justIN: recent additions and planned features

Andrew McNab
University of Manchester

DUNE Collaboration Meeting, May 2024

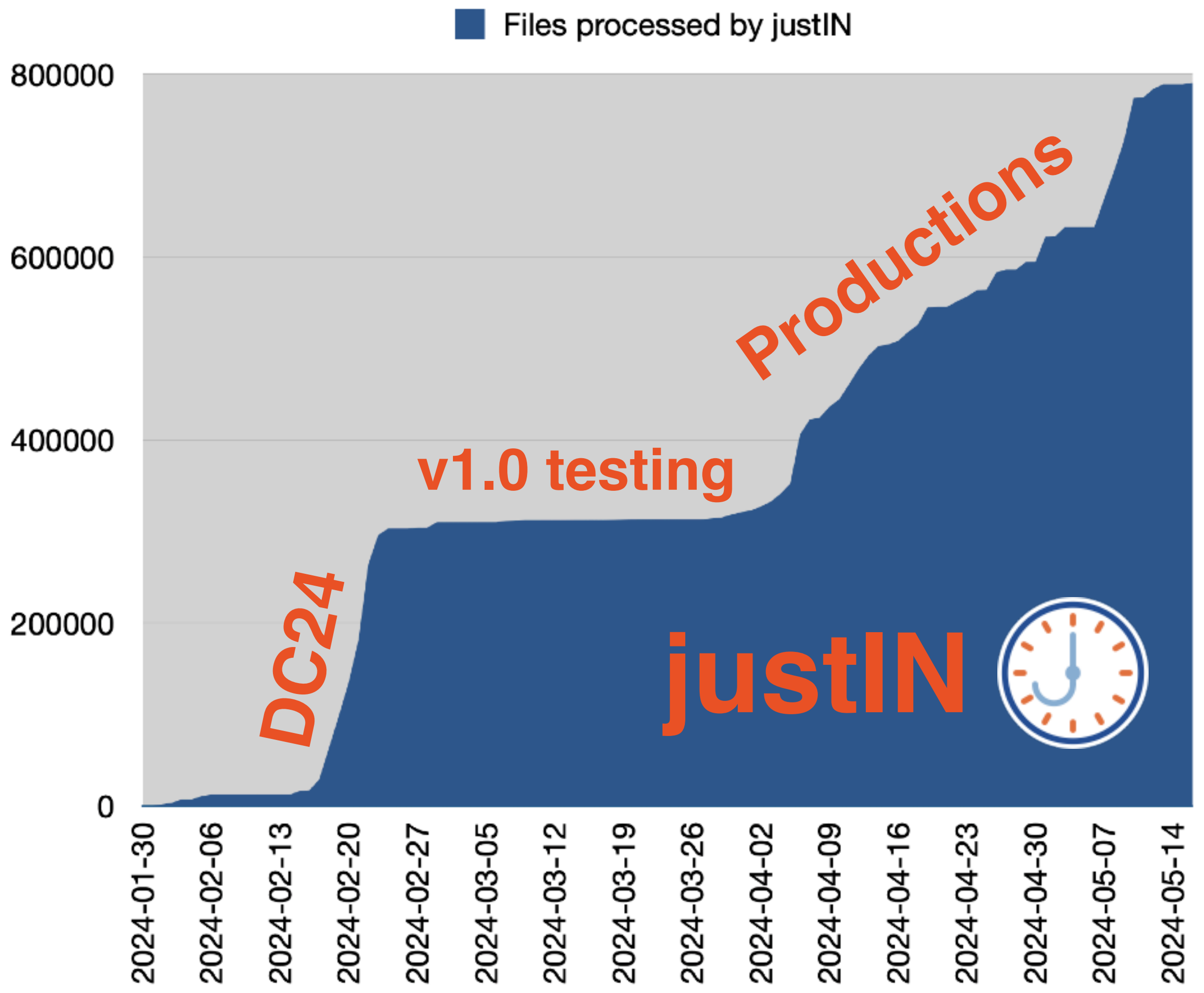
What has changed? What is next?

