Enduro/X Core - Bug #775

Concurrent (from different threads) debug init and tpinit from another thread might cause deadlock at binary startup

05/01/2022 09:03 PM - Madars

Status:	Closed	**Start date:**	05/01/2022
Priority:	Normal (Code 4)	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Thread 2 (Thread 0x7f4c203ae700 (LWP 242533)):
#0 __lll_lock_wait (futex=futex@entry=0x7f4c2065b6a0 <M_dbglock>, private=0) at lowlevellock.c:52
#1 0x00007f4c21d2f0a3 in __GI___pthread_mutex_lock (mutex=0x7f4c2065b6a0 <M_dbglock>) at ../nptl/pthread_mutex_lock.c:80
#2 0x00007f4c205fbf7a in ndrx_dbg_intlock_unset () at /home/user1/projects/endurox/libnstd/ndebug.c:219
#3 0x00007f4c20607358 in ndrx_cconfig_load () at /home/user1/projects/endurox/libnstd/cconfig.c:500
#4 0x00007f4c206f4a25 in ndrx_load_common_env () at /home/user1/projects/endurox/libatmi/init.c:195
#5 0x00007f4c206f89a7 in tpinit (init_data=0x7f4c203ad0c0) at /home/user1/projects/endurox/libatmi/init.c:999
#6 0x00007f4c20896758 in operator() (__closure=0x1fc9188, flags=0) at src/endurox/endurox_xatmi.cpp:814
#7 0x00007f4c208a16a0 in pybind11::detail::argument_loader<long>::call_impl<void, ndrxpy_register_xatmi(pybind11::module&)::<lambda(long int)>&, 0, pybind11::detail::void_type>(struct {...} &, std::index_sequence, pybind11::detail::void_type &&) (this=0x7f4c203ad218, f=...) at pybind11/include/pybind11/cast.h:1227
#8 0x00007f4c2089ff2e in pybind11::detail::argument_loader<long>::call<void, pybind11::detail::void_type, ndrxpy_register_xatmi(pybind11::module&)::<lambda(long int)>&>(struct {...} &) (this=0x7f4c203ad218, f=...) at pybind11/include/pybind11/cast.h:1204
#9 0x00007f4c2089daa3 in operator() (this=0x0, call=...) at pybind11/include/pybind11/pybind11.h:233
#10 0x00007f4c2089db0b in _FUN () at pybind11/include/pybind11/pybind11.h:210
#11 0x00007f4c208577de in pybind11::cpp_function::dispatcher (self=0x7f4c20969ea0, args_in=0x7f4c21351040, kwargs_in=0x0) at pybind11/include/pybind11/pybind11.h:814
#12 0x00000000005f5e79 in PyCFunction_Call ()

Thread 1 (Thread 0x7f4c21b7a740 (LWP 242532)):
#0 __lll_lock_wait (futex=futex@entry=0x7f4c2065c1c0 <M_load_lock>, private=0) at lowlevellock.c:52
#1 0x00007f4c21d2f0a3 in __GI___pthread_mutex_lock (mutex=0x7f4c2065c1c0 <M_load_lock>) at ../nptl/pthread_mutex_lock.c:80
#2 0x00007f4c20607342 in ndrx_cconfig_load () at /home/user1/projects/endurox/libnstd/cconfig.c:472
#3 0x00007f4c20607a69 in ndrx_get_G_cconfig () at /home/user1/projects/endurox/libnstd/cconfig.c:554
#4 0x00007f4c205fb99f in ndrx_init_debug () at /home/user1/projects/endurox/libnstd/ndebug.c:880
#5 0x00007f4c205fd067 in ndrx_dbg_init (module=module@entry=0x7f4c2076d2f1 "ATMI", config_key=config_key@entry=0x7f4c20784ea5 "") at /home/user1/projects/endurox/libnstd/ndebug.c:1489
#6 0x00007f4c206e1cab in tpsetunsol (disp=0x7f4c20895175 <notification_callback(char*, long, long)>) at /home/user1/projects/endurox/libatmi/atmi.c:1553
#7 0x00007f4c208952cf in ndrxpy_pytpsetunsol (func=...) at src/endurox/endurox_xatmi.cpp:352
#8 0x00007f4c20896500 in operator() (__closure=0x1fcd6b8, func=...) at src/endurox/endurox_xatmi.cpp:743
#9 0x00007f4c208a151b in pybind11::detail::argument_loader<pybind11::object const&>::call_impl<void, ndrxpy_register_xatmi(pybind11::module&)::<lambda(const pybind11::object&)>&, 0, pybind11::detail::void_type>(struct {...} &, std::index_sequence, pybind11::detail::void_type &&) (this=0x7ffce6af10d

History

#1 - 05/01/2022 09:06 PM - Madars

https://github.com/endurox-dev/endurox/commit/db94add06fccb689efc9e98af49f16e567f5903c

#2 - 05/01/2022 09:09 PM - Madars

Release notes

Updated early logger (mem-logger) log replication stage, to avoid deadlock possibility.

Available from Enduro/X release 8.0.6+.

#3 - 05/01/2022 09:09 PM - Madars

- Status changed from New to Resolved

- % Done changed from 0 to 100

#4 - 05/01/2022 09:09 PM - Madars

- Status changed from Resolved to Closed