Crystal06 (v.1.0.2) And MPICH-1.2.7p1 In Ubuntu Desktop 8.10 (and 9.04, 64- and 32-bit) Using The Intel Fortran Compiler

Damian G. Allis, Ph.D., damian@somewhereville.com,
www.somewhereville.com, aim/skype/jabber: somewhereville

Introduction

According the Crystal06 manual:

The CRYSTAL package performs ab initio calculations of the ground state energy, energy gradient, electronic wave function and properties of periodic systems. Hartree-Fock or Kohn-Sham Hamiltonians (that adopt an Exchange- Correlation potential following the postulates of Density-Functional theory) can be used. Systems periodic in 0 (molecules, 0D), 1 (polymers, 1D), 2 (slabs, 2D), and 3 dimensions (crystals, 3D) are treated on an equal footing. In each case the fundamental approximation made is the expansion of the single particle wave functions ('Crystalline Orbital', CO) as a linear combination of Bloch functions (BF) defined in terms of local functions (hereafter indicated as 'Atomic Orbitals', AOs).

To the experimental interpretation-based quantum chemist, Crystal06 is an atom-centered basis set density functional theory (DFT) program that employs standard Gaussian-type basis sets and generalized-gradient approximation and hybrid density functionals for the property prediction of molecular and solid-state systems. This makes the program similar in scope to the non-periodic atomic basis set-based quantum chemistry packages Gaussian03 (the solid-state framework is there but still in-process) and GAMESS-US (both of which I assume people performing calculations are familiar with) and the atomic basis set-based (but not Gaussian-type) solid-state DFT program DMol3.

This document describes in detail how to set up an MPI-based cluster, including the compiling of the MPICH code upon which (one of) the parallel version(s) of Crystal06 was built, the installation of the Intel Fortran Compiler (IFC) required for compiling the MPICH (version 1.2.7p1) program, and the setup of the hardware for a Network File System/SSH-based cluster.

This document is written for the non-technical Linux user and contains many additional details and, to the experienced user, obvious steps and clarifications. It is the author's experience that the typical academic researcher knows far less about Linux, networking, and compiling than the developers of most quantum codes expect of those people downloading (or, in this case, buying) their software. Accordingly, this tutorial is a walk-through that, if there are no significant hardware issues, will account for every minor detail in the process taking you from booting the host machine for the first installation all the way to running an MPI-based Crystal06 calculation.

Those who have ever waited for single-molecule calculations to run to completion can appreciate the complication of now waiting for a calculation of an entire unit cell composed of multiple molecules to run to completion. Such calculations are (currently) only practical when performed in a multi-processor (SMP, Beowulf, etc.) environment, the case for which the documentation for Crystal06 parallel execution is useful only if you already know what you're doing.

It is hoped that this document makes more practical to the less technical general user what the Crystal06 code makes possible.

About Version 1.1

19 May 2009 - Two significant changes. First, this tutorial is moved to an actual page instead of a Wordpress post so that the change log can be better maintained. Second, having now had need to try, I discovered that users of the 32-bit version of Ubuntu will receive errors when they try to install the many 32-bit libraries that are required for the Intel Fortran Compiler to install in the 64-bit Ubuntu version. The proper install list is provided in Steps 3 (on the HOST machine) and 14 (on the GUEST machines).

About Version 1.0

This tutorial contains considerable description overkill in program installation and a procedure that just works but, perhaps, does not work as well as it will with another iteration. For instance, no attempt is made to speed up NFS or tweak compiler settings in the MPICH build. The Crystal06 code is provided pre-compiled, so there is nothing one can do to it that doesn't require requests to the developers.

The important Ubuntu-specific step in this document is performed with the package installation program apt-get. This apt-get package installation step installs (1) everything the Intel Fortran Compiler requires, (2) SSH (for use with MPICH), and (3) all of the Network File System (NFS) software. This apt-get installation does not remove ALL warnings from the IFC installation script (as discussed in Step 5), but the warnings you'll see upon installation are unimportant (unless you want Java installed for some other project). If you continue in your Fortran endeavors (beyond MPICH compilation, that is), you may have to install additional programs via apt-get. For the setup described here, no need.

Version 1 of this document will become **Version 2** (or more) as I find ways to optimize the performance of the system or if people send me information relevant to the system setup (and please do).

It is worth noting that very little of this overall procedure is Crystal06-specific. This could just as soon be an MPICH/IFC/Ubuntu instruction set for any MPI-based program installation, with the procedure generating a cluster ready for many Quantum Chemistry and Molecular Dynamics Codes (the procedure has worked just fine for Abinit, GROMACS, and AMBER, one just has to be aware of setting IFC-specific flags in the ./configure process of each). If people find this procedure useful and wish to add their own builds of other programs (apt-get lists, compiler flags, etc.) it would be most beneficial to have a big document full of useful information for general computational chemistry (and not Linux-experienced) audiences.

Demonstration System Setup

To begin, I am assuming that you will be starting with a FRESH installation, meaning either a hard drive is being completely wiped clean of its previous operating system or a new hard drive is being installed and GRUB/LILO is being used to make your machine a multiple-boot machine. It is (currently) beyond the scope of the document to deal with compatibility, proprietary drivers, competing software, and all other things not related to installing MPICH, the IFC, and Crystal06. The procedure (post-Ubuntu install and update) may work just fine anyway, but I've not tried this procedure with other operating environments.

Computers

Three Dell Precision WorkStation T5400 8-core (2.8 GHz) boxes with 16 GB RAM each - The cluster will use the hard drive on one of the machines for all HD-related activities, so make sure this disk is as large as you need for temporary files (250 GB seems to be far beyond sufficient for Crystal06 calculations). Of course, you can use virtually any machine that will install Ubuntu (you do, however, want the machines to be similar in capabilities for best results).

Switch

NETGEAR GS116 10/100/1000Mbps ProSafe Gigabit Desktop Switch with Jumbo Frame Support 16 x RJ45 512 Kbytes per port Buffer Memory - Obviously, buy as big a switch as you need for the number of machines you intend to use, but plan on expansion (someday). You do not have to have a gigabit switch for the calculations to work, but you will come to regret not having spent the money in the long run. Jumbo Frame Support is preferred, but this will be part of Version 2 of this document.

Battery Backup

APC Smart-UPS SUA1500 1440VA 980W 8 Outlets UPS - Three 8-core boxes, one power plug for the switch, and one LCD monitor. Plugging anything else into this power backup will cause it to sing the song of overdrawn battery when a 24-CPU calculation is started.

