Performance tuning Apache Drill on Hadoop Clusters with Genetic Algorithms

Improving industry standards for advanced analytics and business intelligence

Roger Bløtekjær

Thesis submitted for the degree of
Master of science in Informatikk: språkteknologi
30 credits

Institutt for informatikk
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

Performance tuning Apache Drill on Hadoop Clusters with Genetic Algorithms

Improving industry standards for advanced analytics and business intelligence

Roger Bløtekjær

Performance tuning Apache Drill on Hadoop Clusters with Genetic Algorithms

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

0.1 Summary

0.1.1 Research question

How can we make a self optimizing distributed Apache Drill cluster, for high performance data readings across several file formats and database architectures?

0.1.2 Overview

Apache has introduced a new building block that's best suited on top of the Hadoop Stack called Apache Drill. Drill enables the user to perform schema-free querying of distributed data, and ANSI SQL programming on top of NoSQL datatypes like JSON or XML. As it is with the core Hadoop stack, Drill is also highly customizable, with plenty of performance tuning parameters to ensure optimal efficiency. Tweaking these parameters however, requires deep domain knowledge and technical insight, and even then the optimal configuration may not be evident. Businesses will want to use Apache Drill in a Hadoop cluster, without the hassle of configuring it, for the most cost-effective implementation. This can be done by solving the following problems:

- How to apply genetic algorithms in the most cost-effective way to automatically tune a distributed Apache Drill configuration, regardless of cluster environment.
- How do we benchmark the cluster against default Drill settings, as well as known SQL performance tests, to ensure that the algorithm is adding value to the cluster performance, measured as execution time.

0.1.3 Details

Self optimizing

The goal here is that no human intervention is needed to fine tune the Drill configuration. Once the framework / algorithm is executed on the Drill environment, it should only end execution when performance is proven to be optimized.

Genetic Algorithms

The methodology for self optimization in this thesis is genetic algorithms. A branch of evolutionary algorithms that rely on randomly distributed properties given to a population of candidates. The properties are hereditary and define the next generation of candidates, eventually resulting in a "proven optimal candidate", to solve a given problem.

Distributed file systems / cluster environments

Apache Drill can easily be run on any file system, as they are mounted when needed. As such a single node environment is entirely possible to set up, with the local file system mounted as a "Drill Bit". However this thesis focuses on current and future industry standards, and how to optimize Drill in a relevant

a big data environment. This is more representative of its intended use, and gives the thesis more value as a research field advancement.

ANSI SQL

There are similar platforms to Apache Drill, like the popular Apache Spark. Spark also provides SQL querying on data, however this is only a subset of SQL. Only Drill has the full suit of SQL programming capabilities as defines by ANSI (American National Standards Institute), which gives more in depth querying.

NoSQL

Previously NoSQL databases where known as "Non-SQL" databases, but this has now changed to be "Not Only SQL" databases as they have begun supporting SQL-like querying. The main point about them is that they are non-relational, distributed, open-source and horizontally scalable.[27]

0.2 Foreword

I got nothing. Put this on a different page though, maybe with a dramatic font or something.

Contents

Chapter 1

Preface

1.1 List of figures

1.2 List of tables

1.3 Word list & abbreviations

Table 1.1: Technical terms

Load profiling	Explanation1
TTE	Explanation2
OOP	Explanation3
HDFS	Explanation4
PBIL	Explanation5
GA	Explanation6

Chapter 2

Introduction

2.1 Target group

This thesis covers the deep technical aspects of big data analysis and genetic algorithms - However, all techniques used will be explained in detail. Computer science students looking into infrastructure will probably feel the most at home, in terms of discussed concepts and technical terms. For the layman, there is also a word list supplied with brief explanations for technical terms used in this thesis.

2.2 Area of research

Hadoop and MapReduce are already well established technologies employed in countless applications around the world, Apache Drill however is a fairly new concept with little to no research from the community. We propose a new method of implementing Hadoop clusters with Apache Drill, automatically optimizing the performance tuning in a lightweight and low effort way. As such this is a technical thesis regarding performance tuning, and the implementation aspects of an Apache Drill cluster.

2.3 Personal motivation

Subject

The subject for this master thesis is a natural continuation of our previous work *Hadoop MapReduce Scheduling Paradigms*, published in 2017, in the 2nd IEEE International Conference on Cloud Computing and Big Data Analysis (ISSS-BDA 2017). Back then the topic was haphazardly picked from a list of eligible ones, but the more we read into it - the more we understood the incredible use cases for Hadoop within the massive industries that are driving forces for our technological advancements. As is common in IT, levels of abstraction get added on top of proven technologies both to make implementation better, and often to increase performance. Since then Apache Drill has seen plenty of stable builds and proven itself to be potentially industry-changing in the way we handle our data - entering a schema-on-the-fly paradigm.

Apache Drill as a platform

As mentioned previously there is little to no research done on Apache Drill as a platform. Some of the reasoning might be the that some still considers it to be in the early development phase, but as of this writing there has already been one major release, **1.0** in 2015, and 12 subsequent minor releases up till **1.12**.[18] So we would argue the project is past the early development phase, and moving into the early adopter phase. But why choose to use this platform over more established ones, and what other options are there? This is discussed further in a following section: Why is this worthwhile?

2.4 Research method in brief

Throughout this thesis we will develop an entire suite of tools centered around a genetic algorithm for automatically optimizing Apache Drill on top of a Hadoop cluster, tested on big and diverse sample sets. As the framework is expected to change, all of the parameters tweaked will be in relation to the current version being used, **1.12**. The main goal is to achieve a performance increase, measured as TTE (Time to Execute) for single queries. Examining previous studies on SQL performance will provide a good basis for best practices within SQL query execution optimization, and how to measure it.

