Lightweight Fault Detection in Parallelized Programs

Li Tan
UC Riverside

Min Feng
NEC Labs

Rajiv Gupta
UC Riverside

CGO'13, Shenzhen, China
Feb. 25, 2013

UCR
UNIVERSITY OF CALIFORNIA, RIVERSIDE

Program Parallelization

- Parallelism can be achieved via parallelization of sequential programs via easy-to-use *parallel constructs* → OpenMP, SpiceC [PPoPP'11], and TBB.

- Data Dependence Related Concurrency Bugs
  - Data races
  - Atomicity violations

Comparison Checking

- Conventional [PACT'98] [FSE'99]
  - Locate program faults by leveraging the availability of *two versions* of a program – one supposed *correct* version and one *derived* version
- In Our Scenario
  - A sequential version ***S*** and a parallelized version ***P***
  - Faulty parallelization → data dependence violation
    - Data dependences enforced by *S* are not preserved by *P*

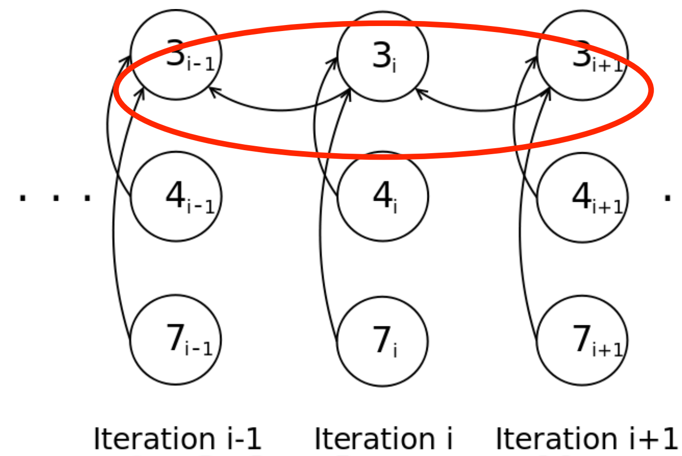
Debugging *Parallelized* Programs

Basic Idea:

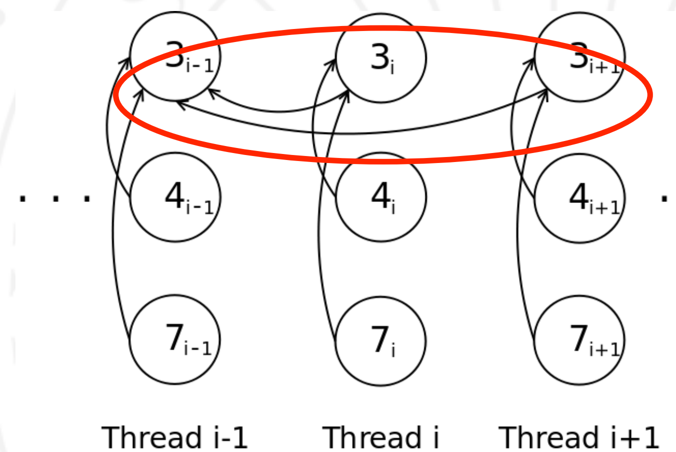
- Comparison check the data dependences exercised by the executions of ***S*** and ***P***
  - dynamic *D*ata *D*ependence *G*raphs:*sDDG*+*pDDG*
    - Nodes: execution instances of statements
    - Edges: data dependences between nodes
  - Faulty parallelization: *different* sDDG and pDDG constructed using the *same* input

Data Race Detection

```
1: #pragma omp parallel for
2: for (;;) {
       #pragma omp critical
3:     if (fgets(s, NUM, stdin) == NULL) break;
4:     if (s[0] == '!') {
5:         /* Other relevant code here */
6:     }
7:     if (!separate_sentence(s)) {
8:         /* Other relevant code here */
9:     }
10: }
```



