Developing Techniques to Improve Performance of Mobile Cloud Applications

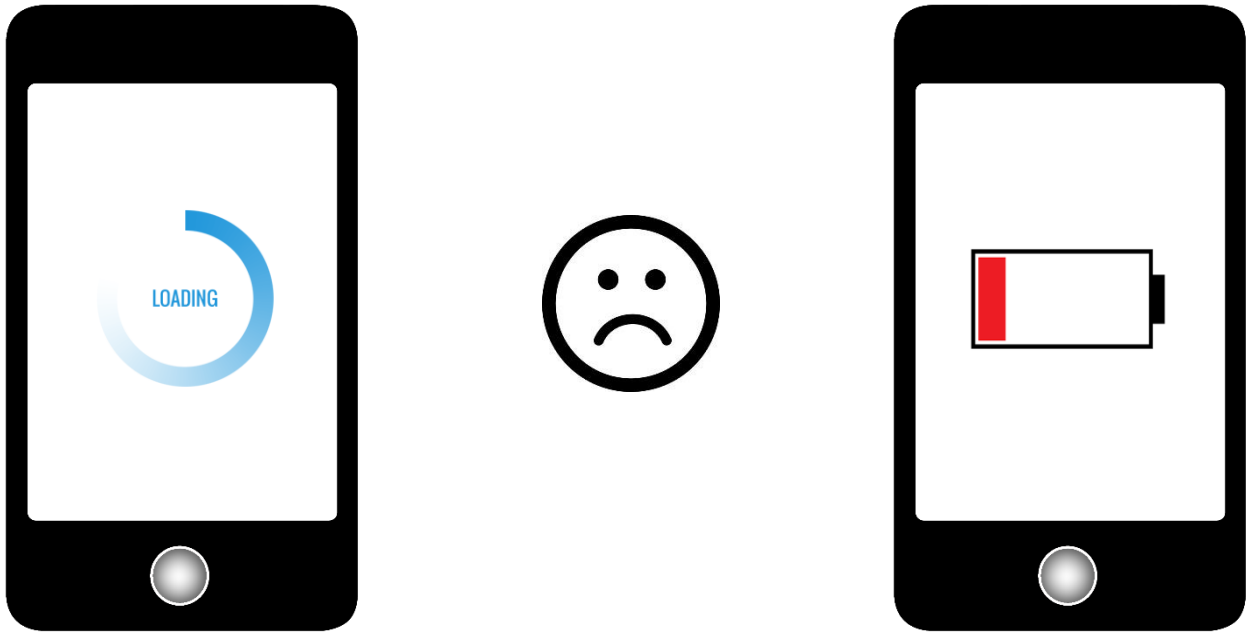
Francisco Airton Silva

Ph.D Research

Advisor: *Dr. Paulo Maciel*

Co-Advisor: *Dr. Alessandro Mei (Università di Roma)*