2.5 Most relevant previous findings

There is little to no research done on the impact of tuning Apache Drill. However, tuning of parameterized frameworks have been done a lot in the past on industry standards like SQL, MySQL, traditional Hadoop clusters (mapreduce scheduling algorithms), and Web applications. Since Apache Drill has its own SQL execution engine, the tuning of SQL systems in previous works has value for how we will benchmark our Drill perfomance.

2.5.1 Starfish

Herodotos et al made on of the most cited papers in the Hadoop research field, when they proposed Starfish [5]. Starfish is a self-tuning system for Hadoop clusters, citing the lackluster performance of default cluster parameters to be the motivation. They designed a modular system where the main parts include:

- A profiler that analyzes jobs and determines cost estimation and the data flow.
- A novel approach to predictive tuning they named the "what-if-engine". It's job is to predict how different parameter configuration tweaks will change the performance of the system.
- A cost-based optimizer, that performs the pure tuning aspects, based on estimations from the what-if-engine.

Our project has very similar goals and motivations compared to this project, except on a framework on top of Hadoop, instead of directly on it.

HAVING TROUBLES HERE....

Other types of performance tuning: Hadoop, Web, MySQL etc.

2.6 Why is this worthwhile, why Drill?

2.6.1 Ease of use

It is safe to say that information technology as a subject is only getting more popular by demand. According to projections made by renown recruitment company Modis, tech employment will see a 12% growth by 2024, vs 6,5% in all other industries[13]. This means that a lot of new engineers will enter the workforce, and start building and maintaining enterprise applications. Apache Drill serves a very low barrier of entry for newcomers with its' SQL ANSI queries when handling big amounts of data, as opposed to i.e the popular Java Enterprise framework Persistence. Using Apache Drill will therefore require less training, thus increasing productivity and agility in young teams.

2.6.2 Increasing relevance of Big Data infrastructures

"From 2005 to 2020, the digital universe is expected to grow dramatically by a factor of 300, from 130 exabytes to 40 trillion gigabytes, i.e more than 5,200 gigabytes per person in 2020. Moreover, the digital universe is expected to double every two years."[22]

Real time data usage

Successful digitalization of businesses often require applications to utilize data in real time, serving customers relevant data instantaneously. *"Now, most applications are generating real-time data, so superfast data capturing and analysis within a fraction of a second are mandatory."*[24]
An interesting example of this is chat applications (instant messaging). Users of messaging apps today expect it to store data indefinitely, and deliver instant communication between users, creating an absolutely enormous global data flow. Bettercloud.com performed a study and found that[20]:

- The majority of organizations (57%) use two or more real-time messaging applications.
- 80% of Skype for Business users, 84% of Google Hangouts users, and 95% of Slack users say communication has improved because of real-time messaging.
- 56% of respondents believe that real-time messaging will displace email as their organization's primary workplace communication and collaboration tool.

From these results we see that adoption of applications using real time data is only increasing. This will provide a future need for fast and scalable infrastructure to cope with it.

Combining real-time and stored data

In reality a lot of current applications that create value combine the usage of analytics on stored data and real-time sensor data. Predictive maintenance is an example of a discipline where you analyze historical data, finding thresholds and patterns for when components in a system break down, and then combining this knowledge with real time sensors to service the components before they fail. This saves money on planned maintenance where healthy parts get serviced, and downtime on systems where they have to be set aside for the service. In a Harvard Business Review on a company called **Servicemax** that has specialized in services like this they wrote that: *"(...) customers, on average, increase productivity by 31%, service revenue by 14% and customer satisfaction by 16%."*[23] All of this is empowered by big data, and the utilization of information.

2.6.3 Application needs

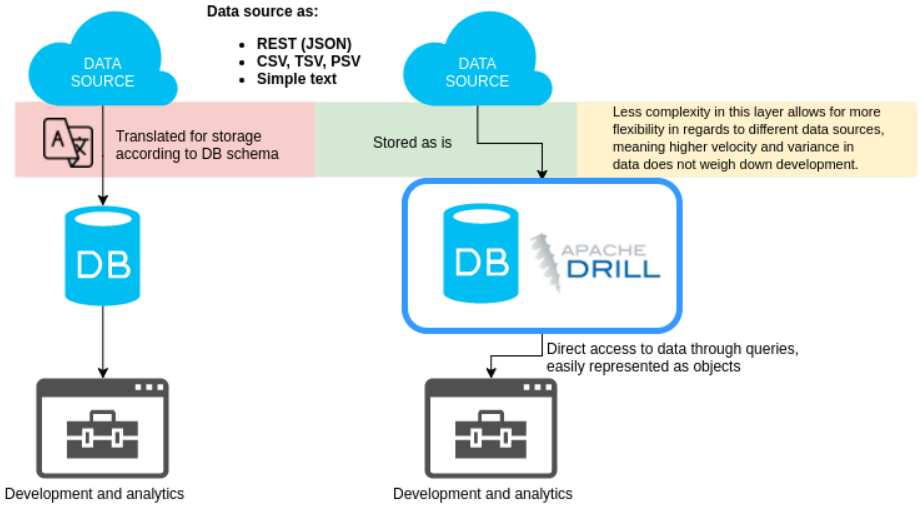

Figure 2.1: Advantages of schema-free querying

"Object and relational technologies are grounded in different paradigms. Each technology mandates that those who use it take a particular view of a universe of discourse. Incompatibilities between these views manifest as problems of an object-relational impedance mismatch."[19]
Traditional databases contain tables (entities) with relations between them. Sometimes these systems consist of thousands of tables, even going as far as tens of thousand (like this database of the human genome with 20000 tables[21]). Object-oriented programming classes is difficult and oftentimes inefficient to map directly to the raw data. Especially when there are large amounts of data, it can take a while just initializing the objects, and then again when serializing them to store them. Adopting complex storage file types like JSON for raw data would overcome this mismatch, and allow for more agile, and reactive development of enterprise applications utilizing huge data sources. The global IT industry is increasingly transitioning to deliver RESTful APIs to their

