Apache Storm 3

Working with Storm Topologies

Date of Publish: 2019-12-17

https://docs.hortonworks.com

Contents

Packaging Storm Topologies

Storm developers should verify that the following conditions are met when packaging their topology into a .jar file:

- Use the maven-shade-plugin, rather than the maven-assembly-plugin to package your Apache Storm topologies. The maven-shade-plugin provides the ability to merge JAR manifest entries, which are used by the Hadoop client to resolve URL schemes.
- Include a dependency for the Hadoop version used in the Hadoop cluster.
- Include both of the Hadoop configuration files, hdfs-site.xml and core-site.xml, in the .jar file. In addition, include any configuration files for HDP components used in your Storm topology, such as hive-site.xml and hbase-site.xml. This is the easiest way to meet the requirement that all required configuration files appear in the CLASSPATH of your Storm topology at runtime.

Maven Shade Plugin

Use the maven-shade-plugin, rather than the maven-assembly-plugin to package your Apache Storm topologies. The maven-shade-plugin provides the ability to merge JAR manifest entries, which are used by the Hadoop client to resolve URL schemes.

Use the following Maven configuration file to package your topology:

```
<plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>1.4</version>
      <configuration>
            <createDependencyReducedPom>true</createDependencyReducedPom>
      </configuration>
      <executions>
           <execution>
                <phase>package</phase>
                <goals>
                     <goal>shade</goal>
                </goals>
                <configuration>
                     <transformers>
                          <transformer
 implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/
>
                          <transformer
 implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                  <mainClass></mainClass>
                          </transformer>
                     </transformers>
                </configuration>
           </execution>
      </executions>
</plugin>
```

Hadoop Dependency

Include a dependency for the Hadoop version used in the Hadoop cluster; for example:

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-client</artifactId>
    <version>2.7.1.2.3.2.0-2950</version>
    <exclusions>
        <exclusion>
            <groupId>org.slf4j</groupId>
```

```
            <artifactId>slf4j-log4j12</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Troubleshooting

The following table describes common packaging errors.

Table 1: Topology Packing Errors

Error	Description
com.google.protobuf. InvalidProtocolBufferException: Protocol message contained an invalid tag (zero)	Hadoop client version incompatibility
java.lang.RuntimeException: Error preparing HdfsBolt: No FileSystem for scheme: hdfs	The .jar manifest files have not properly merged in the topology.jar

Deploying and Managing Apache Storm Topologies

Use the command line interface to deploy a Storm topology after packaging it in a .jar file.

For example, you can use the following command to deploy WordCountTopology from the storm-starter jar:

```
storm jar storm-starter-<starter_version>-storm-
<storm_version>.jar storm.starter.WordCountTopology WordCount -c
 nimbus.host=sandbox.hortonworks.com
```

The remainder of this chapter describes the Storm UI, which shows diagnostics for a cluster and topologies, allowing you to monitor and manage deployed topologies.

Configuring the Storm UI

If Ambari is running on the same node as Apache Storm, ensure that Ambari and Storm use different ports. Both processes use port 8080 by default, so you might need to configure Apache Storm to use a different port.

About this task

If you want to use the Storm UI on a Kerberos-enabled cluster, ensure that your browser is configured to use authentication for the Storm UI. For example, complete the following steps to configure Firefox:

Procedure

1. Go to the about:config configuration page.
2. Search for the network.negotiate-auth.trusted-uris configuration option.
3. Double-click on the option.
4. An "Enter string value" dialog box opens.
5. In this box, enter the value http://storm-ui-hostname:8080.
6. Click OK to finish.
7. Close and relaunch the browser.

