Сравнительный анализ методов оценки производительности узлов в распределенных системах

Мин Тху Кхаинг, С.А. Лупин, Ай Мин Тайк, Д.А. Федяшин

***Аннотация*— Статическая балансировка нагрузки узлов в распределенных вычислительных системах – это один из основных методов, позволяющих повысить реальную производительность гетерогенных вычислителей, основанных на GRID-технологии. Точность балансировки определяется возможностью оценки производительности каждого узла, принимающего участие в распределенных вычислениях. Практический опыт работы с распределенными вычислениями показывает, что этап тестирования производительности узлов занимает достаточно много времени. Решение проблемы может быть связано с однократным определением характеристик узла с помощью одного из общедоступных тестов. В работе представлены результаты сравнительно анализа эффективности различных тестов или бенчмарков для задачи балансировки. Представленные результаты могут быть полезны при проведении распределенных вычислений.**

***Ключевые слова*— распределенная вычислительная система; GRID-технологии; балансировка нагрузки; тесты производительности процессора.**

I. Введение

Несмотря на то, что неуклонно возрастающая вычислительная мощность отдельных процессоров позволяет решать все более сложные задачи, всегда будут существовать и такие, для решения которых требуется значительно более высокая производительность. В ответ на эти вызовы люди создали суперкомпьютеры, объединяющие ресурсы миллионов ядер. Конечно, это вершина современной архитектуры высокопроизводительных вычислительных систем (HPC, High Performance Computing). Такие системы дорого не только создавать, но и использовать. Модели, занимающие верхние строчки в списке TOP500, потребляют десятки мегаватт [1]. Такие решения доступны только крупным корпорациям и государственным структурам, что объясняет неугасающий интерес исследователей к технологиям распределенных вычислений. Их несомненным достоинством является высокая доступность, поскольку для создания вычислительной среды можно использовать все доступные исследователям компьютеры.

Статья получена 8 марта 2023.

С.А. Лупин, профессор, Национальный исследовательский университет «МИЭТ», МСЦ РАН – филиал ФИЦ ФНЦ НИИСИ РАН, (e-mail: lupin@miee.ru);

Мин Тху Кхаинг, аспирант Национального исследовательского университета «МИЭТ»; (email: minthukhaing55@gmail.com).

Ай Мин Тайк, к.т.н., докторант Национального исследовательского университета «МИЭТ» (e-mail: ayeminthike52@gmail.com);

Д.А. Федяшин, начальник ВЦ, Национального исследовательского университета «МИЭТ» (e-mail: varlok-diman@mail.ru).

Подобный подход получил название GRID-computing [2]. Пиковая производительность такого вычислителя равна суммарной производительности всех входящих в его состав узлов [3-5]. GRID-вычисления часто используют для решения различных прикладных оптимизационных задач, используя при этом метод перебора (или грубой силы, Brute Force). Распределение нагрузки между узлами при таком подходе происходит через передаваемые им параметры. Если созданная вычислительная сеть является гомогенной, то нагрузка между ними распределяется равномерно. На практике, особенно в случае добровольных вычислений, сеть получается существенно гетерогенной и возникает проблема балансировки нагрузки узлов. В работе [6] мы представили результаты исследования метода статической балансировки, основанного на тестовом запуске на узлах сети рабочего приложения для определения их реальной производительности. Метод демонстрирует высокую эффективность, но его практическое использование требует проведения затратной по времени процедуры тестирования всех узлов сети. В этой же работе показано, что использование в качестве оценки вычислительной мощности узла значения пиковой производительности его процессора, определяемое производителем, не дает возможности сбалансировать нагрузку. Ниже представлены результаты анализа применимости различных тестов для балансировки нагрузки.

II. Тесты производительности процессоров

Сегодня известно множество различных подходов к оценке вычислительной мощности вычислителя. Некоторые из них представлены в [7]. Следует заметить, что само понятие вычислительной мощности не имеет четкого определения, поскольку является зависимым от множества параметров вычислителя, важность которых, в свою очередь, зависит от решаемой им задачи.

