DEISA Common Production Environment

Distributed
European
Infrastructure for
Supercomputing
Applications

Denis Girou

Denis.Girou@idris.fr

Agenda

Definition 3
Contents 4
Software versions 8
Modules framework 9
Modules implementation 10
Modules usage 12
Management of libraries 16
Software monitoring for the DCPE 18

Definition

- Both a set of software (the *software stack*) and a generic interface to access the software (based on the *Modules* tool)
- Required to both offer a common interface to the users and to hide the differences between local installations
- Essential feature for job migration inside homogeneous super-clusters
- Very high level of coherence inside homogeneous super-clusters (AIX and Altix), with only small discrepancies on minor versions of the software components allowed, but lower compatibility across platforms of different architectures
- DCPE model, with its *Modules* interface, also used for all local (non-DEISA) users by several partners
- See Primer chapter 4

Contents

- The DCPE includes:
  - shells (Bash and Tcsh),
  - compilers (C, C++, Fortran and Java),
  - libraries (for numerical analysis, data formatting, etc.),
  - tools (debuggers, profilers, editors, development tools),
  - applications.

Contents

Software	Class	Description	Restrictions
SHELLS			
bash	shell	Bourne Again Shell	(Bash)
tcsh	shell	Extended C Shell	(Tcsh)
COMPILERS			
c	compiler C	compiler	
c++	compiler C++	compiler	
fortran	compiler Fortran	compiler	
java	compiler Java	compiler	
LIBRARIES			
blacs	numerical	Basic Linear Algebra Communication Subroutines	
blacssmp	numerical	Basic Linear Algebra Communication Sub. SMP	IBM only
blas	numerical	Basic Linear Algebra Subroutines	
blassmp	numerical	Basic Linear Algebra Subroutines SMP	IBM only
essl	numerical	IBM Engineering and Scientific Subroutine Library	IBM only
fftw	numerical	Fast Fourier Transform library	
hdf5	data form.	Hierarchical Data Format (version 5)	
lapack	numerical	Linear Algebra PACKage	

Contents

LIBRARIES (next)			
mass	numerical	IBM basic numerical library	IBM only
mkl	numerical	Math Kernel Library	SGI only
nag	numerical	Numerical library from the Num Algorithms Group	commercial
netcdf	data form.	Network Common Data Format	
pessl	numerical	IBM Parallel ESSL	IBM only
scalapack	numerical	SCAlable Linear Algebra PACKage	
scsl	numerical	Scientific Computing Software Library	SGI only
wsmp	numerical	IBM Watson Sparse Matrix Package	IBM only, commercial
TOOLS			
emacs	editor	Text editor	
gmake	utilities	GNU make	
histx	profiler	Performance counter measurements	SGI only
hpm	profiler	Hardware Profiler Monitor	IBM only
nedit	editor	Text editor	
omniorb	utilities	CORBA implementation	
openssh	utilities	Secure shell	

Contents

TOOLS (next)			
perl	utilities	Practical Extraction and Report Language	
python	utilities	Development language	
tcl	utilities	Tool Command Language	
tk	utilities	GUI toolkit for Tcl	
totalview	debugger	Etnus parallel debugger	commercial
APPLICATIONS			
cpmd	chemistry	Molecular dynamics (Car-Parrinello)	
cpmd2cube	chemistry	Post-processing of CPMD density/wannier files	IBM only
gopenmol	chemistry	Post-processing of cubes-files	IBM only
namd	chemistry	Scalable Molecular Dynamics	SGI only
torb	plasma phy.	Turbulence	IBM only
torbwsmp	plasma phy.	Turbulence (using the WSMP IBM library)	IBM only
wien2k	chemistry	Electronic structure calculations of solids	IBM only

Software versions

- When several versions of a software are available, the default one is supposed to be used in all normal cases
- DEISA default versions are not the same thing as local default versions and they can be different
- Only one version can be used at a given time
- Discrepancies allowed between sites on the minor versions or patch levels (for instance, `fortran` could be 9.1.0.0 in some places and 9.1.0.2 in some others)

