


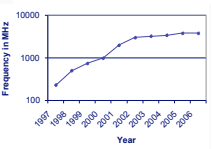
Hybrid Analysis and its Application to Thread Level Parallelization

Lawrence Rauchwerger
Joint work with Silviu Rus

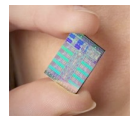


Computing Paradigm Shift


Past: Frequency Scaling



Future: Multi Cores



200+ GFLOPS



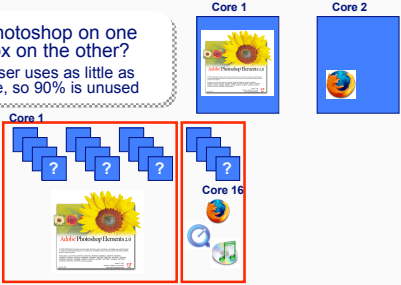
We are here

Leakage current increases
- Heat
- Excessive power consumption
Memory latency gap widens
- Marginal improvement in performance

Parallelism within a Single Application

Why not run Photoshop on one core and Firefox on the other?
- The web browser uses as little as 10% of its core, so 90% is unused

What will we do with 16 cores?



Goal: Single application scalable parallelism
Solution: Thread level parallelization of loops

Compilation for Multi-Cores: Thread Level Parallelization

- Manual**
 - Very expensive for sequential legacy code: MPI - OpenMP
- Automatic**
 - Not very successful so far: Compilers
 - Weak static analysis
 - Decisions depend on dynamic values
- Hybrid Analysis** bridges static & dynamic analysis
 - General analysis method applicable to various optimizations
 - Achieved efficient automatic parallelization

Automatic Parallelization: Data Dependence Analysis

Write Read	$x = 5$ $x = 8$ $\dots = x$ flow	Thread 1 $x = 8$	Thread 2 $\dots = x$	Cannot be done in parallel
Read Write	$x = 5$ $\dots = x$ $x = 8$ anti	Thread 1 $\dots = x$	Thread 2 $x = 8$	Can be done in parallel Privatization $y = 5$ $x = 8$
Write Write	$x = 5$ $x = 8$ $\dots = x$ output	Thread 1 $x = 5$	Thread 2 $x = 8$	Privatization $x = 5$ $y = 8$ $\dots = y$

Dependence = order } Parallel = independent

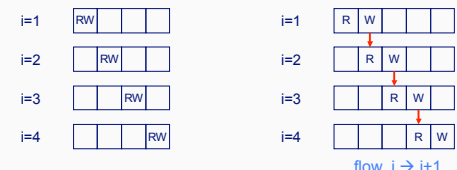
Loop Carried Data Dependences

Parallelizable loop

```
DO i = 1, 100
  a[i] = a[i] + b[i]
ENDDO
```

Sequential loop

```
DO i = 1, 100
  a[i+1] = a[i] + b[i]
ENDDO
```



flow, $i \rightarrow i+1$

Analytical Approach: Analyze array subscript expressions

Static Data Dependence Analysis: Linear Reference Patterns

```

DO j = 1, 10
  a(j) = a(j+40)
ENDDO

```

No cross iteration dependencies \longleftrightarrow No integer solutions:

$$\begin{cases} 1 \leq j_w \leq 10 \\ 1 \leq j_r \leq 10 \\ j_w = j_r \\ j_w = j_r + 40 \end{cases}$$

- Geometric view: Polytope model
 - Some convex body contains no integral points
- Simplified solutions: GCD Test, Banerjee Test etc
 - Potentially overly conservative
- General solution: Presburger formula decidability
 - Omega Test: Precise, potentially slow

Restricted to linear addressing and control: mostly small kernels 7

Alternative: Run-time Analysis

```

READ *, N
DO j=1, N
  a(j) = a(j+40)
ENDDO

```

Linear, very simple, but not decidable statically!

Solution: LRPD Run-time Test (Rauchwerger and Padua '95)

Instrument all relevant memory references
Analyze the resulting trace at run time

Accurate, but the overhead is proportional to the dynamic memory reference count
Minimum necessary information for parallelization: $N \leq 40$ 8

Hybrid Analysis of Memory Reference Patterns

	Compile-time Analysis	Hybrid Analysis	Run-time Analysis
STATIC	Symbolic analysis	Symbolic analysis Extract conditions	
DYNAMIC		Evaluate conditions	Full reference-by-reference analysis

Framework: Hybrid memory reference analysis
Application: Automatic parallelization 9

Hybrid Analysis

```

DO j=1, N
  a(j) = a(j+40)
ENDDO

```

Under what conditions can the loop be executed in parallel?

1. Collect and classify memory references.
2. Aggregate them symbolically.
3. Formulate independence test.

4.a) If we can prove $10 \leq N \leq 30$, Declare loop parallel.
4.b) If N is unknown, Extract run-time test. $N \leq 40$

Hybrid Analysis

```

DO j=1, N
  a(j) = a(j+40)
ENDDO

```

Execute the loop in parallel if possible.

4.a) If we can prove $10 \leq N \leq 30$, Declare loop parallel.
4.b) If N is unknown, Extract run-time test. $N \leq 40$

Compile Time

Run Time

Parallel Loop

```

DO PARALLEL j=1, N
  a(j) = a(j+40)
ENDDO

```

No run-time tests performed if not necessary!

