Технічні науки

УДК 519.6, 004.7

Якимець Роман Вікторович
студент
Національний технічний університет України
«Київський політехнічний інститут»
Якимец Роман Викторович
студент
Национальный технический университет Украины
«Киевский политехнический институт»
Yakymets Roman V.
Student
National Technical University of Ukraine «Kyiv Polytechnic Institute»
Яременко Костянтин Миколайович
студент
Національний технічний університет України
«Київський політехнічний інститут»
Яременко Константин Николаевич
студент
Национальный технический университет Украины
«Киевский политехнический институт»
Yaremenko K. N.
Student
National Technical University of Ukraine «Kyiv Polytechnic Institute»

МАСШТАБУВАННЯ НАВАНТАЖЕННЯ WEB-ДОДАТКІВ
МАСШТАБИРОВАНИЕ НАГРУЗКИ WEB-ПРИЛОЖЕНИЙ
SCALING OF LOAD WEB-APPLICATIONS

Анотація: Досліджені принципи зменшення навантаження на web-додаток шляхом його оптимізації та масштабування. Розглянуті методи масштабування та способи їх застосування.

Ключові слова: зменшення навантаження, оптимізація, масштабування, web-додаток, хост.

Аннотация: Исследованы принципы уменьшения нагрузки на web-приложение путем его оптимизации и масштабирования. Рассмотрены методы масштабирования и способы их применения.

Ключевые слова: уменьшение нагрузки, оптимизация, масштабирование, web-приложение, хост.

Summary: Researched principles of reduce the load on web-application through optimization and scaling. There was considered the methods of scaling and methods of their application.

Key words: load reduction, optimization, scale, web-application, host.

Вступ

Із зростанням популярності web-додатку його підтримка неминуче починає вимагати все більших і більших ресурсів. Перший час з навантаженням можна боротися шляхом оптимізації алгоритмів і архітектури самого додатка. Однак якщо все, що можна було оптимізувати, вже оптимізовано, а додаток все одно не справляється з навантаженням потрібно масштабувати web-додаток.

Оптимізація

Насамперед необхідно перевірити наступні положення:

- Оптимальність запитів до БД.
- Правильність збереження даних в БД.
- Використання кешування.
- Відсутність зайвих запитів до ФС чи БД.
- Оптимальність алгоритмів обробки даних.

Про кожен з цих пунктів можна написати окрему статтю, так що їх детальний розгляд в рамках даної статті явно надлишковий. Важливо лише розуміти, що перед тим як приступити до масштабування додатку, вкрай бажано максимально оптимізувати його роботу – адже можливо тоді ніякого масштабування і не буде потрібно.

Масштабування

Припустимо, що оптимізація вже проведена, але додаток все одно не справляється з навантаженням. В такому випадку рішенням проблеми, може бути його розподілення на декілька хостів, з метою збільшення загальної продуктивності додатку за рахунок збільшення доступних ресурсів. Такий підхід має офіційну назву - «масштабування» (scale) додатку. Точніше кажучи, під «масштабністю» (scalability) називається можливість системи збільшувати свою продуктивність при збільшенні кількості виділених для неї ресурсів. Розрізняють два способи масштабування: вертикальний і горизонтальний. Вертикальне масштабування має на увазі збільшення продуктивності додатка при додаванні ресурсів (процесора, пам’яті, диска) в рамках одного вузла (хоста). Горизонтальне масштабування характерне для розподілених додатків і має на увазі зростання продуктивності додатка при додаванні ще одного вузла (хоста).

Найпростішим способом буде просте оновлення заліза (процесора, пам’яті, диска) – тобто вертикальне масштабування. Крім того, цей підхід не вимагає ніяких доопрацювань додатка. Однак, вертикальне масштабування дуже швидко досягає своєї межі, після чого розробнику та адміністратору нічого не залишається окрім як перейти до горизонтального масштабування додатки.

Архітектура програми

Більшість web-додатків є розподіленими, так як в їх архітектурі можна виділити мінімум три шари: web-сервер, бізнес-логіка, дані (БД, статика).

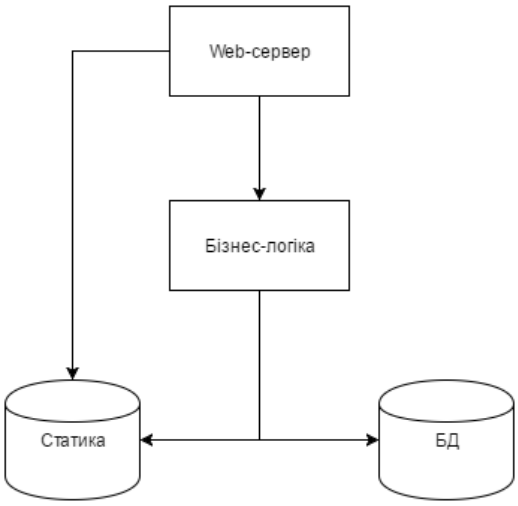

Рисунок 1 Архітектура програми

Кожен з цих шарів може бути масштабований. Тому якщо у системі бізнес-логіка і БД живуть на одному хості – першим кроком повинно стати розподілення їх на різні хости.

Вузьке місце

Приступаючи до масштабування системи, перш за все варто визначити, який з шарів є «вузьким місцем» - тобто працює повільніше за решту системи. Для початку можна скористатися банальними утилітами типу top (htop) для оцінки споживання процесора/пам’яті і df, iostat для оцінки споживання диска. Однак, бажано виділити окремий хост, з емуляцією бойового навантаження (з допомогою AB або Jmeter), на якому можна буде профілювати роботу програми за допомогою таких утиліт як xdebug, oprofile і так далі. Для виявлення вузьких запитів до БД можна скористатися утилітами типу pgFouine.

