Sumo and Kafka Implementation

Reporter: Shao-Yu Chang

Date:2017-09-29

Simulation of Urban MObility (SUMO)

- Space-continuous and time-discrete vehicle movement
- Different vehicle types
- Multi-lane streets with lane changing
- Different right-of-way rules, traffic lights
- A fast openGL graphical user interface
- Manages networks with several 10.000 edges (streets)
- Fast execution speed (up to 100.000 vehicle updates/s on a 1GHz machine)
- Interoperability with other application at run-time
- Network-wide, edge-based, vehicle-based, and detector-based outputs
- Supports person-based inter-modal trips

start up

http://sumo.dlr.de/wiki/Basics/Basic_Computer_Skills

Install

- If you use `sudo apt-get install sumo` you will get old version sumo (0.25.0)
- Use command below to get newest version.

```
$ sudo add-apt-repository ppa:sumo/stable
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install sumo sumo-tools sumo-doc
```

Open SUMO

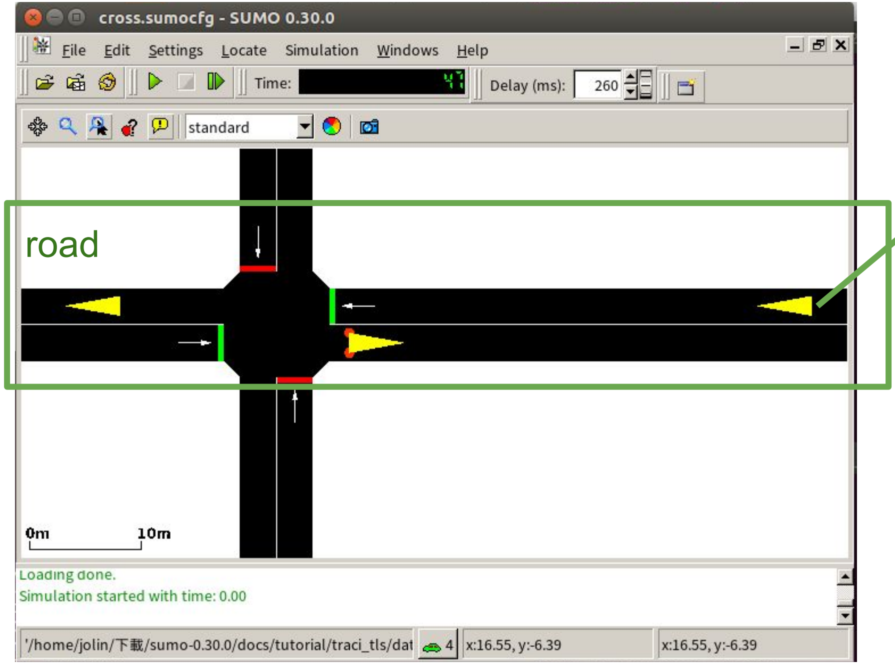
- Open with command :
  ```
  $ sumo-gui
  ```
- File > Open Simulation > /usr/share/doc/sumo-doc/tutorial/traci_tls/data/cross.sumocfg
- This file is a tutorial released by SUMO.
  But, it’s lack of file `cross.rou.xml`. I fonud it on internet and shown below.

cross.rou.xml

```
<routes>
        <vType id="typeWE" accel="0.8" decel="4.5" sigma="0.5" length="5" minGap="2.5" maxSpeed="16.67" guiShape="passenger"/>
        <vType id="typeNS" accel="0.8" decel="4.5" sigma="0.5" length="7" minGap="3" maxSpeed="25" guiShape="bus"/>

        <route id="right" edges="51o 1i 2o 52i" />
        <route id="left" edges="52o 2i 1o 51i" />
        <route id="down" edges="54o 4i 3o 53i" />

        <vehicle id="left_0" type="typeWE" route="left" depart="0" />
        <vehicle id="left_1" type="typeWE" route="left" depart="2" />
        <vehicle id="right_2" type="typeWE" route="right" depart="3" />
        <vehicle id="right_3" type="typeWE" route="right" depart="4" />
        ...

</routes>
```

Position infomation

- Use simulated traffic transportation’s position data to represent simulation devices position.



