Beginner's guide to eX3

James D. Trotter

High-performance Computing,
Simula Research Laboratory

eX3 tutorial

16 March 2022

simula

https://www.ex3.simula.no/

eX3 is an Experimental Infrastructure for Exploration of Exascale Computing

System stability

eX3 is an *experimental cluster:*

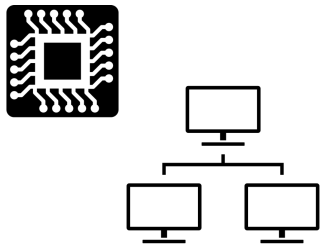
- can be **reconfigured and changed at any time**,
- can be **reserved for specific users** (for short periods)

Therefore:

- you **cannot expect the same uptime** as other systems,
- you **cannot store data or results for longer periods**.

https://www.ex3.simula.no/

Beginner’s guide to eX3



1. Hardware overview

2. Accessing eX3

slurm
workload manager

3. Getting started

[LAN by Stefan Traistaru from the Noun Project], [Microchip By Márcio Duarte, from the Noun Project], [DEC VT100 terminal by Gorthmog, CC BY-SA 4.0, via Wikimedia Commons]

A **highly heterogeneous cluster** with different computing hardware, network interconnects and storage solutions

CPUs:

- AMD Epyc (*Naples, Rome, Milan*)
- Intel Xeon (*Skylake Platinum/Gold/Silver, Cascadelake*)
- ARM Cavium ThunderX2
- HiSilicon Kunpeng

Accelerators:

- NVIDIA V100 and A100 GPUs
- AMD Vega20 and Instinct Mi100 GPUs
- Graphcore GC200 IPUs

Interconnect:

- 200 Gbps Infiniband HDR network
- Dolphin Interconnect PCIE network
- 10/25/100 Gbps Ethernet network

Storage:

- Up to 2 TB RAM per node
- BeeGFS parallel file system
- NVMe/SSD scratch storage

eX3 has more than 40 nodes with lots of different hardware

Compute nodes	Hostnames	Slurm partitions
2x AMD Epyc 7601, 2TB RAM, AMD Vega20 GPU, 4TB NVMe	`n001—n003`	`defq, amdgpuq`
2x AMD Epyc 7601, 2TB RAM, AMD Instinct Mi100 GPU, 4TB NVMe	`n004`	`defq, mi100q`
2x Cavium ThunderX2 CN9980, 1TB RAM, 5TB SSD	`n005—n008`	`armq`
2x HiSilicon Kunpeng 920-6426, 1TB RAM, 4TB SSD	`n009—n012`	`huaq`
2x AMD Epyc 7763, 2TB RAM, AMD Instinct Mi100 GPU, 4TB NVMe	`n013—n014`	`milanq, mi100q`
2x AMD Epyc 7763, 2TB RAM, NVIDIA A100 GPU, 4TB NVMe	`n015—n016`	`milanq, a100q`
2x AMD Epyc 7413	`n017—n020`	`fpgaq`
2x Intel Xeon Platinum 8186, 16x NVIDIA V100 GPUs	`g001`	`dgx2q`
2x AMD Epyc 7763, 2TB RAM, 8x NVIDIA A100 GPUs	`g002`	`hgx2q`
2x Intel Xeon Gold 6130	`srl-login1,srl-adm1, srl-mds1,srl-mds2`	`xeongold16q`
1x Intel Xeon Silver 4112	`n041—n048`	`slowq`
1x AMD Epyc 7302P, 256GB RAM	`n049—n060`	`rome16q`
2x AMD Epyc 7302P, 512GB RAM, Xilinx Alveo U250/U280	`n061—n064`	`fpga32q`
Graphcore IPUPOD64, 64x GC200 IPUs	`srl-login2`	`ipuq`

Beginner’s guide to eX3

Getting access to eX3

Start by requesting access to eX3: https://www.ex3.simula.no/access

Steps:

1. Request access at: *https://www.ex3.simula.no/access*.
2. You will receive an email from the system administrator with your username and password
3. Use an *SSH client* to login and change your password
    - Use, for example, OpenSSH on Linux, PuTTY on Windows or the builtin client on MacOS
4. Set up Public Key authentication
    - See, for example, the following tutorial on connecting to GitHub with SSH using Public Key authentication: https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh.

