УДК 004.75, 004.382.2 DOI: 10.14529/cmse170403

МЕТОДЫ И СРЕДСТВА ОРГАНИЗАЦИИ ГЛОБАЛЬНОЙ ОЧЕРЕДИ ЗАДАНИЙ В ТЕРРИТОРИАЛЬНО РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ*

© 2017 А.В Баранов, А.И Тихомиров

Межведомственный суперкомпьютерный центр Российской академии наук – филиал Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук»

(119334 Москва, Ленинский пр-т., д. 32а)

E-mail: tema4277@rambler.ru

Поступила в редакцию: 01.09.2017

В статье рассмотрена модель территориально распределенной вычислительной системы (ТРС), состоящей из объединенных каналами связи высокопроизводительных вычислительных установок. Вычислительные установки из состава ТРС представляют собой высокопроизводительные кластеры, различающиеся по архитектуре и производительности. Объединяющие их каналы связи имеют разные надежность и пропускную способность. Особенностью рассматриваемой модели является децентрализованная схема управления заданиями. Подобная схема подразумевает, что любая вычислительная установка в любой момент времени может выйти из состава ТРС по причине своей неисправности или неисправности канала связи. Устранение неисправности означает динамическое подключение вычислительной установки к ТРС. В этих условиях в ТРС организуется глобальная очередь заданий с абсолютными приоритетами, из которой задания распределяются по свободным ресурсам вычислительных установок. Абсолютные приоритеты предполагают вытеснение с выполнения низкоприоритетного задания поступившим в очередь высокоприоритетным заданием. Для формирования и хранения глобальной очереди заданий в условиях динамически изменяющегося состава ТРС необходима надежная распределенная информационная система (РИС). В качестве основы для ее построения авторами рассмотрен ряд известных распределенных СУБД. В статье сформулированы требования к РИС, проведен сравнительный анализ и сделан выбор решения, удовлетворяющего требованиям, рассмотрен разработанный авторами макет ТРС с децентрализованной схемой диспетчеризации заданий.

Ключевые слова: территориально распределенная система, распределенная информационная система, абсолютные приоритеты вычислительных заданий.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Баранов А.В., Тихомиров А.И. Методы и средства организации глобальной очереди заданий в территориально распределенной вычислительной системе // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 4. С. 28–42. DOI: 10.14529/cmse170403.

Введение

С целью повышения производительности и надежности расчетов отдельные высокопроизводительные *вычислительные установки* (ВУ) нередко объединяются в *территориально распределенные вычислительные системы* (ТРС). Наибольшее распространение получила многоуровневая форма организации ТРС, когда в качестве ВУ используются высокопроизводительные *вычислительные кластеры* [1]. Для управления

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2017».

отдельной ВУ применяется *локальная система управления ресурсами* (ЛСУР), действия всех ВУ координирует *глобальная система управления ресурсами* (ГСУР). После включения ВУ в состав ТРС вычислительные ресурсы этой ВУ не отчуждаются от их владельца и продолжают использоваться для выполнения локальных заданий, которые образуют *локальный поток заданий* [2]. После включения ВУ в состав ТРС совместно с заданиями локального на вычислительные ресурсы ВУ начинают поступать задания с других ВУ — задания из *глобального потока заданий*.

Очередность выполнения заданий устанавливается с использованием приоритетов (абсолютных или относительных). Для организации в территориально распределенной системе обработки вычислительных заданий в соответствии с приоритетами, требуется ведение *единой глобальной очереди заданий*, в которой все поступившие задания ранжируются, а затем распределяются строго в соответствии с приоритетом. Для ведения глобальной очереди заданий требуется наличие единой для всей ТРС *информационной системы*, исследование методов и средств построения которой и является целью настоящей работы.

Статья организована следующим образом. Раздел 1 посвящен исследуемой авторами модели ТРС, ее отличительным свойствам и характеристикам. В разделе 2 рассматриваются различные варианты организации глобальной системы управления ТРС, делается выбор в пользу децентрализованной схемы диспетчеризации с единой информационной системой. Организации такой информационной системы посвящен раздел 3. В разделе 4 обосновывается выбор распределенной СУБД в качестве основы информационной системы ТРС. Практическая реализация выбранного решения в виде макета ТРС рассмотрена в разделе 5. В заключении подводятся итоги и намечаются перспективы развития дальнейших исследований.

1. Модель территориально распределенной системы

В исследуемой авторами модели ТРС каждая ВУ представляется высокопроизводительным вычислительным кластером. Логически каждая ВУ может быть разделена на четыре взаимосвязанные подсистемы: *серверную, исполнительную, управляющую и подсистему доступа*. Каждая подсистема физически представляется набором из нескольких рабочих станций (вычислительных модулей), на которых установлены программные компоненты, соответствующие назначению подсистемы.

Серверная подсистема представлена серверами баз данных и используется для хранения данных, необходимых для функционирования ВУ: локальной и глобальной очередей заданий, исходных данных заданий.

Исполнительная подсистема представляет собой решающее поле кластера — набор вычислительных модулей, объединенных высокоскоростной низколатентной сетью.

Управляющая подсистема применяется для управления исполнительной подсистемой с использованием ЛСУР и диспетчеров заданий.

Подсистема доступа обеспечивает контроль поступающего в ВУ сетевого трафика и служит точкой входа в ВУ для удаленных пользователей, а также подсистем других ВУ из состава ТРС.

В исследуемой модели (рис. 1) допускается, что различные ВУ могут содержать разное число вычислительных модулей разной производительности и архитектуры, т.е. для ТРС характерна гетерогенность как для системы в целом, так и внутри отдельной ВУ.

