Final Year Project Report

Automated Software to Understand Functional Relationship Between Dynamic Energy and Performance Events

Sukrat Kashyap

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Ravindranath Reddy Manumachu

UCD School of Computer Science

University College Dublin

April 6, 2018

Project Specification

General Information:

A energy model representing a relationship between dynamic energy consumption and performance events (PMCs) is constructed experimentally and the experimental dataset has the following format typically (k events, n records):

$E_1,\ x_{11},\ x_{12},\ x_{13} \ldots x_{1k}$
$E_1,\ x_{21},\ x_{22},\ x_{23} \ldots x_{2k}$
$\ldots$
$E_n,\ x_{n1},\ x_{n2},\ x_{n3} \ldots x_{nk}$

where E_i is the experimentally obtained dynamic energy consumption of i-th record and xij are the experimentally obtained performance events (PMCs).

Given such an experimental dataset as an input, the goal is to determine/understand the functional relationship between the dynamic energy consumption and performance events (PMCs).

Two real-life datasets will be provided to the student.

Core:

The goal is to write automated software that will detect the following:

1. Existence of records where the dynamic energy consumption is the same (within an input tolerance) but all PMCs (with the exception of one) have same values. Then the relationship between energy and the one PMC is visualized to see the nature of the functional relationship.
2. Having accomplished step (1), understand the monotonicity of the relationship between dynamic energy consumption and performance events (PMCs).
3. Existence of records where the dynamic energy consumptions are different (within an input tolerance) but all PMCs have same values (within an input tolerance) suggesting the non-existence of a functional relationship.

The software must be written using any one mainstream language but preferably one of the following: C, C++, Python

The software must be well documented and tested.

Advanced:

Given an experimental energy model dataset as an input, the goal is to write software that performs intelligent but computationally feasible simulations where combinations of inputs are generated to study the existence/non-existence of a functional relationship between dynamic energy consumption and PMCs.

The software must be written using any one mainstream language but preferably one of the following: C, C++, Python.

The software must be well documented and tested.

Abstract

With the advent of technology, the demand of energy has increased dramatically. Increasing number of electric driven equipments such as personal devices, hybrid vehicles and embedded systems have made the management of energy crucial. This projects focus is on understanding the energy consumption by High Performance Computers. It has been known that CPU power consumption correlates to processor performance. And a number of models have been proposed to use performance events e.g. cache miss, DMA access, I/O interrupt, for the estimation of power consumption by the systems. This project takes a novel approach.Instead of estimating the energy consumption by the systems using performance events, a software is developed in order to better understand the functional relation between the performance events and the energy consumption. A number of linear models were proposed due to the trickle-down effect of the performance events on power consumption [6]. The software also tries to analyze the linear dependence between the performance events and the energy consumption. This analysis helps validate the linear models that have already been proposed.

Acknowledgments

I would like to thank my supervisor Ravindranath Reddy Manumachu and mentor Alexey Lastovetsky for the assistance and guidance of this project. I would also like to thank my family and friends for supporting me in my work.

Table of Contents

Chapter 1: **Introduction**

1.1 Motivation

Modern day technology has developed under incredible speed in recent decade and the computing power growth rate is truly phenomenal and lasting impact can be felt and benefit us in many ways. It is important to realise the worldwide effect on environment by the increase in consumption of power by these technology advancements.

According to [9], the power consumption growth rates of PCs are about 7.5% per year. Data Centres and network play much larger role as they both have power consumption rate of 12% each. This considerable growth is due to increasing data to be accessed, stored and processed. The constant expansion of energy consumption leads to increase in carbon emissions. CO_2 emissions from ICT (Information and communications technology) are increasing at a rate of 6% per year, at such rate by 2020 it will account to 12% of worldwide emissions [10].

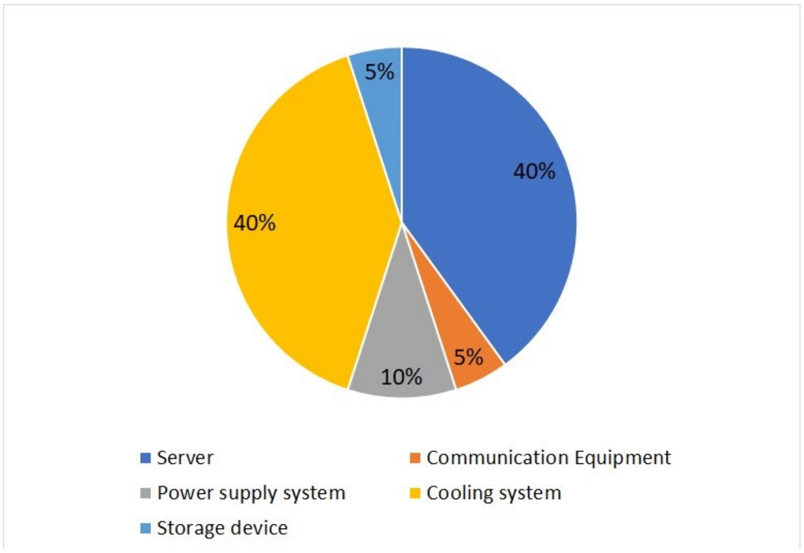


Figure 1.1: Energy consumption distribution of data centers.

The major roles of power consumption by a data ware house is played by the servers and the cooling system which is used to cool down the physical parts of the server. The figure 1.1, shows that both the servers and the cooling systems account to 80% of the total consumption with both accounting for 40% each [10]. Making power and energy one of the key challenges in system and software designs. In order to improve processor's performance while limiting power consumption, designers have increasingly utilized dynamic hardware adaptions. But to guide to these hardware adaptions it is necessary to measure the power of systems accurately. Such adaptations can reduce power consumption and chip temperature by reducing the amount of available performance. Temperature sensors are slow in reponse due to the thermal inertial of the micro processor. Relying on them would give slow response and detection of temperature changes. Many demonstrations have been done to show that performance counter / performance events are effective proxies of power measurement [2].

