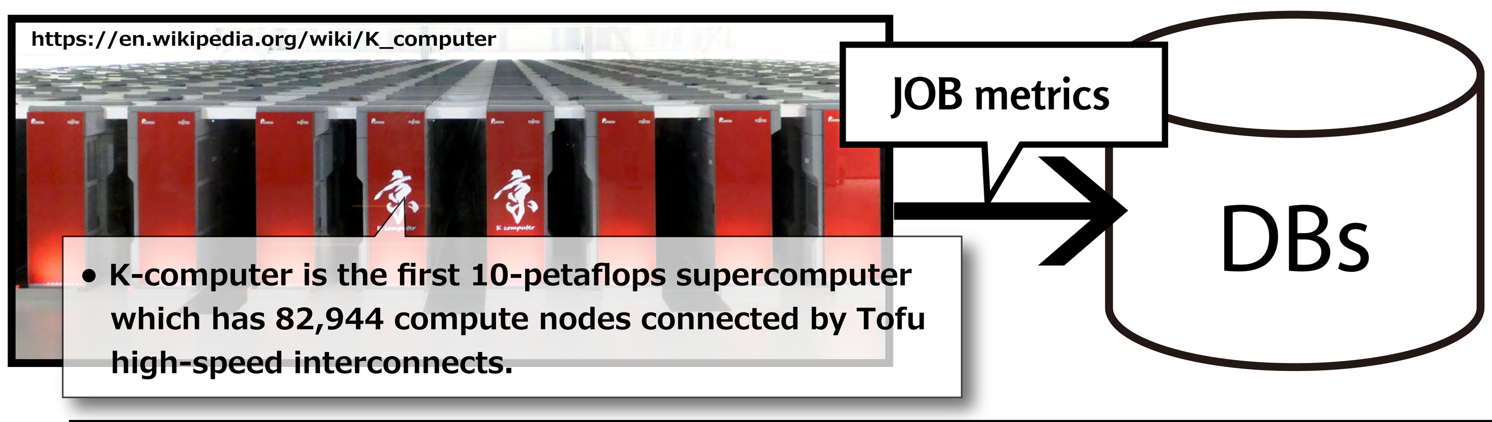
Performance Classification of the K-computer Workloads using Hierarchical Clustering and *k*-means

Masaaki Terai[1], Riku Kashiwaki[2], Fumiyoshi Shoji[1]
1) RIKEN Advanced Institute for Computational Science (AICS), 2) Graduate School of Simulation Studies, University of Hyogo

Background and Motivation

- The K computer [1] collects various metrics (e.g., elapsed time, memory usage, FLOPS, memory throughput, time stamps, and hardware counters [2]) for all jobs and stores them in databases.
- This mechanism was designed for the accounting system to charge a fee based on actual usage.
- A part of the information is directly provided to users with a file automatically generated by the system or monthly report summarized by administrators.



- We think that there is room to exploit more meaningful information based on log analysis for improving our operations, procurements and so on.
- As related works, some interested studies to classify workloads or diagnose system with metrics have been reported [3,4].
- On the other hand, there are no training data set to distinct types of workloads. Therefore, we need to use some unsupervised methods.

Objective:
- In order to get the picture of K's usage based on actual workloads performance for providing more efficient user support (e.g., performance improvement), we attempt to classify them using hierarchical clustering and *k*-means methods.

Preprocessing

- The job manager and peripheral tools collect more than 120 metrics for each job.
- To easily handle them, we only extracted records of the batch job type with the normal model from the databases because the records of this type are dominant and account for more than 75% of all the node-time in the valid records, as shown in Fig. 1 to 4.
- We eliminated the categorical metrics from the original data set and then obtained the 24 metrics as shown in Table 1.
- We ignored stage-in/out metrics.
- In this study, we used the one-year data set collected from Oct. 1, 2016 (2016-2nd) to Sep. 30, 2017 (2017-1st).
- The number of targeted workloads is 363,015. (the number of total records stored in DBs is nearly one million.)

