Decimate: a portable and fault-tolerant scheduler.

Application to big ensembles data assimilation and forecasting in the Red Sea Circulation

Samuel Kortas, Habib Toye, Peng Zhan, Ibrahim Hoteit

KAUST

IXUPG Middle East Conference 2018

As core number grows

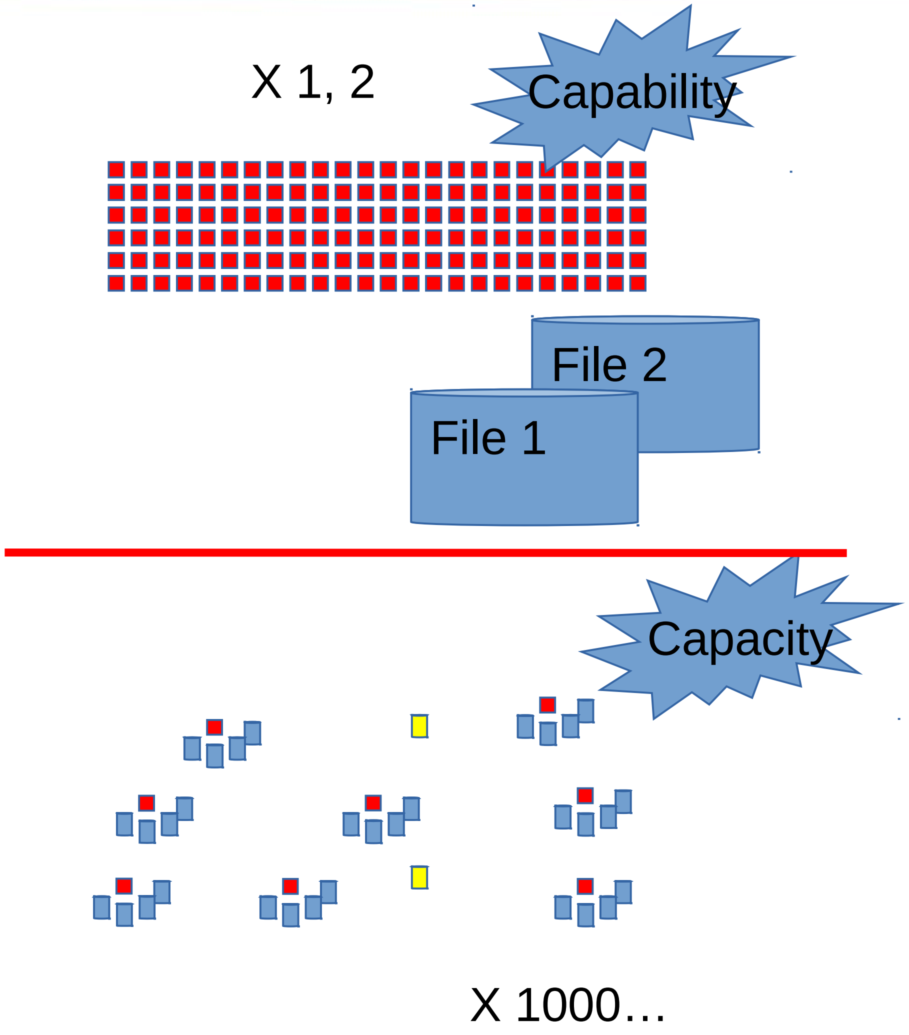
Launch of thousands of jobs became an option...

- Some of our users use Shaheen to run workflows composed of thousands of jobs saving thousands of temporary files
- Need a result in a guaranteed time
- Are not HPC experts, but are challenging problem in terms of scheduling and file system stress

Why is it so challenging?

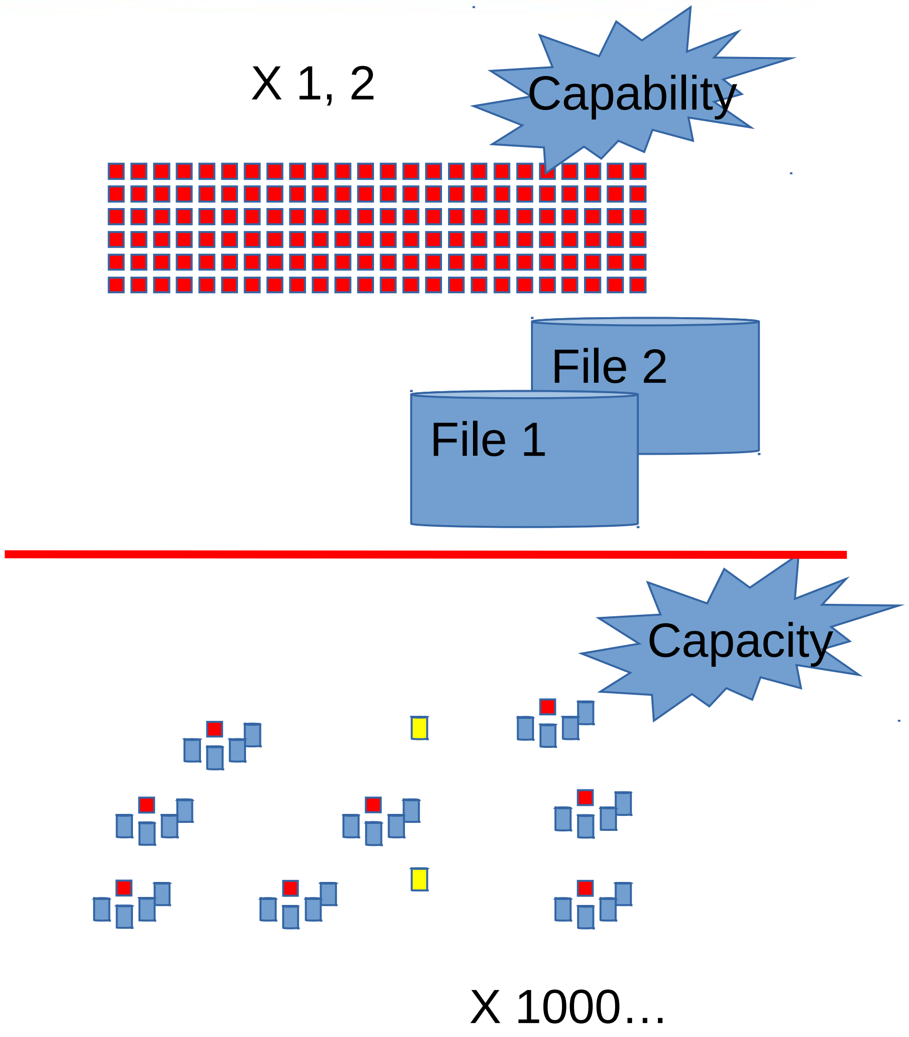
A difference of nature

- Resources are Shaped/Tuned for one of these two cases:
  - Capability (big job, big files)
  - Capacity (numerous jobs, small files)



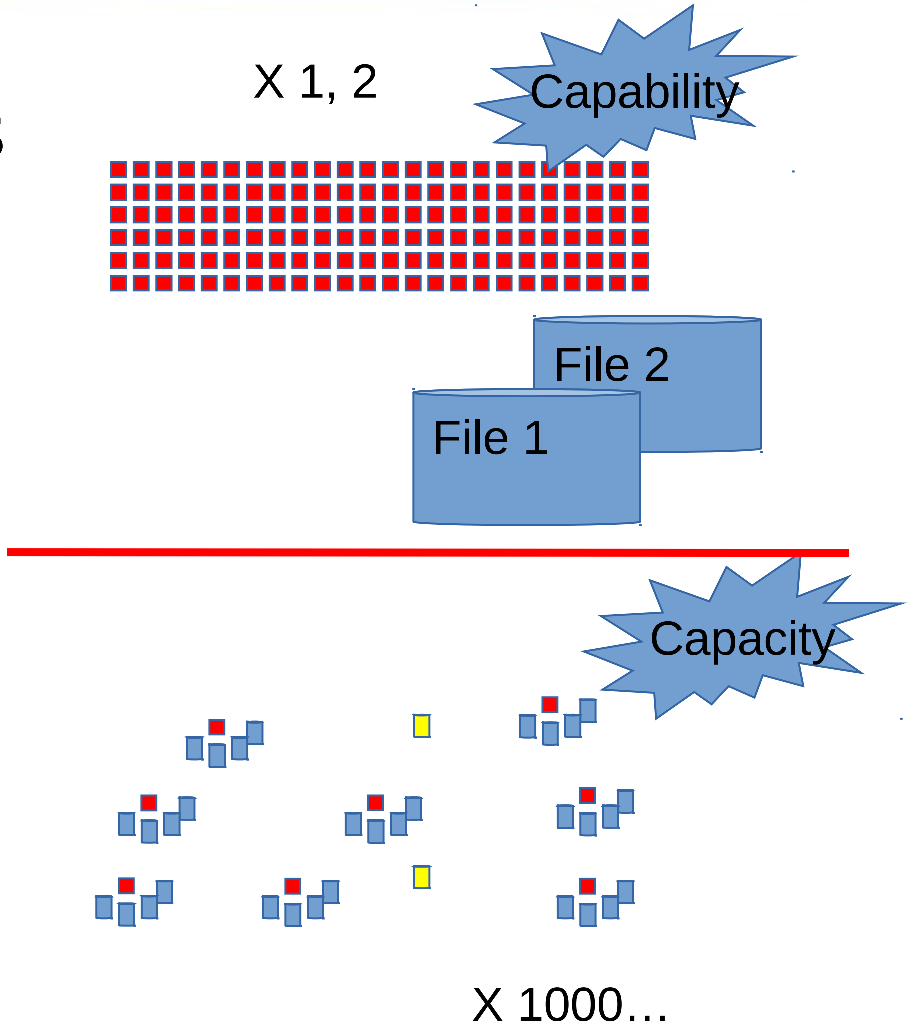
Our Strategy (1/2)

- Pack 'many jobs' together to make them appear as big ones
- Reduce stress on filesystem by using Ramdisk and messages.



