Native Linux on a Mac Mini 2011

Ross Maloney

February 26, 2012

Abstract

A method of creating a dual boot of OS X Lion and Linux on a 2011 released Apple Mac Mini 5.3 server is described. The Linux distribution installed was 64 bit Ubuntu Precise alpha 1 server, selected as a lean environment which could be subsequently built upon. A different installation approach to that used with earlier Mac Minis and Mac OS X combinations followed from upgrades introduced by Apple. Although a usable Linux system resulted from the standard distribution, specific techniques and commands were used to overcoming some of the limitations which remained, so as to customize the resulting system for specific work.

1 Introduction

The Mac Mini 2011 was the fifth generation release of the Mini version of the Macintosh range by Apple. It was released in July 2011 in three versions; standard, enhanced graphics, and server. Each version was powered by an Intel Sandy Bridge processor, as were the 2011 releases of the Macbook Pro and Macbook Air. As opposed to other Macintoshes, the Mac Mini had no screen, keyboard, or mouse. Those components had to be supplied from other sources. However, it did contain fast processors, and hard disk, together with hardwired and wireless networking. Across the back of the 197mm x 197mm x 37mm aluminium enclosure were sockets for connecting power, gigabit ethernet, HDMI display, Firewire 80, USB 2, and Thunderbolt peripheral equipment. OS X Lion was supplied installed.

The Mac Mini was the low cost entry into the Macintosh line of computers. The standard Mac Mini cost $799, progressing to the highest cost of the Mac Mini server which was the same price as the entry end of the Macbook Air range. But despite the low cost, the Mac Mini hardware was currently the best combination of processor, storage, and input/output connectivity on the market.

The aim was to alternate use of this hardware under either OS X Lion 10.7.2 or Linux. The selection of which system to use was made when the hardware was booted, i.e. the hardware was to be made to dual boot.

1.1 Why put Linux on a Mac?

We were told OS X was basically a Unix system. By and large this was true. But a Unix system with bits – important bits – left out or modified. The graphical desktop is predefined in OS X. The `vim` editor was on Lion – but only in a terminal window as opposed to having its own graphical window. Language interpreters for `perl, python`, and `ruby` were present, but compilers such for C and C++ were absent. Also in OS X, the classical `roff` style text formatting system was provided but without a TEX system. For the majority of users of OS X such omissions would go un-noticed. The OS X Lion and Linux environments were different. Selection of which environment to use would occur at boot time; OS X or Linux both being available.

Was it sacrilege to put Linux on a Mac? After all, Mac OS X provided a productive computer system used by millions around the world. It also looks attractive. For a first time Mac user coming from a Linux background, providing both systems on the same hardware provided an appropriate support environment in which to explore OS X. However, the underlying Apple business model, apparently revised with the release of Mac OS X Lion, was a constraint on making an easy comparison. No installation media for OS X software available made tailoring of the system to a specific purpose difficult. Having to supply personal details so as to download program development software, was confronting. The jump from Linux to OS X appeared difficult.

Why was there this interest in using Linux? After all, OS X like Linux is Unix-like, but:

- Linux was an open source and open knowledge operating system;
- Bleeding-edge software was developed on Linux and thus readily installed;
- The windowing system (X Window) was decoupled from the kernel;
- A desktop/GUI was not required or predefined for use;
- Configuration of the kernel and the system could be readily determined;
- Shell programming improved the user's productivity;
- Command line use give in-depth control of the system;
- Linux could process files from a host of foreign formats;
- When desired, open source software alone could be installed on the system.

In short, Linux provided far greater control over the computing environment than OS X allowed.

1.2 Difficulties

All Mac models being sold used Intel processors. On those processors Apple had adopted Intel's Extensible Firmware Interface (EFI) for use within the processor's firmware. By contrast, most other computers, using the same processors, used the older BIOS system. The firmware being used become significant when booting the operating system. Most Linux distributions were setup for booting using BIOS. There were, however, a few Linux distributions supporting EFI. Mac OS X boots using EFI. To have a dual boot, the same boot technology was required for both systems. Since the Mac OS X technology was fixed at EFI, a corresponding Linux system was required.

