Yandex Cloud

KSTATE as [de]serializer of kernel's state

Linux Plumbers Conference 2025
Live Update MC
Andrey Ryabinin

Quick recap

- A common infrastructure for serializing, versioning, and restoring kernel state across live updates.
- [PATCH v3 0/7] KSTATE: [de]serialization framework for KHO
- KSTATE is designed around structs: it saves them in a structured, field-aware format and restores them exactly as they were.
- Description based - subsystem describes their data to preserve, and KSTATE handles serialization/restoring.

KSTATE description

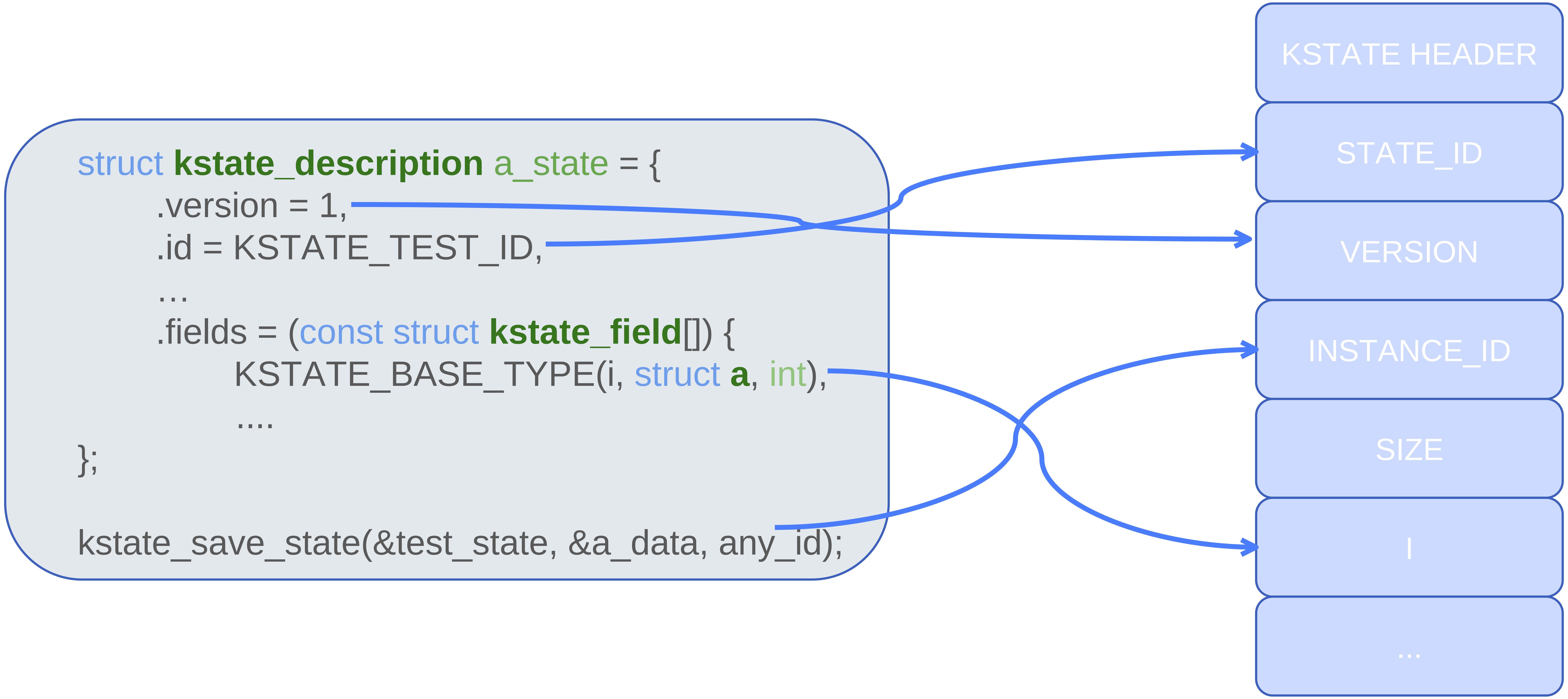
```
struct a {
        int i;
        unsigned long *p_ulong;
        char s[10];
        struct folio *folio;
};
```

```
static struct a a_data;

init_a(&a_data);
....
kstate_save_state(&a_state, &a_data, some_id);
/* after reboot: */
kstate_restore(&a_state, &a_data, some_id);
```

```
struct kstate_description a_state = {
        .name = "a_struct",
        .version_id = 1,
        .min_version_id = 1,
        .id = KSTATE_TEST_ID,
        .fields = (const struct kstate_field[]) {
                KSTATE_BASE_TYPE(i, struct a, int),
                KSTATE_BASE_TYPE(s, struct a, char [10]),
                KSTATE_POINTER(p_ulong, struct a),
                KSTATE_FOLIO(folio, struct a),
                KSTATE_END_OF_LIST()
        },
};
```

KSTATE binary format



```
struct kstate_description a_state = {
        .version = 1,
        .id = KSTATE_TEST_ID,
        ...
        .fields = (const struct kstate_field[]) {
                KSTATE_BASE_TYPE(i, struct a, int),
                ....
};

kstate_save_state(&test_state, &a_data, any_id);
```

KSTATE HEADER

STATE_ID

VERSION

INSTANCE_ID

SIZE

I

...

Plans for future (v4)

- Add more users, mainly LUO side (luo_global/session/flb)
- Drop some stuff (e.g. per-field version in description)
- Make it less struct-focused by providing lower level API

State ID type

State is “type” of preserved data, e.g. PCI device, LUO Session/FLB data, etc

Integer ID (u64)

- Compact and simple to use. Fixed size.
- Fast to compare and serialize/deserialize.
- Namespacing is hard.
- Vendor specific states – Are they needed, do we care?

String ID

- Harder to deal with, variable sizes
- Slower compare and higher cost of desireialization.
- Namespacing comes naturally, e.g. «vendorprefix-»

Instance ID type

An instance represents the state of a specific object, such as a particular PCI device.

- Different states may have different types of ID, e.g.:
  - LUO session uses string
  - LUO FD – u64 token
- Settle on one type? (u64 preferably).
  - Or add mapping u64 <-> string (offset to strings array/, CRC?)
- Or support different type of ID for differnet kinds of states?