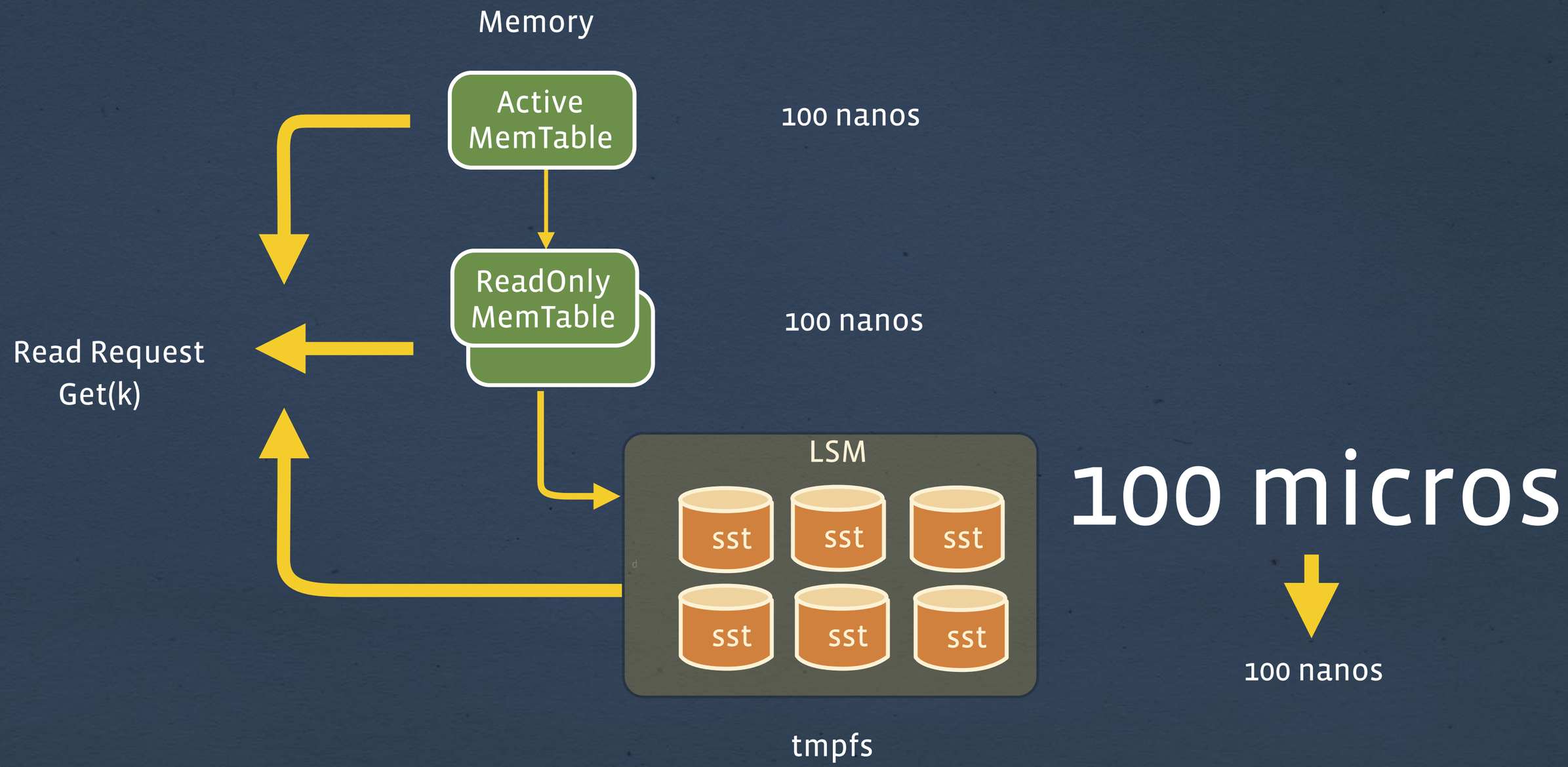
“Lockless” Get() in RocksDB?