- Making justIN more resilient against user errors
- Greater use of GitHub to provide versioning of what users run
- Recording workflow versions and parameters in MetaCat datasets
- Working around Rucio limitations: rules and datasets
- Improving the dashboard as a workflow monitoring tool
- Allowing users to manage workflows entirely with the dashboard
- Recording state transition events and publishing them
- GPU support

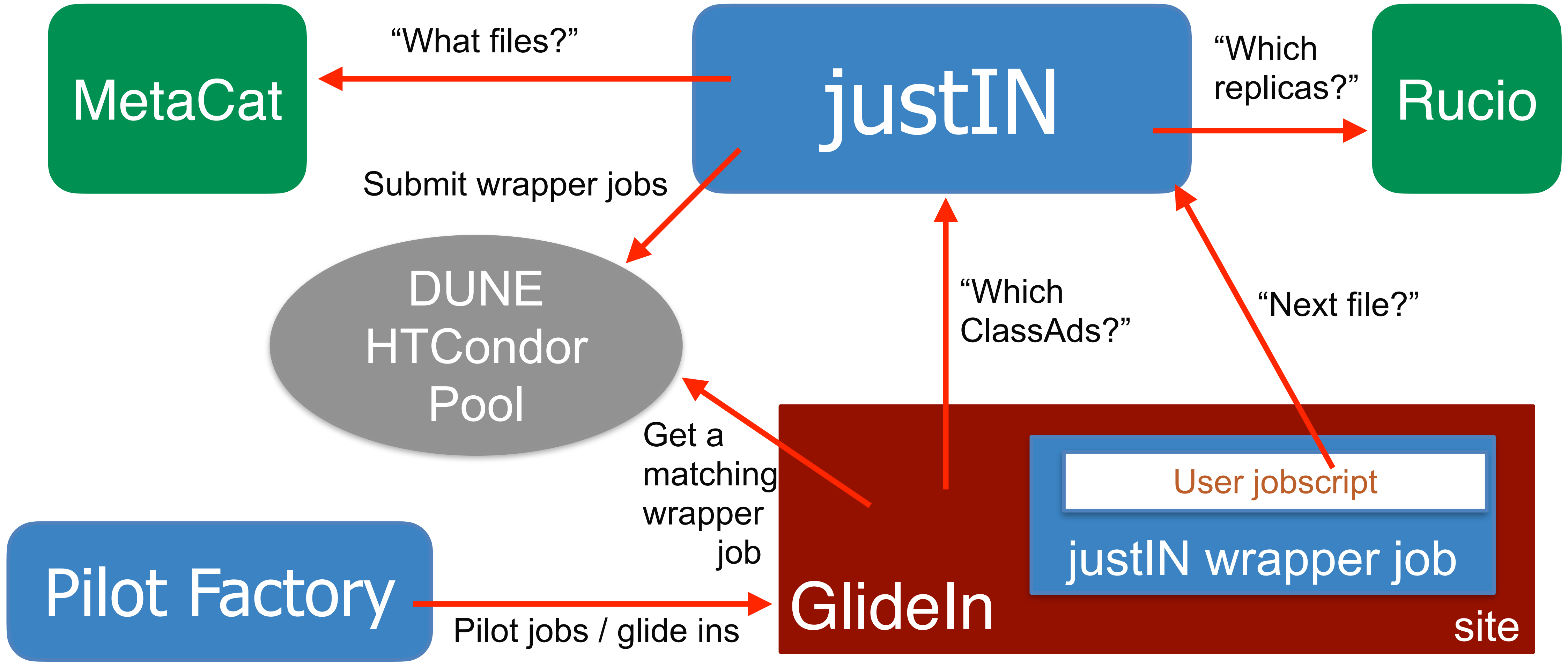
Current system

justIN quick overview

- justIN ties together MetaCat, Rucio, and GlideInWMS
  - It runs your jobscript on files you specify with an MQL metadata query
  - justIN directs your jobs to the optimal sites, and handles all the Rucio storage operations for you
- Workflows have been running since Data Challenge 22 in December 2022
- Since last collaboration meeting, 790k files have been processed with justIN
- v1.0 is the basis for official DUNE productions
- Any DUNE member can use it
  - Tutorial at https://justin.dune.hep.ac.uk/docs/



justIN



MetaCat
"What files?"
justIN
"Which replicas?"
Rucio
Submit wrapper jobs
DUNE HTCondor Pool
"Which ClassAds?"
"Next file?"
Get a matching wrapper job
User jobscript
justIN wrapper job
Pilot Factory
Pilot jobs / glide ins
GlideIn
site

