CluBORun: программный комплекс для использования свободных ресурсов вычислительных кластеров в BOINC-расчетах[1]

М.О. Манзюк, О.С. Заикин, М.А. Посыпкин

Аннотация. В работе предлагается подход к использованию свободных ресурсов вычислительных кластеров в проектах добровольных вычислений на платформе BOINC. При этом необходимы только обычные права пользователя кластера, что отличает подход от аналогов. Подход был реализован в виде программного комплекса и успешно протестирован на двух вычислительных кластерах.

Ключевые слова: добровольные распределенные вычисления, BOINC, вычислительный кластер, MPI, задача выполнимости булевых формул.

Введение

За последние 15 лет добровольные распределенные вычисления стали эффективным и мощным вычислительным ресурсом при проведении научных исследований. Основная идея добровольных вычислений состоит в том, что частные лица (добровольцы) подключают свободные ресурсы своих персональных компьютеров (ПК) к проектам добровольных вычислений через интернет. Каждый проект предназначен для решения одной или нескольких масштабных задач (как правило, научных), которые можно разбить на независимые подзадачи. Процедура добавления ПК в состав того или иного проекта крайне проста. Доброволец устанавливает на свой ПК программу-менеджер, позволяющую быстро выбрать и подключить интересующий его проект, а также регулировать распределение ресурсов между несколькими подключенными проектами. По умолчанию такие менеджеры настроены на использование только свободных ресурсов ПК, чтобы не мешать работе пользователей. За решение каждого задания участнику проекта начисляются кредиты пропорционально затраченным ресурсам. Кредиты не несут никакой материальной составляющей, но стимулируют соревнование между участниками за места в рейтинге проекта. Участники объединяются в команды (например, по национальному признаку), и основная конкуренция происходит именно на командном уровне.

Главное достоинство добровольных распределенных вычислений состоит в низкой стоимости расчетов, т.к. необходимо поддерживать работоспособность только сервера и сайта проекта, а все вычислительные ресурсы предоставляют добровольцы. При этом необходимо учитывать некоторые особенности. Поскольку в проектах добровольных вычислений используются ПК частных лиц, всегда существует веро-

[1]Работа выполнена при частичной финансовой поддержке РФФИ (гранты № 14-07-00403, № 13-07-00768), Совета по грантам Президента РФ для поддержки молодых ученых (стипендия СП-1855.2012.5) и ПФИ РАН №14 "Проблемы создания национальной научной распределенной информационно-вычислительной среды на основе GRID технологий, облачных вычислений и современных телекоммуникационных сетей"

ятность того, что результаты вычислений будут некорректными из-за аппаратного или программного сбоя, либо будут сфальсифицированы пользователем. Для учета этой особенности используются избыточные вычисления – рассылается не один, а нескольких (обычно два) экземпляра каждого задания на ПК разных участников. При совпадении результатов каждый из экземпляров помечается как верный, а при несовпадении создаются и высылаются дополнительные экземпляры задания.

Значительный прогресс добровольных распределенных вычислений достигнут во многом благодаря развитию открытой платформы BOINC [1]. Данная платформа позволяет быстро создавать проекты на основе стандартного шаблона. Существует несколько подобных платформ, но BOINC является самой широко используемой. Если в 1999 году активно действующих проектов добровольных вычислений было всего 3, то сейчас их около 80 и почти все они основаны на BOINC. Сформировалось интернациональное сообщество участников добровольных вычислений. Причины возникновения этого сообщества следующие:

- наличие проектов с интересными и даже претенциозными целями (например, поиск внеземных цивилизаций в проекте SETI@home), а также проектов с более прикладными и понятными целями, такими как поиск новых лекарств (Rosetta@home, POEM@home);
- ощущение причастности участников к научным исследованиям;
- хорошо развитая система соревнований между участниками и командами.

При наличии хорошо налаженной обратной связи с представителями сообщества добровольцев новый проект может за 2-3 месяца получить в свое распоряжение значительные вычислительные ресурсы.

1. Использование вычислительных кластеров в проектах добровольных распределенных вычислений

Некоторые проекты добровольных вычислений обладают производительностью, сравнимой с самыми мощными суперкомпьютерами (более 1 PFLOPs). Несмотря на это, использование в BOINC-расчетах и ресурсов ПК добровольцев, и ресурсов вычислительных кластеров может быть целесообразно по следующим причинам.

