S129 Computing Cluster User's Guide

Julia Dietlmeier and Seán Begley

julia.dietlmeier@eeng.dcu.ie
sean.begley@eeng.dcu.ie

Dublin City University

RINCE/CIPA/VSG

February 27, 2008

Contents

1 Preface

The foundation for the computing cluster in S129 is the ParallelKnoppix software package. This User's Guide is not a substitute for the Tutorial on ParallelKnoppix, written by Michael Creel [1]. It is recommended that anyone who is starting to work on the cluster for the first time should have a copy of Michael Creel's Tutorial together with this User's Guide. It is also recommended to review the supporting resources listed in the "Bibliography" section at the end of this document. To setup and to configure the cluster in S129 a user will need the ParallelKnoppix Live-CD and this User's Guide.

2 Introduction

The main objective of setting up the computing cluster in the RINCE/CIPA vicinity is to provide an integrated hardware/software platform to solve computationally extensive numerical problems. The existing computing facility in S129 consists of 8 high-performance Dell 490 Precision workstations specifically designed to support image processing applications [6]. Which parallel programming environment would suit our hardware setup at best?

We have focused on high-level interpreted languages as they offer a more intuitive way of programming when compared to the compiled languages. A code written with an interpreted language can, for example, fit into one line but at the same time accomplish a lot of work. From the variety of commercially available Matlab-based parallel programming tools we took a closer look at **StarP** from *InteractiveSupercomputing* and **D**istributed **C**omputing **E**ngine (**DCE**) and **D**istributed **C**omputing **T**oolbox (**DCT**) from *Mathworks*. Both products offer a user-friendly way of parallel programming in the sense that the developer does not need to worry about the master-slave communication between cluster machines. StarP would be a clear favorite as it supports Matlab, Python, R and C environments. But its major limitation is its primary compatibility with the expensive commercial platforms like SGI Altix 8-P server. As a stand-alone product Star-P does not support distributed and parallel computing on the self-wired platforms. DCT and DCE from *Mathworks*, on the other hand, are specifically designed to support distributed computing but would limit us to the Matlab environment.

The tradeoff between price and flexibility is the **P**arallel**K**noppix (**PK**) software. The choice of using the ParallelKnoppix Live-CD is based on its relatively easy overseen configuration and maintenance as well as its versatile support for parallel numerical computation. As to date, ParallelKnoppix offers to a parallel programmer the choice of using Octave, Python and C environment to develop her/his applications. Matlab users will find that programming in Octave environment is not much different. If you are lucky, you can execute your Matlab code in Octave without modifications as even the file naming convention ***.m*** is the same. PK makes use of the existing hardware and supports heterogeneous environments. And, as Michael Creel claims, it should take no more than 5 minutes to setup a cluster.

ParallelKnoppix requires that the user has knowledge of using **M**essage **P**assing Parallel **I**nterface (**MPI**) programming. This is the most challenging part as *You* will have to tell the cluster machines how to communicate, which part of the code to execute and how to assemble the end result.

ParallelKnoppix supports single-user configuration. This feature is not necessarily a drawback. Anyone who has a Live-CD can quickly setup a computing cluster. The developer is not required to be physically present in the S129 room while writing her/his parallel code. By using VMware virtualization software in conjunction with ParallelKnoppix Live-CD the user can prototype her/his application on her/his desktop. ParallelKnoppix is not dedicated to providing a stable multi-user environment with job, resources and user account management capabilities. There can only be one PK session at a time. A possible workaround would be to add multiple system users to a single session. This will be discussed in detail in section 8 ”User Management”. At the end - ParallelKnoppix comes free of charge and under the GNU General Public License.

3 Hardware

The introduced cluster configuration will enable a user to develop and prototype her/his parallel application at her/his own desktop and to use the S129 computing cluster mainly for production. Therefore two hardware/software systems should be outlined.

3.1 Desktop

This is the local desktop, which can be used to develop and prototype an application and to remotely access the cluster. The hardware environment should be considered heterogenous. Users with either 32-bit or 64-bit hardware and a choice of supporting operating systems should be able to connect to the cluster.

The following suggestion for 32-bit architectures: the combination of **VMware Server 1.0** and the **ParallelKnoppix 2.7.1** software packages will allow to develop a parallel application in the familiar desktop environment while utilizing the virtualization technology. VMware Server 1.0 supports 32-bit Host Hardware/OS Platform and enables to run for example the Linux guest OS on the Windows host. The ParallelKnoppix 2.7.1 supports either 32-bit or 64-bit architectures. The combination of VMWare Server 1.0 and ParallelKnoppix2.7.1 is tailored to support 32-bit architectures.