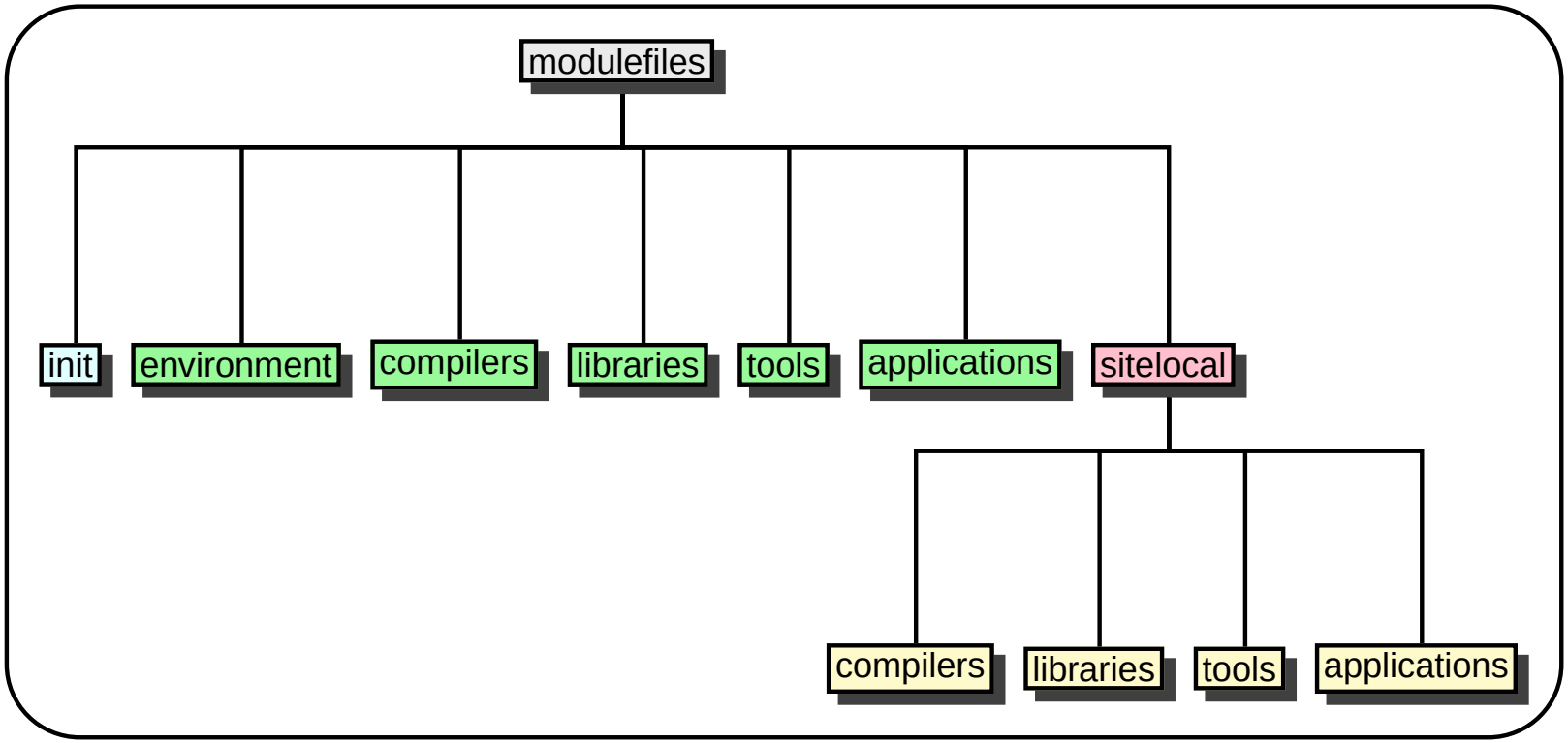
Modules framework

- Modules tool chosen because it was well known by many sites and many users (especially those who used CRAY systems in the past)
- Public domain software
- Tcl implementation used
- Allows:
  - to offer a common interface to access different software components on different subgroups of homogeneous computers,
  - to hide the possibility that different files of a component might be named or installed in different locations depending on local policies of the sites, and on software packaging for different computer types,
  - to manage individually each software and load only those required into the user environment,
  - for each user to change the version of each software independently of the others,
  - for each user to switch independently between the current default version of a software to another one (older or newer).

Modules implementation

- Implemented according to the organization of the DEISA software stack
- Usage has been kept as straightforward as possible, at the expense of some complexity in the development
- Special care was put into the robustness of the system. Various coherence tests are done to prevent the users from defining incompatible choices.

Modules implementation



Modules usage

- Only one command to know: module
- Used mainly to list, load and unload the modulefiles which configure the user's environment to give access to the corresponding software
- Simple syntax to access to a *modulefile*: `modulename[/version]`
- Known weakness of the Modules tool: the dependencies between the *modulefiles* are not managed. We enhanced it, especially for the management of the libraries, but in a limited way.

