Subject: [PATCH 2/2] mm: refault histogram
Posted by Peter Zijlstra on Thu, 26 Jul 2007 17:25:06 GMT
View Forum Message <> Reply to Message

Adds a refault histogram for those policies that use nonresident page tracking.
Based on ideas and code from Rik van Riel.

Signed-off-by: Peter Zijlstra <a.p.zijlstra@chello.nl>
---
include/linux/nonresident.h | 12 +++
mm/Kconfig | 5 +
mm/Makefile | 1
mm/nonresident.c | 2
mm/refault.c | 143 ++
5 files changed, 163 insertions(+)

Index: linux-2.6/mm/Kconfig
===
--- linux-2.6.orig/mm/Kconfig
+++ linux-2.6/mm/Kconfig
@@ -162,6 +162,11 @@ config MM_NONRESIDENT
bool "Track nonresident pages"
def_bool y

+config MM_REFAULT
+ bool "Refault histogram"
+ def_bool y
+ depends on MM_NONRESIDENT
+
config ZONE_DMA_FLAG
int
default "0" if !ZONE_DMA
Index: linux-2.6/mm/nonresident.c
===
--- linux-2.6.orig/mm/nonresident.c
+++ linux-2.6/mm/nonresident.c
@@ -24,6 +24,7 @@
#include <linux/hash.h>
#include <linux/prefetch.h>
#include <linux/kernel.h>
+#include <linux/nonresident.h>

/* Number of non-resident pages per hash bucket. Never smaller than 15. */
#if (L1_CACHE_BYTES < 64)
@@ -99,6 +100,7 @@ nonresident_get(struct address_space *ma
}

out:

```
+ nonresident_refault(distance);
  return distance;
 }

Index: linux-2.6/mm/refault.c
===================================================================
--- /dev/null
+++ linux-2.6/mm/refault.c
@@ -0,0 +1,143 @@
+#include <linux/module.h>
+#include <linux/percpu.h>
+#include <linux/seq_file.h>
+#include <linux/proc_fs.h>
+#include <linux/nonresident.h>
+#include <linux/uaccess.h>
+
+#define BUCKETS 16
+
+DEFINE_PER_CPU(unsigned long[BUCKETS+1], refault_histogram);
+
+void nonresident_refault(unsigned long distance)
+{
+ unsigned long nonres_bucket = nonresident_total() / BUCKETS;
+ unsigned long bucket_id = distance / nonres_bucket;
+
+ if (bucket_id > BUCKETS)
+  bucket_id = BUCKETS;
+
+ __get_cpu_var(refault_histogram)[bucket_id]++;
+}
+
+#ifdef CONFIG_PROC_FS
+
+#include <linux/seq_file.h>
+
+static void *frag_start(struct seq_file *m, loff_t *pos)
+{
+ if (*pos < 0 || *pos > BUCKETS)
+  return NULL;
+
+ m->private = (void *)(unsigned long)*pos;
+
+ return pos;
+}
+
+static void *frag_next(struct seq_file *m, void *arg, loff_t *pos)
+{
+ if (*pos < BUCKETS) {
```

```
+  (*pos)++;
+  (*((unsigned long *)&m->private))++;
+  return pos;
+ }
+ return NULL;
+}
+
+static void frag_stop(struct seq_file *m, void *arg)
+{
+}
+
+unsigned long get_refault_stat(unsigned long index)
+{
+ unsigned long total = 0;
+ int cpu;
+
+ for_each_possible_cpu(cpu)
+  total += per_cpu(refault_histogram, cpu)[index];
+
+ return total;
+}
+
+#define K(pages) (pages << (PAGE_SHIFT - 10))
+
+static int frag_show(struct seq_file *m, void *arg)
+{
+ unsigned long index = (unsigned long)m->private;
+ unsigned long nonres_bucket = nonresident_total() / BUCKETS;
+ unsigned long upper = ((unsigned long)index + 1) * nonres_bucket;
+ unsigned long lower = (unsigned long)index * nonres_bucket;
+ unsigned long hits = get_refault_stat(index);
+
+ if (index == 0)
+  seq_printf(m, "     Refault distance          Hits\n");
+
+ if (index < BUCKETS)
+  seq_printf(m, "%9luK - %9luK     %9lu\n", K(lower), K(upper), hits);
+ else
+  seq_printf(m, " New/Beyond %9luK     %9lu\n", K(lower), hits);
+
+ return 0;
+}
+
+struct seq_operations refault_op = {
+ .start  = frag_start,
+ .next   = frag_next,
+ .stop   = frag_stop,
+ .show   = frag_show,
```

```
+};
+
+static void refault_reset(void)
+{
+ int cpu;
+ int bucket_id;
+
+ for_each_possible_cpu(cpu) {
+  for (bucket_id = 0; bucket_id <= BUCKETS; ++bucket_id)
+   per_cpu(refault_histogram, cpu)[bucket_id] = 0;
+ }
+}
+
+ssize_t refault_write(struct file *file, const char __user *buf,
+        size_t count, loff_t *ppos)
+{
+ if (count) {
+  char c;
+
+  if (get_user(c, buf))
+   return -EFAULT;
+  if (c == '0')
+   refault_reset();
+ }
+ return count;
+}
+
+static int refault_open(struct inode *inode, struct file *file)
+{
+       (void)inode;
+       return seq_open(file, &refault_op);
+}
+
+static struct file_operations refault_file_operations = {
+       .open           = refault_open,
+       .read           = seq_read,
+       .llseek         = seq_lseek,
+       .release        = seq_release,
+       .write          = refault_write,
+};
+
+static int __init refault_proc_init(void)
+{
+ struct proc_dir_entry *entry;
+
+ entry = create_proc_entry("refault", S_IRUSR, NULL);
+ if (entry)
+  entry->proc_fops = &refault_file_operations;
```

```
+
+ return 0;
+}
+
+__initcall(refault_proc_init);
+
+#endif /* CONFIG_PROCFS */
+
Index: linux-2.6/mm/Makefile
===================================================================
--- linux-2.6.orig/mm/Makefile
+++ linux-2.6/mm/Makefile
@@ -16,6 +16,7 @@ obj-y   := bootmem.o filemap.o mempool.o
 obj-$(CONFIG_BOUNCE) += bounce.o
 obj-$(CONFIG_SWAP) += page_io.o swap_state.o swapfile.o thrash.o
 obj-$(CONFIG_MM_NONRESIDENT) += nonresident.o
+obj-$(CONFIG_MM_REFAULT) += refault.o
 obj-$(CONFIG_HUGETLBFS) += hugetlb.o
 obj-$(CONFIG_NUMA)  += mempolicy.o
 obj-$(CONFIG_SPARSEMEM) += sparse.o
Index: linux-2.6/include/linux/nonresident.h
===================================================================
--- linux-2.6.orig/include/linux/nonresident.h
+++ linux-2.6/include/linux/nonresident.h
@@ -31,5 +31,17 @@ static inline unsigned long nonresident_

 #endif /* CONFIG_MM_NONRESIDENT */

+#ifdef CONFIG_MM_REFAULT
+
+extern void nonresident_refault(unsigned long);
+
+#else
+
+static inline void nonresident_refault(unsigned long distance)
+{
+}
+
+#endif
+
 #endif /* __KERNEL */
 #endif /* _LINUX_NONRESIDENT_H_ */

--

_____
Containers mailing list
Containers@lists.linux-foundation.org
```

https://lists.linux-foundation.org/mailman/listinfo/containers