Table 1 Job Metrics

24 metrics

metric name	description
alloc_core_total	num of cores actually allocated to compute nodes
alloc_node_num	num of nodes actually allocated to compute nodes
cpu_mem_read_count_ratio	cpu_ memory_read_counts per cycle_counts
cpu_mem_write_count_ratio	cpu_memory_write_ counts per cycle_counts
elapsed_time	elapsed time in a job
file_io_size	file I/O size
floating_inst_ratio	floating_inst. counts per cycle_counts
flops	num of floating-point op. per elapsed time (FLOPS)
fma_inst_ratio	FMA inst. counts per cycle_counts
io_transfer_size	I/O transfer size
max_cycle_counts	max cycle counts in all threads
max_use_mem	max memory usage in a compute node
mem_th	memory throughput
op_intensity	arithmetic intensity (=flops/mem_th)
read_system_call	num of read system call
req_core_num	num of cores required by a job script
req_node_num	num of nodes required by a job script
simd_floating_inst_ratio	SIMD-floating inst. counts per cycle_counts
simd_fma_inst_ratio	SIMD-FMA inst. counts per cycle_counts
sleep_cycle_ratio	sleep_cycle_counts per cycle_counts
use_core_total	num of cores actually used for a workload
use_node_num	num of nodes actually used for a workload
use_nodetime	product of use_node_num and elapsed_time
write_system_call	num of write system call

target periods

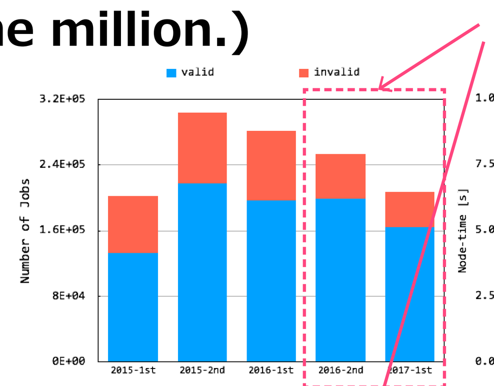


Fig. 1 Number of valid/invalid jobs (count-base)

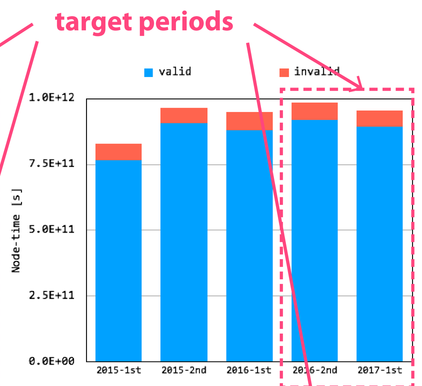


Fig. 2 Number of valid/invalid jobs (nodetime-base)

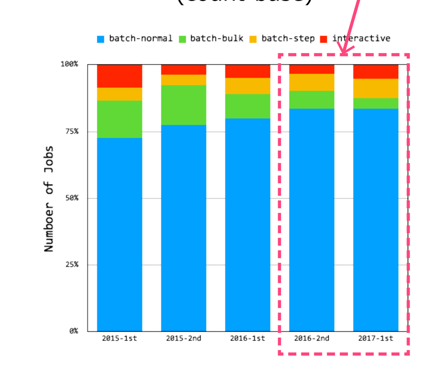


Fig. 3 Rate of job types/models in valid data (count-base)

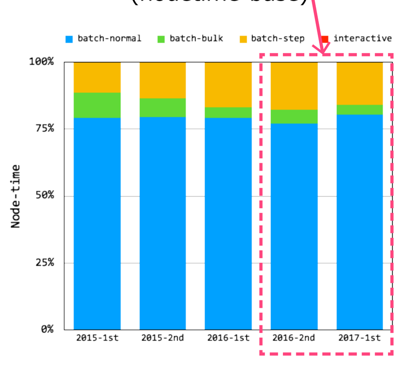


Fig. 4 Rate of job types/models in valid data (node-base)

- To equally compare metrics with other workloads, part of metrics (e.g., "cycle_counts" and "sleep_cycle") derived from hardware counters are normalized by using "cycle_counts."
- The metrics have substantially left-skewed distribution as shown in Fig. 5.
- To be spread between records more uniformly, we applied the logarithmic transformation.
- Furthermore, we standardized all the metrics (μ=0, σ=1) to easily compare one with others.

