Setting Up a pbdR Environment

Installing MPI, R, and pbdR

Version 2.0

Prepared by the pbdR Core Team:

Drew Schmidt
Remote Data Analysis and Visualization Center,
University of Tennessee, Knoxville

Wei-Chen Chen
Department of Ecology and Evolutionary Biology,
University of Tennessee, Knoxville

Pragneskumar Patel
Remote Data Analysis and Visualization Center,
University of Tennessee, Knoxville

George Ostrouchov
Computer Science and Mathematics Division,
Oak Ridge National Laboratory

This publication was typeset using LaTeX.

Contents

1 Allocation

We have tried to make the installation process as simple as possible, and these instructions as thorough as possible. However, this is not an entirely labor-free procedure, and does not even get into the really difficult side of large-scale computing: managing the system.

If you affiliated with a United States institution and are engaged in research that requires large-scale computing resources, we encourage you to consider getting an allocation with us . Not only can we tailor our pbdR development to help your research, but we can manage the hardware, operating system, and software utilities for you, so that you can focus entirely on the thing that matters, your research.

2 Quick Introduction

In this guide, we will detail the necessary steps for how to set up a pbdR environment. What follows in the remaining sections is a very lengthy list of installation instructions; however, most users should find the process fairly straight-forward, and may not need (or want) all of the details we will provide unless something goes wrong. In any case, the short version for setting up a pbdR environment is to:

1. install R ; see http://cran.r-project.org/
2. install an MPI library; http://www.open-mpi.org/, or http://www.mpich.org/ for Windows
3. install the pbdR packages; see http://r-pbd.org/

Items 1 and 2 are interchangeable, and so if you already have R and/or an MPI library installed, then merely skip this/these step(s); there is no need to reinstall anything.

2.1 Installing R

This should be fairly painless. R has binary packages for every operating system you have heard of (and some you haven't), and the install should go fine. Of course, since R is open source, you are free to compile it yourself, should have have reason or need to do so. You can find both the source as well as binaries at the R project's main site: http://cran.r-project.org/.

Additionally, you may wish to customize your R build by compiling from source. For example, you may wish to link R with a high performance linear algebra library, such as MKL. See the *R Installation and Administration Manual* at http://cran.r-project.org/doc/manuals/R-admin.html for full details.

2.2 Installing MPI

2.3 Installing pbdR Packages

All released pbdR packages are available from http://cran.r-project.org/ which is the Comprehensive R Archive Network (CRAN). This is similar to the CPAN for perl or CTAN for LaTeX, although with many improvements and benefits over its competitors.

It is also possible to link pbdR with high performance linear algebra libraries, such as MKL. Figure 1 offers some insight into the package organization. See the pbdSLAP vignette for more details.

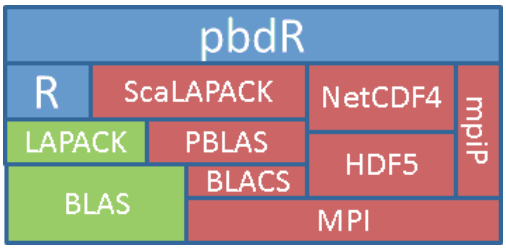

Figure 1: pbdR Relationships to Libraries

3 R and pbdR installation on Cray system

Below is installation instructions of R on Cray machine. Prerequisites for R and pbdR software installation on Cray machine: GCC, Cray-MPICH and ACML/MKL.

We use following software tools/packages: Programming Environment GNU-4.2.34 with **GCC**-4.8.2, **Cray-MPICH**-6.2.0, **ACML**-5.3.1, and R 3.0.2(compied dynamically). The R packages and their corresponding versions used in these: **rlecuyer** 0.3-3, **memuse** 0.2-0, **pbdMPI** 0.2-2, **pbdSLAP** 0.1-8, **SEXPtools** 0.1-0, **pbdBASE** 0.3-0, and **pbdDMAT** 0.3-0.

The below script is also available here.

```
## Load/set appropriate modules and software environments
module swap PrgEnv-cray PrgEnv-gnu/4.2.34
module load java/jdk1.7.0_45
module load acml/5.3.1

## Create installation directory on lustre space
mkdir -p /scratch/sciteam/\$USER/R-3.0.2-Install
export R_WORK_HOME=/scratch/sciteam/\$USER/R-3.0.2-Install
cd \$R_WORK_HOME

## Download R source code
wget http://mirrors.nics.utk.edu/cran/src/base/R-3/R-3.0.2.tar.gz
tar -xzvf R-3.0.2.tar.gz
cd R-3.0.2

## Configure R
./configure --prefix=\$R_WORK_HOME --enable-R-profiling --enable-
   memory-profiling --enable-R-shlib --enable-BLAS-shlib --enable-lto
   --enable-byte-compiled-packages --enable-shared --enable-long-double
    --with-readline --with-tcltk --with-cairo --with-libpng --with-
   jpeglib --with-libtiff --with-system-pcre --with-valgrind-
   instrumentation --with-x --with-blas=``-I/opt/acml/5.3.1/
   gfortran64_mp/include -L/opt/acml/5.3.1/gfortran64_mp/lib -lacml_mp
   '' --with-lapack > configure.log

make > make.log
make check > make_check.log
make install > make_install.log
make check-all > make_check_all.log

## Install R packages
export PATH=\$R_WORK_HOME/bin:\$PATH
```

```
export LD_LIBRARY_PATH=\$R_WORK_HOME/lib64/R/lib:\$LD_LIBRARY_PATH

mkdir -p \$R_WORK_HOME/R_pkg_sources
cd \$R_WORK_HOME/R_pkg_sources


## Download R packages
wget http://cran.r-project.org/src/contrib/rlecuyer_0.3-3.tar.gz
wget http://cran.r-project.org/src/contrib/pbdMPI_0.2-2.tar.gz
wget http://cran.r-project.org/src/contrib/pbdSLAP_0.1-8.tar.gz
wget https://github.com/wrathematics/pbdBASE/archive/master.zip
mv master pbdBASE.zip
unzip pbdBASE.zip

wget https://github.com/wrathematics/pbdDMAT/archive/master.zip
mv master pbdDMAT.zip
unzip pbdDMAT.zip

wget  https://github.com/wrathematics/SEXPtools/archive/master.zip
mv master SEXPtools.zip
unzip SEXPtools.zip

wget https://github.com/wrathematics/memuse/archive/master.zip
mv master memuse.zip
unzip memuse.zip

## R packages installation
R CMD INSTALL --no-test-load rlecuyer_0.3-3.tar.gz

## pbdMPI install
R CMD INSTALL --no-test-load pbdMPI --configure-args=''--with-mpi=/opt
   /cray/mpt/6.2.0/gni/mpich2-gnu/48/ --with-mpi-type=MPICH3''

R CMD INSTALL --no-test-load pbdSLAP_0.1-8.tar.gz

R CMD INSTALL --no-test-load pbdBASE

R CMD INSTALL --no-test-load pbdDMAT

R CMD INSTALL --no-test-load SEXPtools

R CMD INSTALL --no-test-load memuse

## Copy all needed dynamic libraries to Lustre

mkdir -p \$R_WORK_HOME/system_libs
cd \$R_WORK_HOME/
./dyn_libs_copy.sh \$R_WORK_HOME/lib64/R/lib \$R_WORK_HOME/system_libs
./dyn_libs_copy.sh \$R_WORK_HOME/lib64/R/library/pbdMPI/libs \
   $R_WORK_HOME/system_libs

echo ''Now you are ready to use \proglang{R} and \proglang{pbdR}. Good
    Luck !!!''
```

```
#**********************
#**********************
## Below is source code of ''dyn_libs_copy.sh''
## This script uses ldd and copy all needed shared object files
   recursively
OPTIONS_HELP='
Command line options:
     -h Print this help menu
     -v Verbose (lookup_dir_Path destination_dir_path)

Usage: ./dyn_libs_copy.sh lookup_dir_Path destination_dir_path
Usage Example:
./dyn_libs_copy.sh /package/version/os_compiler/bin /package/version/
   os_compiler/system_libs
./dyn_libs_copy.sh -v /package/version/os_compiler/lib /package/
   version/os_compiler/system_libs

Verbose mode: ./dyn_libs_copy.sh -v lookup_dir_Path
   destination_dir_path
'

# Debug function
log(){
if [[ \$debug -eq 1 ]]; then
        echo ``\$@''
    fi
}

## recur_copy function: takes two arguments: filename and Destination
   path
recur_copy()
{
    log -e ``#####\nFile Name(recur_copy) =>  $1''
    destination_copy_path_dir=''$2''
    log -e ``Destination dir path: $destination_copy_path_dir''
    dep_list=`ldd $1 | perl -p -e 's/[^=]*=> ([^\s]*).*/$1/g' | egrep
       '^\/.*'`
    dep_arr=($dep_list)

    for dep_file_path in ``${dep_arr[@]}''
    do
        log -e ``\t\tdependency:  $dep_file_path''
        dep_file_name=`basename $dep_file_path`
        log $dep_file_name
        if [ ! -f $dep_file_name ]
        then
            log -e ``\t\t\tFile $dep_file_name does not exists''
            cp $dep_file_path $destination_copy_path_dir
            log -e ``\t\t\tcopied $dep_file_name TO
               $destination_copy_path_dir''

            recur_copy $dep_file_name $destination_copy_path_dir
        else
            log -e ``\t\t\tFile $FILE DOES exists''
```

```
            continue
        fi
    done
    log -e ``#####\n''
}


## list_file_in_dir: takes two arguments lookup directory name and
   Destination path. Call recur_
copy function
list_file_in_dir()
{
    cd $1
    log ``present working directory `pwd`''
    echo ``lookup dir name $1''
    for file in $1/* ## iterate over all files within directory
    do
        filename=`basename $file`
        log -e ``*****\nFile Name(list_files_in_dir) =>  $file''
        recur_copy $filename $2
        log -e ``*****\n''
    done
}

## Input args processing
echo ``WARNING : This script does not work with relative path. Please
   specify full path when you
pass arguments.''

if [[ $# -le 0 ]]; then
        echo ``Invalid arguments''
        echo ``$OPTIONS_HELP''
        break
fi
while test $# -gt 0; do
    case $1 in
       -v)
          debug=1
          #log ``some text''
        ;;
       -h)
            echo ``$OPTIONS_HELP''
            break
        ;;
        *)
        if [ $# -eq 2 ] && [ ! -z $1 ] && [ ! -z $2 ]
            then
                log ``two args found''
                LOOKUP_DIR=''$1''
                DEST_DIR=''$2''
                log ``Lookup_Dir = $LOOKUP_DIR''
                log ``Destination_Dir = $DEST_DIR''
                list_file_in_dir $1 $2
            else
```

```
                echo ``Invalid arguments''
                echo -e ``\nUsage: `basename $0` -h for help'';
                echo ``$OPTIONS_HELP''
            fi
            break
        ;;
    esac
    shift
done
```