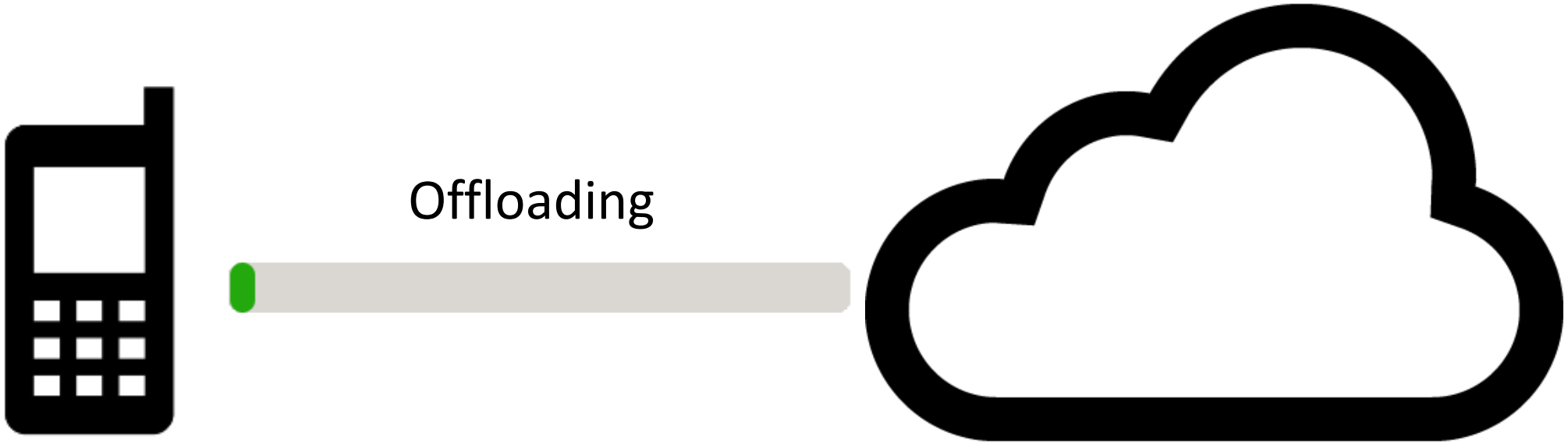
General Problem



Time

Battery

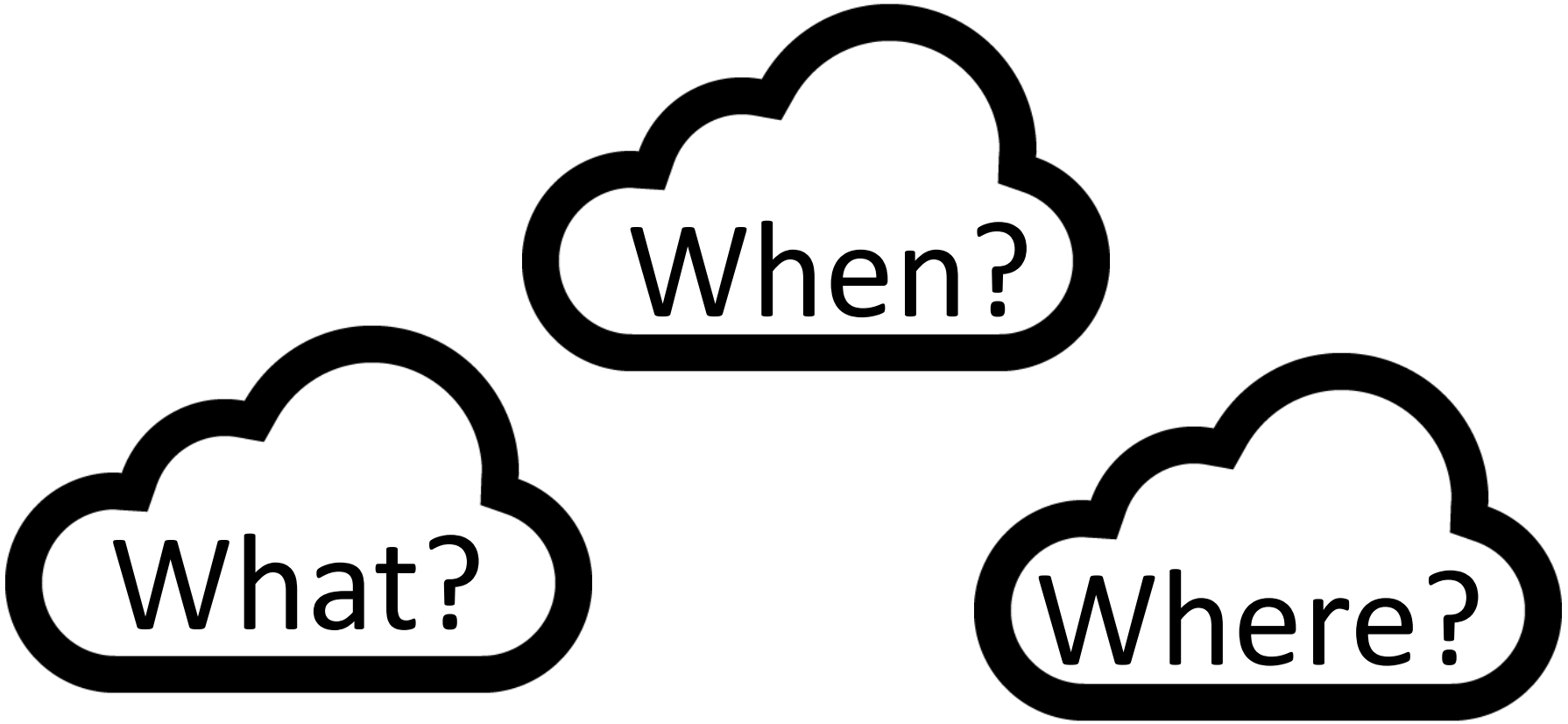
General Solution



Mobile

Cloud

General Challenges




When?
What?
Where?


When?
What?
Where?


When?
What?
Where?

