Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

Computer Applications

Stream Analytics

Deploying and Monitoring Apps

Deployment Modes

Niteesh K R
Computer Applications

Stream Analytics

Deploying Flink Apps

- Flink offers various deployment modes to cater to different requirements of distributed and local environments

  Modes

  - Standalone Cluster – Running the script file / Java file using `flink run`
  - YARN Cluster – High Availability Setup with Hadoop and MapReduce
  - Kubernetes Cluster – Highly Available
  - Docker – Efficient usage of system resources

Stream Analytics

Local Deployment – Python

- Suitable for development, testing, and debugging purposes
- Run a PyFlink application locally using the `StreamExecutionEnvironment`
- Running as a Python Script
- Can also be done using `pyflink-shell.sh` of Flink/bin

Stream Analytics

Standalone Cluster

- Used for running PyFlink applications in distributed environments
- Start a Flink Cluster using `start-cluster.sh` in the `bin` directory of Flink installation
- This starts a Flink cluster, with Web UI localhost:8081
- Run the Flink job using `flink run`

  `./bin/flink run -py flink_job.py`
- Note – `flink run` can be found inside the `bin` directory of Flink installation directory and has to be executed only after starting Flink cluster

Stream Analytics

Flink Web UI

After executing a job

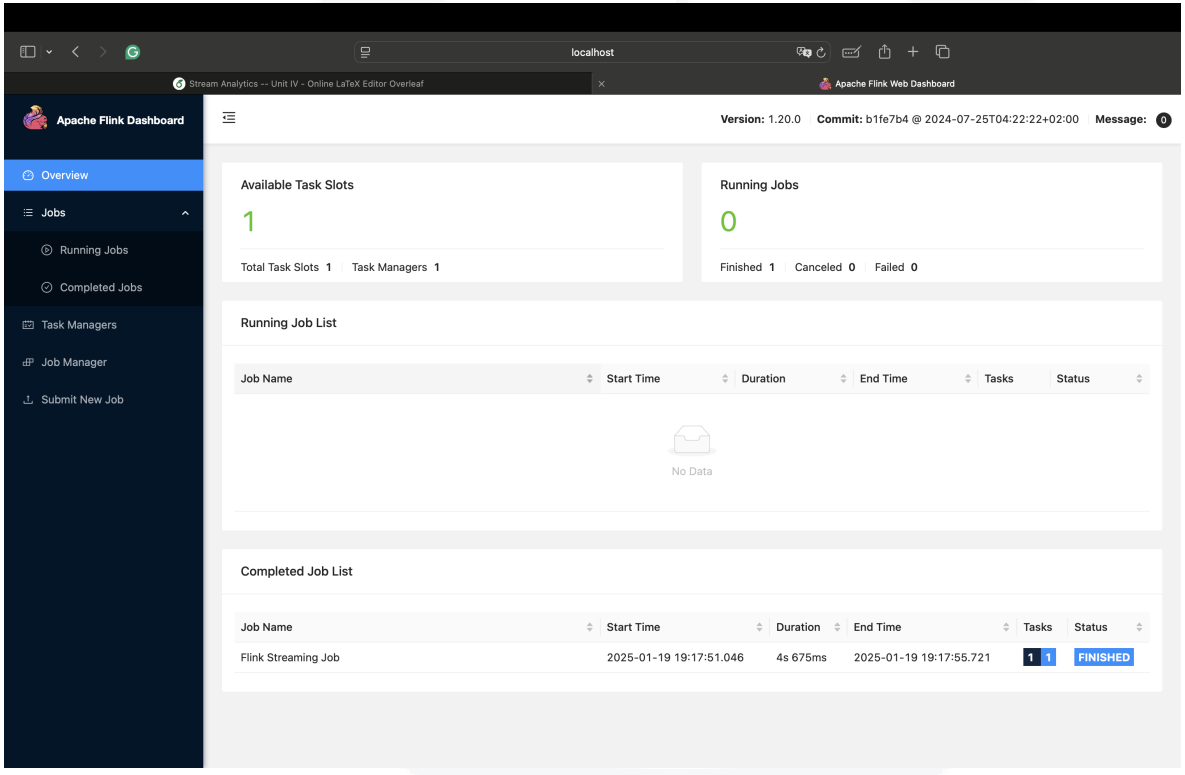


Figure: Flink Web UI (localhost:8081)

Stream Analytics

YARN Cluster

- Utilising Hadoop YARN for resource management
- Using `yarn-cluster` option of `flink run`
- Flink Cluster must be started at first
- Submit a Job to YARN – `./bin/flink run -m yarn-cluster -py flink_job.py`
- Web UI – localhost:8088
- Note – An active Hadoop Cluster must be running
- Add HADOOP_CLASSPATH using

```
export HADOOP_CLASSPATH
=/usr/local/Cellar/hadoop/3.4.0/libexec/
share/hadoop/common/*
```

- Latest official support – Hadoop v2.8.3

Stream Analytics

Kubernetes Cluster and Docker

PES UNIVERSITY

- Flink natively supports Kubernetes for resource orchestration
- Using Flink Docker images to deploy on Kubernetes
- Deploy Flink using Docker Compose

Stream Analytics

Best Practices

- Test the application locally before deploying to a cluster
- Ensure all dependencies are packaged with the application
- Monitor resources using Flink's Web UI or Kubernetes Dashboard
- Use checkpoints and savepoints for fault tolerance in production

Stream Analytics

References

- Apache Flink Documentation: https://nightlies.apache.org/flink/flink-docs-stable/
- PyFlink API Reference: https://nightlies.apache.org/flink/flink-docs-stable/api/python/
- Learning Apache Flink by Tanmay Deshpande: Chapter on Flink Deployment Modes
- Stream Processing with Apache Flink: Concepts, Examples, and Tutorials
- YARN Cluster for Flink – Docs

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

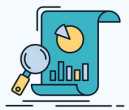
Computer Applications

Stream Analytics

Deploying and Monitoring Apps

HA Setups

Niteesh K R
Computer Applications



Stream Analytics

High Availability (HA) Setup

- Ensures continuous operation of applications even during component failures
- Reduces downtime and prevents data loss in real-time streaming systems
- Key to business-critical applications requiring reliability and fault tolerance

Stream Analytics

HA Setup – Key Terminologies

Checkpointing

- Periodic snapshots of application state stored in durable storage
- Used to resume processing after failures without data loss

Savepoints

- Manually triggered, persistent checkpoints
- Enable application upgrades and migration

Stream Analytics

HA Setup – Key Terminologies

Leader and Leader Election

- The **JobManager** responsible for managing jobs and coordination is the leader
- Leader election ensures only one active JobManager at a time