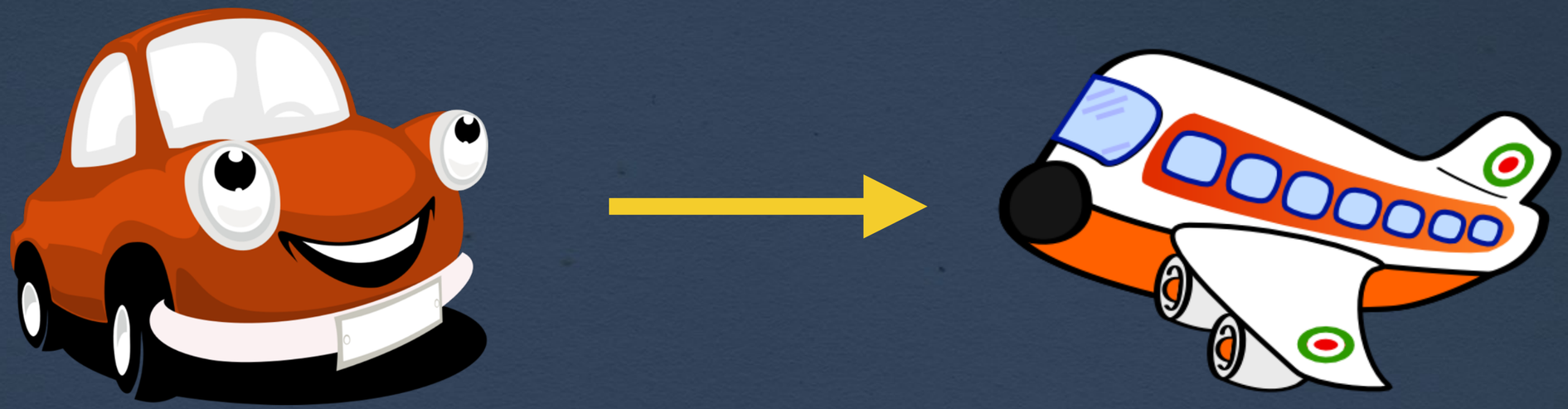
Scale with CPU Cores

Lei Jin
Database Engineering@Facebook

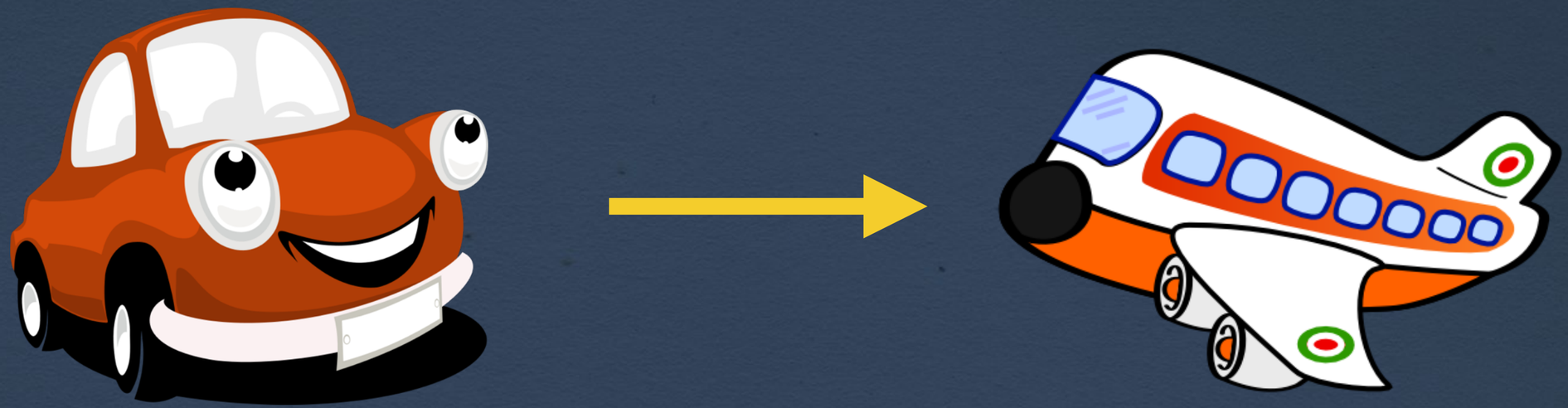
What's Changed in the Picture?



What's Changed in the Picture?