Login using SSH

1. Add the following to the SSH configuration file of your user on your own Linux or Mac PC (`~/.ssh/config`) :

```
Host ex3
  User <username>
  HostName dnat.simula.no
  Port 60441
```

2. Connect to the eX3 login node and change password:

```
~ $ ssh ex3
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-122-generic x86_64)

 General information:  http://srl-ex3.simula.no/dokuwiki/doku.php

[...]

 News and planned maintanence for cluster
 ****************************************************************
[...]

Last login: Tue Jul  6 14:33:39 2021 from 84.210.158.35
james@srl-login1:~$ passwd
(current) LDAP Password:
New password:
```

- You can use VPN instead of connecting to `dnat.simula.no`, as explained here:

  https://intranet.simula.no/accessing-simula-hosts-remote-location

- Windows users should configure their PuTTY client, as explained here:

  https://www.ssh.com/academy/ssh/putty/windows

Login using SSH and Public Key authentication

Public Key authentication allows secure login without requiring a password:

- Login to eX3 and add your **public key** to the file `~/.ssh/authorized_keys`:

```
# ~/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC2D04CXw9AkEgcfv+3sc1Hda2DbLYzW00CDuZLogYp1MYI6o/ZFmbR7DOJFZ3dTYrfX+fzF55
Tl4MmhoI4FxwacANPvWgZqeJEtDtkBzGPwR4Q0dJL5eFVoCT368Q19N4jv8uvl1BUFW/JFGdM+OSKWf1zkW1dwCPEQltcT9dU09zCJw
cLsFWkMXwJ+thFSSM1YTp9tgD0slHdWMpvONvLaCGsZmX/3IOyknrQWaMmzpF3muOvkHhEl7XwzVQCAVcaLIEuWSz1aoLOLQ0BbdHOz
8ZS8UtXelz3L8Qedij0zQhcwu/lfM1Rtj6tmnPHnEh5JPbdGvInhfKfYozwsj39FcKDmPpr3ZkiDPv0vhc7+yKpU3Ml12K2Owsd9f4u
eqn4DBy8UtAW47yp2nRTj9nWZ+HnzAirMvPkgf3bThJSh5DZzNlJuQ8A9FhyCUVGCKgx6i32ngh+JyuY5Fn0DhQGwXzBQExhIl6qXae
dP4iuu02XoZpkOY9TxQL+elNyYJqIldTfCJSruoeyW9lWqkeZt9FRV6uB+77XQIK0yjBt3nuKY3Ex7W/swzir/AMdIr9kH9j4QE1uwQ
zIrJ6dG7FZkdOs7sSiu3/1n3YAffwcZxAmy/jLWfdffWji/FvnUGH/gPNzRwbwF35JOGoubIidcAJ6UC34PgbGG4pq4bkZAQ==
openpgp:0xB429505
7ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBECAIlBfBLDfzw9HuwJrW76jMU5wux4xCGH5fELKW6w6xTNgxl6
EBa20qIyRp8JVm0/iLEF41vKWT6tAFdBxcBA= james@g001
```

- For further details, see, for example, the following tutorial on connecting to GitHub with SSH using Public Key authentication: https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh.

Beginner’s guide to eX3

Getting started on eX3:

Software

Basic examples: Compiling and running a C application, and using a Python interpreter

C example:

```
james@srl-login1:~$ cat hello.c
/* An example of a basic C application. */
#include <stdio.h>

int main(void) {
  printf("Hello, world!\n");
  return 0;
}
james@srl-login1:~$ gcc -o hello hello.c
james@srl-login1:~$ ./hello
Hello, world!
```

Python example:

```
james@srl-login1:~$ python3
Python 3.6.9 (default, Jan 26 2021, 15:33:00)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, World!')
Hello, World!
>>> <Ctrl+d> to exit
```

- You may compile your code on the login node, but you should only run VERY SHORT programs for testing purposes. All other programs should be submitted to eX3 using the Slurm queueing system (to be described later).

