Subject: Per container statistics (containerstats)
Posted by Balbir Singh on Wed, 06 Jun 2007 11:58:13 GMT
View Forum Message <> Reply to Message

Hi, Andrew/Paul,

Here's the latest version of containerstats ported to v10. Could you
please consider it for inclusion

Changelog

1. Instead of parsing long container path's use the dentry to match the
   container for which stats are required. The user space application
   opens the container directory and passes the file descriptor, which
   is used to determine the container for which stats are required.
   This approach was suggested by Paul Menage

This patch is inspired by the discussion at http://lkml.org/lkml/2007/4/11/187
and implements per container statistics as suggested by Andrew Morton
in http://lkml.org/lkml/2007/4/11/263. The patch is on top of 2.6.21-mm1
with Paul's containers v9 patches (forward ported)

This patch implements per container statistics infrastructure and re-uses
code from the taskstats interface. A new set of container operations are
registered with commands and attributes. It should be very easy to
extend per container statistics, by adding members to the containerstats
structure.

The current model for containerstats is a pull, a push model (to post
statistics on interesting events), should be very easy to add. Currently
user space requests for statistics by passing the container file descriptor.
Statistics about the state of all the tasks in the container is returned to
user space.

TODO's/NOTE:

This patch provides an infrastructure for implementing container statistics.
Based on the needs of each controller, we can incrementally add more statistics,
event based support for notification of statistics, accumulation of taskstats
into container statistics in the future.

Sample output

./containerstats -C /container/a
sleeping 2, blocked 0, running 1, stopped 0, uninterruptible 0

./containerstats -C /container/
sleeping 154, blocked 0, running 0, stopped 0, uninterruptible 0

If the approach looks good, I'll enhance and post the user space utility for
the same

Feedback, comments, test results are always welcome!

Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

```
 Documentation/accounting/containerstats.txt |   27 ++++++++++
 include/linux/Kbuild                        |    1
 include/linux/container.h                   |    8 +++
 include/linux/containerstats.h              |   70 ++++++++++++++++++++++++++++
 include/linux/delayacct.h                   |   11 ++++
 kernel/container.c                          |   63 ++++++++++++++++++++++++++
 kernel/sched.c                              |    4 +
 kernel/taskstats.c                          |   66 ++++++++++++++++++++++++++++
 8 files changed, 250 insertions(+)
```

