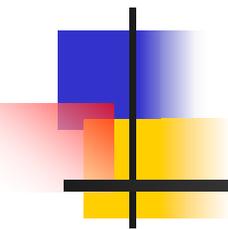
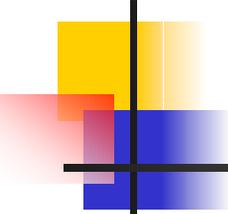


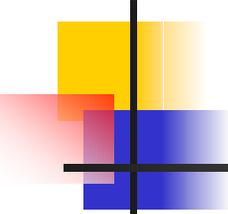
FBSNG Scheduler for Users





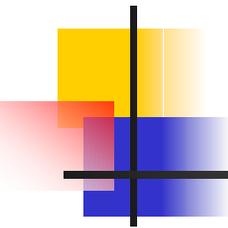
Major Features

- Multi-process jobs
 - FBSNG job section, an array of 1 or more processes, is the unit of Scheduler's operation
- Quotas and Shares
 - When there is no competition, resource utilization is controlled by quotas
 - Competition between queues is controlled by shares set by administrator
- Guaranteed scheduling
 - Regardless of resource requirements and section size, it will start within finite time. Time depends on:
 - Competition
 - Size of the section
 - Share of the queue



Controlled Shares (a.k.a. Fair-Share Scheduling)

- Instead of (A):
 - Keep track of how much resources projects consumed *recently* and try to compensate under- or over-utilization
- FBSNG does this (B):
 - Start jobs of each project in certain proportions defined by the administrator. System will come to statistical equilibrium according to target shares.
- Why not A:
 - No “natural time scale” (what is *recently*?): jobs can run for 10 minutes, 1 hour, 2 days
 - Just because project A was idle last week, it is not fair to give them whole farm this week
- Why B:
 - Simplicity: no hard-coded magic time constants
 - Naturally comes to new equilibrium when conditions change



Dynamic Priorities

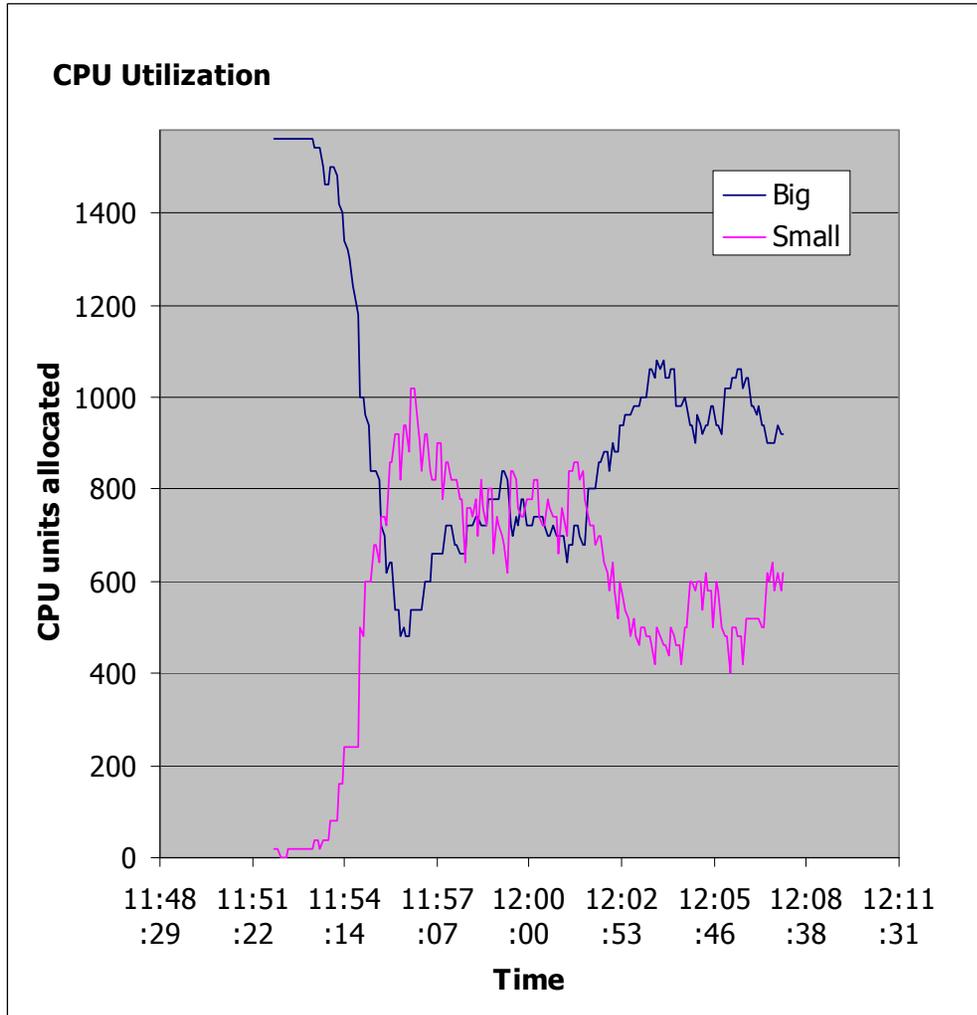
- Queue has “priority”, which is dynamic quantity
- Scheduler picks the queue with highest priority first

```
fnpcb> fbs queues
```

