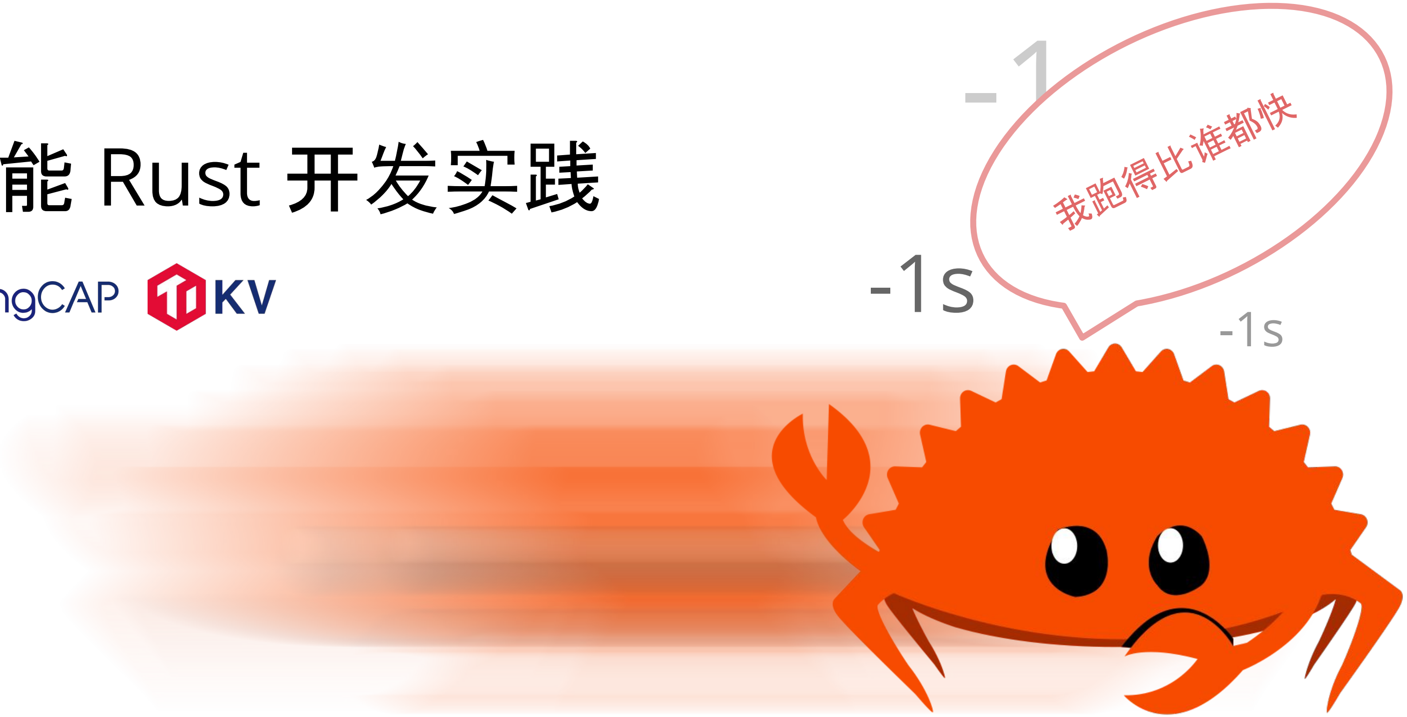
# TiKV 高性能 Rust 开发实践

Wenxuan @ PingCAP TiKV



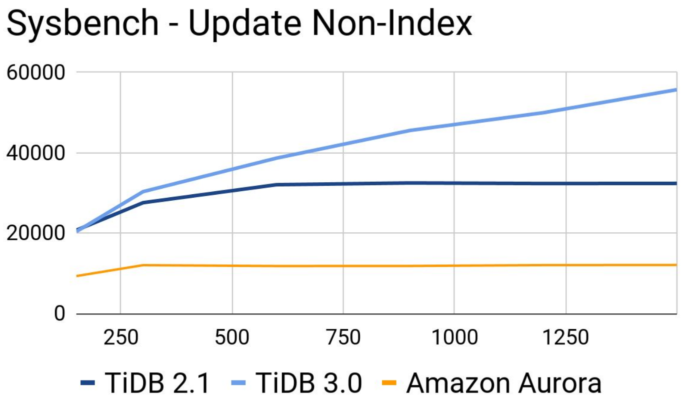
## 施闻轩 · TiKV Engineer

TiKV Optimization & Development

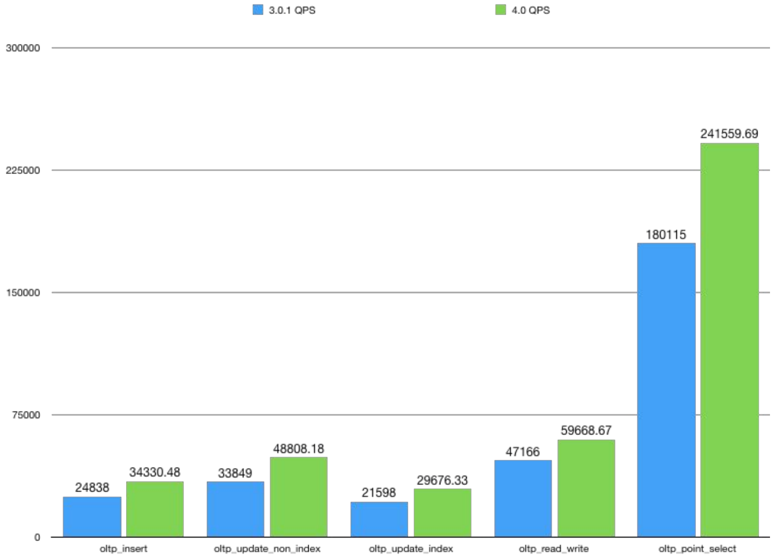
TiKV Coprocessor SIG Tech Lead

# Sysbench

2.1 → 3.0

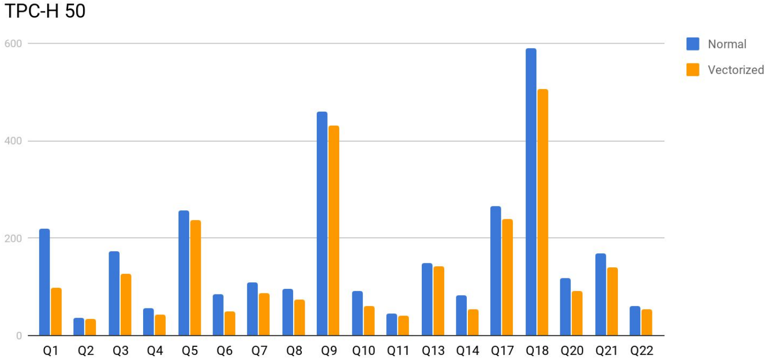


3.0 → 4.0 (not released)

