Online script generator Documentation

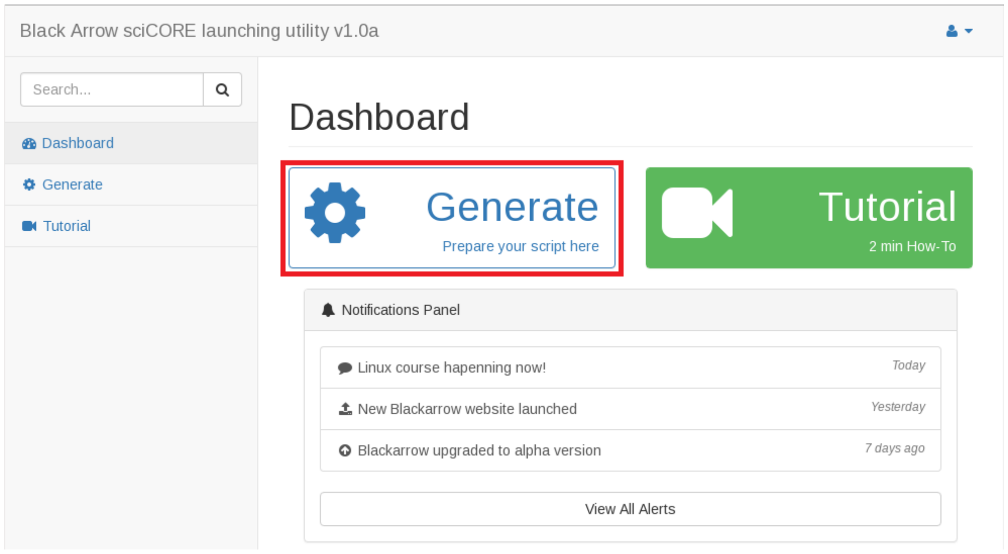
- General overview
- First steps
- Using OSG
- Additional information
- Queues and maximum runtime
- Document history

This is a guide for the usage of the online script generator (OSG) utility. With this web-based tool you can learn how to create a script to launch your applications in a cluster using different schedulers

> Disclaimer
> This is a learning tool that efficiently bridges the know-how gap that is present among many new users when starting to work with distributed computing facilities. Therefore, it is not meant to be used as a production tool.

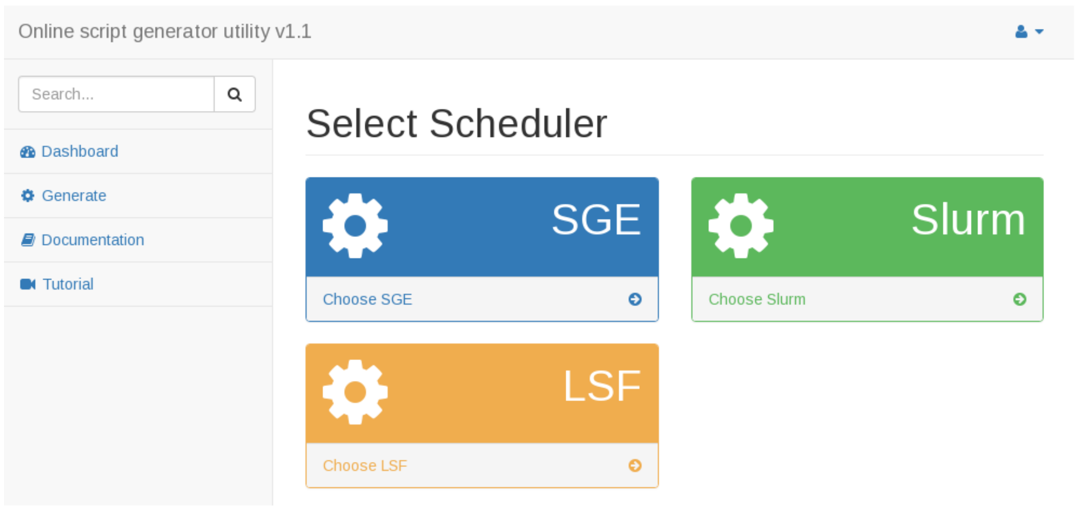
General overview

1. Open your internet browser and go to: http://scriptgen.scicore.unibas.ch
2. You will see the Dashboard. Click in Generate .

