DOI: https://doi.org/10.36910/6775-2524-0560-2023-52-14
УДК 004.2
Скілков Нікіта Володимирович, аспірант,
https://orcid.org/0009-0001-3022-1862
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

ДОСЛІДЖЕННЯ ЧАСОВИХ ХАРАКТЕРИСТИК ПІДЗАДАЧ В БАГАТОПРОЦЕСОРНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

Скілков Н.В. Дослідження часових характеристик підзадач в багатопроцесорних системах реального часу. У статті проведено дослідження часових характеристик підзадач в багатопроцесорних системах реального часу. Зазначається, що системами реального часу це системи, в яких поряд з логічною правильністю обчислювального процесу необхідно забезпечувати також своєчасність його протікання. Наголошено, що весь спектр функціональних можливостей представляється як набір простіших завдань, до виконання яких вибираються процесорні елементи. При формуванні кожного процесорного елемента обов'язковим є дисципліна планування, відповідно до якої проводиться аналіз часових вимог задач. Підкреслено, що організація обчислень, враховує часовий розкид часу виконання, при якому завдання з великим розкидом розбиваються на дві підзадачі, перша з яких виконується за час, що не перевищує середнього часу виконання вихідної задачі, а друга являє собою залишок її виконання. Математично сформовано головну задачу направлену на виконання та наведено алгоритм її розбиття на підзадачі. Перша підзадача планується аналогічно до інших задач з детермінованими характеристиками, а друга ставиться в чергу до спорадичного серверу. Показано, що за допомогою подібних перетворень вдається досягти вищої ефективності використання процесорів. Наголошується, що кожну задачу можна представити сукупністю деяких елементарних блоків, кожен з яких описується своїм профілем часу виконання. Доведено лему, щодо набору задач який складається зі спорадичних задач з обмеженим терміном виконання та може плануватися на кількох ідентичних процесорах для будь-якого алгоритму збереження роботи. Зазначається, що перешкоди у виконанні підзадачі мають два основних джерела. Для того, щоб будь-яка підзадача могла бути запланована, часу її виконання має бути достатньо для її виконання плюс враховується робоче навантаження перешкод. Підкреслено, що головною умовою успішного планування паралельного виконання підзадач є формування низки додаткових локальних параметрів.
Ключові слова: багатопроцесорна система, задача, підзадача, швидкодія, паралельність, ефективність, розподіл, планування, реальний час, черга, алгоритм.

Skilkov N.V. Study of time characteristics of subtasks in real-time multiprocessor systems. The article examines the time characteristics of subtasks in real-time multiprocessor systems. It is noted that real-time systems are systems in which, along with the logical correctness of the computing process, it is also necessary to ensure the timeliness of its flow. It is emphasized that the entire range of functionality is presented as a set of simpler tasks, for the execution of which processor elements are selected. When forming each processor element, planning discipline is mandatory, according to which the time requirements of the tasks are analyzed. It is emphasized that the organization of calculations takes into account the time spread of the execution time, in which tasks with a large spread are divided into two subtasks, the first of which is completed in a time that does not exceed the average time of the original task, and the second represents the remainder of its execution. The main task directed to execution is mathematically formed and the algorithm for dividing it into subtasks is given. The first subtask is scheduled similarly to other tasks with deterministic characteristics, and the second is queued to the sporadic server. It is shown that with the help of such transformations it is possible to achieve a higher efficiency of the use of processors. It is emphasized that each task can be represented by a set of some elementary blocks, each of which is described by its execution time profile. A lemma is proved for a set of tasks that consists of sporadic tasks with a limited execution time and can be scheduled on several identical processors for any job-saving algorithm. It is noted that there are two main sources of obstacles in the performance of the subtask. In order for any subtask to be scheduled, its execution time must be sufficient to complete it plus the workload of obstacles. It is emphasized that the main condition for successful planning of parallel execution of subtasks is the formation of a number of additional local parameters.
Key words: multiprocessor system, task, subtask, speed, parallelism, efficiency, distribution, scheduling, real time, queue, algorithm.