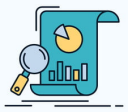
JobManager

- Central component managing scheduling, execution, and checkpoints
- HA setups include multiple JobManagers, with one active and others in standby mode

Stream Analytics

HA Setup in Flink – Benefits

- Ensures fault tolerance by recovering from JobManager or TaskManager failures
- Maintains state consistency via checkpointing and savepoints
- Provides seamless failover to standby JobManagers
- HA in Flink is possible using **ZooKeeper** and **Kubernetes**



Stream Analytics

HA Setup – ZooKeeper – Steps

- Download ZooKeeper v3.9.3 from ZooKeeper Official Site – Downloads – Using wget
- Unzip using `tar -xvzf filename.tar.gz` and `cd` to the directory
- Copy `conf/zoo_sample.cfg` to `conf/zoo.cfg` using `cp conf/zoo_sample.cfg conf/zoo.cfg`
- Setup a permanent directory for ZooKeeper – Modify the line: dataDir=./var/lib/zookeeper
- Create the directory `./var/lib/zookeeper` inside the same directory
- Start ZK – `./bin/zkServer.sh start`
- `./zkServer.sh status` – Must show port 2181 found

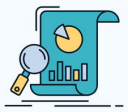
Stream Analytics

HA Setup – Flink – Steps

- Move to Flink Installation directory and create a new directory, `checkpoints`
- Open conf/config.yaml and add the below lines – Make sure to set the path of the `checkpoints` directory as per the system

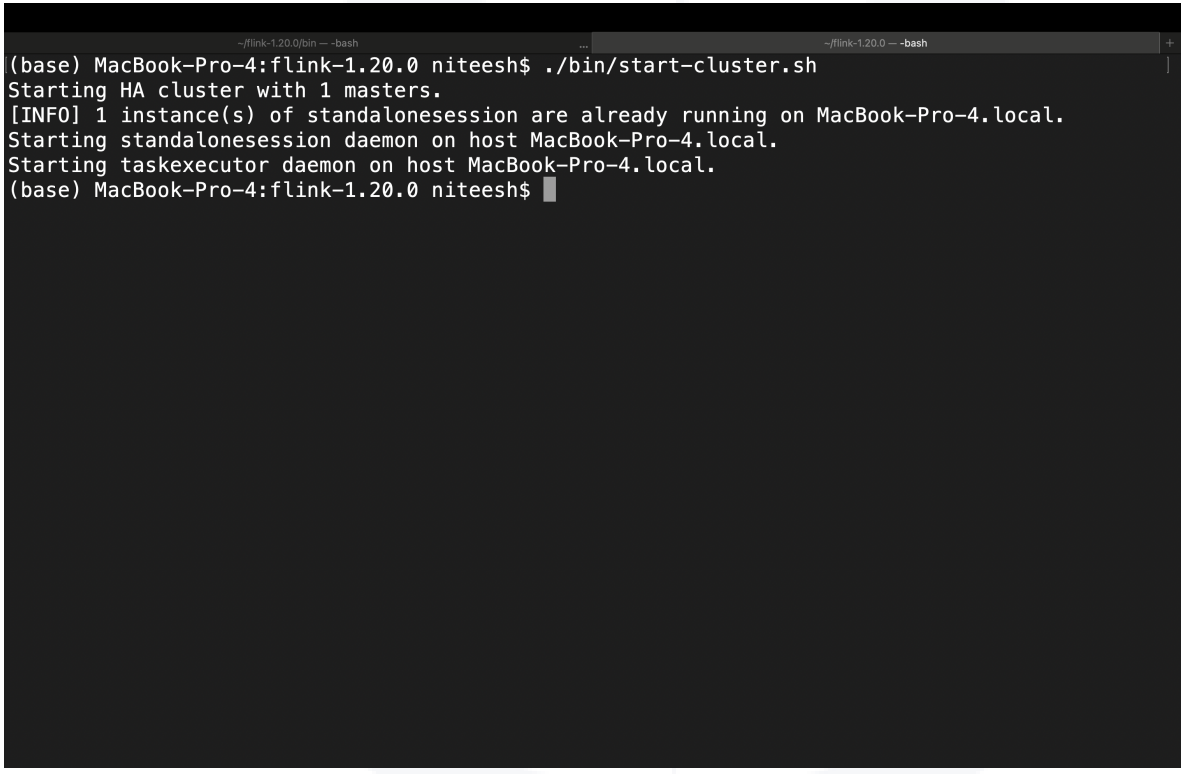
```
high-availability: zookeeper
high-availability.storageDir: file:///Users/niteesh/flink-1.20.0/checkpoints/
high-availability.zookeeper.quorum: 127.0.0.1:2181
high-availability.zookeeper.path.root: /flink
```

- Create a new directory, `sudo mkdir /var/lib/zookeper` and set 777 using `chmod`
- Start Cluster – This now starts a HA Cluster
- Note – Remove the lines added from the YAML file to switch back to the standalone mode as the default



Stream Analytics

Starting Flink Cluster – HA



Stream Analytics

HA Setup – Observations

- Run the `examples/streaming/WordCount.jar` using `./bin/flink run`
- Note the JobID
- Find the PID of JobManager using `jps` and kill it using `kill -9` $< pid >$
- Wait a few seconds and check the JPS command
- This shows a newly-elected JobManager, ensuring HA Setup using ZooKeeper

Stream Analytics

Best Practices

- Using ZooKeeper or Kubernetes depends on the case
- Use ZooKeeper for a native and custom leader election process
- Kubernetes automatically takes care of the leader election
- Configure HDFS or another shared filesystem for durability

Stream Analytics

References

- Flink HA Setup
- Flink HA – ZooKeeper
- Flink HA – Kubernetes

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

Computer Applications

Stream Analytics

Deploying and Monitoring Apps

Apache Hadoop

Niteesh K R
Computer Applications

Stream Analytics

Apache Hadoop

- Hadoop is a framework for distributed storage and processing of large datasets
- Consists of core components and supporting tools for scalable and fault-tolerant data handling

Stream Analytics

Hadoop – Terminologies

- **NameNode** – Master Node, responsible for client interaction and metadata, does not store actual data
- **DataNode** – Slave Node, stores and processes a part of data
- **Secondary NameNode** – Stores timely backup of NameNode, invokes NameNode when it fails with the latest metadata
- **ResourceManager** – Master Daemon, responsible for managing all the processes and the allocated resources
- **NodeManager** – Slave Daemon, reports about the processing done on a timely basis to the ResourceManager
- **Cluster** – 1 NameNode, X^{le} DataNodes, and YARN (RM + NM)