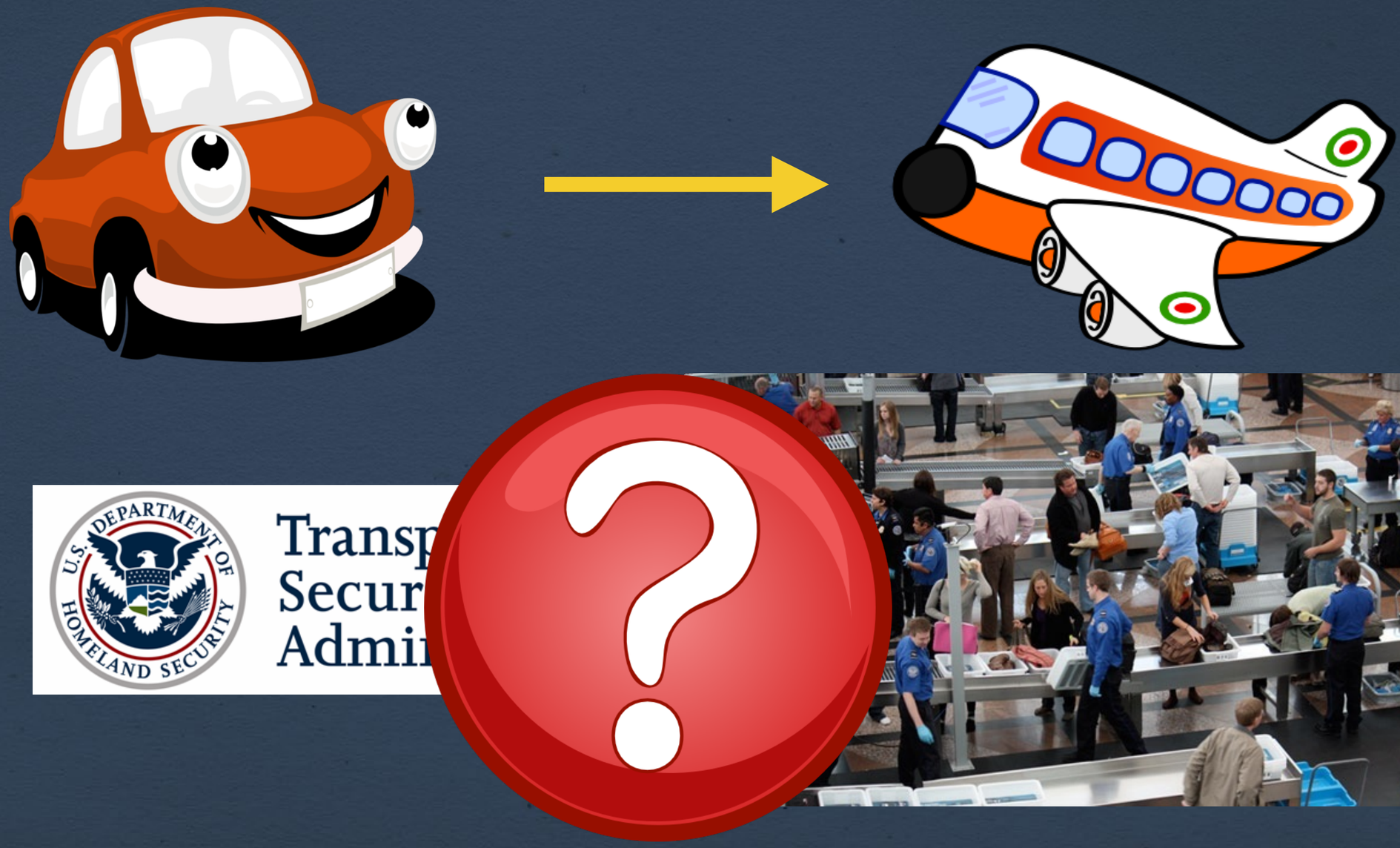


What's Changed in the Picture?



