DOI: 10.15514/ISPRAS–2020–32(4)–9

Метрики эффективности и производительности при использовании эволюционного алгоритма на грид-системах из персональных компьютеров

Н.П. Храпов, ORCID: 0000-0002-8038-7625 <nkhrapov@gmail.com>
Институт проблем передачи информации им. А.А. Харкевича РАН,
127051, Россия, г. Москва, Большой Каретный пер., д. 25, стр. 1

Аннотация. Область применения систем добровольческих вычислений постоянно расширяется. Существует много научных работ по адаптации различных вычислительных алгоритмов к ГСПК. Темой представленной работы является эффективная адаптация эволюционного алгоритма к системам добровольческих вычислений. Рассматриваются причины потерь производительности, предлагаются критерии и метрики оценки качества работы алгоритма. Рассматриваемые метрики могут быть использованы для сравнительного анализа различных политик планирования заданий при проведении вычислений. Вводимые метрики могут быть посчитаны как на имитационных моделях, так и в процессе проведения практических вычислений.

Ключевые слова: грид-системы; распределённые вычисления; эволюционный алгоритм; эффективность; производительность

Для цитирования: Храпов Н.П. Метрики эффективности и производительности при использовании эволюционного алгоритма на грид-системах из персональных компьютеров. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 133–140. DOI: 10.15514/ISPRAS–2020–32(4)–9

Благодарности: Работа выполнена при поддержке гранта РФФИ №19-07-00911

Metrics of efficiency and productivity when using the evolutionary algorithm on desktopgrid

N.P. Khrapov, ORCID: 0000-0002-8038-7625 <nkhrapov@gmail.com>
Kharkevich Institute for Information Transmission Problems of the RAS,
19, Bolshoy Karetny per., Moscow, 127051, Russia

Abstract. The scope of voluntary computing systems is constantly expanding. The BOINC system is currently the most famous for organizing volunteer computing. There are many scientific papers on the adaptation of various computational algorithms to the BOINC. The topic of the presented work is the effective adaptation of the evolutionary algorithm to voluntary computing systems. The reasons for productivity losses are considered, criteria and metrics for evaluating the quality of the algorithm are introduced. The content of the article consists of two main parts. The first part of the article discusses general metrics that can be used to assess the performance of various computational algorithms within BOINC. The problem of adaptation of the evolutionary algorithm is considered in the second part of the article. Two main problem-specific causes of productivity loss are considered: the effect of waiting for the last job and the effect of a common queue. Methods are proposed for quantitative assessment of the influence of various systemic effects on the performance of an evolutionary algorithm. The proposed metrics can be used in a comparative analysis of various job scheduling policies when performing calculations. The metrics can be calculated both during the actual and simulated calculations.

Keywords: grid systems; distributed computing; evolutionary algorithm; efficiency; performance

For citation: Khrapov N.P. Metrics of efficiency and productivity when using the evolutionary algorithm on desktopgrid. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 133–140 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–9

Acknowledgements. The author thanks Russian Foundation for Basic Research (grant №19-07-00911) for financial support.

1. Введение

Технологии грид-систем из персональных компьютеров (далее – ГСПК) существенно повышают доступность вычислительных ресурсов для научных коллективов за счёт привлечения компьютеров добровольцев. Принцип организации вычислений посредством ГСПК состоит в том, что вычислительный узел запрашивает задание у сервера проекта, скачивает и запускает его на исполнение. По окончанию расчётов результат вычислений отправляется обратно на сервер.

Существует множество реализаций программного обеспечения для организации добровольческих вычислений. Наибольшую популярность к настоящему моменту приобрёл программный пакет BOINC (Berkeley Open Infrastructure for Network Computing) [1]. Проект разработки ПО BOINC успешно развивается, начиная с 2004-го года. Технические возможности системы BOINC постоянно расширяются. Существуют расширения, позволяющие объединять в единую инфраструктуру компьютеры добровольцев с кластерами и суперкомпьютерами [2]. Широкое распространение получают модификации системы BOINC, предназначенные для интеграции вычислительных ресурсов на корпоративном уровне [3].

