ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA134

MONITORING SYSTEM FOR IT INFRASTRUCTURE AND EPICS CONTROL SYSTEM AT SuperKEKB

S. Sasaki†, T. T. Nakamura, KEK, Tsukuba, Japan
M. Hirose, Kanto Information Service, Tsuchiura, Japan

Abstract

The monitoring system has been deployed to efficiently monitor IT infrastructure and EPICS control system at SuperKEKB. The system monitors two types of data: metrics and logs. Metrics such as a network traffic and a CPU utilization are monitored with Zabbix. The data stored in Zabbix are visualized on Grafana, which allows us to easily create dashboards and analyze the data. Logs such as text data are monitored with Elastic Stack, which provides log collection, searching, analysis and visualization. We have applied it to monitor broadcast packets in the control network and EPICS control system. In addition, we have developed the EPICS Channel Access client software that sends PV values to Zabbix server and the status of each IOC is monitored with it. We have also developed the Grafana plugin and API gateway to visualize the data from pvAccess RPC servers. Various data such as CSS alarm status data is displayed on it.

INTRODUCTION

SuperKEKB [1] is an asymmetric-energy electron-positron double ring collider. Its target luminosity is 8×10^{35} cm^{-2} s^{-1} which is 40 times higher than that of the preceding project, KEKB.

The SuperKEKB control system is based on EPICS [2]. The control system involves hundreds of computers which are networked together to allow communication between them. Therefore, monitoring the IT infrastructure is essential for a stable accelerator operation. We have deployed the monitoring system to monitor the IT infrastructure and EPICS control system. The system monitors two types of data: metrics and logs. This paper describes the system architectures and its applications that we have implemented.

METRIC MONITORING

Metrics are collection of a measurement at a certain point in time. They are typically collected at fixed-time intervals and referred to as a time series data. We monitor the metrics such as a network traffic and a CPU utilization with Zabbix [3].

Zabbix

Zabbix is an open-source monitoring software tool. It is integrated multiple features for monitoring as shown below:

- Data gathering.
- Alerting.
- Data visualization.
- Historical data storage.
- Network discovery.

We have monitored 22 computers for servers and operator consoles and 88 network switches. About 30000 items and 10000 triggers have been monitored. Data gathering is performed with SNMP or Zabbix agent.

Alerting settings

Zabbix supports triggers, which are logical expressions to evaluate gathered data and represents the current system state. Zabbix allows taking place some operations when trigger status changed. We have applied it to notify system problems by e-mail.

Trigger has severity parameter to defines how important a trigger is. Zabbix supports following trigger severities: Not classified, Information, Warning, Average, High and Disaster. We have configured notification behavior according to trigger severities. Alerts for High or Disaster severity are immediately notified. On the other hand, alerts for Warning or Average severity are notified when its trigger state remains PLOBLEM for 24 hours. There are no notifications for Information. This prevents major severity notifications from being buried in minor severity notifications.

Data Visualization on Grafana

Grafana [4] is an open-source software tool for data visualization and analysis. It allows to visualize data from various data storage backend. We can easily create and view dashboards via a web browser with Grafana.

Grafana is extendable with plugins to add a new panel or a datasource. We have applied Zabbix plugin [5] to Grafana to visualize the data stored in Zabbix data storage. This plugin provides metric processing functions to transform and shape the data. Figure 1 shows computer performance metrics on Grafana.

Figure 1: Grafana dashboard for computer performance metrics. The metrics are retrieved from Zabbix data storage.

† shinya.sasaki@kek.jp

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA134

Figure 2 shows Grafana panel for network traffic monitoring. This panel shows received network traffic on top 5 ports in the core switch using metric processing functions.

