DOI: <https://doi.org/10.15407/emodel.43.02.086>
УДК 519.876

Т.А. Узденов, аспірант
Державний університет «Житомирська політехніка»
Україна, 10005, Житомир, вул. Чуднівська, 103,
тел. (063) 2823417; e-mail: uzdenov.taras@gmail.com

Зменшення часу виконання черги завдань у GRID-системах з невідчужуваними ресурсами

Наведено математичну постановку задачі диспетчеризації потоків завдань в GRID-системах з невідчужуваними ресурсами з врахуванням потужності вузла та потужності задачі як ключових факторів, що впливають на продуктивність системи. Проведено порівняння часу виконання черги завдань при розподілі методами FSA (Flow Scheduling Algorithm), FSA_P (Flow Scheduling Algorithm Parallel) та FCFS (First Come First Serve). Описано клієнт-серверну архітектурну модель побудови програмного забезпечення для розподілених обчислень та задач, які потребують великої обчислювальної потужності системи. Обґрунтовано доцільність порівняння ефективності запропонованих методів з загальновідомим методом FCFS, який зазвичай використовується у GRID-системах. Подано результати тестування, яке засвідчило, що запропоновані методи дають кращий результат, ніж FCFS.

К л ю ч о в і с л о в а: методи диспетчеризації, GRID-система, симулятор GRID, розподілені обчислення, планування потоків.

Однією з проблем, що потребує вирішення при розробці програмного забезпечення для GRID-системи з невідчужуваними ресурсами, є задача диспетчеризації потоків завдань. Вона досить складна і наразі не існує чіткого і однозначного її вирішення [1, 2]. Аналіз існуючих методів диспетчеризації засвідчив, що було запропоновано чимало нових методів для вирішення даної задачі, але наразі в реальних системах зазвичай використовується лише метод FCFS[3].

Це пов'язано з тим, що даний метод є дуже простим та надійним як в розробці так і в роботі. Використання інших методів значно ускладнює систему, що робить її менш надійною. Оскільки такі системи є досить нестабільними, зрозуміло, чому розробники відмовляються від складних методів і віддають перевагу FCFS. Це дає підстави зробити висновок про

© Узденов Т.А., 2021

необхідність розробки нових методів, основними характеристиками яких повинні бути простота і краща продуктивність у порівнянні з FCFS.

Процес диспетчеризації, алгоритми планування та опис простого підходу і методи, розроблені на його основі, наведено у роботах [4, 5]. Подамо результати дослідження їх ефективності у порівнянні з методом FCFS.

Постановка задачі. Нехай є GRID-система з N процесами та M потоками. Під процесом будемо розуміти обчислювальний елемент, а під потоком — обчислювальну задачу. Введемо поняття потужності процесу P_n , тобто обчислювального елемента, та потужності потоку R_m , тобто задачі. Отже, маємо множину потужності процесів $P = \{P_1, P_2, P_3, \dots, P_n\}$, де $n = (1, N)$, та множину потужності потоків $R = \{R_1, R_2, R_3, \dots, R_m\}$, де $m = (1, M)$. Потрібно оптимально розподілити потоки по процесам.

Для цього введемо поняття балансу сил системи B , беручи до уваги поняття балансу сил, тобто коефіцієнт, при якому потужність i -го потоку дорівнює потужності j -го процесу. Тоді $B = P_i / R_j \approx 1$, а для ідеального варіанту $P_i / R_j = 1$.

Даний метод отримав назву алгоритм диспетчеризації потоків (Flow Scheduling Algorithm (FSA)), або алгоритм балансу сил (Forces Balance Algorithm (FBA)).

А л г о р и т м 1.

1. Обчислюємо:

$$P = \{P_1, P_2, P_3, \dots, P_n\}, n = (1, N),$$

$$R = \{R_1, R_2, R_3, \dots, R_m\}, \text{ де } m = 1, M.$$

2. Встановлюємо: $B=1$.

3. Обираємо i потік, у якому $I = (1, M)$.

