Система управления заданиями TORQUE.

1. Введение.

Система управления заданиями Torque предназначена для управления запуском задач на многопроцессорных вычислительных установках (в том числе кластерных). Она позволяет автоматически распределять вычислительные ресурсы между задачами, управлять порядком их запуска, временем работы, получать информацию о состоянии очередей. При невозможности запуска задач немедленно, они ставятся в очередь и ожидают, пока не освободятся нужные ресурсы.

Несмотря на то, что в TORQUE имеется встроенная программа-планировщик (scheduler), **pbs_sched**, обычно он используется единственно как система управления заданиями (resource manager) совместно с планировщиком, делающим запросы к нему. Система управления заданиями обеспечивает низкоуровневый набор фунциональных возможностей для старта, поддержания, отмены и монторинга заданий. Не обладающий этими возможностями планировщик в одиночку не может управлять задачами.

TORQUE главным образом используется на многопроцессорных вычислительных установках. Объединение ресурсов в вычислительных установках обычно уменьшает необходимость в постоянном управлении ресурсами для пользователей. Настроенная однажды правильно вычислительная установка абстрагируется от многих деталей, связанных с запуском и управлением заданиями. Пользователю обычно надо установить в параметрах лишь минимальные требования к задаче, и ему нет необходимости знать даже имена вычислительных узлов, на которых задача выполняется.

Вычислительная установка включает в себя четыре типа различных компонент: * Главный узел * Submit/Interactive Nodes * Вычислительные узлы * Ресурсы

Главный узел

Вычислительная установка обязана иметь главный узел, на котором запущен **pbs_server**. В зависимости ои необходимости или параметров системы главный узел может быть предназначен только для этого или же также исполнять роль других компонент системы.

Submit/Interactive Nodes

Обеспечивают пользователям точку входа в систему, для того чтобы они могли управлять своим объемом работы. На этих узлах пользователи могут ставить на выполнение и отслеживать свои задачи. На них возможно выполнение клиентских команд (например, **qsub**, **qhold**).

Вычислительные узлы

Роль вычислительных узлов – выполнять поставленные задачи. На каждом из них работает **pbs_mom** для того, чтобы начинать, прекращать и управлять поставленными в очередь задачами.

Ресурсы

Система может быть организована так, что необходимо управлять распределением некоторых ресурсов (например, сети и запоминающие устройства) между вычислительными узлами. Наличие этих ресурсов ограничено и должно разумно распределяться для обеспечения повышения производительности.

Жизненный цикл задачи может быть разделе на четыре стадии: * создание * постановка в очередь* выполнение * завершение

Создание

Обычно пишется скрипт, включающий в себя все параметры задачи. Эти параметры включают в себя то, как долго задача может выполняться (**walltime**), какие ресурсы необходимы и собственно то, что требуется запускать. Пример файла, ставящего задачу в очередь:

```
#PBS -N localBlast

#PBS -S /bin/sh

#PBS -l nodes=1:ppn=2,walltime=240:00:00

#PBS -M user@my.organization.com

#PBS -m ea

source ~/.bashrc

cd $HOME/work/dir

sh myBlast.sh -i -v
```

Этот скрипт определяет имя задачи (localBlast), какое окружение использовать (/bin/sh), то что требуется использовать оба процессора на одном узле (nodes=1:ppn=2), то что выполнять задача будет максимум 10 дней, и то что TORQUE должен послать письмо на user@my.organization.com в случае ошибки или окончания выполнения задачи. Также пользователь указывает где и что собственно выполнять.

Постановка в очередь

Постановка в очередь производится командой **qsub**. Приоритет, выставляемый администратором, далее определяет время начала выполнения задачи.

Выполнение

Обычно занимает большую часть жизненного цикла задачи. По время выполнения статус задачи может быть запрошен командой **qstat.**

Завершение

По умолчанию, когда задача выполнена, файлы stdout и stderr копируются в директорию, откуда задача была поставлена в очередь.

Архитектура

- TORQUE кластер состоит из одного главного узла и многих вычислительных узлов. На главном узле запущен демон pbs_server, а на ввчислительных узлах pbs_mom демоны. Клиентские команды для постановки в очередь и управления задачами могут быть установлены на любой хост, включая те, на которых не выполняются pbs_server или pbs_mom.
- На главном узле также должен быть запущен демон планировщика. Он взаимодействует с pbs_server и по результатам распределяет ресурсы и рабочие узлы на задачи. В дистрибутив TORQUE включены простой планировщик и код, позволяющий создавать его более продвинутые версии, но предпочтительнее использовать Maui или Moab.
- Пользователи ставят задачи на выполнение в pbs_server, используя команду qsub. Когда pbs_server получает новую задачу, он сообщает об этом планировщику. Если (когда) планировщик находит узлы для этой задачи, он посылает инструкцию выполнять эту задачу вместе со списком узлов pbs_server, который в свою очередь посылает эту задачу первому узлу из списка. Этот узел назначается “execution host” или “Mother Superior”. Остальные узлы, выполняющие задачу, называются “sister moms.”

2. Управление TORQUE с точки зрения пользователя.

Постановка задачи в очередь

Осуществляется командой qsub. Эта команда принимает набор параметров, запсанных в командной строке, и объединяет их в командный файл PBS. Этот файл может задаваться в строке параметров команды **qsub**, или же он может быть введен через файл STDIN.

- PBS командный скрипт не обязан быть запускаемым.
- The PBS batch script may be piped into qsub (i.e., 'cat pbs.cmd | qsub')
- В случае параллельных задач PBS командный скрипт запускается на первом из назначенных узлов (для запуска на нескольких узлах используется pbsdsh).
- Командный скрипт в любом случае запускается из домашней директории пользователя (скрипт может определить директории, из который задача поставлена в очередь, используя переменную $PBS_O_WORKDIR)
- Командный скрипт запускается, используя набор значений пользовательских настроек по умолчанию, если не заданы флаги '**-V**' or **–v.**.
- Pbsdsh не допускает параметров в комндном скрипте.

По умолчанию, постановка задачи в очередь разрешена только с главного узла (того, на котором запущен pbs_server).

