BOLT: магия посткомпиляционной оптимизации бинарных файлов

Лисицын Сергей lisitsy

Компилятор

```
template <int N> void bear() {
    if (Spawning)
        bear<N - 1>();
    if constexpr (N % Step == 0)
        bear<N - Step>();
    Roar += N;
}
```

```
<bear<10000>>:
1190:   push  %rbp
1191:   mov   %rsp,%rbp
1194:   testb $0x1,0xd8e79(%rip)
119b:   je    11a2 <bear<10000>+0x12>
119d:   call  11c0 <bear<9999>>
11a2:   call  11f0 <bear<9990>>
11a7:   mov   0xd8e6b(%rip),%eax
11ad:   add   $0x2710,%eax
11b2:   mov   %eax,0xd8e60(%rip)
11b8:   pop   %rbp
11b9:   ret
```

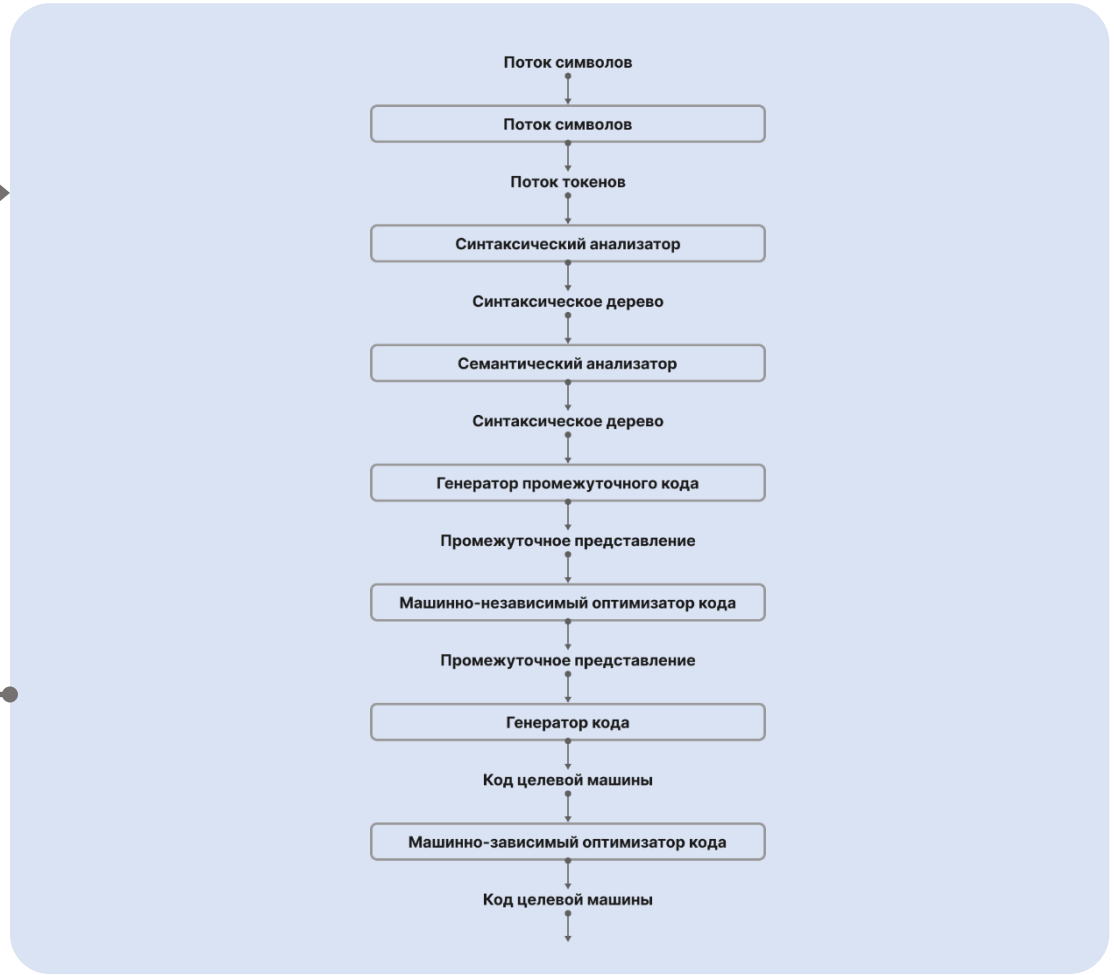
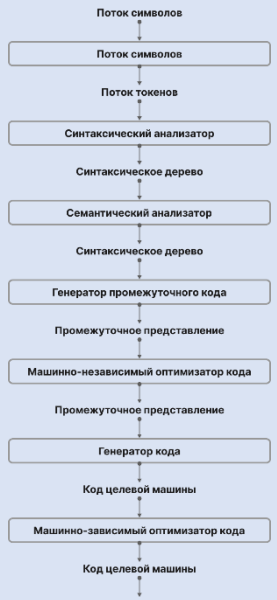


Рисунок из книги «Компиляторы: принципы, технологии и инструменты» авторов Ахо А. В., Лам М. С., Сети Р., Ульман Д. Д

Компилятор - художник

На картине изображён лесной пейзаж
с четырьмя медвежатами, играющими
на поваленном дереве в густом сосновом бору



Утро в сосновом лесу, художник Иван Иванович Шишкин, совместно с Константином Аполлоновичем Савицким, 1889 год. Находится в Государственной Третьяковской галерее, Москва.
Это изображение является точной фотографической репродукцией оригинального двумерного произведения изобразительного искусства. Данное произведение изобразительного искусства само по себе находится в общественном достоянии.

«Я художник, я так вижу»

На картине изображён лесной пейзаж
с четырьмя медвежатами, играющими
на поваленном дереве в густом сосновом бору

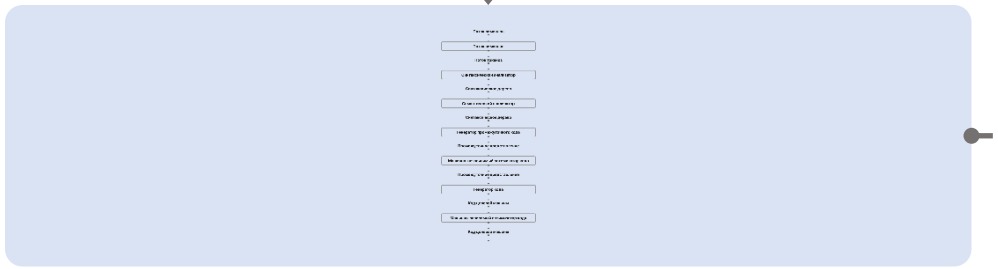
Генеративные изображения, вдохновлённые картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

Продвинутый компилятор

На картине изображён лесной пейзаж
с четырьмя медвежатами, играющими
на поваленном дереве в густом сосновом бору

The painting depicts a forest scene with four bear cubs playing on a fallen tree in a dense pine wood

这幅画描绘了一个森林场景，四只小熊正在一片密林中的倒木上玩耍



Генеративные изображения, вдохновлённые картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

Бинарный оптимизатор – реставратор

Генеративные изображения, вдохновлённые картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

Коротко об авторе

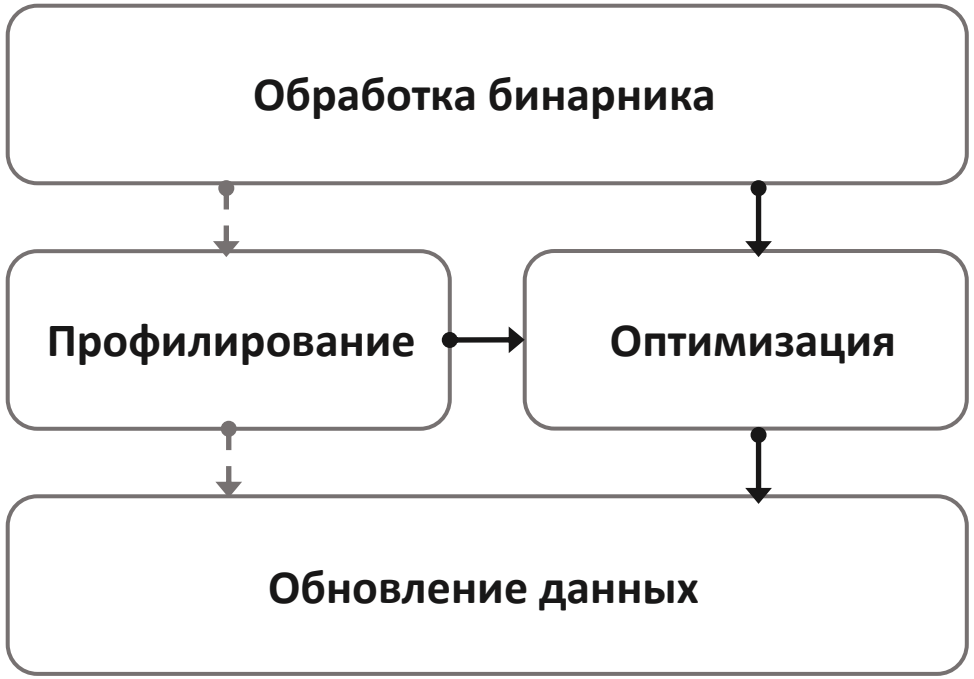
lisitsy

Разработка компиляторов в Intel и Samsung

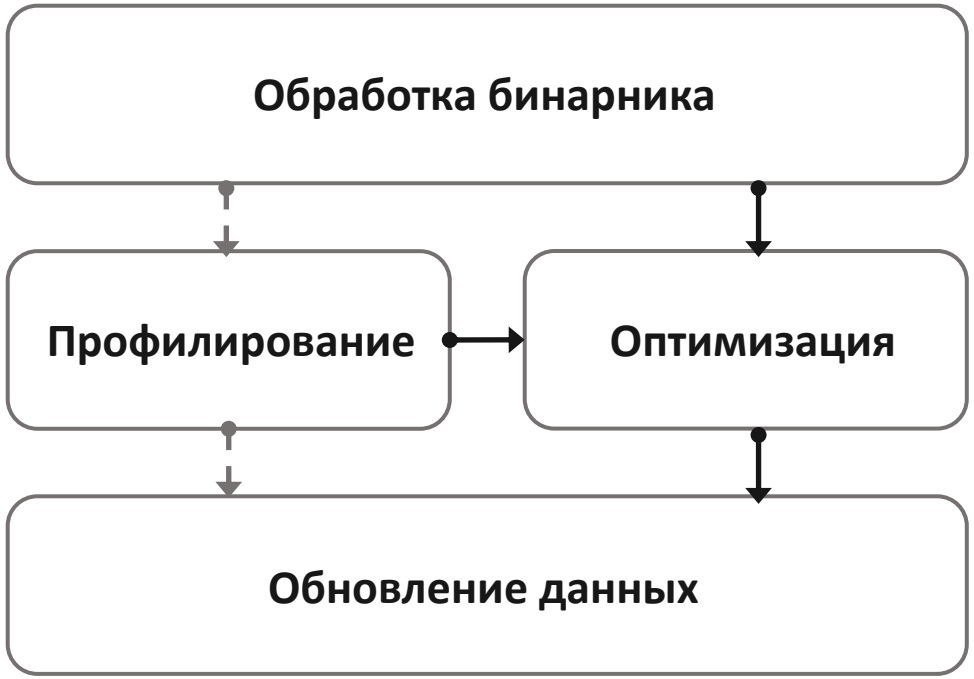
Работа над бинарным оптимизатором под ARM

