Evaluating Distributed Methods for CBR Systems for Monitoring Business Process Workflows

Ioannis Agorgianitis, Miltos Petridis, Stelios Kapetanakis, Andrew Fish

School of Computing, Engineering and Mathematical Sciences, University of Brighton, Watts Building, Lewes Road, Brighton, BN2 4GJ, UK

{I.Agorgianitis, M.Petridis, S.Kapetanakis, Andrew.Fish}@brighton.ac.uk

Abstract. This paper evaluates the potential for distributed business process workflow monitoring and management using the CBR paradigm. Recent developments in distributed computing technologies have shown capable for efficiency gains from effective distribution. Current models of CBR distribution are discussed and an extension is proposed focusing on the challenge posed from large data volumes. This is shown to be also associated with more challenges in the quality of data and the requirement for real time processing. The proposed approach is presented and a novel architecture for distribution of CBR systems is proposed. An evaluation of the approach and architecture is conducted and presented based on a set of experiments in the area of business process workflow management. The experiments establish a serial execution baseline and show promising high speedup gains, especially at large volumes (exceeding millions) of cases. It is shown that at high enough data volumes, there is a clear benefit of distributing some of the early parts of the CBR life cycle to a finer data and process granularity level. Such approach seems to maximise the benefit from the use of modern distribution technologies. Concluding, this paper signposts future areas of research leading to a more generic model that can maximise the efficiency gains of distribution in CBR systems.

Keywords: Case-Based Reasoning, Distributed Architectures, Distributed Case-based Reasoning, Business Process Workflows

1 Introduction

Modern industrial environments can be complex, incorporating multiple interrelated business processes. Increasingly, within organisations, business processes are captured, monitored and controlled by enterprise software systems. A side effect emerging in such organisations is the generation, propagation and utilisation of large datasets on a daily basis. The complexity and interoperability of business processes is responsible for the migration of standalone and “isolated” bespoke systems to large scale, distributed ones, sometimes utilising hundreds of thousands of physical computational nodes.

Advanced systems currently exist, which are capable of performing continuous process monitoring and control, like Distributed Control Systems (DCSs). Such systems can be responsible for storing, controlling, visualizing and analysing huge amounts of data related to internal business processes in real time. However, such systems usually rely upon human monitoring in order to reason upon workflow executions and provide corrective actions whenever they are required [1].

Human intervention can introduce high levels of uncertainty due to possible: malicious user behaviour, regular occurrence of human errors and expert and / or stakeholder absence at key business process execution stages. Additionally, such human related interventions may not be fully captured by systems and actions can be based on further communication and/or information exchange outside digital Business process monitoring systems.

Recent advances in information systems technologies have provided new capabilities for data management and exploitation of data, especially based on modern enterprise architectures and cloud computing. The literature shows a number of successful implementations related to the diagnosis and monitoring of business workflows using the CBR paradigm [2, 14, 16]. However, current attempts invariably focus on small-scale systems, leaving unknowns in terms of scaling and distribution, a frequent prerequisite for modern industrial implementations.

This work attempts to investigate the issues of scalability and distribution on CBR applications specialised on business workflow monitoring. Its structure will be as follows: Section 2 presents the relevant literature in intelligent business process management; Section 3 discusses the key challenges of distribution and presents possible models to address the challenge and proposes an operations-efficient architecture and Section 4 presents an evaluation that verifies its validity and performance. Finally, Section 5 presents the conclusions from this work and proposes further work in this area.

2 Background

Workflows and business process are two interrelated concepts. Nowadays business process definitions have been standardized to a large extend using industry acceptable standards like the Business Process Model and Notation (BPMN) [4], the XML Process Definition Language (XPDL) [5, 6] and the Web Services Business Process Execution Language (WS-BPEL) an executable language standard introduced by OASIS for specifying the behavioural aspect of business processes utilising Web Services [7].

Case-based reasoning [8] has been proven an efficient mechanism in monitoring business process workflows [1, 2, 14, 16] and possibly a competent model upon tackling uncertainty and fuzziness making use of past experiences along with extensive domain knowledge and expertise. Minor et al [14] have presented a CBR approach for representation and index-based retrieval of agile workflows, Dijkman et al [15] have shown a process model ranking against a repository of process models and Kapetanakis et al [2, 16] have proposed a generic architecture and framework, for intelligent monitoring of business workflows using CBR.

