A Heuristic for Optimum Allocation of Real-Time Service Workflows

Tommaso Cucinotta and Gaetano F. Anastasi
Real-Time Systems Laboratory
Scuola Superiore Sant'Anna
Pisa, Italy
Email: {t.cucinotta,g.anastasi}@sssup.it

***Abstract*—In this paper, the problem of optimum allocation of real-time service workflows over a set of heterogeneous resources is tackled. In previous works, this problem was formally stated in terms of a Mixed-Integer Non-Linear Programming optimization program, that could be solved by recurring to commercial solvers. However, due to the big dimension of the solution space to be searched, finding the absolutely optimum solution: might take too much time in order to be concretely useful; it may preclude the use of these techniques in large-scale infrastructures; it makes the technique hardly usable adaptively in response to corrective actions that may be needed when some bad event occurs while the services are running (e.g., hardware-level failures). Therefore, in this paper a heuristic algorithm based on graph-matching is introduced that may find very efficiently a reasonably good, albeit non-necessarily optimum, solution. The algorithm is described, and its performance assessed by a set of synthetic experiments.**

***Keywords*-real-time; workflow; resource allocation; heuristic**

I. Introduction

Nowadays, more and more applications are being developed and deployed according to a distributed computing paradigm. Interactive and multimedia applications are examples of real-time services that can be conveniently virtualized [1] but they possess strict timing requirements in terms of the maximum end-to-end latency that can be tolerated by the users. In this context, it is useful to address the problem of how to deploy distributed service workflows with end-to-end deadline constraints over a network of heterogeneous computing resources, as they may be available within a provider domain. In the decision process, multiple factors may have to be accounted for, including the relative importance among the workflows under admission, their associated revenues for the provider, as well as their associated computation, networking and storage requirements.

This problem may be formally stated in terms of an optimization program [2], maximizing a given objective-function depending on the provider business policy, under a set of constraints due to the availability of physical resources within the domain, and to the maximum tolerable application latencies. However, due to the big dimension of the solution space to be searched, finding the absolutely optimum solution may not be convenient.

Therefore, in this paper this problem is tackled by designing a proper heuristic based on a graph-matching algorithm. This takes into account the information on the availability of resources, the workload requirements of the workflows as well as their timing constraints. The ability of the proposed technique to meet end-to-end real-time constraints of the workflow relies on a clear model of computation for the workflows to be deployed, and on an algorithm that compensates possible shortages in the resource allocation for one service by increasing the allocation on the ones that follow. For example, this allows for trading computing and networking allocations for meeting a given end-to-end deadline. At the same time, the algorithm has some flexibility in that it allows a provider to maximize different business-related objective functions during the allocation process. The obtained execution times for the proposed heuristic are smaller of orders of magnitude, compared to the time needed for finding a theoretically optimum solution.

II. Related Work

The problem of allocating physical resources to workflow applications has been tackled many times in the past, especially in the Grid and service-oriented communities. For example, Mika et al. [3] propose an application model that resembles the one used in this paper. However, end-to-end real-time constraints are not addressed in that work.

In the real-time community, many authors designed architectures for QoS-aware resource allocation and scheduling for distributed applications [4]. However, in this kind of approaches, services are already assigned to hosts, whilst this allocation/mapping is one of the variables of our problem.

In the Cloud Computing, services are encapsulated in Virtual Machines (VMs) and thus scheduling a service may comprise finding enough resources for a VM to be deployed. As an example, the work by Wang et al. [5] addresses the problem of scheduling parallel tasks within a SOA by taking into account multiple resources required by the corresponding VMs. Our work considers multiple resources as well. We assume the presence of underlying scheduling mechanisms able to offer proper guarantees to individual services or VMs [6], [7].

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7 under grant agreement n.248465 S(o)OS Service-oriented Operating Systems.

III. Model of Computation and Notation

In this section, the main notational elements used throughout the paper are introduced. The physical resources constituting the provider infrastructure are defined as follows:

- A set of computing nodes: $\mathcal{H} = \{1, \ldots, N_H\}$. Each host $h \in \mathcal{H}$ is characterized by a computing capacity U_h, expressed in terms of availability of processor(s) share, and a memory capacity Ω_h (in bytes[1]).
- A set of available subnets: $\mathcal{S} = \{1, \ldots, S\}$. Each subnet is characterized by a maximum aggregate bandwidth B_s (in bytes/s), and a latency L_s, that depend on the adopted type of medium, packet scheduling algorithm and protocol for QoS assurance.
- The network topology information, specifying what hosts $\mathcal{H}_s \subset \mathcal{H}$ are connected to each subnet $s \in \mathcal{S}$.