4. Знаходимо пару $i - j$, для якої справедливий вираз $B = P_i / R_j \approx 1$.

5. Відправляємо i потік на j процес.

6. Обчислюємо потужності без P_i та R_j .

7. Якщо є невідправлені задачі, то переходимо до п. 3.

8. Завершення.

Ще два запропоновані методи було визначено експериментально. Вони основані на поєднанні з методом FSA методу Min-Min [6] та методу Max-Min [7]. Різниця між ними полягає в тому, що на початковій стадії розподілу відповідно до методу FSAMin існують пари $i - j$, для яких $P_i = \min\{P_1, P_m\}$, $P_j = \min\{P_1, P_n\}$ та $P_i = \max\{P_1, P_m\}$, а відповідно до методу FSAMax $P_j = \max\{P_1, P_n\}$. Далі розподіл відбувається відповідно до методу FSA.

Зазначені методи розроблено для черги послідовних задач. Якщо в черзі стоять задачі, які можна легко розпаралелювати, то розподіл можна

Рис. 1. Архітектурна модель програмного забезпечення: 1—5 — користувачі; WCF — платформа для створення сервіс-орієнтовних програм

робити за методом, що отримав назву алгоритм диспетчеризації потоків паралельний (Flow Scheduling Algorithm Parallel (FSA_P)).

А л г о р и т м 2.

1. Обчислюємо:

$$P = (P_1, P_2, P_3, \dots, P_n),$$

де $n = (1, N)$, $\sum P$.

2. Обираємо V_i , де V_i — об'єм i -ї задачі з множини $V = \{V_1, V_2, \dots, V_m\}$, відповідно до порядку в черзі.

3. Для всіх $j \in (1, N)$ знаходимо $V_{ij} = V_i P_j / \sum P$.

4. Відправляємо на виконання v_{ij} .

5. Якщо в системі є нерозподілені задачі, переходимо до п. 2.

6. Завершення.

Далі будемо розглядати такі основні випадки початкового стану системи:

кількість потоків перевищує кількість процесів ($M > N$). Розподіляємо N потоків, а решта ($M - N$) чекають своєї черги, доки не звільниться i -й процес, для якого вибираємо задачу P_i з множини ($M - N$) з відповідним значенням P_j ;

кількість потоків менше кількості процесів ($M < N$). Обираємо N процесів з найбільшою потужністю, а решта ($N - M$) чекають, доки на вхід не поступить новий потік;

кількість потоків дорівнює кількості процесів ($M = N$). Розподіляємо потоки за алгоритмом 2.

Для створення програмного забезпечення, яке дозволяє тестувати та досліджувати ефективність запропонованих методів, розроблено його клієнт-серверну архітектурну модель (рис. 1). Дану модель побудовано на основі WCF сервісу [8] для програмного забезпечення, потребуючого розподілених обчислень в комп'ютерних мережах та мережі Internet, а також для створення GRID-систем з невідчужуваними ресурсами. На основі даної моделі розроблено програмний комплекс для симуляції роботи GRID-системи з невідчужуваними ресурсами та дослідження алгоритмів диспетчеризації [9].

