Department of Computer Sciences
University of Salzburg

HPC In The Cloud?

Seminar aus Informatik
SS 2011/2012

July 16, 2012

Michael Kleber, mkleber@cosy.sbg.ac.at

Contents

1 Introduction 2
2 Benchmarks and Analysis 3
2.1 Network and Compute Performance 3
2.2 Performance Variability 6
2.3 Parallel I/O 7
2.4 Cost Analysis 9
3 Application Benchmarks 11
3.1 MUSCLE 11
3.2 NASA 12
4 Conclusion 16
5 References 17

1 Introduction

Cloud services became widely available in the last few years. They provide easy access to compute resources and with their flexibility and pay-as-you-go price model, they have become an interesting computing platform. These services are mainly used in business and web applications, but recently got attention from the HPC community, especially since the release of products targeted at HPC users like the Cluster Computer Instances from Amazon.
Several studies in the last five years have shown, that cloud services cannot compete with local cluster systems in terms of performance. It has been discovered that EC2 was up to 20 times slower than local clusters when running tightly coupled programs.[1] In this paper, I will take a brief look at recent results from different studies and I will answer some questions regarding the feasibility of running HPC applications in public cloud services.

These questions are:

- Can the cloud compete with local cluster performance?
- For which applications are cloud services suitable?
- Are there any drawbacks?
- Is the cloud cost effective?

There are architectural differences between local HPC clusters and the usual compute cloud services. First virtualisation, while applications on clusters usually run on bare metal, cloud service provider use virtualisation technologies for their platforms e.g. for security reasons. There is no guarantee on hardware specification. To be precise the processor type is usually not specified, but computation performance might be specified (Amazon's Compute Units). This means that one cannot use architecture specific compiler optimisations. The next difference is the interconnection between nodes, on cloud services we have 1Gb or 10Gb Ethernet with an unknown topology, on local clusters the interconnect can be optimised for the running applications, e.g. InfiniBand with hypercube or 2D-mesh topology. Aside from that real network capacity and quality are not known and might lead to performance variation. We will have a look at that in section 2 and will also (indirectly) see, if virtualisation overhead has an impact on performance.

For this paper I decided to focus on the well-known and leading IaaS provider Amazon with their EC2, because it is public and available to all, they have a product targeting HPC users, called Cluster Compute Units (CCI) and there is just more literature available. In addition to that I believe EC2 is representative for today's cloud services.

Amazon's Cluster Compute Quadruple Extra Large specification:

[1] Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud

- 33.5 EC2 Compute Units (one CU is equal to a 2007 Opteron or Xeon with 1 to 1.2 GHz)
- 23 GB of RAM
- 1690 GB of local instance storage
- 64 Bit platform
- 10 Gb Ethernet

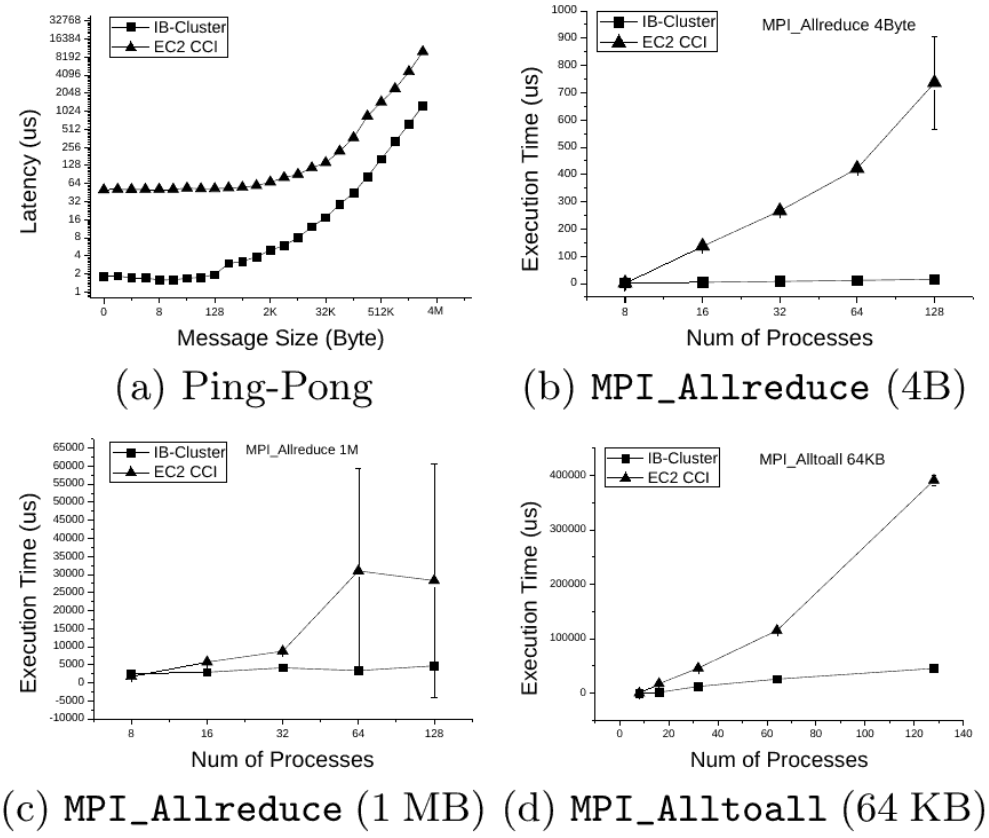
As the local storage (LBS local block storage) is not permanent, the users use Elastic Block Storage (EBS), which can be attached to Instances and accessed through the S3 web service.
Well in all tests so far, the hardware the CCI's run on, are two-way Xeon X5570 processors from Intel based on the Nehalem architecture. The CCI and CGI (Cluster GPU Instances) are in fact the only Instances where the underlying architecture is specified in their definitions.

