main Gen2->Gen3 concerns/pointers from my perspective

processCcd.py (Gen2, v19_0_0) -> Lee's "step1" (Gen3, v23_0_1)

- Major increases in output data volume (~3x) and number of inodes (~3.7x) per raw DECam CCD in Gen3 compared to Gen2
  - Cannot stop major contributors to Gen3 data volume increase from being written without breaking the "Quantum Graph"
  - Gen3 outputs are much more heavily/deeply nested within sub^{N}-directories
  - Gen3 file names much longer and more complex, mixture of conventions regarding how exposure name is specified e.g., "ct4m20170818t232625" versus "670212" within the same exposure/CCD's Gen3 outputs
- Runtime with Gen3 default parameters much longer (factor of ~3+, on our hardware) with Gen3 default parameters
  - Reverting the psfDeterminer back to PSFEx from "piff" and explicitly setting the number of PSF iterations to 1 (default is 2) brings Gen3 runtime down to only a factor of ~1.2-1.25x longer than Gen2 (based on tests reducing exact same raw data in both Gen2/Gen3)

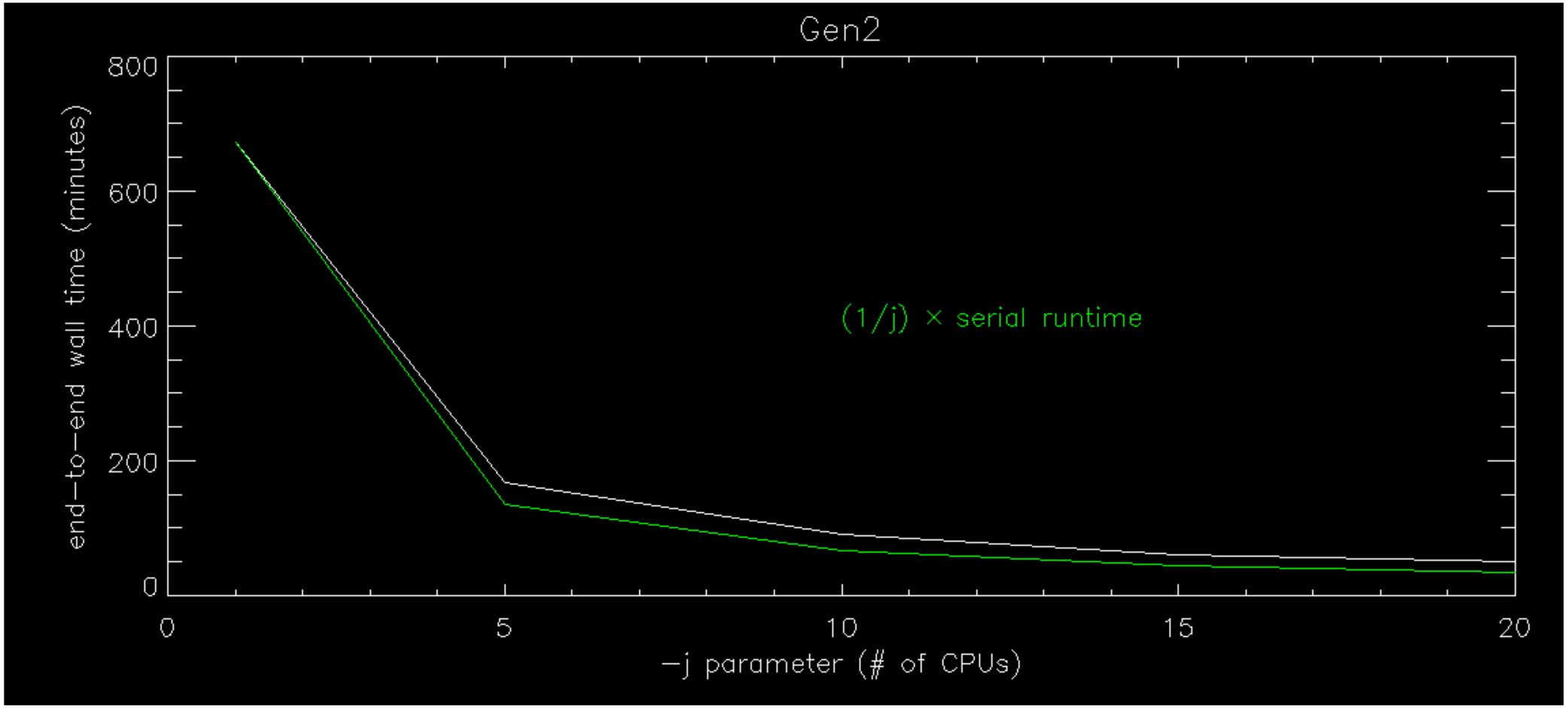
main Gen2->Gen3 concerns/pointers from my perspective

processCcd.py (Gen2, v19_0_0) -> Lee's "step1" (Gen3, v23_0_1)

