

optview2

View and improve optimizations



@OfekShilon

Ofek Shilon
Istra Research
August 2021

What if the compiler could tell us...

- I couldn't inline that function
 - ...because I had only its declaration.
- I'm re-evaluating these loop boundaries on every iteration
 - ...because I couldn't prove they stay fixed.
- I'm doing tons of memory accesses that might be redundant
 - ...because in theory two unrelated variables might refer to the same memory location. You modified one, so I had to reload the other from memory.

Raw optimization data *is* exposed

- Clang/gcc: -Rpass

code.cc:4:25: remark: foo inlined into bar [-Rpass=inline]

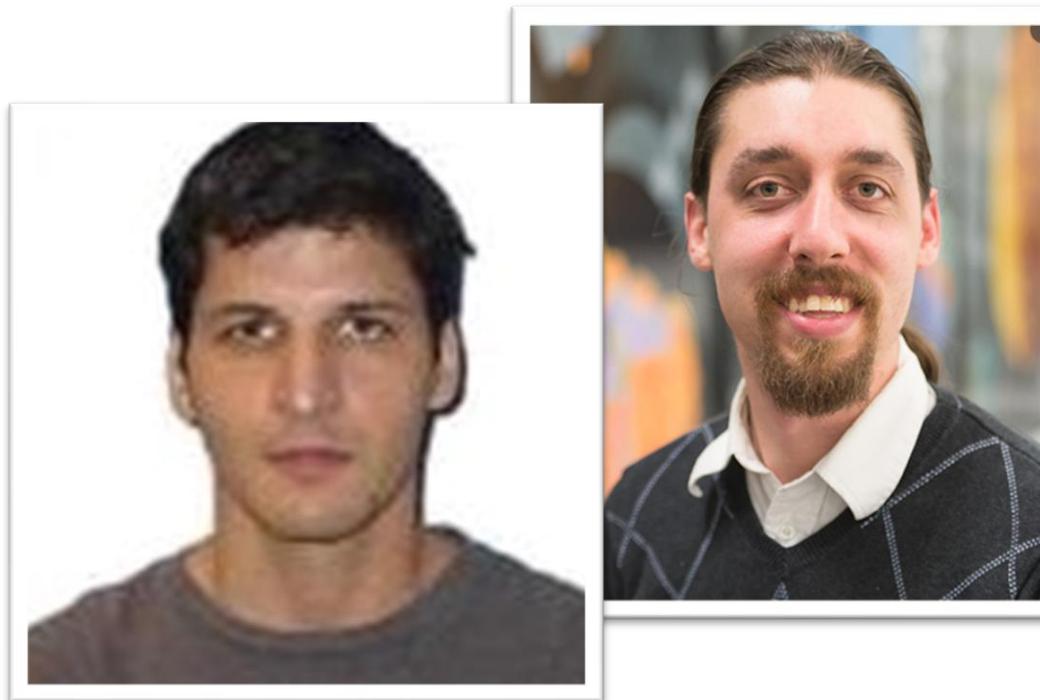
- Intel: -qopt-report=[0..5]

- Microsoft (very partial):
/Qpar-report, /Qvec-report

```
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:446:33: remark: load of type i32 not eliminated in favor of store because of alignment requirement
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:405:8: remark: Func3 can be inlined into Proc6 with cost=0 (threshold=4)
if (! Func3(EnumParIn) )
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:405:8: remark: Func3 inlined into Proc6 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:271:3: remark: Proc5 can be inlined into Proc8 with cost=10 (threshold=4)
Proc5();
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:271:3: remark: Proc5 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:292:3: remark: Proc2 can be inlined into Proc8 with cost=25 (threshold=4)
Proc2(&IntLoc1);
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:292:3: remark: Proc2 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:288:5: remark: Proc6 can be inlined into Proc8 with cost=-5 (threshold=4)
Proc6(Ident1, &EnumLoc);
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:288:5: remark: Proc6 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:287:19: remark: Func1 can be inlined into Proc8 with cost=10 (threshold=4)
if (EnumLoc == Func1(CharIndex, 'C'))
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:287:19: remark: Func1 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:285:3: remark: Proc1 can be inlined into Proc8 with cost=15 (threshold=4)
Proc1(PtrGlob);
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:285:3: remark: Proc1 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:284:3: remark: Proc8 can be inlined into Proc8 with cost=125 (threshold=4)
Proc8(Array1Glob, Array2Glob, IntLoc1, IntLoc3);
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:284:3: remark: Proc8 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:281:4: remark: Proc7 can be inlined into Proc8 with cost=-5 (threshold=4)
Proc7(IntLoc1, IntLoc2, &IntLoc3);
^
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:281:4: remark: Proc7 inlined into Proc8 [-Rpass=inline]
/org/test-suite/SingleSource/Benchmarks/Dhrystone/dry.c:277:16: remark: Func2 can be inlined into Proc8 with cost=90 (threshold=4)
BoolGlob = ! Func2(String1Loc, String2Loc);
^
```

Presentation Matters.

