Using the LLVM Interpreter to Quantify Inherent Application Properties

Victoria Caparrós (CS, ETH), Phillip Stanley-Marbell (Apple), and Markus Püschel (CS, ETH)

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Computer Science

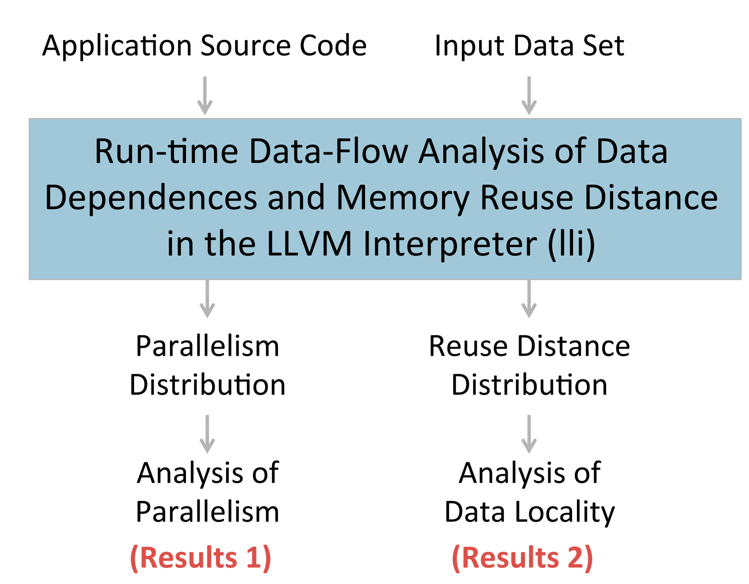
SPIRAL
www.spiral.net

Goal:
Characterization of Inherent Application Properties to Understand Performance

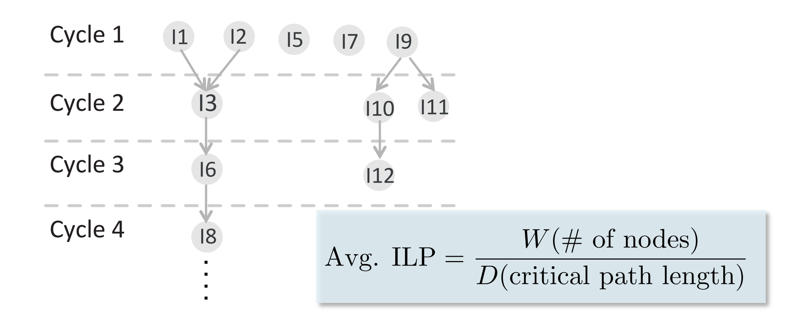
Architectures	Inherent Application Properties: Data/Task-Level Parallelism	Thread-Level Parallelism	Instruction-Level Parallelism	Data Locality
IBM Power7	8 cores	4 threads/core	12-wide superscalar	L1 32KB 8-way L2 256KB 8-way Shared L3 32MB
ARM CortexA9	1-4 cores	1 thread/core	3-wide superscalar	L1 64KB 4-way L2 8MB
Intel Core i7	4 cores	2 threads/core	4-wide superscalar	L1 32KB 8-way L2 256KB 8-way Shared L3 8MB
	Microarchitectural Features			

Research Questions:
- How well does an application match a platform?
- Diagnose performance bottlenecks
- Performance behavior upon platform upgrade?

Tool for Quantifying Applications Properties

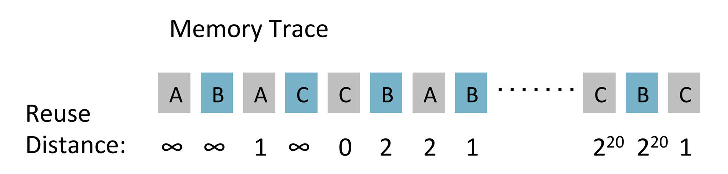


Analysis of Instruction-Level Parallelism (ILP)



$$\text{Avg. ILP} = \frac{W(\#\text{ of nodes})}{D(\text{critical path length})}$$

Analysis of Data Locality – Memory Reuse Distance



Reuse Distance = # of *distinct* intervening memory accesses

Why LLVM?

