Computational Issues in Coupled Climate and Multiphysics Modelling

J. Walter Larson
ANU Supercomputer Facility and APAC
The Australian National University

Presented at the CSIRO Mk3L Workshop
University of New South Wales, 26 May 2006
Overview

• A few comments on climate

• What distinguishes climate models as HPC applications (Part I)

• Some comments on high-performance computing (HPC)
  • Approaches
  • Trends

• How these comments apply to CSIRO Mk3L

• Multiphysics models, or what distinguishes climate models as HPC applications (Part II)

• Conclusions
Climate Defined

**Old Adage:** Climate is what you expect, weather is what you get

**More Rigourously:** Climate is a set of statistics computed from instantaneous samples of data taken from the state of the earth system over a specified period of time

- The sampling period is typically many years (30 being the standard used by climatologists), and depends on what one is studying
  - Temporal subsampling (e.g., long-term monthly/seasonal averages)
  - Geographic subsampling common (e.g., regional and zonal averages)
- Statistics computed include moments (e.g., μ and σ), interquantile ranges, extremes, probability density functions, *et cetera*
- Derived diagnostics computed from data (e.g., meridional circulation)
Climate Modelling

Given our definition of climate, how do we proceed?

- We know our system has limited predictability (about 10 days with 2/3 confidence)

- There really isn’t a set of equations of evolution for the climate

- There are, however, equations of evolution for the components of the climate system (e.g., the primitive equations for atmosphere and ocean general circulation models (GCMs))

- Integration of these equations, recording the system’s history and subsequent analysis can be performed to compute a climate

Thus, climate models are engines for generating climate data, and the faster a model can run, the more data it can generate.
High-Performance Computing Basics
The von Neumann Machine

- Simplified view of a computer
- *Attributed* to von Neumann, but there were others before him
- Over time, a disparity between processor clock cycle speeds (now GHz) and memory access times has widened
- This presents a problem as applications now typically get only a small percentage of theoretical peak performance
- The basic strategy to overcome the clock speed / memory access time disparity
- Place one or more intermediate levels of *faster* (and thus more expensive!) memory between the cpu and memory
- Memory is mapped into *cache lines*
- Data is staged in the cache
- When memory is requested by the processor, the cache is checked. If available, in-cache, it is accessed more quickly
High-Performance Computing

- Integrating the primitive equations is computationally intensive, and climate modelling is considered a grand challenge application in computing.

- The main enabling technologies are faster processors, and parallel computing.
  - Vector computing
  - Shared-memory parallelism (e.g., OpenMP)
  - Distributed-memory parallelism
  - Context-based parallelism (e.g., pthreads)
Vector Processors

• First form of parallelism to get viable commercial HPC implementation (Cray-I, 1977)
• Basic idea is to use vector instructions on vector registers to parallelise operations by performing the same operation repeatedly (pipelining) on many numbers as they flow through the register
• This approach reduces overhead associated with decoding instructions
• Vector processors have multiple vector registers
• Examples: NEC SX-6, Cray X-1

Above: Simplified conceptual view. Actual implementation is more complex and involves pipelining.
Shared-Memory Parallelism

- Multiple processors share one memory bank, and can access any element of memory (1 address space)
- Access is via a shared bus
- Parallelism typically expressed in code via compiler directives (e.g., OpenMP)
- Advantage: easier to implement
- Disadvantage: less precise control over parallelism and this can impact scalability
- Example: High-end PC with multiple processors

Uniform Memory Access Machine
Distributed-Memory Parallelism

- Also known as *message passing parallelism*
- Each Processor has its own independent memory unit
- Data exchanges occur by communication via interconnect or “switch”
- Disadvantage: harder to implement as algorithms must contain *communication* in addition to computation (e.g., using the Message Passing Interface (MPI) standard)
- Advantage: more control over parallelism can yield greater scalability
- Example: APAC’s lc cluster
“Hybrid” Parallelism

- Combination of Shared-Memory and Message-Passing Approaches
- Example: APAC’s ac (SGI Altix)
Performance Metrics

- Raw performance can be measured in terms of \textit{floating point operations per second} (FLOPS), and this is why one hears the terms
  - MegaFlops (MFLOPS) - \textit{Million} floating point operations per second
  - GigaFlops (GFLOPS) - \textit{Billion} floating point operations per second
  - TeraFlops (TFLOPS) - \textit{Trillion} floating point operations per second
  - PetaFlops (PFLOPS) - \textit{Quadrillion} floating point operations per second (as yet unattained)

- Parallel performance can be measured in terms of scalability (i.e. speedup factor on \( N \) processors versus a single processor)

