Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments

Florian Helff[1] Le Gruenwald[1] Laurent d'Orazio[2]

[1] School of Computer Science
University of Oklahoma
Norman, Oklahoma, USA
{fhelff, ggruenwald}@ou.edu

[2] CNRS, UMR 6158, LIMOS
Blaise Pascal University
Clermont-Ferrand, France
laurent.dorazio@isima.fr

ABSTRACT

In a mobile-cloud database environment, different users on multiple mobile devices request services executed on a cloud. During those requests, queries are executed to obtain data, stored on the cloud and partly in caches on the mobile devices. The process of choosing an optimal query execution plan during a query optimization process is difficult because of multiple objectives involved regarding multiple non-static pricing models and different user constrains, such as monetary cost, query execution time and mobile device energy consumption. This paper provides a strategy of how to incorporate those various objectives in this decision process, based on a weighted-sum model, to achieve a good query execution plan. The experimental performance studies show that comparing with strategies, the proposed strategy is able to achieve its goal while incurs almost no additional overhead.

1. INTRODUCTION

In a mobile-cloud database environment [1], queries are issued at mobile devices to retrieve data that is stored on the cloud and optionally on the mobile devices. The process of finding an optimal query execution plan (QEP) in this environment is important in many ways. In an application scenario where many queries are executed per day, organizations try to minimize the monetary cost spent for query execution to fit their budget. They also want to minimize query execution times to meet customers' query response time requirements and to optimize employees' working time. Furthermore, users also want to minimize energy consumption on their mobile devices where queries might be executed [2]. This optimization process is a stretch of contradicting propositions, especially when considering different cloud pricing models [3].

Current decision strategies mostly focus on a single main objective, such as execution time, and order further objectives, like monetary cost and energy consumption, in a descending order. This strategy is called lexicographical ordering [4] which is not sufficient as shown in following example:

Consider the three query execution plans (QEPs) with their costs for monetary costs (M), execution time (T) and energy consumption (E) shown in Figure 1. Focusing on a single objective always leads to the decision to select either plan QEP1 or QEP2 for execution since those QEPs have a minimum cost in one of the three objectives. Since QEP3 does not have a minimum value in any of the three costs, it will never be selected although it is a competitive choice considering all three objectives on the same level of importance. Therefore, a strategy which considers all objectives at the same time is needed in order to make a comprehensive decision.

QEP1: {M= $0.080; T= 0.5s; E= 0.012 mA}

QEP2: {M= $0.050; T= 3.0s; E= 0.300 mA}

QEP3: {M= $0.055; T= 0.6s; E= 0.013 mA}

Figure 1 Execution Plan Costs Example

An existing optimization strategy which incorporates multiple objectives is called the Weighted Sum Model [5]. In this model, every possible alternative (a QEP in our application) is rated by a score including all objectives, individually weighted to stress the importance of different objectives. This model is used in many multi-objective optimization problems in various fields of computer science and also other fields such as economics (Cost-Utility Analysis) [6] [7]. However, the weakness of this model is the process of summarizing the different objectives. The fact that different objectives might have different dimensions and units leads to the problem of "adding apples and oranges" [8]. Since the mobile-cloud database environment has to deal with multi objectives with different units, the Weighted Sum Model cannot be used without major changes in its strategy. This problem is dealt with in the later explanation of our proposed algorithm in Section 4.

To fit in the context of Query Optimization, the Normalized Weighted Sum Algorithm (NWSA), which is proposed in this paper, uses the Weighted Sum Model as basis but makes major changes to cover the weaknesses of it and to fit in the mobile-cloud database environment. To cover multiple units for different objectives, the values are normalized to a user-defined maximum. This process eliminates units and results in distribution on a percentage basis. Additionally, user weights are implemented to situational stress on objectives, according to user preferences and needs. These strategies adapt the ideas of a user based decision [9].

Experiments are conducted to study the performance of NWSA. The experimental results show that NWSA is able to derive a good QEP and incurs almost no additional overhead comparing with the existing strategy that is based on the lexicographical ordering of the optimization objectives [4].