Многие вычислительные алгоритмы, изначально предназначенные для функционирования на параллельных системах, адаптируются к системе BOINC. Проведён ряд исследований по адаптации метода ветвей и границ к ГСПК [4]. Грид-системы из персональных компьютеров успешно используются для решения задач криптоанализа [5]. Описание решения задачи о графах при помощи системы BOINC приведено в [6].

При портировании вычислительного алгоритма на ГСПК необходимо учитывать как специфику самого алгоритма, так и особенности функционирования инфраструктуры. Практика адаптации существующих алгоритмов и ПО к ГСПК показывает, что алгоритм, эффективно функционирующий на вычислительном ресурсе с системой очередей, может неэффективно работать на ГСПК. Для повышения эффективности функционирования эволюционного алгоритма необходимо предварительно выявить его особенности, связанные с организацией вычислений, а также рассмотреть, как эти особенности соотносятся со спецификой функционирования ГСПК.

Практика решения задач оптимизации посредством эволюционного алгоритма (далее – ЭА) ранее описывалась в [7]. Особенности работы ЭА, влияющие на эффективность выполнения на ГСПК рассмотрены в [8]. Темой данной работы является более глубокий анализ причин потери производительности и введение оценивающих метрик для дальнейших работ по выработке оптимальных стратегий управления ресурсами.

2. Метрики производительности и эффективности в ГСПК в общем случае

При анализе источников потерь производительности необходимо учитывать критерии производительности и эффективности вычислительной системы. Для различных решаемых прикладных задач различные критерии является более важными. Рассмотрим два основных критерия производительности:

2.1 Время решения прикладной задачи полностью

Прикладная задача разбивается на некоторое количество независимых друг от друга подзаданий. В процессе вычислений дополнительные подзадания могут генерироваться на основе результатов посчитанных подзаданий. Временем решения прикладной задачи в данном случае будет время между началом расчётов на стороне эволюционного алгоритма и получением результата последнего подзадания.

Рассмотрим возможные метрики в рамках данного подхода. В системах параллельных вычислений Makespan – время между отправкой первого подзадания и получением результата последнего подзадания:

$$M_S = t_{stop} - t_{strart}.$$

Практика проведения вычислений на BOINC показывает, что между началом расчётов эволюционным алгоритмом и отправкой первого задания на вычислительный узел может пройти несколько дней, если очередь заданий на сервере достаточно велика. Поэтому использование параметра *Ms* для системы BOINC потребует важного уточнения, какой момент времени считать началом вычислений: момент поставки первого задания в очередь на сервере или момент отправки первого задания на вычислительный узел. Сложность в детерминировании времени решения задачи обусловлена различием принципов работы системы очередей и ГСПК.

В системе очередей прикладной задаче выделяется необходимый ресурс сразу для всех подзаданий полностью при достижении прикладной задачей своей очереди. Таким образом, время между началом генерации подзаданий эволюционным алгоритмом и получением последнего результата в системе очередей состоит из времени ожидания в очереди и временем выполнения вычислений. Время ожидания в очереди зависит от состояния очереди. А время выполнения вычислений зависит от специфики решаемой задачи и характеристик предоставляемого ресурса.

Отличительной особенностью системы BOINC является единая очередь для всех подзаданий. Это означает, что новые подзадания, генерируемые на основе результатов предыдущих, будут поставлены в конец очереди аналогично первому заданию. Поэтому в системе BOINC состояние очереди заданий влияет как на время отправки первого задания, так и на время проведения непосредственных вычислений. Важно учитывать, что в инфраструктуре ГСПК время решения прикладной задачи не является статической характеристикой. Это означает, что если для одной и той же задачи начать расчёты в различные моменты времени, то продолжительность вычислений может быть различной. Время проведения вычислений будет зависеть от готовности вычислительных узлов, предоставляемых добровольцами, и от загруженности очереди подзаданиями других вычислительных задач. Учитывая, что состояние очереди влияет как на момент начала вычислений, так и на их продолжительность, то наиболее разумным будет за t_{start} считать время постановки первого задания в очередь на BOINC-сервере.

На основе параметра Makespan, можно дать определение ускорения системы (Speedup):

$$Sp = \frac{\sum t_i}{Ms},$$

t_i – время расчёта *i*-го подзадания.

Применимо к инфраструктуре ГСПК использование параметра Speedup также требует двух важных уточнений.

