Analyzing data flows of WLCG jobs at batch job level

Eileen Kuehn, Max Fischer, Manuel Giffels, Christopher Jung and Andreas Petzold

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

E-mail: eileen.kuehn@kit.edu, max.fischer@kit.edu, manuel.giffels@kit.edu, christopher.jung@kit.edu, andreas.petzold@kit.edu

Abstract. With the introduction of federated data access to the workflows of WLCG, it is becoming increasingly important for data centers to understand specific data flows regarding storage element accesses, firewall configurations, as well as the scheduling of batch jobs themselves. As existing batch system monitoring and related system monitoring tools do not support measurements at batch job level, a new tool has been developed and put into operation at the GridKa Tier 1 center for monitoring continuous data streams and characteristics of WLCG jobs and pilots. Long term measurements and data collection are in progress. These measurements already have been proven to be useful analyzing misbehaviors and various issues. Therefore we aim for an automated, realtime approach for anomaly detection. As a requirement, prototypes for standard workflows have to be examined. Based on measurements of several months, different features of HEP jobs are evaluated regarding their effectiveness for data mining approaches to identify these common workflows. The paper will introduce the actual measurement approach and statistics as well as the general concept and first results classifying different HEP job workflows derived from the measurements at GridKa.

1. Introduction

The increase in opportunistic resource usage [1] and federated storage [2, 3, 4, 5] in different computing models of WLCG communities requires network-centric monitoring techniques. Different approaches dealing with this issue implemented user- and job-specific, or VO-specific monitoring [6, 7, 8]. Others deploy agents to data servers of federations itself [9]. Additionally there are mediating [10, 11] and aggregating [12] approaches for the different techniques. But none of the aforementioned approaches tracks the logical connections between a single job and its data transfers. This might be derived by accumulating data flow centric approaches measuring traffic for data federations with job-specific information. However, this is prone to errors but more importantly incomplete. The measurement of data flows requires knowledge about data servers, so opportunistic resources and data outside of data federations is not included in such measurements. Especially in data centers operating a batch system with commodity hardware, it is essential to know about the logical connections. As the network is shared by thousands of batch jobs and users, a differentiation and metric for job-specific network usage is desirable.

For this goal to be achieved a monitoring tool has been implemented at the GridKa data and computing center profiling data flows of WLCG batch jobs (section 2). In addition, first toy analysis show the relevance of measured data (section 3) to gain experience in identifying access patterns and estimating data transfers to finally understand data flows.

2. Monitoring the traffic of batch jobs

Existing monitoring tools in use at GridKa Tier 1 – including Cacti® [13], Icinga [14], Ganglia [15], and Univa Grid Engine [16] – deliver fine grained information about network traffic regarding different ports, the accumulated traffic by rack, or node and different services and alarms based on the data. But they are not capable of measuring network traffic on batch job level. As existing batch system monitoring and related system monitoring tools do not support measurements at batch job level, a new tool has been developed and put into operation at the GridKa Tier 1 center for monitoring continuous data streams and characteristics of WLCG jobs and pilots.

The design and implementation of the tool is based on C/C++. It has been modeled based on the OpenSource tool NetHogs [17]. NetHogs groups bandwidth by process instead of breaking traffic down per protocol or subnet. It does not rely on a special kernel module to be loaded but the Linux operating system and libpcap [18]. Libpcap is a system-independent interface for packet capturing in the user space. Network packets are copied into a buffer for further analysis. By utilizing libpcap, UDP and TCP network packets can be analyzed and due to procfs connections can be tracked. As NetHogs itself has not been implemented with monitoring in mind it can not be used directly for our purpose.

Important data to be analyzed is the destination IP and port as well as the source IP and port. Those are needed for matchmaking each connection with the sockets located in procfs. After the socket has been found the managing process needs to be identified. Therefore again the procfs is being utilized. Each process itself is tracked while monitoring the network packets for a fast identification and association to the batch job itself. Processes executed by root can also be tracked but do not necessarily belong to the traffic of jobs. Therefore different configurations for running the tool can be utilized.

2.1. Hardware and setup

The tool is currently running on two racks consisting of 32 worker nodes. Each worker node has 24 job slots that need to be monitored in parallel. Long term measurements and data collection are in progress. These measurements already have been proven to be useful analyzing misbehaviors and various issues. Therefore we aim for an automated, realtime approach for anomaly detection.

2.2. Data format

All data are time series regarding the actual network traffic (traffic rates, count of inbound and outbound packets, destination and source IP as well as ports), relevant unix process information (`pid`, `ppid`, `uid`) as well as information about the batch job itself (see figure 1). Every 20 seconds a new record for accumulated traffic information is created. Relevant process information are stored directly after the associated system event. For an improved data handling and analysis there is additional metadata associated to the measurements. These are stored in an additional database enabling a fast access to specific data.