Modules usage

```
> module help
Available Commands and Usage:
+ list
+ avail
+ help modulefile [modulefile ...]
+ whatis modulefile
+ display modulefile
+ load modulefile [modulefile ...]
+ unload modulefile [modulefile ...]
+ switch modulefile1 modulefile2
. . .
```

Modules usage

☞ Initialisation of the user's DEISA Modules environment made by a special *modulefile* called deisa

```
> module load deisa
load 64-bit mode (OBJECT_MODE, DEISA_FFLAGS, DEISA_CFLAGS,
                  DEISA_CXXFLAGS) (default)
load integer variables in Fortran on 4-byte (default)
load real variables in single precision in Fortran on 4-byte (default)
load real variables in double precision in Fortran on 8-byte (default)
load big endian multibytes numeric representation (default)
load IBM C compiler version 8.0 (PATH, LIBPATH, NLSPATH, MANPATH)
load IBM C++ compiler version 8.0 (PATH, LIBPATH, NLSPATH, MANPATH)
load IBM Fortran compiler version 9.1 (PATH, LIBPATH, NLSPATH, MANPATH)
load usage of the compiler wrappers (default)
```

Modules usage

- Load some default *modulefiles* (not necessarily the same ones depending on the architecture of the computer used)

```
> module list
Currently Loaded Modulefiles:
1) mode/64            5) endian/big    9) compilerwrappers/yes
2) fortraninteger/4   6) c++/8.0      10) deisa
3) fortranreal/4      7) c/8.0
4) fortrandouble/8    8) fortran/9.1
```

- Initialise some special environment variables, in particular those used to define the file systems in a portable way (`DEISA_HOME`, etc.)
- Initialise some default characteristics of libraries, tools and applications: numeric representation (big/little endian), addressing mode on some systems (64- or 32-bit), size of default Fortran 77 integers and float numbers

Management of libraries

- This is a difficult point, generally not handled or only in a very simple way
- Important because this is difficult for users too! Especially, they want to use libraries like ScaLAPACK without knowing that its required dependant libraries are not the same on the various systems:
  IBM AIX: `-lblacssmp -lpesslsmp -lessl -lscalapack`
  NEC Super-UX: `-lblacsF90init -lblacs -lmpi -llapack -lblas -lscalapack`
- Problem of the dependencies between libraries
- The management of some compiler switches for Fortran 77 by some *modulefiles* (sizes of integers and float numbers) is also linked to the management of libraries

Management of libraries

- Three methods supported to handle the libraries, a default method for most of the users and usages, and two alternative ones for advanced users:
  - default: compiler wrappers with implicit usage of global environment variables
  - no compiler wrappers with explicit usage of individual environment variables
  - no compiler wrappers with explicit usage of global environment variables

Software monitoring for the DCPE

- Internal monitoring to give an up to date overview of the status of all the software components of the DCPE, on all the production platforms
- Choice of the INCA framework, developed by the *San Diego Supercomputer Center*, initially for the TeraGrid project
- Contains version reporters, which test the availability of each software package and its *Modules* interface, and check if the version installed matches the requirements
- Contains also unit reporters, which check not only the availability of the major software, but their effective usage, reporting the current behaviour of the compilers, linkers, the execution of OpenMP and MPI codes, etc.

Software monitoring for the DCPE

Home

DEISA

DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR SUPERCOMPUTING APPLICATIONS

Deisa Supercomputing Grid Infrastructure | Project Organisation | Users' Corner | Press Corner | News & Events | Links

inca.deisa.org Common Production Environment (CPE) - Inca status page

Summary view | Stack view | Status graphs

Summary of DEISA CPE 1.0
Page generated by Inca: 06/07/06 15:51 CEST

This page offers a summary of results for critical grid, development, and cluster tests (view list of tests). Details about a resource's test results are available by clicking on the resource name in the "Site-Resource" column of the table.

Site	Resource	Total Pass	Compilers	Libraries	Shells	Tools	Applications
CINECA	sp5	97%	100% (4 / 4)	94% (18 / 19)	100% (2 / 2)	100% (8 / 8)	100% (5 / 5)
CSC	sp4	96%	100% (4 / 4)	92% (13 / 14)	100% (2 / 2)	100% (8 / 8)	100% (4 / 4)
ECMWF	sp4	100%	100% (4 / 4)	100% (14 / 14)	100% (2 / 2)	100% (8 / 8)	100% (4 / 4)
EPCC	sp5	90%	100% (4 / 4)	85% (12 / 14)	100% (2 / 2)	100% (8 / 8)	75% (3 / 4)
FZJ	sp4	100%	100% (4 / 4)	100% (19 / 19)	100% (2 / 2)	100% (8 / 8)	100% (5 / 5)
IDRIS	sp4	97%	100% (4 / 4)	94% (18 / 19)	100% (2 / 2)	100% (8 / 8)	100% (5 / 5)
LRZ	a3k	85%	100% (3 / 3)	66% (4 / 6)	100% (2 / 2)	87% (7 / 8)	100% (1 / 1)
RZG	sp4	100%	100% (4 / 4)	100% (14 / 14)	100% (2 / 2)	100% (8 / 8)	100% (4 / 4)
SARA	a3k	90%	100% (3 / 3)	100% (7 / 7)	100% (2 / 2)	75% (6 / 8)	100% (1 / 1)

