The following research results were obtained. According to the results of the experiments, the unrolled linked list showed the best time and memory effectiveness. The B+-tree structure showed results close to a hash table. The access time to an individual element is stable in both cases, but the B+-tree has certain advantages – the elements are kept sorted, and when resizing, there is no need to expand the memory area. The worst results were shown by the data structure of the binary decision diagram, both in terms of time consumption and memory consumption. Profiling showed that 75% of the test run time for the option with an unrolled list was taken by generating random data for software simulation of agents and items of the recommendation system, therefore, the data warehouse itself has high performance indicators. Profiling of the variant with an inverted list showed that access to random blocks takes longer due to the inability to cache them, therefore, under real load conditions, the time for inserting new data will be longer, and the relative efficiency of using the inverted list will increase. For the most efficient use of memory, the block size of the linked list should be adapted so that the blocks are as full as possible. Small blocks reduce memory waste, but increase the time to traverse all the elements of the list and increase memory overhead.
recommendation systems, databases, data structures, computer simulation model

Одержано (Received) 10.04.2021 *Прорецензовано (Reviewed) 18.04.2021*
Прийнято до друку (Approved) 26.04.2021

УДК 004.8/681.5 DOI: https://doi.org/10.32515/2664-262X.2021.4(35).16-23

Р.М. Минайленко, доц., канд. техн. наук, **О.Г. Собінов,** викл., **О.К. Конопліцька-Слободенюк,** викл., **К.О. Буравченко**, канд. техн.наук
Центральноукраїнський національний технічний університет, Кропивницький, Україна
e-mail: aron70@ukr.net, sagcob14@gmail.com

Архітектурні особливості систем розподілених обчислень

В статті проведено аналіз архітектурних особливостей систем розподілених обчислень Головним завданням, яке вирішують технології розподілених обчислень є забезпечення доступу до глобально розподілених ресурсів і вирішення задач, що потребують значних обчислювальних потужностей та не можуть бути реалізовані на звичайному комп'ютері. Складність реалізації глобальних задач обумовлена тим, що доступ до необхідних даних може відбуватись на різних комп'ютерах. Крім того, розподілені обчислювальні системи, які формуються із автономних ресурсів, можуть змінювати свою архітектуру динамічно. Керування такими розподіленими обчислювальними системами потребує пошуку нових моделей обчислення і пошуку архітектурних рішень для побудови нових систем які б відповідали сучасному рівню розвитку інформаційних технологій.
комп'ютер, розподілені обчислення, інформаційні технології, архітектурні особливості

Постановка проблеми. Останнім часом спостерігається все більше проникнення інформаційних технологій майже у всі галузі життєдіяльності людства. Розвиток інформаційних технологій пов'язаний з виникненням нових задач, що потребують значних обчислювальних ресурсів і не можуть бути вирішені на звичайному комп'ютері.

Великий об'єм обчислень потребує створення, так званих, суперкомп'ютерів, що реалізувати технічно не завжди можливо. Та існує і інший спосіб вирішення вказаної проблеми, коли складна задача розподіляється на певну кількість підзадач, що виконуються паралельно. І тут стають у пригоді системи розподіленого обчислення.

В загальному випадку, системою розподілених обчислень є віртуальний комп'ютер, який складається з декількох вузлів об'єднаних мережею. Тобто певна об'ємна задача розбивається на декілька простіших підзадач і встановлюються зв'язки між ними. Але така система буде працездатною тільки тоді, коли завдання між вузлами будуть розподілені коректно, а послідовність їх виконання відбуватиметься згідно з заданим алгоритмом.

Аналіз останніх досліджень і публікацій. Існуючі архітектурні конфігурації розподілених обчислювальних систем можна класифікувати за різними ознаками:

– з огляду на будову алгоритму складових, які приймають участь в обчисленнях;

– моделей додатків;

– обмежень якості обслуговування та інших.