- Ultimately, however, the performance figure of merit is \textit{throughput}. That is, the amount of simulated time per unit of wall-clock time (e.g., model years per wall-clock day)
Performance Portability

- Building high-performance numerical codes on modern computer architectures involves balancing various factors to maximise performance
  - Single-processor performance (cache / pipelining)
  - Multiple levels of parallelism

If this all sounds difficult...it is!
Computing Trends
Trend: Multicore Processors

- Chip manufacturers are now putting multiple compute cores on a single processor
- You can buy them now (even in laptops)
- These multiple cores within a processor share the same memory hierarchy (i.e., caches and memory)
- Conventional wisdom says they can be programmed as SMP’s (i.e., using OpenMP)
- Soon one may see chips with as many as 16 or 32 cores!
Trend: Grid Computing

- Grid computing grew out of the computer science fields of metacomputing and distributed computing.

- Original hope was to build virtual parallel computers from geographically distributed machines.

- Main successes, however, are in farming out manageable tasks with good data locality (e.g., projects such as seti@home and climateprediction.net).

- A computational grid is a collection of online resources (compute nodes, data stores, instruments) linked together with network fabric, whose usage is managed by a virtual organisation comprising the various stakeholders.

- Term originates from analogy to electrical power grids.
Earth System Grid (ESG)

• A union of geographically-distributed supercomputers and large-scale data servers and analysis engines

• All of this available to ESG certificate holders via a web portal (N.B., registration is required)

• Funded by US Department of Energy 2001-present

• Web site: http://earthsystemgrid.org
Trend: Software Components

- A software component is an atomic unit of software that performs some useful function.

- A component has well-defined interfaces for interacting with the outside world (such as other components). These interfaces can transcend programming language barriers.

- A framework is a software mechanism that allows one to compose components to form applications.

- Composition can be done either through scripting or through a graphical user interface (GUI).

- Examples include CORBA, COM, DCOM, JavaBeans, ...
Classic Analogue to Components

Land Model

Atmosphere Model

Sea Ice Model

Ocean Model
Common Component Architecture

- Goal: Component specification standards for high-performance computing that transcends language barriers
- CCA components have *ports*
  - “uses” are the means by which one component calls another
  - “provides” ports are implemented by the component
- CCA uses a *peer component* model in which components interact solely via port connections
- Port connection from a ‘uses’ to a ‘provides’ define a caller/callee relationship
- Web site [http://cca-forum.org](http://cca-forum.org)
CCA and Babel

- CCA implements language interoperability between components through use of a Scientific Interface Definition Language (SIDL).
- SIDL is important because it offers science friendly datatypes not normally found in commercial IDLs (e.g., complex numbers and multidimensional arrays).
- Glue code is generated from component interface definitions written in SIDL by the Babel language interoperability tool.
- SIDL and Babel are usable in a non-component context.
- [http://www.llnl.gov/casc/components/babel.html](http://www.llnl.gov/casc/components/babel.html)

Babel can process function and subroutine interface descriptions written in SIDL to create glue code to allow code written in one language to call code written in another.
A Simple Example

Atmosphere and Ocean exchange data with Coupler at regular intervals.

Coupler performs intermesh interpolation implemented as linear transformation.

Parallel Coupling components are leveraged to do the work!
Trend: Python

- Python is an easy-to-learn object-oriented scripting language
- Web site: [http://python.org](http://python.org)
- Highly portable, running on Linux, Unix, Mac OS X, and even Palm PDA’s and Nokia phones
- Highly interoperable with other languages such as C, C++, Fortran, and other languages via tools such as
  - Simplified Wrapper Interface Generator (SWIG; [http://swig.org](http://swig.org))
  - pyFort ([http://pyfortran.sourceforge.net](http://pyfortran.sourceforge.net))
- There exist a wide variety of standard libraries, especially for numerical and scientific computation
Python in the Climate Community

Just a few examples...

• Climate Data Analysis Tools (CDAT)
  http://www-pcmdi.llnl.gov/software-portal/cdat

• PyNGL (Python interface to NCL Graphics Library)
  http://www.pyngl.ucar.edu/

• PyCPL - Python implementation of CCSM Coupler
  http://geosci.uchicago.edu/~tobis/pycpl-poster-stuff/notes.html
Climate and Computing
Back to CSIRO Mk3L...

