Parallel and Concurrent Programming: Fall 2023

Project 1: Design and implementation of distributed node architecture for Hyperledger Sawtooth blockchain

Team members
Souvik Sarkar(CS23MTECH02001)

Report

Goal:

Our main goal in this research is to create a secure, distributed, and effective framework for executing smart contracts in block chain systems, which will improve the scalability and fault tolerance of individual Sawtooth nodes. Our strategy focuses on improving each peer's operation inside the network to remove performance bottlenecks and fully utilise the Sawtooth framework for a range of applications.

Introduction:

The strong enterprise blockchain platform known as Hyperledger Sawtooth is intended to make it easier for distributed ledger networks and applications to be developed. Hyperledger Sawtooth has become an enterprise's first choice when looking to leverage blockchain technology, mostly because of its emphasis on maintaining the decentralised character of ledgers and guaranteeing the security of smart contracts.

The platform separates the application domain from the core system by implementing a distinct design philosophy. With this method, creating blockchain applications is made easier and developers may use the programming languages of their choice to express business rules that are specific to their applications. Notably, an in-depth knowledge of the minute features of the underlying core system is not necessary for this autonomy.

The tremendous modularity of Hyperledger Sawtooth is one of its main advantages. Because of the platform's modular design, corporations and consortia can match the platform to their own requirements and capabilities and make informed policy decisions. Sawtooth's modular design provides applications with the flexibility to select the permissioning methods, consensus algorithms, and transaction rules that best fit their unique business needs.

Essentially, Hyperledger Sawtooth is a flexible and adaptive blockchain system that promotes innovation by giving developers the instruments they need to precisely and efficiently construct decentralised applications. Sawtooth enables businesses to traverse the complicated world of blockchain technology while customising solutions to address their particular business concerns because to its dedication to modularity and adaptability.

Technologies Used:

To make our Sawtooth node distributed we used mainly **Docker** and **Kubernetes** -

Introduction to Docker:

Docker is a transformative containerization platform that simplifies software development and deployment. By encapsulating applications and their dependencies into lightweight, portable containers, Docker ensures consistency across various environments. Developers can build, ship, and run applications seamlessly, fostering a modular and scalable approach. This technology enables microservices architecture, supports continuous integration/deployment, and accelerates software delivery by providing a standardized and efficient development environment. Docker has become a fundamental tool, promoting agility and innovation in modern software engineering.

Introduction to Kubernetes:

Kubernetes is an open-source platform automating the deployment and management of containerized applications. It streamlines tasks like scaling and updates, making applications portable across different environments. With a focus on resilience and scalability, Kubernetes has become the standard for container orchestration, simplifying the development and deployment of modern, cloud-native applications.

Kubernetes and Docker serve distinct roles in the container ecosystem. Docker is a platform for containerizing applications, while Kubernetes is an orchestration platform for automating deployment and management. Kubernetes excels in orchestrating complex, multi-container applications, offers advanced features, and has a larger ecosystem and community support. Docker is user-friendly for containerization and widely used for building and running containers. The choice depends on the specific needs of your applications, and often they are used together, with Docker providing the container runtime and Kubernetes managing orchestration.

Existing Sawtooth Node Architecture :

