One Year Later: And There are Still Things to Improve in Migration!

Red Hat
Juan Quintela

October 22, 2013

Abstract

This talk offers a description of what has changed during the last year on migration. And what you should expect in the future.

In search of the Latencies

The never ending story

Red Hat
Juan Quintela

October 22, 2013

Abstract

It looked easy, how long can it take to find downtimes of tens of seconds!!!

Agenda

1. What have we done?
2. What are the future plans?
3. In Search of the Latencies
4. Focus: migration
5. Anything I have forgot?
6. Questions

Section 1

What have we done?

redhat.

The last year

- Consolidation
- autoconverge (Vinod)
- rdma (mrhines)

redhat.

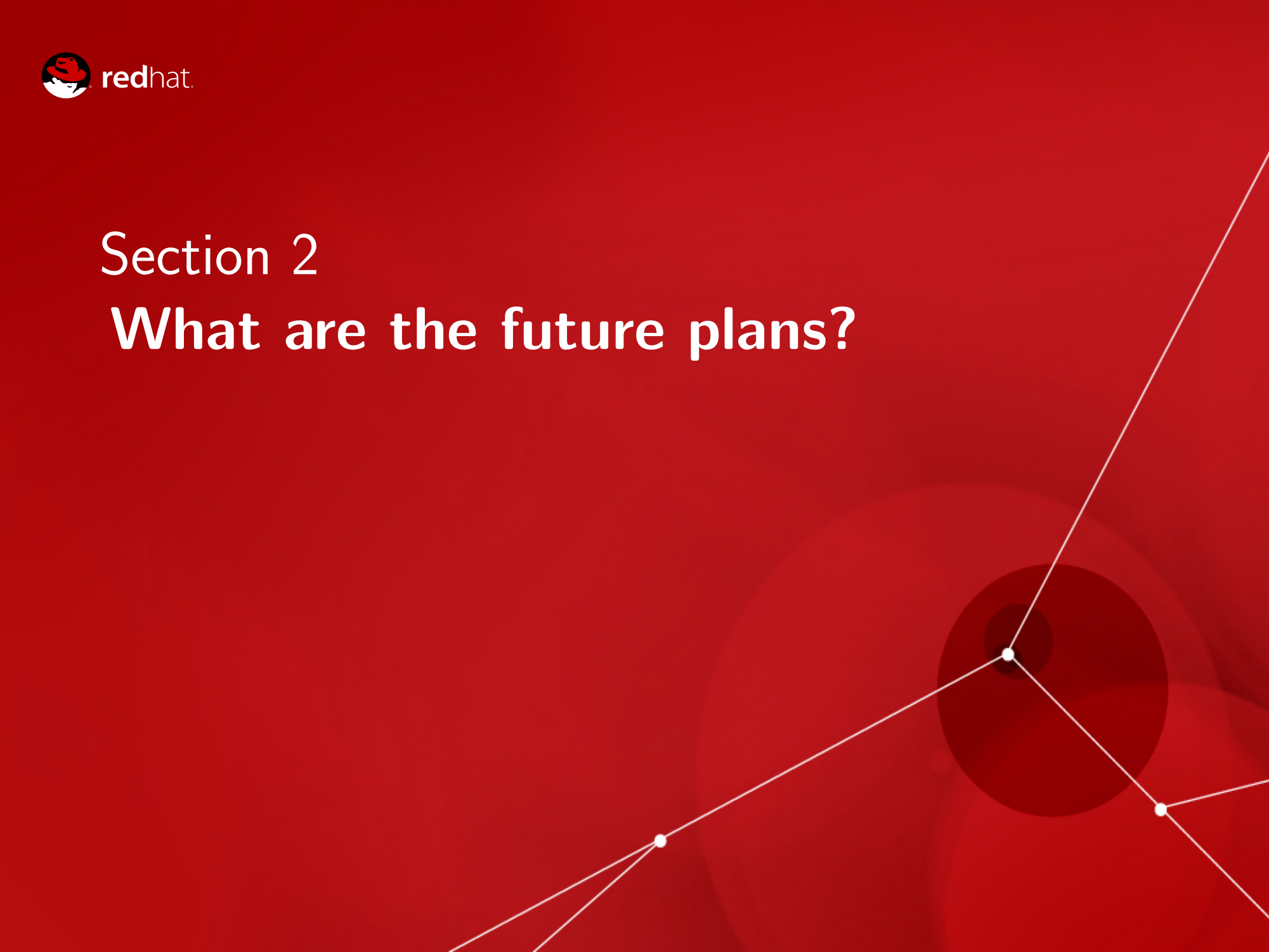
The last year

- Consolidation
- autoconverge (Vinod)
- rdma (mrhines)

redhat.