Stream Analytics

Hadoop – Components

Core Components

- Hadoop Distributed File System (HDFS) – Storage
- Yet Another Resource Negotiator (YARN) – MapReduce 2.0
- MapReduce – Programming Model

Supporting Components

- Apache Hive
- Apache HBase
- Apache Pig
- Apache ZooKeeper

Stream Analytics

HDFS

- Distributed, scalable, and fault-tolerant file system
- Stores data in large blocks distributed across multiple nodes
- Provides high throughput access for batch and streaming data
- Web UI – localhost:9870

Stream Analytics

YARN

- Resource management and job scheduling framework
- Allocates system resources to distributed applications
- Supports multiple computing frameworks like Flink, Spark, and MapReduce
- Web UI – localhost:8088

Stream Analytics

MapReduce

- A programming model for parallel data processing
- Processes data by dividing tasks into ”Map” and ”Reduce” phases
- Provides a simple way to process large-scale datasets

Stream Analytics

Hive and HBase

Hive

- A data warehousing tool built on Hadoop
- Allows querying and analysis of large datasets using SQL-like language

HBase

- A NoSQL database that runs on HDFS
- Optimized for real-time read and write operations

Stream Analytics

Pig and ZK

Pig

- A high-level scripting language for data transformation
- Converts scripts into MapReduce jobs automatically

ZooKeeper

- A coordination service for distributed applications
- Manages configurations, synchronization, and naming

Stream Analytics

References

- Apache Hadoop v3.4.0 (Latest Stable) Docs
- HDFS Architecture
- MapReduce Tutorial

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

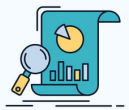
Computer Applications

Stream Analytics

Deploying and Monitoring Apps

Hadoop Integration

Niteesh K R
Computer Applications



Stream Analytics

Flink x Hadoop

- Apache Flink can integrate seamlessly with Hadoop to utilize its storage and resource management capabilities
- HDFS can provide a durable storage solution with a scalable distributed file system
- YARN can enable resource management and job scheduling framework

Stream Analytics

Flink x HDFS

- Distributed and fault-tolerant storage for input/output data
- Flink can read from and write to HDFS using its DataStream and Batch APIs

  Prerequisites
- A running Hadoop Cluster with HDFS configurations set up properly
- Hadoop configuration files (`core-site.xml` and `hdfs-site.xml`) must be added to Flink's classpath

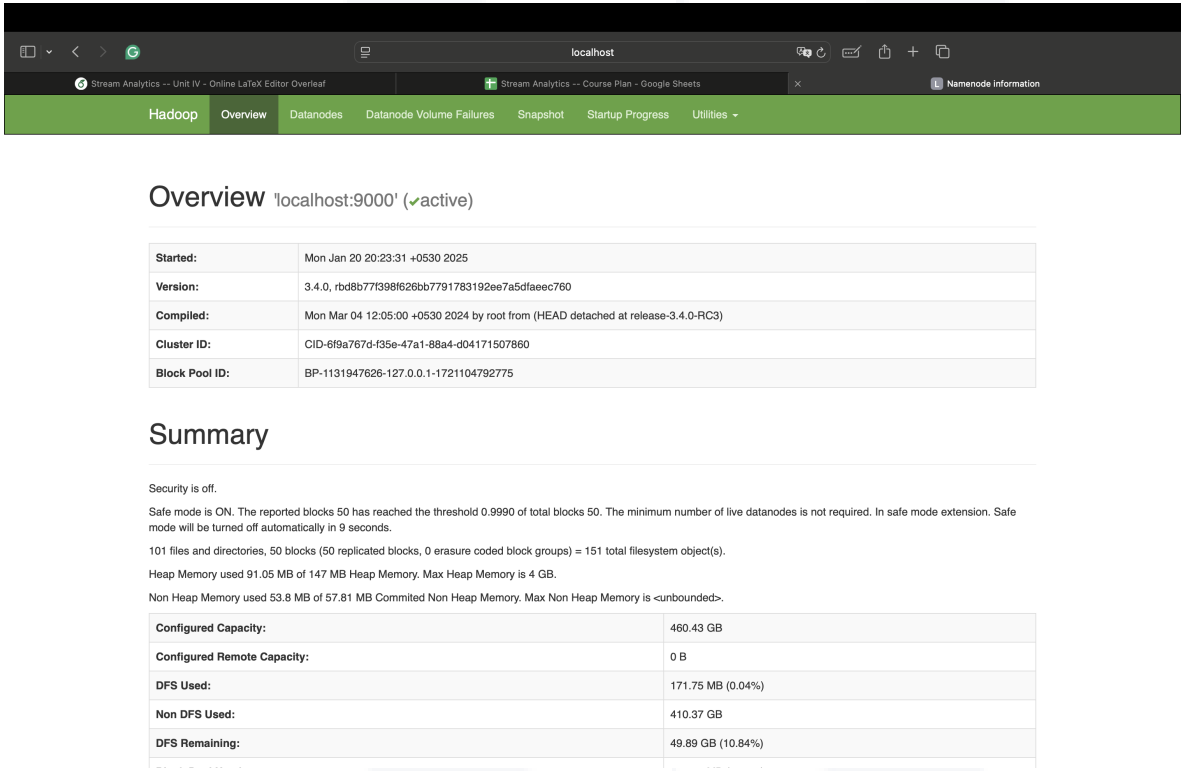
Stream Analytics

Starting HDFS

- Move to the Hadoop installation directory (/usr/local/Cellar/hadoop/3.4.0)
- Navigate to `libexec/sbin` and use `./start-all.sh`
- This shall start all the Hadoop Daemons (NameNode, Secondary NameNode, DataNode (1 only, single node cluster), ResourceManager, and NodeManager)
- Run `jps` to check if the processes are running
- Use `../bin/hdfs dfs namenode -format` to format the HDFS NameNode if there are any problems and if deemed necessary
- Open localhost:9870 on Browser – Web UI for HDFS

Stream Analytics

HDFS Web UI



Stream Analytics

Configuring HDFS on Flink

- Reach `/usr/local/Cellar/hadoop/3.4.0/libexec/etc/hadoop`
- Copy `hdfs-site.xml` and `core-site.xml` to `~/flink-1.20.0/conf`
- Once done, HDFS can be used as a source and sink to Flink
- FileSystem connectors can be used
- URL to have hdfs:// appended to the path

Stream Analytics

HDFS as Source

flink_hdfs_source.py

```
1 from pyflink.datastream import \
2 StreamExecutionEnvironment as SEE
3
4 env = SEE.get_execution_environment()
5
6 env.set_parallelism(1)
7 hdfs_file = 'hdfs://localhost:9000/a.txt'
8
9 data = env.read_text_file(hdfs_file)
10 data.print()
11 env.execute()
```

