What are the metrics or considerations one should take into account when building a CPU scheduler?

Response time - interactive jobs

Fairness - same amount of cpu < user / process / threads - org

Priority

Low overheads < low # migrations / # context switches

Utilization high -

Simplicity - easy to modify, understand, prove properties?
- easy to debug reproduce

No starvation

Avoid contention < load balance across cores / global planning

Non-critical, bg - low priority → deadlines - realtime

power consumption

cache consideration - cache affinity

hw aware - heterogeneity

scalable < multiple cores 64? - load balancing / - single queue
processes

Simplicity, Manageability, Easy to test/understand
Hard to game. prove prop.
High throughput for cpu-bound jobs

Fast turnaround/response time for interactive jobs

High utilization (→ work conserving)

Few context switches

Meet deadlines - real time

Proportional share - across users, groups, processes

No priority inversion -
Energy conservation
Cache re-use / sharing
Heterogeneity
Locks - no spinwaiting on same core.

Run @ same time as cooperating threads. signal → wait

Low lock contention for shared queues - #cores

Scalable w/ # threads.

Balanced load across cores

~~Work~~ Compose w/ other resources { memory, disk

Utilization - Linux ~~Not~~ Work Conserving

no idle resources, if ~~waiting job~~

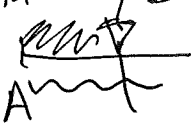
Good to be Non Work conserving?

resources idle?

1) power consumption - batch work

2) time critical deadline - no preemption

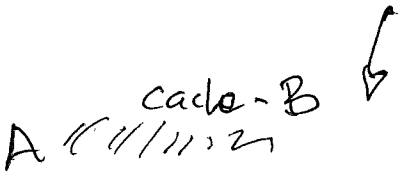
idle B - more important!



preemption costly

- parallel systems

3) cache affinity - state



Disks! anticipatory scheduling