Utrecht University

Thesis Report
Master Business Informatics

Task-Aware Cloud Instance Selection

The right cloud instance for the task.

Author
Wiert Omta
Student number
3803279

Supervisor
Marco Spruit
Second supervisor
Slinger Jansen
Company supervisor
Wienand Omta

December 26, 2019

Abstract

Commercially available cloud computing offers a viable alternative to grid computing. Researchers and organizations can utilize cloud computing to process analytical tasks without the need to invest in hardware. When the number of tasks rises, having automated means of deploying these tasks to a commercial cloud, huge cost savings and availability benefits can be reaped. This research sets out to devise a decision mechanism to deploy analytical tasks to commercial cloud offerings in a cost-efficient way. This is done by a literature review, analysis of the portfolios of cloud providers and an experiment where cloud instances are benchmarked. This data is incorporated in a decision tree, supporting the cloud instance decision.

Keywords: cloud computing, data analytics, virtual machines, cloud selection, virtualization

Contents

1 Introduction

Commercial cloud companies offer a wide variety of cloud-based computing solutions. Each cloud provider offers a number of similar products that can be configured in numerous ways. Cloud users, clients of cloud providers, can be overwhelmed by the sheer number of virtual machines (VMs) that, for instance Amazon Web Services (AWS), has on offer.

All major cloud providers offer virtual machines that are optimized for specific use cases and virtual machines that are intended for general purposes. These virtual machines differ in terms of processor type, available RAM, storage and other hardware related specifications. AWS offers memory-optimized, storage optimized and processor optimized instances, as well as machines that combine these optimizations (Bao, Damon, Landman, & Gokhale, 2016). Other cloud providers offer VMs with similar optimizations.

The variations mentioned above are technical. Next to the technical differences there are also non-technical differences between cloud offerings that are not related to a specific virtual machine. All instances can be purchased in various pricing and availability models (Song & Guerin, 2017). Most cloud providers offer on-demand availability where a user can purchase cloud computing ad-hoc. Users can also reserve instances for a longer period, often at a discount. Because of the highly virtualized nature of cloud computing, it is possible to make capacity available to a client in a matter of seconds (Calcavecchia, Biran, Hadad, & Moatti, 2012). This creates a possibility for demand-based pricing models where availability is a result of supply and demand (Rimal, Choi, & Lumb, 2009). Cloud providers use mechanisms like these to offer instances that are currently not used by the clients paying for them. These instances are offered with lower reliability, but for a much lower price (Agmon Ben-Yehuda, Ben-Yehuda, Schuster, & Tsafrir, 2013). This enables cloud providers to achieve a higher utilization rate of their machines.

1.1 Task type

This research project focuses on a specific category of tasks that are executed by organizations and research facilities. More specifically this research project was inspired by a data analytics company struggling to match analytical tasks to the right cloud instance on the Amazon EC2 cloud. Successful organizations utilize machine learning, deep learning, data analytics, big data analytics and more (Chen, Chiang, & Storey, 2012). Business need to execute a wide variety of tasks as quick as possible to stay ahead of the competition.

Data storage and computation have gotten very affordable in the last decade (Tekiner & Keane, 2013). Companies accumulate vast quantities of data, too large to process with a single computer. These quantities of data are called big data. These data are accumulated in an automated fashion and are often semi-structured or unstructured. To be able to process big data, new techniques are needed (Oussous, Benjelloun, Lahcen, & Belfkih, 2018). Frameworks like MapReduce and Hadoop enable users to analyse big data using commodity hardware with fast network connections. These frameworks are effective as the bring processing capacity to the data, as opposed to the old paradigm: bringing data to the processing capacity (Oussous et al., 2018). The latter resulting in heavy data traffic.

Big data analytics packages like Hadoop, rely on utilizing distributed computing. More specifically distributed commodity hardware (Cohen, Dolan, Dunlap, Hellerstein, & Welton, 2009). This makes cloud computing very suitable for frameworks like Hadoop. Just like big data analytics frameworks, also more traditional packages like R can benefit from using distributed commodity hardware. Companies utilize cloud computing to process more tasks in a shorter period of time, increasing their competitive advantage. Organizations often use frameworks like Hadoop in order to handle the data volume, once the data is summarized and cleansed they use traditional tools to further analyze the data (Özcan et al., 2011).

Next to the aforementioned tasks, also new developments like bitcoin mining can benefit from cloud computing. These tasks have very specific hardware requirements and therefore are not a focus area of this research project. This research project focuses on all analytical tasks that can be executed in the cloud. These tasks can be straight-forward analytical tasks, as the ones executed in the benchmarking experiment in Chapter 4, but also distributed tasks as generated by big data frameworks like Hadoop and MapReduce.

1.2 Task contextualization

As stated by Azvine, Cui, Nauck, and Majeed (2006), running data analytics tasks has always been part of running a competitive business. Successful organizations therefore need to run many tasks to stay competitive. Organizations have to manage and schedule these tasks to run at the right moment. Although task scheduling and distributed computing have been around for decades (Malone, Fikes, & Howard, 1983), users still lack guidance in choosing between all cloud offerings. While cloud computing brought untethered access to computing resources, many of the existing workflow and scheduling algorithms, assume the number of resources to be bounded (Bessai, Youcef, Oulamara, Godart, & Nurcan, 2012).

This characteristic of cloud computing, the illusion of limitless resources (Bessai et al., 2012), introduces organizations to new problems. How to handle the requests for processing of tasks, how to keep track of cloud offerings in use, how to deploy tasks to the right cloud instance or offering. This research project will focus on the last problem.

(Buyya, Pandey, & Vecchiola, 2009) try and solve this problem by introducing a cloud market were users make indirect and direct resource requests. Indirect resource requests are made via software that users use, while direct requests are made directly at the cloud provider. They describe the Cloudbus Toolkit with a broker and a workflow management system. The broker maps analysis tasks to compute resources and can provision to available on-premise, private and public clouds. The Cloudbus Toolkit also provides a Workflow Management System (WMS) that aids the user by enabling to represent applications as a workflow. A WMS manages information about the tasks in the workflow, such as starting and ending timestamps, deadlines, the lifespan (or makespan) (Combi & Pozzi, 2006).

In the light of this research project, organizations need a workflow management system to manage the tasks that are up for execution. While workflow management systems are suitable for managing and keeping track of all tasks that need to be executed within the organization (Fern, Tedeschi, Priol, et al., 2011), the tasks still need to be mapped to the right cloud resource or offering. This project sets out to identify the task requirements and proposes a way of handling task cloud instance selection.

1.3 Research problem and goal

Cloud computing, especially the unutilized cloud instances, can be interesting to decrease operational cost, especially in the field of many task computing (Ghafarian & Javadi, 2015). Many task computing is bridging the gap between high throughput computing and high performance computing and is using a large number of computing resources to accomplish many computational tasks (Raicu, Foster, & Zhao, 2008). The metrics concerned with many task computing are seconds, tasks per second, I/O speed and others. This research sets out to identify the requirements of these tasks in order to match them to a suitable cloud instance. These tasks can have technical requirements in terms of needed resources, like memory, processing power, graphics processing power, data throughput, and latency. These tasks also have business requirements, like budget, deadlines and reliability (Javanmardi et al., 2014a). These requirements vary per organization and task. While this research project focuses on general analytical tasks performed using R, the findings related to cloud selection and cloud performance can be applied to other types of cloud data analytics.

While it is possible to pair tasks to the right cloud offering manually by selecting a server and deploying the task, this gets incrementally harder when the number of tasks rises. Especially when tasks are being created by users that do not know the specific technical requirements. Businesses that use cloud computing, need an efficient way to pair cloud tasks to the right cloud instance (Farshidi, Jansen, de Jong, & Brinkkemper, 2018). While there are cloud brokers, businesses that help pair tasks to the right cloud, these often do not take all requirements into account (Ferrer et al., 2012).

To be able to pair tasks as optimal as possible one has to evaluate both business and technical requirements. There have been some advances in both fields, (Alnemr, Pearson, Leenes, & Mhungu, 2014) have created a Cloud Offerings Advisory Tool that considers non-functional requirements. (Ferry, Song, Rossini, Chauvel, & Solberg, 2014) have created a Cloud Modeling Framework (CloudMF), that makes it possible to create a Cloud Provider Independent Model that can be used to map a technical cloud model to a cloud offering of a cloud provider.

This research aims to create a decision tree that guides users selecting cloud instances for a cloud task. In this selection both business and technical requirements are taken into account. For a schematic overview of the cloud instance selection process, see Figure 1. In this research, we will refer to businesses and institutions that use cloud computing as cloud clients. Businesses that offer cloud services will be referred to as cloud providers. The decision tree enables cloud clients to pair a cloud task to the most optimal cloud instance of a certain cloud provider. We will mainly focus on Amazon Web Services since that is currently the largest cloud provider offering the widest variety of services.

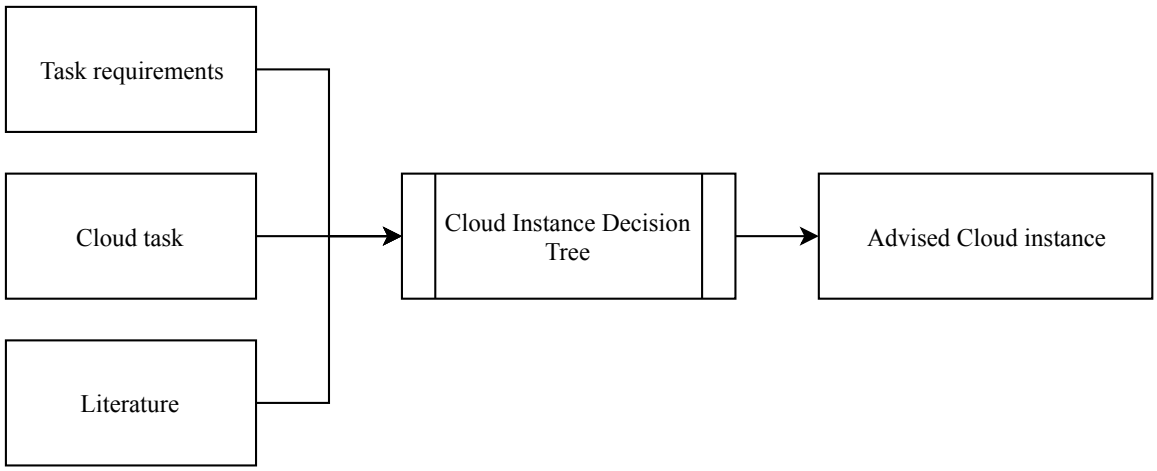


Figure 1: Schematic overview of the cloud decision framework

1.4 Problem relevance

Companies and research institutions are increasingly using cloud computing to run tasks that previously used to be complex and tedious. With the introduction of easy-to-use open-source frameworks like MapReduce and Spark it became easier to use commodity hardware for data analytics (Shi et al., 2015a). Traditional grid-based computing environments are expensive and hard to scale (Srirama, Batrashev, & Vainikko, 2010). Cloud computing delivers a viable alternative and practically limitless scaling possibilities.

The introduction of new pricing models, like spot pricing, offer a cost-efficient solution (Kaulakienė, Thomsen, Pedersen, Çetintemel, & Kraska, 2015). When researchers and companies successfully circumvent the limitations of these models, cloud based analytics can be executed at a competitive price.

1.5 Research contribution

This research projects provides two scientific contributions: a literature review about existing literature and a cloud instance decision tree that aides users in selecting a cloud instance for a specific task. This research combines existing literature to create a new artefact in the form of a decision tree.

1.6 Research questions

The research problem is described in the main research question (RQ) that is formulated as follows:

RQ: How can a cloud computing instance be selected to perform analytical tasks, taking into account technical and non-technical requirements?

To answer the main research question, sub questions (SQs) have been formulated. Each subquestion answers a part of the main research question. To be able to describe

the tasks that are processed using cloud computing we need to describe the tasks. The description of cloud tasks will be addressed in SQ1:

SQ1: How can the requirements and characteristics of a cloud task be described?

Furthermore, we need to describe the cloud instance itself, to be able to match the cloud tasks to the cloud instances. This will be addressed in SQ2:

SQ2: How can cloud instances be described?

Since we now have a way to describe the cloud tasks and the cloud itself, we need a matching-mechanism to match the analytical tasks to the most optimal cloud instance. The matching of tasks and instances is addressed in SQ3:

SQ3: How can tasks and cloud instances be matched?

1.7 Document structure

This chapter (Chapter 1) focuses on introducing the topic and the scientific challenges. It also introduces the research questions. The next chapter, Chapter 2 explains the research approach and research method that has been used to find answers to the research questions. The third chapter reports the results of the literature study that was conducted as a part of the research approach. Chapter 4 explains the set-up of the cloud configuration performance benchmark. Chapter 5 presents the results of the experiments that were run on the cloud environments. Chapter 6 contains the the cloud instance decision tree. The discussion and future work are presented in Chapter 7. Chapter 8 presents the conclusion and answers the research question and sub-questions.

2 Research process

Chapter 1 introduces the problem and the research questions. In this chapter the steps of this research project will be discussed. The first step will be conducting a literature review, the problem investigation step of the design cycle described by Wieringa (2014). The next step will be creating the experiment in order to design a decision tree and the final step will be executing the experiment to validate the outcomes of the decision tree. These are the treatment design and treatment validation as described by Wieringa (2014).

This research tries to identify requirements involved in the deployment of software to a cloud environment. The research method to conduct this research will be explained in this chapter. The steps in the research method have been modelled using a Process Deliverable Diagram (PDD). This is a technique created by van de Weerd and Brinkkemper (2009) and it lists the activities in the method on the left side. The right side of the diagram shows the deliverables. In the remainder of this chapter parts of the PDD are shown to explain the research method. Note that not all relations are shown in these parts of the PDD due to the available space, for a complete overview see the PDD in Appendix A.

2.1 Preliminary Literature Study

The first part of this research project was conducting a preliminary literature review to get acquainted with the topic and to identify the current state of research on the topic. The preliminary literature study was conducted on literature gathered via Google Scholar. Next to the scientific literature that was found via Google Scholar, also the websites of various cloud providers were studied. The cloud providers studied in the preliminary literature review are Amazon Web Services and Microsoft Azure. The results of the preliminary literature review are incorporated in the research proposal.

2.2 Research method

This chapter will explain the steps involved in this research project. An overview of all steps will be given, as well as the details of each step. According to Von Alan, March, Park, and Ram (2004), Information Science (IS) research can be divided in two disciplines. Design science and behavioral science. The first is the method this research project will be using, the latter is the discipline that tries to explain and predict business and human phenomena relevant for analysis, design, implementation and the use of an information system.

Von Alan et al. (2004) describe seven guidelines to ensure the quality of a design science research project. The guidelines are as follows:

1. **Design as an artifact**
   Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.

   The goal of this research project is to create a cloud decision framework that aides the user to find the right cloud offering for a certain task. This decision framework can be seen as an artifact.

2. **Problem relevance**
   The objective of design-science research is to develop technology-based solutions to important and relevant business problems.

   As discussed in Chapter 1.2 the problem in this research, cloud instance selection, is a relevant problem for companies and research institutions. With the introduction of frameworks like MapReduce it became easier to use commodity hardware for data analytics (Shi et al., 2015a). Traditional grid-based computing environments are expensive and hard to scale (Srirama et al., 2010). Cloud computing delivers a viable alternative and practically limitless scaling possibilities. The introduction of new pricing models, like spot pricing, offer a cost-efficient solution (Kaulakienė et al., 2015). When researchers and companies successfully circumvent the limitations of these models, cloud based analytics can be executed at a competitive price.

3. **Design evaluation**
   The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed validation methods.

   The design artifact that will be created in this study will be evaluated as a part of the research process. The benefits of the artifact will be benchmarked and compared to a baseline server to show the effectiveness of the cloud decision framework.

4. **Research contribution**
   Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

   As discussed in Chapter 1.3, this research provides two scientific contributions: a literature review about existing literature and a cloud instance decision tree that aides users in selecting a cloud instance for a specific task.

5. **Research rigor**
   Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.

   The research method shows the complete process that will be used to conduct this research. It provides detailed information of the methods that will be used. For a complete overview of the research method see Appendix A.

6. **Design as a search process**
   The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

   The construction of the artifact will be based on literature study, expert opinions and a benchmark of the artifact itself.

7. **Communication of research**
   Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

   The results of this project will be presented in this thesis document as well as

in several presentations. Both technology-oriented and management-oriented audiences will be provided a clear overview of the results of this research project.

Since this research project is not explaining or predicting the effects of automated cloud deployment, but rather finding a way to automate cloud deployment, this research project will be using a design science approach. Design Science is explained by Wieringa (2014) as a method that iterates over two activities: designing an artifact that improves something for stakeholders and empirically investigating the performance of an artifact in a context. A design cycle is described by Wieringa (2014) as a process with three phases: problem investigation, treatment design and treatment validation. The design science research model for this specific research project is depicted in Figure 2.

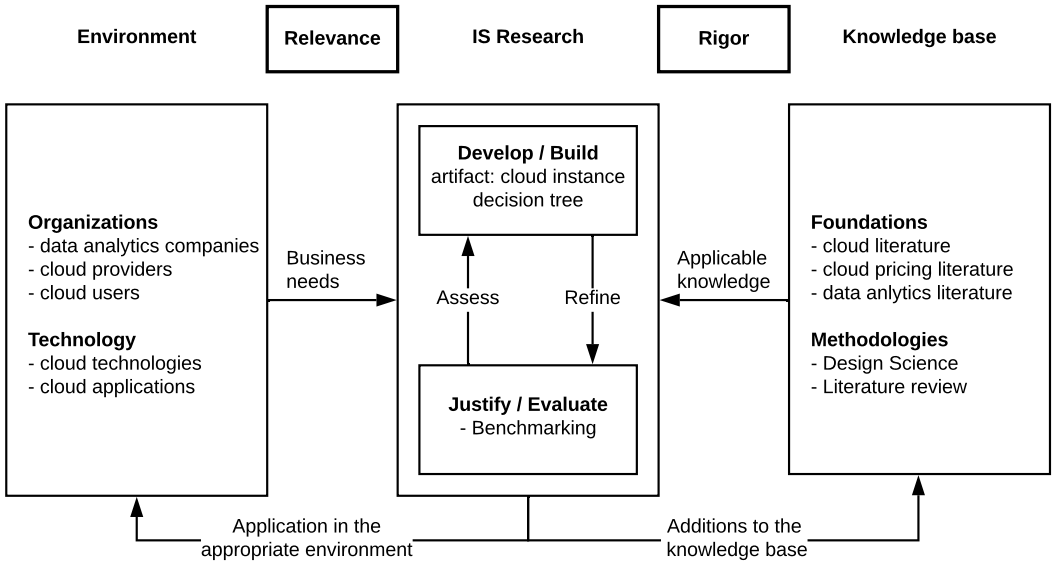


Figure 2: Design Science Research Framework

2.3 Literature review

In this research a structured literature review (SLR) is conducted. (Kitchenham, 2004) define a SLR as *"A systematic review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest."*. The literature study should be reproducible and the protocol should be developed in advance of conducting the review (Okoli & Schabram, 2009).

