SUN GRID ENGINE TUTORIAL pdf

1: Grid Scheduler / Sun Grid Engine HOWTOs

Notice that Sun Grid Engine automatically named the job hostname and created two output files: hostname.e1 and hostname.o1. The e stands for stderr and the o for stdout. The 1 at the end of the files' extension is the job number.

Go to start of metadata Introduction This is a short overview of how to use the batch queuing system on the CCBB clusters. In the tutorial below, text in blocks colored, and set off from the main text like this This is a sample line. Basic Overview Processing data on the cluster is a 5 step process consisting of Naming the processing run, and creating a project directory Uploading data files to project directory Devising proper processing scheme What programs are to be used What needs to be copied over to the node for processing What needs to be copied back to be saved. Write, and submit job script; or submit request for interactive login to a node Once processing is done review logs to make sure no problems occurred The last step is critical. You cannot assume that because your job finishes it actually completed successfully. Any number of things can go wrong, such as a node crashing, a resource disk, memory being exhausted, programming errors, or usage errors. We cannot predict or control when these things happen so you should be prepared for that. As mentioned below the output and error messages generated during a job run on the cluster are captured into output, and error files for you to review. We can help write others for you as needed, or you can use these as a template. Each of our scripts has several variables that tell the script where to find source data, config files, and other parameters that alter the job. Changing just these few items is enough to have a workable script that you can submit. Submitting Jobs In some cases, you may just want to run a very quick job, or otherwise just need to run some commands without packaging them up in a job script. If the footprint of the jobs is light, then they can just be run directly on the headnode after logging in. If you run commands on the script,and those have any amount of significant amount of runtime, please monitor their CPU usage. If we let the too many CPU hogs pile up on the headnode it will quickly become very painful for interactive users trying to move files around, write scripts, and submit them. Here is an example. Your interactive job has been successfully scheduled. Sun Nov 18 By far the nicest way to run jobs, is to write a job script and submit it for a non-interactive run. This lets you exit out of the head node, and walk away waiting until the job is done running. Here is a simple script, in a file called tut1,! If there are slots available, then the job will get run, and this job should exit very quickly after that. We can make one quick simplification to the script. Thus, we can modify tut1 to become tut2! Now look back in your home directory. You should see two files named something like this: Create local working directory Transfer input files data and science config files to working directory Perform science processing Copy results back to home directory Here is a bare bones script that accomplishes this. By default this shows a list of jobs running in the queue and their state. The output in this case is much more verbose, and includes information about the state of the job, and queuing considerations. One final option is to use the -f option to see the status of the queues on the systems. This is true whether the job is still in the wait state, or is running.

SUN GRID ENGINE TUTORIAL pdf

2: Ubuntu Server -- how to install Sun Grid Engine? - Ask Ubuntu

Tutorial to use a Computing Cluster with Sun Grid. Engine software (SGE) Alvaro Sebastian Yague based on the notes written by Biostatistics Information Technology (BIT) Committee.

The problem A common problem is that you have a large number of jobs to run, and they are largely identical in terms of the command to run. For example, you may have data sets, and you want to run a single program on them, using the cluster. The naive solution is to somehow generate shell scripts, and submit them to the queue. This is not efficient, neither for you nor for the head node. Array jobs are the solution There is an alternative on SGE systems â€" array jobs. You put less of a burden on the head node. Submitting an array job to do computations is entirely equivalent to submitting separate scripts, but much less work for you. The basic commands In this section, I assume that you prefer the bash shell. To review, a basic SGE job using bash may look like the following: Assume you have input files input. You could use perl to generate shell scripts, submit them, then clean up the mess later. Or, you could use an array job. The modification to the previous shell script is simple: When the above script is submitted, it will find available nodes, and jobs will execute in order of the task IDs specified by the -t option. Also, the array job is subject to all the fair queueing rules. The above script is entirely equivalent to submitting scripts, but without the mess. A more complex example This is a modification of the above which only runs the program if the output file is not present. For example, the numbers could be random number seeds for a simulation. This can be accomplished by using the Unix awk command: For example, if your jobs all use the same program, but with very different command-line options, you can list all the options in the file, one set per line, and the exercise is basically the same as the above, and you only have two files to handle or 3, if you have a perl script generate the file of command-lines. Alternatives using cat, head and tail, or sed or perl are of course also possible. What if you number files from 0 instead of 1? However, sometimes you need to use R to analyze your data. In order to do this, you have to hardcode file names into the R script, because these scripts are not interactive. This is a royal pain. However, there is a solution that makes use of HERE documents in bash. HERE documents also exist in perl, and an online tutorial for them in bash is at http: The short of it is that a HERE document can represent a skeleton document at the end of a shell script. You have data files, labeled data. Each file contains a single column of numbers, and you want to do some calculation for each of them, using R. If anyone experiments and figures out something neat, let me know. Be aware of the following: In my limited experience, indenting is important for HERE documents. In particular, it seems that the beginning and end i. So, if you use a HERE document in a conditional or control statement, be mindful of this. In the mean command, I escaped the dollar sign with a backslash. However, escaping the dollar sign for the read. This is more useful than just array jobs on an SGE system. If you know bash well enough, you can write a shell script that takes a load of arguments, and processes them with a HERE document. This solves a major limitation with R scripts themselves. You can do the same in perl, too, on your workstation, but you must use a shell language on the cluster. Using Rscript Since around R For submitting jobs this may be cleaner than using a shell script with a HERE document. The only thing you have to take care of is to use the -shell no -b yes options to avoid using a shell like tcsh. R library somelibrary etc. Submitting would be done as follows: In practice you would of course have to supply other arguments to your R script. This is best done using some command line argument parser personally I prefer the simple one by Vincent Zoonekynd, show at the bottom of this post.

