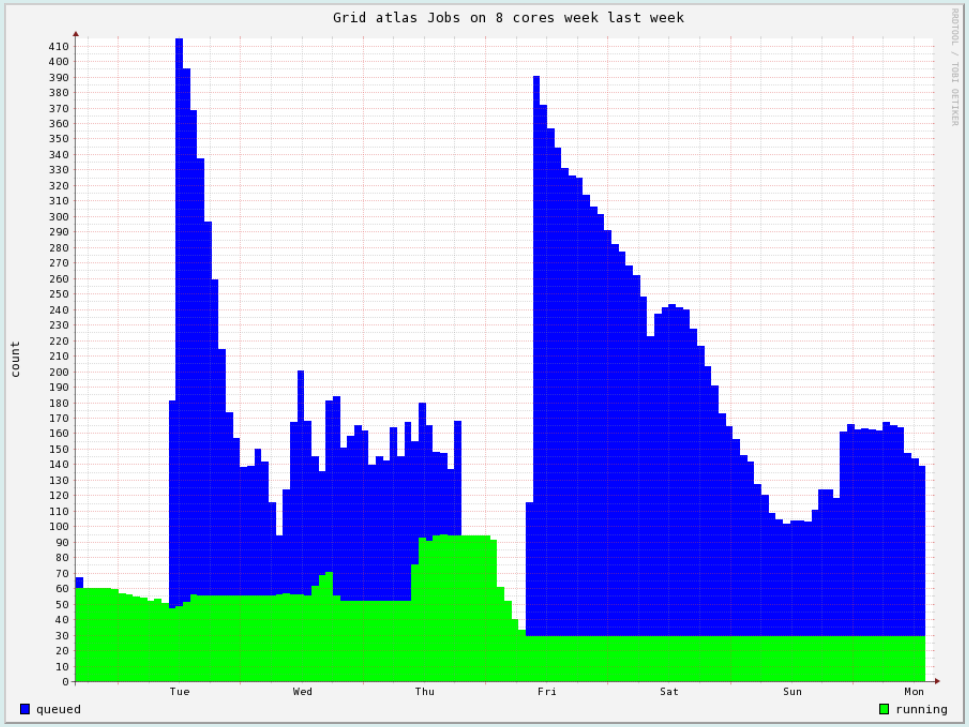
KIT: MCore Status

Week #5

- observed waves of ATLAS multi job slot jobs
- probably related to more general issues while job submission
- manually freed nodes on Thursday for vain

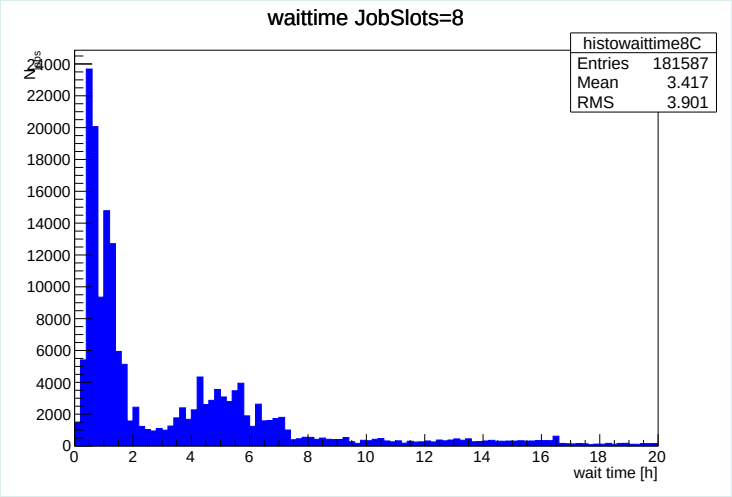
ATLAS MCOREs@KIT 2014.01.27-2014.02.03



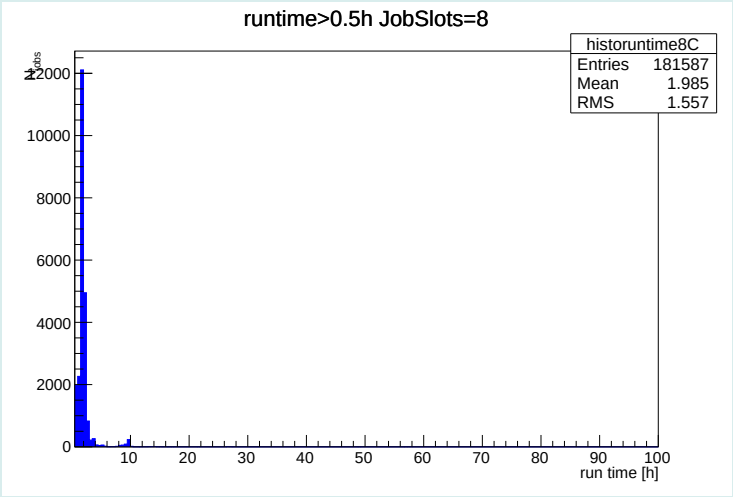
KIT: MCore Statistics 2014.Jan

Karlsruhe Institute of Technology

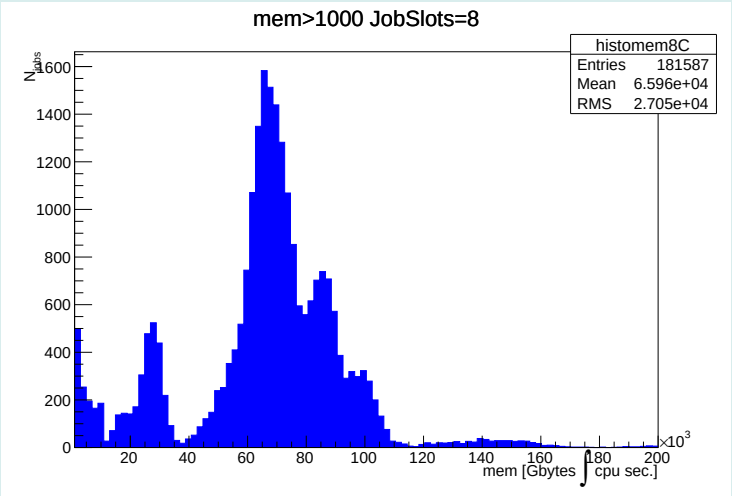
wait time



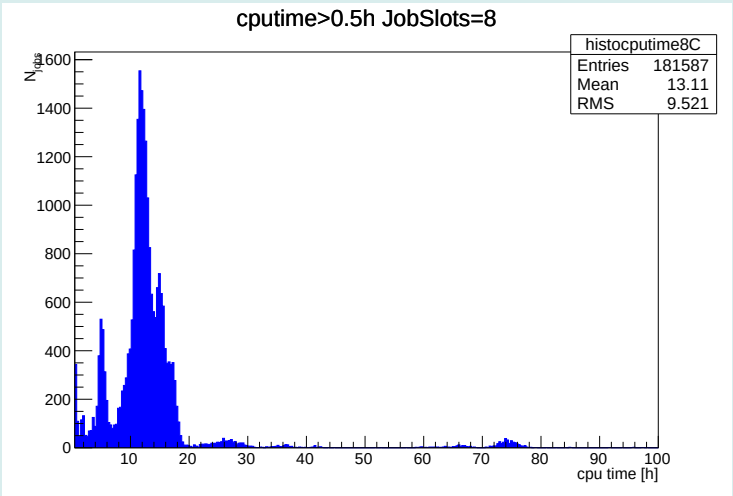
wall time (queue length=120h)



Memory usage

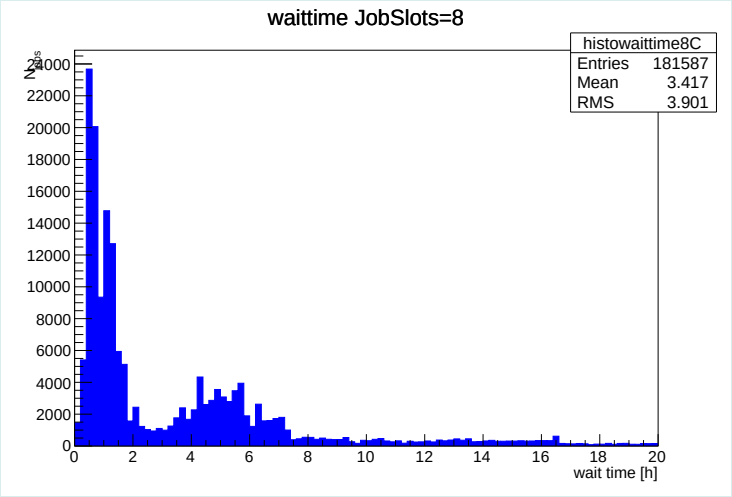


CPU time (∼ ×#Cores)

