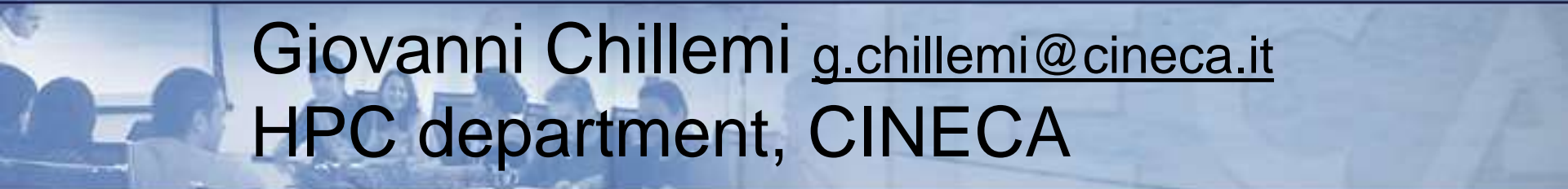
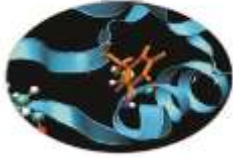


Running MD on HPC architectures II. Bluegene/Q

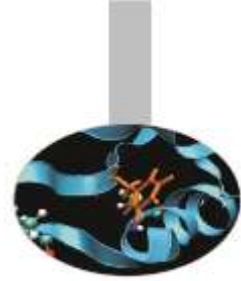


Giovanni Chillemi g.chillemi@ Cineca.it
HPC department, CINECA

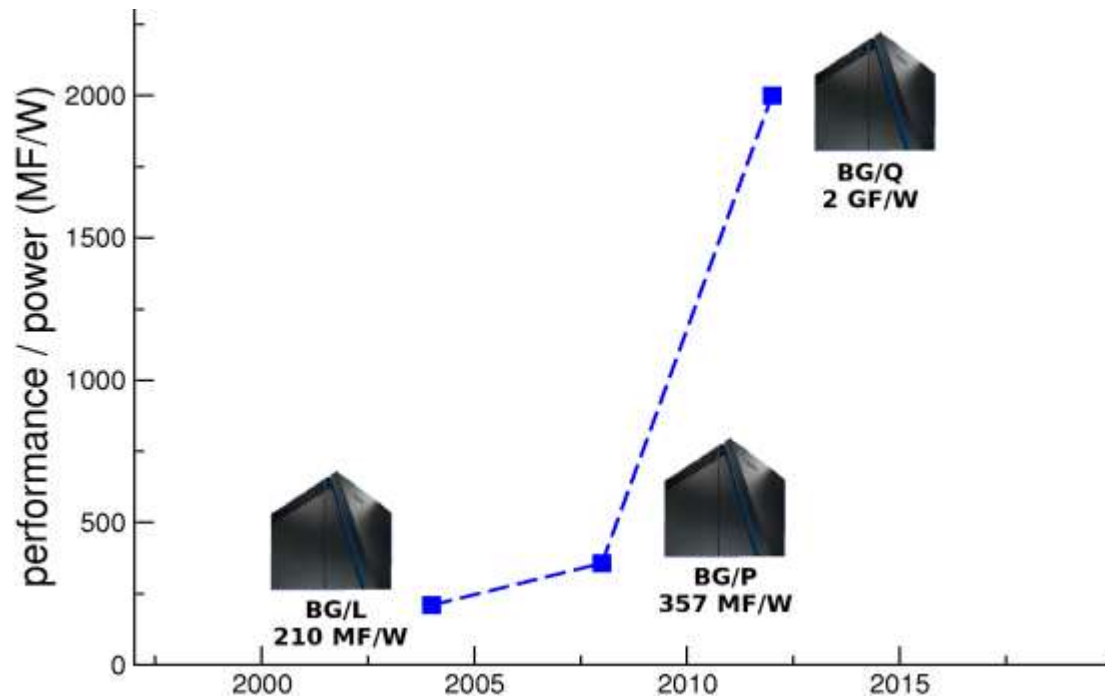
Outline

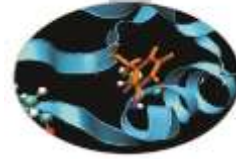


- Overview of Blue Gene/Q architecture
- Production environment
- How to launch a MD code on Fermi
- Example: Satellite Tobacco Mosaic virus with NAMD



- ❑ BG is a **massively parallel** supercomputer
- ❑ It holds different types of nodes (and networks)
- ❑ It is designed to have **high energy-efficiency (performance/power)**





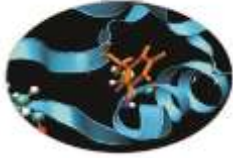
BLUE GENE EVOLUTION

	Total		Biggest Config	Per rack		
	Performance [PF]	Efficiency [MF/W]		Max # of racks	Performance [TF]	Efficiency
BG/L	0.596	210	104	5.7	2.02	2048
BG/P	1	357	72	13.9	4.96	4096
BG/Q	20	2000	96	209	20.83	16384

Towards higher and higher:

- Performance
- Efficiency
- Density of cores per rack





Content of a BGQ Rack

- 2 **midplanes** form a rack
- Each midplane has 16 **Node Cards**
- Each Node Card has 32 **Compute Nodes**
 - $2 \times 32 \times 16 = 1024$ compute nodes per rack
- Flexible I/O nodes – node cards ratio
(at least 512 cores per I/O node)
 - In our configuration: 2 rack con 1024 core per I/O node
8 rack con 2048 core per I/O node

I/O drawers





FERMI @ CINECA

PRACE Tier-0 System



Architecture: 10 BG/Q Racks
Model: IBM-BG/Q
Processor Type: IBM PowerA2, 1.6 GHz
Computing Cores: 163840
Computing Nodes: 10240
RAM: 1 GByte / core
Internal Network: 5D Torus
Disk Space: 2 PByte of scratch space
Peak Performance: 2 PFlop/s
Power Consumption: 1 MWatt





Note that..