What to do next

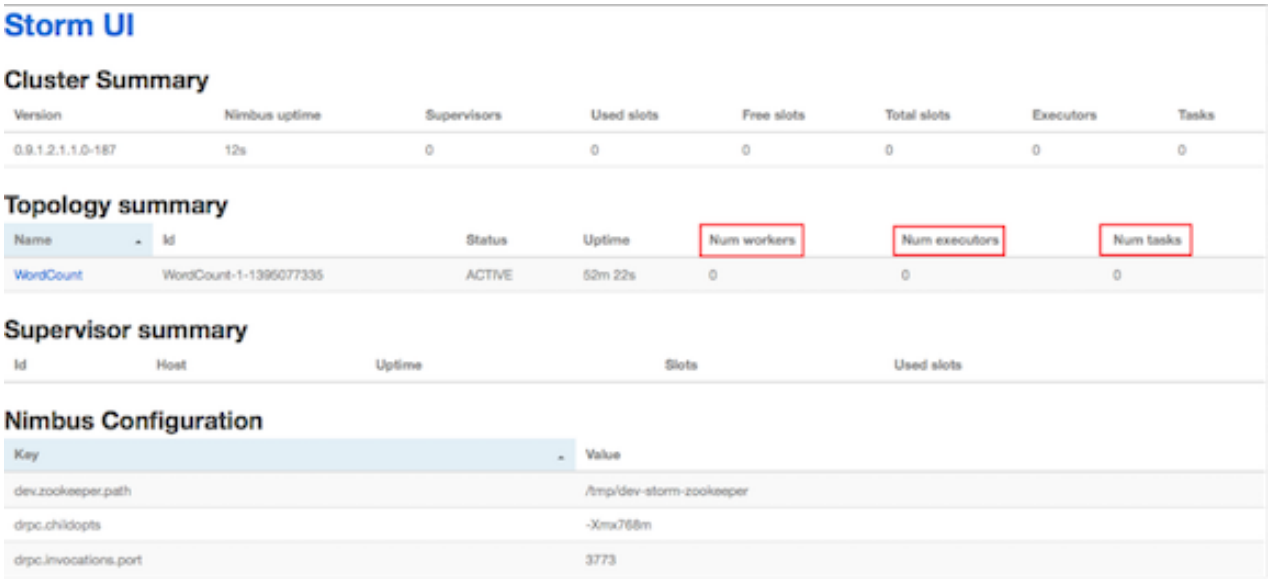
If your cluster is not managed by Ambari, refer to the UI/Logviewer section of Apache Storm security documentation for additional configuration guidelines.

Using the Storm UI

To access the Storm UI, point a browser to the following URL:

http://<storm-ui-server>:8080

In the following image, no workers, executors, or tasks are running. However, the status of the topology remains active and the uptime continues to increase. Storm topologies, unlike traditional applications, remain active until an administrator deactivates or kills them.

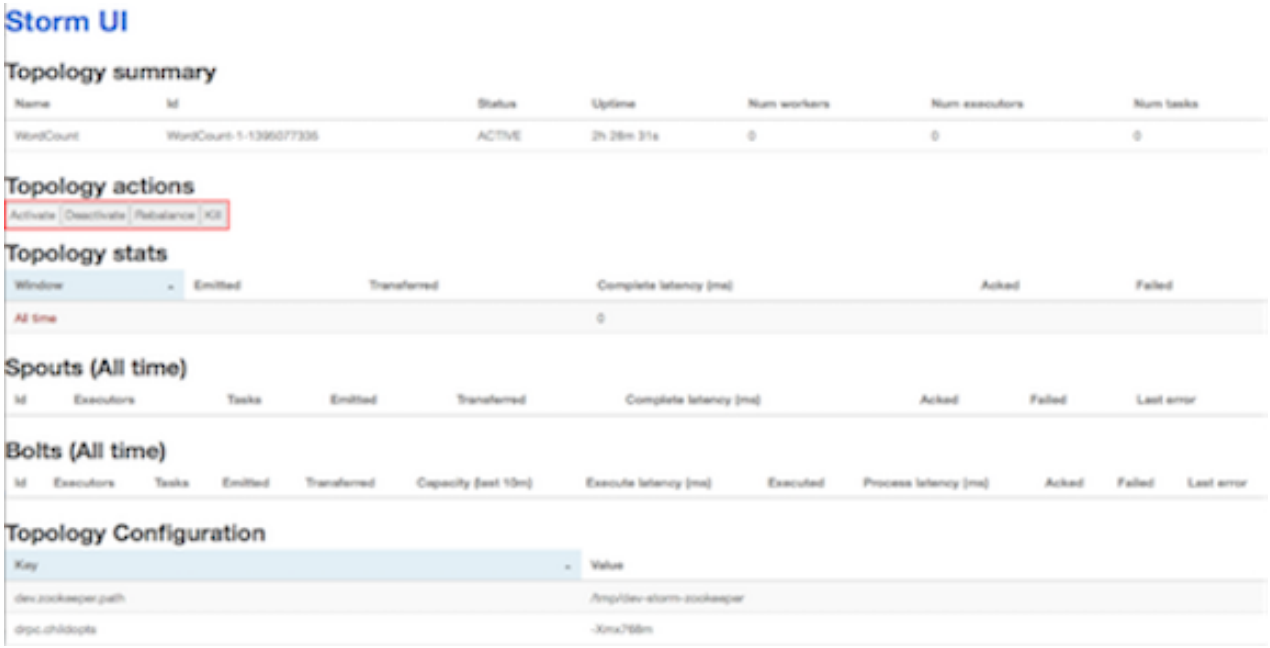


Storm administrators use the Storm UI to perform the following administrative actions:

Table 2: Topology Administrative Actions

Topology Administrative Action	Description
Activate	Return a topology to active status after it has been deactivated.
Deactivate	Set the status of a topology to inactive. Topology uptime is not affected by deactivation.
Rebalance	Dynamically increase or decrease the number of worker processes and/or executors. The administrator does not need to restart the cluster or the topology.
Kill	Stop the topology and remove it from Apache Storm. The topology no longer appears in the Storm UI, and the administrator must deploy the application again to activate it.

Click any topology in the Topology Summary section to launch the Topology Summary page. To perform any of the topology actions in the preceding table, you can click the corresponding button (shown in the following image):



The "Executors" field in the Spouts and Bolts sections shows all running Storm threads, including the host and port. If a bolt is experiencing latency issues, review this field to determine which executor has reached capacity. Click the port number to display the log file for the corresponding executor.

Monitoring and Debugging an Apache Storm Topology

Debugging Storm applications can be challenging due to the number of moving parts across a large number of nodes in a cluster. Tracing failures to a particular component or a node in the system requires collection and analysis of log files and analysis of debug/trace processes running in the cluster. The following subsections describe features designed to facilitate the process of debugging a storm topology.

Enabling Dynamic Log Levels

Storm allows users and administrators to dynamically change the log level settings of a running topology. You can change log level settings from either the Storm UI or the command line. No Storm processes need to be restarted for the settings to take effect. The resulting log files are searchable from the Storm UI and logviewer service.

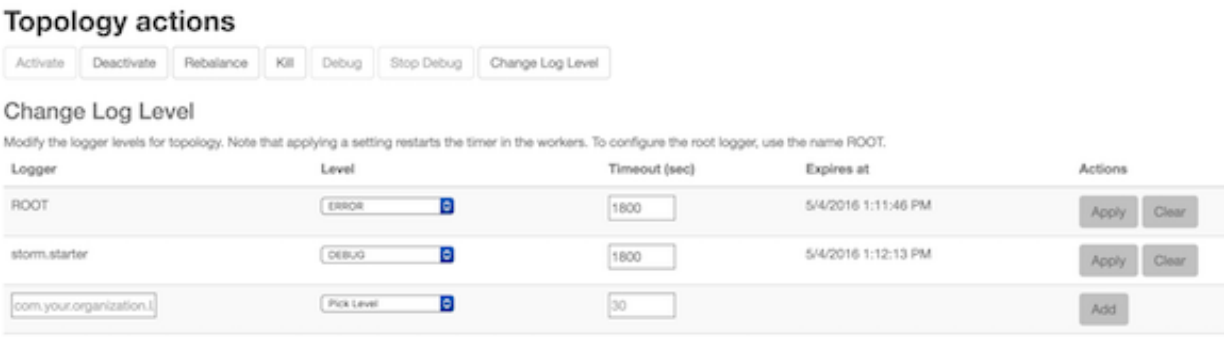
Standard log4j levels include DEBUG, INFO, WARN, ERROR, and FATAL, specifying logging of coarse or finer-grained levels of informational messages and error events. Inheritance is consistent with log4j behavior. For example, if you set the log level of a parent logger, the child loggers start using that level (unless the children have a more restrictive level defined for them).

Setting and Clearing Log Levels Using the Storm UI

To set log level from the Storm UI:

Procedure

1. Click on a running topology.
2. Click on “Change Log Level” in the Topology Actions section:



3. For an existing logger, select the desired log level for the logger. Alternately, add a logger and set the desired log level.
4. Optionally, specify a timeout value in seconds, after which changes will be reverted automatically. Specify 0 if no timeout is needed.
5. Click "Apply".

What to do next

The preceding example sets the log4j log level to ERROR for the root logger, and to DEBUG for storm.starter. Logging for the root logger will be limited to error events, and finer-grained informational events (useful for debugging the application) will be recorded for storm.starter packages.

To clear (reset) a log level setting using the Storm UI, click on the “Clear” button. This reverts the log level back to what it was before you added the setting. The log level line will disappear from the UI.

