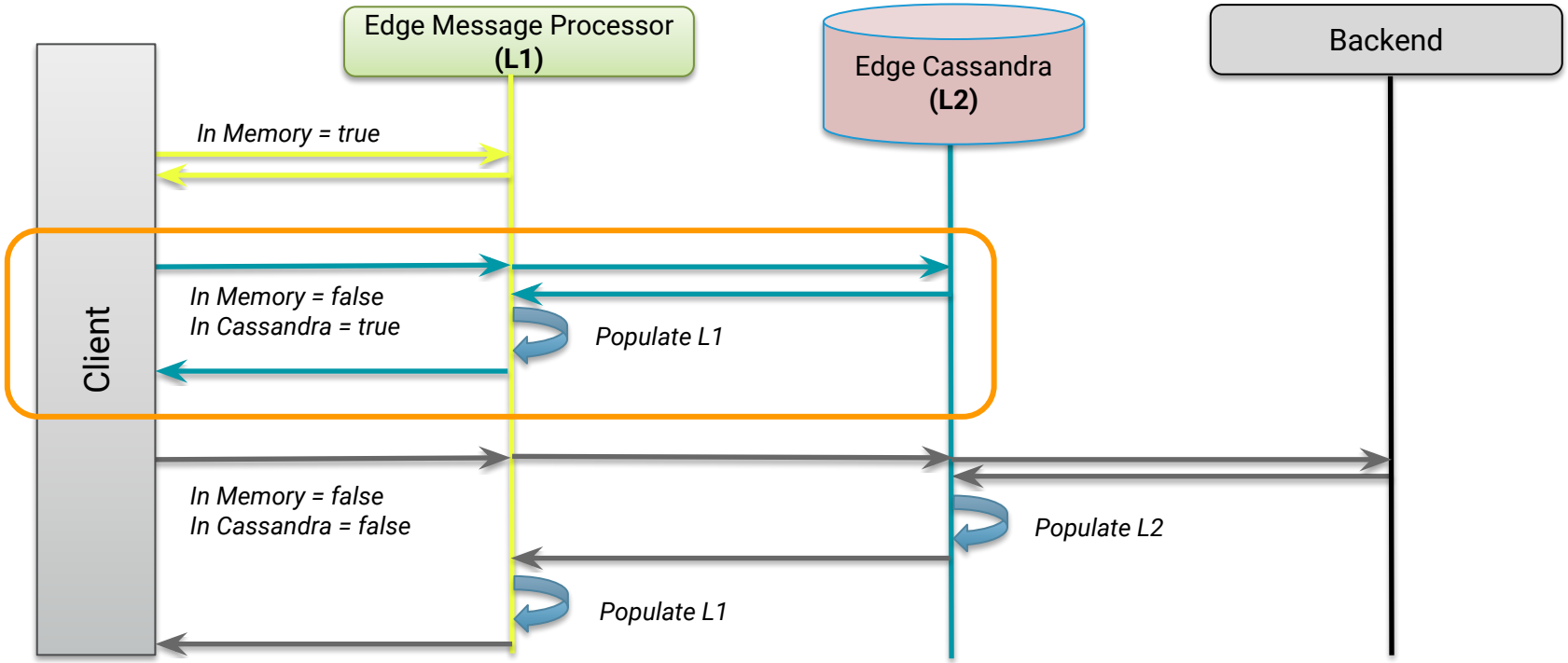
- Populate Cache/Lookup Cache
    - runtime persistence of data across requests
    - full control over caching, store any objects
    - configure time-to-live
    - add/update and read entries using separate policies