К наиболее очевидным параметрам следует отнести:

- тактовую частоту процессора;
- число исполняемых за один такт команд;
- число ядер.

Важность других параметров уже будет зависеть от решаемой задачи:

- объём кэша ядра;
- объём оперативной памяти вычислительного узла;
- её архитектура и быстродействие.

Отдельно следует выделить и те параметры, которые можно отнести к проблемно-зависимым, например:

- для видеоигр это характеристики графического

ускорителя;

- для параллельных вычислений это топология и скорость передачи связывающей узлы сети.

Модным параметром с экологической точки зрения является энергетическая стоимость вычислительной операции.

Споры о возможности построения некоторого синтетического теста, учитывающего все приведенные параметры, не утихают. Однако с учётом очевидных особенностей распределённой среды вычислений, мы будем рассматривать только те тесты, которые позволяют оценить производительность процессора. В таблице 1 представлен не претендующий на полноту список выбранных нами тестов.

Таблица 1. Бенчмарки для процессора

№	Бенчмарк	Core	CPU	Free	Примечание
1	Geekbench 5 [8]	+	+	+/-	Бесплатный пробный период
2	Passmark [9]	+	+	+/-	Бесплатный пробный период
3	SuperPi [10]	+	-	+	
4	wPrime [11]	-	+	+	Запуска от имени администратора
5	Cinebench [12]	+	+	+	
6	7zip [13]	-	+	+	
7	AIDA64 [14]	-	+	+/-	Бесплатный пробный период
8	Dolphin Emulator [15]	-	+	+	

Отметим, что основной целью проводимых исследований является оценка возможности использования получаемых значений мощности процессоров для сбалансированного распределения нагрузки в распределенной вычислительной системе. В таблице представлены только те тесты, которые позволяют бесплатно запустить бенчмарк. Возможность отдельной оценки для ядра процессора обязательным критерием не являлась.

III. Распределенная вычислительная среда

Вычислительная среда была организована в студенческом общежитии МИЭТ с использованием Grid-технологий. Детально процесс её построения описан в [6] и здесь мы приведем только основные моменты. Все узлы работают под ОС Windows, (в основном Windows 10), обладают различными характеристиками (Табл. 2), соединены локальной сетью со скоростью передачи данных около 50 Мбит/с.

Таблица 2. Узлы вычислительной сети

Узел	CPU	Число ядер	Частота (GHz)	Pike Performance (GFlops)
1	i5-3470	4	3,2	102,4
2	i7-8700	6	3,2	307,2
3	i3-4210	2	1,7	54,4
4	i3-9400F	6	2,9	278,4
5	i5-560	2	3,3	52,8
6	i3-6100U	2	2,3	73,6
7	i5-6200U	2	2,3	73,6
8	i5-7300U	2	2,6	83,2
9	i5-10210U	4	1,6	102,4
10	i3-9100T	4	3,1	198,4

c_v	0,673

Запуск распределенного приложения осуществляется удаленно с одного из узлов, с помощью утилиты PsExec, входящей в состав PSTools [16].

Для каждого узла определена пиковая производительность. В качестве характеристики неоднородности вычислительной среды используется значение коэффициента вариации $c_v = \frac{\sigma}{\mu}$.

Представленные значения показывают, что узлы вычислительной среды существенно неоднородны, т.е. это гетерогенная среда, и для эффективного использования её ресурсов требуется балансировка нагрузки.

IV. Распределенное приложение

В качестве распределённого приложения, для которого будет определяться эффективность балансировки, мы используем задачу численного интегрирования. Специальные требования к интегрируемой функции не предъявлялись. Также мы не использовали методы, основанные на изменении шага интегрирования в зависимости от градиента функции, поскольку это делает невозможным применять статическую балансировку нагрузки.