The rest of the paper is structured as follows. Section 2 gives some fundamental information about the Weighted Sum Model, underlying principles and explains the adjustments of this model. Section 3 discusses other related work. Section 4 describes the proposed strategy, the Normalized Weighted Sum Algorithm. The experimental model and results and the underlying use case are discussed in Section 5. Finally, the conclusions and future work are given in Section 6.

2. FUNDAMENTAL INFORMATION

This section describes the Pareto Set, which is fundamental for every Multi-Objective Optimization problem, and the Weighted Sum Model, that will be modified and used in the proposed strategy.

2.1 Pareto set

The Pareto set is a set of dominant alternatives, which does not include dominated alternatives. An alternative 'A' is dominating an alternative 'B' if at least one objective (decision variable) of 'A' is better than the objective in 'B' and all other objectives in 'A' are at least equal to the objectives in 'B'. Those dominating alternatives are called Pareto optimal as defined by Zitzler and Thile [10]. In the application of query optimization, every objective corresponds to a cost, for example, query execution time, monetary cost, or energy consumption cost, and an alternative is equivalent to a single QEP.

The strength of finding a Pareto set is that every alternative in this set is optimal for at least one scoring function. A scoring function describes the stress on the different objectives in order to set the importance to them and to compare alternatives in this Pareto set. [11]

In the context of query optimization, finding a Pareto set of query execution plans is not sufficient for query execution since a single execution plan needs to be selected. The process of selecting a single solution is left open for a user to choose. Regardless, the following Weighted Sum Model functions as a scoring function, using a user's preferences, and always selects a query out of this Pareto set, which is proven in Section 0.

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}$$

$$A_*^{WSM-score} = \max_i \sum_{j=1}^{n} w_j a_{ij}$$

Figure 2 Weighted Sum Model Scoring Function

2.2 The Weighted Sum Model

The Weighted Sum Model (WSM) **[5, 12]** is most commonly used in multi-objective optimization problems. It combines the different objectives and weights corresponding to those objectives to create a single score for each alternative to make them comparable. The formulas used in this model are shown in the following Figure 2.

In these formulas, the WSM-score for an alternative A_i denoted as $A_i^{WSM-score}$ is calculated by adding the products of a weight w_j with its corresponding parameter a_{ij}, the value of this objective. This parameter is, for example, the monetary cost which has to be spent to execute the query. The best alternative is chosen as the one which has the maximum WSM score ($A_*^{WSM-score}$). The different objectives are assumed to be positive: the higher the score, the better the alternative. Assuming objectives to be negative (in case of cost models), the best alternative has equivalently the lowest score.

3. RELATED WORK

The lexicographic ordering is probably the simplest but most used scoring function to solve multi-objective optimization problems [4]. This strategy compares parameters of the most important objective and selects the alternative with the highest/lowest parameter for that objective. If multiple alternatives consist of the same highest/lowest parameter, the selection process starts over with the second most important objective under those alternatives [4]. The complexity of this algorithm is in linear relation to the count of alternatives it selects its solution from since it scans the alternative once for the lowest parameter. This strategy is equivalent to the example explained in Section 1, which also shows the weaknesses of the lexicographical ordering. Multiple objectives are only considered if the selection process on a single objective is not sufficient. Although this strategy does not have to deal with multiple dimensions or units, it is not sufficient in finding an optimal solution for the proposed optimization problem since it cannot handle multiple objectives and, therefore, is not able to give a sufficient solution.

The set of all optimal solutions under every possible scoring function is called the Pareto set. Skyline queries [13, 14, 15] are one example of a strategy of finding a Pareto set. Those strategies are used in situations where no scoring function is available. As also mentioned in [9], it is important to distinguish between the Pareto optimal set, skyline queries which return the Pareto optimal alternatives, the skyline which represents the result of skyline queries and the according algorithms to implement the queries. As already discussed, because one single alternative has to be returned as the output, a strategy that finds the Pareto optimal alternatives alone is not sufficient to solve this problem. There exists work that aims to solve the problem of multi-objective query optimization in combination with Skyline queries [16] or Pareto set computations [17] [18] but all those strategies have to be concluded by a user selecting one of the solutions from the Skyline/Pareto Set. The reason of calculating a Pareto set is because of lack of an existing scoring function during execution time. The weakness of such calculation is the generated overhead of the calculation since this is an expensive computation. Given a scoring function, and making the user decide on his/her preferences prior execution to directly compute a single solution, which is an element in a Pareto set, avoids the additional overhead since computing the Pareto set is not necessary.