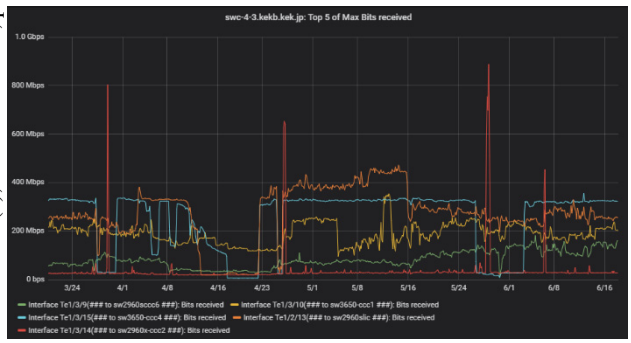


Figure 2: Grafana panel for incoming network traffic on core switch. Only top 5 ports in the switch are displayed using metric processing functions. Orange line indicates the traffic from Linac is reached to 500 Mbps. Red line indicates large data transfer occupies about 1 Gbps bandwidth.

LOG MONITORING

Log is text messages generated from an operating system or other software. Log data are typically generated when some event occurred and usually include timestamps which indicate when the events occurred. In most cases, parsing process is required to extract information from a log. We adopt Elastic Stack [6] for log data collection, searching, analysis and visualization.

Elastic Stack

Elastic Stack is a software stack for log monitoring also known as ELK Stack. It is comprised of Elasticsearch, Logstash, Kibana.

Elasticsearch Elasticsearch is a distributed, open-source search and analytics engine built on top of Lucene [7]. It stores complex data structures that have been serialized as JSON documents.

Logstash Logstash is used to aggregate and process data. It ingests data from multiple sources simultaneously and transform it before send it to Elasticsearch.

Kibana Kibana is a web-based analytics and visualization tool designed to work with Elasticsearch. It is available to search, view and interact with the data stored in Elasticsearch.

We have applied Elastic Stack to monitor broadcast packets and the EPICS control system.

Broadcast Packets Monitoring

We have monitored broadcast packets in the control network. Channel Access (CA) [8] uses broadcast for server beacons and PV name search requests. Analyzing broadcast packets allows to identify IOC that behaves anomaly or sends a lot of PV search requests.

The system architecture for the broadcast packets monitoring is shown in Fig. 3. We use TShark to dump broadcast packets. TShark is a command-line utility to capture and analyze network traffic provided by Wireshark [9]. TShark supports an JSON output format for Elasticsearch Bulk API. The output of TShark is ingested to Logstash and finally stored in Elasticsearch. We have applied cashark [10], which is a Wireshark dissector plugin for CA protocol and provides decoding of CA traffic.

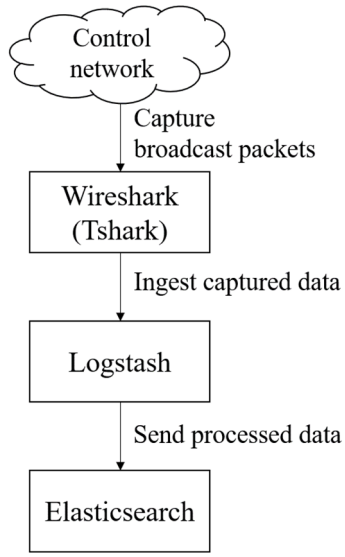


Figure 3: Broadcast packets monitoring pipeline.

Figure 4 shows a dashboard for the broadcast monitoring on Kibana. As the packets are decoded with cashark, CA command information such as command ID, version, searched channel name is allowed to be displayed.

Figure 4: Dashboard for broadcast monitoring on Kibana.

EPICS Control System Monitoring

We have monitored CA search frequencies and IOC beacons. CA search frequencies are monitored with caSnooper [11], which reports search frequencies for each searched PV to standard output. We have configured caSnooper to output the frequencies per 10 minutes and its outputs are parsed and sent to Elasticsearch with Logstash. Beacons are monitored with CASW (Channel Access Server Watcher) and ParseCASW [12], which are beacon diagnostic tools. Monitoring beacons brings a useful information to debug CA network problems.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA134

EPICS PV MONITORING WITH ZABBIX

We had several problems on EPICS IOC. For example, new CA client was rejected due to high CPU utilization or high memory consumption. To keep IOC healthy, monitoring IOC performance is required.