diff -puN /dev/null Documentation/accounting/containerstats.txt
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/Documentation/accounting/contain erstats.txt 2007-06-06 17:16:54.000000000 +0530
@@ -0,0 +1,27 @@
+Containerstats is inspired by the discussion at
+http://lkml.org/lkml/2007/4/11/187 and implements per container statistics as
+suggested by Andrew Morton in http://lkml.org/lkml/2007/4/11/263.
+
+Per container statistics infrastructure re-uses code from the taskstats
+interface. A new set of container operations are registered with commands
+and attributes specific to containers. It should be very easy to
+extend per container statistics, by adding members to the containerstats
+structure.
+
+The current model for containerstats is a pull, a push model (to post
+statistics on interesting events), should be very easy to add. Currently
+user space requests for statistics by passing the container path.
+Statistics about the state of all the tasks in the container is returned to
+user space.
+
+NOTE: We currently rely on delay accounting for extracting information
+about tasks blocked on I/O. If CONFIG_TASK_DELAY_ACCT is disabled, this
+information will not be available.
+
+To extract container statistics a utility very similar to getdelays.c
+has been developed, the sample output of the utility is shown below

```
+
+~/balbir/containerstats # ./containerstats  -C "/container/a"
+sleeping 1, blocked 0, running 1, stopped 0, uninterruptible 0
+~/balbir/containerstats # ./containerstats  -C "/container"
+sleeping 155, blocked 0, running 1, stopped 0, uninterruptible 2
diff -puN include/linux/container.h~containers-taskstats include/linux/container.h
---  linux-2.6.22-rc2-mm1/include/linux/container.h~containers-ta skstats 2007-06-05
17:21:57.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/include/linux/container.h 2007-06-06 16:59:30.000000000 +0530
@@ -12,6 +12,7 @@
 #include <linux/kref.h>
 #include <linux/cpumask.h>
 #include <linux/nodemask.h>
+#include <linux/containerstats.h>

 #ifdef CONFIG_CONTAINERS

@@ -21,6 +22,8 @@ extern void container_init_smp(void);
 extern void container_fork(struct task_struct *p);
 extern void container_fork_callbacks(struct task_struct *p);
 extern void container_exit(struct task_struct *p, int run_callbacks);
+extern int containerstats_build(struct containerstats *stats,
+    struct dentry *dentry);

 extern struct file_operations proc_container_operations;

@@ -288,6 +291,11 @@ static inline void container_exit(struct

 static inline void container_lock(void) {}
 static inline void container_unlock(void) {}
+static inline int containerstats_build(struct containerstats *stats,
+     struct dentry *dentry)
+{
+ return -EINVAL;
+}

 #endif /* !CONFIG_CONTAINERS */

diff -puN /dev/null include/linux/containerstats.h
--- /dev/null 2007-06-01 20:42:04.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/include/linux/containerstats.h 2007-06-05 17:23:56.000000000
+0530
@@ -0,0 +1,70 @@
+/* containerstats.h - exporting per-container statistics
+ *

+ * Author Balbir Singh <balbir@linux.vnet.ibm.com>
+ *
```

```
+ * This program is free software; you can redistribute it and/or modify it
+ * under the terms of version 2.1 of the GNU Lesser General Public License
+ * as published by the Free Software Foundation.
+ *
+ * This program is distributed in the hope that it would be useful, but
+ * WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
+ */
+
+#ifndef _LINUX_CONTAINERSTATS_H
+#define _LINUX_CONTAINERSTATS_H
+
+#include <linux/taskstats.h>
+
+/*
+ * Data shared between user space and kernel space on a per container
+ * basis. This data is shared using taskstats.
+ *
+ * Most of these states are derived by looking at the task->state value
+ * For the nr_io_wait state, a flag in the delay accounting structure
+ * indicates that the task is waiting on IO
+ *
+ * Each member is aligned to a 8 byte boundary.
+ */
+struct containerstats {
+ __u64 nr_sleeping;  /* Number of tasks sleeping */
+ __u64 nr_running;  /* Number of tasks running */
+ __u64 nr_stopped;  /* Number of tasks in stopped state */
+ __u64 nr_uninterruptible; /* Number of tasks in uninterruptible */
+     /* state */
+ __u64 nr_io_wait;  /* Number of tasks waiting on IO */
+};
+
+/*
+ * Commands sent from userspace
