Linux Xen Balloon Driver Improvement

Wei Liu <wei.liu2@citrix.com>

Draft 2

Contents

Version	Date	Changes
2	24/10/2014	Style fixes, more clarifications.
1	22/10/2014	Initial version.

Introduction

This document describe a design to improve Xen balloon driver in Linux.

Motives

1. Balloon pages fragments guest physical address space.

2. Balloon compaction infrastructure can migrate ballooned pages from start of Linux memory zone to end of zone, hence creating contiguous guest physical address space.
3. Having contiguous guest physical address enables some options to improve performance.

Goal of improvement

The balloon driver makes use of as many huge pages as possible, defragmenting guest address space. Contiguous guest address space permits huge page ballooning which helps prevent host address space fragmentation.

This should be achieved without any particular hypervisor side feature.

Design and implementation

When the balloon driver is asked to increase / decrease reservation, it will always start with a huge page. However, due to resource availability in both hypervisor and guest, it's not always possible to get hold of a huge page. In that case the driver will fall back to use normal size page. Balloon driver later will try to coalesce small size pages into huge page. As time goes by, both Xen and guest should use more and more huge pages.

To achieve the said goal, several changes will be made:

1. Make use of balloon page compaction.
2. Maintain multiple queues for pages of different sizes and purposes.
3. Periodically exchange normal size pages with huge pages.

Make use of balloon page compaction

Balloon page migration moves balloon pages from start of zone to end of zone, making guest physical address space contiguous. This gives balloon driver a chance to allocate huge pages in order to coalesce small pages.

Currently, Xen balloon driver gets its page directly from page allocator. To enable balloon page migration, those pages now need to be allocated from core balloon driver. Pages allocated from core balloon driver are subject to balloon page compaction.

The use of Linux balloon page compaction doesn't require introducing new interfaces between Xen balloon driver and the rest of the system. Most changes are internal to Xen balloon driver.

Xen balloon driver will also need to provide a callback to migrate balloon page. In essence callback function receives "old page", which is a already ballooned out page, and “new page”, which is a page to be ballooned out, then it inflates “old page” and deflates “new page”.

The core of migration callback is XENMEM_exchange hypercall. This makes sure that inflation of old page and deflation of new page is done atomically, so even if a domain is beyond its memory target and the target is being enforced, it can still compact memory.

Maintain multiple queues for pages of different sizes and purposes

We maintain multiple queues for pages of different sizes inside Xen balloon driver, so that Xen balloon worker thread can coalesce smaller size pages into one larger size page. Queues for special purposed pages, such as balloon pages used to map foreign pages, are also maintained. These special purposed pages are not subject to migration and page coalescence.

For instance, balloon driver can maintain three queues:

1. queue for 2 MB pages
2. queue for 4 KB pages (delegated to core balloon driver)
3. queue for pages used to mapped pages from other domain

More queues can be added when necessary, but for now one queue for normal pages and one queue for huge page should be enough.

Periodically exchange normal size pages with huge pages

Worker thread wakes up periodically to check if there are enough pages in normal size page queue to coalesce into a huge page. If so, it will try to exchange that huge page into a number of normal size pages with XENMEM_exchange hypercall.

Relationship with NUMA-aware ballooning

Another orthogonal improvement to Linux balloon driver is NUMA-aware ballooning.

The use of balloon page compaction will not interfere with NUMA-ware ballooning because balloon compaction, which is part of Linux’s memory subsystem, is already NUMA-aware.

All the changes proposed in this design can be made NUMA-aware provided virtual NUMA topology information is in place.

Flowcharts

These flowcharts assume normal page size is 4K and huge page size is 2M. They show how two queues are maintained. Please note that “requeue on failure” is not drawn on the flowcharts to make the flowcharts easier to reason.

