Eiger benchmark with DIALS

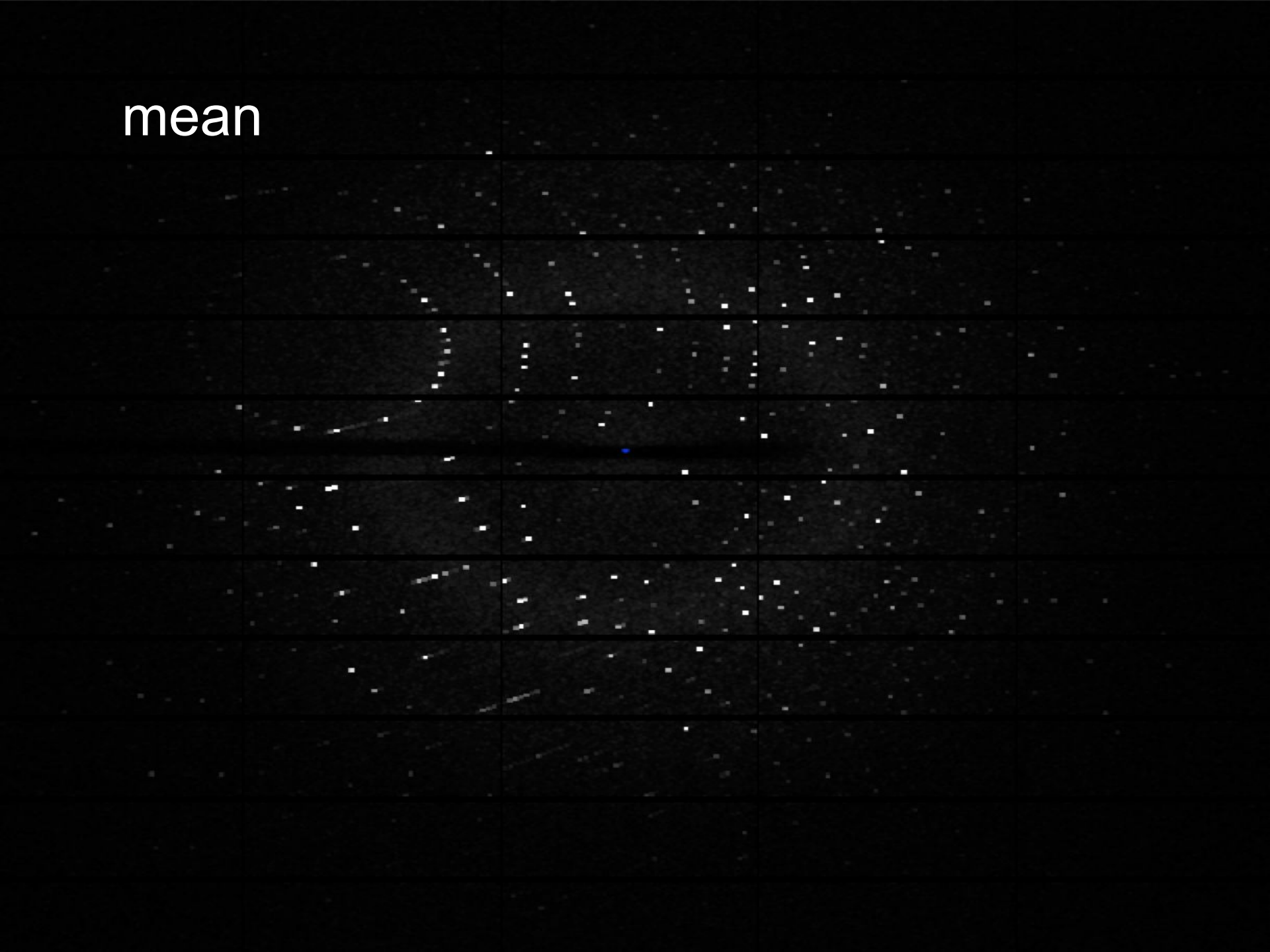
HDRMX

The Problem

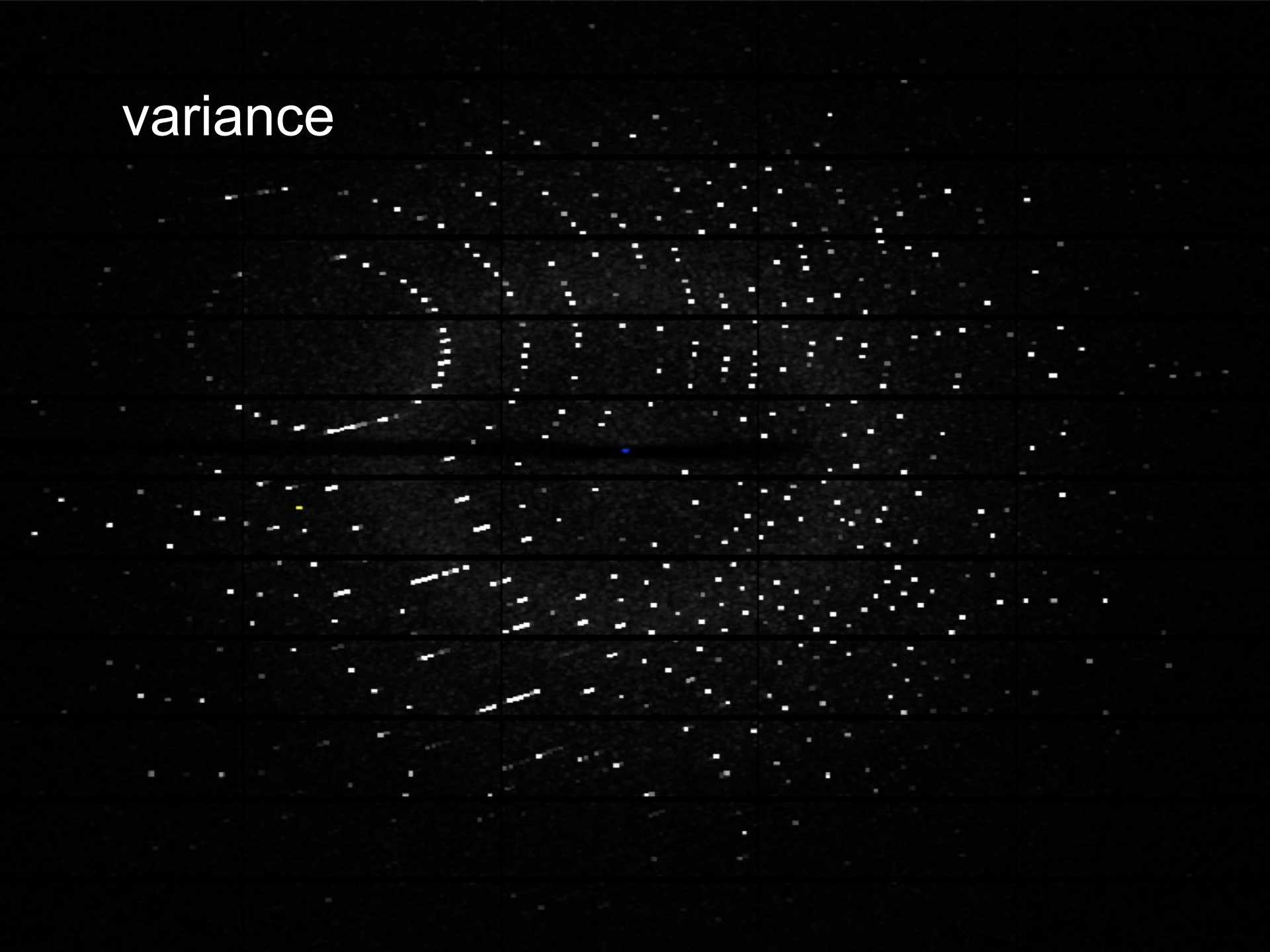
- Current detectors (DECTRIS Pilatus) run at up to 100 frames / s
- Next generation Eiger detectors run up to 750 frames / s (for 4M)
- *This rate will probably not be routinely used for data collection*
- This rate *will* be used for raster scanning i.e. to allow a large loop to be sampled with a fine beam in a short time (e.g. X-ray centering)
- For raster scanning the experiment has to wait for the results so this is time critical
- Therefore in first instance principle benchmarking problem is spot finding

