ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА КЛАСТЕРАХ ИЗ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

А.В. МАЛЫШЕВ, В.В. ШАЙДУРОВ
Институт вычислительного моделирования СО РАН, Красноярск, Россия
e-mail: amal@icm.krasn.ru

Введение.

Работа посвящена использованию многопроцессорной вычислительной системы (логического кластера) на базе серийных персональных ЭВМ, соединенных локальной сетью, с использованием общей дисковой памяти для обмена данными.

Цель исследований состояла в описании процесса параллельного вычисления на таком кластере и оценке предельной оптимальности. Для случая однородной изотропной сети была построена аналитическая модель и получены формулы критериев оптимальности вычислений. В ходе вычислительного эксперимента результаты моделирования были проверены и уточнены.

В качестве модельного теста рассматривалась задача перемножения квадратных вещественных матриц размера $n \times n$. В качестве базового использовался стандартный алгоритм перемножения матриц:

$$\left(a_{ij}\right)\cdot\left(b_{j\ell}\right)=\left(c_{i\ell}\right),\ \text{где}\ \ c_{i\ell}=\sum_{j=1}^{n}a_{ij}b_{j\ell}\,.$$

Для организации параллельности вычислений результирующая матрица разбивалась на одинаковые квадратные подматрицы, которые вычислялись на разных машинах независимо друг от друга.

Из допущения, состоявшего в том, что вычислительная сеть состоит из машин с одинаковым быстродействием *c*, а скорости передачи по сети от каждого вычислителя к серверу также одинаковы и равны *v*, удалось построить аналитическую зависимость времени, ускорения и эффективности параллельных вычислений от *параметра разбиения* и от величины, равной $\left(n\frac{v}{c}+1\right)$, где *n* – это размер перемножаемых матриц. Данную величину было предложено называть *числом вычислительной системы*.

Из построенных зависимостей были получены формулы для разбиения, оптимального в смысле ускорения (отношения времени вычисления на кластере ко времени вычисления на одной ПЭВМ) и разбиения, оптимального в смысле эффективности (отношения ускорения к числу использованных машин). Было показано, что эти оптимальные разбиения существуют и существенно не совпадают друг с другом. Также были получены некоторые вспомогательные результаты, касающиеся динамики эффективности, и различные критерии применимости модели.

Наконец, были определены порядок сложности и эффективность построенных таким образом параллельных алгоритмов перемножения. Порядок вычислительной сложности составил 2.33 для быстрого алгоритма и 2.47 – для эффективного. Предельная эффективность быстрого алгоритма равна 0.66, а эффективность эффективного алгоритма с ростом размера задачи стремится к единице.

1. Методика параллельного разбиения

Пусть *A* и *B* – квадратные матрицы размера *n*×*n*. Пусть матрица *C* размера *n*×*n* – их произведение с элементами $c_{i\ell}=\sum_{j=1}^{n}a_{ij}b_{j\ell}$. С целью упростить рассмотрение положим, что матрица *C* является непрерывным объектом (блоком), поддающимся разбиению на любое количество равных подблоков. Пусти их количество ℓ^2. Сторона подблока (подматрицы) будет иметь размер $k=n/\ell$, где $1 \le \ell \le n$. На практике можно сначала определять ℓ как $\ell\left(n\right)$, так, чтобы $\ell << n$, а затем корректировать *n* в небольших пределах так, чтобы *k* оказалось целым. Это допущение позволяет ограничиться рассмотрением только квадратных

разбиений матрицыпроизведения C, расчленяющих эту матрицу на ℓ^2 подматриц размера $k\times k$. Значение ℓ, таким образом, однозначно определяет способ разбиения матрицы C.

Очевидно, что вычисление подматриц-произведений $C_1,\ldots,C_{\ell^2}$ можно проводить независимо друг от друга и одновременно. В качестве многопроцессорной ЭВМ можно использовалась локальная сеть с выделенным сервером, построенная по протоколу Ethernet.