Specific Goal

Specific Goal

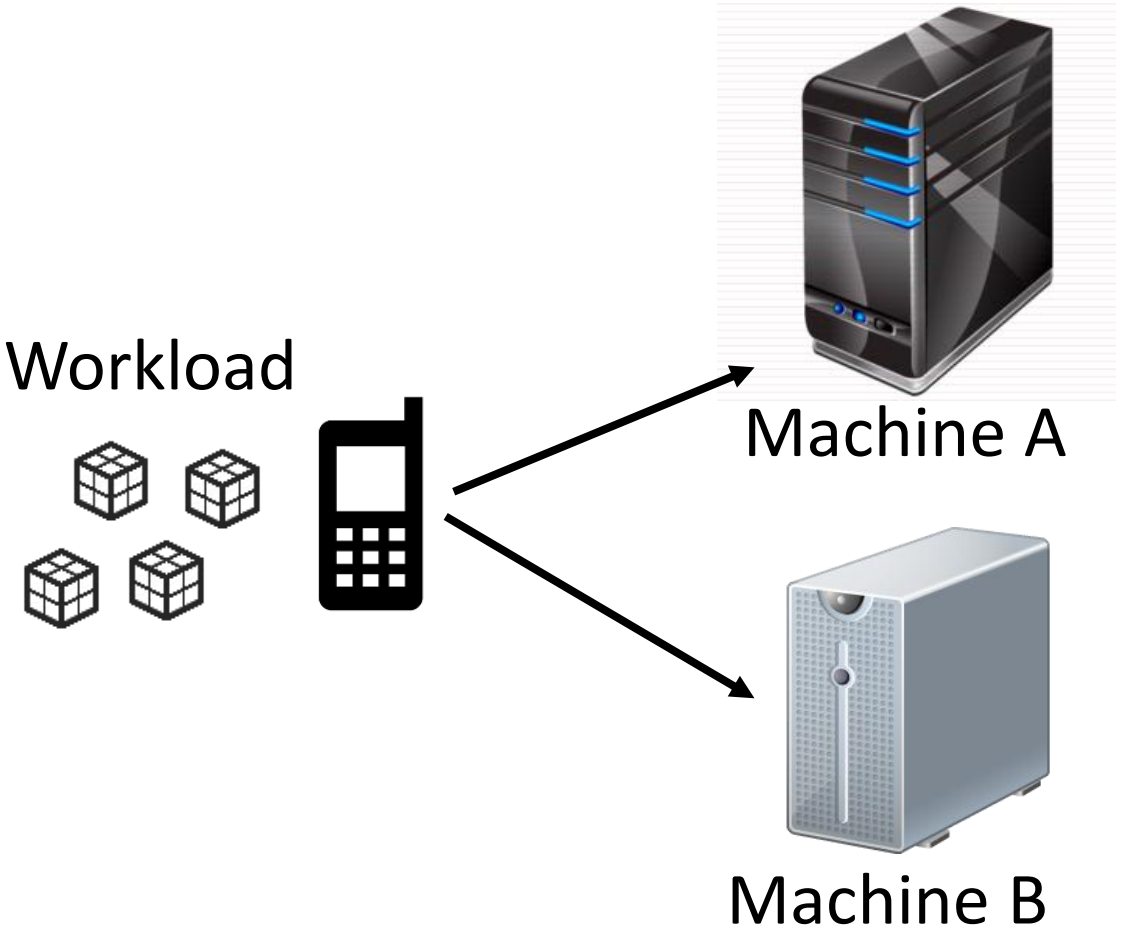
Scenario One

(Jobs Distribution)

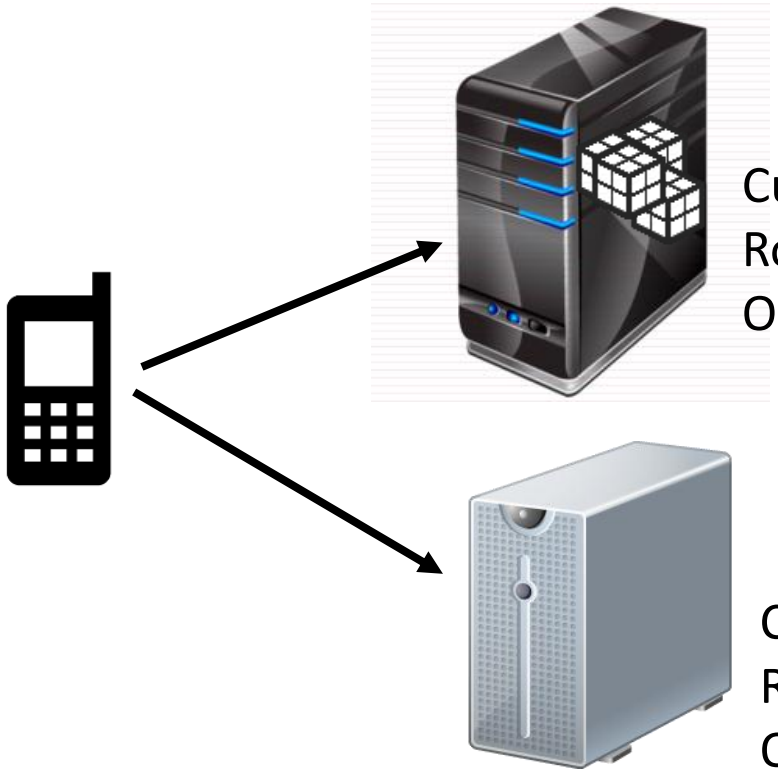
Scenario Two

(System Modelling)

Where to Offload? Scenario One (Jobs Dist.)



Where to Offload? Scenario One (Jobs Dist.)



