УДК 004.2
DOI https://doi.org/10.32782/2663-5941/2023.5/36

Скілков Н.В.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

АНАЛІЗ ЧАСОВИХ ХАРАКТЕРИСТИК ЗАДАЧ У КОМП'ЮТЕРНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

У статті розглянуто проблеми визначення часових характеристик задач у комп'ютерних системах реального часу. Зазначається, що для проектування системи реального часу потрібно вирішити цілий комплекс доволі складних задач, які складаються с функціональної частини, вибору оптимальних технічних засобів задля вирішення поставленої задачі, а також компоновки попередніх задач у єдину цілу оптимальну операційну систему реального часу. Підкреслено, що вирішення цілісної задачі складається з двох напрямків, теоретичного обґрунтування та моделювання реалізації. Методи реалізації формуються на базі статичних моделей та моделей побудованих за допомогою сіток Петрі. Наголошується, що розробка моделей за допомогою апарату сітки Петрі ґрунтується на ідеї розділення аналогової величини – часу виконання певної задачі на процесорі на дискретні кванти, які в свою чергу передаються між позиціями в сітці Петрі як маркери. Сформовано набір параметрів задачі реального часу та здійснено обґрунтування етапу розділення задачі на підзадачі. Графічно представлено задачу змодельовану на одному з процесорів, описано сукупність переходів. Запропоновано загальну схему моделювання для більш детального розуміння процесу. Наведено задачу з вкладеною множиною задач розміткою JSON. Зазначається, що генератор черги задач в свою чергу розпаковує такий список задач у «плоский» список, де всі три задачі рівноправні і поступають у диспетчер задач у такому вигляді та мають свої незалежні часові характеристики. Доведено, що за рахунок того що підзадачі будуть виконані не пізніше ніж їх коренева задача, це дозволяє оперувати ними, як незалежними задачами і застосувати до них запропонований метод моделювання багатопроцесорних систем.

Ключові слова: *багатопроцесорна система, моделювання, метод, часова характеристика,задача, підзадача.*

Постановка проблеми. В системах реального часу, крім стандартних вимог до логічної послідовності програми також, існують обмеження на час виконання та кінець виконання (абсолютний дедлайн) кожної задачі, що призводить до критичних похибок та неправильної роботи системи загалом при неправильній оцінці часових характеристик [1].

Задля успішного проектування системи реального часу потрібно вирішити цілий комплекс доволі складних задач, що у собі складаються с функціональної частини, тобто алгоритмів логіки роботи системи, вибору оптимальних технічних засобів задля вирішення поставленої задачі а також компоновка попередніх задач у єдину цілу оптимальну операційну систему реального часу [2].

Вирішення такої складної задачі з великою кількістю аспектів складається с двох основних напрямків, де першим з них є теоретичні дослідження, які спрямовані на отриманні моделі реалізації яка має у собі усі критерії що відповідають потребам часових характеристик задач у системі реального часу, а другий напрямок – це моделювання даної реалізації для перевірки «життєздатності» теоретичної гіпотези у різних середовищах та режимах роботи.

Задача першого напрямку доволі складна і не гарантує отримання точної моделі яка буде працювати в будь-яких випадках. Швидше, теоретичні дослідження лише можуть спрямувати принциповий напрямок реалізації та принципово оптимальні методи для списку часових характеристик системи реального часу. Тому цей напрямок є лише першим етапом розробки конкретної системи, і потребує подальшої роботи, тобто моделювання теоретичної гіпотези що б переконатись що вибрана модель реалізації задовольняє усі критерії здійсненності для потрібної системи реального часу. Існує декілька методів для реалізації другого етапу, тобто побудови моделі:

1) статичні моделі, вони спираються на використанні стандартних моделей для масового використання;

2) моделі побудовані за допомогою апарату сіток Петрі.

Статичні моделі доволі прості в використанні, але менш надійні, оскільки працюють в основному для середньостатистичних значень часових характеристик, що не є достатнім при розробці жорстким систем реального часу, але може бути достатнім для роботи с м'якими системами реального часу.