На верхньому рівні ієрархії – статичні та динамічні алгоритми плануванні ресурсів. Статичне планування та розрахунок вартісного оцінювання обчислень відбувається до виконання завдання, коли інформація відносно всіх ресурсів в розподіленому обчислювальному середовищі є відомою [1–3] . Головною перевагою статичної моделі є відносно не складна реалізація планувальника. Але вартісна оцінка, яка базується на статичній інформації, погано адаптується до ситуацій, пов'язаних з виходом із ладу одного з обчислювальних вузлів. Тому для вирішення проблеми використовуються механізми перепланування [4].

Динамічне планування частіше всього застосовується тоді, коли потрібно зробити оцінку обчислювальної вартості додатка, що надходить на виконання динамічно в режимі реального часу. Динамічне планування містить в собі два важливих компонента – оцінка стану системи та прийняття рішення про взаємодію завдання із черги з потрібним вибраним ресурсом [5–7] .

Алгоритми динамічного планування представлені в роботі [8] присвячені випадку резервування ресурсів, що часто використовується в розподілених обчисленнях, для отримання деякої гарантії стабільності виробничих ресурсів. При використанні динамічних сценаріїв планування відповідальним за прийняття глобальних рішень може бути один централізований планувальник або декілька розподілених. Використання централізованого планувальника має перевагу за рахунок простоти реалізації, але є і недоліки: недостатнє масштабування та невисока відмовостійкість.

Субоптимальні алгоритми планування можна розділити на наближені, які використовують формальні обчислювальні моделі та евристичні алгоритми, які дають більш реальні дані про навантаження системи та виконання завдання.

Розподілені алгоритми планування, в залежності від того як працюють вузли, що використовуються в процесі планування незалежно чи сумісно, розділяють на зв'язні і незв'язні. У випадку незалежного планування локальний планувальник працює автономно і приймає рішення з урахуванням особливостей своїх функцій. У випадку сумісного планування кожен планувальник відповідає за виконання власної частини завдання, але всі планувальники працюють над виконанням спільного завдання[9].

Тобто архітектура тієї чи іншої системи розподілених обчислень залежить від того, які завдання вона повинна вирішувати.

Постановка завдання. На теперішній час для вдосконалення і спрощення процесу управління і організації систем розподілених обчислень існує велика кількість відкритих і комерційних програмних продуктів[9-15]. Але існує цілий ряд проблем які вирішені частково, або потребують певного вдосконалення[16-20]. До таких можна віднести:

1. Повнота інфраструктури системи.

2. Надання спрощеного розширення системи.
3. Автоматичне розгортання системи без вимкнення.
4. Можливість спрощеного способу доповнення систем.
5. Коректування навантаження системи в процесі використання.
6. Вдосконалення інфраструктури систем.

Тому аналіз існуючих методик і способів побудови систем розподілених обчислень і виявлення їх переваг та недоліків є актуальним завданням.

Виклад основного матеріалу. Особливістю систем з розподіленими обчисленнями, в порівнянні з суперкомп'ютерами, є можливість нарощування продуктивності. Такі системи можна класифікувати за наступними ознаками як:

– однорідність компонентів (однорідні, неоднорідні);

– рівень розв'язності компонентів (не дуже зв'язані, дуже зв'язані);

– організація (централізовані, децентралізовані, кластеризовані);

Проведемо аналіз архітектурних особливостей систем означених вище. На рис.1 представлено структурну схему централізованої системи розподілених обчислень:

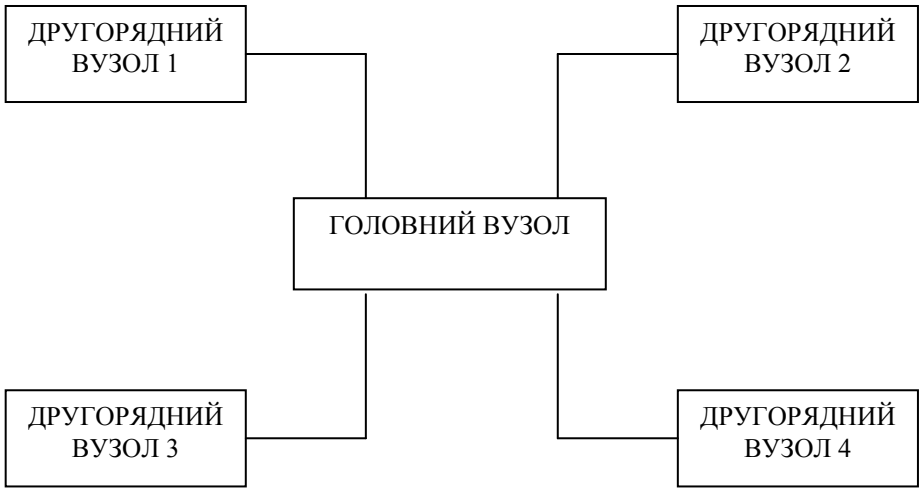

Рисунок 1 – Структурна схема централізованої системи розподілених обчислень
Джерело: [1]

Централізовані системи розподілених обчислень складаються із головного вузла та другорядних вузлів. Головний вузол проводить розподіл задач між другорядними вузлами, контролює пріоритетність та процес виконання задачі. Крім того, головний вузол виконує функції приймання завдань та надає можливість зовнішнього впливу на систему. Розподілені системи з такою архітектурою добре пристосовані до нарощення продуктивності в процесі роботи і забезпечують якісний контроль процесу виконання завдання. При виникненні збоїв в роботі другорядного вузла, така система буде працездатною, хоча буде функціонувати з меншою продуктивністю. Виникненні збоїв у головному вузлі призведе до повної зупинки системи. Крім того, головний вузол контролює всю інформацію, що обробляється другорядними вузлами, що приводить до затримки її передачі. Тому такі системи розподілених обчислень доцільно застосовувати для вирішення завдань в яких час, витрачений на обмін інформацією не є пріоритетом.

Різновидом централізованої системи розподілених обчислень є кластеризовані системи де існує поняття кластеру, роль якого виконує певна частина централізованої системи розподілених обчислень, призначення для виконання обмеженої кількості завдань. Перевагою такої архітектури є раціональне використання доступних ресурсів.

Системам розподілу з кластеризованою архітектурою притаманна більша гнучкість, відмовостійкість та швидкість кінцевої відповіді в порівнянні із системами централізованого розподілу обчислень. Але такі системи потребують певного часу на пошук недозавантажених кластерів.

Структурна схема системи розподілених обчислень з кластеризованою архітектурою представлена на рис.2:

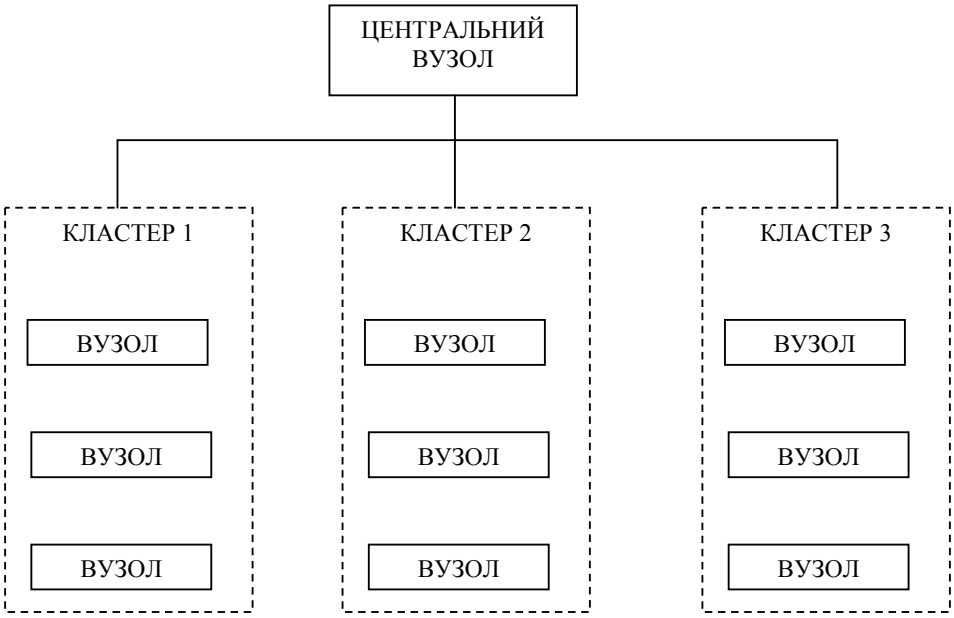

Рисунок 2 – Структурна схема системи розподілених обчислень кластеризованою архітектурою
Джерело: [13]

