PPH Lessons Learned

Lawrence Crowl and Diego Novillo

GNU Cauldron 2012
Prague

PPH Approach

- Constraints
  a. Minimize change to existing sources
  b. Headers must be *modular*
  c. Non-modular headers embedded in PPH images
- GCC implementation
  a. Stream internal data structures
  b. Re-interpret `#include` with user-defined mappings
  c. Symbol merging for non-modular headers

Implementation status

- Significant performance gains
  - parse time: 26% of original time
  - compile time: 57% of original time
  - Implementation not yet tuned
- Limited functionality
  - System headers cause much merging
  - Fixes to merge bugs tend to cause refactoring
  - Bugs walk in a straight line to hide their numbers

GCC Problems

- Reinclusion guards do not always trigger in the way we need
- Parser conflates parsing with codegen
- Circularity in data structures make merging harder
- Compiler tuned to traditional compilation works against decoupling
- Loss of fidelity in representing C++
- Parser state spread out in multiple file statics and globals

Build system problems

- PPH generation potentially doubles number of build actions
- PPH creates more saved state
- Overhead cannot be measured until we have a functional compiler

C++ Language Problems

- Most system headers unsuitable to PPH
- Merging issues costly
  - 1/4 to 1/3 of PPH load time
  - Use significant PPH file space
  - Primary source of bugs in implementation
- PPH need to contain as much info as original headers
  - No export control

The World Changed

- Test source size grew by 3.4x
- Test compile time grew by 3.6x
  - Parser is now 60% instead of 80%
  - Maximum possible speedup is 2.5x vs 5x
- Static analysis tools run in parallel with GCC
  - Cannot speedup more than 3x-4x
  - Must ship both PPH files and headers
- New automatic source changing tools
  - Significant source changes now feasible

Alternate Approaches

- Chainable PCH
- Stream parser actions instead of ASTs
- Make definitions exclusive to a package
- Stream inline functions in optimizer IR form
- Implement export control
- Multi-thread the compiler
- Tools to re-design headers to reduce drag
- Optimize GCC's traditional optimization pipeline

Lessons and Recommendations

- Problem still needs solution
- PPH will deliver significant compile time gains
- Merging is hard; bugs are sequential
- Representation tuned to traditional compilation
- Better header semantics improve performance
- Modules need to cooperate with build system
- Modules affect several communities
- Automated source conversion can simplify the problem