Figure 1: INCA Summary view

Software monitoring for the DCPE

package	version	cne-sp5	csc-sp4	ecm-sp4	epc-sp5	fzj-sp4	idr-sp4	lrz-a3k	rzg-sp4	sur-a3k
aix_applications										
cpmd2cube [download]					[help]					[back to top]
cpmd2cube	=jan05	jan05	jan05	jan05	jan05	jan05	jan05		jan05	
torb [download]					[help]					[back to top]
torb	=1.22	1.22	1.22	1.22	1.22	1.22	1.22		1.22	
aix_compilers										
java [download]					[help]					[back to top]
java	=1.4	1.4.2	1.4.2	1.4.2	1.4.2	1.4.2	1.4.2		1.4.2	
xlC [download]					[help]					[back to top]
xlC	=7.0	7.0	7.0	7.0	7.0	7.0	7.0		7.0	
xlc [download]					[help]					[back to top]
xlc	=7.0	7.0	7.0	7.0	7.0	7.0	7.0		7.0	
xlf [download]					[help]					[back to top]
xlf	=9.1	9.1	9.1	9.1	9.1	9.1	9.1		9.1	
aix_libraries										
blacs [download]					[help]					[back to top]
blacs	=3	3.2.0.1	3.2.0.0	3.1.0.2	3.2.0.0	3.2.0.0	3.2.0.0		3.2.0.0	
blacssmp [download]					[help]					[back to top]
blacssmp	=3	3.2.0.1	3.2.0.0	3.1.0.2	3.2.0.0	3.2.0.0	3.2.0.0		3.2.0.0	
essl [download]					[help]					[back to top]
essl	=4	4.2.0.4	4.2.0.4	4.1.0.1	4.2.0.4	4.2.0.2	4.2.0.2		4.2.0.4	
esslsmp [download]					[help]					[back to top]
esslsmp	=4	4.2.0.4	4.2.0.4	4.1.0.1	4.2.0.4	4.2.0.2	4.2.0.2		4.2.0.4	
mass [download]					[help]					[back to top]
mass	=4.1	4.1.0.4	4.1.0.4	4.1.0.0	4.1.0.3	4.3.0.1	4.1.0.4		4.1.0.0	
pessl [download]					[help]					[back to top]
pessl	=3	3.2.0.1	3.2.0.0	3.1.0.2	3.2.0.0	3.2.0.0	3.2.0.0		3.2.0.0	
pesslsmp [download]					[help]					[back to top]
pesslsmp	=3	3.2.0.1	3.2.0.0	3.1.0.2	3.2.0.0	3.2.0.0	3.2.0.0		3.2.0.0	

