Running RSM on a Linux Box

John Benoit

USDA Forest Service

Riverside Forest Fire Laboratory

Riverside, California

Which would you rather work with?

This...

- Equipment cost: > $30,000
- Set up:
  - Outside consultants & labor needed
  - Custom wiring
- Maintenance:
  - Power consumption
  - Hardware & software updates
  - Monitoring

Which would you rather work with?

Or this??

- Equipment cost: $4,208

- Set up: Minimal

- Maintenance:
  - Less power consumption
  - Occasional hardware
    & software updates
  - Very little monitoring

Which would you rather work with?

versus

System specs

- HP XW8600 Workstation
- 2 Intel Xeon E5410 CPUs @ 2.33 GHz, 4 cores each (8 cores total)
- 64-bit processing
- 4 Gb of RAM
- 1 Tb hard drive
- Cost: $4,208

Installed Linux

(Why Linux?)

- It's free!
- Big user community
- 'Stable'
- It's a *nix (a good platform for RSM)

Installed Linux

- Downloaded Ubuntu 8.04.1 Server 64-bit image from *http://www.ubuntu.com*.

- Burned it onto a bootable CD.

- Booted from CD & followed instructions.

- Asked several questions:
  - Preferred language
  - Initial user accounts, passwords
  - Software selection
  - Etc.

Chose a Desktop Environment

- You have a choice of Desktop Environment (i.e. a GUI):
  (e.g. see *https://help.ubuntu.com/9.04/config-desktop/C/other-desktops.html*)

  - GNOME
    The default. Designed to 'just work'.

  - KDE
    "KDE is a popular, fully-featured desktop environment. Kubuntu is a version of Ubuntu which uses the KDE desktop."

  - Xfce
    "Xfce is a desktop environment which is designed to be fast and lightweight. Xubuntu is a version of Ubuntu which uses the Xfce desktop."

Installed & set options

- Installed SSH (secure shell) for remote logins.

    ```
    $ sudo apt-get install ssh
    ```

- Installed TCSH (similar to CSH).

    ```
    $ sudo apt-get install tcsh
    ```

- Changed from dynamic to a fixed IP

    In KDE, went to
          K button (　)
            ➔ Computer
              ➔ System Settings
                ➔ Network Settings

Useful Linux tools

See *https://help.ubuntu.com/8.04/add-applications/C/advanced.html* .
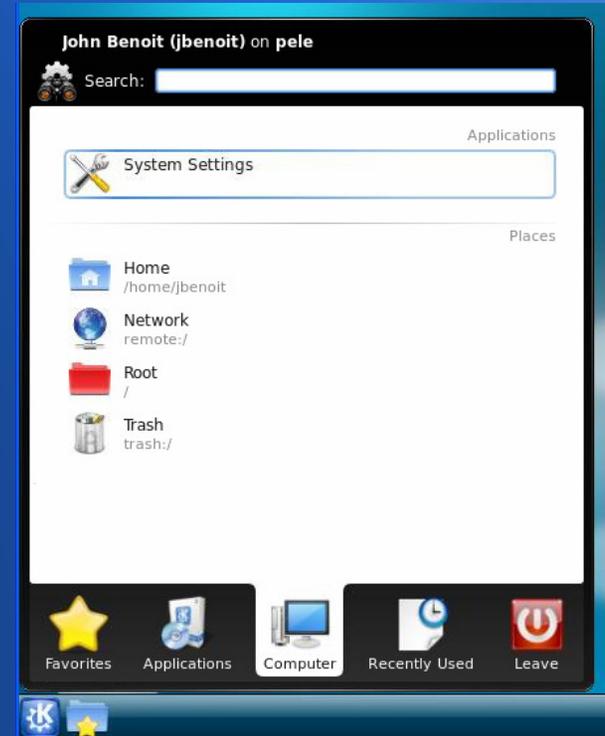
- Advanced Packaging Tool (apt)
  - To update the local list of packages:
    `sudo apt-get update`
  - To search for a package:
    `apt-cache search package`
  - To install a package:
    `sudo apt-get install package`

