Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Masterarbeit

Analysis of Elastic Cloud Solutions in an HPC Environment

vorgelegt von

Johannes Coym

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Studiengang:	M. Sc. Informatik
Matrikelnummer:	6693524
Erstgutachter:	Jun.-Prof. Dr. Michael Kuhn
Zweitgutachter:	Prof. Dr. Thomas Ludwig
Betreuer:	Jannek Squar
	Jun.-Prof. Dr. Michael Kuhn

Hamburg, 2021-10-23

Abstract

Cloud Services, like AWS, Azure, and Google Cloud, are a growing market, and all of them also started providing their cloud services for HPC use cases in the last years. Amazon even developed their own fabric adapter for AWS, claiming greatly improved load distribution compared to existing solutions.

This thesis aims to evaluate the current performance and profitability of cloud services for HPC applications. Specific focus will be on the rentability of running several specific or even all applications in the cloud. For this cloud side, AWS, Azure and Google Cloud will be used for profitability analysis. These cloud providers will have to compare to an on-premise HPC cluster for different job configurations. On the side of the on-premise HPC cluster, the cluster usage of a whole year of one cluster will be analysed to gain the required data. Additionally, the costs of running those variants will be compared on a typical lifespan of an HPC cluster with all of its acquisition costs and running costs.

These cost factors and node requirements will then be taken into a cost function to assist a cluster owner's decision when HPC cloud systems provide a better value than running an independent cluster. The choice between an on-premise HPC system and the cloud will also not just be looked at as an explicit or, as there is also the possibility of owning a smaller cluster and outsourcing some parts to the cloud. The cost function for the comparison against the cloud can then provide a way to outsource specific jobs to the cloud, reducing the total cost of the cluster and potentially even optimising the remaining jobs for the job scheduler.

Contents

1 Introduction

This chapter will feature a brief introduction combined with a small discussion of the motivation and goals of this thesis. Afterwards, an outline of the upcoming chapters of the thesis will follow this.

1.1 Motivation

High Performance Computing (HPC) has always required large data centres with staff managing the cluster on-site. These clusters have fixed hardware for their whole life span, often five or six years. Most of the time, the hardware is chosen explicitly to satisfy the needs of the applications that run primarily on the cluster. These servers also require a high-speed network connection to have the single servers work together on the same tasks and communicate during this process.

On the other hand, cloud services have started primarily with web services and general-purpose computing in mind. In the following years, the cloud services evolved further and began to market themselves as an alternative to HPC data centres. While the cost of a single computing node in the cloud is typically higher than with an on-site cluster, the cloud services balance this issue with particular instance types with lower prices. This aspect is already covered by some papers like one from Purdue University, which is also discussed in chapter 3 [Smi+19].

An advantage of the cloud is the flexibility, as with an on-site cluster, nodes are purchased for a lifespan of about five years typically. In the cloud, on the other hand, nodes are typically rented for just the time they are used so that the user can switch the type of hardware at any time. Additionally, on-site clusters require maintenance and large server rooms to contain them, which leads to the idea that cloud services might be a reasonable solution, despite the potentially higher cost.

The motivation of this thesis also is not specifically to completely substitute on-premise clusters with cloud systems. Instead, the motivation is a potential cost reduction by running some of the jobs in the cloud to reduce the number of nodes needed in the HPC cluster. This cost reduction will be highly dependent on the types of jobs and the behaviour of the users to correctly utilise the provided nodes.

1.2 Thesis Goals

This thesis aims to take an actual on-site cluster at the German Climate Computing Center in Hamburg and use its data to analyse if replacing parts of the cluster, or even the whole cluster, with a cloud solution, would provide benefits. On the cloud side, the largest three cloud providers will be considered to compare against the cluster in Hamburg.

First, creating a cost function for this comparison requires some data on the current usage of the cluster. For this, the clusters job data of a whole year needs to be gathered and analysed for several aspects, like job configuration and utilisation. Based on this analysis, a cost function needs to be established to show how much of the cluster can be outsourced to the cloud while still providing a better value than the on-site cluster. This cost function should also consider all requirements for the cloud system to run these jobs just as well as the on-site cluster.

This analysis only applies to the partial replacement of a cluster, so an additional analysis of the cost of running the whole cluster in the cloud is needed. This analysis has to be split up, with the first part using the regular prices of cloud computing instances and the other part using special prices, which are available with some conditions in the cloud. The raw computing power is not the only factor, as storage, for example, can play a significant role in computing, with the results of the calculations needing to be stored too.

Another aspect will be the configuration of the compute nodes, as the cloud providers have a vast number of different node configurations, with CPU power, not the only aspect to consider. For many applications, the memory can play a significant part, as the application data necessary for the calculations must be stored in the system memory. Some applications can require large amounts of memory, and this also has to be considered in the choice of cloud nodes.

For larger jobs that use multiple nodes simultaneously, the network bandwidth can also be a bottleneck. With the cloud providers, many of their default instances only provide bandwidths of 10 GBit/s or less, which is much less than needed for such large jobs. The bandwidth requirements also vary between different types and sizes of jobs, so this should also be considered when choosing the fitting cloud instances.

1.3 Thesis Outline

The following chapters will start with some deeper insight into the HPC system and the cloud systems used for the comparison in chapter 2. As a further introduction, a quick look into some related work comes after, followed by the first cost comparison for a complete cluster replacement in chapter 4. This cost comparison will not only feature a look into the costs of compute instances but also these of storage.

Chapter 5 will then start with the analysis for the partial replacement of the cluster by analysing the job data gathered from Mistral. Using this analysis, chapter 6 will first come to a deeper look into specific cases that have to be looked at and will weigh up if the specific case would be a good option for the cloud. The following chapter 7 will then use these findings to create a cost function that should help in the decision on how much of the cluster can be outsourced reasonably to the cloud. Chapter 8 then concludes this thesis and goes into some future work.

2 Background

This chapter will provide some background on the HPC system, on which the comparison is based, and the cloud providers used for the comparison. The info on the cloud providers will also contain some general information on cloud services and their instances.

2.1 HPC System

The HPC System used to evaluate the use of Cloud Solutions for High Performance Computing (HPC) is 'Mistral' by the German Climate Computing Center in Hamburg. It features about 3300 nodes for Central Processing Unit (CPU) calculations, with 1550 nodes using Intel Haswell CPUs and 1750 nodes using slightly newer Intel Broadwell CPUs [DKRb]. Additionally, the storage system consists of about 54 PB of HDD storage and a vast tape archive with a capacity of more than 200 PB of storage and the ability to expand this storage further with more tapes. The storage system also contains a small amount of SSDs, which are solely used for the file system's cache.

Infiniband FDR with 54.54 GBit/s bandwidth connects all the nodes and helps achieve a theoretical peak performance of 3.59 PetaFLOPS using all 3300 nodes. Using Lustre as the file system, the HDD system provides a peak speed of more than 450 GB/s [DKRb], which is required for the vast amounts of climate data.

As the name suggests, the German Climate Computing Center is primarily for climate scientists for running their simulations. Due to the lack of GPU optimisations for most climate applications, the cluster mainly consists of CPU nodes, with only a few GPU nodes available for visualisation.

On Mistral, there are several partitions for the nodes with different purposes. The two primary partitions are 'compute' and 'compute2', with 'compute' running the Haswell CPUs and 'compute2' running the Broadwell CPUs. The third relevant partition is 'shared', with 36 Haswell nodes used to run small jobs, which only need fractions of a node instead of a full node. Additionally, Mistral also features twenty GPU nodes for visualisation purposes.

2.2 Cloud Solutions

For the cloud solutions used for this thesis, we are looking at the three largest cloud providers. These cloud providers consist of AWS by Amazon, Azure by Microsoft and Google Cloud.

2.2.1 AWS

Amazon Web Services (AWS) is the Cloud Service from Amazon and is the largest player in the cloud market with more than 30 per cent market share [Sta]. The most prominent parts of AWS and the most relevant parts for this thesis are EC2 and S3, which provide computing and storage capacities.

AWS Elastic Compute Cloud (EC2) is a service to provide cloud servers for computing purposes [Sera]. It provides a large variety of servers using x86 CPUs, ARM CPUs, and in some instances also additional GPUs. These servers can be allocated at any time and have prices per hour, but there are also options to allocate servers for one or three years, significantly reducing the costs. The ARM CPUs are 64-bit CPUs called "Graviton2" and are developed by Amazon themselves [Serc]. While "Graviton2" is a fairly new architecture, its predecessor "Graviton" only featured 16 vCPUs, but in performance analysis, the first generation already showed promising performance [JLZ20].

There also is a third option for pricing, called spot instances. With spot instances, they can be allocated for a reduced price whenever there are free servers available. The downside of these instances is that AWS can interrupt them at any time when AWS needs these instances for on-demand customers.

AWS Simple Storage Service (S3) provides cloud storage solutions and can be connected directly to EC2 instances [Serb]. There are four different storage tiers, with the most expensive one consisting of SSD storage for data that has to be accessed frequently. The cheapest tier is the deep archive tier, which is supposed for data that has to be stored for long times and is typically not accessed or deleted for more than a year.

Another AWS feature is their Nitro cards with an optimised network protocol compared to TCP. The protocol AWS created for this is called Elastic Fabric Adapter (EFA) and resembles Infiniband, but Amazon claims to have eliminated some limitations of Infiniband's model [Sha+20]. They also provided some benchmarks against TCP, showing a greatly improved latency.

2.2.2 Azure

Azure is Microsoft's cloud service and is the second-largest player in the cloud market, making up more than 50 per cent of the market together with AWS [Sta]. It provides similar services like AWS, so the relevant ones for this thesis are virtual machines [Azub] and Azure blob storage [Azua].

Azure's virtual machines also provide a large variety of x86 CPUs and additional GPUs for its instances. Compared to AWS EC2, Azure is missing the ARM CPUs here, but Microsoft is rumoured to be working on ARM instances [ZDN]. The pricing options are also similar for two options with regular instances with prices per hour and significantly cheaper options with allocations over one or three years. The third option, spot instances, is missing with Azure.

Azure blob storage also has many similarities with AWS, with four storage tiers to provide different storage levels. The only significant difference is the option to reserve storage for 1 or 3 years, just like with virtual machines, with much lower prices.

2.2.3 Google Cloud

Google Cloud is significantly smaller than AWS and Azure but still the third-largest cloud provider. It also provides very similar options with Compute Engine [Clob] for compute servers and Cloud Storage [Cloa] for storage solutions.

Both of these services are similar to their competitor's options with only a few differences. First, Compute Engine has no ARM CPUs, just like Azure. Additionally, Compute Engine also has no spot instances, and the discounts for using nodes regularly and not temporarily are only monthly. Googles cloud storage also does not provide discounts for reserved disk space, unlike Azure.

2.2.4 Cloud Instances

Another critical factor for all of the three cloud providers is the different types of cloud instances. There are typically three types of cloud instances with all of these providers: on-demand, reserved, and spot instances.

On-demand instances are the classic cloud instances, which have fixed prices and are available at any time. These instances are also typically paid precisely for the time they are used, with prices provided on an hourly basis. Due to the excellent availability and flexible pricing, these are also the most expensive cloud instances.

The reserved instances provide a price reduction if known beforehand that the nodes will be fully utilised for a long time. These reservations are typically for one or three years and provide up to 70% lower prices. These reserved instances are cheap compared to the on-demand instances, but they also remove the flexibility altogether. Azure also provides five-year reservations on some of their instances, but not as a general offer for all instances.

Spot instances have started as a way for the cloud providers to sell their excess capacities for lower prices, but with the option to always cancel the instance when an on-demand user needs the node. Up until a few years ago, AWS had a system where everybody could place their bid of much they would pay for the spot instance, but this system at times even exceeded the on-demand pricing [Geo+19]. In 2018 AWS changed the way spot works then, that AWS predefines the price, so it does not fluctuate that extremely [Sere]. Additionally, with the way the AWS handles spot instances, the only way of getting evicted is when on-demand customers allocate the nodes used for the spot instance. While Google Cloud also uses more fixed pricing, Azure still has bids that need to be defined before starting the job, and if the price exceeds the maximum bid, the job gets evicted from the node.

Such an eviction of the job is quite normal with all cloud services in spot instances, as the jobs typically get about 30 seconds to two minutes warning if they are about to be evicted. AWS even has a new feature since November 2020 to provide even earlier notifications when the risk of eviction is getting larger [Serd]. That warning can come in handy if the running software supports checkpointing, so it can continue later where it stopped. AWS, for example, also provides the option to name multiple different nodes and server regions, which would be okay to start the job on, so there are more availability options. With the scheduler picking the best option with the highest availability using their internal heuristics, the probability that the job gets evicted is relatively low.

3 Related Work

This chapter will discuss some related work in the area of HPC in the cloud.

There already exist several papers on cost comparisons between on-premise HPC clusters and the cloud. This chapter will begin with two papers that also take a look into a cost comparison between an existing HPC cluster and cloud systems. The second part will then go on with other papers that provide a view on different aspects of running HPC applications in the cloud.

3.1 Cost Comparisons Between HPC Clusters And The Cloud

In this part, the papers provide a view on a similar topic as this thesis, but with a few differences in their cluster's setup and their goals. The first one is a relatively old paper, at least in the space of cloud computing, and was published in 2011 by a group of Purdue University [CHS11]. In this paper, the authors discussed the cost of replacing their community cluster with a cloud solution.

They used a relatively simple approach, only taking their cluster costs and using them to calculate costs per hour to compare with AWS. In the end, they concluded that the cloud might be a good solution for single users without access to a cluster but can not cost-efficiently replace their cluster. They also used so-called "Cluster Compute Instances" by AWS, which did not provide an option for spot instances, and it seems to be discontinued. The lack of discounts like reserved or spot instances made this approach quite simplistic as opposed to this thesis and gave the cloud systems quite a disadvantage against their cluster.

Two of this paper's authors made an updated version of the paper in 2019 [Smi+19], with a new look on the same topic. In this paper, they used their new hardware of the Purdue community cluster and also added comparisons to Azure and Google Cloud instead of just AWS like their first paper. This time they also used one-year reserved instances but no three-year reserved or spot instances.

Additionally, they also looked at the different project groups using their cluster, and by this, they found out that some groups would actually get away cheaper with the cloud. However, the findings looking at all groups were still that the cloud is not as cost-effective, but while they talked about a factor 7 of savings with an own cluster, this time they only came to a cost factor of 2.73 for using AWS for their cloud computing needs. Azure and especially Google Cloud came away worse than AWS, but Google Cloud also does not provide one-year reserved instances. In this renewed paper, the authors considered more different factors, but they still focused on replacing their whole cluster, differentiating their approach from this thesis's goals.

