Plans for data processing in fall 2019

Stefano Lacaprara, Marco Milesi

INFN Padova, Uni Melbourne

Outline

1. Current status of data processing
    a. Proc10 preparation and estimate
2. Run 2019b readiness and plan
3. Description of data processing tools

Status and plan for 2019a data processing

Stefano Lacaprara INFN Padova

Official Phase 3 data processing 2019a

- Current state of the art: **Proc9 + Prompt.**

Exp	Energy	Run range	Lumi good runs	Total good lumi	Lumi all runs	Campaign
3	4S	529-5613	**496.7±0.3±3.5/pb[1]**		-	Proc 9
7	4S	909-4120	**555.62 /pb**	**5.15 fb-1 (4S)** **6.01 fb-1 (total)**	642.8 /pb	Proc 9
8	4S	43-1022, 1036-1554	**1827 /pb**		1982.3 /pb	Proc 9
	Continuum	1703-1835	**39.02 /pb**		39.02 /pb	Proc 9
	Scan	1025-1031	**826.79 /pb**		827.0 /pb	Proc 9
	4S	1836-3123	**2764 /pb**		2973 /pb	Prompt

[1] arXiv:1910.05365

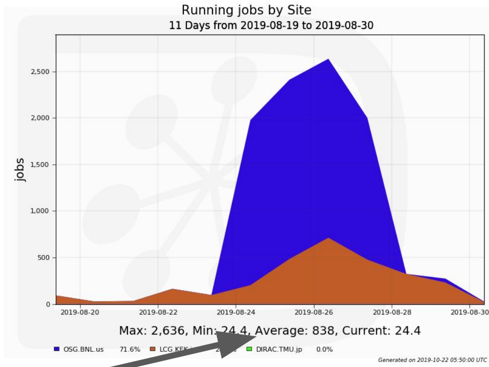
- Full processing available on GRID - HLT skims available at KEKCC
  - More info at: https://confluence.desy.de/display/BI/Phase+3+data
- Next iteration **Proc10** is underway
  - based on release-04 and will include only phase3 data (exp7 and 8) **not experiment 3**

Proc10 cdst processing

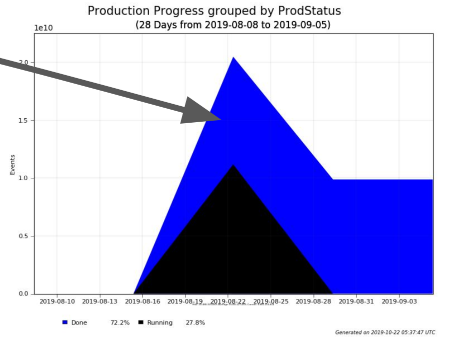
- Prepared json for cdst processing
  - Reminder: release4, only phase III data. Exp7+8
    - All runs (or only good ones? Maybe some can be magically recovered with new processing...)
  - https://agira.desy.de/browse/BIIDP-1928
- Waiting for calibration GT from calib group
  - [ERROR] Correction factor=0 is very small/too large! Resetting to 1.0. No updated payload in old GT.
- Time estimate for cdst.
  - Cdst processing for selected HLT skims (hadron / gamma-gamma / bhabha / mumu)
    - 11,000 hCPU / fb-1
    - With 400/1000/1500/2000 job slots: 1.2 / 0.5 / 0.3 / 0.23 day / fb-1
      - we have 400 on **b2_prod**, more on **l**, but shared and w/o priority. Can this be improved?
      - Eg dedicated resources plus additional ones shared but with priority?
        - When airflow will start using b2_prod as well we might have problem
    - L=6.4 /fb (~6.0 /fb good runs)
      - Only b2_prod: 8 days
      - **With 1000 jobs, about 3 days (+1 contingency ?)**

Proc10 mdst production

- Once cdst are done, second round of calibration: 1 week?
  - So, mdst processing can start maybe around beginning of november
  - mdst processing: priorities:
      - 1) HLT_skim **(also cdst output?)**
      - 2) all events
      - 1a) In parallel to 1), all events on grid
    - 1) takes some time as cdst processing (only output change)
      - So 0.5 day/ (/fb) on 1000 job slots for HLT_skims: **3 days**
    - 2) about 12x events-> 30 days (1000 job slots)
      - **20 days** if 1500 job slots
    - 1a) on the grid: (bucket7 2.9/fb ~7 days)
      - **15 days**
  - On the same period at kekcc we will be processing 2019b data
    - Prompt HLT_skim and bucket8 (?) (and unofficial)
      - Plus run-dependent MC, calibration, etc, etc...