Запрос ресурсов

Различные ресурсы могут быть запршены во время постановки задачи в очередь. Задача может запросить конкретный узел, узел с определенными параметрами или же набор узлов с заданными параметрами. Также может быть задан планировщик (встроенный в TORQUE или внешний). Ниже приведен список ресурсов, которые позволяет запрашивать встроенный планировщик:

Ресурс	Формат	Описание
arch	String	Specifies the administrator defined system architecture required. This defaults to whatever the PBS_MACH string is set to in "local.mk".
cput	seconds, or [[HH:]MM:]SS	Maximum amount of CPU time used by all processes in the job
file	size*	The amount of total disk requested for the job
host	string	Name of the host on which the job should be run. This resource is provided for use by the site's scheduling policy. The allowable values and effect on job placement is site dependent.
mem	size*	Maximum amount of physical memory used by the job
nice	integer	between -20 (highest priority) and 19 (lowest priority). Adjust the process' execution priority
nodes	{<node_count> | <hostname>}	Number and/or type of nodes to be reserved for exclusive use by the job. The value is one or more node_specs joined with the '+'

	[:ppn=<ppn>][:<property>[:<property>]...] [+...]	character, “node_spec[+node_spec...]”. Each node_spec is an number of nodes required of the type declared in the node_spec and a name or one or more property or properties desired for the nodes. The number, the name, and each property in the node_spec are separated by a colon ‘:’. If no number is specified, one (1) is assumed. The name of a node is its hostname. The properties of nodes are: * ppn=# - specifying the number of processors per node requested. Defaults to 1. * property - a string assigned by the system administrator specify a node’s features. Check with your administrator as to the node names and properties available to you. See Example 1 (-l nodes) for examples. : By default, the node resource is mapped to a virtual node or in other words, maps directly to a processor, not a full physical compute node. This behavior can be changed within Maui or Moab by setting the JOBNODEMATCHPOLICY parameter.
opsys	string	Specifies the administrator defined system operating system as defined in the mom config file.
other	string	Allows a user to specify site specific information. This resource is provided for use by the site’s scheduling policy. The allowable values and effect on job placement is site dependent.
pcput	seconds, or [[HH:]MM:]SS	Maximum amount of CPU time used by any single process in the job
pmem	size*	Maximum amount of physical memory used by any single process of the job
pvmem	size*	Maximum amount of virtual memory used by any single process in the job
software	string	Allows a user to specify software required by the job. This is useful if certain software packages are only available on certain systems in the site. This resource is provided for use by the site’s scheduling policy. The allowable values and effect on job placement is site dependent. (see Scheduler License Management)
vmem	size*	Maximum amount of virtual memory used by all concurrent processes in the job
walltime	seconds, or [[HH:]MM:]SS	Maximum amount of real time during which the job can be in the running state

*size format: параметр «size» определяет максимальное значение в единицах байт или слов. Он записывается в виде *целое[суффикс]*. Суффикс – это множитель, определенный в нижеприведенной таблице. (’b’ означает байты (по умолчанию), а ‘w’ означает слова). The size of a word is calculated on the execution server as its word size.

Суффикс		**Множитель**
b	w	1

kb	kw	1024
mb	mw	1,048,576
gb	gw	1,073,741,824
tb	tw	1,099,511,627,776

Пример 1 (-l nodes)

Usage	Description
qsub -l `> qsub -l nodes=12`	Запрос на 12 узлов любого типа
qsub -l `> qsub -l nodes=2:server+14`	Запрос на 2 узла-“сервера” и 14 других узлов (в сумме 16) – это определяет два различных node_specs, “2:server” и “14”
qsub -l `> qsub -l nodes=server:hippi+10:noserver+3:bigmem:hippi`	Запрос на (а) 1 узел, который является “сервером” и имеет “hippi” интерфейс, (б) 10 узлов, не являющихся серверами, и (в) 3 узла, которые имеют большое количество памяти и “hippi” интерфейс
qsub -l `> qsub -l nodes=b2005+b1803+b1813`	Запрос на 3 особых узла по их hostname
qsub -l `> qsub -l nodes=4:ppn=2`	Запрос на 2 процессора на каждом из четырех узлов
qsub -l `> qsub -l nodes=1:ppn=4`	Запрос на 4 процессора на одном узле
qsub -l `> qsub -l nodes=2:blue:ppn=2+red:ppn=3+b1014`	Запрос на 2 процессора на каждом из двух “blue” узлов, трех процессоров на одном “red” узле и вычислительный узел “b1014”

Пример 2

```
qsub -l mem=200mb /home/user/script.sh
```

Эта задача запрашивает узел с 200 Мб доступной памяти.

Пример 3

```
> qsub -l nodes=node01,mem=200mb /home/user/script.sh
```

Эта задача будет ждать до тех пор, пока не освободится узел node01 с 200 Мб доступной памяти.

Запрос Generic ресурсов

When **generic** resources have been assigned to nodes using the server’s nodes file, these resources can be requested at the time of job submission using the other field.

Пример 4

```
> qsub -l other=matlab /home/user/script.sh
```

Задача запустится на любом узле, на котором имеется generic ресурс ***matlab***.

Замечание: То же самое можно запросить во время постановки задачи в очередь, используя флаг**-W x=GRES:matlab**.

Запрос других ресурсов

Когда задача запущена, командный скрипт использует набор переменных для принятия определенных решений, создания выходных файлов и т.д. Эти переменные представлены в этой таблице:

Переменная	Описание
PBS_JOBNAME	user specified jobname
PBS_O_WORKDIR	job's submission directory
PBS_ENVIRONMENT	N/A
PBS_TASKNUM	number of tasks requested
PBS_O_HOME	home directory of submitting user
PBS_MOMPORT	active port for mom daemon
PBS_O_LOGNAME	name of submitting user
PBS_O_LANG	language variable for job
PBS_JOBCOOKIE	job cookie
PBS_NODENUM	node offset number
PBS_O_SHELL	script shell
PBS_O_JOBID	unique pbs job id
PBS_O_HOST	host on which job script is currently running
PBS_QUEUE	job queue
PBS_NODEFILE	file containing line delimited list on nodes allocated to the job
PBS_O_PATH	path variable used to locate executables within job script

Мониторинг задач

TORQUE позволяет пользователям и администраторам контролировать ход выполнения задачи с помощью команды qstat. Если запрос выполняется пользователем, то на экран будут выведены данные только о задачах этого пользователя. Например,

```
> qstat

Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
4807             scatter          user01           12:56:34 R batch
...
```

Отмена задач

TORQUE позволяет пользователям и администраторам отменять поставленные в очередь команды с помощью команды qdel. Заданию будут посланы TERM и KILL сигналы, убивающие запущенные процессы. Когда высокоуровневый скрипт задачи удаляется, тогда и происходит выход из задачи.

Если задача отменяется администратором, пользователю будет отправлено извещение по почте. Администраторы могут добавлять комментарии к этому сообщению с помощью опции –m.

```
$ qstat
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
4807             scatter          user01           12:56:34 R batch
...
$ qdel -m "hey! Stop abusing the NFS servers" 4807
$
```

Прерывание обслуживания задачи

TORQUE поддерживает прерывание обслуживания задачи, что позволяет пользователям приостанавливать и возобновлять задачи. Это реализовано двумя методами. Если поддержка прерывания обслуживания задачи включена в операционную систему, то TORQUE выяснит это во время процесса конфигурации и запустит. В противном случае, MOM может быть настроена так, чтобы запускать «custom checkpoint script» с целью поддержки прерывания задачи. Но это требует того, чтобы задача понимала как возобновить себя после возникшего прерывания.