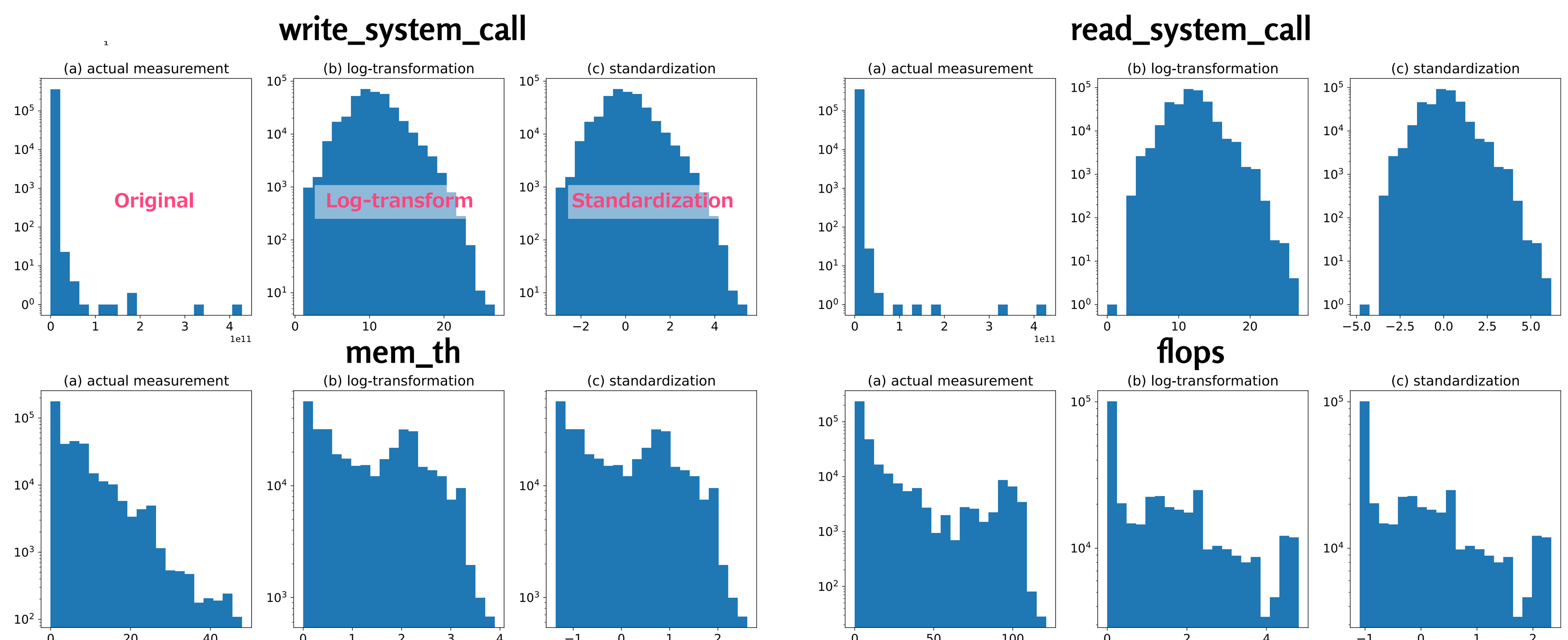