Loading done.
Simulation started with time: 0.00

'/home/jolin/下載/sumo-0.30.0/docs/tutorial/traci_tls/dat

x:16.55, y:-6.39

x:16.55, y:-6.39

Use python API to get the car position on the road

Reference by SUMO Tutorials: http://sumo.dlr.de/wiki/Tutorials

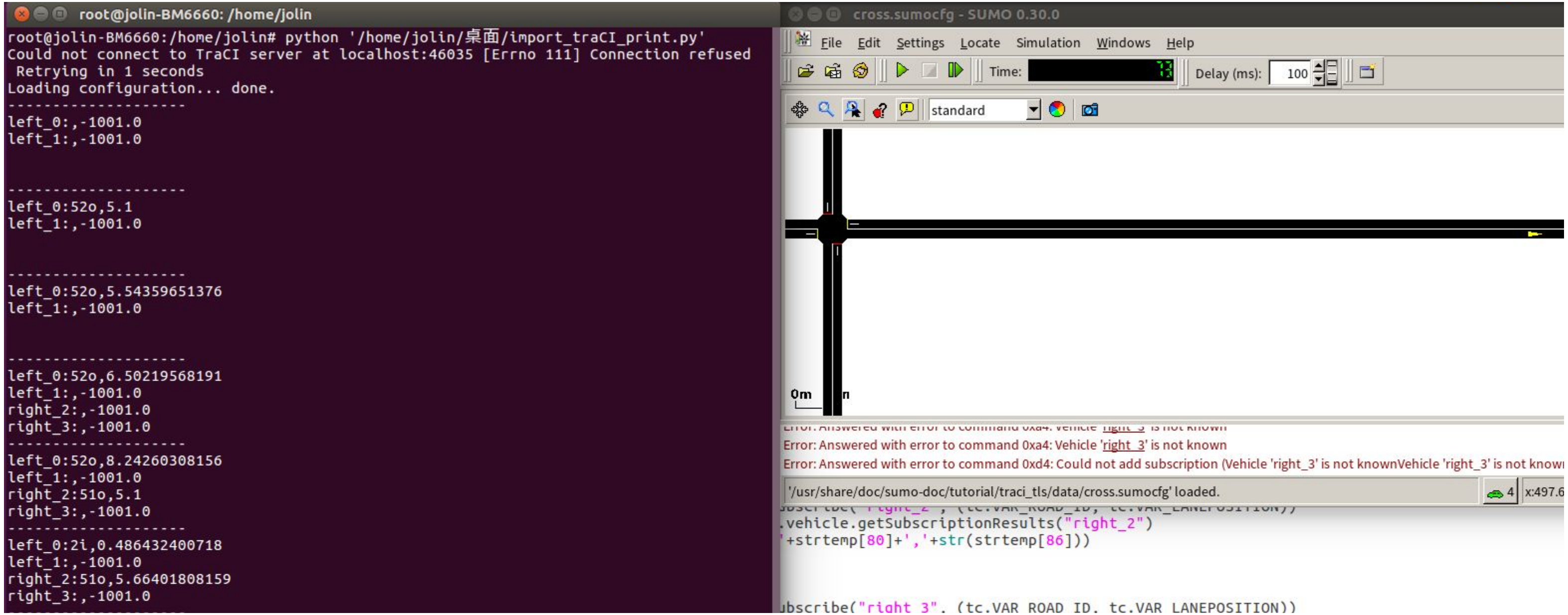
Use python to get the data of SUMO

```
import os, sys

if 'SUMO_HOME' in os.environ:
  tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
  sys.path.append(tools)
else:
  sys.exit("please declare environment variable 'SUMO_HOME'")

import traci

sumoBinary = "/usr/bin/sumo-gui"
sumoCmd = [sumoBinary, "-c",
"/usr/share/doc/sumo-doc/tutorial/traci_tls/data/cross.sumocfg"]
tc = traci.constants
traci.start(sumoCmd)
step = 0
while step < 1000:
  print('--------------------')
  traci.vehicle.subscribe("left_0", (tc.VAR_ROAD_ID,
tc.VAR_LANEPOSITION))
  traci.vehicle.subscribe("left_1", (tc.VAR_ROAD_ID,
tc.VAR_LANEPOSITION))
  strtemp = traci.vehicle.getSubscriptionResults("left_0")
  print('left_0:'+strtemp[80]+','+str(strtemp[86]))
  strtemp = traci.vehicle.getSubscriptionResults("left_1")
  print('left_1:'+strtemp[80]+','+str(strtemp[86]))
```

```
  try:
        traci.vehicle.subscribe("right_2",
(tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
        strtemp =
traci.vehicle.getSubscriptionResults("right_2")
        print('right_2:'+strtemp[80]+','+str(strtemp[86]))
  except:
        print("")
  try:
        traci.vehicle.subscribe("right_3",
(tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
        strtemp =
traci.vehicle.getSubscriptionResults("right_3")
        print('right_3:'+strtemp[80]+','+str(strtemp[86]))
  except:
        print("")
  traci.simulationStep()
  step += 1

traci.close()
```

<SUMO_HOME>:The path of your sumo in usr
ex: /usr/share/sumo
It's a file `tools` in the path.

