[Overview 2](#_Toc381800974)

[Pre-requisites 3](#_Toc381800975)

[Configuration 3](#_Toc381800976)

[Source Code: 3](#_Toc381800977)

[Algorithm 3](#_Toc381800978)

[Design 3](#_Toc381800979)

[DAG 3](#_Toc381800980)

[Operators 4](#_Toc381800981)

[RandomEventGenerator: 4](#_Toc381800982)

[PubSubWebSocketInputOperator: 4](#_Toc381800983)

[PhoneMovementGenerator: 4](#_Toc381800984)

[PubSubWebSocketOutputOperator: 5](#_Toc381800985)

[Functionality 5](#_Toc381800986)

[Scaling up and down 5](#_Toc381800987)

[Launching the application 5](#_Toc381800988)

[Monitoring the application 6](#_Toc381800989)

[Operations 15](#_Toc381800990)

[Stopping the application 18](#_Toc381800991)

[Conclusions 19](#_Toc381800992)

Application ‘Mobile’

# Overview

The ‘Mobile’ application shows us how DataTorrent platform can process huge amount of data in real time. It is a way to demonstrate the ability of the DataTorrent platform to scale both ways i.e. scale-up when load increases and scale down when the load reduces. This scaling is dynamic and is completely taken care of by the platform based on the amount of data to be processed.

The platform decides at run-time whether it needs to scale up. Similarly it also can scale down in run-time.

This demo simulates large number of cell phones in the range of 40K events/sec to 200K events/sec and tracks a given cell number across cell towers.  In reality this demo can be scaled to millions of events by simply adding more partitions in response to the load. It also displays the changing locations of the cell number on a google map.

It showcases the ability of the platform to scale up and down as the phone numbers generated increase and decrease respectively. If the tuples processed per second by an operator increase beyond a certain limit, more partitions of the operator gets deployed. The limit on tuples processed per second on the operator is set artificially to demonstrate dynamic scaling up and down. This number is not indicative of actual performance benchmark of the operator.

If the tuples processed per second drops below a certain limit, the platform merges the operators until the partition count drops down to the original. The load is varied by firing a command on the CLI and changing the value of a property. Dynamic partition up and down is fully stateful, i.e. the state is transferred to the partitions. Users can customize this state transfer if need be.

A large number of cell phone numbers are generated. The application also demos the ability of the DataTorrent platform to accept queries at runtime. A user can add a phone number at runtime and can see the same being tracked right away as a part of the application.

The aim is to demonstrate the following abilities –

* Entering a query dynamically:

The phone numbers are added to locate its gps in run time.

* Changing functionality dynamically:

The load is changed by making functional changes on the load generator operator

* Auto Scale up/Down with load:

Operator increases and decreases the number of partitions as per load

# Pre-requisites

There are no special pre-requisites needed to run the ‘Mobile’ demo application on the DataTorrent platform.

# Configuration

There is no particular configuration that needs to be done to run the ‘Mobile’ demo application on the DataTorrent platform.

# Source Code:

This application is a part of the open sourced ‘Malhar’ repository. Malhar repository contains open source operator and codec library that can be used with the DataTorrent platform to build Real-time streaming applications and demo applications such as this one.

# Algorithm

The ‘Mobile’ demo application written for DataTorrent platform –

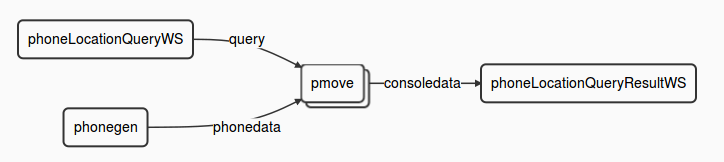
1. Generates random input to simulate incoming data tuples as if they were coming from mobile devices. This simulates the location of each mobile device.
2. Generate a new location
3. Increase the number of the tuples processed per second by the ‘pmove’ operator beyond 30,000. The number of partitions of the ‘pmove’ operator that get deployed increases, until each of the partition processes between 10000 to 30000 tuples per second. The split in partitions is stateful as the original partition state is split into two new partitions.
4. If the tuples processed per second drops below 10,000, the platform merges the operators until the partition count drops down to the original. The merge is fully stateful as states of each partition are transferred to the merged partition.
5. The number of partitions reduces dynamically due to reduction in load.

Note : The ‘tuplesBlast’ property, is used to generate a large number of cell phone numbers.

# Design

The DAG for the application is shown below followed by a description of the operators.

## DAG



## Operators

The ‘Mobile’ demo application involves multiple computational steps and the DataTorrent platform provides operators to perform each of these. Specifically, this application consists of the following operators –

|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Name of the operator in DAG** | **Library** |
| [RandomEventGenerator](#_RandomEventGenerator:) | phonegen | lib/testbench/randomeventgenerator |
| [PubSubWebSocketInputOperator](#PubSubWebSocketInputOperator) | phoneLocationQueryWS | lib/io/ PubSubWebSocketInputOperator |
| [PhoneMovementGenerator](#PhoneMovementGenerator) | pmove | Custom operator |
| [PubSubWebSocketOutputOperator](#PubSubWebSocketOutputOperator) | phoneLocationQueryResultWS | lib/io/ PubSubWebSocketOutputOperator |

### [RandomEventGenerator](https://github.com/DataTorrent/Malhar/library/src/main/java/com/datatorrent/lib/testbench/RandomEventGenerator.java):

This is the first operator in the DAG and performs the role of simulating an input received by the DAG.

This operator generates tuples using random numbers and keeps emitting them on its output port.

The name of the instance of this operator is ‘phonegen’. It generates a random input between a range of 0 and 30,000

### 

### [PubSubWebSocketInputOperator](https://github.com/DataTorrent/Malhar/blob/master/library/src/main/java/com/datatorrent/lib/io/PubSubWebSocketInputOperator.java):

This is an extension of the web socket input operator and keeps firing a query about the location of the phone. The query is dynamically entered by the user. This query is a phone number to be tracked. The application accepts this query input and starts tracking the phone number.

### 

### [PhoneMovementGenerator](https://github.com/DataTorrent/master/demos/src/main/java/com/datatorrent/demos/mobile/PhoneMovementGenerator.java):

This operator is used to generate the GPS locations for the phone numbers specified. The range of phone numbers or a specific phone number can be set for which the GPS locations will be generated. It supports querying the locations of a given phone number. This operator generates a new location for a given phone number. This is equivalent of tracking of a number to identify its co-ordinates. The new location is then passed on to be displayed to the end user.