customers. With Apache Drill, it is extremely easy to perform queries on the results, even joining them against several other data sources or combining results from several APIs at the same time, in real time. For application developers this will eliminate the middleware of a relational database, having to translate the data for different purposes. Instead of having one format for storing, one for interfacing and one for application logic, Apache Drill and the schema-free paradigm will provide one single interface for all dimensions of data.

2.7 How far will this advance the field?

Advances in research

The ambition for this project is to provide a fully functional, open source light weight framework allowing companies to easily deploy Hadoop Clusters with Apache Drill without worrying about tailoring the solution or suboptimal performance. In terms of research, this is the first thesis written about performance tuning of Apache Drill, and as such will lay a foundation for the future of this field.

Advances in industry

As the previous section highlighted (Increasing relevance of Big Data infrastructures), we predict an increase in future global data collection, going as far as doubling the amount of data per person, per year. This data is only collected to add value to a business, and the revenue increase for businesses that embrace big data analytics will not go unnoticed by the industry, leading to wider adoption. Some businesses already make huge profits simply by gathering and selling information, like Google and Facebook, increasing their yearly revenue since 2011 by 289% and 1095% respectively[26]. Data is becoming the world's new natural resource[25]. To be able to more fluidly handle this data in applications, we believe agile teams will want flexible frameworks that empower more cost effective development and big data utilization. When the industry realizes Apache Drill can deliver that, this will lead to a paradigm shift to schema-free querying of data, and on-the-fly data reads without tailoring.

2.8 Structure of the report

Add comments about every chapter here.

Chapter 3

Background

"Apache Drill is one of the fastest growing open source projects, with the community making rapid progress with monthly releases. The key difference is Drill's agility and flexibility. Along with meeting the table stakes for SQL-on-Hadoop, which is to achieve low latency performance at scale, Drill allows users to analyze the data without any ETL or up-front schema definitions. The data can be in any file format such as text, JSON, or Parquet. Data can have simple types such as strings, integers, dates, or more complex multi-structured data, such as nested maps and arrays. Data can exist in any file system, local or distributed, such as HDFS or S3. Drill, has a "no schema" approach, which enables you to get value from your data in just a few minutes."[16]

3.1 History

3.1.1 Creation and adoption of Hadoop

First, Google created the Google File System in 2003[10], predicting the paradigm of distributed storage. Then, the MapReduce concept for sorting / counting data was added in 2004[11]. Hadoop was introduced as a platform in 2006[8], and together with MapReduce, it made an impact in the industry, attracting talent and big companies eager to contribute and deploy. Yahoo was a big early adopter, being one of the first companies deploying big clusters as early as 2006[8]. Then in 2008, Hadoop got the world record for fastest system to sort a terabyte of data[8]. After this, development accelerated with more contributers and interest was at an all time high. Since that point in time Apache Hadoop has become the most widely used platform for Big Data handling, empowering advanced analytics and business intelligence across several industries. The Hadoop stack is now highly scalable, ensures high availability of data, and utilizes parallel processing to deliver high performance data readings.

3.1.2 Google branching out with Dremel

In 2010 Google did further research on distributed data processing and published their paper on Dremel[9]. Dremel is the framework that empowers Google's current platform Google BigQuery, delivered as Infrastructure as a Service (IaaS). Among customers of BigQuery is well recognized brands like Spotiy, Coca Cola,

Philips, HTC and Niantic[12]. The success of Google BigQuery (by extension of Dremel), this inspired Apache to create Drill, the framework being examined in this thesis.

3.2 Genetic algorithms

Genetic algorithms are higher level procedures for generating high-quality configurations / solutions to problems with a wide range of parameters, typically where exhaustive searches are infeasible. The procedure starts out with a population of n candidates, each representing a set of pseudo-randomly generated values for the respective parameters (properties) that will yield a result for the given problem. Once each candidate has attempted to solve the problem by applying their properties to it, a fitness score is given and compared amongst them. The fitness score represents how good their solution to the problem was, and lets us pick out the best candidates among our population. After the optimal candidates of a generation is chosen, a new generation is generated, inheriting the properties of the previous best candidates. This process repeats until a stop criterion is reached and the properties of the candidate is considered an optimal solution. The number of generations to be generated, and the number of parent candidates from which the next generation is generated from are tweakable parameters.

3.2.1 NASA applying genetic algorithms

A very famous application of genetic algorithms is the high-performance antenna NASA made, that actually flew in their Space Technology 5 (ST5) mission[15]. In NASAs case, they needed to make a highly receptive antenna with very small physical dimensions. It could have any number of branches, each going in any arbitrary direction. Because of these seemingly infinite possible configurations for the antenna, applying genetic algorithms to gradually evolve a design by testing randomly generated populations and combining the best traits from each candidate, proved to be a great way of solving this. *"(...) the current practice of designing antennas by hand is severely limited because it is both time and labor intensive and requires a significant amount of domain knowledge, evolutionary algorithms can be used to search the design space and automatically find novel antenna designs that are more effective than would otherwise be developed."*[15]

3.2.2 Limitations of genetic algorithms

Fitness function

At first glance it sounds like applying genetic algorithm techniques to any problem would yield an optimal solution, with relative ease. The problem though, is the fitness function for determining the candidate solution. As the complexity of the problem scales in size the search space for the fitness function will end up being too big to complete in a reasonable time. For instance if one single candidate evaluation took days, performed for the whole population, across generations, it could end up taking months completing just one single optimization. In those cases machine learning approaches or simply manual labor through technical and domain specific knowledge would prove to be more efficient.

