Migration-aware Optimization of Virtualized Computational Resources Allocation in Complex Systems

Piotr Rygielski and Adam Gonczarek
Institute of Computer Science
Wrocław University of Technology
ul. Janiszewskiego 11–17, 50-370 Wrocław, Poland

***Abstract*—In this paper a task of resources allocation in the complex system is considered. Novelty of the formulated task consists of assumption that the applications assigned with resources of one machine can be migrated to another machine during system lifetime. The formulated task has been solved using proposed heuristic optimization method. Due to non-convex set of valid solutions the optimization procedure has been decomposed into two stages and forms approach similar to the relax-and-round approach. Proposed decomposition approach facilitates fast algorithm convergence and guaranties that achieved solution satisfies assumed constraints.**

I. Introduction

In this paper we consider resource allocation task in a distributed execution system. We assume that the system as a whole is a complex system so the services offered to a customer are in general complex services. The complex services — also called in the literature composite service — composed using many atomic services [8] in such a way that user's functional and quality requirements are met. An atomic service is a set of applications which deliver exactly the same atomic functionality but can be located in another location of distributed complex system so the quality may vary.

We formulate a specific optimization task which includes the migration of application from one virtualized computational node to another one [5], [16]. Such an operation is possible due to resources virtualization — proper research in applications migration scope has been investigated in the literature, e.g. [3], [15], [2]. The goal of our research is to optimize the quality of service delivered to the user by means of efficient computational resources allocation taking into consideration the migration possibility.

The motivation for such a problem formulation is non-stationarity of users' behaviour that leads to variable intensities of requests streams incoming to the system. The system needs to react to changes in environment by automatic optimization of the resources allocation in order to keep the quality of service at required level. The migration awareness of the service-oriented system gives the system ability to relocate the system's services if only such an operation leads to improvement the of quality of service.

II. System Model and Task Formulation

The considered execution system is assumed to be complex — contains many distinguishable parts called atomic services — and distributed — system's components are located on various computational nodes connected with network. The system model is presented in the subsection below.

A. Complex System Model

The considered system offers complex services which can be composed dynamically satisfying all the requirements formulated by user. The user formulates its requirement by specifying its functional and nonfunctional requirements bound in a contract. Denote system components — called formerly atomic services — as as_j. Each atomic service can be available in the system in many versions as_{jk}, where each version delivers the same functionality but differs in nonfunctional parameters values. The atomic service version is an software application located in proper computational node of the system.

The atomic service versions are located on the computational nodes within the execution system which is modeled as a graph. The execution system graph is denoted $G = (V, E)$ where V denotes a set of computational nodes and E is a set of communication channels linking the computational nodes. The quantity of computational resources is modeled as a vector $\mathbf{u} = [u_1, u_2, ..., u_N]^T$, $N = |V|$, where each element of $\mathbf{u}$ vector represents computational abilities of respective node — total amount of computational resources. The value of $\mathbf{u}$ is constant over time. The assignment of computational resources to the atomic services versions in t-th moment of time is done with matrix $\mathbf{P}(t)$ defined in the following way:

$$\mathbf{P}(t) := (p_{i,j})_{i=1,...,M;j=1,...,N} \tag{1}$$

where N denotes the number of computational nodes present in the system $N = |V|$, and M denotes the total versions number of all atomic services (applications) $M = \prod_j^J K_j$, where K_j denotes the number of versions of j-th atomic service, and J is the number of all atomic services. The matrix defines how much amount of computational resources of each node is assigned to which application. The value $0 \leq p_{i,j} \leq 1$ means that i-th atomic service version has been assigned with amount of $p_{i,j} \cdot u_j$ resources of j-th computational node. In

example $p_{i,j} = 0.5$ means that half of resources of j-th node is assigned to i-th application.

B. Complex Service Resources Consumption

The whole composition process of composing complex service required by the user has been described in [6], [9] and some QoS-aware composition optimization has been investigated in [8], [18]. In this paper we focus on resources consumption modeling, which is mainly the effect of the composition process.