Figure 2: INCA Stack view: details for each software package

Software monitoring for the DCPE

unit tests		cne-sp5	csc-sp4	ecm-sp4	epc-sp5	fzj-sp4	idr-sp4	lrz-a3k	rzg-sp4	sar-a3k
MPI-test1		passed				passed	passed			
MPI-test2		passed				passed	passed			
OpenMP					[help]					[back to top]
OpenMP	=x.x									
unit tests		cne-sp5	csc-sp4	ecm-sp4	epc-sp5	fzj-sp4	idr-sp4	lrz-a3k	rzg-sp4	sar-a3k
OpenMP		passed				passed	passed			
unix_scripting										
perl [download]					[help]					[back to top]
perl	=5.8	5.8.8	5.8.0	5.8.0	5.8.2	5.8.0	5.8.8	5.8.3	5.8.0	5.8.0
python [download]					[help]					[back to top]
python	=2.4	2.4.3	2.4	2.4.2	2.4.3	2.4.2	2.4.2	2.3.3	2.4.2	2.4.1
tcl [download]					[help]					[back to top]
tcl	=8.4	8.4b1	8.4.4	8.4.12	8.4.12	8.4.12	8.4.12	8.4.6	8.4.12	8.3.5
tk [download]					[help]					[back to top]
tk	=8.4	8.4b1	8.4.4	8.4.12	8.4.12	8.4.12	8.4.12	8.4.6	8.4.12	8.3.5
unix_shells										
bash [download]					[help]					[back to top]
bash	=2,3	3.00.0(1)	3.00.16(1)	3.1.0(3)	2.05a.0(1)	3.00.16(1)	3.00.0(1)	2.05b.0(1)	3.1.5(1)	2.05b.0(1)
tcsh [download]					[help]					[back to top]
tcsh	>=6.11	6.11.00	6.11.00	6.14.00	6.11.00	6.11.00	6.13.00	6.12.00	6.11.00	6.12.00
unix_tools										
emacs [download]					[help]					[back to top]
emacs	=20,21	21.1.1	21.1.1	21.3.1	21.3.1	21.2.2	21.1.1	21.3.1	20.2.1	21.3.1
gmake [download]					[help]					[back to top]
gmake	=3	3.80	3.80	3.80	3.79.1	3.80	3.80	3.80	3.80	3.79.1
nedit [download]					[help]					[back to top]
nedit	=5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.4	5.5
openssh [download]					[help]					[back to top]
openssh	=3,4	3.8.1p1	3.8.1p1	3.6.1p2-CERT-patched	3.7.1p2-pwexp24	3.7.1p2-pwexp26_krb5	3.9p1	4.1p1	3.4p1	4.2p1

Figure 3: INCA Stack view: details for each software package

Software monitoring for the DCPE

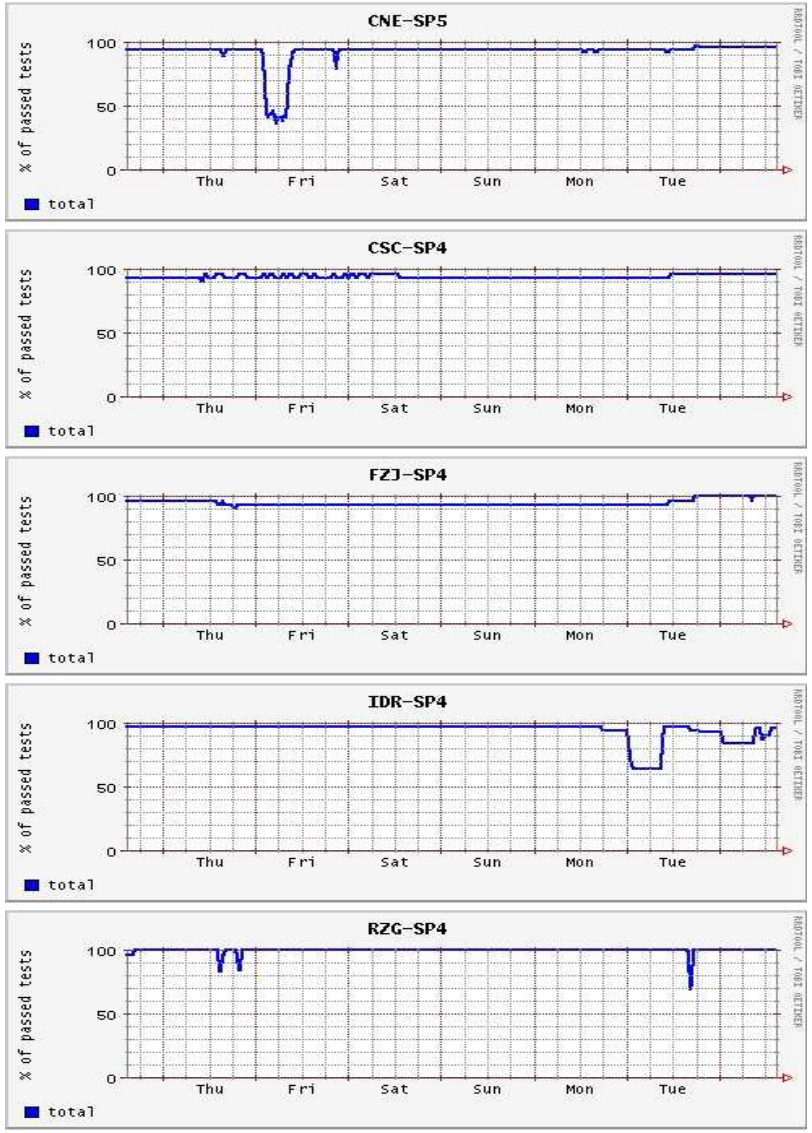


Figure 4: INCA Status graph: graphs describing the status in the last period