Модел. и анализ информ. систем. Т. **21**, № **4** (2014) 13–24

УДК 004.652

Применение стохастических метаэвристик в задаче управления данными в мультиклиентском кластере баз данных

Бойцов Е. А.

Ярославский государственный университет им. П. Г. Демидова
150000 Россия, г. Ярославль, ул. Советская, 14

e-mail: boytsovea@yandex.ru

получена 9 сентября 2014

Ключевые слова: базы данных, SaaS, мультиклиентская архитектура, стохастическая оптимизация

Мультиклиентский кластер баз данных – это концепция хранилища данных для облачных приложений с мультиклиентской архитектурой. Кластер представляет собой набор серверов реляционных баз данных с единой точкой входа, объединенных в одно целое и работающих под управлением контроллера кластера. Данная система нацелена на использование приложениями, разрабатываемыми в соответствии с парадигмой Software as a Service (SaaS), и позволяет разместить на предоставленных серверах данные большого количества клиентов таким образом, чтобы обеспечить их изоляцию, резервирование и наиболее эффективное использование предоставленных вычислительных мощностей. Одной из наиболее важных задач при разработке системы подобного рода является эффективное распределение данных по серверам, которое определяет степень загруженности отдельных узлов, а также устойчивость системы к сбоям. В работе рассматривается подход к управлению данными, основанный на применении функции оценки эффективности балансировки нагрузки. Данная функция применяется как при первичном размещении клиентов, так и для оптимизации уже имеющегося распределения клиентов. Оптимизация размещения ведется по стандартным схемам стохастических метаэвристик: имитации отжига (simulated annealing) и поиска с запретами (tabu search).

Введение

Разработка приложения в соответствии с парадигмой Software as a Service (SaaS) привносит необходимость решать множество задач, не свойственных приложениям с традиционной архитектурой. Основными из них являются обеспечение высокой степени горизонтальной масштабируемости приложения и снижение издержек компании-провайдера на обеспечение его функционирования при обеспечении соглашений об уровне обслуживания (service level agreements, SLA). Данные требования,

в свою очередь, предполагают применение таких архитектурных решений, которые позволили бы использовать одни и те же вычислительные мощности для обслуживания множества компаний-клиентов. Одним из таких решений является использование мультиклиентской (multi-tenant) архитектуры. Данный подход предполагает, что основные ресурсы приложения совместно используются многими компаниями-арендаторами, а за изоляцию их данных отвечает программная логика приложения. Проектирование решения в соответствии с подобными архитектурными принципами позволяет, при наличии достаточных вычислительных мощностей, поставить для обслуживания клиентов практически неограниченное количество экземпляров приложения. Однако указанные соображения не распространяются на серверы БД, так как данный компонент системы плохо масштабируется горизонтально. Несмотря на то, что в настоящее время в наиболее продвинутых РСУБД имеются встроенные средства для горизонтального масштабирования, их применение сопряжено либо с дополнительными сложностями и ограничениями функциональности, либо с большими финансовыми затратами.

Однако, если более внимательно проанализировать требования типичного облачного приложения с мультиклиентской архитектурой к СУБД, то можно выделить ряд особенностей, которые могут упростить разработку подсистемы хранения данных. Эти особенности таковы:

- основу клиентской базы облачного приложения составляют небольшие компании, не готовые нести издержки на собственную IT-инфраструктуру [1];
- данные клиентов облачного приложения должны быть полностью изолированы друг от друга, как будто хранятся в разных базах [2].

Таким образом, рассматриваемое нами приложение имеет дело с потоком запросов, который можно разделить на N независимых и не пересекающихся подпотоков от каждого клиента компании. Так как клиенты – это мелкие и средние предприятия, то они не генерируют вычислительно трудоемких запросов, однако общее количество запросов велико. На основе этих соображений была предложена концепция мультиклиентского кластера баз данных – системы для управления данными облачного приложения. Общая идея предлагаемого подхода к организации кластера состоит в том, чтобы реализовать дополнительный слой абстракции, предоставляющий программный интерфейс, который был бы максимально близок к интерфейсу традиционных СУБД. Основные преимущества такого решения:

- упрощение разработки приложений, связанное с возможностью использования разработчиками уже имеющихся знаний без необходимости заботиться об изоляции данных клиентов друг от друга;
- упрощение администрирования системы.

