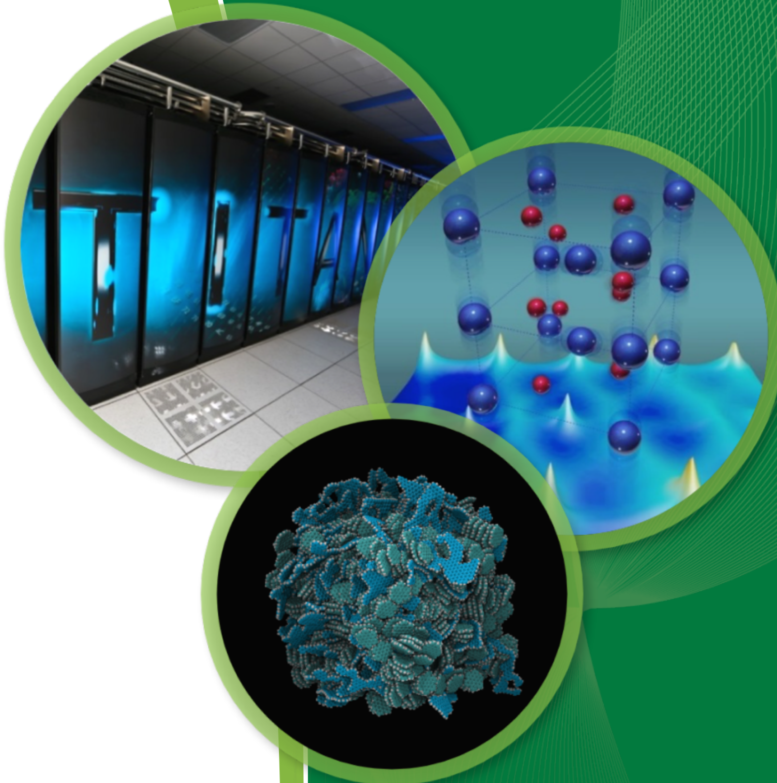
An in-depth evaluation of GCC's OpenACC implementation on Cray systems



Verónica G. Vergara Larrea, ORNL
Wael R. Elwasif, ORNL
Oscar Hernandez, ORNL
Cesar Philippidis, Mentor Graphics
Randy Allen, Mentor Graphics

ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Overview

- OpenACC implementations
- GCC's OpenACC implementation
- Known Limitations
- An Example
- Evaluating GCC's OpenACC
- Conclusions
- Future Work

OpenACC implementations

- Relatively new directive-based specification
  - Current release is v2.5
- Several implementations already support OpenACC:
  - PGI, Cray Compiler Environment, and Pathscale
- Support different targets:
  - PGI can offload to both GPUs and multicore targets
  - CCE can offload to GPUs (craype-accel-nvidia*), host (craype-accel-host)
  - Pathscale can offload to GPUs and host
- Recently, GCC started an effort to add support for OpenACC
- Partial support for OpenACC is already available in GCC 6.3
- This work explores the functionality and performance of GCC's OpenACC implementation

GCC's OpenACC implementation

- Mentor Graphics is developing and maintaining the OpenACC implementation in GCC's gomp-4_0-branch development branch
- GCC is widely used, open source, that supports a subset of CilkPlus, OpenACC 2.0a, and OpenMP 4.5 programming models
- GCC's support for for OpenACC was built on top of its existing support for OpenMP
  - Extensive modifications were required to implement OpenACC efficiently on GPUs
  - GCC does not currently offload OpenMP to GPUs, only to Intel MIC targets

GCC's OpenACC Known Limitations

- Only supports NVIDIA GPUs
  - Single CPU thread is used if executed on multicore hosts
- No support for nested parallelism, `device_type`, and `bind` clauses
- Dynamic arrays in OpenACC data constructs limitations:
  - Pointer-to-arrays not supported
  - Target host not supported
- Loop private variables stored in local memory, rather than shared
- `private` and `firstprivate` clauses do not support subarrays
- Unable to detect parallelism inside `acc kernels` regions
  - Fallbacks to single thread execution