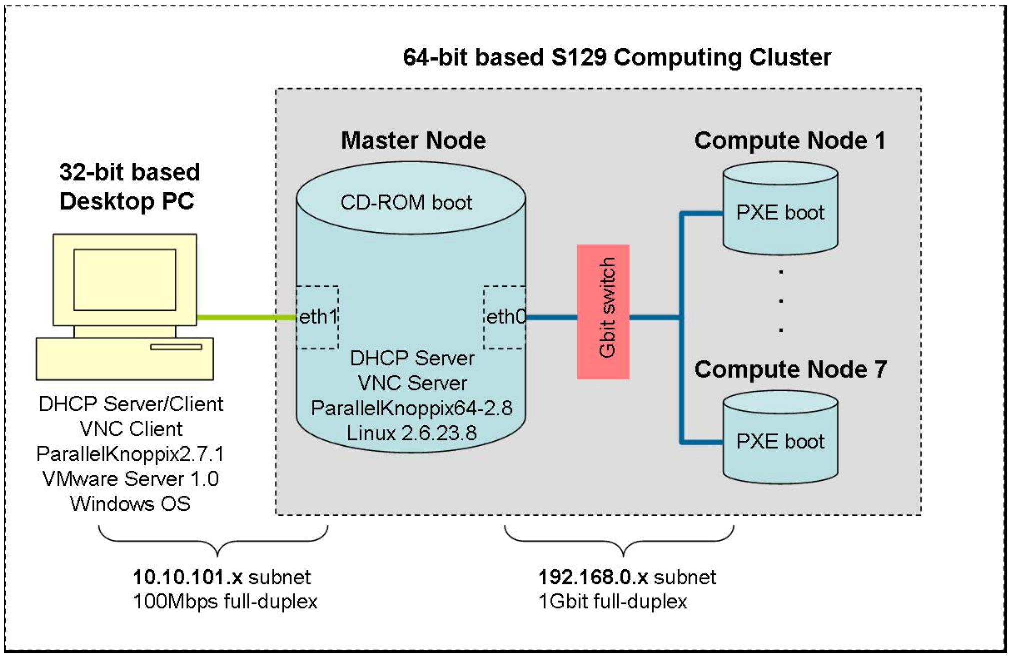

Figure 1: S129 Computing Cluster Configuration

Desktop users with 64-bit based hardware should try using **VMWare Server Beta 2.0** in combination with either **ParallelKnoppix 2.7.1** or **ParallelKnoppix64-**

2.8 packages. The recent Beta 2.0 version of VMWare server however has a Web based management interface which brings up security and network stability issues.

3.2 Cluster

This is the computing cluster in S129 which will be used for production. The cluster consists of eight 64-bit Dell Precision 490 workstations each equipped with a Quad-Core Xeon-based CPU and two 500GByte Hitachi SATA hard drives. All workstations in S129 have Windows XP Operating System installed by default. The cluster will be an ad-hoc Linux cluster which will be using the **ParallelKnoppix64-2.8** version of ParallelKnoppix. It is recommended to refrain from using virtualization software like VMware on the Master Node as it will considerably slow down the overall performance.

A cluster is generally defined as a set of one Master Node (which acts also as a server) and multiple Compute Nodes connected through the network. Parallel-Knoppix makes the cluster flexible and resizable. Given that the total number of available workstations is limited to 8, we could have for example ”1x7” mode: 1 Master Node and 7 Compute Nodes or we could have ”2x3” mode: 2 Master Nodes each attached to 3 Compute Nodes. The main requirement is that the cluster network should be isolated from the other networks. During the setup process of the ParalleKnoppix-based cluster, the Master node enables its own **D**ynamic **H**ost **C**onfiguration **P**rotocol (**DHCP**) service and supplies dynamic IP addresses to anyone present on the subnet. There can not be more than one DHCP server present on the network as there will be a conflict in assigning IP addresses.

4 Networking

The main requirement for the failure-free cluster networking is the physical presence of two network controllers on the Master Node.

Once DHCP enabled, the network structure will be the simple **192.168.0.x** C-class network. All nodes included in the cluster will reside within this network. The Master Node will be the only workstation within the cluster connected with the outside network. Category 5 UTP cable is required for 100Mbps operation of the **eth1** device. Two network controllers on the Master Node are:

ID	PK Driver	Description	Speed	Location
eth0	tg3.ko	Broadcom NetExtreme 57xx Gbit Controller w/Tigon3 chip	1Gbps full duplex	integrated
eth1	8139too.ko	RealTek RTL-8139 Fast Ethernet Driver	10/100Mbps full duplex	PCI2.3 compatible

Table 1: Network Controllers on the Master Node in S129

4.1 Remote Connection

The main requirement for the remote connection is the ability to work in heterogenous environment. It should be possible to connect 32-bit machines with 64-bit machines. The suggestion is to use the **V**irtual **N**etwork **C**omputing (**VNC**) service, which will will use the secure **ssh** link to connect between the Linux guest OS running on the user's **VMware** desktop and the Cluster's ad-hoc Linux OS.

VNC is the remote display system which is implemented in a client/server model. It offers a user to view a computing environment not only on the machine where it is running, but from anywhere on the internet and from the wide variety of machine architectures. To be able to use the VNC service the user will have to start the **VNC server** on the Master Node in S129 and enable **VNC client** on his desktop. The detailed description is given in section 7.2.

5 Software

ParallelKnoppix is a live Knoppix-based Linux distribution. Sounds complicated... Everyone has heard about Linux, which is a free of charge operating system introduced by Linus Torwalds in his Master Thesis "Linux: A Portable Operating System" in mid 1990s. It is the free alternative to Microsoft Windows. In order to use Linux you have to install it on your PC. It is also possible to have multiple operating systems on your PC and when switching on the machine to choose which one to boot. Knoppix is a Linux distribution designed to be run directly from a CD/DVD-drive without the persistent install on the hard drive.