### 

### [PubSubWebSocketOutputOperator](https://github.com/DataTorrent/Malhar/library/src/main/java/com/datatorrent/lib/io/PubSubWebSocketOutputOperator.java):

This is an extension of the web socket output operator. Based on the query entered by the user, a phone number is selected to be tracked. The location of this tracked number is passed as input to this operator which then displays the same on the screen.

|  |  |
| --- | --- |
| **Stream** | **Description** |
| Phonedata | This stream connects the data generated by the phonegen operator to the pmove operators input |
| Query | This is a query that comes in as input and is passed to the pmove operator |
| Consoledata | This is the location of the phone numbers being tracked which is displayed on the console |

# Functionality

The detailed functionality of each operator is covered above. Note that in this application, we do use operators which are stateful as well as some which do not maintain a state.

|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Stateful** | **Supports partitions** |
| [RandomEventGenerator](#_RandomEventGenerator:) | - | Y |
| [PhoneMovementGenerator](#PhoneMovementGenerator) | Y | Y |
| [PubSubWebSocketOutputOperator](#PubSubWebSocketOutputOperator) | - | Y |
| [PubSubWebSocketInputOperator](#PubSubWebSocketInputOperator) | - | Y |

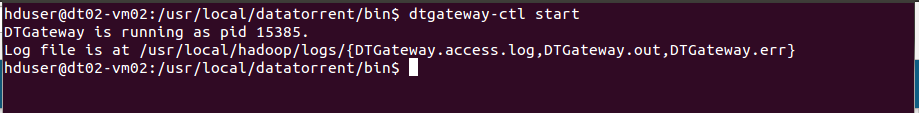
## Scaling up and down

There are a lot of applications which have a very high rate of incoming data (of the order of millions/second) and the DataTorrent platform can scale up and down based on the change in the load and at a very high throughput. The demo also displays this behavior of the DataTorrent platform.

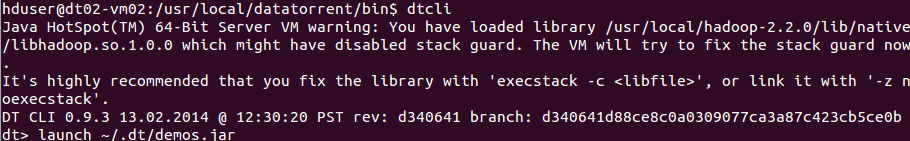
# Launching the application

In order to be able to launch sample Demo Applications, please make sure you have completed following steps,

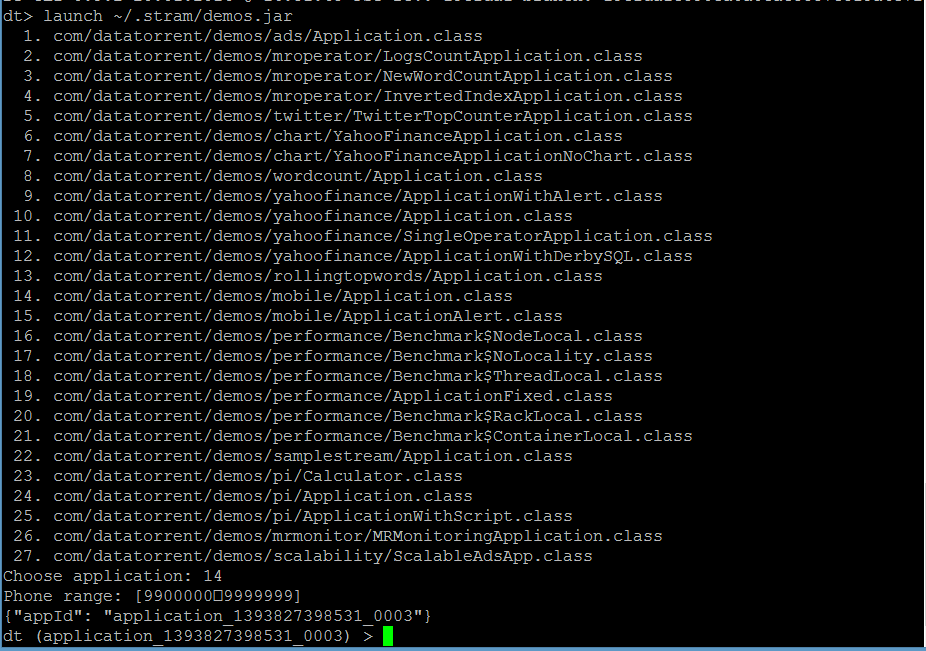
1. Start the DataTorrent Gateway process. This can be done from the command line as –



1. Launch DataTorrent Command Line Interface (dtcli)



1. Launch demo application Jar. (Here, we assume that the Hadoop services are up and running and the demos are being launched in a cluster mode.)



The application has now been launched and is in the running state.

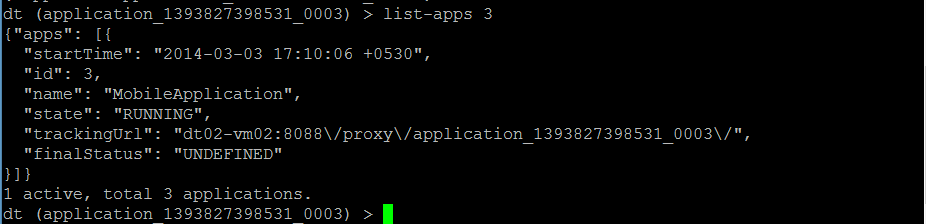
Please refer to DataTorrent Quick Start Guide for additional details.

# Monitoring the application

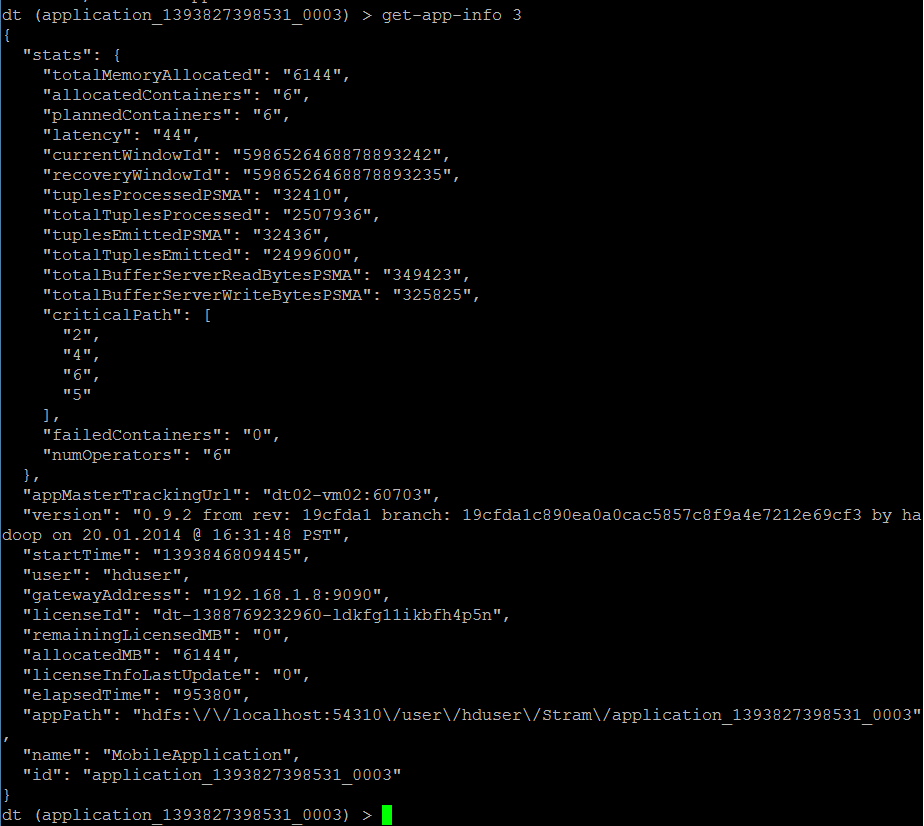
Once launched, the application can be monitored/managed from either the command line viz. ‘dtcli’ or the GUI viz. DataTorrent Gateway Console. Following are some of the sample operations that are typically performed while monitoring the application –