Evolution of GCC's OpenACC implementation

GCC 5
- Highly experimental
- Vector parallelism

GCC 6
- Gang, worker, vector parallelism
- Preliminary support for OpenACC routines

GCC 6.3 (upstream)

GCC 6.3 (gomp4)
- Additional OpenACC functionality
- Enhanced performance for NVIDIA GPUs

GCC 7
- Focuses on performance
- Refines support for OpenACC 2.0a routines
- Adds support for the declare directive

GCC 8
- Includes full support for OpenACC 2.5

An Example: Matrix Multiplication

Porting Matrix Multiplication: Parallel

```
#pragma acc parallel
for (i = 0; i < n; i++)
{
  for (j = 0; j < n; j++)
  {
    int t = 0;

    for (k = 0; k < n; k++)
      t += at(i, k, a) * at(k, j, b);

    at(i, j, c) = t;
  }
}
```

Porting Matrix Multiplication: Parallel Loop

```
#pragma acc parallel
#pragma acc loop
for (i = 0; i < n; i++)
{
  for (j = 0; j < n; j++)
  {
    int t = 0;

    for (k = 0; k < n; k++)
      t += at(i, k, a) * at(k, j, b);

    at(i, j, c) = t;
  }
}
```

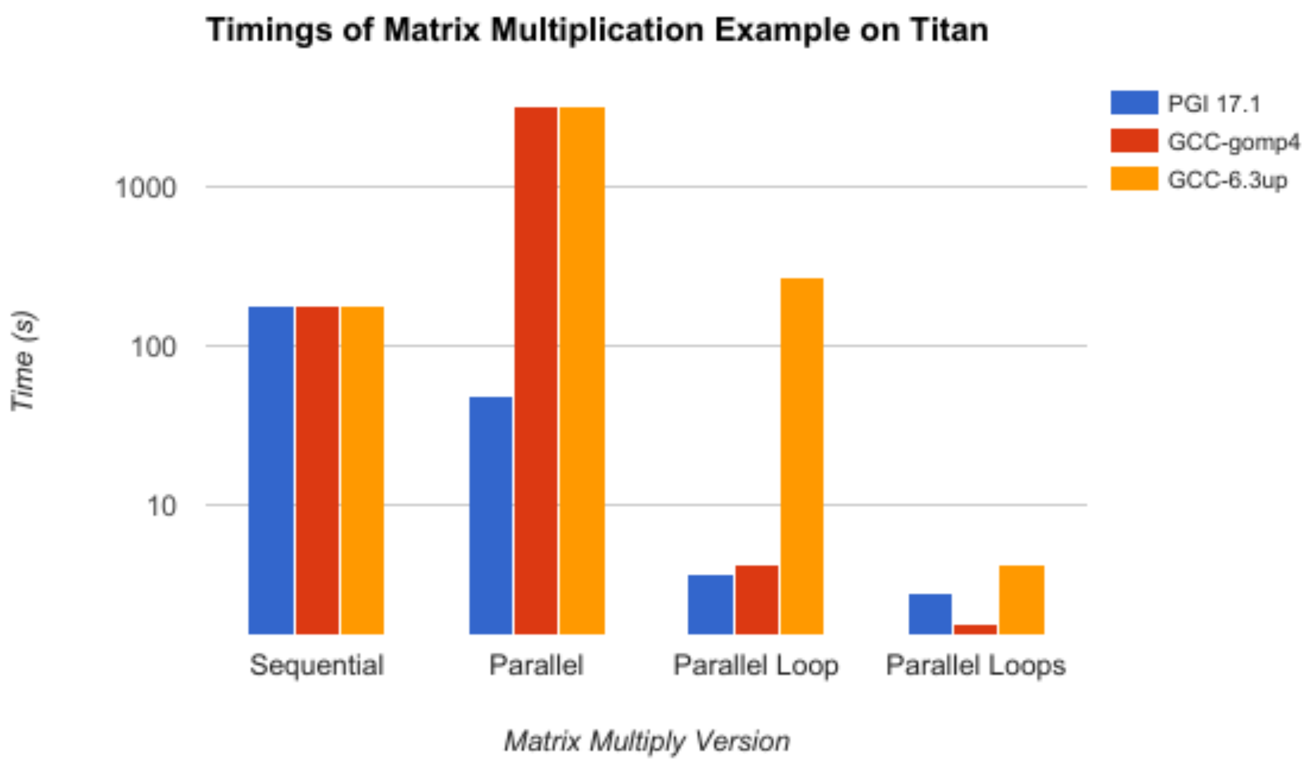
Porting Matrix Multiplication: Parallel Loops + Reductions

```
#pragma acc parallel present (a[0:n*n], \
b[0:n*n], c[0:n*n])
#pragma acc loop
for (i = 0; i < n; i++)
{
  #pragma acc loop
  for (j = 0; j < n; j++)
  {
    int t = 0;

    #pragma acc loop reduction (+:t)
    for (k = 0; k < n; k++)
      t += at(i, k, a) * at(k, j, b);

    at(i, j, c) = t;
  }
}
```

Porting Matrix Multiplication



Evaluating GCC's OpenACC

Evaluating Compliance: OpenACC V&V

- Used the OpenACC Verification and Validation suite from University of Houston
- Validates implementations to the OpenACC v1.0 specification using microtests
  - New version targeting OpenACC v2.5 is expected to be available later this year

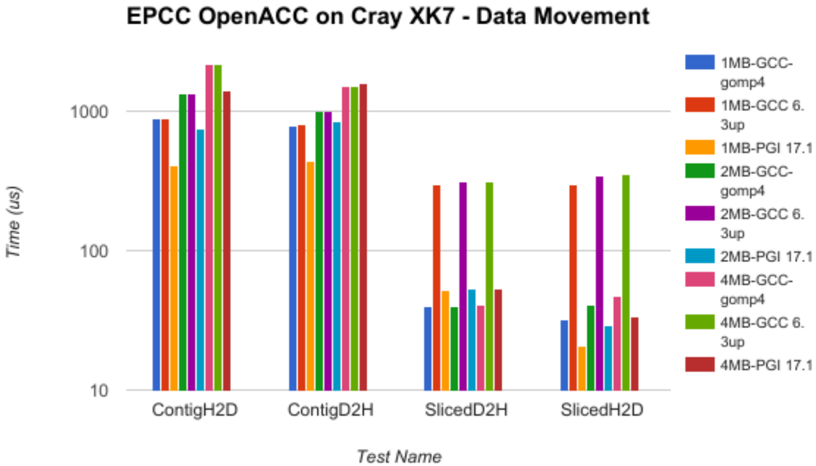
Compiler	Passed	Failed	CE	RE	Total
GCC-gomp4	163	17	56	57	293
PGI 17.1	204	19	20	51	294
CCE 8.5.5	158	27	38	72	295

Measuring OpenACC overheads: EPCC OpenACC benchmark suite

- The EPCC OpenACC benchmark suite was introduced in 2013
  - The suite has not been updated.
- Designed to measure and compare the performance of OpenACC implementations on different architectures
- Contains three levels of tests:
  - Level 0: overheads of certain OpenACC constructs
  - Level 1: performance of computationally intensive linear algebra kernels
  - Level 2: kernels from real-world applications
- A few tests produce compilers and runtime errors
  - Even with mature compilers like PGI

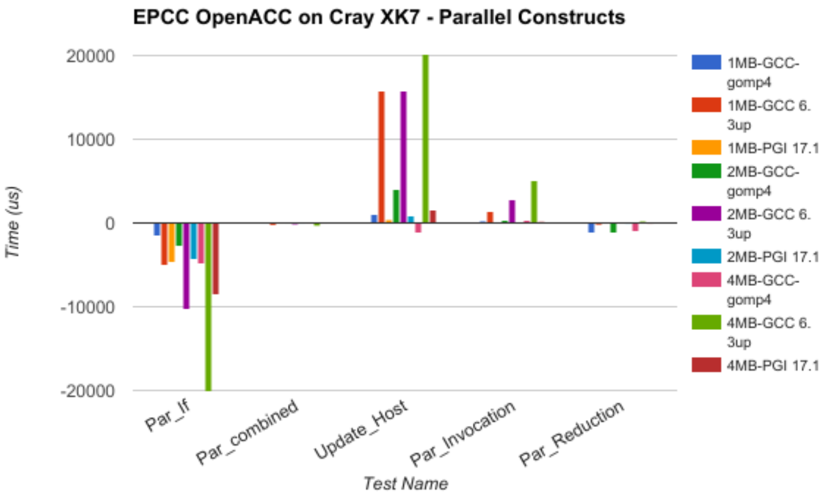
Measuring OpenACC overheads: EPCC OpenACC benchmark suite

Data movement



GCC-gomp4 fastest GPU -> CPU
PGI fastest CPU -> GPU

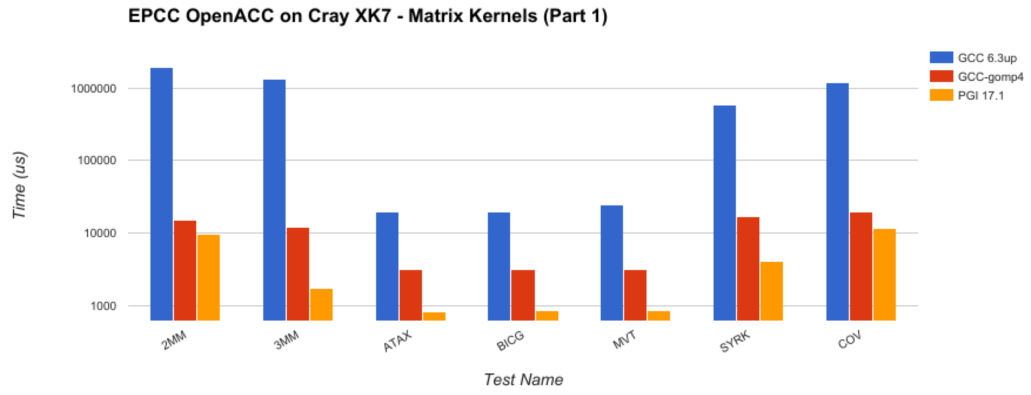
Parallel constructs

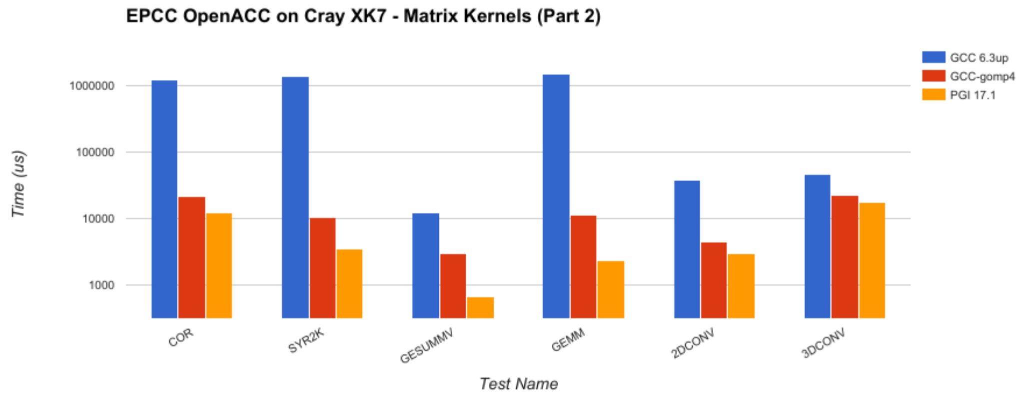


Parallel Reduction much slower
with GCC
(varies by type of reduction)

Measuring OpenACC overheads: EPCC OpenACC benchmark suite

Linear Algebra Kernels



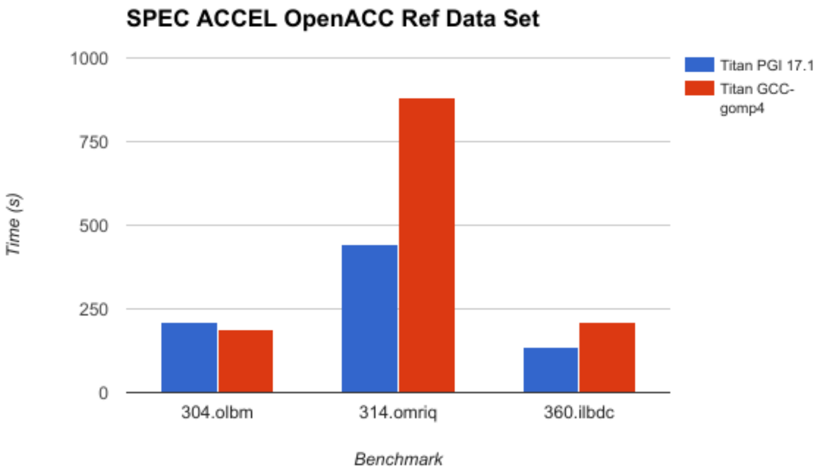


Measuring OpenACC performance: SPEC ACCEL OpenACC

- Developed by SPEC High Performance Group to measure performance for compute intensive parallel applications on accelerators
- Released in September 2015
- Contains two benchmark sets: OpenCL and OpenACC
- OpenACC set contains 15 application kernels: 7 C kernels, 6 Fortran, 2 combined.
- Three data sets: test, train, ref. Only ref is used to compare performance across architectures
- Only three benchmarks that use acc parallel could be used
  - The rest use `acc kernels` and run on a single thread

