



★ 4.74
Оценка

897.08
Рейтинг

Selectel

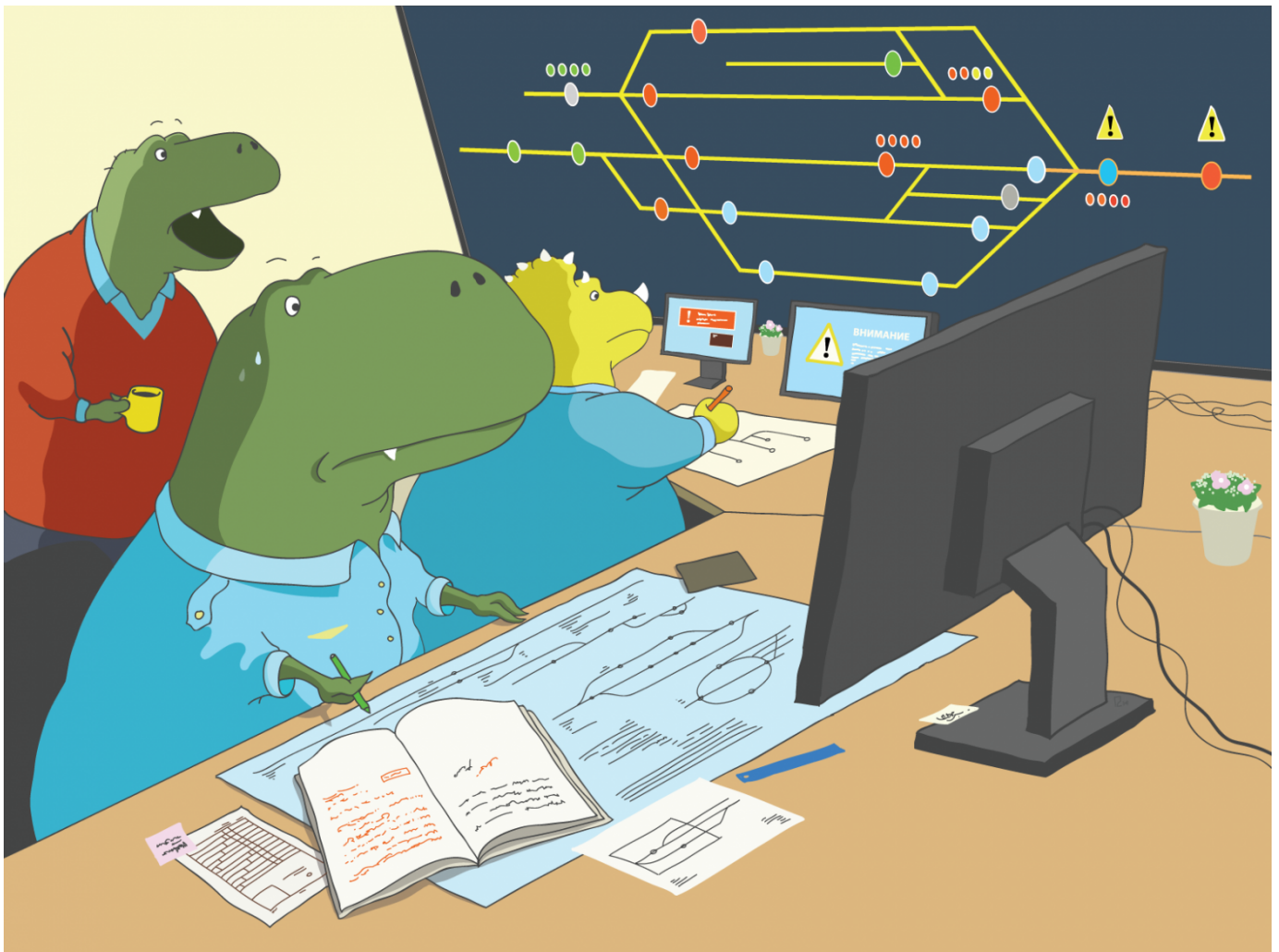
IT-инфраструктура для бизнеса



AndreiYemelianov 10 февраля 2015 в 11:35

Балансировка нагрузки: основные алгоритмы и методы

Блог компании Selectel



Вопрос о планировании нагрузки следует решать ещё на ранней стадии развития любого веб-проекта. «Падение» сервера (а оно всегда происходит неожиданно, в самый неподходящий момент) чревато весьма серьёзными последствиями — как моральными,



+26



123K



533



сервера в связи ростом нагрузок можно решать путем наращивания мощности сервера, или же оптимизацией используемых алгоритмов, программных кодов и так далее. Но рано или поздно наступает момент, когда и эти меры оказываются недостаточными.