3.2 Other Papers On HPC In The Cloud

There are many more papers about HPC in the cloud, but two of these stand out with their relevance to this thesis. The first of these is a paper from 2012 discussing the deployment, performance and also cost of several cloud providers [Rol+12]. As their cloud providers, the authors of the Federal University of Rio Grande do Sul in Brazil chose AWS, Azure and a smaller provider called Rackspace. The paper is also fairly old in the space of cloud computing, so their findings do not have to translate to the current situation, but they still might be interesting.

In the first part, the authors discuss the deployment of cloud nodes, and they found that AWS was providing the quickest deployment, especially for a higher number of nodes. Following this, they analysed the cloud's performances of 8-core instances against their cluster with 8-core nodes. In the performance analysis, the authors found that the cloud providers were scaling better with a higher number of nodes against their cluster, with four nodes of Azure even outperforming their cluster in most benchmarks. Azure's excellent multi-node performance might be caused by the network, as their cluster only used gigabit network, but there is no information on the network Azure used. The papers last part, the cost analysis, found that cost efficiency depends on the specific cloud solution used. In this part, they calculated a break-even point when buying a cluster gets more efficient than running the jobs in the cloud. Two-node jobs on Rackspace reached the break-even point the first, after just 26 days, with four-node jobs on Azure providing the best value with the break-even point at 620 days. While it also is a fascinating approach to the break-even point of a cluster against the cloud, this use case is not as well-suited for big HPC centres with regular hardware upgrades. Additionally, the authors only used the power costs as running costs and did not look at other running costs of a cluster.

The second paper is about two years newer by Rashid Hassani, Md Aiatullah and Peter Luksch of the University of Rostock [HAL14]. In this paper, the authors analyse the performance of a sorting algorithm on their cluster against AWS. The sorting algorithm used is an MPI variant of Radix sort with different sizes of workload data. The authors conclude that AWS provides good HPC performance as in the benchmarks AWS outperformed their cluster and scaled quite well. However, they also remark that the cloud is only suitable for specific applications, as additional optimisations might be necessary.

One aspect that makes their comparisons to their cluster a bit different is the vastly different hardware with their AMD Opteron CPUs against the Intel CPUs AWS used at the time. Additionally, the authors did not mention the network they were using, which might be relevant for the multi-node runs they did, where AWS faired even better. The paper is also not that comparable to this thesis, as it does not go into the cost of the cloud, only into the performance, but this is still an exciting aspect of seeing the cloud perform well against on-site clusters.

4 Cost Comparison

In this chapter, the cost difference between an HPC cluster and cloud solutions will be discussed.

The first simple way to compare the cost of an HPC system to a cloud solution would be to try to match the whole HPC system to the price of a similar cloud system. We will use the standard price per hour in the first comparison, with other variants coming later. For Mistral, that would require 1550 nodes of 12-core nodes from the Haswell generation or better and 1750 nodes of 18-core nodes from the Broadwell generation or better [DKRb]. The storage would also require 54 Petabytes of HDD storage, combined with 200 Petabytes of archive storage.

4.1 Computing

Some nodes only have vCPUs, instead of real cores and would need 48 or 72 vCPUs, respectively. Of course, the cores of the instances running more modern CPU architectures are significantly faster, so fewer vCPUs like the 60 with Google Cloud as the replacement for 72 threads would be reasonable too. Also, for AWS, the "Europe (Frankfurt)" servers are used for calculating, for Azure "West Europe" is used, and for Google Cloud "Europe-west1". The following table first shows the best fitting nodes from each cloud service to replace the Haswell nodes. 50 GBit/s network will be the minimum requirement for actual multi-node operations.

AWS c5n.9xlarge	Azure HB60rs	Google c2-standard-30
36 vCPUs	60 Cores	30 vCPUs
Intel Cascade Lake	AMD EPYC Naples	Intel Cascade Lake
96 GiB RAM	228 GiB RAM	120 GiB RAM
50 GBit/s Network	100 GBit/s Network	50 GBit/s Network
1.86€/hour	2.50€/hour	1.44€/hour

Table 4.1: Cloud nodes for replacement of the Haswell nodes

Azure has quite a disadvantage with these nodes, as they do not provide as many HPC instances with fewer cores. One step down would be their 16-core Haswell nodes, which is less than the 24-cores it would need to replace, so instead, Azure provides 60 real cores with AMD EPYC CPUs. For the replacement of the Broadwell nodes, the following nodes were selected:

AWS c5n.18xlarge	Azure HB60rs	Google c2-standard-60
72 vCPUs	60 vCPUs	60 vCPUs
Intel Cascade Lake	AMD EPYC Gen1	Intel Cascade Lake
192 GiB RAM	228 GiB RAM	240 GiB RAM
100 GBit/s Network	100 GBit/s Network	100 GBit/s Network
3.82€/hour	2.50€/hour	2.89€/hour

Table 4.2: Cloud nodes for replacement of the Broadwell nodes

While especially Google Cloud profits from the relaxed node criteria, Azure, on the other hand, has significantly better nodes than Mistral. While this inequality makes this not an optimal comparison, these are the cheapest HPC nodes at Azure that fit the minimum criteria, as Azure only provides access to full nodes for the newer HPC instances. Setting up some calculations for the full costs over a runtime of 6 years, the formulas would look as following:

1. AWS: $NNodes \cdot Price_{Node} \cdot hours \cdot days \cdot years$
   $= (1550 \cdot 1.86€ + 1750 \cdot 3.82€) \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 503,200,000€$

2. Azure: $(NNodes_{Haswell} \cdot Price_{Haswell} + NNodes_{Broadwell} \cdot Price_{Broadwell}) \cdot hours \cdot days \cdot years$
   $= 3300 \cdot 2.50€ \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 433,900,000€$

3. Google Cloud: $(NNodes_{Haswell} \cdot Price_{Haswell} + NNodes_{Broadwell} \cdot Price_{Broadwell}) \cdot hours \cdot days \cdot years$
   $= (1550 \cdot 1.44€ + 1750 \cdot 2.89€) \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 383,400,000€$

Mistral's total cost is around 35 million euros for the cluster, about 1-1.5 million euros for power per year, about 400,000 euros of rent for the data centre and personnel costs for maintaining the cluster of about 250,000 euros. How these specific values were gathered will also be discussed later in chapter 7. Looking at these costs for Mistral and the cloud, the cloud clearly does not provide a good value. However, on the other hand, this is only the standard pricing and better nodes than Mistral uses.

The next step would be to go with pre-reserved cloud nodes for AWS or Azure to reduce the pricing further, as the value gets significantly better with these, especially in AWS. The pre-reserved cloud nodes are allocated for one or three full years and provide a much better value. Google Cloud also provides reserved instances for some instances, but unfortunately not for the predefined compute instances used in this comparison.

Another factor, which Google Cloud covers as opposed to the reserved instances, is the spot instances with low prices but also low availability. As the cloud provider can also cancel spot instances at any time, they are very risky, but they will still be considered for comparison. Table 4.3 shows the node pricing for reservations and spot instances.

Nodes	AWS c5n.9xlarge	AWS c5n.18xlarge	Azure HB60rs	Google c2-30	Google c2-60
1-Year Reserved	1.20€/hour	2.40€/hour	2.13€/hour	Not Available	Not Available
3-Year Reserved	0.82€/hour	1.64€/hour	1.75€/hour	Not Available	Not Available
Spot	0.58€/hour	1.03€/hour	0.53€/hour	0.36€/hour	0.71€/hour

Table 4.3: Node Cost with reserved and spot instances

As the table shows, these instances provide a vast cost reduction, cutting the cloud systems' costs by up to 60 per cent for the reserved instances and up to 80 per cent for the spot instances.

The calculations for the full costs of the pre-reserved and spot instances can be seen in the following:

1. AWS 1-year: $NNodes \cdot Price_{Node} \cdot hours \cdot days \cdot years$
   $= (1550 \cdot 1.20€ + 1750 \cdot 2.40€) \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 318,700,000€$

2. AWS 3-year: $NNodes \cdot Price_{Node} \cdot hours \cdot days \cdot years$
   $= (1550 \cdot 0.82€ + 1750 \cdot 1.64€) \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 217,800,000€$

3. AWS Spot: $NNodes \cdot Price_{Node} \cdot hours \cdot days \cdot years$
   $= (1550 \cdot 0.58€ + 1750 \cdot 1.03€) \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 142,100,000€$

4. Azure 1-year: $(NNodes_{Haswell} \cdot Price_{Haswell} + NNodes_{Broadwell} \cdot Price_{Broadwell}) \cdot hours \cdot days \cdot years$
   $= 3300 \cdot 2.13€ \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 369,700,000€$

5. Azure 3-year: $(NNodes_{Haswell} \cdot Price_{Haswell} + NNodes_{Broadwell} \cdot Price_{Broadwell}) \cdot hours \cdot days \cdot years$
   $= 3300 \cdot 1.75€ \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 303,700,000€$

6. Azure Spot: $(NNodes_{Haswell} \cdot Price_{Haswell} + NNodes_{Broadwell} \cdot Price_{Broadwell}) \cdot hours \cdot days \cdot years$
   $= 3300 \cdot 0.53€ \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 92,000,000€$

7. Google Spot: $(NNodes_{Haswell} \cdot Price_{Haswell} + NNodes_{Broadwell} \cdot Price_{Broadwell}) \cdot hours \cdot days \cdot years$
   $= (1550 \cdot 0.36€ + 1750 \cdot 0.71€) \cdot 24 \cdot 365.25 \cdot 6$
   $\approx 94,700,000€$

While especially the spot instances show a significant cost reduction, permanently running 3300 of these instances is not realistic. Also, if the user is running an MPI job on multiple nodes, and one node gets evicted by the cloud provider, the whole job will die, so running spot instances is ruled out. Realistically this instance type is made for quicker jobs that do not make issues if they get cancelled or even better if they use checkpointing. A complete comparison of the costs of the different instance types can be found in table 4.4.

	AWS	Azure	Google Cloud
Nodes	1550 c5n.9xlarge + 1750 c5n.18xlarge	3300 HB60rs	1550 c2-30 + 1750 c2-60
On-Demand	503,200,000€	433,900,000€	383,400,000€
1-Year Reserved	318,700,000€	369,700,000€	Not Available
3-Year Reserved	217,800,000€	303,700,000€	Not Available
Spot	142,100,000€	92,000,000€	94,700,000€

Table 4.4: Compute Cost with different pricing models

As spot instances are no realistic scenario for a total replacement, so three-year reserved instances seem like the best solution for this case. However, while these instances would be able to replace the cluster, the cost would still be at least four times as high as the on-site cluster.

Nevertheless, one more aspect is not included, as it is possible to use better nodes after a few years since cloud services have a quicker upgrade cycle than an HPC cluster. Using the 3-year reserved nodes, i.e., the contract for these nodes would make up exactly half of the typical lifetime of a cluster at the DKRZ. For the second 3 years, newer nodes could be used, as the hardware market is evolving quickly in such a period. Additionally, not only newer nodes could be used, but if the need for a different type of architecture came in these three years, additional nodes could always be allocated. For example, this addition of nodes could be the case if new applications run well on ARM or GPU nodes.

The last aspect to potentially save more costs is to choose a different data centre of the cloud. While only European data centres were taken for the first comparison, there is a vast selection of data centres with the three cloud providers. As a quick look, the cheapest AWS data centre will be taken in table 4.5 to provide a quick view of how cheap the nodes could get.

	Node Prices (Frankfurt)	Node Prices (Ohio)	Total Cost (Ohio)	Total Cost (Frankfurt)
On-Demand	1.86€/3.82€	1.66€/3.33€	441,800,000€	503,200,000€
1-Year Reserved	1.20€/2.40€	0.98€/1.96€	260,300,000€	318,700,000€
3-Year Reserved	0.82€/1.64€	0.53€/1.06€	140,800,000€	217,800,000€
Spot	0.58€/1.03€	0.39€/0.59€	86,100,000€	142,100,000€

Table 4.5: AWS Cost Comparison Frankfurt vs. Ohio

This comparison shows a significant difference between the EU and US data centres, with about a 12-40% cost reduction against the nodes in Frankfurt.

4.1.1 Data Security

With the vast discount on the nodes in Ohio, the question arises why the cheapest data centres were not picked directly in the first comparison. While there are cheaper data centres, the choice of the European data centres was made on a data security basis, as data in the US is much easier accessible to US authorities.

For data stored in the EU, the US authorities can still request stored data, but the legislations provide much more restrictions for data in the EU. On the one hand, there is the Clarifying Lawful Overseas Use of Data Act (CLOUD Act) which aims to provide US authorities with ways to request data from US cloud providers if the data is lying outside of the United States [Wik]. While the authorities have much easier access inside the United States, they have to obey the local law and provide their reasoning for illegal activities by the user the data is requested from using this act. If they do not provide sufficient information, the cloud provider can also deny the request for data access.

On the other hand, speaking of the local law, the EU has the General Data Protection Regulation (GDPR), which regulates the transfer of personal data, which is stored in the EU, to the US. The GDPR and CLOUD Act working together is also a widely discussed topic with several aspects between them for when data can be successfully requested in the EU [PC]. To safely apply the GDPR on a CLOUD Act request, it would be required that the interests of the data request outweigh the personal interests of the owner of the data. Another possibility to combine these two acts would be ”in an emergency situation where there is a threat to life or physical harm” [PC].

So while, with data stored at cloud providers in the EU, it would still be possible the data can get accessed, there would have to be a very good reason to access the data, which makes this unwanted data access very unlikely. With US servers, on the other hand, the US authorities have it much easier to access the data, which is also why the European servers were chosen for the earlier comparison.

4.2 Storage

For storage, we need large amounts to replicate the amount of storage available with Mistral. With AWS, we will use the Simple Storage Service (S3) for infrequent access, to use HDDs, for 0.0135$ per GB and month in the Frankfurt data centre. AWS Glacier Deep Archive will be used to replicate the tape archive, as this data is most likely accessed very rarely. Deep Archive costs 0.0018$ per GB and month in the Frankfurt data centre, so it is significantly cheaper than the HDD storage.

With Azure, Storage Blobs would be the pendant to S3, and in the West Europe data centre, the cold storage costs 0.00844€ per GB and month, while archive costs 0.00152€ per month. So after a currency conversion of about 0.85€ per dollar, we see that the archive storage is pretty much the same price, while the HDD storage is cheaper with Azure.

With Google Cloud, the storage solution is just called Cloud Storage, and the storage tiers we use to compare are nearline storage for 0.01$ per GB and month and archive storage for 0.0012$ per GB and month, both of them in the "Europe-west1" data centre. While the HDD storage is about the same price as with Azure, in this case, the archive storage is even cheaper. In total, these values result in the following costs for Mistral over six years.