raw data

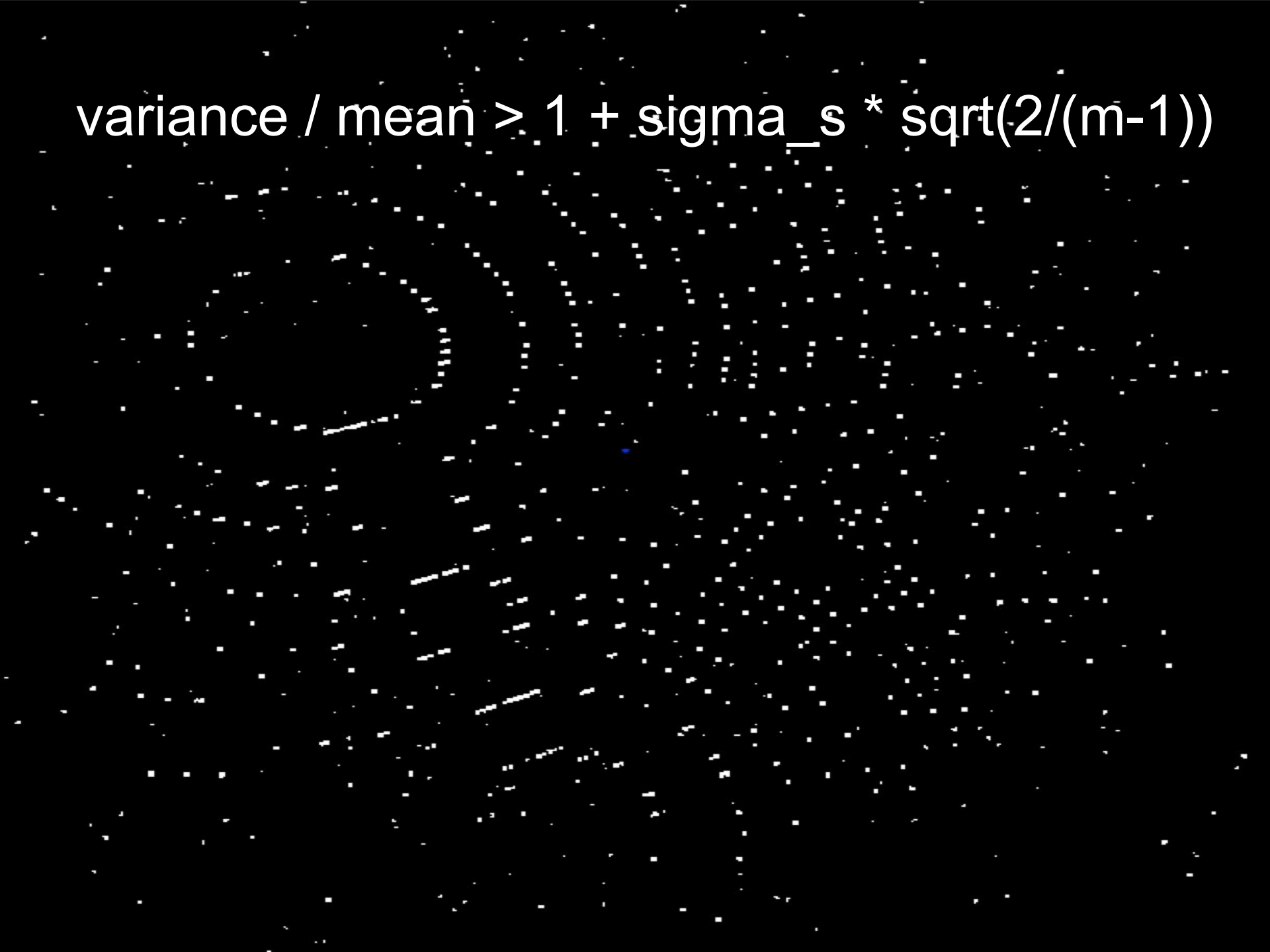
mean

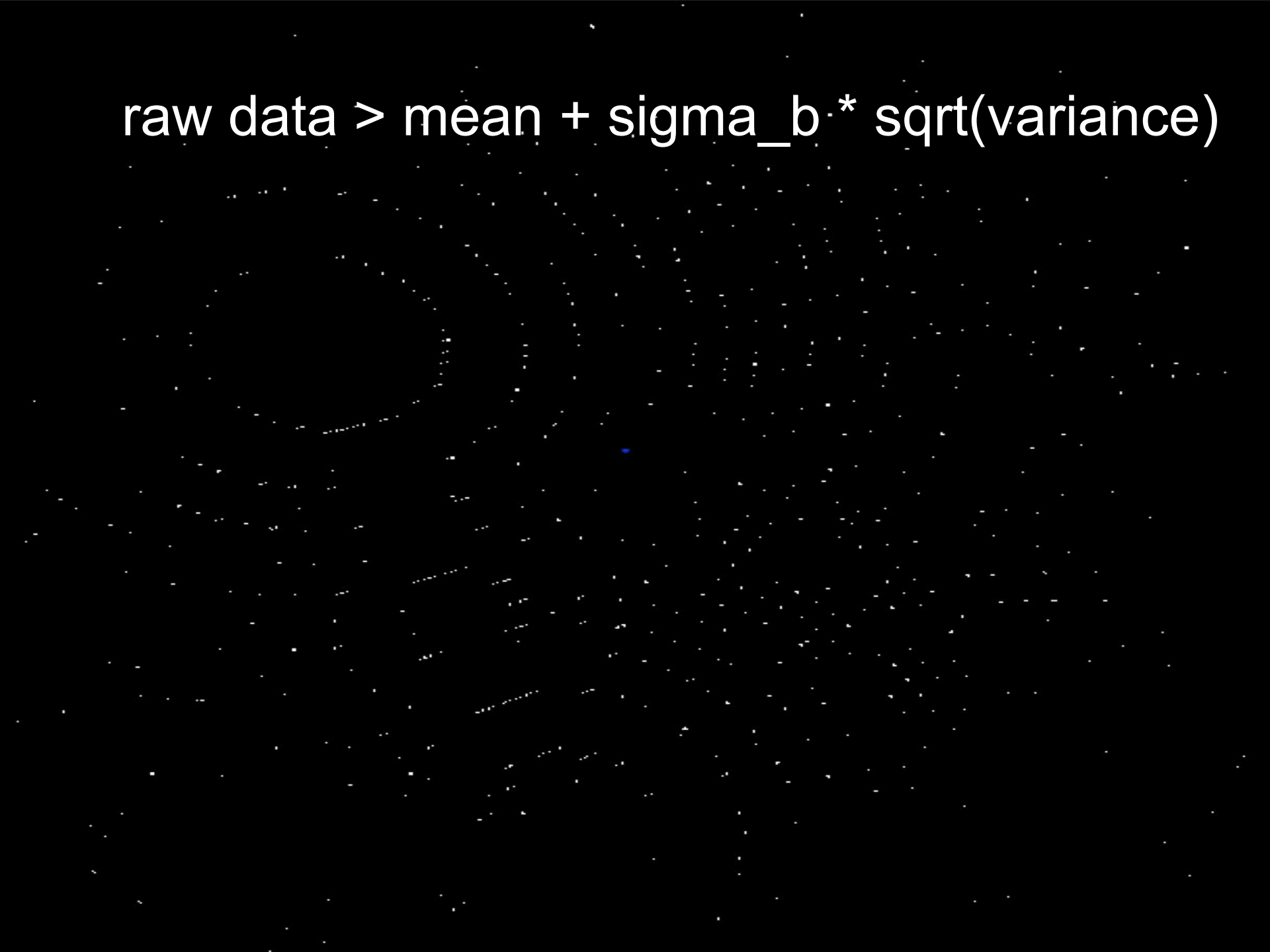


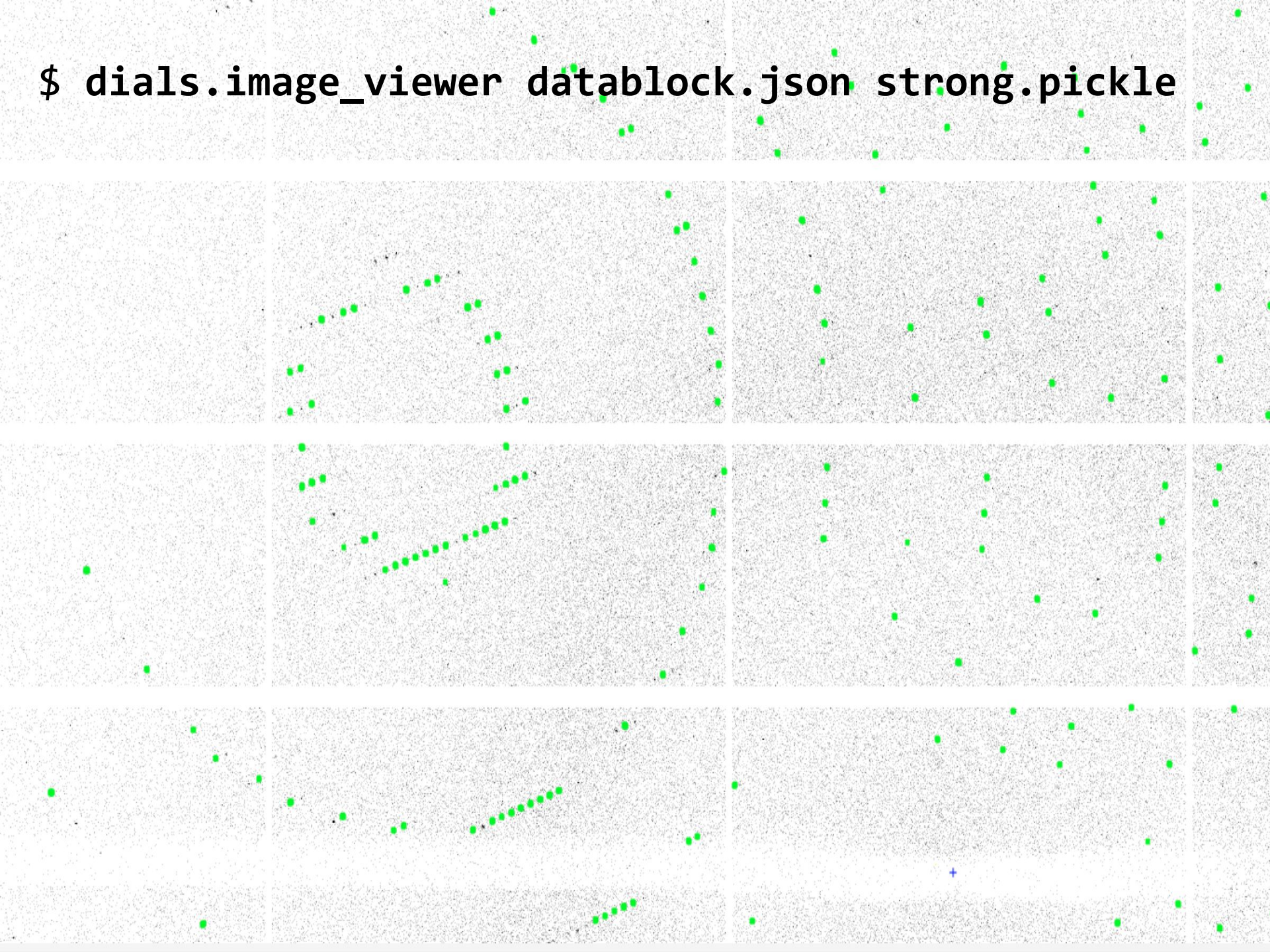
variance



variance / mean


variance / mean > 1 + sigma_s * sqrt(2/(m-1))