SUN GRID ENGINE TUTORIAL pdf

3: installation of sun grid engine on linux

Sun Grid Engine Tutorial The Sun Grid Engine (SGE) is a queue and scheduler that accepts jobs and runs them on the cluster for the user. There are three types of jobs available: interactive, batch, and parallel.

Using a queuing system in these situations has the following advantages: Scheduling - allows you to schedule a virtually unlimited amount of work to be performed when resources become available. This means you can simply submit as many tasks or jobs as you like and let the queuing system handle executing them all. Also allows querying job history to see which tasks were executed on a given date, by a given user, etc. You can run your tasks across the cluster in any way you see fit and the queuing system should not interfere. However, you will most likely end up needing to implement the above features in some fashion in order to optimally utilize the cluster. A job may have specific resource requirements but in general should be agnostic to which node in the cluster it runs on as long as its resource requirements are met. Note All jobs require at least one available slot on a node in the cluster to run. Submitting jobs is done using the qsub command. In this case the command hostname is a single binary. This option takes a y or n argument indicating either yes the command is a binary or no it is not a binary. The -cwd option to qsub tells Sun Grid Engine that the job should be executed in the same directory that qsub was called. The last argument to qsub is the command to be executed hostname in this case Notice that the qsub command, when successful, will print the job number to stdout. After a few seconds, the job will transition into a r, or running, state at which point the job will begin executing. For the simple job submitted above we have: The e stands for stderr and the o for stdout. To do this, we use the qhost command: This is useful for simple jobs but for more complex jobs where we need to incorporate some logic we can use a so-called job script. D" As you can see, this script simply executes a few commands such as echo, date, cat, etc and exits. Since this is just a bash script, you can put any form of logic necessary in the job script i. Do not remove the following line or programs that require network functionality will fail Had something failed, such as a command not found error for example, these errors would have appeared in the stderr file. You can delete a job from the queue using the qdel command in Sun Grid Engine. This integration allows Sun Grid Engine to handle assigning hosts to parallel jobs and to properly account for parallel jobs. You can inspect the SGE parallel environment by running: This defines how to assign slots to a job. This means that if a job requests 8 slots for example, it will go to the first machine, grab a single slot if available, move to the next machine and grab a single slot if available, and so on wrapping around the cluster again if necessary to allocate 8 slots to the job. With this rule, if a user requests 8 slots and a single machine has 8 slots available, that job will run entirely on one machine. If 5 slots are available on one host and 3 on another, it will take all 5 on that host, and all 3 on the other host. In other words, this rule will greedily take all slots on a given node until the slot requirement for the job is met. Compile the code using mpicc, mpicxx, mpif77, mpif90, etc Copy the resulting executable to the same path on all nodes or to an NFS-shared location on the master node Note It is important that the path to the executable is identical on all nodes for mpirun to correctly launch your parallel code. Run the code on X number of machines using: Instead of using the above formulation create a simple job script that contains a very simplified mpirun call: The above example requests 24 slots or processors using the orte parallel environment. You can also do this without a job script like so:

SUN GRID ENGINE TUTORIAL pdf

4: Sun Grid Engine for Linux - Computer How To

Grid Engine Enterprise Edition Sun Grid Engine, Enterprise Edition -- Configuration Use Cases and Guidelines Scheduler Policies for Job Prioritization in the N1 Grid Engine 6 System.

