

# May-Happen-in-Parallel Analysis

Yao Shi, Tsinghua University  
Shanghai Many-Core Workshop  
March 27, 2008



# Outline

- Background
- Challenges
- Our approach in Open64



# Background

- Multi-threaded is the maturest model
- Hard to write and debug multi-threaded programs
- Automatic bug detection
  - Data race detection
- May-Happen-in-Parallel analysis (concurrency analysis)



# Background

- Data race detection

```
int global = 0;
void* foo(void *arg) {
    global++;
}
int main() {
    pthread_t thd1, thd2;
    pthread_create(&thd1, 0,
foo, 0);
    pthread_create(&thd2, 0,
foo, 0);
}
```

// Two foo threads may edit  
"global"  
// in parallel. Data race!

```
int global = 0;
void* foo(void *arg) {
    global++;
}
int main() {
    pthread_t thd1, thd2;
    pthread_create(&thd1, 0,
foo, 0);
    pthread_join(thd1, 0);
    pthread_create(&thd2, 0,
foo, 0);
}
```

// Two foo threads are  
executed  
// sequentially. There is no  
data race.



# Background

- Gleb Naumovich and George S. Avrunin. A Conservative Data Flow Algorithm for Detecting All Pairs of Statements that May Happen in Parallel
  - Ada, nodes < 700
- Gleb Naumovich, George S. Avrunin, and Lori A. Clark . An Efficient Algorithm for Computing MHP Information for Concurrent Java Programs
  - Java, threads  $\leq 7$ , nodes < 450
- Lin Li and Clark Verbrugge. A Practical MHP Information Analysis for Concurrent Java Programs
  - Java, nodes < 300
- Rajkishore Barik . Efficient Computation of May-Happen-in-Parallel Information for Concurrent Java Programs
  - Java, threads  $\leq 3$ , nodes < 2200
- Christoph von Praun and Thomas R. Gross. Static Conf'' Analysis for Multi-Threaded Object-Oriented Programs
  - Objects in Java, up to 30 KLOC



# Challenges

- C/C++ specific problem
- Storage without equivalent relation
- Efficient happen-before in static analysis



# Challenges

- C/C++ specific problem

```
class Java_task extends
Thread {
    public void run() {
        .....
    }
}

Java_task t = new Java_task();
t.start();
t.join();
// the corresponding functions
to
// thread t is "run"s of the
subclasses
// of the static class type
Java_task
```

```
void* foo(void *arg) { ..... }
void* bar(void *arg) { ..... }
void* quu(void *arg) { ..... }

void *ret;
pthread_t t;
quu(NULL);
if (...)
    pthread_create(&t, NULL, foo,
NULL);
else
    pthread_create(&t, NULL, bar,
NULL);
pthread_join(t, &ret);
// how to know thread t is
attached
// with function foo or function
```



# Challenges

- Storage without equivalent relation
  - If May-Parallel is equivalent relation, each statement can be given an identifier of parallel equivalent class
  - If not, ... ?
- Is MHP equivalent relation?

$$\left. \begin{array}{l} \text{STMT1} \parallel \text{STMT3} \\ \text{STMT2} \parallel \text{STMT3} \end{array} \right\} \Rightarrow \text{STMT1} \parallel \text{STMT2} \quad ?$$

```
if (cond)
    pthread_create(&t, 0, branch0, 0); // STMT1, may be parallel with
    STMT3
else
    pthread_create(&t, 0, branch1, 0); // STMT2, may be parallel with
    STMT3
.....
STMT3;
// STMT1 must not be parallel with STMT2 though we can say they
```