При вычислениях определённого интеграла как предела суммы, имеем:

$$\int_a^b f(x)dx \approx \frac{h}{2} \cdot \sum_{i=1}^{N} \big(f(i \cdot h) + f(i \cdot h + h)\big);$$

$$h = (b - a)/N.$$

Это позволяет в качестве параметров, определяющих вычислительную нагрузку узла использовать два равноправных метода:

- узлу передается число шагов и границы интервала, шаг интегрирования определяется самим узлом;

- узлу передается шаг интегрирования и границы интервала, число шагов определяется самим узлом.

Распределенное приложение (***test_1***) для вычислительной среды написано на языке C++.

Поскольку сеть содержит компьютеры, имеющие многоядерные процессоры, для их эффективной загрузки требуется многопоточное приложение, которое и было разработано с использованием технологии OpenMP [17].

Для минимизации влияния планировщика ОС в последовательном режиме приложение запускалось только на одном ядре процессора.

V. Оценка производительности узлов сети

Для оценки производительности узлов сети кроме представленных в табл. 1 бенчмарков было использовано также и разработанное простое приложение (***test_2***), обеспечивающее нагрузку процессора за счёт циклического повторения простых математических операций:

$$y = \frac{1}{cos^2(a) + 1} + \frac{1}{cos^2(b) + 1}$$

Это компактное приложение не требовательно ни к оперативной памяти, ни к кэш, что критично для гетерогенных систем. В табл. 3 показаны полученные для всех узлов оценки и соответствующее значение коэффициента вариации для всей сети.

Таблица 3. Оценка производительности узлов (бенчмарки)

Узел	Бенчмарк							
	1	2	3	4	5	6	7	8
1	4673	2415	1,097	10,45	2238	20,56	106,9	99
2	12974	5785	0,859	4,766	7597	56,67	406,9	69
3	2308	1150	1,503	26,45	1367	9,574	76,52	143
4	9535	4551	0,938	5,034	5632	38,70	369,9	70
5	1722	1105	1,644	27,34	1242	9,675	26,71	153
6	2627	1285	1,453	25,48	1439	10,45	70,15	132
7	3006	1333	1,36	23,94	1310	12,38	86,04	120
8	3698	1389	1,18	22,26	1915	15,46	111,2	117
9	6329	2911	0,969	9,939	2887	28,70	206,8	85
10	5516	2816	1,031	8,16	3430	22,59	216,8	82
σ	3391	1520	0,256	8,967	2024	14,49	123,8	25,8
μ	5239	2474	1,203	16,38	2906	22,47	167,8	107
c_v	0,647	0,614	0,213	0,547	0,696	0,645	0,738	0,241

Отметим, что разные тесты дают разные значения неоднородности вычислительной среды.

Теперь проведем оценку узлов сети с использованием разработанных приложений ***test_1*** и ***test_2***. Поскольку приложение ***test_1*** мы будем использовать в дальнейшем для оценки эффективности балансировки, в этом случае диапазон интегрирования ограничен. Результат показан в таблице 4.

Таблица 4. Оценка производительности узлов (*test*)

Узел	Время вычислений (сек)			
	test_1		***test_2***	
	$Tcore(i)$	$Tprc(i)$	$Tcore(i)$	$Tprc(i)$
1	48,97	12,77	25,16	7,26
2	31,52	5,73	16,20	2,88
3	56,70	32,08	31,37	17,38
4	34,10	5,91	17,65	3,46
5	57,55	29,51	28,59	15,17
6	63,81	33,66	30,80	15,67
7	53,80	30,00	25,80	13,84
8	41,39	23,06	20,41	11,06
9	41,00	9,83	21,20	4,35
10	35,98	9,78	19,07	5,23
σ	10,63	10,90	5,21	5,32
μ	46,48	19,23	23,62	9,63
c_v	0,229	0,567	0,221	0,552

Как было ранее показано в [6], именно оценка неоднородности сети, полученная при запуске реального приложения (***test_1***), позволяет достичь сбалансированной работы узлов. Сравнивая результаты, полученные с использованием бечнмарков, можно отметить, что наиболее близкие к ним показатели даёт ***test_2***. Вычислительные эксперименты покажут, насколько это отличие существенно.