Language-, Architecture-Agnostic Intermediate Representation (LLVM IR)

MIPS instruction trace

```
lw     $5,8($30)
lw     $6,32($30)
mult   $5,$6
mflo   $5
lw     $6,4($30)
addu   $5,$5,$6
addu   $6,$0,$5
sll    $5,$6,0x30
lw     $6,40($30)
addu   $5,$5,$6
l.d    $f2,0($5)
mul.d  $f0,$f0,$f2
```

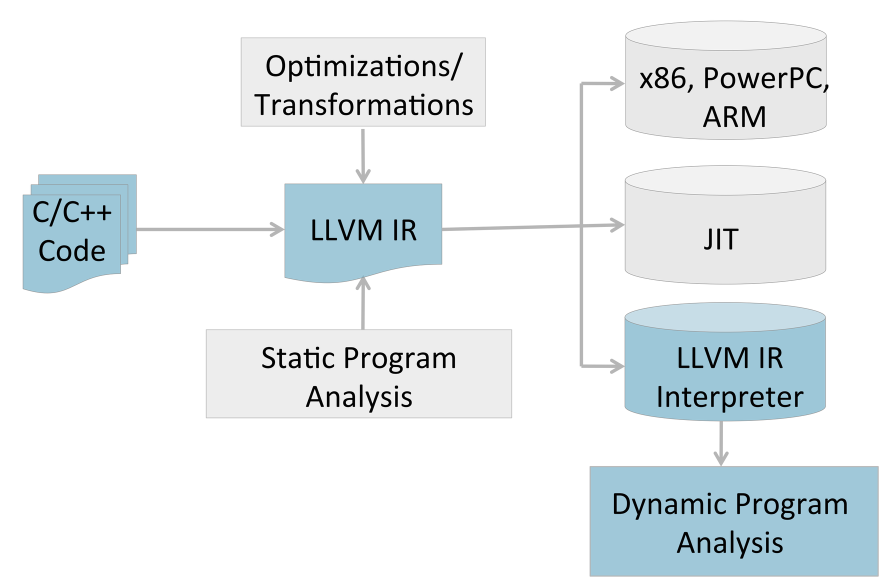
Memory address computation

IR instruction trace

```
%arrayidx16 = getelementptr inbounds double* %b, i64 1
%16 = load double* %arrayidx16, align
%mul = fmul double %0, %16
```

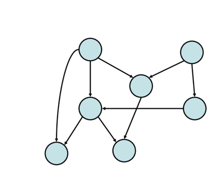
Inherent computation – Load and multiplication

Modular Design

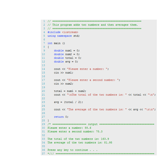


Existing approaches to characterizing application properties

Theoretical DAG analysis



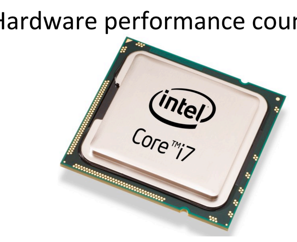
Static source code analysis



Analytical models

$$\frac{pC_0}{\beta}\left(1+\frac{\alpha\beta/L}{Q/D}\right) \le \frac{W}{QL}\left(1+\frac{p}{W/D}\right)$$