raw data > mean + sigma_b * sqrt(variance)

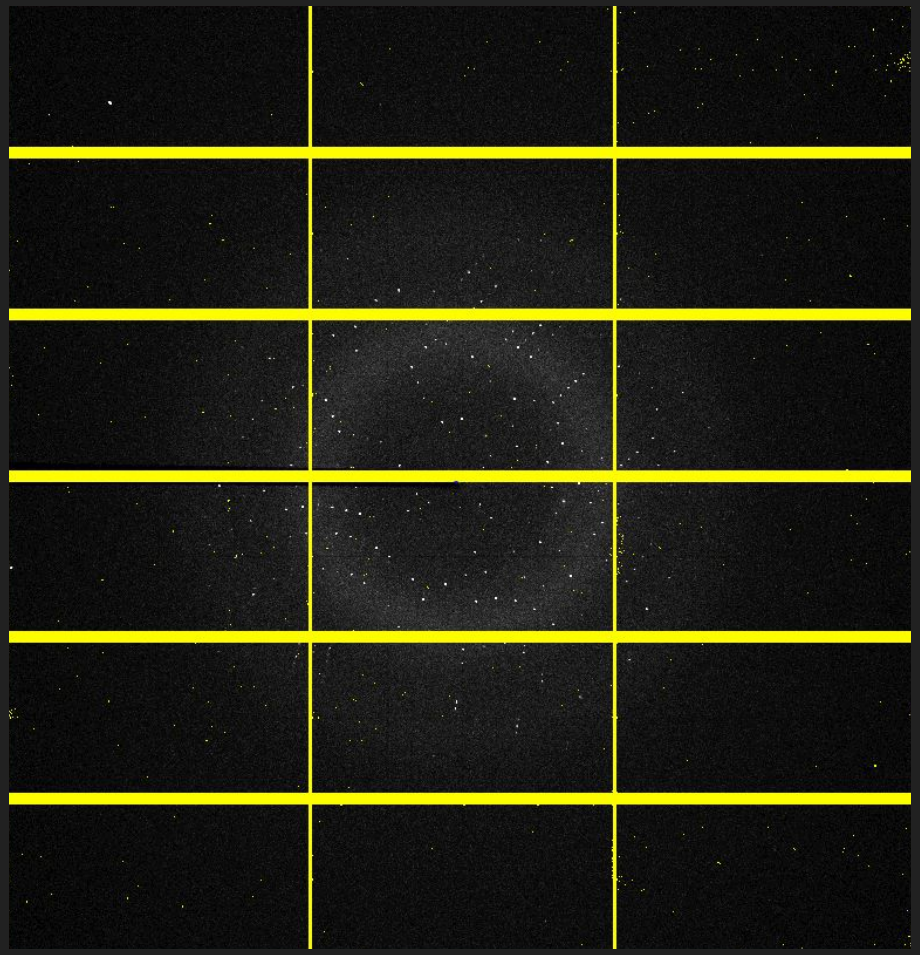

$ dials.image_viewer datablock.json strong.pickle

Optimizations for grid scans

- Reduce overhead - do not keep pixels since data are 2D (for raster scans) and we only want #spots etc
- Divide up data set into uniform large chunks - reduce overhead in launching tasks
- Here only considering single node performance i.e. given grid scan, how long does processing take

