Re-allocation Cost Minimization in Distributed Dynamic Real-Time Systems

[1]Ricardo Garibay-Martínez, [1]Andrei Tchernykh, [2]Klaus H. Ecker
[1]Computer Science Department, CICESE Research Center,
Ensenada, B.C. México, {rgaribay, chernykh}@cicese.mx
[2]Ohio University, Athens, OH, USA, ecker@ohio.edu

Abstract

Autonomously operating technical systems and adaptive systems as in mechatronics are dynamic applications that are able to adjust to changed operation conditions. Typically they are realized as distributed real-time systems requiring high flexibility in control. They have to deal with varying extrinsic parameters that cause changing conditions for the frequency and the run-times of the end-to-end computations. As a consequence, re-allocations of tasks to hosts will become necessary. A re-allocation cost function is introduced that measures the effort of realizing a re-allocation of computational tasks to hosts. In this paper, we focus on the optimization of the task re-allocation by minimizing the weighted sum of task migrations. An optimal algorithm and sub-optimal heuristics for the cases of identical and uniform machines are discussed.

Keywords: reallocation, linear assignment, real time scheduling.

1. Introduction

Distributed real-time and embedded applications occur in such different critical areas as air defense systems, car and robotic control, or satellite positioning. Real-time systems are used to control technical environments being built to serve some intended purpose. Due to their unpredictable nature they will not automatically behave as expected. To enforce a certain behavior, sensors informing about the status of the environment are regularly read, and processed by computational activities, the so-called end-to-end computations [4]. That may result in new settings for actuators in case measured data deviate from the technical specification. Reading of sensor signals, their evaluation and generation of actuator signals must be done periodically, with periods depending on the particular control requirements. Examples of environments we have in mind are autonomously operating technical systems, such as satellites, or dynamic adaptive ones such as mechatronic applications that are able to adjust to changed operation conditions [2, 8]. For controlling such environments, *extrinsic parameters* inform permanently about the status or working conditions of the environment. The controlling real-time system has to react in due time, in order to keep the operating parameters within required bounds, or to guarantee some desired quality of service.

Common real-time engineering approaches use worst-case execution times to characterize task workloads *a priori* and allocate computing and network resources to processes at design time. In contrast, dynamic applications requiring higher flexibility in control have to deal with varying extrinsic parameters that lead to changing conditions for the software components in the real-time system. This may have consequences for the run-times of the end-to-end computations. As a consequence, hosts may become over- or underloaded, and the computational tasks may have to be re-distributed among the hosts. It is obvious that such re-allocations are not free of cost as moving a task (i.e., code and data) to another host consumes time, computer power and network capacity. They can also introduce additional delays and have a strong impact on the schedulability of tasks on a given host [3]. A re-allocation cost function measures the effort of realizing a re-allocation of computational tasks to hosts. Obvious measures are: (i) the weighted sum of task migrations, where the weight represents the size of transmitted code and data; (ii) communication delays in the communication system, such as single bus system, packet switching networks, or others. Optimality could then mean minimizing the communication load on the links. Case (ii) is reduced

to (i) if a single bus is assumed for transmitting the tasks: If the bus capacity allows one transmission at a time, the total migration time is given by the sum of task migration times. Reducing the re-allocation cost thus helps avoiding bus congestions.

This paper addresses the optimization of the task re-allocation by minimizing the weighted sum of task migrations. The remainder of the paper is organized as follows. Section 2 presents corresponding model and the problem definition. Section 3 describes the finding a minimum weight matching in a weighted bipartite graph. Section 4 presents sub-optimal solutions to reduce the expected computation time to be satisfactory in critical real time systems. Section 5 presents the experimental setup and results. Finally, Section 6 summarizes the paper and points out the future work.

2. System model and problem definition

The model introduced in this section follows the classical periodic task model [7, 1]. We assume the existence of a set of periodic tasks $\mathcal{T} = \{T_1, \dots, T_n\}$; each task T_j represents an end-to-end computation [4], that has to be repeatedly executed with a given period $\pi_j := \pi(T_j)$. This means that the ith instance of T_j is completely processed in the interval $[(i-1)\pi_j, i\pi_j]$, $i = 1,2,3, \dots$. Each task T_j has a known processing time (usually the worst case execution time) p_j and memory requirement that depends on extrinsic parameters $E = (e_1, \dots, e_q)$ and service parameters $S = (s_1, \dots, s_r)$. I.e., for task, p_j is a function of both parameters: $p_j := p_j(E, S)$. It is further assumed that the tasks in $\mathcal{T}$ are pair-wise independent.