We have implemented EPICS PV monitoring system with Zabbix for IOC monitoring. PV values are monitored as metrics and they are managed as well as the metrics for IT infrastructure.

CA Client for Zabbix

We have developed zabbix-epics-py [13], CA client software to send PV value as a metric to Zabbix. It uses PyEpics [14] to collect PV values and py-zabbix [15] to send metrics to Zabbix server.

Figure 5 is an example usage of zabbix-epics-py. Zabbix-epics-py requires list of dictionaries including PV name, host name, item key, interval and func. Host name and item key should be registered in Zabbix. Item type sent PV values must be Zabbix trapper.

```
>>> from zbxepics import ZabbixSenderCA
>>> server_ip = '127.0.0.1'
>>> port = 10051
>>> config = False
>>> items = [dict(host='dummyHost', pv='TEST:PV', interval=30,
                  item_key='zabbix-epics-py-test.item', func='last')]
>>> sender = ZabbixSenderCA(server_ip, port, config, items)
>>> sender.run()
```

Figure 5: Example usage of zabbix-epics-py.

Interval is a time between metrics sending. PV values are stored in a buffer in the interval. Func determines what function is applied to the buffer before sending metric. For example, the latest value in the buffer is sent when last is specified to func while averaged value is sent when avg is specified. There are 4 functions for func: last, min, max and avg. All updated values are sent when monitor is set to interval, then specification of func is ignored.

Figure 6 shows behavior of zabbix-epics-py. The monitored PV value is updated to 3, 4 and 5. Therefore, the buffer stores three values. As max is specified to func, 5 which is the max value in the buffer is sent to Zabbix server.

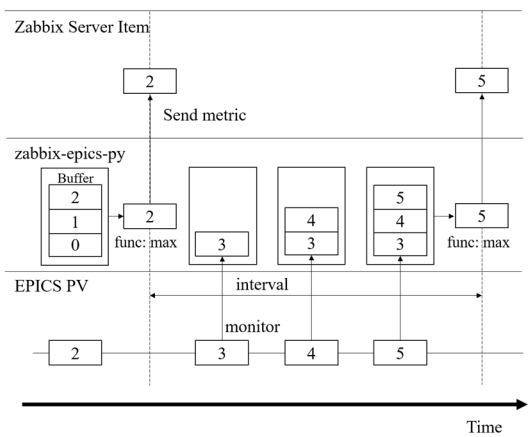


Figure 6: Behavior of zabbix-epics-py. Monitored PV updates 3 times and its values are stored in the zabbix-epics-py buffer. As func is set as max, 5 which is the max value in the buffer is sent to Zabbix server.

We have also developed zetemple [16], which is a wrapper software for zabbix-epics-py. Zetemple determines PV names from item keys in Zabbix template registered to host. Zetemple requires a csv file which specifies host name registered in Zabbix and prefix of PV name.

IOC Monitoring

To monitor IOC performance, devIocStats [17] is running on monitored IOC. DevIocStats provides PVs for CPU utilization, memory usage, number of CA clients and so on. Values of these PVs are sent to Zabbix. We have experimentally monitored 32 IOCs. Interval is set as 180 seconds and func is specified to last to send latest PV value.

DevIocStats has heartbeat PV which counts up every second. IOC might be having problem such as high loading when heartbeat PV doesn't update correctly. We have configured a Warning trigger which evaluates heartbeat has increased more than 178 in 180 seconds and also configured a Disaster trigger which evaluates heartbeat has updated at least once in 180 seconds.

VISUALIZE PVACCESS RPC DATA ON GRAFANA

While the data stored in Zabbix are visualized on Grafana, accelerator data not supported by Grafana are visualized on dedicated web application or standalone GUI application at SuperKEKB. Visualizing these data such as archived PV data or alarm history on Grafana is expected to provide effective monitoring.

We have developed Grafana datasource plugin and HTTP / pvAccess API gateway to visualize arbitrary data. These applications allow to visualize the data from pvAccess RPC servers on Grafana.

PvAccess is the next generation communication protocol on EPICS 7 and supports the integration of all data into microservices [18]. We adopt pvAccess RPC to retrieve the data.