Управление на уровне локальных ресурсов осуществляет ЛСУР, основными функциями которой являются ведение локальной очереди заданий, запуск и контроль процесса выполнения заданий на вычислительных ресурсах ВУ. Примерами ЛСУР ВУ могут служить известные системы пакетной обработки, такие как PBS, SLURM, Moab или отечественная Система управления прохождением параллельных заданий (СУППЗ) [3].

Единицей обработки информации в ТРС является *вычислительное задание*, под которым понимается набор, включающий входные данные, программу их обработки и паспорт задания. *Паспорт задания* – специальный объект, описывающий *ресурсный запрос задания*: количество процессоров (ядер), объем оперативной памяти и дискового пространства, *приоритет задания* и др. Различные задания от разных пользователей образуют поток заданий. В каждой ВУ всегда присутствует *локальный поток заданий*, задания которого допускают обработку только на локальных вычислительных ресурсах ВУ.

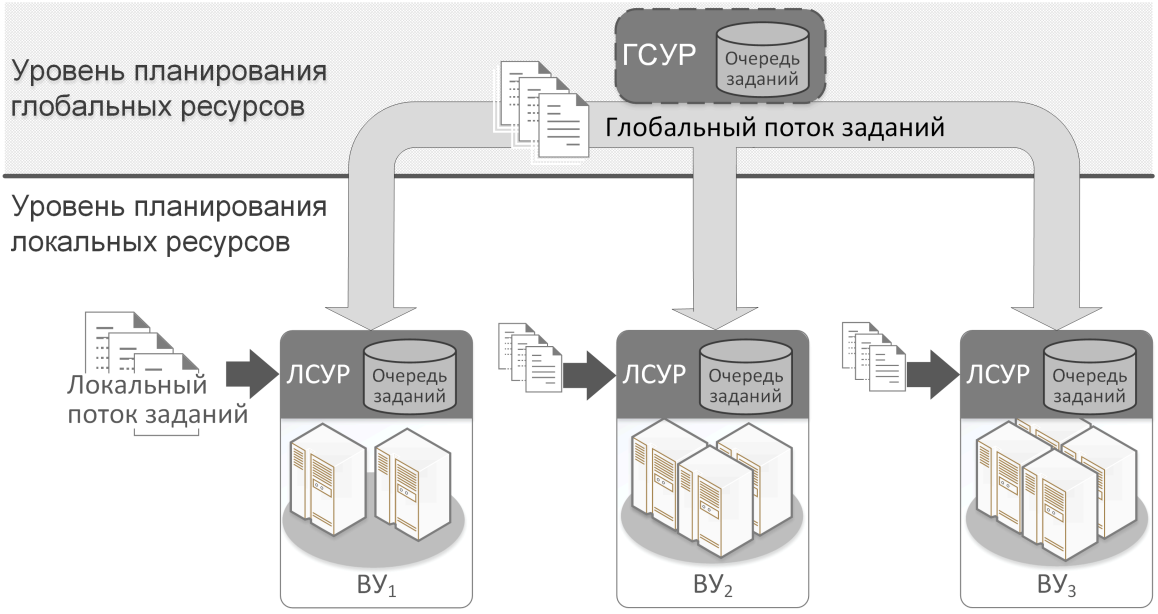

Рис. 1. Модель территориально распределенной вычислительной системы

Управление на уровне планирования глобальных ресурсов ТРС осуществляет ГСУР, основной функцией которой является планирование заданий глобального потока. В отличие от локального, задания глобального потока допускают обработку на вычислительных ресурсах любой ВУ ТРС и распределяются в ВУ, время обработки в которой будет минимальным. Важно отметить, что ГСУР не планирует задания на локальные ресурсы ВУ, а только выбирает т.н. *целевую ВУ* для размещения задания. Размещенное задание глобального потока поступает под управление ЛСУР целевой ВУ и планируется наряду с заданиями локального потока.

Таким образом, на вычислительные ресурсы ВУ поступает два потока задания: локальный и глобальный. Заданиям обоих потоков доступны все вычислительные ресурсы [2]. Очередность предоставления вычислительных ресурсов устанавливается в соответствии с приоритетами заданий. В исследуемой модели ТРС предусматриваются абсолютные приоритеты, отличие которых от относительных состоит в использовании механизма вытеснения высокоприоритетными заданиями менее приоритетных [4].

Существенной характеристикой исследуемой модели ТРС является ее динамически изменяющийся состав. Модель допускает в любой момент времени как включение в состав ТРС новых ВУ, так и исключение из состава ТРС отдельных ВУ. Выключение из состава ТРС (включение в состав ТРС) некоторой ВУ может быть связано как с неисправностью (возвращением в строй) самой ВУ, так и с неисправностью (возвращением в строй) коммуникационных каналов.

2. Варианты организации глобальной системы управления ТРС

Для организации ГСУР могут быть использованы *централизованная, иерархическая, децентрализованная* архитектуры [5]. При централизованной архитектуре в качестве управляющего центра назначается некоторая ВУ, на которую возлагается функция распределения вычислительных заданий глобального потока по доступным ВУ ТРС. Существенным недостатком централизованной архитектуры, ограничивающим возможность ее применения в ТРС, является наличие в системе единой точки отказа – центра управления. Централизованная диспетчеризация применяется в системах [6,7].

