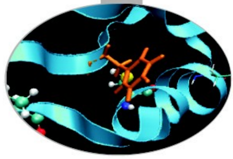
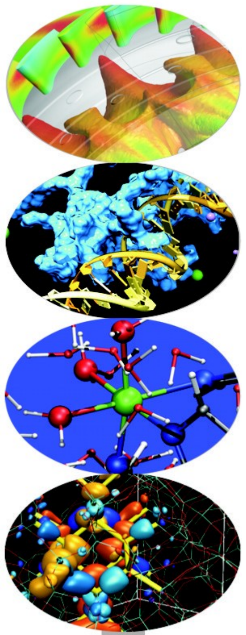
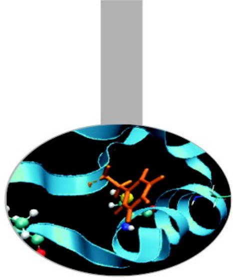


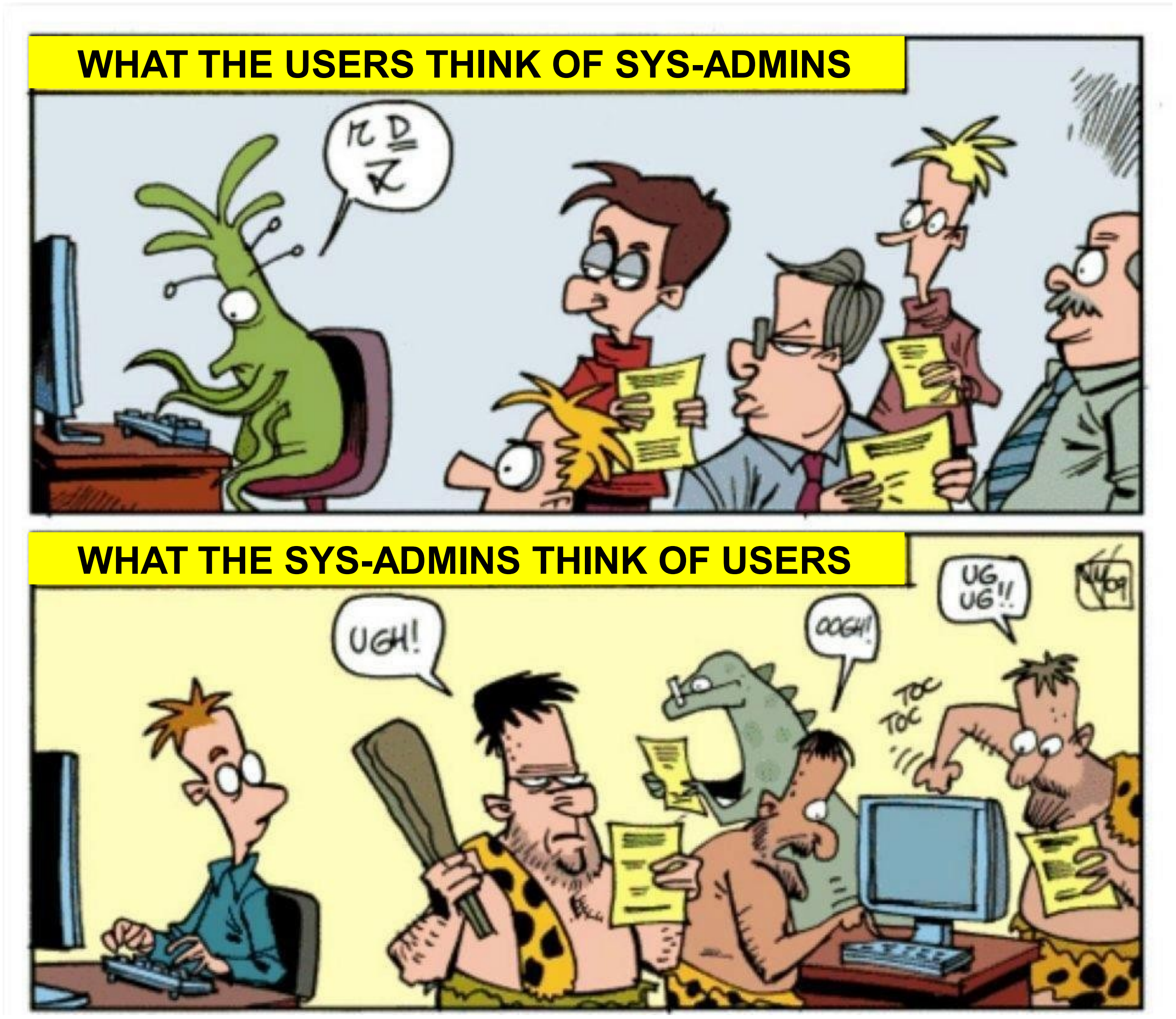
Development Environment on BG/Q FERMI

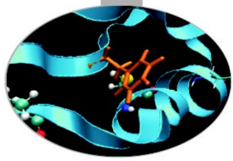


Nicola Spallanzani
n.spallanzani@cineca.it
www.hpc.cineca.it

USER SUPPORT
superc@cineca.it



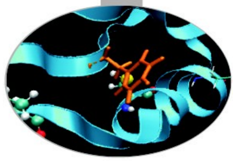

WHAT THE USERS THINK OF SYS-ADMINS
WHAT THE SYS-ADMINS THINK OF USERS
UGH!
OOGH!
UG UG!!
TOC TOC



Outline

- A first step
    - ssh
    - file transfer
- Introduction to the environment
    - module command
    - module usage
- Programming environment
    - cross - compilation
    - available compilers
    - optimization with XL
    - examples
    - libraries, compiling/linking issues
- For further info...
    - useful links and documentation

FERMI: how to login



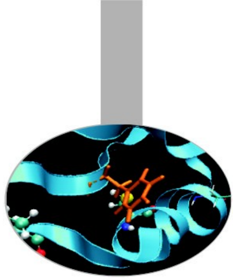
- Establish a ssh connection

  ```
  ssh <username>@login.fermi.cineca.it
  ```

- Remarks:
  - **ssh** available on all linux distros
  - **Putty** (free) or **Tectia** ssh on Windows
  - *secure shell plugin* for Google Chrome!
  - login nodes are swapped to keep the load balanced
  - important messages can be found in the *message of the day*

- Check the **user guide**! http://www.hpc.cineca.it/content/hpc-user-guide

FERMI: File transfer



- **sftp / scp** (always available if sshd is running)

```
$ sftp <user>@login.fermi.cineca.it:/path/to/
$ scp -r <my_dir>  <user>@login.fermi.cineca.it:/path/to/
```

- **rsync**: allows incremental transfer

```
$ rsync -azP <my_dir> <user>@login.fermi.cineca.it:/path/to/
```

