LArTPC simulation and data-processing with Theta

Patrick Green

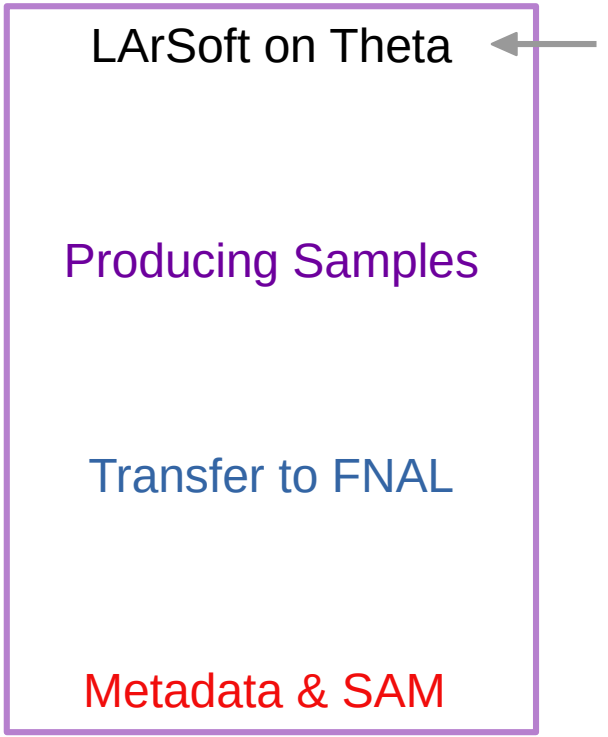
The University of Manchester

Theta

- Theta supercomputer @ Argonne Leadership Computing Facility (ALCF)
  - 4392 nodes, 64 cores per node
  - 281,088 jobs simultaneously
- LArTPC simulation and data processing is event based:
  - large scale processing easily parallelisable
  - in principle ideal for running on an architecture like Theta
- SBND and uBooNE have both been awarded multiple large allocations on Theta

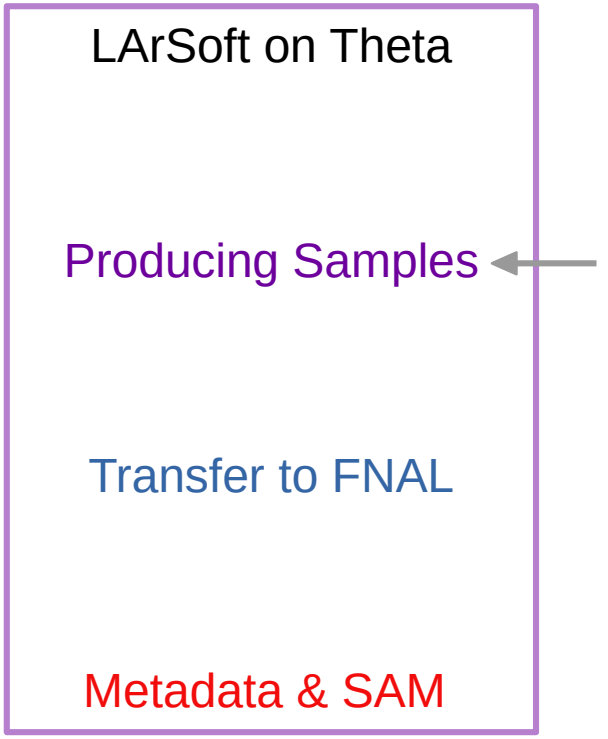
Running LArSoft

- Theta uses Intel Knights Landing Xeon Phi CPUs:
  - can run LArSoft out-of-the-box with minimal modification
  - however, performance could be improved e.g. by making use of AVX-512 vector processing
- LArSoft run using standard releases:
  - binaries copied to Theta via pullProducts
  - run using Singularity containers
  - some minor modifications required to avoid FNAL system specific code – e.g. ifdh, database access
- SBND simulation chain runs without issues on Theta:
  - BNB neutrinos, Corsika cosmics and particle gun events