Line 2, Column 1; Saved ~/flink-1.20.0/programs/flink_hdfs_source.py (UTF-8) Spaces: 4 Python

Stream Analytics

Using YARN with Flink

- YARN can be used as the resource management framework for Flink
- Manages system resources for distributed Flink applications
- Dynamically allocates resources and scales Flink jobs

Modes

- Session Mode – Starts a long-running Flink session on YARN
- Per-job Mode – A separate YARN application is launched for each Flink job

Prerequisites

- Active Hadoop Cluster with YARN Daemons running

Stream Analytics

Configuring Flink to YARN

- Open `conf/config.yaml` in the Flink installation directory
- Add the below lines:

```
high-availability: zookeeper
high-availability.storageDir: file:///Users/niteesh/flink-1.20.0/checkpoints/
high-availability.zookeeper.quorum: 127.0.0.1:2181
high-availability.zookeeper.path.root: /flink
```

- This is the same as the HA setup
- Run this: `export HADOOP_CLASSPATH=$(hadoop classpath)`
- Alternatively, add the above line to ~/.bashrc and run `source ~/.bashrc`

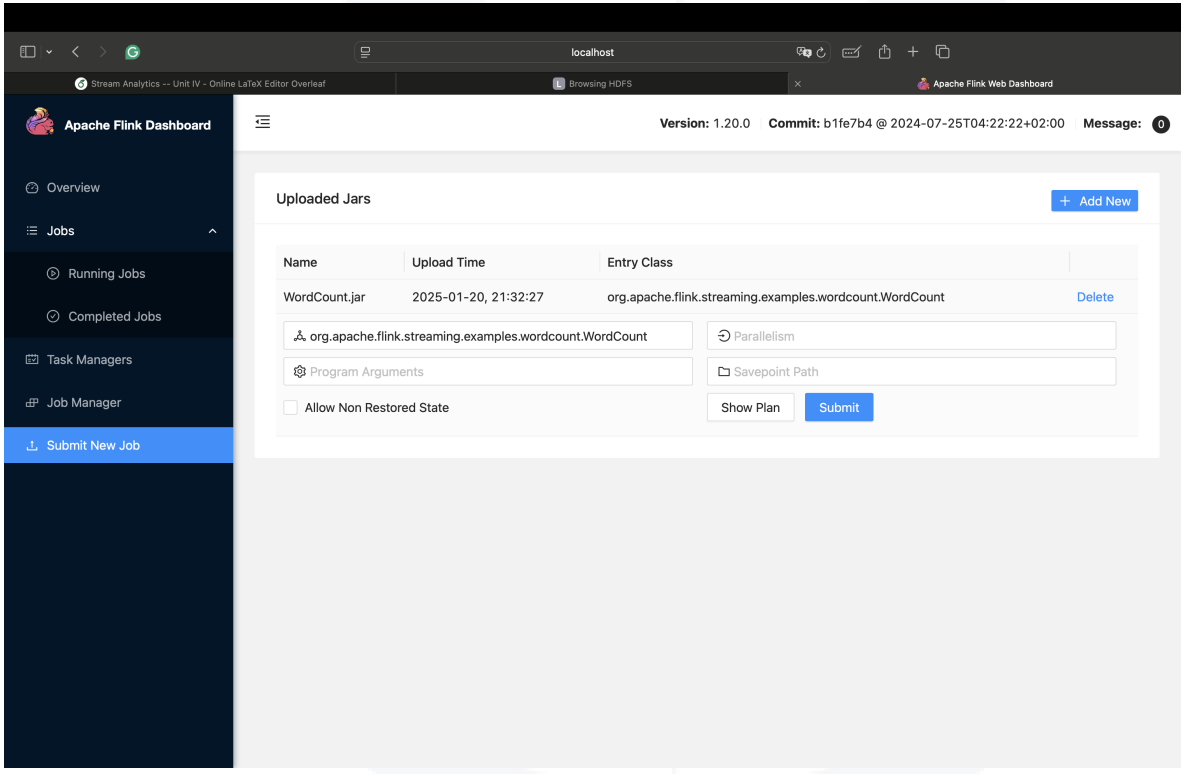
Stream Analytics

YARN x Flink

- Start Flink Cluster – Using `bin/start-cluster.sh` – Starts in HA mode
- Upload a JAR file (From Flink Examples in the examples directory) on Web UI
- The process shall now run on Flink but, using the underlying YARN framework