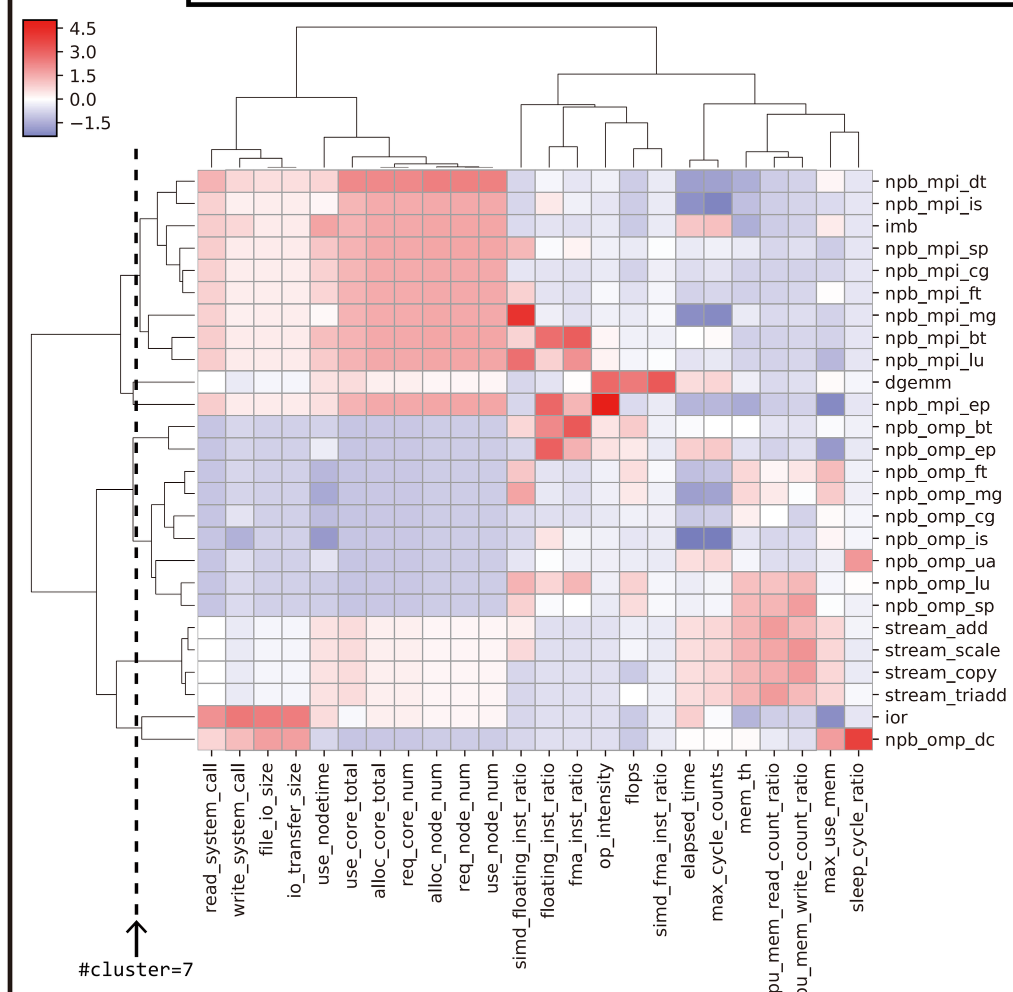


