УДК 004.272.26

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ЧЕРЕЗ СБАЛАНСИРОВАННУЮ НАГРУЗКУ И ОПТИМАЛЬНОЕ ОБРАЩЕНИЕ К ПАМЯТИ

В. В. Филимонов, аспирант кафедры САПР ВС РГРТУ, Рязань, Россия;
orcid.org/0000-0001-6949-0579, e-mail: filvvfil@yandex.ru
В. В. Сускин, д.т.н., профессор кафедры САПР ВС РГРТУ, Рязань, Россия;
orcid.org/0000-0001-5124-9672, e-mail: vsuskin@mail.ru

Рассматриваются методы оценки производительности и возможность их использования для разработки параллельных высокоэффективных алгоритмов, выполняющихся на многоядерных процессорах и многопроцессорных вычислительных системах. ***Целью работы*** *является исследование существующих способов оценки производительности для использования при выборе оптимальной мощности аппаратной части для реализации вычислительных задач. Актуальность проблемы заключается в непрерывном росте объемов информации, которую необходимо быстро обрабатывать и использовать для принятия ключевых решений во многих технологических областях. В работе рассмотрены наиболее важные законы, на которых базируется оценка производительности, проведено численное и графическое сравнение методов оценки, получены графики эффективной зависимости алгоритмов от аппаратной части, а также сформулировано математическое описание обеспечения сбалансированной нагрузки и оптимального обращения к памяти.*

Ключевые слова: *производительность вычислительных систем, методы оптимизации, многопроцессорные вычислительные системы, оценка производительности, эффективность многопроцессорных вычислительных систем, повышение производительности, параллелизм, параллельные вычисления.*

DOI: 10.21667/1995-4565-2021-75-63-70

Введение

Средства вычислительной техники на сегодняшний день являются неотъемлемой составляющей, практически любой технологической отрасли. Активное развитие и высокий уровень внедрения компьютеров в некоторые из областей позволяет получать огромное количество информации, которую требуется анализировать и выводить для пользователя в понятном виде. Объем получаемой информации, и ее достоверность постоянно увеличиваются. Как следствие, актуальность проблемы в скорости обработки и анализа информации очень высока. Наиболее часто, на наш взгляд, задачи повышения производительности вычислений решаются в областях машинного обучения, при распознавании видео и фотоизображений, в частотности при реализации точной динамической навигации, где необходима высокая скорость принятия решений [1].

Для эффективной реализации алгоритмов и оптимального выбора аппаратных средств разработчикам необходимо заранее оценивать производительность и учитывать архитектурные особенности современных вычислительных систем. В данной работе мы рассматриваем возможность повышения быстродействия обработки данных, основываясь на наиболее известных способах оценки производительности, приводим зависимость теоретического ускорения распараллеленных алгоритмов от количества независимых вычислительных устройств в аппаратной части. По итогу определена важность контроля над распределением памяти и нагрузки, составлена математическая задача оптимизации и сбалансированной загрузки. Также сделаны общие выводы о теоретической оценке производительности систем.

Теоретическая часть

В связи с постоянным ростом объема входной информации и увеличением скорости работы многоядерных процессоров разработка алгоритмов требует знаний в организации сложных вычислений [4, 6]. На сегодняшний день пиковые значения тактовых частот процессора достигают примерно 5,5 ГГц, что, безусловно, является довольно большим преимуществом перед старыми аппаратными возможностям, однако это не единственный параметр, отвечающий за быстродействие системы в целом. Более того, чаще всего разработчикам приходиться работать с готовой аппаратной частью или реализовывать алгоритмы, способные быстро выдавать результат на различных вычислительных машинах, таким образом, задача повышения производительности сводится к разработке алгоритмов способных выполнять параллельные вычисления [5]. Даже при использовании мощных процессоров, работающих на пиковой скорости, вопрос распараллеливания вычислений имеет большое значение. Одним из доказательств необходимости распараллеливания является создание технологии гиперпоточности (Hyper-Threading), она позволяет организовать вычисления на одном процессоре так же, как на нескольких, благодаря логическому разделению.

