

VOIT: новый подход к оптимизации производительности без пересборки ПО

Василий Леоненко

  @vleonen

Содержание

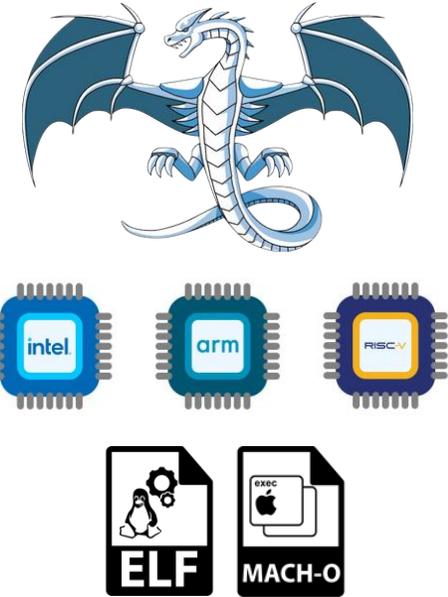
BOLT

- Введение
- Основные фазы работы
- Основные оптимизации
- Как использовать
- Оптимизация приложений с BOLT vs PGO+LTO
- Эффект на реальном ПО
- Известные проблемы
- Улучшения

Введение

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)
- Цели и подход:
 - Увеличить производительность приложений и библиотек без доступа к их исходному коду или IR
 - Улучшает утилизацию I-cache / I-TLB , уменьшает кол-во шэш промахов и промахов переходов для больших исполняемых файлов



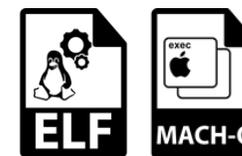
Введение

	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



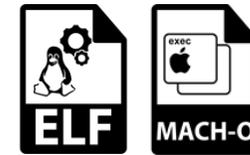
Введение

	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	<ul style="list-style-type: none">• Готовый исполняемый файл/библиотека• Не зависит от компилятора• Содержит символьную таблицу• Содержит статические релокации (linker <code>-emit-relocs</code>)	<ul style="list-style-type: none">• Исходный код• Сборка для инструментирования• Сборка для оптимизации



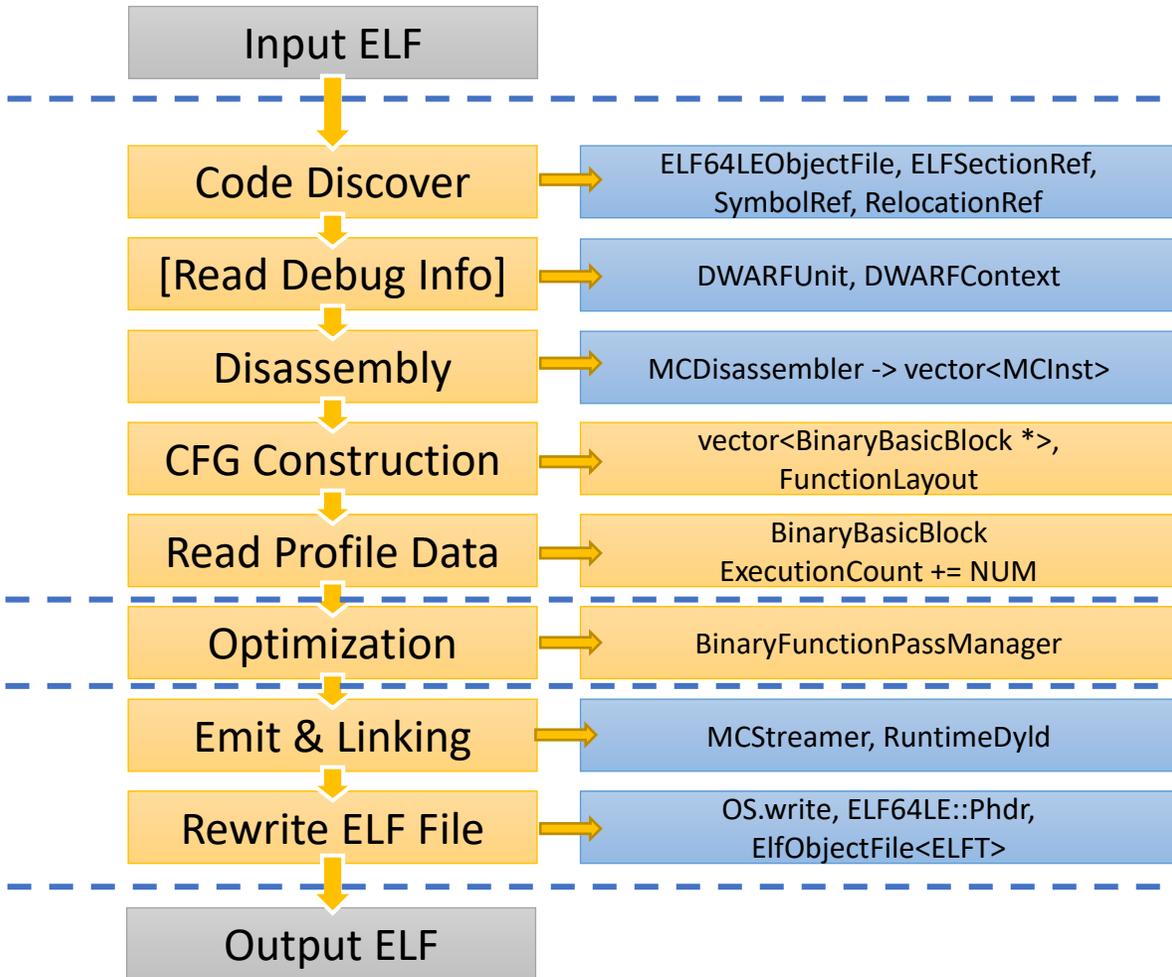
Введение

	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	<ul style="list-style-type: none">• Готовый исполняемый файл/библиотека• Не зависит от компилятора• Содержит символьную таблицу• Содержит статические релокации (linker <code>-emit-relocs</code>)	<ul style="list-style-type: none">• Исходный код• Сборка для инструментирования• Сборка для оптимизации
Используемый уровень представления	Инструкции (MCInst)	IR (LLVM)
Формат профиля	fdata, YAML	profrac / profdata (LLVM)