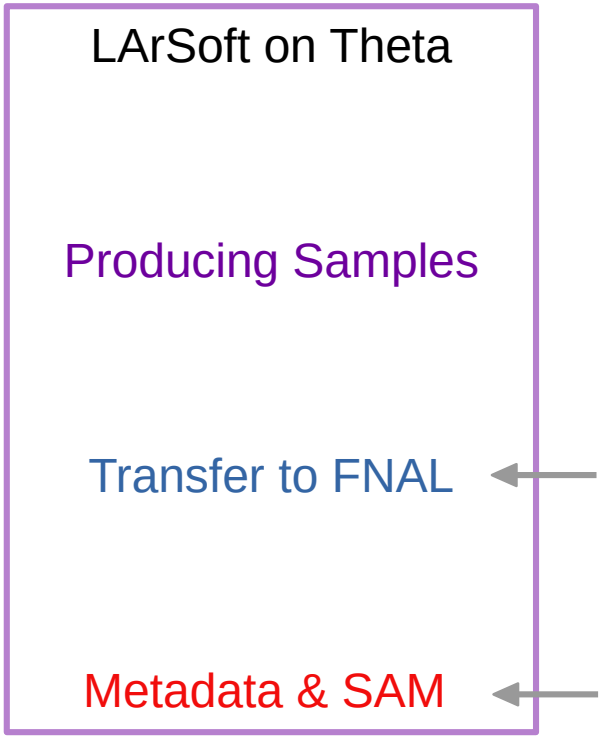
Producing samples

- Approach to simulation / data-processing:
  - sample is split into single (or small number) event jobs
  - each job processed on a separate core
  - output merged into files of N events
- Use Balsam to manage large scale production:
  - workflow management software for ALCF systems
  - fully automates splitting, running and merging of events



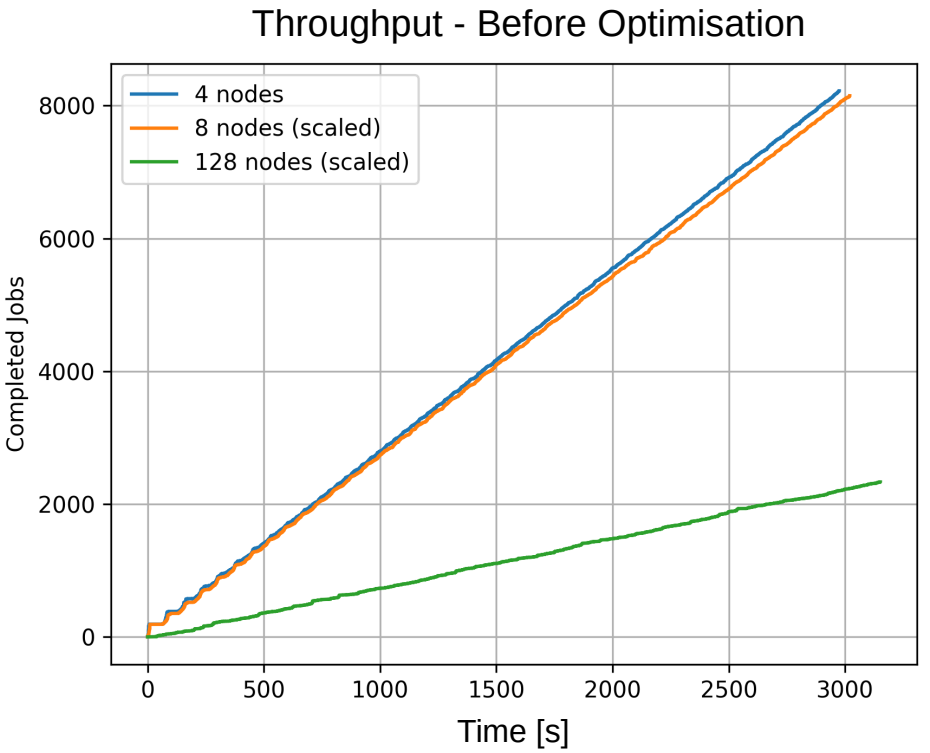
Transfer to Fermilab

- Data transfer between Theta and Fermilab using Globus:
  - achieved speeds of up to ~1.2 GB/s, may be able to increase
  - potential to automate file transfer using Balsam
- Data handling and transfer to tape:
  - POMs metadata extractor scripts have been modified to work with Theta generated files (SBND)
  - FTS plugin / dropbox automates extraction of metadata, declaration to SAM and transfer to tape-backed storage
  - Theta produced samples can then be treated identically to standard production samples by analyzers



