PAWSEY supercomputing centre

Data-driven Workflows on Crays with Hybrid Scheduleding

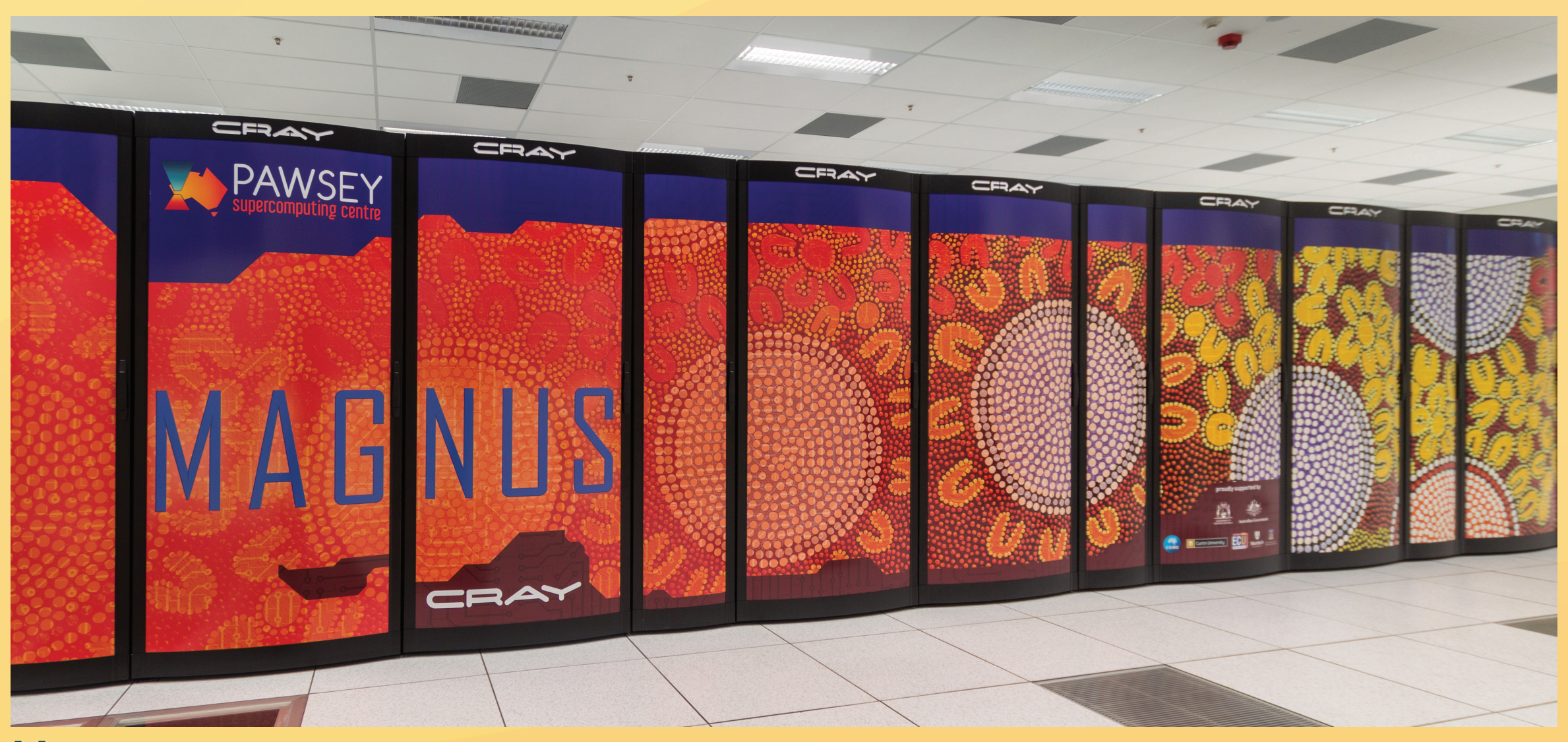
A Case Study of Celera on Magnus

Charlene Yang
Pawsey Supercomputing Centre, Perth, Western Australia
charlene.yang@pawsey.org.au

Seyhan Yazar
George Gooden
Alex Hewitt
Lions Eye Institute, The University of Western Australia
seyhanyazar@lei.org.au
georgegooden@lei.org.au
alexhewitt@lei.org.au

ABSTRACT

This poster presents an example of data-driven workflows running on a SLURM+ALPS scheduled Cray machine at Pawsey Supercomputing Centre. This example adapts the *de novo* assembly software WGS-Celera, which originally has no SLURM support, to be able to dynamically request resources and distribute work through the hybrid scheduling system on Magnus, a Cray XC40 at Pawsey. The changes made to the code are uploaded to GitHub, and some insights about the adaptation are offered in this poster to benefit other users of Celera or other similarly-structured workflows around the world.