1. Нередко возникают ситуации, в которых часть вычислительных ресурсов кластеров простаивает. Запуск BOINC-расчетов позволяет использовать эти ресурсы для научных вычислений. Если владелец кластера (научный институт или университет) задействован в организации некоторого BOINC-проекта, то выполнение расчетов на кластере для этого проекта тем более оправдано.
2. Это важно для репутации эксперимента – если не задействуются имеющиеся у научного сообщества вычислительные ресурсы, то почему они запрашиваются у сообщества добровольцев?
3. Для проектов в их начальной стадии развития или для проектов, тематикой которых не заинтересованы добровольные участники, прирост производительности от кластеров может быть весьма значительным.
4. Вычислительный кластер – весьма надежное устройство. Результаты расчетов на кластере можно принять как эталонные при проверке результатов копий заданий.
5. Одна единица задействованных кластерных мощностей может привлечь еще несколько единиц благодаря конкуренции между командами участников за лидирующие позиции в рейтингах статистики.

Платформа BOINC не оперирует с вычислительным кластером как с одним компьютером. Для BOINC кластер – это набор независимых узлов без возможности взаимодействия этих узлов друг с другом. Тем не менее, в рамках конкретного узла кластера BOINC позволяет запускать не только последовательные и многопоточные приложения, но и параллельные приложения стандарта MPI.

Впервые кластер был подключен к проекту Einstein@home [2] в декабре 2007 года. Благодаря подключению кластера за 2 месяца производительность проекта выросла с 64 до 89 TFLOPs. С тех пор в проекте Einstein@home постоянно используются различные кластеры.

Существует несколько подходов к использованию платформы BOINC для проведения вы-

числений на кластерах. В соответствии с первым подходом узлы кластера исключаются из системы управления очередями заданий и подключаются к проекту добровольных вычислений, созданному на основе BOINC. При этом каждый узел кластера будет рассматриваться как отдельный ПК в проекте. В рамках второго подхода, как и в первом подходе, узлы кластера подключаются к проектам добровольных вычислений, но при этом узлы не исключаются из системы управления очередями заданий. Как и в случае ПК частных лиц, BOINC-вычисления при этом не будут мешать выполнению на узлах задач, запущенных другими пользователями. Третий подход основан на том, что платформу BOINC можно использовать не только для создания проектов добровольных вычислений, но и для создания грид-систем. В этом случае в грид могут быть добавлены вычислительные узлы кластеров. Примером реализации такого подхода является созданный в ИПМИ КарНЦ РАН грид [3], использованный для интеллектуального анализа данных. В состав этого грида входят серверы, ПК сотрудников, а также узлы вычислительного кластера ЦКП КарНЦ РАН.

Для реализации описанных выше подходов необходимы права администратора на использование кластеров. В настоящей работе предлагается другой подход, в рамках которого достаточно иметь доступ к кластеру с правами обычного пользователя. Основная идея подхода состоит в том, что BOINC-расчеты запускаются на кластере в виде MPI-заданий, которые обрабатываются точно так же, как и другие задания в системе управления очередями кластера. При этом BOINC используется на кластере для связи с сервером проекта, получения и обработки данных, а также для отправки результатов на сервер проекта. Ключевой особенностью предлагаемого подхода является использование только свободных ресурсов кластеров. Здесь напрашивается аналогия с BOINC-менеджером, который управляет процессом вычислений на ПК добровольцев, только вместо свободных ресурсов ПК предлагается использовать свободные ресурсы кластера. Актуальность реализации данного подхода обосновывается тем, что на текущий момент ни в самой платформе BOINC, ни среди официальных дополнений к платформе BOINC [4] нет программных средств, предназначенных для использования в BOINC-проектах кластеров через взаимодействие с очередью заданий кластера. Описание реализации предлагаемого подхода представлено в следующем разделе.

Следует отметить, что предлагаемый подход базируется на идеях, используемых в программном комплексе BNB-Grid [5]. BNB-Grid предназначен для создания распределенной системы на основе вычислительных кластеров. С помощью BNB-Grid были успешно решены некоторые задачи глобальной оптимизации [5], а также задача логического криптоанализа генератора ключевого потока A5/1 [6]. В BNB-Grid создаются MPI-задания, которые запускаются через систему управления очередями заданий кластеров (т.е. используются только обычные права пользователя кластера). Особенностью предлагаемого в настоящей статье подхода является ориентация на использование только свободных ресурсов кластеров, а также направленность на взаимодействие с BOINC-проектами.

2. Описание программного комплекса CluBORun

Тезисно описанный в предыдущем разделе подход был реализован в виде программного комплекса Cluster for BOINC Run (CluBORun), третья (последняя на момент публикации) версия которого состоит из следующих основных компонентов.

