Subject: [PATCH 0/7] Profiling sleep times (v3)
Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:28 GMT
View Forum Message <> Reply to Message

Do you want to know where your code waits locks for a long time?
Yes! It's for you. This feature helps you to find bottlenecks.

It's not artificial task. Once one of my colleague was investigating a
scalability problem. He pressed sysrq-t some times and tried to merge
call-chains by hand. But perf can do that.

Problem:
The problem is that events sched_stat_* contain call-chains of
non-target tasks.
About month ago I sent series of patches:

[PATCH 0/3] trace: add ability to collect call chains of non current task.

Peter and Frederic explained me, that this solve isn't good and will be
better to make it in userspace.

Now it's in userspace. This series expands "perf inject" to be able to
merge sched_switch events and sched_stat* events. sched_switch events
contain correct call-chains and sched_stat contains a correct time
slices.

v2:
* Removed all known issues. Now it works completely.
* Improved usability of sched-stat scripts according with Arun's comments.
v3: fixed accoding to comments from David Ahem

Andrew Vagin (7):
  perf: use event_name() to get an event name
  perf: add ability to record event period
  perf: add ability to change event according to sample (v2)
  perf: teach "perf inject" to work with files
  perf: teach perf inject to merge sched_stat_* and sched_switch events
  perf: add scripts for profiling sleep times (v2)
  event: add tracepoint for accounting block time

 include/trace/events/sched.h | 7 ++
 kernel/sched_fair.c | 2 +
 tools/perf/builtin-inject.c | 120 ++++++++++++++++++++++++-
 tools/perf/builtin-record.c | 5 +
 tools/perf/scripts/python/bin/sched-stat-record | 65 ++++++++++++
 tools/perf/scripts/python/bin/sched-stat-report | 5 +
 tools/perf/util/event.h | 2 +
 tools/perf/util/evsel.c | 74 +++++++++++++

```
 tools/perf/util/header.c                   |    2 +-
 tools/perf/util/session.h                  |    9 ++
 10 files changed, 288 insertions(+), 3 deletions(-)
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-record
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-report
```

Subject: [PATCH 2/7] perf: add ability to record event period
Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:30 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/builtin-record.c |    5 +++++
 1 files changed, 5 insertions(+), 0 deletions(-)

diff --git a/tools/perf/builtin-record.c b/tools/perf/builtin-record.c
index 6ab58cc..e3b7fc4 100644
--- a/tools/perf/builtin-record.c
+++ b/tools/perf/builtin-record.c
@@ -62,6 +62,7 @@ static bool   no_samples   =  false;
 static bool   sample_address   =  false;
 static bool   sample_time   =  false;
 static bool   no_buildid   =  false;
+static bool   period    =  false;
 static bool   no_buildid_cache  =  false;
 static struct perf_evlist *evsel_list;

@@ -185,6 +186,9 @@ static void config_attr(struct perf_evsel *evsel, struct perf_evlist *evlist)
  if (evlist->nr_entries > 1)
   attr->sample_type |= PERF_SAMPLE_ID;

+ if (period)
+  attr->sample_type |= PERF_SAMPLE_PERIOD;
+
  /*
   * We default some events to a 1 default interval. But keep
   * it a weak assumption overridable by the user.
@@ -803,6 +807,7 @@ const struct option record_options[] = {
  OPT_BOOLEAN('d', "data", &sample_address,
       "Sample addresses"),
  OPT_BOOLEAN('T', "timestamp", &sample_time, "Sample timestamps"),
+ OPT_BOOLEAN('P', "period", &period, "Sample period"),
  OPT_BOOLEAN('n', "no-samples", &no_samples,
       "don't sample"),
  OPT_BOOLEAN('N', "no-buildid-cache", &no_buildid_cache,
--
1.7.1
```

Subject: [PATCH 3/7] perf: add ability to change event according to sample (v2)
Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:31 GMT
View Forum Message <> Reply to Message

It's opposition of perf_session__parse_sample.

v2: fixed mistakes which David Arhen found

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/util/event.h | 2 +
 tools/perf/util/evsel.c | 74 ++
 tools/perf/util/session.h | 9 +++++
 3 files changed, 85 insertions(+), 0 deletions(-)

```
diff --git a/tools/perf/util/event.h b/tools/perf/util/event.h
index 357a85b..0493101 100644
--- a/tools/perf/util/event.h
+++ b/tools/perf/util/event.h
@@ -187,5 +187,7 @@ const char *perf_event__name(unsigned int id);
 int perf_event__parse_sample(const union perf_event *event, u64 type,
        int sample_size, bool sample_id_all,
        struct perf_sample *sample, bool swapped);
+int perf_event__change_sample(union perf_event *event, u64 type,
+       const struct perf_sample *data, bool swapped);

 #endif /* __PERF_RECORD_H */
diff --git a/tools/perf/util/evsel.c b/tools/perf/util/evsel.c
index e426264..d697568 100644
--- a/tools/perf/util/evsel.c
+++ b/tools/perf/util/evsel.c
@@ -494,3 +494,77 @@ int perf_event__parse_sample(const union perf_event *event, u64 type,

  return 0;
 }
+
+int perf_event__change_sample(union perf_event *event, u64 type,
+       const struct perf_sample *data, bool swapped)
+{
+ u64 *array;
+
+ /*
+  * used for cross-endian analysis. See git commit 65014ab3
+  * for why this goofiness is needed.
+  */
+ union {
+  u64 val64;
+  u32 val32[2];
+ } u;
```

```
+
+ array = event->sample.array;
+
+ if (type & PERF_SAMPLE_IP) {
+  event->ip.ip = data->ip;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_TID) {
+  u.val32[0] = data->pid;
+  u.val32[1] = data->tid;
+  if (swapped) {
+   /* undo swap of u64, then swap on individual u32s */
+   u.val32[0] = bswap_32(u.val32[0]);
+   u.val32[1] = bswap_32(u.val32[1]);
+   u.val64 = bswap_64(u.val64);
+  }
+
+  *array = u.val64;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_TIME) {
+  *array = data->time;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_ADDR) {
+  *array = data->addr;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_ID) {
+  *array = data->id;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_STREAM_ID) {
+  *array = data->stream_id;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_CPU) {
+  u.val32[0] = data->cpu;
+  if (swapped) {
+   /* undo swap of u64, then swap on individual u32s */
+   u.val32[0] = bswap_32(u.val32[0]);
+   u.val64 = bswap_64(u.val64);
```

```
+  }
+  *array = u.val64;
+  array++;
+ }
+
+ if (type & PERF_SAMPLE_PERIOD) {
+  *array = data->period;
+  array++;
+ }
+
+ return 0;
+}
diff --git a/tools/perf/util/session.h b/tools/perf/util/session.h
index 6e393c9..444f121 100644
--- a/tools/perf/util/session.h
+++ b/tools/perf/util/session.h
@@ -167,6 +167,15 @@ static inline int perf_session__parse_sample(struct perf_session
*session,
      session->header.needs_swap);
 }

