Multicore workload scheduling in JUNO

Xiaomei Zhang Kang Li Andrei Tsaregorodtsev Xianghu Zhao

Institute of High Energy Physics

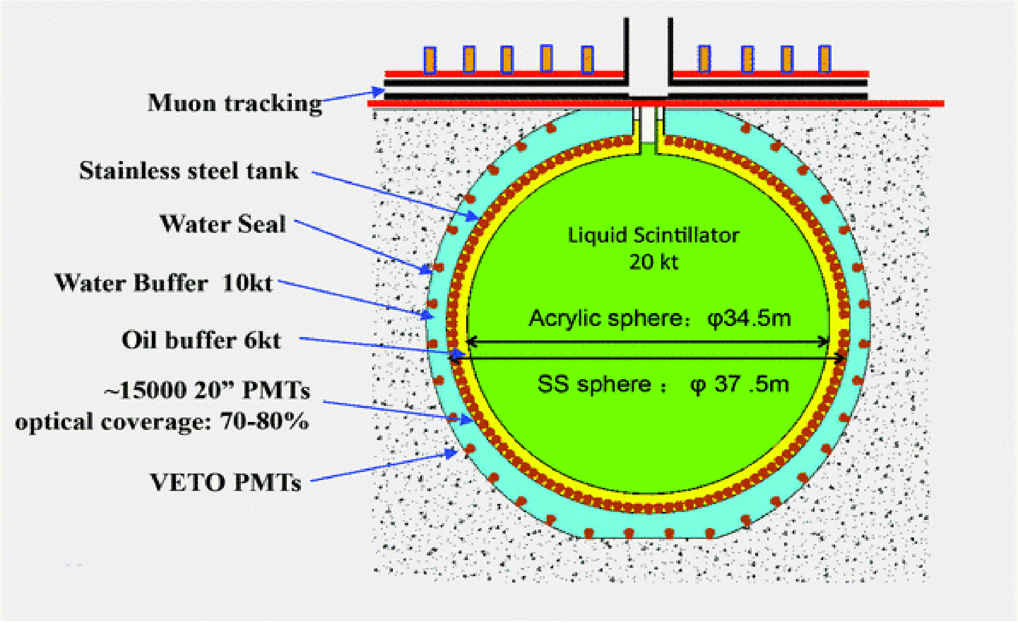
CHEP2018, Sofia

Content

- Motivation for multicore support in JUNO
- Multi-core pilot mode strategy
- Implementation and testing
- Efficiency study and optimization
- Summary

Jiangmen Underground Neutrino Observatory

- JUNO, a multi-purpose neutrino experiment designed to measure the neutrino mass hierarchy and mixing parameters
  - Start to build in 2014, operational in 2019, located at Guangzhou province
  - Estimated to produce 2PB data/year for 10 years
  - 20 kt Liquid Scintillator detector, 700m deep underground
  - 2-3% energy resolution
  - Rich physics opportunities