Recent additions and planned features

Versions

- Everything is in GitHub, with branches and tags
- Documentation appears on the dashboard and is versioned with the code
- Alma 9 Docker containers built on the justIN VMs that host Docker
- Current production version for several weeks is 01.00 (“v1.0”)
- Version on integration instance is 01.01 - this is ready to be deployed for production
  - More testers welcome: https://justin.dune.hep.ac.uk/docs/integration_instance.md
- In these slides
  - “changed” = “in 01.01”
  - “available soon” = “in 01.02 (probably)”

Resilience against user error

- As the userbase broadens, we are seeing new classes of errors (and old friends) more often
- 01.01 has a checklist in the docs for people creating/editing jobscripts
  - Like “When your jobscript has ‘worked’, return 0. You can do this with `exit 0`”
- 01.01 also has a first attempt at automated identification of “problem workflows” which are put into the paused state by the system
- Once 10 jobs have “gone wrong” then
  - if a majority of jobscripts end in an error code, or
  - if a majority of jobscripts never asked for any input files, or
  - if a majority of jobscripts failed to process any input files despite being given some
  - then the workflow is paused
- These tests will be reviewed with experience of course!

Use of GitHub for versioning

- In 01.01, jobscripts can either be given as a local file or a GitHub reference
  - `--jobscript-git ORG/REPO/[…]/XXXXXX.jobscript:TAG`
- If given, this is recorded in the Workflow/Stage definition and eventually into MetaCat
- But, we also have user RCDS / justin-cvmfs-upload areas in cvmfs that jobs use
- So in 01.02, it will be possible to give one or more lines like `—git-repo ORG/REPO/TAG` at submission time
  - justIN then clones/checks out the nominated repo, makes a tar file, and puts it in RCDS
  - The Git commit hash is also captured and recorded in case the user is deleting/ recreating tags!
  - When the job runs, it will appear (via a symbolic link) at `$HOME/workspace/ORG/REPO`
  - The code is substantially written but needs a VM inside the Fermilab firewall

Workflow/stage info in MetaCat

- If justIN has to create a dataset for a set of output files that match a particular file pattern, then it has to create it in MetaCat as well as Rucio
    - Output files are put in the MetaCat dataset now as well as the Rucio one
- In 01.01, the MetaCat dataset is created with a dune.workflow dictionary
    - `dune.workflow {'user': 'amcnab@fnal.gov', 'stage_id': 1, 'rss_bytes': 2097152000, 'pattern_id': 1, 'processors': 1, 'workflow_id': 504, 'file_pattern': '*_reco_data_*.root', 'wall_seconds': 80000, 'jobscript_image': ‘/cvmfs/singularity.opensciencegrid.org/fermilab/fnal-wn-sl7:latest'}`
    - If you used —jobscript-git (or —git-repo in 01.02) that is recorded too
- So this makes it possible to discover exactly how a file was created, but looking at its parent dataset. Potentially years in the future.