+static inline int perf_session__change_sample(struct perf_session *session,
+          union perf_event *event,
+          const struct perf_sample *sample)
+{
+ return perf_event__change_sample(event, session->sample_type,
+     sample,
+     session->header.needs_swap);
+}
+
 struct perf_evsel *perf_session__find_first_evtype(struct perf_session *session,
         unsigned int type);

--
1.7.1
```

Subject: [PATCH 4/7] perf: teach "perf inject" to work with files
Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:32 GMT
View Forum Message <> Reply to Message

Before this patch "perf inject" can only handle data from pipe.

I want to use "perf inject" for reworking events. Look at my following patch.

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/builtin-inject.c | 33 +++++++++++++++++++++++++++++++--

```
 1 files changed, 31 insertions(+), 2 deletions(-)

diff --git a/tools/perf/builtin-inject.c b/tools/perf/builtin-inject.c
index 8dfc12b..8df8b71 100644
--- a/tools/perf/builtin-inject.c
+++ b/tools/perf/builtin-inject.c
@@ -13,7 +13,12 @@

 #include "util/parse-options.h"

-static char  const *input_name = "-";
+static char  const *input_name = "-";
+static const char *output_name  = "-";
+static int  pipe_output  = 0;
+static int  output;
+static u64  bytes_written  = 0;
+
 static bool  inject_build_ids;

 static int perf_event__repipe_synth(union perf_event *event,
@@ -25,12 +30,14 @@ static int perf_event__repipe_synth(union perf_event *event,
  size = event->header.size;

  while (size) {
-  int ret = write(STDOUT_FILENO, buf, size);
+  int ret = write(output, buf, size);
   if (ret < 0)
    return -errno;

   size -= ret;
   buf += ret;
+
+  bytes_written += ret;
  }

  return 0;
@@ -213,8 +220,14 @@ static int __cmd_inject(void)
  if (session == NULL)
   return -ENOMEM;

+ if (!pipe_output)
+  lseek(output, session->header.data_offset, SEEK_SET);
  ret = perf_session__process_events(session, &inject_ops);

+ if (!pipe_output) {
+  session->header.data_size += bytes_written;
+  perf_session__write_header(session, session->evlist, output, true);
+ }
```

```
  perf_session__delete(session);

  return ret;
@@ -228,6 +241,10 @@ static const char * const report_usage[] = {
 static const struct option options[] = {
  OPT_BOOLEAN('b', "build-ids", &inject_build_ids,
       "Inject build-ids into the output stream"),
+ OPT_STRING('i', "input", &input_name, "file",
+       "input file name"),
+ OPT_STRING('o', "output", &output_name, "file",
+       "output file name"),
  OPT_INCR('v', "verbose", &verbose,
    "be more verbose (show build ids, etc)"),
  OPT_END()
@@ -243,6 +260,18 @@ int cmd_inject(int argc, const char **argv, const char *prefix __used)
  if (argc)
   usage_with_options(report_usage, options);

+ if (!strcmp(output_name, "-")) {
+  pipe_output = 1;
+  output = STDOUT_FILENO;
+ } else {
+  output = open(output_name, O_CREAT| O_WRONLY | O_TRUNC,
+        S_IRUSR | S_IWUSR);
+  if (output < 0) {
+   perror("failed to create output file");
+   exit(-1);
+  }
+ }
+
  if (symbol__init() < 0)
   return -1;

--
1.7.1
```

Subject: [PATCH 5/7] perf: teach perf inject to merge sched_stat_* and sched_switch events

Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:33 GMT

View Forum Message <> Reply to Message

You may want to know where and how long a task is sleeping. A callchain may be found in sched_switch and a time slice in stat_iowait, so I add handler in perf inject for merging this events.

My code saves sched_switch event for each process and when it meets stat_iowait, it reports the sched_switch event, because this event

contains a correct callchain. By another words it replaces all
stat_iowait events on proper sched_switch events.

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/builtin-inject.c | 87 ++
 1 files changed, 87 insertions(+), 0 deletions(-)

diff --git a/tools/perf/builtin-inject.c b/tools/perf/builtin-inject.c
index 8df8b71..d0af19d 100644
--- a/tools/perf/builtin-inject.c
+++ b/tools/perf/builtin-inject.c
@@ -12,6 +12,8 @@
 #include "util/debug.h"

 #include "util/parse-options.h"
+#include "util/trace-event.h"
+

 static char const *input_name = "-";
 static const char *output_name = "-";
@@ -20,6 +22,7 @@ static int output;
 static u64 bytes_written = 0;

 static bool inject_build_ids;
+static bool inject_sched_stat;

 static int perf_event__repipe_synth(union perf_event *event,
         struct perf_session *session __used)
@@ -179,6 +182,85 @@ repipe:
  return 0;
 }

+struct event_entry
+{
+ struct list_head list;
+ u32 pid;
+ union perf_event event[0];
+};
+
+static LIST_HEAD(samples);
+
+static int perf_event__sched_stat(union perf_event *event,
+   struct perf_sample *sample,
+   struct perf_evsel *evsel __used,
+   struct perf_session *session)
+{
+ int type;

```
+ struct event *e;
+ const char *evname = NULL;
+ uint32_t size;
+ struct event_entry *ent;
+ union perf_event *event_sw = NULL;
+ struct perf_sample sample_sw;
+ int sched_process_exit;
+
+ size = event->header.size;
+
+ type = trace_parse_common_type(sample->raw_data);
+ e = trace_find_event(type);
+ if (e)
+  evname = e->name;
+
+ sched_process_exit = !strcmp(evname, "sched_process_exit");
+
+ if (!strcmp(evname, "sched_switch") ||  sched_process_exit) {
+  list_for_each_entry(ent, &samples, list)
+   if (sample->pid == ent->pid)
+    break;
+
+  if (&ent->list != &samples) {
+   list_del(&ent->list);
+   free(ent);
+  }
+
+  if (sched_process_exit)
+   return 0;
+
+  ent = malloc(size + sizeof(struct event_entry));
+  ent->pid = sample->pid;
+  memcpy(&ent->event, event, size);
+  list_add(&ent->list, &samples);
+  return 0;
+
+ } else if (!strncmp(evname, "sched_stat_", 11)) {
+  u32 pid;
+
+  pid = raw_field_value(e, "pid", sample->raw_data);
+
+  list_for_each_entry(ent, &samples, list) {
+   if (pid == ent->pid)
+    break;
+  }
+
+  if (&ent->list == &samples) {
+   pr_debug("Could not find sched_switch for pid %u\n", pid);
```

