Nginx docker image for easy file access via HTTP

Description	An alternative for SSHFS, Samba shares etc.
Period	Spring 2016
Languages & Libs	Bash
Tags	Docker, Spark, Nginx, GitHub
GitHub	nikonyrh/docker-scripts
DockerHub	nikonyrh/nginx_bridge

Often I find myself having a SSH connection to a remote server, and I'd like to retrieve some files to my own machine. Common methods for this include Windows/Samba share, SSHFS and upload to cloud (which isn't trivial to do via plain cURL). Here an easy-to-use alternative is described: a single line command to load and run a docker image which contains a pre-configured Nginx instance. Then files can be accessed via plain HTTP at the user-assigned port (assuming firewall isn't blocking it).

I found that writing Dockerfiles is way easier than for example make files, maybe because its operations closely match those you'd execute via bash anyway when setting up a new box. Additionally there are convenient published images to base your images on, thus minimizing the number of custom steps you need to think of and implement.

The implemented docker image is based `"FROM nginx:1.9"`, and just contains a custom `nginx.conf` and `main.sh` files. When `docker run -p 1234:80 -v "$PWD:/volume" -d nikonyrh/nginx_bridge` is executed it starts the container, mounts current working directory to `/volume` path (could be read-only) and exposes its contents as Nginx auto-indexed folder at `http://localhost:1234`. By default access log is available at `http://localhost:1234/logs/logx.txt` but it can be disabled with `-no-log` flag at start-up. The image is about 190 MB, gzip compresses it down to 72 MB and Dockerhub says it is 75 MB.

Also some efficiency experiments were run. A few gigabytes of JPG images (40 - 400 kB in size) were transferred and at best 90% of the 1 Gbps bandwidth was achieved. Files were transferred from an Ubuntu server to a router, to a Windows machine running Ubuntu in a VirtualBox, via `curl` to `/dev/null`. Resulting bandwidth is shown in Figure 1. Parallel execution was achieved via `xargs`, thus the overhead of three-way TCP handshake was significant unless the cURL processes fetched multiple images.

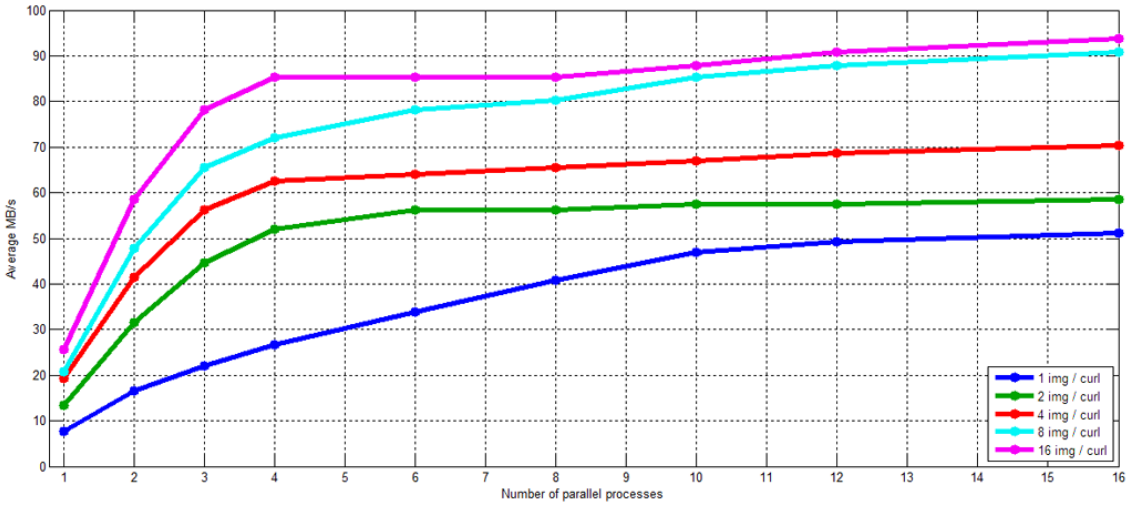

Figure 1: Achieved bandwidth on transferring medium-size files over 1 Gb etherned and HTTP.

In conclusion this seems to be a viable way of distributing JPG images to other machines within the LAN for further processing. The first task might be calculating color histograms of webcam images on different calendar dates and times of day. Calculation distribution will be handled by the Spark framework. Also it would be interesting to measure this performance to alternatives such as HDFS.