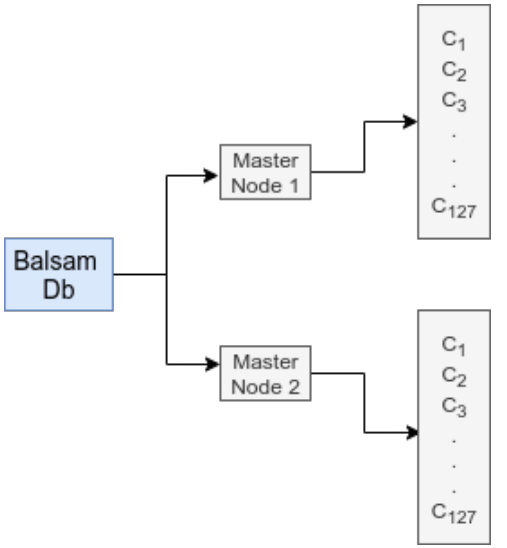
Attempting to scale up

- We have a functioning workflow, now need get running efficiently at large scale
- This is a significant challenge with LArSoft:
  - typical large scale application would internally utilise many cores / nodes
  - LArSoft is single-threaded – forced to run a separate instance on each individual core
- Leads to major bottlenecks / overheads:
  - ~95% efficient at 4 nodes (~250 cores)
  - ~30% efficient at 128 nodes (~8000 cores)



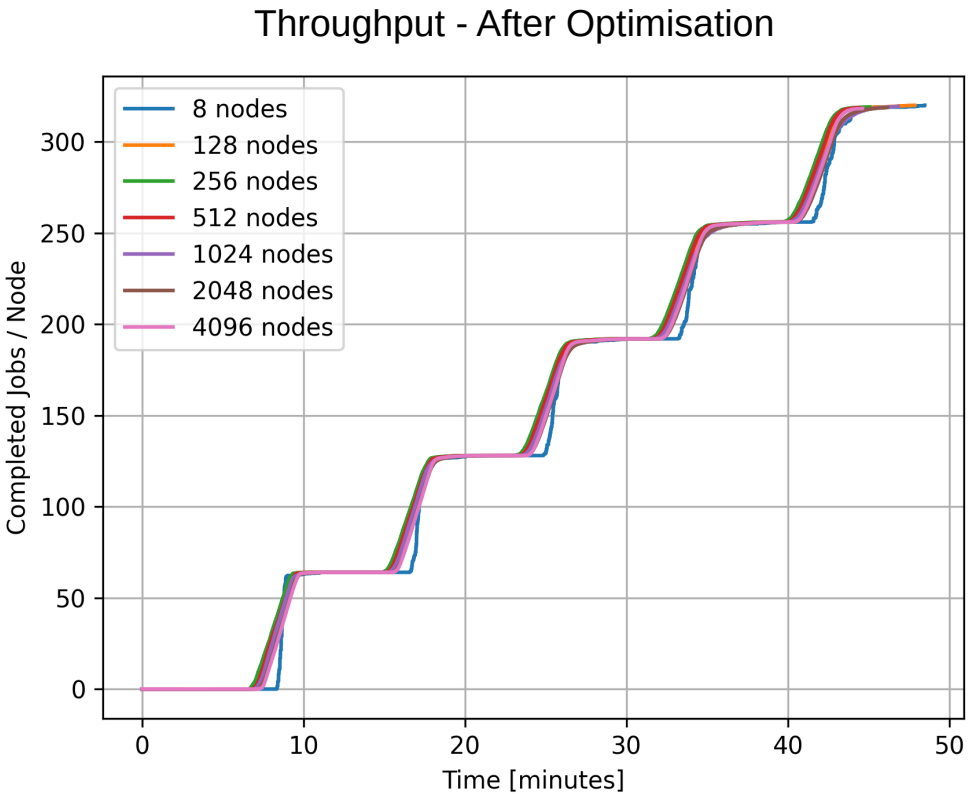
Resolving bottlenecks

- Significant bottlenecks identified in two main areas, both resulting from the need to run separate instances of LArSoft on each core
- Balsam MPI:
  - master process struggled to keep > 50% cores occupied at 128 nodes
  - resolved at small-mid scale via substantial optimisation of Balsam
  - larger scales then achieved by splitting into multiple master processes
- LArSoft I/O:
  - LArSoft binaries are read and outputs are written *per process,* easily overloading central Lustre file-system
  - mitigated by making use of local SSDs available to each node, and minimising copying to/from central file-system
- Further detail about these issues can be found in SBND DocDb 19928 (copied into backups)



