RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ТЕОРЕТИЧЕСКИЕ ОСНОВАНИЯ ПРОГРАММНЫХ СИСТЕМ

УДК 519.687.1:004.75
10.25209/2079-3316-2026-17-2-83-101

Построение расписания при решении задачи поиска в Desktop Grid

Евгений Евгеньевич **Ивашко**[1✉], Илья Александрович **Чернов**[2]
[1,2] Институт прикладных математических исследований КарНЦ РАН
[1✉] *ivashko@krc.karelia.ru*

Аннотация. В статье рассмотрена математическая модель решения в Desktop Grid задачи поиска – перебора некоторого пространства дискретных объектов в поисках единственного, удовлетворяющего заданным условиям. На основе ранее полученных математических результатов представлена оптимальная стратегия поиска для случаев однородных и неоднородных по вычислительной сложности самих заданий и их повторных проверок; предложен новый метод построения расписания при решении задачи поиска в Desktop Grid. Важным практическим результатом при этом является механизм динамической репликации, обеспечивающий высокую доступную производительность. Приведены результаты численного моделирования, демонстрирующие преимущества предложенного расписания.

Ключевые слова и фразы: Desktop Grid, задача поиска, репликация, расписание

Для цитирования: Ивашко Е.Е., Чернов И.А. *Построение расписания при решении задачи поиска в Desktop Grid* // Программные системы: теория и приложения. 2026. **Т. 17**. № 2(71). С. 83–101. https://psta.psiras.ru/read/psta2026_2_83-101.pdf

© Ивашко Е.Е., Чернов И.А. 2026

Введение

Desktop Grid является одним из инструментов высокопроизводительных вычислений, наряду с вычислительными кластерами (суперкомпьютерами) и вычислительными Грид (Computational GRID). Благодаря высокой масштабируемости, такие системы способны достигать экзафлопсной производительности, что делает их востребованными для решения вычислительноемких задач.

Один из классов задач, эффективно решаемых в Desktop Grid, составляют задачи полного или частичного перебора некоторого дискретного пространства объектов с целью нахождения единственного экземпляра, обладающего заданными свойствами – «поиск иголки в стоге сена» или задача поиска. При этом, общепринятый подход к организации вычислений предполагает дублирование (репликацию) подзаданий, что значимо снижает доступную производительность. Цель представленной работы заключается в разработке нового метода построения оптимального расписания перебора объектов при решении задач поиска в Desktop Grid. Эффективность предложенного расписания обеспечивается учетом особенностей задачи в математической модели и формальным доказательством эффективности стратегии перебора, реализованной с помощью механизма динамической репликации. В статье предложен метод построения оптимального расписания перебора объектов при решении таких задач Desktop Grid.

Структура статьи следующая. В разделе 1 представлены базовые сведения о Desktop Grid как инструменте решения вычислительноемких задач. Раздел 2 содержит формальную постановку задачи поиска, обзор смежных работ и ранее полученных результатов. В разделе 3 описана оптимальная стратегия решения задачи поиска в общей и частных постановках. В разделе 4 предложен метод построения расписания подзаданий в Desktop Grid при решении задачи поиска, в разделе 5 представлены результаты численного моделирования.

1. Desktop Grid

Desktop Grid – это концепция, а также совокупность технологий и соответствующего программного обеспечения для организации распределенных вычислений с целью решения вычислительноемких задач, используя простаивающие ресурсы (*idle computing time*) неспециализированных невыделенных вычислителей (как правило, персональных компьютеров), объединенных посредством сети передачи данных общего назначения (как правило, Интернет).

Интерес к Desktop Grid обусловлен, в частности, большим вычислительным потенциалом таких систем. Так, в мире насчитывается более миллиарда персональных компьютеров, из которых более 100 млн. оснащено сопроцессорами GP GPU. Суммарная производительность, которую потенциально могут предоставить эти вычислительные устройства, составляет более 100 ExaFLOPS [1]. Например, с началом пандемии COVID-19 более 700 тысяч новых волонтеров присоединились к крупнейшему проекту добровольных вычислений Folding@home для поиска антивирусного агента против SARC-CoV-2. Такой всплеск интереса привел к быстрому росту производительности системы и сделал Folding@home первой экзафлопсной системой в мире, к тому же, превышающей по пиковой производительности суммарную производительность ста первых суперкомпьютеров мира [2]. На текущий момент существует порядка 60 публичных активных проектов Desktop Grid, объединяющих более 15 млн. вычислительных узлов по всему миру и обеспечивающих суммарную производительность в десятки PetaFLOPS [3], а добровольные вычисления являются важной частью мировой гражданской науки [4].

К достоинствам Desktop Grid относятся следующие:

- *высокая масштабируемость* вследствие простой архитектуры и отсутствия прямого взаимодействия между узлами;
- *большая потенциальная пиковая производительность* вследствие высокой масштабируемости и доступности вычислителей;
- *низкая стоимость создания и поддержки системы* вследствие отсутствия выделенных вычислительных узлов со специализированным программным и/или аппаратным обеспечением.

