LABORATÓRIO NACIONAL
DE ENGENHARIA CIVIL

DEPARTAMENTO DE HIDRÁULICA E AMBIENTE
Núcleo de Tecnologias de Informação em Hidráulica e Ambiente

Proc. 0602/11/16283

UTILIZAÇÃO DA PROGRAMAÇÃO PARALELA NO DHA

LNEC

Lisboa • Novembro de 2007

I&D HIDRÁULICA E AMBIENTE

RELATÓRIO 334/**2007 – NTI**

SUMÁRIO

O Departamento de Hidráulica e Ambiente (DHA) do LNEC tem sentido, desde há algum tempo, a necessidade de aumentar a precisão da simulação na modelação de determinados fenómenos e, simultaneamente, a necessidade de conseguir obter resultados de forma mais rápida. Tirando partido dos desenvolvimentos a nível de equipamento, através de máquinas multi-processador (super-computadores) ou através da ligação, via rede, de diversas máquinas mono- ou multi-processador, a programação paralela apresenta-se como uma forma de responder a estas necessidades. Os agregados de máquinas são usualmente chamados de *clusters*. O desempenho destes *clusters* é inferior ao dos super-computadores, mas o preço de um destes agregados é muitíssimo inferior ao de um super-computador. Tendo o LNEC adquirido recentemente um *Cluster*, era importante que se começasse a explorar as potencialidades da programação paralela com o objectivo de optimizar as simulações realizadas frequentemente nos seus estudos. Assim procedeu o DHA.

Este relatório descreve parte da actividade, durante o ano de 2006, dum Bolseiro de Iniciação à Investigação Cientifica (BIIC), recrutado para apoiar os investigadores do Departamento a explorar modelos numéricos que usam o paradigma da programação paralela.

ABSTRACT

The Hydraulics and Environment Department of LNEC has been urged, for some time, for a quicker and/or more precise method to perform some simulations of certain phenomena.

Hardware development has permitted to create supercomputers based on large multi-processor machines. However, the price of such equipment is prohibitive. Using several standard computers with one or more processor, interconnected through a fast network, similar, yet inferior performances have been achieved, but the financial cost of these clusters of computers is, by far, inferior to the cost of one supercomputer. LNEC has purchased recently a cluster, and there was the need to explore the potential of parallel and distributed programming as a means to optimize the numerical simulations that are frequently used in its studies.

This report describes part of the activity carried out by a trainee hired to help other researchers in the Department to run some numerical modes using parallel programming.

ÍNDICE

INDICE DE FIGURAS

1 | INTRODUÇÃO

Este relatório descreve parte da actividade do Bolseiro de Iniciação à Investigação Cientifica (BIIC), Bruno Lucas, durante o ano de 2006. Esta actividade está de acordo com o seu plano de trabalhos, que se encontra no Anexo IV. Durante esta fase, o BIIC deveria familiarizar-se com os modelos numéricos utilizados no Departamento de Hidráulica e Ambiente em diversas simulações realizadas pelos vários núcleos, nomeadamente pelo NEC (Núcleo de Estuários e Zonas Costeiras), NPE (Núcleo de Portos e Estruturas Marítimas) e pelo NRE (Núcleo de Recursos Hídricos e Estruturas Hidráulicas). Usualmente, este tipo de modelos numéricos encontra-se programado utilizando um paradigma sequencial, mas ultimamente tem-se passado a desenvolver os modelos numéricos sobre um paradigma da programação paralela.

Existe a necessidade de aumentar a precisão da simulação e/ou a vontade de se conseguir obter resultados de forma mais expedita, e a programação paralela apresenta-se como uma forma de responder a estas necessidades. Há já algum tempo, que, a nível de equipamento, apareceram máquinas multi-processador (super-computadores) de preços exorbitantes. Posteriormente, tentou-se recriar as capacidades de processamento destes super-computadores com a ligação de diversas máquinas mono-processador, via rede. Estes agregados são usualmente chamados de *clusters*, e embora o seu desempenho seja inferior ao dos super-computadores, o seu preço é muitíssimo inferior ao de um super-computador.

Existem, por isso, actualmente as ferramentas de hardware e software que permitem utilizar a programação paralela em diversos campos.

A contratação do bolseiro inclui-se na estratégia do DHA de estar na vanguarda das novas tecnologias de simulação, nomeadamente, do processamento paralelo e computação GRID.

No capítulo 2 deste relatório, faz-se uma pequena introdução à programação paralela e às tecnologias disponíveis. No capítulo 3, descrevem-se os estudos realizados com os diferentes modelos testados. No último capítulo, apresentam-se algumas conclusões e trabalho futuro. Finalmente, nos Anexos I e II, apresentam-se os Guias de utilização das versões paralelas dos modelos Adcirc e Elcirc e, no Anexo III, o Guia para uma autenticação segura sem recorrer a palavras-chave. No Anexo IV, apresenta-se o Programas de Trabalhos, que o autor deveria cumprir no seu primeiro ano de actividade no LNEC.

2 | PROGRAMAÇÃO PARALELA

2.1 Introdução

A resolução de problemas, de diversas áreas, através da simulação computacional numérica, está muitas vezes limitada pelas capacidades computacionais de processamento e memória. Estas

limitações levam a que os modelos simulados não possuam o detalhe desejado ou que os cálculos sejam alvo de simplificações muito fortes. A programação paralela apareceu como uma possível solução para a resolução destes problemas, permitindo o acesso a maiores recursos computacionais, conseguido-se assim resolver os problemas em menos tempo. Isto permite aumentar a precisão da solução e/ou diminuir o tempo em que é obtida a resposta ao problema sem o custo financeiro exacerbado de um super-computador com capacidades iguais.

Os ganhos da programação paralela não se limitam a aplicações de cálculo científico como estudo da dinâmica de fluidos, comportamento de partículas ou previsões meteorológicas. Diversas áreas como, por exemplo, as ciências económicas (modelos financeiros) ou a informática (modelos transaccionais das grandes bases de dados ou processamento de língua natural) utilizam paralelismo para ganhar desempenho, inalcançável doutra forma.

2.2 Arquitecturas

Basicamente, pode-se considerar que existem dois tipos de arquitecturas, baseadas no tipo de memória: as arquitecturas de memória partilhada e as arquitecturas de memória distribuída. As primeiras são constituídas por múltiplos processadores que acedem a uma só memória central que se encontra partilhada por todos os processadores. Isto permite partilhas rápidas de dados, mas pode ter efeitos nefastos, por exemplo, dados privados de um processador serem alterados por outro processador ou haver demora no acesso à memória, o qual tem de ser escalonado entre todos os processadores. Este atraso pode ser debelado com a utilização de *caches*[1]. No entanto, a coerência entre as diferentes *caches* e a memória central é também um problema de resolução nem sempre fácil.

Os sistemas de memórias distribuídas são sistemas normalmente compostos de computadores "normais" que possuem assim a sua própria memória central e estão interligados através de uma rede de comunicação, normalmente de bom desempenho. No entanto, o desempenho da rede continua a ser inferior ao conseguido com os barramentos de acesso à memória interna. Nestes sistemas, a comunicação é efectuada através da troca de mensagens (MPI, por exemplo) ou de procedimentos remotos (RPC[2], RMI[3], por exemplo).

Surgiram naturalmente algumas variações destas arquitecturas. Por exemplo, no *cluster* recentemente adquirido pelo LNEC existem 69 máquinas, cada uma possui vários processadores (*multicore*) com a sua própria memória, que pode ser acedida por outros processadores (da mesma máquina) com tempos de acesso superiores aos do processador "dono".

---

1 Memórias de acesso rápido, normalmente de pequena dimensão quando comparadas com a memória central, que armazenam uma cópia dos dados em que o processador está a trabalhar.

2 RPC significa "*Remote Procedure Call*" e é uma tecnologia que permite a execução de um procedimento noutro computador. Mais informações em: http://en.wikipedia.org/wiki/Remote_procedure_call.

3 RMI significa "*Remote Method Invocation*" e está disponível nas soluções tecnológicas Java. Permite a execução de métodos que se encontram em objectos Java em máquinas virtuais (JVM) remotas. Mais informações em: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp.

2.3 Modelos de programação paralela

2.3.1 Introdução

Existindo as capacidades do hardware suportar várias execuções simultâneas, é necessário que o software consiga utilizar esta disponibilidade. A criação de mais uma linguagem de programação, especialmente desenhada para a programação paralela, seria muito possivelmente condenada ao fracasso devido à inércia dos utilizadores de aprender uma nova sintaxe. Foi idealizada assim outra forma de permitir a programação paralela de forma explícita recorrendo a uma interface independente da linguagem ou do sistema operativo (SO). Esta interface serve de ligação entre o programa, que se encontra programado numa linguagem "tradicional", e a biblioteca de funções e procedimentos que permitem a comunicação e coordenação dos diferentes processos envolvidos. Esta biblioteca é implementada especificamente para cada sistema operativo/arquitectura de hardware. No entanto, como está regida por uma interface independente do sistema operativo, os programas permanecem independentes do SO, pois é "preocupação" da biblioteca transformar as instruções de forma a serem compreendidas a nível de SO/hardware. Exemplos destas interfaces são a MPI ou a PVM[1] (para troca de mensagens) ou a OpenMP (para memória partilhada), que se apresentam nas secções seguintes.

Uma outra perspectiva será a de não paralelizar a aplicação em si, mas utilizar bibliotecas matemáticas já paralelizadas, como, por exemplo, o ScaLAPACK[1], que realizam as operações matemáticas com um alto desempenho.

2.3.2 OpenMP

OpenMP, ou "Open Multi-Processing", é uma interface de programação concebida para ser utilizada em sistemas de multi-processador com memória partilhada, o que significa que não pode ser utilizada em mais do que um máquina. Consiste numa série de directivas de compilador, bibliotecas e variáveis de ambiente que transformam o comportamento de um programa acelerando a execução de algumas partes do código, principalmente nos ciclos. As directivas aparecem como comentários quando o código é compilado para execução sequencial, o que significa que as directivas paralelizantes podem ser acrescentadas incrementalmente, ocorrendo uma paralelização gradual. Não são, por isso, necessárias mudanças "radicais" no código-fonte.

Nenhum dos modelos actualmente utilizados no DHA utiliza esta biblioteca, que também não é considerada o *standard de facto* na programação paralela, ao contrário da interface MPI, ver secção seguinte.

É normalmente referido que a utilização desta biblioteca na paralelização é mais simples do que a utilização da MPI. No entanto, a limitação do número de processadores que podem ser utilizados aos

[1] Mais informações podem ser encontradas no site base do PVM: http://www.csm.ornl.gov/pvm/pvm_home.html.

que se encontram disponíveis na máquina, sem possibilidade de recurso a outras, faz com que os seus resultados, a nível do desempenho, possam ser inferiores aos obtidos com a biblioteca MPI. Trata-se, portanto, de uma solução multi-processador e não multi-computador.

2.3.3 MPI

2.3.3.1 Introdução

MPI (Message Passing Interface) é uma especificação que contém especificações LIS (Language Independent Specification), que descriminam as funções e procedimentos que devem existir numa implementação MPI, qualquer que seja a linguagem de programação da implementação.

A comunicação e a troca de dados entre processos MPI ocorre através de troca de mensagens, permitindo assim que cada processo tenha as suas próprias variáveis locais. A interface PVM possui características similares, mas a MPI evoluiu como o *standard de facto*, face à PVM.