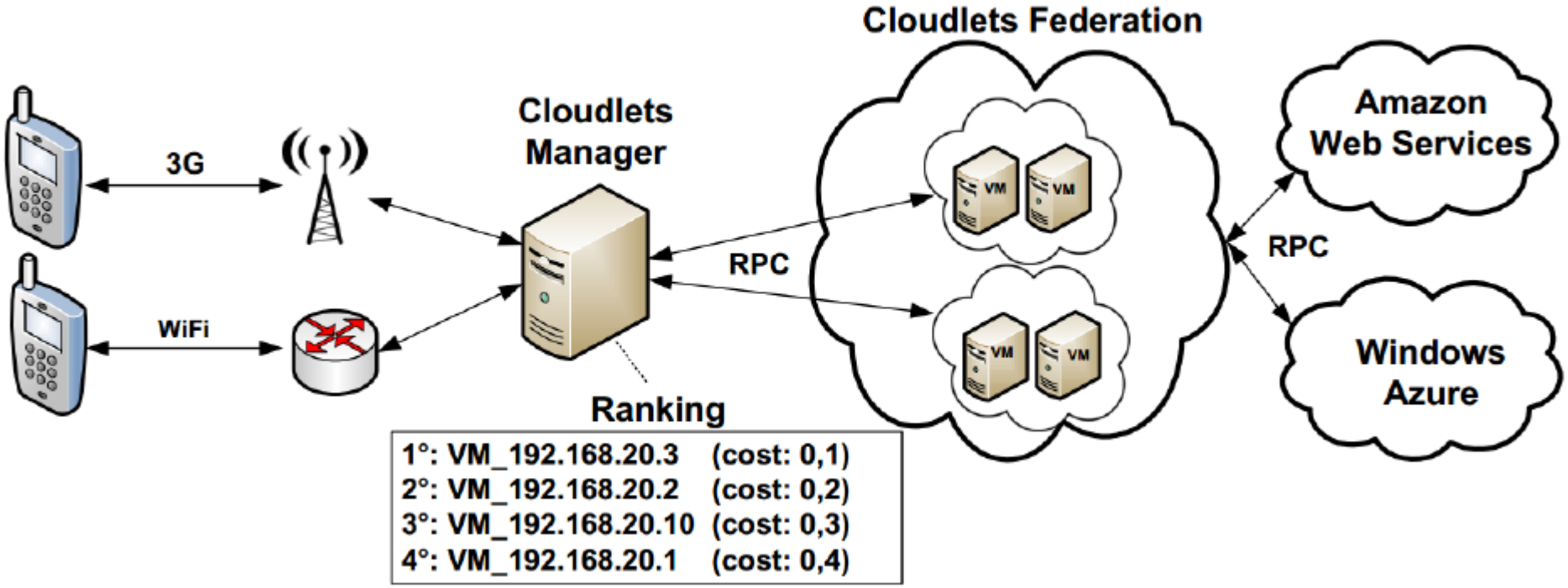
Current CPU (%): 10
Round-Trip Time: 30
Offloading Cost: 70

Current CPU (%): 50
Round-Trip Time: 60
Offloading Cost: 90

SmartRank Algorithm

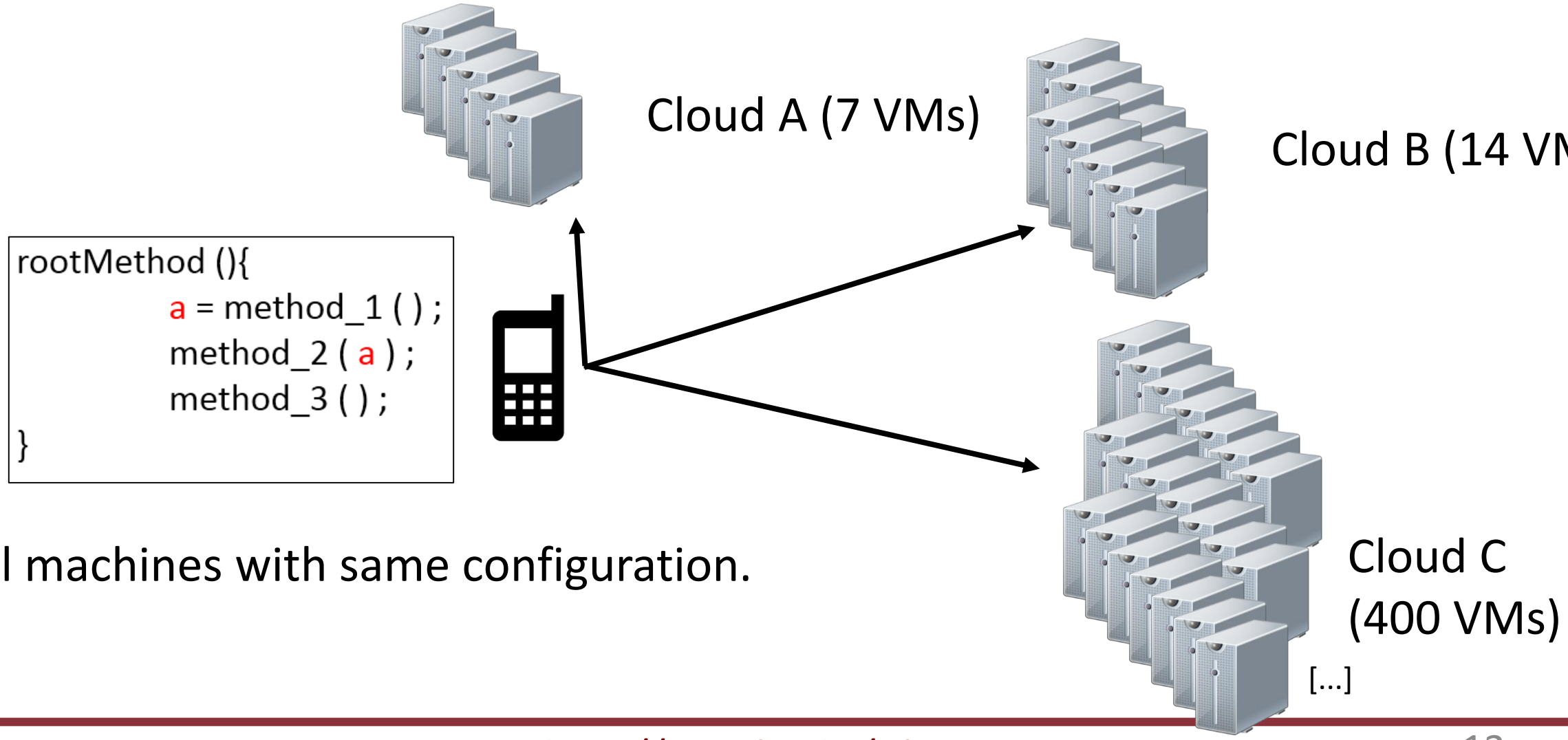
48% of Response Time Improvement

Where to Offload? Scenario One (Jobs Dist.)