+ * Not versioned. New commands should only be inserted at the enum's end
+ * prior to __CONTAINERSTATS_CMD_MAX
+ */
+
+enum {
+ CONTAINERSTATS_CMD_UNSPEC = __TASKSTATS_CMD_MAX, /* Reserved */
+ CONTAINERSTATS_CMD_GET,  /* user->kernel request/get-response */
+ CONTAINERSTATS_CMD_NEW,  /* kernel->user event */
+ __CONTAINERSTATS_CMD_MAX,
+};
+
+#define CONTAINERSTATS_CMD_MAX (__CONTAINERSTATS_CMD_MAX - 1)
+
```

```
+enum {
+ CONTAINERSTATS_TYPE_UNSPEC = 0, /* Reserved */
+ CONTAINERSTATS_TYPE_CONTAINER_STATS, /* contains name + stats */
+ __CONTAINERSTATS_TYPE_MAX,
+};
+
+#define CONTAINERSTATS_TYPE_MAX (__CONTAINERSTATS_TYPE_MAX - 1)
+
+enum {
+ CONTAINERSTATS_CMD_ATTR_UNSPEC = 0,
+ CONTAINERSTATS_CMD_ATTR_FD,
+ __CONTAINERSTATS_CMD_ATTR_MAX,
+};
+
+#define CONTAINERSTATS_CMD_ATTR_MAX (__CONTAINERSTATS_CMD_ATTR_MAX - 1)
+
+#endif /* _LINUX_CONTAINERSTATS_H */
diff -puN include/linux/delayacct.h~containers-taskstats include/linux/delayacct.h
---  linux-2.6.22-rc2-mm1/include/linux/delayacct.h~containers-ta skstats 2007-06-05
17:21:57.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/include/linux/delayacct.h 2007-06-05 17:21:57.000000000 +0530
@@ -26,6 +26,7 @@
  * Used to set current->delays->flags
  */
 #define DELAYACCT_PF_SWAPIN 0x00000001 /* I am doing a swapin */
+#define DELAYACCT_PF_BLKIO 0x00000002 /* I am waiting on IO */

 #ifdef CONFIG_TASK_DELAY_ACCT

@@ -39,6 +40,14 @@ extern void __delayacct_blkio_end(void);
 extern int __delayacct_add_tsk(struct taskstats *, struct task_struct *);
 extern __u64 __delayacct_blkio_ticks(struct task_struct *);

+static inline int delayacct_is_task_waiting_on_io(struct task_struct *p)
+{
+ if (p->delays)
+  return (p->delays->flags & DELAYACCT_PF_BLKIO);
+ else
+  return 0;
+}
+
 static inline void delayacct_set_flag(int flag)
 {
  if (current->delays)
@@ -116,6 +125,8 @@ static inline int delayacct_add_tsk(stru
 { return 0; }
 static inline __u64 delayacct_blkio_ticks(struct task_struct *tsk)
 { return 0; }
```

```
+static inline int delayacct_is_task_waiting_on_io(struct task_struct *p)
+{ return 0; }
 #endif /* CONFIG_TASK_DELAY_ACCT */

 #endif
diff -puN include/linux/Kbuild~containers-taskstats include/linux/Kbuild
---  linux-2.6.22-rc2-mm1/include/linux/Kbuild~containers-tasksta ts 2007-06-05
17:21:57.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/include/linux/Kbuild 2007-06-05 17:21:57.000000000 +0530
@@ -47,6 +47,7 @@ header-y += coff.h
 header-y += comstats.h
 header-y += consolemap.h
 header-y += const.h
+header-y += containerstats.h
 header-y += cycx_cfm.h
 header-y += dlm_device.h
 header-y += dlm_netlink.h
diff -puN kernel/container.c~containers-taskstats kernel/container.c
---  linux-2.6.22-rc2-mm1/kernel/container.c~containers-taskstats 2007-06-05
17:21:57.000000000 +0530
+++ linux-2.6.22-rc2-mm1-balbir/kernel/container.c 2007-06-06 17:17:19.000000000 +0530
@@ -59,6 +59,8 @@
 #include <asm/uaccess.h>
 #include <asm/atomic.h>
 #include <linux/mutex.h>
+#include <linux/delayacct.h>
+#include <linux/containerstats.h>

 #define CONTAINER_SUPER_MAGIC  0x27e0eb

@@ -1607,6 +1609,67 @@ static int pid_array_load(pid_t *pidarra
  return n;
 }

+/**
+ * Build and fill containerstats so that taskstats can export it to user
+ * space.
+ *
+ * @stats: containerstats to fill information into
+ * @dentry: A dentry entry belonging to the container for which stats have
+ * been requested.
+ */
+int containerstats_build(struct containerstats *stats, struct dentry *dentry)
+{
+ int ret = -EINVAL;
+ struct task_struct *g, *p;
+ struct container *cont, *root_cont;
+ struct container *src_cont;
```

```
+ int subsys_id;
+ struct containerfs_root *root;
+
+ /*
+  * Validate dentry by checking the superblock operations
+  */
+ if (dentry->d_sb->s_op != &container_ops)
+   goto err;
+
+ ret = 0;
+ src_cont = (struct container *)dentry->d_fsdata;
+ rcu_read_lock();
+
+ for_each_root(root) {
+  if (!root->subsys_bits)
+   continue;
+  root_cont = &root->top_container;
+  get_first_subsys(root_cont, NULL, &subsys_id);
+  do_each_thread(g, p) {
+   cont = task_container(p, subsys_id);
+   if (cont == src_cont) {
+    switch (p->state) {
+    case TASK_RUNNING:
+     stats->nr_running++;
+     break;
+    case TASK_INTERRUPTIBLE:
+     stats->nr_sleeping++;
+     break;
+    case TASK_UNINTERRUPTIBLE:
