DOI: 10.18372/2310-5461.60.18267
UDC 519.47(045)

Petro Stanko, PhD
National aviation university
orcid.org/0000-0002-2659-7586
e-mail: petro.stanko@npp.nau.edu.ua;

Olena Ohremchuk
National aviation university
orcid.org/0000-0003-2239-0524
e-mail: olena.okhremchuk@npp.nau.edu.ua;

Daria Salamatina
National aviation university
orcid.org/0009-0006-3451-3930
e-mail: 6872386@stud.nau.edu.ua;

Daria Sverdlova
National aviation university
orcid.org/0009-0005-5425-1336
e-mail: 6884094@stud.nau.edu.ua;

TASK SCHEDULING OPTIMISATION OF DISTRIBUTED REAL-TIME COMPUTING SYSTEMS

Introduction

B Parallel and distributed computing systems with multi-core processors are valuable resources that are usually used to solve rather highly specialized production and other problems. Users continually submit jobs to the system, each with unique resource and QoS requirements. The task of job scheduling is to maximize the total utility of the system. The area of scheduling has received much attention over the years, resulting in a significant amount of work and new solutions in the practice of high performance computing. However, it can be argued that the problem of scheduling in parallel systems is still far from a final solution. Scheduling of computing tasks in real-time systems reflects trends in high-performance computing architectures, parallel programming language models, and user priorities. No scheduling strategy is optimal for arbitrary task scenarios, so the scheduling system must be adaptive by definition. These architectural trends have led to the dominance of flexible programming models and hence scheduling with shared computing resources with batch queuing. Distributed and parallel processing are no longer the exclusive domain of supercomputers; it is moving to autonomous network segments, geographically distributed grid ensembles, and even desktop computers. At the same time, rigidity and explicit parallelism are gradually giving way to programming models that provide an alternative to traditional scheduling approaches. In this article, we highlight problems and approaches to the theory and practice of parallel computing in a production environment; still unsolved problems in the field of real-time information and control systems are highlighted. In this paper, a regular approach for the scheduling problems in real-time computing systems is covered. The paper is organized as follows. Section II presents the additional notations, terms, and definitions. In order to limit the combinatorial explosion of the method, we identify a structural property of the set of tasks schedulable at each time instant and a characterization of active schedules [1]. Our algorithm is presented in Section 3, and its correctness is established in Section 4. Section 5 is devoted to its complexity. Section 6 is our conclusion. II. Terms and Definitions Because the terms used have had multiple meanings and implied different assumptions over time, we are redefining them here for this article. This work is about job scheduling, a discipline whose goal is to decide when and where each job should be executed during the operation of a system. This direction follows directly from the theory of scheduling. The specificity of the work is that each of the threads of a separate software application is scheduled independently of the others. The term "multi-core real-time computer" is used to designate a computer shared in real-time information and control systems and capable of performing distributed parallel processing. Such machines typically have distributed