2 Benchmarks and Analysis

2.1 Network and Compute Performance

In this section we will take a brief look at the results from [1] using two synthetic benchmarks to obtain the network and computing performance of Amazon EC2 CCI running Amazon Linux in comparison to a local cluster system.
The cluster had two Intel Xeon X5670 six core processors and 32 GB of RAM per node. The nodes were connected using InfiniBand QDR. NFS was used as shared file system and the cluster was running Red Hat Enterprise Linux 5.4.

Many HPC applications use libraries implementing the Message Passing Interface, e.g. OpenMPI. For tightly coupled applications, latency and throughput of MPI messages are important and so is networking performance. First we take a look at Intel MPI benchmarks:



(a) Ping-Pong (b) `MPI_Allreduce` (4B)

(c) `MPI_Allreduce` (1 MB) (d) `MPI_Alltoall` (64 KB)

Figure 1: Intel MPI Benchmarks

With the ping pong test and message sizes up to 128 KB we are able to see the significant difference in latency between the 10Gb Ethernet interconnect used by Amazon and the InfiniBand interconnect used by the local cluster. With increasing message sizes, the difference is becoming smaller because of the higher bandwidth usage. The local cluster is faster in MPI_Allreduce tests. Graph (b) and (c) show network performance variation with 128 processes in (b) and with 64 and 128 processes in (c) (see vertical bars). MPI_Alltoall with 64 KB message size shows normal behaviour and the local cluster is faster.

NBP or NAS Parallel benchmark is a suite consisting of eight benchmarks derived from computational fluid dynamics (CFD) to evaluate performance of cluster and supercomputer systems. For further information see [2]. The first figure shows the execution times for every benchmark using benchmark class D.

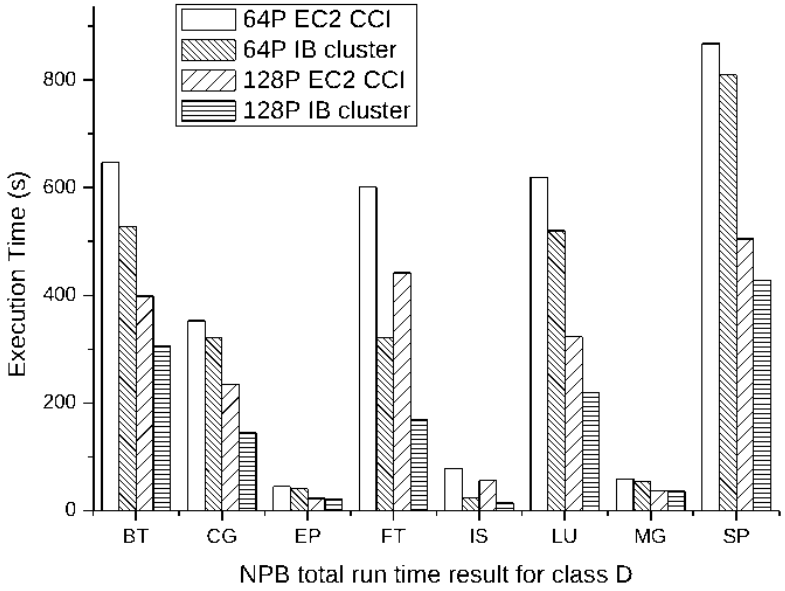


Figure 2: NAS Parallel Benchmark execution times

EP (Embarrassingly Parallel) turned out to be the most competitive on the cloud, while FT (discrete 3D fast Fourier Transform) and IS (Integer Sort) turned out to be the worst case scenario for the Amazon cloud. FT does a lot of MPI_Alltoall calls, IS does random memory access and a lot of communication too. This seems to be the reason for mediocre performance. The CCI is about 23% slower than the cluster.

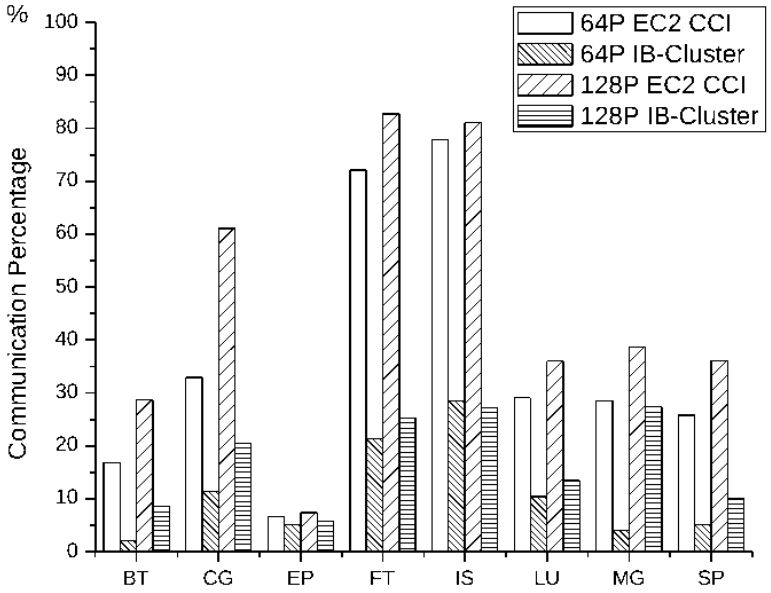


Figure 3: NAS Parallel Benchmark communication overhead

We can see that communication time takes about 75% to 85% of the total execution time running FT and IS on EC2. The communication times on the cluster is about a third of EC2. EP2 does not have much communication and communication times are about the same on both systems.

