УДК 004.42
DOI: 10.15827/0236-235X.131.471-475
Дата подачи статьи: 18.11.19
2020. Т. 33. № 3. С. 471–475

Использование параллельной обработки данных для оптимизации работы программного обеспечения

И.Ю. Артемов [1], *старший преподаватель, oollpp@mail.ru*

[1] *Тверской государственный технический университет, г. Тверь, 170026, Россия*

В статье описываются предпосылки и результаты работ по ускорению времени работы одной из подсистем системы автоматизации процедуры сбора данных сотрудниками вне офиса для ERP-систем. Приводится информация о существующих системах, предназначенных для сбора данных с помощью мобильных устройств. Рассмотрены недостатки систем, работающих в режиме интерактивного (online) или пакетного (offline) обмена данными между серверной частью системы и ее мобильными клиентами, предложены способы их устранения, основанные на совмещении сильных сторон двух вариантов обмена данными (online и offline).

Основное внимание автор уделяет описанию механизма генерации данных, необходимых и достаточных для работы мобильных устройств, на основе которого было разработано ПО. В ходе эксплуатации ПО выявлены недостатки предложенной схемы, связанные с неудовлетворительной оперативностью формируемых данных. Приводится детальное описание вариантов обработки данных, показана возможность их организации в конвейер команд обработки данных.

В статье названы причины возникновения необходимости в проведении работ по оптимизации процедуры обработки данных. Предлагаются варианты ускорения существующих процедур обработки за счет использования возможностей современных серверных систем, основой которых являются кластеры серверов с большим количеством ядер центрального процессора.

Приведены аналитические расчеты теоретически возможного прироста производительности при использовании параллельной обработки данных, а также результаты испытания разработанного алгоритма на основе реальных данных. При этом фактический прирост производительности алгоритма составил 2,38 раза при теоретически возможном 2,51 раза.

Ключевые слова: *параллельная обработка, сервер, мобильный терминал, кластер.*

Многие компании по мере роста бизнеса приходят к пониманию, что им требуется ERP-система, позволяющая хранить и обрабатывать критически важные данные, без которых деятельность невозможна [1]. В ERP-системе большинство собранных данных – действительно необходимые для оперативного контроля и принятия управленческих решений сведения и процессы. Средний и крупный бизнес с каждым днем все активнее пользуется подобными средствами. Территориальные представители несут ответственность за выполнение плана продаж компании, и их работа с клиентами автоматизируется модулем ERP – CRM-системой, предназначенной для автоматизации стратегий взаимодействия с клиентами, в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания путем сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения бизнес-процессов и последующего анализа результатов.

В настоящее время многие компании имеют штат работающих вне офиса сотрудников для сбора тех или иных данных. Например, в фармакологической отрасли это региональные представители, медицинские представители и мерчендайзеры. В их обязанности входят сбор различной информации в торговых точках, проведение презентаций и обучение персонала торговых точек. В зависимости от бизнеса компании количественный и качественный состав собираемых показателей может варьироваться. При этом для автоматизации деятельности полевых сотрудников используются два подхода к организации обмена данными между сервером и мобильным терминалом [2, 3]: offline-режим, в котором обмен выполняется не чаще одного или двух раз в день (как правило, такой обмен требует наличия стабильного выделенного канала, так как объем передаваемых данных довольно значителен – сотни мегабайт), и online-режим, когда все данные получаются непосредственно с сервера в момент запроса (при этом возможно частичное кеширование данных на стороне клиента).

В статье [3] была предложена архитектура системы, в рамках которой реализуется альтернативный подход, объединяющий сильные стороны обоих вариантов. Основная идея заклю-

чается в том, что обмен с сервером осуществляется в режиме online, но данные для обмена (фактически являющиеся статически сгенерированным кешем персональных данных мобильного терминала) подготавливаются на сервере с определенной периодичностью. Эксплуатация данной системы показала правильность выбранных архитектурных решений. Однако с ростом нагрузки на систему в силу увеличения ее функциональных возможностей, а также географии и количества подключенных к системе сотрудников подсистема генерации пакетов данных, необходимых и достаточных для работы мобильных клиентов в полностью автономном режиме (kern_sync) [3], оказалась неспособной обрабатывать массив поступающих данных с приемлемой оперативностью. Использование кластерных технологий, описанных в статье [4], позволило только частично решить возникшую проблему, так как данный подход в большей степени ориентирован на повышение надежности системы в целом, а не ее отдельных подсистем.