System Architecture

The system architecture of the data visualization is shown in Fig. 7. The system consists of pvAccess datasource plugin, HTTP / API gateway and pvAccess RPC servers.

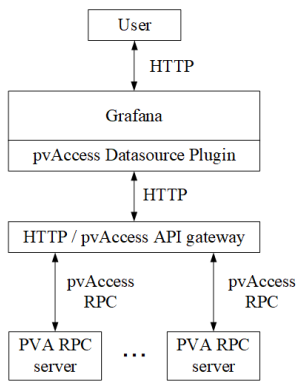


Figure 7: System architecture of the data visualization system for pvAccess RPC data.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA134

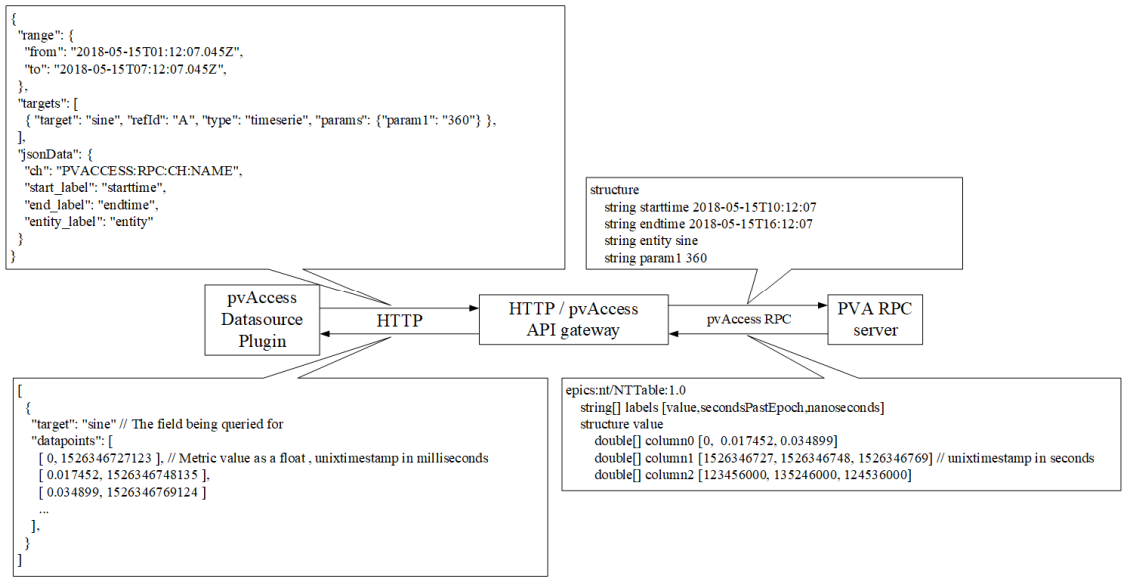


Figure 8: Message flow in the data visualization system. Retrieved data are time series.

Figure 8 shows messages communicated between each component. Procedure of visualizing data is below:

1. PvAccess datasource plugin on Grafana send a data retrieval request to HTTP / API gateway.
2. The API gateway converts a HTTP request into a pvAccess RPC request.
3. PvAccess RPC server send back a response according to the received request.
4. The API gateway converts the response to a HTTP response and send it to the datasource plugin.
5. Grafana visualizes the response data.

Datasource can be added by adding a pvAccess RPC server which has the predetermined interface.

pvAccess datasource plugin We have developed General pvAccess Datasource [19], which is a Grafana plugin to visualize the data retrieved from pvAccess RPC server. The features of this plugin are as shown below:

- pvAccess RPC channel names for query, annotation and search are passed from datasource configuration panel on Grafana.
- Configurable query parameters for each RPC server.
- Show time series data and table data.

The plugin communicates with HTTP / pvAccess API gateway via HTTP. The arguments sent to the gateway are time range, parameters for each target data, channel name and argument labels for RPC.