In this work we focus on the problem of admitting a single real-time application workflow $\mathcal{A}$ into the infrastructure. The application is a linear workflow of n services $\mathcal{A} \triangleq \{1, \ldots, n\}$, denoted also as $(\tau_1, \ldots, \tau_n)$. Each service performs some CPU-intensive computation, then transmits some data to the next service in the workflow, which in turn starts its own computations, and so on. Activation requests to the workflow arrive with a minimum T inter-arrival time.

The following elements denote computing, memory and networking requirements, as well as timing constraints, for the considered set of applications:

- Computation time $c_{i,j}$ exhibited by each service τ_i of application $\mathcal{A}$, if deployed on physical node $j \in \mathcal{H}$.
- Amount of (volatile or virtual) memory ω_i (in bytes) needed by τ_i on the node where it will be deployed.
- Number m_i of bytes to be transmitted by each service $\tau_i \in \mathcal{A}$, to τ_{i+1}, each time τ_i completes an activation, with $i \in \{1, \ldots, n-1\}$.
- Response-time ρ_i of each service τ_i of $\mathcal{A}$.
- End-to-end response-time ρ of the whole application.

A. Scheduling

Each service of the application workflow is assumed to be implemented in form of a process being activated for each workflow item that needs to traverse the whole chain of computations. Also, it is assumed that each host is capable of scheduling the CPU(s) by an algorithm achieving *temporal isolation* among concurrently running services, and allowing for the allocation of processor shares to the individual tasks. Furthermore, each host must allow for a simple utilization-based admission control test, i.e., ensuring that the allocation of all the services deployed on the same processor (or processor group) does not exceed a maximum capacity U_j.

Generally speaking, when using *resource-reservation* [8] schedulers, an allocation is specified for each service in terms of a budget of q_i time units guaranteed every period of d_i time units. It can be shown [9] that the time needed by the service to complete is bounded by: $\rho_i \leq \left\lceil \frac{c_{i,j}}{q_i} \right\rceil d_i$. However, if the budget is sufficient to sustain the worst-case execution time (as assumed from here on), then such value simply reduces to d_i.

It is also assumed that subnets exhibit proper packet scheduling capabilities, so that a precise bandwidth b_i can be assigned to each data flow needed by τ_i for transmitting its result (m_i bytes) to τ_{i+1}, ensuring the temporal isolation among multiple data flows. For example, weighted fair queueing algorithms [10] would meet such a requirement. Note that, as a corner case, a subnet may also represent a point-to-point link, or the local "loopback" connection.

[1]This can be volatile or virtual memory. Note that the model can easily be extended to consider both of them.

IV. Formalization of the problem

Using the definitions and notation introduced in Section III, the problem under study may now be formalized. Let $\{x_{i,j} : i \in \mathcal{A},\ j \in \mathcal{H}\}$ represent unknown boolean variables representing allocations of services to hosts: $\forall i \in \{1, \ldots n\}$, $\forall j \in \mathcal{H}$, $x_{i,j} = 1$ if τ_i is deployed on host j and 0 otherwise. Note that, $\forall i \in \mathcal{A}$, $\sum_{j \in \mathcal{H}} x_{i,j} = 1$. Also, let $\{y_{i,s} : i \in \mathcal{A} \backslash \{n\},\ s \in \mathcal{S}\}$ represent the derivative unknown boolean variables (introduced for clarity) representing allocations of services to subnets: $\forall s \in \mathcal{S}$, $y_{i,s} = 1$ if τ_i is deployed on some node $j \in \mathcal{H}_s$. The relationship between the two sets of variables is given by: $\forall i \in \mathcal{A}$, $\forall s \in \mathcal{S}$, $y_{i,s} = \sum_{j \in \mathcal{H}_s} x_{i,j}$. These variables need to be flanked by other ones quantifying the resource allocations performed on each one of the hosts. Finally, let $u_i = \sum_{j \in \mathcal{H}} x_{i,j} c_{i,j} / d_i$ be the CPU share allocated for τ_i on host j.

