LTTng-UST - Bug #1131

enable-event doesn't match class unless using wildcard

10/04/2017 05:50 AM - Tom Deseyn

Status:	Resolved	**Start date:**	10/04/2017
Priority:	Normal	**Due date:**	
Assignee:	Mathieu Desnoyers	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

I am tracing a .NET Core application. The runtime uses lttng ust and emits events named 'DotNETRuntime:*'.

I am interested in the 'DotNETRuntime:GCAllocationTick_V3' event.
When I 'enable-event DotNETRuntime:GCAllocationTick_V3' it is not part of the trace.
When I enable the more general 'DotNETRuntime:*', 'DotNETRuntime:GCAllocationTick_V3' is in the trace.
When I enable event 'DotNETRuntime:RuntimeInformationStart' (without enabling the wildcard),
'DotNETRuntime:RuntimeInformationStart' in the trace.

I don't understand why tracing works for DotNETRuntime:RuntimeInformationStart but not for DotNETRuntime:GCAllocationTick_V3.

Below you see an attempt to trace 'DotNETRuntime:GCAllocationTick_V3' followed by an attempt when using 'DotNETRuntime:*'.
The dotnet application is started after 'lttng start', and terminates before executing 'lttng stop'.

Attempt 'DotNETRuntime:GCAllocationTick_V3':
```
$ lttng create
Session auto-20171004-111626 created.
Traces will be written in /home/tmds/lttng-traces/auto-20171004-111626
$ lttng enable-event --userspace DotNETRuntime:GCAllocationTick_V3
UST event DotNETRuntime:GCAllocationTick_V3 created in channel channel0
$ lttng start
Tracing started for session auto-20171004-111626
$ lttng list auto-20171004-111626
Tracing session auto-20171004-111626: [active]
Trace path: /home/tmds/lttng-traces/auto-20171004-111626

=== Domain: UST global ===

Buffer type: per UID

Channels:
-------------
- channel0: [enabled]

Attributes:
      overwrite mode: 0
      subbuffers size: 131072
      number of subbuffers: 4
      switch timer interval: 0
      read timer interval: 0
      trace file count: 0
      trace file size (bytes): 0
      discarded events: 0
      lost packets: 0
      output: mmap()

Events:
      DotNETRuntime:GCAllocationTick_V3 (type: tracepoint) [enabled]

$ lttng stop
Waiting for data availability.
Tracing stopped for session auto-20171004-111626
```

```
$ lttng destroy
Session auto-20171004-111626 destroyed
$ babeltrace /home/tmds/lttng-traces/auto-20171004-111626
<nothing>
```

Attempt 'DotNETRuntime:*':
```
$ lttng create
Session auto-20171004-112647 created.
Traces will be written in /home/tmds/lttng-traces/auto-20171004-112647
$ lttng enable-event --userspace DotNETRuntime:*
UST event DotNETRuntime:* created in channel channel0
$ lttng start
Tracing started for session auto-20171004-112647
$ lttng list auto-20171004-112647
Tracing session auto-20171004-112647: [active]
Trace path: /home/tmds/lttng-traces/auto-20171004-112647

=== Domain: UST global ===

Buffer type: per UID

Channels:
-------------
- channel0: [enabled]

Attributes:
      overwrite mode: 0
      subbuffers size: 131072
      number of subbuffers: 4
      switch timer interval: 0
      read timer interval: 0
      trace file count: 0
      trace file size (bytes): 0
      discarded events: 0
      lost packets: 0
      output: mmap()

Events:
      DotNETRuntime:* (type: tracepoint) [enabled]

$ lttng stop
Waiting for data availability.
[warning] 7305 events discarded, please refer to the documentation on channel configuration.
Tracing stopped for session auto-20171004-112647
$ lttng destroy
Session auto-20171004-112647 destroyed
$ babeltrace /home/tmds/lttng-traces/auto-20171004-112647 | grep DotNETRuntime:GCAllocationTick_V3
[11:27:16.666713244] (+0.000006752) puter DotNETRuntime:GCAllocationTick_V3: { cpu_id = 4 }, { AllocationAmount = 108352,
AllocationKind = 0, ClrInstanceID = 0, AllocationAmount64 = 108352, TypeID = 140150218098888, TypeName = "System.Int32[]",
HeapIndex = 0, Address = 140149547677912 }
[11:27:16.669621317] (+0.000006642) puter DotNETRuntime:GCAllocationTick_V3: { cpu_id = 4 }, { AllocationAmount = 104032,
AllocationKind = 0, ClrInstanceID = 0, AllocationAmount64 = 104032, TypeID = 140150218406064, TypeName = "System.Object",
HeapIndex = 0, Address = 140149547770848 }
```