Mangus

Cray XC40	Compute	Intel Xeon E5-2690V3 Haswell, 1,488 nodes, 35,712 cores, 1,097 TeraFLOPS
	Memory	DDR4-2133, 64GB per node, 93 TB total
	Interconnect	Cray Aries, 72 Gb/s per node
	Topology	Cray Dragonfly, 56% populated
	Storage	Sonexion 1600, 3 PB, w/ 70 GB/s, Lustre filesystem
	Other specs	weight 1.7 tonnes/cabinet, power 50 kW/cabinet

I. BACKGROUND

While a few HPC sites are experimenting with native SLURM on their Cray systems, the majority of Cray machines on the TOP500 List are still using a hybrid scheduling mechanism, be it SLURM+ALPS, PBS+ALPS or Torque+ALPS (see Fig. 1). The reason is either trying to avoid disruption to production system or having a long-standing tradition with PBS or other workload managers.

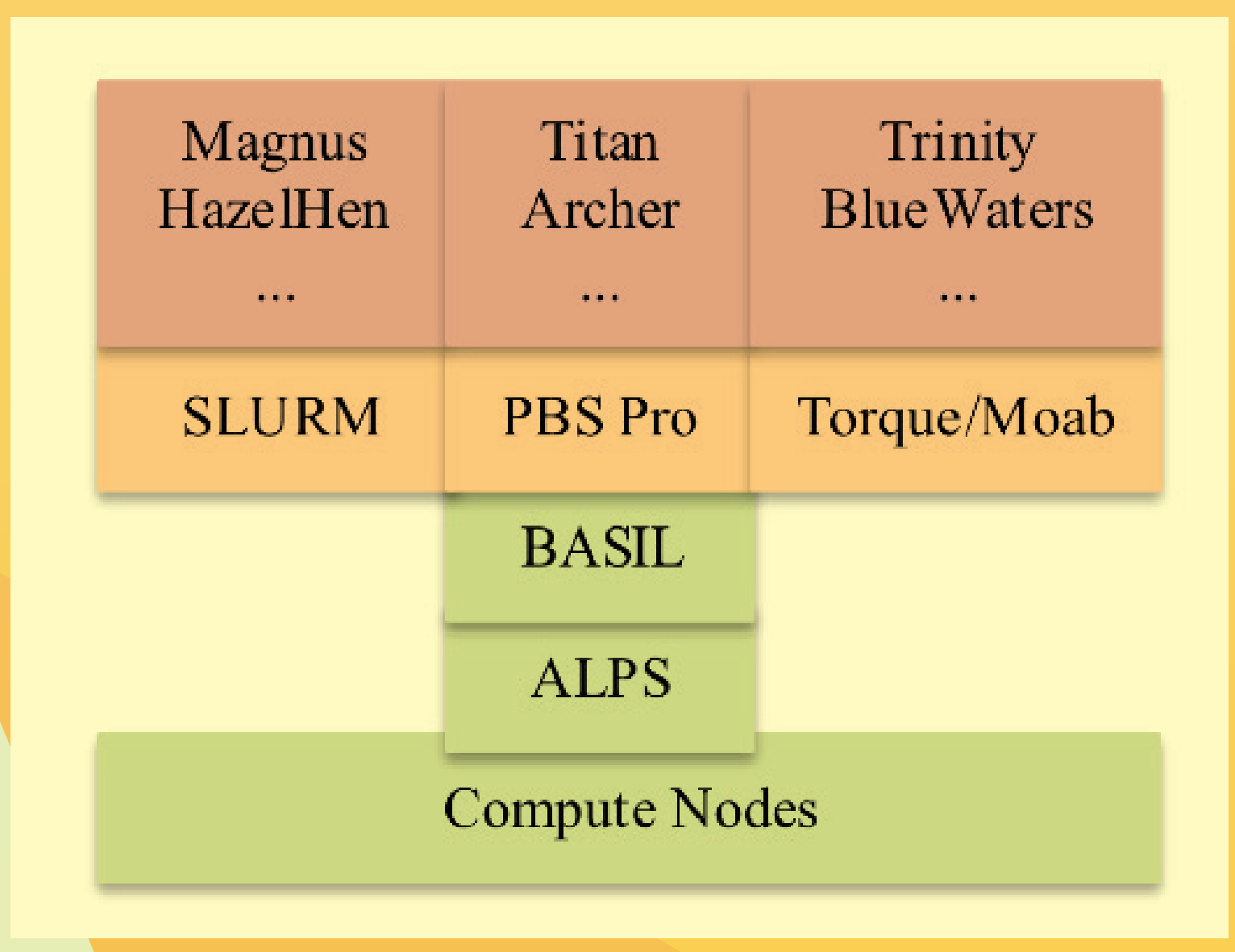


Fig. 1: Hybrid scheduling mechanism on TOP500 listed Crays

This hybrid scheduling mechanism consists of two parts: on the top a third-party workload manager looks after job prioritizing, scheduling and accounting, and at the bottom Cray's primitive resource manager ALPS [1] takes care of task placement and job execution. Communication between the two layers is through BASIL, an XML-based interface [1]. Take Magnus, a Cray XC40 at Pawsey Supercomputing Centre, as an example. Users need to *sbatch* on the external login nodes to request resources and then *aprun* from the MOM nodes to launch jobs to the compute nodes.

II. PROBLEM AND SOLUTION

Since there are no workload scheduler daemons running on the compute nodes, submitting jobs from the backend is impossible. This makes it difficult for some complex data-driven workflows and a few examples are listed in Table I. These workflows consist of a number of analysis stages in the pipeline. Different stages may require different amount of compute resources depending on the output of previous stages (e.g. data size), and the difference can be very significant. Being able to flexibly request resources and distribute jobs therefore is a challenge, especially on the Cray supercomputers where hybrid scheduling mechanism is deployed.

Fields	Software	High-level Scripts	Low-level Programs
Genomics	Celera, VelvetOptimiser	Perl	C
Fluid Dynamics	ANSYS, FLUENT	Shell	Fortran/C/C++
Chemistry	ADF, BAND	Shell	Fortran/C
Geophysics	Madagascar	Python	C