Use prebuilt software with **Environment Modules**

Environment modules is a commonly used system for managing software libraries and applications on HPC clusters:

- Loading a *module* changes a user's environment to make the relevant software package available
- Software can be installed once and shared by several users
- Different versions of the same software can be maintained

Basic commands:

- **`module avail`** – List modules that are installed on the system and which can be loaded
- **`module list`** – Display modules that are currently loaded
- **`module load`** – Load a module by modifying the user's environment
- **`module unload`** – Unload a module by resetting the user's environment
- For more details, see https://modules.readthedocs.io/en/latest/

Example: Loading pre-installed Python modules

```
james@srl-login1:~$ module use /cm/shared/ex3-modules/0.6.1/modulefiles
james@srl-login1:~$ module load python-3.7.4 python-numpy-1.19.2
Loading python-numpy-1.19.2
  Loading requirement: bzip2-1.0.8 xz-5.2.5 ncurses-6.1 readline-8.0
libffi-3.2.1 openssl-1.1.1c sqlite-3.31.1 python-3.7.4
    libgfortran-5.0.0 libstdcxx-6.0.28 openblas-0.3.12 lapack-3.9.0
fftw-3.3.8
james@srl-login1:~$ python3
Python 3.7.4 (default, Apr  1 2021, 09:45:00)
[GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```

Example: Using a different C compiler

```
james@srl-login1:~$ module use /cm/shared/ex3-modules/0.6.1/modulefiles
james@srl-login1:~$ module load intel/compiler/64/2019/19.0.5
james@srl-login1:~$ icc --version
icc (ICC) 19.0.5.281 20190815
Copyright (C) 1985-2019 Intel Corporation.  All rights reserved.