При иерархической архитектуре в системе выделяется несколько логических уровней, а также несколько центров управления, каждый из которых ассоциируется с одним из уровней. Между центрами управления разных уровней устанавливается иерархическая зависимость, то есть каждый центр управляет центрами нижележащего уровня. Логически иерархическая структура представляется в виде дерева, промежуточными вершинами и корнем которого являются центры управления. При этом у каждой вершины помимо связи с вышележащей вершиной может существовать связь с центрами управления одного логического уровня. В случае неисправности одного из центров управления исходное дерево разделяется на одно или более поддеревьев, каждое из которых начинает функционировать автономно. Наличие связей между центрами управления одного уровня позволяет путем перестройки восстановить исходное дерево без восстановления неисправного центра. Механизм перестройки дерева основан на назначении центрам новых зависимостей (могут изменяться уровни). Необходимость перестройки дерева в случае неисправности или недоступности какого-либо центра управления является главным недостатком иерархических систем.

С точки зрения авторов, наибольший интерес для дальнейших исследований представляет децентрализованная архитектура ГСУР, обеспечивающая надежность и масштабируемость ТРС. При децентрализованной архитектуре ГСУР в ТРС управление осуществляется *коллективом равноправных диспетчеров*, которые располагаются локально на каждой ВУ ТРС. Неисправность одного из диспетчеров никак не сказывается на функционировании остальных. При этом решения о распределении каждого задания глобальной очереди принимаются согласованно. Децентрализованный подход применяется, например, в системе PAUA [8].

Согласованность решения о распределении задания достигается коллективом диспетчеров либо в результате их *непосредственного взаимодействия*, либо *взаимодействия с использованием единой информационной системы* [9]. При непосредственном взаимодействии диспетчеры напрямую обмениваются вычислительными заданиями друг с другом, в этом случае распределение заданий основывается на *жадном алгоритме* [10]. Жадный алгоритм предполагает, что все задания глобального потока

заданий распределяется во все ВУ ТРС в момент их поступления. При таком подходе в ТРС отсутствует информация о загруженности ВУ, что приводит к невозможности организации приоритетной обработки заданий. Кроме этого, возникает опасность длительного пребывания заданий в локальной очереди ВУ после их распределения по причине большого числа заданий с высоким приоритетом, поступивших из локального потока. Этот недостаток может быть устранен путем реализации в ГСУР механизма перераспределения заданий, что в свою очередь является отдельной сложной задачей [11].

Второй способ взаимодействия диспетчеров — взаимодействие с использованием *единой информационной системы*. Способ предполагает распределение заданий *ленивым алгоритмом планирования*. В этом случае все задания глобального потока поступают в информационную систему, в которой формируется и хранится *глобальная очередь заданий*. Задания распределяются в ВУ только тогда, когда для их выполнения доступны необходимые вычислительные ресурсы. Контроль доступности вычислительных ресурсов ВУ обеспечивает соответствующий диспетчер. Наличие в системе глобальной очереди позволяет распределять задания с учетом их приоритетов, т.е. сначала диспетчеры обрабатывают задания с максимальным приоритетом, а затем — менее приоритетные.

3. Организация информационной системы ТРС

Рассмотренные авторами в предыдущих работах [4] ленивые алгоритмы планирования позволили определить следующий сценарий работы диспетчеров с глобальной очередью заданий.

1. Добавление заданий в глобальную очередь. Все новые задания глобального потока диспетчер помещает в информационную систему. При этом исходные данные задания остаются на ВУ, поместившей задание в глобальную очередь, и передаются на целевую ВУ только после назначения туда задания.
2. Изменение параметров задания в глобальной очереди. Пользователь, поместивший задание в информационную систему, может изменить его параметры, например, повысить приоритет. Задание может быть изменено и диспетчером, например, в случае если диспетчер изменяет статус выполнения задания.
3. Поиск вычислительных заданий. Диспетчер ВУ, вычислительные ресурсы которой свободны и готовы принять новое задание, осуществляет соответствующий запрос в информационную систему для получения нового задания. В поисковом запросе указываются критерии поиска: минимальный приоритет задания, требуемое число вычислительных модулей и др. Для обработки таких запросов требуется возможность анализа всех параметров всех заданий глобальной очереди.
4. Извлечение распределенного задания из глобальной очереди. Задание, удовлетворяющие критериям поискового запроса, извлекается диспетчером из глобальной очереди, но при этом информация о задании остается в информационной системе для отслеживания состояния его выполнения.

Приведенный сценарий позволяет сформулировать следующие требования к информационной системе для организации глобальной очереди заданий.

1. Доступность и масштабируемость. Информационная система должна всегда быть доступна для всех диспетчеров, а также обеспечивать масштабируемость ТРС.
2. Соответствие децентрализованной архитектуре ГСУР.

3. Надежность хранения глобальной очереди. В информационной очереди глобальная очередь должна сохраняться независимо от динамически изменяющегося состава TPC. Неисправность отдельных компонентов TPC не должна приводить к потере заданий, хранящихся в глобальной очереди.
4. Организация поиска данных по критериям (по приоритетам). Для извлечения заданий из глобальной очереди требуется выполнение сложных поисковых запросов, предусматривающих анализ отдельных полей заданий (например, приоритетов заданий) или поиск с использованием фильтров.
5. Возможность изменения параметров и извлечения заданий из очереди. Требуется обеспечить возможность изменения параметров заданий в глобальной очереди таким образом, чтобы извлеченные определенным диспетчером задания не фигурировали в результатах поиска других диспетчеров.

Рассматривая варианты организации информационной системы TPC, следует учитывать, что, как и в случае с ГСУР, возможно применение одного из трех видов архитектуры: централизованной, иерархической и децентрализованной. Преимущества и недостатки каждого из трех подходов были рассмотрены выше при выборе архитектуры ГСУР. Здесь отметим особенности и примеры каждой архитектур применительно к организации информационной системы TPC.

Централизованная архитектура информационной системы предполагает выделение отдельного компонента, отвечающего за хранение глобальной очереди заданий и обработку поступающих от диспетчеров заданий запросов. Основой централизованной информационной системы может выступать любая из известных СУБД: PostgreSQL, Oracle и др. Достоинствами такого решения является простота организации и контроля целостности заданий, хранящихся в глобальной очереди заданий. Недостаток, присущий всем централизованным архитектурам, заключается в наличии единой точки отказа.