However, over the past years great advances have been noted in the area of highly distributed systems like the HTCondor system which provided multi-scheme utilisation and management of resources including exploitation of idle machines processing power, multiple installations and collaborations of HTCondor instances[17], the Berkley Open Infrastructure for Network Computing (BOINC) [19] which utilizes device idle time and general projects for commodity machine utilization like Apache Hadoop and Spark [18, 20, 21].

Plaza and McGinty[13] have proposed a classification for distributed CBR systems, based on the magnitude of knowledge and processing of data. This work suggested that the CBR knowledge content for distribution purposes is correlated to the number of case bases present. Despite the case that most CBR systems use a single case base, increasingly multiple case bases appear in systems due to the complexity and multi-provenance of data that can be seen in modern enterprise systems [3].

The integration and verification of such distribution techniques is becoming more and more challenging since it requires increased research efforts and labour. Our research work focuses on the identification of a potential effective approach and architecture for distributed CBR based implementations for large-scale business process workflows monitoring and management.

3 An Enhanced Categorisation of Distributed CBR systems

Our presented approach is driven by the current state of technology for data related operations, and additionally, the maximisation of operational utilisation across large data volumes. Our possible solution on the data volume issue comprises four main characteristics, building on top of the existing classification on CBR distributed systems [13]: Ground-up distribution, Agent capability to process big amount of data, Single case base to be distributed on demand and Distributed processing units managed by agents; enhancing distributed CBR systems [13] with the additional aspect of the data volume. Apart from the extra dimension of data volume, the initial classification remains the same.

It is worth mentioning at this point that a number of issues could arise by implementing distribution in a ground-up manner in terms of cases where the used algorithms are optimised for serial execution (such as business process workflows management and monitoring graph-based approaches [2,14,15,16]). In such cases, alternative algorithms may be used, or the distribution model may need to be adapted to provide efficiencies using the most appropriate type and architecture for distribution.

3.1 The Distribution Lifecycle of CBR in Business Process Workflows

Based on our proposed new distributed CBR component for data volume handing, this study proceeds further by presenting the areas on which the distribution could take place along with the identification of the CBR cycle’s operations to be conducted in each and every stage.

The data for the CBR system could come in various forms (Figure 1) such as raw text data, distributed raw text data (e.g. HDFS), document based data (e.g. MongoDB, Elastic), and so on. For their distribution to remote processing units (agents) an agent is selected (Coordinator) which may have control on the data dispatch, data redistribution, etc.

Having distributed the case base (raw) data in various nodes based on its volume, the system proceeds by loading the actual cases of the case base. As already discussed, business process workflow cases would normally be represented using graph based formats [14]. The process of graph initialisation is a computational intensive task, and consequently, a distributed approach could improve performance and avoid bottlenecks in the cycle execution. The indexing of the data may be implemented in various stages: At first, an initial indexing mechanism should occur at case creation stage which is beyond the current phase of the cycle. Moreover, post indexing could be applied at case loading time and / or during the similarity computations. Either approach does require advanced computations and processing power, which leads to this also occurring in a distributed manner.