```
+   return 0;
+  }
+
+  event_sw = &ent->event[0];
+  perf_session__parse_sample(session, event_sw, &sample_sw);
+  sample_sw.period = sample->period;
+  sample_sw.time = sample->time;
+  perf_session__change_sample(session, event_sw, &sample_sw);
+  perf_event__repipe(event_sw, &sample_sw, session);
+  return 0;
+ }
+
+ perf_event__repipe(event, sample, session);
+
+ return 0;
+}
 struct perf_event_ops inject_ops = {
  .sample  = perf_event__repipe_sample,
  .mmap  = perf_event__repipe,
@@ -214,6 +296,9 @@ static int __cmd_inject(void)
   inject_ops.mmap  = perf_event__repipe_mmap;
   inject_ops.fork  = perf_event__repipe_task;
   inject_ops.tracing_data = perf_event__repipe_tracing_data;
+ } else if (inject_sched_stat) {
+  inject_ops.sample = perf_event__sched_stat;
+  inject_ops.ordered_samples = true;
  }

  session = perf_session__new(input_name, O_RDONLY, false, true, &inject_ops);
@@ -241,6 +326,8 @@ static const char * const report_usage[] = {
 static const struct option options[] = {
  OPT_BOOLEAN('b', "build-ids", &inject_build_ids,
      "Inject build-ids into the output stream"),
+ OPT_BOOLEAN('s', "sched-stat", &inject_sched_stat,
+     "correct call-chains for shed-stat-*"),
  OPT_STRING('i', "input", &input_name, "file",
      "input file name"),
  OPT_STRING('o', "output", &output_name, "file",
--
1.7.1
```

Subject: [PATCH 6/7] perf: add scripts for profiling sleep times (v2)
Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:34 GMT
View Forum Message <> Reply to Message

E.g.:
 # perf script record -- sched:sched_stat_sleep -- ./foo

```
 # perf script report sched-stat
or
 # perf script record -- -e sched:sched_stat_sleep

v2: Add ability to record events for a defined process. It executes a small
bash script, then executes perf with filtering by pid and then a bash script
executes a target process.

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/scripts/python/bin/sched-stat-record |   65 ++++++++++++++++++++++++
 tools/perf/scripts/python/bin/sched-stat-report |    5 ++
 2 files changed, 70 insertions(+), 0 deletions(-)
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-record
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-report

diff --git a/tools/perf/scripts/python/bin/sched-stat-record
b/tools/perf/scripts/python/bin/sched-stat-record
new file mode 100644
index 0000000..5e0bd97
--- /dev/null
+++ b/tools/perf/scripts/python/bin/sched-stat-record
@@ -0,0 +1,65 @@
+#!/bin/bash
+# perf script record -- sched:sched_stat_[smth] -- CMD
+# perf script record -- -e sched:sched_stat_[smth]
+#
+set -o monitor
+
+usage()
+{
+ echo "Usage:"
+ echo " perf script record -- sched:sched_stat_[smth] -- CMD"
+ echo " perf script record -- [PERF OPTS] -e sched:sched_stat_[smth]"
+ exit 1;
+}
+
+declare -a opt
+declare -a cmd
+f=0;
+for i in "${@:2}"; do
+ if [ "$i" == "--" ]; then
+  f=1
+  continue
+ fi
+ if [ $f -eq 1 ]; then
+  cmd[${#cmd[*]}]="$i"
+ else
```

```
+  opt[${#opt[*]}]="$i"
+ fi
+done
+
+if [[ "${opt[@]}" != *sched_stat_* ]]; then
+ usage;
+fi
+
+if [ ${#cmd[@]} -eq 0 ]; then
+ if [ ${#opt[@]} -eq 0 ]; then
+  usage;
+ fi
+ exec perf record -agP \
+  -e sched:sched_switch \
+  --filter "prev_state == 1 || prev_state == 2" \
+  "${opt[@]}"
+fi
+
+if [ ${#opt[@]} -ne 1 ]; then
+ usage;
+fi
+
+# Wait until a target process is stopped.
+bash -c 'kill -STOP $$; exec "$@"' -- "${cmd[@]}" &
+pid=$!
+wait %1
+[ $? -eq 147 ] || exit 1;
+
+perf record -agP \
+  -e sched:sched_switch \
+  --filter "prev_pid == $pid && prev_state == 1 || prev_state == 2" \
+  -e sched:sched_process_exit -e "${opt[@]}" --filter "pid == $pid" &
+pid_perf=$!
+kill -CONT %1
+while :; do
+ wait %1
+ [ $? -eq 127 ] && break;
+done
+kill -INT %2
+wait %2
diff --git a/tools/perf/scripts/python/bin/sched-stat-report
b/tools/perf/scripts/python/bin/sched-stat-report
new file mode 100644
index 0000000..e5114ce
--- /dev/null
+++ b/tools/perf/scripts/python/bin/sched-stat-report
@@ -0,0 +1,5 @@
+#!/bin/bash
```

```
+# description: profiling sleep times
+perf inject -s -i perf.data -o perf.data.d || exit
+perf report -i perf.data.d || exit
+unlink perf.data.d
--
1.7.1
```

Subject: [PATCH 7/7] event: add tracepoint for accounting block time
Posted by Andrew Vagin on Mon, 28 Nov 2011 09:03:35 GMT
View Forum Message <> Reply to Message

This tracepoint shows how long a task is sleeping in uninterruptible state.

E.g.
It may show how long and where a mutex is waited.

Signed-off-by: Andrew Vagin <avagin@openvz.org>

```
 include/trace/events/sched.h |    7 +++++++
 kernel/sched_fair.c          |    2 ++
 2 files changed, 9 insertions(+), 0 deletions(-)

diff --git a/include/trace/events/sched.h b/include/trace/events/sched.h
index 959ff18..be077cf 100644
--- a/include/trace/events/sched.h
+++ b/include/trace/events/sched.h
@@ -331,6 +331,13 @@ DEFINE_EVENT(sched_stat_template, sched_stat_iowait,
       TP_ARGS(tsk, delay));

 /*
+ * Tracepoint for accounting block time (time the task is in uninterruptible).
+ */
+DEFINE_EVENT(sched_stat_template, sched_stat_block,
+      TP_PROTO(struct task_struct *tsk, u64 delay),
+      TP_ARGS(tsk, delay));
+
+/*
  * Tracepoint for accounting runtime (time the task is executing
  * on a CPU).
  */
diff --git a/kernel/sched_fair.c b/kernel/sched_fair.c
index 5c9e679..0d7b156 100644
--- a/kernel/sched_fair.c
+++ b/kernel/sched_fair.c
@@ -907,6 +907,8 @@ static void enqueue_sleeper(struct cfs_rq *cfs_rq, struct sched_entity
*se)
     trace_sched_stat_iowait(tsk, delta);
```