Reaching the optimal solution

The only way to evaluate a solution is to compare it against other solutions within the population, all of which have are derived from random mutations. Therefore it is impossible to know whether a truly optimal solution is reached, or if one more generation of mutations will prove to give a better result. In practice a stop criterion for the candidate generation is therefore needed, where a solution is considered to be optimal.

3.3 Population based incremental learning

The distinction between PBIL and the general GA lies in where the evolution happens. GA generally focus on evolving individual candidates, and crossing fit parents to produce extra fit children, each of them with a chance of mutation. Across several generations convergence is expected, but there is always a chance that the next mutation will lead to the considered optimal solution. PBIL on the other hand evolves the whole population on a higher level, where the probability vector for parameter generation is shifted towards a considered favorable state. So instead of generating new candidates defined as a crossover function of two parents, the fittest candidates affect a shared genotype that each candidate is generated from. This allows for a much clearer convergence of properties, where you end up in a state where every parameter has a (close to) 100% chance to be generated. This approach was first proposed back in 1994 by Baluja, where he found that *"The results achieved by PBIL are more accurate and are attained faster than a standard genetic algorithm, both in terms of the number of evaluations performed and the clock speed. The gains in clock speed were achieved because of the simplicity of the algorithm."*[28].

3.3.1 Advantage of PBIL

The main advantage PBIL supplies for this problem is the avoidance of unnecessary mutation on key properties. In the case of Apache Drill there are a number of boolean properties that have big impacts on performance, i.e "planner broadcast join" that based on setting can result in a query that is *"(...) substantially cheaper"*[17]. The impact of flipping this boolean property is based on the type of query being done, the skewness of data locality and cluster size, but for a specific type of query in a set environment the property will always have a strictly optimal value. In PBIL this will become apparent early, and the chance of this value being anything else than the optimal one is likely decreased instantly after first generation run, and in every subsequent run. Whereas in traditional GA there is always a given chance for each generation iteration for this value to mutate in a candidate, essentially wasting resources by running a test with it.

3.3.2 Limitation of PBIL

The main limitation of PBIL is that each candidate property needs a preset of valid values. Thus finding the optimal solution is still based widely on the amount of options, and the effectiveness of them, defined by the developer of the algorithm. Naturally - in terms of boolean properties there lies no challenge. However when linear parameters need to be set, i.e properties with ranges from

zero to infinity, the developer must define a set of valid options existing within that range. In the context of Apache Drill there are default rule of thumb values for each property, so the list of valid options for each property is generated around the default. Another drawback of PBIL lies in the risk of premature convergence. This can however be mitigated by effective mutation functions that ensure diversity.

3.4 Related literature and theoretical focus

3.4.1 Performance tuning MapReduce as a Research Field

There is a ton of research to be read about MapReduce optimization, trying to tune map- and reduce-slots, and improve the inherent job tracker. Looking at a few essential papers within this field shows a general consensus that tuning default performance parameters in a variety of environments will lead to 10-30% performance increase, which naturally results in considerable cost savings. For instance Min Li et. al. could report that "*Our results using a real implementation on a representative 19-node cluster show that dynamic performance tuning can effectively improve MapReduce application performance by up to 30% compared to the default configuration used in YARN.*"[1] Furthermore, manually tuning these clusters are too demanding for most businesses to even consider, requiring both deep technical and domain-specific insight. These findings further maintain our vision of bringing auto-tuning to the Apache Drill framework, which we believe to be the next natural step forwards, even paradigm-defining, for big data handling.

3.4.2 Mapreduce optimization

Gunther

Gunther evaluated methods for optimizing Hadoop configurations using machine learning and cost-based models, but found them inadequate for automatic tuning. Thus they introduced a search-based approach with genetic algorithms, designed to identify crucial parameters to reach near-optimal perfomance of the cluster.[2] This paper tells us that genetic algorithms as an approach to performance tune big data clusters already is a proven method. However, the scope is set to Hadoop as an out-of-the-box enterprise solution, whereas we take the next step within the schema-on-the-fly paradigm of Apache Drill.

mrOnline

"MapReduce job parameter configuration significantly impacts application performance, yet extant implementations place the bur- den of tuning the parameters on application programmers. This is not ideal, especially because the application developers may not have the system-level expertise and information needed to select the best configuration. Consequently, the system is utilized inefficiently which leads to degraded application performance."[1]

3.4.3 Other Hadoop SQL engines

There are plenty of SQL-on-hadoop engines available right now, like CitusDB, Impala, Concurrent Lingual, Hadapt, InfiniDB, JethroData, MammothDB, Apache Drill, MemSQL or Pivotal HawQ, and the amount of independently developed engines speaks volumes of the interest in this field. Why Apache Drill was chosen in this thesis is detailed in the Why Drill section, but the fact that Drill is (at the time of this writing) the only schema-free engine is a major selling point. However Impala and Spark are often considered as alternatives in similar environments.

3.4.4 Impala

Impala, developed by Apache is an opensource SQL engine that was designed to bring parallel DBMS technology to the Hadoop environment[29]. The main idea is shared between Drill and Impala where they both set up daemons on each node of a distributed cluster, to perform MPP (Massively Parallel Processing) for data reads, circumventing MapReduce. Impala is made for Big Data Analyics, priding itself in having *"order-of-magnitude faster performance than Hive"*[30].