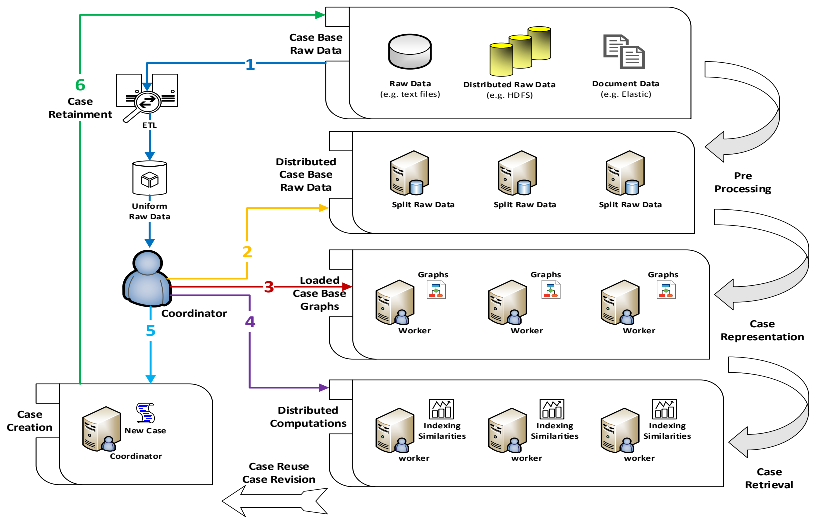

Fig. 1. Distribution Lifecycle of CBR in business process workflows

The case retrieval phase of the CBR cycle computes suitable similarity measures in order to select the most similar case. The complexity of the computed similarities varies depending on the algorithmic approach and techniques to be used in the process. Moreover, it is related to the case representation which in business process workflows is usually quite complex (graphs). In many cases, the computational complexity of graph-based similarity algorithms is reported to be an NP-hard problem requiring heuristics and only computable due to the relatively small size and complexity of the graphs [15]. As a result, the distribution of processing, can reduce performance bottlenecks within the system. Figure 1, shows an architecture based on this approach that allows the distribution of key elements of the CBR lifecycle for Business workflow monitoring systems. This approach and architecture has been adopted and implemented on a number of workflow monitoring systems in order to evaluate the approach and architecture. The experiments and results of the evaluation will be shown in the next section.

4 Evaluation

For the evaluation of this work we focused on the verification of the proposed CBR categorisation focusing on the data volume aspect. The experimental part involved two distinct phases: First, a similarity algorithm was developed, specifically designed for isomorphic and acyclic graphs [11]. Then, a number of experiments were conducted on a specific workflow monitoring domain. The experimental runs involved a large variety of case numbers ranging from: 20 to 10^6.

4.1 The CBR Domain of Business Process Workflow Monitoring

For the needs of our evaluation a business workflow monitoring system was chosen from the retail industry involving: new orders generation, order preparation, passing from the various departments and finally, the dispatching and delivery of the goods. Its business process definition can be seen in Figure 2. The domain knowledge was acquired through past working experience within the described domain throughout all departments of the workflow lifecycle. Key characteristics for this system were the strictly defined times frames between the various actions of the described workflow as presented in the BPMN.

4.2 The Links Isomorphic Graph Similarity Algorithm

The selected business process posed increased data volumes in CBR systems leading to case bases with millions of cases. The domain in question had very large numbers of workflow instances (necessary for our evaluation purposes) as well as a fine-grained business process which could be easily represented in a graph-based format. The latter was important since the complexity of the graph representation was not our investigated element but instead the evaluation of the effect from increased number of stored cases.

The business process (Figure 2) was composed by a certain number of available actions which should be present in any instance in order to have a “completed” workflow instance. Moreover, it was apparent that each action should take place at a given order which means that actions were connected with each other in a specific way. For example, it was impossible to have an “order dispatched” before the “an order generation”. As a result, workflow instances of the selected BPMN could be regarded as isomorphic [11], given the fact that the number of actions (nodes) and the way that actions were connected (links) were known, fixed and unchanged. Workflow instances of the experiments were known to be isomorphic and acyclic. In this respect, a similarity algorithm could be developed capable to measure similarities among isomorphic graphs. The idea behind the development of the proposed algorithm is that given the fact that all workflow instances have the same number of nodes and furthermore, the nodes are connected in the same way, the similarity between two given graphs could be calculated by measuring the distances of the corresponding links between those graphs.

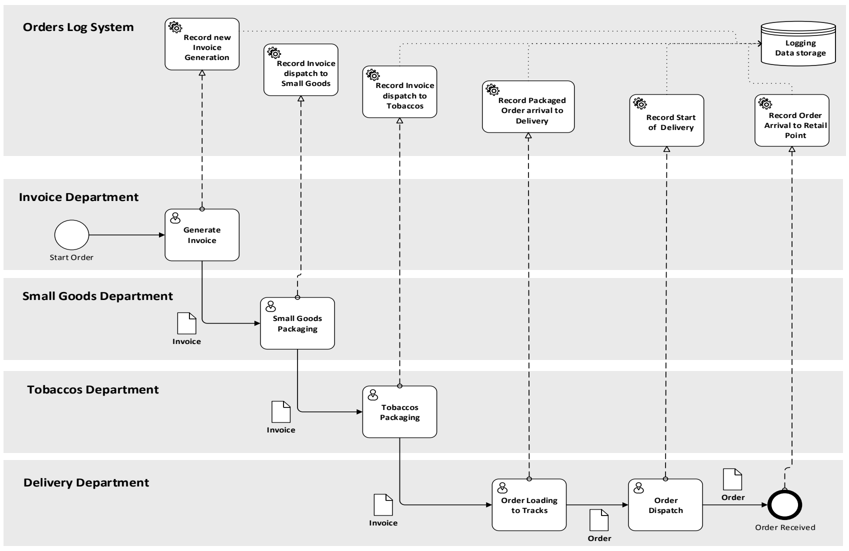

Fig. 2. Business process definition (BPMN) of the investigated business process

Based on the above, given 2 Isomorphic and Acyclic Graphs G, G′, where:

- ***G = {V, E}***, with
  - ***V(nodes)= {v_1, v_2, …, v_n}***
  - ***E(edges) = {(v_1, interval$_1$, v_2), (v_2, interval$_1$, v_3)***, …, ***(v_{n-1}, interval$_n$, v_n)}***, and
- ***G′ = {V′, E′}***, with
  - ***V′ (nodes)= {v_1', v_2', …, v_n'}***
  - ***E′ (edges) = {(v_1', interval$_1'$, v_2'), (v_2', interval$_1'$, v_3')***, …, ***(v_{n-1}', interval$_n'$, v_n')}***, andbb
- ***Count(V) = Count(V′)*** since ***G*** and ***G′*** are isomorphic, and
- ***Count(E) = Count(E′)*** since ***G*** and ***G′*** are isomorphic,

The similarity ***Sim(G, G′)*** *can be computed by,*

$$\boldsymbol{Sim}(\boldsymbol{G},\ \boldsymbol{G}') = \frac{\sum_{i=1}^{i=count(E)} \boldsymbol{\sigma}(\boldsymbol{E}_i,\ \boldsymbol{E}_i{}')}{\boldsymbol{count}(\boldsymbol{E})}$$

where $\boldsymbol{count}(\boldsymbol{E})$ is the number of edges in G graph and $\boldsymbol{\sigma}(\boldsymbol{E}_i,\ \boldsymbol{E}_i{}')$, with $0 \le \boldsymbol{\sigma}(\boldsymbol{E}_i,\ \boldsymbol{E}_i{}') \le 1$, is the similarity measure between 2 individual edges $\boldsymbol{E}_i$ ***and*** $\boldsymbol{E}_i{}'$ from graphs G and G′ correspondingly.

The implementation of the proposed algorithm was developed in such a way that the code base of the similarity computations was used in all experiments.

4.3 Experimental Design

The distributed CBR lifecycle was evaluated using 7 auto-generated datasets. Each dataset contained log entries with information related to actions occurred on workflow instances. Each completed workflow instance represented a new case for our experiments. A typical log entry comprised various comma separated values, such as: case index, event occurrence date, event name (e.g. invoice generation), personnel name, execution time and reasoning for delays if any.

The data generator application was developed in full cooperation with experts from the organization owning the business process definition having random delays in the execution of actions and the provision of random reasoning for each and every time where a delay occurrence was introduced within the workflow instance. Having completed an extensive discussion with business stakeholders, a number of guidelines were established in terms of delay occurrences. Delays were classified in 4 main categories (0, 20, 40 and 60%). Then, business experts provided reasoning for each delay category.

A number of business rules were established based on domain knowledge expertise. As an example, it was known that summer months were hectic due to increased demand and as a result, there was generally an overall increase on delays (10-30%). The afore mentioned knowledge and expertise were incorporated in the data generator application which was responsible for assigning random delays to each action along with random reasoning. Random delays were also added with an additional fluctuation factor in order to provide an additional level of realism.

The serial execution experiments used SQL Server as the medium to store the case base whereas the distributed approach used text files. A delay threshold of 50% was defined which classified any workflow instance with an execution time above the threshold as delayed.

The actual evaluation of the distributed CBR lifecycle was conducted by developing a basic implementation of a k-NN classification algorithm in order to classify any new case. The classification was delivered by a simple voting mechanism of the k most similar cases. There were 3 categories of experimental runs.

The first was the serial execution path where case loading, retrieval, adaptation and classification were performed by a conventional serial execution program written in Python with a case base stored in a MS SQL Server 2014.

The two additional execution run experiments represent a materialization of the proposed distributed CBR lifecycle where case loading, retrieval, adaptation and classification take place in a distributed manner. The first distributed implementation performs exactly the same steps with the serial execution run so as to classify the new case coming to the system, whereas the second one is an optimized version of the first by reducing the number of the data structures used though the CBR phases.

All experiments runs were conducted on the same machine with 8 cores at 2.4 GHz and 16 GB RAM (DDR3 1066 MHz).