Кандидатская работа по бинарному оптимизатору

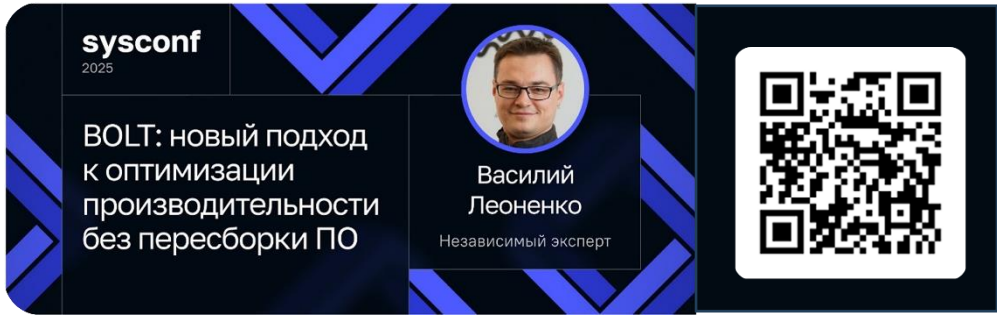
Содержание



Введение в BOLT



Введение в BOLT



BOLT – Binary Optimization and Layout Tool

Авторы (Meta*): Максим Панченко, Амир Аюпов, Рафаель Аулер

Первое применение на FB*(Meta*) продуктах в 2016

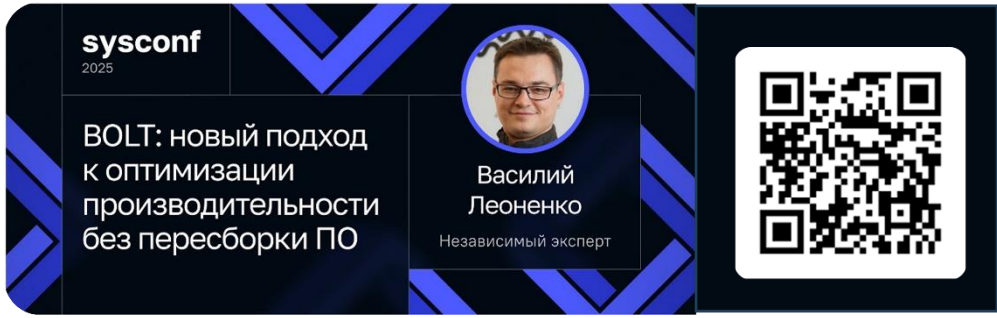
Часть LLVM с 14 релиза (Февраль 2022)

*Признана экстремистской организацией и запрещена в РФ

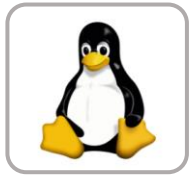
Цель — увеличить производительность приложений и библиотек без доступа к их исходному коду или IR

Подход — с помощью профильной информаци улучшает утилизацию I-cache/I-TLB, уменьшает кол-во кэш промахов и промахов переходов для больших исполняемых файлов

Эффект на реальном ПО



Clang : 8-19% поверх PGO+ThinLTO (LLVM Dev Meeting 2022)



Linux Kernel : 2.5% rockdb (Linux Plumbers 2024)

Chromium : 6% ускорение поверх PGO

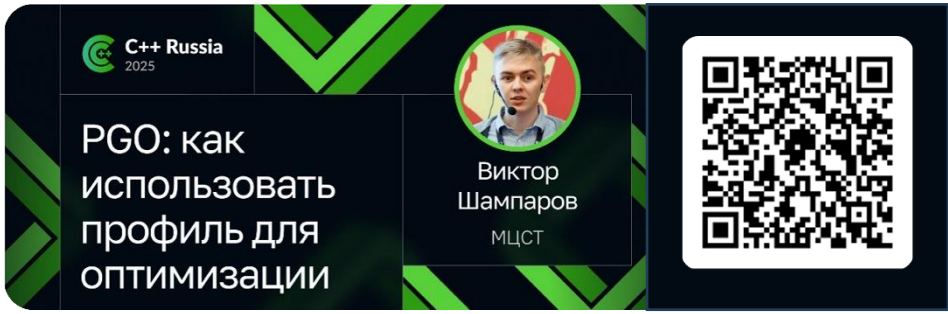
Python : 1-5% in v3.12

“For data-center applications, BOLT achieves up to 8.0% performance speedups on top of profile-guided function reordering and LTO.”

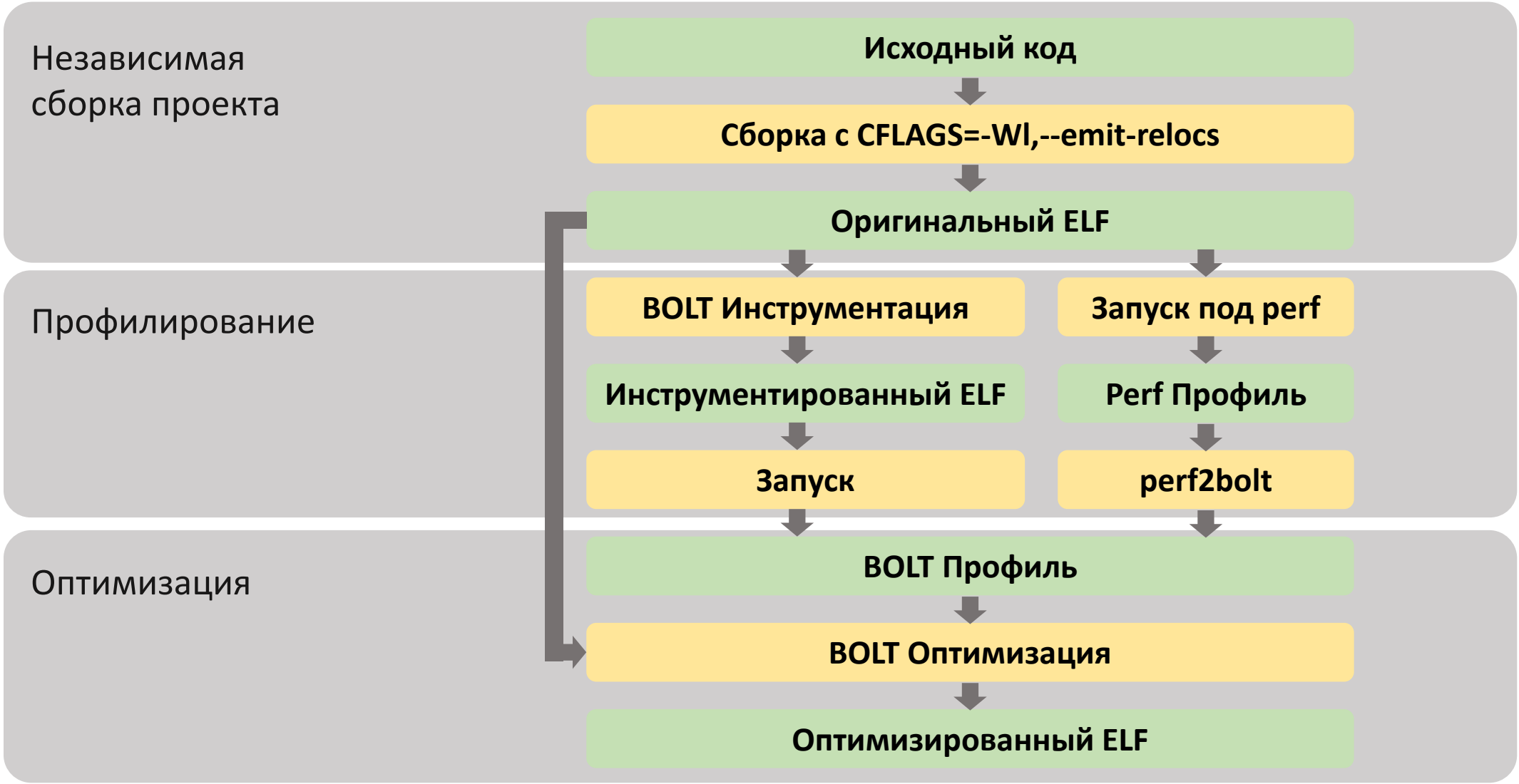
BOLT: A Practical Binary Optimizer for Data Centers and Beyond

