ИНФОРМАТИКА

УДК 004.051

МЕТОД УВЕЛИЧЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ИНТЕРНЕТ-ОРИЕНТИРОВАННЫХ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

С.С. КУЛИКОВ, О.Г. СМОЛЯКОВА

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 14 апреля 2010

На примере конкретной технической задачи рассмотрены алгоритмы обработки данных в интернет-ориентированных клиент-серверных приложениях. Проведено практическое исследование производительности рассмотренных алгоритмов. Предложен метод выбора оптимального алгоритма с целью увеличения производительности интернет-ориентированных клиент-серверных приложений.

Ключевые слова: клиент-серверные приложения, производительность, обработка данных.

Введение

Переход к многоуровневой архитектуре интернет-ориентированных клиент-серверных приложений поставил перед разработчиками проблему распределения функций обработки данных между клиентской и серверной частями приложения.

Обозначенная проблема остается актуальной на стыке любых уровней многоуровневой архитектуры: уровня данных и уровня бизнес-логики, уровня бизнес-логики и уровня представления и т.п. Неверные технологические решения приводят к значительному падению производительности таких приложений, делая невозможной их работу под расчётной нагрузкой, а также значительно осложняя их дальнейшую модификацию и развитие.

В статье предложен метод увеличения производительности интернет-ориентированных клиент-серверных приложений, основанный на технологии тестирования производительности и повышении технической эффективности взаимодействия уровней данных и бизнес-логики.

Формулировка задачи

Дальнейшие рассуждения будут проводиться на основе следующей задачи, являющейся показательной в контексте исследования.

На уровне данных, представленном базой данных под управлением СУБД MySQL, хранится таблица, содержащая некоторое неизвестное количество записей (напр., новостей). Уровень бизнес-логики представлен программой на веб-ориентированном языке программирования PHP. Необходимо сформировать и представить конечному пользователю календарь новостей, содержащий ссылки только на те годы и месяцы, новости за которые представлены в базе данных. Отметим, что подобная задача является типичной для широкого спектра интернет-ресурсов. Фрагмент структуры таблицы news, содержащий представляющие интерес для решения поставленной задачи поля, базы данных представлен в табл. 1.

Рассмотрим варианты решения задачи, связанные с ними проблемы и пути их решения.

Таблица 1. **Фрагмент структуры таблицы news базы данных**

Поле таблицы	Тип поля таблицы	Дополнительная информация
news_uid	int	Первичный ключ
news_dt	int	Индекс

Проблема проектирования базы данных и выбора алгоритма обработки данных

СУБД MySQL поддерживает взаимодействие с клиентскими приложениями посредством языка структурированных запросов SQL. Язык SQL предоставляет программисту возможность реализовать часть обработки данных путём выполнения специальных запросов.

Такое решение позволяет упростить уровень бизнес-логики приложения, однако не всегда оказывается наиболее выгодным по критериям произволительности и затрат оперативной памяти. Оптимизация обработки данных по двум вышеперечисленным критериям является нетривиальной задачей, требующей для своего решения выбора определённого алгоритма обработки данных и специального проектирования структуры базы данных, учитывающей особенности работы выбранного алгоритма.

Проблема выбора алгоритма. Для решения поставленной задачи в общем случае существует три алгоритма [1].

Алгоритм 1, основанный на прямой выборке данных (рис. 1), сводится к извлечению всей информации с уровня данных на уровень бизнес-логики с последующим анализом и обработкой.

Алгоритм 2, основанный на частичной выборке данных (рис. 2), предполагает формирование диапазонов значений, соответствующих указанным в исходной формулировке задачи, с последующей проверкой наличия записей, попадающих в сформированные диапазоны.