HTTP / pvAccess API gateway The gateway provides an API conversion from HTTP to pvAccess RPC. We have developed gfhttpva [20] as the API gateway supposed to use with General pvAccess Datasource. Gfhttpva is a HTTP server using Flask [21] as a web framework. It also uses pvaPy [22] to communicate with pvAccess RPC servers.

pvAccess RPC servers PvAccess RPC servers provide the data to be visualized. Arguments and return values for RPC are determined by gfhttpva. RPC servers are allowed to use arbitrary arguments while arguments as shown below are essential:

- starttime: start time of time range.
- endtime: end time of time range.
- entity: string which specify what data to be retrieved.

Labels for essential arguments are arbitrary. The type of returned value must be NTTable [23] whether time series data or table data. Returned NTTable value for time series data must have 3 columns: value, secondsPastEpoch and nanoseconds. Table data is allowed to include arbitrary columns.

Developed RPC Servers

Example datasource We have developed Example datasource [24] to return time series data. This RPC server returns data according to specified starttime, endtime and entity. The RPC server also supports param1 argument to specify data parameter.

For example, the server returns sine wave data from 0 degree to 360 degree when the entity is specified as sine and param1 is specified as 360. Figure 9 shows example data visualized on Grafana.

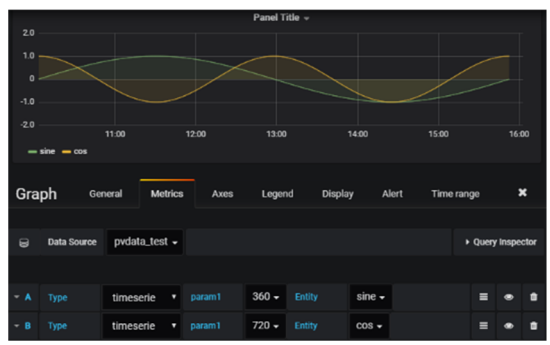
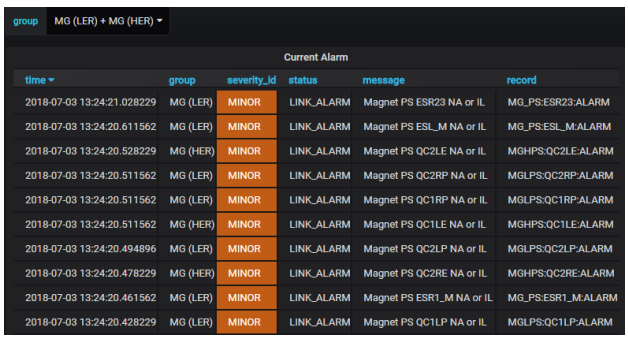


Figure 9: Visualized pvAccess RPC sample data. Entities are set to sine and cos.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA134

CSS alarm datasource CSS alarm management system have been operated at SuperKEKB. We have developed an RPC server [25] to retrieve current status and history of CSS alarm information stored in PostgreSQL database. This RPC server supports only table data.

Current status retrieval uses only entity argument. The entity argument filters alarms by registered group. History data is retrieved in given time range. Entity argument is also used to filter by registered group and message argument is optionally used to filter by alarm message. Figure 10 shows current alarm status on Grafana.


group MG (LER) + MG (HER)

