Internship Report: Preperation for a Monitoring System

Daniel Kutny

October 2017

Abstract

As the BrainScaleS System is moving towards becoming a platform for external users, a monitoring system which allows a resource view and a detection of error is needed for a smooth workflow. In this internship, a visualization and an event data storage were prepared for a central monitoring system, which will allow further development towards a central monitoring system in the bachelor thesis.

Contents

1 Introduction

1.1 Goals

The central goals of a successful monitoring platform are

1. **Resource View** Can wee see any patterns in the usage our system? Can we recognize any bottlenecks?

2. **Error Detection** Can we see values outside of specific ranges or error messages? Are there any correlations in the usage and the error?

3. **Optimization** Can we learn from those errors and patterns? Can we optimize using this knowledge?

1.2 Data Types

There are two types of data which we can monitor. There is the so called time-series data and event data.

Time-Series Data This is a time-dependent, continuous stream of numerical values (Integers, floats,..). The understanding of a physicist would be a time-dependent function $f(t)$. Examples of such values are CPU Temperature, work load, and network traffic.

Event Data Events are notifications about an occurrence at a certain point in time, these are not continuous and do not need to be associated with a numerical value, but contain a message or structured data. Examples of such event data are error messages, log messages, access requests, violations or abortions of a program.

1.3 Requirements

What does a monitoring system need? A monitoring system consists of three parts. At first, data has to be aggregated (*Aggregation*), then it needs to be stored (*Aggregation*) and visualized (*Visualization*).
For the aggregation of time-series data of the computers and servers, for example CPU temperature, ping or memory, the monitoring software Ganglia is used. The wafer data is aggregated with a Raspberry Pi which is connected to the wafer and which sends the data to the storage.
For the storage of time-series data we use Graphite, which is specialized on this type of data. For events we use Elasticsearch, which was deployed as part of the internship.[3] The visualization is made with Grafana, which allows to visualize data from different sources and different types. [2]

Aggregation	Storage	Visualization
Time Series	**Events**	Grafana
Ganglia	Elasticsearch	Kibana
Raspberry Pi	**Time Series**	
Events	Carbon / Graphite	
Filebeat		

Figure 1: Summary of the different parts of a monitoring system

2 The Internship

As part of the internship, a Debian-based server was set up which hosts the visualization and the event data storage. The server is called **monviz** and runs under the address `monviz.kip.uni-heidelberg.de`. In general, the server can be only accessed from inside the KIP network.

2.1 Visualization

The visualization software Grafana has been installed on monviz. The access to Grafana has been exposed to the general with the BrainScaleS reverse proxy. It can now be accessed via `https://brainscales-r.kip.uni-heidelberg.de:12443/grafana`. The access has been restricted to BrainScaleS users with the BrainScaleS LDAP.
The first tries of dashboards have been made and can be viewed in the screenshots. The full and correct creation of the dashboards, however, will be in the bachelor thesis.
Dashboards are divided into rows which can be created, rearranged and deleted. The user can add so called panels, which allow to create different visualizations of dataset.
Examples of such panels are the graph, which visualizes time series data, a single status panel, which can show the average, moving averages or the current value of a metric. The background of the panel can be configured in such a way that its color changes based on trigger values.
A special panel is the text panel. It allows to include raw text, markdown and html. Because of this, we can include any external content using an *iframe* tag. The problem with this method is that the browser blocks resources coming from any other domain or port.
Therefore, one needs to reverse proxy Grafana and the resource one wants to include to the same domain and the same port.
Data sources can be