Our Strategy (2/2)

- Extend the scheduler with tools transparent to users
- Allow them to tune easily part of workflow with parametric sweep on number of threads



What is Decimate? (1/2)
Swiss knife of many-jobs workflows

Makes It easier to

- Generate
- Submit
- balance
- Monitor
- Control
- Cure
- Check
- Terminate

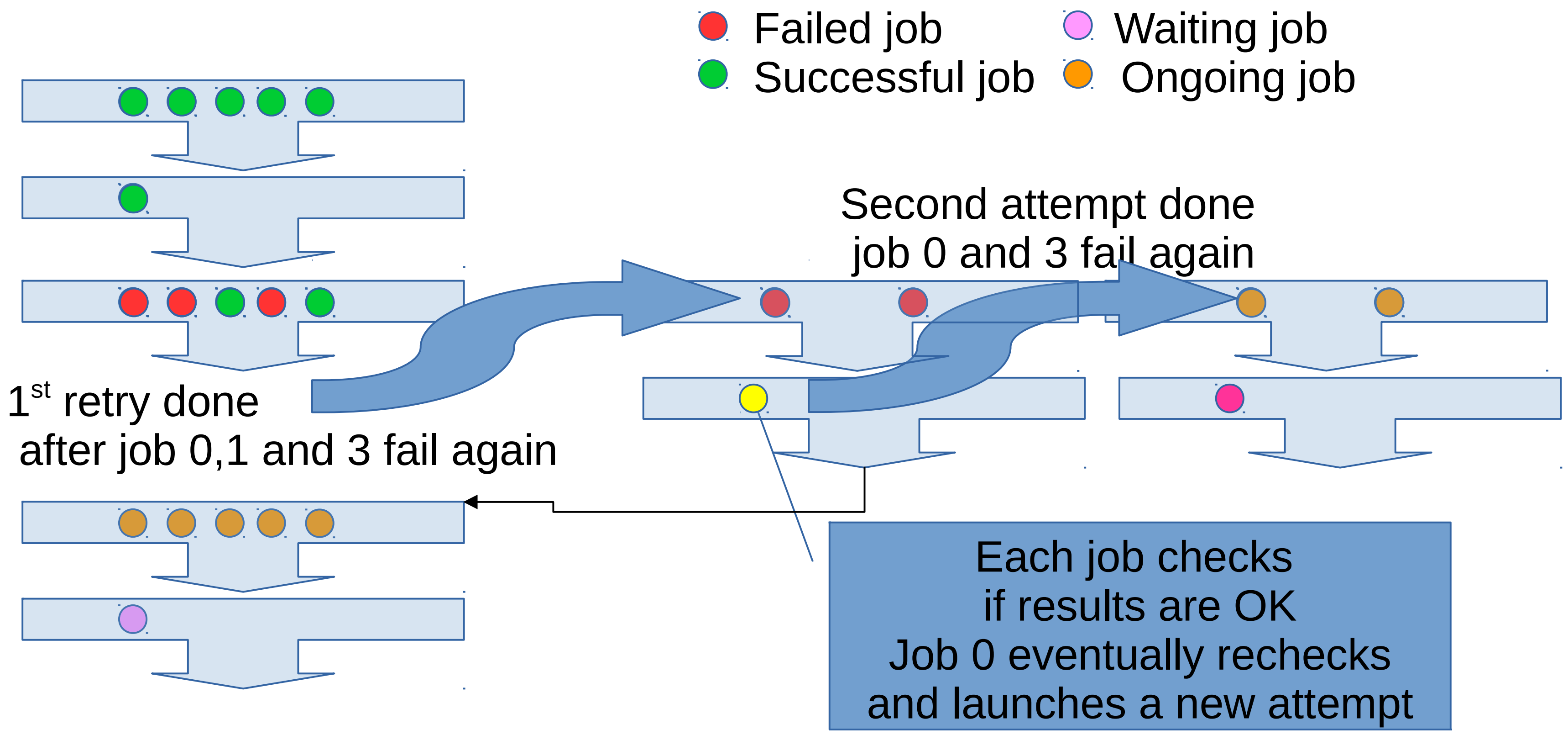
- Many-jobs
- Dependent
- Parametric
- Unstable

Workflows

What is Decimate? (2/2)
A Scheduler extension easy to install

- A scheduler extension (SLURM) written in python 2.7
  fully documented at http://decimate.readthedocs.org
- Installable with no special priviledges:
  pip install decimate   or   conda install -c hpc4all decimate
- Available as an open source software at
  https://github.com/KAUST-KSL/ (FreeBSD License)
- Installed on Shaheen II, Ibex, portable on any machine with SLURM
- Under development but already used heavily but some of our users.
  250 kjobs submitted in production via *Decimate*
- Maintained by KSL (samuel.kortas (at) kaust.edu.sa)

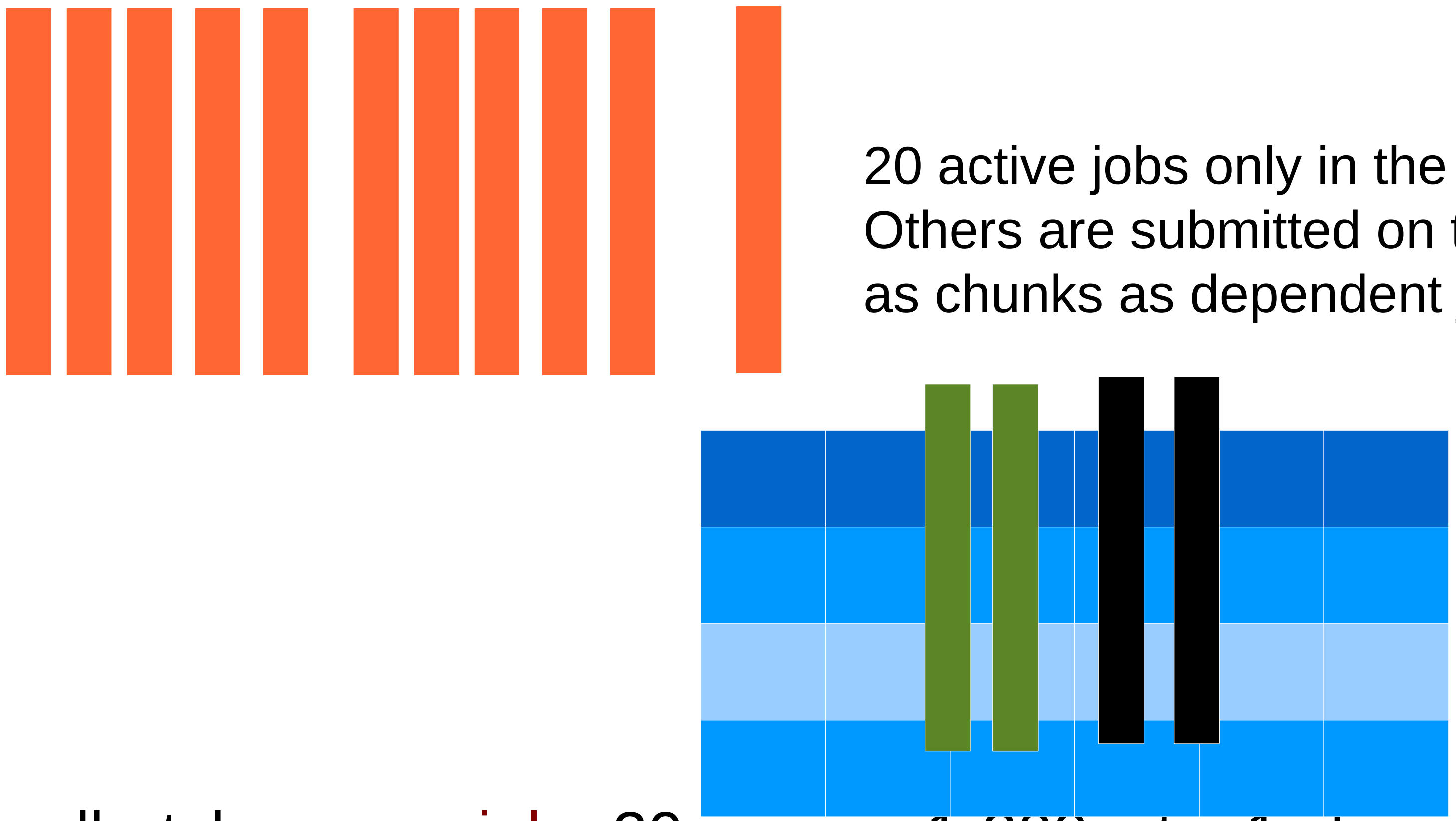
Decimate's Features (1/4)
Automated restart in case of failure



```
dbatch   --array 1-5 --check check1.sh --max-retry 3  step1.sh
```