- Focus on Clang, over Linux
- **opt-viewer**, 2016 work by Adam Nemet (Apple), Hal Finkel (Argonne)



Opt-Viewer Usage

- Build with
-fsave-optimization-record:
 - * `.opt.yaml` files are generated, by default in the `obj` folder.
 - Generate htmls:
opt-viewer.py
 - `--output-dir <htmls folder>`
 - `--source-dir <repo>`
 - `<yamls folder>`

opt-viewer.py Sample Output

Hotness
(PGO)

52%
100%

loop-delete
loop-vectorize

52%

loop-idiom

17%

gvn

17%

gvn

34%

gvn

17%

gvn

```
REG OneToFifty  IntIndex;

    IntLoc = IntParI1 + 5;
    Array1Par[IntLoc] = IntParI2;
    Array1Par[IntLoc+1] = Array1Par[IntLoc];
    Array1Par[IntLoc+30] = IntLoc;
    for (IntIndex = IntLoc; IntIndex <= (IntLoc+1); ++IntIndex)
        loop deleted
        vectorized loop (vectorization width: 4, interleaved count: 2)
            Array2Par[IntLoc][IntIndex] = IntLoc;
            formed memset
            ++Array2Par[IntLoc][IntLoc-1];
            load of type i32 not eliminated in favor of store because it is clobbered by store
            load of type i32 not eliminated in favor of store because it is clobbered by call
            Array2Par[IntLoc+20][IntLoc] = Array1Par[IntLoc];
            load of type i32 not eliminated in favor of store because it is clobbered by store
            load of type i32 eliminated
    IntGlob = 5;
}
```

Inlining
context

[Proc0](#)
[Proc8](#)

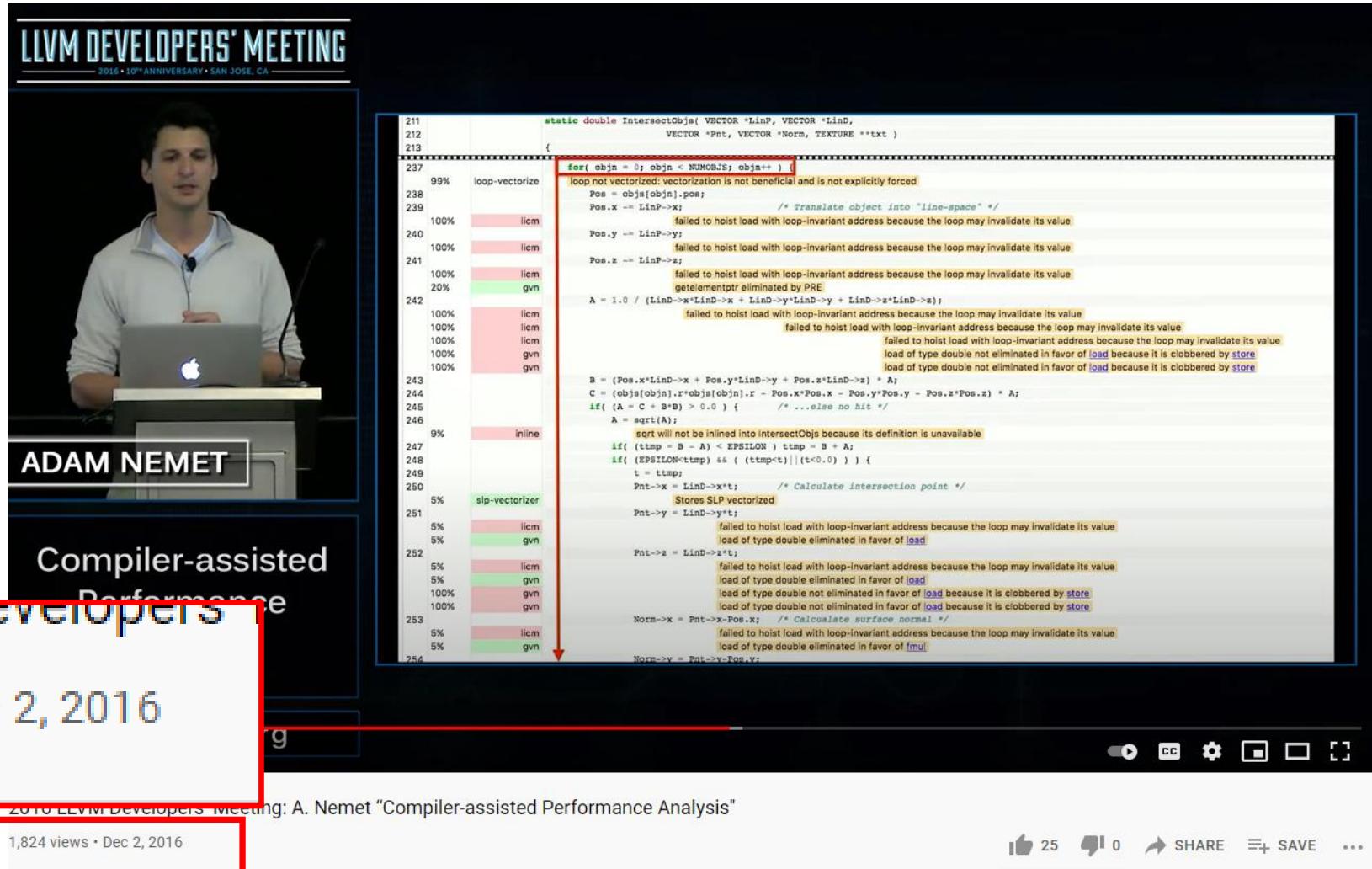
[Proc0](#)