Assume, that the required complex service is composed e.g. with the graph-fold algorithm [10] in order to satisfy user's functional and nonfunctional requirements. Moreover, we assume that all users of the system have been grouped in order to distinguish a user class (e.g.: bronze, silver, gold, platinum) which depends on e.g. how much the user pays for the service. It is obvious, that the system should offer services of higher quality to users who pay more. In order to model the atomic services versions usage we propose a matrix defined in the following way:

$$\mathbf{R}(t) := (r_{i,j})_{i=1,\ldots,M;j=1,\ldots,L}. \tag{2}$$

The R matrix has dimensions $M \times L$, where L denotes the number of user classes, and the $r_{i,j}$ value means that the i-th atomic service version is used by the users from j-th class $r_{i,j}$ times in the t-th period of time. The user classes have weights assigned in order to valuate importance of services chosen by them. The weights are represented by the vector denoted $\mathbf{v} = [v_1, v_2, ..., v_L]^T$. Each atomic service version used by the user from l-th class is v_l times more important than in the situation when all users are treated equal. We obtain weighted atomic services versions usage vector $\mathbf{v}_D$ by multiplying the $\mathbf{R}$ matrix by $\mathbf{v}$ vector:

$$\mathbf{v}_D(t) = \mathbf{R}(t)\mathbf{v} \tag{3}$$

C. Problem Formulation

The task considered in this paper focuses on finding such an allocation of computational resources, that utility function value is maximized. Higher utility value means higher quality of service. The constraints have been formulated in such a way that the migration operation is possible — each atomic service version can be moved from one machine to another. Moreover, formulated task's solution points out, that some atomic services versions should be disabled (resources will not be assigned).

In order to find optimal computational resources allocation, such that the quality of service is maximized, the following task should be solved:

Given:

- Current resource assignment matrix $\mathbf{P}(t)$
- Current weighted atomic services versions usage vector $\mathbf{v}_D(t)$
- Amount of computational resources vector $\mathbf{u}$
- Marginal utility function $f_u = \sum_{j=1}^{N} \left(1 - e^{-\alpha \cdot \mathbf{u}^T \mathbf{P}_j}\right)$

Find: such a resources assignment matrix $\mathbf{P}(t+1)$ which maximizes the following utility function (eq. 4):

$$U(\mathbf{P}(t+1)) = f_u(\mathbf{u}^T \mathbf{P}(t+1))\, \mathbf{v}_D(t), \tag{4}$$

with subject to the following constraints:

$$\underset{1\le i\le M}{\forall} \underset{1\le j\le N}{\forall} : p_{i,j} \ge 0, \tag{5}$$

$$\underset{1\le i\le M}{\forall} : \sum_{j=1}^{N} p_{i,j} \le 1, \tag{6}$$

$$\underset{1\le j\le N}{\forall} : \sum_{i=1}^{M} p_{i,j} = \max_i\{p_{i,j}\} \tag{7}$$

In the equation 4, the $f_u(\mathbf{u}^T\mathbf{P}(t+1))$ function has been used. This function models atomic service version performance with respect to assigned amount of computational resources. In general, a linear function can be used but more realistic would be for example the function given with equation 8:

$$f_u(u^T\mathbf{P}(t+1)) = \sum_{j=1}^{N} f_{u,j}, \tag{8}$$

$$f_{u,j} = 1 - e^{-\alpha \cdot \mathbf{u}^T \mathbf{P}_j}, \tag{9}$$

where $\mathbf{P}_j$ denotes j-th column in matrix $\mathbf{P}(t+1)$. Function 9 involves the law of diminishing marginal utility [12]. The α parameter models the shape of the function f_u and in case of this problem formulation, α should be chosen as $0 \le \alpha \le 1$.