Вступ та постановка проблеми. В останні роки продуктивність систем підвищують за рахунок збільшення швидкості однопроцесорних систем, що пов'язано зі зменшенням розміру чіпів, та призводить до проблем з нагріванням. Багатопроцесорні системи розглядаються як одне з рішень для подолання цих фізичних обмежень шляхом збільшення можливостей виконання завдань за допомогою паралелізму процесорів.

Використання паралелізму в програмному забезпеченні робить їх сумісними з багатопроцесорним обладнанням, оскільки обчислення паралельних додатків виконуються на кількох процесорах одночасно.

У системах реального часу планування паралельних завдань на багатопроцесорних системах є складною проблемою, а розширення умов планування однопроцесорних систем на паралельні багатопроцесорні системи не є тривіальним. Аналіз планування однопроцесорних

систем можна розділити на дві основні категорії: або шляхом перетворення обмежень черговості завдань у незалежні послідовні завдання, які виконуються на багатопроцесорних системах, або шляхом планування паралельних завдань безпосередньо за допомогою адаптованих алгоритмів планування. Перший метод спрощує планування ціною втрати деяких характеристик паралельних завдань, оскільки він усуває залежності підзадач, щоб потім можна було використовувати класичні алгоритми планування. Необхідність синхронізації між паралельними завданнями та процесорами ускладнює процес планування.

Аналіз останніх досліджень і публікацій. Формулювання наукової думки в окресі планування паралельних завдань в багатопроцесорних системах реального часу є різнорідним та масштабним. У сучасній науковій площині з'являються роботи присвячені механізмам та принципам застосування багатопроцесорних систем та підвищення їх швидкодії. Здебільшого наукові роботи зосереджені на традиційній моделі виконання послідовних незалежних завдань у реальному часі. Так, В. Г. Зайцев та Є. І. Цибаєв [1] здійснили оцінку часових характеристик задач в багатопроцесорних системах реального часу з використанням сіток Петрі. Авторами запропоновано метод оцінювання часових характеристик задач в системах реального часу шляхом аналізу даних, отриманих шляхом моделювання розподілу процесорного часу між задачами згідно обраних алгоритмів планувальника з використанням моделі сіток Петрі для багатопроцесорних систем.

А. І. Косолап та Н. С. Волинець [2] запропонували метод точної квадратичної регуляризації для розв'язання задач оптимального розподілення ресурсів. Проведені чисельні експерименти засвідчують значну перевагу методу точної квадратичної регуляризації над існуючими методами.

Алгебри алгоритмів для моделювання розподілу ресурсів в ІТ проектах висвітлили А. Василюк та Т. Басюк [3]. Результатом роботи науковців став прототип програмного забезпечення, що реалізує моделювання розподілу ресурсів із використанням методу Балаша та апарату алгебри алгоритмів.

Ю. С. Клушин [4] провів оцінювання надійності паралельних обчислювальних систем під час виконання заданого комплексу взаємопов'язаних робіт. Це оцінювання безпосередньо пов'язане з ефективністю використання обчислювальних систем.

Із зарубіжних авторів варто відзначити такі роботи як: Джастіна Ікуомола, Камхіє Манар, Джордж Лоран, Мідоннет Серж [5], Куо Чін-Фу, Лінь Цзянь-Сін, Лу Юнг-Фен [6], Донг Чжен, Лю Конг [7], Гавідель Абольфазл [8], Куо Чін -Фу, Чень Цзу-Чіє [9], Сунь Чженью, Го Мен'їн, Лю Сінгу [10], Тейшейра Рікардо, Ліма Джордж [11], Ісмаїл Хабіба, Джававі Даянг, Ахмеді Ісмаїл [12], Баруа Санджой, Бертонья Марко, Буттаццо Джорджо [13], Кумар Аджитеш [14] та інші.

Однак, незважаючи на масштабність наукових досліджень за окресленою тематикою, питання дослідження часових характеристик підзадач в багатопроцесорних системах реального часу залишається відкритим та потребує детального опрацювання.

Постановка завдання. Дослідити часові характеристики підзадач в багатопроцесорних системах реального часу.

Викладення основного матеріалу дослідження. Нехай є набір задач τ з n задачами зі спрямованим ациклічним графом (DAG), запланованих на m ідентичних процесорах. Кожне завдання τ_i DAG являє собою спорадичний граф з обмеженим терміном виконання, що складається з n_i підзадач з обмеженнями пріоритету.

