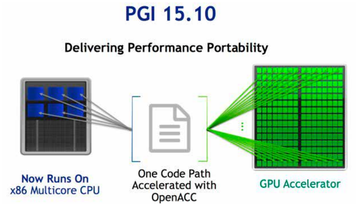
OpenACC Enabled Benchmark Suite on Intel Ivy Bridge

Joel Bricker and Sunita Chandrasekaran
Department of Computer and Information Sciences, University of Delaware, USA
{jbricker,schandra}@udel.edu

Introduction

- As time passes, computing architectures become increasingly sophisticated.
- Architectures are expected to further advance
- Reaching Exascale level of computing will require both advanced technology and sophisticated yet flexible programming models.
- A lot of uncertainty about the programming models for Exascale still prevails
- However the requirements are quite clear.
  - Need for performance portability
  - Need for maintaining a single code base
  - Preserve legacy code
- Tackling conflicting yet critical goals – a real challenge
  - Software abstraction yet machine-specific optimization Programming models are expected to provide multiple device support
- OpenACC[1], a high-level directive-based programming model, has been gaining a lot of traction for the past couple of years for its promising performance on accelerators
- Adoption has grown to over 10,000 + OpenACC developers
- What has made the model even more unique is its recent support for multicore CPUs



Performance Portability on GPUs and CPUs with OpenACC[4]

Motivation & Research Question

- Is it possible to achieve similar performance to OpenMP[2], a widely popular parallel programming model, on x86 multicore with OpenACC?
- Can a single code base be maintained across X86 and accelerators?
- Would the solution be performance portable?

Benchmark, Compiler and Evaluation Platform

- Benchmark Suite: Scalable Heterogeneous Computing Benchmark Suite (SHOC)[3]
- Compiler: PGI's OpenMP and OpenACC compilers, Version 15.10 and GNU 4.9.3
- Evaluation Platform: University of Delaware's cluster consisting of 100 compute nodes which total 2000 Intel "Ivy Bridge" cores, 6.4TB RAM, 256TB Lustre filesystem, and an FDR InfiniBand network backplane.

OpenMP Vs OpenACC (SGEMM)

```
void sgemm(int m, int n, int k, const float *A, const float *B, float *C)
{
   int i, j, l;

#pragma omp parallel shared(n,m,k,A,B,C) firstprivate(i,j,l)
    {
#pragma omp for
       for (i = 0; i<m; i++)
       {
#pragma omp parallel for
          for (j=0; j<n; j++)
          {
             double sum=0.0;
#pragma omp parallel for
             for (l=0; l<k; l++)
             {
                sum+=A[i*n+l]*B[k*n+j];
             }
             C[i*n+j]=sum;
          }
       }
    }
}
```

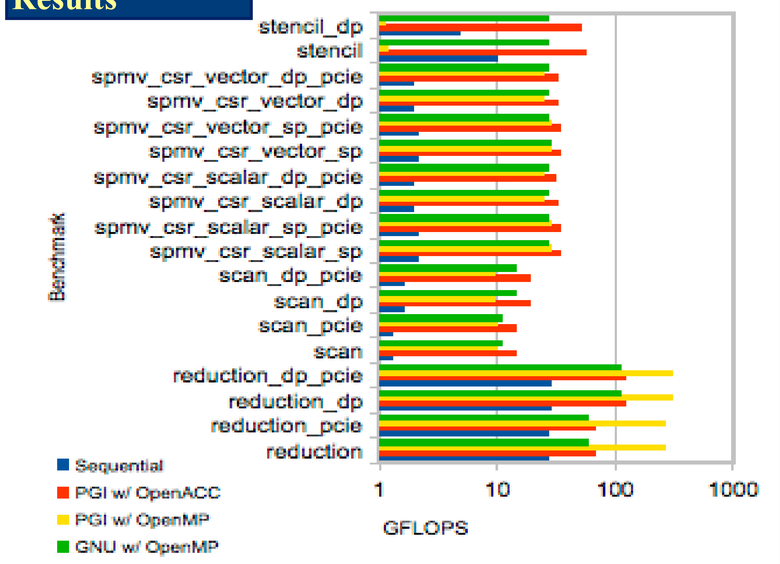
```
void sgemm(int m, int n, int k, const float *A, const float *B, float *C)
{
   int i, j, l;

#pragma acc data copyout(C[0:(n*n)]), copyin(A[0:(n*n)], B[0:(n*n)])
{
#pragma acc kernels
#pragma acc loop gang
   for (i = 0; i<m; i++)
   {
#pragma acc loop vector(8)
      for (j=0; j<n; j++)
      {
         double sum=0.0;
#pragma acc loop seq
         for (l=0; l<k; l++)
         {
            sum+=A[i*n+l]*B[k*n+j];
         }
         C[i*n+j]=sum;
      }
   }
}
}
```

Discussion

- Implementation of OpenMP and OpenACC required the same effort for either standard
- Preliminary results are promising and demonstrate the following:
  - Write once and use in multiple places!
  - PGI's OpenACC compiler performed the best for nearly every benchmark.
  - PGI's OpenMP compiler performed significantly better for reduction benchmarks
  - PGI's OpenACC compiler showcases performance portability for multicore CPUs
- Following table shows the speedup of OpenACC over sequential and OpenMP versions:

Speedup Over	Reduction	Scan	SPMV	Stencil
Sequential	3.48	11.48	16.25	8.21
OpenMP	0.33	1.67	1.25	47.77

Results



Directions for Future Research

- Further investigation required for two results that seem as outliers: PGI with OpenMP for the stencil benchmarks and PGI with OpenMP for the reduction benchmarks.
- We will further explore the results on the rest of the benchmarks in the SHOC suite.
- The most primitive pragmas were added to get the benchmarks to run in parallel. We will explore the more elegant features of both OpenMP and OpenACC to see how tuning will impact the benchmark performance.
- Our end goal is to further explore scalable performance across multiple CPUs and GPUs yet maintaining a single code base using real-world applications

References

[1] OpenACC Specification. http://www.openacc.org/sites/default/files/OpenACC_2pt5.pdf
[2] OpenMP Specification. http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf
[3] Jeff Vetter. "SHOC Benchmark Suite". https://github.com/vetter/shoc/wiki
[4] Mark Harris. "Performance Portability from GPUs to CPUs with OpenACC". http://devblogs.nvidia.com/parallelforall/performance-portability-gpus-cpus-openacc/

ACKNOWLEDGMENT: We would like to thank OpenACC for their continued support. Special thanks to Mat Colgrove (PGI/OpenACC). This research was supported in part through the use of Information Technologies (IT) resources at the University of Delaware, specifically the high-performance computing resources.