memory or a distributed shared memory interface. The processor pools of such machines are usually homogeneous. The term “task” refers to a certain (parallel) program consisting of several parallel threads transmitted for execution by the computer system. Thus, a task is inherently related to the dispatch time, and scheduling algorithms make online scheduling decisions based on the current state of the ordered task flow. Each task is characterized by two parameters: length, measured by execution time, and size, characterized by the number of threads; we assume that each thread runs on a separate processor. The job dimensions are not necessarily fixed before or during execution, although in practice this is often implied. In the literature, the dimensions of the workspace are determined according to the following classification [4]. Hard sizing-The number of processors available to execute a job is specified by the user and is independent of the scheduler software application (hereinafter referred to as “scheduler” for short). This is the number of processors available to the job throughout its execution. Flexible sizing-The number of processors assigned to a job is determined by the scheduler, within certain user-specified limits. Once a job starts, it uses the same number of processors throughout its execution. Size agreed upon by stages. A task goes through different stages that require different numbers of processors. The number of allocated processors may change at any time. The name to execute in response to a request or job failure. In a manufacturing environment, the workspace is almost always fixed, but with different design approaches, configurable workspaces can improve productivity [2]. Evolving and flexible workspaces will require increased programming workload, overhead for migrating code snippets and breakpoints, and operating system support. Given a set of jobs and available processors, a schedule is an ordered series of mappings between some threads of those jobs and the processors. In practice, schedules can easily be changed depending on work flow conditions, but sometimes advance reservations are possible [3]. The term time-sharing refers to any scheduling approach in which threads can be preempted by others during execution and restarted later. The number of jobs that each processor can execute simultaneously is called the multiprogram-ming level. In contrast, space-sharing approaches provide exclusive use of the processor by a thread until its execution is completed, or until a maximum time limit is exceeded and the thread terminates. Space sharing approaches allow you to manage time by placing each task in a queue and simultaneously executing all of its task threads once released from that queue. This division in approaches reflects the duality of sets of job requirements. Interactive jobs that require low latency are typically run using time sharing, while batch jobs that require consistent performance are run on dedicated processors using space sharing. Multi-core computers meet the requirements of both categories by statically dividing the machine's processors into subsets that share time and space.

Schedule quality

The goal of planning and system administration in general is to maximize the utility of the system. Utility cannot be quantified, but is essentially a functionality that depends on the context and many factors. These definitions rely on contexts beyond the planner, making objective assessment difficult. We can define certain qualities of said functionality, including performance, fairness, and predictability. However, there is no consensus on how these indicators relate to desirable qualities, nor on how these desirable qualities relate to utility. It is simply stated that each of them is associated with some unknown but non-decreasing function. Therefore, it is difficult to objectively assess the impact of planning decisions on a specific quality. The observed indicators allow us to make only weak statements regarding the planning policy, for example, “an increase in the indicator A, provided that all other factors remain constant, will lead to a non-negative change in the utility functional of the system.” Unfortunately, all other factors are rarely considered equal; Metric A may conflict with Metric B, although their relative impact on utility is unknown. Thus, the value of indicators lies in supporting administrative decision making by describing planning trade-offs, unfortunately in heterogeneous units of measurement. Next, we discuss each of the three desirable planning qualities mentioned above, some metrics that have been used to monitor each, and the trade-offs that exist between them. Productivity-fairness-predictability triplet The most commonly used measure of planning quality is productivity. For online scheduling algorithms, this score is measured using changes in response time. Response time, also known as thread time or processing time, is the amount of time that elapses between a job being submitted and completion. The intuition behind this metric is that users are more satisfied with faster response times, although the exact correlation is not clear [4]. Job slowdown, sometimes called dilation, is the ratio of a job's response time to its execution time on an unloaded system [5]. This metric overestimates the importance of extremely short work tasks, and "limited dilatation" has been proposed to prevent the risk of overestimating it. [6]:

$$S_{l\,down} = \max\left\{\frac{T_w + T_r}{\max\left[T_{ru}, \tau\right]}, 1\right\}, \quad (1)$$

where T_w is the task waiting time; T_r is execution time; T_{ru} is execution time in unloaded system. Naturally, limited deceleration is sensitive to the value of.

The slowdown problem, like the bounded dilation problem, is that jobs with the same response time and CPU time can have different dilation rates. Accordingly, a widely used indicator is the normalized dilatation per processor (per processor slowdown), obtained by simply dividing the slowdown by the number of processors used [6].

Another problem with dilation measures is the relationship between performance and task duration. As T_r and T_{ru} increases the impact T_w on deceleration decreases. This may have undesirable consequences for policies that aim to minimize these rates. For example, in a resource volume sharing system, this policy rewards jobs that use fewer processors because they run longer and often start quickly [5, 7, 8]. There is a contradiction with the goals of parallel processing. To eliminate this contradiction, the performance indicator is not slowdown, but response time.