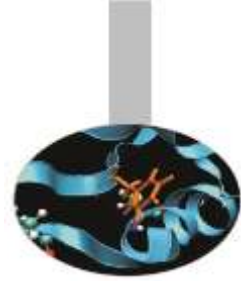
The number of I/O nodes per rack constraints:

-  I/O bandwidth to/from compute racks
(each I/O node has 2 links (4GB/s in 4GB/s out))
-  The minimum partition allocatable on a BG/Q system (“small block” jobs)

For FERMI:

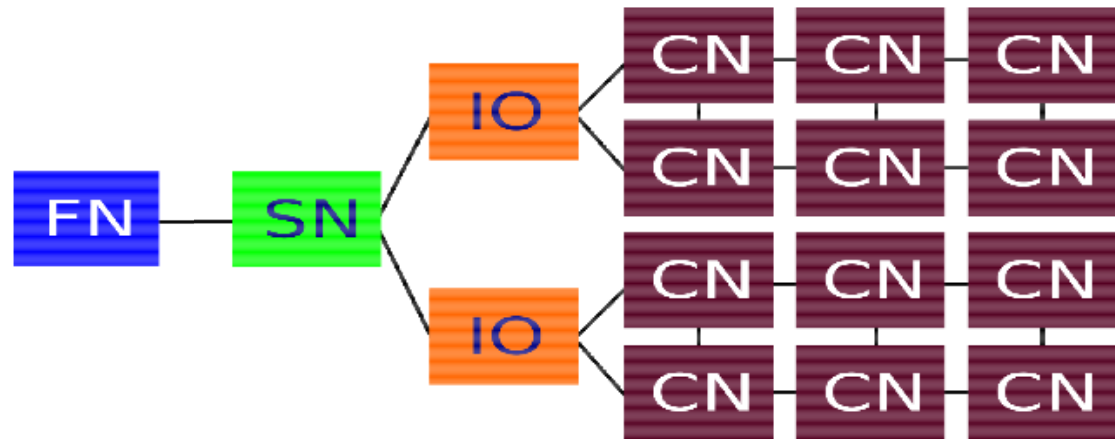
- bg_size=64 (jobs running on R11 and R31)
- bg_size=128 (jobs running on the other racks)

