Monitoring and Predicting Linux Server Performance With Linear Regression

Conference Paper · January 2018

DOI: 10.15308/Sinteza-2018-68-73

3 authors:

Pavel Barca

1 PUBLICATION **0** CITATIONS

SEE PROFILE

Bojan Vujanic

2 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Nemanja Maček

Visoka škola elektrotehnike i računarstva strukovnih studija u Beogradu

62 PUBLICATIONS **115** CITATIONS

SEE PROFILE

MONITORING AND PREDICTING LINUX SERVER PERFORMANCE WITH LINEAR REGRESSION

Pavel Barca[1]*,
Bojan Vujanić[1],
Nemanja Maček[2]

[1]Singidunum University, Belgrade
[2]School of Electrical and Computer Engineering of Applied Studies, Belgrade

Abstract:
This paper presents an approach to develop smart monitoring system for Linux server based on linear regression. Technologies used to develop our system, i.e. create usable monitoring dashboards, are Telegraf, Prometheus and Grafana. In this paper we have focused on using prediction of the linearly dependent variables such as server load, disk space and similar system variables that change throughout time.

Keywords:
linear regression, prediction, monitoring, performance, Linux.

1. INTRODUCTION

The research presented in this paper deals with linear regression used to predict different parameters of Linux server. Telegraf [1] is gathering data from the Linux system and pushing it to the Prometheus [2] client which is then exposing data on specific path (default it is /metrics), in order to let Prometheus server scrape that data. Once Prometheus server scrapes /metrics path, it is storing data in local database. We can use and manipulate this data through GUI that Prometheus provides. In Prometheus GUI we can calculate everyday useful statistical variables from our data, as well as we can create custom functions that can calculate values for more specific purposes. Prometheus also offers monitoring functions integrated with alerting system with custom functions that make the system highly customizable and usable in almost any case that you can think of.

Even though Prometheus GUI is sort of intuitive and highly customizable from the data point of view. From visualization point of view it lacks some basic esthetics and visualization functions to make it useful for screen presenting and data visualization. This is where Grafana [3] steps in, with its' customizable visual representation and esthetically pleasing data visualization. That's why we have also used Grafana to query Prometheus data and show it on the dashboard just to make it more visually pleasing.

However, main focus in this paper is dedicated to Prometheus, since Prometheus is the tool that we have used for linear prediction and that makes all decisions on whether alert should be sent to administrator or not. Grafana and other tools are only used for convenience and data gathering.

Correspondence:
Pavel Barca

e-mail:
pavel.barca@outlook.com

DOI: 10.15308/Sinteza-2018-68-73

2. THE THEORY BEHIND REGRESSION ANALYSIS

The term regression dates back to 19th century where it was forged by British academic Sir Francis Galton. It was used to describe certain biological phenomenon. Regression analysis [4], on the other hand, is a mathematical statistical method used to investigate data and derive relations between variables. It is one of the oldest topics in mathematical statistics, dating back a few hundred years ago.

One of the first forms of linear regression [5] was the least squares method [6]. It was published by French mathematician Adrien-Marie Legendre in 1805, as well as by German mathematician Johann Carl Friedrich Gauss in 1809.

This method was applied by both mathematicians on the problem of calculating orbits of the celestial bodies around the sun. Later in his work, Gauss published additions to his theory of the least squares method in 1821, and also included a version of what we refer today as the Gauss-Markov theorem [7].

Regression analysis is mathematical method used to find correlation between input variable arrays called predictors – also referred to as independent variables – and output variable arrays called predicted variables or dependent variables. There are few types of regression depending on the size of these arrays as well as on the relationship of the instances of the given dataset: simple linear regression, multiple linear regression, multivariable linear regression and non-linear regression.

Simple Linear Regression

Simple linear regression [8] is one of the simplest linear regression forms. It is a linear regression model with only one independent variable and one dependent variable. Our outcome function is depending only on our independent variable, which means that once we have chosen our regression line, prediction will be made by using coefficients (constants) and input variable to predict output variable.

