Scientific Computing & Modelling

Installation Manual

ADF Program System
Release 2010

Scientific Computing & Modelling NV
Vrije Universiteit, Theoretical Chemistry
De Boelelaan 1083; 1081 HV Amsterdam; The Netherlands
E-mail: support@scm.com

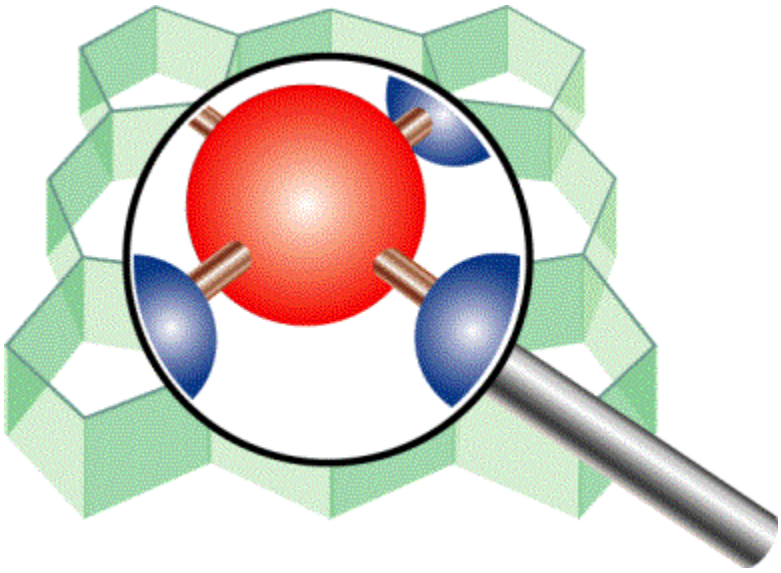

Table of Contents

1 Introduction

The Installation Manual describes the installation of the ADF program package on the platforms on which it is supported. See also the ADF installation manual for Windows and the ADF installation manual for Mac OS X. For optimal performance, some system specific tuning may be needed. We therefore strongly recommend to also read Chapter 4 of this document. This Installation Manual is available from the Documentation part of the SCM web site:

http://www.scm.com/Doc/Welcome.html

The main programs contained within the ADF package are ADF and BAND. There are also several utility and property programs. These are used primarily for pre- and post-processing data of the main programs. You will always install the complete ADF package at once. It includes ADF, BAND, and all available utility programs, as well as all of the ADF-GUI modules.

The ADF package and support scripts are written for use within a UNIX environment (including Linux). If you want to install it and use it effectively you will need some basic UNIX knowledge. We do **not** provide this general UNIX introduction. If you are new to UNIX we strongly suggest you first make yourself comfortable with this operating system. Very nice tutorials and introductions are available for free on the web, simply search for 'UNIX introduction' or 'UNIX tutorial'.

Typically installation of the ADF package is simple and straightforward. If you have problems installing it, contact us for assistance at support@scm.com.

1.1 Requirements

The requirements for the ADF package vary from platform to platform. A list of supported platforms, with information on the operating system (versions), parallel environment (MPI, PVM, ...), and compilers used is available on the Download section of our web site:

http://www.scm.com/Downloads/Welcome.html

Memory

The amount of memory you need depends greatly on the kind of calculation you perform. For most calculations 256 MB per CPU will be sufficient, but if you have lots of memory available most program will run significantly faster. It also reduces the amount of disk I/O, which may speed up your calculation depending on the I/O sub-system and the operating system.

Memory requirement grows with the system size. For example, for a molecule containing 100 atoms with a DZ basis set it may be sufficient to have 256 MB per CPU but for a molecule with 1000 atoms up to 8 **giga**bytes per CPU may be required. Also, if you are going to perform TDDFT, relativistic spin-orbit or analytical frequency calculations then the amount of memory should be larger. As an indication, an analytical vibrational frequency calculation of a organometallic complex containing 105 atoms with a TZP basis set uses up to 1GB of RAM per process but it can be done with less, even though not as efficiently.

Disk

For the installation of the package on Linux/Unix you need from less than 256 MB (without sources) to 1 GB (with sources and compiled objects). On Windows, about 900MB of disk space is required for ADF installation. The (scratch) disk space requirements for running ADF greatly depend on what type of calculation you perform. For most ADF calculations 1 GB of disk space will be enough. For BAND calculations you may need from a few to up to a few tens of gigabytes (for large calculations) of free disk space.

Operating System

The package runs on the following Unix variants: Linux (i686, x86-64, ia64, PowerPC), IBM AIX, Sun Solaris (i386, SPARC), MacOS X.

For Macintosh OS X you will need at least version 10.5, and you need the X11 environment as well. Starting with 10.5, it should be installed by default.

If your UNIX platform is not listed, please contact us for more information.

A Windows version is also available. It has been tested under Windowsi 7, XP and Vista. See the ADF installation manual for Windows for more details.

Network

First of all, on the most popular systems (Linux, MacOS and Windows) a network card must be present in the computer as its hardware MAC address is used as the computer's ID for licensing purpose. Besides that, when using Platform-MPI, the computer must be connected to a network and have a valid IP address and the address must not change while an MPI job is running. This requirement might be lifted in a future version of Platform MPI.

As far as performance concerned, a switched (Gigabit or Fast) Ethernet network is typically sufficient for good parallel performance on up to eight nodes. If you are going to use more nodes you may need faster communication hardware, such as Myrinet or Infiniband, to get the best performance.

All commercial parallel machines (SGI, IBM, HP, ...) have very good communication facilities, so the package should run fine on those platforms as far as the communication system is concerned.

Compiler

If you have a license for the source code, you can compile and link the source yourself, with or without your own modifications.

The source consists mainly of Fortran95 code, with some small parts written in C. You need to use the same Fortran and C compilers as we are using on the same platform, since some parts of the code are available only as object modules. It is very unlikely that other compilers will work with these object modules. We cannot support compilers other than what we are using ourselves.

To check what compiler to use, check the detailed machine information on the Download section of our web site.

2 Installation

To install the ADF package you have to go through the following steps:

- 1. Decide which version to install (serial or MPI)
- 2. Download and unpack the software
- 3. Install external libraries (optional)
- 4. Set up your environment
- 5. Source distribution only: compile the source
- 6. Generate license information
- 7. Setup a nodeinfo file (PVM only)
- 8. Setup scratch space
- 9. Run some tests
- 9. Clean up

Below we discuss each step separately.

2.1 Decide which version to install

If you are installing the package on a single machine that has only one CPU you should install the **serial version**. However, the serial version is not available on all platforms. If this is the case for your platform then you can just install any parallel version.

If you are installing the package on a machine with more than one CPU or on a multi-core machine, or if you are installing ADF on a cluster, you should install a **parallel version**. Please note that the Hyperthreading technique available in some processors does not improve ADF performance. Thus, even though a hyperthreaded processor is seen by the system as two logical processors, it should be considered as one.

For most systems, only an MPI version is provided but for some systems there is also a serial version. If there is no serial version available for your system then you can run an MPI version in serial mode.

PVM is no longer supported by SCM.

2.2 Install pre-requisites (possibly optional)

If you have downloaded an HP-MPI/Platform-MPI or OpenMPI version of ADF, then you do not need to install any additional MPI libraries because these are provided by SCM. If this is the case, then you can move on to the next section.

If, however, for whatever reason, you want to build ADF with a different MPI library then below you will find some information and web links we have gathered about major MPI implementations supported by ADF. Additional information about different MPI libraries, caveats and troubleshooting may be found in the section 4 of this document.

Platform MPI 7 (formerly HP-MPI)

Platform Computing has MPI implementations for the most popular operating systems: x86- and ia64-based Linux, Windows, as well as for HP-UX. Full documentation about HP-MPI is available at the HP website and the information about Platform MPI can be found on the Platform Computing site. Platform-MPI is distributed with (a Platform-MPI versions of) ADF and a complete Platform-MPI directory tree is found in $ADFBIN/platform_mpi. Using Platform-MPI with ADF does not require any additional licenses. In addition to TCP/IP and shared memory supported by every MPI implementation, HP-MPI also supports (without recompilation) all well-known interconnects such as Infiniband and Myrinet. The best available interconnect is chosen

automatically at run time and this can also be modified using environment variables or mpirun command-line switches in the $ADFBIN/start script.

OpenMPI

OpenMPI is an open-source implementation of the MPI standard. You can get both the software and documentation from the OpenMPI website:

http://www.openmpi.org

For some platforms (currently only Mac OS), ADF comes bundled with OpenMPI so you do not have to install it separately. However, you **do** need to install OpenMPI if you are going to recompile ADF and if you want it to use a fast interconnect such as Infiniband or if you need to enable OpenMPI integration with the batch system on your cluster.

Intel MPI

Intel provides its own MPI implementation that is similar to OpenMPI in many ways. ADF supports Intel MPI at the configure script level on ia32-, i64- and ia64-based Linux platforms.

MPICH1

MPICH1 is a portable, though outdated, implementation of the MPI v1 standard. You can get both the software and documentation from the MPICH1 web site:

http://www-unix.mcs.anl.gov/mpi/mpich1/

Vendor-specific MPI implementations

Many hardware manufacturers, such as SGI, IBM, Sun and Cray, have their own MPI implementations that work only on their hardware and/or operating system. In this case, SCM provides an ADF version built with the appropriate MPI library but the run-time environment (RTE) is not distributed since you will most likely have the RTE already pre-installed on your machine. Please contact the machine vendor's support staff if this is not the case.

PVM

NOTE: SCM strongly discourages the use of PVM. It is not actively developed, it does not use modern fast interconnects and it is significantly slower than any modern MPI implementation. So it should only be used as the last resort when no other parallel option works.

Starting from ADF2010, PVM is no longer supported by ADF.

2.3 Download and install the software

If you have not already done so, you need to download the software from the Download section of the SCM web site. You will be asked for a username and password that you should have received from SCM. Contact admin@scm.com if you forgot your username and/or password.

The binary download file, which must be downloaded in all cases, contains all binaries (ADF, BAND, COSMO-RS, ADF-GUI, and BAND-GUI modules, GENNBO, property programs, utility programs), documented examples for ADF and BAND, the basis sets and force field files. In some cases (HP-MPI/ Platform-MPI and OpenMPI) the binary file also contains the corresponding MPI run-time environment. Your license file will determine which of these you can actually use. You need to download the binary tarball for your platform also if you want to compile ADF from sources. The source code itself needs to be downloaded separately if it is included in your license (not the default).

- Download the adf2010.01.bin.tgz and, if applicable, adf2010.01.src.tgz for your platform (from www.scm.com).

Important: skip the step below if you have a source code license and are going compile ADF yourself and move on to the next section.

