COSTA
common set of tools for the assimilation of data

MEMO	CTA memo200801
Date	21-01-2008
Author(s)	Nils van Velzen
Subject	Parallel Computing in COSTA

1 Introduction

Simulation models can be very large in terms of memory usage and computational requirements to perform a simulation. For these large models it is sometimes necessary to use parallel computing in order to be able to perform a simulation in a reasonable time. Most data assimilation and model calibration methods will perform a large number of model runs. Therefore increasing the computational time significantly compared to a normal simulation. Parallel computing is a vital part for large simulation models and for that reason COSTA supports the usage of parallel simulation models as well the auto-parallelization of non-parallel models. In this report we will give a description of the parallel computing capabilities of COSTA.

2 Almost invisible for user

COSTA is an environment that must be easy to use. The parallel computing facilities are therefore set up in such a way that they are easy to use. To simplify usage for users who do not need parallel computing it is developed in such a way that is is completely hidden when it is not needed.

At the installation it is possible to install COSTA without parallel support. This simplifies the installation since no third party MPI library need to be installed on the system. There is only a single additional call to an initialization function necessary in order to use the parallel functionality and a minimum of additional configuration needs to be added to the configuration files compared to sequential runs.

3 Kinds of parallelism

There are various methods to incorporate parallel computing in a simulation model. Two popular methods are:

- a single process with multiple threads. Within a single executable multiple threads are started to execute parts of the code in parallel. These kind of parallel computing can by implemented by using e.g. OpenMP,

- multiple processes that together perform the computations. The data is distributed over the various processes and the processes cannot directly access each other data. Information is pased between the processes by sending messages to each other. The software libraries MPI and PVM are popular for developing these kind of programs.

Models that are parallelized using threads are not different than normal sequential models as far as COSTA concerns. These models can therefore be transformed into a COSTA model component as any normal sequential model.

Special functionality is added to COSTA in order to be able to link to models that are parallelized using the second concept. The model computations will take place in different executables as the data assimilation computations and the various methods of the model are realized by sending messages between the data assimilation method and the model. However all communication is hidden behind the COSTA model interface. The data assimilation code that is used for a sequential model is therefore exactly the same as for parallel models.

The support for parallel computing in COSTA is based on MPI since it is at this time the most widely supported and used library for creating parallel simulation models. In theory it is also possible to create hybrid parallel applications that are based both on PVM and MPI but we have no experience with this.

There are various approaches to introduce parallelism in a model. COSTA supports the following two:

1. Master worker; One process called the master is special. This process solves the problem by giving the workers tasks.

2. Worker worker; The problem is split up into peaces and all processes work together in order to solve part of the problem. This kind of parallelism is used e.g. when:

   - a large computational area is split up into parts.
   - two different models are combined into a larger model. Like a runoff model with a river model.

Figure 1 illustrates the coupling between a data assimilation method and a parallel model. The main difference between the coupling between a Master-Worker and Worker-Worker process is the communication between the data assimilation method and the model. In the Master-Worker situation there is only communication between the data assimilation method and the master process of the model. In case of a Worker-Worker model, the data assimilation method communicates with all worker processes.