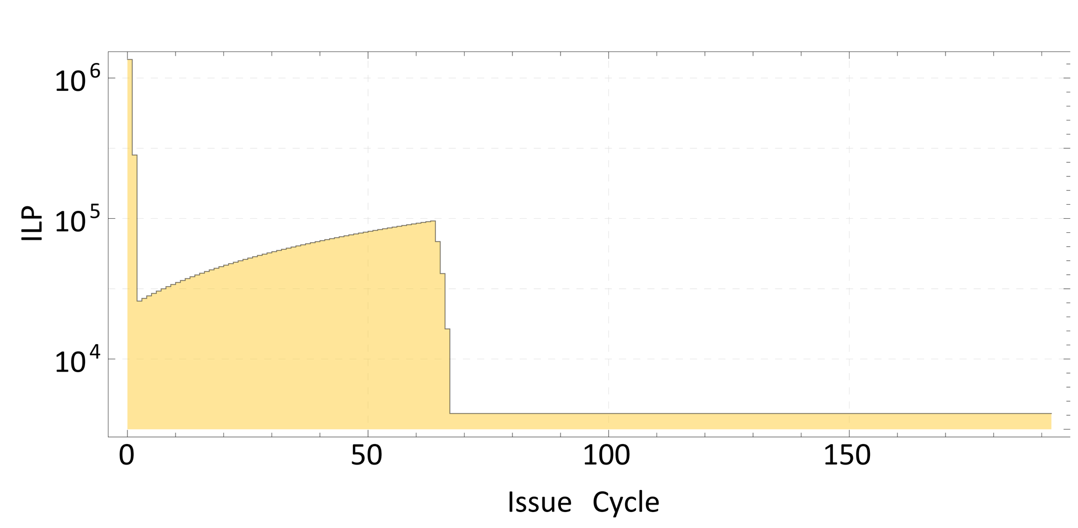
Hardware performance counters



Results 1:
Analysis of Parallelism

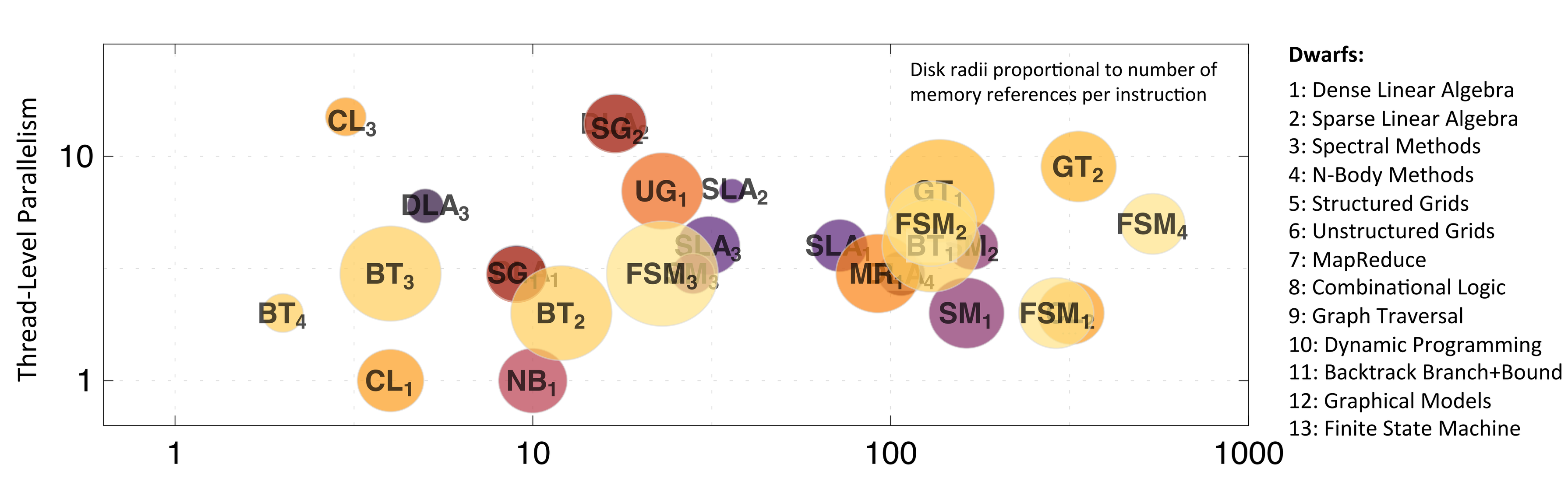
Ideal-Case Instruction-Level Parallelism

