Update for Intel End2End AI Benchmarking

Martyn Winn

2nd Nov 2020

Summary

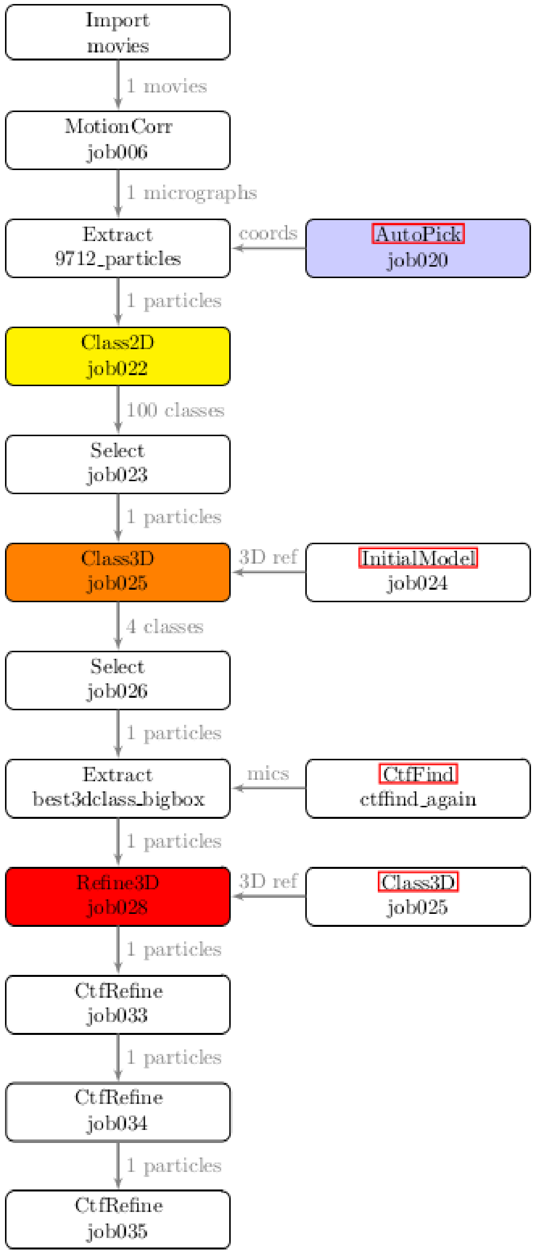
- Workflow
  - Tutorial data
  - Aldolase
- Performance benchmarks
  - Checking the results
- Benchmarking data
  - Intermediate datasets
- Benchmarking scripts

M2: Relion tutorial data

- Class2D, Class3D, Refine3D are the "big" jobs
- All use relion_refine_mpi binary
- For tutorial, these only take a few minutes

- AutoPick job finds particles in micrographs
- This is where we can explore machine learning / image recognition methods