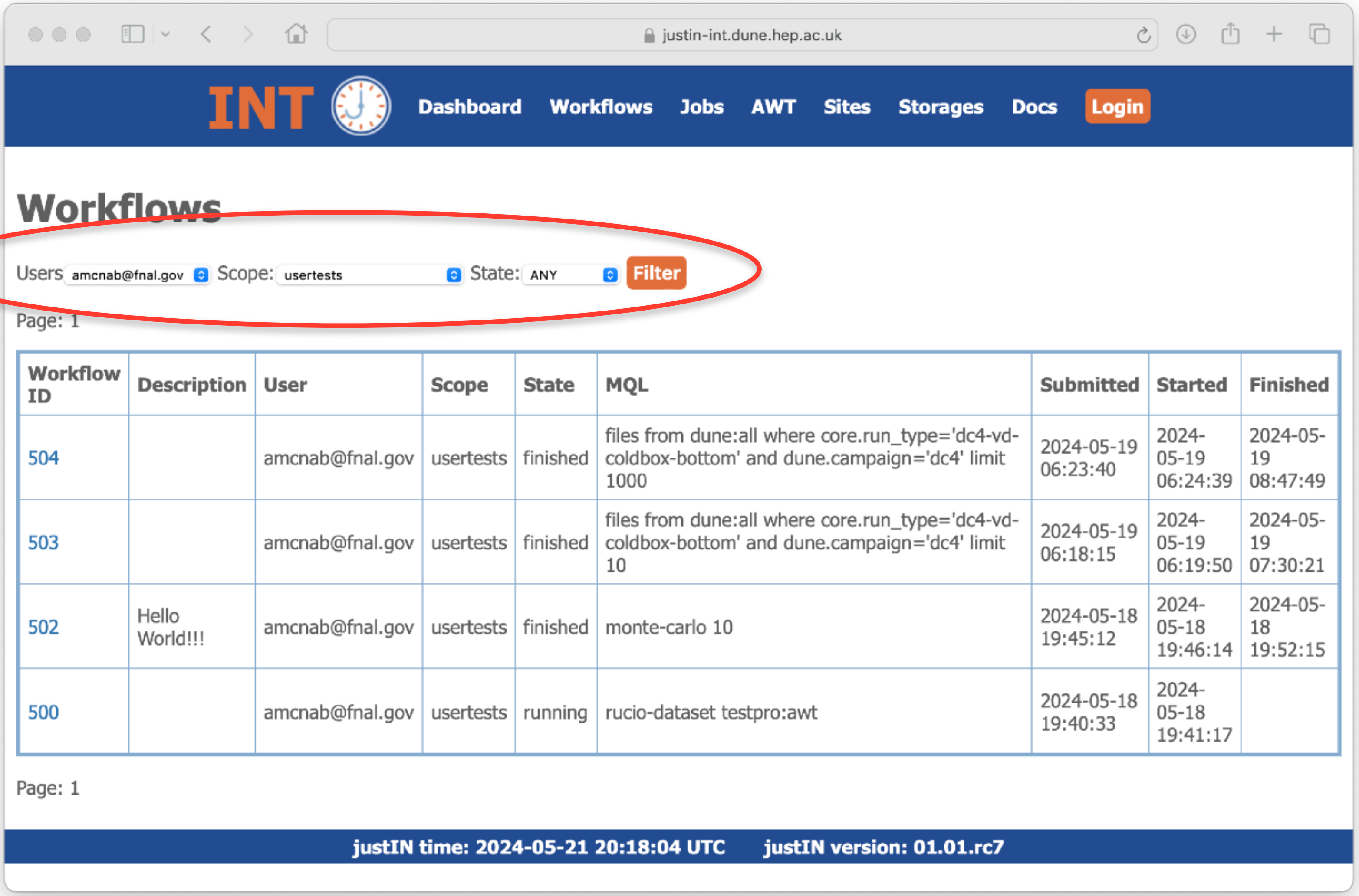
Rucio workarounds

- Data Management is concerned about the number of files associated with a particular Rucio rule and how this is currently scaling
- Workaround in 01.01 is that each output file uploaded to Rucio managed storage is put into a per-RSE dataset with a rule to stay on that RSE
  - Rule lifetime set to the —lifetime-days value given at submission time
  - For example w504s1p1-MANCHESTER for pattern 1 of stage 1 or workflow 504
  - These are not visible in the dashboard and users do not really need to know about them
- Similar approach is used for the justIN automatic logs.tgz files which are uploaded to Rucio managed storage
  - Every 1,000,000 seconds, justIN creates per-RSE datasets with lifetimes of 4,000,000 seconds and writes the logs.tgz file for each job there
  - They can be retrieved by `justin fetch-logs` (no X.509 needed), or with `justin-fetch-logs` or with `rucio get`

Dashboard improvements

- Since Feb 2023, users can log into the dashboard via CILogon / Fermilab SSO
- New in 01.01
  - Some ability to enable/disable sites and storages - really just for emergencies
  - More CSV and JSON downloads of lists of events, files etc
  - Filter forms on list pages for events and workflows
- In future versions
  - More lists with CSV and JSON downloads, and filter forms
  - Manage workflows via the dashboard
  - Create workflows via the dashboard, from scratch or by cloning others

