Высокопроизводительный поиск данных в системах с гибридной архитектурой. Программная платформа VAR.

High-performance data search in systems with heterogeneous architectures. Software platform VAR.

Васильев Николай Петрович
Ровнягин Михаил Михайлович

Аннотация: В статье рассматриваются некоторые вопросы, связанные с организацией хранения и поиска информации в многомашинных системах. Описываются современные принципы построения NoSQL-решений. Выполняется анализ возможностей ускорения работы подобных систем через призму GPGPU-технологий. Приводятся результаты сравнительного тестирования высокопроизводительной системы поиска и защищенного хранения данных VAR и NoSQL-системы Apache Cassandra.

Abstract: This article discusses some of the issues related to the organization of storage and searching in multicomputer systems. The modern principles of NoSQL-systems are described. Examines opportunities to accelerate the work of such systems through the prism of GPGPU-technologies. We present the results of comparative testing of high-performance search and secure storage of data and NoSQL-VAR system Apache Cassandra.

Ключевые слова: NoSQL, шифрование данных, NVIDIA CUDA, Java, программная платформа, GPGPU.

Keywords: NoSQL, data encryption, NVIDIA CUDA, Java, software platform, GPGPU

Введение

В настоящее время процесс генерации новых данных приобрел «лавинный» характер. В результате всеобщей информатизации и активного совершенствования современных вычислительных мощностей темпы роста объема хранимых данных можно охарактеризовать как крайне высокие. Помимо, собственно, хранения данные нуждаются в обработке. При этом анализ данных в общем случае не является разовым. В следствии развития методов Data Mining [1] и концепции BigData, старые данные обычно сохраняются в первозданном виде для последующего анализа с учетом новых тенденций и подходов, которые отсутствовали на момент первоначального анализа.

Классические РСУБД малопригодны в качестве систем хранения для задач Data Mining. Большое распространение в настоящее время получила концепция NoSQL (not only SQL) систем [2] хранения данных. Подобные системы, хоть и не обладают всей полнотой функциональности реляционных СУБД (РСУБД), но позволяют существенно увеличить пропускную способность приложений при работе с большими объемами слабоструктурированных данных. В качестве примера таких данных можно привести: экспериментальные данные, статистика обращений к веб-сервисам, метеоданные, поток информации из социальных сетей, микроблогов и др. В каждом из представленных случаев возможно осуществить параллельную обработку данных на распределенной вычислительной системе. При этом, необходимы системы хранения, способные обрабатывать миллионы довольно простых по форме запросов.

Приведенные выше рассуждения подтолкнули разработчиков к созданию высокопроизводительных систем хранения данных типа NoSQL. Начиная со второй половины 2000-х появилось множество подобных систем с самыми разными возможностями. Некоторые из них реализуют программный интерфейс, позволяющий транслировать некоторые инструкции классических РСУБД во внутреннее представление, некоторые частично реализуют требования ACID. Общей же чертой подобных систем является высокий показатель пропускной способности масштабируемый (в большинстве случаев линейно) в зависимости от количества используемых серверов хранения.

Множество компаний по всему миру используют в настоящее время распределенные высокопроизводительные NoSQL-решения. Сервисы таких корпораций как IBM, Amazon, Facebook, Twitter, Google, Oracle напрямую зависят от того, насколько эффективно функционируют системы хранения и поиска данных. Среди важнейших критериев эффективности применяемых NoSQL-решений можно выделить следующие показатели: пропускная способности, масштабируемость, энергоэффективность, затраты на обслуживание. Как можно заметить, данные показатели перекликаются с требованиями, предъявляемыми к современным суперкомпьютерам.

Действительно, задача поиска данных также является вычислительно задачей, с той лишь разницей, что работа с памятью поставлена на первый план. Однако, в рамках NoSQL-подхода предполагается использование стандартных кластерных решений, в отличии от РСУБД, работающих на специализированных серверах хранения. Таким образом, имеет смысл рассматривать высокопроизводительные распределенные системы хранения и поиска данных через призму суперкомпьютерных технологий.

В настоящее время ведущие позиции в суперкомпьютерных рейтингах занимают гибридные решения [3]. Под гибридностью понимается наличие в составе суперЭВМ вычислителей с различной архитектурой. В качестве сопроцессоров для классических CPU могут применяться GPU-ускорители или микросхемы ПЛИС. Данные устройства позволяют существенно ускорить некоторые операции и повысить энергоэффективность. С точки зрения задач хранения и поиска данных, сопроцессоры могут быть полезными для ускорения операций поиска, проверки членства в множестве, выполнения операций дополнительной обработки (шифрование, обработка изображений, анализ аудиофайлов).

Существующие на данный момент NoSQL-решения не предназначены для использования в гибридных суперкомпьютерных системах. Таким образом, разработка методов и средств решения задач поиска и защищенного хранения данных с применением гибридных вычислительных технологий является актуальной научной и инженерной задачей.

Архитектура NoSQL-систем

Развитие NoSQL-систем в большой степени связано с активной работой научно-исследовательского сообщества. Существует ряд публикаций, оказавший серьезное влияние на процесс проектирования современных высокопроизводительных систем хранения данных. Большинство нынешних NoSQL-систем построены в соответствии с архитектурными принципами, положенными в основу таких решений, как Google's BigTable [4] и Amazon's Dynamo [5]. В NoSQL-системах используется агрегатно-ориентированная модель данных. В отличие от кортежей РСУБД агрегат – сложная структура данных, которая может содержать в себе списки значений или вложенных структур. Агрегаты можно распределить по узлам кластера, что увеличит общую

пропускную способность системы. В рамках одного агрегата обеспечивается согласованный доступ к данным. То, как именно представлен агрегат, определяет тип NoSQL-системы. Можно выделить три распространенных модели данных для агрегатов: «ключ-значение», «документная» и «семейство столбцов».

NoSQL системы типа «ключ-значение» предоставляют пользователю интерфейс, используя который он может получить некий объект по ключу (уникальному идентификатору). В качестве объекта каждому ключу сопоставлена запись, состоящая из нескольких атрибутов. Состав атрибутов для разных записей может отличаться. За обработку неоднородных записей отвечает разработчик приложения. Функциональным развитием систем «ключ-значение» являются документные NoSQL-системы. В качестве значения в них используется некий набор данных фиксированного формата (обычно JSON). Большое распространение получили системы хранения типа «семейство столбцов». Данные в таких системах представляются в виде ассоциативного массива, в котором в качестве ключей выступают имена столбцов. Ассоциативный массив и ключ объединяются в строку. Множество строк представляют собой семейство столбцов, хранящихся в этих строках.

