Teaching: GCU RDP Storage Design

Pi Storage (32Gb SD Card)

Partition	**Mount**	**Size**	**Used**	**Notes**
/dev/mmcblk0p6	/boot	75Mb	23Bb used	The boot partition
/dev/mmcblk0p5	/media/pi/SETTINGS	30Mb	0Mb used	Pi Settings
/dev/root	/	29Gb	5Gb used	User and App partition
Others	/dev, /run, /sys	20Mb	50Mb used	Tmp, Lock, etc

- All Pi's for this project have a 32Gb SD Card.
- We will assume 90% of SD Card will be available for use or about 29Gb (this was determined from an actual Pi with Raspbian and Docker installed).
- We will assume 4Gb will be taken by Docker Images (see below).
- We will then assume that 25Gb will be available for Application and Database Storage.

Docker Image Sizes

Runtime	**Size**
JRE 1.8 (OpenJDK +Alpine)	75Mb
Tomcat	100Mb
TomEE	505Mb
JBoss Wildfly	600Mb
Apache PHP	150Mb ??
Nginx	90Mb
.NET Core	150Mb ??
Python	350Mb ??
Python Tensorflow	350Mb ??
MySQL	125Mb ??
PostgreSQL	83Mb ??

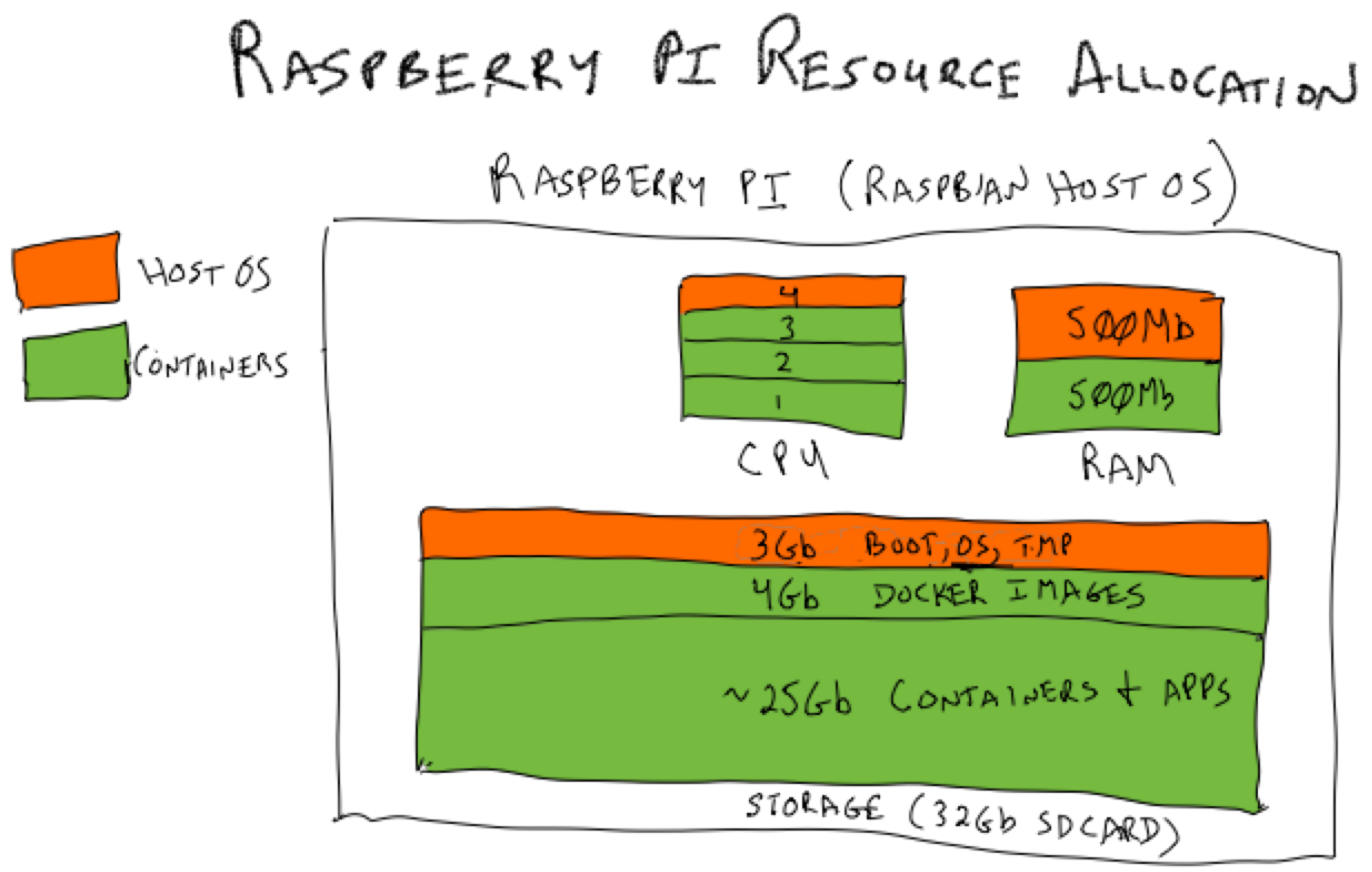
- Some of above image sizes were calculated from GitHub (some sizes are shown compressed in GitHub so really 12x)
- If we preload all the above Images this will take up about 3-4Gb.
- If we do not preload all the Images then running Docker Containers will take a large first time start up penalty and in the long run when Apps are created and deleted then many of the Images will have been downloaded anyway.
- We will create a utility script that will install/configure Docker as well as to build and pre-load all the Images on the SD Card.