Таблиця 1. Результати тестування за тестом 1

Середня потужність системи	Метод							
	Паралельний			Послідовний				
	ETALON_P	FCFS_P	FSA_P	ETALON	FCFS	FSA	FSAMin	FSAMax
0,1	7 638	67 238	67 454	9 069	90 066	90 035	90 076	90 058
0,15	7 813	67 012	45 530	9 048	90 057	70 180	70 200	70 214
0,2	8 007	67 107	34 301	9 045	90 055	70 178	70 182	70 046
0,25	7 987	66 945	27 957	9 061	70 067	30 197	30 177	30 168
0,3	8 078	66 879	23 508	9 062	80 029	60 074	60 049	60 037
0,35	7 828	66 591	20 208	9 051	70 060	30 074	30 081	30 063
0,4	7 990	67 129	17 781	9 061	70 030	30 193	30 224	30 295
0,45	8 045	22 977	16 113	9 052	30 045	23 382	23 533	23 505
0,5	8 046	66 828	14 495	9 024	70 034	30 207	30 162	30 204
0,55	7 730	66 541	13 420	9 078	70 051	30 040	30 071	30 068
0,6	7 897	33 893	12 296	9 057	40 090	30 058	30 056	30 073
0,65	7 919	22 923	11 550	9 059	26 714	23 411	23 393	23 377
0,7	7 909	66 570	10 816	9 024	70 048	30 062	30 067	30 073
0,75	7 987	33 998	10 171	9 051	35 049	15 156	15 147	15 191
0,8	7 913	33 962	9 613	9 061	35 052	15 183	15 190	15 160
0,85	7 883	33 593	9 185	9 042	35 043	15 062	15 036	15 057
0,9	7 988	66 499	8 639	9 066	70 039	30 064	30 057	30 113
0,95	8 037	8 410	8 626	9 068	10 058	10 063	9 079	9 059
1	7 934	7 988	7 834	9 057	9 072	9 062	9 081	9 068

Рис. 2. Діаграми залежності часу виконання черги завдань від середньої потужності системи за тестом 1: *a* — ■ ETALON_P; ♦ FCFS_P; ▲ FSA_P — паралельні методи; *б* — ■ ETALON; ♦ FCFS; ▲ FSA — послідовні методи; *в* — ■ FSA; ♦ FSAmin; ▲ FSAmax — запропоновані методи

З використанням розробленого симулятора було проведено ряд тестів. Для визначення їх ефективності обрано метод диспетчеризації FCFS, тому що він найчастіше застосовується для GRID-систем з невідчуваними ресурсами. Крім того, інші методи неможливо використовувати через те, що в даному випадку вони не передбачають зміни потужності вузлів, а інші характеристики, такі, наприклад, як квант часу чи пріоритети, було вирішено не вводити.

В системі реалізовано механізм тестування, який передбачає виконання всіх завдань, що згенерувала система, використовуючи різні алгоритми розподілу завдань між вузлами. Тобто спочатку розподіляються завдання за алгоритмом 1, потім за алгоритмом 2 і так до останнього завдання.

В систему заведено наступні паралельні та послідовні методи диспетчеризації: ETALON_P; FCFS_P; FSA_P; ETALON; FSA, FSA_Min; FSA_Max.

П а р а л е л ь н і м е т о д и :

ETALON_P — метод взято за еталонний. Це метод FCFS, в якому не враховуються потужності персонального комп'ютера (ПК) та задач.

Рис. 3. Діаграми залежності часу виконання черги завдань від середньої потужності системи за тестом 1: *a* — ■ ETALON_P; ♦ FCFS_P; ▲ FSA_P — паралельні методи; *b* — ■ ETALON; ♦ FCFS; ▲ FSA — послідовні методи; *c* — ▨ FSA; ▩ FSAMin; — FSAMax — запропоновані методи

Рис. 4. Діаграми залежності часу виконання черги завдань від середньої потужності системи за тестом 1: *a* — ■ ETALON_P; ♦ FCFS_P; ▲ FSA_P — паралельні методи; *b* — ■ ETALON; ♦ FCFS; ▲ FSA — послідовні методи; *c* — ■ FSA; ♦ FSAMin; ▲ FSAMax — запропоновані методи

Моделюється система, в якій всі вузли однакові за потужністю. Можливі паралельні завдання.

FCFS_P — метод FCFS, в якому враховано потужності ПК і можливість розподіляти одне завдання на декілька ПК.

FSA_P — запропонований метод, в якому враховано потужності ПК і можливість розподіляти одне завдання на декілька ПК.

П о с л і д о в н і м е т о д и:

ETALON — метод взято за еталонний. Це метод FCFS, в якому не враховуються потужності ПК і завдання не можна розподіляти.

FCFS — метод, в якому враховано потужності ПК і завдання не можна розподіляти між вузлами.

