УДК 004.657 DOI: 10.30748/soi.2019.157.09

А.Г. Савенко, А.С. Гавриленко

Институт информационных технологий Белорусского государственного университета информатики и радиоэлектроники, Минск, Республика Беларусь

РАСПРЕДЕЛЕНИЕ НАГРУЗКИ ПРИ ПОСТРОЕНИИ ОТЧЁТОВ И ЗАПРОСОВ С БОЛЬШИМ ОБЪЁМОМ ДАННЫХ

Предложен подход по перераспределению нагрузки web-, мобильных- и desktop-клиент-серверных приложений для увеличения быстродействия при построении отчётов и запросов с большим количеством данных на технологиях PHP, RabbitMQ, Redis и реляционной базе данных. Предложенный подход позволяет значительно увеличить быстродействие существующих, не предназначенных изначально для обработки больших данных, приложений, а также разрабатывать новые клиент-серверные приложения, предназначенные для обработки больших объёмов информации (большого количества запросов).

Ключевые слова: *распределение нагрузки, big data, запросы, API, клиент-серверные приложения.*

Введение

Постановка проблемы. При увеличении запросов в небольших и средних по объёмам обрабатываемой информации приложениях или сайтах, возникает вопрос о расширении тех или иных возможностей для ускорения отображаемых или загружаемых данных.

Анализ последних исследований и публикаций. В работах [1–12] приводится анализ технологий, языков программирования и инструментов для обработки больших данных, однако не учитывается возможность модернизации существующих проектов для обеспечения работоспособности и повышения быстродействия при возрастающих объёмах обрабатываемой информации.

Цель статьи. Данная статья предлагает подход по перераспределению нагрузки при построении отчётов и запросов с большим объёмом данных в клиент-серверных приложениях. В качестве исходного проекта будет рассмотрен проект, состоящий из web-, мобильного и desktop клиент-серверного приложения.

Изложение основного материала

В настоящее время, практический любой коммерческий или некоммерческий Интернет-проект реализован в виде web-ресурса, мобильного и/или desktop-приложения. Все они подключаются к определенному API для осуществления взаимодействие по схеме клиент-сервер. В какой-то момент, при увеличении числа запросов, единственный воркер (процесс, который выполняется в фоновом режиме), работающий на сервере, начинает отключаться, вследствие чего возникает ошибка. Данные события характеризуются тем, что API не хватает мощностей проекта, и даже после очередной перезагрузки воркера, работоспособность инструмента не может быть восстановлена.

Решением, в данном случае, может являться распределение нагрузки между нодами (сервер, или несколько серверов) расположенных на компьютерах, которые перенаправляют всю нагрузку от одного сервера на другие, тем самым упрощая и ускоряя работу единственного сервера.

Предлагается следующая схема перераспределения нагрузки. Для API, которое изначально выполняло обязанности по получению информации, её обработке и выдаче, оставляется только функция по получению информации. Получая какой-либо запрос, будь-то GET, POST, PUT, DELETE или иной метод запроса, API отдаёт параметры запроса в очередь RabbitMQ. RabbitMQ – это бесплатный брокер сообщений для перераспределения сообщений между подписчиками (в данном случае, это воркеры развернутые на серверах). Rabbit должен состоять из следующих частей: exchange (коммутатор), подписчиков, сообщений, имеющих хэш, который записывается в Redis (быстрая база данных по типу ключ – значение), который выступает в роли ключа и параметров запроса, выступающих в роли значения. К данным параметрам добавляется также статус, в котором находится текущий процесс, (request – запрос, process – в работе, succeed – успешен). В сообщениях также передается routing_key – ключ назначения. В данном случае ключ служит для перенаправления на нужный контроллер, в котором метод осуществляет логику.

Например, следующий routing_key: “user.get” вызовет контроллер user с методом get, где Redis, получив данные от API, переведет процесс в статус запроса. После чего, когда воркер возьмет данный процесс в работу, он изменит статус на “process”, а когда обработает – изменит статус на “succeed”. После данных манипуляций вернется результат (рис. 1).

