AIWC: OpenCL-based Architecture-Independent Workload Characterization

THE AUSTRALIAN NATIONAL UNIVERSITY

Beau Johnston and Josh Milthorpe

The Australian National University

November 12, 2018

AIWC: OpenCL-based Architecture-Independent Workload Characterization

Beau Johnston and Josh Milthorpe
The Australian National University

November 12, 2018

Trends in Supercomputing – A view from the Top500

1. Summit – GV100
2. Sunway TaihuLight
3. Sierra – GV100
4. Tianhe-2A
5. ABCI – V100
6. Piz Daint – P100
7. Titan – K20x
8. Sequoia
9. Trinity – Phi
10. Cori – Phi

Trends in Supercomputing – A view from the Top500

1. Summit – GV100
2. Sunway TaihuLight
3. Sierra – GV100
4. Tianhe-2A
5. ABCI – V100
6. Piz Daint – P100
7. Titan – K20x
8. Sequoia
9. Trinity – Phi
10. Cori – Phi

- As of June
- 7 / 10 use accelerators
- Newest is Summit – based on IBM Power9 with NVLINK and CAPI
- All devices have an OpenCL runtime
- Reliance on accelerators is increasing
- Scheduling to the most appropriate device is an open problem

OpenCL – The Language of Heterogeneous Computing

- Open Computing Language (OpenCL) is an open standard.
- Allows computationally intensive codes – kernels – to be written once and run on any compliant accelerator.
- Most vendors are compliant to basic standards.
- Application code can be written directly in OpenCL, and
- Can be used as a back-end for higher level languages – OpenMP runtime implemented for TI Keystone II DSP architecture[1].
- Increased relevancy for FPGA programming

[1] Mitra, G. et al. 2014. Implementation and optimization of the OpenMP accelerator model for the TI Keystone II architecture. International workshop on openmp (2014), 202–214.

The Problem

- Many heterogenous accelerators running OpenCL in current and future supercomputers, and this trend is continuing
- **But** their performance is as diverse as each accelerators hardware configuration
- An architecture-independent method to characterize OpenCL codes allows us to:
    - measure inherent program behaviour
    - perform accurate performance predictions for scheduling

Workload characterization with AIWC

- Architecture-Independent Workload Characterization (AIWC)
- Plugin for OclGrind – an Extensible OpenCL device simulator[2]
- Beta available – https://github.com/BeauJoh/Oclgrind – and will be merged into default OclGrind

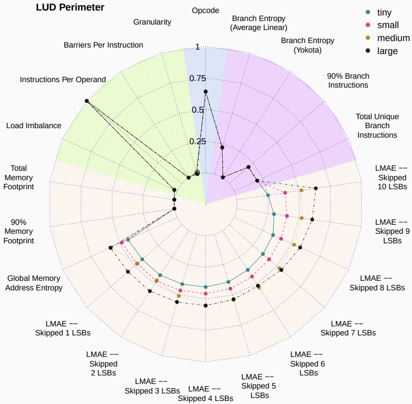
[2] J. Price and S. McIntosh-Smith, "Oclgrind: An extensible opencl device simulator," in Proceedings of the 3rd International Workshop on OpenCL, 2015, p. 12.

Overview of AIWC

- Simulation of OpenCL kernels occur on LLVM IR – SPIR
- AIWC tracks and measures hardware agnostic events
- Metrics carefully selected and collected during simulator execution
- Large number of metrics collected (28)
- Over a wide spectrum computation, thread communication and memory access patterns
- Supports parallel workloads
- Accessible – as part of OclGrind
- High-accuracy – full resolution, not interrupt/sample driven

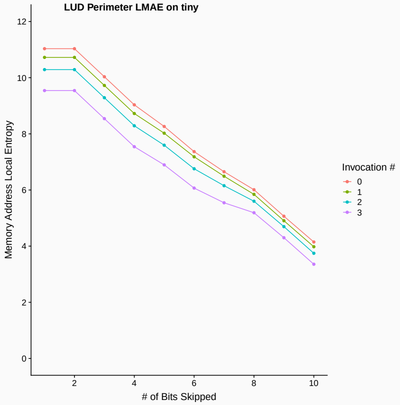
AIWC Example

- Four major classes: Compute, Parallelism, Memory, Control
- Different statistics per metric – distributions, entropy and absolute counts

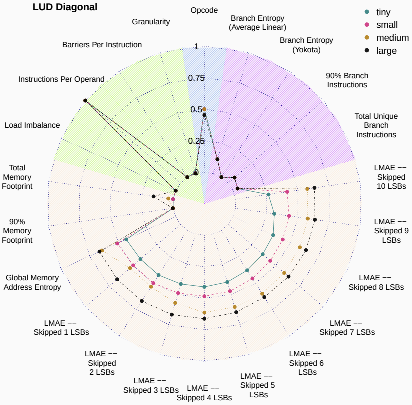


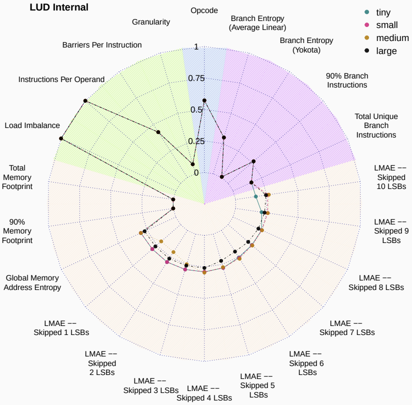
AIWC Example II

- Local Memory Address Entropy
- Kernel launched 4 times – over different problem sizes
- Starting entropy changes with problem size, but same gradient → memory access patterns are the same regardless of actual problem size
- Steeper descent → more localised memory access → better cache utilization



AIWC Example III





Subset of AIWC Metrics

Type	Metric	Description
Compute	Opcode	total # of unique opcodes required to cover 90% of dynamic instructions
Compute	Total Instruction Count	total # of instructions executed
Parallelism	Work-items	total # of work-items or threads executed
Parallelism	Total Barriers Hit	total # of barrier instructions
Parallelism	Median ITB	median # of instructions executed until a barrier
Parallelism	Max IPT	maximum # of instructions executed per thread
Parallelism	Mean SIMD Width	mean # of data items operated on during an instruction
Memory	Total Memory Footprint	total # of unique memory addresses accessed
Memory	90% Memory Footprint	# of unique memory addresses that cover 90% of memory accesses
Memory	Unique Read/Write Ratio	indication of workload being (unique reads / unique writes)
Memory	Reread Ratio	indication of memory reuse for reads (unique reads/total reads)
Memory	Global Memory Address Entropy	measure of the randomness of memory addresses
Memory	Local Memory Address Entropy	measure of the spatial locality of memory addresses
Control	Total Unique Branch Instructions	total # of unique branch instructions
Control	90% Branch Instructions	# of unique branch instructions that cover 90% of branch instructions
Control	Yokota Branch Entropy	branch history entropy using Shannon's information entropy
Control	Average Linear Branch Entropy	branch history entropy score using the average linear branch entropy

For an exhaustive list see the paper
https://arxiv.org/abs/1805.04207

A larger AIWC Corpus

- Just a sample of 1 of 11 applications
- Similar breakdown of 37 kernels
- Extended OpenDwarfs Benchmark Suite used for corpus[3]
- Presented in a docker & jupyter artefact

[3] B. Johnston and J. Milthorpe, "Dwarfs on accelerators: Enhancing OpenCL benchmarking for heterogeneous computing archi- tectures," in Proceedings of the 47th international conference on parallel processing companion, 2018, pp. 4:1–4:10.

