Table of Contents

Spigot Minecraft server on a RASPBERRY PI Model B+

NOTE: *On a Raspberry Pi 3 you can just run a Raspbian jessie, install some java runtime and get the BuildTools.jar to build and run Spigot right on the Raspberry Pi.*

This page will describe a possible way of setting up and running a Spigot 1.8 Minecraft server on a RASPBERRY PI Model B+. Altogether, this enables one to run a very low-cost, decent performing Minecraft server for 2-3 users at home. It even seems to be possible to run a small list of plugins on it, namely:

- ClearLag
- AutoSaveWorld
- Dynmap (Dynmap Support Thread)
- NoSpawnChunks
- WorldBorder
- Multiverse-Core

The document is aimed at people with a decent understanding of computers and linux and is in no way a step-by-step guide or intended for beginners. I just assume that any detail or issue that isn't obvious in this howto is either clear to you or can be resolved by using Google.

Running the Vanilla 1.8 server just won't work on the RASPBERRY PI. Even without any connected users it is unable to keep up ticks and crashes after a while.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. The author shall have no liability for any error or damages of any kind resulting from the use of this document. There is no warranty; not even for merchantability or fitness for a particular purpose.

Prerequisites

Hardware

Obviously, you'll need a RASPBERRY PI Model B+ (i have not the faintest clue wether or not this will work on the B or the A model). It is a good idea to shoot for some copper heatsinks too. I also recommend getting a fast Class 10 MicroSDHC card, they're dirt cheap, so go for a 32GB sized. I used, despite http://elinux.org/RPi_SD_cards#SD_card_performance, the 'Samsung Memory 32GB EVO MicroSDHC UHS-I Grade 1 Class 10' which works without any trouble at all. It did not go bad once on a 2-day hacking marathon with lots of overclock-finetuning and benchmarking, i suppose a bad vendor or product would have shown signs of degradation already within that time frame. You'll need a stable power supply too, it absolutely must be able to provide at least 2000mA and of course 5V at all times.

Having a nice case for your RASPBERRY PI wont hurt, too. You'll need a network cable of course, but i assume you have that somewhere.

The RASPBERRY PI Model B+, copper heatsinks, MicroSDHC, PSU and the case came in at around 70EUR.

Software

My 'client OS' is Mac OSX 10.9.5 using Java 8.

Building Spigot

Why are there no Spigot builds on their website you ask? Read this post for details.

Building Spigot is easy. Any mildly current hardware will build Spigot around half an hour. **It will not work on the RASPBERRY PI (not enough memory)**. I followed these instructions. In the end, you'll have a spigot-1.8.jar which you'll want to copy over to the RASPBERRY PI via scp or so.

Preparing the RASPBERRY PI

OS Setup

I assume you have a fresh Raspbian install (details here) which was stripped off unnecessary packages by using these instructions: http://sirlagz.net/tag/raspbian-server-edition/.

Don't even think about running an Apache web server or similar alongside Spigot. We're about to push our little RASPBERRY PI to his limits without additional services already by running Spigot and a couple of plugins (the average system load is close to 1.0 with 2 connected users).

Tuning Raspbian for performance

To tune Raspbian to a minimum memory footprint i've closely followed this guide: https://extremeshok.com/1081/raspberry-pi-raspbian-tuning-optimising-optimizing-for-reduced-memory-usage/. I left out these steps: ZRAM, 512MB swapfile and enabling preload. Be careful with the overclocking. Better use the raspi-config commandline tool that comes with Raspbian and set the overclock settings to 'High' or 'Turbo' (also see http://www.raspberrypi.org/introducing-turbo-mode-up-to-50-more-performance-for-free/, your warranty is safe), that does the trick too.

Java 8/armel

Download and install this Oracle Java 8/arm distribution to /opt/jdk or so. Sadly, this Java build is not capable of running a -server JVM, but it is hardfloat ABI which provides for a lot of speed.