The formulated constraints should be interpreted as follows. Each resource assignment ratio should be no less than 0 and no more that maximum available resource quantity (eq. 5); All assigned resources of proper computational node should be no less than 0 and no more than maximum available resource quantity (eq. 6); Single atomic service version should be assigned with resources of at most one computational node (eq. 7). The last constraint given with equation 7 can be understood also as a demand of at most one element with value > 0 in the each column vector of matrix $\mathbf{P}(t)$.

Such a formulated task is an optimization problem of nonlinear, monotonically increasing utility function with nonlinear constraint given as an equality and linear constraints given as inequalities.

Noteworthy is the fact, that constraint given with equality 7 reduces the valid solution set only to segments (within the range $\langle 0, 1\rangle$) placed on axes of $(M \cdot N)$-dimensional space. Such a solution set is non-convex, so there is no guarantee to achieve global optimum using standard optimization techniques. In the section III we propose a two-stage method of solving this problem.

D. Related Works

The problem formulated in our work is a transformation of resources allocation problem [17]. We have transformed the constraints set in such a way that migration operation is possible. The resource allocation algorithms has been investigated widely in the literature, e.g. [11], [13], [14], [17] but the modification allowing migration of services has not been found in the literature. The original resource allocation problem has been solved using classic approaches using convex optimization, dynamic programming (centralized global decision making) and using such methods like e.g. auction mechanisms or multiagent systems (distributed and local decision making). To valuate the quality of service the utility function has been used in most related works.

We would like to emphasise that the problem stated in section II-C is the modification of distributed resource allocation problem. The modification consists of introducing the migration operation what causes that the original problem is non-convex. The migration allows to change the application physical location within the system with virtualized computational resources. The consequence of migration is a modified set of constraints in the task formulation, so the classic methods can behave in the different way.

In order to migrate atomic service versions from one physical node to another one, we need to locate the application on virtual machine. As the virtualization platform Xen is assumed [1]. We have chosen Xen because it allows to migrate virtual machines in distributed complex systems without the need of central storage drive what is not possible in e.g. Vmware. Decentralized storage has the advantage that there is no problem with bottlenecks and virtual machines can be migrated directly from the source to destination physical node.

III. Proposed Solution

We propose a two-stage, heuristic method of optimization in order to solve the stated task. Due to the non-convex set of valid solutions we approximate this set by a convex one in the first stage. Solving the optimization problem for such a approximated constraint results in obtaining the proposition of execution location for each atomic service version, but does not assign the resources. In general, obtained solution is not valid by means of constraints given by equations 6 and 7. In the second stage, we propose to make the projection of solution obtained in the first stage onto the feasible solution space of the original problem. The projection causes the method to be heuristic. Each set of feasible solutions in the second stage is convex, so known numerical optimization methods can be used to solve this problem, e.g. interior-point method [4].

A. First Stage Optimization

In the first step we change the constraint given by equation 7 to the following one:

$$\underset{1\leq j\leq N}{\forall} : \sum_{i=1}^{M} p_{i,j} \leq 1. \tag{10}$$

Noteworthy is the fact that the new set of constraints 5, 6, 10 is convex. Furthermore this set is the smallest convex set bounding the set of constraints given by equations 5, 6, and 7. Due to the fact that the function given with eq. 4 is concave, we get convex optimization problem, which has a unique solution. In order to solve the new convex optimization problem we use the numerical optimization method (e.g. interior point method). The solution obtained in this stage is denoted by $\mathbf{P}^{+}(t+1)$.

B. Second Stage Optimization

In the second stage of proposed optimization procedure we project $\mathbf{P}^{+}(t+1)$ onto the set of constraints (eq. 7). The projection $\mathbf{P}^{p}(t+1)$ has the following form:

$$\underset{1\leq j\leq N}{\forall} : p^{p}_{i,j} = \begin{cases} p^{+}_{i,j} & p^{+}_{i,j} = \max\limits_{k}\{p^{+}_{k,j}\} \\ 0 & p^{+}_{i,j} \neq \max\limits_{k}\{p^{+}_{k,j}\} \end{cases} \tag{11}$$

The idea of the projection is to keep only the largest values in every column j and the rest set to zero. Alternative to that can be choosing the values for which we have obtained largest Lagrange multiplier in the first optimization step. Choosing the proper projection is crucial in this problem because it makes the solution heuristic.