The end-to-end response-time ρ of the application workflow to be admitted may now be formalized and constrained to be lower than or equal to the maximum allowed end-to-end value R, as coming from the SLA:

$$\rho = \sum_{i \in \mathcal{A} \backslash \{n\}} \left(d_i + \frac{m_i}{b_i} + \sum_{s \in \mathcal{S}} y_{i,s} L_s \right) + d_n \leq R \quad (1)$$

The resource allocation constraints to be respected are:

$$\begin{cases} \sum_{i \in \mathcal{A}} u_i x_{i,j} & \leq U_j \ \forall j \in \mathcal{H} \\ \sum_{i \in \mathcal{A} \backslash \{n\}} b_i y_{i,s} & \leq B_s \ \forall s \in \mathcal{S} \\ \sum_{i \in \mathcal{A}} \omega_i x_{i,j} & \leq \Omega_j, \ \forall j \in \mathcal{H} \end{cases},$$

where U_j and B_s denote the available resource utilizations at the time of admitting the new application. If the admission test succeeds, then these values are updated as follows:

$$\begin{cases} U_j = U_j - u_i & for\ j\ s.t.\ x_{i,j} = 1 \\ B_s = B_s - b_i & for\ s\ s.t.\ y_{i,s} = 1 \\ \Omega_j = \Omega_j - \omega_i & for\ j\ s.t.\ x_{i,j} = 1 \end{cases} \quad (2)$$

Additionally, in order to ensure that each token is fully processed before the next one arrives along the pipeline, we

need the further additional constraints (see also [2]): $\forall i \in \mathcal{A}$, $d_i \leq T$ and $\forall i \in \mathcal{A}$, $\frac{m_i}{b_i} + \sum_{s \in \mathcal{S}} y_{i,s} L_s \leq T$.

V. Algorithm Description

In this section we propose an algorithm for allocating a distributed real-time application $\mathcal{A}$ on a set of connected nodes $\mathcal{H}$ while respecting an end-to-end constraint R, as previously discussed. The proposed algorithm is capable of returning the following information:

- the Boolean variables $\{x_{i,j}\}$ that specify whether or not the service i is allocated on host j;
- the allocation parameters d_i and b_i for each service i.

From a high-level perspective, the algorithm performs a graph visit, searching for a path capable of hosting the workflow services, given their workload requirements and end-to-end deadline R. Specifically, a deadline-splitting methodology (see Algorithm 3) is used to compute a *tentative* resources requirements along the various services. Then, instead of merely failing the allocation when not enough resources are available at a step of the algorithm, we exploit the ability for the subsequent stages of the pipeline to compensate for possible shortages in the current step. Therefore, the allocation at each step is tuned so as to ensure that the end-to-end response-time till the current service is within the range $[-\alpha_{n\_thr} R, \alpha_{thr} R] \equiv [-\Delta_{n\_thr}, \Delta_{thr}]$, with α_{n_thr} and α_{thr} tunable thresholds ranging in $[0, 1] \subset \mathcal{R}$.

The main procedure of the proposed algorithm is presented in Algorithm 1. Such procedure tries to find a set of nodes satisfying the workflow requirements, by performing a search on the graph G describing the underlying network. It also takes as input the vertex u describing the beginning host of the search and the index i representing the service component τ_i to be allocated, which is increased at each recursive invocation. Basically, the performed search is based on the well-know Depth-First Search (DFS) method, enhanced by considering the arcs at each node in the order dictated by a specific ordering criterion (see line 7 in Algorithm 1), by which particular business policies can be enforced.

The core of the proposed algorithm can be considered the SERVICE_ALLOCATION procedure (see Algorithm 2), that allocates each service component τ_i on a vertex j. If τ_i can be allocated on j, an assignment is performed for its parameters d_i and b_i and the graph is updated (line 6) for reflecting such allocation, according to Eq. (2). In particular, the allocation procedure performs an initial assignment (d_i^0, b_i^0) of (d_i, b_i), taking into consideration the minimum requirements of the whole workflow (see Algorithm 3), derived by inflating the minimum allocation (d_i^{min}, b_i^{min}) due to the arrival rate T of the tokens, by a factor β expressing the ratio between the delay obtained with the minimum allocation and the maximum allowed latency R. At this time the network over which each service will be allocated is unknown, thus the corresponding delay L_s is upper-bounded by the maximum delay among the available networks.