Сравнение с PGO

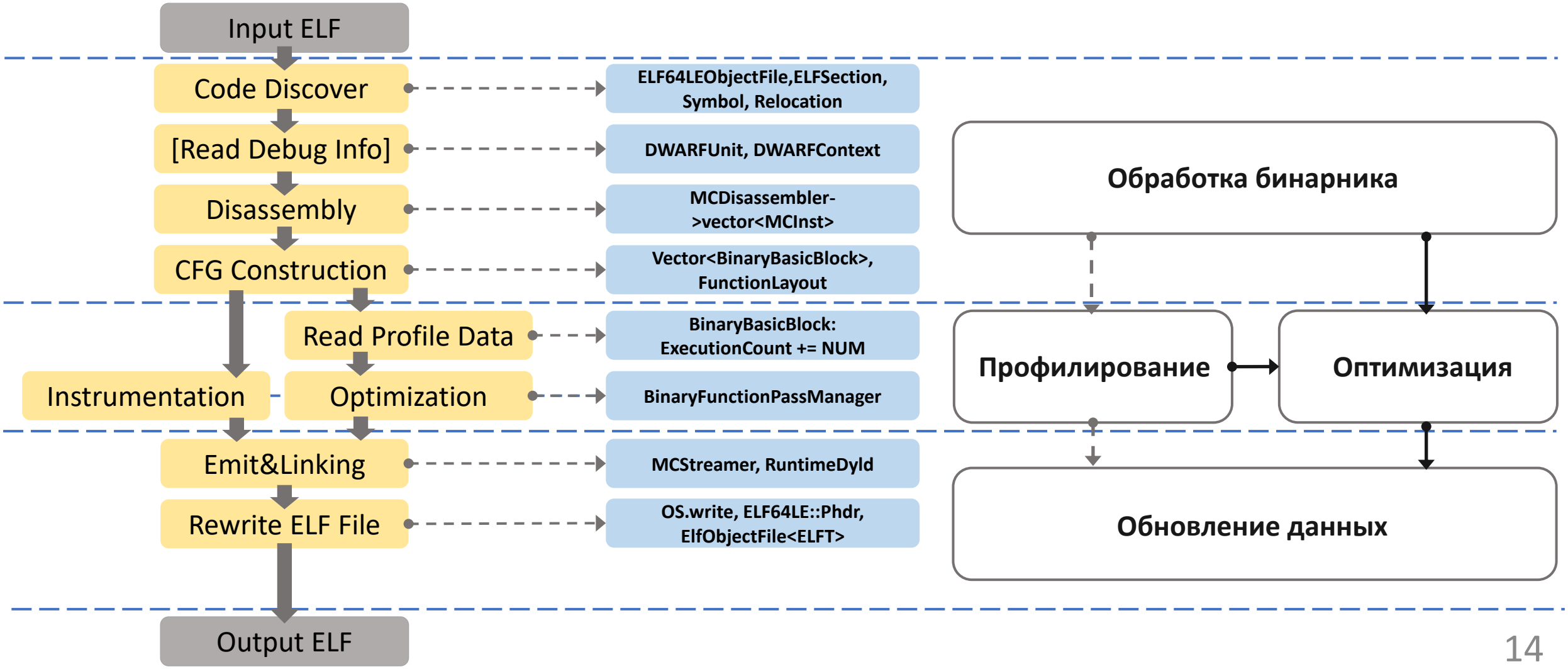
	BOLT	**PGO / LTO**
Поддерживаемые архитектуры	x86_64 / ARM64(LE) / RISC-V (начальная)	x86_64 / ARM64 / RISC-V / …
Поддерживаемые форматы	ELF / MachO	ELF / MachO / PE / COFF / Wasm
Требования к входному проекту и сборке	• Готовый исполняемый файл/библиотека • Не зависит от компилятора • Содержит символьную таблицу • Содержит статические релокации (linker –emit- relocs|-q)	• Исходный код • Сборка для инструментирования • Сборка для оптимизации
Используемый уровень представления	Инструкции (MCInst)	IR (LLVM)
Формат профиля	fdata, YAML	profraw / profdata (LLVM)



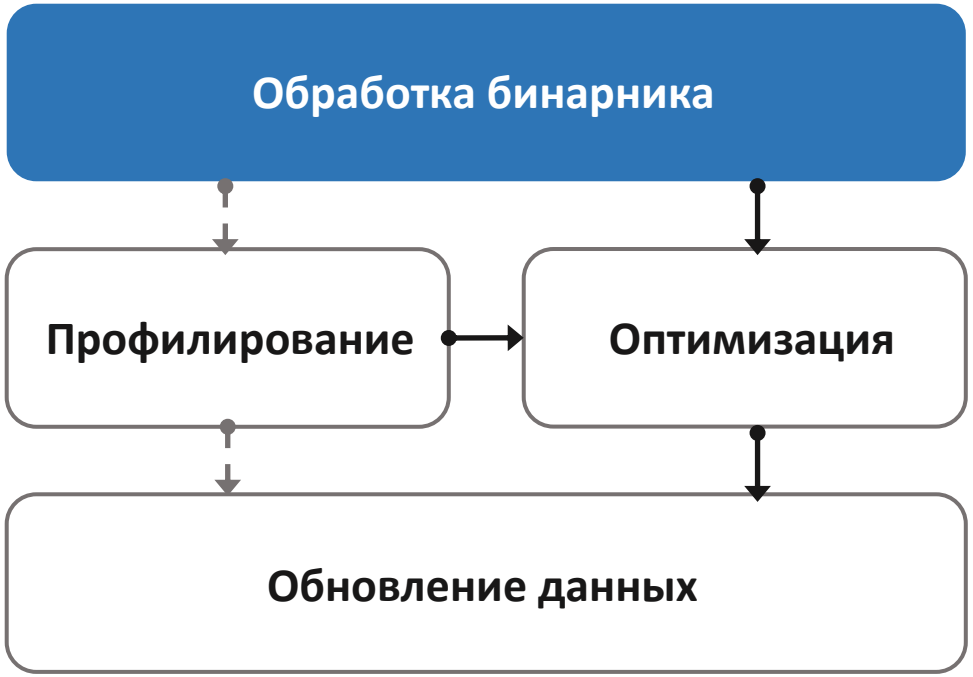
Использование BOLT



Стадии работы BOLT



Обработка бинарного файла



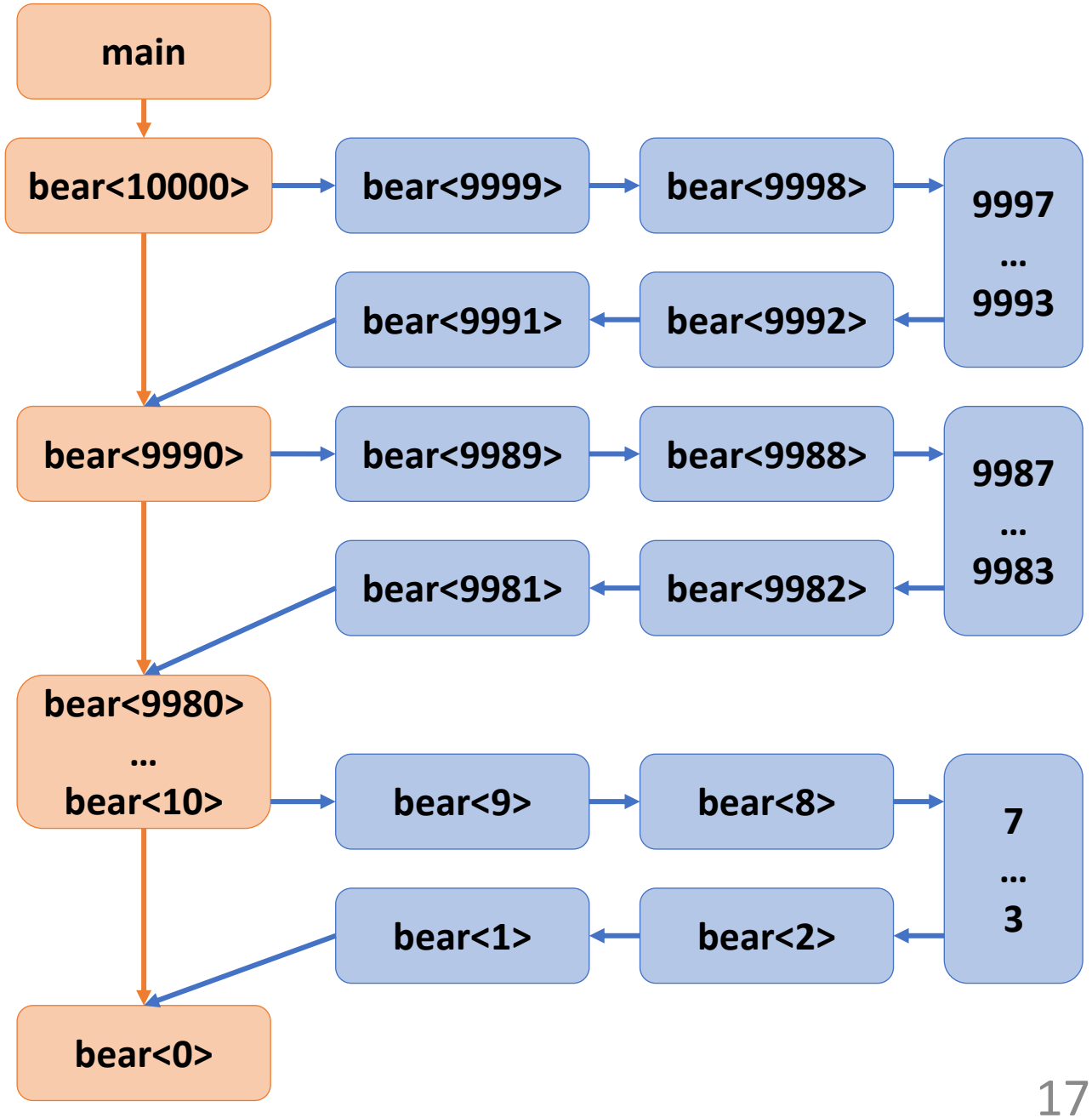
«Большой» проект

```
constexpr int Size = 10000, Step = 10;
bool Spawning = false;
int Roar = 0;

template <int N> void bear() {
    if (Spawning)
        bear<N - 1>();
    if constexpr (N % Step == 0)
        bear<N - Step>();
    Roar += N;
}

template <> void bear<0>() { Roar++; }

int main() {
    for (int i = 0; i < 1000000; i++)
        bear<Size>();
    return Roar;
}
```

Генеративное изображение, вдохновлённое картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

«Большой» проект

```
constexpr int Size = 10000, Step = 10;
bool Spawning = false;
int Roar = 0;

template <int N> void bear() {
    if (Spawning)
        bear<N - 1>();
    if constexpr (N % Step == 0)
        bear<N - Step>();
    Roar += N;
}

template <> void bear<0>() { Roar++; }

int main() {
    for (int i = 0; i < 1000000; i++)
        bear<Size>();
    return Roar;
}
```