В децентралізованих системах розподілених обчислень функцію приймання завдань та зв'язок і зовнішнім середовищем може виконувати будь–який вузол. На рис.3 представлено структурну схему децентралізованої системи розподілених обчислень:

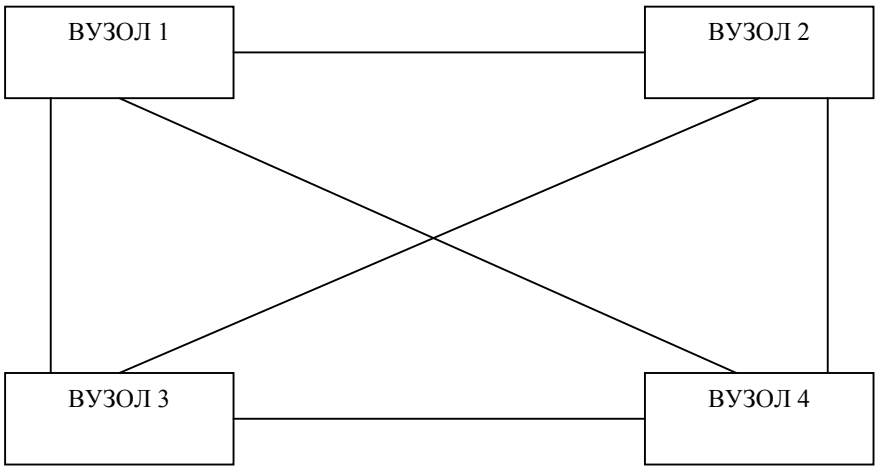

Рисунок 3 – Структурна схема децентралізованої системи розподілених обчислень
Джерело: [18]

При надходженні завдання до певного вузла, відбувається розподіл на менші підзавдання між іншими вузлами. В результаті такого розподілу завдання, при

додаванні нових вузлів, виникають часові затримки пов'язані з ідентифікацією нового вузла сусідніми. В таких системах швидкість відповіді максимальна, а рівень зовнішнього контролю за ходом обчислень нижчий ніж у попередньої системи. Такі системи знайшли своє застосування при вирішенні певних однорідних задач з великим масивом даних де потрібна миттєва реакція на запит.

В розподілених системах із слабкими зв'язками кожен вузол є окремим незалежним компонентом і не використовує спільні ресурси, що значно впливає на надійність (відмовостійкість) системи і окремих вузлів.

В сильнозв'язаних системах вузли використовують спільні ресурси, що значно спрощує створення таких систем і зменшує час обробки завдань якщо їх небагато. Але користування спільними ресурсами знижує стабільність та відмовостійкість таких систем, а збільшення навантаження призводить до зменшення продуктивності, оскільки спільний ресурс не має змоги оперативно обслуговувати всі незалежні вузли.

Однорідні розподілені обчислювальні системи характеризуються тим, що всі вузли таких систем мають однорідні ресурси. Такі системи знаходять застосування у випадку, коли в одному завданні всі задачі використовують обмежену кількість алгоритмів ресурси яких однакові.

Неоднорідні системи складаються з вузлів різної конфігурації і доступ до ресурсів теж різний. Частіше всього такі системи є кластеризованими і кожен кластер спеціалізується на виконанні певних визначених задач. Основною перевагою неоднорідних розподілених систем є більш ефективне (в порівнянні з однорідними системами) використання ресурсів, оскільки для кожної вирішуваної задачі використовуються найбільш пристосовані, для задач певного типу, обчислювальні потужності. Основним недоліком таких обчислювальних систем є їх централізація.