4. NORMALIZED WEIGHTED SUM ALGORITHM

This section describes the proposed algorithm called the Normalized Weighted Sum Algorithm (NWSA), the implementation details and the proof showing that NWSA always selects a query execution plan from the Pareto set.

4.1 Adaption to Multi-Objective Problems

As already pointed out in Section 1, one problem of the WSM is the addition of multiple dimensions or units. This problem can be resolved by normalizing the different parameters [12]. This normalization can be done in relation to a user-defined maximum of acceptance of each objective. The resulting values represent the fraction towards this maximum and do not contain a unit which makes them addable to each other. Additionally, the normalization to a user-defined maximum of parameters adapts another strategy called user-based decision [9]. Another advantage of this user-defined maximum of acceptance of each objective can be seen in the later implementation of the algorithm. Alternatives which violate those regulations can be taken out of consideration to keep the defined conditions.

The second adjustment is made for the weights. To include environmental factors, the used weight is composed of a user-defined weight and an automatically generated environmental weight. Environmental factors are, for example, the current battery status, an ongoing charging process or factors describing the currently used cloud. The environmental weight adjusts the user weight if, for example, a mobile device is being charged and energy consumption is obsolete, or a query is run overnight and execution time should be assigned a minor importance factor.

In conclusion, the Modified Weighted Sum Model Scoring Function can be expressed as in Figure 3.

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j \frac{a_{ij}}{m_j}$$

Figure 3 Modified Weighted Sum Model Scoring Function

a_{ij} is the value of alternative i (QEP_i) for objective j, m_j the user-defined acceptable maximum value for objective j, and w_j the normalized composite weight of user and environment for objective j defined in Figure 4.

$$w_j = \frac{uw_j * ew_j}{\sum(uw * ew)}.$$

Figure 4 Composite Normalized Weight Factor

Figure 4 shows the computation of the composite weight where uw_j and ew_j describe the weight of the user and the environmental weight for objective j, respectively. Since the different objectives are representative of different costs, the algorithm chooses the alternative with the lowest score to minimize costs as shown in Figure 5.

$$A_*^{WSM-score} = \min_i \sum_{j=1}^{n} w_j \frac{a_{ij}}{m_j}$$

Figure 5 Modified Weighted Sum Model Scoring Function: Optimal Alternative

In the following Section 4.2, the proposed algorithm is described. It is then followed by a proof that the chosen alternative in this decision process is always an element of the corresponding Pareto set, independent of the chosen weights.

4.2 Algorithm

The developed algorithm calculates the best alternative in a multi-objective decision process as shown in Figure 6.

Algorithm : *NWSA*
Input:
Alternatives A_i with i=1..m and parameter a_{ij} to objective O_j with j=1..n ; uw_j the user weight for objective O_j; ew_j the environment weight for objective O_j; m_j the maximum accepted value for objective O_j

Output: *best alternative A_i*

```
1.  A_best ← null
2.  A_bestRestrictionViolating ← null

3.  for i=1 to m
4.      A_i^score ← CalculateScore(A_i)
5.  end for

6.  for i=1 to m
7.      violate ← false
8.      for j=1 to n
9.          if(a_ij>m_j)
10.             violate ← true
11.             save violation
12.         end if
13.     end for
14.     if(violate=true)
15.         if(A_i^score< A_bestRestrictionViolating^score)
16.             A_bestRestrictionViolating ← A_i
17.         end if
18.     else
19.         if(A_i^score < A_best^score)
20.             A_best ← A_i
21.         end if
22.     end if
23. end for

24. if(A_best ← null)
25.     return A_bestRestrictionViolating , violations
26. else
27.     return A_best
28. end if
```

Figure 6 Decision Algorithm

The modified WSM function to calculate the score of an alternative Ai (CalculateScore(Ai)), which is used in Line 5 of this algorithm, is the previously defined function in Figure 3. This function is executed for every alternative (Lines 3-5). As it can be seen in this algorithm, each alternative is checked if it violates the user-defined maximum value for each objective (Lines 8-13). The violation itself has to be saved for future use (Line 11) Afterwards, the best alternative (A_{best}) which is the one with the lowest score is selected (Lines 14-22) and returned as the output of the algorithm. If all possible alternatives violate those restrictions, the algorithm will return the lowest score alternative ($A_{bestRestrictionViolating}$) as well as the previously saved restriction(s) that it violates (Lines 24-25). The complexity of this algorithm is in linear relation to the count of alternatives, which is also the complexity of the lexicographical ordering strategy as discussed in Section 3.

