Application Cache

Introduction

Application cache gives an application three advantages:

1. Offline browsing - users can use the application when they're offline
2. Speed - cached resources load faster
3. Reduced server load - the browser will only download updated/changed resources from the server

Varnish

Introduction

Varnish Cache is a web application accelerator also known as a caching HTTP reverse proxy. You install it in front of any server that speaks HTTP and configure it to cache the contents. Varnish Cache is really, really fast. It typically speeds up delivery with a factor of 300 - 1000x, depending on your architecture.

Setup

```
# update packages
$ sudo apt-get update
$ sudo apt-get upgrade
# install apache
$ sudo apt-get install apache2
# install varnish
$ sudo apt-get install apt-transport-https
$ sudo curl https://repo.varnish-cache.org/ubuntu/GPG-key.txt | apt-key add -
$ echo "deb https://repo.varnish-cache.org/ubuntu/ trusty varnish-4.0" >>
/etc/apt/sources.list.d/varnish-cache.list
$ sudo apt-get update
$ sudo apt-get install varnish
```

Configure

```
$ sudo vi /etc/apache2/ports.conf
Listen 127.0.0.1:8080

$ sudo vi /etc/apache2/sites-available/000-default.conf
<VirtualHost 127.0.0.1:8080>

$ sudo vi /etc/default/varnish
DAEMON_OPTS="-a :80 \
```

```
        -T localhost:6082 \
        -f /etc/varnish/default.vcl \
        -S /etc/varnish/secret \
        -s malloc,256m"
$ sudo nano /etc/varnish/default.vcl
backend default {
   .host = "127.0.0.1";
   .port = "8080";
}

$ sudo service apache2 restart
$ sudo service varnish restart
$ sudo service varnishd status  # HTTP accelerator daemon
```

Testing

```
$ curl -I localhost
 HTTP/1.1 403 Forbidden
 Date: Mon, 02 Jun 2015 24:06:10 GMT
 Server: Apache/2.4.6 (Ubuntu) PHP/5.4.16
 Last-Modified: Thu, 16 Dec 2014 19:30:58 GMT
 ETag: "1321-5058a1e728280"
 Accept-Ranges: bytes
 Content-Length: 4897
 Content-Type: text/html; charset=UTF-8
 X-Varnish: 32779
 Age: 0
 Via: 1.1 varnish-v4
 Connection: keep-alive
```

Redis

Introduction

- Redis is a super fast non-relational database that uses keys to map to different data types.
- It is a NoSQL **(In-Memory Databases)** solution which roughly means that there are no tables or relationships between the data you store.
- Like memcached, it is in-memory storage which is why it's so incredibly fast.Unlike memcached though, it allows more complex data structures giving it the ability to handle many different problem sets.

Setup

$ sudo apt-get install redis-server $ sudo redis-server --version
Installing redis-server will give you access to the following commands: ● **redis-server:** The running Redis instance. ● **redis-sentinel:** Redis Sentinel executable (monitoring and failover stuff) ● **redis-cli:** The command line interface for interacting with Redis. ● **redis-benchmark:** Analyze Redis performance.
$ redis-cli redis 127.0.0.1:6379> Note: default Redis port of 6379 Quick test command to ensure that the cli is working for us : redis 127.0.0.1:6379> ping PONG redis 127.0.0.1:6379> CONFIG GET * # To get all keys redis 127.0.0.1:6379> monitor * # monitor server redis 127.0.0.1:6379> set <key> <value> | redis 127.0.0.1:6379> get <key>

Redis Data Types
Strings | *Lists* | *Sets* | *Hashes* | *Sorted Sets*

Ref:

RabbitMQ

Introduction
Message broker server built on the Advanced Message Queuing Protocol (AMQP). RabbitMQ is written in Erlang. It's responsible queuing up tasks and scheduling them.

Setup

# to install rabbitmq server 	
$ sudo apt-get install rabbitmq-server
 	
$ sudo rabbitmqctl status # to check version/status
 	
$ sudo rabbitmqctl stop # to stop rabbitmq
 	
$ sudo invoke-rc.d rabbitmq-server start # to start rabbitmq
	# to install Celery
	$ mkdir ~/TEST/MQ
	$ cd ~/TEST/MQ
	
	$ sudo apt-get update
	$ sudo pip install celery
	
	$ celery --version
	3.1.13 (Cipater)
	$ which celery o/p: /usr/local/bin/celery
	
	# amqp.node client library
	npm install amqplib

Example Using javascript

send.js

```
#!/usr/bin/env node
var amqp = require('amqplib/callback_api');
amqp.connect('amqp://localhost', function(err, conn) {
  conn.createChannel(function(err, ch) {
    var q = 'hello';
    var msg = 'Hello World!';
    ch.assertQueue(q, {durable: false});
    // Note: on Node 6 Buffer.from(msg) should be used
    ch.sendToQueue(q, new Buffer(msg));
    console.log(" [x] Sent %s", msg);
  });
  setTimeout(function() { conn.close(); process.exit(0) }, 500);
});
```

```
# to run sender
$ chmod +x send.js
$ ./send.js
```

receive.js

```
#!/usr/bin/env node
var amqp = require('amqplib/callback_api');
amqp.connect('amqp://localhost', function(err, conn) {
  conn.createChannel(function(err, ch) {
    var q = 'hello';
    ch.assertQueue(q, {durable: false});
    console.log(" [*] Waiting for messages in %s. To exit press CTRL+C", q);
    ch.consume(q, function(msg) {
      console.log(" [x] Received %s", msg.content.toString());
    }, {noAck: true});
  });
});
```

```
# to run receiver
$ chmod +x receiver.js
$ ./receiver.js
```

References

https://varnish-cache.org/

http://try.redis.io/

https://www.rabbitmq.com/