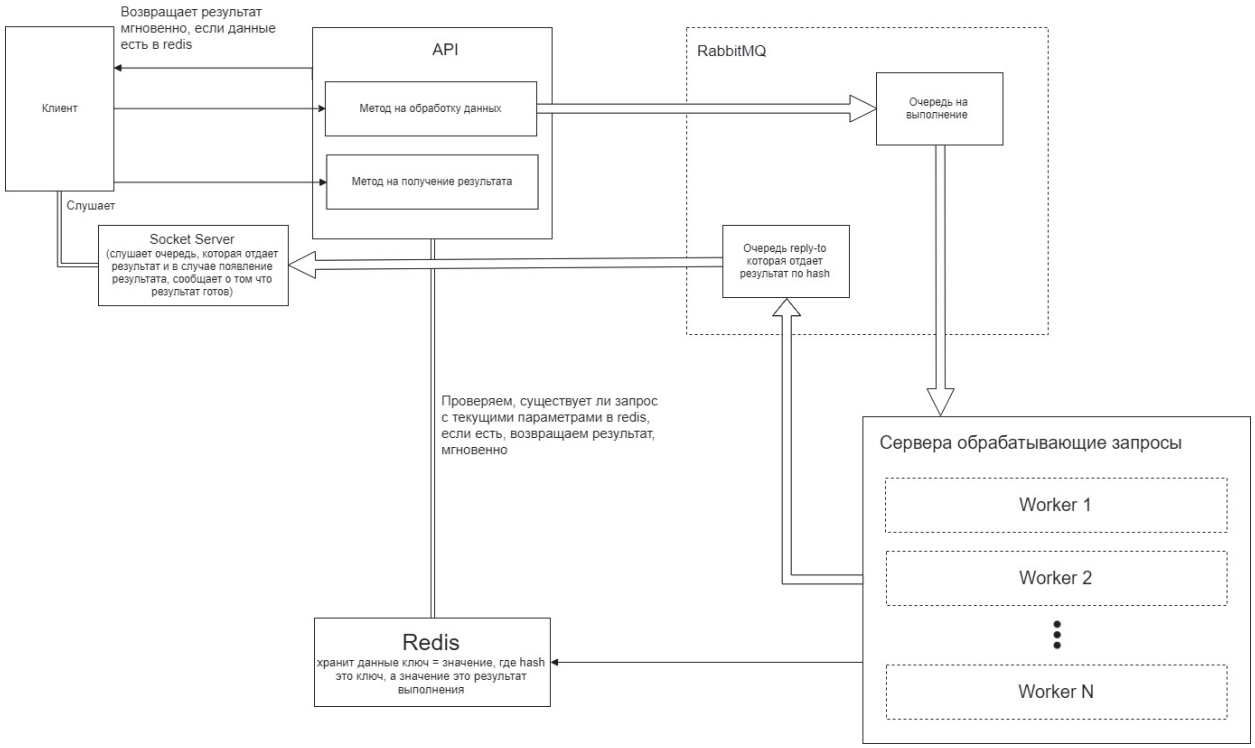

Рис. 1. Обобщённая схема работы приложения

Запустив очередной воркер, осуществляется подписка на коммутатор, который выполняет распределение нагрузки между подписчиками следующим образом: если подписчик занят в текущий момент, коммутатор передаст сообщение тому, кто свободен, или будет ждать того, кто быстрее всего освободится. Таким образом, подписчики выполняют все действия, которые ранее выполняло единственное API. Используя любую страницу, запрос, действие или функцию проекта, осуществляется распределение на воркер с наименьшей нагрузкой. Воркер, который выполнил определенный алгоритм действий, передает ответ в конкретную очередь, предназначенную для выполнения алгоритмов, и которая указана в сообщении. Параллельно всем действиям происходит подписка на очередь ответа (reply-to), в которую передается ответ алгоритма, заданного в воркере. И в случае какого-либо результата – передается ответ клиенту (рис. 2).