Pi RAM

- The Model B+ has 1Gb of memory.
- With the Host OS and Docker running the available free memory would be about 500Mb, which will then be available for Docker Containers (this was determined from an actual Pi with Docker installed).

Pi CPU

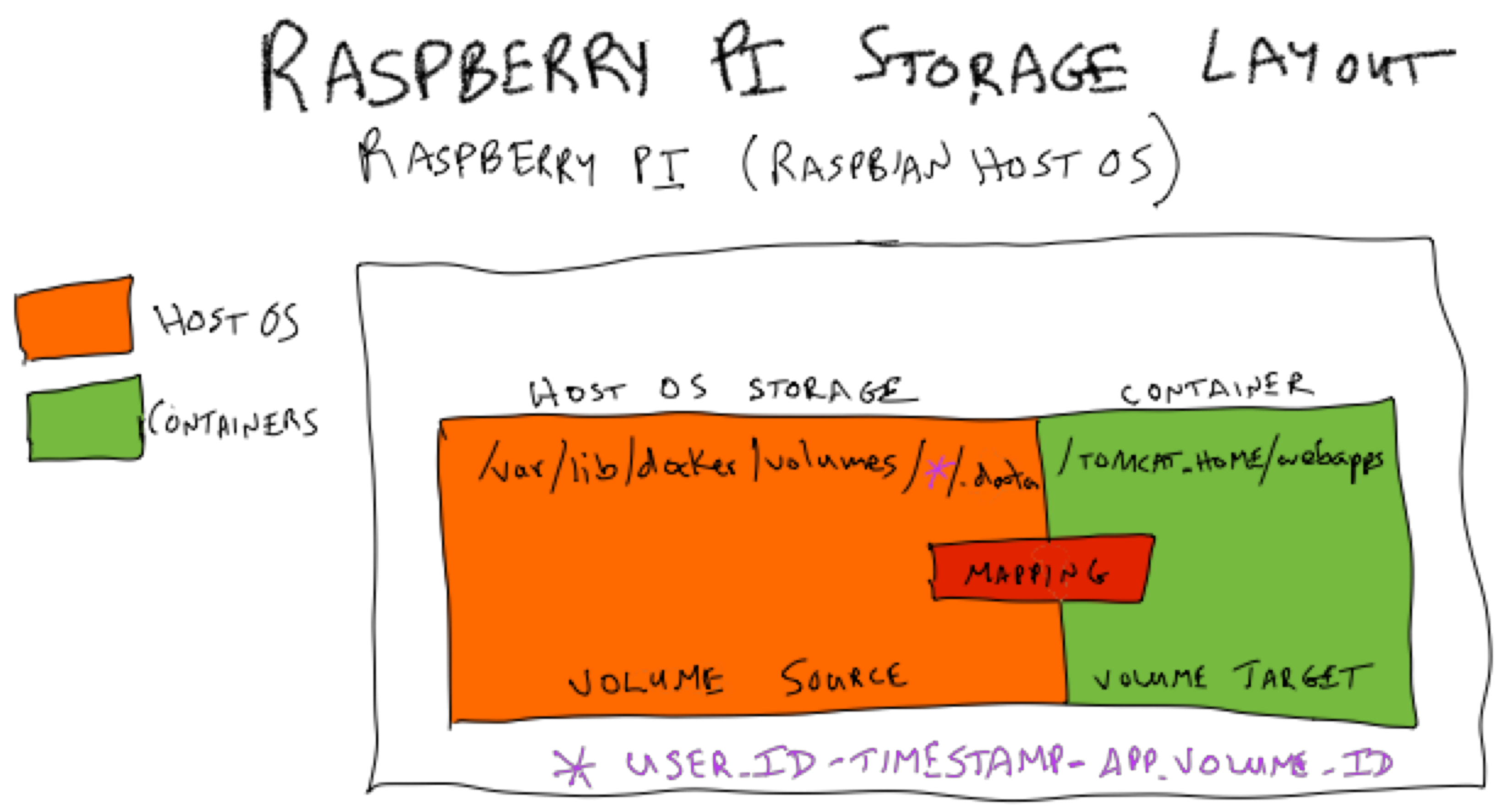
- The Model B+ has 1.4GHz 64 bit 4 Core ARM CPU (but Raspbian is a 32 bit OS).
- Assume we use 3 of the available cores for Docker Containers and control limits with Docker Service/Container control arguments (see Application Sizes below).