Логично, что для объективности в рассмотрении возможности повышения эффективности систем за счет параллелизма необходимо использовать методы оценки производительности. Сравним наиболее известные методики. Первым рассмотрим закон Амдала (1).

Общее время работы нашей программы при однопоточном выполнении обозначим за T, долю от общих вычислений, которая может быть выполнена только последовательными расчетами, обозначим A, количество максимально возможных потоков для вычислений обозначим N, таким образом, получим формулу, описывающую полученное ускорение (S_p), в сравнении с последовательным (однопоточным) решением:

$$S_p = \frac{T}{T \cdot A + \frac{T - T - A}{N}} = \frac{1}{A + \frac{1 - A}{N}}. \quad (1)$$

Рассмотрев данный закон, мы понимаем, что наибольшее влияние на эффективность ускорения влияет та часть программы, которая может быть выполнена параллельно [3]. Таким образом, стоит понимать, что не для каждой задачи увеличение количества потоков и соответственно распараллеливание вычислений будет иметь смысл. Для наглядности рассчитаем и запишем в таблицу 1 выборку значений.

Таблица 1 – Расчетные значения по закону Амдала

Table 1 – Calculated values according to Amdahl's law

N / A	2	4	8	16
0 (0 %)	2	4	8	16
0,25 (25 %)	1,6	2,29	2,91	3,37
0,5 (50 %)	1,33	1,6	1,78	1,89
1 (100 %)	1	1	1	1

Из таблицы видим, что максимальный прирост производительности наблюдается в той программе, где отсутствует необходимость последовательных вычислений и их все можно выполнить параллельно. Однако на практике идеальные условия встречаются крайне редко, поэтому основной вывод, который можно сделать из данных расчетов, что прирост эффективности для каждой задачи ограничен объемом последовательных вычислений, а следовательно, прирост производительности не может превышать значения $\frac{1}{A}$. Так, при половине последовательного кода максимальный прирост производительности не может превысить 2.

Важно отметить, что данные расчеты не учитывают возможности повышения сложности и объема решаемых задач, а также издержек, возникающих при взаимодействии параллельно работающих процессоров между собой и обращения к различным уровням памяти [7, 9]. Чтобы получить более точную оценку ускорения в параллельных вычислениях, предлагаем рассмотреть еще несколько законов и метрик.

Для учета возможного увеличения вычислений воспользуемся законом Густафсона – Барсиса (2). Как и в предыдущем расчете, примем долю расчетов, которые необходимо выполнять только последовательно, за A, количество потоков – за N, а также будем учитывать долю параллельных расчетов программы P. В результате получаем отношение объема параллельных вычислений к последовательным за один промежуток времени (S_A):

$$S_A = \frac{P \cdot N + A}{P + A} = \frac{N(P + A)}{P + A} + \frac{A - N \cdot A}{P + A} = N + (1\text{-}N)\frac{A}{P + A} = N + (1\text{-}N)A. \quad (2)$$

Таким образом, при подстановке реальных значений получаем ускорение масштабирования, выраженное объемом, на который возможно увеличение выполняемых задач.

Таблица 2 – Расчетные значения по закону Густафсона – Барсиса
Table 2 – Calculated values according to Gustafson – Barsis law

N / A	2	4	8	16
0 (0 %)	2	4	8	16
0,25 (25 %)	1,75	3,25	6,25	12,25
0,5 (50 %)	1,5	2,5	4,5	8,5
1 (100 %)	1	1	1	1

Из таблицы 2 видим, что приспособленность алгоритма к параллельным вычислениям и их использование позволяют не только повысить скорость вычислений определенного объема информации, но и значительно увеличить объем обрабатываемой информации за единицу времени. Тем не менее, несмотря на внушительные коэффициенты масштабирования систем, получившиеся в расчетах, стремясь смоделировать наиболее точную систему, не стоит пренебрегать существующими ограничениями.