JUNO Parallel Data Processing

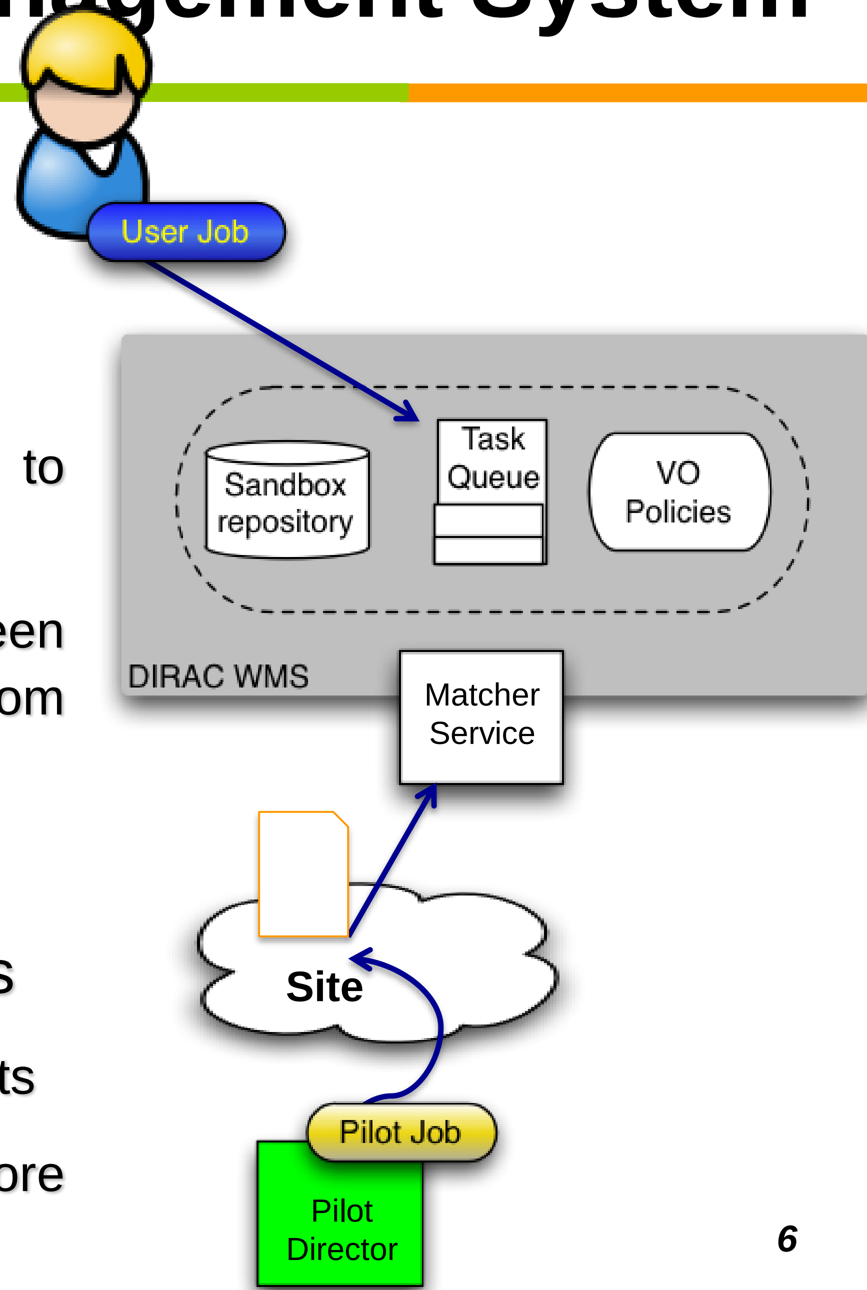
- Parallelization is being introduced into JUNO offline software system based on TBB
  - Fasten JUNO data processing and fully use modern multi-core and many-core hardware
  - Enable multi-thread and multi-process simulation and reconstruction
- Event-level parallel processing of the JUNO offline software framework SNiPER is already in prototype phase
  - See Jiaheng Zou’s talk “The Event Buffer Management for MT-SNiPER ”
- Simulation based on Geant4.10 is in good progress
  - See Tao Lin’s talk “Status of parallelized JUNO simulation software”

Dirac-based JUNO distributed computing

- JUNO Distributed Computing (DC) has been built on DIRAC to organize heterogeneous and distributed resources
  - Able to integrate with Cluster, Grid and Cloud
  - Currently work in single-core mode
- To accept the coming multi-core jobs， new workload scheduling strategy has to be introduced into JUNO DC Workload Management System (WMS)
- Multi-core design objectives
  - Allow to have both single-core and multi-core JUNO jobs coexisting in a long period
  - Capable to share resources with other experiments on the same sites with good efficiency