Приходится прибегать к кластеризации: несколько серверов объединяются в кластер; нагрузка между ними распределяется при помощи комплекса специальных методов, называемых балансировкой. Помимо решения проблемы высоких нагрузок кластеризация помогает также обеспечить резервирование серверов друг на друга.

Эффективность кластеризации напрямую зависит от того, как распределяется (балансируется) нагрузка между элементами кластера.

Балансировка нагрузки может осуществляться при помощи как аппаратных, так и программных инструментов. Об основных методах и алгоритмах и балансировки мы бы хотели рассказать в этой статье.

Уровни балансировки

Процедура балансировки осуществляется при помощи целого комплекса алгоритмов и методов, соответствующим следующим уровням модели OSI:

- сетевому;
- транспортному;
- прикладному.

Рассмотрим эти уровни более подробно.

Балансировка на сетевом уровне

Балансировка на сетевом уровне предполагает решение следующей задачи: нужно сделать так, чтобы за один конкретный IP-адрес сервера отвечали разные физические машины. Такая балансировка может осуществляться с помощью множества разнообразных способов.

- **DNS-балансировка.** На одно доменное имя выделяется несколько IP-адресов. Сервер, на который будет направлен клиентский запрос, обычно определяется с помощью алгоритма Round Robin (о методах и алгоритмах балансировки будет подробно рассказано ниже).
- **Построение NLB-кластера.** При использовании этого способа серверы объединяются в кластер, состоящий из входных и вычислительных узлов. Распределение нагрузки

осуществляется при помощи специального алгоритма. Используется в решениях от компании Microsoft.

- **Балансировка по IP с использованием дополнительного маршрутизатора.**
- **Балансировка по территориальному признаку** осуществляется путём размещения одинаковых сервисов с одинаковыми адресами в территориально различных регионах Интернета (так работает технология Anycast DNS, о которой мы уже писали). Балансировка по территориальному признаку также используется во многих CDN (см. [интересный пример реализации](#)).

Балансировка на транспортном уровне

Этот вид балансировки является самым простым: клиент обращается к балансировщику, тот перенаправляет запрос одному из серверов, который и будет его обрабатывать. Выбор сервера, на котором будет обрабатываться запрос, может осуществляться в соответствии с самыми разными алгоритмами (об этом ещё пойдёт речь ниже): путём простого кругового перебора, путём выбора наименее загруженного сервера из пула и т.п.

Иногда балансировку на транспортном уровне сложно отличить от балансировки на сетевом уровне. Рассмотрим следующее правило для сетевого фильтра pf в BSD-системах: так, например, формально тут идет речь про балансировку трафика на конкретном порту TCP (пример для сетевого фильтра pf в BSD-системах):

```
web_servers = "{ 10.0.0.10, 10.0.0.11, 10.0.0.13 }"

match in on $ext_if proto tcp to port 80 rdr-to $web_servers round-robin sticky-address
```

Речь в нём идет о балансировке трафика на конкретном порту TCP.

Рассмотрим теперь другой пример:

```
pass in on $int_if from $lan_net \
    route-to { ($ext_if1 $ext_gw1), ($ext_if2 $ext_gw2) }\
    round-robin
```

В этом правиле речь о балансировке исходящего трафика на сетевом уровне. В нём

не указано ни конкретного порта, ни конкретного протокола.

Различие между уровнями балансировки можно объяснить следующим образом.

К сетевому уровню относятся решения, которые не терминируют на себе пользовательские сессии. Они просто перенаправляют трафик и не работают в проксирующем режиме.

На сетевом уровне балансировщик просто решает, на какой сервер передавать пакеты. Сессию с клиентом осуществляет сервер.