All three models in the 2011 Mac Mini line-up came without an optical disk (DVD) system. Their firmware would not support booting from an external DVD drive. By contrast, booting from a thumb drive inserted in a USB slot was supported on the Mac Mini. But, most Linux distributions in the form of ISO images, were configured for booting from a DVD (or CD) drive. This contradiction was recognized by the Mac Mini boot firmware and aborted the boot process if a standard Linux installation was copied to a USB thumb drive.

On previous versions of Mac OS X, the installation of dual booting was assisted by the standard OS X `Boot Camp` utility. With OS X Lion, Boot Camp assisted creating a dual boot installation with Windows 7 only. No other version of Windows except 7 would work, and the original installation media for Windows 7 was required.

The Mac Mini 2011 come with OS X Lion installed. Any upgrade required an Internet connection. A new Mac Mini with pre-installed software also required access to the Internet to initially configure OS X Lion. If OS X Lion becomes corrupted, it was replaced over the Internet using the OS X Lion `Recovery Partition` on the Mac Mini. But across a slow Internet connection the transfer of the required 4GB of data took considerable time. However, the Mac Mini 2011 which is supplied with OS X Lion appeared to have hardware common with the Macbook Pro 2011 which came standard with OS X Snow Leopard. Snow Leopard had the Boot Camp utility which enabled setting up a dual boot with Linux. Unfortunately installing Snow Leopard on a Mac Mini 2011 was prohibited by the Mac Mini firmware. So installing Snow Leopard so as to introducing Linux onto the Mac Mini hardware was found not to be an available approach.

Another approach for setting up a boot partition for dual booting used the `rEFIt` tools from `refit.sourceforge.net` in collaboration with OS X Lion's `Boot Camp` utility. This also was unsuccessful. Apple appeared to have changed the EFI on the Mac Mini 2011 so the previously functioning `rEFIt` tools no longer worked, and `rEFIt` had not caught up with such changes. Also, the `Boot Camp` utility had been changed in the way it cooperated in producing a dual boot environment.

A different approach again needed to be taken to provide a dual boot of OS X Lion and Linux.

2 Installation

The hardware used for installing the dual boot was a standard configuration Mac Mini server released in July 2011. It contained 4GB of memory, and two hard disks of 500 GB capacity each. A standard Apple keyboard was attached via a USB port. A Logitech trackball was attached to one of the keyboard's USB ports to provide a mouse. A 23 inch (53cm) Samsung LCD display with a 1920 x 1080 resolution was attached through the HDMI to DVI adapter supplied *in the box* with the Mac Mini.

The two hard disks were known to OS X Lion as `disk0` and `disk1`. OS X Lion was installed on `disk0`. That disk was partitioned into three partitions, one of those being the `Recovery Partition` which was introduced with OS X Lion, another contained the *EFI System Partition* used in booting OS X Lion, and the third contained the OS X Lion system. But OS X Lion did not take up the whole of its allocated partition. The OS X Lion utility `Disk Utility` was used to decrease the size of the OS X partition to approximately 300GB (293GB actually) giving another partition of approximately 200GB (194GB) to act as a partition to be shared by OS X Lion and Linux. After this partitioning, executing the `df` command on OS X Lion showed the partitioning on the disks as:

```
/dev/disk0s2   /
/dev/disk0s4   /Volumes/Common
/dev/disk1s1   /Volumes/NO NAME
```

It appeared OS X Lion hid Partition 1 containing the `EFI System Partition`, and Partition 3 which contains the `Recovery Partition`. Even after disk1 was partitioned during the installation of Linux, this tabulation did not change. This possibly indicates OS X Lion did not understand (or ignored) Linux.

The requirement was to install a functioning Linux system which contained minimal software outside of the requirements for a functioning system. The final purpose of this system was to develop programs and evaluate scientific/mathematical oriented software. The design was to install this minimum Linux system and add software to support the intent of the final system.

The Linux installation image was stored in a 8GB flash drive. The capacity of this drive was far in excess of the CD ISO image used. From a multiple ISO CD Linux distribution, the first ISO CD image was sufficient to install the base Linux system.

2.1 Linux distribution

The `Slackware` and `Slax` Linux distributions were tried initially without successfully installing on the Mac Mini. The Ubuntu distributions were then tried with success. In all cases, the 64 bit AMD directed ISO image distribution was used.