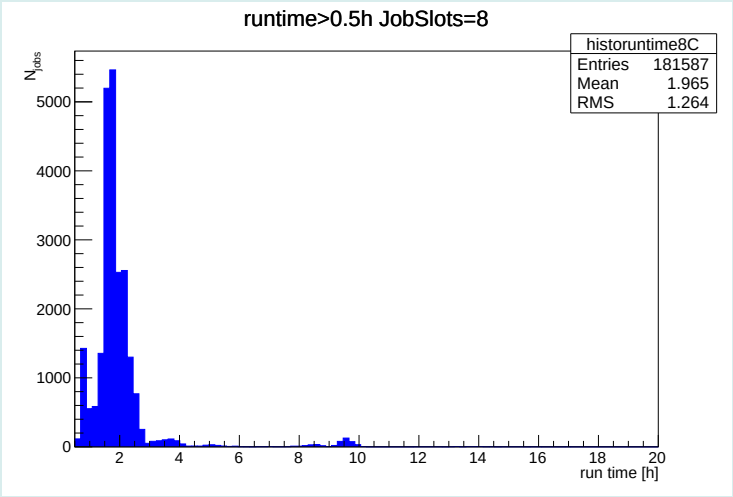


KIT: MCore Statistics 2014.Jan

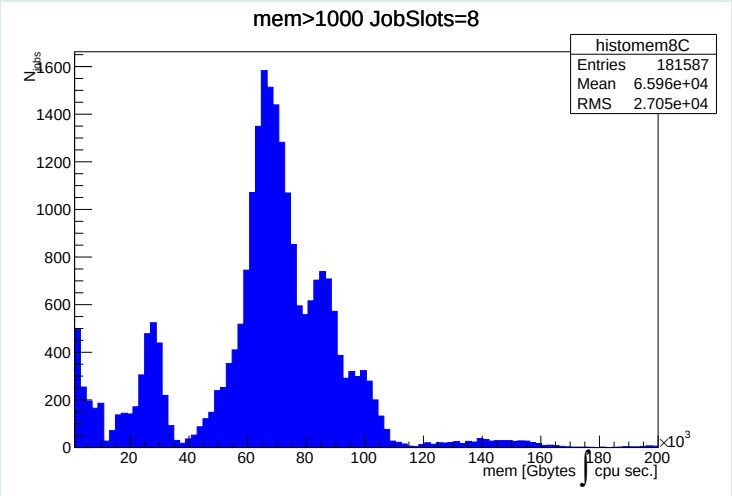
wait time



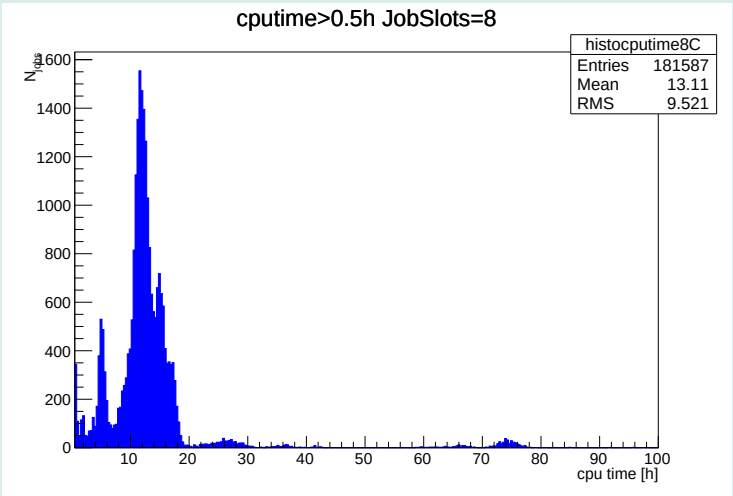
wall time (queue length=120h)(zoom 20h)



Memory usage

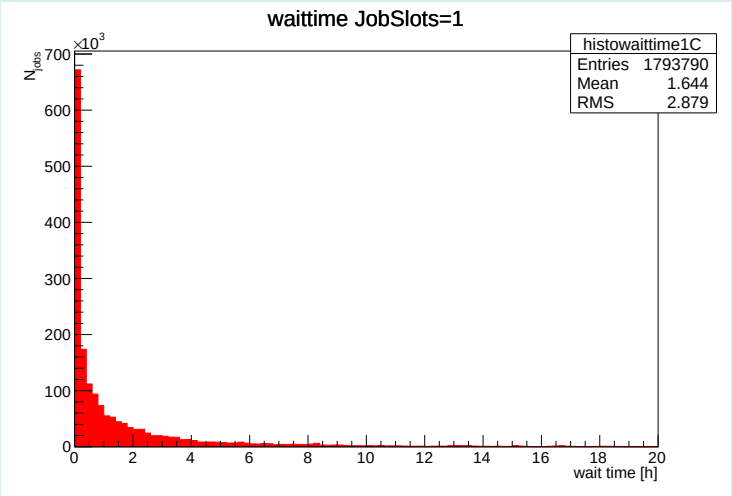


CPU time (∼ ×#Cores)

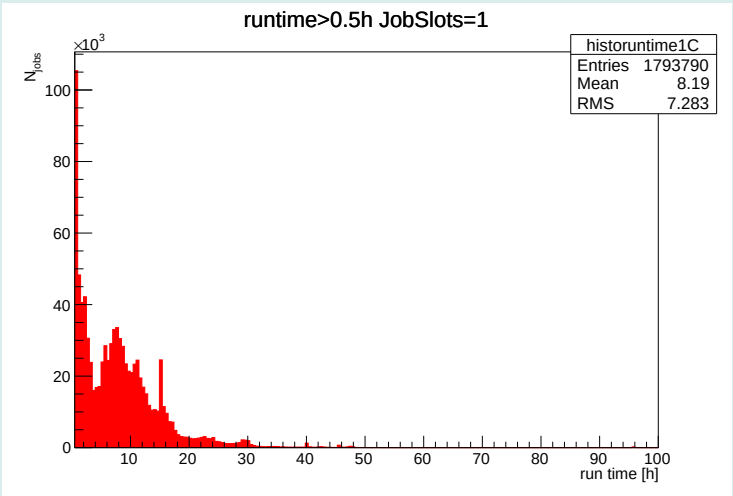


KIT: SingleCore Statistics 2014.Jan

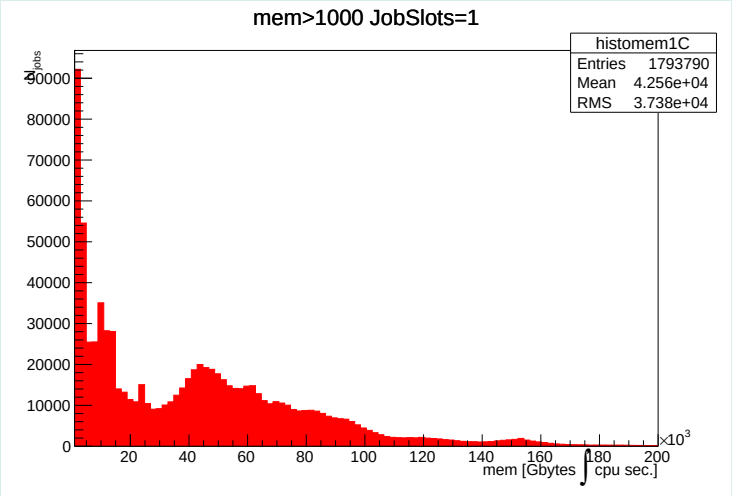
wait time



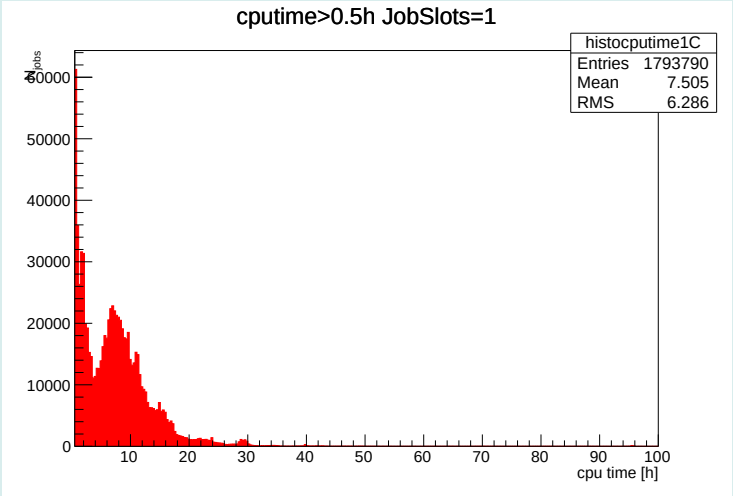
wall time (queue length=120h)



