



PET Performance test documentation

Created by Vilmos Kozma



Contents

PET Performance test documentation	1
Created by Vilmos Kozma	1
1 Introduction	3
1.1 Hardware configuration	3
2 Testing Repast based HeatBugs	4
2.1 Test 1	4
2.1.1 Conditions	4
2.1.2 Results	4
2.2 Test 2	4
2.2.1 Conditions	4
2.2.2 Results	4
2.3 Test 3	4
2.3.1 Conditions	4
2.3.2 Results	4
2.4 Test 4	4
2.4.1 Conditions	4
2.4.2 Results	4
2.5 Test 5	4
2.5.1 Conditions	4
2.5.2 Results	5
2.6 Test 6	5
2.6.1 Conditions	5
2.6.2 Results	5
2.7 Conclusion	5
3 Testing the “Watch the Hunt” model	6
3.1 General conditions	6
3.2 1 running simulation	6
3.3 5 running simulations	6
3.4 10 running simulations	6
3.5 20 running simulations	6
3.6 30 running simulations	6
3.7 40 running simulations	6
3.8 50 running simulations	6
3.9 100 running simulations	6
3.10 Conclusion	6
4 Comparing the two test series	7

1 Introduction

The goal of this document is to introduce performance test results made with PET to let users be aware of hardware requirement and required software setup. Our aim is to monitor CPU load and point out the difference between expected and real applet refresh rate while the number of running simulations and connecting clients are differs in the different tests. We performed two separate test series since PET has two engines with relevant differences. One for the Repast engine and one for the native MAC engine. For monitoring the actual CPU load we used the `top` command.

1.1 Hardware configuration

- CPU: 4 x Xeon E5310, 1.60GHz
- Memory: 2Gb
- Operating system: Linux with 2.6 kernel

2 Testing Repast based HeatBugs

2.1 Test 1

2.1.1 Conditions

- Number of running simulations: 1
- Number of connection clients: 20
- Expected refresh rate: 600 ms

2.1.2 Results

- CPU load: 180%
- Real refresh rate: 580 ms

2.2 Test 2

2.2.1 Conditions

- Number of running simulations: 1
- Number of connection clients: 1
- Expected refresh rate: 100 ms

2.2.2 Results

- CPU load: 185%
- Real refresh rate: 129 ms

2.3 Test 3

2.3.1 Conditions

- Number of running simulations: 1
- Number of connection clients: 1
- Expected refresh rate: 200 ms

2.3.2 Results

- CPU load: 180%
- Real refresh rate: 227 ms

2.4 Test 4

2.4.1 Conditions

- Number of running simulations: 5
- Number of connection clients: 20
- Expected refresh rate: 400 ms

2.4.2 Results

- CPU load: 280%
- Real refresh rate: 433 ms

2.5 Test 5

2.5.1 Conditions

- Number of running simulations: 10

- Number of connection clients: 20
- Expected refresh rate: 400 ms

2.5.2 Results

- CPU load: 320%
- Real refresh rate: 416 ms

2.6 Test 6

2.6.1 Conditions

- Number of running simulations: 15
- Number of connection clients: 20
- Expected refresh rate: 400 ms

2.6.2 Results

- CPU load: 360%
- Real refresh rate: 862 ms

2.7 Conclusion

The CPU load seems to linearly increase with the number of simultaneously running simulations. the number of connecting clients has only few effect on performance thanks to image caching introduced in PET 1.3.0.

3 Testing the “Watch the Hunt” model

3.1 General conditions

- Number of connection clients: 20
- Expected refresh rate: 200 ms

In this test we only varied the number of running simulations.

3.2 1 running simulation

- CPU load: 15%
- Real refresh rate: 221 ms

3.3 5 running simulations

- CPU load: 26%
- Real refresh rate: 215 ms

3.4 10 running simulations

- CPU load: 48%
- Real refresh rate: 206 ms

3.5 20 running simulations

- CPU load: 109%
- Real refresh rate: 206 ms

3.6 30 running simulations

- CPU load: 200%
- Real refresh rate: 221 ms

3.7 40 running simulations

- CPU load: 367%
- Real refresh rate: 221 ms

3.8 50 running simulations

- CPU load: 373%
- Real refresh rate: 209 ms

3.9 100 running simulations

- CPU load: 370%
- Real refresh rate: 370 ms

3.10 Conclusion

Just as in the first series the number of simultaneously running simulation is linearly increases the CPU load. However, after a certain load level - around 370% in our case - the load level is not increasing anymore instead the simulations get slower.

4 Comparing the two test series

As the results points out the simulation featured in the second series (MAC based) performed better than the one based on Repast. Probably, its root cause is in connection with the model parameter "Sleep between steps", which was 0 in the first series and 10 ms in the second one. In the second case this 10 millisecond gave a rest time for the CPU while in the first case this has not happened at all.