Завдання τ_i DAG є можливість представити як:

$$n_i, \{1 \le j \le n_i | \tau_{i,j}\}, G_i, D_i, T_i$$

де n_i – кількість підзадач,

$1 \le j \le n_i | \tau_{i,j}$ – набір підзадач,

G_i – набір спрямованих зв'язків між підзадачами,

D_i – це відносний термін виконання τ_i,

T_i – це мінімальний час між надходженнями послідовних завдань.

Нехай $\tau_{i,j}$ позначає j-у підзадачу з безлічі підзадач, що утворюють задачу DAG τ_i, де $1 \le j \le n_i$. Кожна підзадача $\tau_{i,j}$ – це однопотокове завдання, яке має єдиний параметр синхронізації, який є найгіршим часом виконання (WCET) $C_{i,j}$. Підзадачі завдання τ_i DAG успадковують період і кінцевий термін виконання свого DAG.

Нехай $g_{i,k}^{i,j} \in G_i$ являють собою прямий зв'язок від підзадачі $\tau_{i,j}$ до $\tau_{i,k}$. Прямий зв'язок між підзадачею $\tau_{i,j}$ і $\tau_{i,k}$ означає, що підзадача $\tau_{i,k}$ не може почати своє виконання, поки підзадача $\tau_{i,j}$ не завершить виконання. У цьому випадку підзадача $\tau_{i,j}$ називається батьківською підзадачею $\tau_{i,k}$, де

$$\tau_{i,j} \in parents\left(\tau_{i,k}\right) \in pred\left(\tau_{i,k}\right)$$

де $pred\left(\tau_{i,k}\right)$ – це набір усіх попередників підзадачі $\tau_{i,k}$ DAG які повинні виконуватися опосередковано перед $\tau_{i,k}$. Аналогічно, підзадача $\tau_{i,k} \in children\left(\tau_{i,j}\right)$ називається дочірньою підзадачею $\tau_{i,j}$, а безліч всіх наступників $\tau_{i,k}$ позначається $succ\ (\tau_{i,k})$. Підзадача $\tau_{i,j}$ може містити нуль або більше батьківських / дочірніх підзадач. Початкова підзадача не має батьківських підзадач, а підзадача-приймач – це підзадача без будь-яких наступників.

Нехай C_i позначає загальний WCET завдання τ_i DAG, де $C_i = \sum_{k=1}^{n_i} C_{i,k}$. Нехай L_i позначає довжину критичного шляху завдання τ_i DAG, яка визначається як найдовший шлях виконання в τ_i, коли вона виконується на платформі з нескінченною кількістю процесорів.

Є можливість припустити, що кожне завдання DAG τ_i генерує нескінченну послідовність завдань. Нехай J_i^k це k-те завдання DAG τ_i, яке характеризується (r_i, d_i), де r_i – час виконання завдання, а d_i – його абсолютний крайній термін. Кожне завдання DAG J_i^k складається з набору підзадач, кожна з яких позначається як $J_{i,j}^k, j \in 1 \dots n_i$.

На рисунку 1 наведено схему реалізації задачі DAG τ_1, яка складається з 6 підзадач. Підзадача $\tau_{1,1}$ є джерелом DAG,а $\tau_{1,6}$ – її приймачем. Лінії на рисунку являють собою спрямовані обмеження пріоритету між підзадачами. Критичний шлях τ_1 дорівнює $\{\tau_{1,1}, \tau_{1,2}, \tau_{1,6}\}$, а його довжина дорівнює $L_i = 6$. Для підзадачі $\tau_{1,5}$ її батьківською підзадачею є $\tau_{1,3}$, тоді як $\tau_{1,1}$ є однією з її попередниць.