AWS	Azure	Google Cloud
54 PB Infrequent Access	54 PB Cold Storage	54 PB Nearline Storage
46,900,000€	34,400,000€	34,700,000€
200 PB Glacier Deep Archive	200 PB Archive Storage	200 PB Archive Storage
22,600,000€	23,000,000€	18,100,000€
69,500,000€	57,400,000€	52,800,000€

Table 4.6: Storage Cost for Cloud Systems

The complete calculations of the storage prices for six years can be seen in the following:

1. AWS: $(Storage_{HDD} \cdot Price_{HDD} + Storage_{Tape} \cdot Price_{Archive}) \cdot months \cdot years$
   $= ((54PB \cdot 1024 \cdot 1024 \cdot 0.0115€) + (200PB \cdot 1024 \cdot 1024 \cdot 0.0015€)) \cdot 12 \cdot 6$
   $\approx 69,500,000€$

2. Azure: $(Storage_{HDD} \cdot Price_{HDD} + Storage_{Tape} \cdot Price_{Archive}) \cdot months \cdot years$
   $= ((54PB \cdot 1024 \cdot 1024 \cdot 0.00844€) + (200PB \cdot 1024 \cdot 1024 \cdot 0.00152€)) \cdot 12 \cdot 6$
   $\approx 57,400,000€$

3. Google Cloud: $(Storage_{HDD} \cdot Price_{HDD} + Storage_{Tape} \cdot Price_{Archive}) \cdot months \cdot years$
   $= ((54PB \cdot 1024 \cdot 1024 \cdot 0.0085€) + (200PB \cdot 1024 \cdot 1024 \cdot 0.0012€)) \cdot 12 \cdot 6$
   $\approx 52,800,000€$

As these prices are still significantly higher than the cost of the whole cluster, in the case of Mistral, it does not seem to be a great value to outsource the storage. However, there is one more option to reduce the cloud storage prices, as Azure is the only provider that also offers storage reservations for one or three years, which significantly lowers the total cost. As the following calculations show, these reserved storage prices do bring down the price significantly, but not below the cost of the whole Mistral cluster.

1. Azure 1-year: $(Storage_{HDD} \cdot Price_{HDD} + Storage_{Tape} \cdot Price_{Archive}) \cdot months \cdot years$
   $= ((54PB \cdot 1024 \cdot 1024 \cdot 0.00690€) + (200PB \cdot 1024 \cdot 1024 \cdot 0.00135€)) \cdot 12 \cdot 6$
   $\approx 48,500,000€$

2. Azure 3-year: $(Storage_{HDD} \cdot Price_{HDD} + Storage_{Tape} \cdot Price_{Archive}) \cdot months \cdot years$
   $= ((54PB \cdot 1024 \cdot 1024 \cdot 0.00548€) + (200PB \cdot 1024 \cdot 1024 \cdot 0.00124€)) \cdot 12 \cdot 6$
   $\approx 41,100,000€$

While these storage costs are already not competitive with the on-site cluster, accessing the data is not free either in the cloud. The cost of data transfers can be pretty high, but it is hard to put a number on that, unlike the storage amount, since data transfers vary from case to case.

5 Mistral's Job Data

This chapter will focus on analysing Mistral's job data on specific behaviour usable in the cloud.

As the complete replacement of an HPC cluster is not the best value in most cases, a deeper look into specific situations is needed. The whole job data of 2020 was extracted from SLURM to provide a representative amount of jobs for the analysis. This job data consists of about 7.5 million jobs in total with multiple values provided, like the time they ran or the point of time when the user committed them or when they started. These jobs are anywhere in the range of 1 to 1550 nodes and have vastly different runtimes.

However, there are some restrictions in Mistral's SLURM for the way users may run jobs: The node count must not exceed 500 nodes, and the run time must not exceed 6 hours. There are exceptions to this; if needed, users can always start larger or longer jobs in coordination with the systems department of the DKRZ, but without their help, these are the hard limits.

As the jobs from the 'shared' partition are also vastly different from those on 'compute' and 'compute2', the 'shared' partition was excluded from the analysis. The jobs from this partition would have needed very different treatment, with many jobs running on the same node, and the cost of a job would not be as easy to calculate, as the job does not reserve a full node.

As the data from SLURM already was in the CSV format, Pandas was an obvious choice for the analysis, using its data frames [pan]. The other Python libraries primarily used for the scripts were Numpy, Scipy, Pandas, and Matplotlib.

Later in this thesis, this data will then be used to generate the required data for a cost function for the specific cases. Based on Mistral's data, a suggestion for the use of the cloud for Mistral can then be made, with this data being able to be substituted by the data of any other cluster to create a suggestion for this cluster. However, Mistral's data first has to be analysed to gain the points of potential cost savings.

5.1 Node Utilisation

The first simple analysis of the job data consisted of looking at the node utilisation for every day of 2020. This analysis was then extended with a look at the node distribution per weekday and month. The analysis per day is shown in figure 5.1 with one bar representing one day of 2020.

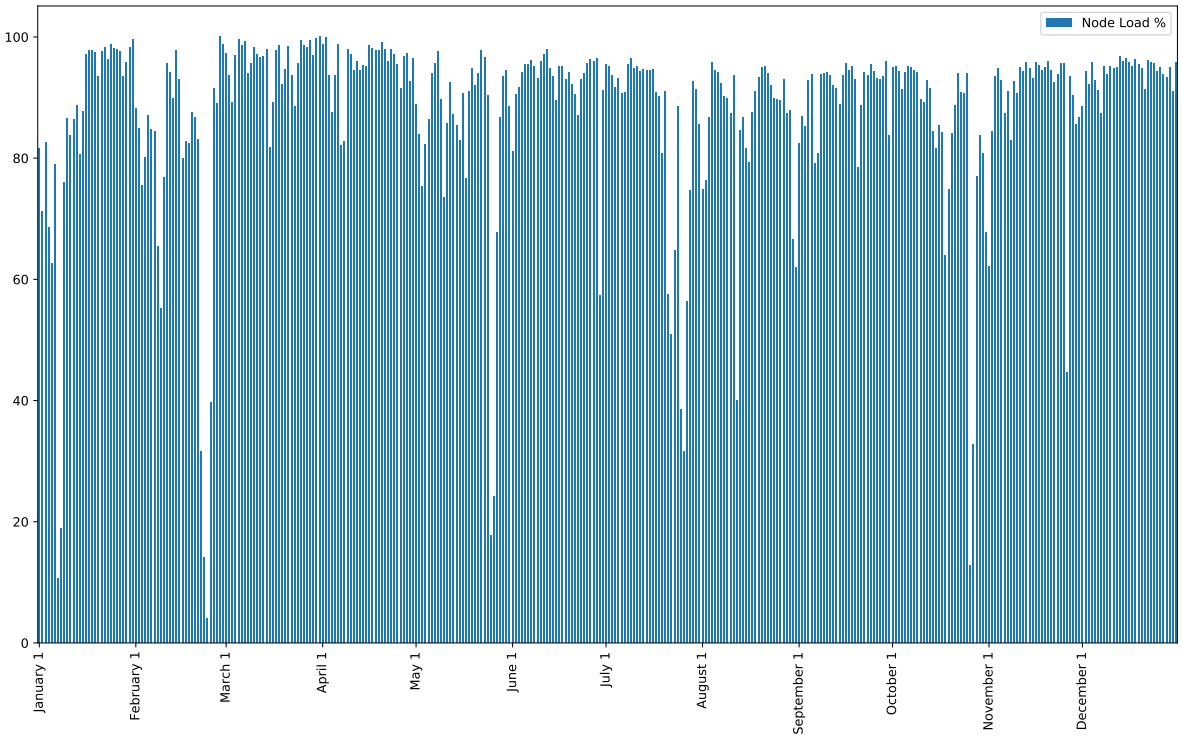

Figure 5.1: Load Percentage per day in 2020

Firstly, this graphic shows a pretty similar load on the Mistral cluster for most of the days, with a few exceptions clearly showing downtimes of the cluster. Instead of an average, a median was used to exclude these exceptions with downtimes of the cluster. Looking at all days, the median load on the nodes is 88.02 %, so the cluster has a pretty high load, with very few nodes idling. The 95 per cent confidence interval for the load is 86.47 to 89.57 %, with the 99 per cent confidence interval being 85.98 to 90.06 %.

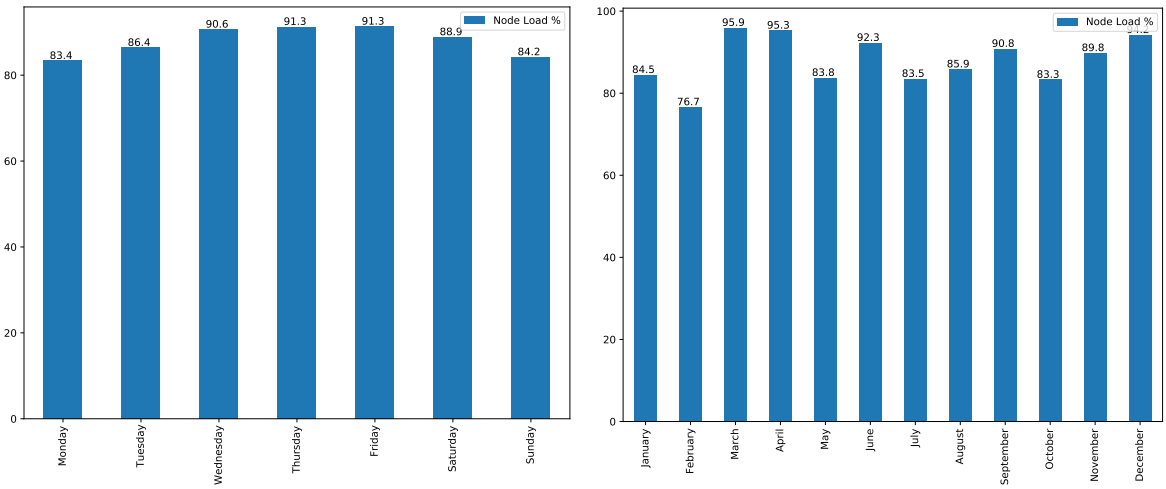

Figure 5.2: Load Percentage per weekday and month in 2020

Figure 5.2 also shows the utilisation per weekday and month to have a deeper look into the factors, which are steering the load of the cluster. As the first graphic, using the weekdays, shows, there is a correlation between weekdays and cluster utilisation. On average, the cluster only exceeds a utilisation of 90 % from Wednesday to Friday, with Sunday and Monday the least utilised days. This change in utilisation might be caused by users mainly starting their jobs at the end of the week to have them run during the weekend to analyse the data when they are back in the office on Monday.

The second graphic does not seem to show a significant seasonality in the utilisation per month. Although there are months with significantly less utilisation, they also correlate with the downtimes witnessed in figure 5.1. Specifically, the months with very low utilisation, February, May, July, and October, all show dips below 50 % utilisation for at least a few days each.

5.2 Node Usage Per Job

Another aspect, which might be very interesting, is the number of nodes a single job uses. For further analysis, all the jobs first got grouped by the number of nodes, and then the number of jobs with this node count was taken, resulting in the following table.

Nodes	1	2	3-8	9-16	17-32	33-99	100-499	500+
Jobs	3,784,716	2,622,661	378,963	363,198	201,669	84,474	26,297	380
Percent	50.72%	35.15%	5.08%	4.87%	2.70%	1.13%	0.35%	0.01%

Table 5.1: Number of jobs for each node count

The result shows that many jobs run only on one or two nodes, while only 0.36% of all jobs use more than 100 nodes. Of course, this does not show how much time the nodes spend for these jobs, as a 100 node job of 1 hour uses the cluster much more than a one-hour single-node job. So we also need to take a look at the distribution of the total node hours, as table 5.2 shows.

Nodes	1	2	3-8	9-16	17-32	33-99	100-499	500+
Node Hours	716,480	165,220	1,491,662	2,768,859	3,753,182	6,282,016	9,320,025	380,350
Percent	2.88%	0.66%	6.00%	11.13%	15.09%	25.25%	37.46%	1.53%

Table 5.2: Node Hours per number of nodes

This table shows a different view, with the single-node jobs only making up less than 3% of the total node hours. The difference is even more significant for dual-node jobs, which indicates that dual-node jobs often have a short runtime. The larger jobs with 33 nodes or more, on the other hand, make up nearly two-thirds of the total node hours, so they have a significant impact on the utilisation of the cluster.

A deeper look into the short jobs is required for the vast difference between dual-node jobs and node time. With user-submitted jobs, we always have jobs that fail immediately on being started because of errors in the job script or jobs that are just very short. Looking at the runtime of the dual-node jobs, nearly 90 per cent, or 2,341,429 of the 2,622,661 dual-node jobs, fall in the sub-60-seconds category. The amount of short jobs is not as significant for other job sizes, but single-node jobs also have a significant number of sub-60-seconds jobs with 1,519,953 of 3,784,716, so about 40 per cent.

5.3 Single-Node Jobs

Although the single-node jobs do not make up many of Mistral's node hours, outsourcing the vast amount of jobs might still lower the load on the scheduler and the cluster's staff. Additionally, while Mistral's scheduler is set up to require multi-node jobs to reserve all CPU cores, the single-node jobs may also use less than a full node. Usually, the "shared" partition is used for such jobs, which do not require a full node, but often enough, the exclusive "compute" partition is used for such jobs.

The following table also supports this claim, showing the distribution of single-node jobs by the number of threads they reserve. The numbers in the table do not exactly sum up to the number of single-node jobs due to the number of invalid jobs that tried to use more than the maximum 72 threads.

NCPUs	1-4	5-8	9-16	17-24	25-48	49-72
Jobs	1,116,301	377,669	792,821	100,151	846,577	545,130
Percent	29.49%	9.98%	20.95%	2.65%	22.37%	14.40%

Table 5.3: Number of jobs for each thread count

With the knowledge that Mistral uses 24-core and 36-core machines, it is clear that all jobs with 16 cores or less do not utilise the full nodes. Of course, it is not as simple as just looking at the used cores because there surely will be some jobs with high RAM usage under these, but many of them do not require an exclusive node.

NCPUs	1-4	5-8	9-16	17-24	25-48	49-72
Node hours	225,746	31,671	42,860	8,976	276,105	131,121
Percent	31.51%	4.42%	5.98%	1.25%	38.54%	18.30%

Table 5.4: Node Hours for each thread count

As before with all jobs, the number of jobs is not solely relevant, but also the number of node hours used. The node hours are also the base for the cost analysis as this can be used to compare the cloud systems.

