EPJ Web of Conferences **214**, 07033 (2019) https://doi.org/10.1051/epjconf/201921407033
CHEP 2018

Monitoring virtual machines and containers with VacMon

Andrew McNab[1]

[1]School of Physics and Astronomy, University of Manchester, Manchester, United Kingdom

Abstract. At the start of 2017, GridPP deployed VacMon, a new monitoring system suitable for recording and visualising the usage of virtual machines and containers at multiple sites. The system uses short JSON messages transmitted by logical machine lifecycle managers such as Vac and Vcycle. These are directed to a VacMon logging service which records the messages in an Elasticsearch database. The records can be viewed graphically using the VacMon web portal, with selections based on time ranges, virtual organisations, sites, subclusters within sites, or individual hypervisors.

1 Introduction

Vac [1] and Vcycle [2] are systems for managing the lifecycles of virtual machines and containers, developed for GridPP by the University of Manchester. They are now managing resources at 10 sites and in 2017 we developed VacMon to monitor usage and the health of the overall system, individual sites, and individual physical hypervisors. Part of the motivation of the system is to allow some sites to replace locally-operated monitoring systems with VacMon, as part of a "Lightweight Sites philosophy" to simplify WLCG sites by reducing the number of services a site must run.

This paper describes the architecture of VacMon, including the JSON [3] messages sent to the central service, its Elasticsearch [4] database, and the web monitoring interface.

2 VacMon architecture

Figure 1 shows the VacMon architecture. Central to the Vac and Vcycle systems at sites are their vacd and vcycled daemons respectively, managing the lifecycle of the virtual machines and containers which execute the virtual organisations' workloads. Both daemons repeatedly cycle through the resources they are monitoring, act where appropriate, and then sleep. They can be configured to send VacQuery messages (section 3) to VacMon during each cycle. The messages are received by the central vacmond server (section 4), which applies some sanity checking and optional access control, before storing the messages in an Elasticsearch database (section 5). To view the status of the sites being monitored, the VacMon website (section 5) is provided by the vacmon-cgi script hosted by an Apache HTTP server [5].

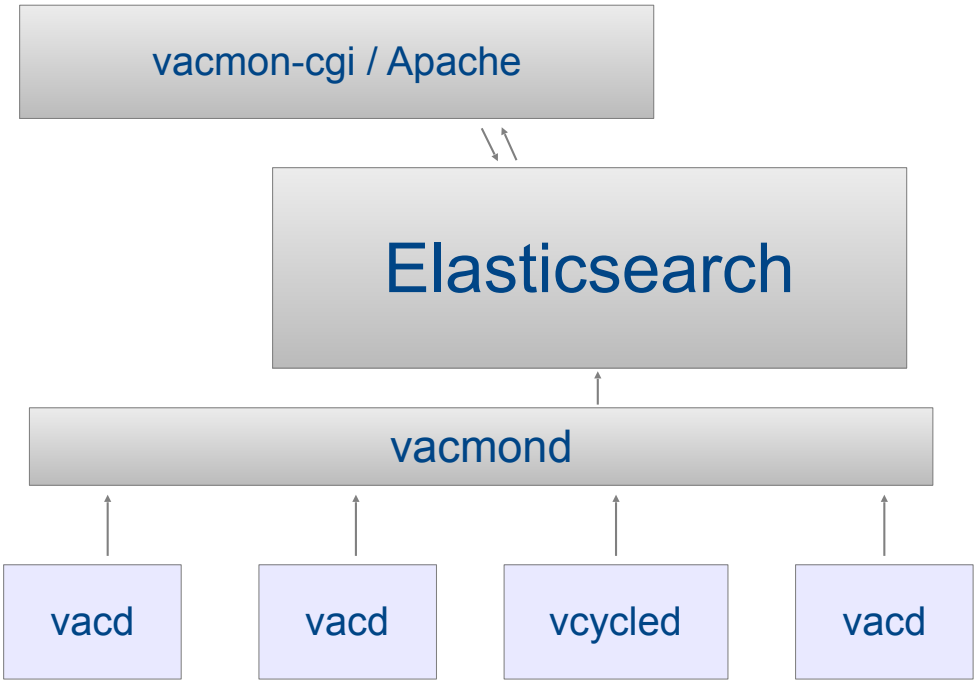

Fig. 1. The VacMon architecture, including vacd and vcycled daemons at sites, and the central vacmond, Elasticsearch and Apache services.