NOTE 1: These machines (the impetus for this document) are being used for the calculation of low-frequency vibrational modes as part of Terahertz simulations. For structure optimizations and normal mode analyses NOT being performed with the INTENS keyword (for the calculation of vibrational mode intensities),

RAM quantity is not a significant issue (above 1 GB per CPU, for instance). When intensity calculations are requested on sufficiently large systems (40 or more atoms in the unit cell), an 8-node box will use all 16 GB of onboard RAM (the use of larger SWAP is a more cost-effective, but absolutely not more time-effective, workaround that I've not worked on further).

NOTE 2: Related to the Jumbo Frame discussion, the proper Broadcom BCM5754 drivers for the integrated ethernet in the T5400 boxes are not installed by Ubuntu and must be installed in separate steps. The generic ethernet drivers being used cap the Maximum Transmission Unit (MTU) at 1500 bytes, whereas it should be possible to boost this number to 9000. It is a straightforward check to determine if a good driver is installed by Ubuntu and if you can boost the MTU to 9000 in the /etc/network/interfaces file (Step 8 and 18). Again, not necessary for the calculations to work, but a potential source of speed-up.

Why Ubuntu?

The author has had great success with it. Provided you start with a new hard drive (or intend on wiping the machine clean as part of the installation), it is the easiest of the well-known Linux distributions to install (that I am aware of). As the Linux distribution that is (openly and advertising that it is) trying to be as user/community-friendly as possible, it is often very easy to get answers to your questions on Ubuntu forums, right from google, etc., which is important if you're new to Linux.

Everything in this document beyond the installation, initial system update, and MPICH/IFC/Crystal06 download employs Terminal and pico, so fixes to any problems you may have will likely be general to all Linux distributions.

There is one major issue with Ubuntu that OpenSuse, Fedora, etc. users may or may not have to deal with. The Ubuntu installation CD installs everything (programs, drivers, and libraries) that the contents of the CD needs. If you find yourself installing additional programs that have Ubuntu packages already configured for them (here, using apt-get), the Ubuntu installation process will install all of the additional programs, drivers, and libraries required without you having to know what those additional packages are (a very nice feature). If you're installing a program (or three) that does not have an Ubuntu package configured for it, you are required to do some searching for those missing programs, drivers, and libraries.

This tutorial employs apt-get (run from Terminal) for program/driver/library package installation. The extensive apt-get list installed after the Ubuntu installation is predominantly for satisfying IFC library requirements, with NFS and SSH installed for MPICH and network support. The IFC-specific list was generated by successive error-and-install steps, with successive re-checks of IFC system dependencies required and additional apt-get operations performed before IFC would install without critical errors. I suspect some non-computational chemists have found this tutorial simply because of the presented list of required IFC-specific apt-get installs.

Why MPICH-1.2.7p1?

Why not MPICH2, OpenMPI, or some other version of MPICH as the Message Passing Interface (MPI) of choice? The specific use of MPICH-1.2.7p1 is because the pre-compiled Crystal06 parallel (Pcrystal) binary used for this document was, according to the Crystal06 website, built with the Intel Fortran Compiler and MPICH-1.2.7p1 and all efforts by the author to use other compiler/MPI combinations failed miserably. I attempted to compile MPICH2 for otherwise identical use (for reasons related to message size errors in very memory-intensive INTENS calculations) but after many, many trials, recompilations, and successful non-Crystal06 MPI test runs, I could never get Pcrystal to run 1 calculation on 16 nodes, only 16 separate serial calculations that would all write to stdout. I've, therefore, come to the conclusion that, despite MPICH2 being MPI-1 compatible, there is something to the way Pcrystal was compiled that requires it to be run with MPICH. If you've compiled OpenMPI or MPICH2 for use in Crystal06 calculations and had it work successfully (and not the Itanium version of Crystal06, which was compiled with MPICH2), please consider providing your configure parameters for addition to this document.

NOTE 1: As an IFC build of MPICH is required for Crystal06 to work, the MPICH version one can install via apt-get is NOT an option. Pcrystal is expecting IFC-built libraries, not GNU-based compiler-built libraries.

NOTE 2: If you're building an SMP-only Crystal06 system (that is, using only the processors available on the motherboard), then a GNU build of MPICH works just fine.

Why Use The Intel Fortran Compiler?

Ignoring the many details of Fortran compilation, libraries, etc., the version of Crystal06 being used here was built with the IFC and the parallel version (Pcrystal) is expecting to see an MPICH compiled with the IFC. That's as good a reason as any.

To the best of my knowledge (and after testing), you cannot build MPICH with GNU Fortran compilers and have MPI-Crystal06 calculations work properly. My choice of the IFC over the Portland Group Compiler is purely out of cost-effectiveness. The IFC is available for free to academics (non-commercial license agreement) and there's a Crystal06 version compiled with it. The Portland Group Compiler, for which Crystal06 builds are also available, is available at an academic discount. If I were not an academic with access to the free Intel version, I would have thought harder about which to purchase/use.

And so it begins...

Step 0 HOST and GUESTs: Things To Be Aware Of In The Overall Procedure

Machine Names And IP Addresses

The cluster documented here will have all machines on a single switch (see picture). For setup purposes, each machine will have an IP address (set in /etc/network/interfaces) and a machine name (set in /etc/hostname). The IP addresses will be the olde reliable 10.1.1.NN, where NN is any number between 1 and 254 (you may see "broadcast" set to 10.1.1.255 in some google searches for /etc/network/interfaces). We will be defining IP addresses in the /etc/network/interfaces steps on HOST and GUESTs below. I'm explaining the 10.1.1.NN IP scheme here to not explain it later. I will be naming the machines crystalhost for the HOST machine and crystalguestN for the GUEST machines, with N being 1 to whatever (I assume no more than 254, of course). These names will be assigned/aliased in the hostnames file (performed in a procedure step. If you're coming from a Windows and OSX file/directory naming mindset, make sure you avoid spaces and crazy characters in the file/folder names).

NFS-Mounted Directory Name

This is a Crystal06 installation, so I will be using the name /crystal06 for the folder being mounted on all of the GUEST machines via NFS from the HOST machine. Obviously, name this folder whatever you want (avoiding spaces and crazy characters in the name) and propagate the name changes in this document accordingly.

pico

pico is the text editor being used to modify various text files in this document. It is installed as part of the Ubuntu CD installation, so it should be on your machine. I'm using pico because there's no need to do all of the file modifications via the GUI and I did not grow up with vi. When file changes are performed in pico, you can save and exit by Crtl-X and hitting "Enter" twice.

Terminal

To a windows user, a Terminal window is one that will not close. To a Linux and OSX user, Terminal windows are your ASCII interfaces to the contents of your hard drive. You should become familiar with moving around the Terminal window if you intend on doing any significant computational chemistry, as you will constantly be moving files, deleting files, renaming files, compressing files, and starting (and canceling) calculations.

