

Programming with OmpSs?-2

- Introduction
- Quick User Guide

Introduction

[OmpSs?](#)-2 is a programming model composed of a set of directives and library routines that can be used in conjunction with a high-level programming language in order to develop concurrent applications. Its name originally comes from two other programming models: **OpenMP** and [StarSs?](#). The design principles of these two programming models constitute the fundamental ideas used to conceive the [OmpSs?](#) philosophy.

OmpSs?-2 **thread-pool** execution model differs from the **fork-join** parallelism implemented in OpenMP.

File Systems

On the DEEP-EST system, three different groups of filesystems are available:

- the [JSC GPFS filesystems](#), provided via [JJUST](#) and mounted on all JSC systems;
- the DEEP-EST (and SDV) parallel BeeGFS filesystems, available on all the nodes of the DEEP-EST system;
- the filesystems local to each node.

The users home folders are placed on the shared GPFS filesystems. With the advent of the new user model at JSC ([?JUMO](#)), the shared filesystems are structured as follows:

- **\$HOME:** each JSC user has a folder under `/p/home/jusers/`, in which different home folders are available, one per system he/she has access to. These home folders have a low space quota and are reserved for configuration files, ssh keys, etc.
- **\$PROJECT:** In JUMO, data and computational resources are assigned to projects: users can request access to a project and use the resources associated to it. As a consequence, each user has a folder within each of the projects he/she is part of. For the DEEP project, such folder is located under `/p/project/cdeep/`. Here is where the user should place data, and where the old files generated in the home folder before the JUMO transition can be found.

The DEEP-EST system doesn't mount the \$SCRATCH and \$ARCHIVE filesystems, as it is expected to provide similar functionalities with its own parallel filesystems.

The following table summarizes the characteristics of the file systems available in the DEEP-EST and DEEP-ER (SDV) systems:

[illegible]

Stripe Pattern Details

It is possible to query this information from the deep login node, for instance:

```
manzano@deep $ fhgfs-ctl --getentryinfo /work/manzano
Path: /manzano
Mount: /work
EntryID: 1D-53BA4FF8-3BD3
Metadata node: deep-fs02 [ID: 15315]
Stripe pattern details:
+ Type: RAID0
+ Chunksize: 512K
+ Number of storage targets: desired: 4

manzano@deep $ beegfs-ctl --getentryinfo /sdv-work/manzano
Path: /manzano
Mount: /sdv-work
EntryID: 0-565C499C-1
Metadata node: deeper-fs01 [ID: 1]
Stripe pattern details:
+ Type: RAID0
+ Chunksize: 512K
+ Number of storage targets: desired: 4
```

Or like this:

```
manzano@deep $ stat -f /work/manzano
File: "/work/manzano"
ID: 0      Namelen: 255      Type: fhgfs
Block size: 524288      Fundamental block size: 524288
Blocks: Total: 120178676 Free: 65045470 Available: 65045470
Inodes: Total: 0        Free: 0

manzano@deep $ stat -f /sdv-work/manzano
File: "/sdv-work/manzano"
ID: 0      Namelen: 255      Type: fhgfs
Block size: 524288      Fundamental block size: 524288
Blocks: Total: 120154793 Free: 110378947 Available: 110378947
Inodes: Total: 0        Free: 0
```

See <http://www.beegfs.com/wiki/Striping> for more information.

Additional infos

Detailed information on the **BeeGFS Configuration** can be found [?here](#).

Detailed information on the **BeeOND Configuration** can be found [?here](#).

Detailed information on the **Storage Configuration** can be found [?here](#).

Detailed information on the **Storage Performance** can be found [?here](#).

Notes

- The /work file system which is available in the DEEP-EST prototype, is as well reachable from the nodes in the SDV (including KNLs and KNMs) but through a slower connection of 1 Gig. The file system is therefore not suitable for benchmarking or I/O task intensive jobs from those nodes
- Performance tests (IOR and mdtest) reports are available in the BSCW under DEEP-ER → Work Packages (WPs) → WP4 → T4.5 - Performance measurement and evaluation of I/O software → Jülich DEEP Cluster → Benchmarking reports: <https://bscw.zam.kfa-juelich.de/bscw/bscw.cgi/1382059>
- Test results and parameters used stored in JUBE:

```
user@deep $ cd /usr/local/deep-er/sdv-benchmarks/synthetic/ior
user@deep $ jube2 result benchmarks

user@deep $ cd /usr/local/deep-er/sdv-benchmarks/synthetic/mdtest
user@deep $ jube2 result benchmarks
```