Decimate: a portable and fault-tolerant scheduler extension efficiently handling a large number of dependent jobs

Samuel KORTAS[a], Habib TOYE[b], Ibrahim HOTEIT[b,c]
(samuel.kortas, habib.toyemahamadoukele, ibrahim.hoteit)@kaust.edu.sa
[a]Kaust Supercomputing Laboratory, [b]Division of Computer, Electrical and Mathematical Sciences & Engineering,
[c]Division of Physical Science & Engineering

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

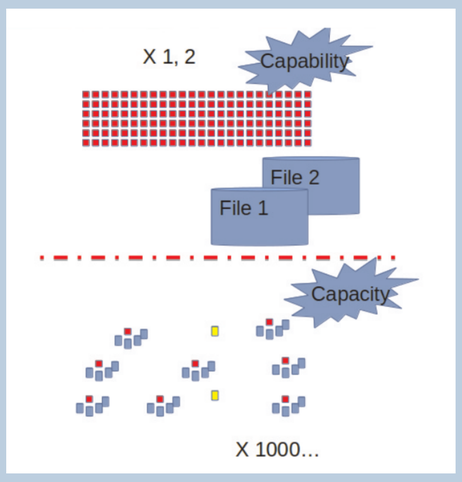
Better handling of capacity & capability jobs on Shaheen XC-40

Some of our scientists use KAUST Cray XC-40 Shaheen to explore parameters sweeping involving thousands of jobs saving thousands of temporary files. Most are non HPC experts but they need a result in a guaranteed time. These complex workflows producing a large number of small files cause challenging problems in terms of scheduling and file system stress.

Resources are configured/tuned for capability jobs, limited in number but running on a high number of nodes, and producing a small number of big files.

All the opposite of what our users' jobs need!

Limitation may come from scheduler not configured for 1000s of jobs, filesystem not tuned to support 1000s jobs each writing small files at scale hardware and numerical failures that have to be taken into account.



Our strategy:

- Pack 'many jobs' together to make them appear as big ones
- Reduce stress on filesystem by using Ramdisk and messages.
- Package these features as a SLURM scheduler extension easy to use and to install, distributed as an Open Source software.

A SLURM-extension available as a python module

Decimate extends the scheduling and executing environment allowing the user to:

- Submit arbitrary number of jobs regardless of limitation in the scheduling policy, manage jobs as a single workflow easing their submission, monitoring, deletion or reconfiguration against a large set of combinations of parameters.
- Benefit from a centralized log file.
- Check for correctness of the results of a job via a user-configurable shell or Python script and make a decision either to stop the whole workflow, to resubmit the failing components as is, or to modify it dynamically.

Once installed in a SLURM-scheduled environment as a regular python module via the command:

```
pip install decimate [ --user ] or conda install -c hpc4all decimate
```

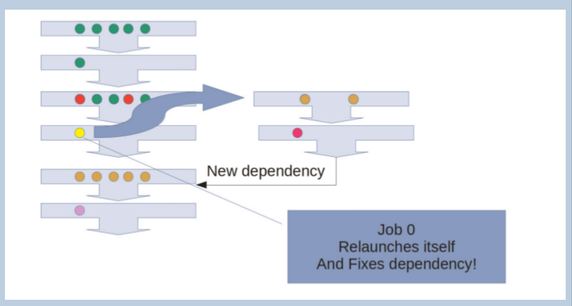
Decimate provides additional commands and parameters:

dbatch	submits the job (SLURM sbatch command with additional parameters handling fault-tolerance, and hidding limitation)	**dstat**	gives a detailed status of the jobs currently running, pending, waiting to be submitted or finished weither completed or returned with failure.
dlog	access to centralized log file for the whole workflow	**dkill**	terminates all job of the whole workflow

Decimate also provides corresponding Python methods for each of these shell commands.