Bucket7 2.9 /fb



Plan for 2019b data processing

Readiness for Fall data taking: 2019b

- Release 4 tested on exp8 data and exp10 cosmic
  - Some issues found and fixed in patch release 04-00-01
- Processing schema as in spring data taking:
  - Prompt HLT skim as data available offline
  - Prompt processing as in exp7-8 (“buckets”)
    - Local calib/align -> Cdst processing for selected HLT skims -> full calib
    - Mdst processing (maybe cdst as well)
    - Good for detector and performances studies, not for publication!
      - Turnaround: week(s)
    - Compute luminosity (lumi group)
  - We can provide a fast unofficial “uncalibrated” (namely using old calibration) processing if needed
  - Official processing to be defined taking into account data-taking and conferences goal.
    - Official processing will be done on GRID as well as at KEKCC
  - List of runs good for physics will be provided.
    - Online luminosity available

Buckets for 2019b

- How often? Every 2/fb ?
  - If we plan to have 20 /fb in 7 weeks, it’s 3/fb per week, so a more than bucket per week.
  - **cdst** processing
    - With 1000 job slots, it is about **1.5 days** of processing, plus overhead.
    - With 400 job slots it is about **3.5 days**, very little contingency
    - If we need to process cdst more than once, the cpu resources are even more a bottleneck
  - **mdst** processing (do we want also cdst for HLT_skims? No time overhead, only space)
    - Same time if we run on HLT_skim only.
    - **12x** if we want to run on all events at kekcc
    - **1 week** for all event processing on the grid
- Offline Luminosity computed for every bucket
  - From HLT_skims (see later)
- Good run list ? Likely only for Proc11 (final reprocessing for physics) not for prompt

Intermezzo

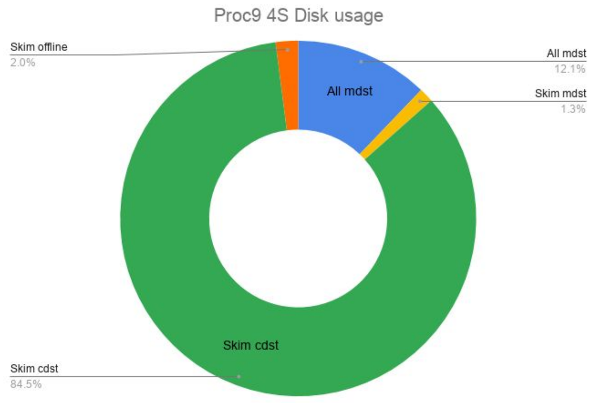
- Today **b2_prod** has 400 slots
  - Dedicated resources for official production
  - Limited access
    - Data processing, MC processing (including run dependent MC)
      - Offline skims
    - Calibration direct and via airflow
- We had it extended to 1500 during 2019a data taking
  - Very much appreciated help from kekcc computing team!
- We have used extensively general l queue when b2_prod was busy
  - Worked well, but same priorities as any other user (actually less, given the large usage)
- Can we have **b2_prod** extended for running period?
  - If not with dedicated resources, even shared ones but with higher priorities would help a lot.
- With automatic calibration (Airflow) the usage will likely increase.