4 Running pbdR Scripts

This information is covered in *much* more detail in the pbdDEMO vignette, and should not be considered a substitute. However, there are two key points one needs to understand in order to use pbdR tools. Namely,

- pbdR codes are written in Single Program/Multiple Data style
- pbdR codes are executed in batch

For full details, see the pbdDEMO package vignette.

Below is a simple pbdR script. This will help you know if things are installed properly or not. To understand what the script is doing, or to learn how to do much more substantial things, you should see the pbdDEMO package vignette.

```
1 library(pbdMPI, quiet = TRUE)
2 init()
3
4 x <- comm.rank()
5
6 comm.print(x, all.rank = TRUE)
7
8 finalize()
```

To run the script, you must do so in batch (i.e., non-interactively). First save its contents to the file `my_script.r`, and then open a .

5 Installation Problems

During the course of installation, you may run into unrecoverable issues. The pbdR team does not support MPI libraries or R core, so if you have problems during that portion of the installation phase, we probably can not directly help you. However, there are still many great resources at your disposal, maintained by those individual projects.

5.1 R and MPI

If you have problems installing or customizing R, see the *R Installation and Administration Manual* at http://cran.r-project.org/doc/manuals/R-admin.html for help.

If you are having trouble installing an MPI library, you should see that library's official documentation. For OpenMPI, see http://www.open-mpi.org/community/help/ and for MPICH, see http://www.mpich.org/documentation/guides/ .

For the remainder, we will be addressing installation issues with pbdR packages.

5.2 pbdR

This is a quick list of potential problems you could encounter when installing pbdR packages. For additional troubleshooting or installation options, each package has a vignette which may offer additional useful information.

- **When compiling pbdMPI from source**, you may be required to pass a configure argument at compile time. So for example, if you have OpenMPI installed and were installing from the command line, then you would issue the command:

  ```
  R CMD INSTALL pbdMPI_0.1-6.tar.gz \
          --configure-args='--with-mpi-type=OPENMPI'
  ```

  or if installing from R:

  ```
  install.packages("pbdMPI", configure.args='--with-mpi-type=OPENMPI')
  ```

  See the **pbdMPI** vignette for more details.

- **If you are installing on a cluster** where you must install on the login node which can not execute `mpirun`, then pass the install option `--no-test-load`. So for example, if installing from the command line, then you would issue the command:

  ```
  R CMD INSTALL pbdMPI_0.1-6.tar.gz --no-test-load
  ```

  or if installing from R:

  ```
  install.packages("pbdMPI", INSTALL_opts='--no-test-load')
  ```

- **If you are installing binaries on MAC OS X**, do not use the gui. You can install from source using the gui, or you can install binaries (or from source) using the terminal. But you can not install binaries using the gui. So if you want to install binaries, you should open Finder, then navigate to `Applications/Utilities/` and select `Terminal`. Next, type `R` and press enter. Now try to install the packages.