The last year

- Consolidation
- autoconverge (Vinod)
- rdma (mrhines)



Section 2

What are the future plans?

redhat.

Bitmap

- Patches posted to move from one byte/page to one bit/page
- Move to sync bitmaps, not bit a time

redhat.

Bitmap

- Patches posted to move from one byte/page to one bit/page
- Move to sync bitmaps, not bit a time

redhat.

Sync bitmap

Current code

```
for (i = 0; i < len; i++) {
    if (bitmap[i] != 0) {
            ....
            memory_region_set_dirty(section->mr, addr,
                                    TARGET_PAGE_SIZE * hpratio);
            ....
    }
}
```

New code

```
bitmap_or(bitmap, kvm_bitmap);
```

redhat.

For SSE/whatever gurus

Is there a simple way of doing

weird ord

```
  1010100011100110 (a)
  1100110100011100 (b)
--------------------
  1110110111111110 (a or  b)
```

And count all the 1's in **b** that are not already in **a**?

redhat.

Do we ever need more optimizations?

- Share the bitmap with migration and put a lock
- Update bitmap in place
- ....
- *Perhaps* it is a good idea to meassure first if it is still needed.

redhat.

Do we ever need more optimizations?

- Share the bitmap with migration and put a lock
- Update bitmap in place
- ....
- *Perhaps* it is a good idea to meassure first if it is still needed.

redhat.

Do we ever need more optimizations?

- Share the bitmap with migration and put a lock
- Update bitmap in place
- ....
- *Perhaps* it is a good idea to meassure first if it is still needed.

redhat.

Do we ever need more optimizations?

- Share the bitmap with migration and put a lock
- Update bitmap in place
- ....
- *Perhaps* it is a good idea to meassure first if it is still needed.

redhat.

`int` indexes?

look at `bits`

```
int slow_bitmap_and(unsigned long *dst, const unsigned long *bitmap1,
                    const unsigned long *bitmap2, int bits);
```

redhat.

Let's do some Math

- 2^{31} index/size
- $2^{31} * 4k$ pages $= 8TB max guest memory$
- $2^{31}/8 = 256MB$ bitmap
- Really, this is the 1st user of such a big bitmap

redhat.

Let's do some Math

- 2^{31} index/size
- $2^{31} * 4k$ pages $= 8 TB max guest memory$
- $2^{31}/8 = 256MB$ bitmap
- Really, this is the 1st user of such a big bitmap

redhat.

Let's do some Math

- 2^{31} index/size
- $2^{31} * 4k$ pages $= 8 TB max guest memory$
- $2^{31}/8 = 256 MB$ bitmap
- Really, this is the 1st user of such a big bitmap

redhat.

Let's do some Math

- 2^{31} index/size
- $2^{31} * 4k$ pages $= 8\,TB max guest memory$
- $2^{31}/8 = 256MB$ bitmap
- Really, this is the 1st user of such a big bitmap

redhat.

Post-copy migration

- Andrea post patches for the Kernel side
- We still need to use that interface on userspace side

redhat.

Post-copy migration

- Andrea post patches for the Kernel side
- We still need to use that interface on userspace side

redhat.

Fault tolerance

- Kemari (orit)
- Curling (jules)
- micro-checkpointing (mrhines)
- COLO (intel)

redhat.

Fault tolerance

- Kemari (orit)
- Curling (jules)
- micro-checkpointing (mrhines)
- COLO (intel)

redhat.

Fault tolerance

- Kemari (orit)
- Curling (jules)
- micro-checkpointing (mrhines)
- COLO (intel)

redhat.

