University of Zurich[UZH]

GC3: Grid Computing Competence Center

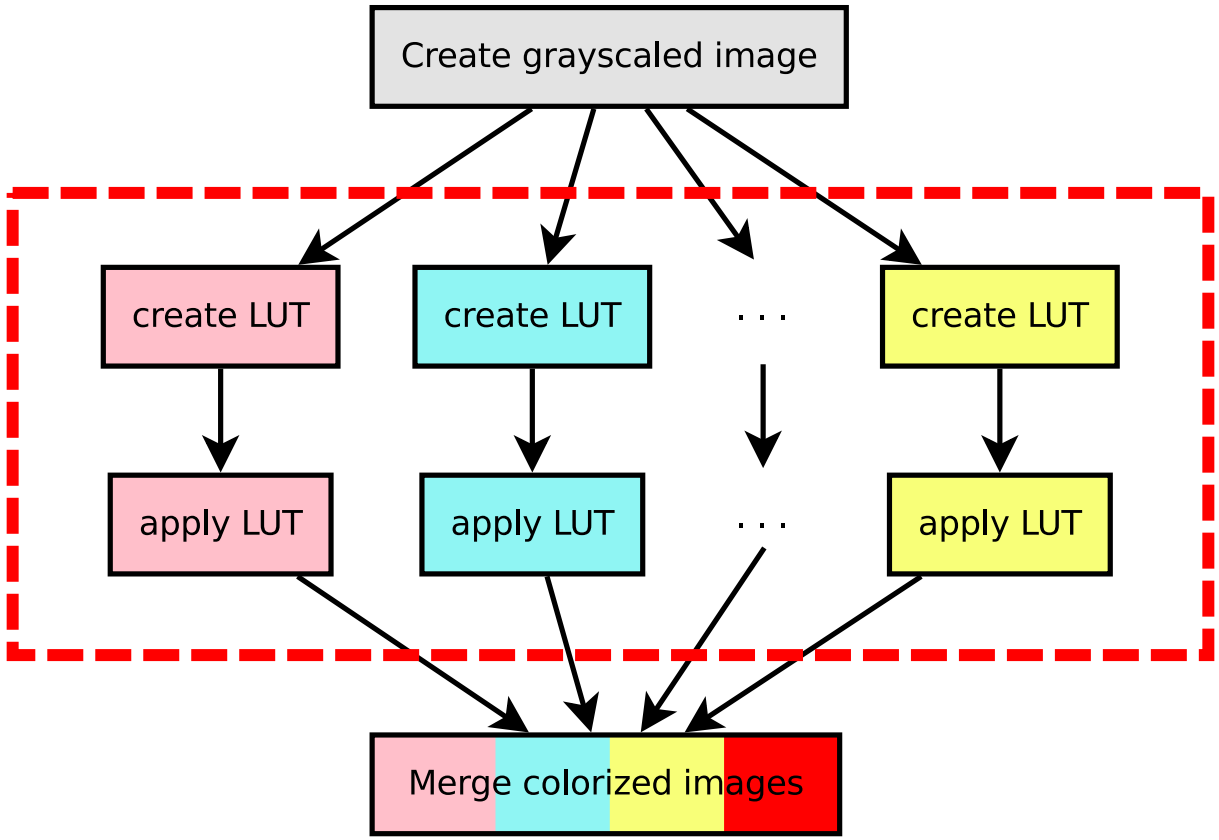
The ParallelTaskCollection

GC3: Grid Computing Competence Center,
University of Zurich

Oct. 2, 2012

Running jobs in parallel

`ParallelTaskCollection` provides an interface for running jobs in parallel.



ParallelTaskCollection - example

```
from gc3libs.workflow import @\HL{ParallelTaskCollection}@

class ParallelHello(@\HL{ParallelTaskCollection}@):
    def __init__(self, hwstring, ncopies, **extra):
        tasks = []
        for i in range(ncopies):
            extra_args = extra.copy()
            extra_args['output_dir'] += ".%d" % i
            task.append(
                GHelloWorld(
                    hwstring,
                    **extra_args))
        # This is IMPORTANT!!!
        ParallelTaskCollection.__init__(self, tasks, **extra)
```

ParallelTaskCollection - example

```
from gc3libs.workflow import ParallelTaskCollection

class ParallelHello(ParallelTaskCollection):
    def __init__(self, hwstring, ncopies, @\HL{**extra}@):
        tasks = []
        for i in range(ncopies):
            extra_args = extra.copy()
            extra_args['output_dir'] += ".%d" % i
            task.append(
                GHelloWorld(
                    hwstring,
                    @\HL{**extra\_args}@))
        # This is IMPORTANT!!!
        ParallelTaskCollection.__init__(self, tasks, **extra)
```

ParallelTaskCollection - example

```
from gc3libs.workflow import @\HL{ParallelTaskCollection}@

class ParallelHello(ParallelTaskCollection):
    def __init__(self, hwstring, ncopies, **extra):
        tasks = []
        for i in range(ncopies):
            extra_args = extra.copy()
            extra_args['output_dir'] += ".%d" % i
            task.append(
                GHelloWorld(
                    hwstring,
                    **extra_args))
        # This is IMPORTANT!!!
        @\HL{ParallelTaskCollection.\_\_init\_\_(self, tasks, **extra
```

Exercise 9.A

Start from the source code of exercise 5.B:

- Modify `new_tasks` so that it returns a list containing only an instance of a `ParallelHello` class.
- Create a `ParallelHello` class which inherits from `ParallelTaskCollection` and runs 20 instances of the `GHelloWorld` application.

Further customizations of `ParallelTaskCollection`

`terminated()` method: Default implementation:

```
def terminated(self):
    """
    Called when the job state transitions to 'TERMINATED',
    i.e., the job has finished execution (with whatever exit
    status, see 'returncode') and the final output has been
    retrieved.

    Default implementation for 'TaskCollection' is to set the
    exitcode to the maximum of the exit codes of its tasks.
    """
    self.execution._exitcode = max(
        task.execution._exitcode for task in @\HL{self.tasks}@
        )
```

The `self.tasks` attribute of the `ParallelTaskCollection` contains a list of all the tasks of the collection

Exercise 9.B

Start from the source code of exercises 5.C

- Create a `ParallelCpuinfo` class which inherits from `ParallelTaskCollection`
- update `new_tasks` method to return a list of `ParallelCpuinfo` instances.
- implement a `terminated` method in the `ParallelCpuinfo` which prints the model names from the various `GArchApplication` applications
- modify the `after_main_loop` method of the `GArchScript` class in order to get the results from the various `ParallelCpuinfo` objects

Exercise 9.C

Create a new script that takes one or more directories from command line, creates a job for each file inside those directories and execute the `md5sum` command on each file in parallel.

- Create a `MD5SumApplication` application that accept one argument `filename` and execute `md5sum` on it.
- Create a `ProcessFilesInParallel` ParallelTaskCollection that, given a directory as argument, creates an `MD5SumApplication` application for each file in that directory.
- Create a `MD5SumScript` SessionBasedScript that takes one or more directories as arguments and runs a `ProcessFilesInParallel` task for each directory