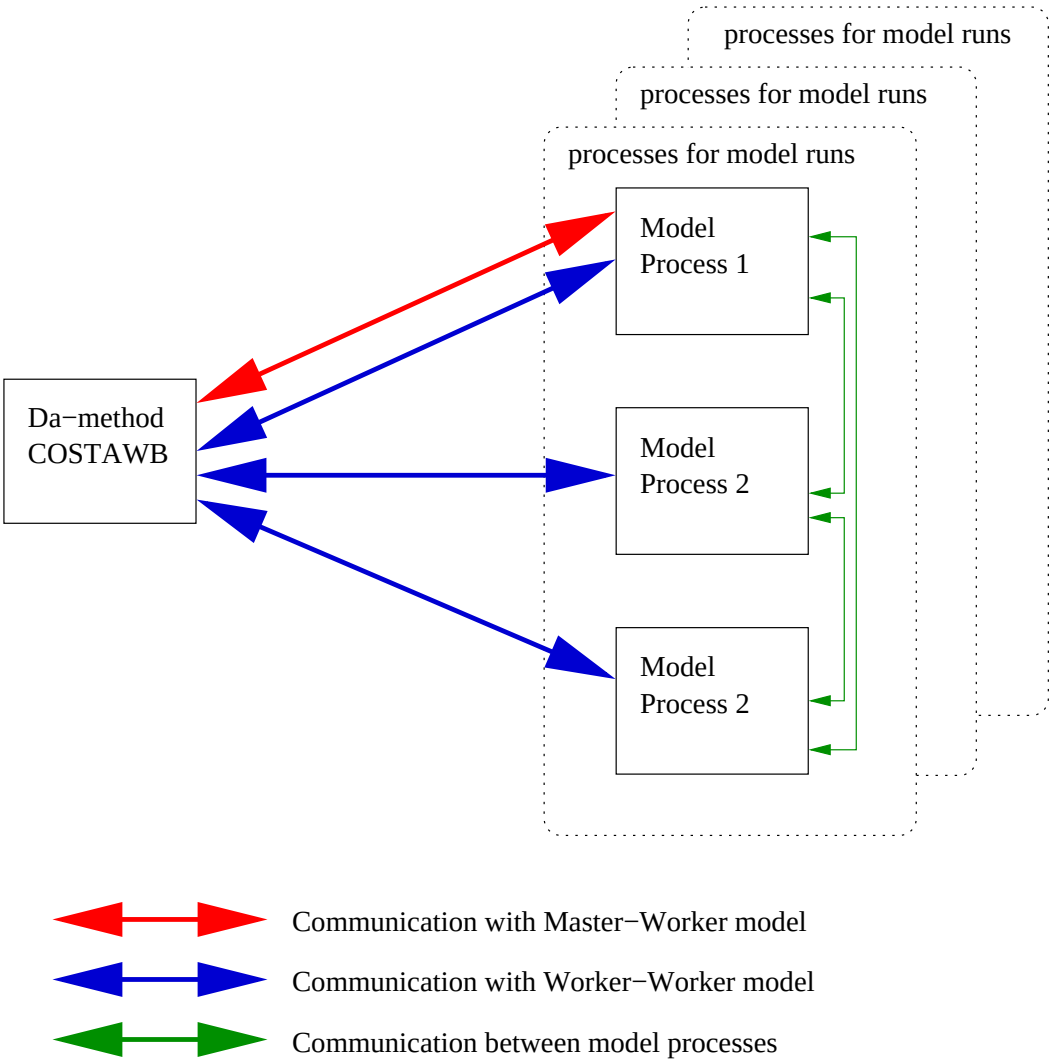

Figure 1: *Example of a coupling with COSTA and a parallel model. When the model is parallel according to the Master-Worker concept, there will only be communication between the data assimilation method and the Master process of the model. When the model is parallel according to the Worker-Worker concept, it is necessary to communicate between the data assimilation method and all worker processes. In this example we see that it is possible to create multiple groups of model processes each holding there own model instances*

4 How does it work

A parallel COSTA application consists of a number of processes (executables that are started). One of the executables implements the data assimilation method and it has the task of a master process. The other executables are worker processes and used for performing the model computations.

When a method is used from the data assimilation method then COSTA will send a message containing all necessary information to the worker processes that implement the model.

In the case the model itself is parallelized using the master-worker principle, the message is only send to the master process of the model. Models that are parallel using the worker-worker principle are handled like a concatenation of COSTA models. All messages are send to all the worker processes and the overall state of the model is the concatenation of the states of all the worker processes. The concatenation is handled by COSTA and invisible.

5 Process groups in MPI

Parallel models that are linked to COSTA will run in a larger group of processes than they are original developed for. Fortunately MPI provides the concept of process groups and communicators.

COSTA defines three communicators that can be used for communication between the various processes:

- CTA_COMM_WORLD; The communicator for all processes in the COSTA universe.
- CTA_COMM_MYWORLD; The communication group of the parallel model. This communicator should be used for all existing communication calls in the model.
- CTA_COMM_MASTER_WORKER; The communication group consisting of master process (data assimilation method) and all worker processes (processes that are used by the model) the master directly communicates with.

The communication groups are created by the function `CTA_Par_CreateGroups` at startup of the processes. This routine needs configuration that is specified in an XML-input file. For all models it must be specified how many processes are needed for a single model instance (`nproc`), how many of these groups we want to create (`ntimes`), and the kind of parallel computing is used inside the model "Master-Worker" or "Worker-Worker" (`parallel_type`). There are two ways to specify this in the input file. In this first form the parallel attributes are added to the definition of the model class. For example:

```
<CTA_MODELCLASS id="modelclass"
                implements="pollute2d"
```

```
                name="CTA_MODBUILD_SP"
                nproc="2"
                parallel_type="Worker-Worker"
                ntimes="*"/>
```

In this example the 2d pollution model is parallelized using the Worker-Worker principle. This model uses 2 processes. The option `ntimes="*"` indicates that as many processes will be used as available to distribute the available model instances.