Настройка Checkpoint скрипта, используя MOM

Чтобы настроить MOM для поддержки checkpoint скрипта, $checkpoint_script параметр должен быть установлен в конфигурационном файле MOM, расположенном в $TORQUEHOME/mom_priv/config. Checkpoint скрипт должен иметь набор разрешений для запуска. Типичный пример:

mom_priv/config

```
$pbsserver          node06
$logevent           255
$restricted         *.mycluster.org
$checkpoint_script  /opt/moab/tools/mom-checkpoint.sh
```

Во-вторых, надо изменить значение MOM_CHECKPOINT на 1 в .../src/include/pbs_config.h. В некоторых случаях, MOM_CHECKPOINT может уже быть определен равным 1. Новая строка должна выглядеть так:

.../src/include/pbs_config.h

```
#define MOM_CHECKPOINT 1
```

Завершение задач

TORQUE предоставляет возможность сообщать о статусе выполненной задачи на протяжении некоторого времени (настраивается) после того, как задача была выполнена. Эту опцию можно включить, установив атрибут **keep_completed.** Этот параметр должен быть установлен равным количеству секунд, сколько задача должна находиться в очереди. Состояние выполненных задачах будет сообщаться как “C” состояние, а причина, по которой закончилось выполнение задачи можно посмотреть в “exit_status” данной задачи.

3. Управление TORQUE с точки зрения администратора.

Добавление узлов.

TORQUE может добавлять и удалять узлы или динамически, используя qmgr, или вручную редактируя файл*$TORQUE_HOME/server_priv/nodes*.

Динамические изменения узлов

TORQUE может динамически добавлять узлы, используя команду qmgr. Например, эта команда добавит узел “node003”:

```
> qmgr -c "create node node003"
```

Это дополнит файл $TORQUE_HOME/server_priv/nodes строкой:

```
node003
```

Можно удалять узлы похожей командой:

```
> qmgr -c "remove node node003"
```

Хотя обычно, администратору скорее удобно изменить состояние узла, а не удалять его.

!!! В версии TORQUE, актуальной на момент описания 2.1.6, эта возможность была реализована ненадежно. Поэтому настоятельно рекомендуется после изменения узлов перезапускать pbs_server или же просто вручную редактировать список узлов, а потом перезапускать сервер.

Свойства узлов

TORQUE может связать определенные свойства с узлами, чтобы помочь в определении групп узлов. Это типично для вычислительной установки объединять неоднородные наборы ресурсов. Чтобы определить различные наборы, свойства могут быть даны каждому узлу из набора. TORQUE может устанавливать, обновлять или удалять свойства или динамически., используя qmgr, или вручную, редактируя файл **nodes**.

Динамические изменения узлов

TORQUE позволяет динамически изменять свойства узла, используя команду qmgr. Например, чтобы дать “node001” свойства “bigmem” and “dualcore”:

```
> qmgr -c "set node node001 properties = bigmem"
> qmgr -c "set node node001 properties += dualcore"
```

Чтобы отказаться от установленного свойства используется “-=” оператор.

Ручное изменение узлов

Свойства каждеого узла перечислены в **$TORQUEHOME/server_priv/nodes**. Нужная особенность должна быть прописана после имени узла. Например, чтобы дать узлу “node001” свойства “bigmem” и “dualcore”, а узлу “node002” свойства “bigmem” и “matlab”, слкдует отредактироватьфайл **nodes** следующим образом:

server_priv/nodes

```
node001 bigmem dualcore
node002 np=4 bigmem matlab
```

Для активации изменений, сделанных в этом файле, необходимо перезапустить **pbs_server.**

Изменение состояния узла

Обычной задачей является предотвращение выполнения задачи на опреленном узле, помечая его как **offline** с помощью **pbsnodes -o nodename**. После того как узел был помечен «offline», планировщик далее не будет рассматривать узел как доступный для выполнения новых заданий. Для продолжения использования этого узла следует набрать команду **pbsnodes -c nodename**.

Также может быть полезна команда **pbsnodes –l,** которая which выводит на экран список всех узлов с состояниями “down”, “unknown” или “offline.” Это быстрый способ установить, на каких узлах могут быть проблемы.

Безопасность

Для систем, которым необходим доступ к вычислительным узлам, TORQUE предоставляет механизм, позволяющий только пользователям, у которых запущена задача, заходить на вычислительные узлы. Чтобы реализовать эту возможность:

- Untar `contrib/pam_authuser.tar.gz` (found in the src tar ball)
- Compile `pam_authuser.c` with **make** and **make install** on every compute node.
- Edit `/etc/system-auth` as described in `README.pam_authuser`, again on every compute node.
- Either make a tar ball of the `epilogue*` and `prologue*` scripts (to preserve the symbolic link) and untar it in the `mom_priv` directory, or just copy `epilogue*` and `prologue*` to `mom_priv/`.

The `prologue` scripts are perl scripts that add the user of the job to `/etc/authuser`. The `epilogue` scripts then remove the first occurance of the user from `/etc/authuser`. File locking is employed in all scripts to eliminate the chance of race conditions. Also, in the `epilogue` scripts, there is code that is commented out that when activated kills all processes owned by the user (using **pkill**), when that user does not have another valid job on the same node.

Конфигурация очереди

В TORQUE конфигурирование очереди выполнякtся, используя команду qmgr. Первый шаг – это создание очереди. Это выполняется, используя субкоманду **create queue** или **qmgr** как в примере:

```
> qmgr -c "create queue batch queue_type=execution"
```

После создания очередь должна быть настроена как рабочая. Как минимум, это означает установка параметров **started** и **enabled**. Дальнейшая настройка возможна, исполбзуя любые комбинации из атрибутов, приведенных в таблице.

Для логических атрибутов «T», «t», «1», «Y» и «y» все являются синонимами с «true», а «F», «f», «0», «N» и «n» все означают «false».

Для queue_type «E» and «R» являются сининимами с «Execution» и «Routing».

Очереди могут быть удалены, используя команду **delete** в qmgr.

Атрибуты очередей

Атрибут	Формат	Описание	Пример
acl_groups	<GROUP>[@<HOST>][+<USER>[@<HOST>]]...	specifies the list of groups which may submit jobs to the queue. If **acl_group_enable** is set to true, only jobs with an effective group listed in acl_groups may utilize the queue. : If the **acl_group_sloppy** queue attribute is set, **acl_groups** will be checked against all secondary groups of the job owner.	> qmgr -c “set queue batch acl_groups=staff” > qmgr -c “set queue batch acl_groups+=ops@h2” > qmgr -c “set queue batch acl_groups+=staff@h3” **Note**: used in conjunction with **acl_group_enable**)
acl_group_enable	<BOOLEAN>	if **TRUE**, constrains TORQUE to only allow jobs submitted from groups specified by the acl_groups parameter. **DEFAULT**: FALSE	> qmgr -c “set queue batch acl_group_enable=true”
acl_group_sloppy		if **TRUE**, acl_groups will be check against all groups of which the job user is a member. **DEFAULT**: FALSE	
acl_hosts	<HOST>[+<HOST>]...	specifies the list of hosts which may submit jobs to the queue	> qmgr -c “set queue batch acl_hosts=h1+h2+h3” **Note**: used in conjunction with **acl_host_enable**
acl_host_enable	<BOOLEAN>	if TRUE, constrains TORQUE to only allow jobs submitted from hosts specified by the **acl_hosts** parameter. **DEFAULT**: FALSE	> qmgr -c “set queue batch acl_host_enable=true”
acl_users	<USER>[@<HOST>][+<USER>	specifies the list of users which may submit jobs to the queue. If acl_user_enable is set to **TRUE**, only users listed in **acl_users** may	> qmgr -c “set queue batch acl_users=john” > qmgr -c “set queue batch acl_users+=steve@h2”