В случае иерархической организации информационная система распределяется между несколькими серверами, каждый из которых ассоциируется с определенным логическим уровнем. Высший уровень содержит агрегированные сведения о нижележащих уровнях. При невозможности выполнить запрос, поступивший на один из уровней, происходит обращение к более высоким уровням системы. Примером иерархической организации является система MDS (Metacomputing Directory Service), основанная на распределенной службе каталогов LDAP (Lightweight Directory Access Protocol) [12]. MDS применялась в составе набора инструментов Globus Toolkit [13] вплоть до версии 4.0.

Децентрализованная архитектура информационной системы применима только при децентрализованной схеме диспетчеризации вычислительных заданий. В этом случае в TPC отсутствует единая информационная система, вместо этого у каждого диспетчера имеется собственное хранилище. Децентрализованное решение обеспечивает масштабируемость и надежность TPC, однако не позволяет распределять задания в соответствии с единым для всей TPC критерием очередности распределения заданий, определяемой их приоритетами. Для выполнения последнего требования необходимо, чтобы хранилище данных каждого диспетчера было согласовано с хранилищами остальных диспетчеров. Добиться этого можно за счет использования распределенного подхода, при котором информационная система представляется коллективом равноправных, взаимодействующих подсистем хранения. В этом случае достигается сочетание преимуществ централизованного и децентрализованного подходов.

Применение распределенной информационной системы для хранения данных известно на примере пиринговой сети BitTorrent и используемой в ней DHT (Distributed Hash Table) [14]. Идея состоит в том, что каждый диспетчер хранит часть общих данных системы, данные при этом физически распределены по всем диспетчерам ТРС, однако логически они являются одним целым. Для хранения данных в состав каждого диспетчера входит локальный сервер данных. Серверы всех диспетчеров управляются распределенной СУБД.

Применяемые в современных распределенных СУБД механизмы сегментирования и дублирования позволяют при отказе одной из ВУ восстановить часть ее данных, тем самым не допустив потери данных. Для этого все данные, хранящиеся в распределенной базе данных, разделяются на один или несколько сегментов. Разные сегменты помещаются на разные серверы, при этом для каждого сегмента (т.н. первичного) может быть создан один или несколько дублирующих сегментов. Дублирующие сегменты хранятся отдельно от первичных, на других серверах. Изменения данных сначала фиксируются в первичном сегменте, а затем во всех дублирующих сегментах. В случае неисправности одного из серверов, хранящего дублирующий сегмент, система продолжает корректно работать, при восстановлении неисправного сервера он заново получает актуальную версию дублирующего сегмента. Неисправность сервера, хранящего первичный сегмент, приведет к восстановлению одного из дублирующих сегментов в качестве первичного. Выбор нового первичного сегмента осуществляется путем голосования. После восстановления ранее отказавшего сервера с первичным сегментом и копирования на него актуальной версии данных он продолжает функционировать в качестве дублирующего сегмента.

4. Выбор распределенной СУБД в качестве основы информационной системы ТРС

Большинство существующих распределенных СУБД относятся к типу noSQL-решений, для которых характерна, прежде всего, нереляционная модель хранения данных. Вместо реляционной (табличной) модели представления и хранения данных в noSQL-решениях применяются документо-ориентированные, ассоциативные, столбцово-ориентрированные или векторные модели. Решения noSQL не поддерживают транзакции, ограничиваясь лишь поддержкой атомарности операций записи и чтений. Из-за нереляционной модели хранения и представления данных большинство существующих решений noSQL не предполагают использование языка запросов SQL. Вместо SQL используется собственный язык запросов, который в большинстве случаев лишен гибкости, свойственной SQL.

Для организации распределенной информационной системы авторами были рассмотрены следующие известные решения: ClickHouse [15], Elasticsearch [16], Redis [17].

Распределенная СУБД *Clickhouse* была разработана для решения аналитических задач, например, расчетов статистических показателей для большого объема данных и др. Особенностями области применения являются большой объем данных, высокая скорость их поступления, отсутствие изменений ранее записанных данных. В виду большого объема данных и выполнения в подавляющем большинстве только агрегационных операций, как правило, к хранению аналитических данных не предъявляются требования надежности и целостности.

Основой распределенной СУБД *Elasticsearch* являются серверы полнотекстового поиска Lucene. Все данные в Elasticsearch *хранятся* в индексе в полуструктурированном

виде – *документе*. Использование документной модели хранения и представления данных позволяют сделать решение универсальным. Использование встроенного языка запросов QyeryDSL совместно с инструментами полнотекстового поиска, предоставляемыми серверами Lucene, позволяют выполнять сложные запросы к хранящимся документам.

Распределенная СУБД *Redis* использует наиболее простую модель данных — ассоциативную. Все добавляемые в Redis данные ассоциируются с идентификатором, который в дальнейшем используется для выполнения запросов к хранящимся данным. Распределенная СУБД Redis часто используется для организации промежуточного хранения данных (кэширования данных) с целью снижения нагрузки на основное хранилище данных.

Результаты проведенного авторами сравнительного анализа существующих распределенных СУБД отражены в приведенной ниже табл.