4.3 Proof for Pareto-Set

This section provides a proof that, independent from possible weights and from the number of objectives, the proposed algorithm always picks an alternative within the Pareto set.

Proof by contradiction:
It is assumed that the chosen algorithm picks an alternative A_{best} which is not an element of the Pareto set. Compliant with the used formula in the proposed algorithm (Figure 3) it can be determined that

$$A_{best}^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij} = \max_i \sum_{j=1}^{n} w_j a_{ij}$$

Since A_{best} is not an element of the Pareto set, an alternative A_{pareto} hast to exist which dominates A_{best}. According to the definition of the Pareto set, this alternative has one higher value for at least one criterion than A_{best} without having a lower value for all other objectives. This is in contradiction to

$$A_{best}^{WSM-score} = \max_i \sum_{j=1}^{n} w_j a_{ij}$$

since the score is better for A_{pareto} with unchanged weights. A similar proof in the context of skyline queries is shown in [13].

5. PERFORMANCE EVALUATION

This section describes an evaluation of the proposed strategy by means of simulation experiments. It also compares the differences of the proposed algorithm with the lexicographical ordering strategy [4].

5.1 Simulation Model

In the proposed mobile-cloud database environment, each QEP consists of three costs: monetary cost for using the cloud provider, query execution time as time to run a certain query plan, and energy used on the mobile device. The last cost becomes important under the condition of using a cache on the mobile device to have the option of receiving partial or total requested data from the mobile device itself [2]. This obviously results in a lower monetary cost since the cloud provider is less or not used, but also results in a higher amount of consumed energy since processing the cache consumes more energy than waiting for incoming data. A full review of such a system is given in [19]. Regarding that background, the simulation is built as follows:

The simulation consists of one million experiments, where the proposed NWSA as well as the lexicographical ordering strategy have to choose a single QEP out of a set of 20 QEPs. The cost of each QEP is generated randomly within the following ranges: Monetary Cost (M) has a range of 0 up to 10 cents and was chosen according to the current Amazon EC2 pricing models [3]; the range for query execution time was selected to be between 0 and 10 seconds (including data transfer time), and energy between 0 and 0.5 mAh. This simulation is repeated for multiple weight compositions.

5.2 Experimental Results

In comparison to the lexicographical ordering strategy the experimental results show two facts: First, the NWSA computes the same results under the same costs as the lexicographical ordering when focusing only on one objective. Second, NWSA produces negligible overhead in computing this selection. As it was already discussed in the previous sections 3 and 4.2, both algorithms are running linear execution time related to the size of QEPs to choose from. That leads to a total algorithm execution time of less than one millisecond per experiment for both algorithms so that the difference is negligible. Concluding this comparison, negligible overhead is incurred and no higher cost alternatives results are selected. Looking at the performance of NWSA, this evaluation shows the possibilities of this strategy.

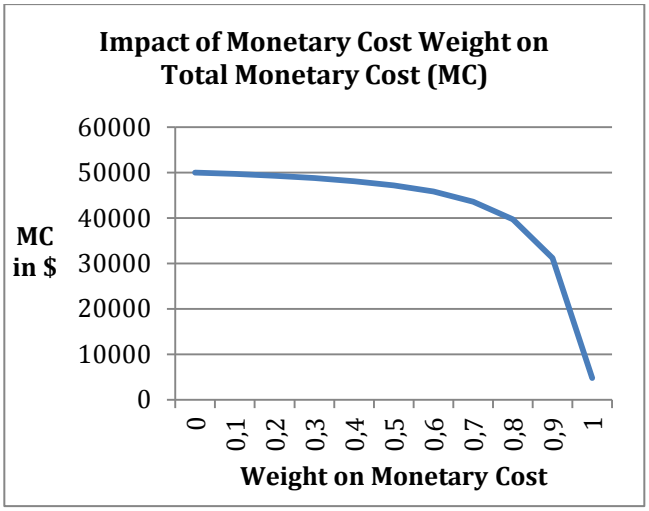

Figure 7 Impact of Monetary Cost Weight on Total Monetary Cost of QEPs selected by NWSA

