IxMem

2018 Intel Corporation

Table of Contents

1 Introduction

Welcome to IxMem

Version 2.11.0.0000

The lxmem module is a test module designed to write a pattern to a specified range of memory, then compare the bytes with an original "gold" copy. The specific ranges of memory, depending on your system configuration, may be defined by NUMA (Non-Uniform Memory Access) where you can specify the testing of specific NUMA nodes in addition to testing the entire range of memory.

To what address ranges lxmem performs these write-read-comparisons is defined by the Subtests that are supported by this module. These subtests are useful for targeting memory regions your system might support. Please read the Subtests section for more information on the supported subtests of lxmem.

A large number of parameters are provided to help target your testing to your specific needs. Please read the Parameters section for more information on the supported parameters of lxmem.

To report an issue with lxmem, please send an email to server.validation@intel.com

1.1 Installation

Installation

IxMem requires a test executor to be installed. This test executor is called iMTA Validation Stress Suite. You may have access to either just the test executor or perhaps you have access to the package containing the test executor and a handful of test modules (including IxMem). That package has a name in the form of IxVSS, where "x" is either "W" or "L", representing Windows or Linux, respectively. Those packages are designed to be installed through an installation process different from the one explained here, but once you install an IxVSS package, you automatically have IxMem installed as well.

Therefore, it is assumed you already have a test executor installed. If so, then simply place the binary in the same folder where the test executor resides.

Now that the IxMem module is installed, it is ready to be used by test packages. However, you will need to determine if package and/or snippet files contain a reference to the binary name for execution. In other words, you must open (with a text editor or with the test executor) the snippet(s) or package(s) that will make use of IxMem and confirm that the "Binary" field lists the full path to the binary.

1.2 Copyright & License

Copyright & License

Enter topic text here.

1.3 FAQ

FAQ

Answers to some commonly asked questions...

1. Why does my multi-node platform say that NUMA is disabled?
Mem requires knowledge of the NUMA configuration as reported by the operating system. Whatever the OS reports, Mem will assume is accurate. It could be that NUMA is disabled for your particular platform via a BIOS parameter or an OS boot parameter.

2. CTC keeps failing with a report that the Mem module exited without sending "DONE" message. What is the source of this problem?
Since free memory can never be considered a static number, the MemorySizeMB parameter is a percentage (or amount) based off of the total memory in the system. As a result, trying to test a system with this parameter set to too high of a value/percentage can cause the system to kill the program for trying to use too much memory. For example, Linux will use the Out-Of-Memory (OOM) killer to terminate Mem for using too much memory. This is one possible cause, but not necessarily the only cause. You should check the system log to see why the process was killed. If this is the reason, then you should reduce the value of MemorySizeMB until the program is no longer terminated prematurely. There is no specific number we can provide you because it's dependent on how your platform is configured and how many peripherals it has. If, however, this is not the cause of Mem crashing, you should copy the contents of the system log and report it to the developers as there might be a separate issue involved.

2 Subtests

Subtests

Top Previous Next

PatIO supports a list of subtests used to target your testing towards different I/O mechanisms supported by the operating system you are running your test on. However, the list below is a super-set of the tests supported as not all of the tests have been implemented.

PatIO is targeted to support the following tests...

Auto-configuration/Reconfiguration
Synchronous I/O
Asynchronous I/O
Completed I/O
Mapped I/O
CopyFile

2.1 Auto/Reconfiguration

Auto/Reconfiguration

Top Previous Next

Auto-configuration/Reconfiguration is a required subtest of any module CTC supports. It is designed to initialize test parameters based off of the platform configuration. If auto/reconfiguration fails, it is highly unlikely any other subtest will function correctly.

2.2 Local Node Test

Local Node Test

Top Previous Next

Regions of memory that are on the same Node as a particular CPU thread are considered LOCAL. This subtest focuses on LOCAL memory testing.

The Test Memory Percent parameter is used to specify the amount of memory tested, with a default value of 80%.

2.3 Remote Node Test

Remote Node Test

Top Previous Next

Regions of memory that are NOT on the same Node as a particular CPU thread are considered REMOTE. This subtest focuses on REMOTE memory testing.