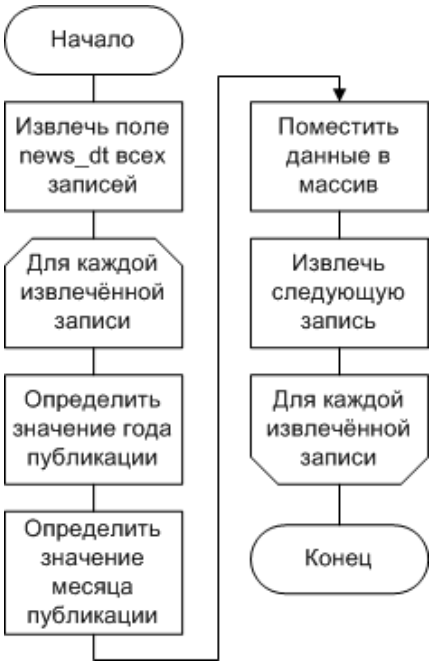

Рис. 1. Алгоритм, основанный на прямой выборке данных

Рис. 2. Алгоритм, основанный на частичной выборке данных

Алгоритм 3 предполагает совершенно иной подход (рис. 3), основанный на формировании готового набора данных средствами СУБД, что позволяет на уровне бизнес-логики выполнить только небольшие преобразования форматов представления данных с целью дальнейшей их передачи на уровень представления.

Проблема проектирования базы данных. В контексте проектирования базы данных нас будут интересовать вопросы использования индексов и выбора типа данных для хранения даты-времени публикации новости.

Ответ на первый вопрос оказывается очевидным в случае, если количество операций чтения данных значительно превышает количество операций вставки и модификации данных.

Рассматриваемая задача относится как раз к такому случаю, поэтому мы используем индекс на поле news_dt (табл. 1).

Ответ на второй вопрос не является столь же очевидным. Несмотря на то, что СУБД MySQL предоставляет широкий набор типов данных для хранения даты-времени, операции с unixtime (целым числом, представляющим собой время, выраженное в количестве секунд, прошедших с 1 января 1970 г.), как правило, оказываются более производительными.

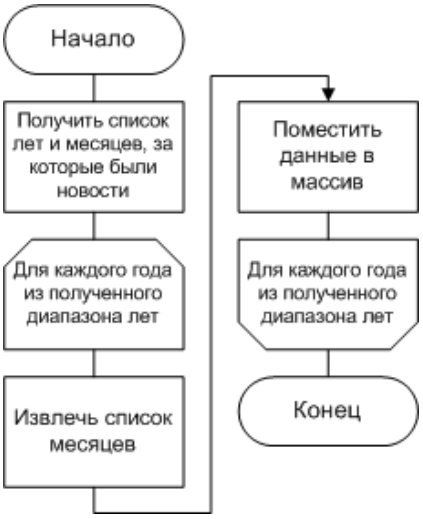

Рис. 3. Алгоритм, основанный на формировании готового набора данных средствами СУБД

Выбор формата хранения даты-времени также влияет на выбор того или иного из рассмотренных выше алгоритмов. Так, алгоритмы 1 и 2 не зависят от формата данных в базе данных, так как выполняют обработку информации на уровне бизнес-логики. Алгоритм 3 ориентирован в первую очередь на специализированный формат представления данных datetime, и его адаптация к формату unixtime негативно сказывается на производительности, так как требует дополнительных операций по преобразованию форматов данных.

Экспериментальное исследование

Перед формулировкой предлагаемого метода увеличения производительности интернет-оиентированнх клиент-серверных приложений предлагается рассмотреть результаты тестирования производительности, проведенного с применением каждого из рассмотренных выше алгоритмов и обоих вариантов хранения данных в базе данных.

Исследование проводилось на наборах данных, содержащих 100, 1000, 10000, 100000 и 1000000 записей со случайными значениями даты-времени в диапазоне от 1970 до 2010 г. Программные средства, реализующие каждый алгоритм на каждом из вариантов хранения информации, запускались в случайном порядке не менее 1000 раз каждое. Для исследования использовалась СУБД MySQL версии 5.1.44, язык программирования PHP версии 5.3.0 и веб-сервер Apache версии 2.2.14.

Аппаратное обеспечение в данном случае не является значимым, поскольку основной интерес представляет относительная производительность рассмотренных выше алгоритмов на каждом из вариантов хранения данных.

Альтернативное исследование, проведенное путем исполнения тестируемых программных средств в консольном режиме (без применения веб-сервера Apache), показало, что для данной задачи наличие или отсутствие дополнительного уровня программного обеспечения (веб-сервера) не является принципиальным, поскольку веб-сервер в данном случае обеспечивает только передачу данных между уровнем представления и уровнем бизнес-логики, что не является вычислительно сложной задачей и занимает сравнительно малое время в сравнении со временем выполнения основной части алгоритма, реализующей основную логику решения поставленной задачи.

