

[PATCH] new bitmap list format (for cpusets)

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-08/1133.html>

From: Paul Jackson (*pj_at_sgi.com*)

Date: 08/05/04

Date: Thu, 5 Aug 2004 03:08:47 -0700 (PDT)

To: Andrew Morton <akpm@osdl.org>

A bitmap print and parse format that provides lists of ranges of numbers, to be first used for by cpusets (next patch).

Cpusets provide a way to manage subsets of CPUs and Memory Nodes for scheduling and memory placement, via a new virtual file system, usually mounted at /dev/cpuset. Manipulation of cpusets can be done directly via this file system, from the shell.

However, manipulating 512 bit cpumasks or 256 bit nodemasks (which will get bigger) via hex mask strings is painful for humans.

The intention is to provide a format for the cpu and memory mask files in /dev/cpusets that will stand the test of time. This format is supported by a couple of new lib/bitmap.c routines, for printing and parsing these strings. Wrappers for cpumask and nodemask are provided.

See the embedded comments, below in the patch, for more details of the format. The input format supports adding or removing specified cpus or nodes, as well as entirely rewriting the mask.

```
include/linux/bitmap.h | 8 ++
include/linux/cpumask.h | 22 ++++++
include/linux/nodemask.h | 22 ++++++
lib/bitmap.c | 142 ++++++
4 files changed, 189 insertions(+), 5 deletions(-)
```

Signed-off-by: Paul Jackson <pj@sgi.com>

Index: 2.6.8-rc2-mm2/include/linux/bitmap.h

```
=====
--- 2.6.8-rc2-mm2.orig/include/linux/bitmap.h 2004-08-04 19:29:15.000000000 -0700
+++ 2.6.8-rc2-mm2/include/linux/bitmap.h 2004-08-04 19:41:10.000000000 -0700
@@ -41,7 +41,9 @@
 * bitmap_shift_right(dst, src, n, nbits) *dst = *src >> n
 * bitmap_shift_left(dst, src, n, nbits) *dst = *src << n
 * bitmap_scnprintf(buf, len, src, nbits) Print bitmap src to buf
- * bitmap_parse(ubuf, ulen, dst, nbits) Parse bitmap dst from buf
```

Linux-Kernel: [PATCH] new bitmap list format (for cpusets)

```
+ * bitmap_parse(ubuf, ulen, dst, nbits) Parse bitmap dst from user buf
+ * bitmap_scnlistprintf(buf, len, src, nbits) Print bitmap src as list to buf
+ * bitmap_parselist(buf, dst, nbits) Parse bitmap dst from list
*/

/*
@@ -98,6 +100,10 @@ extern int bitmap_scnprintf(char *buf, u
        const unsigned long *src, int nbits);
extern int bitmap_parse(const char __user *ubuf, unsigned int ulen,
        unsigned long *dst, int nbits);
+extern int bitmap_scnlistprintf(char *buf, unsigned int len,
+ const unsigned long *src, int nbits);
+extern int bitmap_parselist(const char *buf, unsigned long *maskp,
+ int nmaskbits);
extern int bitmap_find_free_region(unsigned long *bitmap, int bits, int order);
extern void bitmap_release_region(unsigned long *bitmap, int pos, int order);
extern int bitmap_allocate_region(unsigned long *bitmap, int pos, int order);
Index: 2.6.8-rc2-mm2/include/linux/cpumask.h
=====
--- 2.6.8-rc2-mm2.orig/include/linux/cpumask.h 2004-08-04 19:29:34.000000000 -0700
+++ 2.6.8-rc2-mm2/include/linux/cpumask.h 2004-08-04 20:35:10.000000000 -0700
@@ -10,6 +10,8 @@
*
* For details of cpumask_scnprintf() and cpumask_parse(),
* see bitmap_scnprintf() and bitmap_parse() in lib/bitmap.c.
+ * For details of cpulist_scnprintf() and cpulist_parse(), see
+ * bitmap_scnlistprintf() and bitmap_parselist(), also in bitmap.c.
*
* The available cpumask operations are:
*
@@ -46,6 +48,8 @@
*
* int cpumask_scnprintf(buf, len, mask) Format cpumask for printing
* int cpumask_parse(ubuf, ulen, mask) Parse ascii string as cpumask
+ * int cpulist_scnprintf(buf, len, mask) Format cpumask as list for printing
+ * int cpulist_parse(buf, map) Parse ascii string as cpulist
*
* for_each_cpu_mask(cpu, mask) for-loop cpu over mask
*
@@ -268,14 +272,28 @@ static inline int __cpumask_scnprintf(ch
        return bitmap_scnprintf(buf, len, srcp->bits, nbits);
}

-#define cpumask_parse(ubuf, ulen, src) \
- __cpumask_parse((ubuf), (ulen), &(src), NR_CPUS)
+#define cpumask_parse(ubuf, ulen, dst) \
+ __cpumask_parse((ubuf), (ulen), &(dst), NR_CPUS)
static inline int __cpumask_parse(const char __user *buf, int len,
        cpumask_t *dstp, int nbits)
{
        return bitmap_parse(buf, len, dstp->bits, nbits);
}
```