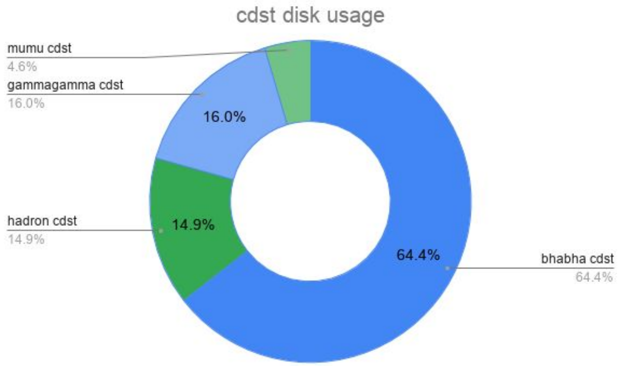
Plan for Winter conferences. **Proc11**

- Fully calibrated data processed by 18/1/2020. Ale's slides on monday.
- Counting backward: 20 /fb (?)
  - **Full re-calibration: 10 days** for 1 round of cdst processing
  - **mdst processing**: depends on what we process.
    - Hadron skim is about **1/10** of all hlt_skims.
      - Retention rate: Bhabha 6%, mumu_2trk 0.15%, gamma-gamma 1%, hadron 0.5%
    - So, **~1 day** to process all **mdst** for **hlt_hadron** Fast!
      - **Discussing with tau group if they can work on dedicated HLT_skim (hadron is not suitable for them)**
      - **Low mult HLT_skim for LeptonID ?**
    - Other 9 day to finish other HLT_skims
      - 15x to process all events at kekcc! (we won't)
    - Processing all events on the grid: **3 weeks**
      - Lumi group needed all events for optimal offline lumi computation, should be able to provide preliminary results starting from bhabha and digamma skims **TBC**
- +25% if we want to reprocess phase 3 2019a data (exp7+8)

Disk space

- The availability of `/dataprod` has solved the issue of staging from `/ghi`
  - Also the waiting time for staging is now almost null
- Data processing need about 10 TB/ (/fb)
  - Including cdst for HLT_skim and mdst for all events
  - Mostly cdst, of course.
- `/dataprod` has 1.5 PB of space
  - Currently we are using about 0.5 PB
  - Lot of stuff that can be likely deleted
    - Unofficial processing, various test
    - Old buckets (4,6)
    - Cdst used for calibration
    - …
  - Will do some cleanup after Proc10 (will be announced)
- **No issue of disk space for Proc10 and 2019b**





Items for discussion

- **cdst processing via airflow? Oh, yeah!**
  - As part of the calibration workflow
  - Maybe at BNL/desy as well? That would help a lot in term of available resources
- **Good runs on the grid: how?**
  - We have subdir at KEKCC, so you can process only the good ones
    - /group/belle2/dataprod/Data/release-03-02-02/DB00000654/proc9/e0007/4S/**GoodRuns**
  - Not on the grid!
    - Dataset contains all runs, so now it is up to the user to remove (a posteriori) the bad ones
- **HLT skim on the grid: how?**
  - Now we run all events on the grid, no HLT skim is available
    - Not a good way to encourage people to run on the grid!
  - At KEKCC: we run PromptSkim at quasi-online
    - We process the HLT raw skims
    - We process all events
  - On the GRID:
    - Full raw are copied to SE and then processed.
      - **a)** HLT skim full processing
      - **b)** copy to SE raw HLT skim (and process them)
      - **c)** run the HLT skim RAW->RAW on the grid (and process them)

Good/Bad runs

- Good/Bad runlist is provided by XXX
  - Now on a Jira ticket with subtickets for all detectors
    - Extensive use of Mirabelle
  - Result is a list of bad run (in a bad format pdf or spreadsheet)
  - Then, ~manually, we create GoodRun/BadRun subdirectories in the processing path, and symlink the runs
    - If you run on **Proc9/GoodRun** you are good
      - If you want you can look at bad runs also
    - If you run on **Proc9/** you need to select GoodRuns by hand
  - RunRegistry will vastly improve the jira ticket/spreadsheet approach, but we need to understand how to use it.
- Not on the grid!
  - Dataset contains all runs, so it is up to the user to remove (a posteriori) the bad ones
    - Interim: provide a decent list of good runs (easy)
- Eventually: RunRegistry. How to use it?
- For dp: should we process bad runs in proc10?
  - We have processed only good runs for phase3/exp3 since long time (including Proc9)

HLT_skim on grid: how?

- Now on grid we are processing all events, as we have raw for all events
- HLT_skim (eg hadron) are available at KEKCC only
  - First step of processing is prompt HLT_skimming of raw
    - Very fast processing, just I/O, independent from calibration.
- How to produce them on the grid?
  - Now CC copies to grid (BNL) raw for all events
    - If we can have also HLT_skim raw we could run just on them as we do at kekcc
    - Coordination with CC. We produce the HLT_skim, they copy the output to the grid.
  - Or:
    - We can produce hlt_skim from mdst.
      - Need to process all events, first, takes time.
  - Or:
    - We can hlt_skim from raw on the grid and them process the raw hlt_skim
- **Dreaming**: if HLT_skim would be produced at sroot->root step, they would be immediately available offline and can be copied to grid

Data Processing tools

Most likely the use cases for data processing are similar to other production (MC and skim), so maybe we can share some of the tools.