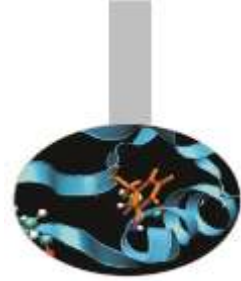




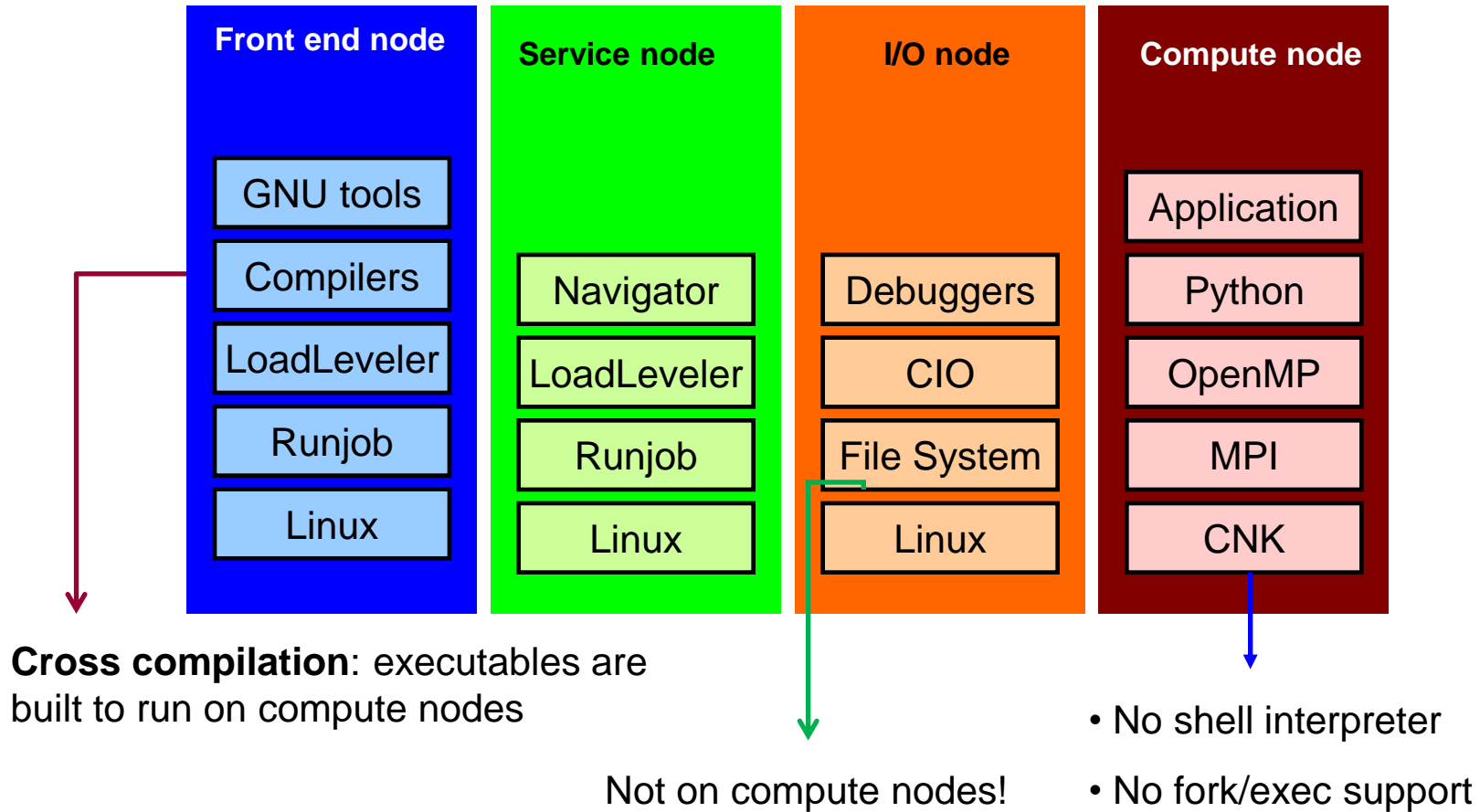
BG/Q I/O architecture

- **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

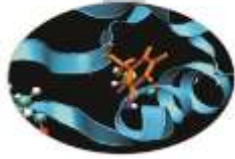




Software stack: Overview



BGQ Networks



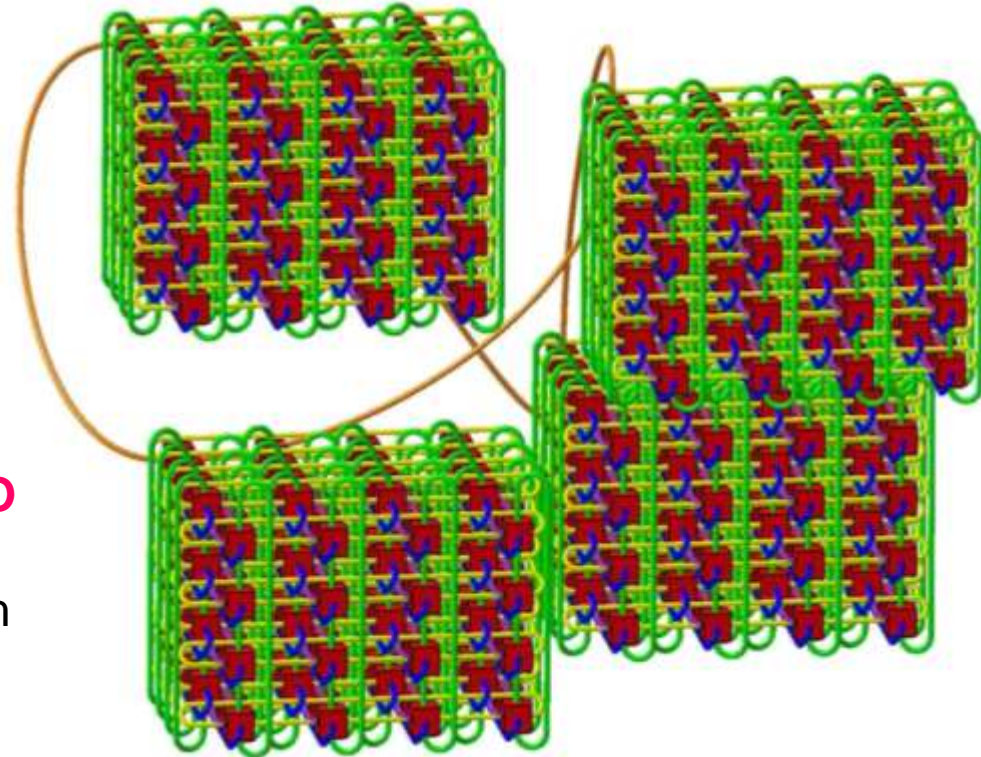
5D Torus Network

5D topology for point-to-point communication

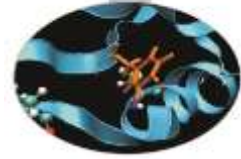
- 2 GB/s bidirectional bandwidth on all (10+1) links, 5D nearest neighbour exchange at 1.75 GB/s per link
- Collective and barrier networks embedded in 5-D torus network.

External, independent and dynamic I/O system

- I/O nodes in separate drawers/rack with private interconnections
- I/O network to/from Compute rack: 2 links (4 GB/s in 4 GB/s out)



Fermi Production environment



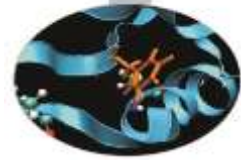
queue name	ComputeNodes	max wall time	defined on	notes	priority
debug	64	00:30:00	2 rack with 16 I/O nodes	Maxqueued(*)=1	high
longdebug	64	24:00:00	2 rack with 16 I/O nodes	Maxqueued(*)=2	low
smallpar	128	24:00:00	8 rack with 16 I/O nodes	Maxqueued(*)=4	high
parallel	256 - 512	24:00:00	2 rack with 8 I/O nodes	Maxqueued(*)=2	high
bigpar	1024 - 2048 (1-2 racks)	24:00:00	6 rack with 8 I/O nodes	Maxqueued(*)=2	very high
keyproject	1024 - 6*1024 (1-6 racks)	24:00:00	8 rack with 8 I/O nodes	Ask to superc@cinca.it Due to the geometry of the system, the max number of usable racks is 6	very high
serial	1 core	06:00:00	front-end node	to be requested with @#class=serial	high