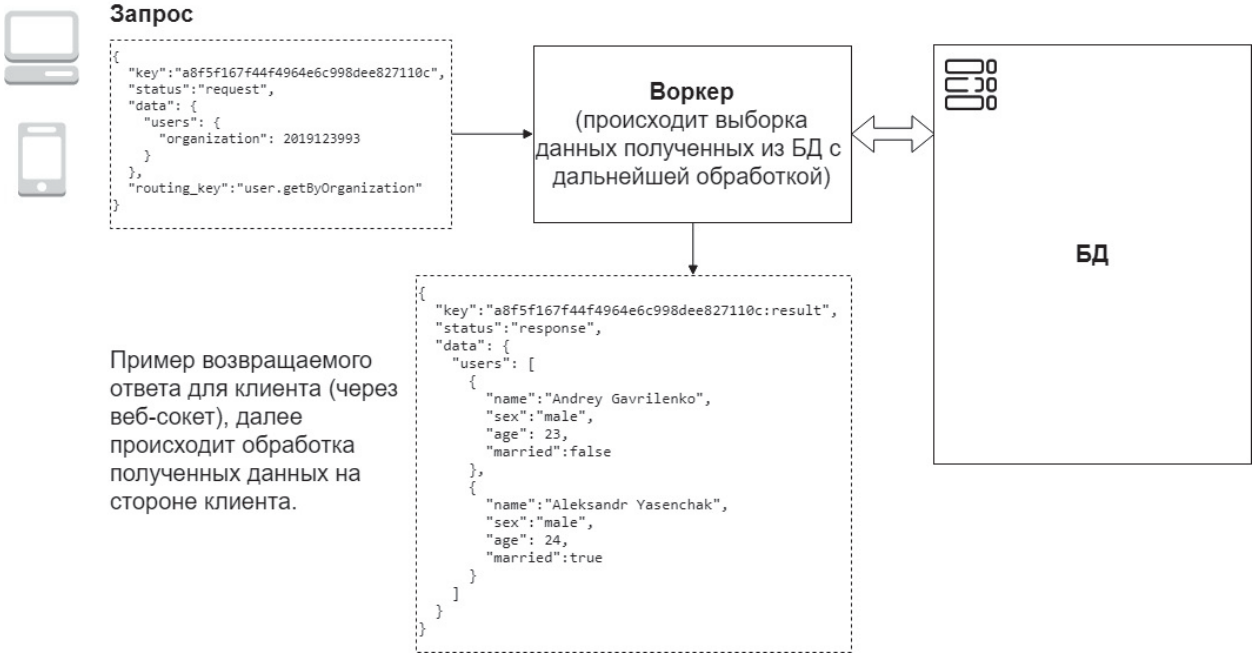

Рис. 2. Передача сообщения воркеру и обработка данных

Проанализируем данную схему на предмет быстродействия. Данные для обработки будут поступать под разнообразные задачи. Может наступить момент, когда воркеры загружены большими длительными задачами, что мешают выполнению мелких задач. Для того чтобы это не происходило, необходимо разграничить воркеры. Для разграничения воркеров нужно выявить мелкие задачи и задачи с большими объемами данных, тем самым разделив воркеры на несколько групп. Первая группа воркеров будет выполнять задачи по обработке больших данных, вторая группа воркеров будет выполнять мелкие задачи, также необходима третья группа воркеров, обрабатывающих отчеты. Этими действиями можно добиться максимального ускорения системы, т.к. воркеры будут разграничены, а мелкие задачи не буду ждать воркеров, обрабатывающих большие данные.

Нужно понимать, что если ограничиться равным количеством воркеров различных групп, то вести речь о быстродействии не уместно, т.к. некоторые воркеры некоторых групп в некоторые моменты времени будут простаивать впустую. Для исключения такой ситуации, нужно распределить количество воркеров:

– для мелких задач необходимо выделить небольшое количество воркеров (т.к. задачи быстро выполняются и не требуют много времени для ответа);

– для более сложных задач следует выделить большее количество воркеров (т.к. они обрабатываются гораздо дольше).

Если для схемы будет использоваться только один клиент, а результат при множественных одинаковых запросах обрабатывается каждый раз, то использование серверных мощностей будет не совсем рационально. Для того чтобы сервера, на которых работают воркеры, использовались только по острой необходимости, необходимо также использовать сервер Redis, как средство для хранения кэша обработанных результатов. В отличие от других баз данных (MSSQL или MySQL), Redis более высокопроизводительна в контексте быстроты получения и складирования данных. Так как Redis – это “легкая” база данных, поиск текущего хэша производится быстрее, чем очередная обработка такого же результата с последующей его обработкой на воркерах. Для осуществления кэширования требуется результат выполнения воркера отправить в Redis, где результат будет значением, а хэш, который был сгенерирован ранее, будет являться ключом.