The literature review was split into two parts. A preparation phase where all necessary prerequisites where taken care of and the review execution where the actual review was conducted.

The search queries will be executed on Google Scholar using a proxy provided by Utrecht University. This search engine uses a number of scientific databases. Only the results on the first ten pages were considered and added to the literature list.

2.3.1 Literature review preparation

To be able to execute the literature review, it has to be prepared first. The preparation phase makes sure all things necessary for the literature review are available.

The first step of the research method that is shown in the first box of the PDD is the literature review preparation phase. The literature review will be conducted to collect and review all available literature on the subject. The review consists of multiple steps and is divided into two parts in the PDD, the preparation phase and the execution phase. The first part, the preparation phase, is shown in Figure 3. The preparation phase contains two steps.

Step one is to create search queries that will be used to find the literature about the topic. This is the first step in the PDD, "create search queries". This step results in a list of search queries, which is shown in the PDD as the SEARCH QUERY deliverable. These queries will be used to find relevant literature about the subject of this research project. The search queries will be based on the findings of the preliminary literature review.

Since a literature review gives a lot of results it is important to be able to filter the relevant findings from the less relevant findings. This will be done by applying criteria. These criteria describe whether a certain finding has to be kept or should be removed from the total list of findings.

Inclusion criteria describe when a piece of literature should be kept. Exclusion criteria describe when a piece of literature should be removed. The creation of the inclusion and exclusion criteria is shown in the second step of the PDD: "Create inclusion and exclusion criteria". This activity has one deliverable, a list of criteria. In the PDD is shown that there are two types of criteria. The criteria will be based on the findings of the preliminary literature review.

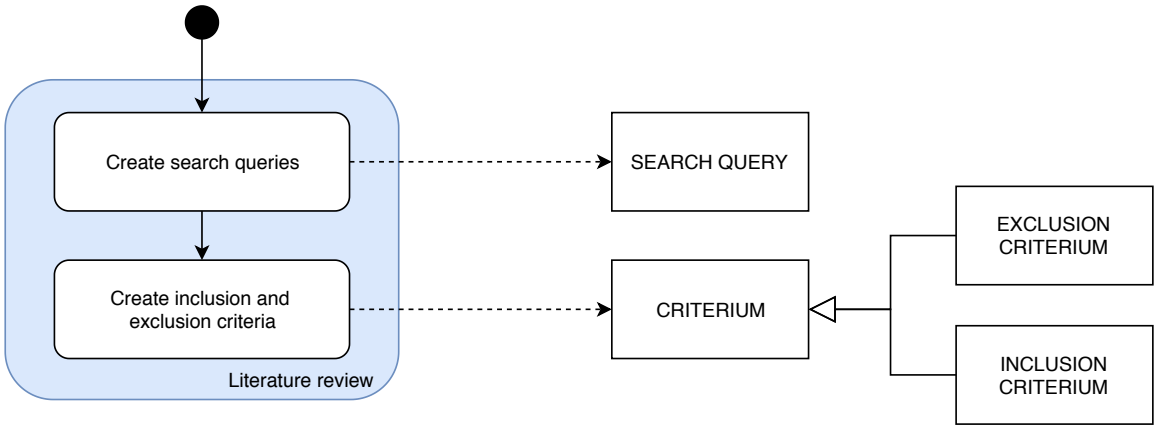


Figure 3: Part 1 of the PDD: the literature review

The initial selection is based on the title and description in Google Scholar.

SQ1	**SQ2**	**SQ3**
How can the requirements and characteristics of a cloud task be described?	*How can cloud instances be described?*	*How can tasks and cloud instances be matched?*
data analytics jobs	cloud modeling AND computing	cloud AND deployment AND automated
data analytics task	cloud environments modeling	cloud scheduling
data analytics hardware requirements	cloud instance types	cloud computing selection
data analytics cloud	cloud requirements	cloud environment selection
scientific tasks cloud	cloud types AND computing	cloud AND deployment AND strategy
grid computing jobs		cloud AND deployment AND automated
grid computing hardware requirements		cloud AND spot pricing
grid job types		cloud AND requirements
		grid computing AND job types
		cloud AND reliability
		cloud scheduling
		provisioning techniques
		software architecture quality metrics
		amazon spot instances
		cloud modeling AND computing

Table 1: Search queries used in the SLR (all queries were between quotes)

2.3.2 Literature review execution

After the literature review is thoroughly prepared it can be executed. In this phase all the input of the previous phase will be used to perform multiple queries on various search engines. This part of the research method is depicted in the second blue box of the PDD which can be found in Figure 4.

2.3.3 Execute search queries

The first step of the literature review execution (Execute queries in the PDD) is to execute the search queries. This is the first rounded box in the PDD in Figure 4. In this step the results of each query will be stored and saved on a list, LIST OF ARTICLES in the PDD. The search queries are listed in Table 2.3.1. This will result in a large list of articles that will have relevant data that can be analyzed.

2.3.4 Apply inclusion and exclusion criteria

The list of articles that is created in the first step of the literature review execution is a large list that also contains irrelevant articles. It is important that these results are refined. This makes the analyzing step more efficient since less articles have to be analyzed. The list of articles will be refined by applying the inclusion and exclusion criteria. The results will be analyzed using NVivo, qualitative data analysis software. By scanning the abstract of each entry the criteria will be applied.

Inclusion criteria

- Peer-review papers
- Studies are written in English

Exclusion criteria

- Patents will be excluded
- Studies not related to the research questions

2.3.5 Analyze articles

After applying the criteria the remainder of the literature will be read in-depth and coded using NVivo. NVivo is qualitative data analysis software that can be used to analyze articles. This is done by applying a technique called coding (Bazeley & Jackson, 2013). By coding an article all relevant material will be labeled and indexed. In this research project NVivo will be used to code all cloud requirements. These requirements will then be placed on a requirement list. In this research we differ between business and technical requirements.

The business requirements describe the business needs in a cloud deployment setting. Business requirements are all requirements that are not technical requirements. We define technical requirements as the requirements that the server hardware has to support in order to complete the job.

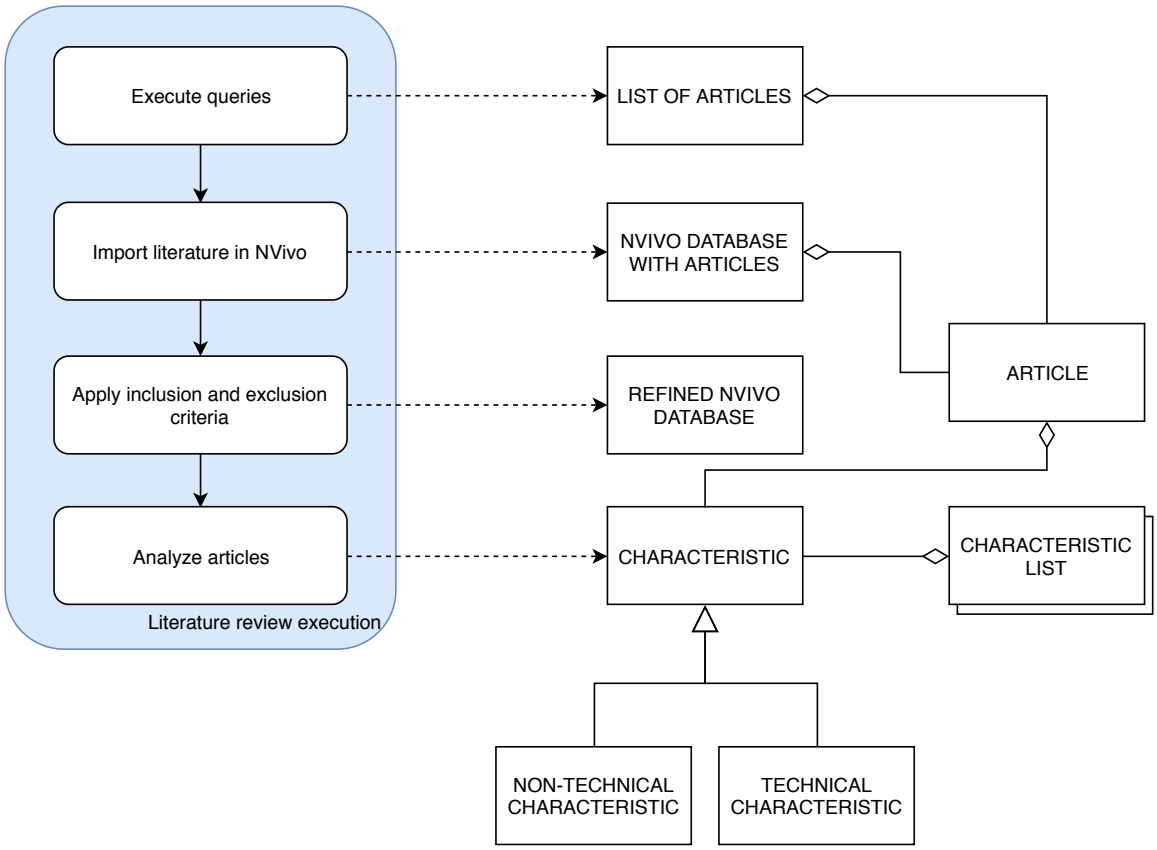


Figure 4: Part 2 of the PDD: the literature review execution

2.4 Experiment set-up

The third part of the research method is the experiment set-up. An overview of this part can be found in Figure 5. In this step the benchmark will be created that will be runned on the Amazon virtual machines. With the data the benchmark yields, a decision tree will be devised. The benchmark results will be compared to the baseline server. The benchmark is a task that is representative for a variety of tasks that can be executed in the cloud. These task will be executed on a baseline server. The performance of the baseline server will be scored on the requirements found in the literature study, as well as on the following aspects:

- Cost: the cost to perform the benchmarks.
- Makespan: the time needed to perform the benchmarks.

The test tasks will then be executed on servers in different configurations. The performance on these servers will be scored on the same aspects. When the benchmarks of both server set-ups are completed, the scores can be compared.

2.4.1 Price data collection

The first step is to collect price data of cloud providers. Since the performance of the decision framework will also be based on cost, comprehensive pricing data of Amazon Web Services is needed. A large part of the offerings are dynamically priced via the spot pricing mechanism. Since these prices are subject to constant change it is necessary to collect this price data over a period of time. To collect this price data an EC2 instance is set-up which writes the price data to a comma separated value file each day.

2.4.2 Devise server profiles

In the second step of the validation set-up server profiles are created. These server profiles are the outcomes of the decision framework. Based on the input of the decision framework, one of the server profiles on the SERVER TYPE LIST will be recommended. The outcome is based on the input given on the requirements found in the literature execution phase.

These server profiles will be based on the server families as stated by Amazon Web Services.

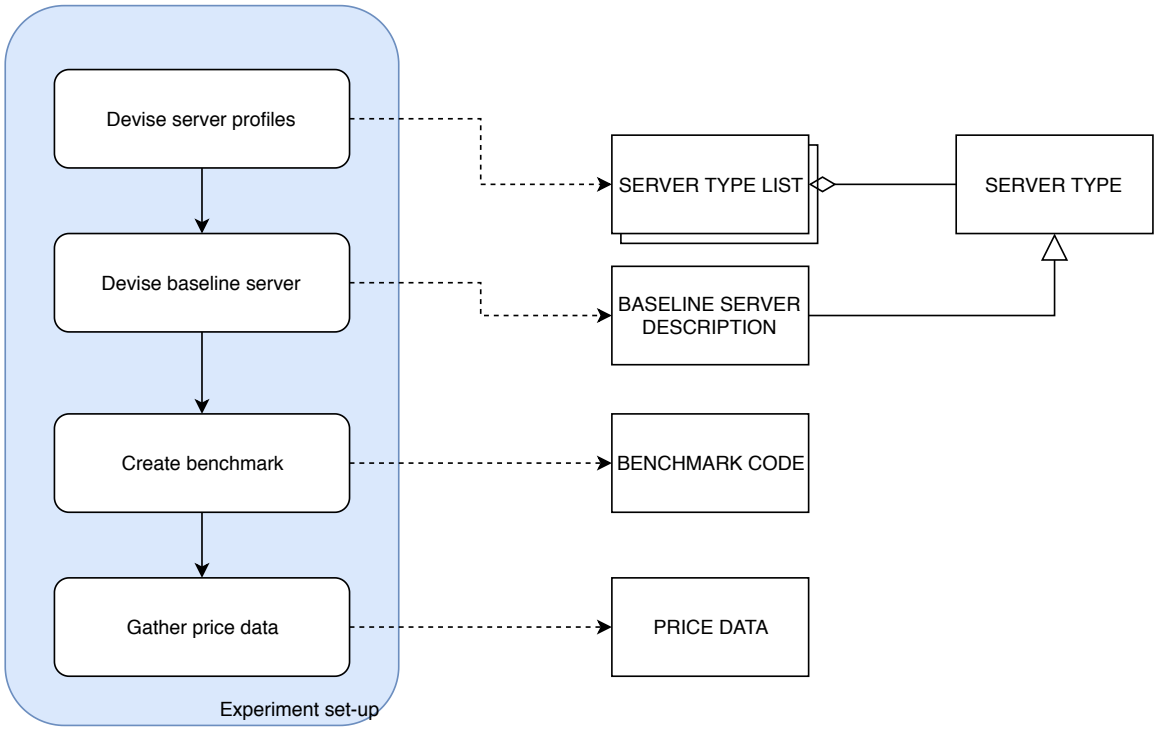


Figure 5: Part 3 of the PDD: the experiment set-up

2.5 Experiment execution

The last part of the research method is to execute the experiment and create the decision tree. The test tasks will be run on the baseline server to get a benchmark score, which consists of various metrics. The benchmark will run on multi core and multi node configured servers. The benchmark results will then be used to create the decision tree.

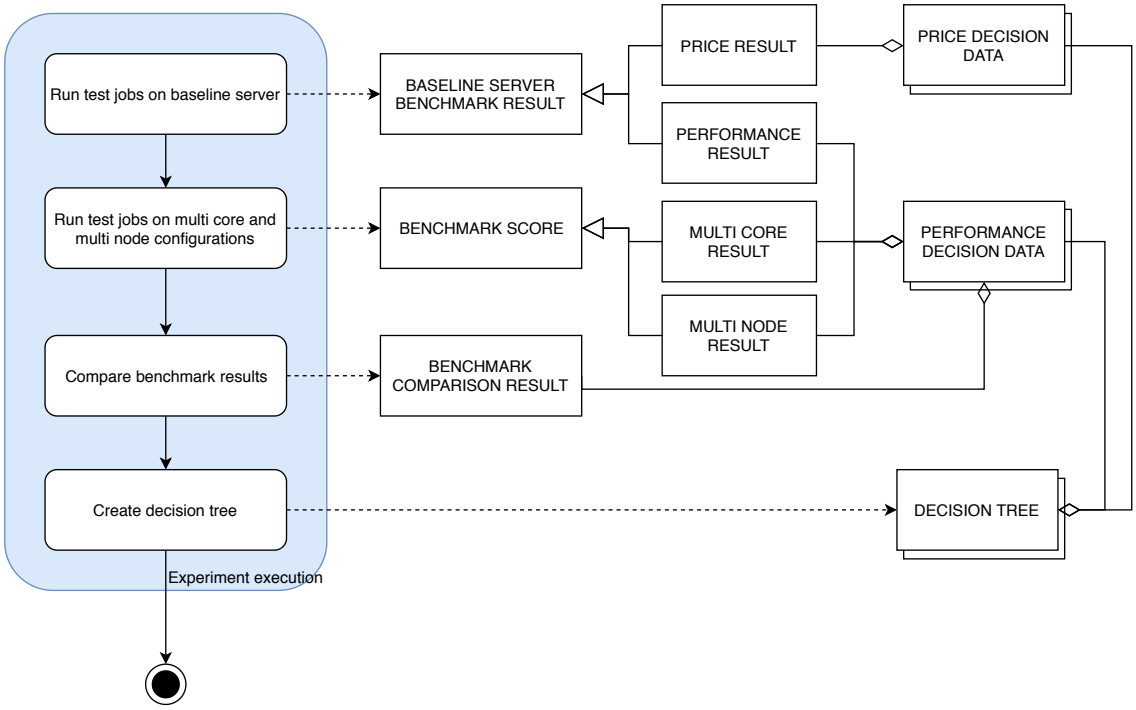


Figure 6: Part 4 of the PDD: the experiment execution

2.5.1 Create decision tree

When all the data is collected the decision tree can be created. This is the artifact of this design science research project. The decision tree enables the end-user to match tasks to cloud instances based on a statistical analysis. The outcome of the decision tree is a list of cloud instances suitable for the properties of the task.

2.6 Research process

For a complete overview of the research process, see the process-deliverable diagram in Appendix A.

3 Theoretical background

In this chapter the results of the literature study will be discussed, to give a theoretical overview of the research field. A comparative analysis of all the authors will be shown and all relevant papers related to the research questions will be discussed. This chapter is divided into five sub-chapters, one for each research sub-question, a chapter about cloud computing in general and a chapter about the literature categories and how this project is positioned in relation to other literature.

For each sub-question with the corresponding search queries a pipeline was devised to retrieve the relevant results. The results of each pipeline can be found in Table 2.

Research question	**SQ1**	**SQ2**	**SQ3**
Number of results considered	88	54	76
Title and abstract review	42 (- 46)	28 (- 26)	34 (- 42)
Accessible	42 (- 0)	28 (0)	34 (0)
Relevant and used	14	7	8
Snowballing	0	0	1
Used results	14	7	9

Table 2: Literature review pipeline

3.1 Literature positioning

In this section this research project will be categorized in relation to other literature. The literature found in this research project will be divided into buckets that show a specific kind of research related to cloud computing and data analytics. To categorize the literature the abstracts of the used literature were extracted using NVivo. If the literature mentioned keywords in the abstract, those were also taken into account. The topics of all research papers were compared in order to create six main topics to categorize the literature. It is important to mention that some research papers overlap multiple categories. The literature was divided into the following categories:

- Cloud service selection (Table 3)
- Cloud pricing (Table 6)
- Data analytics in the cloud (Table 5)
- Cloud performance measuring (Table 6)
- Distributed systems (Table 7)
- Task scheduling (Table 8)

This research project sets out to create a cloud decision framework based on experiments measuring the performance of cloud instances. This decision framework can be used to select a cloud instance, virtual machine in order to do data analytics in the cloud as cost-effective as possible. The experiment uses simple tests that are executed sequentially to perform the benchmark and therefore this research project can not be categorized as a work about task scheduling. Therefore this research project spans the first topics mentioned in the list above.