Первое уточнение связано с использованием репликации. В рамках данной работы, будем считать, что если канонический результат получен из n реплик, то ti – это среднее арифметическое время вычислений.

Второе уточнение связано с тем, какую величину считать временем вычисления отдельного подзадания – время непосредственных вычислений центрального процессора или время между отправкой задания на вычислительный узел и получением результата. В рамках

инфраструктуры ГСПК вычисления на узле могут приостанавливаться и перезапускаться. Поэтому для многих вычислительных узлов время использования центрального процессора и время выполнения подзадания могут сильно отличаться. Параметр ускорения, посчитанный на основе CPU time (далее – Speedupcpu или Spcpu), характеризует численное преимущество инфраструктуры BOINC перед идеально стабильной последовательной машиной с усреднёнными характеристиками вычислительной мощности подключённых вычислительных узлов. Этот параметр актуален при вычислении ускорения гибридной инфраструктуры или при сравнении ускорений, полученных на параллельной системе и на ГСПК.

Рассмотрим параметр ускорения, посчитанный на основе времени пребывания задания на вычислительном узле (далее – Speedup или Sp). Speedup характеризует преимущество системы BOINC перед последовательной вычислительной системой с производительностью, подобной усреднённой *фактической* производительности персонального компьютера добровольца (с учётом перезагрузок, сбросов и отключений). Посчитанная таким образом величина Sp является метрикой ускорения, которое даёт распараллеливание при использовании доступных в ГСПК реальных вычислительных ресурсов. Этот параметр актуален при сравнении различных политик планирования в рамках ГСПК.

Аналогично параметру Makespan, параметр Speedup является динамическим параметром, зависящим от состояния инфраструктуры для конкретного периода проведения вычислений.

Критерий оценки производительности по времени решения прикладной задачи является важным в рамках концепции высокопроизводительных вычислений (High Performance Computing – HPC). На практике, в рамках данной концепции специалист, поставивший прикладную задачу, желает получить результат вычислений в течении периода от нескольких часов до нескольких недель.

2.2 Способность системы выполнить очень большие объёмы вычислений

В рамках данного критерия объём проделанной работы играет более важную роль, чем быстрота получения результата прикладной задачи. Один из способов получения количественных метрик производительности проекта ГСПК состоит в подсчёте суммарного процессорного времени (или количества операций с плавающей точкой) всех выполненных заданий за некоторый достаточно продолжительный период. Эта характеристика может быть отнормирована на единицы времени. Например, общий объём вычислений можно разделить на количество секунд, дней или месяцев.

Данный критерий является важным в рамках концепции высокомасштабных вычислений (High-Throughput Computing – HTC). Согласно этой концепции, специалист, поставивший прикладную задачу, желает получить результат вычислений в течении периода от нескольких недель до нескольких лет.

Для сравнения концепций HPC и HTC можно провести аналогию с двумя транспортными задачами. Рассмотрим два портовых города (напр. Новороссийск и Буэнос-Айрес), расстояние между которыми 12 тыс. км. Первая задача состоит в перевозке одного пассажира между этими городами. Вторая задача состоит в перевозке 10 тысяч тонн груза из одного города в другой. Очевидно, что для транспортировки одного человека важным критерием будет время перевозки. И в соответствии с этим критерием, гражданская авиация будет более производительна, чем морские перевозки. Но для перевозки грузов большой массы наиболее важным критерием является принципиальная возможность такой транспортировки. В соответствии с данным критерием, использование грузовых судов является более производительным.

По каждому из двух критериев можно сравнивать производительность различных систем. Для отдельных вычислительных систем и прикладных задач данные два критерия могут как дополнять друг друга, так и противоречить друг другу.

Под эффективностью понимается способность системы оптимальным образом использовать имеющиеся ресурсы. В параллельной вычислительной системе эффективность определяется как средняя доля выполнения алгоритма, в течении которого процессоры выделены для решения задачи:

$$E_p = \frac{\sum T_i}{p \cdot M_s} = \frac{S_p}{p}.$$

Здесь T_i – время использования i-го процессора при решении задачи, p – количество процессоров. Для системы параллельных вычислений эффективность равна отношению реального ускорения к максимально возможному для данной системы в теории (т.е. к количеству процессоров).

Использование этой формулы для инфраструктуры BOINC требует ряда уточнений.

