1) Introduction & Motivation

• Moving towards Exa-scale and Extreme-scale machines
  - Enabling applications to fully exploit them is not easy

• Software stack
  - Explicitly parallel programming models
  - Optimizing compilers
  - Efficient runtime systems

• Our past work (PoPP) [PACT’15, IMPACT’15]
  - Automatically optimize OpenMP programs
  - Task-parallel, Loop-parallel
  - Polyhedral compilation techniques
  - Fine-grained synchronization in run-time

• Limitations of PoPP:
  - Applicable only to set of parallel constructs that satisfy serial-elasticity property

• Motivation:
  - Analyze and Transform more generic OpenMP constructs such as parallel regions, work-sharing, barriers, sections etc. — PolyOMP

2) Extending Polyhedral Representation

• Polyhedral representation of a statement instance
  - Domain + Schedule + Access relations

• Domain - Set of statement instances

• Schedule - Ordering among statement instances
  - Extended with Phase, Space dimensions
  - Phase - Computation phase id
  - Space - Thread id that executes it

• Access relations - Array subscripts referenced

3) Analysis - Data race Detection

• Step-1: Conditions for Data-race b/w statement instances S&T
  - S & T should touch the same memory location and one being a write
  - S & T should be in same phase of computation
  - S & T should be executed by different threads

• Step-2: Use Z3 solver for the existence of the solutions

• Implementation details
  - Frameworks:
    • PET (Polyhedral extraction tool), Z3 (SMT solver)
  - Implementation is in-progress

• Preliminary Experimental Evaluation
  - Manual evaluation on OMP SRC benchmarks
  - Data races category
    - Accessing overlapped memories
    - Inserting barrier statements
    - Conditional executions among threads.
  - All races were detected and Z3 solver took less time (1-2 seconds) for the existence of the solutions

4) Transformations

• Data dependence relations from S to T
  - S & T should touch same memory location and one of them is write
  - S should happens-before T (Computed based on phase and space)

• Transformations
  - Fusion of work-sharing directives
  - Removal of redundant barriers
  - Fusion of SPMD regions

• Example

```c
#pragma omp parallel {
#pragma omp for schedule(static, c) nowait
#pragma omp for schedule(static, c) 
for(int j = 0; j < N; j++)
# pragma omp for schedule(static, c)
A[i] = B[i]; // S1
for(int i = 0; i < N; i++)
# pragma omp for schedule(static, c)
C[j] = A[j]; // S2
}
```

4) Transformations

- **PolyOMP (Ours)**
  - Automated parallelization
  - Optimized with Z3 solver

<table>
<thead>
<tr>
<th>Target parallelism</th>
<th>Pathg (LCTEES’12)</th>
<th>OAT (ICPP’13)</th>
<th>ompVerify (IWOMP’11)</th>
<th>PolyX10 (PPOPP’13)</th>
<th>PolyOMP (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMP Work sharing (loop), Barriers, Atomic</td>
<td>OpenMP Work sharing (loop, sections), locks, barriers, master, single, critical, atomic</td>
<td>OpenMP Work sharing (loop)</td>
<td>X10 Asynco-parallelism</td>
<td>OpenMP SPMD regions, work sharing (loop, sections), master, single, barrier, doacross (OpenMP 4.1)</td>
<td>PolyOMP (Ours)</td>
</tr>
<tr>
<td>Approach</td>
<td>Extended Thread Automata</td>
<td>Symbolic execution</td>
<td>Polyhedral model</td>
<td>Polyhedral model</td>
<td>Extended Polyhedral model</td>
</tr>
<tr>
<td>Guarantees</td>
<td>Per-no.of threads, chunk size</td>
<td>Per-no. of threads</td>
<td>Per-Program</td>
<td>Per-Program</td>
<td>Per-Program</td>
</tr>
<tr>
<td>Dependent on scheduling techniques</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---

- **PoPP (PACT’15)**
- **PIR-OPT (TOPLAS’13)**
- **OMPd (PPoPP’12)**
- **PolyOMP (Ours)**

- **SPMD regions, work sharing (loop, sections), master, single, barrier, doacross (OpenMP 4.1)**
- **Ordering relations in data dependences**
  - Intersection of conservative dependences with HB relations
  - Dataflow analysis
- **HB relations**
- **Optimizations**
  - Loop-level transformations
  - Finish-elimination, forall-coarsening, loop-chunking
- **Communication reduction**

---

- **Acknowledgements**
  - Jun Shirako and Habanero Extreme Scale Software Research Project

Contact: prasanth@rice.edu

http://habanero.rice.edu