1. List the application -

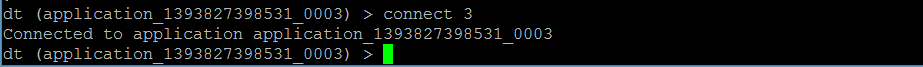
#### Using ‘dtcli’



1. Get detailed information about a given application –



1. Connect to the application –

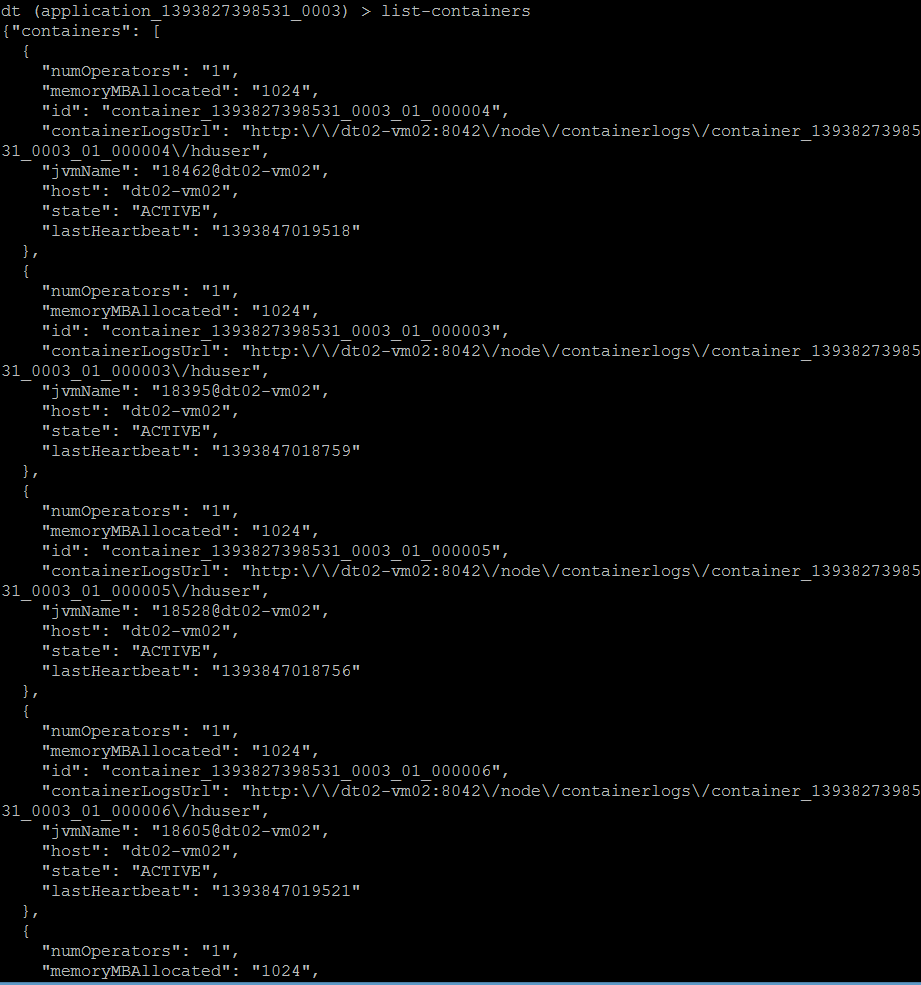


1. Show logical plan



1. List the containers :

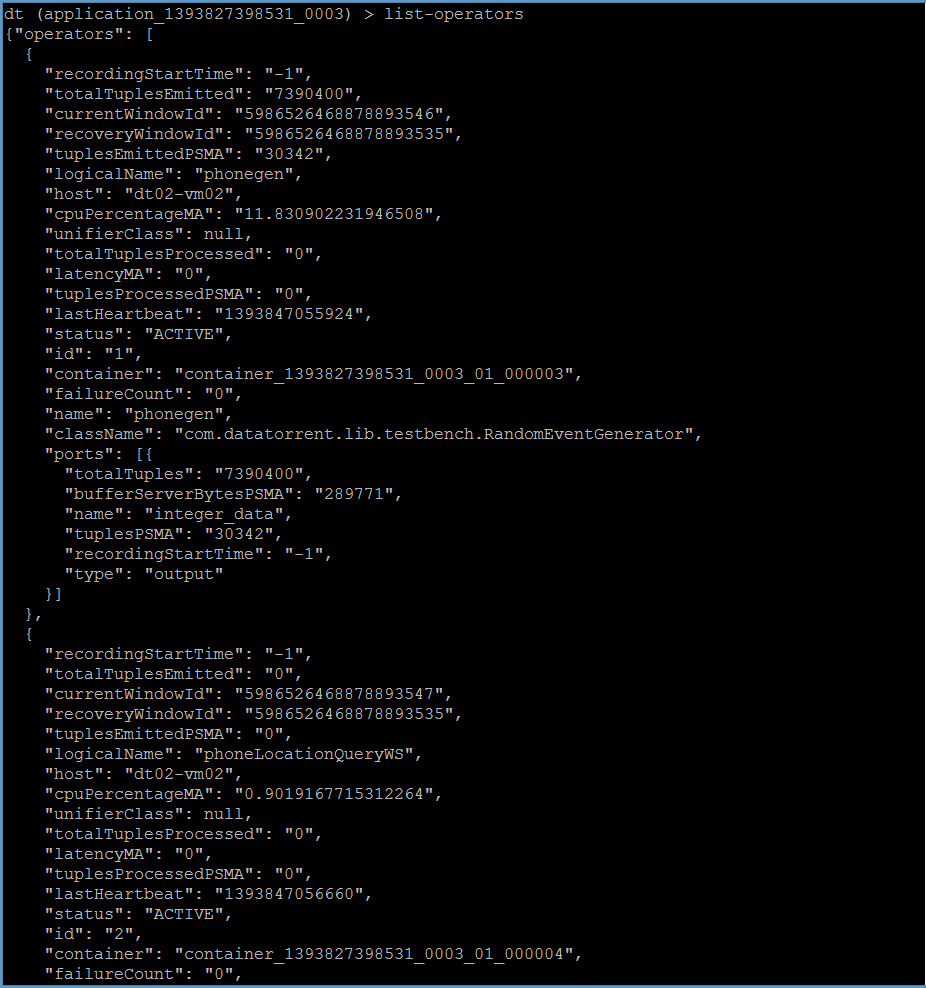
This lists the containers for the application that you are connected to, at this point in time.