[Proc0](#)
[Proc0](#)

[Proc8](#)
[Proc0](#)

Great work, still not much traction

<https://www.youtube.com/watch?v=qq0q1hfzidg>



Great work, still not much traction

The screenshot shows a GitHub commit history for the repository `llvm-project / llvm / tools / opt-viewer /`. A red box highlights the last six commits, all of which occurred on March 25, 2020, just two years ago. The commits are:

- `CMakeLists.txt`: 2 years ago
- `opt-diff.py`: 2 years ago
- `opt-stats.py`: 2 years ago
- `opt-viewer.py`: 17 months ago
- `optpmap.py`: 3 years ago
- `optrecord.py`: 17 months ago
- `style.css`: 3 years ago

The commits were made by `LocutusOfBorg` and `sylvestre...` on Mar 25, 2020.

Why?

- Heavy
 - High I/O
 - High memory
 - >1G htmls
- Designed for compiler writers
 - Mostly non actionable to developers

Introducing optview2

- <https://github.com/OfekShilon/optview2>

Target Developers, Not Compiler Authors

- Denoise:
 - Collect only optimization *failures*
 - Ignore system headers
 - Display a single entry per type/source loc (in index)
 - ...
 - ~1.5M lines ==> 22K lines
- Don't mention 'passes'
- Make the index sortable, resizable & pageable
- Abridged function names
- ...

```
ssh://ofek@n56:22/home/ofek/src/optview2/venv/bin
/python3 -u /tmp/pycharm_project_678/opt-viewer
.py --jobs 25 --output-dir
/data/userdata/optview2_html/istra --source-dir
/home/ofek/repoOptView
/data/userdata/ofek/repoOptView/obj
For faster parsing, you may want to install
libYAML for PyYAML
Reading YAML files...
2413 of 2413
Rendering index page...
501821 optimization-failure remarks
31023 unique source locations
22236 after filtering irrelevant
Rendering HTML files...
2000 of 2000
Ran for 0:46:10.436379
```

Demos - OpenCV

← → ⌂ File | C:/optview2/html/opencv/index.html

Show 100 entries

Location	Description	Function	Message
/home/ofek/src/opencv-clean/modules/ml/test/test_mltests.cpp	MissedDetails	<code>__cxx_global_array_dtor.44</code>	loop not vectorized
/home/ofek/src/opencv-clean/modules/ml/test/test_mltests.cpp	MissedDetails	<code>__cxx_global_array_dtor.51</code>	loop not vectorized
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	MissedDetails	<code>std::_Function_handler::finalize()::Connector>::_M_invoke</code>	loop not vectorized
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	TooCostly	<code>ade::CallbackConnector<>::finalize</code>	<code>std::function<void ()>::operator=<ade::CallbackConnector<>::finalize()::Connector></code> n ade::CallbackConnector<>::finalize because too costly to inline (cost=310, threshold
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	TooCostly	<code>ade::CallbackConnector<>::finalize</code>	<code>std::function<void ()>::operator=<ade::CallbackConnector<>::finalize()::(lambda)#1></code> ade::CallbackConnector<>::finalize because too costly to inline (cost=280, threshold
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	TooCostly	<code>ade::CallbackConnector<>::finalize</code>	<code>std::function<void ()>::operator=<ade::CallbackConnector<>::finalize()::(lambda)#2></code> ade::CallbackConnector<>::finalize because too costly to inline (cost=280, threshold
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	MissedDetails	<code>std::_Function_handler::finalize()::(lambda)#3>::_M_invoke</code>	loop not vectorized
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	TooCostly	<code>ade::CallbackConnector<>::finalize</code>	<code>std::function<void ()>::operator=<ade::CallbackConnector<>::finalize()::(lambda)#3></code> ade::CallbackConnector<>::finalize because too costly to inline (cost=280, threshold
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	TooCostly	<code>ade::CallbackConnector<>::finalize</code>	<code>std::function<void ()>::operator=<ade::CallbackConnector<>::finalize()::(lambda)#4></code> ade::CallbackConnector<>::finalize because too costly to inline (cost=280, threshold
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/common	NoDefinition	<code>ade::ICommChannel::BufferDesc::BufferDesc</code>	<code>ade::MemoryDescriptorRef::MemoryDescriptorRef</code> will not be inlined into ade::ICommChannel::BufferDesc::BufferDesc because its definition is unavailable
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/execut	TooCostly	<code>ade::ExecutionEngine::addPass</code>	<code>ade::ExecutionEngine::addPass<ade::passes::CheckCycles, char const*></code> not inlined int ade::ExecutionEngine::addPass<ade::passes::CheckCycles> because too costly to inline threshold=625)
3rdparty/ade/ade-0.1.1f/sources/ade/include/ade/execut	TooCostly	<code>ade::ExecutionEngine::addPass,</code> <code>cv::gapi::GNetPackage))(ade::passes::PassContext&,</code> <code>cv::gapi::GNetPackage const&), char const*</code>	<code>ade::PassManager<ade::passes::PassContext>::addPass<ade::ExecutionEngine::PassWrapp</code> <code>(*std::Placeholder<1>, cv::gapi::GNetPackage))(ade::passes::PassContext&, cv::gapi</code> <code>const&)>>></code> not inlined into ade::ExecutionEngine::addPass<std::Bind<void (*(std::cv::gapi::GNetPackage))(ade::passes::PassContext&, cv::gapi::GNetPackage const&), c because too costly to inline (cost=250, threshold=250)