3 VacQuery messages

The format of the VacQuery messages has previously been specified in an HEP Software Foundation Technical Note [6] and discussed in a previous CHEP paper [7]. The messages are JSON documents sent in UDP packets to the vacmond server listening on port 8884. The message schemas are designed to keep the message size below about 1200 bytes to avoid fragmentation of UDP packets over ethernet networks with the common MTU of 1500.

For Vac sites, which consist of worker node class machines configured as autonomous hypervisors, the messages are sent from each hypervisor (or "factory"). Since these sites do not have a central headnode or manager, VacMon provides a central point at which monitoring information can be aggregated and viewed. For Vcycle sites, at which virtual machines (VMs) are managed on a cloud system like OpenStack [8], the vcycled daemon does have a complete picture of the site status but for simplicity the same approach is used.

There are three types of VacQuery messages, all of which are used by VacMon.

The factory_status messages give the status of the autonomous hypervisors (Vac) or central daemon (Vcycle). This includes version information about the daemon, operating

system, and kernel, health information such as free disk space and memory usage, and capacity information about the total number of processors managed and currently in use, and the benchmarked power of the machines.

The machinetype_status messages describe the characteristics of each type of VM or container which the factories know how to create, and the number of processors running with that machinetype. The messages also include the VOMS [9] Fully Qualified Attribute Name of the virtual organisation which owns that machinetype.

The machine_status messages are sent when a VM or container finishes execution, and contain information about the resources used and the outcome (success, or failure with the reason).

4 vacmond server

The VacQuery JSON messages will be stored in an Elasticsearch database which uses JSON as its native input format. Consequently it would be possible to expose the Elasticsearch port directly on the network and accept messages. Instead of this, VacMon provides a server daemon, vacmond, which is exposed on the external network and applies checks and transformations to the JSON documents before inserting them into the Elasticsearch database.

The VacQuery protocol supports the use of cookies as shared secrets as a basic defence against "spoofed" messages from attackers, and the vacmond server is a suitable place to check the validity of the cookies in messages sent to VacMon.

The vacmond server also transforms any legacy fields in messages into the current form to keep the database contents consistent. As the server is tied to the version of the VacQuery message schemas, it also has the role of creating the mappings given to Elasticsearch which initialise the indexes and the types (long int, date etc) of the fields. If this is not done before messages are passed to Elasticsearch, it will attempt to guess the types itself and these choices are not efficient or indeed not correct in some cases. Since only vacmond writes to the database, and initialises it correctly before hand if necessary, this problem is avoided.

5 Elasticsearch database

Elasticsearch has emerged as a flexible and scalable solution which can rapidly absorb a flow of documents. It has indexed JSON documents as its central concept, which matches very well with the VacQuery JSON messages.

To date, the Elasticsearch component of VacMon has been provided by a single Elasticsearch instance running in a VM with 20 GB of memory. This has dealt with reports about 8000 virtual machines at ten sites. If a single instance proves to be insufficient in the future, then Elasticsearch provides ways of clustering instances.

6 VacMon website

The VacMon website provides views of the data held by the Elasticsearch database. The highest level view shows all running processors over time, broken down by site. Successively lower level views can be obtained by selecting particular sites, particular spaces (clusters) within these sites, and eventually individual factories (which are hypervisors in the case of Vac sites). At the lowest level, charts of memory and disk usage are included, and at all levels charts breaking down the figures over time by virtual organisation are also shown. The time frame shown, with arbitrary start and end points can be chosen, down to the level of a range of hours and up to the level of months. Figure 2 shows the main "All sites" view.