DP tools: some detail

- Git: ssh://git@stash.desy.de:7999/b2p/data.git **data/Proc**
  - **Documentation (incomplete) at:** https://confluence.desy.de/display/BI/Data+Processing
- Main idea: define the processing specific in just one configuration file (json)
  - Need **input run list** (typically symlink to that from CC group, or fraction in case of procXX)
  - **Produce "DB" (csv file)** with details of processing
    - We can do better than csv, but we did not want to introduce YAODB™ at least not before RunRegistry was in place

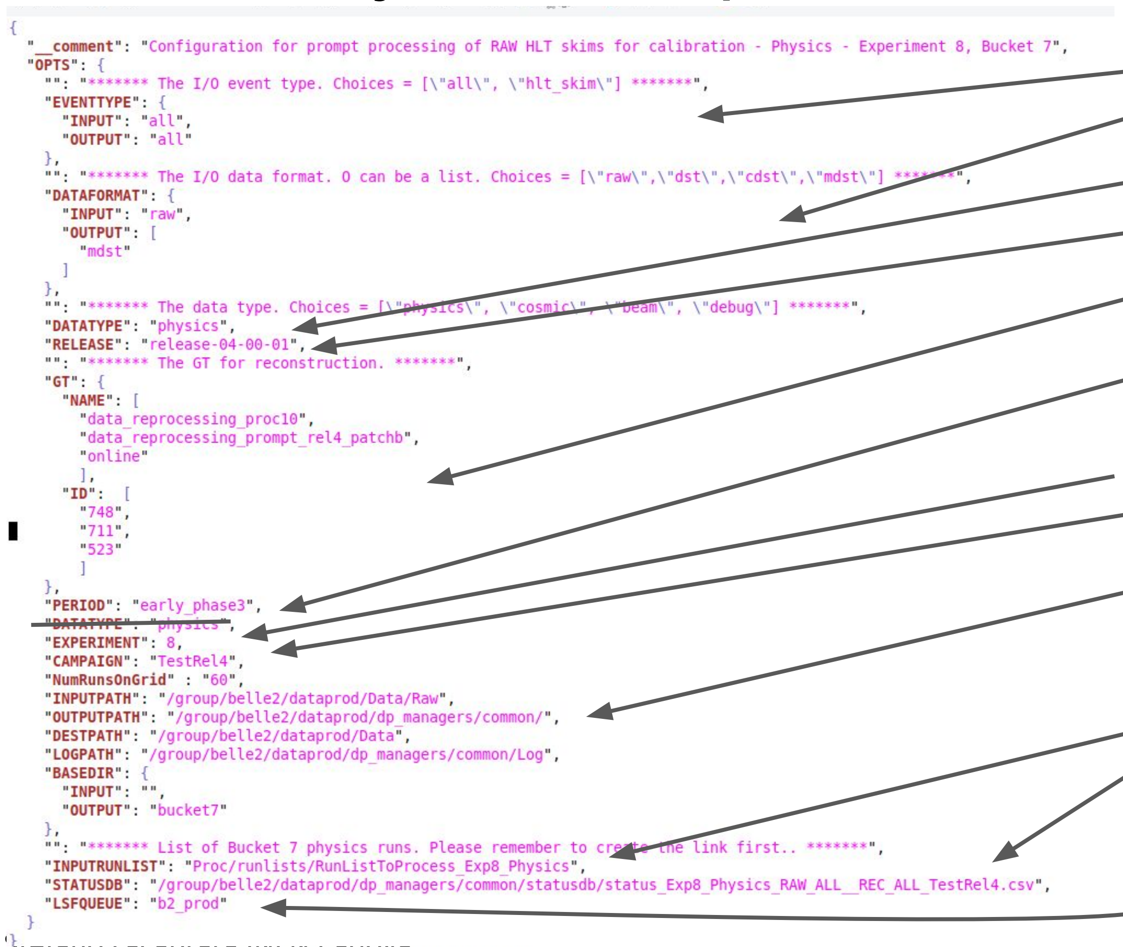
Yet An Other DataBase

  - All in python (and mostly pythonic)
  - Dedicated script to: submit (lsf), check, move
    - **submitRecAll.py** -c <json> --runs <...> …
    - **checkJobs.py** -c <json>
    - **move.py** -c <json>

Better names someday ...