Michael Creel [1] has developed ParallelKnoppix which supports an easy creation of a cluster for parallel programming purposes. ParallelKnoppix includes additional LAM/MPI and numerical computation tools like **MPI T**ool**B**ox (**MPITB**) for Octave. In particular, the PK version which will be used in S129 cluster has the following features:

ParallelKnoppix64-2.8

⇒ v2.8 **64-bit** version only
⇒ /home and /root are NFS exported, which makes it possible to add users
⇒ advanced users can mount a storage device at /home
⇒ Linux kernel 2.6.23.8
⇒ **openmpi** 1.2.4, **octave** 2.9.14, **scipy** 0.60
⇒ **numpi** 1.0.3.1, **parallelpython**2.5rc, **KDE** 3.5.8

6 Creating virtual machines in VMware

To start developing a parallel application on your own 32-bit based desktop by using virtualization with VMware you will have to download the required ParallelKnoppix-2.7.1 [1] version as the ISO image and to store it on the C:/ drive. Then download and install the VMware Server 1.0.4 [2]. This product is free of charge but you will need to register for your free serial number. Good explanatory overview on virtual clustering is given in [5], although this document does not entirely support our cluster architecture.

After installation click on the VMware shortcut or from Start ⇒ Programs ⇒ VMware ⇒ VMware Server select VMware Server Console. In order to create your first virtual machine, press ”**Home**” then ”**New virtual Machine**” and use the **”Custom”** mode. Go through the hardware options. Name your machine **”SomeName”**. Understand the terminology like what is your host operating system, which for example could be 32-bit Windows XP. When using ParallelKnoppix2.7.1, your guest OS will be **”Other Linux 2.6 x kernel”**. Create a harddisk for the guest OS or if you did it already before, use the existing virtual one. If you did select **"Custom"** mode at the beginning, you would have been given a choice between either creating a new disk or using an existing virtual. Try the mode **"typical"** and see the difference.

When creating the harddisk partition use the folder
C:/VirtualMachines/SomeName/ to store the **.vmdk** file. Use **LSILogic** consistently when adding new SCSI hard drives. By default you will have one CD-ROM device, use this to boot the **.iso** image. You can also add the second (external) CD-ROM device, which you can use for moving data for example. Here are the settings which will work:

Device	Settings
Memory	200MB
Hard Disk SCSI0:0	Other Linux 2.6.x kernel.vmdk
CD-ROM (IDE0:0)	Using drive D: Don't "Connect at Power On" ! Use physical drive at Host
CD-ROM 2 (IDE0:1)	Using image C:/parallelknoppix-2.7.1.iso
Ethernet	bridged
Processors	1 (2 if you have two)

Table 2: Configuration of the PK **Master** Node virtual machine

Don't select "Connect at power on" for the external CD-ROM drive or the machine will not boot Linux OS from the ISO CD-ROM!

A few words about networking. By default the newly created Ethernet Adapter will be the one with the bridged networking. Do not "Setup ParallelKnoppix" in this configuration or you will get DHCP conflict on the subnet! If you do "Setup ParallelKnoppix" you will be prompted to select device which connects to the cluster. You need to have more than one Ethernet adapter present or you will have to disconnect LAN cables from your machine. If you have two network cards, add the second physical networking device with the **NAT** service in order to enable internet access. You may even add the third, virtual, Ethernet Adapter with the **"Host Only"** option, when creating multiple virtual machines and wiring them together to create a virtual cluster. This will be described in the next section.

That's all, you should be able now to power on your first virtual machine.

When powered on the virtual machine you will boot ParallelKnoppix. Can you count the penguins? There will probably be 2 Penguins on most desktops as the majority of you has dual-core machines. Each penguine is for one CPU present on the desktop. You don't have to enter any cheatcode as the VMware takes the job of adjusting PK settings to the video and graphic hardware of your PC.

Once powered on in this configuration you will have one PK **Master** Node in your virtual cluster. You will have the familiar **K D**esktop **E**nvironment (**KDE**) screen and if using **Ctrl+Alt** keys you can move the mouse between the Windows and Linux screens.

6.1 Virtual cluster on one physical machine

If you would like to add "compute" nodes to your virtual cluster you will have to create additional virtual machines. Each virtual machine, including the **Master** Node, will need to have two hard drives: the local **SCSI0:0** and the shared **SCSI0:1**. The shared hard drive on each machine should point to the same location. You will also have to modify the configuration **.vmx** file "C:/Virtual Machines/SomeName/Other Linux 2.6 x kernel.vmx". Try the following settings to enable the second shared hard drive:

SCSI0:1.present = "true"
SCSI0:1.sharedBus = "virtual"
disk.locking = "false"

Name the machines for example **PK Compute** and **PK Master**.

For the Compute Node you will need only one "host only" Ethernet Adapter. Use ISO image located on the C:/ drive for CD-ROM (IDE) settings. Add hard drives as described before. For the PK Master virtual machine add one Ethernet adapter with "bridged" networking and the second Ethernet Adapter with "host only" networking option. Check "Connect at power on". We will use the "bridged" ethernet adapter for internet access and the "host only" adapter to wire the virtual machines in our virtual cluster.

Device	Settings
Memory	200MB
Hard Disk SCSI0:0	Other Linux 2.6.x kernel.vmdk
Hard Disk 2 SCSI0:1	Other Linux 2.6.x kernel-000001.vmdk
CD-ROM (IDE0:0)	Using drive D: Don't "Connect at Power On"! Use physical drive at Host!
CD-ROM 2 (IDE0:1)	Using image C:/parallelknoppix-2.7.1.iso
Ethernet 1	bridged
Ethernet 2	Host-Only
Processors	1 (2 if you have two)

Table 3: Configuration of the PK **Master** Node virtual machine

The location of the shared disk in this example is given by C:/Virtual Machines/PK Master/Other Linux 2.6.x kernel-000001.vmdk. The Compute Node does not have its own hard drive in this example, but you can add an additional SCSI drive when needed.