Результаты эксперимента (время выполнения программы, решающей поставленную задачу) приведены в табл. 2. В пользу предлагаемого в настоящей статье метода говорит тот факт, что эксперимент по оценке производительности алгоритмов (который приходится неоднократно повторять при разработке реальных приложений) выполнялся на протяжении двух недель, что не может считаться допустимым при коммерческой разработке программного обеспечения.

Таблица 2. **Время выполнения программы, решающей поставленную задачу**

Алгоритм и формат данных	100 записей	1000 записей	10000 записей	100000 записей	1000000 записей
Алгоритм 1 и unixtime	0,03 с	0,24 с	2,49 с	25,35 с	263,72 с
Алгоритм 2 и unixtime	0,67 с	0,82 с	0,85 с	0,91 с	15,72 с
Алгоритм 3 и unixtime	0,002 с	0,02 с	0,12 с	0,67 с	16,44 с
Алгоритм 1 и datetime	0,002 с	0,017 с	0,08 с	1,74 с	31,84 с
Алгоритм 2 и datetime	0,75 с	0,92 с	0,99 с	1,27 с	26,60 с
Алгоритм 3 и datetime	0,003 с	0,01 с	0,03 с	0,29 с	12,77 с

Как показывает экспериментальное исследование, время работы алгоритма 1 жёстко зависит от количества обрабатываемых данных, в то время как время работы алгоритмов 2 и 3 остаётся на приемлемом уровне, и их производительность снижается незначительно. Оценка производительности предлагаемых алгоритмов представлена в табл. 3.

Таблица 3. **Производительность предложенных алгоритмов (записей в секунду)**

Алгоритм и формат данных	100 записей	1000 записей	10000 записей	100000 записей	1000000 записей
Алгоритм 1 и unixtime	3333,3(3) з/с	4166,6(6) з/с	4016,06 з/с	3944,77 з/с	3791,90 з/с
Алгоритм 2 и unixtime	149,25 з/с	1219,51 з/с	11764,71 з/с	109890,11 з/с	63613,23 з/с
Алгоритм 3 и unixtime	50000 з/с	50000 з/с	83333,3(3) з/с	149253,73 з/с	60827,25 з/с
Алгоритм 1 и datetime	50000 з/с	58823,53 з/с	125000 з/с	57471,26 з/с	31407,04 з/с
Алгоритм 2 и datetime	133,3(3) з/с	1086,96 з/с	10101,01 з/с	78740,16 з/с	37593,98 з/с
Алгоритм 3 и datetime	33333,3(3) з/с	100000 з/с	333333,3(3) з/с	344827,59 з/с	78308,54 з/с

Особый интерес представляет тот факт, что алгоритм 2, показывавший худшую производительность на малых объёмах данных (отставание от лидера в 330–370 раз), на больших объёмах данных показал отставание всего в 1,2–2,1 раза. Экстраполяция полученных результатов на случай с ещё большими объёмами данных позволяет рассматривать данный алгоритм в качестве одного из наилучших кандидатов для решения поставленной задачи.

Отметим также, что все рассмотренные алгоритмы показали снижение производительности при проверке на 1000000 записей в сравнении с проверкой на 100000 записей, что вызвано необходимостью выполнения дополнительных операций с оперативной и внешней памятью на уровне данных и уровне бизнес-логики.

Управление такими операциями контролируется операционной системой и не всегда может быть оптимизировано средствами языка программирования, на котором реализовано решение поставленной задачи. Этот факт также является аргументом в пользу использования алгоритма 2, предъявляющего значительно меньшие требования к объёму используемой памяти.

В силу того факта, что производительность предложенных алгоритмов зависит не только от объёма обрабатываемых данных, ниже предложен метод выбора алгоритма, наиболее подходящего для решения тех или иных технических задач.

Формулировка предлагаемого метода

Предлагаемый в настоящей статье метод увеличения производительности интернет-ориентированных клиент-серверных приложений построен на анализе факторов, влияющих на производительность алгоритмов обработки данных.

