Towards a Democratic Federation for Infrastructure Service Provisioning

Bishakh Chandra Ghosh (goshbishakh@iitkgp.ac.in), Sourav Kanti Addya, Anurag Satpathy, Soumya K Ghosh and Sandip Chakraborty

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Cloud Federations

Collaboration among different **Cloud Service Providers** (CSPs), whereby they agree to mutually share their own resources for their overall benefit.
Cloud Federations

Collaboration among different **Cloud Service Providers** (CSPs), whereby they agree to mutually share their own resources for their overall benefit.
Motivation for cloud federation

- **Sharing** of computing resources.
- **Aggregation of unused resources** from different service providers.
- Bringing services closer to customers by maximizing the **geographical dispersion**.
- **Tackling data protection laws** that requires data to be stored within country’s boundary.
Existing cloud federations

Mostly **Centralized** Approach:

1. Centralized cloud broker
Existing cloud federations

Mostly **Centralized** Approach:

2. Centralized cloud exchange
Limitations of existing federations:

1. Profit sharing with central broker
2. Biasness of broker towards certain service providers
3. Price manipulation (*Broker can be malicious*)
4. Unfair dispute resolution
5. Central point of failure
Objective

Remove the central broker and design a transparent distributed system for cloud federation.
Centralized to Decentralized

Centralized Cloud federation

User Requests

Broker

Centralized layer

Decentralized Cloud federation

No Centralized layer
# Some related works

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>S. Kirkman</td>
<td>Data movement policy framework using smart contracts</td>
<td>[1]</td>
</tr>
<tr>
<td>Sukhodolskiy et al.</td>
<td>Blockchain-based access control system for cloud storage</td>
<td>[2]</td>
</tr>
<tr>
<td>Zhou et al.</td>
<td>Blockchain based witness model for SLA monitoring</td>
<td>[3]</td>
</tr>
<tr>
<td>Margheri et al.</td>
<td>Distributed infrastructure for democratic cloud federations</td>
<td>[4]</td>
</tr>
</tbody>
</table>

Challenges

● A decentralized platform for exchange of infrastructure resources (VM) must be developed.

● The system must allow coordination between service providers while enforcing FLA, without the help of any broker.

● Cloud functions such as VM Placement and VM Migration needs to be coordinated over the decentralized architecture.

● Fair ordering of transactions must be ensured
Proposed Architecture
Proposed Architecture
Proposed Architecture
Proposed Architecture

Permissioned Blockchain
Proposed Architecture

Permissioned blockchain based decentralized exchange for democratic cloud federations: CloudChain
Proposed Architecture

For simplicity we assume that the federation contains two types of service providers namely,

1. **Demanding** service providers:
   
   Suffer from *resource limitations* and require other members of the federation to create instances for them at peak loads.

2. **Supplying** service providers:
   
   Having *abundant resources* which goes unused.
Proposed Architecture

CloudChain

Components of CloudChain

CloudChain model overview
Components

1) *CloudChain* Blockchain: Distributed Ledger & Exchange State

2) Request Queue and Resource Bucket

3) Scheduler

4) Transaction Manager

5) VM Manager
Components

Request Queue (ReQ): queue of incoming multi-tier web application requests

Resource Bucket:

Bucket of available resources, which may include both local resources and exchange resources.

1) Local resource bucket (ResBlocal)

2) Exchange resource bucket (ResBexchange)
Components

Scheduler: Coordinates all the components of CloudChain

Transaction Manager: An interface to the CloudChain Blockchain.

VM Manager: Manages creation, deletion and access to VMs.
**CloudChain Blockchain** serves as an information registry that maintains the current state of available resources and demand patterns. Thus it acts as a common marketplace where different CSPs can offer their unused or excess resources for outsourcing, and rent resources from other CSPs when required.