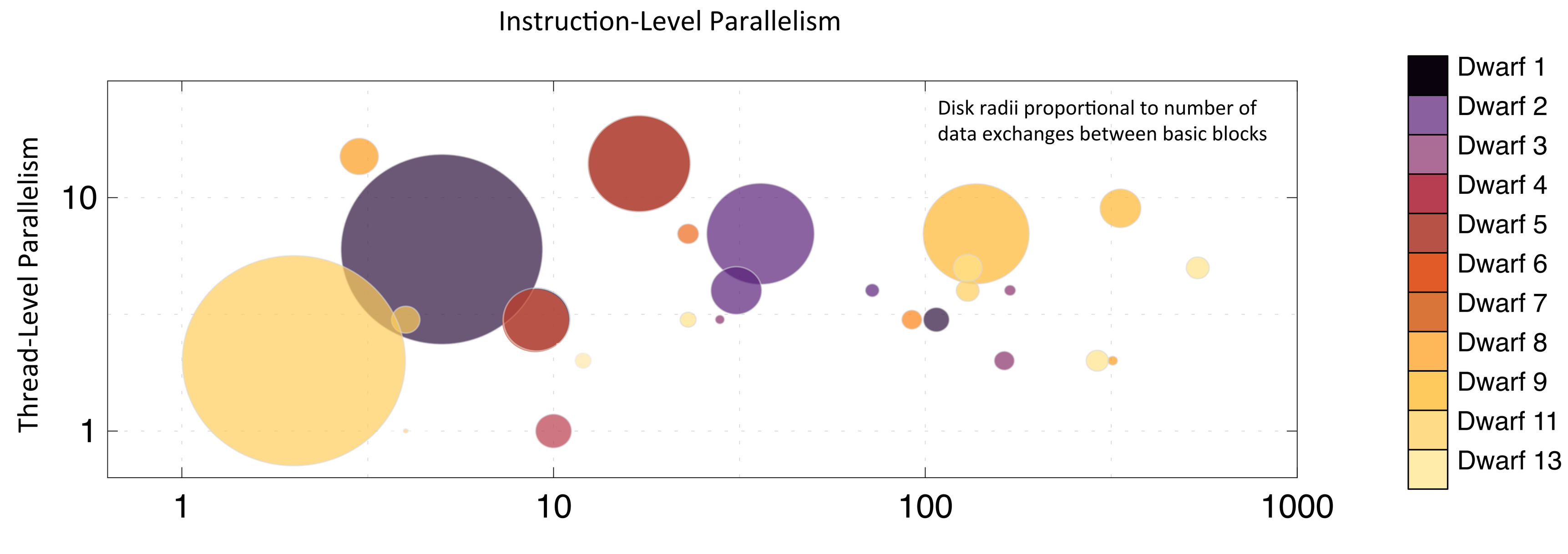
Distribution over execution cycles vs. average value



Characterization of ILP and TLP of 26 Applications and Kernels Classified According to their Constituent Berkeley Dwarf [Caparros, 2011]

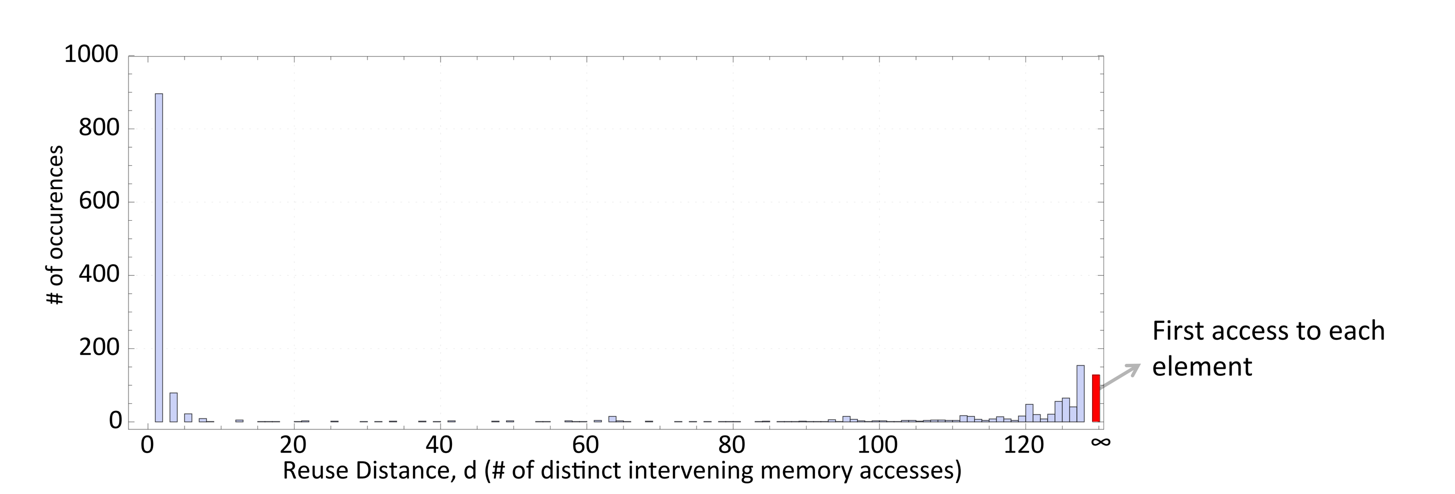
The Berkeley dwarf classification [K. Asanovic, 2009] is intended to categorize **computational motifs** according to their communication and computation patterns. There are 13 dwarfs, e.g., **Dwarf 1:Dense Linear Algebra (DLA)**