- When the program started, sumo-gui will pop out.
  And press the start buttom then the terminal will get the data.



```
root@jolin-BM6660:/home/jolin# python '/home/jolin/桌面/import_traCI_print.py'
Could not connect to TraCI server at localhost:46035 [Errno 111] Connection refused
 Retrying in 1 seconds
Loading configuration... done.
--------------------
left_0:,-1001.0
left_1:,-1001.0


--------------------
left_0:52o,5.1
left_1:,-1001.0


--------------------
left_0:52o,5.54359651376
left_1:,-1001.0


--------------------
left_0:52o,6.50219568191
left_1:,-1001.0
right_2:,-1001.0
right_3:,-1001.0
--------------------
left_0:52o,8.24260308156
left_1:,-1001.0
right_2:51o,5.1
right_3:,-1001.0
--------------------
left_0:2i,0.486432400718
left_1:,-1001.0
right_2:51o,5.66401808159
right_3:,-1001.0
```

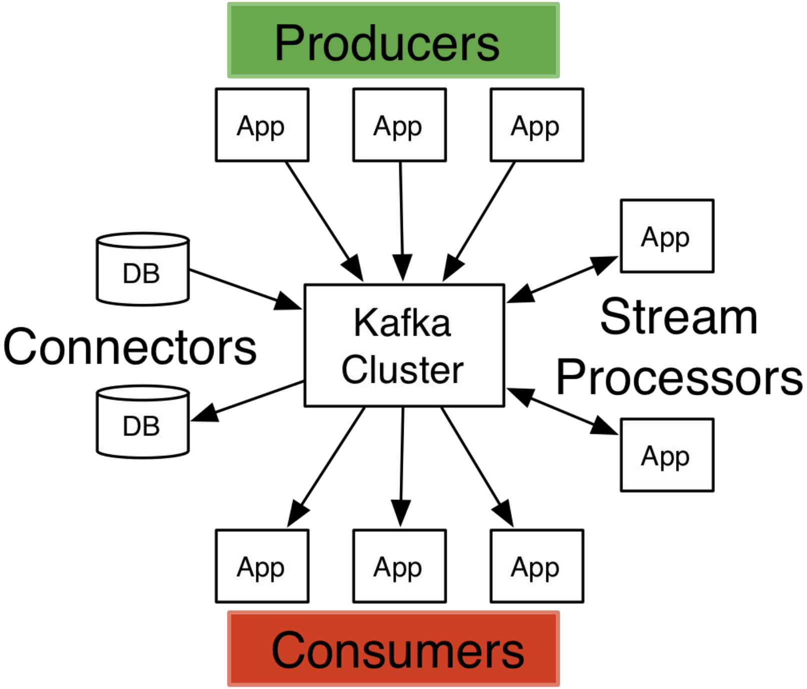
```
Error: Answered with error to command 0xa4: Vehicle 'right_3' is not known
Error: Answered with error to command 0xd4: Could not add subscription (Vehicle 'right_3' is not knownVehicle 'right_3' is not known
'/usr/share/doc/sumo-doc/tutorial/traci_tls/data/cross.sumocfg' loaded.
```

```
.vehicle.getSubscriptionResults("right_2")
'+strtemp[80]+','+str(strtemp[86]))


ubscribe("right_3", (tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
```

Kafka

- It's sort of like database, but it's architecture is benifited to high load I/O. (It's fit to IoT Uses.)
- It's use two component:
  - Producer:
    Produce data to push to kafka.
  - Consumer:
    Consum the data store in kafka.

Reference by Kafka website:
https://kafka.apache.org/documentation/



Install kafka

- Before installation it’s might pre install java’s jdk.

```
$ sudo apt-get install openjdk-8-jre
$ sudo apt-get install openjdk-8-jdk
```

- Install kafka

```
$ wget http://apache.stu.edu.tw/kafka/0.11.0.0/kafka_2.11-0.11.0.0.tgz
$ tar -xzf kafka_2.11-0.11.0.0.tgz
$ tar -jxv -f filename.tar.bz2 -C  欲解壓縮的目錄
$ cd kafka_2.11-0.11.0.0
```

Reference by kafka website

https://kafka.apache.org/quickstart

For mac

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew install zookeeper
```

in mac you have to modify the “config/server.properties”

make the “#listeners=PLAINTEXT://:9092” to became “listeners=PLAINTEXT://172.17.0.2:9092”

Install kafka

- Start server

```
$ bin/zookeeper-server-start.sh config/zookeeper.properties &
$ bin/kafka-server-start.sh config/server.properties &
```

- Create topic