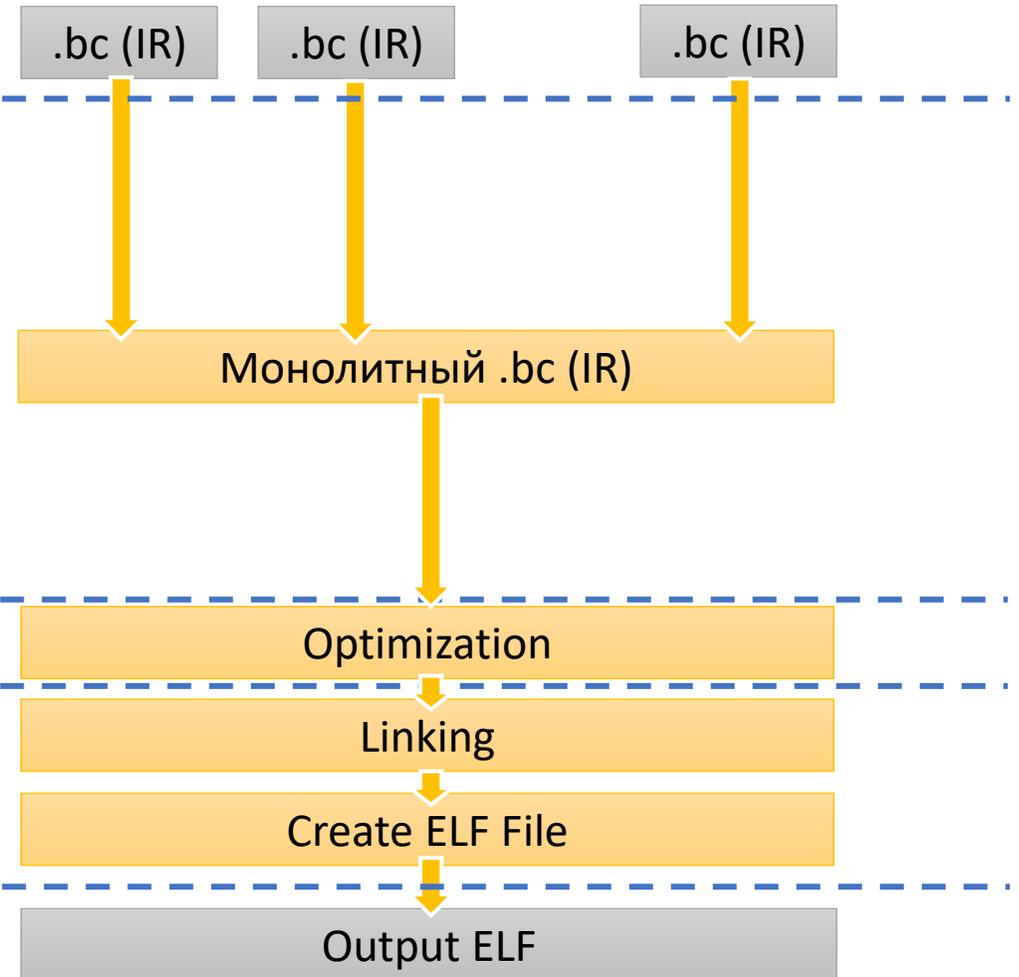


Основные фазы работы

BOLT



LTO



Основные оптимизации

Оптимизационные проходы:

- ReorderBlocks (cache/cache+/ext-tsp/branch-predictor)

```
void
__attribute__((noinline))
foo(int a, int b) {
    if (a < b) {
        bar(a);
    } else {
        baz(a);
    }
    return;
}
```

1	foo	20	1	foo	34	0	10
1	foo	24	1	foo	28	0	1
1	foo	30	1	foo	40	0	1
1	foo	3c	1	foo	40	0	10
1	foo	2c	1	bar	0	0	1
1	foo	38	1	baz	0	0	10

```
<+0>:  sub    sp, sp, #0x20
<+4>:  stp    x29, x30, [sp, #16]
<+8>:  add    x29, sp, #0x10
<+12>: stur   w0, [x29, #-4]
<+16>: str    w1, [sp, #8]
<+20>: ldur   w8, [x29, #-4]
<+24>: ldr    w9, [sp, #8]
<+28>: subs   w8, w8, w9
<+32>: b.ge   0x7b8 <foo+52> // b.tcont
<+36>: b      0x7ac <foo+40>
<+40>: ldur   w0, [x29, #-4]
<+44>: bl     0x754 <bar>
<+48>: b      0x7c4 <foo+64>
<+52>: ldur   w0, [x29, #-4]
<+56>: bl     0x76c <baz>
<+60>: b      0x7c4 <foo+64>
<+64>: ldp    x29, x30, [sp, #16]
<+68>: add    sp, sp, #0x20
<+72>: ret
```

```
<+0>:  sub    sp, sp, #0x20
<+4>:  stp    x29, x30, [sp, #16]
<+8>:  add    x29, sp, #0x10
<+12>: stur   w0, [x29, #-4]
<+16>: str    w1, [sp, #8]
<+20>: ldur   w8, [x29, #-4]
<+24>: ldr    w9, [sp, #8]
<+28>: subs   w8, w8, w9
<+32>: b.lt   0x30438 <foo+56> // b.tstop
<+36>: ldur   w0, [x29, #-4]
<+40>: bl     0x303e4 <baz>
<+44>: ldp    x29, x30, [sp, #16]
<+48>: add    sp, sp, #0x20
<+52>: ret
<+56>: ldur   w0, [x29, #-4]
<+60>: bl     0x303cc <bar>
<+64>: b      0x3042c <foo+44>
```

Основные оптимизации

Оптимизационные проходы:

- ReorderBlocks (cache/cache+/ext-tsp/branch-predictor)
- ReorderFunctions (hfsort/hfsort+/pettis-hansen/cdsort)
- SplitFunctions
- ICF (Identical Code Folding)
- ICP (Indirect Call Promotion)
- Inline
- EliminateUnreachableBlocks (Dead Code Elimination)