Demo 1: inlining

```
template<typename _Tp> inline SeqIterator<_Tp>& SeqIterator<_Tp>::operator ++()
{
    CV_NEXT_SEQ_ELEM(sizeof(_Tp), *this);
inline • cvChangeSeqBlock will not be inlined into cv::SeqIterator<CvSeq*>::operator++ because its definition is unavailable
```

- cvChangeSeqBlock will not be inlined into
cv::SeqIterator<CvSeq*>::operator++ because its definition is unavailable
- Dirty fix:
 - Move cvChangeSeqBlock from datastructs.cpp to types_c.h
 - Copy some macros - CV_SEQ_ELEM, CV_WRITE_SEQ_ELEM_VAR CV_WRITE_SEQ_ELEM
 - Remove declaration from core_c.h
- Less relevant if you're building with LTO

Demo 2:

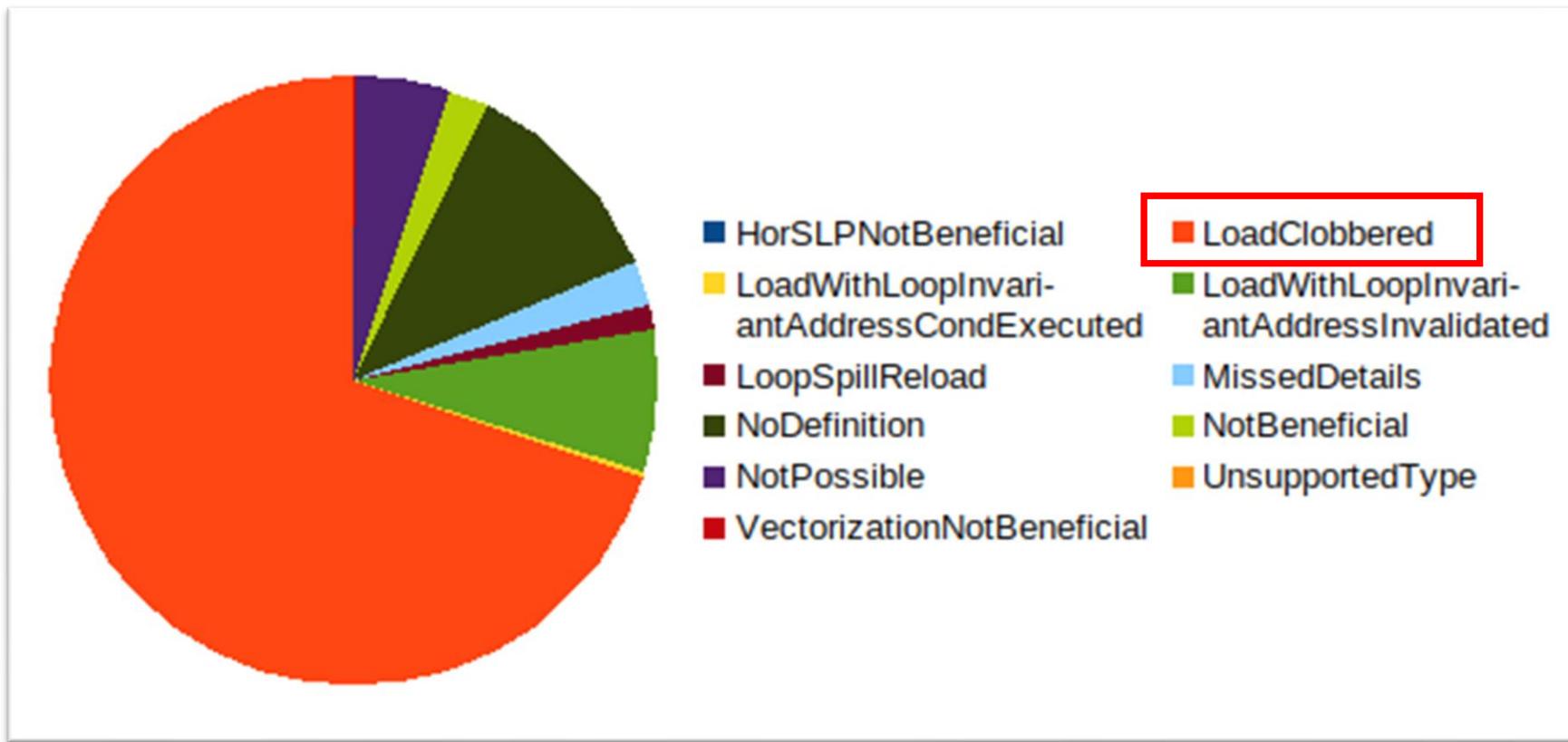
```
for (int i = 0; i < histSize; ++i)
    tileHist[i] += redistBatch;
```