2.2 Performance Variability

In the past, performance variability has been an issue when running HPC application in the cloud. These variations in network or node performance can lead to load balancing problems and thus to performance decrease when running tightly coupled applications. So far, we saw performance variations in section 2.1. The following benchmarks are taken from [1]. The tests have been performed on different times of the day for several days.

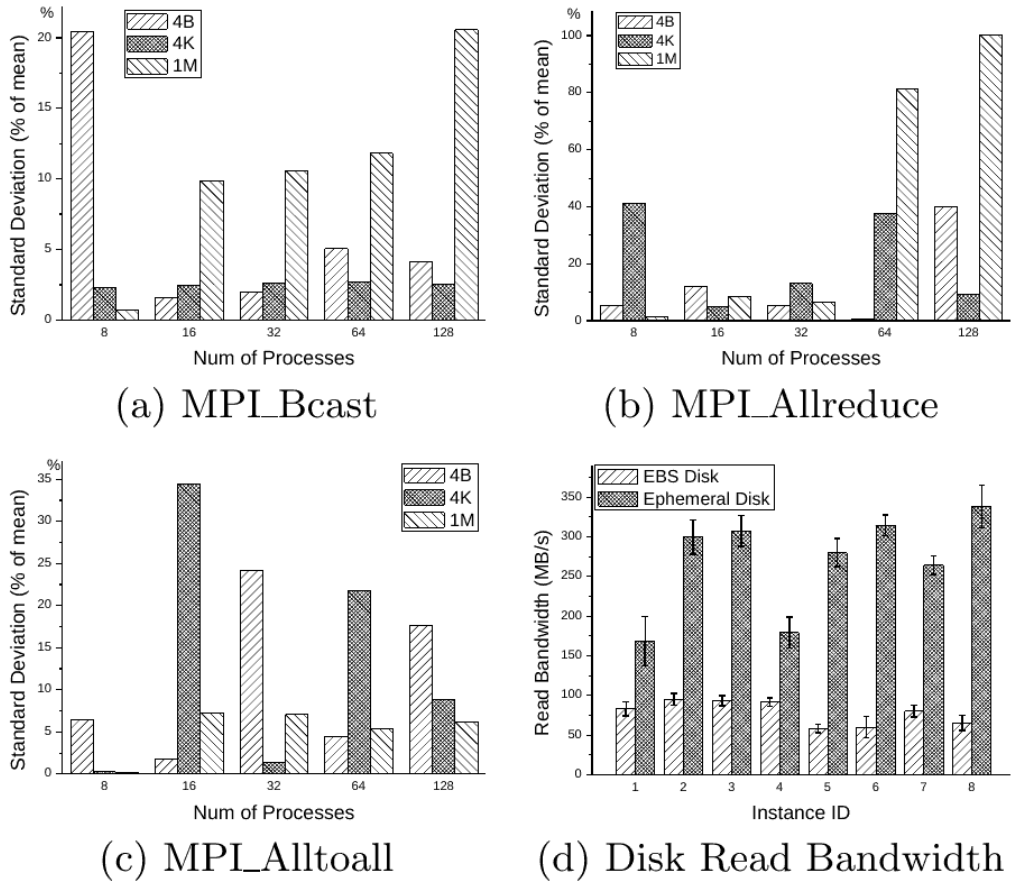


Figure 4: Performance variability of EC2 CCI

MPI Broadcast EC2 performance shows variability with small message size only with eight processes. With increasing number of processes, larger messages show higher variability. MPI_Allreduce shows significant variability with up to 100%. This makes optimisations harder and may lead to load balancing problems. MPI_Alltoall shows up to 35% of mean time.
Disk read performance is determined using hdparm and is quite stable for EBS across

Instances. It is interesting, that the local disks perform different on different instances. This could be a problem when using parallel file systems.

On a side note, a CCI does not support TCP/IP offloading and has to be handled by the CPU. By default all IRQs are directed to the same core and that cannot be changed by a user. This might lead to a load balancing issue. An easy way to avoid this has been suggested by [1] by simply using a round robin algorithm to schedule the processes on an instance.

2.3 Parallel I/O

HPC applications usually depend on a shared file system. NFS is a popular shared FS especially on small and mid range clusters, but parallel FS are used too. The latter are popular on large clusters. Some examples for such FS are PVFS, GPFS and Lustre.
One interesting characteristic of cloud services is the storage system. On local clusters the FS is usually static. Using cloud services, it is possible to change the file system between sessions and thus one could choose the right file system for the task. In this section we will look at parallel FS benchmarks from [1]. The setup includes PVFS FS using one, two and four dedicated I/O servers and one instance as NFS server in asynchronous mode.

The first benchmarks are IOR runs. Every Instance runs eight processes and reads/writes a 16 GB file collectively with MPI-IO.

