Performance Prediction on Heterogeneous Architectures: Challenges and Insight

SILVIO STANZANI, ROGÉRIO IOPE, RAPHAEL CÓBE, JEFFERSON FIALHO, MARCO GOMES, ARTUR BARUCHI, JÚLIO AMARAL

Agenda

NCC (Center for Scientific Computing)

Motivations

Performance prediction approaches

Performance prediction challenge

Performance models

performance prediction Examples

Center for Scientific Computing

GridUnesp

High Performance Computing

- First Campus Grid in Latin America
- HPC for 400+ users

SPRACE

High Energy Physics

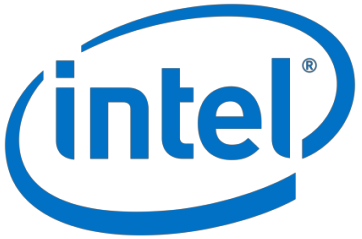
- CMS / CERN
- Search for DM
- Tier-2 of WLCG
- Instrumentation

Openlab

Digital Social Innovation

- Cloud Computing
- Code Parallelization
- Machine Learning
- SDN

Intel R&D Projects



Manycore Testing Lab

- First manycore testing lab outside US
- First hands-on activities with Xeon Phi

Intel Parallel Computing Center

- Parallelization of Geant code
- Broad impact
  - HEP + Dosimetry + Radiation-hard electronics
- Goals
  - Develop GeantV: massive parallelism natively

Intel Modern Code Program (IMC)

- 1700+ students trained
- 7 International training events
- 26 tutorials at Brazilian Institutions

CoE in Machine Learning

- R&D, consulting, and training in ML
- High Energy Physics (boosted jets)
- SERPRO, DataPrev, Banks, City Halls, etc.

Motivations

Apply Machine Learning techniques on data about the execution of applications in order to predict the performance

Static:

- Parameters
- Input and Output Data
- Scheduler trace information

  Amount of resources chosen;