Benchmark data

- Data provided by Martin Savko @ Soleil
- Eiger 9M grid scan, 750 points (30x25, 10μm pitch) HDF5 format, collected from Transthyretin sample



Benchmarking systems

- System 0 @ DLS (standard cluster node)
  20 core (2 x 10 core “Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz” + 128GB RAM) no HT
- System 1 @ ALS BL831 (James Holton)
  72 core (4 x 18 core “Intel(R) Xeon(R) CPU E7-8870 v3 @ 2.10GHz” + 1024GB RAM) with HT so 144 max threads
- System 2 @ CCI (viper.lbl.gov)
  32 core (4 x 8 core “AMD Opteron(tm) Processor 6284 SE” + 256GB RAM) with HT equivalent so max 64 threads

Benchmark

Performed with dials 1.3.1 linux binaries (same binary set for all systems)

```
dials.find_spots datablock.json nproc=${nproc} shoebox=false
```

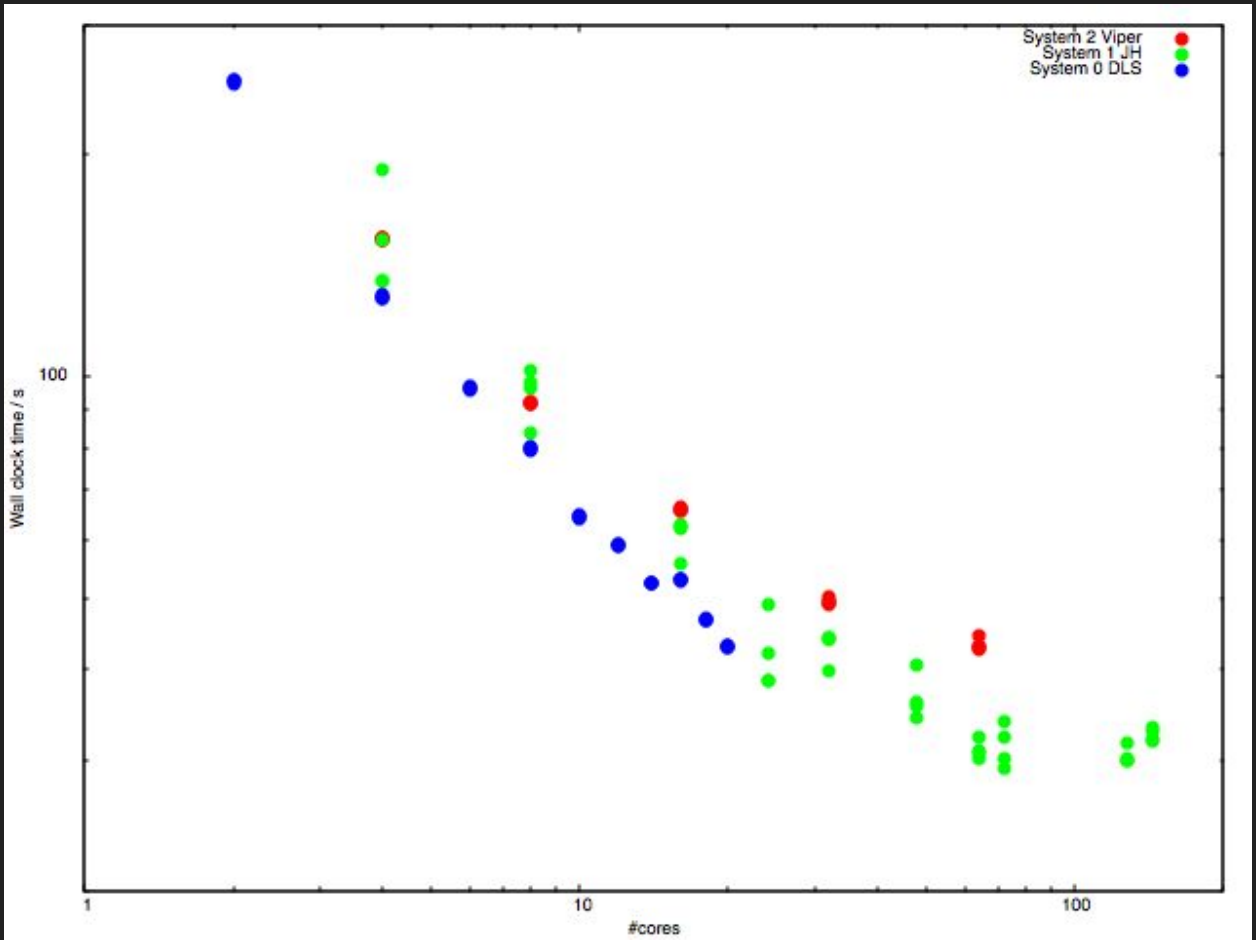
Principle consideration wall clock time i.e. from starting process to results becoming available