Fig. 5 Histograms for the metrics

Clustering for Benchmarks



- To confirm metric behavior, we classified typical HPC benchmarks (e.g., DGEMM, STREAM, IOR, Intel MPI Benchmarks) with hierarchical clustering.
- In Fig. 6, each column was standardized. The red and blue color describe the values higher and lower than the mean values.
- "Flops" reflects DGEMM behavior well. Also, "mem_th" reflects STREAM behavior.
- To determine a minimum number of clusters, we used a criterion that divides intensive workloads (DGEMM and STREAM) into different groups.
- Based on the criterion, we obtained 7 clusters.

Fig. 6 Comparison between typical HPC benchmarks and metric behavior using hierarchical clustering

Clustering for Real Workloads

- We classified the real workloads by *k*-means using the number of clusters: *k*=2 to 50.
- Based on the elbow methods as shown in Fig. 7, we can find some candidate number of clusters (*k*=7, 20 and 37), where *k*=37 is ignored due to space limitation of this poster.

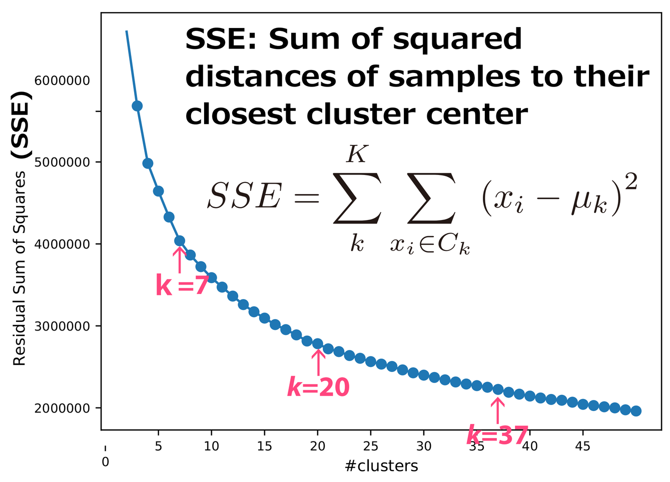


Fig. 7 Sum of squared distances

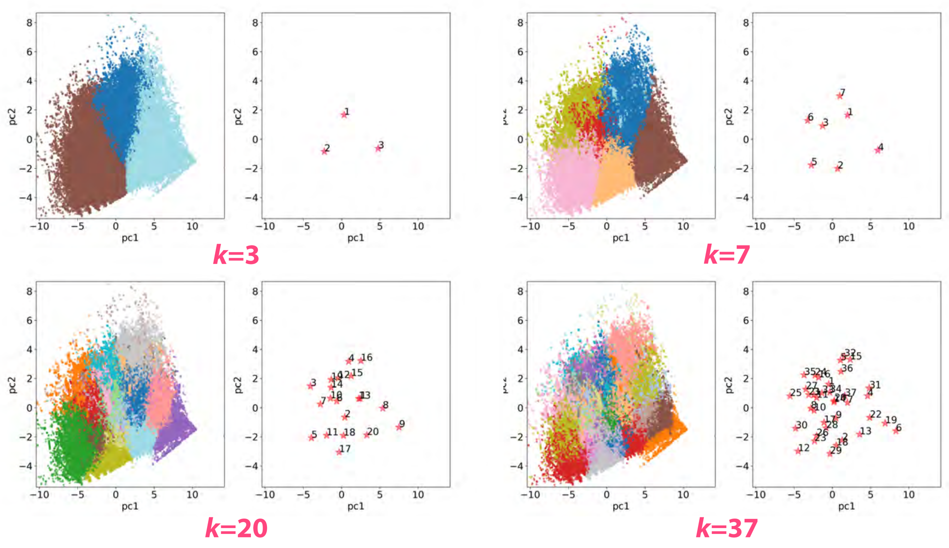


Fig. 8 Job distribution and centroids plotted on PCA subspace

To extract workloads required for some improvements, we use the criterion:
use_node_num, elapsed_time > 0
flops , mem_th < 0

