Numerical Simulation of Rayleigh-Bénard Convection

Harvard IACS AC 290R - Group 1
Michael S. Emanuel
Jonathan Guillotte-Blouin
Yue Sun

14-March-2019

1 Problem Statement and Motivation

AC 290R is a course on extreme computing with a focus on the application domain of fluid dynamics. Fluid dynamics is one of the most mature areas of high performance computation. Long before there were social networks, scientists and engineers have used the most powerful computers available to understand weather systems and aerodynamics. For our project in this module, we carried out a large scale numerical simulation of Rayleigh-Bénard Convection using the Drekar code running on Harvard's Odyssey supercomputing cluster. Rayleigh-Bénard Convection (RBC) is a classical problem in fluid dynamics. It is the phyical phenomenon that arises any time a fluid in a gravitational field has a temperature gradient. As anyone who lives in an old building can attest, hot air rises and cold air falls. This buoyancy effect of warm fluids expanding and becoming less dense is quite general. When the temperature driven thermal forces are large enough compared to viscous forces, the fluid will move with characteristic circular flows bringing hot fluid higher and cold fluid lower. If the the thermal forces become large enough, a turbulent flow will emerge.

A number of scientifically important physical phenomena involve RBC. Some of the more prominent ones include:

- Stellar evolution: The gas at the center of a star is hotter than the gas at the outside, and a spherical RBC flow develops
- Plate tectonics: The center of the earth hot, with a layer of magma on top and a cool crust. Over geologic time scales, the earth's mantle behaves like a fluid, and plate tectonics is an instance of RBC.
- Weather systems: Warm air rises in the earth's atmosphere and RBC drives weather processes. A concrete example is a mesoscale convective system.

Our goals in this project are two-fold. We would like to learn some basic precepts of fluid dynamics and gain a working knowledge of RBC. We acknowledge however that fluid dynamics is far too large and complicated a field for us to gain a deep understanding in such a short time. Our primary goal, as suggested by the course title **Extreme Computing** is to learn state of the art techniques in scientific computing at a large scale. We choose to work on a real problem, RBC, rather than a toy problem because it is both more interesting and we will learn more.

The particular problem we simulated numerically is a two dimensional turbulent Rayleigh-Bénard Convection. This is described mathematically with a set of three equations called the Boussinesq equations. The Boussinesq approximation is based on a linearization of the buoyancy effect. The buoyancy effect is

approximated as

$$\rho(T_0 + \delta T) \approx \rho(T_0) - \frac{\partial \rho}{\partial T}\Big|_{T=T_0}$$

We define the coefficient of volume expansion as [1]

$$\alpha_V = -\frac{1}{\rho_0}\frac{\partial \rho}{\partial T}\Big|_{T=T_0}$$

The coefficient of volume expansion is a property of a fluid. The second simplifying assumption made in the Boussinesq equations is that fluids are incompressible (except for this linear sensitivity of their density to temperature). The Navier-Stokes equations then simplify to the three Boussinesq equations:

$$\frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} \otimes \mathbf{u}) = \frac{1}{\rho_0}\boldsymbol{\nabla} P + \nu \boldsymbol{\nabla}^2 \mathbf{u} + \alpha_V g T \hat{y} \tag{1}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \tag{2}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{\nabla} \cdot (uT) = \kappa \boldsymbol{\nabla}^2 T \tag{3}$$

In these equations, ρ_0 is is the density at the reference temperature T_0. ν is the fluid viscosity and κ is the thermal diffusivity.

The presentation above includes the physical constants and is suitable for engineering. For mathematics, it is often preferable to use nondimensional equations. [2] There are three choices that have been used for nondimensionalizing the Boussinesq equations. We follow the classical approach and choose the following dimensionless parameters:

- $\Delta T = T_{bot} - T_{top}$ the temperature difference; this is the temperature scale
- $H = y_{top} - y_{bot}$ the height of the channel; this is the length scale
- $\tau = {}^{H^2}/_{\kappa}$ is the time scale; this choice is called "thermal scaling"
- $U = {}^{H}/_{\tau}$ is the velocity scale
- $\rho_0 U^2$ is the pressure scale

With this choice of dimensionless parameters, we have the non-dimensional Boussinesq equations:

$$\frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} \otimes \mathbf{u}) = \boldsymbol{\nabla} P + \mathrm{Pr} \boldsymbol{\nabla}^2 \mathbf{u} + \mathrm{RaPr} T \hat{y} \tag{4}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \tag{5}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{\nabla} \cdot (uT) = \boldsymbol{\nabla}^2 T \tag{6}$$

There are two dimensionless ratios appearing:

- Pr is the Prandtl number, $\mathrm{Pr} = {}^{\nu}/_{\kappa}$. This is the ratio of the thermal viscosity to the thermal diffusivity and is a property of the fluid.
- Ra is the famous Rayleigh number. It is defined by

$$\mathrm{Ra} = \frac{\alpha_V \Delta T g H^3}{\nu \kappa}$$

[1] See here a discussion of thermal expansion generally
[2] See this article on the technique of nomdimensionalization.