To obtain unbiased comparative estimates of different scheduling schemes, the workload must be kept constant. For this reason, many studies use the so-called make pan [2, 9–11], a throughput indicator that denotes the amount of time required for a particular machine to complete a certain set of tasks with similar characteristics. Intuition suggests that a shorter turnaround time for a given set of jobs may indicate higher production productivity. Presumably higher throughput means greater utility.

Of course, high utility requires more than just high performance. For example, a scheduler that selects only fast-executing jobs that minimize schedule fragmentation can easily achieve high throughput at low performance. This is because fairness is traded for productivity.

Unless otherwise specified, the scheduler must assume that utility is best achieved by providing comparable services to each task.

The composition of jobs is a product of foreign policy and market forces that the planner cannot predict. Providing uneven levels of service to certain jobs compromises the overall usefulness of the machine. Thus, fairness is an important quality of schedules, although it is rarely quantified.

Most scheduling algorithms guarantee minimum fairness, that no job will be stalled, that is, every job will eventually complete. Stricter guarantees of fairness depend on the planning scheme. In space sharing, fairness may imply some queue order (e.g., FCFS) or that the current job will not be delayed by any job behind it in the queue [20]. With time sharing, it may be that each thread receives the same amount of CPU time, or a slice weighted by job size [12].

As you can see from the example above, fairness is often in direct conflict with productivity, and assessing the trade-off between the two is difficult.

Predictability is the gap between the response or time to complete a task and the user's expectations based on previous experience. Predictability can indirectly improve productivity by allowing users to predict when jobs will take place and plan resource usage accordingly.

Let us now consider some of the work of researchers on the problem and the results obtained.

Related work

In the literature, a number of previous work for scheduling of transportation, communication, and other systems have been done, which address the scheduling design challenges in different aspects. The theory of scheduling is characterized by a huge number of problem types (see, e.g. [5, 7, 8, 13]). This approach is based on a scheduling scheme extended with original setting to carry out the various activities. The works [2, 9] proposes to estimate the system productivity as key performance indicator (RPI) of online planning algorithms, at that the quantitative estimate of this KPI is expressed through variations of response time.

Most of the scheduling problems are combinatorial in nature [2, 3, 13]. One of the major challenges faced by high-end computing machines or supercomputers, which are widely used in scientific computing area, is energy and power efficiency [4, 14, 15]. A promising way to improve the energy and power efficiency is to employ the low-power architecture developed for optimal scheduling. The experimental results show that the scheduler can manage the thread running with lowers overhead and less storage order, thereby, improving the multi/many-core system performance. However, for fine-grained scientific workload, data communication is more complicated [16, 17]. The authors [18–21] analysed the scheduling predictability and they had shown that it can be even more important for user skill than productivity.

Other researches are mainly concerned about the runtime scheduling algorithms, which assure the maximum system throughput with acceptable system cost.

Problem statement

In this paper we will follow two guidelines. One guideline is a distinction between scheduling models, which comprise a set of scheduling

problems solved by dedicated algorithms. Thus, the aim of this paper is to present scheduling models for parallel processing, problems defined on the grounds of certain scheduling models, and algorithms solving the scheduling problems. Therefore, the second guideline is the methodology of computational complexity research.

In the scheduling theory, the focus is on the optimal distribution of the finite set of orders serviced by deterministic systems with one or more devices, with different assumptions about the nature of their service.

The dominant resource for parallel processing in recent years has been the multi-core computer. Therefore, research into parallel planning of work in this area is very relevant. In Fig. Figure 1 shows a diagram of a conceptual approach to building a multi-core computer with task distribution. Here, the job distribution coprocessor contains M cores, the job distribution coprocessor contains N cores; in the general case $M \neq N$. Sporadically occurring tasks pass through the priority channel.