The reason i don't use the provided oracle-java8-jdk Raspbian package simply is that i prefer one big distribution sitting under /opt over having to install 54 (yes.) packages, see below for details:

```
# apt-get install oracle-java8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  fontconfig fontconfig-config hicolor-icon-theme libasound2 libatk1.0-0
libavahi-client3 libavahi-common-data libavahi-common3 libcairo2 libcups2
libdatrie1 libfontconfig1 libfreetype6 libgdk-pixbuf2.0-0
  libgdk-pixbuf2.0-common libgraphite2-2.0.0 libgtk2.0-0 libgtk2.0-bin
libgtk2.0-common libharfbuzz0a libjasper1 libjbig0 libpango-1.0-0
libpango1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpangox-1.0-0
  libpangoxft-1.0-0 libpixman-1-0 libpng12-0 libthai-data libthai0 libtiff4
libx11-6 libx11-data libxau6 libxcb-render0 libxcb-shm0 libxcb1
libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 libxext6
  libxfixes3 libxft2 libxi6 libxinerama1 libxrandr2 libxrender1 libxtst6
ttf-dejavu-core x11-common
Suggested packages:
  libasound2-plugins cups-common librsvg2-common gvfs libjasper-runtime ttf-
baekmuk ttf-arphic-gbsn00lp ttf-arphic-bsmi00lp ttf-arphic-gkai00mp ttf-
arphic-bkai00mp
The following NEW packages will be installed:
  fontconfig fontconfig-config hicolor-icon-theme libasound2 libatk1.0-0
libavahi-client3 libavahi-common-data libavahi-common3 libcairo2 libcups2
libdatrie1 libfontconfig1 libfreetype6 libgdk-pixbuf2.0-0
  libgdk-pixbuf2.0-common libgraphite2-2.0.0 libgtk2.0-0 libgtk2.0-bin
libgtk2.0-common libharfbuzz0a libjasper1 libjbig0 libpango-1.0-0
libpango1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpangox-1.0-0
  libpangoxft-1.0-0 libpixman-1-0 libpng12-0 libthai-data libthai0 libtiff4
libx11-6 libx11-data libxau6 libxcb-render0 libxcb-shm0 libxcb1
libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 libxext6
  libxfixes3 libxft2 libxi6 libxinerama1 libxrandr2 libxrender1 libxtst6
oracle-java8-jdk ttf-dejavu-core x11-common
0 upgraded, 54 newly installed, 0 to remove and 0 not upgraded.
Need to get 87.4 MB/102 MB of archives.
After this operation, 237 MB of additional disk space will be used.
Do you want to continue [Y/n]? n
Abort.
```

Of course you can use OpenJDK or Oracle Java 7 or whatnot, but after some comparison (sorry for not publishing any results, you'll have to believe me) the Java 8 turned out to be somewhat 2-4x faster than the others. For more information about why it is so much faster read about hard- vs. soft-float ABIs.

Setting up Spigot and Plugins

Spigot and a bit of glue

I'm using 'screen' to be able to run the Spigot server in the background. You might want to apt-get install it if you haven't done so by yourself. My .screenrc looks like this:

```
startup_message off
defscrollback 3000
hardstatus alwayslastline " [ %H ] %{wb} %c:%s | %d.%m.%Y %{wr} Load: %l
%{wb} %w "
```

Create a user called 'minecraft', make sure to specify the dash shell when creating (otherwise the optimization done before above is rendered useless). Login as minecraft, setup ssh keys, the environment and such.

Create a 'spigot' directory, place the spigot-1.8.jar here. To start spigot, i use this shell script called 'start.sh' (the default value of the restart-script option in spigot.yml) (chmod +x) in the spigot directory:

```
#!/bin/dash

JAVA_HOME="/opt/jdk"
JAVA="/opt/jdk/bin/java"
JAVA_OPTS="-Xms256M -Xmx496M"
PERF_OPTS="-XX:ReservedCodeCacheSize=8M -Djava.awt.headless=true -
XX:+UseG1GC"
SPIGOT_OPTS="-Dcom.mojang.eula.agree=true"

${JAVA} ${JAVA_OPTS} ${PERF_OPTS} ${SPIGOT_OPTS} -jar ./spigot-1.8.jar nogui
```

If you want to try this on a Rev. 1/2 Model B or A RASPBERRY PI, you'll have to adapt the -Xms and -Xmx options for the probably lower memory these models provide (i.e. -Xms128M -Xmx242M).