3.4.5 Spark

Apache also developed Spark, a popular open-source platform for large-scale data processing that is well-suited for iterative machine learning tasks[31]. Spark is built with Hive, and runs very similarly to MapReduce but also has extended functionality. The key difference is that Spark keeps things in memory, while MapReduce keeps shuffling data in and out of disk. This allows Spark to facilitate machine learning algorithms on top of big data clusters, reading and processing data in the same workflow.

Chapter 4

Presentation of domain where technology is used

BMC Software, Inc states the following about Apache Hadoop:
Financial services companies use analytics to assess risk, build investment models, and create trading algorithms; Hadoop has been used to help build and run those applications. Retailers use it to help analyze structured and unstructured data to better understand and serve their customers. In the asset-intensive energy industry Hadoop-powered analytics are used for predictive maintenance, with input from Internet of Things (IoT) devices feeding data into big data programs. Telecommunications companies can adapt all the aforementioned use cases. For example, they can use Hadoop-powered analytics to execute predictive maintenance on their infrastructure. Big data analytics can also plan efficient network paths and recommend optimal locations for new cell towers or other network expansion. To support customer-facing operations telcos can analyze customer behavior and billing statements to inform new service offerings. Examples of Hadoop There are numerous public sector programs, ranging from anticipating and preventing disease outbreaks to crunching numbers to catch tax cheats. Hadoop is used in these and other big data programs because it is effective, scalable, and is well supported by large vendor and user communities. Hadoop is a de facto standard in big data.[3]

Chapter 5

Method

5.1 Technology

5.1.1 Hadoop stack

Hadoop common

Hadoop common consists of a few select core libraries, that drives the services and basic processes of Hadoop, such as abstraction of the underlying operating system and file system. It also contains documentation and Java implementation specifications.

HDFS

HDFS (Hadoop Distributed File System) is a highly fault tolerant, distributed file system designed to run efficiently, even on low-cost hardware. HDFS is tailor made to express large files and huge amounts of data, with high throughput access to application data and high scalability across an arbitrary amount of nodes.

YARN

YARN (Yet Another Resource Negotiator) is actually a set of daemons that run in the cluster to handle how the jobs get distributed.

- There is a NM (NodeManager) that represents each node in the cluster, monitoring and reporting to the RM whether or not they are idle, and resource usage (CPU, memory, disk, network). A node in a Hadoop cluster divides its resources into abstract Containers, and reports on how many containers there are available for the RM to assign jobs to.
- There is an AM (ApplicationMaster) per job, or per chain of jobs, representing the task at hand. This AM gets inserted into containers on nodes, when a job is running.
- Finally there is a global RM (ResourceManager), which is the ultimate authority on how to distribute loads and arbitrate resources. The RM consists of two entities, the Scheduler and the ApplicationsManager.

- The ApplicationsManager is responsible for accepting job submissions (at this point represented as an ApplicationMaster), i.e by doing code verification on submitted jobs, changing their status from "submitted" to "accepted". Once a job is accepted, the ApplicationsManager sends the ApplicationMaster to the scheduler to negotiate and supply containers for the job on the nodes in the cluster. The ApplicationsManager also has services to restart failed ApplicationMasters in containers, and retry entire jobs.
- The Scheduler is a pure scheduler in the sense that it only cares about delivering tasks to idle slots, based on free resources on the node. It calculates distributions across multiple nodes / containers, and can prioritize - i.e compute smaller jobs in front of large ones to more effectively complete job queue, even though the large job was submitted first. The Scheduler does not care for monitoring task completions or failures, simply distributing loads on the cluster.

The ResourceManager and the NodeManager form the data-computation framework. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.[4]

MapReduce

MapReduce is the heart of a traditional Hadoop cluster, for which every other component is built around, to maximize its efficiency. It is a model for distributed processing of big data, to process and consolidate. It is defined by two stages - the mapping phase, and the reduce phase. Both of these phases require resources from the node it is run on, defined by a set amount of map slots and reduce slots. The amount of slots on a node for each task is parameterized and can be set by an administrator / or typically algorithmically. The map phase consists of parallel map tasks, that map a key to a value. All of these map tasks then consolidate into the reduce phase, where they are combined so that one key exists for all accumulated values. So for instance if we're counting cards in several decks across several nodes, they would each map something like "(Queen of Hearts, 1)". In the end the reduce task consolidates all these single queens, so it ends up looking like "(Queen of Hearts", 27)". This is a far more effective approach than for instance looping through all the decks and incrementing a key by one for each matching key we find. Especially when the data that typically gets handled by a Hadoop cluster is diverse and hard to predict.

5.1.2 Zookeeper

Zookeeper is not a part of the Hadoop core utilities, but is is often to be found in Hadoop clusters. It is used as a distributed storage platform for configuration information, synchronization and naming. While YARN handles the distribution of tasks between the nodes in the cluster, Zookeper takes care of failover, race conditions and sequential consistency. This tool looks more like orchestration frameworks such as Puppet or Chef, in that it organizes the amount of YARN nodes and distributes configurations, availability and atomicity.

Zookeeper Quorum

Even though Zookeeper is made as a distributed configuration management and failover safeguard, it can be installed on a single node. Running Zookeeper in this way is called a Standalone Operation, and give a user a interface to remotely connect to, to get some data on running services etc. However, we need Zookeper distributed across all our nodes. This is called a Quorum. In a Quorum, every node will check the health of the others. If there is a consensus among nodes that one is down, Zookeeper is able to communicate this to YARN, letting it distribute loads across the remaining healthy nodes.