Cloud Service Selection
Sundareswaran, Squicciarini, and Lin (2012)
Redl, Breskovic, Brandic, and Dustdar (2012)
Voorsluys and Buyya (2012)
ur Rehman, Hussain, and Hussain (2011)

Table 3: Literature about cloud service selection

Cloud pricing
Agmon Ben-Yehuda et al. (2013)
Mattess, Vecchiola, and Buyya (2010)
Qu, Calheiros, and Buyya (2016)
Javadi, Thulasiram, and Buyya (2013)

Table 4: Literature about cloud pricing

Data analytics
Archenaa and Anita (2015)
Assunção, Calheiros, Bianchi, Netto, and Buyya (2015)
Mukherjee et al. (2012)
Najafabadi et al. (2015)
Shi et al. (2015a)
Schuster et al. (2015)

Table 5: Literature about data analytics

Cloud performance measuring
Iosup et al. (2011)
Iosup, Sonmez, Anoep, and Epema (2008)
Ousterhout, Rasti, Ratnasamy, Shenker, and Chun (2015)
Schad, Dittrich, and Quiané-Ruiz (2010)

Table 6: Literature review about cloud pricing

Distributed systems
Andrews (1991)
Raicu et al. (2008)
Subhlok, Stichnoth, O'hallaron, and Gross (1993)

Table 7: Literature review about distributed systems

Task scheduling
Dogar, Karagiannis, Ballani, and Rowstron (2014)
Javanmardi et al. (2014b)
Kaulakienė et al. (2015)
Lee and Katz (2011)
Farley et al. (2012)
Durillo, Prodan, and Huang (2013)

Table 8: Literature about task scheduling

3.2 Cloud Computing

A widely used definition of cloud computing is the definition created by Mell, Grance, et al. (2011). They define cloud computing as follows:

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

The essential characteristics Mell et al. (2011) is referring to are:

- On-demand self-service
  Users are able to purchase cloud services via a web portal. Often without a contract and paid directly by credit card.
- Broad network access
  Cloud services are used via a network connection and can be accessed on various devices.
- Resource pooling
  Users use a virtual instance, which is often shared with other users.
- Rapid elasticity
  Cloud services can be scaled, often automatically, at any given moment.
- Measured service
  Cloud clients pay per use, often per minute of used service.

Mell et al. (2011) is also referring to three service models: software as a service, platform as a service and infrastructure as a service and four deployment models: private cloud, community cloud, public cloud and hybrid cloud. This research project is primarily concerned with infrastructure as a service in the public cloud deployment model.

3.2.1 Cloud Computing History

Cloud computing, which is basically a form of grid computing, became increasingly popular since 2006, when Amazon introduced Amazon Web Services (Fox et al., 2009).

Most companies that are cloud providers today started with a different business model, since there are large investments involved with offering cloud services. Companies with an existing and profitable business model could start renting out their unused capacity. For instance, Amazon, did have a lot of excess capacity that their online web shop only needed during peak hours. With the introduction of faster internet connections the need of a direct connection to a server became significantly less important and cloud computing started becoming a viable alternative to on-premise solutions.

3.3 Cloud Task Description

There is a lot of research on data analytics in the cloud. Many researchers have written about running analytical tasks on commodity cloud hardware. The majority of the research is about scheduling tasks in an efficient manner.

The first step is to establish a clear definition of a task. Applications perform rich and complex tasks, such as a search query or build a feed. These tasks involve hundreds or thousands of components, which have to be finished before the a task is complete (Dogar et al., 2014). (Dogar et al., 2014) gives the following definition for a task:

> "The unit of work for an application that can be linked to a waiting user."

Agmon Ben-Yehuda et al. (2013) divide data analytics software into three categories. High-performance computing, which is concerned with solving a single task, high throughput computing which is concerned with handling many tasks and many task computing which is concerned with handling a high number of tasks with a large input size. They make this categorisation based on two dimensions: number of tasks and task size. Iosup et al. (2008) use the term bags-of-tasks (BoT). BoT being large sets (bags) of sequential tasks. BoT is a term used for over a decade, well before cloud computing existed (Andrews, 1991).

The similarity between these descriptions is that a task consists of a number of components that can often be handled in a sequential fashion. The hardware requirements of a task depend on the specific task-components and the input size of the data. This is illustrated by Agmon Ben-Yehuda et al. (2013) in Figure 7.

Agmon Ben-Yehuda et al. (2013) states that the number of tasks and the input data size is important when selecting the right hardware. Since loosely coupled tasks can be run independent the demand on the network transfer speeds is lower. For highly coupled tasks that are dependent on each other an environment with fast network connections is more suitable. Tasks with a higher data input size also have higher CPU demands to be handled efficiently. This is also stated by Ousterhout et al. (2015), who show that CPU and network performance often have an impact on the performance and task completion time.

Hardware requirements of a task can be divided into these categories: light hardware (usually cheap instances) and heavy hardware. Next to these categories there is a third category, specialised hardware, for tasks can make use of a GPU (Lee & Katz, 2011).

Next to the hardware characteristics of a task, also non-technical characteristics can be important in the cloud selection process (Assunção et al., 2015). Since none of the authors created an extensive overview of all task characteristics (technical and non-technical), but only mentioned tasks characteristics related to their specific research project an overview was created to devise a complete list of task characteristics. This was done in the form of a comparative analysis of all the authors. To create this analysis

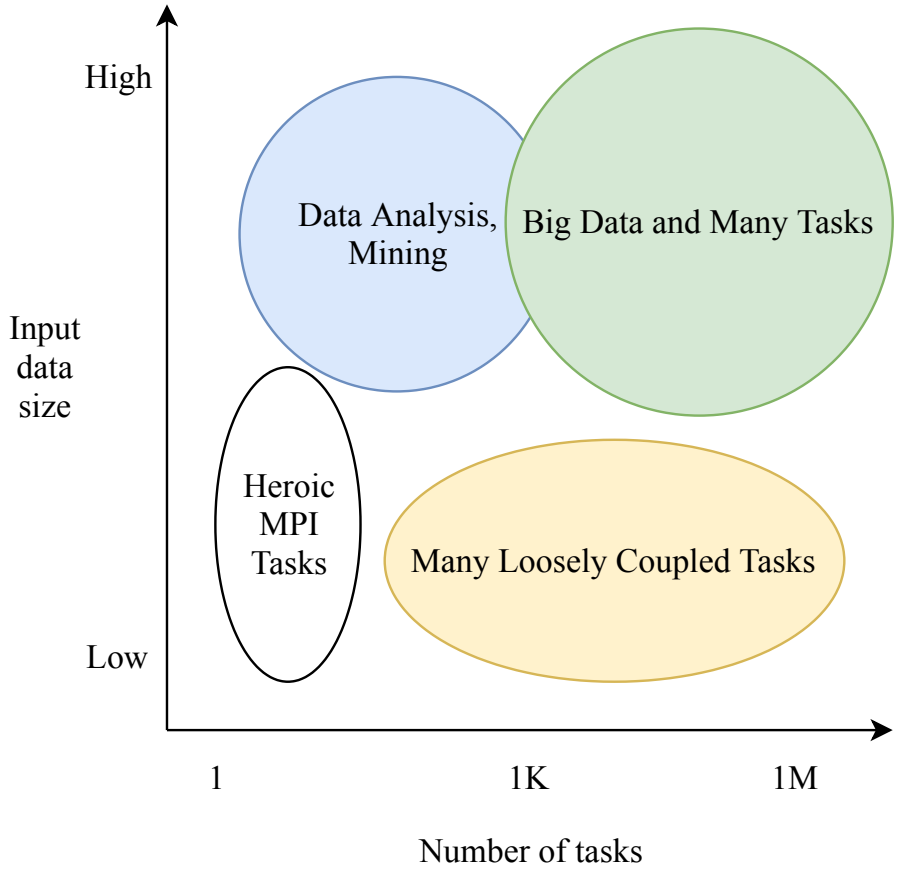


Figure 7: Input Data Size / Number of Tasks overview

all literature was read and task characteristics were extracted. An overview of all task characteristics extracted can be found in Table 9.

The task characteristics in Table 9 are ordered based on the number of mentions. The following task characteristics have been mentioned at least three times: file system characteristics, data locality, data parallelism, resource usage, task size, data classification and data velocity.

3.3.1 File system

Since cloud based analytics is usually performed on a platform comprised of several nodes, the file system has an important role in connecting the nodes via storage. Especially big data platforms are heavily reliant on the file system to supply nodes of a stream of data (Archenaa & Anita, 2015).

More recently, (Ousterhout et al., 2015) found however, that I/O optimizations cannot improve task runtime by more than 19%. One reason for this is that CPU utilization is typically close to 100% whereas disk usage is typically around 25%. This

	Assunção et al. (2015)	Archenaa and Anita (2015)	Mukherjee et al. (2012)	Raicu et al. (2008)	Dogar et al. (2014)	Ousterhout et al. (2015)	Najafabadi et al. (2015)	Shi et al. (2015b)	Javanmardi et al. (2014b)	Schuster et al. (2015)	Number of mentions
File system	✓	✓	✓	✓		✓					5
Data locality	✓		✓	✓				✓			4
Data parallelism	✓		✓	✓		✓					4
Resource usage	✓					✓		✓	✓		4
Task size				✓	✓			✓	✓		4
Data classification	✓	✓	✓								3
Data velocity	✓		✓				✓				3
Data variety	✓						✓				2
Security	✓									✓	2
Categories of analytics	✓										1
Data veracity							✓				1
Data volume							✓				1
Input size / output size ratio								✓			1
Makespan						✓					1
Meta-data dependancy				✓							1
Privacy	✓										1
Task flows					✓						1
Task interactivity	✓										1
Task volume				✓							1

Table 9: Cloud Task Characteristics Comparative Analysis. *The papers with zero characteristics were omitted from this table.*

means that the primary bottleneck is CPU-bound instead of I/O-bound.

File system requirements are a cloud task characteristic depending on the type of task. Especially analytical platforms for big data are making extensive use of the file system.

3.3.2 Data locality

Data locality is concerned with the location of data. Since data sets can get very large the location of the data in relation to the location of the processing capacity can play an important role. (Raicu et al., 2008) speaks about the gap between compute power and storage performance.

While typically data was transferred to the high performance computing systems in order to process it, bringing the computing power to the data in order to avoid data transfer is becoming the preferred approach when handling large sets of data (Assunção et al., 2015).

3.3.3 Data parallelism

Data parallelism is a form of parallelization across multiple nodes (Subhlok et al., 1993). It is opposed to task parallelism, where tasks are executed in parallel, often on the same data set. In cloud environments it is often used in big data platforms, like Hadoop, to apply load balancing (Assunção et al., 2015).

3.3.4 Resource usage

Resource usage is concerned with the amount of resources needed to perform a certain task. While CPU utilization tends to be very high (Ousterhout et al., 2015), this is not the case for disk I/O rate and network throughput. These last characteristics are task dependent. Based on the literature the most important resource related task characteristics are: cpu utilization, disk I/O rate and network throughput.

3.3.5 Task size

Dogar et al. (2014) mentions task size as a characteristic of an application task. They continue explaining that task size is the network footprint of a task, the sum of the sizes of network flows involved by the task.

In Figure 8 (left) the normalized distribution of task sizes for the query-response workflow of a search engine is presented. In Figure 8 (right) the normalized distribution of input size of MapReduce jobs is presented. This shows that tasks can have a high variability in task size.

3.3.6 Data classification

Data classification refers to the structure of the data. Big data consists of mostly unstructured data (Assunção et al., 2015). Data can be classified in three ways: unstructured, semi-structured and structured. Unstructured data is data like hand written physician notes, semi-structured data is a form of structured data that is not according to a specific data model. Data has to be classified in order to perform meaningful analysis (Archenaa & Anita, 2015).

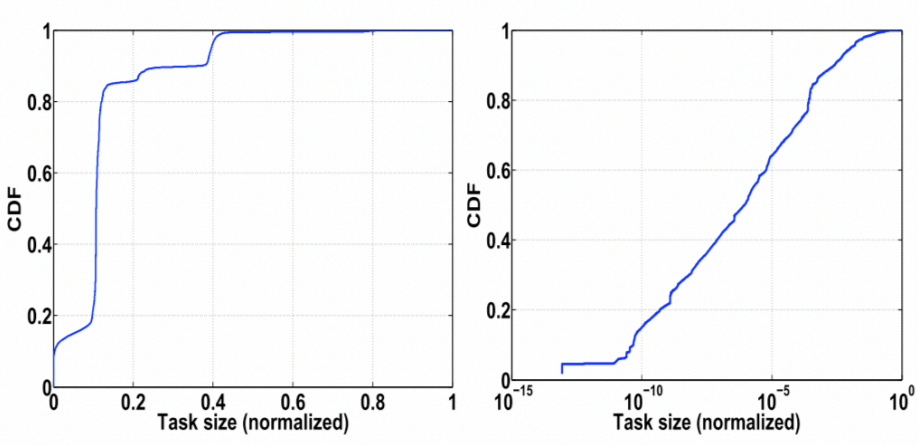


Figure 8: Normalized distribution of task sizes for search (left) and data analytics (right) workflows. The task size was normalized, the Y-axis shows the probability. Dogar et al. (2014).

3.3.7 Data velocity

Data velocity is a term primarily used in the big data research field. It refers to the rate at which data is collected and obtained (Najafabadi et al., 2015). According to (Assunção et al., 2015) data arrive at different speeds:

- Batch: *At time intervals*
- Near-time: *At small time intervals*
- Real-time: *Continuous input, process, output*
- Streams: *Data flows*

3.4 Cloud Environment Description

Cloud providers have an extensive portfolio of products with a variety of configuration options. We took the cloud providers own description of the various cloud offerings as a starting point. In the following chapters we will shed light on the descriptions of three major cloud providers: Amazon Web Services, Microsoft Azure and Google Cloud.

3.4.1 Amazon Web Services

Amazon has over 150 EC2 Virtual Machine (VM) instances on offer. Amazon divides these VMs into five instance families: general purpose, compute optimized, memory optimized, accelerated computing and storage optimized. Within these instance families Amazon offers different instance types which are comprised of a number of instance models (Amazon Web Services, 2019).

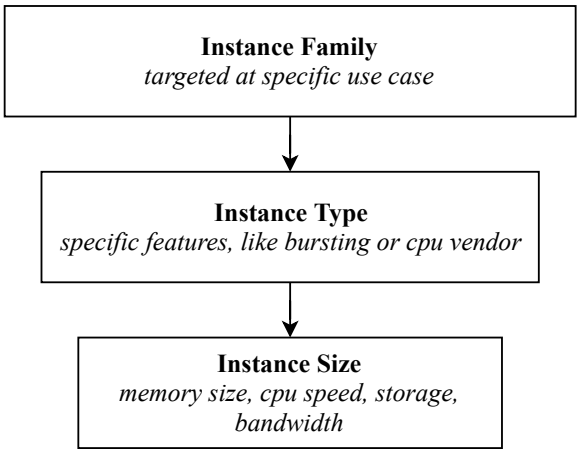


Figure 9: Amazon EC2 product structure

The instance families distinguish between specific use cases, where the the instance types differentiate between hardware vendors and components. The instance models differentiate between hardware configurations. An illustration of AWS EC2 portfolio can be found in Figure 9.

3.4.2 Microsoft Azure

Azure offers VMs in six different categories. These categories are similar to the instance families used by Amazon. Azure currently offers the following categories of VMs: general purpose, compute optimised, memory optimised, storage optimised, GPU and High performance compute (HPC) (Microsoft, 2019).

Equal to Amazon, Azure offers various configuration options per instance category.

3.4.3 Google Cloud

The approach of Google Cloud is different from the other large providers. There are four machine type families: general purpose, compute optimised, memory optimised

and shared core (Google, 2019b).

In the first category, general purpose, Google offers the user to create a custom machine type. This allows the user to create an instance with customized virtualized hardware settings (Google, 2019a).

3.4.4 Cloud Provider Comparison

The offerings of the three largest cloud providers are similar, as can be seen in Table 10. While the VMs in each category are similar, the definitions of the types are not standardized. This means cloud providers can indicate slightly different capabilities when talking about, for instance, a memory-optimized instance.

	Amazon	**Google Cloud**	**Azure**
General purpose	✓	✓	✓
Compute optimized	✓	✓	✓
Memory optimized	✓	✓	✓
Accelerated computing	✓		✓
Storage optimized	✓		
High performance compute			✓
Shared core		✓	

Table 10: Cloud Provider Instance Type Comparison

3.4.5 Technical characteristics

The selection of an appropriate instance will be primarily based on hardware requirements of the task at hand. Schad et al. (2010) measure the following components in their research on VM performance: instance startup time, cpu performance, memory speed, disk i/o, network bandwidth (internal and external).

These characteristics are often mentioned in cloud research papers. In their research on cloud performance (Iosup et al., 2011) also mention the architecture of the machine: 32 bit or 64 bit. Since then, however, 32 bit architectures have been phased out.

Cloud providers typically use an abstract unit, elastic computing unit (ECU) or virtual CPU (vCPU), to describe the level of CPU power (Farley et al., 2012). I/O performance is usually rated with low, moderate or high, where only the system memory is indicated with absolute values.

Summarizing the above, we found that cloud providers typically use a family type to describe the intended use case of group VMs. These VMs are described with abstract unit and absolute units. In the literature found, a VM was described with the following technical characteristics:

	Schad et al. (2010)	Iosup et al. (2011)	Farley et al. (2012)	Sundareswaran et al. (2012)	Voorsluys and Buyya (2012)	Number of mentions
Number of processing units	✓	✓	✓		✓	4
Processing speed	✓	✓	✓			3
Memory in gigabytes	✓	✓	✓			3
Network bandwidth	✓	✓	✓			3
Storage in gigabytes	✓			✓		2
Storage speed	✓		✓			2

Table 11: Cloud Environment Technical Characteristics

In Table 11 the technical characteristics mentioned by the authors from the literature study can be found. The three major cloud providers, Amazon, Azure and Google mention the same characteristics in their virtual machine descriptions.

3.4.6 Non-technical characteristics

Thus far we have looked at the technical characteristics of cloud environments, to make a selection we also need to have insights in the non-technical characteristics of the cloud environment (Sundareswaran et al., 2012; ur Rehman et al., 2011; Redl et al., 2012). The main characteristics per cloud provider can be found in Table 12.

	Amazon	**Azure**	**Google**
Price	Billing per second	Billing per second	Billing per second
Pricing model	On-demand, reserved and unutilized	On-demand, reserved and unutilized	On-demand, reserved and unutilized
Locality	18 regions worldwide	46 regions worldwide	20 regions worldwide

Table 12: Non-technical characteristics of major cloud providers

Price As can be seen in Table 12 all cloud providers offer billing per second. This makes it possible to acquire VMs even for tasks with a very short makespan.

Pricing models All major cloud providers offer the same pricing models, with some slight differences.

In the *on-demand model* VMs can be rented at any given time for an undefined period.

The *reserved model* offers users a discount when they commit for a longer period of time, usually one to three years.

The *unutilized model* is the most interesting model for batch tasks. The major cloud providers, Amazon, Azure and Google, call this pricing model respectively spot instances, low-priority VMs and preemptible VMs. Where Azure and Google offer these VMs at a discount up to 80%, Amazon offers a bidding mechanism. Users have to set a maximum *bid price* they are willing to pay. As long as the spot price is lower than the bid price the VM will run. Prices vary based on supply and demand. When the spot price rises above the bid price, the VM is cancelled automatically (Voorsluys & Buyya, 2012). Figure 10 shows the spot price of a c3.2xlarge-instance over time.