Для повышения доступности данных в NoSQL-системах применяется механизм репликации (тиражирования) данных. Большинство NoSQL-систем используют одноранговую репликацию, что позволяет преодолеть ограничение производительности по записи. С узлами хранения (storage nodes) взаимодействует либо приложение напрямую (с использованием драйвера - NoSQL database driver), либо от имени пользователя в системе действует случайным образом выбранный сервер одноранговой сети (coordinator - Рисунок 1). Распределение данных по узлам сети происходит при помощи механизма непрерывного хеширования (хеш-кольца).

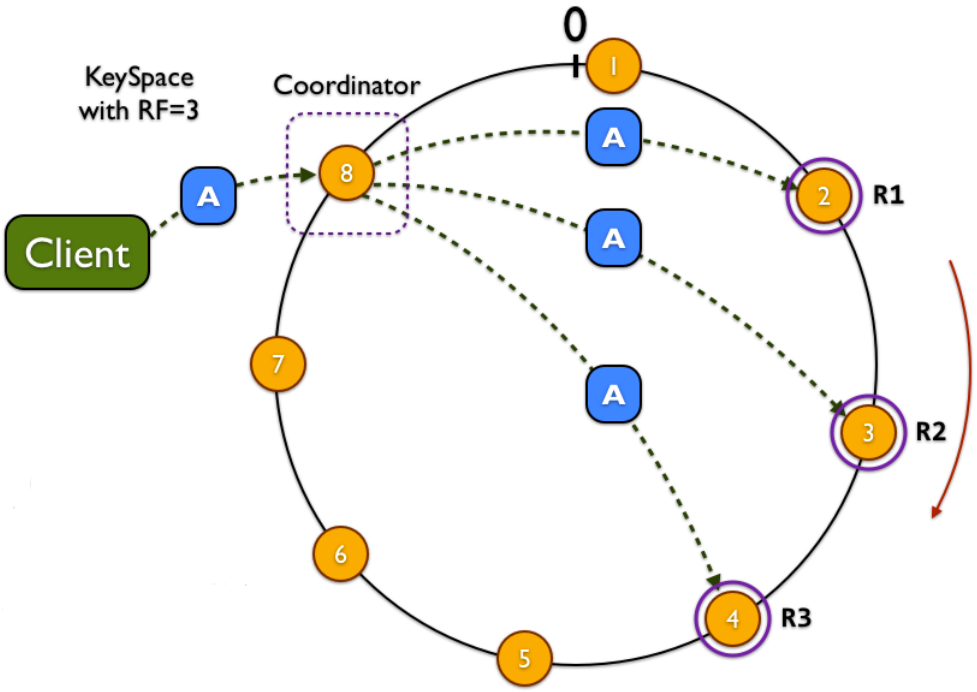

Рисунок 1 Непрерывное хеширование в системе. Коэффициент репликации - 3

Данный подход позволяет добавлять и исключать узлы, гибко настраивая балансировку в зависимости от производительности узлов. Основной проблемой одноранговой

репликации является задача обеспечения согласованности. Отсутствие согласованности может возникать, например, вследствие неполадок в сети.

Большинство NoSQL-систем являются системами хранения в основной памяти с возможностью «выталкивания» неиспользуемых таблиц во внешнюю память. Например, высокопроизводительная NoSQL-система Apache Cassandra (разработана в Facebook, используется компаниями Cisco, IBM, Cloudkick, Reddit, Digg, Rackspace и Twitter) задействует при работе с данными в ОЗУ журнал закрепления (commit log [6]) и файлы таблиц на жестком диске. При переполнении памяти (Java-heap [7]) создается новая таблица в памяти (memtable), а старая записывается на диск (SSTable). Операции записи в лог и выгрузки таблиц порождают последовательные обращения к жестким дискам, что ускоряет работу системы. Схема работы с памятью Apache Cassandra представлена на Рисунке 2.