Основными задачами мультиклиентского кластера станут [3]:

- обеспечение изоляции данных клиентов друг от друга;
- управление размещением клиентских данных на серверах БД, динамическое перераспределение данных в зависимости от загрузки отдельных серверов и характера активности клиентов во времени;

- обеспечение отказоустойчивости в случае сбоя одного или нескольких серверов;
- маршрутизация запросов серверов приложений на соответствующий сервер БД;
- оценка эффективности использования ресурсов и диагностика состояния системы.

Схематично архитектура кластера представлена на рисунке 1. Более подробно с концепцией мультиклиентского кластера баз данных можно ознакомиться в [3].

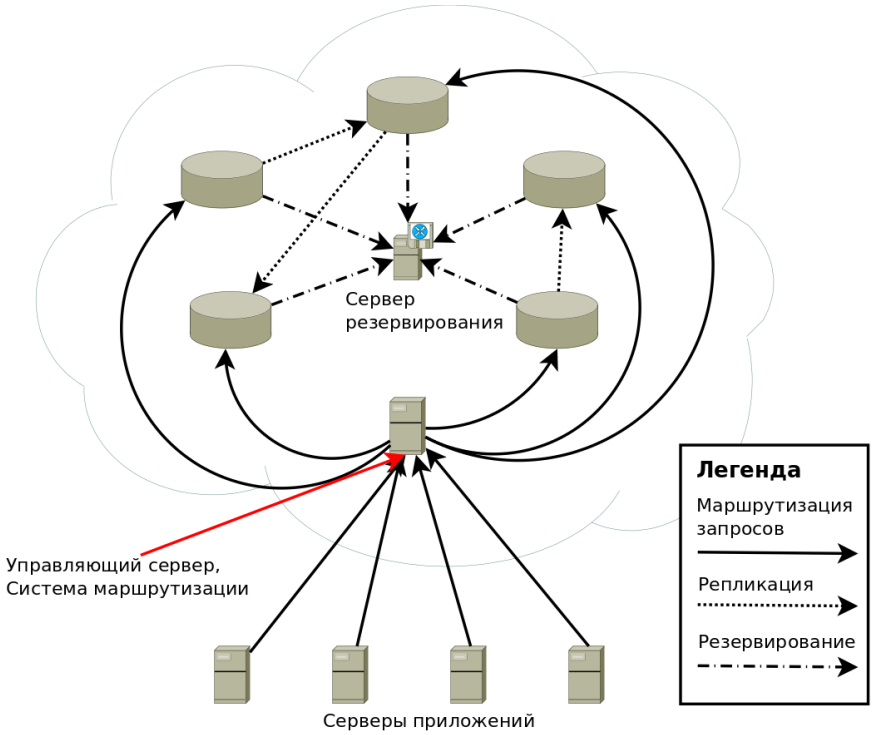

Рис. 1. Архитектура мультиклиентского кластера баз данных

1. Задача управления данными в мультиклиентском кластере

Одной из основных задач мультиклиентского кластера баз данных является управление размещением данных клиентов. От успешности ее решения зависят качество балансировки нагрузки, эффективность использования предоставленных аппаратных ресурсов, а также устойчивость системы к сбоям отдельных серверов в составе кластера. Для решения этой задачи необходимо разработать формальную модель системы и ввести метрику оценки эффективности работы подсистемы размещения данных, которая бы хорошо соотносилась с потребительскими характеристиками кластера (например, время обработки запроса кластером). На основе экспериментов на имитационной модели кластера [4] была предложена следующая метрика

эффективности балансировки нагрузки [5] для кластера из M серверов, обслуживающего N клиентов:

$$f = \sum_{i=1}^{M} \left(\frac{\sum_{j=1}^{N} load(i,j)}{\sum_{j=1}^{N} \lambda_j} - \frac{\bar{\lambda}_i}{\sum_{i=1}^{M} \bar{\lambda}_i} \right)^2, \quad (1)$$

где

- λ_j – нагрузка на кластер от потока запросов j-го клиента;
- $load(i,j)$ – нагрузка на i-й сервер от потока запросов j-го клиента (с учетом репликации данных в кластере);
- $\bar{\lambda}_i$ – мощность i-го сервера БД.