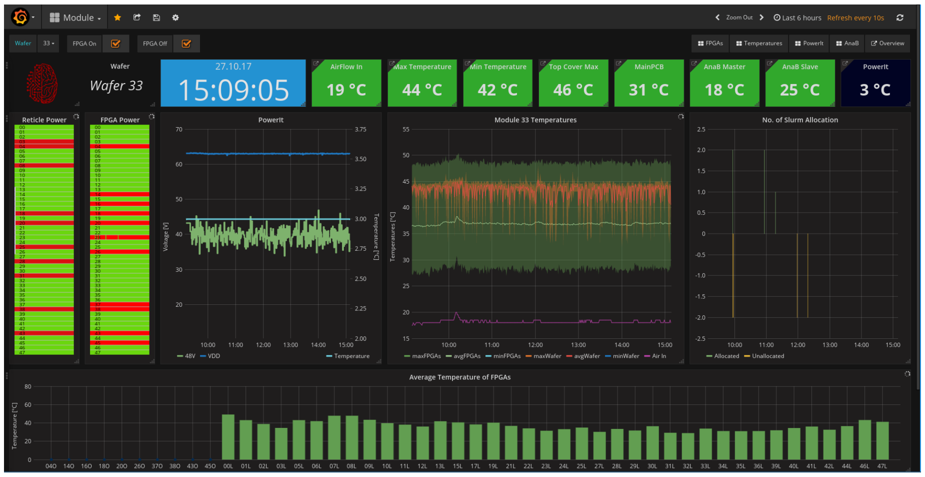


Figure 2: Dashboard with an Overview of a Wafer Module

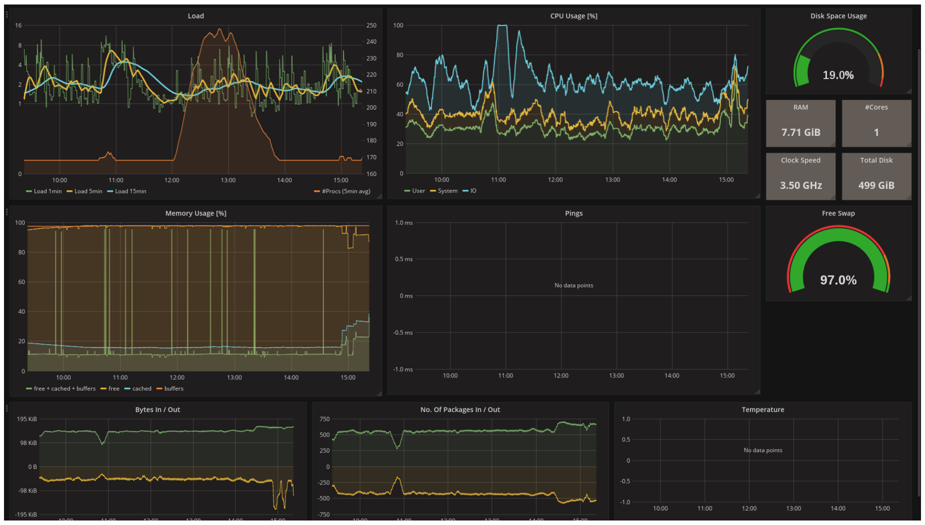
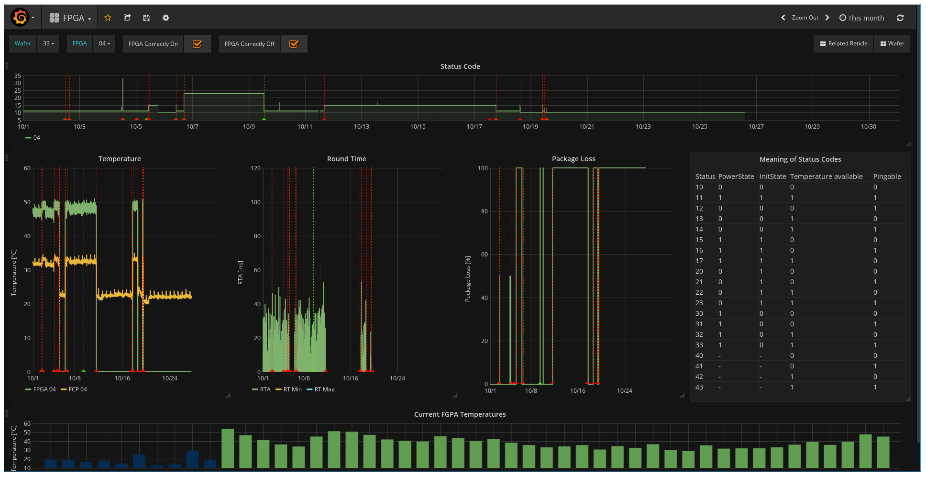


Figure 3: View of the Client Information

2.2 Event data storage

For the event data, we decided to use Elasticsearch as a storage. The Elasticsearch server is now running on **monviz.kip.uni-heidelberg.de:9200**. Elasticsearch requires Java 8, therefore it had to be installed and the environment variables $JAVA_HOME and $JAVA_HOME_BIN had to be adapted.