Four distributions of Ubuntu Linux were used: Ubuntu server 12.10, Xubuntu 11.10, and Precise alpha 1 versions of Xubuntu and Ubuntu server. The first two distributions were downloaded from `http://www.ubuntu.com` and `http://www.xubuntu.com`, while the remaining two from `http://www.xubuntu.com` and `http://cdimage.ubuntu.com/releases/precise/alpha-1`. Each was installed in succession, replacing the former on the Mac Mini. Ubuntu server 11.10 was tried first. This functioning system minimized the packages it installed. Potentially, *standard* packages can be of no interest and minimizing their installation was appropriate. However, a server distribution resulted in a system which would function as a foundation onto which other packages could be added to satisfy specific requirements. In contrast, a Xubuntu distribution was taken as a system which included most of what was required in the final Linux system. However they use a *standard* package inclusion which install in excess of what was finally needed. These two distributions provided a comparison between a tailored approach and a *standard* distribution approach in achieving the final ends. The Precise Ubuntu server was the latest server distribution available and used as the final system. This, having the latest kernel, was thought to best support the Mac Mini hardware.

The Ubuntu server distribution appeared best to fulfil the system requirements. That distribution contained no predefined desktop/windowing environment. Although the Ubuntu documentation recommends a windowing system not be used on a server, X could be added onto the basic server installation. A server distribution was also lean. For the Precise 12.04 alpha 1 distribution, the basic server installation resulted in 372 packages being install. The command `du -s /` gave 702285, indicating 702MB of disk space was absorbed by the installation. This installation did not include X Window nor gcc. Installation of those components after completion of the initial basic installation gave a total of 388 packages installed (as reported by the `attitude` utility) giving 1030349 (1,031MB) in response to `du -s /`. The installation of the Xubuntu distribution of the same Precise 12.04 alpha 1 release showed 2028857 (2,029MB) disk space was absorbed, but that system included both X Window and gcc. The Xubuntu installation also included components which were not wanted in the final installation. By comparison, the standard OS X Lion system gave 11394641 (11,395MG) and that was without a C compiler system.

2.2 Setting up the boot thumb drive

The Linux installation media for the Mac Mini 2011 was a thumb drive. This thumb drive had to be appropriately formatted. The current format of a drive while it was mounted on a Linux system was determined by issuing the command:

```
df -T
```

The thumb drive format was required to be 32 FAT. At that time most thumb drives being sold had such a format. If this was not the case, then formatting was needed. Formatting destroyed all data on the drive. So formatting was also used to clear a used thumb drive of its current content.

The first stage of formatting was performed using `fdisk`, which was an interactive program. With the thumb drive unmounted but plugged into a USB port, the command:

```
fdisk /dev/sdb
```

started `fdisk` referencing the thumb drive in the USB ports `/dev/sdb`. Successive commands for `fdisk`:

d	deletes the current partition table
n	creates a new partition table
1	creates one partition on the drive
p	lists the contents of the new partition table
w	writes the new partition table to the thumb drive and exits `fdisk`

prepare the thumb drive to contain the format. At this point, the thumb drive was cleared, but not formatted.

Formatting was performed using the `mkdosfs` program, which was part of the `dosfstools` source package available from `http://linux.softpedia.com`. With the thumb drive unmounted but inserted in the USB port, and with the address of that USB port being `/dev/sdb1`, the command to put a FAT 32 format on the thumb drive was:

```
mkdosfs -F 32 /dev/sdb1
```

This thumb drive was then mounted for subsequent use.

The ISO image of a Linux distribution was written on this thumb drive in a special way. The program `unetbootin`, available in binary form only from `http://unetbootin.sourceforge.net` was used for that purpose. For this program to run, the `mtools` and `p7zip` packages were required to be installed. Installation of the `mtools` source package was straight forward. Installation of the `p7zip` package was more difficult, however, the command sequence:

```
cp makefile.linux_x86_asm_gcc_4.X makefile.linux
make all_test
make install
```

was found to work successfully.

With the thumb drive mounted, the `unetbootin` program was started by the keyboard command:

```
./unetbootin-linux-555 &
```

Execution of that program required X Window to be operating. The Linux ISO image was downloaded and stored in the `/tmp` directory. The `unetbootin` program was directed through its GUI, to take its input from the ISO in, say the `/tmp` directory, and write the files to the device mounted on the `/opt` directory (the thumb drive). This file extraction and copying took several minutes as there were thousands (1733 in one case) files involved, but progress was indicated on the GUI. All this preparation was performed on a 32 bit Linux system, not the Mac Mini.