Розробка моделей за допомогою апарату сітки Петрі ґрунтується на ідеї розділення аналогової величини – часу виконання певної задачі на процесорі на дискретні кванти, які в свою чергу передаються між позиціями в сітці Петрі як маркери [3]. Таким чином сума квантів в певній позиції сітки Петрі є доступним процесорним часом, а інша позиція вказує на кількість часу процесора, що виділена конкретній задачі. Розподілення квантів, або маркерів, реалізована за допомогою імітації роботи певного планувальника і диспетчера операційної системи.

Аналіз останніх досліджень і публікацій. Побудову моделей на основі апарату сіток Петрі наведено у дослідженнях [4, 5]. Наприклад, у першому з них досліджено як побудувати модель завдяки розподіленню квантів процесорного часу планувальником між задачами в операційній системі реального часу, а також показано як тип планувальника та технічні характеристики процесора впливають на часові характеристики змодельованої системи реального часу.

Проте перше дослідження описує цей метод тільки для моделей що використовують один процесор.

Друге дослідження продовжує ідею цього методу і розглядає моделювання системи за допомогою апарату сіток Петрі вже багатопроцесорних систем. В ній також показано як технічні характеристики процесорів та типи планувальників впливають на характеристики побудованої моделі системи реального часу.

Однак друге дослідження може працювати лише с «плоским» списком задач, кожна з яких являється атомарною. Для широкого впровадження методу моделювання за допомогою апарату сіток Петрі необхідно проводити дослідження і моделювання систем що мають велику кількість складних задач, тобто таких що мають у собі список підзадач, кожна з яких має свої часові характеристики, які мають бути також дотримані для коректної роботи всієї системи.

Із зарубіжних авторів варто відмітити роботи таких науковців як: Блага Флорін, Поп Алін, Хуле Войчіца, Індре Клаудіу [6], Карай Мехмет, Костін Олександр [7], Сінгх Суджит, Джангід Рідді, Сінгх Гаджендра Пратап [8], Юань Фенгліан, Хуан Бо, Ван Джіпен, Пан Чунрон [9], Ло Б'янко Ріккардо, Дейкман Ремко, Нуйтен Вім, Яарсвельд Віллем [10], Каїд Хусам, Дабван Абдулмаджид, Аль-Ахмарі Абдулрахман [11] та інших.

Однак незважаючи на масштабність наукових досліджень питання актуальності даної роботи не викликає сумнівів.

Метою дослідження є моделювання, яке дозволяє використовувати багатопроцесорні та багатоядерні процесори для розробки систем, що мають складну вкладену структуру задач.

Викладення основного матеріалу дослідження. Задля використання квантів часу процесора є декілька варіантів як реалізувати розподіл:

1) статичний розподіл задач між процесорами;

2) міграція задач між процесорами;

3) міграція задачі, що зараз виконується, між процесорами.

Для першого варіанту процесори вважаються незалежними відносно один одного, тобто кожен окремий процесор моделюється окремо і має свій тип планувальника та набір часових характеристик задач. По суті, маємо одно процесорний метод що впроваджений для кожного процесору багатопроцесорної системи реального часу.

Другий варіант дає контроль над розподіленням задач між процесорами. Для наступної задачі процесор вибирається кожен раз та може змінюватися, якщо це необхідно. Проте задача не може змінювати процесор на інший, якщо вона вже почала виконуватися на цьому процесорі та має продовжувати виконуватися на тому ж самому процесорі на якому і почала виконуватися. Це означає що задача не має попередньої прив'язки до процесора, і однакові задачі можуть бути розміщені на різних процесорах. Завдяки динамічному розподілу задач між процесорами ефективність такого методу більша ніж попереднього, оскільки процесорний час використовується краще.

Третій варіант дає найбільший контроль над процесорним часом та використовує його оптимально. В додаток до попереднього виконувана задача може мігрувати між процесорами в системі навіть в самому процесі її виконання, тобто після переривання її виконання вона може відновити своє виконання вже на іншому процесорі. Звичайно, реалізація такого варіанту неможлива без додаткової апаратної підтримки. Потрібне спільне адресне середовище, завдяки якому наприклад дані задачі (її стан) будуть доступні для кожного з процесорів, щоб виконувана задача оновилася коректно на іншому процесорі, а також архітектурний контроль спільного записує/читання

з спільних даних, для уникнення проблем з одночасним читанням/записом цих самих даних.