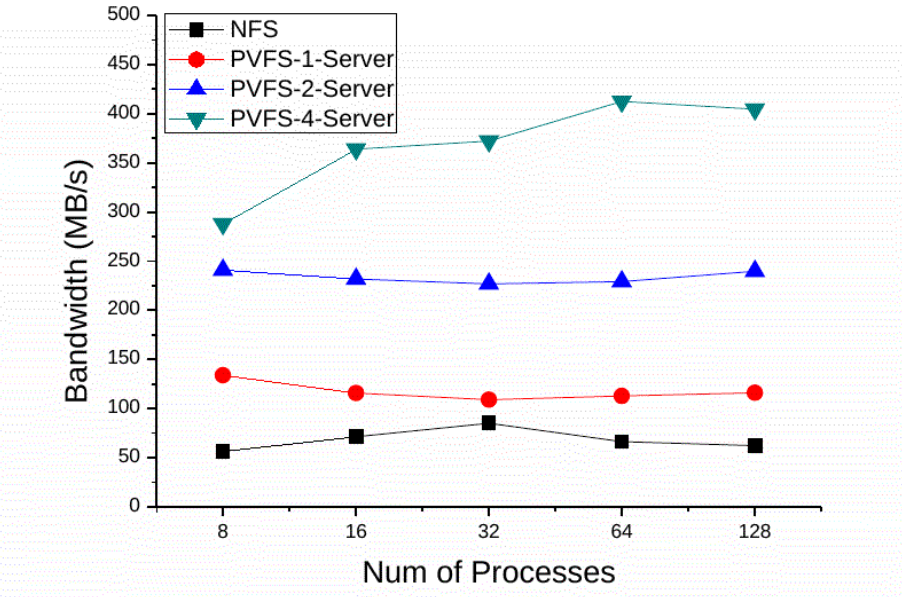


Figure 5: IOR read rates

Read bandwidth is higher on PVFS than NFS, even when just one server is used. Available bandwidth scales up when attaching new servers.

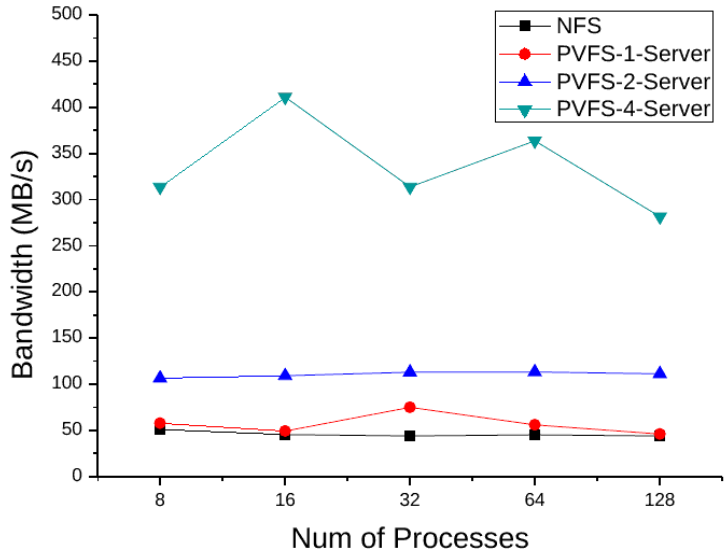


Figure 6: IOR write rates

Write bandwidth does not show an advantage for the single PVFS server over NFS, which was a surprise to me. The bandwidth scaling across instances of the 4-server setup is interesting. This might be caused by the network infrastructure. The higher bandwidth might be a result of the higher combined buffers.

Secondly we will look at BT-IO (class C problem size and SUBTYPE I/O size) performance on both FS.

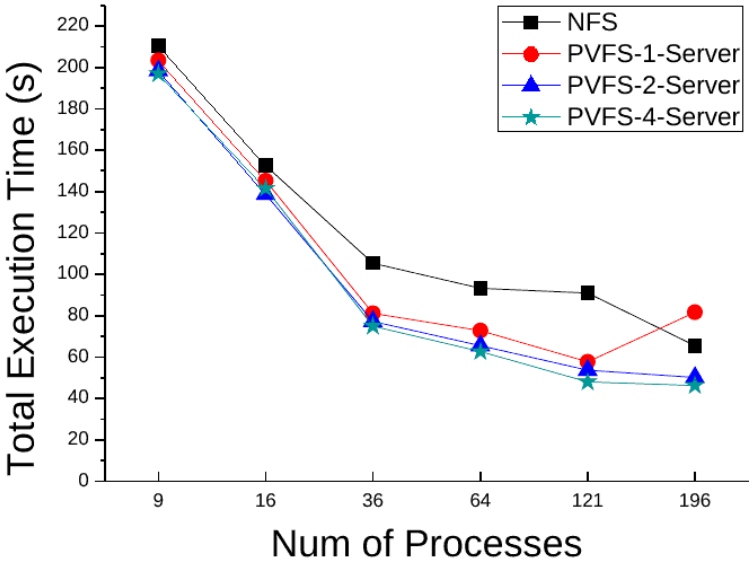


Figure 7: BT-IO completion times

We can see an execution time reduction when using PVFS. With 121 processes it takes

37% less time to complete with a single PVFS server compared to NFS. It needs more processes to gain a bigger plus from multiple PVFS servers, though two servers make sense with 196 and more processes.