Parallel Loop

```

IF (N <= 40) THEN
  DO PARALLEL j=1, N
    a(j) = a(j+40)
  ENDDO
ELSE
  DO j=1, N
    a(j) = a(j+40)
  ENDDO
ENDIF

```

Sequential Loop

Hybrid Analysis Example

```

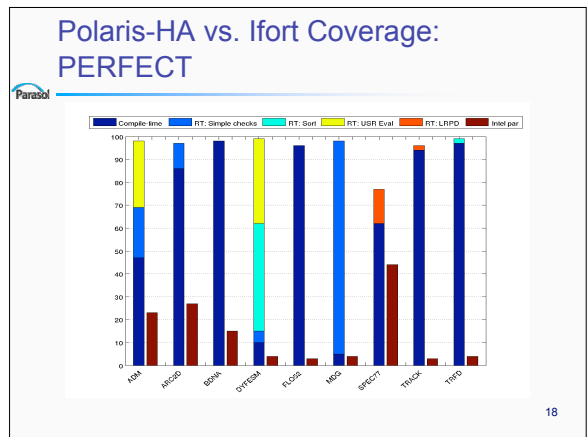
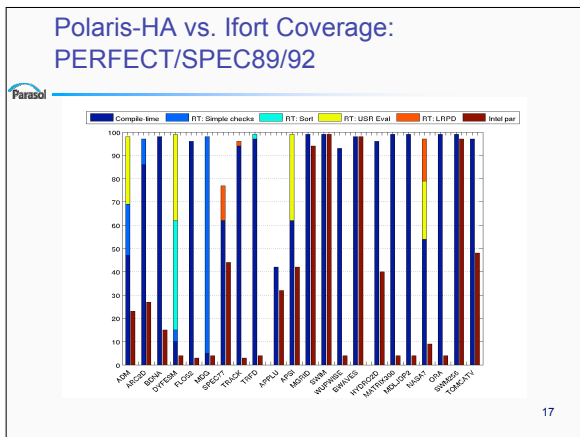
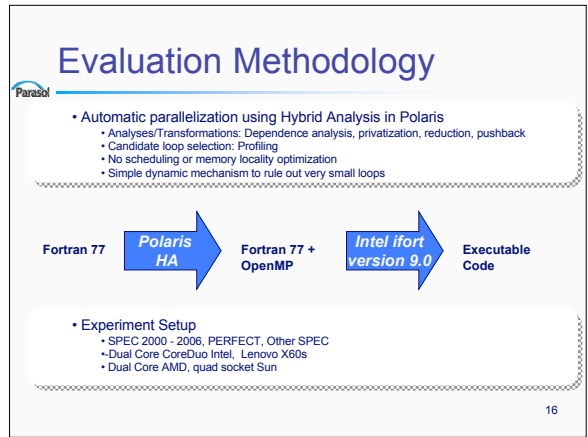
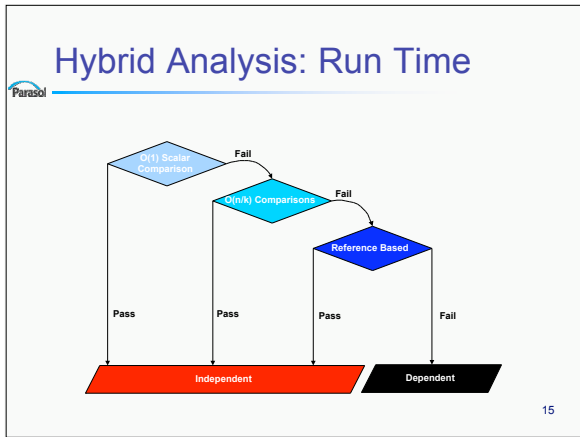
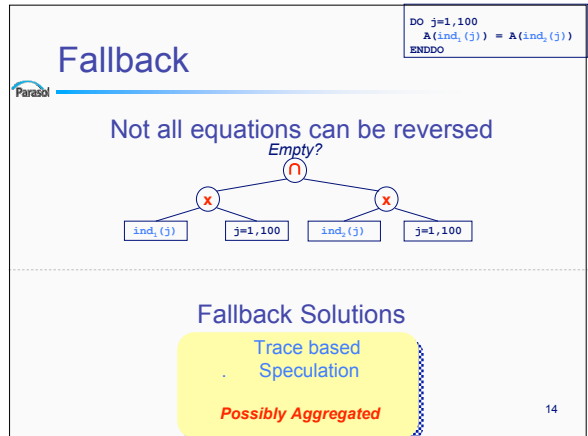
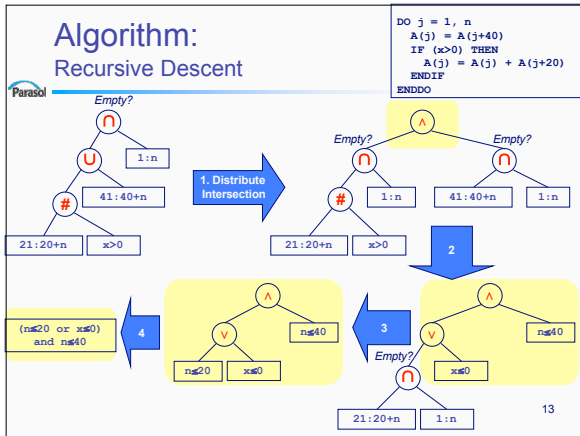
DO j = 1, n
  A(j) = A(j+40)
  IF (x>0) THEN
    A(j) = A(j) + A(j+20)
  ENDDIF
ENDDO

```

$READ \cap WRITE = Empty?$

Distribute Intersection

12



Conclusions: HA + Autopar

- Hybrid memory reference analysis is a general framework for optimization
 - USR
 - Closed-form representation that tolerates analysis failure
 - PDAG:
 - Input sensitivity of optimization decisions
 - Continuum of compile-time to run-time solutions
- Efficient automatic parallelization
 - Speedups on FP benchmark applications

<http://parasol.tamu.edu/compilers/ha>