  - Json is read and interpreted by a single python class, with many methods to provide what is needed for different stages
  - Actual basf2 script start from a **template one**, using parameters from json
    - Plus output and log destination, etc
- **Also grid production starts from same json**

DP tools json example



```
{
  "__comment": "Configuration for prompt processing of RAW HLT skims for calibration - Physics - Experiment 8, Bucket 7",
  "OPTS": {
    "": "******* The I/O event type. Choices = [\"all\", \"hlt_skim\"] *******",
    "EVENTTYPE": {
      "INPUT": "all",
      "OUTPUT": "all"
    },
    "": "******* The I/O data format. O can be a list. Choices = [\"raw\",\"dst\",\"cdst\",\"mdst\"] *******",
    "DATAFORMAT": {
      "INPUT": "raw",
      "OUTPUT": [
        "mdst"
      ]
    },
    "": "******* The data type. Choices = [\"physics\", \"cosmic\", \"beam\", \"debug\"] *******",
    "DATATYPE": "physics",
    "RELEASE": "release-04-00-01",
    "": "******* The GT for reconstruction. *******",
    "GT": {
      "NAME": [
        "data_reprocessing_proc10",
        "data_reprocessing_prompt_rel4_patchb",
        "online"
        ],
      "ID":  [
        "748",
        "711",
        "523"
        ]
    },
    "PERIOD": "early_phase3",
    ~~"DATATYPE": "physics",~~
    "EXPERIMENT": 8,
    "CAMPAIGN": "TestRel4",
    "NumRunsOnGrid" : "60",
    "INPUTPATH": "/group/belle2/dataprod/Data/Raw",
    "OUTPUTPATH": "/group/belle2/dataprod/dp_managers/common/",
    "DESTPATH": "/group/belle2/dataprod/Data",
    "LOGPATH": "/group/belle2/dataprod/dp_managers/common/Log",
    "BASEDIR": {
      "INPUT": "",
      "OUTPUT": "bucket7"
    },
    "": "******* List of Bucket 7 physics runs. Please remember to create the link first.. *******",
    "INPUTRUNLIST": "Proc/runlists/RunListToProcess_Exp8_Physics",
    "STATUSDB": "/group/belle2/dataprod/dp_managers/common/statusdb/status_Exp8_Physics_RAW_ALL__REC_ALL_TestRel4.csv",
    "LSFQUEUE": "b2_prod"
  }
}
```

- Type of input/output (define type of processing)
- Dataformat input (raw) output raw/m/c/dst.
  - Used by template basf2 processing
- Dataype (physics, cosmic, beam, …)
- Release
- GT
  - Multiple GT can be used
- Period
  - Needed by reconstruction process
- Experiment
- Campaing
  - #jobs to split grid processing
- Input/output/log path
  - Base path, full is build from release/exp/campaing/GT/
- Input run list
- DB (csv file)
- Lsf queue
  - Can be overridden by CLI

Example of actual job submission

```
basf2 <path>/rec/runRec.py -- -a REC
--data_type cosmic
-c <path>/json/config_Exp10Unofficial_Cosmic_RAW_ALL__REC_ALL.json
-i=<input_file> -o <output_path>
--filekey 00829.HLT5.f00003
```

- Templated basf2 script
- Options:
  - Action (REC|SKIM):
  - Data_type (cosmic, physics, beam, debug)
    - Together they define which python is used runRecPhysics, Cosmic, Skim, ...
  - Json parameters
  - input/output
  - Filekey (for DB/csv)
- Plus LSF specific parameters
  - `bsub -env 'all,~DISPLAY' -q b2_prod -oo <log> -eo <err> basf2 ...`