Dashboard improvements



Workflows

Users amcnab@fnal.gov Scope: usertests State: ANY Filter

Page: 1

Workflow ID	Description	User	Scope	State	MQL	Submitted	Started	Finished
504		amcnab@fnal.gov	usertests	finished	files from dune:all where core.run_type='dc4-vd-coldbox-bottom' and dune.campaign='dc4' limit 1000	2024-05-19 06:23:40	2024-05-19 06:24:39	2024-05-19 08:47:49
503		amcnab@fnal.gov	usertests	finished	files from dune:all where core.run_type='dc4-vd-coldbox-bottom' and dune.campaign='dc4' limit 10	2024-05-19 06:18:15	2024-05-19 06:19:50	2024-05-19 07:30:21
502	Hello World!!!	amcnab@fnal.gov	usertests	finished	monte-carlo 10	2024-05-18 19:45:12	2024-05-18 19:46:14	2024-05-18 19:52:15
500		amcnab@fnal.gov	usertests	running	rucio-dataset testpro:awt	2024-05-18 19:40:33	2024-05-18 19:41:17	

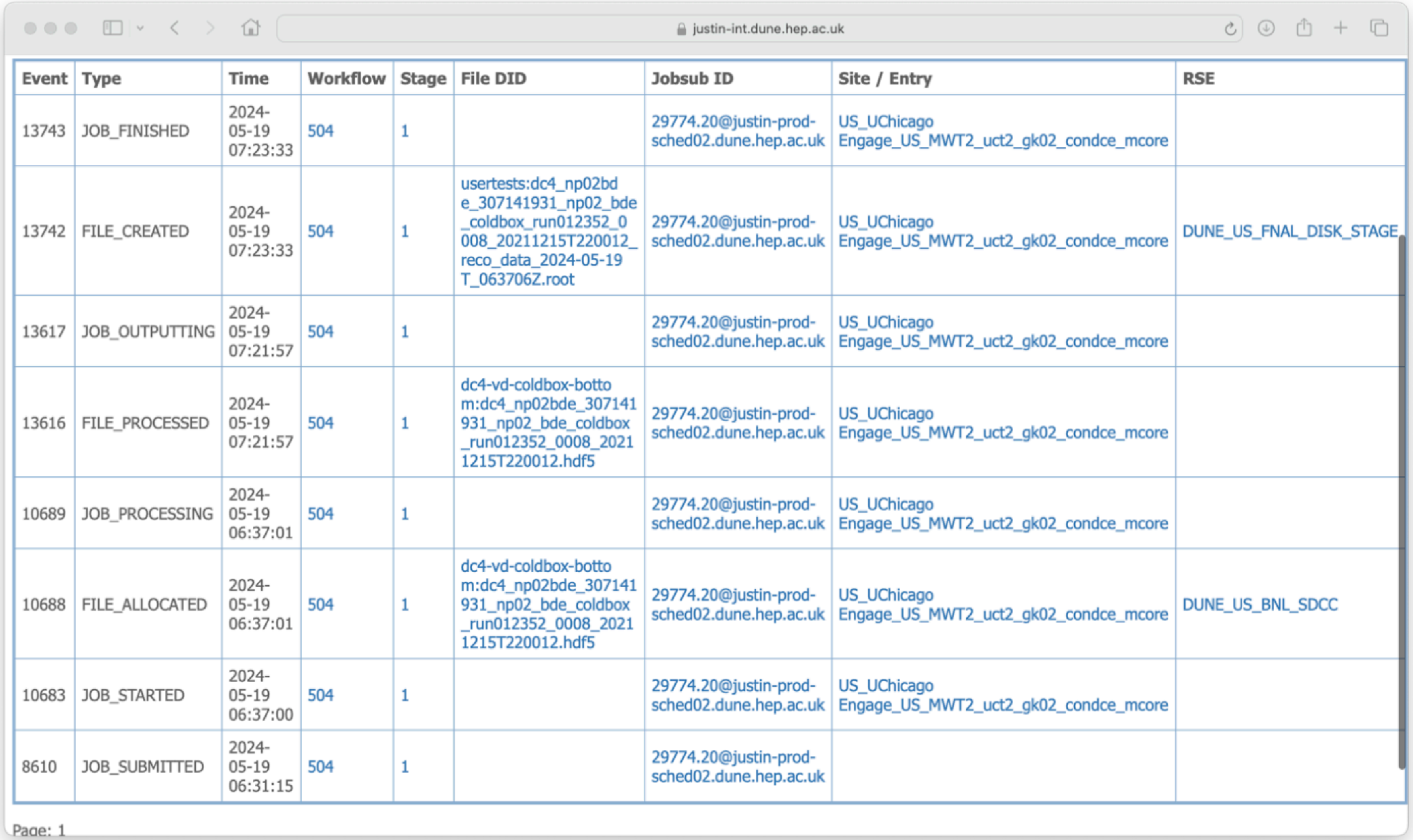
Page: 1

justIN time: 2024-05-21 20:18:04 UTC justIN version: 01.01.rc7

Improved event recording

- There was a major reworking of event types in 01.01
  - More types created
  - More of the existing types actually used!
  - Aim is to record all significant changes of state of workflows, stages, jobs, and files
- There is already a page on the dashboard to view events in a table, now with filters on workflow, site, job, input DID, RSE, event type etc
- See Wenlong's talk for the work by RAL-PPD and Edinburgh to funnel justIN events into ElasticSearch and view them
  - This will allow much more intuitive graphing of events to identify problems at sites and storages, and with particular workflows
- More of all this in 01.02 including recording significant changes made via dashboard and who did them

