Preliminary Evaluation of Chapel capabilities with NAS Parallel Benchmarks

Marcelo Cintra, Diego R. Llanos

September 2, 2010

Abstract

Chapel is a new high-level programming language aimed at the programmability aspect required for HPC. This document shows a preliminary comparison between the OMP implementation of several NAS Benchmarks and their corresponding versions written in Chapel.

1 Introduction

In 2002, DARPA launched their High Productivity Computing Systems (HPCS) programme, offering funding for industry and academia to research the development of computing systems which focus on high productivity along with high performance. Part of this programme concentrates on the specification of novel languages for the HPC community. In Phase 2 of the programme (July 2003 - July 2006) the three remaining partners that were awarded funding were Cray, with its Chapel language; IBM, with its X10 language; and SUN, with its Fortress language. SUN were eventually dropped from the programme at the start of Phase 3. An overview of these languages can be found in [1].

2 Scope and planning

The motivation for this work is to evaluate the performance obtained by several NAS Parallel Benchmarks [2] written in Chapel [3]. Chapel allows to easily program for both shared- and distributed-memory environments, although the code needs some adjustments to run in different architectures. Therefore, it is possible to compare Chapel single-locale code with the shared-memory reference implementation (written using OpenMP directives) and Chapel multiple-locale code with the distributed-memory implementation (written with MPI). For each benchmark considered, we have measured both sequential and shared-memory performance.
3 Benchmarks used

3.1 NPB EP

EP is the “embarrassingly parallel” benchmark. It generates pairs of Gaussian random deviates according to a specific scheme and tabulate the number of pairs in successive square annuli. It provides an estimate of the upper achievable limits for floating point performance, i.e., the performance without significant interprocessor communication.

The shared version of the benchmark is originally written in Fortran with OpenMP parallel directives. We have used Class B problem size in our experiments.

3.2 NPB FT

FT is a 3D fast-Fourier transform partial differential equation benchmark. It solves numerically a certain partial differential equation using forward and inverse FFT. This kernel performs the essence of many “spectral” codes. It is a rigorous test of long-distance communication performance.

The shared version of the benchmark is originally written in Fortran with OpenMP parallel directives. We have used Class B problem size in our experiments.

3.3 NPB IS

IS sorts $N$ small, integer keys in parallel. The keys are generated by a sequential key generation and initially must be uniformly distributed in memory. This operation is important in “particle method” codes. It tests both integer computation speed and communication performance.

The shared version of the benchmark is originally written in C with OpenMP parallel directives. We have used Class B problem size in our experiments.

4 Experimental setup

Experiments were carried on Ness, a system with 16 AMD Opteron cores running at 2.6 GHz, each one with 2 GB of memory. The system runs Scientific Linux. All threads had exclusive access to the processors during the execution of the experiments, and we use the parallel time returned by NAS Benchmarks in our measurements.

We have used GCC 4.1.2 with -O3 optimization level to build the C OpenMP versions, G95 4.1.2 with -O3 optimization level to build the Fortran OpenMP versions, and Chapel 1.1 with GCC 4.1.2 as underlying C compiler to build the Chapel versions.
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Original version</th>
<th>Naive Time</th>
<th>Speedup</th>
<th>Naive with --fast Time</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP</td>
<td>112.37</td>
<td>1052.8</td>
<td>0.10</td>
<td>231.25</td>
<td>0.48×</td>
</tr>
<tr>
<td>FT</td>
<td>168.76</td>
<td>+3600</td>
<td>n/a</td>
<td>2379.44</td>
<td>0.07×</td>
</tr>
<tr>
<td>IS</td>
<td>8.66</td>
<td>487.282</td>
<td>0.02</td>
<td>269.21</td>
<td>0.03×</td>
</tr>
</tbody>
</table>

Table 1: Sequential execution time (in secs), naive versions.

5 Sequential performance

5.1 Execution time of naive versions

Table 1 shows the sequential execution time of the naive versions. According to Chapel documentation, the effect of the --fast flag is to disable --checks, that perform different integrity checks at runtime (bound checking, locality of references within local blocks and nil object references), disable --ieee-float, thus allowing some optimizations that may affect IEEE 754 conformance, and enable different Chapel compiler optimizations.

5.2 Execution time of cache-aware versions

Since EP and FT benchmarks are originally written in Fortran, they use column-based storage of arrays. Chapel generates C code, thus is expected (although not confirmed yet) to use row-based storage. To show the effect of changing the Chapel code to take advantage of access locality is still a work-in-progress.

6 Shared-memory performance

6.1 Execution time of shared memory versions

Figure 1 shows the execution times of the shared memory versions developed so far. Both Chapel and the original versions present a good scalability, so we expect that any sequential improvements we may suggest to the Chapel compiler and runtime system will benefit parallel processing as well.

7 Conclusions

This work evaluates Chapel capabilities in the execution of three well-known NAS parallel benchmarks. From the programmability point of view, Chapel is an excellent language for HPC, although the preliminary results show that there is considerable room for improvement in the compiler and runtime system provided by Chapel. Our plan is to further explore this point, both for sequential and parallel performance.
Figure 1: Performance results for shared-memory versions.
Acknowledgments

This work was carried out under the HPC-EUROPA2 project (project number: 228398), with the support of the European Community - Research Infrastructure Action of the FP7.

References