5.1.3 Apache Drill

Apache drill is the newest technology to be introduced in this chapter. All the previously mentioned frameworks are tried and tested - proven over time. Apache drill rests on top of all these technologies, and provides a way to query almost any non-relational database. This means that we can set up a cluster on a data lake with a very diverse data type, and still perform standard ANSI SQL queries on it. Even in formats like JSON, a simple SQL query can provide all the insights one might need.
Apache Drill is inspired by Google Dremel, which again is the driving force behind their advanced Big Data Analytics tool - Google BigQuery.

NEED MORE HERE. SHOULD BE BIGGEST ONE?

5.1.4 Drill ODBC Driver, and unixODBC

To be able to programatically perform queries to the Drill cluster we need an interface to connect to. This is done via a Open DataBase Connectivity (ODBC), that is installed on the NameNode of the Hadoop cluster. unixODBC is the self-proclaimed definitive standard for ODBC on non MS Windows platforms[6]. Once unixODBC was set up, the Drill ODBC Driver was installed on a arbitrary drillbit, although we chose the Hadoop NameNode for this as well. This then allows for setting up a Datasource referencing the Zookeeper Quorum which can then be contacted programmatically, interfacing Drill as if it were a standard SQL server / database.

5.1.5 pyodbc

There are plenty of ways to interface with an ODBC. We chose python as a preferred language because of familiarity. Therefore we chose to use pyodbc[7] to communicate with the ODBC, allowing a fully functional interface against our Drill configuration. pyodbc is a well established open source module, implementing the DB API 2.0 specification which is an API defined to encourage similarity between the Python modules that are used to access databases. Defining the Drill ODBC Driver as a datasource, which again points to the Zookeeper Quorum, proved effortless and fast.

5.2 Development

5.2.1 Infrastructure

To emulate a enterprise environment we set up a cluster consisting of four nodes, each with the same spec - 4 VCPUs and 8GB RAM. These were all VMs in OpenStack, each running Ubuntu 17.10. The gateway has a outward-facing network interface for SSH access, and from there each of the other nodes can be managed. They all had Apache Drill installed, and were set up as a Hadoop cluster running HDFS between them. The infrastructure stack consists of the

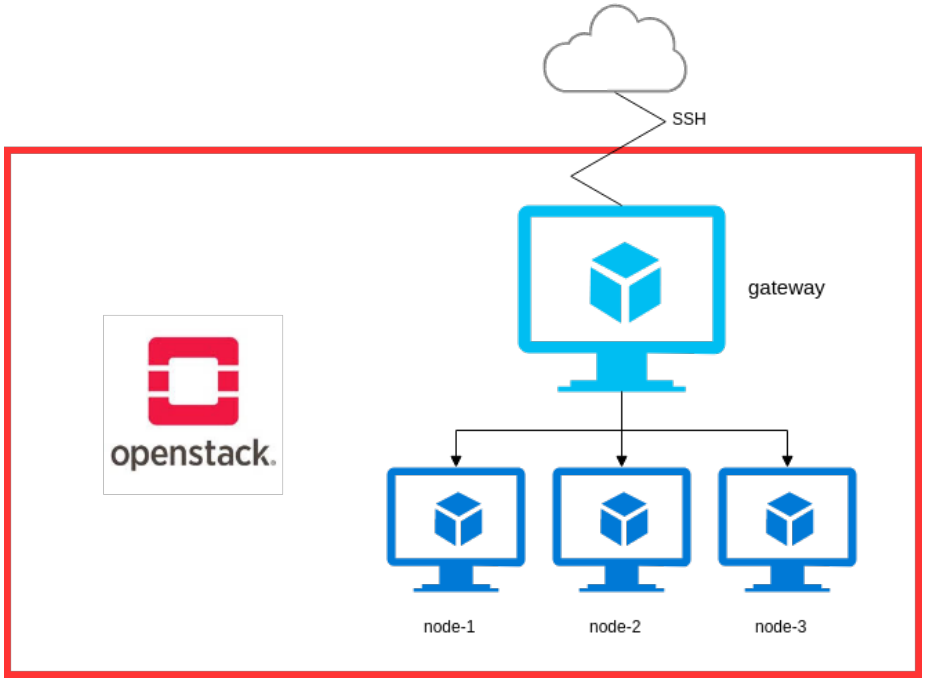

Figure 5.1: Cluster architecture

technologies listed in the technology section. All technologies are installed on all machines, except for the ODBC drivers, that are exclusively on the gateway. During a drill exection, any node can act as the orchestrator, named the *foreman*.

5.3 Testing

5.3.1 Load profiles

It should be infeasible for one configuration to solve all problems. After all, the concept of optimization takes great consideration to its specific context. Expecting to solve all aspects of a complex problem with one remedy leads to a candidate being a jack of all trades but master of none. Hence the solution should be tailored to a specific context, by letting the algorithm aggressively tune the system to work in its current state. To make sure this algorithm