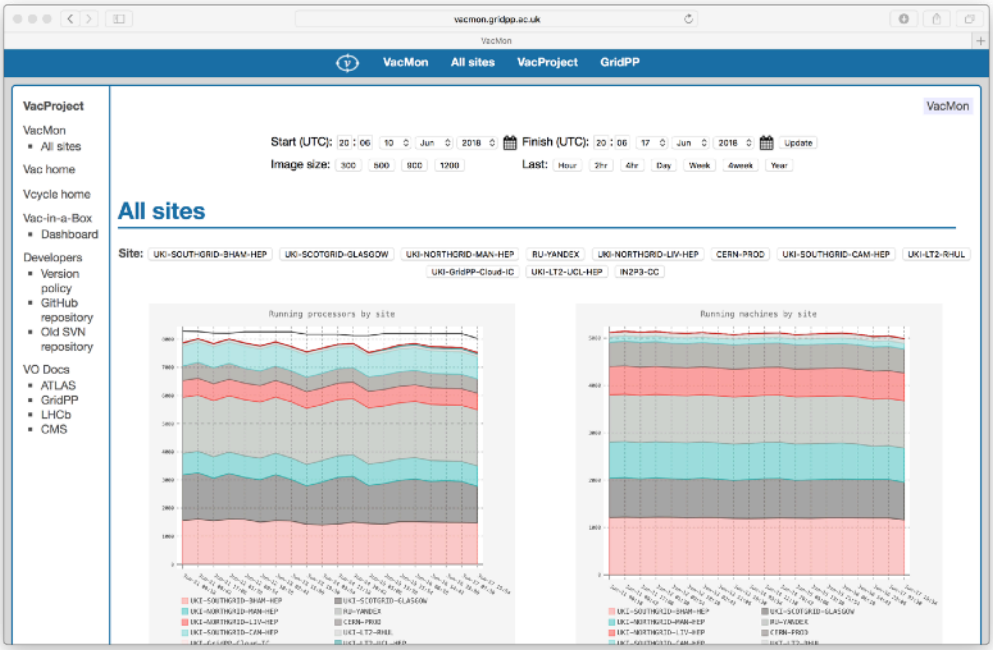

Fig. 2. The main view presented by the VacMon website.

The website application is a Python script using the PyGal [10] package to create embedded SVG images of the charts on demand. The appearance of the website is intentionally similar to Ganglia [11], with an accessible black on white colour scheme. This approach was chosen over using an off-the-shelf solution such as Kibana [12], for simplicity, security, speed, and the ability to bookmark or link directly to even the most complex views the site can generate.

Since VacMon was deployed, it has proved to be an invaluable source of information about the status of sites, and has been used by some sites to replace the functionality of

Ganglia or Nagios [13] but without the need to run equivalent services themselves, as part of their Light Weight Sites philosophy.

Figures 3, 4 and 5 show individual charts generated by the website.