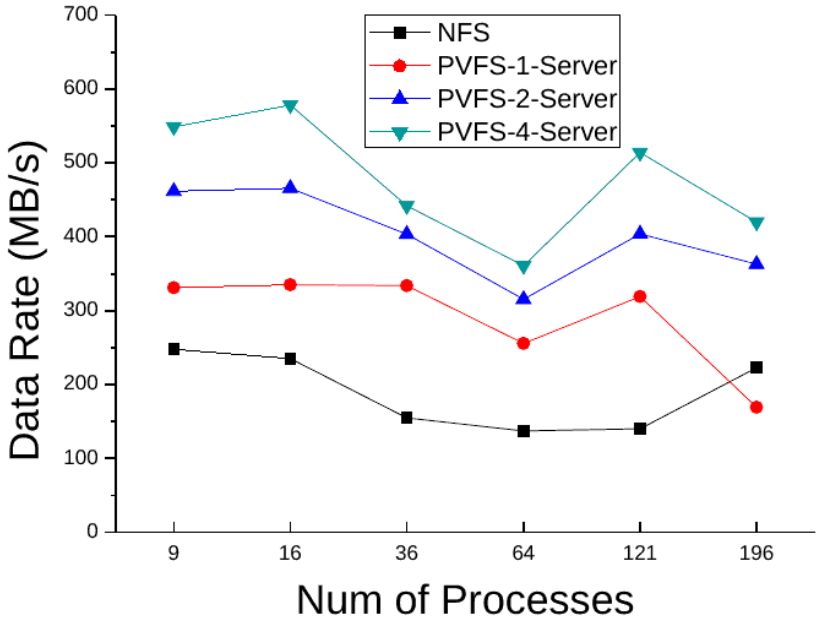


Figure 8: BT-IO data rates

We see a minimum data rate at 64 processes. In general, the computation and I/O are executed in interleaved mode and that is why BT-IO manges to have higher bandwidth than IOR.

On a side note, the above setup took just a few bash scripts with 200 lines of code each, so this should not be a problem for a cluster administrator.

On a second side note, costs running the benchmarks in the cloud and on the local cluster have been about the same.[1]
However I wanted to present a different cost study.

2.4 Cost Analysis

In this section, we will discuss a case study done at Perdue [4]. They are running two community clusters. A community cluster is a system that is owned by a faculty and used by the broader campus community. At the time of writing, they had a 902-node Intel Xeon E5410 cluster (7216 cores) with 1Gb Ethernet as interconnect called Steele and a 993-node AMD Opteron 2380 cluster (7944 cores) with 10Gb Ethernet as interconnect called Coates.

A node-hour (or instance-hour) on EC2 CCI costs $1.6, storage and data transfer add about $0.10-$0.15 per node-hour, so total cost for one node-hour is around $1.70 to $1.75.

The community cluster had high utilisation rates, typical clusters out there have usually around 20-30% utilisation. Steele had 68.1% and Coates 80.1%.

The EC2 CCI equivalent cost results summarise up to:

Resource	Q2 2010 Core-Hours	Nodes Purchased	TCO per Node	EC2 Equivalent / Node-Hour
Steele	10,815,169	902	$7,643.10	$0.42
Coates	13,936,475	993	$8,315.82	$0.40

Figure 9: Community cluster costs

TCO is the total cost of ownership and includes staffing, utilities, purchase costs etc. Disk storage shared across clusters is not included.

Such clusters are usually running four to five years, which leads to lower computation performance compared to CCI Quadruple Extra Large.

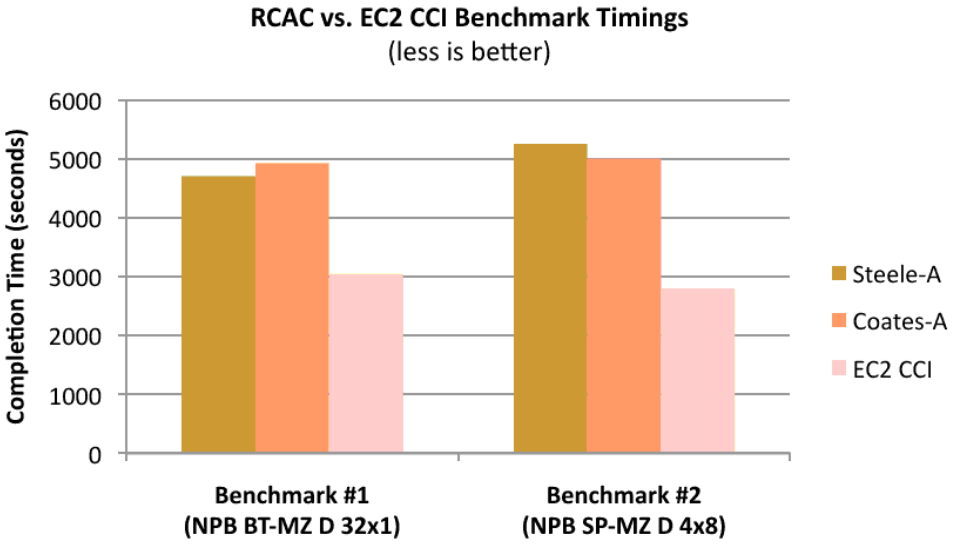


Figure 10: NAS Parallel Benchmark results

A subset of NAS Parallel Benchmark suite with class D has been used to determine the data above. As one can see, CCI outperforms the two clusters by roughly 41%.

This is just one example. The actual costs of a local cluster depend on many factors. Those factors include organisation size, IT staff structure and size of the user base. The key for low costs is (stable and) high utilisation. If users of a local cluster cannot assure high utilisation, cloud services might be an option. It all depends on the applications, loose coupled ones should give good results on clouds while tightly coupled ones might have performance problems.

3 Application Benchmarks

This section gives an overview of some real-world application performance of Amazon EC2 CCI, which include tightly and loose coupled applications. Testing such applications is especially interesting because we can see if the assumptions from section 2 based on synthetic benchmarks are accurate.