FERMI User's Guide:

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

<http://www.hpc.cineca.it/content/batch-scheduler-loadleveler-0>



Batch Scheduler LoadLeveler ... x

www.hpc.cineca.it/content/batch-scheduler-loadleveler-0

ABOUT US RESOURCES SERVICES FOR USERS TRAINING PROJECTS

FERMI status PLX status PICO status

Home > For users > Documentation > HPC User Guide > System Specific > IBM-FERMI

Batch Scheduler LoadLeveler

In this page

- [The basic LL commands](#)
- [LL script file syntax](#)
 - [General LL keywords](#)
 - [BGQ specific LL keywords](#)
- [The runjob command](#)
- [Serial and parallel jobs](#)
 - [Ex1: serial batch job](#)
 - [Ex2: pure MPI batch job](#)
 - [Ex3: pure MPI batch job](#)
 - [Ex4: MPI + OpenMP hybrid job](#)
- [Multi-step jobs](#)
 - [Ex5: pure MPI multistep job](#)
 - [Ex6: serial and MPI multistep job](#)
- [Sub-block jobs](#)

For users

- UserDB
- Getting started
- Get in touch
- Help desk
- Documentation
 - HPC User Guide
 - Introduction
 - General Info
 - System Specific
 - Services
 - Other Documents
 - FAQ
 - Documentation at a glance

Help desk

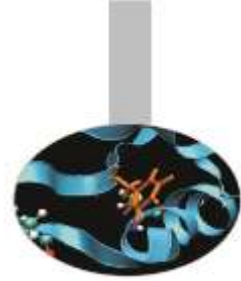
Support

Silvia Giuliani

Center news



How to submit your job script: LL commands



llsubmit

Example:

```
$> llsubmit job.cmd
```

llq

Example:

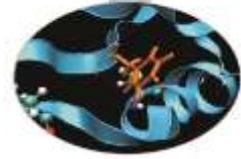
```
llq -u $USER
```

```
$> llq -u amarani0
```

Id	Owner	Submitted	ST	PRI	Class	Running	On
fen04.7334.0	amarani0	9/21 15:11	I	50	parallel		

1 job step(s) in query, 1 waiting, 0 pending, 0 running, 0 held, 0 preempted

How to submit your job script: LL commands



```
$> llq -l <job_id>
```

- will print a more verbose output will be generated for job_id
- In particular you'll be notified about the bgsizes you requested and the real bgsizes allocated:

```
.....  
.....  
BG Size Requested: 1024  
BG Size Allocated: 1024  
BG Shape Requested:  
BG Shape Allocated: 1x1x1x2  
BG Connectivity Requested: Mesh  
BG Connectivity Allocated: Torus Torus Torus Torus  
.....  
.....
```

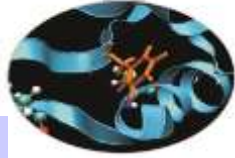
```
$> llcancel
```

Example:

```
>llcancel <job_id>
```



Job script: general structure



```
#!/bin/bash
# @ job_name = bgsizex$(jobid)
# @ output = z.$(jobid).out
# @ error = z.$(jobid).err
# @ shell = /bin/bash
# @ job_type = bluegene
# @ wall_clock_limit = 02:00:00
# @ notification = never
# @ bg_size = 256
# @ class = keyproject
# @ account_no = cinstaff
# @ restart = no
# @ queue
```

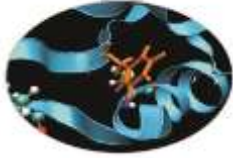
LL keywords

```
cd /gpfs/scratch/userinternal/cin0753a/mydir
```

```
runjob -ranks-per-node 64 ./program.exe
```

} **Application
block**

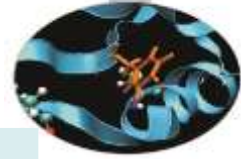




runjob

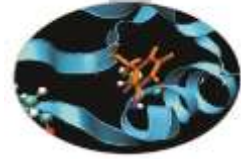
Applications always need to be executed with **runjob** command to run on the compute nodes

- ▶ **Syntax:**
 - runjob [options]
 - runjob [options] binary [arg1 arg2 ... argn]
- ▶ **Parameters** can be set
 - by command-line options (higher priority!)
 - environment variables
- ▶ **runjob -h** for a complete list



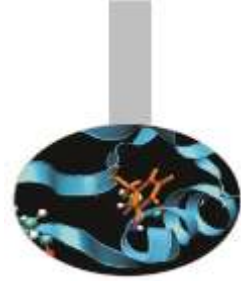
runjob options – I (job)

Command line option	Environment variable	Description
<code>--exe executable</code>	<code>RUNJOB_EXE=executable</code>	Specifies the full path to the executable (this argument can be also specified as the first argument after “:.”) <code>runjob --exe /home/user/a.out</code> <code>runjob : /home/user/a.out</code>
<code>--args “prg_args”</code>	<code>RUNJOB_ARGS=“prg_args”</code>	Passes “prg_args” to the launched application on the compute node <code>runjob : a.out hello world</code> <code>runjob --args hello world --exe a.out</code>
<code>--envs “ENVVAR=values”</code>	<code>RUNJOB_ENVS=“ENVVAR=value”</code>	Sets the environment variable <code>ENVVAR=value</code> in the job environment on the compute nodes
<code>-exp_env ENVVAR</code>	<code>RUNJOB_EXP_ENV=ENVVAR</code>	Exports the environment variable <code>ENVVAR</code> in the current environment to the job on the compute nodes



runjob options – II (resources)