Here nproc=4…# in system

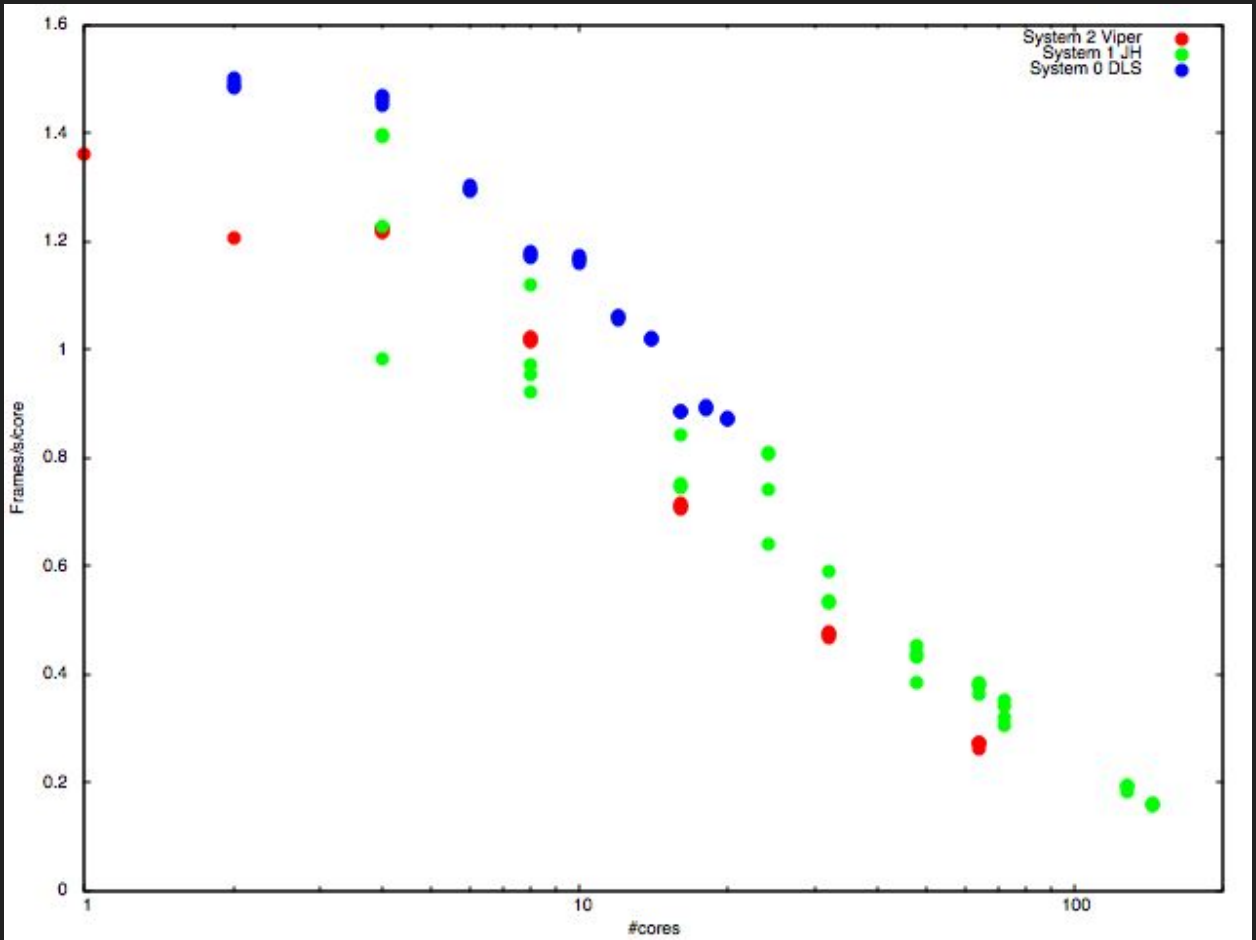
Data come from RAMDISK => file system performance not a consideration