SmartRank: a smart scheduling tool for mobile cloud computing
The Journal of Supercomputing

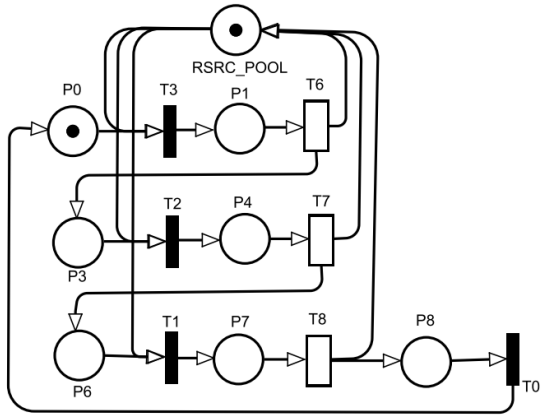
Where to Offload? Scenario Two (System Modelling)

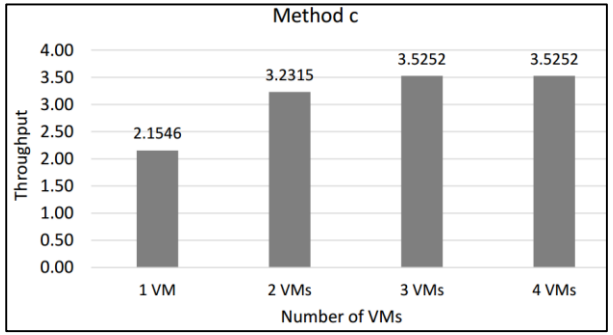


*All machines with same configuration.

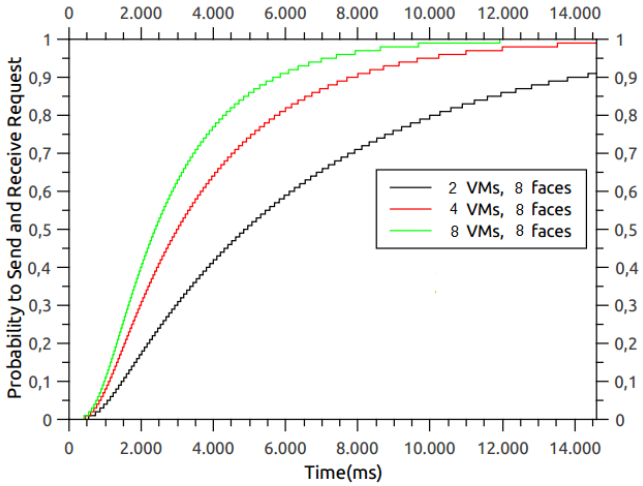
Where to Offload? Scenario Two (System Modelling)

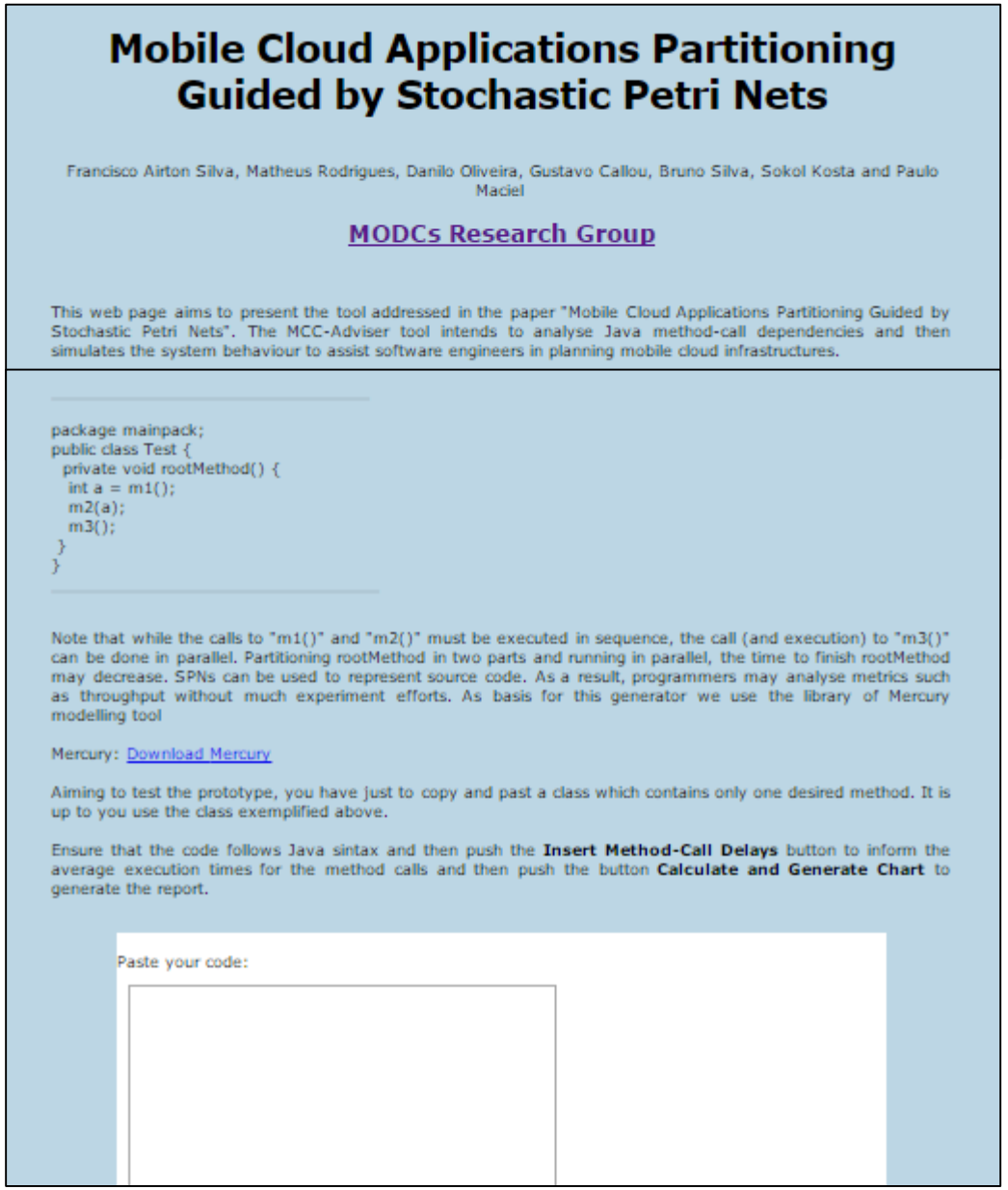
```
public List<Image> method_A (Image image ){
        Image image2 = reduceColor ( image ) ;
        Image image3 = reduceColor ( image2 ) ;
        Image image4 = reduceColor ( image3 ) ;
results.add ( image4 ) ;
return results;
}
```





