ICHEC
Irish Centre for High-End Computing

Investigating Performance Benefits from OpenACC Kernel Directives

Ben Eagan[1,2], Gilles Civario[1], **Renato Miceli**[1,3]

[1] Irish Centre for High-End Computing (ICHEC), Ireland

[2] BlackBerry, Canada

[3] Université de Rennes 1, France

beagan@blackberry.com, {gilles.civario, renato.miceli}@ichec.ie

Science Foundation Ireland

Ireland's EU Structural Funds Programmes 2007 - 2013

Co-funded by the Irish Government and the European Union

EUROPEAN REGIONAL DEVELOPMENT FUND

AN ROINN OIDEACHAIS AGUS EOLAIOCHTA | DEPARTMENT OF EDUCATION AND SCIENCE

HEA

Higher Education Authority

An tÚdarás um Ard-Oideachas

Motivations and Goals

- Motivations:
  - Several OpenACC compilers exist
  - Shift to GPU accelerated applications is now
  - Performance optimisation can provide big gains
- Goals:
  - Evaluate the sensitivity of OpenACC work splitting configuration parameters
  - Understand how far one can go on real-life practical OpenACC performance tuning
  - Provide insights for future auto-tuning heuristics

OpenACC: Overview

- Directive-based API for accelerators (C/C++, Fortran)
- Allows for incremental parallelisation, quick accelerator porting
- Easier to use than CUDA and OpenCL
- CAPS, PGI/NVIDIA and Cray currently have OpenACC compiler support
  - Other minor compilers such as accULL exist
  - Likely more to come

OpenACC

Directives for Accelerators

Worksharing

- Worksharing loops available with OpenACC, where iterations are divided amongst *device* processors
- Kernels can also be written for more general parallelisation
- Accelerators sensitive to scheduling based on hardware restrictions
- Several constructs available for further customisation of computation (e.g. data transfers, loop scheduling)

OpenACC Scheduling

- Three different scheduling constructs:
  - **gangs**: denotes the number of parallel workgroups to be created on the GPU
  - **vectors**: describes the number of iterations to be computed with vector/SIMD processing
  - **workers**: how many groups of parallel work units are distributed within a single gang

Our Approach

- Focused on **gang** and **vector** scheduling constructs for measuring scheduling impact
- Examined two codes:
  - Matrix-matrix multiply
  - Classical Gram-Schmidt orthonormalisation
- Verified with two OpenACC compilers:
  - CAPS
  - PGI/NVIDIA
- On two NVIDIA GPUs: M2090 and K20

Our Approach

- Runtime capture:
  1. Serial code
  2. 4-threaded shared memory OpenMP
  3. OpenACC code
     (all-pairs testing of 2^n of gang/vector values)
- Best pair of values for one platform trialed on another platform
  - Values capped at 512 (PGI) and 1024 (CAPS)

Matrix-Matrix Multiply

- Naïve algorithm with $O(n^3)$ time complexity
- Takes dense 8000x8000 arrays of floats
- Parallelisation: single loop
  - **kernels loop independent** directive
  - **gang** on outermost
  - **vector** on middle
  - **worker** on innermost

Matrix-Matrix Multiply

Table 1. Runtimes for different implementations of the matrix-matrix multiply using the PGI compiler

Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	311.9s	1.00
OpenMP	–	–	Intel Xeon X5560	–	81.0s	3.85
OpenACC	auto	auto	Intel Xeon X5560	NVIDIA M2090	39.1s	7.98
OpenACC	64	256	Intel Xeon X5560	NVIDIA M2090	22.6s	13.80

Table 2. Runtimes for different implementations of the matrix-matrix multiply using the CAPS compiler.

Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	207.0s	1.0
OpenMP	–	–	Intel Xeon X5560	–	82.5s	2.51
OpenACC	auto	auto	Intel Xeon X5560	NVIDIA M2090	65.3s	3.17
OpenACC	1024	256	Intel Xeon X5560	NVIDIA M2090	21.0s	9.86
OpenACC	1024	512	Intel Xeon X5560	NVIDIA M2090	21.3s	9.72
OpenACC	auto	auto	Intel Xeon E5-2643	NVIDIA K20	17.3s	11.96
OpenACC	1024	256	Intel Xeon E5-2643	NVIDIA K20	17.8s	11.63
OpenACC	1024	512	Intel Xeon E5-2643	NVIDIA K20	16.2s	12.78

Matrix-Matrix Multiply

Table 1. Runtimes for different implementations of the matrix-matrix multiply using the PGI compiler

Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	311.9s	1.00
OpenMP	–	–	Intel Xeon X5560	–	81.0s	3.85
OpenACC	128	??	Intel Xeon X5560	NVIDIA M2090	39.1s	7.98
OpenACC	64	256	Intel Xeon X5560	NVIDIA M2090	22.6s	13.80

Table 2. Runtimes for different implementations of the matrix-matrix multiply using the CAPS compiler.

Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	207.0s	1.0
OpenMP	–	–	Intel Xeon X5560	–	82.5s	2.51
OpenACC	auto	auto	Intel Xeon X5560	NVIDIA M2090	65.3s	3.17
OpenACC	1024	256	Intel Xeon X5560	NVIDIA M2090	21.0s	9.86
OpenACC	1024	512	Intel Xeon X5560	NVIDIA M2090	21.3s	9.72
OpenACC	auto	auto	Intel Xeon E5-2643	NVIDIA K20	17.3s	11.96
OpenACC	1024	256	Intel Xeon E5-2643	NVIDIA K20	17.8s	11.63
OpenACC	1024	512	Intel Xeon E5-2643	NVIDIA K20	16.2s	12.78

Matrix-Matrix Multiply

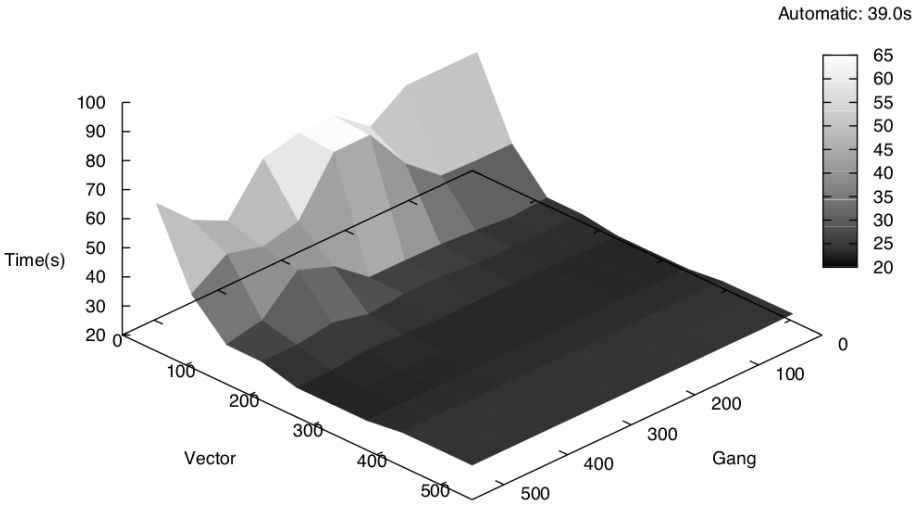
Table 1. Runtimes for different implementations of the matrix-matrix multiply using the PGI compiler

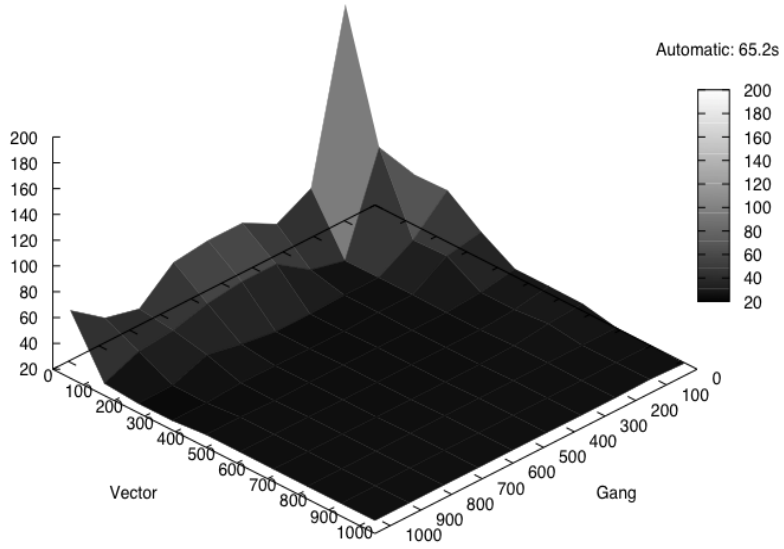
Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	311.9s	1.00
OpenMP	–	–	Intel Xeon X5560	–	81.0s	3.85
OpenACC	128	??	Intel Xeon X5560	NVIDIA M2090	39.1s	7.98
OpenACC	64	256	Intel Xeon X5560	NVIDIA M2090	22.6s	13.80

Table 2. Runtimes for different implementations of the matrix-matrix multiply using the CAPS compiler.

Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	207.0s	1.0
OpenMP	–	–	Intel Xeon X5560	–	82.5s	2.51
OpenACC	128	32	Intel Xeon X5560	NVIDIA M2090	65.3s	3.17
OpenACC	1024	256	Intel Xeon X5560	NVIDIA M2090	21.0s	9.86
OpenACC	1024	512	Intel Xeon X5560	NVIDIA M2090	21.3s	9.72
OpenACC	auto	auto	Intel Xeon E5-2643	NVIDIA K20	17.3s	11.96
OpenACC	1024	256	Intel Xeon E5-2643	NVIDIA K20	17.8s	11.63
OpenACC	1024	512	Intel Xeon E5-2643	NVIDIA K20	16.2s	12.78

Matrix-Matrix Multiply





PGI®

CAPS

Results: Matrix-Matrix Multiply

- Hints:
  - Vectors smaller than 128 give bad performance
  - Gang sizes on PGI: invariant
    (we suspect PGI handles one of the dimensions to fit the no. of iterations to the problem size)
  - Best pair of values on one platform is unlikely to yield good performance on a different platform
- PGI does not provide enough info about its automatic behaviour

CGS Orthonormalisation

- Algorithm of $O(n^2)$ time complexity
- Takes arrays of 2000x2000 pseudorandom floats
- Parallelisation:
  - Several parallel loops (cf. matrix-matrix multiply)
  - **parallel loop** directive
  - **vector** and **gang** sizes inline

CGS Orthonormalisation

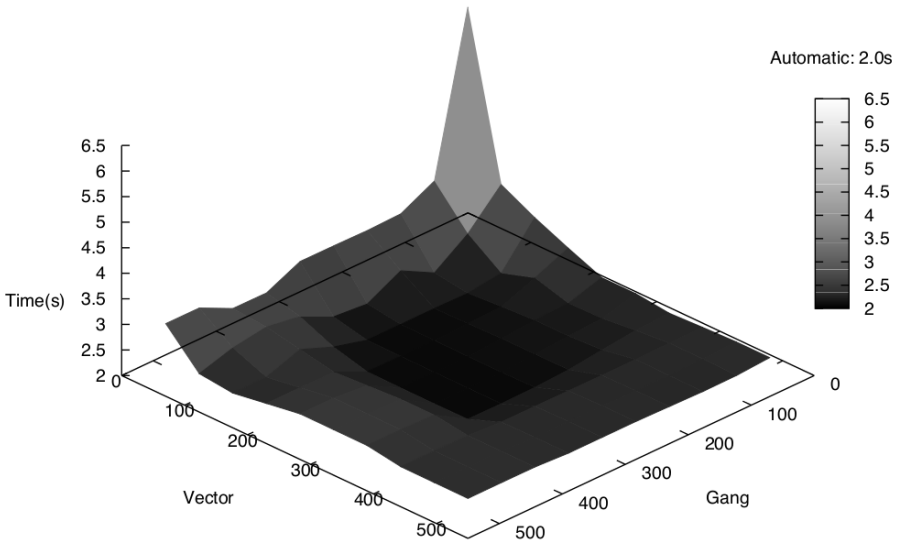
Table 3. Runtimes for different implementations of CGS orthonormalisation using the PGI compiler.

