Cloud Computing - Exercise 1
Amazon EC2 - Benchmarking

Florian Feigenbutz - Matriculation number
Tim Strehlow - Matriculation number
Till Rohrmann - 343756

May 9, 2012

2 Creating the AMI

We created our custom AMI by executing the following steps after setting up the AWS EC2 environment:

1. Find AMI with Ubuntu 64bit Instance-Backed:

```
flo@flo-ubuntu:~$ ec2-describe-images -a | grep ubuntu | grep 64 |
grep instance-store > /tmp/ec2-images

less /tmp/ec2-images
....
IMAGE   ami-1de8d369    099720109477/ubuntu/images/ubuntu
-precise-12.04-amd64-server-20120424   099720109477    available
public          x86_64  machine aki-62695816
instance-store  paravirtual     xen
...
```

2. Run new instance with found AMI

```
flo@flo-ubuntu:~$ ec2-run-instances ami-1de8d369 -n 1 -k ubuntu-development

RESERVATION r-904822d9 228315521052 default
INSTANCE i-8df134c5 ami-1de8d369 pending ubuntu-development 0
m1.small 2012-05-04T11:51:09+0000 eu-west-1c aki-62695816 monitoring-disabled
instance-store paravirtual xen sg-2c54a75b default
```

3. Show instances

```
flo@flo-ubuntu:~$ ec2-describe-instances i-8df134c5
```

```
RESERVATION r-904822d9 228315521052 default
INSTANCE i-8df134c5 ami-1de8d369 ec2-176-34-83-252.eu-
west-1.compute.amazonaws.com ip-10-63-1-191.eu-west-1.compute.internal
running ubuntu-
development 0 m1.small 2012-05-04T11:51:09+0000 eu-west-1c
aki-62695816 monitoring-disabled 176.34.83.252 10.63.1.191
instance-store paravirtual xen sg-2c54a75b
default
```

4. Log in using SSH

```
flo@flo-ubuntu:~$ ssh -i [KEY] ubuntu@ec2-176-34-83-252.eu-west-
1.compute.amazonaws.com
```

5. Install tools

```
ubuntu@ip-10-63-1-191:~$ sudo apt-get install gcc iperf build-essential
r-base ec2-api-tools ec2-ami-tools
```

6. Install benchmark scripts

```
flo@flo-ubuntu:~$ scp -i [KEY] CC_SS12_Blatt1_additional_material.zip
ubuntu@ec2-176-34-83-252.eu-west-1.compute.amazonaws.com:/home/ubuntu/
```

7. Set EC2 region

```
ubuntu@ip-10-63-1-191:~$ export EC2_URL=https://ec2.eu-west-1.amazonaws.com
```

8. Create AMI

```
ubuntu@ip-10-63-1-191:~$ sudo ec2-bundle-vol -k /tmp/pk.pem -u 228315521052
-c /tmp/cert.pem
```

9. Change region in AMI manifest

```
ubuntu@ip-10-63-1-191:~$ sudo ec2-migrate-manifest -c /tmp/cert.pem
-k /tmp/pk.pem -m /tmp/image.manifest.xml --region eu-west-1
-a ACCESS_KEY -s SECRET_ACCESS_KEY
```

10. Upload AMI

```
ubuntu@ip-10-63-1-191:~$ ec2-upload-bundle -b assignment1 -m
/tmp/image.manifest.xml -a ACCESS-KEY -s SECRET-ACCESS-KEY
```

11. Register AMI

```
flo@flo-ubuntu:~/Studium/CloudComputing/assignment1/additional_material
/benchmarks$ ec2-register assignment1-ami/image.manifest.xml -n
"AMI for Assignment 1"

IMAGE ami-8f80bbfb
```

12. Start m1.large instance

```
flo@flo-ubuntu:~$ ec2-run-instances ami-8f80bbfb -n 1 -t m1.large -k
ubuntu-development --region eu-west-1 --availability-zone eu-west-1a

RESERVATION r-90563cd9 228315521052 default
INSTANCE i-e111d4a9 ami-8f80bbfb pending 0 m1.large 2012-05-04T13:05:23+0000
eu-west-1a aki-62695816 monitoring-disabled instance-store paravirtual
xen sg-2c54a75b default
```

13. Describe instance

```
flo@flo-ubuntu:~$ ec2-describe-instances i-e111d4a9

RESERVATION r-90563cd9 228315521052 default
INSTANCE i-e111d4a9 ami-8f80bbfb ec2-176-34-86-75.eu-west-1.compute
.amazonaws.com ip-10-58-55-7.eu-west-1.compute.internal running
0
m1.large 2012-05-04T13:05:23+0000 eu-west-1a aki-62695816
monitoring-disabled 176.34.86.75 10.58.55.7 instance-
store paravirtual xen sg-2c54a75b default
```