Improved event recording



justin-int.dune.hep.ac.uk

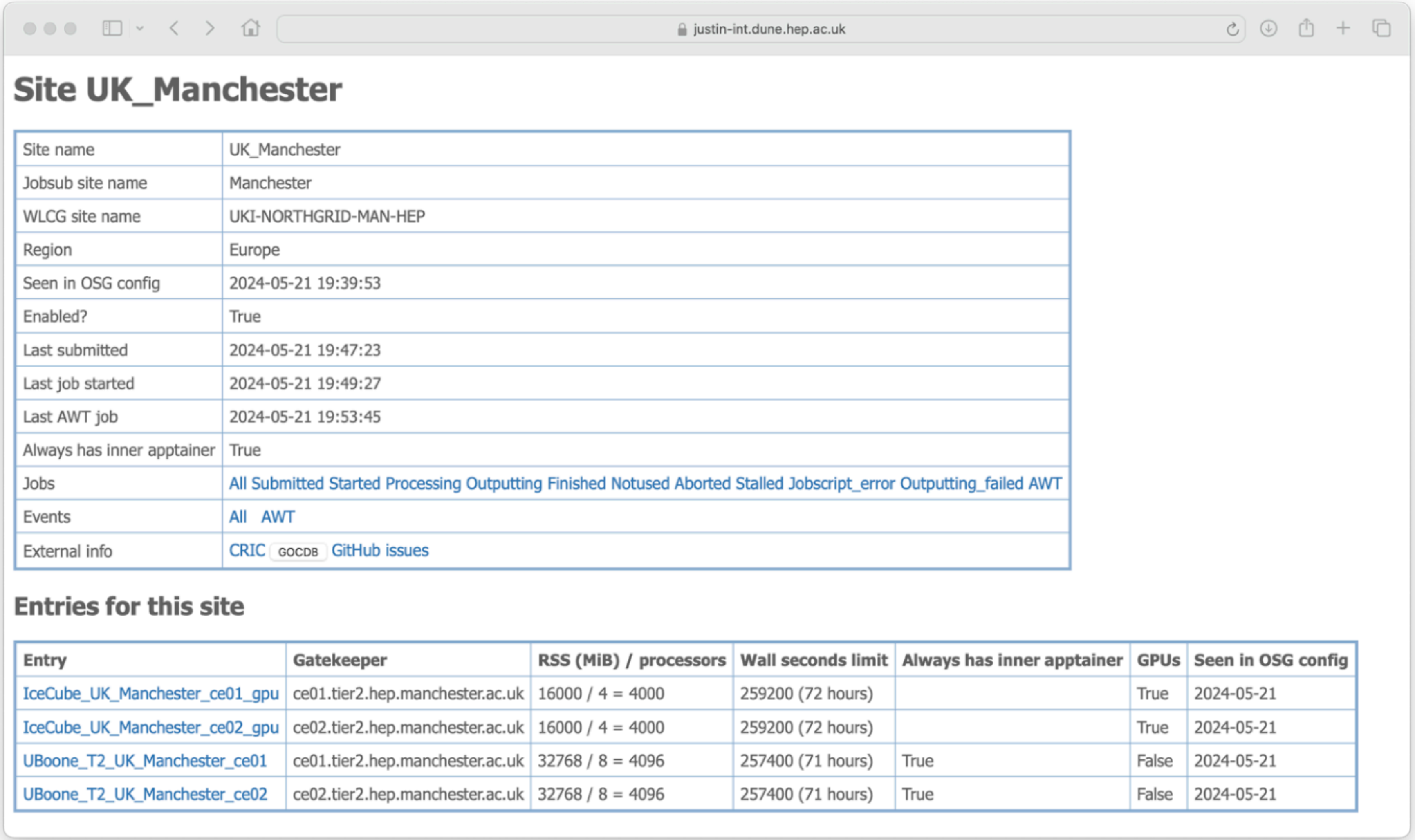
Event	Type	Time	Workflow	Stage	File DID	Jobsub ID	Site / Entry	RSE
13743	JOB_FINISHED	2024-05-19 07:23:33	504	1		29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	
13742	FILE_CREATED	2024-05-19 07:23:33	504	1	usertests:dc4_np02bde_307141931_np02_bde_coldbox_run012352_0008_20211215T220012_reco_data_2024-05-19T_063706Z.root	29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	DUNE_US_FNAL_DISK_STAGE
13617	JOB_OUTPUTTING	2024-05-19 07:21:57	504	1		29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	
13616	FILE_PROCESSED	2024-05-19 07:21:57	504	1	dc4-vd-coldbox-bottom:dc4_np02bde_307141931_np02_bde_coldbox_run012352_0008_20211215T220012.hdf5	29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	
10689	JOB_PROCESSING	2024-05-19 06:37:01	504	1		29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	
10688	FILE_ALLOCATED	2024-05-19 06:37:01	504	1	dc4-vd-coldbox-bottom:dc4_np02bde_307141931_np02_bde_coldbox_run012352_0008_20211215T220012.hdf5	29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	DUNE_US_BNL_SDCC
10683	JOB_STARTED	2024-05-19 06:37:00	504	1		29774.20@justin-prod-sched02.dune.hep.ac.uk	US_UChicago Engage_US_MWT2_uct2_gk02_condce_mcore	
8610	JOB_SUBMITTED	2024-05-19 06:31:15	504	1		29774.20@justin-prod-sched02.dune.hep.ac.uk		

Page: 1

GPU support