Висновки. В статті проведено аналіз архітектурних особливостей систем розподілених обчислень Головним завданням, яке вирішують технології розподілених обчислень є забезпечення доступу до глобально розподілених ресурсів і вирішення задач, що потребують значних обчислювальних потужностей та не можуть бути реалізовані на звичайному комп'ютері. Складність реалізації глобальних задач обумовлена тим, що доступ до необхідних даних може відбуватись на різних комп'ютерах. Крім того, розподілені обчислювальні системи, які формуються із автономних ресурсів, можуть змінювати свою архітектуру динамічно. Керування такими розподіленими обчислювальними системами потребує пошуку нових моделей обчислення і пошуку архітектурних рішень для побудови нових систем які б відповідали сучасному рівню розвитку інформаційних технологій.

Список літератури

1. Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems / R. Braun, H. Siegel et al. *Parallel and Distributed Computing*. 2001. Vol. 61, No. 6. P. 810–837.
2. Heuristics for Scheduling Parameter Sweep Applications in Grid Environ-ments / H. Casanova, A. Legrand et al. *Heterogeneous Computing Workshop (HCW'00):* Proceedings of the 9th Workshop (Cancun, Mexico, May 1, 2000). IEEE Computer Society, 2000. P. 349–363.
3. You, S.Y. Task Scheduling Algorithm in GRID Considering Heterogeneous Environment / S.Y. You, H.Y. Kim et al. *Parallel and Distributed Processing Techniques and Ap-plications (PDPTA '04)*: Proceedings of the International Conference (Nevada, USA, June 21–24, 2004). CSREA Press, 2004. Vol. 1. P. 240–245.
4. Cooper, K. New Grid Scheduling and Rescheduling Methods in the GrADS Project /Cooper, A. Dasgupta et al. *International Parallel and Distributed Processing Sym-posium (IPDPS'04)*: Proceedings of the 18th

International Symposium (Santa Fe, New Mexico USA, April 26–30, 2004). IEEE Computer Society, 2004. P. 199–206.

5. Improving Grid Level Throughput Using Job Migration And Rescheduling / K. Kurowski, B. Ludwiczak et al. *Scientific Programming*. 2004. Vol. 12, No. 4. P. 263–273.
6. Takefusa, A. A Study of Deadline Scheduling for Client-Server Systems on the Computa-tional Grid / A. Takefusa, S. Matsuoka et al. *High Performance Distributed Computing (HPDC-10)*: Proceedings of the 10th IEEE International Symposium (San Francisco, Cal-ifornia, USA, August 7–9, 2001). IEEE Computer Society, 2001. P. 406–415.
7. Chen, H., Maheswaran M. Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems . *International Parallel and Distributed Processing Symposium (IPDPS 2002)*: Proceedings of the 16th International Symposium (Fort Lauderdale, FL, USA, April 15-19, 2002). IEEE Computer Society, 2002. P. 88–97.
8. Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids / N. Muthuvelu, J. Liu et al. *Grid Computing and e-Research (AusGrid 2005)*: Proceedings of the 3rd Australasian Workshop (Newcas-tle, NSW, Australia, January 30 – February 4, 2005). Australian Computer Society, 2005. P. 41–48.
9. Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration / H. Shan, L. Oliker et al. *Advanced Computing and Communication (ADCOM 2004)*: Proceedings of the 12th IEEE International Conference (Ahmedabad Gujarat, India, De-cember 15–18, 2004). IEEE Computer Society, 2004. P. 1–8.
10. Dong, F., Akl S.G. Scheduling algorithms for grid computing: State of the art and open problems. Technical Report No. 2006-504 . Queen's University, Canada, 2006. P. 55.
11. Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests / V. Subramani, R. Kettimuthu et al. *High Performance Distributed Computing (HPDC 2002)*: Proceedings of 11th IEEE Symposium (Edinburgh, Scotland, July 23–26, 2002). IEEE Computer Society, 2002. P. 359–366.
12. El-Rewini, H. Task Scheduling in Parallel and Distributed Systems / H. El-Rewini, T. Lewis, H. Ali — Prentice Hall, 2010. 290 p.
13. Radulescu, A., Gemund A.J.C. On the Complexity of List Scheduling Algorithms for Distributed Memory Systems . *Supercomputing (SC'99)*: Proceedings of 13th International Conference (Portland, Oregon, USA, November 13–19, 1999). IEEE Computer Society, 1999. P. 68–75.
14. Sakellariou, R., Zhao H. A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems . *Scientific Programming*. 2017. Vol. 12, No. 4. P. 253–262.
15. Darbha, S., Agrawal, D.P. Optimal Scheduling Algorithm for Distributed Memory Machines. *IEEE Transactions on Parallel and Distributed Systems*. 1998. Vol. 9, No. 1. P. 87–95.
16. Ranaweera, S., Agrawal D.P. A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems. *International Parallel and Distributed Processing Sym-posium (IPDPS'00)*: Proceedings of 14TH International Symposium (Cancun, Mexico, May 1–5, 2018). IEEE Computer Society, 2005. P. 445–450.
17. Bajaj, R., Agrawal, D.P. Improving Scheduling of Tasks in A Heterogeneous Environment . *IEEE Transactions on Parallel and Distributed Systems*. 2004. Vol. 15, No. 2. P. 107–118.
18. Yang, T., Gerasoulis, A. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors. *EEE Transactions on Parallel and Distributed Systems*. 1994. Vol. 5, No. 9. P. 951–967.
19. Liou, J., Palis, M.A. A Comparison of General Approaches to Multiprocessor Scheduling . *International Parallel Processing Symposium (IPPS '97):* Proceedings the 11th International Symposium (Geneva, Switzerland, April 1–5, 1997). IEEE Computer Society, 1996. P. 152–156.