Memory usage

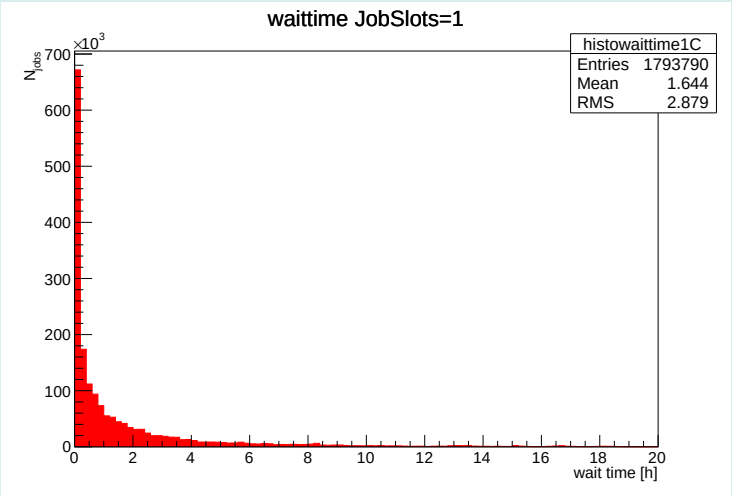


CPU time

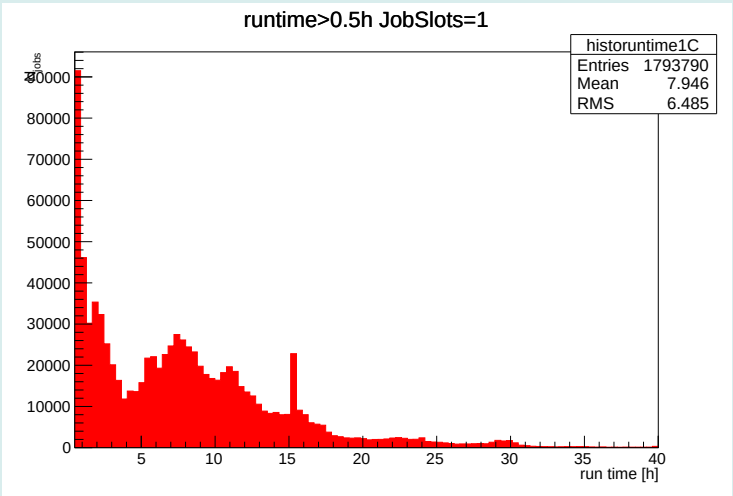


KIT: SingleCore Statistics 2014.Jan

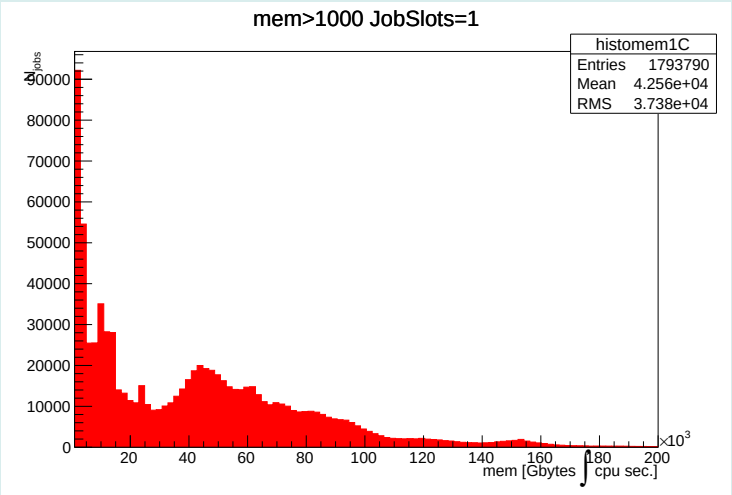
wait time



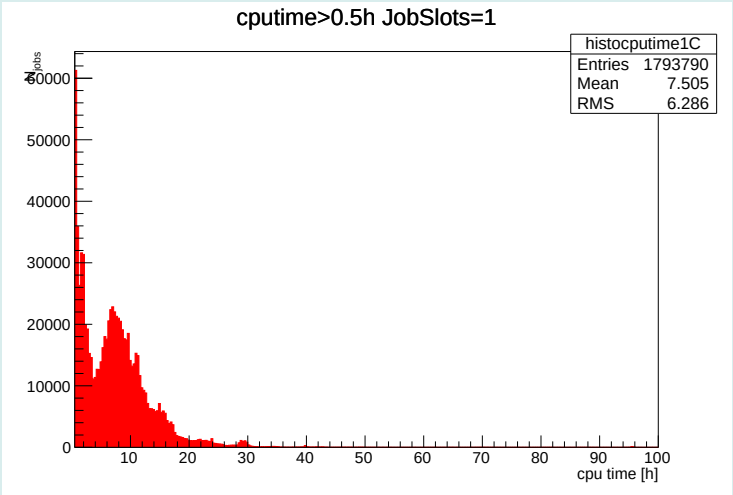
wall time (queue length=120h)(zoom 40h)



Memory usage



CPU time



HPC Experiences@KIT

HPC vs. WNs

- exchanged experiences with HPC team
  - $\rightsquigarrow$ cluster utilization and job wait times
- HPC cluster #1 (2848 cores)
  - for $\text{jobs}_{513..1024\ \text{cores}} \sim \text{cluster size}_{18\%..36\%}$
  - wait time: 10h..150h
  - cluster utilization: 80%..95%
- HPC cluster #2 (6560 cores)
  - for $\text{jobs}_{1025..2048\ \text{cores}} \sim \text{cluster size}_{15\%..31\%}$
  - wait time: 60h..280h
  - cluster utilization: 77%..89%

$\sim$ compare with GridKa WorkerNodes

- $\text{mcore}_{8\ \text{cores}} = 33\%$ on $\text{node}_{24\ \text{Cores}}$
- GridKa WN utilisation: 2013: 91.8%, since 2013.Nov 94.5%

Addendum: Questions/Discussion

Addendum: Questions/Discussion

some illustrations

(got the impression that sometimes discussions were going talking at cross purposes)

Addendum: Questions/Discussion

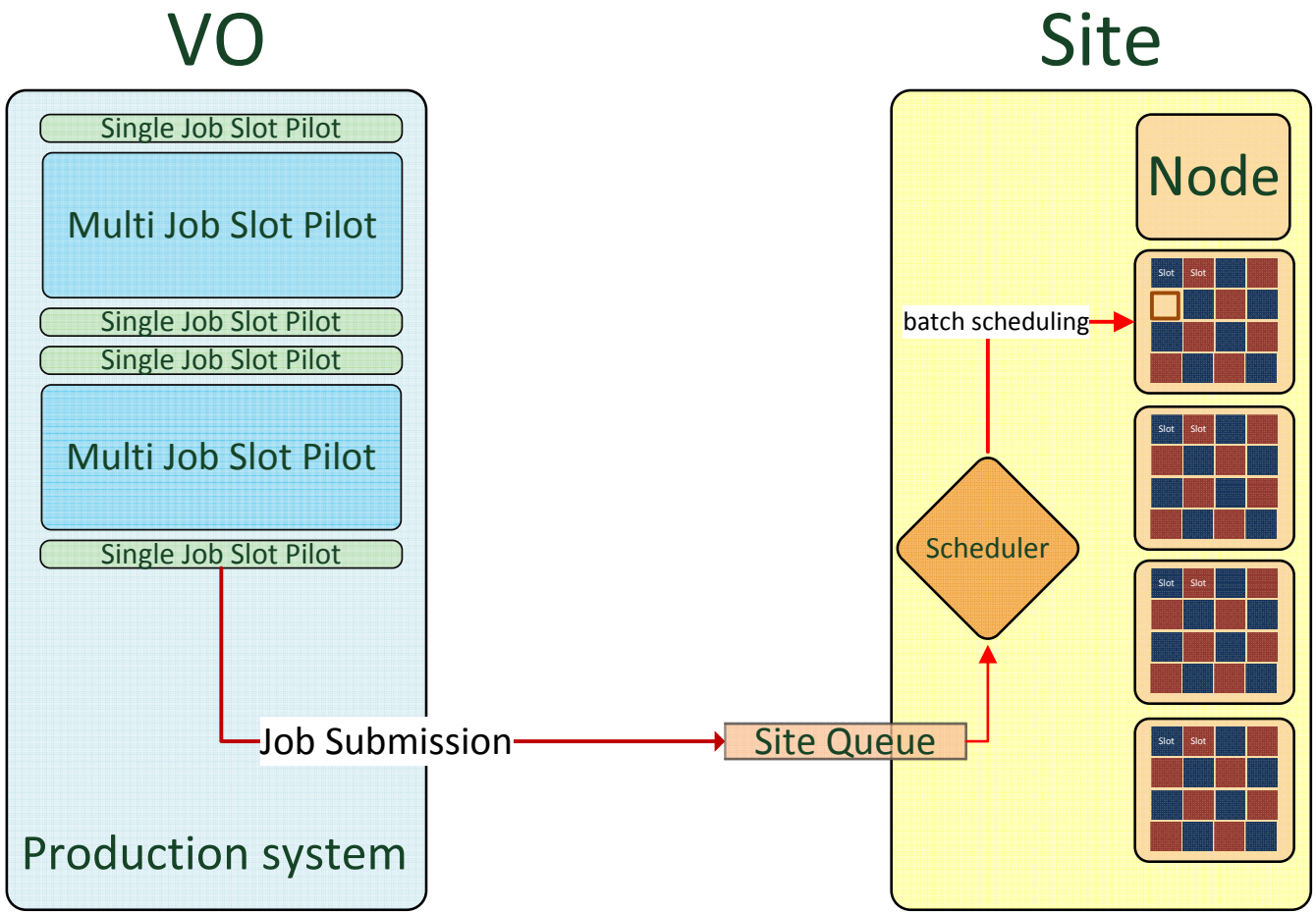
- wall time etc is measured in HEPSpec seconds
  - how *excact* have HEPSpecs to be for a VO/for a site?
- regardless where mcore jobs are scheduled $\rightarrow$ efficient scheduling relies on wall time prediction a priori
- how *good* can a VO predict a job's run time? I.e., how is the variance distributed between comparing predicted and actual wall times?