Next we formulate the following set of constraints:

$$\underset{\substack{1\leq j\leq N \\ p^{+}_{i,j}\neq \max\limits_{k}\{p^{+}_{k,j}\}}}{\forall} : p_{i,j} = 0. \tag{12}$$

Now we want to maximize the function (eq. 4) with subject to constraints given by equations 5, 6 and 11. The new optimization problem is also convex and has a unique solution.

Once again we can easily obtain the optimal solution using interior-point method. Let $\mathbf{P}^{*}(t+1)$ denotes the final solution. Obviously, this solution satisfies the constraints given by equations 5, 6 and 7. However, it is not necessarily the optimal solution of the optimization task stated in section II-C. Nevertheless, empirical study shows that the presented optimization method gives very good results as shown in the next section.

IV. Results Discussion

The two-stage heuristic optimization method proposed in section III has been implemented and evaluated using MATLAB. To evaluate the optimization quality of the proposed method, we have set $\alpha = 0.1$ and randomized the $\mathbf{u}$ and $\mathbf{v}_D(t)$ vectors values every experiment. The experiment consisted of generating 1 million of random valid solutions and comparing the best random solution with the one calculated by our method. The starting point for the optimization method has been chosen randomly every run. The relative (divided by maximum) difference between the best randomized value and the value obtained by our method is depicted in fig. 1.

Considering the large size problem (50 nodes and 80 services) our method returns better results than random search maximum. For the medium size problem (10 nodes and 15

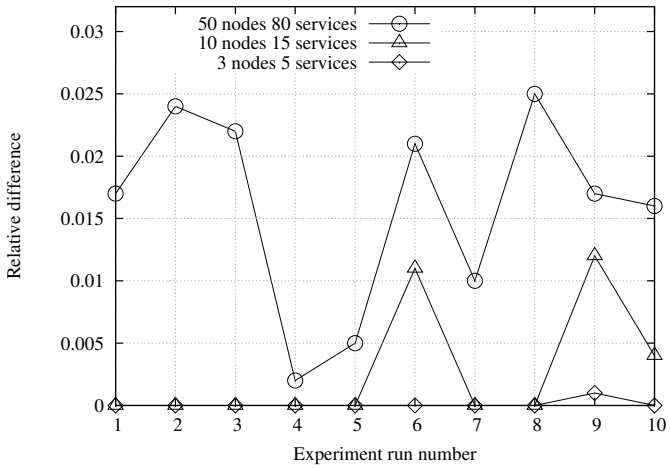

Fig. 1. Relative difference of utility function value obtained in random search and in optimization using the proposed optimization method. The solution is verified for large (50 nodes, 80 services), medium (10 nodes, 15 services) and small (3 nodes, 5 services) problem sizes.

services) solution is very close to optimal except the two experiment runs when our method gave better results. For the small size problem (3 nodes and 5 services) solution returned by our method is almost alike to random search maximum (but obtained in much shorter time).

Obtained results are very interesting. The disproportion of obtained results for our method and random search are caused by variable percentage numerical optimization of all possible solutions size of set explored by random search. This observations let us suspect that the proposed optimization method returns the solution that is very close to the optimum.

Due to heuristic nature of the proposed method we compared our results to the results obtained with exhaustive search method. We have implemented the exhaustive search method in MATLAB and set the matrix value minimum step to 0.01 for each variable. That corresponds to the 1% of physical machine resources. This means that we have examined 100 values for each value in matrix $\mathbf{P}(t+1)$. Due to very large possible solution space — in this case $(100N)^M$ — we have picked $N = 2$ machines and $M = 5$ services for the testing purposes. The solution space size in this example was about $32 \cdot 10^{10}$.