Также выделяют ряд недостатков:

- *низкая скорость обмена данными* из-за использования коммуникационной сети общего назначения (Интернет или локальная сеть);
- *низкая вычислительная мощность отдельных узлов*, каждый из которых, как правило, является рядовым персональным компьютером;
- *высокая аппаратная и программная гетерогенность* из-за большого разнообразия используемых вычислительных узлов;
- *низкая надежность узлов* из-за использования вычислителей добровольцев, аппаратного обеспечения общего назначения и невыделенных узлов;
- *недоверенность узлов* или высокая вероятность получения неправильного результата вследствие программных или аппаратных ошибок, а также возможного саботажа (намеренной передачи неправильного результата).

Фактически, из-за перечисленных недостатков, эффективное использование Desktop Grid ограничено задачами с независимым параллелизмом, где общая вычислительноемкая задача разбивается на множество подзаданий, решаемых независимо. К таким задачам, в частности, относят перебор некоторого дискретного пространства объектов.

Для борьбы с последними двумя указанными недостатками Desktop Grid используется механизм репликации: подзадание имеет несколько копий (*реплик*), каждое из которых выполняется на отдельном вычислительном узле, и результаты расчетов нескольких реплик сравниваются для получения «эталонного» результата подзадания. При необходимости (например, большое число ошибочных или потерянных подзаданий), количество реплик может быть увеличено непосредственно в процессе расчетов (механизм динамической репликации). Репликация значительно снижает доступную производительность Desktop Grid, поэтому при организации вычислений приходится решать проблему выбора коэффициента репликации, чтобы обеспечить требуемые параметры качества вычислительного процесса при сохранении высокой производительности.

2. Постановка задачи и обзор смежных работ

Большое число задач, решаемых в Desktop Grid, связано с «просеиванием» пространства допустимых решений для нахождения единственного объекта с требуемыми свойствами; такие задачи также называют «поиском иголки в стоге сена». Далее будем называть единственный объект с требуемыми свойствами *искомым*, а остальные – *обычными*.

Примером задачи «поиска иголки в стоге сена» является подбор пароля по известному хешу: для каждого допустимого пароля вычисляется его хеш, который сравнивается с заданным; искомый пароль считается найденным, когда вычисленный хеш совпадет с заданным. Подобная задача решается, например, в проекте распределенных вычислений на базе Desktop Grid *distributed.net* (подпроект RC5), направленном на атаку на симметричный блочный шифр RC5[1]. Другой пример – поиск перебором целочисленных корней системы уравнений (Диофантовых уравнений). Решение такой задачи было реализовано в подпроекте Euler (6,2,5) проекта добровольных вычислений Yoyo@home[2]. Еще один пример – это поиск перебором контр-примера для некоторой гипотезы, как это реализовано в проекте Collatz Conjecture[3].

[1] http://www.distributed.net/RC5/
[2] http://www.rechenkraft.net/yoyo/
[3] https://boinc.thesonntags.com/collatz/

Desktop Grid является хорошим инструментом для решения задач «поиска иголки в стоге сена» (далее – задача поиска), так как каждый элемент большого пространства допустимых решений проверяется независимо от других, а значит, при решении задачи могут быть задействованы все доступные узлы вычислительной системы.

Как уже отмечалось ранее, существенной особенностью Desktop Grid является возможность возврата узлом корректного, но неправильного результата вследствие программно-аппаратных особенностей или саботажа. При этом, механизм репликации, призванный бороться с такой проблемой, значимо снижает доступную производительность всей вычислительной сети. Использование следующих особенностей позволяет повысить вычислительную эффективность решения задачи поиска в Desktop Grid:

- пространство допустимых решений исследуется не полностью, а лишь до нахождения искомого объекта;
- ошибка первого рода (искомый объект ошибочно определен как обычный) ведет к существенным вычислительным потерям, в отличие от ошибки второго рода (обычный объект ошибочно определен как искомый) – при ошибке второго рода достаточно перепроверить один объект, в то время как при ошибке первого рода необходимо заново «просеивать» все пространство допустимых решений;
- при решении задачи интересует только искомый объект, результаты проверки обычного объекта неважны.

Решение задачи поиска с помощью Desktop Grid, учитывая возможную ошибку первого рода, сформулируем в следующей математической постановке.

Дано счетное число коробок с номерами $i = 0, 1, \ldots$ Коробка i содержит k_i шаров, общее число шаров составляет n: $\sum\limits_{i=0}^{\infty} k_i = n$. Один из шаров в коробке белого цвета (*искомый*), все остальные — черного (*обычные*). За один шаг из одной из непустых коробок i вслепую случайным образом достается один шар и определяется его цвет. Черный шар всегда верно определяется как черный, а белый шар может быть ошибочно определен как черный с вероятностью q. Если шар определен как черный, то он перекладывается в коробку $i + 1$; если найден искомый шар (белый), то процесс заканчивается.

Обозначим последовательность (распределение) шаров как $\{k\} = \{k\}_{i=0}^{\infty} = k_0, k_1, \ldots$. Когда из коробки $s : k_s > 0$ достается шар, который определен как черный (делается *шаг*), это изменяет

распределение на $k_0, \ldots, k_{s-1}, k_s - 1, k_{s+1} + 1, k_{s+2}, \ldots$, которое записывается как $\{k\}^s$. При этом, перестановка ходов не меняет распределения, т.е. $\{k\}^{s,t} = \{k\}^{t,s}$, если $k_s > 0$ и $k_t > 0$.

Требуется за минимальное ожидаемое число шагов найти искомый шар.