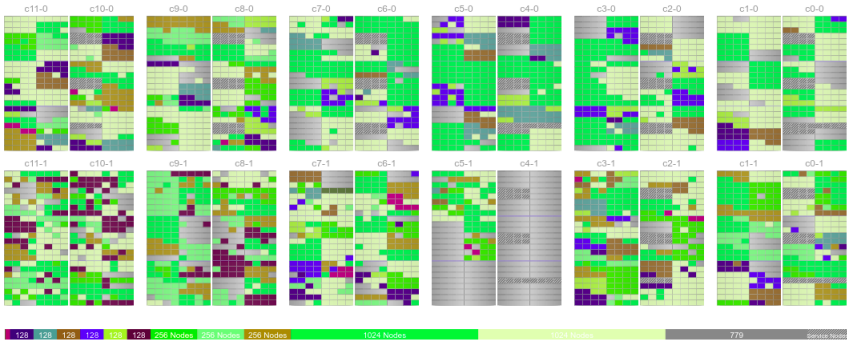
Running at large scale

- Scaling tested with using reproducible testing LArSoft application:
  - reconstruction of simulated Corsika cosmics in SBND (reco1 & reco2)
  - sbndcode v09_28_01_02
- Efficient throughput achieved at large scale:
  - ~95% efficient at 4096 nodes
  - 260,000 simultaneous jobs
  - 6.5M cosmics reconstructed in 1 hour
- A huge amount of work has gone into getting this working, with special thanks to Misha Salim and Corey Adams at ALCF



Test production sample for SBND

- Large sample of standard corsika cosmics (SBND):
  - 250,000 events, gen –> full reconstruction
  - sbndcode v09_09_00_01
- Bulk of sample run in single large scale job:
  - 1024 nodes, 3 hours
  - 201,653 events completed (~80% of sample)
- Make-up jobs run at 256 and 128 nodes
  - 249,790 events completed in total
  - ~60 TB total size
- Transferred to FNAL and saved to tape:
  - copying took ~14-16 hours @ ~1.2GB/s
  - FTS took ~4 days to process files + another ~3-4 days to finish transferring to tape



Total Running Jobs: 13

Job Id	Project	Nodes	Start Time	Run Time	Walltime
482891	neutrinoADSP	1024	9:54:31 AM	02:03:19	03:00:00
483226	SeismicHazard_2	1024	9:48:45 AM	02:09:05	02:50:00
483483	Bubble_Collapse	256	8:35:23 AM	03:22:26	06:00:00
483480	Bubble_Collapse	256	8:35:02 AM	03:22:47	06:00:00
480434	HHPMT_5	256	6:54:30 AM	05:03:20	06:00:00
483711	OptADDN	128	11:01:30 AM	00:56:20	01:10:00
483658	QMCPACK_aesp	128	10:51:02 AM	01:06:48	02:00:00
479186	QMCPACK_aesp	128	9:54:57 AM	02:02:53	03:00:00
483661	QMCPACK_aesp	128	10:50:30 AM	01:07:19	02:00:00
483659	QMCPACK_aesp	128	10:50:34 AM	01:07:15	02:00:00
483660	QMCPACK_aesp	128	10:50:58 AM	01:06:52	02:00:00
483656	APSDataAnalysis	25	8:35:28 AM	03:22:21	06:00:00
483714	CSC250STDM10	4	11:16:20 AM	00:41:30	01:00:00

Issues that remain

- Currently forced to run separate instances of LArSoft on each core:
  - inefficient – pay large initial cost in initialization of classes, etc. that can be significant fraction of total run time when generating smaller numbers of events
  - adds to stress on file-system, large number of instances all trying to write in uncoordinated way
  - more optimal scenario - one instance of LArSoft per node, that internally can run one event per core
- Size of output files challenging:
  - I/O requirement overwhelming, e.g. cosmic reconstruction jobs writing ~350TB per hour @ 4096 nodes
  - unable to save files during benchmarking, had to immediately delete due to space limitations
- File transfer to FNAL and saving to tape primary bottleneck for production:
  - this would need to be sped up substantially to keep up with pace of production on Theta
  - alternative – store files at ALCF (on Eagle) and create system for analyzers to access via globus