Говоря неформально, данная функция устроена таким образом, чтобы максимально приблизить нагрузку на каждый конкретный сервер в составе кластера к его доле в вычислительной мощности системы. Задача оптимального распределения данных в этом случае сводится к поиску такого состояния системы, при котором значение функции (1) является минимальным среди всех допустимых. Подробнее с формальной постановкой задачи балансировки нагрузки в мультиклиентском кластере баз данных с постоянными интенсивностями входящих потоков запросов можно ознакомиться в [6].

В самой задаче динамического управления данными можно выделить два аспекта:

- первичное размещение данных новых клиентов;
- перераспределение данных в кластере для оптимизации балансировки нагрузки на основе накопленных данных.

Первый аспект (первичное размещение данных), очевидно, является в большей степени областью применения эвристических инженерных приемов. Так как система еще не располагает данными о характере нагрузки от клиента, то и принять обоснованное решение о размещении его данных на том или ином узле она не в состоянии. На данном этапе можно обеспечить лишь устойчивость копии данных клиента к сбоям, а также зарезервировать для нее некоторые вычислительные мощности, основываясь на предположениях о характере будущей нагрузки.

Для задачи перераспределения данных уже можно применять подходы, основанные на формальной модели системы. В этом случае задача может быть сформулирована следующим образом: перевести кластер в состояние с меньшим значением целевой функции (1), не парализовав при этом его работу и не создавая избыточной нагрузки на систему. При ее решении следует принимать во внимание следующее соображение: нагрузка на кластер от конкретного клиента не является постоянной величиной и может существенно меняться с течением времени. Кроме того, в кластер постоянно добавляются новые клиенты, а какие-то клиенты наоборот снижают интенсивность использования приложения.

2. Применение стохастических метаэвристик в задаче управления данными

При более подробном рассмотрении метрики (1) оказывается, что задача минимизации данной метрики является частным случаем обобщенной квадратичной задачи о назначениях (Generalized Quadratic Assignment Problem, GQAP), которая, в свою очередь, является обобщением квадратичной задачи о назначениях (Quadratic Assignment Problem, QAP), впервые сформулированной в 1957 году Koopmans и Beckmann [7]. В их работе эта задача возникла как задача о размещении n фабрик в n городах с условием минимизации издержек, описываемых квадратичной функцией от комбинации расстояний между городами и объема перевозок между ними. Обобщенная квадратичная задача о назначениях впервые появляется в работе Lee и Ma [8]. Ее отличие состоит в том, что снимается ограничение о необходимости размещения по одной фабрике в каждом городе (то есть число городов и фабрик, вообще говоря, различно), однако добавляется ограничение на "вместимость" каждого города. Решением задачи по-прежнему является такое размещение фабрик, при котором достигается наименьшая стоимость взаимодействия и перевозки грузов между ними.

Известно, что QAP является NP-трудной [9] и на практике даже задачи небольшого размера до сих пор считаются очень сложными. Более подробно о QAP можно узнать в [10] и [11]. Библиографию по данной тематике можно найти в [12]. Так как GQAP является обобщением QAP, то она также является NP-трудной и еще более трудноразрешимой.

Если применить эти знания к задаче балансировки нагрузки на мультиклиентский кластер баз данных с метрикой (1), то станет очевидно, что решить ее точно не получится. Действительно, в данном случае мы будем иметь дело с десятками (а возможно, и сотнями) серверов БД в составе кластера и несколькими сотнями тысяч клиентов системы. Таким образом можно заключить, что для поиска оптимальной стратегии размещения данных необходимо использовать быстрые эвристические методы.

Одним из общеизвестных подходов для построения производительных оптимизационных эвристик является использование так называемых метаэвристик. Среди наиболее известных метаэвристик можно упомянуть:

- генетические алгоритмы (genetic algorithms [13]);
- имитацию отжига (simulated annealing [14]);
- перебор с запретами (tabu search [15]).