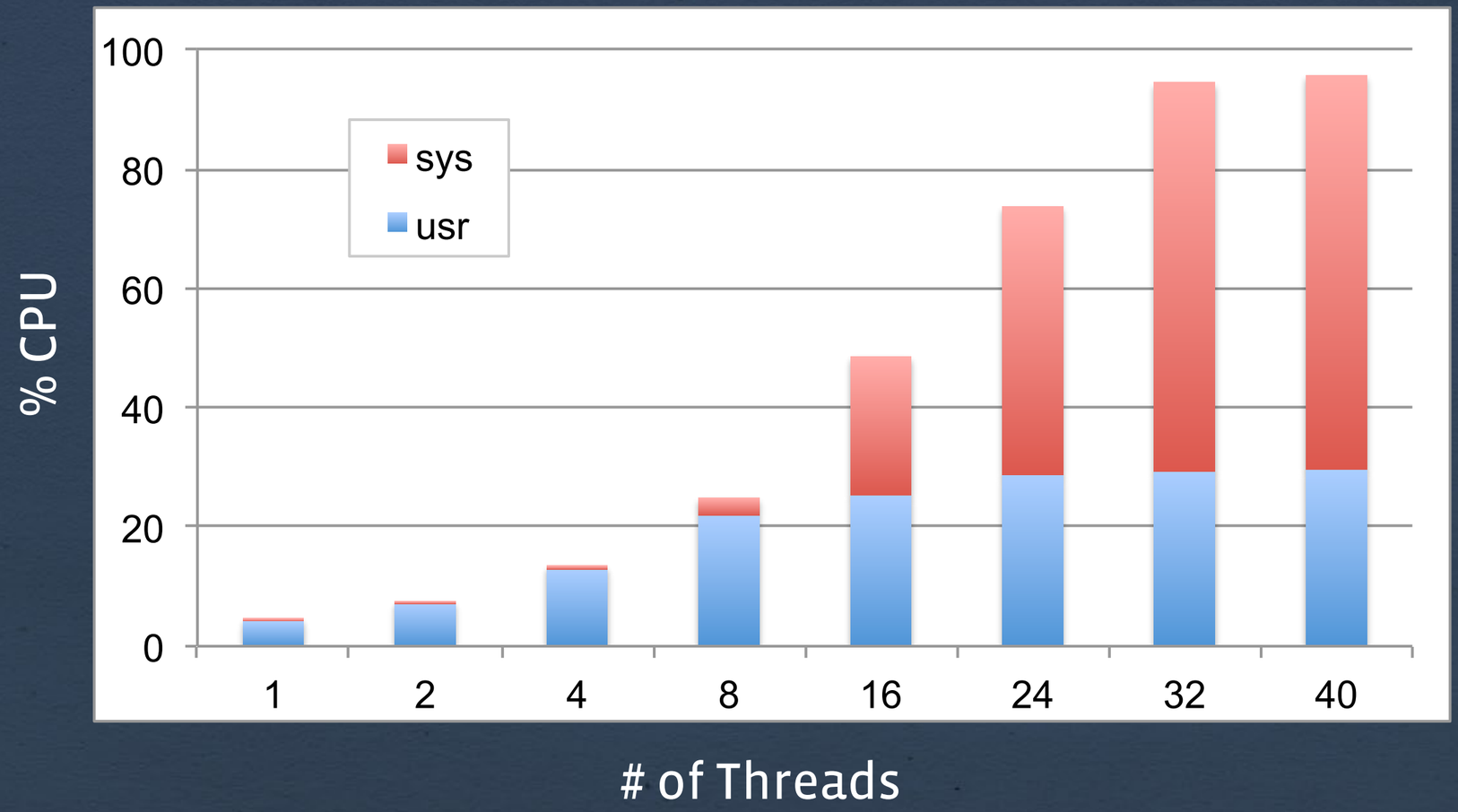
Transportation Security Administration

What's Changed in the Picture?



What's Changed in the Picture?

- Mutex becomes extremely expensive when storage access is fast



Under the Mutex

- What is done inside mutex?

```
mutex_.lock()
auto* sv = SuperVersion.ref()          // ++int64_t & pointer assignment
mutex_.unlock()

{ data retrievel }

mutex_.lock()
if (sv->unref()) {
   delete sv
}
mutex_.unlock()
```

Observation

- SuperVersion change is a relatively infrequent event

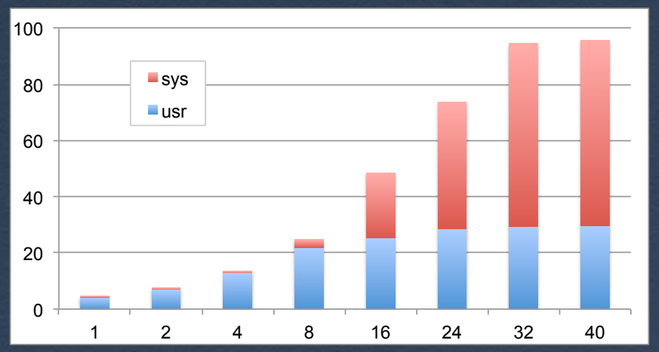
```
mutex_.lock()
auto* sv = SuperVersion.ref()
mutex_.unlock()

{ data retrievel }

mutex_.lock()
if (sv->unref()) {
   delete sv
}
mutex_.unlock()
```