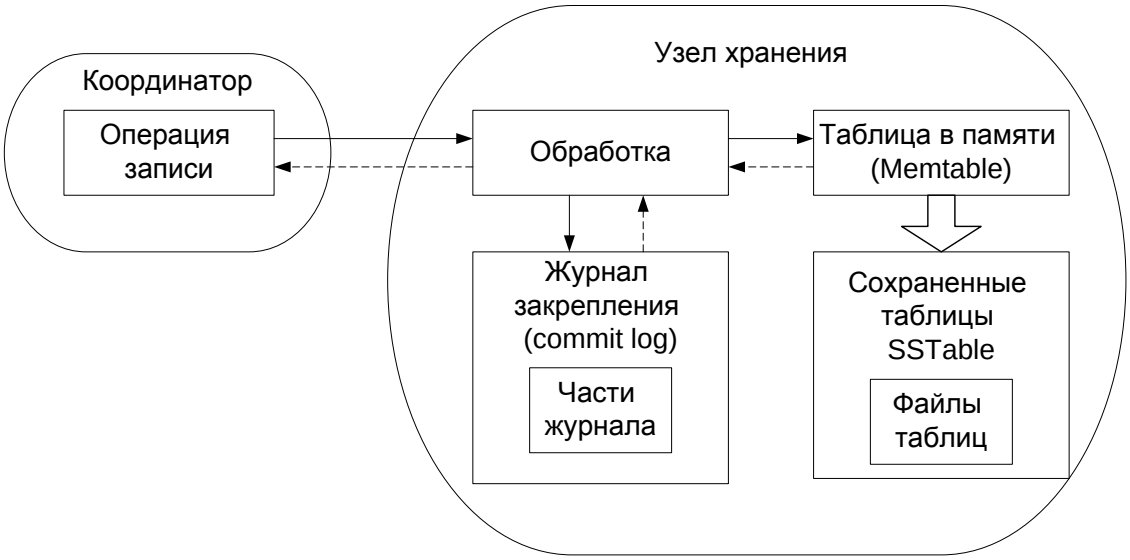

Рисунок 2 Организация памяти Apache Cassandra

Некоторые NoSQL-системы написаны на C++ (MongoDB), Erlang (CouchDB) и др., но большинство подобных решений изначально были разработаны на Java. Использование технологии Java при сопоставимых показателях производительности позволяет существенным образом повысить защищенность, посредством использования среды исполнения JRE. Однако, ни одна из существующих NoSQL-систем не использует сопроцессоры для ускорения вычислений. Отсутствие соответствующих средств и методов не позволяет эффективно использовать гибридные системы в качестве аппаратной платформы для NoSQL. Тем не менее, работа целого ряда механизмов NoSQL-систем может быть ускорена. Например, выполнение операций хеширования, в массовом порядке выполняющихся Координатором для проверки состояния вектора Блума [8] или хеш-кольца, существенным образом может быть ускорено за счет использования GPGPU-сопроцессора. Для ускорения поиска данных можно организовать соответствующий индекс в памяти GPGPU-сопроцессора, расположенного на узле хранения. Безусловно, в следствие недостаточного объема памяти ускорителя размер индекса будет ограничен. Но такой подход может оказаться полезным для ускорения поиска в какой-либо конкретной таблице по требованию. Наконец, сопроцессоры могут в десятки раз ускорить выполнение операций дополнительной обработки данных (прозрачное шифрование данных [9], вычисление сигнатур изображений и пр.). Для реализации вышеперечисленных возможностей необходимо: наличие расширяемого программного каркаса, позволяющего

подключать/отключать необходимые модули дополнительной обработки, наличие соответствующих параллельных алгоритмов, приспособленных для выполнения на GPGPU-сопроцессорах.

Связанные работы

С появлением гибридных вычислительных комплексов большое внимание стало уделяться вопросам применения GPU для ускорения операций поиска в древовидных структурах. В статье [10] Jordan Fix и др. показали, что применение технологии CUDA [11] для поиска в B+ дереве позволяет достичь более чем 10-кратного ускорения в операциях выборки. Но, если появляется необходимость в замене набора данных для поиска в памяти GPU на новый, такой подход оказывается неэффективным и приводит к многократному падению производительности по сравнению с CPU реализацией. Значительные накладные расходы появляются также вследствие необходимости синхронизации нитей CUDA. Комплексное исследование производительности многопоточных CPU и GPU реализаций B-деревьев провели Changkyu Kim и др. в работе [12]. Для GPU-реализации удалось достичь 1.7-кратного роста производительности по сравнению с предыдущей GPU-версией алгоритма. Исследователи столкнулись с серьезными проблемами в тех случаях, когда деревья не помещались в памяти графического процессора. В основном, ускорения удалось достичь за счет привязки реализации к конкретной архитектуре. Размеры узлов дерева, их количество на уровне и способ передачи напрямую определялись архитектурными особенностями ускорителя GTX 280. Экспериментальные результаты показывают, что из-за сильно ограниченного объема памяти GPU и большой латентности интерфейса передачи данных выигрыша в производительности удается достичь лишь в случае работы с редко меняющимися данными при условии, если программа хорошо адаптирована под конкретную разновидность GPU.

Существует несколько работ [13,14,15] в которых исследуется возможность применения CUDA-сопроцессора для ускорения фильтра Блума. В одних версиях вектор Блума размещается в разделяемой памяти, в других – в глобальной. Вектор Блума может и вовсе храниться в ОЗУ, основной выигрыш в производительности достигается за счет выполнения операций хеширования в АЛУ сопроцессора.

Значительное число работ связано с вопросами повышения производительности алгоритмов криптозащиты с использованием технологии CUDA. В работе [16] авторы анализируют преимущества технологии CUDA в сравнении с ранее использовавшейся OpenGL. Согласно логике работы, реализованной исследователями параллельной версии алгоритма AES, входная информация хранилась в глобальной памяти CUDA-устройства, а блоки замен были помещены в область кэшируемой константной памяти для ускорения процесса чтения данных из S-блока.

Авторы исследования [17] в своей работе выполнили обзор нескольких возможных подходов к реализации многопоточной версии AES. В статье приводятся результаты сравнения производительности последовательной, параллельной CPU и GPU версий. Как результат, самой производительной оказалась GPU-реализация.

В статье [18] авторы приводят обзор всех режимов шифрования AES и выделяют те из них пригодные для распараллеливания: ECB (Electronic codebook), CTR (Counter). Для режима CBC (Cipher-block chaining) возможно выполнить распараллеливание только операции

расшифрования. Для данных режимов авторы приводят графики пропускной способности и некоторые советы по повышению производительности.

Помимо статей, посвященных реализации алгоритма AES на CUDA существует также масса работ, описывающих реализацию параллельных версий прочих алгоритмов шифрования и хеширования на GPGPU-системах.

Программная платформа VAR

Как и любая современная NoSQL-система, программная платформа VAR предназначена для использования в одноранговых многомашинных сетях, состоящих из серверов доступа и узлов хранения данных. Схема распределения ролей по узлам системы может быть произвольной, одна и та же машина может быть и сервером доступа, и узлом хранения одновременно. Как показано на рисунке 3 в общем случае в системе присутствует K серверов доступа и M узлов хранения данных.