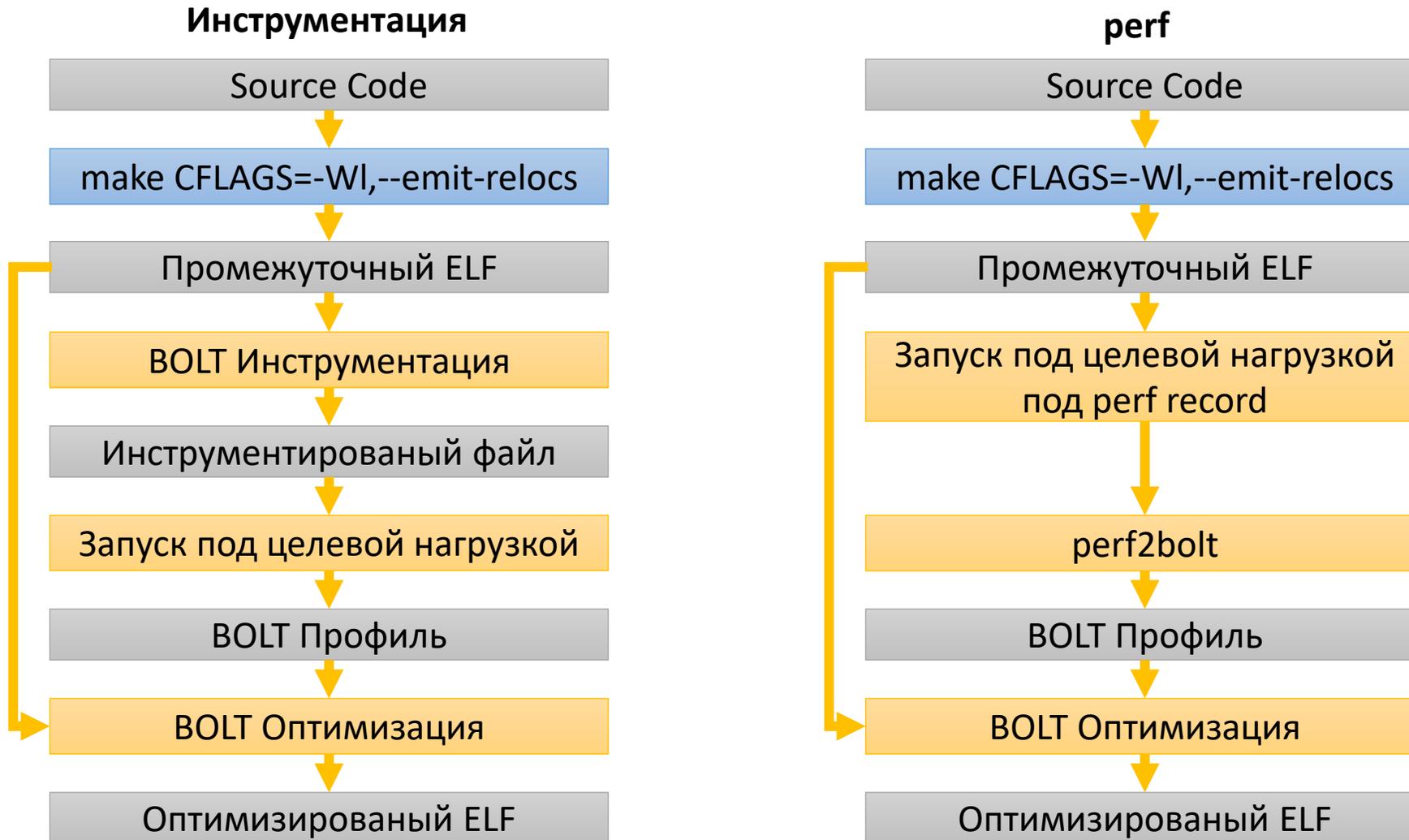
Сервисные проходы:

- Instrumentation
- LongJump

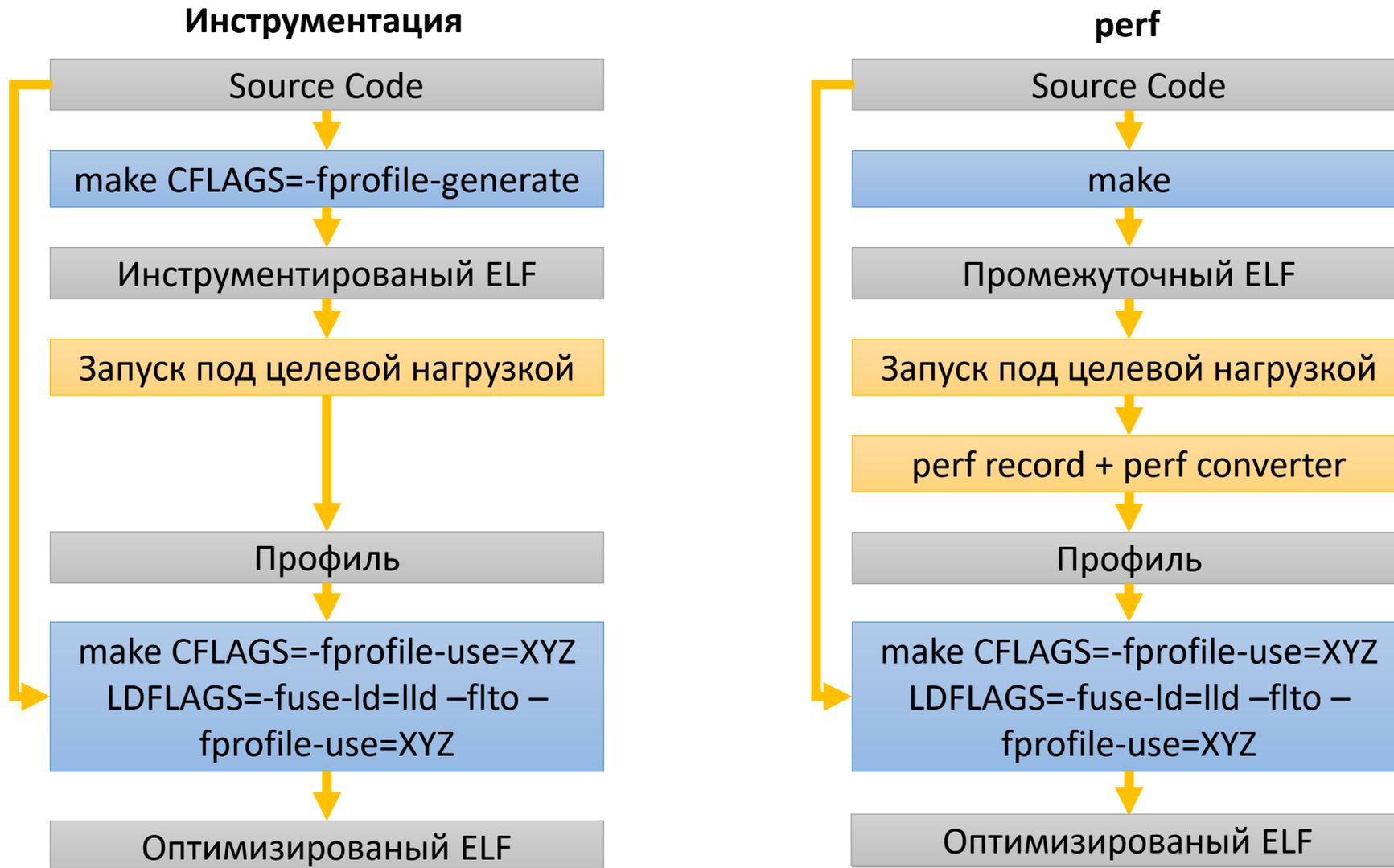
Сбор профиля исполнения

	Инструментация	PMU Sampling
	<code>bolt --instrument</code>	<code>perf record / script</code>
Точность профиля	Максимальная	Высокая с Intel LBR Аналог для ARM64: BRBE не поддерживается Без LBR эффект меньше
Замедление при сборе профиля	Существенное (x10)	Приемлемое (<10%)
Поддержка	X86_64 & ARM64	X86 (LBR)

