Subject: [RFC][for -mm] memory controller enhancements for NUMA [8/10] move reclaim_mapped calc routine (cle
Posted by KAMEZAWA Hiroyuki on Wed, 14 Nov 2007 08:53:06 GMT
View Forum Message <> Reply to Message

Just for clean up for later patch for avoiding dirty nesting....

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

```
 mm/vmscan.c |  184 ++++++++++++++++++++++++++++++++++++++++----------------------------------
 1 file changed, 97 insertions(+), 87 deletions(-)

Index: linux-2.6.24-rc2-mm1/mm/vmscan.c
===================================================================
--- linux-2.6.24-rc2-mm1.orig/mm/vmscan.c
+++ linux-2.6.24-rc2-mm1/mm/vmscan.c
@@ -950,6 +950,98 @@ static inline int zone_is_near_oom(struc
 }

 /*
+ * Determine we should try to reclaim mapped pages.
+ */
+static int calc_reclaim_mapped(struct zone *zone, int priority, int swappiness)
+{
+ long mapped_ratio;
+ long distress;
+ long swap_tendency;
+ long imbalance;
+ int reclaim_mapped;
+
+ if (zone_is_near_oom(zone))
+  return 1;
+ /*
+  * `distress' is a measure of how much trouble we're having
+  * reclaiming pages.  0 -> no problems.  100 -> great trouble.
+  */
+ distress = 100 >> min(zone->prev_priority, priority);
+
+ /*
+  * The point of this algorithm is to decide when to start
+  * reclaiming mapped memory instead of just pagecache.  Work out
+  * how much memory
+  * is mapped.
+  */
+ mapped_ratio = ((global_page_state(NR_FILE_MAPPED) +
+   global_page_state(NR_ANON_PAGES)) * 100) /
+    vm_total_pages;
+ /*
+  * Now decide how much we really want to unmap some pages.  The
+  * mapped ratio is downgraded - just because there's a lot of
+  * mapped memory doesn't necessarily mean that page reclaim
+  * isn't succeeding.
+  *
+  * The distress ratio is important - we don't want to start
+  * going oom.
+  *
+  * A 100% value of vm_swappiness overrides this algorithm
+  * altogether.
+  */
+ swap_tendency = mapped_ratio / 2 + distress + swappiness;
+
+ /*
+  * If there's huge imbalance between active and inactive
+  * (think active 100 times larger than inactive) we should
+  * become more permissive, or the system will take too much
+  * cpu before it start swapping during memory pressure.
+  * Distress is about avoiding early-oom, this is about
+  * making swappiness graceful despite setting it to low
+  * values.
+  *
+  * Avoid div by zero with nr_inactive+1, and max resulting
+  * value is vm_total_pages.
+  */
+ imbalance  = zone_page_state(zone, NR_ACTIVE);
+ imbalance /= zone_page_state(zone, NR_INACTIVE) + 1;
+
+ /*
+  * Reduce the effect of imbalance if swappiness is low,
+  * this means for a swappiness very low, the imbalance
+  * must be much higher than 100 for this logic to make
+  * the difference.
+  *
+  * Max temporary value is vm_total_pages*100.
+  */
+ imbalance *= (vm_swappiness + 1);
+ imbalance /= 100;
+
+ /*
+  * If not much of the ram is mapped, makes the imbalance
+  * less relevant, it's high priority we refill the inactive
+  * list with mapped pages only in presence of high ratio of
+  * mapped pages.
+  *
+  * Max temporary value is vm_total_pages*100.
+  */
+ imbalance *= mapped_ratio;
+ imbalance /= 100;
+
+ /* apply imbalance feedback to swap_tendency */
+ swap_tendency += imbalance;
+
+ /*
+  * Now use this metric to decide whether to start moving mapped
+  * memory onto the inactive list.
+  */
+ if (swap_tendency >= 100)
+  reclaim_mapped = 1;
+
+ return reclaim_mapped;
+}
+
+/*
  * This moves pages from the active list to the inactive list.
  *
  * We move them the other way if the page is referenced by one or more
@@ -966,6 +1058,8 @@ static inline int zone_is_near_oom(struc
  * The downside is that we have to touch page->_count against each page.
  * But we had to alter page->flags anyway.
  */
+
+
 static void shrink_active_list(unsigned long nr_pages, struct zone *zone,
     struct scan_control *sc, int priority)
 {
@@ -979,93 +1073,9 @@ static void shrink_active_list(unsigned
  struct pagevec pvec;
  int reclaim_mapped = 0;

- if (sc->may_swap) {
-  long mapped_ratio;
-  long distress;
-  long swap_tendency;
-  long imbalance;
-
-  if (zone_is_near_oom(zone))
-   goto force_reclaim_mapped;
-
-  /*
-   * `distress' is a measure of how much trouble we're having
-   * reclaiming pages.  0 -> no problems.  100 -> great trouble.
-   */
-  distress = 100 >> min(zone->prev_priority, priority);
-
-  /*
-   * The point of this algorithm is to decide when to start
-   * reclaiming mapped memory instead of just pagecache.  Work out
-   * how much memory
-   * is mapped.
-   */
-  mapped_ratio = ((global_page_state(NR_FILE_MAPPED) +
-    global_page_state(NR_ANON_PAGES)) * 100) /
-     vm_total_pages;
-
-  /*
-   * Now decide how much we really want to unmap some pages.  The
-   * mapped ratio is downgraded - just because there's a lot of
-   * mapped memory doesn't necessarily mean that page reclaim
-   * isn't succeeding.
-   *
-   * The distress ratio is important - we don't want to start
-   * going oom.
-   *
-   * A 100% value of vm_swappiness overrides this algorithm
-   * altogether.
-   */
-  swap_tendency = mapped_ratio / 2 + distress + sc->swappiness;
-
-  /*
-   * If there's huge imbalance between active and inactive
-   * (think active 100 times larger than inactive) we should
-   * become more permissive, or the system will take too much
-   * cpu before it start swapping during memory pressure.
-   * Distress is about avoiding early-oom, this is about
-   * making swappiness graceful despite setting it to low
-   * values.
-   *
-   * Avoid div by zero with nr_inactive+1, and max resulting
-   * value is vm_total_pages.
-   */
-  imbalance  = zone_page_state(zone, NR_ACTIVE);
-  imbalance /= zone_page_state(zone, NR_INACTIVE) + 1;
-
-  /*
-   * Reduce the effect of imbalance if swappiness is low,
-   * this means for a swappiness very low, the imbalance
-   * must be much higher than 100 for this logic to make
-   * the difference.
-   *
-   * Max temporary value is vm_total_pages*100.
-   */
-  imbalance *= (vm_swappiness + 1);
-  imbalance /= 100;
```

```
-
-  /*
-   * If not much of the ram is mapped, makes the imbalance
-   * less relevant, it's high priority we refill the inactive
-   * list with mapped pages only in presence of high ratio of
-   * mapped pages.
-   *
-   * Max temporary value is vm_total_pages*100.
-   */
-  imbalance *= mapped_ratio;
-  imbalance /= 100;
-
-  /* apply imbalance feedback to swap_tendency */
-  swap_tendency += imbalance;
-
-  /*
-   * Now use this metric to decide whether to start moving mapped
-   * memory onto the inactive list.
-   */
-  if (swap_tendency >= 100)
-force_reclaim_mapped:
-   reclaim_mapped = 1;
- }
+ if (sc->may_swap)
+  reclaim_mapped = calc_reclaim_mapped(zone, priority,
+           sc->swappiness);

  lru_add_drain();
  spin_lock_irq(&zone->lru_lock);
```