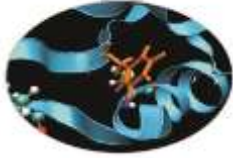
Command line option	Environment variable	Description
--ranks-per-node		Specifies the number of processes per compute node. Valid values are:1,2,4,8,16,32,64
--np n	RUNJOB_NP=n	Number of processes in the job(<= block_size*ranks-per_node)
--mapping	RUNJOB_MAPPING=mapfile	Permutation of ABCDET or a path to a mapping file containing coordinates for each rank



runjob options – III

Command line option	Environment variable	Description
--start_tool		Path to tool to start with the job
--tool_args		Arguments for the tool

Fermi Production environment



```
$>module av
```

```
----- /cineca/prod/modulefiles/base/applications -----  
abinit/6.12.3      crystal09/1.01      qe/5.0bgq  
amber/12           dl_poly/4.03        siesta/3.1  
bigdft/1.6.0      gromacs/4.5.5       vasp/5.2.12  
cp2k/2.3          lammmps/20120816    vasp/5.3.2  
namd/2.9          cpmd/v3.15.3
```

To load a specific module, i.e. set specific env vars

```
$> module load <module_name>
```

Show the variables set by a module

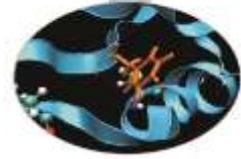
```
$> module show <module_name>
```

Retrieve informations of a module

```
$> module help <module_name>
```



Available MD packages on Fermi



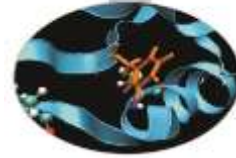
Some of the currently installed MD codes on Fermi and available for use:

```
$> module av
```

```
----- /cineca/prod/modulefiles/base/applications -----
```

```
abinit/6.12.3  
amber/12(default)  
bigdft/1.6.0  
cp2k/2.3  
cp2k/2.4(default)  
cpmd/3.15.3_rev2606  
cpmd/3.17.1(default)  
namd/2.10  
crystal09/1.01  
crystal09/2.0.1(default)  
dl_poly/4.03  
dl_poly/4.05(default)  
gromacs/4.5.5(default)  
gromacs/4.6.1  
lammmps/20120816  
namd/2.9(default)  
enzo/2.2  
nwchem/6.3  
octopus/4.1(default)  
openfoam/2.1.1  
qe/5.0.3b(default)  
qe/5.0bgq  
siesta/3.1  
siesta/3.1-TS  
vasp/5.2.12  
gromacs/4.5.5  
vasp/5.3.3(default)
```

NAMD template script for Fermi



```
#!/bin/bash
# @ job_name = namd.$(jobid)
# @ output = $(job_name).out
# @ error = $(job_name).err
# @ shell = /bin/bash
# @ job_type = bluegene
# @ wall_clock_limit = 01:00:00
# @ bg_size = 64
# @ account_no = your_account_name
# @ queue
```

$64 * 4 = 256$ MPI tasks

$* 16 = 4096$ threads

```
module load namd/2.9
```

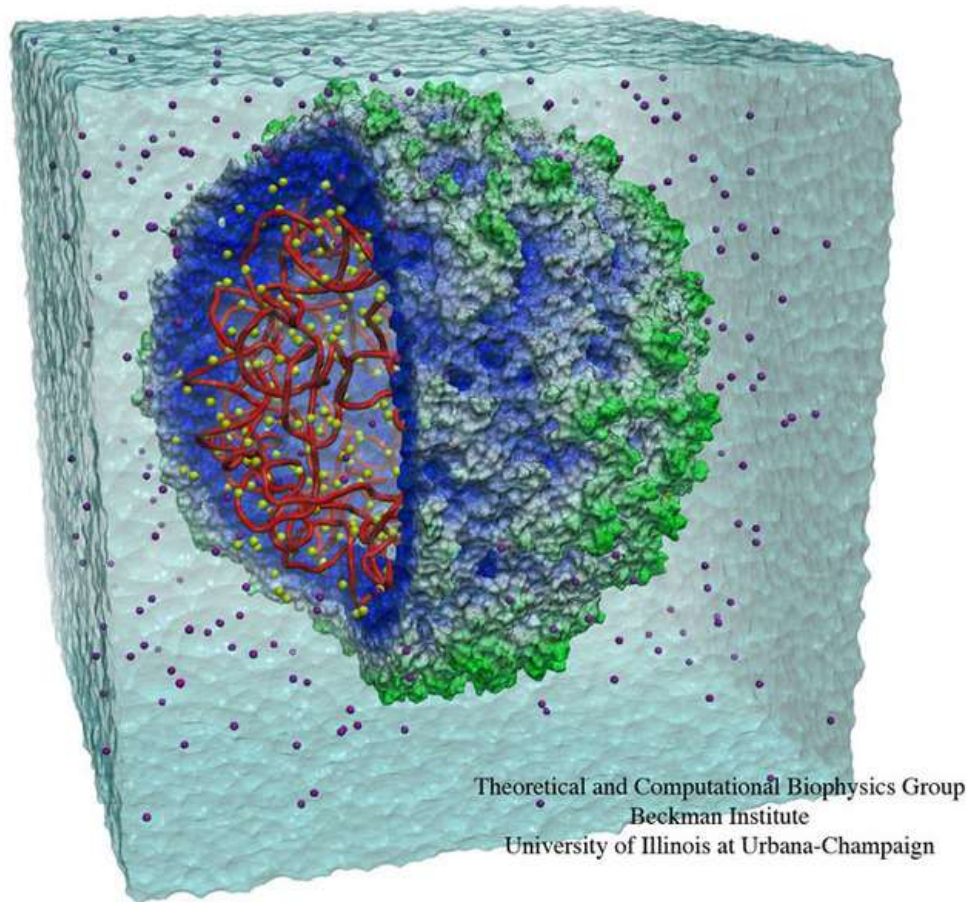
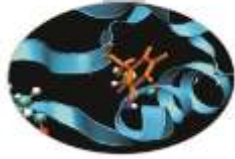
```
#launch with 256 processes (grouping 4 processes per node) using multi-thread (16 threads per process)
runjob --rank-per-node 4 : $NAMD_HOME/namd2 +ppn16 input.namd > output.log
```

