Redesigning numerical modelling algorithms for efficient large-scale cloud deployment

James Hobro (Schlumberger)
Anindya Sharma (Schlumberger, now at University of Cambridge)

Introduction

The ready availability of cloud computing resources presents an opportunity for rapid turnaround in large numerical modelling tasks. This opens up new possibilities for interactive simulation. However, it is only feasible for numerical modelling algorithms that scale well over large computing clusters with high network latency. This is relatively straightforward for algorithms that are embarrassingly parallel, but achieving linear scaling for coupled numerical modelling problems is much more difficult. We explore the improvements that can be achieved in scalability for this type of algorithm by moving away from a sequential synchronous multiprocessor approach as conventionally used with the Message Passing Interface (MPI), which encourages large-scale synchronisation across a parallel system. Instead, we explore an approach based on the actor programming model that removes all unnecessary synchronisation during inter-node communication. It avoids synchronisation by introducing flexibility in the order of computation which allows data exchange to be deferred and handled asynchronously.

Limitations of synchronous multiprocessoring

Multi-processing approaches based on synchronisation primitives have facilitated the parallelisation of many high-performance computing (HPC) algorithms, notably via the Message Passing Interface (MPI) and Open-MP. However, many of these algorithms suffer from scaling problems which arise from the requirement for frequent synchronisation. This synchronisation may be explicitly coded or implicitly caused by chains of wait relationships (e.g. between neighbouring domains in the domain decomposition of a time-stepping scheme). In extreme cases, this can dominate the overall runtime of a program.

As HPC algorithms approach Exascale it is becoming more important to reduce or eliminate unnecessary synchronisation. Within a synchronised, programming model, the data exchange normally achieved at an MP or task base by explicitly overlapping tasks where wait time is detected. Current HPC approaches also rely heavily on specialised low-latency network infrastructure, restricting the range of scale over which these algorithms can be deployed.

Fully-asynchronous design approaches

Asynchronous programming: Using an actor based model

Aims of asynchronous task-based design

• N-Core: Scalable throughput.
• Workload decomposed into a “tree” of heterogeneous chores.
• Lots of latent parallelism we can map down to N cores.

Actor-based programming

Actors: Independently running tasks

Actor communications: Asynchronous

Performance and scaling

The performance of the OpenMP/MP implementation is not as scalable as the Actor implementation despite the latter’s ability to run inefficiently over large virtual machines.

Conclusions

With task based and actor based techniques it was possible to redesign an existing finite-difference elastic wave equation solver C++ code base to remove all explicit synchronisation.

The asynchronous design approach brings significant scaling and performance benefits due to efficient overlap of computation and data movement.

The increased tolerance to network latency allows this type of algorithm to run and scale efficiently on commodity cloud computing resources.

The core scientific code (solver kernels, operators, interpolation kernels etc.) was preserved with either zero or minimal modification.

The fine-grained task-based approach opens up new possibilities for flexible deployment and efficient integration within larger workflows.

References and further reading