- Lots of debugging of the configuration was done to get GPU jobs working in the DUNE Global Pool - much appreciated
- justIN 01.01 gathers GPU-enabled entry info from OSG pilot factories along with non-GPU ones
- Next step for 01.02 is to allow GPUs to be requested by stages in workflows just as with the `—processors` option
- Try to use Apptainer's native support for "finding" CUDA libraries/modules which are there already?
- Just as with jobs requiring an inner Apptainer container, we can direct jobs to entries (ie CE endpoints) based on the jobs need or lack of need for GPUs
  - Should "just work" for advertised GPU sites (Manchester, Nebraska, QMUL) and for NERSC if accessed via HEPCloud?
- Significant requirement for GPUs from UK Pandora developers and others
  - Want to see how much can be addressed via justIN alongside dedicated GPU VMs for development being provided by the UK

GPU support



justin-int.dune.hep.ac.uk

Site UK_Manchester

Site name	UK_Manchester
Jobsub site name	Manchester
WLCG site name	UKI-NORTHGRID-MAN-HEP
Region	Europe
Seen in OSG config	2024-05-21 19:39:53
Enabled?	True
Last submitted	2024-05-21 19:47:23
Last job started	2024-05-21 19:49:27
Last AWT job	2024-05-21 19:53:45
Always has inner apptainer	True
Jobs	All Submitted Started Processing Outputting Finished Notused Aborted Stalled Jobscript_error Outputting_failed AWT
Events	All AWT
External info	CRIC GOCDB GitHub issues