Применение генетических алгоритмов в задаче управления размещением данных весьма затруднительно, так как в данном случае сложно понять, что взять за "начальную популяцию" системы, ведь оптимизация всегда стартует из одной точки – текущего распределения клиентов в кластере. На основе двух оставшихся метаэвристик и имитационной модели кластера было разработано несколько вариантов алгоритмов управления данными.

2.1. Алгоритмы на основе имитации отжига

В рамках данной работы было предложено два варианта алгоритма управления данными на основе имитации отжига. Оба они работали по одной схеме и отличались лишь характером выполняемых над кластером преобразований: первый вариант алгоритма использовал полностью случайные преобразования (предполагалось, что он будет работать быстрее), второй же был запрограммирован искать такие преобразования, которые с большей вероятностью должны были перевести кластер в состояние с меньшим значением метрики. Оба алгоритма использовали три типа элементарных преобразований размещения данных в кластере:

1. Перемещение реплики данных клиента с одного сервера на другой. Параметрами преобразования являются номер клиента, а также номера сервера-источника и сервера-приемника.

2. Перемещение мастер-копии данных клиента с одного сервера на другой. Параметрами преобразования являются номер клиента, а также номера сервера-источника и сервера-приемника.

3. Обмен местами двух экземпляров данных двух различных клиентов. Параметрами преобразования являются номера клиентов, между которыми идет обмен, а также номера серверов, на которых расположены обмениваемые копии данных.

У каждого из элементарных преобразований задавалась некоторая вероятность его выбора на очередном шаге работы алгоритма. Первый вариант алгоритма выбирает параметры преобразований случайным образом из множества всех допустимых. Второй вариант алгоритма при выборе параметров преобразований анализирует текущую загруженность серверов и пытается перенести часть нагрузки с тех серверов, загруженность которых превышает их долю в вычислительной мощности кластера, на те серверы, чья нагрузка ниже их доли.

На все преобразования выделялся бюджет – некоторый процент от суммарного объема данных, хранимого кластером. В проводимых экспериментах использовались небольшие значения данного процента (0.1 – 1). Бюджет используется как ограничение на итерацию шага оптимизации: если стоимость преобразования из текущего состояния кластера в наилучшее из найденных на данной итерации оптимизации превышает бюджет, то итерация считается завершенной. Начальная температура системы в алгоритме имитации отжига полагается равной единице. За стоимость выполнения одного элементарного преобразования полагается отношение размера перемещаемых данных к m бюджетам итерации (m выбирается экспериментально). Таким образом, на каждом шаге производятся следующие действия:

1) согласно назначенным весам выбирается одно из элементарных преобразований и его параметры, вычисляется стоимость данного преобразования;

2) вычисляется метрика кластера после выполнения выбранного преобразования;

3) если полученное значение метрики меньше текущего, то система переходит в новое состояние, ее температура уменьшается на стоимость преобразования;

4) если полученное значение метрики не меньше текущего, то система переходит в новое состояние с вероятностью, зависящей от текущей температуры системы. Если преобразование принимается, то температура системы уменьшается в соответствии со стоимостью преобразования;

5) если несколько последовательно выбранных элементарных преобразований не привели к изменению состояния системы (всегда находились варианты хуже текущего, а температура системы уже слишком мала, чтобы допустить переход в них), то итерация завершалась.

Результатом работы итерации оптимизации является распределение данных клиентов с наименьшим значением метрики (не обязательно то состояние, в котором алгоритм завершил свою работу).

2.2. Алгоритмы на основе перебора с запретами

Похожие варианты алгоритмов управления данными были реализованы на основе метаэвристики "перебор с запретами". Применялись те же элементарные преобразования, что и в случае с имитацией отжига. Общая схема работы алгоритма такова:

1) для текущего положения системы находится N "соседей" при помощи применения к ней элементарных преобразований;

2) в рассмотренной окрестности находится состояние с наименьшим значением метрики, и если в нем значение метрики меньше текущего, то оно запоминается как текущий глобальный минимум;

3) система переходит в новое состояние, а старое вносится в список запрещенных.

