Data access and ATLAS job performance

Charles G Waldman
University of Chicago

OSG Storage Workshop, Sep 21-22 2010

Factors affecting job performance

- Algorithmic efficiency and code optimization
- VM footprint (swapping)
- I/O wait – data access (mostly inputs)

We can measure events/sec or CPU time/walltime. Here we're mostly using CPU/walltime

- 1 - Observe and advise
- 2 - Provision enough RAM, fight bloat
- 3 - Of great interest to storage community!

2 types of data access

- Stage-in
  - Files copied to /scratch and (usually) cleaned up after job completion
- Direct-access (and other names)
  - dcap, xroot, others (Hadoop, Lustre, other Posix)
- “Run across the bridge or walk across?”
  - If the bridge is sound, why not walk?
  - If it's not sound – let's fix it!

Stage-In

- Good if inputs are reused (pcache)
  - See http://www.mwt2.org/~cgw/talks/pcache
- Good if entire files are read mostly sequentially
- Allows for good control of timeout/retry behavior (lsm-get)
- Allows for checksum verification

Stage-In cont'd

- BUT:
  - Creates high I/O load on local disk (esp. ATLAS analysis jobs). File is first written to disk, read back for checksum, then read again for use by job... (could disable checksum)
  - Major performance degradations seen with 8 cores / 1 spindle (will only get worse with hyperthreading)
  - Do we equip all worker nodes with RAID0, or ...

Direct-Access

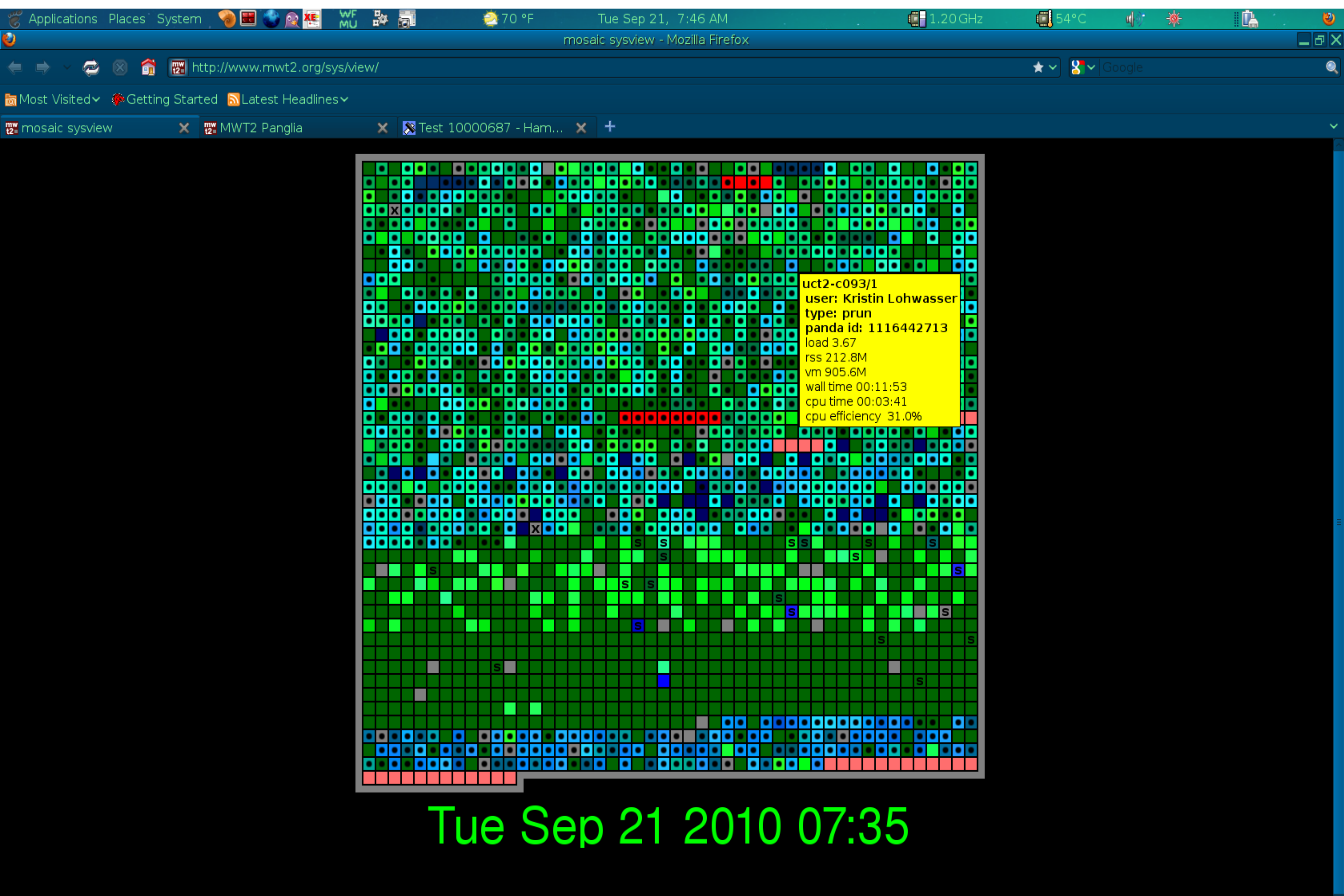
- Concentrates investment in high-performance storage hardware (e.g. Dell MD1000s)
- Good for jobs with sparse data access patterns, or files which are not expected to be reused
- In use at SLAC (xroot)
- Currently testing at MWT2/AGLT2 (dCache)
- Same amount of data (or less!) moved, but latency is a consideration since job is waiting