	[@<HOST>]]...	utilize the queue	> qmgr -c “set queue batch acl_users+=stevek@h3” **Note**: used in conjunction with **acl_user_enable**
acl_user_enable	<BOOLEAN>	if **TRUE**, constrains TORQUE to only allow jobs submitted from users specified by the acl_users parameter. **DEFAULT**: FALSE	> qmgr -c “set queue batch acl_user_enable=true”
acl_logic_or		if **TRUE**, user and group acls are logically OR’d together, meaning that either acl may be met to allow access. If false or unset, then both acls are AND’d, meaning that both acls must be satisfied. **DEFAULT**: FALSE	
enabled	<BOOLEAN>	specifies if the queue accepts new job submissions. **DEFAULT**: FALSE	> qmgr -c “set queue batch enabled=true”
keep_completed		specifies the number of seconds jobs should be held in the Completed state after exiting. Defaults to 0.	
kill_delay	<INTEGER>	specifies the number of seconds between sending a SIGTERM and a SIGKILL to a job being cancelled. **DEFAULT**: 2 seconds	> qmgr -c “set queue batch kill_delay=30”
max_queuable	<INTEGER>	specifies the maximum number of jobs allowed in the queue at any given time (includes idle, running, and blocked jobs). **DEFAULT**: unlimited	> qmgr -c “set queue batch max_queuable=20”
max_user_queuable	<INTEGER>	specifies the maximum number of jobs, per user, allowed in the queue at any given time (includes idle, running, and blocked jobs). **DEFAULT**: unlimited. Version 2.1.3 and greater.	> qmgr -c “set queue batch max_user_queuable=20”
max_running	<INTEGER>	specifies the maximum number of jobs in the queue allowed to run at any given time. **DEFAULT**: unlimited	> qmgr -c “set queue batch max_running=20”
max_user_running	<INTEGER>	specifies the maximum number of jobs, per user, in the queue allowed to run at any given time. **DEFAULT**: unlimited	> qmgr -c “set queue batch max_running=20”
priority	<INTEGER>	specifies the priority value associated with the queue. **DEFAULT**: 0	> qmgr -c “set queue batch priority=20”
queue_type	one of **e**, **execution**, **r**, or route	specifies the queue type. **Note**: This value must be explicitly set for all queues.	> qmgr -c “set queue batch queue_type=execution”
resources_	<STRING	specifies to cumulative resources	> qmgr -c “set queue batch

available	>	available to all jobs running in the queue. **DEFAULT**: N/A	resources_available.nodect=20” **Note:** pbs_server must be restarted for changes to take effect. Also, resources_available is constrained by the smallest of queue.resources_available and the server.resources_available.
resources_default	<STRING>	specifies default resource requirements for jobs submitted to the queue. **DEFAULT**: N/A : Default **walltime** and **nodes** resources is recommended.	> qmgr -c “set queue batch resources_default.walltime=3600”
resources_max	<STRING>	specifies the maximum resource limits for jobs submitted to the queue. **DEFAULT**: N/A : max **nodes** resource is meaningless.	> qmgr -c “set queue batch resources_max.nodect=16”
resources_min	<STRING>	specifies the minimum resource limits for jobs submitted to the queue. **DEFAULT**: N/A : min **nodes** resource is meaningless.	> qmgr -c “set queue batch resources_min.nodect=2”
route_destinations	<queue>[@<host>] [+<queue>[@<host>]]...	specifies the potential destination queues for jobs submitted to the associated routing queue. **Note**: This attribute is only valid for routing queues. **DEFAULT**: N/A	> qmgr -c “set queue route route_destinations=fast” > qmgr -c “set queue route route_destinations+=slow”
started	<BOOLEAN>	specifies if jobs in the queue are allowed to execute. **DEFAULT**: FALSE	> qmgr -c “set queue batch started=true”

Resources могут включать в себя одно или более их следующих параметров: **arch**, **mem**, **nodes**, **ncpus**, **nodect**, **pvmem**, и **walltime**

Пример конфигурации очереди

Следующая серия из команд **qmgr** создает и настраивает очередь, названную “batch”:

```
qmgr -c "create queue batch queue_type=execution"
qmgr -c "set queue batch started=true"
qmgr -c "set queue batch enabled=true"
qmgr -c "set queue batch resources_default.nodes=1"
qmgr -c "set queue batch resources_default.walltime=3600"
```

Эта очередь будет принимать новые задачи и, если это явно не указано в задаче, будет присваивать параметру nodecount значение 1 и параметру walltime значение 1 час на каждую задачу.

Установка очереди по умолчанию

По умолчанию, задача должна явно указывать, в какой очереди ей выполняться. Чтобы это изменить, нужно указать серверный параметр default_queue may, как в следующем примере:

```
qmgr -c "set server default_queue=batch"
```

Mapping очереди к подмножеству ресурсов

TORQUE в настоящее время не позволяет привязывать очереди к узлам. Однако, такие планировщики, как Moab и Maui, могут это обеспучить.

Простейший метод – это использование default_resources.neednodes на выполняемой очереди, связывая с ней атрибут определенного узла. Maui/Moab будет использовать эту информацию, чтобы убедиться, что задачи в этой очереди будут назначены на выполнение на узлах с этим атрибутом. Например:

```
$TORQUE_HOME/server_priv/nodes:
node01 np=2 chem
node02 np=2 chem
node03 np=2 bio
node04 np=2 bio

qmgr:
set queue chem resources_default.neednodes=chem
set queue bio  resources_default.neednodes=bio
```

Замечание: этот пример не предотвращает использования другими очередями этих узлов. Одно из решений – это использование какого-то другого generic атрибута со всеми остальными узлами и очередями.

Создание очереди-марштрутизатора

Очередь- маршрутизатор (routing queue) направляет задачу очереди-адресату (destination queue), основываясь на атрибутах задачи и состоянии очередей. Очередь-маршрутизатор устанавливается созданием очереди типа queue_type Route с route_destinations атрибутом, настроенным как в нижеприведенном примере.

```
# routing queue
create queue route
set queue route queue_type = Route
set queue route route_destinations = reg_64
set queue route route_destinations += reg_32
set queue route route_destinations += reg
set queue route enabled = True
set queue route started = True

# queue for jobs using 1-15 nodes
create queue reg
set queue reg queue_type = Execution
set queue reg resources_min.ncpus = 1
set queue reg resources_min.nodect = 1
set queue reg resources_default.ncpus = 1
set queue reg resources_default.nodes = 1
set queue reg enabled = True
set queue reg started = True
```

```
# queue for jobs using 16-31 nodes
create queue reg_32
set queue reg_32 queue_type = Execution
set queue reg_32 resources_min.ncpus = 31
set queue reg_32 resources_min.nodes = 16
set queue reg_32 resources_default.walltime = 12:00:00
set queue reg_32 enabled = True
set queue reg_32 started = True

# queue for jobs using 32+ nodes
create queue reg_64
set queue reg_64 queue_type = Execution
set queue reg_64 resources_min.ncpus = 63
set queue reg_64 resources_min.nodes = 32
set queue reg_64 resources_default.walltime = 06:00:00
set queue reg_64 enabled = True
set queue reg_64 started = True

# have all jobs go through the routing queue
set server default_queue = batch
set server resources_default.ncpus = 1
set server resources_default.walltime = 24:00:00
...
```

В этом примере вычислительные узлы двухпроцессорные, а время выполнения задачи по умолычанию выставляется в зависимости от количества процессоров/узлов, занятых выполнением задачи. Задачам, использующим 32 узла (64 процессора) или более будет дано по умолчанию время на выполнение 6 часов. Задачам, использующим 16-31 узлов (31-62 процессора) будет дано по умолчанию время на выполнение 12 часов. Всем остальным задачам время на выполнение присваивается равным server default walltime, то есть 24 часа.