1. Каталоги с экземплярами BOINC-клиента.
2. Файлы-флаги, соответствующие MPI-задачам. MPI-задачи при наличии свободных ресурсов запускаются на кластере. Каждая MPI-задача представляет собой несколько экземпляров BOINC-клиентов. При запуске задачи создается start-флаг, при необходимости его остановки – stop-флаг.
3. Текстовый файл all_tasks.txt – перечень запускаемых задач. Для каждой задачи указывается имя start-флага, имя stop-флага, путь к каталогу с экземплярами BOINC-клиентов и перечень экземпляров, которые надо запустить. Пример содержимого файла all_tasks.txt с описанием двух задач представлен ниже.

task001 task001.start task001.stop /home/user/SAT node001 node002 ... node008
task002 task002.start task002.stop /home/user/SAT node009 node010 ... node016

4. Исполняемый файл start_boinc, на вход которому в качестве параметров подается время работы, а также пути к соответствующим экземплярам BOINC-клиентов и файлам-флагам. Файл start_boinc запускает на каждом выделенном для задачи узле один BOINC-клиент. Работа BOINC-клиента прерывается при исчерпании времени работы, либо экстренно – при появлении stop-флага.

5. Скрипт catch_node.sh, предназначенный для анализа загрузки кластера и запуска модуля start_boinc. Данный скрипт запрашивает список всех задач, запущенных на кластере, а также список задач, стоящих в очереди (если таковые есть). Анализируя данный список, скрипт определяет, сколько BOINC-задач уже запущено на кластере и сколько их должно быть, исходя из числа свободных узлов и состава очереди кластера.

Один из основных принципов работы CluBORun состоит в том, что управляемые им BOINC-вычисления не должны препятствовать запуску других задач в очереди кластера. Если запущенные BOINC-задачи требуется остановить (в очереди появились задачи, которые могли бы начать выполняться на высвобождаемых узлах), то скрипт catch_node.sh остановит необходимое количество BOINC-задач путем формирования stop-флагов.

При обнаружении свободных узлов скрипт catch_node.sh считывает список start-флагов. При наличии незапущенных задач они запускаются через систему управления заданиями с помощью команды mpirun. При вызове команды mpirun скрипт catch_node.sh передает ей следующие параметры:

- число ядер, занимаемых у вычислительного кластера (оно равно числу ядер CPU на узлах, которые предполагается занять задачей);
- время, на которое производится захват узла;
- путь к MPI-приложению, которое команда mpirun должна запустить на выбранном узле;
- параметры, которые должны быть переданы самому MPI-приложению.

Изначально платформа BOINC не предназначена для проведения расчетов на вычислительных кластерах. Кроме этого, при запуске приложений на кластере пользователям разрешены только те действия, которые не противоречат политике безопасности данного кластера.

Для задействования BOINC на кластерах мы решили использовать специальное MPI-приложение. Основное требование к этому приложению – возможность запустить на каждом используемом узле кластера один BOINC-клиент и корректно передать ему все ресурсы этого узла. При этом MPI-процессы не должны выполнять никаких вычислений, а нужны лишь для корректного запуска BOINC-расчетов на узлах (т.е. все вычисления выполняет BOINC-клиент). В нашем случае в роли MPI-приложения выступает описанное выше приложение start_boinc, которое запускается с помощью команды mpirun на n процессорных ядрах на узлах кластера (каждому ядру соответствует один процесс приложения start_boinc). Запущенные процессы start_boinc сначала распределяют между собой роли, а затем – следуют им. Всего есть три роли:

- координатор – процесс, распределяющий остальные роли между узлами, задающий соответствие между подчиненными процессами и запускаемыми экземплярами BOINC-клиентов, выдающий сигнал на остановку запущенных экземпляров BOINC-клиентов, сигнализирующий об окончании работы MPI-приложения в целом;
- владелец BOINC-клиента – процесс, запускающий на своем узле указанный ему координатором экземпляр BOINC-клиента;
- спящий процесс, который присутствует в системе до получения указания о том, что ему пора завершаться.

Таким образом, в рамках MPI-задачи, запускаемой на k узлах, среди всех параллельных процессов выделяется 1 координатор и k владельцев BOINC-клиентов. Остальные процессы являются спящими и ожидают от координатора команды на остановку.

