A Concurrent Real-Time White Paper

2881 Gateway Drive
Pompano Beach, FL 33069
(954) 974-1700
www. real-time.ccur.com

An Overview of Kernel Text Page Replication in RedHawk™ Linux® 6.3

By: John Blackwood
Concurrent Real-Time Linux Development Team

September 2012

Abstract

This paper describes the RedHawk Linux kernel feature that replicates kernel text and read-only data pages on Non-Uniform Memory Access (NUMA) nodes on RedHawk v6.3 x86_64 systems. Replication of kernel text and read-only data helps reduce system inter-node memory traffic by keeping kernel instruction and read-only data accesses local to each NUMA node. This RedHawk feature applies to both statically-built and dynamically-loaded kernel modules. Several tests were run to determine the performance improvement of replicated versus non-replicated execution.

Overview

On NUMA computer systems, memory access time depends upon the memory location relative to a processor. NUMA multiprocessor systems are made up of two or more interconnected NUMA nodes, where each node contains one or more processor cores and local memory. In NUMA systems, all of main memory is accessible to all processors, however memory accesses can be either local or remote – local access to memory that resides on the same node as the processor and remote access to memory that resides in a different node. Remote accesses can be more expensive in terms of latency and increased bus contention, especially when the remote memory resides more than one NUMA node bus interconnection away from the accessing processor.

At system boot time, the RedHawk Linux kernel will create copies of the resident kernel text and read-only data pages in each NUMA node. It will also set up the additional data structures needed for supporting per-node kernel virtual address translations to the pages. In addition, when a kernel module is dynamically loaded, the load processing will create copies of its text and read-only data pages in each NUMA node and set up the appropriate per-node translation table entries to the replicated pages.

At context switch time, when a task (process) is switched onto a CPU and that task had previously executed on a CPU that resides in a different NUMA node, the kernel will update the task's virtual address space translation tables so that memory accesses to kernel text and read-only data are local to the new node.

Performance Measurements

Some system service execution-time measurements were taken in order to view the amount of improvement gained by having kernel text locally replicated in all NUMA nodes. The system calls were first executed without kernel text replication enabled, and then executed again with replication enabled:

The tests were executed on a CPU located on a NUMA node other than the first node in the system, with the entire test node shielded of system activity and interrupts. All CPUs located on the other non-shielded nodes executed a user-space read/write memory test as a means of placing some background memory-bus activity on the system.

Each test was executed with *hot* caches, where no data cache manipulation was done, and then with *cold* caches, where the test processor's data caches were flushed upon entry to the system service call being measured. Cold caches result in more kernel text cache line fills from memory.

The tests that were run include:

- *fork*

A loop of *fork*(2) system service calls was made, and the time between entering and exiting this system service call was measured.

- *kill / signal*

A signal handler for catching *SIGUSR1* was setup, and then a loop of *SIGUSR1* signals were sent to the currently executing process using *kill*(2). The time between entering the *kill*(2) system service call and exiting from the signal handler was measured in the kernel.

- *mmap / munmap*

A set of 1,024 pages of the */dev/zero* device file was mapped and unmapped in a loop. The time between entering the *mmap*(2) system service call and exiting the *munmap*(2) system service call was measured.

- *open / close*

The */dev/null* device file was opened and closed in a loop. The time between entering the *open*(2) system service call and exiting the *close*(2) system service call was measured.

These tests were run on two types of NUMA systems:

- A four-node Intel® Xeon® E5-4610 system with 6 processor cores on each node
- An eight-node AMD Opteron™ 6272 system with 8 processor cores on each node

The results of these tests indicate that with kernel text replication it is possible to see a performance improvement as high as 41% in some circumstances, although a more typical improvement of 2-7% can be expected in many cases. The test results are shown in the tables below:

Four-node 24-core Intel System

Test	Cache	No Replication (in usecs.)	Replication (in usecs.)	Improvement
fork	hot	23.686	23.321	1.5 %
	cold	156.510	91.647	41.4 %
kill	hot	3.618	3.617	0.0 %
	cold	569.461	528.493	7.1 %
mmap	hot	2847.247	2484.434	12.7 %
	cold	19097.210	17640.744	7.6 %
open	hot	4.027	4.007	0.4 %
	cold	548.105	536.488	2.1 %

Eight-node 64-core AMD System

Test	Cache	No Replication (in usecs.)	Replication (in usecs.)	Improvement
fork	hot	44.362	43.510	1.9 %
	cold	210.517	133.549	36.5 %
kill	hot	7.377	7.218	2.1 %
	cold	1311.115	1265.412	3.4 %
mmap	hot	4463.533	4458.603	0.1 %
	cold	43008.612	41119.776	4.3 %
open	hot	15.216	14.925	1.9 %
	cold	1367.260	1307.358	4.3 %

Configuration

Kernel text and read-only data page replication is automatically enabled in pre-built RedHawk Linux 6.3 x86_64 kernels. Thus, on NUMA systems with two or more NUMA nodes, the kernel will automatically setup and enable kernel text replication support during system boot.