Fault tolerance

- Kemari (orit)
- Curling (jules)
- micro-checkpointing (mrhines)
- COLO (intel)

redhat.

Inter Version Mess

- New → Old
- Old → New
- static checker (amit)
- debugging live migration (alex)

redhat.

Inter Version Mess

- New $\rightarrow$ Old
- Old $\rightarrow$ New
- static checker (amit)
- debugging live migration (alex)

redhat.

Inter Version Mess

- New → Old
- Old → New
- static checker (amit)
- debugging live migration (alex)

redhat.

Inter Version Mess

- New $\rightarrow$ Old
- Old $\rightarrow$ New
- static checker (amit)
- debugging live migration (alex)

redhat.

Automatic testing

- We have virt-test
- And a lot of tests to add
- cpu_physical_memory_*
- vmstate machinery
- subsection stuff
- ...

redhat.

Automatic testing

- We have virt-test
- And a lot of tests to add
- cpu_physical_memory_*
- vmstate machinery
- subsection stuff
- ...

redhat.

Automatic testing

- We have virt-test
- And a lot of tests to add
- cpu_physical_memory_*
- vmstate machinery
- subsection stuff
- ...

redhat.

Automatic testing

- We have virt-test
- And a lot of tests to add
- cpu_physical_memory_*
- vmstate machinery
- subsection stuff
- ...

redhat.