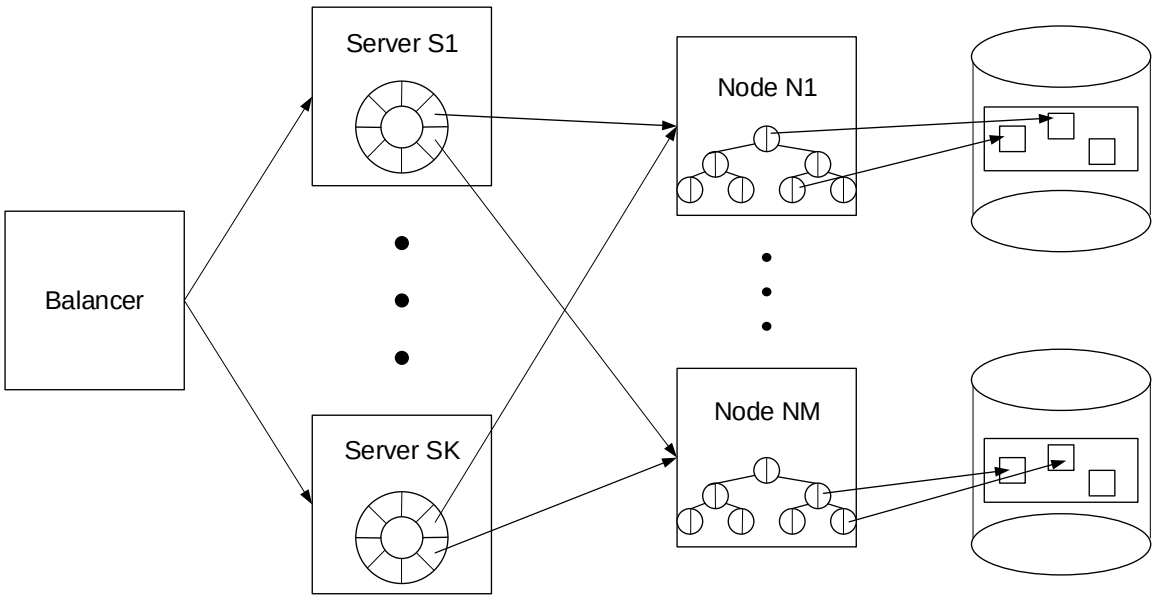

Рисунок 3 Обобщенная схема распределенной высокопроизводительной системы хранения и поиска данных VAR

Серверы доступа осуществляют координацию процессов поиска/добавления/удаления данных, отвечают за поддержку согласованности, транзакционности, содержат информацию о схеме расположения данных в системе

Узлы хранения управляют записью информации в оперативную или внешнюю память, осуществляют дополнительную обработку получаемых данных

Подсистема балансировки является достаточно специфическим компонентом, участие которого непосредственно в процессе работы с данными минимально, а основным предназначением является скорейшая переадресация нового входящего подключение на наименее загруженный сервер.

На рисунке 4 представлено схематичное изображение сервера доступа к данным. Состав сервера полностью определяется его назначением. Данный сервер одновременно должен «находиться» в двух сетях: внутренней сети хранения данных и внешней сети из которой поступают запросы пользователей. Для инициализации новых сетевых подключений в

составе сервера предусмотрены сетевые подсистемы внешних и внутрисистемных обменов, представляющие собой программные модули, создающие сокетные TCP-подключения и выбирающие соответствующий обработчик. После инициализации управление вновь созданным подключением передается подсистеме управления сетевой активностью.

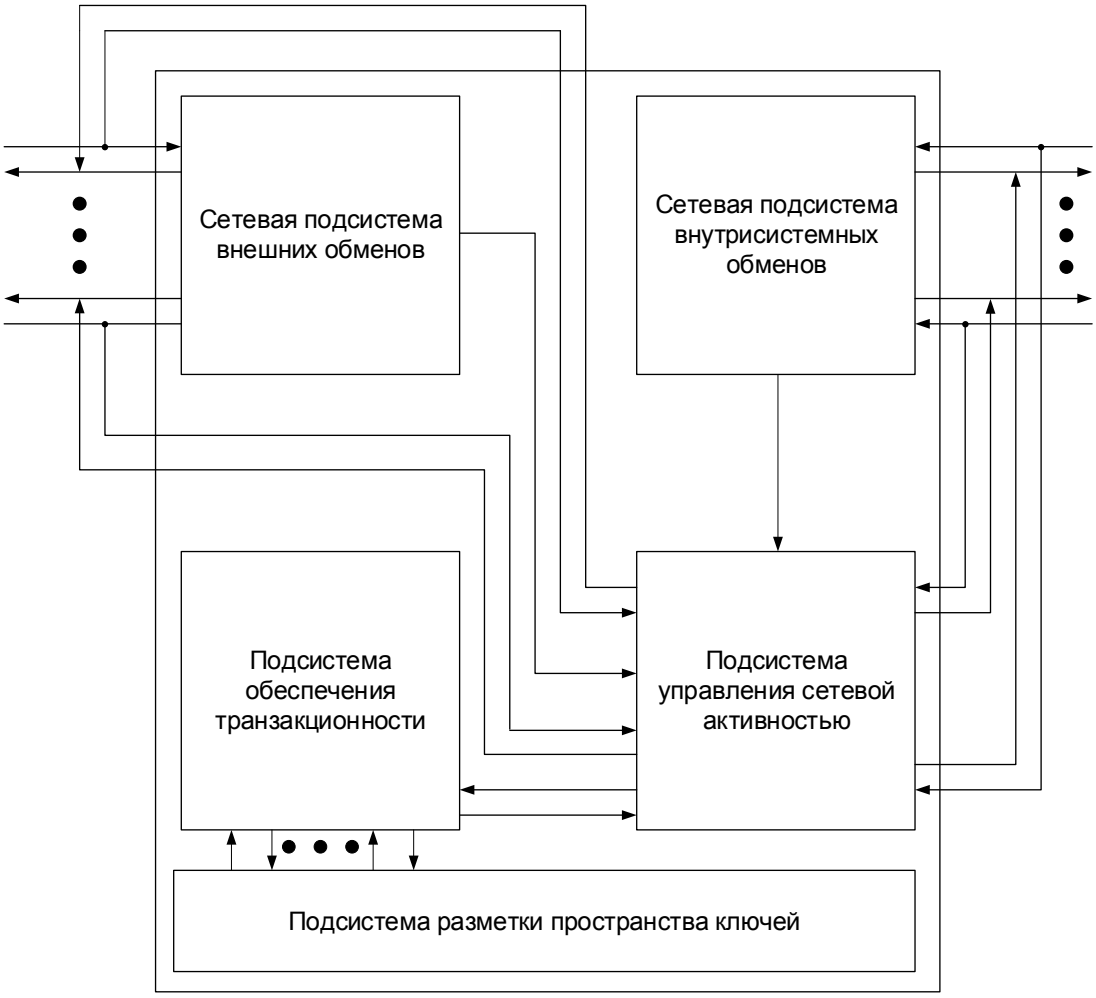

Рисунок 4 Схема сервера доступа к данным

Подсистема обеспечения транзакционности собирает статистику об этапах выполнения запросов в распределенной системе и отправляет управляющие воздействия подсистеме управления сетевой активностью. Характер управляющих воздействий может быть различным:

- Подтверждение успешности выполнения записи, фиксация новых данных на узлах хранения
- Отправка повторного запроса о записи узлам хранения
- Отправка пользователю системы сообщения об успешной фиксации изменений
- Запрос информации о версиях хранимых данных
- Отправка периодического сигнала синхронизации версий
- и др.