К таким факторам относятся: вычислительная сложность обработки информации на уровне данных, вычислительная сложность [2] обработки информации на уровне бизнес-логики, объём обрабатываемых данных, объём данных, передаваемых между уровнями приложения, вычислительная сложность дополнительной обработки данных для их передачи на уровень представления, вычислительная сложность преобразования формата данных для их последующей обработки на уровне данных, плотность распределения данных по анализируемым диапазонам.

Выбор оптимального алгоритма осуществляется путём вычисления коэффициентов (1, 2, 3) и принятия конечного решения на основе формулы (4).

$$K_{\text{сложности}} = \frac{C_{\text{данных}} + C_{\text{обработки}}}{C_{\text{логики}} + C_{\text{представления}}}, \quad (1)$$

где $K_{\text{сложности}}$ — отношение вычислительной сложности обработки информации на уровне данных к вычислительной сложности обработки информации на уровне бизнес-логики; $C_{\text{данных}}$ — вычислительная сложность обработки информации на уровне данных; $C_{\text{обработки}}$ — вычислительная сложность преобразования формата данных для их последующей обработки на уровне данных; $C_{\text{логики}}$ — вычислительная сложность обработки информации на уровне бизнес-логики; $C_{\text{представления}}$ — вычислительная сложность дополнительной обработки данных для их передачи на уровень представления.

$$K_{\text{данных}} = \frac{V_{\text{обр}} + V_{\text{перед}}}{V_{\text{техн}}}, \quad (2)$$

где $K_{\text{данных}}$ — отношение суммарного объёма обрабатываемых и передаваемых для обработки данных к объёму передаваемых технических данных; $V_{\text{обр}}$ — объём обрабатываемых данных; $V_{\text{перед}}$ — объём передаваемых для обработки данных; $V_{\text{техн}}$ — объём передаваемых технических данных.

$$K_{\text{заполнения}} = \frac{Q_{\text{зап}}}{Q_{\text{диап}}}, \quad (3)$$

где $K_{\text{заполнения}}$ — отношение количества заполненных анализируемых диапазонов значений к общему количеству анализируемых диапазонов значений; $Q_{\text{зап}}$ — количество заполненных анализируемых диапазонов значений; $Q_{\text{диап}}$ — общее количество анализируемых диапазонов значений.

Решение о выборе оптимального алгоритма принимается по формуле

$$A = \begin{cases} 1,\ K_{\text{сложности}} >> 1,\ K_{\text{данных}} \approx 1 \\ 2,\ K_{\text{сложности}} > 1,\ K_{\text{заполнения}} \approx 1, \\ 3,\ K_{\text{сложности}} < 1, K_{\text{данных}} > 1 \end{cases} \quad (4)$$

где A — номер оптимального для решения конкретной задачи алгоритма.

Также отметим, что написание вспомогательного программного средства или использование среды автоматизации тестирования позволяет проводить сбор и анализ данных, необходимых для расчёта номера оптимального для решения конкретной задачи алгоритма A (4) в автоматизированном режиме.

Заключение

Применение предложенного метода позволяет при определении оптимального алгоритма избежать необходимости выполнять длительное тестирование производительности, которое, в свою очередь, требует реализации всех конкурирующих алгоритмов. Таким образом, применение предложенного метода позволяет не только увеличить производительность интернет-ориентированных клиент-серверных приложений, но и сократить время и трудоёмкость их разработки.

Учёт количества обрабатываемой информации (2) и результаты практических исследований, представленные в табл. 3, позволяют прогнозировать изменение производительности программного средства, решающего сформулированную в данной статье задачу. Такой прогноз позволяет ещё на стадии проектирования программного средства снизить риск недопустимого падения производительности, приводящего к необходимости внесения модификаций в программное средство на стадии его эксплуатации.

THE METHOD FOR PERFORMANCE ACCELERATION OF INTERNET-ORIENTED CLIENT-SERVER APPLICATIONS

S.S. KULIKOV, O.G. SMOLYAKOVA

Abstract

Three data-processing algorithms for client-server applications are reviewed and analysed for their performance. Database architecture changes for performance optimisation are reviewed. The method for performance acceleration of internet-oriented client-server applications is proposed.

Литература

1. *Дюбуа П.* MySQL — сборник рецептов. М., 2007.
2. *Липский В.* Комбинаторика для программистов. М., 1988.