В настоящее время существуют несколько систем, которые изначально создавались для сбора данных с помощью мобильных терминалов в условиях нестабильной работы канала передачи данных [5, 6]. Они используют модифицированный offline-режим, в котором обмен данными можно проводить достаточно часто, так как размер передаваемых данных ограничен. К сожалению, при проектировании этих систем были допущены серьезные просчеты, которые привели к избыточности функций (дублирование бизнес-процессов *корпоративной информационной системы* (КИС), например, ведение остатков) или к жесткой привязке к программным платформам (например, 1С). В результате многолетней эксплуатации несколько имеющихся в данный момент времени систем (в том числе и лидер в данной отрасли – CDC Optimum) либо требуют постоянного контроля за их работой, либо сложно масштабируются в случае использования в крупных проектах (автоматизация крупных производителей).

Работа подсистемы генерации пакетов данных включает следующие основные этапы (шаги обработки данных):

– загрузка данных из БД в оперативную память [7–9];

– обработка загруженных в память данных с целью формирования индивидуального набора данных для каждого подключенного к системе пользователя;

– формирование для каждого пользователя индивидуального набора пакетов данных в текстовом формате;

– уменьшение размера сформированных пакетов с помощью алгоритмов потокового сжатия;

– разностная публикация сформированного набора пакетов на общий ресурс для дальнейшей загрузки на мобильные устройства.

При этом обработка загруженных в память данных с целью формирования индивидуального набора данных для каждого подключенного к системе пользователя выполняется с помощью следующих алгоритмов обработки данных.

1. Персонализация данных по уникальному критерию на основе параметров пользователя (пример – в таблице 1).

Таблица 1

Персонализация списка клиентов для пользователя с user_id = dc_m_2

Table 1

Customer list personalizing for user with user_id = dc_m_2

№	user_id	account_id
1	dc_m_1	138066
2	dc_m_1	126013
3	dc_m_1	137727
4	dc_m_2	5090
5	dc_m_2	ZIGL3
6	dc_m_3	126013

⇓

№	user_id	account_id
1	dc_m_2	5090
2	dc_m_2	ZIGL3

2. Фильтрация данных одного справочника на основе данных из другого справочника (пример – в таблице 2).

3. Построение иерархического списка на основе линейного списка данных (пример – в таблице 3).

4. Группировка данных в списке по ключевому полю (пример – в таблице 4).

Организация перечисленных алгоритмов в конвейер последовательно выполняемых операций над данными позволяет осуществлять обработку данных достаточно высокой степени сложности. Для улучшения скоростных характеристик процедуры обработки данных можно использовать механизмы параллельной обработки в алгоритме подготовки данных, а также оптимизировать внутренние механизмы работы алгоритмов обработки данных.

Таблица 2

Фильтрация справочника accounts по my_accounts

Table 2

Catalog cleaning accounts by means of my_accounts

№	account_id
1	138066
2	5090
3	ZIGL3

№	user_id	account_id
1	dc_m_2	5090
2	dc_m_2	ZIGL3

⇓

№	account_id
1	5090
2	ZIGL3

Таблица 3

Построение иерархического списка products

Table 3

The hierarchical list generation of products

№	prod_id	pid
1	SPECIALTY	
2	TURBOMAX	
3	5001635	SPECIALTY
4	5001636	SPECIALTY
5	5001637	SPECIALTY
6	5001638	TURBOMAX

⇓

№	prod_id	pid
1	SPECIALTY	
2	5001635	SPECIALTY
3	5001636	SPECIALTY
4	5001637	SPECIALTY
5	TURBOMAX	
6	5001638	TURBOMAX

Проведенные исследования показали, что для работы всех элементов системы вполне достаточно выделить от 4 до 6 процессорных ядер. Учитывая то, что для работы системы, как правило, используются серверы с двумя процессорами серии Intel Xeon E5-26xx, суммарное количество ядер в котором составляет от 12 до 20, получается, что под задачи генерации данных можно без каких-либо серьезных последствий выделить минимум 8 процессорных ядер. При этом профиль нагрузки представляет собой массив данных, состоящих из 130 структурно разнородных наборов данных суммарным объемом 45 миллионов строк, который после загрузки в оперативную память занимает около 5,5 GB.