64-bit And 32-bit

I have introduced a mild complication to this installation procedure by installing the 64-bit version of Ubuntu Desktop on the cluster machines. The complication is due to Crystal06 being compiled 32-bit and 32-bit programs will not run in a 64-bit OS unless several 32-bit libraries are also installed. We will install these files in the apt-get steps for both HOST and GUESTs below (and these libraries NEED to be on all machines for Crystal06 to work). If you installed/want to install the 32-bit version of Ubuntu Desktop, you can still use the same apt-get package lists to install all of the other missing programs (you will simply get messages saying "library already installed."). There's no point in pruning this list down for 32- and 64-bit users. You will, however, need to make sure to install the 32-bit IFC (not the 64-bit I'll be installing here) if you install the 32-bit Ubuntu Desktop version.

sudo

The one aspect of Ubuntu that differs from most other Linux distributions is the differentiation between root, Administrator, and user right from the installation. Whereas you set up the root user in Suse and Fedora as part of the installation process (and set up the users separately), you set up an Administrator account during Ubuntu installation that is distinct from root (with many, but not all, root privileges). As a result, if you do not set up the root account to perform installations and system-level modifications, you are left in the Administrator account to use the sudo (super-user do…) command to allow you, the Administrator, to build and install programs outside of your home ($HOME) directory.

"Do I have to constantly sudo everything?" No. Accessing a pure "root" terminal for installations is straightforward after the root password is assigned, it is simply argued by many (including the Ubuntu wiki) that it is safer to use sudo. If you want to go the root route, check out help.ubuntu.com/community/RootSudo. You do not need to use sudo to make changes to files in your user directory, only to make modifications to system files (which we will be doing in this document).

A@B:

When staring at a Terminal window, the prompt will look like A@B:, with A being the username and B being the machine name. I use this A@B convention in the document and you do NOT type these as they show up (type only the text that follows as marked up in this document).

[TAB]

For the absolute Linux newbie, you can complete long file names automatically by hitting the TAB key. Of course, this only works if you've only one of the file in your directory (for l_cprof_p_11.0.083_intel64.tgz in a directory containing other files that start with "l", you could simply type the l_ and [TAB] to complete the whole file. Works for directories, too).

Step 1 HOST: Ubuntu Installation

Steps in this document are based on my installing the 64-bit Desktop version of Ubuntu. I mention this because the Crystal06 version I'll be using for calculations was compiled 32-bit. The program still runs just fine, but ONLY IF 32-bit libraries are installed after the CD installation. These libraries must reside on both the HOST and GUEST machines. They are also required/strongly recommended for the IFC installation on the HOST machine. The setup below will only install the IFC on the HOST machine for the MPICH build (and we'll use the NFS mount and PATH specification to make MPICH accessible to all machines). When you go to run Pcrystal from/on a GUEST node (there are some webpages recommending that you never run an MPI executable from the HOST machine, instead logging into a GUEST machine to perform calculations. I've found that it makes no difference) and do not have the 32-bit libraries installed, you'll get the common "compiled 32-bit but running on a 64-bit OS" error:

"Pcrystal: No such file or directory."

Ubuntu automates practically all of the installation. For simplicity, set up the same administrative/default username and password on all machines. Do not bother performing a custom partition and making a "crystal06" partition (the place where these runs will be performed on the host machine). We'll be making a "work" directory after installation, so just run with the default settings.

INSTALL NOTE: I have had random trouble with onboard video drivers on a number of Dell models during installation. I find the quality of the graphical interface to be far less important than the speed of the computation, so consider specifying "Safe Graphics Mode" (**F4** at the install screen after you specify your language) to avoid resolution issues during and after installation.

Step 2 HOST: Post-Installation System Update

Unless you plan on transferring Ubuntu packages, the Intel Fortran compiler, the MPICH source, and the Crystal06 binaries by flashdrive or CD/DVD, a working network interface is required for this step (and, of course, you will have to have a working ethernet port for the cluster anyway, so I assume that your network card works). You probably do not need to update the installation, but you need the network connection working for apt-get anyway, so you might as well do a full system update. If you installed 8.10, I wouldn't bother upgrading to 9.04 at this time (just as an aside, Ubuntu 9.04 does work with the procedure).

If you're installing at a campus, you (1) may have to register your machine in the BOOTP table (see your computing services website for more information), (2) "force" an IP address from those available on your subnet (your computing people will hate you for it… if they catch you doing it), (3) may have a router and can simply "plug in" to get online. After the update, apt-get, and software downloads, you will not need to be "online" again (we'll be hard-coding IP addresses for this switch-friendly cluster). In theory, you could do all of this without ever being online (provided you have a machine online for downloading purposes).

To perform a System Update from the Desktop, go to System -> Administration –> System Update. This may/will take some time and a restart will be required (in theory, you can continue with the steps below and reboot after all changes are made). I recommend performing the apt-get procedure in the next step and THEN rebooting (so that all web-related installations are complete).

Step 3 HOST: apt-get Installations Required For SSH, NFS, and IFC

I assume the network still works from Step 2. Open a Terminal Window (Applications -> Accessories -> Terminal) and type (or copy + paste) the following in a 64-bit Ubuntu install:

```
A@B:~$ sudo apt-get install ia32-libs lib32asound2 lib32ncurses5 lib32nss-mdns lib32z1 lib32gfortran3 gcc-4.3-multilib gcc-multilib lib32gomp1 libc6-dev-i386 lib32mudflap0 lib32gcc1 lib32gcc1-dbg lib32stdc++6 lib32stdc++6-4.3-dbg libc6-i386 libstdc++5 g++ g++-4.3 libstdc++6-4.3-dev g++-multilib g++-4.3-multilib gcc-4.3-doc libstdc++6-4.3-dbg libstdc++6-4.3-doc nfs-common nfs-kernel-server portmap ssh
```

Note the many 32-bit libraries. These are required by both IFC and Pcrystal (making the 64-bit installation I used for this document back-compatible (in a way) with the 32-bit Pcrystal build). If you're working with a 32-bit Ubuntu installation, you would use the following program list:

```
A@B:~$ sudo apt-get install gcc-4.3-multilib gcc-multilib libstdc++5 g++ g++-4.3 libstdc++6-4.3-dev g++-multilib g++-4.3-multilib gcc-4.3-doc libstdc++6-4.3-dbg libstdc++6-4.3-doc nfs-common nfs-kernel-server portmap ssh
```

Step 4 HOST: Software Downloads

We will be setting up the cluster in such a way that all of the programs below will only need to be installed on the HOST machine, with NFS used to mount a HOST machine directory on all of the GUEST machines (who will then see MPICH and PCrystal when we set the GUEST PATHs). If you're on the Desktop of a fresh install, the Firefox icon should be obvious in the upper left (or, if your PDF viewer shows them, simply click on the links in this document to open the pages). For organizational purposes, simply download all of these files to the Desktop (if you're using the Server Edition, I assume you know your way around your $HOME directory and know where to make changes below) and I will assume these files reside on the Desktop in all commands typed in following steps.