Алгоритм прекращает свою работу в случае исчерпания бюджета итерации оптимизации, т.е. когда для достижения найденного на данном этапе решения из текущего состояния кластера необходимо переместить больше данных, чем это разрешено сделать. Как и в предыдущем случае, было разработано два варианта алгоритма: первый рассматривал полностью случайную окрестность текущего решения, а второй был запрограммирован так, чтобы оптимизировать перебор.

2.3. Общие черты разработанных алгоритмов

Разработанные алгоритмы имеют ряд общих черт, делающих их привлекательными для решения задачи управления данными. Среди них:

1) простота реализации и восприятия;

2) хорошая производительность – так как алгоритмы основаны на трех элементарных преобразованиях распределения клиентов, то они работают достаточно быстро;

3) адаптивность по отношению к изменениям в структуре кластера (добавление/удаление серверов) – новые вычислительные мощности немедленно задействуются, удаляемые – оперативно освобождаются;

4) отсутствие больших накладных расходов на перемещение данных клиентов ввиду наличия ограничения бюджетом итерации;

5) возможность итеративного применения оптимизации.

Все указанные характеристики делают подход, основанный на применении стохастических метаэвристик, весьма перспективным.

3. Экспериментальная оценка алгоритмов

Для экспериментальной оценки разработанных алгоритмов было проведено две серии экспериментов на имитационной модели мультиклиентского кластера баз данных.

В первой серии экспериментов разработанные алгоритмы сравнивались с созданными ранее неоптимизирующими алгоритмами распределения клиентов [6]. Рассматривалось шесть алгоритмов:

1) алгоритм, основанный на разделении кластера на группы серверов и балансировке клиентов между группами (SG);

2) неоптимизирующий алгоритм, основанный на минимизации метрики (1) с помощью вариации метода ветвей и границ (NOM);

3) алгоритм, использующий имитацию отжига со случайным выбором элементарных преобразований на каждом шаге (OMSAR);

4) алгоритм, использующий имитацию отжига с оптимизированным выбором элементарных преобразований на каждом шаге (OMSAD);

5) алгоритм, использующий перебор с запретами со случайным выбором элементарных преобразований на каждом шаге (OMTSR);

6) алгоритм, использующий перебор с запретами с оптимизированным выбором элементарных преобразований на каждом шаге (OMTSD);

В ходе эксперимента модель кластера, состоящего из 16 серверов, была настроена таким образом, чтобы обеспечивать постоянное добавление новых клиентов. Так как вычислительная мощность кластера не менялась, то в какой-то момент времени он должен был перестать справляться с потоком запросов и начать копить очередь. В каждом случае моделирование велось до момента образования на кластере совокупной очереди длиной в тысячу запросов либо на отдельном сервере длиной в две сотни. Интенсивности входящих потоков запросов от клиентов в этой серии экспериментов оставались постоянными на протяжении всего моделирования. В целях обеспечения отказоустойчивости для каждого клиента должны были быть созданы две реплики его данных. Дополнительным параметром эксперимента было соотношение запросов только на чтение и модифицирующих запросов в общем потоке. Для каждой комбинации (алгоритм, соотношение числа запросов на чтение/запись) проводилось по 30 экспериментов. Результаты этой серии экспериментов приведены в таблице 1.

Таблица 1. Результаты первой серии экспериментов

Номер алгоритма	Соотношение запросов чтение/запись	Среднее кол-во размещенных клиентов
1 (SG)	70/30	682
1 (SG)	50/50	610
1 (SG)	30/70	448
2 (NOM)	70/30	724
2 (NOM)	50/50	723
2 (NOM)	30/70	666
3 (OMSAR)	70/30	722
3 (OMSAR)	50/50	719
3 (OMSAR)	30/70	666
4 (OMSAD)	70/30	725
4 (OMSAD)	50/50	720
4 (OMSAD)	30/70	668
5 (OMTSR)	70/30	724
5 (OMTSR)	50/50	723
5 (OMTSR)	30/70	665
6 (OMTSD)	70/30	724
6 (OMTSD)	50/50	717
6 (OMTSD)	30/70	662