Algorithm 1 WORKFLOW_ALLOC(G, u, i)

```
1: oc ← get_vertices_ordering_criterion()
2: if all services are allocated then
3:     return true
4: end if
5: for all j ∈ Adjacent(u) do
6:     if mark[j]=WHITE then
7:         queue ← push(j, oc)
8:     end if
9: end for
10: while queue is not empty do
11:     j ← pop()
12:     if SERVICE_ALLOCATION(i, j)=false and j satured then
13:         mark[j] ← BLACK
14:     else if WORKFLOW_ALLOC(G, j, i+1) = true then
15:         return true
16:     end if
17: end while
18: return false
```

Algorithm 2 SERVICE_ALLOCATION(i, j)

```
1: (d_i^0, b_i^0) ← INIT_PARAMS(i)
2: tc_ij ← d_i^0 − c_i/U_j
3: ts_ij ← m_i/b_i^0 − m_i/r_j
4: if VERIFY REQUIREMENTS(i) = true then
5:     (d_i, b_i) ← ASSIGNMENTS(i, j)
6:     update_status()
7:     return true
8: else
9:     return false
10: end if
```

After the initial allocation, Algorithm 2 calculates the residual times tc_{ij} and ts_{ij}, respectively referring to computing and networking power, that are positive if host j has enough resources to sustain the allocation of service i. As an example, if $tc_{ij} = 1$ it means that the host, after allocating a CPU share for sustaining a service response equal to d_i^0, can allocate additional shares for sustaining at most a response equal to 1 time unit. The same reasoning can be done for ts_{ij}, except that in this case r_j represents the remaining outbound bandwidth of j.

The computation of ts_{ij} and tc_{ij}, along with the residual $\Delta t^{(i-1)}$ coming from previous allocations (please note that for service $i = 1$, $\Delta t^{(0)} = 0$), permits to decide if the service i can be allocated on host j. In particular, the VERIFY REQUIREMENTS subprocedure calculates such quantity $\Delta t_{tmp}^{(i)} \triangleq ts_{ij} + tc_{ij} + \Delta t^{(i-1)}$ and denies the allocation (by returning FALSE) if $\Delta t_{tmp}^{(i)} < 0 \wedge |\Delta t_{tmp}^{(i)}| > \Delta t_{n\_thr}$. In case of $i = n$, it is sufficient that $\Delta t_{tmp}^{(i)} < 0$ for denying the allocation, because a deficit on the last service allocation cannot be compensated further.

Once verified the timing requirement for allocating the service τ_i on the host j, the ASSIGNMENTS procedure is performed for assigning the pair (d_i, b_i) that could potentially differ from the initial assignment (d_i^0, b_i^0). The ratio behind this procedure, not detailed due to space constraints,

Algorithm 3 INIT_PARAMS(i)

1: $N \leftarrow 2n-1$
2: $(d_i^{min},\, b_i^{min}) \leftarrow (T,\, m_i/T)$
3: $\beta \leftarrow \frac{NT}{R-(n-1)\max_{s\in\mathcal{S}}\{L_s\}}$
4: **if** $\beta > 1$ **then**
5: $\quad (d_i^0,\, b_i^0) \leftarrow (d_i^{min}/\beta,\, \beta\, b_i^{min})$
6: **else**
7: $\quad (d_i^0,\, b_i^0) \leftarrow (d_i^{min},\, b_i^{min})$
8: **end if**
9: **return** $(d_i^0,\, b_i^0)$

Table I
ALLOCATION STATISTICS BY USING DRU CRITERION.

$(\alpha_{n\_thr},$ $\alpha_{thr})$	admitted wf(%)	used hosts(%)	allocated util.(%)	util. on used hosts(%)	avg exec time(ms)
$(0, 0)$	0.610	0.816	0.452	0.555	0.962
$(0.1, 0)$	0.666	0.874	0.507	0.579	0.941
$(0.2, 0)$	0.703	0.910	0.551	0.605	0.922
$(0.1, 0.1)$	0.653	0.882	0.562	0.639	0.878
$(0.2, 0.2)$	0.500	0.726	0.512	0.599	0.732
$(0.2, 0.1)$	0.676	0.906	0.591	0.653	0.867

is keeping $\Delta t^{(i)}$ constrained in the range $[-\Delta t_{n\_thr}, \Delta t_{thr}]$. Briefly, if the condition $\Delta t_{tmp}^{(i)} > \Delta t_{thr}$ is not verified, all the remaining resources of j can be assigned to τ_i. Vice versa, we only allocate resources needed for obtaining a temporal surplus equal to $\Delta t_{mis} \triangleq \Delta t_{thr} - \Delta t^{(i-1)}$.