Порядок route_destinations важен. Очередь-маршрутизатор присваивает задачу первой из доступных очередей-адресатов, основываясь на resources_max, resources_min, acl_users, и acl_groups attributes. В вышеприведенном примере атрибуты работы, задействующей один процессор, будут сначала проверены на соответствие очереди reg_64, затем очереди reg_32 и, наконец, очереди reg.

Добавление следующих настроек в вышеприведенную конфигурацию разъясняет требования очередей к ресурсам:

```
set queue reg resources_max.ncpus = 30
set queue reg resources_max.nodect = 15

set queue reg_16 resources_max.ncpus = 62
set queue reg_16 resources_max.ncpus = 31
```

Время считывания настроек по умолчанию сервера и очереди важно в этом примере. TORQUE применяет настройки по умолчанию сервера и очереди по-разному в «*job centric»* и «*queue» centric* методах. В «job centric» методе, TORQUE не применяет настройки по умолчанию сервера и очереди до тех пор, пока задача не попадет в исполняющую (последнюю) очередь. В «queue centric» методе, он сначала считывает настройки по умолчанию сервера, прежде чем поставить задачу в очередь-маршрутизатор. В любом из этих методов настройки по умолчанию очереди аннулируют настройки по умолчанию сервера. По умолчанию, TORQUE работает по «job centric» методу. Для того,

чтобы активировать «queue centric» метод, надо установить queue_centric_limits, как в примере:

```
set server queue_centric_limits = true
```

Особенностью «job centric» метода является то, что если в задаче не выставлено значение параметра, настройки по умолчанию сервера и очереди-маршрутизатора не применяются в тот момент, когда проверяются ограничения для очереди по ресурсам. Следовательно, задача, которой требуется 32 узла, не будет проверена на соответствие min_resource.ncpus limit. Также в вышеприведенном примере задача без каких-либо атрибутов будет помещена в очередь reg_64, так как значение since ncpus по умолчанию с сервера будет применено только после того, как задача будет поставлена в выполняющую очередь.

Замечание: Если появляется ошибка ‘“qsub: Job rejected by all possible destinations”’ после постановки задачи в очередь, то возможно следует добавить queue location information, (например, в атрибуте route_destinations очереди-администратора изменить ‘“batch”’ на ‘“batch@localhost”‘)

Параметры сервера

В TORQUE параметры сервера указываются, используя команду qmgr. Более точно, внутри команды qmgr субкоманда **set** должна быть использована для изменения параметров сервера, как в примере:

qmgr

```
> qmgr -c 'set server default_queue=batch'
```

Параметр	**Формат**	**Значение по умолчанию**	**Описание**
acl_hosts	[+]hostname.domain[,...]	---	List of hosts which may request services from this server. This list contains the network name of the hosts. Local requests, i.e. from the server host itself, are aways accepted even if the host is not included in the list. See section 10.1, Authorization, in the PBS External Reference Specification. Requires full manager privilege to set or alter.
acl_host_enable	<BOOLEAN>	FALSE	Attribute which when true directs the server to use the acl_hosts access control lists. Requires full manager privilege to set or alter.
acl_logic_or	<BOOLEAN>	FALSE	specifies if user and group queue ACL‘s should be logically AND’d or logically OR’d
Allow_node_submit	<BOOLEAN>	FALSE	specifies if users can submit jobs directly from any trusted compute host directly or from within batch jobs (See Configuring Job Submit Hosts)

default_queue	<STRING>	---	indicates the queue to assign to a job if no queue is explicitly specified by the submitter
job_stat_rate	<INT>	45 (set to 30 in TORQUE 1.2.0p5 and earlier)	specifies the maximum age of mom level job data which is allowed when servicing a qstat request. If data is older than this value, the pbs_server daemon will contact mom's with stale data to request an update. **Note**: For large systems, this value should be increased to 5 minutes or higher.
job_nanny	<BOOLEAN>	FALSE	Enables the experimental "job deletion nanny" feature. All job cancels will create a repeating task that will resend KILL signals if the initial job cancel failed. Further job cancels will be rejected with the message "job cancel in progress." This is useful for temporary failures with a job's execution node during a job delete request.
log_level	<INT>		specifies the pbs_server logging verbosity. Maximum value is 7.
log_file_max_size	<INT>		specifies a soft limit, in kilobytes, for the server's log file. The filesize is checked every 5 minutes, and if the filesize is greater than or equal to this value then it will be rolled from X to X.1 and a new empty log will be opened. Any value ⇐ 0 will be ignored by pbs_server (the log will not be rolled)
log_file_roll_depth	<INT>	1	This parameter controlls how deep log files will be rolled, if log_file_max_size is set, before they are deleted.
mail_domain	<STRING>		Override the default domain for outgoing mail messages. If set, emails will be addressed to "euser@mail_domain". If unset, the job's Job_Owner attribute will be used.
mom_job_sync	<BOOLEAN>		Enables the experimental "job sync on MOM" feature. When MOMs send a status update, and it includes a list of jobs, server will issue job deletes for any jobs that don't actually exist. This is an EXPERIMENTAL feature and may be removed in the future.
node_check_rate	<INT>	600	specifies the minimum duration (in seconds) that a node can be unresponsive to server queries before being marked down by the pbs_server daemon
node_pack	<BOOLEAN>		Controls how multiple processor nodes are allocated to jobs. If this attribute is set to true, jobs will be assigned to the multiple processor nodes with the fewest free processors. This packs jobs into the fewest possible nodes leaving multiple processor nodes free for jobs which need many processors on a node. If set to false, jobs will be scattered across nodes reducing conflicts over memory between jobs. If unset, the jobs are

			packed on nodes in the order that the nodes are declared to the server (in the nodes file). Default value: unset - assigned to nodes as nodes in order that were declared.
node_ping_rate	<INT>	300	specifies the maximum interval (in seconds) between successive *pings* sent from the **pbs_server** daemon to the **pbs_mom** daemon to determine node/daemon health.
poll_jobs	<BOOLEAN>	TRUE (FALSE in TORQUE 1.2.0p5 and earlier)	if set to **TRUE**, pbs_server will poll job info from mom's over time and will not block on handling requests which require this job information. If not set, no polling will occur and if job information is requested which is stale, pbs_server may block while it attempts to update this information. **Note**: For large systems, this value should be set to **TRUE**.
query_other_jobs	<BOOLEAN>	FALSE	specifies whether or not non-admin users may view jobs they do not own
resources_available	<STRING>	N/A	allows overriding of detected resource quantity limits (see queue resources_available) **Note:** pbs_server must be restarted for changes to take effect. Also, resources_available is constrained by the smallest of queue.resources_available and the server.resources_available.
submit_hosts	"<HOSTNAME>[,<HOSTNAME>]..."	---	specifies the list of hosts beyonds the host running pbs_server which can submit batch or interactive jobs. See Configuring Job Submit Hosts)
tcp_timeout	<INT>	8	specifies the maximum amount of time any pbs process will wait for data on a tcp connection to another pbs process