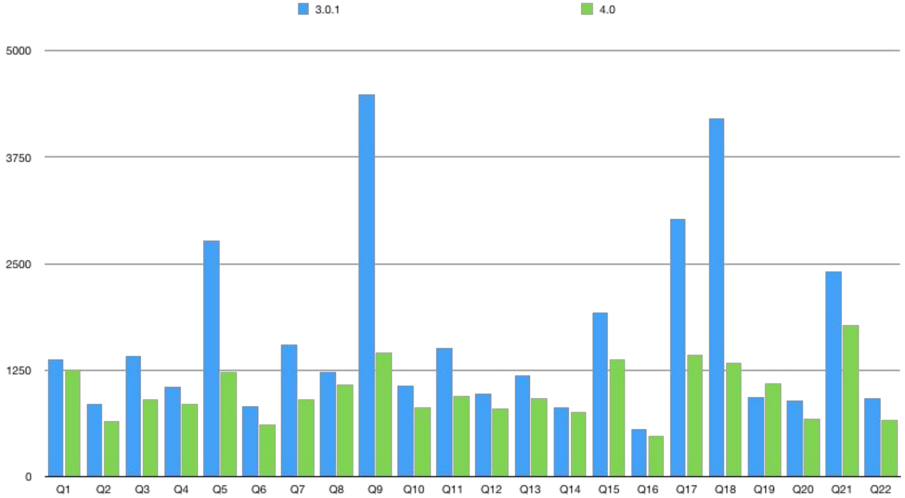


# TPC-H

2.1 → 3.0



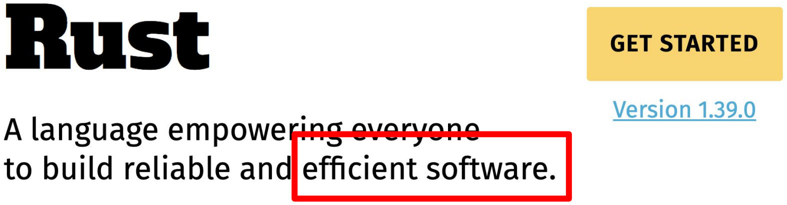
3.0 → 4.0 (not released)



# Performance boost comes from...

1. New Feature / Algorithm / Data Structure
   - v3.0: Multi-thread Raft
   - v3.0: Batch gRPC messages
   - v3.0: Titan Engine
   - v3.0: Vectorization
   - v4.0: Will be officially announced in future
2. **Engineering Improvement / Constant Optimization**

# Rust Claims…

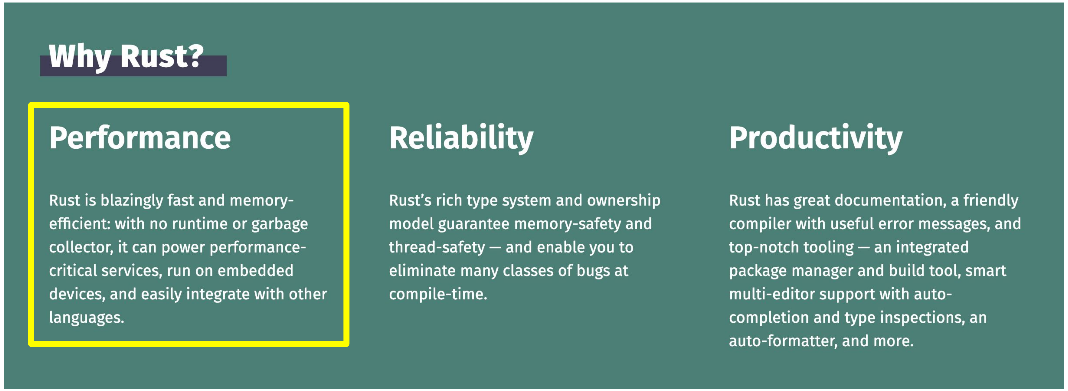


Rust

GET STARTED

Version 1.39.0

A language empowering everyone to build reliable and efficient software.



Why Rust?

**Performance**

Rust is blazingly fast and memory-efficient: with no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages.