VI. БАЛАНСИРОВКА НАГРУЗКИ И ЭКСПЕРИМЕНТЫ

Существует достаточно много работ, посвященных проблеме балансировки нагрузки в различных вычислительных средах. Их можно разделить на две категории: статические и динамические [7]. Для распределенных вычислений используют статическую балансировку, реализуемую через передаваемые узлам параметры. При динамической балансировке нагрузку узлов можно изменять во время работы приложения [18]. Конечно, такой метод позволяет устранить дисбаланс, но требует активного управления процессом вычислений, что трудно реализуется в распределенной среде.

Для оценки применимости методов оценки производительности узлов для статической балансировки нагрузки были рассчитаны соответствующие коэффициенты для каждого из них. Для этого используется следующее выражение:

$$K(i) = T(i) / \sum_{i=1}^{10} T(i)$$

где $T(i)$ значение производительности узла, полученное одним из бенчмарков. Рассчитанные для двух тестов коэффициенты представлены в таблице 5.

Таблица 5. Балансировочные коэффициенты (бенчмарки)

Узел (i)	Geekbench		PASSMARK	
	$Tcore(i)$	$Tprc(i)$	$Tcore(i)$	$Tprc(i)$
	$K1(i)$	$K2(i)$	$K3(i)$	$K4(i)$
1	0,09747	0,09762	0,08025	0,08921
2	0,15297	0,23383	0,14853	0,24766
3	0,07263	0,04649	0,07927	0,04406
4	0,13956	0,18395	0,10915	0,18201
5	0,06361	0,04466	0,09187	0,03286
6	0,07448	0,05193	0,07863	0,05015
7	0,0741	0,05387	0,09072	0,05738
8	0,07794	0,05614	0,11591	0,07058
9	0,12588	0,11766	0,09283	0,12081
10	0,12137	0,11383	0,11284	0,10528

В таблице 6 показаны результаты балансировки. Тест Geekbench позволяет снизить значение коэффициента вариации до 14%, что нельзя считать удовлетворительным результатом. Тест PASSMARK менее эффективен. С его помощью не удается существенно уменьшить дисбаланс вычислительной среды. Коэффициент вариации в этом случае остается равным 22%.

Таблица 6. Результаты балансировки (*test*)

Узел *(i)*	Время вычислений (сек)			
	Geekbench		PASSMARK	
	$Tcore(i)$	$Tprc(i)$	$Tcore(i)$	$Tprc(i)$
1	48,28	12,58	39,33	12,00
2	48,47	14,32	46,86	15,73
3	42,84	15,86	46,67	15,49
4	54,95	10,78	37,7628	13,82
5	41,87	15,19	39,88	16,48
6	46,55	17,31	56,37	17,62
7	37,96	16,25	52,62	18,57
8	31,97	14.17	52,89	18,03
9	53,53	15,07	47,83	15,52
10	45,91	12,01	35,55	6,377
σ	6,57	2,03	6,79	3,42
μ	45,23	14,37	45,58	14,97

c_v	0,145	0,141	0,149	0,228

Теперь перейдем к методам оценки производительности узлов, основанным на использовании разработанного теста и рабочей задачи. Результаты их запуска на узлах сети представлены в таблице 7.

