Design and Implementation of a Server Health Monitoring System

Quan Nguyen
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
qmn@mit.edu

Abstract

Computer architecture research at CSAIL requires significant computing resources. These resources are distributed across the Stata center and sometimes crash or run into anomalous conditions. Server problems have been hitherto discovered on an ad-hoc basis. In this paper, we present the design and implementation of Gibraltar, which simplifies checking the status of many machines at once. It tries to avoid external software dependences and tries to be secure. We present results and conclude.

1. Introduction

Compute- and memory-intensive simulations of future architectures form some of the pioneering research in the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology (MIT). In particular, the Sanchez group can carry out thousands of simulations at a time across a cluster comprised of dozens of servers. These servers, acquired over many years, may need attention because they run out of memory or get overloaded. Monitoring the health of these servers is crucial for research productivity

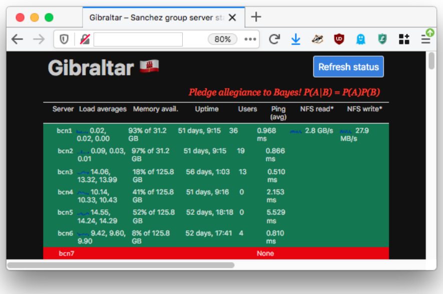
However, it is difficult to easily determine which servers need help unless someone notices – and when they do, it is often too late. Checking the server state is also time-consuming.

The COVID-19 pandemic [3] has exacerbated the need for remote server diagnostics, as students were no longer able to easily enter the server rooms to reboot any dead machines. Futhermore, it has become important not to overload the machines, lest they get so badly wedged so as to require a reboot, which can only be initiated by a person on-campus.

2. Prior Work

libstatgrab [1] has already implemented software for checking the status of many machines. One crucial difference in their approach is that they rely on each machine having a program installed. Instead,

DOI: http://dx.doi.org/10.1145/(to come)



Pledge allegiance to Bayes! $P(A|B) = P(A)P(B)$

Server	Load averages	Memory avail.	Uptime	Users	Ping (avg)	NFS read*	NFS write*
bcn1	0.02, 0.02, 0.00	93% of 31.2 GB	51 days, 9:15	36	0.968 ms	2.8 GB/s	27.9 MB/s
bcn2	0.09, 0.03, 0.01	97% of 31.2 GB	51 days, 9:15	19	0.866 ms		
bcn3	14.06, 13.32, 13.99	18% of 125.8 GB	56 days, 1:03	13	0.510 ms		
bcn4	10.14, 10.33, 10.43	41% of 125.8 GB	51 days, 9:16	0	2.153 ms		
bcn5	14.55, 14.24, 14.29	52% of 125.8 GB	52 days, 18:18	0	5.529 ms		
bcn6	9.42, 9.60, 9.90	8% of 125.8 GB	52 days, 17:41	4	0.810 ms		
bcn7					None		

Figure 1. Screenshot of the Gibraltar web interface.

Gibraltar avoid external dependencies by relying on tools known to be installed on CSAIL Ubuntu. Also, I already did a ton of the work before I found this. Their web interface also lacks snarky mottos.

3. Design

Gibraltar provides vital statistics of the entire cluster in one screen, including average load, memory availability, users logged in, and ping. This information can help determine who might be causing excess server load or figure whether a machine has crashed.

Gibraltar's design balances several competing needs:

- keep current information,
- place low requirements on servers, and
- maintain some semblance of security.

3.1 Reduce server requirements

To minimize the burden of other machines, Gibraltar uses a *polling* approach: the Gibraltar process running on a separate server logs into each machine in a periodic interval (say, every 30 minutes), recording the statistics of each machine. Polling machines eliminates the need for specially-installed software to communicate back to the Gibraltar server. Instead, we rely on system utilities already installed by default on each server, such as the `w` command or checking the `/proc` filesystem. We pipe the polled server's output back to Gibraltar for parsing and storage in log files. By being as unintrusive as possible, we not only reduce load on these servers but also avoid unnecessary software dependencies. This approach also avoids the need for a special daemon, or a special user, especially on machines that are peripherally managed by TIG.

3.2 Maintain some semblance of security

We wanted to avoid creating additional users, which could introduce security vulnerabilities and complicate maintenance. So, all server information collecting is carried out on behalf of my user. Users typically connect to CSAIL servers through Kerberos, but Kerberos tickets have a limited lifetime, even when using the `krenew` command. Instead, I have specially-designated SSH keys, while passwordless, are not allowed to execute commands other than exactly the server-checking scripts.