- load of type i32 not eliminated because it is clobbered by store

- What if redistBatch references an element of tileHist?
 - *Aliasing*
 - Forces reloading redistBatch on every loop iteration
 - Compilers are remarkably bad at alias analysis.

Aliasing – the silent killer

- Distribution of 22K remarks in a C++ project:



Aliasing II: What can we do?

- LTO ?
- `__restrict`

 - Non standard, tells the compiler *an argument* doesn't alias with anything else

- `__attribute__((pure)), __attribute__((const))`

 - Non standard, tell the compiler *a function* doesn't write/read global state.
 - Potentially resolve 'clobbered by call' opt failures,
 - No successes seen in the wild yet.

- Strict-Aliasing

 - The only C++ standard compliant way.

Aliasing III: Types and Strict-Aliasing

- “**Strict aliasing is an assumption made by the compiler, that objects of different types will never refer to the same memory location (i.e. alias each other.)**”

Mike Acton <https://cellperformance.beyond3d.com/articles/2006/06/understanding-strict-aliasing.html>

- Allowed optimization. on by default for -O2 +
- char* is an exception – always allowed to alias with any type.
- Can be disabled with –fno-strict-aliasing (pessimization!)
- In the example above – all vars are int.

Aliasing IV: Forcing type difference

- `typedef` / `using` ?
- Inherit `int`?
 - “I considered ...to allow derivation from built-in classes ... However, I restrained myself. ... the C conversion rules are so chaotic that pretending that `int`, `short`, etc., are well-behaved ordinary classes is not going to work. They are either C compatible, or they obey the relatively well-behaved C++ rules for classes, but not both.”
Bjarne Stroustrup, The Design and Evolution of C++, §15.11.3.
- Strong-Typedefs
 - Discussion typically motivated by type safety, not optimization
 - `boost/serialization/strong_TYPEDEF.hpp` comes close.

Aliasing IV: BOOST starting point

https://www.boost.org/doc/libs/1_67_0/boost/serialization/strong_typeof.hpp

```
#define BOOST_STRONG_TYPEDEF(T, D) \
struct D : boost::totally_ordered1< D, boost::totally_ordered2< D, T> > \
{ \
    T t; \
    explicit D(const T& t_) : t(t_) {} \
    D() : t() {} \
    D(const D & t_) : t(t_.t) {} \
    D& operator=(const D& rhs) {t = rhs.t; return *this;} \
    D& operator=(const T& rhs) {t = rhs; return *this;} \
    operator const T&() const {return t;} \
    operator T&() {return t;} \
    bool operator==(const D& rhs) const {return t == rhs.t;} \
    bool operator<(const D& rhs) const {return t < rhs.t;} \
};
```

Aliasing IV: New Types From *Primitive*

```
#define CUSTOM_STRONG_TYPEDEF(T, D) \
struct D \
{ \
    T t; \
    D(const T t_) : t(t_) {}; \
    D(){}; \
    D(const D & t_) : t(t_.t){} \
    D & operator=(const D & rhs) { t = rhs.t; return *this;} \
    D & operator=(const T & rhs) { t = rhs; return *this;} \
    operator const T & () const {return t; } \
    operator T & () { return t; } \
};
```

Aliasing V: Impact

```
cv::AutoBuffer<int> _tileHist(histSize);
int* tileHist = _tileHist.data();
std::fill(tileHist, tileHist + histSize, 0);
```