While the number of jobs is partly representative of the node hours, the node hours significantly differ from the number of jobs in some parts. For example, with the jobs from 9 to 16 nodes, we have about 20% of all jobs, but only a bit more than 5% of the node hours, so these jobs seem to be pretty short typically.

Another analysis of the single-node jobs showed that about 478,000 of the 716,000 single-node hours were spent for long jobs with more than one hour of runtime. With that being about two-thirds of the single-node hours, these jobs got additional focus. This additional focus was also due to the systems department of the cluster saying that shorter single-node jobs can benefit the scheduler, but that will be discussed further in the following chapter.

While these longer single-node jobs make up about two-thirds of the node hours, they are only 106,630 jobs, so less than three per cent of the whole single-node jobs. As the following table shows, a third of these jobs are still running on a maximum of 4 cores and not utilising the nodes fully.

NCPUs	1-4	5-8	9-16	17-24	25-48	49-72
Jobs	33,382	3,773	9,622	1,617	36,564	21,672
Percent	31.31%	3.54%	9.02%	1.52%	34.29%	20.32%
Node Hours	129,538	10,434	20,266	7,338	212,588	97,508
Percent	27.12%	2.18%	4.24%	1.54%	44.51%	20.41%

Table 5.5: Job statistics for long single-node jobs

These numbers show that the primary focus should be on putting the jobs in the cloud, which do not utilise all of the node's cores, and therefore not to the full potential of the node, but this discussion will continue in the next chapter.

5.4 Job Wait Times

Wait times are another essential aspect, which means how long each job has to wait until it starts. Depending on the user's workflow, these wait times can delay further work, as further tasks might require this job's result.

5.4.1 All Jobs

Looking at the average wait time of all jobs, compared to their runtime, in figure 5.3, it shows that there is a definitive correlation between the wait time and the job's runtime. This metric also supports the before-mentioned claim that jobs with a low runtime benefit the scheduler, but this will be discussed later in more detail. The provided wait times are in minutes, with the wait times of 3-to-4-hour jobs standing out by having a wait time of about 4 hours on average, while most of the other lengths are more in the range of 2 to 2.5 hours.

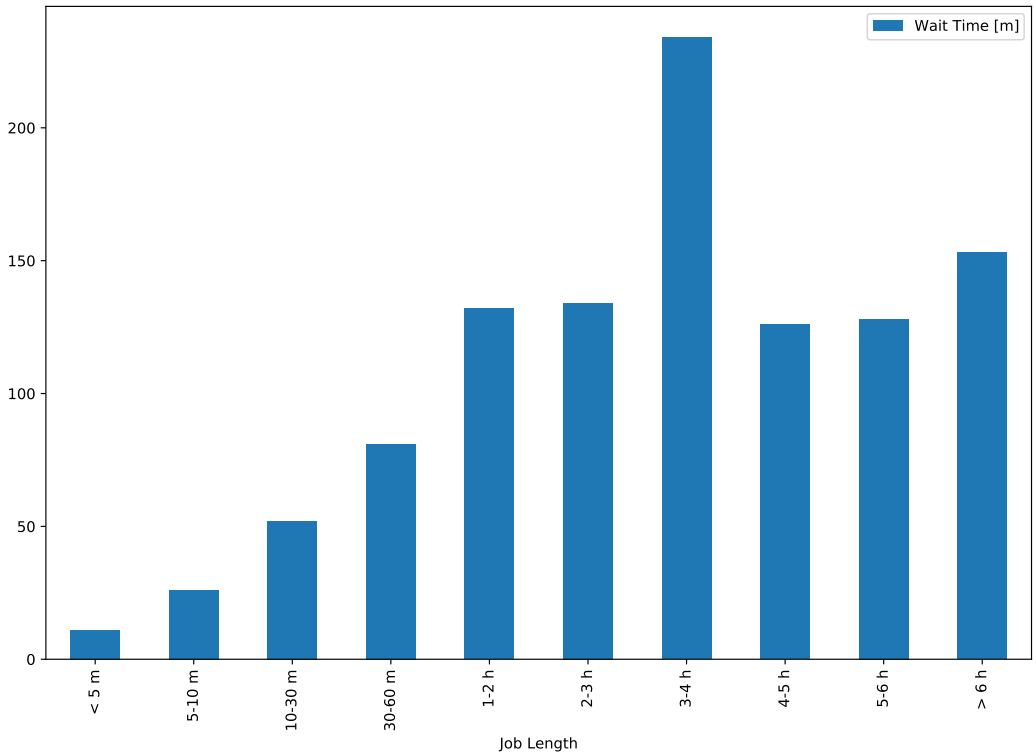

Figure 5.3: Average Wait Times For All Jobs

As this graph shows, up until one hour, the wait times are scaling with the runtime, with them building a plateau at about 2 hours then, with the only exception being the before-mentioned exception of the 3-to-4-hour jobs. This scaling also assisted in the decision to analyse the single-node jobs with runtimes of more than one hour because it seems that this is the point where the scheduler gets the most problems with scheduling the jobs.

The question of what has caused the high wait times for 3-to-4-hour jobs can be solved using the following scatter plot of all wait times. While most wait times are below 2000 minutes, so below 1.5 days, with a runtime of about 200 minutes, there are exceptionally many jobs with high wait times of up to 18000 minutes or 12.5 days.

After further analysis into these specific jobs, they use most of the time about 16 to 20 nodes and are submitted in larger batches of jobs. These job batches are then often scheduled serially by the scheduler, with each following job waiting even longer to be started. Most of these jobs are from the same user account, so they seem to belong together and might even need to be run consecutively.

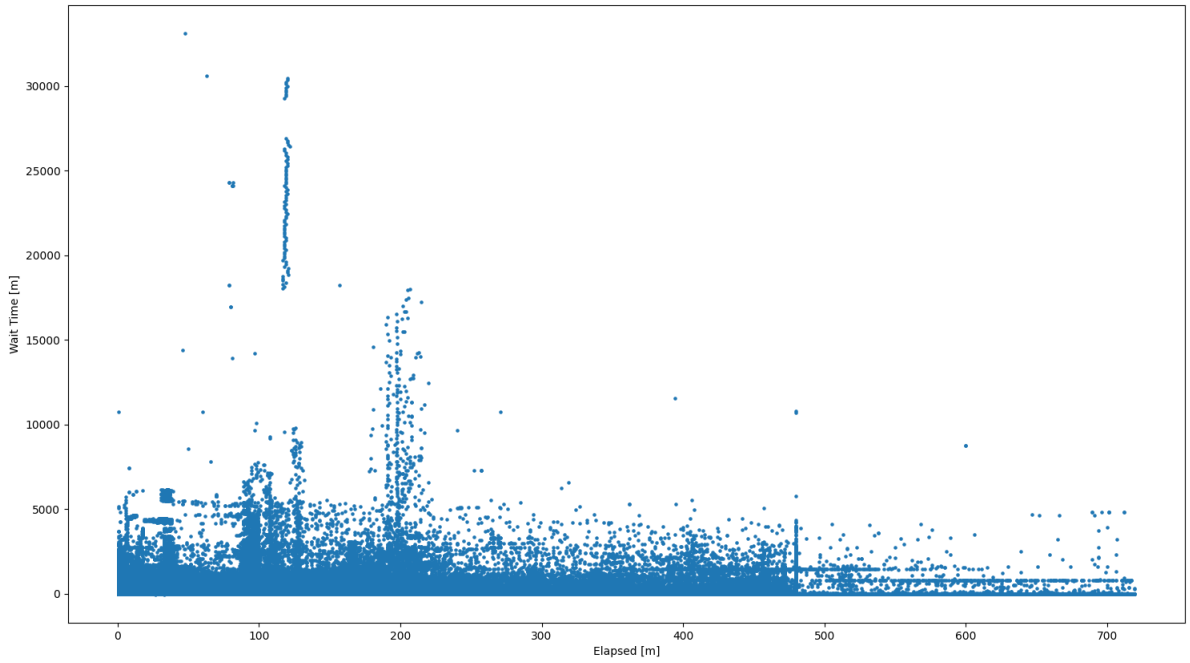

Figure 5.4: Wait Times Scatter Plot For All Jobs

The scatter plot also shows a concentration of jobs at every full hour mark up until the six hours or 480 minutes. This concentration of points is due to the users often using full hours as a time limit for their jobs, which also causes many jobs with long wait times at a similar runtime.

Figure 5.5 shows another approach of splitting the jobs into 30-minute timeframes and creating a box plot for that. This approach takes away the extreme outliers of the runtimes, with the boxes representing everything between the upper and lower quartile. As this figure shows, many jobs are started with no wait time, independently from their runtime.

While the boxes do not paint as clear of a picture, like the bars in figure 5.3, there still are some interesting findings. One interesting point of this figure is that it again shows the walls of high wait times at every full hour. On the other hand, Every other 30-minute timeframe, which does not end on a full hour, has relatively low wait times.

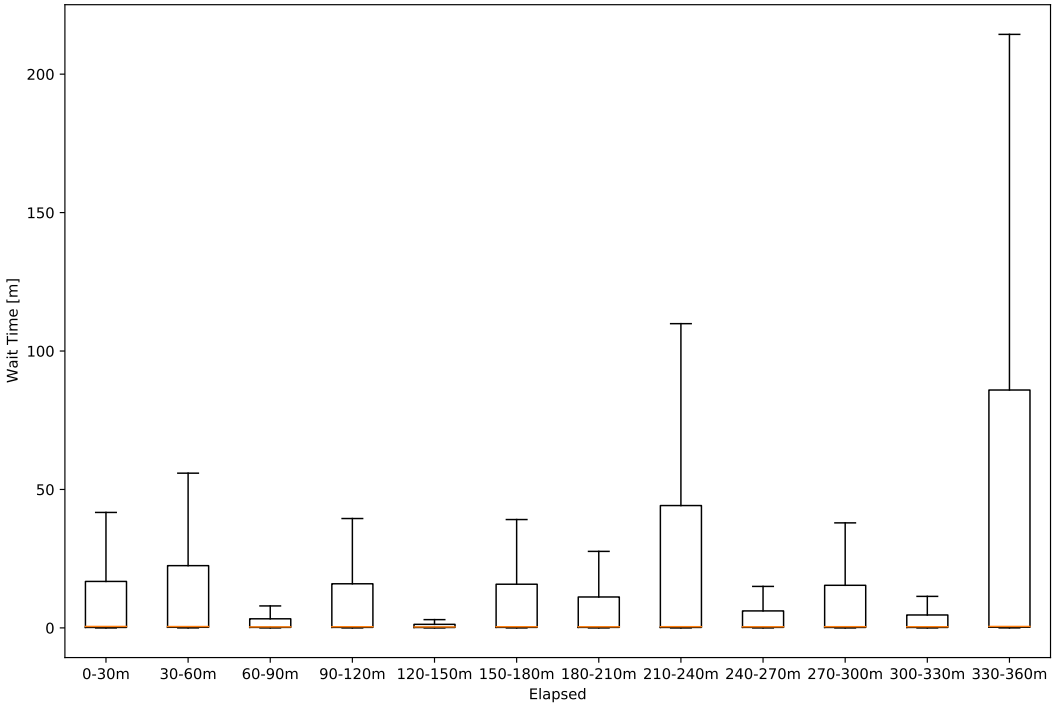

Figure 5.5: Wait Times Box Graph For All Jobs

Another analysis was on the effect of the number of nodes on the wait time, but this was not as telling due to the difference in runtimes. For example, the single-node jobs had an average wait time of nearly 34 minutes, while the dual-node jobs only had about 2.5 minutes. This difference is probably caused by many dual-node jobs being much shorter as their total node hours are much less than the node hours of the single-node jobs.

The 16-node jobs are another excellent example of this, with an average wait time of nearly four hours. Many of the earlier mentioned jobs with high wait times and runtimes of 3-4 hours were also 16 node jobs, which helped to give the 16-node jobs such a high average wait time. This high average is caused by the high wait times of these specific jobs, as wait times of up to 12.5 days raise the average significantly.

NNodes	1	2	4	8	16	32
Jobs	3,784,716	2,622,661	133,911	55,755	40,829	2,948
Average Wait Time	0:34 h	0:03 h	0:14 h	0:11 h	3:53 h	0:14 h

Table 5.6: Wait times for some specific node amounts

To show this effect, table 5.6 shows the average wait times for the job configurations using powers of 2, up to 32 nodes, as these numbers are pretty standard job sizes. As the wait times show, the 16-node jobs are the only of these jobs to have a wait time of more than an hour, with it even being close to four hours.

5.4.2 Single-Node Jobs

The same analysations were also done with only the single-node jobs, as these seem to be an exciting job class for outsourcing to the cloud. While the bar graph in figure 5.6 is not that dissimilar to the one using all jobs, there are some interesting differences. First, it seems that the plateau of the 3+ hour wait times already starts after 30 minutes, not only after 60 minutes. Secondly, the spike for the 3-to-4-hour jobs is gone in this graph, as expected after the earlier analysis. Finally, the graph also shows some fluctuations for the longer runtimes, but with values between 2 and 3.5 hours.