The study of the relations between different performance counters and energy consumption has become crucial. The existence and analysis of the functinal relation between them will enable minimization of power consumption leading to less heat generation from the CPU. This decrease in temperature will further reduce the usage of cooling system. Which all in turn leads to cost minimization and a step towards eco-friendly environment.

Thus, discovery of the functional relation between the power consumption and the performance events (PMCs) of system, will enable the ability to predict the energy consumption for certain computations.

1.2 Aims

The project aims to create a piece of software that will help understanding and finding relations between energy consumption and performance counters. The software must aim to be extensible, easy to use and performant. It should be able to perform analysis like regression and clustering on the data. These actions will help in showcasing the relationship of various parameters and the output. The parameters are the performance events and output is the energy consumption by the systems. Following are the objectives that are attempted to be accomplished.

1.2.1 Existence of functional relation

Datasets are mostly pair of multiple inputs and a single output. And the possibility of the existence of a relation is what have to be found. Is it possible to define the data in a form of function? Question like this is one of the aims. Defining the data in a form of function can be thought of as explaining the dataset in a form of a formula which can combine various parts of the input parameters and match the output. It is quite impossible to tell that whether a function definetly exists. This is because the data is always a subset of the population and there are number of records which are either not in the dataset or it is not know. But, regardless it can definitely explain the non-existence of a functional relation by finding a pair of input parameters and output that violate the definition of a function.

The pair which violates the definition of a function must be looked at as there is high probability of that data set record being corrupt or if not it can actually act as a contradiction and explain the non existence. If this outlier is in fact corrupt, this process can be used for data cleanups as well.

1.2.2 Analysis of functional relation

Once, it is not possible to prove functional non existence in the dataset anymore. The dataset must be analysed in order to better understand the the functional relation of the parameters and the output. The analysis of the relation explains the contribution of a parameter to the result. It gives better understanding about the correlation between two quantities. Does high page faults correlates to high energy consumption? Questions like this is what the analysis tries to answer. This can also be thought of an explaination of the resultant parameter.

In other words, understanding the correlation is the other aim. Once it is know that there is high correlation. Further investigation can be done in order to find the form of the relation.

1.3 Approach

For each aim, two approaches have been employed. Both the approaches tries to achieve the same conclusions but with different methodology.

The experimental data sets provided has the following format:

$E_1,\ x_{11},\ x_{12},\ x_{13} \ldots x_{1k}$
$E_1,\ x_{21},\ x_{22},\ x_{23} \ldots x_{2k}$
$\ldots$
$E_n,\ x_{n1},\ x_{n2},\ x_{n3} \ldots x_{nk}$

where k is the number of parameters (performance events), n is the number of records in the dataset. E_i is the experimentally obtained dynamic energy consumption and x_{ij} are the experimentally obtained performance events (PMCs).

1.3.1 Existence of functional relation

The main objective here is to find the nonexistence of functional relationship. In other words, it means proving that the dataset cannot be explained in terms of a function/formula.

First approach here is to find two performance events tuples $(x_{i1},\ x_{i2},\ x_{i3} \ldots x_{ik})$ and $(x_{j1},\ x_{j2},\ x_{j3} \ldots x_{jk})$ that are equal within some tolerance, but their corresponding E_i and E_j dynamic energy consumption are different. Existence of such a tuple in database will lead us to prove the non-existence of a functional relationship.

It is infact easier to find two equal records by ordering the dataset by their parameters and going through the entire dataset once to find equal records and comparing their dynamic energy consumption. The order of complexity of such is $O(N \log(N))$ which is the complexity of sorting as that is the only heavy duty task involved. But it gets complicated when tolerance comes into play. Likewise, task at hand is to find similar records rather than equal records. The project employs 2 methods to measure the similarity between the two data records.

First approach:

The data records are imagined as data points in $k - space$ as we have k number of parameters. Then the whole space is divided into small $k - dimension$ cubes with dimensions $(t_1, t_2, \ldots t_k)$ where t_i is the tolerance of each parameter. The data points are then put in their respective cubes. When two points lie in the same cube that means there paramters are similar and so must be there output. If they donot have similar output within some tolerance then they violate the functional relation.

Second approach:

The data points in $k - space$ are said to be similar if the euclidean distance between them is less than the t provided, where t is the total/max tolerance. In this method, clustering is done using the distance between the parameter vector of the data points.

1.3.2 Analysis of functional relation

Many different type of relations can exists between two variables. But, since the correlation between the variables are know to be linear. Linear regression is done between the dynamic energy consumption with the performance events. Many researches have have been successful in estimating using linear models to estimate energy consumption using performance events. [8] The preference given to linear models is due to the trickle down effect of the performance events. [3] Hence, Linear regression is used to analyse the relationship.

Linear regression is done with 2 variables but there are k variables relation to energy consumption making it hard to understand the nature. Multiple linear regression tries to find the best fit to the data. But best fit to the data hinders the tolerance factor in the dataset and also undermines the actual relation. As multiple regression is also interested in predicting the output variable. Hence to analyse one of the variable, the data point must be grouped using their other $k-1$ variables. Each grouped cluster is then visited and linear regression is performed. Two approaches are employed to cluster the dataset. These are similar to clustering techniques used in finding the existence of the relation.

First approach:

Data points are clustered by putting the data points in the respective $k-1$ dimension cube. Forming a cluster with almost similar $k-1$ variables. Each data point is put in one of the cube. These data points have similar $k-1$ performance event vector but varying one of the parameters and output variable. Regression line calculated corresponds to the effect of the parameter variable to the output variable in a particular situation. The situation is the similar $k-1$ parameter values.