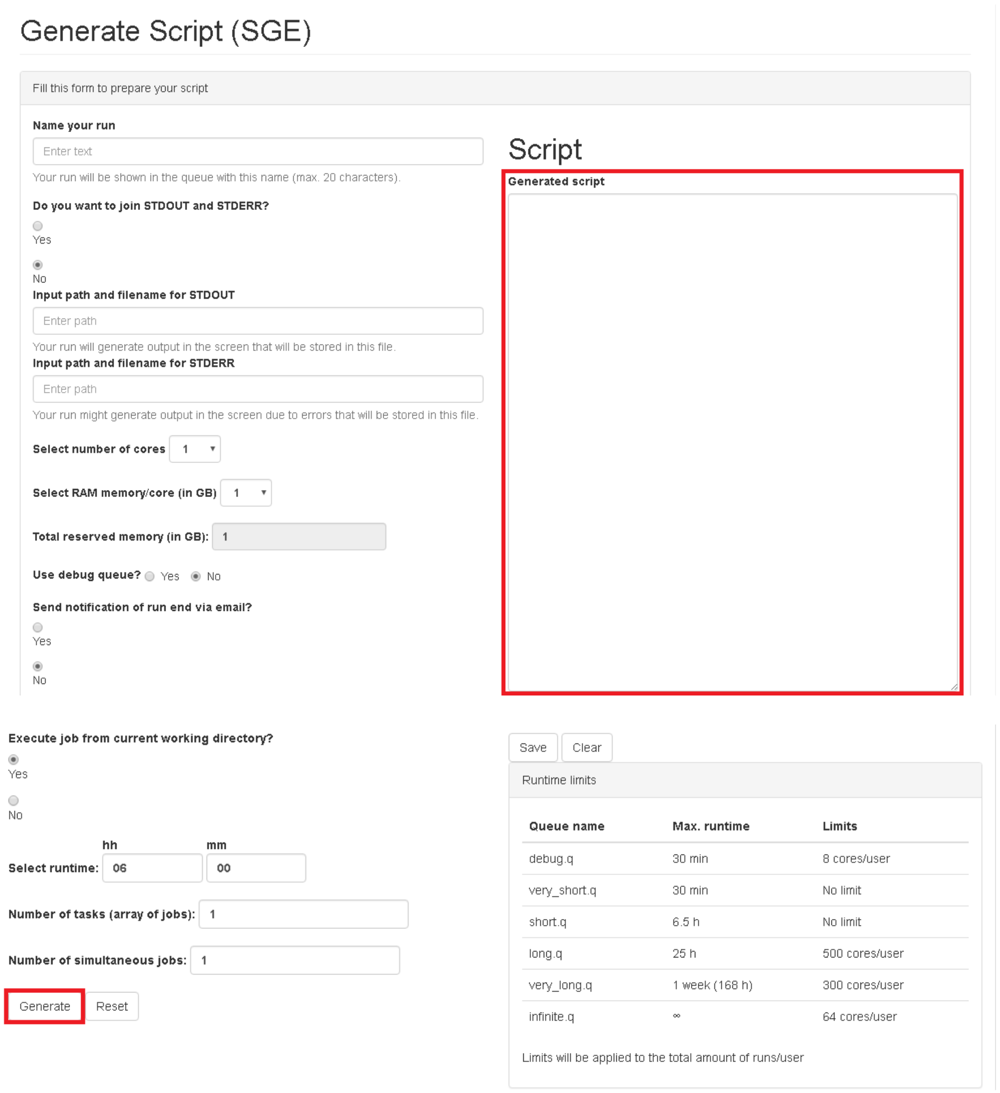


Linux course hapenning now! Today
New Blackarrow website launched Yesterday
Blackarrow upgraded to alpha version 7 days ago

3. OSG can handle different schedulers. If the following page appears, choose the scheduler that you want to use for your submission scripts. Otherwise, depending on the installation by your system administrator, it might automatically jump to the scheduler that is relevant for the current system (point 4 below). In sciCORE/UniBas SGE and Slurm are used, while in Vital-IT/SIB the scheduler is LSF.



4. This is the main page of OSG. You will use this form to generate scripts that can be copy-pasted to create working scripts. We will cover the details of each field below. Once the form is filled up with the information that you need, click in Generate and the script will appear in the big text field Generated Script. In the following we use the SGE scheduler as an example to explain the fields. These might vary from one scheduler to another, but the main concept behind is the same.



Generate Script (SGE)

Fill this form to prepare your script

Name your run: Enter text

Your run will be shown in the queue with this name (max. 20 characters).

Do you want to join STDOUT and STDERR? Yes [✓ No]

Input path and filename for STDOUT: Enter path

Your run will generate output in the screen that will be stored in this file.

Input path and filename for STDERR: Enter path

Your run might generate output in the screen due to errors that will be stored in this file.