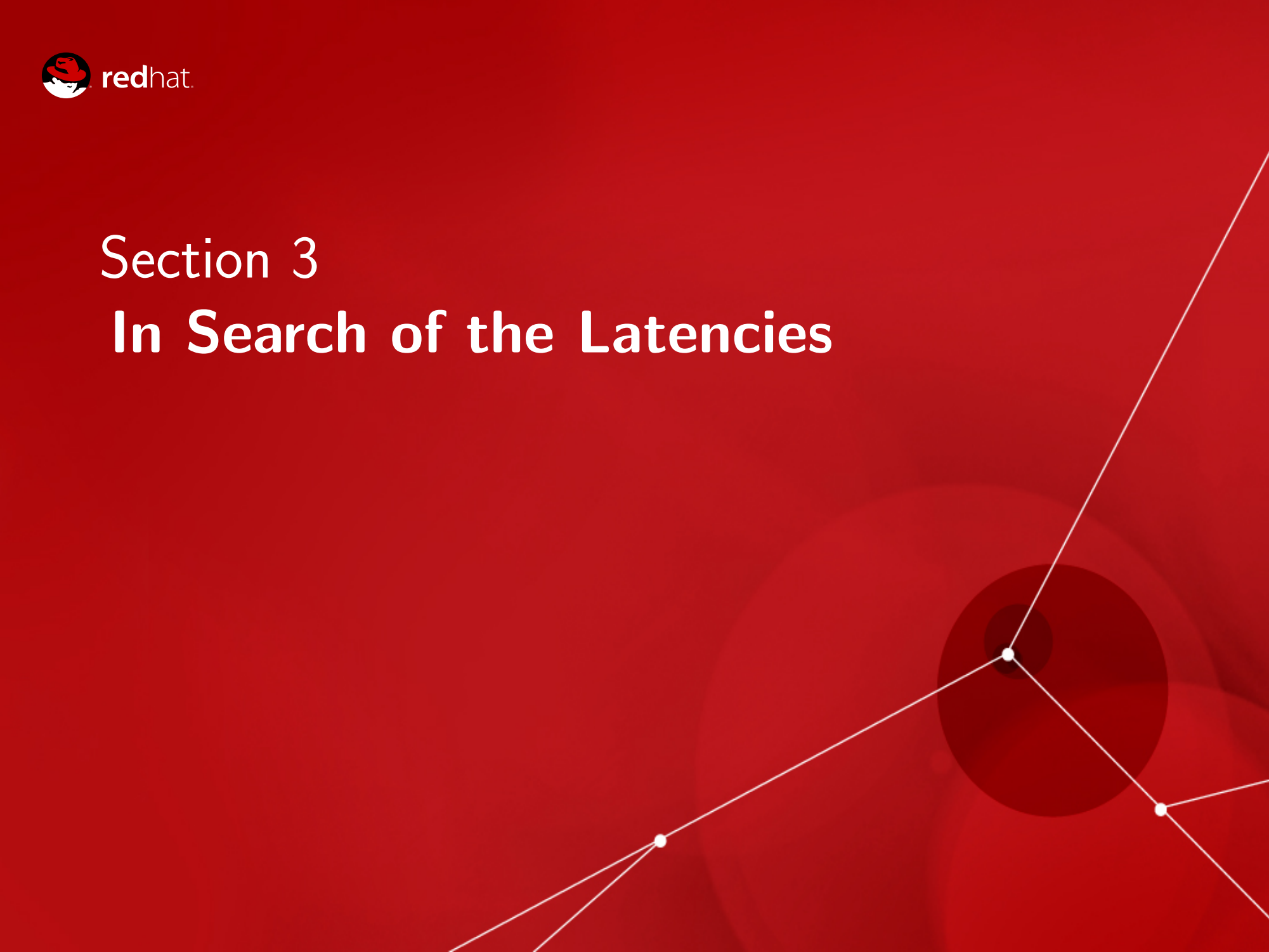
Automatic testing

- We have virt-test
- And a lot of tests to add
- cpu_physical_memory_*
- vmstate machinery
- subsection stuff
- ...

redhat.

Automatic testing

- We have virt-test
- And a lot of tests to add
- cpu_physical_memory_*
- vmstate machinery
- subsection stuff
- ...



Section 3

In Search of the Latencies

redhat.

Setup

- This testing/search was done on RHEL6
- 0.12 with lots of backporting
- parts of it still relevant to upstream

redhat.

Setup

- This testing/search was done on RHEL6
- 0.12 with lots of backporting
- parts of it still relevant to upstream

redhat.

Setup

- This testing/search was done on RHEL6
- 0.12 with lots of backporting
- parts of it still relevant to upstream

redhat.

The history

There have been reports of weird latencies that happens during migration but it always happened:

- Lots of vcpus
- Lots of RAM
- Weird storage
- And not reproducible

redhat.

The history

There have been reports of weird latencies that happens during migration but it always happened:

- Lots of vcpus
- Lots of RAM
- Weird storage
- And not reproducible

redhat.

The history

There have been reports of weird latencies that happens during migration but it always happened:

- Lots of vcpus
- Lots of RAM
- Weird storage
- And not reproducible

redhat.

The history

There have been reports of weird latencies that happens during migration but it always happened:

- Lots of vcpus
- Lots of RAM
- Weird storage
- And not reproducible

redhat.

There are users and users

And this one was very stubborn, and was able to reproduce it with

- 1GB RAM
- 1 VCPU
- Just running a program that dirtied memory
- And he/she was able to reproduce

redhat.

There are users and users

And this one was very stubborn, and was able to reproduce it with

- 1GB RAM
- 1 VCPU
- Just running a program that dirtied memory
- And he/she was able to reproduce

redhat.

There are users and users

And this one was very stubborn, and was able to reproduce it with

- 1GB RAM
- 1 VCPU
- Just running a program that dirtied memory
- And he/she was able to reproduce

redhat.

There are users and users

And this one was very stubborn, and was able to reproduce it with

- 1GB RAM
- 1 VCPU
- Just running a program that dirtied memory
- And he/she was able to reproduce

redhat.

So, here we go

This looked really, really easy, but (there is always a but)

- I was not able to reproduce
- After some twisting of arms, I got their network configuration
- And ... I would call it broken
- it happened when there was congestion
- I had to simulate seting network card to 100Mbit/s
- ..... after waiting 4-5hours for a migration that didn't ended

redhat.

So, here we go

This looked really, really easy, but (there is always a but)

- I was not able to reproduce
- After some twisting of arms, I got their network configuration
- And ... I would call it broken
- it happened when there was congestion
- I had to simulate seting network card to 100Mbit/s
- ..... after waiting 4-5hours for a migration that didn't ended

redhat.