A set of m hosts $\mathcal{H} = \{H_1, \dots, H_m\}$ is used for processing the tasks. Given m task subsets $(\mathcal{T}_1, \dots, \mathcal{T}_m)$ need to be allocated. An allocation can be specified by a 1-1 function $\alpha = (\mathcal{T}_1, \dots, \mathcal{T}_m) \to \{H_1, \dots, H_m\}$. Let at *current* time an allocation $alloc^{cur} = \alpha^{cur}(\mathcal{T}^{cur})$ is defined by some partition $\mathcal{T}^{cur}$ and a mapping α^{cur} is realized. W.l.o.g. let $\alpha^{cur}(\mathcal{T}^{cur}) = H_i$ for $i = 1,2,3, \dots, m$. Due to some external conditions, a new allocation defined by partition $\mathcal{T}^{new} = \{\mathcal{T}_1^{new}, \dots, \mathcal{T}_m^{new}\}$ should be performed. As a consequence, some of the tasks have to migrate to other hosts. The objective is to determine a mapping $\alpha^{new}: \mathcal{T}^{new} \to \mathcal{H}$ such that the sum of the migration cost is minimized. Moving a task T_j from host H_i ($\mathcal{T}_i^{cur}$) to host H_k ($\mathcal{T}_k^{new}$) will incur a cost $c(T_j, H_i, H_k)$ that depends, besides on the weight of the task, on the network structure. If T_j remains on the same processor, then $c(T_j, H_i, H_k) = 0$. Starting from $alloc^{cur}$ realizing the new allocation $\alpha^{new}: \mathcal{T} \to \mathcal{H}$ causes the total re-allocation cost $C(cur, new) = \sum_{j,i,k} c(T_j, H_i, H_k)$. In terms of $\mathcal{T}^{cur}$, $\mathcal{T}^{new}$, and mapping α^{new}, the re-allocation cost can be calculated as

$$C(cur, new) = \sum \{c(T_j, H_i, H_k) \mid T_j \notin \mathcal{T}_i^{cur} \cap \mathcal{T}_l^{new} \text{ and } \alpha^{new}(\mathcal{T}_l^{new}) = H_k\}$$

$T_j \notin \mathcal{T}_i^{cur} \cap \mathcal{T}_l^{new}$ represents tasks of $\mathcal{T}^{cur}$ that are not in $\mathcal{T}^{new}$, hence contributes a cost. The problem of minimizing the re-allocation cost can hence be formulated as: Given $\mathcal{T}^{cur}, \alpha^{cur}$ and $\mathcal{T}^{new}$ find α^{new} such that $C(cur, new)$ is minimized. The mapping α^{new} defines the new allocation $alloc^{cur}$ in a straight forward manner. For practical reasons we define a cost matrix C, where C_{ij} is the cost of shifting the elements of $\mathcal{T}_i^{new}$ to processor H_i.

$$C_{ij} = \sum_{k \neq j} \sum_{T \notin \mathcal{T}_k^{cur} \cap \mathcal{T}_i^{new}} c(T, H_k, H_j).$$

Re-allocation can now be formulated in the following way: Given the cost matrix (C_{ij}) find a permutation matrix (Q_{ij}) such $\sum_{ij} Q_{ij} C_{ij}$ is minimized.

3. Minimizing re-allocation cost

We now discuss strategies for the optimal allocating a partition $\mathcal{T}^{new}$ from a current allocation $alloc^{cur} = \alpha^{cur}(\mathcal{T}^{cur})$. For communication between the hosts we assume a single bus system with CSMA/CD (carrier-sense-multiple-access/collision-detection) as can be for example found in the CAN bus [3]. In this kind of access, all transmission traffic has to be sequenced. Correspondingly, since the re-allocation time should be minimized, optimization means minimizing of the weighted sum of task migrations, where weights represent the size of transmitted code and data. Since we assume a common bus system, this cost does not depend on the particular source and target host. Regarding the computational capabilities of the hosts, we first discuss the case of identical hosts, and then extend consideration to uniform hosts.

3.1. Identical hosts

If hosts are identical, any 1-1 function α^{new} defines a feasible allocation, but possibly at different cost. In order to solve the optimization problem we transform it to the weighted matching problem in bipartite graphs. The latter can be formulated as follows.

Given a bipartite graph with arc weights, find a perfect matching for which the sum of weights of the chosen arcs is maximal. In order to demonstrate the equivalence, let us note that the complete bipartite graph $G = (S, T, S \times T)$, where $|S| = |T| = m$ corresponds to the $m \times m$ matrix $C = (C_{ij})$ with C_{ij} corresponding to the appropriate arc cost. Since in the weighted matching problem a maximum is looked for,

we have to assume them to be $K - C_{ij}$, $i = 1, \dots, m$; $j = 1, \dots, m$, for $K > \max\{C_{ij}\}$. Hence minimizing re-allocation cost in our basic problem corresponds to the linear assignment problem maximizing a total weight in a perfect bipartite matching. The Hungarian algorithm with complexity $O(m^3)$ proposed by H. W. Kuhn [5] is one of many algorithms that have been devised that solve the problem within time bounded by a polynomial expression of the number of hosts.