A MPI, comparativamente à OpenMP, pode ser utilizada quer em sistemas de memória partilhada quer em sistemas de memória distribuída. No entanto, as alterações ao código, necessárias para a paralelização, são bastante superiores às da OpenMP, podendo também o *debug* ter uma dificuldade acrescida face a uma solução OpenMP.

O tipo e a capacidade da rede de comunicação entre os nós têm uma influência muito grande no desempenho de um programa paralelizado com MPI.

A especificação MPI-1 foi completada em 1994 e a especificação MPI-2 em 1998, embora esta última só recentemente tenha sido implementada. No entanto, ao longo dos anos, têm vindo a ser acrescentadas funcionalidades da especificação MPI-2 às implementações MPI-1.

2.3.3.2 Versões

Existem duas versões da especificação em uso correntemente: a versão 1.2, que enfatiza a troca de mensagens e tem um ambiente de execução estático; e, a versão 2.1, na qual existem novas funcionalidades, tais como I/O escalável, gestão dinâmica de processos, comunicação colectiva com dois grupos de processos.

A versão 1.2 está especificada, para além da LIS, em ANSI C e FORTRAN 90. A versão 2.1 está especificada em ANSI C, ANSI C++ e FORTAN 90, existindo interoperabiblidade de objectos entre diferentes linguagens.

Na versão 1.2 existem cerca de 130 funções ou procedimentos, enquanto que na versão 2.1 existem mais de 500, estando presentes as funções/procedimentos da versão 1.2, embora o uso de algumas seja desaconselhado a favor de outras mais recentes.

---

[1] Mais informações podem ser encontradas no site base do ScaLPACK: http://www.netlib.org/scalapack/.

2.3.3.3 Implementações

Para cada versão das especificações existem várias implementações do *standard* MPI, quer sejam feitas por empresas (Intel[1], HP[2], por exemplo) ou por utilizadores segundo uma politica de Código Aberto. Para códigos que correm em *clusters*, as implementações mais conhecidas e utilizadas são MPICH e LAM[3]. Realizaram-se alguns testes com esta última implementação, mas acabaram por não ser bem sucedidos, ocorrendo erros de execução, nomeadamente com o modelo ADCIRC. A implementação MPICH foi a implementação utilizada na generalidade dos testes efectuados, por ser a implementação mais usada e estável e por vir referenciada na documentação de alguns modelos.

2.3.3.4 Compilação de programas baseados em MPI

A compilação de programas baseados em MPI deve-se efectuar com o compilador MPI específico para a linguagem "base" do programa. Este compilador MPI é simplesmente um *wrapper* que envolve o compilador "original", que se encontra no sistema – por exemplo o GCC[4] ou IFORT[5] – com as variáveis e bibliotecas MPI necessárias para a compilação do código. Por exemplo, a compilação de um programa Fortran será a seguinte: "`mpif90 hello.f90`" em que `mpif90` é o *wrapper* que envolve o compilador de Fortran 90 e "`hello.f90`" é o ficheiro de código-fonte.

Um factor a ter em conta durante a compilação e que tem uma grande influência na execução do programa é a escolha do tipo de comunicação que os processos MPI irão utilizar. O tipo de comunicação é especificado através da variável *device* em tempo de compilação da biblioteca MPI. Um programa compilado com uma biblioteca MPI com um determinado *device*, não se conseguirá executar se, em tempo de execução, estiver em uso outro *device* diferente (na biblioteca MPI).

Durante os testes com os diferentes modelos, foram utilizados os seguintes *devices (Palha Fernandes e Lucas, 2007)*:

- `ch_mem` (comunicação por memória partilhada na mesma máquina – MPI 1);
- `ch4` (comunicação por mensagens, mesma máquina ou máquinas diferentes – MPI 1);
- `ch4mpd` (comunicação por mensagens, mesma máquina ou máquinas diferentes, com a utilização de um *daemon* MPD – MPI 1);
- `ch3:ssm` (comunicação por memória partilhada na mesma máquina e por mensagens em máquinas diferentes – MPI 2);

---

[1] Mais informações em: www.intel.com/cd/software/products/asmo-na/eng/cluster/mpi/ .

[2] Mais informações em: http://h21007.www2.hp.com/dspp/tech/tech_TechDocumentDetailPage_IDX/1,1701,1238,00.html .

[3] A implementação LAM, neste momento, está a desaparecer devido ao facto de passar a fazer parte do consórcio Open MPI (www.open-mpi.org).

[4] Compilador Open Source para a linguagem C. Mais informações em: http://gcc.gnu.org/ .

[5] Compilador da Intel para a linguagem Fortran. Mais informações em: http://www.intel.com/cd/software/products/asmo-na/eng/compilers/282048.htm .

- `ch3:nemesis` (comunicação por memória partilhada na mesma máquina e por mensagens em máquinas diferentes – MPI 2).
- `ch3:shm` (comunicação por memória partilhada na mesma máquina – MPI 2).

Os *devices* podem ser agrupados segundo o tipo de comunicação (memória partilhada/mensagens/mista); a utilização de um *deamon*[1] passa a ser obrigatória na versão 2 (MPICH), como se poderá ver na secção seguinte.

2.3.3.5 Preparação da execução

Na implementação MPICH da especificação 2, é necessário lançar previamente, à execução, um *daemon* que gere os processadores disponíveis. Este programa pode ser iniciado quando se (re)inicia a máquina, ou simplesmente antes de uma execução que necessite de MPICH-2. Quer a execução seja distribuída por diversas máquinas quer somente numa, é necessário executar um *daemon* por máquina. Após o lançamento do primeiro *deamon*, os seguintes irão ligar-se ao primeiro (dado o endereço IP e a porta onde este se encontra à escuta), formando assim um anel lógico de máquinas disponíveis para a simulação.

Para a utilização de modelos baseados em MPI-1 não é necessário o lançamento destes *deamons*.

2.3.3.6 Execução de programas baseados em MPI

A execução de um programa MPI (v1.2) é a seguinte: "`mpirun -np X nomeProg`", em que `X` corresponde ao número de processadores disponíveis e "`nomeProg`" é o nome do ficheiro executável. Outros parâmetros podem ser passados ao sistema MPI, como por exemplo, uma lista de máquinas a utilizar, a arquitectura das máquinas, o número de processadores de cada máquina a utilizar, ficheiros de entrada e saída, etc. Para a versão 2 da especificação MPI, o exemplo acima continua a funcionar (por compatibilidade) embora seja aconselhada uma nova sintaxe "`mpiexec -n X progNome`". De notar que os executáveis (`mpiexec e/ou mpirun`) devem estar na `$PATH`[2].

3 | MODELOS NUMÉRICOS TESTADOS

3.1 ADCIRC

3.1.1 Apresentação

O modelo ADCIRC ("*A ADvanced CIRCulation model for oceanic, costal and estuarine waters*") é um modelo hidrodinâmico utilizado para estudar águas pouco profundas, desenvolvido por diversas

[1] Programa que corre em *background*, normalmente em actividades de monitorização, executando uma tarefa quando se dá um evento específico.

[2] Lista de caminhos onde estão os ficheiros executáveis dos diversos programas.

Universidades Americanas (University of North Carolina at Chapel Hill, University of Notre Dame, entre outras). É um modelo utilizado pelo Núcleo de Estuários e Zonas Costeiras (NEC).

Existem duas implementações deste modelo, uma de execução sequencial e outra de execução paralela, normalmente designada pela sigla `pAdcirc`.

3.1.2 Compilação

O código do modelo encontra-se programado na linguagem Fortran 90 e utiliza a biblioteca de comunicação MPI, ver secção 2.3. O modelo foi compilado recorrendo à especificação MPI versão 1.2 e à implementação MPICH na sua versão 1.2.7.p1. Posteriormente, foi compilado com a especificação 2 implementada na biblioteca MPICH, na sua versão 1.0.5.p3.

Inicialmente foi testada a versão `ADCIRC` v43.03, que revelou alguns problemas em certas sub-rotinas Fortran. A nova versão, 45.11, na qual estes problemas já não aconteciam, passou a ser a versão utilizada.

Para compilar o código-fonte foi utilizado o compilador "`ifort`" da Intel, cuja utilização no sistema operativo Linux é grátis. Durante o processo de avaliação de desempenho das propostas do futuro *cluster* do LNEC (Palha Fernandes e Lucas, 2007), este modelo foi compilado com outros compiladores (Ekopath Pathscale[1] – Linux/x86_64; IBM XLF – Linux/PPC) com resultados positivos.

3.1.3 Execução

Existe uma fase de pré-processamento que deve ser realizada antes da execução da simulação. Para tal, é necessário executar um dos programas gerados aquando da compilação, o "`adcprep`". Este programa vai dividir o domínio do problema em sub-domínos, gerando para tal uma sub-directoria (de nome "`PE000x`") para cada um dos processadores disponíveis. Para realizar esta divisão é utilizada uma biblioteca (Metis[2]) especializada na divisão de malhas.

A execução do "`pAdcirc`" faz-se de forma normal para este tipo de programas, ou seja "`mpirun -np X ./padcirc`", em que `X` é o número de processadores.

Para realizar o pós-processamento, deve-se executar o programa "`adcpost`", gerado aquando da compilação do modelo. A execução deste programa transforma o formato dos ficheiros de saída da versão paralela no mesmo formato dos ficheiros da versão sequencial, podendo ser tratados da forma convencional.

3.1.4 Resultados

Os bons resultados obtidos mostraram a validade desta metodologia paralela, tendo-se conseguido obter tempos quase-linearmente inferiores ao número de processadores envolvidos, como mostra a

[1] Disponível no cluster Medusa do LNEC.

[2] Mais informações em: http://glaros.dtc.umn.edu/gkhome/views/metis .

evolução dos tempos de execução do modelo, nos gráficos das Figuras 1 e 2. Nestes gráficos quanto menor for o tempo de execução, melhor.

No Anexo I, encontra-se um guia mais técnico da compilação e execução do modelo `Adcirc` em modo paralelo (p`Adcirc`).