Second approach:

Euclidean distance is used to isolate similar $k-1$ variables. A $k-1$ dimension sphere can be imagined for simplicity with its radius being the total/max tolerance. Here, each data point has its own small sphere. The regression line calculated for each data point act as the tangent to the data point in the direction of its neighbours.

1.4 Structure of report

In this report, the motivation behind the project and brief description of the approaches that will be undertaken to reach the objective are seen already.

This Introductory chapter is followed by Background research, design aspects of the software, implementation of the software, testing and evaluation of the tool followed by the conclusion and the future works.

Background research explains about performance events and how do they influence energy consumption by the system. It also explains the approaches in detail mentioned above and analyses their complexity and how they can be optimised.

Following Background research, design and implementation of the software is deep dig into as we want an extensible, performant tool. Testing and evaluation of the software is explained. How the software is tested and its evaluation on its performance. The report ends with conclusions and future works that shows how the project can be extended and applied to various other domains.

Chapter 2: **Background Research**

Many designers are increasingly utilizing dynamic hardware adaptations to improve performance while limiting the power consumption. Whereas Some are using software to decrease power usage for e.g. putting the system in sleep mode when it's in idle state. The main goal remains the same, which is to extract maximum performance while minimizing the temperature and power. The study and examination of the events affecting the consumption will help us to predict and minimize the consumption of energy with very high accuracy.

2.1 Energy consumption and Performance Events

First let's look at energy consumption. Energy consumption is the power (usually in watts) consumed by a system. This system could be the processor/CPU, memory, disk, I/O (Input/Output) system, chipset or the whole computer system itself. It has been known that power consumption of any system has a high correlation to its usage. Since the correlation is high for its usage. It means usage is a is a good way to understand the power consumption. Performance events are one of the better ways to measure the usage of a system as they are known much more faster than the temperature sensors. As temperature sensors are slow in response due to the thermal inertial of the micro processor.

Now let's look at what are performance events, performance events are any events that affect the consumption of energy in some way. Selection of performance events is quite challenging. A simple example would be the effect of cache misses in the processor. For a typical processor, the highest level of cache would be L3 or L2 depending on the type of processor. Now for some transaction which could not be found in the highest level of cache (cache miss) would cause a cache block size access to the main memory. Thus, number of main memory access would be directly proportional to the cache misses. Since these memory access is off-chip, power is consumed in the memory controller and DRAM. Even though, the relation is not simple as it seems but a strong casual linear relationship between the cache miss and the main memory power consumption [3].

Another performance event can be instructions executed. The more instruction being executed, will turn on and use more units of the system. Hence, power is consumed as opposed to when the processor is in its idle state [6].

Cache miss, TLB misses are also a good performance events as they seem to have a strong relationship between the power consumption as processor needs to handle memory page walks. Same can be said for Page faults where a program is not able to find mapped address in physical memory as it has not been loaded yet. This causes a trap which can result into number of situations, one of them which is to get the data from disk. In simple terms it is longer walk from cache miss. This walk to the disk and raising of exceptions would consume more energy by the disk as well as the CPU. Another thing to note here most of the relations that we saw above are directly proportional to each other. Increase in the variable like cache miss, number of cycles in CPU etc gives rise to energy consumption by the system. This makes linear model a good point to start when analysing the relationship between the performance events variables and energy consumption.

2.2 Related Work

In this section, some of the prior researches are discussed. Hardware performance counter's links to processor power consumption was first demonstrated by Bellosa. In [2], Bellosa demonstrated the high correlation between performance counters such as memory references, L2 cache, floating point operations to processor power consumption.

Gilbert in [6], predicted the power consumption for Intel XScale processors using performance monitoring unit events. Since power consumption is greatly dependent on executing workload, power estimation was done using HPCs (Hardware performance counters) such as Instruction cache miss, TLB misses etc. Linear parmaterisation of power consumption based on performance events was done based on performance events.

In [12], proposed a full system power model for CPU-intensive and memory intensive applications with active cycles, instruction retired and LLC missed as performance events. A full power model for I/O intensive applications was also proposed considering the system level utilization as a complement for performance events. Many machine learning based algorithms like logistic regression, elastic net and k-nearest neighbours were applied to real world application.

Bircher approached in a distinct way by using events local to the processor and eliminating the need for sensors spread across various parts of the systems [4]. Linear regression modeling was done in order to predict the power consumption at runtime. Multiple linear and polynomial regression was done only when accuracy was not obtained.

High correlation between performance events and power consumption was demonstrated by all of them. But all of them tried to predict the power consumption using sytem events. They use various methodolgies in predicting so. In [12], a number of machine learning algorithms were used. But in this project, an attempt is made to understand the monotonic relation between the events and power. The predictive models would work on a particular specification of system. But understanding the relation will help define a model on the architecture level. This will also help in verifying the models that already exist.

2.3 Existence of functional relation

Definition: *Given a dataset of pairs (x_i, y_i) where $i \in [1, n]$ of two variables x and y, and the range X of x, y is a function of x iff for each $x_0 \in X$, there is exactly one value of y, say y_0, such that (x_0, y_0) is in dataset.* [13]

Above is the definition of functional relation. In our case the $x_i = (p_1, p_2, \ldots p_k)$ and $y_i = E_i$ where p are the performance events and k is the number of events and E is the energy consumption.