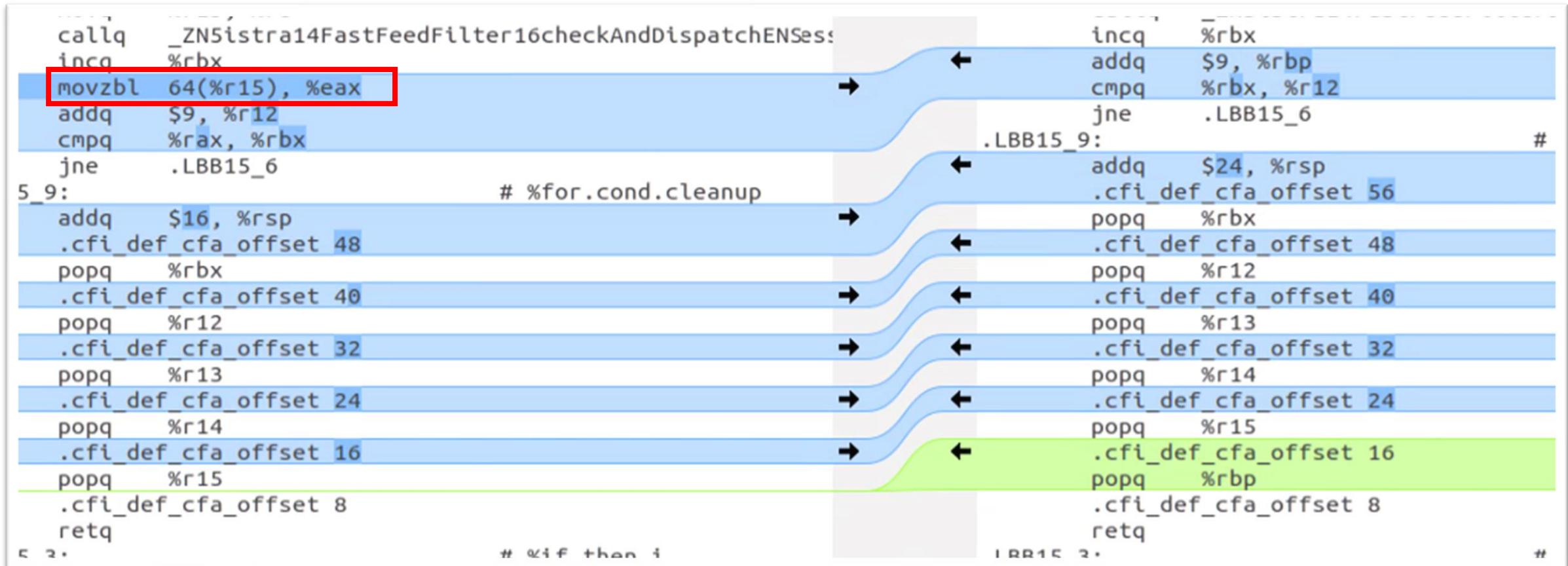
```
CUSTOM_STRONG_TYPEDEF(int, tHistDat)
cv::AutoBuffer<tHistDat> _tileHist(histSize);
tHistDat* tileHist = _tileHist.data();
std::fill(tileHist, tileHist + histSize, 0);
```

```
// redistribute clipped pixels
int redistBatch = clipped / histSize;
int residual = clipped - redistBatch * histSize;
```

```
for (int i = 0; i < histSize; ++i)
    tileHist[i] += redistBatch;
    • load of type int
    • failed to move load with loop-invariant address because the loop may invalidate its value
    ... , ... , ... , ...
```

```
for (int i = 0; i < histSize; ++i)
    tileHist[i] += redistBatch;
    • failed to move load with loop-invariant address because the loop may invalidate its value
    ... , ... , ... , ...
```

Aliasing V: Impact



Detour

Impact

```
class CV_EXPORTS Mat
{
...
    //! pointer to the data
    uchar* data;
    tDat data;
```

```
struct tDat
{
    uchar* t;
    tDat(uchar* const & t_) : t(t_) {};
    tDat() {};
    tDat(const tDat & t_) : t(t_.t){};
    tDat(tDat & t_) : t(t_.t){};
    tDat & operator=(tDat & rhs) { t = rhs.t; return *this;};
    tDat & operator=(uchar* rhs) { t = rhs; return *this;};
    operator uchar* () const { return t; }
    operator const uchar* () { return t; }
    bool operator==(uchar* rhs) const {return t == rhs;};
    bool operator==(const uchar* rhs) const {return t == rhs;};
    bool operator==(tDat& rhs) const {return t == rhs.t;};
    bool operator==(const tDat& rhs) const {return t == rhs.t;};
    bool operator==(nullptr_t) const { return t == nullptr;};
    bool operator!=(uchar* rhs) const {return t != rhs;};
    bool operator!=(tDat& rhs) const {return t != rhs.t;};
    bool operator!=(const tDat& rhs) const {return t != rhs.t;};
    uchar* operator+(long i) const { return t + i; }
    uchar* operator+=(long i) { t += i; return t;};
    uchar* operator-=(long i) { t -= i; return t;};
    uchar& operator[](long i) { return t[i]; }
    uchar operator[](long i) const { return t[i]; }
    template<typename T> operator T*() const { return (T*)(t); }
};
```

Impact

<https://github.com/opencv/opencv/wiki/HowToUsePerfTests>

```
$ cd ~/src/opencv/modules/ts/misc/  
$ python3 ./run.py ~/src/opencv-clean/build/ -w ~/logs-opencv-clean  
$ python3 ./run.py ~/src/opencv-opt/build/ -w ~/logs-opencv-opt  
$ python3 ./summary.py ~/logs-opencv-clean/core* ~/logs-opencv-opt/core*  
-u mks -m median
```

- Seems substantial.
- Will publish more detailed steps in the optview2 github page (and possibly fork opencv)

Demo 3: loop hoisting

```
// 2 - the pixel does belong to an edge  
for (int i = rowStart; i <= boundaries.end; ++i)
```

- failed to move load with loop-invariant address because the loop may invalidate its value
- load of type i32 not eliminated in favor of `load` because it is clobbered by `store`
- load of type i32 not eliminated because it is clobbered by `store`

```
if (L2gradient)
```