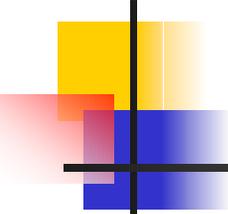
Name	State	Def Proc	Type	Prio	NPend	NRun	NTotal
A	OK	A		0	0	2	2
B	OK	B		25	3	14	17
C	OK	C		1032	10	1	11
IOQ	OK	IO		500	0	0	0

- Relative queue priority
 - decreases as sections from *this* queue start
 - increases as sections from *other* queues start
- Relative priority change determines relative queue share:
 - The bigger my priority loss per job is
 - The less often my jobs will start
 - The lower my share will be

Controlled Shares: Simulation

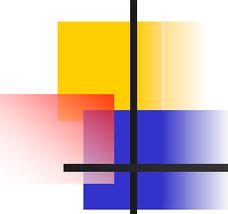


- Farm "capacity"
 - 1560 CPU units
- Big
 - 3 processes/section
 - 3-5 minutes
- Small
 - 1 process/section
 - 40-80 seconds
- Initially
 - Equal shares
- Then
 - 2:1



Big jobs vs. small jobs

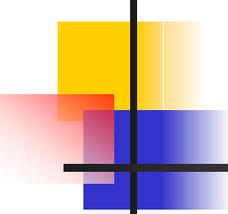
- As described, fair share schema works well when competing jobs are of equal size (resource requirements)
- What happens if my jobs are significantly bigger than my competitor's ?
- Big jobs are at disadvantage just because they require bigger portion of the farm to be free to start
- Example:
 - Big queue – queue with big jobs (10 processes)
 - Small queue – small jobs (1 process)
- Unless all processes of the same big job finish exactly at the same time, there will never be a room for a big job



Controlled Shares & Guaranteed Scheduling

- How does FBSNG solve this problem ?
- As Small jobs keep starting, Big queue priority will go up... infinitely ? No.
- When Big queue priority reaches its QPGap (set by administrator), Scheduler will block Small queue.
- It starts draining the farm. Running jobs finish, but new jobs do not start.
- Eventually, enough resources are available for big job to start.
- Scheduler starts big job and decrements priority of the big queue.
- Small queue is free to go again for a while...

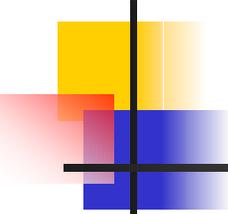
- In this scenario, queue priority difference will fluctuate around the QPGap of the big queue.
- Shares will come eventually to the target equilibrium



Why my job does not start ?

Possible reasons:

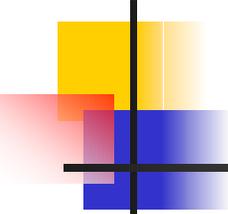
- Your Process Type is near its quota
 - Use "fbs ptypes -qu" to see current utilization/quota
- Not enough resources available
 - Use "fbs resources" to get resource utilization status
 - Use "fbs nodes" to see if any nodes are down
- Your queue is blocked by higher priority queue
 - Use "fbs queues" to see queue priorities



How to make my jobs start sooner ?

Suggestions, not requirements:

- Make your jobs smaller, but not necessarily microscopic
 - Farm size is 180, job of 20 processes is $\sim 10\%$ - OK
 - This will help in short, but *not* long run
- Ask only what you really need, avoid “exotic” combinations:
 - I want 40 CPUs on exactly 20 nodes
 - On the farm of 180 CPUs, this will be available when ~ 84 CPUs are free ($\sim 47\%$), much more than you actually need
- Keep your queues full. Every time your queue is empty, you lose some priority
- If possible, make your jobs shorter in time
 - Typical draining time \sim Job lifetime / Number of running jobs



We need your feedback

- Different groups have different requirements
 - Some need fast start and fast finish
 - Some need steady long-term run, but do not care about short-term fluctuations
 - Some run “production lines”
 - Some run, analyze results and then run again
- FBSNG Scheduler is highly flexible and can meet many (not all, but many) different requirements. It is important to hear from users what should be improved.
- Do not hesitate to complain !