В многопроцессорных вычислительных системах обычно каждый процессор имеет собственную локальную память, которая в совокупности образует общую память и используется при решении задач. Таким образом, доступная емкость памяти создает ограничение на максимальный объем выполняемых вычислений [8, 10]. Для математического описания данного ограничения, на наш взгляд, хорошо подходит закон Сана-Ная (3). Его постановка, как и в предыдущем расчете, заключается в том, что при увеличении числа параллельных потоков увеличивается объем выполняемых за единицу времени задач, но в условиях ограничения объема по памяти.

В качестве переменных данного выражения будем использовать обозначения аналогичные предыдущим: A – доля последовательных вычислений, N – количество потоков и G_n – коэффициент масштабируемости распараллеленной части. В результате получаем ускорение производительности, ограниченное памятью (S_N):

$$(S_N) = \frac{A + (1\text{-}A) \cdot G_n}{A + (1\text{-}A) \cdot \frac{G_n}{N}}. \quad (3)$$

Из полученного выражения следует, что при $G_n = 1$ размер задачи фиксирован, а при $G_n = N$ емкость памяти пропорционально возрастает нагрузке. Таким образом, при наличии

дефицита памяти ($G_n < N$) расчетное ускорение производительности будет меньше, однако при наличии запаса по памяти ($G_n > N$) оценка ускорения будет более оптимистична.

Экспериментальная часть

На практике производительность работы нашей системы и скорость выполнения необходимых нам задач зависят от множества факторов. В первую очередь, от аппаратной части самой системы: частоты работы процессора и его непосредственной памяти, наличия видеокарты, объема оперативной памяти, скорости коммуникационных связей (пропускной способности), чтения/записи данных и прочих факторов [2]. Предположим, что система с определенными параметрами у нас есть, вторым важнейшим условием эффективного решения поставленной задачи является алгоритм. От его выбора зависит, сможет ли вообще наша задача быстро решаться, если выбранный алгоритм изначально не оптимален, то дальнейшее его ускорение не принесет необходимого результата [10]. Соответственно после выбора алгоритма основной задачей является его оптимизация под имеющуюся аппаратную часть с учетом производительности и пропускной способности. В этом и будет заключаться основная цель нашей работы.

Для того чтобы получить более точную оценку производительности, представим графическую зависимость ускорения от количества процессоров на основании ранее рассмотренных законов (1, 2, 3). Долю расчетов в программе, которая может быть выполнена только последовательно, возьмем 5 %.