Рассматриваемая задача принадлежит к классу урновых, в которых реальные объекты моделируются как шары различных цветов, размещенные в нескольких урнах [5]. В качестве наиболее известного примера урновой задачи можно привести задачу о коллекционере купонов, где оценивается ожидаемое число купонов (шаров), которые нужно приобрести (достать), чтобы собрать всю коллекцию (т.е. чтобы каждый шар достали как минимум один раз, при этом шары возвращаются обратно в урну) (см., например, [6]). В задачах «о размещении частиц по ячейкам» шары случайным образом раскладываются в корзины и необходимо найти распределение числа шаров в корзинах (загрузку корзин) (см., например, [7]).

В работе [8] рассматривается задача поиска цели в известном числе мест (корзин) с известными вероятностями нахождения цели в них. Автором представлено аналитическое решение для случая, когда проверка корзины происходит безошибочно; доказательство выполнено по схеме сравнения стоимости перестановок в стратегии поиска). Для более общего случая, когда при проверке возможна ошибка, в статье приводится лишь численный пример.

В статье [9] рассматриваются три варианта задачи поиска с возможностью ошибки: поиск (1) на отрезке вещественной оси (необходимо определить цель с заданной точностью), (2) на ограниченном и (3) неограниченном дискретных множествах. Поиск заключается в вопросах и ответах, где вопросы задаются в форме неравенств, а получаемые ответы могут быть ошибочны.

В работе [10] изучается задача поиска человека в социальной сети. Структура графа социальной сети моделируется в виде урновой схемы, где урна – это вершина сети, а шары в урне – это смежные вершины. Несмотря на отличающуюся постановку, задача имеет схожее решение с представленной в данном разделе задачей поиска: необходимо проверить до конца наиболее вероятную урну, затем перейти к следующей наиболее вероятной.

Приведенные примеры демонстрируют, что урновые модели могут применяться для решения самых различных прикладных задач.

В работах [11–13] анализируется математическая модель задачи поиска, выводится и аналитически доказывается оптимальность стратегии поиска. В частности, в этих работах можно найти формальные доказательства тех математических результатов, которые приводятся в следующем разделе 3, а также формулы вычисления ожидаемого числа шагов до обнаружения искомого объекта при различных исходных постановках. В представленной статье на основе ранее полученных математических результатов предложена методика динамической репликации заданий в Desktop Grid.

3. Стратегия решения задачи поиска

Обозначим вероятность корректного определения белого шара за $p = 1 - q$. Отметим, что вероятность того, что искомый шар достигнет коробки j (т.е. не будет найден за j проверок) составляет q^j.

Для скорейшего нахождения искомого шара необходимо определить стратегию того, в какой последовательности проверять коробки. Более формально, *стратегия* – это правила $S(\{k\})$, определяющие для любого распределения шаров $\{k\} = \{k\}_{i=0}^{\infty}$ выбор коробки j, из которой достается шар. Требуется определить стратегию, позволяющую минимизировать ожидаемое число шагов до нахождения искомого шара.

Для любого начального распределения шаров стратегия определяет бесконечную *последовательность шагов*, прерываемую при нахождении искомого шара. Отметим, что после каждого хода изменяются вероятности $P_j(\{k\})$ нахождения шара в коробке j.

В каждый момент времени все свойства системы (такие как оптимальная стратегия, ожидаемое число шагов, вероятности нахождения шаров в коробках и т.д.) определяются только распределением шаров $\{k\}$. Таким образом, выполняется Марковское свойство: состояние системы зависит только от распределения шаров и не зависит от последовательности шагов, приведших к этому состоянию.

Далее, если не оговорено противное, будем считать, что начальное распределение в задаче поиска следующее: $k_0 = n$, $k_i = 0\ \forall i > 0$, т.е. изначально все шары находятся в коробке с номером 0.

Можно показать (см. [11–13]), что:

- оптимальная стратегия заключается в том, чтобы выбирать первую непустую коробку, т.е. $S(\{k\}) = \min\limits_i\{k_i > 0\}$;

- ожидаемое число шагов до нахождения искомого шара при использовании оптимальной стратегии составляет $E(\{n, 0, 0, \dots\}) = \frac{n+1}{2} + \frac{q}{p}n$;
- исходная постановка задачи предполагает, что все шары находятся в первой коробке, тогда, в ходе реализации оптимальной стратегии все шары из первой коробки последовательно перекладываются во вторую, и процесс повторяется заново – значит, на каждом шаге все шары будут распределены не более, чем по двум соседним коробкам, а ожидаемое число шагов вычисляется по формуле $E(\{u, m, 0, \dots\}) = \frac{up}{u+mq} \cdot \frac{u+1}{2} + \frac{nq}{u+mq}\left(u + \frac{n+1}{2} + \frac{q}{p}n\right)$, где $n = u + m$.

В исходной постановке задачи предполагается, что все проверки шаров имеют одинаковую стоимость (вычислительную сложность). Однако на практике встречаются задачи, в которых затраты на проверку зависят от:

- номера шара, т.е. от самого объекта: например, при поиске простых чисел в общем случае чем больше число, тем больше вычислительных ресурсов требуется для его проверки на простоту;
- от номера коробки, т.е. от числа уже выполненных проверок: например, в виртуальном скрининге лекарств повторная оценка энергии взаимодействия лиганда и белка-мишени может проводиться с большей точностью, что требует больше вычислительных ресурсов.