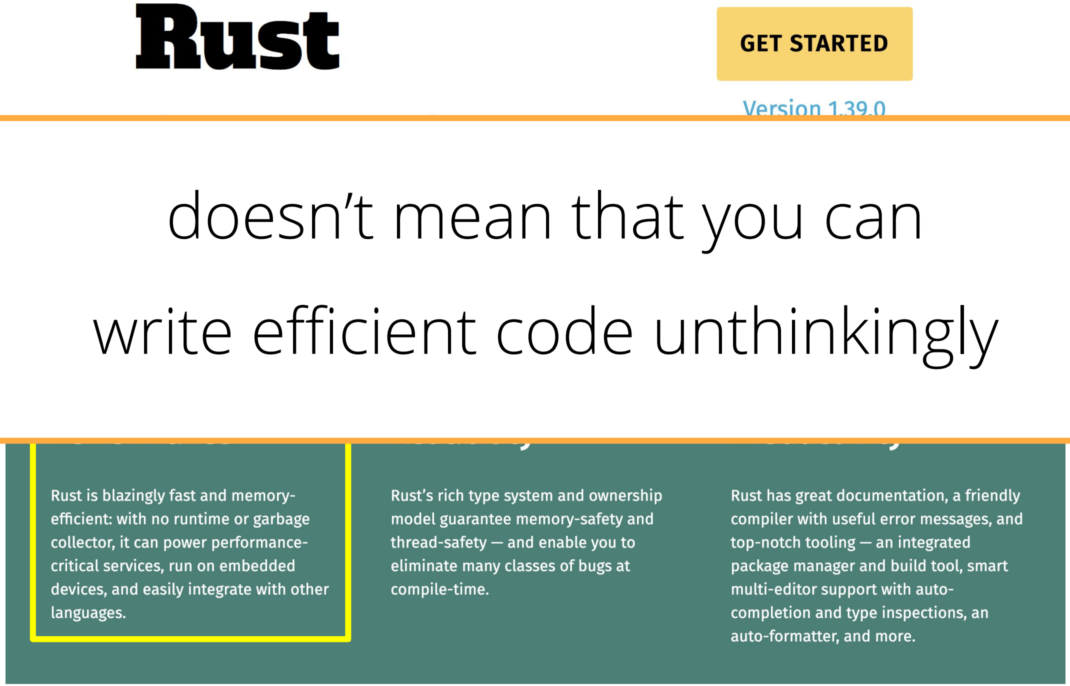
**Reliability**

Rust's rich type system and ownership model guarantee memory-safety and thread-safety — and enable you to eliminate many classes of bugs at compile-time.

**Productivity**

Rust has great documentation, a friendly compiler with useful error messages, and top-notch tooling — an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more.

# Rust Claims…



Rust is blazingly fast and memory-efficient: with no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages.

Rust’s rich type system and ownership model guarantee memory-safety and thread-safety — and enable you to eliminate many classes of bugs at compile-time.

Rust has great documentation, a friendly compiler with useful error messages, and top-notch tooling — an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more.

doesn’t mean that you can

write efficient code unthinkingly

# Find Hotspots

# Why?

Amdahl's law

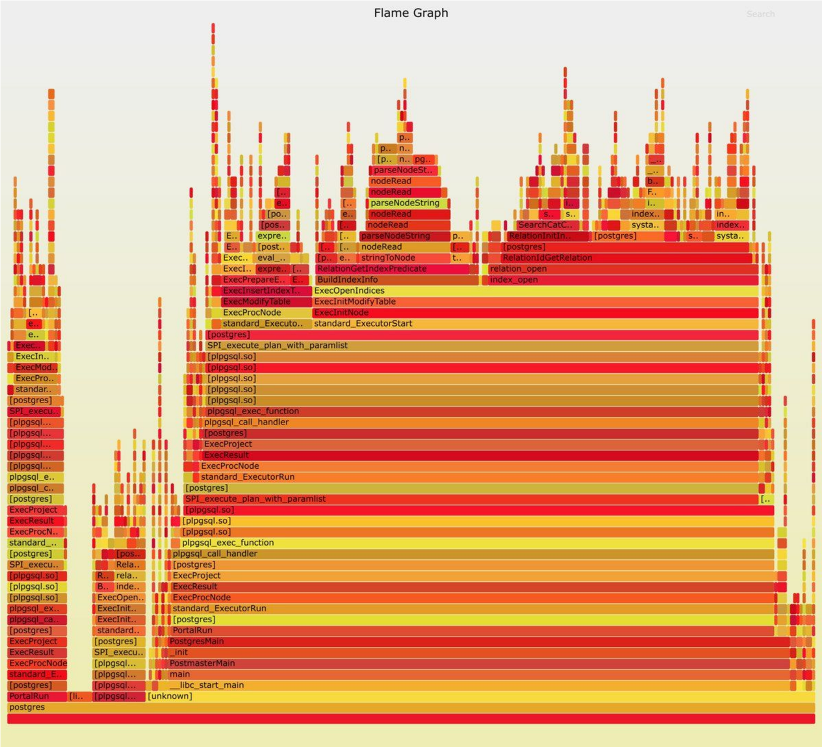
$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

# perf + FlameGraph

[https://github.com/brendangregg/FlameGraph](https://github.com/brendangregg/FlameGraph)

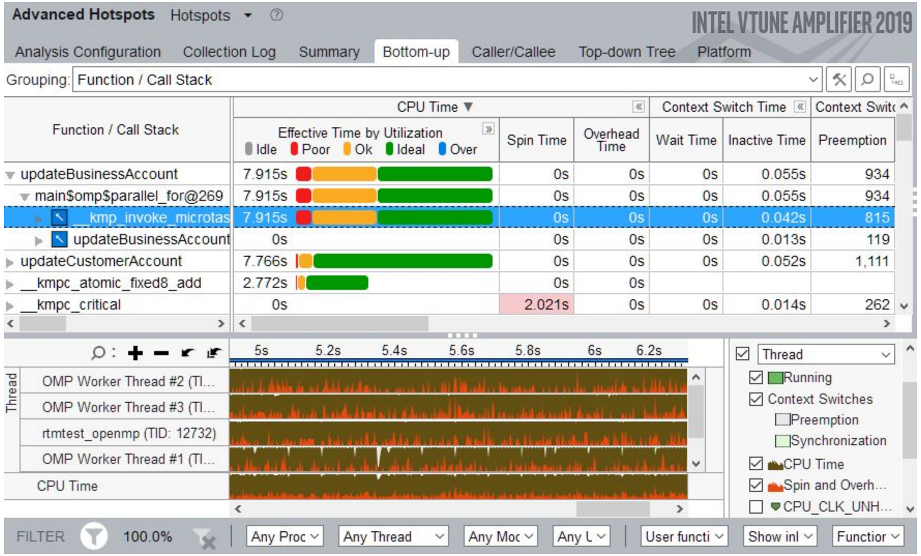
```
$ perf record …
$ perf script |
 ./stackcollapse-perf.pl |
 ./flamegraph.pl
```

Even very useful in production!