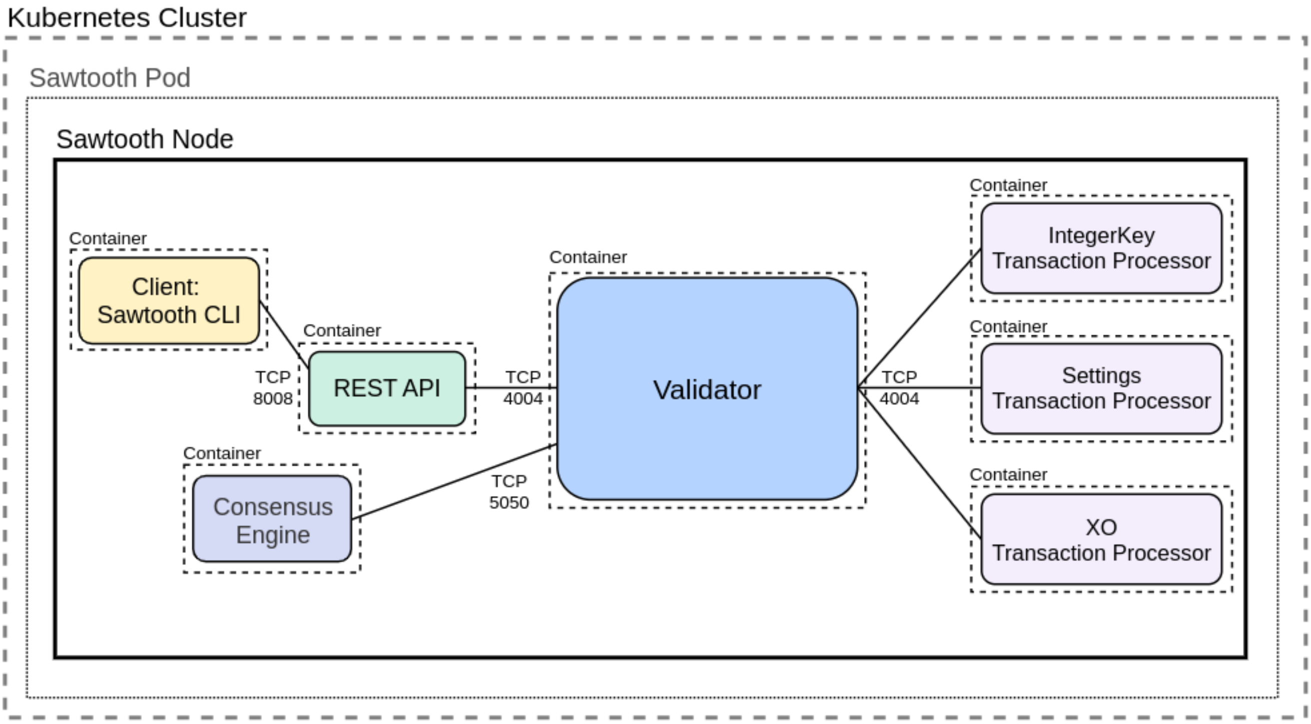

Figure 1: Sawtooth Node

In this Kubernetes setup, a lone Sawtooth node is active, hosting a validator, a REST API, the Devmode consensus engine, and three transaction processors. Utilizing the Devmode consensus and parallel transaction processing, this configuration showcases a complete deployment structure for Sawtooth.

Drawback:

In this Kubernetes setup, only one node is active, hosting a validator, a REST API, the Devmode consensus engine, and three transaction processors. But failiure of a single module cause whole node to crash and this node will lost its representation in the network.