На основании полученных результатов можно сделать вывод, что применение оптимизации при постоянных интенсивностях входящих потоков запросов не оказало сколь бы то ни было существенного эффекта в сравнении с неоптимизирующей версией этого же алгоритма (хотя в обоих случаях результаты были примерно на 6–18% лучше, чем результаты алгоритма 1(SG), не основанного на метрике эффективности балансировки нагрузки). Подобные результаты объясняются самой схемой эксперимента: за счет постоянного добавления новых клиентов неоптимизирующий алгоритм в какой-то степени все-таки проводил оптимизацию распределения нагрузки на серверы в составе кластера. Неожиданным оказался тот факт, что алгоритмы номер 3(OMASR) и 5(OMTSR) со случайным выбором элементарных преобразований работали гораздо дольше своих "направленных" вариаций. Так как показанные ими при работе результаты были несколько хуже, то дальнейшие эксперименты с их использованием не проводились. Также было выяснено, что при примерно аналогичных результатах алгоритм, основанный на имитации отжига, демонстрирует лучшую производительность, чем алгоритм, основанный на схеме перебора с запретами.

Вторая серия экспериментов проводилась по несколько измененной схеме:

- интенсивности потоков входящих запросов от клиентов менялись с течением времени случайным образом;
- количество клиентов, регистрируемых на кластере, было ограничено значениями, близкими к полученным в первой серии экспериментов критическим параметрам;

- на основе первой серии экспериментов было добавлено ограничение на время моделирования в 2500 единиц модельного времени (среднее время моделирования в первой серии экспериментов для алгоритмов на основе метрики (1) составило чуть больше 1400 единиц модельного времени).

Целью данной серии экспериментов было выяснить, как ведут себя тестируемые алгоритмы при нагрузке, близкой к максимально допустимой (с точки зрения данного алгоритма). Наилучшим результатом здесь является прекращение моделирования в результате срабатывания триггера прерывания по времени. Результаты экспериментов приведены в таблице 2.

Таблица 2. Результаты второй серии экспериментов

Номер алгоритма	Соотношение запросов чтение/запись	Среднее кол-во размещенных клиентов	Среднее время моделирования
2 (NOM)	70/30	700	1721
2 (NOM)	50/50	700	1703
2 (NOM)	30/70	650	1674
4 (OMSAD)	70/30	700	2500
4 (OMSAD)	50/50	700	2500
4 (OMSAD)	30/70	650	2500
6 (OMTSD)	70/30	700	2500
6 (OMTSD)	50/50	700	2500
6 (OMTSD)	30/70	650	2500

При анализе полученных результатов становится очевидно, что во второй серии экспериментов оптимизирующие алгоритмы показали себя лучше своих неоптимизирующих версий. Во всех экспериментах кластер с помощью примененных оптимизаций был приведен в такое состояние, что мог и дальше успешно продолжать обслуживать поступающие запросы, постоянно адаптируясь под изменения в потоке запросов. Не оптимизирующие же алгоритмы во всех случаях становились "жертвами" флуктуаций в интенсивностях потоков запросов и рано или поздно приводили отдельные части системы в состояние, в котором она не справляется с имеющейся нагрузкой.

4. Выводы и дальнейшие направления исследования

Результаты экспериментов показали, что разработанный подход к управлению данными, основанный на применении метрики качества балансировки нагрузки и оптимизации при помощи стохастических метаэвристик, имеет право на существование и успешно справляется с основными задачами, возлагаемыми на систему управления данными мультиклиентского кластера баз данных. Разработанные в рамках данного этапа исследования алгоритмы являются простыми и производительными, что очень важно для их успешной эксплуатации и сопровождения. Наилучшие показатели по совокупности получаемых результатов и скорости работы показал алгоритм,

основанный на метаэвристике "имитация отжига". Вместе с тем, сохраняют свою актуальность следующие вопросы:

- каким образом в реальном окружении определить интенсивность потока запросов конкретного клиента;
- какой алгоритм следует использовать для выполнения переноса данных клиента, если подсистема управления кластера сочтет это необходимым.

Для их решения потребуется дальнейшая доработка имитационной модели кластера, поскольку в данный момент модель абстрагирована от указанных деталей. В этом вопросе могут быть востребованы результаты исследований по сходной тематике, например [16].