In other words, functional relationship is an one to one mapping between our input variables $p_1, p_2, \ldots p_k$ and output E_i. This reasoning can be explained quite intuitively as assuming functional relation we can formulate a $f(p_1, p_2, \ldots p_k) = E_i$ where f is function. Now if one $(p_1, p_2, \ldots p_k)$ can give more than one output E_i then that f function is either not correct or f does not exist. A question that immediately arises is what if two different $(p_1, p_2, \ldots p_k)$ gives same output E_i. The answer is it is possible and it doesnot violate the functional relation definition as there is still 1 to 1 mapping from input to output. The only difference is the functional relation is not surjective any more which means that one cannot figure out the input values from output values (i.e. the other way around). But we are only interested in predicting the output rather than inputs from the output variables.

The Proof of the existence of functional relation is given below:

But first let's look at our dataset that will be provided. We know that the data will be in the following format:

Let k be the number of parameters for the energy and n be the number of records in the dataset

$E_1,\ x_{11},\ x_{12},\ \ldots x_{1k}$
$E_2,\ x_{21},\ x_{22},\ \ldots x_{2k}$
$\ldots$
$E_n,\ x_{n1},\ x_{n2},\ \ldots x_{nk}$
where E_n is the dynamic energy for the nth tuple and x_{nk} corresponds to the kth performance event for nth record.

We will use mathematical definition of functional relationship given above to prove the approaches:

To Prove: We need to prove that finding atleast 2 equal performance events with different dynamic energies ensures that there exists no functional relationship in the dataset.

Proof. Let us assume that there exists a functional relation such that:
$f(x_{n1},\ x_{n2},\ \ldots x_{nk}) = E_n$
where f is the functional relation for the dataset.

If we have $f(x_{i1},\ x_{i2},\ \ldots x_{ik}) = E_i$ and $f(x_{j1},\ x_{j2},\ \ldots x_{jk}) = E_j$ where $E_i \neq E_j$ and $(x_{i1},\ x_{i2},\ \ldots x_{ik}) = (x_{j1},\ x_{j2},\ \ldots x_{jk})$

If such i and j exists. Then, we can conclude that the f is not a function by using the definition of a function as this assumed function has two images.

Which contradicts from the hypothesis stated above. Hence by proof of contradiction we could say that f is not a function on the dataset. $\square$

Now the task is to find similar $x_{11},\ x_{12},\ \ldots x_{1k}$ input variables. If it was equality, it could be performed trivially by sorting the data records on the basis of $x_{11},\ x_{12},\ \ldots x_{1k}$. Followed by going throught the data records in that order to find equal records. As equal records will be next to each other when sorted.

But the dataset accumulated is collected from experimental setup. Experimental setups data have some error/tolerance associated to it. As the equipment or software cannot accurately measure and have some error associated to it. e.g. Energy consumption is measured by a power meter which does have a confidence interval. The tolerance associated with the dataset no longer allows to perform equality on input variables. Hence, a method is needed to measure the equality with the tolerance.

Now after knowing the tolerances the data records in the dataset now looks more of the form:

$E_n \pm e_E,\ x_{n1} \pm e_1,\ x_{n2} \pm e_2,\ \ldots x_{nk} \pm e_n$
where e is now the error associated with the variable.

Hence dataset must be clustered with the tolerances associated to them. Data points having similar k parameters will fall in the same cluster. The output variable $E_n \pm e_E$ must be close to each other within some tolerance for all the points in the cluster. If they are not, it violates the functional relation and the data points are picked and must be given to the user to either discard if corrupt. This removal of corrupt data records is known as Data cleanups. If they are not corrupt then it proves the non-functional relation in the dataset. Analysis of functional relation can be done when no records are found that contradicts the functional relation definition.

2.4 Analysing functional relation

Now this section corresponds to analysing the functional relation. A functional relation can of many forms, there can be logarithmic functional relation, exponential, linear, polynomial etc. According to [3], shows that functional relation between performance event and dynamic energy is of the linear form. This is due to the trickle down effect of the performance events. Hence, Linear model is used to understand the relationship between the events and the consumption.

This project is more interested in finding whether a strong relation of monotonicity exists in the dataset or not. And linear regression are best suited for this kind of task. Linear regression is an approach determine how strongly two variables correlate with each other. Correlation not necessarily means causation. Looking at the data points and linear regression values such as Pearson coefficient, R^2 can determine how strong the correlation is between 2 values but it cannot explain the cause of it.

Dataset contains k parameter variables, to analyse relation between the parameter variables and the output variable. One parameter is chosen at a time. To see the change in the parameter variable chosen and output variable, isolation of the other variables is required. Isolation is done by grouping the data by $k-1$ parameter variable forming a number of clusters with similar $k-1$ paramters. These clusters are then visited and linear regression is performed on the k^{th} variable and the output variable.

From the steps above, it can be observed the parameter to be analysed is isolated by grouping the dataset with the variables that are not being analysed and are equivalent. Since, clustering of dataset is performed again but with different vector dimension of $k-1$, the clustering algorithms are needed by both the objectives.

2.5 Clustering Methodologies

This section introduces two clustering algorithm that was used to cluster data points which are similar or close to each other needed by both of the above objectives. There are many clustering methodologies out there [11]. No clustering algorithm can be universally solve all the problems. Algorithm which favor certain observations, assumptions and favor some type of biases are designed and used.

Clustering involves grouping similar data by their attributes. There were number of clustering algorithms like Heirarchial clustering, K means clustering, Graph based clustering e.g. Chameleon. But we chose, Grid based clustering and Distance based clustering.

Most of clustering algorithms like K means clustering requires the number of clusters to be formed. In the dataset, number of clusters need to be form is not known. In Heirarchial clustering, heirarchy of dataset is used but the relation is functional and not heirarchial. Graph based clustering usually requires edges, and edges could be formed but they would increase the space complexity exponential. As more storage will be required to store the edges of the dataset.

The two types of clustering algorithms chosen, uses some facts about the dataset. Datasets with tolerances for clustering would be best suited to use these algorithms.