So, here we go

This looked really, really easy, but (there is always a but)

- I was not able to reproduce
- After some twisting of arms, I got their network configuration
- And ... I would call it broken
- it happened when there was congestion
- I had to simulate seting network card to 100Mbit/s
- ..... after waiting 4-5hours for a migration that didn't ended

redhat.

So, here we go

This looked really, really easy, but (there is always a but)

- I was not able to reproduce
- After some twisting of arms, I got their network configuration
- And ... I would call it broken
- it happened when there was congestion
- I had to simulate seting network card to 100Mbit/s
- ..... after waiting 4-5hours for a migration that didn't ended

redhat.

So, here we go

This looked really, really easy, but (there is always a but)

- I was not able to reproduce
- After some twisting of arms, I got their network configuration
- And ... I would call it broken
- it happened when there was congestion
- I had to simulate seting network card to 100Mbit/s
- ..... after waiting 4-5hours for a migration that didn't ended

redhat.

So, here we go

This looked really, really easy, but (there is always a but)

- I was not able to reproduce
- After some twisting of arms, I got their network configuration
- And ... I would call it broken
- it happened when there was congestion
- I had to simulate seting network card to 100Mbit/s
- ..... after waiting 4-5hours for a migration that didn't ended

redhat.

A bit of good luck

- After ended a day without luck
- During poweroff the problem happens
- And it is reproductible
- notice that I have the network set to 100mbit
- at 1GB freeze don't exist

redhat.

A bit of good luck

- After ended a day without luck
- During poweroff the problem happens
- And it is reproductible
- notice that I have the network set to 100mbit
- at 1GB freeze don't exist

redhat.

A bit of good luck

- After ended a day without luck
- During poweroff the problem happens
- And it is reproductible
- notice that I have the network set to 100mbit
- at 1GB freeze don't exist

redhat.

A bit of good luck

- After ended a day without luck
- During poweroff the problem happens
- And it is reproductible
- notice that I have the network set to 100mbit
- at 1GB freeze don't exist

redhat.

A bit of good luck

- After ended a day without luck
- During poweroff the problem happens
- And it is reproductible
- notice that I have the network set to 100mbit
- at 1GB freeze don't exist

redhat.

Let's start

After fixing all the migration paths, nothing really get solved

- So, we go tried to look where the time was spent
- And there is no trace that shows how long input handlers take to run
- or how long a vcpu takes between an exit and a re-enter
- (remember we have 1VCPU and 1GB RAM, no overcommit of anything)

redhat.

Let's start

After fixing all the migration paths, nothing really get solved

- So, we go tried to look where the time was spent
- And there is no trace that shows how long input handlers take to run
- or how long a vcpu takes between an exit and a re-enter
- (remember we have 1VCPU and 1GB RAM, no overcommit of anything)

redhat.

Let's start

After fixing all the migration paths, nothing really get solved

- So, we go tried to look where the time was spent
- And there is no trace that shows how long input handlers take to run
- or how long a vcpu takes between an exit and a re-enter
- (remember we have 1VCPU and 1GB RAM, no overcommit of anything)

redhat.

Let's start

After fixing all the migration paths, nothing really get solved

- So, we go tried to look where the time was spent
- And there is no trace that shows how long input handlers take to run
- or how long a vcpu takes between an exit and a re-enter
- (remember we have 1VCPU and 1GB RAM, no overcommit of anything)

redhat.

Instrumentation? What is that?

I ended with something like this.

printf

```
struct timespec start, end;
uint64_t t0;

clock_gettime(CLOCK_REALTIME, &start);
foo();
clock_gettime(CLOCK_REALTIME, &end);
t0 = (end.tv_sec - start.tv_sec) * 1000
   + (end.tv_nsec - start.tv_nsec)/ 1000000;
if (t0 > 100) {
  printf("foo: %lu ms\n", t0);
}
g();
clock_gettime(CLOCK_REALTIME, &end);
t0 = (end.tv_sec - start.tv_sec) * 1000
   + (end.tv_nsec - start.tv_nsec)/ 1000000;
if (t0 > 100) {
  printf("bar: %lu ms\n", t0);
}
```

redhat.