# Intel VTune

- It is free now
- Easy to use GUI
- Top-down Microarchitecture Analysis



Grouping: Function / Call Stack

| Function / Call Stack | CPU Time ▼ |  |  | Context Switch Time |  | Context Switc |
|---|---|---|---|---|---|---|
|  | Effective Time by Utilization<br>Idle Poor Ok Ideal Over | Spin Time | Overhead Time | Wait Time | Inactive Time | Preemption |
| updateBusinessAccount | 7.915s | 0s | 0s | 0s | 0.055s | 934 |
| main$omp$parallel_for@269 | 7.915s | 0s | 0s | 0s | 0.055s | 934 |
| __kmp_invoke_microtas | 7.915s | 0s | 0s | 0s | 0.042s | 815 |
| updateBusinessAccount | 0s | 0s | 0s | 0s | 0.013s | 119 |
| updateCustomerAccount | 7.766s | 0s | 0s | 0s | 0.052s | 1,111 |
| __kmpc_atomic_fixed8_add | 2.772s | 0s | 0s |  |  |  |
| __kmpc_critical | 0s | 2.021s | 0s | 0s | 0.014s | 262 |

PingCAP.com

# Low-hanging Fruits

# Replace Memory Allocators

Use efficient memory allocators like jemalloc:

```
#[global_allocator]
static GLOBAL: Jemalloc = Jemalloc;
```

Improve ~25% for TPC-H Q1.

# Link Time Optimization

LTO can apply cross-crate and even cross-language optimizations!

```toml
[profile.release]
lto = true
```

Improve ~15% for TPC-H Q1.

# Set Target CPU

If not set (default):

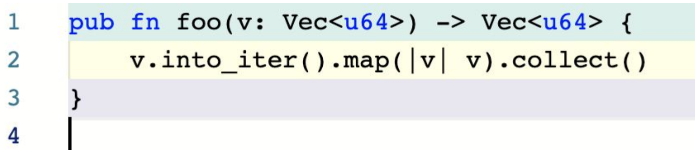
- Portable release (e.g. x86_64)
- Modern instructions like AVX is not used

# (Almost) Drop-in Replacement

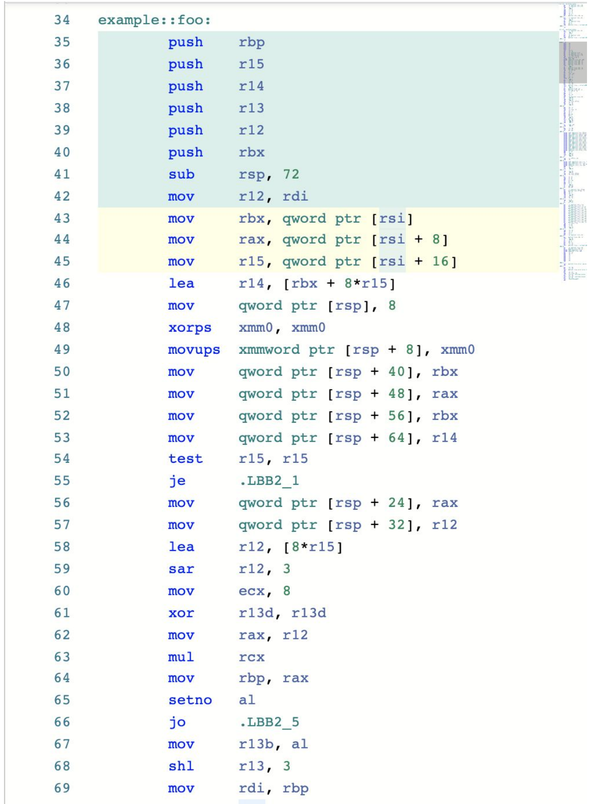
- Hash Collections like `std::collections::HashMap` → hashbrown *(already in std)*
  - 2.6x faster insert
- Concurrent utilities like `std::sync::mpsc` → crossbeam
  - It is even faster than Go channel
- Synchronization primitives like `std::sync::Mutex` → parking_lot
  - 5x faster Mutex

# Micro Optimizations

# Compiler Can Be Stupid



```rust
pub fn foo(v: Vec<u64>) -> Vec<u64> {
    v.into_iter().map(|v| v).collect()
}
```



```
example::foo:
        push    rbp
        push    r15
        push    r14
        push    r13
        push    r12
        push    rbx
        sub     rsp, 72
        mov     r12, rdi
        mov     rbx, qword ptr [rsi]
        mov     rax, qword ptr [rsi + 8]
        mov     r15, qword ptr [rsi + 16]
        lea     r14, [rbx + 8*r15]
        mov     qword ptr [rsp], 8
        xorps   xmm0, xmm0
        movups  xmmword ptr [rsp + 8], xmm0
        mov     qword ptr [rsp + 40], rbx
        mov     qword ptr [rsp + 48], rax
        mov     qword ptr [rsp + 56], rbx
        mov     qword ptr [rsp + 64], r14
        test    r15, r15
        je      .LBB2_1
        mov     qword ptr [rsp + 24], rax
        mov     qword ptr [rsp + 32], r12
        lea     r12, [8*r15]
        sar     r12, 3
        mov     ecx, 8
        xor     r13d, r13d
        mov     rax, r12
        mul     rcx
        mov     rbp, rax
        setno   al
        jo      .LBB2_5
        mov     r13b, al
        shl     r13, 3
        mov     rdi, rbp
```

# Less Allocation, Less Copy

Pre-allocate collections when possible:

- `Vec::with_capacity()`

# Less Allocation, Less Copy

Use memory pool techniques to reduce allocation:

- Dynamic collections over pre-allocated memory: bumpalo
- Dynamically allocate same object: TypedArena
- 1d `Vec<Vec<T>>`: nested

# Less Allocation, Less Copy

Prefer stack allocation instead of heap allocation:

- Static array allocates on stack: `[T; N]`
- `Box<T>` allocates on heap
- `Vec<T>` allocates data on heap (obviously)
- SmallVec can be very useful
    - allocates N bytes on stack but is also growable (on heap for len > N).

# Less Allocation, Less Copy

Move less data

- `Box<[T]>` is smaller than `Vec<T>`
- Wrap *big* and *frequently moved* structures with `Box<T>` to keep it small

# Less Allocation, Less Copy

Error size matters:

- `Result<T, E>` is an enum: `sizeof(Result<T, E>) == max(sizeof(T), sizeof(E))`
- The size of `E` is the minimum size of your `Result`.
- Try `Result` is not zero cost sometimes!