На транспортном уровне общение с клиентом замыкается на балансировщике, который работает как прокси. Он взаимодействует с серверами от своего имени, передавая информацию о клиенте в дополнительных данных и заголовках. Таким образом работает, например, популярный программный балансировщик HAProxy.

Балансировка на прикладном уровне

При балансировке на прикладном уровне балансировщик работает в режиме «умного прокси». Он анализирует клиентские запросы и перенаправляет их на разные серверы в зависимости от характера запрашиваемого контента. Так работает, например, веб-сервер Nginx, распределяя запросы между фронтендом и бэкендом. За балансировку в Nginx отвечает модуль Upstream. Более подробно об особенностях балансировки Nginx на основе различных алгоритмов можно прочитать, например, [здесь](#).

В качестве ещё одного примера инструмента балансировки на прикладном уровне можно привести pgpool — промежуточный слой между клиентом и сервером СУБД PostgreSQL. С его помощью можно распределять запросы по серверам баз данных в зависимости от их содержания, например, запросы на чтение будут передаваться на один сервер, а запросы на запись — на другой. Подробнее о pgpool и специфике работы с ним можно почитать [в этой статье](#)).

Алгоритмы и методы балансировки

Существует много различных алгоритмов и методов балансировки нагрузки. Выбирая конкретный алгоритм, нужно исходить, во-первых, из специфики конкретного проекта, а во-вторых — из целей, которые мы планируем достичь.

В числе целей, для достижения которых используется балансировка, нужно выделить следующие:

- **справедливость**: нужно гарантировать, чтобы на обработку каждого запроса выделялись системные ресурсы и не допустить возникновения ситуаций, когда один запрос обрабатывается, а все остальные ждут своей очереди;

- **эффективность**: все серверы, которые обрабатывают запросы, должны быть заняты на 100%; желательно не допускать ситуации, когда один из серверов простаивает в ожидании запросов на обработку (сразу же оговоримся, что в реальной практике эта цель достигается далеко не всегда);
- **сокращение времени выполнения запроса**: нужно обеспечить минимальное время между началом обработки запроса (или его постановкой в очередь на обработку) и его завершения;
- **сокращение времени отклика**: нужно минимизировать время ответа на запрос пользователя.

Очень желательно также, чтобы алгоритм балансировки обладал следующими свойствами:

- **предсказуемость**: нужно чётко понимать, в каких ситуациях и при каких нагрузках алгоритм будет эффективным для решения поставленных задач;
- **равномерная загрузка ресурсов системы**;
- **масштабируемость**: алгоритм должен сохранять работоспособность при увеличении нагрузки.

Round Robin

Round Robin, или алгоритм кругового обслуживания, представляет собой перебор по круговому циклу: первый запрос передаётся одному серверу, затем следующий запрос передаётся другому и так до достижения последнего сервера, а затем всё начинается сначала.

Самой распространённой имплементацией этого алгоритма является, конечно же, метод балансировки Round Robin DNS. Как известно, любой DNS-сервер хранит пару «имя хоста — IP-адрес» для каждой машины в определённом домене. Этот список может выглядеть, например, так:

```
example.com xxx.xxx.xxx.2  
www.example.com xxx.xxx.xxx.3
```

С каждым именем из списка можно ассоциировать несколько IP-адресов:

```
example.com xxx.xxx.xxx.2
www.example.com xxx.xxx.xxx.3
www.example.com xxx.xxx.xxx.4
www.example.com xxx.xxx.xxx.5
www.example.com xxx.xxx.xxx.6
```

DNS-сервер проходит по всем записям таблицы и отдаёт на каждый новый запрос следующий IP-адрес: например, на первый запрос — xxx.xxx.xxx.2, на второй — xxx.xxx.xxx.3, и так далее. В результате все серверы в кластере получают одинаковое количество запросов.

В числе несомненных плюсов этого алгоритма следует назвать, во-первых, независимость от протокола высокого уровня. Для работы по алгоритму Round Robin используется любой протокол, в котором обращение к серверу идёт по имени.

Балансировка на основе алгоритма Round Robin никак не зависит от нагрузки на сервер: кэширующие DNS-серверы помогут справиться с любым наплывом клиентов.