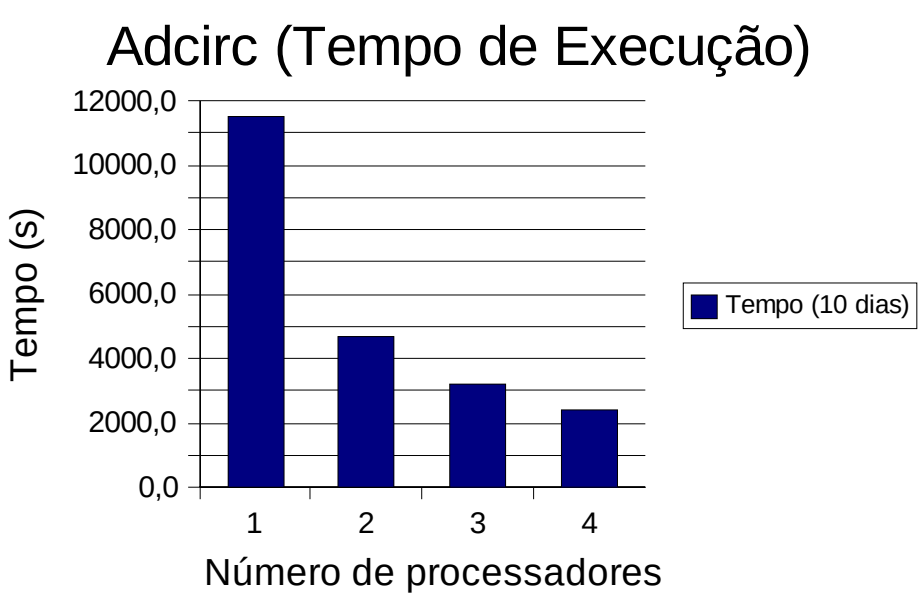

Figura 1 | Tempo de execução do modelo Adcirc, simulação de 10 dias, nas estações de trabalho do DHA-NEC

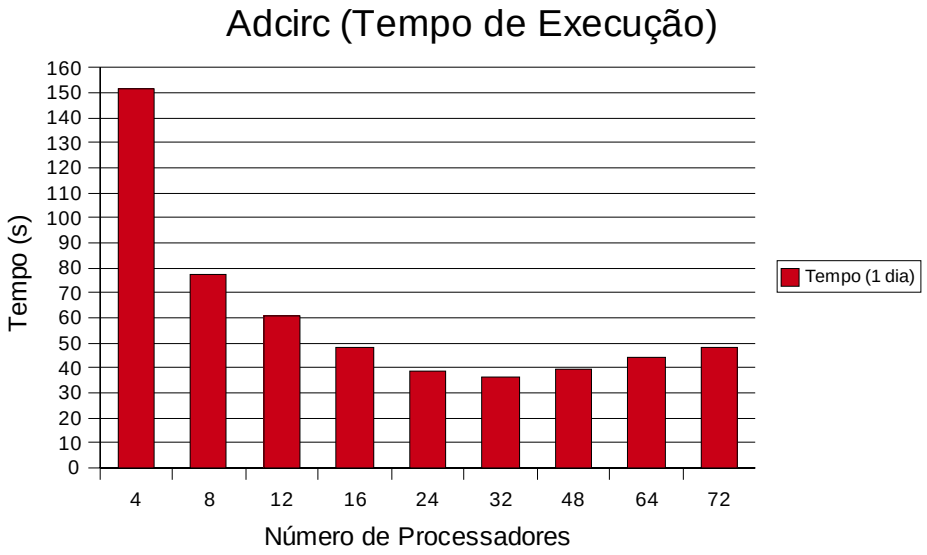

Figura 2 | Tempo de execução do modelo Adcirc, simulação de 1 dia no cluster Medusa do LNEC (resultados provisórios)

3.2 ELCIRC

3.2.1 Apresentação

O modelo ELCIRC[1] é um modelo hidrodinâmico que utiliza uma malha não estruturada para a simulação a 3 dimensões da circulação baroclínica em estuários e zonas costeiras. O código sequencial do modelo é utilizado pelo NEC em diversos estudos. O modelo ELCIRC utiliza um algoritmo baseado no método das diferenças finitas, volumes finitos e técnicas Eulerianas-Lagrangianas para resolver as equações de águas pouco-profundas.

3.2.2 Compilação

A versão paralela, `pElcirc`, está desenvolvida na linguagem de programação FORTRAN 90 e na paralelização foi utilizada a biblioteca de troca de mensagens MPI, ver secção 2.3.3. O modelo utiliza a versão 2 desta biblioteca. Durante os testes foi inicialmente utilizada a versão 1, o que conduziu a vários problemas.

É necessário gerar a biblioteca de divisão de Malhas "`PartMetis`" antes da compilação do modelo em si. Para tal, deve-se editar o ficheiro "`Makefile.in`" existente na directoria desta biblioteca, de forma a utilizar os compiladores e definições correctas para o sistema em causa. A biblioteca está na linguagem C e foi utilizado o compilador GCC.

Posteriormente, compila-se o modelo ELCIRC, devendo-se alterar o ficheiro "Makefile" para que o modelo compile nas máquinas do DHA-NEC. O compilador FORTRAN utilizado foi o "`ifort`" da Intel. Existem diversos parâmetros de compilação a ter em conta, sendo "`-DORDRED_SUM`" o que mais influencia o desempenho do código

No Anexo 2, encontra-se um guia mais técnico da compilação e execução do ELCIRC em modo paralelo.

3.2.3 Execução

Visto o `pElcirc` utilizar a versão 2 da implementação MPICH, é necessário lançar previamente ao início da corrida o *deamon* MPD: Para mais informação ver a secção 2.3.3.5.

Embora não exista um procedimento de pré-processamento explícito para o utilizador, é necessário executar, da primeira vez, o programa `convert_param`, gerado aquando da compilação, para gerar um novo ficheiro de configuração ("`param.in`") compatível com a execução paralela.

A execução da versão paralela do modelo ELCIRC, inicia-se com o comando, "`mpiexec -n X ./pelcirc`". Embora, segundo a documentação, não seja necessário realizar pós-processamento, levantaram-se alguns problemas com os outputs resultantes dos testes, que apresentam

[1] Mais informações em: www.ccalmr.ogi.edu/CORIE/modeling/elcirc/ .

visualizações incorrectas com o Xmvis (programa utilizado normalmente na versão sequencial para a visualização de resultados).

3.2.4 Resultados

3.2.4.1 Simulações em 2D

Embora se tenha verificado uma diminuição do tempo de execução entre 1 e 4 processadores (Figura 3), utilizando sempre a versão de código paralelo, a verdade é que quando se compara o tempo de execução da versão paralela (4 processadores) com a versão série (1 processador) os tempos são similares (Figura 4). Adicionando mais processadores (noutra máquina) à simulação poderá permitir que a execução paralela ganhe vantagem sobre a execução sequencial.

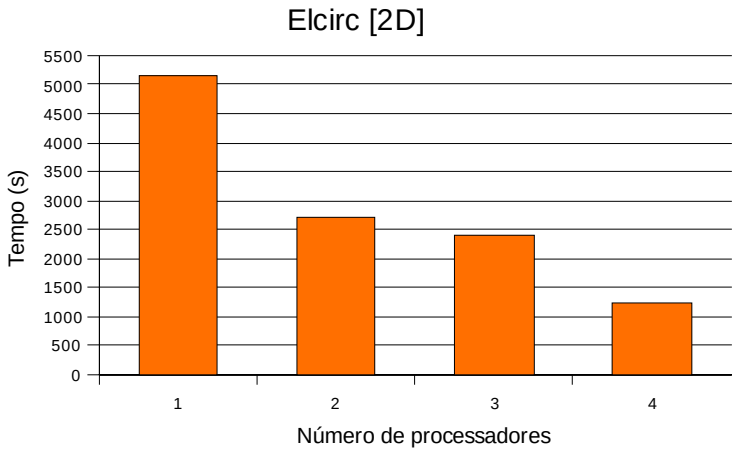

Figura 3 | Tempos de execução do código paralelo

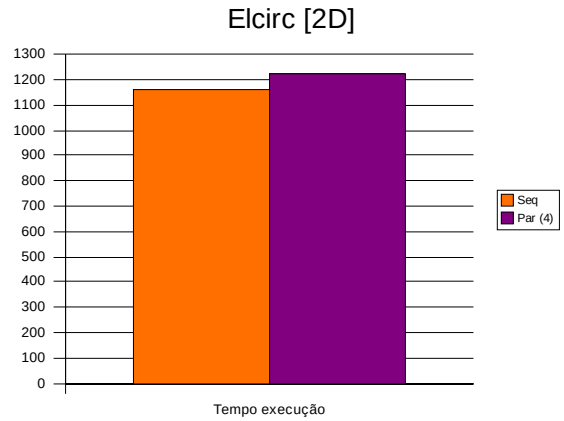

Figura 4 | Comparação dos tempos de execução do código sequencial (esquerda) com o código paralelo a 4 processadores (direita)

3.2.4.2 Simulações em 3D

Nas simulações em 3D, verifica-se que os tempos de execução do código paralelo continuam a descer com o aumento do número de processadores (Figura 5) e as diferenças dos tempos de

execução entre o código sequencial e o código paralelo são evidentes na Figura 6. No entanto, surgiram problemas com corridas com tempos de simulação mais longos que 3 dias, problemas que ainda não foi possível resolver.

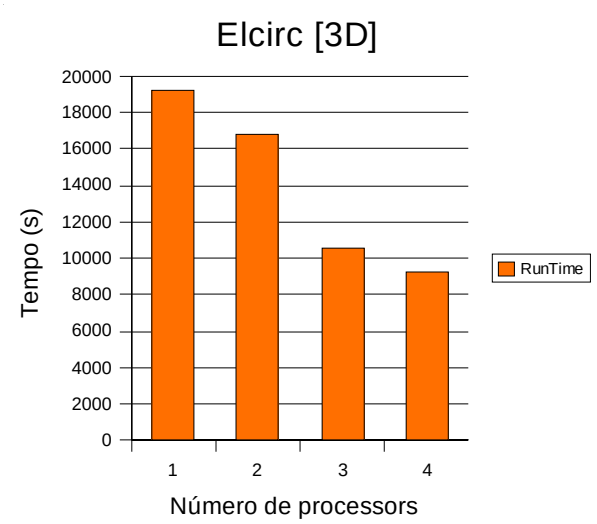

Figura 5 | Tempos de execução do código paralelo

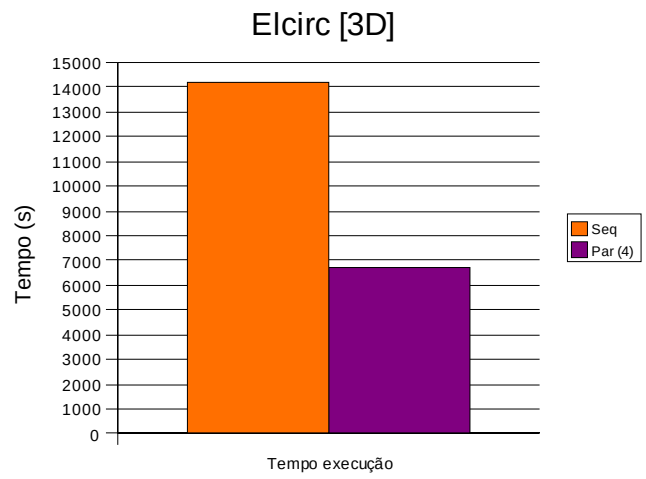

Figura 6 | Comparação dos tempos de execução do código sequencial (esquerda) com o código paralelo a 4 processadores (direita)

3.3 COULWAVE

3.3.1 Apresentação

O modelo Coulwave[1] é um modelo numérico utilizado na geração, propagação e interacção de ondas de superfície que utiliza as equações de ondas dispersas com integração da profundidade. É um modelo desenvolvido principalmente por Patrick Lynett, da Universidade do Texas. O modelo utiliza equações altamente não-lineares de Boussinesq, equações muito pesadas computacionalmente para serem resolvidas por um só CPU. Os autores do modelo Coulwave decidiram dividir o domínio do problema de forma a que cada processador tratasse de uma parte do problema. A versão paralela é uma versão de Julho de 2005, que o autor cede, a pedido. A versão sequencial não havia sido utilizada previamente pelo LNEC. Neste momento, o modelo em versão sequencial e paralela é utilizado pelo Núcleo de Portos e Estruturas Marítimas (NPE).

3.3.2 Compilação

Este modelo está desenvolvido utilizando a linguagem de programação Fortran e foi compilado com algum sucesso com os compiladores "`ifort`" da Intel, nas estações de trabalho do DHA e com o compilador "`pathscale`" da EKOPath, nos testes de aquisição do *cluster*, e com o compilador XLF da IBM. A compilação com o compilador "`pathscale`" obteve resultados díspares entre as diversas implementações. Com a configuração da Fujistu-Siemens só se conseguiu obter resultados positivos com uma compilação a 32 *bits* e sem as extensões `sse2`.

Nas estações do DHA, foi utilizada a versão 1 da especificação MPI, e nos testes de aquisição, a versão 2.