Summary/Conclusions

- We've achieved efficient running of LArSoft at large scale on Theta (~260,000+ cores)
- Demonstrated end-to-end production workflow for an SBND test sample:
  - generation and file processing automated, hence scalable
  - first time full scale production with LArSoft has been done on an HPC
- Theta could now be used to generate much larger sample(s) in future campaigns:
  - ideal for large computationally demanding samples, e.g. could be used to produce very high-stat cosmic samples that would be difficult / impossible on the grid
  - challenges remain in handling output files, sample transfer from ANL –> FNAL slow for large scales
- Documentation for running SBND production on Theta:
  - https://sbnsoftware.github.io/sbndcode_wiki/SBND_at_Theta

Back-up

IFDH seg-faults

- Significant fraction (~20%) of Corsika jobs were seg-faulting during calls to ifdh:
  - used select random corsikaDB file and to copy to job directory
  - seg-fault occurs during calls to calls ifdh::findMatchingFiles, which calls ifdh::lss, which calls ifdh::pick_proto_path, which calls ifdh::make_canonical then segfaults
- Bypassed this part of IFDH by only passing single CorsikaDB to job (so never calls ifdh::findMatchingFiles), but still segfaults later in code:
  - calls ifdh::fetchInput, which calls ifdh::cp, which calls ifdh::make_canonical, then segfaults
- Segfault seems to be occurring in ifdh::make_canonical, but exact reason was not identified. Was not able to reliably reproduce.
- Instead added option to CORSIKA_module.cc to pass direct path to file, allowing ifdh to be avoided entirely (similar to option in GENIE_module.cc):
  - https://github.com/LArSoft/larsim/pull/46 (merged Oct 2020)

Database access

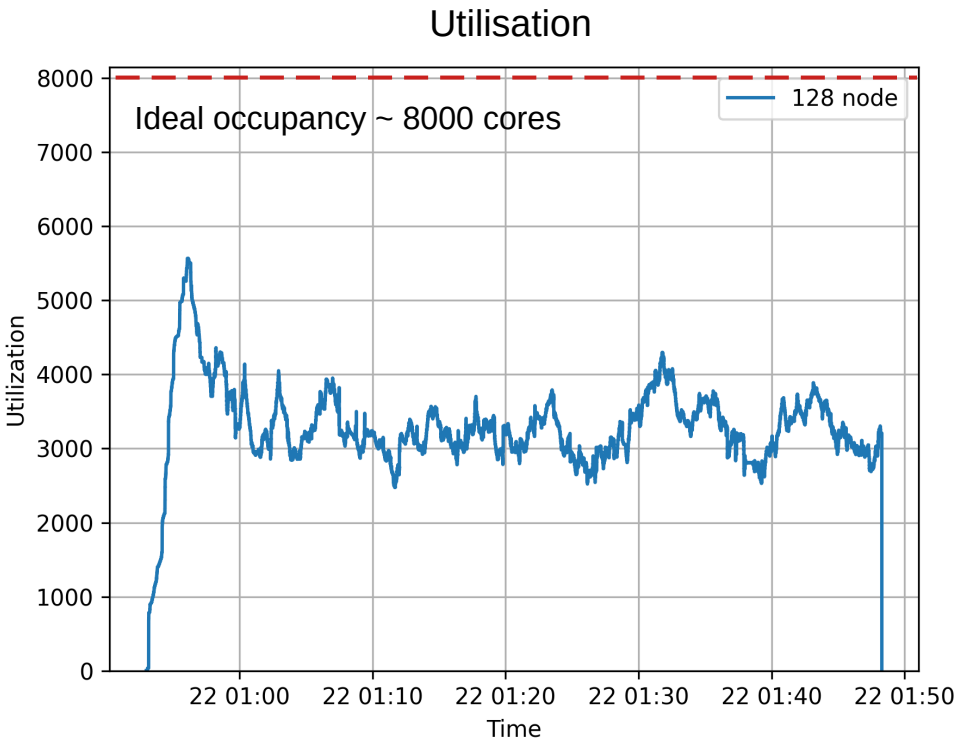
- Data reconstruction jobs require access to calibrations databases (MicroBooNE)
  - connects using DatabaseRetrievalAlg in larevt/CalibrationDBI/Providers/
  - uses DBFolder to create objects that store a dataset corresponding to an Interval of Validity
  - these use libwda to connect to a http server which caches data from the postgres DB so that the Postgres server is not swamped with connections.
- Once the webserver receives a query (url) it queries the DB for the relevant data set and serves it as a webpage
  - webpage is stored on the webserver so that future, identical, queries can download the existing webpage without querying the DB.
  - but, DB queries use a timestamp to calculate the relevant IOV. The timestamp is in ns, and therefore unique for each event.
- Running on Theta caused overwhelming number of database connections (>> number from grid jobs), especially at start of run where all jobs starting up at ~the same time

