Procedia Computer Science

Volume 29, 2014, Pages 1677–1687

ICCS 2014. 14th International Conference on Computational Science

Hybrid Scheduling Algorithm in Early Warning Systems

Denis Nasonov[1] and Nikolay Butakov[1]

[1]*ITMO University, Saint-Petersburg, Russia*

denis.nasonov@gmail.com, alipoov.nb@gmail.com

Abstract

Investigations in development of efficient early warning systems (EWS) are essentially for prediction and warning of upcoming natural hazards. Besides providing of communication and computationally intensive infrastructure, the high resource reliability and hard deadline option are required for EWS scenarios processing in order to get guaranteed information in time-limited conditions. In this paper planning of EWS scenarios execution is investigated and the efficient hybrid algorithm for urgent workflows scheduling is developed based on traditional heuristic and meta-heuristic approaches within state-of-art cloud computing principles.

Keywords: workflow scheduling, EWS, hybrid algorithm, hard deadline, urgent computing

1 Introduction

Today Early Warning Systems (EWS) play highly critical role in monitoring, prediction and reaction on upcoming hazards. Their applicability can be found in the following crucial areas: natural catastrophic detection – tsunami, flood, earthquakes; detection of sudden and significant economic changes; distribution of an epidemic; virus infection, etc. Examples of engineering and scientific area where EWS are extremely needed can be found in (Borensztein, 2004) (Mostashari, 2003) (Zou, 2003) (Hong, 2007). The growing emphasis on EWS has stimulated research activity towards its architecture development aspects and main design principals.

Nowadays EWS middleware is based on computational and communication infrastructure. Moreover communication infrastructure is used not only to support distributed computations but also to provide interaction between all participants of the system (Grasso V. F., 2011). The most adopted approach to build EWS is service-oriented architecture (SOA) with loosely-coupled replaceable components. It was used to build EWS for *UrbanFlood FP7* (Krzhizhanovskaya, 2011) (Balis, 2011).

Often a simulation step in EWS needs to be completed in a limited certain time and requires significant computational resources thus it can be considered as urgent computations. For example, *Japanese earthquake early warning system* uses the primary wave data to compute the destructive

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.153

secondary wave in approximately two minutes and then issues warnings to the civilians (Leong, 2013) (Yamasaki, 2012).

For now, dedicated resources are the most widely adopted and reliable method for supporting urgent computing. This method is used for the mentioned Japanese earthquake EWS, the *German National Meteorological Service* and the *North Carolina Forecast System* for storm surge (Leong, 2013) (Blanton, 2012). The main drawbacks of this approach are economical inefficiency and vulnerability of computation process to damaging of dedicated resource. To address this issues Beckman et al created a system called *SPRUCE* (Beckman, 2006). Its main features are an elevated priority, preemption and/or next-to-run techniques. It also includes procedures to prepare in advance the required scientific applications on the resources. *SPRUCE* provides perhaps the most relevant work and complete solution on several *TeraGrid* resources to address the urgent computation infrastructural challenges (Leong, 2013). But it has problems with task preemption since restriction policies of resource providers exist. Also it doesn't address issues like fitted algorithms, reliability, etc.

Leong (Leong, 2013) brings the idea that for rare and intensive urgent computations, leveraging of the existing e-Infrastructures can be proper idea in economical and other meanings. He notes that getting access to an e-Infrastructure typically involves strict and time-costly procedures, but this is particularly true for expensive HPC infrastructures, e.g. PRACE, that has restricted access to external resources as one of the drawbacks. At the same time, public cloud has ideal resources in terms of the configuration and accessibility. Also cloud providers like Amazon offer both HPC and HTC resources. It also has the advantage of being available on demand and thus is suitable for rarely occurring use cases when cost is taken into consideration. Leong concluded, in order to prepare existing e-Infrastructures for urgent computing, aspects of scalability, robustness, fault tolerance, have to be carefully considered.

