OsmoTRX - Feature #3962

osmo-trx: Use gcc attribute no_sanitize to run fine with ASan enabled

04/28/2019 08:30 AM - pespin

Status:	New	**Start date:**	04/28/2019
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:			
Target version:			
Spec Reference:			

Description

So far we don't enable ASan during osmo-trx build because it adds too much overhead and makes it wrong incorrectly (too slow).

zecke informed that there's a gcc compile-time attribute to flag specific functions to be not covered by ASan. So we can run perf on osmo-trx, see the most CPU intensive functions and disable ASan on those.

https://gcc.gnu.org/onlinedocs/gcc/Common-Function-Attributes.html#Common-Function-Attributes

History

#1 - 04/28/2019 09:35 AM - pespin

Since we usually compile with "-fsanitize=address -fsanitize=undefined" in osmocom projects, we should probably use flags: "no_sanitize_address no_sanitize_undefined"

I need to check what happens if compiled with a compiler which doesn't support those flags. Do we need to put it into some OSMO_SANITIZE_AVOID define which is set to empty based on compiler support?

#2 - 08/26/2019 01:02 PM - pespin

On my current laptop (i7) I can run osmo-trx with asan enabled without any issue. Using linux perf on a running osmo-trx, it can be seen ASan doesn't take a big amount of CPU:

```
0.66%     0.66%  TxUpper0       libasan.so.5.0.0           [.] __asan::Allocator::Allocate
0.39%     0.39%  RxUpper0       libasan.so.5.0.0           [.] __asan::Allocator::Allocate
0.32%     0.32%  RxLower        libasan.so.5.0.0           [.] __asan::Allocator::Allocate
0.31%     0.31%  RxUpper0       libasan.so.5.0.0           [.] __interceptor_memset.part.0
0.26%     0.26%  TxLower        libasan.so.5.0.0           [.] __asan::Allocator::Allocate
0.21%     0.21%  RxUpper0       libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::Put
0.16%     0.16%  TxUpper0       libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::Put
0.16%     0.16%  TxLower        libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::Put
0.15%     0.15%  RxUpper0       libasan.so.5.0.0           [.] __sanitizer::BufferedStackTrace::FastUnwindStac
k
0.11%     0.11%  RxUpper0       libasan.so.5.0.0           [.] __asan::QuarantineCallback::Recycle
0.11%     0.11%  TxUpper0       libasan.so.5.0.0           [.] __sanitizer::BufferedStackTrace::FastUnwindStac
k
0.10%     0.10%  TxUpper0       libasan.so.5.0.0           [.] __sanitizer::CombinedAllocator<__sanitizer::Siz
eClassAllocator64<__asan::AP64>
0.09%     0.09%  RxUpper0       libasan.so.5.0.0           [.] __interceptor_memset
0.09%     0.09%  TxUpper0       libasan.so.5.0.0           [.] __sanitizer::Quarantine<__asan::QuarantineCallb
ack, __asan::AsanChunk>::DoRecycle
0.09%     0.09%  TxUpper0       libasan.so.5.0.0           [.] __asan::QuarantineCallback::Recycle
0.08%     0.08%  TxLower        libasan.so.5.0.0           [.] __sanitizer::BufferedStackTrace::FastUnwindStac
k
0.08%     0.08%  RxUpper0       libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::find
0.08%     0.08%  RxLower        libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::Put
0.07%     0.07%  RxUpper0       libasan.so.5.0.0           [.] __sanitizer::Quarantine<__asan::QuarantineCallb
ack, __asan::AsanChunk>::DoRecycle
```

```
0.07%     0.07%  TxLower        libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::find
0.07%     0.07%  RxLower        libasan.so.5.0.0           [.] __sanitizer::mem_is_zero
0.06%     0.06%  TxLower        libasan.so.5.0.0           [.] __asan_region_is_poisoned
0.06%     0.06%  TxLower        libasan.so.5.0.0           [.] __interceptor_memcpy.part.0
0.06%     0.06%  TxUpper0       libasan.so.5.0.0           [.] __interceptor_memcpy.part.0
0.06%     0.06%  RxLower        libasan.so.5.0.0           [.] __sanitizer::BufferedStackTrace::FastUnwindStac
k
0.06%     0.06%  RxUpper0       libasan.so.5.0.0           [.] __sanitizer::CombinedAllocator<__sanitizer::Siz
eClassAllocator64<__asan::AP64>,__sanitizer::SizeClassAllocatorLocalCache<__sanitizer::SizeClassAllocato
0.06%     0.06%  TxUpper0       libasan.so.5.0.0           [.] __sanitizer::StackDepotBase<__sanitizer::StackD
epotNode, 1, 20>::find
0.06%     0.06%  RxUpper0       libasan.so.5.0.0           [.] __asan::Allocator::QuarantineChunk
0.06%     0.06%  TxLower        libasan.so.5.0.0           [.] __asan::Allocator::QuarantineChunk
0.06%     0.06%  TxLower        libasan.so.5.0.0           [.] __sanitizer::CombinedAllocator<__sanitizer::Siz
eClassAllocator64<__asan::AP64>,sanitizer::SizeClassAllocatorLocalCache<__sanitizer::SizeClassAllocato
0.05%     0.05%  RxLower        libasan.so.5.0.0           [.] __asan_region_is_poisoned
...
```

So roughly around 5% of CPU time per thread.

#3 - 08/26/2019 01:25 PM - pespin

Most noticeable resource user in Asan (1.14% of CPU) is here:

```
-    1.14%     0.55%  TxUpper0       libasan.so.5.0.0           [.] __asan::Allocator::Allocate

     0.59% __asan::Allocator::Allocate

   - 0.55% TxUpperLoopAdapter

        Transceiver::driveTxPriorityQueue

-    1.14%     0.00%  TxUpper0       libasan.so.5.0.0           [.] __asan::asan_memalign

     __asan::asan_memalign

     __asan::Allocator::Allocate
```

So some memory allocation under Transceiver::driveTxPriorityQueue, probably this part of the code:

```
  BitVector newBurst(burstLen);
  BitVector::iterator itr = newBurst.begin();
  uint8_t *bufferItr = dl->soft_bits;
  while (itr < newBurst.end())
    *itr++ = *bufferItr++;

  GSM::Time currTime = GSM::Time(fn, dl->common.tn);

  addRadioVector(chan, newBurst, dl->tx_att, currTime);
```

That's probably some code we want to optimize and simply pass a pointer to dl->soft_bits (or using mempcy) to get rid of all that heap allocation + loop copying.

#4 - 08/26/2019 02:04 PM - pespin

- Priority changed from Normal to Low

Switching priority to low.