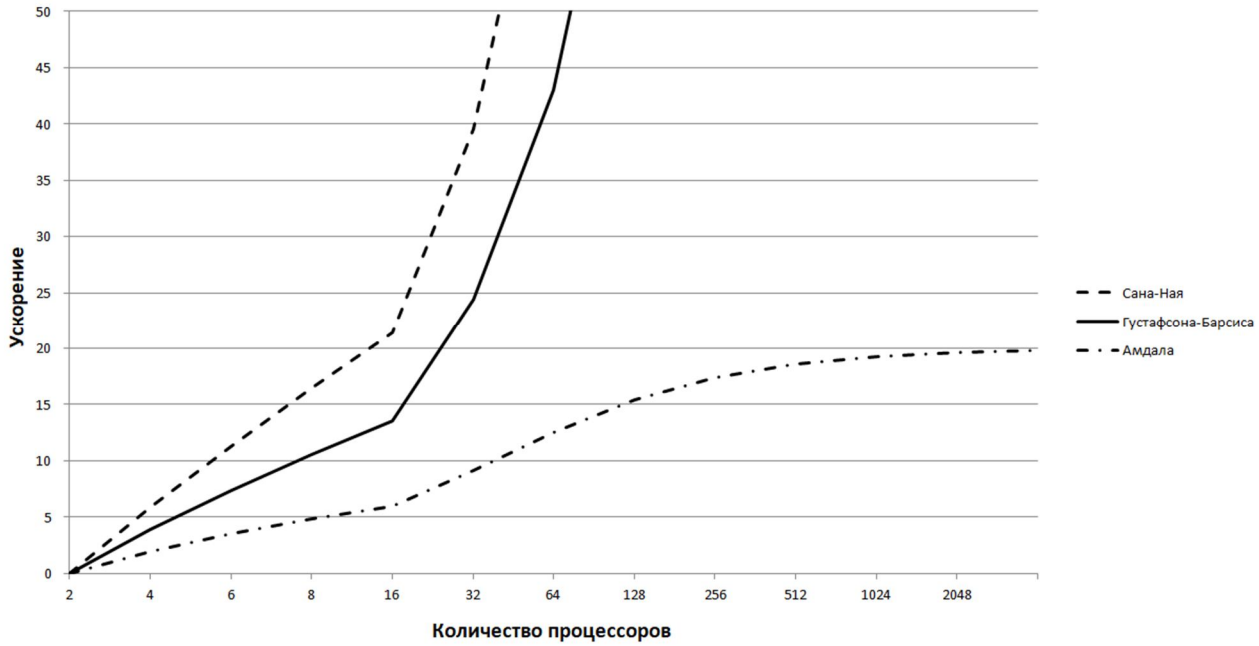

Рисунок 1 – Зависимость ускорения от количества процессоров
Figure 1 – Dependence of acceleration on the number of processors

Из графика может показаться, что увеличение количества процессоров полностью решает проблемы производительности, однако для оптимального описания возможности увеличения производительности многопроцессорной системы также требуется рассматривать взаимодействие между параллельно работающими процессорами. Воспользуемся метрикой Карпа – Флэтта (4), в ней предложено использовать эквивалент объема распараллеленной части программы – показатель e, который получают экспериментальным путем на конкретной вычислительной машине.

$$e=\frac{\frac{1}{S_N}-\frac{1}{N}}{1-\frac{1}{N}}. \quad (4)$$

Оценкой эффективности в данном выражении служит сама метрика, чем меньше ее значение, тем лучше может быть распараллелен код, в идеальных ситуациях, аналогичных постановке задачи Амдала, при увеличении числа процессоров эффективность будет уменьшаться. Также благодаря данным вычислениям с экспериментальными данными мы можем определить узкое место, которое приводит к снижению эффективности, оно может быть выражено ограничением возможности распараллеливания или, наоборот, издержками коммуникаций при параллельных вычислениях. Рассмотрим график (рисунок 2), показывающий зависимость эффективности от количества процессоров для программ с различным уровнем возможного распараллеливания (5 %, 10 %, 25 % и 50 %).

