A Punnet of Berries Documentation

1

TeamPi

August 26, 2013

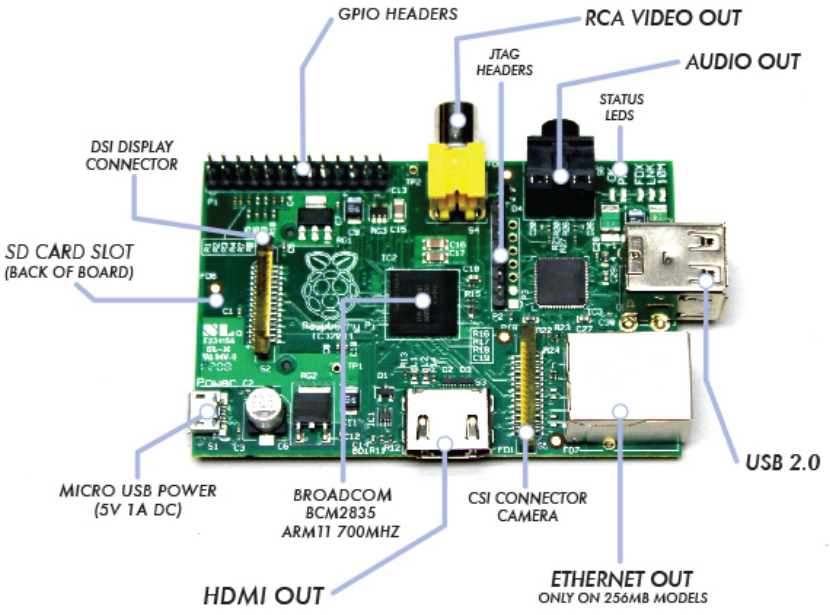
CONTENTS

CHAPTER

ONE

INTRODUCTION

The Raspberry Pi is a single-board computer. This credit card sized device is capable of running Linux, and other light-weight operating systems, with its ARM processors.



A Beowulf cluster is a collection of (typically) inexpensive computers, networked together by a local area network, usually an Ethernet, and running some parallel processing software. This concept puts the power of supercomputing into the hands of schools and small research groups.

Developed to educate and inspire the next generation of programmers, the Raspberry Pi is a powerful, yet, low-cost "mini-computer".

The **Punnet of Berries** project aims to create a *Beowulf cluster* out of several Raspberry Pis.

CHAPTER

TWO

SYSTEM OVERVIEW

In order for a compute cluster to function properly and fairly, its resources must be managed and monitored by a batch system scheduler. Batch systems enforce limits on the number of jobs running at one time and the resources available to them. Compute cluster users request resources by submitting jobs to the batch system, which then determines the scheduling that best utilises the resources available.

The Punnet of Berries compute cluster will be managed by a custom batch system, the *Berry Batch*.

2.1 Goals and Objectives

2.1.1 The Punnet

CHAPTER

THREE

DESIGN CONSIDERATIONS

CHAPTER

FOUR

ARCHITECTURE

4.1 System

Each Raspberry Pi will form a compute node in the Punnet of Berries cluster.

4.2 Program Design

The Berry Batch will consist of a centralised manager daemon and worker daemons.

Each node in the Punnet of Berries cluster will run a Berry Batch worker daemon. These workers exist to execute the jobs submitted to the system. The workers will:

1. Wait to be assigned a job.
2. Execute the job.
3. Returns the job's exit status.
4. Back to 1.

A job finished execution, with one of the following status:

1. **Successful** The job finished execution within the requested time frame and resources and without encountering any errors.
2. **Error** The job encountered an error during execution.
3. **Walltime Exceeded** The job exceeded the requested walltime and was terminated by Berry Batch.
4. **Resources Exceeded** The job required more system resources than requested and was terminated by Berry Batch.

The Berry Batch manager will monitor the system's resources and jobs. Users will interact directly with the Berry Batch manager to manage their job requests. A user is able to:

- Submit jobs.
- View all queued jobs.
- View the status of all of a particular user's jobs.
- View the status of a particular job.
- Cancel their own jobs.

4.2.1 Submitting Jobs

Users will write job scripts, which will be passed to the Berry Batch manager. A job script will consist of header and execution sections. The header section will specify the job details for queueing by Berry Batch. This could include a name for the job, the memory required, an estimated walltime and the number of nodes to run on. The execution section will contain the actual commands to be executed as a part of the job. While the job script's specific format is yet to be decided, an example is:

```
## Header Start ##
Name="JobHello"
Walltime=00:50:00
Nodes=2
Memory=256mb
## Header End ##

mpirun -np 2 helloWorld
```

Once a job has been submitted, the Berry Batch manager will determine which priority queue the job should wait in based on the resources and walltime requested in the header section. The queues could be:

- Short
- Medium
- Long
- Special (requires permission from the Punnet of Berries administration to run)

The *special* queue has first priority, followed by the *long* queue, and so on. If the resources are not available for any job in the *special* queue, the manager will look in the *long* queue for a suitable job, and so on. Each queue will operate in a *First in First Out* fashion.

4.2.2 Job Execution

Once a job has been submitted, it will move through the following states:

1. Queued
2. Running
3. Complete
4. Cancelled

When a job has completed, a Berry Batch job summary file will be written. This summary will contain details of the job's execution, such as the resources and walltime that were requested as well as what was actually used. Any standard output generated during execution will also be included in the summary file.

CHAPTER

FIVE

TESTING

5.1 Compute Cluster

- OpenMPI benchmarking

5.2 Batch System Tests

To prove that the Berry Batch is functioning correctly, it will undergo a range of tests. These include, but are not limited to, the following:

Test Cases	Expected Result
Submit 1 short job.	Confirmation. Job put in short queue.
Submit 1 medium job.	Confirmation. Job put in medium queue.
Submit 1 long job.	Confirmation. Job put in long queue.
Submit 1 special job.	Job set to wait for permission.
One user submits a short job followed by one medium job.	First job put in short queue. Second job put in medium queue.
One user submit a short job. Second user submits a short job.	First user's job put in short queue. Second user's job put in short queue after the first.
One user submit a short job. Second user submits a medium job.	First user's job put in short queue. Second user's job put in medium queue.
One user submit a short job. Second user submits a medium job. Third user submits a medium job.	First user's job put in short queue. Second user's job put in medium queue. Third user's job put in long queue.
User views all current jobs.	List of jobs currently in the system.
User views a particular user's jobs.	List of user's jobs currently in the system.
User views a specific job.	Details of the given job.
First user views all current jobs. Second user views a specific jobs.	First user sees list of jobs currently in the system. Second user sees the details of the given job.
User cancels a queued job.	Confirmation of job cancellation.
User cancels a running job.	Confirmation of job cancellation.
User cancels a completed job.	Warning of invalid state.
User cancels an already cancelled job.	Warning of invalid state.
One user cancels another user's job.	Permission denied.

kjad;lkfja;dsf sdf;lajdf;kadfaf;lkja kajd;lfa

CHAPTER

SIX

ROLES AND RESPONSIBILITIES

The members of TeamPi are:

- **Alyssa Biasi:**

  Primary Role: Documentation, Testing and Management

  Secondary Role(s): Application Development, Operating System Configuration

- **Adrian Zielonka**

  Primary Role: Application Development

  Secondary Role(s):

- **Zach Ryan**

  Primary Role: Hardware and Operating System Configuration

  Secondary Role(s):