2.3 Overview of the Linux system installed

In Linux nomenclature the two hard disks of the Mac Mini server were `sda` and `sdb`. OS X Lion called them `disk0` and `disk1`. Linux was installed on `sdb` since `sda` was the standard installation drive for OS X Lion.

With the thumb drive containing the Ubuntu files inserted in a USB port of the Mac Mini, the Mac Mini was booted while holding down the `option` (alt) key of the keyboard. The white Mac screen with several options appeared. The option with the USB symbol and the title *EFI Boot* was selected, and the Linux installation commenced.

In each Linux installation the *expert* install option was used with the same installation procedure followed. Each of the following were installed:

1. Ubuntu server 11.10;
2. Xubuntu 11.10;
3. Xubuntu Precise;
4. Ubuntu server Precise.

Manual partitioning option for the hard disk was used. Hard disk `sdb` was partitioned into 9 partitions. Partition 1 was sized at 20MB for use as the EFI boot. Table 1 shows the partition setup used on that 500GB hard disk of the Mac Mini server. All partitions were formatted as `ext4`. Functioning systems resulted in each case. It was the Precise Ubuntu server which was finally adopted.

Table 1: Structure set on the hard disk

Partition	Size	Mount point
1		EFIboot
2	6	/
3	330	/home
5	3	/usr
6	10	swap
7	100	/home/research
8	30	/tmp
9	21	/usr/local

Two boot procedures resulted from this dual installation. After installing Linux, if the `option` key on the keyboard was depressed at boot up, the Mac Mini went to the standard white Mac boot screen inviting the booting of OS X Lion. Without depressing any keyboard key at boot, the `grub` black boot screen appeared. Booting Ubuntu was an option available there. If that menu option was selected, and the return key on the keyboard pressed, then Ubuntu was booted. Options for booting OS X Lion also appeared in the grub screen. However if selected, OS X Lion **did not** boot. A way around booting OS X Lion from the grub menu was to type a `c` character. This resulted in a `grub >` prompt, to which typing `exit` resulted in the OS X Lion booting in the same manner as when the `option` key was depressed at boot up.

After each installation, dual booting of OS X Lion and Linux was verified. Each of the two systems were tested for stability. Once Precise Ubuntu server had been verified as functional, `gcc`, `g++`, `make`, and the X Window system packages were installed on Ubuntu to fulfil the system required. This functioning procedure was completed on Saturday, 17 December 2011 (11:00am). An announcement of this work was made to the `comp.os.linux.misc` USENET newsgroup on 19 December 2011.

3 Results

This installation produced a usable Linux system. The capacity of the system obtained is summarized in Table 2. Because of the newness of the Mac Mini 2011 system, some hardware present was expected not to be supported by Linux. However, the left column of Table 2 indicates most significant hardware was supported. Most of what was not supported, and listed in the right-hand column, were provided by additional work beyond the Linux distribution. Later distributions of Linux were be expected to improve support for the Mac Mini 2011 hardware.

Table 2: Installed Linux out-of-the-box behaviour

Working	Not working
Ethernet	WiFi networking
USB	ALSA sound system
display graphics	keyboard `beep`
hard disk	keyboard caps lock
keyboard and mouse	

For this implementation of Linux, functioning IEEE 1394 Firewire and Thunderbolt were not tested. Although there was a connection port on the back of the Mac Mini for each, no hardware was available for connecting. However, Firewire was understood to be supported by Linux in general, but Thunderbolt support was not, nor expected soon due to the limited number of devices using it outside of the Mac environment.

4 Modifications after the standard Linux installation

No standard Linux distribution contains *everything*. But rarely is *everything* needed. For example, the Ubuntu server installation, deliberately did not install X Window. It also did not install the `gcc` and `g++` compiler systems although they were part of the Ubuntu server distribution. Even a Xubuntu installation which did include X Window with the `gcc` and `g++` compilers, did not include `gfortran` which was needed to build source packages such as `octave` and `R` from source. But additions could be made to the packages installed by a standard installation using the Ubuntu package management program `apt` which itself was installed as part of a standard Ubuntu installation.