Для определения целевых узлов хранения координатор взаимодействует с подсистемой разметки пространства ключей, которая представлена хеш-кольцом.

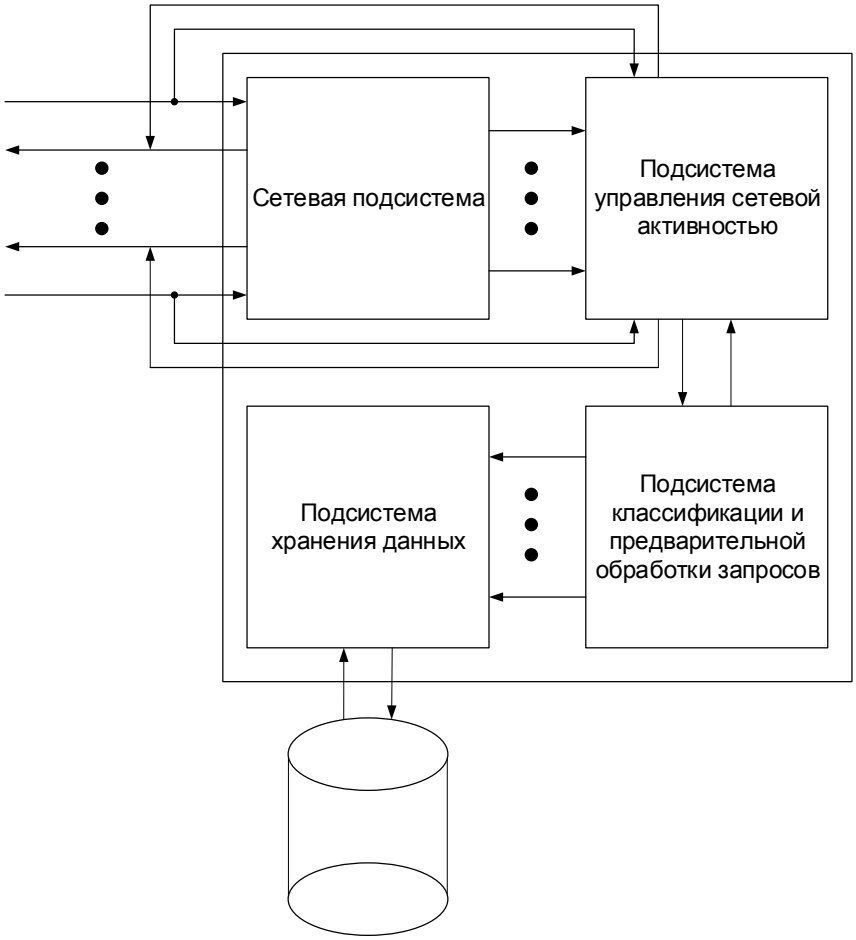

Рисунок 5 Схема узла хранения данных

На рисунке 5 приведено схематичное изображение узла хранения данных. Состав узла хранения частично повторяет состав сервера доступа к данным. Для обработки внутрисистемных сетевых соединений присутствуют сетевая подсистема и подсистема управления сетевой активностью.

Получаемые сетевыми подсистемами запросы поступают на вход подсистемы классификации и предварительной обработки запросов. На данном этапе считывается тип запроса и при необходимости выполняется дополнительная обработка получаемых данных.

Непосредственно за хранение и поиск данных отвечает подсистема хранения данных, которая представляет собой memcached-систему с возможностью выгрузки данных во внешнюю память.

Система VAR написана на Java. Сетевая подсистема разработана на базе технологии Java NIO.2 [19] и является масштабируемым многопоточным решением. Формат сообщений, обрабатываемых в системе - бинарный. Ускорение вычислений организовано посредством использования CUDA-сопроцессора, для взаимодействия с которым используется библиотека jCuda [20]. Программная платформа VAR включает в себя программный каркас, состоящий из иерархии классов, интерфейсов и их базовых реализаций, которые могут быть изменены и дополнены посредством использования механизма Dependency Injection [21] в зависимости от потребностей пользователя и аппаратной конфигурации. Подсистема обеспечения транзакционности сервера доступа и подсистема хранения данных используют CUDA-сопроцессор для ускорения операций:

хеширования, создания индексов таблиц в памяти GPU, шифрования данных при выгрузке во внешнюю память и рандомизации ключей на входе стохастического дерева поиска [22,23]. Шифрование данных и выгрузка их во внешнюю память осуществляется отдельной Java-нитью, использующей собственный контекст CUDA-потока, и выполняется в фоновом режиме.

Конфигурация

В настоящей статье представлены результаты тестирования производительности узла хранения программного каркаса VAR. Для сравнения приводятся результаты замеров производительности Apache Cassandra в одноузловой конфигурации с коэффициентом репликации 1. Конфигурация тестовой системы: Intel Core i7 – 3770K, 32 Gb DDR3, NVIDIA GeForce GTX680 (4Gb GDDR5). Генерация тестовых пакетов происходила посредством использования утилиты нагрузочного тестирования производительности Apache Jmeter [24]. Данная утилита позволяет создавать множество потоков, имитирующих работу пользователей. В нашем случае в тесте участвовало до 100 виртуальных пользователей, каждый из которых генерировал запросы на добавление (1Кбайт данных в каждом запросе) и поиск данных. В системе VAR был включен модуль шифрования данных по алгоритму ГОСТ 28147-89 в фоновом режиме при выгрузке во внешнюю память. Шифрование осуществлялось на GPU. Для сравнения на Рисунке 6 приведены результаты unit-тестирования последовательной однопоточной CPU и параллельной GPU версий.