Таблица 4

Группировка данных

Table 4

Drilling down a database

№	user_id	pid
1	dc_m_1	dc_ise_1
2	dc_m_1	dc_ise_2
3	dc_m_2	dc_ise_1
4	dc_m_3	dc_ise_1
5	dc_m_3	dc_ise_2
6	dc_m_4	dc_ise_2

⇓

№	user_id	pid
1	dc_m_1	dc_ise_1
2	dc_m_2	dc_ise_1
3	dc_m_3	dc_ise_1
4	dc_m_4	dc_ise_2

Для сокращения времени работы подсистемы сбора данных сотрудниками применена параллельная обработка. Согласно закону Амдала [10], ускорение выполнения программы за счет распараллеливания ее инструкций на множестве вычислителей ограничено временем, необходимым для реализации ее последовательных инструкций: $Sp = 1 / (a + (1 – a)/p)$, где p – количество ядер процессора, которые можно задействовать под обработку данных; a – доля операций от общего объема вычислений, которая может быть получена только последовательными расчетами.

Анализ работы подсистемы генерации данных показал, что ее работа состоит из следующих этапов (исследования проводились на тестовом сервере Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz 4 cores, 16GB RAM):

– загрузка данных из БД – 110 сек.;

– обработка данных – 1 057 сек. (с учетом подготовки пакетов и их сжатия);

– публикация данных на общий ресурс – 4 сек.

Загрузка данных не может быть распараллелена, так как БД является центральным элементом системы и повышение нагрузки на нее должно всячески снижаться.

Распараллеливание публикации данных не имеет смысла, так как ее вклад в суммарные временные задержки незначителен и трудозатраты на ее оптимизацию никогда не окупятся.

В результате получается, что в данном случае доля последовательных операций составляет 9,7 %. В таблице 5 показано теоретически возможное ускорение работы при параллельной обработке данных.

Таблица 5

Теоретически возможное ускорение работы при параллельной обработке данных (доля последовательных операций *a* = 9,7 %)

Table 5

Possible in the theory of work intensification when data-parallel (share of sequence operations *a* = 9,7 %)

Количество используемых ядер	Прирост производительности	Общее время обработки данных, сек.	Время генерации данных, сек.
1	1	1171,041	1057,061
2	1,82	642,510	528,531
3	**2,51**	**466,333**	**352,354**
4	3,10	378,245	264,265
5	3,60	325,392	211,412
6	4,04	290,157	176,177

Учитывая, что данные, содержащиеся в исходном наборе, никогда не меняются (то есть к ним возможен read-only-доступ из параллельно работающих потоков без необходимости установки блокировки в момент доступа к данным), каждый поток обработки данных создает свою копию адресов участков памяти, в которой содержится загруженная из БД информация, и работает только с этой локальной копией адресного пространства. Такой подход позволяет свести к нулю количество блокировок при обращении к общим массивам данных и за счет этого приблизиться к теоретически возможному повышению скорости работы. Обратной стороной медали является значительное увеличение (от 40 до 300 %) оперативной памяти, требуемой для обработки данных. В таблице 6 показан фактический прирост производительности подсистемы генерации данных после внедрения разработанной параллельной обработки данных.

Таблица 6

Фактическое ускорение работы при параллельной обработке данных (доля последовательных операций *a* = 9,47 %)

Table 6

The factual work intensification when data-parallel (share of sequence operations *a* = 9,47 %)

Количество ядер CPU	Загрузка, сек.	Генерация, сек.	Публикация, сек.	Итого, сек.	Прирост производительности
1	98,302	981,607	4	1079,909	1
2	98,32	504,865	4	603,185	1,79
3	**98,582**	**356,595**	**4**	**455,177**	**2,38**
4	93,662	296,164	4	389,826	2,74

Как видно из приведенных данных, фактический прирост производительности при работе алгоритма на трех физических CPU составил 2,38 раза при теоретически возможном 2,51 раза, уровень оптимальности работы созданного алгоритма параллельной обработки данных составил 94,8 %.

В заключение необходимо отметить, что внедрение методики параллельной обработки данных позволило значительно ускорить работу подсистемы подготовки данных в реальных проектах, использующих описанную в статье [1] систему.

Литература

1. Кинзябулатов Р. Что такое ERP система. URL: https://habr.com/ru/company/trinion/blog/333018/ (дата обращения: 03.11.2019).

2. Siebel CRM Products. URL: https://www.oracle.com/applications/siebel/products.html#service (дата обращения: 03.11.2019).

3. Калабин А.Л., Артемов И.Ю. Программный комплекс автоматизации процедуры сбора данных // Программные продукты и системы. 2013. № 2. С. 137–141.

4. Артемов И.Ю. Кластер высокой доступности программного комплекса автоматизации процедуры сбора данных // Программные продукты и системы. 2013. № 4. С. 149–153.