And there are time spent on io_handlers

It could take more than 1 second during migration

- They can block the io_thread for more than 1second during migration
- They block the io_thread for more than than 150ms out of migration

redhat.

And there are time spent on io_handlers

It could take more than 1 second during migration

- They can block the io_thread for more than 1second during migration
- They block the io_thread for more than than 150ms out of migration

redhat.

Non migration case

- On one hand why io_handlers took so long
- and we go back to migration

redhat.

Non migration case

- On one hand why io_handlers took so long
- and we go back to migration

redhat.

After hunting lots and lots

- qemu_aio_wait() has a select() without timeout
- and NFS over a saturated 100Mbit takes a long time

redhat.

After hunting lots and lots

- qemu_aio_wait() has a select() without timeout
- and NFS over a saturated 100Mbit takes a long time

redhat.

Time to work

- You can't debug performance problems with gdb, as if you stop it to see what function is taking too much, you are changing times too much
- so we are back to the old time of printf
- putting printf's all around qemu takes forever
- so, adding printf's, see where the time is being spent, and then instrument that function.
- This works, but it is slow

redhat.

Time to work

- You can't debug performance problems with gdb, as if you stop it to see what function is taking too much, you are changing times too much
- so we are back to the old time of printf
- putting printf's all around qemu takes forever
- so, adding printf's, see where the time is being spent, and then instrument that function.
- This works, but it is slow

redhat.

Time to work

- You can't debug performance problems with gdb, as if you stop it to see what function is taking too much, you are changing times too much
- so we are back to the old time of printf
- putting printf's all around qemu takes forever
- so, adding printf's, see where the time is being spent, and then instrument that function.
- This works, but it is slow

redhat.

Time to work

- You can't debug performance problems with gdb, as if you stop it to see what function is taking too much, you are changing times too much
- so we are back to the old time of printf
- putting printf's all around qemu takes forever
- so, adding printf's, see where the time is being spent, and then instrument that function.
- This works, but it is slow

redhat.

Time to work

- You can't debug performance problems with gdb, as if you stop it to see what function is taking too much, you are changing times too much
- so we are back to the old time of printf
- putting printf's all around qemu takes forever
- so, adding printf's, see where the time is being spent, and then instrument that function.
- This works, but it is slow

redhat.

1st additions

Just io_read

```
      qemu_mutex_lock_iothread();
      if (ret > 0) {
          IOHandlerRecord *pioh;
+struct timespec start, end;
+uint64_t t0;

          QLIST_FOREACH(ioh, &io_handlers, next) {
              if (!ioh->deleted && ioh->fd_read && FD_ISSET(ioh->fd, &rfds)) {
+clock_gettime(CLOCK_REALTIME, &start);
                  ioh->fd_read(ioh->opaque);
+clock_gettime(CLOCK_REALTIME, &end);
+t0 = (end.tv_sec - start.tv_sec) * 1000
+    + (end.tv_nsec - start.tv_nsec)/ 1000000;
+if (t0 > 100) {
+   printf("io_read: %lu ms\n", t0);
+}
+
```

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read: 0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr: 3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read: 0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr: 3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read: 0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr: 3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read: 0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr: 3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read: 0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr: 3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read:  0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr:  3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read:  0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr:  3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read:  0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr:  3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read:  0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr:  3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

An unexpected guest

- pun intended
- Some of the iohandlers take more than 100ms
- we only have pointers at that point, getting names gets interesting, but I digress
- `fd_read:  0x7ffff7df2390 358 ms`
- a read io_handler is taking 358ms. I have seen so much as 800ms. Notice that this is without migration, without playing with network characteristics, ...
- Investigation continues
- `vphnr:  3 358 ms 0x7ffff915e440`
- this is `virtio_pci_host_notifier_read` for you
- /me starts blamethrower, clearly virtio_net stuff
- just one last check....

redhat.

A plot twist

- `vbho 2:  t0 358 t1 358 ms 16 num_writes`
- This is `virtio_block_handle_other` for you.
- problem are `VIRTIO_BLK_T_OUT` and `OTHER`

redhat.

