

[PATCH] Isolated sched domains for 2.6.8-rc2-mm1

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-07/6223.html>

From: Dimitri Sivanich (sivanich_at_sgi.com)

Date: 07/30/04

Date: Fri, 30 Jul 2004 12:46:51 -0500

To: Ingo Molnar <mingo@elte.hu>, Nick Piggin <nickpiggin@yahoo.com.au>, Jesse Barnes <jbarnes@eng

Hi all,

Here's a version of the isolated scheduler domain code that I mentioned in an RFC on 7/22. This patch applies on top of 2.6.8-rc2-mm1 (to include all of the new arch_init_sched_domain code). This patch also contains the 2 line fix to remove the check of first_cpu(sd->groups->cpumask)) that Jesse sent in earlier.

Note that this has not been tested with CONFIG_SCHED_SMT. I hope that my handling of those instances is OK.

Signed-off-by: Dimitri Sivanich <sivanich@sgi.com>

Index: linux/kernel/sched.c

```
=====
--- linux.orig/kernel/sched.c 2004-07-30 09:38:57.000000000 -0500
+++ linux/kernel/sched.c 2004-07-30 12:03:24.000000000 -0500
@@ -3714,6 +3714,31 @@ __init static int cpu_to_node_group(int
 }
 #endif

+/* Groups for isolated scheduling domains */
+static struct sched_group sched_group_isolated[NR_CPUS];
+__init static int cpu_to_isolated_group(int cpu)
+{
+ return cpu;
+}
+
+cpumask_t __initdata cpu_isolated_map = CPU_MASK_NONE; /* cpus with isolated domains */
+
+/* Setup the mask of cpus configured for isolated domains */
+static int __init
+isolated_cpu_setup(char *str)
+{
+ int ints[NR_CPUS], i;
```

```

+
+ str = get_options(str, ARRAY_SIZE(ints), ints);
+ cpus_clear(cpu_isolated_map);
+ for (i=1; i<=ints[0]; i++) {
+ cpu_set(ints[i], cpu_isolated_map);
+ }
+ return 1;
+}
+
+
+__setup ("isolcpus=", isolated_cpu_setup);
+
+/*
+ * init_sched_build_groups takes an array of groups, the cpumask we wish
+ * to span, and a pointer to a function which identifies what group a CPU
+ @@ -3762,21 +3787,52 @@ __init static void init_sched_build_grou
+ __init static void arch_init_sched_domains(void)
+ {
+     int i;
+ cpumask_t cpu_default_map;
+
+ /*
+ * Setup mask for cpus without special case scheduling requirements.
+ * For now this just excludes isolated cpus, but could be used to
+ * exclude other special cases in the future.
+ */
+ cpus_complement(cpu_default_map, cpu_isolated_map);
+ cpus_and(cpu_default_map, cpu_default_map, cpu_possible_map);
+
+     /* Set up domains */
+     for_each_cpu(i) {
+         int group;
+         struct sched_domain *sd = NULL, *p;
+         cpumask_t nodemask = node_to_cpumask(cpu_to_node(i));
+ cpus_and(nodemask, nodemask, cpu_default_map);
+
+ #ifndef CONFIG_NUMA
+ - if (i != first_cpu(sd->groups->cpumask))
+ /*
+ * Set up isolated domains.
+ * Unlike those of other cpus, the domains and groups are
+ * single level, and span a single cpu.
+ */
+ if (cpu_isset(i, cpu_isolated_map)) {
+ #ifdef CONFIG_SCHED_SMT
+ + sd = &per_cpu(cpu_domains, i);
+ #else
+ + sd = &per_cpu(phys_domains, i);
+ #endif
+ + group = cpu_to_isolated_group(i);
+ + *sd = SD_CPU_INIT;
+ + cpu_set(i, sd->span);

```

Linux-Kernel: [PATCH] Isolated sched domains for 2.6.8-rc2-mm1

```

+ sd->balance_interval = INT_MAX; /* Don't balance */
+ sd->flags = 0; /* Avoid WAKE_ */
+ sd->groups = &sched_group_isolated[group];
+ printk(KERN_INFO "Setting up cpu %d isolated.\n", i);
+ /* Single level, so continue with next cpu */
+       continue;
+ }
+
+#ifdef CONFIG_NUMA
+       sd = &per_cpu(node_domains, i);
+       group = cpu_to_node_group(i);
+       *sd = SD_NODE_INIT;
+       /* FIXME: should be multilevel, in arch code */
+       sd->span = sched_domain_node_span(i, SD_NODES_PER_DOMAIN);
+ cpus_and(sd->span, sd->span, cpu_default_map);
+       sd->groups = &sched_group_nodes[group];
+endif

@@ -3794,6 +3850,7 @@ __init static void arch_init_sched_domai
+       group = cpu_to_cpu_group(i);
+       *sd = SD_SIBLING_INIT;
+       sd->span = cpu_sibling_map[i];
+ cpus_and(sd->span, sd->span, cpu_default_map);
+       sd->parent = p;
+       sd->groups = &sched_group_cpus[group];
+endif

@@ -3802,19 +3859,30 @@ __init static void arch_init_sched_domai
+#ifdef CONFIG_SCHED_SMT
+       /* Set up CPU (sibling) groups */
+       for_each_cpu(i) {
- if (i != first_cpu(cpu_sibling_map[i]))
+ cpumask_t this_sibling_map = cpu_sibling_map[i];
+ cpus_and(this_sibling_map, this_sibling_map, cpu_default_map);
+ if (i != first_cpu(this_sibling_map))
+       continue;

- init_sched_build_groups(sched_group_cpus, cpu_sibling_map[i],
+ init_sched_build_groups(sched_group_cpus, this_sibling_map,
+                           &cpu_to_cpu_group);
+       }
+endif

+ /* Set up isolated groups */
+ for_each_cpu_mask(i, cpu_isolated_map) {
+ cpumask_t mask;
+ cpus_clear(mask);
+ cpu_set(i, mask);
+ init_sched_build_groups(sched_group_isolated, mask,
+ &cpu_to_isolated_group);
+ }
+

```

Linux-Kernel: [PATCH] Isolated sched domains for 2.6.8-rc2-mm1

```
/* Set up physical groups */
for (i = 0; i < MAX_NUMNODES; i++) {
    cpumask_t nodemask = node_to_cpumask(i);

- cpus_and(nodemask, nodemask, cpu_possible_map);
+ cpus_and(nodemask, nodemask, cpu_default_map);
    if (cpus_empty(nodemask))
        continue;

@@ -3824,12 +3892,12 @@ __init static void arch_init_sched_domai

#ifdef CONFIG_NUMA
    /* Set up node groups */
- init_sched_build_groups(sched_group_nodes, cpu_possible_map,
+ init_sched_build_groups(sched_group_nodes, cpu_default_map,
                          &cpu_to_node_group);
#endif

    /* Calculate CPU power for physical packages and nodes */
- for_each_cpu(i) {
+ for_each_cpu_mask(i, cpu_default_map) {
    int power;
    struct sched_domain *sd;
#ifdef CONFIG_SCHED_SMT
-

```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>