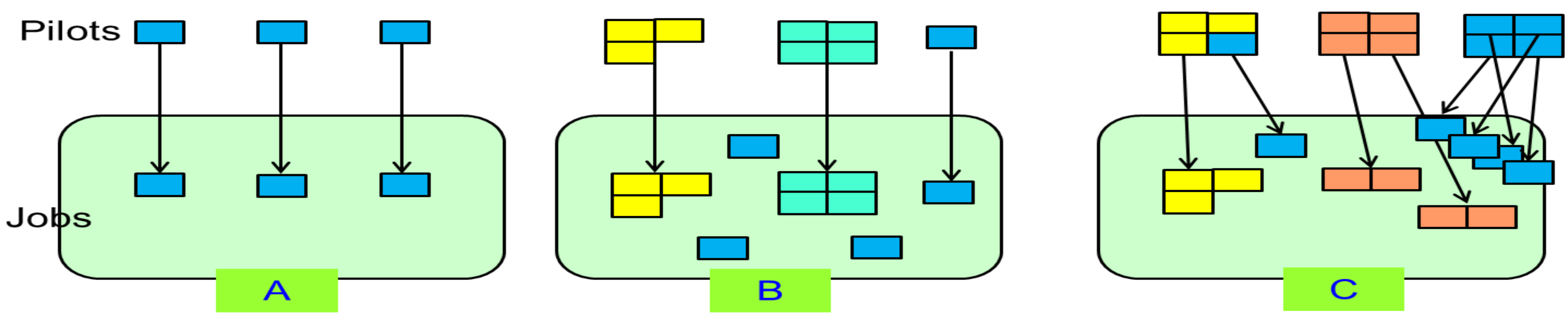
DIRAC Workload Management System

- DIRAC workload scheduling based on pilots strategy
  - User jobs arrive in **TaskQueue**
  - **Pilot Director** submits pilot jobs to sites
  - **Matcher** does the matching between Pilot jobs and users jobs from TaskQueue
  - **Pilots** accept and start user jobs
- Key point for multi-core supports
  - single-core Pilots to multi-core Pilots
  - Matching between multi-core resource and multi-core jobs



Multi-core pilot designs (1)

- In current single-core(SC) pilot mode
  - Each pilot takes one slot from local resource
  - Pull one SC job from job pools
- In multi-core(MC) pilot mode, to accept MC jobs
  - Each pilot need to occupy one or more slot
  - Each pilot can pull one or more jobs from job pools

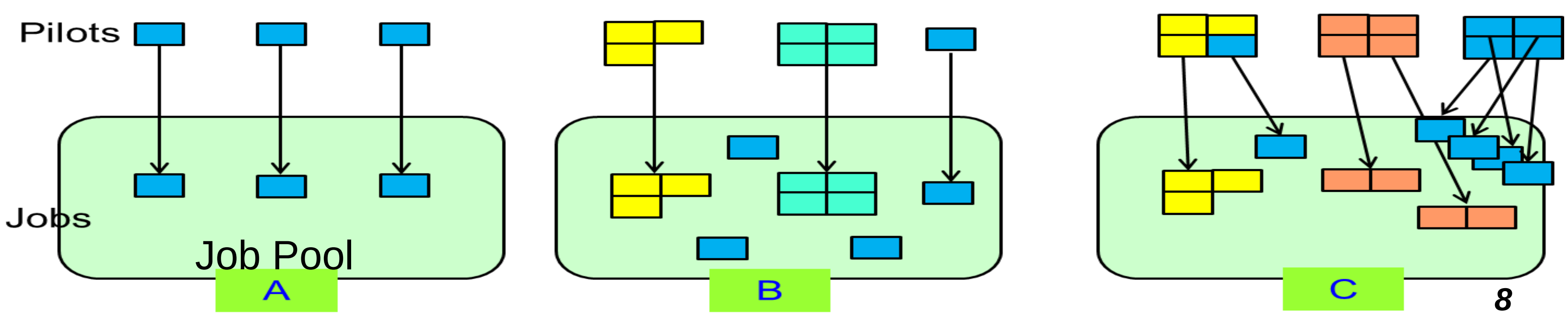


Multi-core pilot designs (2)

- There are two strategies to provide multi-core pilots

(1) **Customized pilots** (B)

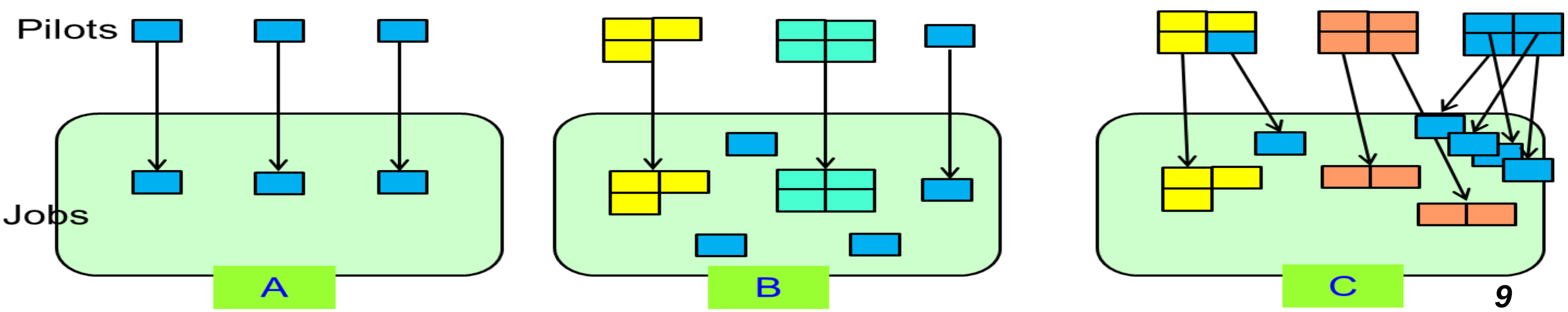
- Send pilots with the same size as the jobs to be pulled
- M-core pilots occupy M slots and pull M-core jobs
- Can accept both single-core and multi-core jobs
- But low efficient when matching with a hybrid of various-core jobs
  - pilot “starving” will happen



