PeerSimGrid 1.0

Khaled Baati

June 15, 2015

PeerSimGrid (PSG) is an interface developed in Java and allows users to simulate and execute their code under PeerSim or Simgrid simulator, using PeerSim implementation policy. For a quick start please refer to the README file, included in the distribution.

Introduction

In this Tutorial we will explain step by step how to use *PeerSimGrid* (PSG) through a basic example. In this tutorial it is supposed that you have access to:

◆ knowledge of the Java language .

◆ a Java compiler for Java version 1.8 or higher.

◆ *PeerSim* documentation.

We consider the Event driven version of averaging example where nodes collectively calculate the average of some parameter via periodically exchanging messages and performing pairwise averaging steps.

The development will always remain the same as under *PeerSim*, so please refer to its documentation at peersim.sourceforge.net . The only changes are made in the configuration file of the simulation.

What is PSG?

PeerSim is a Peer-to-Peer simulator. It has been designed to be both dynamic and scalable. The engines use a simple ASCII file based configuration mechanism.

The philosophy of *PeerSim* is to use a modular approach, as the preferred way of coding with it is to re-use existing modules. These modules can be of different kinds, for example there are modules which can initialize the network, modules which can handle the different protocols, modules to control and modify the network.

PeerSim can also work in two different modes: cycle-based or event-based. The cycle- based engine is based on a very simple time scheduling algorithm and is very efficient and scalable. The event-based engine is based on a more complex but more realistic approach.

PeerSim presents an easy and simply way of coding applications.

In the other side, *Simgrid* is a simulation framework to study the behavior of large-scale distributed systems such as Grids, P2P systems and Cloud, is able to simulate pretty accurately the real

behavior of applications. *Simgrid* presents many platforms and network topologies to perform execution.

The Idea of our tool *PeerSimGrid,* is to combine the two philosophy of *PeerSim* and *Simgrid* as the user have the choice to execute his application under *PeerSim* or *Simgrid* with the same code.

The user develops their applications under *PeerSim* implementation policy which is more easier and simple, and according to the parameters entered in the configuration file, *PeerSim* or *Simgrid* engine will be used.

	# A folder name, where will be stored the simulation output. OutputName edaggregation # the platform file for simgrid simulator. platform platforms/psg.xml # the unit of measure: sec /ms /us unit ms
SIZE 100 CYCLES 100 CYCLE SIZE*100 random.seed 1234567890 network.size SIZE simulation.endtime CYCLE*CYCLES simulation.logtime CYCLE ##########protocols ########## protocol.link peersim.core.IdleProtocol protocol.avg example.edaggregation.AverageED protocol.avg.linkable link protocol.avg.step CYCLE protocol.avg.transport tr protocol.tr UnreliableTransport protocol.tr.transport urt protocol.tr.drop DROP protocol.urt UniformRandomTransport protocol.urt.mindelay (CYCLE*MINDELAY)/100 protocol.urt.maxdelay (CYCLE*MAXDELAY)/100 ########## initialization ########## init. ########## control ########## control..	SIZE 100 CYCLES 100 CYCLE SIZE*100 random.seed 1234567890 network.size SIZE simulation.duration CYCLE*CYCLES simulation.logtime CYCLE ##########protocols ########## protocol.link peersim.core.IdleProtocol protocol.avg example.edaggregation.AverageED protocol.avg.linkable link protocol.avg.step CYCLE protocol.avg.transport tr protocol.tr UnreliableTransport protocol.tr.transport urt protocol.tr.drop DROP protocol.urt psgsim.PSGTransport protocol.urt.mindelay (CYCLE*MINDELAY)/100 protocol.urt.maxdelay (CYCLE*MAXDELAY)/100 ########## initialization ########## init. ########## control ########## control..
(a)	(b)

Fig 1.Configuration file of the edaggregation example for (a) PeerSim and (b) Simgrid simulation

PeerSim simulation

To simulate your code under *PeerSim,* nothing new to change as we saying before   just for launching the simulation, you execute:

> ./run *configs/edaggregation.txt*

Where *edaggregation.txt* is the configuration file (see *Fig 1 (a)).*

To have a better output presentation we suggest to add a new properties: the "*outputName*" which define the folder's name where the simulation's results will be stored under a file name "*ps.txt*".

You can find the protocol code of the user "*AverageED.java*" under *src/example/edaggregation* and the complete configuration file "*edaggregation.txt*" (if you want to perform *PeerSim* simulator) and *edaggregationPSG (if you want perform Simgrid simulator)* under *configs/* .