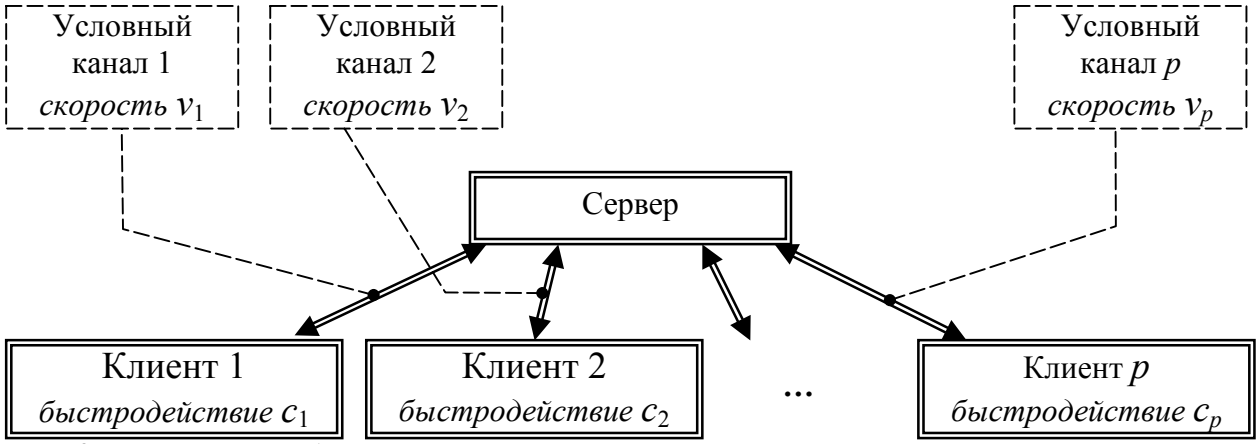
Назначение программы-сервера – получить задание, выполнить разбиение и поместить на НЖМД общего доступа все подзадачи. Каждая подзадача состоит в выполнении перемножения двух матриц: $(k\times n)\cdot(n\times k)$, результатом которого будет подматрица размера $k\times k$. Каждая программа-клиент, обнаружив свободную подзадачу, захватывает её, выполняет и выкладывает решение на НЖМД общего доступа. Программа-сервер, в свою очередь, собирает все решения и компонует результирующую матрицу-произведение C.

2. Модель процесса вычисления

Для построения модели процесса параллельного вычисления необходимо принять ряд допущений.

1. Предположим, что время передачи данных по цепочке «сервер - НЖМД - клиент и обратно» зависит только от объёма передаваемых данных. Тогда можно построить простую схему прямого обмена с использованием условных каналов связи (рис.1). Здесь v_i (скорость условного канала, напрямую соединяющего сервер с i-ым клиентом) зависит как от скорости работы сети, так и от быстродействия НЖМД. Единица измерения скорости канала – [числ/сек], то есть количество вещественных чисел, переданных за секунду. Быстродействие клиента c_i – это совокупная величина, зависящая как от скорости ЭВМ, на которой запущен i-ый клиент, так и от ресурсов, выделенных клиенту операционной системой (приоритет, объём памяти, размещённой в ОЗУ и др.). Единица измерения быстродействия – [умн/сек], количество перемножений вещественных чисел в секунду.

Рис. 1. Упрощённая схема вычислительной системы.