In order to use proper and efficiently computational resources provided by cloud providers like Amazon and Google, shared between many users in any time, we need planning algorithm which is able to account many specific characteristics of different computational resources. Moreover, the scheduling algorithm must be dynamic because it needs to respond to environmental changes such as: time-dependent changes of the data transfer channels and computational node performances, the agility of the available resources, bad-estimated times of execution or transfer of the data, appeared task failures. From another hand, scheduling algorithm should be effective in the planning on private dedicated resources as well as comply EWS workflow's certain restricted conditions imposed on the execution time and resource reliability. Moreover these conditions are variable for the different system stages, types of workflow and user exclusive rights.

The main objective of this work is to present our vision on open research challenges in EWS scenarios planning, and develop efficient hybrid algorithm for the scheduling of urgent workflows to meet hard constraints.

Generally workflow scheduling problem is an Non-deterministic polynomial (NP)-complete problem. For now, many classes of heuristics scheduling algorithms are invented. In this paper we are interested in two of them, mostly used in workflow planning. The first one is class of list-based heuristics. Basically it sorts workflow tasks by some criteria and then sequentially maps to best fitted resources. The main advantages of this class are small working time and suitable quality of generated solution. The second one is class of meta-heuristics, represented by GA (Genetic Algorithm), PSO (Particle Swarm Optimization), ACO (Ant Colony Optimization) and others. It searches through all space of solutions and thus generates better solution than previous class, but it requires much more time for its work. Also working time of such algorithms rapidly increases with increasing of the resources amount and the amount of tasks in a workflow.

Rahman M. et al in (Rahman, 2013) makes review of general adapted scheduling algorithms. He starts with the simplest job scheduling algorithms "Min–min" and "Max–min", which operate only with ready tasks and don't consider whole workflow. Also algorithms HEFT (Heterogeneous Earliest Finish Time), GRASP (Greedy randomized adaptive search procedure), GA, DCP (digital signal

processing) are analyzed. Than Rahman compares scheduling algorithms in the static and dynamically changing conditions. In static conditions GRASP and GA shows much better results than others, but in case of dynamical changing environment DCP and HEFT appears to show better than GRASP and GA. Also, GRASP and GA significantly loses in computation time. The author pays attention only to makespan of workflow and doesn't guarantee meeting any deadline.

Fard H. (Fard, 2012) proposes a multi-objective workflow scheduling algorithms based on HEFT. In contrast to most part of other multi-objective algorithms, it optimizes more than two parameters: makespan, cost, energy consumption, reliability. But the algorithm works only in static environment.

Durillo J. (Durillo, 2013) proposes bi-criterial optimization algorithm for workflow scheduling in Amazon EC2. It builds schedule for composite application with using of different configurations of virtual machines (VMs) and different count of reserved VMs. It works only in static environment.

Bolze R. (Bolze, 2009) presents algorithm based on HEFT heuristics for working with multiple workflows and dynamic adding workflows for executing.

Mao M. et al (Mao, 2011) optimize computational cost for composite workflows with pre-defined soft deadlines in Amazon EC2. The authors use different workload patterns to add composite application to system in the course of time. For scheduling, simple heuristics called EDF are used. The author accepts dynamic conditions but don't try to meet deadline precisely.

Papers (Abrishami S. N., 2012), (Abrishami S. N., 2013) are dedicated to list-based heuristics for scheduling with hard deadlines. The first work presents PCP heuristic for GRID systems. The second one proposes two modifications of PCP for cloud computing: IC-PCP и IC-PCPD2, accounting pay-as-you-go model. All algorithms demonstrate better performance than HEFT, but works only in static environment.

In (Wieczorek, 2009) authors made a good survey of theoretical workflow scheduling approaches among which were selected several hybrid-based ideas that were presented in (Sakellariou, 2004) and (Yu, 2007). One of the first hybrid heuristic algorithms was proposed in (Sakellariou, 2004). The key idea of the hybrid heuristic was to use a standard list scheduling approach to rank the tasks of the workflow and then use it to assign tasks to groups of tasks that can be subsequently scheduled independently. In (Yu, 2007) author made dynamical rescheduling based on HEFT algorithm. In the both papers deadlines were not taken into consideration as well as no meta-heuristic was used.

Hybrid algorithms with heuristics and meta-heuristics combination were in detail discussed in (Kołodziej, 2012). But all the scheduling study is based on tasks planning issue in the Grids only and there is no extension on workflow cases.