Optimized by IBM namd version: adopts a mixed MPI/OpenMP thread approach for the parallel computation.

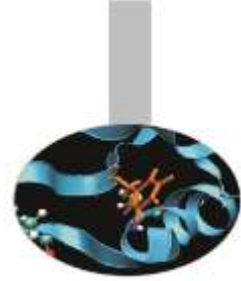
The number of MPI process per node are selected with the --ranks-per-node option of LoadLeveler, while the number of OpenMP threads per MPI process with the +ppn flag of namd.



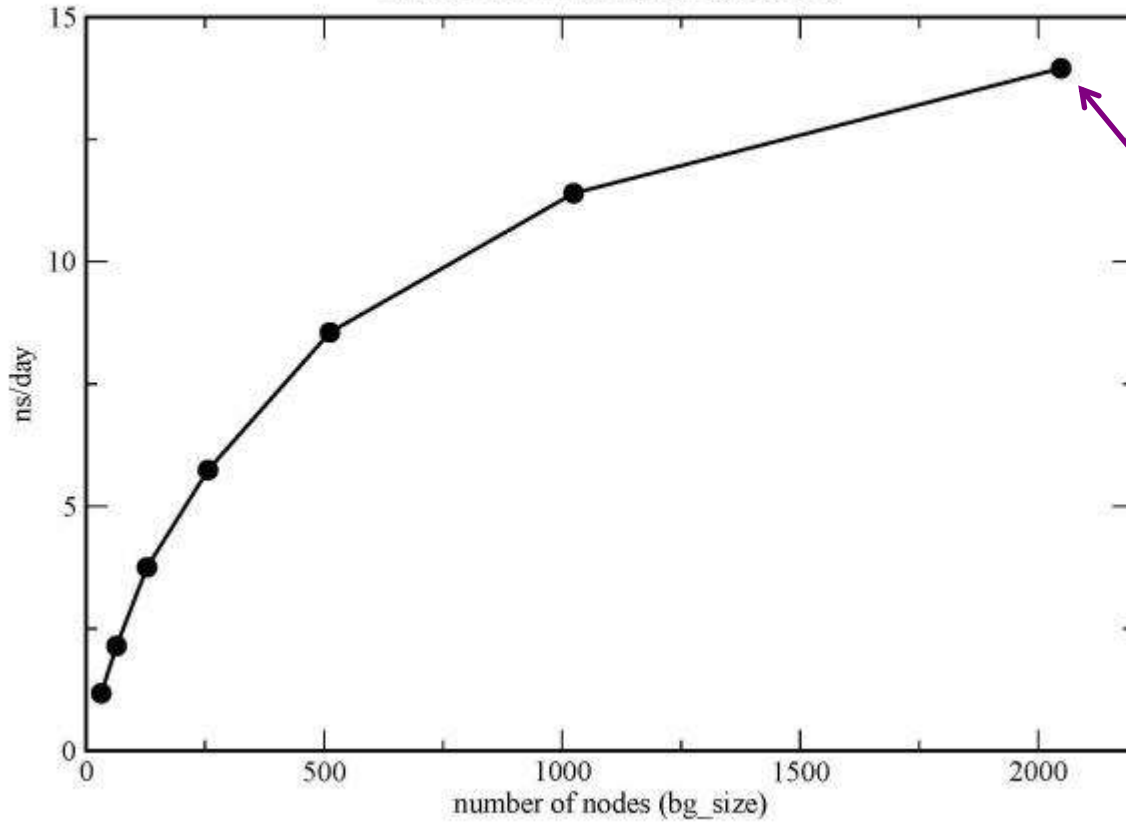
Satellite Tobacco Mosaic virus



STMV (virus) benchmark (1,066,628 atoms, periodic, PME)