Именно для увеличения скорости обработки данных, необходимо вынести часть работы API на сторонние сервера.

Для ещё большего увеличения быстродействие предложенной схемы можно для каждой выборки базы данных создать “интерфейсные обёртки” – специально созданную оптимизированную функцию, по которой будет производиться выборка.

Выводы

В данной статье был предложен подход по распределению нагрузки между серверами для увеличения быстродействия клиент-серверной системы при увеличении объёмов обрабатываемой информации. В предложенном решении используются технологии и инструменты, поддерживаемые как мобильными, web-, а также desktop-приложениями. Предложенный подход позволяет значительно увеличить быстродействие существующих, не предназначенных изначально для обработки больших данных, приложений, а также разрабатывать новые клиент-серверные приложения, предназначенные для обработки больших объёмов информации (большого количества запросов).

Список литературы

1. Выбор языка программирования для решения задач, связанных с применением технологии Big Data / В.Д. Львович, В.И. Анисимов, А.Л. Хотеев, М.В. Стержанов // BIG DATA и анализ высокого уровня: сборник материалов V Международной научно-практической конференции, Минск, 13–14 марта 2019 г. В 2 ч. Ч. 2 / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2019. – С. 117-120.

2. Боровиков С.М. Большие данные и принципы разработки аналитических систем / С.М. Боровиков, С.К. Дик, С.С. Дик // BIG DATA и анализ высокого уровня: сборник материалов V Международной научно-практической конференции, Минск, 13–14 марта 2019 г. В 2 ч. Ч. 2 / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2019. – С. 167-171.

3. Успенский,Н. Новые подходы управления данными / Н. Успенский // BIG DATA и анализ высокого уровня: сборник материалов V Международной научно-практической конференции, Минск, 13–14 марта 2019 г. В 2 ч. Ч. 1 / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2019. – С. 28-33.

4. Распределённые файловые системы для организации хранилищ структурированных данных / И. Н. Цырельчук, Е.Н. Шнейдеров, П.А. Берашевич, Н.А. Лос, А.С. Терешков // BIG DATA и анализ высокого уровня: сборник материалов IV Международной научно-практической конференции, Минск, 3–4 мая 2018 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2018. – С. 463-466.

5. Хадасевич А.И. Обработка больших объемов информации с использованием платформы Hadoop и службы облачных вычислений Microsoft Azure / А.И. Хадасевич, В.И. Швец // Компьютерные системы и сети: материалы 54-й на-

учной конференции аспирантов, магистрантов и студентов, Минск, 23 – 27 апреля 2018 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2018. – С. 208.

6. Heger D. Future of big data / D. Heger // BIG DATA and Predictive Analytics. Использование BIG DATA для оптимизации бизнеса и информационных технологий: сборник материалов международной научно-практической конференции. – Минск: БГУИР, 2015. – С. 72-75.

7. Карау Х. Эффективный Spark. Масштабирование и оптимизация / Х. Карау, Р. Уоррен. – СПб.: Питер, 2018. – 352 с.

8. Franks B. Taming the big data. Finding Opportunities in Huge Data Streams with Advanced Analytics / B. Franks. – Hoboken, New Jersey: Wiley, 2014. – 336 p.

9. Borovikov S. Prediction in Big Data Technology / S. Borovikov // BIG DATA and Advanced Analytics. Использование BIG DATA для оптимизации бизнеса и информационных технологий: сборник материалов II международной научно-практической конференции; Минск, 15-17 июня 2016 г. – Минск: БГУИР, 2016. – С. 98-101.

10. Лучшие практики разработки Big Data приложений на базе Hadoop / М. Хороненко, Н. Харитонов, М. Медуницкий, М. Стержанов // BIG DATA и анализ высокого уровня: сборник материалов V Международной научно-практической конференции, Минск, 13–14 марта 2019 г. В 2 ч. Ч. 2 / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2019. – С. 188-193.