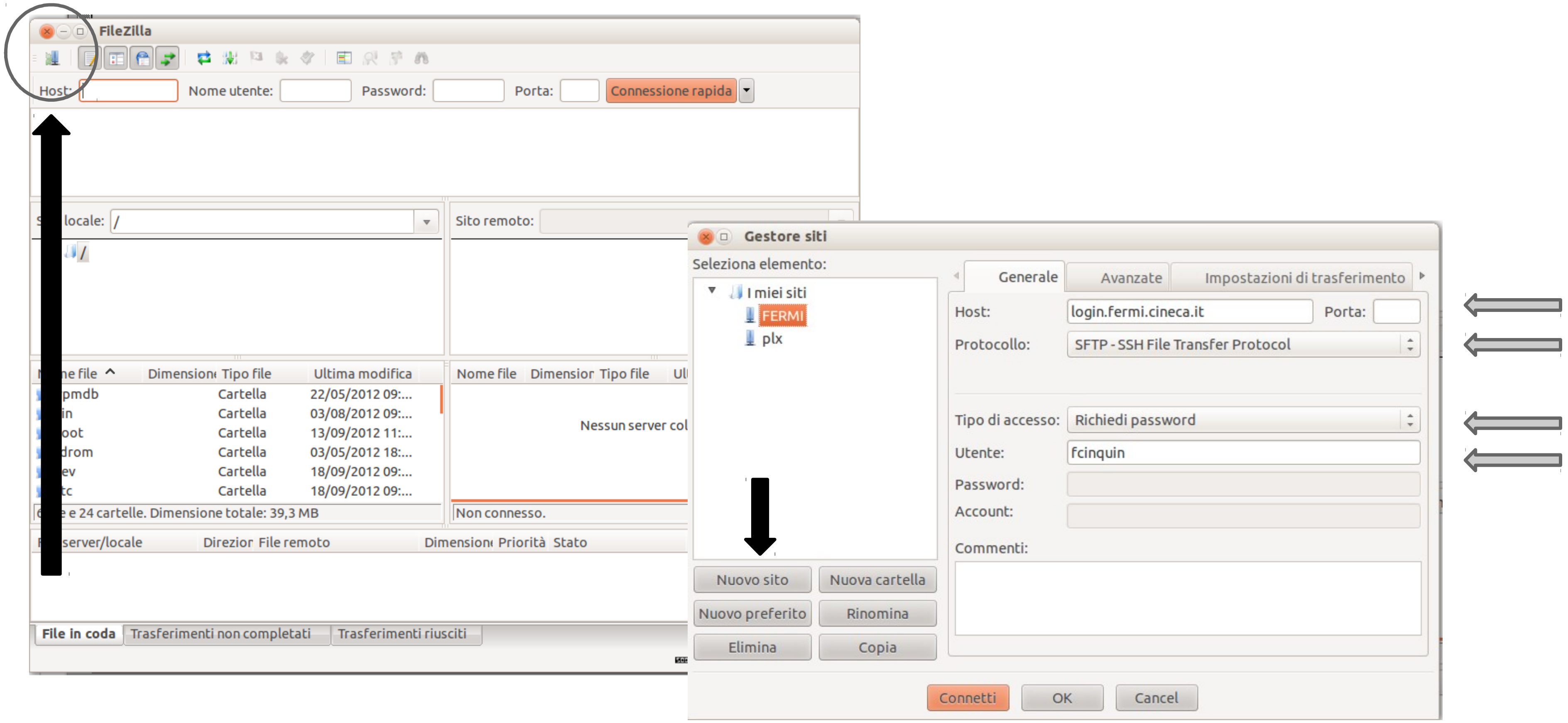
- **gridftp**: allows for stream transfer and much more
  (~10x transfer!)

```
globus-url-copy -vb -r -p 16 -sync -sync-level 2 \
                file:/path/to/files/ \
                sshftp://user@login.fermi.cineca.it/path/to/
```

http://www.hpc.cineca.it/content/data-transfer

FERMI: file transfer (2)

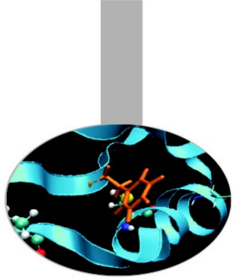
filezilla: free, open source (S)FTP client: *http://filezilla-project.org/*



Host:	login.fermi.cineca.it
Porta:	
Protocollo:	SFTP - SSH File Transfer Protocol
Tipo di accesso:	Richiedi password
Utente:	fcinquin
Password:	
Account:	
Commenti:	

simple, browse your local and remote directories, drag & drop

"module", my best friend

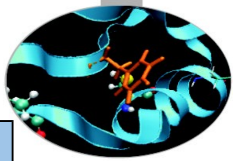


- all the optional software on the system is made available through the **"module" system**
  - provides a way to rationalize software and its env variables

- modules are divided in 3 *profiles*
  - **profile/base** (stable and tested modules for back-end)
  - **profile/front-end** (stable and tested modules for front-end)
  - **profile/advanced** (software not yet tested or not well optimized)

- each profile is divided in 4 *categories*
  - **compilers** (IBM xl, gnu)
  - **libraries** (e.g. LAPACK, BLAS, FFTW, ...)
  - **tools** (e.g. Scalasca, GNU make, VNC, ...)
  - **applications** (software for chemistry, physics, ...)

module env



COMMAND	DESCRIPTION
module av	list all the available modules
module load <module_name(s)>	load module <module_name>
module list	list currently loaded modules
module purge	unload all the loaded modules
module unload <module_name>	unload module <module_name>
module help <module_name>	print out the help (hints)
module show <module_name>	print the env. variables set when loading the module

NB: some modules rely on other modules

```
$ module load boost

WARNING: boost/1.51.0--bgq-xl--1.0 cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first: bgq-xl/1.0
```