Automated restart in case of failure

In case of failure of one part of the workflow, *Decimate* automatically detects the failure, signals it to the user and launches the misbehaving part after having fixed the job dependency.
By default if the same failure happens three consecutive times, *Decimate* cancels the whole workfow.




```
dbatch --job-name step1 --array 1-5 --check
--check-file check.sh --max-retry 3 step1.sh
```

```
dbatch --job-name step2 --depends step1 step2.sh
```

Example of checking script check.sh

```
echo $job_step -- $attempt -- $task_id
echo is_job_completed=$is_job_completed

grep 'job DONE' $output_file
```

Expected Results

`SUCCESS`: job went fine
`FAILED`: job needs to retart
`HALT`: the whole workflow failed

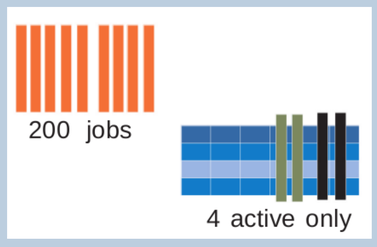
As the checking script is evaluated by the first job of the next step, it has knowledge of the status of the whole workflow. In case of failure, one can easily decide at that point either to resubmit the failed parts of the last step, either to stop the whole workflow or even to modify dynamically the environment before retempting a step.

Three extended execution modes

Sustained and automated feeding of the queue: *Decimate* guarantees that at the most a certain number of jobs appears in the queue at the same time (satisfying limitation in place). The remaining jobs are automatically submitted at the completion of the previous ones:

```
dbatch --max-jobs=4 --array=1-200 my_job.sh
```

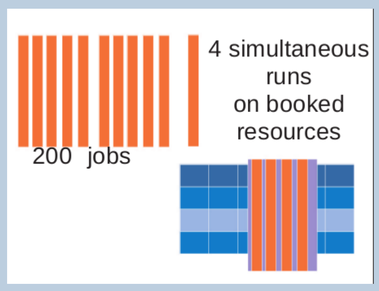
executes 200 jobs handled as arrays with at most 4 jobs appearing in the scheduling queue.



Execution on a pool of nodes: *Decimate* can transparently schedule all the jobs inside a given set of nodes booked for a longer duration period: 4 simultaneous runs on booked resources:

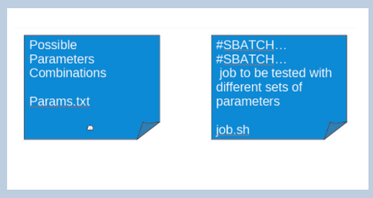
```
dbatch --parallel-runs=4 --array=1-200 my_job.sh
```

Books 4x3 nodes for 30 minutes and executes 4 jobs of 3 nodes simultaneously on these resources.



Parametric jobs support
Decimate computes every job possible, group them in blocks of similar footprint (nodes x time), submit them as separate arrays and manage them in a fault tolerant environment

```
dbatch --param--file=my_params.txt my_job.sh
```



History and current status

Decimate is the result of a continuous ongoing effort started in 2007 at Electricité de France to provide a portable framework capable of running hundreds of scenarios on a user workstation and transparently scaling to thousands of possibilities explored on the company's Supercomputer.

As the main developper moved to KAUST, the need for a robust and fault-tolerant environment to run 1000 simultaneous forcasts lead to its complete rewritting and packaging as a scheduler extension available as a python module and distributed under Open Source License.

Development and maintenance of *Decimate* are currently supported by KAUST Supercomputing Core Laboratory

Perspectives

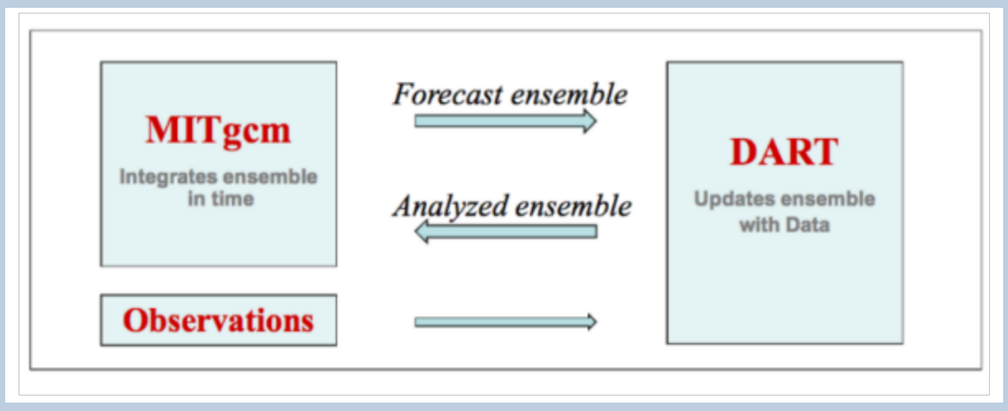
- Runnning an Ocean Ensemble data Assimilation with even more members.
- Comparison with other tools... To our knowledge, none of them has a shell interface.
- Support of other scheduler: PBS.
- Support of no-scheduler mode, running on a regular workstation.

more infos, white papers, sources, technical documentation...