Setting and Clearing Log Levels Using the CLI

To set log level from the command line, use the following command:

```
./bin/storm set_log_level [topology name] -l [logger name]=[LEVEL]:[TIMEOUT]
```

The following example sets the ROOT logger to DEBUG for 30 seconds:

```
./bin/storm set_log_level my_topology -l ROOT=DEBUG:30
```

To clear (reset) the log level using the CLI, use the following command. This reverts the log level back to what it was before you added the setting.

```
./bin/storm set_log_level [topology name] -r [logger name]
```

The following example clears the ROOT logger dynamic log level, resetting it to its original value:

```
./bin/storm set_log_level my_topology -r ROOT
```

For more information, see Apache STORM-412.

Enabling Topology Event Logging

The topology event inspector lets you view tuples as they flow through different stages of a Storm topology. This tool is useful for inspecting tuples emitted from a spout or a bolt in the topology pipeline while the topology is running; you do not need to stop or redeploy the topology to use the event inspector. The normal flow of tuples from spouts to bolts is not affected by turning on event logging.

Configuring Topology Event Logging

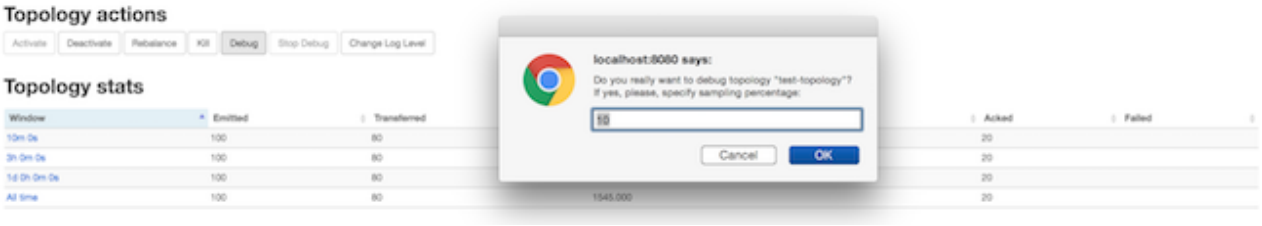
Event logging sends events (tuples) from each component to an internal eventlogger bolt.

Event logging is disabled by default, due to a slight performance degradation associated with eventlogger tasks.

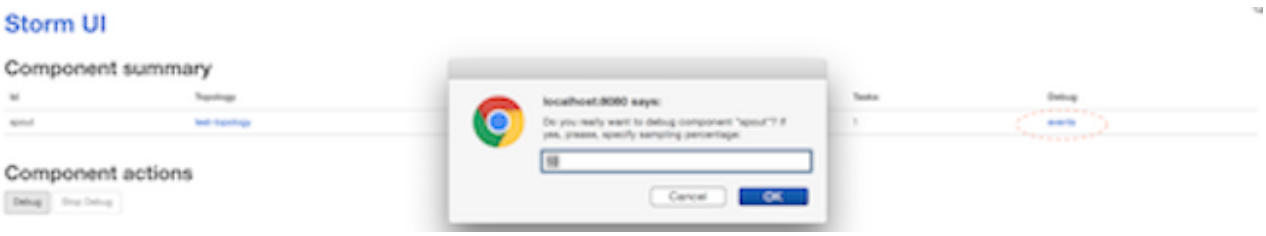
To enable event logging, set the topology.eventlogger.executors property to a non-zero value when submitting your topology. You can set the property globally in the storm.yaml file, or use the command line. For more information about topology.eventlogger.executors and other property settings, see *Configuring Apache Storm for Production Environments.*

Enabling Event Logging

To log events for an entire topology, click the "Debug" button under "Topology actions" in the topology view. This setting logs tuples from all spouts and bolts in a topology at the specified sampling percentage.



To log events at a specific spout or bolt level, navigate to the corresponding component page and click "Debug" under component actions:



Viewing Event Logs

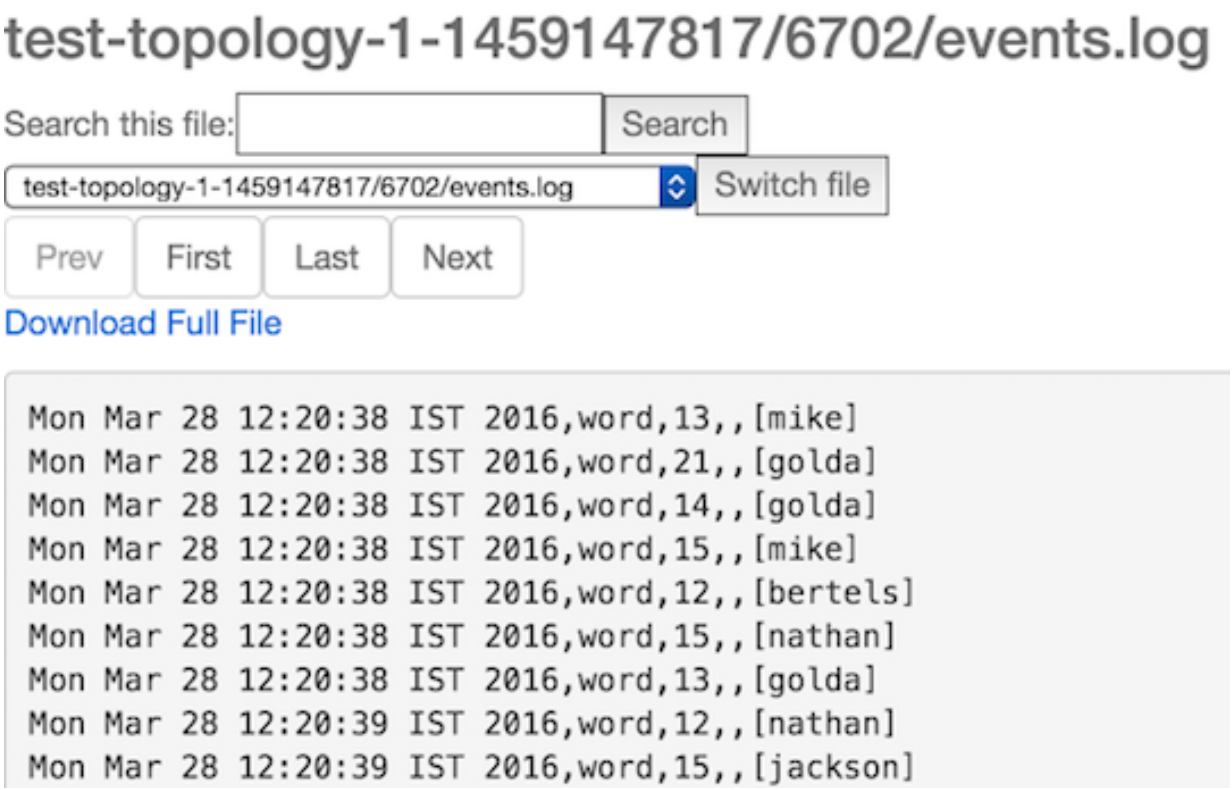
Before you begin

The Storm "logviewer" process should be running so that you can view the logged tuples. If it is not already running, start the log viewer by running the following command from the storm installation directory:

```
bin/storm logviewer
```

Procedure

1. From the Storm UI, navigate to the specific spout or bolt component page.
2. Click on the "events" link in the Debug column of the component summary. This will open a view similar to the following:



Each line in the event log contains an entry corresponding to a tuple emitted from a specific spout or bolt, presented in a comma-separated format:

```
Timestamp, Component name, Component task-id, MessageId (incase of
 anchoring), List of emitted values
```

3. Navigate between different pages to view logged events.

Accessing Event Logs on a Secure Cluster

If you want to view logs in secure mode, ensure that your browser is configured to use authentication with all supervisor nodes running logviewer. This process is similar to the process used to configure access to the Storm UI on a secure cluster (described in *Configuring the Storm UI*).

About this task

Add domains to the white list by setting network.negotiate-auth.trusted-uris to a comma-separated list containing one or more domain names and URLs. For example, the following steps configure Firefox to use authentication with two nodes:

Procedure

1. Go to the about:config configuration page.
2. Search for the network.negotiate-auth.trusted-uris configuration option.
3. Double-click on the option.
4. An "Enter string value" dialog box opens.
5. In this box, enter the values http://node1.example.com, http://node2.example.com.
6. Click OK to finish.
7. Close and relaunch the browser.

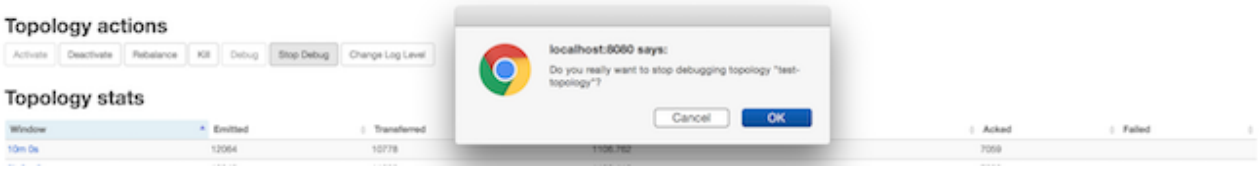
What to do next

If your cluster is not managed by Ambari, refer to the UI/Logviewer section of Apache Storm security documentation for additional configuration guidelines.