Branched flowchart for CtfRefine/job035



Import movies

1 movies

MotionCorr job006

1 micrographs

Extract 9712_particles

coords

AutoPick job020

1 particles

Class2D job022

100 classes

Select job023

1 particles

Class3D job025

3D ref

InitialModel job024

4 classes

Select job026

1 particles

Extract best3dclass_bigbox

mics

CtfFind ctffind_again

1 particles

Refine3D job028

3D ref

Class3D job025

1 particles

CtfRefine job033

1 particles

CtfRefine job034

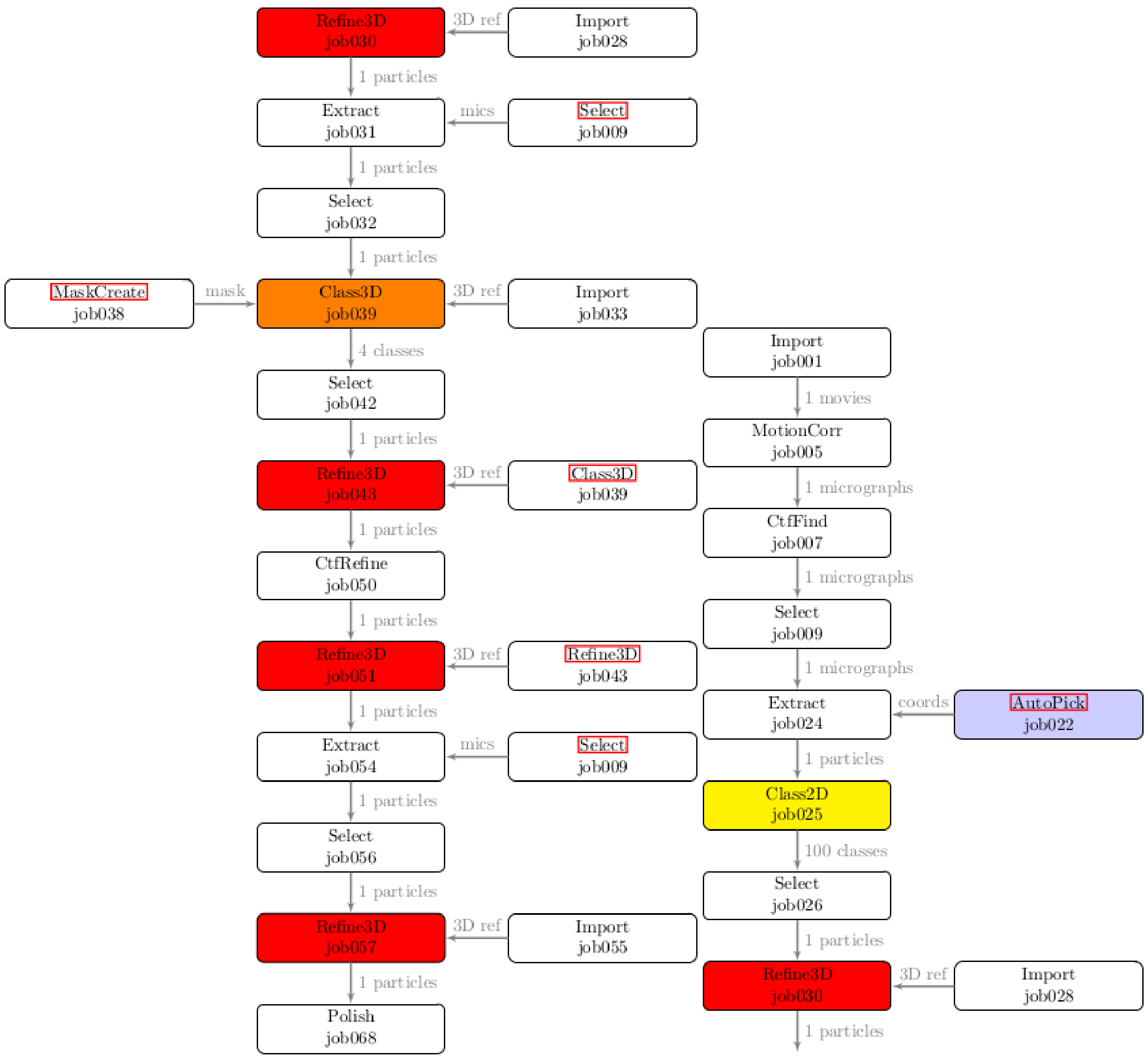
1 particles

CtfRefine job035

M2: Aldolase data

- More complicated workflow.
- Big jobs are now big, see benchmarking below.

Branched flowchart for Polish/job068



Refine3D job030

3D ref

Import job028

1 particles

Extract job031

mics

Select job009

1 particles

Select job032

1 particles

MaskCreate job038

mask

Class3D job039

3D ref

Import job033

4 classes

Select job042

1 particles

Refine3D job043

3D ref

Class3D job039

1 particles

CtfRefine job050

1 particles

Refine3D job051

3D ref

Refine3D job043

1 particles

Extract job054

mics

Select job009

1 particles

Select job056

1 particles

Refine3D job057

3D ref

Import job055

1 particles

Polish job068

Import job001

1 movies

MotionCorr job005

1 micrographs

CtfFind job007

1 micrographs

Select job009

1 micrographs

Extract job024

coords

AutoPick job022

1 particles

Class2D job025

100 classes

Select job026

1 particles

Refine3D job030

3D ref

Import job028

1 particles

M5: Performance benchmarking

Class2D

job	binary	hardware	Tasks / threads	Nodes	pool	Runtime h:m	
job025	basic	Broadwell	9 / 6	3	3	27:17	
job063	basic	Skylake	9 / 6	3	3	23:12	Change CPU
job064	basic	Skylake	9 / 6	3	12	23:06	Change pool
job065	accelerated	Skylake	9 / 6	3	12	6:40	Change binary

Refine3D

job	binary	hardware	Tasks / threads	Nodes	pool	Runtime h:m
job030	basic	gpu	5 / 1 / 4 gpu	1	30	16:16
TBC	basic	Skylake	11 / 6		30	Est. 6 days
job069	accelerated	Skylake	11 / 6	3	30	35:00

Class3D

job	binary	hardware	Tasks / threads	Nodes	pool	Runtime h:m
job039	basic	gpu	5 / 1 / 4 gpu	1	30	7:57
job066	basic	Skylake	9 / 6	3	30	34:32
job067	accelerated	Skylake	9 / 6	3	30	8:47

4-fold increase in speed on Skylake ☺

But GPU still preferable
[Using dual K80 cards (4 K80 devices available on node)]

M5: Correct answer?

E.g. jobs for Class3D run with identical parameters on 3 different platforms / binaries:

job	**binary**	**hardware**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
job039	basic	gpu	0.12	**0.49**	0.18	0.22
job066	basic	Skylake	0.21	**0.45**	0.21	0.14
job067	accelerated	Skylake	0.19	**0.37**	0.28	0.16

Similar but different!
Expected some stochasticity.
Dominant class not so clear for accelerated run.

M4: Delivering data … which data?