A compilação com `ifort` é relativamente trivial: "`mpif90 *.f -o pcoulwave`", embora com o compilador "`pathscale`" tenham surgido algumas dificuldades nos testes de aquisição e de aceitação do *cluster*.

3.3.3 Execução

Todas as simulações deste programa devem ser executadas sobre um conjunto de ficheiros pré-existentes ou utilizando os dados exemplificativos incluídos no código fornecido.

No ficheiro "`sim_set.dat`" estão as configurações da simulação. Este ficheiro é constituído por uma série de valores, um por linha, os quais não estão identificados no ficheiro e se encontram parcialmente identificados no manual[1]. O programa, quando iniciado, ainda numa fase de "pré-processamento", questiona o utilizador sobre os valores a utilizar na simulação, mostrando como valores por omissão os que se encontram neste ficheiro. As respostas dadas são guardadas no

[1] Mais informações em: http://ceprofs.tamu.edu/plynett/COULWAVE/ .

ficheiro `sim_set.dat` que será utilizado nesta corrida e ficará disponível para a próxima corrida. Assim, para a mesma simulação, é possível só ter de responder às questões de configuração uma só vez, visto que a confirmação posterior passa a ser só dada pela introdução do carácter '0'. É possível criar um ficheiro, in.txt por exemplo, que contenha a sequência correcta de um 1 e oito 0, um por linha, de forma a que a simulação arranque de imediato com as configurações anteriores. É necessário redireccionar o dito ficheiro para o *input* do Coulwave, passando a execução a efectuar-se da seguinte forma: "`mpiexec -n X ./pcoulwave < in.txt`". Caso não se queira utilizar este método de redirectionamento de *input,* a execução do Coulwave é igual à de outro qualquer programa MPI: "`mpiexec -n X ./pcoulwave`", se for compilado com MPI-2, ou "`mpirun -np X ./pcoulwave`", se for compilado com MPI-1.

São gerados diversos ficheiros de dados que variam consoante o tipo de simulação a executar. A visualização e o tratamento destes ficheiros devem ocorrer utilizando o programa Mathlab. Com o código-fonte existem diversos ficheiros Matlhlab com a extensão "`.m`", que devem ser utilizados: "`bath_loc.m`", "`seqplot.m`", "`maxplot.m`", "`spectrum_1D.m`".

O ficheiro "`bath_loc.m`" tem uma dupla função, a de ler os dados da batimetria a partir dos ficheiros gerados na simulação e/ou de criar a batimetria para o Coulwave, tendo como base os dados do utilizador e guardando os "novos" dados numa forma numérica que o programa Coulwave consegue ler. Este *script* vai criar quatro ficheiros de dados: "`f_topo.dat`", "`size_topo.dat`", "`x_topo.dat`" e "`y_topo.dat`".

O ficheiro "`seqplot.m`" contém o *script* que vai carregar e desenhar o gráfico da simulação numérica, ver exemplo na figura 7. Este *script* pode ser utilizado para se ver o resultado da simulação, mesmo quando esta ainda está a decorrer.

O *script* "`maxplot.m`" vai carregar e desenhar o gráfico da superfície máxima de elevação obtida durante toda a simulação.

Para a geração de espectros em 1HD ou 2HD existem os ficheiros "`spectrum_1D.m`" e "`spectrum_2D.m`", respectivamente.

[1] Disponível em: http://ceprofs.tamu.edu/plynett/COULWAVE/COULWAVE_manual.pdf .

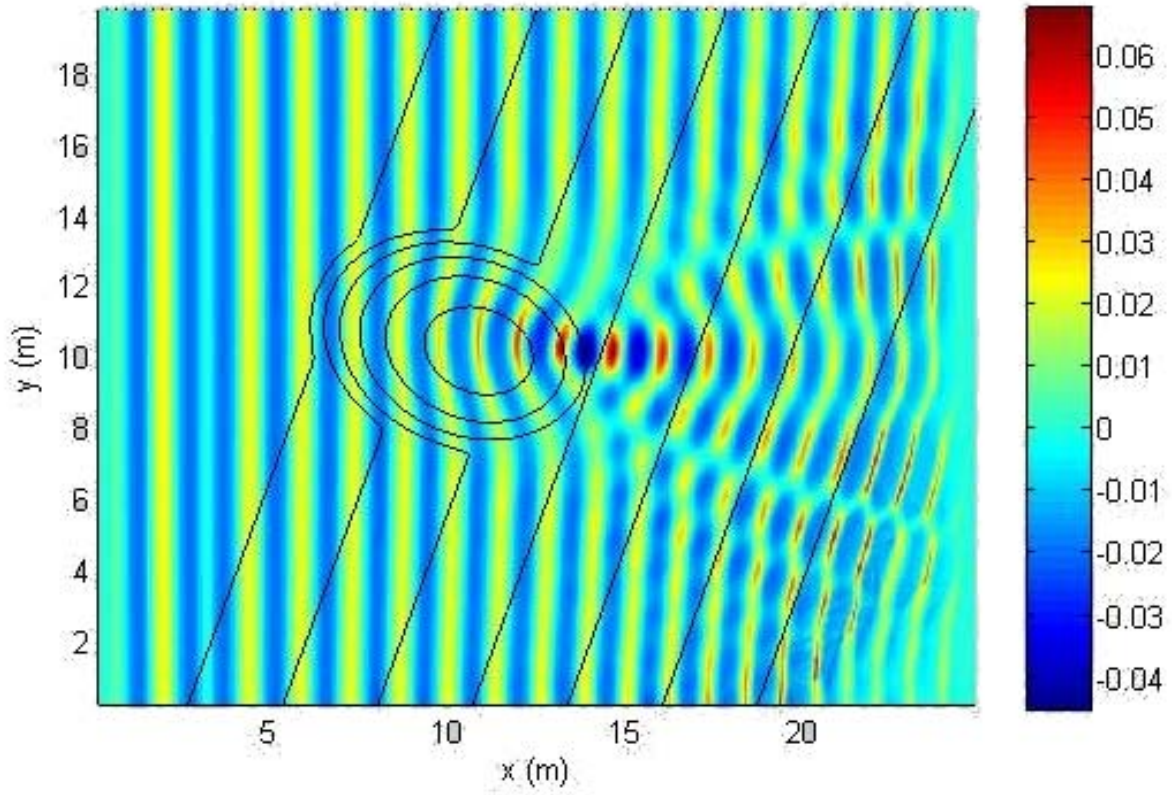

Figura 7 | Exemplo de visualização do output do programa Coulwave

Durante a fase de "pré-processamento", o utilizador será questionado sobre diversos parâmetros da simulação. Em primeiro lugar, o número de processadores e a forma como o domínio vai ser dividido entre os diversos processadores. Apesar da documentação do modelo não advertir sobre a melhor forma de como dividir o domínio, a experiência mostrou que uma divisão de 1 na direcção X e N na direcção Y é a melhor opção. Com outros valores (outras divisões), os resultados obtidos não eram válidos. No entanto, com um elevado número de divisões em Y poderão também ocorrer situações em que os resultados não sejam válidos. Este assunto deve ser melhor explorado no *cluster* Medusa.

Depois desta fase, escolhe-se o tipo de simulação a realizar; o programa pede alguns parâmetros, que variam consoante o tipo de simulação a efectuar. A visualização de resultados deve ser efectuada recorrendo aos *scripts* Mathlab referidos anteriormente.

3.3.4 Resultados

A Análise dos vários testes realizados parece levar à conclusão de que existe um limite de expansibilidade do modelos Coulwave, verificando-se que, a partir de 16 processadores, não se consegue melhorar o desempenho. Isto advém da natureza do código-fonte em si, e do facto de cada processador extra trazer maior acréscimo de comunicações de controlo e de cooperação, que acaba por ter um peso superior ao benefício do seu trabalho. Diversos factores têm influência sobre este número de processadores limite, sendo os mais relevantes o tipo de ligação entre os processadores,

o compilador e os seus parâmetros de compilação, e a biblioteca MPI. Os desenvolvimentos do modelo Coulwave têm sido escassos e passaram a ocorrer somente na versão paralela. Possivelmente esta falta de maturidade do código é a razão dos diversos problemas sentidos, tanto na sua compilação como na execução.

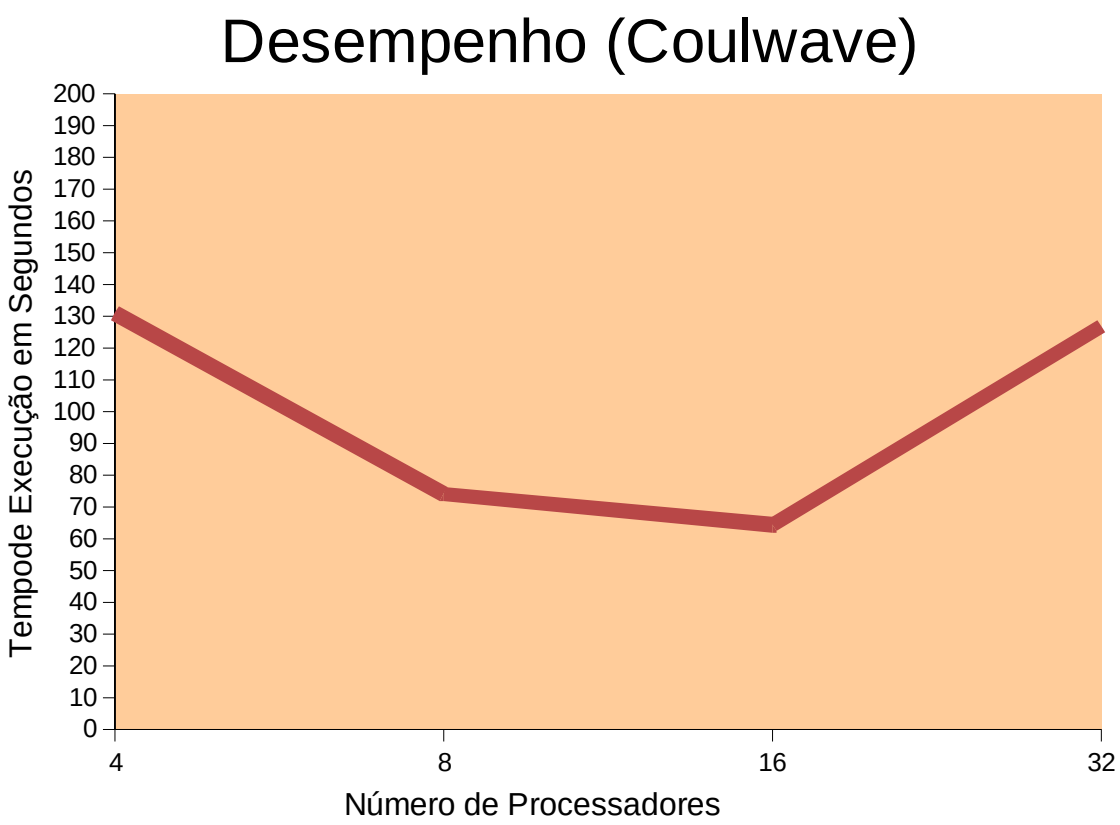

Figura 8 | Resultados conseguidos com o modelo Coulwave nos testes de aquisição do cluster

4 | CONCLUSÕES

O trabalho descrito no relatório teve como objectivo testar se a metodologia paralela seria útil ao LNEC, e ao DHA em particular, de forma a resolver os problemas de forma mais eficiente.

Verificou-se que a metodologia paralela mostrou ser viável para se conseguir obter resultados mais rápidos (ou mais precisos). No entanto, a transformação de programas sequenciais em programas paralelos não é trivial e os resultados poderão não ser os idealizados.

