LIGO Computing model

Duncan Macleod
Chair, LSC Computing and Software WG

IGWN Computing model

Duncan Macleod
Chair, LSC Computing and Software WG

IGWN computing demand

IGWN computing demand has evolved ‘organically’ through Initial and Advanced LIGO and Virgo

Initial detectors (plus O1)

- principally HTC ‘offline’ workflows

Advanced detectors

- ‘traditional’ offline HTC CPU
- low-latency HTC CPU
- HTC GPU
- HPC CPU
- any of the above in large bursts

IGWN computing supply

Similarly, compute provision has evolved organically.

Traditionally:

- large, isolated HTC resources pledged to the LSC/Virgo
- providers would make their resources look like everyone else's
- dedicated hardware for specific needs

Now:

- large, isolated HTC pools
- massive, distributed HTC pool
- multiple prioritisation layers

IGWN Computing Model - Offline HTC (CPU+GPU)

For 2G observing, demand is still dominated by HTC workflows

Standard workflow model

1. data access/pre-processing
2. highly-parallelisable compute-intensive analysis
3. post-processing (final statistics, generating figures, HTML, etc) - typically needing *everything* from stage 2.

IGWN Grid infrastructure:

- powered by HTCondor
- multiple technically-independent, heterogeneous resource pools all talk to a central 'factory' that routes each job to any execute point (EP) based on its requirements
    - no local 'submit node'
    - no need for large, persistent storage
    - no *need* for local copies of data
- a few homogeneous Access Points (APs)
    - all users have access
    - large, persistent storage (web server, etc)
    - fast access to data

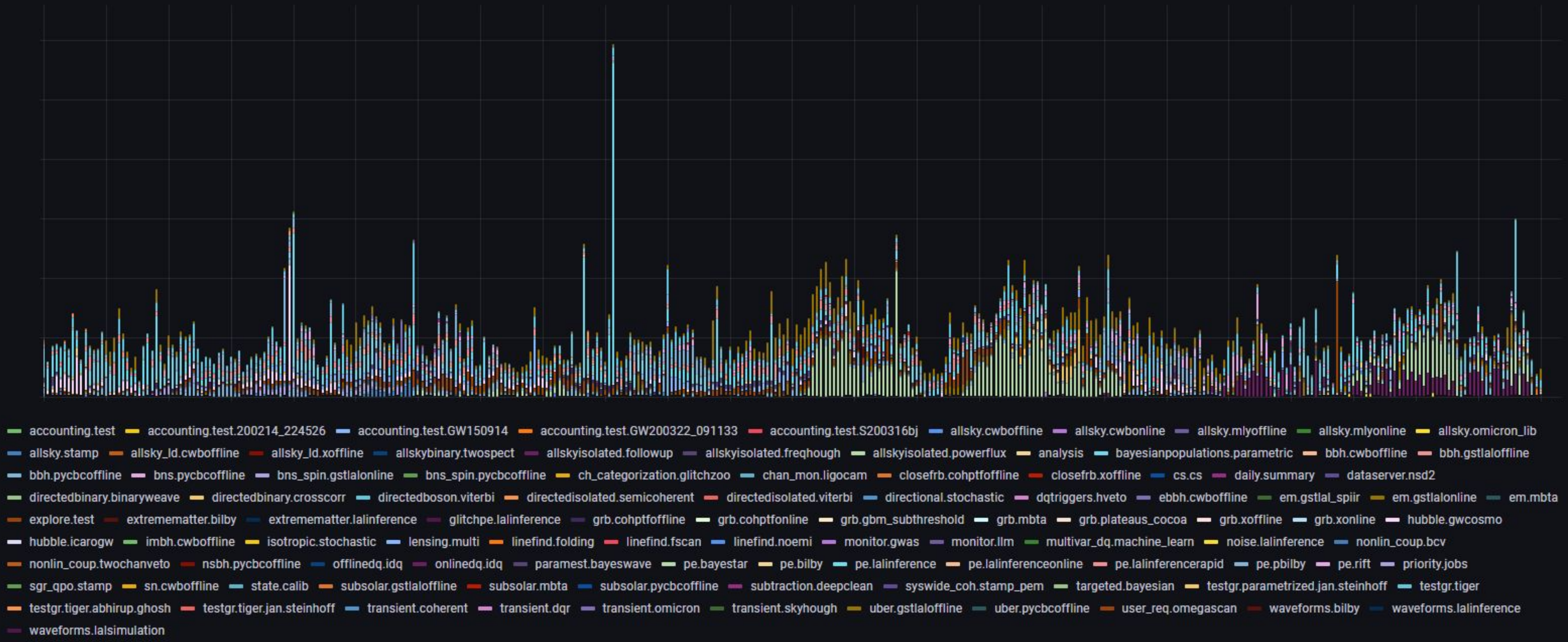
IGWN Computing Model - Offline HTC bursts