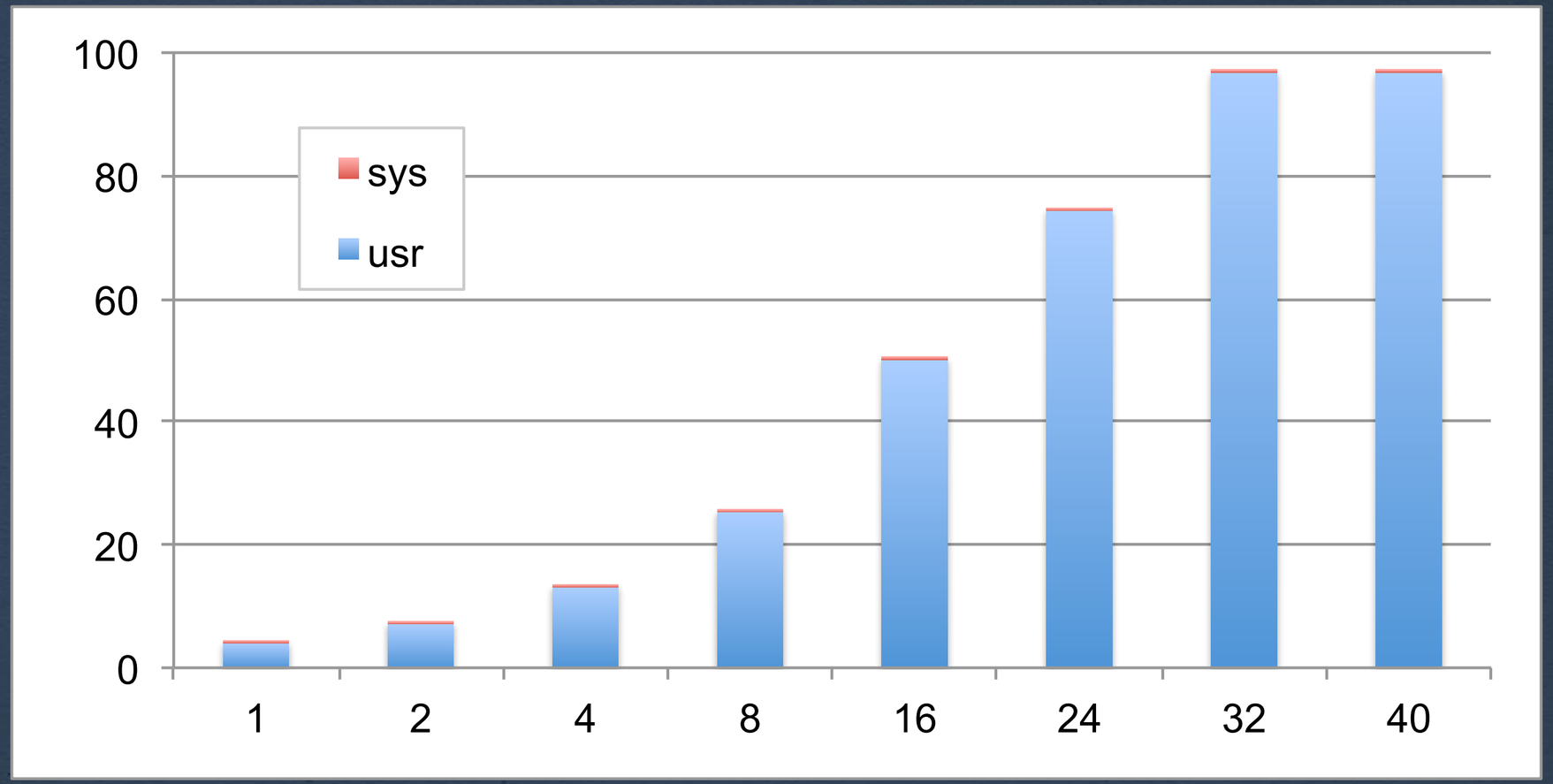
Proposed Solution

- SuperVersion change is a relatively infrequent event

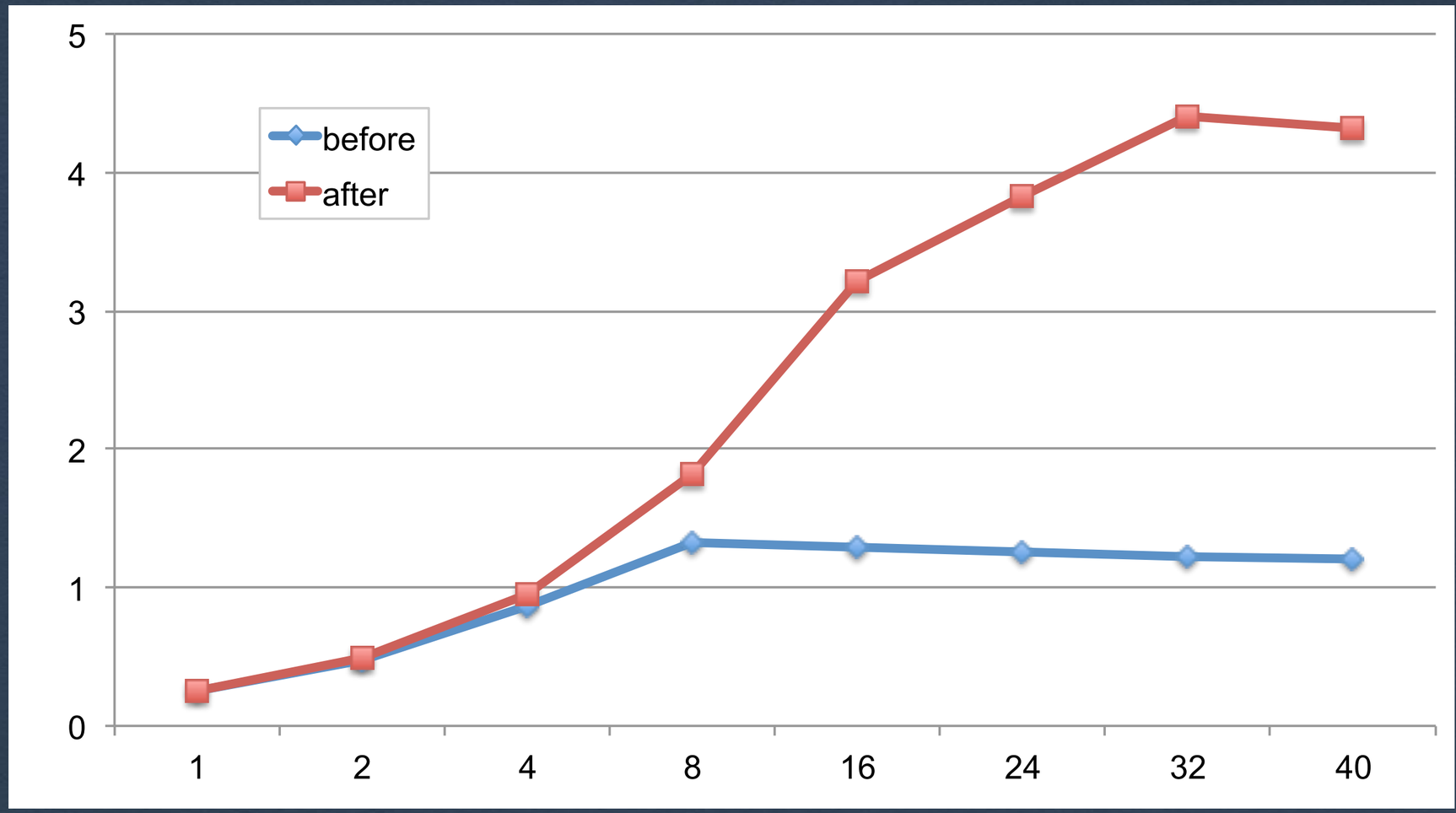
```
auto* sv = ThreadLocal.Get()

if (UNLIKELY(sv->version != global_version)) {
    mutex_.lock()
    auto* sv = SuperVersion.ref()
    mutex_.unlock()
}

{ data retrievel }

ThreadLocal.Put(sv)
```