Таблица

Сравнительный анализ распределенных СУБД

Критерий сравнения	Clickhouse	Elasticserach	Redis
Доступность информационной системы всем ВУ	+	+	+
Соответствие децентрализованной архитектуре ГСУР	+	+	+
Надежность сохранения глобальной очереди заданий	— асинхронное дублирование данных	+ синхронное дублирование данных	+ синхронное дублирование данных
Возможность изменения и извлечения заданий из очереди	— данные не изменяются, а удаление возможно только за месяц	+ данные могут быть изменены и удалены с использованием идентификатора	+ данные могут быть изменены и удалены с использованием идентификатора
Организация поиска данных по критериям (по приоритетам)	+ диалект SQL	+ QueryDSL + инструменты полнотекстового поиска	— обращение только по идентификатору

Табл. показывает, что наиболее полно всем предъявленным требованиям удовлетворяет распределенных СУБД Elasticsearch и используемая в ней документо-ориентированная модель хранения и представления данных. Каждый поступающий паспорт задания представляется отдельным документом, поля которого могут быть проанализированы при выполнении поискового запроса.

5. Макет ТРС с распределенной информационной системой

Для экспериментальной проверки применимости выбранных решений авторами был разработан модельный макет ТРС с распределенной информационной системой, основанной на распределенной СУБД Elasticsearch. Распространенной практикой создания подобных модельных макетов является использование в качестве моделируемых серверов (рабочих станций, вычислительных модулей) виртуальных машин. При этом каждая подсистема ВУ — управляющая, исполнительная, серверная и подсистема доступа, моделируется одной или несколькими виртуальными машинами. Большое число одновременно выполняющихся виртуальных машин ведет к интенсивному расходу ресурсов физического сервера, используемого для макета. Авторы при построении макета применили альтернативный подход — контейнерную виртуализацию с использованием платформы Docker [18], требующей значительно меньшего объема аппаратных ресурсов по сравнению с гипервизорной виртуализацией.

Использование контейнерной виртуализации предполагает, что каждая рабочая станция любой подсистемы ВУ запускается в безопасной изолированной среде – контейнере. Для создания контейнера подготавливается образ – шаблон инфраструктуры, настроенный к запуску определенного программного обеспечения (СУБД, ЛСУР и др.).

Помимо снижения накладных расходов использование контейнерной виртуализации при подготовке макета ТРС позволяет добиться высокой скорости запуска как отдельного контейнера, так и всей многоконтейнерной системы, а также осуществлять контроль состояния ее выполнения. Контроль состояния выполнения предполагает, что при отказе одного из контейнеров ядро Docker может перезапустить неисправный контейнер.

Подготовленный модельный макет ТРС состоит из трех вычислительных установок, на каждой из которых располагается один сервер Elasticsearch. База данных разделена на 6 сегментов, каждый сегмент имеет два дубликата: первичный и вторичный (рис. 2).

С помощью подготовленного макета авторы провели исследования надежности и производительности распределенной информационной системы, основанной на Elasticsearch.

Исследование надежности осуществлялось следующим образом. В информационную систему был направлен поток заданий, состоящий из $M = 5000$ заданий. Размер каждого задания не превышал 15 Кбайт. Интервалы времени между поступлением заданий в очередь имели *пуассоновское распределение* с интенсивностью $\lambda = 20$. На фоне обработки поступающего входного потока заявок из состава ТРС исключались отдельные ВУ вместе с серверами баз данных, используемыми для хранения глобальной очереди. Через установленный интервал времени, достаточный для восстановления дублирующих копий сегментов неисправного сервера на оставшихся серверах, ранее исключенная ВУ вновь добавлялась в состав ТРС. При этом на основе анализа служебных данных, хранящихся в базе данных, авторами контролировалось автоматическое размещение части сегментов с других серверов ВУ на серверах вновь добавленной ВУ. Операция добавления и исключения ВУ, имитирующая систематические отказы, повторялась постоянно на фоне поступающего потока заданий. Выбор исключаемой ВУ и моментов времени исключения ВУ из состава ТРС осуществлялся случайным образом. После окончания поступления заданий в информационную систему на одном из серверов ВУ выполнялся агрегирующий запрос, определяющий количество документов, хранящихся (добавленных в результате эксперимента) в базе данных. Результат выполнения запроса сравнивался с исходным