```
$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1
--topic test
$ bin/kafka-topics.sh --list --zookeeper localhost:2181
test
```

- Send some messages

```
$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
>This is a message
>This is another message
```

- Start a consumer

```
$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

This is a message

This is another message
```

Import kafka API to python program

```
import os, sys

if 'SUMO_HOME' in os.environ:
  tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
  sys.path.append(tools)
else:
  sys.exit("please declare environment variable 'SUMO_HOME'")

from kafka import KafkaProducer
from kafka.errors import KafkaError
import traci

producer = KafkaProducer(bootstrap_servers=['localhost:9092'])

sumoBinary = "/usr/bin/sumo-gui"
sumoCmd = [sumoBinary, "-c",
"/usr/share/doc/sumo-doc/tutorial/traci_tls/data/cross.sumocfg"]
tc = traci.constants
traci.start(sumoCmd)
```

```
step = 0
while step < 1000:
  producer.send('test', '--------------------')
  traci.vehicle.subscribe("left_0", (tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
  traci.vehicle.subscribe("left_1", (tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
  strtemp = traci.vehicle.getSubscriptionResults("left_0")
  producer.send('test', 'left_0:'+strtemp[80]+','+str(strtemp[86]))
  strtemp = traci.vehicle.getSubscriptionResults("left_1")
  producer.send('test', 'left_1:'+strtemp[80]+','+str(strtemp[86]))
  try:
          traci.vehicle.subscribe("right_2", (tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
          strtemp = traci.vehicle.getSubscriptionResults("right_2")
          producer.send('test', 'right_2:'+strtemp[80]+','+str(strtemp[86]))
  except:
          print("")
  try:
          traci.vehicle.subscribe("right_3", (tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
          strtemp = traci.vehicle.getSubscriptionResults("right_3")
          producer.send('test', 'right_3:'+strtemp[80]+','+str(strtemp[86]))
  except:
          print("")
  traci.simulationStep()
  step += 1

traci.close()
```

KafkaConsumer

```
from kafka import KafkaConsumer

# To consume latest messages and auto-commit offsets
consumer =
KafkaConsumer('my-topic',group_id='my-group',bootstrap_servers=
['localhost:9092'])

for message in consumer:
  # message value and key are raw bytes -- decode if necessary!
  # e.g., for unicode: `message.value.decode('utf-8')`
  print ("%s:%d:%d: key=%s value=%s" % (message.topic,
message.partition,message.offset, message.key,message.value))

# consume earliest available messages, don't commit offsets
KafkaConsumer(auto offset_reset='earliest',
enable_auto_commit=False)

# consume json messages
KafkaConsumer(value deserializer=lambda m:
json.loads(m.decode('ascii')))

# consume msgpack
KafkaConsumer(value_deserializer=msgpack.unpackb)
```

```
# StopIteration if no message after 1sec
KafkaConsumer(consumer_timeout_ms=1000)


# Subscribe to a regex topic pattern
consumer = KafkaConsumer()
consumer.subscribe(pattern='^awesome.*')

# Use multiple consumers in parallel w/ 0.9 kafka
brokers
# typically you would run each on a different server /
process / CPU
consumer1 = KafkaConsumer('my-topic',
                          group_id='my-group',

bootstrap servers='my.server.com')
consumer2 = KafkaConsumer('my-topic',
                          group_id='my-group',

bootstrap_servers='my.server.com')
```

There are many configuration options for the consumer class. See **`KafkaConsumer`** API documentation for more details.

KafkaProducer

```python
from kafka import KafkaProducer
from kafka.errors import KafkaError

producer = KafkaProducer(bootstrap_servers=['broker1:1234'])

# Asynchronous by default
future = producer.send('my-topic', b'raw_bytes')

# Block for 'synchronous' sends
try:
    record_metadata = future.get(timeout=10)
except KafkaError:
    # Decide what to do if produce request failed...
    log.exception()
    pass

# Successful result returns assigned partition and offset
print (record_metadata.topic)
print (record_metadata.partition)
print (record_metadata.offset)
```

```python
# produce keyed messages to enable hashed partitioning
producer.send('my-topic', key=b'foo', value=b'bar')

# encode objects via msgpack
producer = KafkaProducer(value_serializer=msgpack.dumps)
producer.send('msgpack-topic', {'key': 'value'})

# produce json messages
producer = KafkaProducer(value_serializer=lambda m:
json.dumps(m).encode('ascii'))
producer.send('json-topic', {'key': 'value'})

# produce asynchronously
for _ in range(100):
    producer.send('my-topic', b'msg')

# block until all async messages are sent
producer.flush()

# configure multiple retries
producer = KafkaProducer(retries=5)
```

Thanks