In the experiment we used the following values:

- $\mathbf{u} = [2944, 3217]^T$
- $\mathbf{v}_D(t) = [105.211, 315.737, 315.737, 411.937, 411.937]^T$
- $\alpha = 0.1$

The values of the $\mathbf{v}_D$ and $\mathbf{u}$ was taken from the simulation environment in the random moment of simulation. The simulator was used to model the distributed complex system and to monitor the intensities of the requests streams [7]. However, we do not focus on the simulation environment in this paper, we just use is as a source of data.

In the exhaustive search we have obtained the following results. The $\mathbf{P}^*_{exhaust}(t)$ matrix as in the equation 13:

$$\mathbf{P}^*_{exhaust}(t+1) = \begin{bmatrix} 0 & 0.425 & 0 & 0.575 & 0 \\ 0.12 & 0 & 0.33 & 0 & 0.55 \end{bmatrix}. \tag{13}$$

The utility value for such a matrix is presented in equation 14:

$$U\big(\mathbf{P}^*_{exhaust}(t+1)\big) = 3205.4. \tag{14}$$

Moreover, our method was examined on the same data. We run the proposed optimization method 10 times and collected the proper data. The results were the following:

$$U_{avg}\big(\mathbf{P}^*_{opt}(t+1)\big) = 3198.9, \tag{15}$$

$$U_{max}\big(\mathbf{P}^*_{opt}(t+1)\big) = 3206.2, \tag{16}$$

$$U_{min}\big(\mathbf{P}^*_{opt}(t+1)\big) = 3177.2. \tag{17}$$

For the 10 runs of our method we have noticed that the standard deviation of the utility function value was equal to 9.6137. Noteworthy is the fact, that maximum utility function value obtained for our algorithm is higher than the one obtained in exhaustive search. This is the result of limiting the resolution in the exhaustive search to 0.01 what makes the domain of the function discrete. The $\mathbf{P}^*_{opt}(t+1)$ obtained for the best utility function value (eq. 16) is presented in equation 18.

$$\mathbf{P}^*_{opt}(t+1) = \begin{bmatrix} 0 & 0.4267 & 0 & 0 & 0.5733 \\ 0.1118 & 0 & 0.379 & 0.5093 & 0 \end{bmatrix}. \tag{18}$$

Obtained results show that the solution proposed by our method is near to optimal value. The exhaustive search experiment results estimated the optimal value of the utility function quite precise. However, the matrix $\mathbf{P}^*_{opt}(t+1)$ is slightly different than $\mathbf{P}^*_{exhaust}(t+1)$ — services 4 and 5 has been switched and service 3 obtained 5% more resource than in exhaustive search. This is caused by the fact, that services 4 and 5 were used with same intensity in this example (see $\mathbf{v}_D$ vector values) so replacement between machines would not decrease the utility function value. Moreover, the fact that service 3 obtained more resources is caused by the shape of the marginal utility function (eq. 9) which is flattening with increasing resources assignment ratio. This leads to the observation, that there are a lot of suboptimal solutions for this task and the effort to find the optimal one may be much higher than the difference in quality of service of resulting allocation.

Unfortunately we were unable to compare our method to the related works because exactly such an problem formulation was not examined before.

V. Conclusions and Further Work

The disproportion of obtained results for our method and random search are caused by keeping fixed sample for increasing solution space size in random search approach. This observations let us suspect that the proposed optimization

method returns the solutions that are very close to the optimum. Moreover, this conclusion has been confirmed for small sized problem by comparison to the exhaustive search results.

The formulated task along with the proposed solution allows the system to self-optimize by means of computational resources allocation with migration operation in order to maximize the offered services quality. However, in the real complex system (e.g. based on the service-oriented architecture paradigm) the communication should be taken into account in order to optimize the quality properly. This can be obtained assuming that the communication between system's components is also modeled as a service. The resources of communication service can be also managed in the very similar way — share the total channel capacity between the data streams that are using the channel.