When finished creating both machines, power on PK Master at first, see how it is booting in PK **Master** Node. When the KDE is up, open Konsole and type in **ifconfig**, which will give you information about activated network interfaces. Most likely you will see that **eth0** has the IP address in the range of 136.202.36.x or 10.10.101.x. This is the device on the DCU's subnet, which connects to the outside world. The **eth1** has the IP in the range of 192.168.x.x. You should use **eth1** when connecting your virtual machines together.

Go through the **"Setup ParallelKnoppix"** procedure, select **eth1** device to connect to the cluster and when prompted to switch on your compute nodes, go and power on your second virtual machine. And this one should boot in PK **Compute** Node. Now you have setup your virtual cluster and you can develop and test you parallel code before you upload this on the real cluster in S129.

The small difference will be that if using the virtual cluster, unless you'll write a sophisticated MPI programm that will make use of your either dual- or quad-

Device	Settings
Memory	200MB
Hard Disk SCSI0:0	Other Linux 2.6.x kernel-000001.vmdk
CD-ROM (IDE1:0)	Using image C:/parallelknoppix-2.7.1.iso
Ethernet	Host-Only
Processors	1 (2 if you have two)

Table 4: Configuration of the PK **Compute** Node virtual machine

core CPU, you will not have any gain in computational performance. The virtual cluster is a pretend cluster.

6.2 Virtual cluster across two physical machines

If you wish to do better and still stay at your desktop, you can create a small ”real” personal cluster. Use your desktop PC and a laptop. The requirement for this is that your Master Node, whether this is your desktop PC or your laptop MUST have two network cards. One for the cluster and the second for the internet access. Don't forget that your cluster subnet should not be bridged to the external one! If you do not intend to use the internet and/or have only one networking device, disconnect your cables from the LAN ports. Use a crossover cable and connect PC and laptop together. In Windows setup a small office network and check your LAN connection and your IP address. If the connection does not work, try to disable Firewall settings on both machines. Install VMware on both. Store the same version of ParallelKnoppix2.7.1 on the C:/ drive. Create virtual machines on both of them. Use bridged networking, you don't need NAT in this configuration. Power on the first virtual machine let's say on your PC. In Windows's Command Prompt, check your IP address by typing in **ipconfig /all**. As you are disconnected from the rest of the world your IP address will be the fixed one assigned to your networking device, something like 168.x.x.x or 154.x.x.x. and not the one from the DCU's subnet like 136.206.26.x or 10.10.101.x.

You are now ready to start setting up the PK cluster and this will start the DHCP service and assign a new IP address to your machine.

You will have to select the right networking device which connects to the cluster and then to choose the right Linux driver for that device. The default selection will probably work but if not: Try to figure out the manufacture of the network card. This could be seen in Settings ⇒ Control Panel ⇒ Network Connections. then double click on your network connection which connects to your cluster and on Properties. See the entry for ”Connect using”. In the worst case this will give you the manufacturer of the card but still no clue about the right .ko driver. In this case try to figure out if the **.ko** driver is linked to the chip manufacturer. The integrated Gigabit Controller on the Master Node in S129 is given as Broadcom Netextreme 57xx but the listed bn2x.ko driver won’t work with this selection. The controller chip on the Broadcom Netextreme 57xx card is the Tigon3 device and the tg3.ko driver is the right option to choose.

Once the driver is selected and confirmed, the DHCP service will be configured.

In Windows Command Prompt check your IPs once again with **ipconfig**, if there is no change, type **ipconfig /renew** and you should be able to see that the IP address did change to 192.168.0.x by now. Check and renew your IP address on your laptop. Then if asked to ”go and turn on your compute nodes” power on the virtual machine on your laptop. This one should boot as PK **Compute** Node. Now you have a real cluster consisting of 2 machines which run Linux kernels on Windows hosts using VMware virtualization technology.

A word on how to enhance the computing performance of your desktop. As mentioned before you can write an MPI program which can use the multi-core capabilities of your CPU. You can also check the BIOS/Performance section. Newer workstations can have some entries worth experimenting with. Something like Virtualization On, HWPrefetch and so on.

One last good tip on how to move your files from your Windows share to your Linux share:

In ParallelKnoppix, open Konqueror ⇒ Go ⇒ Network Folders ⇒ SAMBA shares, then connect to the shared folder on your machine.

7 Working with the S129 cluster

7.1 Live-CD

To make your own Live-CD containing ParallelKnoppix software, download the ISO image from the listed web sites in the PK Tutorial [1]. Check the md5 checksum. Whichever Burner software you will use, follow instructions on making Live-CDs. If for example, using Nero software select Start ⇒ Programs ⇒ Nero OEM ⇒ Nero Burning ROM SE. From the Panel choose Recorder ⇒ Burn Image and select the required **.iso** image. Select the lowest possible writing speed and the setting "Disk-at-once" to ensure the best possible quality of recording.