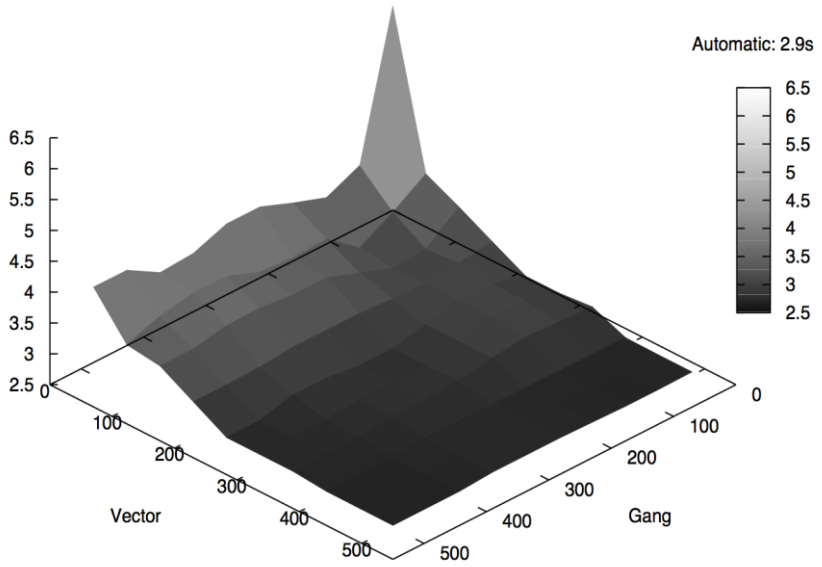
Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	13.8s	1.00
OpenMP	–	–	Intel Xeon X5560	–	3.9s	3.54
OpenACC	auto	auto	Intel Xeon X5560	NVIDIA M2090	2.0s	6.90
OpenACC	256	256	Intel Xeon X5560	NVIDIA M2090	2.0s	6.90

Table 4. Runtimes for different implementations of CGS orthonormalisation using the CAPS compiler.

Variant	Gang	Vector	CPU	GPU	Runtime	Speedup
Serial	–	–	Intel Xeon X5560	–	8.9s	1.00
OpenMP	–	–	Intel Xeon X5560	–	3.0s	2.97
OpenACC	auto	auto	Intel Xeon X5560	NVIDIA M2090	2.9s	3.07
OpenACC	512	512	Intel Xeon X5560	NVIDIA M2090	2.7s	3.30

CGS Orthonormalisation





PGI®

CAPS

Results: CGS Orthonormalisation

- Hints:
  - The **parallel** directive provides better performance than the **kernels** directive
  - Performance losses for small gangs and vectors (possible causes are (i) added overhead of parallelising workgroups and small vectors, and (ii) wasting warps by using only half of it per schedule)

Conclusions

- Demonstrated the ability to improve OpenACC parallelisations using the **gang** and **vector** clauses
  - Hints about the sensitivity of work sharing directives
- Shows OpenACC as an exciting tool for simple parallelisation, portability, and auto-tuning canvas
- Future work potential:
  - Newer releases of the standard: OpenACC 2.0+
  - More mature compilers and other accelerators
  - Different code/algorithm structures
  - Knowledge reverted to auto-tuning strategies

Questions?

sfi Science Foundation Ireland

Ireland's EU Structural Funds Programmes 2007 - 2013

Co-funded by the Irish Government and the European Union

EUROPEAN REGIONAL DEVELOPMENT FUND

AN RÓINN OIDEACHAIS AGUS EOLAÍOCHTA | DEPARTMENT OF EDUCATION AND SCIENCE

HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas

Software Packages

- PGI 12.8
- CAPS 3.3.1
- Intel 2011.9.293
- CUDA 5.0