A very interesting idea was proposed in (Rahman, 2013). The idea is to develop Adaptive Hybrid Heuristic, which will be able to utilize better solution generated by meta-heuristic algorithm and dynamic adaptation to changing world provided by DCP-G algorithm

Our work has several significant differences from previously discussed works. As EWS might contain complex multilevel scenarios, which include high computation simulations with composition of different models, resources and attraction of the large experts groups, it requires particular other approaches for the development and operating EWS stages than traditionally can be suggested. We propose new hybrid scheduling algorithm with supporting hard deadlines that is based on provided resource reliability and model performance data. We also propose procedure of public offered resources usage to increase accuracy of simulation and reliability of tasks. In additional, we use principals of the cloud computing platform CLAVIRE (Knyazkov, 2012) with deadline-driven resource management concept (Kovalchuk, 2013).

2 Background: aspects and requirements for EWS scheduling

Before we can introduce our scheduling approach for EWS workload, crucial aspects are presented below.

2.1 Environment

By assumption, EWS architecture used in this paper is based on the second generation cloud infrastructure. It provides effective functionality and unique features for EWS development and support, including:

- supporting of composite applications (workflows);
- models (software packages) unification;
- computational resource heterogeneity : pc, clusters, supercomputers, grids, first generation computing clouds, virtual machines;
- concept of private and public clouds;
- resource and package hot deploy;
- runtime workflow modification and so on.

All EWS scenarios are formed as composite applications or workflows (WF) and consist of the linked tasks. Tasks or jobs can be run on the different heterogeneous resources that can be provided through the private and public clouds. Private (dedicated) cloud includes resources that can be used and are fully controlled only by its owner-organization and have no access from extranet. Public cloud is based on all other resources that are held by external providers and can be used in computational purposes only through the SLA.

Generally EWS lifecycle can be divided in the three main states (phases): normal state, urgent state and recovery state. Normal state represents scenarios that run continuously day-by-day with the main purpose of monitoring input data from external sensors and prediction abnormal situations which can lead to the certain type of disaster. Also this state includes execution of workflows that were developed during scientific community's activity, system enhancement and stuff learning process.

Urgent state starts at the time when abnormal behavior is detected. Usually it contains scenarios with significant critical computational part, complex logic and involvement of large expert groups. Also it is absolutely critical to provide scenarios result timely and guarantee system reliability.

Recovery step is needed to return whole process back to normal state. The reason why it should be broken out is the "starving" option for all currently running tasks when urgent scenarios start to execute. During urgent phase all monitoring and prediction workload jointly with other less critical activity should pass to resource-saving mode or have to be preempted. After urgent phase is finished, all activities that were preempted have to be returned back to executed phase if it is supported by the infrastructure and all the monitoring and prediction workflows have to process all cached for disaster period data and pass back to normal running.

2.2 EWS scheduling priorities

As it was noted before, beside other characteristics, in order to be effective warning must be timely and reliable. In this case, EWS scenario workflows should have the most critical priority among all the other system workload. Moreover, all internal EWS workflow tasks are marked with their own deadline type. Three deadline types are supported in the system: non-deadline type; soft deadline and hard deadline.

Non-deadline workflows are irregular package executions usually connected with scientific research and system enhancement. Other activity is represented by testing runs and EWS modules learning processes.

Soft deadline workflows have several principal use cases: all the jobs created at runtime for checking some expert's assumptions or for other crucial reasons that only extend main scenario and are submitted to the system for execution during the urgent phase; all tasks made by parameter sweep option only for elaboration and increasing accuracy of any result.

Hard deadline workflows are related to all EWS constructed scenarios. It's necessity to get information just in time in order to make right decisions. Even a minute delay can lead to devastating

consequences. To meet these requirements we expect that resource reliability information is provided. The reliability information such as failure rate, execution time distribution can be statistically extracted from historical execution data.

Combine deadline types with required reliability and heterogeneous computational resources, provided by the infrastructure from private and public clouds we get input conditions for scheduling algorithm.

2.3 Evaluation of traditional approaches

Today, as was mentioned before, different classes of scheduling algorithms and a great amount of proposed solutions for workflow planning exist. Many of them can be used one way or the other to EWS scenarios scheduling, but there is no algorithm that can take in account all described conditions and be effective enough to meet EWS needs.