CPU Breakdown on 32 Core Server

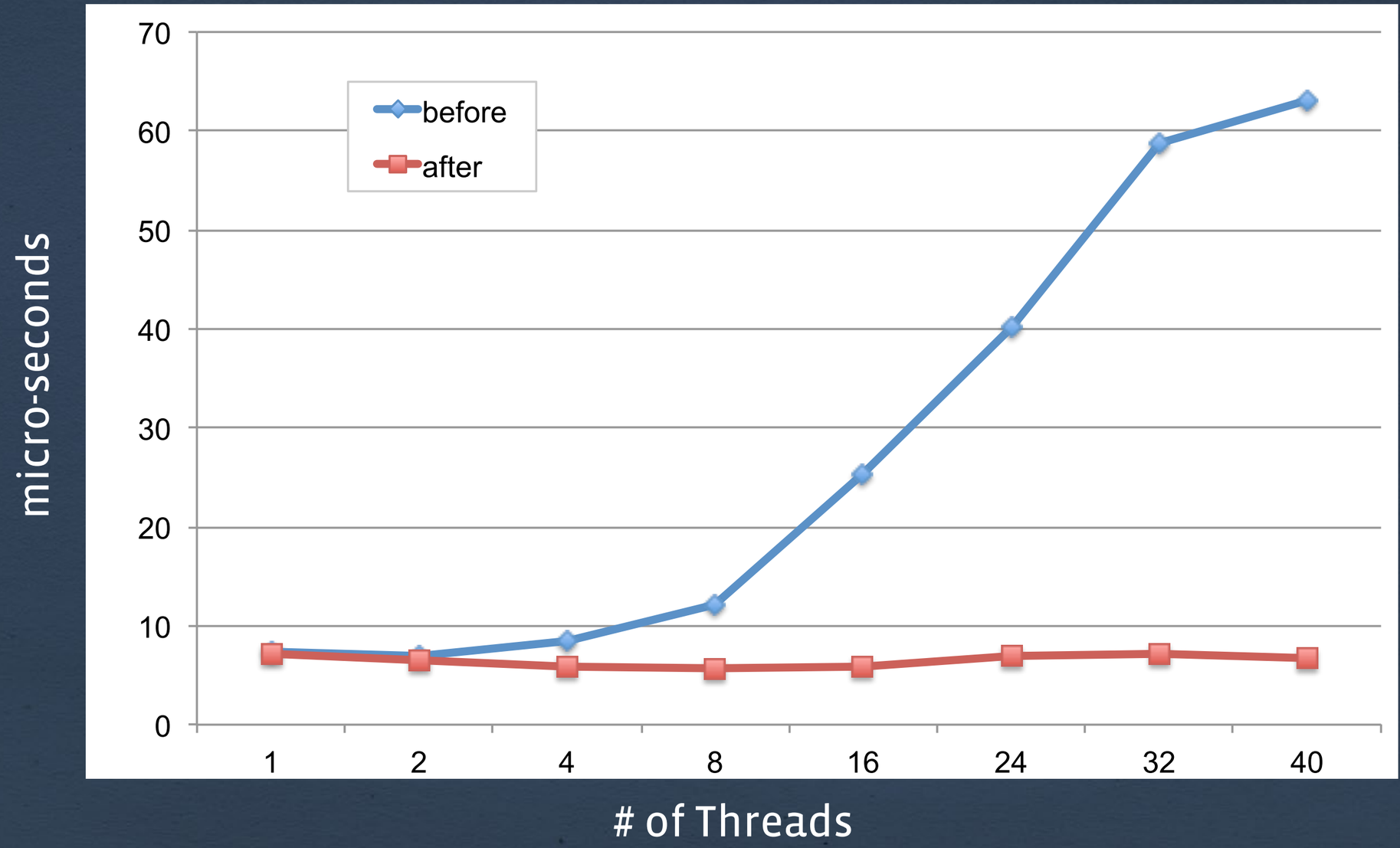




Get Performance on 32 Core Server



P99 latency on 32 Core Server



Experiment Setup

- Server
  - CPU @2.2GHz, 32 cores
  - 20480KB cache & 144GB ram
- Key 20 bytes, value 100 bytes
- Prefix 12 bytes, 10 keys per prefix
- No compression
- 500M keys loaded with filluniquerandom (~64G data)
- Readwhilewriting with 10k/s write speed
- No backup performed

Take away

- By separating read-only and read-write data structures in the design, RocksDB makes performance scaling much easier

- Future work
  - Iteration performance
  - Write performance
  - Improve cache efficiency for critical data structures