A plot twist

- `vbho 2:  t0 358 t1 358 ms 16 num_writes`
- This is `virtio_block_handle_other` for you.
- problem are `VIRTIO_BLK_T_OUT` and `OTHER`

redhat.

A plot twist

- `vbho 2:  t0 358 t1 358 ms 16 num_writes`
- This is `virtio_block_handle_other` for you.
- problem are `VIRTIO_BLK_T_OUT` and `OTHER`

redhat.

virtio-net

```
static void virtio_net_handle_tx_bh(VirtIODevice *vdev, VirtQueue *vq)
{
    VirtIONet *n = VIRTIO_NET(vdev);
    VirtIONetQueue *q = &n->vqs[vq2q(virtio_get_queue_index(vq))];

    if (unlikely(q->tx_waiting)) {
        return;
    }
    q->tx_waiting = 1;
    /* This happens when device was stopped but VCPU wasn't. */
    if (!vdev->vm_running) {
        return;
    }
    virtio_queue_set_notification(vq, 0);
    qemu_bh_schedule(q->tx_bh);
}
```

redhat.

virtio-blk

```
static void virtio_blk_dma_restart_bh(void *opaque)
{
    VirtIOBlock *s = opaque;
    VirtIOBlockReq *req = s->rq;
    MultiReqBuffer mrb = {
        .num_writes = 0,
    };

    qemu_bh_delete(s->bh);
    s->bh = NULL;

    s->rq = NULL;

    while (req) {
        virtio_blk_handle_request(req, &mrb);
        req = req->next;
    }

    virtio_submit_multiwrite(s->bs, &mrb);
}
```

Section 4

Focus: migration

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(....., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(....., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(....., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(....., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(...., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(...., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(...., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Our problem was with migration

- This was more complicated that it looks, but at the end, investigations ended with:
- migration calls `bdrv_flush_all()`
- `bdrv_flush_all()` calls `qemu_aio_flush()`
- `qemu_aio_flush()` calls `qemu_aio_wait()`
- `qemu_aio_wait()` calls `select(...., NULL)`
- from the iothread
- I have meassured that select() to take 40-50 seconds.
- yes, unit is right, seconds, not milliseconds

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

Proposal: Put a timeout in the select

- For migration (notice only migration), put a timeout in the select, if we get out through the timeout, just got back to the iterative stage
- For upstream: we need to put the timeout always
- And audit all the callers
- And probably add a coroutine
- And problably we need a new toplevel state: stopped waiting for IO to finish
- And ....
- Guess where I am stuck right now

redhat.

This is where things are?

```
io_read: 13 0x7ffff7e1da40
qaw 2: 9010 ms
qaw 3: 10011 ms rfd 1 wfd 0
io_read: 13 0x7ffff7e1da40
qaw 2: 10011 ms
qaw 3: 11012 ms rfd 1 wfd 0
io_read: 13 0x7ffff7e1da40
qaw 2: 11012 ms
qaw 3: 12013 ms rfd 1 wfd 0
io_read: 13 0x7ffff7e1da40
qaw 2: 12013 ms
qaw 3: 13014 ms rfd 1 wfd 0
io_read: 13 0x7ffff7e1da40
qaw 2: 13014 ms
qaw 3: 14015 ms rfd 1 wfd 0
io_read: 13 0x7ffff7e1da40
```

redhat.

Back to the user

- Get back to iterative stage when taking so long
- Problem fixed, right?
- No, it was enough for getting ping to answer, but not for nothing that runs on userspace
- So we ended puting a limit on how soon we can get back to the completion stage

redhat.

Back to the user

- Get back to iterative stage when taking so long
- Problem fixed, right?
- No, it was enough for getting ping to answer, but not for nothing that runs on userspace
- So we ended puting a limit on how soon we can get back to the completion stage

redhat.

Back to the user

- Get back to iterative stage when taking so long
- Problem fixed, right?
- No, it was enough for getting ping to answer, but not for nothing that runs on userspace
- So we ended puting a limit on how soon we can get back to the completion stage

redhat.