The scenarios core is designed at the development EWS phase and can be considered as unchanging part of the system. All modifications that take place during scientific researches are rare and in the majority not fundamental. This statement allows us to take class of evolution scheduling algorithms and try to use the following schema.

1. At normal phase, learning procedure is performed. As this phase takes almost all execution time in comparison with other phases process should find a plan that will overbid all other dynamic-based algorithm.
2. If some modifications of EWS scenarios are implemented, procedures from first and second points are repeated.

This schema has several crucial drawbacks: if some task exceeds planned time or some node of the dedicated resources (private cloud) become down evolution algorithm dramatically lose its chance to get in time which is limited by deadline. Especially this is actual when such issue happens at the beginning of scenario execution. Other limitation is impossibility to run immediate unexpected tasks that can form branched scenario from main scenario or executed as separated workflow. Otherwise using only list-based dynamic scheduling will be not so effective and moreover no guarantee that hard deadlines will be met especially in the case of task computation delays and resource fails.

3 Hybrid algorithm for EWS workflows scheduling

In this paper hybrid scheduling algorithm with hard deadline execution possibility is proposed. The logical core of hybrid algorithm consists of meta-procedure that integrates evolutionary, list-based and dynamic job planning algorithms in time changing environmental and computational conditions. Due to used replication's reliability, workload priorities and strong parts of each traditional scheduling approaches the proposed hybrid solution provides sufficient capability to comply with EWS execution hard requirements.

The main planning procedure of scheduling algorithm is shown on the figure 1. It starts planning iteration on coming system events which are generated on the following actions: new workflow is submitted; task is completed or interrupted; task's time exceeded the limit; resource is unavailable. As plan for resources allocation was generated by evolution algorithm (GA) during the learning process, the first step checks its consistency. The consistency check compares resources from scheduling plan with live available resources for ready-to-run tasks. If contradictions aren't found, planed tasks go to execution module. If unsuspected WF has been submitted to the system, adapted dynamic algorithm is used. In this case scheduler allocates resources only from remain part of private cloud and all public cloud. If GA generated plan is inconsistent all scenario and unsuspected WF are planned by HEFT and dynamic algorithms, using all available resources. Concurrently new static plan is generated by evolution algorithm. Till this plan will be worse that suggested by HEFT, HEFT scheduling is used.

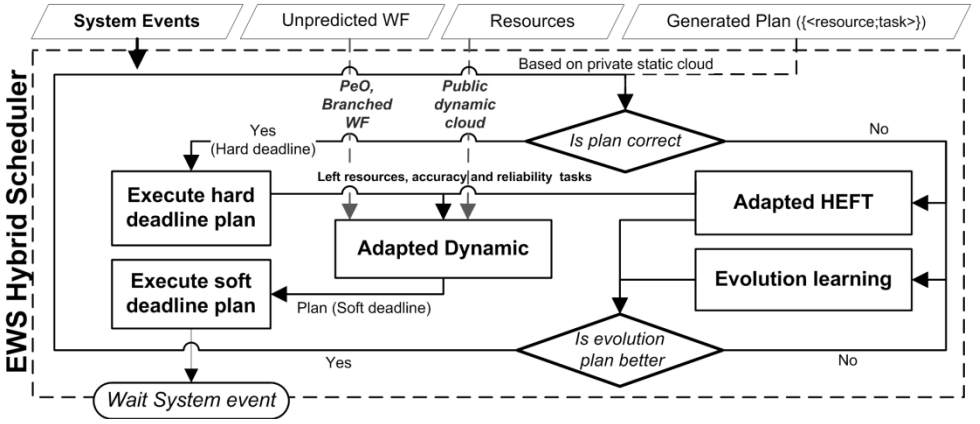

Figure 1: Main schedule procedure

3.1 Reliability consideration

The reliability of the resource consists of two parts - fail rate and execution time probability. Let's consider that we have M resources. The fail rate for each resource is:

$$P_m(t) = \frac{N_m - n(t)_m}{N_m}, \quad (1)$$

where $m = \overline{1..M}$, N_m- number of successful task completions on resource m, and $n(t)_m$- resource fails during tasks execution. Task execution time is estimated by normal distribution $\mathcal{N}(\mu_{mk}, \sigma^2_{mk})$ for each package k with mean μ_{mk} and variance σ^2_{mk} by assumption. Since we consider all historical data as independent sample for each package κ with sample mean $\bar{X}_{mk}$ and variance S^2_{mk}, we can use confidence interval for needed probability measurement:

$$Pr_{mk}\left(\bar{X}_{mk} - t_{\frac{1-\alpha}{2},n-1}\frac{S^2_{mk,}}{\sqrt{n}} \le \mu \le \bar{X}_{mk} + t_{\frac{1-\alpha}{2},n-1}\frac{S^2_{mk,}}{\sqrt{n}}\right) = \alpha = CI_{mk}(\alpha), \quad (2)$$

where $t_{\frac{1-\alpha}{2},n-1}$- percentile of Student's t-distribution.

Composing $P_m(t)$ and $CI_{mk}(\alpha)$ we can get aggregated probability at the each iteration of planning for the each package on the each resource:

$$Pc_{mk} = P_m CI_{mk}(\alpha), \quad (3)$$

The required reliability for each job can be achieved by replication on the public resources. Probability of occurrence of at least one of the events (task completion in time) is:

$$P_{req}(A) = 1 - P(\overline{Pc}_{m_1k}\overline{Pc}_{m_2k}..\overline{Pc}_{m_ik}). \quad (4)$$

It's important that job planned execution time should satisfy following condition $t\_calc_k \le \bar{X}_{mk} + t_{\frac{1-\alpha}{2},n-1}\frac{S^2_{mk,}}{\sqrt{n}}$ this can be achieved by variating of α parameter. Finally we can have optimization by makespan and resource amount or price:

$$optR\left(k, t, P_{req}\right) = \min_{1-P\left(\overline{Pc}_{m_1k}\overline{Pc}_{m_2k}..\overline{Pc}_{m_ik}\right) \ge P_{req}(A)}(i) \quad (5)$$

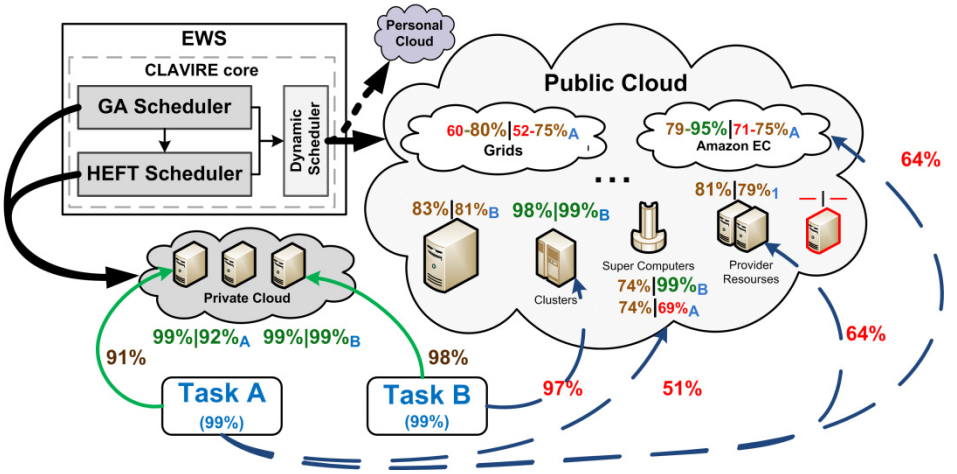

Figure 2: Example of using reliability principals (for each task on each resource P_m and CI_{mk} are shown)

If public cloud contains different types of resources with their own SLA and performance characteristics optimization task by cost and resource amount can be applied. In this paper optimization by used resources amount only used. On the figure 2 reliability principals are shown, two tasks should have 99% guarantee to be finished in time. On private resources by scheduling with GA or HEFT algorithms they have probability of successful execution 91% and 98% corresponding. All deficient reliability is filling by public cloud with help of dynamic reactive algorithm.