Первое уточнение связано с тем, какой параметр можно считать количеством процессоров. В рамках данной работы будем считать, что p – это суммарное количество процессоров компьютеров, которые посчитали минимум одно подзадание. Посчитанная таким образом эффективность основана на средней стабильности вычислительных узлов, не зависящей от политики управления ресурсами.

Второе уточнение связано со способом вычисления ускорения при оценке эффективности. Аналогично двум способам вычисления ускорения, можно составить два способа вычисления эффективности E_p и $E_{сри}$.

Также возможен специфический для ГСПК способ оценки эффективности работы инфраструктуры, основанный на анализе количества запросов. Вычисления могут быть организованы таким образом, что в некоторые промежутки времени сервер может не содержать заданий. Если вычислительный узел запрашивает задания у сервера задания, а сервер таковых заданий не имеет, то узел будет некоторое время бездействовать. Чем больше суммарное время бездействия вычислительных узлов, тем ниже эффективность работы инфраструктуры. Запросы, в ответ на которые отправляется вычислительное задание будем называть обслуженными, а запросы, переводящие вычислительный узел в состояние бездействия – необслуженными. Дополнительная метрика оценки эффективности работы инфраструктуры равна отношению числа обслуженных запросов со стороны вычислительных узлов к их общему числу. При равномерном распределении запросов во времени данный параметр эффективности будет стремиться к отношению времени, когда очередь содержит какое-либо количество заданий к общему времени рассмотрения:

$$E_{req} = \frac{N_{handle}}{N} \approx \frac{T_{queue}}{T},$$

N_{handle} – количество обслуженных запросов со стороны вычислительных узлов; N – общее количество запросов; T_{queue} – время, в течении которого очередь содержала одно или более заданий; T – общее время проведения вычислений.

При рассмотрении влияния различных системных эффектов на производительность и эффективность необходимо учитывать по какому именно критерию оцениваются указанные величины. Принципиально важными является ожидания специалистов в прикладной области, использующих ресурсы инфраструктуры ГСПК.

3. Анализ причин потери производительности и эффективности при реализации эволюционного алгоритма на ГСПК

Рассмотрим основные причины снижения производительности и эффективности работы ЭА на инфраструктуре ГСПК и введём метрики, характеризующие степень влияния конкретных причин. Данные метрики могут использоваться при сравнении различных политик

управления ресурсами для выработки оптимальной стратегии. Также система вычисления метрик может быть интегрирована в инфраструктуру в качестве системы поддержки принятия решений для оператора вычислительной системы в режиме реального времени.

3.1 Ожидание всех результатов поколения (проблема “семеро одного не ждут”)

Логика эволюционного алгоритма предполагает несколько поколений вычислений. Последовательность проведения вычислений ЭА USPEX с использованием решателя VASP отображена на рис. 1.

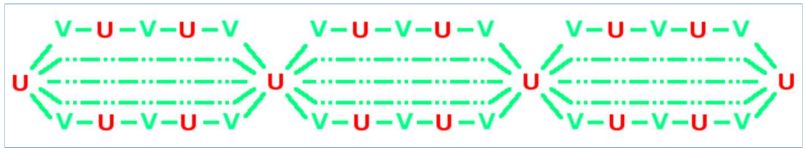

Рис. 1. Пример нитей трёх поколений вычислений. V – решатель VASP, выполняется на вычислительном узле. U – эволюционный алгоритм USPEX, выполняется на сервере
Fig. 1. An example of threads of three generations of computation. V - VASP solver, executed on the computational node. U - evolutionary USPEX algorithm, executed on the server

Рисунок показывает схему использования ресурсов в эволюционном алгоритме для 3-х поколений. Каждое поколение состоит из N независимых друг от друга нитей. Каждая нить представляет собой последовательность действий, выполняемых поочерёдно эволюционным алгоритмом на стороне сервера (*U – эволюционный алгоритм USPEX*) и решателем на стороне вычислительного узла (*V – VASP*). В пределах нити новые подзадания генерируются эволюционным алгоритмом на основе результатов предыдущих. Для типовых задач моделирования структуры вещества одна нить состоит из 3-х или 4-х участков работы VASP, генерируемых эволюционным алгоритмом USPEX. В пределах одной нити различные VASP-задания могут выполняться на различных вычислительных узлах. Число нитей в поколении является постоянным для решаемой научной задачи и может принимать значения от 1 до нескольких сотен. Число поколений также может достигать нескольких сотен. Новое поколение вычислений генерируется после того как все нити вычислений предыдущего поколения полностью посчитаны.

Рассмотрим типовой сценарий возникновения задержки одного из заданий поколения. На практике задержки возникают в 5-10% случаев. Допустим, в рамках решаемой задачи поколение состоит из восьми нитей вычислений, семь из которых были посчитаны инфраструктурой BOINC за первые сутки, а восьмая нить вычислений обрабатывалась 5 суток по причине нестабильности отдельного вычислительного узла. Таким образом, расчёт всего поколения займет пять суток. Последние 4 суток поколение будет ожидать оставшееся последнее задание.

Пример поколения, в котором сбой на отдельной нити вычислений задерживает генерацию нового поколения, показан на рис. 2:

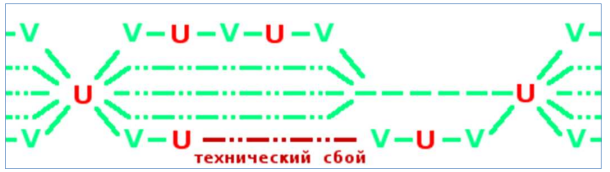

Рис. 2. Пример нитей последовательности вычислений в поколении после возникновения технического сбоя
Fig. 2. An example of threads of a sequence of calculations in a generation after a technical failure occurs

Эффект задержки последнего задания способен в несколько раз увеличить время решения научной задачи. Но этот эффект ряде случает не будет снижать параметр эффективности E_{req}. Например, если в инфраструктуре BOINC одновременно решается несколько научных задач (возможно, поставленных разными специалистами), то во время ожидания поколения одной научной задачи инфраструктура будет успешно выполнять решение других задач. Преодоление эффекта снижения времени выполнения вычислительного задания возможно посредством применения различных научных методов к планированию вычислениями и предоставления потребителю ресурсов возможности настройки параметров вычислений в соответствии с его приоритетами.

Для количественной оценки влияния ожидания всех результатов поколения при решении конкретной USPEX-задачи необходимо ввести ряд величин.

a) Суммарное время ожидания для всех нитей всех поколений. Временем ожидания для отдельной нити поколения является время между окончанием вычислений данной нити и окончанием расчётов всех нитей поколения. Для последней посчитанной нити в поколении время ожидания равно нулю.

$$T_w = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij},$$

n – количество поколений, m – количество нитей в поколении, t_{ij} – время ожидания j-й нити в i-м поколении.

b) Коэффициент ожидания последней нити в поколении, равный отношению суммарного полного времени всех поколений к суммарному времени ожидания. Под полным временем поколения понимается произведение числа нитей в поколении на время самого поколения.

$$K_w = \frac{T_w}{\sum_i^n m t_i},$$

T_w – суммарное время ожидания всех нитей, n – количество поколений, m – количество нитей в поколении, t_i – время выполнения i-го поколения.

3.2 Влияние очерёдности для подзаданий

Данный источник потерь производительности аналогичен проблеме «семеро одного не ждут». Как уже отмечалось выше, в системе BOINC каждое подзадание отправляется в общую очередь. Предположим, что одно подзадание требует час процессорного времени для расчётов, а среднее время ожидания в очереди – 10 часов. Таким образом, USPEX получит результат задания через 11 часов после отправки. Аналогичным образом последующие подзадания вычислительной задачи будут отправлены на вычислительный узел после прохождения очереди.

При отсутствии репликации наилучшими количественными характеристиками влияния очереди будут суммарное и среднее время пребывания задания в очереди, а также отношение суммарного времени пребывания в очереди к суммарному времени всех заданий.

Совокупное влияние единой очереди и эффекта ожидания последнего результата способно привести к снижению уровня производительности в десятки и сотни раз. Рассмотрим пример. Пусть в поколении 10 нитей, каждая из которых предполагает 1 час расчётов на стороне вычислительного узла. 10 заданий отправляются на 10 вычислительных узлов. Через час времени возвращается 9 результатов вычислений. 10-й результат не был получен по причине выхода вычислительного узла из строя. Тогда по истечении времени дедлайна (предположим, 10-ти часов) будет сгенерировано дублирующее задание и поставлено в общую очередь. По прошествии ещё 10-ти часов пребывания в очереди дублирующее задание отправится на вычислительный узел, где будет успешно посчитано в течении часа.

Таким образом, время расчёта всего поколения составит 21 час, 20 из которых составит ожидание последнего задания. Из этих 20-ти часов 10 часов составит дедлайн, и 10 часов составит ожидание дублирующего задания в очереди. На практике проведения вычислений

при большом количестве нитей в поколении наблюдались эффекты многократной задержки дублирующих заданий, сильно снижающих производительность.

4. Заключение

Представленная работа является результатом анализа практики применения эволюционного алгоритма USPEX для решения задач оптимизации. Следующий этап исследований состоит в применении методов имитационного моделирования процесса проведения вычислений для выработки оптимального алгоритма управления ресурсами. Полученную на моделируемой инфраструктуре стратегию предполагается применять при проведении практических вычислений. Предложенные в этой работе метрики могут быть использованы для количественной оценки эффективности и производительности системы на дальнейших этапах исследований.

Список литературы / References

[1]. Berkeley Open Infrastructure for Network Computing. Available at: https://boinc.berkeley.edu/. Accessed 19.08.2020.

[2]. Манзюк М.О., Заикин О.С., Посыпкин М.А. CluboRUN: программный комплекс для использования свободных ресурсов вычислительных кластеров в boinc-расчетах. Информационные технологии и вычислительные системы, № 4, 2014 г., стр. 3-11. / Manzyuk M.O., Zaikin O.S., Posypkin M.A. CluBORun: tool for utilizing idle resources of computing clusters in BOINC computing. Informacionnye tekhnologii I vichslitel'nye sistemy (Journal of Information Technologies and Computing Systems, № 4, 2014, pp. 3-11 (in Russian).

[3]. Ivashko E., Golovin A., Partition algorithm for association rules mining in BOINC-based enterprise desktop grid. Lecture Notes in Computer Science, vol. 9251, 2015, pp. 268-272.

[4]. Posypkin M., Khrapov N. Branch and bound method on desktop grid systems. In Proc. of the 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference, 2017, pp. 526-528.

[5]. Богачкова И.А., Заикин О.С., Кочемазов С.Е., Отпущенников И.В., Семёнов А.А. Применение алгоритмов решения проблемы булевой выполнимости к криптоанализу хэш-функций семейства MD. Прикладная дискретная математика. Приложение, № 8, 2015 г., стр. 139–142 / Bogachkova I.A., Zaikin O.S., Kochemazov S.E., Otpuschennikov I.V., Semenov A.A. Application of algorithms for solving the problem of Boolean satisfiability to cryptanalysis of hash functions of the MD family. Applied discrete mathematics. Application, № 8, 2015, pp. 139–142 (in Russian).

[6]. Vatutin E. Comparison of Decisions Quality of Heuristic Methods with Sequential Formation of the Decision in the Graph Shortest Path Problem. In Proc. of the Third International Conference BOINC-based High Performance Computing, 2017, pp. 67-76.

[7]. Khrapov N., Rozen V., Samtsevich A., Posypkin M., Sukhomlin V., Oganov A. Using virtualization to protect the proprietary material science applications in volunteer computing. Open Engineering, vol. 8, issue 1, 2018, pp. 57-60.

[8]. Храпов Н. П. Анализ причин потери производительности при адаптации эволюционного алгоритма к системам добровольных вычислений. Сборник научных трудов международного конгресса «Современные проблемы компьютерных и информационных наук», 2019 г., стр. 21-26. / Khrapov N.P. Analysis of the performance reasons for adapting the evolutionary algorithm to voluntary computing systems. In Proc of the International Congress on Modern Problems of Computer and Information Sciences], 2019, pp. 21-26 (in Russian).

Информация об авторе / Information about the author

Николай Павлович ХРАПОВ – и.о. научного сотрудника. Научные интересы: распределённые вычисления, виртуализация, грид-системы из персональных компьютеров.

Nikolay Pavlovich KHRAPOV – researcher. Research interests: distributed computing, virtualization, desktopgrid.