2.5.1 Grid based clustering

Since tolerance must be used as a measure of clustering datapoints. The definition of equality for a variable changes from $x_{ia} = x_{ja}$ to $|x_{ia} - x_{ja}| \leq e_a$. Now, if simple sort and search for similar variable would be employed, it would not work correctly as equivalent records would not be next to each other. They might non-equivalent records in between.

Example data to illustrate the same:

E	x_1	x_2
3.5	4.5	6.5
4.1	4.6	10.6
0.2	4.7	6.5
1.6	4.7	7.6

The above three records are sorted by their parameters x_1, x_2. We can see that if the $e = 0.5$ is the absolute error. Then record number 1 and 3 are similar to each other but donot lie next to each other. This increases the complexity of finding similar records from $O(N \log(N))$ to $O(N^2)$ as we donot know where the similar records will lie, so N into N search must be performed which is quite inefficient.

To make it effecient, instead of finding similar records in the dataset. The whole $k-dimensional$ space is divided into small $k-cubes$ whose dimensions are $(e_1 * e_2 * \dots e_k)$. Each $k-cube$ has its own integer coordinates in space. The data records are grouped together with respect to their respective $k-cube$ coordinates. Since the coordinates are integers (equality can be performed). They can be grouped in $O(N \log(N))$ time. The calculation of the coordinates in which the data point belongs to can be calculated in $O(1)$ time by the following.

$$coordinate = (\left\lfloor \frac{x_{n1}}{e_1} \right\rfloor, \left\lfloor \frac{x_{n2}}{e_2} \right\rfloor, \dots \left\lfloor \frac{x_{nk}}{e_k} \right\rfloor)$$

Every data point will have a corresponding coordinate they belong to. Data points are then grouped together by their coordinates. Each and every grouped coordinate is a $k-dimensional$ cube.

Algorithm 1 Grid based clustering

```
1: procedure GRID($a, b$)
2:     Input $dataset$: a list of data points, $e$: errors for each variable
3:     Result clusters created with each cluster having its unique index
4:     $list \leftarrow List.empty$
5:     for each $point$ in $dataset$ do                          ▷ $N$ iterations
6:         $coordinate \leftarrow (\lfloor point.x_1/e_1 \rfloor, \dots \lfloor point.x_k/e_k \rfloor)$
7:         $list.add([coordinate, point])$
8:     sort $list$ by the $coordinate$ value                   ▷ $N \log(N)$ for sorting
9:     $result \leftarrow$ group $list$ by the $coordinate$ value          ▷ $N$ iterations
10:    return $result$
```

It can be seen, since the space is divided into cube of certain dimension. The cubes are disjoint and they donot overlap. It can be imagined as a building of boxes stack on top and side of each other. There will be points which lie on the edges of the $n-cube$. Since the cubes donot overlap. They will be close to some of points in the cubes next to them. But in the algorithm one data point is assigned to only one cluster. This problem is solved by the next approach which is the distance based clustering. This problem comes with performance advantage.

The algorithm complexity can be measured.

- Time complexity $O(Nlog(N))$: This is because of sorting which is the slowest operation in the algorithm, but neverthless optimizations can be done by using a Dictionary or Binary search tree to group data with equal coordinates. Using Dictionary will give complexity of $O(N)$.
- Space complexity $O(N)$: For each data point only the corresponding $coordinate$ is stored.

2.5.2 Distance based clustering

In this method, data points are thought of as vectors in euclidean space of $n-dimension$ where n is number of parameters considered when clustering. The equality of vectors according to which clustering should take place is done by using the euclidean distance between two vectors. Two vectors are said to be equivalent if the distance between the two vectors is less than the tolerance specified.

Let u and v be two vectors in space with dimension n,
then the euclidean distance is define as:

$$dist(u,v) = \sqrt{(u[1]-v[1])^2 + (u[2]-v[2])^2 + \dots (u[n]-v[n])^2}$$

Now, two vectors u and v are said to be equal or close enough when:

$$dist(u,v) \leq tol$$

where tol is the maximum distance between two points to call them neighbours.

This tolerance can be thought of as the total tolerance of the between two data points. Because, the difference between two points in all of their different dimensions are squared and "added" together and then square rooted. It signifies the total tolerance that is allowed between two datapoints.

This clustering is creating spheres of $k-dimension$ for each data point. Here, overlapping of spheres is allowed. Incuring huge performance cost.

Algorithm 2 Distance based clustering

```
1: procedure DISTANCE_BASED($a, b$)
2:     Input $dataset$: a list of data points, $tol$: tolerance, $oTol$: output tolerance
3:     Result subset of the data points that violate functional relation
4:     $cluster \leftarrow Dictionary.empty$
5:     for each $point$ in $dataset$ do                          ▷ $N$ iterations
6:         $cluster[point] \leftarrow List.empty$
7:         for each $neighbour$ in $dataset$ do                  ▷ $N$ iterations
8:             if $dist(point, neighbour) < tol$ then
9:                 $cluster[point].add(neighbour)$
10:    return $cluster$
```

The algorithm complexity can be measured.

- Time complexity $O(N^2)$): This can be seen as for every data point in the dataset, every datapoint is again visited to find its neighbour. Optimizations like using indexes and parallelisation are used to optimize the running time.
- Space complexity $O(N^2)$: For each data point all of its neighbour are stored. This overhead can be reduced by instead of storing the all the neighbours, whatever computation is needed must be done and stored and the cluster neighbours are thrown away. The only draw back is when doing a different computation, the neighbour will have to be found again.

2.6 Software

This section introduces the software and the framework which were chosen to implement the software. The software basic requirements were to be on of the mainstream programming language, open source, cross-platform and which has long term community support. So the top options from which we had to choose from was Java, C/C++ and Python.