Decimate's Features (2/4)
Automated feeding of the job queue

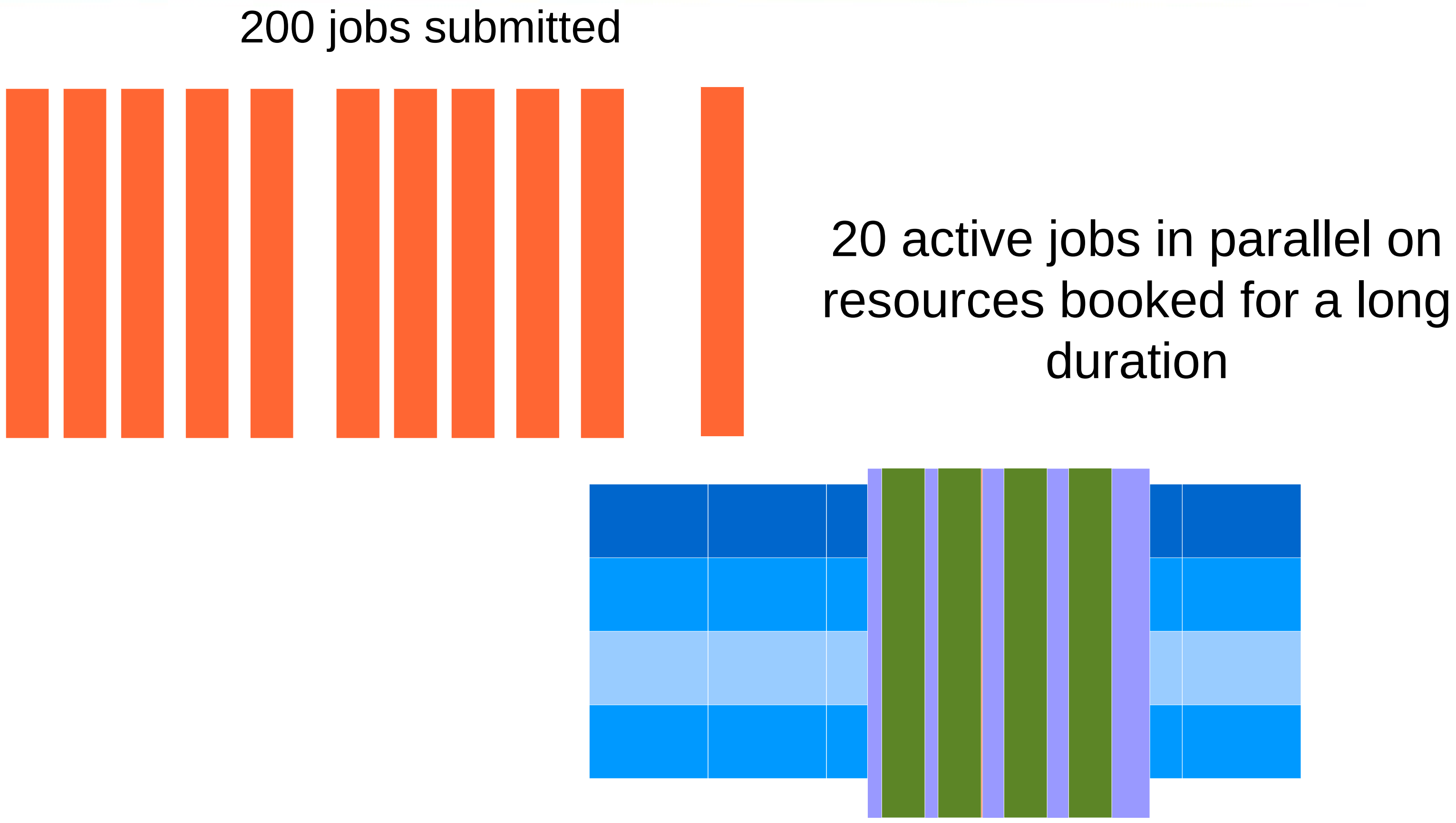
200 jobs submitted



20 active jobs only in the queue
Others are submitted on the fly
as chunks as dependent jobs

dbatch –max-jobs 20 --array 1-200 step1.sh

Decimate's Features (3/4)
Execution on a pool of nodes


200 jobs submitted
20 active jobs in parallel on resources booked for a long duration

dbatch –parallel-runs 20 --array 1-200 --nodes=3 step1.sh

Decimate's Features (4/4)
Extended support of parametrized jobs

params.txt

```
nodes ntasks
   1        32
   1        16
   2        64
   4        128
#DECIM nthreads=nodes*32/ntasks
#DECIM COMBINE dim =[10,100,1000]
```

my_job.sh

```
#SBATCH…

export OMP_NUM_THREADS=$nthreads
srun -N $nodes -n $ntasks \
        --cpu-per-tasks=$nthreads \
              my_app.exe $dim
```

```
dbatch   –param-file params.txt my_job.sh
```

Decimate computes every job possible, group them in blocks, submit them and manage them in a fault tolerant environment