The Test Memory Percent parameter is used to specify the amount of memory tested, with a default value of 80%.

2.4 Memory Controller Test

Memory Controller Test

Top Previous Next

Memory Controller Test.

The Test Memory Percent parameter is used to specify the amount of memory tested, with a default value of 80%.

2.5 Pseudo-Random Node Test

Pseudo-Random Node Test

Top Previous Next

This subtest pseudo-randomly tests both LOCAL and REMOTE memory.

The Test Memory Percent parameter is used to specify the amount of memory tested, with a default value of 80%.

2.6 Non-NUMA Test

Non-NUMA Test

Top Previous Next

Only a single NODE is present. Memory available to the NODE are tested.

The Test Memory Percent parameter is used to specify the amount of memory tested, with a default value of 80%.

2.7 All Test

All Test

Top Previous Next

The All subtest is designed to perform all of the NUMA-related subtests in sequence. The process is essentially the same as specifying Local Node, Remote Node, Pseudo-Random Node, and Memory Controller tests all in the test executor to be run serially. There is only a slight time savings in running the All test simply because it will not have to clean up resources and free memory before moving to the next test.

2.8 Cache Test

Cache Test

Top Previous Next

The Cache test is similar to the Non-NUMA test with the exception that it relies on the Cache size parameter, not the Test Memory Percent parameter. However, the value of the Cache size parameter is meant to represent the size of the cache of the memory/processor. Thus, in order to exercise that cache, we need to allocate a buffer of the same size and test it as we would with any other subtest.

If Cache size is not set, this subtest will return an error.

NOTE: This is a non-deterministic, implicit test of cache. Since cache is not exposed through any interface by the OS, we are making a best effort at testing it. If a failure happens, the test itself is reproducible, but how hardware has populated the cache with the pattern data may not be reproducible.

3 Parameters

Parameters

Top Previous Next

This section describes in detail the user-definable parameters available for PatIO. Note that while each of them may be user-definable, some may be better left alone. Please read the help for each parameter used in your testing.

3.1 Cache size

Cache size

Top Previous Next

This parameter is meant to be used with the Cache subtest. It specifies the size of the cache to be tested implicitly. Meaning, the user is to define this parameter based off of what they know of their system. As of this writing, no method is known to determine the size of the cache (MDCRAM).

3.2 Invert

Invert

Top Previous Next

Invert specifies whether you want to perform an inverted pattern test. An inverted pattern test is an additional test performed per pattern where the original "gold" buffer is filled with pattern data that is bit-flipped from the original pattern. Testing against the inverted pattern is performed during the same iteration as the regular pattern test. Thus, your test sequence will look like this...

Pattern0, **~Pattern0**, Pattern1, **~Pattern1**, ..., Pattern(N-1), **~Pattern(N-1)**

...where the inverted pattern tests are represented in bold.

Optional
Default: 0
Range: 0 = Does not perform inverted pattern test, 1 = Performs inverted pattern test

3.3 Iterations

Iterations

Top Previous Next

Iterations is a loop limiter to specify the number of write-read-compare operations per pattern file. One iteration is equivalent to performing all Writes, all Reads and comparing the full memory allocation. Keep in mind that although "0" is a valid value for this parameter, at least one iteration is required to be performed. This value is ignored if the Runtime parameter is specified.

Optional
Default: 0
Range: 0 - 4294967295 (2^{32} - 1)

3.4 No siblings

No siblings

Top Previous Next

This parameter tells the lxmem subtests (exception: auto/reconfigure) to not test the "thread siblings" of a processor. In other words, it's similar to disabling hyper-threading in the system BIOS. For example, the logical CPU 0 might have a thread sibling representing logical CPU 2. Enabling this parameter tells lxmem to not use logical CPU 2 in the testing.

This parameter is mutually exclusive to the nodes_override parameter since you are able to effectively make the same change in that parameter.