+     stats->nr_uninterruptible++;
+     break;
+    case TASK_STOPPED:
+     stats->nr_stopped++;
+     break;
+    default:
+     if (delayacct_is_task_waiting_on_io(p))
+      stats->nr_io_wait++;
+     break;
+    }
+   }
+  } while_each_thread(g, p);
+ }
+ rcu_read_unlock();
+err:
+ return ret;
+}
+
 static int cmppid(const void *a, const void *b)
```

```
 {
  return *(pid_t *)a - *(pid_t *)b;
diff -puN kernel/sched.c~containers-taskstats kernel/sched.c
--- linux-2.6.22-rc2-mm1/kernel/sched.c~containers-taskstats 2007-06-05 17:21:57.000000000
+0530
+++ linux-2.6.22-rc2-mm1-balbir/kernel/sched.c 2007-06-05 17:21:57.000000000 +0530
@@ -4280,11 +4280,13 @@ void __sched io_schedule(void)
 {
  struct rq *rq = &__raw_get_cpu_var(runqueues);

+ delayacct_set_flag(DELAYACCT_PF_BLKIO);
  delayacct_blkio_start();
  atomic_inc(&rq->nr_iowait);
  schedule();
  atomic_dec(&rq->nr_iowait);
  delayacct_blkio_end();
+ delayacct_clear_flag(DELAYACCT_PF_BLKIO);
 }
 EXPORT_SYMBOL(io_schedule);

@@ -4293,11 +4295,13 @@ long __sched io_schedule_timeout(long ti
  struct rq *rq = &__raw_get_cpu_var(runqueues);
  long ret;

+ delayacct_set_flag(DELAYACCT_PF_BLKIO);
  delayacct_blkio_start();
  atomic_inc(&rq->nr_iowait);
  ret = schedule_timeout(timeout);
  atomic_dec(&rq->nr_iowait);
  delayacct_blkio_end();
+ delayacct_clear_flag(DELAYACCT_PF_BLKIO);
  return ret;
 }

diff -puN kernel/taskstats.c~containers-taskstats kernel/taskstats.c
---  linux-2.6.22-rc2-mm1/kernel/taskstats.c~containers-taskstats 2007-06-05 17:21:57.000000000
+0530
+++ linux-2.6.22-rc2-mm1-balbir/kernel/taskstats.c 2007-06-06 17:01:12.000000000 +0530
@@ -23,6 +23,9 @@
 #include <linux/tsacct_kern.h>
 #include <linux/cpumask.h>
 #include <linux/percpu.h>
+#include <linux/containerstats.h>
+#include <linux/container.h>
+#include <linux/file.h>
 #include <net/genetlink.h>
 #include <asm/atomic.h>
```

```
@@ -50,6 +53,11 @@ __read_mostly = {
  [TASKSTATS_CMD_ATTR_REGISTER_CPUMASK] = { .type = NLA_STRING },
  [TASKSTATS_CMD_ATTR_DEREGISTER_CPUMASK] = { .type = NLA_STRING },};

+static struct nla_policy
 +containerstats_cmd_get_policy[CONTAINERSTATS_CMD_ATTR_MAX+1 ] __read_mostly = {
+ [CONTAINERSTATS_CMD_ATTR_FD] = { .type = NLA_U32 },
+};
+
 struct listener {
  struct list_head list;
  pid_t pid;
@@ -369,6 +377,51 @@ err:
  return NULL;
 }

+static int containerstats_user_cmd(struct sk_buff *skb, struct genl_info *info)
+{
+ int rc = 0;
+ struct sk_buff *rep_skb;
+ struct containerstats *stats;
+ struct nlattr *na;
+ size_t size;
+ u32 fd;
+ struct file *file;
+ int fput_needed;
+
+ na = info->attrs[CONTAINERSTATS_CMD_ATTR_FD];
+ if (!na)
+  return -EINVAL;
+
+ fd = nla_get_u32(info->attrs[CONTAINERSTATS_CMD_ATTR_FD]);
+ file = fget_light(fd, &fput_needed);
+ if (file) {
+  size = nla_total_size(sizeof(struct containerstats));
+
+  rc = prepare_reply(info, CONTAINERSTATS_CMD_NEW, &rep_skb,
+     size);
+  if (rc < 0)
+   goto err;
+
+  na = nla_reserve(rep_skb, CONTAINERSTATS_TYPE_CONTAINER_STATS,
+     sizeof(struct containerstats));
+  stats = nla_data(na);
+  memset(stats, 0, sizeof(*stats));
+
+  rc = containerstats_build(stats, file->f_dentry);
+  if (rc < 0)
```

```
+   goto err;
+
+  fput_light(file, fput_needed);
+  return send_reply(rep_skb, info->snd_pid);
+ }
+
+err:
+ if (file)
+  fput_light(file, fput_needed);
+ nlmsg_free(rep_skb);
+ return rc;
+}
+
 static int taskstats_user_cmd(struct sk_buff *skb, struct genl_info *info)
 {
  int rc = 0;
@@ -519,6 +572,12 @@ static struct genl_ops taskstats_ops = {
  .policy  = taskstats_cmd_get_policy,
 };

+static struct genl_ops containerstats_ops = {
+ .cmd  = CONTAINERSTATS_CMD_GET,
+ .doit  = containerstats_user_cmd,
+ .policy  = containerstats_cmd_get_policy,
+};
+
 /* Needed early in initialization */
 void __init taskstats_init_early(void)
 {
@@ -543,8 +602,15 @@ static int __init taskstats_init(void)
  if (rc < 0)
   goto err;

+ rc = genl_register_ops(&family, &containerstats_ops);
+ if (rc < 0)
+  goto err_container_ops;
+
  family_registered = 1;
+ printk("registered taskstats version %d\n", TASKSTATS_GENL_VERSION);
  return 0;
+err_container_ops:
+ genl_unregister_ops(&family, &taskstats_ops);
 err:
  genl_unregister_family(&family);
  return rc;
diff -puN kernel/cpuset.c~containers-taskstats kernel/cpuset.c
_
--
```

Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

Subject: Re: Per container statistics (containerstats)
Posted by Andrew Morton on Thu, 07 Jun 2007 22:54:45 GMT
View Forum Message <> Reply to Message

On Wed, 6 Jun 2007 17:28:13 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> Hi, Andrew/Paul,
>
> Here's the latest version of containerstats ported to v10. Could you
> please consider it for inclusion
>
> Changelog
>
> 1. Instead of parsing long container path's use the dentry to match the
>   container for which stats are required. The user space application
>   opens the container directory and passes the file descriptor, which
>   is used to determine the container for which stats are required.
>   This approach was suggested by Paul Menage
>
> This patch is inspired by the discussion at http://lkml.org/lkml/2007/4/11/187
> and implements per container statistics as suggested by Andrew Morton
> in http://lkml.org/lkml/2007/4/11/263. The patch is on top of 2.6.21-mm1
> with Paul's containers v9 patches (forward ported)
>
> This patch implements per container statistics infrastructure and re-uses
> code from the taskstats interface. A new set of container operations are
> registered with commands and attributes. It should be very easy to
> *extend* per container statistics, by adding members to the containerstats
> structure.
>
> The current model for containerstats is a pull, a push model (to post
> statistics on interesting events), should be very easy to add. Currently
> user space requests for statistics by passing the container file descriptor.
> Statistics about the state of all the tasks in the container is returned to
> user space.
>
> TODO's/NOTE:
>
> This patch provides an infrastructure for implementing container statistics.
> Based on the needs of each controller, we can incrementally add more statistics,
> event based support for notification of statistics, accumulation of taskstats

> into container statistics in the future.
>
> Sample output
>
> # ./containerstats -C /container/a
> sleeping 2, blocked 0, running 1, stopped 0, uninterruptible 0
>
> # ./containerstats -C /container/
> sleeping 154, blocked 0, running 0, stopped 0, uninterruptible 0
>
> If the approach looks good, I'll enhance and post the user space utility for
> the same
>
> Feedback, comments, test results are always welcome!
>
>
>
> Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
> ---
>
> Documentation/accounting/containerstats.txt | 27 ++++++++++
> include/linux/Kbuild | 1
> include/linux/container.h | 8 +++
> include/linux/containerstats.h | 70 ++++++++++++++++++++++++++++
> include/linux/delayacct.h | 11 ++++
> kernel/container.c | 63 ++++++++++++++++++++++++++
> kernel/sched.c | 4 +
> kernel/taskstats.c | 66 ++++++++++++++++++++++++++++
> 8 files changed, 250 insertions(+)

I'd have hoped to see containerstats.c in here.

> diff -puN /dev/null include/linux/containerstats.h
> --- /dev/null 2007-06-01 20:42:04.000000000 +0530
> +++ linux-2.6.22-rc2-mm1-balbir/include/linux/containerstats.h 2007-06-05 17:23:56.000000000 +0530
> @@ -0,0 +1,70 @@
> +/* containerstats.h - exporting per-container statistics
> + *
> + * __ Copyright IBM Corporation, 2007
> + * Author Balbir Singh <balbir@linux.vnet.ibm.com>
> + *
> + * This program is free software; you can redistribute it and/or modify it
> + * under the terms of version 2.1 of the GNU Lesser General Public License
> + * as published by the Free Software Foundation.
> + *
> + * This program is distributed in the hope that it would be useful, but
> + * WITHOUT ANY WARRANTY; without even the implied warranty of

> + * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
> + */
> +
> +#ifndef _LINUX_CONTAINERSTATS_H
> +#define _LINUX_CONTAINERSTATS_H
> +
> +#include <linux/taskstats.h>

I don't understand the relationship between containerstats and taskstats.
afacit it's using the same genetlink channel?

