Flexible Coupling for Performance

Chris Armstrong
Rupert Ford
Graham Riley
Overview

- Introduction
- Deployment flexibility (BFG1)
- Argument Passing (BFG2)
- GENIE results
- Conclusions
Introduction

- Flexible Coupling Approach: metadata describing individual models (e.g. subroutines or methods), their composition into a coupled model and their deployment onto resources, and generate the required wrapper code (e.g. main(s) and communication code).
- BFG\{1,2\} are implementations of the above approach
Introduction: BFG Implementation

- Model XML
- Composition XML
- Deployment XML
- Coupled XML
- XSLT
- XSLT engine
- Script
- TMP XML
- Code and scripts
Introduction: Model Wrapping

Main(s), Control, Coupling wrapper

Atm  Ocn  Sea-ice

Coupling wrapper

Target Coupling Infrastructure

- Existing code/library code
- BFG-generated wrapper code
BFG1: Deployment Flexibility

- Support for many targets, therefore choose most appropriate:
  - in-sequence, mpi, Oasis3, tdt, web services
  - (Oasis4, esmf, …)
- No change to model code (or composition)
BFG1: Deployment Flexibility

- Ability to choose most appropriate mapping of models to executables, with no change to model code (or composition)
module m1model
...  
real :: a,x,y
...
subroutine m1init()
  ! Do things
  call put(a,3)
end subroutine m1init
...
subroutine m1()
  call get(x,6)
  ! Do things
  call put(y,3)
end subroutine m1
...
end module m1model

module m2model
...
real :: a,b,c
...
subroutine m2init()
  call get(c,2)
  ! Do things
end subroutine m2init
...
subroutine m2()
  call get(a,1)
  ! Do things
  call put(b,1)
end subroutine m2
...
end module m2model
Running in Sequence?

BFG (In-place) style control

program mycoupledmodel

use m1model
use m2model

call m1init()
call m2init()

do i=1,nts
   call m1()
call m2()
end do

end program mycoupledmodel

Hand-crafted Arg-passing comms (and data allocation)

program mycoupledmodel

use m1model
use m2model

real :: a,b,c

call m1init(a)
call m2init(a)

do i=1,nts
   call m1(b,c)
call m2(b,c)
end do

end program mycoupledmodel
module m1model

... real :: a,x,y ...

subroutine m1init()
    ! Do things
    call put(a,3)
end subroutine m1init

... subroutine m1()
    call get(x,6)
    ! Do things
    call put(y,3)
end subroutine m1

... end module m1model
module m1model

...

subroutine m1init(a)
  real, intent(out):: a
  ! Do things
end subroutine m1init

...

subroutine m1(x,y)
  real, intent(in) :: x
  real, intent(out):: y
  ! Do things
end subroutine m1

...

end module m1model

module m1modelwrap

...

use m1model

...

real :: a,x,y

...

subroutine m1initwrap()
  call m1init(a)
  call put(a,3)
end subroutine m1initwrap

...

subroutine m1wrap()
  call get(x,6)
  call m1(x,y)
  call put(y,3)
end subroutine m1wrap

...

end module m1modelwrap
BFG2: Mixed Arg-passing/In-place Communication

- Choose most appropriate for model developer and for required use
- e.g. in-place for some diagnostics

```fortran
module m1model
  ...
  real :: y
  ...
  subroutine m1init(a)
    real, intent(out) :: a
    ! Do things
  end subroutine m1init
  ...
  subroutine m1(x)
    real, intent(in) :: x
    ! Do things
    call put(y,3)
  end subroutine m1
  ...
end module m1model
```
BFG2: Mixed, concurrent/in-sequence

threads/processes

- call atm(a)
- call atm_chem(a)
- call put(a,tag)
- call ocn(a)
- call ocn_chem(a)

mpi

(Oasis4)

threads/processes

- call get(a,tag)
- call ocn(a)
- call ocn_chem(a)
GENIE example

- ESM system [http://www.genie.ac.uk](http://www.genie.ac.uk)
- Models implemented with arg passing, all run in sequence, hand crafted control code (and data allocation)
- Made 4 genie models compliant and generate 2 configurations (ig_fi_sl, ig_sl_sl)
- Same performance as hand crafted implementations with slightly less memory use.
GENIE example
FLUME, PRISM, BFG

- Graham and I are consultants to the Met Office on FLUME.
- BFG1 and BFG2 were originally implemented to test out the ideas being developed in FLUME.
- It is hoped that FLUME models will be compatible with BFG2. The current plan is to follow the same model coding rules and to ensure that the metadata describing models are at least compatible and hopefully the same.
- FLUME will use Oasis4 (the latest generation PRISM Coupler) to couple models concurrently (plan to use wrapping code approach).
- Oasis4 will be a BFG2 target
Conclusions

- Flexibility in deployment allows
  - choice of most efficient “target” infrastructure
  - choice of most efficient mapping of models to “main’s”
- Argument passing interface allows
  - as efficient generated code (in memory and time) as hand crafted code when running in sequence
  - all models to be run in-sequence, some models to be run in-sequence and some concurrently, all models to be run concurrently. Can choose most appropriate target and mapping to mains for concurrent models.
  - (potentially) More fine grain coupling without loss of performance
  - (potentially) Coupling for both NWP and ESM.
Thanks …

http://www.cs.manchester.ac.uk/cnc/projects/bfg

ECMWF Workshop
11th October 2006