1. MPICH-1.2.7p1 - http://www.mcs.anl.gov/research/projects/mpi/mpich1/download.html

2. Intel Fortran Compiler - http://software.intel.com/en-us/articles/non-commercial-software-development/

This will take you to the Agreement Page. After you hit "accept," look for (under "Compilers"):

Intel® Fortran Compiler Professional Edition for Linux

Fill out the Noncommercial Product Request information on the next screen, then simply follow the instructions in the email (you will need the license number in the email during the installation). Currently, the file to download is l_cprof_p_11.0.083_intel64.tar.gz (version number will change). If you're using a 32-bit Ubuntu Desktop installation disk, download the 32-bit version.

3. Crystal06 - http://www.crystal.unito.it/

You will need a username/password from the Crystal developers to download the Crystal binaries. I assume you've already taken care of this step. For this document, we'll be using the version of Pcrystal found in crystal06_v1_0_2_Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6.tar.gz.

Step 5 HOST: Installing The Intel Fortran Compiler

The installation of all of the libraries and programs in the apt-get step above makes the Intel installation much easier (and, for that matter, possible). As this is a 64-bit OS I've installed, well be installing the Intel-64 version of the Fortran compiler (currently l_cprof_p_11.0.083_intel64, the one I assume is now residing in .tar.gz format on your Desktop).

At a Terminal Window:

A@B:~$ cd ~/Desktop
A@B:~$ gunzip l_cprof_p_11.0.083_intel64.tgz
A@B:~$ tar xvf l_cprof_p_11.0.083_intel64.tar

[you should now see the folder "l_cprof_p_11.0.083_intel64" on your Desktop]

A@B:~$ cd l_cprof_p_11.0.083_intel64/
A@B:~/l_cprof_p_11.0.083_intel64$ sudo ./install.sh

You will now pass through a series of installation screens. The only thing you will need to have available is the serial number provided in the email sent to you (ABCD-12345678).

If you've not installed all of the proper libraries, you'll see the following screen at Step 4.

```
Step no: 4 of 7 | Installation configuration - Missing Critical Pre-requisite
--------------------------------------------------------------------------------
There is one or more critical unresolved issue which prevents installation to
continue. You can fix it without exiting from the installation and re-check. Or
you can quit from the installation, fix it and run the installation again.
--------------------------------------------------------------------------------
Missing critical pre-requisite
-- missing system commands
-- missing system commands
-- 32-bit libraries not found
--------------------------------------------------------------------------------
1. Show the detailed info about issue(s) [default]
2. Re-check the pre-requisites

h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

With a proper apt-get installation, you should see the following at Step 4.

```
Step no: 4 of 7 | Installation configuration - Missing Optional Pre-requisite
--------------------------------------------------------------------------------
There is one or more optional unresolved issues. It is highly recommended to fix
it all before you continue the installation. You can fix it without exiting from
the installation and re-check. Or you can quit from the installation, fix it and
run the installation again.
--------------------------------------------------------------------------------
Missing optional pre-requisite
-- No compatible Java* Runtime Environment (JRE) found
-- operating system type is not supported.
-- system glibc or kernel version not supported or not detectable
-- binutils version not supported or not detectable
--------------------------------------------------------------------------------
1. Skip missing optional pre-requisites [default]
2. Show the detailed info about issue(s)
3. Re-check the pre-requisites

h. Help
b. Back to the previous menu
q. Quit
--------------------------------------------------------------------------------
Please type a selection or press "Enter" to accept default choice [1]:
```

Note the word "optional." As for additional installations to remove these errors, Java is installed easily enough via apt-get (although we will not be using it for Crystal06, so it is not necessary). If you check the "operating system type" error, you will see that the IFC installer currently recognizes the following OS's:

```
Asianux* 3.0
Debian* 4.0
Fedora* 9
Red Hat Enterprise Linux* 3, 4, 5
SGI ProPack* 5 (IA-64 and Intel(R) 64 only)
SuSE Linux* Enterprise Server* 9, 10
Turbo Linux* 11
Ubuntu* 8.04 (IA-32 and Intel(R) 64 only)
```

I assume the developers wrote in simple system checks for directory structure or the like and it will be updated as time goes on. The warning is completely irrelevant for our purposes. I suspect the installation is looking for Kernel 2.4.XX (this version of Ubuntu will install 2.6), but that's also not a problem. The binutils are installed with the original Ubuntu install, so I assume this is just another version conflict (and not important).

The IFC will be installed in /opt/intel/Compiler/11.0/083, which we will add to the PATH on the HOST machine in a later step for MPICH compilation.

Step 6 HOST: Make The Crystal06 Directory

This directory will be THE directory that all of the cluster work reads to and writes from. I've now had two people mention that NFS is probably not the fastest and most efficient way to go, but it is the most common way for MPI systems to be set up according to my google research. Hopefully, this means you'll find websites addressing specific problems with your installation if you're attempting this installation with other Linux distributions (or if you have a problem with this procedure).

We'll be placing the work directory right at the base of the directory tree (/). Accordingly, you'll need Administrative access to do this (hence the sudo). The three steps below make the directory, change the group permissions for the directory, and lets all users read + write + execute content in this directory.