TiKV TPC-H improved by 14% by reducing errors from 200 bytes to 8 bytes:

- Reduced 3×200 bytes memory copy each KV iterate
- `Result<T, Error>` → `Result<T, Box<Error>>`

# Less Allocation, Less Copy

Use reference where possible:

- Annotate the structure with a lifetime parameter to contain references.

```
pub struct WriteRef<'a> {
    pub write_type: WriteType,
    pub start_ts: TimeStamp,
    pub short_value: Option<&'a [u8]>,
}
```

# Less Allocation, Less Copy

Use reference where possible:

- Use `Arc<T>` instead of a raw reference when lifetime is hard to write.

# Less Allocation, Less Copy

Use reference where possible:

- Use DSTs when you only want to wrap a DST reference (like `&[u8]`).

Example: Need `&[u8]` internally, but encapsulate with a type to provide extra feature.

```
struct Key(Vec<u8>);
fn get(k: &Key);
```

✕ Not zero cost

```
struct KeyRef<'a>(&'a [u8]);
fn get(k: KeyRef<'_>);
```

✓ Zero cost, but interface doesn't accept reference any more

```
struct KeyRef([u8]);
fn get(k: &KeyRef);
```

✓ Best: DST wrapper

# Less Allocation, Less Copy

Use reference where possible:

- Unbounded lifetime / raw pointer:

```
unsafe fn erase_lifetime<'a, T: ?Sized>(v: &T) -> &'a T {
    &*(v as *const T)
}
```

- Useful to simulate a `'Self` lifetime.
- Hate `unsafe{}` ? owning_ref or rental can help.
  - *However sometimes "unsafe" but **clear & simple** code is better.*

# Prefer Static Dispatch

```
fn foo(callback:
    Box<dyn FnOnce()>
);
```

✕ Box & Dynamic Dispatch

```
fn foo<F: FnOnce()>
    (callback: F);
```

✓ Static Dispatch

```
fn foo(callback:
    impl FnOnce()
);
```

✓ Syntax Sugar

Use `Box<dyn T>` only when you want different base types in one type:

- Return different base types in one function: `fn foo() -> Box<dyn T>`
- Store different base types in one container: `Vec<Box<dyn T>>`

# Reduce Branches

- To reduce instructions
- To reduce misprediction

Example: Memory-comparable encoding bytes (padding = 4).

| | | | | | |
|---|---|---|---|---|---|
| F | o | o | | | Source |
| F | o | o | \1 | \FE | Asecnding |
| F | o | o | \FE | \1 | Descending |

```
fn memcmp_encode(bytes: &[u8], is_desc: bool);
```

✕ Branch

# Reduce Branches

- To reduce instructions
- To reduce misprediction

Example: Memory-comparable encoding bytes (padding = 4).

| | | | | | |
|---|---|---|---|---|---|
| F | o | o | | | Source |
| F | o | o | \1 | \FE | Asecnding |
| F | o | o | \FE | \1 | Descending |

```
fn memcmp_encode_asc(bytes: &[u8]);

fn memcmp_encode_desc(bytes: &[u8]);
```

✓ No branch, some duplicate code

# Reduce Branches

- To reduce instructions
- To reduce misprediction

Example: Memory-comparable encoding bytes (padding = 4).

| | | | | | |
|---|---|---|---|---|---|
| F | o | o | | | Source |
| F | o | o | \1 | \FE | Asecnding |
| F | o | o | \FE | \1 | Descending |

```
trait Mode { const FLIP_BITS: bool; }

struct ModeAsc;
impl Mode for ModeAsc {
    const FLIP_BITS: bool = false;
}
struct ModeDesc;
impl Mode for ModeDesc {
    const FLIP_BITS: bool = true;
}

fn memcmp_encode(bytes: &[u8], mode: impl Mode);
```

✓ No branch (generics)

# Reduce Branches

- To reduce instructions
- To reduce misprediction

Example: Memory-comparable encoding bytes (padding = 4).

| | | | | | |
|---|---|---|---|---|---|
| F | o | o | | | Source |
| F | o | o | \1 | \FE | Asecnding |
| F | o | o | \FE | \1 | Descending |

```
fn memcmp_encode<const T: bool>
    (bytes: &[u8], is_desc: T);
```

✓ No branch (const generics) NIGHTLY

# Useful Friends

# Micro Benchmark: cargo bench

```
#[bench]
fn bench_foo(b: &mut Bencher) {
   b.iter(|| foo());
}
```

Run benchmark:

```
$ cargo bench
```

# Micro Benchmark: cargo bench

Remember to add `black_box`:

```rust
#[bench]
fn bench_add(b: &mut Bencher) {
    b.iter(|| add(1, 2));
}
```

× Constants are likely to be optimized away

```rust
#[bench]
fn bench_add(b: &mut Bencher) {
    b.iter(|| add(
        black_box(1),
        black_box(2),
    ));
}
```

✓ Wrap constants with `black_box`

# Micro Benchmark: criterion

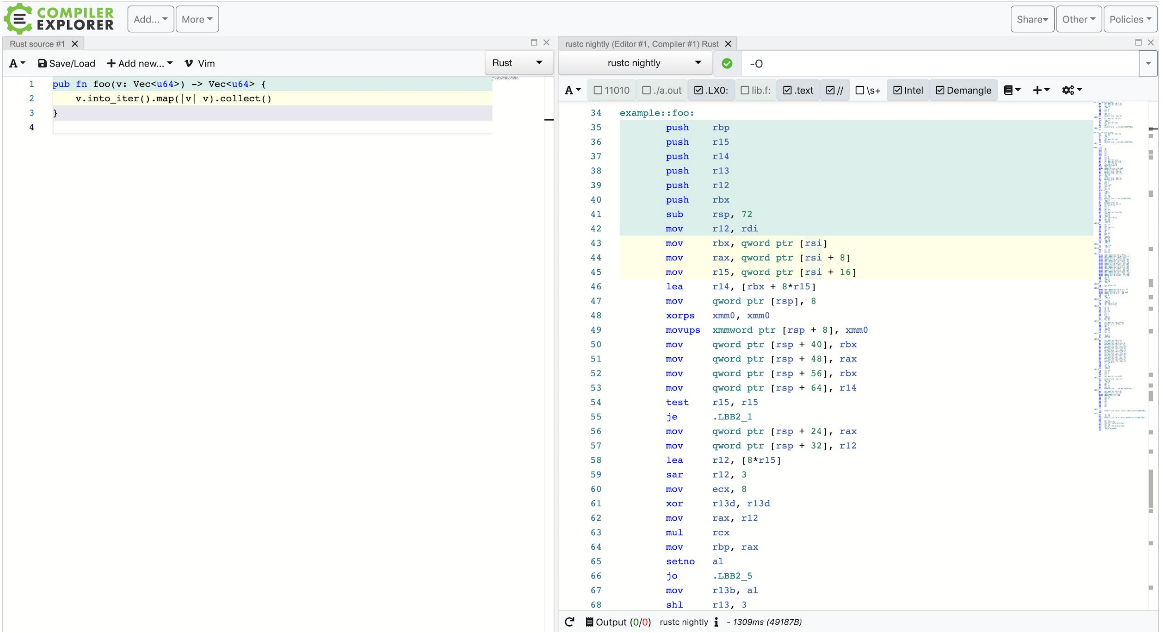
criterion is a replacement for cargo bench.