Пусть стоимость проверки шара m, находящегося в коробке j, составляет $C_{m,j}$. Будем считать, что выбор шара осуществляется не наугад, а целенаправленно. Упорядочим шары так, чтобы в каждой коробке стоимость проверки возрастала по m и положим $C_{1,0} = 1$. Заметим, что от «стоимости» шара не зависит его вероятность оказаться искомым, поэтому из выбранной коробки всегда берется самый «дешевый».

Оптимальная стратегия следующая [**13**]: при распределении $\{k\}$, выбирать шар из коробки t такой, что выполняется следующее условие:

$$\frac{C_{m_t,t}}{q^t} < \frac{C_{m_s,s}}{q^s}$$

для всех $s \neq t$.

Если проверки одинаковы по стоимости, т.е.

$$\frac{C_{m_t,t}}{q^t} = \frac{C_{m_s,s}}{q^s},$$

то может быть выбрана любая коробка.

3.1. Если стоимость проверки зависит от номера шара

Рассмотрим частный случай задачи, при котором стоимость проверки зависит только от номера шара, но не от номера коробки, т.е. $C_{m,s} = C_m$.

Из доказанных ранее результатов следует:

- после первого шага (проверка шара 1 из коробки 0) необходимо решить, проверять ли снова этот же шар или перейти к шару 2; условие перехода к другому шару: $C_2 q < C_1$, т.е. $C_2 < q^{-1}$.
- Стратегия «брать шар из первой непустой коробки» является оптимальной для постоянной стоимости ($C_m \equiv 1$) и остается оптимальной для C_m, медленно растущей по m. Под «медленно растущей» здесь и далее понимается «в сравнении с q^{1-m}». Тогда для «медленно растущей» C_m нужно один раз проверить шар 1, после чего один раз проверить шар 2, и т.д. до последнего шара n; когда коробка 0 опустеет, все шары окажутся в коробке 1, после перенумерации проверки можно начать заново по той же стратегии.
- Пусть $C_m = q^{1-m}$. Тогда после n шагов (при условии, что искомый шар еще не был найден) шар $m-$единственный в коробке $n - m$. Дальнейшая оптимальная стратегия следующая: проверить шар 1, затем шары 1 и 2 (в любом порядке), затем шары 1, 2 и 3 в любом порядке и т.д.

Положим, что в случае неопределенности (когда два или более шара подходят для выбора в соответствии с оптимальной стратегией) выбирается шар с наибольшим номером: т.е. если нет разницы в выборе шара 1 из коробки 1 или шара 2 из коробки 0, то выбирается шар 2. Оптимальная стратегия для быстро растущей C_m выглядит следующим образом (далее h_m обозначает коробку, содержащую шар m):

(1) проверять шар 1 до тех пор, пока его «эффективная стоимость» $C_1 q^{-h_1}$ не превышает C_2,

(2) проверить один раз шар 2,

(3) снова проверить шар 1,

(4) повторять шаги 1, 2 до тех пор, пока не станет $C_1 q^{h_1} > C_3$ или $C_2 q^{h_2} > C_3$,

(5) проверить один раз шар 3,

(6) повторять процедуру, пока к циклу не присоединится шар 4,

(7) и т.д.

3.2. Если стоимость проверки зависит от коробки

Теперь рассмотрим частный случай задачи, в котором стоимость проверки не зависит от номера шара, но зависит от номера коробки:

$$C_{m,s} = G_s.$$

Если G_s является убывающей последовательностью, то дальнейшая проверка того же шара дешевле, чем проверка нового. Условие оптимальности стратегии – это выбирать такой шар t, для которого

$$\frac{G_t}{q^t} < \frac{G_s}{q^s}$$

для всех $s \neq t$. В частности, стратегия «брать шар из первой непустой коробки» является оптимальной для любой неубывающей и для медленно убывающей последовательности G_s, такой, что $G_s q^{-s}$ не убывает. Однако для быстро убывающей последовательности G_s оптимальная стратегия становится парадоксальной и бесполезной: проверять шар 1 бесконечное число раз, т.е. после первой проверки шар перекладывается в следующую коробку, где стоимость проверки настолько низка, что повторная проверка выгоднее, чем проверка нового шара.

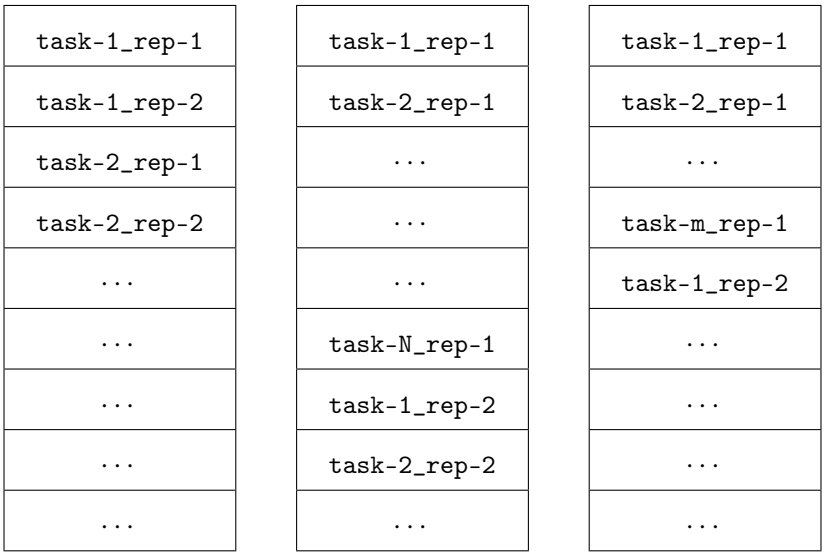
4. Построение расписания

Задача построения расписания является частным случаем задачи планирования, определяя порядок исполнения заявок без учета ограничений по ресурсам и времен освобождения отдельных вычислительных узлов. Выстраивание оптимальной последовательности заданий позволяет повысить эффективность вычислительной системы при решении отдельных классов задач. Представленные выше математические результаты позволяют оптимизировать расписание заданий в Desktop Grid при решении задачи поиска.