We emphasize that the proposed method is a heuristic so the quality of optimization may decrease with increasing number of computational nodes and services number. Moreover, the migration operation consumes quite a lot of communication resources, so the number of migrations should be taken into account during the optimization process in order to minimize the network's load. Noteworthy is the fact, that the quality profit obtained after the migration operation increases in time during the system work but the migration itself is done only once — it can be understood as an single investment for the the future profits. This involves that the optimization time step length should be investigated in order to maximize the quality along with the number of migration operations minimization what we plan to focus on in the near future.

ACKNOWLEDGMENT

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program no. POIG.01.03.01-00-008/08.

Fellowship co-financed by European Union within European Social Fund.

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield: *Xen and the art of virtualization*, Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA

[2] L. Cherkasova, D. Gupta, and A. Vahdat. *When virtual is harder than real: Resource allocation challenges in virtual machine based IT environments*. Technical Report HPL-2007-25, HP Laboratories Palo Alto, Feb. 2007.

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, *Live migration of virtual machines*, In Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, Vol. 2, 273–286, (2005).

[4] A. Forsgren, P. E. Gill, M. H. Wright, *Interior Methods for Nonlinear Optimization*, SIAM Rev. 44, 525–597, (2002).

[5] H. Jin; L. Deng; S. Wu; X. Shi; X. Pan, *Live virtual machine migration with adaptive, memory compression*. CLUSTER '09. IEEE International Conference on Cluster Computing and Workshops pp.1–10, (2009).

[6] A. Grzech, P. Rygielski, *Translations of Service Level Agreement in Systems Based on Service Oriented Architecture*, KES 2010, LNAI vol. 6277, 523–532, (2010).

[7] A. Grzech, and P. Swiatek, *Parallel processing of connection streams in nodes of packet-switched computer communication systems*, Cybernetics and Systems, 39(2), pp. 155–170, 2008.

[8] A. Grzech, and P. Swiatek, *Modeling and optimization of complex services in service-based systems*, Cybernetics and Systems, 40(08), pp. 706–723, 2009.

[9] A. Grzech, P. Swiatek, and P. Rygielski, *Translations of Service Level Agreement in Systems Based on Service Oriented Architectures*, Cybernetics and Systems, 41, pp. 1–18, 2010.

[10] P. Rygielski, and P. Swiatek, *Graph-fold: an efficient method for complex service execution plan optimization*, Systems Science vol. 36, no.3, pp. 25–32, 2010.

[11] T. Ibaraki, and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, 1988.

[12] J. H. McCulloch, and J. Huston, *The Austrian Theory of the Marginal Use and of Ordinal Marginal Utility*. Zeitschrift fur Nationaloekonomie, Vol. 37, No. 3/4,249–280, (1977).

[13] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, *A resource allocation model for QoS management*, Proceedings of the 18th IEEE Real-Time Systems Symposium, p.298, 1997

[14] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, *Practical Solutions for QoS-Based Resource Allocation*, Proceedings of the IEEE Real-Time Systems Symposium, p.296, 1998

[15] M. Rosenblum, and T. Garfinkel, *Virtual Machine Monitors: Current Technology and Future Trends*. Computer 38, 5, 39–47, (2005).

[16] F. Travostino, et.al, *Seamless live migration of virtual machines over the MAN/WAN*. Future Gener. Comput. Syst. 22, 8, 901—907, (2006).

[17] W. Teresa , N. Ye, and Z. Dawei, *Comparison of distributed methods for resource allocation*, International Journal of Production Research, Vol. 43, No. 3, 1 , 515–536, 2005.

[18] T. Yu, Y. Zhang, and K. J. Lin, *Efficient algorithms for Web services selection with end-to-end QoS constraints*. ACM Trans. Web 1, 1, Article 6, 2007.