Linux–Kernel: [PATCH] new bitmap list format (for cpusets)

```

}

+#define cpulist_scnprintf(buf, len, src) \
+ __cpulist_scnprintf((buf), (len), &(src), NR_CPUS)
+static inline int __cpulist_scnprintf(char *buf, int len,
+ const cpumask_t *srcp, int nbits)
+{
+ return bitmap_scnlistprintf(buf, len, srcp->bits, nbits);
+}
+
+#define cpulist_parse(buf, dst) __cpulist_parse((buf), &(dst), NR_CPUS)
+static inline int __cpulist_parse(const char *buf, cpumask_t *dstp, int nbits)
+{
+ return bitmap_parselist(buf, dstp->bits, nbits);
+}
+
+#if NR_CPUS > 1
+#define for_each_cpu_mask(cpu, mask) \
+ for ((cpu) = first_cpu(mask); \
Index: 2.6.8-rc2-mm2/include/linux/nodemask.h
=====
--- 2.6.8-rc2-mm2.orig/include/linux/nodemask.h 2004-08-04 19:29:29.000000000 -0700
+++ 2.6.8-rc2-mm2/include/linux/nodemask.h 2004-08-04 20:28:50.000000000 -0700
@@ -10,6 +10,8 @@
*
* For details of nodemask_scnprintf() and nodemask_parse(),
* see bitmap_scnprintf() and bitmap_parse() in lib/bitmap.c.
+ * For details of nodelist_scnprintf() and nodelist_parse(), see
+ * bitmap_scnlistprintf() and bitmap_parselist(), also in bitmap.c.
*
* The available nodemask operations are:
*
@@ -46,6 +48,8 @@
*
* int nodemask_scnprintf(buf, len, mask) Format nodemask for printing
* int nodemask_parse(ubuf, ulen, mask) Parse ascii string as nodemask
+ * int nodelist_scnprintf(buf, len, mask) Format nodemask as list for printing
+ * int nodelist_parse(buf, map) Parse ascii string as nodelist
*
* for_each_node_mask(node, mask) for-loop node over mask
*
@@ -271,14 +275,28 @@ static inline int __nodemask_scnprintf(c
+ return bitmap_scnprintf(buf, len, srcp->bits, nbits);
}

-#define nodemask_parse(ubuf, ulen, src) \
- __nodemask_parse((ubuf), (ulen), &(src), MAX_NUMNODES)
+#define nodemask_parse(ubuf, ulen, dst) \
+ __nodemask_parse((ubuf), (ulen), &(dst), MAX_NUMNODES)
+static inline int __nodemask_parse(const char __user *buf, int len,
+ nodemask_t *dstp, int nbits)

```

Linux-Kernel: [PATCH] new bitmap list format (for cpusets)

```

{
    return bitmap_parse(buf, len, dstp->bits, nbits);
}

#define nodelist_scnprintf(buf, len, src) \
+ __nodelist_scnprintf((buf), (len), &(src), MAX_NUMNODES)
+static inline int __nodelist_scnprintf(char *buf, int len,
+ const nodemask_t *srcp, int nbits)
+{
+ return bitmap_scnlistprintf(buf, len, srcp->bits, nbits);
+}
+
+#define nodelist_parse(buf, dst) __nodelist_parse((buf), &(dst), MAX_NUMNODES)
+static inline int __nodelist_parse(const char *buf, nodemask_t *dstp, int nbits)
+{
+ return bitmap_parselist(buf, dstp->bits, nbits);
+}
+
+#if MAX_NUMNODES > 1
+#define for_each_node_mask(node, mask) \
    for ((node) = first_node(mask); \
Index: 2.6.8-rc2-mm2/lib/bitmap.c
=====
--- 2.6.8-rc2-mm2.orig/lib/bitmap.c 2004-08-04 19:29:15.000000000 -0700
+++ 2.6.8-rc2-mm2/lib/bitmap.c 2004-08-04 21:44:41.000000000 -0700
@@ -291,6 +291,7 @@ EXPORT_SYMBOL(__bitmap_weight);
#define nbits_to_hold_value(val) fls(val)
#define roundup_power2(val, modulus) (((val) + (modulus) - 1) & ~((modulus) - 1))
#define unhex(c) (isdigit(c) ? (c - '0') : (toupper(c) - 'A' + 10))
+#define BASEDEC 10 /* fancier cpuset lists input in decimal */

/**
 * bitmap_scnprintf - convert bitmap to an ASCII hex string.
@@ -409,6 +410,147 @@ int bitmap_parse(const char __user *ubuf
}
EXPORT_SYMBOL(bitmap_parse);

+/*
+ * bscnl_emit(buf, buflen, rbot, rtop, bp)
+ *
+ * Helper routine for bitmap_scnlistprintf(). Write decimal number
+ * or range to buf, suppressing output past buf+buflen, with optional
+ * comma-prefix. Return len of what would be written to buf, if it
+ * all fit.
+ */
+
+int bscnl_emit(char *buf, int buflen, int rbot, int rtop, int len)
+{
+ if (len)
+ len += scnprintf(buf + len, buflen - len, ",");
+ if (rbot == rtop)

```

Linux–Kernel: [PATCH] new bitmap list format (for cpusets)

```
+ len += scnprintf(buf + len, buflen - len, "%d", rbot);
+ else
+ len += scnprintf(buf + len, buflen - len, "%d-%d", rbot, rtop);
+ return len;
+ }
+
+ /**
+ * bitmap_scnlistprintf – convert bitmap to an ASCII hex string, list format
+ * @buf: byte buffer into which string is placed
+ * @buflen: reserved size of @buf, in bytes
+ * @maskp: pointer to bitmap to convert
+ * @nmaskbits: size of bitmap, in bits
+ *
+ * Output format is a comma–separated list of decimal numbers and
+ * ranges. Consecutively set bits are shown as two hyphen–separated
+ * decimal numbers, the smallest and largest bit numbers set in
+ * the range. Output format is a compatible subset of the format
+ * accepted as input by bitmap_parselist().
+ *
+ * The return value is the number of characters which would be
+ * generated for the given input, excluding the trailing '\0', as
+ * per ISO C99.
+ */
+
+ int bitmap_scnlistprintf(char *buf, unsigned int buflen,
+ const unsigned long *maskp, int nmaskbits)
+ {
+ int len = 0;
+ /* current bit is 'cur', most recently seen range is [rbot, rtop] */
+ int cur, rbot, rtop;
+
+ rbot = cur = find_first_bit(maskp, nmaskbits);
+ while (cur < nmaskbits) {
+ rtop = cur;
+ cur = find_next_bit(maskp, nmaskbits, cur+1);
+ if (cur >= nmaskbits || cur > rtop + 1) {
+ len = bscnl_emit(buf, buflen, rbot, rtop, len);
+ rbot = cur;
+ }
+ }
+ return len;
+ }
+ EXPORT_SYMBOL(bitmap_scnlistprintf);
+
+ /**
+ * bitmap_parselist – parses a more flexible format for inputting bit masks
+ * @buf: read nul–terminated user string from this buffer
+ * @mask: write resulting mask here
+ * @nmaskbits: number of bits in mask to be written
+ *
+ * The input format supports a space separated list of one or more comma
```