3. Analyzing data flows of batch jobs

The actual measurements contain very detailed data as well as implicit tree structures of the executed unix processes per job. Each job is defined by a single multidimensional time series being correlated to the jobs on the same worker node and also to all other worker nodes inside the GridKa. Thus, the creation and collection of monitoring data is one important component of the monitoring chain. The other one is the data mining process to provide an insight into usage patterns and create statistics.

Before actually performing a clustering to identify patterns inside the dataset, there is the need to examine the general value and significance of the monitored data. For this purpose,

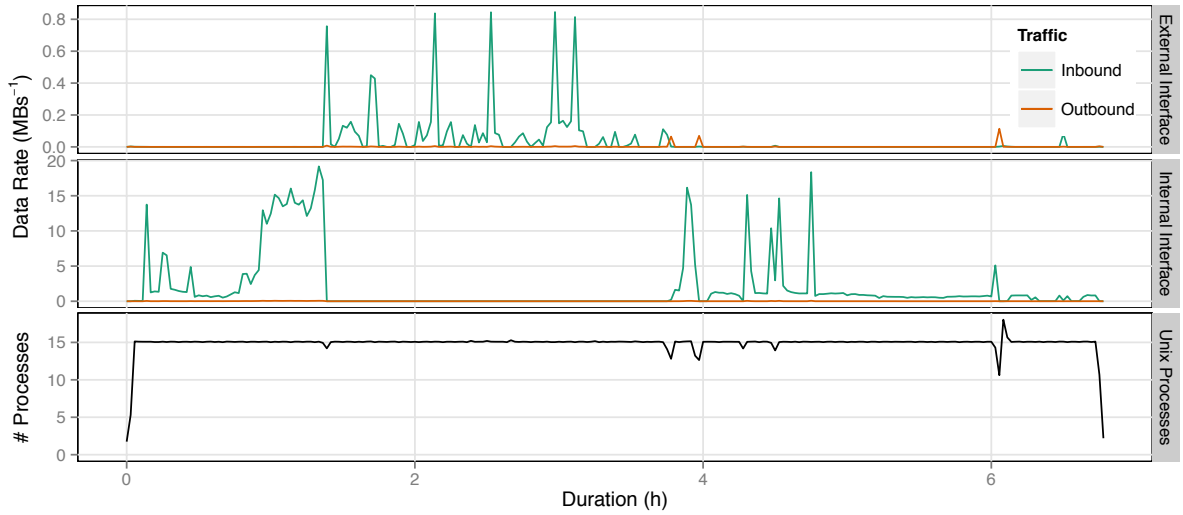

Figure 1. For each job detailed information about inbound and outbound traffic on internal and external interfaces, as well as running processes and metadata are available. The first plot visualizes incoming and outgoing traffic for the external interface. The second plot visualizes the same information but for the internal interface. The third plot displays the current number of running processes with regard to their proportion of the current timeframe.

a toy analysis to classify the executing VO of a batch job was designed. By performing the analysis it should be proven that the corresponding VO can be predicted by analyzing only the measured traffic profiles with an efficiency and purity greater than 90%. This use case can clearly be validated by verifying the monitored `uid`.

3.1. Preprocessing

To ensure the best quality of the model a filtering of batch jobs is performed in preprocessing. Imbalanced data sets are a special case for classification problems. The class distribution is not uniform and therefore complicates the data mining process, since standard classification algorithms usually consider a balanced training set. To avoid biased results towards the majority class only the four main LHC experiments ALICE, ATLAS, CMS, and LHCb were chosen to be included in the final dataset. Additionally the matching jobs were evaluated for their completeness: Unfinished jobs, misbehaving jobs as well as jobs with missing data should be excluded. Missing data is recognized by reconstructing the unix process tree from the logs and assigning the appropriate traffic. As soon as a parent process can not be found and the current process is not the batch job shepherd itself, some data must be missing from the job. There are also jobs whose processes run longer inside the system than the batch jobs itself. As this behavior is not evaluated yet, those jobs also need to be excluded from the data set. Furthermore short jobs like Service Availability Monitoring (SAM) [19] tests or pilots running into a timeout are excluded by explicitly selecting jobs with a duration longer than 2.100 seconds.

From the whole variable space the following set of variables is included into the modeling: connection category (internal or external interface), incoming and outgoing volume (kB), incoming and outgoing rate (kB/s), duration and the job id itself.

The different variables are accumulated for every job so that each observation of a job consists of a six-dimensional feature set. In figure 2 pair-wise scatterplots and densities are visualized for selected features of the internal network interface.