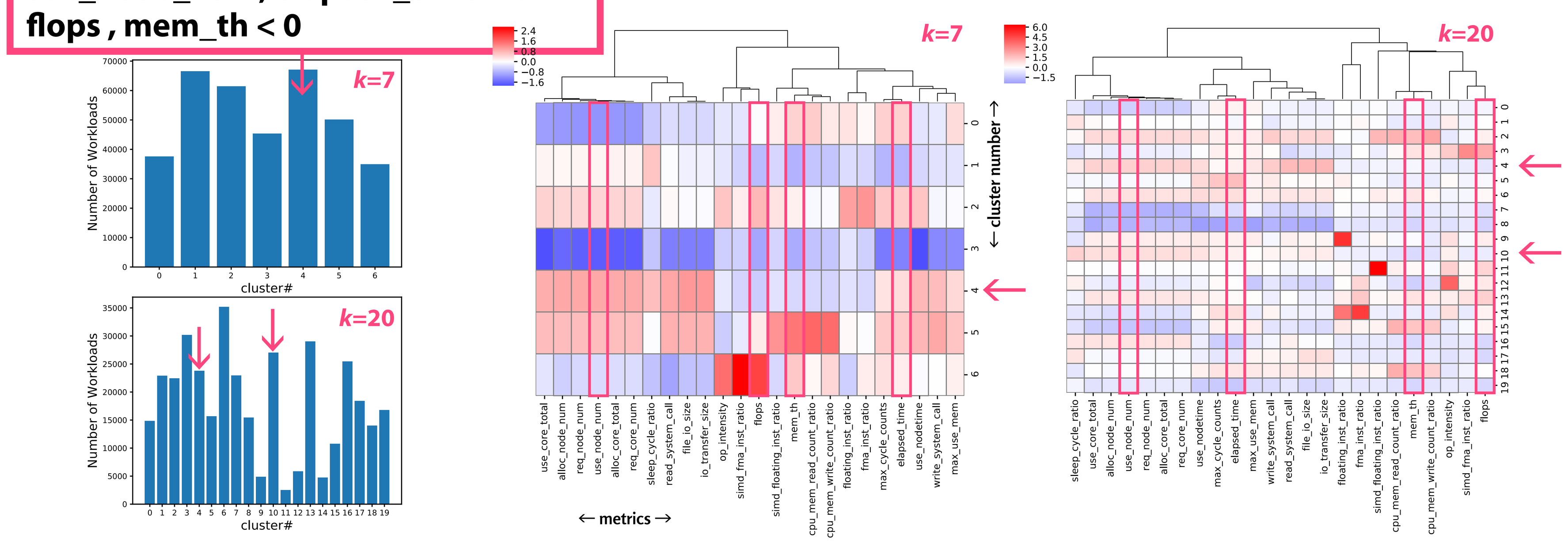


Fig. 9 The number of classified workloads and heat maps of metric behavior in *k*=7 and 20

- As shown in Fig. 9, for *k*=7 and 20, we extracted 67,043 and 50,835 workloads that consumed considerable resources with low performance.
- They're candidates for the performance check in more detail.

Summary

- To evaluate the capability of data mining with the metrics of the massive workloads on K, we classified typical benchmarks and real workloads using hierarchical clustering and *k*-means, respectively.
- In the evaluation using the typical HPC benchmarks, we classified workloads regarding arithmetic, memory throughput, and I/O intensive workloads with the 24 metrics. (These metrics worked out we had expected.)
- Also, *k*-means worked well. We classified the 1-year workloads (363,015 workloads) and obtained some candidates for the workloads we need to confirm and improvement for performance.
- Base on the result of SSE, we think that the appropriate number of clusters is expected the range from *k*=7 to 37.
- We need other clustering methods (e.g., DBSCAN) to precisely determine the number of clusters.
- In future, we will use other metrics from the databases and analyze the entire of the workloads.

References

[1] K. Yamamoto et al. 2014. The K computer Operations: Experiences and Statistics. Procedia Comp. Sci. 29, Supplement C (2014), 576 – 585. https://doi.org/10.1016/j. procs.2014.05.052
[2] Fujitsu Limited. 2010. SPARC64VIIIfx Extensions. (April 2010). Retrieved January 14, 2018 from http://www.fujitsu.com/downloads/TC/sparc64viiifx-extensions.pdf
[3] O. Tuncer et al. 2017. Diagnosing Performance Variations in HPC Applications Using Machine Learning. In High Performance Comp. - 32nd Inter. Conf., ISC High Performance 2017, Frankfurt, Germany, June 18-22, 2017, Proc. 355–373. https://doi.org/10.1007/978-3-319-58667-0_19
[4] Fei Xing et al. 2014. HPC Benchmark Assessment with Statistical Analysis. Procedia Computer Science 29, Supplement C (2014), 210 – 219. https://doi.org/10.1016/j. procs.2014.05.019 2014 International Conference on Computational Science.