In [7], vary nice comparison is done between Java, Python and C. When it comes to portability, Java programs can compile to portable executable bytecode which can be run on any computer as long as it supports Java virtual machine. ANSI C, which can achieve the same portability require re-compilation of the source files. This means Java programs can be distributed as binary files that can run on any platform where as C would require recompilation tools to recompile on the specific hardware. While Python code can enjoy this advantage and can run on any machine with a Python interpreter installed, the C-based Python extension modules, as well as the central Python interpreter itself, are only portable after (sometimes painful) recompilation of C source. With ease of portability and cross-platform we also know that Java program are robust and can never have segmentation fault, and most of the errors are catchable runtime exceptions. On the other hand in C uncatchable, destructive errors all too frequently at runtime [7].

Garbage collection is supported by both Python and Java with the exception of C. Making it easier to code and build software. But since, performance is another aspect, C and Java tops Python because of the interpretated nature of the Python [7]. C is more performant than Java because of AOT (ahead of time compilation) in C than JIT (Just in time compilation) of Java. But writing and reading multi-threaded code is much easier in Java because of their nice set of portable API's and well documentation. Python GIL (global interpreter lock) limits thread performance vastly [1].

Since, we have to search through a lot of records, a lot of operation like grouping, summing, counting were required to be done on the dataset. Storing the dataset on memory using array list, map any data structure would be hard as well as error prone with too much care needed for multi-threading. So, we decided to do the heavy dataset related operation from database. Which can later be seen to use indexes to fasten up the search. With a little performance penalty, we wouldnot have to worry about the memory limit anymore and disk space would be utilized by the database when not enough main memory is available.

A number of database models are out there namely relational model (e.g. MySQL), document model (e.g. MongoDB), graph model (e.g. Neo4j) and multi-model (e.g. ArangoDB). The dataset provided to the software can be with any number of columns to analyse. Dataset will be in the form of k columns separated by comma with each column containing “double” values where $k \in \mathbb{Z} \wedge k \geq 2$. So the storage of these datasets donot have pre-defined schema. Grouping operations is one of the most important operations required. Also, we want to be able to represent the operation with simple query language. ArangoDB stood out from all of the databases, relational databases has really readable query language but its strict schema negates are requirement. In document model MongoDB does support our dynamic model requirement but its query syntax is harder to read. Graph model is made for graph based analysis whereas our analysis donot requires graph. ArangoDB completes all of our requirement by supporting our dynamic model, having simple query language similar to MySQL named AQL and does support grouping operation providing graph operations as a bonus if ever required in the future.

With this, conclusion was made to use Java as the programming language and ArangoDB as the database for hard core data computing. Java has a large community support and ArangoDB fully supports Java client apps.

2.7 Applications

The reason for making a software like this verifies and gives the confidence about the dataset. If data do not fit any functional hypothesis in a space, much time could be saved by preventing the unneeded search. It also analysis the linear relation of the dataset. It assumes data is linear, does linear regression on various cluster and returns multitude of values describing the data (e.g. number of cluster formed, number of outliers, mean of Pearson's coefficient R etc.).

It can be used to confirm the linear models already presented by [3] [2] [8]. The tool can be used to find linear relationship in the dataset.

The software is not restricted to the use of only on dataset which consists of performance events and power consumption. It is a general-purpose software which will for work for any kind of dataset in which user wants to know the existence of functional relation. The refuting of the claim of functional relation on dataset is the objective of the software.

We also know that the dataset provided is usually experimentally measured values which are not accurate. Every measuring device has some margin of error. The software will be flexible in the sense that the equality comparison of values in the approaches will always be done keeping in the margin of error provided. It can also iterate certain of tolerance which will further be used to study the effect of tolerance on the dataset.

Chapter 3: **Design and Optimisations**

This section discusses the various details about software implemented in detail.

3.1 Design

The software is implemented in Java using ArangoDB as backend database. The application has 3 layer structure as per Figure 3.1. The database layer which is the ArangoDB database itself. The business layer, this layer contains all the logic for talking to the backend ArangoDB, the clustering algorithms and other functions. The application has 2 frontend: Web application which is made using the Spring framework in Java and a simple command line application. Both the front end uses the same business layer hence providing same functionality.

The way this project was strategised was to make command line first for easy testing of the core functionality like clustering, inserting the data, getting the data etc. Then the same core libraries are called via the web application. Web application was created for ease of use and multiple simultaneous analysis. Another reason for keeping the command line application was, some of the commands took too long to process. The connection between the server and the client used to timeout. Websocket's helped keeping the connection alive but once the connection is broken the results were lost in the process.

The whole project has 3 modules namely core, cmdapp, webapp. Gradle build tool was used for compiling, testing and running the application.

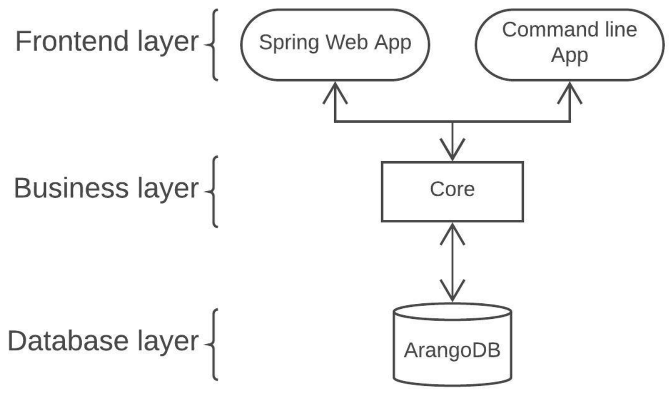


Figure 3.1: Structure of the app

3.1.1 ArangoDB

ArangoDB is a multi-model database. It supports three data models namely key/value, documents, graphs and a unified query language called AQL. AQL is very similar to SQL query language. AQL is used for CRUD operations. It provides scalabe and highly efficient queries. It uses JSON as a default storage schema.