sDDG



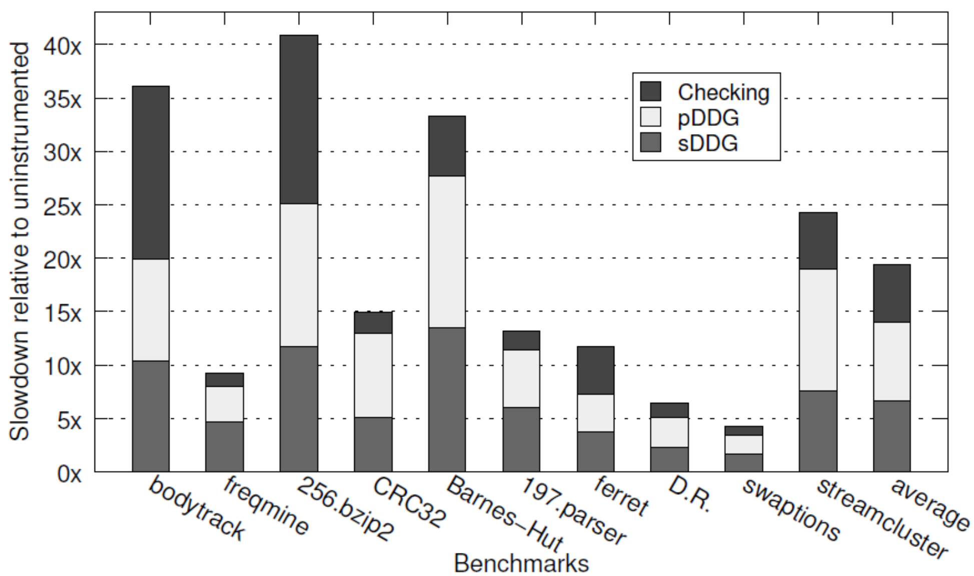
pDDG

UNIVERSITY OF CALIFORNIA, RIVERSIDE

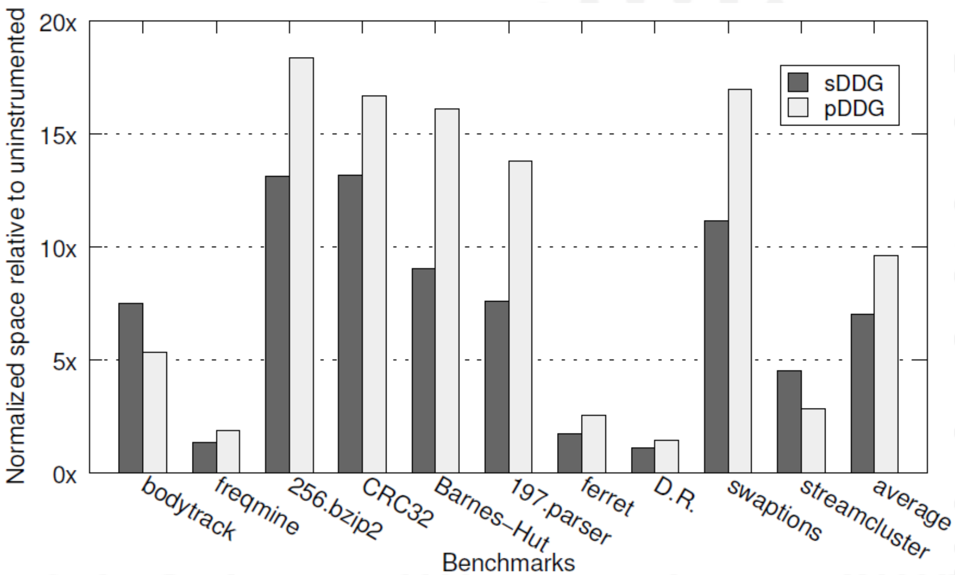
UC RIVERSIDE UNIVERSITY OF CALIFORNIA

Limitations

- DDG Construction Overhead



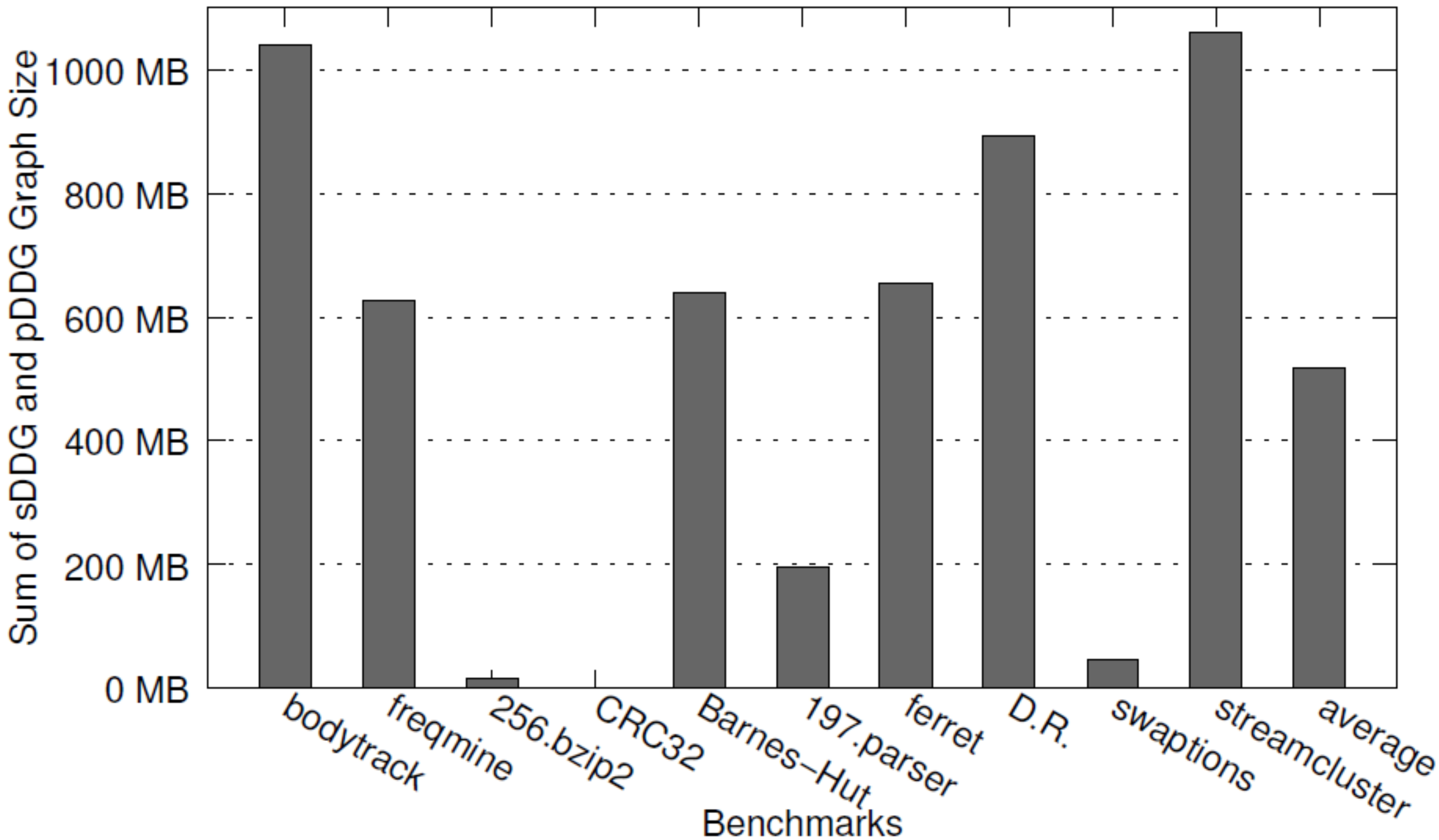
Execution Time



Memory Space

Limitations (Cont.)

- Graph Size and Checking Time



Limitations (Cont.)

- Dependence Violation May Not Occur
  - *Not every* interleaving causes violation
  - As low as 10% chance to expose a data race; up to 22 hours to expose an atomicity violation [ASPLOS'09]
- Validity of Comparing Two Runs
  - *Random numbers* alter control flow [ISSTA'07]
  - Inconsistency ≠ a concurrency bug

Significant limitations…
How can we get rid of them?

OPT-1: Region Graphs

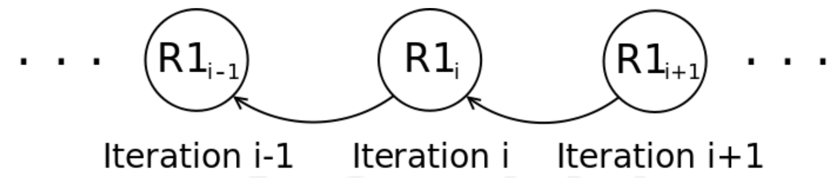
- Eliminate *irrelevant* dependences
  - Data dependences in sequentially executed code
  - ***Savings***: time + space for tracking and checking

- *fine*-grained graphs → *coarse*–grained graphs
  - Statements (DDG) → Code regions (DRG)
  - ***Savings***: graph size