Table I: Complex data-driven workflows

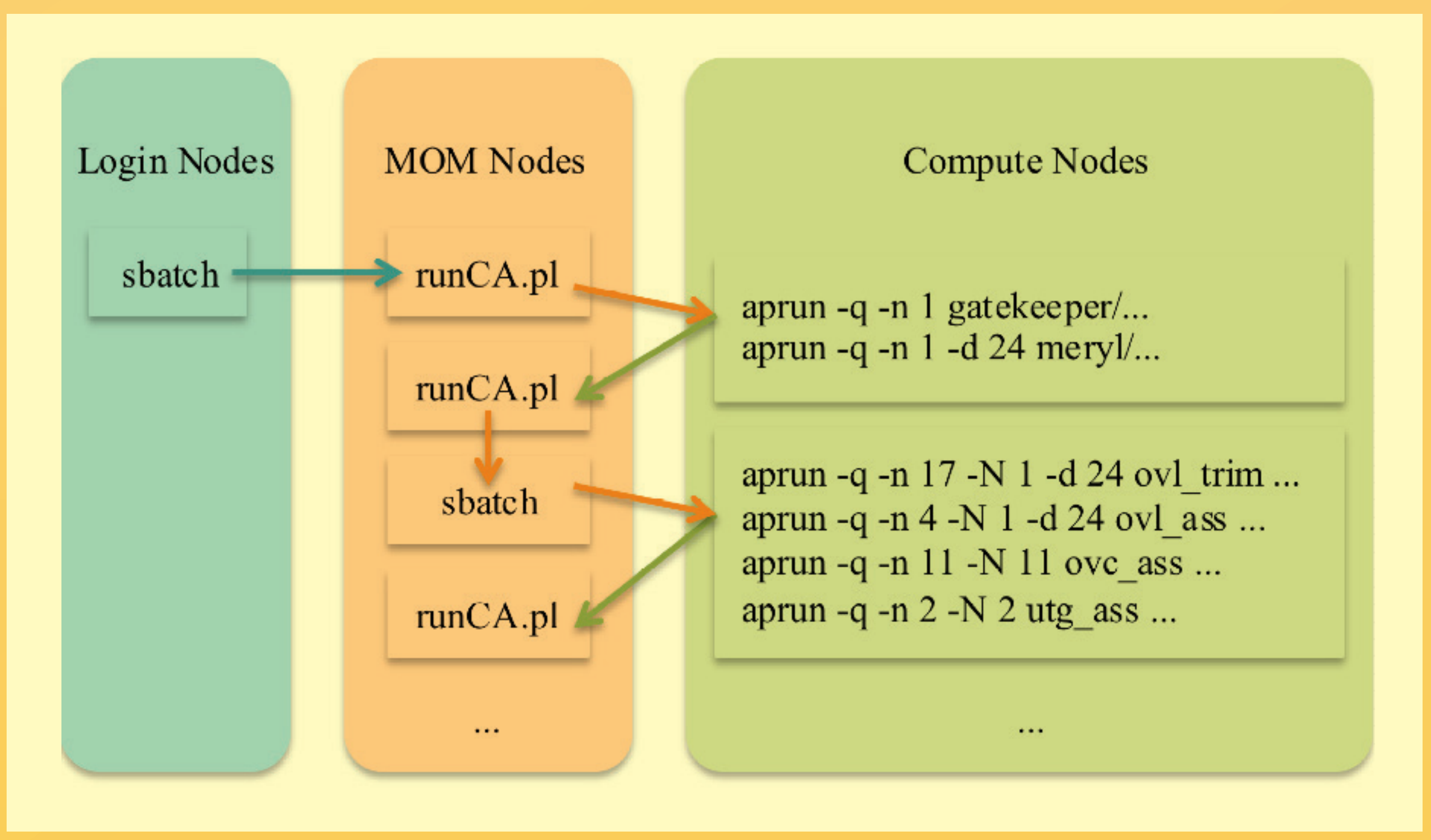
WGS-Celera [2] is an example of these data-driven workflows. It is a *de novo* assembly pipeline, and like other examples in Table I, it has a Perl script on the high level, managing roughly 10 steps of the assembly process, and on the low level it has about 50 C programs (some serial and some are OpenMP threaded) to perform heavy-duty computation. The Perl script *runCA* controls the overall flow of the pipeline, and it occasionally dispatches separate jobs to the queue (e.g. SLURM on Magnus) to run different steps on different pools of compute resources (of different sizes). When that finishes, it then resumes to the main workflow, i.e. detect the completed steps and restart from the next unfinished step.

The latest version of Celera is 8.3rc and it doesn't have SLURM support originally. Therefore work was first done to add SLURM options in so it can submit, delete and alter jobs, job arrays and support job dependencies. Then the following three steps were taken in order to fully adapt Celera to run on Magnus, a Cray XC40 at Pawsey with SLURM+ALPS hybrid scheduling.

1. Categorize the work in Celera to three tiers:
   Tier-1) workflow management, such as detecting pipeline progression, locating restart points, creating directories and secondary job scripts,
   Tier-2) light computation, such as serial and small threaded jobs that can run on one node, and
   Tier-3) heavy computation, which is mainly jobs that require significantly more resources than Tier-2 jobs.

2. Place Tier-1 work on MOM nodes, Tier-2 on smaller pool of compute resources and submit Tier-3 as separate jobs to the queue so they can leverage different sizes (bigger than Tier-2) of pools of resources flexibly. (Note: submission needs to be initiated from MOM nodes).