4.4 Experimental Results

The experiments phase was composed of 21 experimental runs of the CBR cycle. The initial 7 were the serial execution approach which was also used as the base line measure. Then, there were two batches of 7 runs, one for the initial distributed approach and a second attempt being an optimised version of the first.

Table 1. Experiments - CBR Cycle Execution Times

Cases	**Serial Execution**	**1st - Approach**	**2nd - Approach**
	Execution Time (secs)	**Execution Time (secs)**	**Execution Time (secs)**
20	0.279	7.141	1.725
10^3	0.370	7.236	1.789
10^4	0.783	7.861	1.943
10^5	6.322	14.112	3.685
10^6	58.591	57.578	14.629
$5 * 10^6$	1847.182	278.475	73.432
10^7	2370.668	595.346	140.628

The serial execution ran for case bases greater than 10^6 stored cases start to increase considerably in terms of execution time (secs) [Table 1]. The increase in question is strictly related to the underlying sorting algorithm. The sorting algorithm used in all cases was timesort and its implementation varies between serial and distributed approaches. Alteration of the algorithm in question is beyond the scope of this work and it is based on Python and PySpark implementations [9, 10].

As indicated in Table1, both distributed lifecycle approaches outperform the serial execution especially for large numbers of stored cases. The first attempt begins to provide performance gains (speedup > 1) around 10^6 of stored cases whereas the second one at some point after 100000. This was due to the fact that distribution comes always with overheads related to the orchestration and setup of the distribution itself.

In terms of speedup [12], the optimised distributed approach achieved a massive improvement of 25. This number is extremely high (Figure 3). Nonetheless, we have to consider that this version of the code not only uses distribution for case loading, similarity computations, indexing and retrieval but also changes the implementation logic by reducing the used data structures utilised throughout the CBR cycle. Moreover, it is important to note that serial execution and distributed approaches harness two completely different forms of media for data store. The former used a relational database whereas the last ones utilised text files with degradation of data. The latter by itself increased performance since text storage is faster in retrieval operations, let alone in our case in which data partitioning takes place in both distributed approaches.

Finally, the speedup reduction factor for very large number of stored cases (Table 2 – 10^7 stored cases experiment). This is strictly related to the data partitioning approach used in both distributed approaches. The partitioning of the data, and as a result the processing, was performed in a fixed way based on the dataset size. This is not the most optimum scenario and further research and developed should take place on the domain of dynamic data size aware partitioning algorithms for large datasets. This issue is also adversely affected by the fact that we operated our experiments with a fixed size of computational cores.

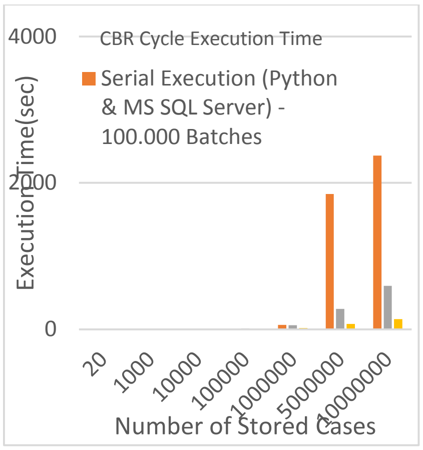

Fig. 3. Relative execution time for serial and distributed experiments

Number of Cases	1st Approach	2nd Approach
	Speedup	**Speedup**
20	0.039	0.162
1000	0.051	0.207
10000	0.010	0.403
100000	0.447	1.715
1000000	1.018	4.005
5000000	6.633	25.155
10000000	3.982	16.858

Table 2. Distributed Approaches Speedups

5 Conclusions and future work

This research has argued that the current perspective of distribution in CBR does not take into consideration the importance of data volume as a key prerequisite in the integration of distribution in CBR for business process workflows. We proposed a new categorisation of distributed CBR systems with a new dimension, this of data volume. Our research shows that gains from distribution can be found in the parts of the CBR cycle where data volume could generate deficiencies. In this respect, initial CBR stages can be massively distributed so as to enhance the performance of CBR systems. An approach and associated architecture was proposed. A number of experiments were conducted trying to evaluate the proposed distributed CBR approach and architecture. The experimental results show that distribution does increase performance of the CBR cycle reaching high speedup factors, only at large number of stored cases which is due to the fact that current technology and available frameworks allow the development of highly optimized serial executions such as ones with extensive in memory exploitation, as used in our experiments to establish baseline measures.