FSA, FSA_Min, FSA_Max — запропоновані методи, в яких враховано потужності ПК і завдання не можна розподіляти.

Таблиця 2. Результати тестування за тестом 2

Середня потужність системи	Метод							
	Паралельний			Послідовний				
	ETALON_P	FCFS_P	FSA_P	ETALON	FCFS	FSA	FSAMin	FSAMax
0,1	15 596	60 829	61 130	9 201	90 205	90 232	90 329	90 230
0,15	16 329	57 489	43 400	9 215	90 244	70 264	50 245	50 213
0,2	15 507	57 884	34 756	9 266	90 206	70 217	50 221	50 225
0,25	16 383	56 309	29 947	9 235	80 083	70 115	30 311	30 228
0,30	12 518	56 219	26 654	9 217	80 092	70 274	30 241	30 186
0,35	15 357	56 024	23 429	9 244	70 068	60 047	22 747	22 728
0,40	16 304	56 245	21 471	9 225	80 066	23 435	20 205	20 263
0,45	11 473	21 865	18 226	9 201	30 205	26 746	20 243	20 246
0,50	11 440	56 219	17 046	9 221	80 047	20 226	20 218	20 243
0,55	12 348	55 744	16 305	9 200	70 067	20 095	20 083	20 079
0,60	11 967	29 690	15 525	9 221	40 030	15 232	15 235	15 252
0,65	11 594	21 639	14 833	9 212	30 216	16 732	16 738	16 772
0,70	11 351	55 989	14 455	9 200	70 095	20 076	20 076	20 069
0,75	12 017	29 165	13 529	9 212	40 079	10 202	10 212	10 224
0,80	11 469	29 021	12 857	9 213	40 090	10 224	10 203	10 207
0,85	11 952	29 236	12 578	9 225	35 118	10 199	10 062	10 082
0,90	11 482	55 985	12 488	9 201	70 077	20 087	20 059	20 102
0,95	11 766	11 332	11 454	9 233	10 190	10 248	9 242	9 202
1	10 938	12 826	12 859	9 222	9 248	9 228	9 252	9 221

Таблиця 3. Результати тестування за тестом 3

Середня потужність системи	Метод							
	Паралельний			Послідовний				
	ETALON_P	FCFS_P	FSA_P	ETALON	FCFS	FSA	FSAMin	FSAMax
0,1	15 596	60 829	61 130	9 201	90 205	90 232	90 329	90 230
0,15	16 329	57 489	43 400	9 215	90 244	70 264	50 245	50 213
0,2	15 507	57 884	34 756	9 266	90 206	70 217	50 221	50 225
0,25	16 383	56 309	29 947	9 235	80 083	70 115	30 311	30 228
0,3	12 518	56 219	26 654	9 217	80 092	70 274	30 241	30 186
0,35	15 357	56 024	23 429	9 244	70 068	60 047	22 747	22 728
0,4	16 304	56 245	21 471	9 225	80 066	23 435	20 205	20 263
0,45	11 473	21 865	18 226	9 201	30 205	26 746	20 243	20 246
0,5	11 440	56 219	17 046	9 221	80 047	20 226	20 218	20 243
0,55	12 348	55 744	16 305	9 200	70 067	20 095	20 083	20 079
0,6	11 967	29 690	15 525	9 221	40 030	15 232	15 235	15 252
0,65	11 594	21 639	14 833	9 212	30 216	16 732	16 738	16 772
0,7	11 351	55 989	14 455	9 200	70 095	20 076	20 076	20 069
0,75	12 017	29 165	13 529	9 212	40 079	10 202	10 212	10 224
0,8	11 469	29 021	12 857	9 213	40 090	10 224	10 203	10 207
0,85	11 952	29 236	12 578	9 225	35 118	10 199	10 062	10 082
0,9	11 482	55 985	12 488	9 201	70 077	20 087	20 059	20 102
0,95	11 766	11 332	11 454	9 233	10 190	10 248	9 242	9 202
1	10 938	12 826	12 859	9 222	9 248	9 228	9 252	9 221

Розглянемо три стани системи для тестування:

1. Кількість завдань перевищує кількість ПК — тест 1.
2. Кількість завдань дорівнює кількості ПК — тест 2.
3. Кількість завдань менше кількості ПК — тест 3.