Simgrid simulation

To be able to simulate with *Simgrid*, a few changes must be done in the configuration file (see *Fig 1 (b)).* Some properties must be changed and other will be added, as follow:

1) Replaces "*simulation.endtime*" by "*simulation.duration*": This change is the key to indicate to PSG which simulator will be use.

2) Replaces "*UniformRandomTransport*" by "*psgsim.PSGTransport*": This properties define the transport protocol.

3) Define a new properties "platform" : An xml file describes on which you want run your application. You can define your platform on the configuration file as:

platform path/to/your/file.xml

This platform will be used by *Simgrid* simulator to perform the simulation, if not defined, "*psg.xml*" platform will be the default value. You can find several platform files under platforms folder.

4) Define a new properties: "*OutputName*" a folder name, where the simulation output will be stored under a file named "*psg.txt*". This output is the result of your "*System.out*" instruction.

5) Define the unit of measure: "unit" which is defined the unit of measurement for the application (duration, steps of protocols and controls..). There are three possibilities units: *sec* / *ms* / *us* if not defined, sec will be the default value.

To execute the ed-aggregation example:

> ./run *configs/edaggregationPSG.txt*

One last thing in the case the user want to send an event/message with a size not null, the message class must implement *Sizable* interface to define the *getSize* method and the constructor of this class includes this parameter (size in bytes).

For example, in the bit-torrent example (you will find all code in the example folder), you can create a message "*IntMsg*" with a size 16*1024 bytes (the last parameter) as shown in the follow:

```
event = new IntMsg(PIECE, node, req.id,16*1024);
```

The Message class "*IntMsg*" presented in the Fig 2 implements the *Sizable* interface and define the *getSize* method. *Simgrid* is based on this return value to perform a message tracking with size not null, unlike *PeerSim* where all messages are supposed with null size.

```java
import psgsim.Sizable;

public class IntMsg extends SimpleMsg implements Sizable {

    private double size;

    /**
     * The basic constructor of the message.
     *
     * @param type
     *            the type of the message
     * @param sender
     *            The sender node
     * @param value
     *            The data value of the message
     */
    public IntMsg(int type, Node sender, int value, double size) {
        super.type = type;
        super.sender = sender;
        this.integer = value;
        this.size = size;
    }

    @Override
    public double getSize() {
        return size;
    }

    //other methods
     ...
}
```
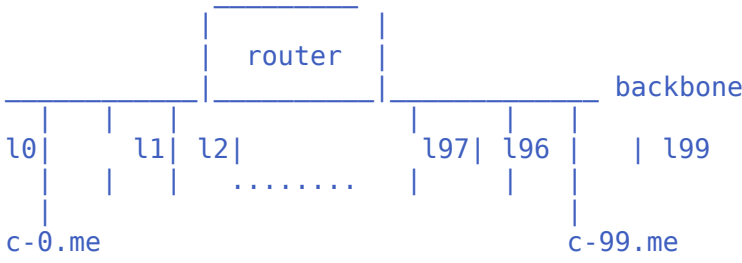
Fig 2.Message Class example

Configure your network platform

For the case of *PeerSim* to add a delay to your messages, you define it as a properties in the configuration file, in the case of *Simgrid* to define this properties you must added to the platform file, as shown in the fig... with the properties "*lat*" for the latency and "*bb_lat*" for the backbone latency. Note that there are other properties as "*bw*" for the bandwidth, "*radical*" for the number of host (in this case our network contains maximum 999 hosts).

For more information about platform configuration please refer to *Simgrid* documentation (platform description section): simgrid.gforge.inria.fr/simgrid/latest/doc/platform.html

```xml
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
<!--
                    _____   |
                   |          |
                   |  router  |
       _____|_____|_____   backbone
        |    |    | |          |    |    |
     l0|     l1| l2|        l97| l96 |    | l99
        |    |    |  ........   |    |    |
        |    |    |             |         |
     c-0.me                           c-99.me


-->
<platform version="3">
<config>
<prop id="network/latency_factor" value="1.0"/>
</config>
<AS  id="AS0"  routing="Full">
  <cluster id="my_cluster_1" prefix="" suffix=""
           radical="0-1000"  power="1Gf"  bw="200Mbps" lat="0ms"
        bb_bw="200Mbps" bb_lat="0ms"/>
</AS>
</platform>
```

Fig 3.Example of platform file

Limits

There are some limits in PSG using *Simgrid* simulator:

- The number of thread: in the actual version of PSG, the maximal threads supported is 20 000, this number is proportional to the size of the network (for each node at least one thread will be created).

- The platform file must contain a number of hosts matching the size of nodes declared in the configuration file.

No limits for the case of *PeerSim* (refers to *PeerSim* documentation) .