«Большой» проект

```
constexpr int Size = 10000, Step = 10;
bool Spawning = false;
int Roar = 0;

template <int N> void bear() {
    if (Spawning)
        bear<N - 1>();
    if constexpr (N % Step == 0)
        bear<N - Step>();
    Roar += N;
}

template <> void bear<0>() { Roar++; }

int main() {
    for (int i = 0; i < 1000000; i++)
        bear<Size>();
    return Roar;
}
```

```
bear<0>
main
bear<10000>
bear<9999>
bear<9990>
bear<9998>
... (9997-9992)
bear<9991>
bear<9989>
bear<9980>
bear<9988>
... (9987-22)
bear<21>
bear<19>
bear<10>
bear<18>
... (17-12)
bear<11>
bear<9>
bear<8>
... (7-2)
bear<1>
```

Генеративное изображение, вдохновлённое картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

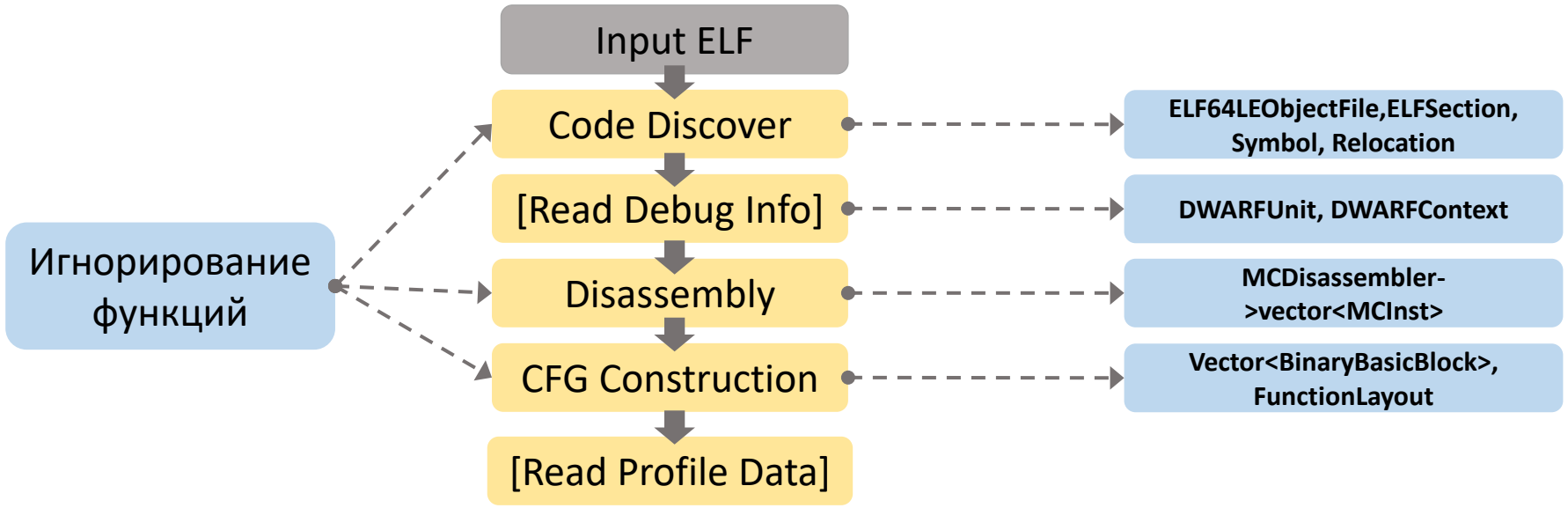
Оригинальная функция

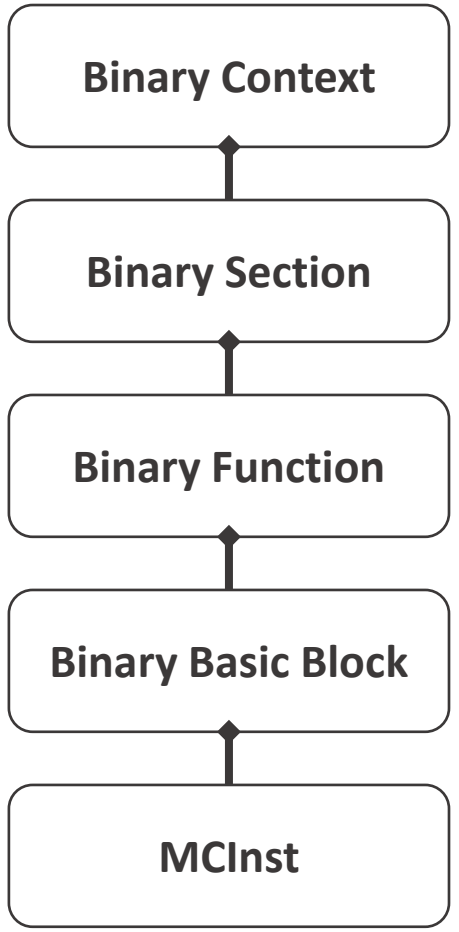
```
template <int N> void bear() {
    if (Spawning)
        bear<N - 1>();
    if constexpr (N % Step == 0)
        bear<N - Step>();
    Roar += N;
}
```

```
clang++ bears.cpp -Wl,-q -o bears
-ftemplate-depth=10000
```

```
                1190 <bear<10000>>:
template <int N> void bear() {
                1190:   push  %rbp
                1191:   mov   %rsp,%rbp
    if (Spawning)
                1194:   testb  $0x1,0xd8e79(%rip)
                119b:   je     11a2 <bear<10000>+0x12>
        bear<N - 1>();
                119d:   call   11c0 <bear<9999>>
    bear<N - Step>();
                11a2:   call   11f0 <bear<9990>>
    Roar += N;
                11a7:   mov    0xd8e6b(%rip),%eax
                11ad:   add    $0x2710,%eax
                11b2:   mov    %eax,0xd8e60(%rip)
}
                11b8:   pop    %rbp
                11b9:   ret
```

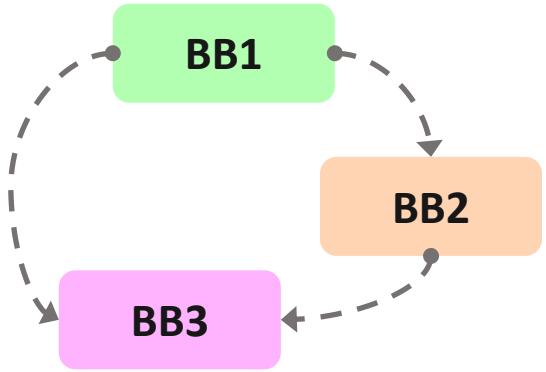
Основные фазы работы

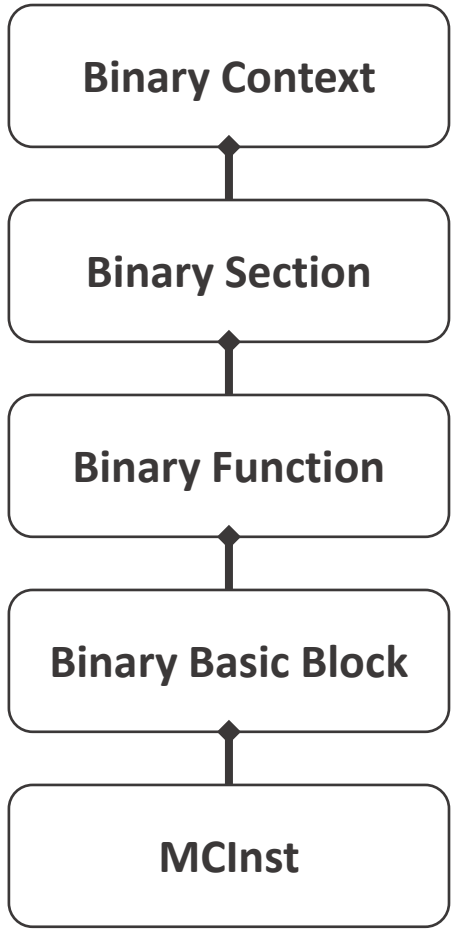




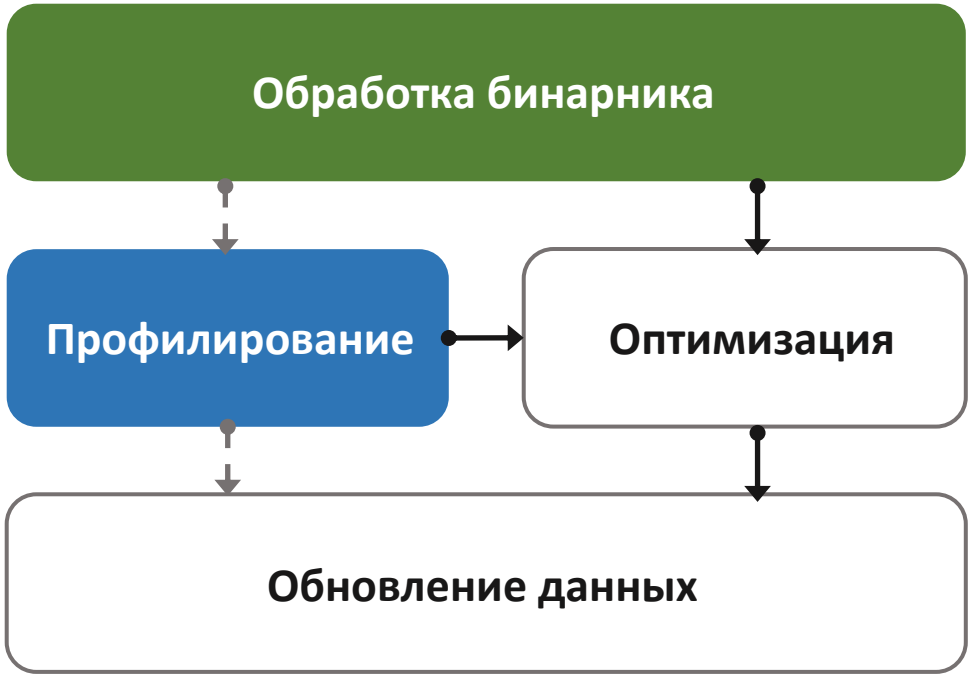
Оригинальная функция

```
<bear<10000>>:
1190:   push  %rbp
1191:   mov  %rsp,%rbp
1194:   testb $0x1,0xd8e79(%rip)
119b:   je      11a2 <bear<10000>+0x12>
119d:   call    11c0 <bear<9999>>
11a2:   call    11f0 <bear<9990>>
11a7:   mov   0xd8e6b(%rip),%eax
11ad:   add   $0x2710,%eax
11b2:   mov   %eax,0xd8e60(%rip)
11b8:   pop   %rbp
11b9:   ret
```