4. Implementation

The operation of Gibraltar can be divided into two main components: *fetching* the data and *presenting* it. Most users are generally concerned with presentation, but those looking to maintain Gibraltar (if I ever leave MIT, this will probably fall to Yifan) will be interested in how Gibraltar processes its data.

4.1 Fetching and processing data

Gibraltar's implementation was carefully planned from the start to result in a steadily growing tangled mass of Python and Bash scripts. A single Bash script executes the following Python scripts:

```
            _  _                 _  _
   __ _ (_) | |__   _ __   __ _ | | | |_   __ _  _ __
  / _` | | | | '_ \ | '__| / _` | | | | __| / _` | | '__|
 | (_| | | | | |_) | | |   | (_| | | | | |_ | (_| | | |
  \__, | |_| |_.__/  |_|    \__,_| |_|  \__|  \__,_| |_|
  |___/

            "Guarding over the Sanchez cluster since 1713"

Cluster status:
Server          Load averages             Uptime Users       Ping   NFS read  NFS write
------------------------------------------------------------------------------------------
bcn1          0.02, 0.02, 0.00    51 days,  9:15    36   0.968 ms   2.8 GB/s  27.9 MB/s
bcn2          0.09, 0.03, 0.01    51 days,  9:15    19   0.866 ms
bcn3       14.06, 13.32, 13.99    56 days,  1:03    13   0.510 ms
bcn4       10.14, 10.33, 10.43    51 days,  9:16     0   2.153 ms
bcn5       14.55, 14.24, 14.29    52 days, 18:18     0   5.529 ms
bcn6          9.42, 9.60, 9.90    52 days, 17:41     4   0.810 ms
bcn7                                                   unreachable
...
```

Figure 2. Sample telnet interface output.

- `gibraltar.py`, which tests each server in succession, first pinging for liveness. Once the script establishes the server is alive, it logs in and retrieves server information using `server_status.sh`, which obtains information like currently logged in users, memory usage, and file system performance. It aggregates each server's information into a JSON file stored on Gibraltar.
- `condor-status.sh`, which counts the number of jobs currently running as part of a Condor cluster.
- `sparklines.py` and `loadplot.py`, which uses Matplotlib [2] to generate plots to get at-a-glance information when viewing the web interface.
- `slack_dispatch.py`, which detects anomalous conditions (discussed later) to send a message to the group's Slack organization to alert users.

The server checking Bash script, as well as the telnet interface (described next), are managed as systemd services.

4.2 Data presentation

The most complete information is presented over a web interface hosted using lighttpd, shown in Figure 1. We also provide a telnet interface, as shown in Figure 2. From the web interface, we can easily initiate server status refreshes.

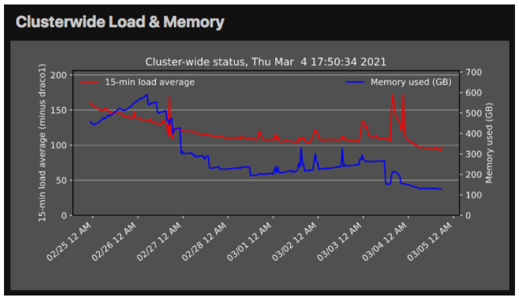


Figure 3. Cluster-wide CPU load and memory usage plot.

5. Results

Gibraltar occupies a pinned tab on our web browsers; we also aggregate server information to plot cluster-wide performance. From these plots, we can glean many interesting patterns. Figure 3 shows the 15-minute load average summed over all machines as well as the total memory used over the course of several days in February and March of 2021.

The plots demonstrate Gibraltar's usefulness; Figure 3 shows the effect of a slow memory leak caused by a long-running application in the form of a steadily rising blue line (not accounting from brief spikes resulting from new jobs sent to the cluster) from February 28 to March 3. The visual presentation of this information is not only greatly amusing but also actually somewhat helpful in diagnosing cluster issues.

6. Acknowledgments

The authors would like to thank the members of the Sanchez group who offered their valuable feedback. We especially recognize Victor Ying and Yifan Yang for submitting pull requests with improvements as well as suggesting new features. Finally, the authors would also like to thank their adviser who, perhaps like some citizens of his country, may or may not acknowledge the existence of Gibraltar.

References

[1] libstatgrab: a cross platform library for accessing system statistics. URL `https://libstatgrab.org/`.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: `10.1109/MCSE.2007.55`.

[3] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao, and W. Tan. A Novel Coronavirus from Patients with Pneumonia in China, 2019. *New England Journal of Medicine*, 382(8):727–733, 2020. doi: `10.1056/NEJMoa2001017`. URL `https://doi.org/10.1056/NEJMoa2001017`. PMID: 31978945.