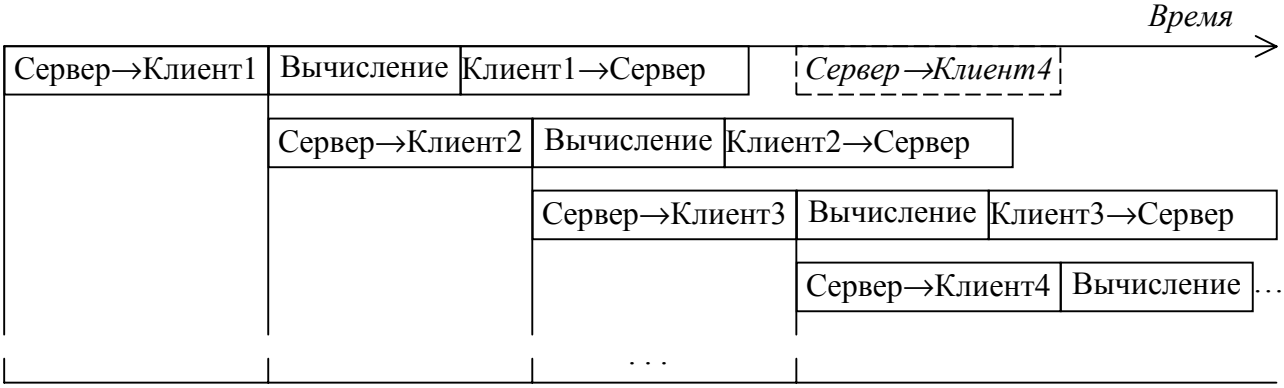


2. Для того, чтобы получить аналитические решения уравнений оптимальности и оценить порядок сложности, примем, что быстродействие *всех* клиентов одинаково, и скорость *всех* условных каналов тоже одинакова, иначе говоря, $c_1=c_2=\ldots=c_p=c$, и $v_1=v_2=\ldots=v_p=v$. Для большей наглядности модели условимся также, что перемножение подматриц будет производиться по классическому алгоритму перемножения.

3. Чтобы не ограничивать общность модели конкретной конфигурацией сети, положим, что количество клиентов p всегда достаточно для того, чтобы никакое новое задание не ожидало, пока для него освободится клиент. Для этого достаточно, чтобы $p \geq \ell^2$. В дальнейшем будет видно, что, во-первых, это неравенство вполне выполнимо на практике, и, во-вторых, что его можно сильно смягчить.

Сообразуясь с ограничениями 2 и 3, можно построить временную диаграмму работы процедуры перемножения.

Рис. 2. Временная диаграмма работы процедуры перемножения.



На рис.2 каждая «строка» описывает работу одного клиента. Выполнение задания клиентом состоит из трёх основных частей: загрузки задания, вычисления произведения и передачи результата. Из временной диаграммы видно, что время вычисления всех подматриц матрицы-произведения равно сумме времени, необходимого для загрузки всех заданий, и времени вычисления и возврата результата для задания, выполняющегося последним. Диаграмма показывает также, что необязательно иметь столько клиентов, сколько заданий (подматриц) образуется вследствие разбиения; в данном случае достаточно всего трёх клиентов, так как четвёртое задание может без потери времени взять клиент номер один (на диаграмме этот момент показан пунктиром).

Тривиальными рассмотрениями можно установить следующие равенства.

Время передачи одного задания клиенту от сервера $$T_{SC} = \frac{2nk}{v} = \frac{2n^2}{\ell v}. \qquad (1)$$

Время вычисления одного задания $$T_C = \frac{nk^2}{c} = \frac{n^3}{\ell^2 c}. \qquad (2)$$

Время передачи результата серверу от клиента $$T_{CS} = \frac{k^2}{v} = \frac{n^2}{\ell^2 v}. \qquad (3)$$

Просуммировав T_{SC} по всем заданиям, получим, что

суммарное время передачи всех заданий от сервера $$\tilde{T}_{SC} = \sum_{i=1}^{\ell^2} T_{SC} = \frac{2n^2\ell}{v}. \qquad (4)$$

Из (2) – (4), в соответствии с рис. 2, следует, что общее время вычисления произведения матриц размера ($n\times n$) в вычислительной системе составляет:

$$T_{общ} = \tilde{T}_{SC} + T_C + T_{CS} = \frac{2n^2\ell}{v} + \frac{n^3}{\ell^2 c} + \frac{n^2}{\ell^2 v} = n^2 \frac{2\ell^3 c + nv + c}{\ell^2 vc}. \qquad (5)$$

Из-за того, что доступ к НЖМД осуществляется последовательно, формула (5) справедлива только в том случае, если время вычисления одного задания больше, чем время передачи серверу от клиента $\left(\ell^2 - 1\right)$ блоков, содержащих результаты вычислений. Это требование, согласно (2) и (3), выражается следующим неравенством:

$$T_C \geq T_{CS}\left(\ell^2 - 1\right) \quad \Rightarrow \quad \ell \leq \sqrt{\left(n\frac{v}{c} + 1\right)}. \qquad (6)$$

Ниже будет показано, как можно добиться безусловного выполнения (6).

