Optimization Diary

June 02, 2006
Version number: 0.12

Table of Contents

Objective

In order to improve productivity of developers, design and implement new mechanism to present Compiler's feedback to developers in suitable way.

Motivation

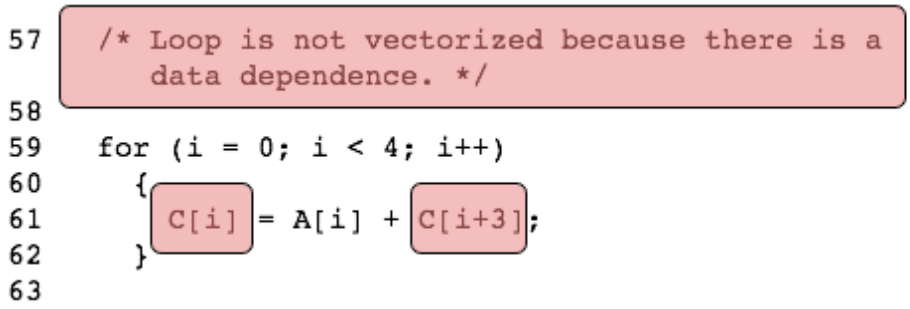
Modern compilers are capable of optimizing user code to improve performance of generated code. However, code optimizer works like a black box. It receives user code in compiler's internal representation form and generates instructions. Developers analyze generated instructions to understand what optimizer did. Sometimes developers use performance tools (e.g. Shark in Mac OS X) to analyze and fine tune their program. In both cases, developers rely on compiler generated instructions to interpret compiler's actions. Developers are often interested in receiving direct feedback from compiler regarding optimizer's various actions.

Solution

We propose that compiler emits optimization diary that is available to external tools to collect information about optimizer's behavior. Information in this diary is concise and well formated.

```
73  /* Loop is automatically vectorized.  */
74
75  for (i = 0; i < N; i++){
76    a[i] = b[i] + c[i];
77    d[i] = b[i] + c[i];
78    ia[i] = ib[i] + ic[i];
79  }
80  ibar (ia);
```

```
65  /* Loop is not automatically vectorized.  */
66
67  for (i = 0; i < N/2; i++){
68    a[i] = b[2*i+1] * c[2*i+1] - b[2*i] * c[2*i];
69    d[i] = b[2*i] * c[2*i+1] + b[2*i+1] * c[2*i];
70  }
71  fbar (a);
```



Optimization Diary Content

It is important to not make diary extremely verbose. Compiler optimizer does many things that are not suitable for diary entry. The goal is to include items that are useful to developer.

Data Store Design

There are certain requirements that data store design is required to meet.

- **Extensible**. Data store should be extensible to accommodate new compiler optimization techniques.
- **Compact**. It is very important that data store is able to pack as much information as possible in small size. If optimization diary is used to analyze very large project then its size should not have negative impact on the performance.
- **Quick Navigation.** Tools should be able to search and navigate through diary content very fast.

Optimization diary content can be divided into different categories. For example,

1. Parameter Manipulation Hint

Compilers allow developers to set various parameters to control optimizer's behavior. Information related to optimizer's decision based on user selected (or default) parameter falls into this category. For example, GCC allows user to set inlinining limit. If optimization diary entry indicates that a function foo is not inlined because inlining limit is too low then this makes tuning code to newer GCC compilers much easier.

2. Optimization Hint

Information about how to update source code to get better performance falls into this category. Naturally, developer has more knowledge about input data set and compiler does not know if developer is using optimal algorithm or not.

3. Optimization Limit

Information about theoretical as well as implementation limitations of optimizer falls into this category. For example, GCC 4.0 is not able to automatically vectorize reduction operations.

4. Optimization Report

This is default category to keep log of various optimizer activities.

DWARF extensions for Optimization Diary

Optimization diary content is stored using following new tag and attributes..

DW_TAG_GNU_OD_entry (0x5001) Each diary entry is represented by a debugging info die with DW_TAG_GNU_OD_entry tag. DW_AT_GNU_OD_category attribute is always used to categorize diary entry. Each optimization entry overloads DW_AT_decl_file and DW_AT_decl_line to identify code location. DW_AT_decl_column is optionally used to identify column position. DW_AT_GNU_OD_msg attribute is included most of the times, but it is optional. It may also include DW_AT_GNU_OD_cmd attribute.

DW_AT_GNU_OD_msg (0x2401) This attribute holds optimizer message, whose value is a ULEB128 constant. Following table describes meaning of each message.

Value	Meaning
1	Loop is automatically vectorized
2	Loop is not automatically vectorized
3	Loop versioning is used during automatic vectorization
4	Loop peeling is used during automatic vectorization
5	Loop is not vectorized because it has muiltiple exits
6	Loop is not vectorized because of bad data reference
7	Loop is not vectorized because these operation is not supported in vector form
8	Loop is not vectorized because there is a data dependence
9	Loop is not vectorized because alignment for data reference is not suitable
10	Loop is not vectorized because reduction operations are not supported
11	Function is inlined
12	Function is not inlined because inline limit is too low
13	Function is not inlined because it is marked as “Do Not Inline”
14	Function is not inlined because it is very big
15	Function is not inlined because it is a recursive function
16	This function is a potential inlining candidate if function body is visible at call site
17	Loop is unrolled
18	Loop is not unrolled because maximum unrolled iteration limit is reached
19	Loop is not unrolled because maximum unrolled instruction limit is reached

DW_AT_GNU_OD_cmd (0x2402) This attribute holds diary command (e.g. highlight this text), whose value is ULEB128 constant.

Value	Meaning
1	Highlight Text
2	Dead code (Gray Text)

DW_AT_GNU_OD_category (0x2403) This attribute indicates category of optimization diary entry. Its value is one bitwise ULEB128 constant. Each bit is used to represent one category.

Value	Meaning
0x0000 0001	Parameter manipulation hint. Optimization diary entry is referring GCC command line parameter.
0x0000 0002	Optimization Limit. Optimization diary entry refers to implementation limitations.
0x0000 0004	Optimization Report. This is used to leave optimizer action trails.
0x0000 0008	Action bit. This is a toggle bit. When it is set, it indicates action performed by optimizer, for example function is inlined. When it is reset, it indicates action avoided by optimizer, for example loop is not vectorized.
0x0000 0016	Optimization Hint.

DW_AT_GNU_OD_version (0x2404) This attribute indicates Optimization Diary version number, whose value is an integer constant. DW_TAG_compiler_unit has one DW_AT_GNU_OD_version attribute if Optimization Diary is available.

Example

```
DW_TAG_GNU_OD_entry
      DW_AT_GNU_OD_category (0x0000 0012)
      DW_AT_decl_file (”foo.c”)
      DW_AT_decl_line (42)
      DW_AT_GNU_OD_msg (1)
```