```
   }

+   trace_sched_stat_block(tsk, delta);
+
    /*
     * Blocking time is in units of nanosecs, so shift by
     * 20 to get a milliseconds-range estimation of the
--
1.7.1
```

Subject: Re: [PATCH 7/7] event: add tracepoint for accounting block time
Posted by Peter Zijlstra on Mon, 28 Nov 2011 11:42:11 GMT
View Forum Message <> Reply to Message

On Mon, 2011-11-28 at 12:03 +0300, Andrew Vagin wrote:
> This tracepoint shows how long a task is sleeping in uninterruptible state.
>
> E.g.
> It may show how long and where a mutex is waited.

Fair enough, makes one wonder how much it would take to make
account_scheduler_latency() go away..

> Signed-off-by: Andrew Vagin <avagin@openvz.org>
> ---
> include/trace/events/sched.h | 7 +++++++
> kernel/sched_fair.c | 2 ++
> 2 files changed, 9 insertions(+), 0 deletions(-)
>
> diff --git a/include/trace/events/sched.h b/include/trace/events/sched.h
> index 959ff18..be077cf 100644
> --- a/include/trace/events/sched.h
> +++ b/include/trace/events/sched.h
> @@ -331,6 +331,13 @@ DEFINE_EVENT(sched_stat_template, sched_stat_iowait,
>   TP_ARGS(tsk, delay));
>
> /*
> + * Tracepoint for accounting block time (time the task is in uninterruptible).
> + */
> +DEFINE_EVENT(sched_stat_template, sched_stat_block,
> +   TP_PROTO(struct task_struct *tsk, u64 delay),
> +   TP_ARGS(tsk, delay));
> +
> +/*
>   * Tracepoint for accounting runtime (time the task is executing
>   * on a CPU).
>   */

> diff --git a/kernel/sched_fair.c b/kernel/sched_fair.c
> index 5c9e679..0d7b156 100644
> --- a/kernel/sched_fair.c
> +++ b/kernel/sched_fair.c
> @@ -907,6 +907,8 @@ static void enqueue_sleeper(struct cfs_rq *cfs_rq, struct sched_entity *se)
> trace_sched_stat_iowait(tsk, delta);
> }
>
> + trace_sched_stat_block(tsk, delta);
> +
> /*
> * Blocking time is in units of nanosecs, so shift by
> * 20 to get a milliseconds-range estimation of the

Subject: Re: [PATCH 7/7] event: add tracepoint for accounting block time
Posted by Arjan van de Ven on Mon, 28 Nov 2011 14:02:27 GMT
View Forum Message <> Reply to Message

On 11/28/2011 3:42 AM, Peter Zijlstra wrote:
> On Mon, 2011-11-28 at 12:03 +0300, Andrew Vagin wrote:
>> This tracepoint shows how long a task is sleeping in uninterruptible state.
>>
>> E.g.
>> It may show how long and where a mutex is waited.
>
> Fair enough, makes one wonder how much it would take to make
> account_scheduler_latency() go away..

I would *love* to switch latencytop to using trace points / perf events.
But as long as this just means I get yelled at more for using "internal
ABIs" and the like at various occasions, I'm rather hesitant to turn
more tools into using perf.

(and we all know what the next steps to resolve this are, they just have
not happened yet; not all hope is lost)

Subject: Re: [PATCH 7/7] event: add tracepoint for accounting block time
Posted by Arnaldo Carvalho de M[2] on Mon, 28 Nov 2011 14:31:40 GMT
View Forum Message <> Reply to Message

Em Mon, Nov 28, 2011 at 06:02:27AM -0800, Arjan van de Ven escreveu:
> On 11/28/2011 3:42 AM, Peter Zijlstra wrote:
> > On Mon, 2011-11-28 at 12:03 +0300, Andrew Vagin wrote:
> >> This tracepoint shows how long a task is sleeping in uninterruptible state.

> >> E.g.
> >> It may show how long and where a mutex is waited.

> > Fair enough, makes one wonder how much it would take to make
> > account_scheduler_latency() go away..

> I would *love* to switch latencytop to using trace points / perf events.
> But as long as this just means I get yelled at more for using "internal
> ABIs" and the like at various occasions, I'm rather hesitant to turn
> more tools into using perf.

Have you read the the discussion with Robert Richter about that?

https://lkml.org/lkml/2011/11/24/237

perf_evlist is what you call perf_bundle and perf_evsel is what you call
perf_event in powertop.

That part of the API should be ok for wider use and is in fact exported
in the python binding.

I'm rearranging my tmp.perf/trace4 branch into perf/core to ask for
merging, that is a step in the direction of having the 'perf tool' class
added to what will become libperf.so.

I almost embarked into an attempt to make powertop use it, but there are
other stuff to do first, like making the tracepoint based tools already
in perf stop using long if-strcmp(evname, "foo")-call-process_foo_event
(we have IDs, a hash, better use that, etc).

The 'strace' will be Ingo & tglx's 'trace' tool using these changes,
should be done in a way that shows how to use the resulting abstractions
in libperf.so, that together with the other tools already in the kernel
(kmem, lock, etc).

> (and we all know what the next steps to resolve this are, they just have
> not happened yet; not all hope is lost)

Sure its not :-)

- Arnaldo

Subject: Re: [PATCH 7/7] event: add tracepoint for accounting block time
Posted by Arjan van de Ven on Mon, 28 Nov 2011 14:43:35 GMT
View Forum Message <> Reply to Message

>
> perf_evlist is what you call perf_bundle and perf_evsel is what you call
> perf_event in powertop.
>
> That part of the API should be ok for wider use and is in fact exported
> in the python binding.

I don't care about the snake language.

frankly all that's missing is a "safe" accessor library as Steve has
promised will appear.

that library really needs to be a proper shared library and not come
from/with the kernel package, so that distributions can independently
package it properly.
(and this obviously needs to at least look at the things that the
systemd guys pointed us at at the kernel summit)

I'm not interested if the code for a library is somewhere deep in the
kernel source code, not installed by default in distros (or tied in with
loads of other mess) and/or uses the kernel makefiles/etc like perf does.

Subject: Re: [PATCH 7/7] event: add tracepoint for accounting block time
Posted by Arnaldo Carvalho de M[2] on Mon, 28 Nov 2011 14:59:17 GMT
View Forum Message <> Reply to Message

Em Mon, Nov 28, 2011 at 06:43:35AM -0800, Arjan van de Ven escreveu:
> > perf_evlist is what you call perf_bundle and perf_evsel is what you call
> > perf_event in powertop.

> > That part of the API should be ok for wider use and is in fact exported
> > in the python binding.
>
> I don't care about the snake language.

I haven't suggested that you use any other language than C.

What I tried to say was that the way the python binding was written
provides an initial libperf.so, just that it is not exposed yet.

That is, the subset of C files there is (or should be) untangled from
all the rest of tools/perf.

> frankly all that's missing is a "safe" accessor library as Steve has
> promised will appear.

> that library really needs to be a proper shared library and not come

> from/with the kernel package, so that distributions can independently
> package it properly.

I can probably have something like:

```
[acme@emilia linux]$ make help | grep perf
  perf-tar-src-pkg    - Build perf-3.2.0-rc1.tar source tarball
  perf-targz-src-pkg  - Build perf-3.2.0-rc1.tar.gz source tarball
  perf-tarbz2-src-pkg - Build perf-3.2.0-rc1.tar.bz2 source tarball
  perf-tarxz-src-pkg  - Build perf-3.2.0-rc1.tar.xz source tarball
[acme@emilia linux]$
```

i.e.:

```
$ make libperf-tar-src.pkg
```

> (and this obviously needs to at least look at the things that the
> systemd guys pointed us at at the kernel summit)

Yes, I'll take that into account.

> I'm not interested if the code for a library is somewhere deep in the
> kernel source code, not installed by default in distros (or tied in with
> loads of other mess) and/or uses the kernel makefiles/etc like perf does.

Right, I'm working on that, don't know when it will get to the point
you'd consider using it, but I'll try to address your concerns in the
process.

- Arnaldo

Subject: Re: [PATCH 0/7] Profiling sleep times (v3)
Posted by Andrey Vagin on Tue, 06 Dec 2011 07:15:07 GMT
View Forum Message <> Reply to Message

Hello Arnaldo,

I found, that you commit patches for perf. Could you review this part?

Arun Sharma said, that the second versions of patches works ok for him.
(Arun is the first user of this functionality after me.)

Thanks.

On 11/28/2011 01:03 PM, Andrew Vagin wrote:
> Do you want to know where your code waits locks for a long time?
> Yes! It's for you. This feature helps you to find bottlenecks.

>
> It's not artificial task. Once one of my colleague was investigating a
> scalability problem. He pressed sysrq-t some times and tried to merge
> call-chains by hand. But perf can do that.
>
> Problem:
> The problem is that events sched_stat_* contain call-chains of
> non-target tasks.
> About month ago I sent series of patches:
>
> [PATCH 0/3] trace: add ability to collect call chains of non current task.
>
> Peter and Frederic explained me, that this solve isn't good and will be
> better to make it in userspace.
>
> Now it's in userspace. This series expands "perf inject" to be able to
> merge sched_switch events and sched_stat* events. sched_switch events
> contain correct call-chains and sched_stat contains a correct time
> slices.
>
> v2:
> * Removed all known issues. Now it works completely.
> * Improved usability of sched-stat scripts according with Arun's comments.
> v3: fixed accoding to comments from David Ahem
>
> Andrew Vagin (7):
>   perf: use event_name() to get an event name
>   perf: add ability to record event period
>   perf: add ability to change event according to sample (v2)
>   perf: teach "perf inject" to work with files
>   perf: teach perf inject to merge sched_stat_* and sched_switch events
>   perf: add scripts for profiling sleep times (v2)
>   event: add tracepoint for accounting block time
>
>   include/trace/events/sched.h | 7 ++
>   kernel/sched_fair.c | 2 +
>   tools/perf/builtin-inject.c | 120 ++++++++++++++++++++++++++-
>   tools/perf/builtin-record.c | 5 +
>   tools/perf/scripts/python/bin/sched-stat-record | 65 ++++++++++++
>   tools/perf/scripts/python/bin/sched-stat-report | 5 +
>   tools/perf/util/event.h | 2 +
>   tools/perf/util/evsel.c | 74 ++++++++++++++
>   tools/perf/util/header.c | 2 +-
>   tools/perf/util/session.h | 9 ++
>   10 files changed, 288 insertions(+), 3 deletions(-)
>   create mode 100644 tools/perf/scripts/python/bin/sched-stat-record
>   create mode 100644 tools/perf/scripts/python/bin/sched-stat-report
>

Subject: Re: [PATCH 0/7] Profiling sleep times (v3)
Posted by Ingo Molnar on Tue, 06 Dec 2011 08:30:44 GMT
View Forum Message <> Reply to Message

* Andrey Vagin <avagin@openvz.org> wrote:

> Hello Arnaldo,
>
> I found, that you commit patches for perf. Could you review this part?

To help out this effort i have applied your new
kernel/sched/fair.c event to patch the scheduler
tree (tip:sched/core).

Note, i have renamed it from sched_stat_block to
sched_stat_blocked - which is really the term we
use for such task states.

Thanks,

Ingo

Subject: Re: [PATCH 0/7] Profiling sleep times (v3)
Posted by Arnaldo Carvalho de M[2] on Tue, 06 Dec 2011 13:45:26 GMT
View Forum Message <> Reply to Message

Em Tue, Dec 06, 2011 at 11:15:07AM +0400, Andrey Vagin escreveu:
> Hello Arnaldo,
>
> I found, that you commit patches for perf. Could you review this part?
>
> Arun Sharma said, that the second versions of patches works ok for him.
> (Arun is the first user of this functionality after me.)

I'm going to check if the patches still apply and then review,

- Arnaldo

> Thanks.
>
> On 11/28/2011 01:03 PM, Andrew Vagin wrote:
> >Do you want to know where your code waits locks for a long time?
> >Yes! It's for you. This feature helps you to find bottlenecks.
> >
> >It's not artificial task. Once one of my colleague was investigating a
> >scalability problem. He pressed sysrq-t some times and tried to merge
> >call-chains by hand. But perf can do that.

> >
> >Problem:
> >The problem is that events sched_stat_* contain call-chains of
> >non-target tasks.
> >About month ago I sent series of patches:
> >
> >[PATCH 0/3] trace: add ability to collect call chains of non current task.
> >
> >Peter and Frederic explained me, that this solve isn't good and will be
> >better to make it in userspace.
> >
> >Now it's in userspace. This series expands "perf inject" to be able to
> >merge sched_switch events and sched_stat* events. sched_switch events
> >contain correct call-chains and sched_stat contains a correct time
> >slices.
> >
> >v2:
> >* Removed all known issues. Now it works completely.
> >* Improved usability of sched-stat scripts according with Arun's comments.
> >v3: fixed accoding to comments from David Ahem
> >
> >Andrew Vagin (7):
> > perf: use event_name() to get an event name
> > perf: add ability to record event period
> > perf: add ability to change event according to sample (v2)
> > perf: teach "perf inject" to work with files
> > perf: teach perf inject to merge sched_stat_* and sched_switch events
> > perf: add scripts for profiling sleep times (v2)
> > event: add tracepoint for accounting block time
> >
> > include/trace/events/sched.h | 7 ++
> > kernel/sched_fair.c | 2 +
> > tools/perf/builtin-inject.c | 120 ++++++++++++++++++++++++++++++-
> > tools/perf/builtin-record.c | 5 +
> > tools/perf/scripts/python/bin/sched-stat-record | 65 ++++++++++++
> > tools/perf/scripts/python/bin/sched-stat-report | 5 +
> > tools/perf/util/event.h | 2 +
> > tools/perf/util/evsel.c | 74 ++++++++++++++
> > tools/perf/util/header.c | 2 +-
> > tools/perf/util/session.h | 9 ++
> > 10 files changed, 288 insertions(+), 3 deletions(-)
> > create mode 100644 tools/perf/scripts/python/bin/sched-stat-record
> > create mode 100644 tools/perf/scripts/python/bin/sched-stat-report
> >

Subject: Re: [PATCH 2/7] perf: add ability to record event period

Em Mon, Nov 28, 2011 at 12:03:30PM +0300, Andrew Vagin escreveu:
> Signed-off-by: Andrew Vagin <avagin@openvz.org>
> ---
> tools/perf/builtin-record.c | 5 +++++
> 1 files changed, 5 insertions(+), 0 deletions(-)

This patch doesn't apply in current tip/perf/core as this logic was
moved to perf_evsel/perf_evlist, so that it can be used by other tools.

Please add the 'period' variable to struct perf_record_opts and the
sample_type or'ing to perf_evsel__config().

I suggest you use https://github.com/acmel/linux/commits/perf/core for
that.

- Arnaldo

> diff --git a/tools/perf/builtin-record.c b/tools/perf/builtin-record.c
> index 6ab58cc..e3b7fc4 100644
> --- a/tools/perf/builtin-record.c
> +++ b/tools/perf/builtin-record.c
> @@ -62,6 +62,7 @@ static bool   no_samples   = false;
> static bool   sample_address   = false;
> static bool   sample_time   = false;
> static bool   no_buildid   = false;
> +static bool   period   = false;
> static bool   no_buildid_cache = false;
> static struct perf_evlist *evsel_list;
>
> @@ -185,6 +186,9 @@ static void config_attr(struct perf_evsel *evsel, struct perf_evlist *evlist)
>   if (evlist->nr_entries > 1)
>   attr->sample_type |= PERF_SAMPLE_ID;
>
> + if (period)
> + attr->sample_type |= PERF_SAMPLE_PERIOD;
> +
>   /*
>   * We default some events to a 1 default interval. But keep
>   * it a weak assumption overridable by the user.
> @@ -803,6 +807,7 @@ const struct option record_options[] = {
>   OPT_BOOLEAN('d', "data", &sample_address,
>   "Sample addresses"),
>   OPT_BOOLEAN('T', "timestamp", &sample_time, "Sample timestamps"),
> + OPT_BOOLEAN('P', "period", &period, "Sample period"),
>   OPT_BOOLEAN('n', "no-samples", &no_samples,
>   "don't sample"),

> OPT_BOOLEAN('N', "no-buildid-cache", &no_buildid_cache,
> --
> 1.7.1

Subject: Re: [PATCH 3/7] perf: add ability to change event according to sample (v2)
Posted by Arnaldo Carvalho de M[2] on Tue, 06 Dec 2011 14:19:42 GMT
View Forum Message <> Reply to Message

Em Mon, Nov 28, 2011 at 12:03:31PM +0300, Andrew Vagin escreveu:
> It's opposition of perf_session__parse_sample.
>
> v2: fixed mistakes which David Arhen found

Ok, I'm taking this one, David, can I added an 'Acked-by: you"? Or even "reviewed-by:" ?

I'm just changing 'data' to 'sample', data is way to vague, I kept it for a while in the past just to reduce patch size, but this is something completely new, so better use 'sample'.

- Arnaldo

> Signed-off-by: Andrew Vagin <avagin@openvz.org>
> ---
> tools/perf/util/event.h | 2 +
> tools/perf/util/evsel.c | 74 ++
> tools/perf/util/session.h | 9 +++++
> 3 files changed, 85 insertions(+), 0 deletions(-)
>
> diff --git a/tools/perf/util/event.h b/tools/perf/util/event.h
> index 357a85b..0493101 100644
> --- a/tools/perf/util/event.h
> +++ b/tools/perf/util/event.h
> @@ -187,5 +187,7 @@ const char *perf_event__name(unsigned int id);
> int perf_event__parse_sample(const union perf_event *event, u64 type,
> 			 int sample_size, bool sample_id_all,
> 			 struct perf_sample *sample, bool swapped);
> +int perf_event__change_sample(union perf_event *event, u64 type,
> +			 const struct perf_sample *data, bool swapped);
>
> #endif /* __PERF_RECORD_H */
> diff --git a/tools/perf/util/evsel.c b/tools/perf/util/evsel.c
> index e426264..d697568 100644
> --- a/tools/perf/util/evsel.c
> +++ b/tools/perf/util/evsel.c
> @@ -494,3 +494,77 @@ int perf_event__parse_sample(const union perf_event *event, u64 type,

