Subject: Re: [PATCH 3/6] perf: add ability to record event period
Posted by avagin on Tue, 20 Dec 2011 08:07:41 GMT
View Forum Message <> Reply to Message

On 12/20/2011 12:58 AM, Arnaldo Carvalho de Melo wrote:
> Em Fri, Dec 16, 2011 at 11:13:07AM +0400, Andrew Vagin escreveu:
>> Hi Arnaldo,
>>
>> Could you review and commit this patch. It's quite common
>> functionality, which allow to get events more effectively and to
>> avoid losing events.
>>
>> All other patches may be postponed, because Arun Sharma wants to
>> suggest your version of "Profiling sleep times".
> It would help if you provided a more detailed patch description, this
> one came with just a title :-\
Look at the comment below. In it I try describe why we need this
functionality.

The problem is that when SAMPLE_PERIOD is not set, kernel generates a
number of samples in proportion to an event's period. Number of these
samples may be too big and a kernel throttles all samples above a
defined limit.

E.g.: I want to trace when a process sleeps. I created a process, which
sleeps for 1ms and for 4ms. perf got 100 events in both cases.

```
swapper     0 [000]  1141.371830: sched_stat_sleep: comm=foo pid=1801 delay=1386750 [ns]
swapper     0 [000]  1141.369444: sched_stat_sleep: comm=foo pid=1801 delay=4499585 [ns]
```

In the first case a kernel want to send 4499585 events and
in the second case it wants to send 1386750 events.
perf-reports shows that process sleeps in both places equal time.

Instead of this we can get only one sample with an attribute period. As
result we have less data transferring between kernel and user-space and we
avoid throttling of samples.

The patch "events: Don't divide events if it has field period" added a
kernel part of this functionality.
>
> You started to ellaborate above when stating that "which allows to get
> events more effectively", could you please expand on that and mention
> that it will be used by the following patches that will implement
> feature X, etc?
>
> I get that Arun is in agreement, everything seems OK, but we need to do
> a better job on describing why we add code, the context we have now from

```
> all these discussions will be mostly lost, say, 5 years from now when we
> try to figure out why something was done in some way,
>
> Thanks,
>
> - Arnaldo
>
>> Thanks.
>>
>> On 12/07/2011 05:55 PM, Andrew Vagin wrote:
>>> Signed-off-by: Andrew Vagin<avagin@openvz.org>
>>> ---
>>>   tools/perf/builtin-record.c |    1 +
>>>   tools/perf/perf.h           |    1 +
>>>   tools/perf/util/evsel.c     |    3 +++
>>>   3 files changed, 5 insertions(+), 0 deletions(-)
>>>
>>> diff --git a/tools/perf/builtin-record.c b/tools/perf/builtin-record.c
>>> index 766fa0a..f8fd14f 100644
>>> --- a/tools/perf/builtin-record.c
>>> +++ b/tools/perf/builtin-record.c
>>> @@ -700,6 +700,7 @@ const struct option record_options[] = {
>>>    OPT_BOOLEAN('d', "data",&record.opts.sample_address,
>>>        "Sample addresses"),
>>>    OPT_BOOLEAN('T', "timestamp",&record.opts.sample_time, "Sample timestamps"),
>>> + OPT_BOOLEAN('P', "period",&record.opts.period, "Sample period"),
>>>    OPT_BOOLEAN('n', "no-samples",&record.opts.no_samples,
>>>        "don't sample"),
>>>    OPT_BOOLEAN('N', "no-buildid-cache",&record.no_buildid_cache,
>>> diff --git a/tools/perf/perf.h b/tools/perf/perf.h
>>> index ea804f5..64f8bee 100644
>>> --- a/tools/perf/perf.h
>>> +++ b/tools/perf/perf.h
>>> @@ -200,6 +200,7 @@ struct perf_record_opts {
>>>    bool     sample_time;
>>>    bool     sample_id_all_avail;
>>>    bool     system_wide;
>>> + bool     period;
>>>    unsigned int freq;
>>>    unsigned int mmap_pages;
>>>    unsigned int user_freq;
>>> diff --git a/tools/perf/util/evsel.c b/tools/perf/util/evsel.c
>>> index e2d1b22..8550018 100644
>>> --- a/tools/perf/util/evsel.c
>>> +++ b/tools/perf/util/evsel.c
>>> @@ -108,6 +108,9 @@ void perf_evsel__config(struct perf_evsel *evsel, struct
perf_record_opts *opts)
>>>    if (opts->system_wide)
>>>      attr->sample_type |= PERF_SAMPLE_CPU;
>>>
>>> + if (opts->period)
>>> +  attr->sample_type |= PERF_SAMPLE_PERIOD;
>>> +
>>>     if (opts->sample_id_all_avail&&
>>>     (opts->sample_time || opts->system_wide ||
>>>          !opts->no_inherit || opts->cpu_list))
```