- The specific config parameters used for the aforementioned two runtime optimizations were:
  - '-c characterizeImage:measurePsf.psfDeterminer.name="psfex"'
  - '-c characterizeImage:psfIterations=1'
- The scaling of runtime with "-j" level of parallelism (on our hardware) seems much worse for Gen3 than for Gen2, though this could still use some further investigation
  - Off the cuff hypothesis would be that this is due to increased I/O from much larger numbers of inodes and ~3x larger output data volume
- Thanks to Lee for his HackMD notes and Shenming for his excellent Gen3/DECam Google Doc!!

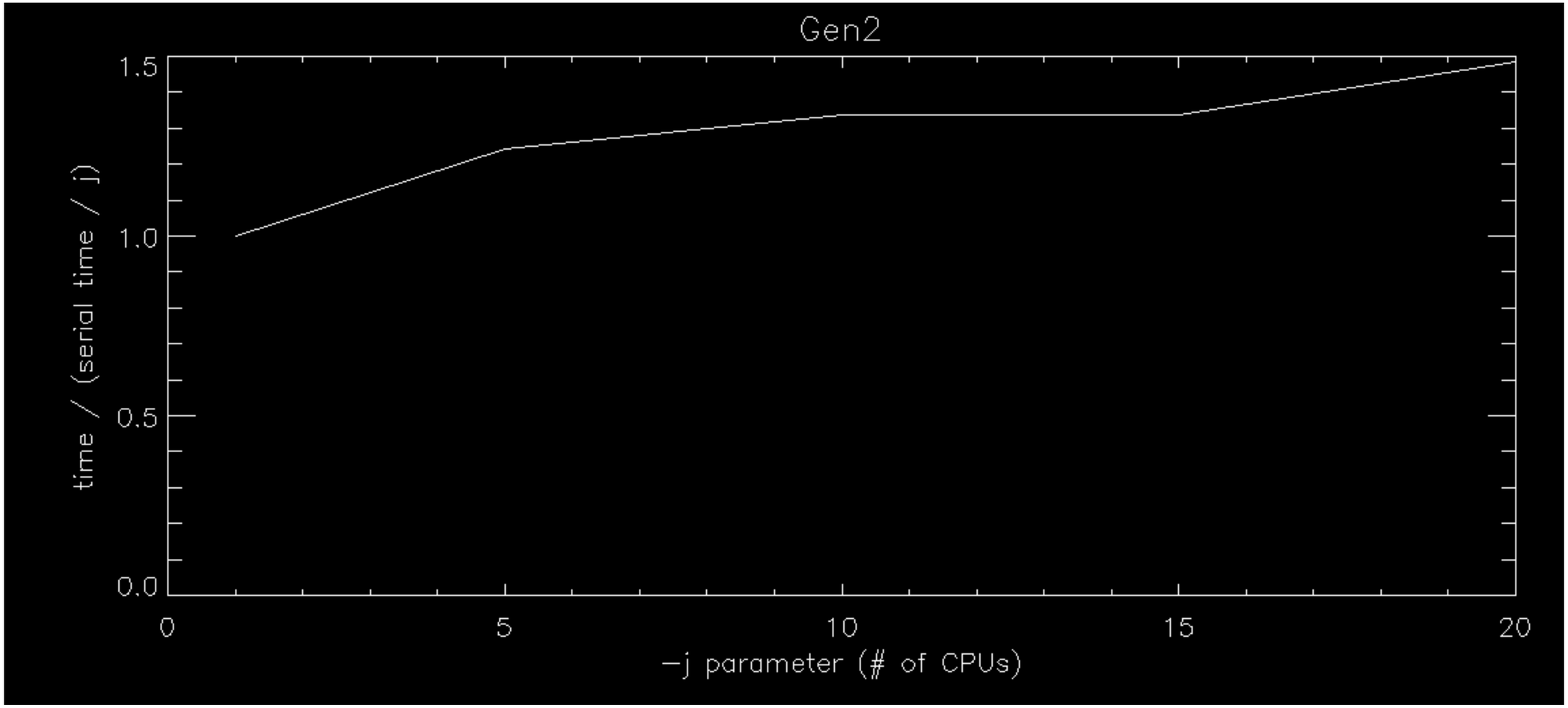
scaling with number of CPU's (-j arg)