7.2 Master Node

As we run an ad-hoc Linux cluster from the Live-CD, all cluster configuration procedures have to be redone every time the user is starting a new ParallelKnoppix session. The presented configuration procedure is the required minimum on tasks, a user should perform every time setting up the cluster. Every single step offers a user a chance to understand the course of events and if needed to troubleshoot the session. It is not recommended to replace this configuration procedure by a start-up script. However for her/his own reasons, an advanced user can possibly create a configuration file *myconf.conf* or a script and store it on the hard disk or on the USB drive. In addition, the user will have to "remaster" [1] the Live-CD in order to include the link to this configuration file. For now

Be sure that the Boot option in BIOS is set to CD-ROM drive. Power on the machine and hold down the F2 key. You will enter the Setup BIOS menu. Go to the "**Boot sequence**" section and move the "**Onboard or USB CD-ROM Drive**" line (Shift u/d keys) to the front of the list. This will make the CD-ROM drive the choice number one for the machine when it is trying to find an operating system. If there is no disk in the CD-ROM drive, the machine will work down the list and at some point boot Windows XP from the SATA Hitachi drive. Insert Live-CD into the CD-ROM drive of the master node. Save and Exit BIOS menu.
The master node should boot Linux OS immediately. The Screen "Welcome to the ParallelKnoppix64" should appear. Enter the following cheatcode:

knoppix vga=normal xmodule=vesa

If you will not start typing in the cheatcode then after approximately 1 minute after the screen appears the machine will boot with default (hidden) settings. The issue with this is that the default settings will not support the video and graphic capabilities of the Dell Precision 490's workstation.

Troubleshooting:

- if the machine does not boot Linux, check BIOS setting and if necessary change the 1st Boot priority to the CD-ROM drive
- if the machine starts booting Linux but after entering init5 (graphic) mode swithes to the black, sometimes flickering screen - check the cheatcode.

If failure-free the machine will boot in **PK Master Mode**. You will also see a list of detected CPUs on board. Should be 4 present - that's our quad core! Don't worry about failed dhcpd3 setup - that's a good message, we will setup DHCP later manually. Wait a while until the KDE and the Konqueror screens appear. We will configure the internet and the remote access at first and then will proceed with setting up the cluster and booting up the compute nodes.

2. Call in the Konsole and check the network configuration, whether the second networking device has been recognized. Enter **ifconfig -a**. Device **eth0** is the integrated NIC and the device **eth1** is the RealTek PCI Card. If the **eth1** is not active:

login as the superuser with the **su** command.

to activate **eth1** PCI card: type in **ifconfig eth1 up**

If there is no IP address assigned to the **eth1** interface, check the entry in the **interfaces** file:

cd /etc/network
vi interfaces

enter the additional term **eth1** in the line **auto lo eth0 eth1**. Check if the following is present:

```
iface eth1 inet dhcp
ifconfig eth1 up
```

Exit insert mode, save and exit the file. Then stop, start and restart the networking. Enter

```
/etc/init.d/networking stop
/etc/init.d/networking start
/etc/init.d/networking restart
```

After that check the IP address of the **eth1** interface once again. The IP address now should be in the range of **10.10.101.x**. You have to configure proxy settings for the **eth1** interface. Start Konqueror ⇒ Go ⇒ Settings ⇒ Internet/Network ⇒ Proxy. Check **"Manually specify the proxy settings"**, **Setup**. Change to **proxy.eeng.dcu.ie** and Port=**80**. Check **"use the same proxy for all protocols"**. Apply the settings, start the new Konqueror session and test the internet access.

We proceed with setting up the cluster:

3. Press the second button on to the right (Penguins) on the KDE panel. Select **"Setup ParallelKnoppix"**. Follow the process. You will see a list with networking devices available. Note that the **eth0** does not have the IP address in the range **"192.168.0.x** yet. That is fine as we did not start the DHCP service on the cluster subnet yet. Observe the settings of the **eth1** PCI card. The speed mode should be **"full-duplex at 100 Mbps"**. Select the interface device which will connect to the cluster. This will be **eth0**. You will prompt to select the Linux drivers to support the existing NIC. Select **tg3.ko Broadcom Tigon3** which supports the integrated NIC. Press **Ok**. The ParallelKnoppix will enable and configure the dhcp-service on the cluster subnet. You will see the message **"starting dhcpd server... running"**. For advanced users - if you ever need to disable the DHCP server for a while, do the following as root (**su**):

/etc/init.d/dhcp3-server stop
to restart
/etc/init.d/dhcp3-server restart

If you do it before pinging the compute nodes and setting up the HPC cluster - it's okay. If you will disable the DHCP server while you have the live connections to your compute nodes - you will loose your work as you will shutdown the cluster subnet. After you manually restart the DHCP server, you will have to **”Restart HPC”**.

Check the IP address of the **eth0**. Your master node is the DHCP server now and the IP address should be like **192.168.0.1**

You will be asked to enter cheatcode for your compute nodes. This should the (almost) same as for the master node:

vga=normal xmodule=vesa

Don’t type in **knoppix** here. Now you will be prompted to go and switch on your compute nodes. Presumed that the NIC and PXE boot settings are enabled, the compute nodes will automatically boot Linux in **PK Compute Mode**.

7.3 Compute Nodes

For the compute nodes to be able to boot the operating system from the server (Master Node) over the network the integrated network controller should be enabled with the PXE option. Switch on the machine, hold down F2, go to Onboard Devices $\Rightarrow$ Integrated NIC, enable ”**ON w/PXE**”. Go to ”Boot sequence” and use Shift+u/d keys to move the Integrated NIC boot option into the first line. Save and exit the BIOS menu.