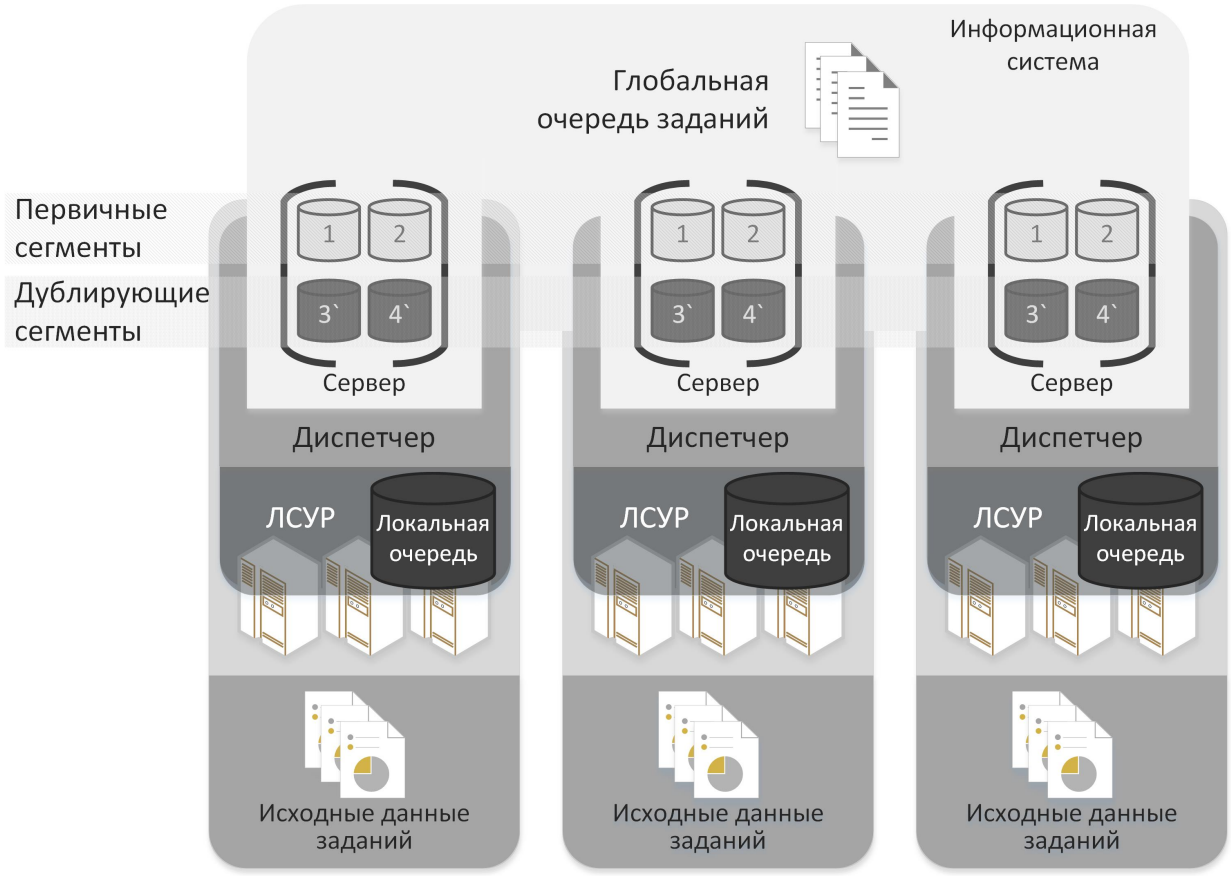

Рис. 2. Макет ТРС с использованием распределенной информационной системы на основе Elasticsearch

числом заданий (M). В результате проведения эксперимента авторами было установлено, что в конфигурации из трех серверов выход из строя одного из них не приводит к потере информации о заданиях глобальной очереди. Вывод из строя двух серверов приводит к потере информации о 20%-30% заданий. Дальнейшие эксперименты показали, что доля потерянных заданий может быть снижена за счет увеличения числа дубликатов каждого из сегментов с некоторым снижением производительности распределенной СУБД. Кроме этого, было обнаружено, что на надежности хранения данных отрицательно сказывается включение по умолчанию механизма кэширования поступающих в Elasticsearch данных. Все поступающие данные сохраняются в оперативной памяти сервера и не записываются на диск, пока объем поступивших данных не достигнет определенного значения, заданного в конфигурации сервера. Отказ сервера приводит к потере всех данных, не записанных на диск. Запрещение кэширования данных увеличивает надежность их хранения, но также приводит к некоторым потерям производительности.

Для оценки того, насколько существенными для информационной системы ТРС будут потери производительности при увеличении числа дубликатов и запрете кэширования, авторами был проанализирован опыт эксплуатации Распределенной инфраструктуры суперкомпьютерных приложений (РИСП) [1], объединяющей в ТРС вычислительные ресурсы филиалов МСЦ РАН нескольких городов России. Было установлено, что приемлемая скорость работы с очередью заданий составляет 10-15 документов (паспортов заданий) в секунду. Эксперимент по оценке производительности проводился в несколько этапов. На каждом из этапов эксперимента в информационную систему поступал поток

документов M = 1000, при этом интервалы времени между поступлением документов задавались согласно экспоненциальному распределению с интенсивностью (плотностью) $\lambda = [10, 50, 100]$ соответственно. В результате было установлено, что разработанная авторами распределенная информационная система способна обеспечить скорость 500-600 документов в секунду, что заведомо превышает необходимый уровень производительности.

Результаты проведенных экспериментов по оценке надежности и производительности разработанной информационной системы показали возможность ее использования в ТРС.

Заключение

При организации глобальной очереди заданий в территориально распределенной вычислительной системе неизбежно возникает противоречивый выбор между централизованной и децентрализованной схемами управления. С одной стороны, непрерывные изменения состава ТРС требуют децентрализованной схемы, как единственной способной обеспечить целостность распределенной системы. С другой стороны, глобальная очередь заданий является единой для всей ТРС, а абсолютные приоритеты заданий требуют оперативного распределения по вычислительным ресурсам системы. Для организации глобальной очереди необходима информационная система, являющаяся логическим центром ТРС.

Исследование, проведенное авторами в настоящей работе, показало, что информационная система, организованная на основе распределенной базы данных, обладает преимуществами централизованной и децентрализованной архитектур. В статье авторы рассмотрели распространенные модели распределенных СУБД и пришли к выводу, что для сценариев работы с глобальной очередью заданий в ТРС наиболее подходящей является документо-ориентированная модель хранения и представления данных, реализованная, в частности, в распределенной СУБД Elasticsearch. С целью экспериментальной проверки полученного вывода авторами был разработан макет ТРС, в рамках которого на базе СУБД Elasticsearch была реализована информационная система для формирования и хранения глобальной очереди заданий. Проведенные эксперименты показали как общую работоспособность системы, так и приемлемый уровень ее надежности и производительности.

Полученные авторами результаты относятся к начальному этапу построения ТРС суперкомпьютерных центров коллективного пользования. Для успешного завершения проекта авторы считают необходимым решение следующих научно-технических задач:

- исследование и разработка методов и алгоритмов оптимального планирования заданий в ТРС;
- разработка подсистемы информационной безопасности глобальной очереди заданий;
- разработка интерфейсов подключения к глобальной очереди заданий для различных локальных систем управления ресурсами.