MWT2 tests

- Stage-in (lsm-get/pcache) for production, direct-access for analysis
- dCache tests using ANALY_MWT2
  - pcache for non-root files (DBRelease / *lib.tgz)
- xrd tests on ANALY_MWT2_X
  - pcache not currently enabled
- Some IU nodes in UC queue, for non-local I/O testing

Monitoring

- Hammercloud link
- effcy.py link
- SysView link
  - new feature - local SQL db



uct2-c093/1
user: Kristin Lohwasser
type: prun
panda id: 1116442713
load 3.67
rss 212.8M
vm 905.6M
wall time 00:11:53
cpu time 00:03:41
cpu efficiency 31.0%

Tue Sep 21 2010 07:35

dCache-specific observations

- Movers must not queue at pools!
  - set max_active_movers to 1000
- Setting correct ioscheduler is crucial
  - cfq = total meltdown (throughput, not fairness!)
  - noop is best – let RAID controller handle it
- Hot pools must be avoided
  - spread datasets on arrival (space cost=0), and/or use p2p. “Manual” spreading so far not needed
  - HOTDISK files are replicated to multiple servers

dCache cont'd

- Many jobs hanging when direct-access was first enabled...
- dcap direct access is a less-tested code path
- Invalid inputs causing hangups due to brittleness in dcap protocol (buffer overflows, unintentional \n in file name)
- All job failures turned out to be due to such issues (sframe, prun...)
- dcap library patch submitted to dcache.org

dCache read-ahead

- Readahead is key, esp. for non-local nodes
- DCACHE_RAHEAD=TRUE
- DCACHE_RA_BUFFER=32768
  - 32 kilobytes of read-ahead
  - These settings are common in ATLAS, may need to be studied
  - Too much readahead is clearly harmful
    - Relation of dcache readahead to blockdev readahead

dcap++ (LCB: Local Cache Buffer)

- Gunter Duckeck, Munich (link)
- 100 RAM buffers, 500 KB each
  - Hardcoded, needs to be tuneable
  - Sensitive to layout of ATLAS data files
  - Tuned for earlier release, 500KB is too big
- In use in .de cloud (and mwt2) w/ good results
- Awaiting upstream merge (6 months pending)

Xroot observations

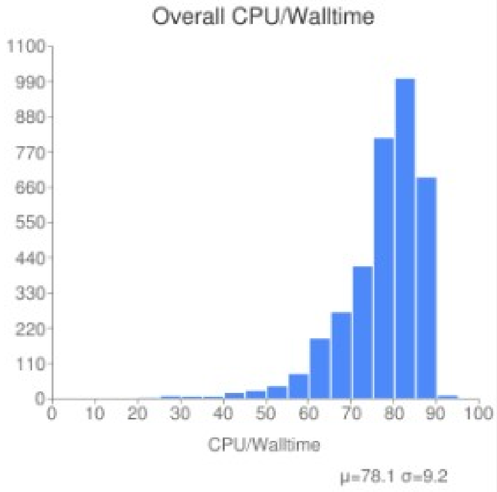
- Read-ahead in xroot is complex – subject of someone's PhD thesis
- Tuned for BaBAR?
- Working w/ Wei Yang and Andy H. to tune readahead for ATLAS needs