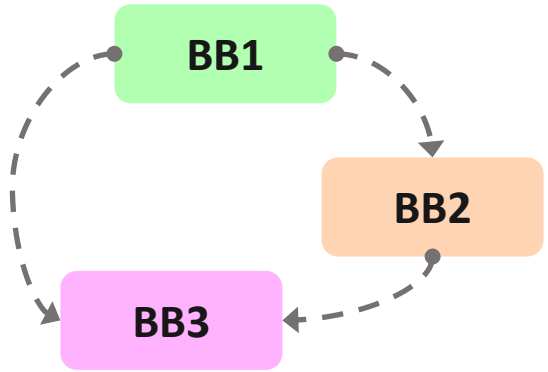
Сбор профильной информации



Профильная информация fdata

Source	Target	Branch Info
1 main 1f	1 bear<10000> 0	0 1000000
1 bear<10000> b	1 bear<10000> 12	0 1000000
1 bear<10000> 12	1 bear<9990> 0	0 1000000
1 bear<10000> b	1 bear<10000> d	0 0
1 bear<10000> d	1 bear<9990> 0	0 0

```
<bear<10000>>:
1190:  push  %rbp
1191:  mov  %rsp,%rbp
1194:  testb $0x1,0xd8e79(%rip)
119b:  je     11a2 <bear<10000>+0x12>
119d:  call   11c0 <bear<9999>>
11a2:  call   11f0 <bear<9990>>
11a7:  mov    0xd8e6b(%rip),%eax
11ad:  add    $0x2710,%eax
11b2:  mov    %eax,0xd8e60(%rip)
11b8:  pop    %rbp
11b9:  ret
```



Семплирование

PMU Sampling: perf record / script

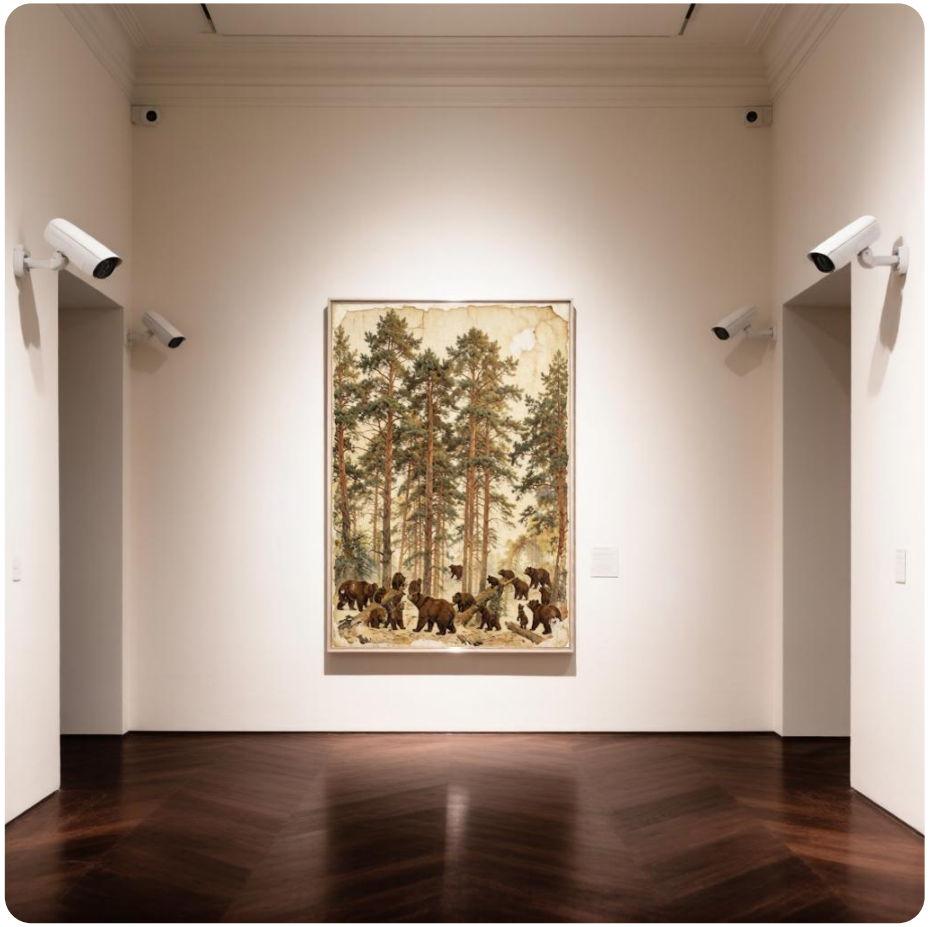
Высокая точность с Intel LBR (без эффект меньше)

Аналог для ARM: BRBE не поддерживается

Без замедления приложения

Преобразователь perf2bolt

Source	Target	Branch Info
1 main 1f	1 bear<10000> 0	0 786510
1 bear<10000> b	1 bear<10000> 12	0 683502
1 bear<10000> 12	1 bear<9990> 0	0 724580
1 bear<10000> b	1 bear<10000> d	0 0
1 bear<10000> d	1 bear<9990> 0	0 0

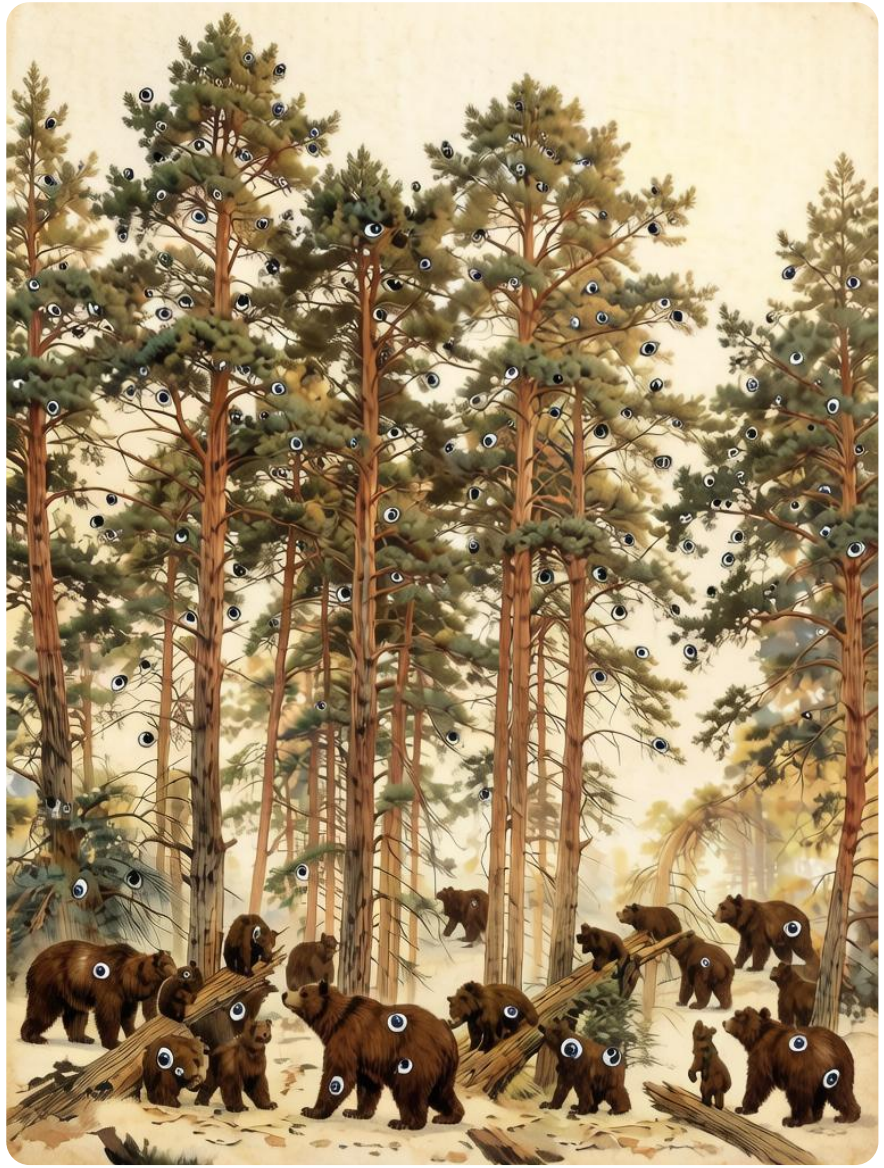


Генеративное изображение, вдохновлённое картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

Инструментация

- Специальный режим llvm-bolt --instrument
- Максимальная точность профиля
- Не требует специальных HW расширений
- Существенное замедление (x10)
- Имплементирован для X86 и ARM

Source	Target	Branch Info
1 main 1f	1 bear<10000> 0	0 786510
1 bear<10000> b	1 bear<10000> 12	0 683502
1 bear<10000> 12	1 bear<9990> 0	0 724580
1 bear<10000> b	1 bear<10000> d	0 0
1 bear<10000> d	1 bear<9990> 0	0 0



Генеративное изображение, вдохновлённое картиной «Утро в сосновом лесу» Ивана Шишкина и Константина Савицкого, 1889 год. Создано с помощью GigaChat.