```
A@B:~$ sudo mkdir /crystal06
A@B:~$ sudo chgrp –R users /crystal06
```

A@B:~$ sudo chmod a+wrx /crystal06

Step 7 HOST: PATH Statement

The PATH specification adds the IFC directory, the work directory (/crystal06), and the MPICH executable directory (/crystal06/mpich-1.2.7/bin) to the user path (where Ubuntu looks when you type the name of an executable into the Terminal) on the HOST machine. Note that one of these directories does not yet exist and the /crystal06 directory is currently empty. This does not affect the specification of the PATH statement (and this directory will be occupied soon enough).

A@B:~$ cd $HOME
A@B:~$ pico .profile

[At the bottom of the .profile file, add the following:]

PATH="/opt/intel/Compiler/11.0/083/bin/intel64:/crystal06/mpich-1.2.7/bin:/crystal06:$PATH"

[Ctrl-X to exit (and Enter twice to save)]

A@B:~$ source .profile

Step 8 HOST: Network Interface Setup And Machine Name

This step assigns the IP address to the HOST machine, names the HOST machine crystalhost, and adds the list of all machine names in the cluster. We begin with the IP address:

A@B:~$ sudo pico /etc/network/interfaces

[Ignoring explanations, make your interfaces file look like the following]

/etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.1.1.1
netmask 255.255.255.0
broadcast 10.1.1.255
network 10.1.1.0

[Ctrl-X to exit (and Enter twice to save)]

Now, we change the name of this HOST machine by modifying /etc/hostname

A@B:~$ sudo pico /etc/hostname

[Change the name to crystalhost by simply replacing whatever is there with the following]

crystalhost

[Ctrl-X to exit (and Enter twice to save)]

Finally, we add the IP addresses and machine names for the rest of the machines in the cluster.

A@B:~$ sudo pico /etc/hosts

[Make your hosts file look like this (adding lines for each GUEST machine)]

```
127.0.0.1    localhost
127.0.1.1    crystalhost
10.1.1.1     crystalhost
10.1.1.2     crystalguest1
10.1.1.3     crystalguest2
...
10.1.1.N     crystalguestN
```

[This may or may not be at the bottom of the hosts file and does not require modification]

```
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

[Ctrl-X to exit (and Enter twice to save)]

We will propagate all of these changes by a reboot after all of the HOST machine setup is complete.

Step 9 HOST: Network File System Setup

The NFS installation will open the HOST machine to directory mounting by all of the GUEST machines. We will be making the /crystal06 directory permanently accessible to other machines (even after reboot, crashes, etc.) by adding this directory and a few important NFS-specific flags to /etc/exports. The apt-get installation of the NFS server did the vast majority of the dirty work.

First, we configure portmap. When you run the dpkg-reconfigure command, you'll be taken to a color configure screen. When it asks you if 127.0.0.1 should be assigned to LOOPBACK, say NO.

```
A@B:~$ sudo dpkg-reconfigure portmap
```

[Select NO when asked (navigate with the TAB key)]

```
A@B:~$ sudo /etc/init.d/portmap restart
A@B:~$ sudo pico /etc/exports
```

[This file should look something like the following]

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync) hostname2(ro,sync)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt)
# /srv/nfs4/homes  gss/krb5i(rw,sync)
#
```

[add the following line to the bottom of this file]

```
/crystal06 *(rw,sync,no_subtree_check)
```

[Ctrl-X to exit (and Enter twice to save)]

```
A@B:~$ sudo /etc/init.d/nfs-kernel-server restart
```

A@B:~$ sudo exportfs -a

Step 10 HOST: Building MPICH-1.2.7p1

This should be the most complicated-looking step in this document, but should run smoothly with everything above accounted for.

A@B:~$ cd ~/Desktop
A@B:~$ gunzip mpich-1.2.7p1.tar.gz
A@B:~$ tar xvf mpich-1.2.7p1.tar

[You will have the directory mpich-1.2.7p1 on your Desktop]

A@B:~$ cd mpich-1.2.7p1/
A@B:~/mpich-1.2.7p1$ ./configure --with-device=ch_p4 --with-arch=LINUX --prefix=/crystal06/mpich-1.2.7 --with-romio=--with-file-system=nfs --enable-f90modules -fc=ifort -f90=ifort

[Considerable output should appear on the screen. Note that the --prefix flag will install mpich-1.2.7 into /crystal06. The –fc and –f90 flags direct the ./configure script to use the IFC]

A@B:~/mpich-1.2.7p1$ make

[More considerable output should appear on the screen]

A@B:~/mpich-1.2.7p1$ sudo make install

[This step will generate /crystal06/mpich-1.2.7]

The output of the make install step should be as follows:

```
if [ "/crystal06/mpich-1.2.7" = "/crystal06/mpich-1.2.7" ] ; then \
            ./bin/mpiinstall  ; \
        else \
            ./bin/mpiinstall -prefix=/crystal06/mpich-1.2.7  ; \
        fi
Installing documentation ...
Done installing documentation
Installing manuals
Done installing manuals
Installing MPE
Copying MPE include files to /crystal06/mpich-1.2.7/include
Copying MPE libraries to /crystal06/mpich-1.2.7/lib
Copying MPE utility programs to /crystal06/mpich-1.2.7/bin
About to run installation test for C programs...

** Testing if C application can be linked with logging library
/crystal06/mpich-1.2.7/bin/mpicc  -DMPI_LINUX  -DUSE_STDARG  -DHAVE_PROTOTYPES     -I/crystal06/
mpich-1.2.7/include   -c cpi.c
/crystal06/mpich-1.2.7/bin/mpicc  -DMPI_LINUX -DUSE_STDARG -DHAVE_PROTOTYPES  -o cpi_log cpi.o -
L/crystal06/mpich-1.2.7/lib -llmpe -lmpe   -lm
** C application can be linked with logging library

** Testing if C application can be linked with tracing library
/crystal06/mpich-1.2.7/bin/mpicc  -DMPI_LINUX -DUSE_STDARG -DHAVE_PROTOTYPES  -o cpi_trace cpi.o
-L/crystal06/mpich-1.2.7/lib -ltmpe   -lm
** C application can be linked with tracing library

** Testing if C application can use both automatic and manual logging together
/crystal06/mpich-1.2.7/bin/mpicc  -DMPI_LINUX  -DUSE_STDARG  -DHAVE_PROTOTYPES     -I/crystal06/
mpich-1.2.7/include   -c cpilog.c
```

```
/crystal06/mpich-1.2.7/bin/mpicc  -DMPI_LINUX -DUSE_STDARG -DHAVE_PROTOTYPES   -o cpilog cpilog.o
-L/crystal06/mpich-1.2.7/lib -llmpe -lmpe   -lm
** C application can use both automatic and manual logging together

About to run installation test for Fortran programs...