Использование алгоритма Round Robin не требует связи между серверами, поэтому он может использоваться как для локальной, так и для глобальной балансировки. Наконец, решения на базе алгоритма Round Robin отличаются низкой стоимостью: чтобы они начали работать, достаточно просто добавить несколько записей в DNS.

Алгоритм Round Robin имеет и целый ряд существенных недостатков. Чтобы распределение нагрузки по этому алгоритму отвечало упомянутым выше критериями справедливости и эффективности, нужно, чтобы у каждого сервера был в наличии одинаковый набор ресурсов. При выполнении всех операций также должно быть задействовано одинаковое количество ресурсов. В реальной практике эти условия в большинстве случаев оказываются невыполнимыми.

Также при балансировке по алгоритму Round Robin совершенно не учитывается загруженность того или иного сервера в составе кластера. Представим себе следующую гипотетическую ситуацию: один из узлов загружен на 100%, в то время как другие — всего на 10 — 15%. Алгоритм Round Robin возможности возникновения такой ситуации не учитывает в принципе, поэтому перегруженный узел все равно будет получать запросы. Ни о какой справедливости, эффективности и предсказуемости в таком случае не может быть и речи.

В силу описанных выше обстоятельств сфера применения алгоритма Round Robin весьма

ограничена.

Weighted Round Robin

Это — усовершенствованная версия алгоритма Round Robin. Суть усовершенствований заключается в следующем: каждому серверу присваивается весовой коэффициент в соответствии с его производительностью и мощностью. Это помогает распределять нагрузку более гибко: серверы с большим весом обрабатывают больше запросов. Однако всех проблем с отказоустойчивостью это отнюдь не решает. Более эффективную балансировку обеспечивают другие методы, в которых при планировании и распределении нагрузки учитывается большее количество параметров.

Least Connections

В предыдущем разделе мы перечислили основные недостатки алгоритма Round Robin. Назовём ещё один: в нём совершенно не учитывается количество активных на данный момент подключений.

Рассмотрим практический пример. Имеется два сервера — обозначим их условно как А и Б. К серверу А подключено меньше пользователей, чем к серверу Б. При этом сервер А оказывается более перегруженным. Как это возможно? Ответ достаточно прост: подключения к серверу А поддерживаются в течение более долгого времени по сравнению с подключениями к серверу Б.

Описанную проблему можно решить с помощью алгоритма, известного под названием *least connections* (сокращённо — *leastconn*). Он учитывает количество подключений, поддерживаемых серверами в текущий момент времени. Каждый следующий вопрос передаётся серверу с наименьшим количеством активных подключений.

Существует усовершенствованный вариант этого алгоритма, предназначенный в первую очередь для использования в кластерах, состоящих из серверов с разными техническими характеристиками и разной производительностью. Он называется *Weighted Least Connections* и учитывает при распределении нагрузки не только количество активных подключений, но и весовой коэффициент серверов.

В числе других усовершенствованных вариантов алгоритма *Least Connections* следует прежде всего выделить *Locality-Based Least Connection Scheduling* и *Locality-Based Least Connection Scheduling with Replication Scheduling*.

Первый метод был создан специально для кэширующих прокси-серверов. Его суть заключается в следующем: наибольшее количество запросов передаётся серверам с наименьшим количеством активных подключений. За каждым из клиентских серверов закрепляется группа клиентских IP. Запросы с этих IP направляются на «родной» сервер,

если он не загружен полностью. В противном случае запрос будет перенаправлен на другой сервер (он должен быть загружен менее чем наполовину).

В алгоритме Locality-Based Least Connection Scheduling with Replication Scheduling каждый IP-адрес или группа IP-адресов закрепляется не за отдельным сервером, а за целой группой серверов. Запрос передаётся наименее загруженному серверу из группы. Если же все серверы из «родной» группы перегружены, то будет зарезервирован новый сервер. Этот новый сервер будет добавлен к группе, обслуживающей IP, с которого был отправлен запрос. В свою очередь наиболее загруженный сервер из этой группы будет удалён — это позволяет избежать избыточной репликации.

Destination Hash Scheduling и Source Hash Scheduling

Алгоритм Destination Hash Scheduling был создан для работы с кластером кэширующих прокси-серверов, но он часто используется и в других случаях. В этом алгоритме сервер, обрабатывающий запрос, выбирается из статической таблицы по IP-адресу получателя.