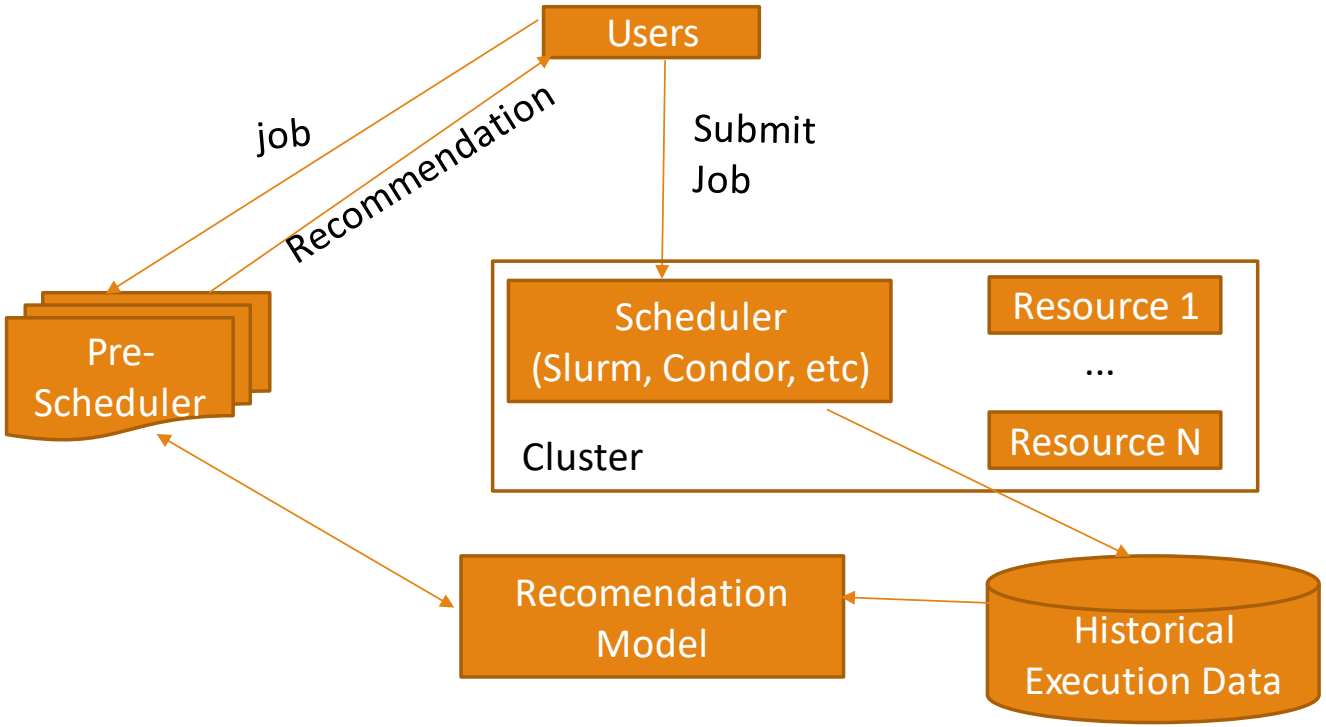
  Type of resources

  Execution time

  Wait Time

- Dynamic:
  - Profilers

    Vtune, Advisor, perf, ...
  - Performance Monitoring Units (PMU)



Motivations

Predict the performance of applications is useful for several areas

- Job scheduling
  - Any information about performance can increase the quality of job scheduling
- Runtime Decisions
  - Change the execution plan of applications

    Increase amount of threads, migrate to another resources, etc
- Guide optimization
  - Predict the performance that different combinations of optimizations can present on the same architecture
- Understand the impact of computational architecture resources on an application
  - "My code will run faster on a new generation of Intel processor?"

Performance Prediction - Approaches

Knowledge about application's input Data

- Several small kernels are used to obtain a pattern about performance of each phase (hotspot) of application
  - Each hotspot profile is compare against other similar profiles in a dataset to infer the performance
- Iterative applications
  - Measure the execution time of each iteration across different resources
- Machine learning methods to approximate input data to execution time
- Combination between input data and hardware events
- Based on the execution using only small set of resources, predict the scalability on a huge amount of resources

Performance Prediction - Approaches

Knowledge about application's Source Code

- include annotation in source code in order to model performance
  - Intel Advisor (Thread Analysis)
- Compile with LLVM compiler to characterize the performance of applications on different architectures
- Static analysis of memory model to predict performance
  - compilers provides performance reports by default
  - Directives can improve this reports

Performance Prediction - Approaches

Analysis based model based on simulation

- Transform source code into an application model describing its computation and communication behavior,
  - Execute the application model on a simulated HPC architecture for performance analysis.
- Static analysis of source code and analytical evaluation to predict performance
  - "COMPASS: A Framework for Automated Performance Modeling and Prediction"

Performance Prediction - Challenge

Performance prediction transparent for the user

- No access to source code
  - Without directives, annotations, or special compilations directives (**-g** for instance)
- No relation between Input data and performance need to be informed by the user
- Time limit constraint
  - We want to deliver this prediction executing only a small fraction of application

Performance Prediction -Challenge

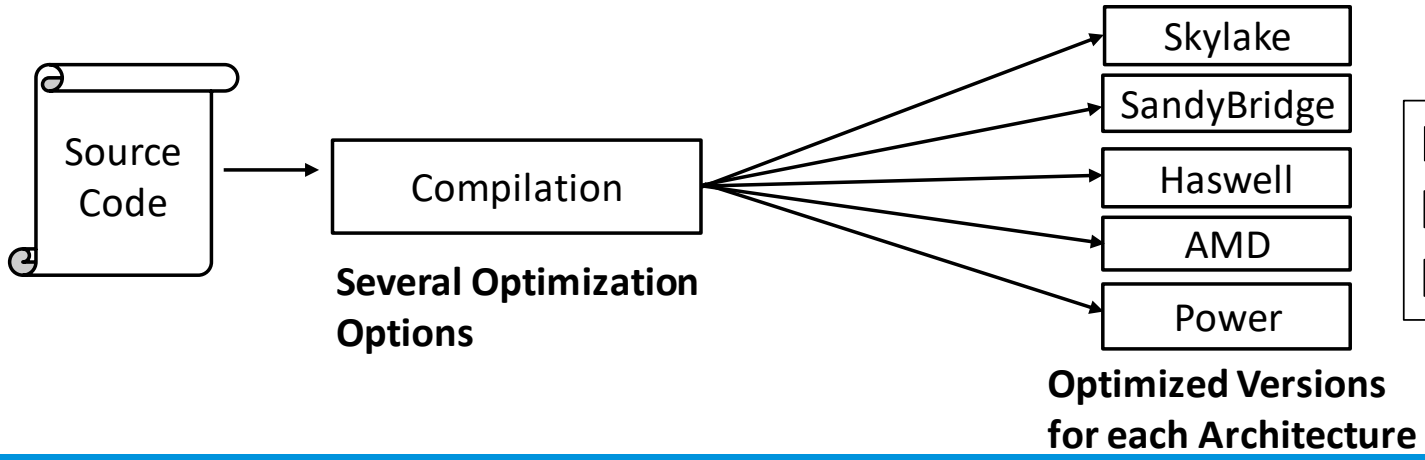
Applications typically perform **very** different on different architectures