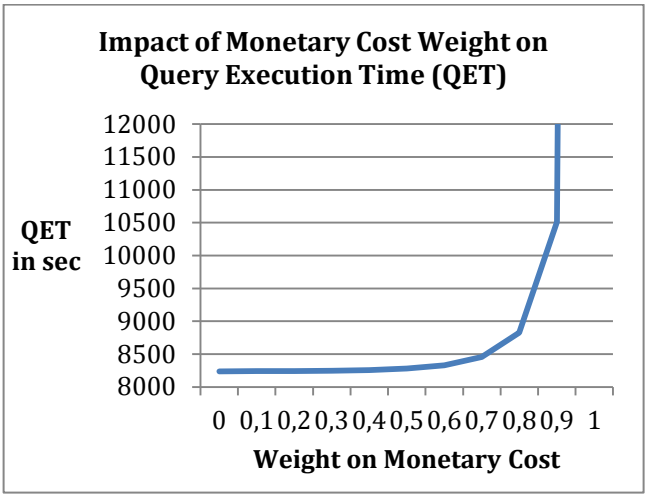

Figure 8 Impact of Monetary Cost Weight on Total Execution Time of QEPs selected by NWSA

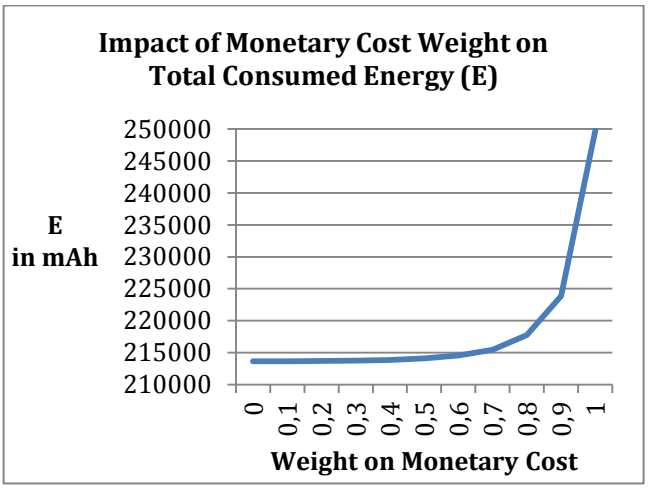

Figure 9 Impact of Monetary Cost Weight on Total Consumed Energy of QEPs selected by NWSA

To have an option of deciding how to stress the weights on the different objectives can change a lot in terms of total cost as it can be seen in Figure 7-9. The figures show the changes of the total cost of the one million chosen QEPs as the weight on monetary cost increases. The remaining weight is divided equally between execution time and energy consumption.

It can be seen that when the monetary cost weight increases, the monetary costs decreases, while the query execution time and energy consumption increase. It is notable that the minimum and maximum values of an objective span a large gap, so an impact of having weights is easily seen. Already having a small weight on one objective can lead to a big difference in the total cost.

While not shown in this paper, the graphs plotting the impacts of increasing weights on the execution time and energy consumption show an equivalent trend as the impact of increasing weights on the monetary cost.

6. CONCLUSION AND OUTLOOK

In this paper, a new algorithm, called Normalized Weighted Sum Algorithm (NWSA), was proposed, to select the best query execution plan for query optimization that includes multiple objectives, such as monetary cost, query execution time, and energy consumption, in the decision process. The simulation experiments evaluating NWSA in the context of a mobile-cloud query optimization have been presented. NWSA is able to select the query execution plan that is an element of the Pareto set, while avoiding the expensive cost of computing the Pareto set. NWSA is highly adaptable to any multi-objective decision problem since it is not limited to any number of objectives. The experimental results show that NWSA incurs negligible computational overhead in comparison to the existing lexicographical ordering strategy. Additionally, the use of weights enables a more precise selection of a query execution plan since the minimum and maximum values of an objective span a wide gap.