Host Volume Strategy

- You can specify a Docker Volume when you run a Container. This will create the Volume if it does not exist.
  - The Docker run command with a mount specification is used:
    - --mount source=[VOLUME_NAME],target=[PATH_IN_APP]
  - The Volume is then create on the Host as follows: /var/lib/docker/volumes/[VOLUME_NAME]/[_data].
- Since we could have more than one user leveraging the Pi Host OS we will need to isolate users data and each of the users app data. We will use a combination of a User ID, Timestamp, and App Volume Name as the VOLUME_NAME that will be specified when a Container is run.
- This will allow us to create the Docker Volume at run time and not require a Docker Volume to be created before the Container is run. The GCU Docker Client API will support the VOLUME_NAME to be passed to the Container and Service create API's.
- A set of predefined App Volume Names will be coded into the Portal Application based on the Application Stack needs.
- To support application deployments App Volume Name will be set to 'deployments' and used for Host based Deployment of Application Deployment Artifacts.
  - /var/lib/docker/volumes/[USER_ID]-[TIMESTAMP]-deployments]/_data
- To support database storage App Volume Name will be set to 'database'

- /var/lib/docker/volumes/[USER_ID]-[TIMESTAMP]-database]/_data


RASPBERRY PI STORAGE LAYOUT
RASPBERRY PI (RASPBIAN HOST OS)
HOST OS
CONTAINERS
HOST OS STORAGE
CONTAINER
/var/lib/docker/volumes/*/.data
/TOMCAT_HOME/webapps
MAPPING
VOLUME SOURCE
VOLUME TARGET
* USER_ID-TIMESTAMP-APP_VOLUME_ID

Application Sizes

Size	vCPU	RAM	Storage
Tiny	0.50	128Mb	2Gb
Medium	0.75	256Mb	4Gb
Large	1.00	512Mb	10Gb

- We are memory constrained by the limited 1Gb of memory in the Pi:
  - We could run 3 Tiny Applications per Pi.
  - We could run 2 Medium Applications per Pi.
  - We could run 1 Large Applications per Pi.

Application Stacks

Stack Name	Runtime	Available Sizes	Container Storage Mounts	Host Volume
Web App	Nginx	S	TBD (see default nginx.conf)	See above strategy
Java App (Spring Boot)	JRE 1.8 + MySQL	S/M	TBD /var/lib/mysql	See above strategy
Java Web App 1	Tomcat 8.5	S/M	/TOMCAT_HOME/webapps	See above strategy
Java Web APP 2	Tomcat 8.5 + MySQL	S/M	/TOMCAT_HOME/webapps	See above strategy

			/var/lib/mysql	
Java Web App 3	Tomcat 8.5 + Postgres	S/M	/TOMCAT_HOME/webapps /var/lib/postgresql/data	See above strategy
Java Web App 4	TomEE + MySQL	L	/TOMCAT_HOME/webapps /var/lib/mysql	See above strategy
Java Web App 5	TomEE + Postgres	L	/TOMCAT_HOME/webapps /var/lib/postgresql/data	See above strategy
Java Web App 6	JBoss Wildfly + MySQL	L	/JBOSS_HOME/standalone/deployments /var/lib/mysql	See above strategy
Java Web App 7	JBoss Wildfly+ Postgres	L	/JBOSS_HOME/standalone/deployments /var/lib/postgresql/data	See above strategy
.NET Web App	.NET Core + MySQL	L	/app/out /var/lib/mysql	See above strategy
PHP Web App	PHP 7.2 + MySQL	S/M	/PHP_HOME/htdocs /var/lib/mysql	See above strategy
Python App	Python	S/M	TBD	See above strategy
Python AI App	Python + Tensorflw	S/M	TBD	See above strategy

TODO:

- [ ] How will we “deploy a Spring Boot application since it run as a Java application?
- [ ] Update GCU Docker Client API so that a Volume Name can be specified when creating a Container or Service when creating a Service
- [ ] Update GCU Docker Client API so that a Volume is removed when removing a Container or Service (also check out the run —rm flag as this is supposed remove the Volume on remove)