Инструментированная функция

```
<bear<10000>>:
4002dc:  push  %rbp
4002dd:  mov   %rsp,%rbp
4002e0:  testb $0x1,-0x3262d3(%rip)
4002e7:  jne   400309 bear<10000>+0x2d>
```

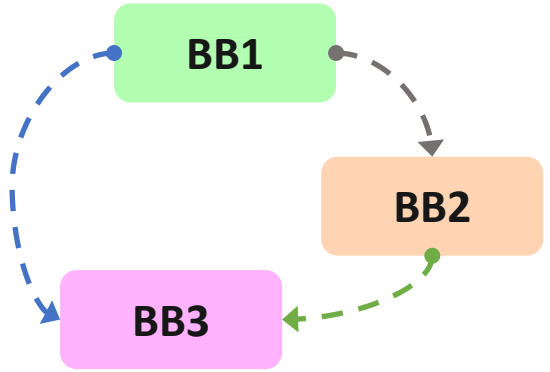
```
4002e9:  push  %rax
4002ea:  mov   $0x0,%eax
4002ef:  lahf
4002f0:  push  %rax
4002f1:  mov   $0x0,%eax
4002f6:  seto  %al
4002f9:  lock incq 0x29cd77(%rip)
400300:  add   $0x7f,%al
400304:  pop   %rax
400305:  sahf
400306:  pop   %rax
400307:  jmp   40030e <bear<10000>+0x32>
```

```
400309:  call  400344 <bear<9999>>
```

```
40030e:  push  %rax
40030f:  mov   $0x0,%eax
400314:  lahf
400315:  push  %rax
400316:  mov   $0x0,%eax
40031b:  seto  %al
40031e:  lock incq 0x29cd5a(%rip)
400325:  add   $0x7f,%al
400329:  pop   %rax
40032a:  sahf
40032b:  pop   %rax
```

```
40032c:  call  4003c0 <bear<9990>>
400331:  mov   -0x32631f(%rip),%eax
400337:  add   $0x2710,%eax
40033c:  mov   %eax,-0x32632a(%rip)
400342:  pop   %rbp
400343:  ret
```

```
push  %rax
mov   $0x0,%eax
lahf
push  %rax
mov   $0x0,%eax
seto  %al
lock incq 0x29cd77(%rip)
add   $0x7f,%al
pop   %rax
sahf
pop   %rax
```

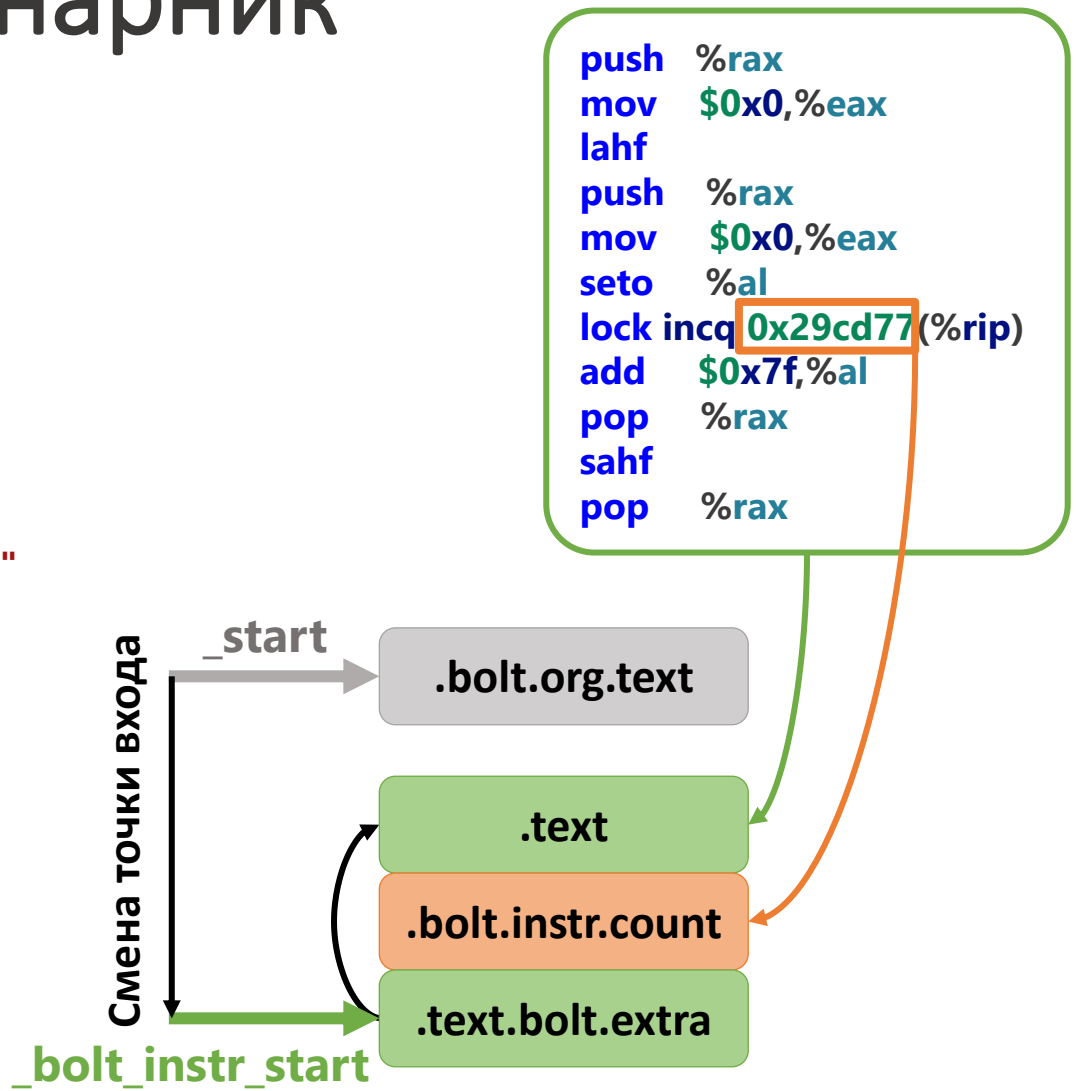



```
llvm-bolt-20 -instrument bears -o bears.instr
```

Инструментированный бинарник

```
/// This is hooking ELF's entry, it needs to save all machine state.
extern "C" __attribute((naked)) void __bolt_instr_start()
{
#if defined(__aarch64__)
    __asm__ __volatile__(SAVE_ALL
                         "bl __bolt_instr_setup\n"
                         RESTORE_ALL
                         "adrp x16, __bolt_start_trampoline\n"
                         "add x16, x16, #:lo12:__bolt_start_trampoline\n"
                         "br x16\n"
                         :::);
#else
    __asm__ __volatile__(SAVE_ALL
                         "call __bolt_instr_setup\n"
                         RESTORE_ALL
                         "jmp __bolt_start_trampoline\n"
                         :::);
#endif
}
```

bolt/runtime/instr.cpp



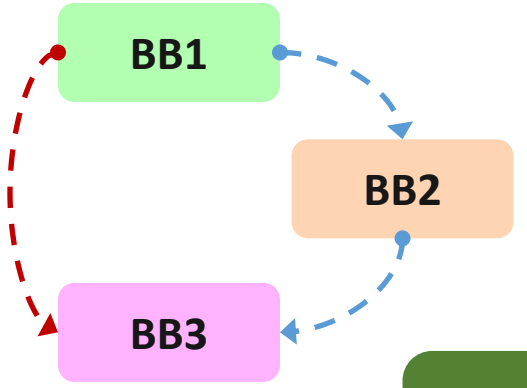
Сбор профильной информации

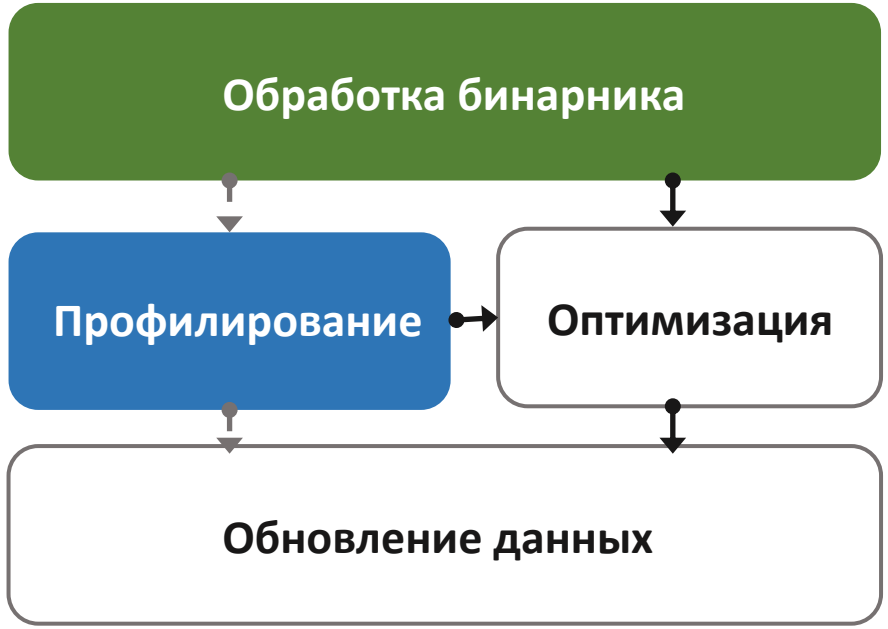
Source	Target	Branch Info
1 main 1f	1 bear<10000> 0	0 1000000
1 bear<10000> b	1 bear<10000> 12	0 1000000
1 bear<10000> 12	1 bear<9990> 0	0 1000000
1 bear<10000> b	1 bear<10000> d	0 0
1 bear<10000> d	1 bear<9990> 0	0 0