all 1500 CCD's from Shenming's GW170817 data set



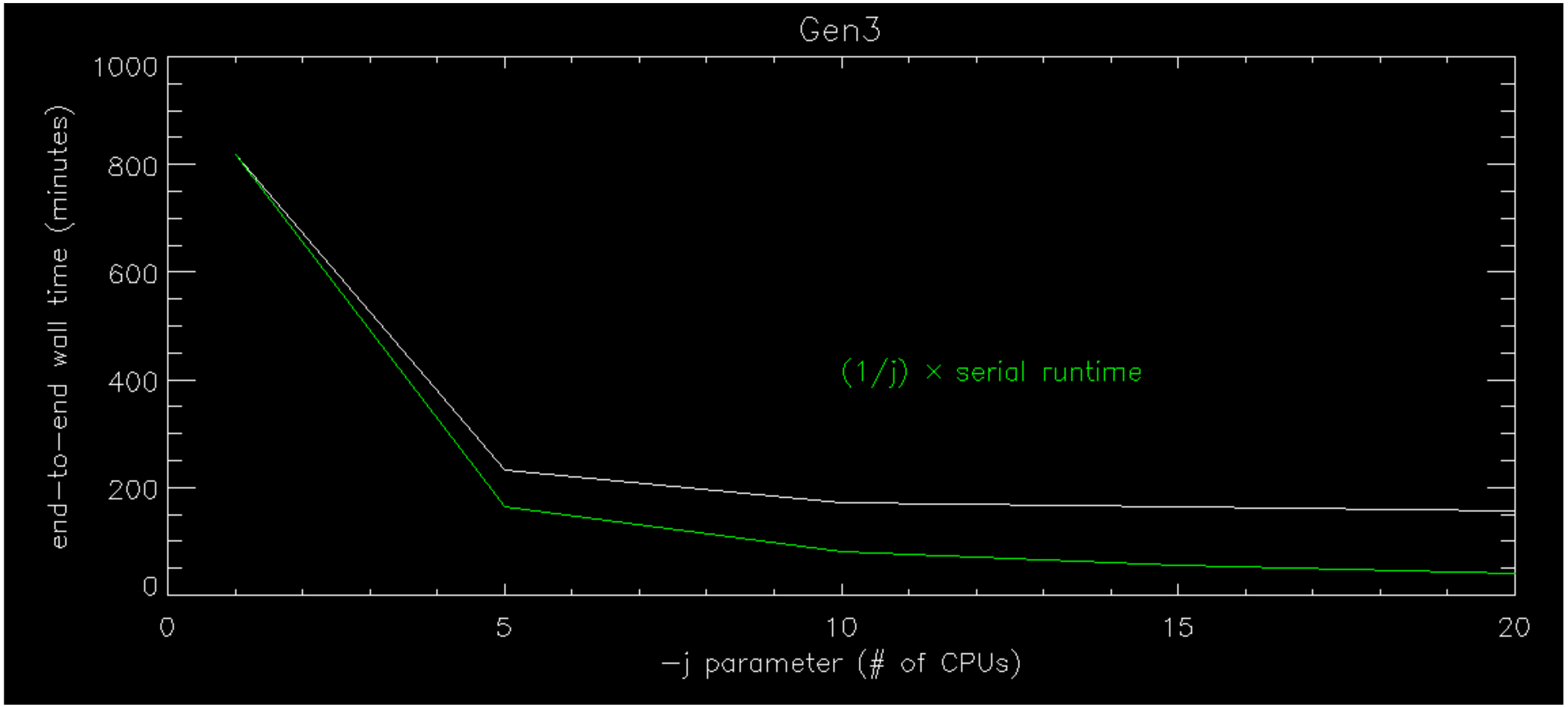
scaling with number of CPU's (-j arg)