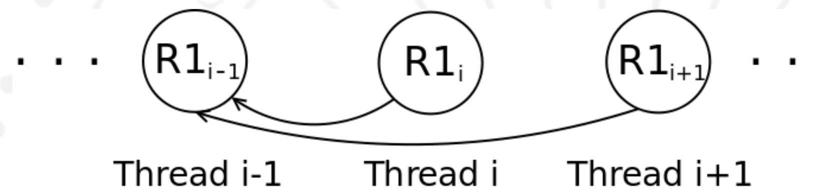
OPT-1: Region Graphs (Cont.)

```
#pragma omp parallel for
for (;;) {
    if (fgets(s, NUM, stdin) == NULL) break;
    if (s[0] == '!') {
        /* Other relevant code here */
    }
    if (!separate_sentence(s)) {
        /* Other relevant code here */
    }
}
```

Region *R1*



sDRG



pDRG

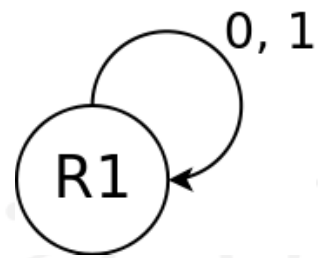
/ the same data race example... */*

OPT-2: Summarize Region Instances

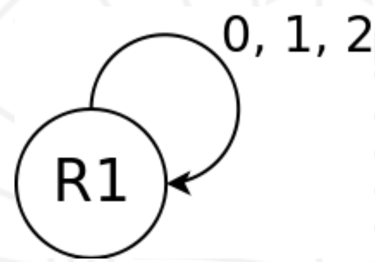
- A single node in a *DRG* represents all execution instances of a region
  - Different dependences need to be distinguished
  - ***Savings***: time for tracking and checking + graph size
- Annotate each edge by *dependence distances*
  - 0 indicates an *intra-iteration* dependence
  - A non-zero value indicates a *cross-iteration* dependence

OPT-2: Summarize Region Instances

```
#pragma omp parallel for
for (;;) {
    if (fgets(s, NUM, stdin) == NULL) break;
    if (s[0] == '!') {
        /* Other relevant code here */
    }
    if (!separate_sentence(s)) {
        /* Other relevant code here */
    }
}
```



sDRG '



pDRG '

/ the same data race example... */*

OPT-3: Static Region Graph

- Only the sequential version needs to be run
  - *Statically analyzing* the parallel constructs in the parallelized version
  - ***Savings***: time + space for tracking and checking

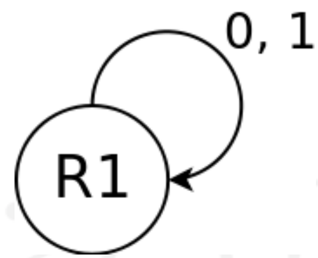
- Simplified concurrency bug detection
  - Check if data dependences *allowed* by OpenMP, SpiceC, and TBB *violate* sequential semantics
  - ***Eliminate the limitations***:
    - Reproducibility rate + validity of comparing two runs

OPT-3: Static Region Graph (Cont.)

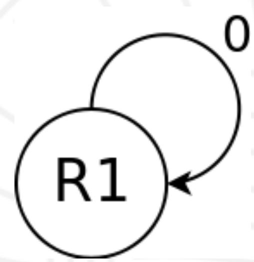
Construct	Allowed Dependences
OpenMP	
parallel [for|do]	Intra-iteration dependences
section	Intra-iteration dependences
critical	Intra-iteration/cross-iteration dependences
ordered	Intra-iteration/cross-iteration dependences
SpiceC	
doall	Intra-iteration dependences
doacross	Intra-iteration dependences
pipelining	Intra-iteration/cross-iteration dependences
after(ITER-x, R_y)	Intra-iteration/cross-iteration dependences from region R_y to current region with a distance x
atomicity_check	Intra-iteration/cross-iteration dependences
TBB	
parallel_for	Intra-iteration dependences
parallel_reduce	Intra-iteration dependences and cross-iteration dependences of join
parallel_scan	Intra-iteration dependences and cross-iteration dependences of reverse_join
parallel_pipeline	Intra-iteration dependences and cross-iteration dependences of filter

OPT-3: Static Region Graph (Cont.)

```
#pragma omp parallel for
for (;;) {
    if (fgets(s, NUM, stdin) == NULL) break;
    if (s[0] == '!') {
        /* Other relevant code here */
    }
    if (!separate_sentence(s)) {
        /* Other relevant code here */
    }
}
```



sDRG '