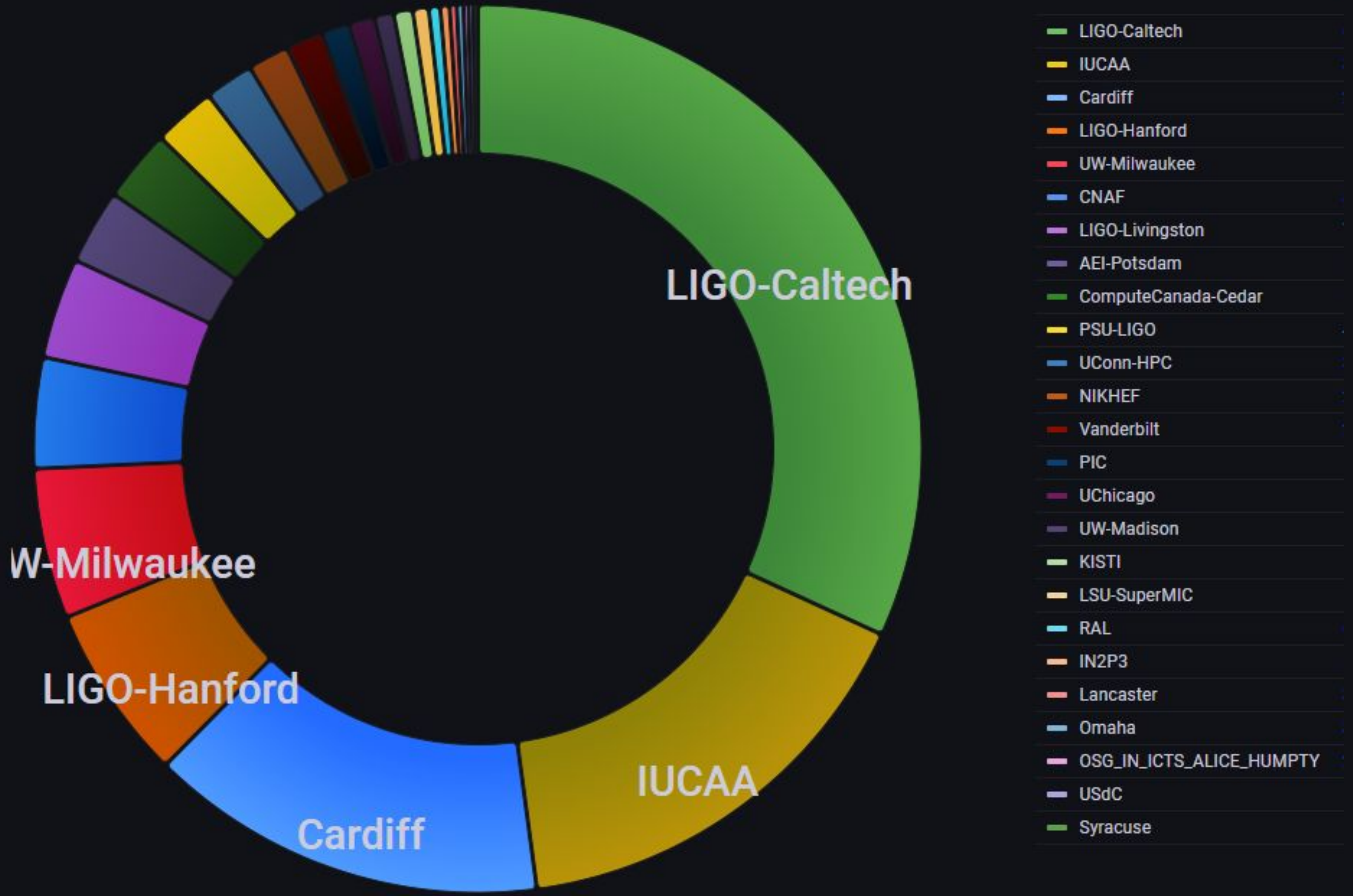
HTC demand is very unpredictable, mainly related to scientifically interesting signals in the data.

The distributed pool acts as a load balancer between the individual computing centres.

The distributed pool also includes opportunistic resources from a number of research institutions working in other (often related) fields. This relies on the generosity of like-minded individuals (so we should be prepared to return the favour).