Disabling Event Logs

To disable event logging for a component or for a topology, click "Stop Debug" under the Component actions or Topology actions page (respectively) in the Storm UI.

The following example disables topology "test-topology":



Extending Event Logging

The Storm eventlogger bolt uses the IEventLogger interface to log events. The default implementation is aFileBasedEventLogger, which logs events to a log file at logs/workers-artifacts/<topology-id>/<worker-port>/events.log.

To extend event logging functionality (for example, to build a search index or log events to a database), add an alternate implementation of the IEventLogger interface.

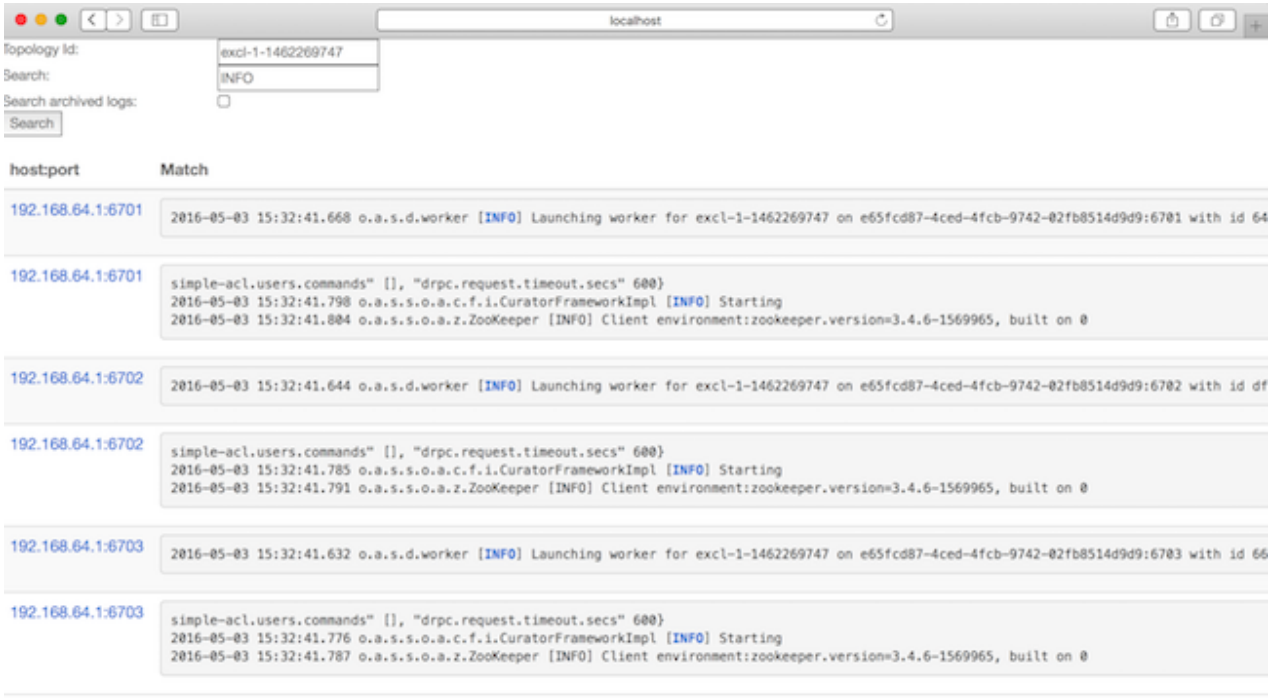
```
/**
      * EventLogger interface for logging the event info to a sink like log
 file or db
      * for inspecting the events via UI for debugging.
      */public interface IEventLogger {
          void prepare(Map stormConf, TopologyContext context);
          /**
          * Invoked when the {@link EventLoggerBolt} receives a tuple from
 the spouts or bolts that
          * have event logging enabled.
          *
          * @param e the event
          */
          void log(EventInfo e);
          /**
          * Invoked when the event logger bolt is cleaned up
          */
          void close();
          }
```

See JIRA STORM-954 for more details.

Enabling Distributed Log Search

The distributed log search capability allows users to search across log files (including archived logs) to find information and events for a specific topology. Results include matches from all supervisor nodes.