Elasticsearch does not support security natively, therefore the Elasticsearch server can only be accessed internally until a security solution like SafeGuard



Meaning of Status Codes

Status	PowerState	InitState	Temperature available	Pingable
10	0	0	0	0
11	1	1	1	1
12	0	0	0	1
13	0	0	1	0
14	0	0	1	1
15	1	1	0	0
16	1	1	0	1
17	1	1	1	0
20	0	1	0	0
21	0	1	0	1
22	0	1	1	0
23	0	1	1	1
30	1	0	0	0
31	1	0	0	1
32	1	0	1	0
33	1	0	1	1
40	-	-	0	0
41	-	-	0	1
42	-	-	1	0
43	-	-	1	1

Figure 4: View of a FPGA

or X-Pack is implemented.
In Elasticsearch, we can save any data which can be represented as a JSON file. As an example, we might want to create a movie database, we would save the movies as follows:

```
{
    "title": "Pulp Fiction",
    "director": "Quentin Tarantino",
    "year": 1994,
    "actors": [
        "John Travolta",
        "Samuel L. Jackson",
        ...
    ]
}
```

Such a JSON file is called a document. Every such document has a type and is saved in an index. A meaningful type for this document might be ”movie”, which would be saved in an index called ”media”. An example for a use-case for us would be the time at which a reticle was allocated.

```
{
    "Wafer":"33",
    "Reticle":"12",
    "Allocated":1
}
```

Datasets in Elasticsearch can be queried using a Lucene Query. The base url for a query is `monviz.kip.uni-heidelberg.de:9200/_search?q=query`, where we can replace query with the our expression. A Lucene Query consists of a key and a value which need to match to our search result. For example, Wafer:33 would filter out all documents containing a key Wafer and the value 33. There are also other special queries which can be important, for example

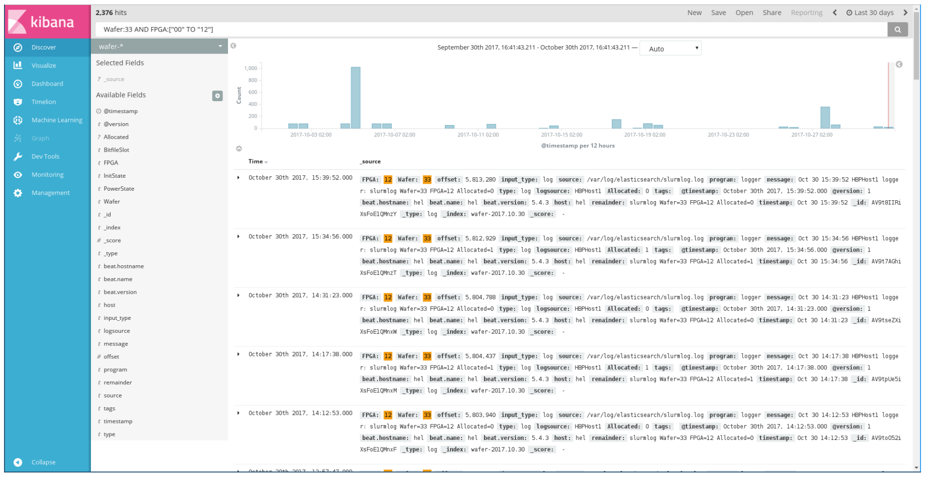
1. **AND** Require two key-values, i.e. Wafer:33 AND Reticle:12 would show all documents of the Reticle 12 from Wafer 33.

2. **OR** Requires one of two key-values, i.e. Wafer:33 OR Wafer:21 returns all documents from the Wafer 33 and Wafer 21.

3. [**x TO y**] Requires a range of values, i.e. Status:[0 TO 5] would return all documents with a status code between 0 and 5.

4. **_exists_** Checks if the key exists in the document, i.e _exists_:Wafer would show all documents where the key "Wafer" is present.