3.2 Hybrid algorithm implementation

In the figure 3 we provide the algorithm description that we used in our experimental investigations. It includes HEFT and GA generation algorithms with slightly modified classical implementation, also account reliability probabilities of the resources is taken into consideration.

```
1. PLANNIG_ALGORITHM:
2. PROCEDURE: Hybrid
3. Input: workflows Wst(static); Wd(dynamic); GA plan P;
4.        resource sets PrR(private), PbR(public); reliability R
5. begin
6.   if (!CorrectGAPlan(P)) begin
7.     StaticTaskList ← HEFT(Wst, PrR)
8.     P ← GAGeneration(Wst, PrR)
9.   end
10.       DynamicTaskList ← DynamicPlan(Wd, Wst, PbR,
   StaticTaskList)
11.      end
12.      PROCEDURE: CorrectGAPlan
13.      Input: workflows Wst; GA plan P; resource sets
   PrR(private)
14.      begin
15.        foreach w ∈ Wst begin
16.          ReadyToRunTasks.add(w.select(T ← T.ready == true))
17.        end
```

```
18.         foreach t ∈ ReadyToRunTasks begin
19.           if   (!P[t].Resource.Availible   ||   (t.startTime   <
    currTime && !P[t].Resource.Free)) return false;
20.         end
21.       end
22.       PROCEDURE: DynamicPlan
23.       Input: Wst, Wd; PrR, PbR; R
24.       Output: Plan dP
25.       begin
26.         sort Wst by makespan
27.         foreach w ∈ Wst begin
28.           if (w.reliability < R) begin
29.             dP.add(Find [pbr m] ∈ PbR using (5))
30.           end
31.         end
32.         sort Wd by deadline and makespan
33.         foreach w ∈ Wd begin
34.           dP.add(Find pbr ∈ PbR)
35.         end
36.       end
```

Figure 3: General steps of used hybrid algorithm

4 Experiments

For the experimental investigations five practically used workflows were taken. Their descriptions and XML representations can be found in (Applications, 2014). One of them is CyberShake workflow that was invented by Southern California Earthquake Center and can be used in the EWS to characterize earthquake hazards. For the experiments following restrictions are assumed. There are three workflow configurations: small(20-30 tasks), medium(45-60 tasks) and large(100 tasks). Every task can be computed only on one computational node at the same time. Every computational node has two predefined characteristics: computational power in flops units and reliability rate in percents. There are three resource reliability rate configurations: 60%, 75% and 95%. Computational task capacity is taken from runtime attribute in the XML representation of workflow and is multiplied by predefined constant value (20 flops). For dedicated resource we take 4 computational nodes with following computational powers: 10, 15, 25 and 30. In case of public resources configuration with 3 resources with different nodes count is used (table 1). Simulation experiments run 100 times for every pair <WF size; reliability rate> of the parameters.

I resource (node count/flops)	II resource (node count/flops)	III resource (node count/flops)
6/25, 3/30, 3/15, 3/10	2/30, 4/25, 3/15, 3/10	1/30, 4/25, 2/15, 2/10

Table 1: Public cloud resources and nodes characteristics

Also, we assume that transfer time between nodes is constant, but different for the resources, and task preempting is supported only on dedicated resource. The last feature is useful if a replicated task on public resource finishes earlier than original task on dedicated resource.

For simulating dynamic changes in the resources we use the following strategy: when a task starts on a node we generate random number from (0,1) and compare with reliability rate if generated

number is greater than the rate we determine time of failure by multiple execution time to another generated random number. In public resources we also assume that nodes have variable performance and simulate it by using execution time probability distribution. Temporal unavailability of the node is used as the resource failure.

HEFT and HEFT with improved reliability. We compared classical HEFT with HEFT enriched by increased reliability on public resources (figure 4).