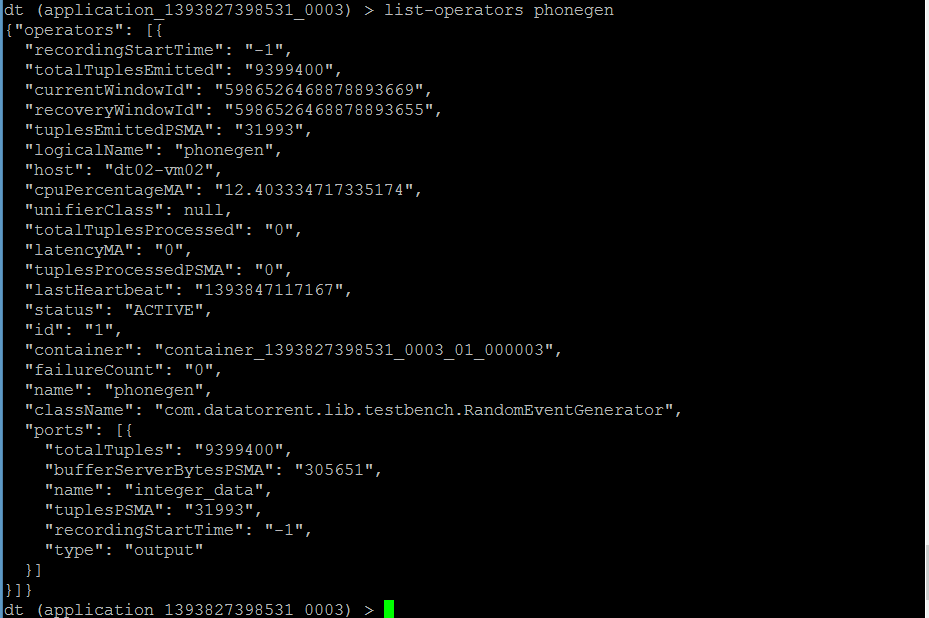


1. List all the operators –

The operators being used in the application are listed by the list-operators command. You can pass a pattern to this command to filter the operators which you want to see. For example, in the second screen shot below, the command has been passed a pattern “phonegen”. This will list all the operators which are named ‘phonegen’.

Not passing a pattern will, by default list all the operators.





1. Get application attributes :



1. Similarly the user can use other dtcli to monitor/manage the application.
2. Scaling up and scaling down

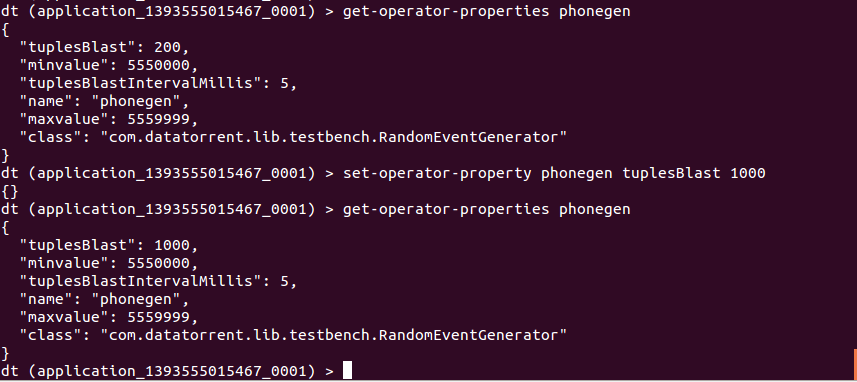
To simulate the changes in the amount of data to be processed in real time, we can use the ‘dtcli’.

The ‘tuplesBlast’ property is used to generate a large number of cell phone numbers.

The load is changed by making changes on the load generator operator (phonegen). This is achieved by changing the value of the ‘tuplesblast’ attribute.

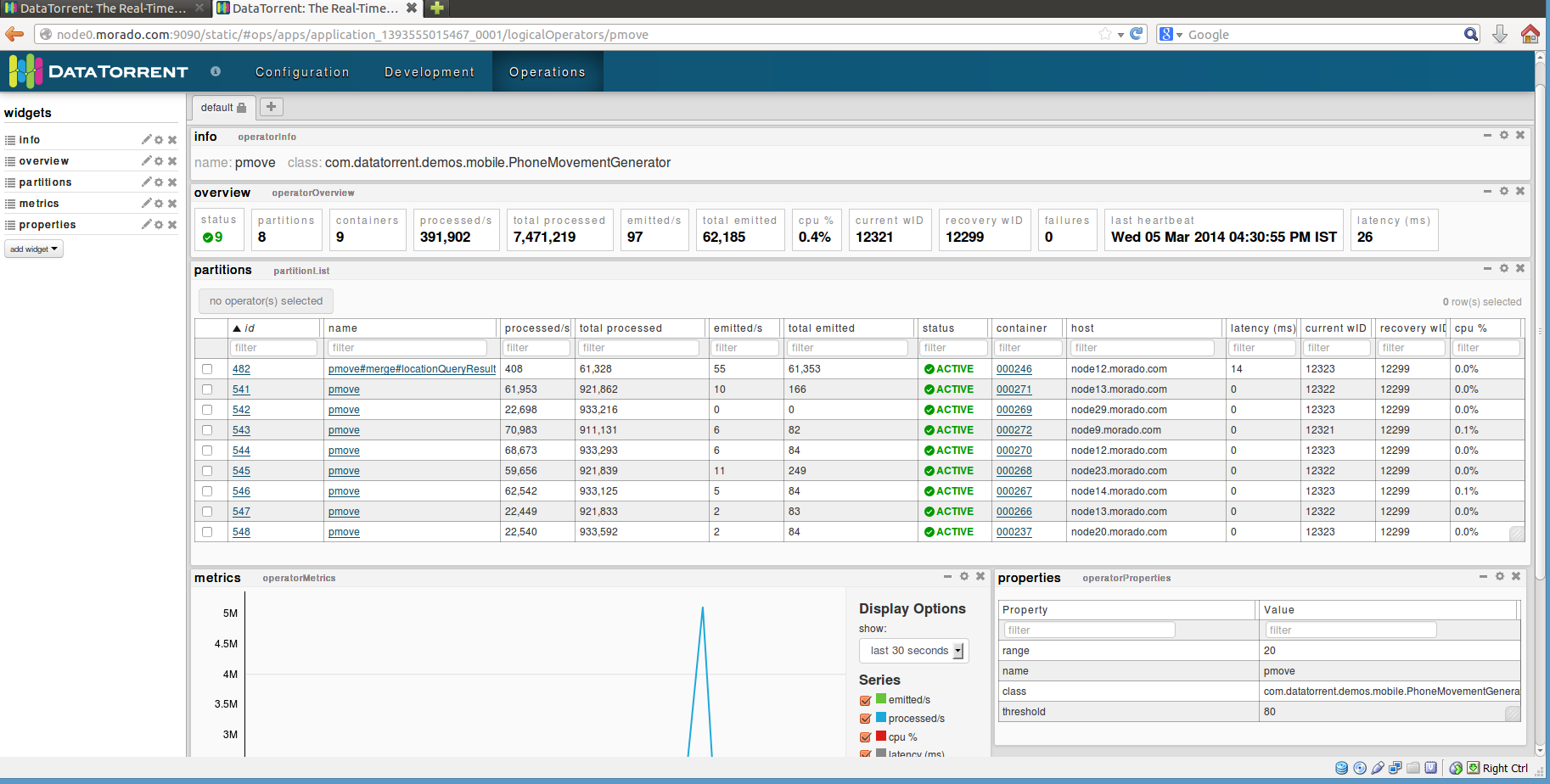
In order to increase the load, give the following command from the ‘dtcli’ -