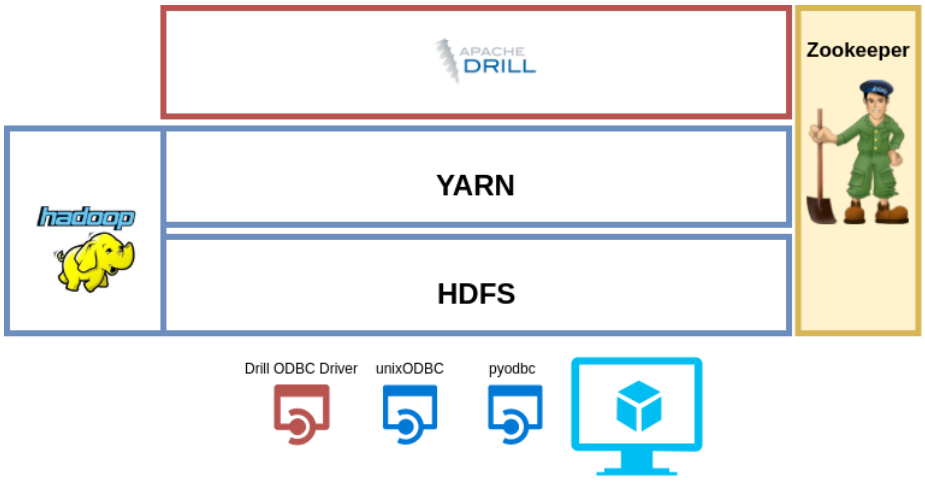

Figure 5.2: Cluster architecture

performs this way, testing was done on different kinds of data sets, with different kinds of queries.