14. Login via SSH

```
flo@flo-ubuntu:~$ ssh -i [KEY] ubuntu@ec2-176-34-86-75.eu-west-
1.compute.amazonaws.com
```

3 Single Instance Benchmarks

We used the following commands to start the 3 differently typed instances:

1. Find the custom AMI

```
rohrmann@nb-rohrmann:~$ ec2-describe-images
IMAGE ami-8f80bbfb 228315521052/AMI for Assignment 1 228315521052
available private x86_64 machine aki-62695816 instance-store paravirtual
xen
```

2. Starting a small, medium and large instance

```
rohrmann@nb-rohrmann:~$ ec2-run-instances ami-8f80bbfb -k
ubuntu-development -t m1.small --region eu-west-1 --availability-zone
eu-west-1a
rohrmann@nb-rohrmann:~$ ec2-run-instances ami-8f80bbfb -k
ubuntu-development -t m1.medium --region eu-west-1 --availability-zone
eu-west-1a
rohrmann@nb-rohrmann:~$ ec2-run-instances ami-8f80bbfb -k
ubuntu-development -t m1.large --region eu-west-1 --availability-zone
eu-west-1a
```

3. Finding out the URIs to connect onto the machines

```
rohrmann@nb-rohrmann:~$ ec2-describe-instances

...
```

4. Connect via SSH onto machines

```
rohrmann@nb-rohrmann:~$ ssh -i [KEY] ubuntu@URI
```

5. Run benchmarks

```
ubuntu@ip-10-63-1-191:~$ unzip CC_SS12_Blatt1_additional_material.zip;
cd additional_material/benchmarks/; ./linpack.sh; ....
```

6. Shut down medium instance

```
flo@flo-ubuntu:~$ ec2-describe-instances

RESERVATION r-90563cd9 228315521052 default
INSTANCE i-e111d4a9 ami-8f80bbfb ec2-176-34-86-75.eu-west-1.compute
.amazonaws.com ip-10-58-55-7.eu-west-1.compute.internal running
0
m1.medium 2012-05-04T13:05:23+0000 eu-west-1a aki-62695816
monitoring-disabled 176.34.86.75 10.58.55.7 instance-
store paravirtual xen sg-2c54a75b default

flo@flo-ubuntu:~$ ec2-terminate-instances i-e111d4a9
```

5 Evaluation of Benchmark Results

Based on your benchmarks and your knowledge about XEN paravirtualization, provide short answers to the following questions.

5.1 LINPACK Benchmark

5.1.1 Find out what the LINPACK benchmark measures (try google :-). Would you expect paravirtualization to affect the LINPACK benchmark? Why?

The LINPACK benchmark measures how fast a computer solves linear algebra calculations, LU decomposition and solving a system of linear equations. Its results are returned in number of floating point operations per second (FLOPS). As these tasks involve only unprivileged instructions and thus are basically all executed on the CPU we would not expect paravirtualization to affect the benchmark results as long as the virtualized machine has dedicated access to the host's resources.

5.1.2 Look at your LINPACK measurements. Are they consistent with your expectations. If not, what could be the reason?

Among the paravirtualized hosts we recognize a performance increase of 250% between the small and medium instance. Between medium and large there is no notable difference whereas we would also have expected to also see a similar increase in performance as the instances are supposed to have different number of AWS computing units [1]. The reason for this result could be that the benchmark only runs on a single core and therefore could only make use of a single of both provided EC2 computing units. This would explain the nearly identical results of medium and large as they both provide the same EC2 computing units with the only difference that medium only offers a single while large offers two cores. Establishing a relation to the "novirt" results is difficult as we don't know the exact hardware of the host hardware used for our AWS benchmarks. We can only state that the single E2 Computing Unit provided in our small instance [1] approximately provides 1/4 of the linear algebra computation power of an Intel(R) Xeon(R) CPU X5355 @ 2.66GHz CPU which had been used for the "novirt" benchmarks.

5.2 Memsweep Benchmark

5.2.1 Find out how the memsweep benchmark works by looking at the C code. Would you expect paravirtualization to affect the memsweep benchmark? Why?

This benchmark measures the required time to access (write and clean) heap memory at different locations. The step width between to consecutive accesses is chosen such that a cache miss occurs and the data really has to be loaded from the memory. Because the hypervisor still needs to validate write requests we expect the results to show a slower performance of all virtual machines compared to the "novirt" test results. Moreover, differences between the paravirtualized machines should be the result of different memory and a different clock speed of the actual CPU.

5.2.2 Look at your memsweep measurements. Are they consistent with your expectations. If not, what could be the reason?

The sweep time of the "novirt" machine is considerably slower than the results of the paravirtualized machines. This is somehow surprising and implies that the Amazon hosts have a much faster memory than the reference computer "novirt". Comparing the results among the virtualized systems shows that the small machine is about 3 times slower than the medium and the large machine, which have nearly identical results. These differences could possibly be caused by different hardware configurations. The equality of the sweep times between the medium and the large machine suggest the conclusion that both machine types are using the same memory hardware. Again, the guest system cannot benefit from the second core available on the large machine.