Sample of the AIWC Corpus

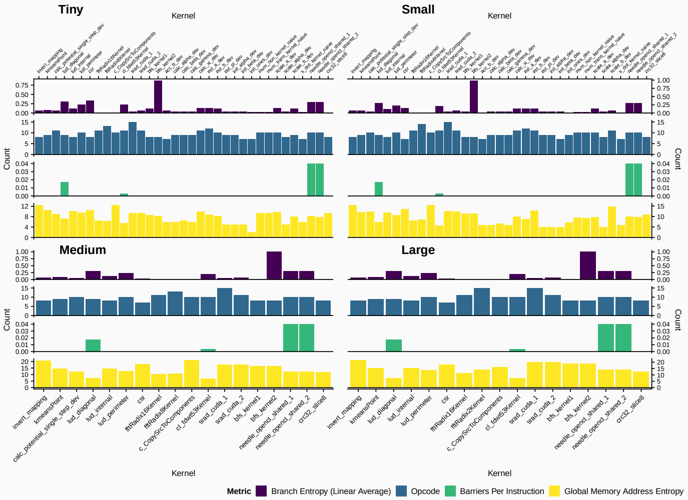


Figure 1: Selected AIWC metrics from each category over all kernels and 4 problem sizes.

EOD

- 4 sizes per application
- Runtime data already collected on:
  - 15 devices – GPU, CPU and MIC
  - Time, performance events and energy (x50)
- High-resolution measurements per kernel with LibSciBench[4]

[4] T. Hoefler and R Belli. "Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015. ACM, 73.

What now?

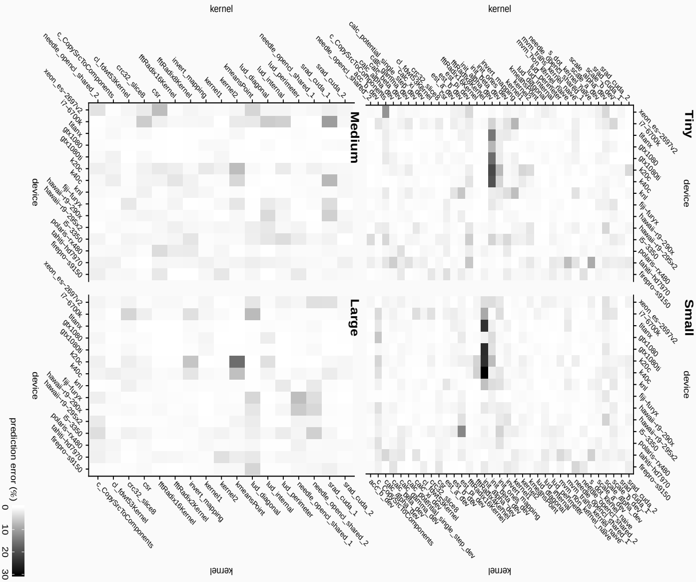
- Benchmark suite (all dwarfs) → wide diversity of scientific application codes
- Forms a large range of result times with matched AIWC feature-spaces

Performance Prediction – Combining EOD and AIWC

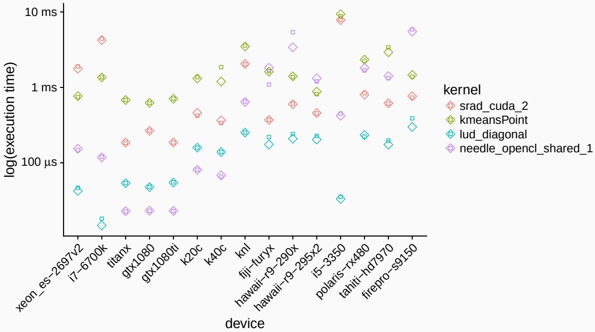
- Regression model from AIWC was developed[5]
- Predictor variables: 28 AIWC metrics
- Response variable: time or energy of kernel execution
- R language and Ranger – a Random Forest implementation – was used
- Performs recursive partitioning of high dimensional data
- Accepts 3 parameters:
  - num.trees – number of trees grown in forest – 10-10k by 500
  - mtry – number of features tried to split in a node – 1-34
  - min.node.size – minimal size per tree – 1-50
- Optimal model needs careful tuning

[5] B. Johnston, G. Falzon and J. Milthorpe, "OpenCL Performance Prediction using Architecture-Independent Features," in 2018 International Conference on High Performance Computing & Simulation (HPCS) (pp. 561-569). IEEE.

