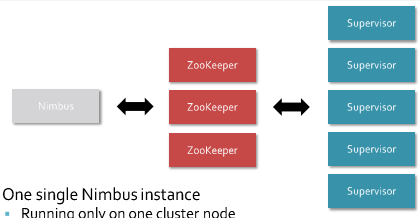
Apache Storm

Storm architecture

Storm architecture

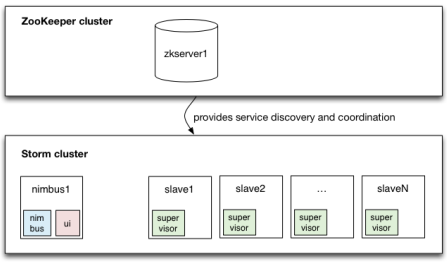
- A Storm cluster follows a master-slave model where the master and slave processes are coordinated through ZooKeeper
- The following are the components of a Storm cluster
  - Nimbus (master)
  - Supervisors (slaves)
  - Zookeeper

Storm architecture



- One single Nimbus instance
  - Running only on one cluster node
- Many instances of ZooKeeper
  - Each instance runs on a different cluster node
- Many Supervisors
  - Each Supervisor runs on a different cluster node

Storm architecture



Nimbus

- The Nimbus node is the master in the Storm cluster
- Nimbus is responsible for
  - Distributing the application code across various worker nodes
  - Assigning tasks to different machines/nodes
  - Monitoring tasks for any failures
  - Restarting tasks as and when required

Nimbus

- Nimbus is stateless and stores all of its data in ZooKeeper
- There is a single Nimbus node in a Storm cluster
- Nimbus is designed to be fail-fast
  - When Nimbus dies, it can be restarted without having any effects on the already running tasks on the worker nodes

7

Supervisor nodes

- Supervisor nodes are the worker (slave) nodes in a Storm cluster
- Each supervisor node runs a supervisor daemon that is responsible for
  - Creating, starting, stopping worker processes
  - Executing the tasks assigned to each worker process

8

Supervisor nodes

- The supervisor daemon is fail-fast
  - It stores all of its state in ZooKeeper and hence it can be restarted without any state loss
- A single supervisor daemon normally handles multiple worker processes running on the same server

9

ZooKeeper

- ZooKeeper is the application that is used to coordinate the processes of Storm and share the configuration information
- Nimbus and the supervisor nodes do not communicate directly with each other
- All of the states associated with the cluster and the various tasks submitted to the Storm are stored in ZooKeeper
  - Hence, both Nimbus and the supervisor daemons can be killed abruptly without adversely affecting the cluster

10

Comparison with Hadoop vers. 1

Hadoop v1	Storm	
JobTracker	**Nimbus (only 1)**	▪ distributes code around cluster ▪ assigns tasks to machines/supervisors ▪ failure monitoring ▪ is fail-fast and stateless (you can "kill -9" it)
TaskTracker	**Supervisor (many)**	▪ listens for work assigned to its machine ▪ starts and stops worker processes as necessary based on Nimbus ▪ is fail-fast and stateless (you can "kill -9" it) ▪ shuts down worker processes with "kill -9", too
MR job	**Topology**	▪ processes messages forever (or until you kill it) ▪ a running topology consists of many worker processes spread across many machines

11

Storm architecture: fault tolerance

- What happens when Nimbus dies (master node)?
  - If Nimbus is run under process supervision as recommended (e.g. via supervisord), it will restart like nothing happened
- While Nimbus is down:
  - Existing topologies will continue to run, but you cannot submit new topologies
  - Running worker processes will not be affected
  - Also, Supervisors will restart their (local) workers if needed
  - However, failed tasks will not be reassigned to other machines, as this is the responsibility of Nimbus

12

Storm architecture: fault tolerance

- What happens when a Supervisor dies (slave node)?
  - If Supervisor runs under process supervision as recommended (e.g. via supervisord), will restart like nothing happened
  - Running worker processes will not be affected

13

Storm architecture: fault tolerance

- What happens when a worker process dies?
  - It's parent Supervisor will restart it
  - If it continuously fails on startup and is unable to heartbeat to Nimbus, Nimbus will reassign the worker to another machine

14

Storm architecture: fault tolerance

- Also ZooKeeper is fault-tolerant
  - This goal is achieved by creating multiple instances of Zookeeper, each one running on a different node of the cluster
- The ZooKeeper cluster keeps working as long as the majority of the its nodes are running
  - For example
    - if we have four ZooKeepernodes, then we can handle only one node failure (at least 3 nodes must always be running)
    - if we have five nodes, then we can handle two node failures (at least 3 nodes must always be running)

15

Storm hardware specifications

- ZooKeeper
  - Preferably use dedicated severs because ZooKeeper is a bottleneck for Storm
    - One ZooKeeper instance per server
  - I/O is a bottleneck for ZooKeeper
    - Put ZooKeeper storage on its own disk device
    - SSDs dramatically improve performance
  - Preferably run ZooKeeper with a number of nodes >= 3 in production environments so that you can tolerate the failure of 1 ZooKeeper server

16

Storm hardware specifications

- Nimbus (master node)
  - Comparatively little load on Nimbus
    - A medium-sized server can be used

17

Storm hardware specifications

- Storm Supervisors (slave nodes)
  - Exact specifications depend on anticipated usage
    - CPU heavy
      - e.g., machine learning applications
    - CPU light
      - e.g. rolling windows, pre-aggregation (in this case get more RAM)
  - CPU cores
    - More is usually better
      - The more you have the more threads you can support (i.e. parallelism)
      - Storm potentially uses a lot of threads

18

Storm hardware specifications

- Storm Supervisors (slave nodes)
  - Memory
    - Highly specific to actual use case
    - Considerations: #workers (= JVMs) per node?
    - Are you caching and/or holding in-memory state?