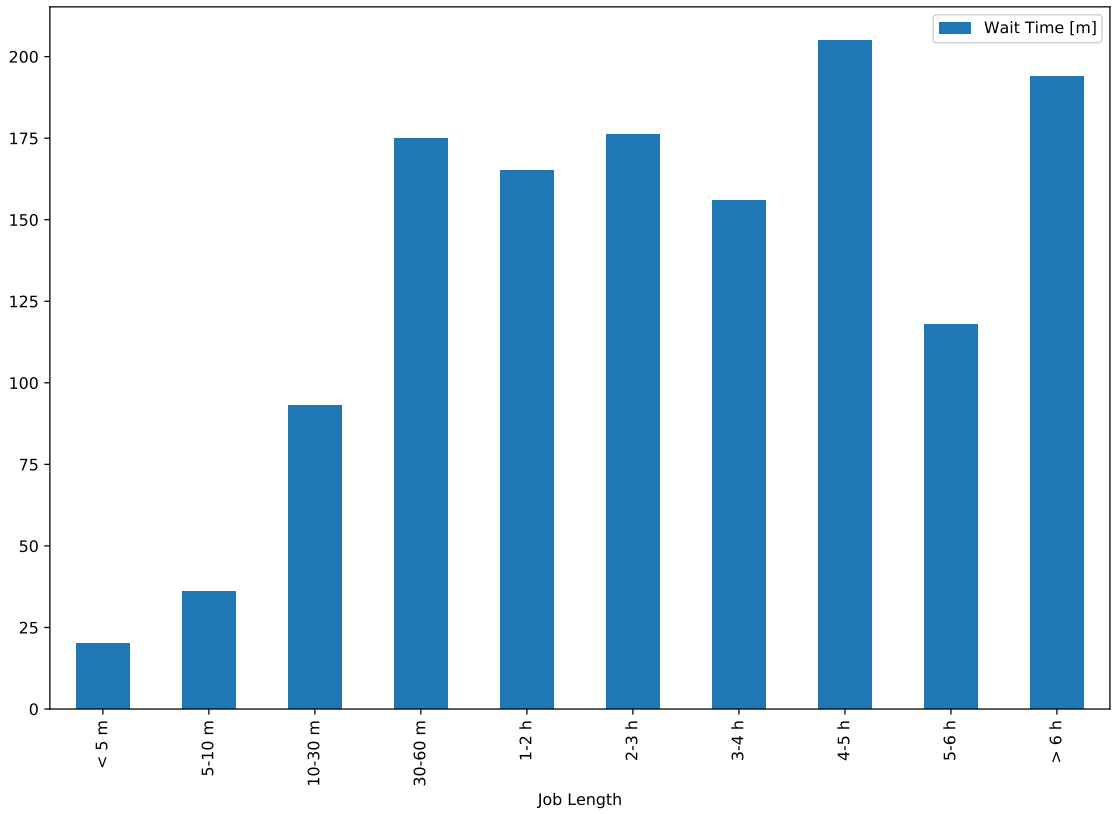

Figure 5.6: Average Wait Times For Single-Node Jobs

On the other hand, the scatter plot in figure 5.7 looks vastly different than before, with most of the extreme outliers gone, as they were all caused by the difficulty of scheduling jobs with more nodes. The scatter plot also shows why the 5-to-6-hour jobs in the previous bar graph had a lower average wait time, as there are not many jobs in that runtime range and no more prominent outliers.

The number of jobs at every complete hour is also much less than with all jobs, except for the three- and eight-hour marks, where it seems to be that more jobs were run at these runtimes. Also, there seem to be some horizontal ”lines” at about 900 and 1500 minutes, so it seems the scheduler prefers jobs that are waiting for a longer time.

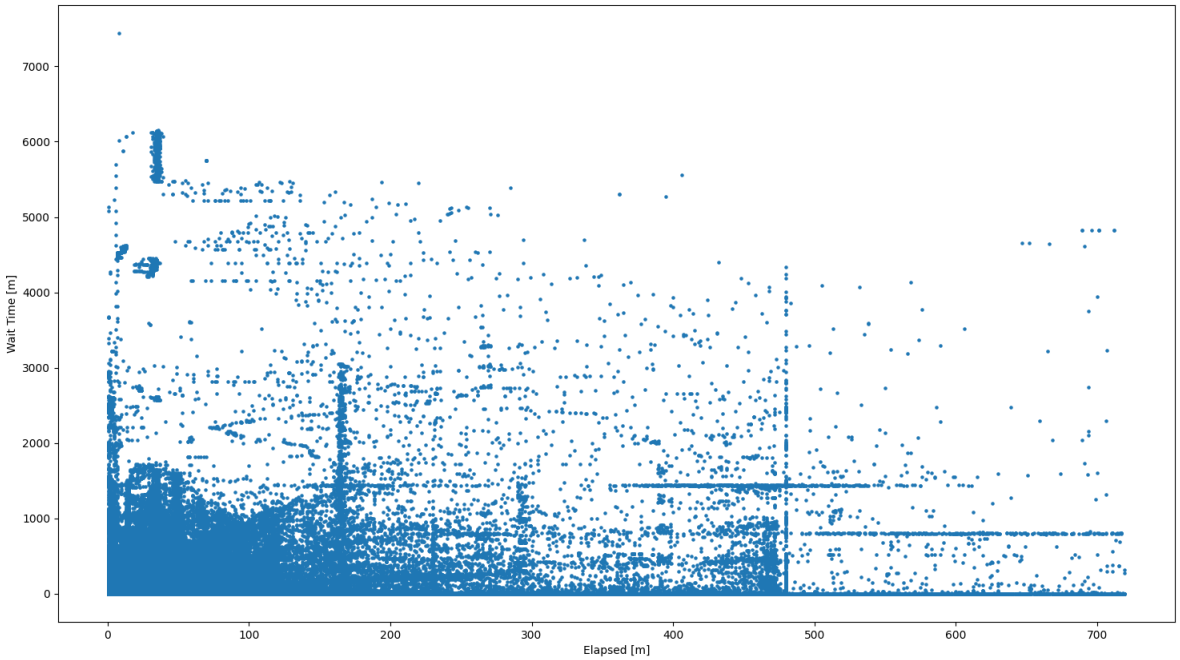

Figure 5.7: Wait Times Scatter Plot For Single-Node Jobs

6 Job-specific Cases

This chapter will feature a deeper dive into specific job cases using Mistral's job data and how it is possible to transfer them to the cloud. While not all claims will be proven directly in this chapter, some will be shown in chapter 7 in further detail.

6.1 Single-Node Jobs

As the previous analysis of the job data in section 5.3 shows, many single-node jobs do not utilise the full nodes. Sometimes, there is a good reason for the jobs to run on full nodes to utilise each node's 64 GiB of memory. However, in other times, it can also be the missing knowledge that a four-core job is not running faster on an exclusive node than on a shared node with four reserved cores.

This assumption was also confirmed by talking with the systems department, saying that sometimes non-parallelised Python scripts are run on exclusive nodes. On the other hand, the systems department also said there are single-node jobs, which benefit the scheduler, as it can use the single-node jobs with shorter runtimes to fill gaps. Jobs often create these gaps in the scheduling with high node counts where the scheduler keeps some nodes free until enough nodes are reserved. For example, if more nodes will only be available in 25 minutes, and there is a single-node job with a 20-minute time limit, the scheduler can then run the short job on a node reserved for the large job.

Also, as the wait times showed, the longer single-node jobs have much higher wait times. That means that the users running these jobs would also profit from outsourcing them to the cloud, reducing the wait times. These longer single-node jobs also create more problems for the scheduler, as the schedule rarely has gaps large enough for these kinds of jobs.

As section 5.2 showed, these longer single-node jobs make up about two-thirds of the total single-node hours. Therefore, these jobs are the first contenders to get outsourced into the cloud, providing many options for these jobs. These options include various CPU core and memory combinations, with nodes available that provide only four cores with 64 GiB of memory. These high memory options in the cloud are not as expensive due to the low core count, so even the jobs, which are not on the 'shared' partition due to memory constraints, could run more efficiently in the cloud.

Another aspect of why single-node jobs are well suited for the cloud is because they are not as dependent on the network as multi-node jobs. While the cloud services all provide specific compute nodes with good network connections, they provide a much larger number of instances without the dependency on a high-speed network.

6.2 Dual-Node Jobs

Dual-node jobs are still similar to single-node jobs, except they need good network bandwidth to run jobs parallel. So with about 50 GBit/s of network or better on the cloud nodes, the cloud nodes could still provide a value for dual-node jobs. This constraint reduces the number of different cloud nodes usable for this purpose, but the cloud services still have a good amount of nodes with a better network.

Additionally, the jobs that run on two nodes typically also utilise full nodes and are required by the scheduler to reserve full nodes. This kind of utilisation also means that there will be no real option to go to smaller nodes to replace these jobs in the cloud. Instead, the cloud nodes have to be similar to Mistral's nodes with somewhere in the range of about 30 cores and 64 GiB of RAM.

While the full utilisation also further reduces the number of different cloud nodes usable, there are still several options with the cloud services. With the availability of on-demand instances being great but expensive, the availability should be good enough, even with spot instances. Without the spot instances, the dual-node jobs probably would not be that profitable in the cloud compared to an on-premise cluster, but with them, it might be an option to put them in the cloud.

While, in the case of Mistral, the node hours of dual-node hours are pretty low, for a general analysis, these jobs still are an exciting aspect. Compared to the on-premise system, profitability will probably not be as good as with single-node jobs because single-node jobs can use substantial price reductions when using smaller nodes with fewer cores.

6.3 Multi-Node Jobs

As dual-node jobs already provide fewer options than single-node ones, the more nodes a job requires, the more challenging it gets to run these jobs cheaper than the on-premise cluster. One aspect causing this are the high node requirements like good network bandwidth available for the specific nodes. Like with the dual-node jobs, 50 GBit/s network bandwidth would be required, with higher network options like 100 or 200 GBit/s being a nice addition, as offered on some larger cloud nodes.

Another problem with these jobs is the availability of instances. With on-demand instances, the availability for such a job would be no problem, but the costs would be a lot higher than with the on-premise cluster. The cost would be significantly lower with reserved instances, but the lack of flexibility would require putting most of the jobs to the cloud to ensure high node usage for the entire reservation time. Both of these instance types were already looked at before in chapter 4, which concluded that an on-premise cluster would be more profitable.

So after this, only spot instances remain, just like they were already the choice for the earlier use cases. However, these spot instances also have a problem. To run, for example, a 100-node job on AWS, 100 nodes of the exact specification will be needed, and these also have to be in the same availability zone. This constraint is due to the connection of the nodes, which is required to distribute them between several nodes. As already mentioned, this requires a great network, and this network is only available in the local network, so these spot instances have to be in the same availability zone.

So if for this job suddenly 100 nodes in the same availability zone and at least 32 cores have to be found, the probability of one or more of these nodes being evicted would be quite high. Nevertheless, to get to this point, it would not be sure that 100 nodes of one kind would even be available at that specific time to start the job. Moreover, even if the 100 spot instances can be found, there would be many insecurities if the job would run successfully without single nodes being evicted. So as this makes spot instances also not an option for these jobs, it seems that general outsourcing of large jobs to the cloud might not be a good option.

6.4 Specific Nodes

While the general outsourcing of the larger jobs is no option, this is not necessarily the case for every job. With an on-premise system like Mistral, the nodes which are purchased stay in most cases the same for five years at least. In this amount of time, the types of jobs can change, and these new jobs could utilise some other hardware better than the current hardware in the cluster.

In the example of Mistral, the nodes are nearly exclusively using just Intel CPUs of the Haswell and Broadwell generations. These nodes were purchased for a runtime of six years, and if then after four years a user has a new application that is well optimised for GPUs, the user has to run it much slower on CPUs. Also, there might be another application that is able to utilise the AVX-512 instruction set, which is not supported by the Haswell and Broadwell architectures.

These are just a few examples of other architectures that would be able to improve the runtimes of specific jobs. Architectures using one or multiple GPUs, or CPUs with AVX-512, are available with all major cloud providers, so these might be an option. Another example might be ARM architectures, with the newest AWS compute generations also providing ARM CPUs.

While these instances using other hardware might be much more expensive than the nodes on the on-premise cluster, they could also provide much better runtimes, with GPU optimised applications often running multiple times faster on GPUs than on CPUs. If such a case would occur with an existing HPC system, in many cases, adjustments to the hardware can only be made with the next iteration of the cluster, where the whole cluster gets replaced.

With Mistral, the nodes were purchased in 2015, so as they would be replaced soon, this would not be a big problem currently. However, suppose this new architecture requirement of an application comes up one or two years after purchasing a cluster. In that case, there might be the possibility that this specific application has to run on non-optimised hardware for three or four years.

While this use case could be a valid reason to plan with at least some cloud capacity, it is not easy to use it for a cost function. There is no specific value to put on the architecture choice because it depends on the applications a cluster is running and how often these applications change. Also, it is impossible to predict if the users suddenly need such specific hardware for their applications in many cases, so it is not easy to plan for this. Because of these difficulties, this use case will also not be a part of the cost function. However, it remains a valid reason for a cluster owner to decide, at least with partial cloud usage, to provide more flexibility with the hardware.

7 Cost Function

This chapter will establish the costs of Mistral's nodes and formalise a cost function to determine how the cloud can be utilised optimally for HPC systems.

7.1 Mistral Node Costs

First, it has to be established what the cost of a single node hour on Mistral is to compare the job costs against the costs in the cloud. Mistral was built for use over six years, so the purchase costs of about 35 million euros need to be distributed over these six years. Additional to these purchase costs come the costs of running the cluster. The financial statements of the German Climate Computing Center have to be public, so they can be used to extract an approximation of the running costs. The newest financial report is from 2019 at the "Bundesanzeiger", and with the following report not available until early 2022, this will be the basis for the running costs.

The power consumption is the first and also largest aspect of the running costs, with the energy and water costs being nearly 1.8 million euros in 2019 [Gmb21]. Of course, the offices also cause some energy and water consumption, but it would be safe to assume that about 1.5 million euros per year are the cluster's energy costs. Another considerable aspect of the running costs is the rent for the building, with Mistral taking up about 1.5 of the building's five floors just for the cluster. With the yearly rent of 848,000 euros, about 250,000 euros can be taken as the rent just for Mistral using these floor counts [Gmb21].

The last aspect is the personnel costs for running the cluster, with fewer people required to maintain the hardware if the hardware is outsourced to the cloud. With Mistral, on the other hand, their systems department only sets up infrastructure like the scheduler, but that would be required in the cloud as well. The hardware is maintained by staff from Bull, as Bull also built Mistral. So while this working group would be fully required in the cloud as well in Mistral's case, this value will remain in the function, as in other cases where the cluster is maintained by its own staff, it could be relevant.

In total, these costs add up to a cluster cost of about 7.6 million euros per year:
$serverCost/years + powerCost + rent + personnelCost$
$= 35,000,000€/6 + 1,500,000€ + 250,000€ + 0€$
$\approx 7,600,000€$

Dividing by the number of servers, days in a year, and hours of a day, the result is the node cost per hour of a single Mistral node, ready for comparison with the cloud:
$(yearlyClusterCost/(days \cdot hours))/nodeNumber$
$= (7,600,000€/(365.25 \cdot 24))/3,300$
$\approx 0.263€$

7.2 Single-Node Function

As established before, single-node jobs are the best fit for outsourcing jobs to cloud services. These jobs will either run as an on-demand instance to guarantee the job does not get cancelled or on spot instances to significantly reduce costs. While the on-demand instances are relatively expensive, they can still be cost-effective, especially for jobs with a lower core count.

To compare against the previously established cost of a Mistral node, for each cloud service, some fitting nodes for the different number of threads/vCPUs have to be selected. Additionally, each of these instances will also need an alternative with 64 GiB of RAM, as some of the jobs run on full nodes to utilise more memory.

On the side of Mistral, the cost function will require not only the costs of a node hour but also some job data. The other two relevant aspects are the number of node hours for each core count and the percentage of jobs with high memory usage. As the typical core counts for the cloud instances are 4, 8 and 16, the number of node hours will be split at these core counts, with the jobs with more cores going to a node that represents a full Mistral node. Also, only one hour or longer jobs will be considered, with jobs shorter than one hour not considered as the scheduler can profit from these jobs.