  $\rightarrow$ how efficient could an ideal scheduler be under which variances?

  i.e. how large would the inefficiency in node utilization become under which variance?
- if the wall time prediction is binned, i.e., in finite numbers of attainable run times/queues, how does the efficiency evolve with the number of prediction time slots?
- how are inefficiencies be accounted?
  - is it reasonable to account a VO when a job's run time deviates by $\times\sigma$ from the prediction and spoiling the scheduler?
  - it is accountable to the VO submitting a mcore job?
  - is it solemnly a site issue?

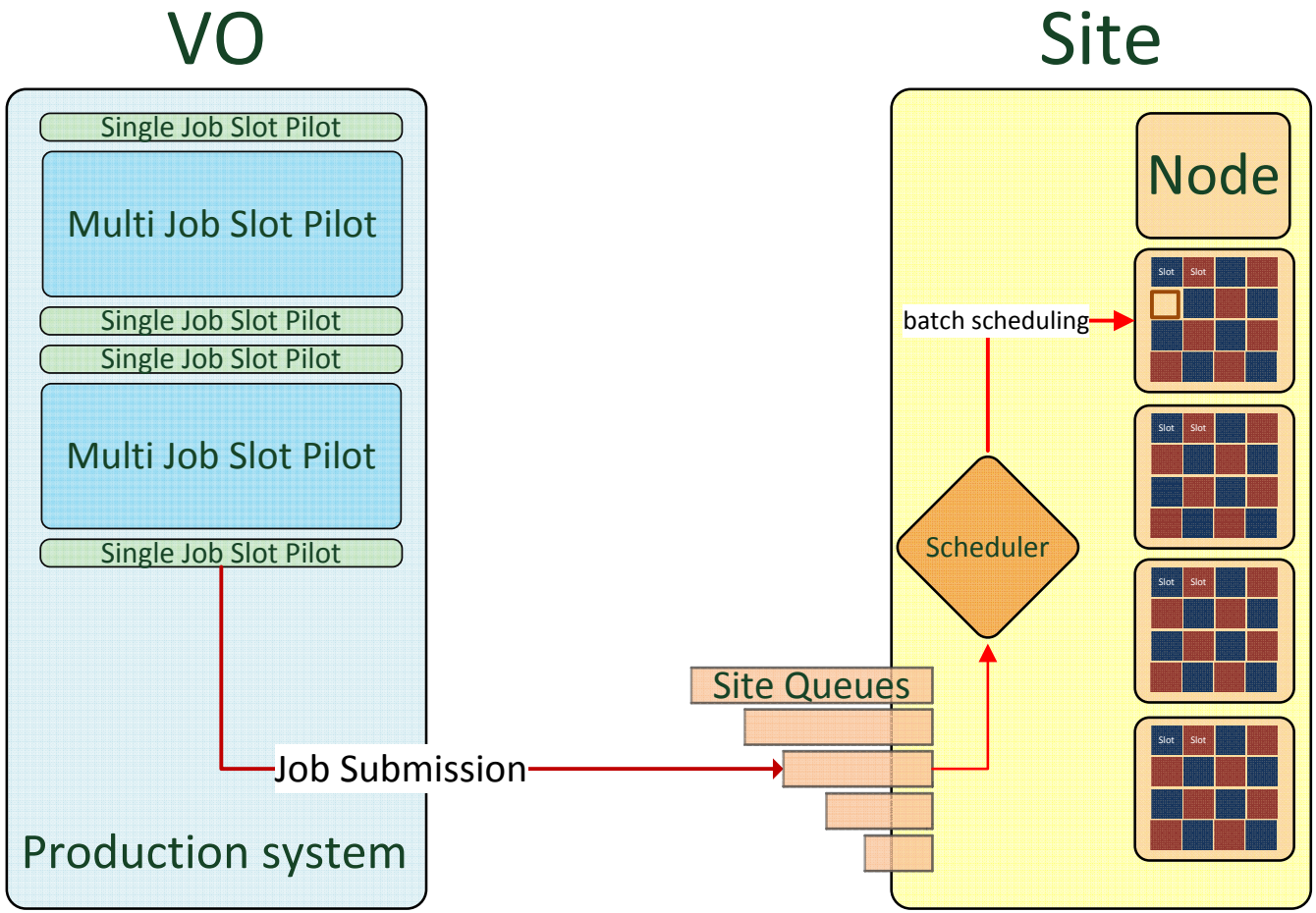
VO-Site job submission

- site setup: one long run queue
  - crucial: batch system efficiency
  - inefficient $\rightsquigarrow$ oscillating fair shareadjustment/job allocation



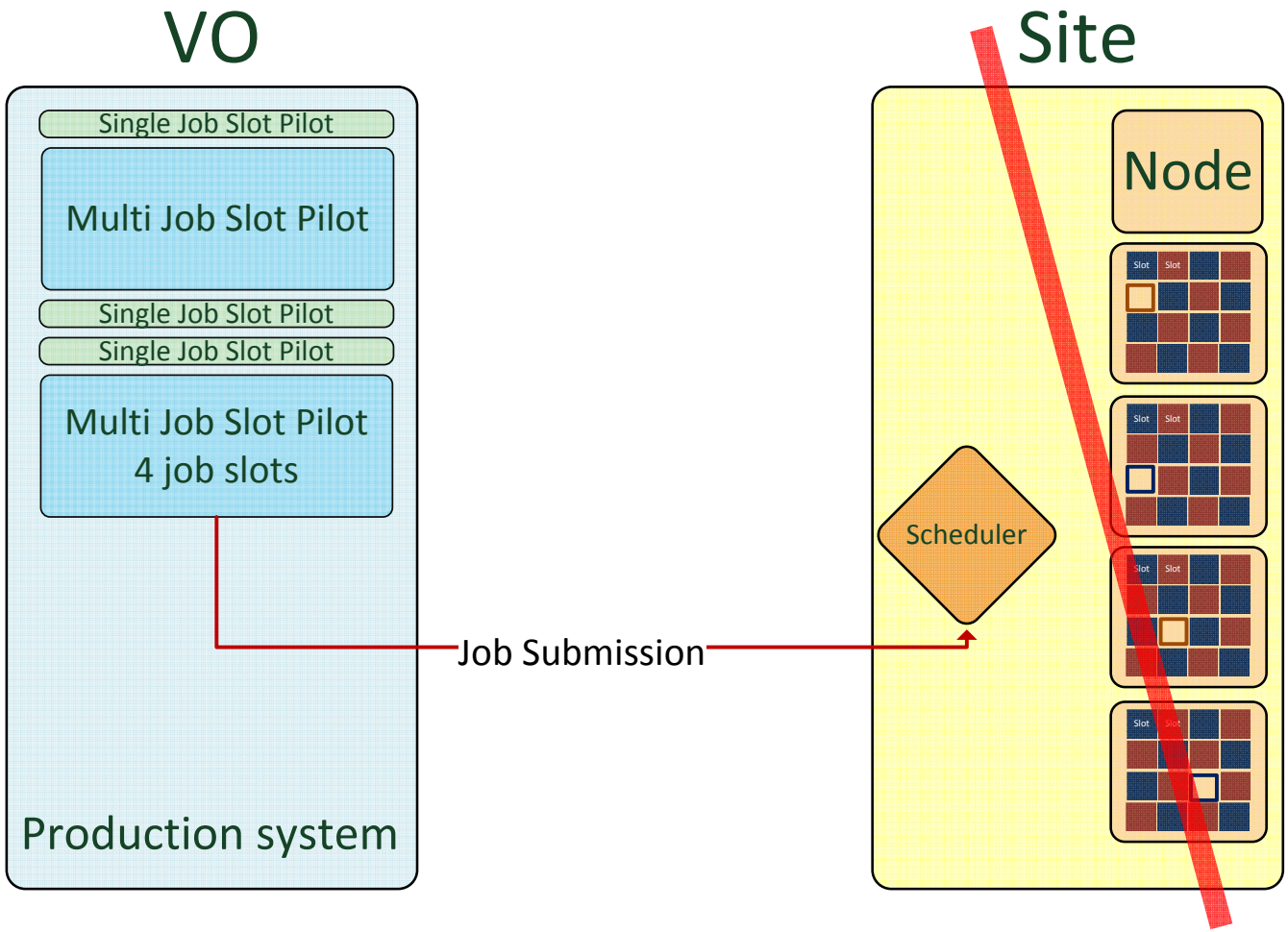
VO-Site job submission

- site setup: multiple queues various length
  - administration effort
  - non-WLCG users?