Алгоритм Source Hash Scheduling основывается на тех же самых принципах, что и предыдущий, только сервер, который будет обрабатывать запрос, выбирается из таблицы по IP-адресу отправителя.

Sticky Sessions

Sticky Sessions — алгоритм распределения входящих запросов, при котором соединения передаются на один и тот же сервер группы. Он используется, например, в веб-сервере Nginx. Сессии пользователя могут быть закреплены за конкретным сервером с помощью метода IP hash (подробную информацию о нём см. в [официальной документации](#)). С помощью этого метода запросы распределяются по серверам на основе IP-адреса клиента. Как указано в документации (см. ссылку выше), «метод гарантирует, что запросы одного и того же клиента будет передаваться на один и тот же сервер». Если закреплённый за конкретным адресом сервер недоступен, запрос будет перенаправлен на другой сервер. Пример фрагмента конфигурационного файла:

```
upstream backend {  
    ip_hash;  
  
    server backend1.example.com;  
    server backend2.example.com;  
    server backend3.example.com;  
    server backend4.example.com;  
}
```


Начиная с версии 1.2.2 в Nginx для каждого сервера можно указывать вес.

Применение этого метода сопряжено с некоторыми проблемами. Проблемы с привязкой сессий могут возникнуть, если клиент использует динамический IP. В ситуации, когда большое количество запросов проходит через один прокси-сервер, балансировку вряд ли можно назвать эффективной и справедливой. Описанные проблемы, однако, можно решить, используя cookies. В коммерческой версии Nginx имеется специальный модуль sticky, который как раз использует cookies для балансировки. Есть у него и бесплатные аналоги — например, [nginx-sticky-module](#).

Можно использовать метод sticky-sessions и в HAProxy — подробнее об этом можно прочитать, например, [здесь](#).

Заключение

Эта статья по сути представляет собой введение в проблематику балансировки нагрузки. Обсуждение этой темы мы продолжим и в дальнейших публикациях. Если у вас есть вопросы, замечания и дополнения — добро пожаловать в комментарии. Будем также признательны, если вы поделитесь нетривиальными практическими примерами организации балансировки нагрузки для различных проектов.

Читателей, которые по тем или иным причинам не могут оставлять комментарии здесь, приглашаем в наш блог.

Теги: балансировка нагрузки, load balancing, селектел, selectel

Хабы: Блог компании Selectel

Редакторский дайджест

Присылаем лучшие статьи раз в месяц



Selectel

IT-инфраструктура для бизнеса

[Facebook](#) [ВКонтакте](#) [Telegram](#)

**140**

Карма

0

Рейтинг

Андрей Емельянов @AndreiYemelianov

Пользователь

Комментарии **15**

А у вас уже появились IP-адреса с возможностью перебрасывания их между локациями)?

Мы постоянно модернизируем нашу сеть и как раз работаем над внедрением новых сервисов. Можно вас попросить более подробно рассказать о том, что вам требуется?

«Elastic IP», который можно перебрасывать между двумя датацентрами, независимыми по питанию и аплинкам. Вроде hetzner-овского динамического IP, но в России. Вообще прям щас вряд ли надо, но осенью пригодились бы в связи с новым ФЗ =)

Если вас не затруднит, напишите на sales@selectel.ru поподробнее, что вам требуется. Если вы уже наш клиент, то можем даже попробовать организовать тестирование.

А что за новый ФЗ?

Ну он не то, чтобы новый — я о хранении ПД на российских серверах.

Алгоритмы довольно простые, теоретики Computer Science еще не добрались до этой области?

Это описание у них простое. Легко сказать «сервер выбирается из статической таблицы», вопрос в том, как эту статическую таблицу можно оптимально реализовать.

Все серверы, которые обрабатывают запросы, должны быть заняты на 100%; желательно не допускать ситуации, когда один из серверов простаивает в ожидании запросов на обработку.

Скорее «нагрузка между серверами должна быть распределена равномерно». Желательно не допускать ситуации, когда серверы нагружены до 100%.

В подавляющем большинстве случаев загрузка серверов на 100% означает, что вы не сможете обслуживать некоторые запросы пользователей, то есть будете иметь меньший «эффективный аптайм», чем 100%, и очень недовольных пользователей. Поэтому я бы не стал бы формулировать так мягко :).