Some or all of the kernel text replication support may be disabled at kernel build time via the use of two kernel configuration parameters. Users may disable just the kernel module page replication support and leave replication of kernel resident text and read-only data enabled, or alternatively, they may disable all kernel text replication. Two corresponding grub options are also provided for disabling kernel text replication support at boot time when booting a kernel that was compiled with kernel text replication enabled.

Kernel Text Modification

RedHawk Linux kernel text replication also supports the following features that modify kernel text by ensuring that all copies of a text page are updated appropriately:

- Kprobes
- Dynamic ftracing
- Kernel debugger breakpoints

Viewing Systems Information

A RedHawk Linux system utility, *ktrinfo*(1), can be used to display the kernel text replication information and statistics. When invoked with no options, *ktrinfo*(1) will display the virtual-to-physical translations of the resident kernel text and read-only data areas, as well as the per-node statistics. For example, on a two-node NUMA system:

```
> ktrinfo

Node 0 Text Translations

virtual_address                physical_address              size

ffffffff81000000-ffffffff811fffff  0000000001000000-00000000011fffff 2048 kB

ffffffff81200000-ffffffff813fffff  0000000001200000-00000000013fffff 2048 kB

ffffffff81400000-ffffffff81479fff  0000000001400000-0000000001479fff  488 kB

Node 0 Read-only Data Translations

virtual_address                physical_address              size

ffffffff81600000-ffffffff817fffff  0000000001600000-00000000017fffff 2048 kB

ffffffff81800000-ffffffff8187efff  0000000001800000-000000000187efff  508 kB
```

```
Node 0 statistics
total_repli_pages          892 kB
repli_resident_pages         0 kB
repli_module_pages         880 kB
repli_module_pgtbls         12 kB
page_alloc_failures          0
pagetbl_alloc_failures       0

Node 1 Text Translations
virtual_address                physical_address              size
ffffffff81000000-ffffffff811fffff  0000000236e00000-0000000236ffffff 2048 kB
ffffffff81200000-ffffffff813fffff  0000000236400000-00000002365fffff 2048 kB
ffffffff81400000-ffffffff81479fff  0000000236600000-0000000236679fff  488 kB
Node 1 Read-only Data Translations
virtual_address                physical_address              size
ffffffff81600000-ffffffff817fffff  0000000236000000-00000002361fffff 2048 kB
ffffffff81800000-ffffffff8187efff  0000000236200000-000000023627efff  508 kB
Node 1 statistics
total_repli_pages         9204 kB
repli_resident_pages      7140 kB
repli_module_pages        2052 kB
repli_module_pgtbls         12 kB
page_alloc_failures          0
pagetbl_alloc_failures       0
```

ktrinfo(1) will also display the size (in Kbytes) of the replicated text and read-only data areas located in each node on a per-module basis when the *-m* option is used. A NUMA node with dash ("-") for a given module size indicates that the module pages were originally loaded in that node before being replicated into the other nodes in the system:

```
> ktrinfo -m
```

```
Module           Text_RO_sz    Node0    Node1
ip6table_filter        8192       -     8192
ip6_tables            16384       -    16384
ebtable_nat            8192       -     8192
ebtables              20480    20480       -
ipt_MASQUERADE         8192       -     8192
iptable_nat            8192       -     8192
nf_nat                12288       -    12288
nf_conntrack_ipv4      12288       -    12288
nf_defrag_ipv4         8192       -     8192
xt_state               8192       -     8192
nf_conntrack          45056       -    45056
ipt_REJECT             8192       -     8192
xt_CHECKSUM            8192       -     8192
iptable_mangle         8192       -     8192
iptable_filter         8192       -     8192
ip_tables             16384       -    16384
tun                   12288       -    12288
bridge                53248       -    53248
rcim_emu               8192       -     8192
autofs4               24576       -    24576
sunrpc               139264       -   139264
target_core_iblock     8192       -     8192
target_core_file       8192    8192       -
target_core_pscsi      12288       -    12288
target_core_mod       176128   176128       -
configfs              16384       -    16384
...
```

Conclusions

Testing has shown that reducing inter-node memory accesses through the use of locally replicated kernel text can be beneficial in increasing performance and improving determinism in both large and small NUMA-based systems. RedHawk Linux kernel text replication automatically replicates kernel text and read-only data without any additional system administration or user application programming requirements.

As the trend toward larger NUMA systems with more processors and memory continues, RedHawk Linux kernel text replication, along with RedHawk memory shielding with user page cache per-node page replication, provide powerful facilities for significantly reducing inter-node memory traffic and memory-bus contention - without placing any significant requirements on the application writer.

About Concurrent

Concurrent (NASDAQ:CCUR) is a global leader in multi-screen video delivery, media data management and real-time computing solutions. Concurrent Real-Time is the industries' foremost provider of high-performance real-time Linux computer systems, solutions and software for commercial and government markets. The company focuses on hardware-in-the-loop and man-in-the-loop simulation, data acquisition, and industrial systems, and serves industries that include aerospace and defense, automotive, energy and financial. Concurrent is headquartered in Atlanta with offices in North America, Europe and Asia. Concurrent Real-Time is located in Pompano Beach, Florida. For more information, please visit Concurrent Real-Time at www.real-time.ccur.com.