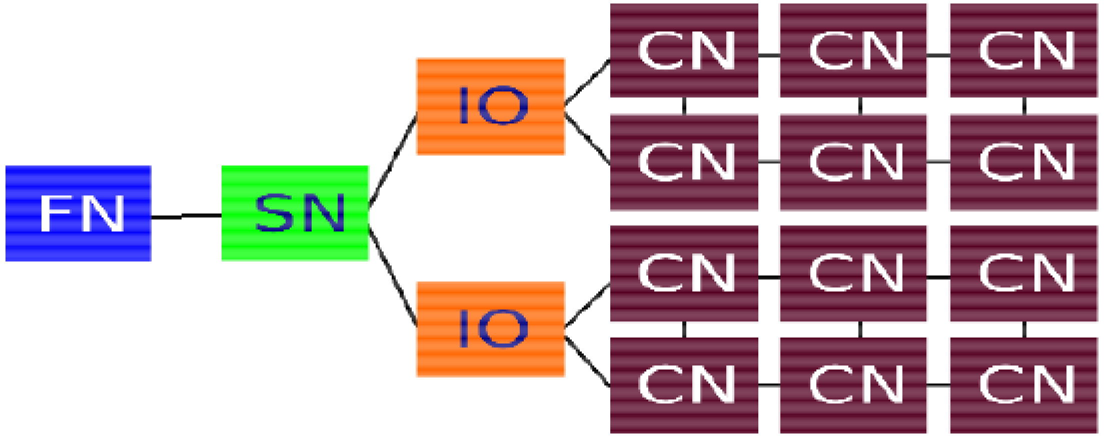
Load the "autoload" module which check the dependencies and load all the modules needed:

```
$ module load autoload boost
```

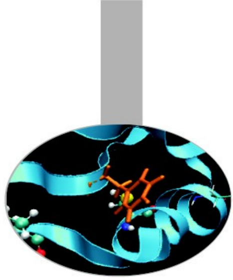
So far so good...
... BUT

Blue Gene Blocks Hierarchical Organization

- **Front-end nodes (FN)**, dedicated for user's to login, compile programs, submit jobs, query job status, debug applications
- **Service nodes (SN)**, perform system management services, create and monitoring processes, initialize and monitor hardware, configure partitions, control jobs, store statistics
- **I/O nodes (IO)**, provide a number of OS services, such as files, sockets, process management, debugging
- **Compute nodes (CN)**, run user application, limited OS services