```
>
>   return 0;
>  }
> +
> +int perf_event__change_sample(union perf_event *event, u64 type,
> +        const struct perf_sample *data, bool swapped)
> +{
> + u64 *array;
> +
> + /*
> +  * used for cross-endian analysis. See git commit 65014ab3
> +  * for why this goofiness is needed.
> +  */
> + union {
> +  u64 val64;
> +  u32 val32[2];
> + } u;
> +
> + array = event->sample.array;
> +
> + if (type & PERF_SAMPLE_IP) {
> +  event->ip.ip = data->ip;
> +  array++;
> + }
> +
> + if (type & PERF_SAMPLE_TID) {
> +  u.val32[0] = data->pid;
> +  u.val32[1] = data->tid;
> +  if (swapped) {
> +   /* undo swap of u64, then swap on individual u32s */
> +   u.val32[0] = bswap_32(u.val32[0]);
> +   u.val32[1] = bswap_32(u.val32[1]);
> +   u.val64 = bswap_64(u.val64);
> +  }
> +
> +  *array = u.val64;
> +  array++;
> + }
> +
> + if (type & PERF_SAMPLE_TIME) {
> +  *array = data->time;
> +  array++;
> + }
> +
> + if (type & PERF_SAMPLE_ADDR) {
> +  *array = data->addr;
> +  array++;
> + }
```