Рисунок 1 схематически представляет примеры расписаний при решении задачи поиска:

Вариант (*а*): стандартный подход Desktop Grid предполагает установление коэффициента репликации (в примере он равен 2) для независимого решения одного и того же задания на различных узлах;

Вариант (*б*): при однородных по вычислительной сложности заданиях при безуспешном завершении всех вычислений расписание полностью повторяется;



(*а*) стандартный подход

(*б*) оптимальное с однородными заданиями

(*в*) оптимальное с неоднородными заданиями

Рисунок 1. Примеры расписаний

Вариант (*в*): при неоднородности по вычислительной сложности, реплика завершенного задания вставляется в очередь, отсортированную по сложности.

На схеме подзадания нумеруются в соответствии с шаблоном $task - N_{rep} - M$, где N – номер задания, а M – номер реплики этого задания.

Метод составления оптимального расписания определяется тем, одинакова ли вычислительная сложность всех заданий и растет ли вычислительная сложность при повторной проверке. В простейшем случае, при полной однородности, подзадания выбираются из базы данных по очереди (в произвольном порядке), механизм репликации не задействуется. При исчерпании заданий, база данных повторно наполняется теми же подзаданиями, и процесс расчетов повторяется в тех же условиях пока не будет найден искомый элемент (который существует и будет найден за конечное время по условиям задачи). Ниже представлен алгоритм управления расписанием для случая однородных заданий.

Входные данные: набор заданий $task_1, task_2, ..., task_n$.

Шаг 1: составление расписания – списка заданий $N = \{task_1, task_2, ..., task_n\}$.

Шаг 2: распределение заданий по вычислительным узлам: узлу i назначается задание $task_i$ из списка N.

Шаг 3: ожидание результата расчета.

Шаг 3.1: получен результат расчета задания $task_k$ от узла m.

Шаг 3.2: $task_k$ – искомый объект? Если да, то переход на *Шаг 4*.

Шаг 3.3: добавление в конец списка N задания $task_k$.

Шаг 3.4: назначение узлу m первого задания из списка N. Переход на *Шаг 3*.

Шаг 4: вывод искомого объекта $task_k$.

Выходные данные: $task_k$ – искомый объект.

В случае различной вычислительной сложности, подзадания упорядочиваются от наиболее простых к наиболее сложным. При получении каждого из результатов расчетов, динамически генерируется реплика подзадания и оценивается ее вычислительная сложность. Новая реплика занимает свое место в расписании – упорядоченной по сложности последовательности заданий. По построению, такой метод формирования расписания соответствует оптимальной стратегии поиска, описанной в разделе 3. Ниже представлен алгоритм управления расписанием для случая неоднородных заданий.

Входные данные: набор заданий $task_1, task_2, ..., task_n$.

Шаг 1: формирование оценок вычислительной сложности $C_1, C_2, ..., C_k$ для заданий $task_1, task_2, ..., task_n$.

Шаг 2: составление расписания – упорядоченного по вычислительной сложности списка заданий $N = \{task_1^*, task_2^*, ..., task_n^*\}$ такого, что $C_{j-1}^* \le C_j^* \le C_{j+1}^*$.

Шаг 3: распределение заданий по вычислительным узлам: узлу i назначается задание $task_i^*$ из списка N.

Шаг 4: ожидание результата расчета.

Шаг 4.1: получен результат расчета задания $task_k^*$ от узла m.

Шаг 4.2: $task_k^*$ – искомый объект? Если да, то переход на *Шаг 5*.

Шаг 4.3: пересчет сложности задания $task_k^*$ по формуле $C_k^* = C_k^*/q$.

Шаг 4.4: включение задания $task_k^*$ в расписание N так, чтобы $C_{j-1}^* \le C_k^* \le C_j^*$.

Шаг 4.5: назначение узлу m первого задания из списка N. Переход на *Шаг 5*.

Шаг 5: вывод искомого объекта $task_k^*$.

Выходные данные: $task_k^*$ – искомый объект.

В Desktop Grid не предполагается существования механизма передачи сообщения от сервера к вычислительным узлам, поэтому, несмотря на то, что искомый объект уже найден, вычислительные узлы будут продолжать решение уже полученных неактуальных заданий.

5. Результаты численного моделирования

Для оценки преимуществ разработанного расписания, было выполнено численное моделирование. Эксперименты проводились с помощью специально разработанной программы имитационного моделирования решения задачи поиска. Полученные результаты усреднялись на 10 млн. итераций при числе объектов $N = 100$ и различной вероятности ошибки. В таблице 1 и на графике 2 представлено сравнение результатов моделирования.