Оптимизация приложений с BOLT

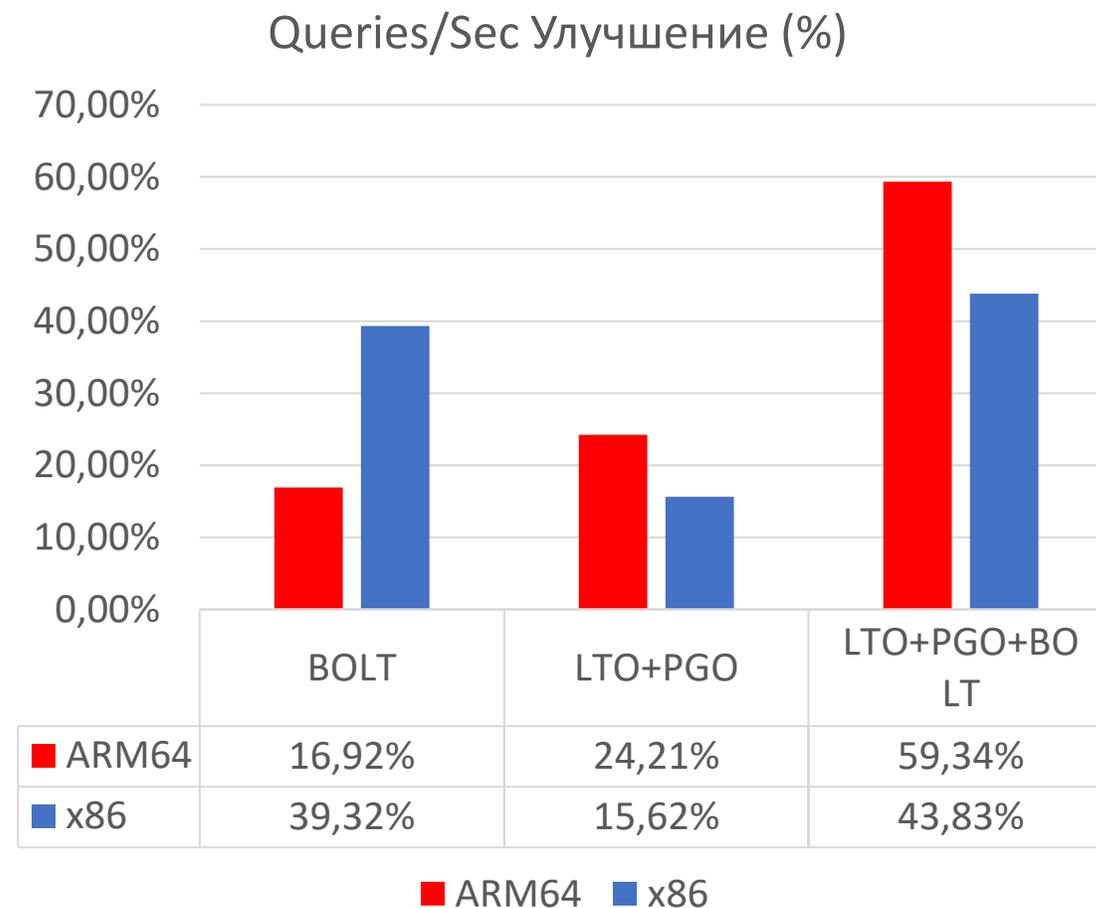


Оптимизация приложений с PGO+LTO



Эффект на реальном ПО

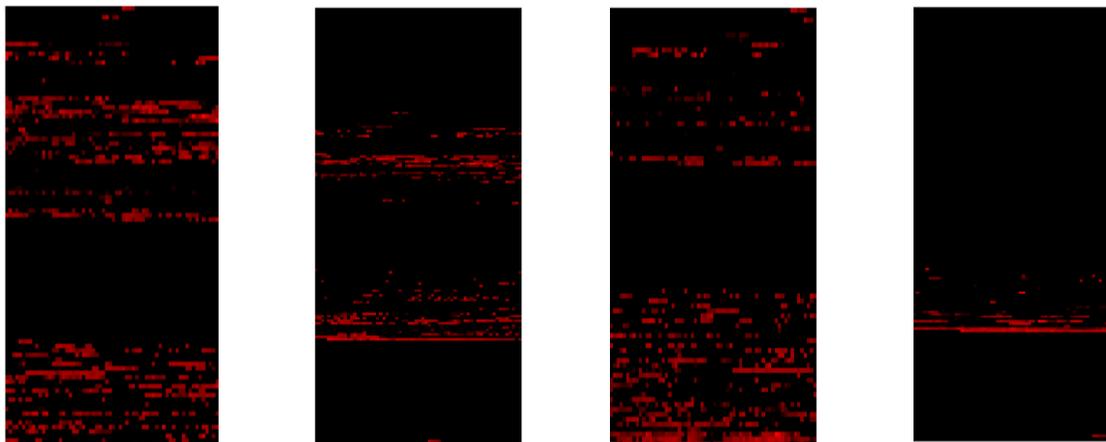
- ARM64: Kunpeng 920 (ARMv8.2) @2.6GHz
- x86: Xeon 6230N @ 2.3GHz
- RAM: 378GB
- Linux 4.18 (ARM64) / 4.19 (X86)
- LLVM 19.1.6 (Clang/LLD)
- LLVM-BOLT 15.0.4 custom
- MySQL 8.4.0
- Sysbench 1.0.20 (oltp_read_write)
- 4 потока на сервер / 4 потока на бенчмарк
- Метрика: Queries/sec
- CFLAGS/LDFLAGS: -O2 -flto -fprofile-use=X