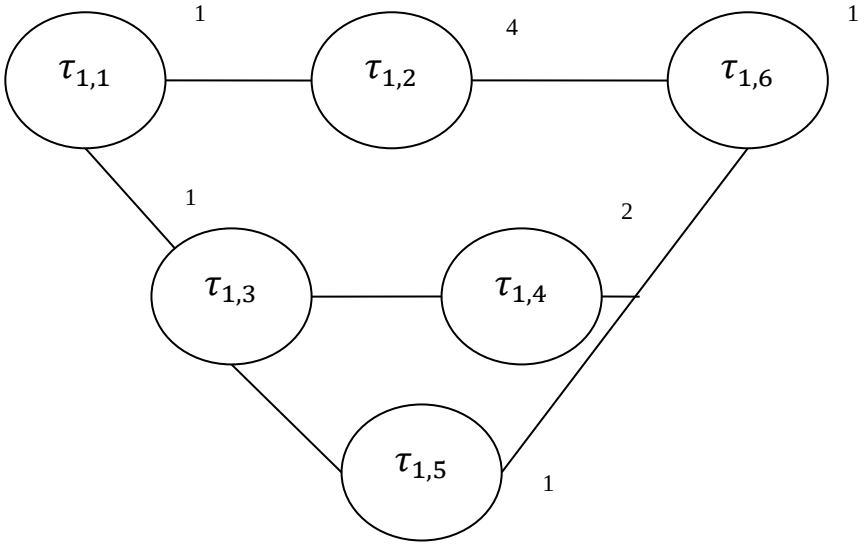

Рисунок 1 – Схема реалізації задачі DAG τ_1

Для будь-якого набору задач DAG існують дві необхідні базові умови, якщо хоча б одна з них хибна, набір завдань неможливий:

$$\sum_{\tau_i \in \tau} \frac{C_i}{T_i} \leq m$$
$$\forall \{\tau_i \in \tau\}: L_i \leq D_i$$

Для описаного набору формування задачі проаналізуємо можливість планування з використанням будь-якого глобального алгоритму планування зі збереженням роботи. Глобальний алгоритм допускає міграцію завдань і переважне завантаження між процесорами, в той час як

алгоритм збереження роботи не дозволяє затримку виконання активного завдання, якщо в системі є незайнятий процесор.

При плануванні наборів DAG в багатопроцесорних системах перешкоди для кожного завдання DAG виникають у випадках коли:

– надходять зовнішні перешкоди від завдань з більш високим пріоритетом, в яких деякі або всі підзадачі можуть сприяти виникненню перешкод,

– присутнє внутрішнє втручання підзадач однієї і тієї ж групи DAG по відношенню один до одного.

Для паралельних завдань DAG алгоритм планування на рівні DAG означає, що рішення планування базуються на глобальних параметрах завдань DAG. Відповідно до цього розподіл пріоритетів здійснюється на задачі DAG, які потім застосовуються до їх відповідних підзадач. Аналіз планованості на рівні DAG залежить від глобальних параметрів DAG, таких як їх термін, період, загальний WCET та тривалість їх критичного шляху. Зазвичай результати тестів на планованість є песимістичними, і внутрішня структура DAGS не враховується в проведеному аналізі.

Отже, аналіз перешкод при плануванні на рівні DAG важко розрахувати, і ще важче визначити точні джерела перешкод. Однак, якщо алгоритм планування використовує додаткові знання щодо підзадач і потоків їх виконання, то аналіз перешкод може бути більш точним. У цьому випадку кажуть, що процес планування виконується на рівні підзадачі. Відповідно до цього, підзадачам будуть присвоєні пріоритети на основі алгоритму планування. Однак аналіз планованості вимагає додаткових часових параметрів для кожної підзадачі, відмінних від її WCET, що надаються моделлю DAG.

У випадку застосування методу додавання локальних параметрів часу до підзадач на основі їх залежностей та обмежень пріоритету між ними, проблема планування паралельних завдань DAG у багатопроцесорних системах буде спрощена до планування набору незалежних послідовних підзадач у багатопроцесорних системах, що широко вивчено в літературі.

Додавання двох часових параметрів, на додаток до параметра WCET (наданого моделлю), до кожної підзадачі в наборі DAG здійснюється для поліпшення аналізу перешкод при плануванні DAG у разі планування GEDF. Ці два параметри – це локальне зміщення та кінцевий термін виконання кожної підзадачі, отримують їх із внутрішньої структури завдання DAG та потоку виконання підзадач. Цей сценарій реалізується, коли набір DAG виконується на нескінченному числі процесорів, а всі його підзадачі виконуються паралельно максимально швидко, без затримок через перешкоди.

