Raspberry Pi Project

Keith E.S. Allatt

Febuary 2019

1 Project Description

1.1 About

Much like some companies purchase 'processing power' from other computers around the world to 'crunch the numbers' when it comes to accomplishing computationally heavy work, this Raspberry Pi Project (R.P.P.) is aimed at off-loading computational work to a network of Raspberry Pis. Using a network of model A+s, we plan on making a small scale, proof of concept cluster, using one master RasPi and many slave RasPis. Much of the software required is implemented on a single RasPi and a configuration file will coordinate the rest.

1.2 Motivation

The motivation behind this project is not one of necessity, but one of learning. We're attempting this project to learn more about the Linux operating system, networking, single board computers and their uses, and automation.

This project can also be used as an addition to my portfolio, and all code will be open sourced on GitHub. Full tutorials and documentation will be provided on GitHub.

1.3 Hardware

1.3.1 Raspberry Pis

For this project, we're going to use a collection of Raspberry Pi Model 3 A+ single board computers. Since these don't have an Ethernet connection, all work will be done over WiFi.

Raspberry Pi Model 3 A+ specifications:

- **System on Chip**: Broadcom BCM2837B0 quad-core A54 (ARMv8) 64-bit @ 1.4GHz
- **GPU**: Broadcom VideoCore IV
- **Networking**: 2.4GHz and 5GHz 802.11b/g/n/ac wireless LAN
- **RAM**: 512MB LPDDR2 SDRAM
- **Bluetooth**: Bluetooth 4.2, Bluetooth Low Energy
- **GPIO**: 40-pin GPIO header, populated
- **Storage**: micro SD
- **Ports**: HDMI, 3.5 mm analogue audio-video jack, 1× USB 2.0, Camera Serial Interface, Display Serial Interface
- **Dimensions**: 67×56×11.5 mm

Many Raspberry Pi kits come with a dedicated power supply. To cut down on size, we may change from the default RasPi power supply to using USB wall outlet adapters and USB to Micro USB cables. For proof of concept we'll most likely preserve the original power supply.

Cost per unit:

From `https://www.canakit.com/raspberry-pi-3-model-a-plus.html?cid=cad&src=raspberrypi`

Board Only 32.95 CAD

– Raspberry Pi 3 Model A+ Board

Basic Kit (16GB) 49.95 CAD

– Raspberry Pi 3 Model A+ Board
– 16 GB SanDisk Class 10 MicroSD (with NOOBS)
– CanaKit 2.5A Micro USB Power Supply
– Aluminum Heat Sink

Starter Kit (32GB) 64.95 CAD

– Raspberry Pi 3 Model A+ Board
– 32 GB Samsung EVO+ Class 10 MicroSD (with NOOBS)
– CanaKit 2.5A Micro USB Power Supply
– Aluminum Heat Sink
– USB Card Reader
– 6-Foot HDMI Cable (with CEC support)
– GPIO Reference Card

RasPi Zero W

With on board WiFi and Bluetooth, a 1GHz processor with 512MB RAM, Mini HDMI and USB On-The-Go ports, Micro USB power, and dimensions of only 65×30×5mm, the RasPi Zero W is a solid alternative to cut down on costs.

For a kit that comes with an 8GB SD card and a power supply comes to about the same cost of the Model A+ board alone, using these to create this computational farm would be more economic, but unfortunately for small numbers of boards, this is not possible. Most retailers only offer one board per customer, unless you are buying hundreds, in which case the cost per unit increases.

1.3.2 Memory / Storage

For storage of either past tasks, scripts, and outputs, potentially backups of other information on the master node, a solid state drive (to remove bottlenecks and increase reliability) will

```
media
└─ SSD
   ├─ scripts
   ├─ tasks
   └─ outputs
```

1.4 Software

This is a project designed to take control of a set of RasPis to schedule, and complete tasks. The way this will be achieved through a scheduling task that will run every specified time interval. When this happens, a script will check for a new task to be run. If there is a new task, it will execute said task. When the script is executing a task, no other tasks are allowed to run. With enough RasPis, using SSH, we should be able to auto distribute tasks (Explained Later)

Each task will consist of some input data, either a single instance or a list, and a given script. In a bin folder (TBA), a given task will be formatted as an XML or other markup language, where parsing is easy. A field labeled 'script' will contain a path to the script to be executed. Each 'input' tagged field will contain a set of data that will be parsed and passed to the script, and the script, input and output will be stored in a row of output, to be checked later.