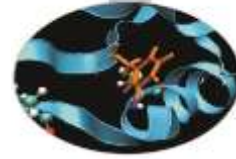


Satellite Tobacco Mosaic Virus NAMD 2.9, 1.2 MAtoms, BlueGene/Q



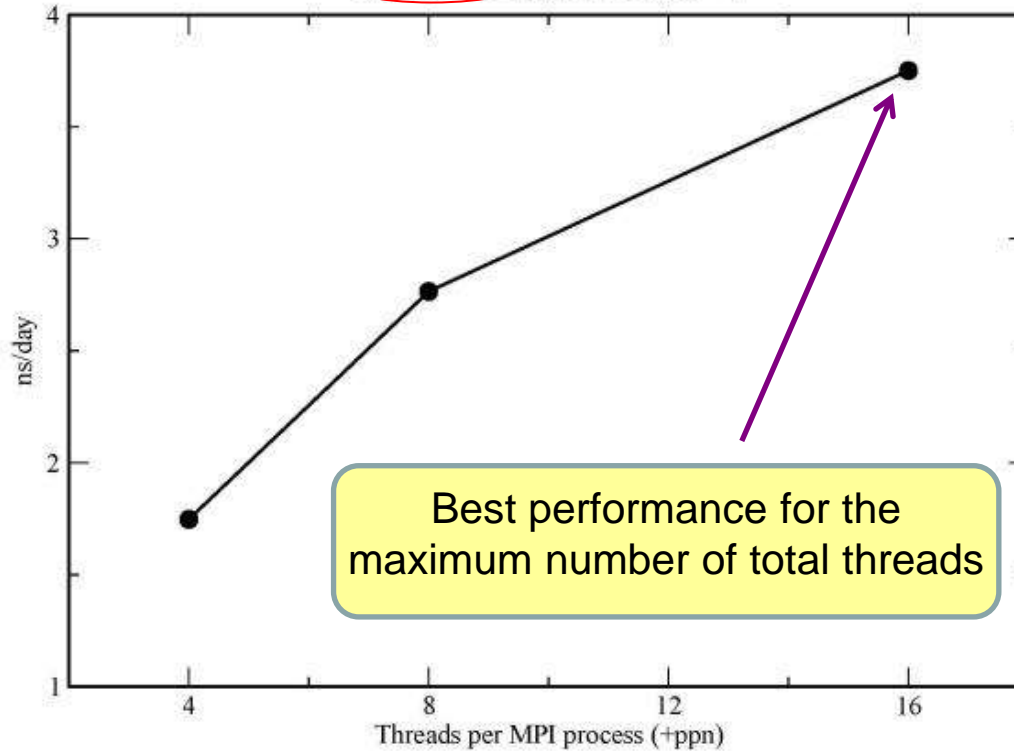
Increase of performance up to 32,768 cores! (bg_size=2048)



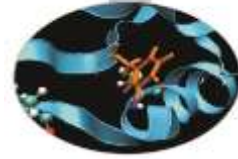


Satellite Tobacco Mosaic Virus

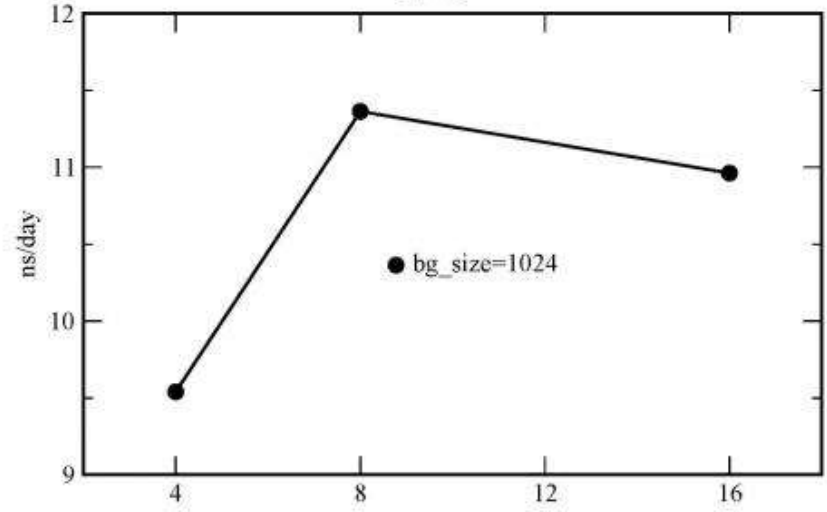
bg_size = 128; ranks_per_node = 4



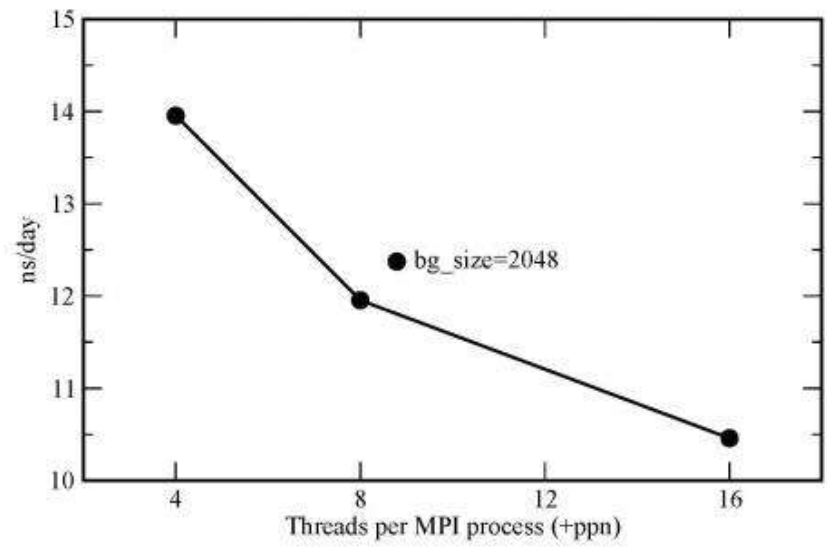
```
runjob --np 512 --ranks-per-node 4 : $NAMD_HOME/namd2 stmv_ori.namd +ppn4 > stmv.out  
runjob --np 512 --ranks-per-node 4 : $NAMD_HOME/namd2 stmv_ori.namd +ppn8 > stmv.out  
runjob --np 512 --ranks-per-node 4 : $NAMD_HOME/namd2 stmv_ori.namd +ppn16 > stmv.out
```