Локальне зміщення $O_{i,j}$ підзадачі $\tau_{i,j}$ визначається як мінімально можливий час активації підзадачі порівняно з терміном формування її завдання τ_i DAG.

Локальний термін виконання $D_{i,j}$ підзадачі $\tau_{i,j}$ – це максимальний час завершення виконання підзадачі $\tau_{i,j}$ такий, щоб у її наступників залишалося достатньо часу для виконання локальних крайніх термінів в межах реалізації загального завдання.

Для кожної підзадачі локальне зміщення та кінцевий термін обчислюються за допомогою простих алгоритмів пошуку глибини. Локальне зміщення підзадачі враховує час, необхідний для виконання попередніх підзадач. Відповідно, локальний термін виконання підзадачі залишає достатньо часу для виконання наступних підзадач.

Якщо підзадача $\tau_{i,j} \in \tau_i$ пропустить свій локальний термін $D_{i,j}$, то її задача DAG τ_i, безумовно, пропустить свій термін D_i.

Виходячи з визначення локального терміну, коли підзадача пропускає свій локальний термін, часу, що залишився до глобального терміну завдання DAG, в кращому випадку недостатньо для виконання наступних завдань. Таким чином, ранній збій у плануванні може бути оголошений на основі терміну виконання підзадачі замість того, щоб чекати пропуску терміну DAG.

Локальні зміщення та терміни виконання підзадач обчислюються на основі їх найкращого сценарію активації, в якому всі підзадачі виконуються відразу після їх звільнення без перешкод або затримок, ці параметри допомагають визначити найдовше вікно виконання кожної підзадачі. Однак активація підзадачі може бути відкладена через втручання підзадач з більш високим пріоритетом. Відповідно, остання можлива активація підзадачі відбувається, коли всі її попередники виконуються якомога пізніше (безпосередньо перед їх локальними термінами). Відповідно до цього,

активація підзадачі може відбутися в будь-який час протягом цього інтервалу, і це можна розглядати як максимальний джиттер випуску підзадачі.

Максимальний джиттер випуску $\hat{J}_{i,j}$ підзадачі $\tau_{i,j}$ визначається як різниця між самим раннім і самим пізнім часом випуску підзадачі щодо активації завдання DAG.

$$\hat{J}_{i,j} = \max_{\forall \tau_{i,k} \in Parents(\tau_{i,j})} \left(D_{i,k} - \left(O_{i,j} - O_{i,k} \right) \right)$$

На основі визначення джиттеру випуску підзадачі всі її завдання звільняються в межах інтервалу джиттеру:

$$\forall J_{i,k}^{j}, j \in N, \tau_{i,k} \in \tau_i : r_{i,k}^{j} \in \left[O_{i,k}, O_{i,k} + \hat{J}_{i,j} \right]$$

На основі вищевикладеного видно, що розрахунок джиттеру вивільнення підзадач є песимістичним і завжди враховує, що попередні підзадачі виконуються якомога пізніше. Відповідно до цього, критичні підзадачі задачі DAG (підзадачі, що формують її критичний шлях) не матимуть часу на реалізацію, якщо вони активовані при максимальному джиттеру випуску.

Що стосується періоду підзадач (мінімальний час між надходженнями для спорадичних задач), то кожна підзадача успадковує період своєї задачі DAG, де $\forall \tau_{i,j} \in \tau_i, T_{i,j} = T_i$.

В результаті підзадачі даної задачі DAG тепер характеризуються локальним зміщенням, WCET, локальним кінцевим терміном і джиттером випуску підзадачі. Ці параметри дозволять обробляти підзадачі індивідуально і незалежно один від одного, щоб забезпечити аналіз планованості на рівні під задач та паралельність виконання.

Для процесу планування на рівні підзадач, що використовує будь-який алгоритм економії роботи, виконання підзадачі може бути заблоковано підзадачами з більш високим пріоритетом. Перешкода в підзадачі $\tau_{k,h} \in \tau_k$ визначається наступним чином:

$I_{k,h}(a,b)$ – це довжина всіх інтервалів, в яких підзадача $\tau_{k,h}$ готова до виконання, але заблокована підзадачами з вищим пріоритетом в інтервалі $[a,b)$.