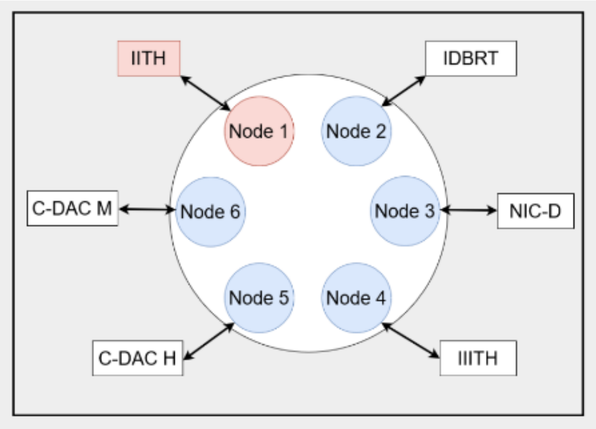

Figure 2: Distributed sawtooth node

Kubernetes YAML Configuration

[!h]

```yaml
---
apiVersion: v1
kind: List

items:

- apiVersion: apps/v1 #extensions/v1beta1
  kind: Deployment
  metadata:
    name: sawtooth-0-module2-pod
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: sawtooth-0-module2

    template:
      metadata:
        labels:
          app: sawtooth-0-module2
      spec:
        containers:
          - name: sawtooth-settings-tp
            image: hyperledger/sawtooth-settings-tp:nightly
            command:
              - bash
            args:
              - -c
              - "settings-tp -vv -C tcp://10.244.0.108:4004" #- "settings-tp -vv -C tcp://$HOSTNAME:4004"

          - name: sawtooth-intkey-tp-python
            image: hyperledger/sawtooth-intkey-tp-python:nightly
            command:
              - bash
            args:
              - -c
              - "intkey-tp-python -vv -C tcp://10.244.0.108:4004"

          - name: sawtooth-xo-tp-python
            image: hyperledger/sawtooth-xo-tp-python:nightly
            command:
              - bash
            args:
              - -c
              - "xo-tp-python -vv -C tcp://10.244.0.108:4004"
```

Listing 1: Your Kubernetes YAML code

Distributed sawtooth node :

Our distributed framework's main objective is to improve fault tolerance and scalability in the context of blockchain nodes.The blockchain's overall resilience against malfunctioning nodes is contrasted with the vulnerability of an individual node.This vulnerability is especially noticeable in permissioned configurations, as the loss of even a single node can result in a significant reduction in the entity's representation. Figure 4 shows how to exploit this weakness. When a distributed node is set up, a cooperative group of systems works together to perform the tasks of a single blockchain node.

Our Work :

In our implementation we basically divide the whole Sawtooth node into two pods. One module haveing all the validation modules like validator,REST-Api, and Consensus engine and Client modules. And another pod having the execution modules like all the Transaction processors.