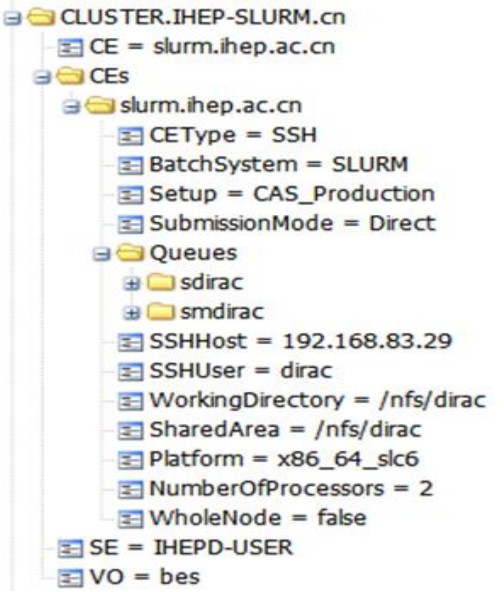
Multi-core pilot designs (3)

(2) Shared partitionable pilots (C)

- Send Pilots with same number of cores
- The size of pilots can be whole-node, 4-node, 8-node….., adjusted according to site policy
- M-core Pilots pull more than one N-core jobs (N<=M) until internal slots used up
- For a hybrid of various-core jobs, expected to be more efficient than customized pilots since pilots can be shared by different-core jobs



Tags for matching



- In multi-core case
  - Jobs have requirements on cores
  - Sites have different number of cores to provide
- Tags introduced to mark jobs and resources for matching
  - Sites define number of cores to be accepted in DIRAC CS
    - NumberOfProcessors: Number of cores can be got from the site
    - RequiredTag: Number of cores can be pulled
  - Jobs define number of cores required in JDL
    - Tags=Nprocessors
    - Tags=WholeNode occupy all slots in one WN
  - Job Tag information will be kept in TaskQueue
  - Matcher uses these tags to do final matching

Multi-core pilots Implementation

- In customized Multi-core mode
  - MC pilot directors are introduced to submit MC pilots corresponding to the job tags in TaskQueue
- In partitionable Multi-core mode
  - Pilot directors are adjusted to submit pilots with same number of cores
  - New pilot working mode is introduced in pilots
    - Can accept more than one job
    - Auto-detect the available cores and do simple scheduling, just like little "cluster"
- Matching service takes care of matching using tags from JobDB and DIRAC central configuration service

Interface to sites

- To completely enable multi-core modes, also need sites to accept multi-core jobs
- For Batch system or Grid
  - A multi-processor queue or whole node queue need to be created to accept multi-core pilot jobs
  - The interface to submit jobs to sites also need to add supports of multi-core jobs submission commands
- For Cloud
  - VM Director, in the same role of Pilot Director, need to be adjusted to create multi-core VMs instead of submitting multi-core pilots
  - Multi-core pilots auto-booted up in VMs to get multi-core jobs

Monitoring for each pilot and job

- In Job Monitoring, Number of cores used by Jobs is added

Items per page: 25 Page 1 of 1 Updated: 2018-04-12 02:39 [UTC] Displaying topics 1 - 23 of 23