Останній варіант є доповненням попередніх двох і тому немає сенсу розглядати його окремо у цій роботі, так як часові характеристики, методи та принципи планування відносно перших двох варіантів залишаться незмінними.

Розглянемо параметри задачі реального часу. У кожної задачі є такий список параметрів:

- r (Realize Time) – момент часу, коли задача потребує передачі управління їй;
- d (Absolute deadline) – момент часу, коли задача повинна бути завершена;
- S (Start time) – момент часу, в який задача реально почала вкионуватися на процесорі;
- C (Completion time) – момент часу, коли задача фактично завершила роботу, виконавши свою логіку;
- D (Relative Time) – відносний крайній термін: $D = d - r$;
- E (Execution Time) – час фактичного виконання задачі: $E = c - s$;
- R (Response Time) – час відгуку: $R = c - r$;
- T – період існування задач (задається при розробці системи);
- M – множина підзадач які повинні бути виконані в рамках цієї задачі. Кожна з підзадач має такий же набір характеристик, що і «кореневa» задача. Підзадача може мати пусту множину підзадач, що означає, що вона є листком (end vertex) графа задач. Параметри кожної підзадачі в множині M мають обмеження, так для кожної задачі A_k в множині M мають виконуватися обмеження:

– $d_i \le d_r$, тобто абсолютний дедлайн має бути меншим ніж у кореневої задачі;

– $S_i \ge S_r$, підзадача не може почати виконуватися раніше ніж основна задача

– $C_i \le C_r$, підзадача не може бути завершена пізніше ніж основна задача.

В усіх випадках і є індексом задачі в множині M, а індекс r позначає кореневу (основну) задачу яка володіє множиною M. Задача A_r може вважатися завершеною лише тоді коли всі задачі з множини M були завершені.

Для подальшої індексації номеру задачі а також номеру процесора який обробляє цю задачу застосуємо індексацію, де i – це номер задачі, а j – номер процесора на якому виконується задача. Тобто e_{ij} – це час який виконувалася задача A_i на процесорі j.

Сітка Петрі на рис.1 демонструє як моделюється задача A_i на процесорі j. Сукупність задач A_i, які можуть бути як список «кореневих» задач, так і список підзадач множини M, що належить задачі A_i. Кожна с цих задач отримує позицію P_{j0}. Конкретно ця модель більш детально розглянута у [4].

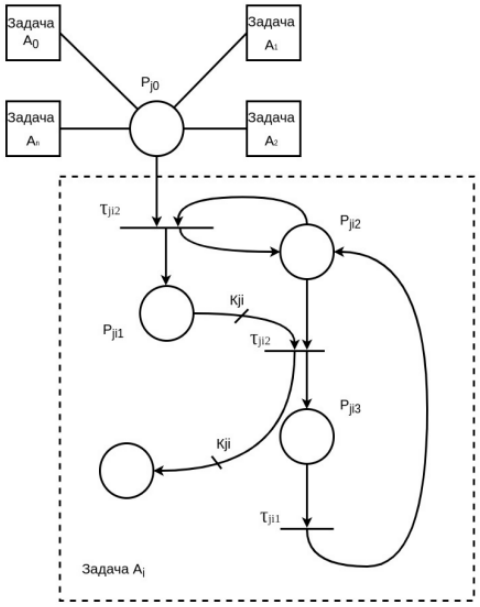

Рис. 1. Задача A_i змодельована на j-ому процесорі

Сітка Петрі – це сукупність позицій та переходів τ_{ijm}, де m=1/3. Орієнтовані дуги це переходи між позиціями та переходами.

Розглянемо детальніше що показує дана схема. Дуга, яка йде від позиції P до переходу τ ідентифікує вхід до переходу через позицію P. Дуга від переходу τ і до позиції P є вихідною дугою. Кількість дуг визначає кратність входів та виходів, в даній сітці Петрі за це відповідає k_{ji}, що знаходиться на дузі [12]. Детальніше ця сітка Петрі розглянута в роботі [5].

