Minimum Size Function Coverage

Ellis Hoag
Kyungwoo Lee
-preinline-threshold=5

- Extra inlining pass added before PGOGen
  - [https://reviews.llvm.org/D21405](https://reviews.llvm.org/D21405)
  - Default inline threshold = 75

- Threshold should match optimization mode

```c
// llvm/include/llvm/Analysis/InlineCost.h

/// Use when minsize (-Oz) is specified.
const int OptMinSizeThreshold = 5;

/// Use when -O3 is specified.
const int OptAggressiveThreshold = 250;
```
-enable-post-pgo-loop-rotation=false

- Loop rotation transformation added after PGOGen
  - [https://reviews.llvm.org/D34085](https://reviews.llvm.org/D34085)
  - Can increase code size
  - Not useful for function entry coverage
-pgo-function-size-threshold

- Do not instrument small functions
  - #(LLVM-IR Instructions) < Threshold
- Imprecise
  - Good enough to identify very small functions
noprofile & skipprofile

● Attribute::NoProfile
  ○ ✅ Prevents instrumentation
  ○ ❌ Prevents **inlining** if callee/caller disagree on attribute
    ■ Needed to guarantee safety
    ■ Disables most inlining

● Attribute::SkipProfile
  ○ ✅ Prevents instrumentation
  ○ ✅ No restrictions on inlining
    ■ No codesize/performance surprises
-fprofile-list

- **Special case list** to specify functions and files to instrument

```plaintext
[llvm]
#define Attribute::SkipProfile
#define Block C++ standard library functions
function:_ZSt*_=skip

#define Attribute::NoProfile
source:lib/unsafe/*.cc=forbid

#define Otherwise we allow profiling.
default:allow
```
-fprofile-function-groups

- Partition functions into \( N \) groups
- Only instrument functions in group \( i \)
- Collect profiles independently from each group
- Merge profile offline

```bash
$ clang++ -Oz -fprofile-generate=g0/ -fprofile-function-groups=3 -fprofile-selected-function-group=0 code.cc -o code.0
$ clang++ -Oz -fprofile-generate=g1/ -fprofile-function-groups=3 -fprofile-selected-function-group=1 code.cc -o code.1
$ clang++ -Oz -fprofile-generate=g2/ -fprofile-function-groups=3 -fprofile-selected-function-group=2 code.cc -o code.2
$ llvm-profdata merge -output=code.profdata g*//*.profraw
```
Clang-16 Size Overhead

.text Size Overhead | Binary Size Overhead

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Size Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Empty&quot; IRPGO</td>
<td>15.73%</td>
</tr>
<tr>
<td>Function Coverage</td>
<td>18.77%</td>
</tr>
<tr>
<td>Preinline Threshold (5)</td>
<td>3.96%</td>
</tr>
<tr>
<td>No Loop Rotation</td>
<td>5.19%</td>
</tr>
<tr>
<td>Size Threshold (4)</td>
<td>5.71%</td>
</tr>
<tr>
<td>Profile List</td>
<td>5.70%</td>
</tr>
<tr>
<td>Function Groups (2)</td>
<td>4.88%</td>
</tr>
<tr>
<td></td>
<td>3.51%</td>
</tr>
<tr>
<td></td>
<td>1.90%</td>
</tr>
<tr>
<td></td>
<td>1.47%</td>
</tr>
<tr>
<td></td>
<td>1.79%</td>
</tr>
<tr>
<td></td>
<td>0.74%</td>
</tr>
</tbody>
</table>