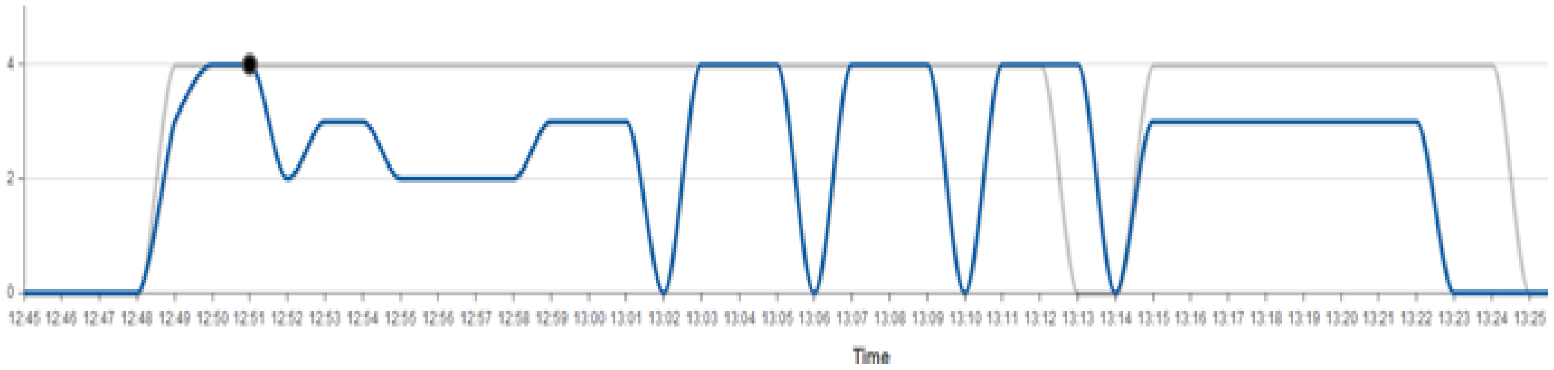
JobId	Status	MinorStatus	ApplicationStat	Site	JobName	LastUpdate[UTC]	LastSignOfLife[UTC]	SubmissionTime[UTC]	Owner	JobCores
112581	Done	Execution Complete	Unknown	CLUSTER.IH...	jobagent	2017-10-16 12:38:57	2017-10-16 12:38:57	2017-10-16 12:38:11	likang	2
112580	Done	Execution Complete	Unknown	CLUSTER.IH...	jobagent	2017-10-16 12:38:56	2017-10-16 12:38:56	2017-10-16 12:38:10	likang	2
112579	Done	Execution Complete	Unknown	CLUSTER.IH...	jobagent	2017-10-16 12:39:36	2017-10-16 12:39:36	2017-10-16 12:38:10	likang	2
112578	Done	Execution Complete	Unknown	CLUSTER.IH...	jobagent	2017-10-16 12:39:37	2017-10-16 12:39:37	2017-10-16 12:38:09	likang	2
112577	Done	Execution Complete	Unknown	CLUSTER.IH...	jobagent	2017-10-16 12:36:57	2017-10-16 12:36:57	2017-10-16 12:34:31	likang	2

- In Pilot monitoring, Cores information of pilots are added to
  - TotalCores to know the total number of processors the pilot got
  - UsedCores to know current cores being occupied

Site	ComputingElem	Broker	CurrentJobID	GridType	TaskQueueID	BenchMark	OwnerGroup	LastUpdateTime[UTC]	SubmissionTime[UTC]	PilotCores	RequiredTag	AvailableCores
CLUSTER.IH...	tslurm.ihep....	besdirac01.i...	69871	SSH	5836	22.7	bes_pilot	2017-08-07 07:48:00	2017-08-07 07:39:49	8		24
CLUSTER.IH...	tslurm.ihep....	besdirac01.i...	-	SSH	5836	0	bes_pilot	2017-08-07 07:55:32	2017-08-07 07:39:50	8		24
CLUSTER.IH...	ttslurm.ihep...	besdirac01.i...	71892	SSH	78	22.4	bes_pilot	2017-09-18 09:11:20	2017-09-18 09:05:05	12		12
CLUSTER.IH...	ttslurm.ihep...	besdirac01.i...	71874	SSH	78	22.7	bes_pilot	2017-09-18 09:11:11	2017-09-18 09:05:06	12		12
CLUSTER.IH...	ttslurm.ihep...	besdirac01.i...	71877	SSH	78	22.3	bes_pilot	2017-09-18 09:11:18	2017-09-18 09:05:07	12		12

Monitoring for each pilot and job

- Pilots monitoring graph shows scheduling efficiency for the chosen pilot
  - X: Time, Y: Cores
  - Gray line shows available core in pilots
  - Blue line shows cores used by jobs
- From graph, we can see cores of pilots are not fully used in its life cycle