- **Equivalent relation is correct but imprecise**



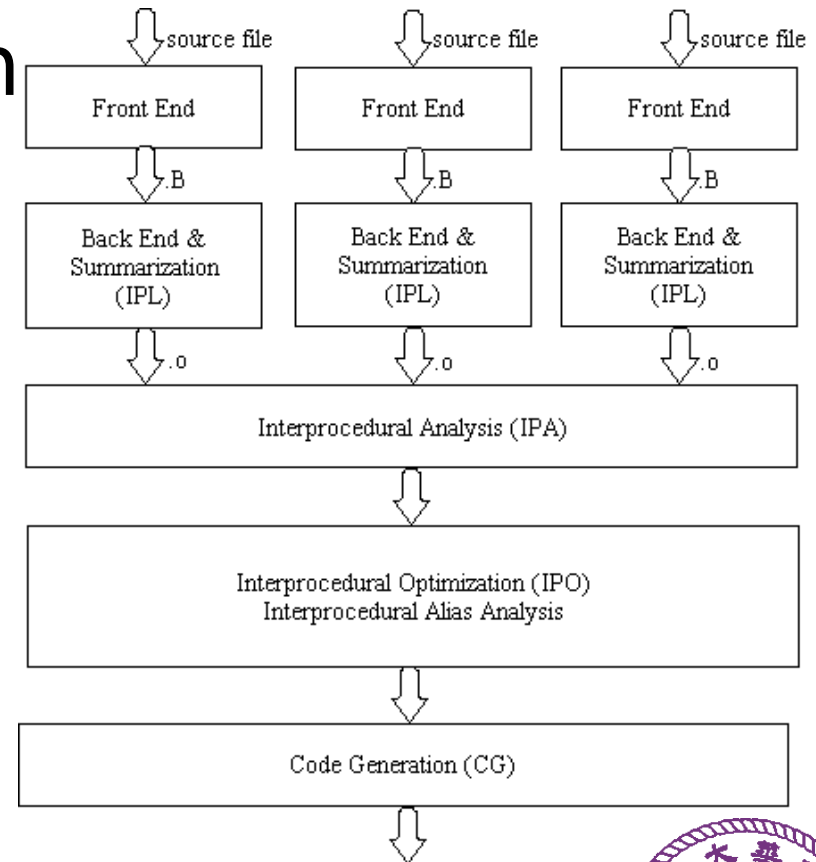
# Challenges

- Reachability or happen-before
  - One statement can reach/happens before another one
  - It is the fundament of many MHP algorithms, but...
- How to determine happen-before efficiently in static analysis?
  - Temporal priority : keep all relations of all stmts, use  $O(n^2)$  space, which  $n$  is the number of stmts
  - Spatial priority : only keep relations between adjacent stmts, use linear space and  $O(n^2)$  query time
  - Tradeoff between time and space

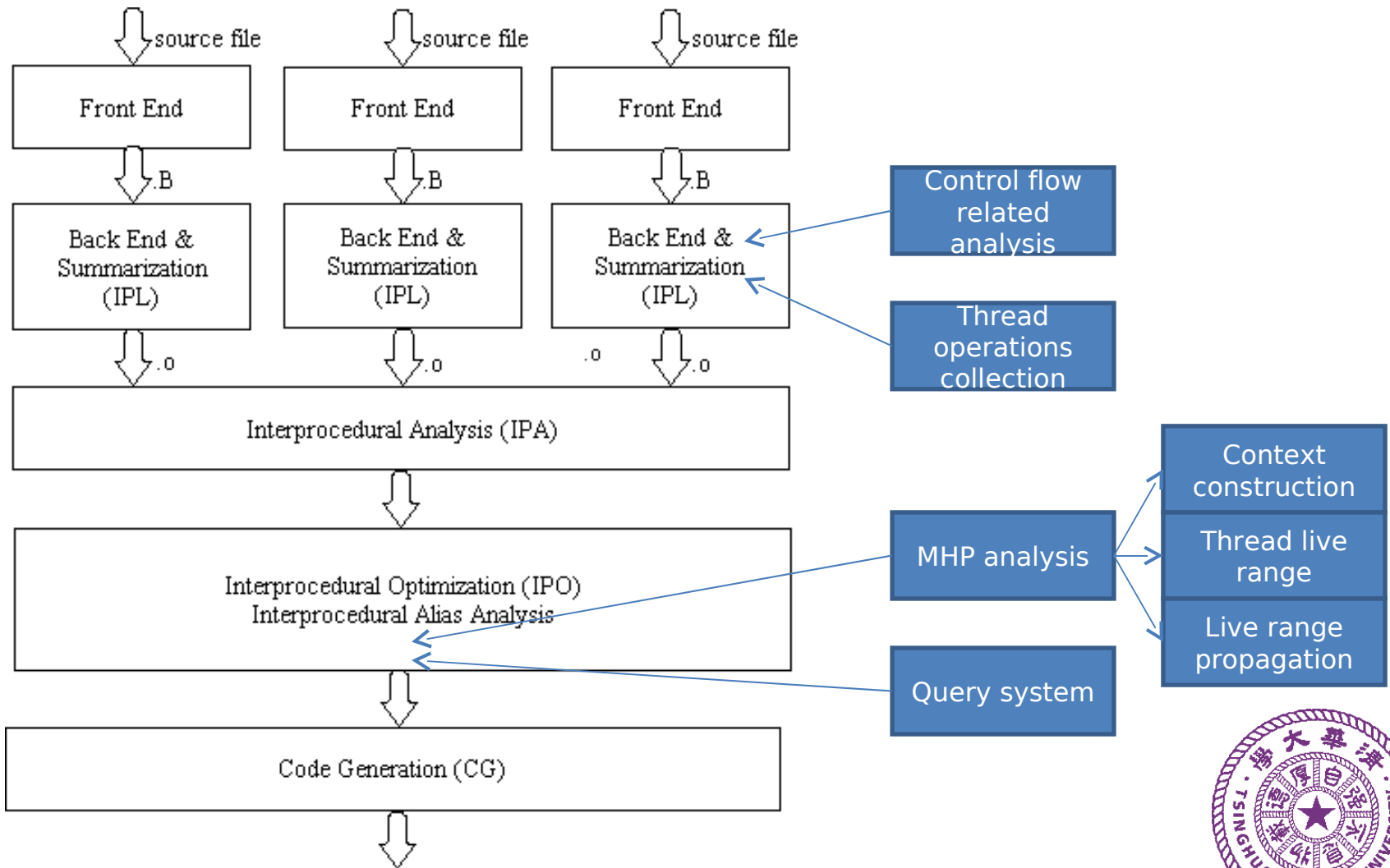


# Approach in Open64

- What do we have in Open64 Compiler?
  - Interprocedural analysis framework
  - Interprocedural context-free alias analysis
  - Intraprocedural control flow graph



# Approach in Open64



# Thank You!

- Q & A