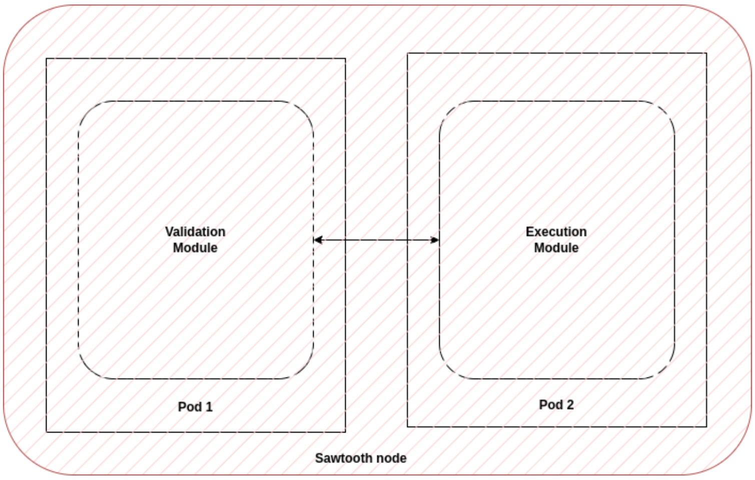

Figure 3: Blockchain network

Kubernetes YAML Configuration : Validation module

```
apiVersion: v1
kind: List

items:

- apiVersion: apps/v1 #extensions/v1beta1
  kind: Deployment
  metadata:
    name: sawtooth-0
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: sawtooth-0

    template:
      metadata:
        labels:
          app: sawtooth-0
      spec:
        containers:
```

```
        - name: sawtooth-devmode-engine
          image: hyperledger/sawtooth-devmode-engine-rust:nightly
          command:
            - bash
          args:
            - -c
            - "devmode-engine-rust -C tcp://$HOSTNAME:5050"

        # - name: sawtooth-settings-tp
        #   image: hyperledger/sawtooth-settings-tp:nightly
        #   command:
        #     - bash
        #   args:
        #     - -c
        #     - "settings-tp -vv -C tcp://$HOSTNAME:4004"

        # - name: sawtooth-intkey-tp-python
        #   image: hyperledger/sawtooth-intkey-tp-python:nightly
        #   command:
        #     - bash
        #   args:
        #     - -c
        #     - "intkey-tp-python -vv -C tcp://$HOSTNAME:4004"

        # - name: sawtooth-xo-tp-python
        #   image: hyperledger/sawtooth-xo-tp-python:nightly
        #   command:
        #     - bash
        #   args:
        #     - -c
        #     - "xo-tp-python -vv -C tcp://$HOSTNAME:4004"

        - name: sawtooth-validator
          image: hyperledger/sawtooth-validator:nightly
          ports:
            - name: tp
              containerPort: 4004
            - name: consensus
              containerPort: 5050
            - name: validators
              containerPort: 8800
          command:
            - bash
          args:
            - -c
            - "sawadm keygen \
            && sawtooth keygen my_key \
            && sawset genesis -k /root/.sawtooth/keys/my_key.priv \
            && sawset proposal create \
              -k /root/.sawtooth/keys/my_key.priv \
              sawtooth.consensus.algorithm.name=Devmode \
              sawtooth.consensus.algorithm.version=0.1 \
              -o config.batch \
            && sawadm genesis config-genesis.batch config.batch \
            && sawtooth-validator -vv \
                --endpoint tcp://$SAWTOOTH_0_SERVICE_HOST:8800 \
                --bind component:tcp://eth0:4004 \
                --bind consensus:tcp://eth0:5050 \
                --bind network:tcp://eth0:8800"

        - name: sawtooth-rest-api
          image: hyperledger/sawtooth-rest-api:nightly
          ports:
            - name: api
              containerPort: 8008
          command:
              - bash
            args:
              - -c
              - "sawtooth-rest-api -C tcp://$HOSTNAME:4004"

          - name: sawtooth-shell
            image: hyperledger/sawtooth-shell:nightly
            command:
              - bash
            args:
              - -c
              - "sawtooth keygen && tail -f /dev/null"

- apiVersion: v1
  kind: Service
  metadata:
    name: sawtooth-0
  spec:
    type: ClusterIP
    selector:
      name: sawtooth-0
    ports:
      - name: "4004"
        protocol: TCP
        port: 4004
        targetPort: 4004
      - name: "5050"
        protocol: TCP
        port: 5050
        targetPort: 5050
      - name: "8008"
        protocol: TCP
        port: 8008
        targetPort: 8008
      - name: "8800"
        protocol: TCP
        port: 8800
        targetPort: 8800
```

Listing 2: Validation module

Kubernetes YAML Configuration : Execution module

```
---
apiVersion: v1
kind: List

items:

- apiVersion: apps/v1 #extensions/v1beta1
  kind: Deployment
  metadata:
    name: sawtooth-0-module2-pod
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: sawtooth-0-module2

    template:
      metadata:
        labels:
          app: sawtooth-0-module2
      spec:
        containers:
          - name: sawtooth-settings-tp
            image: hyperledger/sawtooth-settings-tp:nightly
            command:
              - bash
            args:
              - -c
              - "settings-tp -vv -C tcp://10.244.0.108:4004" #- "settings-tp -vv -C tcp://$HOSTNAME:4004"

          - name: sawtooth-intkey-tp-python
            image: hyperledger/sawtooth-intkey-tp-python:nightly
            command:
              - bash
            args:
              - -c
              - "intkey-tp-python -vv -C tcp://10.244.0.108:4004"

          - name: sawtooth-xo-tp-python
            image: hyperledger/sawtooth-xo-tp-python:nightly
            command:
              - bash
            args:
              - -c
              - "xo-tp-python -vv -C tcp://10.244.0.108:4004"
```

Listing 3: Execution module

Next step :

The first stage is to group each module into a distinct pod.Creating a distributed framework to get around the constraints found in the load testing experiments is the second step. The distributed node architecture has the potential to significantly improve cluster systems' scalability, performance, and reliability, allowing them to adapt to the changing needs of large-scale data processing and analysis across a range of industries. In order to reduce network latency and increase data throughput, we are now investigating consensus protocols and data distribution strategies.