Several combination of optimizations can be used for the same application

For each architecture a very different binary code with different behavior can be generated

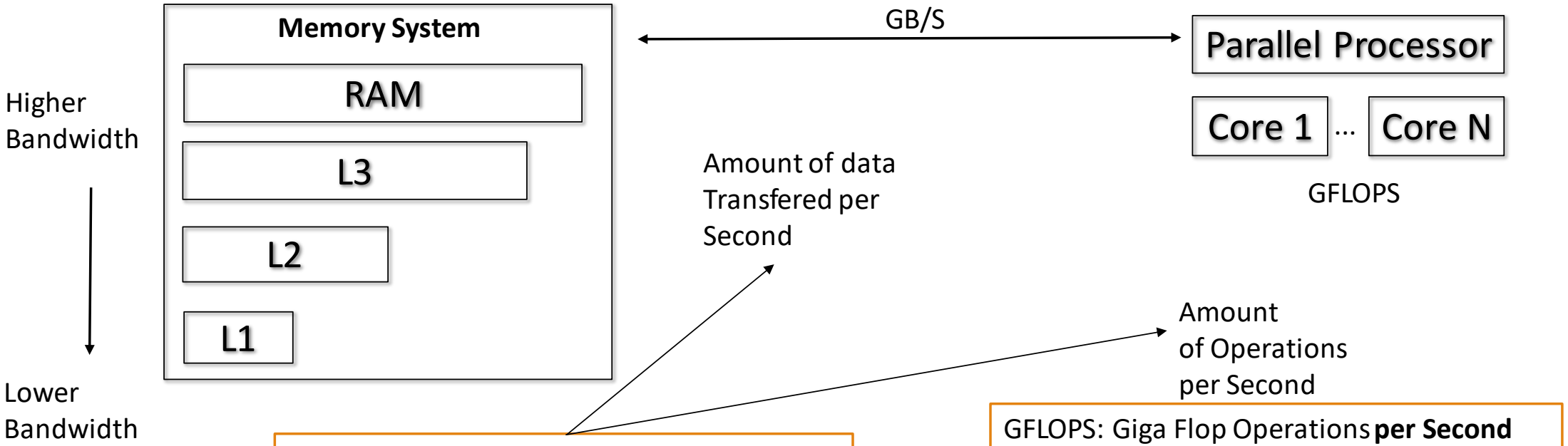
- avx, avx2, avx512, …

Performance Monitoring Units very different across architectures



How to identify patterns to learn how an application will behave on each architecture?

Performance Model



Balance between processing and transfer:

- Higher GFLOPS Demands more GB/S
- Lower GLOPS limits potential GB/s

GFLOPS: Giga Flop Operations **per Second**
GFLOPs: Giga Flop Operations

How to Measure such Balance?
Arithmetic Intensity (AI): Ratio between work performed and data transfered: Flop/Byte.