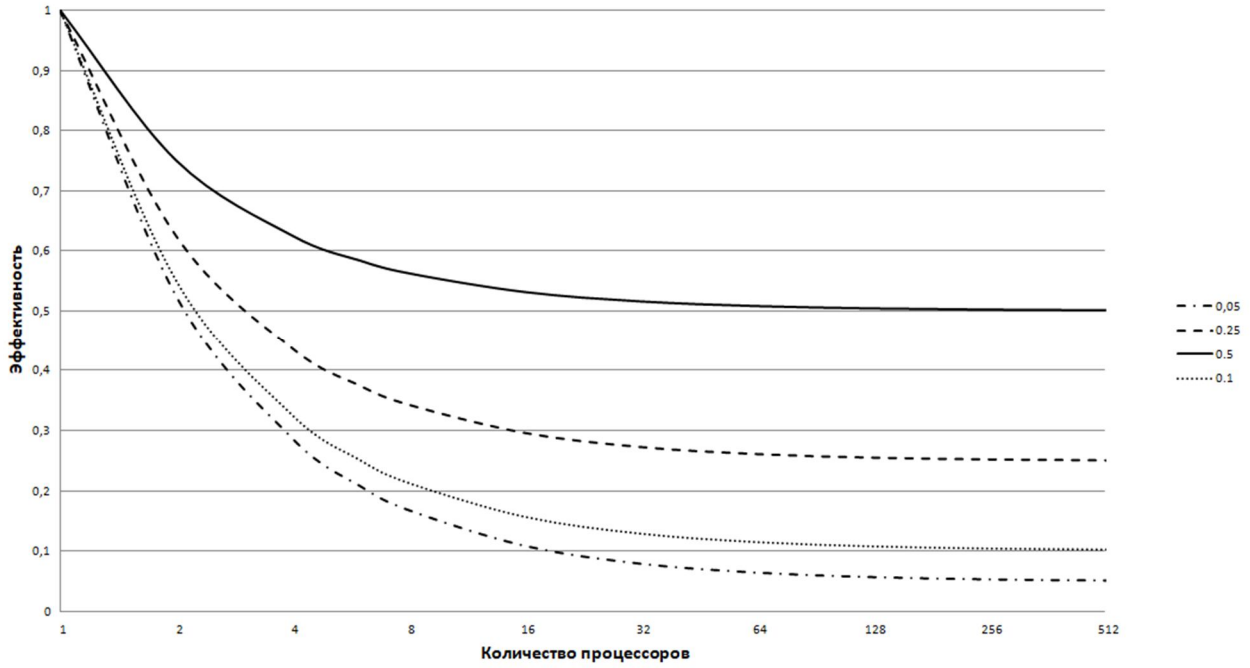

Рисунок 2 – Зависимость эффективности от количества процессоров
Figure 2 – Dependence of efficiency on the number of processors

Сопоставив графики роста производительности и эффективности по количеству используемых ядер (рисунок 3), получим более реалистичный взгляд на работу вычислительных систем. Таким образом, рассмотрев эффективность и ускорение в зависимости от количества процессоров, можно заметить, что после увеличения процессоров до определенного уровня эффективность значительно снижается и дальнейшее их увеличение становится нецелесообразным. Также при оценке производительности различных вычислительных систем стоит учитывать существенные потери скорости обработки информации, связанные с коммуникациями между вычислительными устройствами и различными уровнями памяти.

Сформулируем задачу обеспечения сбалансированной нагрузки и оптимального обращения к памяти для получения максимальной производительности (5):

$$K=\frac{N_{max}}{N_p}, \quad 0<K\le 1, \quad (5)$$

N_{max} – максимальный объем задачи, помещающийся в память. Если размер задачи превысит данный объем, произойдет обращение к памяти более высокого уровня и как следствие –

снижение производительности алгоритма; N_p – количество ядер процессора, которые можно использовать для параллельной обработки.