VI. EXPERIMENTS

This section describes a set of experiments performed for evaluating the performance of the proposed algorithm. In particular, three workflows have been subsequently submitted to the algorithm for deciding about the admission on a sample grid of 5 homogeneous hosts, each one with a network capacity of $100Mb/s$ and a maximum utilization $U_j = 0.95$. Each workflow is composed of three services, whose requirements have been randomly generated. A total of 100 repetitions have been performed for different values of α_{n_thr} and α_{thr} and by varying the criterion for ordering hosts during the graph visit phase.

Initially, we have evaluated our algorithm by ordering the hosts using the *decreasing residual utilization* (DRU) criterion and the statistics are reported in Table I.

We repeated the experiment (results are not reported due to space constraints) by ordering the hosts with the *increasing residual utilization* (IRU) criterion and we noticed that it leads to higher server consolidation levels, with the use of generally a lower number of hosts that tend to be more saturated. In both cases, it can be well appreciated the impact of the α_{thr} and α_{n_thr} parameters of the allocation heuristic on the achieved results.

Regarding the average execution times of the algorithm, they are in both cases in the order of ms. It could be interesting to note that the former optimum problem [2] applied on a comparable case study has been solved by a commercial product requiring a time in the order of seconds. Please note that this comparison is quite rough, as the model used in this paper has been simplified with respect to the former one. A more thorough comparison is deferred to future work.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, a heuristic based on graph-matching was proposed to solve the problem of allocation of distributed services over a heterogeneous network. From the presented preliminary results, the heuristic promises to be very efficient so as to be usable for on-line allocation decisions to be taken in dynamic SOA environments. In the future, a more extensive evaluation needs to be performed on the proposed technique, especially considering large networks of resources with high numbers of workflows to be deployed.

REFERENCES

[1] T. Cucinotta, F. Checconi, G. Kousiouris, D. Kyriazis, T. Varvarigou, A. Mazzetti, Z. Zlatev, J. Papay, M. Boniface, S. Berger, D. Lamp, T. Voith, and M. Stein, "Virtualised e-learning with real-time guarantees on the irmos platform," in *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010)*, Perth, Australia, 12 2010, pp. 1–8.

[2] K. Konstanteli, T. Cucinotta, and T. Varvarigou, "Optimum allocation of distributed service workflows with probabilistic real-time guarantees," *Service Oriented Computing and Applications*, vol. 4, pp. 229–243, 2010.

[3] M. Mika, G. Waligra, and J. Wglarz, "Modelling and solving grid resource allocation problem withnetwork resources for workflow applications," *Journal of Scheduling*, vol. 14, pp. 291–306, 2011.

[4] K. Nahrstedt, H.-h. Chu, and S. Narayan, "Qos-aware resource management for distributed multimedia applications," *J. High Speed Netw.*, vol. 7, pp. 229–257, December 1998.

[5] L. Wang, G. von Laszewski, M. Kunze, and J. Tao, "Schedule distributed virtual machines in a service oriented environment," in *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, ser. AINA '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 230–236.

[6] T. Cucinotta, A. Mancina, G. F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina', "A real-time service-oriented architecture for industrial automation," *IEEE Trans. on Industrial Informatics*, vol. 5, no. 3, Aug 2009.

[7] T. Cucinotta, G. F. Anastasi, and L. Abeni, "Respecting temporal constraints in virtualised services," in *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, vol. 2, July 2009, pp. 73–78.

[8] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-CS-93-157, 1993.

[9] L. Palopoli, T. Cucinotta, L. Marzario, and G. Lipari, "AQuoSA — adaptive quality of service architecture," *Software – Practice and Experience*, vol. 39, no. 1, pp. 1–31, 2009.

[10] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5(5), pp. 675–689, October 1997.