В решении указанных задач авторы видят перспективы развития представленной работы.

Работа выполнена в рамках государственного задания с использованием вычислительных ресурсов Межведомственного суперкомпьютерного центра РАН (МСЦ РАН).

Литература

1. Савин Г.И., Шабанов Б.М., Корнеев В.В., Телегин П.Н., Семенов Д.В., Киселев А.В., Кузнецов А.В., Вдовикин О.И., Аладышев О.С., Овсянников А.П. Создание распределенной инфраструктуры для суперкомпьютерных приложений // Программные продукты и системы. 2008. № 2. С. 2–7.
2. Корнеев В.В., Семенов Д.В., Телегин П.Н., Шабанов Б.М. Отказоустойчивое децентрализованное управление ресурсами грид // Известия вузов. Электроника. 2015. Т. 20, № 1. С. 83–90.
3. Баранов А.В., Киселёв А.В., Старичков В.В., Ионин Р.П., Ляховец Д.С. Сравнение систем пакетной обработки с точки зрения организации промышленного счета. Научный сервис в сети Интернет: поиск новых решений // Труды Международной суперкомпьютерной конференции (Новороссийск, 17-22 сентября 2012 г.). М.: Изд-во МГУ, 2012. С. 506–508.
4. Баранов А.В., Тихомиров А.И. Планирование заданий в территориально распределенной системе с абсолютными приоритетами //Вычислительные технологии. 2017. Т. 22, № S1. С. 4–12.
5. Березовский П.С., Коваленко В.Н. Состав и функции системы диспетчеризации заданий в грид с некластеризованными ресурсами // Препринты ИПМ им. М. В. Келдыша. 2007. № 67. С. 29.
6. WMS Architecture overview. URL: `http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wms.shtml` (дата обращения: 27.03.2017).
7. Internal Architecture 5.14. URL: `http://www.gridway.org/doku.php?id=documentation:release_5.14:iashtml` (дата обращения: 20.03.2017).
8. Cirne W., Brasileiro F., Costa L., Paranhos D., Santos-Neto E., Andrade N. Scheduling in Bag-of-Task Grids: PAUA Case // 16th Symposium on Computer Architecture and High Performance Computing. Oct. 2004. pp. 124–131. DOI: 10.1109/CAHPC.2004.37.
9. Коваленко В.Н., Орлов А.В. Управление заданиями в распределенной среде и протокол резервирования ресурсов// Препринты ИПМ им. М. В. Келдыша. 2002. №1. С. 1–25.
10. Buncic P., Saiz P., Peters A.J. The AliEn System, Status and Perspectives // 2003 Conference for Computing in High-Energy and Nuclear Physics, La Jolla, CA, USA, 24-28 Mar 2003. P. MOAT004. URL: `http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOAT004.PDF` (дата обращения: 20.03.2017).
11. Топорков В.В., Емельянов Д.М., Потехин П.А. Формирование и планирование пакетов заданий в распределенных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2015. Т. 4, № 2. С. 44–57. DOI: 10.14529/cmse150204.
12. Валиев М.К., Китаев Е.Л., Слепенков М.И. Служба директорий LDAP как инструментальное средство для создания распределенных информационных систем // Препринты ИПМ им. М.В. Келдыша. 2000. № 23. С. 1–22.
13. Kesselman C., Fitzgerald S., Foster I., Tuecke S., Smith W. A Directory Service for Configuring High-Performance Distributed Computations // 6th IEEE Symposium on High Performance Distributed Computing. 1997. pp. 365–375. DOI: 10.1109/HPDC.1997.626445.

14. Loewenstern A. Norberg A. DHT Protocol. 2008. URL: http://bittorrent.org/beps/bep_0005.html (дата обращения: 11.03.2017).

15. ClickHouse Reference Manual. 2015. URL: https://clickhouse.yandex/reference_en.html (дата обращения: 16.02.2017).

16. Elastic Stack and Product Documentation. 2016. URL: https://www.elastic.co/guide/index.html (дата обращения: 22.01.2017).

17. Programming with Redis. 2016. URL: https://redis.io/documentation (дата обращения: 12.02.2017).

18. A. Prasad. Announcing Docker Compose. 2015. URL: https://blog.docker.com/2015/02/announcing-docker-compose/(дата обращения: 26.02.2017).

Баранов Антон Викторович, к.т.н., доц., в.н.с., Межведомственный суперкомпьютерный центр Российской академии наук – филиал Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук» (Москва, Российская Федерация)

Тихомиров Артем Игоревич, стажер-исследователь, Межведомственный суперкомпьютерный центр Российской академии наук – филиал Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук» (Москва, Российская Федерация)

DOI: 10.14529/cmse170403

METHODS AND TOOLS FOR ORGANIZING THE GLOBAL JOB QUEUE IN THE GEOGRAPHICALLY DISTRIBUTED COMPUTING SYSTEM

© 2017 A.V. Baranov, A.I. Tikhomirov

Joint Supercomputer Center of the Russian Academy of Sciences - Branch of Federal State Institution «Scientific Research Institute for System Analysis of the Russian Academy of Sciences», (pr. Leninsky 32a, Moscow, 119334 Russia)

E-mail: tema4277@rambler.ru

Received: 01.09.2017

The geographically distributed computing infrastructure (DCI) considered in the paper includes high performance computing systems united by communication channels. Computing systems from the DCI are high-performance clusters differing in architecture and performance. Communication channels uniting clusters have different reliability and bandwidth. The considered model of DCI has a decentralized jobs management and dispatching scheme. This scheme implies that at any time malfunction of any computing cluster or a failure in the communication channel can cause cluster's leaving the DCI. Cluster's or channel's troubleshooting means dynamically connecting the cluster to the DCI. The global job queue is organized in this computing infrastructure. Computing jobs have absolute priorities, and high priority job can interrupt low priority running jobs. Jobs from the global queue allocate on idle resources of computing systems. Forming and storing global job queue in conditions of dynamically changing DCI composition needs the reliable information system. The authors reviewed some distributed DBMSs as the basis of this information system. The article outlines the requirements for a distributed information system. The authors conducted a comparative analysis and selected a solution that

satisfies the requirements, and designed prototype of the geographically distributed computing infrastructure with the decentralized scheme of jobs dispatching.