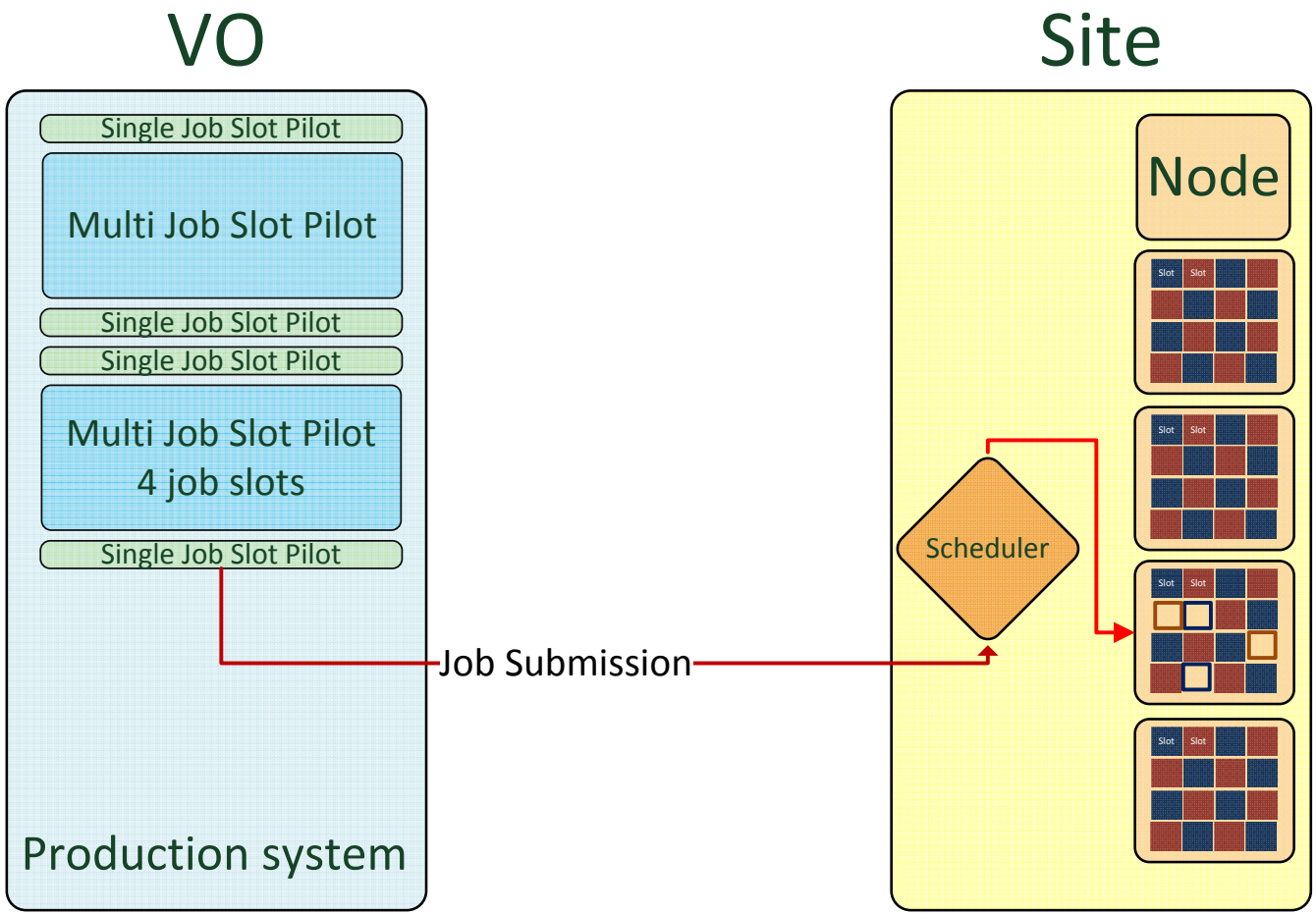
Multi Job Slot Jobs

- request for multiple job slots for one job
  - constraint: congruent job slots



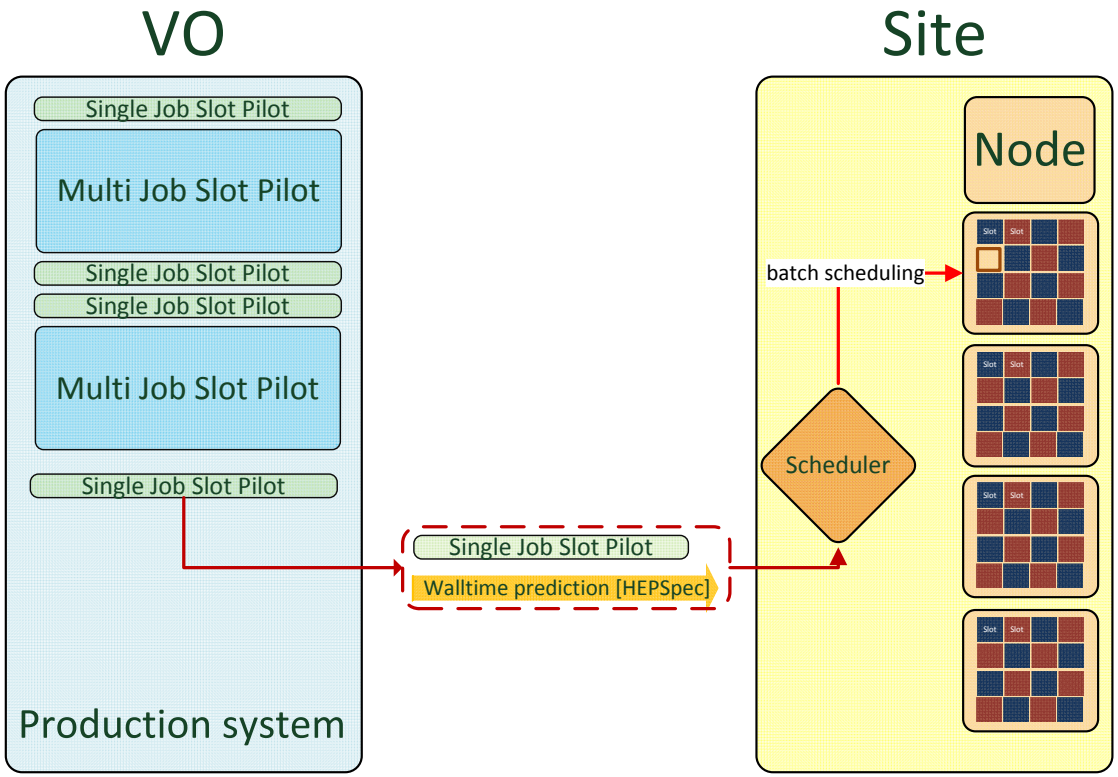
Multi Job Slot Jobs

- all job slots on one node
- farm $\neq$ HPC cluster



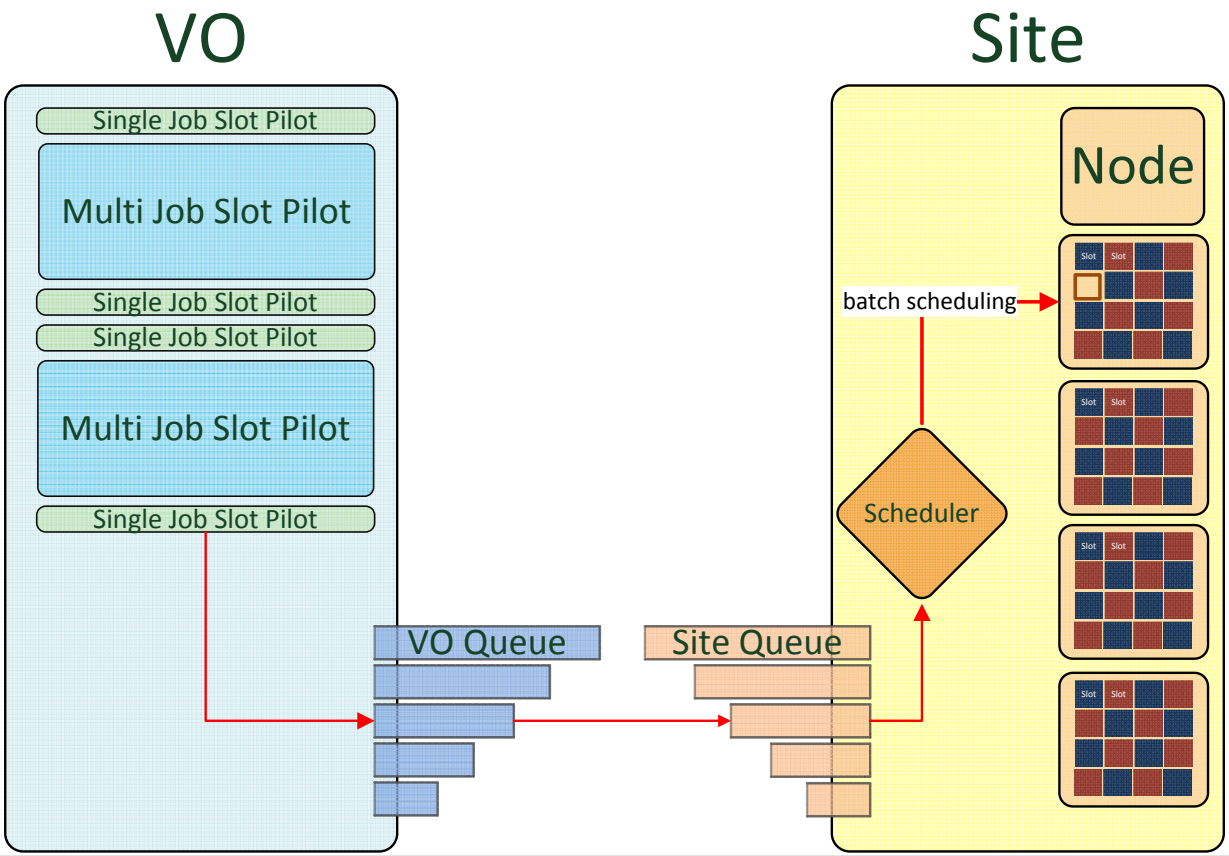
Job scheduling

- efficient scheduling [valid for scheduling either at site and at VO]
  - → crucial: reliable walltime prediction for job
- scheduling at site
  - would need to scale walltime in HEPSpec seconds!
  - BDII published HEPSpecs accurate enough?



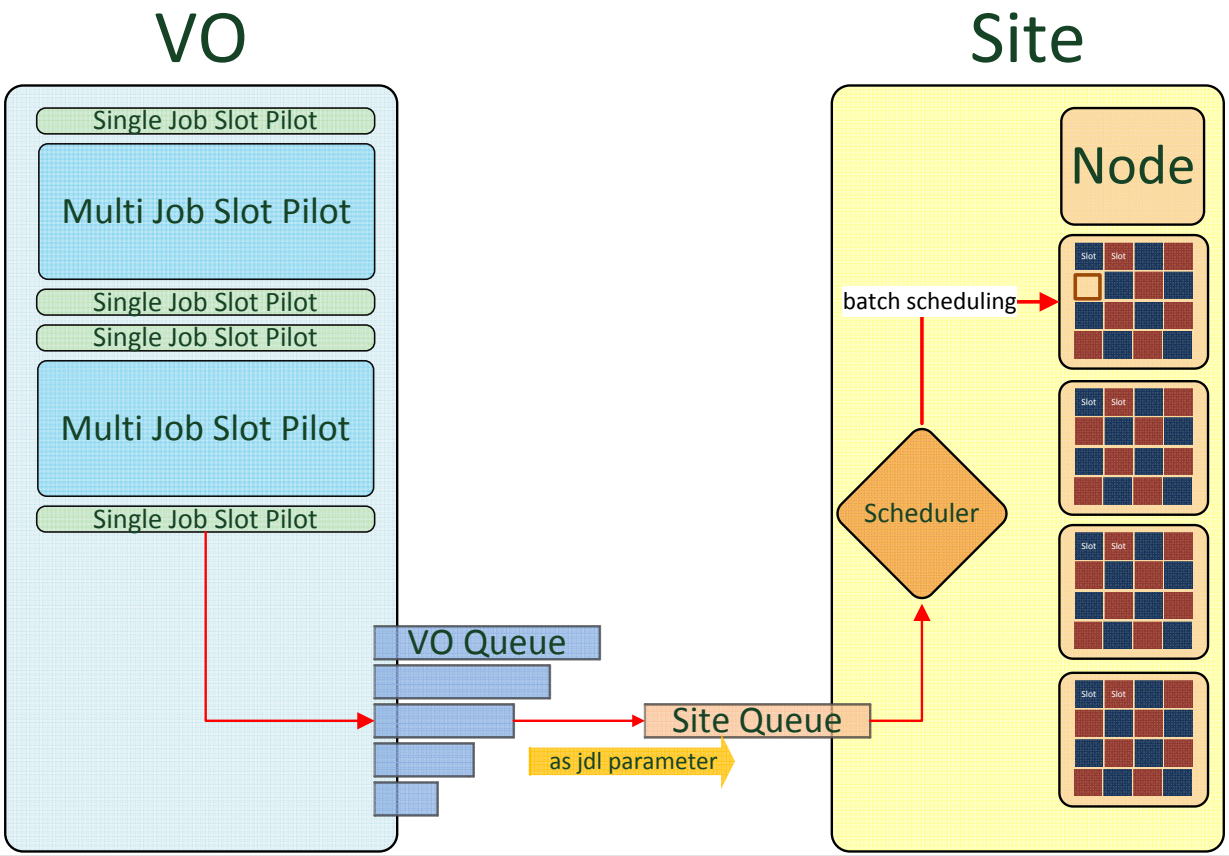