james@srl-login1:~$ icc -o hello hello.c
james@srl-login1:~$ ./hello
Hello, World!
```

Many other compilers are also available—check the list of available modules.

You may still need to install your own software…

Need help installing new software?

- Our eX3 system administrator, Tore Larsen, can sometimes help to install new software. Please send an email to: ex3-helpdesk@simula.no.

- We have also collected a lot of software installation scripts in a GitHub repository: https://github.com/jamtrott/ex3modules.

Beginner’s guide to eX3

Getting started on eX3:

Scheduling jobs

eX3 uses the Slurm workload manager for allocating resources and scheduling jobs on the cluster

Parallel and distributed computing terminology:

- **Process:** An instance of a computer program with its own memory address space
- **Distributed memory:** Multiple processors, where each processor has its own *private* memory
- **Shared memory:** Multiple processors sharing a single memory (typically within a single node)
- **Message passing interface (MPI):** A standardised programming model for distributed memory parallel computing

Slurm Terminology:

- **Node:** A machine belonging to a cluster
- **Task:** A process that is assigned to a node by the Slurm workload manager.
  There can be ***multiple tasks per node***, but each task is a *separate process*.
- **CPU:** A CPU core on a node with multiple cores.
  There can be ***multiple CPUs per task***, for programs that use *multithreading*.

eX3 uses the Slurm workload manager for allocating resources and scheduling jobs on the cluster

Basic Slurm commands:

- **`salloc`** – Allocate resources, such as CPUs, GPUs, or entire compute nodes
- **`srun`** – Execute an application using allocated resources
- **`sbatch`** – Submit a *batch script* to execute commands when resources become available
- **`scancel`** – Free allocated resources or halt a running job
- **`squeue`** – Show current and queued allocations for all users

Documentation:

- For more details, see https://slurm.schedmd.com/tutorials.html
  - Quick start guide: https://slurm.schedmd.com/quickstart.html
  - Introduction to Slurm: http://www.youtube.com/watch?v=NH_Fb7X6Db0&feature=relmfu

eX3 uses the Slurm workload manager for allocating resources and scheduling jobs on the cluster

slurm®
workload manager

Job Submission

salloc - Obtain a job allocation.
sbatch - Submit a batch script for later execution.
srun - Obtain a job allocation (as needed) and execute an application.

--array=<indexes> (e.g. "--array=1-10")	Job array specification. (sbatch command only)
--account=<name>	Account to be charged for resources used.
--begin=<time> (e.g. "--begin=18:00:00")	Initiate job after specified time.
--clusters=<name>	Cluster(s) to run the job. (sbatch command only)
--constraint=<features>	Required node features.
--cpu-per-task=<count>	Number of CPUs required per task.
--dependency=<state:jobid>	Defer job until specified jobs reach specified state.
--error=<filename>	File in which to store job error messages.
--exclude=<names>	Specific host names to exclude from job allocation.
--exclusive[=user]	Allocated nodes can not be shared with other jobs/users.
--export=<name[=value]>	Export identified environment variables.
--gres=<name[:count]>	Generic resources required per node.
--input=<name>	File from which to read job input data.
--job-name=<name>	Job name.
--label	Prepend task ID to output. (srun command only)
--licenses=<name[:count]>	License resources required for entire job.
--mem=<MB>	Memory required per node.
--mem-per-cpu=<MB>	Memory required per allocated CPU.
-N<minnodes[-maxnodes]>	Node count required for the job.
-n<count>	Number of tasks to be launched.
--nodelist=<names>	Specific host names to include in job allocation.
--output=<name>	File in which to store job output.
--partition=<names>	Partition/queue in which to run the job.
--qos=<name>	Quality Of Service.
--signal=[B:]<num>[@time]	Signal job when approaching time limit.
--time=<time>	Wall clock time limit.
--wrap=<command_string>	Wrap specified command in a simple "sh" shell. (sbatch command only)

Accounting

sacct - Display accounting data.

--allusers	Displays all users jobs.
--accounts=<name>	Displays jobs with specified accounts.
--endtime=<time>	End of reporting period.
--format=<spec>	Format output.
--name=<jobname>	Display jobs that have any of these name(s).
--partition=<names>	Comma separated list of partitions to select jobs and job steps from.
--state=<state_list>	Display jobs with specified states.
--starttime=<time>	Start of reporting period.

SchedMD
Slurm Support and Development

sacctmgr - View and modify account information.
Options:

--immediate	Commit changes immediately.
--parseable	Output delimited by '|'

Commands:

add <ENTITY> <SPECS> create <ENTITY> <SPECS>	Add an entity. Identical to the **create** command.
delete <*ENTITY*> where <*SPECS*>	Delete the specified entities.
list <ENTITY> [<SPECS>]	Display information about the specific entity.
modify <*ENTITY*> where <*SPECS*> set <*SPECS*>	Modify an entity.

Entities:

account	Account associated with job.
cluster	*ClusterName* parameter in the *slurm.conf*.
qos	Quality of Service.
user	User name in system.

Job Management

sbcast - Transfer file to a job's compute nodes.

sbcast [options] SOURCE DESTINATION

--force	Replace previously existing file.
--preserve	Preserve modification times, access times, and access permissions.

scancel - Signal jobs, job arrays, and/or job steps.

--account=<name>	Operate only on jobs charging the specified account.
--name=<name>	Operate only on jobs with specified name.
--partition=<names>	Operate only on jobs in the specified partition/queue.
--qos=<name>	Operate only on jobs using the specified quality of service.

Example: Use *sbatch* to submit a job on AMD Epyc (*defq*)

```
#!/bin/bash
# Basic usage:
#
#  $ sbatch hello.sbatch
#
# The job is queued and starts as soon as resources are available. The
# script is then executed on one of the allocated tasks, and standard
# output and standard error streams will be redirected to files that
# are prefixed by the job ID and job name. Commands prefixed with
# `srun' are executed on every task acquired by the job allocation.
#
# The sbatch options below allocate a single task on a single node,
# using a single CPU core with a one-hour time limit. To override
# these defaults, you can also supply sbatch options on the command
# line. For example:
#
#  $ sbatch --cpus-per-task=32 --time=02:00:00 hello.sbatch

#SBATCH --job-name="hello"
#SBATCH --partition=defq
#SBATCH --time=0-01:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --output=%j-%x-stdout.txt
#SBATCH --error=%j-%x-stderr.txt

echo ”1. Hello from $(hostname)"      # This command is run by ONE task
srun echo ”2. Hello from $(hostname)" # This command is run by EVERY task
```

1. These are *shell scripts*, so you can use ordinary Unix shell programming techniques.

2. Lines beginning with

   `#SBATCH <OPTION>`

   can be used to provide options to the Slurm workload manager.

3. Use **`srun`** to execute a command on every task that has been allocated for the job.

4. Standard output and error streams are redirected to files in the current working directory.

Example: Use *sbatch* to submit a job on AMD Epyc (*defq*)