В начале своей работы координатор создает start-флаг, сигнализирующий о запуске задачи, а после распределения ролей между остальными процессами входит в цикл ожидания, в котором каждые 30 секунд проверяет, не пора ли завершать работу из-за того, что отведенное

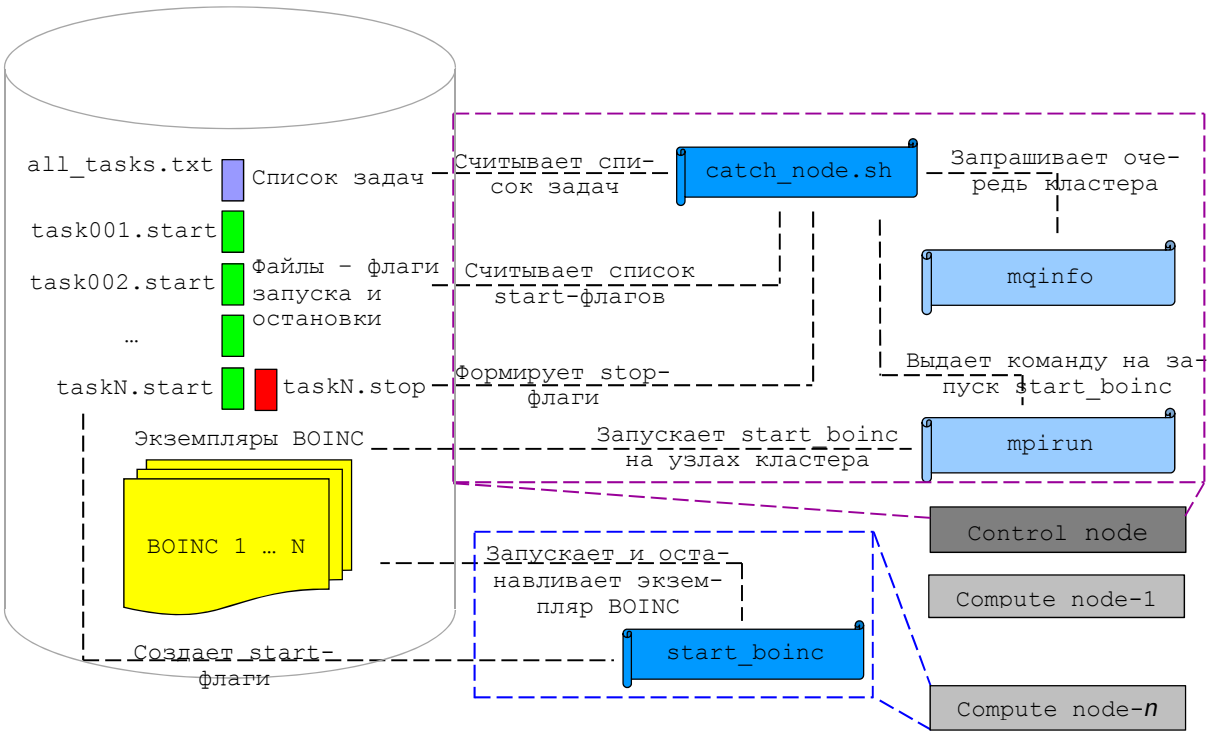

Рис. 1. Схема работы программного комплекса CluBORun

для задачи время подошло к концу или из-за появления stop-флага.

При необходимости завершить работу процесс-координатор отправляет остальным процессам соответствующее сообщение и завершает свою работу. Подчиненный спящий процесс, получив сообщение о завершении работы, завершает свою работу. Процесс, являющийся владельцем BOINC-клиента, выдает команду синхронизации с сервером проекта, ждет 60 секунд, после чего штатно завершает свою работу, а также работу BOINC-клиента.

Реализация комплекса CluBORun в виде простой MPI-программы и набора sh-скриптов позволяет легко менять логику его работы или вводить в нее дополнительные параметры. В текущей версии, например, число узлов, запрашиваемых у системы, не является фиксированным и зависит от того, сколько экземпляров BOINC-клиентов перечислено в строке с описанием задачи в файле all_tasks.txt.

На Рис. 1 представлена схема работы CluBORun. Присутствуют следующие компоненты:

- mqinfo – команда просмотра очереди заданий кластера;
- mpirun – команда постановки MPI-задачи в очередь заданий кластера;
- catch_node.sh, start_boinc, экземпляры BOINC-клиентов, файлы-флаги, all_nodes.txt – компоненты CluBORun;
- Control node – управляющий узел кластера, в памяти которого выполняются mqinfo, mpirun и catch_node.sh;
- Compute node-1..*n* – вычислительные узлы кластера, в памяти которых mpirun с помощью start_boinc запускает экземпляры BOINC-клиента.