Database access

- We tried modifying the DBFolder and DatabaseRetrievalAlg classes to add options that enabled scaling down the number of DB connections:
  - scaling down by a constant factor
  - inserting a timestamp manually
- These didn't solve the problem for us, but could be useful for someone for future debugging
- What worked was using libwda to read a pre-downloaded .html file. Needed to use libwda v2_27_00
- Code exists in larevt branch: feature/andrzej_dburloverride.
  - was not merged at the time, probably needs updating (may be on redmine only?)

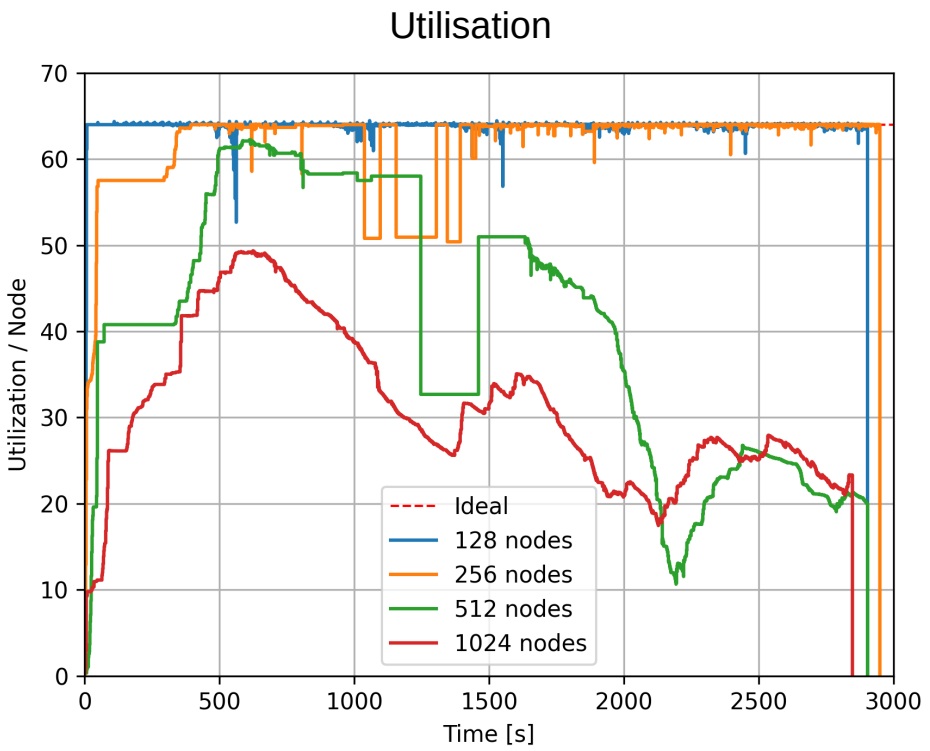
Improving efficiency: Balsam

- Significant bottlenecks were seen in Balsam:
  - struggled to keep > ~50% of cores occupied
- Each job has to communicate with the Balsam master process whenever changing state:
  - balsam now has to manage 1 job / core
  - designed to run far fewer e.g. 1 job / node
  - when it cannot keep up, jobs stuck waiting on master process
- To separate this from any unquantified bottlenecks in LArSoft, debugging was done using a toy matrix multiplication script