Таблица 7. Балансировочные коэффициенты (*test*)

Узел (i)	***test_1***		***test_2***	
	$Tcore(i)$	$Tprc(i)$	$Tcore(i)$	$Tprc(i)$
	$K5(i)$	$K6(i)$	$K7(i)$	$K8(i)$
1	0,08997	0,09864	0,08937	0,08922
2	0,13981	0,21861	0,13874	0,22468
3	0,0777	0,03907	0,07166	0,03723
4	0,12864	0,21223	0,12735	0,187
5	0,07656	0,04248	0,07861	0,04264
6	0,06905	0,03723	0,07301	0,04132
7	0,0819	0,04178	0,08715	0,04678
8	0,10645	0,05435	0,11018	0,05853
9	0,10747	0,12754	0,10604	0,14883
10	0,12246	0,12808	0,11789	0,12378

Поскольку при запуске тестовых приложений мы фиксируем время их работы, а не производительность, как в случае с бенчмарками, то методика расчета балансировочных коэффициентов несколько изменится – чем меньше время работы узла, тем выше его производительность. При этом использованы следующие соотношения:

$$K(i) = 1/T(i) / \sum_{i=1}^{10} (1/T(i))$$

здесь $T(i)$ время работы соответствующего узла. Отметим, что аналогичное выражение используется и при расчете коэффициентов для тех бенчмарков, которые определяют время решения задачи, а не производительность процессора, например, ***SuperPi***.

В таблице 8 показаны результаты балансировки, полученные для тестовых приложений.

Таблица 8. Результаты балансировки (*test*)

Узел *(i)*	Время вычислений (сек)			
	test_1		***test_2***	
	Core	CPU	Core	CPU
1	43,9686	12,7235	43,7666	11,6406
2	45,0001	14,004	44,0851	14,2594
3	44,417	13,5162	42,7474	13,4239
4	43,7894	12,4789	45,3923	11,5678
5	45,55656	14,214	46,2547	14,2345
6	43,8745	13,4827	45,7403	13,4949
7	44,067	12,7174	44,8215	14,1969
8	44,1939	12,8353	46,2079	13,6843
9	44,8765	12,3493	44,8765	11,4123
10	45,578	13,2444	45,4851	13,4376
σ	0,64	0,61	1,07	1,09
μ	44,53	13,16	44,94	13,14
c_v	0,014	0,046	0,024	0,083

Как и следовало ожидать, при использовании коэффициентов, полученных при запуске реального приложения, удается достичь высокого уровня сбалансированности работы распределенной среды. При работе приложения на одном ядре дисбаланс составляет всего 1,4%. В многопоточном варианте дисбаланс составляет 4,6%.

Если посмотреть на результаты, полученные для второго теста, то мы увидим, что хотя они и уступают рабочему приложению, но значительно превосходят те, что дают бенчмарки. Для однопоточного приложения дисбаланс составляет 2,4%, а для многопоточного - 8,3%.

Конечно, некоторый дисбаланс сохраняется, но при таком подходе, также как и для бенчмарков, мы можем только один раз оценить производительность узла, и в дальнейшем уже опираться на полученное значение.

В заключительные эксперименты были направлены на поиск коэффициентов, позволяющих снизить уровень дисбаланса узлов вычислительной среды. Коэффициенты нагрузки узлов для *test_1*, приведенные в таблице 7, были скорректированы с учетом результатов запуска приложения (Табл. 8). Полученные значения показаны в таблице 9. При этом использовалась следующие соотношения:

$$K9(i) = K5(i) * T(i)/T_{mid\_core}$$

$$K10(i) = K6(i) * T(i)/T_{mid\_cpu}.$$

Если время работы узла было меньше среднего, то его коэффициент нагрузки повышался, и понижался в противном случае.

Таблица 9. Коррекция коэффициентов (*test_1*)

Узел *(i)*	$K9(i)$	$K10(i)$
1	0,09118	0,10133
2	0,13842	0,20481
3	0,07796	0,03741
4	0,13088	0,22305
5	0,07489	0,03864
6	0,07014	0,03562
7	0,08281	0,04265
8	0,10732	0,05499
9	0,10672	0,13503
10	0,11968	0,12647

В таблице 9 представлены результаты, подтверждающие эффективность корректировки.

