FBSNG v1.2 Scheduler.

Scheduler Features

FBSNG Scheduler has the following general features:

1. Mixed scheduling. Even though it is generally true that section, which was submitted earlier, will most likely start earlier, FBSNG also takes into account how many resources the section requires. That is why section that was submitted first but requires more resources may start after another section, which was submitted later, but needs less resources to start. FBSNG takes balanced approach to weigh both criteria to ensure "guaranteed scheduling".
2. Guaranteed scheduling. Regardless of the section resource requirements and those of other sections waiting to be started, or being submitted, the section wait time is guaranteed to be finite. For example, if there is a pending section requiring relatively big portion of resources (big section), and constant inflow of sections with small resource requirements (small sections), the big section should not have to be pending forever. After some time, if properly configured, the Scheduler will block small sections from starting to free up resources required by big section.
3. Fair share scheduling. The Scheduler can be configured in such a way that it will control relative frequency with which sections of different projects start on the farm. Assuming that in average over long period of time, sections of the same project consume more or less the same amount of resources and have more or less the same time duration, statistically, the starting frequency is directly proportional to the share of resources consumed by different projects.

All these features are achieved using scheduling algorithms based on the idea of dynamic priorities. The scheduling algorithms are designed to provide desired long-term balance between (sometimes contradicting) requirements of guaranteed, fair-share scheduling and efficient farm resources utilization.

Generally, section priority reflects the number of "smaller" sections started ahead of this section. The higher the section priority, the more chances it has to start comparing to sections with lower priorities. Queue priorities are used in similar way. Relative queue priority is growing as long as for some reason, the Scheduler starts sections from other queues, but not from this one. Queue priority goes down when sections do start from it.

FBSNG scheduling is not pre-emptive. That is why some other mechanism must be used to ensure guaranteed scheduling. This is what priority gaps are used for. They determine the moment when higher priority object (queue or section) is considered to be "stuck" due to resources being consumed by objects with lower priorities, and in order to "let it go" (start any section from the queue or start the section which has been waiting for too long time) it is necessary to "hold" other objects for a while, even though it may mean that for some time a portion of farm resources will remain unused.

FBSNG Administrator can control the behavior of the Scheduler by adjusting various parameters used to determine how priorities change over time.

Scheduling Parameters

FBSNG configuration includes the following parts:

- Farm nodes configuration defines what resources farm nodes provide.
- Process types configuration, along with user job description file (JDF), define what resources user processes consume.
- Queues configuration defines what queues are available and scheduling their parameters.

The following queue parameters are used to control and customize FBSNG Scheduling behavior. They determine how queue and section priorities change over time.

- Maximal section priority (SPmax)
- Section priority gap (SPG)
- Minimal queue priority (QPmin)
- Maximal queue priority (QPmax)
- Queue priority increment (QPI)
- Queue priority decrement (QPD)
- Queue priority gap (QPG)

The Scheduler currently does not use other queue parameters such as CPU and real time limits to choose next section to start.

See FBSNG Administration and Installation Guide for more details on configuration file format and FBSNG configuration procedures.

General Scheduling Algorithm

FBSNG Scheduler runs periodically, but not more often than once per 5 seconds. The following events may trigger the Scheduler to run.

- New job is submitted
- User process finishes and de-allocates some resources
- User section finishes and de-allocates some resources
- User releases a section which was held
- New farm node becomes available
- FBSNG BMGR process finishes initial recovery process
- Certain time passes since last time scheduler ran

While running, Scheduler repeats the following two steps until it cannot start any more sections. Both steps are described in more details below.

1. Inter-queue scheduling. During this step, Scheduler determines the list of queues to be considered for scheduling at this time. Scheduler takes into account queue priorities and whether each queue has at least one section ready to start. This list will be referred to as "active queue list".
2. Internal queue scheduling. Scheduler goes through resulting ordered list of queues until it finds a section to start in one of queues, and starts it successfully. This step is described in more details below.
3. If a section was started from some (*selected*) queue in step 2, Scheduler re-calculates queue priorities and returns to step 1. Otherwise it stops. New queue priorities are calculated in the following way:
    - Priority of the selected queue is decreased by its QPD multiplied by a quantity proportional to the product of the section size and estimated section life time;

- If the priority of the selected queue fell below its QPmin, priorities of all queues on the active list, including the selected queue, are incremented by the same amount to return the active queue priority to its QPmin.
- Priorities of all other queues on the active list are increased by their QPI values;
- Finally, if priority of any queue becomes higher than its QPmax, it is lowered to QPmax.

QPI and QPD parameters determine how relative queues priorities change every time the Scheduler starts new section. Normally, QPI parameter is set to 0, and QPD parameter controls relative share of resources similar processes started from each queue will utilize in average over time. For similar processes started from different, this relative average share is reversly proportional to QPD parameter:

Share ~ 1/QPD

Inter-Queue Scheduling

1. Scheduler makes the list of those queues that are not held and have at least one section in state "ready" (section is not held, and its dependencies are satisfied, if any), and the farm has enough resources to satisfy the section's resources requirements;
2. The list of queues is sorted by current queue priority (QP) and then by the submission time of the oldest ready section in the queue;
3. Scheduler calculates *queue priority threshold* (QPT) by subtracting QPG of the first queue on the list from its priority (QP). Scheduler removes from the list all queues with priorities lower than QPT;

As you can see, queue priority gap (QPG) is used to temporarily hold low priority queues until a section from the queue with highest priority can start.