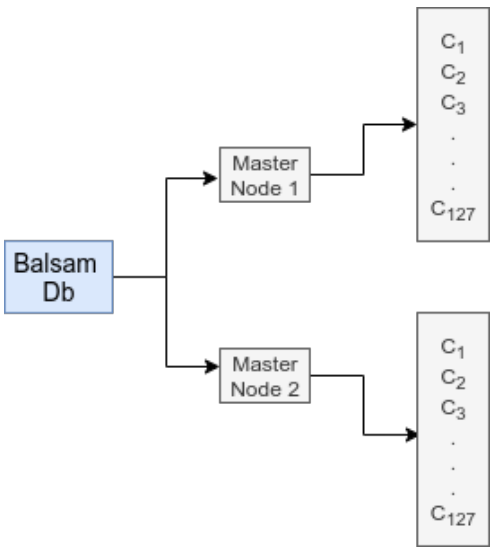
Improving efficiency: Balsam

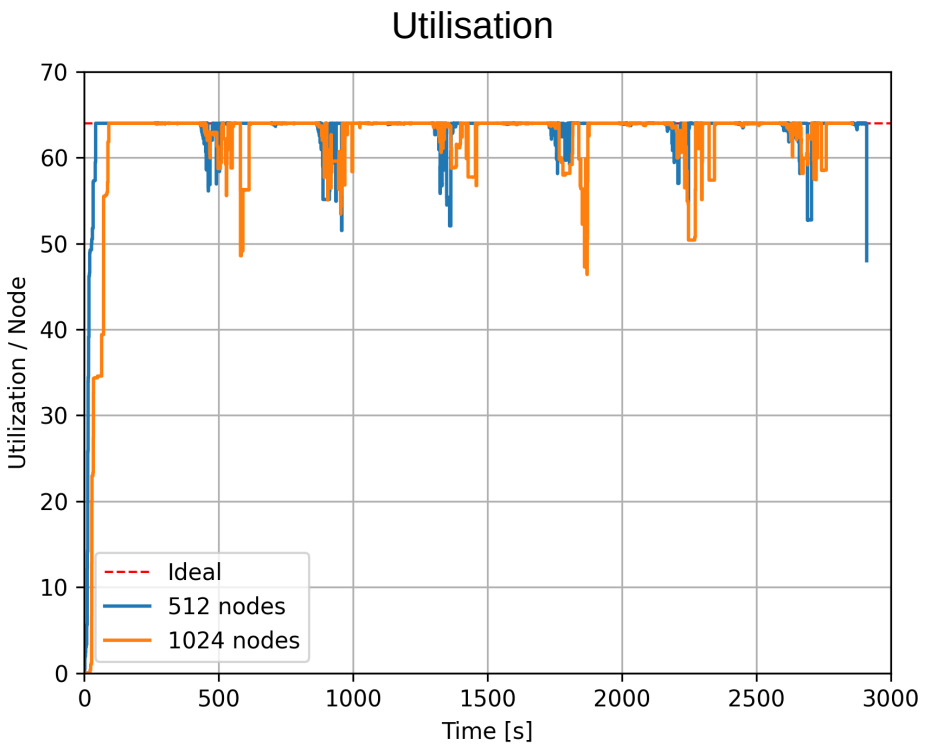
- Substantial improvements have been made to Balsam's efficiency at large scale:
  - fixed several bugs affecting running many processes per node
  - non-blocking messaging and job caching
  - ZeroMQ messaging instead of Cray-MPI
  - minimised logging and split into multiple files
- A huge amount of work has gone into this from Misha Salim and Corey Adams at ALCF
- These changes enabled near 100% utilisation at 128 nodes, but still struggles at larger scales



Improving efficiency: Balsam

- Balsam split into multiple master processes running on separate nodes:
  - master node manages 128 compute nodes
  - all master processes communicate with same central database of jobs





Minimising I/O bottleneck

- I/O bottleneck can be mitigated by making use of node local SSDs:
  - LArSoft binaries and job inputs copied to each node via MPI-I/O
  - LArSoft forced to run on the SSDs rather reading/writing to the Lustre file-system
  - minimal required final outputs copied to the central file-system
- Throughput at scale achieved with LArSoft:
  - ~95% efficient at 1024 nodes, sbndcode v09_09_00_01 (note this was old benchmarking run, see SL8 for newer)
  - larger scales should not be an issue, provided that I/O is carefully managed