Back to the user

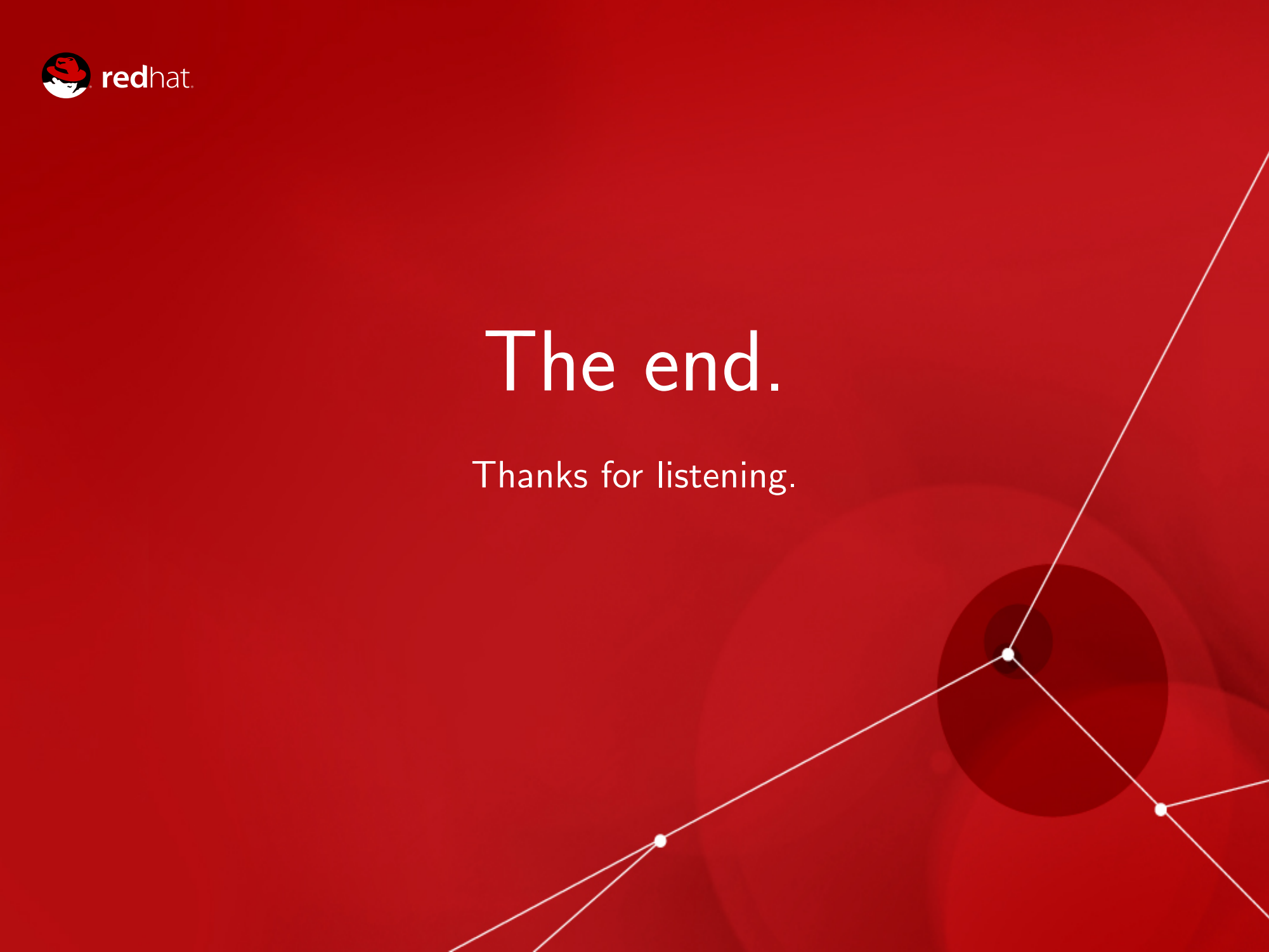
- Get back to iterative stage when taking so long
- Problem fixed, right?
- No, it was enough for getting ping to answer, but not for nothing that runs on userspace
- So we ended puting a limit on how soon we can get back to the completion stage

Section 5

Anything I have forgot?

Section 6

Questions


redhat.
The end.
Thanks for listening.