Em todos os modelos testados os resultados foram positivos ao nível do tempo de execução e, no caso do ADCIRC, foram mesmo muito bons.

A instalação do cluster Medusa no LNEC abre grandes perspectivas de se conseguir o aumento do desempenho de diversos programas, das diversas áreas de estudo existentes no Laboratório, pelo que deve ser fomentada a utilização de programas já paralelizados ou avançar com a paralização de programas sequenciais.

No entanto, a disponibilização de tal número de processadores permite também que se avance para soluções baseadas em Computação GRID, em que a paralelização pode ser realizada num nível lógico mais elevado.

Lisboa, LNEC, Novembro de 2007

AUTORIAS

Bruno Lucas
Bolseiro de Iniciação à Investigação Científica.

VISTOS

Maria Alzira Santos
Investigadora Coordenadora
Chefe do Núcleo de Tecnologias de Informação

Rafaela de Saldanha Matos
Investigadora Coordenadora
Directora do
Departamento de Hidráulica e Ambiente

BIBLIOGRAFIA

Fernandes, J. P.; Lucas, B. (2007) - *Validação e Implementação do Medusa: Um Instrumento Científico para Modelações Matemáticas Complexas*. Relatório Interno, Lisboa, LNEC (em preparação).

ANEXOS

ANEXO I – Guia Adcirc

Guia de utilização do *P*ADCIRC

Versão Paralela

Bruno Lucas
blucas@lnec.pt

2007

1 Introdução

O modelo ADCIRC[1], ou "A **AD**vanced **CIRC**ulation model for oceanic, costal and estuarine waters", é um modelo hidrodinâmico utilizado no estudo de águas pouco profundas.

Dependendo do tipo e do detalhe dos dados, o tempo de uma corrida de simulação deste modelo pode chegar a ser bastante elevado, superior a duas semanas. De forma a diminuir o tempo de corrida do modelo foi criada uma versão paralela, o **P***ADCIRC*. Esta versão está desenvolvida na linguagem de programação FORTRAN 90[2] e na paralelização foi utilizada a biblioteca de troca de mensagens MPI, *Message Passing Interface*[3]. O modelo utiliza a versão 1 desta biblioteca; os exemplos que se encontram neste manual utilizaram a versão 1.2.7p1 (implementação MPICH[4]).

2 Compilação

Este guia refere-se à versão 45.11 do modelo ADCIRC. Para outras versões poderá ser necessário recorrer a alguma adaptação dos exemplos presentes neste manual.

Neste guia assume-se que o utilizador tem disponível uma implementação da biblioteca MPI no seu sistema. Se esta não estiver disponível o utilizador poderá falar com o administrador do sistema. Poderá ainda descarregá-la[4] da Internet e instala-la na sua área pessoal.

Depois de se descomprimir o código fonte vai-se encontrar na directoria base do ADCIRC, pelo menos, quatro directorias: `metis`; `prep`; `src` e `work`. Na directoria `work` está o ficheiro `makefile` que deve ser editado. Procure-se a linha:

```
include cmplrflags.mk
```

e substitua-se por:

```
include cmplrflags.mk.mine
```

Deve-se criar o ficheiro `cmprflags.mk.mine` na directoria `work`, podendo basear-se no ficheiro `cmplrflags.mk` que se encontra na mesma directoria. O conteúdo do ficheiro `cmplrflags.mk.mine` deve ser o seguinte:

```
ifeq (1,1)
 FC := #localização do IFORT, por exemplo:
      /opt/intel/fce/9.0/bin/ifort
 PFC := #localização do compilador paralelo (MPI) de FORTRAN
      #por exemplo:
      /opt/mpi/mpiVersion/bin/mpif90
 FFLAGS1 := -O2 -r8 -FI -Vaxlib -Qoption,link,-noinhibit-exec
      -assume byterecl -132
 FFLAGS2 := FFLAGS1
 FFLAGS3 := FFLAGS1
 FFLAGS4 := FFLAGS1
 DA := -DREAL8 -DLINUX -DCSCA
 DP := -DREAL8 -DLINUX -DCSCA -DCMPI
 DPRE := -DREAL8 -DLINUX
 DPRE2 := -DREAL8 -DLINUX -DCMPI
 IMODS := -I
 CC := gcc
 CFLAGS := -I. -O2 -DLINUX
 CLIBS :=
 LIBS := -L ../metis -lmetis
 MSGLIBS :=
endif
```

As *flags* passadas ao compilador têm o seguinte significado[10]: `-O2` (Código optimizado para velocidade); `-r8` (Os valores reais são representados por 8 bytes); `-FI` (Utilização de formato fixo dos ficheiros); `-Vaxlib`[11] (Utilização da algumas funções existentes *Portability Library*); `-Qoption,link,-noinhibit-exec` (Ignorar um determinado *warning* que levaria a um erro interno do *linker*); `-assume byterecl` (Relativo ao formato dos ficheiros); `-132` (A linha termina na coluna 132).

Em seguida, estando na directoria `work`, deve-se executar o comando ***`make all`***. Uma compilação bem sucedida cria os executáveis: `adcirc`; `adcpost`; `adcprep`; `adcprep2` e `padcirc`.

3 Preparação da Corrida

É aconselhável ter uma directoria isolada com uma cópia dos ficheiros de dados para a corrida. Dentro desta directoria devem estar os ficheiros dos parâmetros de definição da corrida (`fort.15`[6]) e o ficheiro da malha de cálculo (`fort.14`[7]). Deve-se copiar os executáveis criados anteriormente (`adcirc`, `adcpost`, `adcprep`, `adcprep2` e `padcirc`) para esta directoria.

4 Pré-Processamento

Encontrando-se na directoria escolhida anteriormente para a corrida o utilizador deve executar o pré-processamento dos dados de forma a permitir a corrida paralela. Para isso execute-se o programa *adcprep*.

Será-lhe perguntado o número de processadores em que se quer fazer a corrida:

```
****************************************
ADCPREP Fortran90 Version 2.0 02/02/2006
Parallel version of ADCIRC Pre-processor
****************************************

Input number of processors for parallel ADCIRC run:
```

Deve introduzir o número desejado, 4 por exemplo. Posteriormente é apresentado o seguinte menu:

```
Preparing input files for subdomains.
Select number or action:
1.  prepall
- Decompose full-domain grid, input, and hot start
files; use default names (fort.14, fort.15, etc).

2.  prepspec
- Same as 1 except user may specify the names of the
input files.

3.  prep14_15
- Use a full domain fort.15 input file to create new
subdomain fort.15 files in existing subdirectories.
Meant for use in continuing a parallel job with
modified input data.  DISCLAIMER: This option
performs domain decomposition again, including
the writing of subdomain fort.14 files.
```

A opção usual é *1* na qual o domínio é decomposto utilizando os ficheiros`fort.14` e `fort.15`.

Se tudo tiver corrido bem, o *output* terminará com:

```
Finished pre-processing input files
```

Na directoria corrente estarão outras directorias de nome`PE000X`, onde `X` varia entre `0` e o número de processadores escolhidos anteriormente.

5 Execução

Para a execução da corrida deve-se correr o comando:

```
mpirun -np X ./padcirc
```

Onde X é o número de processadores escolhidos anteriormente. Caso se queira utilizar várias máquinas, pode-se recorrer ao comando:

```
mpirun -machinefile maquinas.txt -np X ./padcirc
```

Este ficheiro `maquinas.txt` deve ter o formato `Maquina:Processadores`, por exemplo:

```
tucana:4
outramaquina:8
...
...
...
```

Mais informações sobre a utilização de várias máquinas pode ser consultada na documentação do **MPI**[5]. Para a utilização de várias máquinas é aconselhado que o *login* sem *password* por `ssh` seja possível, utilizando por exemplo o `ssh-agent`[12, 8, 9].

6 Pós-Processamento

Depois da execução terminar deve-se correr o programa `adcpost`. No menu de escolha a opção usual é `post`.

```
initializing post-processor
 *********************************
 ADCPOST version 1.5 (02/02/2006)
 Parallel ADCIRC Post-processor
 *********************************

 Select operation

 post = Globalize output files
 compare = Compare outputs in two directories
 diffmerge = Merge difference in two directories
 quit = Exit program
```

Os ficheiros `fort.53/54` resultantes do pós-processamento podem ser utilizados da mesma forma que os ficheiros resultantes de uma corrida em série.

Referências

[1] ADCIRC.
<http://www.marine.unc.edu/C_CATS/adcirc/index.htm>
(Maio de 2006)

[2] Fortran - Wikipedia, the free encyclopedia.
<http://en.wikipedia.org/wiki/Fortran#Fortran_90>
(Maio de 2006)

[3] Message Passing Interface - Wikipedia, the free encyclopedia.
<http://en.wikipedia.org/wiki/Message_Passing_Interface>
(Maio de 2006)

[4] MPICH2 Homepage.
<http://www-unix.mcs.anl.gov/mpi/mpich/> (Maio de 2006)

[5] MPIRun Usage.
<http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chshmem/node120.htm#Node120> (Maio de 2006)

[6] ADCIRC User Manual. <http://www.marine.unc.edu
/C_CATS/adcirc/document/fort_14.html> (Maio de 2006)

[7] ADCIRC User Manual. <http://www.marine.unc.edu
/C_CATS/adcirc/document/fort_15.html> (Maio de 2006)

[8] SSH and ssh-agent. Brian Hatch.
<http://www.securityfocus.com/infocus/1812> (Abril de 2006)

[9] Using SSH and ssh-agent. Mark A. Hershberger.
<http://mah.everybody.org/docs/ssh> (Maio de 2006)

[10] for_main_webhelp. Intel.
<http://www.intel.com/software/products/compilers
/flin/docs/main_for/> (Maio de 2006)

[11] Intel Fortran Compiler For Linux - When to link to the
Portability Library. Intel. <http://www.intel.com/support
/performancetools/fortran/linux/sb/cs-007815.htm>
(Maio de 2006)

[12] Como realizar autenticação sem palavra-chave. Bruno Lucas. Relatório NTI, LNEC. Em preparação.

ANEXO II – Guia Elcirc

Guia de utilização do *p*ELCIRC

Versão Paralela

Bruno Lucas
blucas@lnec.pt

2007

1 Introdução

O modelo ELCIRC[1] é um modelo hidrodinâmico que utiliza uma malha não estruturada para a simulação a 3 dimensões da circulação baroclínica em estuários e zonas costeiras. Este modelo utiliza um algoritmo baseado no método das diferenças finitas, volumes finitos e técnicas Eulerianas-Lagrangianas para resolver as equações de águas pouco-profundas.

De forma a diminuir o tempo de execução das simulações, foi criada uma versão paralela daquele modelo, o **pELCIRC**. A versão paralela está desenvolvida na linguagem de programação FORTRAN 90[2] e na paralelização foi utilizada a biblioteca de troca de mensagens MPI, *Message Passing Interface*[3]. O modelo utiliza a versão 2 desta biblioteca. Os exemplos que se encontram neste manual utilizaram a versão 1.0.4p1 da implementação MPICH2, pelo que poderão não funcionar com outras implementações/versões. Por exemplo a utilização da implementação MPICH1 (versão 1.2.7p1) conduziu a vários problemas.

Neste guia é apresentada a forma de compilar e correr o modelo ELCIRC em modo paralelo.

2 Compilação

Este guia refere-se à versão 1.5 *beta* do modelo pELCIRC. Para outras versões poderá ser necessário recorrer a alguma adaptação dos exemplos presentes neste manual.