***dtcli> set-operator-property phonegen tuplesBlast 1000***



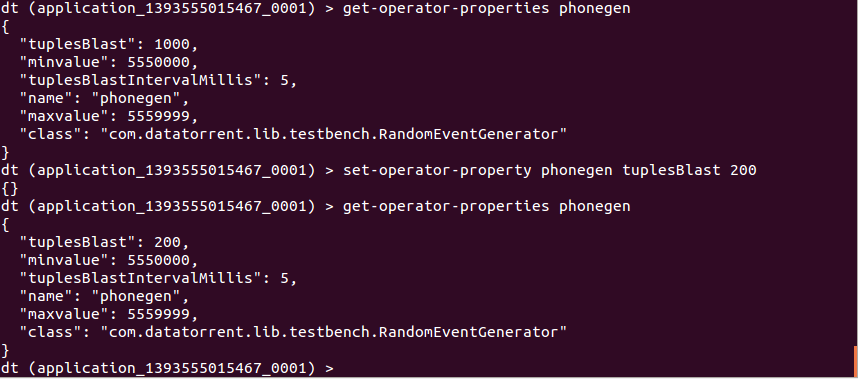
This will result in a large numbers of phone numbers being generated and in order that their movement be tracked, the user will see the operator ‘pmove’ being partitioned into more than one partitions. This clearly displays how the DataTorrent platform dynamically scales up to handle the increased load.

The increase in the number of partitions can be seen on the gateway console as shown in the screenshot below -

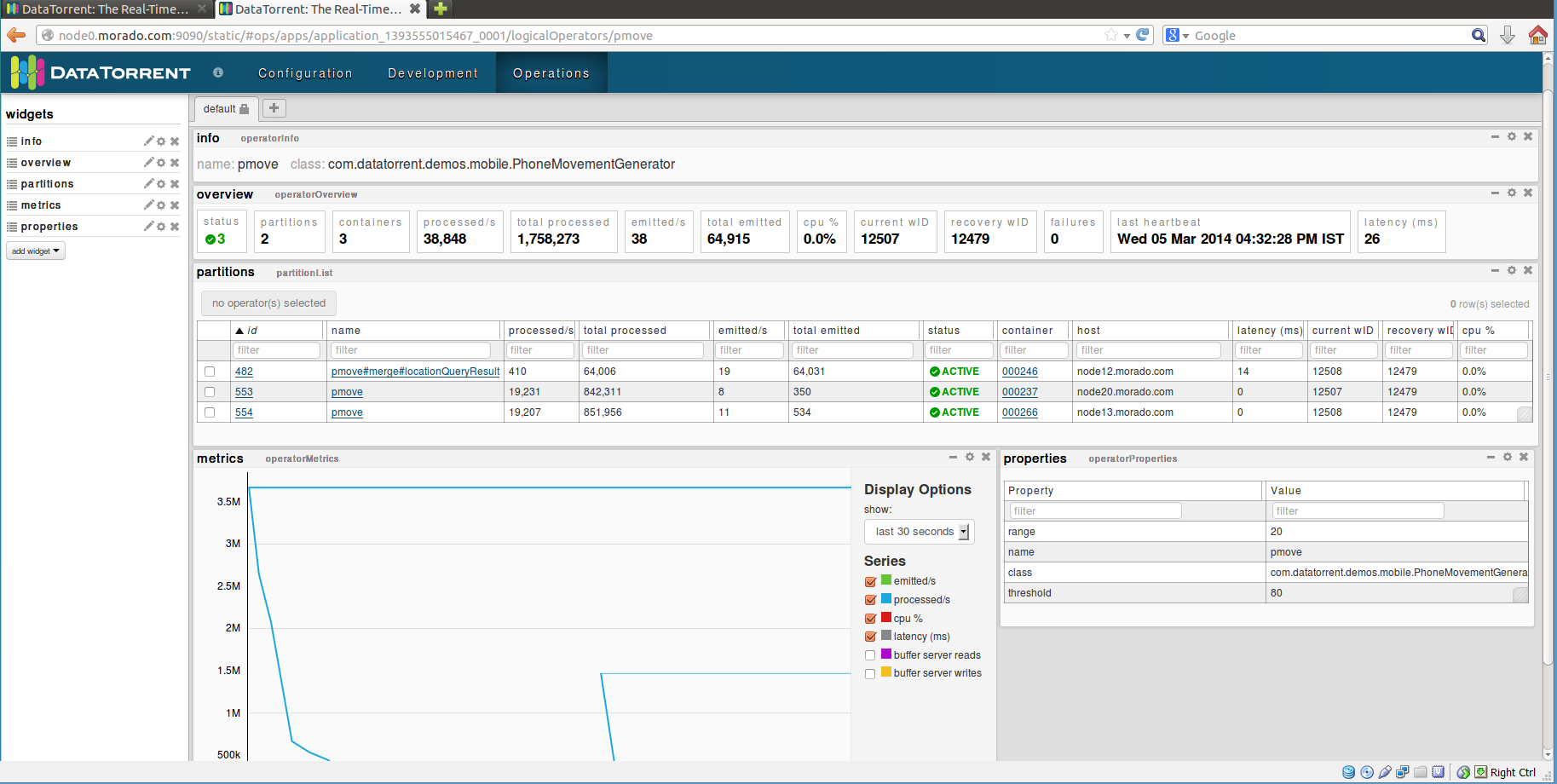


Similarly, to view how the platform releases the resources and scales down when the load decreases, give the following command form the ‘dtcli’ -

***dtcli> set-operator-property phonegen tuplesBlast 200***



This will result in the number of phone numbers being generated going down i.e. effectively reducing the load. A result of this will be that the ‘pmove’ operator, which is having multiple partitions, will now be seen having lesser number of partitions. Note that the number of partitions have gone down (as seen in the screenshot below) after setting the ‘tuplesBlast’ to 200.