Performance Prediction Example



Scheduling – A Prototype



- 4 random kernels.
- square → mean measured time
- diamond → mean predicted time
- order is important!

Future Work – Scheduling

- The predictions are highly accurate, differing from the measured experimental run-times by an average of only 1.2%
- Correspond to execution time mispredictions of 9 μs to 1 sec according to problem size.
- Previously unencountered code can be instrumented once, AIWC metrics embedded for quick performance prediction.
- Scheduler integration? StarPU, Ompss, CoreTsar or AutoMatch?

Future Work – Guiding Optimization

- Vendor-recommended/device-specific optimizations and how do the AIWC metrics change?
- Guide a developer to assess:
  - how algorithmic changes → broader characteristics – e.g. no device benefits from more sporadic memory accesses
  - performance portability
  - predicting execution time without having access to these systems (or time to test)

Future Work – OpenACC and Sunway

- Colleagues at Shanghai Jiao Tong University's HPC Center,
- Interested in performance properties of scientific codes on the Sunway TaihuLight and Tianhe-2A and future Tianhe-3,
- Specifically by extending AIWC for OpenACC,
- TaihuLight's Computer Processing Element (CPE) 8x8 mesh of cores – just another accelerator really...
- Only Sequoia left without AIWC support in top 10 of the Top500.

Conclusions

- Presented Architecture-Independent Workload Characterization tool:
  - Supports the collection of architecture-independent features of OpenCL application kernels.
  - First workload characterization tool to support multi-threaded and/or parallel workloads.
- Features can be used to:
  - predict the most suitable device for a particular kernel – for scheduling,
  - or to determine the limiting factors for performance,
  - allows developers to try alternative implementations (e.g. by reorganizing branches, eliminating intermediate variables . . .).
  - inform accelerator designers & integrators of a scientific workloads – ensures compute architectures are suitable for intended workloads.
- AIWC metrics generate a comprehensive feature-space

Open Questions and Next Steps

- Potential to be more prescriptive than the Berkeley Dwarf Taxonomy
- Do you have unusual/real-world applications:
  - that you think we've missed?
  - large applications that could benefit from predictions?
  - where you think the predictions will fail?
- Quickly test with Docker & Jupyter artefact.
- Building a prototype scheduler using our predictions?
- Access/Interest in other hardware?
- What does this mean for FPGAs?

Open Questions and Next Steps – Continued.

- Have suggestions for more/better AIWC metrics – or think we've missed some?
- Interested in presenting a reduced feature space?
- Think this descriptive tool to guide developers would be useful? Your input would be appreciated!
- Have applications with weird optimization results? Will changes in metrics show this?
- Anyone an expert in OpenACC back-end – translation to OpenCL kernels for Oclgrind?
- We should collaborate!

Thanks

- The University of Bristol's High Performance Computing Research group for the use of "The Zoo" Research cluster
- You!

beau.johnston@anu.edu.au

Extended Open Dwarfs Benchmark Suite

- Extended Open Dwarfs (EOD) Benchmark Suite
- Based off the OpenDwarfs benchmark suite[6]
- Benchmarks selected following diversity analysis and 13 Berkeley Dwarfs taxonomy
- Built in OpenCL
- Purpose of OpenDwarfs was a to characterize a diverse set of parallel applications and architectures using a common language, but had deficiencies. . .