Таблица 1. Среднее число проверок $N = 100$ *равной сложности* объектов до нахождения искомого в зависимости от коэффициента репликации k и вероятности q

q	0.01	0.05	0.1	0.2	0.3	0.4	0.45
$k = 1$	51.51	55.77	61.62	75.49	93.91	117.14	132.3
$k = 2$	100.02	100.56	102.14	108.5	119.97	138.32	151.1
$k = 3$	149.53	149.6	149.91	152.17	158.17	170.48	180.09

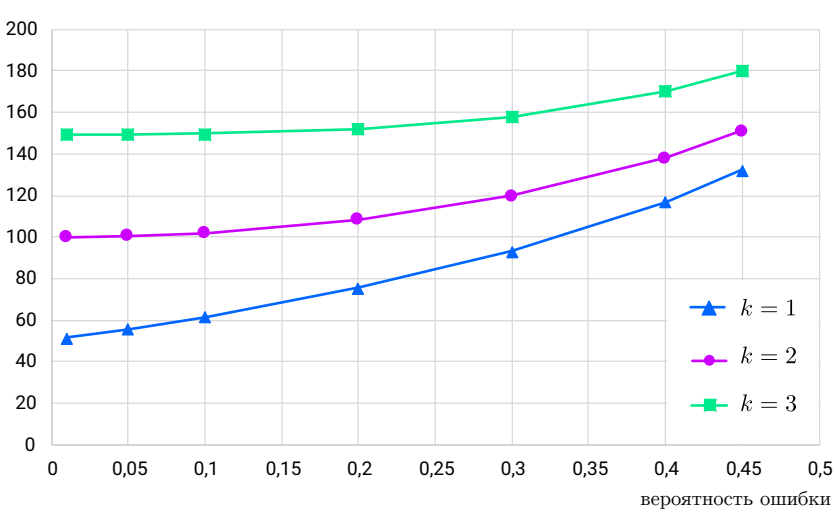


Рисунок 2. Зависимость среднего числа проверок $N = 100$ *равной сложности* до нахождения искомого от вероятности q при различных коэффициентах репликации k

В таблице 2 и на графике 3 сравниваются оптимальное расписание ($k = 1$) со стандартным подходом при неоднородных объектах (вычислительная сложность задания равняется его номеру).

Таблица 2. Сравнение среднего числа проверок $N = 100$ объектов *с равномерно возрастающей сложностью проверки* до нахождения искомого при различных значениях вероятности q и коэффициентах репликации k

q	0.01	0.05	0.1	0.2	0.3	0.4	0.45
$k = 1$	1768.37	1982.43	2277.9	2980.67	3879.6	5083.49	5849.89
$k = 2$							
без сортировки	5051.25	5074.48	5137.86	5413.62	5911.12	6723.16	7291.7
с сортировкой	3384.92	3412.29	3488.02	3813.09	4397.66	5322.77	5960.05
$k = 3$							
без сортировки	7548.75	7553.17	7569.89	7663.04	7919.7	8447.52	8867.85
с сортировкой	5053.07	5054.81	5071.33	5183.74	5485.28	6108.03	6596.56

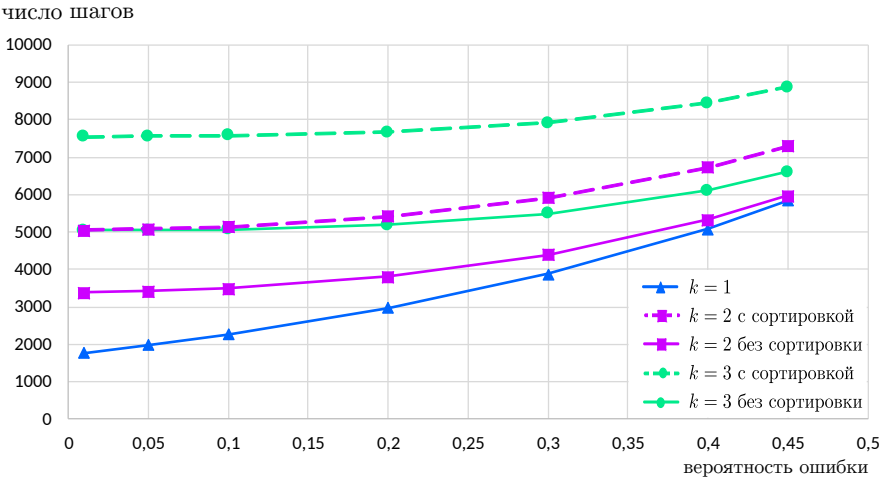


Рисунок 3. Зависимость среднего числа проверок $N = 100$ объектов *с равномерно возрастающей сложностью проверки* до нахождения искомого от вероятности q при различных коэффициентах репликации k

Как видно из представленных результатов, при однородных заданиях

стратегия без репликации использует от 10% (25%) до 200% (300%) меньшее число шагов до нахождения искомого объекта по сравнению со стандартным подходом с репликацией $k = 2$ ($k = 3$) в зависимости от вероятности ошибки. Причем, чем менее вероятна ошибка идентификации объекта, тем более невыгодным становится использование репликации.