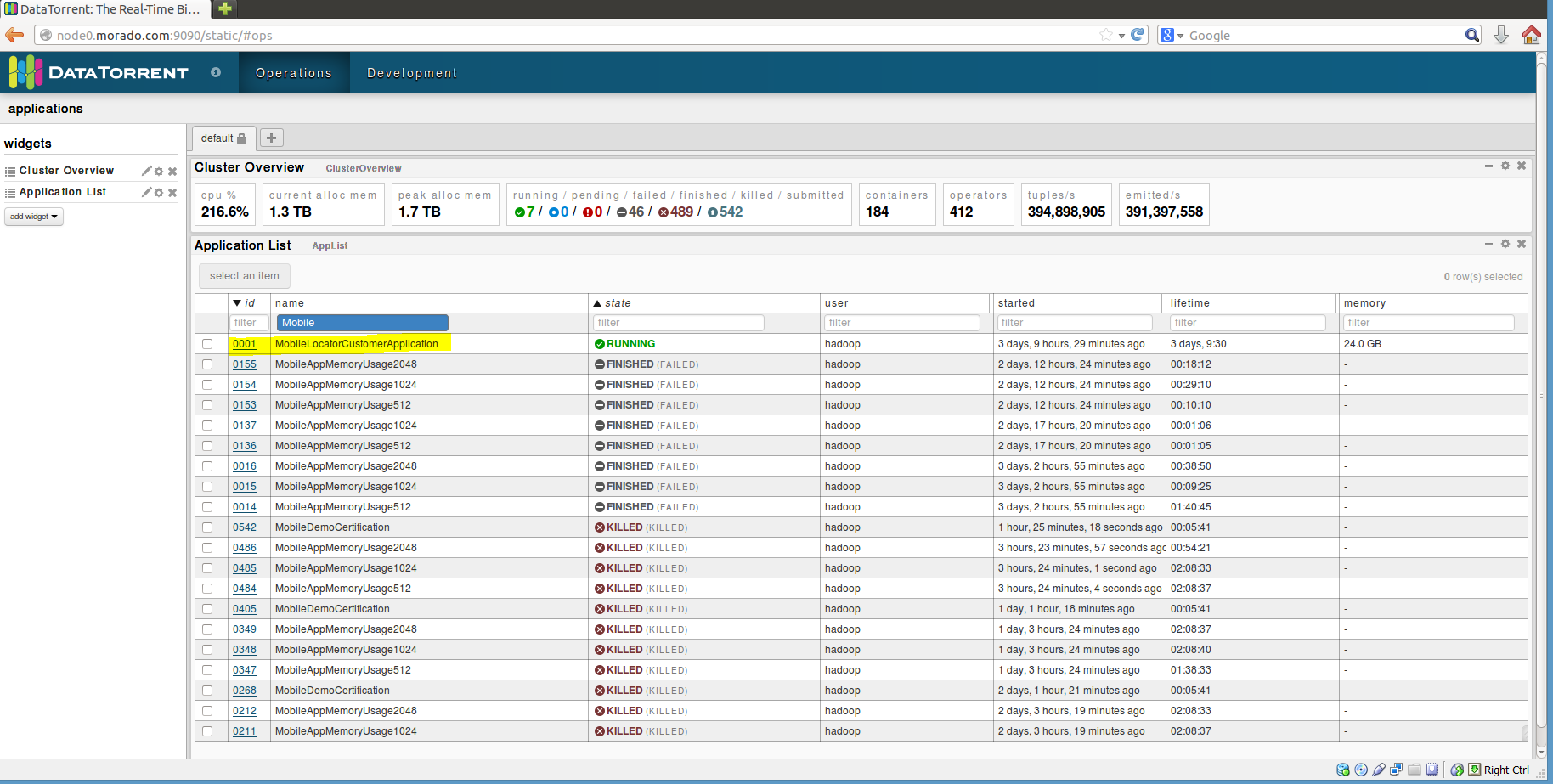
The actual number of partitions needed will depend on the value being specified for ‘tuplesBlast’

#### Using ‘Gateway Console’

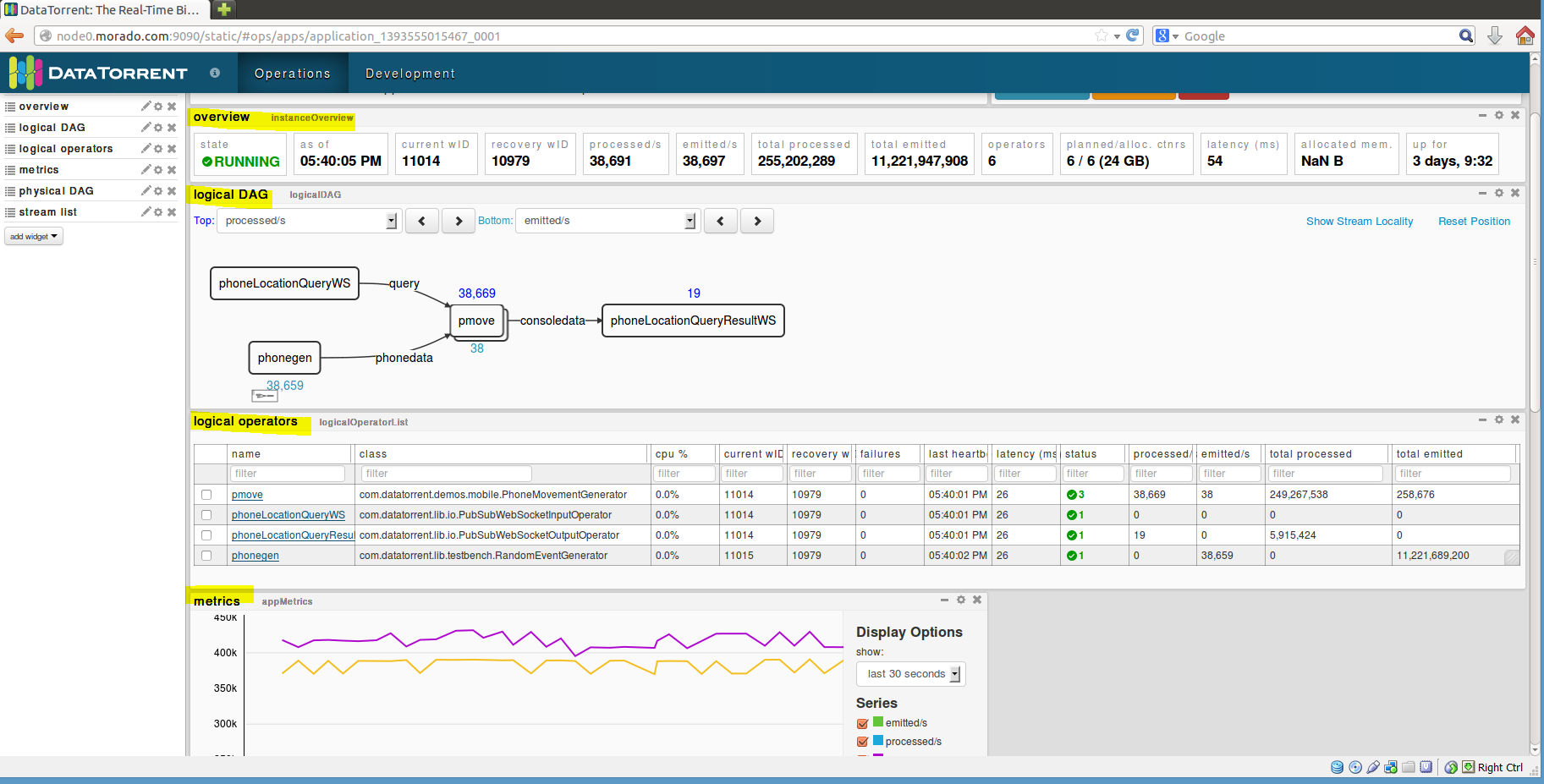
# Operations

1. On the Gateway console, you will see the apps as below –

Here, we have filtered the applications on the name, by specifying the string “Mobile”. This will list only those applications which have the specified string viz. “Mobile” in their name.