In simple linear regression we have independent variable (predictor) denoted as X and dependent variable denoted as Y. As we have mentioned before, there is only one independent and one dependent variable and relation between those two is given as:

$$Y = \alpha X + \beta \quad (1)$$

where β is the interception with y axis, and α is the slope of the regression line. There are few different methods to determine aforementioned parameters. Most common, as well as one of the first ones, is the called least squares method. Let's presume that we have the data that is consisting of ten (x_i, y_i) pairs, where x and y-axis denote time and load, respectively. To fit our model, at first we need to get optimal line (regression line) which we could use to predict load in the future. This can be done pretty easy with simple linear regression. To determine slope, as first we need to calculate the mean both for x and y values. Once we have calculated these means, we need to calculate difference between each coordinate from corresponding mean and use following equation to calculate slope β:

$$\beta = \Sigma\left((x_i - x_m)(y_i - y_m)\right) \ / \Sigma(x_i - x_m)^2 \quad (2)$$

where x_m and y_m represent means for corresponding coordinates from 10 pairs that we have used as dataset for fitting.

To determine α we need to use previously calculated x_m, y_m and β. When we add these values into (1) we get the simple equation (3)

$$\alpha = (Y - \beta) \ / X \quad (3)$$

Multiple Linear Regression

The difference between simple and multiple linear regression [9] is that in multiple linear regression we use more than one explanatory variable (independent variable) while still using only one dependent variable.

In simple linear regression we had simple equation, while in multiple we need to have a little bit complex equation due to more than one explanatory variable. One of the use cases for this type of linear regression would be predicting load on servers, e.g. monitoring Web server where sometimes it's not enough just to monitor load over time. It could be much more effective to use more than one variable that can explain dependent variable in this case load. As an example, number of requests and number of SSL [10] connections during the certain period of time can improve our monitoring and prediction of load due to the fact that server handling SSL requests can take a lot of resources for SSL handshake.

In multiple linear regression we have multiple independent variables (predictors) denoted by X_1, X_2,,..., X_k and we still have only one dependent variable Y. In this case our regression line is given by:

$$Y = \alpha_1 X_1 + \alpha_2 X_2 + \ ... \ + \alpha_k X_k + \beta \quad (4)$$

In the equation (4) we can notice that we have the parameter α_i i for each X_i and we have one parameter β which represents interception with *y* axis.

Multivariable Linear Regression

Multivariable linear regression [11] or general linear regression is specific form of linear regression where we have more than one dependent variables, i.e., we are predicting more than one variable. In fact we can write regression line as:

$$\mathrm{Y} = \alpha \mathrm{X} \quad (5)$$

where Y is the matrix of the dependent variables, α is matrix of parameters that need to be determined and X is matrix of independent variables. Equation (5) is self-sufficient explanation on why this type of linear regression is also called general linear regression – if not so, one might notice that it is due to the fact that from this equation you can get both of the previous equations (1) and (4) in special cases when we limit number of elements in these matrices.

Non-Linear Regression

Even though this isn't the topic of this paper we are going to mention basic differences between linear and non-linear regression [12]. While in linear we had regression line (equation) that was linear function between independent variables and dependent ones, and also parameters that we would try to determine for the optimal line, in non-linear regression we are still searching for optimal equation that would best describe training data. However, they differ as there is no restriction to linear function as in linear regression. In other words regression "line" is no longer linear function – it can be any other function that best describes relation between independent and dependent variables in the dataset that we have use for fitting.

Another difference is clear when the equation used in general non-linear regression is paid attention to:

$$\mathrm{Y} = \mathrm{f}\ (\alpha, \mathrm{X}) \quad (6)$$

where Y denotes the matrix of the dependent variables, α is the matrix of parameters that we are trying to determine and X is matrix of independent variables (explanatory variables) same as in (5). However, with non-linear regression, we can use non-linear function to suffice f().