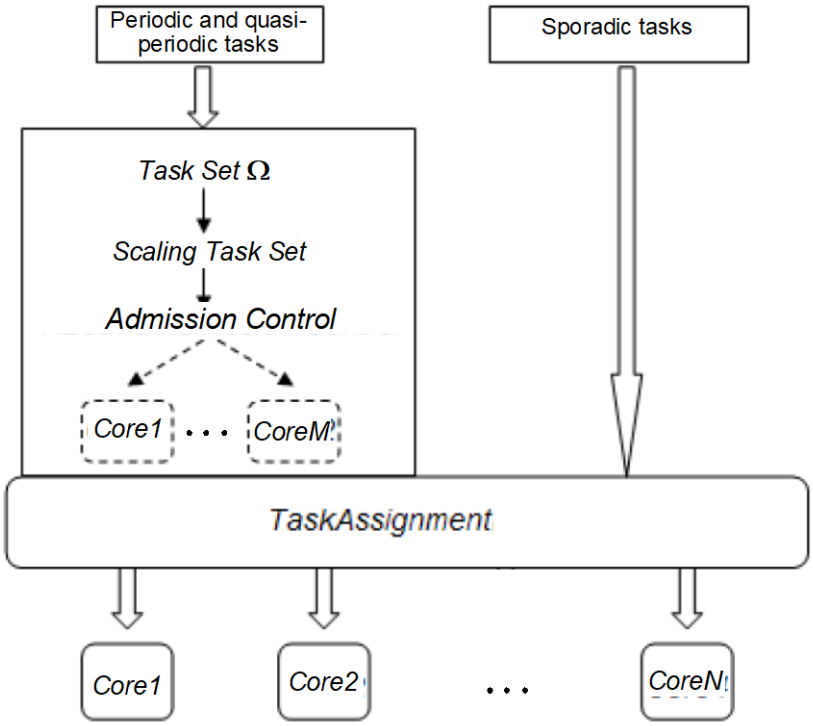

Fig. 1. Multi-core computer with task distribution; $M \neq N$

The simplest way to schedule a parallel system is with a queue. Each job is placed in a queue and, upon reaching the processing device, executed until completion. Hypothetically, the queue discipline can be standard FIFO, LIFO or another, but without loss of generality the scheme also applies to priority queues. Although this scheme provides maximum fairness and predictability, it is not very effective. Because each application uses only a subset of the system's processors, processors outside of that subset ($M \notin N$) remain idle during execution. This effect is known as fragmentation [2], and its reduction is the main goal of many studies of planning problems. The most natural extension of the queue design is space sharing, which is the simple idea of allowing another task in a queue. A queue is built for execution on idle processors, if there are enough of them. What is deceptive, however, is that even simple scheduling models, such as space partitioning using queuing, hide many assumptions, leading to intense research interest. In the remainder of this section, we discuss some of the heuristics used to select the next job to execute, as well as the implications, assumptions, and implications of such choices.

Method of optimisation servicing discipline in real-time operating systems

An important problem is the development of basic mathematical methods and equations, convenient for solving specific practical network problems. Representing the network in the form of a deterministic system and describing it with appropriate equations with deterministic parameters will give a very rough, practically useless result for the following reasons. First, it is necessary to have complete a priori information about the parameters and state of the network at any moment. Such a task is practically impossible in the vast majority of cases. Secondly, equipment failures, abnormal situations, network disruptions, overloads are fundamentally random events that

we cannot inspect and cannot control - they can only be predicted with a certain accuracy. Thirdly, even in the ideal case of having complete a priori information about the parameters, structure and instantaneous state of the network, these data will be practically useless. The systems of equations that describe the network will have an order comparable to the number of network and terminal nodes. For the numerical solution of such a system of equations in real time, an almost unrealistic amount of computing resources will be required. Therefore, at present, only statistical methods of network description, data exchange processes, network structure synthesis and parameter estimation, network management can provide results with satisfactory asymptotic accuracy. Monitoring and analysis technology is a set of diagnostic tools that allow you to objectively assess the quality of network applications (including network operating systems and other network software) and justify recommendations for improving their work. Simulation methods of real-time systems (RTS) can also provide fairly comprehensive results for specific cases of interaction of a real-time operating system (RTOS) with a real-time production hardware and software complex. Fig. 2 shows a conventional diagram of the RTS model.