11. Deepak V. Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools / V. Deepak. – New York: Apress, 2016. – 165 p.

12. Официальный сайт BaseGroup Labs. Анализ больших объёмов данных [Електронний ресурс]. – Режим доступу: https://basegroup.ru/community/articles/very-large-data/.

References

1. Lvovich, V.D., Anisimov, V.Y., Khoteev, A.L. and Sterzhanov, M.V. (2019), “Vybor yazyka programmirovaniya dlya resheniya zadach, svyazannyh s primeneniem tekhnologii Big Data” [Programming language selection for solving tasks related to Big Data], *BIG DATA and Advanced Analytics: collection of materials of the V International Scientific Practical Conference, Minsk, March 13–14. At 2 parts, Part 2*, Minsk, pp. 117-120.

2. Borovikov, S.M., Dzick, S.K. and Dzick, S.S. (2019), “Bolshie dannye i printsipi razrabotki analiticheskih sistem” [BIG DATA and principles of development of analytical systems], *BIG DATA and Advanced Analytics: collection of materials of the V International Scientific Practical Conference, Minsk, March 13–14. At 2 parts, Part 2*, Minsk, pp. 167-171.

3. Uspenskiy, N. (2019), “Novye podkhody upravleniya dannymi” [New data management approaches], *BIG DATA and Advanced Analytics: collection of materials of the V International Scientific Practical Conference, Minsk, March 13–14. At 2 parts, Part 1*, Minsk, pp. 28-33.

4. Tsyrelchuk, I.N., Shneiderov, E.N., Berashevich, P.A., Los, N.A. and Tereshkov, A.S. (2018), “Raspredelennye failovye sistemy dlya organizatsii khranilishch strukturirovannykh dannykh” [Distributed file systems for structured datastores organization], *BIG DATA and Advanced Analytics: collection of materials of the IV International Scientific Practical Conference, Minsk, May 3–4*, Minsk, pp. 463-466.

5. Khadasevich, A.I. and Shvets, V.I. (2018), “Obrabotka bolshih obemov informatsii s ispolzovaniem platformy Hadoop i sluzhby oblachnuh vycheslenii Microsoft Azure” [Processing large amounts of information using the Hadoop platform and Microsoft Azure cloud computing services], *Computer systems and networks: materials of the 54th scientific conference of graduate students, undergraduates and students, Minsk, April 23-27*, Minsk, pp. 208.

6. Heger, D. (2015), Future of big data, *BIG DATA and Predictive Analytics. Using BIG DATA to optimize business and information technology: a collection of materials of the international scientific-practical conference*, Minsk, pp. 72-75.

7. Karau, H. and Warrens, R. (2018), “*Effektivnyi Spark. Masshtabirovanie i optimizatsiya*” [*Effective Spark. Scaling and optimization*], Piter, St. Petersburg, 352 p.

8. Franks, B. (2014), *Taming the big data. Finding Opportunities in Huge Data Streams with Advanced Analytics*, Wiley, Hoboken, New Jersey, 336 p.

9. Borovikov, S.M., Shneiderov, E.N., Tsyrelchuk, M.I. and Dzik, S.S. (2019), Prediction in Big Data Technology, *BIG DATA and Advanced Analytics. Using BIG DATA to optimize business and information technology: a collection of materials of the II International Scientific and Practical Conference; Minsk, June 15-17*, Minsk, pp. 98-101.

10. Horoneko, M., Haritonov, N., Medunetski, M. and Sterjanov, M. (2019), “Luchshie praktiki razrabotki Big Data prilozhenii na baze Hadoop” [Best practices of Big Data applications development on the base of Hadoop], *BIG DATA and Advanced Analytics: collection of materials of the V International Scientific Practical Conference, Minsk, March 13–14. At 2 parts, Part 2*, Minsk, pp. 188-193.

11. Deepak, V. (2016), *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*, Apress, New York, 165 p.

12. The official site of BaseGroup Labs (2019), “*Analiz bolshih obemov dannyh*” [*Big Data Analysis*], available at: https://basegroup.ru/community/articles/very-large-data/ (accessed 24 March 2019).