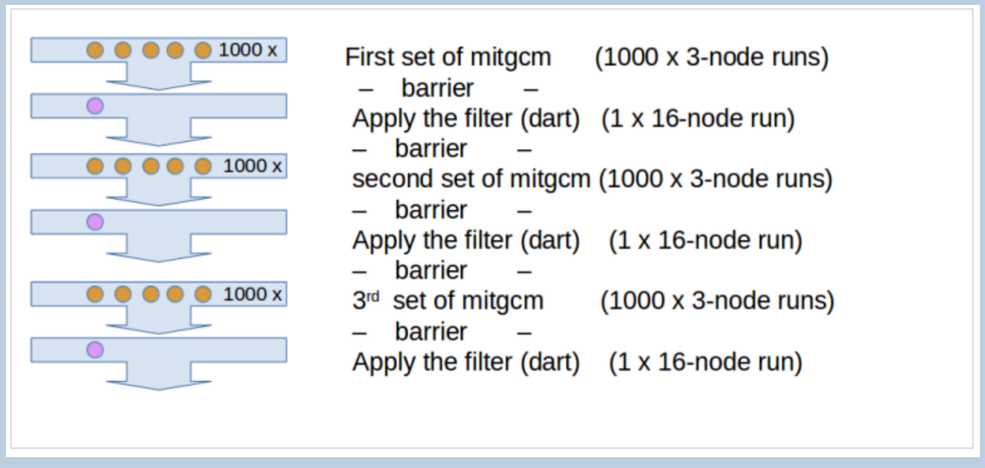
http://decimate.hpc4all.org

An Ocean Ensemble data Assimilation challenging use case

DART-MITgcm assimilation system combines MITgcm forecast ensemble simulations with observational data to compute the best possible estimate of the state of the ocean through DART Ensemble Kalman filters. It also quantifies uncertainties in the final solution (often described by the covariance matrix of the state estimate), derived from the knowledge of the prior uncertainties and the observational data errors covariance.

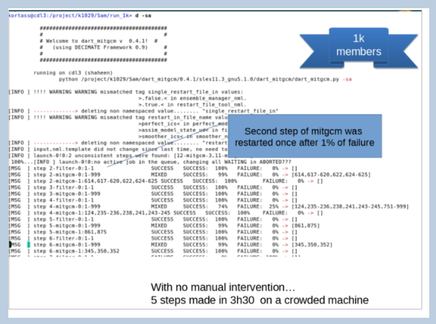


A workflow of 130,000 jobs, running 1000 MITgcm members on 3 Haswell 32-core nodes followed by one DART filter running on one node.

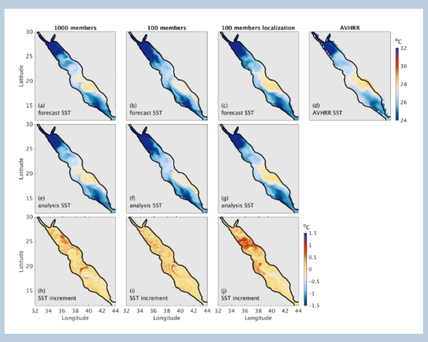


Results

Decimate allowed to alleviate the burden of constantly having to monitor the ongoing workflow and manually acting in case of numerical or hardware failure. A rough estimation was reporting that 20% of restitution time was effectively consumed into these manual operations.



Valuable scientific results were published running the assimilation model on 1000 members. [1]



Decimate is now extensively used at KAUST in Computational Chemistry, Uncertainty Quantification, or to explore any sets of parameters.

References

[1] A Fault-Tolerant HPC Scheduler Extension for Large and Operational Ensemble Data Assimilation: Application to the Red Sea, Toye H., Kortas S., Zhan P., Hoteit I., Journal of Computational Science 27 (2018) 46-56
[2] http://decimate.readthedocs.org

Acknowledgements

Research exposed in this poster was supported by King Abdullah University of Science and Technology (KAUST), and made use of the resources of the KAUST Supercomputing Core Laboratory.