Cross-Compilation

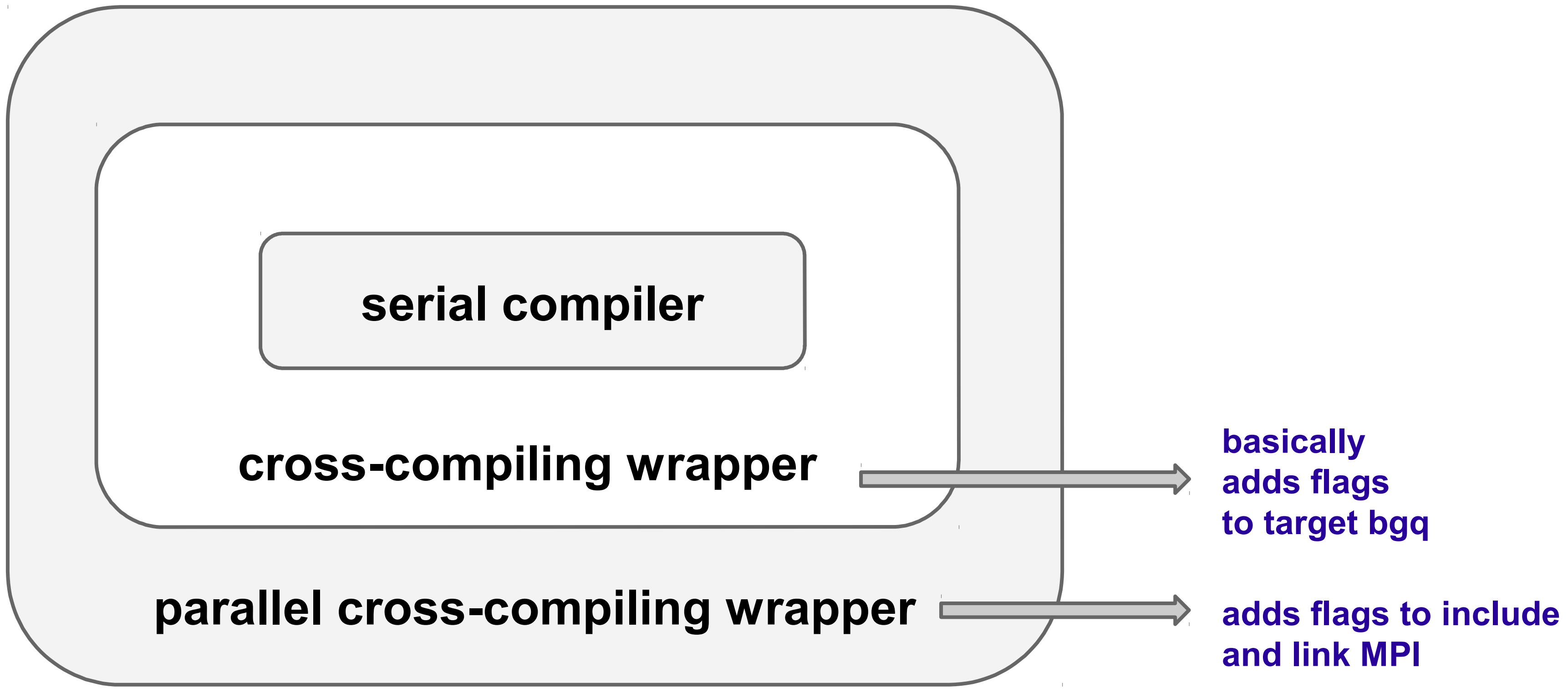


- Architectures of **compute nodes (back-end)** and **login nodes (front-end)** are different

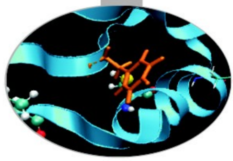
```
$ cat /proc/cpuinfo
```

- ... And you cannot login to compute node
- you need to **compile on the login nodes targeting the compute node** architecture **(*cross-compiling*)**
- you can rely on the available **wrappers** (they do all the dirty work for you)
- it's only matter of **picking the right wrapper**

Compiler Hierarchy


serial compiler
cross-compiling wrapper
parallel cross-compiling wrapper
basically adds flags to target bgq
adds flags to include and link MPI

Compiler Families



Two Different compilers family for both front-end and back-end nodes

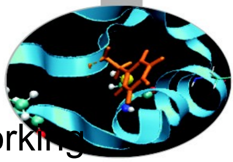
- **IBM Compilers**
- **GNU Compilers**

	Back-end Compilers		**Front-end Compilers**	
	XL family	GNU family	XL family	GNU family
C	bgxlc, mpixlc_r	gcc, mpicc	xlc	gcc
C++	bgxlc++, mpixlcxx	g++. mpicxx	xlc++, xlC	g++
Fortran	bgxlf,bgxlf90,… mpixlf90,…	gfortran, mpif90	xlf, xlf90,…	gfortran

Cross compilation:
mpixlc –O3 **–qarch=qp –qtune=qp** myprog.c

Compilation:
xlc –q64 myprog.c

... A step backward
Parallel Programming



Parallel programming is a programming technique that involves the use of *multiple processors* working together *on a single problem*. The global problem is split in different sub-problems, each of which is performed by a different processor in parallel. The code needs a programming language that allows to formally describe non-sequential algorithms. ... And of course you need the right machine architecture!

Parallel Program : different tasks communicate with each other to achieve an overall computational target.

Process

- Algorithm : the sequence of logical steps that must be followed to solve a given problem.
- Program : implementation of the algorithm, by means of a suitable formalism (programming language) so that it can be executed on a specific computer.
- Sequential process : sequence of events (execution of operations) which gives place the computer when operates under the control of a particular program.

A **task** is a Unix process.

TASKS...

- ... have their own data space
- ... run a single instance of a serial application or a *Message Passing* Interface (MPI) application.
- ... can belong to different users
- ... can be different programs that a single user is running concurrently

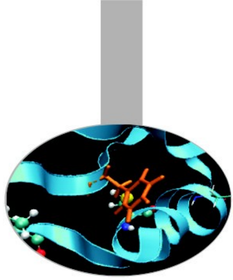
A **thread** is an independent instruction stream, but as part of a Unix process.

THREADS...