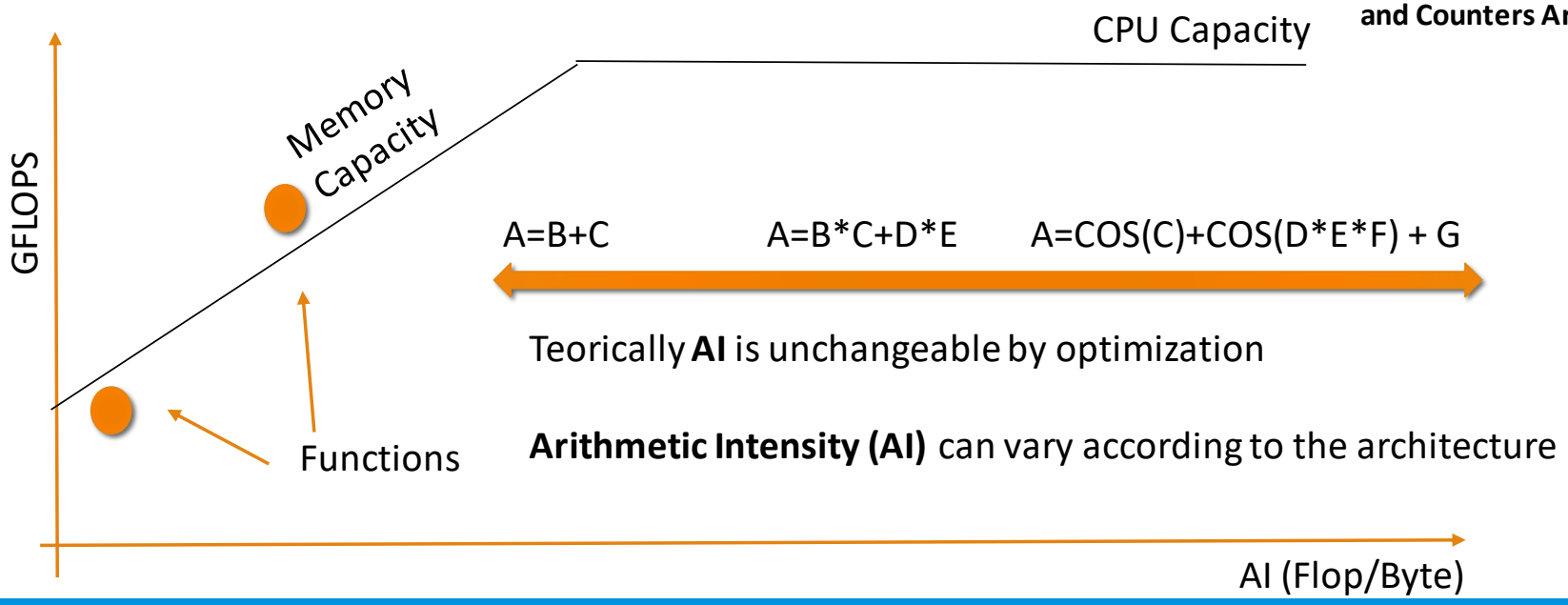
"Memory bandwidth and machine balance in high performance computers" John D. McCalpin. IEEE TCCA 1995.

Performance Model

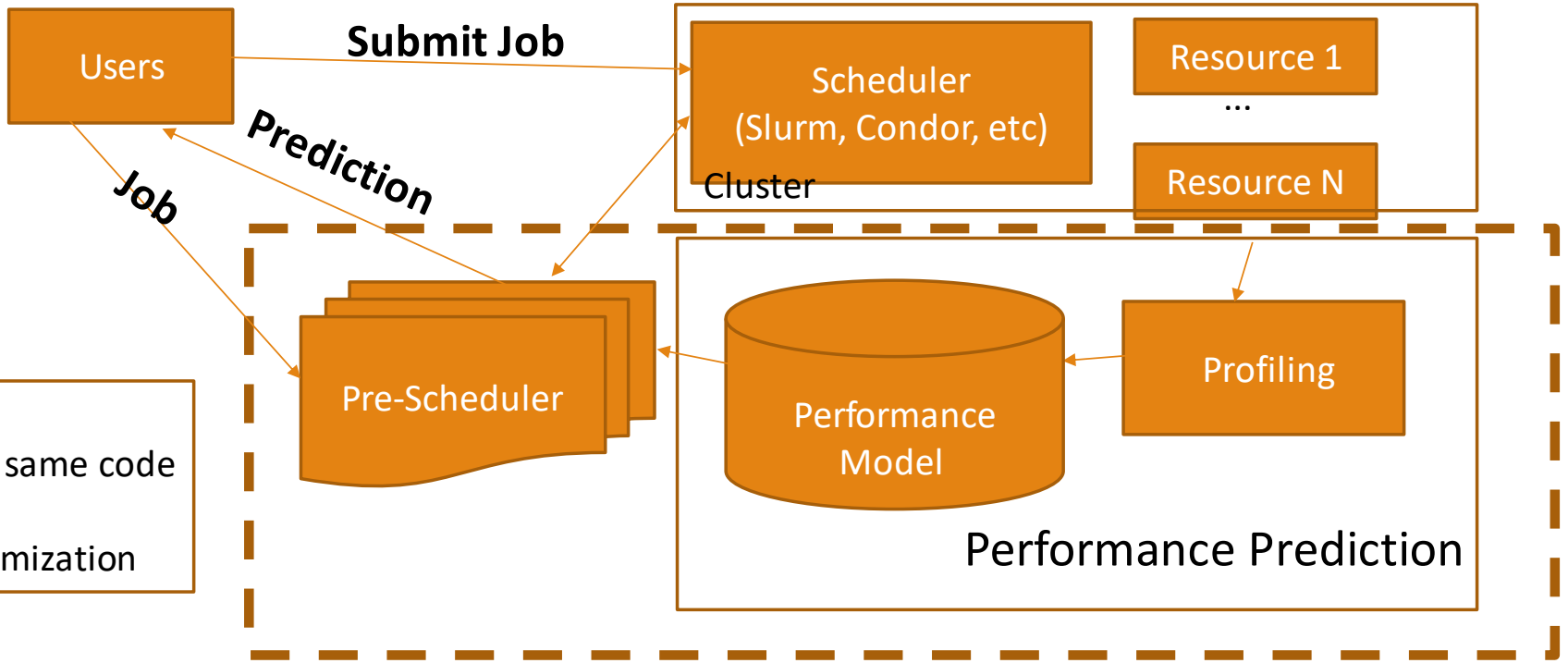
Roofline Model: [1]
- Visualization of how far the performance of functions if from theoretical limits (ceiling)
- Guide optimization using hardware profiling **[2]**