Таблица 9. Результаты коррекции коэффициентов (*test_1*)

Узел *(i)*	Время вычислений (сек) ***test_1***	
	Core	CPU
1	44,5599	13,0705
2	44,5527	13,12
3	44,5656	12,9419
4	44,5519	13,1151
5	44,5628	12,9291
6	44,5671	12,8996
7	44,5566	12,9822
8	44,5551	12,9864
9	44,5633	13,0745
10	44,5433	13,0779
σ	0,64	0,077
μ	44,5578	13,02
c_v	0,00016	0,0059

Корректировка позволяет достичь баланса, близкого к идеальному. Для однопоточного приложения дисбаланс составляет 0,016%, а для многопоточного - 0,59%.

VII. Заключение

Полученные в ходе исследований результаты позволили ответить на вопрос о том, какие методы можно использовать для оценки производительности узлов в гетерогенных распределенных вычислительных системах. Кроме очевидного решения – использовать для этого рабочее приложение, мы показали, что и простой вычислительный тест дает достаточно хорошее решение, обеспечивая высокий уровень сбалансированности работы узлов.

Использование бенчмарков значительно менее эффективно. Во-первых, они по-разному оценивают разнородность узлов, а во-вторых, требуют от пользователя достаточно высокой активности, что трудно реализовать при добровольных вычислениях. Вопрос о том, почему бенчмарки дают разные значения производительности процессоров, широко дискутируется в сообществах, но однозначного ответа не имеет, что также не позволяет выбрать какой-то один тест, обеспечивающий адекватную оценку. Необходимо отметить, что даже тест ***SuperPi***, который является счетным, даёт оценки производительности, сильно отличающиеся от реально наблюдаемых при вычислениях.

Наш опыт реализации распределенных вычислений, проводимых в среде, сформированной на добровольной основе из существенно гетерогенных узлов, показывает, что достижение «нулевого дисбаланса» не должно являться основной целью. Важной особенностью добровольных вычислений является то, что реальная производительность узла может изменяться, поскольку в момент проведения общих вычислений он, например, выполняет некоторые системные задачи и это приводит к дисбалансу в ранее равновесной системе.

Дальнейшие исследования в этой области будут направлены на разработку простого механизма хранения найденных оценок производительности узлов для их автоматического использования при балансировке нагрузки.

Поддержка

Статья подготовлена в МИЭТ и МСЦ РАН в рамках государственного задания по теме FNEF-2022-0016.

Библиография

[1] TOP500. [Online]. https://www.top500.org/500

[2] [Online]. https:en.wikipedia.org/wiki/Grid_computing

[3] Md. Firoj Ali. Distributed Computing: An Overview, // Int. J. Advanced Networking and Applications Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290.

[4] M. N. Durrani and J. A. Shamsi. Volunteer computing: requirements, challenges, and solutions, // Journal of Network and Computer Applications,vol. 39, pp. 369–380, 2014.

[5] Sheng Y., Gui L., Wei Z., Duan J., and Liu Y. Layered Models for General Parallel Computation Based on Heterogeneous System. // 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. doi:10.1109/pdcat.2012.85.

[6] Мин Тху Кхаинг, С.А. Лупин, Аунг Тху. Оценка эффективности методов балансировки нагрузки в распределенных вычислительных системах. // International Journal of Open Information Technologies., Vol.9, №11, 2021, с. 30-36

[7] [Online] https://trashexpert.ru/hardware/general-issues/best-cpu-benchmark-tools/ Retrieved: May, 2023

[8] [Online]. Доступ: https://browser.geekbench.com. Retrieved: May, 2023

[9] [Online]. Доступ: https://www.passmark.com Retrieved: May, 2023

[10] [Online]. Доступ: https://www.superpi.net/ Retrieved: May, 2023

[11] [Online]. Доступ: https://www.wprime.net/ Retrieved: May, 2023

[12] [Online]. Доступ: https://www.maxon.net/ru Retrieved: May, 2023

[13] [Online]. Доступ: https://www.7-zip.org Retrieved: May, 2023

[14] [Online]. Доступ: https://www.aida64.combsoft.com Retrieved: May, 2023

[15] [Online]. Доступ: https://www.dolphinemulator.org Retrieved: May, 2023

[16] JJ'S WEBSITE, PsExec Simple Tutorial, [Online]. Доступ: http://jjasser.com/2014/04/psexec-simple-tutorial. Retrieved: May, 2023