Стоит добавить к недостаткам DNS Round Robin метода: низкая степень контроля за исполнителями (кэши DNS, игнорирование дополнительных адресов со стороны клиента и т.д.)

Вопрос. Как связаны балансировка нагрузки в кластерах и теория расписаний?

НЛО прилетело и опубликовало эту надпись здесь

Проблема в том, что часто старт проекта начинается с неясности, какова будет посещаемость и как впоследствии поменяются требования. Можно с нуля потратить кучу времени и ресурсов на масштабируемость и получить мёртвый проект, ибо более шустрые конкуренты уже отжали долю рынка, получили фидбэк и перепиливают архитектуру своих проектов на полученные деньги. Можно, таким же образом обеспечив масштабируемость, получить новые требования, из-за чего всё придётся переделывать. В общем, сколько водки ни бери, всё равно два раза бегать делать.

Про IPv6 и DNS rr и её тонкости ни слова.

Только полноправные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

ПОХОЖИЕ ПУБЛИКАЦИИ

22 июня 2021 в 14:10

Отказоустойчивый балансировщик нагрузки: как работает и где протестировать бесплатно

◆ +33

👁 1.1K

🔖 4

💬 0

16 октября 2015 в 13:30

Удалённый узел K-root в Селектеле

◆ +19

👁 7.9K

🔖 30

💬 7 +7

19 декабря 2013 в 16:47

Как устроена волоконно-оптическая сеть Селектел

◆ +25

👁 21K

🔖 33

💬 10 +10

ВАКАНСИИ КОМПАНИИ «SELECTEL»

Team Lead в команду облачного хранилища (Golang)

Selectel · Москва · Можно удаленно

Руководитель отдела знаний

Selectel · Москва

Python-разработчик в команду Compute

Selectel · Москва · Можно удаленно

UX проектировщик в команду IAM

Selectel · Санкт-Петербург · Можно удаленно

Системный администратор в команду SRE ядра облака

Selectel · Санкт-Петербург · Можно удаленно

Больше вакансий на Хабр Карьере

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 12:31

«За границей»: кратко о бюджетных вариантах

 +84 33K 162 75 +75

сегодня в 00:12

Программирование необычных шахмат

 +33 2.5K 31 4 +4

вчера в 13:00

Тим Бёрнерс-Ли и день WWW: 33 года всемирной паутине

 +19 1.7K 15 2 +2

вчера в 19:27

Термоядерный синтез [своими руками]

 +16 7.8K 15 11 +11

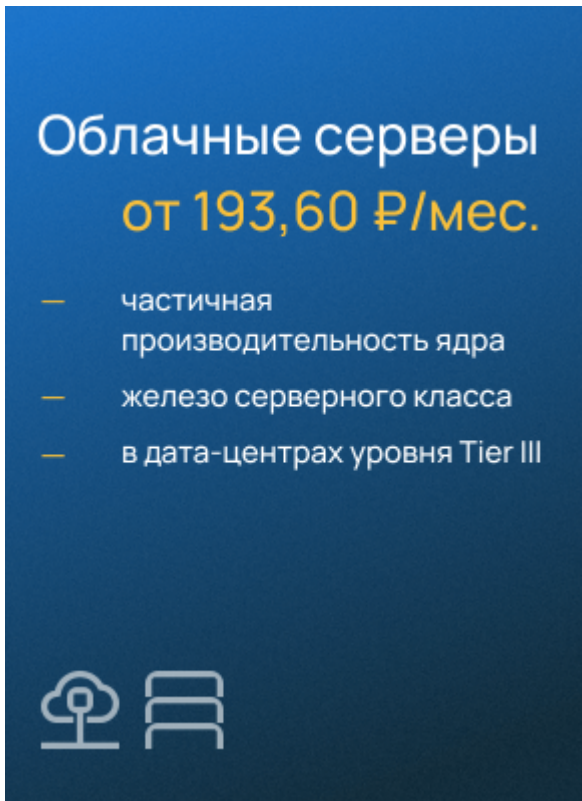
вчера в 17:04

Психолог в IT-компании: этический вопрос

ИНФОРМАЦИЯ



Дата основания	11 сентября 2007
Местоположение	Россия
Сайт	selectel.ru
Численность	501–1 000 человек
Дата регистрации	15 марта 2010

ВИДЖЕТ



Облачные серверы
от 193,60 ₽/мес.