These job statistics were already discussed before, but the job data will only be looked at for 1-4 threads, 5-8 threads, 9-16 threads and 17+ threads. Table 7.1 shows a quick overview of the number of jobs for each number of threads and node hours which will be used as the values for the cost function.

NCPUs	Jobs	Node Hours
1-4 Cores	33,382	129,538
5-8 Cores	3,773	10,434
9-16 Cores	9,622	20,266
17+ Cores	59,853	317,434

Table 7.1: Job statistics for long single-node jobs

For the cost functions, the following variables are relevant:
t_{node4c}: Number of 1-4 core node hours
t_{node8c}: Number of 5-8 core node hours
$t_{node16c}$: Number of 9-16 core node hours
$t_{nodeBig}$: Number of 17+ core node hours
$c_{cluster}$: Cluster cost per node and hour
$c_{cloud4c}$: Cloud cost per 4-core node and hour
$c_{cloud8c}$: Cloud cost per 8-core node and hour
$c_{cloud16c}$: Cloud cost per 16-core node and hour
$c_{cloudBig}$: Cloud cost per full node and hour
p_{mem}: Percentage of jobs using the full memory
$c_{cloud4cMem}$: Cloud cost per 4-core node with 64GB RAM and hour
$c_{cloud8cMem}$: Cloud cost per 8-core node with 64GB RAM and hour
$c_{cloud16cMem}$: Cloud cost per 16-core node with 64GB RAM and hour

The following definition of the cost difference between the cloud and the cluster can be created using these variables:
$c_{diff4c} = ((1 - p_{mem}) \cdot t_{node4c} \cdot (c_{cluster} - c_{cloud4c})) + (p_{mem} \cdot t_{node4c} \cdot (c_{cluster} - c_{cloud4cMem}))$
$c_{diff8c} = ((1 - p_{mem}) \cdot t_{node8c} \cdot (c_{cluster} - c_{cloud8c})) + (p_{mem} \cdot t_{node8c} \cdot (c_{cluster} - c_{cloud8cMem}))$
$c_{diff16c} = ((1 - p_{mem}) \cdot t_{node16c} \cdot (c_{cluster} - c_{cloud16c})) + (p_{mem} \cdot t_{node16c} \cdot (c_{cluster} - c_{cloud16cMem}))$
$c_{diffBig} = t_{nodeBig} \cdot (c_{cluster} - c_{cloudBig})$

The cost differences from these functions have to be > 0. Otherwise, the selected cloud nodes are more expensive than the cluster and jobs of this size should be excluded. The cost differences that are > 0 can then be added up to receive the total cost that could be saved by running these specific jobs in the cloud.

Some of the values used for the functions are already set for the calculations using Mistral's data which will be used in the following parts:
$t_{node4c} = 129,538$
$t_{node8c} = 10,434$
$t_{node16c} = 20,266$
$t_{nodeBig} = 317,434$
$c_{cluster} = 0.263€$

7.2.1 On-Demand Instances

For the on-demand instances, table 7.2 shows the different nodes and their prices selected for the different cloud providers. The prices of these nodes will again be taken from the same data centres used for the calculations in chapter 4. Additionally, nodes for general computing are avoided, as HPC nodes typically have the highest and best comparable performance. The only exception to this is Azure, with them only offering HPC nodes without hyperthreading starting at eight cores.

As their HPC instances would give them a huge disadvantage for the lower core counts and with Azure also providing benchmarks for their other compute nodes, showing comparable performance to their HPC nodes [Azuc], these nodes will be allowed too. What also sets the Azure HPC nodes apart from their other compute nodes is the Infiniband network, which is not needed for single-node jobs. For AWS and Google Cloud, this exception would not make a significant difference as their HPC nodes with a low number of vCPUs already have competitive pricing.

	AWS	Azure	Google Cloud
4 Core	c5a.xlarge(0.15€)	D4 v4(0.19€)	c2-standard-4(0.20€)
4 Core/High RAM	r5a.2xlarge(0.47€)	E8a v4(0.51€)	c2 Custom(4C/64GB)(0.40€)
8 Core	c5a.2xlarge(0.30€)	D8 v4(0.39€)	c2-standard-8(0.39€)
8 Core/High RAM	r5a.2xlarge(0.47€)	E8a v4(0.51€)	c2 Custom(8C/64GB)(0.53€)
16 Core	c5a.4xlarge(0.59€)	D16 v4(0.78€)	c2-standard-16(0.78€)
16 Core/High RAM	r5a.4xlarge(0.93€)	D16 v4(0.78€)	c2-standard-16(0.78€)
Full Node	c5a.16xlarge(2.38€)	HB60rs(2.50€)	c2-standard-60(2.94€)

Table 7.2: Cloud on-demand configurations for single-node jobs

There is an option for jobs on Google Cloud to create a custom configuration for the "c2-standard" instances. These custom configurations were used for the 4- and 8-core jobs using 64 GiB of RAM to create cheaper configurations with few CPU cores but much memory.

These values can then be substituted in the previously created cost difference functions. The only remaining value is then p_{mem}, and for the calculations, 20% will be used as an estimate of how many low-thread single-node jobs require the entire memory. For AWS these substitutions result in the following functions:

$$\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.15€)) + (0.2\cdot 129,538\cdot(0.263€-0.47€)\\
&\approx 6,350€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.30€)) + (0.2\cdot 10,434\cdot(0.263€-0.47€)\\
&\approx -740€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.59€)) + (0.2\cdot 20,266\cdot(0.263€-0.93€))\\
&\approx -8,000€\\
c_{diffBig} &= 317,434\cdot(0.263€-2.38€)\\
&\approx -672,000€
\end{aligned}$$

Although these results show a potential cost saving of more than 6,000€ with the 4-core instances, these savings can quickly be eaten up by the work needed to set up the scheduler for the cloud system. The cost savings could also be slightly higher if the 4-core high memory jobs remained on the cluster, as these jobs are more expensive on AWS than on Mistral. For every job larger than that, the AWS on-demand instances are more expensive than the cluster, so this does not seem to be the best way of outsourcing.

Azure then shows a slightly worse image, with the cost savings for 4-core jobs being even smaller, while the larger jobs are slightly more expensive than with AWS:

$$\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.19€))+(0.2\cdot 129,538\cdot(0.263€-0.51€)\\
&\approx 1,170€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.39€))+(0.2\cdot 10,434\cdot(0.263€-0.51€)\\
&\approx -1,580€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.78€))+(0.2\cdot 20,266\cdot(0.263€-0.78€))\\
&\approx -10,480€\\
c_{diffBig} &= 317,434\cdot(0.263€-2.50€)\\
&\approx -710,100€
\end{aligned}$$

With Google Cloud, the image is also quite similar, with the 4-core instances being a bit cheaper than with Azure, but not with a significant amount:

$$\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.20€))+(0.2\cdot 129,538\cdot(0.263€-0.40€)\\
&\approx 2,980€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.39€))+(0.2\cdot 10,434\cdot(0.263€-0.53€)\\
&\approx -1,620€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.78€))+(0.2\cdot 20,266\cdot(0.263€-0.78€))\\
&\approx -10,480€\\
c_{diffBig} &= 317,434\cdot(0.263€-2.94€)\\
&\approx -849,770€
\end{aligned}$$

Of course, these calculations, like in chapter 4, assume the requirement of running the jobs in the EU to have the data secure from the US authorities. If the location of the nodes is not relevant, it is possible to choose the cheapest location for each cloud provider. This location change results in the prices mentioned in table 7.3 using the server locations Ohio for AWS, East US for Azure and Iowa for Google Cloud.

	AWS	Azure	Google Cloud
4 Core	c5a.xlarge(0.13€)	D4 v4(0.16€)	c2-standard-4(0.18€)
4 Core/High RAM	r5a.2xlarge(0.39€)	E8a v4(0.43€)	c2 Custom(4C/64GB)(0.37€)
8 Core	c5a.2xlarge(0.26€)	D8 v4(0.32€)	c2-standard-8(0.36€)
8 Core/High RAM	r5a.2xlarge(0.39€)	E8a v4(0.43€)	c2 Custom(8C/64GB)(0.48€)
16 Core	c5a.4xlarge(0.53€)	D16 v4(0.65€)	c2-standard-16(0.71€)
16 Core/High RAM	r5a.4xlarge(0.77€)	D16 v4(0.65€)	c2-standard-16(0.71€)
Full Node	c5a.16xlarge(2.11€)	HB60rs(1.92€)	c2-standard-60(2.68€)

Table 7.3: Cheapest cloud on-demand configurations for single-node jobs

With these values, the following calculations show how the price differences improved:

1. AWS

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.13€))+(0.2\cdot 129,538\cdot(0.263€-0.39€)\\
&\approx 10,490€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.26€))+(0.2\cdot 10,434\cdot(0.263€-0.39€)\\
&\approx -240€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.53€))+(0.2\cdot 20,266\cdot(0.263€-0.77€))\\
&\approx -6,380€\\
c_{diffBig} &= 317,434\cdot(0.263€-2.11€)\\
&\approx -586,300€
\end{aligned}
$$

2. Azure

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.16€))+(0.2\cdot 129,538\cdot(0.263€-0.43€)\\
&\approx 6,350€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.32€))+(0.2\cdot 10,434\cdot(0.263€-0.43€)\\
&\approx -820€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.65€))+(0.2\cdot 20,266\cdot(0.263€-0.65€))\\
&\approx -7,840€\\
c_{diffBig} &= 317,434\cdot(0.263€-1.92€)\\
&\approx -525,990€
\end{aligned}
$$

3. Google Cloud

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.18€))+(0.2\cdot 129,538\cdot(0.263€-0.37€)\\
&\approx 5,830€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.36€))+(0.2\cdot 10,434\cdot(0.263€-0.48€)\\
&\approx -1,260€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.71€))+(0.2\cdot 20,266\cdot(0.263€-0.71€))\\
&\approx -9,060€\\
c_{diffBig} &= 317,434\cdot(0.263€-2.68€)\\
&\approx -767,240€
\end{aligned}
$$

For all of these instances, the value is significantly better than with the European instances. However, despite these improvements, with none of the cloud providers, the 8-core instances get to a point where they are cheaper than Mistral too. While the 8-core instances on AWS are very close to the break-even point and Azure is not far behind, too, Mistral is still slightly cheaper.

7.2.2 Spot Instances

As the on-demand instances already can be more cost-effective than the cluster for low-core configurations, this can only be improved by using spot instances. While the risk of job eviction remains with these instances, there are several options to reduce this risk. The first option is to outsource just the jobs that provide checkpointing and can create a checkpoint when they receive an eviction notice from the cloud provider.

This option does limit the number of jobs that can be put on the cloud system, so it would be even better to reduce the risk of eviction as much as possible. So this leads to the second option, which is not to select a fixed instance at a specific location but to provide a list of eligible instances and locations. For example, it would be possible with AWS to select the different 8-core HPC instances on all European locations as eligible nodes. The AWS spot scheduler would then look for the node with the best availability, which also means it has the lowest chance of being evicted. Leaving this to the AWS scheduler can raise the price a bit, as the cheapest spot instance does not necessarily have the best availability, but the chance of the job being evicted would be very low.

The cheapest and the most expensive data centre for spot instances will then be taken from all European data centres to represent the range in prices from the cloud. These can then be used to select an average price, roughly representing the average price of the nodes the cloud scheduler will select from the list of eligible nodes. For AWS, Stockholm is the cheapest location, and Dublin is the most expensive location with roughly a 20 per cent price difference. As Frankfurt is roughly in the middle of the prices of the European locations, it will be used for the analysis.

For Azure, only West Europe has all of the selected nodes, so this will be the selected location for the European costs. With Google Cloud, there are three European servers with HPC spot instances. As Belgium is the cheapest location and London is the most expensive location, the prices in the Netherlands will be taken, although they are only marginally higher than in Belgium.

This results in the following selection of nodes for EU nodes using the servers:

	AWS	Azure	Google Cloud
4 Core	c5ad.xlarge(0.06€)	D4 v4(0.04€)	c2-standard-4(0.05€)
4 Core/High RAM	r5n.2xlarge(0.14€)	E8a v4(0.11€)	c2 Custom(4C/64GB)(0.10€)
8 Core	c5d.2xlarge(0.11€)	D8 v4(0.08€)	c2-standard-8(0.10€)
8 Core/High RAM	r5n.2xlarge(0.14€)	E8a v4(0.11€)	c2 Custom(8C/64GB)(0.13€)
16 Core	c5.4xlarge(0.23€)	D16 v4(0.16€)	c2-standard-16(0.19€)
16 Core/High RAM	r5dn.4xlarge(0.25€)	D16 v4(0.16€)	c2-standard-16(0.19€)
Full Node	c5a.16xlarge(0.92€)	HB60rs(0.53€)	c2-standard-60(0.72€)

Table 7.4: Cloud spot configurations for single-node jobs

These instances have a pretty significant discount compared to the on-demand instances, so even some of the 16-core nodes are cheaper than a Mistral node. When putting these values into the cost functions, again with 20% high memory jobs, the result looks as follows.

1. AWS

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.06€)) + (0.2\cdot 129,538\cdot(0.263€-0.14€)\\
&\approx 24,220€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.11€)) + (0.2\cdot 10,434\cdot(0.263€-0.14€)\\
&\approx 1,530€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.23€)) + (0.2\cdot 20,266\cdot(0.263€-0.25€))\\
&\approx 590€\\
c_{diffBig} &= 317,434\cdot(0.263€-0.92€)\\
&\approx -208,550€
\end{aligned}
$$

2. Azure

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.04€)) + (0.2\cdot 129,538\cdot(0.263€-0.11€)\\
&\approx 27,070€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.08€)) + (0.2\cdot 10,434\cdot(0.263€-0.11€)\\
&\approx 1,850€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.16€)) + (0.2\cdot 20,266\cdot(0.263€-0.16€))\\
&\approx 2,090€\\
c_{diffBig} &= 317,434\cdot(0.263€-0.53€)\\
&\approx -84,750€
\end{aligned}
$$

3. Google Cloud

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.05€)) + (0.2\cdot 129,538\cdot(0.263€-0.10€)\\
&\approx 26,300€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.10€)) + (0.2\cdot 10,434\cdot(0.263€-0.13€)\\
&\approx 1,640€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.19€)) + (0.2\cdot 20,266\cdot(0.263€-0.19€))\\
&\approx 1,480€\\
c_{diffBig} &= 317,434\cdot(0.263€-0.72€)\\
&\approx -145,070€
\end{aligned}
$$

Unlike the on-demand instances, using the spot instances, every cloud provider has options with lower pricing than the on-site cluster. All cloud providers are also cheaper for the 4-core instances, using spot instances, but for the 8- and 16-core instances too. They also provide cost reductions of about 30,000€ per year using these spot instances, which would make nearly 200,000€ for Mistral's lifetime.