- In terms of computer architecture and parallel implementation CSIRO Mk3L
- Employs pure shared-memory parallelism implemented with OpenMP
- Has undergone cache optimisation
- Remains performance-portable to vector platforms
- In terms of emerging technologies:
  - Ensembles could be run and managed using grid technology (e.g., on the APAC Grid)
  - Multicore processors should be exploitable by the model as-is using its OpenMP parallelism
  - Could eventually be re-factored into components or Python modules
Coupled Climate Models as Multiphysics Models
Multiphysics Models

• An ambitious attempt to model complete systems as opposed to subsystems or individual processes

• Comprised of 2 or more mutually interacting or coupled models

  • Interactions may be either 1-way or 2-way

• Models may provide either driving or boundary condition data (or both)

• The coupled climate model is the classic example
In the Future, Multiphysics Modeling Will Be Routine...

Two professional multiphysics modelers

...As will the use of software components!
The Coupling Problem

**Given:** N mutually interacting models \((C_1, C_2, \ldots, C_N)\)

**Goal:** Build an efficient coupled model from them

Aspects of the problem:

- **Architecture**—which components are coupled, interaction frequency, order of execution, *et cetera*

- **Data processing**—crudely put, the “plumbing and wiring” for the data interactions
  - Data description
  - Data transfer, e.g., common blocks (bad) or function/subroutine interfaces
  - Data transformation, e.g., interpolation, diagnostics,...
Parallel Coupling Problem

**Given:** N mutually interacting models \((C_1, C_2, ..., C_N)\), each of which may employ message-passing parallelism

**Goal:** Build an efficient parallel coupled model from them

Aspects of the problem:

- **Architecture**—which components are coupled, interaction frequency, order of execution, resource allocation, ...

- **Parallel data processing**—crudely put, the “plumbing and wiring” for the data interactions
Architecture

• Adding message-passing parallelism complicates coupled model architectural decision-making

• One key problem is resource allocation in terms of both cpu time and processors

• **Serial Composition** Schedule execution of each the models \((C_1,C_2,...C_N)\) to run *sequentially* on all of the processors as an event loop (e.g., CSIRO Mk3L, PCM)

• **Parallel Composition** Schedule execution of the models to run \((C_1,C_2,...C_N)\) *concurrently*, with each model on its own distinct pool of processors (e.g., CCSM)

• Combine the two above approaches (e.g., FOAM)
Serial v. Parallel Compositions

- Serial Composition
  - Pro: easy to synchronise and diagnose performance
  - Con: disparity in scaling of applications can lead to wasted processor time
- Parallel Composition
  - Pro: can reduce processor time waste dramatically
  - Con: synchronisation (hard) and intermodel data dependencies can create a cascade of stalls (nasty)
Parallel Data Processing

- **Description** of data to be exchanged during coupling
  - Physical fields/variables
  - Mesh or representation associated with the data
  - Domain decomposition
- **Transfer** of data--a.k.a. the MxN problem
- **Transformation** of data
  - Intermesh interpolation/transformation between representations
  - Time transformation
  - Diagnostic/variable transformations
  - Merging of data from multiple sources
A Solution: MCT and the CCSM Coupler

- The Community Climate System Model (CCSM) is an open-source coupled climate model

- The intercomponent coupling in CCSM is implemented in its coupler, which was built on top of a software package called the Model Coupling Toolkit (MCT)

- MCT is generic (i.e., no climate-specific assumptions) and is applicable to other fields (e.g., plasma physics)

CCSM’s Coupler

Hub and Spokes: Models interact via the coupler

Model substitution with relative ease supported

Smart, configurable plugs connecting models to coupler

Bottom Line: CCSM is an application-specific software framework

CSIRO Mk3L Workshop 26 May 2006
Back to CSIRO Mk3L

- By avoiding message-passing parallelism, CSIRO Mk3L only has to solve the coupling problem, not the parallel coupling problem.

- CSIRO Mk3L is a serial composition (the natural choice for SMP).

- If MPI parallelism is introduced into the model, the parallel coupling problem will arise.

- If somebody wants to add message-passing to CSIRO Mk3L, MCT could be leveraged to build the parallel coupling connective tissue.
We are Not Alone...

Fusion  Climate/NWP  Reactive Flow

Space Weather  Fluid-Structure Interactions

...But We are Out in Front!

CSIRO Mk3L Workshop 26 May 2006
Conclusions

• Computing and the Climate/Weather/Ocean (CWO) sciences have had a long relationship

• Climate is a grand-challenge application and is unique in its model-to-generate-data approach

• Parallel computing has created opportunities and challenges for CWO scientists/developers (i.e., multiple levels of parallelism in algorithms)

• Those challenges are being met

• Things are about to get very interesting
Unsolicited Advice

• Don’t hesitate to be bold and adventurous
• Don’t be afraid to read code
• Don’t be afraid to write code
  • But do it well and document it--because the code is the apparatus
• Don’t re-invent the wheel
  • Look for re-use opportunities (i.e., libraries)
• Don’t be afraid to ask for help (e.g., from APAC)