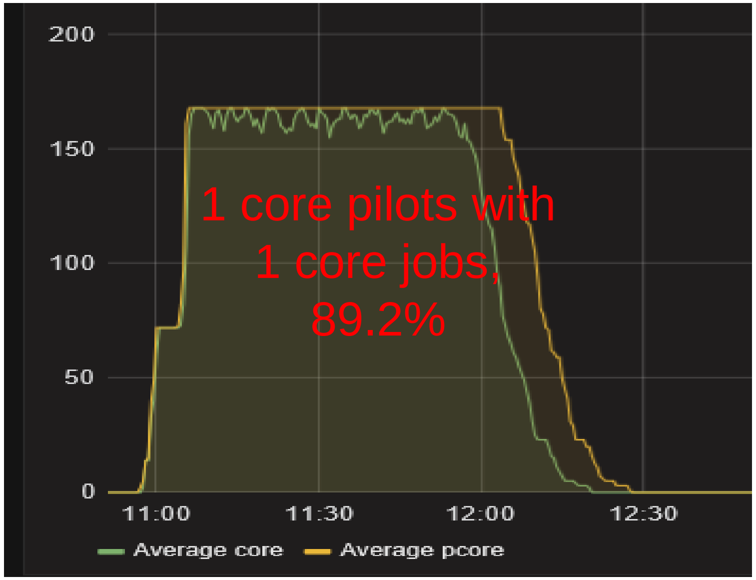


Tests

- Tests have been done wtih SLURM and HTCondor sites
  - JUNO Geant4 Monte Carlo jobs
  - 216 CPU core, each nodes with 12/24 cores
- Three job type input included
  - Single-core, whole-node
  - Mixture of SC and MC jobs
- Monitoring and accounting use ElasticSearch and Ganglia
- Three modes are tested and working well
  - Single-core
  - Customized Multi-core
  - Partitionable Multi-core

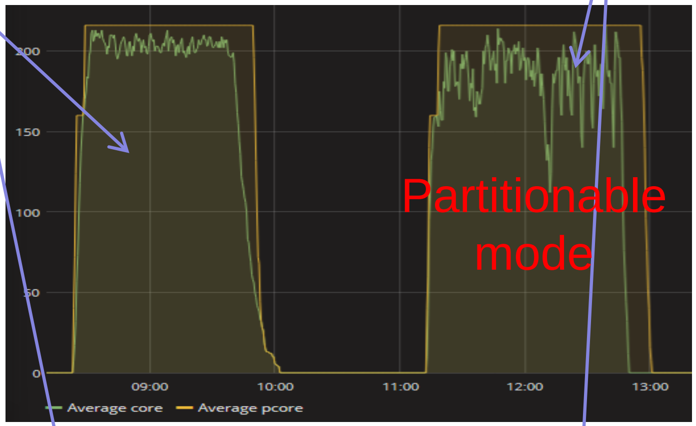
Efficiency study (1)

- With SC jobs, scheduling efficiency of three modes has no big differences
  - With same input of jobs
- Overhead and tail come from the pilot itself who need time for its life cycle

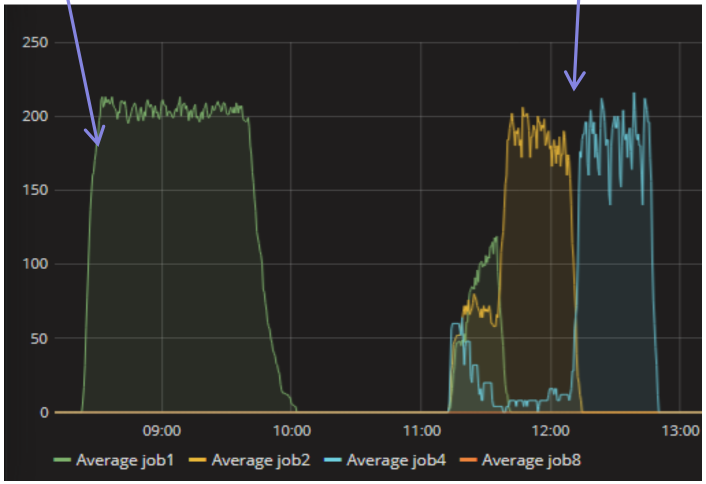


Efficiency study (2)

- Tests also done with a hybrid of various-core jobs
- Scheduling efficiency of Customized pilots (48%) much worse than that of Partitionable pilots(81%) as expected
  - More idle pilots in customized pilot mode due to its one-to-one matching policy
- Scheduling efficiency of Partitionable pilots mode also not good than SC mode
  - Resources occupied not fulfilled