3.1 MUSCLE

One example for loose coupled applications are multi-scale simulations. Such simulations are widely used for simulations of e.g. blood flow, stellar systems, virtual reactors or solid tumour model. These results are from [4] using a framework for constructing multi-scale applications, called the Multi-scale Coupling Library and Environment or MUSCLE, by single kernel which are connected by unidirectional conduits.

Aside from Cluster Compute Quadruple Extra Large Instances, they benchmarked High CPU Extra Large Instances with eight virtual cores with 2.5 Compute Units each, totalling 20 Compute Units as well as 7 GB of RAM and 1690 GB of local storage and 64 Bit Linux. The local cluster was a HP Cluster Platform with Intel Xeon 2260MHz connected with InfiniBand. This system was on the 81st place in the Top500 list in June 2011. Such systems usually have some kind of management system for task queuing and dispatching. In this case they included the PBS-Queue wait time.
In this benchmark they used an in-stent restenosis application as a multi-scale application example. It consists of three modules, blood flow, muscle cell simulation and drug diffusion for simulation of restenosis of artery after surgery. The modules send and receive a total of about 10 MB of data each iteration, but the amount of data exchanged during execution is just in the KB range. Two partial runs have been done, one with 15 iterations and one with 150.

ISR 2D 15 iterations								
Infrastructure	**Setting up**		**Execution**		**Sending Output**		**Total**	
	min - max (sec)		avg (sec)	σ			min - max (sec)	
Local HPC Cluster	6 - 363		190	16	N/A		196 - 553	
	avg(sec)	σ	avg (sec)	σ	avg(sec)	σ	avg(sec)	σ
AWS Cloud								
m1.xlarge	81	6	250	20	100	10	430	20
m2.4xlarge	80	10	187	3	130	20	400	20
ISR 2D 150 iterations								
Infrastructure	**Setting up**		**Execution**		**Sending Output**		**Total**	
	min - max (sec)		Avr (sec)	σ			min - max (sec)	
Local HPC Cluster	6 - 363		1500	130	N/A		1506 - 1863	
	avg(sec)	σ	avg (sec)	σ	avg(sec)	σ	avg(sec)	σ
AWS Cloud								
m1.xlarge	72	4	2068	15	120	20	2260	30
m2.4xlarge	74	4	1526	4	110	60	1710	60

Figure 11: Performance comparison MUSCLE

The cluster setting up phase includes PBS-Queue waiting and dispatching time and EC2 setting up includes everything needed to run the application. Sending output means sending the results to S3, which is not necessary on a local cluster due to its shared file system. As shown in figure 11, the waiting time on the cluster can be as high as six minutes and as low as three seconds. That alone makes the completion time on EC2 more predictable and can be longer than the setting up and sending output phase on EC2 combined on both Instances. Execution times on the local cluster and the EC2 Compute Instance are about the same for 15 and 150 iterations, only the smaller EC2 Instance, as expected, cannot compete with the local cluster. However, the execution time still looks promising for this Instance.
The EC2 CCI is on par with the local cluster and is therefore a feasible alternative.

3.2 NASA

The next round of benchmarks was done by [5] using four real world applications from NASA's programs. All programs have more or less similar performance characteristics, but differ when it comes to the point of scalability.
For comparison NASA's Pleiades local cluster was used along Amazons EC2 Cluster Compute Quadruple Extra Large Instances. Both systems used the same OS (SLES11SP1) and compiler (Intel 12.04) but different MPI libraries. On Pleiades it was MPT 2.04 and on Amazon OpenMPI 1.4.4. The reason for this choice was that the authors wanted to use the best environments for each platform and thus SGI MPI was a better choice for the local cluster. Pleiades itself is an SGI ICE cluster using dual Xeon X5570 Nehalem quad cores, the same processors Amazon uses for their CCI nodes, with 24 GB of RAM and InfiniBand 4XQDR as interconnect with hypercube topology. On Amazon NFS was the file system of choice while Pleiades was running Lustre, a well-known parallel file system.

The first application for benchmarking was Enzo version 2.0. It was written to do simulations of cosmological structure formations.[2] Initial state has been read in from text files and the cosmos was allowed to evolve.