Features:

- Statistics-driven, avoid measurement noise
- Support different measurements (e.g. number of instructions instead of wall time)
- Generate charts
- Support parameterized benchmark

# Compiler Explorer

https://godbolt.org/

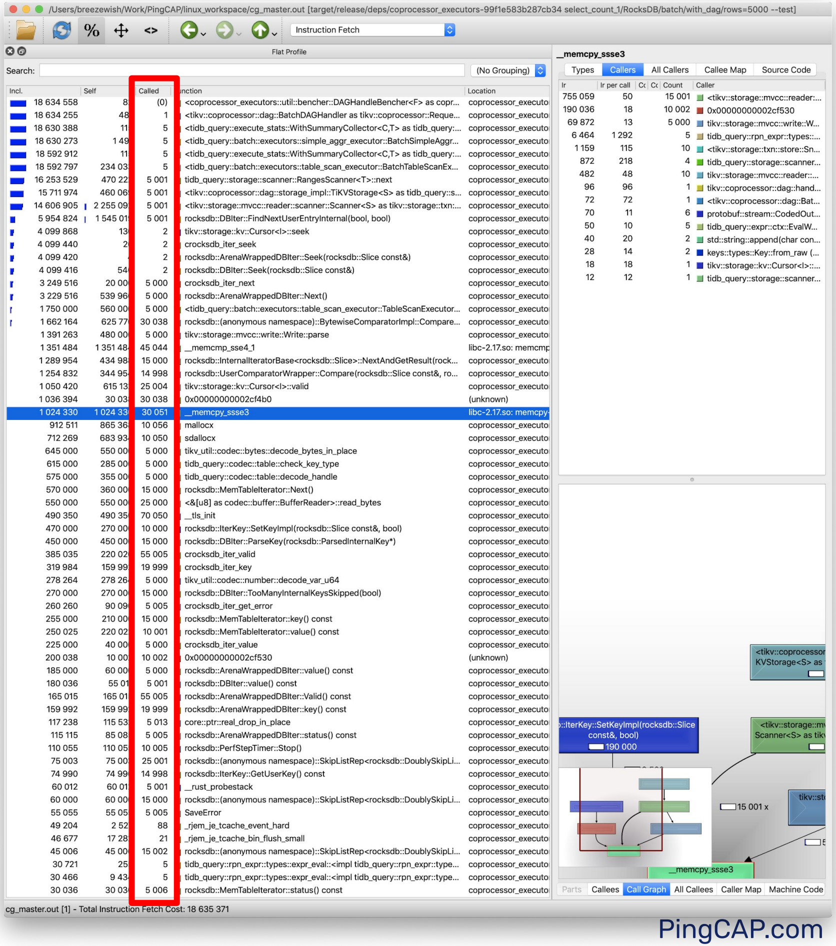


```rust
pub fn foo(v: Vec<u64>) -> Vec<u64> {
    v.into_iter().map(|v| v).collect()
}
```

```
example::foo:
        push    rbp
        push    r15
        push    r14
        push    r13
        push    r12
        push    rbx
        sub     rsp, 72
        mov     r12, rdi
        mov     rbx, qword ptr [rsi]
        mov     rax, qword ptr [rsi + 8]
        mov     r15, qword ptr [rsi + 16]
        lea     r14, [rbx + 8*r15]
        mov     qword ptr [rsp], 8
        xorps   xmm0, xmm0
        movups  xmmword ptr [rsp + 8], xmm0
        mov     qword ptr [rsp + 40], rbx
        mov     qword ptr [rsp + 48], rax
        mov     qword ptr [rsp + 56], rbx
        mov     qword ptr [rsp + 64], r14
        test    r15, r15
        je      .LBB2_1
        mov     qword ptr [rsp + 24], rax
        mov     qword ptr [rsp + 32], r12
        lea     r12, [8*r15]
        sar     r12, 3
        mov     ecx, 8
        xor     r13d, r13d
        mov     rax, r12
        mul     rcx
        mov     rbp, rax
        setno   al
        jo      .LBB2_5
        mov     r13b, al
        shl     r13, 3
```

# Callgrind + KCacheGrind

- Heavyweight
- Not a sample based profiler
- Run in Valgrind
- **Precise** function calls
- You can use the callgrind crate to control start / stop (to skip counting unwanted part)



/Users/breezewish/Work/PingCAP/linux_workspace/cg_master.out [target/release/deps/coprocessor_executors-99f1e583b287cb34 select_count_1/RocksDB/batch/with_dag/rows=5000 --test]