Neste guia assume-se que o utilizador tem disponível a implementação da biblioteca MPI no seu sistema. A biblioteca pode ser descarregada[4] da Internet e instalada na sua área pessoal.

Na directoria gerada aquando da descompressão do código fonte do modelo vão estar vários ficheiros e uma directoria, `ParMetis-3.1`. Na directoria `ParMetis-3.1` está o código fonte da biblioteca ParMetis[11], que trata da divisão

da malha utilizada pelo modelo. A malha será dividida em N-divisões, onde N são o número de processadores utilizados na simulação.

O primeiro passo para a compilação do modelo corresponde à compilação desta biblioteca. Para isso é necessário aceder à directoria `ParMetis-3.1`. As opções de compilação estão no ficheiro `Makefile.in`[1]. Se o ficheiro `Makefile.in`, tiver as opções correctas (a nível de compilador, parâmetros de compilação, etc.) basta fazer ***make*** e a biblioteca será gerada. Caso seja necessário alterar alguma das variáveis de compilação, deve-se editar o ficheiro referido. Normalmente basta alterar o compilador (`CC`) e os parametros de compilação (`OPTFLAGS`). Os valores por omissão são: `CC=mpicc`, o compilador C da biblioteca MPI; e `OPTFLAGS=-g`, para compilar com as opções de *debug* ou `OPTFLAGS=-O2` para uma execução optimizada.

Depois de compilada a biblioteca ParMetis, é necessário compilar o modelo Elcirc. Deve-se começar por editar o ficheiro `Makefile`[2] que se encontra da directoria base. No ficheiro encontram-se referências diversas ao ambiente de *hardware* e *software* que se devem encontrar comentados[3]. Deve-se tornar activo o ambiente adequado à máquina em questão, removendo para isso o carácter `#`. Se for necessário criar uma novo ambiente de compilação, a maneira mais fácil de o fazer passa pela cópia de um ambiente existente e a sua alteração e activação (remoção do`#`).

Em seguida mostra-se um exemplo de um ambiente de compilação para máquinas de 64 bits e que possui o compilador de FORTRAN `ifort`.

```
############################################################
# Enviroment for Linux / x86-64 / Intel Compiler / MPICH
############################################################
FCS = ifort #Compilador FORTRAN
FCP = mpif90 #Compilador paralelo (MPI) de FORTRAN
FLD = $(FCP)
#####Flags for debug version
# PPFLAGS = -DDEBUG -DINCLUDE_TIMING
                -DMPIVERSION=1
# FCSFLAGS = -g -mcmodel=medium -assume byterecl
# FCPFLAGS = $(PPFLAGS) -g -assume byterecl
# FLDFLAGS = -g
#####Flags for optimized version
PPFLAGS = -DINCLUDE_TIMING -DMPIVERSION=2
FCSFLAGS = -O2 -mcmodel=medium -assume byterecl
FCPFLAGS = $(PPFLAGS) -O2  -mcmodel=medium
              -assume byterecl
FLDFLAGS = -O2 -mcmodel=medium
#####Libraries
MTSLIBS = -L./ParMetis-3.1 -lparmetis -lmetis
#MTSLIBS = # Not used
#ALTLIBS = # Not used
############################################################
```

[1] Note que o ficheiro a alterar tem o nome `Makefile.in` e não `Makefile` sem extensão, que também existe na mesma directoria.

[2] Desta vez sem extensão.

[3] A linha que começa com o carácter `#` será ignorada.

No exemplo anterior está activa uma configuração optimizada[4]. De seguida, faz-se uma breve explicação das variáveis de compilação do `makefile`.

Compiladores:

- `FCS`: O compilador para o código sequencial;
- `FCP`: O compilador para o código paralelo;
- `FLD`: O *linker* a utilizar, normalmente o mesmo que o compilador paralelo.

Parâmetros de compilação:

- `FCSFLAGS`: Parâmetros para o compilador de código sequencial;
- `FCPFLAGS`: Parâmetros para o compilador de código paralelo;
- `FLDFLAGS`: Parâmetros para o *linker*.
  - § `-g`: Adiciona símbolos de depuração (*debug*);
  - § `-mcmodel=medium`: Serve para assinalar que a secção de dados pode exceder os 2 GB, pelo que é necessário gerar endereçamentos de 64 bits e não de 32 bits. É uma opção disponível somente em sistemas operativos Linux de 64 bits.
  - § `-assume byterecl`: Relativo ao formato dos ficheiros;
  - § `-O2`: Optimização por omissão, optimiza a velocidade do código.
  - § `-O3`: Activa a optimização `O2` e algumas opções de optimização mais agressivas, que podem nem sempre funcionar da forma esperada.
  - § `-fast`: Opção para optimizar a velocidade em todo o programa. É equivalente às opções `-ipo`, `-O3`, `-no-prec-div`, `-static`, e `-xP`. Notar que esta última opção poderá ser incompatível com alguns processadores[10].
  - § `-ipo`: Activa optimização interprecedural entre ficheiros. O compilador expande as funções com código *inline*.
  - § `-no-prec-div`: Aumenta a precisão das operações de divisão com virgula flutuante. Tem impacto sobre a velocidade do programa.
  - § `-static`: Evita a *linkagem* com bibliotecas partilhadas.
  - § `-xP`: Código específico para um tipo de processador.

Parâmetros de Pré-Processamento

- `-DDEBUG`: Indica que a compilação é feita em modo de *Debug*;
- `-DORDRED_SUM`: Activa globalmente as somas ordenadas e produtos escalares *(ordered sums & dot-products)* . **Nota:** esta opção pode diminuir significativamente o desempenho) ;
- `-DINCLUDE_TIMING`: Activar tempo real *(wallclock timming)* do código. Nota: esta opção pode ter um pequeno efeito negativo sobre o desempenho.
- `-DMPIVERSION=2`: Especificação da versão da biblioteca MPI 1 ou 2. Neste guia utiliza-se a versão 2, e é a versão recomendada para este modelo.

[4] Recorrendo a parâmetros como `-O3` ou `-fast` poder-se-á obter um desempenho *teoricamente* superior.

Tendo o ficheiro `Makefile` preparado, basta executar o comando

```
make all
```

para se iniciar a compilação.

Uma compilação bem sucedida cria os executáveis: `pelcirc`; `convert_param`; `ptrack2`; `read_iwrite1`, `read_iwrite2` e `read_output5`.

3 Preparação da Simulação

É aconselhável ter uma directoria isolada com uma cópia dos ficheiros de dados para a corrida. Deve-se copiar os executáveis criados anteriormente (`pelcirc`; `convert_param`; `ptrack2`; `read_iwrite1`, `read_iwrite2` e `read_output5`) para a directoria em questão.

Dentro desta directoria devem estar os ficheiros de dados necessários para a simulação. No mínimo devem estar presentes os ficheiros `hgrid.gr3` `vgrid.in` e `param.in`. Os ficheiros de entrada na versão paralela do ELCIRC são iguais aos da versão série mas com algumas excepções. Na versão série o ficheiro de configuração é o `param.in`, mas para esse mesmo ficheiro funcionar com a versão paralela é necessário fazer a conversão de alguns parametros, como por exemplo do *solver* ou das condições fronteira de força das marés. Essa conversão pode ser realizada utilizando o programa `convert_param`, gerado anteriormente. Para tal é preciso mudar o nome do ficheiro de dados de `param.in` para `ser_param.in` e executar o comando ***`convert_param`*** que irá criar um novo `param.in`, já preparado para a corrida em paralelo.

Quando o ELCIRC é compilado em modo *debug* é necessário incluir um outro elemento no ficheiro `ser_param.in`. No fim desse ficheiro (nova linha) deve ser acrescentado o valor `0` (zero). Esta alteração para além de permitir a execução do programa vai gerar ficheiros `*_dynam`. Caso se corra em modo optimizado não serão gerados tais ficheiros.

Os ficheiros de *hotstart* também têm sintaxes diferentes.

4 Execução

4.1 MPD

Ao contrário da implementação MPICH1 a implementação MPICH2 necessita de um *daemon* para iniciar e gerir os processos MPI nas máquinas, inclusive na máquina local. O *daemon*[5] por omissão é o `MPD`[12]. Para a execução de somente um *deamon* na máquina local pode-se utilizar o seguinte comando[6] em qualquer directoria:

```
mpd &
```

[5] Um programa que corre em *background*, invés de directamente sobre o controlo do utilizador, tratando automaticamente de determinadas funções.

[6] A directoria que contém os executáveis MPICH (`dir_base_mpich/bin/`) deve estar na `$PATH`

Para a execução do modelo em mais do que uma máquina é necessário lançar o processo `mpd` na(s) máquina(s) seleccionadas. Isto pode ser conseguido com o comando:

```
mpd -h tucana -p 12345 --deamon
```

em que o parâmetro `-h` indica o **h**`ost` remoto onde corre o processo `mpd` inicial, neste caso é na máquina `tucana`. O parâmetro `-p` indica a **p**orta onde o processo inicial espera o contacto para adesão ao *anel*[7] de processos `mpd`, neste caso é a porta `12345`. Caso não sejam indicados parâmetros ficarão dois *daemons* isolados, sem a utilidade desejada.

Para se iniciarem vários *daemons* pode-se recorrer ao comando:

```
mpdboot -n 3 -f mpd.hosts
```

em que `-n` indica o **n**úmero de processos `mpd` a lançar, neste caso 3. O parâmetro `-f` indica o **f**icheiro onde estão indicados o nome das máquinas em que irão correr os *daemons*. Este ficheiro tem o formato de uma nome de máquina por linha.

As máquinas existentes no ficheiro devem permitir acesso `ssh` sem pedido de *passwords*, recorrendo por exemplo a um `ssh-agent`[13, 6, 7].

Alguns comandos a ter em conta são: `mpdtrace -l` (mostra os nomes e portas de todos os `mpd` do *anel*); `mpdringtest` (testa o *anel* formado pelos diversos *daemons* `mpd`); `mpdcleanup` e `mpdallexit` (executados por esta ordem terminam todos os *daemons* `mpd` de forma ordeira); `mpdlistjobs` (lista de trabalhos por terminar).

4.2 pElcirc

No modelo pELCIRC, não é necessário[8] executar explicitamente um comando para o pré-processamento dos dados.

Para a execução da corrida deve-se correr o comando:

```
mpiexec -np X ./pelcirc
```

Onde X é o número de processadores escolhidos para a execução. Este comando `mpiexec` vem substituir o comando `mpirun` da implementação MPICH1, embora este ainda seja válido e funcione por questões de retrocompatibiliade. Caso se queira utilizar várias máquinas, deve-se ter previamente preparado um *daemon* `mpd` em cada uma dessas máquinas, ver secção anterior.

5 Pós-Processamento

Os ficheiros resultantes da execução em paralelo podem ser utilizados da mesma forma que os ficheiros resultantes de uma corrida em série, utilizando as versões dos programas `read_iwrite1`, `read_iwrite2` e `read_output5` que são distribuídas com o código paralelo do Elcirc.

[7] É considerado que existe um anel virtual com todas as máquinas onde se encontra o `mpd` em execução.

[8] Se o `param.in` já estiver convertido para o formato paralelo, caso contrário é necessário executar o *convert_param*, ver secção 3.