7.4 Shutdown procedure

To shutdown the cluster, select ”Shutdown PK” from the PK menu and confirm. This will shutdown your compute nodes and switch off the machines. To shutdown the Master node, use selection ”logout” and choose ”turn-off”. Linux enters init6 mode, which is the shutdown mode. You will be prompted to take the Live-CD off the CD-ROM drive and to press ”Enter”.

8 User Management

As outlined before, ParallelKnoppix is developed to support a single-user environment. Here we consider a case where a user has developed her/his application and would like without leaving her/his desktop to upload the file onto the cluster and to run the application. It is also possible to add multiple system users as described below.

8.1 Adding System Users

In order to give multiple users access to a single PK session we can create multiple desktops. This should be done on Master Node machine in the S129 at the root level. By default there is only one session user ”knoppix” present on the desktop. As an example we will add a system user called ”elmo”.

adduser - -system elmo - -home /home/elmo or if you wish to place ”elmo into the same group as ”knoppix”:
adduser - -system - -home /home/elmo - -shell /bin/bash - -gid 1000 elmo
to check the group and user ID numbers on KDE select System ⇒ KUser ⇒ User-Manager. Assign the password to the new user with

passwd elmo

Note that when adding a system user, the newly created directorie /home/elmo is empty. The SKEL contents are not copied as we do not intend to start a separate session.

Now we provide login screens with XDMCP. To do this open the file

vi /etc/kde3/kdm/kdmrc

and set **[Xdmrc]** to **Enable=true**

8.2 Uploading files onto the cluster

The user has to know the current IP address of the PCI Network Card **eth1**. If, for example the IP address is **10.10.101.38** and the proxy settings are adjusted, the user can invoke Konqueror and in the address line type in:

fish://elmo@10.10.101.38

You will connect to the home directory of ”elmo” on the Master Node in S129 and can upload your application file, for example *parallel.m* using copy and paste.

8.3 User’s desktop

To run the *parallel.m* remotely, the desktop user ”elmo” has to start remotely the VNC server through the secure connection shell **ssh** and the command **vncserver**. Type the following in the Linux Konsole:

ssh elmo@10.10.101.38 vncserver :1 - - geometry 1024x768 –depth 24

You will create a desktop **:1** on the Master Node in S129 and start the VNC server on your desktop with the optional settings of geometry and depth. Please, see the **man - -vncserver** for a list of all options. The geometry and depth fields are given as an example. If you receive a warning message that the desktop **:1** is not available, start numbering with **:2**. If successful you will be asked to enter your password to enable the connection. And if successful again and if logging in for the first time, you will be asked (that’s because we did the XDMCP setup before) to provide the password to access your desktops. It will look like this:

elmo@10.10.101.38 password:

Enter this and verify.

Start the VNC client service by entering

vncviewer

You will be prompted for an IP address or a host name which you wish to connect to. Enter

10.10.101.30 :1

Enter and then enter your passwords. If that all works you will see a new screen appear and this will be the one of the Master Node in S129. You can adjust the appearance of the Konsole through Settings ⇒. To start your Octave application, change to the directory where it is stored and run octave *parallel.m*. To see which nodes are present in the cluster type in ksysgrd. If you'll find out a more user-friendly way for the remote access, please share your knowledge.

9 Creating parallel code in Octave

The main aim of using ParallelKnoppix and octave together is of course to do computations in parallel across many different computing nodes. This will yield significant time savings for each user wishing to run large simulations. At present, the most suitable task for which to use parallel octave, is the repetitive computation of some function in a loop like the loop in the example below.

```
% The loop to be carried out
for i=1:1000,
   for j=1:1000,
      c(i,j)=test(i,j);
   end;
end;

% The function that is to be repeated
function r=test(i,j),
a=0.1*i;
b=1.1*j;
r=a+b;
```

With just a few minutes of work on your function and wrapping code, you could save over 90% of the execution time if running the code on a number of slave nodes. The necessary steps to parallelise your code are outlined in the following sections. For a quick start guide, just read ”The Function” and ”Quick Start”, but for a more in-depth understanding, you should read the final two sections ”To Parallelise your code” and ”Dissection of Send_Cmd.m”

9.1 The Function

In contrast to the example ”test” function above where the function accepts the loop indices directly, your function should be written so that it accepts a cell array as its input. The reason a cell array is used is that it is easier to pack a cell array into a message when it comes time to send the data. The loop variable(s) should be operated on within the function. The loop variable(s) should also be returned from the function, so that when your calling program is receiving messages from slave nodes, it can reconstruct matrices correctly using the returned loop variables and the associated results. Here, the first two elements of r are set to be the loop variables (first two elements of the input f_args). The third and subsequent elements of r can be used to store your results. Of course, if you are using just one loop index, you can write your code so that only the first element of f_args is used to contain the loop index and all other elements can be used to store the results.

```
function r=test(f_args),
% Retrieve the loop variables and do your calculation
a=0.1*f_args{1};
b=1.1*f_args{2};
% Store the indices in the output array
r{1}=f_args{1};
r{2}=f_args{2};
% Store the result of your calculation
r{3}=a+b;
```

9.2 Quick Start

To quickly get your function, that you created above, working on a number of different slave nodes, simply edit parallel_compute.m. The following aspects need to be changed.

Change k, the number of slaves, to the required number.

