Group 6: Manikanta, Srinivas, Karthik, Shilpa
What is an Event Store?

It’s a database which supports concept of event sourcing.
Event Store Features

- Open Source
- Event Sourcing
- Append only
- Projections
Event Store Features:

- Client Interfaces
- Multiplatform
- High Availability
How can be used?

• CQRS architectures
• Message queuing
• Storing events and notifications
• Auditing and archiving
Why?

Event Store
Old World

Relational Database Management Systems (RDBMS) like SQL Server

• Optimised for write, read *and* update
• Flexible, dynamic queries
• Indices help with returning data, but are often incorrectly defined, or aren’t even created
• Object-relational impedance mismatch
• Designed for client/server operation rather than HTTP/TCP based communication
New World

Document databases such as ... MongoDB, CouchDB, RavenDB and Event Store (strictly a streaming database)

• Store objects, as ... well objects, in JSON, BSON (Binary JSON-ish), XML
• Often include REST and HTTP interfaces for GET, POST, UPDATE, DELETE
• Metadata can be as important as the object data itself – not just versioning
Architecture with CRUD operations

- Model reads from database
- Service provides information for the presentation
- User makes a change in the UI
- Change forwarded to model
- Model executes validation, and consequential logic
- Model updates database
Different Read and Update Models:
Event Store Fundamentals

- CQRS
- Event Sourcing
CQRS

• Command Query Responsibility Segregation

• Pattern first heard from Greg Young

• Notion of using different model to update information

• Can be valuable

• Can add risky complexity
Event

- Event is something that happened in the past and has a business meaning.
- Events are persisted in event streams.
- Event stream is a time ordered sequence of events in time.
- Conceptually its append only.
Event Model

Consists of event data and some system data

- **eventId** - event identifier (could be generated by db)
- **eventType** - defines type of event
- **data** - custom event data
- **metadata** - event metadata
Sample Event

{
    "eventId": "8cfedd64-7e40-47ee-a16c-e57e2987783b",
    "eventType": "TemperatureMeasured",
    "data": {
        "zone": "Ireland",
        "server": "web1",
        "temperature": 64
    }
}
Event Metadata

$correlationId - application level correlation id

$causationId - application level causation id
Event Sourcing

• In traditional systems, we persist the current state of an object.

• In event sourced systems, we persist all changes that lead to the current state of an object.

• Every change is an immutable event
Object state is restored by replaying the entire stream of events

**Account Debited**
- number: 1536666
- amount: 2000.00
- date: 2015-01-18
- description: ATM

**Account Debited**
- number: 1536666
- amount: 14000.00
- date: 2015-01-16
- description: Salary

**Account Credited**
- number: 1536666
- amount: 50000.00
- date: 2015-01-12
- description: Salary

**Account Opened**
- number: 1536666
- openedDate: 2015-01-01

**Account**
- number: 1536666
- balance: 34000.00

**Account**
- number: 1536666
- balance: 36000.00

**Account**
- number: 1536666
- balance: 50000.00

**Account**
- number: 1536666
- balance: 0.00
Event Sourcing with CQRS
Types of Event Processing

- **Simple Event Processing**
  - events created for state changes or external occurrences, usually drive state machines
  - often all that is needed for enterprise applications

- **Event Stream Processing (ESP)**
  - filtering and processing of streams of events

- **Complex Event Processing (CEP)**
  - complex events are those derived from other events
  - CEP is the process of creating & processing complex events
Storage

Event stream

- Ordered sequence of events in time
- The partition point of the system
Stream category

- Streams could be categorized
- Category is resolved from stream name (after character ":")

[stream name]-[category]

Examples:

temperatures_by_zone-Ireland
temperatures_by_server-web1
Stream Metadata

$maxAge$ - maximum age of events in a stream

$maxCount$ - maximum number of events in a stream

$cacheControl$ - controls the cache of the head of a stream

$acl$ - access control list
Event Snapshots

- Summary of arbitrary amount of continuous past events
- Why we need Snapshots?
  - Aggregates lifetime
  - Current State changes
  - Large amount of Events
- Rebuilding aggregate state will have a performance impact
Data Model

CRUD: Create Read Update Delete (Relational Database Models)