```
> +
> + if (type & PERF_SAMPLE_ID) {
> +  *array = data->id;
> +  array++;
> + }
> +
> + if (type & PERF_SAMPLE_STREAM_ID) {
> +  *array = data->stream_id;
> +  array++;
> + }
> +
> + if (type & PERF_SAMPLE_CPU) {
> +  u.val32[0] = data->cpu;
> +  if (swapped) {
> +   /* undo swap of u64, then swap on individual u32s */
> +   u.val32[0] = bswap_32(u.val32[0]);
> +   u.val64 = bswap_64(u.val64);
> +  }
> +  *array = u.val64;
> +  array++;
> + }
> +
> + if (type & PERF_SAMPLE_PERIOD) {
> +  *array = data->period;
> +  array++;
> + }
> +
> + return 0;
> +}
> diff --git a/tools/perf/util/session.h b/tools/perf/util/session.h
> index 6e393c9..444f121 100644
> --- a/tools/perf/util/session.h
> +++ b/tools/perf/util/session.h
> @@ -167,6 +167,15 @@ static inline int perf_session__parse_sample(struct perf_session
*session,
>       session->header.needs_swap);
>  }
>
> +static inline int perf_session__change_sample(struct perf_session *session,
> +         union perf_event *event,
> +         const struct perf_sample *sample)
> +{
> + return perf_event__change_sample(event, session->sample_type,
> +     sample,
> +     session->header.needs_swap);
> +}
> +
>  struct perf_evsel *perf_session__find_first_evtype(struct perf_session *session,
```

> unsigned int type);
>
> --
> 1.7.1

Subject: Re: [PATCH 3/7] perf: add ability to change event according to sample (v2)
Posted by Arnaldo Carvalho de M[2] on Tue, 06 Dec 2011 14:24:46 GMT
View Forum Message <> Reply to Message

Em Tue, Dec 06, 2011 at 12:19:42PM -0200, Arnaldo Carvalho de Melo escreveu:
> Em Mon, Nov 28, 2011 at 12:03:31PM +0300, Andrew Vagin escreveu:
> > It's opposition of perf_session__parse_sample.
> >
> > v2: fixed mistakes which David Arhen found
>
> Ok, I'm taking this one, David, can I added an 'Acked-by: you"? Or even
> "reviewed-by:" ?
>
> I'm just changing 'data' to 'sample', data is way to vague, I kept it
> for a while in the past just to reduce patch size, but this is something
> completely new, so better use 'sample'.

Also I'm changing it from perf_event__change_sample() to
perf_event__synthesize_sample() as that is what is happening, you're not
taking some perf_sample and changing it just a bit, but synthesizing it
completely.

Since synthesize is used elsewhere when we synthesize PERF_RECORD_MMAP,
etc for existing threads, I'll use it here too.

> - Arnaldo
>
> > Signed-off-by: Andrew Vagin <avagin@openvz.org>
> > ---
> > tools/perf/util/event.h | 2 +
> > tools/perf/util/evsel.c | 74 ++
> > tools/perf/util/session.h | 9 +++++
> > 3 files changed, 85 insertions(+), 0 deletions(-)
> >
> > diff --git a/tools/perf/util/event.h b/tools/perf/util/event.h
> > index 357a85b..0493101 100644
> > --- a/tools/perf/util/event.h
> > +++ b/tools/perf/util/event.h
> > @@ -187,5 +187,7 @@ const char *perf_event__name(unsigned int id);
> > int perf_event__parse_sample(const union perf_event *event, u64 type,
> > 			 int sample_size, bool sample_id_all,
> > 			 struct perf_sample *sample, bool swapped);

```
> > +int perf_event__change_sample(union perf_event *event, u64 type,
> > +        const struct perf_sample *data, bool swapped);
> >
> >  #endif /* __PERF_RECORD_H */
> > diff --git a/tools/perf/util/evsel.c b/tools/perf/util/evsel.c
> > index e426264..d697568 100644
> > --- a/tools/perf/util/evsel.c
> > +++ b/tools/perf/util/evsel.c
> > @@ -494,3 +494,77 @@ int perf_event__parse_sample(const union perf_event *event, u64
type,
> >
>   return 0;
>  }
> > +
> > +int perf_event__change_sample(union perf_event *event, u64 type,
> > +        const struct perf_sample *data, bool swapped)
> > +{
> > + u64 *array;
> > +
> > + /*
> > +  * used for cross-endian analysis. See git commit 65014ab3
> > +  * for why this goofiness is needed.
> > +  */
> > + union {
> > +  u64 val64;
> > +  u32 val32[2];
> > + } u;
> > +
> > + array = event->sample.array;
> > +
> > + if (type & PERF_SAMPLE_IP) {
> > +  event->ip.ip = data->ip;
> > +  array++;
> > + }
> > +
> > + if (type & PERF_SAMPLE_TID) {
> > +  u.val32[0] = data->pid;
> > +  u.val32[1] = data->tid;
> > +  if (swapped) {
> > +   /* undo swap of u64, then swap on individual u32s */
> > +   u.val32[0] = bswap_32(u.val32[0]);
> > +   u.val32[1] = bswap_32(u.val32[1]);
> > +   u.val64 = bswap_64(u.val64);
> > +  }
> > +
> > +  *array = u.val64;
> > +  array++;
> > + }
```