If you want to install the precompiled executables only then gunzip and untar the downloaded files:

```
gunzip adf2010.01.bin.tgz
tar -xf adf2010.01.bin.tar
```

A new directory adf2010.01 will be created containing your new adf installation.

2.4 Set up your environment

You will need to define a couple of environment variables, preferably in your login script. For your convenience, a sample adfrc.sh script is provided with ADF, which you can find in $ADFHOME.

If you are using a batch system, such as PBS, SGE or LSF, then make sure that the variables are set not only for interactive work but also for batch use. In a distributed parallel environment, like a Linux cluster, make sure that these environment variables are set on all nodes.

The most important environment variables are automatically passed on from the master process to the child processes in case of a parallel run. Thus the values set on the node where you start the program will override any value you may have set on the nodes where the child processes are running. Typically, you should still set the environment variables on all nodes since you won't know in advance on which node the master will run.

Hint: set the SCM_DEBUG environment variable to "yes" before running ADF to see values of relevant environment variables for each process. This may be especially useful if you suspect that some variables are not propagated through mpirun command to the master node.

The following environment varables must always be set for all ADF versions.

Name	Possible value	Description
ADFHOME	$HOME/adf2010.01	full path of the ADF installation directory
ADFBIN	$ADFHOME/bin	full path ADF directory for scripts and binaries
ADFRESOURCES	$ADFHOME/atomicdata	full path of the directory with the ADF database (basis sets and so on)
SCMLICENSE	$ADFHOME/license.txt	full path to the license file
SCM_TMPDIR	/scratch/$USER	full path of the directory where all processes will create their temporary files. See also sections 2.7 and 4.1

The following environment variables may be required for any parallel (MPI or PVM) version.

Name	Possible value	Description
NSCM	16	default number of parallel processes; do **not** put it in the shells's rc file but rather set this variable per job

The following environment variable is optional and is relevant for some MPI versions only.

Name	Possible value	Description

SCM_MACHINEFILE	$HOME/.machine	full path-name of the file containing a list nodes to use for a job; **Important:** this variable should be set **per job** and it should **only** be set if multiple computers are used without a batch system. The file pointed to by the variable must already exist and it must contain a list of nodes on which a particular job is about to be executed, with their processor count. See section 4.2 for more information.

The following environment variables are obsolete as they are relevant for a no longer supported PVM version only.

Name	Possible value	Description
PVM_ROOT	$HOME/pvm3	full path of the main PVM installation directory
PVM_ARCH	SGI6	architecture code for your machine (NOTE: use SGI6 on SGI IRIX)
SCMSPAWNSCRIPT	$ADFBIN/adfs	full path of the script used by the PVM daemon to start slave nodes

The following environment variables are relevant for the GUI modules

Name	Default Value	Description
SCM_ERROR_MAIL	no error e-mail sent	e-mail address for error reports
SCM_GUIRC	$HOME/.scm_guirc	location of the preferences file
SCM_TPLDIR	none (no extra templates loaded)	location of the templates directory
SCM_STRUCTURES	none (no extra structures loaded)	location of the structures directory
SCM_RESULTDIR	none (current directory used)	location of the results directory
DISPLAY	required (for all X11 programs) except on Windows	X-window display to use
SCM_MOPAC	none (default script $ADFBIN/ mopac.scm will be used)	command to start MOPAC
SCM_QUEUES	none (ADFjobs will search the remote $HOME/.scmgui)	path to the dynamic queues directory

The following environment variables are relevant for source distributions only, and only at the configure time.

Name	Possible value	Description
MPIDIR	/opt/hpmpi	full path of the directory where MPI library is installed. This variable is required at configure time when a non-standard MPI library is used. **Important:** Do **not** set MPIDIR when building an HP-MPI or OpenMPI version of ADF with MPI libraries delivered with ADF.
MATHDIR	/opt/intel/mkl/10.0.011	(optional) installation directory of the Intel MKL library

The following environment variables may be set to modify other aspects of ADF execution. All of them are optional and some are used for debugging only.

Name	Possible value	Description
SCM_IOBUFFERSIZE	64	Size of memory buffer for scratch files, in megabytes. See also section 4.

SCM_VECTORLENGTH	128	Block length used in the numerical integration routines. See also section 4
SCM_RCV_TIMEOUT	900	(PVM only) time (seconds) to wait for messages
SCM_MAX_RCV_TIMEOUTS	2	(PVM only) maximum number of timeouts
SCM_ALARMTIME	1800	Time (seconds) to wait for messages using ALARM signal, 0 disables timeouts. Currently the default setting is zero, because this feature may not work on some platforms.
SCMBCOP	1	Broadcast algorithm (not to be changed for MPI versions)
SCMGAOP	5	Gather algorithm (not to be changed for MPI versions)
SCMCBOP	1	Combine alogrithm (not to be changed for MPI versions)
SCM_DEBUG	yes	Set to debug SCM_TMPDIR usage and environment variables (also see above)
SCM_NOMEMCHECK	yes	Disable memory checks
SCM_NODOMAINCHECK	yes	Disable domain check for license
SCM_TRACETIMER	yes	For debugging only: produce additional output on ADF internal timer entry/exit
SCM_DEBUG_ALL	yes	For debugging only: equivalent to specifying DEBUG $ALL in the input

2.5 Source distribution only: unpack, configure and compile the source

The downloaded sources should be unpacked first (i.e. before unpacking binaries), for example, as follows:

```
gzip -cd adf2010.01.src.tgz | tar xf -
```

The result will be a new adf2010.01 directory containing the sources.

Next, unpack the binary distribution **without changing the current working directory**. This installs a couple of programs in the $ADFBIN directory that are not included in the source distribution.

```
gzip -cd adf2010.01.bin.tgz | tar xf -
```

After unpacking everything and setting up your environment properly (see previous section), you should run the configure script. This script is located in the $ADFHOME/Install directory, and it must be executed from the $ADFHOME directory. The script replaces some files in the bin directory with versions specifically targeted for your system. For example, if you want to install an OpenMPI version of ADF on a platform for which only a Platform-MPI version is available, then *configure* will make an $ADFBIN/start script suitable for OpenMPI and overwrite the old one which was made for Platform-MPI. Further, *configure* creates the Makeflags and settings files that you will need to compile ADF.

Example:

```
cd $ADFHOME
Install/configure -p platform_mpi
```

The configure script accepts the following options:

- **-h** with this option configure will print the system name, processor type and, for Linux, the GLIBC version. It will also print the default machine name used internally by ADF and a list of known parallel implementations for this combination, as well as a list of possible optional parameters, if any (see below).

- **-p par** specifies the parallel implementation to use. The list of possible **par** values varies per platform and can be obtained by running configure with the **-h** option. The default is "serial".

- **-o option** specifies an optional parameter if available. For example, on IBM AiX you can specify *-o 64* to configure a version with 64-bit addressing.

Next, you can compile and link the ADF sources by executing the yam (Yet Another Make) script created in the $ADFBIN directory by configure.

```
cd $ADFHOME
bin/yam
```

After a while, depending on the speed of your computer, ADF should be ready to use, just as if you had installed the precompiled executables.

2.6 Generate license information

Most of the programs that form the ADF package require a license file to run. This file contains information about your machine, the version(s) of the package for which you have a license, and the end date of your license.

If you are a trial user, then you should have already received a valid 'demo' license file by e-mail. If you install ADF from a CD, then you should be able to find a demo license file on the CD. In most other cases you will receive a host-locked license file from SCM. To generate this file SCM needs some information from you about your machines. You can generate this information by running a program from the package on all machines where ADF is installed as follows:

```
$ADFBIN/dirac check
```

If a license file already exists, this command will show some information about your license such as the date until which it is valid and list of packages it is valid for with their respective version numbers. It will also show you up to how many processes can be in an ADF parallel job. An explanatory error message will be printed if there is no valid license found. After that, information about the computer's ID is displayed. This information is necessary for SCM to generate a license file for this particular computer. To obtain only the information needed for generating license file you can use the following command:

```
$ADFBIN/dirac info
```

Thus, in order to obtain a license file it is sufficient to send to license@scm.com output of the *$ADFBIN/dirac info* command but in case of problems with the license one should send output of the *$ADFBIN/dirac check* command together with the support request.

After receiving the machine ID(s) SCM will prepare a license file matching your license conditions and e-mail it to you with instructions on how to install it.

After receiving a license file you will need to save it to an appropriate location on your computer (usually in the directory where you installed ADF) and set the environment variable $SCMLICENSE pointing to the file. For example, if you saved the license file named *license.txt* in */opt/ADF* then SCMLICENSE must be set to */opt/ADF/license.txt*. Add the SCMLICENSE definition to your login script.

Ensure that permissions on the license file allow read access for everyone who is going to run ADF.

After installing the license you should make sure everything is fine by running:

```
$ADFBIN/dirac check
```

This will produce the output similar to the following:

```
Checked: /home/testadf/license

License termination date (mm/dd/yyyy):  7/10/2010

According to it, you are allowed to run:

   ADF version  2010.99
  BAND version  2010.99
ADFGUI version  2010.99
BANDGUI version  2010.99
   GUI version  2010.99
 Utils version  2010.99
   NBO version  2010.99
   CRS version  2010.99

Number of procs you are allowed to use for:

ADF  :     128 procs
BAND :     128 procs

=====================================================
release: 2010.01
:chem.vu.nl:
:Linuxmycomputer00:15:17:76:EC:5C:
ncores :           4
=====================================================
LICENCE INFO READY
```

Please refer to ADF for Windows release notes for details on how to obtain the license information under Windows.

Corrupted License File

You may find that, after having installed the license file, the program still does not run and prints a message "LICENSE CORRUPT". There are a few possible causes. To explain how this error may come about, and how you overcome it, a few words on license files.

Each license file consists of pairs of lines. The first of each pair is text that states in a human-readable format a couple of typical aspects: A 'feature' that you are allowed to use (for instance 'ADF'), the expiration date, a (maximum) release (version) number of the software and so on. The second line contains the same information in encrypted format: a long string of characters that appear to make little sense. The program reads the license file and checks, with its internal encrypting formulas, that the two lines match. If not, it stops and prints a "LICENSE CORRUPT" message.

So, there are two common reasons why this may happen:

1. You are using a license file for a version of the software other than your executables correspond to. Newer (major) releases may use a different encryption, so that the match in old license files is not recognized anymore. In particular, the "LICENSE CORRUPT" error will arise if you run ADF1999 (or later) with a license file that was generated for ADF2.3 (or earlier). Verify that your license file and executable belong to the same major release.

2. The license file as it has been created has been modified in some way. Sometimes, people inspect it and 'clean it up' a little bit, for instance by removing 'redundant spaces', or by making some other 'improvements'. Any such modification will break the encryption match and lead to the "LICENSE CORRUPT" error. Sometimes the reason lies in the mailing system: if the file contains long lines the mailer may break them into shorter lines. To verify (and correct) this open the license file in a text editor and see if it really consists of pairs of lines as described above. If not, re-unify the broken lines and try again.

You can use the **fixlic** utility to try to fix this automatically. Please be aware that the **fixlic** utility will try to fix the file pointed to by the $SCMLICENSE environment variable and replace it with the fixed copy. Thus, you need to make a backup of your license file first and you need to have write permissions for it.

```
cp $SCMLICENSE $SCMLICENSE.backup
$ADFBIN/fixlic
```

2.6.1 Floating License

If you have a floating license, you will need to follow the instructions below to make it work. If you do not have a floating license, you may skip this section.

Create a floating license directory

Make a new directory to keep track of the running ADF processes. This directory must be located on a file system that is shared between all nodes on which you want to run ADF with this floating license.

Thus, for example, if you wish to use the name FloatADF to count the processes:

```
cd $ADFHOME
cd ..
mkdir FloatADF
chmod 1777 FloatADF
```

In this example, we have given all users read, write and execute permissions to this directory. If you wish to restrict this for security reasons, you may do so as long as all ADF users will have read, write and search permission for this directory.

Note that the FloatADF directory was created in the same directory in which your $ADFHOME direcotry is located. In that case it will probably accessable by everyone who needs to access it.

It should NOT be located inside your $ADFHOME directory as that will make it more difficult to install new ADF versions.

the FloatADF directory should not be moved, deleted or renamed for the duration of your license. Doing such things will invalidate your license! And it cannot be fixed by you without requesting a new license.

E-mail us the license information

Send the result of the following commands (using again the name FloatADF as an example) to license@scm.com:

```
cd $ADFHOME
cd ..
ls -lid FloatADF
```

Also, execute the command below on each machine (or cluster node) where you want to run ADF (the ADF package must be installed before running this command) and include the output in your e-mail.

$ADFBIN/dirac info

Important:

- You need to mail the **exact output** of the above command, thus with a **full** path to the FloatADF directory.
- You should **not** delete, move or rename this directory once you have sent this information to SCM. Doing so will break your floating license.

Install license file

SCM will send you a license file by e-mail. Its installation is identical to that for non-floating licenses. Copy it to your system (most conveniently also in a shared directory) and make sure all users have read permission to it:

```
cp license.txt $ADFHOME/license.txt
chmod ugo+r $ADFHOME/license.txt
```

Next make sure ADF will use this license file by setting the SCMLICENSE environment variable (Bourne shell syntax is shown):

```
SCMLICENSE=$ADFHOME/license.txt
export SCMLICENSE
```

2.7 Set up scratch space

Most programs from the ADF package use disk for temporary data. This data often takes a significant amount of space and is used frequently. To avoid run-time errors and performance loss you may want to make sure that the file system used for temporary files is both big and fast. Use the SCM_TMPDIR environment variable to tell the programs where to create their scratch directories. Please note that SCM_TMPDIR should always be set. If it is not set then each process will create its own directory in the current working directory where is was started.

You can set the SCM_TMPDIR environment variable, for example, as follows:

```
SCM_TMPDIR=/local/scratch/$USER
export SCM_TMPDIR
```

In the next few paragraphs we will explain exactly what this variable does.

Child processes

Every parallel job consists of one master and one or more slave tasks. Master and slaves behave a bit differently with respect to their scratch directories. Slave processes will always create a directory for their scratch files in $SCM_TMPDIR and chdir to it to avoid any risk that shared files are updated by more that one process at the same time. For efficiency reasons, that directory should reside on a local disk unless you are using very, very fast shared file system for large files. You need write access to that directory, and the file system should have enough free space. Please note that the SCM_TMPDIR environment variable will be passed from the master to slaves.

After the job is finished, slave processes will delete their scratch directories. This can disabled by setting the SCM_DEBUG environment variable to any text, for example, to "yes". In this case the scratch directory and all its contects will be left intact.

Each slave writes its text output to a file called KidOutput located in its scratch directory. In case of an error this file will likely contain some sensible error message. If an error occurs then in order to avoid losing the file ADF will copy it to the directory, from which the job was started, as KidOutput__#, where # is the process' rank.

Master process or serial runs

The master process (which is the only process in a serial run) will also create its temporary files in its own sub-directory of $SCM_TMPDIR. There are some exceptions. Some files, such as logfile and TAPE13, will be created in the directory where ADF was started because they are not performance-critical but are convenient to have in the current directory for different reasons. For example, logfile is nice to have in the current directory in order to follow the calculation progress and the TAPE13 is an emergency restart file that can be used if ADF crashes or is killed.

At the end of a calculation, the master will copy all result files from its scratch directory to the directory where it was started.