Internal Queue Scheduling

The following algorithm is used to determine which section in the queue to start and how to update section priorities:

1. Scheduler forms list of sections in "ready" state, then orders it by priority, and then by submission time;
2. Scheduler calculates *section priority threshold* (SPT) by subtracting SPG from priority (SP) of the first section on the list;
3. Scheduler removes from the list all sections with priorities lower than SPT;
4. For each section in this ordered and truncated list of ready sections, Scheduler determines if there are sufficient resources currently available to start the section. If this is the case, Scheduler starts the section and stops internal scheduling for this queue and proceeds to the next step of the algorithm. Otherwise, it goes to the next section on the list.
5. If a section was started in step 4, Scheduler updates priorities of sections of the queue by increasing priorities of all sections with priority higher than the priority of the started section. Section priority always remains limited by SPmax.

As you can see, generally, section priority never goes down. Section priority remains unchanged until it reaches "ready" state. The section priority goes up as other sections submitted later start "passing" the section because they require fewer resources to start. If a section priority is much higher than priorities of some other sections, lower priority sections are blocked until the high priority section starts.

Scheduler Configuration Guidelines

It is recommended that FBSNG Administrator and users while configuring and using FBSNG Scheduler follow certain guidelines.

Long-term projects should be assigned one or more dedicated queues. The number of queues should be determined based on the following considerations:

- Separate queue should be created for every activity within the project that uses sections with similar resource requirements, resource consumption patterns and time duration;
- Separate queue or group of queues should be dedicated to an activity or repeating task within the project if it is desired to control resource distribution among the activities or tasks. Since FBSNG does not maintain accounting information on per-queue basis, if such information is desired, each task or project should be assigned separate process type.

For short-term projects, or project for which it is impossible or impractical to identify common tasks or activities with similar resource requirements, generic queues should be used. In this case this should be user's responsibility to choose appropriate queue for every section.

Common Configuration Problems and Solutions

The following are suggested solutions for common configuration problems. They illustrate use of various configuration parameters. These solutions can be used as a guideline for initial Scheduler configuration in various simple cases. However, in order to achieve the desired functionality, the administrator may need to re-adjust Scheduler parameters based on resource utilization statistics.

Occasional High Priority Tasks

Goal: The farm is used for two projects. Most of the time only background project is active, and occupies all resources of the farm. Occasionally, jobs of the other project are submitted. It is desired to set up the farm in such a way that jobs of high priority project start as soon as possible, and as long as there are some pending jobs of the high priority project, background project jobs should not be started.

Solution: Two queues should be set up for two projects. High priority project queue should be assigned higher minimal priority (QPmin) than background project queue. In case when background project jobs take fewer resources than jobs of the other process, in addition, QPG for high priority queue should be set slightly higher or equal to the difference between QPmin parameters. QPI of high priority queue may be used to further reduce initial start delay for jobs in the queue. In this case, QPI should be comparable to the difference in QPmin.

Example:

Parameter	High priority queue	Background queue
QPmin	5	0
QPG	7	100

Stationary Utilization Shares

Goal: There are two projects. Both projects use sections with similar resource requirements, potentially with different, even variable number of processes, and with different life times. Both project run constantly without long interruptions. It is desired that in average, at any time, 70% of resources are allocated to first project (project A), and 30% - to second project (project B).

Solution: Set a queue for each project (A and B). Set QPD values according to the following formula:

$$QPD(A)/QPD(B) = share(B)/share(A)$$

QPG and QPmax for both queues must be much much greater than QPD.

Example:

Parameter	Queue A	Queue B
QPmin	0	0
QPmax	100	100
QPG	100	100
QPD	3	7

Bulk and Light Queues

Goal: There are two projects, *bulk* and *light*. Sections of bulk project require considerably big number of resources, or use sections with bigger number of processes comparing to light project. Naturally, light project has advantage over bulk simply because light sections require fewer resources, and therefore have more opportunities to start. It is required however that bulk sections start as often as light sections even though it is impossible without having some portion of farm resources to remain "reserved" and unused for some time intervals.

Solution: Set up two queues, bulk and light. Use queue priority gap (QPG) to hold light queue until a section starts from bulk queue.

Example:

Parameter	Bulk queue	Light queue
QPmin	0	0
QPmax	1000	1000
QPG	10	100
QPD	1	1

Bulk and Light Sections in the Same Queue

Goal: The goal is the same as in the previous case, but for some reason, single queue has to be used for both bulk and light sections. It is required that bulk sections do not have to wait infinitely to start in presence of pending light sections. The goal is to start in average one bulk section per N light sections.

Solution: Use Section Priority Gap (SPG) to use employ guaranteed scheduling feature of the Scheduler. SPG for the queue should be set to N.

Hard Farm Partitioning

Goal: It is required to allocate fixed sets of worker nodes to some projects. It is acceptable that while certain project is idle, nodes assigned to it will remain unused.

Solution: Use node attributes to assign nodes to projects. Each node should be assigned one or more attributes corresponding to projects allowed to run there. Each project should be associated with its own process type. Attribute requirements should be entered into default resource requirements for process types. Note that if certain projects are allowed to share certain nodes, those nodes should be assigned more that one attribute.

See also "FBSNG Resources" document for more information.

Soft Farm Partitioning

Goal: It is required that certain projects are limited to certain number of resources used in the same time. It is not required to restrict project to use certain nodes. It is acceptable that while one or more projects are idle, some portion of farm resources remains unused.

Solution: In this case, individual process type should be assigned to each project. Resource allocation quotas should be defined to limit resource allocation for each process type. To avoid resource underutilization, project quotas can be defined so that farm resources are slightly over-booked.

See also "FBSNG Resources" document for more information on FBSNG resources.