Use case from Ocean Modeling

PI: Pr Ibrahim Hoteit (PSE) & Habib Toye

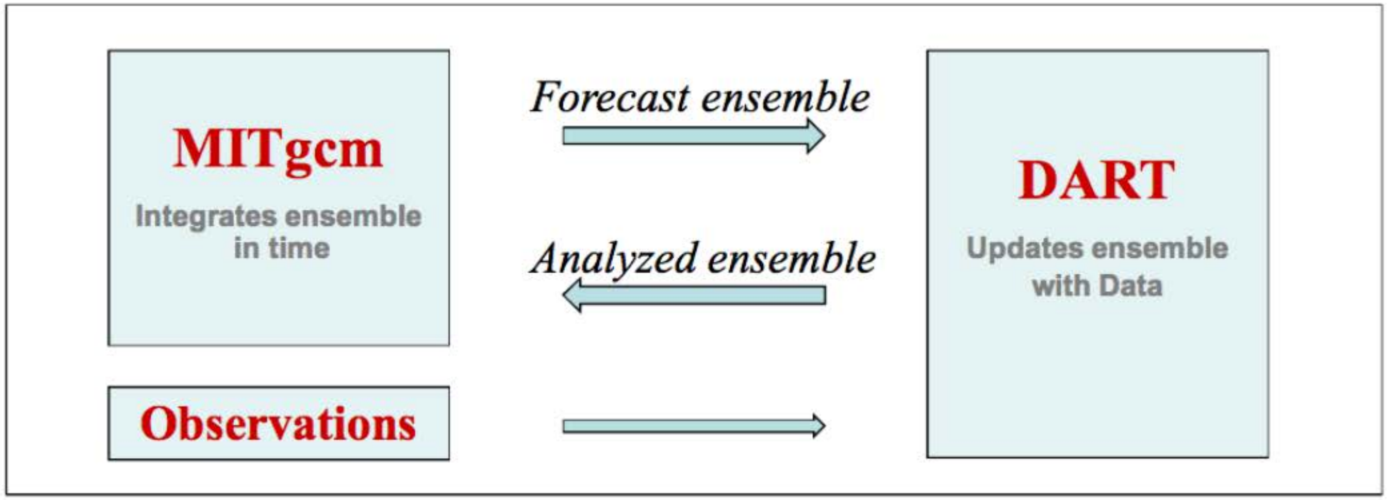


Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model

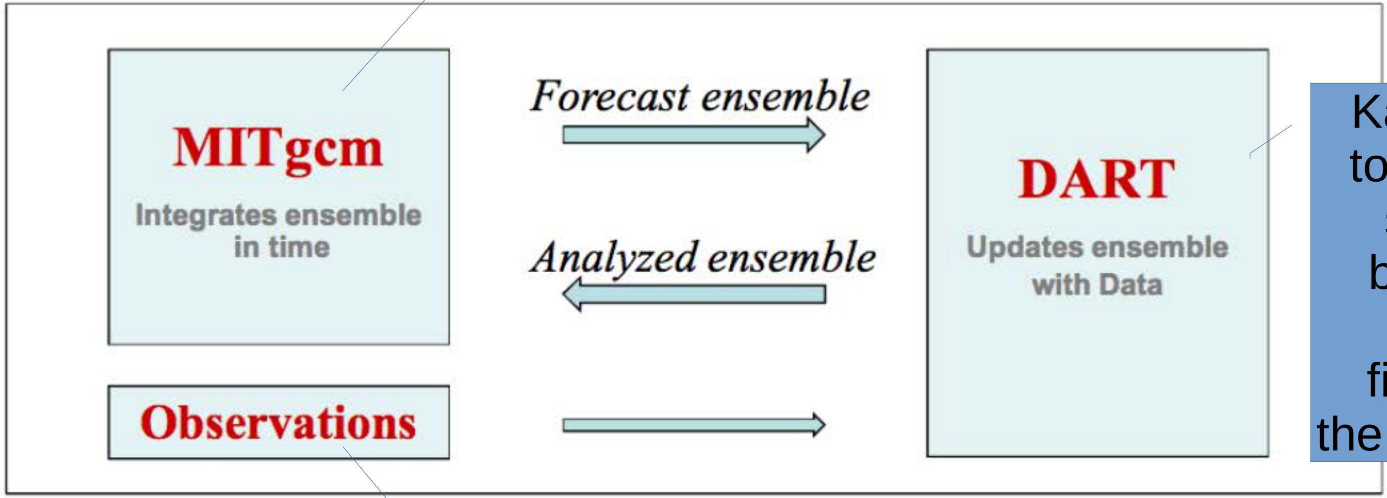


Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

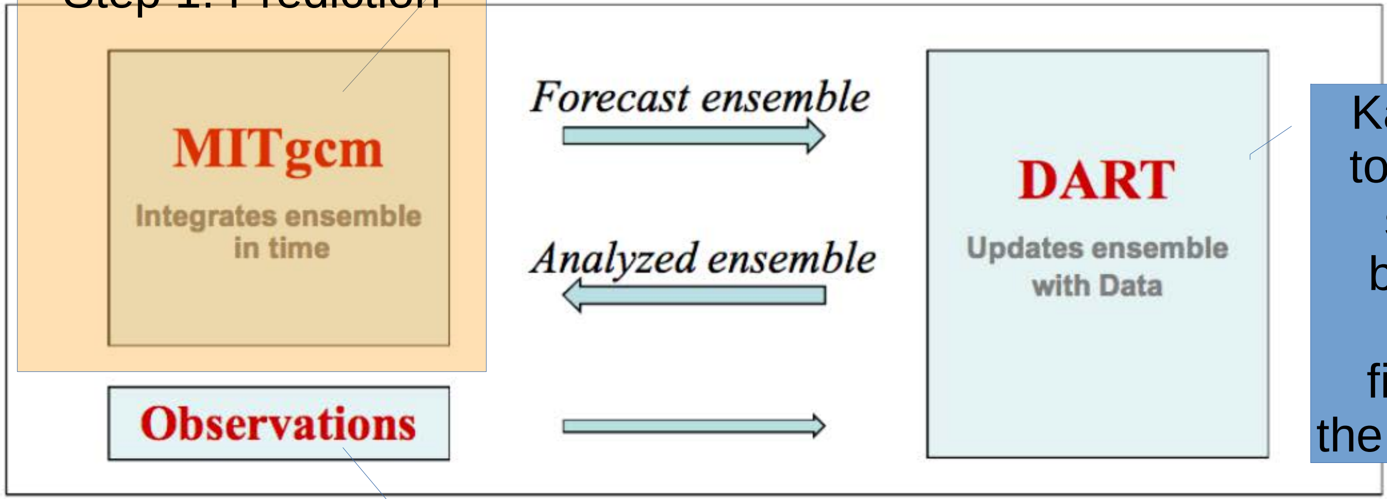
Kalman Filter to reroute the simulation back on the right track fitting better the observations

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model

Step 1: Prediction



Kalman Filter to reroute the simulation back on the right track fitting better the observations

Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model

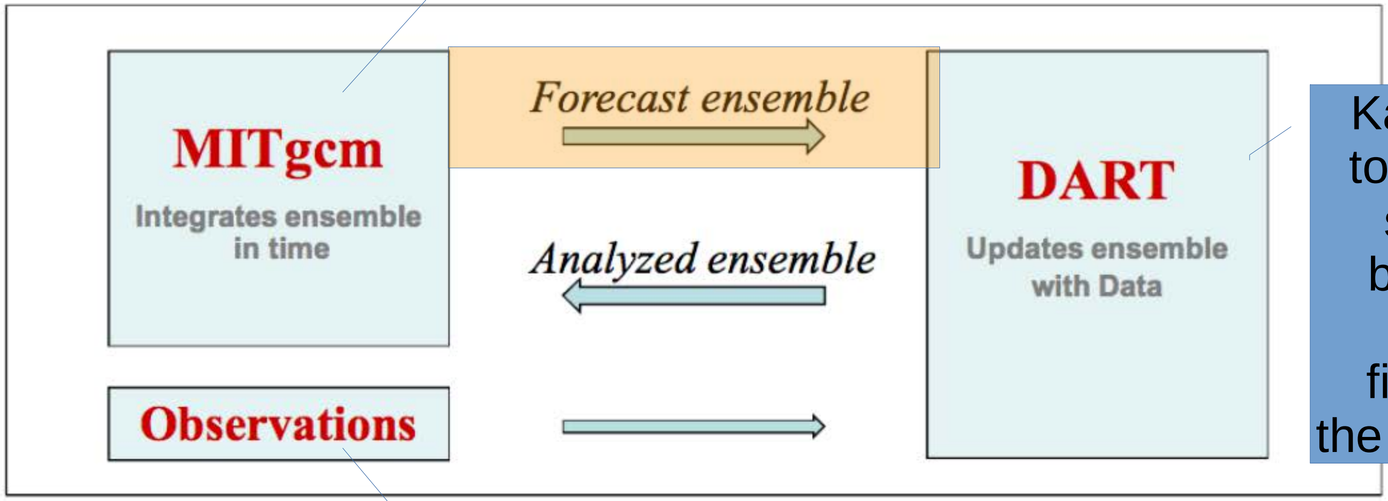


Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Kalman Filter to reroute the simulation back on the right track fitting better the observations