5. Мобильная торговля ОПТИМУМ ИСУМТ. Сбор заказов, мерчендайзинг, торговля с колес. URL: http://www.cdc.ru/solutions/optimum-asumt (дата обращения: 03.11.2019).

6. Надежные решения для управления дистрибуцией и цифровизации продаж. URL: http://www.sys4tec.com (дата обращения: 03.11.2019).

7. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика; [пер. с англ.]. М., 2003. 1440 с.

8. Моргунов Е.П., Рогов Е.В., Лузанов П.В. PostgreSQL. Основы языка SQL. СПб: БХВ-Петербург, 2018. 336 с.

9. Стивенс У.Р., Раго С.А. UNIX. Профессиональное программирование. СПб: Питер, 2018. 944 с.

10. Лацис А.О. Параллельная обработка данных. М.: Академия, 2010. 336 с.

Software & Systems
DOI: 10.15827/0236-235X.131.471-475

Received 18.11.19
2020, vol. 33, no. 3, pp. 471–475

Parallel processing for software system performance optimization

I.Yu. Artemov [1], *Senior Lecturer, eoollpp@mail.ru*

[1] *Tver State Technical University, Tver, 170026, Russian Federation*

Abstract. The article describes the supposition and working results to improve the operating time of one of the subsystems of the automation system of data harvesting procedure by operators that are out of the office for EPR-system. There is information about the current systems which are dedicated to data harvesting with mobile devices. The author determines the system weakness which works in the mode of interactive (online) or batch (offline) data exchange between the server part of the system and its mobility part, and proposes its remedies for, based on the connection of two ways of exchange data (online and offline).

The author devotes the main attention to the mechanism description for generating data necessary and sufficient for the mobile device operation, on the basis of which it was developed software. During the operation of the software, the problems of the proposed scheme were identified, due to the unsatisfactory efficiency of the generated data. The author provides a detailed description of the data processing options. The author shows their organization's capability in the instruction pipeline of data processing.

The paper gives reasons for the need for work to optimize data processing procedures.

The author proposes ways for speeding up existing processing procedures by using the capabilities of modern server systems, which are based on clusters of servers with a large number of CPU cores.

The author provides analytical calculations of theoretically possible performance gains when using parallel data processing, as well as the results of testing the developed algorithm based on real data. At the same time, the actual performance increase of the algorithm was 2.38 times, while the theoretically possible 2.51 times.

Keywords: parallel processing, server, mobile terminal, cluster.

References

1. Kinzyabulatov R. *What is an ERP System*. Available at: https://habr.com/ru/company/trinion/blog/333018/ (accessed November 03, 2019).
2. *Siebel CRM Products*. Available at: https://www.oracle.com/applications/siebel/products.html#service (accessed November 03, 2019).
3. Kalabin A.L., Artemov I.Yu. Automated software system of data collection procedures. *Software & Systems*, 2013, no. 2, pp. 137–141 (in Russ.).
4. Artemov I.Yu. Ha-cluster of data collection system. *Software & Systems*, 2013, no. 4, pp. 149–153 (in Russ.).
5. *Mobile Sales OPTIMUM ISUMT. Pre Selling, Merchandising, Van Selling*. Available at: http://www.cdc.ru/solutions/optimum-asumt (accessed November 03, 2019).
6. *Reliable Solutions for Managing Distribution and Automation of Visiting Field Workers*. Available at: http://www.sys4tec.com (accessed November 03, 2019).
7. Connolly T.M., Begg C.E. *Database Systems. A Practical Approach to Design, Implementation, and Management*. Addison Wesley, 2002, 1236 p. (Russ. ed.: Moskow, 2003, 1440 p.).
8. Morgunov E.P., Rogov E.V., Luzanov P.V. *PostgreSQL. SQL Fundamentals*. St. Petersburg, 2018, 336 p.
9. Stevens W.R., Rago S.A. *Advanced Programming in the UNIX Environment.* Addison-Wesley, 2013, 1024 p. (Russ. ed.: St. Petersburg, 2018, 944 p.).
10. Latsis A.O. *Parallel Data Processing*. Moscow, 2010, 336 p. (in Russ.).

Для цитирования

Артемов И.Ю. Использование параллельной обработки данных для оптимизации работы программного обеспечения // Программные продукты и системы. 2020. Т. 33. № 3. С. 471–475. DOI: 10.15827/0236-235X.131.471-475.

For citation

Artemov I.Yu. Parallel processing for software system performance optimization. *Software & Systems*, 2020, vol. 33, no. 3, pp. 471–475 (in Russ.). DOI: 10.15827/0236-235X.131.471-475.