Для неоднородных заданий величина выигрыша в использовании оптимального расписания существенно зависит (как и конечный вид самого расписания) от вычислительной сложности объектов. При этом простейший способ улучшения стандартных расписаний с репликациями заключается в сортировке объектов по их сложности, что обусловлено спецификой задачи.

6. Заключение

Задачи «поиска иголки в стоге сена» – перебора некоторого пространства дискретных объектов в поисках единственного, удовлетворяющего заданным условиям, – являются важным классом задач, часто требующих привлечения инструментов высокопроизводительных вычислений.

Desktop Grid являются хорошим инструментом решения задач поиска, так как обеспечивают высокую масштабируемость для задач с независимым параллелизмом. Однако специфика Desktop Grid приводит к довольно высокой вероятности получения неправильного результата вследствие программно-аппаратной гетерогенности или саботажа. Репликация позволяет бороться с этим недостатком, однако приводит к значительному снижению доступной производительности расчетов.

В статье рассмотрена математическая модель решения задачи поиска в Desktop Grid. На основе ранее полученных математических результатов, представлены оптимальные стратегия поиска для случаев однородных и неоднородных по вычислительной сложности самих заданий и их повторных проверок. На основе представленных оптимальных стратегий предложен метод построения расписания в Desktop Grid при решении задачи поиска. Важным практическим результатом при этом является то, что при однородных подзаданиях необходимо отказаться от репликации, а при неоднородных – использовать механизм динамической репликации, обеспечивающий высокую доступную производительность. Предложенный метод демонстрирует свои преимущества перед стандартными расписаниями с репликациями, обеспечивая значительный выигрыш в числе шагов до обнаружения искомого объекта.

Ограничением используемого подхода является предположение об известной вычислительной сложности заданий и их повторной проверки, которое однако, в ряде случаев может обеспечиваться набором необходимой статистики.

Список использованных источников

[1] Cérin C., Fedak G. (eds.) *Desktop Grid Computing.*– V. 4.– Boca Raton: CRC Press.– 2012.– ISBN 978-0367381189.– 388 pp. ↑85

[2] Zimmerman M. I., Bowman G. *SARS-CoV-2 simulations go exascale to capture spike opening and reveal cryptic pockets across the proteome* // Biophysical Journal.– 2021.– Vol. **120**.– No. 3, Supplement 1.– id. 299a. ↑85

[3] Ivashko V., Ivashko E. *BOINC-based volunteer computing projects: dynamics and statistics* // *Supercomputing. RuSCDays 2022* (Moscow, Russia, 26–27 September 2022), Lecture Notes in Computer Science.– vol. **13708**, Cham: Springer.– 2022.– ISBN 978-3-031-22940-4.– Pp. 619–631. ↑85

[4] Литовченко В. С., Ивашко Е. Е. *Проекты добровольных вычислений в гражданской науке: динамика и статистика* // Информационные технологии и вычислительные системы.– 2024.– № 1.– С. 11–22. ↑85

[5] Johnson N. L., Kotz S. *Urn Models and Their Application: An Approach to Modern Discrete Probability Theory.*– New York–London: Wiley.– 1977.– ISBN 9780471446309.– xiii+402 pp. ↑88

[6] Anceaume E., Busnel Y., Sericola B. *New results on a generalized coupon collector problem using Markov chains* // Journal of Applied Probability.– 2015.– Vol. **52**.– No. 2.– Pp. 405–418. ↑88

[7] Энатская Н. Ю., Хакимуллин Е. Р., Колчин А. В. *Анализ схемы размещения неразличимых частиц по неразличимым ячейкам* // Труды Карельского научного центра Российской академии наук.– 2014.– № 4.– С. 143–154. ↑88

[8] Craswell K. J. *How to find a needle in a haystack* // The Two-Year College Mathematics Journal.– 1973.– Vol. **4**.– No. 3.– Pp. 18–22. ↑88

[9] Pelc A. *Searching with known error probability* // Theoretical Computer Science.– 1989.– Vol. **63**.– No. 2.– Pp. 185–202. ↑88

[10] Marks C. E., Zaman T. *A multi-urn model for network search.*– 2016.– 38 pp. arXiv 1608.08080 ↑88

[11] Chernov I., Ivashko E. *622–633* // *Supercomputing. RuSCDays 2020* (Moscow, Russia, 21–22 September 2020), Communications in Computer and Information Science.– vol. **1331**, Cham: Springer.– 2020.– ISBN 978-3-030-64615-8. ↑89

[12] Ivashko E., Chernov I. *Search for an object with a recognition error probability in a desktop grid environment* // *2021 Ivannikov Ispras Open Conference (ISPRAS)* (Moscow, Russian Federation, 02–03 December 2021).– IEEE.– 2021.– ISBN 9781665423304.– Pp. 96–99. ↑89

[13] Ivashko E., Chernov I. *Search for an object with a recognition error by a desktop grid // Mathematics and its Applications in New Computer Systems. MANCS 2021*, Lecture Notes in Networks and Systems.– vol. **424**, Cham: Springer.– 2021.– ISBN 978-3-030-97019-2.– Pp. 207–217. ↑89, 90

Поступила в редакцию 16.02.2026;
одобрена после рецензирования 03.04.2026;
принята к публикации 03.04.2026;
опубликована онлайн 05.05.2026.