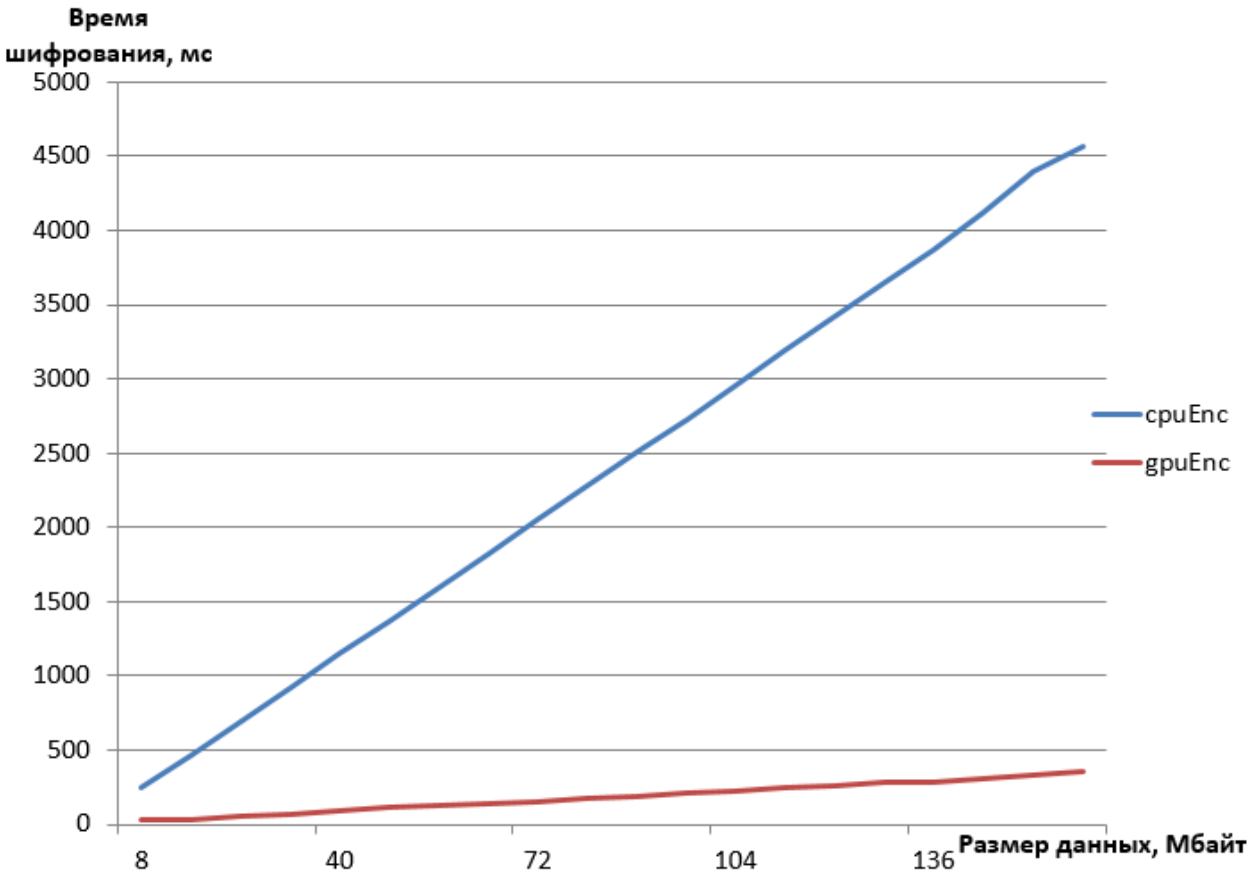

Рисунок 6 Производительность шифрования данных по алгоритму ГОСТ 28147-89 для CPU и GPU версий

Основные результаты и дальнейшие исследования

Нагрузочное тестирование осуществлялось следующим образом: Jmeter наращивал чило виртуальных пользователей (VU на рисунках ниже) до 100, после чего снова снижал до 0 и завершал тестирование. При этом фиксировались три основных показателя – количество транзакций в секунду (TPS), наличие ожидаемого кода возврата и время отклика (response time).

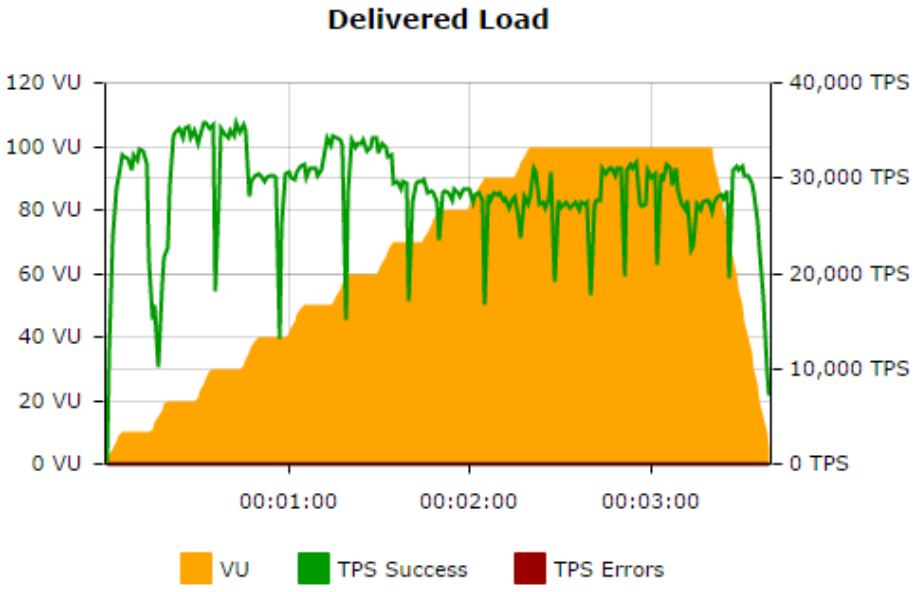

Рисунок 7 Производительность (система VAR)

Из графика на рисунке 7 видно, что при 100 виртуальных пользователях один узел хранения системы VAR способен поддерживать производительность на уровне 27k транзакций в секунду с небольшими «просадками» во время переинициализации memcached-таблицы. Из распределения времени отклика (Рисунок 8) видно, что средний показатель времени отклика составляет 2 ms, что является отличным показателем для NoSQL-систем.