For grid submission

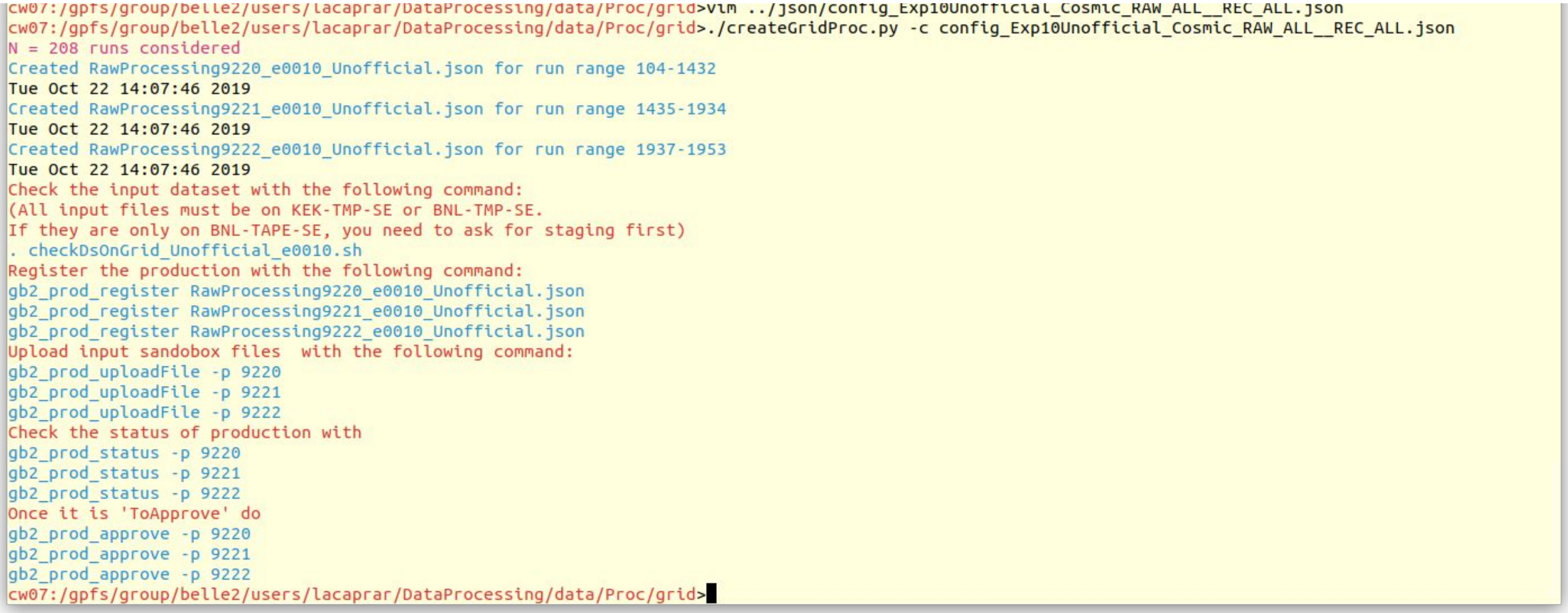
Possibly more interesting for MC production and skim production

```
./createGridProc.py -c <json>
```

- Check grid proxy and environment
- Check that release in json is available on grid
  - And match that in current env
- Check latest ProdID used on the grid
  - Create grid specific json from a template, using the relevant information from DP json
    - Name is **RawProcessing9220_e0010_Unofficial.json**
      - ProcId is setup automatically
  - After yesterday discussion with CC, we can prabbly skip adding ProdID to Json name
    - Search ProdID by passing ProdName name to `gb2_prod_summary -p prodName`
    - Not yet for `gb2_prod_status`