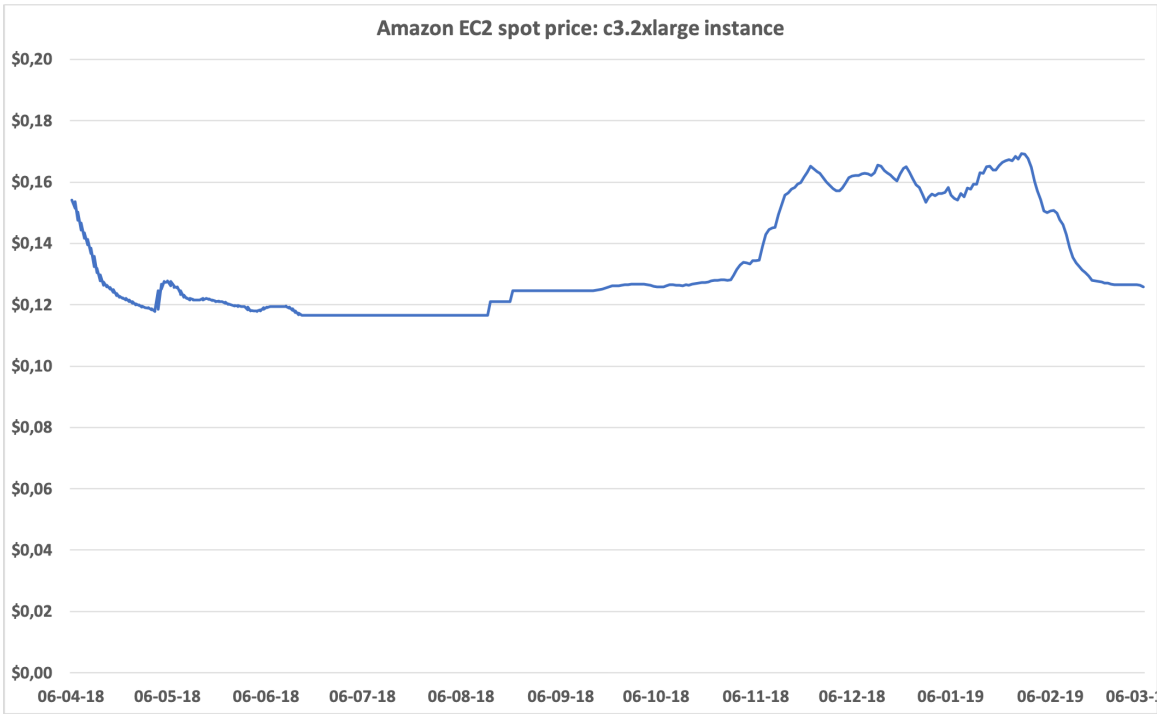


Figure 10: Spot price over time

Locality All major cloud providers are active in all regions over the world. Using instances in a different part of the world is especially important when latency issues can arise.

3.5 Cloud Task Matching

Based on the literature so far, we found the characteristics to describe tasks and to describe cloud environments. In this chapter, we will take a look at literature about task / cloud environment matching. The majority of research literature about this topic is specifically about scheduling algorithms. Since task scheduling is important in all grid-based computing environments, this research field has been around for over a decade (Subhlok et al., 1993). Grid computing existed long before cloud computing was invented. While task scheduling is also important in cloud computing environments, the number of possible environments is much higher as opposed to a typical grid environment that has the same hardware components in each node. Therefore this research focuses on finding the right cloud environment based on the task at hand.

There have been authors writing about the use of spot instances in order to gain economic benefit for scientific workloads. Since spot instances are requested via a bidding mechanism the price can vary. When the current price exceeds the bid of the user, the instance will be terminated (Zhang, Zhu, & Boutaba, 2011). Because of this, spot instances are very suitable for tasks that have a relatively short make span (shorter than the live of the spot instance) and fault-tolerance (Javadi et al., 2013).

Amazon offers an API that allows the user to request data about all VMs, including the available spot instances. This API also supplies the user of information about the cost of each instance (Qu et al., 2016). Currently it is only possible to retrieve the pricing information about the last three months.

3.5.1 Using spot instances

With regard to spot instances, two things should be considered when selecting a cloud environment for a task:

1. Is the task suitable to be run on a spot instance, regarding the task characteristics?
2. Are there any suitable spot instances available?

When the task is unsuitable to be run on a spot instance, the second question can be skipped and a selection of available on-demand or reserved instances can be made directly.

Mattess et al. (2010) have researched how cloud computing can be used to add additional resources to an existing dedicated cluster. While they focused mainly on extending capacity, they also investigated the use of spot instances in particular. Their research is based around a task provisioner where a maximum queue time is configured. Each task has a maximum queue time. When this maximum queue time is exceeded, the task is of loaded to the cloud environment to ensure making the deadline. When this happens a spot instance is selected, if that is preferred.

Kaulakienė et al. (2015) have created a framework, SpotADAPT, to remedy the limitations of spot instances. In their research they focus on two problems: the execution time of a task is unknown on different AWS instance types and the cost of execution is unknown. Because of these problems it is difficult for users to select a good trade-off between either the cheapest execution or the fastest execution. Kaulakienė et al. (2015) propose to circumvent these problems by executing small tests, so called micro-runs, with a small subset of each task on several instances. By executing these micro-runs

an estimate of the total run time of a task is obtained. The total price of execution is predicted based on the current spot price.

3.5.2 Pareto optimal matching

When running tasks the makespan is mostly influenced by the hardware configuration of the cloud environment. Because of this, the selection of a cloud environment is not purely a mathematical equation. Users have to make a trade-off between economic cost and makespan. When there is enough time, users can opt for cheaper, but slower instances. When time is scarce, users can spend more to speed up the workflow. This trade-off analysis can be solved using a Pareto analysis (Durillo et al., 2013). Pareto is an economic theory that tries to find a balance between two values (Hochman & Rodgers, 1969).

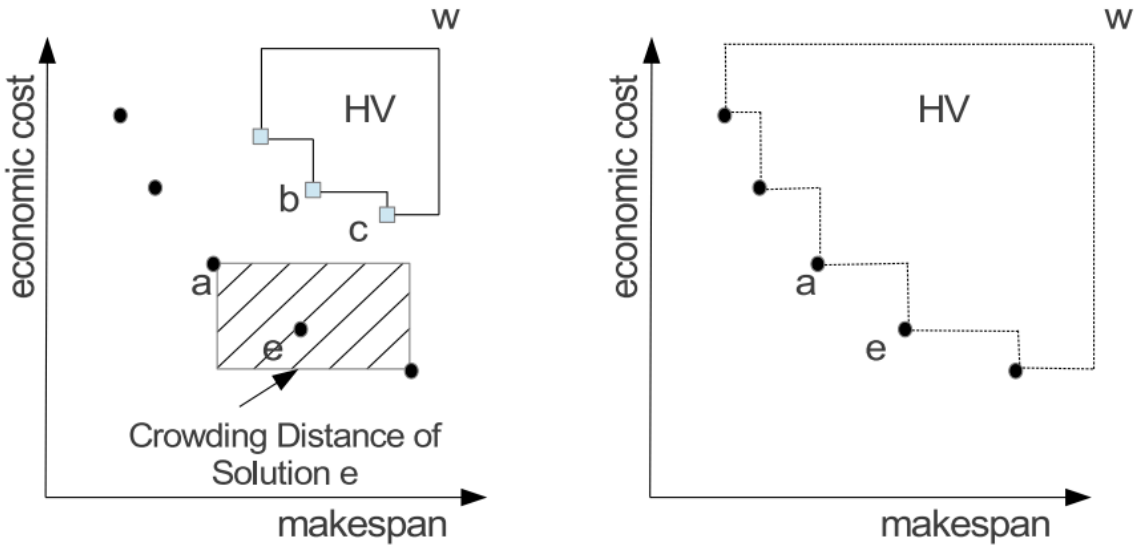


Figure 11: Pareto Front Analysis of makespan and economic cost (Durillo et al. 2013).

Durillo et al. (2013) have researched Pareto-based scheduling algorithms in the context of commercial clouds. They have found that a pareto front can be used as a tool for decision support. They have created a method of optimising the two conflicting criteria mentioned above: economic cost and makespan. Durillo et al. (2013) have used the performance benchmarks by (Iosup et al., 2011) to calculate the makespan and the economic cost of the workflow. This way they were able to visualize the pareto front. Pareto optimality is a state where two criteria are both optimal. In this context that means a short makespan and low cost. The pareto front shows all pareto efficient allocations. This pareto front is shown in Figure 11. This figure also shows the possibilities of using a pareto front for selecting the right cloud environment, since it clearly shows the pay-off between various instances. The figure shows pareto-efficient allocations of economic cost and makespan.

3.6 Summary

Summarizing the results found in literature we have found that tasks can be described with the following characteristics: file system characteristics, data locality, data parallelism, resource usage, task size, data classification and data velocity.

Cloud environments are classified using instance families, instance types and their size. The three largest cloud providers, Amazon, Microsoft Azure and Google have a similar portfolio, all of them offer VMs that are optimized for a specific use case, for instance. VMs can be purchased, or rented, in various pricing models: on-demand, reserved and unutilized.

When it comes to scheduling or provisioning tasks to a cloud environment, there are a number of options. We found approaches that did micro-runs in order to estimate the total run-time of a task, other approaches estimated this based on the task itself. Pareto analysis seems a suitable way of balancing between makespan and economic cost of a task.

4 Experiment set-up

We have now established the current state of science about cloud job provisioning. The remainder of this thesis will be about devising the decision support tool and validating whether this artefact solves the aforementioned problem.

In this chapter a baseline VM will be selected first. Amazon offers many VMs, too many to consider for this experiment. Therefore a list of available VMs will be devised. These VMs are available for the decision support tool to select.

4.1 Baseline server

This research will be comparing the results of the optimal VMs to a baseline server. To make this comparison a baseline server needs to be chosen. Since Amazon does not provide any information on the usage statistics of the VMs on offer, a frequency table of mentioned EC2 instances in the literature from the literature review was made.

Using NVivo a the number of references of all common EC2 instances was counted. Table 13 shows an overview of the EC2 instances and the number of papers that referred to each instance. Since most literature found was written before the introduction of the m5 instance and the fact that the m4 family is still available, the m5 family is not mentioned in the literature.

The m1.small instance was mentioned by most papers, with 36 mentions. This is probably due to the fact that this is the cheapest instance that Amazon offers. The xlarge instance size is the second most mentioned instance size. It also offers a well balanced configuration suitable for many purposes.

		Family							
		m1	m2	m3	m4	r3	c1	c3	c4
Size	small	36	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
	medium	6	n.a.	9	n.a.	n.a.	27	n.a.	n.a.
	large	29	n.a.	6	3	6	n.a.	5	2
	xlarge	26	11	9	3	9	24	4	2
	2xlarge	n.a.	16	6	4	6	n.a.	5	3

Table 13: Number of references per EC2 instance

Since the m1, m2 and m3 instance families are phased out and the m5 instance is just released, the m4 instance family is taken as the baseline instance. The xlarge is the most mentioned instance size that is suitable for data analytics workloads. Therefore the m4.xlarge instance is selected as the baseline server.

The m4.xlarge instance family uses Intel Xeon processors from the Haswell and Broadwell generation. The baseline server features 4 vCPUs and 16 gigabytes of RAM. Furthermore the network performance is rated as high (Amazon Web Services, 2019).

The m4.xlarge is available for $ 0.222 per Hour in the on-demand model. The reserved price varies between $ 0.094 and $ 0.174, depending on the exact terms and upfront payment. The spot price, at the moment of retrieval is $ 0.0708 per hour. The spot price is approximately a third of the on-demand price. The aforementioned prices are in the EU (Ireland) region.

A graph of the spot price over three months can be found in Figure **??**.

4.2 Benchmark set-up

To perform the benchmark and create a comparison between spot-based instances and regular instances a benchmark will be performed. This benchmark is created using R (Team et al., 2013). R is a software package used for statistical analysis. To be able to boot a server in an efficient manner the Amazon Machine Image functionality of Amazon was used. This image can be deployed to each instance that Amazon has on offer. The R-Studio server software was installed on a VM and configured to run the benchmark after the server had completed the booting process. After the virtual machine was booted the exact CPU family was checked to make sure all instances used during the benchmark were using the same CPU family. This is important since Amazon sells different CPU families under the same instance name (Ou, Zhuang, Nurminen, Ylä-Jääski, & Hui, 2012). None of the benchmarks had to be restarted due to different CPU families. Next to R-Studio server a number of packages where used. The following packages were used:

- data.table
- ggplot2
- aws.s3
- plyr
- microbenchmark

The microbenchmark package was used to execute the `computeTest` function for a number of iterations. The results of each iteration, the time to complete the iteration, was then stored in an array. The results of each benchmark were stored on Amazon's object storage: S3. This S3 Object Store was shared by each benchmarked EC2 instance. An overview of the benchmark set-up can be found in Figure 12.

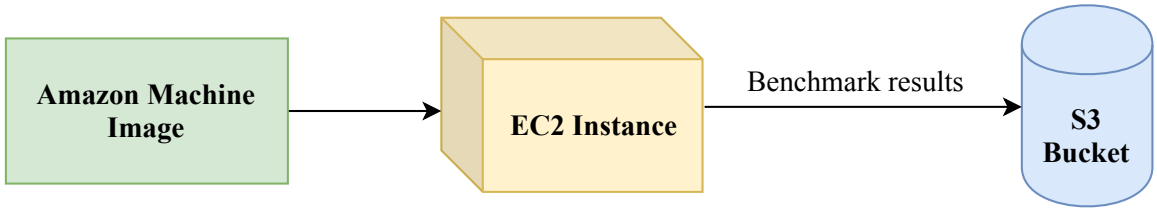


Figure 12: Overview of the benchmark set-up

The following code was written and ran in the microbenchmark package in order to obtain benchmark results for a server. The code executes multiple statistical analyses on a dataset of baseball players containing 21.699 records. The statistical analyses are common analyses that are often used, contributing to the representativity of this benchmark. This dataset is part of the R package `plyr` and can be created using the command `baseball`. It uses batting statistics of 1228 baseball players between 1871 and 2007.

```
computeTest = function(x) {
  L = length(x)
  Mis = sum(is.na(x))
  Mean = mean(x, na.rm=T)
  MEDIAN = median(x, na.rm=T)
  SD = sd(x, na.rm=T)
  var = var(x, na.rm=T)
  Q1 = quantile(x, prob = 0.25, na.rm=T)
  Q3 = quantile(x, prob = 0.75, na.rm=T)
  MODE = mode(x)
  MAD = mad(x, na.rm=T)
  MAX = max(x, na.rm=T)
  MIN = min(x, na.rm=T)
  UNIQUE = length(unique(x))
  return(c(L,Mis, Mean, MEDIAN, SD, var, Q1, Q3, MODE, MAD,MAX,MIN))
}
```

Configuration #	**Running mode**	**Instance**
1.	Single node	m4.xlarge
2.	Multi core	m4.xlarge
3.	Multi node	m4.large

Table 14: Overview of benchmarked instances

In Table 14 the configurations that were benchmarked are listed. The first configuration, single node, is the baseline configuration. This configuration makes use of a single core, so all benchmark runs are calculated on the same core. The second configuration uses the same instance, but uses multiple cores. This means the benchmark runs were split among all four cores. Each core handled 30 runs. The third configuration is a multi node configuration were the benchmark runs were divided among multiple servers. It is similar to the multi core setup as each node handled 30 runs.

5 Results

In this section the results will be discussed. The results can be divided in two parts: the performance results which are about the performance of the benchmarked configurations and the pricing results which show the price of each benchmarked configuration. Finally an overview of the price-efficiency will be given.

5.1 Performance results

In this section the performance results of the benchmark will be discussed. The full results can be found in the Appendix section (9.3, 9.4, 9.5). An overview of all results is provided in Table 15.

	baseline	**multi core**	**multi node**
instance	*m4.xlarge*	*m4.xlarge*	*m4.large*
benchmarks executions divided between	*1 core*	*4 cores*	*4 nodes*
average time per benchmark execution	*21.8 seconds*	*42.0 seconds*	*23.1 seconds*
total completion time (sec)	*2620 seconds*	*1277 seconds*	*705 seconds*
total completion time (min)	*43.67 minutes*	*21.28 minutes*	*11.75 minutes*

Table 15: Overview of performance benchmark results. Each configuration performed 120 benchmark executions. The executions were divided over the available cores or nodes.

A two-sample t-test was performed between the single core and multi node (twosample t-test, p <2.2e-16), the single core and multi core (twosample t-test, p <2.2e-16) and the multi core and multi node results (twosample t-test, p <2.2e-16).

For the benchmark runs that ran on a multi core or multi node configuration the slowest benchmark run was taken and normalized by dividing it by four.

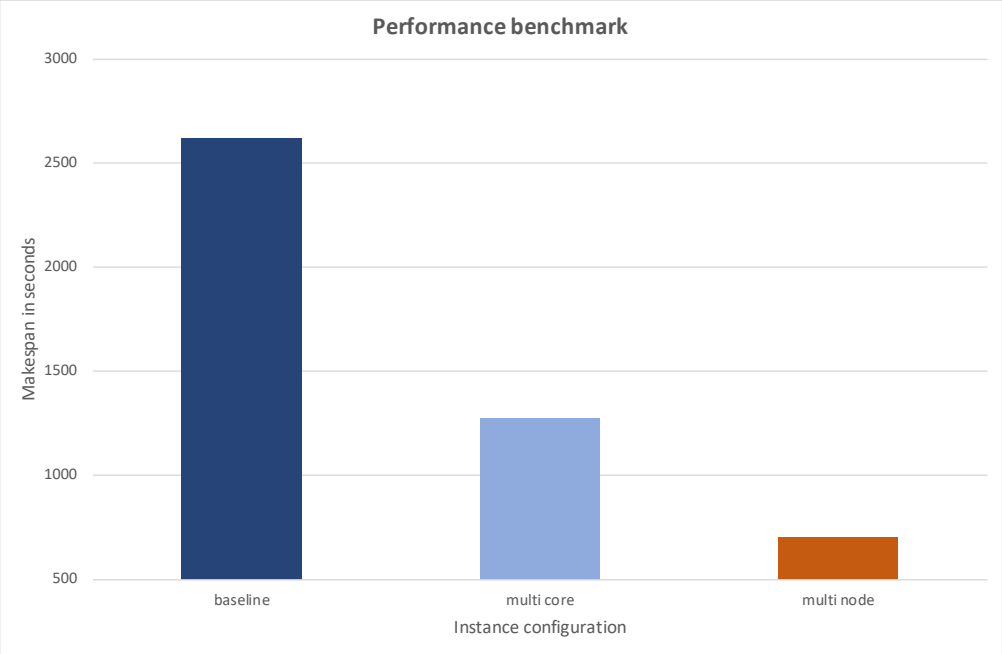


Figure 13: Graph of the benchmark results

5.1.1 Baseline

The baseline instance was a single node configuration and had an average time of 21.8 seconds per benchmark run. Since all benchmark runs where run on a single core, the completion time is the sum of all benchmark runs:

$$makespan = \sum_{i=0}^{120} time \approx 43.67\, minutes$$

The baseline configuration completed the benchmark in roughly 44 minutes (2620 seconds / 60).