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

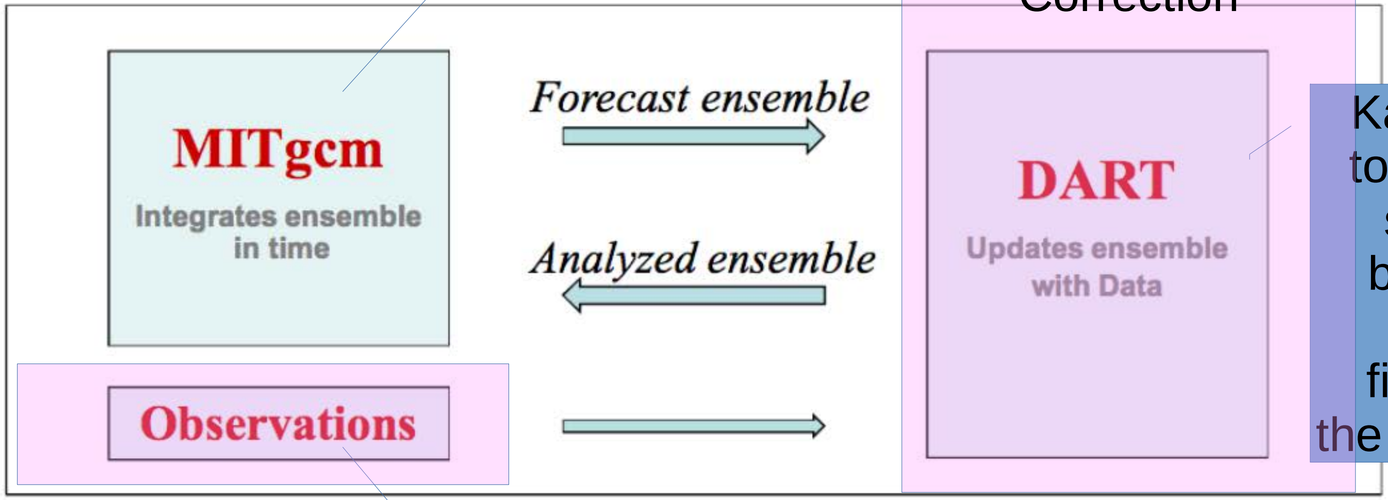


Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Ocean model

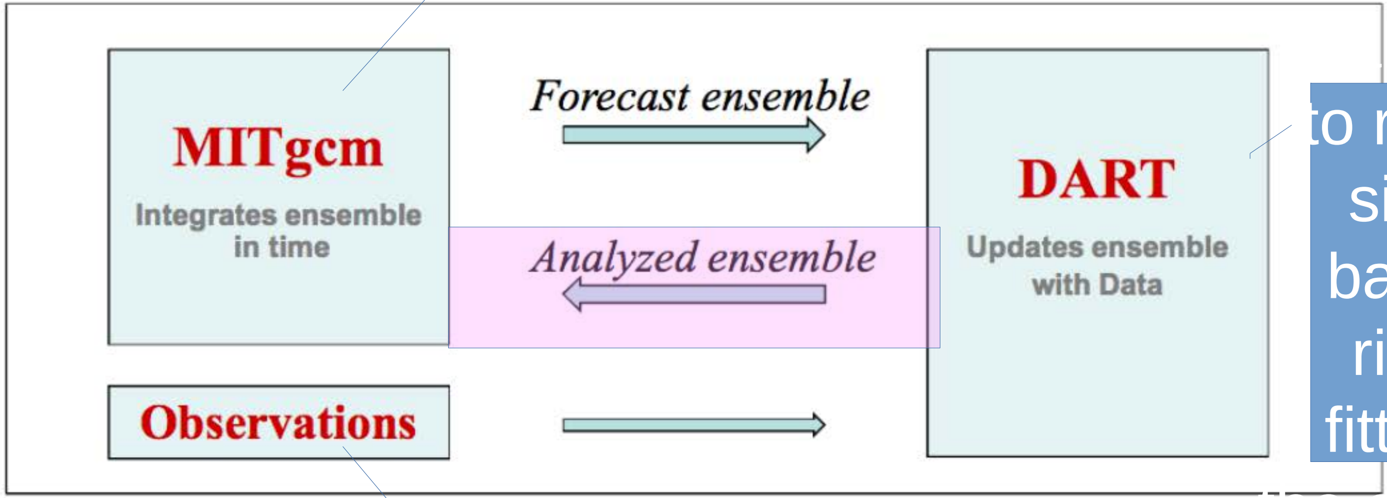
Step 2: Filter & Correction

Kalman Filter to reroute the simulation back on the right track fitting better the observations

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model



to reroute the simulation back on the right track fitting better

Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

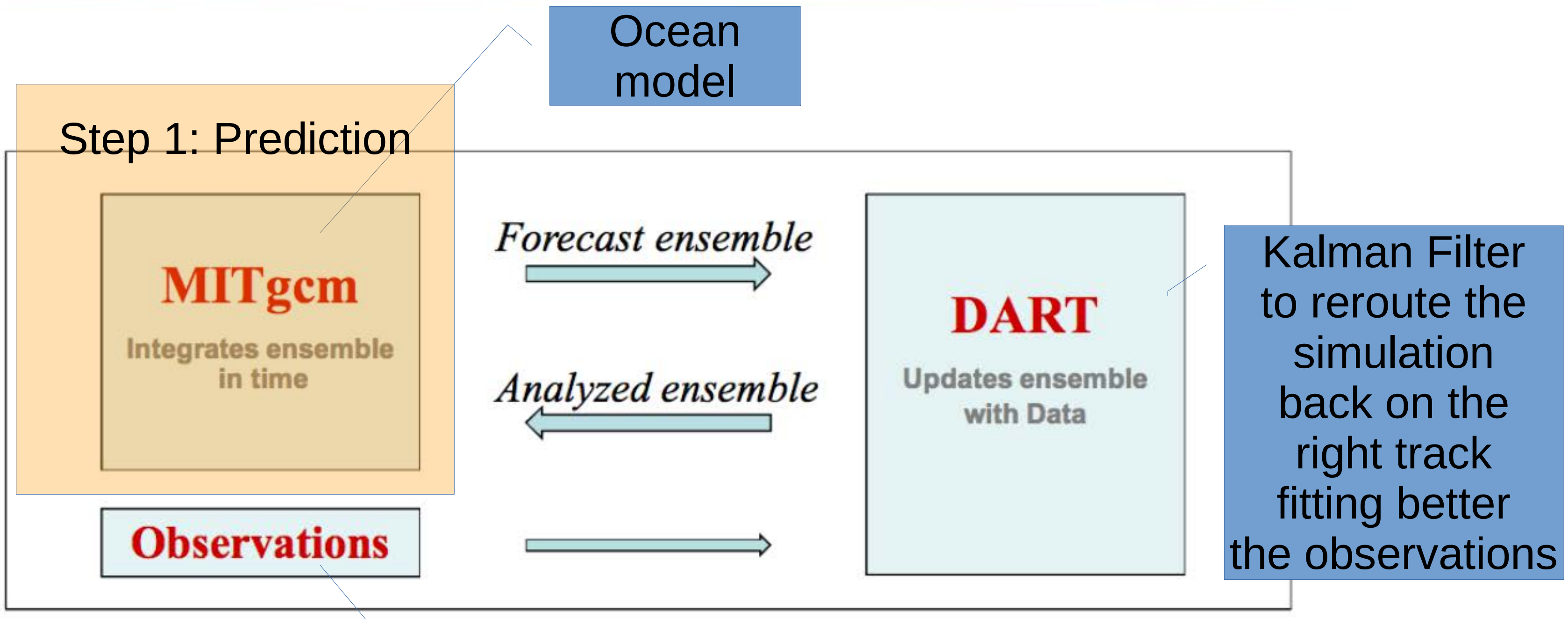

Ocean model
Step 1: Prediction
MITgcm
Integrates ensemble in time
Forecast ensemble
Analyzed ensemble
DART
Updates ensemble with Data
Observations
Kalman Filter to reroute the simulation back on the right track fitting better the observations

Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model

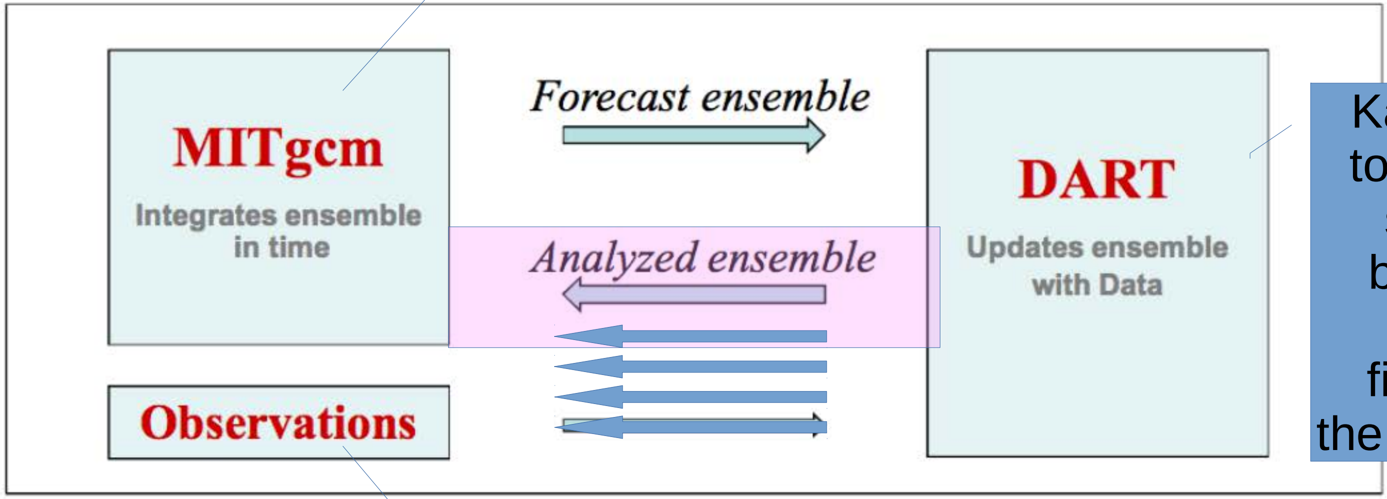


Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

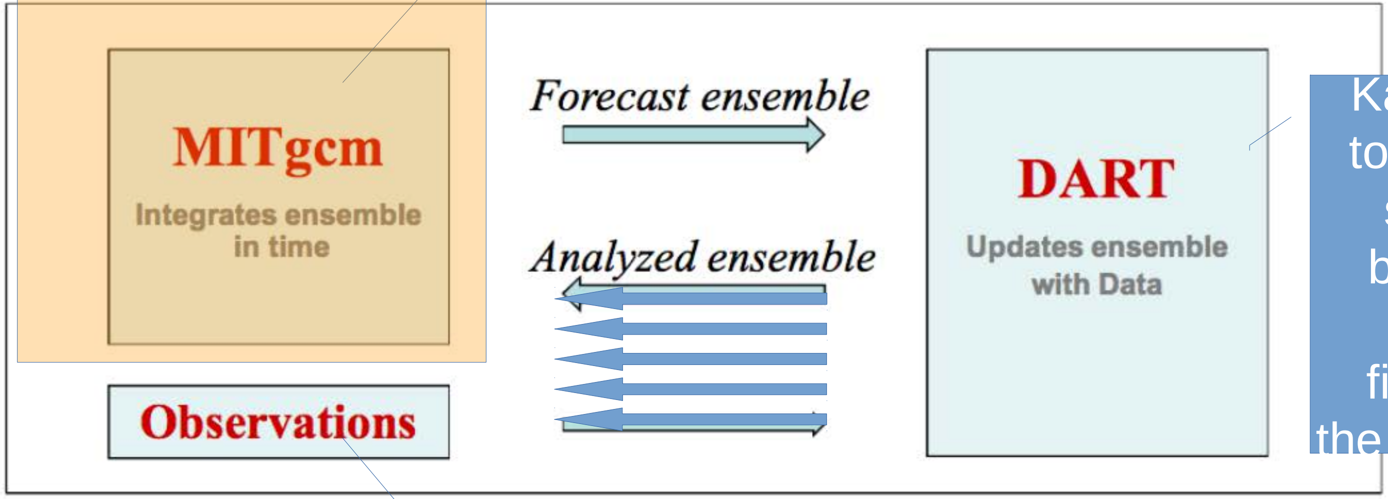
Kalman Filter to reroute the simulation back on the right track fitting better the observations