This feature is useful when searching for patterns across workers or supervisors of a topology. (A similar log search is supported within specific worker log files via the Storm UI.)



For more information about log search capabilities, see Apache JIRA STORM-902.

Dynamic Worker Profiling

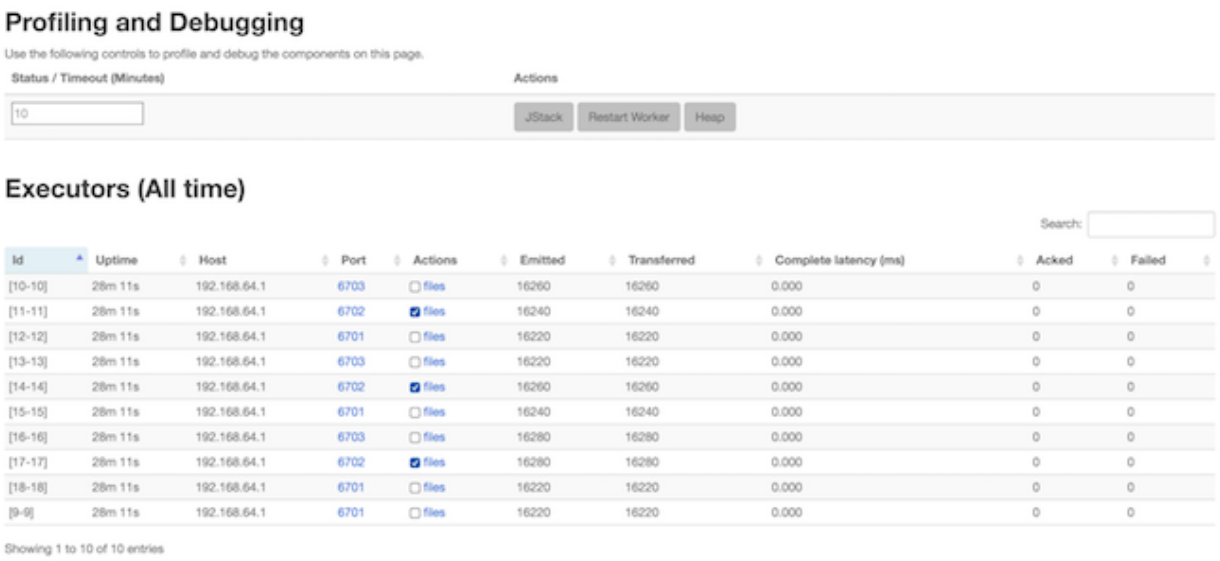
You can request the following types of worker profile data directly from the Storm UI , without restarting the topologies:

About this task

- Heap dumps
- JStack output
- JProfile recordings

Procedure

1. Navigate to the “Profiling and Debugging” section on the Spout/Bolt component summary page. There you will see buttons to request JStack output or generate a Heap dump:



Note that you can also restart worker processes from this page.

2. To view output, click the “files” link under “Actions”.

3. To download output for offline analysis, click the associated link under the "Actions" column on the Profiling and Debugging page, or click "Download Full File" on the output page.

What to do next

See Apache JIRA STORM-1157 for more information.