The application uses this database for all the heavy duty data operations like grouping, summing, filtering neighbours etc. The reason a database was used instead of doing it directly in Java was for faster development, thread-safety and also the indexes provided by the database for faster queries.

3.1.2 Core

This business layer of the application contains all the core logic. It contains all the query, the driver to talk to ArangoDB, the logic for converting from and to csv etc. The core module uses spring-context which is an application IoC (Inversion of Control) container. This container can be consumed by any application by adding the library and providing the database connection configuration. This saves the other application from creating each and every object. The IoC container creates the object for the application automatically at runtime. The strategy was to create a library which contains all main logic and which can be consumed by any kind of application. This can be seen as we have implemented in two applications command line and web application.

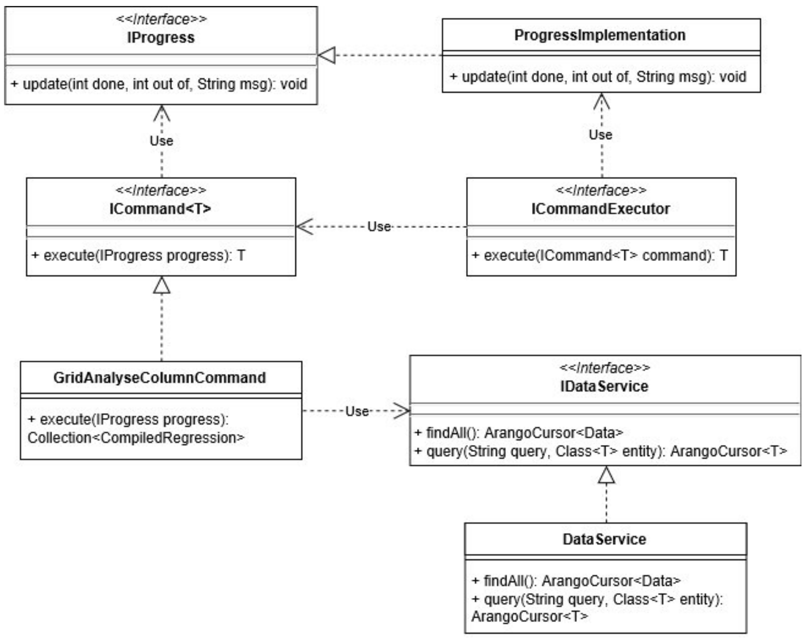
Apart from using IoC containers, the main design pattern used by this module of application is the Command Pattern [5]. In this pattern, an object is encapsulated with all the information needed to perform an action. The biggest feature of the command pattern is that it can be executed at any point in time. The command in our case is executed by a Command executor. This separation of the actual command and executor allows for greater changes in the future of the project. It provides a lot of flexibility and ease of adding and replacing commands in later stages of the development. One of the purposes of using command pattern was the certain routines in the application are quite slow e.g. distance based clustering. The command pattern creates a unified way for getting the updates/progress of a task being executed.

The drawbacks of this pattern is it makes the source code larger quickly because one file can only represent one command. The number of files in the project increases very quickly. Command names are usually big and so goes for the file names. The order of commands becomes important at later stages and can jeopardize the reliability of the system [5].

Figure3.2, shows the implementation of command pattern. Only one Concrete command (GridAnalyseColumnCommand) is been shown. There are various commands implemented for data insertion, data deletion, distance based analysing etc. It can be seen from the figure3.2 that no implementation is defined for ICommandExecutor. This is because the modules consuming this core module library can implement according to their need. Similarly for the IProgress, modules can implement their own version depending on their type of the application. The parameters of the command are supplied when creating the instance of the object. The command object constructor contains all the necessary arguments needs for execution at later stage.

3.1.3 Webapp

The web application is built on Spring framework. It comes with IoC container which can easily pass in the database connection to the core library being consumed. In this module, the progress and command executor interfaces are implements to the needs of the web application. The progress interface implementation uses WebSocket for sending the update of how much work has been completed by the command. WebSocket is full-duplex communication channel over TCP



<<Interface>>
IProgress
+ update(int done, int out of, String msg): void
ProgressImplementation
+ update(int done, int out of, String msg): void
Use
Use
<<Interface>>
ICommand<T>
+ execute(IProgress progress): T
Use
<<Interface>>
ICommandExecutor
+ execute(ICommand<T> command): T
GridAnalyseColumnCommand
+ execute(IProgress progress): Collection<CompiledRegression>
Use
<<Interface>>
IDataService
+ findAll(): ArangoCursor<Data>
+ query(String query, Class<T> entity): ArangoCursor<T>
DataService
+ findAll(): ArangoCursor<Data>
+ query(String query, Class<T> entity): ArangoCursor<T>

Figure 3.2: UML diagram of the command pattern in the core module

which allows server and client to send messages to each other asynchronously.

The application maps the REST APIs to the Commands in the core module. The UI of the web application uses AngularJS which is a client side front end web framework. AngularJS framework allows to makes calls to the REST Endpoints and also act accordingly to the messages received by the server through WebSocket.

3.1.4 Command Line

The command line application also uses the core module to do similar task. The mapping is from command line arguments to core module's command. In this module, as well a different implementation is done for the progress and command executor interfaces as this time the diplay and execution is in commandline. Since the core module uses IoC container. Spring-context was added and application context was created at the start of the applicaton. The arguments were parsed via the JCommander library.

3.2 Optimizations

In this section, we discuss Optimizations that were done to speed up the computations.