1. In order to get detailed about the application, click on ‘ID’ for the application and you will get to the screen below –

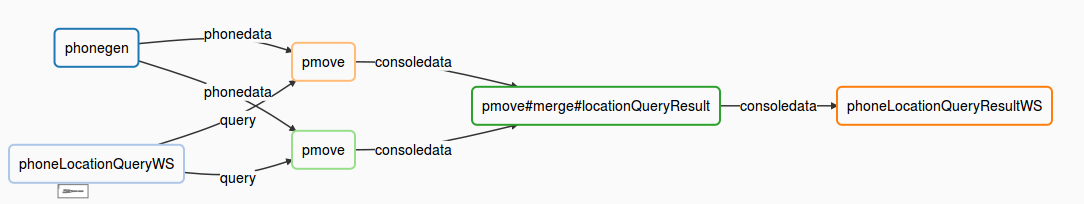


As you can see, it shows you a lot of details about the application, such as –

(You may have to scroll down for more details mentioned below)

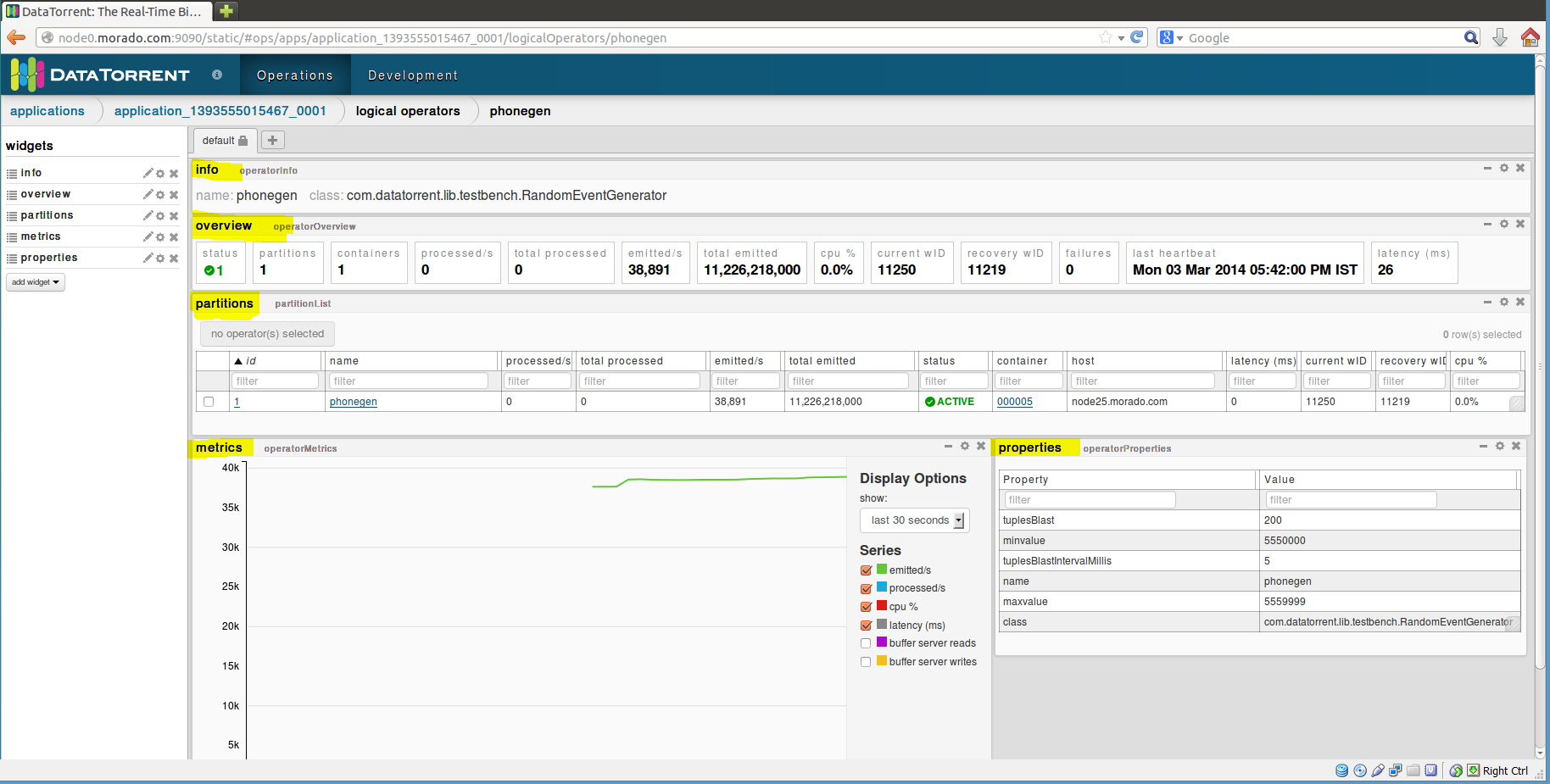
* Overview
* Logical DAG
* Logical operators, etc.
* Physical DAG

As an example, following is a screenshot captured showing the physical DAG –



Additionally, you can select additional widgets as well as rearrange the widgets as per your wish.

1. Clicking on any of the operators will give you the details for that particular operator. For example, the screenshot below is for the ‘phonegen’ operator.