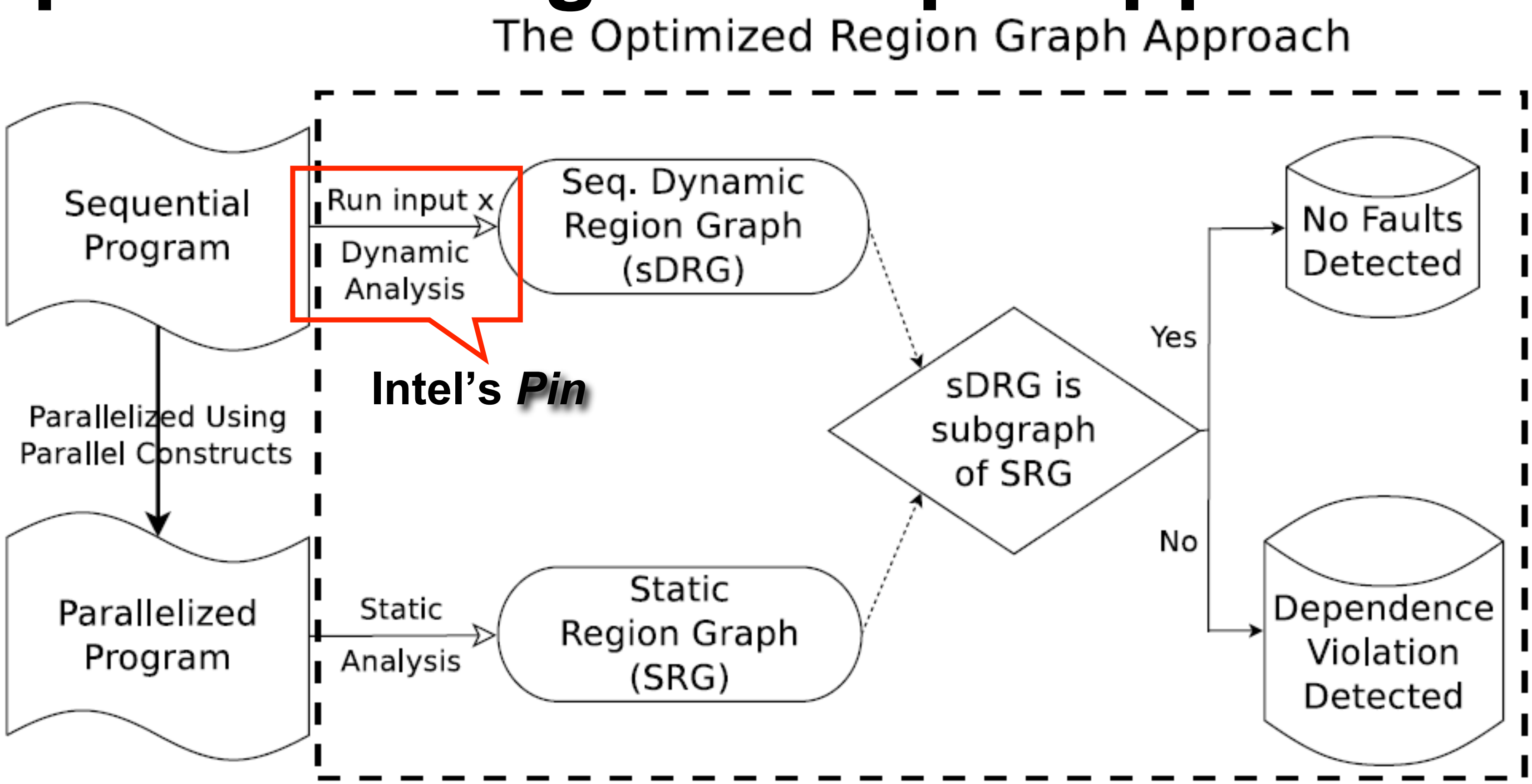
SRG

/ the same data race example… */*

Optimizing Comparison Checking

Optimization	Execution Time		Memory Space	
	Tracking	Checking	Tracking	Graph Size
OPT-1	√	√	√	√
OPT-2	√	√		√
OPT-3	√		√	