Для того щоб цей метод можливо було застосувати до складних задач, множина *M* яких не пуста – потрібно видозмінити генератор черги задач, який приведений як елемент на рисунку 2. Генератор черги являє собою складову багатопроцесорної моделі, що відповідає за генерацію задач в моделі багатопроцесорної системи реального часу.

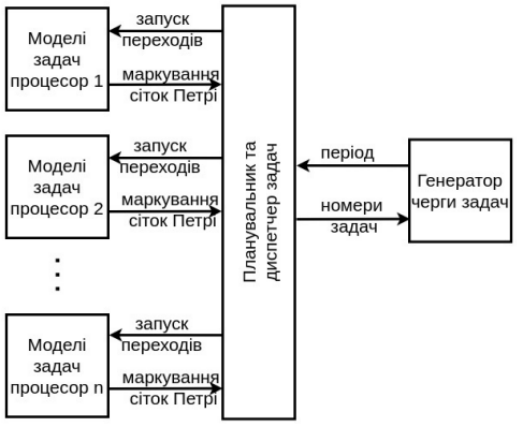

Рис. 2. Загальна схема моделювання ОСРЧ

Загалом, задачу з вкладеною множиною задач M розміткою JSON можливо описати таким чином:


```
{
 "RealizeTime": 50,
 "AbsoluteDeadline": 300,
 "StartTime": 5,
 "CompletionTime": 100,
 "T": 10000,
 "Subtasks": [
  {
   "RealizeTime": 20,
   "AbsoluteDeadline": 270,
   "StartTime": 10,
   "CompletionTime": 30,
   "T": 10000,
   "Subtasks": [
    {
     "RealizeTime": 30,
     "AbsoluteDeadline": 250,
     "StartTime": 40,
     "CompletionTime": 70,
     "T": 10000,
     "Subtasks": []
    }
   ]
  }
 ]
}
```


В даному прикладі є задача, у якої є одна підзадача також зі своєю підзадачею. Генератор черги задач в свою чергу розпаковує такий список задач у «плоский» список, де всі три задачі рівноправні і поступають у диспетчер задач у такому вигляді та мають свої незалежні часові характеристики:


```
[
 {
  "RealizeTime": 50,
  "AbsoluteDeadline": 300,
  "StartTime": 5,
  "CompletionTime": 100,
  "T": 10000
 },
 {
  "RealizeTime": 20,
  "AbsoluteDeadline": 270,
  "StartTime": 10,
  "CompletionTime": 30,
  "T": 10000
 },
 {
  "RealizeTime": 30,
  "AbsoluteDeadline": 280,
  "StartTime": 40,
  "CompletionTime": 70,
  "T": 10000
 }
]
```


За рахунок того що $d_i \le d_r$, підзадачі будуть виконані не пізніше ніж їх коренева задача, що дозволяє оперувати ними, як незалежними задачами і застосувати до них цей метод моделювання багатопроцесорних систем.

Швидкість обробки та розбиття складної задачі буде мати алгоритмічну складність $O(n + k)$, де n – кількість вершин, а k – кількість ребер дерева задач. Проте якщо порівнювати цю величину з часом розв'язання задач на процесорах при їх виконанні – ця величина буде майже непомітною, оскільки цей процес робиться лише один раз для кожної нової задачі. Для оптимізації під конкретні випадки (наприклад при великій кількості однакових задач) можливо додати кеш, у якому зберігати результати переводів попередніх задач з множиною підзадач у «плоскі» масиви з рівноправними задачами.

Висновки. Запропоноване доповнення до методу моделювання багатопроцесорних систем реального часу дозволяє значно збільшити спектр використання цього методу, не потребує додаткових архітектурних та апаратних змін і гарантує правильну роботу цього методу для більш складних сценаріїв з деревом задач.