5.3 Syscall Benchmark

5.3.1 Find out how the syscall benchmark works by looking at the C code. Would you expect paravirtualization to affect the syscall benchmark? Why?

The syscall benchmarks executes 50 million times the same syscall which returns the thread group id of the current process. As paravirtualization uses exceptions and offers the guest OS to provide exception handlers for performance reasons this benchmark should perform much quicker than on fully virtualized machines. Compared to an unvirtualized host we expect small performance drawbacks because of the required execution power to create software exceptions. Among the three test instances there should be notable differences between all instances.

5.3.2 Look at the syscall measurements. Are they consistent with your expectations. If not, what could be the reason?

The results show that the medium instance runs the benchmark within less than half the time the small instance requires. The nearly doubled clock rate and the faster memory of the medium machine could be the reason for the difference. On the other side there is no difference between medium and large which could be explained by the test running again on a single core only.

5.4 Fork Benchmark

5.4.1 Find out how the fork benchmark works by looking at the C code. Would you expect paravirtualization to affect the fork benchmark? Why?

This benchmark measures the time it takes to duplicate the benchmark process itself for 1 million times. Since a fork requires the execution of privileged instructions and write operations to the memory we would assume that the paravirtualized machines are slower than the "novirt" reference system. The actual execution of the operations depends on the clock rate and the access time to the

memory. Thus, the small instances should be slightly slower than the medium and the large instances.

5.4.2 Look at the fork measurements. Are they consistent with your expectations. If not, what could be the reason?

The results show that the "novirt" system is about 3 times slower than the small instance and 4 times slower than the medium instance. That was not expected but could be attributed to a different hardware configuration. Moreover, experiments showed that executing the benchmark with root privileges slows the benchmark considerably down. The measurements on the virtual machines were all done in the context of a normal user. It might thus be the case that the "novirt" measurements can be explained by that. The performance increase between the small and medium instance is probably the result of the more powerful hardware of the latter instance type. However, the measurements of the large instance, which is 6 times slower than the "novirt" machine and 24 times slower than the medium instance, are quite surprising and hard to explain. Since the guest systems are identical and the hardware of the large machine is the most powerful among those tested, the results have to be connected to the hypervisor and concurrently running systems. They have to interfere with each other in a way such that their performance is massively degraded.

5.5 Disk Benchmark

5.5.1 Find out how the disk benchmark works by looking at the shell code. Would you expect paravirtualization to affect the disk benchmark? Why?

The disk benchmark copys 1 million blocks from /dev/xvda1 to /dev/null, 4096 bytes per block. We expected the io-access from a paravirtualized machine to be slower than a non-virtualized one, but compared to a full-virtualized solution it should be faster. Among the three instances there should be performance increases on the more powerful instances. As Amazon states the io-performance of both the small and the medium instances should be "medium" whereas the large instances should have a "high" performance.

5.5.2 Look at the disk measurements. Are they consistent with your expectations. If not, what could be the reason?

Among the three virtualized instances we see an increasing disk read performance as they get more powerful as expected. We have a larger gap between the small and the medium instance's average value than between the medium and the large one. So therefore Amazon's io-classification might not be that meaningful, whereas changing from 1 to 2 computing units have increased the performance more noticeable than the change to 4 units. The difference between the novirt value and the virtualized one might be due to more powerful hard disks used in the ec2.

5.6 Network Throughput Benchmark

5.6.1 Find out how the network benchmark works by looking at the shell code (the iperf manpage should be helpful). Would you expect paravirtualization to affect the network benchmark? Why?

The network benchmark starts a simple iperf server on the one machine and generates traffic to it from the other machine. The script runs the iperf client in different configurations with varying buffer sizes from 16 to 4096 KB. The client also has an options that disables Nagle's algorithm in TCP, so that the packets are sent immediately. We expect that the virtualized machines perform worse than the "novirt" system due to the virtualization overhead. Moreover, the large instance should reach a higher throughput because of its better io-performance [1].

5.6.2 Look at the network throughput measurements. Are they consistent with your expectations. If not, what could be the reason?

In general we have smaller throughput values for the virtual instances than for the non-virtual. This could be due to the slower io-virtualization. For up to a buffer size of 512 KB we have nearly the same values for small and large instances. But the large instances can handle larger buffer sizes better due to the fact that they get more io-access assured by amazon, so they get prioritized. The small instance reaches its peak rate at a buffer size of 1024 KB, whereas the large instance reaches the best value with a 2048 KB buffer. Both instances' throughput decreases a little when using larger buffers.

References

[1] *Amazon EC2 Instance Types*. http://aws.amazon.com/ec2/instance-types/.