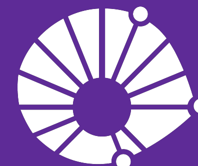
Using OpenMP to Detect and Speculate Dynamic DOALL Loops (IWOMP 2020)

Bruno Chinelato Honorio
João P. L. de Carvalho
Prof. Dr. Munir Skaf
Prof. Dr. Guido Araujo

University of Campinas (Unicamp)
Brazil



UNICAMP

DOALL Loop vs DOACROSS Loop

Loop Carried Dependences

Read After Write (RAW)

```
for(j = 1; j < n; j++)
   S1: a[j] = a[j-1];
```

Write after Write (WAW)

```
for(j = 0; j < n; j++)
   S1: c[j] = j;
   S2: c[j+1] = 5;
```

Write after Read (WAR)

```
for(j = 0; j < n; j++)
   S1: b[j] = b[j+1];
```

Loop Carried Dependences

Read After W

```
for(j = 1;
    S1: a[j]
```

rite (WAW)

```
< n; j++)
 j;
= 5;
```

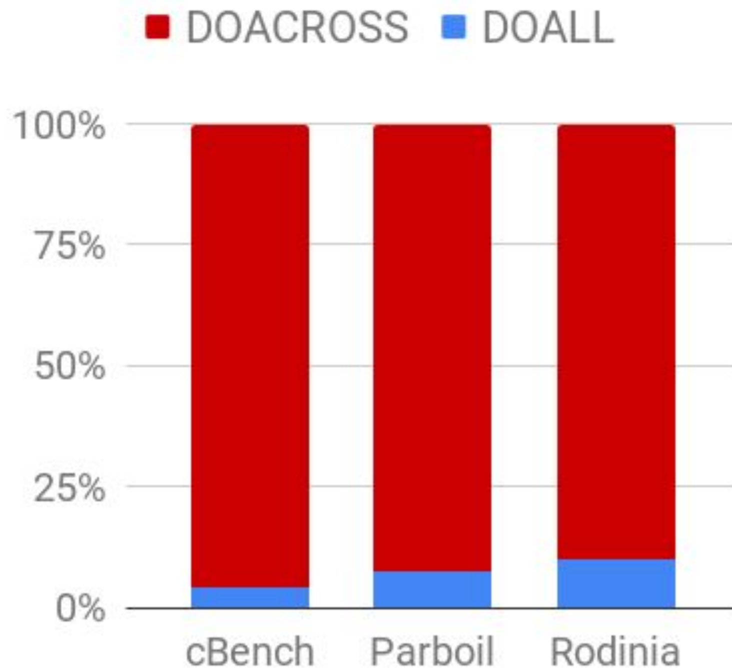
Bad Loops For Parallelization!

Motivation

- DOALL and DOACROSS loops can be detected
- Why Dynamic Detection and Not Static?
- 180 loops (>10% TOTAL CPU TIME)
- 45 benchmarks
- 3 benchmark suites (cBench, Parboil, Rodinia)

Motivation

ICC Vectorization report flags:
-qopt-report5 and
-qopt-report-phase=vec



GOAL:
Extend OpenMP to Enable Runtime Loop Analysis

Runtime Loop Analysis

- The main goal is to discover loops that can be parallelized but compilers could not determine statically that they were free of dependences.
  - Looked at the DOACROSS Loops.
  - If the Runtime Analysis says a loop has no LCD, we call it Dynamic Doall (D-DOALL). Otherwise, we call it D-DOAX.
- Runtime analysis can be more accurate and offer more loop information. Overhead is high, taking more time and memory consumption.
- Dependence Report is limited to the input used.

Runtime Loop Analysis

- The main goal is to discover loops t… …uld not determine statically that they were…
  - Looked at the DOACROSS Loops.
  - If the Runtime Analysis says a loop has… …wise, we call it D-DOAX.
- Runtime analysis can be more accu… …verhead is high, taking more time and memory consumption.
- Dependence Report is limited to the input used.

Speculate Loops

- Hardware based speculation (HTM - Hardware Transactional Memory)
- Start, commit or abort transactions.
- Aborts happen when data conflicts happen or hardware capacity resources are exhausted.