Введем обозначение для выражения, которое в дальнейшем будем называть числом или параметром вычислительной системы:

$$d = \left(n\frac{v}{c} + 1\right).$$

Можно теперь определить количество клиентских машин, которое было введено априорно в допущении №3 о достаточности количества клиентов. Из рис. 2 следует, что

$$p = \frac{T_C + T_{CS}}{T_{SC}} + 1 = \frac{nv + c}{2\ell c} + 1 = \frac{d}{2\ell} + 1. \qquad (7)$$

Выше уже упоминалось, что параметр ℓ полностью определяет разбиение общей задачи на подзадачи, распределяемые по клиентам. Зафиксировав в формуле (5) значения n, c и v, получим, что $T_{общ}=T_{общ}(\ell)$ [и, кстати говоря, $p=p(\ell)$ в (7)]. Таким образом, представляется возможным выбрать параметр разбиения ℓ в некотором смысле оптимальным образом для фиксированных параметров сети и размера задачи.

Особо интересными представляются два критерия оптимальности: ускорение и эффективность. Рассмотрим критерий оптимальности ускорения.

Ускорение понимается здесь как отношение времени выполнения последовательного алгоритма ко времени выполнения параллельного:

$$S = \frac{n^3}{c} : T_{общ} = \frac{nv\ell^2}{2\ell^3 c + nv + c} = \frac{\ell^2 (d-1)}{2\ell^3 + d}. \tag{8}$$

Оптимальное разбиение в смысле ускорения выглядит следующим образом:

$$\begin{cases} \left.\dfrac{\partial S}{\partial \ell}\right|_{\ell=\ell_a} = 0 \quad \Rightarrow \quad \ell_a = \sqrt[3]{\left(n\dfrac{v}{c}+1\right)} = \sqrt[3]{d}, \\ \left.\dfrac{\partial^2 S}{\partial \ell^2}\right|_{\ell=\ell_a} = -\dfrac{2}{3}\dfrac{nv}{(nv+c)} < 0 \quad \Rightarrow \quad \ell_a \text{ } \textit{является точкой максимума.} \end{cases} \tag{9}$$

Очевидно, что выбор $\ell = \ell_a$ приводит к безусловному выполнению неравенства (6), что гарантирует применимость модели.

Можно показать, что разбиение исходной задачи на ℓ_a^2 подзадач приводит также и к минимизации времени выполнения всей задачи. Это минимальное время равно

$$T_{мин} = T_{общ}(\ell_a) = 3\frac{n^2}{v} \cdot \sqrt[3]{\left(n\frac{v}{c}+1\right)} = 3\frac{n^2}{v}\ell_a. \tag{10}$$

Итак, определив выбор параметра разбиений $\ell = \ell_a$, мы получили алгоритм параллельного перемножения матриц размера (n×n) на локальной сети с условными параметрами c и v. Алгоритм оптимален в смысле ускорения, или времени решения задачи. Назовём его *быстрым* алгоритмом.

Преобразуем (10) так, чтобы увидеть порядок сложности этого алгоритма:

$$T_{мин} = 3\frac{n^2}{v} \cdot \sqrt[3]{\left(n\frac{v}{c}+1\right)} \approx 3\frac{n^2}{v}\sqrt[3]{\left(n\frac{v}{c}\right)} = \frac{3}{\sqrt[3]{(v^2 c)}} n^{2{,}333(3)}. \tag{11}$$

Эффективность параллельного алгоритма понимается как отношение ускорения к количеству процессоров (клиентских машин). Для нашей модели это

$$E = \frac{S}{p} = \frac{2n\ell^3 vc}{(2\ell^3 c + nv + c)(2\ell c + nv + c)}. \tag{12}$$

Эффективность быстрого алгоритма составляет

$$E_a = E(\ell_a) = \frac{2}{3}\left(\frac{nv/c}{2\ell_a + \ell_a^3}\right) \approx \frac{2}{3}\left(\frac{\ell_a^2}{\ell_a^2 + 2}\right). \tag{13}$$

Нетрудно убедиться, что $\dfrac{\partial E_a}{\partial \ell_a} > 0$ при любом $\ell_a \geq 1$, следовательно, E_a возрастает, причём при $n \to \infty$ (или, что то же, при $d \to \infty$) $E_a \to \dfrac{2}{3}$. Таким образом, эффективность быстрого алгоритма возрастает с увеличением размера задачи и не превышает величины 0.6(6). Это значит, что каждый «рабочий» в быстром алгоритме занят в среднем около 66% от общего времени решения задачи и выполняет за все время работы либо одну, либо две подзадачи.