19

Running Apache Storm Securely

- Original design was not created with security in mind
- Apache Storm version 1.0.2 offers a range of configuration options for securing your cluster
  - By default all authentication and authorization is disabled but can be turned on as needed
  - Storm offers pluggable authentication support through thrift, SASL, or Kerberos
  - You may need to enable a Firewall to limit the access to the services of Storm and IPsec to encrypt all traffic being sent between the hosts in the cluster if you are managing sensible data

20

Monitoring Storm

21

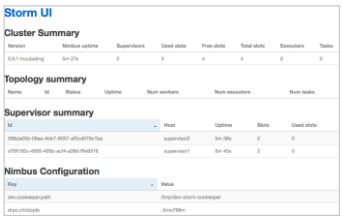
Monitoring Storm

- Storm topologies can be managed and monitored using
  - The Storm Web UI (User Interface)
  - The Storm CLI (Command Line Interface)
- Cluster statistics can also be collected by using the Nimbus thrift client
- You can also use
  - Standard monitoring tools such as Graphite & friends
  - Consider collecting log files into a central place such as Logstash/Kibana

22

Storm Web UI

- Built-in Storm Web UI
  - Listens on 8080/tcp by default



23

Storm Web UI

- Storm Web UI provides the following information
  - Cluster Summary
    - Version of Storm
    - Uptime of the nimbus node
    - Number of free worker slots
    - Number of used worker slots
    - …
    - While submitting a topology to the cluster, the Free slots column should not be zero
      - Otherwise, the topology doesn't get any worker for processing and will wait in the queue till a worker becomes free

24

Storm Web UI

- Nimbus Configuration
  - This portion of the Storm UI shows the configuration of the Nimbus node
- Supervisor summary
  - The list of supervisor nodes running in the cluster along with their
    - Id
    - Host
    - Uptime
    - Slots
    - Used slots

25

Storm Web UI

- Topology summary
  - List of topologies running in the Storm cluster along with their
    - ID
    - Number of workers assigned to the topology
    - Number of executors
    - Number of tasks
    - Uptime
    - ..

26

Storm Web UI

- By means of the Storm Web UI, each running topology can be
  - Deactivated
    - The topology is paused
  - Activated
    - It is used to reactivate a topology that has been deactivated
  - Killed
  - Rebalanced
    - The number of workers and executors can be changed in order to balance the topology

27

Storm Web UI – Topology statistics

- The Storm Web UI allows also monitoring the topologies and their components
- Topology statistics
  - Number of emitted tuples
  - Number of transferred tuples
  - Number of acknowledged tuple
  - Capacity latency within the window of 10 minutes, 3 hours, 1 day, and since the start of the topology

28

Storm Web UI – Spout statistics

- Main statistics for each Spout
  - Number of executors
  - Number of tasks
  - Number of tuples emitted
  - Number of transferred tuples
  - Average complete latency of a tuple
    - The complete latency is the difference in the timestamp when the spout emits the tuple to the timestamp when the ACK tree is completed for the tuple

29

Storm Web UI – Bolt statistics

- Main statistics for each Bolt
  - Number of executors
  - Number of tasks
  - Number of tuples emitted
  - Number of transferred tuples
  - Capacity (last 10m)
    - Percent of the time spent by the bolt in actually

30

Storm Web UI – Bolt statistics

- Processing tuples in the last 10 minutes.
  - If the value of the Capacity (last 10m) is close to 1, then the bolt is at capacity, and we will need to increase the parallelism of the bolt to avoid an "at capacity" situation.
- Process latency (ms)
  - Process latency means the actual average time (in milliseconds) taken by the bolt to process a tuple
- Execute latency (ms)
  - Execute latency is the sum of the processing time and the time used in sending the acknowledgment.

31

Monitoring Storm topologies

- Why does the Storm UI report seemingly incorrect numbers?
  - Storm samples incoming tuples when computing statistics in order to increase performance
    - Sample rate is configured via topology.stats.sample.rate
    - 0.05 is the default value
      - Storm will pick a random event of the next 20 events in which to increase the metric count by 20
    - 1.00 forces Storm to count everything exactly
      - This gives you accurate numbers at the cost of a big performance hit

32

Nimbus thrift client

- Thrift is a binary protocol and is used for cross-language communication
- The Nimbus node in Storm is a thrift service, and the topologies structure is also defined in the thrift structure
- We can write code in any language to connect to the Nimbus node and retrieve statistics

33

Nimbus thrift client

- Nimbus thrift Java API can be used to perform the following tasks
  - Collecting the Nimbus configuration
  - Collecting supervisors statistics
  - Collecting topologies statistics
  - Collecting spouts statistics
  - Collecting bolts statistics
  - Killing topologies

34

Monitoring ZooKeeper

- Latency of requests
  - Metric target is 0 ms when using SSD in ZooKeeper machines
    - Why? Because SSD's are so fast they typically bring down latency below ZooKeeper's metric granularity (which is per-ms)
- Cluster availability
  - LinkedIn run 5-nodes = tolerates 2 dead
  - Twitter run 13-nodes = tolerates 6 dead

35