Список літератури:

1. Комп'ютерні системи реального часу: навчальний посібник / Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського". В. Г. Зайцев, Є.І. Цибаєв. Київ, 2019. URL: https://ela.kpi.ua/handle/123456789/29604. (дата звернення: 18.08.2023).
2. Операційні системи: навчальний посібник / Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»: В. Г. Зайцев, І.П. Дробязко. Київ, 2019. URL: https://ela.kpi.ua/handle/123456789/29600. (дата звернення: 18.08.2023).
3. Супруненко О. Комбінований підхід до імітаційного моделювання динаміки програмних систем на основі інтерпретацій мереж Петрі. KPI Science News. 2021. С. 43-53.
4. Зайцев В. Г., Цибаєв Є.І. Модель оцінки часових характеристик у комп'ютерних системах реального часу з використанням сіток Петрі. *Управління розвитком складних систем*.2019. № 40. С. 76-86.
5. 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.

6. Using machine learning approaches for multi-omics data analysis: A review / P. S. Reel et al. *Biotechnology Advances*. 2021. Vol. 49. P. 107739. URL: https://doi.org/10.1016/j.biotechadv.2021.107739 (date of access: 13.08.2023).

7. Karay M., Kostin A. Using extended Petri nets for modeling and simulation of queuing systems with priorities. *International Journal of Science and Advanced Technology*. 2014. № 4. P. 1–6.

8. Singh S. K., Jangid R., Singh G. P. On characterizing binary Petri Nets. *International Journal of System Assurance Engineering and Management*. 2023. URL: https://doi.org/10.1007/s13198-023-01892-6 (date of access: 18.08.2023).

9. Yuan Feng-Lian, Huang Bo, Wang Ji-Peng, Pan Chun-Rong. A survey of modeling and scheduling of cluster tools based on Petri nets. *Acta Automatica Sinica*. 2023. № 49(5). P. 929−948 doi: 10.16383/j.aas.c210951

10. Bianco R. L., Dijkman R., Nuijten W., & van Jaarsveld W. Action-Evolution Petri Nets: A framework for modeling and solving dynamic task assignment problems. In *arXiv [cs.AI]*. 2023. http://arxiv.org/abs/2306.02910

11. Kaid Husam, Dabwan Abdulmajeed, Al-Ahmari Abdulrahman. Modeling and Simulation of Queuing Systems Using Stochastic Petri net and Arena Software: A Case Study. *8th International Conference on Industrial Engineering and Operations Managemen*t. Bandung, Indonesia. 2018. P. 26–39.

12. Fonseca João, Sousa Alexandre, Tavares José. Modeling and controlling IoT-based devices' behavior with high-level Petri nets. *Procedia Computer Science.* 2023. № 217. P. 1462-1469. Doi: 10.1016/j.procs.2022.12.345.

Skilkov N.V. ANALYSIS OF TIME CHARACTERISTICS OF TASKS IN REAL-TIME COMPUTER SYSTEMS

This work considers the problems of determining the time characteristics of tasks in real-time computer systems. It is noted that in order to design a real-time system, it is necessary to solve a whole set of rather complex problems, which consist of the functional part, the selection of optimal technical means to solve the task, as well as the arrangement of previous problems into a single optimal real-time operating system. It is emphasized that solving a holistic problem consists of two directions, theoretical justification and modeling of implementation. Implementation methods are formed on the basis of static models and models built using Petri nets. It is emphasized that the development of models using the Petri net apparatus is based on the idea of dividing an analog value - the time of execution of a certain task on the processor into discrete quanta, which in turn are transmitted between positions in the Petri net as markers. A set of parameters of the real-time task was formed and the justification of the stage of dividing the task into subtasks was carried out. The task simulated on one of the processors is graphically presented, the set of transitions is described. A general modeling scheme is proposed for a more detailed understanding of the process. A task with a nested set of tasks with JSON markup is given. It is noted that the generator of the task queue, in turn, unpacks such a list of tasks into a "flat" list, where all three tasks are equal and enter the task manager in this form and have their own independent time characteristics. It is proved that due to the fact that the subtasks will be completed no later than their root task, it allows to operate them as independent tasks and to apply the proposed method of modeling multiprocessor systems to them.

Key words: *multiprocessor system, modeling, method, time characteristic, task, subtask.*