System info:
```
$ lttng --version
lttng (LTTng Trace Control) 2.8.1 - Isseki Nicho
$ uname -a
Linux puter 4.11.8-200.fc25.x86_64 #1 SMP Thu Jun 29 16:13:56 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
$ dotnet --version
2.0.0
```

History

#1 - 10/04/2017 08:27 AM - Tom Deseyn

To build a dotnet program to reproduce this with:

1. Install dotnet 2.0 from www.dot.net/core.
2. Create a console application
$ cd /tmp
$ dotnet new console --output console
$ cd console
3. Edit the Program.cs so it is allocaty
static void Main(string[] args)   {
for (int i = 0; i < 1_000_000; i++)   {
new object();
}
}
4. Build the program
$ dotnet build

To execute the program:

1. Enable lttng logging:
$ export COMPlus_EnableEventLog=1
2. Execute the program
$ dotnet bin/Debug/netcoreapp2.0/console.dll

#2 - 10/04/2017 02:21 PM - Mathieu Desnoyers

Weirdly it works if I explicitly enable V2 and V3 of that event:

lttng create
lttng enable-event -u 'DotNETRuntime:GCAllocationTick_V2'
lttng enable-event -u 'DotNETRuntime:GCAllocationTick_V3'
lttng start
[...] -> has events.

It starts to look like the "V3" event would be using the tracepoint declaration "enabled state" of the V2 event.

#3 - 10/04/2017 02:54 PM - Mathieu Desnoyers

Can you double-check that when FireEtXplatGCAllocationTick_V3 is invoked, it does indeed reach the do_tracepoint() ? Perhaps there are some arguments that are not prepared before calling FireEtXplatGCAllocationTick_V3, which might depend on having V2 of the event enabled as well.

#4 - 10/04/2017 03:13 PM - Tom Deseyn

Mathieu, looks like you found the bug!

gc.cpp (https://raw.githubusercontent.com/dotnet/coreclr/master/src/gc/gc.cpp) has some code:

```
if (EventEnabledGCAllocationTick_V2())
{
                fire_etw_allocation_event (etw_allocation_running_amount[etw_allocation_index], gen_number, ac
ontext->alloc_ptr);
            }
```

I will create a bug in the coreclr repo for this. We can close this issue. Thank you very much for looking into this!!

#5 - 10/04/2017 03:24 PM - Mathieu Desnoyers

Possible hint (untested):

```
src/gc/gc.cpp:
int gc_heap::try_allocate_more_space (alloc_context* acontext, size_t size,
                                      int gen_number)

#if defined(FEATURE_EVENT_TRACE)
            if (EventEnabledGCAllocationTick_V2())
            {
                fire_etw_allocation_event (etw_allocation_running_amount[etw_allocation_index], gen_number, ac
ontext->alloc_ptr);
            }
#endif //FEATURE_EVENT_TRACE
```

That would be an hard-coded check for the "V2" enable state (which ends up checking the tracepoint enable state for V2 of the event), but which ends

up invoking the V3 tracepoint:

src/gc/gcee.cpp:

```
void gc_heap::fire_etw_allocation_event (size_t allocation_amount, int gen_number, uint8_t* object_address)

    if (typeId != nullptr)
    {
        FireEtwGCAllocationTick_V3((uint32_t)allocation_amount,
                                   ((gen_number == 0) ? ETW::GCLog::ETW_GC_INFO::AllocationSmall : ETW::GCLog:
:ETW_GC_INFO::AllocationLarge),
                                   GetClrInstanceId(),
                                   allocation_amount,
                                   typeId,
                                   name,
                                   heap_number,
                                   object_address
                                   );
    }
```

#6 - 10/04/2017 03:26 PM - Mathieu Desnoyers

- Status changed from New to Resolved

- Assignee set to Mathieu Desnoyers

Just saw that your answer crossed mine. Good! I'll close the ticket then.

It would be a good thing to grep for similar usage patterns across the coreclr code base, just in case. So far grep for "EventEnabled.*V" yields no match, so it should be OK.

Thanks,

Mathieu