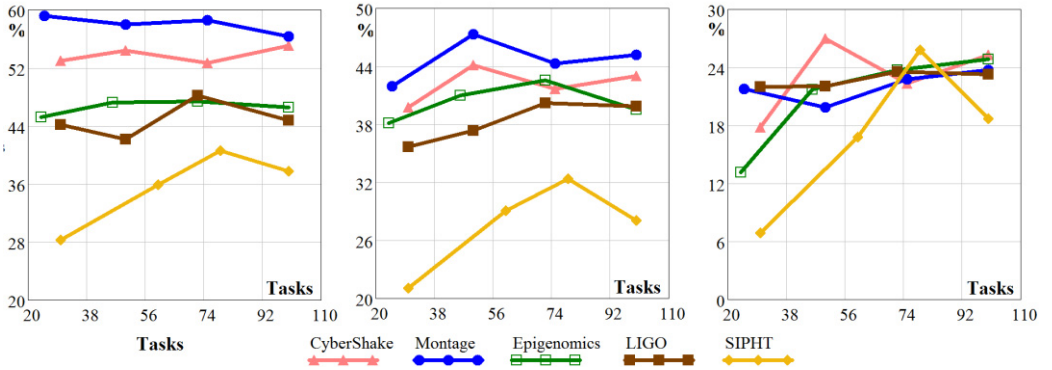

Figure 4: Makespan improvement for 60%, 75%, 95% dedicated resource reliability rate

As we can observe makespan difference is highly variational with workflow type. But absolutely in the all cases significant improvement is shown. For the small workflows with resource reliability 60% improvement fluctuates from 28% to 60% and reduces its speedup with increasing WF size. The reason is hidden in the quantitative public resources limitation used in these simulations. The more tasks require public resources for reliability increasing, the lower chance to comply with every request. Also this is the reason for increased speedup from small to medium workflows on the second and third cases with higher initial reliability rate 75%, 95% (smaller resources required for reliability compliance). The significant difference in 2-4 times between SIPHT with lowest result and Montage with the highest score can be attributed to the WF structure. Montage has the most pipelined structure and thus has the most chance to escape task fails with improved reliability. SIPHT by contrast has splitted structure.

GA, GA with HEFT, GA with HEFT and improved reliability. We compared classical GA with GA and adapted pure HEFT and the case with HEFT enriched by increased reliability on public resources (figure 5).

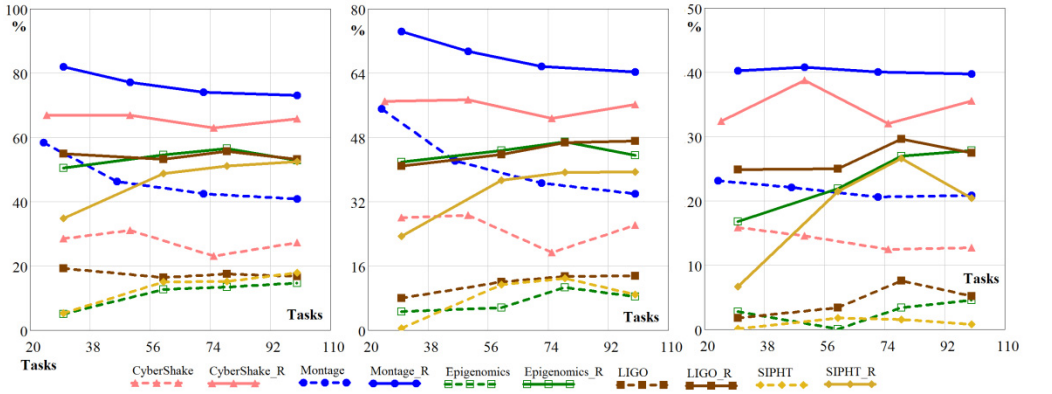

Figure 5: Makespan improvement for 60%, 75%, 95% dedicated resource reliability rate (broken line GA+HEFT, solid line GA+HEFT+reliability)

From the plot for GA and GA + HEFT boots we can assert the presence of the same but less distinct tendencies which were found in the previous experiments but with other occasion. So CyberShake and Montage show the best improvement up to 36% and to 60% consequently. Epigenomics, LIGO and SIPHT have valuable makespan reduce only on the small workflow or low initial reliability 60%, 75%. For SIPHT there is almost no effect on the high reliability. These results can be explained by HEFT rescheduling advantages, when some resource is unavailable GA has to wait it, while HEFT changes its plan. Best results were achieved for GA + HEFT with reliability. Even for 95% case we have up to 41% of makespan profit that indeed confirms effectiveness of hybrid method with included reliability.