NUMA structure could be a consideration

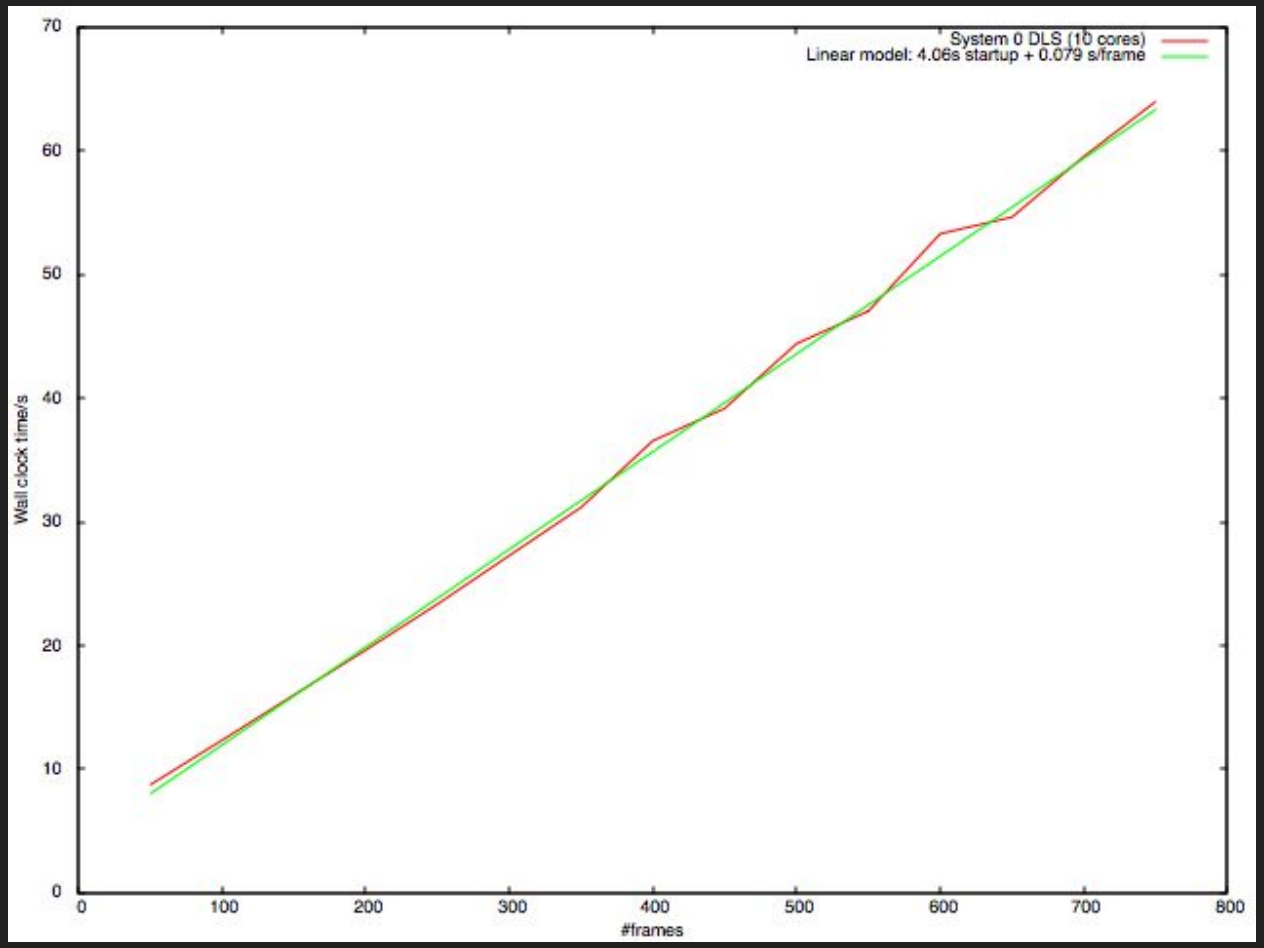
Results 1: #cores - wall clock



Results 1: #cores - rate / “efficiency”



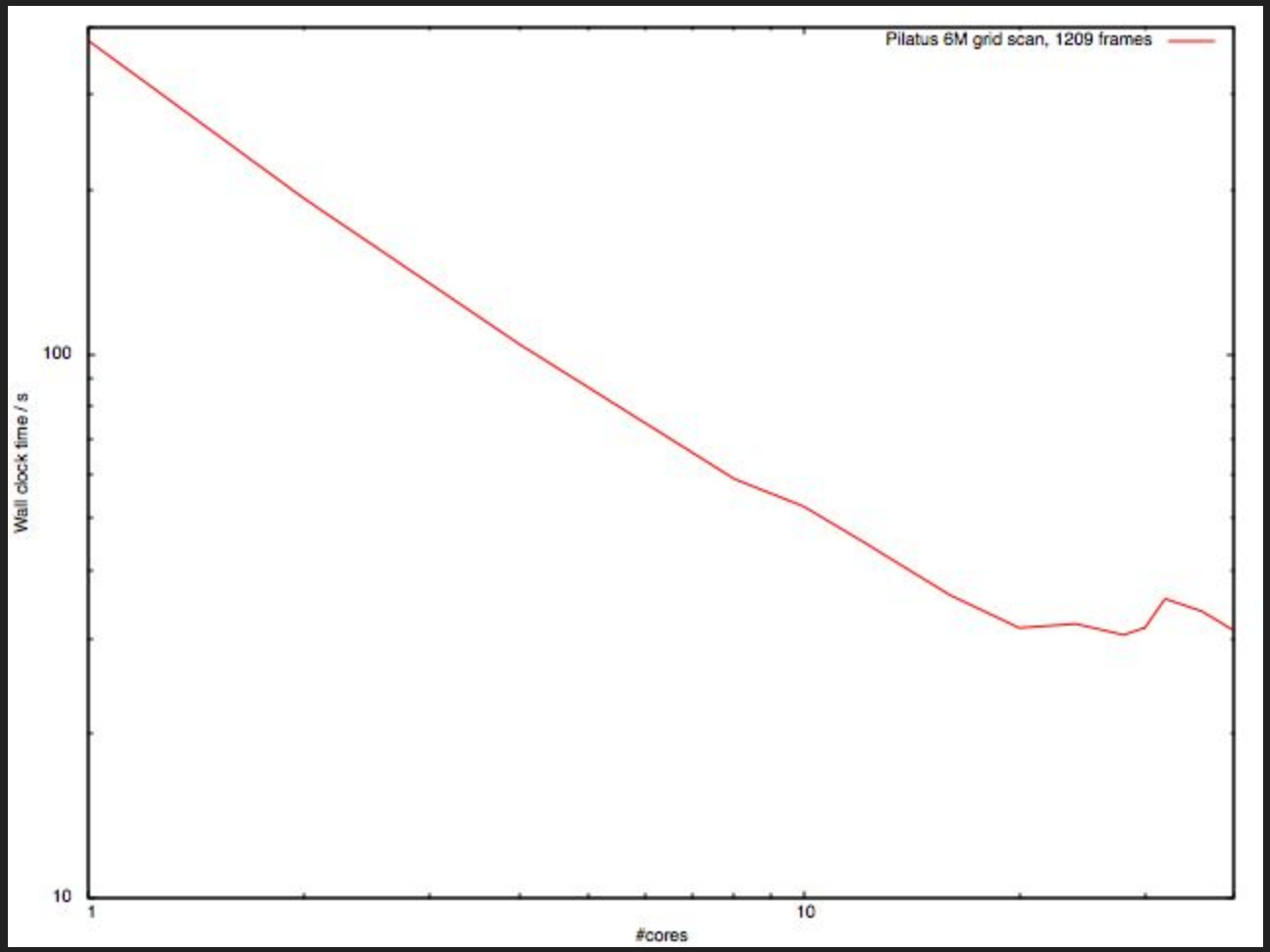
Benchmark 2: #frames (10 cores on system 0)