As these prices are also not the cheapest options available, these amounts of cost reductions are pretty safe with spot instances, as there are also instances that might be up to 10% cheaper. However, as before with the on-demand instances, there is the option to run these instances globally and not only in Europe, which reduces the spot prices even further. For this comparison, the cheapest spot instances which are available will be taken to provide a best-case scenario of how large the savings can be using the jobs of Mistral. Therefore, AWS will be represented by the Ohio data centre, Google Cloud by the Iowa data centre, and Azure by a mix of "West Central US" and "East US". While the "West Central US" data centre is cheaper for the D4 v4, D8 v4, D16 v4 and E8a v4 nodes, it does not have larger nodes, so these have to be taken from "East US".

	AWS	Azure	Google Cloud
4 Core	c5.xlarge(0.03€)	D4 v4(0.03€)	c2-standard-4(0.04€)
4 Core/High RAM	r5n.2xlarge(0.07€)	E8a v4(0.08€)	c2 Custom(4C/64GB)(0.09€)
8 Core	c5d.2xlarge(0.07€)	D8 v4(0.06€)	c2-standard-8(0.09€)
8 Core/High RAM	r5n.2xlarge(0.07€)	E8a v4(0.08€)	c2 Custom(8C/64GB)(0.12€)
16 Core	c5.4xlarge(0.13€)	D16 v4(0.12€)	c2-standard-16(0.17€)
16 Core/High RAM	r5a.4xlarge(0.14€)	D16 v4(0.12€)	c2-standard-16(0.17€)
Full Node	c5a.16xlarge(0.53€)	HB60rs(0.40€)	c2-standard-60(0.65€)

Table 7.5: Cheapest cloud spot configurations for single-node jobs

The prices from 7.5 then result in the following calculations for the three cloud providers:

1. AWS

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.03€)) + (0.2\cdot 129,538\cdot(0.263€-0.07€)\\
&\approx 29,150€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.07€)) + (0.2\cdot 10,434\cdot(0.263€-0.07€)\\
&\approx 2,010€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.13€)) + (0.2\cdot 20,266\cdot(0.263€-0.14€))\\
&\approx 2,650€\\
c_{diffBig} &= 317,434\cdot(0.263€-0.53€)\\
&\approx -84,750€
\end{aligned}
$$

2. Azure

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.03€))+(0.2\cdot 129,538\cdot(0.263€-0.08€)\\
&\approx 28,890€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.06€))+(0.2\cdot 10,434\cdot(0.263€-0.08€)\\
&\approx 2,080€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.12€))+(0.2\cdot 20,266\cdot(0.263€-0.12€))\\
&\approx 2,900€\\
c_{diffBig} &= 317,434\cdot(0.263€-0.40€)\\
&\approx -43,490€
\end{aligned}
$$

3. Google Cloud

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 129,538\cdot(0.263€-0.04€))+(0.2\cdot 129,538\cdot(0.263€-0.09€)\\
&\approx 27,590€\\
c_{diff8c} &= ((1-0.2)\cdot 10,434\cdot(0.263€-0.09€))+(0.2\cdot 10,434\cdot(0.263€-0.12€)\\
&\approx 1,740€\\
c_{diff16c} &= ((1-0.2)\cdot 20,266\cdot(0.263€-0.17€))+(0.2\cdot 20,266\cdot(0.263€-0.17€))\\
&\approx 1,880€\\
c_{diffBig} &= 317,434\cdot(0.263€-0.65€)\\
&\approx -122,850€
\end{aligned}
$$

These calculations show a very similar amount of cost savings between AWS and Azure, with both being about 33,000€ per year. While Google Cloud is slightly more expensive, the about 30,000€ of cost savings would still result in nearly 200,000€ of potential savings during Mistrals' lifetime, with Azure and AWS saving 200,000€.

The number of nodes would have to be reduced with the purchase already to achieve these cost savings. The total node hours of the nodes being outsourced must be divided by the number of hours in a year to calculate how many nodes have to be purchased less. As for all three cloud services, jobs up to 16 cores are more cost-effective in the cloud. Therefore, the calculations do not have to be split up between the cloud providers. Additionally, the calculations are also valid for the European servers, as the 16-core jobs are cheaper there than on Mistral too. The results can are shown in the following calculation:

$$
\begin{aligned}
NNodes_{outsource} &= (t_{node4c}+t_{node8c}+t_{node16c})/(days\cdot hours)\\
&= (129,538+10,434+20,266)/(365.25\cdot 24)\\
&\approx 18.3\ nodes
\end{aligned}
$$

So only about 18 nodes to outsource does not sound like much, but these nodes already can make up cost savings of more than 200,000€ for the clusters lifespan. Another option would be to keep the size of the cluster and get the cloud nodes as an addition to optimise the clusters scheduling. This option, of course, would require the costs of the cloud nodes not exceeding the cluster's budget.

With Mistral only having a small number of 8- and 16-core jobs, it would be interesting to look at a fictional job setup with more 8- and 16-core jobs. For this scenario, the number of total single-node node hours will also be increased to provide data for a fictional data centre that uses more single-node jobs. The calculations will be done with 200,000 4-core jobs, 100,000 8-core jobs, 100,000 16-core jobs, and 400,000 jobs using complete nodes. The node setup will be assumed to be the same as with the Mistral cluster, so also with costs of 0.263€ per node hour. This scenario results in the following calculations using the cheapest cloud instances available:

1. AWS

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 200,000\cdot(0.263€-0.03€)) + (0.2\cdot 200,000\cdot(0.263€-0.07€)\\
&= 45,000€\\
c_{diff8c} &= ((1-0.2)\cdot 100,000\cdot(0.263€-0.07€)) + (0.2\cdot 100,000\cdot(0.263€-0.07€)\\
&= 19,300€\\
c_{diff16c} &= ((1-0.2)\cdot 100,000\cdot(0.263€-0.13€)) + (0.2\cdot 100,000\cdot(0.263€-0.14€))\\
&= 13,100€\\
c_{diffBig} &= 400,000\cdot(0.263€-0.53€)\\
&= -106,800€
\end{aligned}
$$

2. Azure

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 200,000\cdot(0.263€-0.03€)) + (0.2\cdot 200,000\cdot(0.263€-0.08€)\\
&= 44,600€\\
c_{diff8c} &= ((1-0.2)\cdot 100,000\cdot(0.263€-0.06€)) + (0.2\cdot 100,000\cdot(0.263€-0.08€)\\
&= 19,900€\\
c_{diff16c} &= ((1-0.2)\cdot 100,000\cdot(0.263€-0.12€)) + (0.2\cdot 100,000\cdot(0.263€-0.12€))\\
&= 14,300€\\
c_{diffBig} &= 400,000\cdot(0.263€-0.40€)\\
&= -54,800€
\end{aligned}
$$

3. Google Cloud

$$
\begin{aligned}
c_{diff4c} &= ((1-0.2)\cdot 200,000\cdot(0.263€-0.04€)) + (0.2\cdot 200,000\cdot(0.263€-0.09€) \\
&= 42,600€ \\
c_{diff8c} &= ((1-0.2)\cdot 100,000\cdot(0.263€-0.09€)) + (0.2\cdot 100,000\cdot(0.263€-0.12€) \\
&= 16,700€ \\
c_{diff16c} &= ((1-0.2)\cdot 100,000\cdot(0.263€-0.17€)) + (0.2\cdot 100,000\cdot(0.263€-0.17€)) \\
&= 9,300€ \\
c_{diffBig} &= 400,000\cdot(0.263€-0.65€) \\
&= -154,800€
\end{aligned}
$$

In this scenario, with cost savings between 68,000€ and 79,000€ per year, outsourcing is quite a good option. With the cost savings upscaled to a 6-year lifespan, it would be possible to save about 450,000€ in total with this setup of node hours. Looking at Azure, even with outsourcing all of the longer single-node jobs, it would still be possible to save money, despite the jobs using the full nodes being more expensive in the cloud. When outsourcing all of these jobs on Azure, the cost savings would still be 24,000€ per year, just slightly less than the actual job data from Mistral.

The calculation of the number of nodes that could be removed from the data centre also changes with these new values. With the increased amount of node hours, the following two calculations are for how many fewer nodes are required by outsourcing all jobs up to 16-cores or just all single-node jobs.

1. Up to 16 cores:

$$
\begin{aligned}
NNodes_{outsource} &= (t_{node4c} + t_{node8c} + t_{node16c})/(days\cdot hours) \\
&= (200,000 + 100,000 + 100,000)/(365.25\cdot 24) \\
&\approx 45.6\ nodes
\end{aligned}
$$

2. All jobs:

$$
\begin{aligned}
NNodes_{outsource} &= (t_{node4c} + t_{node8c} + t_{node16c} + t_{nodeBig})/(days\cdot hours) \\
&= (200,000 + 100,000 + 100,000 + 400,000)/(365.25\cdot 24) \\
&\approx 91.3\ nodes
\end{aligned}
$$

With these amounts of nodes, they would make a much more significant impact on the node setup of the cluster. By outsourcing all of these jobs, the size of Mistral could be reduced by nearly 100 nodes, putting its total number of nodes to 3,200.

7.3 Dual-Node Function

For the dual-node function, the story is a bit different. With the dual-node jobs required to allocate the full nodes, the replacement nodes also always have to replace the full nodes. Additionally, the replacement nodes will need at least 50 GBit/s network, as the jobs have to communicate between the two nodes.

The cost function then looks pretty similar to the cost function of the single-node instances with full replacement nodes but with only one cost function. The variables used, on the other hand, are much fewer and a bit different:
t_{dual}: Number of node hours of dual-node jobs
$c_{cluster}$: Cluster cost per node and hour
c_{cloud}: Cloud cost per full node and hour

The resulting function then looks as follows:
$c_{diff} = t_{dual} \cdot (c_{cluster} - c_{cloud})$

For the cluster costs, the 0.263€ per node hour still stand, and the node hours for dual-node jobs were already established in table 5.2. Filling in these values, the function looks as follows, with only the cloud costs missing:
$c_{diff} = 165,220 \cdot (0.263€ - c_{cloud})$

For the cloud costs, the single-node calculations already showed that on-demand instances are no option for full replacement nodes, as they are far more expensive than Mistral's nodes. While the spot instances were still more expensive, the spot pricing is much closer to the costs of Mistral. Therefore the following instances will be taken for spot instances in Europe and the US, using the same data centres as for the single-node jobs:

	AWS	Azure	Google Cloud
EU instances	c5n.18xlarge(1.04€)	HB60rs(0.53€)	c2-standard-60(0.72€)
US instances	c5n.18xlarge(0.59€)	HB60rs(0.40€)	c2-standard-60(0.65€)

Table 7.6: Cloud spot configurations for dual-node jobs

Except for the AWS instances, the same instances as with the single-node jobs are used. For AWS, it was necessary to switch the nodes as the previously used "c5a.16xlarge" nodes only provide 20 GBit/s networking. The "c5n" family of nodes has to be used to have higher network bandwidths, with the "c5n.18xlarge" nodes being the best comparable to Mistral's nodes.

When putting these values into the function, they result in the following cost differences.

1. AWS EU

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 1.04€) \\ &\approx -128,400€ \end{aligned}$$

2. Azure EU

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 0.53€) \\ &\approx -44,100€ \end{aligned}$$

3. Google Cloud EU

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 0.72€) \\ &\approx -75,500€ \end{aligned}$$

4. AWS US

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 0.59€) \\ &\approx -54,000€ \end{aligned}$$

5. Azure US

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 0.40€) \\ &\approx -22,600€ \end{aligned}$$

6. Google Cloud US

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 0.65€) \\ &\approx -63,900€ \end{aligned}$$

While AWS and Google Cloud are much more expensive than Mistral, at least with the US instances of Azure, the cost difference is not as significant. Nevertheless, this variant still is about 20,000€ more expensive per year than Mistral, which is why it would not make sense to outsource these jobs to the cloud.

While this use case does not fit Mistral, it does not mean it has to be a bad value for every cluster. With a cluster that has smaller or more expensive nodes, the differences between the cloud costs and the cluster costs could quickly get much smaller. For example, if a cluster has better nodes than Mistral with more than 100 cores per node, the cloud costs would not be much higher, as Azure provides a cheap option. Currently, the HB120rs v2 instance with 120 cores is only 0.47€ in "North Central US", and therefore only 0.07€ more expensive than the instance used to compare to Mistral. So it is possible to gain much performance for a slight price difference with some instances in the cloud, while it might be possible that the cost difference of the better cluster nodes is more than 0.07€.

With this specific HB120rs v2 instance, even Mistral's dual-node jobs could be optimised further. As this instance has about 2-3 times the performance of the instance used previously for the function [Azuc], it would be possible that this instance might be able to run the dual-node jobs faster on a single node than Mistral on two nodes. The memory would be no problem, too, as the HB120rs v2 instance has 456 GB of memory, while Mistral's nodes only have 64 GB.

So to use this instance for the cost function, the cost of the cloud node would have to be divided by two to represent the requirement of one less node per job. This then results in the following function:

$$\begin{aligned} c_{diff} &= 165,220 \cdot (0.263€ - 0.47€/2) \\ &\approx 4,600€ \end{aligned}$$

While this only brings a small cost saving of about 4,600€, it still shows the possibility of a cost-saving even for dual-node jobs. The low number of node hours of the dual-node jobs also makes the cost-saving pretty small. The cost-savings would be much more significant with a higher number of node hours like with the single-node jobs.

So while the single-node jobs provide a more obvious way to optimise the costs of Mistral, there is a way even to optimise the costs of dual-node jobs. While this optimisation is not as simple, it might even provide some lower job complexity as only a single node is required for the same job.

8 Summary, Conclusion And Future Work

This chapter will summarise the thesis, and a conclusion of the results gained from this thesis will be taken. Additionally, some discussion of possible future work will follow to build upon the results of this thesis.

8.1 Summary

Cloud services are used more widely year after year and are also attempting to gain momentum in the HPC space. This thesis used these cloud HPC offerings to analyse the possibilities and profitability of HPC in the cloud.

As a start, the cloud services and the cluster were introduced, and some related work was discussed. Chapter 4 then used a simple approach to compare a complete replacement of an on-premise cluster with the cloud. While this approach showed a direct comparison to the actual cluster, it quickly showed that this approach does not lead to a recommendation of the cloud. Therefore, chapter 5 then moved on to look at the type of jobs a specific cluster was running.