- How many VMs are needed to get an optimal mobile cloud execution?
- What is the amount of method-calls per time unit (throughput)?
- How long it takes in average to finish the application execution (MTTE)?





Mobile Cloud Applications Partitioning Guided by Stochastic Petri Nets

Francisco Airton Silva, Matheus Rodrigues, Danilo Oliveira, Gustavo Callou, Bruno Silva, Sokol Kosta and Paulo Maciel

MODCs Research Group

This web page aims to present the tool addressed in the paper "Mobile Cloud Applications Partitioning Guided by Stochastic Petri Nets". The MCC-Adviser tool intends to analyse Java method-call dependencies and then simulates the system behaviour to assist software engineers in planning mobile cloud infrastructures.

```
package mainpack;
public class Test {
  private void rootMethod() {
   int a = m1();
   m2(a);
   m3();
 }
}
```

Note that while the calls to "m1()" and "m2()" must be executed in sequence, the call (and execution) to "m3()" can be done in parallel. Partitioning rootMethod in two parts and running in parallel, the time to finish rootMethod may decrease. SPNs can be used to represent source code. As a result, programmers may analyse metrics such as throughput without much experiment efforts. As basis for this generator we use the library of Mercury modelling tool

Mercury: Download Mercury

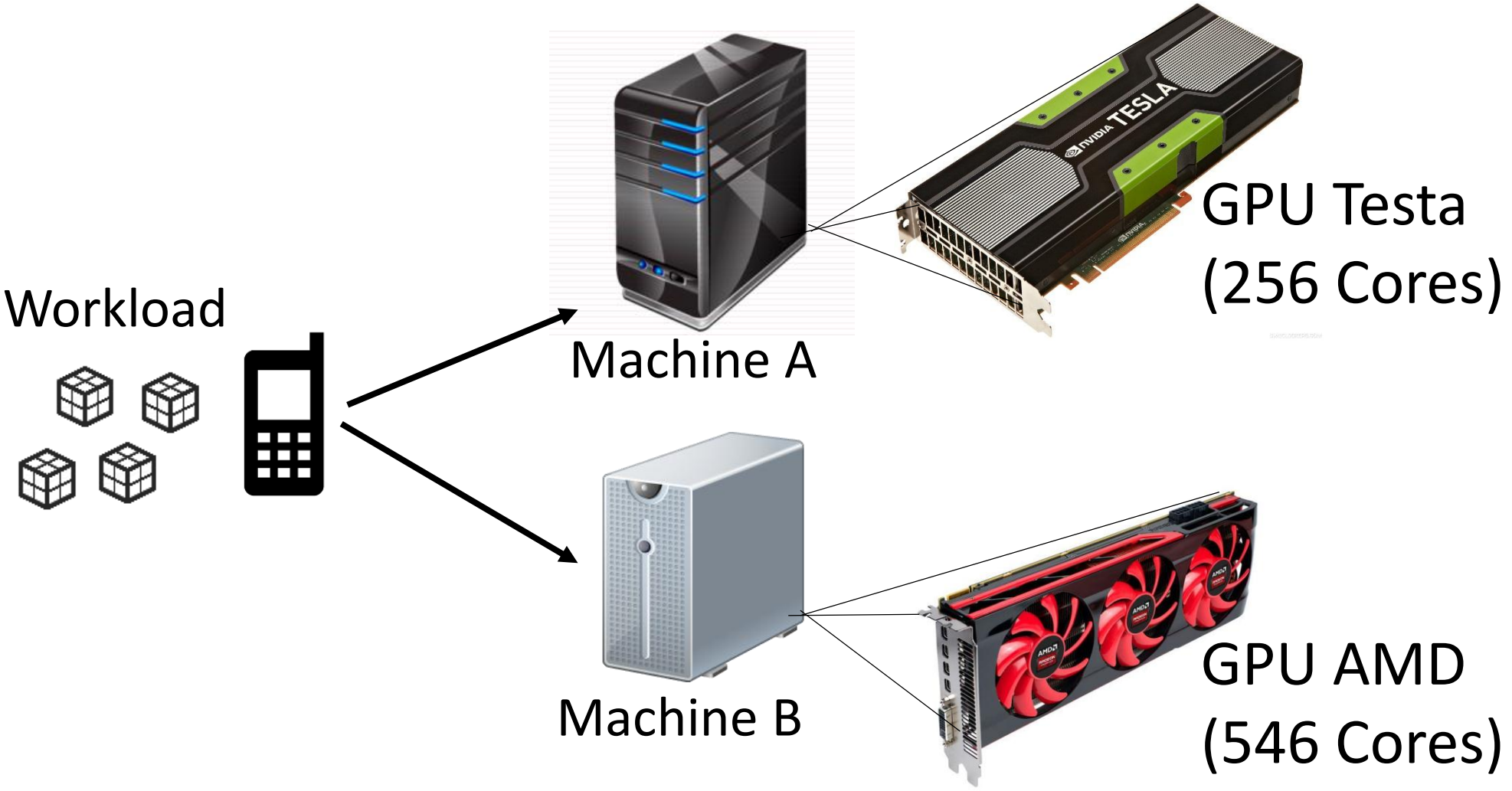
Aiming to test the prototype, you have just to copy and past a class which contains only one desired method. It is up to you use the class exemplified above.

