Nuclear Mathematical and Computational Sciences: A Century in Review, A Century Anew
Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2003)

MCNP 5™ IMPROVEMENTS FOR WINDOWS PCS

Tim Goorley, Forrest Brown, and Lawrence J. Cox
Los Alamos National Laboratory
X-Division, X-5, MS F663
Los Alamos, NM 87545
jgoorley@lanl.gov, fbrown@lanl.gov, ljcox@lanl.gov

ABSTRACT

With the release of MCNP 5, much more emphasis has been placed on improving its functionality on PCs running Microsoft Windows® (9X/NT/2000/XP). Enhanced installation and build systems, support for more Fortran compilers, integration with X11 graphics build files, and MPI and PVM parallel capabilities have been implemented in MCNP 5 for Windows PCs. MCNP 5 can be installed with an InstallShield setup programs, similar to other Windows programs, for users who only need to install executables and data libraries. For those users who need to compile the source, the GNU make utility can used in conjunction with three supported Fortran compilers: Compaq Visual Fortran (CVF), Lahey Fortran 95 (LF95) or Absoft Fortran 95 (AF95). Alternatively, Compaq Developer Studio can be used to compile MCNP 5. The X-Windows plotting capabilities have been improved, and all the appropriate open source X11R6 files for compiling MCNP 5 are bundled with the MCNP 5 source code. X client software is still needed, however, to display geometry, cross-section or tally plots. Parallel capabilities which exist on other platforms have been extended to Windows PCs, allowing users to utilize dual CPU PCs, clusters of homogeneous Windows PCs (preferably with MPI), or heterogeneous clusters (preferably with PVM). Wall-clock runtimes show that MCNP 5 compiled with CVF runs 1.4 times faster than when it is compiled with LF95 or AF95. Wall-clock runtimes also show MCNP 5 with MPI more effectively utilizes a dual-processor Windows 2000 PC than MCNP 5 with PVM.

Key Words: MCNP 5, Microsoft Windows, Parallel, MPI, PVM

1. INTRODUCTION

MCNP Version 5[1,2,3][1], a general purpose radiation transport computer code, has been released to the Radiation Safety Information Computational Center[4]. MCNP is developed and supported by the Eolus team at Los Alamos National Laboratory (LANL) to support Accelerated Strategic Computing Initiative (ASCI) projects. With this release, MCNP has been rewritten into Fortran 90 and several new features have been added. These new features include a new build and installation process, color plotting enhancements, Doppler energy broadening for photons, a neutral particle radiography imaging capability, improved source options, time importance variance reduction, support for parallel communications with MPI, new random number generators, dynamic memory on all platforms, and a new mesh tally capability[2].

Enhancements for Microsoft Windows® PCs have also been included with this release, since desktop PCs increase their power and popularity for scientific computing applications. In addition to the new code features which are available on all platforms, improvements for

[1] MCNP is a trademark of the Regents of the University of California, Los Alamos National Laboratory.

Windows PCs include two new methods for installing and compiling the source, support for three commercial FORTRAN compilers, and easier building of plotting versions with included X11 graphics files. Support for parallel capability using Message Passing Interface MPI[5,6] or Parallel Virtual Machine PVM[7,8] protocols, originally on other platforms[9], has been extended to Windows PCs. These new parallel capabilities allow MCNP to be run on homogeneous and heterogeneous clusters and/or dual processor Microsoft Windows PCs. These PC enhancements are the subject of this paper. The following two sections describe how to install and build MCNP 5 on a PC. The subsequent two sections discuss a comparison of the wall clock runtimes for MCNP 5 on a dual processor desktop and a network of different CPU speed laptops.

2. INSTALLING MCNP 5 ON WINDOWS PCS

There are two different ways to install MCNP 5 on a PC running a Windows operating system (95/98/NT/2000/XP/ME). The simplest method is to use the InstallShield® setup programs, similar to that of other Windows programs. The first setup program copies the MCNP executables, source code, and test problems to a user-selected directory and then sets two environmental variables. The MCNP Visual Editor and MCNP documentation are also installed. The second setup program installs the data libraries, MAKXSF (a cross-section library compression program), the files XSDIR and SPECS, and sets an environmental variable. The user is then asked to logout and login and then run the test problems to verify that MCNP has been installed correctly. The main advantage of using this method is that no compilers are needed and no source code needs to be built. This option meets the needs of most users.

Alternatively the user can copy the MCNP directory tree to the desired location and use the install script to build MCNP 5, MAKXSF, and/or run the test problem suite. The install script can be used to build MCNP only if the appropriate compilers are already installed. The advantage to this method is that the executables can be rebuilt to apply patches or modifications to the source.

After installing the MCNP executable, additional software may need to be installed and appropriate environmental variables may need to be set or changed to take advantage of X11 graphics or parallel communications capabilities in MCNP. X-windows client software, not provided with MCNP 5, needs to be running to display geometry, tally, or cross-section plots. X-Windows software is discussed in section 2.4 of this paper. To use the parallel versions of the MCNP executables, parallel communications software will need to be installed prior to running with this capability, and prior to building a parallel version of MCNP, if the InstallShield executables are not used. Specific instructions on how to install MPI and PVM are given in section 2.5. The environmental variables PATH and DATAPATH can be modified to make file management easier. DISPLAY may need to be set to use the plotting capabilities of MCNP. MCNP environmental variables are discussed in section 2.6.

2.1. The InstallShield Setup Program

The InstallShield programs for MCNP 5 are similar to that of other windows applications. Double clicking on the setup file will start the InstallShield program. After starting the MCNP 5 Executables installer, the initial setup window is displayed, then the next two windows present the Copyright notice and Software License Agreement, and request user information. The following window asks where MCNP should be installed. It does not need to be in the default directory of /Program Files/LANL/MCNP5/. The installation package will then copy the plotting-sequential and parallel executables, MCNP source code, documentation (including the MCNP Manual), Visual Editor, and problem test suite into the chosen directory. The final screen queries the user if it can change the appropriate environmental variables: PATH and DISPLAY. Since PATH is already present, the MCNP5 directory path is appended to this variable. If these environmental variables are changed, the user must log out and then log back in (or reboot for some operating systems) before they will take effect. The user should be aware that if another executable with the name MCNP5 is already present in the path, the first executable in the path, i.e. the previously existing MCNP5, may be unintentionally used. The InstallShield setup program cannot be used to build MCNP executables.

The second InstallShield setup program installs the data libraries, the MAKXSF program and the XSDIR and SPECS files. The installer queries the user for a directory to place these files, which may or may not be a subdirectory where MCNP5 was installed. After this directory is specified, the environmental variable DATAPATH is set. Administrative privileges are also required, as well as write permission to at least 2.5 Gigabytes of hard disk space. This large amount of space is mostly used by the ASCII format (type 1) updated ENDF/B-VI cross-section libraries, which can be compressed with MAKXSF and the SPECS file to ~800 Megabytes. For more information about using MAKXSF, see section XIV of Appendix C in the MCNP 5 Manual.

After the InstallShield programs are completed, the user should run the test suite to verify that the executable has been installed and operates correctly on the user's specific operating system and hardware. This testing procedure can be started by double clicking on the runprob.bat icon located in the directory Installation, where MCNP5 was installed. After the test problems are run, files that list the differences between the tally or output files generated and the expected results are displayed. These difference files should be reviewed by the user to determine if the differences are simple round-off errors or something more substantial, indicating incompatible hardware or software and that MCNP may give incorrect results.

To uninstall either the MCNP 5 Executables or MCNP Data install packages, the user should remove them via the Windows Control Panel, with the Add/Remove Programs function. This will delete any files that were installed (files created while running the test problems will not be deleted) and will modify the registry appropriately. The environmental variables will not be removed, however, but these can be removed manually.

2.2. The Install Script

A second method to install MCNP 5 uses an install script, which interactively queries the user for various build options and then executes the make utility to build MCNP, MAKSXF and/or run

the test problem suite. The various build options include which Fortran and C compilers should be used, the location of appropriate X11 files, and the path to the xsdir file. The script also gives the user the opportunity to only generate custom makefiles. These custom makefiles contain the build options selected, and will be automatically used whenever the make utility is used. An answer file, which lists the options chosen in the install script, is also created. The answer file can only be used to set options in the install script, and only if it is specified on the install command line. If MCNP is to be built, or the test suite is to be run, then install script will then execute *gmake*, (the GNU version of *make*), and write most of the output to the file install.log. This script, which uses the *gmake* utility to build the code, can only be used if a Unix based shell is installed.

2.3. Installing a Unix shell - Cygwin

If the install script or *gmake* utility is used, a Unix command shell must be installed. Unix command shells are not standard on Windows operating systems and must be installed. Cygwin, a freeware port of a Unix command shell for Windows PCs, can be obtained at http://www.cygwin.com or http://www.redhat.com/apps/download/. The setup program can be downloaded or run from the Redhat website. This setup program will step users through the Cygwin installation process, allowing them to select web installation or download installation files to a local drive. The location where the cygwin software should be installed (a path without spaces is recommended) and a temporary directory where files can be downloaded are specified in the next two windows.

After selecting a website to download or install from, the user selects Cygwin packages to install. In addition to the Cygwin packages that are selected by default, the *gcc* and *make* packages (located under the Devel directory) will also need to be selected to build MCNP with the *make* utility. The *perl* package (located under the Interpreters directory) is also recommended, and is required if the Absoft or Lahey Fortran compilers are going to be used to compile MCNP.

2.4. Installing Plotting Software – X Windows Client

In order to display MCNP plots, an X11 windows client software package is needed. Several commercial X-windows clients are available: Reflection X (http://www.wrq.com/products/), Hummingbird's Exceed_NT (http://www.hummingbird.com/products/nc/exceed/index.html), and Starnet's X-win32 (http://www.starnet.com/). No single commercial product is recommended. A freeware X client is also available with Cygwin, X-Free86. It has also been tested with MCNP5. These client software packages do not need to be the developer or professional versions, since the X11 header and library files are included with the MCNP 5 distribution.

2.5. Installing Parallel Software – MPICH.NT or PVM

In order to use or build MCNP 5 with parallel capabilities, appropriate parallel communications software must be installed. Either MPI or PVM communications protocols are supported. To build a parallel version of MCNP 5 for Windows, see the section 3.6.

The MPI port for Windows is MPICH.NT, developed at Argonne National Laboratory, and can be downloaded from http://www-unix.mcs.anl.gov/~ashton/mpich.nt/. This website also offers the helpful references MPICH.NT FAQ and MPICH Users Manual. If there is no need to rebuild parallel MCNP 5, only the runtime dlls and mpirun package will be needed (mpich.nt.1.2.4.exe). This package uses an InstallShield setup program which requires installation from an administrative account on all PCs in the cluster. If MCNP 5 needs to be built, then the source code (package mpich.nt.1.2.4.src.exe) should be downloaded and unzipped as well. The program MPIConfig must be run on each computer after MPICH.NT installation. The local host name must be added and the settings applied.

Once MPICH.NT is installed, a few additional steps are required. MPI enabled MCNP must be copied to the same directory on all hosts. MCNP can be executed through either the Windows MPIRun GUI or command line MPIRun. For the MPIRun GUI, hosts must be added by selecting or typing their names in the hosts section. The DATAPATH may need to be set under the advanced options. For the command line MPIRun, the hosts must be specified by the –hosts option, which can be used to specify the number of processes started on each host. Typically the number of processes is equal to the number of CPUs utilized plus one. The first process listed is the master process, and does not run any histories. For example, the command to start three MCNP MPI processes on ComputerA (a dual CPU machine) and one process on ComputerB is:

```
mpirun –hosts 2 ComputerA 3 ComputerB 1 mcnp5mpi inp=test
```

Alternatively, MCNP 5 can be built and run with PVM, developed at Oak Ridge National Laboratory. The PVM port for Windows can be downloaded from http://www.csm.ornl.gov/~sscott/PVM/Software/. The file ParallelVistualMachine3.4.3.zip contains the source and binaries needed to install and run PVM on a single computer. This InstallShield program must be run on all Windows PCs in the cluster. PVM requires additional communications software, a remote shell (RSH) client/server package, before it will run across a cluster of Windows PCs. Two commercial RSH packages can be obtained from http://www.winrshd.com/ and http://www.ataman.com/. The RSH package must be installed on each computer in the cluster, and the permissions must be set to allow RSH or REXEC connections for the desired user accounts. The Ataman RSH package was successfully tested.

Similar post-installation steps are required to run the PVM version of MCNP 5 on Windows PCs. The MCNP executable must be copied into the %PVM_ROOT%/pvm3/bin/WIN32 directory on all hosts. PVM must be started on a single computer before a PVM enabled MCNP can be executed. After PVM is started, additional hosts can be added to the PVM cluster, with the "add host" command from the PVM console prompt. PVM operability can be tested and verified with the PVM example programs, such as hello. MCNP can then be started from a separate command shell with the following command:

```
mcnp5pvm inp=test tasks n
```

where `n` is the number of slave processes. The number of tasks is usually the number of CPUs in the cluster. A negative number entered for `n` causes MCNP to skip the initial load-balancing feature, and is recommended for a homogeneous cluster. Additional information on how to

install MPI or PVM can be obtained in their respective user's manuals. Additional information on running MCNP in parallel can be found in Appendix C, section VIII of the MCNP 5 Manual.

2.6. Setting Environmental Variables

After installation, it may be necessary to change or add three Windows environmental variables. They are set differently for Windows 95/NT/2000/XP/ME, but for each of these operating systems the variables can be viewed the same way. The value of an environmental variable PATH, for example, can be printed in a command shell window (i.e. a "DOS prompt") with the command `echo %PATH%`.

The first environmental variable that may need to be changed is PATH, a semicolon separated list of directories used to find executables and dynamic link libraries. The directory where MCNP is installed should be included, so that MCNP can be executed from any directory, making file management more convenient. If several programs with the same name exist within PATH, then the executable in the first occurring directory will be executed. Appending the directory of the newly installed version of MCNP to the PATH may not change which MCNP is executed if an older version with the same name is given earlier in the directory listing. Failure to change PATH will mean that the MCNP executable must be located in the directory where the input file is located. To use the plotting features of MCNP, the location of the dynamic link library X11.dll or Xlib.dll may need to be added to PATH as well. Failure to make sure that X11.dll or Xlib.dll is in your path may mean that a plotting-enabled executable cannot run, even if the plotting features of MCNP are not used.

The second environmental variable that may need to be added or changed is DATAPATH. It is the directory path to the file XSDIR, which is used by MCNP to locate the cross-section data libraries. MCNP also searches in the local directory and a directory specified at compile time. If XSDIR does not exist in any of these three locations, MCNP will issue a fatal error. Additionally, the command line option `XSDIR=name` may be used to specify an XSDIR formatted file with a different name located in the directory given in DATAPATH. Failure to set DATAPATH means that the file XSDIR must be located in the directory where the input deck is located.

The third environmental variable is DISPLAY, and is only needed for a plotting version of MCNP. This environmental variable is used by the X windows client to route X windows. It is usually set to display windows on the same computer where MCNP is executed. In this case, the value of DISPLAY should be localhost:0. Failure to set DISPLAY may mean that the plotting executable will not be able to open a window and plot. A graphics version of MCNP will still be able to create a postscript file of the plots, even without the X-windows client software or the environmental variable DISPLAY.

3. BUILDING MCNP ON WINDOWS PCS

MCNP 5 must be built if the install script is used or the source needs to be modified or patched. The install script uses the make utility to direct one of three supported Fortran 90/95 compilers, and one of three supported C compilers, to compile and link MCNP. Alternatively, the

interactive graphical interface, Compaq Developer Studio, uses Compaq Fortran 90 and Microsoft C to build MCNP. Neither the *make* utility nor a Unix shell is needed to build MCNP with Compaq Developer Studio. These two methods will be discussed in the following paragraphs.

3.1. The *make* Utility

Make is a utility which understands user defined relationships between different files which are used to build programs. In an attempt to supervise the building of a target program, *make* controls preprocessors, compiler, linkers, archive utilities, or other programs to manipulate these files. MCNP 5 can be built on all supported platforms, including Windows PCs, with the *make* utility.

Unlike most Unix or Linux based operating systems, the *make* utility is not standard on Windows. The recommended version of make for a Windows PC is the GNU *make* utility, which is an optional addition with Cygwin, described above. The *make* utility does not include any compilers, which also must be installed to build MCNP. The *make* utility can be used to build MCNP by typing "make build" in a Cygwin command prompt in the MCNP 5 /Source directory.

One of the first things make does is read an operating-system dependent file in the MCNP 5 /Source/config directory. For all Windows installations, this file is Windows_NT.gcf. It contains all the default paths to the supported compilers, header files, and libraries, and specifies the compiler and linker options. This file should be modified if the compilers and other necessary files are not in the default locations. The make utility will also read the custom_Windows_NT.gcf file created by the install script if present. Additional information about these files and make commands is given in Appendix C, section II of the MCNP5 Manual.

3.2. Three Fortran Compilers

The *make* utility uses one of three Fortran 90/95 compilers and one of three C compilers to build MCNP 5 on Windows PCs. The supported Fortran 90 compilers are: Compaq Visual Fortran (CVF version 6.6B) [formerly known as Digital Visual Fortran], Lahey Fortran 95 Pro (LF95 version 5.70c), and Absoft Pro Fortran (F95 version 8.0). The supported C compilers are the Microsoft C/C++ compiler (MSC version 12.00.8168) and GNU gcc compiler (version 2.95.2-5 [Cygwin special]), either of which may be used with any of the three Fortran compilers. The Fujistu C compiler (FCC version 3.0) is also supported, but only in conjunction with Lahey Fortran 95. Table I shows the versions of MCNP which can be compiled with the *make* utility. For example, the command "make build CONFIG='plot cheap compaq cl' " at the Cygwin command prompt in the MCNP 5 /Source directory will build a plotting version of MCNP 5 with the Compaq Fortran and Microsoft C compilers.

Table I. Supported versions and compilers with the Cygwin make utility

MCNP Version	Supported Compiler Sets*	Make Command Line	Notes
Sequential	Compaq (v6.6B) Lahey (v5.70c) Absoft (v8.0)	CONFIG='seq compaq' CONFIG='seq lahey' CONFIG='seq absoft'	GNU gcc is used*** Absoft & Lahey need preprocessor. Absoft does not support control-c interrupts.
Plotting	Compaq + MSC Compaq + gcc** Lahey + MSC Lahey + gcc Lahey + fcc Absoft + MSC Absoft + gcc	CONFIG='plot compaq cl' CONFIG='plot compaq gcc' CONFIG='plot lahey cl' CONFIG='plot lahey gcc' CONFIG='plot lahey fcc' CONFIG='plot absoft cl' CONFIG='plot absoft gcc'	X11 library and headers required to compile. X client running required to display. Absoft & Lahey need preprocessor. Absoft does not support control-c interrupts.

*Only the professional version of the three Fortran compilers is supported.
**The default. "`make build`" will create a sequential plotting executable with CVF and gcc.
***The *make* utility expects there to be an object file from the c source, so gcc is used to build a nearly empty object file which the Fortran compilers link.

All of the compilers use the environmental variables LIB and INCLUDE to locate the compiler's libraries and include files, respectively. These should be set when the compilers were installed. Additionally, the directory path to the compiler executables (f90, lf95, f95) should also be located in the PATH environmental variable. If more than one compiler with the same name is installed, the explicit path and compiler can be specified in the Windows_NT.gcf file.

3.3. A *Perl* Preprocessor

Unlike the Compaq Fortran compilers, the Absoft and Lahey Fortran compilers do not have the capability to preprocess #ifdef statements that are used in the MCNP Fortran and C source to specify version (sequential, plotting, etc) and platform (Unix, Linux, Dec, etc.) specific code. The MCNP Fortran source must be preprocessed before the Absoft and Lahey compilers can be used. This can be done with the *perl* script fpp.pl, which is provided with the MCNP distribution. This script processes out the #ifdef statements and substitutes the appropriate values for the MCNP version, compile date, and version number. If the environmental variable DATAPATH is set, it will also be substituted directly into the source code before MCNP is compiled. The processed files are saved with .F95 extensions for Absoft or .i extensions for Lahey, which are then compiled. The intermediate files are deleted after linking.

3.4. Compaq Developer Studio

As an alternative to the *make* utility, the Compaq Developer Studio build utility is also supported. It is an independent GUI which requires neither Cygwin nor the *make* utility. Using the Developer Studio, it is possible to build a sequential, plotting, MPI or PVM version of MCNP 5. Included in the MCNP 5 distribution are Developer Studio projects and project workspaces (.dsp and .dsw files, respectively) for each of these versions. To build one of these four versions, open the corresponding .dsw file located in the /Source/CVF directory. If CVF is installed on the computer, this can be done simply by double clicking on the appropriate file in the Windows Explorer.

The only settings that may need to be changed are the paths to the X11, MPICH.NT or PVM files, if they were not installed in the default directory. The paths to these files are specified in the C preprocessor and link input fields of the project settings. Unlike make, the CVF Developer Studio has no problems when paths to these files contain spaces. An additional path to the file XSDIR can also be specified at compile time. The Fortran preprocessor keyword DPATH and its value can be set on the project settings' Fortran tab. While the directory path must not contain spaces, the DOS style format will work. For example, instead of C:\Program Files\LANL\MCNP5\data, C:\Progra~1\LANL\MCNP5\data should be specified.

The option to build is the second item under the build menu on the topmost menu bar. Alternatively, the F7 button can be pressed to start the build. In some cases Developer Studio cannot determine which files need to be built prior to others, and the code may need to be rebuilt after the first failed build completes. Additionally, the user is encouraged to build the release version of MCNP by setting the active configuration (under the build menu item). The versions of MCNP 5 which have been built and tested with the Compaq Developer Studio are shown in Table II.

Table II. Supported versions with Compaq Developer Studio

MCNP Version	Supported Compiler Sets	Notes
Sequential	Compaq	No C compiler is needed.
Plotting	Compaq + MSC	X11 library and headers required to compile. X client required to display.
Parallel (MPI or PVM)	Compaq + MSC	PVM or MPICH.NT also required.

3.5. Building Plotting Versions

To display MCNP geometry, tally, or cross-section plots, it is necessary to have a plotting enabled version of MCNP and X window client software. A plotting version of MCNP can be built with either the *make* utility or Developer Studio. With *make*, a plotting executable is the default. To force *make* to build a plotting enabled version of MCNP with *make*, it is necessary to specify "plot" on the CONFIG keyword.

One of the enhancements for Windows PCs is that all of the appropriate source files (including the X11.lib, X11.dll, and header files) are included in the MCNP distribution and are located in the Source/X11r6 directory. The original source can be downloaded from http://www.X.org. The X11 library may be downloaded from ftp://ftp.cc.utexas.edu/microlib/nt/x11r6/ or built from the X.org source. Proprietary X11 header files and library files may also be included with a commercial X windows client (if the developer / professional version was purchased), but using the X11 files with the MCNP distribution is recommended.

3.6. Building Parallel Versions

To use the parallel features of MCNP, it is necessary to have a parallel enabled version of MCNP and the appropriate parallel communications software installed and running. Parallel enabled versions of MCNP can only be built with Compaq Developer Studio after the header files and appropriate libraries have been installed.

The Developer Studio project files mcnp5mpi.dsw and mcnp5pvm.dsw already have all the appropriate settings configured. These changes include the addition of the preprocessor definition MULTP (and MPI for the MPI version), additional Fortran and c source, and MCNP include files, and the additional paths to these directories and the MPI or PVM directories. The only changes needed are the paths to the MPI or PVM include files and libraries if these are not installed in the default directories. The include file path will need to be listed in the C++ tab (preprocessor category). The MPI or PVM library should be listed in the Object/library modules line on the link tab (the general category). The necessary libraries for MPI are the ws2_32.lib and mpich.lib libraries. The PVM libraries are ws2_32.lib, libpvm3.lib and libgpvm3.lib. The path to this library should be listed in the input category.

4. RESULTS

MCNP5 was installed and built on several Windows 2000 PCs with each of the methods discussed above. The sequential executables were subsequently tested by running the accompanying suite of test problems. Both the runprob.bat script (for the InstallShield installation) and the *make* utility's "make test" command (for the Cygwin & *make* installation) were used to run the test suite. All three compilers produced identical tally results to what was expected. The PVM and MPI versions were also tested with the make utility by specifying PRUN="mpirun –np 3" and TASKS="tasks 3" keywords respectively. Neither the PVM nor the MPI version of MCNP5 produced any tallies different than expected. They both had four files for which the frequencies of the tfc updates were different than the sequential version, but this is expected.

A realistic problem, well suited for timing studies, was chosen to illustrate differences in wall clock runtimes from the three Fortran compilers and the two parallel versions of MCNP 5. The model, a voxelized human head with an incident photon beam, was used for treatment planning dosimetry calculations[10]. Additional patches to MCNP will greatly speed these treatment planning calculations, but are not needed for these illustrative purposes. The wall clock runtimes for this problem are shown in Tables III-V. Table III shows the wall clock run-time differences between the Fortran compilers for runs on a Dell Precision 520 (Dual 2.0 GHz Pentium Xeon® Processors, 768 Megabytes RAM, 512 kbytes L2 cache, 100 MHz bus) running Windows 2000. These runs only utilize a single processor on this PC.

Table III. Sequential MCNP 5 wall-clock runtimes.

Wall Clock Runtimes (min:sec)	**Sequential**		
Compiler	Absoft	Lahey	Compaq
Optimization	-o1	-o1	/optimize:5
Nps 10,000	10:29	11:12	7:29
Nps 100,000	100:57	107:55	72:36

MCNP 5 from the Compaq compiler was a factor of 1.39 and 1.48 times faster than the Absoft and Lahey compilers, respectively, for the 100,000 history run. All three of these optimization levels include local optimization, and the CVF and Lahey compilers include global optimization and inlining of small procedures. Since MCNP subroutines are compiled individually, as separate files, not all global optimizations are possible.

This problem was also run in parallel to illustrate the advantages of multiprocessing. The problem was executed separately with MPI and PVM on a homogeneous cluster (a dual processor PC) and a heterogeneous cluster (a stand-alone cluster of laptops). Both methods of using PVM (tasks n and tasks -n) were employed, as well as manually changing the slave process priority with the Windows Task Manager. All parallel executables were compiled with the CVF compiler. Table IV shows the wall clock runtimes on the Dell Precision 520 running Windows 2000.

Table IV. Parallel MCNP 5 wall-clock runtimes on a dual processor PC.

Wall Clock Runtimes (min:sec)	Sequential	PVM tasks 2	PVM tasks -2	PVM* tasks 2	PVM tasks 3	MPI 3 processes
Nps 10,000	7:36	9:42	5:38	4:47	7:15	4:18
Nps 100,000	72:34	90:55	41:24	40:37	54:13	38:44

*The slave PVM process priority was set in the Windows Task Manager to "above normal", immediately after the slave process was spawned.

There was considerable reduction in wall clock runtimes of the MPI run and certain PVM runs. The sequential job was 1.87 times slower than the MPI job, which was faster than any of the PVM jobs. Differences in the PVM runs result from how the master and slave processes are treated by the CPUs. One of the possible ways to run on a dual processor, arguably the most intuitive way, tasks 2, was slower than the sequential run. Two of the PVM jobs, where the spawned tasks skipped the load-balancing test or had their priority changed with the task manager, were nearly as fast as the MPI run. The reason for these runtime differences is addressed in the discussion.

To test the MPI and PVM versions of MCNP5 on a heterogeneous cluster of PCs, a small cluster of two laptops running Windows 2000 was assembled. The two laptops were quite different in hardware specifications: DELL Inspiron 8200 – Pentium IV®, 1.6 GHz, 1024 Mbytes RAM, 512kbytes L2 Cache, and a DELL Latitude C840 - Pentium III®, 1.0 GHz, 512 Mbytes RAM, 256 kbytes L2 Cache). They were connected through a 3COM® 100 Mbyte Office Connect switch and form a stand-alone cluster. The wall-clock runtimes for the treatment planning problem when run over this cluster with PVM and MPI are given in Table V.

Table V. Windows PC MCNP 5 wall-clock runtimes on a small laptop cluster

Wall Clock Runtimes (min:sec)	Sequential		PVM tasks 2	PVM* tasks 2	MPI 3 processes
Task Distribution	Pentium 4	Pentium 3	P4: Master + Slave P3: Slave	P4: Master + Slave P3: Slave	P4: Master + Slave P3: Slave
Nps 10,000	9:41	30:25	11:41	10:05	16:33
Nps 100,000	90:55	298:54	143:32	83:27	153:29

*The slave PVM process priority was set in the Windows Task Manager to "above normal", immediately after the slave process was spawned.

These wall clock runtimes show that the PVM version of MCNP is much better at utilizing a cluster of heterogeneous CPUs than the MPI version. They also show that for short jobs, there may not be an advantage to running in parallel on a cluster, with either MPI or PVM. When the master and slave tasks are each located on a single processor, both of them default to equal priority and compete for CPU time. This causes the slave task to be slowed down considerably. Increasing the slave task's priority in the Windows Task Manager causes the behavior of PVM's master task to more closely resemble MPI's process zero task; i.e. the master used much less CPU time while it is waiting for the slaves to finish, but the PVM version also utilizes an initial load-balancing test which allows it to more effectively utilize the heterogeneous cluster. This change in the slave process priority must be made before the initial load-balancing test is performed. In fact, this PVM configuration was the only MCNP 5 parallel job to be completed in less time than the sequential run. The run is only slightly faster than the sequential run, but this is expected since the P3 processor is less than a third as fast as the P4 processor.

5. DISCUSSION

5.1. Fortran Compilers

While Compaq F90 produces MCNP executables which are faster than those compiled with Lahey and Absoft, speed is not the only consideration for all MCNP users. The Compaq GUI debugger is quite powerful and easy to use. The Lahey compiler is very strict, providing warnings about improper Fortran usage that other compilers do not. The Absoft coverage tool is quite useful to determine which routines are using most of the processor power. These features are quite advantageous to MCNP developers. No single Fortran compiler is recommended.

5.2. MPI vs. PVM

While other calculations will show different relationships between the wall clock runtimes, depending on the number of particles run, the hardware configuration, the amount of communication between processes, and the type of problem (shielding vs. criticality), there are some underlying concepts and functions, illustrated by the test problem for the two presented cases, which are applicable to many more MCNP parallel calculations running on Windows PCs.

Both parallel versions of MCNP use the same master/slave task model. The user initiated PVM MCNP process spawns tasks to perform particle tracking, while the master collects their results at intervals designated by the fifth entry of the prdmp card. The MPI process zero performs the same function as the PVM master process. Thus an MPI job with 4 processes has the same number of processes doing transport as a PVM job with 3 tasks. This difference should be considered when matching processes to the number of available hosts and processors.

The MPI and PVM versions also print "the number of histories processed by each task" and the "estimated system efficiency" with other information in the problem summary section of the output file. If the problem ran as expected, the first task, the PVM master task or MPI process zero, will not have processed any histories. If one of the PVM tasks dies, the master will attempt to recover and run the slave tasks's remaining histories. If successful, the first entry will be non-zero. A warning will also be given that a remote process terminated prematurely.

While basic operation of the two parallel versions of MCNP is the same, there are considerable differences between the Windows PC MPI and PVM versions of MCNP 5. These differences are consequences of the communications protocols themselves, how MCNP uses each of them, and how the Windows operating system interacts with them. These differences will be discussed in three categories: error handling, compatibility and execution.

The PVM version is much better suited than the MPI version to handle potential errors and interrupts in the execution of MCNP. If a single PVM process dies (such as a single machine power failure), as long as that process is not the MCNP master process, the master will recognize that the slave process has died and will attempt to run histories assigned to the slave. If any MPI process dies, then the entire job will stop running. If the entire job dies for either version, the MCNP 5 runtape can be used to recover the calculation from the last dump. To restart, the number of processes (but not the original locations of the processes) must be the same. User initiated interrupts (control-c events) are handled correctly by PVM, bringing up the MCNP status menu. MPICH.NT immediately kills all processes when a control-c event occurs. MCNP 5 contains a much more elaborate error messaging system than MCNP4C to inform the user what kind of communications error has occurred.

The PVM version is also much more compatible with other PCs than MPICH.NT. PVM can be used to build a cluster of not only mixed capacity Windows machines, but also computers running Linux, Unix, and other operating systems. MPICH.NT currently only supports clusters of PCs with Windows NT/2000/XP operating systems[11]. While MPICH.NT will run on individual Win9X PCs, they cannot be linked together to build clusters.

The MPI version, however, is clearly much more efficient when running on a homogeneous cluster or dual processor machine, as evidenced by the runtimes in Table IV. The PVM version of MCNP does initial load balancing, by timing how long it takes spawned processes to track 200 histories, to determine how many particles to run on each host between rendezvous. MCNP does not correct for a change in processor usage after this test. This test is optional and can be avoided by using a negative entry with the tasks keyword. When this negative tasks option is used, the PVM version behaves much more like the MPI version, and the runtimes on a homogeneous system are closer to the MPI version. The MPI version of MCNP does not do load balancing because it assumes that the cluster is homogeneous. During each rendezvous, all tasks must report before any task begins the next set of assigned particle histories. Thus a cluster of mixed-processor PCs running MCNP with MPI or PVM with a negative tasks option, would be hindered by the slowest processor since all slave tasks are idle while the slowest processor finishes its assigned particles.

Greatly affecting their efficiency on Windows PCs, the root processes also behave differently for the two protocols. During a PVM MCNP run, the PVM master process utilizes a significant amount of CPU time waiting for the slaves to rendezvous, (i.e. listening for the slave processes to report). For clusters with few CPUs, the additional computational expense of the PVM master process is significant. The MPI process zero and PVM master (when the negative tasks entry is used) only uses CPU time when it is collecting information during rendezvous. Alternatively, the PVM master process uses significantly less CPU time if the slave tasks are given a higher

priority, which can be accomplished by selecting the slave task from the Windows Task Manager. Using the positive entry on the tasks keyword and re-prioritizing the slave task should be the most efficient method of running parallel MCNP over a heterogeneous cluster of Windows PCs. Since the reprioritizing of the slave task competing with the master only increases the performance of one CPU, this tactic would be less advantageous on a large cluster.

Additional problems arise when using a dual processor PC. Due to the way that the Windows operating system assigns processes to processors, both of the slave tasks were assigned to the same processor, while the master utilized its own processor, creating very inefficient load balancing. When three slave tasks were created, the third process was assigned to the same processor as the master, allowing the job to run more net particles per unit time, increasing efficiency. This is why the runtimes for the PVM "tasks 3" job are shorter than the PVM "tasks 2" job, even though the "tasks 3" job is oversubscribed. While even more processes could possibly run more net particles per unit time, an increasing overhead of handling more processes would diminish returns. Thus the PVM version of MCNP will not be as efficient as the MPI version of MCNP on a dual processor machine, and possibly on a homogeneous cluster.

6. CONCLUSIONS

MCNP version 5 is a much more modernized MCNP which contains significant new features in categories ranging from physics to variance reduction to installation and building. Additional effort has been made to ensure the full capabilities of MCNP can be utilized on Microsoft Windows operating systems. These improvements include a new install and build system, convenient inclusion of X11 graphics files, and the extension of MPI and PVM support for parallel calculations on dual processor or clusters of Windows PCs. With these enhancements users should find MCNP 5 more capable of utilizing increasingly powerful desktop and laptop Windows PCs.

The parallel capabilities of MCNP 5 were tested on a dual processor Windows PC and two networked laptops, representing a homogeneous and heterogeneous cluster respectively. For parallel runs on the dual processor PC, the MPI version was slightly to significantly faster than the PVM versions, depending on the PVM options used. The sequential job was 1.87 times slower than the MPI job, which used two CPUs. For a heterogenous cluster of two networked laptops, the PVM job with load balancing (a positive tasks entry) and increased slave task priority was significantly faster than the other PVM configurations and the MPI job.

ACKNOWLEDGMENTS

This work was funded by the Los Alamos National Laboratory, which is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

REFERENCES

1. J.F.Briesmeister, Editor, *MCNP-A General Monte Carlo N-Particle Transport Code – Version 4C*, Los Alamos National Laboratory report LA-13709-M (March 2000).

2. L. Cox, et. al, "MCNP Version 5", *Program and Abstracts of the 12th Biennial RPSD Topical Meeting*, Santa Fe, New Mexico, April (2002).
3. "MCNP Homepage," http://laws.lanl.gov/x5/MCNP/index.html (2002).
4. "Radiation Safety Information Computational Center Homepage," http://www-rsicc.ornl.gov/rsic.html (2002).
5. Message Passing Interface Forum. MPI: A Message-Passing Interface standard. International Journal of Supercomputer Applications, Vol. 8(3/4), pp. 165-414 (1994).
6. H. Pedroso, J.G. Silva, "The WMPI library evolution: experience with MPI development for Windows environments," *Proceedings of the European Conference on Parallel Computing*, Munich, Germany, 29 Aug.-1 Sept., pp. 1157-64 (2000).
7. A. Geist, et. al, *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Network Parallel Computing*, MIT Press, Cambridge, Mass. (1994).
8. S.L. Scott, M. Fischer, and A. Geist. "PVM on windows and NT clusters," *Lecture Notes in Computer Science* **1497** pp. 231-38 (1998).
9. G.W. McKinney, "A practical guide to using MCNP with PVM," *Transactions of the American Nuclear Society*, **71**, pp.397-8 (1994).
10. R. Zamenhof, et. al, "Monte Carlo-based treatment planning for boron neutron capture therapy using custom designed models automatically generated from CT data," *International Journal Radiation Oncology Biology and Physics*, **35**, pp. 383-397 (1996).
11. "MPICH.NT FAQ," http://www-unix.mcs.anl.gov/~ashton/mpich.nt/mpich.nt.faq.html (2002).