

AlphaFold Workshop

Lesson 6

Revision 2

Petr Kulhánek

kulhanek@chemi.muni.cz

Laboratory of Computational Chemistry
National Centre for Biomolecular Research
Faculty of Science
Masaryk University
Kamenice 5
CZ-62500 Brno

Infinity

Infinity

Infinity is an environment for managing software and jobs on computational clusters and supercomputers. It was developed at Laboratory of Computational Chemistry.

It has a long history (~ 20 years). The original version was written in bash. Later it was rewritten several times in C/C++.

While the implementation changed significantly, **the original idea, philosophy, and typical usage remained unchanged.**

Technical issues are handled by Infinity.

The user focuses on the job specification.

Native Approach vs Infinity

Native usage in MetaCentrum

```
#!/bin/bash

#ensure removing the temporary data if the job ends or fails
trap "clean_scratch" TERM EXIT

DATADIR="/storage/brno2/home/$LOGNAME"
JOBNAME="myjob"      # myjob.com -> myjob.log

# sanity checks
if [[ -z "$SCRATCHDIR" ]]; then
    echo "use scratch_local, scratch_ssd, or scratch_shared in qsub (resource specification)"
    exit 1
fi

if [[ ! (-f "$DATADIR/${JOBNAME}.com") ]]; then
    echo "the input file '$DATADIR/${JOBNAME}.com' does not exist"
    exit 1
fi

# copy input file from shared network disk to local disk
cp $DATADIR/${JOBNAME}.com $SCRATCHDIR/ || exit 1
cd $SCRATCHDIR/ || exit 2

# let's load the Gaussian module
module add g09

# myjob.com is the input file
# setup the resource requirements within the input file so that they correspond to the resources reserved
g09-prepare ${JOBNAME}.com

# start the computation (use g16 instead of g09 for the g16 version) , myjob.log will be the output file
g09 ${JOBNAME}.com
# alternatively: g09 < ${JOBNAME}.com > ${JOBNAME}.log

# copy the output from local scratch to shared network disk
cp ${JOBNAME}.log $DATADIR/ || export CLEAN_SCRATCH=false
```

37 lines

Employing Infinity

```
#!/usr/bin/env infinity-env
module add gaussian:16.C1
psanitize myjob.com
g16 myjob
```

4 lines

Native Approach vs Infinity

Native usage in MetaCentrum

```
#!/bin/bash

#ensure removing the temporary data if the job ends or fails
trap "clean_scratch" TERM EXIT

DATADIR="/storage/brno2/home/$LOGNAME"
JOBNAME="myjob" # myjob.com -> myjob.log

# sanity checks
if [[ -z "$SCRATCHDIR" ]]; then
    echo "use scratch_local, scratch_ssd, or scratch_shared in qsub (resource specification)"
    exit 1
fi

if [[ ! (-f "$DATADIR/${JOBNAME}.com") ]]; then
    echo "the input file '$DATADIR/${JOBNAME}.com' does not exist"
    exit 1
fi

# copy input file from shared network disk to local disk
cp $DATADIR/${JOBNAME}.com $SCRATCHDIR/ || exit 1
cd $SCRATCHDIR/ || exit 2

# let's load the Gaussian module
module add g09

# myjob.com is the input file
# setup the resource requirements within the input file so that they correspond to the resources reserved
g09-prepare ${JOBNAME}.com

# start the computation (use g16 instead of g09 for the g16 version) , myjob.log will be the output file
g09 ${JOBNAME}.com
# alternatively: g09 < ${JOBNAME}.com > ${JOBNAME}.log

# copy the output from local scratch to shared network disk
cp ${JOBNAME}.log $DATADIR/ || export CLEAN_SCRATCH=false
```

37 lines

Employing Infinity

```
#!/usr/bin/env infinity-env
module add gaussian:16.C1
psanitize myjob.com
g16 myjob
```

4 lines

activate the software

adjust the input file for
requested computational
resources (the speciality of
Gaussian software)

run the calculation

Infinity Subsystems

Infinity is composed of two subsystems:

- **AMS** - **Advanced Module System for software management**
- **ABS** - Advanced Batch System for job management

AMS - Advanced Module System

Scientific and technical applications need to be installed in several versions.

- We need older versions to finish old projects.
- New versions are required to start new projects employing new features available.

Management of several versions of the same application is impossible with the standard package systems. These systems support only the installation of the newest application.

The version problem can be overcome by a module system. There are several implementations available:

- Environment Modules (MetaCentrum)
- Lmod (IT4I)
- AMS (Infinity: WOLF, SOKAR, optionally MetaCentrum)

AMS - Advanced Module System

Scientific and technical applications need to be installed in several versions.

- We need older versions to finish old projects.
- New versions are required to start new projects employing new features available.