$I^{i,j}{}_{k,h}(a,b)$ – це довжина всіх інтервалів, в яких підзадача $\tau_{k,h}$ готова до виконання, але заблокована підзадачею $\tau_{i,j}$, яка має вищий пріоритет в інтервалі $[a,b)$.

Оскільки підзадачі є однопотоковими послідовними задачами реального часу, співвідношення між $I_{k,h}(a,b)$ та $I^{i,j}{}_{k,h}(a,b)$ позначається наступним рівнянням:

$$I_{k,h}(a,b) = \frac{1}{m} \cdot \sum_{\forall \tau_{i,j} \in \tau_i \in \tau} I^{i,j}{}_{k,h}(a,b)$$

Через характеристики задач DAG та обмеження пріоритету між підзадачами перешкоди в підзадачі $\tau_{k,h}$ поділяються на два джерела, зовнішні та внутрішні перешкоди. Нехай $Ie_{k,h}(a,b)$ позначають перешкоди від підзадач з більш високим пріоритетом завдань DAG, відмінних від τ_k в наборі, який визначається наступним чином:

$$Ie_{k,h}(a,b) = \frac{1}{m} \cdot \sum_{i \neq k, \forall \tau_{i,j} \in \tau_i} I^{i,j}{}_{k,h}(a,b)$$

Де деякі або всі підзадачі задачі τ_i $(i \neq k)$ можуть впливати на $\tau_{k,h}$ залежно від їх пріоритетів.

Крім того, підзадачі τ_k можуть блокувати виконання $\tau_{k,h}$, що визначається як внутрішнє втручання $Ii_{k,h}(a,b)$. Оскільки розглядається задача з обмеженими термінами виконання, для неї кожна підзадача вносить свій внесок в перешкоди. Внутрішні перешкоди залежать від типу підзадач, які діляться на наступні категорії:

попередня підзадача $\tau_{k,x} \in pred(\tau_{k,h})$ підзадачі $\tau_{k,h}$: ця підзадача затримає активацію $\tau_{k,h}$, але як тільки підзадача $\tau_{k,x}$ завершить своє виконання, підзадача $\tau_{k,h}$ запуститься сама по собі, і подальшого ефекту $\tau_{k,x}$ не буде на $\tau_{k,h}$,

споріднена підзадача $\tau_{k,x} \in sibling(\tau_{k,h})$ – це підзадача, яка виконується паралельно без залежностей з підзадачею $\tau_{k,h}$. $sibling(\tau_{k,h})$ або споріднена задача $(\tau_{k,h})$ – це набір підзадач, які не є попередниками або наступниками підзадачі $\tau_{k,h}$,

наступна підзадача $\tau_{k,x} \in succ(\tau_{k,h})$ не впливає на підзадачу $\tau_{k,h}$, оскільки обидві підзадачі не можуть виконуватися паралельно, і підзадача $\tau_{k,x}$ починає своє виконання після того, як $\tau_{k,h}$ завершиться. Відповідно до цього, ${I^{k,h}}_{k,h}(a,b) = 0$,

підзадача $\tau_{k,h}$ сама по собі не впливає, оскільки розглядається завдання DAG з обмеженим терміном виконання, в якому в будь-який момент часу t активується тільки одне завдання кожного завдання DAG. Отже, ${I^{k,h}}_{k,h}(a,b) = 0$.

Виходячи з наведених вище визначень, внутрішня перешкода $Ii_{k,h}(a,b)$ для підзадачі $\tau_{k,h}$ в інтервалі $[a,b)$ визначається як:

$$Ii_{k,h}(a,b) = \frac{1}{m} \cdot \sum_{\forall \tau_{k,i} \notin succ(\tau_{k,h}), i \neq h} {I^{k,i}}_{k,h}(a,b)$$

Нехай $J^*_{k,h}$ – завдання підзадачі $\tau_{k,h}$, яке має максимальну перешкоду, і нехай $I_{k,h}(r^*_{k,h}, d^*_{k,h})$ позначають найгіршу перешкоду для підзадачі $J^*_{k,h}$ із $\tau_{k,h}$ в інтервалі $[r^*_{k,h}, d^*_{k,h})$. Позначимо $I_{k,h}(r^*_{k,h}, d^*_{k,h})$ як $\hat{I}_{k,h}$.