all 1500 CCD's from Shenming's GW170817 data set



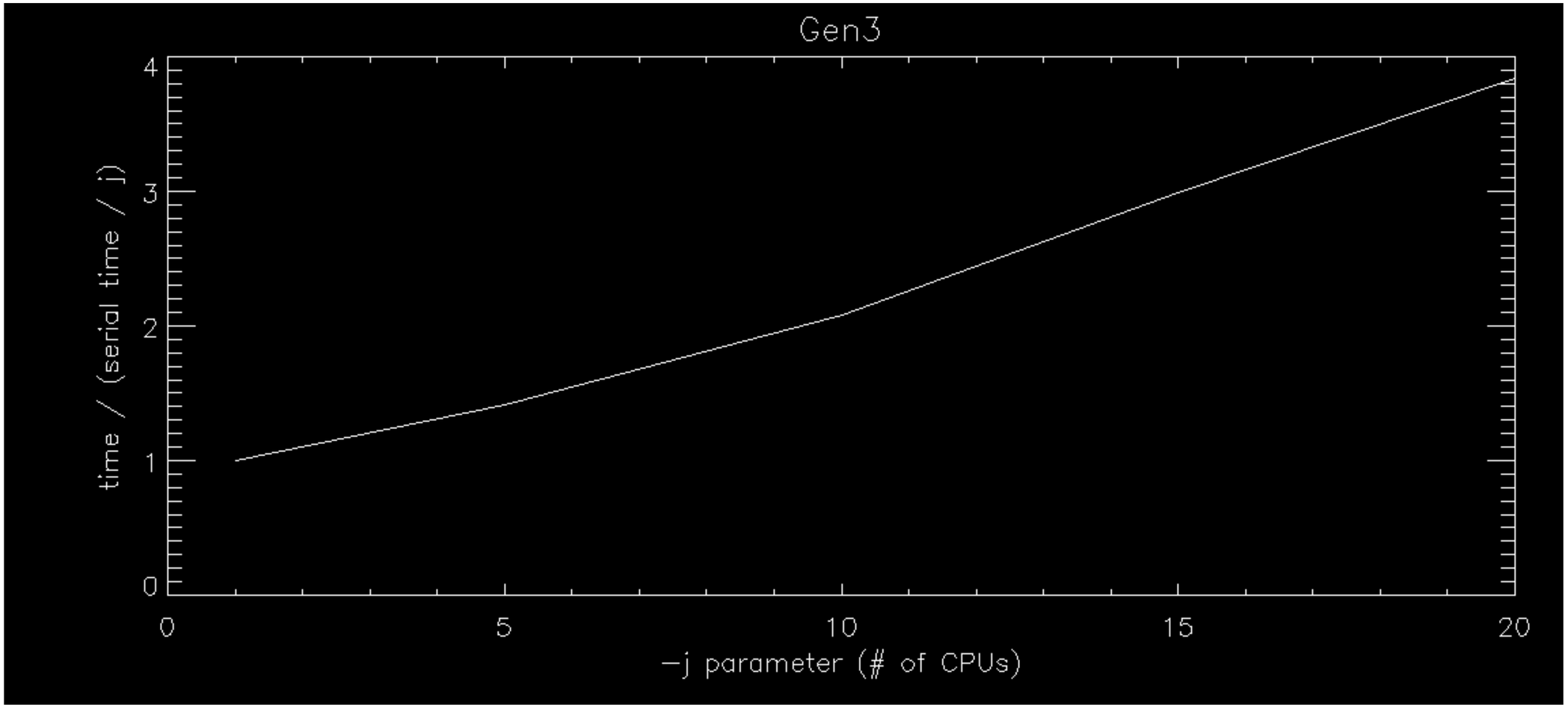
scaling with number of CPU's (-j arg)

all 1500 CCD's from Shenming's GW170817 data set



scaling with number of CPU's (-j arg)

all 1500 CCD's from Shenming's GW170817 data set



scaling relative to serial is ~2.5x worse at j = 20 for Gen3 relative to Gen2

Question

- Do the LSST pipelines include some sort of “telemetry” utilities to monitor the resource usage, for instance the peak memory and runtime (preferably for each CCD)?
  - I have my own bespoke code that parses log files and can sort of do some stuff like this.