Change ITER_I and ITER_J, the number of loop iterations, to the required number. Note, if you only want one loop to run, you can set ITER_J to 1.

You may change the amount of work done by the master node by changing the fraction of the work it computes. To do this, uncomment the required line, and set the value to what you require 0.0-1.0.

Change the word ”test” in the example code to the name of your function.

It should appear in two places, once where the messages are being sent, and the second time where the master's work is being carried out.

If you have to pass extra arguments to your function, change f_args as required.

Save your file as your_file.m and copy it, your_function.m and Send_Cmd.m into a directory on the linux system. Open a command window and navigate to your folder. Run octave with the command ”octave”. Finally, type ”your_file” and your program will be run on a parallel system!

9.3 To parallelise your code

As an alternative to the quick start, some explanation of what is going on in the code is given here.

To replace the simple loop described at the start of this section, some parallel code will have to be used. An example piece of code is given in parallel_compute.m. The easiest and fastest way to write your code will be to alter the example file and save it with a different filename (eg. your_file.m). The example code assumes that two loops are required, but if you require only one loop, you may simply comment out the lines associated with the inner loops, these places are marked in the code with comments. The example code is broken into five parts.

The first part of the code sets the number of slave nodes to be used, the number of iterations in the inner and outer loop, and the amount of work to be done by the master node. Change these according to what you require. The fraction of the total work to be done by the master node is set as $\frac{1}{k+1}$ in the code, where k is the number of slaves. The master node should do slightly less work than the slaves because of the messaging tasks it must complete, but the difference is fairly small. The fraction can however be changed if desired by uncommenting the line of code where indicated in the file. Finally, the communication link is started between the nodes, and some global variables are set up for the message passing interface.

```
% Initialisation
% --------------
k=1; % The number of slave nodes (k)
% The number of iterations in your loop(s)
ITER_I=1; ITER_J=1;
% Sets the fraction that the master will do
MAS_FRAC=1/(1+k);
% Uncomment the line below to manually set the master
% fraction MAS_FRAC=0.5;
% Determines how many iterations the master will do
MAS_ITER=ceil(MAS_FRAC * ITER_I);
% Initialise slave (s) and loop variables (i,j)
s=1; i=1; j=1;
% Set up the network with one master and k slaves
LAM_Init(k);
% set up global message tag, communication world and
% number of slaves
global TAG NEWORLD NSLAVES;
```

The second part of the code is concerned with sending the work to each of the slaves. This is achieved in a loop from 1 up to ITER_I-MAS_ITER (the slaves' portion of the work). Each time the loop is run, the loop variables are packed into a cell array. Then the Send_Cmd function is called. The arguments it accepts are the function name to be evaluated as a string, the cell array of arguments (f_args) and the number of the slave that the work is being sent to. The loop finishes by updating the current slave to sent the data to. The slave number rotates around when it reaches the number of slaves so that more work can be sent to each of the slaves again. The only thing that you will have to change here is the name of the function, change it from ”test” to the name of your function.

```
% Send Work to slaves
% --------------------
for i=1:ITER_I-MAS_ITER,
% Comment the next line and its "end" if using one loop
   for j=1:ITER_J,
% Pack the variables into a cell array
      f_args={i,j};
% Send the command and its arguments to slave 's'
      Send_Cmd('test',f_args,s);
      s=s+1;
% Cycle through the slaves
      if s==k+1,
         s=1;
      end;
   end;
end;
```

The third part of the loop executes the master's share of the work on the master node. The loop is essentially the same as the loop that you started out with, except you are passing in a cell array containing the loop variables rather than the variables themselves. Another variation from the starting loop is that the returned value is contained in the third cell of the returned cell array. The cell array is stored in a variable r which is then accessed to get the return value.

```
% Compute the remainder of the work on the master
%------------------------------------------------
for i=ITER_I+1-MAS_ITER:ITER_I,
% Comment the next line and its "end" if using one loop
   for j=1:ITER_J,
      f_args={i,j};
      r=test(f_args);
      c(i,j)=r{3};
   end;
end;
```

The fourth section retrieves the return values from the slaves. Each slave, on completing its task will send back a message with a cell array named r which contains the loop variables and any results. The program checks in the communication world "NEWORLD" using the Iprobe function for any new messages. A new message will set "flag" to true and a number of parameters are also returned in "stat". A buffer is created to contain the received message. The message is then received using the information contained in "stat". At this point, the message has been received and can now be unpacked, which in our case will return the cell array r. Now the matrix being filled can have the returned value in r{3} put in at position (r{1},r{2}) in the array c. This loop continues until all messages have been received, the number of which should be the same as the number of sent messages.

```
% Retrieve all the results from the slave nodes
% -----------------------------------------------
for i=1:ITER_I-MAS_ITER,
% Comment the next line and its "end" if using one loop
   for j=1:ITER_J,
      flag=0;
```

```
% Look for new messages until one is found
      while flag==0,
         [info flag stat]=MPI_Iprobe(-1,-1,NEWORLD);
      end;
% Initialise a buffer of length stat.len
      buf=blanks(stat.len);
% Receive message from stat.src in NEWORLD with tag TAG
      [info stat]=MPI_Recv(buf, stat.src, TAG, NEWORLD);
% Unpack the variable "r" from the buffer.
      [info retn]=MPI_Unpack(buf,0,NEWORLD);
% Store the return variables from r, store r{3}
% in c(r{1},r{2})
      c(r{1},r{2})=r{3};
   end;
end;
```

The final section of code kills off any communication between master and slave nodes, it closes the communication channels.