Tuning an Apache Storm Topology

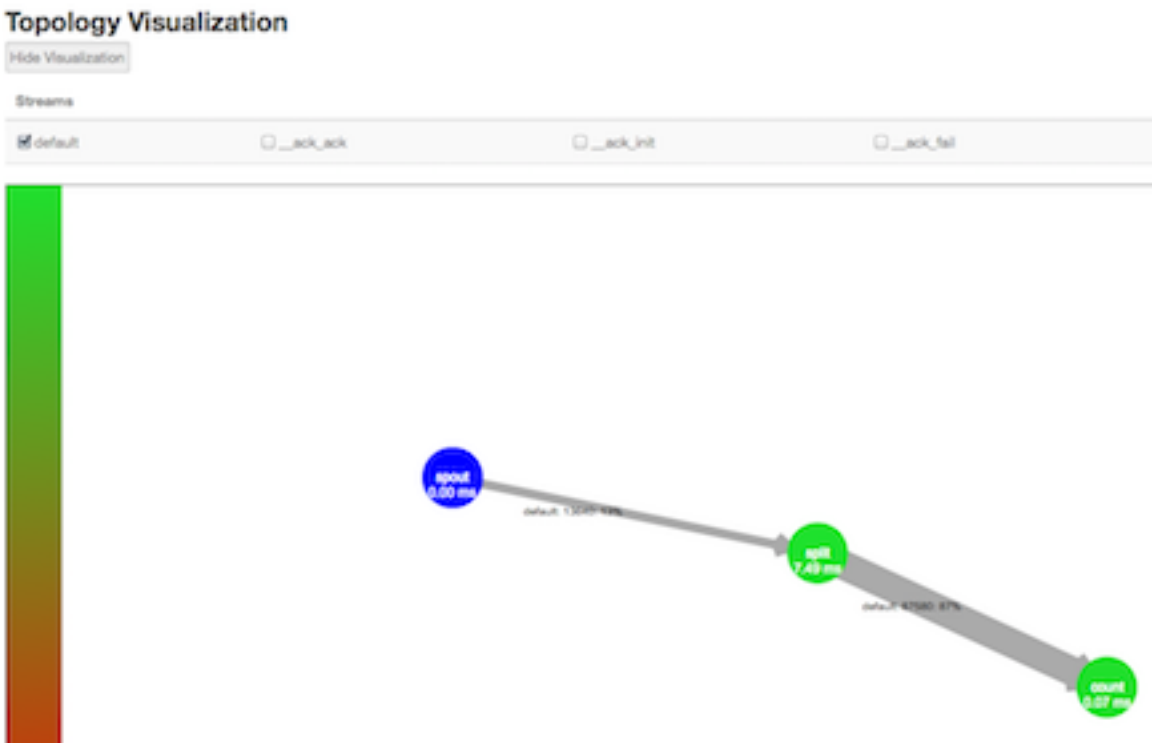
Because Storm topologies operate on streaming data (rather than data at rest, as in HDFS) they are sensitive to data sources. When tuning Storm topologies, consider the following questions:

About this task

- What are my data sources?
- At what rate do these data sources deliver messages?
- What size are the messages?
- What is my slowest data sink?

Procedure

1. Identify the bottleneck.
   a. In the Storm UI, click Show Visualization to display a visual representation of your topology and find the data bottleneck in your Storm application.
      - Thicker lines between components denote larger data flows.
      - A blue component represents the the first component in the topology, such as the spout below from the WordCountTopology included with storm-starter.
      - The color of the other topology components indicates whether the component is exceeding cluster capacity: red components denote a data bottleneck and green components indicate components operating within capacity.



Note:

In addition to bolts defined in your topology, Storm uses its own bolts to perform background work when a topology component acknowledges that it either succeeded or failed to process a tuple. The

names of these "acker" bolts are prefixed with an underscore in the visualization, but they do not appear in the default view.

To display component-specific data about successful acknowledgements, select the _ack_ack checkbox. To display component-specific data about failed acknowledgements, select the _ack_fail checkbox.

b. To verify that you have found the topology bottleneck, rewrite the execute() method of the target bolt or spout so that it performs no operations. If the performance of the topology improves, you have found the bottleneck.

Alternately, turn off each topology component, one at a time, to find the component responsible for the bottleneck.

2. Refer to "Performance Guidelines for Developing a Storm Topology" for several performance-related development guidelines.
3. Adjust topology configuration settings.
4. Increase the parallelism for the target spout or bolt. Parallelism units are a useful conceptual tool for determining how to distribute processing tasks across a distributed application.

   Hortonworks recommends using the following calculation to determine the total number of parallelism units for a topology.

   ```
   (number of worker nodes in cluster * number cores per worker node) -
    (number of acker tasks)
   ```

   Acker tasks are topology components that acknowledge a successfully processed tuple.

   The following example assumes a Storm cluster with ten worker nodes, 16 CPU cores per worker node, and ten acker tasks in the topology. This Storm topology has 150 total parallelism units:

   ```
   (10 * 16) - 10 = 150
   ```

   Storm developers can mitigate the increased processing load associated with data persistence operations, such as writing to HDFS and generating reports, by distributing the most parallelism units to topology components that perform data persistence operations.

What to do next

In a Storm cluster, most of the computational burden typically falls on the Supervisor and Worker nodes. The Nimbus node usually has a lighter load. For this reason, Hortonworks recommends that organizations save their hardware resources for the relatively burdened Supervisor and Worker nodes.

The performance of a Storm topology degrades when it cannot ingest data fast enough to keep up with the data source. The velocity of incoming streaming data changes over time. When the data flow of the source exceeds what the topology can process, memory buffers fill up. The topology suffers frequent timeouts and must replay tuples to process them.

Use the following techniques to identify and overcome poor topology performance due to mismatched data flow rates between source and application: