Fourth Annual UK Systems Research Challenges Workshop

Authors
Alan Dearle – University of St Andrews (al@st-andrews.ac.uk)
Graham Kirby – University of St Andrews (graham.kirby@st-andrews.ac.uk)
Richard Connor – Stirling University (richard.connor@stir.ac.uk)

Title Observations of a shared memory Java program on NUMA

Abstract
Non Uniform Memory Access (NUMA) is most prevalent architectural pattern in usage today. This paper reports on some experiments with a computational and data intensive Java program running on NUMA. The algorithm in question is a Metric space search algorithm called BitBlaster. It creates a large in memory binary matrix which is then computed over to find approximate solutions to a metric space query which are then filtered to produce exact solutions.
When the algorithm was parallelised and ported for execution on a NUMA architecture (an Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz with 2 running SElinux kernel-3.10.0-514.16.1.el7.x86_64 with 2 CPUs each with 6 cores with hyperthreading and 132 GB of memory in 2 banks) it ran much slower than expected.
This paper reports on a series of experiments changing the number of CPUs available, the degree of parallelism in the algorithm, the way in which data was structured, the data structures were initialised and how threads were created with the aim of achieving better performance on the NUMA platform. The final version of the code involved the creation of a *actors* style dispatch mechanism to ensure threads and memory associated appropriately with respect to processors.
The paper reports on the experiments that were conducted and how the implementation of the algorithm evolved over time. The results highlight the effect of the various options and how similar programs written for NUMA architectures may benefit from similar restructuring.
The results of these experiments and even their necessity surprised the authors and in this paper we seek to share them with the community to gain better understanding for ourselves and for others. We believe that the results of these experiments leave an outstanding question – do we need new libraries, languages or middleware to make it easier for programmers to write such programs?