Parallel Profiling of the FLITE CFD Software on HPC Wales

Introduction

FLITE is a Computational Fluid Dynamics (CFD) code developed by Swansea University. It is being used in an HPC Wales project concerned with the development of the supersonic car "Bloodhound SSC", where FLITE is used to model the flow of air around the car body.

The Bloodhound SSC is a unique, high-technology project to design and build a car that will break the 1,000 mph barrier and set a new world land speed record[1]. At such high speeds, accurate understanding of the behaviour of the airflow around the car will be vital to ensure that it remains stable and on the ground. CFD is an important part of this process since it allows engineers a fast, inexpensive method of testing design changes rather than manufacturing components or using real wind tunnels.

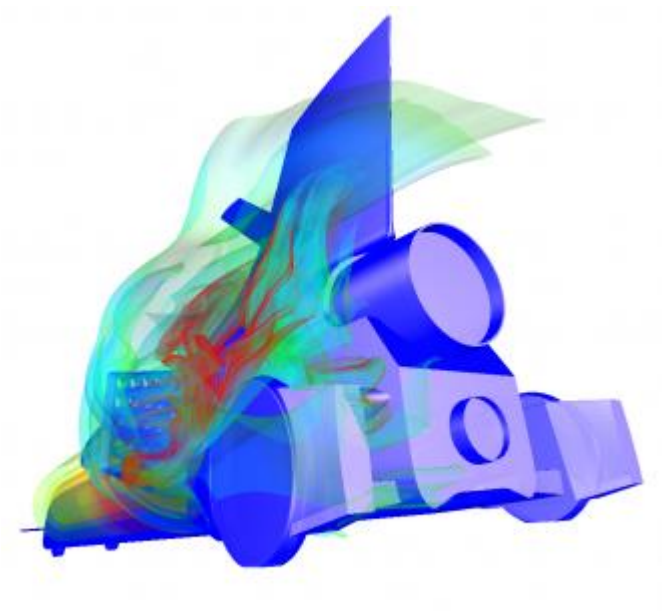

Figure 1 : CFD model of the Bloodhound SSC

The resources available in HPC Wales have given the code designers of FLITE access to large parallel computational resources, which are vital for the accurate computation of cases such as Bloodhound SSC. This is because it is important to have highly detailed computational meshes describing the car's geometry. These place significant demands on the computational storage needed for a simulation and on the ability to perform calculations in a reasonable length of time.

HPC Wales were keen to ensure that the simulations undertaken were able to leverage the full power of the computational resources available. This was hoped to increase the overall efficiency of software running on their system in order to maximise good parallel programming practice.

[1] http://www.bloodhoundssc.com/

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Parallel scalability

The FLITE software solves Navier-Stokes equations on a complex mesh made up of hexahedra, prisms and tetrahedra numbering into the tens of millions.[2] These are then solved using the finite element method. MPI is used to run in parallel on distributed memory resources. This allows enormous flexibility in the communication patterns between neighbouring processes in the 3D mesh representing the total domain.

In this work, focus was placed on enabling the code to be developed in a manner that would help it to be as portable as possible. Much time was dedicated to ensuring that the start-up procedures were able to take full advantage of the functionality provided on the HPC Wales machines, and that memory issues that would manifest on larger processor-count runs were written out of the code.

The parallel scalability of the code was assessed using a number of criteria and packages. One of these is shown on the right. This is a comparison of the performance of a single test case on 16 to 64 cores when run on the two different Intel chip architectures available in the HPC Wales system. It can be seen that the older Westmere processors perform noticeably slower than the Sandy Bridge nodes. In addition it is seen on this test example that there is still a performance increase up to 64 cores, although the parallel efficiency is lower than at 32 cores.

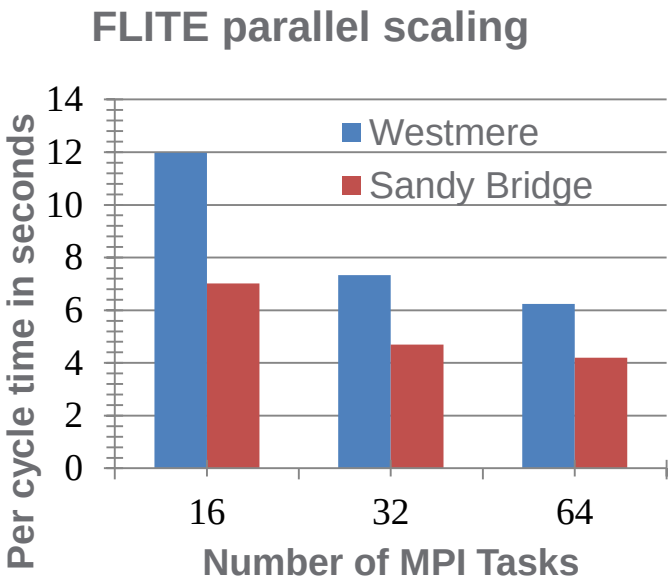

Figure 2 : Parallel scaling on HPC Wales Westmere and Sandy Bridge architectures

Additional profiling work was then performed using Tau[3] and Jumpshot[4] in order to look at the parallel scalability to better understand where the loss in performance was being incurred. Several functions which were frequently called and accounted for significant proportions of time were highlighted in order to help the developers to perform more targeted profiling and optimisation work.

[2] http://www.bloodhoundssc.com/project/car/aerodynamics/computational-fluid-dynamics
[3] http://www.cs.uoregon.edu/research/tau/home.php
[4] http://www.mcs.anl.gov/research/projects/perfvis/download/