[6] Krommydas, K. OpenDwarfs: Characterization of dwarf-based benchmarks on fixed and reconfigurable architectures. Journal of Signal Processing Systems, vol. 85, no. 3, pp. 373-392, 2016

EOD Extensions

- Selection of problem size is critically affects HPC benchmarking
- Highest impact on CPU architectures.
- A major contribution of the work is facilitating 4 different problem sizes for all applications presented in the suite.
- Selected according to levels of cache
    - **tiny** : $<$ 32 KiB L1
    - **small**: $<$ 256 KiB L2
    - **medium**: $<$ 8192 L3
    - **large**: $>$ 8192 L3

EOD Extensions – Continued

- Diverse:
  - 4 different problem sizes per application
  - Added applications – currently 11 and 37 kernels
  - Real applications sampled from Bioinformatics, Computational Biology, Computational Chemistry and other fields
- Reproducible: Minimum of 2 sec runs per benchmark
- Precise:
  - High resolution timers with LibSciBench
  - Reported with one cycle resolution and roughly 6 ns of overhead
  - Also allows collection of energy and hardware events
- Portable:
  - Based on an OpenCL backend
  - Tested on a wide range of hardware
  - Consistent tuning – i.e. workgroup size arguments

Applications

Table 1: List of Extended OpenDwarfs Applications and their respective dwarfs

Dwarf	Extended OpenDwarfs Application
Dense Linear Algebra	LU Decomposition
Sparse Linear Algebra	Compressed Sparse Row
Spectral Methods	DWT2D, FFT
N-Body Methods	Gemnoui
Structured Grid	Speckle Reducing Anisotropic Diffusion
Unstructured Grid	Computational Fluid Dynamics
Map Reduce	K-Means
Combinational Logic	Cyclic-Redundancy Check
Graph Traversal	Breadth First Search
Dynamic Programming	Smith-Waterman
Backtrack and Branch and Bound	N-Queens
Graphical Methods	Hidden Markov Models
Finite State Machines	Temporal Data Mining

Hardware

Name	Vendor	Type	Series	Core Count	Clock Frequency (MHz) (min/max/turbo)	Cache (KiB) (L1/L2/L3)	TDP (W)	Launch Date
Xeon E5-2697 v2	Intel	CPU	Ivy Bridge	24∗	1200/2700/3500	32/256/30720	130	Q3 2013
i7-6700K	Intel	CPU	Skylake	8∗	800/4000/4300	32/256/8192	91	Q3 2015
i5-3550	Intel	CPU	Ivy Bridge	4∗	1600/3380/3700	32/256/6144	77	Q2 2012
Titan X	Nvidia	GPU	Pascal	3584†	1417/1531/–	48/2048/–	250	Q3 2016
GTX 1080	Nvidia	GPU	Pascal	2560†	1607/1733/–	48/2048/–	180	Q2 2016
GTX 1080 Ti	Nvidia	GPU	Pascal	3584†	1480/1582/–	48/2048/–	250	Q1 2017
K20m	Nvidia	GPU	Kepler	2496†	706/–/–	64/1536/–	225	Q4 2012
K40m	Nvidia	GPU	Kepler	2880†	745/875/–	64/1536/–	235	Q4 2013
FirePro S9150	AMD	GPU	Hawaii	2816‖	900/–/–	16/1024/–	235	Q3 2014
HD 7970	AMD	GPU	Tahiti	2048‖	925/1010/–	16/768/–	250	Q4 2011
R9 290X	AMD	GPU	Hawaii	2816‖	1000/–/–	16/1024/–	250	Q3 2014
R9 295x2	AMD	GPU	Hawaii	5632‖	1018/–/–	16/1024/–	500	Q2 2014
R9 Fury X	AMD	GPU	Fuji	4096‖	1050/–/–	16/2048/–	273	Q2 2015
RX 480	AMD	GPU	Polaris	4096‖	1120/1266/–	16/2048/–	150	Q2 2016
Xeon Phi 7210	Intel	MIC	KNL	256‡	1300/1500/–	32/1024/–	215	Q2 2016