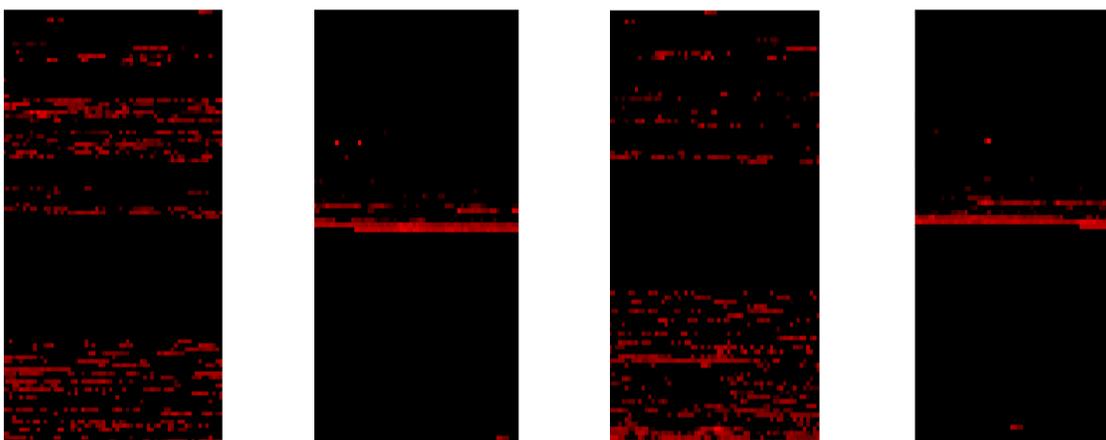
Эффект на реальном ПО

Оригинал BOLT LTO+PGO LTO+PGO+BOLT

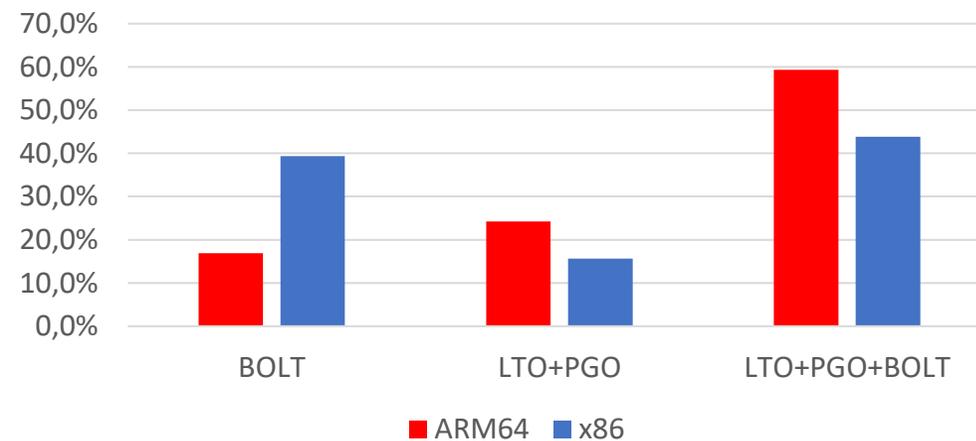
ARM64



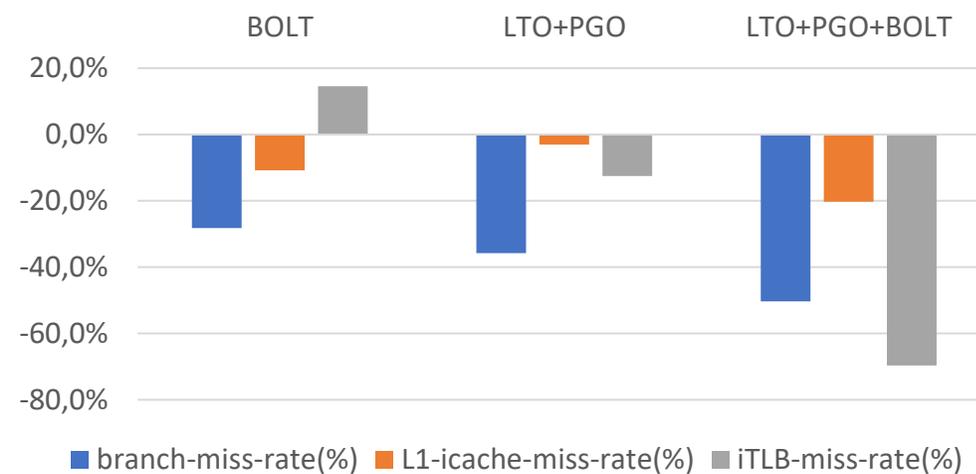
X86



Queries/Sec Улучшение (%)

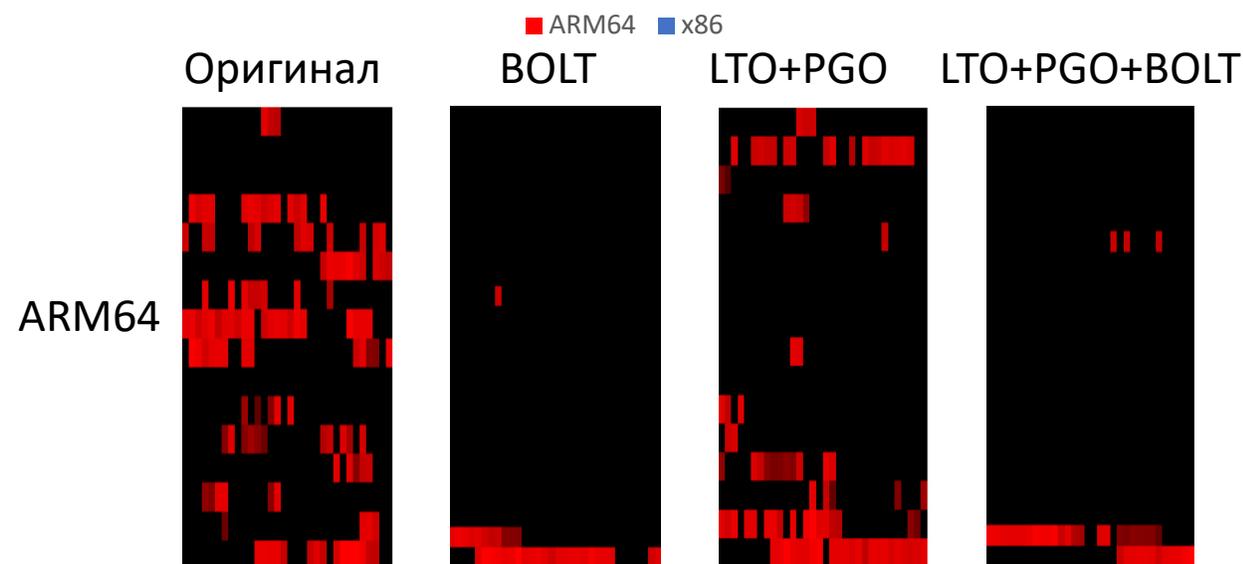
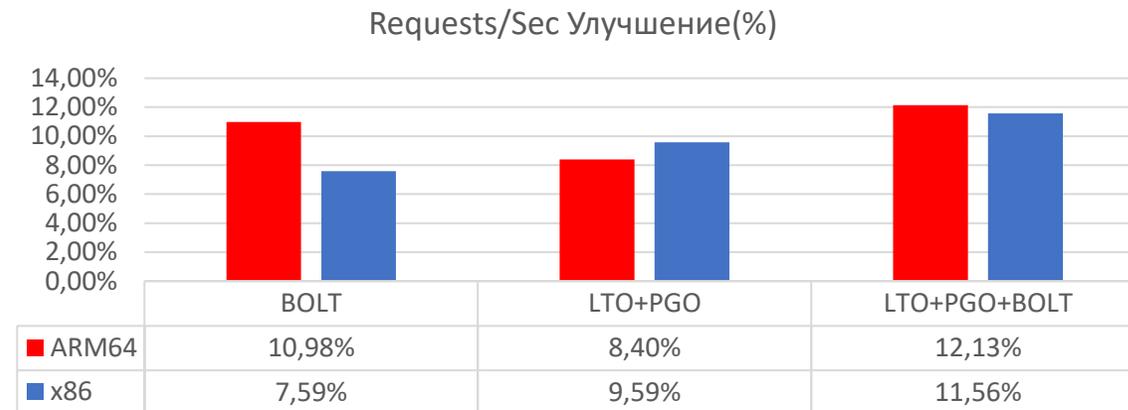