4.1 Using the Ubuntu package management system

The Ubuntu package management system, with the user interface programs `aptitude` and `apt-get`, takes its information about where to find requested files from the `/etc/apt/sources.list` file. After an installation this file points to an Internet archive. However, the installation thumb drive (equivalent to an installation CD/DVD) was an alternative to the Internet for installing further packages. Although this resource was limited, it generally contains more packages than were installed during a standard installation. The thumb drive was used as a preferred resource than the Internet. For the Ubuntu management programs to use this resource, the `/etc/apt/sources.list` file was configured to use this resource. Say this installation media was mounted on the folder `/opt`, then, in the case of a Ubuntu installation of the Precise version, the lines:

```
deb file:/opt precise main deb
file:/opt precise restricted
```

needed to be inserted in the /etc/apt/sources.list file. All other lines in that file were commented out.

In the case of this Precise Ubuntu installation the required gcc, g++, and the alsa sound system were on the installation thumb drive. With the source.list file changed as above, the aptitude program was used to install those packages.

The Ubuntu organisation which produced the Ubuntu Linux distributions also maintained archives of packages for each Ubuntu version. These were accessed across the Internet. For that access, the file /etc/apt/sources.list had to use one of those archives. The archive at http://us.archive.ubuntu.com was used, so lines:

```
deb http://us.archive.ubuntu.com/ubuntu precise main restricted universe
deb-src http://us.archive.ubuntu.com/ubuntu precise restricted universe
```

were used in place of the existing deb and deb-src lines in the sources.list file. The apt-get command-line interface to the Ubuntu package management system was then used to install additional components of the system from this archive.

Before such an archive could be used, the computer on which the installation was performed needed to be connected to the Internet. An Internet browser was not needed on this computer, only hardware/software which accessed the Internet. Displayed time response lines resulting from the command:

```
ping www.google.com
```

provided an indication this connection was available. The Mac Mini had the hardware, and the Ubuntu installation provided the software, for a hardwired Internet connection.

As an example, to install the gfortran package, the command:

```
apt-get gfortran install
```

was used. The apt-get program determined other packages which the requested package depended upon and installed those required. All running of apt was done as root or by using sudo.

4.2 Making the WiFi network operational

WiFi on the Mac Mini was provided by the BCM4331 chip which required firmware to be installed for it to function. This firmware worked in collaboration with the b43 driver which was a module within the Linux kernel. Although the driver was part of the kernel distribution, the firmware was not.

Producing this firmware was done in two steps. First a firmware cutter program was downloaded from http://bu3sch.de/b43/fwcutter. The source code of the program used was b43-fwcutter-015.tar.bz2. This source was compiled and installed by the standard make; make install command sequence. This resulted in the file b43-fwcutter being created in the /usr/local/bin directory. Next the file broadcom-wl-5.100.138.tar.bz2 containing the firmware was downloaded from http://www.lwfinger.com/b43-firmware. Detarring this file produced the broadcom-wl-5.100.138 directory. The linux sub-directory there contained several object files – the required firmware. To install the BCM4331 specific firmware the command:

```
b43-fwcutter -w /lib/firmware/tmp/broadcom-wl-5.100.138/linux/wl_apsta.o
```

was used, where `/tmp/broadcom-wl-5.100.138/linux/wl_apsta.o` was the absolute path of the firmware file to be installed. All this was done as `root`, or by using `sudo`.

Upon completion of the above, the command `lsmod` showed the module `b43` as being loaded. The `b43` was the module which controlled the Broadcom Corporation BCM4331 WiFi chip on the Mac Mini server (as shown by the command `lspci`). It was made operational by the following. An entry for the WiFi network was was made in the `/etc/network/interfaces` file which specified all network interfaces on the system. The existing lines pertaining to the `eth0` interface were duplicated. In the duplicate `eth0` was changed to `wlan0` and the entries `address, network`, and `broadcast` were changed to correspond to those desired for the WiFi network.

Linux was then rebooted for the above changes to be put into affect. Towards the end of the boot up sequence, the Ubuntu-red screen appeared on the display, and then after approximately 15 seconds a text notification lines:

```
Waiting for network configuration...
Waiting up to 60 more seconds for network configuration...
```

appeared, those lines being separated by approximately 60 seconds. The system then completed a normal boot up.

After logging into the system, the `iwconfig` program, which was part of the standard Linux installation, was run to associate the tag `wlan0` to the WiFi network. An example of this is:

```
iwconfig wlan0 essid "YenolamNET"
```

where the WiFi interface tag `wlan0` is requested to be connected into the `YenolamNET` WiFi network.