Observations taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model

Step 1: Prediction



Kalman Filter to reroute the simulation back on the right track fitting better the observations

Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Observations
taken from satelites

Some example of workflow (2/2) MITgcm/DART assimilation system

Ocean model

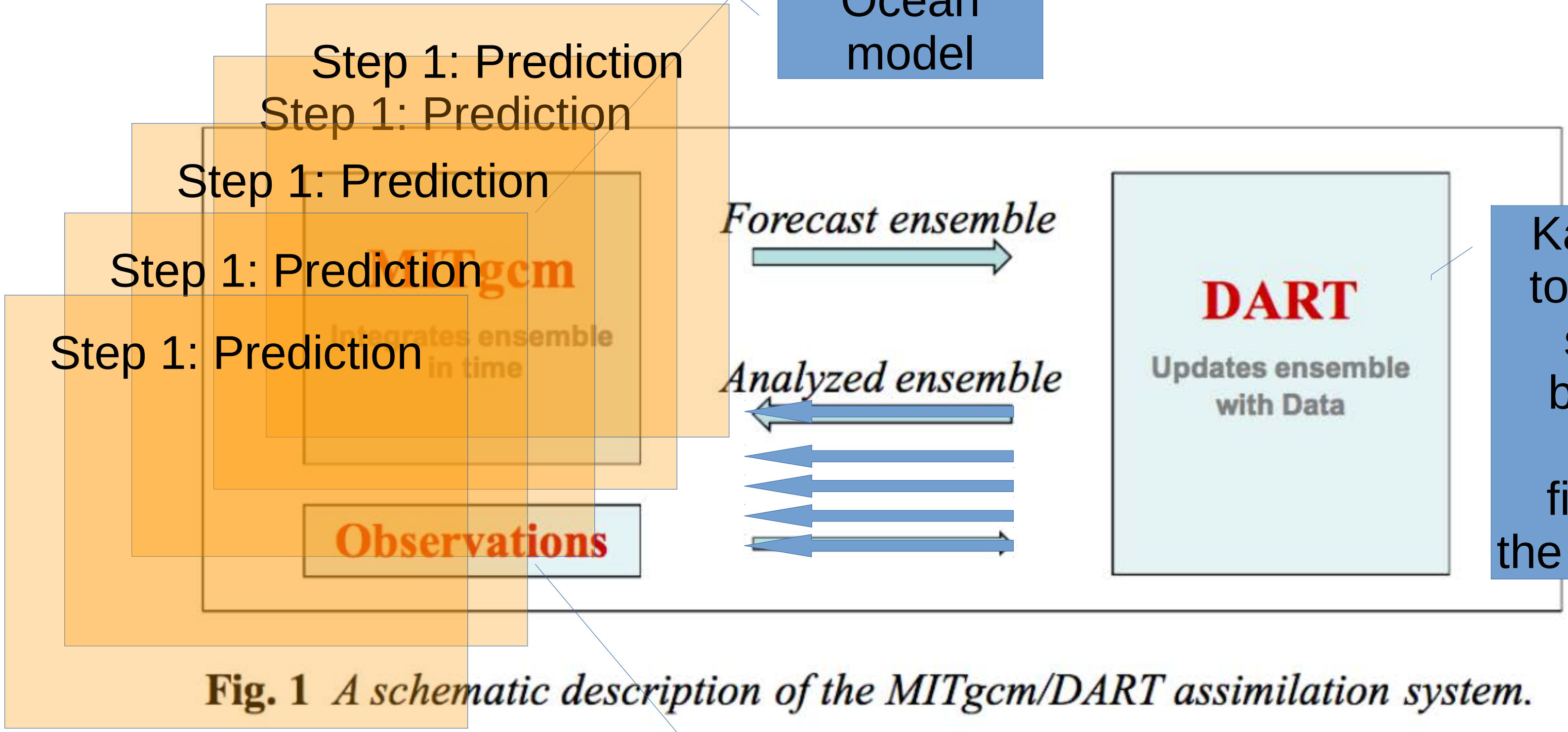
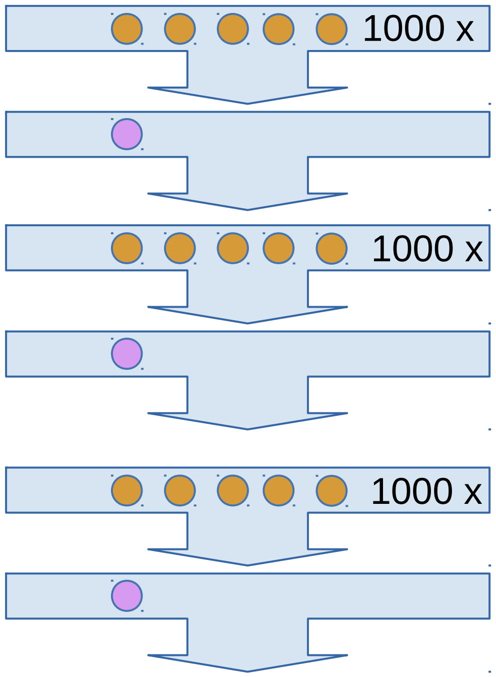


Fig. 1 *A schematic description of the MITgcm/DART assimilation system.*

Kalman Filter to reroute the simulation back on the right track fitting better the observations

Observations taken from satelites

Typical MITgcm/DART assimilation workflow

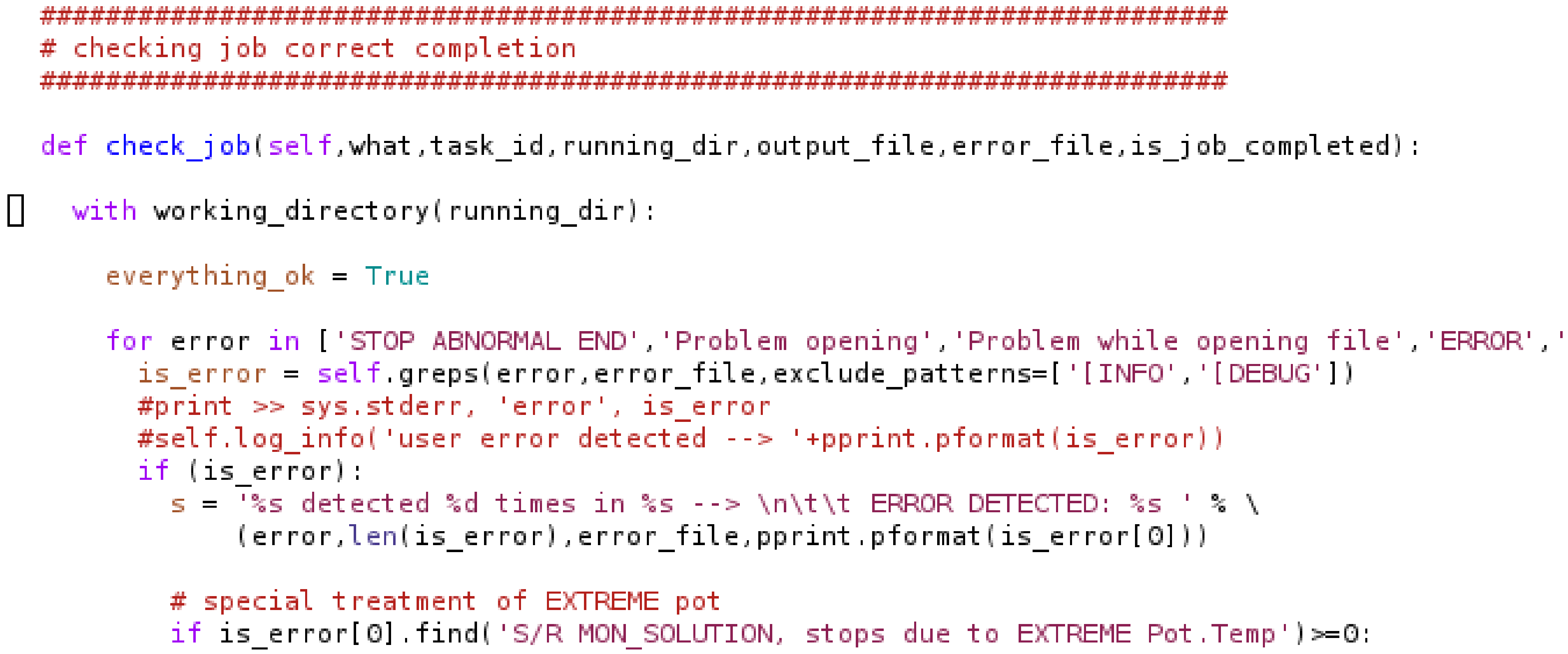


First set of MITgcm (1000 x 3-node runs)
– barrier –
Apply the filter (DART) (1 x 16-node run)
– barrier –
second set of MITgcm (1000 x 3-node runs)
– barrier –
Apply the filter (DART) (1 x 16-node run)
– barrier –
3rd set of MITgcm (1000 x 3-node runs)
– barrier –
Apply the filter (DART) (1 x 16-node run)