> +/*
> + * Data shared between user space and kernel space on a per container
> + * basis. This data is shared using taskstats.
> + *
> + * Most of these states are derived by looking at the task->state value
> + * For the nr_io_wait state, a flag in the delay accounting structure
> + * indicates that the task is waiting on IO
> + *
> + * Each member is aligned to a 8 byte boundary.
> + */
> +struct containerstats {
> + __u64 nr_sleeping; /* Number of tasks sleeping */
> + __u64 nr_running; /* Number of tasks running */
> + __u64 nr_stopped; /* Number of tasks in stopped state */
> + __u64 nr_uninterruptible; /* Number of tasks in uninterruptible */
> +   /* state */
> + __u64 nr_io_wait; /* Number of tasks waiting on IO */
> +};
> +
> +/*
> + * Commands sent from userspace
> + * Not versioned. New commands should only be inserted at the enum's end
> + * prior to __CONTAINERSTATS_CMD_MAX
> + */
> +
> +enum {
> + CONTAINERSTATS_CMD_UNSPEC = __TASKSTATS_CMD_MAX, /* Reserved */

This seems to mean that the containerstats commands all get renumbered if
we add new taskstats commands. That would be bad?

> + */
> +int containerstats_build(struct containerstats *stats, struct dentry *dentry)
> +{
> + int ret = -EINVAL;
> + struct task_struct *g, *p;
> + struct container *cont, *root_cont;

```
> + struct container *src_cont;
> + int subsys_id;
> + struct containerfs_root *root;
> +
> + /*
> +  * Validate dentry by checking the superblock operations
> +  */
> + if (dentry->d_sb->s_op != &container_ops)
> +   goto err;
> +
> + ret = 0;
> + src_cont = (struct container *)dentry->d_fsdata;
```

Unneeded cast.

```
> + rcu_read_lock();
> +
> + for_each_root(root) {
> +  if (!root->subsys_bits)
> +   continue;
> +  root_cont = &root->top_container;
> +  get_first_subsys(root_cont, NULL, &subsys_id);
> +  do_each_thread(g, p) {
```

this needs tasklist_lock?

```
> +   cont = task_container(p, subsys_id);
> +   if (cont == src_cont) {
> +    switch (p->state) {
> +    case TASK_RUNNING:
> +     stats->nr_running++;
> +     break;
> +    case TASK_INTERRUPTIBLE:
> +     stats->nr_sleeping++;
> +     break;
> +    case TASK_UNINTERRUPTIBLE:
> +     stats->nr_uninterruptible++;
> +     break;
> +    case TASK_STOPPED:
> +     stats->nr_stopped++;
> +     break;
> +    default:
> +     if (delayacct_is_task_waiting_on_io(p))
> +      stats->nr_io_wait++;
> +     break;
> +    }
> +   }
> +  } while_each_thread(g, p);
```

> + }
> + rcu_read_unlock();
> +err:
> + return ret;
> +}
> +
> static int cmppid(const void *a, const void *b)
> {
>   return *(pid_t *)a - *(pid_t *)b;
> diff -puN kernel/sched.c~containers-taskstats kernel/sched.c
> --- linux-2.6.22-rc2-mm1/kernel/sched.c~containers-taskstats 2007-06-05 17:21:57.000000000 +0530
> +++ linux-2.6.22-rc2-mm1-balbir/kernel/sched.c 2007-06-05 17:21:57.000000000 +0530
> @@ -4280,11 +4280,13 @@ void __sched io_schedule(void)
> {
>   struct rq *rq = &__raw_get_cpu_var(runqueues);
>
> + delayacct_set_flag(DELAYACCT_PF_BLKIO);
>   delayacct_blkio_start();

Would it be suitable and appropriate to embed the delayacct_set_flag() call
inside delayacct_blkio_start()?

Subject: Re: Per container statistics (containerstats)
Posted by Balbir Singh on Fri, 08 Jun 2007 02:21:12 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
>
> I'd have hoped to see containerstats.c in here.
>

The current statistics code is really small, so it fit into taskstats.c.
May be in the future, we could re-factor it and move it out.

>> +#ifndef _LINUX_CONTAINERSTATS_H
>> +#define _LINUX_CONTAINERSTATS_H
>> +
>> +#include <linux/taskstats.h>
>
> I don't understand the relationship between containerstats and taskstats.
> afacit it's using the same genetlink channel?
>

I've registered containerstats_ops with the same family as taskstats to
reuse the taskstats interface.