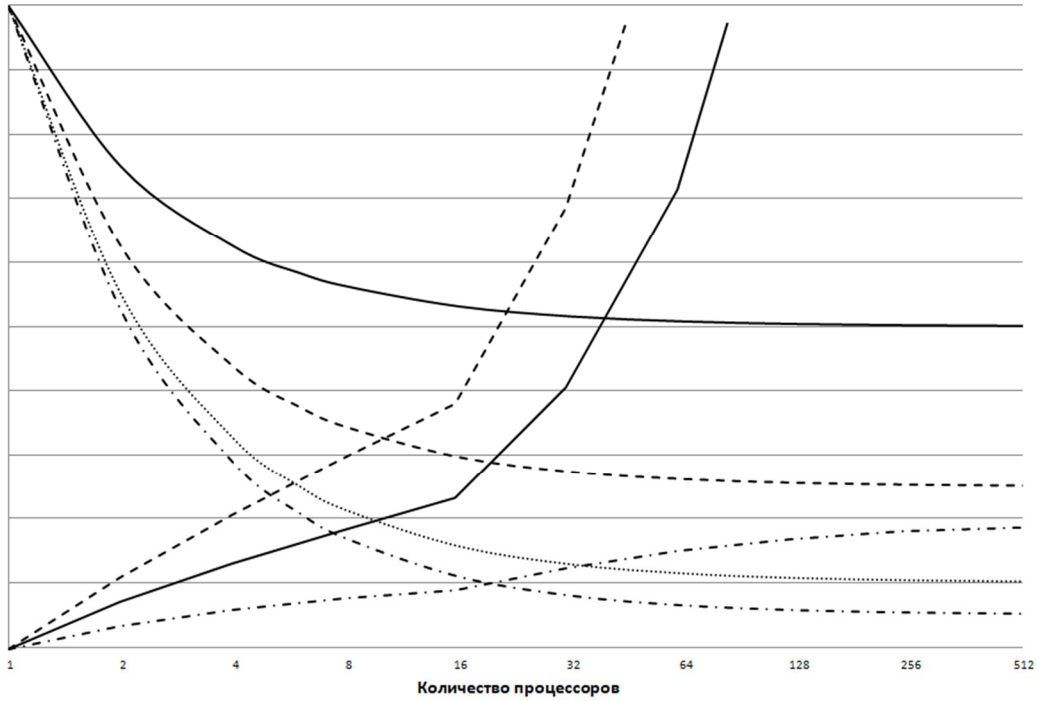

Рисунок 3 – Совмещенные графики эффективности и ускорения
Figure 3 – Combined efficiency and acceleration charts

Учитывая загрузку каждого ядра, можно сформулировать следующее выражение (6):

$$K=\sum_{i=1}^{p}\frac{T_{ci}}{T_f}\cdot\frac{p_i(t_i)}{\sum_{j=1}^{P}P_j(t_j)}, \tag{6}$$

где *i*-е ядро процессора реализует свой объем вычислений за время T_i. Время выполнения распределенной программы можно найти по формуле (7).

$$T_f = max\{T_i\}, \quad i=1,...,P, \tag{7}$$

где P – число ядер процессора, а время T_i учитывает в себе и время T_{ci}, которое показывает время вычислений и время передачи данных, $p_i(t_i)$ производительность процессора в момент времени t_i. Приняв объем вычислений за W, получим (8):

$$T_{ci}=\frac{W_{ci}}{p_i(t_i)}. \tag{8}$$

Таким образом, задача повышения производительности сводится к максимизации значения K в промежутке времени T_f при определенном количестве ядер процессора P.

Понимание данного процесса позволяет разрабатывать оптимальные алгоритмы и оптимизировать обращение к памяти.

Заключение

Рассмотрены методы оценки производительности многопроцессорных вычислительных систем, в их соответствии рассчитаны теоретически возможные ускорения распараллеленных систем с разным уровнем содержания последовательных вычислений и построены графики, демонстрирующие эффективность выполнения алгоритмов в зависимости от аппаратных возможностей вычислительных систем. Определены ограничения, которые влияют на производительность. Сформулировано математическое описание задачи для сбалансирован-

ной загрузки многоядерной процессорной системы. Исходя из полученных результатов и исследований, можно сделать вывод, что теоретические методы оценки производительности дают более позитивный прогноз в сравнении с реальной скоростью вычисления, следовательно, для качественной разработки необходимо учитывать не только ускорение, возможное при добавлении новых аппаратных средств, но и эффективность алгоритма.

Библиографический список

1. **Сюрмаков Р.** Применение машинного обучения и Data Science в промышленности [Электронный ресурс] URL: https://vc.ru/ml/79368-primenenie-mashinnogo-obucheniya-i-data-science-v-promyshlennosti (дата обращения: 12.12.2020).

2. **Орлов С. А., Цилькер Б. Я.** Организация ЭВМ и систем: учебник для вузов. СПб.: Питер. 2011. 688 с.

3. **Фаддеева В. Н., Фаддеев Д. К.** Параллельные вычисления в линейной алгебре // Кибернетика. 1977. № 6. С. 28-40.

4. **Феррари Д.** Оценка производительности вычислительных систем. М.: Машиностроение. 1989. 248 с.

5. **Корячко В. П.** Конструирование микропроцессорных систем контроля радиоэлектронной аппаратуры. М.: Радио и связь, 1987. 160 с.

6. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: «БХВ-Петербург». 2004, 400 с.