Scheduling@Site: *prediction emulation*

- $\rightsquigarrow$ no per job walltime prediction
  - scheduling@site via queue length
    - $\approx$ more coarse walltime prediction $\leftrightarrow$ multiple queues with increasing length
      - multiple VO and site queues



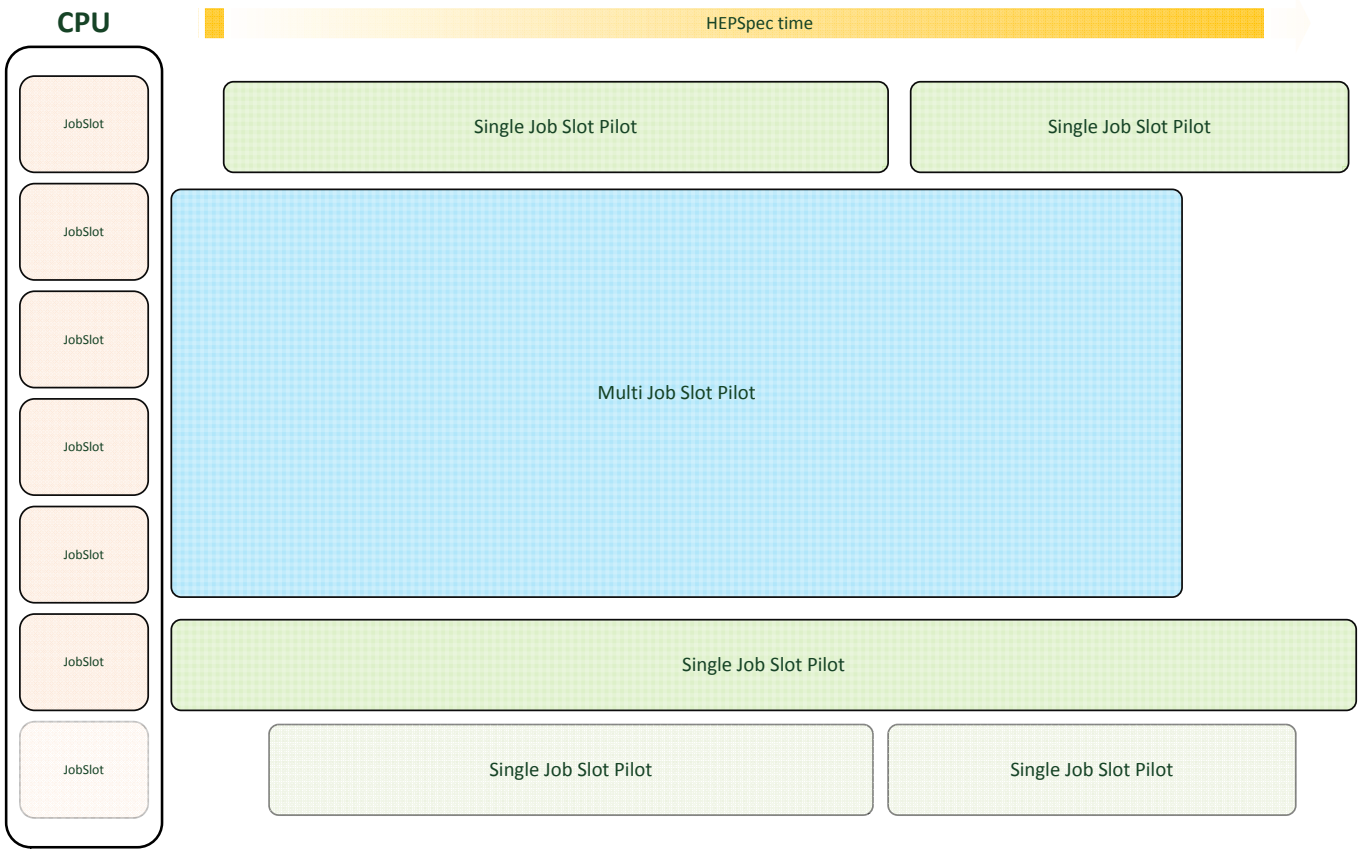
Scheduling@Site: *prediction emulation*

- $\rightsquigarrow$ no per job walltime prediction
  - scheduling@site via queue length
    - $\approx$ more coarse walltime prediction $\leftrightarrow$ multiple queues with increasing length
      - multiple VO queues to job parameter translation