```
> > +
> > + if (type & PERF_SAMPLE_TIME) {
> > +  *array = data->time;
> > +  array++;
> > + }
> > +
> > + if (type & PERF_SAMPLE_ADDR) {
> > +  *array = data->addr;
> > +  array++;
> > + }
> > +
> > + if (type & PERF_SAMPLE_ID) {
> > +  *array = data->id;
> > +  array++;
> > + }
> > +
> > + if (type & PERF_SAMPLE_STREAM_ID) {
> > +  *array = data->stream_id;
> > +  array++;
> > + }
> > +
> > + if (type & PERF_SAMPLE_CPU) {
> > +  u.val32[0] = data->cpu;
> > +  if (swapped) {
> > +   /* undo swap of u64, then swap on individual u32s */
> > +   u.val32[0] = bswap_32(u.val32[0]);
> > +   u.val64 = bswap_64(u.val64);
> > +  }
> > +  *array = u.val64;
> > +  array++;
> > + }
> > +
> > + if (type & PERF_SAMPLE_PERIOD) {
> > +  *array = data->period;
> > +  array++;
> > + }
> > +
> > + return 0;
> > +}
> > diff --git a/tools/perf/util/session.h b/tools/perf/util/session.h
> > index 6e393c9..444f121 100644
> > --- a/tools/perf/util/session.h
> > +++ b/tools/perf/util/session.h
> > @@ -167,6 +167,15 @@ static inline int perf_session__parse_sample(struct perf_session
*session,
> >       session->header.needs_swap);
> >  }
> >
```

> > +static inline int perf_session__change_sample(struct perf_session *session,
> > + union perf_event *event,
> > + const struct perf_sample *sample)
> > +{
> > + return perf_event__change_sample(event, session->sample_type,
> > + sample,
> > + session->header.needs_swap);
> > +}
> > +
> > struct perf_evsel *perf_session__find_first_evtype(struct perf_session *session,
> > unsigned int type);
> >
> > --
> > 1.7.1

Subject: Re: [PATCH 3/7] perf: add ability to change event according to sample (v2)
Posted by David Ahern on Tue, 06 Dec 2011 14:57:38 GMT
View Forum Message <> Reply to Message

On 12/06/2011 07:19 AM, Arnaldo Carvalho de Melo wrote:
> Em Mon, Nov 28, 2011 at 12:03:31PM +0300, Andrew Vagin escreveu:
>> It's opposition of perf_session__parse_sample.
>>
>> v2: fixed mistakes which David Arhen found
>
> Ok, I'm taking this one, David, can I added an 'Acked-by: you"? Or even
> "reviewed-by:" ?
>
> I'm just changing 'data' to 'sample', data is way to vague, I kept it
> for a while in the past just to reduce patch size, but this is something
> completely new, so better use 'sample'.
>
> - Arnaldo
>
>> Signed-off-by: Andrew Vagin <avagin@openvz.org>
>> ---
>> tools/perf/util/event.h | 2 +
>> tools/perf/util/evsel.c | 74 ++
>> tools/perf/util/session.h | 9 +++++
>> 3 files changed, 85 insertions(+), 0 deletions(-)
>>
>> diff --git a/tools/perf/util/event.h b/tools/perf/util/event.h
>> index 357a85b..0493101 100644
>> --- a/tools/perf/util/event.h
>> +++ b/tools/perf/util/event.h
>> @@ -187,5 +187,7 @@ const char *perf_event__name(unsigned int id);
>> int perf_event__parse_sample(const union perf_event *event, u64 type,

```
>>          int sample_size, bool sample_id_all,
>>          struct perf_sample *sample, bool swapped);
>> +int perf_event__change_sample(union perf_event *event, u64 type,
>> +        const struct perf_sample *data, bool swapped);
>>
>>  #endif /* __PERF_RECORD_H */
>> diff --git a/tools/perf/util/evsel.c b/tools/perf/util/evsel.c
>> index e426264..d697568 100644
>> --- a/tools/perf/util/evsel.c
>> +++ b/tools/perf/util/evsel.c
>> @@ -494,3 +494,77 @@ int perf_event__parse_sample(const union perf_event *event, u64
type,
>>
>>   return 0;
>>  }
>> +
>> +int perf_event__change_sample(union perf_event *event, u64 type,
>> +        const struct perf_sample *data, bool swapped)
>> +{
>> + u64 *array;
>> +
>> + /*
>> +  * used for cross-endian analysis. See git commit 65014ab3
>> +  * for why this goofiness is needed.
>> +  */
>> + union {
>> +  u64 val64;
>> +  u32 val32[2];
>> + } u;
>> +
>> + array = event->sample.array;
>> +
>> + if (type & PERF_SAMPLE_IP) {
>> +  event->ip.ip = data->ip;
>> +  array++;
>> + }
>> +
>> + if (type & PERF_SAMPLE_TID) {
>> +  u.val32[0] = data->pid;
>> +  u.val32[1] = data->tid;
>> +  if (swapped) {
>> +   /* undo swap of u64, then swap on individual u32s */
```

Comment is from the parse sample code; this is the inverse of that code
so the comment needs to be updated here.

```
>> +   u.val32[0] = bswap_32(u.val32[0]);
>> +   u.val32[1] = bswap_32(u.val32[1]);
```

```
>> +   u.val64 = bswap_64(u.val64);
>> +  }
>> +
>> +  *array = u.val64;
>> +  array++;
>> + }
>> +
>> + if (type & PERF_SAMPLE_TIME) {
>> +  *array = data->time;
>> +  array++;
>> + }
>> +
>> + if (type & PERF_SAMPLE_ADDR) {
>> +  *array = data->addr;
>> +  array++;
>> + }
>> +
>> + if (type & PERF_SAMPLE_ID) {
>> +  *array = data->id;
>> +  array++;
>> + }
>> +
>> + if (type & PERF_SAMPLE_STREAM_ID) {
>> +  *array = data->stream_id;
>> +  array++;
>> + }
>> +
>> + if (type & PERF_SAMPLE_CPU) {
>> +  u.val32[0] = data->cpu;
>> +  if (swapped) {
>> +   /* undo swap of u64, then swap on individual u32s */
```

Ditto on that comment.

I did not test it, but the logic is in the inverse of parse_sample so it should be correct.

Arnaldo: you could use remove the 2 comment lines on the commit since there are no logic impacts.

Reviewed-by: David Ahern <dsahern@gmail.com>

David

Subject: Re: [PATCH 3/7] perf: add ability to change event according to sample (v2)
Posted by Arnaldo Carvalho de M[2] on Tue, 06 Dec 2011 15:06:23 GMT

Em Tue, Dec 06, 2011 at 07:57:38AM -0700, David Ahern escreveu:
> Ditto on that comment.
>
> I did not test it, but the logic is in the inverse of parse_sample so it
> should be correct.
>
> Arnaldo: you could use remove the 2 comment lines on the commit since
> there are no logic impacts.
>
> Reviewed-by: David Ahern <dsahern@gmail.com>

Thanks, will do that,

- Arnaldo

Subject: Re: [PATCH 0/7] Profiling sleep times (v3)
Posted by Arun Sharma on Wed, 07 Dec 2011 20:33:48 GMT

On 12/5/11 11:15 PM, Andrey Vagin wrote:

> Arun Sharma said, that the second versions of patches works ok for him.
> (Arun is the first user of this functionality after me.)

Yes - Andrey's patches (v2) have been functional for me when used via:

perf record -agP -e sched:sched_switch --filter "prev_state == 1 ||
prev_state == 2" -e sched:sched_stat_sleep,sched:sched_stat_iowait --
sleep 3
mv perf.data{,.old}; perf inject -s -i perf.data.old -o perf.data
perf report --stdio -g graph --sort pid -C command-name

There are two major issues though:

* The above command lines cause us to collect way too much data and perf
can't keep up on a busy server

Eg:

Warning:
Processed 55182 events and lost 13 chunks!

Check IO/CPU overload!

* Requires root access

I suspect there is a way to stash the delay information available at enqueue_sleeper() into the task_struct (or some place else) and make it available at the sched:sched_switch tracepoint. This would solve both of the problems above, but I don't have a patch yet.

-Arun