- ... use a *Shared Memory Paradigm* (ex. openMP)
- ... can have private data, but they can collaborate on the same data.
- ... are part of **one process** and therefore share each other's data.

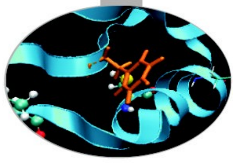
CINECA SCAI SuperComputing Applications and Innovation

Compiler Families (2)
IBM XL common options



Option	Meaning
-qarch=qp	Produces object code for the BGQ platform and: Enables BGQ vector data type Sets the –qsimd=auto option
-qtune=qp	Default with –qarch=qp or without –qarch, -qtune options and bg-prefixed compilers. Also set if specifying –q64 or –O4,-O5
-q64	Sets 64-bit compiler mode
-qstaticlink	The compiler links only static libraries with the object file being produced. (Enabled by default; specify -qnostaticlink to dynamically link your programs).
-qtm	Enables support for transactional memory Default is –qnotm To use with thread-safe compilation
-qsimd	-qsimd=auto enables automatic generation of QPX vector instructions. Enabled by default at all optimization levels. To disable automatic generation of QPX instructions, use -qsimd=noauto.
-qsmp	Enables parallelization of program code. -qsmp=omp enables strict openMP compliance. The -qsmp option must be used together with thread-safe compiler invocation modes (_r-suffixed)

Example



COMPILING...

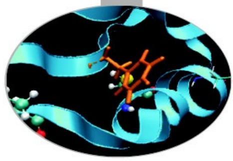
```
$ module purge
$ module load bgq-xl/1.0 lapack/3.4.1--bgq-xl--1.0
$ module list
Currently Loaded Modulefiles:
 1) profile/base                 3) lapack/3.4.1--bgq-xl--1.0
 2) bgq-xl/1.0
$ mpixlf90_r -o check.x check.f90 -L$LAPACK_LIB -llapack
** checkmpi   === End of Compilation 1 ===
1501-510  Compilation successful for file check.f90.
```

... WITH OPTIONS

```
$ module purge
$ module load bgq-xl/1.0 lapack/3.4.1--bgq-xl--1.0
$ module list
Currently Loaded Modulefiles:
 1) profile/base                 3) lapack/3.4.1--bgq-xl--1.0
 2) bgq-xl/1.0
$ mpixlf90_r -O2 -qlistopt -qreport -qsmp=omp -o check.x check.f90
-L$LAPACK_LIB -llapack
** checkmpi   === End of Compilation 1 ===
1501-510  Compilation successful for file check.f90.
```

CINECA SCAI SuperComputing Applications and Innovation

Optimization with IBM XL

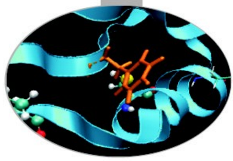


- Add **-qlistopt -qreport** to generate a file.lst containing the optimization flags
- **SIMD** (**S**ingle **I**nstruction **M**ultiple **D**ata) vectorization activated by default where possible (deactivate it with **-qsimd**=noauto)

  NB: loops are not SIMD vectorized when step "i" depends on step "i+1" or "i-1"
- **Unrolling** loops: **-qunroll**=yes (default with -O3 optimization, deactivate it with **-qnounroll)**
- -O3 optimization uses -qnostrict: the semantic of your code could be altered. If you want to keep the semantic use -qstrict option
- **Threading**: **-qsmp**=auto|omp|noauto|...
  NB If you specify -qsmp=omp, where omp is not possible the default is autothreading. to disable autothreading use **-qsmp**=omp:noauto (in this case only program code explicitly parallelized with OpenMP directives will be optimized)

for further info: http://pic.dhe.ibm.com/infocenter/compbg/v121v141/index.jsp

Libraries: Use the right LINK!



- Many compilation errors are due to wrong or incomplete library linking (**undefined reference**): don't panic!
- Remember to load your modules (module avail, module load):

```
module load library/version
```

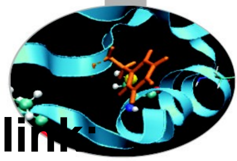
(fftw/2.1.5--bgq-xl--1.0, lapack/3.4.1--bgq-xl--1.0... ecc.)