3.2.1 Grid clustering algorithm

As per the pseudocode1, it can be seen that the Time complexity is $O(N\log(N)$ which was due to the sorting performed on the coordinate of the box calculated for each data point. But as one can see from the pseudocode the two N iterations were performed. First for calculating the coordinate of the $k - cube$ they belong to and then for the grouping the datapoints into the result. This result is then used for either finding violation of functional relation or for analysing the relation between the parameters.

The two extra N iterations were optimised by the ArangoDB by dividing the 3 iterations into 2 nodes. First node sorted the data points by calculating the coordinates of the $k - cube$ on the go and attaching it temporarily to the dataset for reuse if the data set is accessed again by the sort. After sorting the datapoints with its coordinates are send to the second node. The second node then groups the data on the fly as the groups are accessed. This is because the all the groups with equal coordinate lies next to each other. The groups and the data coming from the second node is pipelined to our computations that are performed on the nodes.

It is impossible to parallelize this algorithm because of its aggregating nature, where one point cannot be in more than one cluster. But due to its $O(N \log(N))$ complexity, overall the algorithm is quite fast and linear.

3.2.2 Distance based clustering

In the distance based clustering2, we calculated the time complexity to be $O(N^2$. This was due to nature of the algorithm to find neighbouring data points for every data point in the dataset.

It is not possible to reduce the first outer iteration, as we have to go through each data point for analysing the whole dataset. The second one, it is not required to go through the entire data points as long as we can find all the points that are close to each other within some tolerance. This makes it possible for optimisations. One of the approaches is to save N^2 distances between each and every data point and then making it easier for all the subsequent queries. This is very practical and blazingly fast as only one time slow operation is required for saving all the distances between each and every point. Then finding neighbours is a trivial task of going through the entire edges and getting all those points where the distance is less than the specified tolerance. This is practical for smaller dataset but due to the square space complexity. But as the number of records get larger it becomes in efficient to not only run the one time insertion but also storing the same.

So, another way was chosen in which indexes were created to fasten the search to find the neighbours. We filter the data with conditions such that data points whose possibility of becoming neighbour is zero is filtered before the distance is actually and calculate and confirmed.

We use the fact that $dist(v, u) \geq 0$ always where v, u are two data points vectors in space.

Let the tol be the max distance (tolerance) allowed between two data points. Let u be the data point vector with dimension k whose neighbours are to be found. Let v be any vector in the dataset.

Then for any value of:

$$v[i] > u[i] + tol \vee v[i] < u[i] - tol$$

where $i \in [0, k - 1]$

$$\implies dist(u, v) > tol$$

The reason is if any one of the values in vector v is more than tol units away then because the distance is added from all the other values of the vector as well. The total distance will definitely be more than tol.

But this filtering method is still not good as we still have to go through all the dataset to check whether any of the value is greater than tol or not. ArangoDB provides APIs to create indexes on the fields of documents for faster access to documents. It provides many different types of indexes such as Hash index, skip list index, full text index etc. Each index uses a special data structure to fasten different types of searches. As we are interested in range based queries. Skip list comes into play. Skiplist maintains a separate linked heirarchy of ordered values. They fasten up the equality lookups, range queries and sorting. The skiplist is added on to every parameter of the dataset and then when the filtering out condition is used before calculating the distance. The number of iteration will drop from N to the $Amortised - N$.

Another performance improvement that is done is by parallelising the outer loops. As ArangoDB supports multiple reads at the same time and the inner loop has no dependency on other iteration. Outer loop is parallelised which further decreases the running time by the number of threads the CPU can handle at a time.

Chapter 4: **Testing and Evaluation**

Chapter 5: **Conclusions and Future Work**

Bibliography

[1] David Beazley. Understanding the python gil. In *PyCON Python Conference. Atlanta, Georgia*, 2010.

[2] Frank Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, pages 37–42. ACM, 2000.

[3] W Lloyd Bircher and Lizy K John. Complete system power estimation: A trickle-down approach based on performance events. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 158–168. IEEE, 2007.

[4] William Lloyd Bircher and Lizy K John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61(4):563–577, 2012.

[5] Benoit Dupire and Eduardo B Fernandez. The command dispatcher pattern. In *8th Conference on Pattern Languages of Programs (PLOP 01)*, 2001.

[6] C Gilberto and M Margaret. Power prediction for intel xscale processors using performance monitoring unit events power prediction for intel xscale processors using performance monitoring unit events. In *ISLPED*, volume 5, pages 8–10, 2005.

[7] Jim Hugunin. Python and java: The best of both worlds. In *Proceedings of the 6th international Python conference*, volume 9, pages 2–18, 1997.

[8] Kenneth Obrien, Ilia Pietri, Ravi Reddy, Alexey Lastovetsky, and Rizos Sakellariou. A survey of power and energy predictive models in hpc systems and applications. *ACM Computing Surveys (CSUR)*, 50(3):37, 2017.

[9] Mario Pickavet, Willem Vereecken, Sofie Demeyer, Pieter Audenaert, Brecht Vermeulen, Chris Develder, Didier Colle, Bart Dhoedt, and Piet Demeester. Worldwide energy needs for ict: The rise of power-aware networking. In *Advanced Networks and Telecommunication Systems, 2008. ANTS'08. 2nd International Symposium on*, pages 1–3. IEEE, 2008.

[10] Huigui Rong, Haomin Zhang, Sheng Xiao, Canbing Li, and Chunhua Hu. Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674–691, 2016.

[11] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[12] Shu Yang, Zhongzhi Luan, Binyang Li, Ge Zhang, Tianming Huang, and Depei Qian. Performance events based full system estimation on application power consumption. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 749–756. IEEE, 2016.

[13] Robert Zembowicz and Jan M Zytkow. Testing the existence of functional relationship in data. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, volume 93, pages 102–111, 1993.