Эти параметры устанавливаются, используя команду **qmgr**. Например,

```
> qmgr -c set server tcp_timeout=8
```

Network File System

При запуске задачи, ее stdin файл (если он указан) копируется на удаленный узел. Этот файл помещается в директорию “$PBSMOMHOME” на «mother superior» узле (например, “/usr/spool/PBS/spool”). Во время выполнения задачи создаются stdout и stderr файлы, и они также помещаются в эту директорию, под именами **$JOBID.OU** и **$JOBID.ER**.

Когда задача выполнена, MOM копирует эти файлы в директорию, откуда задача была запущена. По умолчанию, это копирование файлов будет выполнено с испольованием **rcp** или **scp**.

Если доступна общая файловая систем, например NFS, DFS, или AFS, то вычислительная система может быть настроена так, чтобы MOM испольовал это, указывая “$usecp” директиву внутри MOM “config” файла (расположенного в директории “$PBSMOMHOME/mom_priv”), используя следующий формат записи **$usecp <HOST>:<SRCDIR> <DSTDIR>**

Здесь “HOST” может быть заменен на символ ’*’. Пример использования директивы:

mom_priv/config

```
        # /home is NFS mounted on all hosts
        $usecp *:/home  /home
        # submission hosts in domain fte.com should map '/data' directory on
submit host to
        # '/usr/local/data' on compute host
        $usecp *.fte.com:/data  /usr/local/data
```

Если по какой-то причине демон MOM не может скопировать output или error файлы в директорию, из которой запущена задача, то тогда эти файлы копируются в директорию “undelivered”, также расположенную в “$PBSMOMHOME”.

Поддержка MPI (Message Passing Interface)

Библиотека MPI используется для увеличения обмена между частями задач, выполняемыми на различных узлах кластера. TORQUE может работать с любой из MPI библиотек, а некоторые из них могут быть в него интегрированы.

MPICH

При использовании этой библиотеки, следует пользоваться mpiexec для запуска MPI-приложений. Поддержка **mpiexec** интегрирована в TORQUE.

mpiexec это замена скрипта **mpirun**, являющегося частью пакета **mpich,** который используется для того, чтобы запустить параллельную задачу. Причины использования mpiexec вместо mpirun:

- Запуск задачи с помощью TM интерфейса присходит намного быстрее, чем используя rsh для каждого из процессов.
- Ресурсы, используемые запущенными процессами, корректно подсчитываются с помощью mpiexec и записываются в PBS лог-файлы, так как все процессы параллельной задачи находятся под контролем PBS, в отличие от случая использования скрипта mpirun.
- Задачи, которые превысили заказанный предел CPU time, wallclock time, memory usage, или disk space полностью уничтожаются PBS. Для процесса достаточно сложно избежать контроля менеджера ресурсов при использовании mpiexec.
- Можно также использовать mpiexec для усиления безопасности. Если все задачи запускаются с использованием mpiexec, то отпадает необходимость в предоставлении rsh или ssh доступа к вычислительным узлам кластера.

Мониторинг ресурсов

Основной задачей любой системы управления заданиями является мониторинг состояния, работоспособности, конфигурации и использования ресурсов. TORQUE сообщает об атрибутах вычислительных узлов, разбивая их на три основные категории: конфигурация, использование и состояние.

Конфигурация

Включает в себя и hardware конфигурацию, и установленные атрибуты очереди.

Параметр	Описание	Подробности
Architecture (arch)	operating system of the node	the value reported is a derivative of the operating system installed
Node Features (properties)	arbitrary string attributes associated with the node	no node features are specified by default. If required, they are set using the “nodes” file located in the $TORQUEHOME/server_priv directory. They may specify any string and are most commonly used to allow users to request certain subsets of nodes when submitting jobs.
Local Disk (size)	configured local disk	by default, local disk space is not monitored. If the mom config size parameter is set, TORQUE will report, in kilobytes, configured disk space within the specified directory.
Memory (physmem)	local memory/RAM	local memory/RAM is monitored and reported in kilobytes.
Processors (ncpus/np)	real/virtual processors	the number of processors detected by TORQUE is reported via the **ncpus** attribute. However, for scheduling purposes, other factors are taken into account. In its default configuration, TORQUE operates in **dedicated** mode with each node possessing a single virtual processor. In dedicated mode, each job task will consume one virtual processor and TORQUE will accept workload on each node until all virtual processors on that node are in use. While the number of virtual processors per node defaults to 1, this may be configured using the “nodes” file located in the $TORQUEHOME/server_priv directory. An alternative to dedicated mode is *timeshared* mode. If TORQUE’s *timeshared* mode is enabled, TORQUE will accept additional workload on each node until the node’s **maxload** limit is reached.
Swap (totmem)	virtual memory/Swap	virtual memory/Swap is monitored and reported in kilobytes.

Использование

Включает в себя информацию о количестве ресурсов, доступных и используемых, на вычислительных узлах, а также о том, кто или что потребляет их.

Параметр	Описание	Подробности

Disk (size)	local disk availability	by default, local disk space is not monitored. If the mom config size parameter is set, TORQUE will report configured and currently available disk space within the specified directory in kilobytes.
Memory (availmem)	real memory/RAM	available real memory/RAM is monitored and reported in kilobytes.
Network (netload)	local network adapter usage	reports total number of bytes transferred in or out by the network adapter
Processor Utilization (loadave)	node's cpu load average	reports the node's 1 minute bsd load average

Состояние

Включает в себя информацию о работоспособности главного узла, administrative status, и general usage status

Параметр	Описание	Подробности
Idle Time (idletime)	time since local keyboard/mouse activity has been detected	time in seconds since local keyboard/mouse activity has been detected
State (state)	monitored/admin node state	a node can be in one or more of the following states: * **busy** - node is full and will not accept additional work * **down** - node is failing to report, or is detecting local failures with node configuration or resources * **free** - node is ready to accept additional work * **offline** - node has been instructed by an admin to no longer accept work * **unknown** - node has not been detected * **job-exclusive** - all available virtual processors are assigned to jobs

Команды клиента

Команда	Описание
momctl	manage/diagnose MOM (node execution) daemon
pbsdsh	launch tasks within a parallel job
pbsnodes	view/modify batch status of compute nodes
qalter	modify queued batch jobs
qdel	delete/cancel batch jobs
qhold	hold batch jobs
qmgr	manage policies and other batch configuration
qrls	release batch job holds
qrun	start a batch job
qsub	submit jobs
qstat	show status of pbs batch jobs
qterm	shutdown pbs server daemon
tracejob	trace job actions and states recorded in TORQUE logs

Команды сервера

Команда	Описание
pbs_iff	Interprocess authentication service
pbs_mom	start MOM (node execution) daemon
pbs_server	start server daemon

Конфигурирование Node Manager (MOM)

Конфигурирование MOM выполняется с использованием файла “mom_priv/config”, расположенного в директории $PBS_HOME на каждом из запущенных узлов.