Лема 1. Набір задач τ, що складається зі спорадичних задач DAG з обмеженим терміном виконання, може плануватися на m ідентичних процесорах для будь-якого алгоритму збереження роботи, якщо:

$$\forall \tau_{k,h} \in \tau_k \in \tau$$
$$\hat{I}_{k,h} = \widehat{Ie}_{k,h} + \widehat{Ii}_{k,h} \leq (D_{k,h} - C_{k,h})$$

Доказ. Доказ цієї леми є прямолінійним. Перешкоди у виконанні підзадачі мають два основних джерела: внутрішній і зовнішній. Для того, щоб будь-яка підзадача могла бути запланована, часу її виконання (між її активацією та кінцевим терміном) має бути достатньо для її виконання плюс враховується робоче навантаження перешкод.

Висновки. У роботі проведено дослідження часових характеристик підзадач в багатопроцесорних системах реального часу. Доведено, що на планування задач DAG у реальному часі впливає внутрішня структура DAG і потік виконання її підзадач. Запропоновано використання алгоритмів планування на рівні підзадач замість рівня DAG. Це означає, що рішення щодо планування базуються на локальних параметрах підзадач замість глобальних параметрів DAG. Головною умовою успішного планування паралельного виконання підзадач є формування низки додаткових локальних параметрів, таких як локальне зміщення, крайній термін і джиттер випуску для кожної підзадачі. Проведено аналіз перешкод і робочого навантаження для алгоритму планування.

Перспективами подальшого дослідження є аналіз показників продуктивності, таких як коефіцієнт прискорення та коефіцієнт наближення. У сукупності ці показники є показниками ефективності планування.

Список бібліографічного опису

1. Зайцев В. Г., Цибаєв Є.І. Оцінка часових характеристик задач в багатопроцесорних системах реального часу з використанням сіток Петрі. Управління розвитком складних систем. 2020. № 42. С. 43–50; dx.doi.org\10.32347/2412-9933.2020.42.43-50.
2. Косолап, А., Волинець, Н. (2018). Оптимальний розподіл ресурсів у багатопроцесорних системах. Математичне моделювання. № 2(39). С. 89–94. DOI: 10.31319/2519-8106.2(39)2018.154226.
3. Василюк А. Басюк Т. Алгебри алгоритмів для моделювання розподілу ресурсів в IT проектах. *Information Systems And Networks,* 2023. С. 156-166. https://doi.org/10.23939/

4. Клушин Ю. С. Оцінювання надійності паралельних обчислювальних систем під час виконання заданого комплексу взаємопов'язаних робіт. Комп'ютерні системи і мережі, 2019. Т. 1, № 1. С. 15-23.
5. Qamhieh Manar, George Laurent, Midonnet Serge. Stretching algorithm for global scheduling of real-time DAG tasks. Real-Time Systems. 2019. №55. https://doi.org/10.1007/s11241-018-9311-1.
6. Kuo Chin-Fu, Lin Jian-Xing, Lu Yung-Feng. Energy-efficient scheduling algorithm for real-time tasks with multiple parallel segments in multiprocessor systems. RACS '19: Proceedings of the Conference on Research in Adaptive and Convergent Systems. 2019. P.14-19. https://doi.org/10.1145/3338840.3355649.
7. Dong Zheng, Liu Cong. New Analysis Techniques for Supporting Hard Real-Time Sporadic DAG Task Systems on Multiprocessors. 2018.
8. Ghavidel Abolfazl. Safety Verification of Rate-Monotonic Least-Splitting Real-Time Scheduler on Multiprocessor System. Journal of Computer and Knowledge Engineering. 2016. №1. https://doi.org/10.22067/cke.v1i2.58792.
9. Kuo Chin-Fu, Lu Yung-Feng, Chen Tzu-Chieh. Scheduling Algorithm for Tasks with Multiple Parallelization Options on Multiprocessor Systems. 2016. P. 175-180. https://doi.org/10.1145/2987386.2987390.
10. Sun Zhenyu, Guo Mengying, Liu Xingwu. A Survey of Real-Time Scheduling on Multiprocessor Systems. 2021. https://doi.org/10.1007/978-981-16-7443-3_7.
11. Teixeira Ricardo, Lima George. Shared resources in multiprocessor real-time systems scheduled by RUN. Real-Time Systems. 2022. № 58. https://doi.org/10.1007/s11241-021-09374-3.
12. Ismail Habibah, Jawawi Dayang, Ahmedy Ismail. A Hybrid Real-Time Scheduling Mechanism Based on Multiprocessor for Real-Time Tasks in Weakly Hard Specification. 2022. https://doi.org/10.1007/978-3-031-10461-9_15.
13. Baruah Sanjoy, Bertogna Marko, Buttazzo Giorgio. Multiprocessor Scheduling for Real-Time Systems. 2015. https://doi.org/10.1007/978-3-319-08696-5.
14. Kumar Ajitesh. A Systematic Survey of Multiprocessor Real-Time Scheduling and Synchronization Protocol. International Journal of Sensors, Wireless Communications and Control. 2022. № 12. https://doi.org/10.2174/2210327912666220105141851.