Using the collected data, the following chapter 6 then analysed the data further to establish some specific job cases for a more complex cost function. As this showed that only long single-node jobs and dual-node jobs are good use-cases for the cloud, these job types were then taken for the cost functions. The cost functions were then finally created and used in chapter 7 to create a better comparison to the cloud than with the previous approach. Different cloud instances were also looked at for these cost functions, as the cloud providers give cheaper options with some disadvantages.

8.2 Conclusion

As chapter 4 showed, replacing an entire cluster with a cloud system is far more expensive than an own cluster. Even the storage of Mistral alone was more expensive in the cloud for the six years lifetime than the total purchase cost of Mistral. Also, when considering all of the running costs of Mistral, a complete replacement would still not be an option as the only cloud instances, getting closer to the cost of Mistral, are spot instances that are not practicable for an entire cluster.

The cost functions of chapter 7 then showed a vastly different image with only a limited number of jobs considered. With these specific jobs, outsourcing to the cloud can make much more sense, as some single- and even dual-node jobs provide a good chance for cost-savings. For these jobs, the spot instances also make much more sense, and so for the single-node jobs, it is possible to have significant cost reductions using the spot instances.

What also showed with the dual-node jobs is that the outsourcing decision is also very dependent on the cloud nodes that are selected as a replacement. While a simple replacement of these jobs on Mistral using spot instances was more expensive than keeping them on the cluster, it was more cost-effective to use vastly better nodes. With nodes like on Mistral, this option might work quite well, as the 120-core cloud instances could run even faster than the dual-node jobs on Mistral since there is no network communication required.

All in all, outsourcing single- or dual-node jobs to the cloud can provide a good value but is hugely dependent on the local cluster's configuration and the requirements for the cloud instances. With the requirement of European cloud servers, for example, the value of the cloud is already slightly worse than with US servers being an option. So while the cloud is not ready to replace a complete HPC cluster, it is an option for some specific use cases, like single- and dual-node jobs, and is worth considering for those cases.

Another factor that was only mentioned briefly is the option to use the cloud for newer or other CPU architectures, which can further improve the value of a cloud solution. As the cloud providers constantly use new CPU generations for new instances, the jobs could be run continuously on the newest hardware to reduce the execution times of the applications. Also, the cloud providers have multiple different node architectures, which could fit better for specific applications. These options can make a big difference in the decision to use the cloud, but it is hard to put a number on that advantage as it is very dependent on the type of jobs on the cluster.

Storage is also an essential factor with Mistral, but as chapter 4 showed, just pushing all of the data into the cloud would not work profitably. The archive storage costs might be lowered by using a progressively growing archive for the calculation instead of 200 PB from the start, as the archive is constantly growing. However, since the archive does not start at 0 PB since the archive from the previous cluster does not simply get thrown away, the cost saving of this change in the calculation would be pretty low. Additionally, the archive only makes up the smaller amount of the costs, with the higher costs being the HDD system. The HDD file system of Mistral has a much higher fluctuation in data but is also pretty full most of the time, so there is not much potential for cost savings when looking only at the used storage. So, all in all, even with these changes, cloud storage can not compete with the storage costs of a local data centre like Mistral.

8.3 Future Work

With Mistral currently being replaced and the successor being available in December 2021, the same analysis should be done with the new cluster whenever enough data is available. As this analysis was done with a whole year of data, this would mean waiting at least until the end of 2022 to gain comparable data to Mistral. Mistral's successor is called "Levante" and features much more modern hardware with two AMD EPYC CPUs of the third generation per node [DKRa]. The nodes with standard memory also feature four times the amount of Mistral's nodes, with 256 GB, while nodes with 512 and 1024 GB are available too.

In total, this new cluster will feature 2,878 nodes, so slightly fewer nodes than Mistral, but with every node featuring two 64-core AMD EPYC 7763 CPUs, these nodes have vastly better performance. Sixty of these nodes also feature 4 NVIDIA A100 GPUs each and 512 GB of memory, so nodes are available even for GPU-heavy and visualisation tasks. Two thousand five hundred of the total nodes will be the standard nodes with 256 GB of memory, with 300 nodes with 512 GB and 18 nodes with 1024 GB of memory remaining. The HDD file system and network are also being upgraded to 120 PB of storage and 200 GBit/s, respectively.

The purchase costs of Levante are 32.5 million euros [DKRc], so using this and the number of nodes, it is again possible to get the costs of a single node with the following formulas using similar running costs to Mistral:

$$\begin{aligned} yearlyClusterCost &= serverCost/years + powerCost + rent + personnelCost \\ &= 32,500,000€/6 + 1,500,000€ + 250,000€ + 0€ \\ &\approx 7,200,000€ \\ nodeCost &= (yearlyClusterCost/(days \cdot hours))/nodeNumber \\ &= (7,200,000€/(365.25 \cdot 24))/2,878 \\ &\approx 0.285€ \end{aligned}$$

With about 0.29€ per node hour, Levante is only marginally more expensive than Mistral, with nodes that are several times faster, have four times the amount of memory, and four times the network bandwidth. The CPUs themselves are even more than four times faster, with Levante having a peak performance of 16 PetaFLOPS [DKRc], while Mistral only has 3.59 PetaFLOPS featuring even more nodes. To replace these nodes with cloud nodes, Azure provides pretty similar nodes, also using third-generation AMD EPYC CPUs. While the Azure HBv3 nodes have the same architecture, 200 GBit/s Infiniband, and 448 GB of memory, they only provide 120 instead of 128 cores, compared to the Levante nodes.

While the hardware is comparable to the Levante nodes, even the cheapest spot instances in "East US" are four times as expensive with 1.21€ per node hour. So with the full nodes being that much more expensive, currently, most likely, only the smaller single-node jobs would be an option for outsourcing, but this would have to be analysed further.

Another aspect of what can be further optimised in the cost function is representing the performance of different nodes. Like already mentioned with the dual-node jobs earlier, this could be used to have a smaller number of nodes replacing the original number of nodes, like one faster node being used instead of two slower nodes. Also, with the same node amount as the original job, a faster node could result in a shorter execution time, therefore reducing the node hours and the cost. Additionally, a paper by the University of Berkeley has proven that the cloud nodes can actually have better performance than a cluster, depending on the application and the hardware [Gui+21]. They also concluded that the large hardware variety and frequent upgrades to newer hardware are a significant advantage of the cloud, so benchmarks of the specific applications would be even better for comparison.

To represent this in the cost function, the original cluster's nodes and all cloud nodes would have to be benchmarked for comparison. The difference in performance would then have to be multiplied by the number of node hours to represent the actual execution times. The only difficulty with these benchmarks would be that every application behaves differently with hardware differences. So while one application might perform very similarly on the cloud node, another might perform much better. To mitigate this, a mix of different benchmarks would have to be used, or the most used applications of the cluster should be run as a benchmark for comparison.

Another issue with the current calculations of the cost function is that the 20% of high memory jobs is only an estimate and was not analysed further. The current dataset used for this thesis does not contain any information on memory usage, so an estimate had to be chosen. So in a deeper analysis, the jobs would have to be analysed on how much memory they are actually using to get a more accurate percentage of high memory jobs.

Storage also remains relevant, as the cloud instances also need the job data to be lying in cloud storage to have access to them. While the storage is only needed for the job duration, the data still has to be transferred to and from the cloud, which adds additional costs that have to be calculated for a deeper analysis. The storage performance is also not negligible, with Amazon S3 showing in benchmarks for HPC quite bad performance compared to a Lustre file system [GK21]. While block storage performance in the cloud might be better, this is another factor to be analysed further.

While the cost function for Mistral only looks at the theoretical cost savings and not at the expenses needed to outsource the computing power, this is another essential aspect that needs to be analysed. The job scheduler would then need to be extended using a plugin or something similar to distribute the jobs between the cluster and the cloud, as otherwise, the scheduler would have no way to use the cloud nodes. While this might potentially be a one-time cost since once the scheduler is set up, it just needs to be maintained, so in the long-term, the calculated cost savings might be more accurate. However, to get to this point, there might be several things the staff has to learn about the cloud systems to be able to utilise the cloud to its full potential.

Additionally, as only Mistral and a brief look on Levante have been done so far, the cost function could also be used on the data of multiple different clusters to show how the cloud would perform against many different workloads. With this thesis only using the data of Mistral and one fictional scenario, other real scenarios might show even more use-cases for cloud instances in HPC systems.

Bibliography

[Azua] Microsoft Azure. *Azure Blob Storage*. URL: https://azure.microsoft.com/de-de/services/storage/blobs/ (visited on 06/14/2021).

[Azub] Microsoft Azure. *Azure Virtual Machines*. URL: https://azure.microsoft.com/de-de/services/virtual-machines/ (visited on 06/14/2021).

[Azuc] Microsoft Azure. *Compute benchmark scores for Linux VMs*. URL: https://docs.microsoft.com/en-us/azure/virtual-machines/linux/compute-benchmark-scores (visited on 10/03/2021).

[CHS11] Adam Carlyle, Stephen Harrell, and Preston Smith. "Cost-Effective HPC: The Community or the Cloud?" In: Jan. 2011, pp. 169–176. DOI: 10.1109/CloudCom.2010.115.

[Cloa] Google Cloud. *Cloud Storage*. URL: https://cloud.google.com/storage (visited on 06/14/2021).

[Clob] Google Cloud. *Compute Engine*. URL: https://cloud.google.com/compute (visited on 06/14/2021).

[DKRa] DKRZ. *Contract Signing Levante*. URL: https://doc.dkrz.de/doc/levante/ (visited on 10/08/2021).

[DKRb] DKRZ. *HLRE-3 "Mistral"*. URL: https://www.dkrz.de/en/systems/hpc (visited on 05/31/2021).

[DKRc] DKRZ. *Introduction Levante*. URL: https://www.dkrz.de/pdfs/presse-und-artikel/pm-offizielle-vertragsunterzeichnung-fur-hlre-4 (visited on 10/08/2021).

[Geo+19] Gareth George et al. "Analyzing AWS Spot Instance Pricing". In: *2019 IEEE International Conference on Cloud Engineering (IC2E)*. 2019, pp. 222–228. DOI: 10.1109/IC2E.2019.00036.

[GK21] Frank Gadban and Julian Kunkel. "Analyzing the Performance of the S3 Object Storage API for HPC Workloads". In: *Applied Sciences* 11 (Sept. 2021), p. 8540. DOI: 10.3390/app11188540.

[Gmb21] Deutsches Klimarechenzentrum GmbH. "Jahresabschluss zum Geschäftsjahr vom 01.01.2019 bis zum 31.12.2019". In: Jan. 2021. URL: https://www.bundesanzeiger.de/.

[Gui+21] Giulia Guidi et al. “10 Years Later: Cloud Computing is Closing the Performance Gap”. In: *Companion of the ACM/SPEC International Conference on Performance Engineering*. ICPE ’21. Virtual Event, France: Association for Computing Machinery, 2021, pp. 41–48. ISBN: 9781450383318. DOI: 10.1145/3447545.3451183. URL: https://doi.org/10.1145/3447545.3451183.

[HAL14] Rashid Hassani, Md Aiatullah, and Peter Luksch. “Improving HPC Application Performance in Public Cloud”. In: *IERI Procedia* 10 (Dec. 2014), pp. 169–176. DOI: 10.1016/j.ieri.2014.09.072.

[JLZ20] Qingye Jiang, Young Choon Lee, and Albert Y. Zomaya. “The Power of ARM64 in Public Clouds”. In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. 2020, pp. 459–468. DOI: 10.1109/CCGrid49817.2020.00-47.

[pan] pandas. *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/ (visited on 09/12/2021).

[PC] Caitlin Potratz Metcalf and Peter Church. *U.S. CLOUD Act and GDPR – Is the cloud still safe?* URL: https://www.linklaters.com/en/insights/blogs/digilinks/2019/september/us-cloud-act-and-gdpr-is-the-cloud-still-safe (visited on 09/24/2021).

[Rol+12] Eduardo Roloff et al. “High Performance Computing in the cloud: Deployment, performance and cost efficiency”. In: Dec. 2012, pp. 371–378. ISBN: 978-1-4673-4511-8. DOI: 10.1109/CloudCom.2012.6427549.

[Sera] Amazon Web Services. *Amazon EC2*. URL: https://aws.amazon.com/de/ec2/ (visited on 06/14/2021).

[Serb] Amazon Web Services. *Amazon S3*. URL: https://aws.amazon.com/de/s3/ (visited on 06/14/2021).

[Serc] Amazon Web Services. *AWS Graviton-Prozessor*. URL: https://aws.amazon.com/de/ec2/graviton/ (visited on 10/04/2021).

[Serd] Amazon Web Services. *EC2 spot instance rebalance recommendations*. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/rebalance-recommendations.html (visited on 10/18/2021).

[Sere] Amazon Web Services. *New Amazon EC2 Spot pricing model: Simplified purchasing without bidding and fewer interruptions*. URL: https://aws.amazon.com/de/blogs/compute/new-amazon-ec2-spot-pricing/ (visited on 09/17/2021).

[Sha+20] Leah Shalev et al. “A Cloud-Optimized Transport Protocol for Elastic and Scalable HPC”. In: *IEEE Micro* 40.6 (2020), pp. 67–73. DOI: 10.1109/MM.2020.3016891.

[Smi+19] Preston Smith et al. “Community Clusters or the Cloud: Continuing cost assessment of on-premises and cloud HPC in Higher Education”. In: July 2019, pp. 1–4. ISBN: 978-1-4503-7227-5. DOI: 10.1145/3332186.3333155.

[Sta] Statista. *Amazon Leads 130-Billion Cloud Market*. URL: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ (visited on 06/14/2021).

[Wik] Wikipedia. *CLOUD Act*. URL: https://de.wikipedia.org/wiki/CLOUD_Act (visited on 09/24/2021).

[ZDN] ZDNet. *Microsoft is designing its own Arm chips for datacenter servers: Report*. URL: https://www.zdnet.com/article/microsoft-is-designing-its-own-arm-chips-for-datacenter-servers-report/ (visited on 06/14/2021).

Appendices

Abbreviations

AWS Amazon Web Services

CLOUD Act Clarifying Lawful Overseas Use of Data Act

CPU Central Processing Unit

EC2 AWS Elastic Compute Cloud

EFA Elastic Fabric Adapter

GDPR General Data Protection Regulation

HPC High Performance Computing

S3 AWS Simple Storage Service

List of Figures

List of Tables

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Wirtschaftsinformatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum	Unterschrift

Veröffentlichung

Ich bin damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik eingestellt wird.

Ort, Datum	Unterschrift