Job Reservations

Qube!'s reservation system is important for correct management of host resources. This is accomplished with the relationship between reservations and resources.

Each host has a list of resources including system resources such as memory and number of processors. Resources are integer based (meaning '1' and '99' are valid resource counts; 'a' and '4.5' are not) and automatically decrement/increment upon acceptance of a job. A resource can be "discovered" by the system (eg, number of cores) or defined by admins (eg, number of licenses).

The user must know whether the resource is global or host-based (see System-Wide Resource Tracking). Global resources are tracked through the entire system, for example, license tracking. A host resource is local and specific to each host, for example the number of cores.

Resources are defined and reserved through this syntax:

type.name=quantity[,*type.name=quantity*…]

This is set on the Supervisor (for global resources, see supervisor_global_resources), on the host (for host-based resources, see worker_resources) and/or in the job submission UI in the reservations field.

Specifying Quantity

Syntax	Meaning
`host.processors=1+`	Dispatch to a worker with at least 1 open slot, then occupy all currently open slots. The general form is host.processors=*N*+, where *N* is a positive integer.
`host.processors=1*`	Dispatch to a worker with at least 1 open slot *and* no used slots (i.e. *worker must be idle!*), then occupy all currently open slots. The general form is host.processors=N*, where N is a positive integer
`host.processors=all`	Equivalent to `host.processors=1*`
`host.processors=N-M`	Dispatch to a worker with at least N to M open slots. Upon being dispatched to a worker, it occupies as many slots as it can, up to M, as slots become available.

Dynamic Thread Assignment

When a 1+ or similar job picks up on a Worker, we don't know how many slots were available or assigned to that instance. That number is made available dynamically in the running job's environment as an environment variable, `QB_JOBSLOTS=6`, (or whatever the value) and stored in the Qube! database in the job's subjob table as "allocations". One use of this could be that the job references $QB_JOBSLOTS on the command line to specify the number of threads a renderer should use.

Examples

Reservation	Explanation
`"host.memory=200"`	Reserve 200MB of memory on the host
`"host.processors=1+"`	Reserve all processors on a host, but at least 1 must be available in order to start
`"host.processors=1-20"`	Reserve 20 processors on a single host, but at least 1 must be available in order to start
`"global.maya=1"`	Reserve a global resource called maya

See Also

supervisor_global_resources
worker_resources
System-Wide Resource Tracking