Ну и, наконец, собственно ускорение алгоритма составляет

$$S_a = \frac{n^3}{c} : T_{мин} \approx \frac{1}{3}\left(\sqrt[3]{n\frac{v}{c}+1}\right)^2 = \frac{\ell_a^2}{3}. \tag{14}$$

Рассмотрим теперь критерий оптимальной эффективности:

$$\left.\frac{\partial E}{\partial \ell}\right|_{\ell=\ell_e} = 0 \quad \Rightarrow \quad \ell_e \approx 0.93 \cdot \sqrt{\left(n\frac{v}{c}+1\right)} + 0.33. \tag{15}$$

Для очень грубой оценки пригоден вид $\ell_e \approx \sqrt{d}$.

Чтобы при выборе $\ell = \ell_e$ обеспечить безусловное соблюдение условия (6), необходимо наложить дополнительные ограничения на соотношение между параметрами сети и размером задачи. Потребуем, чтобы

$$\sqrt{d} \geq 5. \tag{16}$$

Тогда, выбрав $\ell = \ell_e$, мы получим алгоритм параллельного перемножения матриц размера (n×n) на сети с условными параметрами c и v, оптимальный в смысле эффективности. Назовём его *эффективным* алгоритмом.

Подставив (15) в (5), можно оценить порядок этого алгоритма. Численная оценка даёт ω=2.47, для оценки сверху можно пользоваться значением ω=2.5.

Подставив (15) в (12), можно получить, что $\left[E_e = E\left(\ell_e\right)\right] \to 1$ при $d \to \infty$, то есть предельная эффективность равна единице.

Вопреки ожиданиям, максимум ускорения не совпадает с максимумом эффективности. Сравнив (15) и (9), нетрудно увидеть, что $\ell_e > \ell_a$. Согласно (9), переход от ℓ_a к ℓ_e, то есть от быстрого алгоритма к эффективному, приведёт к увеличению времени вычисления, и, соответственно, уменьшению ускорения. Однако из (7) следует, что необходимое количество клиентов также уменьшается при $\ell = \ell_e > \ell_a$: для достижения меньшего ускорения нужно меньшее количество клиентов. Максимум же отношения (ускорение/количество клиентов) как раз и достигается при $\ell = \ell_e$.

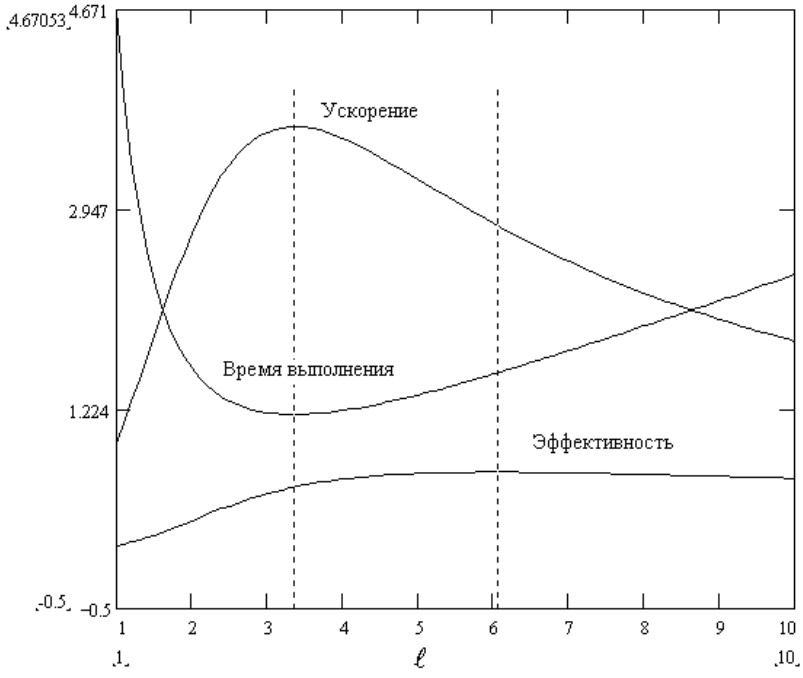


Рис. 3. Зависимость характеристик от параметра разбиения.