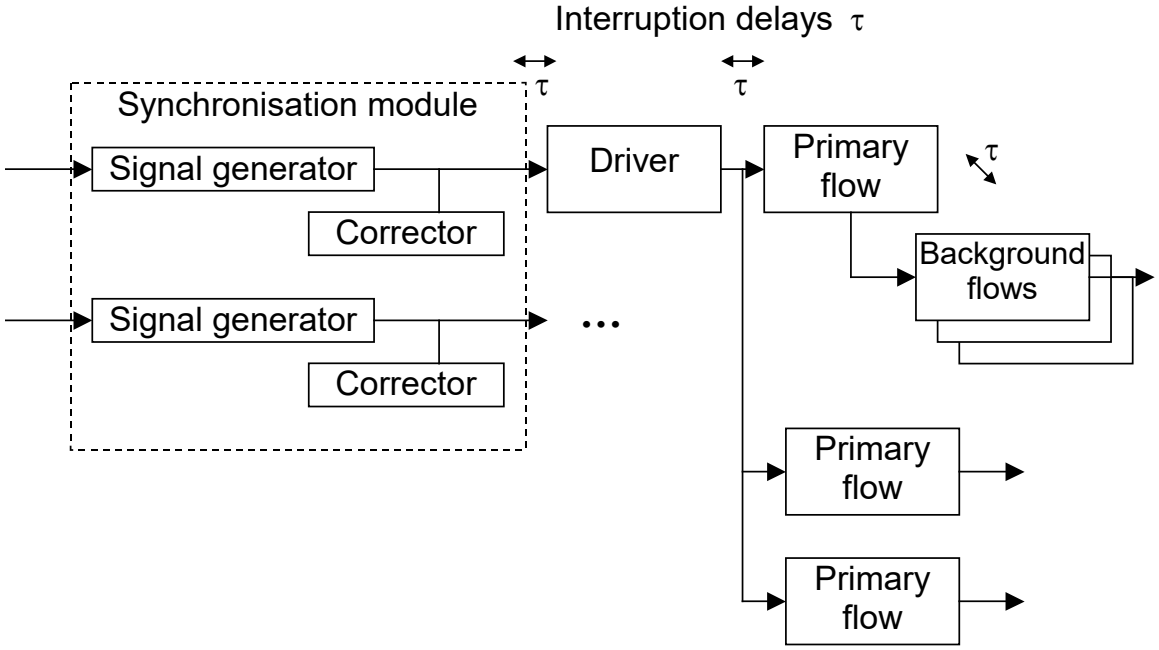

Fig. 2. Real-time system in hardware and software implementation. Background streams are activated by RTOS commands and embedded in primary streams

To optimize the service discipline, we will construct, following [20], an expression for the weighted sum of waiting times in the queue. For OS RTas a mass service system of the M/G/1 type for any service discipline, the following equality is valid:

$$\sum_{p=1}^{P} \rho_p \tau_p = \begin{cases} \dfrac{\rho \tau_0}{1-\rho}, & \rho < 1, \\ \infty, & \rho \geq 1. \end{cases} \qquad (2)$$

Here ρ is the portion of time when system is busy (ρ<1); ρ_p is partial portion of time when system serves the priority p^{th} class tasks (e.g., sporadic tasks, see fig.1).