Keywords: grid, information system, absolute priorities.

FOR CITATION

Baranov A.V., Tikhomirov A.I. Methods and Tools for Organizing the Global Job Queue in the Geographically Distributed Computing System. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 4. pp. 28–42. DOI: 10.14529/cmse170403.

References

1. Savin G.I., Shabanov B.M., Korneev V.V., Telegin P.N., Semenov D.V., Kiselev A.V., Kuznecov A.V., Vdovikin O.I., Aladyshev O.S., Ovsjannikov A.P. Creation of Distributed Infrastructure for Supercomputer Applications. *Programmnye produkty i sistemy* [Software & Systems]. 2008, no. 2, pp. 2–7. (in Russian).

2. Korneev V.V., Semenov D.V., Telegin P.N., Shabanov B.M. Resilient Decentralized GRID Resources Control. *Izvestija vysshih uchebnyh zavedenij. Jelektronika* [Proceedings of Universities. Electronics]. 2015, vol. 20, no. 1, pp. 83–90. (in Russian).

3. Baranov A.V., Kiselev A.V., Starichkov V.V., Ionin R.P., Lyakhovets D.S. Comparison of Workload Management Systems from the Point of View of Organizing an Industrial Computing. *Nauchnyj servis v seti Internet: poisk novyh reshenij: Trudy mezhdunarodnoy superkomp'yuternoy konferentsii (Novorossiysk, 17–22 Sentyabrya 2012)* [Scientific Services and Internet: Search for New Solutions: Proceedings of the International Supercomputing Conference (Novorossiysk, Russia, September, 17-22, 2012)]. Moscow, Publishing of Lomonosov Moscow State University, 2012, pp. 506–508. (In Russian).

4. Baranov A.V., Tihomirov A.I. Scheduling of Jobs in a Territorially Distributed Computing System with Absolute Priorities. *Vychislitel'nye tehnologii* [Computational Technologies]. 2017, vol. 22, no. S1, pp. 4–12. (in Russian).

5. Berezovskij P.S., Kovalenko V.N. Structure and Functionality of the Job Management System for Grid with Non-Clustered Resources. *Preprinty IPM im. M. V. Keldysha* [KIAM Preprints]. 2007, no. 67, pp. 1–29. (in Russian).

6. WMS Architecture overview. Available at: `http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wms.shtml` (accessed: 27.03.2017).

7. Internal Architecture 5.14. Available at: `http://www.gridway.org/doku.php?id=documentation:release_5.14:iashtml` (accessed: 20.03.2017).

8. Cirne W., Brasileiro F., Costa L., Paranhos D., Santos-Neto E., Andrade N. Scheduling in Bag-of-Task Grids: PAUA Case. 16th Symposium on Computer Architecture and High Performance Computing. Oct. 2004, pp. 124–131. DOI: 10.1109/CAHPC.2004.37.

9. Kovalenko V.N., Orlov A.V. Metascheduling in GRID and Resource Reservation Protocol. *Preprinty IPM im. M. V. Keldysha* [KIAM Preprints]. 2002, no. 1, pp. 1–25. (in Russian).

10. Buncic P., Saiz P., Peters A.J. The AliEn System, Status and Perspectives. 2003 Conference for Computing in High-Energy and Nuclear Physics, La Jolla, CA, USA, 24–28 Mar 2003. Available at: http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOAT004.PDF (accessed: 20.03.2017).

11. Toporkov V.V., Emel'janov D.M., Potehin P.A. Job Batch Generation and Scheduling in Distributed Computing Environments. *Vestnik Yuzho-Uralskogo gosudarstvennogo universiteta. Seriya: Vychislitel'naja matematika i informatika* [Bulletin of South Ural State University. Series: Computational Mathematics and Software Engineering]. 2015, vol. 4, no. 2. pp. 44–57. DOI: 10.14529/cmse150204 (in Russian).

12. Valiev M.K., Kitaev E.L., Slepenkov M.I. LDAP Directory Service as a Tool for Implementation of Distributed Information Systems. *Preprinty IPM im. M. V. Keldysha* [KIAM Preprints]. 2000. no. 23. pp. 1–22. (in Russian).

13. Kesselman C., Fitzgerald S., Foster I., Tuecke S., Smith W. A Directory Service for Configuring High-Performance Distributed Computations. 6th IEEE Symposium on High Performance Distributed Computing. 1997. pp. 365–375. DOI: 10.1109/HPDC.1997.626445.

14. Loewenstern A. Norberg A. DHT Protocol. 2008. Available at: `http://bittorrent.org/beps/bep_0005.html` (accessed: 11.03.2017).

15. ClickHouse Reference Manual. 2015. Available at: `https://clickhouse.yandex/reference_en.html` (accessed: 16.02.2017).

16. Elastic Stack and Product Documentation. 2016. Available at: `https://www.elastic.co/guide/index.html` (accessed: 22.01.2017).

17. Programming with Redis. 2016. Available at: `https://redis.io/documentation` (accessed: 12.02.2017).

18. Prasad A. Announcing Docker Compose. 2015. Available at: `https://blog.docker.com/2015/02/announcing-docker-compose/` (accessed: 26.02.2017).