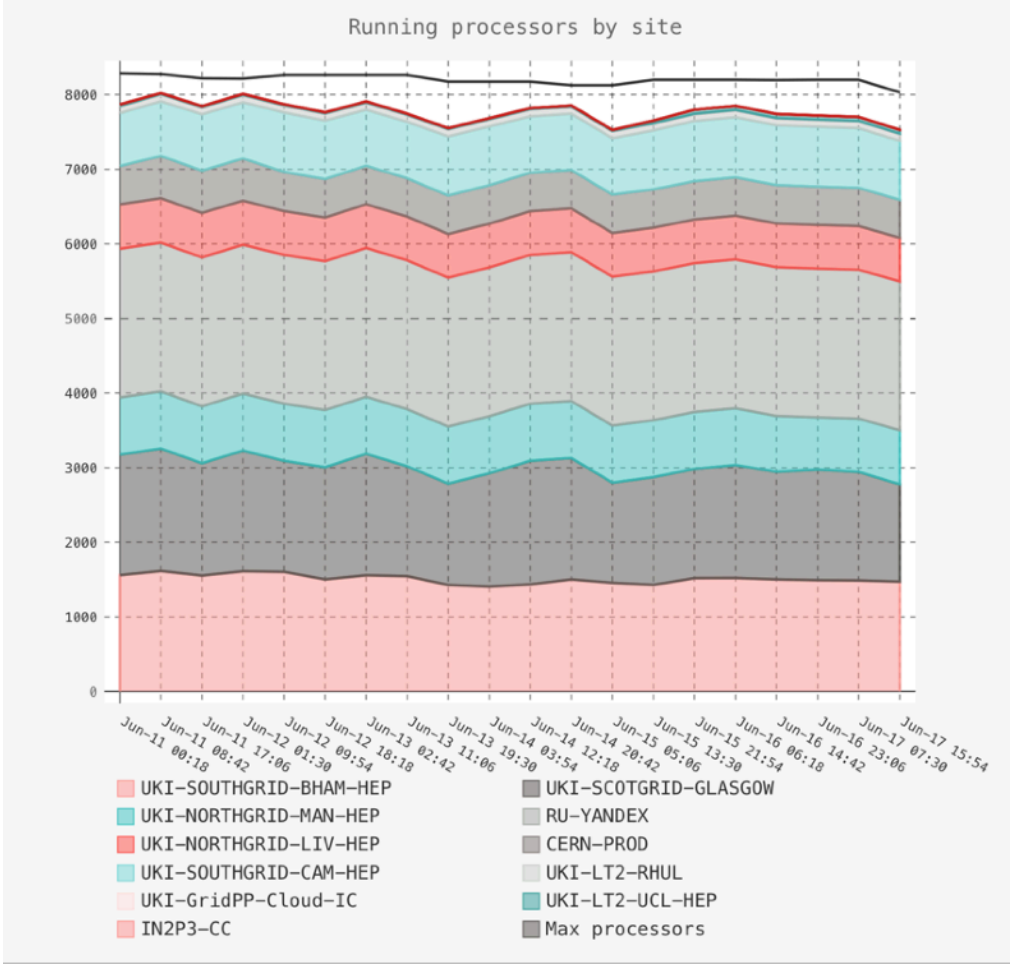

Fig. 3. The number of processors occupied by running VMs and containers at all participating sites, broken down by site.

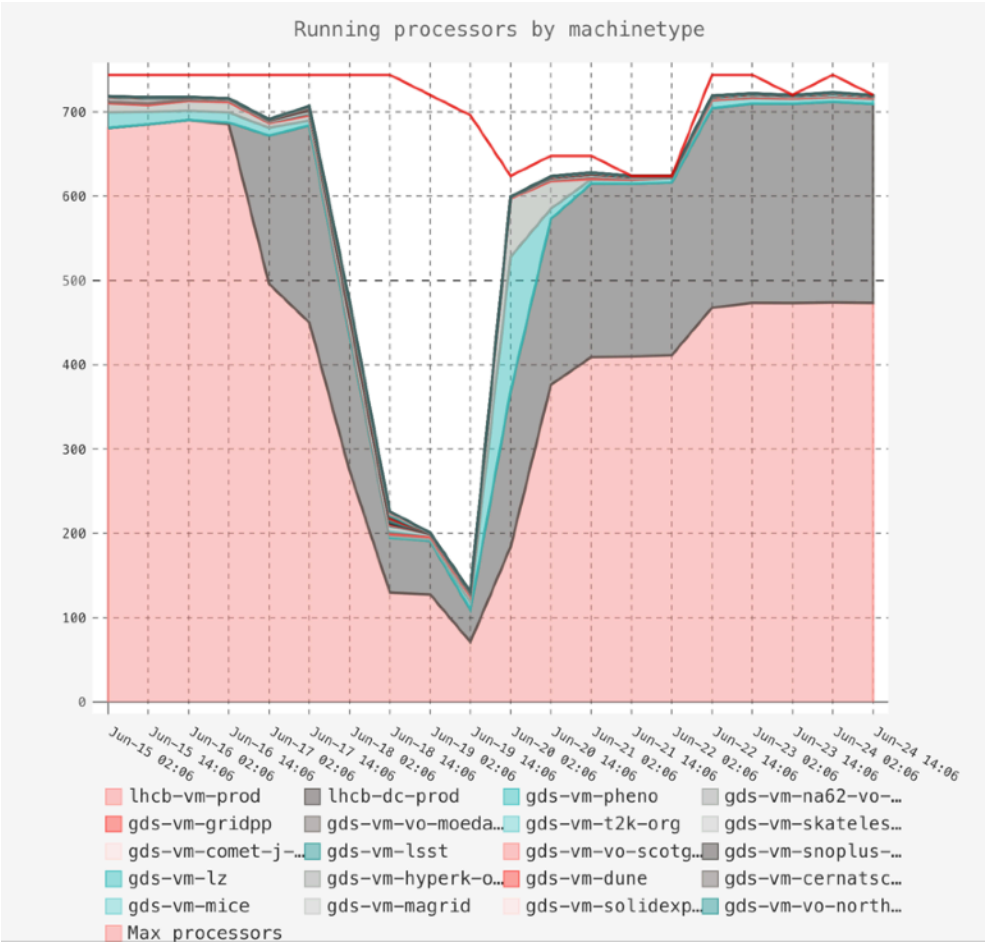

Fig. 4. The number of processors occupied by different types of virtual machine (“vm”) and container (“dc”) at one site before and after a software update.