На графике (рис. 3), построенном для n = 1000 и параметров реальной сети (Ethernet 10MB/s, Pentium-166/Windows 95/98/NT4), видно, что время вычисления имеет точку минимума, соответствующую максимуму ускорения. Видно также, что максимум эффективности находится совсем в другой точке, где меньше количество клиентов и больше время счёта. В данном случае переход из максимума ускорения в максимум эффективности приводит к уменьшению ускорения в 1,35 раз, зато количество клиентов уменьшается в 1,6 раз, что обуславливает прирост эффективности.

При таком переходе происходит как бы «уплотнение» режима работы клиентов; относительное время работы каждого растёт, а вот их количество уменьшается, так как с увеличением значения ℓ подматрицы становятся меньше и для их обработки хватает меньшего количества клиентов. На рис. 4 показаны различия в динамике роста необходимого количества клиентов по мере увеличения размера задачи. Вообще, термин «оптимальная эффективность» можно понимать как «экономичность», так как точка оптимальной эффективности представляет собой компромисс между требованиями решить задачу быстрее и задействовать минимум клиентов.

Единичная предельная эффективность означает, что для больших задач на конечном (и существенно меньшем, чем нужно для быстрого алгоритма) количестве машин можно получить очень высокую степень параллелизма.

3. Заключение

1. Параллельное перемножение матриц может быть организовано с помощью самой обычной локальной сети, причём так, что характеристики процесса вычисления будут достаточно близки к оптимальным.
2. Два различных способа управления алгоритмом делают его оптимальным с точки зрения или максимума ускорения, или максимума эффективности. Эти состояния не совпадают, другими словами, максимальное ускорение не соответствует максимальной эффективности и наоборот. Утверждение, что чем больше исполнителей решают задачу, тем быстрее она будет решена, следует, таким образом, признать несостоятельным. Максимум занятости исполнителей не обязательно соответствует максимальному ускорению работы в целом, и наоборот.
3. Порядок алгебраической сложности алгоритма зависит от способа управления алгоритмом, а в рамках выбранного способа все изменения параметров сети влияют только на мультипликативную постоянную. Сам порядок в наилучшем своём значении составляет 2.33, а максимальная эффективность достигается ценой увеличения порядка сложности до 2.5.