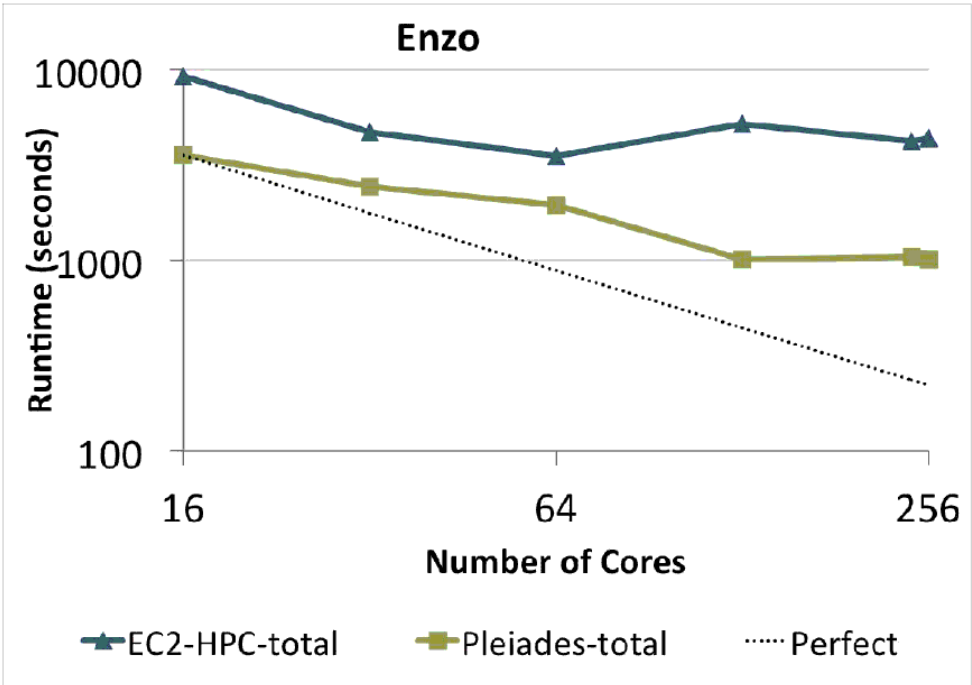


Figure 12: Enzo performance comparison

As can be seen, Enzo does not scale perfect, but scales better on Pleiades and runs faster on 16 cores. The performance difference ranges from 2.6 to 4.3 times slower on 16 to 32 cores.

Cart3D[3] version 1.3.5 is an analysis program for conceptual and preliminary aerodynamic design (CFD application). It solves the Euler equations of fluid dynamics. For this benchmark the geometry of the Space Shuttle Vehicle has been used for the simulations. Input data size was 1.8 GB and the applications needed up to 16 GB of memory.

[2]http://enzo-project.org/
[3]http://people.nas.nasa.gov/ aftosmis/cart3d/

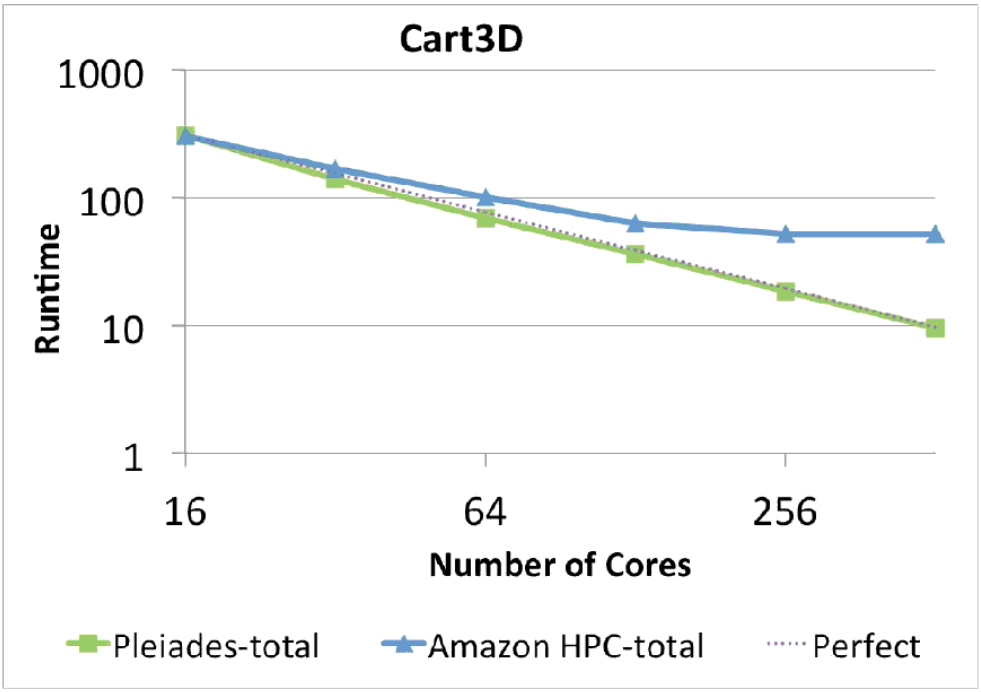


Figure 13: Cart3D results

EC2 is nearly on par with Pleiades when run at lower core count (mainly 16 cores). It suffers degradation when the core count rises. The local cluster on the other hand shows excellent scaling up to 256 cores. Performance difference varies between 1.2 at 32 cores to 2.8 times slower at 256 cores.

MITgcm, the MIT General Circulation Model from the ECCO project (Estimating the Circulation and Climate of the Ocean) is able to simulate fluid phenomena over a wide range of scales.[4] It is used to study the ocean, climate and atmosphere. It makes heavy use of the MPI_Allreduce function. They used a quarter degree two-day-long global ocean simulation. It writes 8 GB of data, needs 20 GB of disk and 32 GB of RAM.