perf stat vs Original (ARM64)



Эффект на реальном ПО

- ARM64: Kunpeng 920 (ARMv8.2) @2.6GHz
- x86: Xeon 6230N @ 2.3GHz
- RAM: 378GB
- Linux 4.18 (ARM64) / 4.19 (X86)
- LLVM 19.1.6 (Clang/LLD)
- LLVM-BOLT 15.0.4 custom
- Nginx 1.23.3
- Wrk 4.2.0
- 1 поток на сервер / 4 потока на бенчмарк
- Metric: Requests/sec



Эффект на реальном ПО



Clang : [8-19%](#) поперх PGO+ThinLTO (LLVM Dev Meeting 2022)



Linux Kernel : [2.5%](#) rockdb (Linux Plumbers 2024)

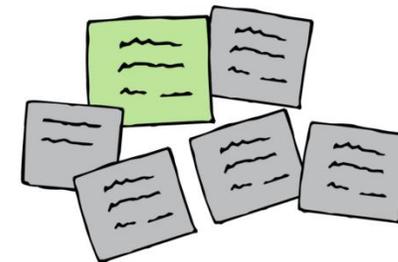


Chromium : [6%](#) ускорение поперх PGO



Python : [1-5%](#) in v3.12

Известные проблемы



- Входной файл:
 - Проблемы с дизассемблированием (ассемблерные вставки смешанные с данными, пример: OpenSSL)
 - Некорректные релокации (ошибки в линкерах)
- Выходной файл:
 - Существенное увеличение размера выходного файла (до ~x2)
 - Дополнительные сегменты в ELF файле
- Существенное потребление памяти BOLT-ом



Инструментация для ARM64

Цель: Добавить поддержку инструментации для ARM64

- perf профиль на ARM64 предоставляет менее точный профиль без BRBE
- Инструментация позволяет получить точный профиль

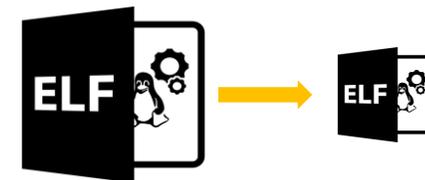
Расширен функционал:

- Платформенная часть:
 - AArch64 специфичные инструкции для создания сниппетов для инструментации прямых и косвенных вызовов/переходов
- Библиотека инструментации:
 - Системные вызовы для AArch64
 - Перехват инициализации/финализации ELF
 - Обработка косвенных вызовов

Статус:

- Изменения приняты в LLVM 18

Размера выходного файла



Экспериментальная опция “rewrite”

Проблема:

- VOLT не создаёт ELF файл заново, а модифицирует структуру входного файла
- VOLT добавляет новые секции и сегменты после исходных
- Совместимость с strip/objcopy

```
[ 8] .dynstr          STRTAB          00000000002aaa9c 2aaa9c 82de0d 00   A 0 0 1
[ 9] .rela.dyn        RELA             0000000000ad88b0 ad88b0 5f34d8 18   A 4 0 8
[10] .rela.plt        RELA             00000000010cbd88 10cbd88 005118 18  AI 4 26 8
[11] .rodata          PROGBITS         00000000010d0ea0 10d0ea0 cdec90 00  AMS 0 0 16
[12] .bolt.org.eh_frame PROGBITS         0000000001dafb30 1dafb30 359f40 00   A 0 0 8
[13] .bolt.org.gcc_except_table PROGBITS         0000000002109a70 2109a70 102bd4 00   A 0 0
[14] .bolt.org.eh_frame_hdr PROGBITS         000000000220c644 220c644 08fa04 00   A 0 0 4
[15] .bolt.org.text    PROGBITS         00000000022ac080 229c080 18633f0 00  AX 0 0 64
[16] .init            PROGBITS         0000000003b0f470 3aff470 000018 00  AX 0 0 4
[17] .fini            PROGBITS         0000000003b0f488 3aff488 000014 00  AX 0 0 4
[30] .bss             NOBITS           000000000407b7c0 404b7a0 554903 00  WA 0 0 64
[31] .text            PROGBITS         00000000045e0340 40502d8 0e1d80 00  AX 0 0 64
[32] .text.cold        PROGBITS         00000000046c20c0 4132058 18113ac 00  AX 0 0 64
[33] .eh_frame         PROGBITS         0000000005ed3470 5943404 73df58 00   A 0 0 8
[34] .eh_frame_hdr     PROGBITS         00000000066113c8 608135c 123334 00   A 0 0 1
[35] .gcc_except_table PROGBITS         00000000067346fc 61a4690 212c27 00   A 0 0 4
[36] .comment          PROGBITS         0000000000000000 63b72b7 00003b 01  MS 0 0 1
[37] .gnu.build.attributes NOTE              0000000000000000 63b72f4 000024 00   0 0 4
[38] .note.bolt_info   NOTE             0000000000000000 63b7318 000144 00   0 0 1
[39] .shstrtab         STRTAB           0000000000000000 63b745c 000193 00   0 0 1
```