** Testing if Fortran77 application can be linked with logging library
/crystal06/mpich-1.2.7/bin/mpif77  -I/crystal06/mpich-1.2.7/include  -c fpi.f
eval: 1: ifort: not found
make[2]: *** [fpi.o] Error 127
** Fortran77 application CANNOT be linked with logging library

Copying SLOG2SDK's lib
Copying SLOG2SDK's doc
Copying SLOG2SDK's logfiles
Creating SLOG2SDK's bin
Installed SLOG2SDK in /crystal06/mpich-1.2.7
/crystal06/mpich-1.2.7/sbin/mpiuninstall may be used to remove the installation
creating Makefile
About to run installation test...
/crystal06/mpich-1.2.7/bin/mpicc  -c cpi.c
/crystal06/mpich-1.2.7/bin/mpicc -o cpi cpi.o -lm
/crystal06/mpich-1.2.7/bin/mpicc  -c simpleio.c
/crystal06/mpich-1.2.7/bin/mpicc -o simpleio simpleio.o
/crystal06/mpich-1.2.7/bin/mpif77  -c pi3.f
eval: 1: ifort: not found
make[2]: *** [pi3.o] Error 127
/crystal06/mpich-1.2.7/bin/mpif77  -c pi3p.f
eval: 1: ifort: not found
make[2]: *** [pi3p.o] Error 127
make[1]: *** [all] Error 2
rm -f *.o *~ PI* cpi  pi3 simpleio hello++ pi3f90 cpilog
rm -rf SunWS_cache ii_files pi3f90.f pi3p cpip *.ti *.ii
installed MPICH in /crystal06/mpich-1.2.7
/crystal06/mpich-1.2.7/sbin/mpiuninstall may be used to remove the installation.
```

You will note several warnings when you do this that are annoying but not important. The reason for these errors is because the IFC directory (ifort) is in your user PATH, but not in the root PATH (so, when you sudo, Ubuntu doesn't know where ifort is). You can remedy this by repeating Step 7 above but doing so in the / root directory.

A@B:~$ cd /root
A@B:~$ sudo pico .profile

[At the bottom of the .profile file, add the following:]

PATH="/opt/intel/Compiler/11.0/083/bin/intel64:/crystal06/mpich-1.2.7/bin:/crystal06:$PATH"

[Ctrl-X to exit (and Enter twice to save)]

A@B:~$ sudo source .profile

You can check that the PATH and the MPICH compilation are correct by typing "mpirun" at the prompt. You should receive the message:

Missing: program name

This is an mpirun-can't-find-the-program-you-want-to-run error and means that the PATH is right and mpirun at least starts properly.

Step 11 HOST: machines.LINUX

This file provides the list of machines that MPICH has accessible to it when performing a calculation (effectively, an address book of machines to call upon as processors are requested). We will refer to these machines by the names we used in the hostname file. This file will reside in /crystal06, the same place as the Pcrystal executable. When running the calculation with mpirun, you call this file with the -machinefile flag.

Now, a potential problem. The format for the machines.LINUX file is recommended to be:

machinename:number of CPUs on the machine

So, for the 8-core HOST machine and the two 8-core GUEST machines, the lines would be:

crystalhost:8
crystalguest1:8
crystalguest2:8

When I do it this way, I have the strange issue of having more than 8 Pcrystal processes running on a single machine on occasion, as if MPICH doesn't know that it is supposed to run one process per processor. This does not occur continuously, but having 9 or 10 processes running on a single box is not only bad for speed, but also bad for overworking processors. My solution to this problem is to simply have one-line-per-processor in the machines.LINUX file so that each processor exists by machine name in the file. MPICH seems to hold to never sending more than one process per processor this way. I've not diagnosed this problem further and likely will not without someone else instigating another check.

To generate the machines.LINUX file...

A@B:~$ pico /crystal06/machines.LINUX

[In this new file, add the following (these are for 8-core boxes. Obviously, add one hostname for each processor on the box). You can mix-and-match machines if you like (single-core, dual, quad, etc.), just make sure you've one hostname line per processor.]

crystalhost
.. 6 more ..
crystalhost
crystalguest1
.. 6 more ..
crystalguest1
crystalguest2
.. 6 more ..
crystalguest2
...
crystalguestNN
.. 6 more ..
crystalguestNN

Step 12 HOST: Crystal06 Selection And Setup

One final setup step before we move to the GUEST machines and, when the cluster is all together, set up SSH. The Crystal06 download in question is Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6.tar.gz, which you'll have downloaded from the Crystal06 website (and, I assume, saved to the Desktop).

A@B:~$ cd ~/Desktop
A@B:~$ gunzip crystal06_v1_0_2_Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6.tar.gz
A@B:~$ tar xvf crystal06_v1_0_2_Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6.tar

[You should now have the directory [bin] on your Desktop]

A@B:~$ cd bin/crystal06_v1_0_2_Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6/v1_0_2/
A@B:~/bin/Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6/v1_0_2$ ls

[You should see crystal, Pcrystal, and properties. We will copy these to the crystal06/ folder]

A@B:~/bin/Linux-ifort_pentium4_mpich_1.2.7.fedora5_2.6/v1_0_2$ cp * /crystal06/

That's the last step for the HOST machine until the GUEST machines are ready. Now, reboot the HOST machine, plug the ethernet cable into the switch, and bring on the GUEST machines. You will follow the series below for ALL machines, with the only difference at each machine being the IP address and hostname specification (I'll make sure to point out).

Step 13 GUEST: Installation And System Update

See Step 1 HOST and Step 2 HOST. Exactly the same.

Step 14 GUEST: apt-get Installations Required For SSH, NFS, And 32-bit Pcrystal

Open a Terminal Window (Applications -> Accessories -> Terminal) and type (or copy + paste) the following:

A@B:~$ sudo apt-get install ia32-libs lib32asound2 lib32ncurses5 lib32nss-mdns lib32z1 lib32gfortran3 gcc-4.3-multilib gcc-multilib lib32gomp1 libc6-dev-i386 lib32mudflap0 lib32gcc1 lib32gcc1-dbg lib32stdc++6 lib32stdc++6-4.3-dbg libc6-i386 libstdc++5 g++ g++-4.3 libstdc++6-4.3-dev g++-multilib g++-4.3-multilib gcc-4.3-doc libstdc++6-4.3-dbg libstdc++6-4.3-doc nfs-common portmap ssh

[REPEAT: Note the many 32-bit libraries. These are required by both IFC and Pcrystal (making the 64-bit installation I used for this document back-compatible (in a way) with the 32-bit Pcrystal build).] If you're working with a 32-bit Ubuntu installation, you would use the following program list:

A@B:~$ sudo apt-get install gcc-4.3-multilib gcc-multilib libstdc++5 g++ g++-4.3 libstdc++6-4.3-dev g++-multilib g++-4.3-multilib gcc-4.3-doc libstdc++6-4.3-dbg libstdc++6-4.3-doc nfs-common portmap ssh

The only difference between the GUEST list and the HOST list is the absence of nfs-server-kernel in the GUEST list.

Step 15 GUEST: Make The Crystal06 Directory

We'll be placing the work directory right at the base of the directory tree (/) for each GUEST machine. Accordingly, you'll need Administrative access to do this (hence the sudo). The three steps below make the directory, change the group permissions for the directory, and lets all users read + write + execute content in this directory.

A@B:~$ sudo mkdir /crystal06
A@B:~$ sudo chgrp –R users /crystal06
A@B:~$ sudo chmod a+wrx /crystal06

We will be mounting the HOST machine /crystal06 folder in each GUEST machine/crystal06 folder.

Step 16 GUEST: PATH Statement

The PATH specification adds the work directory (/crystal06) and the MPICH executable directory (/crystal06/mpich-1.2.7/bin). These folders are now populated on the HOST machine (these directories do not need to be accessible for you to specify the PATH statement).

A@B:~$ cd $HOME
A@B:~$ pico .profile

[At the bottom of the .profile file, add the following:]