apigee

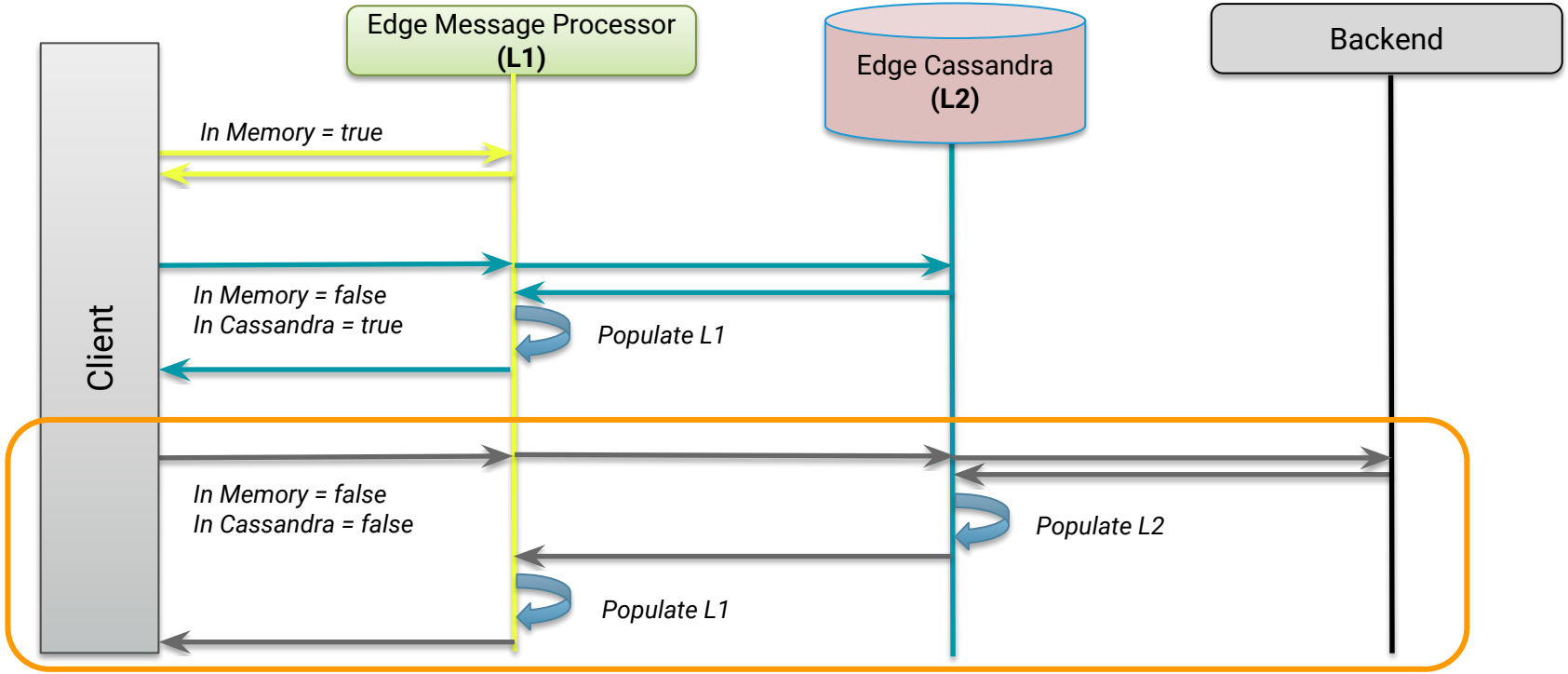
Cache Lookup/Populate Sequence



Cache Lookup/Populate Sequence



Cache Lookup/Populate Sequence



Response Cache

```
<ResponseCache async="false" continueOnError="false" enabled="true"
name="Response-Cache-For-Stores">
    <DisplayName>Response-Cache-For-Stores</DisplayName>
    <Properties/>
    <CacheResource>CacheStores</CacheResource>
    <CacheKey>
        <KeyFragment ref="request.queryparam.nearby" />
        <KeyFragment ref="request.queryparam.range" />
        <KeyFragment ref="request.queryparam.limit" />
        <KeyFragment ref="request.queryparam.offset" />
        <KeyFragment ref="request.queryparam.storetype"/>
        <KeyFragment ref="request.queryparam.locale" />
        <KeyFragment ref="storeId" />
    </CacheKey>
    <Scope>Application</Scope>
    <UseAcceptHeader>true</UseAcceptHeader>
    <UseResponseCacheHeaders>true</UseResponseCacheHeaders>
    <ExpirySettings>
        <TimeOfDay>10:00:00</TimeOfDay>
    </ExpirySettings>
    <SkipCacheLookup>(request.header.Skip-Cache Equals "true") or
(request.verb NotEquals "GET")</SkipCacheLookup>
    <SkipCachePopulation>(request.header.Skip-Cache Equals "true") or
(request.verb NotEquals "GET"))</SkipCachePopulation>
</ResponseCache>
```