An example of these functions used in biochemistry is Michaelis-Menten equation [13] which is one of the most famous models in enzyme kinetics. It is named after German scientist Leonor Michaelis and Canadian scientist Maud Leonora Menten. This equation is suitable example for non-linear regression due to the fact that it is used to describe the rate of enzymatic reactions. It's given as:

$$v = \mathrm{d}[P]/\mathrm{dt} = \left(V_{\max}[S]\right) \ / \ \left(K_m + [S]\right) \quad (7)$$

where v is reaction rate – rate of formation of product [P] –which is the variable that we are trying to predict. $V_{\max}$ denotes the maximum rate noticed by the system at saturating substrate concentration, while [*S*] denotes substrate and K_m substrate concentration at which reaction rate is half of the $V_{\max}$, which is also called Michaelis constant [14].

3. SETTING UP THE SYSTEM

Setting up Telegraf

As we have previously mentioned, we are using Telgraf to gather and expose system data so that Prometheus server could scrape that data.

Telegraf service has already integrated plug-ins for gathering system data as well as for exposing data that was gathered for different use-cases. There are four different plug-in types. We are only using two types in this research: input plug-ins that are used to gather data, and output plug-ins that are used to push or expose gathered data to other service that can utilize it.

We have used input plug-ins for getting system data and output plug-in for exposing this data, so that Prometheus could scrape data in question. Telegraf service configuration is given in the following code snippet:

```
# Get kernel statistics from /proc/stat
[[inputs.kernel]]
 # no configuration
```

```
# Read metrics about memory usage
[[inputs.mem]]
 # no configuration
# Get the number of processes and group them
#by status
[[inputs.processes]]
 # no configuration
# Read metrics about swap memory usage
[[inputs.swap]]
 # no configuration
# Read metrics about system load & uptime
[[inputs.system]]
 # no configuration
# Write (expose) metrics so that
#Prometheus server could scrape it.
[[outputs.prometheus_client]]
 # Prometheus server will need to
 #scrape this host and port
 listen = ":9273"
 path = "/metrics"
 expiration_interval = "60s"
```

Now we have exposed all data that has been gathered by Telegraf on *localhost:9273/metrics*. If we visit that page, we should expect to retrieve data similar to:

```
# HELP cpu_usage_guest Telegraf collected metric
# TYPE cpu_usage_guest gauge
cpu_usage_guest{cpu="cpu-total",host="parrot"} 0
cpu_usage_guest{cpu="cpu0",host="parrot"} 0
cpu_usage_guest{cpu="cpu1",host="parrot"} 0
cpu_usage_guest{cpu="cpu2",host="parrot"} 0
cpu_usage_guest{cpu="cpu3",host="parrot"} 0
...
```

Setting up Prometheus

Once we had exposed data that Telegraf has gathered, the next step is to configure Prometheus server and make sure that Prometheus can scrape endpoint that we have exposed within Telegraf configuration.

Prometheus is using yaml (.yml) file format for configuration file. YAML stands for Yaml Ain't Markup Language. In our case we have used pretty simple configuration with only one host:

```
global:
 # By default, scrape_interval is 1m.
 scrape_interval: 5s
 evaluation_interval: 15s
 # Attach these labels to any time series
 # or alerts when communicating with
 # external systems (example Alertmanager).
 external_labels:
  monitor: 'parrot-monitor'
alerting:
 alertmanagers:
 - static_configs:
  - targets:
rule_files:
 - "free_space.yml"
# Here it's our testing host with parrotOS.
scrape_configs:
 - job_name: 'parrot-System'
  # Override the global default.
  scrape_interval: 5s
  static_configs:
   - targets: ['192.168.0.13:9273']
    labels:
      group: 'production'
```

Once we had setup Prometheus server, we have fully functioning main part of our monitoring system. All gathered data from Linux server is located within Prometheus local database. We can analyze and visualize this data from GUI. As well as create functions for linear prediction. Let's visualize current data for free space on one of the drives since we can simulate this pretty easily with USB drive.