5.1.2 Multi core

The multi core instance had an average time of 42.0 seconds for each benchmark run. Since the m4.xlarge instance has four cores the multicore configuration could handle four benchmark runs in parallel. The total time of all benchmark runs was 5045.0 seconds. Since the benchmark runs where split among four cores the total benchmark time is equal to the core that took the longest to complete $\frac{120}{4} = 30$ benchmark runs.

Core 1

$$makespan = \sum_{i=1}^{30} time \approx 20.75\, minutes$$

Core 2

$$makespan = \sum_{i=31}^{60} time \approx 20.72\, minutes$$

Core 3

$$makespan = \sum_{i=61}^{90} time \approx 21.28\, minutes$$

Core 4

$$makespan = \sum_{i=91}^{120} time \approx 20.87\, minutes$$

In 1277 seconds all nodes completed the benchmark runs, the total completion time was roughly 21 minutes.

5.1.3 Multi node

The multi node configuration consisted of four m4.large instances. Each instance had to perform 30 benchmark runs. The average benchmark run took 23.1 seconds. The total time of all benchmarks was 2769.57 seconds. Since the benchmark runs were split among four nodes the total benchmark time is equal to the node that took the longest to complete $\frac{120}{4} = 30$ benchmark runs.

Node 1

$$makespan = \sum_{i=1}^{30} time \approx 11.75\, minutes$$

Node 2

$$makespan = \sum_{i=31}^{60} time \approx 11.7\, minutes$$

Node 3

$$makespan = \sum_{i=61}^{90} time \approx 11.2\, minutes$$

Node 4

$$makespan = \sum_{i=91}^{120} time \approx 11.08\, minutes$$

Since node 1 took 705 seconds to complete 30 benchmark runs, the total completion time was roughly 12 minutes.

5.2 Conclusion - performance results

The performance results show that the multi-core and multi-node configurations completed the benchmarks faster than the baseline configuration. The multi-core configuration completed the benchmarks in 49% of the baseline time, where the multi-node configuration completed the benchmarks in 27% of the baseline time. The multi-node completed the benchmarks in 55% of the multi-core time.

- The multi-core configuration is more than twice as fast as the baseline server
- The multi-node configuration is almost four times faster than the baseline server
- The multi-node configuration is almost twice as fast as the multi-core configuration

The performance results show that the multi-node configuration is the fastest configuration of the tested configurations. The single core performance of the m4.xlarge instance is higher than the m4.large instance, with an average time of 21.8 seconds per benchmark run for the m4.xlarge and an average time of the 23.1 seconds for the m4.large instance. When running the benchmark at multiple cores the average time per benchmark run increases to 42.0 seconds.

5.3 Pricing results

In this section the financial side of the benchmarks will be discussed. This is done by using historical spot price data and the spot price at the moment the benchmarks were run. The following graphs show the spot price of the used instances over the period of 6 April 2018 to 6 March 2019.

Figure 14: Graph of the spot price of m4.xlarge and m4.large

There were small differences between the spot prices of the instances in the available regions. The average price followed the same pattern. The average price over the measured period of the m4.large instance was $ 0.03 cents per hour and the average price of the m4.xlarge instance was $ 0.07 cents per hour.

The prices at the moment of the benchmark can be found in Table 16. The results discussed in this section are based on the prices at the moment of execution. The prices did not vary during the benchmark. This means that exploiting price fluctuations during task executing will not be very feasible. A price change is not happening often enough to wait for lower prices, unless task execution can be postponed for long periods.

In the Table 17 the total price of each benchmark can be found. To calculate the total price of execution, the total time of execution was multiplied with the price per second. The price per second was calculated by dividing the price per minute by 60.

While the multi core configuration was 51.3% cheaper than the baseline configura-

	m4.large	**m4.xlarge**
on-demand price	$ 0.111	$ 0.222
spot price	$ 0.0344	$ 0.0703
spot price vs on-demand price	31.0%	31.7%

Table 16: Benchmark instance price overview

	baseline	**multi core**	**multi node**
instance	m4.xlarge	m4.xlarge	m4.large
no of nodes	1	1	4
no of cores	1	4	1
average time per benchmark run	21.8 sec	42 sec	23.1 sec
price of execution on-demand	$ 0.1616	$ 0.0787	$ 0.0869
price of execution spot	$ 0.0512	$ 0.0249	$ 0.0269

Table 17: Total execution cost

tion in the on-demand price model and spot price model, the multi node configuration shows promising results in terms of achieving high cost-effectiveness. The results show that, although the multi node configuration uses four instances instead of one instance the total price is similar to the total price of the multi core configuration. While the performance of the multi node configuration was almost twice that of the multi core configuration, the price is only 10.4% higher. When running the multi node configuration in the spot price model, the multi node configuration is 8.1% higher.

Compared to the baseline configuration the multi node configuration was 46.2% cheaper in the on-demand model and 47.3% cheaper when using spot pricing. Figure 15 shows an overview of all costs related to running the benchmarks.

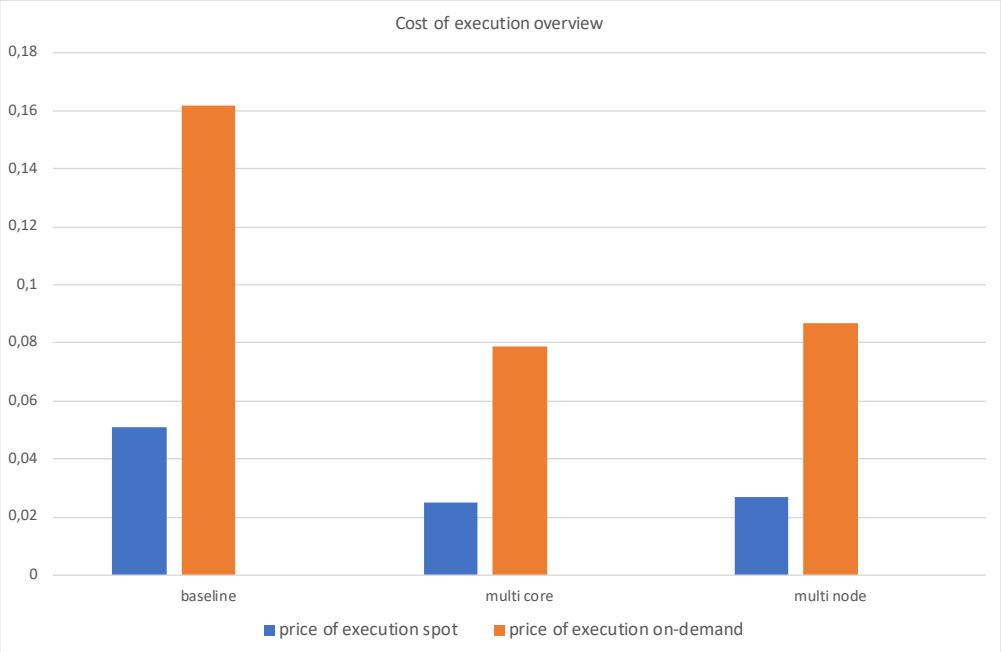


Figure 15: Price of execution

6 Cloud Selection Model

In this chapter the cloud selection model will be discussed. The results of both the experiment and the literature study were used to assemble the cloud decision model. The cloud decision model helps the user selecting the right Amazon EC2 instances and takes technical and non-technical requirements into account. The model is comprised of multiple decisions that guide the user to the right instance. The model also takes spot instance availability and suitability into account. In this chapter each decision that a user of the model has to make will be discussed and explained.

The first part of the decision tree, containing the decision about the hardware requirements can be found in Figure 16. The hardware requirement decisions determine the suitable instance families. The second part of the decision tree can be found in Figure 18. The second part of the decision tree determines the technical configuration and pricing model. The complete decision tree can be found in Appendix 9.6.

While this approach, using a decision tree, gives a clear overview the model looks rather complex and has duplicated elements. Therefore also a cloud family matrix is proposed, this shows an overview of the available instances with the average memory available and the cpu speed. The matrix can be found in Figure 17.

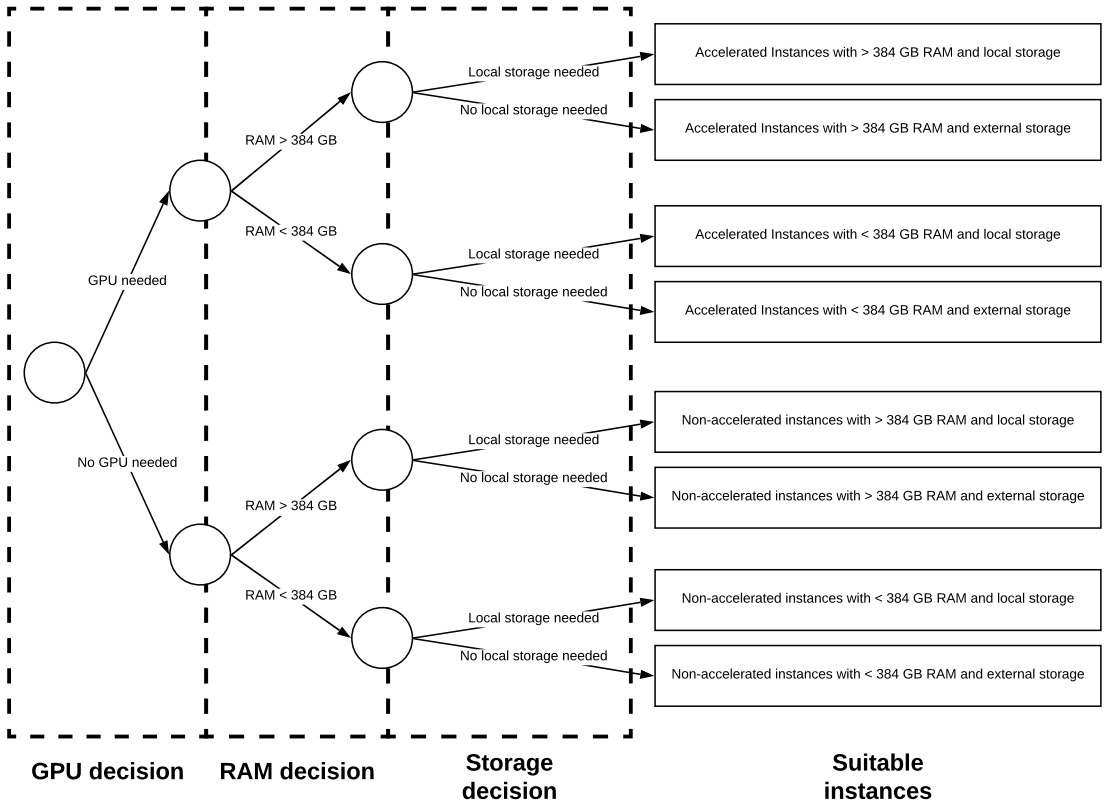


Figure 16: Decision tree: hardware requirement decisions

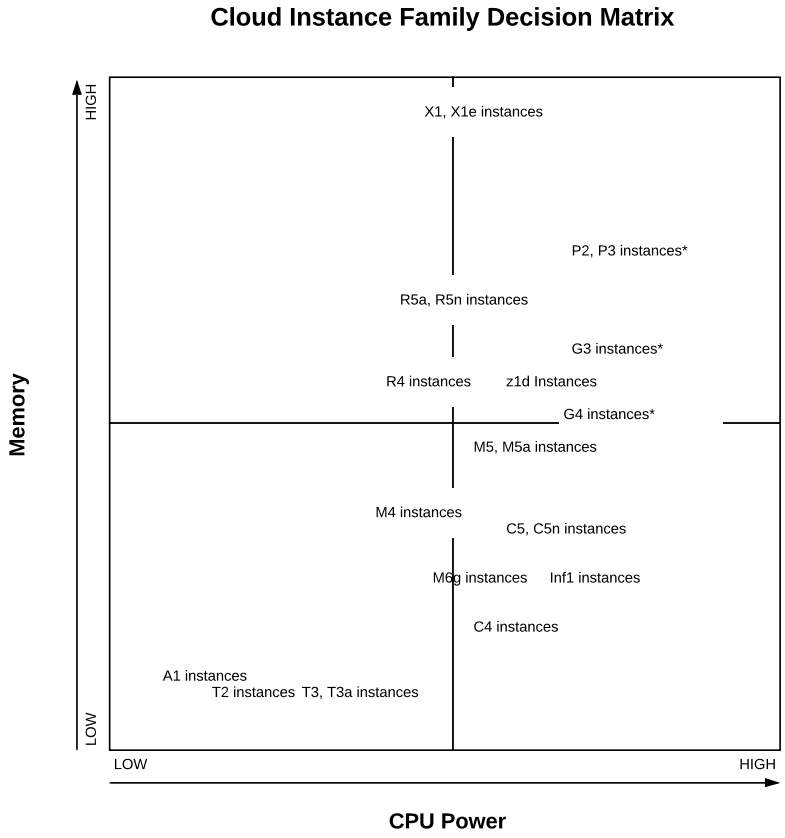


Figure 17: Cloud Family Matrix

6.1 Decision 1: GPU requirement

Although originally intended to perform operations linked to graphics, a Graphics Processing Unit (GPU) is also suitable for non-graphics applications. Because of the physical per-core restraints and the architectural differences between GPUs and CPUs, GPUs can perform certain operations faster than CPUs (Brodtkorb, Hagen, & Sætra, 2013).

Amazon offers GPU-enabled instances in one instance family, the accelerated instances. The first question of the decision tree is therefore whether a GPU is needed or not since it disqualifies all other instance families that have no GPU available.

6.2 Decision 2: RAM requirement

The second decision in the decision tree is about the memory requirements. Depending on the size of the dataset the memory requirements increase. All major cloud providers offer memory optimized instances that have large amounts of RAM. Amazon offers a memory optimized instance family with instances that offer up to 24576 GB of memory. The memory optimized family is the only instance family that offers instances over 384 GB RAM. The other instance families do not have more than 384 GB RAM. Therefore,

when a task needs more than 384GB of RAM the only suitable cloud instances are in the memory optimized family. Since this limits the number of suitable instances this number is also used in the decision tree.

6.3 Decision 3: Storage requirement

The third decision in the decision model has to do with how cloud instances are constructed. Since cloud computing leans highly on virtualization also storage is often virtualized and offloaded to another server. The storage instance is then accessed via a network connection. While this solution is suitable for some applications, some tasks require higher throughput speeds than network connections can offer. In these cases local storage in the form of an solid state drive (SSD) is needed. Not all Amazon instances offer local storage, thus the third hardware decision in the decision tree is whether local storage is needed.

6.4 Decision 4: Pricing model

Apart from deciding about the hardware requirements, also a number of business decisions have to be made. The second part of the decision tree, see Figure 18 contains the business decisions. The first decision is about the pricing model. Since Amazon and other cloud providers offer an unutilized instances model which offers significant discounts this is an important decision. To be able to make this decision two questions have to be answered.

The first question to answer is whether there are unutilized instances available. Because the availability depends on the utilization of other customers this pricing model is not always available.

Secondly it is important to weigh in the task duration. Since the availability of unutilized instances can not be guaranteed, this pricing model is not suitable for tasks with higher makespans since an outage would mean the task has to be started again.

When the questions above are answered, a decision about the unutilized pricing model can be made. When opting for a regular pricing model a decision between reserved and on-demand instances has to be made.

6.5 Decision 5: Parallelisation suitability

The fifth decision in the decision model is concerning the parallelisation of the task at hand. If a task can be parallelised it can be performed in a multi-core or a multi-node approach. Given the results of the experiment in Chapter 5.2 parallelisation can deliver significant performance gains.

6.6 Decision 6: Parallelisation approach

If parallelisation is possible for the task at hand, then an approach has to be chosen. The decision tree offers two parallelisation approaches: multi-core parallelisation and multi-node parallelisation. The results in Chapter 5.2 show that multi-node parallelisation is significantly faster than multi-core parallelisation, at a minor price increase. Depending on the business requirements in terms of deadline a multi-core approach can be desirable if there are no time constraints and strict budget constraints.

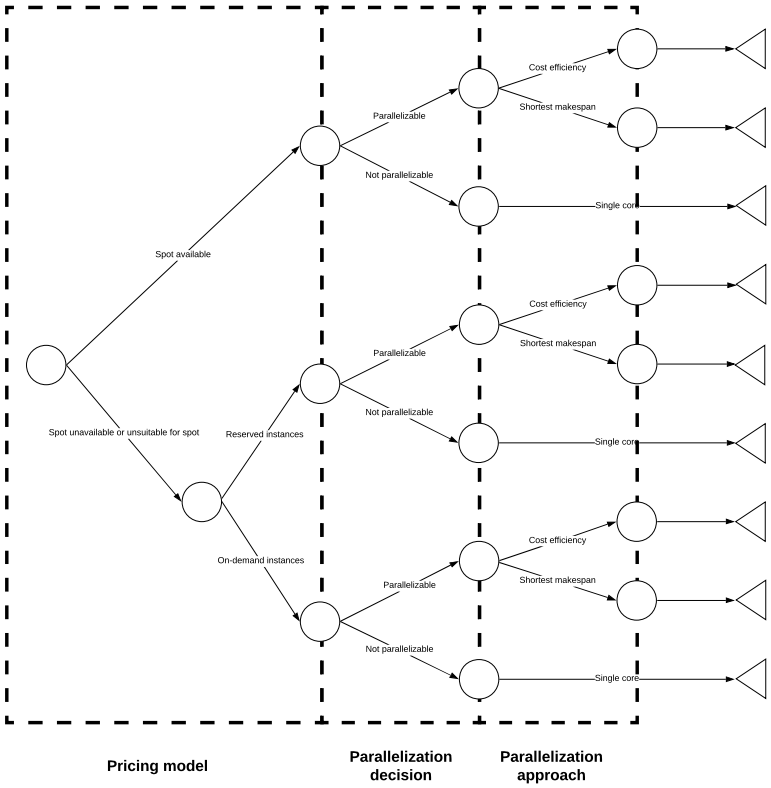


Figure 18: Decision tree: non-hardware related decisions

6.7 Cloud Instance Decision Tree - summary

The decision tree which has been discussed in this table helps a user in finding one or more suitable cloud instances based on the hardware requirements of the task at hand, a pricing model and a parallelisation approach based on the business requirements. When the outcomes of all decision tree nodes are known the right cloud instance can be deployed.

The cloud instances which are listed on the right side of the decision tree comply to the hardware requirements. Next to the list of suitable cloud instances or VMs is a parallelisation approach and the pricing model. The outcome of the top node is shown in Figure 19. All possible outcomes can be found in Appendix 9.7.