The Rayleigh number describes the ratio of temperature driven forces to viscous forces. For this reason, a very low Rayleigh number leads to no convective flow; an intermediate Rayleigh number leads to a somewhat regular flow; and a high Rayleigh number leads to a turbulent flow.

2 Description of Code

Drekar is a large scale computational fluid dynamics (CFD) code. The name Drekar comes from the norse word describing a Viking Longship. It is under active development at Sandia National Laboratories and was recently made partially open source; details can be found at the US Department of Energy Office of Scientific and Technical Information at OSTI.gov. Because some parts of this code are sensitive, e.g. they could be used for simulations of nuclear weapons, these parts are not being released to the public. Sensitive components will remain restricted to employees and affiliates of Sandia.

Drekar solves the partial differential equations (PDEs) of fluids using the Finite Element Method. In addition to simulating traditional fluid dynamics problems, Drekar also has the capability to handle electromagnetic forces. This is important when modeling plasma physics and stars for example. Drekar is able to accurately simulate different kinds of physical systems including incompressible fluids, low-mach compressible fluids, and compressible / incompressible magnetohydrodynamics. It can also handle multi-species plasmas interacting with a magnetic field; this is used to simulate Tokamak reactors. Researchers at Sandia are collaborating with the team building the ITER in Southern france, which will be the world's largest megnetic confinement plasma physics experiment when it is constructed.

The heart of Drekar is a massively parallel implementation of the Finite Element Method. The Finite Element Method is a numerical method for solving partial differential equations. We will discuss it further in the next section. Drekar is a modern software project written in C++ and making extensive use of classes and templates. This allows it to be flexible and to run efficiently on heterogeneous hardware.

Drekar implements parallelization at multiple levels. At the top level, it uses Message Passing Interface (MPI) to distribute computations across multiple computing nodes that are connected on the same high performance network. Back end modules of Drekar also permit parallelization at the level of C++ threads; OpenMP (multiple cores on one host); and GPUs with CUDA.

Drekar has basic components that handle primitive linear algebra tasks such as distributing a matrix or a vector across multiple compute nodes. Building up in complexity, it knows how to add vectors, perform vector dot products and matrix / vector products. It can solve linear equations exactly or approximately using Krylov methods, and similarly for eigenvalue problems. And it can solve nonlinear equations using iterative methods (typically refinements of Newton's method).

These building blocks enable Drekar to simulate a wide variety of PDEs using the finite element method, and to perform these scales efficiently at scale. Drekar handles continuous, discontinuous and stabilized finite elements. The flexibility and wide range of the C++ classes in Drekar allow it to simulate mixed-integration finite element bases include nodal, edge, face and volume elements. This allows a wide range of problems to be described. The breadth of elements are particularly useful when enforcing complex boundary conditions. Drekar also has the flexibility of simulating multiple physics regimes, called blocks, in one system.

Drekar includes algorithms for time integrating differential equations that are fully explicit, fully implicit, and mixed implicit / explicit (IMEX). Some degree of implicit solution is often required for real world problems when there is a wide range of time scales. If the fastest time scale is too many orders of magnitude shorter than the longest one, an explicit solver will either take time steps too large to capture the physics, or take too long to run. Implicit and IMEX solvers permit the simulation to implicitly solve for physical effects that are on too short a time scale to solve explicitly.

Drekar has strong tools for automatic differentiation. These are necessary because many methods used in Drekar require taking derivatives of functions, e.g. solving nonlinear equations with a variant of Newton's method. Some other software libraries required users to supply hand coded derivatives, which can be time consuming and error prone; or to use numerical derivatives, which can be slow. Automatic differentiation allows the best of both worlds.

Drekar is built as a part of a collection of packages called Trilinos. Trilinos arose out of an effort at Sandia to avoid duplication of effort across different research groups and allow each group to focus on the frontier of discovery in their area without reinventing the wheel. Some packages in Trilinos relevant to Drekar include:

- Linear solvers: *Amesos* (direct) and *Aztec00*, *Belos* (indirect)
- Preconditioners: *lfpack* (algebraic), *ML* (multilevel), *Meros* (block)
- Eigenvalue solvers: *Anasazi*
- Nonlinear solvers: *NOX*
- Time Integrators: *Rhythmos, Tempus*
- Automatic Differentiation: *Sacado*

Consider for a moment how many of the basic building blocks used by Drekar are provided by the packages listed above. By building on this existing work, the developers of Drekar were able to concentrate their efforts on the distinctive aspects of the problem, namely solving PDEs with the Finite Element Method. Having seen a number of software packages over the year, our first impression of Drekar is that it is a very sophisticated and impressive piece of software engineering.

3 Overview of Numerical Methods Used

4 Parameters of the Simulation

5 Results

6 Conclusions and Future Work

Your text goes here.

6.1 A subsection

More text.