Create a directory to hold your job file and any associated data matlab scripts, etc. Open a new file, in this case we will call it matlab-test. Save this job script and submit to the queue with qsub matlab-test. When the job is completed, you can check the output of the job in the filename given above, matlabTest. You may see the following in the output: Thus no job control in this shell. This is not the place to run long running, very computationally intensive, or other jobs better suited to run in a batch job. You can launch an interactive job, develop the script and write the job file. But when it comes to running the job itself, it needs to be submitted as a batch job. To run an interactive job, simply type qlogin. Running Parallel Jobs with SGE A parallel job is where a single job is run on many nodes in an interconnected fashion, generally using MPI to communicate in between individual processes. If you are running the same program on the cluster as you would on your desktop, chances are you will want to use a serial job, not a parallel job. Parallel jobs generally are only for specially designed programs which will only work on machines with cluster management software installed. Also, not just any program can run in parallel; it must be programmed as such and compiled against a particular MPI library. In this case, we build a simple program that passes a message between processes and compiles it against the OpenMPI, the main MPI library of the cluster. Note that the scheduler will only accept parallel jobs between 4 to 8 slots. It is currently setup to start parallel processes on a single node to limit the overhead of inter-process communication over the network, which adds considerable runtime to the job. For most jobs, more slots is not always best. Like the batch job, create a directory to hold this job and related files. Open a new file and create the job script. And like above, you can use "qsub" to check on your job Below are a few commands to know: There is a known bug in the scheduler that sometimes causes it to not respond, resulting in the following message: Simply wait a minitue and retry your command again. These variables are listed below: To set up your environment to use mpich2 instead of openmpi, you will have to alter your shell environment. Launching an mpich2 Job.

SUN GRID ENGINE TUTORIAL pdf

5: Simple-Job-Array-Howto - GridWiki

In this tutorial we will deal with everything related to job submission, scheduling, and execution on a cluster under the control of Sun Grid Engine software2 (SGE). The Grid Engine project was sponsored by Sun Microsystems3 as an open source community effort to facilitate the adoption of distributed computing solutions.

Univa now own the copyright â€” see below. The idea is to encourage sharing, in the spirit of the original project, informed by long experience of free software projects and scientific computing support. Please contribute, and share code or ideas for improvement, especially any ideas for encouraging contribution. Currently most information you find for the gridengine v6. Note that this changes the communication protocol due to the MUNGE support, and really should have been labelled 8. The sge one actually only goes back to the V60 tag â€” i. Only the repository trunk was converted in each case, but see History. Not really relevant for a distribution, although some of it can probably be converted into useful documentation for distribution later; review top-level directory: Just a set of tick sheets for Sun internal use; gep top-level directory: The portal, which may not actually be functional now, and is anyway based on a proprietary server; HTML documentation: The head of the source tree is actually available directly via the Repository URL links for each repository, and if you really need all the current source without a darcs or hg client, this should do the trick: You can access the repos with Mercurial hg , e. You can also pull with git, e. Also note that pulls with git or hg leave scripts not executable, but the initial aimk fixes that. Since darcs is patch based, the ordering of patches e. The parallel "release" repo does have a linear history with at least recent release tag if you want to track actual releases. Building Building the source is rather a pain, which is something to improve at high priority, although there is a spec file for RPM-based systems which may need adjusting for non-RHEL ones. To avoid Java stuff, try: Bug reporting, patches, and mail lists There is a subscribers-only general mail list sge-discuss liv. You can report bugs by mail in the convenient time-honoured fashion to sge-bugs liv. They are gatewayed to the issue tracker and to a list. You can create an account to put tickets in the tracker via a web form, or use the "tourist" account with "sge" as the password. The best way to send patches is to record them in a clone darcs repository and use darcs send. Until arrangements are made for direct contributions, they will be pushed into the main repository by hand if they look reasonable. Patches against the git or hg repository mirrors are also fine, but are more likely to need merging by hand. Otherwise, you can attach normal patches made against whatever source you have but preferably a recent release or snapshot to tickets or just mail them see Contact. If you do record changes with darcs, make the patch name a useful short summary of the patch, and include the number of any ticket it fixes in the form fix N, where N is the ticket number. Oracle promised to donate artifacts from the old site, but never did. The active IZ issues have been transferred to the issue tracker here, though some have somewhat suffered in the process; unfortunately I only discovered after the fact that they were accessible as an XML dump. Material here has an explicit or implicit copyright under the same terms, i. The original source code is obviously under the original rather odd! Note that, contrary to multiple claims, there is no evidence that any Sun or other material here is being distributed without an appropriate licence allowing it. Reports of licensing bugs are very welcome, of course. This is a superset of their functionality, though the code may be different in parts, e.