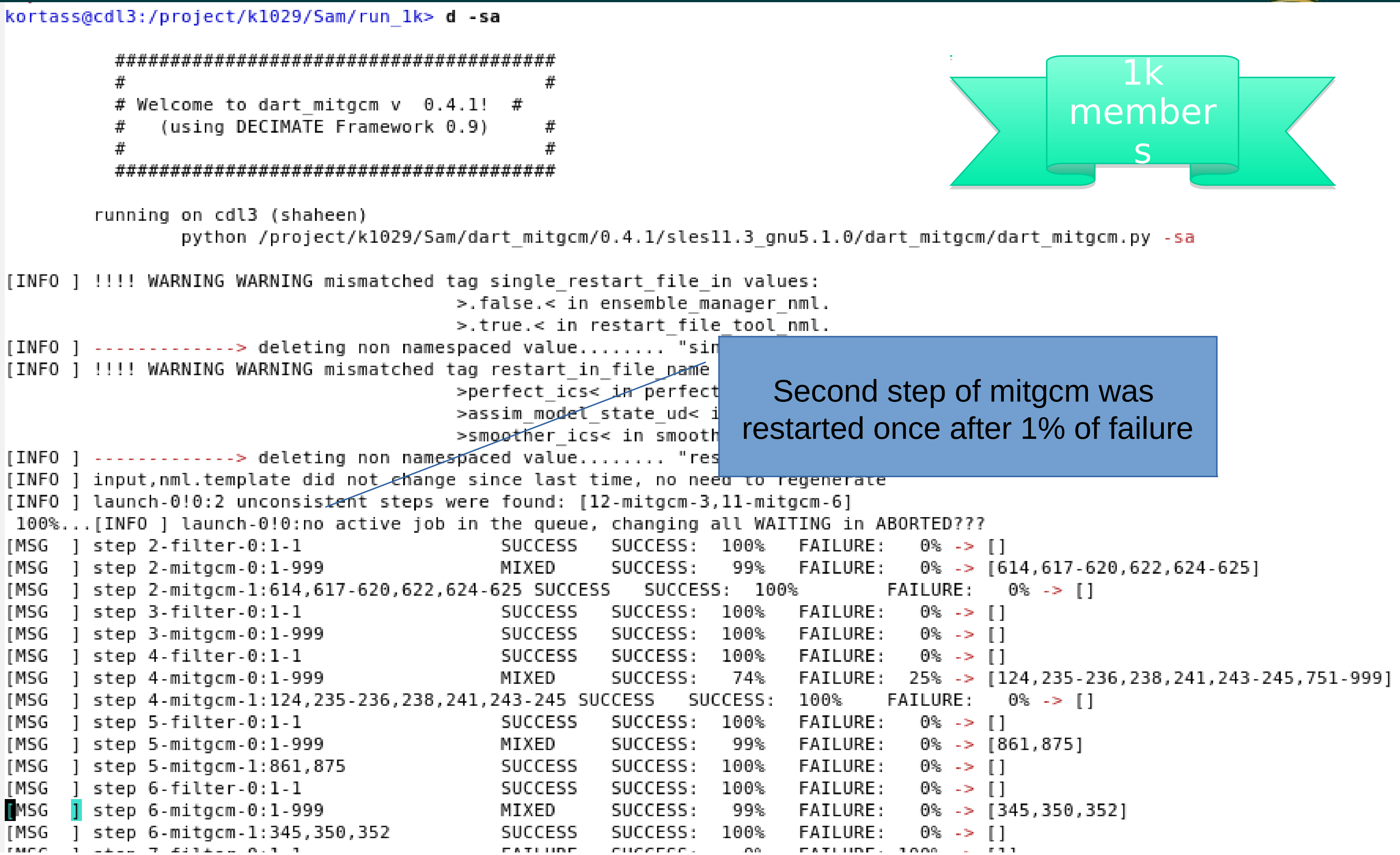
- Some intermediate steps may break
  → dependency will break
  → the workflow will remain idle,

Decimate is the solution! user-define result checking function

- the user can set his own rules thanks to a python function.



```
##########################################################################
# checking job correct completion
##########################################################################

def check_job(self,what,task_id,running_dir,output_file,error_file,is_job_completed):

  with working_directory(running_dir):

    everything_ok = True

    for error in ['STOP ABNORMAL END','Problem opening','Problem while opening file','ERROR','
      is_error = self.greps(error,error_file,exclude_patterns=['[INFO','[DEBUG'])
      #print >> sys.stderr, 'error', is_error
      #self.log_info('user error detected --> '+pprint.pformat(is_error))
      if (is_error):
        s = '%s detected %d times in %s --> \n\t\t ERROR DETECTED: %s ' % \
            (error,len(is_error),error_file,pprint.pformat(is_error[0]))

        # special treatment of EXTREME pot
        if is_error[0].find('S/R MON_SOLUTION, stops due to EXTREME Pot.Temp')>=0:
```



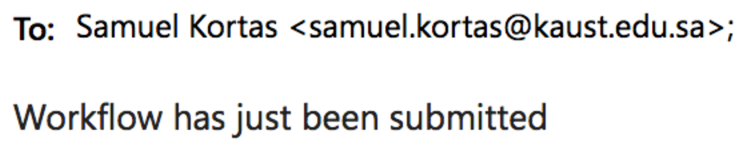
```
kortass@cdl3:/project/k1029/Sam/run_1k> d -sa

          ########################################
          #                                      #
          # Welcome to dart_mitgcm v  0.4.1!  #
          #   (using DECIMATE Framework 0.9)     #
          #                                      #
          ########################################

        running on cdl3 (shaheen)
                python /project/k1029/Sam/dart_mitgcm/0.4.1/sles11.3_gnu5.1.0/dart_mitgcm/dart_mitgcm.py -sa

[INFO ] !!!! WARNING WARNING mismatched tag single_restart_file_in values:
                                        >.false.< in ensemble_manager_nml.
                                        >.true.< in restart_file_tool_nml.
[INFO ] -------------> deleting non namespaced value........ "sin
[INFO ] !!!! WARNING WARNING mismatched tag restart_in_file_name
                                        >perfect_ics< in perfect
                                        >assim_model_state_ud< i
                                        >smoother_ics< in smooth
[INFO ] -------------> deleting non namespaced value........ "res
[INFO ] input,nml.template did not change since last time, no need to regenerate
[INFO ] launch-0!0:2 unconsistent steps were found: [12-mitgcm-3,11-mitgcm-6]
 100%...[INFO ] launch-0!0:no active job in the queue, changing all WAITING in ABORTED???
[MSG  ] step 2-filter-0:1-1                    SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 2-mitgcm-0:1-999                  MIXED     SUCCESS:   99%   FAILURE:   0% -> [614,617-620,622,624-625]
[MSG  ] step 2-mitgcm-1:614,617-620,622,624-625 SUCCESS   SUCCESS:  100%        FAILURE:   0% -> []
[MSG  ] step 3-filter-0:1-1                    SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 3-mitgcm-0:1-999                  SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 4-filter-0:1-1                    SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 4-mitgcm-0:1-999                  MIXED     SUCCESS:   74%   FAILURE:  25% -> [124,235-236,238,241,243-245,751-999]
[MSG  ] step 4-mitgcm-1:124,235-236,238,241,243-245 SUCCESS   SUCCESS:  100%    FAILURE:   0% -> []
[MSG  ] step 5-filter-0:1-1                    SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 5-mitgcm-0:1-999                  MIXED     SUCCESS:   99%   FAILURE:   0% -> [861,875]
[MSG  ] step 5-mitgcm-1:861,875                SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 6-filter-0:1-1                    SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 6-mitgcm-0:1-999                  MIXED     SUCCESS:   99%   FAILURE:   0% -> [345,350,352]
[MSG  ] step 6-mitgcm-1:345,350,352            SUCCESS   SUCCESS:  100%   FAILURE:   0% -> []
[MSG  ] step 7-filter-0:1-1                    FAILURE   SUCCESS:    0%   FAILURE: 100% -> [1]
```