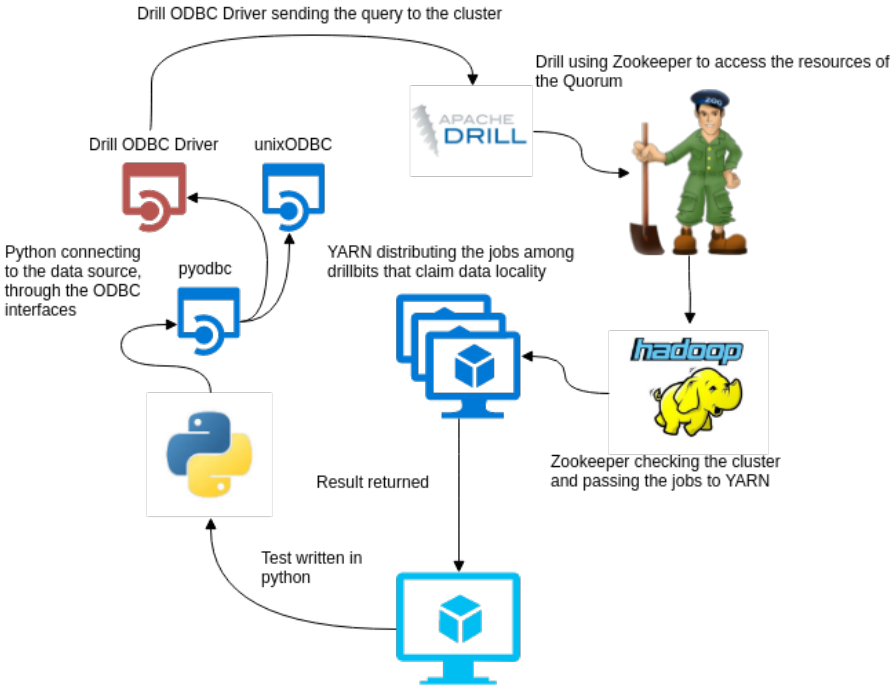

Figure 5.3: How tests are conducted

Chapter 6

Results

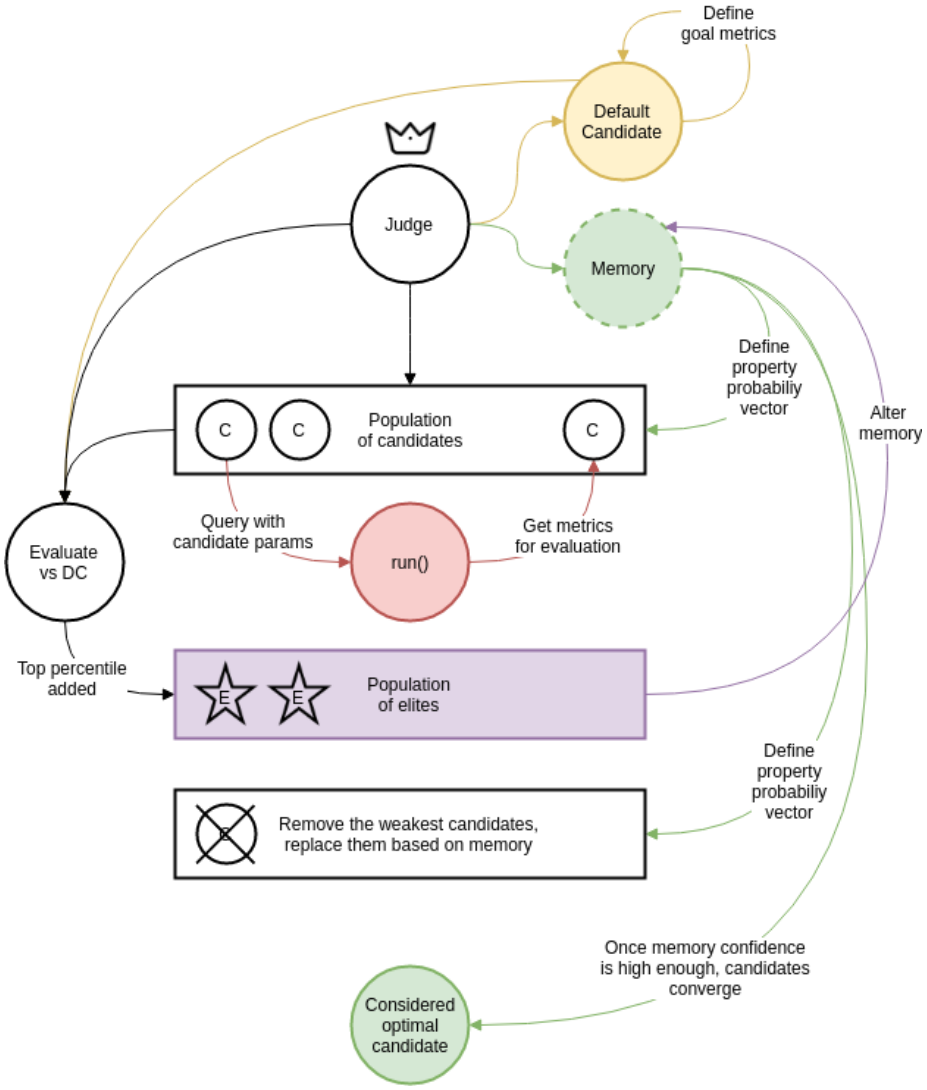

Figure 6.1: System overview

Chapter 7

Discussion

Chapter 8

Conclusion

Bibliography

[1] Li, M., Zeng, L., Meng, S., Tan, J., Zhang, L., Butt, A. R., & Fuller, N. (2014, June). *Mronline: Mapreduce online performance tuning. In Proceedings of the 23rd international symposium on High-performance parallel and distributed computing (pp. 165-176).* ACM.

[2] Liao, G., Datta, K., & Willke, T. L. (2013, August). *Gunther: Search-based auto-tuning of mapreduce.* In European Conference on Parallel Processing (pp. 406-419). Springer Berlin Heidelberg.

[3] BMC Software, Inc (2018, January) *http://www.bmc.com/guides/hadoop-examples.html*

[4] Apache Software Foundation (January 2018) *https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html*

[5] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., & Babu, S. (2011, January). *Starfish: A Self-tuning System for Big Data Analytics.* In Cidr (Vol. 11, No. 2011, pp. 261-272).

[6] Nick Gorham (2018, February) *http://www.unixodbc.org/*

[7] Michael Kleehammer (2018, February) *https://github.com/mkleehammer/pyodbc*

[8] White, T. (2012). *Hadoop: The definitive guide.* " O'Reilly Media, Inc.".

[9] Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). *Dremel: interactive analysis of web-scale datasets.* Proceedings of the VLDB Endowment, 3(1-2), 330-339.

[10] Ghemawat, S., Gobioff, H., & Leung, S. T. (2003). *The Google file system* (Vol. 37, No. 5, pp. 29-43). ACM.

[11] Dean, J., & Ghemawat, S. (2008). *MapReduce: simplified data processing on large clusters.* Communications of the ACM, 51(1), 107-113.

[12] Google LLC (2018, February) *https://cloud.google.com/customers/*

[13] Modis (2018, February) *http://www.modis.com/it-insights/infographics/top-it-jobs-of-2018/*

[14] Johannessen, R., Yazidi, A., & Feng, B. (2017, April). *Hadoop MapReduce scheduling paradigms. In Cloud Computing and Big Data Analysis* (ICCCBDA), 2017 IEEE 2nd International Conference on (pp. 175-179). IEEE.

[15] Hornby, G., Globus, A., Linden, D., & Lohn, J. (2006). *Automated antenna design with evolutionary algorithms.* In Space 2006 (p. 7242).

[16] Apache Software Foundation (2018, January) *https://drill.apache.org/docs/analyzing-the-yelp-academic-dataset/*

[17] Apache Software Foundation (2018, January) *https://drill.apache.org/docs/join-planning-guidelines/*

[18] Apache Software Foundation (2018, February) *https://drill.apache.org/docs/release-notes/*

[19] Ireland, C., Bowers, D., Newton, M., & Waugh, K. (2009, March). *A classification of object-relational impedance mismatch.* In Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA'09. First International Conference on (pp. 36-43). IEEE.

[20] 2018 BetterCloud Monitor (2018, February) *https://www.bettercloud.com/monitor/real-time-enterprise-messaging-comparison-data/*

[21] Stack Exchange Inc, posted by user: Thoth (2018, February) *https://stackoverflow.com/questions/42403229/mysql-database-with-thousands-of-tables*

[22] Gantz, J., & Reinsel, D. (2012). *The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east.* IDC iView: IDC Analyze the future, 2007, 1-16

[23] Porter, M. E., & Heppelmann, J. E. (2015). *How smart, connected products are transforming companies.* Harvard Business Review, 93(10), 96-114.

[24] Bendre, M. R., & Thool, V. R. (2016). *Analytics, challenges and applications in big data environment: a survey.* Journal of Management Analytics, 3(3), 206-239.

[25] Huang, G., He, J., Chi, C. H., Zhou, W., & Zhang, Y. (2015, June). *A Data as a Product Model for Future Consumption of Big Stream Data in Clouds.* In 2015 IEEE International Conference on Services Computing (SCC), (pp. 256-263). IEEE.

[26] Statista, Inc (2018, February) *https://www.statista.com/statistics/ + [266206/googles-annual-global-revenue/, 277229/facebooks-annual-revenue-and-net-income/]*

[27] Github user Edlich, curated list (2018, February) *http://nosql-database.org/*

[28] Baluja, S. (1994). *Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning* (No. CMU-CS-94-163). Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science.

[29] Bittorf, M. K. A. B. V., Bobrovytsky, T., Erickson, C. C. A. C. J., Hecht, M. G. D., Kuff, M. J. I. J. L., Leblang, D. K. A., ... & Yoder, M. M. (2015). *Impala: A modern, open-source SQL engine for Hadoop.* In Proceedings of the 7th Biennial Conference on Innovative Data Systems Research.

[30] Apache Software Foundation (2018, March) *https://impala.apache.org/overview.html*

[31] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Xin, D. (2016). *Mllib: Machine learning in apache spark.* The Journal of Machine Learning Research, 17(1), 1235-1241.