Hierarchical Data Storage And Processing on the Edge of the Network

Seyed Hossein Mortazavi
mortazavi@cs.toronto.edu
Cloud Computing

• Enables ubiquitous, and convenient access to a shared pool of configurable computing resources
  • Servers
  • Storage
  • Applications
  • Services
  • Network
Next Generation Apps

• Next generation applications require lower latencies, better response time
  • Virtual/Augmented Reality, wearable devices, safety-critical applications, etc.

• They need high bandwidth
  • Video analysis, Internet of Things(IoT) devices

• Other requirements: privacy, exact location detection, scalability,
Edge Computing

Bring the cloud closer to the users (Mobile Edge Computing, Cloudlets, Fog Computing)

- Computation and Storage units are placed near the user
- On the “Edge” of the network
Web Applications on the cloud

- Collection of independent stateless handlers
  - REST (flexibility)
  - Scalability
  - Fault tolerance

- What Enables this model?
  - Shared storage layer
CloudPath

- A platform that enables the execution of 3rd party applications on a progression of datacenters deployed along the geographical span of the network

- Separation between application code and data
  - Developers: organize applications as a collection of stateless functions
  - CloudPath: on-demand replication of code and data

- Provides a common runtime on all cloud nodes
Each node consists of:

- Common execution environment --> **PathExecute**
- On-Demand deployment --> **PathDeploy**
- Routing --> **PathRoute**
- Common storage layer --> **PathStore**
- Monitoring --> **PathMonitor**
Example

F1: Record flight info

F2: Get sensory data, make decisions on flight path

F3: Get frame, do object avoidance
Pathstore: A Distributed Storage Layer For The Edge

- PathStore is a data storage layer for a multi-tier cloud architecture

- Hierarchy of data stores
  - Each PathStore node runs an independent Cassandra ring and PathStore copies data between rings

- Support Eventual consistency
  - Extension called SessionStore supports session consistency
PathStore Design

- Data store at the root of the hierarchy is persistent
- Other act as temporary partial replicas
  - Data replicated on-demand
  - All reads and writes executed against local replica
- CQL based interface
- Row-level replication
Example

```
select * from marbles
select * from marbles where color = blue
select * from marbles
select * from marbles where color = green
select * from marbles
select * from marbles where color = blue
select * from marbles
select * from marbles where color = green
```
SessionStore

- SessionStore ensures session consistency on a top of otherwise eventually consistent replicas.

- Enforces session consistency by grouping related data accesses into a session, and using a session-aware reconciliation algorithm to reconcile only the data that is relevant to the session when switching between data centers.
  - Only replicates relevant data (row level)
  - Only replicates to the relevant destination
SessionStore

Switching: Transfer data for neighbors

 asserted

select * from marbles

where color=blue

Insert green_marble into marbles
Session Tracking

- We identify each session using a Session Token, or stoken.
- Stoken consists of four fields:
  - A unique session id (SID)
  - Timestamp
  - Current replica
  - Status
- The stoken is encrypted and signed to prevent forging and misrepresentation.

- To keep track of data related to a session, a command-Cache is added to each PathStore replica that stores all the CQL SELECT statements that were run on behalf of a session.
Network Requirements

- Routing requests for applications based on functions
- Synchronizing clocks between Datacenters
- Providing locking service for data
  - Better guarantees for data
Questions

Contact: mortazavi@cs.toronto.edu