3. Вычислительные эксперименты

Первая версия CluBORun [7] была создана для работы с системами управления очередями заданий кластеров Cleo [8] (НИВЦ МГУ) и СУПЗ [9] (МСЦ РАН). Эта версия CluBORun была запущена на кластере Blackford (Институт динамики систем и теории управления СО РАН, ИДСТУ СО РАН) [10] с 144 ядрами. Результаты работы CluBORun на кластере Blackford с 12 апреля по 13 мая 2013 года представлены на Рис. 2. Расчеты проводились для проекта добровольных вычислений SAT@home

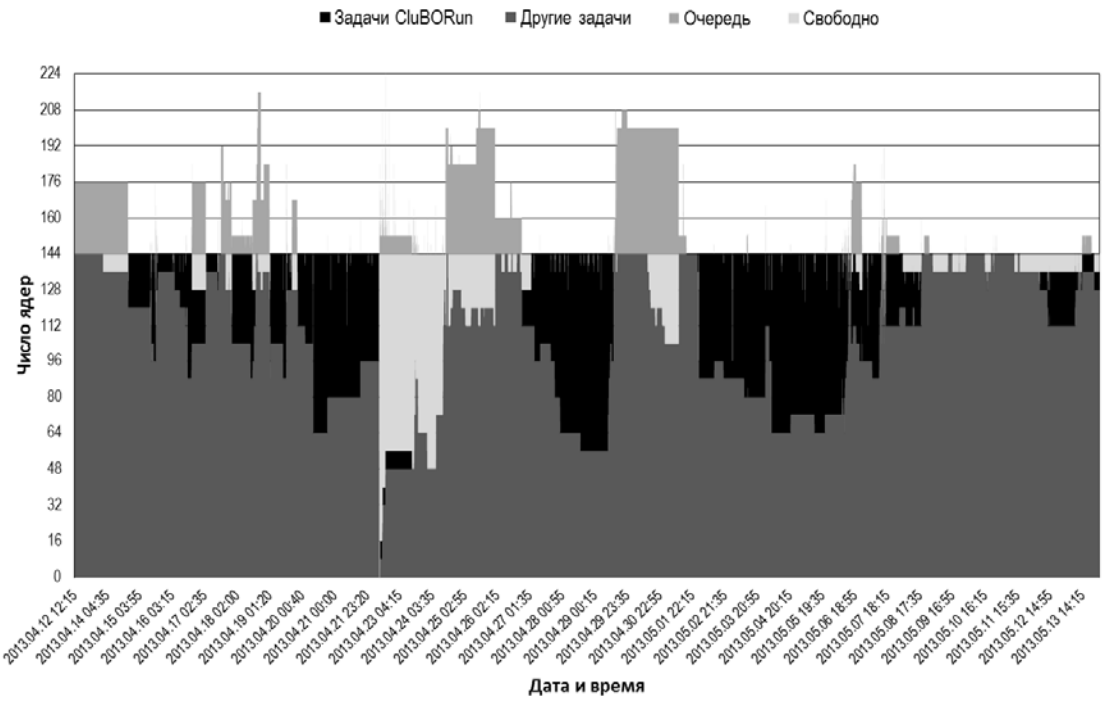

Рис. 2. Загрузка кластера Blackford ИДСТУ СО РАН с 12.04.2013 по 13.05.2013

[11, 12], в разработке которого принимают участие авторы данной статьи. При появлении в очереди задач сторонних пользователей скрипт catch_node.sh останавливал запущенные BOINC-задачи до того момента, пока не запускались все задачи из очереди. Если какая-либо ожидающая задача запрашивала ресурсов больше, чем было занято BOINC-задачами, то все BOINC-задачи все равно останавливались.

С 22 апреля по 27 апреля 2013 года на работу CluBORun накладывались жесткие ограничения по числу свободных узлов, из-за этого более половины ресурсов кластера простаивало. В следующий промежуток времени эти ограничения либо смягчались, либо снимались вовсе, т.е. все незаблокированные узлы кластера были задействованы либо в обычных расчетах, либо в BOINC-расчетах. Также на графике можно увидеть, как CluBORun освобождает от BOINC-расчетов узлы при появлении в очереди заданий других пользователей.

В декабре 2013 года CluBORun был запущен на кластере МВС-100k [13], на котором установлена система СУПЗ. Специфика работы с МВС-100k привела сначала к появлению ***второй***, а потом и третьей версии CluBORun.

Во второй версии CluBORun была использована более гибкая схема работы с очередью задач кластера. В частности, в сценарии запуска BOINC-задач важную роль стала играть информация о первой задаче в очереди. Так, если первой задаче в очереди требовалось ресурсов больше, чем было свободно на данный момент, принималось решение запуска BOINC-расчетов на этих ресурсах. При этом осуществлялся периодический мониторинг, и как только совокупное число свободных ресурсов и ресурсов, занятых BOINC-расчетами, оказывалось достаточным для запуска первой задачи в очереди, все BOINC-расчеты останавливались. При очевидных преимуществах данной схемы из-за ограничений на число запускаемых задач (10 на одного пользователя) в ряде случаев она позволяла задействовать лишь небольшую часть свободных вычислительных ресурсов. Речь идет о временных промежутках, в которые на кластере были свободны десятки и даже сотни узлов (например, в декабре 2013 года на МВС-100k было свободно в среднем около 98 узлов).