Optional
Default: 0
Range: 0 = Does not disable thread siblings, 1 = Disables thread siblings (only lowest logical processor number of the siblings enabled)

3.5 Nodes

Nodes

Top Previous Next

The Nodes parameter is filled with the processor and memory configuration of the system. This parameter is typically populated during the Auto/reconfigure subtest. The fields are as follows:

node	This is the node instance. If non-NUMA, node[0] should be the only column.
status	This is the status of the node. Currently, a node is only disabled if the physical memory field is 0. This scenario has never been encountered, thus operations with a disabled node have yet to be tested.
physical memory	This is the total memory of the node (as reported by the OS). The MemorySizeMB parameter is a percentage of this value minus the reserved memory described next. "Free" memory (as reported by the OS), can be quite nebulous, so it is not relied upon.
reserved memory	This is the default amount of memory that the module will not include when calculating the MemorySizeMB amount. It's a best guess estimate that should allow all threads of this module to operate without starvation. It is not necessarily enough for other modules running in parallel to lxmem.
CPUs	All rows under the reserved memory field will list the CPUs associated with the node instance(s). Some configurations may not have any CPUs assigned to the node. This is not necessarily an error condition, but if no CPUs are assigned to any other node, then no testing will be performed.

This is a parameter not meant to be modified by end-users. Running any subtest (other than auto/reconfigure) with this table populated incorrectly will result in immediate failure. Thus it is always recommended to perform an auto/reconfigure before running any other subtest.

Required
Default: As set in the snippet or package file
For developers only - DO NOT MODIFY

3.6 nodes_override

nodes_override

Top Previous Next

The nodes_override parameter is of little use to the normal user, but included for debug purposes. It is a parameter array that follows the same format as the Nodes parameter. However, a user may modify this parameter to override what is shown in the Nodes parameter. The restriction to this is that all values within the nodes_override parameter must be less than or equal to the values in the Nodes parameter. For example, the total memory value or the number of processors listed. The only exception to this is the reserved memory field, where the user may decide to increase the amount of memory not to be used by the lxmem subtest.

The user may choose to shift processors from one node to another; remove processors completely; decrease the amount of total memory; disable nodes (status = off). For most of these cases, an advisory message will appear letting the user know what has changed.

If this parameter must be used, the recommendation is to use this parameter for testing scenarios where the user doesn't want a specific node's CPUs to be enabled. Or testing a scenario where the module views certain processors as belonging to a different node. This parameter is not affected by an auto/reconfigure, so care must be taken when using this parameter.

Optional
Default: Does not exist.
Values: Copy/paste the Nodes parameter and rename to "nodes_override". Make the desired adjustments. Recommended for advanced users only.

3.7 Offset

Offset

Top Previous Next

Offset determines how many bytes to incrementally progress through a pattern file for each iteration of testing. For example, if you have a pattern file with contents "0123456789" and Offset of 2, then the first iteration will compare the original data set as is, the second iteration will compare against "2345678901", the third iteration will compare against "4567890123", etc. Eventually, this will loop back to somewhere around the beginning of the pattern, but not necessarily at byte[0]. For example, if you have the same pattern file and an offset of 3, the iterations will be as such: 0123456789, 3456789012, 6789012345, 9012345678, 2345678901, etc.

The Offset parameter can be used regardless if Runtime or Iterations is set. Just note that an increment into the pattern file by an Offset amount counts as one iteration.

Optional
Default: 0 (No incremental offset. The comparison will always start at byte[0] of the pattern)
Range: 0 - (2^{32} - 1) (However, this value will be modded against each pattern file size during testing)

3.8 Pattern directory

Pattern directory

Top Previous Next

Pattern directory specifies the directory where the local pattern files are installed. This is a required parameter as no comparisons can be made without patterns.

Required
Default: Linux: /LinuxVSS/patterns, Windows: TBD
Range: Absolute path with maximum of 512 characters

3.9 Patterns

Patterns

Patterns is an array parameter where you can specify multiple pattern files (ending in .pat) for testing against the memory. If there are no slashes in the specified pattern name, then it is assumed this is a pattern file contained within the Pattern directory. Otherwise, the entry will be taken as a full path to the pattern file.