Instruction Fetch

Flat Profile

Search: (No Grouping)

| Incl. | Self | Called | unction | Location |
|---|---|---|---|---|
| 18 634 558 | 8 | (0) | <coprocessor_executors::util::bencher::DAGHandleBencher<F> as copr... | coprocessor_executo |
| 18 634 255 | 48 | 1 | <tikv::coprocessor::dag::BatchDAGHandler as tikv::coprocessor::Reque... | coprocessor_executo |
| 18 630 388 | 11 | 5 | <tidb_query::execute_stats::WithSummaryCollector<C,T> as tidb_query:... | coprocessor_executo |
| 18 630 273 | 1 49 | 5 | <tidb_query::batch::executors::simple_aggr_executor::BatchSimpleAggr... | coprocessor_executo |
| 18 592 912 | 11 | 5 | <tidb_query::execute_stats::WithSummaryCollector<C,T> as tidb_query:... | coprocessor_executo |
| 18 592 797 | 234 03 | 5 | <tidb_query::batch::executors::table_scan_executor::BatchTableScanEx... | coprocessor_executo |
| 16 253 529 | 470 22 | 5 001 | tidb_query::storage::scanner::RangesScanner<T>::next | coprocessor_executo |
| 15 711 974 | 460 06 | 5 001 | <tikv::coprocessor::dag::storage_impl::TiKVStorage<S> as tidb_query::s... | coprocessor_executo |
| 14 606 905 | 2 255 09 | 5 001 | <tikv::storage::mvcc::reader::scanner::Scanner<S> as tikv::storage::txn:... | coprocessor_executo |
| 5 954 824 | 1 545 01 | 5 001 | rocksdb::DBIter::FindNextUserEntryInternal(bool, bool) | coprocessor_executo |
| 4 099 868 | 13 | 2 | tikv::storage::kv::Cursor<I>::seek | coprocessor_executo |
| 4 099 440 | 2 | 2 | crocksdb_iter_seek | coprocessor_executo |
| 4 099 420 |  | 2 | rocksdb::ArenaWrappedDBIter::Seek(rocksdb::Slice const&) | coprocessor_executo |
| 4 099 416 | 54 | 2 | rocksdb::DBIter::Seek(rocksdb::Slice const&) | coprocessor_executo |
| 3 249 516 | 20 00 | 5 000 | crocksdb_iter_next | coprocessor_executo |
| 3 229 516 | 539 96 | 5 000 | rocksdb::ArenaWrappedDBIter::Next() | coprocessor_executo |
| 1 750 000 | 560 00 | 5 000 | <tidb_query::batch::executors::table_scan_executor::TableScanExecutor... | coprocessor_executo |
| 1 662 164 | 625 77 | 30 038 | rocksdb::(anonymous namespace)::BytewiseComparatorImpl::Compare... | coprocessor_executo |
| 1 391 263 | 480 00 | 5 000 | tikv::storage::mvcc::write::Write::parse | coprocessor_executo |
| 1 351 484 | 1 351 48 | 45 044 | __memcmp_sse4_1 | libc-2.17.so: memcmp |
| 1 289 954 | 434 98 | 15 000 | rocksdb::InternalIteratorBase<rocksdb::Slice>::NextAndGetResult(rock... | coprocessor_executo |
| 1 254 832 | 344 95 | 14 998 | rocksdb::UserComparatorWrapper::Compare(rocksdb::Slice const&, ro... | coprocessor_executo |
| 1 050 420 | 615 13 | 25 004 | tikv::storage::kv::Cursor<I>::valid | coprocessor_executo |
| 1 036 394 | 30 03 | 30 038 | 0x00000000002cf4b0 | (unknown) |
| 1 024 330 | 1 024 33 | 30 051 | __memcpy_ssse3 | libc-2.17.so: memcpy- |
| 912 511 | 865 36 | 10 056 | mallocx | coprocessor_executo |
| 712 269 | 683 93 | 10 050 | sdallocx | coprocessor_executo |
| 645 000 | 550 00 | 5 000 | tikv_util::codec::bytes::decode_bytes_in_place | coprocessor_executo |
| 615 000 | 285 00 | 5 000 | tidb_query::codec::table::check_key_type | coprocessor_executo |
| 575 000 | 355 00 | 5 000 | tidb_query::codec::table::decode_handle | coprocessor_executo |
| 570 000 | 360 00 | 15 000 | rocksdb::MemTableIterator::Next() | coprocessor_executo |
| 550 000 | 550 00 | 25 000 | <&[u8] as codec::buffer::BufferReader>::read_bytes | coprocessor_executo |
| 490 350 | 490 35 | 70 050 | __tls_init | coprocessor_executo |
| 470 000 | 270 00 | 10 000 | rocksdb::IterKey::SetKeyImpl(rocksdb::Slice const&, bool) | coprocessor_executo |
| 450 000 | 450 00 | 15 000 | rocksdb::DBIter::ParseKey(rocksdb::ParsedInternalKey*) | coprocessor_executo |
| 385 035 | 220 02 | 55 005 | crocksdb_iter_valid | coprocessor_executo |
| 319 984 | 159 99 | 19 999 | crocksdb_iter_key | coprocessor_executo |
| 278 264 | 278 26 | 5 000 | tikv_util::codec::number::decode_var_u64 | coprocessor_executo |
| 270 000 | 270 00 | 15 000 | rocksdb::DBIter::TooManyInternalKeysSkipped(bool) | coprocessor_executo |
| 260 260 | 90 09 | 5 005 | crocksdb_iter_get_error | coprocessor_executo |
| 255 000 | 210 00 | 15 000 | rocksdb::MemTableIterator::key() const | coprocessor_executo |
| 250 025 | 220 02 | 10 001 | rocksdb::MemTableIterator::value() const | coprocessor_executo |
| 225 000 | 40 00 | 5 000 | crocksdb_iter_value | coprocessor_executo |
| 200 038 | 10 00 | 10 002 | 0x00000000002cf530 | (unknown) |
| 185 000 | 60 00 | 5 000 | rocksdb::ArenaWrappedDBIter::value() const | coprocessor_executo |
| 180 036 | 55 01 | 5 001 | rocksdb::DBIter::value() const | coprocessor_executo |
| 165 015 | 165 01 | 55 005 | rocksdb::ArenaWrappedDBIter::Valid() const | coprocessor_executo |
| 159 992 | 159 99 | 19 999 | rocksdb::ArenaWrappedDBIter::key() const | coprocessor_executo |
| 117 238 | 115 53 | 5 013 | core::ptr::real_drop_in_place | coprocessor_executo |
| 115 115 | 85 08 | 5 005 | rocksdb::ArenaWrappedDBIter::status() const | coprocessor_executo |
| 110 055 | 110 05 | 10 005 | rocksdb::PerfStepTimer::Stop() | coprocessor_executo |
| 75 003 | 75 00 | 25 001 | rocksdb::(anonymous namespace)::SkipListRep<rocksdb::DoublySkipLi... | coprocessor_executo |
| 74 990 | 74 99 | 14 998 | rocksdb::IterKey::GetUserKey() const | coprocessor_executo |
| 60 012 | 60 01 | 5 001 | __rust_probestack | coprocessor_executo |
| 60 000 | 60 00 | 15 000 | rocksdb::(anonymous namespace)::SkipListRep<rocksdb::DoublySkipLi... | coprocessor_executo |
| 55 055 | 55 05 | 5 005 | SaveError | coprocessor_executo |
| 49 204 | 2 52 | 88 | _rjem_je_tcache_event_hard | coprocessor_executo |
| 46 677 | 17 28 | 21 | _rjem_je_tcache_bin_flush_small | coprocessor_executo |
| 45 006 | 45 00 | 15 002 | rocksdb::(anonymous namespace)::SkipListRep<rocksdb::DoublySkipLi... | coprocessor_executo |
| 30 721 | 25 | 5 | tidb_query::rpn_expr::types::expr_eval::<impl tidb_query::rpn_expr::type... | coprocessor_executo |
| 30 466 | 9 43 | 5 | tidb_query::rpn_expr::types::expr_eval::<impl tidb_query::rpn_expr::type... | coprocessor_executo |
| 30 036 | 30 03 | 5 006 | rocksdb::MemTableIterator::status() const | coprocessor_executo |