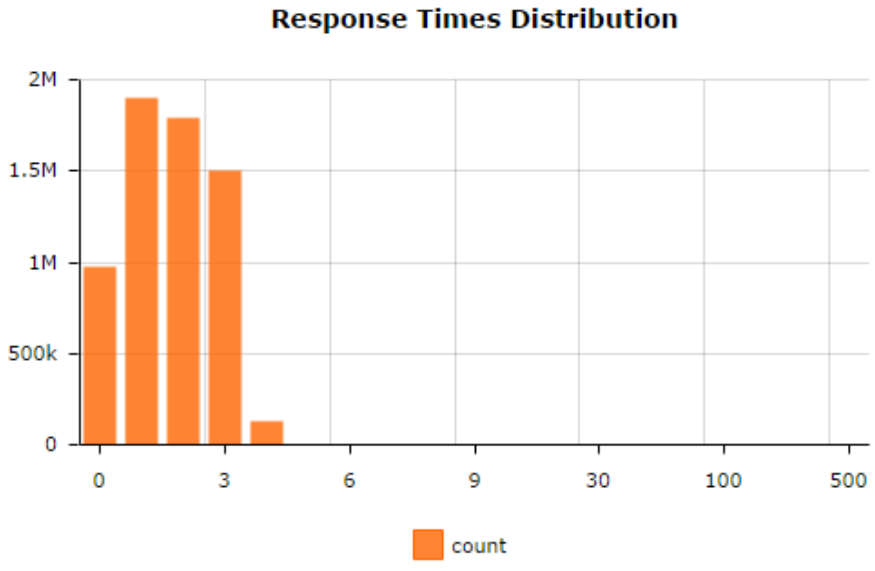

Рисунок 8 Время отклика (система VAR)

Результаты однотипного испытания аналогичного экземпляра Apache Cassandra представлены на рисунках 9 и 10. В среднем, можно отметить, что Apache Cassandra уступает по производительности системе VAR на 35-40% и имеет в двое большее среднее время отклика.

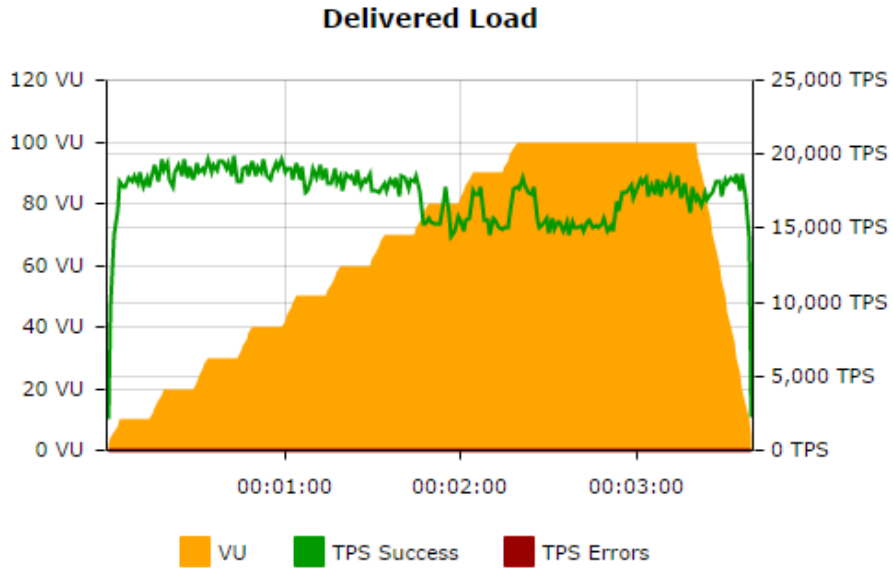

Рисунок 9 Производительность (Apache Cassandra)

Важно также отметить, что при более скромных показателях Cassandra не осуществляет шифрование выгружаемых на диск данных.

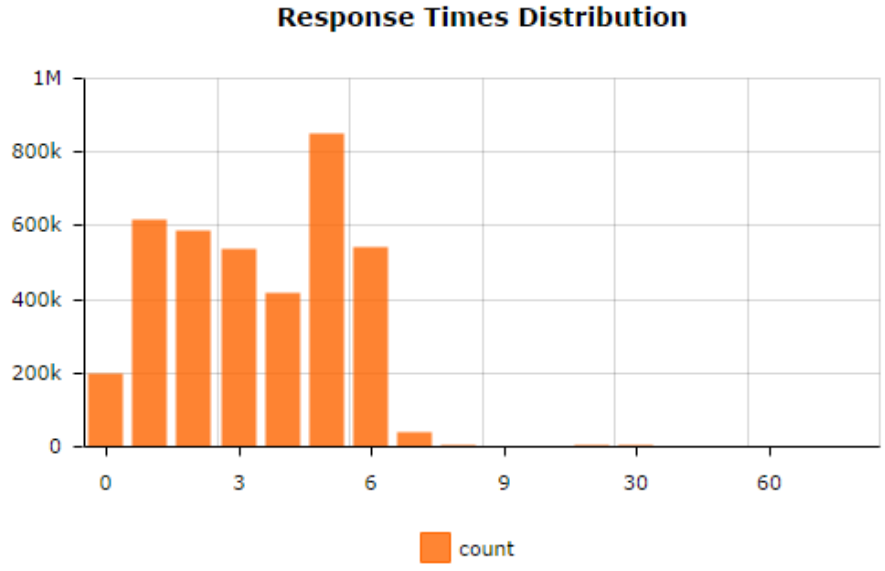

Рисунок 10 Время отклика (Apache Cassandra)

В качестве дальнейших исследований планируется повторить аналогичные испытания для индексируемых на GPU таблиц, протестировать многомашинную конфигурацию VAR, исследовать прочие алгоритмы шифрования и их работу в качестве модуля дополнительной обработки данных перед выгрузкой на диск.

Заключение

Согласно стратегии импортозамещения в сфере ИТ планируется разработать целый ряд программных продуктов (операционная система, гипервизор и др.). На закрытом заседании Совета Федерации РФ [25] принята дорожная карта по разработке необходимых нормативных правовых актов. В соответствие с этим решением планируется установить квоту на закупку госзаказчиками российского ПО и закрепить основные полажения постановлением правительства. Стратегия импортозамещения предполагает некий список критически-важного ПО, которое должно быть разработано отечественными компаниями. К такому ПО, в частности, относят системы управления базами данных.

Описанная в настоящей статье высокопроизводительная система хранения и поиска данных VAR представляет собой СУБД, разработанную на языке Java без использования проприетарных библиотек и технологий. По своим показателям VAR не только не уступает, но и превосходит ведущие зарубежные аналоги. Программная платформа VAR является расширяемым решением, использующим GPU-сопроцессор для ускорения некоторых операций (включая шифрование по стандарту ГОСТ 28147-89) и повышения общей энергоэффективности системы. Таким образом, можно утверждать, что программная платформа VAR может быть использована для замещения зарубежных высокопроизводительных NoSQL-систем.

Литература

1. Ian H. Witten, Eibe Frank and Mark A. Hall Data Mining: Practical Machine Learning Tools and Techniques. — 3rd Edition. — Morgan Kaufmann, 2011. — P. 664.