Event Store: Append Only!! (NO Update) in CRUD
Basic Stream Operations

- Create
- Append
- Read
- Delete
- Subscriptions
Basic Stream Operations: Create

Implicit Creation

curl -i -d @event.txt "http://127.0.0.1:2113/streams/newstream" -H "Content-Type:application/json"

events.txt

```json
[
  {
    "eventId": "fbf4a1a1-b4a3-4dfe-a01f-ec52c34e16e4",
    "eventType": "event-type",
    "data": {
      "a": "1"
    }
  }
]
```
Basic Stream Operations: Append

- Single event write
- Batch write
Basic Stream Operations: Append

- Single event write
  - curl -i -d@myevent.txt "http://127.0.0.1:2113/streams/newstream" -H "Content-Type:application/json" -H "ES-EventType: SomeEvent" -H "ES-EventId:C322E299-CB73-4B47-97C5-5054F920746E"

- Batch write
  
  ```json
  {
    "something": "has data"
  }
  ```
ES : Events Media Types

Content-Type for Posting Events :

- application/(json/xml)
- application/vnd.eventstore.events(+json/+xml)
Basic Stream Operations: Append

- Single event write
- Batch write

```
curl -i -d@myevent.txt "http://127.0.0.1:2113/streams/newstream" -H "Content-Type: application/vnd.eventstore.events+json" -H "ES-EventType: SomeEvent" -H "ES-EventId: C322E299-CB73-4B47-97C5-5054F920746E"
```
Basic Stream Operations: Append

What happens when you post the same query repetitively to the ES?
Basic Stream Operations: Append

Are the writes Idempotent?
Basic Stream Operations: Append

EventID
Basic Stream Operations: Append

EventID

Client Side

Server Side
Basic Stream Operations: Append

EventID

Client Side

Server Side (redirect-to-idempotent-URI-Pattern)
Redirect-to-idempotent URI Pattern

Query:

curl -i -d @myevent.json "http://127.0.0.1:2113/streams/newstream" -H "Content-Type:application/json" -H "ES-EventType: SomeEvent" ---- NO EVENT ID
Redirect-to-idempotent URI Pattern

Query:
```
curl -i -d @myevent.json "http://127.0.0.1:2113/streams/newstream" -H "Content-Type:application/json" -H "ES-EventType: SomeEvent"  
```

Response:
```
HTTP/1.1 301 FOUND
Access-Control-Allow-Methods: POST, DELETE, GET, OPTIONS
Access-Control-Allow-Headers: Content-Type, X-Requested-With, X-PINGOTHER, Authorization, 
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Location, ES-Position
Location: http://127.0.0.1:2113/streams/newstream/incoming/c7248fc1-3db4-42c1-96aa-a071c926
Content-Type: ; charset=utf-8
Server: Mono-HTTPAPI/1.0
Date: Mon, 21 Apr 2014 21:11:59 GMT
Content-Length: 28
Keep-Alive: timeout=15 max=100
```
Redirect-to-idempotent URI Pattern

New Query:  
curl -i -d @myevent.json  
"http://127.0.0.1:2113/streams/newstream/incoming/c7248fc1-3db4-42c1-96aa-a071c92649d1"  
-H "Content-Type: application/json" -H "ES-EventType: SomeEvent"

Response:

HTTP/1.1 201 Created
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: Content-Type, X-Requested-With, X-PINGOTHER, Authorization,
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Location, ES-Position
Location: http://127.0.0.1:2113/streams/newstream/0
Content-Type: text/plain; charset=utf-8
Server: Mono-HTTPAPI/1.0
Date: Mon, 21 Apr 2014 21:14:28 GMT
Content-Length: 0
Keep-Alive: timeout=15,max=100
Basic Stream Operations : Append

- EventID
  - Client Side
  - Server Side
    - (redirect-to-idempotent-URI-Pattern)
Basic Stream Operations: Read

- All streams are exposed as atom feeds.