- частичная производительность ядра
- железо серверного класса
- в дата-центрах уровня Tier III

ССЫЛКИ

Новостной канал в Telegram
[slc.tl](https://t.me/slc_tl)

Панель управления
slc.tl

Серверы по выгодной цене

slc.tl

YouTube-канал

slc.tl

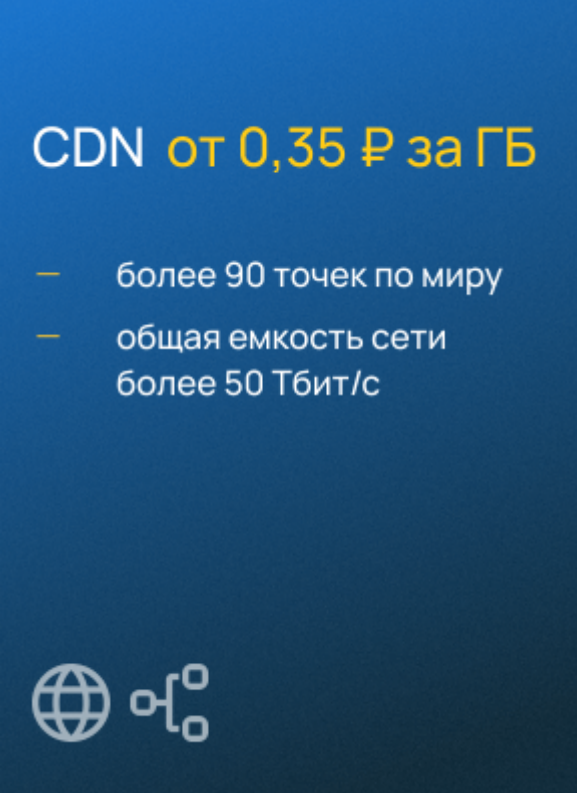
База знаний

slc.tl

Облачная платформа

slc.tl


ВИДЖЕТ

A blue rectangular widget with a gradient background. At the top, the text "CDN от 0,35 ₽ за ГБ" is displayed, with "от 0,35 ₽ за ГБ" in yellow. Below this, there are two bullet points, each preceded by a yellow dash: "более 90 точек по миру" and "общая емкость сети более 50 Тбит/с". At the bottom left, there is a white icon consisting of a globe and a network diagram with three nodes.

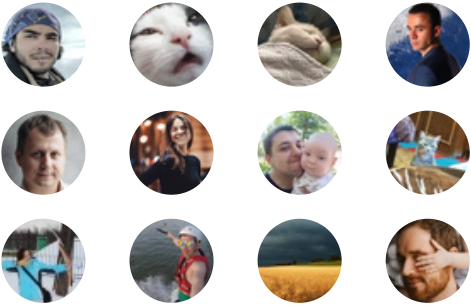
CDN от 0,35 ₽ за ГБ

- более 90 точек по миру
- общая емкость сети более 50 Тбит/с

В КОНТАКТЕ

 **Selectel**

66 502 читателя



Подписаться на новости

БЛОГ НА ХАБРЕ

сегодня в 03:27

И еще одна эргономическая кастомная клавиатура: встречаем ErgoDox 76 'Hot Dox' V2

 1.7K  1 +1

10 марта в 05:04

Мини-ПК начала 2022 года, на которые стоит обратить внимание

 11K  25 +25

7 марта в 04:45

Лучшие проекты на Raspberry Pi начала 2022 года, на которые стоит обратить внимание

 19K  21 +21

5 марта в 05:20

Ваш аккаунт	Разделы	Информация	Услуги
Войти	Публикации	Устройство сайта	Реклама
Регистрация	Новости	Для авторов	Тарифы
	Хабы	Для компаний	Контент
	Компании	Документы	Семинары
	Авторы	Соглашение	Мегапроекты
	Песочница	Конфиденциальность	



[Настройка языка](#)

[О сайте](#)

[Техническая поддержка](#)

[Вернуться на старую версию](#)

© 2006–2022 «Habr»