2. Pramod J. Sadalage, Martin Fowler NoSQL Distilled. — 1 edition — Addison-Wesley Professional, 2012. — P. 192.

3. List of Top500 supercomputers – Официальный портал URL: http://top500.org (дата обращения: 23.09.14)

4. Fay Chang, Jeffrey Dean, Sanjary Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber: "BigTable: a Distributed Storage System for Structured Data". *Proc. 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*. USENIX Association, 2006

5. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels: "Dynamo: Amazon's Highly Available Key-Value Store". *SOSP'07: Proc. Twenty-First ACM SIGOPS Symposium on Operating Systems Principles*, pages 205–220, 2007

6. Apache Cassandra™ 2.1 – Официальная документация Apache Cassandra URL: http://www.datastax.com/documentation/cassandra/2.1/cassandra/gettingStartedCassandraIntro.html (дата обращения: 23.09.14)

7. Herbert Schildt. Java The Complete Reference, 8th Edition, McGraw-Hill Osborne Media, pp.1152, 2011

8. B. Bloom. Space / time trade-offs in hash coding with allowable errors. Communications of ACM, 13(7): 422-426, 1970

9. Transparent Data Encryption (TDE) – Официальная документация SQL Server 2014 URL: http://msdn.microsoft.com/en-us/library/bb934049.aspx (дата обращения: 23.09.14)

10. Jordan Fix, Andrew Wilkes, Kevin Skadron «Accelerating Braided B+ Tree Searches on a GPU with CUDA» A4MMC 2011 : 2nd Workshop on Applications for Multi and Many Core Processors, 2011

11. Боресков А.В. и др. Параллельные вычисления на GPU. Архитектура и программная модель CUDA: Учеб. пособие, М.:Издательство МГУ, 2012

12. Changkyu Kim, Jatin Chhugani, Nadathur Satish et al, «FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs», ACM SIGMOD International Conference on Management of data, pp. 339-350, 2010

13. Lin Ma, Roger D. Chamberlain, Jeremy D. Buhler, and Mark A. Franklin, «Bloom Filter Performance on Graphics Engines», in Proc. of International Conference on Parallel Processing, September 2011, pp. 522-531

14. Venkatesh Rao «Implementation of the Bloom Filter on GPU using CUDA», University of Minnesota, 2010

15. Vasilyev N.P., Rovnyagin M.M. et al. Modified Bloom filter for high-performance data search in hybrid computing systems // International Conference «The Radio-Electronic Devices and Systems for The Infocommunication Technologies» (RES-2013), Moscow, P. 167-170, 2013

16. S. A. Manavski CUDA compatible GPU as an efficient hardware accelerator for AES cryptography // ICSPC 2007 IEEE Los Alamitos, pp. 65-68, November 2007.

17. Ortega, J. Trefftz, H. Trefftz, Parallelizing AES on multicores and GPUs, IEEE International Conference on Electro/Information Technology (EIT), 2011, vol., no., pp.1-5, 15-17 May 2011.

18. Qinjian Li et al. Implementation and Analysis of AES Encryption on GPU, Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, p.843-848, June 25-27, 2012.

19. Anghel Leonard Pro Java 7 NIO.2. — Apress, 2011. — P. 296.

20. jcuda.org - Java bindings for CUDA (официальный веб-сайт) URL: http://www.jcuda.org/ (дата обращения: 23.09.14).

21. Craig Walls Spring in Action, Third Edition — Manning, 2011. — P. 424.

22. Ровнягин М.М. Стохастическое бинарное дерево поиска в задачах поиска и хранения данных для высокопроизводительных NoSQL-систем // Сборник докладов 17-й Международной телекоммуникационной конференции молодых ученых и студентов «Молодежь и наука», М.:МИФИ, С. 38-39, 2014

23. Васильев Н.П., Ровнягин М.М. Организация поиска данных в суперкомпьютерных системах на базе NoSQL-подхода и технологии NVIDIA CUDA // Национальный Суперкомпьютерный Форум, Переславль-Залесский, http://2013.nscf.ru/TesisAll/Section%206/13_1350_RovnyaginMM_S6.pdf, 2013 (электронная)

24. Apache JMeter™ – Официальный Web-ресурс URL: http://jmeter.apache.org/ (дата обращения: 23.09.14)

25. Меры по импортозамещению в сфере IT – ГОСЗАКАЗ: портал государственных закупок URL: http://www.pro-goszakaz.ru/news/95474/ (дата обращения: 23.09.14)

Сведения об авторах

ФИО: Васильев Николай Петрович
Место работы и должность в настоящее время: НИЯУ МИФИ, доцент
Год окончания учебного заведения и его полное название: 1995 г. Московский инженерно-физический институт
Ученая степень и звание: к.т.н., доцент
Количество печатных работ и монографий: более 70
Область научных интересов: Облачные вычисления, гибридные суперкомпьютерные технологии, защита информации
Контакты: NPVasilyev@mephi.ru, +7(916)339-7937
Name: Vasilyev Nikolay Petrovich
Position: Associate Professor
Year of graduation: 1995
Academic degree: PhD
Number of publications and monographs: More than 70 publications
Research interests: Cloud computing, Hybrid supercomputer technologies, Information Security
Contact: NPVasilyev@mephi.ru, +7(916)339-7937

ФИО: Ровнягин Михаил Михайлович
Место работы и должность в настоящее время: аспирант, ассистент кафедры компьютерные системы и технологии НИЯУ МИФИ
Год окончания учебного заведения и его полное название: 2012 Национальный исследовательский ядерный университет «МИФИ»
Ученая степень и звание: нет
Количество печатных работ и монографий: более 10
Область научных интересов: Суперкомпьютерные технологии, организация поиска данных в высоконагруженных системах, разработка и верификация аппаратуры
Контакты: rovnyagin@gmail.com +7(915)2270086
Name: Michael M. Rovnyagin
Position: a postgraduate student, instructor in the Department of Computer Systems and Technologies NRNU MEPhI
Year of graduation: 2012 National Research Nuclear University «MEPhI»
Academic degree: no
Number of publications and monographs: More than 10 publications
Research interests: Supercomputer technology, data search in high load systems, development and verification of digital hardware
Contact: MMRovnyagin@mephi.ru +7(915)2270086