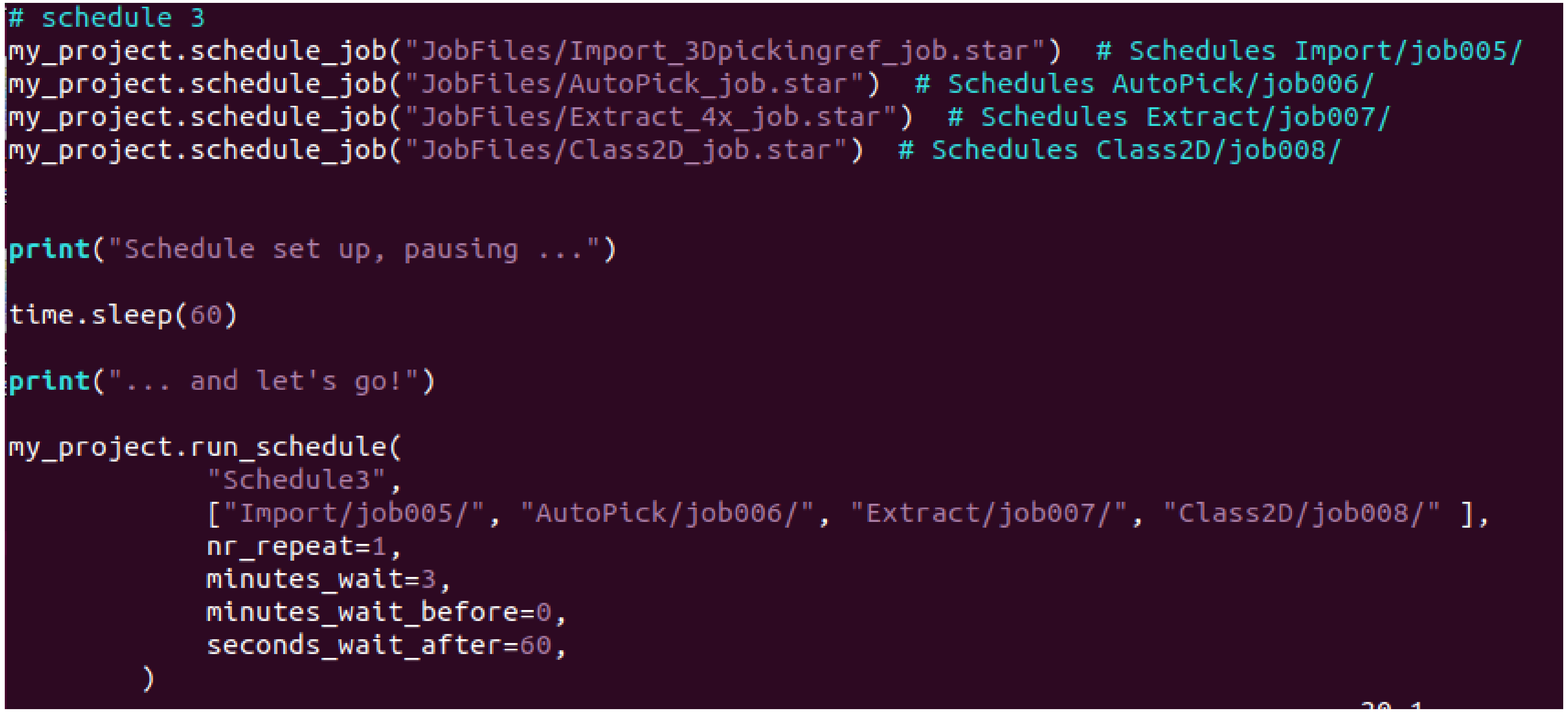
Tutorial	
Input movies	3793 MB
Total for completed project	11 GB
Largest subdir (Refine3D)	2820 MB
Total after gentle clean	8 GB
Largest subdir (Extract)	1649 MB
Largest subdir (MotionCorr)	1650 MB
Total input + selected intermediate	8 GB

Aldolase (July)	
Input movies	337 GB
Total for completed project	1,545 GB
Largest subdir (Extract)	489 GB
Total input + selected intermediate	???

Select output from some intermediate jobs to allow easy running of Class2D, Class3D, Refine3D
Rest could be re-generated from scripts
For aldolase, still considering appropriate split.

M5: Delivering scripts

- Command line using newly developed python API to Relion:



```
# schedule 3
my_project.schedule_job("JobFiles/Import_3Dpickingref_job.star")  # Schedules Import/job005/
my_project.schedule_job("JobFiles/AutoPick_job.star")  # Schedules AutoPick/job006/
my_project.schedule_job("JobFiles/Extract_4x_job.star")  # Schedules Extract/job007/
my_project.schedule_job("JobFiles/Class2D_job.star")  # Schedules Class2D/job008/


print("Schedule set up, pausing ...")

time.sleep(60)

print("... and let's go!")

my_project.run_schedule(
            "Schedule3",
            ["Import/job005/", "AutoPick/job006/", "Extract/job007/", "Class2D/job008/" ],
            nr_repeat=1,
            minutes_wait=3,
            minutes_wait_before=0,
            seconds_wait_after=60,
        )
```

- Job files contain necessary parameters to re-run jobs.
- We will bundle with binaries and configuration scripts. Mount data.