[1] Samuel Williams, Andrew Waterman, and David Patterson. **Roofline: an insightful visual performance model for multicore architectures**. *Commun. ACM* 52, 4 (April 2009)

[2] A Top-Down Method for Performance Analysis and Counters Architecture, Ahmad Yasin. ISPASS 2014



Performance Prediction - Challenge



- **User's Job:**
- Several versions of the same code
- One for each:
  - Architecture/Optimization

Tools for PMU and Roofline

Tools for Roofline:

- Spiral : http://www.spiral.net/software/roofline.html
- Empirical Roofline Tool (ERT): https://crd.lbl.gov/departments/computer-science/PAR/research/roofline/software/ert/
- Intel Advisor: https://software.intel.com/en-us/advisor

Collect hardware events from PMU

- Linux Perf: https://perf.wiki.kernel.org/index.php/Main_Page
- likwid: http://www.nersc.gov/users/software/performance-and-debugging-tools/likwid/
- Intel Vtune: https://software.intel.com/en-us/vtune

Preliminary Evaluation

Workload

- A matrix multiplication (Intel Intrinsics);
- A numeric model in finance (AVX-512 Exponentials and Reciprocals);
- A N-Body simulation (Vectorized);
- A Diffusion simulation (Scalar).

Set of architectures

- Different Intel Xeon Generations:
  - Sandy Bridge (2013)
  - Haswell (2015)
  - Skylake (2017)

The strategy inputs:

- Application:
  - One or more optimized version for each code;

Statistical analysis of profiling:

- Vtune
  - HPC characterization
  - Memory Access
  - Memory Consumption
- Advisor
  - Roofline

Execution Time

	Execution Time (Seconds)		
Application	Skylake	Haswell	SandBridge
Finance	445,77	962,19	3.443,56
nbody_v1	172,15	544,52	1.637,79
nbody_v2	43,36	76,05	222,89
matrix mkl	27,3	48,7	152,71
matrix intrinsics	77,73	57,29	139,81
matrix multiplication	42,76	20,49	62,68
matrix v1	48,48	20,65	64,63
matrix v2	42,58	19,51	66,19
matrix v3	32,8	37,43	55,39
matrix v4	33,07	36,78	58,73
diffusion v1	64,53	40,75	28,6
diffusion v2	60,54	30,51	24,67

Compiler parametrization

Compiler presents several parametrization that influences performance

- Ex: -zmm: **high** X **low** (for Skylake)
  - Defines a level of zmm registers usage (AVX-512).
  - The **low** setting causes the compiler to generate code with zmm registers very carefully, only when the gain from their usage is proven.
  - The **high** setting causes the compiler to use much less restrictive heuristics for zmm code generation.
  - Which option (**high** X **low)** provides best results for each application?