На рис. 2 наведено діаграми, побудовані за результатами тестування 1 (табл. 1). На основі отриманих результатів можна зробити висновок про те, що методи FSA і FSA_P дають кращі результати, ніж методи FCFS і FCFS_P.

Як видно на рис. 2, а, крива методу FSA_P має плавну форму, а крива методу FCFS_P є ламаною. Це дозволяє зробити висновок, що метод FSA_P є добре прогнозованим і, знаючи потужність системи та об'єм

завдань, можна складати прогноз завершення обчислень. Щоправда, вирішуючи задачі методом FCFS_P, досить складно прогнозувати завершення обчислень, адже багато залежить від того, на який вузол, яка задача прийде на виконання.

На рис. 2, б, видно, що послідовний метод FSA має більш гладку криву, ніж метод FCFS, що також свідчить про кращу його прогнозованість.

На рис. 2, в, де зображені додаткові методи FSA Min та FSA Max в порівнянні з методом FSA, видно, що відмінності в результатах незначні, але лише для тесту 1.

На рис. 3 наведено діаграми, побудовані за результатами тестування 2 (табл. 2), а на рис. 4 — діаграми, побудовані за результатами тестування 3 (табл. 3). На основі отриманих результатів можна зробити висновок про те, що при однаковій кількості ПК та завдань на виконання запропоновані алгоритми дають кращі результати ніж FCFS та FCFS_P. При цьому прогнозованість їх зберігається. На рис. 4, в, видно, що в цьому випадку методи FSA Min та FSA Max дають кращий результат, ніж метод FSA, і найкращий з них належить методу FSA Max.

Висновки

Результати дослідження ефективності запропонованих методів засвідчили, що їх використання для розподілу завдань у GRID-системах з невідчужуваними ресурсами забезпечує зменшення часу виконання черги завдань у два-три рази у порівнянні з методом планування FCFS за умови, що кількість завдань перевищує кількість вузлів. Водночас, у початковому стані системи, коли кількість завдань дорівнює кількості вузлів, а середня потужність системи складає 0,15—0,35, використання методів Min-Min та Max-Min забезпечує зменшення часу виконання черги завдань (в порівнянні з FSA) приблизно у два рази.

Всі запропоновані методи досить стабільні та добре прогнозовані, а це означає, що використання їх в GRID-системах дасть переваги не тільки в часі та продуктивності, а й дозволить більш ефективно планувати роботу системи. Метод FCFS працює добре, але для GRID-систем, які мають різномірні за потужністю ресурси, його продуктивність суттєво залежить від випадкового факту, що можна вважати значним недоліком.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кривиницька В.Б., Клим Б.В., Романчук, А.Г., Слабінога М.О. Дослідження алгоритмів диспетчеризації в комп'ютерних системах // Розвідка та розробка нафтових і газових родовищ, 2011, № 2(39), с. 93—105.
2. Desktop grids: Connecting everyone to science. Режим доступу <https://sciencenode.org/feature/desktop-grids-connecting-everyone-science.php>
3. Choi S. J., Kim H. S., Byun E. J., Hwan C. S. A Taxonomy of Desktop Grid Systems Focusing on Scheduling // Technical Report: KU-CSE-2006-1120-02, 2006, Department of Computer Science and Engineering, Korea University. Режим доступу <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.5256&rep=rep1&type=pdf>
4. Данильченко О.М., Узденов Т.А. Процес диспетчеризації для GRID-системи з невідчужуваними ресурсами // Вісник ЖДТУ. Серія «Технічні науки», 2012, № 2(61), с. 147—154.
5. Uzdenov T. New Task Scheduling Algorithm for GRID Systems with Non - Alienable Resources // Systems, Decision and Control in Energy II. Studies in Systems, Decision and Control, 202, Vol. 346. Doi: 10.1007/978-3-030-69189-9_12.
6. Kokilavani T., Amalarethinam D.I. G. Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing // Intern. Journal of Computer Applications (0975–8887), 2011, Vol. 20, No. 2, pp. 43—49.
7. Kanani B., Maniyar B. Review on Max-Min Task scheduling Algorithm for Cloud Computing // Journal of Emerging Technologies and Innovative Research (JETIR), 2015, Vol. 2, Iss. 3, pp. 781—784.
8. What is WPF? — Visual Studio | Microsoft Docs. Режим доступу <https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019>
9. Узденов Т.А. Симулятор процесу диспетчеризації задач в GRID-системах з невідчужуваними ресурсами // Електрон. моделювання, 2021, 43, № 1, с. 117—129.

