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

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.

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?

0.2 Foreword

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

Contents

Chapter 1

Introduction

1.1 Target group

This thesis covers the deep technical aspects of big data analysis and genetic algorithms. All techniques used will be explained in detail, but it is advised to have a certain degree of technical insight before reading.

1.2 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.

1.3 Personal motivation

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.

1.4 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. Metrics will NEED MORE RESEARCH TO DECLARE

1.5 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.

1.6 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.

Other types of performance tuning: Hadoop, Web, MySQL etc. Load profiling - type of queries with load profiling.

1.7 Why is this worthwhile

During my previous research the motivation for all the papers surveyed was mostly the same. **First and foremost, it was well acknowledged that Hadoop clusters show very suboptimal performance with out-of-the-box settings. Secondly, it was shown that finding optimal parameters for a Hadoop cluster is very time consuming and hard. As organizations often run with a lack of resources, time and domain-specific engineers, this is a field ripe for improvement.**

1.8 How far will this advance the field?

Hopefully this will provide a fully functional, open source light weight framework allowing companies to easily deploy Hadoop Clusters without worrying about tailoring the solution or suboptimal performance. If the task proves to be too big, this will lay the foundation for further work to make a de facto solution. Or something.

1.9 Structure of the report

Add comments about every chapter here.

Chapter 2

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]

2.1 History

2.1.1 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.

2.1.2 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.

2.2 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.

2.2.1 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]

2.2.2 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.

2.2.3 Tweak this part, or delete it

The properties are not truly random, as it would be inefficient to have too much variation. For instance, if a parameter has a rule of thumb value of 50, it would be expected that the optimal solution is within a range of this value, and not in the millions, or negative millions. Thus setting 50 as a central value for this property, and generating this property for the candidates as a normally distributed variation (bell curve) with 50 at its' peak, will avoid spending hundreds of generations tweaking this value.

Chapter 3

Related literature and theoretical focus

3.1 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.

3.2 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.

3.3 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]

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]

4.1 Why Drill?

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]. In addition to this DATA STORAGE INCREASE HERE

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.

Chapter 6

Results

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 (January 2018) *https://drill.apache.org/docs/analyzing-the-yelp-academic-dataset/*