Further research could be on the distribution techniques and approaches to be used in the case representation and case retrieval. The current approach focuses on Data Volume. Further work could focus on the other associated challenges posed by large volumes of data. This work aims at the production of a more generic distribution model and architecture. Additionally, it is expected that more research areas and challenges will emerge in this area, including data and process distributed pipelines, dynamic data size aware partitioning algorithms for large datasets.

6 References

1. Kapetanakis, S., Petridis, M., Knight, B., Ma, J.,Bacon, L. : A CBR Approach for the Monitoring of Business Workflows, 18th Int Conf on CBR, ICCBR 2010, Alessandria, Italy, LNAI (2010)
2. Kapetanakis S., Petridis M., Ma J., Bacon L., "Providing explanations for the intelligent monitoring of business workflows using case-based reasoning", 2010, Proceedings of the Fifth International Workshop on Explanation-aware Computing, ExaCt 2010, Lisbon, Portugal
3. Teodorescu, I., Petridis, M. "An agent based framework for Multiple, Heterogeneous Case Based Reasoning" in Proceedings of ICCBR2013, Saratoga Springs, NY, Springer LNAI,2013
4. Object Management Group: BPMN Version 2.0: OMG Specifications, January, 2011, http://www.omg.org/spec/BPMN/2.0/, accessed August 2015
5. Workflow Management Coalition: Workflow Standard, Process Definition Interface, XML Process Definition Language, Version 2.2, August, 2012, http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf, accessed August 2015
6. XML Process Definition Language (XPDL): A standard of the Workflow Management Coalition (WfMC), http://www.xpdl.org/index.html, accessed August 2015
7. OASIS: Web Services Business Process Execution Language, Version 2.0, April, 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, accessed August 2015
8. Aamodt, A. and Plaza. E., "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", March 1994, Artificial Intelligence Communications, Vol. 7, no. 1, 39-52
9. Python.org, Mercurial Repositories: listsort, https://hg.python.org/cpython/file/default/Objects/listsort.txt, v. March 2016
10. Databricks: Spark the fastest open source engine for sorting a petabyte, Octomber, 2014, Reynold Xin, https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html, accessed March 2016
11. Ruohonen, K. "Graph Theory", 2008, Tampere University of Technology, Chapter 1, Section 5
12. El-Nashar, A. I."To parallelize or not to parallelize, speed up issue", 2011, International Journal of Distributed and Parallel Systems (IJDPS) Vol.2, No.2
13. Plaza. E. and McGinty L., "Distributed case-based reasoning", 2006, Knowledge Engineering Review, Vol. 20:3, 261–265
14. Minor, M., Tartakovski, A. and Bergmann, R., "Representation and structure-based similarity assessment for Agile workflows", 2007, Weber, R.O., Richter, M.M. (eds.) CBR Research and Development, Proceedings of the 7th Int Conf on CBR, ICCBR 2007, Belfast. LNAI, vol. 4626, pp. 224–238.
15. Dijkman R.M., Dumas, M., Garcia-Banuelos, L., "Graph matching algorithms for business process model similarity search", 2009, Dayal, U., Eder, J. (eds.), Proceedings of the 7th International Conference on Business Process Management. LNCS, vol. 5701, pp. 48–63. Springer, Berlin (2009)
16. Kapetanakis S., Petridis M., "Evaluating a Case-Based Reasoning Architecture for the Intelligent Monitoring of Business Workflows", 2014, Successful Case-based Reasoning Applications-2, Studies in Computational Intelligence 494, Springer-Verlag Berlin Heidelberg
17. Thain, D., Tannenbaum, T., Livny, M. "Distributed Computing in Practice: The Condor Experience", 2005, *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pp. 323-356
18. Zaharia, M., Chowdhury, M. , Franklin, M. J., Shenker, S., Stoica,I. "Spark: Cluster Computing with Working Sets", Proceedings of the 2nd USENIX conference, 2010, pp.10-10
19. Anderson, D.P., "BOINC: A System for Public-Resource Computing and Storage", 2004, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing
20. Spark: Lightning-fast cluster computing, http://spark.apache.org/, Accessed August 2015
21. M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Franklin, S. Shenker, and I. Stoica, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing", 2012, Proceedings of the NSDI, pp.2-2