3.2 Uniform hosts

Hosts are now assumed to have uniform capabilities in the sense that the tasks can be processed on any host H_i, but possibly at different speeds. Sizes of local memories may differ as well. Since hosts may have different processing capabilities for $i \in 1, \dots, m$, it may happen that not all hosts are able to process the tasks of a subset $\mathcal{T}_i^k$. Hence the possible allocations will be restricted, and not every 1-1 function α^{new} will define a feasible allocation. If at run-time the current allocation $\alpha^{cur}(\mathcal{T}^{cur})$ has to be changed to a new allocation realizing a partition $\mathcal{T}^{new}$, we have to check which processors are able to process the elements of $\mathcal{T}^{new}$. Define by H_i the subset of processors capable of performing the tasks of $\mathcal{T}_i^{new}$, $i = 1, \dots, m$. From our general assumption that there exists a feasible allocation realizing $\mathcal{T}^{new}$ we conclude that the sets H_i are non-empty. Furthermore, since each host has to process exactly one subset of $\mathcal{T}^{new}$ we must have $\bigcup_{i=1}^{m} H_i = H$. Such restrictions can easily be modeled in the cost matrix C by excluding certain matrix elements from consideration. For this we introduce a value "not possible" in the corresponding positions of C. Notice that the corresponding bipartite graph is not complete. The maximum matching algorithm discussed in 3.1 can also be used introducing biggest possible cost value to the certain cost matrix element to exclude it from the minimization problem solution.

4. Re-allocation heuristics

The time complexity of the optimal allocation is $O(m^3)$. It is not necessarily satisfactory from the viewpoint of computation time in critical real time systems. Algorithms to reduce the expected computation time have been extensively studied because of their practical implications in many real world situations. Yamada and Kasai [9] derived an upper bound of the search space so as not to miss an optimal solution and designed an algorithm with expected computation time $O(n^{1.6})$. Lin et al. studied variable-depth-search heuristic to produce quality near-optimal solutions [6]. Two basic operations are considered: reassign and swap.

In this paper, we make further observations on effective heuristics. Several operations are studied: Min, Rand, Diag, Swap, and Eval_swap.

Min. For given the cost matrix C *Min* chooses the smallest value for each $i = 1, \dots, m$ considering column that are not yet selected. The time complexity is $O(m^2)$.

Rand chooses randomly a host for each $i = 1, \dots, m$ considering hosts that are not yet selected. The time complexity is $O(m^2)$.

Diag selects the diagonal elements of the cost matrix C. The time complexity is $O(m)$.

Diag_swap selects diagonal elements of *C* as an initial solution. Then the swap operation widely used in the literature is applied m times. The time complexity is $O(m)$.

Eval_swap is the modification of the Diag_swap. The swap operator is performed if the cost is reduced. The time complexity is $O(m + k)$., where k is the number of swaps performed.

Figure 1 shows the relation between the number of swaps and the solution quality. It shows the average percentage (over 30 experiments) of the cost reduction versus the average percentage of the time (swap) increase and their absolute difference. Maximum number of swaps tested is m^2 (100%). The diagonal elements are chosen as an initial solution and its cost is considered as 100%. It can be seen that application of Eval_swap improves the solution quality. 10% of the time increase leads to the near 80% cost reduction. Then the cost reduction is slow down with the time increase. A tradeoff between the number of swaps and the quality of solution can be found.

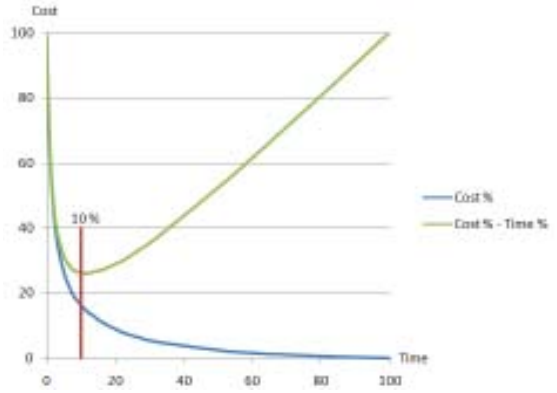

Figure 1. Cost reduction versus time increase.

5. Simulation results

The Hungarian algorithm and eight sub-optimal heuristics Min, Rand, Diag, Diag_swap, Eval_swap

with k=m, k=m^2, k=10%m^2, and k=50%m^2 are presented. The following parameters were set for the simulation. Subsets $\mathcal{T}_i^{cur}$ and $\mathcal{T}_i^{new}$ have been generated with a uniform distribution, $m = 200$ and $n = 2000$. 30 different data sets are generated.