Let us consider the RT OS with the relative priority of some labeled requirement from the priority class p. The first component of the wait time for a tagged request is related to the request it finds in the server. The second component of the waiting time for a marked request is determined by the fact that other requests that the marked request has queued up are served before the marked request. Let us denote the number of requests from class i, which the marked request (from class p) has caught in the queue and which are served before it, by N_{ip}. The average N_{ip} is average of delay component

$$\sum_{i=1}^{P} \overline{x}_i \overline{N}_{ip}.$$

The third component of the delay is related to requests that arrived after the marked request arrived, but were served before it. We denote the number of such requests by M_{ip}. The average value of this delay component is found similarly and is

$$\sum_{i=1}^{P} \overline{x}_i \overline{M}_{ip}.$$

Taking into account these components, we will write down the expression for the average waiting time in the queue for a labeled request:

$$\tau_p = \tau_0 + \sum_{i=p}^{P} \overline{x_i}\lambda_i\tau_i + \sum_{i=p+1}^{P} \overline{x_i}\lambda_i\tau_p \ ,$$

$$\tau_p = \frac{\tau_0 + \sum_{i=p}^{P} \rho_i\tau_i}{1 - \sum_{i=p+1}^{P} \rho_i} \ . \qquad (3)$$

Equations (3) are solved recursively, starting from $\tau_1, \tau_2, \ldots$.

The problem of optimizing service discipline is put in the following form: $\vec{\mathbf{U}}$ is vector of software attributes; $\vec{\mathbf{Q}}$ is vector of software quality; $\vec{\mathbf{W}}$ is vector of software exploitation parameters; $\Re_i\left(\vec{\mathbf{U}}, \vec{\mathbf{Q}}, \vec{\mathbf{W}}\right) \le \Re_{i\max}$, $i = \overline{1, N_c}$ is vector of software limit constraints.

By varying the parameters of the vectors and priority classes of the service system, we search for the extremum of the efficiency functional:

$$\Psi_{en}\left(\vec{\mathbf{U}}, \vec{\mathbf{Q}}, \vec{\mathbf{W}}\right) = \sum_{i=1}^{N_U} \alpha_i u_i + \sum_{j=1}^{N_Q} \beta_j q_j + \sum_{k=1}^{N_W} \gamma_k w_k \underset{\substack{u_i = u_{i\,opt} \\ q_j = q_{j\,opt} \\ w_k = w_{k\,opt}}}{\rightarrow} \max, \qquad (4)$$

$$i = \overline{1, N_U}, \ j = \overline{1, N_Q}, \ k = \overline{1, N_W}; \quad N_U \neq N_Q \neq N_W.$$

According to the obtained expressions (2 - 4), the characteristics of the quality of service for all priority classes are calculated and the service discipline of the RT OS is optimized. An auxiliary criterion of optimality inherent in the RT OS is the above-considered current processor load when processing explicit and background task streams.

Conclusion

The article considers the task of optimising service discipline in a real-time operating system used in production management systems and critical application systems. Analysed response latencies depending on the type of real-time operating system model as a mass service system. We derived expressions for the average waiting time of calls to the operating system kernel for processes with different priorities. It is shown that the most effective step in this process is the optimisation of the activity of the enterprises or organizations at all levels. The results of the analysis of possible disciplines of maintenance of real-time operating systems used in production and technological process management systems are presented. Using the obtained ratios for service quality characteristics, it is possible to choose the OS RV maintenance discipline depending on the state of the managed object and the type of problem to be solved.

In the future, it is planned to conduct a study of optimisation problems of real-time operating systems, which are used in multiprocessor computing systems designed to serve tasks with different priorities. it is advisable to choose the method of frequency-monotonic analysis as the theoretical foundation of research. This method, in our opinion, is the most suitable for optimisation of real-time multiprocessor computing systems designed for solving problems whose parameters, in particular, execution length, vary within very wide limits.

REFERENCES

[1] Dziyauddin, R. A., Niyato, D., Luong, N. C., Mohd Atan, A. A. A., Mohd Izhar, M. A., Azmi, M. H., & Mohd Daud, S. Computation offloading and content caching and delivery in Vehicular Edge Network: A survey. *Computer Networks*. 2021. 197. 108228. 22 p. https://doi.org/10.1016/j.comnet.2021.108228

[2] Levner E. (Ed.) Multiprocessor Scheduling: Theory and Applications. - I-Tech Education and Publishing, Vienna, Austria, 2007. 436 p.