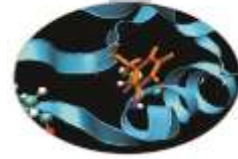


Satellite Tobacco Mosaic Virus ranks_per_node = 4



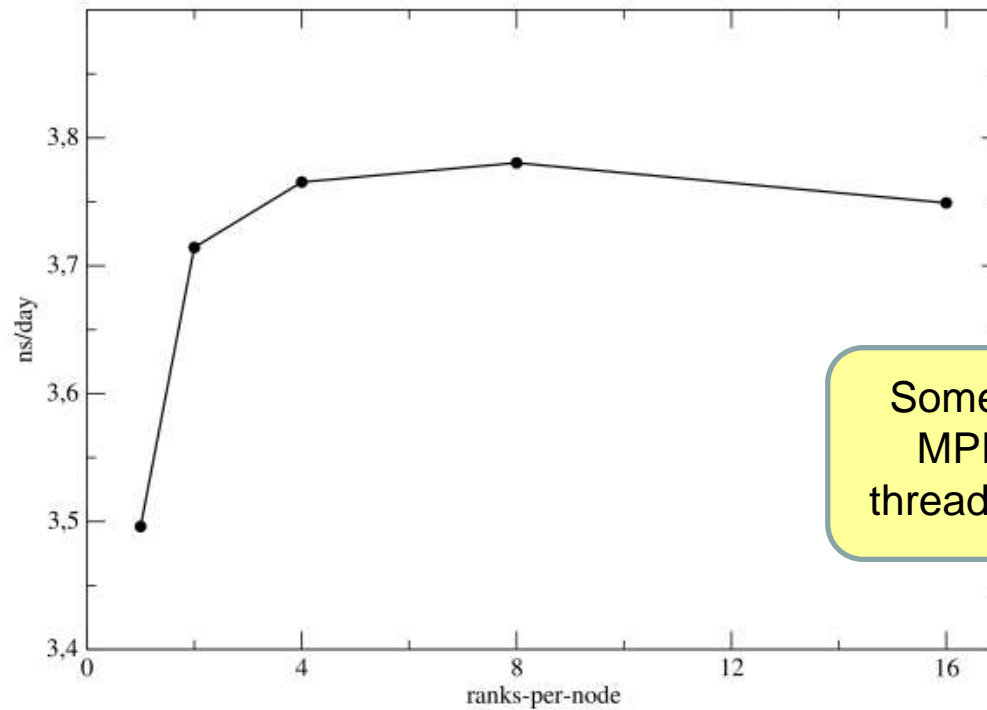
That is not true anymore when increasing bg_size: Check the performance on your system!





Satellite Tobacco Mosaico Virus

bg_size = 128; n. threads 8192



Some combinations of MPI tasks/OpenMP threads are less efficient

```
runjob --np 2048 --ranks-per-node 16 : $NAMD_HOME/namd2 stmv_ori.namd +ppn4 > stmv.out  
runjob --np 1024 --ranks-per-node 8 : $NAMD_HOME/namd2 stmv_ori.namd +ppn8 > stmv.out  
runjob --np 512 --ranks-per-node 4 : $NAMD_HOME/namd2 stmv_ori.namd +ppn16 > stmv.out  
runjob --np 256 --ranks-per-node 2 : $NAMD_HOME/namd2 stmv_ori.namd +ppn32 > stmv.out  
runjob --np 128 --ranks-per-node 1 : $NAMD_HOME/namd2 stmv_ori.namd +ppn64 > stmv.out
```

Amber template script for Fermi

```
#!/bin/bash
# @ job_name = amber.$(jobid)
# @ output = z.$(jobid).out
# @ error = z.$(jobid).err
# @ shell = /bin/bash
# @ job_type = bluegene
# @ wall_clock_limit = 01:00:00
# @ notification = always
# @ bg_size = 64
# @ account_no = my_account_no
# @ queue

module load amber/12

# do any amber pre-processing on the front-end nodes (antechamber, tleaps, etc...)

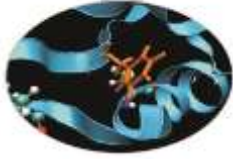
# get the path of the sander executable for the backend nodes
sander=$(which sander.MPI)

# add any sander options
exe="$sander -i mdin -o mdout -p prmtop -c inpcrd -r restrt -ref refc -mtmd mtmd -x mdcrd"

# launch sander on all the allocated back-end nodes (note the : which must be present with this syntax)
# if you have memory problems reduce the --ranks-per-node option
runjob --ranks-per-node 16 --env-all : $exe
```



GROMACS template script for Fermi



```
#!/bin/bash
# @ job_name = gromacs.$(jobid)
# @ output = $(jobid).out
# @ error = $(jobid).err
# @ shell = /bin/bash
# @ job_type = bluegene
# @ wall_clock_limit = 01:00:00
# @ notification = always
# @ bg_size = 64
# @ account_no = <my_account_no>
# @ queue
```

64*16 = 1024 MPI tasks
*4 = 4096 threads

mixed MPI/OpenMP thread version

```
module load gromacs/4.6.5
```

```
# get the path of the mdrun executable for the backend nodes
mdrun=$(which mdrun_bgq)
```

```
# add any mdrun options.
# Here we are using 4 OpenMP threads for MPI task.
exe="$mdrun -v -s topol.tpr -ntomp 4"
```

```
#launch single precision mdrun on all the back-end nodes
runjob --ranks-per-node 16 --env-all : $exe
```