Node view of single & multi job slot jobs

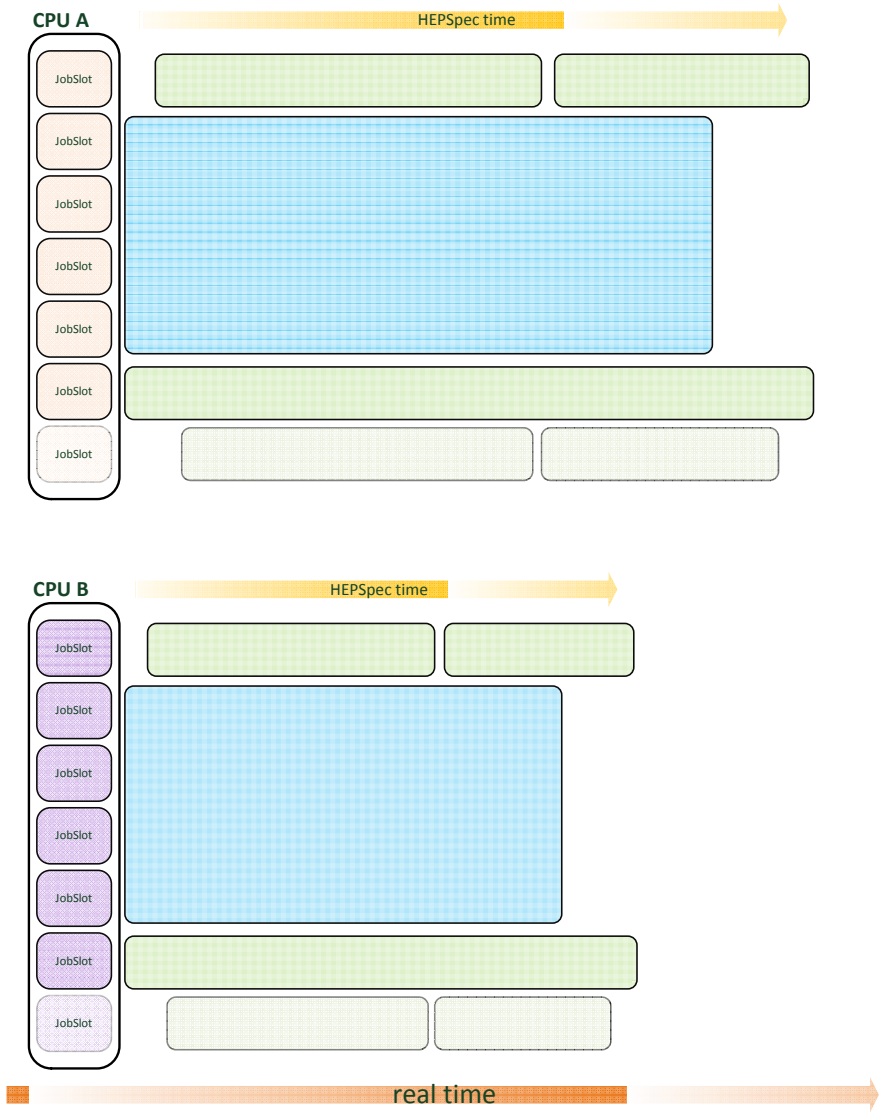
- CPUs with n job slots
- multiple single & multi job slot jobs from different VOs
- time in measured in HEPSpec sec.!



Real time vs HEPSpec time

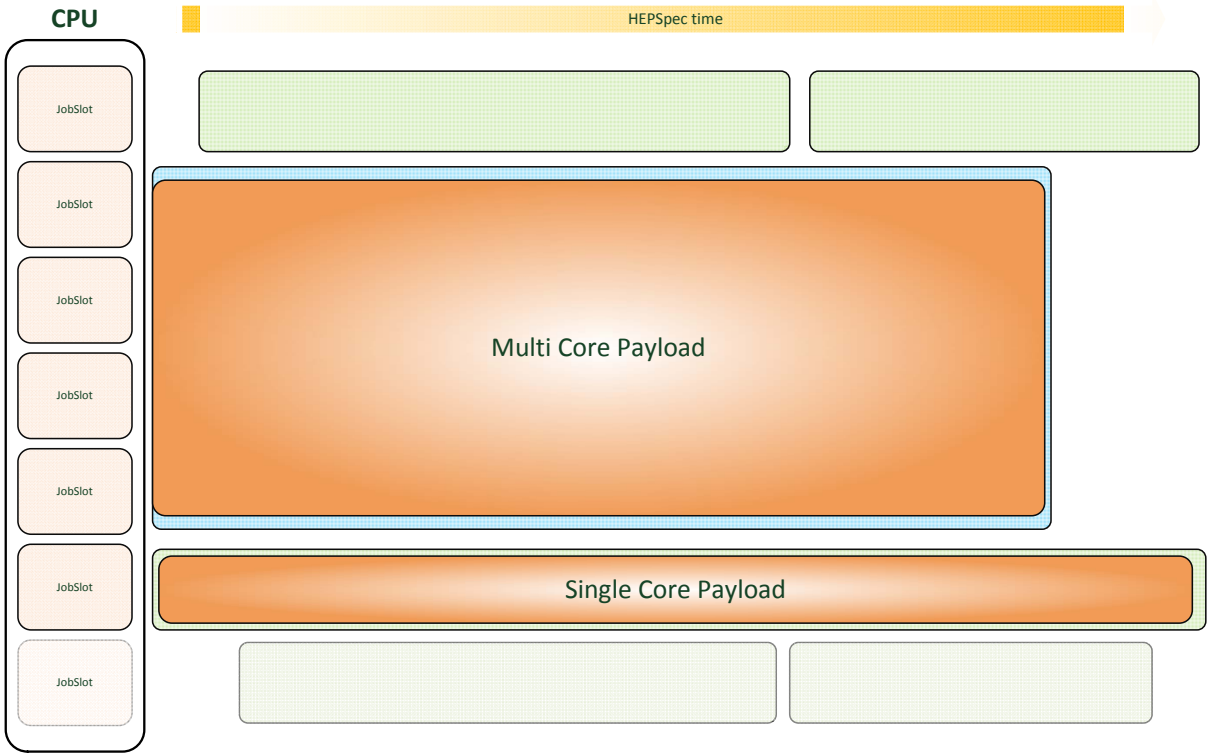
- walltime etc in HEPSpec sec.
- batch system would need to scale according to HEPSpecs