Response Cache

```
<ResponseCache async="false" continueOnError="false" enabled="true"
name="Response-Cache-For-Stores">
    <DisplayName>Response-Cache-For-Stores</DisplayName>
    <Properties/>
    <CacheResource>CacheStores</CacheResource>
    <CacheKey>
        <KeyFragment ref="request.queryparam.nearby" />
        <KeyFragment ref="request.queryparam.range" />
        <KeyFragment ref="request.queryparam.limit" />
        <KeyFragment ref="request.queryparam.offset" />
        <KeyFragment ref="request.queryparam.storetype"/>
        <KeyFragment ref="request.queryparam.locale" />
        <KeyFragment ref="storeId" />
    </CacheKey>
    <Scope>Application</Scope>
    <UseAcceptHeader>true</UseAcceptHeader>
    <UseResponseCacheHeaders>true</UseResponseCacheHeaders>
    <ExpirySettings>
        <TimeOfDay>10:00:00</TimeOfDay>
    </ExpirySettings>
    <SkipCacheLookup>(request.header.Skip-Cache Equals "true") or
(request.verb NotEquals "GET")</SkipCacheLookup>
    <SkipCachePopulation>(request.header.Skip-Cache Equals "true") or
(request.verb NotEquals "GET"))</SkipCachePopulation>
</ResponseCache>
```

apigee

Response Cache

```
<ResponseCache async="false" continueOnError="false" enabled="true"
name="Response-Cache-For-Stores">
    <DisplayName>Response-Cache-For-Stores</DisplayName>
    <Properties/>
    <CacheResource>CacheStores</CacheResource>
    <CacheKey>
        <KeyFragment ref="request.queryparam.nearby" />
        <KeyFragment ref="request.queryparam.range" />
        <KeyFragment ref="request.queryparam.limit" />
        <KeyFragment ref="request.queryparam.offset" />
        <KeyFragment ref="request.queryparam.storetype"/>
        <KeyFragment ref="request.queryparam.locale" />
        <KeyFragment ref="storeId" />
    </CacheKey>
    <Scope>Application</Scope>
    <UseAcceptHeader>true</UseAcceptHeader>
    <UseResponseCacheHeaders>true</UseResponseCacheHeaders>
    <ExpirySettings>
        <TimeOfDay>10:00:00</TimeOfDay>
    </ExpirySettings>
    <SkipCacheLookup>(request.header.Skip-Cache Equals "true") or
(request.verb NotEquals "GET")</SkipCacheLookup>
    <SkipCachePopulation>(request.header.Skip-Cache Equals "true") or
(request.verb NotEquals "GET"))</SkipCachePopulation>
</ResponseCache>
```

Cache Name

Cache Key

Expiry Settings

Skip Cache

apigee

Which caching policy to use?

- Use Response Cache:
  - identical requests and responses
  - reduce unnecessary traffic to backend
  - reduce latency for common requests

Which caching policy to use?

- Use Response Cache:
  - identical requests and responses
  - reduce unnecessary traffic to backend
  - reduce latency for common requests
- Use Populate Cache / Lookup Cache:
  - storing custom data objects
  - to persist across multiple API transactions

Cache Utilization and Optimization

- use only when needed
- ensure your cache key is built correctly
- take advantage of the cache scope for re-use
- use the built-in Cache performance dashboard within Analytics

apigee

Clearing the Cache

- using the UI
- using the Management API
- using the InvalidateCache policy

apigee

Thank You

Google Cloud