Зазвичай все залежить від архітектури додатку, але найбільш ймовірними кандидатами на «вузьке місце» в загальному випадку є БД і код. Якщо додаток працює з великим об’ємом призначених для

користувача даних, то «вузьким місцем», відповідно, швидше за все буде зберігання статики.

Масштабування БД

Як вже говорилося вище, найчастіше вузьким місцем в сучасних додатках є БД. Проблеми з нею діляться, як правило, на два класи: продуктивність і необхідність зберігання великої кількості даних.

Знизити навантаження на БД можна розділивши її на кілька хостів. При цьому гостро встає проблема синхронізації між ними, вирішити яку можна шляхом реалізації схеми master/slave з синхронної або асинхронної реплікацією. У випадку з PostgreSQL реалізувати синхронну реплікацію можна за допомогою Slony-I, асинхронну - PGPool-II або WAL (9.0). Вирішити проблему поділу запитів читання і запису, а так само балансування навантаження між наявними slave'амі, можна за допомогою настройки спеціального шару доступу до БД (PGPool-II).

Проблему зберігання великого обсягу даних в разі використання реляційних СУБД можна вирішити за допомогою механізму партиціонування ("поділ" в PostgreSQL), або розгортаючи БД на розподілених ФС типу Hadoop DFS.

Проте, для зберігання великих обсягів даних найкращим рішенням буде «шардінг» (Sharding) даних, який є вбудованим перевагою більшості NoSQL БД (наприклад, MongoDB).

Крім того, NoSQL БД в загальному працюють швидше своїх SQL-братів за рахунок відсутності overhead'а на розбір / оптимізацію запиту, перевірки цілісності структури даних і т.д.

Окремо варто відзначити досвід Facebook, який використовують MySQL без JOIN-вибірок. Така стратегія дозволяє їм значно легше масштабувати БД, переносячи при цьому навантаження з БД на код , масштабується простіше БД.

Масштабування коду

Складнощі з масштабуванням коду залежать від того, скільки розподілених ресурсів необхідно хостам для роботи вашої програми. Чи будуть це тільки сесії, або буде потрібно загальний кеш і файли? У будь-якому випадку в першу чергу потрібно запустити копії програми на декількох хостах з однаковим оточенням.

Далі необхідно налаштувати балансування навантаження / запитів між цими хостами. Зробити це можна як на рівні TCP (HAProxy), так і на HTTP (Nginx) або DNS.

Наступним кроком потрібно зробити так, що б файли статики, кеш і сесії веб-додатки були доступні на кожному хості. Для сесій можна використовувати сервер, який працює через мережу (наприклад, Memcached). Як сервер кеша цілком розумно використовувати той же Memcached, але на іншому хості.

Файли статики можна змонтувати з якогось загального файлового сховища по NFS / CIFS або використовувати розподілену (HDFS ФС, GlusterFS, Ceph).

Так само можна зберігати файли в БД (наприклад, Mongo GridFS), вирішуючи тим самим проблеми доступності та масштабованості (з урахуванням того, що для NoSQL БД проблема масштабованості вирішена за рахунок шардінга).

Окремо варто відзначити проблему розміщення на кілька хостів. Найпростішим рішенням того щоб споживач, натискаючи «Оновити», не бачив різні версії програми буде виключення з конфігураційного файла балансувальника навантаження (веб-сервера) не оновлених хостів, і послідовного їх включення в міру оновлення. Так само можна прив'язати користувачів до конкретних хостам по куки або IP. Якщо ж оновлення вимагає значних змін у БД, найпростіше, взагалі тимчасово закрити проект.

Масштабування ФС

При необхідності зберігання великого обсягу статики можна виділити дві проблеми: брак місця і швидкість доступу до даних. Як вже було написано вище, проблему з нестачею місця можна вирішити як мінімум трьома шляхами: розподілена ФС, зберігання даних в БД з підтримкою шардінга і організація шардінга «вручну» на рівні коду.

При цьому варто розуміти, що роздача статики теж не найпростіше завдання, коли мова йде про високі навантаження. Тому в цілком резонно мати безліч серверів призначених для роздачі статики. При цьому, якщо ми маємо загальне сховище даних (розподілена ФС або БД), при збереженні файлу ми можемо зберігати його ім'я без урахування хоста, а ім'я хоста підставляти випадковим чином при формуванні сторінки. У разі, коли шардінг реалізується вручну (тобто, за вибір хоста, на який будуть залиті дані, відповідає логіка в коді), інформація про хоста заливки повинна або обчислюватися на основі самого файлу, або генеруватися на підставі інших даних (інформація про користувача, кількості місця на дисках-сховищах) і зберігатися разом з ім'ям файлу в БД.

Висновок

В статті коротко розглянуто безліч варіантів рішень проблем масштабування веб-додатків. Кожен з них має свої переваги і недоліки. Не існує деякого рецепта, як зробити все добре і відразу - для кожного завдання знайдеться безліч рішень зі своїми плюсами і мінусами.

Література:

1. John H. Howard. Scale and performance in a distributed file system/ John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. ACM Transactions on Computer Systems, 1988.
2. Title:RESTful Web Services By: Leonard Richardson, Sam Ruby Publisher:O'Reilly Media May , 2007 P. 454

3. Clifford Neuman .Scale in Distributed Systems, 1994
4. Martin Kalin Java Web Services: Up and Running, 2nd Edition, O'Reilly, August 2013, P. 360