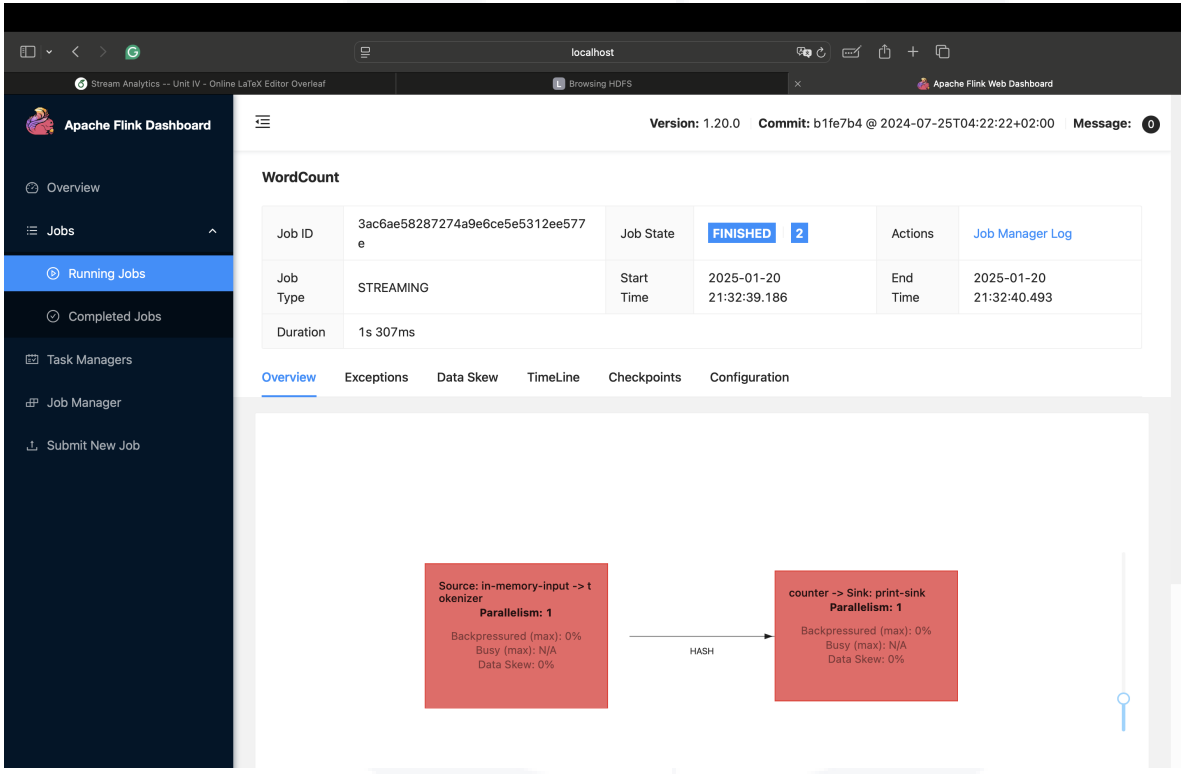
Stream Analytics

Uploading Job



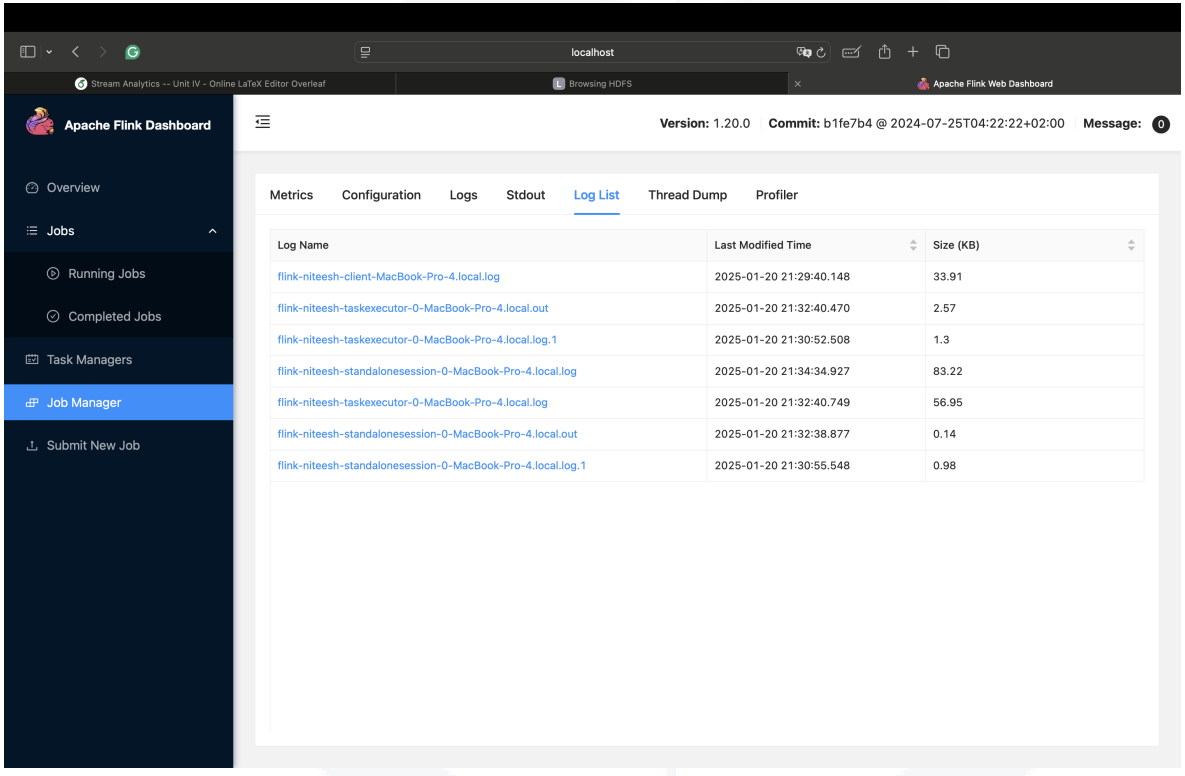
Stream Analytics

Running Flink Job – UI



Stream Analytics

Flink Logs



- .out file consists of the outputs

Stream Analytics

References

- YARN for Flink
- Standalone Deployment
- ZooKeeper HA Setup

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

Computer Applications

Stream Analytics

Deploying and Monitoring Apps

File System Configuration

Niteesh K R
Computer Applications

Stream Analytics

FS Config for Flink

PES UNIVERSITY

- Flink provides a unified abstraction to interact with various file systems
- Supports local and distributed file systems for reading, writing, and state storage
- File systems are identified using URI schemes (e.g., `file://`, `hdfs://`, `s3://`)

Stream Analytics

Supported File Systems

- Local File System
- Hadoop Distributed File System
- Amazon S3
- Aliyun Object Storage Service
- Azure Data Lake Store Gen2
- Azure Blob Storage
- Google Cloud Storage

Stream Analytics

Local FS

- URI begins with `file://`
- Default file system used by Flink when no URI scheme is provided
- No special configuration is required
  Example:
  - `state.backend: filesystem`
  - `state.checkpoints.dir: file:///path/to/checkpoints`

Stream Analytics

HDFS

- URI begins with `hdfs://`
- Provides scalable, fault-tolerant distributed storage
- Requires Hadoop configuration files (`core-site.xml` and `hdfs-site.xml`) to be in Flink's 'conf' directory

  Necessary JARs:
  - `hadoop-common.jar`
  - `hadoop-hdfs.jar`

  Configuration (In config.yaml):
  - `fs.hdfs.hadoopconf: /path/to/hadoop/etc/hadoop`
  - `state.backend: filesystem`
  - `state.checkpoints.dir: hdfs://namenode:9000/checkpoints`

Stream Analytics

Amazon S3

- S3 – Secured Storage Service
- URI begins with `s3://`
- Used for scalable object storage
- Requires Flink S3 connectors

  Necessary JARs:
  - `flink-s3-fs-hadoop.jar` (Hadoop-based connector)
  - `flink-s3-fs-presto.jar` (Presto-based connector)

  Configuration (In config.yaml):
  - `s3.access-key: your-access-key`
  - `s3.secret-key: your-secret-key`
  - `state.backend: filesystem`
  - `state.checkpoints.dir: s3://your-bucket/checkpoints`

Stream Analytics

Azure Object Storage and Blob Storage

- URI begins with `abfs://`
- Blob – Binary Large Object
- Optimized for use with Azure Data Lake Storage

  Necessary JARs:
  - `hadoop-azure.jar`
  - `azure-storage.jar`

  Configuration (In config.yaml):
  - `fs.azure.account.key.account-name.dfs.core.windows.net: access-key`
  - `state.backend: filesystem`
  - `state.checkpoints.dir: abfs://your-container@account-name.dfs.core.windows.net/chec`