[3] Gawiejnowicz S. Models and Algorithms of Time-Dependent Scheduling, Second Ed. Springer-Verlag GmbH Germany, part of Springer Nature 2008, 2020. 538 p.

[4] Okhremchuk O.S. Scheduling Optimisation Under Contradictions in Criteria Functions. *Science-Based technologies*, 2019. Vol. 2(42). P. 184–188. doi.10.18372/2310-5461.42.13750

[5] Pinedo M.L. Scheduling: Theory, Algorithms, and Systems, Sixth Ed. Springer Nature Switzerland, 2022. 698 p.

[6] Slomka F. Beyond the limitations of real-time scheduling theory: a unified scheduling theory for the analysis of real-time systems. *Software-Intensive Cyber-Physical Systems*, 2021. 35. p. 201–236. https://doi.org/10.1007/s00450-021-00429-1

[7] Tanaev V. S., Sotskov Y. N., Strusevich V. A. Scheduling Theory. Multi-Stage Systems. Springer Science+Business Media, Dordrecht, 1994. 404 p.

[8] Sinnen O. Task Scheduling for Parallel Systems, John Wiley & Sons, Inc., 2007. 296 p.

[9] Chakrabarti A., Chakrabarti A., Sharma N., Balas V. (Eds.) Advances in Computing Applications. Springer Singapore, 2016. 299 p.
[10] Tan D. Automatic determining optimal parameters in multi-kernel collaborative fuzzy clustering based on dimension constraint. *Neurocomputing*. 2021. 443. p. 58–74. doi.10.1016/j.neucom. 2021.02.0
[11] Baruah S. Mixed-criticality scheduling theory: scope, promise, and limitations. IEEE Xplore Digital library, 2017. 5 p. http://ieeexplore.ieee.org/Xplore
[12] Xu, S., & Hall, N. G. Fatigue, personnel scheduling and operations: Review and research opportunities. European Journal of Operational Research. 2021. 295(3), 807–822. https://doi.org/10.1016/j.ejor.2021.03.036
[13] Blazewicz J., Ecker K., Pesch E., Schmidt G., Sterna M., Weglarz J. Handbook on Scheduling: From Theory to Practice, Second Ed. Springer Nature Switzerland AG, 2019. 833 p.
[14] Baital K. Dynamic Scheduling of Tasks for Multi-core Real Time Systems based on Optimum Energy and Throughput. ReView by River Valley Technologies – IET Review Copy, 2018. 12 p.
[15] Baital, K. A. Dynamic Scheduling of Real-Time Tasks in Heterogeneous Multicore Systems. IET Comput. Digit. Tech. 2019. Vol. 13 Iss. 2, pp. 93-100. https://doi.org/10.1049/iet-cdt.2018.5114.
[16] Jinyi Xu J. Real-time task scheduling for FPGA-based multicore systems with communication delay. Microprocessors and Microsystems. 2022. Vol. 90. P. 104468.
[17] Kohútka L. A New FPGA-Based Task Scheduler for Real-Time Systems. Electronics. 2023, 12(8), 1870. https://doi.org/10.3390/electronics12081870
[18] Capota, E. A., Stangaciu, C. S., Micea, M. V., & Curiac, D.-I. Towards mixed criticality task scheduling in cyber physical systems: Challenges and perspectives. Journal of Systems and Software. 2019. 156. 204–216. https://doi.org/10.1016/j.jss.2019.06.099
[19] Ben Messaoud, M. A thorough review of aircraft landing operation from practical and theoretical standpoints at an airport which may include a single or multiple runways. Applied Soft Computing. 2020. 106853. 88 p. https://doi.org/10.1016/j.asoc.2020.106853
[20] Gerofi B. Performance and Scalability of Lightweight Multi-Kernel based Operating Systems. 2018 IEEE International Parallel and Distributed Processing Symposium. p. 116–125. doi.10.1109/IPDPS.2018.00022
[21] Jansen K. Total Completion Time Minimization for Scheduling with Incompatibility Cliques Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021). p. 192–200.