$\rightsquigarrow$ job walltime predictions in HEPSpec



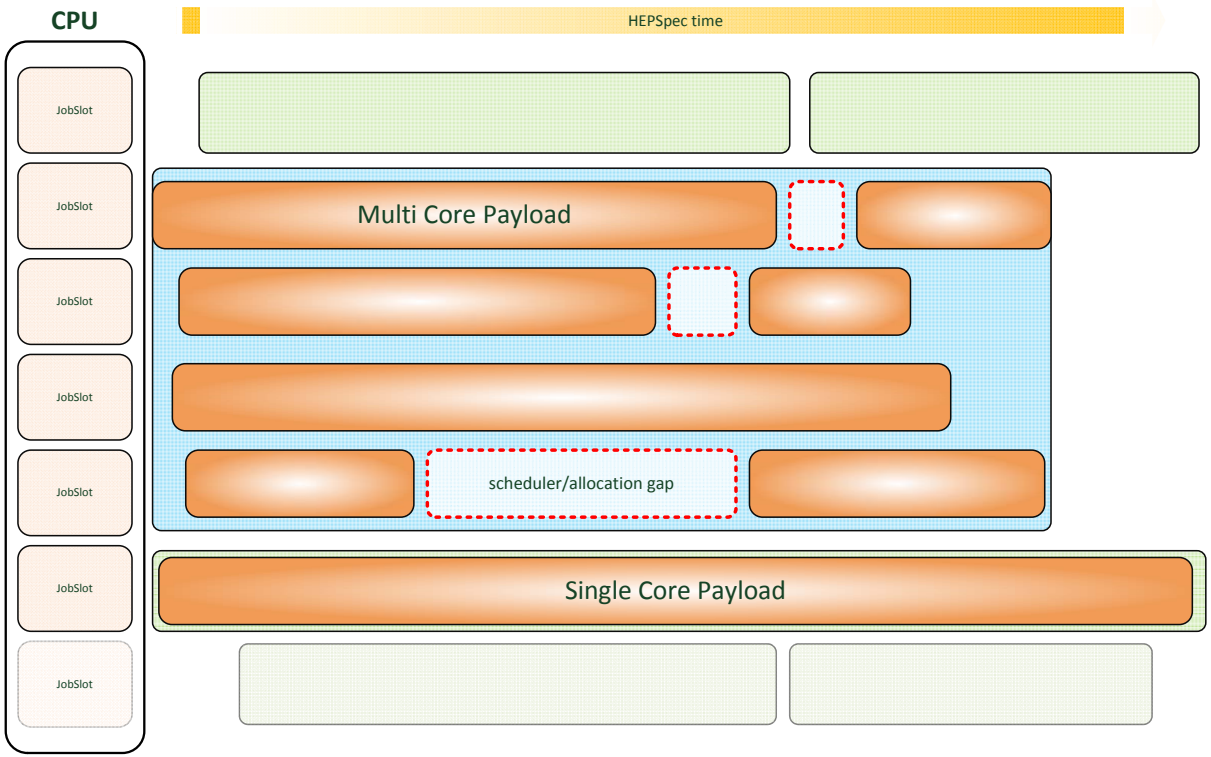
Congruent Payloads

- Congruent Payloads
- one payload per single job slot job
- one payload per multi job slot job



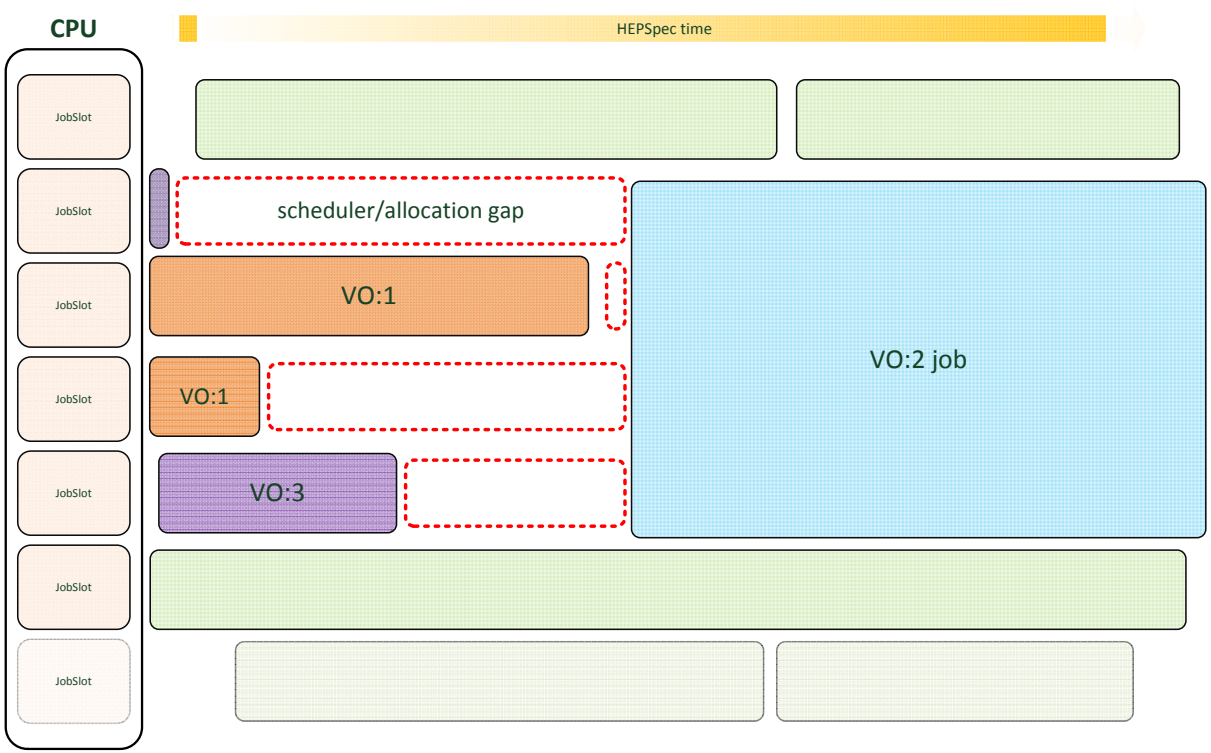
VO Scheduler: Multiple Payloads per Multi Job Slot Job

- multiple payloads reloaded in multi job slot pilot
- scheduler within pilot
- scheduling/job allocation gaps within VO



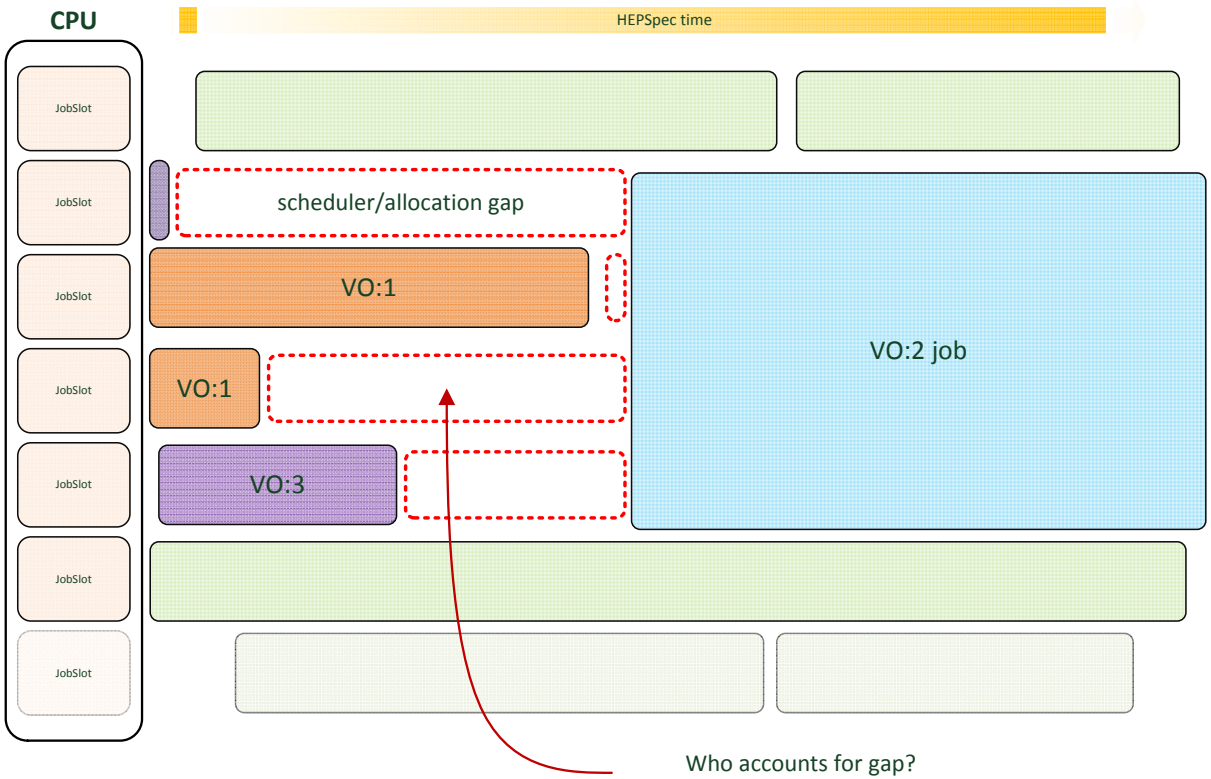
Site Scheduler

- scheduling multi job slot job at site
- applying to both: single/multiple payloads per multi job slot job
- constraint: x *congruent* job slots, i.e., x free slots on a *discrete* node
- for optimal scheduling: scheduler depends on accurate walltime prediction in HEPSpec



Site Scheduler

- who is accounted for scheduling gaps?
  - VO requesting multi job slot resource?
  - *VO of preceeding single job slot job?*



Site Scheduler

i.e. how to handle in-accurate walltime predictions

- VO requesting resource for xh while job submission
- acutal resource usage is yh $< x$h

- sites with WLCG and non-WLCG users?