Runtime Loop Analysis

- The main goal is to discover loops that can be parallelized but compilers could not determine statically that they were free of dependences.
  - Looked at the DOACROSS Loops.
  - If the Runtime Analysis says a loop has no LCD, we call it Dynamic Doall (D-DOALL). Otherwise, we call it D-DOAX.
- Runtime analysis can be more accurate and offer more loop information. Overhead is high, taking more time and memory consumption.
- Dependence Report is limited to the input used.
- D-DOAX loops could be speculated too!

Metrics

- Can a loop be speculated?
- Does it has enough iterations?
- If it is DOACROSS, how many dependences?
- What is the frequency of these dependences?

Metrics

Metric	Description
Number of Visits	The number of times a loop was visited and fully executed.
Total Number of Iterations	Average number of iterations a single loop visit has.
Innermost Loop Indicator	Indicates if a loop is the innermost in a loop nest.
First Eviction Iteration (FEI)	Indicates in which iteration of a loop the first cache eviction happens.
Total Loop-Carried Dependences (LCD)	The total sum of unique loop-carried dependences (LCD) of a loop.
Total Loop-Carried Probability (LCP)	Total probability of a LCD appearing in the loop.

The Check Clause

- Implemented on LLVM compiler framework
- Using libtooling for source-to-source transformations

The Check Clause

Syntax:

```
#pragma omp parallel check (attributes)
Loop
  Loop-body
```

The Check Clause Attributes

Attribute	Operation (What it Reports)
Time	File Name, Line Number, CPU Time, Iterations and Visits
Dependence	LCD, LCP, FEI, INNER
First	Detects if loop has at least one LCD or not
Verbose	Visual representation of loop dependence

check (verbose)

```
1  #pragma omp parallel \
2   check(verbose)
3  for(i=0; i < N; ++i){
4   const int value = img[i];
5   if(histo[value] < UINT8_MAX)
6   {
7    ++histo[value];
8   }
9  }
```

```
1
2  for(i=0; i < N; ++i){
3      .
4      .
5  +--> if (histo[value]...
6  |   .
7  |   .
8  +<-- >--< ++histo[value];
9      .
```

Heuristics

- How to decide when a loop is parallelizable or not?

Heuristics

Metric	Threshold
Visits	Lower or equal to 1000
ITER	Higher or equal to 2
LCD	If LCP is higher than 30%, LCD is at most 15; Else, LCD is at most 30.
FEI (condition 1)	If FEI>1, Loop is parallelizable
FEI (condition 2)	If FEI = 1, Search through perfect nested loops until a loop that satifies these two conditions is found: ● Visits is lower or equal to 1 Million ● FEI>1

Experimental Results

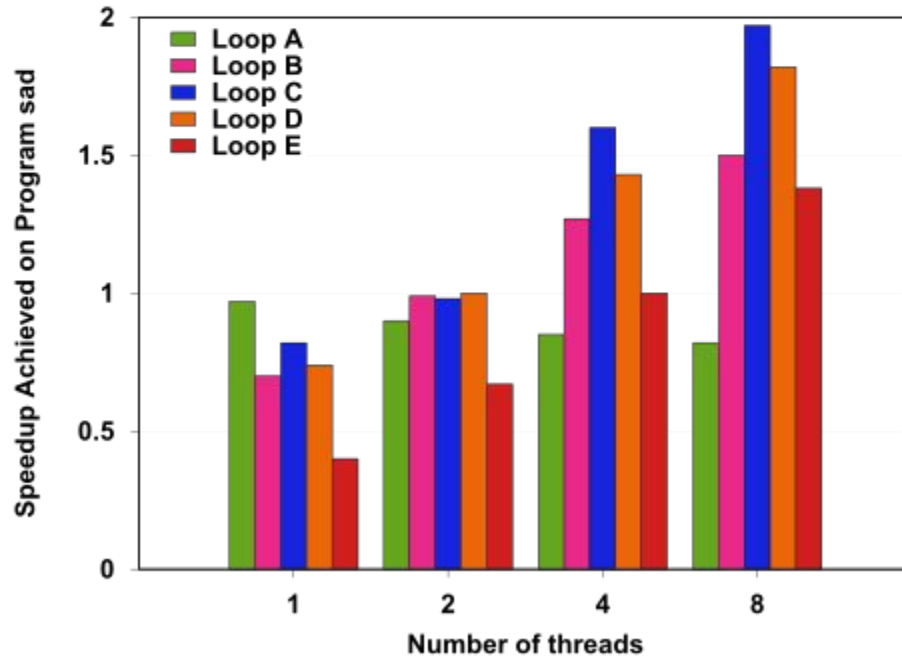
Do the Heuristics Work?

Loops		CPU Time					Metrics						
ID	Benchmark	Filename	Line	%	Total(s)	Mean(s)	Type	Visits	INNER	ITER	FEI	LCP	LCD
A	sad	sad_cpu.c	39	96.88	52.29	7.80e-01	D-DOALL	67	NO	120.0	1	-	-
B	sad	sad_cpu.c	69	96.88	52.29	6.50e-03	D-DOAX	8040	NO	33.0	20	96.70	13
C	sad	sad_cpu.c	70	96.86	52.28	1.97e-04	D-DOALL	265320	NO	33.0	-	-	-
D	sad	sad_cpu.c	74	96.29	51.96	5.93e-06	D-DOAX	8755560	NO	4.0	-	75.0	10
E	sad	sad_cpu.c	75	93.40	50.41	1.44e-06	D-DOALL	35022240	NO	4.0	-	-	-

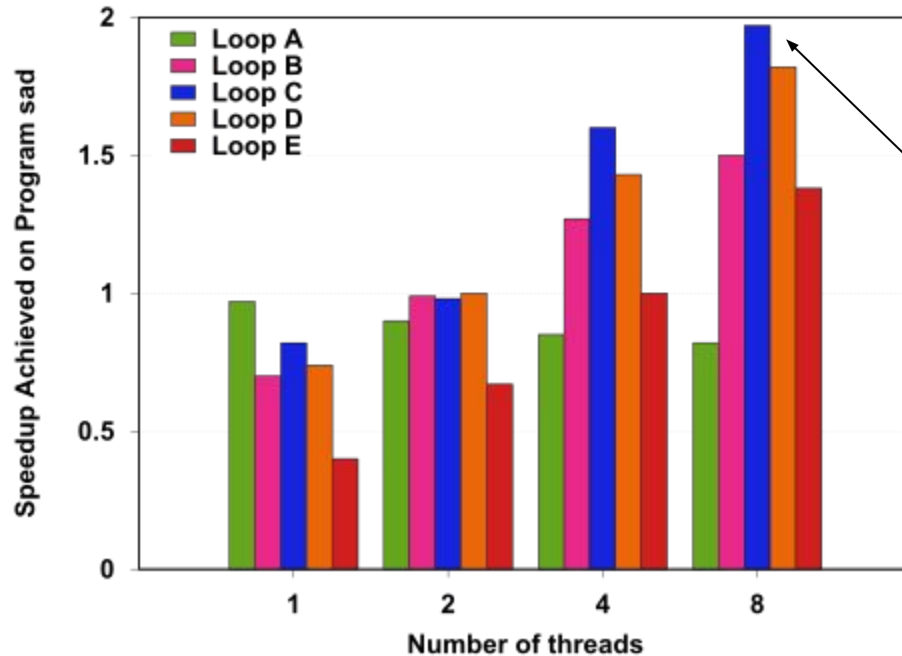
Do the Heuristics Work?

Loops		CPU Time					Metrics						
ID	Benchmark	Filename	Line	%	Total(s)	Mean(s)	Type	Visits	INNER	ITER	FEI	LCP	LCD
A	sad	sad_cpu.c	39	96.88	52.29	7.80e-01	D-DOALL	67	NO	120.0	1	-	-
B	sad	sad_cpu.c	69	96.88	52.29	6.50e-03	D-DOAX	8040	NO	33.0	20	96.70	13
C	sad	sad_cpu.c	70	96.86	52.28	1.97e-04	D-DOALL	265320	NO	33.0	-	-	-
D	sad	sad_cpu.c	74	96.29	51.96	5.93e-06	D-DOAX	8755560	NO	4.0	-	75.0	10
E	sad	sad_cpu.c	75	93.40	50.41	1.44e-06	D-DOALL	35022240	NO	4.0	-	-	-

Do the Heuristics Work?