```
PATH="/crystal06/mpich-1.2.7/bin:/crystal06:$PATH"
```

[Ctrl-X to exit (and Enter twice to save)]

```
A@B:~$ source .profile
```

Step 17 GUEST: Network Interface Setup And Machine Name

This step assigns the IP address to each GUEST machine, names the GUEST machine crystalguestN, and adds the list of all machine names in the cluster. We begin with the IP address:

```
A@B:~$ sudo pico /etc/network/interfaces
```

[Ignoring explanations, make your interfaces file look like the following]

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.1.1.N
netmask 255.255.255.0
broadcast 10.1.1.255
network 10.1.1.0
```

[Ctrl-X to exit (and Enter twice to save)]

Now, we change the name of this GUEST machine by modifying /etc/hostname

```
A@B:~$ sudo pico /etc/hostname
```

[Change the name to crystalguestN by simply replacing whatever is there with the following]

```
crystalguestN
```

[Ctrl-X to exit (and Enter twice to save)]

Finally, we add the IP addresses and machine names for the rest of the machines in the cluster.

```
A@B:~$ sudo pico /etc/hosts
```

[saving the explanations for other websites, make your hosts file look like this (adding lines for each machine). And there is no problem with having the machine you're on in this list. Just make sure to keep the N values straight for 127.0.1.1.]

```
127.0.0.1    localhost
127.0.1.1    crystalguestN
10.1.1.1     crystalhost
10.1.1.2     crystalguest1
10.1.1.3     crystalguest2
...
10.1.1.N     crystalguestN
```

[This may or may not be at the bottom of the hosts file and does not require modification]

```
# The following lines are desirable for IPv6 capable hosts
```

```
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

[Ctrl-X to exit (and Enter twice to save)]

We will propagate all of these changes by a reboot after all of the GUEST machine setups are complete.

Step 18 GUEST: NFS Folder Mounting

We do not set up NFS here, as that was done on the HOST machine. With the nfs-common package installed, the underlying code is already ready already for mounting the HOST directory (that we opened to the world in the HOST setup).

There are two ways to do this. The first is the immediate-connect method that we will use first to make sure everything works right. The second method permanently adds the HOST machine /crystal06 folder to the GUEST file system so that this drive will automatically be mounted at reboot (provided the HOST machine is on and plugged into the switch, of course).

A) Immediate-connect

```
A@B:~$ sudo mount crystalhost:/crystal06 /crystal06
A@B:~$ cd /crystal06/
A@B:~$:/crystal06$ ls
```

The ls command should list the contents of the folder, which should be Pcrystal, crystal, utils, and the mpich-1.2.7 directory. If you see these, you're all set!

B) Permanent-connect

This is the final step in the GUEST setup before we reboot the GUEST machines, plug them into the switch, and set up SSH. We will add the HOST /crystal06 folder to /etc/fstab so that this directory is always mounted upon reboot (provided the HOST machine is up, of course).

```
A@B:~$ sudo pico /etc/fstab
```

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
# /dev/sda1 UUID=1be7d1a8-1098-40be-8ac4-f6f1dfa46c32 / ext3 relatime,errors=remount-ro 0 1
# /dev/sda5 UUID=d5750659-faba-4a0b-b659-ab3c7e9cde9f none swap sw 0 0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto,exec,utf8 0 0
```

[Add the following line to the bottom]

```
10.1.1.1:/crystal06 /crystal06 nfs   rw,noac   0   0
```

That completes the basic GUEST setup. To propagate all changes, reboot the machines and plug into the switch. For the remaining steps, we will NOT need to log into the GUEST machines directly, instead performing the final setup steps directory from the HOST machine.

Step 19 From HOST: Testing The Cluster

At this point, I assume that all machines are rebooted and plugged in. To check connectivity, open a Terminal window at type the following:

A@B:~$ ping 10.1.1.**N** (where N is the number of each GUEST machine)

With luck, you'll see something like the following for each machine:

```
PING 10.1.1.N (10.1.1.N): 56 data bytes
64 bytes from 10.1.1.N: icmp_seq=0 ttl=50 time=0.102 ms
64 bytes from 10.1.1.N: icmp_seq=1 ttl=50 time=0.100 ms
```

Step 20 From HOST: Setting Up SSH

The Secure SHell network protocol is familiar to any quantum chemist with NCSA time and is the method of choice for cross-cluster communication by MPICH. What we are going to do below is make the login from HOST to each GUEST password-less by placing a generated HOST authentication key on each GUEST (so the GUEST will see the login attempt from HOST, confirm the HOST is the HOST, and use the stored, encrypted copy of the password on the GUEST machine to allow the HOST to log in).

A@B:~$ ssh-keygen -t dsa

[You will see something like the following. Don't bother entering a passphrase.]

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/terahertz/.ssh/id_dsa):
Created directory '/home/terahertz/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/terahertz/.ssh/id_dsa.
Your public key has been saved in /home/terahertz/.ssh/id_dsa.pub.
The key fingerprint is:
33:8a:1e:14:59:ba:11:b7:7d:75:68:4d:af:09:4e:65 terahertz@terahertz2
The key's randomart image is:
+--[ DSA 1024]----+
|    . o     .+E  |
|     * o   .o+.. |
|    = . . ..o   .|
|     +   . o . o |
|    o   S   . o  |
|   . . . o       |
|    o .          |
|   . .           |
|    .            |
+-----------------+
```

A@B:~$ cd ~/.ssh
A@B:~/.ssh$ mv id_dsa.pub authorized_keys2

[The authorized_keys2 file will be copied onto each GUEST machine]

A@B:~/.ssh$ chmod 644 authorized_keys2

[Sets the permissions on this file]

This series of steps is performed on ALL GUEST machines.]

A@B:~/.ssh$ sftp crystalguest**N**

[As we've not logged into any GUEST machine yet from the HOST, this first login will add the HOST machine to the "list of known hosts" on each GUEST machine. This screen will look like the following:]

```
The authenticity of host 'crystalhost (10.1.1.1)' can't be established.
RSA key fingerprint is 5a:ab:43:5c:7a:82:1c:c2:30:ab:17:1d:7b:dc:35:5d.
Are you sure you want to continue connecting ? yes
```

```
Warning: Permanently added '10.1.1.1' (RSA) to the list of known hosts.
username@10.1.1.1's password:
```

[This is the password for the Administrative account on the GUEST machines, and we did use the same username/password combination for all machines.]

And you should now be logged into the GUEST machine.