Program Headers:

```
Type      Offset  VirtAddr  PhysAddr  FileSiz MemSiz  Flg Align
PHDR      0x404b7a0 0x00000000045db7a0 0x00000000045e0000 0x0002d8 0x0002d8 R 0x8
INTERP    0x0002e0 0x00000000000002e0 0x00000000000002e0 0x00001b 0x00001b R 0x1
[Requesting program interpreter: /lib/ld-linux-aarch64.so.1]
LOAD      0x000000 0x0000000000000000 0x0000000000000000 0x229c048 0x229c048 R 0x10000
LOAD      0x229c080 0x00000000022ac080 0x00000000022ac080 0x1866a50 0x1866a50 R E 0x10000
LOAD      0x3b02ad0 0x0000000003b22ad0 0x0000000003b22ad0 0x1b1e58 0x1b2530 RW 0x10000
LOAD      0x3cb4940 0x0000000003ce4940 0x0000000003ce4940 0x396e60 0x8eb783 RW 0x10000
LOAD      0x404b7a0 0x00000000045db7a0 0x00000000045e0000 0x236bb17 0x236bb17 R E 0x10000
TLS       0x3b02ad0 0x0000000003b22ad0 0x0000000003b22ad0 0x000008 0x0002b0 R 0x8
DYNAMIC   0x3ca1868 0x0000000003cc1868 0x0000000003cc1868 0x000640 0x000640 RW 0x8
```

Размера выходного файла

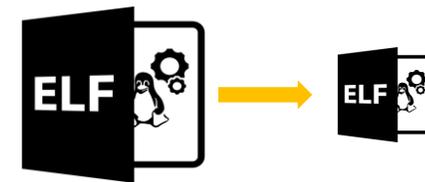
Экспериментальная опция “rewrite”

Возможности и ограничения:

- Уменьшает размер выходного файла
- Сохраняет порядок секций/сегментов как в входном файле
- Требуется полное дизассемблирование входного файла

Статус и эффект:

- Превышение размера выходных файлов снижается с 60-130% до 0-30%
- Опубликовано в LLVM [Discourse](#) и [Phabricator](#)
- Переносим на актуальную версию LLVM



Nginx	Размер (KB)	Overhead
Original	821	-
BOLT (без rewrite)	1546	88%
BOLT (с rewrite)	867	6%

MySQL	Размер (KB)	Overhead
Original	65840	-
BOLT (без rewrite)	102111	55%
BOLT (с rewrite)	68074	3%

Поддержка Golang



Проблема:

- PGO в Go с 1.20 (базовая)
- Go GC & Scheduler – используют внутренние структуры
- Внутренние структуры содержат позиции функций и базовых блоков внутри функций
- Внутренние структуры меняются от версии к версии

```
type moduledata struct {
    pcHeader *pcHeader
    funcnametab []byte
    cutab []uint32
    filetab []byte
    pctab []byte
    pclntable []byte
    ftab []functab
    findfunctab uintptr
    minpc, maxpc uintptr

    text, etext uintptr
    noptrdata, enoptrdata uintptr
    data, edata uintptr
    bss, ebss uintptr
    noptrbss, enoptrbss uintptr
    end, gcdata, gcbss uintptr
    types, etypes uintptr

    textsectmap []textsect
    typelinks []int32 // offsets
    itablinks []*itab

    ptab []ptabEntry

    pluginpath string
    pkghashes []modulehash

    modulename string
    modulehashes []modulehash

    hasmain uint8 // 1 if module c
    gcdatamask, gcbssmask bitvecto
    typemap map[typeOff]*_type //
    bad bool // module failed to l
    next *moduledata
}

type pcHeader struct {
    magic uint32
    pad1, pad2 uint8
    minLC uint8
    ptrSize uint8
    nfunc int
    nfiles uint
    funcnameOffset uintptr
    cuOffset uintptr
    filetabOffset uintptr
    pctabOffset uintptr
    pclnOffset uintptr
}

type functab struct {
    entry uintptr
    funcOff uintptr
}

type _func struct {
    entry uintptr //
    nameOff int32 //

    args int32
    deferreturn int32

    pcsp uint32
    pcfile uint32
    pcln uint32
    npcdata uint32
    cuOffset int32 //
    funcID funcID //
    flag funcFlag
    _ [1]byte /
    nfuncdata uint8 /
}

type _type struct {
    size uintptr
    ptrdata uintptr
    hash uint32
    tflag tflag
    align int
    fieldAlign uint8
    kind uint8
    // function for com
    // (ptr to object /
    equal func(unsafe.I
    // If the KindGCPro
    // Otherwise it is
    gcdata *byte
    str nameOff
    ptrToThis typeOff
}

type method struct {
    name nameOff
    mtyp typeOff
    ifn textOff
    tfn textOff
}

type uncommontype struct {
    pkgpath nameOff
    mcount uint16 // r
    xcount uint16 // r
    moff uint32 // c
    _ uint32 // t
}
```