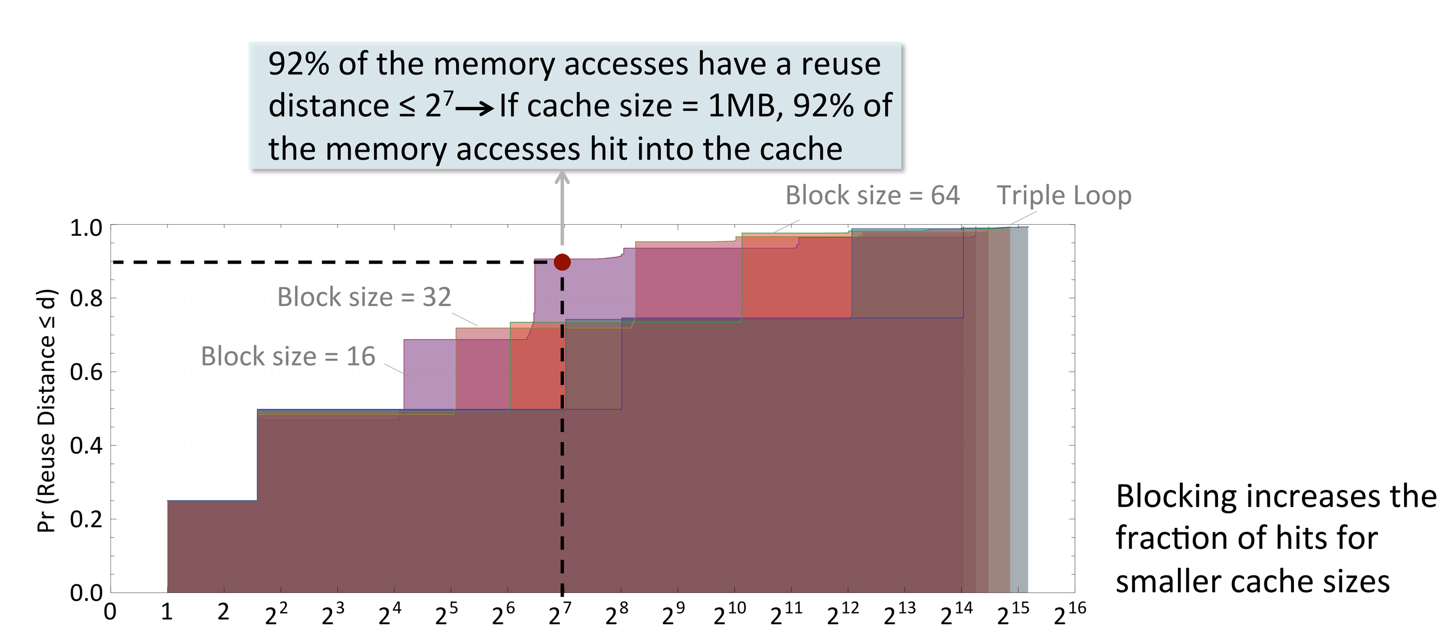


Results 2:
Analysis of Data Locality

Histogram of Reuse Distances (PDF)



Reuse Distance Cumulative Distribution (CDF) for Matrix-Matrix Multiplication, N=128



References

Limits of Instruction-Level Parallelism
David P. Wall. ASPLOS, 1991

Predicting whole-program locality through reuse distance analysis
C. Ding and Y. Zhong. PLDI, 2003.

Parallelism and Data Movement Chracterization of Contemporary Application Classes
Victoria Caparros and Phillip Stanley-Marbell. SPAA, 2011

The Landscape of Parallel Computing Research: A View from Berkeley
K. Asanovic *et al.* Communications of the ACM, 2009