Рекомендовал к публикации *к.т.н. В. П. Фраленко*

Информация об авторах:

Евгений Евгеньевич Ивашко

К. ф.-м. н., с.н.с. Института прикладных математических исследований Карельского научного центра РАН, г. Петрозаводск. Область научных интересов: высокопроизводительные вычисления, распределенные вычисления, добровольные вычисления, Desktop Grid.

0000-0001-9194-3976
e-mail: ivashko@krc.karelia.ru

Илья Александрович Чернов

д.т.н., с. н. с. Института прикладных математических исследований КарНЦ РАН, доцент Кафедры математического анализа Петрозаводского государственного университета. Область научных интересов: математическое моделирование физических процессов, краевые задачи математической физики, крупномасштабная динамика моря, вычислительная математика.

0000-0001-7479-9079
e-mail: chernov@krc.karelia.ru

Авторы внесли равный вклад в подготовку публикации.

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article THEORETICAL FOUDATIONS OF SOFTWARE SYSTEMS

UDC 519.687.1:004.75
10.25209/2079-3316-2026-17-2-83-101

Scheduling for solving a search problem in a Desktop Grid

Evgeny **Ivashko**[1], Ilya **Chernov**[2]
[1,2] Institute of Applied Mathematical Research KarRC of RAS
[1]✉*ivashko@krc.karelia.ru*

Abstract. The article investigates a mathematical model for solving a search problem in Desktop Grid — the enumeration of a certain space of discrete objects to find the single one that satisfies given conditions. Based on previously obtained mathematical results, an optimal search strategy is presented for cases where the computational complexity of the tasks themselves and their rechecks is homogeneous and heterogeneous; a method for constructing a schedule for solving the search problem in Desktop Grid is proposed. An important obtained practical result is the mechanism of dynamic replication, which ensures high available performance. The results of numerical simulations demonstrating the advantages of the proposed schedule are presented. (*In Russian*).

Key words and phrases: Desktop Grid, search problem, replication, scheduling

2020 *Mathematics Subject Classification*: 68Q85; 68M20, 68P10

For citation: Evgeny Ivashko, Ilya Chernov. *Scheduling for solving a search problem in a Desktop Grid*. Program Systems: Theory and Applications, 2026, **17**:2(71), pp. 83–101. (*In Russ.*). https://psta.psiras.ru/read/psta2026_2_83-101.pdf

© Ivashko E., Chernov I. 2026

References

[1] C. Cérin, G. (eds.) Fedak. *Desktop Grid Computing*. V. 4, CRC Press, Boca Raton, 2012, ISBN 978-0367381189, 388 pp.

[2] M. I. Zimmerman, G. Bowman. "SARS-CoV-2 simulations go exascale to capture spike opening and reveal cryptic pockets across the proteome", *Biophysical Journal*, **120**:3, Supplement 1 (2021), id. 299a.

[3] V. Ivashko, E. Ivashko. "BOINC-based volunteer computing projects: dynamics and statistics", *Supercomputing. RuSCDays 2022* (Moscow, Russia, 26–27 September 2022), Lecture Notes in Computer Science, vol. **13708**, Springer, Cham, 2022, ISBN 978-3-031-22940-4, pp. 619–631.

[4] V. S. Litovchenko, E. E. Ivashko. "Volunteer computing projects in citizen science: dynamics and statistics", *Informacionnye texnologii i vychislitel'nye sistemy*, 2024, no. 1, pp. 11–22 (in Russian).

[5] N. L. Johnson, S. Kotz. *Urn Models and Their Application: An Approach to Modern Discrete Probability Theory*, Wiley, New York–London, 1977, ISBN 9780471446309, xiii+402 pp.

[6] E. Anceaume, Y. Busnel, B. Sericola. "New results on a generalized coupon collector problem using Markov chains", *Journal of Applied Probability*, **52**:2 (2015), pp. 405–418.

[7] N. Yu. Enatskaya, E. R. Xakimullin, A. V. Kolchin. "Analysis of a scheme of allocating indistinguishable particles to indistinguishable cells", *Trudy Karel'skogo nauchnogo centra Rossijskoj akademii nauk*, 2014, no. 4, pp. 143–154 (in Russian).

[8] K. J. Craswell. "How to find a needle in a haystack", *The Two-Year College Mathematics Journal*, **4**:3 (1973), pp. 18–22.

[9] A. Pelc. "Searching with known error probability", *Theoretical Computer Science*, **63**:2 (1989), pp. 185–202.

[10] C. E. Marks, T. Zaman. *A multi-urn model for network search*, 2016, 38 pp. arXiv:1608.08080

[11] I. Chernov, E. Ivashko. "622–633", *Supercomputing. RuSCDays 2020* (Moscow, Russia, 21–22 September 2020), Communications in Computer and Information Science, vol. **1331**, Springer, Cham, 2020, ISBN 978-3-030-64615-8.

[12] E. Ivashko, I. Chernov. "Search for an object with a recognition error probability in a desktop grid environment", *2021 Ivannikov Ispras Open Conference (ISPRAS)* (Moscow, Russian Federation, 02–03 December 2021), IEEE, 2021, ISBN 9781665423304, pp. 96–99.

[13] E. Ivashko, I. Chernov. "Search for an object with a recognition error by a desktop grid", *Mathematics and its Applications in New Computer Systems. MANCS 2021*, Lecture Notes in Networks and Systems, vol. **424**, Springer, Cham, 2021, ISBN 978-3-030-97019-2, pp. 207–217.