```
sftp> cd .ssh
sftp> put authorized_keys2
sftp> bye
```

Now, when you ssh into each guest machine (ssh crystalguest**N**), you should automatically be taken to a prompt instead of immediately being prompted to provide a password. Try it.

And, in theory, we are ready for a Crystal06 calculation.

Step 21: Executing A Parallel Pcrystal Run

Finally, we attempt a calculation to see it everything is working. For the parallel version, your input file should ONLY be named INPUT. While we would normally direct stdout to a textfile (FILENAME.log), for the moment we'll simply run Pcrystal and have the output fly across the screen (and use Crtl-C to cancel the calculation when you see it working properly).

Quick Test: From a Terminal Window on the HOST machine:

```
A@B:~$ cd /crystal06
A@B:~/crystal06$ pico INPUT
```

[copy + paste the contents of an INPUT file. One is provided below. Crtl-X and Enter twice to Exit.]

```
A@B:~/crystal06$ mpirun -machinefile machines.LINUX -np NN /crystal06/Pcrystal
```

[where **NN** in the number of processors you wish to use (here, 24). With luck, you see output scrolling on your screen.]

Long Run: From a Terminal window on the HOST machine:

```
A@B:~$ cd /crystal06
A@B:~/crystal06$ pico INPUT
```

[copy + paste the contents of an INPUT file. One is provided below. Crtl-X and Enter twice to Exit.]

```
A@B:~/crystal06$ mpirun -machinefile machines.LINUX -np NN /crystal06/Pcrystal >& FILENAME.log &
```

[where **NN** in the number of processors you wish to use (here, 24). With luck, you see output scrolling on your screen.]

Step 22: Clean Up Your Work Area

Abbreviations: IFC = Intel Fortran Compiler; NFS = Network File System; SSH = Secure Shell; GUI = Graphical User Interface; GNU = GNU's Not Unix; MPI = Message Passing Interface; MTU = Maximum Transmission Unit; ASCII = American Standard Code for Information Interchange.

Sample INPUT File

This file should be in the same directory as Pcrystal and must be named **INPUT**. Simply copy + paste this text into **pico**.

```
Test - HMX B3LYP/6-31G(d,p) Optimization
CRYSTAL
0 0 0
14
6.54 11.05 8.7 124.3
14
6     -1.916099596340E-01  6.676349517289E-02 -2.182885523722E-01
6     -2.503804839553E-01 -1.120680815923E-01 -5.474423890830E-02
1     -8.473363602905E-02  2.156436070832E-02 -2.624745229043E-01
1     -3.118388852293E-01  1.333438306584E-01 -3.228685284194E-01
1     -3.403156096494E-01 -1.982400318802E-01 -1.157824844356E-01
1     -2.869510613627E-01 -8.418976395712E-02  4.756872577177E-02
7      4.052599333247E-01  3.611547587878E-03 -2.964084420881E-01
7     -3.482877872090E-01 -2.004714602231E-02 -2.045486619658E-01
7     -1.370629884135E-02  1.239068455614E-01 -3.942995393365E-02
7     -9.996263652745E-02  2.044290396001E-01  3.204467273623E-02
8      3.112065051423E-01  7.742872382639E-02 -4.246789313176E-01
8      2.919005769488E-01 -5.519244772114E-02 -2.454349230033E-01
8      5.311693851955E-02  2.484511082277E-01  1.848582062126E-01
8     -3.250235105003E-01  2.228619176219E-01 -6.094957200050E-02
OPTGEOM
TOLDEG
0.000010
TOLDEX
0.000040
END
END
8 4
0 0 6 2.0 1.0
 5484.671700          0.1831100000E-02
 825.2349500          0.1395010000E-01
 188.0469600          0.6844510000E-01
 52.96450000          0.2327143000
 16.89757000          0.4701930000
 5.799635300          0.3585209000
0 1 3 6.0 1.0
 15.53961600         -0.1107775000          0.7087430000E-01
 3.599933600         -0.1480263000          0.3397528000
 1.013761800           1.130767000          0.7271586000
0 1 1 0.0 1.0
 0.2700058000           1.000000000           1.000000000
0 3 1 0.0 1.0
 0.800000000           1.00000000
7 4
0 0 6 2.0 1.0
        4173.51100          0.183480000E-02
        627.457900          0.139950000E-01
        142.902100          0.685870000E-01
        40.2343300          0.232241000
        12.8202100          0.469070000
        4.39043700          0.360455000
0 1 3 5.0 1.0
        11.6263580         -0.114961000          0.675800000E-01
        2.71628000         -0.169118000          0.323907000
       0.772218000           1.14585200          0.740895000
0 1 1 0.0 1.0
       0.212031300           1.00000000           1.00000000
0 3 1 0.0 1.0
 0.800000000           1.00000000
6 4
0 0 6 2.0 1.0
    .3047524880D+04   .1834737130D-02
    .4573695180D+03   .1403732280D-01
    .1039486850D+03   .6884262220D-01
    .2921015530D+02   .2321844430D+00
    .9286662960D+01   .4679413480D+00
    .3163926960D+01   .3623119850D+00
0 1 3 4.0 1.0
    .7868272350D+01  -.1193324200D+00   .6899906660D-01
    .1881288540D+01  -.1608541520D+00   .3164239610D+00
    .5442492580D+00   .1143456440D+01   .7443082910D+00
0 1 1 0.0 1.0
    .1687144782D+00   .1000000000D+01   .1000000000D+01
0 3 1 0.0 1.0
    .8000000000D+00   .1000000000D+01
1 3
0 0 3 1.0 1.0
    .1873113696D+02   .3349460434D-01
    .2825394365D+01   .2347269535D+00
    .6401216923D+00   .8137573262D+00
0 0 1 0.0 1.0
    .1612777588D+00   .1000000000D+01
0 2 1 0.0 1.0
    .1100000000D+01   .1000000000D+01
99 0
END
DFT
B3LYP
XLGRID
END
EXCHSIZE
4050207
BIPOSIZE
4050207
TOLINTEG
7 7 7 7 14
MAXCYCLE
100
SCFDIR
TOLDEE
8
SHRINK
4 8
LEVSHIFT
5 1
FMIXING
40
END
END
```