Параметры

Параметр	Формат	Описание	Пример
arch	<STRING>	specifies the architecture of the local machine. This information is used by the scheduler only.	“arch ia64”
$clienthost	<STRING>	specifies the machine running **pbs_server** (**Note:** This parameter is deprecated, use pbsserver)	“$clienthost node01.teracluster.org”
$configversion	<STRING>	specifies the version of the config file data	“$configversion 113”
$cputmult	<FLOAT>	cpu time multiplier. **Note:** if set to 0.0, MOM level cputime enforcement is disabled.	“$cputmult 2.2”
$ideal_load	<FLOAT>	ideal processor load	“$ideal_load 4.0”
$ignwalltime	<BOOLEAN>	ignore walltime (do not enable mom based walltime limit enforcement)	“$ignwalltime true”

$logevent	<STRING>	specifies a bitmap for event types to log	“$logevent 255”
$loglevel	<INTEGER>	specifies the verbosity of logging with higher numbers specifying more verbose logging. Values may range between 0 and 7.	“$loglevel 4”
$log_file_max_size	<INT>	Soft limit for log file size in kilobytes. Checked every 5 minutes. If the log file is found to be greater than or equal to log_file_max_size the current log file will be moved from X to X.1 and a new empty file will be opened.	“$log_file_max_size = 100”
$log_file_roll_depth	<INT>	specifies how many times a log fill will be rolled before it is deleted.	“$log_file_roll_depth = 7”
$max_load	<FLOAT>	maximum processor load	“$max_load 4.0”
$node_check_script	<STRING>	specifies the fully qualified pathname of the health check script to run. (see Health Check for more information)	“$node_check_script /opt/batch_tools/nodecheck.pl”
$node_check_interval	<INTEGER>	specifies the number of MOM intervals between subsequent executions of the specified health check. This value default to 1 indicating the check is run every mom interval. (see Health Check for more information)	“$node_check_interval 5”
opsys	<STRING>	specifies the operating system of the local machine. This information is used by the scheduler only.	“opsys RHEL3”
$pbsclient	<STRING>	specifies machines which the mom daemon will trust to run resource manager commands via momctl. This may include machines where monitors, schedulers, or admins require the use of this command.)	“$pbsclient node01.teracluster.org”
$pbsserver	<STRING>	specifies the machine running **pbs_server** (**Note:** This parameter replaces the deprecated parameter clienthost)	“$pbsserver node01.teracluster.org”
$prologalarm	<INTEGER>	Specifies maximum duration (in seconds) which the mom will wait for the job prolog or job job pilog to complete. This parameter default to 300 seconds (5 minutes)	“$prologalarm 60”
$restricted	<STRING>	Specifies hosts which can be trusted to access mom services as non-root. By default, no hosts are trusted to access mom services as non-root.	“$restricted *.teracluster.org”
size[fs=<FS>]	N/A	Specifies that the available and configured disk space in the <FS> filesystem is to be reported to the pbs_server and scheduler. **Note**: To request disk space on a per job basis, specify the “file” resource as in “‘qsub -l nodes=1,file=1000kb”’**Note**: unlike most mom config options, the **size**	size[fs=/localscratch] the available and configured disk space in the “/localscratch” filesystem will be reported.

		parameter is not preceded by a '**$**' character.	
$status_update_time	<INTEGER>	Specifies the number of seconds between subsequent mom-to-server update reports. Default is 20 seconds	$status_update_time 120 mom will send server update reports every 120 seconds.
$timeout	<INTEGER>	Specifies the number of seconds before mom-to-mom messages will timeout if RPP is disabled. Default is 60 seconds	timeout 120 mom-to-mom communication will allow up to 120 seconds before timing out.
$usecp	<HOST>:<SRC DIR> <DSTDIR>	Specifies which directories should be staged (see TORQUE Data Management)	"$usecp *.fte.com:/data /usr/local/data"
wallmult	<FLOAT>	wall time multiplier. Note: if set to 0.0, MOM level walltime enforcement is disabled.	"$wallmult 2.2"

MAUI CLUSTER SCHEDULER

Maui is an advanced job scheduler for use on clusters and supercomputers. It is a highly optimized and configurable tool capable of supporting a large array of scheduling policies, dynamic priorities, extensive reservations, and fairshare and is acknowledged by many as 'the most advanced scheduler in the world'. It is currently in use at hundreds of leading government, academic, and commercial sites throughout the world. It improves the manageability and efficiency of machines ranging from clusters of a few processors to multi-teraflop supercomputers.

Benefits:

- Optimize resource utilization by an additional 35%
- Focus resources on your organization's priorities
- Manage the complexities of sharing and scheduling shared resources
- Ensure quality of service guarantees
- Enforce usage policies

Features:

Maui extends the capabilities of base resource management systems by adding the following features:

- Extensive job priority policies and configurations
- Multi-resource admin and job advance reservation support
- Metascheduling interface
- QOS support including service targets and resource and function access control
- Extensive fairness throttling policies
- Multi-attribute fairshare
- Configurable node allocation and load balancing policies
- Multiple configurable backfill policies
- Detailed system diagnostics, utilization tracking, and reporting
- Allocation manager support and interface
- Extensive resource utilization tracking and statistics
- Non-intrusive 'Evaluation/Test' mode
- Advanced built-in HPC simulator for analyzing workload, resource, and policy changes

Maui Installation

Building Maui

To install Maui, untar the distribution file, enter the **maui-<VERSION>** directory, then run **configure** and **make** as shown in the example below:

```
> gtar -xzvf maui-3.0.7.tar.gz
> cd maui-3.0.7
> ./configure
> make
```

Installing Maui (Optional)

When you are ready to use Maui in production, you may install it into the install directory you have configured using **make install**

```
> make install
```

Note: Until the *install* step is performed, all Maui executables will be placed in **$MAUIHOMEDIR/bin**. (i.e., maui-3.0.7/bin in the above example)

2.2 Initial Maui Configuration

After you install Maui, there are a few decisions which must be made and some corresponding information which will need to be provided in the Maui configuration file, **maui.cfg**. The **configure** script automatically sets most of these parameters for you. However, this document provides some additional information to allow further initial configuration. If you are satisfied with the values specified in **configure** then you can probably skip this section. The parameters needed for proper initial startup include the following:

- SERVERHOST

This specifies where Maui will run. It allows Maui client commands to locate the Maui server. It must specify the fully qualified hostname of the machine on which Maui will run. (Example: `SERVERHOST cw.psu.edu`)