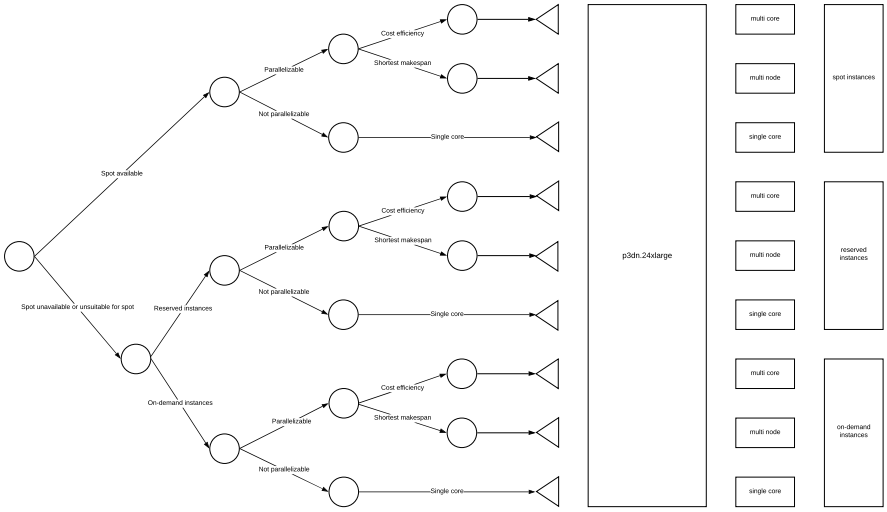


Figure 19: Decision tree: Instance with a GPU, over 384GB of RAM and local storage

7 Discussion

This section discusses the findings of this research project. In this section the limitations of the research project and the cloud instance decision tree will be discussed as well as the recommendations for future work.

7.1 Limitations of the research project

This research project set out to solve the problem of selecting suitable cloud instances when deploying analytical tasks to the cloud. The portfolios of multiple cloud providers were analyzed to find similarities between their offerings. We found that all cloud providers have a similar portfolio, all major cloud providers offer the same virtual machines that are suitable for a wide variety of (analytical) tasks. This research mainly focused on the, currently, largest cloud provider Amazon Web Services (AWS). While choosing a single provider to focus this research project has some implications for the generalizability, the way of deploying and selecting cloud instances is the same for the large cloud providers. Users of cloud computing should be aware of vendor lock-in, which can result in risks regarding business continuity and switching costs. Because all major cloud providers offer industry standard deployment methods, for instance Docker, these risks can be mitigated.

In order to create a decision tool to aid researchers utilizing cloud computing, an extensive analysis of literature about data analytics in the cloud was carried out. The literature review yielded results on cloud task description, cloud environment description and cloud task matching. Next to a literature study an experiment was conducted to discover the benefits of multiple cloud configurations. The experiment was executed using AWS EC2 instances. The m4.large and m4.xlarge instances were compared using an R-based benchmark running on single core, multi core and multi node configured instances. The results showed an advantage of using m4.large instances in a multi node configuration in terms of performance en cost efficiency. The multi node configuration performed almost four times better than the single core, baseline configuration, while being more than 46% cheaper.

There are limitations to this experiment. In the experiment the boot-up time was disregarded in the experiment, since we wanted to benchmark the performance of task processing. The boot-up time has an effect on the total makespan, but for tasks with a longer makespan the boot-up time is negligible. The boot-up time is equal for all instances, regardless of the specifications. Boot-up time should be accounted for when task makespan is short.

The second limitation of the experiment is the loading time when processing large datasets. The dataset in the experiment was relatively small, such that it could be stored in memory. When processing large datasets the dataset the virtual machine has to read the dataset from the harddrive which takes time. This is especially important in a multinode set-up were multiple nodes have to load the dataset.

The third limitation of the experiment is the size of the experiment, this research project can be improved by incorporating more types of data analytics and machine learning approaches and by using a broader set of cloud instances. The most frequently used cloud instances in literature were used for the benchmark, but there are more cloud instances that could provide more information about optimal task-cloud matches. Other types of data analytics were beyond the scope of this research project but would

improve the applicability of the decision tree.

The results of the literature study and the experiment where then used to devise a cloud decision tree. The decision tree uses the following parameters to guide the user to a cloud instance:

- Suitability of the task with unutilized instances
- Availability of unutilized instances
- Task hardware requirements
- Budget

The decision tree incorporates these parameters in order to aid the user in selecting suitable AWS EC2 instances for a workload. A limitation of this approach is that the user needs information about the parameters of a dataset or workload. While experienced researchers know the parameters this can be a problem for users with less experience or knowledge about the dataset. To further improve the decision tree more questions can be added in order to simplify the decision process for users. Another limitation of the current decision tree is that it focuses on Amazon Web Services. Amazon is currently the largest cloud provider with a broad portfolio which makes it an obvious choice. By replacing the instances in the decision tree with similar instances of another cloud provider the decision tree can also be used to select instances of other cloud providers, making the decision tree a generic solution.

8 Conclusion

In this chapter the research questions will be answered. This research aimed to identify a strategy to match analytical tasks to virtual machines or cloud instances. Based on a literature study and an experiment with cloud instances a cloud instance decision tree was devised taking all requirements involved with the decision into account. The decision tree aids users in selecting an optimal cloud instance.

8.1 Sub questions

The research sub questions will be answered first. In this section the sub questions of this research project are listed and answered.

SQ1: How can the requirements and characteristics of a cloud task be described?

A cloud task is a unit of work, on which a user is waiting. Tasks can be described with the following characteristics: file system characteristics, data locality, data parallelism, resource usage, task size, data classification and data velocity. These characteristics describe the technical requirements of a task. These requirements are necessary in order to select the right cloud instance.

SQ2: How can cloud instances be described?

To effectively describe a cloud instance, the instance family, instance type and instance size are used to describe the technical characteristics of an instance. All major cloud providers offer instances with similar characteristics. Next to a technical description a cloud instance is obtained in multiple pricing models, on-demand, reserved and unutilized. To select the most optimal cloud instance for a given task, the pricing model is important since it has a large impact on the total cost of processing said task.

SQ3: How can tasks and cloud instances be matched?

To match tasks to a cloud instance it is important to know the task characteristics and the instance characteristics, as well as the business requirements. To select the instance a number of decisions should be taken, which are listed in the Cloud Instance Decision Tree.

8.2 Main research question

The main research question will be answered below.

RQ: How can analytical tasks be matched to a cloud instance taking into account all requirements?

This research project set out to devise a way to match analytical tasks to a cloud instance. This was done using a design science approach were literature and data was gathered to devise an artifact. The literature yielded knowledge about cloud task and instance description and task provisioning. The experiment, which existed of benchmarks of cloud configurations, showed the advantages of various cloud configurations.

The multi node approach proved to be an improvement over the baseline (single core) and multi core approach in terms of cost efficiency and performance.

The results of the benchmark were incorporated in the design artifact. This artifact is the Cloud Instance Decision Tree. To match analytical tasks to cloud instances the technical and non-technical requirements and characteristics of that task should be known. The task requirements differ per task and should be known to the user. Furthermore the cloud instances should be described and the characteristics of the instances should be known. The artifact on this design science research project, the Cloud Instance Decision Tree, is based on Amazon's EC2 virtual machine portfolio. Amazon is currently the largest cloud provider. The decision tree incorporates all technical and non-technical characteristics of the cloud instances that Amazon has on offer.

The decision tree takes all requirements into account by incorporating both the task characteristics and the cloud instance characteristics. By using the Cloud Instance Decision Tree a user is enabled in selecting the right cloud instance with the necessary technical requirements, the optimal pricing model and the configuration that is optimal for the requirements. The decision tree offers one or more actual instances in the form of the Amazon EC2 instance API name, for instance an m4.xlarge. The pricing model is based on the availability of unutilized instances and also takes the other pricing models, on-demand and reserved, into account. Finally the user is also guided to the right configuration, which can be single core, multi core or multi node. In a single core configuration, only one core of the CPU is used. In a multi core configuration one instance is used, but all available CPU cores are used to process the calculations. In the multi node approach the workload is divided over multiple instances of which a single CPU core is used. This way cheaper instances can be used which results in a cost effective approach to cloud data analytics.

8.3 Future work

It was beyond the scope of this research project to create a truly automated selection tool. To be able to use the decision tree in a completely automated fashion it should be built as a software component that integrates the AWS API to retrieve actual data about spot instances. Since spot prices vary over time based on the current utilization rate and demand it is necessary to retrieve this data in real-time to select the most optimal instance. Next to the price fluctuations of the unutilized instances the portfolios of all major cloud providers are continuously changing. To enable unattended instance selection it is recommended to retrieve information about the current portfolio automatically.

Next to the automated retrieval of information about the cloud portfolio and unutilized instances the cloud instance decision tree could be extended with more job specific information and budget related information to make a narrower selection of instances possible. Although all cloud instances selected via the decision tree are suitable, the instances vary in terms of price per hour and exact specifications.

References

Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., & Tsafrir, D. (2013). Deconstructing amazon ec2 spot instance pricing. *ACM Transactions on Economics and Computation*, *1*(3), 16.

Alnemr, R., Pearson, S., Leenes, R., & Mhungu, R. (2014). Coat: cloud offerings advisory tool. In *Cloud computing technology and science (cloudcom), 2014 ieee 6th international conference on* (pp. 95–100).

Amazon Web Services. (2019). *Instance types.* Amazon. Retrieved 2019-07-06, from `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html`

Andrews, G. R. (1991). Paradigms for process interaction in distributed programs. *ACM Computing Surveys (CSUR)*, *23*(1), 49–90.

Archenaa, J., & Anita, E. M. (2015). A survey of big data analytics in healthcare and government. *Procedia Computer Science*, *50*, 408–413.

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., & Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, *79*, 3–15.

Azvine, B., Cui, Z., Nauck, D. D., & Majeed, B. (2006). Real time business intelligence for the adaptive enterprise. In *The 8th ieee international conference on e-commerce technology and the 3rd ieee international conference on enterprise computing, e-commerce, and e-services (cec/eee'06)* (pp. 29–29).

Bao, S., Damon, S. M., Landman, B. A., & Gokhale, A. (2016). Performance management of high performance computing for medical image processing in amazon web services. In *Medical imaging 2016: Pacs and imaging informatics: Next generation and innovations* (Vol. 9789, p. 97890Q).

Bazeley, P., & Jackson, K. (2013). *Qualitative data analysis with nvivo.* Sage Publications Limited.

Bessai, K., Youcef, S., Oulamara, A., Godart, C., & Nurcan, S. (2012). Resources allocation and scheduling approaches for business process applications in cloud contexts. In *4th ieee international conference on cloud computing technology and science proceedings* (pp. 496–503).

Brodtkorb, A. R., Hagen, T. R., & Sætra, M. L. (2013). Graphics processing unit (gpu) programming strategies and trends in gpu computing. *Journal of Parallel and Distributed Computing*, *73*(1), 4–13.

Buyya, R., Pandey, S., & Vecchiola, C. (2009). Cloudbus toolkit for market-oriented cloud computing. In *Ieee international conference on cloud computing* (pp. 24–44).

Calcavecchia, N. M., Biran, O., Hadad, E., & Moatti, Y. (2012). Vm

placement strategies for cloud scenarios. In *2012 ieee fifth international conference on cloud computing* (pp. 852–859).

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS quarterly*, *36*(4).

Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J. M., & Welton, C. (2009). Mad skills: new analysis practices for big data. *Proceedings of the VLDB Endowment*, *2*(2), 1481–1492.

Combi, C., & Pozzi, G. (2006). Task scheduling for a temporalworkflow management system. In *Thirteenth international symposium on temporal representation and reasoning (time'06)* (pp. 61–68).

Dogar, F. R., Karagiannis, T., Ballani, H., & Rowstron, A. (2014). Decentralized task-aware scheduling for data center networks. In *Acm sigcomm computer communication review* (Vol. 44, pp. 431–442).

Durillo, J. J., Prodan, R., & Huang, W. (2013). Workflow scheduling in amazon ec2. In *European conference on parallel processing* (pp. 374–383).

Farley, B., Juels, A., Varadarajan, V., Ristenpart, T., Bowers, K. D., & Swift, M. M. (2012). More for your money: exploiting performance heterogeneity in public clouds. In *Proceedings of the third acm symposium on cloud computing* (p. 20).

Farshidi, S., Jansen, S., de Jong, R., & Brinkkemper, S. (2018). A decision support system for cloud service provider selection problem in software producing organizations. In *2018 ieee 20th conference on business informatics (cbi)* (Vol. 1, pp. 139–148).

Fern, H., Tedeschi, C., Priol, T., et al. (2011). A chemistry-inspired workflow management system for scientific applications in clouds. In *2011 ieee seventh international conference on escience* (pp. 39–46).

Ferrer, A. J., HernáNdez, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C., ... others (2012). Optimis: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, *28*(1), 66–77.

Ferry, N., Song, H., Rossini, A., Chauvel, F., & Solberg, A. (2014). Cloudmf: applying mde to tame the complexity of managing multi-cloud applications. In *Utility and cloud computing (ucc), 2014 ieee/acm 7th international conference on* (pp. 269–277).

Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... Stoica, I. (2009). Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, *28*(13), 2009.

Ghafarian, T., & Javadi, B. (2015). Cloud-aware data intensive workflow scheduling on volunteer computing systems. *Future Generation Computer Systems*, *51*, 87–97.

Google. (2019a). *Creat-*

ing a vm instance with a custom machine type. Retrieved 2019-07-12, from `https://cloud.google.com/compute/docs/instances/creating-instance-with-custom-ma`

Google. (2019b). *Machine types.* Retrieved 2019-07-12, from `https://cloud.google.com/compute/docs/machine-types`

Hochman, H. M., & Rodgers, J. D. (1969). Pareto optimal redistribution. *The American economic review*, *59*(4), 542–557.

Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., & Epema, D. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed systems*, *22*(6), 931–945.

Iosup, A., Sonmez, O., Anoep, S., & Epema, D. (2008). The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on high performance distributed computing* (pp. 97–108).

Javadi, B., Thulasiram, R. K., & Buyya, R. (2013). Characterizing spot price dynamics in public cloud environments. *Future Generation Computer Systems*, *29*(4), 988–999.

Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014a). Hybrid job scheduling algorithm for cloud computing environment. In *Proceedings of the fifth international conference on innovations in bio-inspired computing and applications ibica 2014* (pp. 43–52).

Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014b). Hybrid job scheduling algorithm for cloud computing environment. In *Proceedings of the fifth international conference on innovations in bio-inspired computing and applications ibica 2014* (pp. 43–52).

Kaulakienė, D., Thomsen, C., Pedersen, T. B., Çetintemel, U., & Kraska, T. (2015). Spotadapt: Spot-aware (re-) deployment of analytical processing tasks on amazon ec2. In *Proceedings of the acm eighteenth international workshop on data warehousing and olap* (pp. 59–68).

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), 1–26.

Lee, G., & Katz, R. H. (2011). Heterogeneity-aware resource allocation and scheduling in the cloud. In *Hotcloud.*

Malone, T. W., Fikes, R. E., & Howard, M. T. (1983). Enterprise: A market-like task scheduler for distributed computing environments.

Mattess, M., Vecchiola, C., & Buyya, R. (2010). Managing peak loads by leasing cloud infrastructure services from a spot market. In *2010 ieee 12th international conference on high performance computing and communications (hpcc)* (pp. 180–188).

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

Microsoft. (2019). *Linux virtual machines pricing.* Retrieved 2019-07-12, from `https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/`

Mukherjee, A., Datta, J., Jorapur, R., Singhvi, R., Haloi, S., & Akram, W. (2012). Shared disk big data analytics with apache hadoop. In *2012 19th international conference on high performance computing* (pp. 1–6).

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1.

Okoli, C., & Schabram, K. (2009). Protocol for a systematic literature review of research on the wikipedia. In *Proceedings of the international conference on management of emergent digital ecosystems* (p. 73).

Ou, Z., Zhuang, H., Nurminen, J. K., Ylä-Jääski, A., & Hui, P. (2012). Exploiting hardware heterogeneity within the same instance type of amazon ec2. In *Presented as part of the.*

Oussous, A., Benjelloun, F.-Z., Lahcen, A. A., & Belfkih, S. (2018). Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, *30*(4), 431–448.

Ousterhout, K., Rasti, R., Ratnasamy, S., Shenker, S., & Chun, B.-G. (2015). Making sense of performance in data analytics frameworks. In *12th {USENIX} symposium on networked systems design and implementation ({NSDI} 15)* (pp. 293–307).

Özcan, F., Hoa, D., Beyer, K. S., Balmin, A., Liu, C. J., & Li, Y. (2011). Emerging trends in the enterprise data analytics: connecting hadoop and db2 warehouse. In *Proceedings of the 2011 acm sigmod international conference on management of data* (pp. 1161–1164).

Qu, C., Calheiros, R. N., & Buyya, R. (2016). A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances. *Journal of Network and Computer Applications*, *65*, 167–180.

Raicu, I., Foster, I. T., & Zhao, Y. (2008). Many-task computing for grids and supercomputers. In *2008 workshop on many-task computing on grids and supercomputers* (pp. 1–11).

Redl, C., Breskovic, I., Brandic, I., & Dustdar, S. (2012). Automatic sla matching and provider selection in grid and cloud computing markets. In *Proceedings of the 2012 acm/ieee 13th international conference on grid computing* (pp. 85–94).

Rimal, B. P., Choi, E., & Lumb, I. (2009). A taxonomy and survey of cloud computing systems. In *2009 fifth international joint conference on inc, ims and idc* (pp. 44–51).

Schad, J., Dittrich, J., & Quiané-Ruiz, J.-A. (2010). Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of*

the VLDB Endowment, *3*(1-2), 460–471.

Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., & Russinovich, M. (2015). Vc3: Trustworthy data analytics in the cloud using sgx. In *2015 ieee symposium on security and privacy* (pp. 38–54).

Shi, J., Qiu, Y., Minhas, U. F., Jiao, L., Wang, C., Reinwald, B., & Özcan, F. (2015a). Clash of the titans: Mapreduce vs. spark for large scale data analytics. *Proceedings of the VLDB Endowment*, *8*(13), 2110–2121.

Shi, J., Qiu, Y., Minhas, U. F., Jiao, L., Wang, C., Reinwald, B., & Özcan, F. (2015b). Clash of the titans: Mapreduce vs. spark for large scale data analytics. *Proceedings of the VLDB Endowment*, *8*(13), 2110–2121.

Song, J., & Guerin, R. (2017). Pricing and bidding strategies for cloud computing spot instances. In *2017 ieee conference on computer communications workshops (infocom wkshps)* (pp. 647–653).

Srirama, S., Batrashev, O., & Vainikko, E. (2010). Scicloud: scientific computing on the cloud. In *Proceedings of the 2010 10th ieee/acm international conference on cluster, cloud and grid computing* (pp. 579–580).

Subhlok, J., Stichnoth, J. M., O'hallaron, D. R., & Gross, T. (1993). Exploiting task and data parallelism on a multicomputer. In *Acm sigplan notices* (Vol. 28, pp. 13–22).