References

1. Zaitsev V. G., Tsybaev E. I. Estimation of time characteristics of tasks in multiprocessor real-time systems using Petri nets. Management of the development of complex systems. 2020. № 42. P. 43-50; dx.doi.org\10.32347/2412-9933.2020.42.43-50.
2. Kosolap, A., Volynets, N. (2018). Optimal resource allocation in multiprocessor systems. Mathematical modeling. № 2(39). С. 89-94. DOI: 10.31319/2519-8106.2(39)2018.154226.
3. Algebraic algorithms for modeling resource allocation in IT projects. Information Systems And Networks, 2023. С. 156-166. https://doi.org/10.23939/.
4. Estimating the reliability of parallel computer systems during the execution of a given set of interrelated works. Computer systems and networks, 2019. Т. 1, № 1. С. 15-23.
5. Qamhieh Manar, George Laurent, Midonnet Serge. Stretching algorithm for global scheduling of real-time DAG tasks. Real-Time Systems. 2019. №55. https://doi.org/10.1007/s11241-018-9311-1.
6. Kuo Chin-Fu, Lin Jian-Xing, Lu Yung-Feng. Energy-efficient scheduling algorithm for real-time tasks with multiple parallel segments in multiprocessor systems. RACS '19: Proceedings of the Conference on Research in Adaptive and Convergent Systems. 2019. P.14-19. https://doi.org/10.1145/3338840.3355649.
7. Dong Zheng, Liu Cong. New Analysis Techniques for Supporting Hard Real-Time Sporadic DAG Task Systems on Multiprocessors. 2018.
8. Ghavidel Abolfazl. Safety Verification of Rate-Monotonic Least-Splitting Real-Time Scheduler on Multiprocessor System. Journal of Computer and Knowledge Engineering. 2016. №1. https://doi.org/10.22067/cke.v1i2.58792.

9. Kuo Chin-Fu, Lu Yung-Feng, Chen Tzu-Chieh. Scheduling Algorithm for Tasks with Multiple Parallelization Options on Multiprocessor Systems. 2016. P. 175-180. https://doi.org/10.1145/2987386.2987390.
10.Sun Zhenyu, Guo Mengying, Liu Xingwu. A Survey of Real-Time Scheduling on Multiprocessor Systems. 2021. https://doi.org/10.1007/978-981-16-7443-3_7.
11.Teixeira Ricardo, Lima George. Shared resources in multiprocessor real-time systems scheduled by RUN. Real-Time Systems. 2022. № 58. https://doi.org/10.1007/s11241-021-09374-3.
12.Ismail Habibah, Jawawi Dayang, Ahmedy Ismail. A Hybrid Real-Time Scheduling Mechanism Based on Multiprocessor for Real-Time Tasks in Weakly Hard Specification. 2022. https://doi.org/10.1007/978-3-031-10461-9_15.
13.Baruah Sanjoy, Bertogna Marko, Buttazzo Giorgio. Multiprocessor Scheduling for Real-Time Systems. 2015. https://doi.org/10.1007/978-3-319-08696-5.
14.Kumar Ajitesh. A Systematic Survey of Multiprocessor Real-Time Scheduling and Synchronization Protocol. International Journal of Sensors, Wireless Communications and Control. 2022. № 12. https://doi.org/10.2174/2210327912666220105141851.