Optimized Region Graph Approach



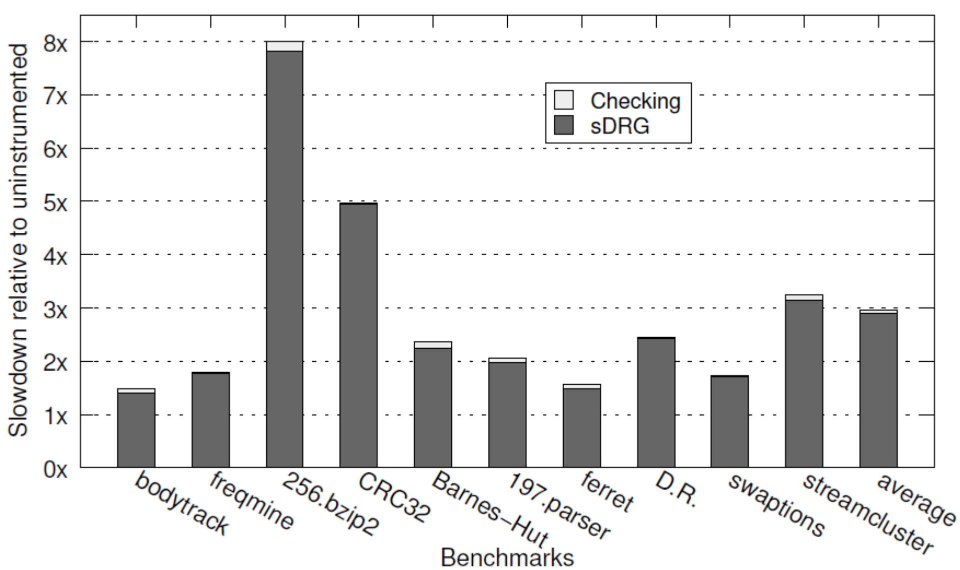
Observation: A dependence present in **sDRG**, but not allowed by **SRG**, represents *violation* against sequential program semantics by *parallelization* expressed by parallel constructs.

Evaluation

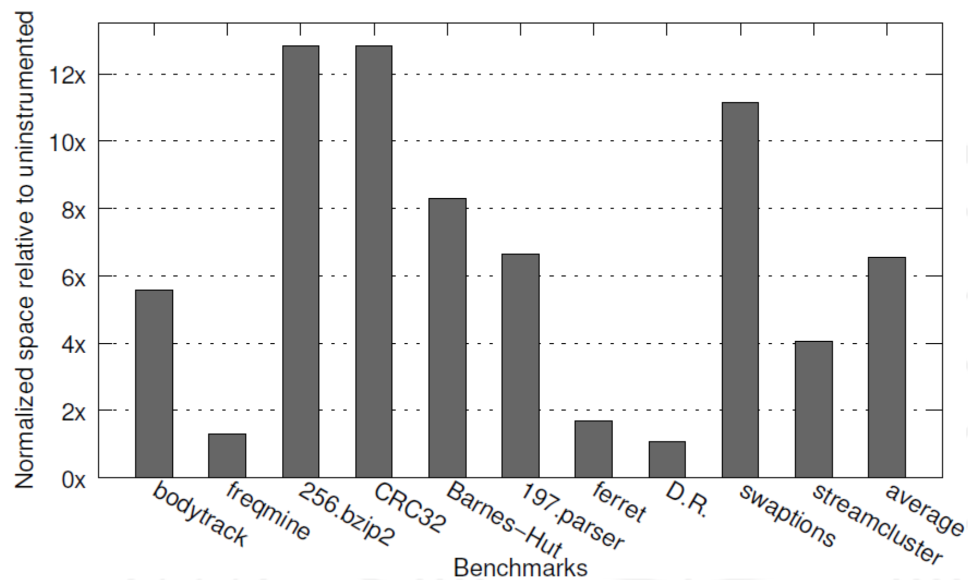
- Benchmarks
  - Applied our technique to ten benchmarks parallelized using OpenMP, SpiceC, and TBB
  - Selected from MiBench, SPEC CPU2000, Lonestar, and PARSEC benchmark suites