Accepted Content Types for GET are:
  - application/xml
  - application/atom+xml
  - application/json
  - application/vnd.eventstore.atom+json
  - text/xml
  - text/html
Deleting a Stream

**Soft delete** - stream could be recreated later

**Hard delete** - stream couldn't be recreated later
Deleting a Stream

Soft delete:

Hard delete

Using http DELETE Method

curl -v -X DELETE http://127.0.0.1:2113/streams/foo
Deleting a Stream

Using ES header attribute & http DELETE Method

**Soft delete**


**Hard delete**:

Scavenging: Disk space retention
Subscriptions

Live Only

Catch Up

From this point onwards.
Subscriptions

Live Only

Catch Up

From any point onwards (position passed as argument).
Projections

The process of taking an event Stream/s and converting it to some other form (event state / stream)
Projections

Indexing: Build state, emit new events or link to existing events

Temporal Queries: Concept of continuous queries
Projections : Functional Principles

Transform(f3(f2(f1(initial(), e1), e2), e3)

f(state, event) => state f is run over the series of events

transform(state) => result transform can transform the state to the form of result you want to receive

initial() => state initial returns the initial state
Projections: Event Selection

fromAll: $any

fromStream: select all events from a specific stream

fromStreams*: select all events from all categories from all streams

fromCategory: selecting streams from categories of many streams (subset)
Projections : Event Matching

Custom Event Matchers :

$init

$any

when([

[SomePatternMatch]: function(state, event) {
  return new state;
},

[OtherPatternMatch]: function(state, event) {
  return new state;
}
])
### Projections: Event Indexing

**Before Indexing:**

<table>
<thead>
<tr>
<th>Stream : Chat 1</th>
<th>Stream : Chat 2</th>
<th>Stream : Chat 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greg : hi</td>
<td>John : yo</td>
<td>Jill : anyone there?</td>
</tr>
<tr>
<td>John : Hey Greg</td>
<td>Jill : donuts!</td>
<td>Greg : sure</td>
</tr>
<tr>
<td>Stream : Chat 1</td>
<td>Stream : Chat 2</td>
<td>Stream : Chat 3</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Greg : hi</td>
<td>John : yo</td>
<td>Jill : anyone there?</td>
</tr>
<tr>
<td>John : Hey Greg</td>
<td>Jill: donuts!</td>
<td>Greg : sure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stream : Greg</th>
<th>Stream : John</th>
<th>Stream : Jill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chat1 : hi</td>
<td>Chat2 : yo</td>
<td>Chat3: anyone there?</td>
</tr>
<tr>
<td>Chat3 : sure</td>
<td>Chat2 : Hey Greg</td>
<td>Chat2 : donuts!</td>
</tr>
</tbody>
</table>
Projections: Internal Indexing

‘UseEventIndices’ - Indexing based on $et<eventtype>
Replication

2 Quorums used - Read, Write

No quorum yet - Paxos Election!

Client Retries if transaction fails.
Security

1. Internal authentication: Using stream’s Access Control List

Example: 

```
{ "acl": { 
  "$w": "greg",
  "$r": ["greg", "john"],
  "$d": "$admins",
  "$mw": "$admins",
  "$mr": "$admins"
}
```

Mw: write for metadata

Mr: read for metadata
Security Cont..

2. External authentication: Use reverse Proxy servers
Security Cont..

3. Hybrid Option - trusted intermediary header

Figure 9: Message forwarding
Communication with ES

TCP
- Push events to subscribers
- Suggested for high-performance environment

HTTP
- Subscribers pool to check events availability
- AtomPub Interface
- Intermediary caching of Atom feeds
<table>
<thead>
<tr>
<th></th>
<th>HTTP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalability</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Network traffic</strong></td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td><strong>Time for a transaction</strong></td>
<td>1 second</td>
<td>10 ms</td>
</tr>
<tr>
<td><strong>Write/sec</strong></td>
<td>2000</td>
<td>15,000-20,000</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>Heterogeneous</td>
<td>Homogeneous</td>
</tr>
</tbody>
</table>
Use cases

- Audit log [ who, when ]
- BI applications: Fraud detection - incorrect CVV (4 attempts)
- Complex historical analysis of data
Drawbacks and Limitations

Every database on a planet sucks. And they all suck it their own unique original ways.

Greg Young, Polyglot Data talk

- Complex - Not ready to learn technologies
- Eventual consistency
Questions?