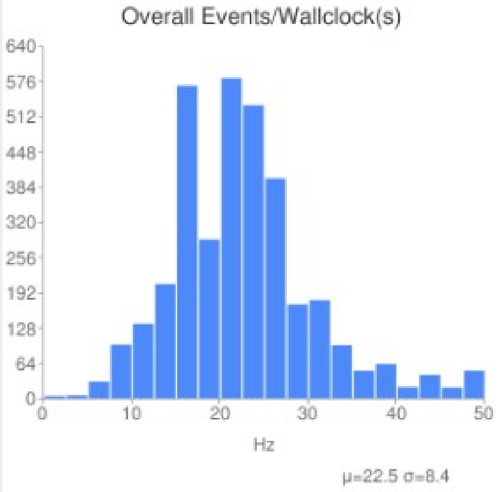
Read-ahead in general

- We need to make sure we don't optimize for one particular job at the expense of others (e.g. are we just tuning for Hammercloud?)
- Needs to be flexible so parameters can be tuned for different ATLAS releases or user jobs (advanced user may want to control these values themselves)
- No “one-size-fits-all” answer

Hammercloud plots

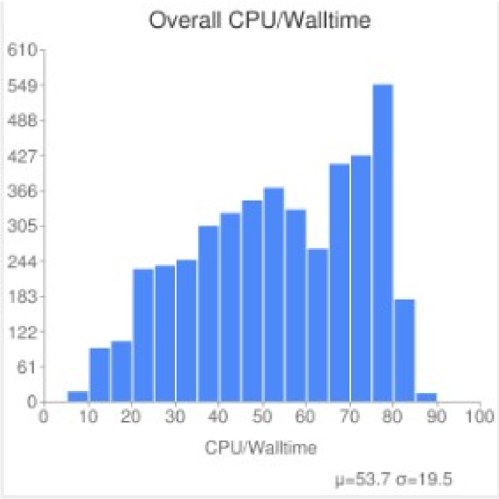
1000687, libdcap++, local nodes only

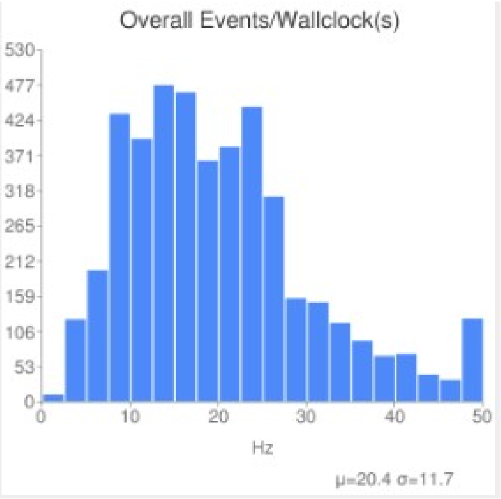




Hammercloud plots 2

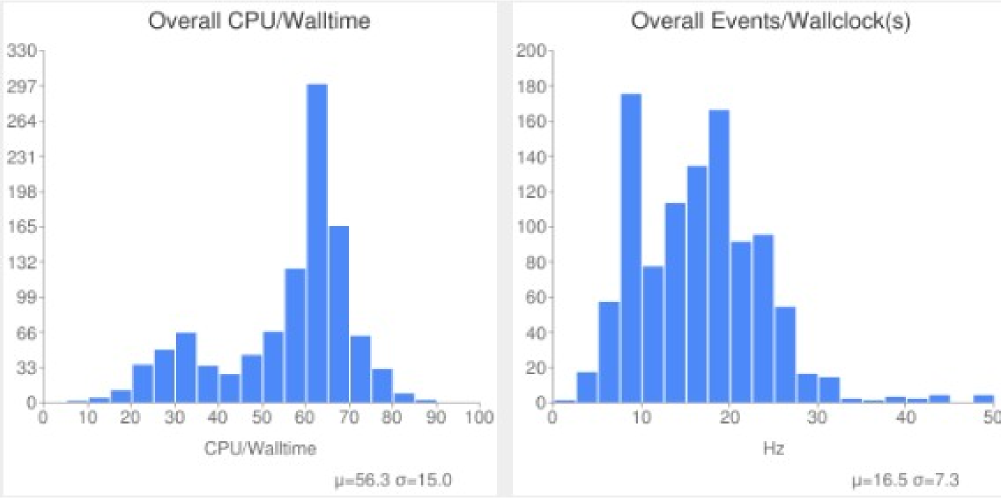
10001055 dcap++, local+remote nodes





Hammercloud plots 3

10000957: std. dcap, local+remote



Some results

- CPU/Walltime efficiency (rough #'s):

	Local I/O	Remote I/O
dcap	65%	~35%
dcap++	78%	~55%
xroot	78%	40%

References

stage-in vs direct-access studies