For example: when 5 processes are started then one process will run the data assimilation method and there will be two groups of two processes both handing half of the created model instances.

An other option is to specify the information separate from the model class definition. This can be useful for the configuration of worker processes that do not need to define the model classes of other models. For example

```
<parallel>
   <process_groups>
       <group name="group_of_2", nproc="2", use_for_model="pollute2d",
              parallel_type="Worker-Worker" ntimes="*">
       <group name="group_of_3", nproc="3", use_for_model="rainfall",
              parallel_type="Master-Worker" ntimes="2">
   </process_groups>
</parallel>
      :
      :
<CTA_MODELCLASS id="modelclass"
                implements="pollute2d"
                name="CTA_MODBUILD_SP"/>
```

Note that the `implements` tag must correspond to the `use_for_model` in the group specification. It is allowed to mix both ways of specification in a single input file.

6 Starting up parallel runs

The COSTA workbench program `costawb` can be used both for parallel as sequential runs. The only difference is the need of an additional argument `-p` to indicate that a parallel run is started. Depending on the MPI distribution the processes must be started using `mpiexec` or `mpirun`. Note that the exact usage of these startup programs differs between the various MPI distributions. The examples we give here work for the Lam-MPI distribution. A parallel run can be started with the command:

```
mpiexec -np 3 costawb -p input.xml
```

This will start 3 processes. Two are available for the model and one for the data assimilation method itself.

The 2D pollution model is one of the test models in COSTA. There are two parallel versions of the model available. A Worker-Worker and a Master-Worker version.

The Worker-Worker version of the model can be started using

```
mpiexec -np 3 costawb -p ens_pollute2d_ww.xml
```

since all processes are the same. The Master-Worker version is however different. The Worker processes of the model are different executables. The processes can be started using the command:

```
mpiexec -np 2 costawb -p ens_pollute2d_mw.xml :\
        -np 1 pollute2d_worker ens_pollute2d_mw.xml
```

This will start 3 processes. The first two `costawb` handle the data assimilation method and the Master process of the model and `pollute2d_worker` is the worker process of the model.

7 Creating a parallel COSTA model

7.1 Master-Worker models

The Master process implements the COSTA model interface just like a sequential model in case of a Master-Worker parallelization of the model. The worker processes only need to call the function `CTA_Par_CreateGroups` at startup to create the process groups and communicators. The worker processes have in general their own executable. For example the code of the executable of the worker processes of the pollution model is give by:

```
program pollute2d_worker
use pollute2d_params, only:pollute2d_params_init
use pollute2d,        only: create_state_vector
implicit none
include 'cta_f77.inc'

integer ::state   !State vector
integer ::ierr    !Error code
integer ::hfile   !Name of configuration file
integer ::htree   !Costa tree representation of input file
```

```
character(len=1024) ::inpfile

   !Get name of input file from command line
   if (iargc()/=1) then
      print *,'Program must have 1 argument'
      call exit(-1)
   endif
   call getarg(1, inpfile)

   call CTA_Initialise(ierr)

   ! Read configuration file (contains the procress-group information)
   call CTA_String_Create(hfile, ierr)
   call CTA_String_Set(hfile, inpfile, ierr)
   call CTA_XML_Read(hfile,htree, ierr)
   if (ierr/=CTA_OK) then
      print *, 'Error opening or parsing file ',inpfile
      call exit(1)
   endif

   ! Create process groups but do not start model builder
   call CTA_Par_CreateGroups(htree,CTA_FALSE, ierr)

   ! Perform initializations
   call pollute2d_params_init

   ! Create state vector
   call create_state_vector(1, 0, state, ierr)

   ! Wait for tasks of the master
   do
      call pollute2d_compute(CTA_NULL,state, CTA_NULL, CTA_FALSE , &
                             CTA_NULL, CTA_NULL, ierr, .false.)
   enddo

end program
```

7.2 Worker-Worker models

The model component is programmed exactly in the same way as a sequential model. When one of the methods of the model is used from the data assimilation method it will result in a call to the corresponding method at all worker processes. The communicator

`CTA_COMM_MYWORLD`) can be used by the model to communicate to the other processes that implement the model.