Comparative Analysis of OpenACC Compilers

Daniel Barba, Arturo Gonzalez, Diego R. Llanos
Grupo Trasgo
Universidad de Valladolid

TAPEMS, ICA3PP2016
December 15th 2016, Granada, Spain

Grupo Trasgo
Universidad de Valladolid

Departamento de Informática
Universidad de Valladolid

Universidad de Valladolid

Introduction

Objective

Presenting a study about the level of support of OpenACC in available compilers, including maturity level and relative performance of generated code.

State of the Art

About OpenACC

- It is a new parallel programming model.
- It guides automatic parallelization of sequential code.
- It is based on the use of compiler directives or **pragmas**.
- It was designed for GPU and Xeon Phi accelerators.

State of the Art

Levels of Parallelism in OpenACC

Tries to unify concepts for GPUs and Xeon Phi:

- `Gangs`: Coarse-grain level.
- `Workers`: Middle level.
- `Vector Length`: Fine-grain level. Designed for exploiting vectorization capabilities of the Xeon Phi.

State of the Art

Existing OpenACC Compilers

Commercial:

- PGI Compiler.
- CRAY Compiler.
- ENZO Compiler.

Free or Open Source:

- OpenUH: University of Houston, USA.
- accULL: Universidad de La Laguna, Spain.
- Omni: University of Tsukuba, Japan.
- OpenARC: Oak Ridge National Laboratory, USA.
- RoseACC: University of Delaware, USA.
- GCC.

State of the Art

Available OpenACC Compilers

(when this stage of our work was done)

Commercial:

- PGI Compiler.
- CRAY Compiler.
- ENZO Compiler.

Free or Open Source:

- OpenUH: University of Houston, USA.
- accULL: Universidad de La Laguna, Spain.
- Omni: University of Tsukuba, Japan.
- OpenARC: Oak Ridge National Laboratory, USA.
- RoseACC: University of Delaware, USA.
- GCC.

State of the Art

Benchmarking Tools

- OpenACC Validation Testsuite: Pragma, directive and clause validation.
- EPCC OpenACC Benchmark Suite: Three benchmark levels: microbenchmarks, synthetic applications and real applications.
- Rodinia for OpenACC: designed for accelerators. Explores a wide range of problems.

Evaluation

Experimental Setup

Host:

- Xeon E5-2690v3, 12 cores @1.9GHz.
- 64GB memory, 4 * 12GB modules.
- Nvidia GTX Titan Black GPU. 2880 cores @ 980Mhz, 15 SMs, 6GB memory.

Compilers:

- PGI Compiler bundled in Nvidia OpenACC Toolkit version 15.7-0.
- OpenUH: version 3.1.0.
- accULL: version 0.4alpha.

Evaluation

A) Completeness of OpenACC Features Supported

From compiler's documentation and our testing, we conclude that:

- The OpenACC standard is not fully implemented yet by any compiler.
- PGI Compiler is the most complete implementation to date.
- There is work to be done, but compilers are reaching a respectable maturity level.

Evaluation

B) Robustness and Pragma Implementation

We use EPCC OpenACC Level 0 Benchmark Suite.

EPCC Level0	PGI	openUH	accULL
kernels_if	-37.50	Fail	4.54
parallel_if	-30.76	-0.48	1237.02
parallel_private	-21.94	Fail	51.09
parallel_1stpriv	Fail	Fail	-213.83
kernels_comb.	-1.67	-108.43	-127.17
parallel_comb.	-0.05	-2.74	33.38
Update_host	478.63	373.22	548.77
Kernels_Invoc.	Fail	12.76	2398.20
Parallel_Invoc.	31.81	13.47	1377.88
Parallel_reduct.	-14.85	-164.41	-2168.12
Kernels_reduct.	-8.49	-172.31	-2009.11

Evaluation

C) Relative Performance of Generated Code

Data Movement: We use data transfer benchmarks from EPCC OpenACC Level 0 Benchmark Suite.

Using PGI compiler as reference and the geometric mean of ratios to show the results.

Data Size	PGI	openUH	accULL
1kB	1.0	18.93	19.58
1MB	1.0	4.14	4.24
10MB	1.0	2.64	2.72
1GB	1.0	8.75	6.76

Evaluation

C) Relative Performance of Generated Code
Execution: We use EPCC OpenACC Benchmark Suite Level 1 and Application Level.

Using PGI compiler as reference and the geometric mean of ratios to show the results.

Data Size	PGI	openUH	accULL
1kB	1.0	4.39	24.38
1MB	1.0	1.92	4.11
10MB	1.0	1.59	1.63

It was not possible to test bigger data sizes because the binaries produced by some of the compilers could not handle them.

Evaluation

C) Relative Performance of Generated Code

Execution: We try to use Rodinia for OpenACC.

- Many compilation issues with the selected compilers.
- The table shows execution times in milliseconds.
- Only benchmarks compiled with at least 2 compilers are shown.

Exec. time 3 reps	PGI	OpenUH	accULL
gaussian	2440.206	52.491	15422.944
nw	2640.497	652.180	322.101
lud	3803.756	1723.576	Fail
cfd	2677.387	0.846	Fail
hotspot	2386.325	53.219	Fail
pathfinder	5137.865	34.738	Fail
srad2	2488.895	692.063	Fail

Conclusions

- We have developed a benchmarking tool called TORMENT (to be presented in PDP 2017 in March).
- OpenACC standard and its compiler implementations are on their way to a reasonable maturity level.
- Many details are still not completely developed, but the efforts are promising.
- No compiler fully supports the standard.
- In terms of robustness and pragma implementation, the PGI Compiler show the best behaviour, with a smaller overhead and several optimizations.
- Regarding performance, the PGI Compiler gets the best results, but both OpenUH and accULL show promising numbers.

Future Work

- We are working on our benchmark tool, TORMENT. Working on adding support for more compilers.
- We are also working with relative performance comparison of OpenACC and CUDA code.
- We are studying the impact of specific block geometries in the behaviour of OpenACC kernels.

Comparative Analysis of OpenACC Compilers

Daniel Barba, Arturo Gonzalez, Diego R. Llanos
Grupo Trasgo
Universidad de Valladolid

TAPEMS, ICA3PP2016
December 15th 2016, Granada, Spain

Grupo Trasgo
Universidad de Valladolid

Departamento de Informática
Universidad de Valladolid

Universidad de Valladolid