5 Conclusion

We presented EWS-oriented hybrid scheduling algorithm based on the second generation cloud platform. It includes: hard deadline planning for main EWS workflows assured by reliability increase; dynamic appeared workflows execution on the left public and dedicated cloud resources; adjustability to environment changes. We also experimentally show its makespan effectiveness in comparison to traditional approaches that vary from 6% to 60% for improved HEFT algorithm and is up to 82% for hybrid GA with HEFT and reliability algorithm, depending on initial conditions.

As the future works we plan to involve more complex probability models to better estimate and taking into account potential risks related to execution time. Using other improvement characteristics such a cost computational nodes, model allocation, time dependent availability also need investigations.

This work was financially supported by Government of Russian Federation, Grant 074-U01 and 11.G34.31.0019.

References

Abrishami, S. N. (2012). Cost-driven scheduling of Grid workflows using partial critical paths. *Parallel and Distributed Systems, IEEE Transactions on*. IEEE.

Abrishami, S. N. (2013). Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. *Future Generation Computer Systems* .

Applications. (2014). Retrieved from Pegasus Workflow Management System: http://pegasus.isi.edu/

Balis, B. e. (2011). *The UrbanFlood Common Information Space for Early Warning Systems*. Procedia Computer Science.

Beckman, P. e. (2006). SPRUCE: A System for Supporting Urgent High-Performance Computing. *Proceedings of the IFIP WoCo9 Conference* (p. USA). Springer.

Blanton, B. J. (2012). Urgent Computing of Storm Surge for North Carolina's Coast. *Procedia Computer Science* .

Bolze, R. D. (2009). Evaluation of Online Multi-Workflow Heuristics based on List-Scheduling Algorithms. *GWENDIA(ANR-06-MDCA-009)*.

Borensztein, E. B. (2004). *Assessing early warning systems: how have they worked in practice?* International Monetary Fund.

Durillo, J. J. (2013). Multi-objective workflow scheduling in Amazon EC2. *Cluster Computing*.

Fard, H. M. (2012). A Multi-Objective Approach for Workflow Scheduling in Heterogeneous Environments. *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE.

Grasso V. F., S. A. (2011). *Early warning systems: State-of-art analysis and future directions.* Draft report, UNEP.

Hong, Y. A. (2007). Towards an early-warning system for global landslides triggered by rainfall and earthquake. *International journal of remote sensing* .

Knyazkov, K. V. (2012). CLAVIRE: e-Science infrastructure for data-driven computing. *Journal of Computational Science* .

Kołodziej, J. (2012). *Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems.* Heidelberg: Springer.

Kovalchuk, S. V. (2013). Deadline-Driven Resource Management within Urgent Computing Cyberinfrastructure. *Procedia Computer Science*, (pp. 2203-2212).

Krzhizhanovskaya, V. V. (2011). *Flood early warning system: design, implementation and computational modules.* Procedia Computer Science.

Leong, S. H. (2013). *Leveraging e-Infrastructures for Urgent Computing.* . Procedia Computer Science.

MacLaren, J. S. (2004). Towards service level agreement based scheduling on the grid. *Workshop on Planning and Scheduling for Web and Grid Services* , (pp. 3-7).

Mao, M. H. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows. *High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE.

Mostashari, F. (2003). Dead bird clusters as an early warning system for West Nile virus activity. *Emerging infectious diseases* .

Rahman, M. H. (2013). Adaptive workflow scheduling for dynamic grid and cloud computing environment. *Concurrency and Computation: Practice and Experience* .

Sakellariou, R. Z. (2004). A hybrid heuristic for DAG scheduling on heterogeneous systems. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International* (p. 111). IEEE.

Wieczorek, M. H. (2009). Towards a general model of the multi-criteria workflow scheduling on the grid. . *Future Generation Computer Systems, 25(3)* , 237-256.

Yamasaki, E. (2012). What We Can Learn From Japan's Early Earthquake Warning System. *Momentum* .

Yu, Z. S. (2007). An adaptive rescheduling strategy for grid workflow applications. *In Parallel and Distributed Processing Symposium IPDPS 2007. IEEE International*. IEEE.

Zou, C. C. (2003). Monitoring and early warning for internet worms. *Proceedings of the 10th ACM conference on Computer and communications security* .