A future modification is to also consider non-linear functions of the normalization of objectives as well as of the composition of user and environmental weights. As far as the usage of this algorithm is concerned, we intend to incorporate it into the query optimization process to calculate fast estimations of query costs for clouds [20, 21, 22, 23]. Another future field of usage of this algorithm is a new Cache Replacement Policy for the mobile Cache to extend semantic Caching [2]. Based on the computed score of a QEP, the new policy can help to keep more valuable data in the semantic cache (the higher the score is, the higher the cost to regain those results will be).

ACKNOWLEDGEMENT

This work is partially supported by the National Science Foundation Award No. 1349285.

References

[1] H. T. Dinh, C. Lee, D. Niyato und P. Wang, „A survey of mobile cloud computing: architecture, applications, and approaches,“ in *Wireless communications and mobile computing 13.18*, 2013.

[2] M. Perrin, "Time-, Energy-, and Monetary Cost-Aware Cache Design for a Mobile Cloud Database System," May 2015.

[3] Amazon, „Amazon EC2 Pricing,“ 2015. [Online]. Available: https://aws.amazon.com/ec2/pricing/?nc1=h_ls. [Zugriff am 18 08 2015].

[4] A. Ben-Tal, Characterization of Pareto and lexicographic optimal solutions, Springer Berlin Heidelberg, 1980.

[5] E. Triantaphyllou, Multi-criteria decision making methods: a comparative study, 44 ed., Springer Science & Business Media, 2013.

[6] P. Kind, J. E. Lafata, K. Matuszewsk und D. Raisch, „The use of QALYs in clinical and patient decision-making: Issues and prospects,“ in *VALUE IN HEALTH 12*, 2012.

[7] M. G. M. Hunink und M. C. Weinstein, Decision Making in Health and Medicine: Integrating Evidence and Values, 2nd Hrsg., Cambridge University Press, 2014.

[8] P. C. Fishburn, "Methods of estimating additive utilities.: Management science 13.7," 1967.

[9] C.-H. Goh, Y.-C. A. Tung and C.-H. Cheng, " A revised weighted sum decision model for robot selection," in *Computers & Industrial Engineering Vol.30(2)*, 1996.

[10] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms:," in *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 3, NO. 4,* , NOVEMBER 1999.

[11] D. V. Lindley, Scoring Rules and the Inevitability of Probability, 50 ed., International Statistical Review Vol 50, 1982, pp. 1-11.

[12] I. Kim and O. de Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," in *Structural and multidisciplinary optimization 29.2* , 2005, pp. 149-158.

[13] J. Chomicki, P. Ciaccia and N. Meneghetti, "Skyline queries, front and back," in *ACM SIGMOD Record 42.3*, 2013.

[14] D. Kossmann, F. Ramsak and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proceedings of the 28th international conference on Very Large Data Bases (VLDB)*, 2002.

[15] D. Papadias, Y. Tao, G. Fu and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data (ACM)*, 2003.

[16] C. H. Papadimitriou und M. Yannakakis, „Multiob jective Query Optimization," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2001.

[17] I. Trummer und C. Koch, „Multi-Objective Parametric Query Optimization," in *Proceedings of the VLDB Endowment*, 2014.

[18] C. Lei, Z. Zhuang, E. A. Rundensteiner und M. Eltabakh, „Shared Execution of Recurring Workloads in MapReduce∗," in *Proceedings of the VLDB Endowment*, 2015.

[19] J. Mullen, M. Perrin, F. Helff, L. Gruenwald and L. d'Orazio, "A Vision of Time-, Energy-, and Monetary Cost-Aware Query Processing in Mobile Cloud Database Environment," March 2015.

[20] N. Bruno, S. Jain and J. Zhou, "Continuous cloud-scale query optimization and processing," in *Proceedings of the VLDB Endowment 6.11* , 2013.

[21] H. Kllapi, E. Sitaridi und M. M. Tsangaris, „Schedule Optimization for Data Processing Flows on the Cloud," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (ACM)*, 2011.

[22] M. Armbrust and e. al., "A view of cloud computing," in *Communications of the ACM 53.4*, 2010.

[23] H. M. Fard, R. Prodan, J. J. D. Barrionuevo and T. Fahringer, "A Multi-Objective Approach for Workflow Scheduling in Heterogeneous Environments," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2012.