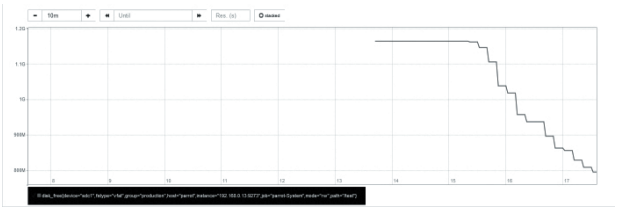

Fig. 1. *Real free space values for /test partition in last 10 m*

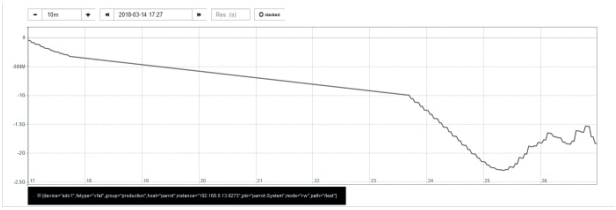

Fig. 2. *Linear prediction of free space for / test partition in next 10m*

As depicted on Fig. 1, our drive is being filled with data at really alarming speed. We got data for past 10 minutes and we used it to predict free space value in next 10 minutes as reader can see on Fig. 2 – we are in the negative zone, which means that our drive is being filled at really high speed and we will soon reach out of free space. This points up to a fact that it would be a good time to alert administrator e.g. at 90% of the free space, or 30 minutes before we have predicted full storage.

Seting up Prometheus Alert

Setting up Prometheus alert was probably the easiest task at this point since the most extensive task is analyzing data and figuring out correct thresholds for different variables. In this case we have used configuration where we are checking past 10 minutes of the data to determine free space value in the next 10 minutes and if this value is below 10% of the total disk space then we wanted to receive alert from Prometheus Alert Manager.

```
groups:
- name: free_space
  rules:
  - alert: Disk_at_90_percent_in_10m
    expr: predict_linear(disk_
free{device=”sdc1”,path=”/test”}[10m],10*60) <
scalar(disk_total{device=”sdc1”,path=”/test”})*0.1
    for: 1m
    labels:
      severity: page
```

As previously mentioned with Prometheus, we can create any function that we need for monitoring or visualization of our data.

Grafana integration

As we have already mentioned, Grafana is very intuitive and esthetically pleasing (see Fig. 3), as well as very easy and simple to integrate. When you login first time to Grafana, you get prompted setup wizard. So it is pretty simple to integrate it with Prometheus as everything is click based – at least during setup. When Prometheus is setup as endpoint from which Grafana will query data, we can go and create hundreds and thousands of dashboards for our data.

Although Grafana is typically used for esthetic purposes, Grafana is quite powerful tool since it has also its own alert manager, which can be utilized in a lot of use-cases even with Prometheus Alert manager.

Fig. 3. *Grafana dashboard for free space on drives*

4. TOOLS USED IN THE RESEARCH

Telegraf

Telegraf is an agent used to gather, process and push (expose) data from different systems. It is written in language Go. It is really resource low-cost due to using plugins that define what data should be gathered and how it should be exposed or pushed. This way administrators can determine if some of the features need to be used or not.

Prometheus

Prometheus is a monitoring tool that is used by almost every company that is currently working with cloud technology. It is tool that provides high range of features for data analysis, visualization, prediction and much more. It is pretty efficient due to it being written in Go. It also has integrated Alert Manager which makes this tool really useful for creating alerts that can be used with other tools for direct notification or just email notifications.

Another bonus-point for Prometheus is the way it stores data. It is done in memory as well as on local disk in custom format. It is highly scalable due to efficient sharding and federation.

Even though it doesn't have really esthetically pleasing visualization it has really easy Grafana integration which makes this tool one of the high-end tools in monitoring at this time.