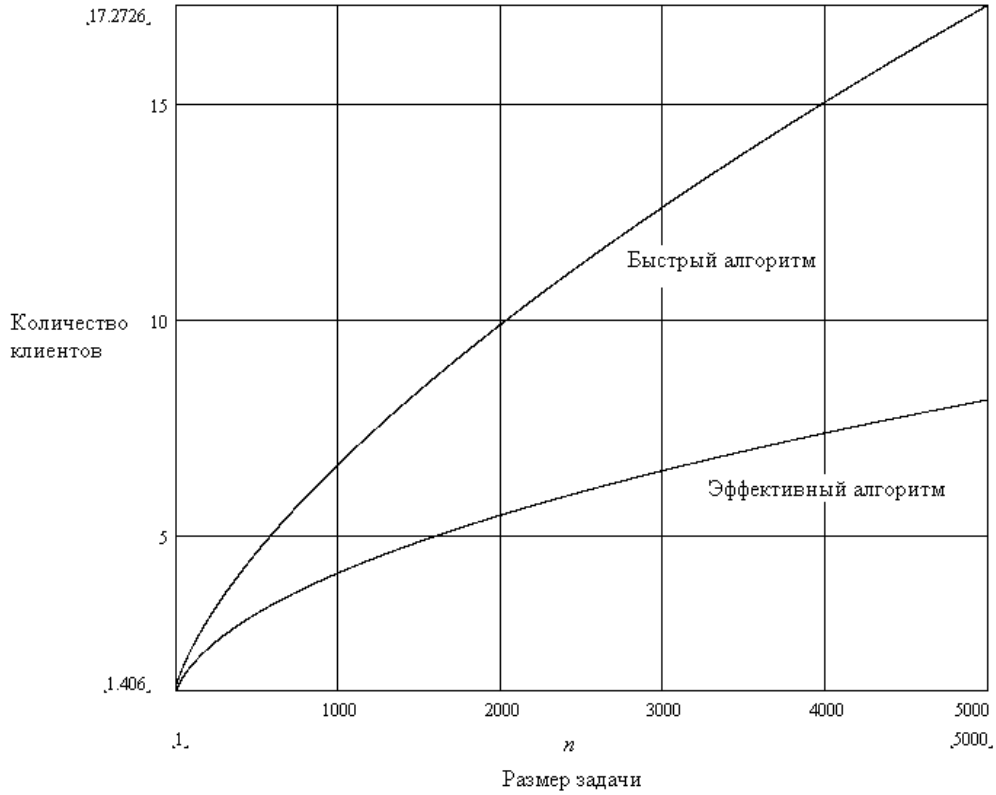


Рис. 4. Требования к количеству клиентов.

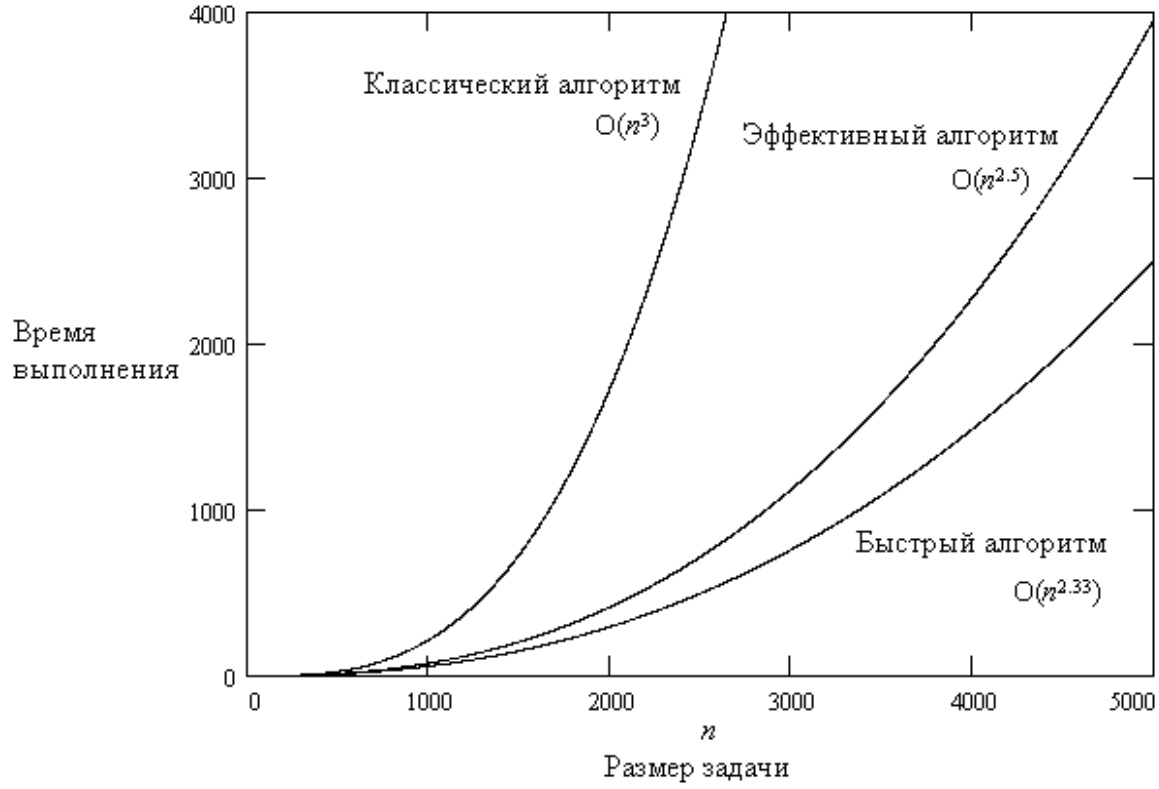


Рис. 5. Временная сложность рассмотренных алгоритмов.

Список литературы.

[1] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.:Мир, 1979.
[2] Лебедев В.И., Бахвалов Н.С. и др. Параллельные алгоритмы решения некоторых стационарных задач математической физики. – М.: Отдел вычислительной математики АН СССР, 1984.
[3] Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. – М.:Мир, 1991.
[4] Peter Bürgisser, Michael Clausen, M. Amin Shokrollahi. Algebraic complexity theory. – Springer, 1997.