В третьей версии CluBORun алгоритм анализа очереди кластера был оставлен от второй версии, а механизм запуска BOINC экземпляров на его

узлах был кардинально изменен. Ранее (в первой версии) MPI-задача представляла собой sh-скрипт, который запускался на одном узле кластера в нескольких (равном числу ядер) экземплярах, один из которых (назначенный ведущим) запускал экземпляр BOINC на этом узле. В третьей версии комплексу CluBORun потребовалось оперировать несколькими узлами в рамках одной задачи. Для этого на языке C++ было создано MPI-приложение start_boinc (описание в предыдущем разделе). После своего запуска на узлах, процессы start_boinc определяли свои роли и, выполняя их, обеспечивали запуск экземпляров BOINC на каждом из узлов, выделенных задаче. Далее *стоимостью* BOINC-задачи будет называться число ядер, которое для нее запрашивается в очереди заданий.

Создание MPI-приложения, позволившего оперировать несколькими узлами кластера в рамках одной задачи, дало возможность задействовать те ресурсы кластера, которые время от времени простаивали. При этом мы столкнулись со следующими проблемами.

1. Ограничение на число одновременно запущенных задач оставалось прежним (10 задач на одного пользователя). А с учетом того, что у всех BOINC-задач по умолчанию одинаковая стоимость, для задействования большого числа свободных ресурсов (например, несколько сотен ядер) требовалось вручную устанавливать большую стоимость BOINC-задач. Но это не позволяло задействовать свободные ядра кластера, если их число было меньше стоимости BOINC-задачи (например, если стоимость BOINC-задачи 320, а свободно 280 ядер).

2. В некоторых BOINC-каталогах могли оставаться незавершенные задания, полученные с сервера проекта добровольных вычислений. Если к некоторым BOINC-каталогам не обращаться длительное время, часть произведенных вычислений пропадет.

Данные проблемы были решены созданием асимметричного списка задач. В этом списке BOINC-задачи отсортированы по возрастанию числа запрашиваемых ядер. Например, в случае МВС-100к, в начале списка располагались задачи на 64 ядра (8 узлов по 8 ядер), затем – на 128 ядер и так далее. При этом в скрипте catch_node.sh стоимость каждой задачи назначалась равной стоимости первой задачи из списка (в случае МВС-100к это 64 ядра). В результате, если были свободны 64 ядра, а все задачи на 64 ядра из списка уже были запущено, скрипт catch_node.sh запускал задачи на 128 или на 256 ядер. Использование асимметричного списка привело к постоянной «перетасовке» BOINC-задач, благодаря чему каждый экземпляр BOINC-клиента получал достаточно времени для обработки полученных с сервера проекта заданий до истечения deadline.

Версия CluBORun, развернутая на МВС-100k в настоящее время, позволяет использовать значительную часть ядер, не занятых обычными расчетными задачами. Так, с 5 по 12 марта 2014 года в среднем было свободно 1824 ядра кластера, из которых примерно 512 ядер были задействованы для выполнения вычислительных задач SAT@home с помощью CluBORun. Результаты работы CluBORun на кластере МВС-100к с 5 по 12 марта 2014 года представлены на Рис. 3.

Благодаря тому, что задачи CluBORun в среднем меньше по своей длительности, чем обычные задачи на кластере, в ситуациях, описанных в пункте 3, получилось хотя бы частично задействовать свободные ресурсы кластера. В обычном режиме работы кластера получается задействовать большую часть свободных ресурсов.

Программный комплекс CluBORun был использован для подключения кластеров к решению задач в проекте добровольных вычислений SAT@home. С помощью CluBORun были найдены 17 новых пар ортогональных диагональных латинских квадратов порядка 10 [14] (до этого были известны 3 такие пары). Также в SAT@home были решены 3 ослабленные задачи криптоанализа генератора ключевого потока Bivium. Ослабление заключалось в подстановке известных значений последних 10 из 177 бит искомого заполнения регистров генератора Bivium. Отметим, что в [15] рассматривалось решение на кластере ослабленных задач криптоанализа Bivium, но при 12 известных битах из 177. В некоторые периоды времени вклад задействованного с помощью CluBORun кластера МВС-100к в производительность SAT@home достигал 40 %.