Stream Analytics

Google Cloud Storage (GCS)

- URI starts with `gs://`
- Requires Hadoop GCS connector

  Necessary JARs:
  - `gcs-connector-hadoop3-latest.jar`

  Configuration (In config.yaml):
  - `fs.gs.project.id: your-project-id`
  - `fs.gs.auth.service.account.json.keyfile: /path/to/keyfile.json`
  - `state.backend: filesystem`
  - `state.checkpoints.dir: gs://your-bucket/checkpoints`

Stream Analytics

Alluxio

- URI begins with `alluxio://`
- A distributed storage system that unifies access to multiple storage backends

  Necessary JARs:
  - `alluxio-core-client.jar`

  Configuration (In config.yaml):
  - `alluxio.master.hostname: master-hostname`
  - `alluxio.master.port: 19998`
  - `state.backend: filesystem`
  - `state.checkpoints.dir: alluxio://hostname:19998/checkpoints`

Stream Analytics

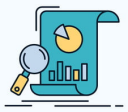
Testing FS Configuration

- Run a simple Flink job to read from and write to the configured file system
  Example command:
  - ```
    ./bin/flink run -c
    org.apache.flink.streaming.examples.wordcount.WordCount
    \
    examples/streaming/WordCount.jar \
    --input hdfs://namenode:9000/input \
    --output hdfs://namenode:9000/output
    ```
- Verify the output in the specified path
    ```

Stream Analytics

Common Issues with FS Config

- **Unsupported File System Scheme:** Ensure required JARs are in `lib/`
- **Authentication Failures:** Check credentials for S3 or Azure Blob Storage
- **File Not Found:** Verify input/output paths are accessible



Stream Analytics

Best Practices for FS Config

- Use distributed file systems like HDFS or S3 for scalability
- Optimize checkpoint and state storage paths for performance
- Monitor file system performance for large-scale jobs

Stream Analytics

References

- Apache Flink Documentation
- File Systems for Flink – Documentation
- Common FS Config – Documentation

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

Computer Applications

Stream Analytics

Deploying and Monitoring Apps

Flink Clusters

Niteesh K R
Computer Applications

Stream Analytics

Flink Clusters

A Flink cluster consists of two primary components:

- **JobManager:** Coordinates task execution, scheduling, and fault tolerance
- **TaskManager:** Executes individual tasks and manages data flow

Flink can run on various cluster environments:

- Standalone cluster
- Resource managers like YARN, Kubernetes, or Mesos
- Cloud environments like Amazon EMR, Azure, and GCP

Stream Analytics

Cluster Modes

Standalone Cluster:

- Flink's native cluster mode
- Suitable for small setups or testing

Cluster on YARN:

- Leverages Hadoop YARN for resource management
- Enables dynamic allocation of resources for jobs

Cluster on Kubernetes:

- Deploys Flink as containerized applications
- Provides auto-scaling and high availability using Kubernetes orchestration

Stream Analytics

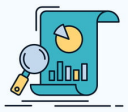
Standalone Cluster Setup

Update Configuration in `conf/config.yaml`:

- `jobmanager.rpc.address: localhost`
- `taskmanager.numberOfTaskSlots: 2`
- `state.backend: filesystem`
- `state.checkpoints.dir: file:///tmp/flink-checkpoints`

Start the Cluster:

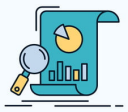
- Start JobManager: `./bin/jobmanager.sh start cluster`
- Start TaskManager: `./bin/taskmanager.sh start`