Referências

[1] ELCIRC.
<www.ccalmr.ogi.edu/CORIE/modeling/elcirc/>
(Junho de 2006)

[2] Fortran - Wikipedia, the free encyclopedia.
<http://en.wikipedia.org/wiki/Fortran#Fortran_90>
(Maio de 2006)

[3] Message Passing Interface - Wikipedia, the free encyclopedia.
<http://en.wikipedia.org/wiki/Message_Passing_Interface>
(Maio de 2006)

[4] MPICH2 Homepage.
<http://www-unix.mcs.anl.gov/mpi/mpich/> (Maio de 2006)

[5] MPIRun Usage.
<http://www-unix.mcs.anl.gov/mpi/mpich1/docs/mpichman-chshmem/node120.htm#Node120> (Maio de 2006)

[6] SSH and ssh-agent. Brian Hatch.
<http://www.securityfocus.com/infocus/1812> (Abril de 2006)

[7] Using SSH and ssh-agent. Mark A. Hershberger.
<http://mah.everybody.org/docs/ssh> (Maio de 2006)

[8] for_main_webhelp. Intel.
<http://www.intel.com/software/products/compilers
/flin/docs/main_for/> (Maio de 2006)

[9] Intel Fortran Compiler For Linux - When to link to the Portability Library. Intel. <http://www.intel.com/support
/performancetools/fortran/linux/sb/cs-007815.htm>
(Maio de 2006)

[10] for_main_webhelp. Intel.
<http://www.intel.com/software/products/compilers
/flin/docs/main%5Ffor/mergedProjects/copts_for/
common_options/option_fast.htm>

[11] METIS - Family of Multilevel Partitioning Algorithms. Karypls Labs. <http://glaros.dtc.umn.edu/gkhome/views/metis/parmetis>
(Junho de 2006)

[12] Gropp, W. et al 2006. MPICH2 User's Guide, version 1.0.4. Argonne National Laboratory. Disponível em <www-unix.mcs.anl.gov/mpi/mpich2/downloads/mpich2-doc-user.pdf>

[13] Como realizar autenticação sem palavra-chave. Bruno Lucas. Relatório NTI, LNEC. Em preparação.

ANEXO III - Guia para a autenticação segura sem palavras-chave

Como realizar autenticação sem palavra-chave

Bruno Lucas
blucas@lnec.pt

5 de Setembro de 2007

1 Introdução

Em algumas situações é necessário automatizar a autenticação (*login*) numa outra máquina remota evitando a escrita de palavra-passe (*password*).

O método da chave pública pode ser utilizado na autenticação remota de utilizadores, em alternativa à utilização de *passwords*. Este método tem a vantagem de evitar que as *passwords* sejam constantemente digitadas, o que aumenta o nível de segurança (evitando testemunhas observadoras ou *keyloggers*), incrementando ainda a satisfação do utilizador por digitar menos *passwords*.

A criptografia de chave pública permite comunicar de maneira segura sem que tenha sido partilhada, de forma prévia, uma chave combinada entre os interlocutores. Para tal existem duas chaves, uma pública e uma privada, que se encontram relacionadas matematicamente. É muito difícil descobrir a chave privada a partir da chave pública e de dependendo dos algoritmos e do tamanho da chave pode ser mesmo impossível.

O método da chave pública tem uma larga utilização por exemplo na cifração de mensagens ou na autenticação das mesmas. A chave privada é mantida escondida, enquanto a chave pública é amplamente distribuída. Um documento, mensagem, etc., cifrado com uma chave pública $\mathbf{A}^{pb}$ só pode ser descodificado com pela pessoa que possuir a chave privada $\mathbf{A}^{pv}$. Tal como uma mensagem cifrada com a chave privada $\mathbf{B}^{pv}$ só poderá ser decifrada por quem tenha a chave pública $\mathbf{B}^{pb}$.

A máquina remota deve possuir activo um servidor SSH, mais propria-

mente um *deamon* (`sshd`). Esse servidor SSH tem uma lista de chaves públicas de utilizadores nas quais confia, se um utilizador conseguir provar que possui a chave privada correspondente, é-lhe dado acesso sem pedido de palavras-passe. Neste guia é exemplificado como conseguir a autenticação remota de duas formas. A forma **segura** encontra-se no **método A**. Esta forma evita que caso a máquina *base* seja comprometida o atacante consiga aceder a todas as outras máquinas. O **método B** é mais **inseguro** mas de execução mais simples. É aconselhada a utilização do método A.

Nota: Neste guia são utilizados vários nomes de máquinas e utilizadores que devem ser substituídos pelos que se adequam à situação em questão:

- `meuNome`: nome de utilizador;
- `minhaMaquina`: nome da máquina local;
- `outraMaquina`: nome da máquina remota.

2 Método A

2.1 Geração de Chaves

O primeiro passo para se automatizar o *login* noutra máquina remota é a geração de chaves (públicas e privadas). Depois de se estar ligado à máquina a partir da qual deseja efectuar os *logins* deve-se executar os seguintes comandos:

```
ssh-keygen -t rsa
```

Sendo o *output* similar ao seguinte:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/meuNome/.ssh/id_rsa):
<ENTER>
Enter passphrase (empty for no passphrase):<password[1]>
Enter same passphrase again:<password>
Your identification has been saved in
     /home/meuNome/.ssh/id_rsa.
Your public key has been saved in
      /home/meuNome/.ssh/id_rsa.pub.
The key fingerprint is:
68:c9:c6:8a:63:5a:60:a0:d7:50:11:58:b0:31:6f:76
     meuNome@minhaMaquina.lnec.pt
```

[1] Esta palavra-passe não necessita de ser a mesma utilizada na autenticação normal na máquina remota.

Poderá ser verificado que na directoria `$HOME/.ssh` se encontram as duas chaves geradas anteriormente. A chave `id_rsa` é a chave privada que se encontra protegida com a *password* introduzida anteriormente. A chave `id_rsa.pub` é a chave pública que pode ser difundida livremente, visto não apresentar dados comprometedores.

Nota: Existem três tipos de chaves SSH: RSA[3]; DSA[4]. O comando `ssh-keygen` permite criar qualquer uma delas, alterando o argumento da *flag* `-t` para `rsa` ou `dsa` respectivamente.

2.2 Colocação de chaves

2.2.1 Transferência de chaves entre máquinas

Caso a área pessoal (`/home/meuNome`) seja partilhada entre as diversas máquinas a que deseja aceder pode-se saltar para o ponto 2.2.2. Em caso contrário deve transferir a chave pública (`id_rsa.pub`) para a(s) máquina(s) remota(s).

```
minhaMaquina$ cd $HOME/.ssh
minhaMaquina$ scp id_rsa.pub outraMaquina:id_rsa_mM.pub
meuNome@outraMaquina's password: <Introduzir password>
```

Na máquina remota deve-se criar a directoria `.ssh`, onde ficarão as informações sobre chaves.

```
minhaMaquina$ ssh outraMaquina
meuNome@outraMaquina's password:<Introduzir password>
outraMaquina$ mkdir .shh
outraMaquina$ chmod 700 .shh
outraMaquina$ cd .shh
```

2.2.2 Chaves autorizadas

É necessário adicionar a chave pública anteriormente gerada à lista de chaves autorizadas.

```
outraMaquina$ cat ../id_rsa_mM.pub » authorized_keys
outraMaquina$ chmod 600 authorized_keys
outraMaquina$ exit
```

2.3 Activação do agente

O `ssh-agent` é um programa que armazena as chaves privadas que são utilizadas no método de autenticação por chave pública. O agente permite a execução de uma série de comandos remotos (sobre SSH) sem que seja necessário realizar a autenticação para cada comando.

Pode-se iniciar o agente **sempre** que se inicia a sessão, ou somente quando é preciso.

2.3.1 Utilização esporádica

Para uma utilização pouco frequente é possível simplesmente executar o comando a partir de uma linha de comandos:

```
ssh-agent bash
```

O qual inicia o agente (`ssh-agent`) com uma nova linha de comandos (`bash`). O programa iniciado com o `ssh-agent` poderá ser a sua linha de comandos favorita ou por qualquer outro programa. Quando o programa iniciado com o agente terminar o agente *morrerá*. Digitado `exit` na nova linha de comandos esta terminará assim como a execução do agente.

Na nova linha de comandos pode-se executar a autenticação junto da máquina remota ou qualquer outro tipo de programa. Nesta linha de comandos o agente está activo e sempre que for necessário uma ligação SSH a autenticação é tratada automaticamente.

Depois de iniciar o agente é necessário adicionar a chave ao mesmo para que este possa autenticar o utilizador. Isto deve ser feito com o comando:

```
ssh-add
Enter passphrase for /home/meuNome/.ssh/id_rsa:<password>
Identity added:  /home/meuNome/.ssh/id_rsa
      (/home/meuNome/.ssh/id_rsa)
```

Este comando adiciona a chave por omissão (`$HOME/.ssh/id_rsa` ou `$HOME/.ssh/id_dsa`). Se a chave a adicionar estiver noutra localização deve-se acrescentar ao comando anterior o caminho para a chave em questão, por exemplo `ssh-add $HOME/minhasChaves/minhaChave`. Se esta chave estiver protegida com uma *password*, e deve estar, terá de introduzi-la neste passo. Esta *password* é a que introduziu aquando da geração das chaves através do método B, secção 2.1. Não será necessário digitar mais *passwords* ou *passphrases* enquanto o agente estiver *vivo*.

Para saber que chaves estão activas no agente pode recorrer ao comando `ssh-add -l` e para apagar uma determinada chave do agente deve utilizar o comando `ssh-add -d <chave>`.

Quando o programa iniciado com o agente terminar este terminará também.

2.3.2 Utilização permanente

Caso seja necessário uma utilização mais frequente do agente, pode-se iniciar o mesmo no inicio da sessão. Mas será-lhe sempre pedido que introduza a *passphrase*

que protege a sua chave. Isto acontece de cada fez que abre uma nova linha de comandos por exemplo.

A melhor solução é de adicionar as seguintes linhas ao seu ficheiro `$HOME/.bash_profile`:

```
#SSH-AGENT
if [ -z "$SSH_AUTH_SOCK" -a -x "ssh-agent" ]; then
    eval `ssh-agent -s`
    ssh-add $HOME/.ssh/id_rsa
    trap "kill $SSH_AGENT_PID" 0
fi
```

Nota: na linha 3 depois do comando `eval` está um acento grave, o comando a executar e outro acento grave, ` <comando> `.

Para que o agente termine a sua execução aquando do término da sessão deve adicionar ao seu ficheiro `$HOME/.logout` as seguintes linhas:

```
#SSH-AGENT
if [ ${SSH_AGENT_PID+1} == 1 ]; then
    ssh-add -D
    eval `ssh-agent -k`
fi
```

Nota: Sendo a máquina local a máquina `MqL` e as máquinas remotas `MqR1` e `MqR2`. Se estiver autenticado na máquina remota `MqR1` (ligado via `ssh` de `MqL`) e quiser aceder a outra máquina remota `MqR2` pode recorrer a mais uma conexão `ssh`, por exemplo `ssh MqR2`, e deverá conseguir aceder sem ter de digitar *passwords*, visto o agente estar activo. Caso isso não aconteça pode ser necessário alterar as configurações de *forwarding* do agente. Para tal deve editar (ou criar se não existir) o ficheiro `$HOME/.ssh/config` para que permita o acesso a uma terceira (quarta, quinta, etc.) máquina. O conteúdo do ficheiro deve ser o seguinte:

```
Host *
      ForwardAgent yes
```

Alterado * para o nome da máquina remota pode-se configurar o encaminhamento automático do agente somente para algumas máquinas. Mais informações sobre este assunto podem ser encontradas aqui[5].

Nota: Normalmente em qualquer sistema Unix/Linux é possível escrever na linha de comandos `man <nome_do_comando>` e obter informação sobre o próprio comando, o que faz, que opções aceita, etc.

3 Método B

3.1 Gerar chaves

O primeiro passo para se automatizar o *login* numa máquina remota é a geração de chaves (públicas e privadas). Depois de se estar ligado à máquina a partir da qual deseja efectuar os *logins* deve-se executar os seguintes comandos:

```
ssh-keygen -t rsa
```

Sendo o *output* similar ao seguinte:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/meuNome/.ssh/id_rsa):
      <ENTER>
Enter passphrase (empty for no passphrase): <ENTER>
Enter same passphrase again:<ENTER>
Your identification has been saved in
      /home/meuNome/.ssh/id_rsa.
Your public key has been saved in
      /home/meuNome/.ssh/id_rsa.pub.
The key fingerprint is:
68:c9:c6:8a:63:5a:60:a0:d7:50:11:58:b0:31:6f:76
      meuNome@minhaMaquina.lnec.pt
```

Poderá ser verificado que na directoria `$HOME/.ssh` se encontram as duas chaves geradas anteriormente. A chave `id_rsa.pub` é a chave pública que pode ser difundida livremente, visto não apresentar dados comprometedores. A chave`id_rsa` é a chave privada que deve estar segura.