Subsequent boots of Linux stalled towards the end of the boot sequence. It then repeated the behaviour noted above but without the appearance of the Ubuntu-red screen. But as above, the boot's completed after about a minute delay, this delay being stated to be due to network configuration. The `iwconfig` command also needed to be used after each reboot for the Mac Mini to participate in the WiFi network.

4.3 ALSA sound

The Mac Mini had an Intel High Definition audio controller chip which implemented the sound system used to play audio files and sound tracts of DVDs. This was supported by the ALSA system which was standard part of a Linux distribution. The command `lsmod` would list currently loaded modules and those with `snd_` in their name implemented the ALSA system for the Mac Mini. Although all component modules which provided support for the system components of controller, card support, etc. appeared to be present, the system did not function.

Utilities such as `aplay` to play a sound file, `arecord` to record sound into a file, `amixer` to make command line adjustments to the sound system, and `alsamixer` to adjust the system via a graphical interface were included in the standard installation. The graphical interface of the `alsamixer` program was curses based. When it was executed, `00` was shown under the bars showing the volume setting of the `master` but `MM` under the corresponding bar for `surround speakers`. The `MM` indicated *muting* of that function. By moving the cursor to the `surround speaker` and press the m key on the keyboard, this un-muted the surround speaker. The system then worked. More volume could be obtained by repeated pressing the up-arrow key on the keyboard while the cursor was positioned under the component's bar. Once alsamixer was exited by pressing the Esc key on the keyboard, the aplay and arecord programs could be used to verify the system was working.

4.4 PC Speaker

The PC speaker was an alarm signal. A keyboard action such as a *left arrow* key at the start of a line, or a *down arrow* key were errors and would result in a *beep*, the beep produced by the PC speaker. It was also know as the ASCII bell character (decimal 007) and would be heard when control g was given at a command line. Alternately, the shell command:

```
echo -e "\0007"
```

would also generate a bell sound.

On the Mac Mini the PC speaker was part of the ALSA system. This was shown by the command aplay -l where cards 0 and 1 were listed; card 1 being that of the PC speaker.

The PC speaker was implemented as two loadable modules named pcspkr.ko and snd_pcsp.ko. Generally, only one of would be loaded at any one time. The pcspkr.ko module was used when the PC speaker was independent of the ALSA audio system, and snd_pcsp.ko when it was part of ALSA. The standard Ubuntu system inhibits the loading of both PC speaker modules by commenting out blacklist pcspkr and blacklist snd_pcsp in the /etc/modprobe.d/blacklist.conf file.

4.5 Keyboard caps lock

Pressing the keyboard cap lock key did not set the LED on the key indicating it was in use. This occurred when using the shell interface to the Linux kernel. If a X Window window manager was used, the LED behaved normally. However, whether the LED was lit or not, the purpose of the cap lock behaved as expected.

4.6 System monitoring

The aluminium housing of the Mac Mini become warm (estimate to be 45degC) when operating normally under Linux. It appeared warmer than when operating under Mac OS X Lion. This may be due to more power being applied to the processors under Linux than under Lion. Linux sees 8 CPUs on the Mac Mini server (as shown in the directory /sys/devices/system/cpu). They correspond to the 4 processor cores, each supporting 2 hardware threads. The frequency of operation of these 4 CPUs could be changed. This was known as *turbo boost* whereby the frequency of operation of a CPU was increased for a short time to hasten processing. But higher operating frequency resulted in greater heat generation. The Linux kernel kept and used information about operation of each CPU through a directory structure accessible only by root. It contained a sub-directory for each processor, named CPU0 to CPU7. Each of those sub-directories contained a cpufreq sub-directory which contained the variables:

cpuinfo_cur_freq
cpuinfo_max_freq
cpuinfo_min_freq
cpuinfo_transition_latency
related_cpus
scaling_available_frequencies
scaling_available_governors
scaling_cur_freq
scaling_driver
scaling_governor
scaling_max_freq
scaling_min_freq
scaling_setspeed

The contents of all the entries in this `/sys` directory were used directly by the kernel. Changing a value of these variables effected the operation on the kernel. Different manners of changing those variables were available. The operating frequency of a processor could be changed by a software governor. There were 5 available CPU frequency governors for the Linux kernel. As described by `help` when configuring the kernel, these governors set the CPU operating frequency as shown in Table 3.