Поступила в редколлегию 29.03.2019
Одобрена к печати 21.05.2019

Відомості про авторів:

Савенко Андрій Геннадійович
аспірант магістр технічних наук
старший викладач Інституту інформаційних технологій
Білоруського державного університету інформатики
та радіоелектроніки,
Мінськ, Республіка Білорусь
https://orcid.org/0000-0002-1890-9453

Гавриленко Андрій Сергійович
інженер-програміст ТОВ "ТКП-Софт",
студент Інституту інформаційних технологій Білоруського
державного університету інформатики
та радіоелектроніки,
Мінськ, Республіка Білорусь
https://orcid.org/0000-0001-8154-0583

Information about the authors:

Andrei Savenko
Doctoral Student Master of Technical Sciences
Senior Instructor of the Institute of Information
Technologies of the Belarusian State University
of Informatics and Radioelectronics, .
Minsk, Republic of Belarus
https://orcid.org/0000-0002-1890-9453

Andrei Gavrilenko
Software Engineer of Limited liability company "TKP-Soft",
Student of the Institute of Information Technologies
f the Belarusian State University of Informatics and
Radioelectronics,
Minsk, Republic of Belarus
https://orcid.org/0000-0001-8154-0583

РОЗПОДІЛ НАВАНТАЖЕННЯ ПРИ ПОБУДОВІ ЗВІТІВ ТА ЗАПИТІВ З ВЕЛИКИМ ОБСЯГОМ ДАНИХ

А.Г. Савенко, А.С. Гавриленко

Запропоновано підхід щодо перерозподілу навантаження web-, мобільних- і desktop- клієнт-серверних додатків для збільшення швидкодії при побудові звітів і запитів з великою кількістю даних на технологіях PHP, RabbitMQ, Redis і реляційної базі даних. Запропонований підхід дозволяє значно збільшити швидкодію існуючих, не призначених спочатку для обробки великих даних, додатків, а також розробляти нові клієнт-серверні додатки, призначені для обробки великих обсягів інформації (великої кількості запитів).

Ключові слова: *розподіл навантаження, big data, запити, API, клієнт-серверні додатки.*

LOAD DISTRIBUTION OF BIG DATA REQUEST AND REPORTS

A. Savenko, A. Gavrilenko

An approach is proposed for redistributing the load of web-, mobile- and desktop client-server applications to increase speed big data reports and queries on PHP, RabbitMQ, Redis technologies and a relational database. The following scheme of redistribution of load is proposed. For the API, which initially was responsible for receiving information, processing and issuing it, only the function for receiving information is left. Receiving any request, the API returns the request parameters to the RabbitMQ queue. By launching the next worker, a subscription to the switch is performed, which performs load distribution between subscribers in the following way: if the subscriber is busy at the current moment, the switch will send a message to the one who is free or will wait for the one who is the fastest to be released. Thus, the subscribers perform all the actions that were previously performed by the sole API. Using any page, request, action or function of the project, it is distributed to the worker with the least load. A worker who has performed a specific algorithm of actions sends the answer to a specific queue intended for the execution of algorithms, which is indicated in the message. There may come a time when the workers are loaded with large long-term tasks that interfere with the implementation of small tasks. In order for this not to happen, it is necessary to distinguish between workers. To distinguish between workers, you need to identify small tasks and tasks with large amounts of data, thereby dividing the workers into several groups. The first group of workers will perform big data processing tasks, the second group of workers will perform small tasks, and a third group of workers who need to process reports is also needed. These actions can achieve maximum system acceleration, since workers will be demarcated, and small tasks, I will not wait for workers who process big data. If we confine ourselves to an equal number of workers of different groups, then it is not appropriate to talk about speed, as some workers of some groups will stand idle for some time. To eliminate this situation, you need to distribute the number of workers: for small tasks, it is necessary to allocate a small number of workers (since tasks are performed quickly and do not require much time to answer) and for more complex tasks, it is necessary to allocate a larger number of workers (since they are processed much longer). In order for the servers on which the workers are working to be used only for urgent needs, it is also necessary to use the Redis server as a means to store the cache of the processed results.The proposed approach allows us to significantly increase the speed of existing, not originally intended for processing big data applications, as well as to develop new client-server applications designed to process big data (a large number of requests).

Keywords: *load distribution, big data, requests, API, client-server applications.*