Stream Analytics

Configuring High Availability

- Ensures fault tolerance by using multiple JobManagers
  HA Configuration in `conf/config.yaml`**:**
  - `high-availability: zookeeper`
  - `high-availability.zookeeper.quorum: localhost:2181`
  - `high-availability.zookeeper.path.root: /flink`
  - `high-availability.storageDir: file:///tmp/flink/ha`



Stream Analytics

Flink Web UI (localhost:8081)

PES UNIVERSITY

- UI monitoring and managing Flink clusters and jobs

Overview Page:

- Displays the cluster status, including total and available task slots
- Shows running and completed jobs

Job Page:

- View details of running and completed jobs
- Monitor job execution time, parallelism, and tasks

Task Manager Page:

- Displays information about active TaskManagers, including slots and metrics

Configuration Page:

- Displays the current Flink cluster configuration loaded from `config.yaml`

Stream Analytics

Flink Web UI – Landing Page

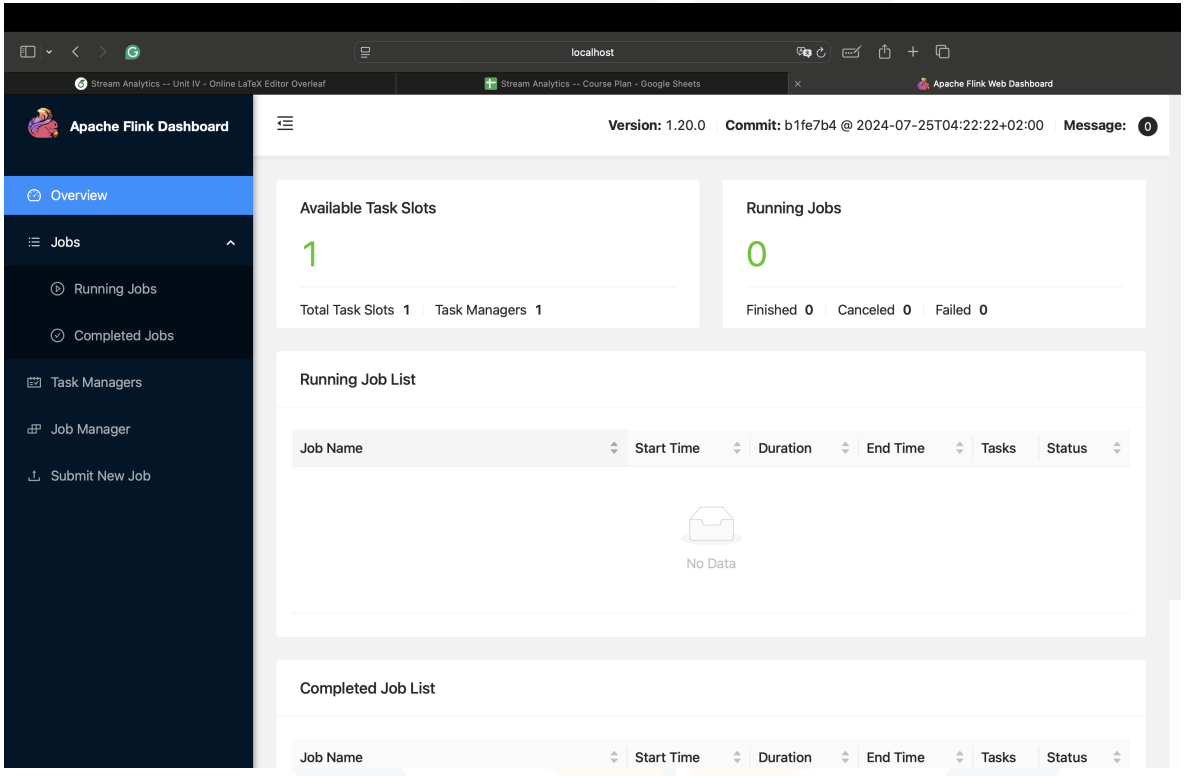


Figure: Flink Web UI – Landing Page

Stream Analytics

Submitting Jobs

Use the Flink CLI to submit jobs:

- `./bin/flink run examples/streaming/WordCount.jar`
- For Python Jobs – `./bin/flink -py run path/to/flink/program.py`

Use the Web UI to submit jobs:

- Navigate to Flink's Web UI
- Go to **Submit New Job**
- Upload the JAR file and provide program arguments

Stream Analytics

Submitting New Job – Flink Web

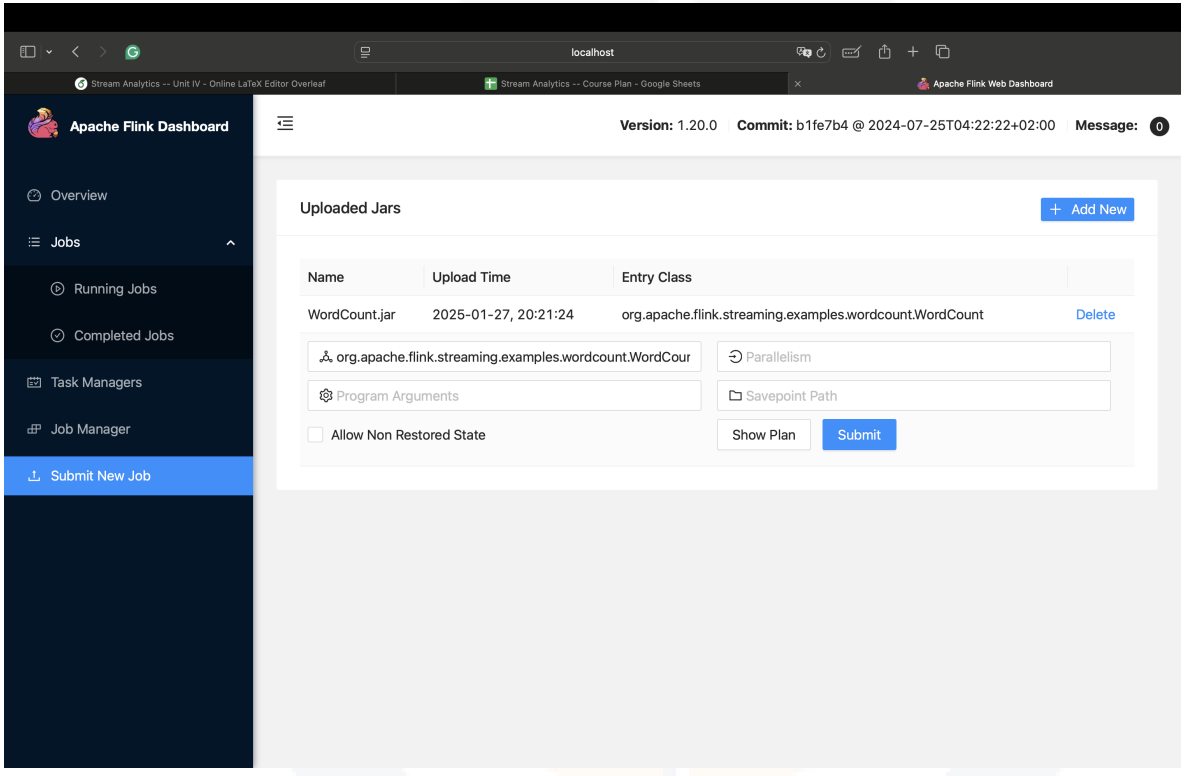


Figure: Submitting Job – Flink Web UI

Stream Analytics

Debugging

Check logs for JobManager and TaskManager:

- JobManager: `logs/jobmanager.log`
- TaskManager: `logs/taskmanager.log`

- Monitor task execution in the Web UI
- Use metrics from **Task Manager Page** for resource analysis

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

Computer Applications

PES UNIVERSITY

Stream Analytics

Deploying and Monitoring Apps

Running and Managing Flink Apps

Niteesh K R

Computer Applications

Stream Analytics

Running Flink Jobs

PES UNIVERSITY

Using Flink CLI:

- Submit a job: `./bin/flink run examples/streaming/WordCount.jar`
- List running jobs: `./bin/flink list`

Using Flink Web UI:

- Open `http://localhost:8081`
- Navigate to **Submit New Job**
- Upload a JAR and provide job arguments
- Click **Submit**

Stream Analytics

Monitoring Flink Jobs

Checking Job Status:

- View jobs in **Running Jobs** or **Completed Jobs**
- Check execution time and state

Task and Operator-Level Performance:

- Monitor parallelism and execution time per task
- Identify straggler tasks

Stream Analytics

Flink UI – Key Metrics

- **Throughput** – Number of events processed per second
- **Latency** – Time taken for an event to be fully processed
- **Watermark Lag** – Delay in event processing

Checkpointing Metrics

- Number of successful and failed checkpoints
- Checkpoint duration

Task and Operator Metrics

- CPU utilization
- Memory consumption
- Network I/O

Stream Analytics

Accessing and Configuring Metrics

PES UNIVERSITY

Enabling Metrics in `conf.yaml`

- `metrics.reporter.prom.class: org.apache.flink.metrics.prometheus.PrometheusReporter`
- `metrics.reporter.prom.port: 9249`

Retrieving Metrics via REST API

- Fetch job metrics: item[] `curl http://localhost:8081/jobs/<job-id> | python -m json.tool`
  (Pretty printing)

Stream Analytics

Job Metrics