- In KDE, K button (◈)
  ➔ Applications
    ➔ System
      ➔ Software Management
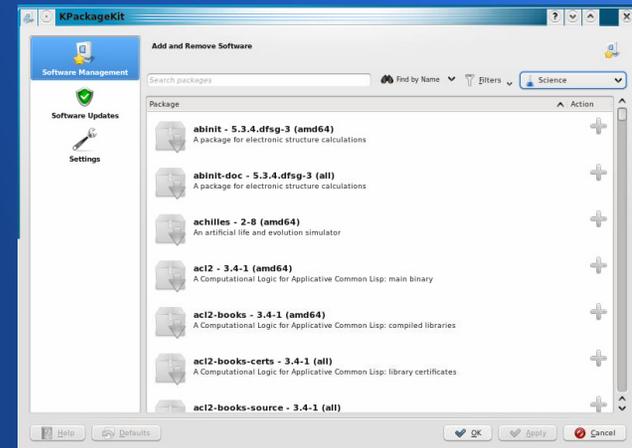
Useful Linux tools

- Remote desktop software

  e.g. *http://www.nomachine.com*
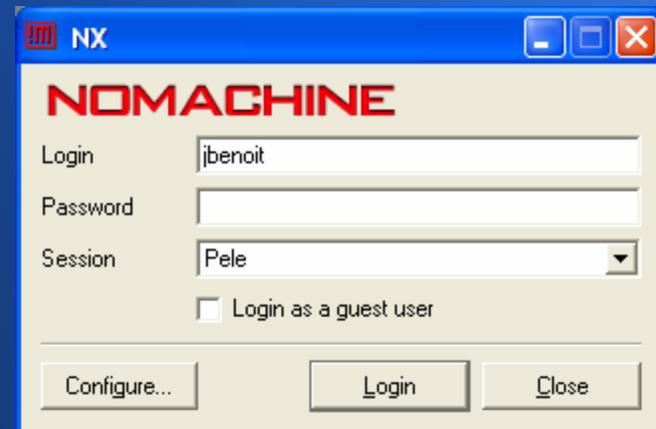
Problems

Some hardware-related (?) issues:

- Logging out of KDE (Kubuntu) produced a blank screen.

- Upgrade to Ubuntu 9.04 on an identical system crashed.

Installed Fortran

- Installed Intel 64-bit Fortran

- Now hard to find, but is probably at

  http://software.intel.com/en-us/articles/non-commercial-software-development/

  (You have to register & install a compiler bundle)

Installed Fortran

- Had to install a library:
    ```
    sudo apt-get install ia32-libs
    ```

- Have to add a line to the shell's resource file:
    - For *.bashrc*:
        ```
        source /opt/intel/fce/10.1.018/bin/ifortvars.sh
        ```
    - For *.cshrc* or *.tcshrc*:
        ```
        source /opt/intel/fce/10.1.018/bin/ifortvars.csh
        ```

- May need to add to path:
    ```
    /opt/intel/fce/10.1.018/bin
    ```

Installed LAM-MPI

- From *http://www.lam-mpi.org/*
  Version 7.1.4 or 7.1.5 beta

- Possible problem with 7.1.4 install / configure/ make:

  "compiler does not support 'bool'"

  known bug – try 7.1.5 beta version

- Added to path (in *.bashrc*):
  ```
  PATH="/usr/local/lam/bin:${PATH}";
  ```

# Tested LAM-MPI

Ran LAM-MPI on different numbers of cores:

```
$ mpif77 cpi.f -o cpi

$ lamboot -v lamhosts

$ mpirun -np n cpi >& log

$ lamhalt >& /dev/null

$ lamclean
```

where n = 1, 2, 3, 4, 5, 6, 7, 8
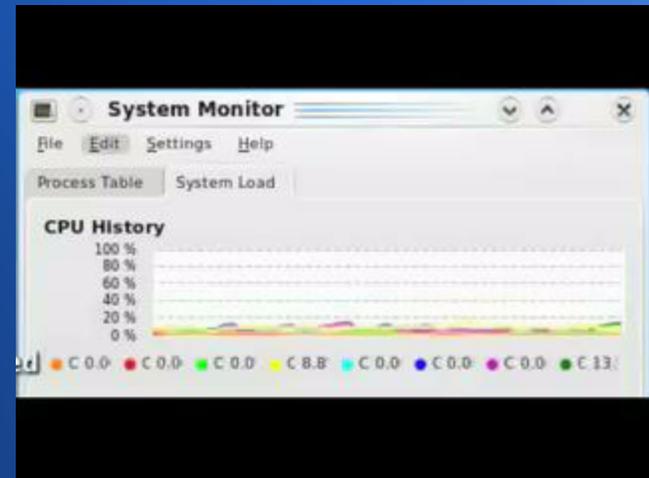
Tested LAM-MPI
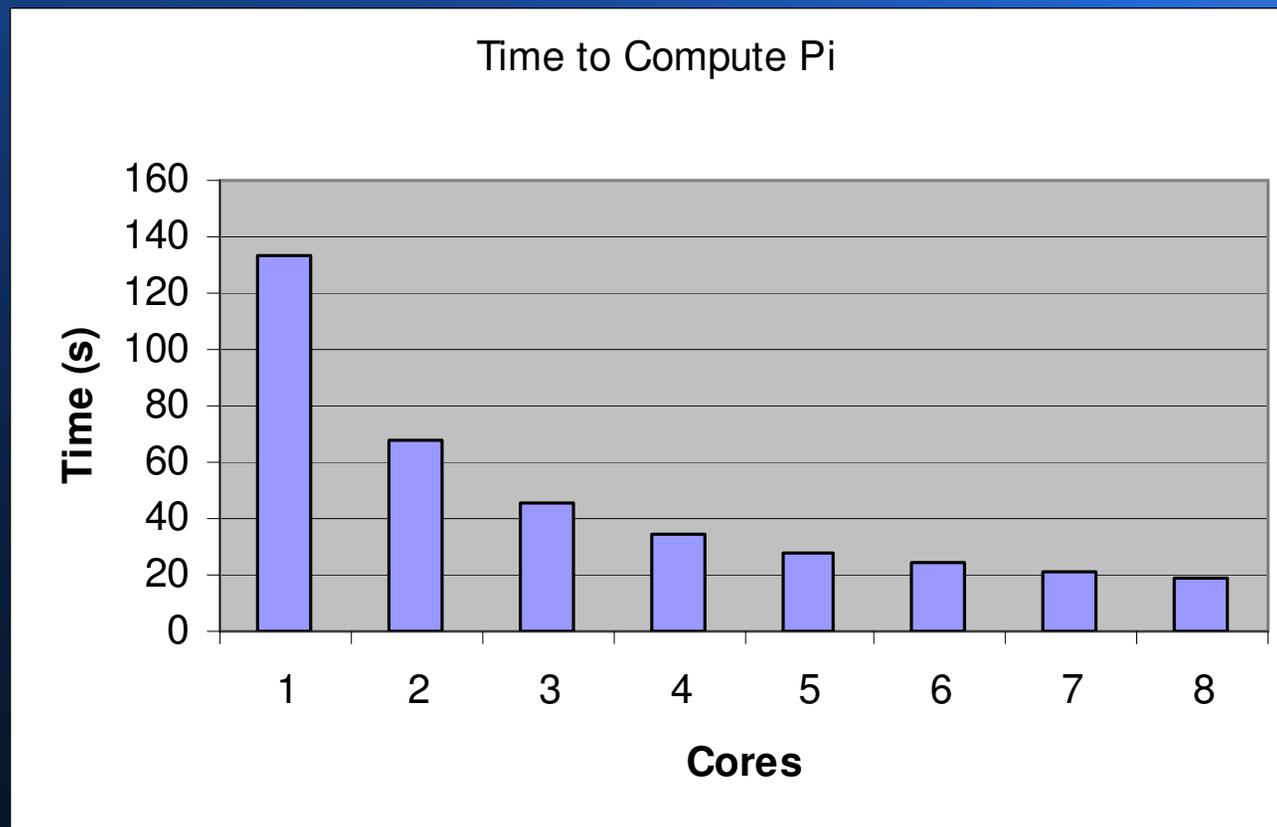
Installed CVS

- Manual to follow (old, but still usable)

  http://ecpc.ucsd.edu/projects/RSM/manual/rsm_cvs_v2.1.pdf

- CVS is <u>not</u> at ~~*https://www.cvshome.org*~~

  Try *http://www.nongnu.org/cvs/* or

  http://ftp.gnu.org/non-gnu/cvs/source/stable
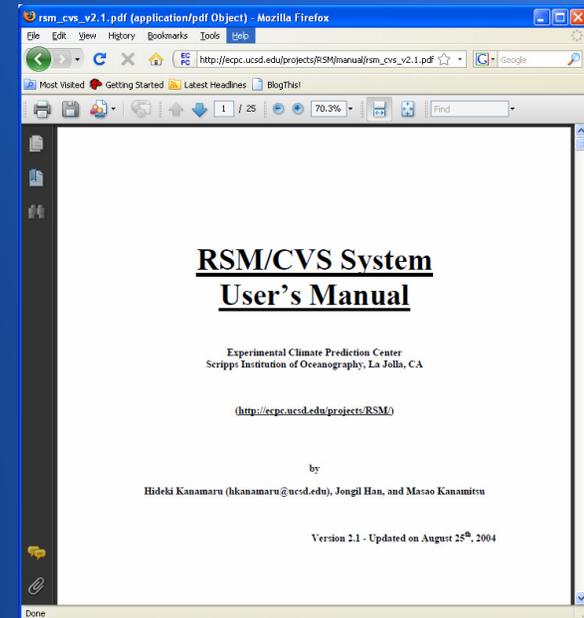
Installed CVS

- Installed the usual *nix way:

```
gunzip cvs-1.11.16.tar.gz
tar xvf cvs-1.11.16.tar
cd cvs-1.11.16
./configure --prefix= cvs_install_directory
make
make install
```

- Make sure to set Path & Environment Variables!

```
set PATH=($path cvs_install_directory/bin )

setenv CVSROOT :pserver:anoncvs@rokka.ucsd.edu:/rokka1/
                kana/cvs-server-root/cpscvs
```

Add to .cshrc, etc.

Installed RSM

- Created an `RSM/` directory.

- In `RSM/`, ran: `cvs co Install`
  (downloads the install script for RSM)

- Ran `./install` and chose options:
  - Selected '*rsm*' model
  - Selected '*single*' machine (vs. '*mpi*' version)
  - Selected '*linux*' as system
  - Selected '*ecpc*' version
  - Selected '*3*' (default) (rsm108x69_g62k28_africa_60km_str)
  - Selected '*rsm*' (default) for which script to run
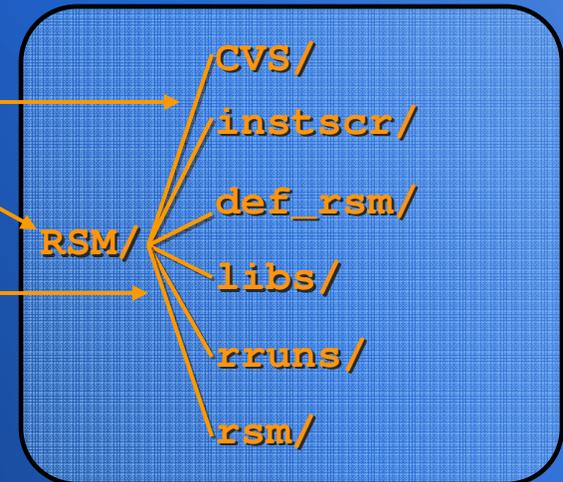
  Install takes about 5-10 minutes.

```
            CVS/
            instscr/
            def_rsm/
  RSM/      libs/
            rruns/
            rsm/
```

Installed RSM

Had some errors at end of install

```
creating etc/inchour
creating etc/Makefile
creating etc/mapinfo
creating etc/mpiset
creating etc/nt
creating etc/rmpiset
cd lib ; make || exit 8
make[1]: Entering directory `/home/rsm3/Desktop/RSM/RSM/libs/lib'
cd w3lib_linux ; make || exit 8
make[2]: Entering directory `/home/rsm3/Desktop/RSM/RSM/libs/lib/w3lib_linux'
pgf90 -Mrecursive -Mdalign -DLINUX -byteswapio -c aea.f
make[2]: pgf90: Command not found
make[2]: *** [aea.o] Error 127
make[2]: Leaving directory `/home/rsm3/Desktop/RSM/RSM/libs/lib/w3lib_linux'
make[1]: *** [w3lib_linux/w3lib.a] Error 8
make[1]: Leaving directory `/home/rsm3/Desktop/RSM/RSM/libs/lib'
make: *** [libs] Error 8
pele:~/Desktop/RSM/RSM>
pele:~/Desktop/RSM/RSM>
```

Ignored them for now.

Installed RSM

Edited *libs/configure-libs* file:
Changed `MACHINE=linux`
to `MACHINE=intel`

```
#! /bin/sh
#
#   cps library compilation configuration script.
#
#
#-------------------------------------------------
#
#   Change the following variable
#
#   machine stands for name of the machine dependent
#
#   MACHINE=[sgi/origin/ibmsp/sun/dec/hp/cray/t90/t3
#
MACHINE=linux
#
#   machine architecture
#
```

```
                    CVS/

                    instscr/

                    def_rsm/
        RSM/
                    libs/

                    rruns/

                    rsm/
```
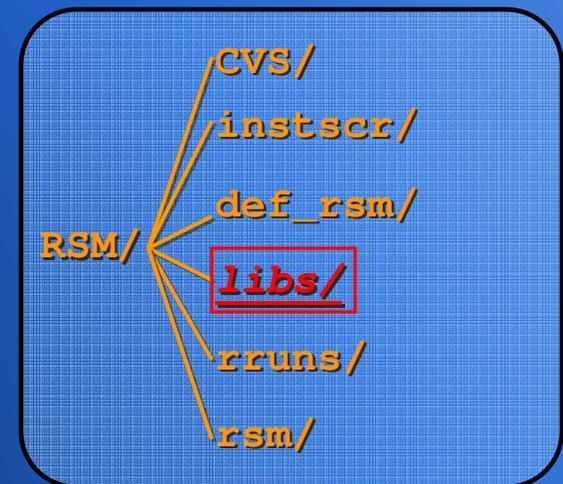
Installed RSM

Edited *libs/opt_libs/options-intel-single* file:

Changed `CC=icc`

to `CC=cc`

```
SHELL=/bin/sh
AR=ar
AS=as
CP=cp
RM=rm
CD=cd
CC=icc
CPP='gcc -E'
MAKE=make
MKDIR=mkdir
CHMOD=chmod
ECHO=echo
FTNID=fort.
#
#   Model compiler options (except fcst related)
#
F77=ifort
FORT_FLAGS="-r8 -O3 -convert big_endian"
LOAD_FLAGS="-r8 -O3 -convert big_endian"
```

CVS/
instscr/
def_rsm/
RSM/
libs/
opt_libs/
rruns/
rsm/
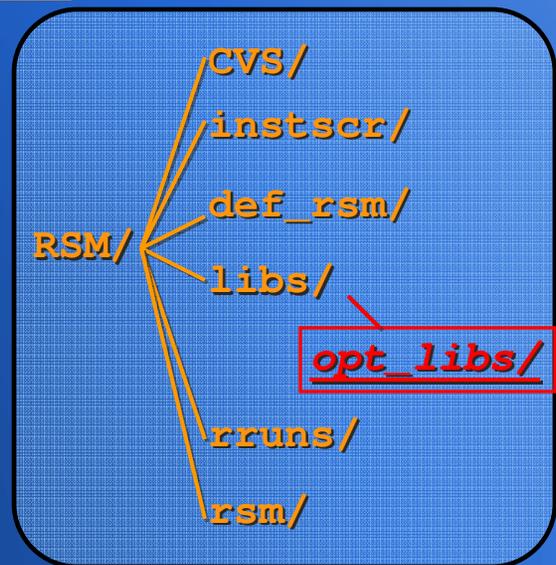
Some fixed issues

- Edited *rsm/src/fcst/setras.F*, inserted

    USEIFPORT

  After "**subroutine setras**" statement.

- Edited **rsm/opt/options-intel-single**, replaced **f77** with ifort

Fixed

```
CVS/
instscr/
def_rsm/
RSM/    libs/
rruns/
rsm/
                opt/
```
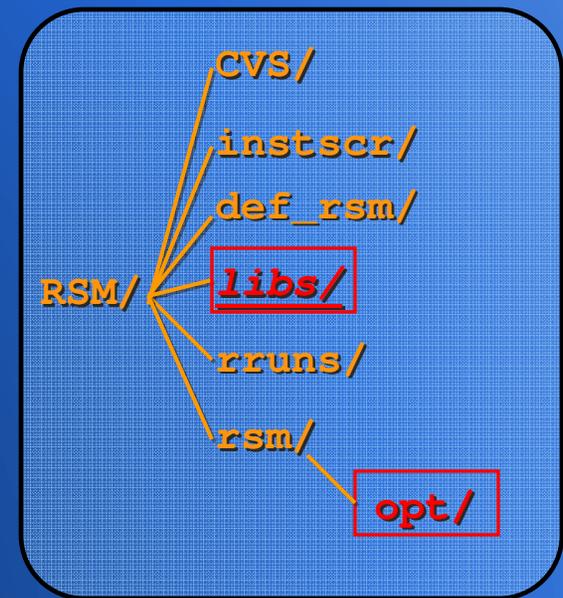
Configured RSM

Set your limits!

In the shell's resource file (e.g. `.tcshrc`, etc.):

```
limit   cputime             unlimited
limit   filesize            unlimited
limit   datasize            unlimited
limit   stacksize           65536
limit   coredumpsize        unlimited
limit   memoryuse           unlimited
limit   descriptors         unlimited
limit   memorylocked        unlimited
limit   maxproc             100
```

Configured RSM

- In *libs/*:

  ```
  configure-libs
  make clean
  make
  ```

```
        CVS/
       instscr/
      def_rsm/
RSM/  libs/
      rruns/
       rsm/
```
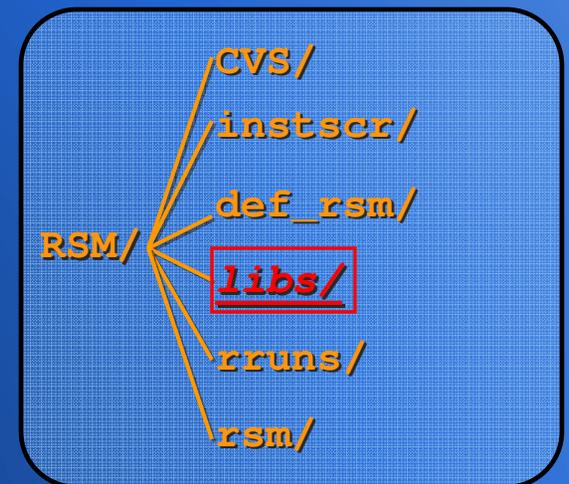
Configured RSM

- In *rsm/*:

  ```
  configure-model
  make clean
  make
  ```

```
                    CVS/

                    instscr/

                    def_rsm/
          RSM/
                    libs/

                    rruns/

                    rsm/
```
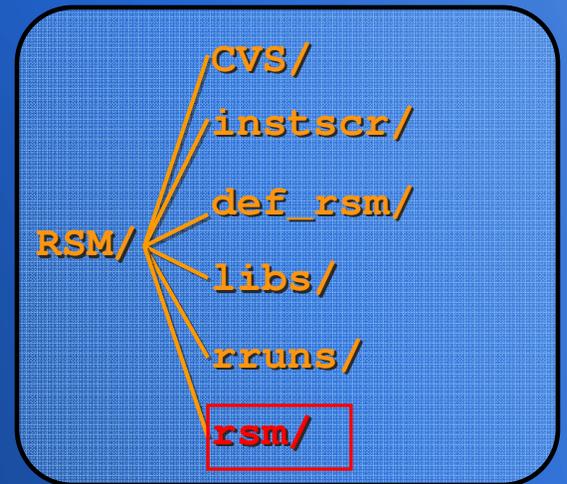
Ran RSM

(Finally!)

- In **rruns/**:
 `configure-scr rsm`

- Then, to submit a job:
  `rsm >& rsm.log &`
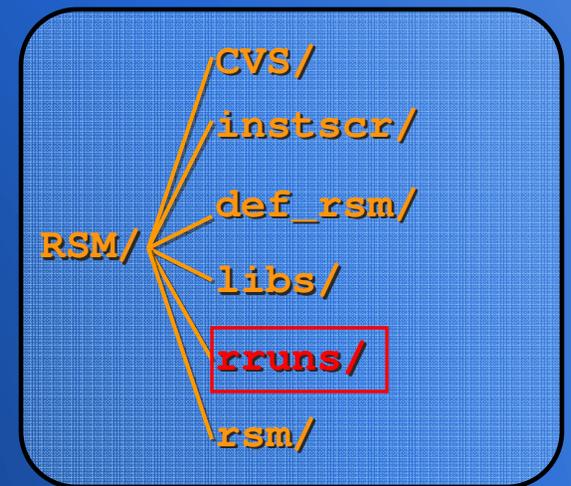
Ran RSM

Tested different numbers of cores

- In *rsm/*:
  ```
  configure-model
  make clean
  make
  ```

- In *rruns/*:
  ```
  configure-scr rsm
  ```

- Then, to submit a job:
  ```
  rsm >& rsm.log &
  ```

```
          CVS/
          instscr/
          def_rsm/
RSM/      libs/
          rruns/
          rsm/
```
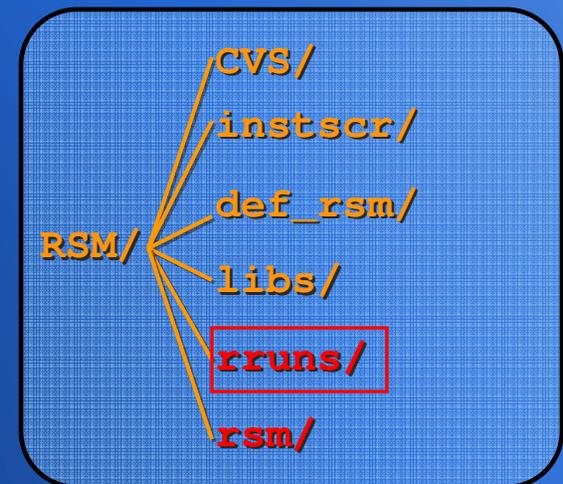
Tested RSM Runs

RSM run time
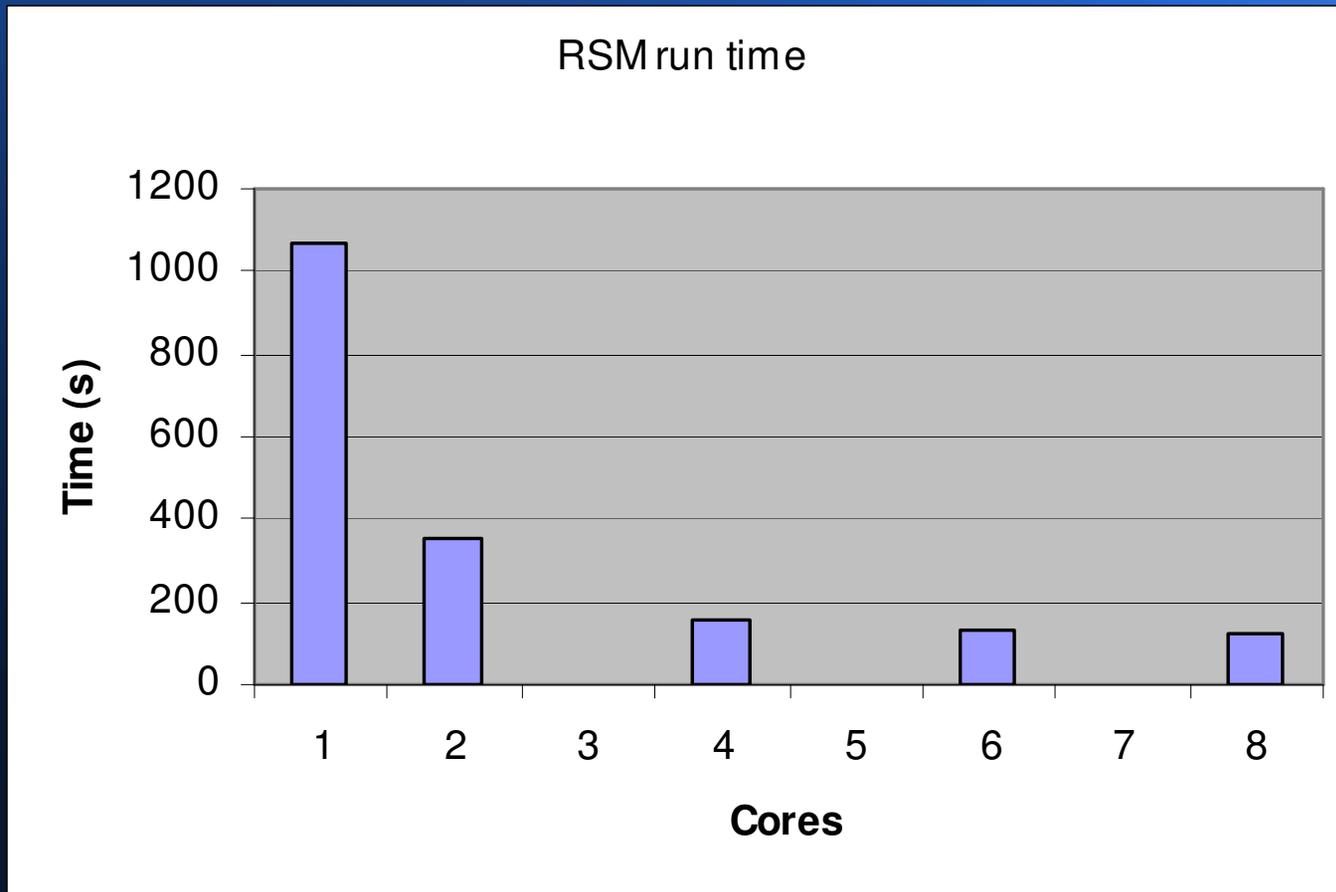
Compared memory usage
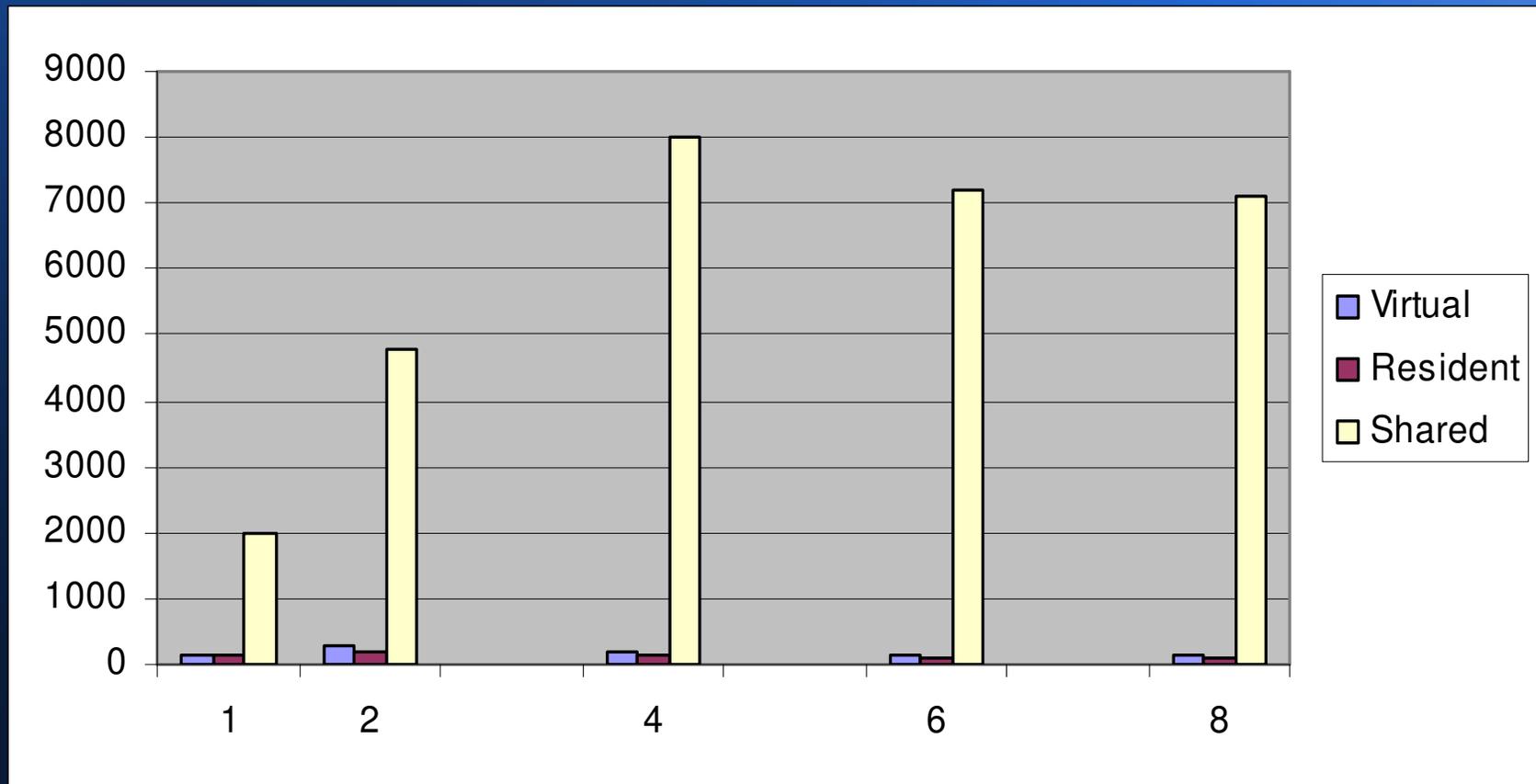
Summary notes

- Inexpensive
- Fast to set up
- *Mostly* stable
- Large Linux support community
- Using additional cores does provide shorter computation times
- A larger amount of memory may not benefit the computation times

Future Work

- Still have to benchmark against cluster

Vs.

???

- Network more than one workstation? (MPI)
- Test longer runs?