- SERVERPORT

This specifies the port on which the Maui server will listen for client connections. Unless the default port of 40559 is unacceptable, this parameter need not be set. (Example: `SERVERPORT 50001`)

- ADMIN1

Maui has 3 major levels of admin access. Users which are to be granted full control of all Maui functions should be indicated by setting the **ADMIN1** parameter. The first user in this list is considered the *primary* admin. It is the ID under which Maui should always run. Maui will only run under the primary admin user id and will shut itself down otherwise. In order for Maui to properly interact with both PBS and Loadleveler, it is important that the primary Maui admin also be configured as a resource manager admin within each of those systems. (Example: `ADMIN1 joe charles`)

- RMTYPE[X]

Maui must be told which resource manager(s) to talk to. Maui currently has interfaces to Loadleveler, Wiki, and PBS. To specify a resource manager, typically only the resource manager type needs to be indicated using the keywords LL, WIKI, or PBS (Example: `RMTYPE[0] PBS`). The array index in the parameter name allows more than one resource manager to be specified. In these multiple resource manager situations, additional parameters may need to be specified depending on the resource manager type. Some of the related resource management parameters are listed below. Further information about each is available in the parameters documentation.

Установка TORQUE

Главный узел

Распаковать и собрать дистрибутив на том узле, который будет “TORQUE server”, на котором будет запущен pbs_server daemon, и который будет управлять и контролировать все остальные узлы. Пример (где XXX отвечает за версию дистрибутива, например e.g., “-1.2.0p4”):

```
> tar -xzvf torqueXXX.tar.gz
> cd torqueXXX
> ./configure
> make
> make install
```

- (НЕОБЯЗАТЕЛЬНО) Установить другое значение переменной окружения PATH. По умолчанию исполняемые файлы устанавливаются в are или /usr/local/bin, или /usr/local/sbin

$TORQUE_HOME соответствует тому, где TORQUE хранит свои конфигурационные файлы. По умолчанию:

```
/var/spool/torque      для TORQUE 2.1 и выше
/usr/spool/PBS         для более старых версий
```

НЕ РАБОТАЕТ(по крайней мере, мне не удалось)**:**

TORQUE 2.0p2 and higher includes a standard spec file for building your own rpms. Simply run ‘rpmbuild –ta torqueXXX.tar.gz’ to generate RPM packages. It is also possible to use the checkinstall program to create your own RPM, tgz, or deb package.

Вычислительные узлы

Возможны несколько методов по установке TORQUE на вычислительные узлы. Если имеется RPM-based Linux дистрибутив,то можно самим собрать RPM прямо из исходного архива двумя способами:

- Если допустимы настройки по умолчанию, то просто запустите `rpmbuild -ta torque-xxx.tar.gz`.
- Если необходимы особые параметры, тогда надо распаковать и собрать архив, а затем запустить `make rpms`.

Во всех операционных системах можно воспользоваться встроенной “tpackage” системой, которая просто создает самораспаковывающиеся архивы, которые затем могут быть разосланы на вычислительные узлы и запущены там. Чтобы создать эти архивы, надо после выполнения обычной поцедуры (./configure make make install) запустить `make packages`. Затем скопировать получившиеся архивы на любые другие машины и запустить их с флагом –install.

Возможно также использовать TORQUE сервер, как вычислительный узел, и тогда понадобится установить pbs_mom вместе с pbs_server демоном на него.

Пример:

```
> make packages
Building ./torque-package-clients-linux-i686.sh ...
Building ./torque-package-mom-linux-i686.sh ...
Building ./torque-package-server-linux-i686.sh ...
Building ./torque-package-gui-linux-i686.sh ...
Building ./torque-package-devel-linux-i686.sh ...
Done.

... копирование на вычислительные узлы
> ./torque-package-mom-linux-i686.sh --install
> ./torque-package-clients-linux-i686.sh --install
```

Настройка TORQUE сервера

Настройка pbs_server демона выполняеться, используя команду **qmgr**. На новой системе настройка должна предворяться командой **pbs_server -t create**. После выполнения этого, надо настроить желаемую структуру очередей и подключить планировщик.

Пример настройки с использованием одной очереди:

```
pbs_server -t create
qmgr -c "set server scheduling=true"
qmgr -c "create queue batch queue_type=execution"
qmgr -c "set queue batch started=true"
qmgr -c "set queue batch enabled=true"
qmgr -c "set queue batch resources_default.nodes=1"
qmgr -c "set queue batch resources_default.walltime=3600"
qmgr -c "set server default_queue=batch"
```

Эти команды должны выполняться администратором системы.

В этом примере инициализируется настроечный алгоритм и активируется планировщик (используя ‘“scheduling=true”). Следующий шаг заключается в создании очереди и указании ее типа. В случаеPBS, очередь должна быть объявлена “execution”, чтобы в ней могли выполняться задачи Дополнительные настройки (например, установка параметров очереди “started” и “enabled”) позволяет очереди принимать заявки на выполнение задач и запускать поставленные в очередь задачи.

Следующие две строки необязательны, они устанавливают значения по умолчанию для параметров “node” и “walltime” для поставленных в очередь задач. Эти значения используется при постановке задачи в очередь, если пользователем не указаны другие значения. Последняя строка “default_queue=batch” также необязательна, она указывает на то, что задача будет помещена в очередь “batch”, если для нее не будет конкретно указана другая очередь.

Другой способ настройки TORQUE сервера – запуск команды torque.setup. После установки TORQUE сервера настрйте pbs_server демона, выполнив команду torque.setup <USER> из директории с дистрибутивом, где <USER> это имя пользователя, который будет TORQUE администратором. Этот скрипт настроит простую очередь.

Определение вычислительных узлов

Для того чтобы pbs_server мог взаимодействовать с каждым из mom демонов, необходимо объяснить ему, с какими машинами связываться. Каждый вычислительный узел, который должен стать частью вычислительной системы, должен быть указан в server nodes file. Этот файл расположен в $TORQUE_HOME/server_priv/nodes. В большинстве случаев достаточно указать просто имя узла:

server_priv/nodes

```
node001
node002
node003
node004
```

Если вычислительные узлы являются многопроцессорными, то следует указывать количество процессоров с np={number of processors}. Например:

server_priv/nodes

```
node001 np=2
node002 np=4
...
```

Настройка TORQUE на вычислительных узлах

На каждом вычислительном узле, MOM сервер должен быть настроен так, чтобы доверять **pbs_server** демону. Это делается созданием “$(TORQUECFG)/mom_priv/config” файла и установкой параметра $pbsserver. Или можно это сделать, создав “$(TORQUECFG)/server_name” файл и поместив туда имя сервера.

mom_priv/config

```
$pbsserver      headnode          # note: hostname running pbs_server
$logevent       255               # bitmap of which events to log
```

Этот файл одинаков для всез вычислительных узлов.

Запустите pbs_mom демон на всех узлах.

Завершение настройки

Перезапустите pbs_server:

```
qterm -t quick
pbs_server
```

pbsnodes -a теперь должна показывать, что все узлы находятся в состоянии **free**.

Теперь можно запустить **pbs_sched.**