3. Identify different data parallelism in Tier-3 and implement it by either using job arrays (for threaded jobs) or job packing via environment variable ALPS_APP_PE (for serial jobs).



```
aprun -q -n 1 gatekeeper/...
aprun -q -n 1 -d 24 meryl/...
```

```
aprun -q -n 17 -N 1 -d 24 ovl_trim ...
aprun -q -n 4 -N 1 -d 24 ovl_ass ...
aprun -q -n 11 -N 11 ovc_ass ...
aprun -q -n 2 -N 2 utg_ass ...
```

Fig. 2: Celera pipeline on Magnus

Fig. 2 depicts the adapted Celera workflow on Magnus. The numbers, 17, 4, etc, are for a specific dataset (see Section III) and other user cases will have different sizes for those jobs. In general, the workflow is kick started with a primary job being submitted from the external login nodes. This job runs *runCA* on the MOM nodes and launches Tier-2 jobs such as *gatekeeper* and *meryl* to the compute node(s) it's allocated to. When up to large data/compute-intensive steps such as *ovl_trim* and *ovc_ass*, *runCA* submits a separate job to the queue, accompanied by itself as another job which depends on the finish of *ovl_trim* or *ovc_ass*. The new *runCA* job then detects the restarting point and proceed to the next unfinished step of the pipeline.

Overall, this adaptation equates to 295 changes to the original *runCA* script and the patch has been made available on GitHub [3].

III. VERIFICATION AND TESTS

Verification on this adaption is carried out on an *Escherichia Coli* (E.coli) K12 MG1655 bacterium dataset [4]. The dataset consists of one SMRT Cell of data sequenced by P4-C2 PacBio RS II, with a library size of 20 kb. The assembly is run on Magnus, where each node has 24 Haswell cores and 64GB memory. In total it takes 333.6 core hours to complete and the breakdown of the runtime is in Table II.

Magnus Jobs	Celera Pipeline	CPUs	Time
job script	Setting up for trimming 0-Estimate mer threshold	24	00:03:29
ovl ecoli-trim[17]	0-Build overlap store for trimming	408	00:37:23
rCA ecoli-trim	0-Deduplicate and trim reads	24	00:00:59
job script	Setting up for assembly	24	00:01:35
ovl ecoli-assembly[4]	1-Build overlap store for assembly	96	00:19:57
rCA ecoli-assembly	Connecting	24	00:00:28
frg ecoli-assembly[1]	3-Fragment error correction	24	00:04:16
rCA ecoli-assembly	Connecting	24	00:00:24
ovc ecoli-assembly[11]	3-Overlap error correction	24	00:05:15
rCA ecoli-assembly	4-Construct unitigs	24	00:00:45
utg ecoli-assembly[2]	5-Compute unitig consensus	24	02:17:38
rCA ecoli-assembly	6-Compute insert sizes 7-Scaffolding with CGW	24	00:01:23
ctg ecoli-assembly[1]	8-Compute contig consensus	24	00:00:24
rCA ecoli-assembly	9-Terminate	24	00:01:20

Table II: Assembling E. Coli K12 with Celera on Magnus

The resulting assembly is then compared with the reference genome GI49175990 Ref NC000913.3 [5]. As is evident in Fig. 3, the Celera assembly (y-axis) matches the reference sequence (x-axis) (i.e. the two diagonal lines) over the entire range of the genome with one single contig (i.e. N50=4737053 which is very close to the reference size, 4641651). Note that there are two diagonal lines instead of one (the frame shift) and it's due to the fact that E. Coli chromosome is circular and the experimental assembly may start from a different point than the reference sequence.

With this success, the research group from Lions Eye Institute has now started assembling a larger data, tawny dragon lizard genome (1.96 Gb 0.5TB), on Magnus. Two steps have finished using 148 nodes and 214 nodes respectively.