```
(base) MacBook-Pro-4:flink-1.20.0 niteesh$ curl http://localhost:8081/jobs/8e94fd25b8fe409c1211aa3b566261d5 | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2576  100  2576    0     0   299k      0 --:--:-- --:--:-- --:--:--  314k
{
    "jid": "8e94fd25b8fe409c1211aa3b566261d5",
    "name": "WordCount",
    "isStoppable": false,
    "state": "FINISHED",
    "job-type": "STREAMING",
    "start-time": 1738488216186,
    "end-time": 1738488217520,
    "duration": 1334,
    "maxParallelism": -1,
    "now": 1738488410137,
    "timestamps": {
        "FAILED": 0,
        "SUSPENDED": 0,
        "RECONCILING": 0,
        "CANCELLING": 0,
        "CANCELED": 0,
        "INITIALIZING": 1738488216186,
        "RUNNING": 1738488216413,
        "FINISHED": 1738488217520,
        "RESTARTING": 0,
        "FAILING": 0,
        "CREATED": 1738488216289
    },
    "vertices": [
        {
            "id": "cbc357ccb763df2852fee8c4fc7d55f2",
            "slotSharingGroupId": "ac1a6d95801f983f32425f43a06d7c40",
            "name": "Source: in-memory-input -> tokenizer",
            "maxParallelism": 128,
            "parallelism": 1,
            "status": "FINISHED",
            "start-time": 1738488216753,
            "end-time": 1738488217508,
            "duration": 755,
            "tasks": {
```

Figure: Flink Job Metrics – curled using REST API

Stream Analytics

Debugging and Optimizing

Identifying Bottlenecks:

- Check TaskManager overload in the Web UI.
- Monitor task execution times.

Optimizing Performance:

- Adjust `parallelism.default` in `flink-conf.yaml`.
- Optimize checkpoint intervals:

```
execution.checkpointing.interval:  60000
```

Checking Logs for Issues:

- View logs from Web UI or check:

```
logs/jobmanager.log
logs/taskmanager.log
```

Stream Analytics

References

- Apache Flink Documentation
- Flink Metrics
- Flink Config Guide
- Flink REST API for Metrics
- Flink Web UI Overview
- Flink GitHub Repo (Official)

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

Computer Applications

Stream Analytics

Deploying and Monitoring Apps

Task Scheduling

Niteesh K R
Computer Applications

Stream Analytics

Task Scheduling on Flink

- Flink schedules tasks as Directed Acyclic Graphs (DAGs)

  Key Scheduling Components

  - **JobManager** – Assigns tasks to available TaskManagers
  - **TaskManager** – Executes tasks using available slots

  Scheduling Modes

  - **Pipelined Scheduling** – Executes tasks as soon as data arrives
  - **Batch (Lazy) Scheduling** – Waits for all input data before execution

Stream Analytics

Parallelism and Task Slots

- **Parallelism** – Determines the number of concurrent task instances
- **Task Slots** – Each TaskManager has limited slots for execution

  Configuring Parallelism in `conf.yaml`
  - `parallelism.default: 4`
  - `taskmanager.numberOfTaskSlots: 2`

Stream Analytics

Slot and Resource Grouping

- **Slot Sharing** – Allows multiple tasks to run in the same slot
- **Co-Location Groups** – Ensures that related tasks execute on the same node
- **Setting Parallelism in Code** – `env.set_parallelism(4)`

Fine-Grained Resource Management

CPU and Memory Allocation

- `taskmanager.cpu.cores: 2.0`
- `taskmanager.memory.process.size: 1728m`

- **Dynamic Scaling** – Adjust resources dynamically based on job load

Stream Analytics

Scheduling in Batch and Stream Jobs

PES UNIVERSITY

- **Batch Jobs** – Use **lazy scheduling** for efficient execution
- **Streaming Jobs** – Use **pipelined execution** for lower latency

Stream Analytics

Debugging and Optimization

- **Monitoring via Web UI** – Check TaskManager slots and execution status
- **Log Analysis** – Inspect `logs/taskmanager.log` for bottlenecks
- **Optimizing Resource Usage** – Adjust parallelism and slot sharing

Stream Analytics

References

- Apache Flink Docs
- Fine-Grained Resource Management
- Elastic Scaling
- Speculative Execution
- Metric Reporters

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231

Stream Analytics

Deploying and Monitoring Apps

Niteesh K R

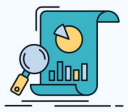
Computer Applications

Stream Analytics

Deploying and Monitoring Apps

Monitoring Flink Clusters

Niteesh K R
Computer Applications



Stream Analytics

Monitoring Flink Clusters

- Monitoring ensures that Flink clusters are healthy and performing optimally

Key Areas to Monitor

- **Cluster Health:** JobManager and TaskManager availability
- **Resource Utilization:** CPU, memory, and task slot usage
- **Job Performance:** Throughput, latency, and backpressure
- **Checkpointing:** Success rate, duration, and failures

Stream Analytics

Cluster Health Monitoring

- **JobManager Status:** Ensures a leader is active and scheduling tasks
- **TaskManager Availability:** Monitors available and lost TaskManagers

  Checking Cluster Health Using Web UI:
  - Open `http://localhost:8081`
  - Navigate to the **Overview** page
  - Check available TaskManagers and slots

Stream Analytics

Performance Metrics for Clusters

- **CPU & Memory Usage** – Monitors JobManager and TaskManager resource consumption
- **Task Slot Utilization** – Tracks available vs used slots

Checkpointing Metrics

- **Checkpoint Duration** – Measures the time taken for state snapshots
- **Checkpoint Failures** – Alerts when state persistence fails

Network and I/O Metrics

- **Throughput** – Events processed per second
- **Backpressure** – Indicates congestion in task execution

Stream Analytics Using Web UI

- Open `http://localhost:8081` to access the dashboard
- **Overview Page** – Displays cluster-wide metrics and available slots
- **TaskManager Page** – Shows CPU, memory, and task execution status
- **Job Page** – Provides real-time execution statistics for running jobs

Stream Analytics

External Monitoring Tools

Prometheus + Grafana

- Collects and visualizes real-time Flink metrics
- Example Flink metrics
  - `flink_taskmanager_Status_JVM_CPU_Load`
  - `flink_jobmanager_job_duration`

- **Elasticsearch + Kibana** – Stores and visualizes logs from JobManagers and TaskManagers
- **Datadog / New Relic** – Provides cloud-based monitoring and alerting

Stream Analytics

Using CLI – `curl`

- List running jobs –

  `curl http://localhost:8081/jobs`
- Fetch TaskManager metrics –

  `curl http://localhost:8081/taskmanagers`
- Note – Use | **python -m json.tool** for pretty-printing

Stream Analytics

References

- Apache Flink Docs
- Fine-Grained Resource Management
- Elastic Scaling
- Speculative Execution
- Metric Reporters

Thank You

Niteesh K R

Assistant Professor

Department of Computer Applications

niteeshkr@pes.edu

080-26721983 Extn: 231