Nota: Existem três tipos de chaves SSH: RSA[3]; DSA[4]. O comando`ssh-keygen` permite criar qualquer uma delas, alterando o argumento da *flag* `-t` para `rsa` ou `dsa` respectivamente.

3.2 Colocação de Chaves

Este passo é exactamente igual ao passo 2.2 do método A. Depois de efectuar a colocação das chaves poderá tentar efectuar o *login* remoto (`ssh outraMaquina`) e verificar que não deve ser pedida *password*.

Referências

[1] SSH and ssh-agent. Brian Hatch.
<http://www.securityfocus.com/infocus/1812> (Abril de 2006)

[2] Using SSH and ssh-agent. Mark A. Hershberger. <http://mah.everybody.org/docs/ssh> (Maio de 2006)

[3] RSA - Wikipedia, the free encyclopedia. Wikipedia. <http://en.wikipedia.org/wiki/RSA> (Julho de 2006)

[4] Digital Signature Algorithm - Wikipedia, the free encyclopedia. Wikipedia. <http://en.wikipedia.org/wiki/Digital_Signature_Algorithm> (Julho de 2006)

[5] An Ilustrated Guide to SSH Agent Forwarding. Stephen J. Friedl. <http://www.unixwiz.net/techtips/ssh-agent-forwarding.html> (Agosto 2006)

[6] Public-Key criptography - Wikipedia, the free encyclopedia. Wikipedia. <http://en.wikipedia.org/wiki/Public_key> (Agosto de 2006)

Anexo IV – Programa de Trabalhos

Ministério das Obras Públicas, Transportes e Comunicações
Laboratório Nacional de Engenharia Civil
Departamento de Hidráulica
Núcleo de Tecnologias da Informação

Plano de Trabalhos do Bolseiro de Iniciação à Investigação Científica

Bruno Lucas

Lisboa, Junho de 2006

Conteúdo

1 Introdução

Este documento apresenta o Plano de Trabalhos do Bolseiro de Iniciação à Investigação Científica, Eng.º Bruno Galvão Lucas.

Bruno Manuel Galvão Lucas terminou, em Janeiro de 2005, a licenciatura em Engenharia Informática, pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL). O projecto final de curso foi realizado no INETI (Instituto Nacional de Engenharia, Tecnologia e Inovação) sob o tema "Piloto de inteligência artificial para pilotar simulador".

Actualmente está inscrito no Mestrado em Engenharia Informática, perfil Sistemas de Informação, na FCT-UNL. A parte escolar do mestrado, um ano lectivo, encontra-se na fase final. O trabalho de dissertação, que terminará no próximo ano lectivo, 2006/07, terá como base o tema "Simulações Baseadas em Agentes em Sistemas de Informação Geográfica".

A dissertação tem como base o trabalho realizado no período posterior à licenciatura durante o qual esteve integrado no projecto MAGIC (Proc. 0602/14/114560). O Bolseiro realizou trabalho com a Doutora Armanda Rodrigues, antiga colaboradora do LNEC, e com o Eng. António Gonçalves, actualmente Bolseiro de Doutoramento LNEC/FCT do NTI.

A 15 de Fevereiro de 2006, o signatário entrou no NTI como Bolseiro de Iniciação à Investigação Científica, após ter sido seleccionado no concurso aberto para a atribuição de uma Bolsa LNEC de Iniciação à Investigação na área Sistemas de Informação.

O Bolseiro tem vindo a desenvolver trabalho na área da paralelização de modelos hidráulicos, de forma a obter melhor desempenho destes modelos numéricos. Este desempenho será aumentado especialmente quando estiver à disposição o futuro *cluster* do LNEC.

2 Actividades de Investigação

2.1 Investigação programada

A actividade do Bolseiro integra-se no Programa de Investigação Programada 2005-2008 do LNEC.

Desde a sua entrada no NTI o bolseiro tem vindo a realizar trabalho no âmbito do projecto "Distribuição e paralelização de modelos numéricos em Hidráulica e Ambiente" (Proc. 0602/11/16283).

Prevê-se a apresentação de propostas à FCT para o financiamento de projectos no âmbito da Iniciativa Nacional GRID. A experiência adquirida com esta actividade permitirá ao bolseiro e ao LNEC aplicar os conhecimentos adquiridos no desenvolvimento/implementação da resoluções de problemas concretos (estudos por contrato) que venham a ser solicitados.

O Bolseiro Bruno Lucas vai desenvolver a sua actividade no domínio da Computação Paralela aplicada à Hidráulica. Nesse âmbito desenvolveu trabalho com as versões paralelas dos modelos hidráulicos:

- ADCIRC;
- COULWAVE;
- ELCIRC.

O modelo ADCIRC é um modelo hidrodinâmico utilizado no estudo de águas pouco profundas, o seu método de cálculo utiliza elementos finitos. O modelo ELCIRC é também um modelo hidrodinâmico para o estudo de águas pouco profundas, mas este modelo utiliza um algoritmo de diferenças finitas. O modelo COULWAVE é um modelo de ondas.

Estes modelos encontram-se em diferentes fases de paralelização. Desde a sua entrada no LNEC, o Bolseiro familiarizou-se com estes modelos, e desenvolveu tarefas que levaram à sua compilação e execução.

O bolseiro irá elaborar guias de utilização das versões paralelas dos modelos: ADCIRC; ELCIRC; COULWAVE.

Está ainda previsto que o bolseiro disponibilize uma metodologia que permita a execução sistemática de um modelo, com alteração da parametrização, de forma automática e distribuída (paralela).

O Bolseiro irá seleccionar e paralelizar modelos simples, da área da hidráulica e ambiente. Estes modelos tirarão partido do *cluster* que se encontra em processo de aquisição.Os resultados desses modelos serão confrontados com os resultados sequenciais.

O Bolseiro irá participar na elaboração dos *benchmarks* que serão utilizados como critério de escolha na compra do equipamento para o *cluster* do LNEC.

2.2 Investigação por contrato

O Bolseiro participará em actividades de investigação por contrato no âmbito de projectos a serem desenvolvidos no NTI.

3 Plano de Formação

3.1 Aquisição e Aprofundamento de Conhecimentos Científicos e Técnicos

Matérias de Base e de Especialidade

No âmbito da sua actividade, o Bolseiro Bruno Lucas tem aprofundado os seus conhecimentos na área da Informática e Hidráulica, nomeadamente a nível da Computação Paralela e Distribuída.

3.2 Frequência de Cursos, Seminários e Outras Acções

O Bolseiro frequenta neste momento o Mestrado em Engenharia Informática, na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

É do interesse do bolseiro vir a participar em congressos e seminários nacionais e internacionais que estejam relacionados com a sua área de formação e com a sua área de especialização.

Prevê-se que o Bolseiro frequente o curso do INA (Instituto Nacional de Administração): "Análise de Sistemas Orientada por Objectos", de forma a aprofundar o seu conhecimento no processo de análise e desenho de sistemas informáticos.

3.3 Contacto com Outros Departamentos ou Instituições

O Bolseiro tem trabalhado vários investigadores de diversos núcleos do Departamento de Hidráulica e Ambiente (NPE e NEC) para os quais está a trabalhar com os modelos anteriormente referidos. Poderá ainda a vir ter contactos com outros núcleos/departamentos do LNEC devido ao interesse na utilização de uma metodologia paralela da resolução dos problemas, nomeadamente em áreas como as barragens e os sismos.

O Bolseiro tem mantido contacto com vários professores da Universidade Nova de Lisboa especialistas na área da computação paralela e distribuída.

4 Trabalho de Investigação Científica de Especialização

O Bolseiro pretende desenvolver uma dissertação de doutoramento na área da computação paralela e distribuída, área na qual o LNEC tem interesse em possuir competências. O Bolseiro pretende identificar possíveis problemas que possam ser alvo da sua tese de doutoramento.

5 Elaboração de relatórios anuais

O Bolseiro irá elaborar relatórios anuais da actividade desenvolvida durante o período de duração da Bolsa de Iniciação à Investigação Científica.

6 Orientação

A actividade de formação do BIIC será orientada pela Doutora Maria Alzira Santos, chefe do Núcleo de Tecnologias da Informação, tendo como co-orientador a Doutora Anabela Oliveira do Núcleo de Estuários e Zonas Costeiras.

Lisboa, Laboratório Nacional de Engenharia Civil, Junho de 2006

Maria Alzira Santos
Investigadora Coordenadora

Anabela Oliveira
Investigadora Auxiliar

Bruno Galvão Lucas
Bolseiro de Iniciação à Investigação Científica

Anexo

Plano de trabalhos apresentado ao bolseiro no início do estágio

A qualidade e a dimensão dos estudos apoiados por modelação matemática no domínio da Hidráulica são frequentemente dependentes da capacidade computacional disponível. Limitações de CPU são frequentes em várias áreas de actividade no DHA, conduzindo por vezes ao recurso a simplificações dos processos a serem resolvidos nos modelos matemáticos e/ou a discretizações espaciais e temporais grosseiras. A computação paralela tem surgido em todo o mundo como uma resposta adequada para resolver estas limitações. Dada a disponibilidade de diversos equipamentos de computação (*workstations*) com vários processadores, a aquisição próxima de um *cluster* de computadores no LNEC para computação de elevado desempenho, bem como a ligação do LNEC à rede de computação em grelha nacional, justifica-se o investimento quer na formação dos vários grupos de modeladores do DHA no uso desta ferramenta, quer na adaptação dos instrumentos de modelação actualmente disponíveis e em preparação para a computação paralela. Este bolseiro, com os seus conhecimentos em programação, terá por objectivo do seu trabalho a paralelização de alguns dos modelos mais usados no Departamento de Hidráulica e Ambiente.

Tarefas	1º Trimestre	2º Trimestre	3º Trimestre	4º Trimestre
1. Familiarização com os modelos existentes				
2. Formação em computação paralela e distribuída				
3. Utilização de modelos seleccionados em modo paralelo (códigos já paralelizados)				
4. Selecção de modelos simples para paralelização (em várias áreas de hidráulica e ambiente)				
5. Desenvolvimento dos modelos seleccionados em modo paralelo				
6. Testes destes modelos em modo paralelo e sequencial				
7. Avaliação do interesse e viabilidade da distribuição e paralelização de modelos numéricos na área da hidráulica fluvial e das águas subterrâneas.				

Figura 1: Cronograma do plano de trabalhos.