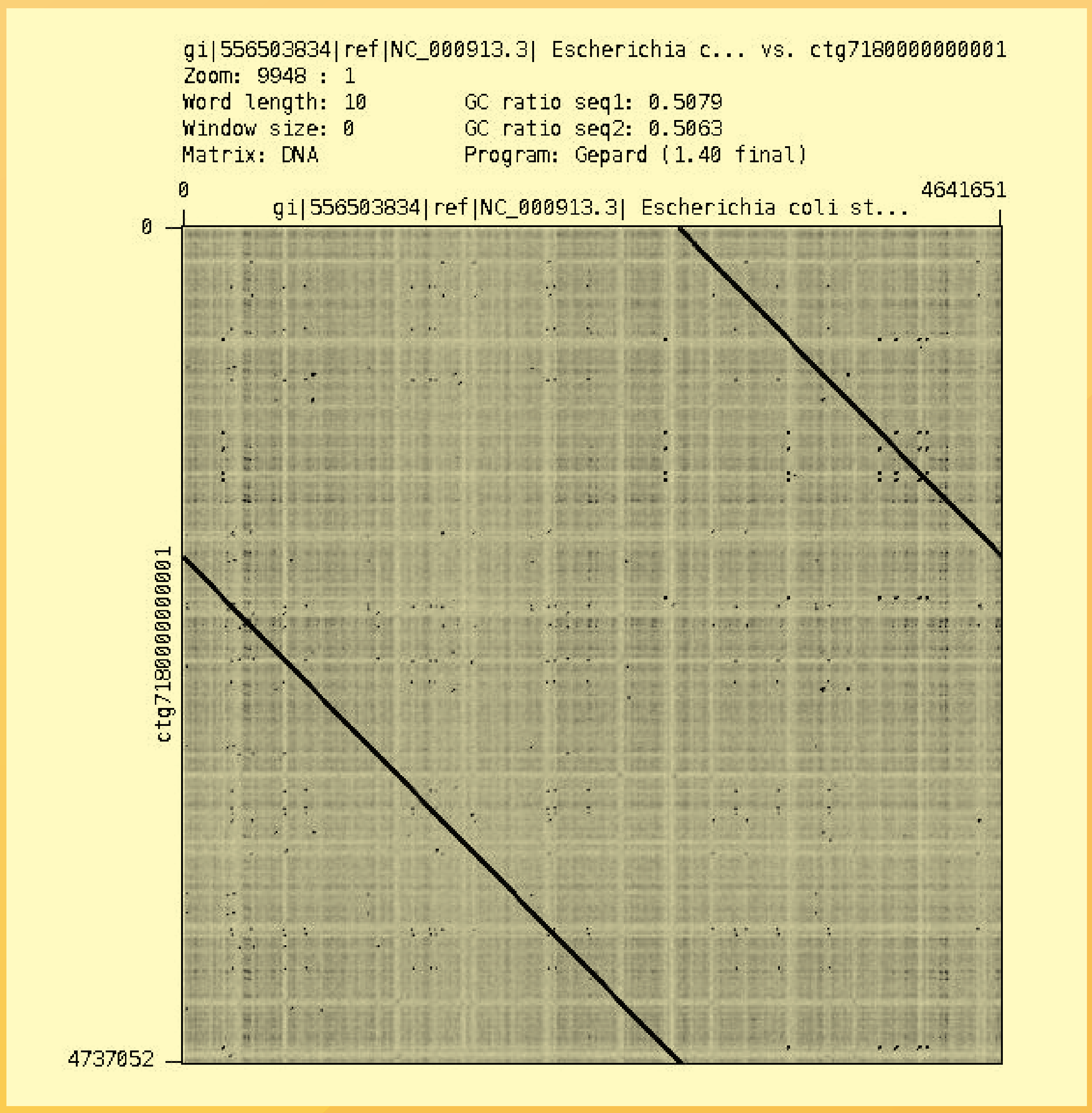


Fig. 3: Dotplot of Celera assembly against NCBI reference genome

IV. REMARKS

In summary, this case study has successfully adapted a complex data-driven workflow WGS-Celera to run on a Cray XC40 system where hybrid SLURM and ALPS scheduling is deployed. Different stages of the workflow have been able to dynamically request resources and distribute work based on work type and data size. Insights are offered for other similarly-structured workflows to be tuned to work in a hybrid scheduling environment at other leadership HPC sites.

Acknowledgements

This work is supported by Pawsey Supercomputing Centre in Western Australia through the use of advanced computing resources.

References

[1] http://docs.cray.com/books/S-2425-52xx.
[2] https://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/wgs-8.3.
[3] https://github.com/PawseySupercomputing/Celera-Assembler-Users.
[4] http://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/wgs-8.0/datasets.
[5] http://www.ncbi.nlm.nih.gov/nuccore/U00096.3.

GOVERNMENT OF WESTERN AUSTRALIA   Australian Government   NCRIS National Research Infrastructure for Australia An Australian Government Initiative   CSIRO   Curtin University   AUSTRALIA ECU EDITH COWAN UNIVERSITY   Murdoch UNIVERSITY   THE UNIVERSITY OF WESTERN AUSTRALIA   Lions Eye Institute