Sundareswaran, S., Squicciarini, A., & Lin, D. (2012). A brokerage-based approach for cloud service selection. In *2012 ieee fifth international conference on cloud computing* (pp. 558–565).

Team, R. C., et al. (2013). R: A language and environment for statistical computing.

Tekiner, F., & Keane, J. A. (2013). Big data framework. In *2013 ieee international conference on systems, man, and cybernetics* (pp. 1494–1499).

ur Rehman, Z., Hussain, F. K., & Hussain, O. K. (2011). Towards multi-criteria cloud service selection. In *2011 fifth international conference on innovative mobile and internet services in ubiquitous computing* (pp. 44–48).

van de Weerd, I., & Brinkkemper, S. (2009). Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications* (pp. 35–54). IGI Global.

Von Alan, R. H., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, *28*(1), 75–105.

Voorsluys, W., & Buyya, R. (2012). Reliable provisioning of spot instances for compute-intensive applications. In *2012 ieee 26th international conference on advanced information networking and applications* (pp.

542–549).

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.

Zhang, Q., Zhu, Q., & Boutaba, R. (2011). Dynamic resource allocation for spot markets in cloud computing environments. In *2011 fourth ieee international conference on utility and cloud computing* (pp. 178–185).

9 Appendices

9.1 Appendix A

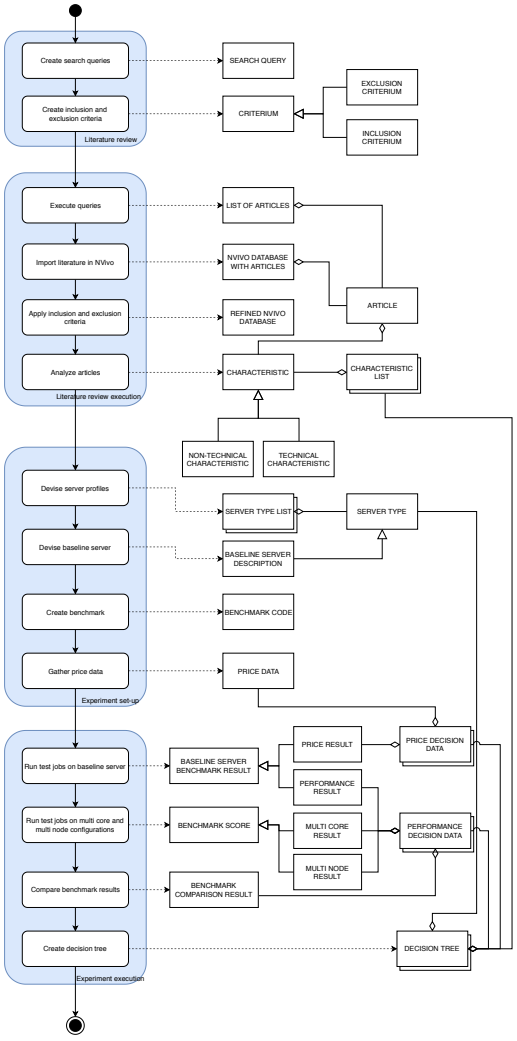


Figure 20: Process-Deliverable Diagram (PDD) of the research method

9.2 Appendix B

```
require(data.table)
require(ggplot2)
require(aws.s3)
require(microbenchmark)

Sys.setenv("AWS_ACCESS_KEY_ID" = "mykey",
           "AWS_SECRET_ACCESS_KEY" = "mysecretkey",
           "AWS_DEFAULT_REGION" = "us-east-2")

selectedBucket = 'wiertscriptie'

dt = data.table(plyr::baseball)



mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}


computeTest = function(x) {
  #Mode <<- Mode
  L = length(x)
  Mis = sum(is.na(x))
  Mean = mean(x, na.rm=T)
  MEDIAN = median(x, na.rm=T)
  SD = sd(x, na.rm=T)
  var = var(x, na.rm=T)
  Q1 = quantile(x, prob = 0.25, na.rm=T)
  Q3 = quantile(x, prob = 0.75, na.rm=T)
  MODE = mode(x)
  MAD = mad(x, na.rm=T)
  MAX = max(x, na.rm=T)
  MIN = min(x, na.rm=T)
  UNIQUE = length(unique(x))
  return(c(L,Mis, Mean, MEDIAN, SD, var, Q1, Q3, MODE, MAD,MAX,MIN))
}

SE = function(x) {
  N = length(x)
  SE = sd(x, na.rm=T)/sqrt(N)
  return(SE)
}
```

```
#finalRS = rs
ITERATIONS = 1000

for (i in 1:20) {
  cols = names(dt)[6:22]
  time <- data.table(ms=microbenchmark(
   rs <- dt[,lapply(.SD, function(x) computeTest(x)),
   .SDcols = cols, by='team'],
   times=ITERATIONS)$time/1000000
   )
  #rs = data.table(SUM = sum(time$ms, na.rm=T))
  #rs = data.table(MIN = min(
   time$ms, na.rm=T),
   Q1 = quantile(time$ms, prob = 0.25, na.rm=T),
   Q2 = median(time$ms, na.rm=T),
   Q3 = quantile(time$ms, prob = 0.75, na.rm=T),
   MAX = max(time$ms, na.rm=T)
  )
  rs = data.table(MEAN=sum(time$ms, na.rm=T), SE=SE(time$ms))
  if (i==1) {
    finalRS = rs
  }

  rs$ID=21
  rs$CLASS = 'Single-Node'
  finalRS = rbind(finalRS, rs)
}

finalRS$ID = 1:nrow(finalRS)
finalRS$CLASS = 'Multi-Node'

p = ggplot(finalRS, aes(ID, MEAN))
#p = p+geom_boxplot(aes(
ymin = MIN,
lower =Q1 ,
middle = Q2,
upper = Q3,
ymax = MAX,
group=ID,
fill=CLASS
), stat = "identity"
)
p = p + geom_bar(aes(fill=CLASS), stat = "identity")
#p = p + geom_errorbar(aes(x=ID, ymin=MEAN-SE, ymax=MEAN+SE))


#p = ggplot(dt, aes(ab,r))
#p = p+geom_point()
```

```
# fileName =  paste0("multi-node", sample.int(10000000, 1), ".csv")
fileName = 'single-node.csv'
#ggsave(fileName, p)

fwrite(time, fileName)

scriptFile = 'simple_plot_script.R'
put_object(file = scriptFile, object = scriptFile, bucket = selectedBucket)

put_object(file = fileName, object = fileName, bucket = selectedBucket)
```

9.3 Appendix C

Table 18: Benchmark results of the single node configuration

Benchmark run	**Time**
1	22.026
2	21.983
3	22.1
4	22.058
5	21.551
6	21.552
7	21.643
8	21.713
9	21.532
10	21.562
11	21.669
12	21.495
13	21.843
14	21.593
15	21.483
16	21.571
17	21.685
18	21.605
19	21.432
20	22.162
21	21.822
22	21.466
23	21.577
24	21.418
25	21.981
26	21.824
27	22.0559999999999
28	22.0169999999999
29	21.9770000000001
30	22.106
31	22.1610000000001
32	22.261
33	22.1940000000001
34	21.7520000000001
35	21.6469999999999
36	21.894
37	21.636
38	21.8900000000001
39	21.984
40	22.183
41	22.165

42	22.054
43	22.0119999999999
44	22.073
45	21.904
46	22.028
47	21.96
48	21.943
49	22.0330000000001
50	22.0639999999999
51	21.9360000000001
52	21.9010000000001
53	22.027
54	21.9440000000002
55	22.1410000000001
56	22.3599999999999
57	22.165
58	22.297
59	21.9110000000001
60	21.8969999999999
61	21.8979999999999
62	21.96
63	21.894
64	22.068
65	21.932
66	22.058
67	21.9159999999999
68	21.867
69	21.9829999999999
70	21.925
71	21.7180000000001
72	21.867
73	21.704
74	21.837
75	21.6229999999998
76	21.5809999999999
77	21.6279999999999
78	22.095
79	21.8150000000001
80	21.8180000000002
81	21.9290000000001
82	22.1130000000001
83	21.723
84	21.9440000000002
85	21.8409999999999
86	21.8209999999999

87	21.8340000000001
88	21.9559999999999
89	21.9849999999999
90	21.884
91	21.7570000000001
92	22.212
93	21.895
94	21.5840000000001
95	21.71
96	21.8210000000004
97	21.48
98	21.6729999999998
99	21.6769999999997
100	21.9989999999998
101	21.5140000000001
102	21.6419999999998
103	21.569
104	21.6959999999999
105	21.694
106	21.5360000000001
107	21.7739999999999
108	21.721
109	21.5459999999998
110	21.6859999999997
111	21.6680000000001
112	21.5810000000001
113	21.71
114	21.703
115	21.6160000000004
116	21.7530000000002
117	21.5709999999999
118	21.7709999999997
119	21.5729999999999
120	21.806

9.4 Appendix D

Table 19: Benchmark results of the multi core configuration

Benchmark run	**Time**
1	41.704
2	41.325
3	41.552
4	41.762
5	41.446
6	41.851
7	41.527
8	41.81
9	41.612
10	41.323
11	41.542
12	41.768
13	41.267
14	41.7589999999999
15	41.434
16	41.5999999999999
17	41.537
18	41.372
19	41.4659999999999
20	41.4150000000001
21	41.5409999999999
22	41.66
23	41.4019999999999
24	41.62
25	41.588
26	41.4360000000001
27	41.4190000000001
28	41.4300000000001
29	41.5410000000002
30	40.682
31	41.635
32	41.234
33	41.411
34	41.602
35	41.286
36	41.681
37	41.382
38	41.671
39	41.457
40	41.187
41	41.373

42	41.596
43	41.123
44	41.582
45	41.352
46	41.5649999999999
47	41.518
48	41.3539999999999
49	41.447
50	41.3919999999999
51	41.519
52	41.619
53	41.364
54	41.603
55	41.586
56	41.373
57	41.4150000000002
58	41.3799999999999
59	41.5170000000001
60	41.731
61	43.395
62	42.936
63	43.29
64	43.545
65	43.178
66	43.538
67	43.305
68	43.465
69	43.273
70	43.005
71	43.221
72	43.4329999999999
73	42.954
74	43.462
75	43.135
76	43.371
77	43.294
78	43.138
79	43.242
80	43.1080000000001
81	43.29
82	43.439
83	43.152
84	43.352
85	43.3879999999999
86	43.2059999999999

87	43.173
88	43.204
89	39.9300000000001
90	25.5910000000001
91	43.463
92	43.049
93	43.4
94	43.639
95	43.266
96	43.648
97	43.428
98	43.596
99	43.429
100	43.097
101	43.358
102	43.583
103	43.072
104	43.5889999999999
105	43.237
106	43.4880000000001
107	43.399
108	43.261
109	43.38
110	43.245
111	43.393
112	43.5809999999999
113	43.265
114	43.4950000000001
115	43.4690000000001
116	43.337
117	43.3099999999999
118	43.3230000000001
119	37.3310000000001
120	26.5400000000002

9.5 Appendix E

Table 20: Benchmark results of the multi node configuration

Benchmark run	Time
1	23.581
2	23.578
3	23.452
4	23.541
5	23.434
6	23.566
7	23.447
8	23.577
9	23.47
10	23.544
11	23.485
12	23.532
13	23.528
14	23.525
15	23.532
16	23.518
17	23.446
18	23.503
19	23.52
20	23.518
21	23.525
22	23.52
23	23.522
24	23.504
25	23.4949999999999
26	23.551
27	23.481
28	23.513
29	23.475
30	23.4699999999999
31	23.524
32	23.538
33	23.362
34	23.509
35	23.38
36	23.492
37	23.355
38	23.488
39	23.351
40	23.443
41	23.375

42	23.493
43	23.4
44	23.387
45	23.392
46	23.425
47	23.417
48	23.468
49	23.423
50	23.43
51	23.417
52	23.435
53	23.3720000000001
54	23.429
55	23.452
56	23.489
57	23.4399999999999
58	23.427
59	23.342
60	23.454
61	22.509
62	22.441
63	22.328
64	22.491
65	22.424
66	22.478
67	22.434
68	22.481
69	22.354
70	22.413
71	22.305
72	22.482
73	22.432
74	22.364
75	22.405
76	22.418
77	22.443
78	22.417
79	22.486
80	22.378
81	22.351
82	22.456
83	22.424
84	22.428
85	22.413
86	22.447

87	22.348
88	22.4069999999999
89	22.4639999999999
90	22.389
91	23.044
92	22.978
93	22.889
94	22.974
95	22.892
96	22.952
97	22.895
98	23.011
99	22.988
100	23.016
101	22.899
102	23.017
103	22.977
104	22.927
105	23.013
106	22.934
107	22.931
108	22.936
109	22.994
110	22.965
111	22.877
112	23.008
113	22.916
114	22.9359999999999
115	22.9880000000001
116	22.961
117	22.942
118	23.0039999999999
119	22.92
120	22.909

9.6 Appendix F

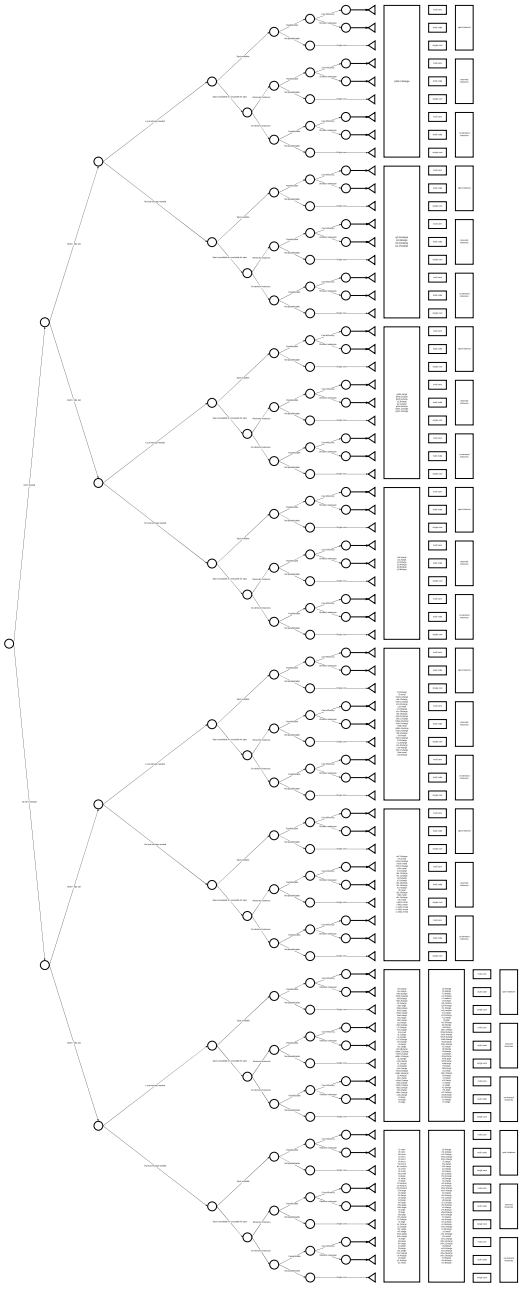
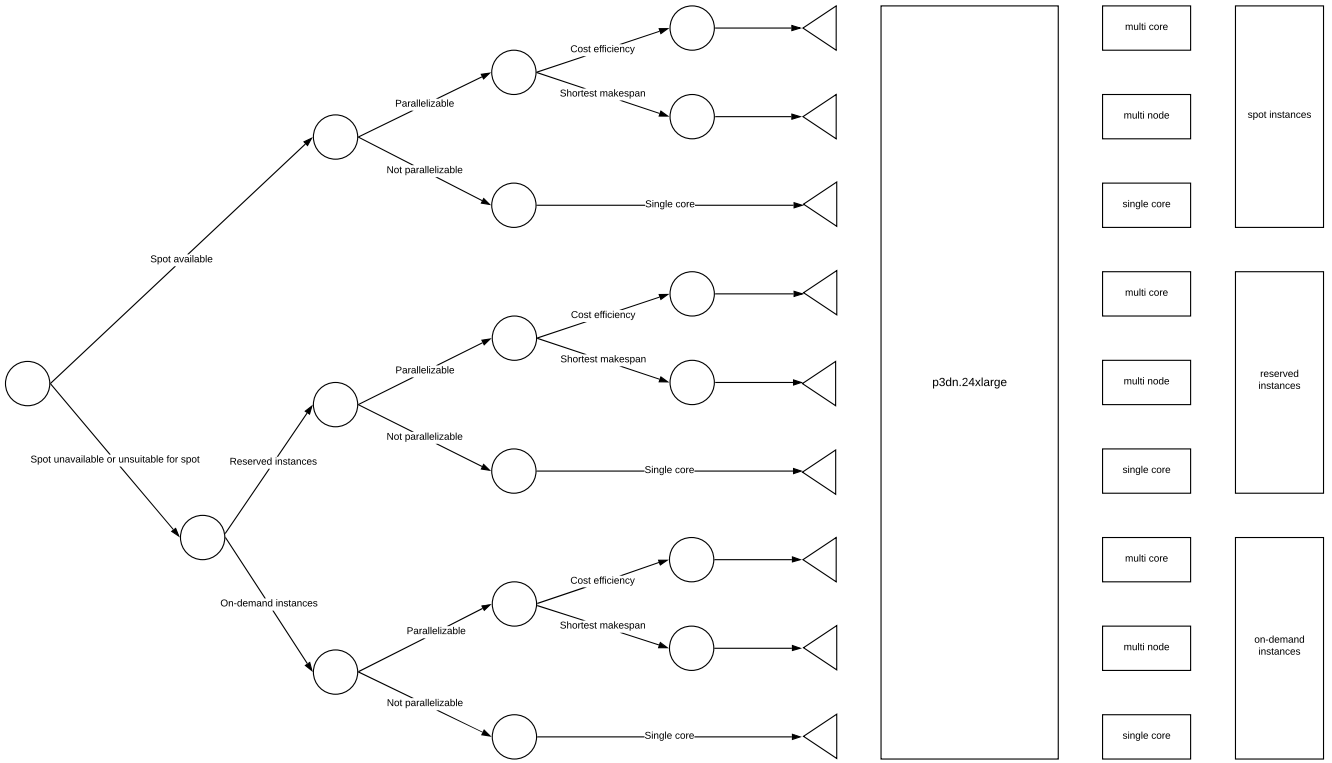