<table>
<thead>
<tr>
<th>Offerings</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Supplying SP</td>
<td>Specs</td>
<td>Price/BTU</td>
</tr>
<tr>
<td>o₁</td>
<td>Cloud 1</td>
<td>2 Cores, 4GB, UK</td>
<td>15$/Month</td>
</tr>
<tr>
<td>o₂</td>
<td>Cloud 2</td>
<td>2 Cores, 8GB, US</td>
<td>18$/Month</td>
</tr>
<tr>
<td>o₃</td>
<td>Cloud 1</td>
<td>2 Cores, 2GB, UK</td>
<td>0.16$/Day</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requests</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Demanding SP</td>
<td>Offering Id</td>
<td>Duration</td>
</tr>
<tr>
<td>r₁</td>
<td>Cloud 2</td>
<td>o₁</td>
<td>6 Months</td>
</tr>
<tr>
<td>r₂</td>
<td>Cloud 3</td>
<td>o₂</td>
<td>12 Months</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Associations</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Demanding SP</td>
<td>Offering Id</td>
<td>Duration</td>
</tr>
<tr>
<td>a₁</td>
<td>Cloud 3</td>
<td>o₅</td>
<td>3 Months</td>
</tr>
<tr>
<td>a₂</td>
<td>Cloud 5</td>
<td>o₄</td>
<td>7 Days</td>
</tr>
</tbody>
</table>
CloudChain Blockchain

The high level operations that the CSPs can perform on the exchange are:

1) Offer a new resource
2) Modify an existing offering
3) Query for available resources offerings
4) Request to rent a resource
5) Grant/Reject a request
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request $R_{user}$

Local resource available

Yes

Choose best offering

$O_{opt}$

Transaction Manager

Execute request offering transaction

Remote request

VM Manager

Local placement

Update blockchain

Request rejected

Request $T_{req}$

CloudChain blockchain

VM requests from remote CSPs

Send for remote placement

Local request accepted
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request

Yes

Choose best offering

No

Remote request

VM Manager

Local placement

Update blockchain

Request accepted

Request rejected

CloudChain blockchain

Transaction Manager

Execute request offering transaction

Request $\tau_{req}$

$\mathcal{R}_{user}$

Local resource available

Yes

VM requests from remote CSPs

Send for remote placement

Request $\mathcal{O}_{opt}$
System Flow

Multi-tier application requests from users

\( R_{user} \)

Type of request

Local request available

Yes

No

Choose best offering

\( O_{opt} \)

Transaction Manager

Execute request offering transaction

VM Manager

Local placement

Update blockchain

Request rejected

\( \mathcal{T}_{req} \)

CloudChain blockchain

Request accepted

Send for remote placement

Remote request

Local request

VM requests from remote CSPs
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request

Local resource available

Yes

Choose best offering

Remote request

VM Manager

Local placement

Update blockchain

VM requests from remote CSPs

Send for remote placement

CloudChain blockchain

Transaction Manager

Execute request offering transaction

Request rejected

Request accepted

Yes

No
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request $R_{user}$

Local resource available

Yes

Choose best offering

No

Transaction Manager

Execute request offering transaction

VM Manager

Local placement

Update blockchain

Request $\mathcal{T}_{req}$

Request rejected

CloudChain blockchain

Update blockchain

Request accepted

Send for remote placement

VM requests: from remote CSPs:

Remote request
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local resource available

Local request

Yes

Choose best offering

Remote request

No

Transaction Manager

VM Manager

Local placement

Update blockchain

Request rejected

Request \( \mathcal{T}_{req} \)

Execute request offering transaction

VM requests from remote CSPs

Send for remote placement

CloudChain blockchain

Request accepted

Request \( \mathcal{R}_{user} \)
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request \( R_{user} \)

Local resource available

Yes

Choose best offering

\( O_{opt} \)

Transaction Manager

Execute request offering transaction

Request rejected

Request \( T_{req} \)

CloudChain blockchain

Update blockchain

Local placement

VM Manager

Remote request

Send for remote placement

VM requests from remote CSPs

No
System Flow

Multi-tier application requests from users

Local request $\mathcal{R}_{user}$

Type of request

Local resource available

Yes

Choose best offering

No

Transaction Manager

Execute request offering transaction

VM Manager

Local placement

Update blockchain

CloudChain blockchain

Request accepted

Send for remote placement

Request rejected

VM requests from remote CSPs
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request $R_{\text{user}}$