>> +enum {
>> + CONTAINERSTATS_CMD_UNSPEC = __TASKSTATS_CMD_MAX, /* Reserved */
>
> This seems to mean that the containerstats commands all get renumbered if
> we add new taskstats commands. That would be bad?
>

As per comment above, since we register containerstats_ops with the taskstats
family, the commands need to be unique, hence we start where taskstats ended
(left off)

>> + */
>> +int containerstats_build(struct containerstats *stats, struct dentry *dentry)
>> +{
>> + int ret = -EINVAL;
>> + struct task_struct *g, *p;
>> + struct container *cont, *root_cont;
>> + struct container *src_cont;
>> + int subsys_id;
>> + struct containerfs_root *root;
>> +
>> + /*
>> + * Validate dentry by checking the superblock operations
>> + */
>> + if (dentry->d_sb->s_op != &container_ops)
>> + goto err;
>> +
>> + ret = 0;
>> + src_cont = (struct container *)dentry->d_fsdata;
>
> Unneeded cast.
>

Will remove

>> + rcu_read_lock();
>> +
>> + for_each_root(root) {
>> + if (!root->subsys_bits)
>> +   continue;
>> + root_cont = &root->top_container;
>> + get_first_subsys(root_cont, NULL, &subsys_id);
>> + do_each_thread(g, p) {
>
> this needs tasklist_lock?
>

rcu_read_lock() should be fine. From Eric's patch at

2.6.17-mm2 - proc-remove-tasklist_lock-from-proc_pid_readdir.patch

The patch mentions that "We don't need the tasklist_lock to safely
iterate through processes anymore."

```
>> +   cont = task_container(p, subsys_id);
>> +   if (cont == src_cont) {
>> +    switch (p->state) {
>> +    case TASK_RUNNING:
>> +     stats->nr_running++;
>> +     break;
>> +    case TASK_INTERRUPTIBLE:
>> +     stats->nr_sleeping++;
>> +     break;
>> +    case TASK_UNINTERRUPTIBLE:
>> +     stats->nr_uninterruptible++;
>> +     break;
>> +    case TASK_STOPPED:
>> +     stats->nr_stopped++;
>> +     break;
>> +    default:
>> +     if (delayacct_is_task_waiting_on_io(p))
>> +      stats->nr_io_wait++;
>> +     break;
>> +    }
>> +   }
>> +  } while_each_thread(g, p);
>> + }
>> + rcu_read_unlock();
>> +err:
>> + return ret;
>> +}
>> +
>>  static int cmppid(const void *a, const void *b)
>>  {
>>   return *(pid_t *)a - *(pid_t *)b;
>> diff -puN kernel/sched.c~containers-taskstats kernel/sched.c
>> --- linux-2.6.22-rc2-mm1/kernel/sched.c~containers-taskstats 2007-06-05 17:21:57.000000000
+0530
>> +++ linux-2.6.22-rc2-mm1-balbir/kernel/sched.c 2007-06-05 17:21:57.000000000 +0530
>> @@ -4280,11 +4280,13 @@ void __sched io_schedule(void)
>>  {
>>   struct rq *rq = &__raw_get_cpu_var(runqueues);
>>
>> + delayacct_set_flag(DELAYACCT_PF_BLKIO);
>>   delayacct_blkio_start();
>
```

> Would it be suitable and appropriate to embed the delayacct_set_flag() call
> inside delayacct_blkio_start()?
>

Yes, I should have done that, will do.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

Subject: Re: Per container statistics (containerstats)
Posted by Paul Menage on Fri, 08 Jun 2007 02:28:03 GMT
View Forum Message <> Reply to Message

On 6/7/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> > this needs tasklist_lock?
> >
>
> rcu_read_lock() should be fine. From Eric's patch at
>
> 2.6.17-mm2 - proc-remove-tasklist_lock-from-proc_pid_readdir.patch
>
> The patch mentions that "We don't need the tasklist_lock to safely
> iterate through processes anymore."

Containers V10 includes an iterator interface for listing the member
tasks of a container, which avoids scanning the entire tasklist.
Downside is that it currently requires taking a read_lock on a global
lock, but I hope to improve on that in the future.

Paul

Subject: Re: Per container statistics (containerstats)
Posted by Andrew Morton on Fri, 08 Jun 2007 02:39:17 GMT
View Forum Message <> Reply to Message

On Fri, 08 Jun 2007 07:51:12 +0530 Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> Andrew Morton wrote:
> >
> > I'd have hoped to see containerstats.c in here.
> >

>
> The current statistics code is really small, so it fit into taskstats.c.
> May be in the future, we could re-factor it and move it out.

I was referring to your userspace tool which reads this stuff. The one
which you described in the changelog.