Current Alarm

time	group	severity_id	status	message	record
2018-07-03 13:24:21.028229	MG (LER)	MINOR	LINK_ALARM	Magnet PS ESR23 NA or IL	MG_PS:ESR23:ALARM
2018-07-03 13:24:20.611562	MG (LER)	MINOR	LINK_ALARM	Magnet PS ESL_M NA or IL	MG_PS:ESL_M:ALARM
2018-07-03 13:24:20.528229	MG (HER)	MINOR	LINK_ALARM	Magnet PS QC2LE NA or IL	MGHPS:QC2LE:ALARM
2018-07-03 13:24:20.511562	MG (LER)	MINOR	LINK_ALARM	Magnet PS QC2RP NA or IL	MGLPS:QC2RP:ALARM
2018-07-03 13:24:20.511562	MG (LER)	MINOR	LINK_ALARM	Magnet PS QC1RP NA or IL	MGLPS:QC1RP:ALARM
2018-07-03 13:24:20.511562	MG (HER)	MINOR	LINK_ALARM	Magnet PS QC1LE NA or IL	MGHPS:QC1LE:ALARM
2018-07-03 13:24:20.494896	MG (LER)	MINOR	LINK_ALARM	Magnet PS QC2LP NA or IL	MGLPS:QC2LP:ALARM
2018-07-03 13:24:20.478229	MG (HER)	MINOR	LINK_ALARM	Magnet PS QC2RE NA or IL	MGHPS:QC2RE:ALARM
2018-07-03 13:24:20.461562	MG (LER)	MINOR	LINK_ALARM	Magnet PS ESR1_M NA or IL	MG_PS:ESR1_M:ALARM
2018-07-03 13:24:20.428229	MG (LER)	MINOR	LINK_ALARM	Magnet PS QC1LP NA or IL	MGLPS:QC1LP:ALARM

Figure 10: Table of current CSS alarm data. Displayed groups are MG (LER) and MG (HER).

CONCLUSION

The monitoring system for metrics and logs have been deployed. Visualized data helps us to understand the current and past status of the control system. Notifications of a system problem from Zabbix allows early response to its problem.

In order to expand monitoring system, we have developed EPICS PV monitoring system with Zabbix and data visualization system with pvAccess RPC and Grafana. These systems provide consolidated monitoring for various types of data. We have applied it for monitoring IOC performance and status monitoring such as CSS alarm status.

REFERENCES

[1] Y. Ohnishi *et al.*, "Accelerator design at SuperKEKB", *Prog. Theor. Exp. Phys.,* vol. 2013, no. 3, p. 03A011, Mar. 2013. doi:10.1093/ptep/pts083

[2] EPICS, https://epics-controls.org

[3] Zabbix, https://www.zabbix.com

[4] Grafana, https://grafana.com

[5] Zabbix plugin for Grafana, https://grafana.com/grafana/plugins/alexanderzobnin-zabbix-app

[6] Elastic Stack, https://www.elastic.co/products/elastic-stack

[7] Luence, https://lucene.apache.org

[8] Channel Access Protocol Specification, https://epics.anl.gov/base/R3-16/1-docs/CAproto/index.html

[9] Wireshark, https://www.wireshark.org

[10] cashark, https://github.com/mdavidsaver/cashark

[11] caSnooper, https://epics.anl.gov/extensions/caSnooper/index.php

[12] ParseCASW, https://epics.anl.gov/extensions/ParseCASW/index.php

[13] zabbix-epics-py, https://github.com/sasaki77/zabbix-epics-py

[14] PyEpics, http://pyepics.github.io/pyepics

[15] py-zabbix, https://github.com/adubkov/py-zabbix

[16] zetemple, https://github.com/sasaki77/zetemple

[17] devIocStats, https://www.slac.stanford.edu/comp/unix/package/epics/site/devIocStats

[18] L. R. Dalesio *et al.*, "EPICS 7 Provides Major Enhancements to the EPICS Toolkit", in *Proc. ICALEPCS'17,* Barcelona, Spain, Oct. 2017, pp. 22-26. doi:10.18429/JACoW-ICALEPCS2017-MOBPL01

[19] General pvAccess Datasource, https://github.com/sasaki77/generalpvaccess-datasource

[20] gfhttpva, https://github.com/sasaki77/gfhttpva

[21] Flask, https://palletsprojects.com/p/flask

[22] S. Veseli, "PvaPy: Python API for EPICS PV Access", in *Proc. ICALEPCS'15,* Melbourne, Australia, Oct. 2015, pp. 970-973. doi:10.18429/JACoW-ICALEPCS2015-WEPGF116

[23] EPICS V4 Normative Types, http://epics-pvdata.sourceforge.net/alpha/normativeTypes/normativeTypes.html

[24] pvAccess RPC sample for gfhttpva, https://github.com/sasaki77/grafana-pvarpc-sample

[25] CSS alarm datasource, https://github.com/sasaki77/cssalmrpc