- failed to move load with loop-invariant address because the loop may inval

Demo 3: loop hoisting

```
const int end = boundaries.end;
  • load of type i32 not eliminated in favor of load because it is clobbered by store
const bool localL2gradient = L2gradient;
  • load of type i8 not eliminated because it is clobbered by invoke
  • load of type i8 not eliminated because it is clobbered by call
  • load of type i8 not eliminated because it is clobbered by call

for (int i = rowStart; i <= end; ++i)
  • 64 spills 1 folded spills 115 reloads 18 folded reloads generated in loop
{
    // Scroll the ring buffer

if (localL2gradient)
{
    int j = 0, width = src.cols * cn;
```

PGO integration

- “Hotness” column, can be used for sorting or filtering
- Demo
- Not necessary. If you have profiling data – use that to narrow your search.

!

... 2017 במרץ 15 · @matpow2 **Mathias Kærlev** 

@mattgodbolt Have you considered adding opt-viewer (LLVM) support to Compiler Explorer? Seems useful: github.com/androm3da/optv...

↪ ⬆️ ❤️ ↻ 1 💬

תשובות

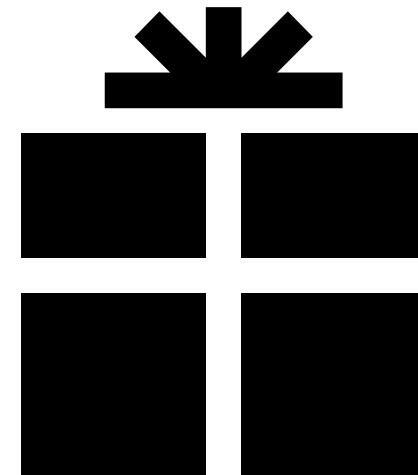
... 2017 במרץ 15 · @mattgodbolt **Matt Godbolt** 

מшиб ל- matpow2@-

not as yet no. But looks like a great idea

↪ ⬆️ 1 ❤️ ↻ 1 💬

[iuuqt ;00hpecpm/psh0{0c5t : cX Xdp](https://github.com/androm3da/optv...)



GCC work

- <https://gcc.gnu.org/legacy-ml/gcc-patches/2018-05/msg01675.html>
- <https://github.com/davidmalcolm/gcc-opt-viewer>
- YAML -> JSON
 - Actually a better choice
<https://stackoverflow.com/questions/27743711/can-i-speedup-yaml>
- Worse starting point
- Active only during 2018

[PATCH 00/10] RFC: Prototype of compiler-assisted performance analysis

- *From:* David Malcolm
- *To:* gcc-patches at gcc.gnu.org
- *Cc:* David Malcolm
- *Date:* Tue, 29 May 2018 11:25:45 +0100
- *Subject:* [PATCH 00/10] RFC: Prototype of compiler-assisted performance analysis

I want to provide more code, both for us (GCC

For us, I want to make things like how GCC can be improved (bug-fixes, etc). For end-users, I want to make it easier to see what flags to try, or how to use them.

In particular, given a command-line like this:
out:

- * "what did the optimizer do?"
- * "what *didn't* they do?"
- * "how can I make the optimizer do X?"

and as GCC developers:
* "how can we make GCC do X?"

The screenshot shows a GitHub repository page for 'davidmalcolm / gcc-opt-viewer'. The repository has 7 stars and 0 forks. It features a 'Star' button and a 'Watch' dropdown. The main navigation tabs are 'Code', 'Issues', 'Pull requests', and '...', with 'Code' being the active tab. A dropdown menu for the 'master' branch is open. The repository was created by davidmalcolm on Feb 1, 2019. The commit history shows five files: 'templates' (3 years ago), '.gitignore' (3 years ago), 'opt-viewer.py' (3 years ago), and 'optrecord.py' (3 years ago).

davidmalcolm / gcc-opt-viewer

7 stars 0 forks

Star Watch

Code Issues Pull requests ...

master ...

davidmalcolm ... on Feb 1, 2019

File	Created
templates	3 years ago
.gitignore	3 years ago
opt-viewer.py	3 years ago
optrecord.py	3 years ago

Conclusions applicable to other compilers?

- ?

Future work

- Enhance filtering,
- Consume binary optimization remarks,
- Reduce run time & memory,
- Adapt (possibly integrate) gcc opt-viewer,
- Support code annotations,
- Script code quality improvements,
- Report LLVM bugs

- Lots of help needed ☺
- <https://github.com/OfekShilon/optview2>

<https://github.com/OfekShilon/optview2>

- ofekshilon@gmail.com
- Questions?

The screenshot shows the GitHub README.md page for the optview2 repository. At the top, there is a commit card from OfekShilon. Below it is the README.md file itself, which contains the title "optview2 - User-oriented fork of LLVM's opt-viewer" and a paragraph about its history. The README.md file has a "View code" button.

OfekShilon Re-add os.path.abspath to remark.File... ... 21 hours ago ⏰ 25

View code

README.md

optview2 - User-oriented fork of LLVM's opt-viewer

In the beginning, Adam Nemet created the [opt-viewer python script family](#), as part of LLVM. He [presented it at the 2016 LLVM Developers' Meeting](#), and lo it was good.