```
james@srl-login1:~$ sbatch --partition armq hello.sbatch
Submitted batch job 167194
james@srl-login1:~$ ls
167194-hello-stderr.txt    167194-hello-stdout.txt    hello.sbatch
james@srl-login1:~$ cat 167194-hello-stdout.txt
1. Hello from n006
2. Hello from n006
james@srl-login1:~$ sbatch -p armq --nodes=2 --ntasks=2 hello.sbatch
Submitted batch job 167195
james@srl-login1:~$ cat 167195-hello-stdout.txt
1. Hello from n006
2. Hello from n006
2. Hello from n006
```

1. These are *shell scripts*, so you can use ordinary Unix shell programming techniques.

2. Lines beginning with

   `#SBATCH <OPTION>`

   can be used to provide options to the Slurm workload manager.

3. Use **`srun`** to execute a command on every task that has been allocated for the job.

4. Standard output and error streams are redirected to files in the current working directory.

Example: Use *sbatch* to submit a GPU job

```
james@srl-login1:~$ salloc --partition dgx2q --time=04:00:00 --gres=gpu:1
salloc: Pending job allocation 211047
salloc: job 211047 queued and waiting for resources
...
james@srl-login1:~$ srun --pty /bin/bash --login
james@g001:~$ nvidia-smi
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.156.00   Driver Version: 450.156.00   CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM3...  Off  | 00000000:34:00.0 Off |                    0 |
| N/A   55C    P0   275W / 350W |  25310MiB / 32510MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

1. Use **`salloc --gres=gpu:1`** to allocate a GPU.
2. Use **`srun --pty`** to get an interactive terminal.

Beginner’s guide to eX3

Getting started on eX3:

Best practices

Best practices for allocating resources on eX3

Please follow these guidelines for resource allocation:

1. Always allocate resources using Slurm—otherwise, you risk interfering with the work of other users
2. Do not let allocated resources stand idle or go unused—others could use those resources instead
3. Do not run CPU- or memory-demanding workloads on the login node
4. Set a reasonable *time limit* when allocating resources
   - use the `--time` option with `salloc`/`srun`/`sbatch`
5. Only use the CPU cores or GPUs that you have allocated
   - For example, do not spawn more threads than the number of CPU cores that were allocated.
6. Do not login to compute nodes with `ssh`.
   - Interactive jobs can be run with `srun --pty`, but this is rarely needed.

Managing files and storage

Please follow these guidelines for storing data on eX3:

- **`/global/D1`** is a global, parallel file system (BeeGFS), which *should be used for most storage*.
  - Every user has a home directory at **`/global/D1/homes/<user>`**
  - For data related to a project or shared by multiple users, project directories can be created under **`/global/D1/projects`**. Contact the system administrator (ex3-helpdesk@simula.no) if you would like to have a project directory.
- **`/home/<user>`**, the user's home directory, is a network file system with *limited capacity* and should only be used for very small amounts of data (e.g., source code, but *not* large data sets).
- **`/work`** is a fast, local storage on each node, only to be used for temporary data.

Where can I get help?

Got questions?

- Take a look at the eX3 wiki: http://wiki.ex3.simula.no/
- If you cannot find the answer to your question there, please contact the eX3 system administrator, Tore Larsen, by email to ex3-helpdesk@simula.no.

The eX3 project is hosted by Simula Research Laboratory and funded by the Research Council of Norway

Whenever you publish research based on the eX3 infrastructure, please notify us by email to ex3-contact@simula.no.

Consider acknowledging eX3 in your published research:

The research presented in this paper has benefited from the Experimental Infrastructure for Exploration of Exascale Computing (eX3), which is financially supported by the Research Council of Norway under contract 270053.

https://www.ex3.simula.no/publications

simula

The Research Council of Norway

With funding from
The Research Council of Norway