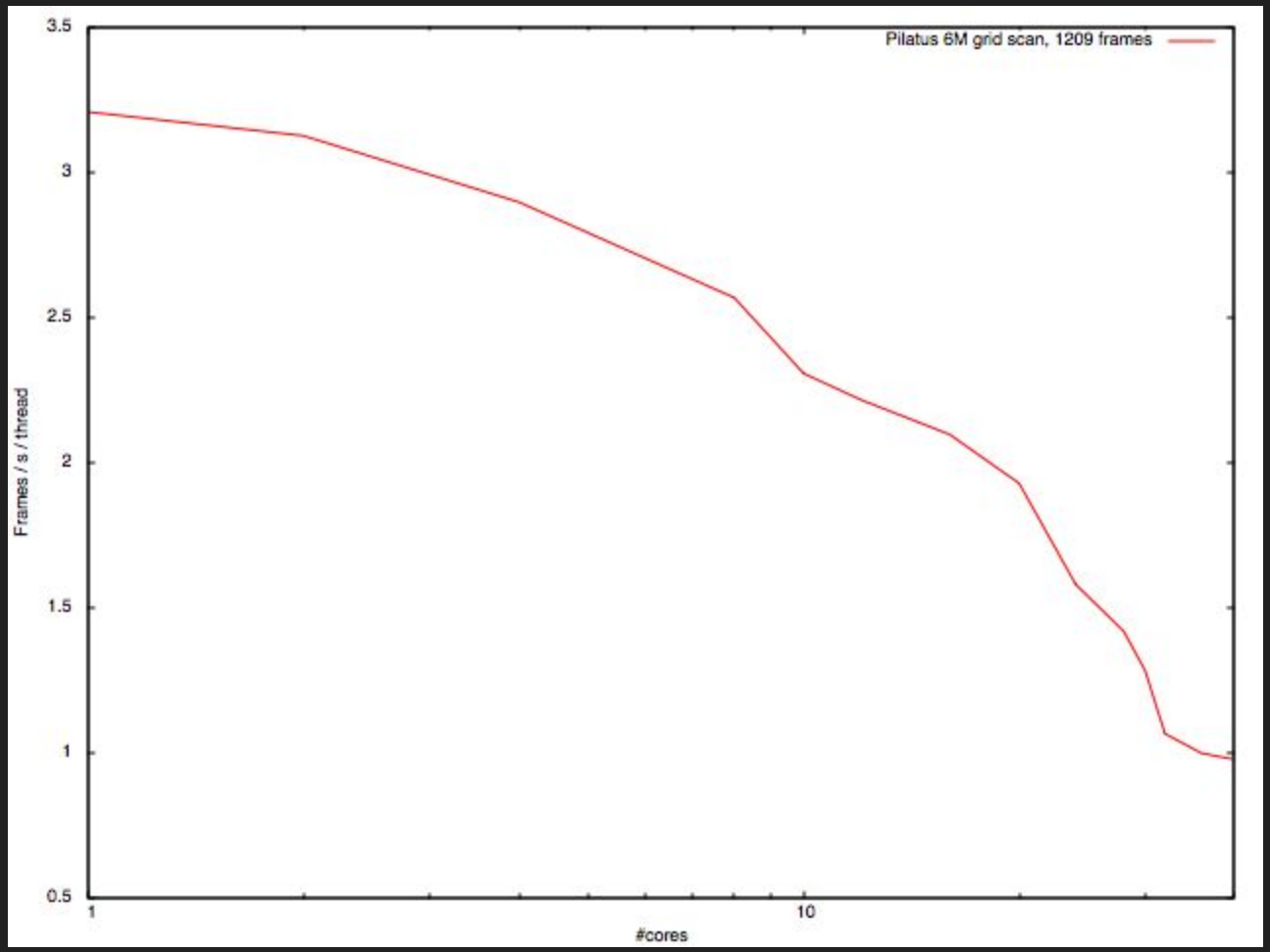
Second test: Pilatus 6M data

- 1209 image grid scan recorded as part of ongoing development work at Diamond Light Source I04
- Run processing on system 0 to make sure there are not artefacts relating to HDF5 in the analysis behaviour
- Extend #threads beyond #cores to see if I/O waiting is a problem

#2 wall clock


Pilatus 6M grid scan, 1209 frames
Wall clock time / s
100
10
1
10
#cores

#2 “efficiency”



#2 comments

- Overloading the system does not appear to be helpful i.e. #threads > #cores (for our system) though penalty is also not substantial
- Overall behaviour very similar => HDF5 structure is not really a problem
- For Pilatus 2 current performance of one 20 core node adequate (collection takes minimum of 50s, analysis 30s)

Conclusions

- “Efficiency” drops off rather quickly with increasing #cores [1]
- Wall clock time flattens off - around 40 s for system 0 using 20 cores; ~ 30 s for system 1 using 144 cores
- YMMV: I strongly recommend testing systems
- For small #frames start up time (~ 4s) dominates
- For large #frames wall clock time ~ linear 0.08 s / frame (10 cores)
- We maybe need to put some effort into optimizing DIALS for many core architectures (e.g. system 1 above; Xeon phi; …)
- Using small #cores but analysing each row of a grid scan on a separate node in a round-robin manner may be optimum for responsiveness

[1] this was a known thing at DLS for other tasks e.g. XDS hence our cluster systems having 2 sockets x 10 core Xeon