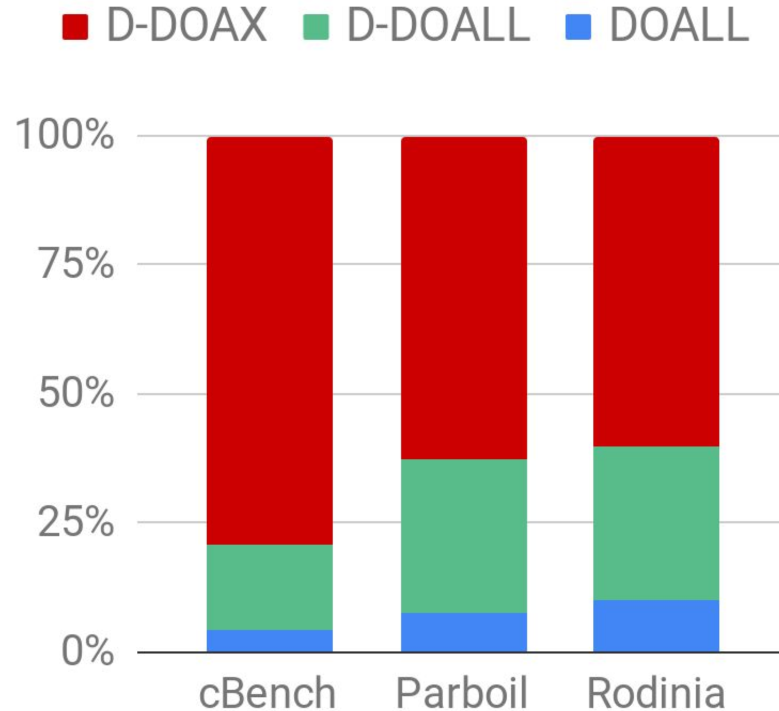


Do the Heuristics Work?

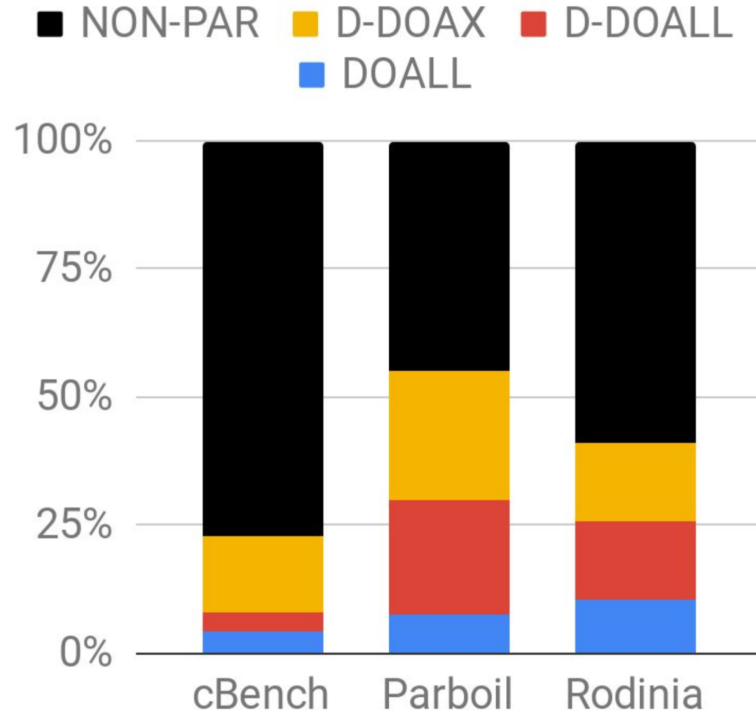


1.92x Speedup!

Breakdown without Heuristics



Breakdown With Heuristics



Conclusion

- 36% of loops with parallelization opportunities are missed by compilers. (53 out of 167)
- Compilers only manage to determine 7.8% of the loops to be DOALL. (13 out of 180)
- Future work is to exploit these opportunities.

Thank you!