References

1. Braun R., Siegel H. et al. (2001). Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems .*Parallel and Distributed Computing. Vol. 61, No. 6*. P. 810–837[in English].

2. H. Casanova, A. Legrand et al. (2000). Heuristics for Scheduling Parameter Sweep Applications in Grid Environ-ments. *Heterogeneous Computing Workshop (HCW'00)*: Proceedings of the 9th Workshop (Cancun, Mexico, May 1, 2000). IEEE Computer Society, P. 349–363 [in English].
3. You S.Y., Kim H.Y. et al. (2004). Task Scheduling Algorithm in GRID Considering Heterogeneous Environment. *Parallel and Distributed Processing Techniques and Ap-plications (PDPTA '04)*: Proceedings of the International Conference (Nevada, USA, June 21–24, 2004). CSREA Press. *Vol. 1*. P. 240–245 [in English].
4. Cooper, K., Dasgupta A. et al. (2004). New Grid Scheduling and Rescheduling Methods in the GrADS Project . *International Parallel and Distributed Processing Sym-posium (IPDPS'04)*: Proceedings of the 18th International Symposium (Santa Fe, New Mexico USA, April 26–30, 2004). IEEE Computer Society, P. 199–206 [in English].
5. Kurowski K., Ludwiczak B. et al. (2004). Improving Grid Level Throughput Using Job Migration And Rescheduling / *Scientific Programming*. *Vol. 12, No. 4*. P. 263–273 [in English].
6. Takefusa, A., Matsuoka S. et al. (2001). A Study of Deadline Scheduling for Client-Server Systems on the Computa-tional Grid . *High Performance Distributed Computing (HPDC-10)*: Proceedings of the 10th IEEE International Symposium (San Francisco, Cal-ifornia, USA, August 7–9, 2001). IEEE Computer Society, P. 406–415 [in English].
7. Chen, H. & Maheswaran, M. (2002). Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Sys-tems . *International Parallel and Distributed Processing Symposium (IPDPS 2002)*: Proceedings of the 16th International Symposium (Fort Lauderdale, FL, USA, April 15-19, 2002). IEEE Computer Society, P. 88–97 [in English].
8. Muthuvelu, N., Liu, J. et al. (2005). Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids . *Grid Computing and e-Research (AusGrid 2005)*: Proceedings of the 3rd Australasian Workshop (Newcas-tle, NSW, Australia, January 30 – February 4, 2005). Australian Computer Society, P. 41–48 [in English].
9. Shan H., Oliker L. et al. (2004). Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration *Advanced Computing and Communication (ADCOM 2004)*: Proceedings of the 12th IEEE International Conference (Ahmedabad Gujarat, India, De-cember 15–18, 2004). IEEE Computer Society, P. 1–8 [in English].
10. Dong, F. & Akl, S.G. (2006). Scheduling algorithms for grid computing: State of the art and open problems. Technical Report No. 2006-504 . Queen's University, Canada, P. 55 [in English].
11. Subramani, V., Kettimuthu, R. et al. (2002). Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests . *High Performance Distributed Computing (HPDC 2002)*: Proceedings of 11th IEEE Symposium (Edinburgh, Scotland, July 23–26, 2002). IEEE Computer Society, P. 359–366 [in English].
12. El-Rewini, H. & Lewis, T. (2010). Task Scheduling in Parallel and Distributed Systems. H. Ali - Prentice Hall, 290 p. [in English].
13. Radulescu, A. & Gemund, A.J.C. (1999). On the Complexity of List Scheduling Algorithms for Distributed Memory Systems . *Supercomputing (SC'99)*: Proceedings of 13th International Conference (Portland, Oregon, USA, November 13–19, 1999). IEEE Computer Society, P. 68–75 [in English].
14. Sakellariou, R &, Zhao, H. (2017). A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems . *Scientific Programming*. *Vol. 12, No. 4.* P. 253–262 [in English].
15. Darbha, S. & Agrawal, D.P. (1998). Optimal Scheduling Algorithm for Distributed Memory Machines. *IEEE Transactions on Parallel and Distributed Systems*. *Vol. 9, No. 1.* P. 87–95 [in English].
16. Ranaweera, S. & Agrawal, D.P. (2005). A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems. *International Parallel and Distributed Processing Sym-posium (IPDPS'00)*: Proceedings of 14TH International Symposium (Cancun, Mexico, May 1–5, 2018). IEEE Computer Society, P. 445–450 [in English].
17. Bajaj, R. & Agrawal, D.P. (2004). Improving Scheduling of Tasks in A Heterogeneous Environment . *IEEE Transactions on Parallel and Distributed Systems*. *Vol. 15, No. 2.* P. 107–118 [in English].
18. Yang, T. & Gerasoulis, A. (1994). DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors. *EEE Transactions on Parallel and Distributed Systems*. *Vol. 5, No. 9.* P. 951–967 [in English].