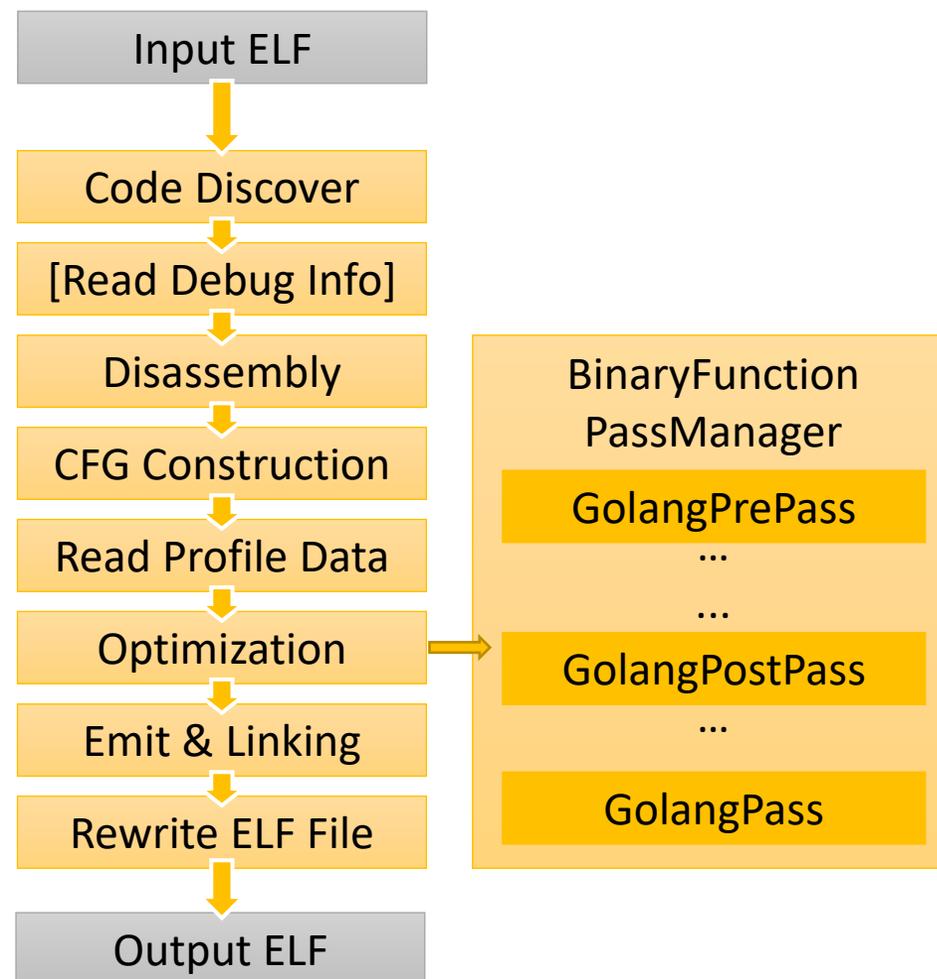
Поддержка Golang

Реализация:

- 3 дополнительных BOLT прохода:
 - GolangPrePass: Парсинг внутренних структур данных
 - GolangPostPass, GolangPass: Исправление внутренних структур + добавление инструкций в код
- Поддержка Go версий 1.14-1.22 для x86_64 и ARM64

Статус и эффект:

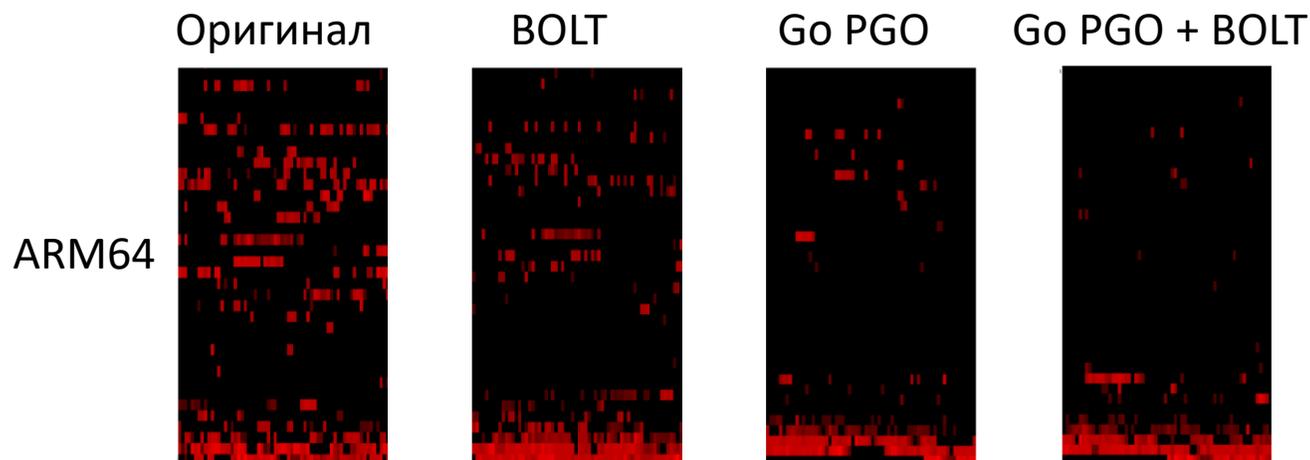
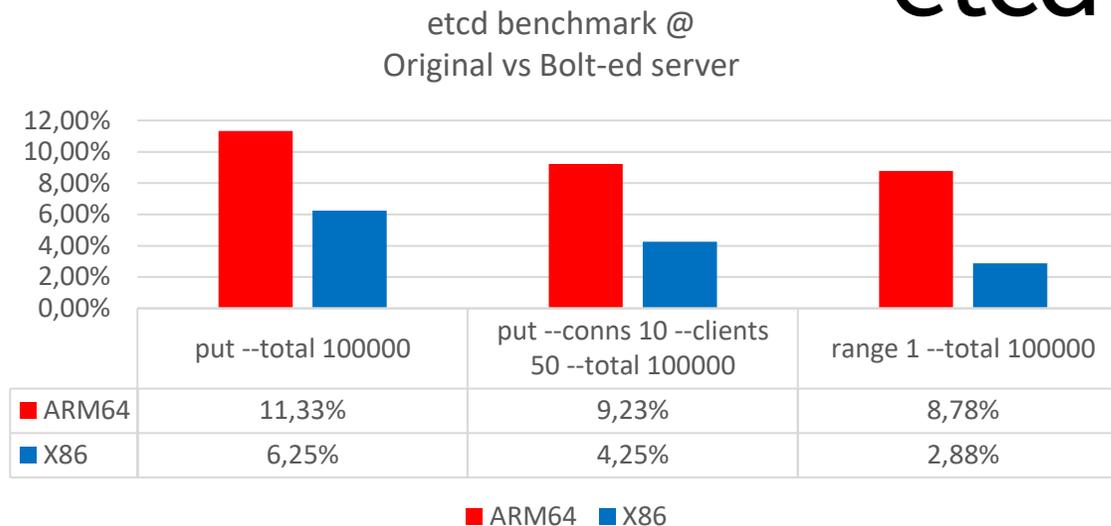
- Опубликован RFC на [LLVM Phabricator](#)
- Текущая реализация используется для оптимизации реальных Go приложений.
- Некоторые приложения ускорились до **13%**.



Golang: Эффект на реальном ПО



- ARM64: Kunpeng 920 (ARMv8.2) @2.6GHz
- x86: Xeon 6230N @ 2.3GHz
- RAM: 378GB
- Golang 1.17.8 (ARM64) / 1.18.7 (x86)
- Linux 4.18 (ARM64) / 4.19 (X86)
- LLVM-BOLT 15.0.4 custom
- Etcd 3.6.0-alpha.0
- 4 потока на сервер / 4 потока на бенчмарк
- Metric: Requests/sec



Вопросы

Спасибо!

Василий Леоненко

  @vleonen