Local resource available

Yes

Choose best offering $O_{\text{opt}}$

No

Transaction Manager

Execute request offering transaction

VM Manager

Local placement

Update blockchain

Request accepted

CloudChain blockchain

Request rejected

Send for remote placement

VM requests from remote CSPs
System Flow

Multi-tier application requests from users

ReQ

Type of request

Local request $\mathcal{R}_{user}$

Local resource available

Yes

Choose best offering

$\mathcal{O}_{opt}$

Transaction Manager

Execute request offering transaction

Remote request

VM Manager

Local placement

Update blockchain

Request accepted

Request rejected

CloudChain blockchain

Send for remote placement

VM requests from remote CSPs
System Flow

Multi-tier application requests from users

- ReQ

Type of request

- Local request
  - \( R_{user} \)
  - Local resource available
    - Yes
    - Choose best offering
      - \( O_{opt} \)
      - Transaction Manager
      - Execute request offering transaction
        - Request rejected
        - \( T_{req} \)
  - No
  - Remote request
    - VM Manager
      - Local placement
        - Update blockchain
        - Request accepted
        - CloudChain blockchain
        - Send for remote placement
          - VM requests from remote CSPs
System Flow

Multi-tier application requests from users

$\mathcal{R}_{user}$

Type of request

Local request available

Yes

No

Choose best offering

Transaction Manager

Execute request offering transaction

VM Manager

VM requests from remote CSPs

Send for remote placement

Request accepted

Request rejected

Local placement

Update blockchain

CloudChain blockchain

Local request

Remote request

Yes

No

Choose best offering

$O_{opt}$
Implementation Details

Testbed Setup:

- 3 Hosts, each acting as a cloud connected over the network.

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of CSP</td>
<td>3</td>
</tr>
<tr>
<td>Number of DC per CSP</td>
<td>3</td>
</tr>
<tr>
<td>$C_1$ Config</td>
<td>2.7 GHz, Intel Xeon(R) 48 core, 256 GB Memory</td>
</tr>
<tr>
<td>$C_2$ Config</td>
<td>3.2 GHz, Intel Core i5 4 core, 20 GB Memory</td>
</tr>
<tr>
<td>$C_3$ Config</td>
<td>2.7 GHz, Intel Core i3 4 core, 8 GB Memory</td>
</tr>
<tr>
<td>Containerization</td>
<td>Docker 18.06</td>
</tr>
<tr>
<td>Language used</td>
<td>Go 1.10, Python 2.7</td>
</tr>
</tbody>
</table>
Implementation Details

Testbed Setup:

- 3 Hosts, each acting as a cloud connected over the network.

- Hyperledger fabric for blockchain (v1.3.0)
Implementation Details

Testbed Setup:

- 3 Hosts, each acting as a cloud connected over the network.
- Hyperledger fabric for blockchain (v1.3.0).
- Each cloud belongs to a separate organization, and runs a peer.
Implementation Details

Testbed Setup:

- Each cloud runs its own `orderer`.
Implementation Details

Testbed Setup:

● Each cloud runs its own orderer

● Create a **docker swarm**.

● Create **overlay network**.
Implementation Details

Testbed Setup:

- Each cloud runs its own orderer.
- Create a **docker swarm**.
- Create **overlay network**.
- **Chaincodes** for *CloudChain* logic.
Implementation Details

Testbed Setup:

- Each cloud runs its own orderer.
- The orderers use BFT protocol.
- **Chaincodes** for Cloud Exchange logic.
- **Endorsement policy** requiring the endorsement of the concerned demanding SP, supplying SP and the majority of other endorsing peers.
Results

- Mean VM placement time in broker based federation and *CloudChain*
- Three scenarios
- Each CSP receives 4, 6, and 10 VM requests in first, second and third scenarios respectively.
- In case of broker based federation all the requests arrive at the broker first.

Very little compromise in performance
Results

- 34 multi-tier application requests
- $C1$, $C2$ and $C3$ receiving 16, 8 and 10 requests respectively.
- In case of broker based federation, all requests arrive at the broker.

CloudChain shows fair distribution.
Conclusion and Future work

*CloudChain* over Federation brokers:

- Decentralized
- Transparent
- Autonomy
- Immutable
- Fairness
Conclusion and Future work

Future work

- Support for live VM migration
- FLA monitoring
Acknowledgements

Special thanks to:

Microsoft Research Travel Grant.

CNeRG for Travel Grant and constant support. (cnerg.iitkgp.ac.in)
Thank You

Bishakh Chandra Ghosh (ghoshbishakh@iitkgp.ac.in), Sourav Kanti Addya, Anurag Satpathy, Soumya K Ghosh and Sandip Chakraborty