In a nutshell ([watch the talk!](#)) it is a tool for collecting optimization remarks generated by [LLVM](#), [Clang](#), [GCC](#) and [Intel Compiler](#).

Bkp

llvm-opt-report

```
$ clang -O3 -o /tmp/v.o -c /tmp/v.c -fsave-optimization-record  
$ llvm-opt-report /tmp/v.yaml > /tmp/v.lst  
$ cat /tmp/v.lst
```

- <https://reviews.llvm.org/D25262>
- <https://github.com/llvm/llvm-project/tree/main/llvm/tools/llvm-opt-report>

```
< /tmp/v.c
  2      | void bar();
  3      | void foo() { bar(); }
  4
  5      | void Test(int *res, int *c, int *d, int *p, int n) {
  6      |   int i;
  7
  8      | #pragma clang loop vectorize(assume_safety)
  9 V4,2  |   for (i = 0; i < 1600; i++) {
10          |     res[i] = (p[i] == 0) ? res[i] : res[i] + d[i];
11          |
12
13 U16   |   for (i = 0; i < 16; i++) {
14          |     res[i] = (p[i] == 0) ? res[i] : res[i] + d[i];
15          |
16
17 I     |   foo();
18
19 I     |   foo(); bar(); foo();
20 I     |   ^
21 I     |   ^
22 I     | }
```

Name of Test	core	core	core
	clean	opt	opt
20210825-025513	20210825-035217	20210825-035217	
	vs		
	core		
	clean		
subtract::BinaryOpTest::(1920x1080, 16SC4)	3189.750	3163.450	1.01
subtractScalarDouble::BinaryOpTest::(640x480, 8UC1)	143.450	103.900	1.38
subtractScalarDouble::BinaryOpTest::(640x480, 8SC1)	91.400	86.600	1.06
subtractScalarDouble::BinaryOpTest::(640x480, 16SC1)	78.100	79.800	0.98
subtractScalarDouble::BinaryOpTest::(640x480, 32SC1)	60.800	55.900	1.09
subtractScalarDouble::BinaryOpTest::(640x480, 32FC1)	59.950	56.100	1.07
subtractScalarDouble::BinaryOpTest::(640x480, 16SC2)	170.000	162.700	1.04
subtractScalarDouble::BinaryOpTest::(640x480, 8UC3)	361.100	291.200	1.24
subtractScalarDouble::BinaryOpTest::(640x480, 16SC3)	262.750	249.200	1.05
subtractScalarDouble::BinaryOpTest::(640x480, 8UC4)	405.900	346.250	1.17
subtractScalarDouble::BinaryOpTest::(640x480, 16SC4)	338.100	333.800	1.01
subtractScalarDouble::BinaryOpTest::(1280x720, 8UC1)	300.050	265.100	1.13
subtractScalarDouble::BinaryOpTest::(1280x720, 8SC1)	343.700	262.100	1.31
subtractScalarDouble::BinaryOpTest::(1280x720, 16SC1)	319.800	233.100	1.37
subtractScalarDouble::BinaryOpTest::(1280x720, 32SC1)	189.000	196.800	0.96
subtractScalarDouble::BinaryOpTest::(1280x720, 32FC1)	187.000	196.300	0.95
subtractScalarDouble::BinaryOpTest::(1280x720, 16SC2)	630.350	536.000	1.18
subtractScalarDouble::BinaryOpTest::(1280x720, 8UC3)	925.650	821.550	1.13
subtractScalarDouble::BinaryOpTest::(1280x720, 16SC3)	942.100	921.600	1.02
subtractScalarDouble::BinaryOpTest::(1280x720, 8UC4)	1486.450	1063.900	1.40
subtractScalarDouble::BinaryOpTest::(1280x720, 16SC4)	1130.700	1259.200	0.90
subtractScalarDouble::BinaryOpTest::(1920x1080, 8UC1)	705.650	591.050	1.19
subtractScalarDouble::BinaryOpTest::(1920x1080, 8SC1)	713.450	609.800	1.17
subtractScalarDouble::BinaryOpTest::(1920x1080, 16SC1)	615.850	627.650	0.98
subtractScalarDouble::BinaryOpTest::(1920x1080, 32SC1)	762.300	759.900	1.00
subtractScalarDouble::BinaryOpTest::(1920x1080, 32FC1)	768.750	795.300	0.97
subtractScalarDouble::BinaryOpTest::(1920x1080, 16SC2)	1311.400	1327.700	0.99
subtractScalarDouble::BinaryOpTest::(1920x1080, 8UC3)	2252.000	2382.750	0.95
subtractScalarDouble::BinaryOpTest::(1920x1080, 16SC3)	2587.250	2258.700	1.15
subtractScalarDouble::BinaryOpTest::(1920x1080, 8UC4)	3004.700	3235.900	0.93
subtractScalarDouble::BinaryOpTest::(1920x1080, 16SC4)	3333.300	2812.950	1.18