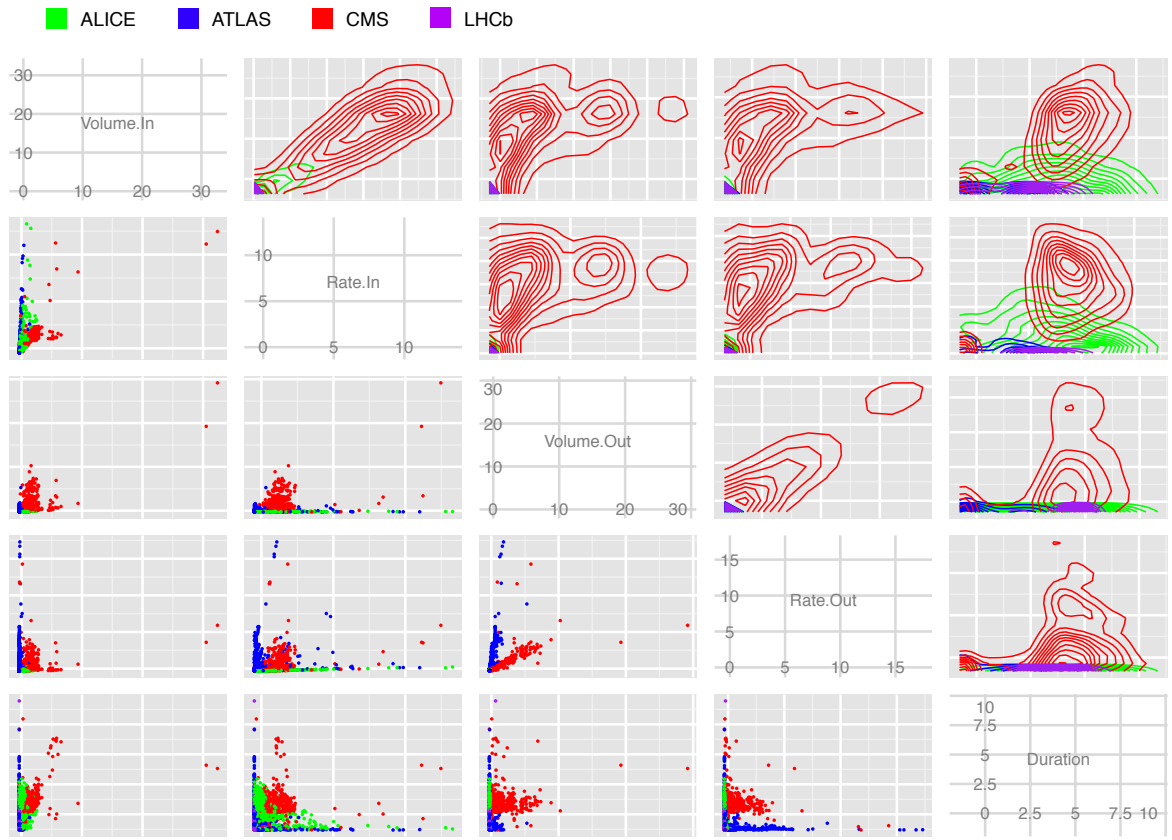

Figure 2. Scatter matrix visualizing selected scaled features for the internal network interfaces. Below the diagonal pair-wise scatter plots of variables are presented. Above the diagonal the density is shown. The colors represent the single experiments.

During data preprocessing care is taken that the data is normalized since the range of values of raw data varies widely. The majority of classifiers calculate distance between points by distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. By applying normalization each feature contributes approximately proportional to the final distance. To this end, the standard value is calculated by applying (1) to each variable j. Each value of variable j is centred around 0 by subtracting the mean μ_j and is additionally weighted by the standard deviation σ_j.

$$z_j = \frac{x_j - \mu_j}{\sigma_j} \tag{1}$$

3.2. Training a decision tree classifier

For the actual test analysis a decision tree classifier is used. The classifier is capable of performing multi-class classification by learning simple decision rules inferred from the data features. It was chosen because decision trees are simple to understand, interprete, and by using statistical tests the reliability of the resulting model can be validated. The analysis with the decision tree classifier is done with the software RapidMiner [20]. RapidMiner is an integrated environment for machine learning, data mining, predictive analytics, and business analytics. For measuring the quality and especially the stability of the selected data mining model the dataset is separated into three different and independent subsets: training data (60%), test data (20%), and validation

Table 1. Classification Performance Confusion Matrix visualizing the classification performance for the validation set by using the learned decision tree classifier based on traffic patterns of ALICE, ATLAS, CMS, and LHCb.

	ALICE	**ATLAS**	**CMS**	**LHCb**	**Efficiency (%)**
Pred. ALICE	206	2	6	8	92.8
Pred. ATLAS	2	330	3	0	98.5
Pred. CMS	0	0	121	0	100.0
Pred. LHCb	1	1	2	154	97.5
Purity (%)	98.6	99.1	91.7	95.1	

data (20%).