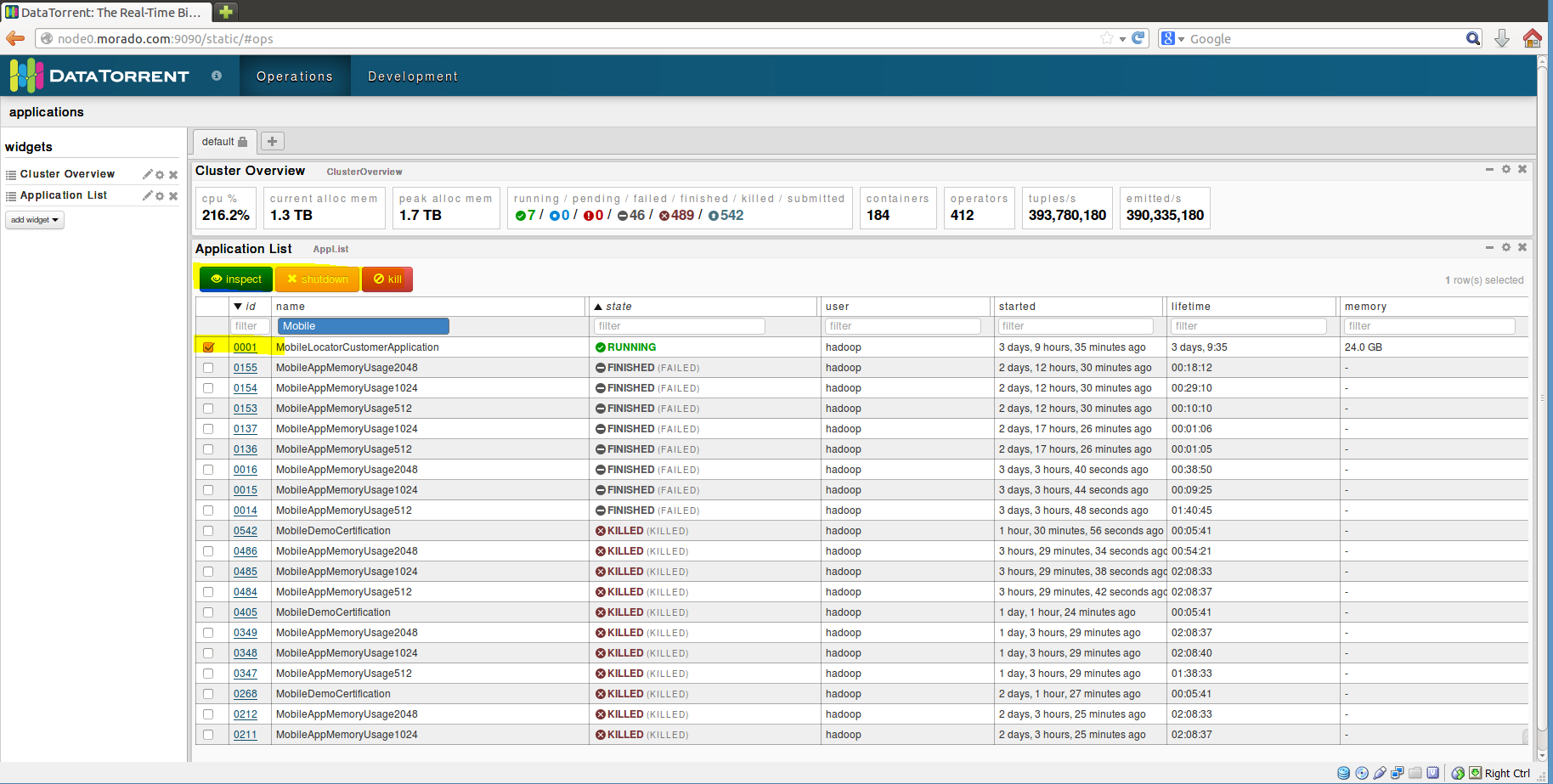


You can see detailed information about –

* partitions
* Metrices
* Properties

You can add more widgets here as well

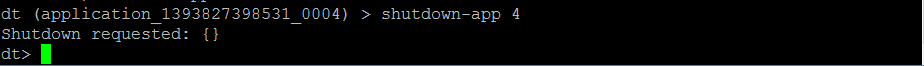
1. Selecting an application from the list of applications also enables the operations to control the application viz. inspect, shutdown, kill. The user can click on any of the buttons to perform the corresponding operation.



# Stopping the application

From dtcli

The user can use the ‘shutdown-app’ command to shut down the application from the dtcli.

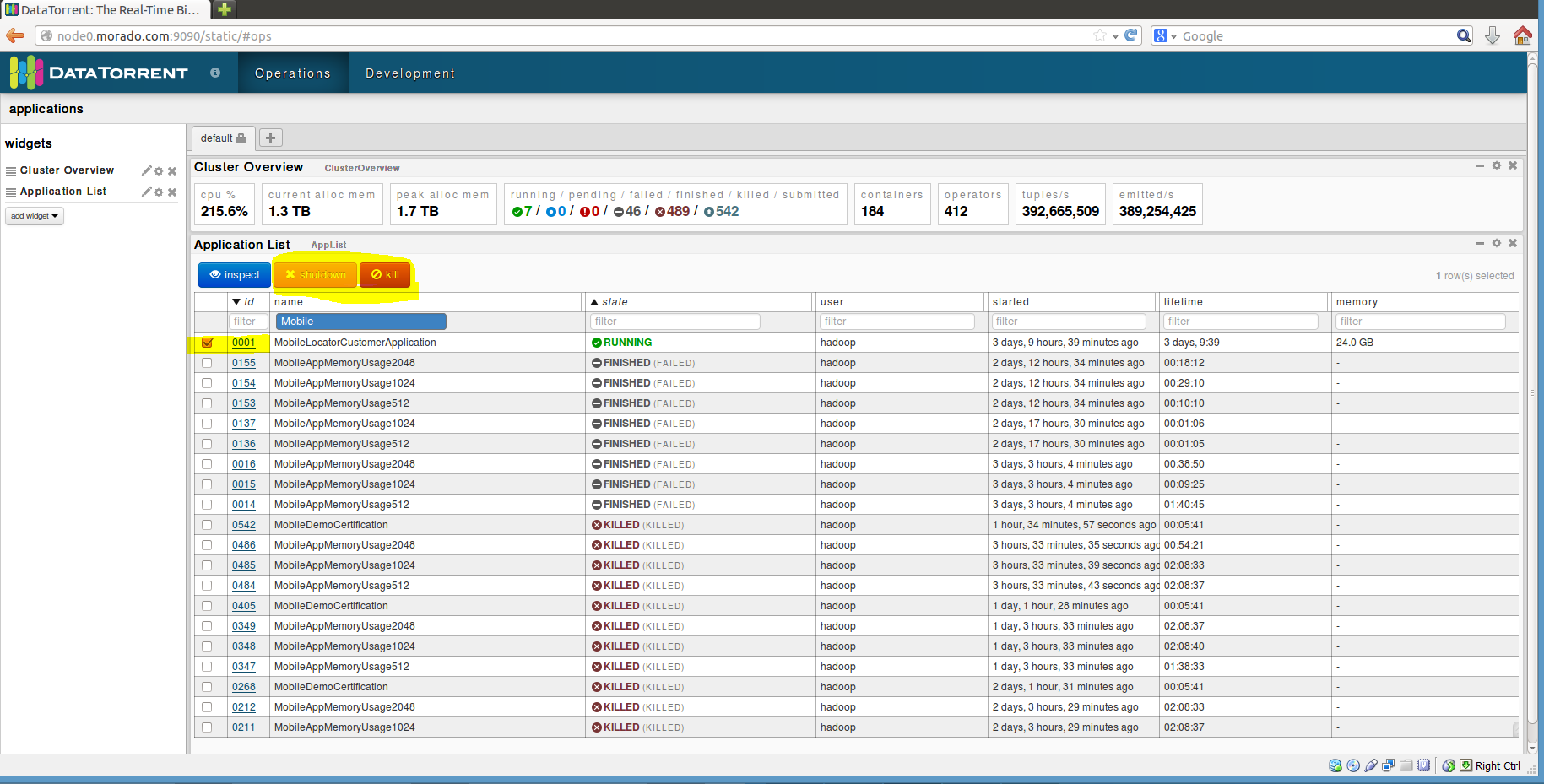


Since this is a platform for streaming applications, in most of the cases, the application would not be shutting down. The user will have to kill the application. In order to achieve this, a kill-app command is provided which can be used.

To check the state of the application after the shutdown has been fired, you can use the list-apps command as shown previously.

From DT Gateway Console

An application can be either shutdown or killed from the DT Gateway console. You can do so when you select an application from the list of applications. Once the user does this, the ‘shutdown’ and ‘kill buttons get activated along with the ‘inspect’ button. The user can then shutdown or kill the application using these buttons.



After the application has been shutdown, you will see the change in state for the application on the dashboard and the application will be in the ‘FINISHED’ state. If killed, it will be shown to be in the ‘KILLED’ state.

# Conclusions

The ‘Mobile’ demo application is used to display dynamically changing loads and how the DataTorrent platform adapts to the same during runtime.