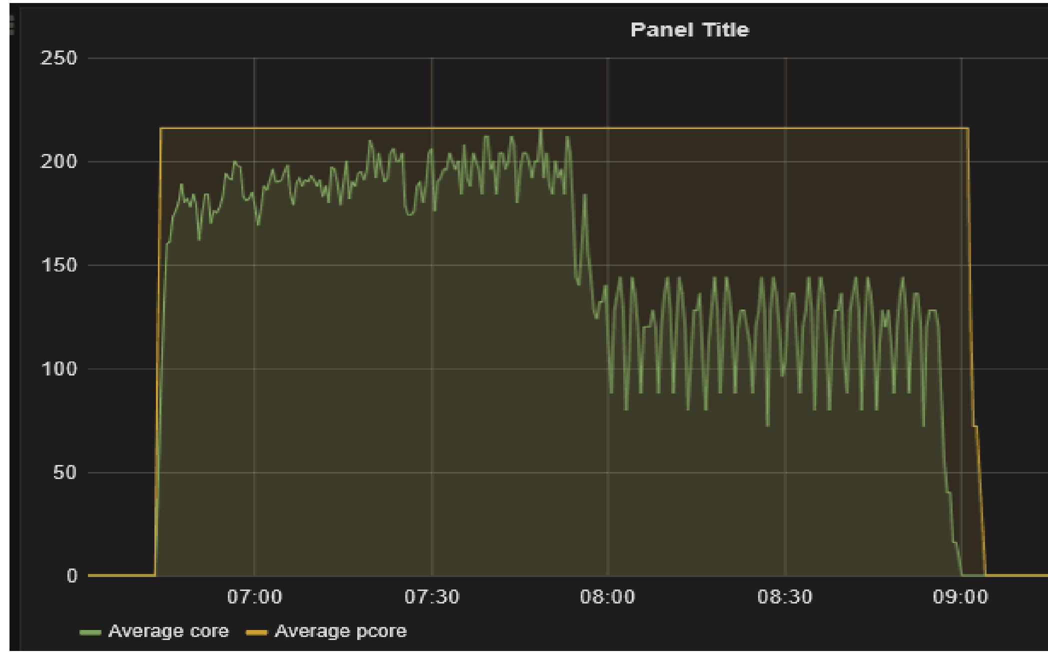
pilot scheduling efficiency

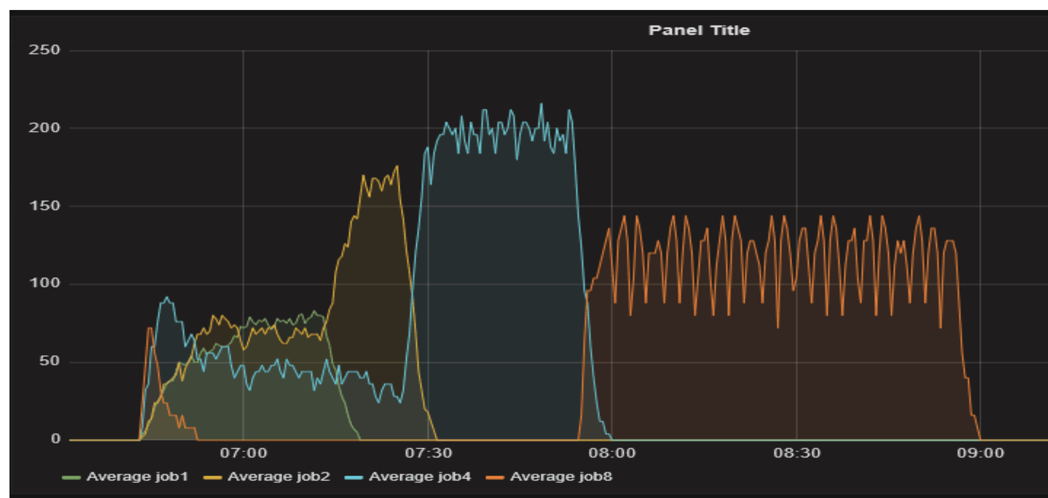


jobs scheduled in pilots

Efficiency study (3)