For low volume tasks, such as home use, 1 to 2 tasks every hour should be enough, unless the input sizes are very large. To accommodate this, the scheduler will have to gauge data input size and make appropriate choices as to how to change timing to better accommodate the number of tasks (Idle time should be minimized).

If one given RasPi in the pseudo-cluster is loaded with a lot of tasks, that RasPi should be able to SSH into another RasPi, send the file, verify it was sent correctly (send the file with a hash / checksum), and have the other RasPi handle the task.

1.4.1 File System

The file system, as seen to the right, contains all the necessary files for this application to work without a web server. With a web server, some directories are moved to `/var/www/html` to give access to certain files to the server while maintaining access to the rest of the software.

```
root
├── scripts
│   ├── dependencies
│   │   ├── run_task.py
│   │   ├── scheduler.py
│   │   └── task_dependencies
│   ├── SAMPLE.py
│   └── ...
├── funnel
│   └── ...
├── output
│   ├── SAMPLE
│   │   ├── SAMPLE.out
│   │   └── task.xml
│   └── ...
├── task_bin
│   └── ...
└── left_overs
    └── ...
```

- `scripts`
- `scripts/dependencies`
- `task_bin`
- `output`
- `funnel`
- `left_overs`

1.4.2 Loaded Modules

Each Raspberry Pi will be initialized with the newest form of Raspbian, the Debian flavor of Linux optimized for the Raspberry Pi. The Lite edition is a minimal image based specifically on Debian Stretch.

- Python 3.* (Preferably 3.7.2 / latest).
- `crontab -e` (For scheduling).
- Apache Web Server (For hosting results on the local network).
- PHP for file submissions.
- SSH / SCP (For networking and file transfer).

- The Python 3.* code written specifically for this project:
  - For web server updates.
  - For task completion.
  - For general tasks manipulating files.

1.5 Networking

1.5.1 Web Server

The point of the web server is to display the results locally. At no point will this be linked to a domain name or hosted on the internet, but potentially hosted locally with minimal security.

1.5.2 SCP

SCP is a file transfer protocol tool to transfer files between Unix based systems, perfect for this application.

I plan on using one 'master' RasPi to act as the main connection between all the slave RasPis. This way there are no unnecessary files on the other RasPis. When a task is offloaded, the XML file and the script it requires will be transferred to the new node, so each node only keeps track of the scripts it has needed. A data dump can be achieved from the master node to keep each node.

1.6 Next Steps

1.6.1 Adding an LCD Screen

LCD Screen **Cost:** 34.95 CAD

By adding an LCD screen (not necessarily high resolution, Adafruit makes one that is plug and play), It becomes easier to observe changes in the state of each RasPi. A single shield display for the master node should be enough to display all necessary information, especially if a shell script or a python script gathered information about each RasPi and displayed it every given time interval.

1.6.2 Adding a power button

Jumper Cables: **Cost:** 5.59 CAD
Solder-less Breadboard: **Cost:** 0.79 CAD
Smart Electronics Starter Kit: **Cost:** 11.76

By adding a power button to certain pins on each raspberry pi, we can temporarily short out two pins on the GPIO input on every RasPi. If a RasPi has been halted but still has power, shorting these pins starts each RasPi up. By writing a script to start on boot and continuously check for that connection, we can automate shutting down that way too. By connecting each RasPi to the same button, we can start and stop the entire collection simultaneously.

1.6.3 Adding external storage

2 Set Up

install packages

install custom code

add flag for maintenance in folder

set up ssh and keys

set up config

restart web server

set up crontab

remove maintenance flag

3 General Usage