EOD Evaluation

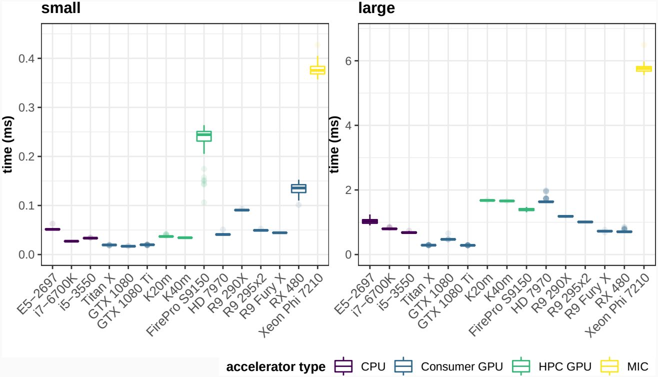
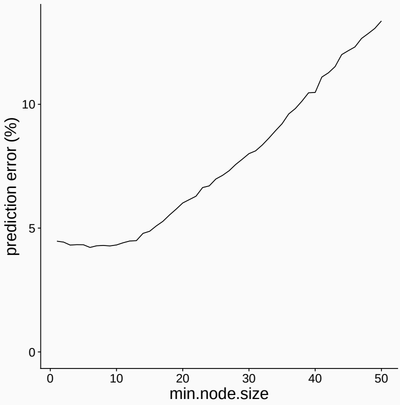


Figure 2: Comparison of performance on 2 sizes of csr application.

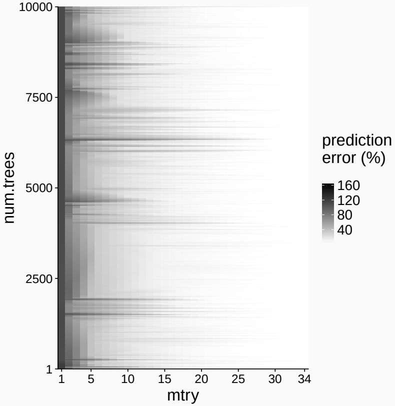
min.node.size



- Full coverage of min.node.size with fixed tuning parameters: num.trees = 300 and mtry = 30
- Smallest out-of-bag prediction error for values $<$ 15
- Selection made to fix min.node.size = 9

num.trees and mtry

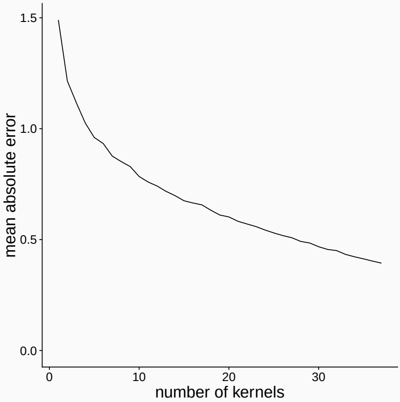
- optim_sa function used to find global minimum
- Full coverage achieved – 4 outer-most points and 8 random starting internal points
- intermediate results used and interpolation performed – using akima
- Model performance varies significantly for last 2 variables
- mtry $>$ 25, offers good fit
- num.trees less impact – fewer



Choosing Parameters for the Future

- num.trees=500, mtry=32, and min.node.size=9 look good
- train on a random selection of N kernels and test on remainder
- see paper for details but final values are num.trees = 505, mtry = 30 and min.node.size = 9

Increased Training Data



- How many kernels to add for training – what's enough?
- Another study performed to see how error changes w.r.t. number of kernels in training
- Uses random selection for each random count – again see paper for full details
- Error tapers off for more kernels!
- gradient still significant at 37 kernels → could still benefit from more.

Prediction Accuracy