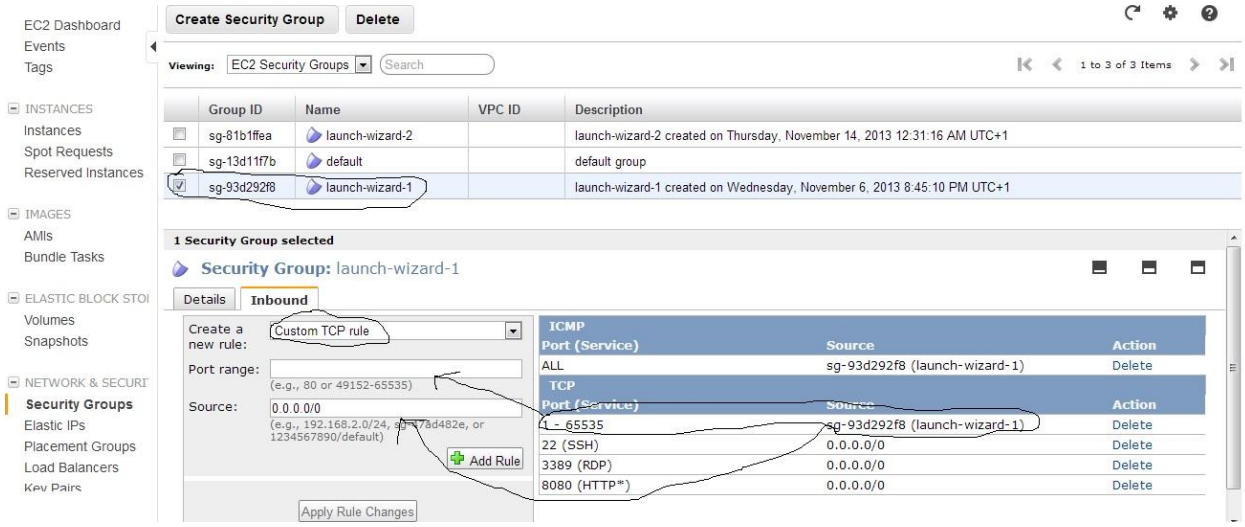
Exericse Sheet - 10

1) Create a new security group for custom TCP rule, allow all ports and a source as the launch wizard. This launch wizard shall be used for all three instances to be created later.



2) Create 3 EC2 instances in same availability zone. Use the security group created earlier for all three instances.
3) Install dependencies on all three instances.
   sudo apt-get -y install make gcc g++ openmpi-bin openmpi-common libopenmpi-dev
4) Generate key on master
   ssh-keygen -t rsa
5) Append the contend of .ssh/id_rsa.pub (master) to .ssh/authorized_keys(slaves)
6) Create a file hosts.mpi with the contents
   master
   node1
   node2
7) Edit file /etc/hosts and add entry
   10.45.178.13 ip-10-45-178-13 master
   10.181.143.105 ip-10-181-143-105 node1
   10.181.143.183 ip-10-181-143-183 node2

   Where first entry is the IP address, can be found using 'ifconfig' command
   Second entry is the host name, can be found using 'hostname' command
8) Create a file monte_carlo.c with the contents at the end of the document on the master
9) Compile the program using 'mpicc monte_carlo.c -o monte_carlo' on the master
10) Open file ~/.bashrc and add the following entries at the end of the file.
    export PATH=/usr/lib/openmpi:$PATH
    export LD_LIBRARY_PATH=/usr/lib/openmpi/lib
11) Do the above change in slaves as well.

12) After the changes execute the command '. .bashrc'
13) Copy the compiled file to the slave nodes using the following commands
    scp monte_carlo node1:~
    scp monte_carlo node2:~
14) Now run the following command on the master
    mpirun -np 6 --hostfile hosts.mpi monte_carlo 14000
15) The last argument in the above command (14000) is the number of points used for PI estimation.
16) The above command shall print the number of points used, the PI value and the time elapsed.

File : monte_carlo.c

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <math.h>
#include <time.h>

int main(int argc, char* argv[]){
int i,id, np,N;
double x, y,double_N,eTime,sTime,pTime;
int lhit;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &np);
MPI_Comm_rank(MPI_COMM_WORLD, &id);

if(argc != 2){
   //if (id==0){
      fprintf(stderr,"Wrong number of arguments \n");
      fflush(stderr);
   //}
   MPI_Abort(MPI_COMM_WORLD,1);
} else {
    sscanf( argv[1], "%d", &N);
}

//sscanf(argv[1], "%lf", &double_N);
//N = 1000;//lround(double_N);
MPI_Barrier(MPI_COMM_WORLD);
sTime = MPI_Wtime();
lhit = 0;
```

```
srand((unsigned)(time(0)));
int lN = N/np;

for(i = 0; i<lN;i++){
x = ((double)rand())/((double)RAND_MAX);
y = ((double)rand())/((double)RAND_MAX);
if (((x*x) + (y*y)) <= 1) lhit++;
}

int hit=0;
MPI_Allreduce(&lhit,&hit,1,MPI_INT,MPI_SUM,MPI_COMM_WORLD);
double est;
est = (hit*4)/((double)N);
MPI_Barrier(MPI_COMM_WORLD);
eTime = MPI_Wtime();
pTime = fabs(eTime - sTime);

if (id == 0) {
printf("Number of Points Used:      %d\n",N);
printf("Estimate of Pi:         %24.16f\n",est);
printf("Elapsed Wall time:      %5.3e\n",pTime);
}

MPI_Finalize();
return 0;
}
```