Compiler parametrization

Applications	(Speedup / slowdown) low againt high	Gflops (%)	AI	FP Arith/Mem Wr Instr. Ratio	FP Arith/Mem Rd Instr. Ratio	L1 Bound	L2 Bound	L3 Bound
Finance	1,63	62,09	0,08	9,24	1,19	5,8	0	0
diffusion v1	1,17	8,32	0,00	0,12	0,06	0,4	-0,7	-0,5
diffusion v2	1,24	13,05	0,00	-0,05	0,04	-0,4	-1,3	1,6
matrix v3	1,60	8,23	0,00	0,05	0,30	7,4	-0,3	7,7
matrix v4	1,56	21,54	0,00	0,04	0,21	7,4	-0,4	10,3
matrix mkl	0,96	10,02	0,00	-6,25	-0,12	0,3	0,4	0,4
nbody_v2	0,91	60,00	-0,16	1026,53	2,57	-1,8	-0,9	-0,8
matrix v1	0,88	-30,69	0,00	2,63	-0,13	0,5	0	0,8
matrix v2	0,93	-18,81	0,00	2,10	-0,16	1,3	0	2,4
matrix intrinsics	0,99	-3,80	0	0	0	0	0	0

Rank Architectures

Rank application performance across different architectures

- Exploring correlation between:
  - **GFLOPS**

    indicates how much "work" the application is performing per second (overhead, hazards, latency...)

  - **Instructions per cycle (IPC)**

    Indicates the amount of instructions that can be executed per cycle (Metric that shows overhead and latency)

  - **AI (Arithmetic Intensity)**

    Shows how much data have to be transferred in order to execute the application

Stanzani S. et al. (2019) **Towards a Strategy for Performance Prediction on Heterogeneous Architectures.** In: Senger H. et al. (eds) High Performance Computing for Computational Science – VECPAR 2018.

Rank Architectures - Results

Numeric Model in Finance					
Architecture	Skylake	Haswell	SandyBridge	KNL (FlatMode)	KNL (Cache Mode)
Execution Time (Seconds)	458	1036	3443	235	224
Rank	3	2	1	4	5
Diffusion					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	511	309	220	425	1557
Rank	3	4	5	2	1
N-Body					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	306	467	1253	343	347
Rank	5	4	3	1	2
Matrix Multiplication (Intrisincs)					
Architecture	Skylake	Haswell	SandyBridge	KNL (Flat Mode)	KNL (Cache Mode)
Execution Time (Seconds)	172	159	344	132	227
Rank	4	5	1	3	2

Discussion

How to choose a representative workload

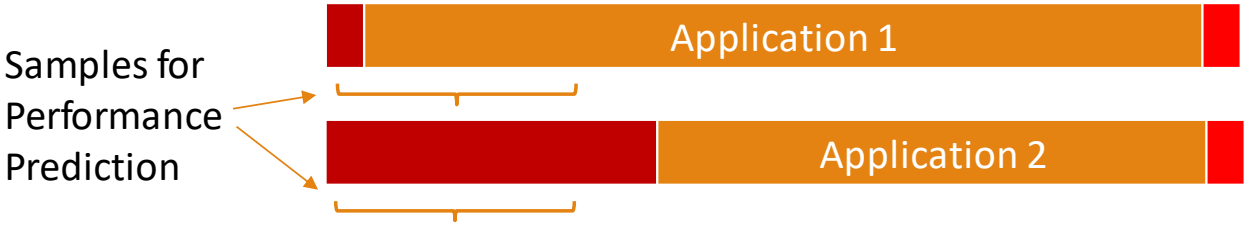
- Representative applications for HPC:
  - For example the "13 dwarfs"
- Representative range of performance patterns on each architecture

Asanovic, K. et al. ***The Landscape of Parallel Computing Research: A View from Berkeley***. UCB/EECS-2006-183, University of California, Berkeley, Dec. 18, 2006.

How to model performance of applications with difference phases

How to model impact of:

- Communication (MPI)
- Irregularities
- GPUs



Conclusion

PMU can be very useful to characterize the performance of HPC applications

Performance prediction model based on PMU has the potential to:

- Improve Data Center resources usage
- Support job scheduling

As future work:

- We will improve the dataset about application's execution
- We will investigate how machine learning techniques can help with performance prediction with this data

Thanks!

Paper, Slides and source code:

https://github.com/silviostanzani/PerformancePrediction

Questions?