The response itself is again a JSON object containing all documents which match the query. There is also another method for querying called Request Body Search, which is very powerful, but goes beyond the scope of this internship. [1] To be able to better visualize and debug Elasticsearch queries, Kibana was installed. Kibana is a tool focused on visualization of Elasticsearch Data. Like Grafana, it is web-based and can be accessed using `https://brainscales-r.kip.uni-heidelberg.de:12443/kibana`. Below is a screenshot of the software with an example query.



```
Wafer:33 AND FPGA:["00" TO "12"]
```

```
October 30th 2017, 15:39:52.000  FPGA: 12 Wafer: 33 offset: 5,813,280 input_type: log source: /var/log/elasticsearch/slurmlog.log program: logger message: Oct 30 15:39:52 HBPHost1 logge
                                 r: slurmlog Wafer=33 FPGA=12 Allocated=0 type: log logsource: HBPHost1 Allocated: 0 tags:  @timestamp: October 30th 2017, 15:39:52.000 @version: 1
                                 beat.hostname: hel beat.name: hel beat.version: 5.4.3 host: hel remainder: slurmlog Wafer=33 FPGA=12 Allocated=0 timestamp: Oct 30 15:39:52 _id: AV9t8IIRi
                                 XsFoE1QMnzY _type: log _index: wafer-2017.10.30 _score:  -

October 30th 2017, 15:34:56.000  FPGA: 12 Wafer: 33 offset: 5,812,929 input_type: log source: /var/log/elasticsearch/slurmlog.log program: logger message: Oct 30 15:34:56 HBPHost1 logge
                                 r: slurmlog Wafer=33 FPGA=12 Allocated=1 type: log logsource: HBPHost1 Allocated: 1 tags:  @timestamp: October 30th 2017, 15:34:56.000 @version: 1
                                 beat.hostname: hel beat.name: hel beat.version: 5.4.3 host: hel remainder: slurmlog Wafer=33 FPGA=12 Allocated=1 timestamp: Oct 30 15:34:56 _id: AV9t7AGhi
                                 XsFoE1QMnzT _type: log _index: wafer-2017.10.30 _score:  -

October 30th 2017, 14:31:23.000  FPGA: 12 Wafer: 33 offset: 5,804,788 input_type: log source: /var/log/elasticsearch/slurmlog.log program: logger message: Oct 30 14:31:23 HBPHost1 logge
                                 r: slurmlog Wafer=33 FPGA=12 Allocated=0 type: log logsource: HBPHost1 Allocated: 0 tags:  @timestamp: October 30th 2017, 14:31:23.000 @version: 1
                                 beat.hostname: hel beat.name: hel beat.version: 5.4.3 host: hel remainder: slurmlog Wafer=33 FPGA=12 Allocated=0 timestamp: Oct 30 14:31:23 _id: AV9tseZXi
                                 XsFoE1QMnxW _type: log _index: wafer-2017.10.30 _score:  -

October 30th 2017, 14:17:38.000  FPGA: 12 Wafer: 33 offset: 5,804,437 input_type: log source: /var/log/elasticsearch/slurmlog.log program: logger message: Oct 30 14:17:38 HBPHost1 logge
                                 r: slurmlog Wafer=33 FPGA=12 Allocated=1 type: log logsource: HBPHost1 Allocated: 1 tags:  @timestamp: October 30th 2017, 14:17:38.000 @version: 1
                                 beat.hostname: hel beat.name: hel beat.version: 5.4.3 host: hel remainder: slurmlog Wafer=33 FPGA=12 Allocated=1 timestamp: Oct 30 14:17:38 _id: AV9tpUe5i
                                 XsFoE1QMnxM _type: log _index: wafer-2017.10.30 _score:  -

October 30th 2017, 14:12:53.000  FPGA: 12 Wafer: 33 offset: 5,803,940 input_type: log source: /var/log/elasticsearch/slurmlog.log program: logger message: Oct 30 14:12:53 HBPHost1 logge
                                 r: slurmlog Wafer=33 FPGA=12 Allocated=0 type: log logsource: HBPHost1 Allocated: 0 tags:  @timestamp: October 30th 2017, 14:12:53.000 @version: 1
                                 beat.hostname: hel beat.name: hel beat.version: 5.4.3 host: hel remainder: slurmlog Wafer=33 FPGA=12 Allocated=0 timestamp: Oct 30 14:12:53 _id: AV9to052i
                                 XsFoE1QMnxF _type: log _index: wafer-2017.10.30 _score:  -
```

Figure 5: Screenshot of Kibana

3 Outlook

3.1 Bachelor Thesis

Now that the basis for a modern monitoring system is ready, the dashboards for the most important components of the BrainScaleS System can be created. These include Wafer Temperatures, AnaB voltages, Reticle voltages, PowerIt currents and voltages, etc.
It will also be important to visualize information about the servers and clients that are in use at the Electronic Visions cluster.
Another goal of the Bachelor thesis is to make use of Elasticsearch so that we can query and plot unstructured data.

3.2 Long-Term

The long-term goal for Electronic Visions' monitoring system is to centralize and save all important logs and errors, so that smooth workflow for external scientists can be assured.
An important part of monitoring is alerting. If some temperature stays above a certain threshold, then administrator should be informed. For meaningful alerting, a good system control and calibration of sensors is needed.
Another important long-term goal is the visualization of the experiment status and its usage, for example, how many neurons, synapses or HICANNs are in use?

References

[1] Elastic Documentation. `https://www.elastic.co/guide/index.html`.

[2] Grafana Documentation. `http://docs.grafana.org/`.

[3] Graphite Documentation. `https://graphite.readthedocs.io/en/latest/`.