Grafana

Grafana is tool mostly used for visualization of the data that is gathered by other tools like Telegraf or Prometheus. It is highly intuitive tool. And most of the integrations are done by point-and-click instead of writing configuration files, which causes Grafana to be used among small and medium IT companies as well as giants among the cloud market.

Main processing part of Grafana is written in programming language Go, while front end is mostly written in Node. Grafana has also integrate its own Alert Manager which makes this monitoring tool even more interesting for cloud businesses

5. CONCLUSION

Cloud services are evolving each day. Big companies are working on smart algorithms and smart services for everyday use. Their goal is to utilize machine learning algorithms and artificial intelligence on day-to-day basis even for simple stuff like driving to work or rating and suggesting application to customers, based on their preferences. All these services as well as most of the others that are currently being used or developed require high number of servers as well as big storage for all the data that is being gathered to be stored. Once data is stored servers need to analyze this data and derive a conclusion for further application. These tasks are highly resource intensive and as such need to be monitored constantly and administrator team need to be notified as soon as anomaly is detected on that system.

This paper presents an approach to smart monitoring system for cluster of servers. Or any number of hosts that need to be monitored for continued and effective performance. Even though we have used basic metrics and most of the data is from one host that has Linux operating system. Setup described in this paper can be scaled to undefined number of hosts as well as for other metrics. Telegraf is gathering data from the hosts and pushing or exposing data so that Prometheus could scrape this data. Telegraf is not necessary since Prometheus can scrape nodes on its own. However for maximal utilization we used Telegraf. Once Prometheus scrape data in question it stores it locally and can start analyzing and manipulating data as well as checking if there are any anomalies that should be reported to administrator. For the esthetic visualization of this data we used Grafana because it has high number of data visualization features as well as simple integration with Prometheus.

REFERENCES

[1] https://www.influxdata.com/time-series-platform/telegraf/, last time visited March 18, 2018.

[2] https://prometheus.io/, last time visited March 18, 2018.

[3] https://grafana.com/, last time visited March 18, 2018.

[4] J. Fox, Applied regression analysis, linear models, and related methods, Sage Publications, Inc, 1997.

[5] G. Seber, A. J. Lee, Linear regression analysis, vol. 329, John Wiley & Sons, 2012.

[6] E. Hinton, J. S. Campbell, "Local and global smoothing of discontinuous finite element functions using a least squares method", Int. J. Numerical Methods in Engineering, vol. 8, no. 3, 1974, pp. 461-480.

[7] J. S.Chipman, "Gauss-Markov theorem", Int. Encyclopedia of Stat. Science, Springer Berlin Heidelberg, 2011, pp. 577-581.

[8] S. Chatterjee, A. S. Hadi, "Simple linear regression", Regression Analysis by Example, 4th ed., pp. 21-51, 2006.

[9] G. Grégoire, "Multiple linear regression", European Astronomical Society Publications Series, vol. 66, 2014, pp 45-72.

[10] D. Wagner, B. Schneier, "Analysis of the SSL 3.0 protocol", 2nd USENIX Workshop on Electronic Commerce Proceedings, vol. 1, no. 1, 1996.

[11] D. Zelterman, "Multivariable Linear Regression", Applied Multivariate Statistics with R, pp. 231-25, Springer, Cham, 2015.

[12] R. E. Kass, "Nonlinear regression analysis and its applications", Journal of the American Statistical Association, vol. 85, 1990, pp. 594-596.

[13] B. P. English, "Ever-fluctuating single enzyme molecules: Michaelis-Menten equation revisited", Nature chemical biology, vol. 2, no. 2, 2006, pp. 87-94.

[14] G. L. Atkins, I. A. Nimmo, "The reliability of Michaelis constants and maximum velocities estimated by using the integrated Michaelis–Menten equation", Biochemical journal, vol. 135, no. 4, 1973, p. 779.