Our total usage on opportunistic resources, averaged over time, is small but gives flexibility to request more resources than pledged for a short time.


accounting.test
accounting.test.200214_224526
accounting.test.GW150914
accounting.test.GW200322_091133
accounting.test.S200316bj
allsky.cwboffline
allsky.cwbonline
allsky.mlyoffline
allsky.mlyonline
allsky.omicron_lib
allsky.stamp
allsky_ld.cwboffline
allsky_ld.xoffline
allskybinary.twospect
allskyisolated.followup
allskyisolated.freqhough
allskyisolated.powerflux
analysis
bayesianpopulations.parametric
bbh.cwboffline
bbh.gstlaloffline
bbh.pycbcoffline
bns.pycbcoffline
bns_spin.gstlalonline
bns_spin.pycbcoffline
ch_categorization.glitchzoo
chan_mon.ligocam
closefrb.cohptfoffline
closefrb.xoffline
cs.cs
daily.summary
dataserver.nsd2
directedbinary.binaryweave
directedbinary.crosscorr
directedboson.viterbi
directedisolated.semicoherent
directedisolated.viterbi
directional.stochastic
dqtriggers.hveto
ebbh.cwboffline
em.gstlal_spiir
em.gstlalonline
em.mbta
explore.test
extremematter.bilby
extremematter.lalinference
glitchpe.lalinference
grb.cohptfoffline
grb.cohptfonline
grb.gbm_subthreshold
grb.mbta
grb.plateaus_cocoa
grb.xoffline
grb.xonline
hubble.gwcosmo
hubble.icarogw
imbh.cwboffline
isotropic.stochastic
lensing.multi
linefind.folding
linefind.fscan
linefind.noemi
monitor.gwas
monitor.llm
multivar_dq.machine_learn
noise.lalinference
nonlin_coup.bcv
nonlin_coup.twochanveto
nsbh.pycbcoffline
offlinedq.idq
onlinedq.idq
paramest.bayeswave
pe.bayestar
pe.bilby
pe.lalinference
pe.lalinferenceonline
pe.lalinferencerapid
pe.pbilby
pe.rift
priority.jobs
sgr_qpo.stamp
sn.cwboffline
state.calib
subsolar.gstlaloffline
subsolar.mbta
subsolar.pycbcoffline
subtraction.deepclean
syswide_coh.stamp_pem
targeted.bayesian
testgr.parametrized.jan.steinhoff
testgr.tiger
testgr.tiger.abhirup.ghosh
testgr.tiger.jan.steinhoff
transient.coherent
transient.dqr
transient.omicron
transient.skyhough
uber.gstlaloffline
uber.pycbcoffline
user_req.omegascan
waveforms.bilby
waveforms.lalinference
waveforms.lalsimulation



LIGO-Caltech
IUCAA
Cardiff
LIGO-Hanford
W-Milwaukee
LIGO-Caltech
IUCAA
Cardiff
LIGO-Hanford
UW-Milwaukee
CNAF
LIGO-Livingston
AEI-Potsdam
ComputeCanada-Cedar
PSU-LIGO
UConn-HPC
NIKHEF
Vanderbilt
PIC
UChicago
UW-Madison
KISTI
LSU-SuperMIC
RAL
IN2P3
Lancaster
Omaha
OSG_IN_ICTS_ALICE_HUMPTY
USdC
Syracuse

IGWN Computing Model - low-latency

Scientific motivation for as-fast-as-possible detection and publication of potential signal detections

Multiple stages of low-latency processing:

1. distribution of instrumental data
2. calibration and basic data quality analysis
3. distribution of calibrated data
4. signal detection and significance calculation
5. localisation and parameter estimation
6. publication (alerts)

Steps 1. and 2. use a small set of dedicated resources (absolute priority, no risk of competition) running system-level services

Step 3 (which supports 4. and 5.) uses an industry-standard data-streaming platform to distribute our custom data packets to a wide array of receivers

Steps 4. and 5. use the same HTC resource pool as offline workflows but with extremely high priority on the EP (other jobs keep running, but with limited access to system resources)

Step 6 is handled with dedicated resources (mix of on-premise, research cloud, commercial cloud)

IGWN Computing Model - HPC

We don't really have one.

HTCondor supports single-node, multi-core jobs extremely well, but…

- not that many high-core-count nodes
- hard and inefficient to defragment a pool to make space for very large jobs

HTCondor does support the 'parallel' universe, but it is not widely deployed or understood inside IGWN.

In the end users gain direct access to a 'native' HPC system at a partner institution and run directly on the relevant batch system there (commonly: slurm).

HPC demand is growing, so will likely need a 'real' solution for this for O5.

IGWN Computing Model - Data (and software)

The IGWN Grid system relies on systems to widely distributed data and software.

For data we use the Open Science Data Federation:

- data are published from local 'origins'
- end user requests a file on the EP
- the request is routed through the nearest cache which fetches the data from the origin (or a more remote cache) and returns it to the user, but caches it for the next request

CVMFS can provide a POSIX interface to the data that is more familiar to users

Desire to move to a more intuitive access model

- 'give me data for stream X from time A to time B'
- HTCondor figures out where to go, what files to read and returns just the data the user asked for

Software are distributed in a number of ways:

- user sends the software with the job (HTCondor manages the transfers)
- user runs the job in a self-managed container
- centrally-managed 'IGWN' software distributions are distributed using CVMFS

Summary

IGWN Computing Model has evolved over time

Still dominated by ‘traditional’ offline HTC workflows

HTCondor-enabled distributed grid platform simplifies connecting resources and managing load

Low-latency a special case

HPC using ad-hoc solutions