Linux-Kernel: [PATCH] new bitmap list format (for cpusets)

```
+ * separated sequences of ascii decimal bit numbers and ranges. Each
+ * sequence may be preceded by one of the prefix characters '=',
+ * '-', '+', or '!', which have the following meanings:
+ * '=': rewrite the mask to have only the bits specified in this sequence
+ * '-': turn off the bits specified in this sequence
+ * '+': turn on the bits specified in this sequence
+ * '!': same as '-'.
+ *
+ * If no such initial character is specified, then the default prefix '='
+ * is presumed. The list is evaluated and applied in left to right order.
+ *
+ * Examples of input format:
+ * 0-4,9 # rewrites to 0,1,2,3,4,9
+ * -9 # removes 9
+ * +6-8 # adds 6,7,8
+ * 1-6 -0,2-4 +11-14,16-19 -14-16 # same as 1,5,6,11-13,17-19
+ * 1-6 -0,2-4 +11-14,16-19 =14-16 # same as just 14,15,16
+ *
+ * Possible errno's returned for invalid input strings are:
+ * -EINVAL: second number in range smaller than first
+ * -ERANGE: bit number specified too large for mask
+ * -EINVAL: invalid prefix char (not '=', '-', '+', or '!')
+ */
+
+int bitmap_parselist(const char *buf, unsigned long *maskp, int nmaskbits)
+{
+ char *p, *q;
+ int masklen = BITS_TO_LONGS(nmaskbits);
+
+ while ((p = strsep((char **)&buf, " ")) != NULL) { /* blows const XXX */
+ char op = isdigit(*p) ? '=' : *p++;
+ unsigned long m[masklen];
+ int maskbytes = sizeof(m);
+ int i;
+
+ if (op == ' ')
+ continue;
+ memset(m, 0, maskbytes);
+
+ while ((q = strsep(&p, ",")) != NULL) {
+ unsigned a = simple_strtoul(q, 0, BASEDEC);
+ unsigned b = a;
+ char *cp = strchr(q, '-');
+ if (cp)
+ b = simple_strtoul(cp + 1, 0, BASEDEC);
+ if (!(a <= b))
+ return -EINVAL;
+ if (b >= nmaskbits)
+ return -ERANGE;
+ while (a <= b) {
+ set_bit(a, m);
```

Linux-Kernel: [PATCH] new bitmap list format (for cpusets)

```
+ a++;
+ }
+ }
+
+ switch (op) {
+ case '=':
+ memcpy(maskp, m, maskbytes);
+ break;
+ case '!':
+ case '-':
+ for (i = 0; i < masklen; i++)
+ maskp[i] &= ~m[i];
+ break;
+ case '+':
+ for (i = 0; i < masklen; i++)
+ maskp[i] |= m[i];
+ break;
+ default:
+ return -EINVAL;
+ }
+ }
+ return 0;
+}
+EXPORT_SYMBOL(bitmap_parselist);
+
+/**
+ * bitmap_find_free_region - find a contiguous aligned mem region
+ * @bitmap: an array of unsigned longs corresponding to the bitmap
```

--

```
    I won't rest till it's the best ...
    Programmer, Linux Scalability
    Paul Jackson <pj@sgi.com> 1.650.933.1373
```

-

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/>