[17] OpenMP. [Online]. Доступ: https://www.openmp.org Retrieved: May, 2023

[18] Бабичев С. Л. Распределенные системы: учебное пособие для вузов / С. Л. Бабичев, К. А. Коньков. — Москва: Издательство Юрайт, 2019. — 507 с.

Comparative analysis of methods for evaluating performance of nodes in distributed systems

Min Thu Khaing, S. Lupin, Aye Min Thike, D. Fedyashin

***Abstract*— Static load balancing of nodes in distributed computing systems is one of the main methods for increasing the real performance of heterogeneous computers based on GRID technology. The balancing accuracy is determined by the ability to evaluate the performance of each node involved in distributed computing. Practical experience with distributed computing shows that the node performance testing stage takes quite a lot of time. The solution to the problem can be associated with a single determination of the nodes characteristics using one of the publicly available tests. This paper presents the results of a comparative analysis of the effectiveness of various tests or benchmarks for the balancing task. The presented results can be useful in distributed computing.**

***Keywords*— distributed computing system; GRID technologies; load balancing; processor performance tests.**

REFERENCES

[1] TOP500. [Online]. https://www.top500.org/500

[2] [Online]. https:en.wikipedia.org/wiki/Grid_computing

[3] Md. Firoj Ali. Distributed Computing: An Overview, // Int. J. Advanced Networking and Applications Volume: 07 Issue: 01 Pages: 2630-2635 (2015) ISSN: 0975-0290.

[4] M. N. Durrani and J. A. Shamsi. Volunteer computing: requirements, challenges, and solutions, // Journal of Network and Computer Applications,vol. 39, pp. 369–380, 2014.

[5] Sheng Y., Gui L., Wei Z., Duan J., and Liu Y. Layered Models for General Parallel Computation Based on Heterogeneous System. // 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. doi:10.1109/pdcat.2012.85.

[6] Min Thu Khaing, S.A. Lupin, Aung Thu. Ocenka jeffektivnosti metodov balansirovki nagruzki v raspredelennyh vychislitel'nyh sistemah. // International Journal of Open Information Technologies., Vol.9, #11, 2021, s. 30-36.

[7] [Online] https://trashexpert.ru/hardware/general-issues/best-cpu-benchmark-tools/ Retrieved: May, 2023

[8] [Online]. URL: https://browser.geekbench.com. Retrieved: May, 2023

[9] [Online]. URL: https://www.passmark.com Retrieved: May, 2023

[10] [Online]. URL: https://www.superpi.net/ Retrieved: May, 2023

[11] [Online]. URL: https://www.wprime.net/ Retrieved: May, 2023

[12] [Online]. URL: https://www.maxon.net/ru Retrieved: May, 2023

[13] [Online]. URL: https://www.7-zip.org Retrieved: May, 2023

[14] [Online]. URL: https://www.aida64.combsoft.com Retrieved: May, 2023

[15] [Online]. URL: https://www.dolphinemulator.org Retrieved: May, 2023

[16] JJ'S WEBSITE, PsExec Simple Tutorial, [Online]. URL: http://jjasser.com/2014/04/psexec-simple-tutorial. Retrieved: May, 2023

[17] OpenMP. [Online]. URL: https://www.openmp.org Retrieved: May, 2023

[18] Babichev S. L. Raspredelennye sistemy: uchebnoe posobie dlja vuzov / S. L. Babichev, K. A. Kon'kov. — Moskva: Izdatel'stvo Jurajt, 2019. — 507 s.