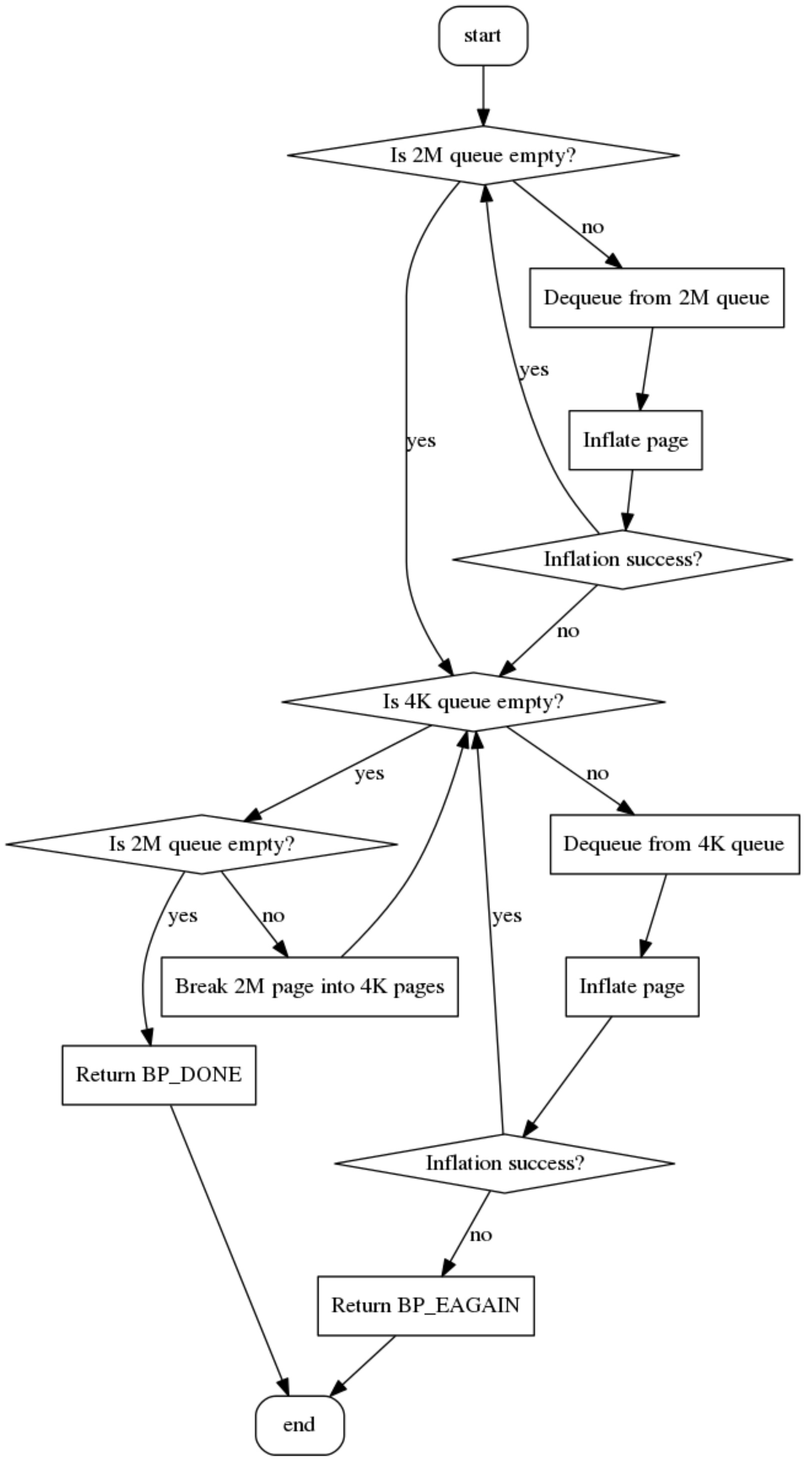

Figure 1: Increase Reservation