- Deep into partitionable pilots mode
  - 12-core pilots
  - 1 core: 2 core: 4 core: 8core = 1:1:1:1
  - Efficiency is 75%
- One of main efficiency loss is due to scheduling policy
  - Most jobs with less cores are easily selected at beginning
  - 8-core jobs are finished at last past with 4-core idle





Efficiency optimization

- Improvements on Scheduling policy in Matcher
- Old : Randomly choose jobs matched
- New: Choose jobs with high priority
  - Define priority with related factors, including
    - Jobs waiting time, rest of cores in pilots and cores requested by jobs
  - An example to count priority of job（i）, you can add more factors in

$$P_i = ae^{k(v-c_i)^2} + b(w_i + r_i)/r_i$$

The first part is to choose “Big” jobs to reduce resource gap

- The smaller core gap between pilot（v）and the job（c）, the higher priority the job get

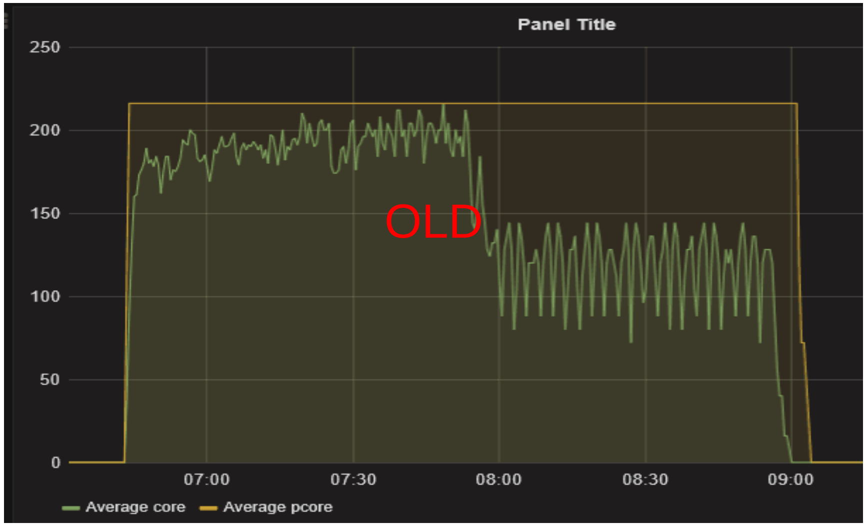
The second part is to avoid “starving” of “Small” jobs

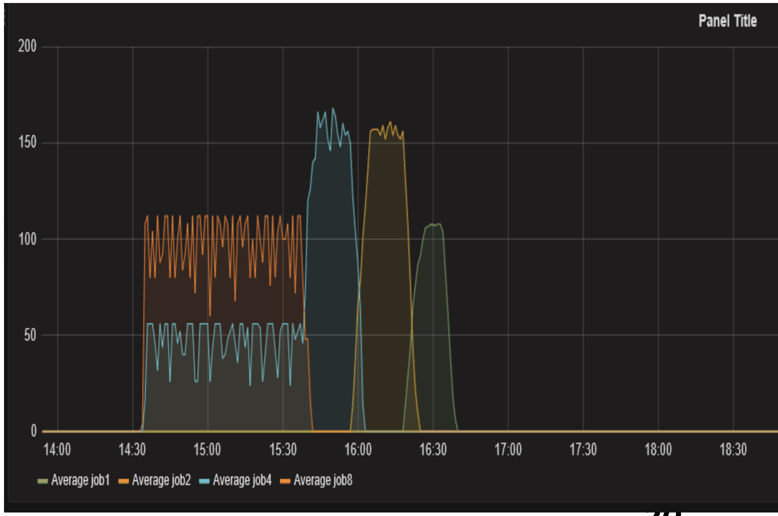
- The higher waiting time（w）above average waiting time (r), the higher priority got

Experiments can tune parameters a、b、k according to different cases

Efficiency optimization

- The tests with new policy showed that the efficiency can be improved 15%
  - "Big" jobs are matched first
  - Single-core jobs can fill the remaining gaps





Summary and outlook

- Two multi-core pilot modes have been implemented
- The prototype of multi-core supports in DIRAC-based JUNO distributed computing platform is working properly
- Scheduling efficiency is a concern hybrid of various-core jobs
- Efficiency study shown that the partitionable pilot mode is more promising in hybrid of various-core jobs
- With improvement of scheduling policy, the scheduling efficiency of partitionable pilot mode can be improved a lot
- Parameters need to be tuned with future real user cases and job pressure