SUN GRID ENGINE TUTORIAL pdf

6: Sun Grid Engine (SGE) QuickStart â€" StarCluster vrc2 documentation

Sun Grid Engine for beginners. Using the BIMSB (soon to be called MAX) cluster environment is similar to using unix/linux environments for your job submission (e.g running your scripts or other software).

Usually you will have to formally request to your research center or faculty for it. After that you will be provided with an user, password and the domain name or IP of the cluster. Computer clusters are useful to perform high-performance calculations, that means calcula- tions that will require a normal computer running during several days, weeks or months. If a calculation last only several hours it is not worth to use a computer cluster. Also it is impor- tant to highlight that it is required some advanced knowledge of computer science to use the cluster infraestructure, average computer users without previous training will be not able to use properly the cluster neither to take advantage of it. Computer clusters can be used for research in physics, chemistry, computer science, statistics, engineering, aeronautics, computational biology and bioinformatics, among other topics. It is important to know how many nodes, cores or slots and RAM memory has the cluster. Each node is like an individual computer that will have several cores commonly called proces- sors or slots and some amount of RAM memory that should be shared among all the cores. For example, a cluster with 20 nodes of 40 cores and 60GB per node will have in total cores and 1. In this example each core will be able to use as maximum 60GB, but if we have the 40 cores working simultaneously, each core will be allowed to use only 1. Furthermore, clusters are not homogeneous and they can have nodes with different amounts of cores and memory. This topic will be explained later in subsection 4. We will call job to every single process, task, program, script or interactive session that we run or execute in the cluster. In this tutorial we will deal with everything related to job submission, scheduling, and execution on a cluster under the control of Sun Grid Engine software2 SGE. The Grid Engine project was sponsored by Sun Microsystems3 as an open source community effort to facilitate the adoption of distributed computing solutions. Among other things, SGE can be used to limit the total number of slots each user is allowed to run simultaneously on the cluster. You may submit jobs which request more than the slot limit and they will all be queued to run as your other jobs finish. When the cluster nodes are all at maximum capacity, jobs waiting to run may be subject to a functional share priority algorithm as we have defined it using SGE. Secure Shell SSH is a cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers that connects, via a secure channel over an insecure network, a server and a client running SSH server and SSH client programs, respectively. The best- known application of the protocol is for access to shell accounts on Unix-like operating systems, 2http: It was designed as a replacement for Telnet and other insecure remote shell protocols such as the Berkeley rsh and rexec protocols, which send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet. There are many options for Windows, perhaps the most popular and free is Putty4. For Linux systems OpenSSH5 is the prefered one, installing it in an Ubuntu system is very easy other distributions have similar ways to install: RSA key fingerprint is 3c: You will be prompted to enter your password. You should enter the password provided by the cluster administrator. If you have any problem following the former instructions, you can try to connect in verbose mode to try to find more info about errors: This machine is not for doing any sort of computation. Rather, it is ONLY for managing files, text editing and for submitting jobs to the whole cluster. So how to run any program in the cluster? This answer will be answered in 4http: Now we will talk about how to copy files between our local machine and the cluster. There are some differences between them: To be able to use one of them, it must be installed in the remote machine as server and you have to install in your computer the client utility in a similar way than SSH client. You must specify username and the IP or Domain Name of the SSH server the cluster , also the connection port if different than default Local and remote folder are the scp command arguments and they must be in the right order, first origin and then destination folder for the file copy: Most of the time you can get away with just knowing a few commands. The most immediately relevant