Ensure that the code follows Java sintax and then push the **Insert Method-Call Delays** button to inform the average execution times for the method calls and then push the button **Calculate and Generate Chart** to generate the report.

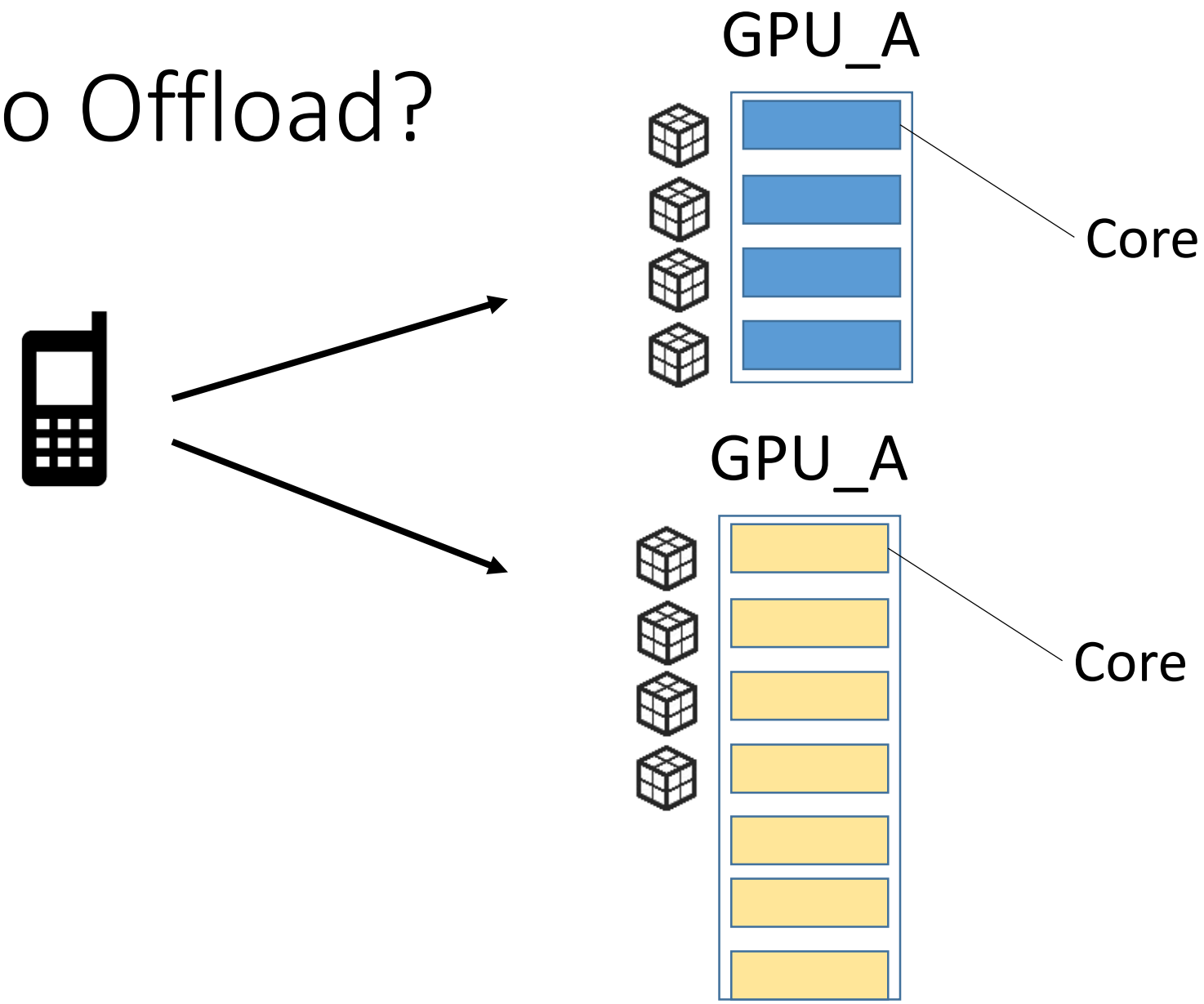
Paste your code:

http://cin.ufpe.br/~faps/mcc-adv/

Where to Offload? Scenario Two (System Modelling)

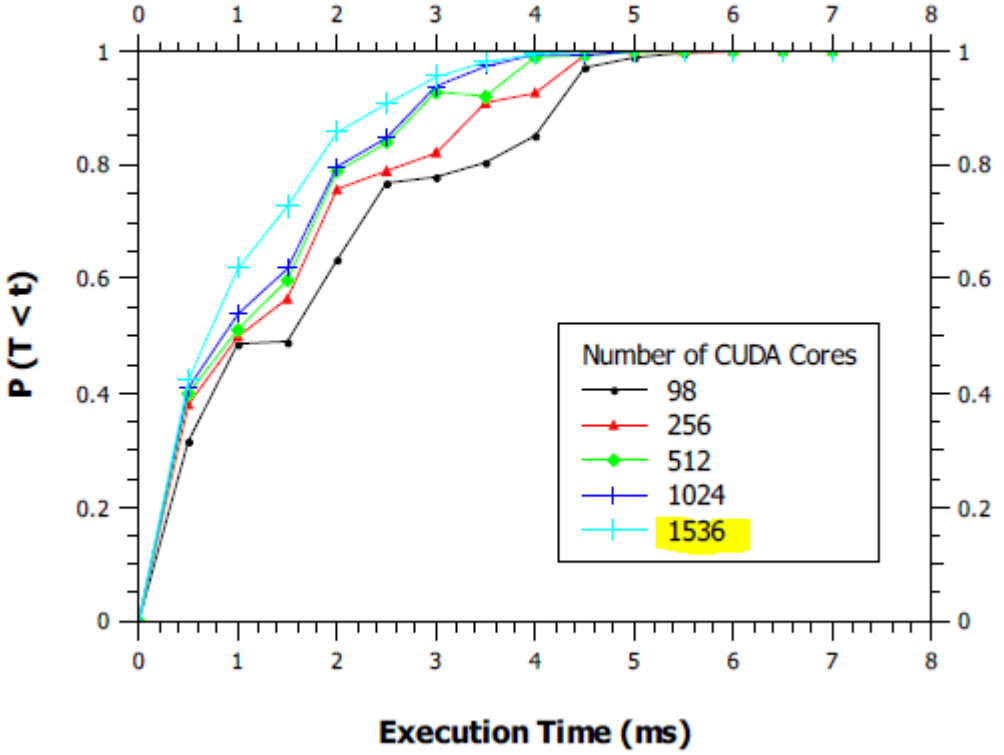


Where to Offload?



Where to Offload? Scenario Two (System Modelling)

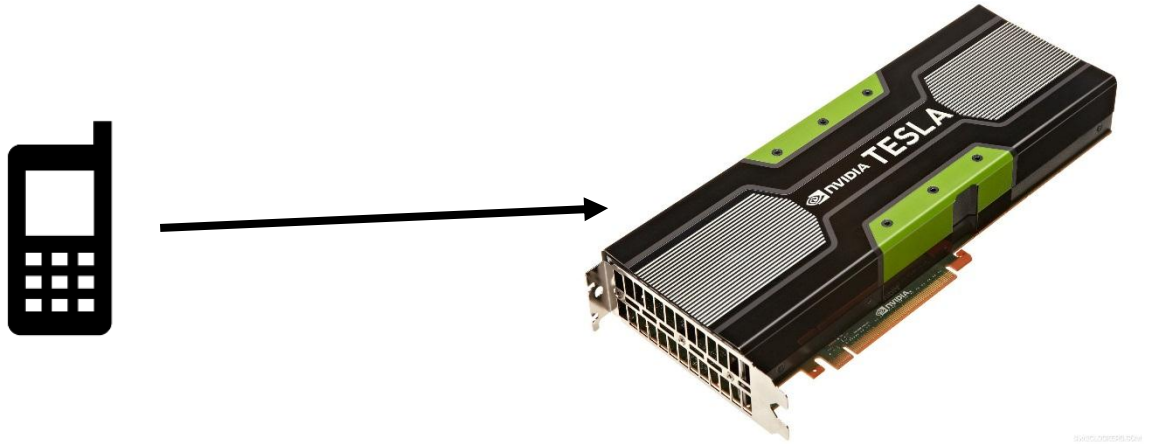
Planning Mobile Cloud Infrastructures Using Stochastic Petri Nets and Graphic Processing Units



Amazon's GPU (1536 cores): $2,6/h

Next Steps

- Implements offloading to GPU, in fact.



- Thanks