__memcpy_ssse3

Types | Callers | All Callers | Callee Map | Source Code

| Ir | Ir per call | Cc | Cc | Count | Caller |
|---|---|---|---|---|---|
| 755 059 | 50 |  |  | 15 001 | <tikv::storage::mvcc::reader:... |
| 190 036 | 18 |  |  | 10 002 | 0x00000000002cf530 |
| 69 872 | 13 |  |  | 5 000 | tikv::storage::mvcc::write::W... |
| 6 464 | 1 292 |  |  | 5 | tidb_query::rpn_expr::types::... |
| 1 159 | 115 |  |  | 10 | <tikv::storage::txn::store::Sn... |
| 872 | 218 |  |  | 4 | tidb_query::storage::scanner... |
| 482 | 48 |  |  | 10 | tikv::storage::mvcc::reader::... |
| 96 | 96 |  |  | 1 | tikv::coprocessor::dag::hand... |
| 72 | 72 |  |  | 1 | <tikv::coprocessor::dag::Bat... |
| 70 | 11 |  |  | 6 | protobuf::stream::CodedOut... |
| 50 | 10 |  |  | 5 | tidb_query::expr::ctx::EvalW... |
| 40 | 20 |  |  | 2 | std::string::append(char con... |
| 28 | 14 |  |  | 2 | keys::types::Key::from_raw (... |
| 18 | 18 |  |  | 1 | tikv::storage::kv::Cursor<I>::... |
| 12 | 12 |  |  | 1 | tidb_query::storage::scanner... |

cg_master.out [1] - Total Instruction Fetch Cost: 18 635 371

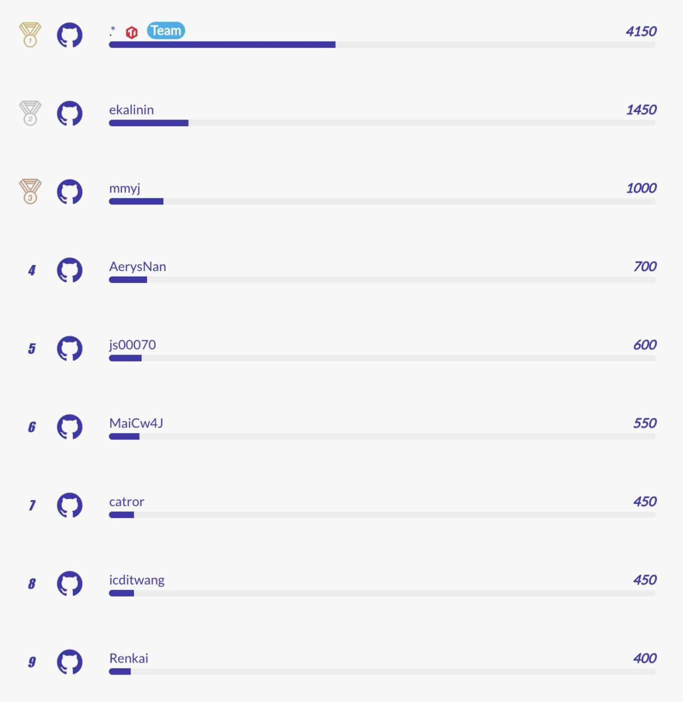
# Want to Contribute?

# TiKV Community

- TiKV Help Wanted Issues
- TiKV PCP Issues
- TiDB Hackathon

# PCP Season 1

- TiDB Performance Challenge Program
- Focus on *performance improvement*
- Season 1: 2019.11.04 - 2020.02.04
- You will be mentored
- You can be rewarded

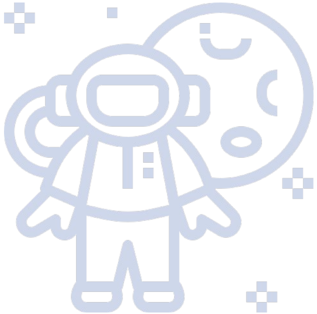


| | | |
|---|---|---|
| 1 | .* Team | 4150 |
| 2 | ekalinin | 1450 |
| 3 | mmyj | 1000 |
| 4 | AerysNan | 700 |
| 5 | js00070 | 600 |
| 6 | MaiCw4J | 550 |
| 7 | catror | 450 |
| 8 | icditwang | 450 |
| 9 | Renkai | 400 |

# Special Interest Group

- Focus on specific module.
- Promote to SIG Reviewer / SIG Committer.
- You can receive SIG credit and PCP credit at the same time!
- TiKV Coprocessor SIG
- TiKV Engine SIG
- TiDB Expression SIG

# Thank You !



Infra Meetup No. 120 活动交流群

该二维码7天内(12月13日前)有效，重新进入将更新