Tracking Data Structures for Postmortem Analysis

Xiao Xiao, Jinguo Zhou, Charles Zhang

Computer Science @ HKUST

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Issues of Program Tracing

- Target:
  - Collect the ***field*** execution data for program diagnosis.

- Significance:
  - Improve the software testing efficiency;
  - Capture/replay the obscured program behaviours;
  - Identify the performance problems.

- Challenges (e.g. whole program execution trace produced by iDNA, WET):
  - High volume information
  - Poor information privacy
  - High collection overhead

Issues of Program Tracing

- Target:
  - Collect the ***field*** execution data for program diagnosis.

- Significance:
  - Improve the software testing efficiency;
  - Capture/replay the obscured program behaviours;
  - Identify the performance problems.

- Challenges (e.g. whole program execution trace produced by iDNA, WET):
  - High volume information → partial trace
  - Poor information privacy → partial trace
  - High collection overhead

Issues of Program Tracing

- Target:
  - Collect the ***field*** execution data for program diagnosis.

- Significance:
  - Improve the software testing efficiency;
  - Capture/replay the obscured program behaviours;
  - Identify the performance problems.

- Challenges (e.g. whole program execution trace produced by iDNA, WET):
  - High volume information
  - Poor information privacy
  - High collection overhead → incomplete trace

Partial Trace Collection

- Data structure evolution trace provides rich but non-confidential information for diagnosing the program.

- Consider the following literature:

  - Data structure verification, [PLDI 07, Shankar];
  - Heap bug detection, [ASPLOS 06, Chilimbi];
  - Dynamic ownership detection, [ECOOP 06, Mitchell];
  - Dynamic slicing, [ICSE 04, Tao. W]
  - Memory leak detection, [ECOOP 03, Mitchell];
  - Copy bloating analysis, [PLDI 09, Harry X.];
  - Data layout optimization, [POPL 02, Rubin];
  - Heap visualization, [SOFTVIS 10, Aftandilian];
  - .....

Data Structure Trace

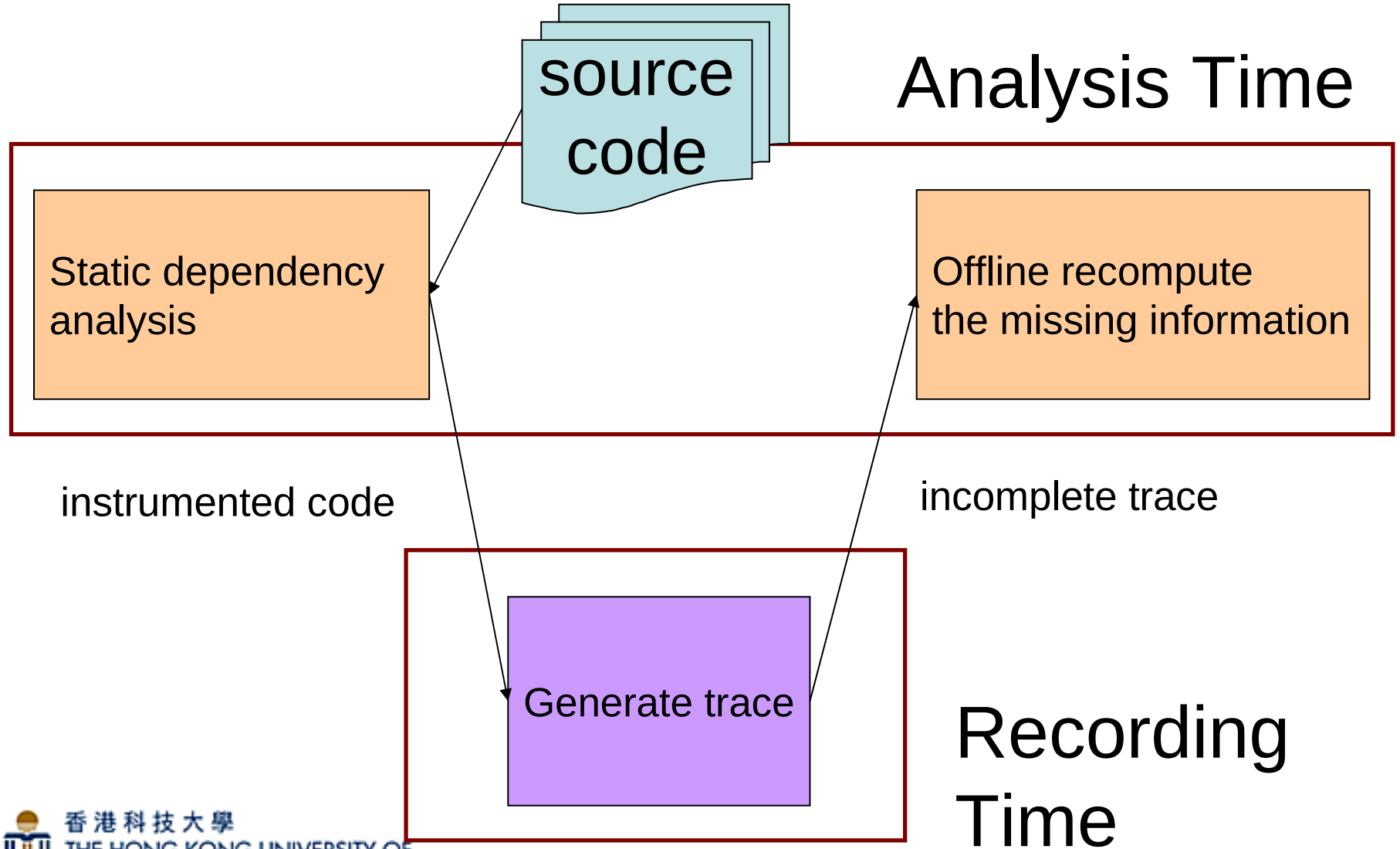
- A trace that can reconstruct the whole structural evolution of the data structures:
  - Do not need reexecute the program;
  - Enable forward and backward replay.