Once you are satisfied with any changes that you have made to the code, save the code in a file named ”your_File.m”. Now you can copy that file, the file with your function in it and the Send_Cmd.m file into a linux directory on the master node. Open a command prompt and navigate to your folder. Start Octave with the command ”octave”. Finally, you can run your code by typing ”your_file” at the command prompt. When the code has finished running, the variable c will be available to you in Octave. You may save this to a mat file using the same commands you'd use in Matlab.

```
save filename.mat c
```

NOTE: You may have to add other files to the working directory on the master node depending on what functions your code uses. Octave supports most of the major functions that Matlab has, but one or two optimisation functions have different function names and syntaxes to those in Matlab, but other than these, the function names and usage syntax is the same.

9.4 Dissection of Send_Cmd.m

It is this function that takes care of telling the slave nodes exactly what to do and what to send back to the master. The arguments this function accepts are: the name of the function to be carried out on the slave as a string; the cell array of function arguments to be passed to the named function on the slave; and the number of the slave to send the command to.

The same global variables that were set up in the parallel_compute.m code are also set up here. This allows the functions to communicate properly within the same communication world and with a common message tag.

The actual command that is to be carried out on the slave node is contained within a string. When this string is sent to the slave node, each of the commands contained within the string are carried out exactly as if the text of the string had been entered at the octave command prompt.

```
cmd='r=feval(f,f_args); [info sz]=MPI_Pack_size(r,NEWORLD);
msg=blanks(sz); MPI_Pack(r,msg,0,NEWORLD);
MPI_Send(msg,0,TAG,NEWORLD);'
```

This string comprises of four parts:

`r=feval(f,f_args);`

This evaluates the function f with function arguments f_args and stores the result in a variable named r on the slave. This is just a standard function readily available in Matlab and Octave.

`[info sz]=MPI_Pack_size(r,NEWORLD); msg=blanks(sz);`

This part first determines the size of the buffer that is required to contain the variable r when it is being sent back to the master node. Once the size of the buffer is known, the second command creates a blank message buffer that will contain the information.

`MPI_Pack(r,msg,0,NEWORLD);`

The variable r is packed into the message msg which will be sent back to the master node with address 0 in the NEWORLD communication world.

`MPI_Send(msg,0,TAG,NEWORLD);`

The message msg is sent back to the master node with address 0. The message is sent in the NEWORLD communication world and has a tag TAG to make it easily identifiable.

So the whole command tells the slave node to evaluate the function, store the result, determine the size of the message buffer that is required, pack the result into the message buffer and send the message back to the master node.

The command string has to be sent to the slave node before it can be carried out. For the command string to execute on the slave node correctly, the string containing the function name f and the cell array f_args have to be available to the slave node as these are used in the slave's command (cmd). So these must also be sent.

The total size of the message to be sent to the slave node is determined by finding the space required to send each of the elements individually and summing their size. A blank message buffer is then created and each of the elements is packed into the message starting with f_args and f and finishing with cmd. The whole message msg is then sent to node $slave$ across communication world NEWORLD with message tag TAG.

On the slave side, the message is automatically received and unpacked. The variables f and f_args are now available on the slave. The string cmd now gets evaluated and the commands get executed as described above.

10 Special Thanks

Many thanks to Paul Wogan, John Whelan, Brendan Byrne and Aubrey Dunne for helpful discussions and help in hardware assembly. Many thanks to Seán Begley for developing an MPI-based Octave demonstration on the S129 cluster.

11 Bibliography

[1]. ParallelKnoppix Tutorial and Downloads are on:
$http://pareto.uab.es/mcreel/ParallelKnoppix/Tutorial/Tutorial.html$

[2]. 32-bit compatible VMware Server 1.0 download site:
$http://www.vmware.com/download/server/$

[3]. 64-bit compatible VMware Server Beta 2.0 download site:
$http://www.vmware.com/beta/server/$

[4]. VMware Server 1.0 User Guide:
$http://www.vmware.com/support/pubs$

[5]. Setup for Microsoft Cluster Service. ESX Server 3.0.1 and VirtualCenter 2.0.1. Online on: $http://www.vmware.com/pdf/vi3\_301\_201\_mscs.pdf$

[6]. Dell Precision 490 workstation User's Guide:
$http://support.dell.com/support/edocs/systems/ws490/en/ug/advfeat.htm$

[7]. MPI Tutorial: *http : //www.stanford.edu/ alfw/Talks/Parallel/mpi.pdf*
official LAM/MPI web site: *www.lam − mpi.org*

[8]. ”User-Friendly Parallel Computations with Econometric Examples”, M. Creel, Computational Economics (2005) 26: 107-128. DOI: 10.1007/s10614-005-6868-2, Springer 2005. Online on: *http : //pareto.uab.es/wp/*2005/63705*.pdf*

[9] ”I Ran Four Million Probits Last Night: HPC Clustering With ParallelKnoppix”, M. Creel, Journal of Applied Econometrics 22:215-223 (2007). Published online in Wiley InterScince (www.interscience.wiley.com) DOI:10.1002/jae.945

[10]. Wikipedia: online encyclopedia on *www.wikipedia.org*

[11]. MPI Toolbox for Octave. Tutorial on *http://atc.ugr.es/javier-bin/mpitb*