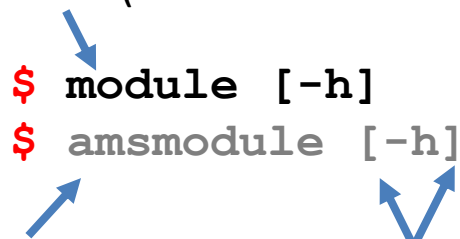
Management of several versions of the same application is impossible with the standard package systems. These systems support only the installation of the newest application.

The version problem can be overcome by a module system. There are several implementations available:

- Environment Modules (MetaCentrum)
- Lmod (IT4I)
- AMS (Infinity: WOLF, SOKAR, optionally MetaCentrum)

the main command (as alias to amsmodule)

```
$ module [-h]  
$ amsmodule [-h]
```



the AMS main command

AMS supports autocompletion!

[] indicates that "-h" is an optional option.
If specified, then the command prints a simple help page.

Modules - List available modules

\$ module

\$ module [avail]

```
[kulhanek@sokar ~]$  
[kulhanek@sokar ~]$ module  
  
AVAILABLE MODULES (Infinity Software Base | amsmodule)  
  
--- GPU Enabled Software ---  
alphafold          colabfold          gromacs  
alphapulldown     fair-esm           pmemd-cuda  
  
--- Molecular Mechanics and Dynamics ---  
amber              delphi             pmemd-cuda         red  
apbs               dynutil           pmemd-pmf          sander-pmf  
balloon           gromacs           pmflib             solvate  
cicada            mmpbsa-py         raspa              unres  
  
--- Quantum Mechanics and Dynamics ---  
aimall             clumo             gaussview          nbo                respect  
adf               cm5pac           gennbo            nciplot           siesta  
amber2gromos      dftb+            horton            orca               turbomole  
atom              dftd3            inelastica        postg  
bader             gamess-cina      mag               promolden  
chargemol         gamess-us        molpro            psi4  
cfour             gaussian          multiwfn          qmutil  
  
--- Docking and Virtual Screening ---  
adfr-suite        autodock-vina     dock               mgltools  
autodock          cheminfo          meeko             xscore  
  
--- Chemo & Bioinformatics ---  
annovar           clustalw          gmap-gsnap        pindel            samtools  
bedtools          colabfold         htlib             prinseq           seqtk  
bcftools          copasi            i-tasser          pysam             shrimp  
blast             cutadapt          igv               R                 snpeff  
blast+           dproperties       igvtools          rate4site         sratoolkit  
bowtie           dragon            mc-annotate       reaper            stampy  
breakdancer      fastqc            modeller          rcount           stride  
bwa              fastx_toolkit     muscle            rna-seqc          subread  
caver-analyst    gatk              ngsutils          rnaview           tabix  
cd-hit           genomertools     picard-tools      rseqc             vcftools  
  
--- Protein Structure Prediction ---  
alphafold         alphafold-tools   colabfold         modeller  
alphafold-conda  alphapulldown     fair-esm          omegafold
```

software is categorized

software for protein structure prediction



Modules - Work with a module

Overview of available module versions:

```
$ module versions pymol
```

```
pymol:2.5.0.0
```

```
pymol:2.2.0.0
```

← default version is bold

Module documentation:

```
$ module help pymol
```

Module activation:

Activate the default version.

Activate specified module version.

The version is delimited from the module name by a colon.

```
$ module add pymol
```

```
$ module add pymol:2.2.0.0
```

Run the application from the `pymol` module

```
$ pymol
```

← The module name and application name do not have to be the same

Exercise 1

1. In your home directory, create the "AlphaFold" directory. Then, create the "L06.pymol" subdirectory.

```
$ mkdir -p ~/AlphaFold/L06.pymol
$ cd ~/AlphaFold/L06.pymol
$ pwd
/home/kulhanek/AlphaFold/L06.pymol
```

2. In a web browser, visit the PDB database and download the PDB and FASTA files for PDB ID **3C1E** into the AlphaFold directory.
3. Ensure files are stored in the directory.

```
[kulhanek@wolf L06.pymol]$ ls -l
total 136
-rw-r----- 1 kulhanek ncbr 131463 May 26 13:59 3c1e.pdb
-rw-r----- 1 kulhanek ncbr    204 May 26 13:59 rcsb_pdb_3C1E.fasta
```

Exercise II

1. Open the 3C1E structure in PyMol.

```
$ module add pymol  
$ pymol 3c1e.pdb
```

2. Show the structure in cartoon and line models simultaneously.

In the Infinity software base, the following visualization software is available:

- VMD (module help vmd)
- PyMOL (module help pymol)
- Chimera (module help chimera)
- ChimeraX (module help chimerax)

Use whatever is conformable for you.