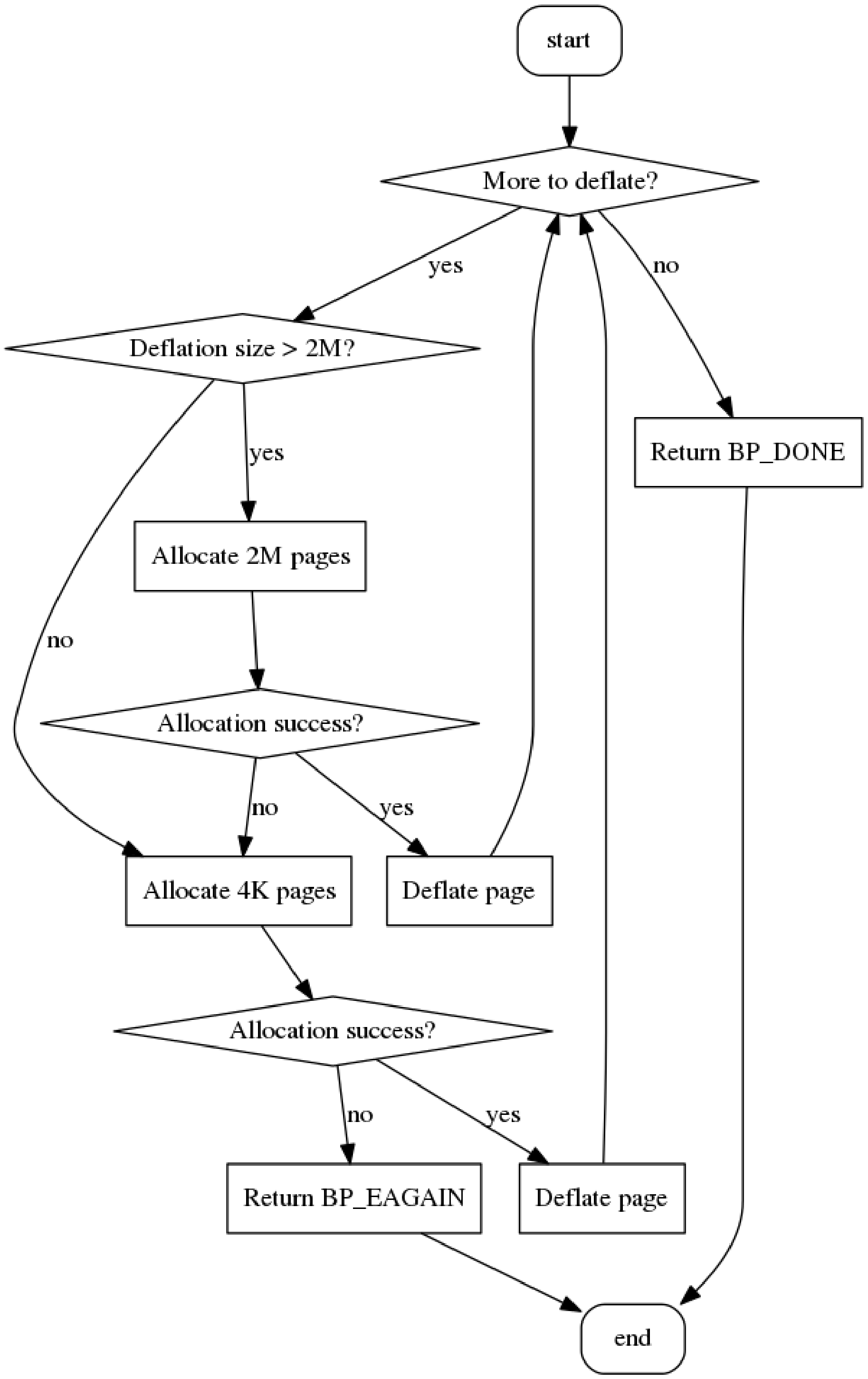

Figure 2: Decrease Reservation