7. **Топорков В. В.** Модели распределенных вычислений. М.: ФИЗМАТЛИТ. 2004. 320 с.

8. **Федотов И. Е.** Модели параллельного программирования. М.: Солон-Пресс. 2012. 384 с.

9. **Формалев В. Ф., Ревизников Д. Л.** Численные методы. М.: ФИЗМАТЛИТ. 2004. 400 с.

10. **Шпаковский Г. И., Серикова Н. В.** Программирование для многопроцессорных систем в стандарте MPI. Минск: Изд-во БГУ. 2002. 323 с.

UDC 004.272.26

INCREASED PERFORMANCE OF MULTIPROCESSOR COMPUTING SYSTEMS THROUGH BALANCED LOAD AND OPTIMUM MEMORY ACCESS

V. V. Filimonov, postgraduate student, RSREU, Ryazan, Russia;
orcid.org/0000-0001-6949-0579, e-mail: filvvfil@yandex.ru
V. V. Suskin, Dr. in technical sciences, full professor, RSREU, Ryazan, Russia;
orcid.org/0000-0001-5124-9672, e-mail: vsuskin@mail.com

Methods for evaluating performance and the possibility to use these methods for the development of parallel highly efficient algorithms running on multi-core processors and multi-processor computing systems are considered. ***The aim of this*** *work is to study the existing methods for assessing performance to be used in choosing the optimal power of hardware for the implementation of computational tasks. The urgency of the problem lies in continuous growth of the amount of information that must be quickly processed and used to make key decisions in many technological areas. The paper considers the most important laws on which performance assessment is based, a numerical and graphical comparison of assessment methods is carried out, graphs of effective dependence of algorithms on hardware are obtained, and mathematical description of ensuring balanced load and optimal memory access is formulated.*

Key words: *performance of computing systems, optimization methods, multiprocessor computing systems, performance evaluation, efficiency of multiprocessor computing systems, performance improvement, parallelism, parallel computing.*

DOI: 10.21667/1995-4565-2021-75-63-70

References

1. **Syurmakov R.** *Primenenie mashinnogo obucheniya i Data Science v promyshlennosti* [Elektronnyj resurs] URL: https://vc.ru/ml/79368-primenenie-mashinnogo-obucheniya-i-data-science-v-promysh-lennosti (data obrashcheniya: 12.12.2020).

2. **Orlov S. A., Cil'ker B. YA.** *Organizaciya EVM i sistem: uchebnik dlya vuzov*. SPb.: Piter. 2011, 688 p. (in Russian).

3. **Faddeeva V. N., Faddeev D. K.** Parallel'nye vychisleniya v linejnoj algebra. *Kibernetika*. 1977, no. 6, pp. 28-40. (in Russian).

4. **Ferrari D.** *Ocenka proizvoditel'nosti vychislitel'nyh sistem*. Moscow: Mashinostroenie. 1989, 248 p. (in Russian).

5. **Koryachko V. P.** *Konstruirovanie mikroprocessornyh sistem kontrolya radioelektronnoj apparatury*. Moscow: Radio i svyaz'. 1987, 160 p. (in Russian).

6. **Voevodin V. V., Voevodin Vl. V.** *Parallel'nye vychisleniya*. SPb.: «BHV-Peterburg». 2004, 400 p. (in Russian).

7. **Toporkov V. V.** *Modeli raspredelennyh vychislenij*. Moscow: Fizmatlit. 2004, 320 p. (in Russian).

8. **Fedotov I. E.** *Modeli parallel'nogo programmirovaniya*. Moscow: Solon-Press. 2012, 384 p. (in Russian).

9. **Formalev V. F., Reviznikov D. L**. *Chislennye metody*. Moscow: Fizmatlit. 2004, 400 p. (in Russian).

10. **Shpakovskij G. I., Serikova N. V.** *Programmirovanie dlya mnogoprocessornyh sistem v standarte MPI*. Minsk: Izd-vo BGU. 2002, 323 p. (in Russian).