Станко П. О., Охремчук О. С., Саламатіна Д. Р., Свердлова Д. І.

ОПТИМІЗАЦІЯ ПЛАНУВАННЯ ЗАВДАНЬ В РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

Розподілені обчислювальні структури для виробничих та спеціальних цілей представляють ресурси м'яких або жорстких систем реального часу. Проблема планування завдань полягає в визначенні оптимального розподілу згідно із загальним критерієм корисності системи. У статті розглядаються методи побудови графіка, засновані на теорії планування. Показано, що найефективнішим кроком у цьому процесі є оптимізація діяльності підприємств або організацій на всіх рівнях - економічному, технічному, інформаційному і т.д., в умовах природних обмежень часових ресурсів. Оскільки оптимізація планування розкладу займає фундаментально важливе місце в процесі організації ефективної роботи розподіленої багатопроцесорної обчислювальної системи, розглядаються можливості використання цієї теорії при створенні оптимального розкладу на основі теорії черг зі звичайними і позначеними заявками. Теоретичною основою проблеми оптимізації планування є багаторівнева система з M блоками пам'яті та набором послуг N×M. З метою організації критеріїв оптимальності розкладу для забезпечення зручності опису, зберігання та програмної реалізації запропоновано умовний розподіл критеріїв на географічні, технічні або транзитні категорії із вказівкою відповідного пріоритетного значення. Враховуючи ці компоненти, отримано вираз для середнього часу очікування в черзі на заявку з позначкою. Представлена схема концептуального підходу до створення багатоядерного комп'ютера з розподілом періодичних та спорадичних завдань. Також представлена схема системи реального часу (СРЧ) в апаратній та програмній реалізації. Задачі в фоновому режимі в СРЧ активуються командами операційної системи реального часу (ОСРЧ) і вбудовуються в основні потоки. Вищезазначене демонструє універсальність запропонованого підходу до планування багатопроцесорної обчислювальної структури та його здатність відповідати вимогам користувача в реальному часі.

Ключові слова: теорія планування; багатокритеріальна оптимізація; теорія черг; оптимальний розклад; позначені заявки.

Stanko P., Ohremchuk O., Salamatina D., Sverdlova D.

TASK SCHEDULING OPTIMISATION OF DISTRIBUTED REAL-TIME COMPUTING SYSTEMS

Distributed computing structures for production and special purposes represent the resources of soft or hard real-time systems. The problem of task planning is to determine the optimal distribution according to the generalized criterion of system utility. The article discusses the methods of building a schedule, based on the theory of planning. It is shown that the most effective step in this process is the optimisation of the activity of the enterprises or organizations at all levels - economic, technical, informational, etc., under the conditions of natural limitations on time resources. Since the optimisation of schedule planning occupies a fundamentally important place in the process of organizing the effective operation of a distributed multiprocessor computing system, the possibilities of using this theory in creating an optimal schedule based on the theory of queues with ordinary and marked applications are considered. The theoretical basis of the planning optimisation problem is a multistage system with M storage units and a set of N×M services. In order to organize the schedule optimality criteria to ensure ease of description, storage and software implementation, a conditional division of criteria into geographic, technical or transit categories with a description of the corresponding priority value is proposed. Taking into account these components, an expression for the average waiting time in the queue for a marked demand is derived. The scheme of the conceptual approach to the construction of a multi-core computer with the distribution of periodic and sporadic tasks is presented. The scheme of the real-time system (RTS) in hardware and software implementation is also presented. Background streams in RTS are activated by RT operating system (RT OS) commands and are embedded in primary streams. The above demonstrates the versatility of the proposed approach to scheduling a multiprocessor computing structure and its ability to meet the user's work requirements in real time.

Keywords: scheduling theory; multi-criteria optimisation; queuing theory; optimal schedule; marked applications.

Стаття надійшла до редакції 15.11.2023 р.
Прийнято до друку 19.12.2023 р.