- all library paths are in the form $LIBRARY**_LIB** ($FFTW**_LIB**, $LAPACK**_LIB** ecc.) ; include paths are in the form $LIBRARY**_INC**

```
$ module load hdf5/1.8.9_ser--bgq-xl--1.0
$ ls $HDF5_LIB
libhdf5.a       libhdf5_cpp.la     libhdf5_fortran.la  libhdf5_hl_cpp.a
libhdf5hl_fortran.a   libhdf5_hl.la  libhdf5.settings libhdf5_cpp.a
libhdf5_fortran.a  libhdf5_hl.a  libhdf5_hl_cpp.la  libhdf5hl_fortran.la
libhdf5.la
```

How to find which library I need?

Libraries (2)



Use the command "nm" to find the reference and the right library to link:

```
$ for i in `ls $HDF5_LIB/*.a` ; do echo $i ; nm $i | grep H5Z_xform_copy ; done
[...]
/cineca/prod/libraries/hdf5/1.8.9_ser/bgq-xl--1.0/lib/libhdf5.a
                 U H5Z_xform_copy
0000000000000168 D H5Z_xform_copy
0000000000000150 d H5Z_xform_copy_tree
[...]
```

2 ways to link a library:

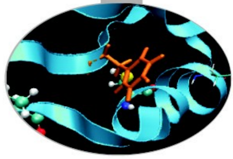
-L$LIBRARY_LIB **-l**name --- or --- $LIBRARY_LIB/**lib**name**.a**

```
1) mpixlc_r -I$HDF5_INC input.c -L$HDF5_LIB -lhdf5 \
            -L$SZIP_LIB -lsz -L$ZLIB_LIB -lz
2) mpixlc_r -I$HDF5_INC input.c $HDF5_LIB/libhdfc5.a \
            $SZiP_LIB/libsz.a $ZLIB_LIB/bibz.a
```

Example:

```
$ module load bgq-xl
$ module load hdf5/1.8.9_ser--bgq-xl--1.0
$ module load szip zlib
$ mpixlc_r -I$HDF5_INC input.c -L$HDF5_LIB -lhdf5 \
           -L$SZIP_LIB -lsz -L$ZLIB_LIB -lz
```

Compilation/Linking Issues



Linking an object compiled with XL using GNU compiler is possible, but sometimes you can get an "undefined reference" error

```
$ mpif90 -fopenmp -o myprog.x myprog.f90 test.o -L$LAPACK_LIB -llapack
test.o:(.text+0xb8): undefined reference to `_xlfBeginIO'
test.o:(.text+0xd0): undefined reference to `_xlfWriteLDChar'
test.o:(.text+0xe8): undefined reference to `_xlfWriteLDInt'
test.o:(.text+0xf4): undefined reference to `_xlfEndIO'
```

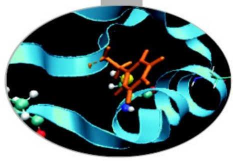
Find the library where the symbol is defined in the XL library path
fortran library path: **/opt/ibmcmp/xlf/bg/14.1/lib64/**
C/C++ library path: **/opt/ibmcmp/vacpp/bg/12.1/lib64/**

```
$ for i in `ls /opt/ibmcmp/xlf/bg/14.1/lib64/*.a`; do echo $i ; \
  nm $i | grep _xlfBeginIO ; done
[...]
/opt/ibmcmp/xlf/bg/14.1/lib64/libxlf90_t.a
0000000000000000 D _xlfBeginIO
                 U _xlfBeginIO
[...]
```

Add the library to the linker:

```
$ mpif90 -fopenmp -o myprog.x myprog.f90 test.o \
          -L$LAPACK_LIB -llapack \
          -L/opt/ibmcmp/xlf/bg/14.1/lib64/ -lxlf90_t
```

Useful Links
Documentation



- **FERMI reference guide**:

  http://www.hpc.cineca.it/content/ibm-fermi-user-guide

- **IBM compilers guide**:

  http://pic.dhe.ibm.com/infocenter/compbg/v121v141/index.jsp

- **Stay tuned with the HPC news:**

  http://www.hpc.cineca.it/content/stay-tuned

- **"man" command:** man bgxlf90, man bgxlc, man rsync ...
- **HPC CINECA User Support:** mail to superc@cineca.it
- ... And if you are highly motivated to better understand HPC... (Or just curious!)

  V. Eijkhout, *Introduction to High Performance Scientific Computing* (2011)

And in case of panic...