To glue all this together, i've put a cronjob in place that looks like this:

```
@reboot /usr/bin/screen -s /bin/bash -dmUS spigot
/home/minecraft/spigot/start.sh
```

This starts Spigot on boot. Should spigot fail, it should take care of restarting by itself (see the restart-on-crash and restart-script options in spigot.yml). If this is not enough for you, put a while loop around the start.sh script from above.

Make a directory called 'plugins' and put the NoSpawnChunks plugin in there already. It will dramatically decrease Spigots startup duration.

Run Spigot with the start.sh now, then just enter 'stop' at the '>' prompt to stop the server again. We just wanted it to generate all the config files.

Config changes

In the server.properties, reduce the view-distance to something between 3 and 5, maybe 6. This has exponential consequences on Ticks and TPS, go easy on that. 5 seems to be the maximum that does not suck too much. Do the same in spigot.yml. While you're already there, disable anti-xray by setting its 'enabled' property to false.

If you like, play around with the other options. Besides the above, i've only set a custom MOTD and a custom whitelist message. I've seen a couple of howtos doing wild changes in all sorts of config files. I found this to be unnecessary (even making stuff worse in some cases), handle these with care!

You might want to think about enabling whitelisting and setting fine-granular permissions if you're planning on letting other users into your server.

Plugins

PluginMetrics

In ~/spigot/plugins/PluginMetrics/config.yml, set opt-out to true.

ClearLag

Download and install ClearLag to ~/spigot/plugins. Reload Spigot to acitivate ClearLag. See ~/spigot/plugins/ClearLag/config.yml for configuration.

DynMap

Download and install DynMap to ~/spigot/plugins. Reload Spigot. When the reload is completed, direct your browser to http://raspi-hostname:8123. At first, the map will be empty. But as you discover more and more of your world, the map will fill more and more.

MultiVerse

Download and install MultiVerse-core to ~/spigot/plugins. See the /mv command for help. While connected to the Spigot server with Minecraft, teleport (see /mv tp) to each world at least once and execute /mv setspawn.

WorldBorder

Download and install WorldBorder to ~/spigot/plugins. See the /wb command for help. For each world you want to pregenerate, execute the following steps (on the spigot console):

- wb <world> set 4096 spawn
- wb shape round
- wb dynmap on
- wb fill

The 4096 as argument to wb <world> set defines the radius in blocks which is pregenerated upon executing wb fill. This is quite time-consuming, a radius of 4096 blocks takes more than 24 hours to pregenerate.

Additional resources not linked above

- http://www.dachboden-server.de/tuts/151-05-1-info-tipps-zur-server-performance general info about minecraft server performance in german.
- http://www.spigotmc.org/wiki/lag-types-and-how-to-fix-them/ about different lag types.
- http://www.minecraftforum.net/forums/archive/alpha/alpha-survival-multiplayer/823328-making-your-server-lag-less-by-tuning-java about java garbage collection.
- http://www.sk89q.com/2013/03/improving-your-minecraft-servers-performance/ about minecraft server tuning.
- https://gaming.stackexchange.com/questions/20692/is-it-possible-to-turn-a-minecraft-single-player-map-into-multi-player-server and https://gaming.stackexchange.com/questions/27728/moving-from-vanilla-server-to-bukkit about migrating a map from the minecraft client into spigot or vanilla server.
- http://timings.aikar.co/ to analyze Spigot timings deeply.

Additional readings about Java GC:

- https://stackoverflow.com/questions/2101518/difference-between-xxuseparallelgc-and-xxusepar newgc
- http://www.umbrant.com/blog/2012/twitter_jvm_tuning.html
- http://www.oracle.com/technetwork/java/javase/tech/g1-intro-jsp-135488.html
- http://www.oracle.com/technetwork/tutorials/tutorials-1876574.html

I'm not entirely sure if (and if, how) http://bukkit.org/threads/bukkit-sends-chunks-slow.15798/page-2 relates to Spigot. Anyways, removing the FPS limit (or setting it as high as possible) on the client side seems to have a positive effect on Spigots chunk send performance.

raspberrypi, minecraft