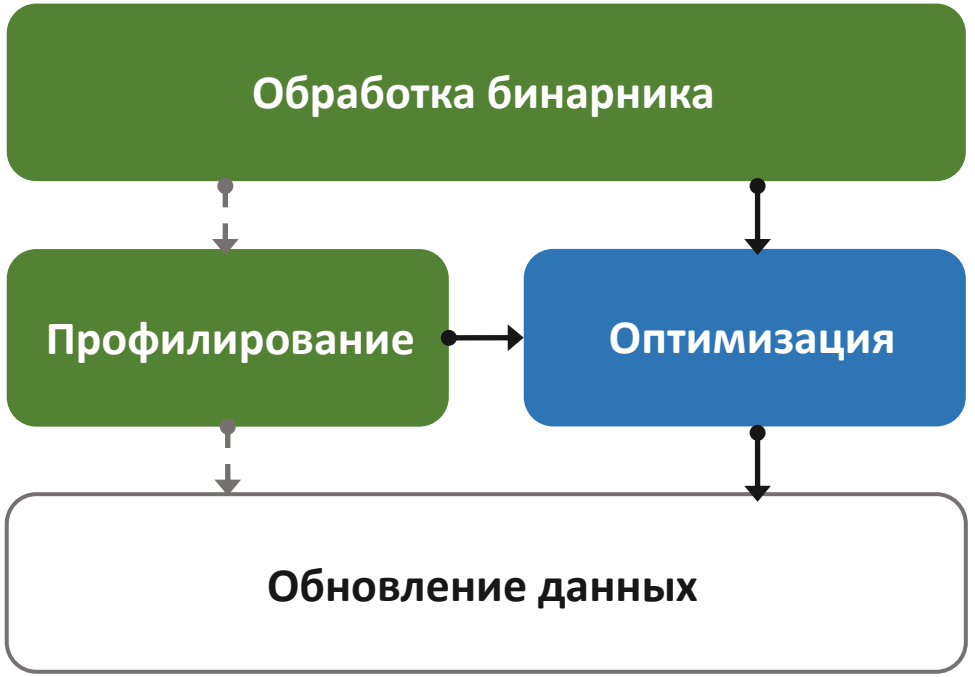
BOLT профиль

Семплирование

Инструментация







Оптимизация кода бинарного файла

Основные оптимизации

Pass Name	Description
1. strip-rep-ret	Strip `repz` from `repz retq` instructions used for legacy AMD processors
2. icf	Identical code folding
3. icp	Indirect call promotion
4. peepholes	Simple peephole optimizations
5. inline-small	Inline small functions
6. simplify-ro-loads	Fetch constant data in .rodata whose address is known statically and mutate a load into a mov
7. icf	Identical code folding (second run)
8. plt	Remove indirection from PLT calls
9. reorder-bbs	Reorder basic blocks and split hot/cold blocks into separate sections (layout optimization)
10. peepholes	Simple peephole optimizations (second run)
11. uce	Eliminate unreachable basic blocks
12. fixup-branches	Fix basic block terminator instructions to match the CFG and the current layout (redone by reorder-bbs)
13. reorder-functions	Apply HFSort [25] to reorder functions (layout optimization)
14. sctc	Simplify conditional tail calls
15. frame-opts	Removes unnecessary caller-saved register spilling
16. shrink-wrapping	Moves callee-saved register spills closer to where they are needed, if profiling data shows it is better to do so

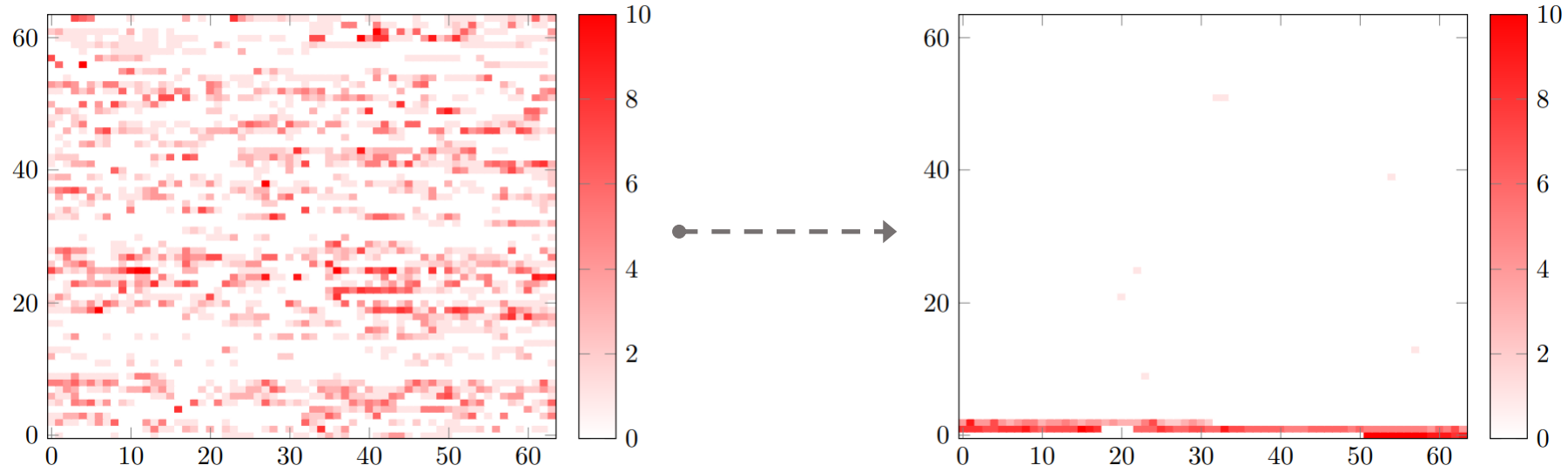
BOLT: A Practical Binary Optimizer for Data Centers and Beyond

```
template <int N> __attribute__((noinline))
void bear() { return; }

int main() {
   bear<0>();
   bear<1>();
   bear<2>();
   bear<3>();
   return 0;
}

1130 <main.org>:
1130:   push %rax
1131:   call   1150 <bear<0>>
1136:   call   1160 <bear<1>>
113b:   call   1170 <bear<2>>
1140:   call   1180 <bear<3>>
1145:   xor    %eax,%eax
1147:   pop    %rcx
1148:   ret

400008 <main.opt>:
400008:   push %rax
400009:   xor    %eax,%eax
40000b:   pop    %rcx
40000c:   ret
```

Основные оптимизации

ReorderBlocks | Split Functions | Function reordering

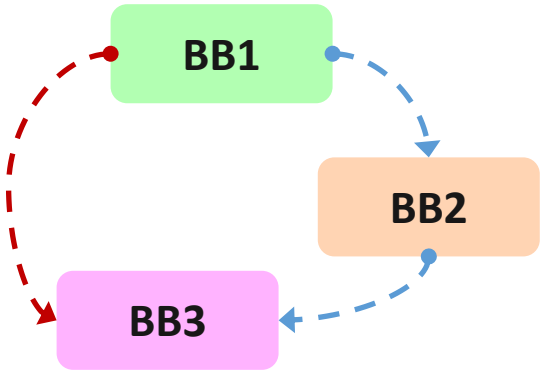
Тепловые карты доступа к коду HHVM



BOLT: A Practical Binary Optimizer for Data Centers and Beyond

Оптимизированная функция

```
<bear<10000>>:
40e300:   push   %rbp
40e301:   mov    %rsp,%rbp
40e304:   testb  $0x1,-0x3342f7(%rip)
40e30b:   jne    41214a <bear<10000>.cold.0>
40e311:   call   40e340 <bear<9990>>
40e316:   mov    -0x334304(%rip),%eax
40e31c:   add    $0x2710,%eax
40e321:   mov    %eax,-0x33430f(%rip)
40e327:   pop    %rbp
40e328:   ret

<bear<10000>.cold.0>:
41214a:   call   11c0 <bear<9999>>
41214f:   jmp    40e311 <bear<10000>+0x11>
```



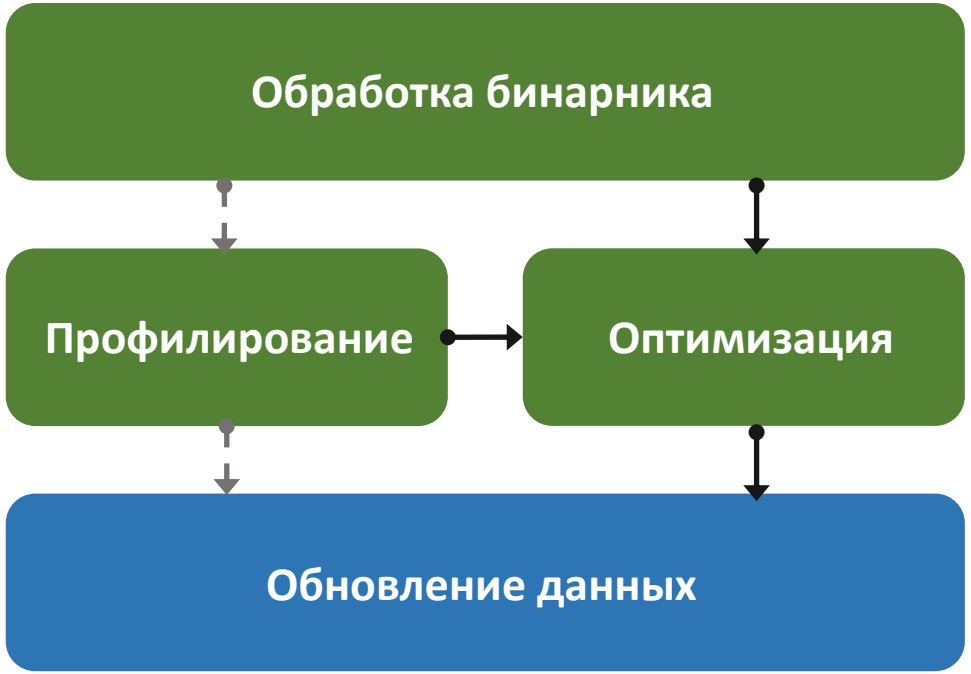

```
llvm-bolt-20 bears -o bears.opt -data prof.fdata
-reorder-blocks=ext-tsp -reorder-functions=cdsort
-split-functions -split-all-cold -split-eh -dyno-stats
```

Оптимизированный проект

.bolt.org.text
bear<0>
main
bear<10000>
bear<9999>
bear<9990>
bear<9998>
... (9997-9992)
bear<9991>
bear<9989>
bear<9980>
bear<9988>
... (9987-22)
bear<21>
bear<19>
bear<10>
bear<18>
... (17-12)
bear<11>
bear<9>
bear<8>
... (7-2)
bear<1>

.text
bear<0>
main
bear<10000>
bear<9990>
...
bear<20>
bear<10>

.text.cold
bear<10000>.cold
bear<9990>.cold
...
bear<20>.cold
bear<10>.cold



Обновление данных бинарного файла

Переписывание заголовков

PHDR (-use-gnu-stack)

Измененные и новые секции

Измененные и новые сегменты

```
Entry point address:               0x1040
Start of program headers:          64 (bytes into file)
Start of section headers:          2544888 (bytes into file)
Flags:                             0x0
Size of this header:               64 (bytes)
Size of program headers:           56 (bytes)
Number of program headers:         14
Size of section headers:           64 (bytes)
Number of section headers:         39
Section header string table index: 38
```

```
Entry point address:               0x6c99a0
Start of program headers:          2097152 (bytes into file)
Start of section headers:          9665936 (bytes into file)
Flags:                             0x0
Size of this header:               64 (bytes)
Size of program headers:           56 (bytes)
Number of program headers:         16
Size of section headers:           64 (bytes)
Number of section headers:         47
Section header string table index: 44
```