[4] http://mitgcm.org/

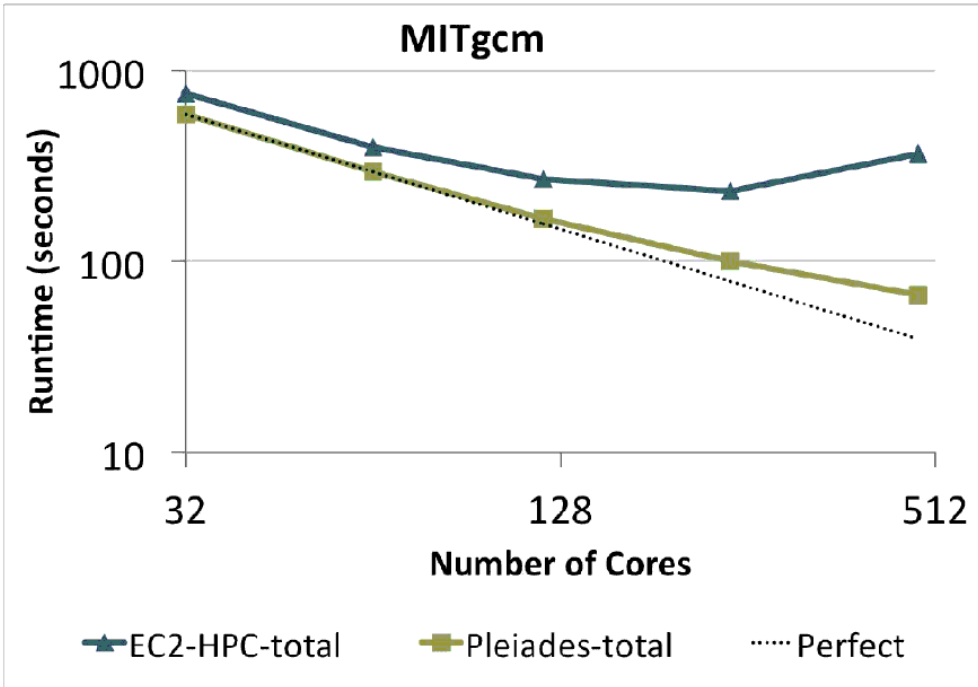


Figure 14: MITgcm on EC2 and Pleiades

MITgcm performs similar to Cart3D. It scales excellent on Pleiades up to 480 cores, but EC2 cannot scale beyond 256 cores. Thus, the local cluster is 1.2 times to 5.4 times faster than the Cluster Compute Instances.

Discrete Dipole Scattering (DDSCAT) is a program for calculating scattering and absorption of light by irregular particle using discrete dipole approximations. It is highly parallel and scalability is only limited by the number of orientations. The code uses FFT.[5]

[5] http://code.google.com/p/ddscat/

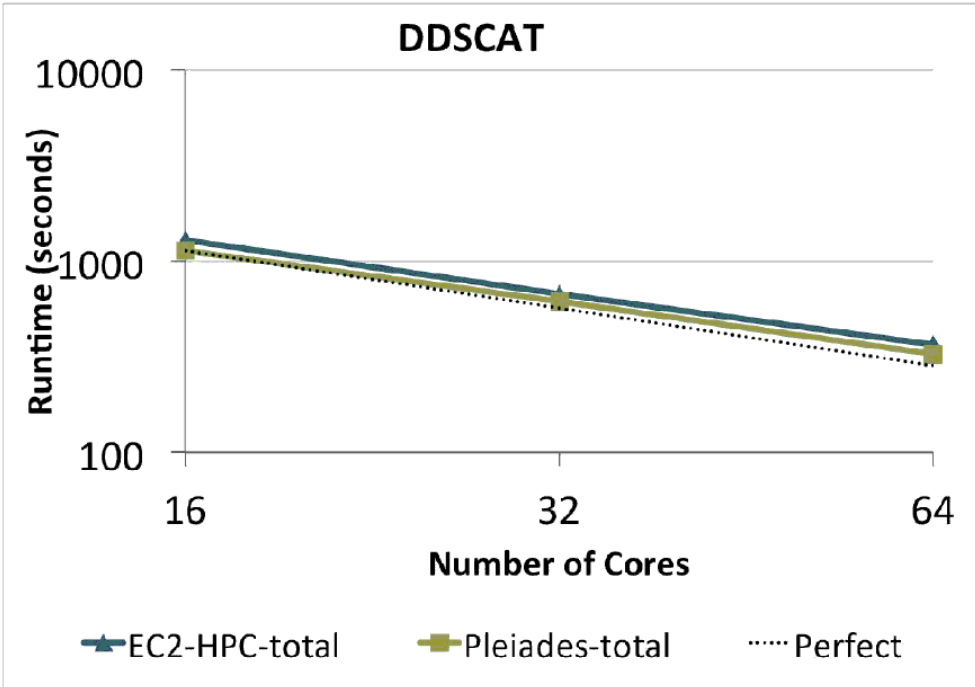


Figure 15: DDSCAT test results

As a program with small communication overhead, it performs excellently on both, EC2 CCI and Pleiades. The latter is about 11% to 13% faster, so in the same area.

4 Conclusion

In general, cloud services offer lower maintenance overhead. They are easy to maintain for experienced administrators, but it can be a bit tricky for users. Users are able to install additional software on Instances (they have root access). Therefore, there is no need to get an administrator to do this. The setup scripts for parallel I/O in section 2.3 are just about 200 lines of code long. The possibility to easily change the file system between sessions is a unique feature of cloud services.

Cloud services provide cost-effective computation resources as long as the utilisation of the in-house cluster is not high, which is not so common aside from community clusters.

We saw that running loose coupled application on EC2 is feasible, tightly coupled application on the other hand run slower on the cloud and the local cluster is still the better option. For loose coupled application we saw more stable results due to lack of PBS-Queue waiting time. One issue remains, the 10Gb Ethernet interconnection is still a bottleneck. Latencies are too high for at least tightly coupled applications with many small messages. The lack of TCP/IP offloading in conjunction with the interrupt handling might lead to load imbalance when all cores of a node are full utilised by MPI processes. Performance variability of the network and storage system is still an issue too, but not as dramatic as it has been in the past, it can be challenging to design and tune parallel programs with

lots of I/O operations.
At the end of the day, cloud services showed to be more competitive now than they were in the past.

5 References

1. Cloud Versus In-house Cluster: Evaluation Amazon Cluster Compute Instances for Running MPI Applications

2. http://www.nas.nasa.gov/publications/npb.html

3. Cost-effective HPC: The Community or the Cloud?

4. Comparison of Cloud and Local Cluster HPC approach for MUSCLE-based Multi-scale Simulations

5. Performance Evaluation of Amazon EC2 for NASA HPC Applications