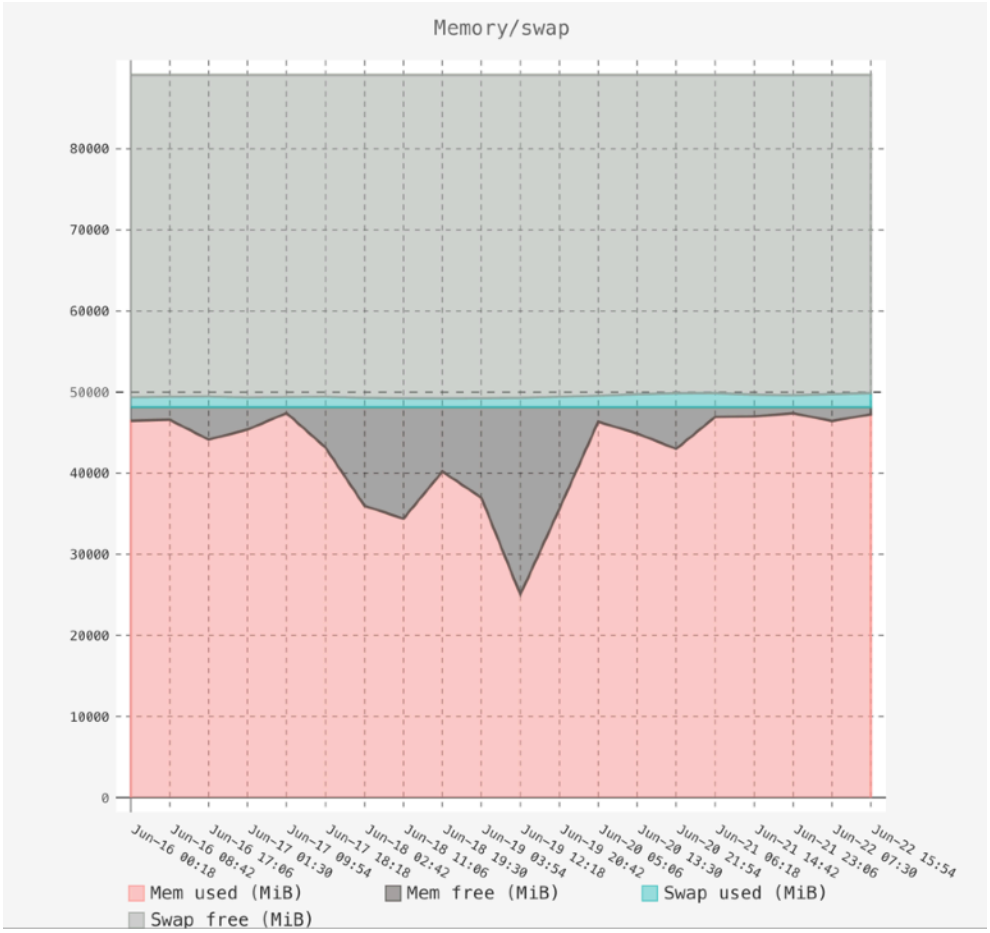

Fig. 5. Memory and swap usage on one physical node.

7 Summary

We have explained how VacMon has been designed to record and present monitoring information from Vac and Vcycle sites. The system uses a mixture of custom (vacmond, vacmon-cgi) and off-the-shelf (Elasticsearch) components as appropriate. In operation, VacMon has proved itself to be an invaluable monitoring tool, particular in the context of Light Weight Site models which aim to reduce the number of services a site must operate.

References

1. A. McNab, F. Stagni, M. Ubeda Garcia J. Phys.: Conf. Ser. **513** 032065 (2014)
2. P. Love, E. MacMahon, A. McNab J. Phys.: Conf. Ser. **664** 022031 (2015)
3. T. Bray *RFC 8259: The JSON Data Interchange Format* (Internet Engineering Task Force, 2017)
4. Elasticsearch https://www.elastic.co/products/elasticsearch
5. Apache HTTP server http://httpd.apache.org
6. A. McNab *HSF-TN-2016-04 The Vacuum Platform* (HEP Software Foundation, 2016)
7. A. McNab J. Phys.: Conf. Ser. **898** 052028 (2017)
8. OpenStack https://www.openstack.org
9. R. Alfieri et al https://arxiv.org/abs/cs/0306004
10. PyGal http://pygal.org
11. Ganglia http://ganglia.sourceforge.net
12. Kibana https://www.elastic.co/products/kibana
13. Nagios https://www.nagios.org