19. Liou, J. & Palis, M.A. (1996). A Comparison of General Approaches to Multiprocessor Scheduling . *International Parallel Processing Symposium (IPPS '97):* Proceedings the 11th International Symposium (Geneva, Switzerland, April 1–5, 1997). IEEE Computer Society, P. 152–156 [in English].

Roman Minailenko, Assoc. Prof., PhD tech. sci., **Olexandr Sobinov,** lecturer**, Oksana Konoplitska-Slobodenyuk,** lecturer, **Kostiantyn Buravchenko**, PhD tech. sci.
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine

Architectural Features of Distributed Computing Systems

Recently, there has been an increasing penetration of information technology in almost all areas of human life. The development of information technology is associated with the emergence of new tasks that require significant computing resources and can not be solved on a conventional computer.

A large amount of computing requires the creation of so-called supercomputers, which is not always technically possible. But there is another way to solve this problem, when a complex task is divided into a number of subtasks that run in parallel. And here come in handy distributed computing system. In general, a distributed computing system is a virtual machine that consists of several nodes connected by a network. That is, a certain three-dimensional problem is divided into several simple subtasks and connections are established between them. But such a system will be operational only when the tasks between the nodes are distributed correctly, and the sequence of their execution will take place according to a given algorithm.

The article analyzes the architectural features of distributed computing systems. The main task of distributed computing technologies is to provide access to globally distributed resources and solve problems that require significant computing power and can not be implemented on a conventional computer. The complexity of global tasks is due to the fact that the necessary data can be accessed on different computers. In addition, distributed computing systems, which are formed from autonomous resources, can change their architecture dynamically. Management of such distributed computer systems requires the search for new computational models and the search for architectural solutions to build new systems that would meet the current level of development of information technology.

computer, distributed computing, information technology, architectural features

Одержано (Received) 27.01.2021 *Прорецензовано (Reviewed) 19.02.2021*
Прийнято до друку (Approved) 26.04.2021