- Hardware Configuration
  - A 2.66 GHz Intel Core Duo DELL Dimension 9200 machine with 4 GB RAM
  - Linux kernel 2.6.32

sDRG Construction Overhead

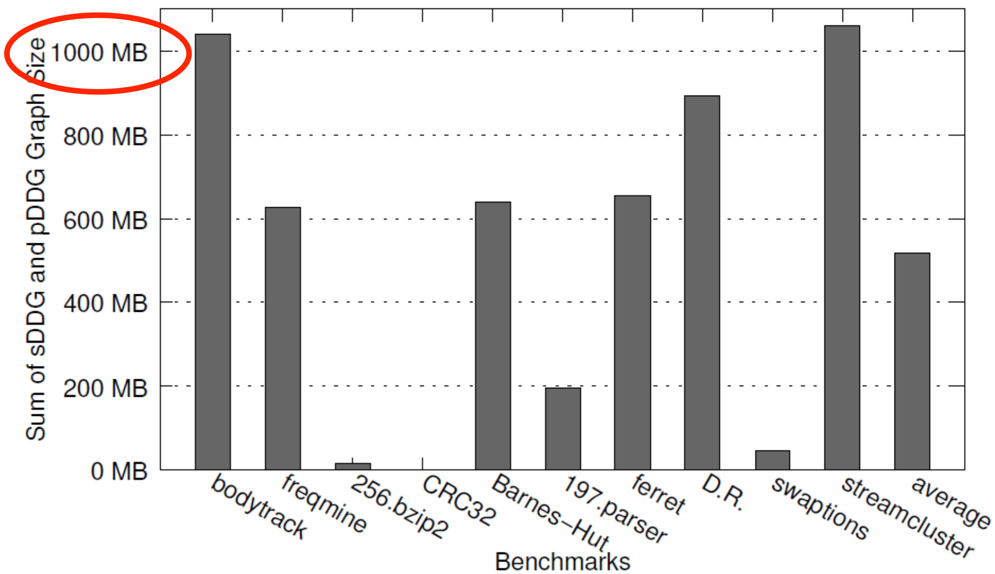


Execution Time

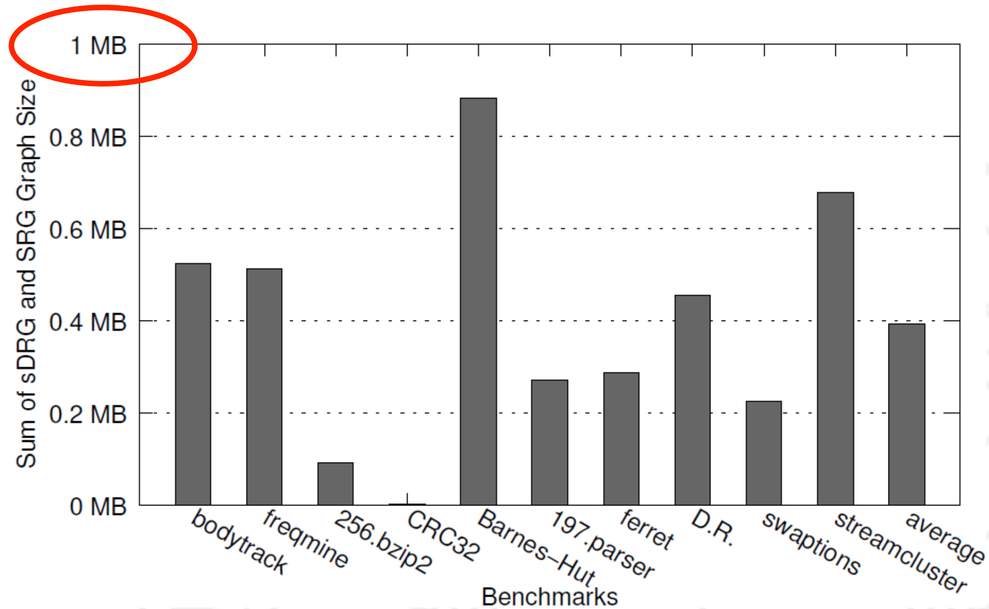


Memory Space

Graph Size Comparison



Previously



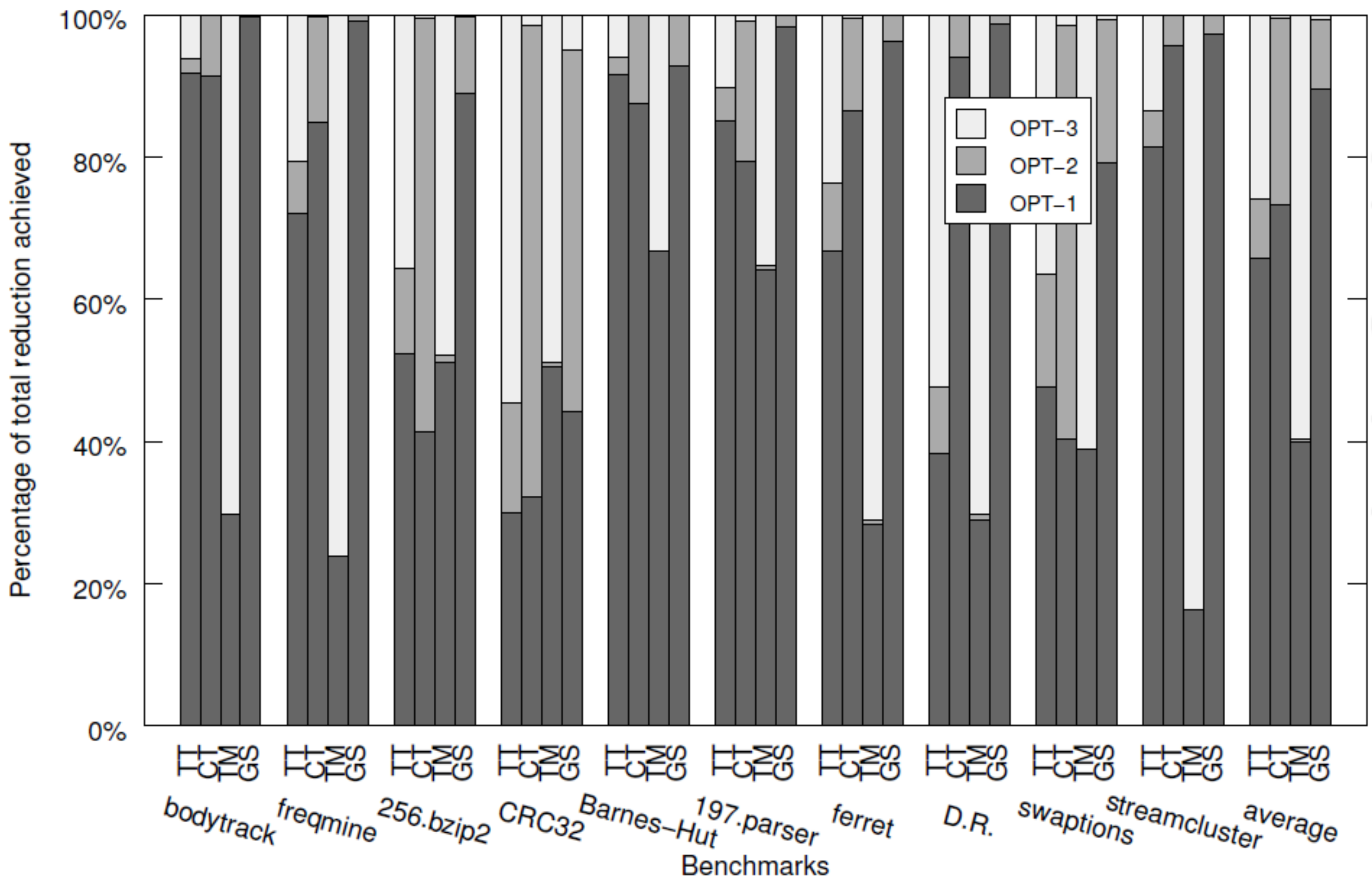
Now

Comparison of DRG and DDG

Cost of Using DRG as a Percentage of DDG

Benchmark	Execution Time		Memory Space	
	Tracking	Checking	Tracking	Graph Size
bodytrack	4.487%	0.371%	45.765%	0.049%
freqmine	14.416%	0.586%	51.985%	0.082%
256.bzip2	17.594%	0.623%	46.155%	0.584%
CRC32	22.854%	0.694%	47.240%	1.923%
Barnes-Hut	5.819%	1.519%	32.463%	0.139%
197.parser	9.474%	2.632%	34.419%	0.138%
ferret	12.713%	1.067%	33.196%	0.044%
DelaunayRefinement	33.483%	0.371%	71.006%	0.051%
swaptions	32.437%	1.976%	36.379%	0.494%
streamcluster	9.884%	1.282%	39.001%	0.064%
GeoMean	**13.463%**	**0.907%**	**42.534%**	**0.151%**

Breakdown of Overhead Reduction



Conclusions

- Debugging *Parallelized* Programs
  (OpenMP, SpiceC, and TBB)

  Versatility
  - Support for multiple types of concurrency bugs
  - Support for multiple parallel programming models

  Novelty
  - No requirement for execution of parallel programs
  - Elimination of *reproducibility* and *validity* problems
  - Region level data dependence graphs
  - Only 3x slowdown on average