Список литературы

1. Chong F., Carraro G. Architecture strategies for catching the long tail. — 2006. — URL: http://msdn.microsoft.com/en-us/library/aa479069.aspx.
2. Chong F., Carraro G., Wolter R. Multi-tenant data architecture. — 2006. — URL: http://msdn.microsoft.com/en-us/library/aa479086.aspx.
3. Boytsov E., Sokolov V. The problem of creating multi-tenant database clusters // Proceedings of SYRCoSE. — Perm, 2012. — P. 172–177.
4. Boytsov E. Designing and development of the imitation model of a multi-tenant database cluster // Modeling and Analysis of Information Systems. — 2013. — Vol. 20, no. 4. — P. 136–149.
5. Boytsov E., Sokolov V. The formal statement of the load-balancing problem for a multi-tenant database cluster with a constant flow of queries // Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering. — Kazan, 2013. — P. 117–12.
6. Boytsov E., Sokolov V. Comparison of data management strategies for multi-tenant database cluster // Proceedings of the International Symposium on Business Modelling and Software Design. — Luxembourg, 2014. — P. 217–222.
7. Beckman M., Koopmans T. Assignment problems and the location of economic activities // Econometrica. — 1957. — Vol. 25. — P. 53–76.
8. The generalized quadratic assignment problem : Rep. / University of Toronto, Department of Mechanical and Industrial Engineering ; Executor: C.-G. Lee, Z. Ma. — Toronto, Canada : 2004.
9. Sahni S., Gonzalez T. P-complete approximation problems. // Journal of ACM. — 1976. — Vol. 23, no. 3. — P. 555–565.
10. Burkard R. Locations with spatial interactions: The quadratic assignment problem. // Discrete location theory. — 1991. — P. 387–437.
11. Rendl F., Pardalos P., Wolkowicz H. The quadratic assignment problem: A survey and recent developments // Proceedings of the DIMACS Workshop on Quadratic Assignment Problems. — Vol. 16. — American Mathematical Society, 1994. — P. 1–42.

12. Burkard R., Cela E. Quadratic and three-dimensional assignment problems // Annotated Bibliographies in Combinatorial Optimization. — Chichester : Wiley, 1997. — P. 373–392.

13. Holland J. H. Adaptation in Natural and Artificial Systems. — Cambridge : MIT Press, 1992.

14. Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by simulated annealing // SCIENCE. — 1983. — Vol. 220, no. 4598. — P. 671–680.

15. Glover F. Future paths for integer programming and links to artificial intelligence // Computers and Operation Research. — 1986. — Vol. 13, no. 5. — P. 533–549.

16. Elmore A., Das S., Agrawal D., El Abbadi A. Zephyr: Live migration in shared nothing databases for elastic cloud platforms // Proceedings of the ACM SIGMOD International Conference on Management of Data. — New York, 2011. — P. 301–312.

Applying Stochastic Metaheuristics to the Problem of Data Management in a Multi-Tenant Database Cluster

Boytsov E. A.

P.G. Demidov Yaroslavl State University, Sovetskaya str., 14, Yaroslavl, 150000, Russia

Keywords: databases, SaaS, multi-tenant architecture, stochastic optimization

A multi-tenant database cluster is a concept of a data-storage subsystem for cloud applications with the multi-tenant architecture. The cluster is a set of relational database servers with the single entry point, combined into one unit with a cluster controller. This system is aimed to be used by applications developed according to Software as a Service (SaaS) paradigm and allows to place tenants at database servers so that providing their isolation, data backup and the most effective usage of available computational power. One of the most important problems about such a system is an effective distribution of data into servers, which affects the degree of individual cluster nodes load and fault-tolerance. This paper considers the data-management approach, based on the usage of a load-balancing quality measure function. This function is used during initial placement of new tenants and also during placement optimization steps. Standard schemes of metaheuristic optimization such as simulated annealing and tabu search are used to find a better tenant placement.

Сведения об авторе:
Бойцов Евгений Александрович,
Ярославский государственный университет им. П.Г. Демидова,
аспирант кафедры теоретической информатики