SAND REPORT

SAND2003-2952
Unlimited Release
Printed August 2003

Trilinos Users Guide

Michael A. Heroux and James M. Willenbring

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1110

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/ordering.htm

SAND2003-2952
Unlimited Release
Printed August 2003

Trilinos Users Guide

Michael A. Heroux and James M. Willenbring
Computational Math and Algorithms Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1110

Abstract

The Trilinos Project is an effort to facilitate the design, development, integration and ongoing support of mathematical software libraries. A new software capability is introduced into Trilinos as a *package*. A Trilinos package is an integral unit usually developed by a small team of experts in a particular algorithms area such as algebraic preconditioners, nonlinear solvers, etc.

The Trilinos Users Guide is a resource for new and existing Trilinos users. Topics covered include how to configure and build Trilinos, what is required to integrate an existing package into Trilinos and examples of how those requirements can be met, as well as what tools and services are available to Trilinos packages. Also discussed are some common practices that are followed by many Trilinos package developers. Finally, a snapshot of current Trilinos packages and their interoperability status is provided, along with a list of supported computer platforms.

Acknowledgments

The authors would like to acknowledge the support of the ASCI and LDRD programs that funded development of Trilinos and recognize all Trilinos contributors: Teri Barth, Ross Bartlett, David Day, Bob Heaphy, Robert Hoekstra, Jonathan Hu, Tammy Kolda, Richard Lehoucq, Kevin Long, Eric Phipps, Roger Pawlowski, Andrew Rothfuss, Andrew Salinger, Paul Sery, Ken Stanley, Heidi Thornquist, Ray Tuminaro and Alan Williams.

Contents

Figures

Tables

Trilinos	The name of the project. Also a Greek term which, loosely translated means "a string of pearls," meant to evoke an image that each Trilinos package is a pearl in its own right, but is even more valuable when combined with other packages.
Package	A self-contained collection of software in Trilinos focused on one primary class of numerical methods. Also a fundamental, integral unit in the Trilinos framework.
new_package	A sample Trilinos package containing all of the infrastructure to install a new package into the Trilinos framework. Contains the basic directory structure, a collection of sample configuration and build files and a sample "Hello World" package. Also a website.
Anasazi	An extensible and interoperable framework for large-scale eigenvalue algorithms.The motivation for this framework is to provide a generic interface to a collection of algorithms for solving large-scale eigenvalue problems.
AztecOO	Linear solver package based on preconditioned Krylov methods. A follow-on to the Aztec solver package [7]. Supports all Aztec interfaces and functionality, but also provides significant new functionality.
Belos	A Greek term meaning "arrow." Belos is the next generation of iterative solvers. Belos solvers are written using "generic" programming techniques. In other words, Belos is written using TSF abstract interfaces and therefore has no explicit dependence on any concrete linear algebra library. Instead, Belos solvers can be used with any concrete linear algebra library that implements the TSF abstract interfaces.
Ifpack	Object-oriented algebraic preconditioner, compatible with Epetra and AztecOO. Supports construction and use of parallel distributed memory preconditioners such as overlapping Schwarz domain decomposition, Jacobi scaling and local Gauss-Seidel relaxations.
Komplex	Complex linear equation solver using equivalent real formulations [3], built on top of Epetra and AztecOO.
LOCA	Library of continuation algorithms. A package of scalable stability analysis algorithms (available as part of the NOX nonlinear solver package). When integrated into an application code, LOCA enables the tracking of solution branches as a function of system parameters and the direct tracking of bifurcation points.
Meros	Segregated preconditioning package. Provides scalable block preconditioning for problems that couple simultaneous solution variables such as Navier-Stokes problems.

ML Algebraic multi-level preconditioner package. Provides scalable preconditioning capabilities for a variety of problem classes.

NOX A collection of nonlinear solvers, designed to be easily integrated into an application and used with many different linear solvers.

Petra A Greek term meaning "foundation." Trilinos has three Petra libraries: Epetra, Tpetra and Jpetra that provide basic classes for constructing and manipulating matrix, graph and vector objects. Epetra is the current production version that is split into two packages, one core and one extensions.

- Epetra Current C++ production implementation of the Petra Object Model. The "E" in Epetra stands for "essential" implying that this version provides the most important capabilities that are commonly needed by our target application base. Epetra supports real, double-precision floating point data only (no single-precision or complex). Epetra avoids explicit use of some of the more advanced features of C++, including templates and the Standard Template Library, that can be impediments to portability.
- Tpetra The future C++ version of Petra, using templates and other more advanced features of C++. Tpetra supports arbitrary scalar and ordinal types, and makes extensive use of advanced C++ features.
- Jpetra A Java implementation of Petra, supporting real, double-precision data. Written in pure Java, it is designed to be byte-code portable and can be executed across multiple compute nodes.

Teuchos A collection of classes and service software that is useful to almost all Trilinos packages. Includes reference-counted pointers, parameter lists, templated interfaces to BLAS, LAPACK and traits for templates.

TSF A collection of abstract interfaces that supports application access to a variety of Trilinos capabilities, supports interoperability betweeen Trilinos packages and provides future extensibility. TSF is composed of several packages. The primary user packages are:

- TSFCore TSFCore provides a basic collection of abstract interfaces to vectors, linear operators, solvers, etc. These interfaces provide a common interface for applications to access one or more packages that implement the abstract interface. These interfaces can also be used by other packages in Trilinos to accomplish the same purpose.
- TSFExtended TSFExtended builds on top of TSFCore, providing implicit aggregation capabilities and overloaded operators.

1 Introduction

The Trilinos Project is an effort to facilitate the design, development, integration and ongoing support of mathematical software libraries. Trilinos also provides a set of core utility libraries that provide common vector, graph and matrix capabilities, as well as a common abstract interface for applications to access any appropriate Trilinos package.

The overall objective of Trilinos is to promote rapid development and deployment of high-quality, state-of-the-art mathematical software in an environment that supports interoperability of packages while preserving package independence.

The Trilinos Users Guide is meant to assist new and existing Trilinos users. Topics covered include how to configure, build, and install Trilinos, as well as how to run tests to insure proper installation. In addition, issue reporting and how to sign up for and use Trilinos mail lists are discussed. Finally, directions for obtaining Trilinos itself and documentation for individual Trilinos packages are provided.

For a higher-level view of the Trilinos project, please see An Overview of Trilinos [4]. A document is also available specifically for Trilinos developers; please see the Trilinos Developers Guide [5]. The Developers Guide contains both tutorial and general reference material.

The current set of packages in Trilinos is shown in Figure 1.

1.1 Typographical Conventions

Our typographical conventions are found in Table 1.

Notation	Example	Description
`Verbatim text`	`../configure --enable-mpi`	Commands, directory and file name examples, and other text associated with text displayed in a computer terminal window.
`CAPITALIZED_TEXT`	`CXXFLAGS`	Environment variables used to configure how tools such as compilers behave.
`<text in angle brackets>`	`../configure <user parameters>`	Optional parameters.

Table 1. Typographical Conventions for This Document.

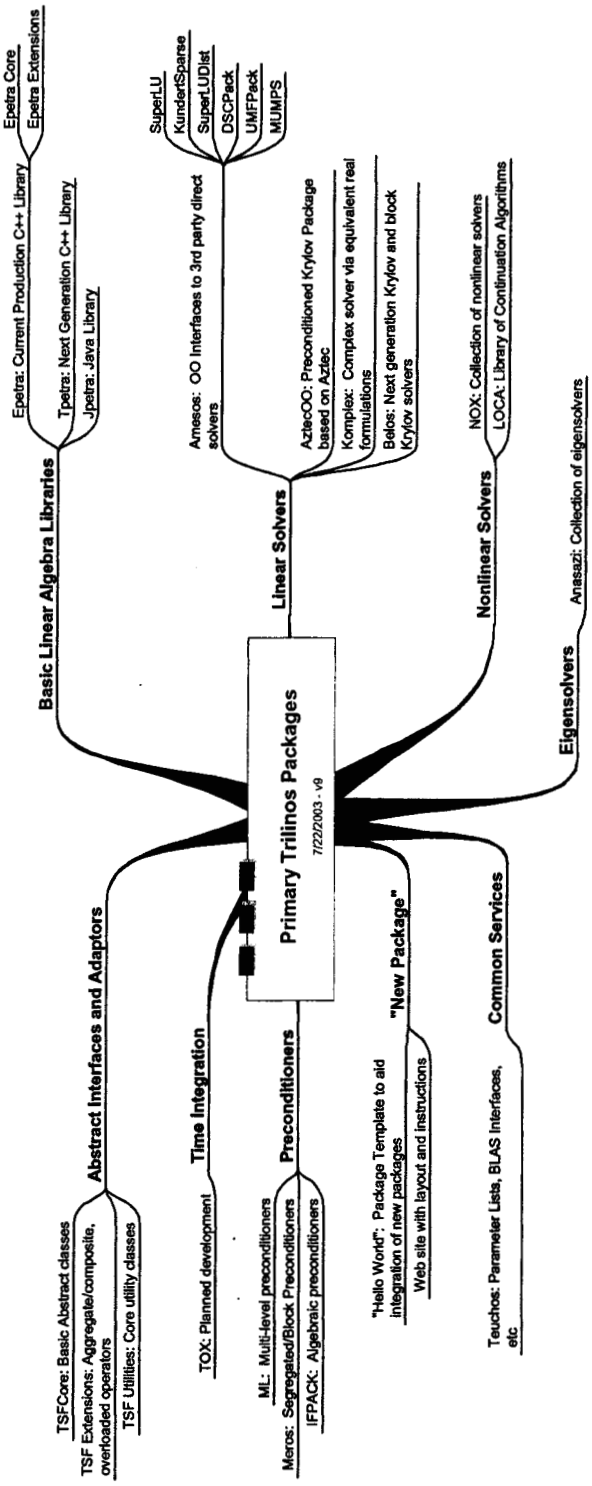

Figure 1. Current collection of Trilinos Packages

2 Getting Started

This chapter covers some of the basics that a user will need to know when beginning to use Trilinos. Specifically, we address how to obtain, configure, build and test Trilinos. Although this user guide is current at the time of printing, the Trilinos Project is under active development. For the most up-to-date information, please visit the online Trilinos Home Page. The most accurate and complete information will be kept at this site.

Tip: Check out the Trilinos Home Page at `http://software.sandia.gov/Trilinos`.
At the website a user can:

- Find up-to-date information about Trilinos and its packages.
- Download Trilinos.
- Sign up for mail lists.
- Find documentation.
- Submit a bug report.

2.1 Obtaining a Copy of Trilinos

Trilinos can be obtained in two different ways. Users most commonly download Trilinos in the form of a tarball from the Trilinos website at `http://software.sandia.gov/trilinos/downloads.html` .

Users may be able to obtain a copy of Trilinos via the Trilinos CVS repository. Access to the Trilinos repository is granted only to co-developers, Sandians, and in select special cases.

To access the repository, an account on software.sandia.gov is required. To request an account, send a note to `trilinos-help@software.sandia.gov` . The following two environment variables must be set to access the repository:

Command: `CVSROOT :ext:your_user_name@software.sandia.gov:/space/CVS`

Command: `CVS_RSH ssh`

Replace "your_user_name" with your user name on software.sandia.gov.

To checkout a working copy of the development branch of Trilinos in the current directory, type

Command: `cvs checkout Trilinos`

To checkout a working copy of a release branch of Trilinos in the current directory, type

Command: `cvs checkout -r name_of_release_branch Trilinos`

Replace "name_of_release_branch" with the name of the release branch. The name of the current release branch can be obtained by sending a note to `trilinos-help@software.sandia.gov` .

To checkout a working copy of the development version of only one Trilinos package in the current directory, type

Command: `cvs checkout <package_name>`

Replace "package_name" with the name of the package. Please note that many packages have dependencies on other Trilinos packages.

For those not familiar with CVS, a brief discussion covering some of the most common CVS commands is available in Section 3.2. For a more complete listing of CVS commands, see the GNU CVS Home Page [2].

2.2 Recommended Build Directory Structure

Via Autoconf and Automake the Trilinos configuration facilities provide a great deal of flexibility for configuring and building the existing Trilinos packages. However, unless a user has prior experience with Autotools, we recommend the following process to build and maintain local builds of Trilinos.

To start, we defined two useful terms:

- Source tree - The directory structure where source files are found. A source tree is obtained by expanding a distribution tar ball, or by checking out a copy of the Trilinos repository.
- Build tree - The directory structure where object and library files, as well as executables are located.

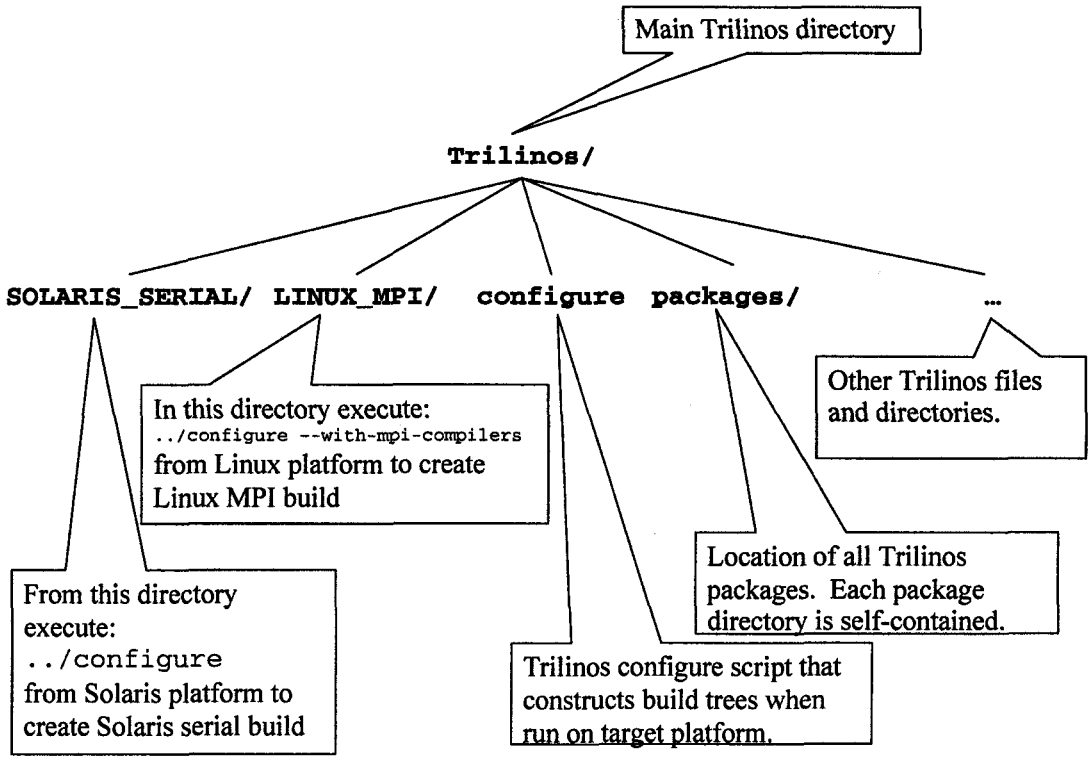

Figure 2. Recommended Layout for Trilinos Build Directories

Although it is possible to run `./configure` from the source tree (in the directory where the configure file is located), we recommend separate build trees. The greatest advantage to having a separate build tree is that multiple builds of the libraries can be maintained from the same source tree. For example, both serial and parallel libraries can be built. This approach also eliminates problems with configuring in a 'dirty' directory (one that has already been configured in).

Key Point: ... we recommend separate build trees ... multiple builds of the libraries can be maintained from the same source tree ... problems with configuring in a 'dirty' directory (are eliminated) ...

Setting up a build tree is straight-forward. Figure 2 illustrates the recommended layout. First, from the highest directory in the source tree (Trilinos for a repository copy, Trilinos-3.0.2 for a distribution), make a new directory - for an MPI build on a

Linux platform, a typical name is `LINUX_MPI` . Finally, from the new directory, type

Command: `../configure --with-mpi-compilers`

(Note that various configure options might be necessary, see Section 2.3 for details.) Finally, type

Command: `make`

In summary:

```
cd Trilinos
mkdir LINUX_MPI
cd LINUX_MPI
../configure --with-mpi-compilers
make
```

At this point, the MPI version of Trilinos on a Linux platform is built and completely contained in the `LINUX_MPI` directory. No files outside this directory have been modified. This procedure can be repeated for any number of build targets.

Note: Although we recommend the above location for build trees, they can be set up anywhere.

2.3 Configuring Trilinos

The most common issue encountered when configuring Trilinos is that it is nearly impossible to determine what caused configure to fail based on the standard output. If the output from configure is inadequate, look at the config.log file (in the buildtree) for the package that failed to configure properly.

> **Key Point:** ... to determine what caused configure to fail ... look at the config.log file ...

To determine which package failed to configure, look at the bottom of the output from the `configure` command. One of the last lines will say something like:

```
configure: error: /bin/sh '../../../packages/epetra/configure'
failed for packages/epetra
```

This particular error indicates to look in `packages/epetra/config.log` .

To configure from a remote build tree, simply run the configure script in source tree from the root of the build tree. In the example above, cd to the SOLARIS_SERIAL directory and type

Command: `../configure <configure options>`

A detailed list of configure options can be seen by typing

Command: `./configure --help=recursive`

from the top level of the source tree. This will display the help page for the Trilinos level as well as all Trilinos packages that use Autoconf and Automake. The output from this command is quite extensive. To view the help page for an individual package, cd to the home directory for the package in the source tree and type

Command: `./configure --help`

This command will also display the help page for Trilinos level options when used from the Trilinos home directory in the source tree.

Many of the Trilinos configure options are used to describe the details of the build. For instance, serial or mpi, all of the packages, or just a proper subset.

To configure for serial libraries, no action is necessary, but to configure for parallel libraries, a user must append appropriate arguments to the configure invocation line as described in "Trilinos Configuration Options", section 2.4.

Also, to build the default set of Trilinos libraries, no action is necessary, but to exclude a package that is built by default, Komplex for example, append `--disable-komplex` to the configure invocation line. Similarly, to include a package that is not currently built by default, NOX for example, append `--enable-nox` to the configure invocation line. It is recommended that users always configure from the Trilinos level and use `--disable-<package>` as necessary, rather than trying to configure from a lower level. To see which packages are built by default and which ones aren't, simply cd to the Trilinos home directory and type

Command: `./configure --help`

NOTES:

1. **Enabling/Disabling package builds:** The configure process is set up to detect when a `--disable-<package>` option would break a package dependency. For example, Ifpack depends on Epetra, so if a user wants to build Ifpack, but types `--disable-epetra` , Epetra will be configured and built anyway.

2. **Installing libraries and header files:** To install libraries and header files in a particular location, use `--prefix=<dir>` on the configure line. If this option is used, libraries will be located in `<dir>/lib` and header files in `<dir>/include/<package>` .

3. **Providing additional information to Autotools:** Although Autotools will try to determine all configuration information, the user must provide anything that Autotools needs and cannot find. Also, if Autotools selects, for example, the wrong BLAS library by default, the user must indicate which BLAS library to use. Other issues such as standards non-compliance are also dealt with here. If all required libraries (often the BLAS and LAPACK) are located in standard places and no special compiler flags are required, try configuring without providing additional information.

4. **Sample configure invocation scripts:**

   Sample configure invocation scripts for a wide variety of platforms can be found in `Trilinos/config` . These scripts are named using the following convention: `arch_comm_machine` . For example, `sgi64_mpi_atlantis` .

   Key Point: Sample configure invocation scripts for a wide variety of platforms can be found in `Trilinos/config` .

   Note that these scripts are examples only and are primarily useful for the values of options such as `LDFLAGS` , `CPPFLAGS` , and `CXXFLAGS` . Do not expect to be able to find a script that can be used without modification; try to find a script for a similar machine to use as a guide.

   The scripts in the repository are not always up to date. If a user submits a script for a machine that few Trilinos developers have an account on, that script may become obsolete if it is not updated by the user who submitted it.

   Users who create scripts for other machines are encouraged to check them into the repository for the benefit of other users. Users who do not have access to the repository can send scripts to the Trilinos Library Manager.

   The following is an example configure invocation script for an SGI machine:

```
../configure --enable-mpi --with-mpi-libs=-lmpi \
--with-cflags=-64 --with-fflags=-64 \
--with-cxxflags="-64 -LANG:std  -LANG:ansi-for-init-scope=ON \
```

```
-ptused -DMPI_NO_CPPBIND" \
LDFLAGS=" -64 -L/usr/lib64/mips4/r10000 -L/usr/lib64/mips4 \
-L/usr/lib64 " \
--enable-epetraext --enable-new_package \
--disable-komplex --enable-tsfcoreutils
```

2.4 Trilinos Configuration Options

The following options apply to all Trilinos packages unless an option doesn't make sense for a particular package (for example, a package that does not include any Fortran code will not be sensitive to `F77=g77`), or otherwise noted. For options specific to an individual package, cd to the home directory of the package and type

Command: `./configure --help`

.

Basic Options

- `--enable-debug`

  (NOX only.) This turns on compiler debugger flags. It has not been fully tested. As an alternate, specify CXXFLAGS on the configure line.

- `--enable-opt`

  (NOX only.) This turns on compiler optimization flags. It has not been fully tested. As an alternate, specify CXXFLAGS on the configure line.

- `--with-cppflags`

  Specify additional preprocessor flags (e.g., "-Dflag -Idir")

- `--with-cxxflags`

  Specify additional C++ flags

- `--with-ldflags`

  Specify additional linker flags (e.g., "-Ldir")

- `--with-ar`

  Specify a special archiver command, the default is "ar cru".

Influential Environmental Variables

- `CC`

  C compiler command.

- `CFLAGS`

  C compiler flags.

- `CXX`

  C++ compiler command.

- `CXXFLAGS`

  C++ compiler flags.

- `LDFLAGS`

  Specify linker flags.

- `CPPFLAGS`

  C/C++ preprocessor flags.

- `CXXCPP`

  C++ preprocessor.

- `F77`

  Fortran 77 compiler command.

- `FFLAGS`

  Fortran 77 compiler flags.

MPI-Related Options

- `--enable-mpi`

  Enables MPI mode. Defines HAVE_MPI in the (Package)_Config.h file. Will test for the ability to preprocess the MPI header file and may test ability to link with MPI. This option is rarely necessary as many of the below options also turn MPI on.

- `--with-mpi-compilers`

  Sets CXX = mpicxx (or mpiCC if mpicxx not available), CC = mpicc and F77 = mpif77. Automatically enables MPI mode. To use compilers other than these, specify MPI locations with the below options. If none of these options are necessary, use `--enable-mpi` to enable MPI mode. In this case, CXX, CC, and F77 have to be set if the correct compilers are not chosen by default.

- `--with-mpi=MPIROOT`

  Specify the MPI root directory. Automatically enables MPI mode. If this option is set, `--with-mpi-incdir` and `--with-mpi-libdir` should not be used. `--with-mpi` is a shortcut for setting `--with-mpi-libdir=MPIROOT/lib` and `--with-mpi-incdir=MPIROOT/include` .

- `--with-mpi-libdir=DIR`

  Specify the MPI libraries location. Defaults to MPIROOT/lib if `--with-mpi` is specified. If multiple directories must be specified, try `--with-ldflags="-L<dir1> -L<dir2>"` instead.

- `--with-mpi-libs="LIBS"`

  Specify the MPI libraries. Defaults to `"-lmpi"` if either `--with-mpi` or `--with-mpi-libdir` is specified.

- `--with-mpi-incdir=DIR`

  Specify the MPI include files location. Defaults to `MPIROOT/include` if `--with-mpi` is specified. If multiple directories must be specified, try `--with-cppflags="-I<dir1> -I<dir2>"` instead.

Developer-Related Options

- `--enable-maintainer-mode`

  Enable make rules and dependencies not useful (and sometimes confusing) to the casual installer.

2.5 Building Trilinos

If the configure stage completed successfully, just type

Command: `make`

and then, if `--prefix` was specified,

Command: `make install`

Other important notes about the configure and build processes.

- Any code that links to Trilinos should define `HAVE_CONFIG_H`.
- When building with LAM under RH9 Linux, configure complains that it cannot find mpi++.h. The message in the config.log file is:

  ```
  /usr/include/mpi.h:1064:19: mpi++.h: No such file or directory
  ```

  The following modified configure invocation works:

  Command: `../configure --enable-mpi CXX="mpiCC -DLAM_BUILDING"`

- The build process will fail on OSX if "DropZip" is used to unzip the Trilinos tarball. This utility truncates long file names.

2.6 Testing the Build

Trilinos contains a script that can be used for the purpose of verification testing. This script is located in `Trilinos/testharness/checkin-test-harness`. The script provides an easy way to run a set of tests for the packages that have been included in a build (not all packages support testing). Instructions for running the script can be found within the script itself.

3 Ongoing Use and Support

In this section we discuss how to report issues (bugs) and how to sign up for mail lists.

3.1 Reporting Bugs

Feature and issue reports are tracked using Bugzilla [1]. Bugzilla can be found on the web at `http://software.sandia.gov/bugzilla`. A Bugzilla account is necessary for submitting bugs. Those interested can sign up at the website. All bugs related to any Trilinos package that uses Bugzilla should be submitted to Bugzilla. A secondary method of submitting bug reports is to send a note to `Trilinos-Bugs@software.sandia.gov`.

Regardless of how an issue is submitted, the bug report should be filled out with as much detail as possible. Specifically, be sure to include which version of Trilinos the bug was discovered in and which platform(s) and compiler(s) the bug is associated

with. Also include any specific error messages and any additional information that can be provided.

NOTE: In the context of Bugzilla, "bug" can refer not only to an error in existing code, but also to a desired enhancement. For example, a bug report should be submitted to Bugzilla to report a segmentation fault that occurs when using an existing Ifpack preconditioner, and a bug report should also be submitted to request a new Ifpack preconditioner. "Issue" and "bug" are used interchangeably in this discussion of Bugzilla.

3.2 Signing Up for Mail Lists

Email lists are maintained for Trilinos as a whole and for each package through Mailman [6]. This tool can be found on the web at `http://software.sandia.gov/mailman/listinfo` . Those interested in signing up for one or more lists may do so at the website. A digest mode is available for those who wish to receive a daily summary of list activity. Non-Sandians are able to sign up for the "Users" and "Announce" lists. Sandians should keep this in mind when posting to these lists.

The example mailing lists mentioned below are to be used for issues relating to all of Trilinos. The names for the lists pertaining to individual packages follow the same naming scheme, simply replace "Trilinos" with the name of the package. For example the list for Trilinos users is called Trilinos-Users and the email address is `trilinos-users@software.sandia.gov` The list for Epetra users is called Epetra-Users and the associated email address is `epetra-users@software.sandia.gov`

Tip: While those who use any individual Trilinos package are also "Trilinos users", the lists are not set up to recognize this. In other words, those who subscribe to the Epetra-Users mailing list do not necessarily form a subset of those who subscribe to the Trilinos-Users mailing list. This is also true of all other list types. Keep this in mind when subscribing and posting to lists.

- Trilinos-Announce `trilinos-announce@software.sandia.gov`
  All Trilinos release announcements and other major news.
- Trilinos-Users `trilinos-users@software.sandia.gov`
  List for Trilinos Users. General discussions about the use of Trilinos.

References

[1] Mozilla bugzilla home page. http://www.mozilla.org/projects/bugzilla.

[2] Gnu cvs home page. http://www.gnu.org/software/cvs.

[3] David Day and Michael A. Heroux. Solving complex-valued linear systems via equivalent real formulations. *SIAM J. Sci. Comput.*, 23(2):480–498, 2001.

[4] Michael Heroux and et.al. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[5] Michael A. Heroux, James M. Willenbring, and Robert Heaphy. Trilinos Developers Guide. Technical Report SAND2003-1898, Sandia National Laboratories, 2003.

[6] Gnu mailman home page. http://www.gnu.org/software/mailman/mailman.html.

[7] Ray S. Tuminaro, Michael A. Heroux, Scott. A. Hutchinson, and J. N. Shadid. *Official Aztec User's Guide, Version 2.1.* Sandia National Laboratories, Albuquerque, NM 87185, 1999.

Commonly Used CVS Commands

To access the Trilinos CVS repository, an account on software.sandia.gov is required. To request an account, send a note to `trilinos-help@software.sandia.gov`. The following two environment variables must be set to access the repository:

Command: `CVSROOT :ext:your_user_name@software.sandia.gov:/space/CVS`

Command: `CVS_RSH ssh`

(Replace "your_user_name" with your user name on software.sandia.gov.)

Below is a brief description of the CVS commands that are most commonly used by Trilinos users. For a more complete listing of CVS commands, see the GNU CVS Home Page [2].

- **Checking Out a Working Copy:** To checkout a working copy of the development branch of Trilinos in the current directory from the CVS repository, type

  Command: `cvs checkout Trilinos`

  To checkout a working copy of only one package of Trilinos in the current directory, type

  Command: `cvs checkout <package_name>`

  (Replace "package_name" with the name of the package.

  To checkout a different branch or a tagged version of Trilinos, type

  Command: `cvs checkout -r <name_of_branch_or_tag> Trilinos`

- **Updating a Working Copy:** To update after a version has been obtained use the `cvs update` command. First, cd to the directory that is to be updated (often the Trilinos root directory). Then type:

  Command: `cvs -q update -dP`

  The "-q" option means "be somewhat quiet". Try an update without the "-q" to see exactly what the option does.

The "-d" option means to get any new directories. For example, if a new package is added to the repository, but the "-d" option is not used, that new package will never appear in the working copy. However, the first time that the "-d" option is used, all of the new package directories will be downloaded, and from that time on, all CVS updates will update the directories that were downloaded. It is good practice to include this option for every CVS update.

Finally the "-P" option "prunes" empty directories. This helps to keep the directory structure from getting more cluttered than it needs to. For example, the old "petra" and "tsf" packages were removed from the repository, but the directory structures will remain if this option is not specified. If an empty directory is needed, simply issue one update command without the "-P" and the empty directories will be restored.

Distribution list:

Internal Distribution:

1	MS 1111	J. N. Shadid, 9233
1	MS 0316	S. A. Hutchinson, 9233
1	MS 1111	E. R. Keiter, 9233
1	MS 9217	J. J. Hu, 9214
1	MS 1110	J. M. Willenbring, 9214
1	MS 9217	K. R. Long, 8962
10	MS 1110	M. A. Heroux , 9214
1	MS 1110	R. A. Bartlett, 9214
1	MS 1110	R. B. Lehoucq, 9214
1	MS 1110	R. Heaphy, 9215
1	MS 0316	R. Hooper, 9233
1	MS 0316	R. J. Hoekstra, 9233
1	MS 1110	K. W. Kampshoff, 9214
1	MS 0316	R. P. Pawlowski, 9233
1	MS 9217	R. S. Tuminaro, 9214
1	MS 9217	T. Kolda, 8962
1	MS 9217	V. E. Howle, 8962
1	MS 0835	K. H. Pierson, 9142
1	MS 0826	A. B. Williams, 8961
1	MS 9915	N. M. Nachtigal, 8961
1	MS 9915	A. J. Rothfuss, 8961
1	MS 1110	H. K. Thornquist, 9214
1	MS 1152	J. D. Kotulski, 1642
1	MS 1166	C. R. Drumm, 15345
1	MS 0819	A. C. Robinson, 9231
1	MS 0827	H. C. Edwards, 9143
1	MS 0316	S. J. Plimpton, 9212
1	MS 0847	M. S. Eldred, 9211
1	MS 1111	K. D. Devine, 9215
1	MS 0819	M. A. Christon, 9231
1	MS 1111	A. G. Salinger, 9233
1	MS 1111	E. T. Phipps, 9233
1	MS 0316	G. L Hennigan, 9233
1	MS 0828	C. C. Ober, 9233
1	MS 0316	W. F. Spotz, 9233
1	MS 0835	A. A. Lorber, 9141
1	MS 9217	P. T. Boggs, 8962
1	MS 0835	S. R. Subia, 9141
1	MS 0826	J. R. Stewart, 9143
1	MS 0835	S. W. Bova, 9141
1	MS 0834	R. P. Schunk, 9114

1	MS 0834	R. R. Rao, 9114
1	MS 0827	P. A. Sackinger, 9113
1	MS 0834	M. M. Hopkins, 9114
1	MS 0834	H. K. Moffat, 9114
1	MS 0847	G. M. Reese, 9142
1	MS 1110	D. M Day, 9214
1	MS 0847	B. G. van Bloemen Waanders, 9211
1	MS 9018	Central Technical Files, 8945-1
2	MS 0899	Technical Library, 9616

External distribution:

Steve Ashby
University of California
Lawrence Livermore National Laboratory
7000 East Ave., Livermore, CA 94550-9234
P.O. Box 808, Livermore, CA 94551-0808

Richard Barrett
Los Alamos National Laboratory
Mail Stop B272
Los Alamos, NM 87545

Jason Cross
Box 429
St. John's University
Collegeville, MN 56321

Tim Davis, Assoc. Prof.
Room E338 CSE Building
P.O. Box 116120
University of Florida
Gainesville, FL 32611-6120

Jeffrey J. Derby
CE1 MS Department, U. of MN
421 Washington Ave SE
Minneapolis, MN 55455-0132

Jack Dongarra
Computer Science Department
1122 Volunteer Blvd
Knoxville, TN 37996-3450

Craig Douglas
325 McVey Hall - CCS
Lexington, KY 40506-0045

Victor Eijkhout
Department of Computer Science,
203 Claxton Complex, 1122 Volunteer Boulevard,
University of Tennessee at Knoxville,
Knoxville TN 37996, USA

Anshul Gupta
IBM
P. O. Box 218
1101 Kitchawan Road
Yorktown Heights, NY 10598

Paul Hovland
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

C.T. Kelley
Department of Mathematics , Box 8205
Center for Research in Scientific Computation
North Carolina State University
Raleigh, NC 27695-8205

David Keyes
Appl Phys & Appl Math
Columbia University
200 S. W. Mudd Building
500 W. 120th Street
New York, NY, 10027

Xiaoye Li
Lawrence Berkeley Lab
50F-1650
1 Cyclotron Rd
Berkeley, CA 94720

Lois Curfman McInnes
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

Juan Meza
Department Head, High Performance Computing Research
Lawrence Berkeley National Laboratory
Mail Stop 50B-2239
Berkeley, CA 9472

Mike Phenow
PO Box 1392
St. John's University
Collegeville, MN 56321

Padma Raghavan
Department of Computer Science and Engineering
308 Pond Laboratory
The Pennsylvania State University
University Park, PA 16802-6106

Chuck Romine
Program Manager, Applied Mathematics
U.S. Department of Energy
1000 Independence Ave., SW
Washington, DC 20585-1290

Paul Sexton
Box 1560
St. John's University
Collegeville, MN 56321

Barry Smith
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

Ken Stanley
322 W. College St.
Oberlin OH 44074

Danny Sorenson
Dept. of Computational & Applied Mathematics
Rice University
6100 Main St. - MS 134
Houston, TX 77005-1892

Carol Woodward
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Box 808, L-561
Livermore, CA 94551