> >> + rcu_read_lock();
> >> +
> >> + for_each_root(root) {
> >> + if (!root->subsys_bits)
> >> + continue;
> >> + root_cont = &root->top_container;
> >> + get_first_subsys(root_cont, NULL, &subsys_id);
> >> + do_each_thread(g, p) {
> >
> > this needs tasklist_lock?
> >
>
> rcu_read_lock() should be fine. From Eric's patch at
>
> 2.6.17-mm2 - proc-remove-tasklist_lock-from-proc_pid_readdir.patch
>
> The patch mentions that "We don't need the tasklist_lock to safely
> iterate through processes anymore."
>

oh, OK. rcu_read_lock() is the new lock_kernel() - always hard to tell
what it's locking.

Subject: Re: Per container statistics (containerstats)
Posted by Balbir Singh on Fri, 08 Jun 2007 02:44:01 GMT

View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Fri, 08 Jun 2007 07:51:12 +0530 Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
>> Andrew Morton wrote:
>>> I'd have hoped to see containerstats.c in here.
>>>
>> The current statistics code is really small, so it fit into taskstats.c.
>> May be in the future, we could re-factor it and move it out.
>
> I was referring to your userspace tool which reads this stuff. The one
> which you described in the changelog.
>

Ahh.. yes.. I'll make the changes and repost with the tool.

>>>> + rcu_read_lock();
>>>> +
>>>> + for_each_root(root) {
>>>> + if (!root->subsys_bits)
>>>> +   continue;
>>>> + root_cont = &root->top_container;
>>>> + get_first_subsys(root_cont, NULL, &subsys_id);
>>>> + do_each_thread(g, p) {
>>> this needs tasklist_lock?
>>>
>> rcu_read_lock() should be fine. From Eric's patch at
>>
>> 2.6.17-mm2 - proc-remove-tasklist_lock-from-proc_pid_readdir.patch
>>
>> The patch mentions that "We don't need the tasklist_lock to safely
>> iterate through processes anymore."
>>
>
> oh, OK. rcu_read_lock() is the new lock_kernel() - always hard to tell
> what it's locking.
>

:-) I'll add a comment to make it clearer.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

Subject: Re: [ckrm-tech] Per container statistics (containerstats)
Posted by Balbir Singh on Fri, 08 Jun 2007 10:50:48 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
>>>   return *(pid_t *)a - *(pid_t *)b;
>>> diff -puN kernel/sched.c~containers-taskstats kernel/sched.c
>>> --- linux-2.6.22-rc2-mm1/kernel/sched.c~containers-taskstats 2007-06-05
17:21:57.000000000 +0530
>>> +++ linux-2.6.22-rc2-mm1-balbir/kernel/sched.c 2007-06-05 17:21:57.000000000 +0530
>>> @@ -4280,11 +4280,13 @@ void __sched io_schedule(void)
>>> {
>>>   struct rq *rq = &__raw_get_cpu_var(runqueues);
>>>

>>> + delayacct_set_flag(DELAYACCT_PF_BLKIO);
>>>   delayacct_blkio_start();
>> Would it be suitable and appropriate to embed the delayacct_set_flag() call
>> inside delayacct_blkio_start()?
>>
>
> Yes, I should have done that, will do.
>
>

My last reply was wrong, the flag needs to be set outside so that we know the
context that initiated block I/O. We use this flag to track the time used
by the swap block I/O.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

Subject: Re: [ckrm-tech] Per container statistics (containerstats)
Posted by Balbir Singh on Fri, 08 Jun 2007 10:57:36 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Balbir Singh wrote:
>>>>   return *(pid_t *)a - *(pid_t *)b;
>>>> diff -puN kernel/sched.c~containers-taskstats kernel/sched.c
>>>> --- linux-2.6.22-rc2-mm1/kernel/sched.c~containers-taskstats 2007-06-05
17:21:57.000000000 +0530
>>>> +++ linux-2.6.22-rc2-mm1-balbir/kernel/sched.c 2007-06-05 17:21:57.000000000 +0530
>>>> @@ -4280,11 +4280,13 @@ void __sched io_schedule(void)
>>>> {
>>>>   struct rq *rq = &__raw_get_cpu_var(runqueues);
>>>>
>>>> + delayacct_set_flag(DELAYACCT_PF_BLKIO);
>>>>   delayacct_blkio_start();
>>> Would it be suitable and appropriate to embed the delayacct_set_flag() call
>>> inside delayacct_blkio_start()?
>>>
>> Yes, I should have done that, will do.
>>
>>
>
> My last reply was wrong, the flag needs to be set outside so that we know the
> context that initiated block I/O. We use this flag to track the time used
> by the swap block I/O.

>

Please ignore this email.. it escaped my drafts folder... my fault.. apologies.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL