Compile and run Weather Research and Forecasting data on an IBM POWER8 system

Accelerate weather analysis for faster notification

Qiang Liu

October 28, 2014

Compile and install data for the Weather Research and Forecasting (WRF) Model and its dependent packages on the IBM POWER8 system, which provides parallel computing capabilities.

The Weather Research and Forecasting (WRF) Model is a major application that can be used to predict weather for atmospheric research and operational forecasting needs. WRF is an open source program that delivers its source code as a tar ball, so users need to compile it to binary on a different platform. WRF often runs on a high-performance computing environment with parallel computing support. The Intel® x86 platform and IBM Power® platform both enable high-performance computing.

In this tutorial, you’ll learn how to:

• Compile WRF and its dependent libraries on the IBM POWER8™ platform with the IBM XL Fortran and XL C/C++ compilers
• Run WRF on the POWER8 platform with IBM Parallel Environment™ (PE)

Operating system information and directory settings

The operating system I used for this tutorial is Red Hat Enterprise Linux Server release 6.5 (Santiago) for Power. The software and source code versions I used include:

• IBM XL Fortran for Linux v15.1
• IBM XL C/C++ for Linux v13.1
• IBM Parallel Environment 1.3.0.8
• WRF 3.5.1
• NetCDF 3.6.3
• WRF Preprocessing System (WPS) 3.5.1
• zlib 1.2.8
• JasPer 1.900.1
• libpng 1.2.34
Figure 1. Software dependencies

| WPS | WRF | NetCDF | Zlib | Jasper | libpng |

**Tips:** I recommend that you create a non-root user (for example, loadl) to do the compile and installation. Then you can compile and install the WRF and its relative libraries to the loadl home directory. Also, when you use PE to run a parallel calculation job, you must execute it as a non-root user.

**Listing 1. Definitions for this tutorial**

```
/home/loadl/project/bin --> The folder where the compiled binaries are installed.
/home/loadl/project/include --> The folder where the header files are installed.
/home/loadl/project/lib --> The folder where the libraries are installed.
/home/loadl/project/share --> The folder where the manual, document are installed.
/home/loadl/project/source --> The folder where the source code packages are saved.
/home/loadl/project/netcdf --> The folder where the compiled netcdf are installed.
   It has sub folders such as bin,include,lib,share.
```

You also need to generate an SSH key pair for user loadl, and then add the public key to the authorized keys. This enables the silent login of user loadl and is a prerequisite to run a PE job on this node.

**Install and compile the WRF model**

**Install the XL Fortran and XL C/C++ compilers on Linux**

Use root to install the XL Fortran and XL C/C++ compilers. **Note:** The following packages are required for the XL C/C++ compiler. If they aren't available, use yum to install these packages. The 32-bit packages have the `.ppc` extension. The 64-bit packages use `.ppc64`.

**Listing 2. Required packages for the XL C/C++ compiler**

```
yum install libstdc++*.ppc
yum install gcc
yum install gcc-c++
yum install glibc-devel*.ppc64
yum install glibc-devel*.ppc
yum install libstdc++-devel*.ppc64
yum install compat-libstdc++-33*.ppc
yum install compat-libstdc++-33*.ppc64
yum install ksh
```

**Install PE on Linux**

Be sure you have all the necessary packages installed first. PE requires the bind and xinetd packages. (Use yum to install the bind package, `yum install bind`, and the xinetd package, `yum install xinetd`.) Then disable SELinux by editing the `/etc/selinux/config` file and setting SELINUX=disabled.

Reboot the operating system for those changes to take effect.
Then as root, follow these steps to install the PE.

1. export IBM_PPEDEV_LICENSE_ACCEPT=yes
2. rpm -i ppedev_pa_license-1.3.0-0.ppc64.rpm
3. rpm -i ppedev_pa_runtime-1.3.0-0.ppc64.rpm
4. rpm -i ppedev_pa_hpct-1.3.0-0.ppc64.rpm
5. export IBM_PPE_RTE_LICENSE_ACCEPT=yes
6. rpm -i ppe_rte_license-1.3.0.8-s008a.ppc64.rpm
7. rpm -i ppe_rte_1308-1.3.0.8-s008a.ppc64.rpm

At the end of this package installation, the system prompts you for recommended system setting changes. Make these changes as root. Starting the node setting diagnostic script:

**Issue 1:** net.core.wmem_max is 124928, but 1048576 is recommended.
```
sysctl -w net.core.wmem_max=1048576
```

**Issue 2:** net.core.rmem_max is 124928, but 8388608 is recommended.
```
sysctl -w net.core.rmem_max=8388608
```

**Issue 3:** net.ipv4.ipfrag_low_thresh is 196608, but 1048576 is recommended.
```
sysctl -w net.ipv4.ipfrag_low_thresh=1048576
```

**Issue 4:** net.ipv4.ipfrag_high_thresh is 262144, but 8388608 is recommended.
```
sysctl -w net.ipv4.ipfrag_high_thresh=8388608
```

**Issue 5:** ulimit for nofile is 1024, but 4096 is recommended.
Update nofile to be 4096 in /etc/security/limits.conf

**Issue 6:** ulimit for memlock is 64, but unlimited is recommended.
Update memlock to be unlimited in /etc/security/limits.conf

Restart xinetd with service xinetd restart.

**Issue 7:** per_source is 10 in /etc/xinetd.conf, but 80 is recommended.
Change per_source to 80 in /etc/xinetd.conf

Restart xinetd with service xinetd restart.

According to the ulimit relative setting, set ulimit -n 4096 and ulimit -l unlimited in the /etc/profile. Then restart the xinetd service:
**Listing 3. Restarting the xinetd service**

```
rpm -i ppertes-1.3.0.8-s008a.ppc64.rpm
rpm -i ppe_rte_samples-1.3.0.8-s008a.ppc64.rpm
rpm -i ppertesamples-1.3.0.8-s008a.ppc64.rpm
```

**Compile NetCDF**

First, log in as `loadl` to do the next steps. Before you can compile NetCDF, you need to enable the system environment variables:

**Listing 4. System environment variables**

```
export CC="xlc_r"
export CFLAGS="-q64 -O3 -qstrict -DIBMR2Fortran"
export CPP="xlc_r -E"
export CXX="xlc_r"
export CXXFLAGS="-q64 -O3 -qstrict"
export CXXCPP="xlc_r -E"
export F77="xlf_r"
export FC="xlf_r"
export F90="xlf90_r"
export FFLAGS="-q64 -O3 -qstrict"
```

Use `xlc_r` as the C/C++ compiler and `xlf_r` as the Fortran compiler. Use this environment variable setting for all the compiling from NetCDF (and other relative libraries) to WRF. Doing so makes the WRF-dependent libraries compatible. Otherwise, when you call the dependent libraries during the WRF compile, you get incompatibility errors.

To specify the target folder where the compiled output should be installed, use this code:

**Listing 5. Install compiled output in target folders**

```
./configure --prefix=/home/loadl/project/netcdf
make
make install
```

**Compile zlib**

Follow the steps shown in this listing:

**Listing 6. Compile zlib**

```
./configure --prefix=/home/loadl/project
make
make install
```

**Compile JasPer**

Follow the steps shown in this listing:

**Listing 7. Compile JasPer**

```
./configure --prefix=/home/loadl/project
make
make install
```
Compile libpng

Follow the steps shown in this listing:

Listing 8. Compile libpng

```bash
export LD_LIBRARY_PATH=/home/loadl/project/lib/:$ LD_LIBRARY_PATH
export LDFLAGS="-L/home/loadl/project/lib/ -L/home/loadl/project/netcdf/lib"
export CPPFLAGS="-I/home/loadl/project/include/ -I/home/loadl/project/netcdf/include"
./configure --prefix=/home/loadl/project
make
make install
```

Compile WRF

As a prerequisite to compiling WRF, you need to enable the environment variables:

Listing 9. Enable environment variables for WRF compilation

```bash
export JASPERLIB=/home/loadl/project/lib
export JASPERINC=/home/loadl/project/include
export WRF_EM_CORE=1
export NETCDF=/home/loadl/project/netcdf
```

To enable the parallel computing capabilities of WRF, use `xlc_r` and `xlf95_r` with `-lmpi` to compile.

Extract the `WRFV3.5.1.TAR.gz` and run the configure script first, because that helps you create the `make` file. It generates a `configure.wrf` file, which contains a lot of variable definitions and is read during the `make`.

Run `configure`. At the prompt, select from among the supported platforms:

1. Linux ppc64 BG /L blxlf compiler with blxlc (dmpar)
2. Linux ppc64 BG /P xlf compiler with xlc (smpar)
3. Linux ppc64 BG /P xlf compiler with xlc (dmpar)
4. Linux ppc64 BG /P xlf compiler with xlc (dm+sm)
5. Linux ppc64 IBM Blade Server xlf compiler with xlc (dmpar)

Because POWER8 is new, the configure script recognizes only BG (BlueGene) as the CPU, but this doesn't affect the compiling result. Select the third option, **Linux ppc64 BG /P xlf compiler with xlc (dmpar)**.

You'll then get a prompt:

Listing 10. Prompt from "compiling for nesting" in WRF compiling configuration

```bash
Compile for nesting?
  1=basic
  2=preset moves
  3=vortex following
[default 1]
```
Select 1 basic, the default. It generates a configuration template file for your edition. Edit the configure.wrf file. Update the fields as shown:

**Listing 11. Updating fields in the configuration file**

```bash
SFC = xlf95_r -q64 -I/opt/ibmhpc/pecurrent/mpich2/gnu/include64 -L/opt/ibmhpc/pecurrent/mpich2/gnu/fast/lib64
SCC = xlc_r -q64 -I/opt/ibmhpc/pecurrent/mpich2/gnu/include64 -L/opt/ibmhpc/pecurrent/mpich2/gnu/fast/lib64
CCOMP = xlc_r -q64 -I/opt/ibmhpc/pecurrent/mpich2/gnu/include64 -L/opt/ibmhpc/pecurrent/mpich2/gnu/fast/lib64
DM_FC = xlf95_r -q64 -I/opt/ibmhpc/pecurrent/mpich2/gnu/include64 -L/opt/ibmhpc/pecurrent/mpich2/gnu/fast/lib64 -lmpi
DM_CC = xlc_r -q64 -I/opt/ibmhpc/pecurrent/mpich2/gnu/include64 -L/opt/ibmhpc/pecurrent/mpich2/gnu/fast/lib64 -DMPI2_SUPPORT -lmpi
CPP = /opt/ibm/xlf/15.1.0/exe/cpp -C -P
```

Then execute `./compile em_real >& compile.log`, which compiles and writes the compiling log to the `compile.log` for debugging. If the compile succeeds, an exe generates:

**Listing 12. An exe generates**

```bash
-bash-4.1$ ll run/*.exe
lrwxrwxrwx 1 loadl loadl 17 Aug 22 07:55 run/ndown.exe -> ../main/ndown.exe
lrwxrwxrwx 1 loadl loadl 15 Aug 22 07:55 run/nup.exe -> ../main/nup.exe
lrwxrwxrwx 1 loadl loadl 16 Aug 22 07:55 run/real.exe -> ../main/real.exe
lrwxrwxrwx 1 loadl loadl 14 Aug 22 07:55 run/tc.exe -> ../main/tc.exe
lrwxrwxrwx 1 loadl loadl 15 Aug 22 07:53 run/wrf.exe -> ../main/wrf.exe
```

**Compile WPS**

Extract the `wpsv3.5.1.tar.gz` and run the configure script to create the `make` file. Run the configure script. You are then prompted to make a choice:

**Listing 13. Results of WRF compiling the configuration script**

Please select from among the following supported platforms.

1. Linux ppc64 Power775 xl compilers & MPICH2 comms (serial)
2. Linux ppc64 Power775 xl compilers & MPICH2 comms (serial_NO_GRIB2)
3. Linux ppc64 Power775 xl compilers & MPICH2 comms (dmpar)
4. Linux ppc64 Power775 xl compilers & MPICH2 comms (dmpar_NO_GRIB2)
5. Linux ppc64 BG bglxf compiler with blxlc (dmpar)

Enter selection [1-5]:

Select 3. Linux ppc64 Power775 xl compilers & MPICH2 comms (dmpar).

WPS3.5.1 still can’t recognize the latest POWER8 system. It generates a file named `configure.wps`, which contains variable definitions and is read during the make. Modify the WRF relative path value in `configure.wps`.

```bash
WRF_DIR = ../../../wrf_arw/WRFV3
```

**Note:** Leave no space after `wrf_arw/WRFV3`. Doing so results in an error.
Modify this field as shown:

**Listing 14. Modify field**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPP</td>
<td>/opt/ibm/xlf/15.1.0/exe/cpp -C -P</td>
</tr>
<tr>
<td>MPICH2_SYS</td>
<td>/opt/ibmhpc/pecurrent/mpich2/gnu</td>
</tr>
<tr>
<td>MPI_INC</td>
<td>-I$(MPICH2_SYS)/include64</td>
</tr>
<tr>
<td>MPI_LIB</td>
<td>-L$(MPICH2_SYS)/lib64 -lmpi</td>
</tr>
<tr>
<td>FC</td>
<td>xlf95_r</td>
</tr>
<tr>
<td>SFC</td>
<td>xlf95_r</td>
</tr>
<tr>
<td>CC</td>
<td>mpicc -DMPI2_SUPPORT -DFSEEK064_OK</td>
</tr>
<tr>
<td>CPPFLAGS</td>
<td>-DAIX -DIBM4 -DIO_NETCDF -DIO_BINARY -DIO_GRIB1 -DBIT32 -D_MPI</td>
</tr>
</tbody>
</table>

Save the change and execute `./compile > &compile.log` to kick off the compiling and write the log into the `compile.log` file. After WPS compiles, this exe file is generated:

**Listing 15. Exe is generated after compilation succeeds**

```
[load1@tul237p5 WPS]$ ll *.exe
lrwxrwxrwx 1 loadl loadl 23 Aug 12 00:37 geogrid.exe -> geogrid/src/geogrid.exe
lrwxrwxrwx 1 loadl loadl 23 Aug 12 00:37 metgrid.exe -> metgrid/src/metgrid.exe
lrwxrwxrwx 1 loadl loadl 21 Aug 12 00:37 ungrib.exe -> ungrib/src/ungrib.exe
```

**Run WRF on POWER8**

To run WRF, you need to prepare the WRF input data, which requires WPS to preprocess the weather input data. Use `geogrid.exe` to generate the static terrestrial data. Use `ungrib.exe` to unpack the weather input GRIB meteorological data. Use `metgrid.exe` to generate the input for WRF, which will interpolate the meteorological data horizontally onto the model domain. Use `real.exe` to interpolate the data onto the model coordinates vertically.

**Figure 2. The WRF data process**

```
ungebr.exe

geogrid.exe

metgrid.exe

real.exe

wrf.exe
```

`ungebr.exe`, `geogrid.exe`, and `metgrid.exe` are executed in a single process. `real.exe` and `wrf.exe` are executed in parallel mode. That means that PE executes both `real.exe` and `wrf.exe` in interactive mode.

Here’s an example of how to run WRF in interactive mode with PE.

On a POWER8 system with **SMT4** enabled, the CPU number is 4x of the core number. I recommend that you use `nmon` to monitor the system resource and usage as shown in this example for a 20-core POWER8 system:
The POWER8 chip is so new that nmon can't recognize its model correctly, but it can read the CPU frequency (4.1GHz) and SMT value (SMT=4).

With `MP_TASK_AFFINITY=core` set, the maximum number of parallel tasks is the total core number. So on my 20-core POWER8 environment, the number is 20.

With `MP_TASK_AFFINITY=cpu` set, the maximum number of parallel tasks is the total CPU number. So on my 20-core POWER8 (SMT=4) environment, the number is 80.

The command `poe` is the main executable of PE and is used to call the parallel computing tasks. Poe needs a hostfile, which is used to describe the total CPU resource available on this host. The hostfile could be named `host.list`. You can describe the resource as `<hostname>*number each line. For example:

```
myhost01*20
myhost02*10
```

In this case, I set the hostfile content as `myhost*80`.

Then use this `poe` command to kick off the WRF parallel calculation.

```
poe /home/loadl/project/wrf.exe -hostfile /home/loadl/project/host.list -procs 78
```

`-procs 78` means a total of 78 parallel tasks are initiated. This number must be less than the total resource number described in the hostfile.
If you set `MP_TASK_AFFINITY=core` and still try to initiate 78 parallel tasks, you'll get an error like these:

ERROR: 0031-758 AFFINITY: Oversubscribe: 78 tasks in total, each task requires 1 resource, but there are only 20 available resource. Affinity can not be applied.

ERROR: 0031-161 EOF on socket connection with node myhost

I did some tests against one WRF domain model with different settings on my 20-core POWER8 box. I found that:

- Setting `MP_TASK_AFFINITY=cpu` and `procs=78` gives the best performance. When WRF is running, the CPU usage looks like this:
Figure 4. CPU usage with settings `MP_TASK_AFFINITY=cpu` and `procs=78`

![CPU Utilization](image)

- Set `MP_TASK_AFFINITY=cpu` and `procs=80`. Although this uses up all the CPU resources, the performance is worse because there is no delegated CPU resource to manage the I/O. This affects the overall performance.

- Setting `MP_TASK_AFFINITY=core` and `procs = 20` is good, but not as good as when the setting is `MP_TASK_AFFINITY=cpu`. When WRF is running, the CPU usage look like this output:
Figure 5. CPU usage with settings MP_TASK_AFFINITY=core

Conclusion

This tutorial shows how to compile and run the basic Weather Research and Forecasting (WRF) model on the POWER8 system. The XL C/Fortran compiler and POWER8 CPU support the WRF parallel calculation. By using the POWER8 CPU, you can shorten the process of forecasting the weather and create more business value.
Related topics

- IBM Power Systems
- Weather Research and Forecasting (WRF) Model
- WRF Preprocessing System (WPS)
- JasPer, a toolkit for handling image data

© Copyright IBM Corporation 2014
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)