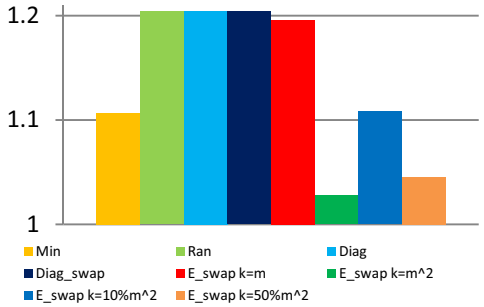

Figure 2. Cost over optimal cost

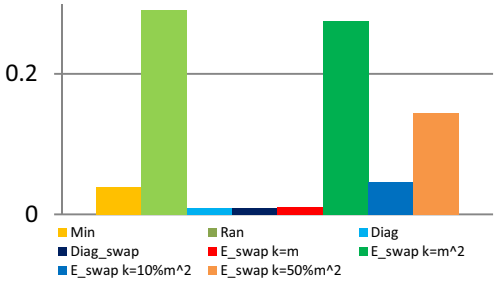

Figure 3. Execution time over the time of the optimal solution.

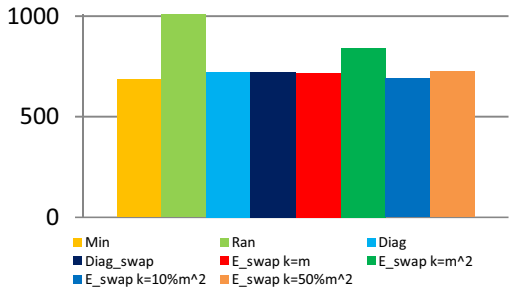

Figure 4. Relative cost.

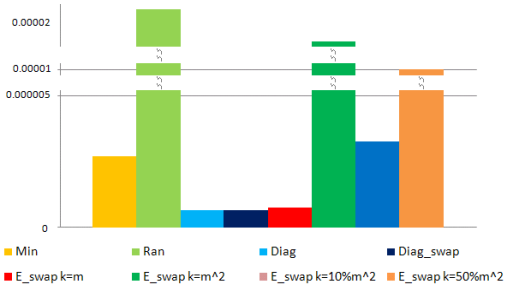

Figure 5. Relative time.

Figure 2 shows the ratio between the cost obtained by different heuristics and the optimal cost derived from the Hungarian algorithm. Figure 3 shows the ratio between the computation time of the heuristic algorithms and the computation time of the optimal algorithm. Figure 4 presents the relative cost per execution time decrease. Figure 5 presents the relative time spent per the cost reduction.

6. Conclusions and future work

In this paper, we discussed adaptive resource management in distributed real-time systems and presented several sub-optimal heuristics for the re-allocation problem. We showed that known optimal algorithm is not necessarily satisfactory in critical real time systems and tradeoff between the execution time and suitable cost can be found. We provided simulation results comparing the effectiveness of heuristics against known algorithm. Future investigations will be carried out for the heuristics in the context of non linear allocation cost and different topologies.

References

[1] I. Bate, and A. Burns, "An Integrated Approach to Scheduling in Safety-Critical Embedded Control Systems". *Real-Time Systems* 25, 2003.

[2] K. Ecker, D. Juedes, L. Welch, D. Chelberg, C. Bruggeman, F. Drews, D. Fleeman, D. Parrot, and B. Pfarr, "An Optimization Framework for Dynamic Distributed Real-Time Systems". IPDPS 2003, Workshop on Parallel and Distributed Real-Time Systems, IEEE Computer Society, 2003.

[3] K. Ecker, A. Tchernykh, F. Drews, S. Schomann, Continuous Mode Changes in Mechatronic Systems, *Proceedings of the Mexican International Conference on Computer Science* IEEE Computer Society Press, p. 407-414, 2004.

[4] R. Gerber, S. Hong, and M. Saksena, "Guaranteeing End-to-End Timing Constraints by Calibrating Intermediate Processes", *Proc. of the IEEE Real-Time Systems Symposium*, Dec. 1994.

[5] H. Kuhn, Variants of the Hungarian Method for the Assignment Problems, *Naval Res. Logist Quart*. 3,1956.

[6] B. Lin, Y. Huang, H Yu, "On the variable-depth-search heuristic for the linear-cost generalized assignment problem", *Int. journal of computer mathematics,* 2001, v. 77, no. 4, pp. 535-544.

[7] C. Liu, and J. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment", *Journal of the ACM* 20, 1973.

[8] S. Schomann, *Adaptive Resource Management in Distributed Real-Time Systems with continuous Mode Changes*, PhD Thesis, Clausthal University of Technology, Germany. 2006.

[9] S. Yamada, T. Kasai, "An efficient algorithm for the linear assignment problem", *Electronics and Comm. in Japan,* 1990, vol. 73, no. 12 pp.28-36.