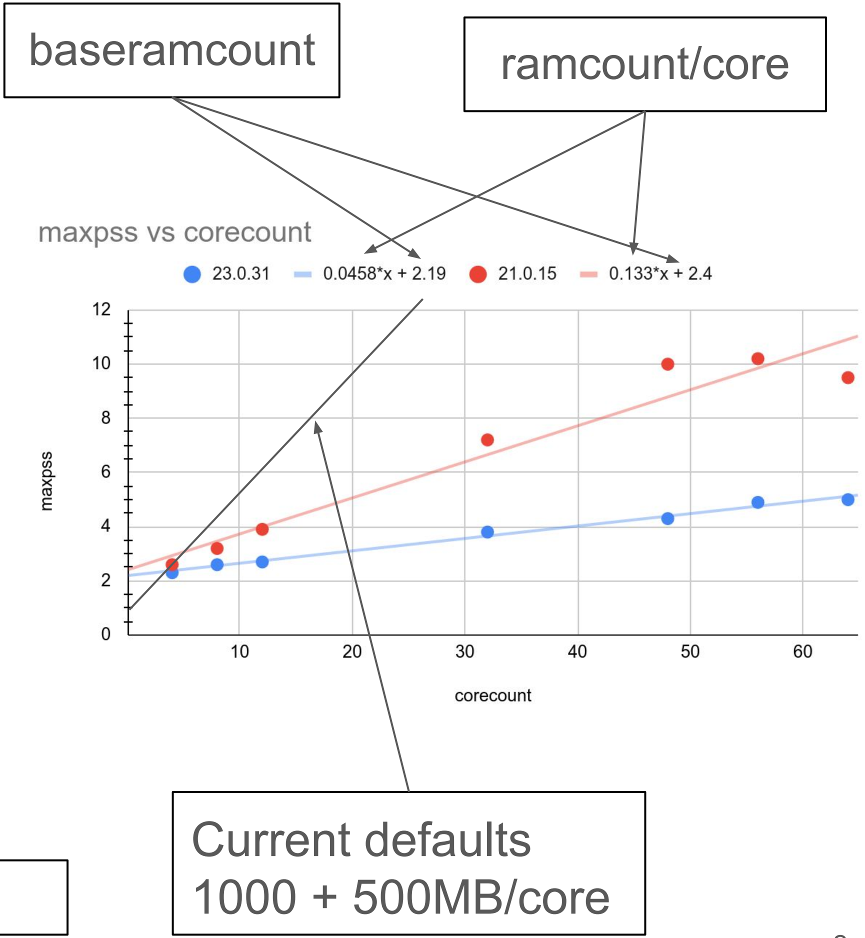
High memory resources

- Rod Walker , LMU

Motivation

- ATLAS have high memory workloads - some irreducible
  - Sherpa evgen, HI, AOD merge, ...
  - million histograms for user systematics
- Grid hardware does not change quickly
  - unclear we would ask for more RAM per core
  - need to make better use of what we have.
- MCORE simulation uses very little RSS
  - 300MB/core on 8cores, 80MB/core on 64
  - many other workloads ~1GB/core
  - but all reserve 2GB/core
- more cores of high RSS resources
- colocation with data, e.g. to merge
  - leads to much data transfer

Some justification from Gen/Sim people, e.g. HepMC3, AF3, HI, #volumes.



How to run himem at more sites

- Submit with requirements that CE pass to BS
- Batch Systems can pack nodes according to requirements
    - mix hi and lomem jobs on a node to keep cores full
- *Pull* model has streams of pilots with the same requirements
        - 4 sub-resources: SCORE, SCORE_HIMEM, MCORE, MCORE_HIMEM
            - 2 memory ranges(per core): 0-2GB, 2GB-maxrss
            - wide range leads to over-provisioning, e.g. job needing 2100MB, has 6GB reserved
    - increase granularity
        - minimum required granularity 0-1,1-2,2-4,4-6 . Better 1GB range and up to 8GB
- *Push* pilot submitted with specific requirements of pre-loaded job
    - MB granularity on memory but also walltime, disk space, cores

Maintaining job mix

- Staying below 2GB/core on PQ avoids site admin grief and accounting issue
  - 2GB is site dependent, often higher. Needs to be site config.
- Have crude limit to stop himem jobs
  - resource_type_limits.HIMEM - limit running cores, i.e. N_SCORE_HIMEM + corecount*N_MCORE_HIMEM
- Better mechanism to stay below site meanrss/core (in dev)
  - Running job sum(job.minramcount)/sum(job.corecount) < site.meanrss GB/core
    - stop dispatch of jobs with minramcount>PQ.meanrss
  - overshoot and oscillation may need tweaks
- What if we want to use pledged resources inefficiently
  - have high priority tasks and willing to leave cores idle
  - easy: don`t maintain job mix
  - unhappy site admins would need accounting solace.

Sites and accounting

- Can VOs request this flexibility of pledged resources? Yes if…
  - stay below mean RSS/core OR accounting includes idle cores
- Current accounting is core HS23 * walltime seconds
  - site wants full HS23 accounted when cores full OR RSS full
  - reserve 2 cores for 4GB serial job? Works but no, because we only use 1 core
    - someone else(maybe same VO) can use that core
- sum(job.minramcount)/sum(job.corecount) /site.meanrss, over running jobs
  - <= 1: account all jobs with full HS23/core
  - >1 means cores *could* be idle.
    - effective HS23 scaled by requested RSS per core / site mean RSS
      - no queued jobs, not blocked: still account higher HS23?
  - Jobs effectively using more than 1 core, but some using less than 1 - not integer

Other VOs

- Do they also have high memory workloads?
- CMS already have some flexibility from glide-in but at a cost
  - packing works best with many cores, i.e. whole-node
  - this is the case for native BS but not for CMS 8-core, 16GB glide-in
    - internal packing restrictive and wasteful. Rules out intra-VO sharing
    - but nothing prevents larger glide-in/whole-node
  - similar comment to Cobald-Tardis but can share between VOs
- Hope to put on agenda for WLCG workshop in May
  - feedback welcome to hone argument