Parallel Code Parsing Working Group Outbriefs

Revisiting parallel code parsing

Large scale static binaries from the labs

- 200MB ~ 300MB code sections
- 4GB total in size
- Takes about an hour to analyze in serial

Main issues of parallel code parsing

- Lazy initialization
- Static non-thread-safe global variables
- Lock contention

The current status

Correctness

- 1 out of 200 runs crashed
  - There are still data races
- one-thread runs and multi-thread runs do not always have the same results
  - The order of parsing impacts final results

Performance

- Good scaling if only considering parallel code parsing phase
- Reasonable scaling end-to-end

Memory consumption

- Analyzing 1.5GB binary consumes about 22GB memory

Action items

Stabilize parallel code parsing

- Use Cilk data race detector
- Fix undeterministic parsing results

In the near future

- Fix non-thread-safe problems in libdw for parallel DWARF parsing
- improve parallel code parsing
  - Memory issues
  - Parallelize serial parsing initialization
  - Implement lazy interval tree updates to get rid of serial function finalization

Since the working group

- Installed and ran the cilk race detector
- Fixed one real data race
- There are 21 unique call stacks up in the data race report
  - About 6 independent data race issues