Entries for this site

Entry	Gatekeeper	RSS (MiB) / processors	Wall seconds limit	Always has inner apptainer	GPUs	Seen in OSG config
IceCube_UK_Manchester_ce01_gpu	ce01.tier2.hep.manchester.ac.uk	16000 / 4 = 4000	259200 (72 hours)		True	2024-05-21
IceCube_UK_Manchester_ce02_gpu	ce02.tier2.hep.manchester.ac.uk	16000 / 4 = 4000	259200 (72 hours)		True	2024-05-21
UBoone_T2_UK_Manchester_ce01	ce01.tier2.hep.manchester.ac.uk	32768 / 8 = 4096	257400 (71 hours)	True	False	2024-05-21
UBoone_T2_UK_Manchester_ce02	ce02.tier2.hep.manchester.ac.uk	32768 / 8 = 4096	257400 (71 hours)	True	False	2024-05-21

Conclusion

- justIN use is ramping up for production and the user community is increasing
  - Self-service tutorial means interested people just appear and ask a question!
- A series of improvements have been made in 01.01 which should go live this week
  - Prompted by experience with 01.00, DC24, readiness tests, and productions
  - Mostly "fixing" things in the broad sense, including workarounds and making it easier to see what is going on
- For subsequent versions, starting with 01.02 more changes are in the pipeline
  - Making it easier to use the system with an emphasis on the web dashboard
  - Making it easier to reconstruct exactly how files were made, potentially years in the future

Backup

Just-in-time workflow system

- To satisfy our Rucio/MetaCat/LArSoft/GlideInWMS constraints we designed a just-in-time workflow management system: **justIN**
- Central idea: once we get a job slot at a site, we choose and run a workflow that needs “nearby” unprocessed files, and we feed it the details (URLs, Rucio DIDs, …) of nearby unprocessed files to work on.
- We have a central database that caches workflow requests from users, relevant Rucio file locations, and information about sites and storages
  - Agents update this information as necessary
- justIN job factory submits jobs to the DUNE HTCondor Global Pool
- justIN services respond to queries and tell the jobs what to do
- Web and command line interfaces allow submission/monitoring of workflows

justIN simple workflows

- In the simplest case, the user defines
  - A MetaCat Query Language expression specifying the inputs
    - `"files from dune:all where core.run_type='dc4-vd-coldbox-bottom' and dune.campaign='dc4' limit 10"`
  - Job requirements of memory, processors, duration and maximum network “distance” to storage with input files
  - Environment variables to be passed to the user’s jobscript
  - A list of preferred output storages to try
  - A jobscript to be run — in containers using the same Apptainer image at all sites
- Currently workflow requests are submitted with the `justin` command in cvmfs
  - Soon the justIN dashboard will allow the submission of workflow requests too

justIN multistage workflows

- Output files are always specified using filesystem wildcard patterns in the jobscript's working directory
  - You also specify the Rucio scope+dataset or a Fermilab scratch URL for the upload
  - The justIN wrapper job actually does the upload
- If you create a workflow request with multiple stages, you can also mark Rucio-stored output files as needed for the the next stage
  - These are then allocated to the jobs created for the second stage which use them as their inputs (and the third stage, and the fourth …)
  - Multiple input files can be allocated to the same job
    - This allows merging scenarios, or processing multiple files in the same file for efficiency

justIN jobscripts

- The model is that the user supplies a jobscript in Bash (maybe Python) to be run in the jobs of each stage of a workflow request
- The jobscript runs a script `$JUSTIN_PATH/justin-get-file` which returns the Rucio DID, suggested Rucio Storage Element, and suggested URL from which to get the file.
- The jobscript can run `justin-get-file` multiple times at the start or during the lifetime of the job — justIN does also support the SAM next file API that art can use
- The jobscript leaves it outputs, including metadata JSON, in the working directory
  - and specifies which input files were successfully processed
- The jobscript is run with reading-enabled proxies/tokens: all output should be done by the justIN wrapper job that ran the jobscript for the user
  - The wrapper jobs have access to higher level write-enabled credentials
  - They record what they do & flag up failures so justIN can manage retries in other jobs