The model is built on the training set, refined with the test set and its quality is finally estimated on the validation set. The data has been taken over a timespan of 15 days on six different worker nodes. Three of them are located in an own rack respectively. The data set is split by utilizing stratified sampling to obtain the original distribution of VOs in the data. Stratified sampling is a sampling method that generates samples whilst maintaining the original proportion of each class. This way the resulting data sets still represent the original distribution of VOs. It is ensured that the validation set is entirely separate and distinct from the training and test set to get an independent validation of the resulting model.

3.3. Evaluation

Table 1 displays the confusion matrix of classification performance on the validation set for the resulting decision tree. Although the results outperform our requirements of 90% efficiency and purity the effect of imbalanced data sets is still visible. The best results are gained for ATLAS as most data was available. For ALICE important jobs are not considered in the analysis as parts of analysis trains do not pass the preselection stop regarding the job duration. The specific influence of pilots also has to be measured. It might influence the results as well. E. g. some of the jobs of CMS are no regular jobs but pilots.

Summarizing, this first test analysis shows that traffic data of batch jobs seem to be a good indicator for further analysis. But to improve the performance and ensure comparability of different data sets substructures need to be recognized.

4. Discussion and final remarks

The actual measurements and analysis undertaken at the GridKa Tier 1 center show that network traffic profiles of WLCG batch jobs are a good indicator to predict the VO ownership and therefore there is some good indication that network patterns themselves might be a good indicator for clustering. The data being taken is dependent on the actual community needs and evolution of computing models in use. Changes might therefore invalidate previously discovered patterns. As a result, there is a need for incremental methods that may update existing models and still evolve and update with temporal changes in data.

Nonetheless, pilots are currently not taken into account correctly. They are treated as single jobs but instead they are local batch systems themselves [21]. By introducing some further analysis to detect actual jobs inside the pilots might even further improve the classifier.

Acknowledgments
The authors wish to thank all people and institutions involved in the project Large Scale Data Management and Analysis (LSDMA), as well as the German Helmholtz Association and the Karlsruhe School of Elementary Particle and Astroparticle Physics (KSETA) for supporting and funding the work.

References

[1] Kreuzer P *et al.* 2014 *Journal of Physics: Conference Series* **513** 062028
[2] Bauerdick L *et al.* 2012 *Journal of Physics: Conference Series* **396** 042009
[3] Gardner R *et al.* 2014 *Journal of Physics: Conference Series* **513** 042049
[4] Bonacorsi D 2012 CMS storage federations *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE* (IEEE) pp 2012–2015
[5] Bloom K and The CMS Collaboration 2014 *Journal of Physics: Conference Series* **513** 042005
[6] Lorenz D *et al.* 2009 *Future Generation Computer Systems* **25** 308–314
[7] Arkhipkin D, Lauret J and Zulkarneeva Y 2014 *Journal of Physics: Conference Series* **513** 032002
[8] Legrand I *et al.* 2009 *Computer Physics Communications* **180** 2472–2498
[9] Andreeva J *et al.* 2014 *Journal of Physics: Conference Series* **513** 032004
[10] Cooke A W *et al.* 2004 *Journal of Grid Computing* **2** 323–339
[11] Aiftimiei C *et al.* 2008 *Journal of Physics: Conference Series* **119** 062003
[12] Andreeva J J *et al.* 2007 Experiment Dashboard: the monitoring system for the LHC experiments *GMW '07: Proceedings of the 2007 workshop on Grid monitoring* (New York, New York, USA: ACM) pp 45–49
[13] Cacti Group 2013 Cacti®- the complete rrdtool-based graphing solution. URL `http://www.cacti.net`
[14] Icinga Project 2014 Icinga — open source monitoring. URL `https://www.icinga.org`
[15] Massie M L, Chun B N and Culler D E 2004 *Parallel Computing* **30** 817–840
[16] Univa Corporation 2014 Univa products: Grid engine software for workload scheduling and management. URL `http://www.univa.com/products/grid-engine.php`
[17] Engelen A 2013 Nethogs: What program is using that bandwidth? URL `http://nethogs.sourceforge.net`
[18] 2014 Tcpdump/libpcap public repository URL `http://www.tcpdump.org`
[19] Duarte A *et al.* 2008 *Journal of Physics: Conference Series* **119** 052014
[20] RapidMiner 2014 Predictive analytics, data mining, self-service, open source - rapid miner. URL `http://rapidminer.com`
[21] Thain D, Tannenbaum T and Livny M 2005 *Concurrency and Computation: Practice & Experience* **17** 323–356