Using multiple disks

Sometimes, if you have multiple non-RAID disks in your system, you may want to spread scratch files across different physical disks to get better performance. It is possible to request that every ADF MPI-rank creates its files in a different directory by adding "%d" in $SCM_TMPDIR. If a "%d" string is encountered in the value of SCM_TMPDIR variable it will be replaced by the MPI rank number of the process at run-time. This means, however, that you may have to create up to 128 or more (depending on the maximum number of processes in one parallel calculation) symbolic links on each node where ADF is supposed to run. You should also create a directory matching the SCM_TMPDIR's value literally so that any process that does not interpret '%d' could also run. Example: suppose there are two scratch file systems, /scratch1 and /scratch2 located on two different physical disks of an 8-core workstation. We want the odd rank numbers to use /scratch1 and the even numbers to use /scratch2. One of the ways to achieve this is to create an empty /scratch directory and create nine symlinks in it as follows:

```
ln -s /scratch1 /scratch/%d
ln -s /scratch1 /scratch/0
ln -s /scratch2 /scratch/1
ln -s /scratch1 /scratch/2
ln -s /scratch2 /scratch/3
ln -s /scratch1 /scratch/4
ln -s /scratch2 /scratch/5
ln -s /scratch1 /scratch/6
ln -s /scratch2 /scratch/7
```

After that set SCM_TMPDIR to "/scratch/%d" and the ranks 0, 2, 4, 6 will use /scratch1 while ranks 1, 3, 5, and 7 will go to /scratch2. When running ADF on a cluster it is better to combine multiple disks in a RAID 0 (striping) configuration.

2.8 Run some examples and test performance

Verify that you can now run examples provided with ADF and that they give correct results. We recommend that you consult the Examples document for notes on such comparisons: non-negligible differences do not necessarily indicate an error.

Note: the sample runs are complete Bourne shell scripts that should be executed as such, they are **not** input files to be fed into any program.

Test parallel performance

It is very important to make sure that computer resources are utilized with the maximum efficiency. Therefore, you should check that each ADF job uses all processors/cores allocated to it and that the network is not overloaded with the disk I/O traffic.

Processor usage: Typically, when you submit a parallel job to the batch system you have a possibility to specify how many processors per node (ppn) to use. If the batch system you are using allows this, then

make sure that you request ppn equal to the number of physical cores on each node. The latter is usually equal to the number of logical CPUs reported by the system, except when hyperthreading is turned on. With the hyperthreading on, you should set ppn to half of the number of logical processors.

It is also possible that your batch system does not allow you to specify ppn but instead it always assumes that there only one processor per node. In this case, you will need to edit the $ADFBIN/start file and add some commands for processing the hostfile.

So, now it is time to see that everything is working as intended. To this end, create a reasonably small ADF job and submit it, possibly to run on more than one node. After the job has finished, open the job's .out file and find the table that looks like the following:

```
 Parallel Execution: Process Information
 ================================================================================
 Rank   Node Name                                NodeID   MyNodeRank  NodeMaster
    0   compute-0-0                                 0           0           0
    1   compute-0-0                                 0           1          -1
    2   compute-0-0                                 0           2          -1
    3   compute-0-0                                 0           3          -1
 ================================================================================
```

Check the names in the "Node Name" column and verify that the number of tasks per node is correct. If this looks as expected then move to the end of the file to the table called "Timing Statistics". Before the table, there is a line of text beginning with "Total Used" that might look as follows:

```
 Total Used :   CPU=     8064.87      System=     1144.31     Elapsed=
9212.99
```

This shows how much time (in seconds) was spent in the ADF code (CPU) and in the kernel (System), and how long did it take for the calculation to complete (Elapsed). Ideally, the system time should be small compared to the CPU time and the latter should be close to the Elapsed time. The system time will not always be a small number, however, the sum of the System and CPU times should always give a number very close to the Elapsed time. If this is not the case then it means that ADF has to spend a lot of time waiting for something. This can be, for example, disk I/O or network communication. Note: with Platform-MPI the network communication time contributes to the System time while with other MPI implementations it may contribute to the difference between the Elapsed and the CPU+System time.

If you notice that the system time portion is enormously large or that there is a large difference between the CPU+System and the Elapsed time, then repeat the job with the "PRINT TimingDetail" keyword in the input and contact the SCM support.

2.9 Clean up

At this point you may clean up the $ADFHOME directory a little.

You may remove the *.tar files if you have not already done so, and if you are short on disk space you may remove the examples directories. We do advise, however, to keep them around since they contain useful examples that may be used to get acquainted with ADF.

3 Directory structure of the ADF package

Below is the list of major parts of the ADF package.

3.1 The bin directory

When the programs have been installed, the binary executables have been placed in $ADFBIN, typically $ADFHOME/bin, with names like 'adf.exe', 'band.exe', and so on. There will also be files 'adf', 'band'. The latter are scripts that execute the binaries.

You should use the script files, rather than the executables directly. The script files take care of several convenient features (like the BASIS key described in the ADF User's Guide) and provide a correct environment for running in parallel. See also the sample run scripts and the Examples document.

The $ADFBIN/start script takes care of setting up the environment and starting the binaries. If necessary, it parses the information provided by a batch system and sets up a machinefile (or an appfile) and runs tasks in parallel. Edit the file if you need to customize the way MPI programs are started.

3.2 The atomicdata directory

The directory atomicdata/ contains a large amount of data needed by programs from the ADF package at run time. For example, it contains basis sets for all atoms. Generally you should **not** modify any of the files in this directory.

The basis sets are described in detail in the ADF BasisSet document, available from the Documentation part of the SCM web site (http://www.scm.com).

3.3 The examples directory

The directory examples/ contains sample script and output files. The files with extension .run are shell scripts that also contain embedded input. Thus, these files should be executed, and not given as input to ADF.

The example calculations are extensively documented in the Examples document, available from the SCM web site (http://www.scm.com).

3.4 The source code (adf, band, libtc, Install, ...)

The source code files found in the program and library directories and subdirectories thereof have extensions .d, .d90, .dmod* or .cd and need to be pre-processed by the parser (scu) before compilation. The parser produces files with a suffix .f, .f90 or .c containing plain FORTAN or C code. These files are placed in the FFILES directory in each program's or library main directory. Other source files (used by the parser) are include files (files with extension .fh or .h) and files that define configuration parameters (the file "settings").

Parsing and compilation are controlled by *$ADFBIN/yam*.

The Install directory contains a script configure, some data files which provide generic input for configure (start, starttcl, and some more), a portable preprocessor cpp (based on Mouse cpp), the adf preprocessor (adfparser) and files that will be addressed by the *$ADFBIN/yam* command.

Apart from a few more files that are of little importance in the installation, you'll find in Install/ a number of subdirectories with names like pentium_linux_ifc, etc.: they contain precompiled binaries for supported platforms.

Install/configure usually guesses the configuration parameters from the system but in some cases you may want to run it interactively. In this case, start it without command-line parameters and it will ask you some questions: hardware platform, parallel variety (PVM, MPI, serial), and the maximum number of CPU's to use.

Based on the files in the Install directory and the chosen configuration parameters, the configure script will create in $ADFHOME/ two files: settings and Makeflags. It will also populate the bin directory with important files.

3.5 The Doc directory

All the user documentation for ADF is present in the PDF format in $ADFBIN/Doc. The same documentation is also available on the SCM website

3.6 The scripting directory

This directory contains some useful scripts that are part of the ADF package but are distributed under the GNU General Public License.

4 Additional Information

4.1 Performance-related environment variables

You can customize and optimize your ADF package by changing or setting several environment variables, and/or use some input options. We advise you to test the effects of changing them for some of your typical calculations, and then setting the optimal values in the scripts that define the environment for your login shell. Don't forget to make sure that the changes you make will also apply for batch jobs, and for all nodes of a distributed computing system.

A complete list of environment variables used by the ADF package can be found in the section 2.4. Here we will discuss a few that are particularly important.

SCM_IOBUFFERSIZE

Most programs within the ADF package use the KF I/O system. This is coupled to a facility to store (parts of) files in memory, if you have enough memory available. Thus, the programs will cache the IO data instead of the operating system, and reduce the amount of IO significantly.

It depends on your operating system and hardware if you can benefit from this scheme. The default buffer size depends on the platform but is typically set to 64 megabytes, which should be sufficient for running small jobs without much disk I/O. In some cases you can have a major performance improvement by making this buffer much larger, for example 512MB. You do this by setting the SCM_IOBUFFERSIZE environment variable to a number corresponding to the buffer size **in megabytes**.

Please try for yourself, with your typical calculation on your production machine to find out the optimal value for your situation.

SCM_VECTORLENGTH

Almost all programs within the ADF package use numerical integration, and this is normally the most time-consuming part of the code. Numerical integration involves summing together results for each 'integration point'. By handling a number of points together you can greatly influence the performance. The number of integration points handled together is called the block length.

If the block length is too small, you will have a significant overhead and the programs will be very slow.

If the block length is too large, lots of data will not fit in cache memory and again the program will not run at the maximum speed.

The optimal block length is somewhere in between, ranging from 32 to 4096 depending on your hardware. Sometimes it pays off to set the block length explicitly NOT to a power of 2 to avoid memory bank conflicts.

Again, try for yourself, with your typical calculation on your production machine to find out the optimal value for your situation. On most machines, 128 is a good compromise value.

SCM_TMPDIR

This environment variable determines where all processes create their scratch files. Setting this variable correctly is extremely important for performance. If this variable is not set, the scratch directories will be created in the current working directory. For a detailed description see section 2.7.

4.2 MPI

MPI (Message Passing Interface) is a standard describing how to pass messages between programs running on the same or different machines.

MPI is a formal standard and it is actively supported by all major vendors. Some vendors have highly-optimized MPI libraries available on their systems. There are also a couple of open-source implementations of the MPI standard, such as MPICH and OpenMPI. There are also numerous commercial MPI implementations that support a wide range of systems and interconnects, for example, Platform-MPI and IntelMPI.

Support for a particular MPI implementation in ADF can be considered at three levels: the source code, the configure script, and pre-compiled binaries. At each level different MPI implementations may be supported.

The ADF source code is not implementation-specific and thus theoretically it supports any MPI library. Many popular MPI implementations are supported at the level of the configure script. For example on 32-bit Linux these are: MPICH1, Intel MPI, OpenMPI, Platform-MPI, HP-MPI, LAM-MPI, and Score. This means that proper compiler flags will be used and and an appropriate $ADFBIN/start script will be generated at configure time.

However, when choosing an MPI implementation for pre-compiled binaries SCM considers many factors including (but not limited to) the re-distribution policy, performance, and built-in support for modern interconnects. Platform-MPI is currently the standard MPI implementation supported by SCM because it has the most favorable combination of these factors at this moment. For platforms where Platform-MPI is supported it is distributed with ADF. A different MPI implementation will be standard on a platform where Platform-MPI is not available. It may or may not be distributed with ADF. For example, SGI MPT is standard on SGI machines and OpenMPI is standard on MacOS platforms, but only the latter is distributed together with ADF.

When pre-compiled binaries do not work on your computer(s) due to incompatibility of the standard MPI library with your soft- and/or hardware, the SCM support staff will be glad to assist you in compiling ADF with the MPI implementation supported on your machine(s).

If you are going to use an MPI version of the ADF package, and it not Platform-MPI or OpenMPI, you will need to determine if the corresponding MPI run-time environment is already installed on your machine. If not, you will need to install it separately from ADF. As it has been already mentioned, Platform-MPI and OpenMPI are distributed together with the corresponding version of ADF so you don't need to worry about installing them separately.

Running with MPI on more than one node

When running on more than one machine (for example on a cluster **without** a batch system) you need to specify a list of hosts on which mpirun needs to spawn processes. In principle, this is implementation-specific and may be not required if the MPI is tightly integrated with your operating and/or batch system. For MPICH1 and Platform-MPI, you can do this by preparing a file containing hostnames of the nodes (one per line) you will use in your parallel job. Then you set the SCM_MACHINEFILE environment variable pointing to the file.

When you submit a parallel job to a batch system the job scheduler usually provides a list of nodes allocated to the job. The $ADFBIN/start shell script has some logic to extract this information from the batch system and pass it to the MPI's launcher command (typically mpirun). In some cases, depending on your cluster configuration, this logic may fail. If this happens, you should examine the $ADFBIN/start file and edit the relevant portion of it. For example, you may need to change commands that process the batch system-provided nodelist or change mpirun's command-line options or even replace the mpirun command altogether.

4.3 More on Platform MPI 7 (formerly HP-MPI)

HP-MPI has recently been acquired from HP by Platform Computing and has subsequently been rebranded as Platform MPI 7. The use of Platform MPI 7 is governed by our standard End User License Agreement (EULA)

As it has been already mentioned, Platform MPI 7 (formerly HP-MPI) is currently the standard MPI implementation on some systems, namely under Windows and on Intel-based Linux machines.

For more information about Platform MPI 7, including User's Guide please visit the MPI page at Platform Computing.

Full documentation about HP-MPI is still available at the HP website. Platform MPI 7 (formerly HP-MPI) is distributed with (HP-MPI versions of) ADF and a complete HP-MPI directory tree is found in $ADFBIN/hpmpi. Using HP-MPI with ADF does not require any additional fees. In addition to TCP/IP and shared memory supported by every MPI implementation, Platform-MPI also supports, without recompilation, all well-known interconnects such as Infiniband and Myrinet. The best available interconnect is chosen automatically at run time and this can also be modified using mpirun command-line switches in the $ADFBIN/start script.

A few words about the mpirun commands found in the start script. If you look inside the $ADFBIN/start file on Linux, you will likely see three mpirun commands. They differ in the way the list of nodes is specified, if specified at all. The mpirun command with a *-lsb_hosts* switch is used under the LSF batch system. In employs the HP-MPI's tight integration with LSF and lets it pick up the job configuration directly from the batch system.

The mpirun command with a *-hostfile* switch is used for multi-node interactive jobs and for all parallel jobs started under PBS or SGE batch systems. In the case of a batch system, the hostfile provided by the scheduler is passed to the mpirun command as-is, that is without modifications. The latter can be a problem if the format of the hostfile is different from that supported by HP-MPI, in which case it should be modified accordingly. Please see the HP-MPI User's Guide for information about supported hostfile formats.

Finally, the simplest form of the mpirun command is used for single-node jobs when the calculation is performed on the localhost. In this case the NSCM environment variable determines how many parallel processes will be started. Please note that if "$NSCM" is not equal to "1" (digit one) then the exact value of the NSCM variable has no effect when the job is started under any of the batch systems or with a hostfile. In this case the number of parallel processes to start is determined by the contents of the hostfile.

One important point about HP-MPI is that it needs a remote shell command, such as ssh, to start tasks on compute nodes. By default ssh is used but this can be changed using the MPI_REMSH environment variable. For example, executing the following from the bash prompt will make mpirun use rsh instead of ssh:

```
export MPI_REMSH=rsh
```

As usual, if you want to make the change permanent, you need to add this command to a shell resource file.

There are cases when a Linux cluster is configured in such a way that both ssh and rsh communication from/to/between compute nodes is disabled. One of the most common examples is TORQUE with the MAUI scheduler. In this case, there is a remote shell replacement utility called *pbsdsh*. This utility checks that it is executed under PBS and, if yes, allows you to start remote programs only on the nodes allocated to that particular job. In principle, this is all that mpirun needs. The only problem is that *pbsdsh* uses different command-line options. To solve this, we provide in $ADFBIN a script called *torque_ssh*. To make Platform-MPI always use torque_ssh instead of ssh simply set MPI_REMSH in the shell resource file as follows (of course **after** the ADFBIN definition):

```
MPI_REMSH=$ADFBIN/torque_ssh
export MPI_REMSH
```

4.4 Known HP-MPI/Linux issue: libibverbs error message

HP-MPI stops with the following (or similar) error message

```
libibverbs: Fatal: couldn't open sysfs class 'infiniband_verbs'.
```

This occurs, for example, on GigE-connected ROCKS clusters with OpenMPI installed. The error is caused by the fact that there are Infiniband libraries installed without corresponding kernel drivers and/or hardware. In this case, one has to enforce the use of TCP by HP-MPI, which can be done using one of the methods below:

- edit the $ADFBIN/start script and add a -tcp option to all mpirun commands:
  `$ADFBIN/hpmpi/bin/mpirun.mpich -TCP ...`
- or set the MPIRUN_OPTIONS environment variable in your shell resource file (for example, ~/.bash_profile):
  `export MPIRUN_OPTIONS=-TCP`
- or set the MPI_IC_ORDER environment variable in your shell resource file (for example, ~/.bash_profile):
  `export MPI_IC_ORDER="TCP"`

Any of these options will make sure all other interconnects but TCP are ignored.