Table 3: Function of available Linux frequency governors for CPUs

Governor	Frequency setting performed
performance	highest available CPU frequency
powersave	lowest available CPU frequency
userspace	a manual value or from a userspace program
ondemand	a value determined by a dynamic policy
conservative	a value determined by a dynamic policy but slower

In the standard Ubuntu server Precise distribution all those governors were built into the kernel as opposed to being loadable modules.

As an example of the use of this CPU and governor information, to set the governor on CPU 4 to powersave, the command:

```
echo powersave > /sys/devices/system/cpu/cpu4/cpufreq/scaling\_governor
```

was used. The command:

```
cat /sys/devices/system/cpu/cpu1/cpufreq/scaling\_governor
```

was used to see what was the current governor of CPU 1.

In general, for each CPU numbered X the system kept information on that CPU. Such information had the directory `/sys/devices/system/cpu/cpuXi/cpufreq/` as its base. For each CPU there was a discrete set of CPU frequencies which could be used, between the minimum contained in `cpuinfo_min_freq` and the maximum contained in `cpuinfo_max_freq`. Those frequencies were tabulated in `scaling_available_freq`. For each CPU, the time spent in each frequency was contained in `stats/time_in_state` while the number of transitions between each frequency was contained in `stats/trans_table`.

A reasonable approach to changing such variables was automatic in response to changes in processing conditions. The standard means of doing that was by the `cpufreqd` daemon. However, the `cpupower` project is actively being developed as a userspace extension on it. From version 3.1 of the linux kernel, code of the `cpupower` was included under the `tools/power/cpupower` directory of the source distribution. Building of `cpupower` had to be done separate to building of the Linux kernel.

To build `cpupower` required `pciutils` and its shared libraries. To generate those shared libraries when building `pciutils`, `SHAR` in `Makefile` was set to `yes`.

Linux kernel 3.2.5 source was used to build `cpupower`. The `tools/power/cpupower` directory of the kernel source contained the required source. Many problems were encountered in building `cpupower` from such source. Upon resolving those problems, the sub-directory `utils/` contained the `cpupower` executable and the `bench/` sub-directory the `bench` executable, which was also of interest. Information available from `cpupower` was obtained from the command:

```
./cpupower help
```

Based on that information, to determine the state of CPU 1, the command:

```
./cpupower -c 1 frequency-info
```

was used. By contrast the command:

```
./bench --sleep=50000 --load=50000 --cpu=1 --cycle=10 --verbose
```

executed a benchmark simulation process using CPU 1 with verbose output to the terminal.

For the best combination of processing speed and low power dissipation the `ondemand` governor offered the best choice. This was the default governor. Although the above `frequency-info` command give a reasonable display of the frequency behaviour of a CPU, the command:

```
cat /sys/devices/system/cpu/cpu1/cpufreq/stats/time
```

tabulated the time the CPU has spent in each frequency step available.

There remained a need from a visualization to show the frequency behaviour over time of all CPUs present. Spot checking using the manual techniques given here of the time CPUs spend in each frequency suggests processing is not being distributed across processors. Maybe this indicates need for software which automatically splits a computing load into parallel streams with each stream going to a processor. Multiple processors will then deliver more value for their cost.

5 Concluding remarks

The Mac Mini 2011 server was a significant advance in hardware available in the commercial market. Understandably Apple with the release of OS X Lion, which occurred concurrently with the Mac Mini 2011 release, actively constrained software which can be executed on such hardware. But, as opposed to the standard way of doing things imposed by OS X, Linux offers and alternative. For example only one is available under OS X, but a choice of window managers which can use floating, monocle, and tiled placement of windows on the screen are available under Linux. Different windowing is advantageous under different purposes to which the computer is put. There are many other reasons why Linux might be chosen. But there are also situations where OS X is the better choice. The best of both worlds become available in dual booting OS X Lion and Linux.

Most Linux distributions are oriented towards Intel processors which use BIOS. The Mac Mini 2011 used Intel processors using EFI which is completely different, and newer, than BIOS. This single difference makes installing Linux on the Mac challenging. Versions of OS X prior to Lion provided assistance to overcome this difficulty, but that assistance had disappeared with Lion. This work showed that difficulty can be overcome which enabled Linux to be run in native mode on Mac Mini 2011 hardware.