Example

- Produce script to check if the input files are available and staged on the grid
  - . checkDsOnGrid_Unofficial_e0010.sh
    - Cannot run internally: python2 vs python3
  - The prompt user for command to **register**, **uploadFile**, **approve**
    - Can be done by the script, but need to wait between register and upload, so better by hand.



```
cw07:/gpfs/group/belle2/users/lacaprar/DataProcessing/data/Proc/grid>vim ../json/config_Exp10Unofficial_Cosmic_RAW_ALL__REC_ALL.json
cw07:/gpfs/group/belle2/users/lacaprar/DataProcessing/data/Proc/grid>./createGridProc.py -c config_Exp10Unofficial_Cosmic_RAW_ALL__REC_ALL.json
N = 208 runs considered
Created RawProcessing9220_e0010_Unofficial.json for run range 104-1432
Tue Oct 22 14:07:46 2019
Created RawProcessing9221_e0010_Unofficial.json for run range 1435-1934
Tue Oct 22 14:07:46 2019
Created RawProcessing9222_e0010_Unofficial.json for run range 1937-1953
Tue Oct 22 14:07:46 2019
Check the input dataset with the following command:
(All input files must be on KEK-TMP-SE or BNL-TMP-SE.
If they are only on BNL-TAPE-SE, you need to ask for staging first)
. checkDsOnGrid_Unofficial_e0010.sh
Register the production with the following command:
gb2_prod_register RawProcessing9220_e0010_Unofficial.json
gb2_prod_register RawProcessing9221_e0010_Unofficial.json
gb2_prod_register RawProcessing9222_e0010_Unofficial.json
Upload input sandobox files  with the following command:
gb2_prod_uploadFile -p 9220
gb2_prod_uploadFile -p 9221
gb2_prod_uploadFile -p 9222
Check the status of production with
gb2_prod_status -p 9220
gb2_prod_status -p 9221
gb2_prod_status -p 9222
Once it is 'ToApprove' do
gb2_prod_approve -p 9220
gb2_prod_approve -p 9221
gb2_prod_approve -p 9222
cw07:/gpfs/group/belle2/users/lacaprar/DataProcessing/data/Proc/grid>
```

Summary

- Proc10 is currently underway, cdst is expected to start soon, mdst will follow after second round of calibration
  - Full exp7 + exp8 data, NO exp3 (phase2)
  - mdst for HLT_skims in about 2 weeks time scale
    - Full dataset (on grid only) +1.5 weeks
- Plan for 2019b runs:
  - Almost continuus prompt processing
  - Plan proposal to meet winter conferences
- Data processing tools
  - Can be shared also for other processing task?

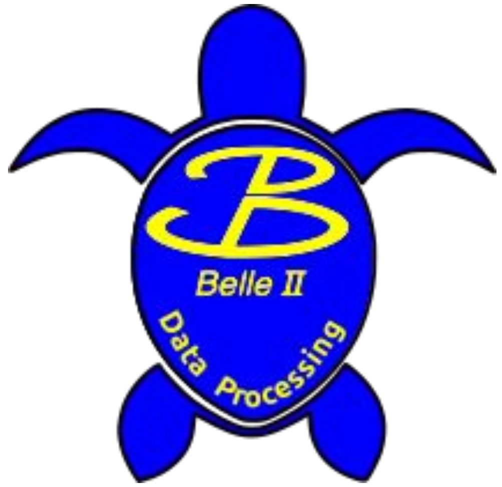
We need a Data Processing Logo!

Calibration

calibration

Software

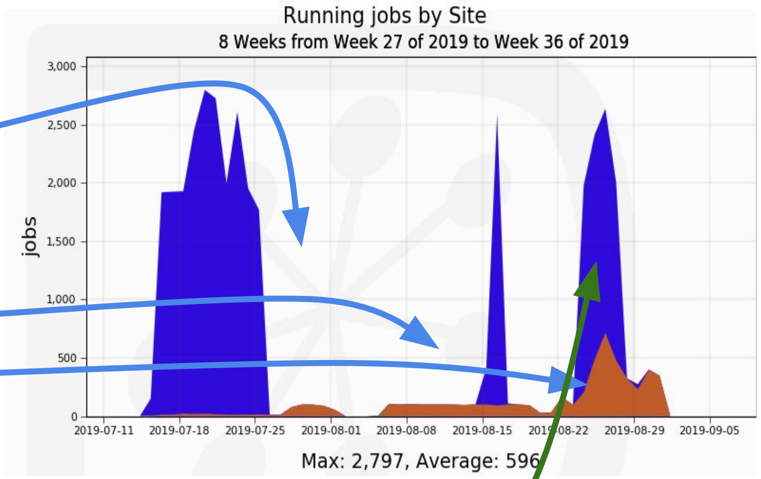
Data processing



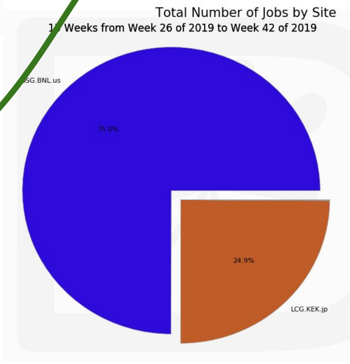
Backup slides

History of Proc9/Bucket7 on the grid

- Start on 13/7: fast start at BNL (and kekcc)
  - BNL finish after 10 days, kecc goes on slowly
  - kekcc stop due to shutdown (august)
- Kekcc restart (slow)
  - Some runs with more than 1000 jobs not properly handled by dp tools: fixed. Immediately run at BNL
  - Increase of resources at kekcc to finish
- Overall, took >1 month to complete!
  - At KEKCC 8.5 days
- Bucket7 started and finished quickly