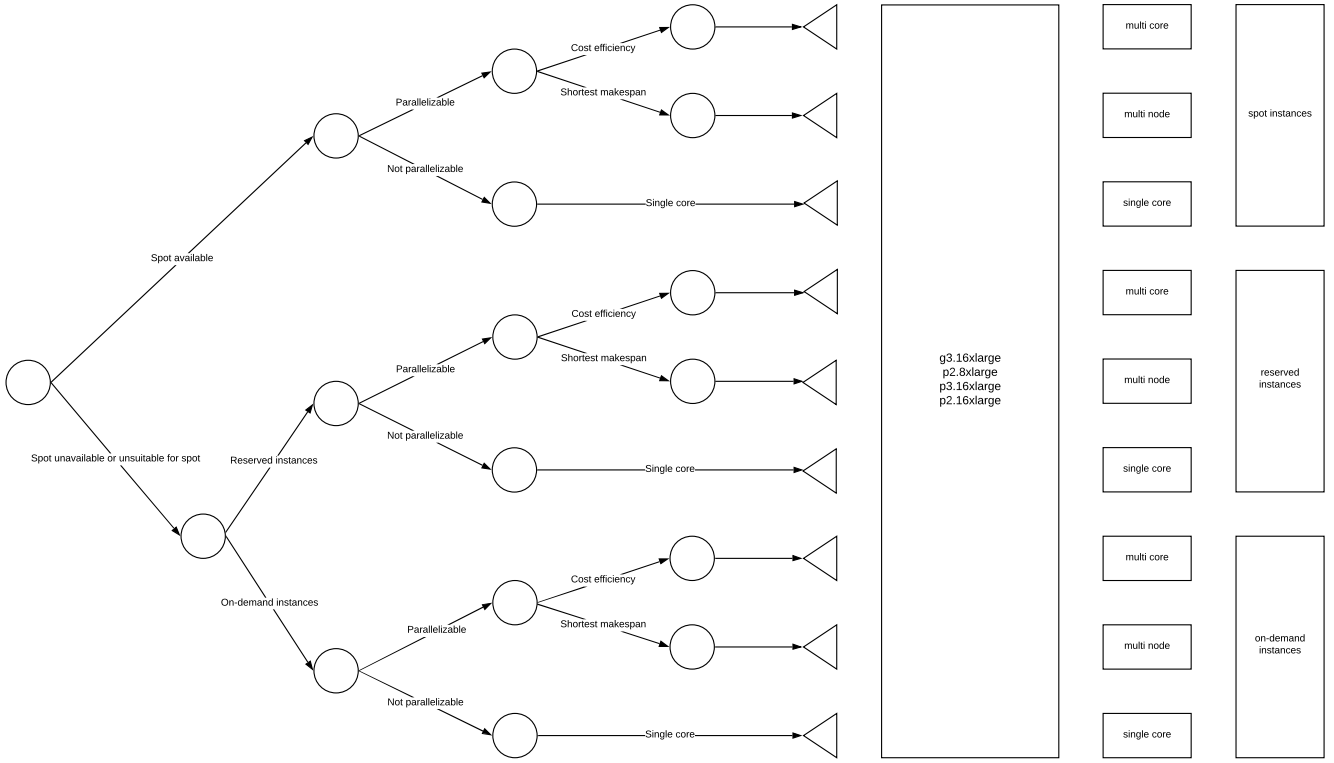
Figure 21: Complete decision tree

9.7 Appendix G

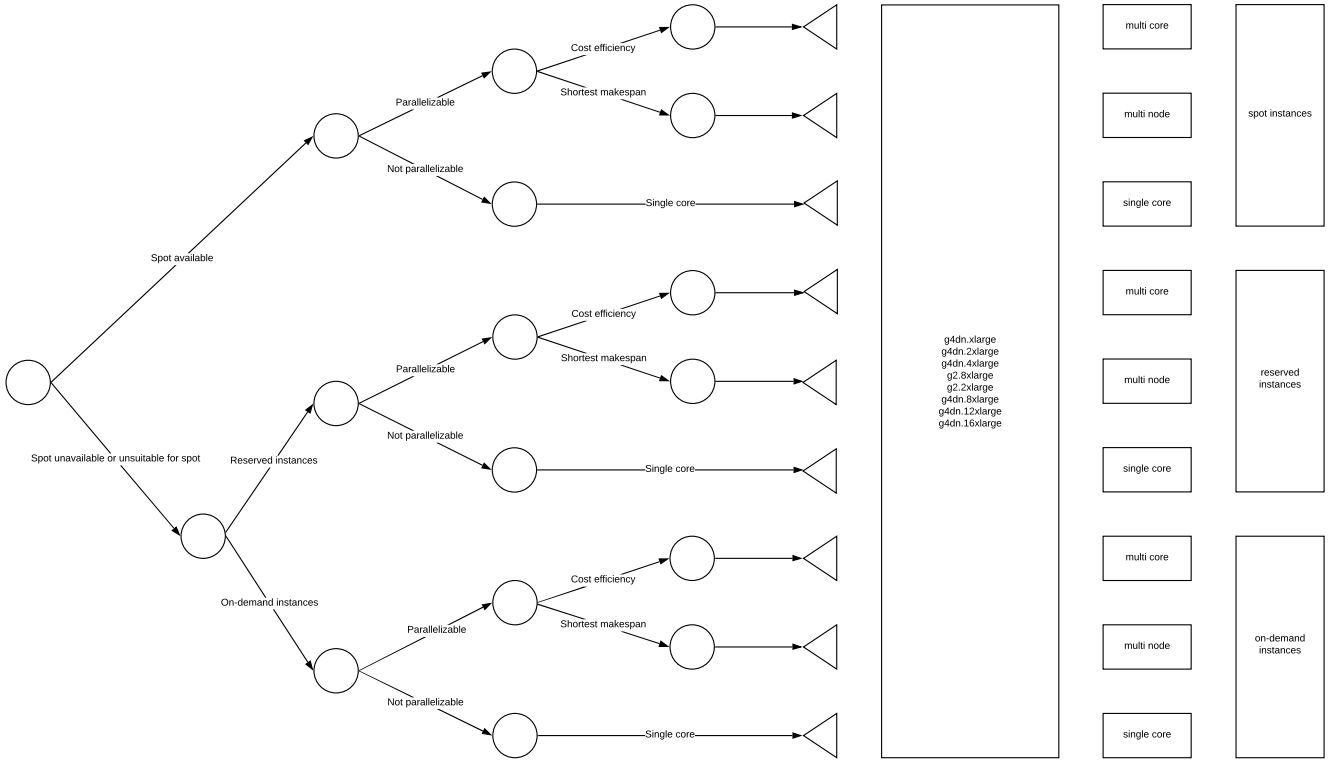
9.7.1 GPU enabled, >384GB RAM, local storage



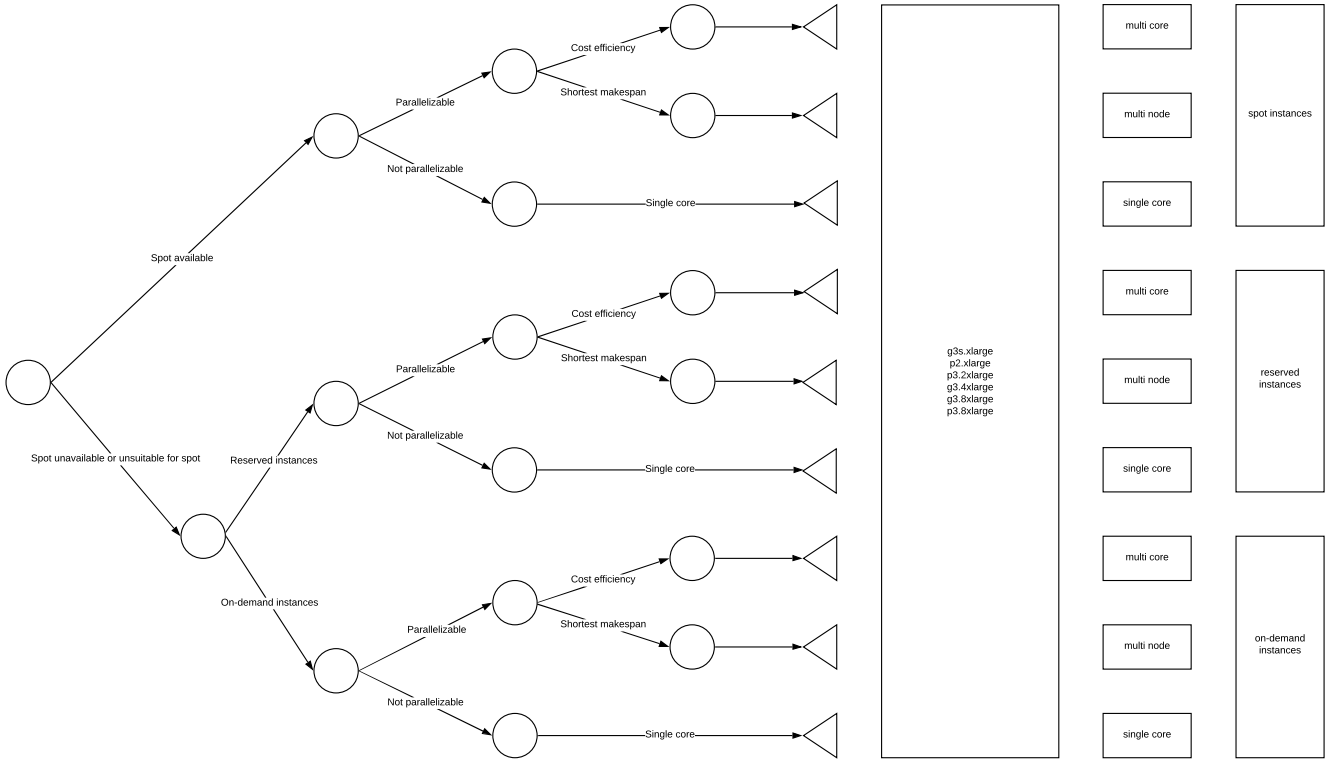
9.7.2 GPU enabled, >384GB RAM, external storage



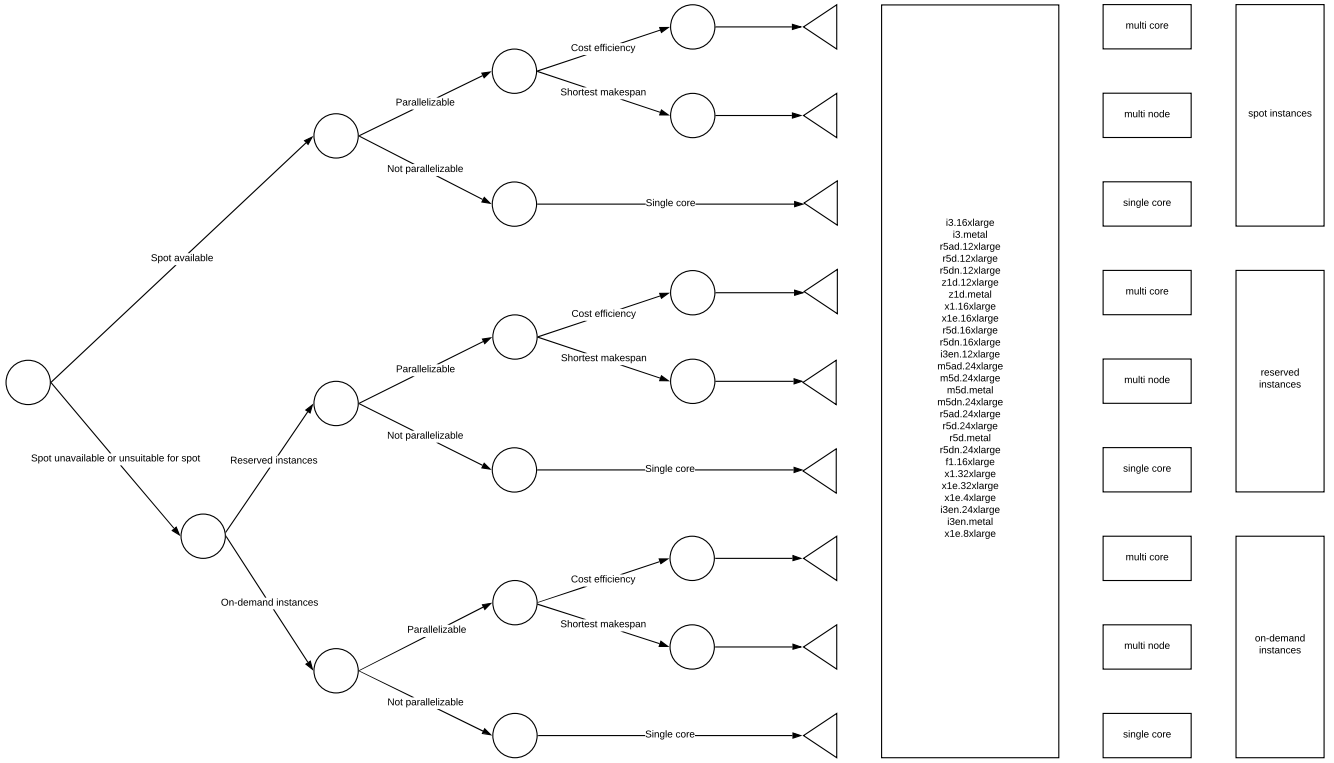
9.7.3 GPU enabled, <384GB RAM, local storage



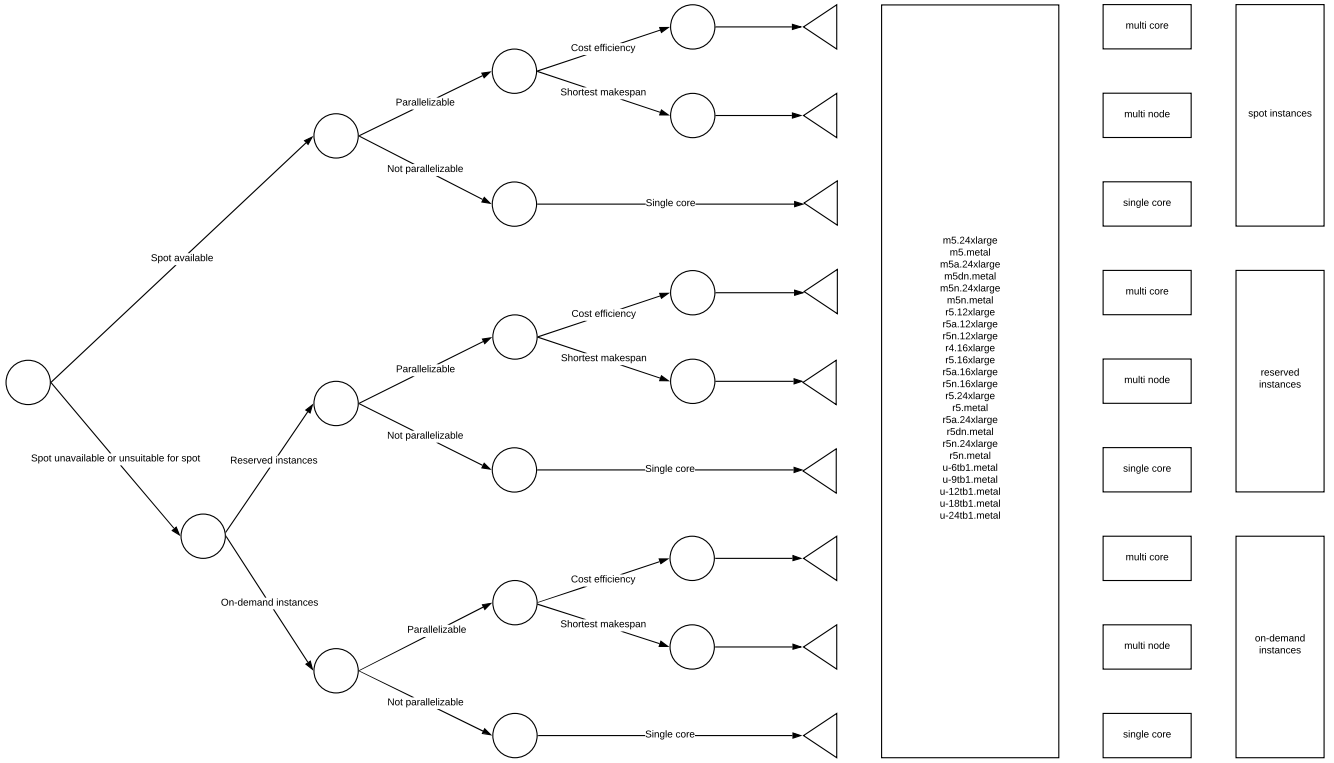
9.7.4 GPU enabled, <384GB RAM, external storage



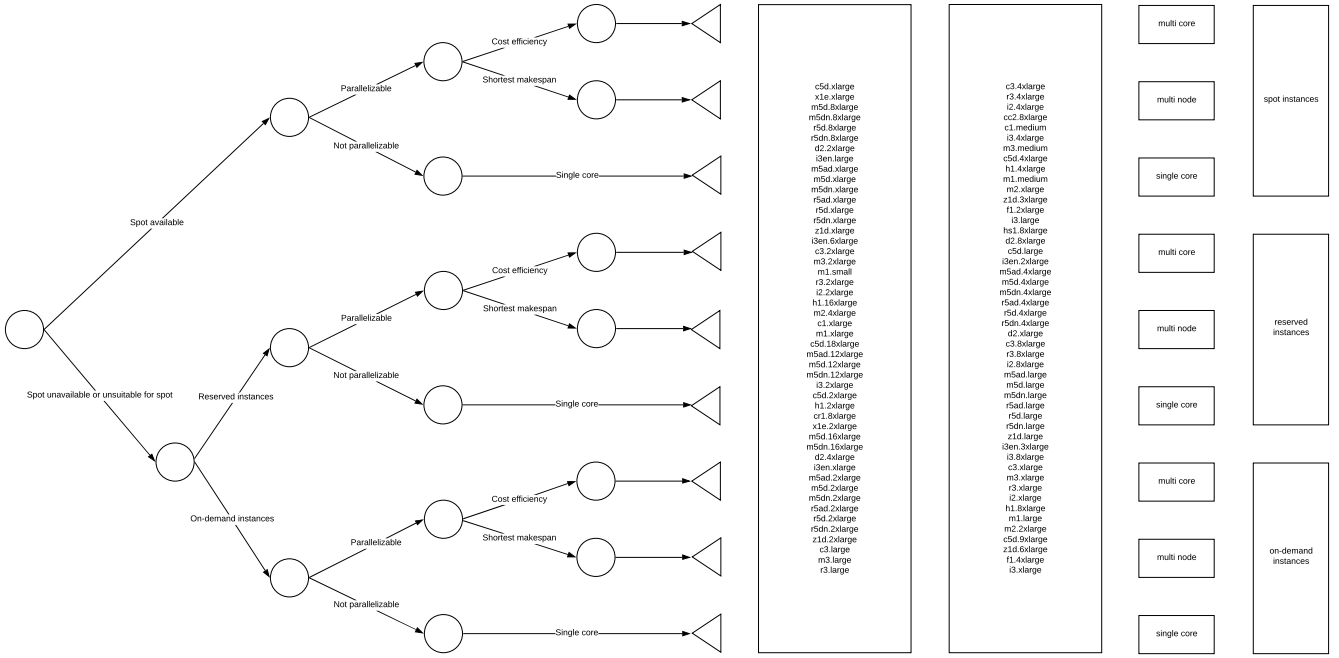
9.7.5 >384GB RAM, local storage



9.7.6 >384GB RAM, external storage



9.7.7 <384GB RAM, local storage



9.7.8 <384GB RAM, external storage