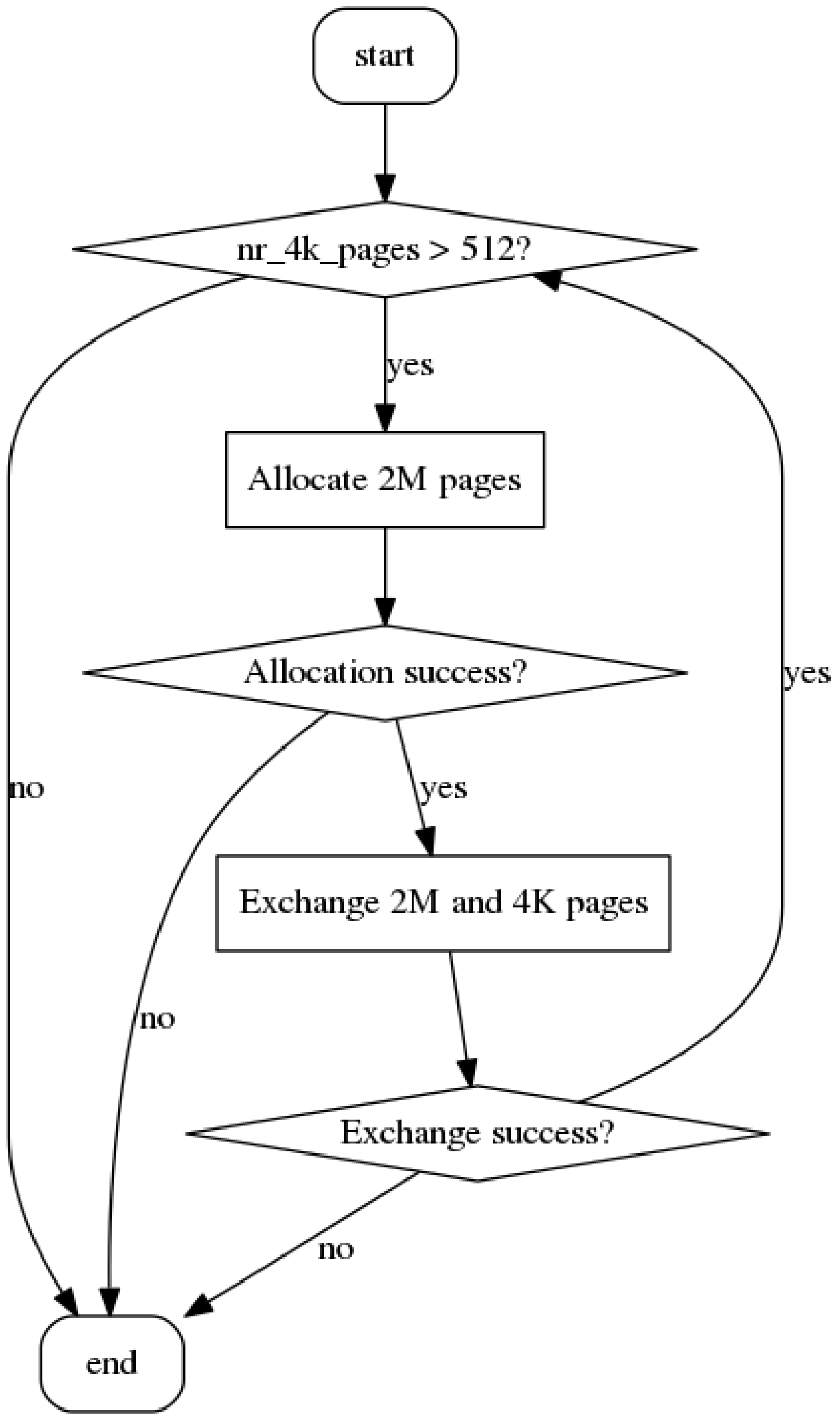

Figure 3: Exchange Pages