Userspace memory persistence over kexec

Pratyush Yadav <pratyush@kernel.org>

Amazon Web Services

Agenda

- Why?
- How?
- Design
- Current state
- Future extensions

Agenda

- Why?
- How?
- Design
- Current state
- Future extensions

Why?

- You need to reboot to apply kernel patches.

Why?

- You need to reboot to apply kernel patches.
- For stateless hosts this isn't a big problem.

Why?

- You need to reboot to apply kernel patches.
- For stateless hosts this isn't a big problem.
- It is a bigger problem for stateful hosts like database servers or storage nodes.

Why?

- You need to reboot to apply kernel patches.
- For stateless hosts this isn't a big problem.
- It is a bigger problem for stateful hosts like database servers or storage nodes.
- Also useful if you don't control underlying workload.

Agenda

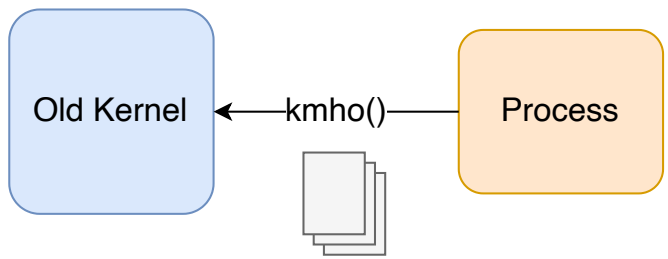
- Why?
- How?
- Design
- Current state
- Future extensions

How?

- Allow handing over userspace memory over kexec.
- Applications aware of being kexec-ed can serialize/deserialize state.
- For unaware applications, we can use Checkpoint/Restore in Userspace (CRIU).

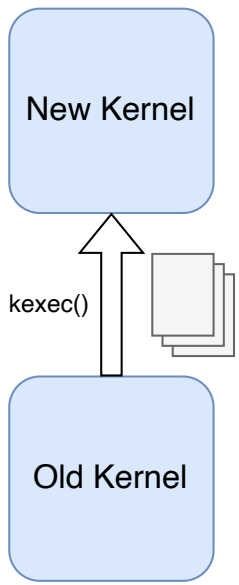
How?

How?



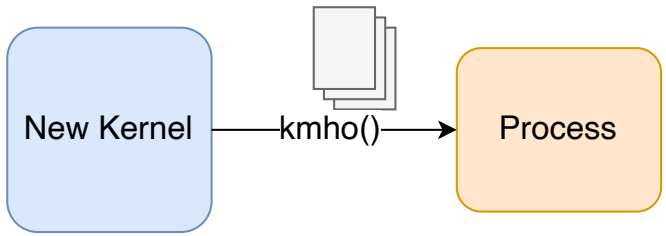
How?

How?

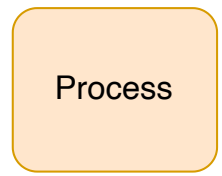


How?

How?



How?



Agenda

Why?

How?

Design

Current state

Future extensions

System call vs file system?

- Two ways to implement the feature.
- Similar file systems already proposed in the past like guestmemfs, pkram, pkernfs, etc.

System call

A new system call named `kmho()` with two modes of operation: take over memory and hand over memory.

```
int kmho(unsigned int opcode, void *op);
```

System call

For handing over memory (before kexec), one can call the `KMHO_HANDOVER` operation. `op` should be a `struct kmho_op_handover`.

```
struct kmho_range_handover {
        unsigned long base;
        unsigned long length;
};

struct kmho_op_handover {
        unsigned long key;
        unsigned long num_ranges;
        struct kmho_range_handover *ranges;
};
```

System call

An example call would look like:

```
struct kmho_range_handover range = {
        .base = base,
        .length = len,
};

struct kmho_op_handover op = {
        .key = 0xabcd1234,
        .num_ranges = 1,
        .ranges = &range,
};
kmho(KMHO_HANDOVER, &op);
```

System call

For taking over memory (after kexec), one can call the `KMHO_TAKEOVER` operation. `op` should be a `struct kmho_op_takeover`.

```
struct kmho_range_takeover {
        unsigned long base;
        unsigned long len;
        unsigned long remap_addr;
};

struct kmho_op_takeover {
        unsigned long key;
        unsigned long num_ranges;
        struct kmho_range_takeover *ranges;
};
```

System call

An example call would look like:

```
struct kmho_range_takeover range = {
        .base = base, // memory addr during handover
        .length = len,
        .remap_addr = new_addr, // New addr to map to
};

struct kmho_op_takeover op = {
        .key = 0xabcd1234,
        .num_ranges = 1,
        .ranges = &range,
};
kmho(KMHO_TAKEOVER, &op);
```

File system

Mount file system:

```
mount -t khofs none /khofs
```

Mapping memory would look like:

```
fd = open("/khofs/my_mem", O_RDWR | O_CREAT | O_EXCL,
          0600);
mem = mmap(NULL, length, PROT_READ | PROT_WRITE,
           MAP_SHARED, fd, 0);
// Do stuff...
munmap(mem, length);
close(fd);
```

Comparison

FS:

- Naming and permissions easier.
- Can use same old APIs.

Syscall:

- Using syscall is simpler.
- Not possible to have anonymous memory with FS.

Agenda

- Why?
- How?
- Design
- Current state
- Future extensions

Current state

- Implemented proof-of-concept using system call.
- Some hacky patches for CRIU to use this functionality.
- Plan to send out RFC soon.

Demo!

https://asciinema.org/a/3LZjzIe53Uvdhi7GenUxakrqy

Agenda

- Why?
- How?
- Design
- Current state
- Future extensions

Future extensions

- Handover swap contents across kexec.
- Handover page cache across kexec.

Thank you for attending the talk!

What about fragmentation?

- Yes, if you use `kmho()` the new kernel gets the old kernel's fragmentation.
- But that is an independent problem.
- We should solve that regardless of whether you use `kmho()` or not.

Do we really need a new system call?

- We can overload `mmap()`, `munmap()`, and `mremap()` to do this.
- This makes interface of these system calls more complex.
- Having a new system call keeps things nicely separated, and leaves more room for growth later.