ones are: This can be done using the SGE command qrsh. This command essentially opens up a remote shell or runs a program on a cluster node. By default qrsh will use only one slot core in some of the configured nodes for this task. Now you can run whatever program you want. For example, you can run R interactive shell or compiling some code. Otherwise, you will be taking up a slot in the queue which will not be available to others. To come back to the access node, just type: You may also specify memory requirements, multi-core or special queues on your qrsh command just as you do on the qsub command subsection 4. If you still have running programs and no session appears for you in qstat subsection 4. If it is, get the job-ID and kill the job using qdel. So, all input data are preselected through scripts, command-line parameters, or job control language. This is in contrast to interactive jobs which prompt the user for such input. In batch processing a program takes a set of data files as input, processes the data, and produces a set of output data files. Batch processing has some benefits: It can shift the time of job processing to when the computing resources are less busy. It avoids idling the computing resources with minute-by-minute manual intervention and supervision. By keeping high overall rate of utilization, it amortizes the computer, especially an ex- pensive one. It allows the system to use different priorities for batch and interactive work. SGE comes with a number of command line tools to help you manage your jobs 6 on the cluster. Not to worry, wrapping your program with a shell script is not as difficult as it might sound! Below are instructions for how to run a R batch job on the cluster. Prints a listing of all options. The name of the job. Defines or redefines the time and date at which a job is eligible for execution. Execute the job from the current working directory. Execute the job from the directory specified in working dir. Defines or redefines the priority of the job relative to other jobs. Priority is an integer in the range to The default priority value for jobs is 0. Specific resource request list. Will be explained in subsection 4. Doing so will result in your program not running. Your job will be assigned a job id and will be appear if you immediately do a qstat command. But, after a minute or so, the job will cancel and a message will be written in an error file. While you are logged into a cluster node via qrsh, if you start another session and run qstat form the access node: But the main utility of qstat is checking the status of jobs submitted via qsub. By default, qstat with no arguments shows the status of your current cluster jos: Some of the codes are: Another important thing to note is the job-ID for your job. You need to know this if you ever want to make changes to your job. For example, to delete your job from the cluster as will be explained in the next section. Tue Sep 17 To delete a job from the cluster, you can obtain the job-ID running qstat subsection 4. After calculating approximately how much memory your job will need, you should add a memory resource requirement to your qsub or qrsh command. Here is one example explaining why you should specify memory requeriment even if you expect your job to use as little as 3G of memory. It is possible that a user has requested 18G of memory and is using all of that or more on a 20G node. That user may be the only one on that node and if it has 8 cpu-cores there could be 7 slots open. If you send in that moment a job without memory requirement, your job might go to that node since it looked lightly loaded as far as jobs. It will happily accept your job and begin swapping and slow down both jobs now on the node. To solve this problem system administrators usually define a default minimum memory for each submitted job. But if you know your job will consume less memory, you should specify a lower limit to allow more jobs to be ran at the same time. This does not reserve memory for your job. It simply puts your job on a node with that amount of memory currently available. Sometimes a node will have enough free memory available but it will be assigned to another job requirements and will not be able for ours. It is advisable that all users use this parameter to stop a runaway job from crashing the node, so as a rule is also set as default by the system administrator. It is useful to avoid the consequences of any bug in their program that might, under certain conditions, cause a file to grow without bounds. In the following example, it is required a minimum of 4G of RAM memory on a node to start the job, the job will be stopped if it consumes more than 6G of memory or create files bigger than 1 GB. It would behave similarly on a qrsh command. After submitting your job with qsub, remember to use the qstat command to check which queue node your job actually went to see subsection 4. We should prepare a faster example of our job and do some running tests with or without qsub.

7: Grid Engine documents and source

SUN GRID ENGINE TUTORIAL pdf

SUN Grid Engine Introduction. The SUN Grid Engine is a batch www.enganchecubano.com submit jobs which are placed in queues and the jobs are then executed, depending on the system load, the opening hours of the queues and the job priority.

8: SUN Grid Engine

Before you begin, ensure that you have the host names of all hosts in the grid. Create a separate list of host names for each type of host in the grid: Become superuser of the cluster node where you are installing Sun Grid Engine. Install the Sun Grid Engine distribution files. You have to choose.

9: An Intro to SGE

I am trying to search around a tutorial to install Sun Grid Engine on Ubuntu Server without much success. I came to this article.

SUN GRID ENGINE TUTORIAL pdf

Some flight and wind-tunnel longitudinal stability measurements on the BAC slender-wing aircraft Davenport analytic number theory Van Lills South African miscellany My very own potty! History of racism in america Math makes sense 7 workbook answers Genius the natural history of creativity The kafir project pirat Cultural citizenship in the age of P2P networks William Uricchio Literature-based reading programs at work NATEF Standard Jobsheet A1 (Natef Standards Job Sheets) Desire and expectations Ritual as ideology Decline and fall, 1899-1903 The best of Grant MacEwan Claritas investment certificate study material The Colt Engraving Book A Submarine Forest The effects of Goodmans nominalism and worldmaking on his aesthetics. Introduction to the study of national music Genesis and effect of the Popular Front in France Darwinian dynamics Mine health and safety engineering Reduce Your Breast Cancer Risks Short-term financial management The preseason: it happens before it happens Directions for use Fences: posthumous justice Life of Charles T. Walker . Degas and the business of art Congressional directory, 2003-2004 All-Amer Chili Book A handbook of human resource management practice 12th edition 2016 whole dog journal wet food What is and what is not a practice? Cape Town and environs street guide The reviews reviewed They built for eternity. Using cooperative learning and classroom research with culturally diverse students Susan Obler . [et al.] An Outline of Set Theory