🟦	OSG.BNL.us	85.5%
🟧	LCG.KEK.jp	14.5%



Lesson learned for grid processing

- Good progress in coupling the DP tools to grid processing tools
  - Grid specific configuration files and submission is generated automatically by the same tools used for local kekcc processing
  - Single source of configuration, more robust against errors
    - Can be shared with MC production and analysis skims, discussion is ongoing
  - Some data-related issue fixed during the process
    - Long run with >1000 jobs
    - Temporary invalidation of problematic input files (eg reconstruction fails), and revalidation when patch available
- Major issue with Proc9 was raw data staging
  - All data staged, but only partially at BNL, the rest at KEKCC
  - Jobs created and scheduled according to data availability, so many at kekcc.
    - Ask for staging (in advance) and wait before submission
    - Could have rescheduled the jobs at kekcc once raw data available at BNL
  - Introduced step to check staging before submission

Release 4

- Updated production script for new interface
  - https://agira.desy.de/browse/BIIDP-1776
    - **change GT handling in release 4**
      - **Support multiple GT in json (previously only one, the rest hardcoded)**
    - Remove SetDataFlagModule (not needed anymore)
      - isMC: 0
    - add SuppressErrorMask=0xFFFFFFFFFFFFFFFF to PXDUnpacker
    - EventsOfDoomBuster now in add_reconstruction() with tuned threshold cuts
    - Better handling of path for Cosmic
      - /group/belle2/dataprod/Data/release-04-00-00/DB00000711/Unofficial/e0010/**Cosmic**/
    - Other minor improvement.
- Question: path is now: /path/<release>/<**GTID**>/<Campaign>/<EXP>/<Type>
  - But we are not using a single GTID anymore
    - Eg: **data_reprocessing_prompt_rel4_patchb** and **online**
  - I'm using the first as GTID in the path, but that ID is not telling the full story. Should we revise it?
    - globalTag: data_reprocessing_prompt_rel4_patchb,online

Release 4 test on collision data

- Tested on several exp 8 runs
  - https://agira.desy.de/browse/BII-5621
  - Found a number of [ERROR]
  - Full list in jira ticket
    - [ERROR] Correction factor=0 is very small/too large! Resetting to 1.0. { module: ECLShowerCorrector } ... message repeated 49 times [INFO] Processed: 1 runs, 2 events [ERROR] Correction factor=0 is very small/too large! Resetting to 1.0. { module: ECLShowerCorrector } https://agira.desy.de/browse/BII-5446
      - Patch forgotten in release-4
    - [ERROR] DHP data loss (CM=63) in 1220527 times, on 4 runs: 00224 00301 00799 02430 https://agira.desy.de/browse/BII-5622
      - ERROR should have been WARNING
    - Other about event too crowded for reconstruction or corrupted data

No major issues.

Release 4 test on Cosmic

- Tested on several exp 10 cosmic runs
  - **[FATAL] ERROR_EVENT : Invalid header size of FTSW data format(= 0x00000008 words). Exiting…**
    - https://agira.desy.de/browse/BIIDP-1932
    - inconsistency between RawFTSW unpacker merged on Aug. 20 and data which Nakao-san started sending with an updated data-format
  - https://agira.desy.de/browse/BII-5654
  - From run 1500 Nakao-san restored original RawFTSW format / version number, until the RawFTSW unpacker is further updated to be compatible
      - Tested and confirmed
    - RawFTSW is used to remove random trigger from processing
      - in principle we can use any RawXXX (all contains trigger type), but not always present.
  - **Error in <TStreamerInfo::Build>: Belle2::PXDRawROIs, discarding: int* m_rois, no [dimension]**
    - [Bjoern] Did someone **AGAIN** killed the root streamer by changing the comment to be "doxygen" conform. wow. i am impressed.
      - https://stash.desy.de/projects/B2/repos/software/pull-requests/5071/overview