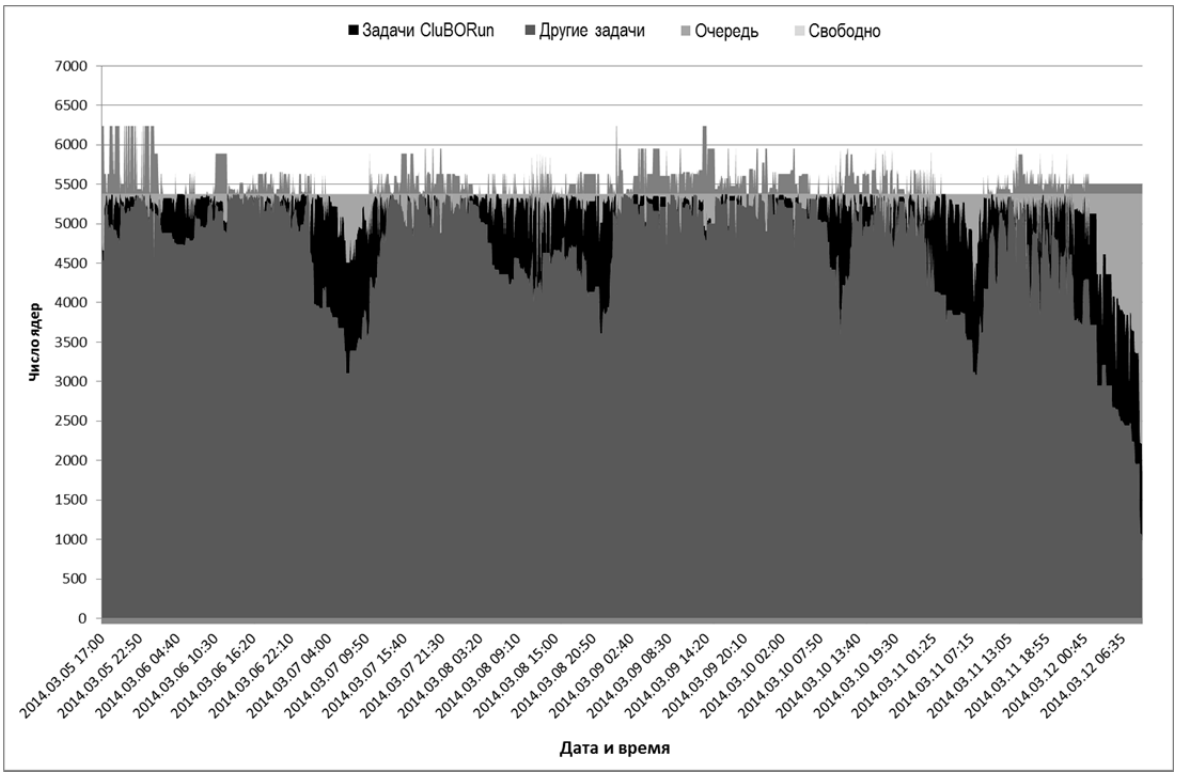

Рис. 3. Загрузка кластера МВС-100k с 01.04.2014 по 20.04.2014

4. Сравнение CluBORun с другими разработками

Существуют два принципиально различных подхода к интеграции распределенных ресурсов. Первый состоит в том, что ресурсы интегрируются на прикладном (верхнем) уровне. Такому подходу следует множество различных средств организации распределенных сценариев, например MathCloud [16] и WS-PGRADE [17].

Альтернативный подход состоит в интеграции ресурсов на системном уровне. Программный инструментарий 3G Bridge [18] предназначен для обмена заданиями между разными грид-системами. Данный инструментарий может быть использован для отправки заданий из одной грид-системы в другую. Основное отличие 3G Bridge от CluBORun состоит в том, что 3G Bridge ориентирован на использование уже готовых сервисных грид-систем, а CluBORun может быть использован при доступе к кластерам с правами обычного пользователя. Кроме этого, в CluBORun реализован принцип «использовать только свободные ресурсы кластера», которого нет в 3G Bridge.

Заключение

В ходе эксплуатации CluBORun успешно справился с задачей использования свободных ресурсов кластеров для BOINC-расчетов, а также позволил существенно увеличить производительность проекта добровольных вычислений SAT@home. Мы планируем добавить возможность работы с другими системами управления очередями кластеров (SLURM, PBS). Исходный код CluBORun и описание алгоритма его работы выложены в открытый доступ [19]. В ближайшее время разработчикам BOINC будет предложено добавить CluBORun в официальный список дополнений BOINC [4].

Литература

1. Anderson, D.P. BOINC: A System for Public-Resource Computing and Storage // In: Buyya, R. (ed.) GRID. pp. 4-10. IEEE Computer Society, 2004.
2. B. Knispel, R. P. Eatough, H. Kim, E. F. Keane, B. Allen, D. Anderson et al. Einstein@Home Discovery of 24 Pul-

sars in the Parkes Multi-beam Pulsar Survey, arXiv:1302.0467 [astro-ph.GA]
3. Ивашко Е.Е., Никитина Н.Н. Использование BOINC-грид в вычислительноемких научных исследованиях // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11, вып. 1. С. 53–57.
4. BOINC add-on software. URL: http://boinc.berkeley.edu/addons.php (дата обращения: 29.05.2014).
5. Посыпкин М.А. Решение задач глобальной оптимизации в среде распределенных вычислений // Программные продукты и системы. № 1. 2010. С. 23-29.
6. Alexander Semenov, Oleg Zaikin, Dmitry Bespalov, Mikhail Posypkin. Parallel Logical Cryptanalysis of the Generator A5/1 in BNB-Grid System // Lecture Notes in Computer Science. Vol. 6873, 2011, pp 473-483
7. Манзюк М.О., Заикин О.С. CluBORun: средство использования свободных ресурсов вычислительных кластеров для BOINC-расчетов // Труды научной конференции «Высокопроизводительные вычисления на базе BOINC: фундаментальные исследования и разработки». ИПМИ КарНЦ РАН. 2013. С. 9-14.
8. Система управления заданиями Cleo. URL: http://parcon.parallel.ru/cleo.html (дата обращения: 29.05.2014).
9. Система управления прохождением задач (СУПЗ). URL: http://suppz.jscc.ru/ (дата обращения: 29.05.2014).
10. Суперкомпьютерный центр Иркутского научного центра СО РАН. URL: http://hpc.icc.ru (дата обращения: 29.05.2014).
11. SAT@home: проект добровольных вычислений для решения трудных SAT-задач. URL: http://sat.isa.ru/pdsat/ (дата обращения: 29.05.2014).
12. Заикин О.С., Посыпкин М.А., Семенов А.А., Храпов Н.П. Опыт организации добровольных вычислений на примере проектов OPTIMA@home и SAT@home // Вестник ННГУ. № 5(2). 2012. С. 338-346.
13. Суперкомпьютерный центр РАН. URL: http://www.jscc.ru/scomputers.shtml (дата обращения: 29.05.2014).
14. Заикин О. С., Кочемазов С.Е., Семенов А.А. Поиск систем ортогональных латинских квадратов в проекте добровольных вычислений SAT@home // Труды первой российской конференции «Высокопроизводительные вычисления на базе BOINC: фундаментальные исследования и разработки». ИПМИ КарНЦ РАН. 2013. С. 3-8.
15. Заикин О.С., Семенов А.А. Применение метода Монте-Карло к прогнозированию времени параллельного решения проблемы булевой выполнимости // Вычислительные методы и программирование: Новые вычислительные технологии. 2014. Вып. 1. С. 22-35.
16. Afanasiev A., Sukhoroslov O., Voloshinov V. MathCloud: Publication and Reuse of Scientific Applications as RESTful Web Services // Parallel Computing Technologies. – Springer Berlin Heidelberg, 2013. – С. 394-408.
17. A. Balasko, Farkas, Z., and Kacsuk, P. Building Science Gateway by Utilizing the Generic WS-PGRADE/gUSE Workflow System, Computer Science, vol. 14, no. 2, p. 307, 2013.
18. Zoltan Farkas, Peter Kacsuk, Zoltan Balaton, Gabor Gombas. Interoperability of BOINC and EGEE // Future Generation Computer Systems. Vol. 26, no. 8, pp. 1092-1103, 2010.
19. Программный комплекс CluBORun. URL: https://github.com/Nauchnik/SAT-at-home/tree/master/ CluBORun (дата обращения: 29.05.2014).

Заикин Олег Сергеевич. Научный сотрудник Института динамики систем и теории управления СО РАН. Окончил Институт математики, экономики и информатики Иркутского государственного университета в 2005 году. Кандидат технических наук. Автор 23 печатных работ. Область научных интересов: параллельные вычисления, распределенные вычисления, искусственный интеллект, криптография. E-mail: zaikin.icc@gmail.com

Манзюк Максим Олегович. Бакалавр Волгоградского государственного технического университета. Автор одной печатной работы. Область научных интересов: распределенные вычисления, системное программирование, базы данных. E-mail: hoarfrost@rambler.ru

Посыпкин Михаил Анатольевич. Ведущий научный сотрудник Института проблем передачи информации РАН. Окончил МГУ имени М.В. Ломоносова в 1996 году. Кандидат физико-математических наук, доцент. Автор 120 печатных работ. Область научных интересов: распределенные вычисления, параллельные вычисления, компиляторы, формальные методы, методы оптимизации. E-mail: mposypkin@gmail.com