Replay with incomplete trace

- A trace that is lacking information for replay. The missing information should be statically computed.

- Advantages:
  - Lightweight instrumentation;
  - Small trace size;
  - Low recording overhead.

Processing Incomplete Trace



Conclusion

- We propose the data structure trace for diagnosing the program:
  - Rich Information for postmortem analysis;
  - Confidential;
  - Compact;
  - Low recording overhead;

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Additional Notes

Case Study: Loop-incomplete Trace

```
// Traverse and truncate linked list
// header is a global pointer

1.  p = header;
2.  q = null;
3.  while ( p != null ) {
4.      temp = p.next;
5.      if ( p.data > 10 ) {
6.          p.next = q;
7.          q = p;
8.      }
9.      p = temp;
10. }
11. header = q;
```

Baseline: instrumenting and monitoring the variables in the statements at 4, 6, 11

Case Study: Loop-incomplete Trace

```
// Traverse and truncate linked list
// header is a global pointer

1.  p = header;
2.  q = null;
3.  record p;
4.  record q;
5.  while ( p != null ) {
6.      temp = p.next;
7.      if ( p.data > 10 ) {
8.          p.next = q;
9.          q = p;
10.     }
11.     p = temp;
12. }
13. header = q;
```

Motivation: Tracing loop is expensive.

Insights: Any piece of code can be reexecuted deterministically beginning with the same program states.

Loop-incomplete trace: Insert two lines (3, 4) to record the initial values of p and q at runtime, and maintain the execution sequence of the statements 6, 8, 9, 11 within the loop.

Case Study: Loop-incomplete Trace

```
// Traverse and truncate linked list
// header is a global pointer

1.  p = header;
2.  q = null;
3.  record p;
4.  record q;
5.  while ( p != null ) {
6.      temp = p.next;
7.      if ( p.data > 10 ) {
8.          p.next = q;
9.          q = p;
10.     }
11.     p = temp;
12. }
13. header = q;
```

Loop-incomplete trace: After the program terminated, we symbolically execute the statements 6, 8, 9, 11, 13 in their runtime execution order. For example,

6, 8, 9, 11, 6, 11, 6, 11, 6, 8, 9, 11, 13