Select number of cores: 1

Select RAM memory/core (in GB): 1

Total reserved memory (in GB): 1

Use debug queue? Yes [✓ No]

Send notification of run end via email? Yes [✓ No]

Execute job from current working directory? [✓ Yes] No

Select runtime: hh 06 mm 00

Number of tasks (array of jobs): 1

Number of simultaneous jobs: 1

Runtime limits

Queue name	Max. runtime	Limits
debug.q	30 min	8 cores/user
very_short.q	30 min	No limit
short.q	6.5 h	No limit
long.q	25 h	500 cores/user
very_long.q	1 week (168 h)	300 cores/user
infinite.q	∞	64 cores/user

Limits will be applied to the total amount of runs/user

First steps

Starting to use OSG is very simple. You just need to click Generate . This will create a script with default values:

```
#!/bin/bash
#$ -N myrun
#$ -pe smp 1
#$ -l membycore=1G
# Total memory reserved: 1GB
#$ -l runtime=06:00:00
#$ -o $HOME/myrun.o$JOB_ID
#$ -e $HOME/myrun.e$JOB_ID
#$ -cwd

#load your required modules below
############################

#export your required environment variables below
#########################################

#add your command lines below
##########################
```

As you can see, the generator gave default values to several fields that were not filled up: name (myrun), number of cores (1), reserved memory per core (1GB), runtime (6h), and standard output and standard error location files (in you home directory with names myrun.o$JOB_ID and myrun.e$JOB_ID). It also tells you where you should add the commands to load required modules (for example compilers, libraries, third-party software, etc...), to export environment variables, and to add your actual command lines (precisely what you want to calculate in the cluster).

Nevertheless, it is very likely that you need to reserve different resources depending on the tasks that you want to perform. To do that, you have to fill the form up.

Using OSG

Field	Default	Description	Command example	Tips
Name your run	myrun	This is simply the name that your task will receive when showed up in the queue. The name is limited to 20 characters. No spaces are allowed.	#$ -N	Use a name that helps you to recognize what is actually the task about

Do you want to join STDOUT and STDERR?	No	STDOUT and STDERR stand for Standard Output and Standard Error. These are two files in which the system will write down any output that your task would have generated in the screen if it was launched interactively. STDOUT stores all output related to the task running command and STDERR stores all error messages emitted by the system during the time your task is running and related to it. If you choose No , two independent files will be created. If you choose Yes , the STDERR and STDOUT output will be stored jointly with the STDOUT file. No spaces are allowed.	#$ -j y	Use environment variables to define the path of these files (f.e. $HOME) instead of writing static paths. In this way, your script may still work if the name of the paths (f.e. your home path) change.
Select number of cores	1	This reserves a number of cores to do your calculation. The default parallel environment is SMP, meaning that all calculations are performed in the same node. Therefore, 64 cores is the maximum value. For massive parallel calculations using MPI, please contact us.	#$ -pe smp	
Select RAM memory/core	1 GB	Reserve this amount of memory per core. If you reserve more than 1 core the total reserved memory is multiplied! You can see the total reserved memory in the next field, as a reminder of the amount that is being requested by your script.	#$ -l membycore=1G	
Use debug queue?	No	The debug queue is a very short queue that is meant for development and debugging of your scripts. Therefore, only testing runs should be submitted here.	#$ -q debug.q	
Send notification of run end via email?	No	If you choose Yes , a new field will appear for you to input your email. Then the scheduler will send you an email after your job has finished or aborted.	#$ -m ea -M <email>	Use this option with caution. If you launch thousands of jobs you won't like to receive thousands of emails!
Execute job from current working directory?	Yes	If you choose No , a new field will appear for you to input the path where the scheduler can find the executable of your command lines. No blank spaces are allowed.	#$ -cwd	
Select runtime	6h	Here you input the estimated time that your task require. The time that you specify here will select the queue in which your task will run. You can find the runtime limits in a table in the OSG page and here below.	#$ -l runtime=06:00:00	It is important to be as accurate as possible in this, so that the scheduler can efficiently back-filing the available resources. As a consequence, your tasks will be executed sooner.
Number of tasks (array of jobs)	1	Array jobs is an efficient way of submitting many independent tasks to the scheduler with one single script. The number here determines the amount of tasks to be done.	#$ -t 1-100	OSG assumes that the array of jobs always starts on 1, but this can be changed directly in the script.

Number of simultaneous jobs	1	If you launched an array of many jobs, it is advisable to group them in batches, so that you always have a reasonable number of tasks running. The number specified here determines the amount of maximum simultaneously running tasks of your array of jobs. Obviously, it cannot be bigger than the total amount of tasks.	#$ -tc 10	

Additional information

- Reset button will clear the form but not any script previously generated.
- Clear button will clear the previously generated script.
- Save button will save the previously generated script in the local computer.

Queues and maximum runtime

Queue Name	Max. Runtime	Limits
debuq.q	30 min	8 cores/user
very_short.q	30 min	No limit
short.q	6.5 h	No limit
long.q	25 h	500 cores/user
very_long.q	1 week (168 h)	300 cores/user
infinite.q	∞	64 cores/user

Document history

Date	Author	Comment
28-jun-2016	Rubén Cabezón	Changed name from Blackarrow to OSG.
13-abr-2016	Rubén Cabezón	Original Document