Measuring OpenACC performance: SPEC ACCEL OpenACC

Measured Estimates



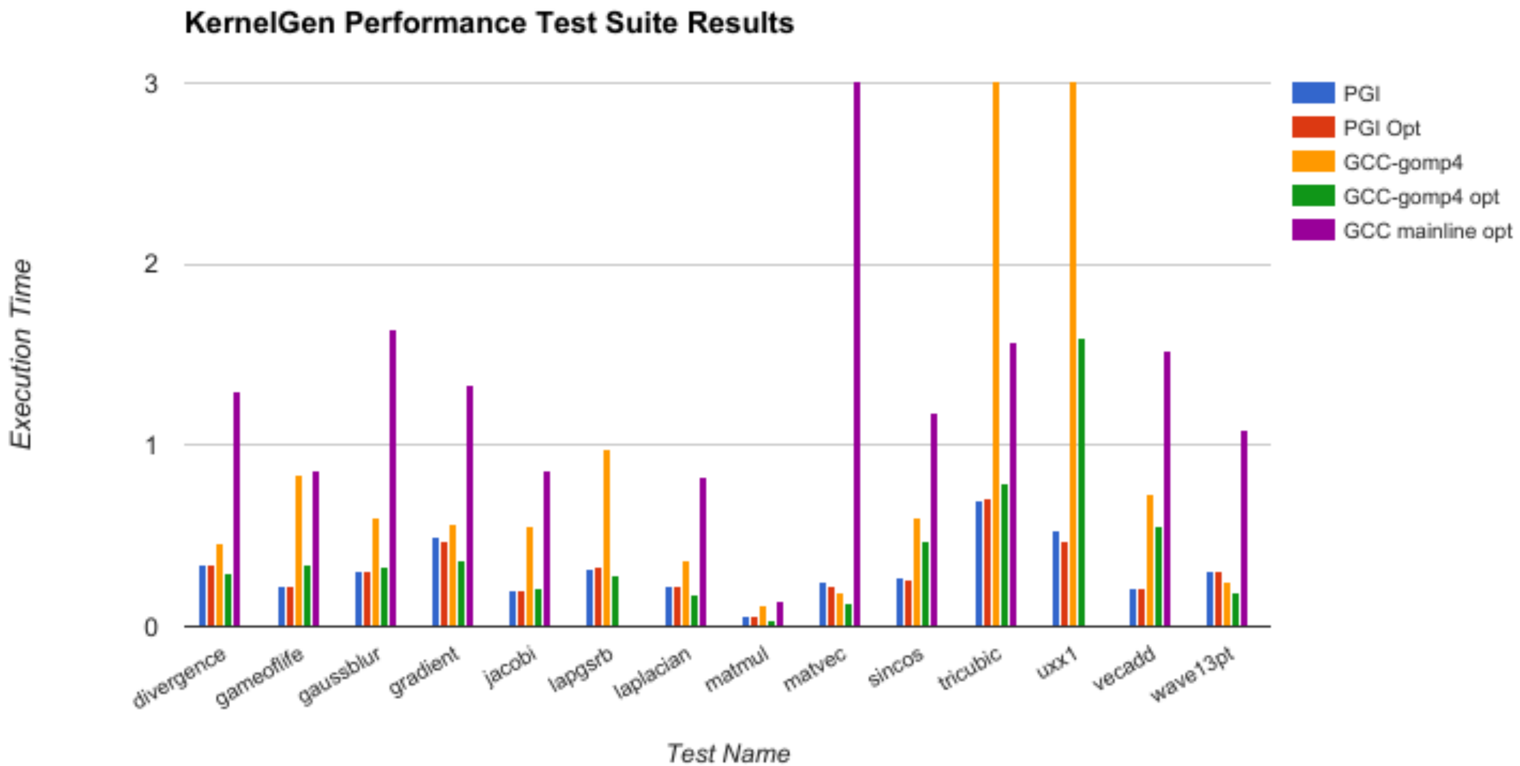
Performance Difference

Benchmark	Perf. Diff
304.olbm	11.48%
314.omriq	-100.00%
360.ilbdc	-51.21%

Measuring OpenACC performance: KernelGen

- Set of OpenACC codes developed as part of the KernelGen project
- Evaluates the ability of compilers to exploit “easy” parallelism
- Consists of single precision numerical algorithms in 2D and 3D grids
- 10 tests use C, 3 use Fortran
  - Tests were modified to update OpenACC syntax to latest specification
  - Also modified tests to use `acc parallel` where `acc kernels` were used
- Tests executed with and without optimization flags

Measuring OpenACC performance: KernelGen



Conclusions

- GCC's OpenACC implementation is now available with partial support for OpenACC v2.0a
  - Mentor Graphics public GCC branch gomp-4_0-branch has the latest updates
- GCC-gomp4 can in some cases outperform more mature implementations.
  - As was the case with the SPEC ACCEL 304.olbm benchmark
  - Overall, GCC is ~47% slower than PGI for SPEC ACCEL measured estimates
- Known limitations of the implementation reduce the number of tests available for the evaluation

Conclusions (cont'd)

- For portability, OpenACC implementations should support many targets
  - e.g., PGI achieves good performance on both GPU-based and manycore-based systems
  - To compare performance, support for additional architectures is needed in GCC's OpenACC implementation
- An open source implementation is useful to expand the adoption of OpenACC
- Many of the benchmarks available have not been recently updated
  - Community involvement could improve and encourage updates to benchmarks

Future Work

- Evaluation should be repeated when GCC 7 is released
  - And again with GCC 8
- Work on validation benchmarks for OpenACC 2.5 is on-going
- A larger study including more implementations should be conducted once GCC's OpenACC implementation is more mature
  - Should include newer hardware as well as additional compilers
- Experiments using a Cray XC40 KNL system were conducted using PGI.
  - Need GCC to also support multicore architectures to fully evaluate and compare implementations

Thank you!

Questions?

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.