A Cluster of PI’s

For RSSC
By HLHowell

A Cluster of PI’s

Why would one want a cluster of PI’s?

Well, for robotics, as we have advanced beyond the BasicStamp, Assembly language on the Microchip controllers, and Arduino, we come upon a quandry. A laptop consumes a lot of power, which could run our robot. It lacks IO pins for robot stuff, and is physically large.

- We need a new solution. One pi is not enough, so…

A Cluster of PI’s.

A Cluster of PI’s

- A hopeful agenda for this session
  - Hardware needed
  - Software needed
  - A process to follow
  - Lots of patience
  - Examples
  - The goal
  - How does MPICH work?
  - How do I intend to use it?
  - What do I have so far?
  - What needs improvement?

Hardware needed

- Raspberry PI’s. I decided upon 5, but you could use more or less.
- USB power hub
- Network switch
- USB power cables
- Network cables
- microsdcards
- Fans
- standoff’s.
- **Total cost for my 5 PI example is $254.**
- You should also have a monitor, keyboard and mouse to aid in setup. Or you can run headless and log in remotely. The instructions I posted require the terminal and keyboard.
- Your display requirement for the robot would depend on your desires and the robot
- A desktop to load the images onto the sdcards.

Software needed

- From scratch
  - Raspian desktop version: free from raspberrypi.org
  - Raspian lite version: free from raspberrypi.org
  - MPICH using wget and build it locally
  - MPI4PY and build it locally
  - Add the following to your pi’s:
    - Development software and libraries of gfortran, gcc, python at a minimum. Other languages may be used if you desire, but I don’t document them nor do I know them.
  - Install clamav. The av software will protect your stack. Put it on all pi’s.
- Using my software:
  - Download the images and verify their setup.
  - Follow the instructions for the slaves and master as required. Most of the master stuff is already done, but some may need to be updated for your installation.
  - Install clamav the AV software will protect your stack, put it on all pi’s.

A process to follow

- On the website you will find a directory called RobotCluster. It will contain the step by step instructions called Building Beowulf, an image for the master, and an image for the slaves, and this presentation. These are for PI 3B+, but the PI 3B should also work but they are I think too slow due to the ethernet of 100Mhz and only getting about 30.

Lots of patience

- MPICH stands for Message Passing Interface Chameleon.
  - It can run on a cluster made up of different systems and do multiple tasks on each process on each system. On Raspberry PI, the processors are single threaded, using the 3B+ if you setup four or fewer processes on each they will be divided between the processors.
  - Memory is distributed among them, but not common, so one PI does NOT affect the memory on another.
  - The code runs simultaneously, but not controlled on all processes.
  - Each PI gets its own copy of the code, operates in its own memory, and the only shared resource is the Network File System shared directory. And using it consumes bandwidth of the network.
  - Communications consist of sending and receiving ethernet packets through the mpich buffers.
- This means learning the MPICH communications protocol and limitations. The full documentation and lots of other pieces are available at www.mpich.org The full manual is called mpi31-report.pdf. It is not an easy read, but every command is documented there.
- Many of the web examples are on different versions of MPICH and some of them are mistakenly stated to be MPICH when they are openMPI, which is slightly different, and will require some tuning.
- In addition, MPICH and MPI4PY must be installed separately and require some setup time, and in the case of MPICH, a few hours of build and install time. Once you understand it and get it working, though the system is in essence fully expandable. That means you could conceivably expand it to the size of the 750 pi ANL system. It’s only money, right??

Examples

- Most useful is a python program called mypi.py.
- It will show you some statistics about your Pis including the temp, which we like to know. Here is a sample output:

```
• $ python mypi.py
• ----------------------------------------
• Pi Model              : Raspberry Pi 3 Model
  B Plus Rev 1.3
• ----------------------------------------
• System                : Linux-4.14.62-v7+-
  armv7l-with-debian-9.4
• Revision Number       : a020d3
• Serial Number         : 00000000eb0ee1d1
• Python version        : 2.7.13
• ----------------------------------------
• I2C enabled           : False
• SPI enabled           : False
• Bluetooth enabled     : True
•
```

```
• ----------------------------------------
• Ethernet Name        : eth0
• Ethernet MAC Address : b8:27:eb:0e:e1:d1
• Ethernet IP Address  : 192.168.1.200
• Wireless MAC Address : b8:27:eb:5b:b4:84
• Wireless IP Address  : 192.168.0.9
• ----------------------------------------
• CPU Clock            : 1400MHz
• CPU Temperature      : 32.70%C
• GPU Temperature      : 32.17%C
• RAM (Available)      : 927MB (82MB)
• Disk (Available)     : 57G (50G)
• ----------------------------------------
• pi@ppim:~/cloud $
•
```

More Examples

- The website will host this slide show, the build instructions, the disk images for the slaves and the master, and a directory called Cloud which contains my hosts files, a README detailing the programs and the requirements for running each of the debugged examples. I have tried to capture most of the things you will need to get up and running.

The GOAL

- Robot navigation by vision
- Lots of robot control and sensor pins
- Separation of requirements
- Central control
- Remote control with absolute shutdown
- Low power consumption for the processing power used
- Expandable system with common architecture

How does MPICH work?

MPI stands for Message passing interface

CH stands for chameleon, for running on many OS’s

- A shared file structure for control and software sharing across systems, a message control structure with buffering to simplify messaging that is independent of the code with a common interface supporting several languages and appropriate compiler libraries.
- The whole program is made available to all processors via the shared file system, but the knowledge by the MPI interface of the number of processes, the id of a specific process, and the ability to share the computational load or run different process concurrently with common control makes it adaptable to very large or computationally difficult problems.
- It is not necessarily limited to common parallel processing limitations, although many authors seem to think so. It is this flexibility that makes it a good platform for robotics.
- Many algorithms are not separable to parallel processing, but with a cluster, those algorithms can be handed off to a separate process, and allowed to complete, furnishing their data when done. Meanwhile other routines can be passed off to either subclusters or other processors.

Not a state machine

- Most simple robotics yeild to a state machine analysis. One example is IFTTT, where one state leads to another. A line following robot likely uses a state machine that says go left or go right then check if centered to go straight.
- As robots get more complex, similar to humans, multiple things happen all the time, we are NOT state machines and our advanced robotics will not be that either. Humans process information in many ways simultaneously, for example balance, vision, hearing, touch, smell, speaking and walking all at the same time. For a robot to be successful at humanoid tasks, it has to be able to chew gum and walk.
- Cluster computing is one path to that capability and MPICH is one branch of that path.

How do I intend to use it?

- Visual reduction for location maybe using a form of SLAM.
- Control of simulated emotional response depending on the “face” and “arms” of my eventual structure.
- Walking bipedal I hope??!!
- Some degree of a sense of touch
- Voice I/O at least some useable vocabulary.
- Self locating for recharge
- LOTS of sensors and drives for the requisite hardware to make it useful.
- Runtime of at least 4 hours with 12 being the eventual target for the prototype.
- And whatever I can figure out before I quit or give up in frustration.

What do I have so far

- The cluster on AC
- The ability to run it on 12V DC
- Examples of code and communications that work.
- A nebulous concept in my head that is evolving.
- A setup that can be replicated somewhat easily for any one that is interested, with a couple of hundred bucks for a dream.

What needs improvement?

- A container!
- LED’s to make it more interesting.
- Better fan placement, they block gpio access.
- Routing for the gpio pins to some sort of standard set of wiring interfaces to make bot connections easier
- More time!