Отладочная информация

Не обновляется по умолчанию:
-update-debug-section

Генерация таблиц:
gdb_index, debug_names

DWARF5 поддержан

Разбиение DWARF поддержано

Поддержка исключений C++

```
.LBB07 (11 instructions, align : 1)
  Entry Point
  Exec Count : 104
  CFI State : 0
    00000000:   pushq   %rbp # exception4.cpp:22
    00000001:   !CFI    $0      ; OpDefCfaOffset -16
    00000001:   !CFI    $1      ; OpOffset Reg6 -16
    00000001:   movq    %rsp, %rbp # exception4.cpp:22
    00000004:   !CFI    $2      ; OpDefCfaRegister Reg6
    00000004:   subq    $0x10, %rsp # exception4.cpp:22
    00000008:   movl    %edi, -0x4(%rbp) # exception4.cpp:22
    0000000b:   movl    -0x4(%rbp), %eax # exception4.cpp:23
    0000000e:   movl    %eax, %edi # exception4.cpp:23
    00000010:   callq   _Z3fooi # handler: .LLP0; action: 1
                               # exception4.cpp:23
    00000015:   jmp     .Ltmp9 # exception4.cpp:24
  Successors: .Ltmp9 (mispreds: 0, count: 100)
  Landing Pads: .LLP0 (count: 4)
  CFI State: 3
.LLP0 (2 instructions, align : 1)
  Landing Pad
  Exec Count : 4
  CFI State : 3
  Throwers: .LBB07
    00000017:   cmpq    $-0x1, %rdx # exception4.cpp:24
    0000001b:   je      .Ltmp10 # exception4.cpp:24
  Successors: .Ltmp10 (mispreds: 0, count: 4),
              .LFT8 (inferred count: 0)
  CFI State: 3
```

Section Headers:

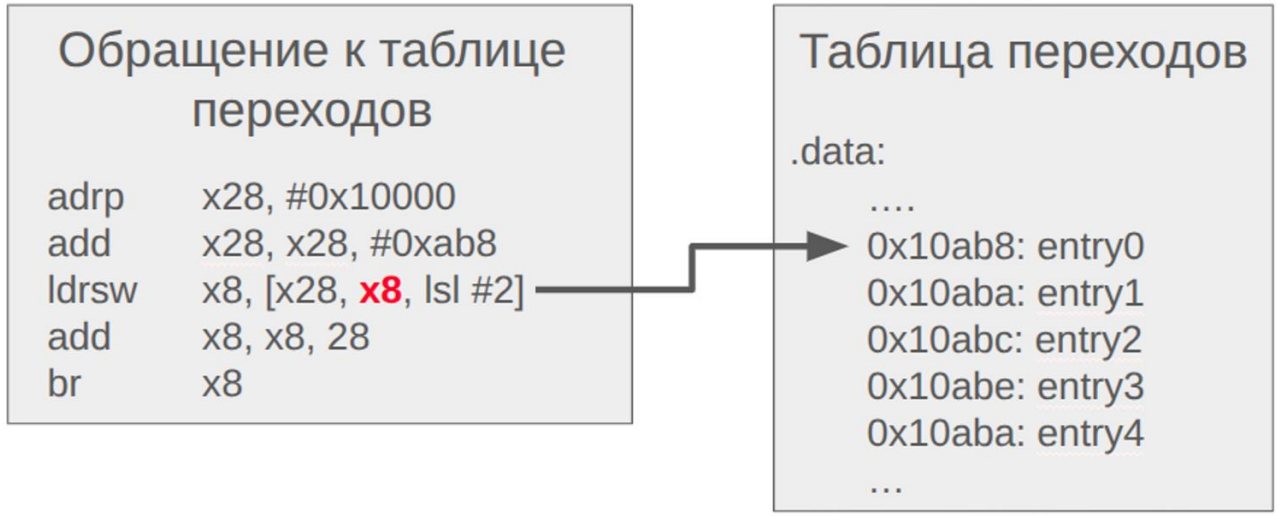
...
[13] .bolt.org.text
...
[16] .bolt.org.eh_frame_hdr
[17] .bolt.org.eh_frame
...
[28] .text
[29] .text.cold
[30] .eh_frame
[31] .eh_frame_hdr
...

Таблицы переходов

```
switch (argc) {
    case 0:
        bear<0>();
        break;
     case 1:
        bear<1>();
        break;
    case 2:
        bear<2>();
        break;
    case 3:
        bear<3>();
        break;
}
```

```
lea     0xea6(%rip),%rcx    # 2004
movslq  (%rcx,%rax,4),%rax
add     %rcx,%rax
jmp     *%rax
```

[17] .rodata



Указатели на функции

```
int main() {
   if (bear<0> < bear<1>)
      for (int i = 0; i < 1000000; i++)
         bear<Size>();
   return Roar;
}
```

```
1150 <main.org>:
115f:   lea   -0x36(%rip),%rax # 1130 <bear<0>>
1166:   lea   0x33(%rip),%rcx  # 11a0 <bear<1>>
116d:   cmp %rcx,%rax
1170:   jae   1194 <main+0x44>

40e2c0 <main.opt>:
40e2cf:   lea   -0xcc2e(%rip),%rax     # 4016a8 <bear<0>>
40e2d6:  lea   -0x40d13d(%rip),%rcx # 11a0 <bear<1>>
40e2dd:  cmp %rcx,%rax
40e2e0:   jae   40e302 <main+0x42>
```

.bolt.org.text

bear<0>
main
bear<10000>
bear<9999>
bear<9990>
bear<9998>
... (9997-9992)
bear<9991>
bear<9989>
bear<9980>
bear<9988>
... (9987-22)
bear<21>
bear<19>
bear<10>
bear<18>
... (17-12)
bear<11>
bear<9>
bear<8>
... (7-2)
bear<1>

.text

bear<0>
main
bear<10000>
bear<9990>
...
bear<20>
bear<10>

.text.cold

bear<10000>.cold
bear<9990>.cold
...
bear<20>.cold
bear<10>.cold

Известные проблемы

Входной файл	Выходной файл	Общие проблемы
Сложные PC-relative паттерны	Существенное увеличение размера выходного файла (до ~x2)	Существенное потребление памяти BOLT-ом
Нехватка восстановленной информации	Дополнительные сегменты в ELF файле	Стоимость ошибки
Защита от модификации		

Заключение

0. Подготовить BOLT

0.a Собрать из llvm-project

```
cmake ... -DLLVM_ENABLE_PROJECTS="bolt"
```

0.b Установить

```
sudo apt install libbolt-20-dev bolt-20
```

1. Собрать целевое приложение

Добавить --emit-relocs/-q для линкера
Не удалять символы и релокации (без strip/objcopy)

2. Собрать профиль

2.a Инструментация с BOLT

```
llvm-bolt-20 -instrument <APP> -o <APP>.instr
<APP>.instr <args>
```

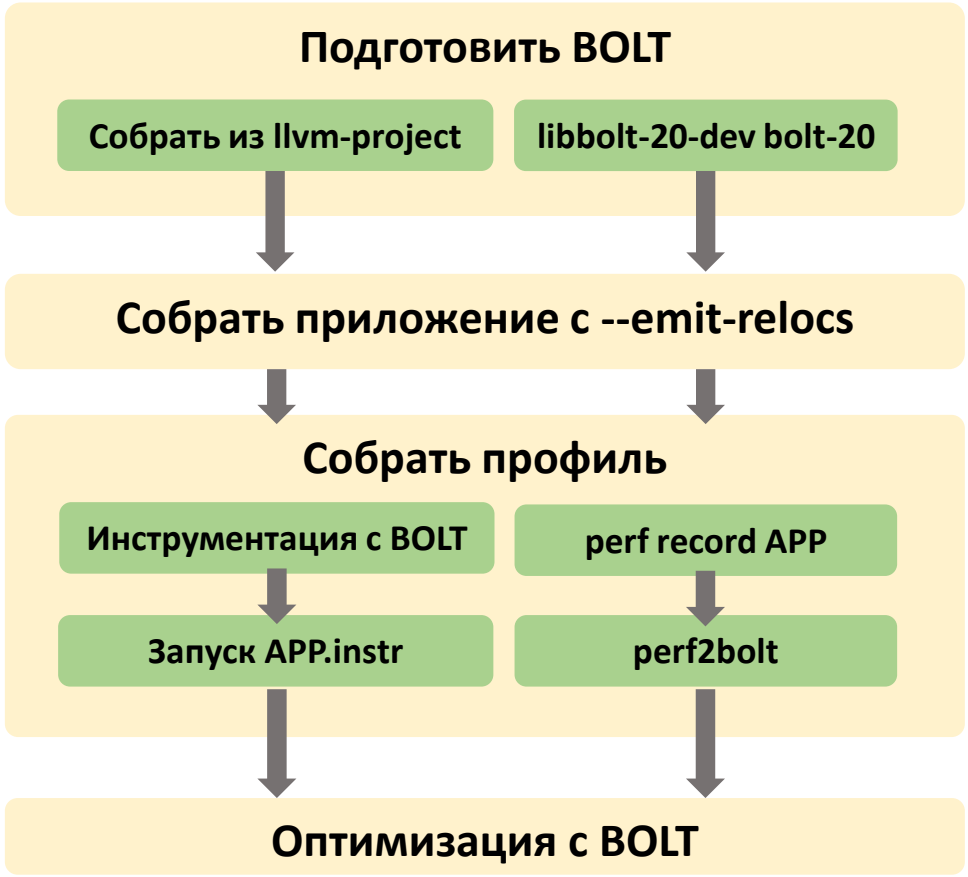
2.b Семплирование с perf

```
perf record -e cycles:u -j any,u -a -o perf.data -- <APP> <args>
perf2bolt -p perf.data -o perf.fdata <APP>
```

3. Оптимизация приложения с BOLT

```
llvm-bolt-20 <APP> -o <APP>.opt -data prof.fdata -reorder-blocks=ext-tsp  -
reorder-functions=cdsort -split-functions -split-all-cold -split-eh -dyno-stats
```

Готово!



Q&A

BOLT: магия посткомпиляционной оптимизации бинарных файлов

Лисицын Сергей lisitsy