Отримано 31.01.2021

REFERENCES

1. Kropyvnytska, V.B., Klim, B.V., Romanchuk, A.G. and Slabinoga, M.O. (2011), “Investigation of scheduling algorithms in computer systems”, *Rozvidka ta rozrobka naftovykh i ha-zovykh rodovyshch*, Vol. 2, no. 39, pp. 93-105.
2. Desktop grids: Connecting everyone to science. URL: <https://sciencenode.org/feature/desktop-grids-connecting-everyone-science.php>.
3. Choi, S.J., Kim H.S., Byun, E.J. and Hwan, C.S. (2006), “A Taxonomy of Desktop Grid Systems Focusing on Scheduling”, Technical Report: KU-CSE-2006-1120-02, Department of Computer Science and Engineering, Korea University. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.5256&rep=rep1&type=pdf>
4. Danilchenko, O.M. and Uzdenov, T.A. (2012), “The process of task scheduling in geographically distributed computer systems with non-alienable resources”, *Visnyk ZHDTU, ser. «Tekhnichni nauky»*, Vol. 2, no. 61, pp.147-154.
5. Uzdenov, T. (2021), “New Task Scheduling Algorithm for GRID Systems with Non — Alienable Resources” *Systems, Decision and Control in Energy II. Studies in Systems, Decision and Control*, Vol. 346. DOI: 10.1007/978-3-030-69189-9_12.

6. Kokilavani, T. and Dr. George Amalarethnam, D.I. (2011), "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", *International Journal of Computer Applications*, Vol. 20, no. 2, pp. 43–49.
7. Kanani, B. and Maniyar, B. (2015), "Review on Max-Min Task scheduling Algorithm for Cloud Computing", *Journal of Emerging Technologies and Innovative Research (JETIR)*, Vol. 2, pp. 781–784.
8. What is WPF? — Visual Studio. URL: <https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019>.
9. Uzdenov, T. (2021), "Simulator of task scheduling in GRID system with non – alienable resources", *Electronic modeling*, ISSN 0204-3572, Vol. 1, no. 43, pp. 120.

Received 31.01.2021

Т.А. Узденюв

REDUCTION OF TASK QUEUE EXECUTION TIME IN GRID-SYSTEMS WITH INALIENABLE RESOURCES

This paper describes the mathematical formulation of the problem of task flow scheduling in GRID – systems with inalienable resources, taking into account the power of the node and the power of the problem, as key factors affecting system performance. There are four methods for solving it. One for easily parallel tasks, the other three for sequential tasks. Described client - server architectural model for building software for distributed computing and tasks that require high computing power of the system. The expediency of comparing the efficiency of the proposed methods with the existing FCFS method, as well – known and widely used in various GRID – systems, is substantiated. Describes the test results, which showed that all new methods give a better result than FCFS.

К е у в о р д s: scheduling algorithms, GRID system, GRID simulator, distributed computing, flows scheduling.

УЗДЕНОВ Тарас Амурович, аспірант Державного університету «Житомирська політехніка». У 2007 р. закінчив Житомирський державний технологічний університет. Область наукових досліджень — розподілені обчислення, планування завдань, GRID-системи.