1k members

Second step of mitgcm was restarted once after 1% of failure

With no manual intervention…
5 steps made in 3h30 on a crowded machine

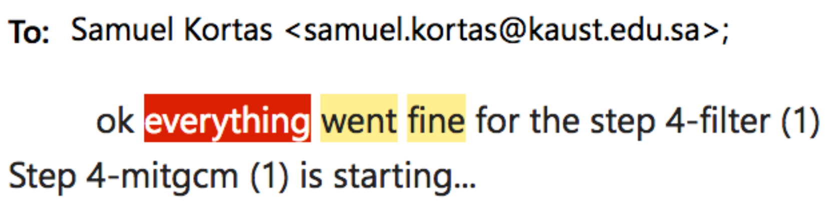
Example of Mails sent by Decimate



To: Samuel Kortas <samuel.kortas@kaust.edu.sa>;

Workflow has just been submitted

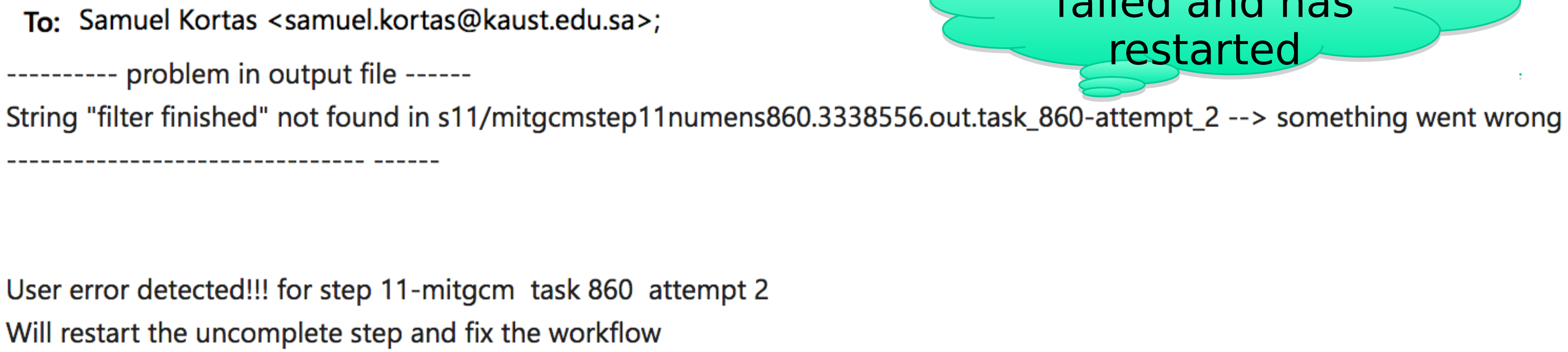
Job just started



To: Samuel Kortas <samuel.kortas@kaust.edu.sa>;

ok everything went fine for the step 4-filter (1)
Step 4-mitgcm (1) is starting...

Step is successful. Moving further!



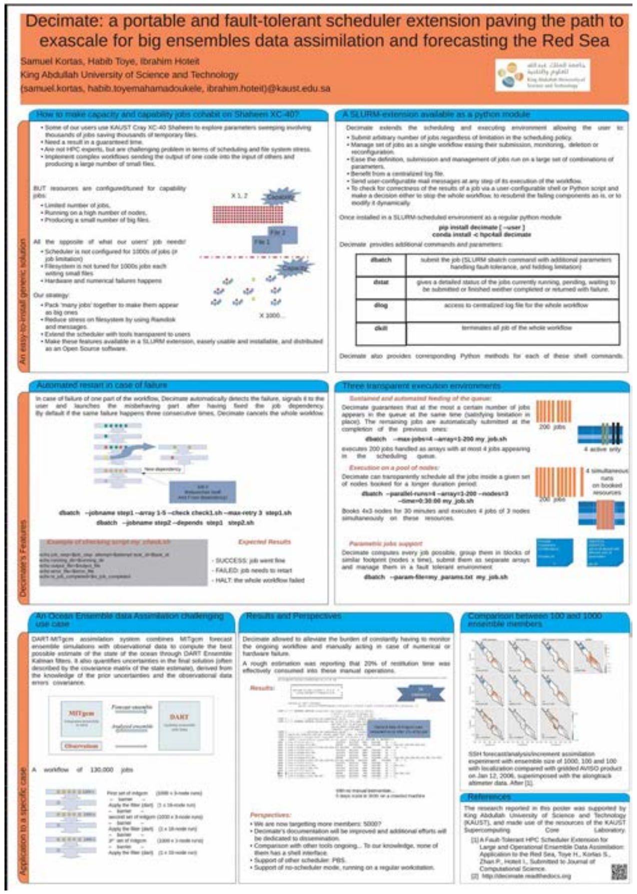
To: Samuel Kortas <samuel.kortas@kaust.edu.sa>;

```
---------- problem in output file ------
String "filter finished" not found in s11/mitgcmstep11numens860.3338556.out.task_860-attempt_2 --> something went wrong
-------------------------------- ------
```

User error detected!!! for step 11-mitgcm task 860 attempt 2
Will restart the uncomplete step and fix the workflow

Previous step failed and has restarted

Want to know more?



Decimate: a portable and fault-tolerant scheduler extension paving the path to exascale for big ensembles data assimilation and forecasting the Red Sea

Samuel Kortas, Habib Toye, Ibrahim Hoteit
King Abdullah University of Science and Technology
(samuel.kortas, habib.toyemahamadoukele, ibrahim.hoteit)@kaust.edu.sa

An easy-to-install generic solution

How to make capacity and capability jobs cohabit on Shaheen XC-40?

- Some of our users use KAUST Cray XC-40 Shaheen to explore parameters sweeping involving thousands of jobs saving thousands of temporary files.
- Need a result in a guaranteed time.
- Are not HPC experts, but are challenging problem in terms of scheduling and file system stress.
- Implement complex workflows sending the output of one code into the input of others and producing a large number of small files.

BUT resources are configured/tuned for capability jobs:

- Limited number of jobs,
- Running on a high number of nodes,
- Producing a small number of big files.

All the opposite of what our users' job needs!

- Scheduler is not configured for 1000s of jobs (# job limitation)
- Filesystem is not tuned for 1000s jobs each writing small files
- Hardware and numerical failures happens

Our strategy:

- Pack 'many jobs' together to make them appear as big ones
- Reduce stress on filesystem by using Ramdisk and messages.
- Extend the scheduler with tools transparent to users
- Make these features available in a SLURM extension, easely usable and installable, and 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 set of jobs as a single workflow easing their submission, monitoring, deletion or reconfiguration.
- Ease the definition, submission and management of jobs run on a large set of combinations of parameters.
- Benefit from a centralized log file.
- Send user-configurable mail messages at any step of its execution of the workflow.
- To 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:

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

Decimate provides additional commands and parameters:

dbatch	submit the job (SLURM sbatch command with additional parameters handling fault-tolerance, and holding 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.

Decimate's Features

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 workflow.

```
dbatch --jobname step1 --array 1-5 --check check1.sh --max-retry 3 step1.sh
dbatch --jobname step2 --depends step1 step2.sh
```

Example of checking script my_check.sh

Expected Results

- SUCCESS: job went fine
- FAILED: job needs to restart
- HALT: the whole workflow failed

Three transparent execution environments

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 submited 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.

```
dbatch --parallel-runs=4 --array=1-200 --nodes=3
       --time=0:30:00 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
```

Application to a specific case

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

First set of mitgcm (1000 x 3-node runs)
- barrier -
Apply the filter (dart) (1 x 16-node run)
- barrier -
second set of mitgcm (1000 x 3-node runs)
- barrier -
Apply the filter (dart) (1 x 16-node run)
- barrier -
3rd set of mitgcm (1000 x 3-node runs)
- barrier -
Apply the filter (dart) (1 x 16-node run)

Results and Perspectives

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.

Results:

Perspectives:

- We are now targetting more members: 5000?
- Decimate's documentation will be improved and additional efforts will be dedicated to dissemination.
- Comparison with other tools ongoing... 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.

Comparison between 100 and 1000 ensemble members

SSH forecast/analysis/increment assimilation experiment with ensemble size of 1000, 100 and 100 with localization compared with gridded AVISO product on Jan 12, 2006, superimposed with the alongtrack altimeter data. After [1].

References

The research reported 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.

[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., Submitted to Journal of Computational Science.
[2] http://decimate.readthedocs.org

SCAN ME

http://decimate.hpc4all.org
decimate@hpc4all.org

help@hpc.kaust.edu.sa