Required
Default: Some patterns may be specified by default depending on the type of test being performed.
Range: Filename(s) with a maximum of 512 characters each.

3.10 Random node seed

Random node seed

Random node seed is the beginning seed the user may specify when enabling the Randomize parameter for randomizing the node entries during the Pseudo-Random Node test. If you do not specify a seed, and Randomize is enabled, a seed will be generated for the test based off of the current time. If you do not specify a seed, and Randomize is disabled, the node order will be based off of a seed of 0.

Optional
Default: 0
Range: 0 - (2^{32} - 1)

3.11 Random pattern seed

Random pattern seed

Random pattern seed is the beginning seed the user may specify when enabling the Randomize parameter. This parameter is ignored if Randomize is disabled. This seed is used for the randomization of the pattern file order. By default, the order of the patterns tested against is the same list as the Patterns parameter. If you do not specify a seed, and Randomize is enabled, a seed will be generated for the test based off of the current time.

Optional
Default: None
Range: 0 - (2^{32} - 1)

3.12 Randomize

Randomize

Randomize will enable sorting the Patterns list and the Pseudo-Random Node list in a random order. If no Random pattern seed or Random node seed is specified, a seed will automatically be generated based off of the current time. The seed, in either case, will be reported out so the user knows what seed was used if they need to repeat the same test.

Optional
Default: 0 (Disabled)
Range: 0/1 (false/true)

3.13 Reads

Reads

Top Previous Next

This parameter specifies the number of reads performed from the destination address in memory before any comparisons are performed. No offsets are changed between read operations (of Window size amount), so additional read operations are meant to be redundant. A user may want to increase this value if, for example, they are concerned about cache operations not working as expected.

Optional
Default: 1
Range: 1 - 4294967295 (2^{32} - 1), 0 produces error

3.14 Reverse

Reverse

Top Previous Next

Reverse is a parameter that will switch the bytes of a pattern as if it was read starting from the last byte and traversing towards the first byte. For example, "ABCD", after reversing, would become "DCBA". So the latter would become the pattern tested and the original pattern would not be used.

The reversal also works for odd pattern lengths: "ABCDE" -> "EDCBA".

Optional
Default: 0
Range: 0 = Does not reverse the pattern data, 1 = Reverses the pattern data

3.15 Runtime

Runtime

Top Previous Next

Runtime specifies the number of seconds this test should run. The module will divide this amount equally amongst the Patterns list. However, due to setup processes and other system variables that can consume time, each pattern may not be allotted the full section of time for testing. This module accounts for that extra process time and will adjust accordingly. In other words, this module should finish in Runtime seconds +/- 1 second. The exception is if the specified Runtime is very small. This module runs on the premise that each pattern is tested against at least once. So for the minimum value, a good rule-of-thumb is to provide at least 3 seconds per pattern.

Optional (but highly encouraged)
Default: 3600 seconds (1 hour)
Range: 0 - (2^{32} - 1)

3.16 Test Memory Percent

Test Memory Percent

Top Previous Next

Test Memory Percent is the percentage of system memory to test. This percentage is taken against the total system memory, not the free system memory. Free system memory can be arbitrary due to other system programs/services or other test modules running in parallel. Because of this, the user must tune the percentage to fit the system resource capabilities. If the percentage is too high, the risk of running out of memory increases. If the percentage is too low, the amount of memory tested may be insufficient.

NOTE: The test executor, CTC, may report an error that the lxmem module never sent a "DONE" message. The first thing to check is the system log to see if the operating system terminated lxmem due to the Out-Of-Memory Killer (OOM). This is an indication that this parameter is set too high during this particular stage of the test flow.

Optional
Default: 80%
Range: 1 - 100 (anything outside this range produces an error, anything above 90 is highly discouraged)

3.17 Window size

Window size

Top Previous Next

Window size specifies the amount of data (in bytes) to transfer to/from the target with each copy/compare operation. This parameter may be larger than the size of any pattern file, but it must be smaller than the allocated memory size.

Optional
Default: 0
Range: Minimum of 8 B, 0 = (total memory * Test Memory Percent) / number of CPUs

3.18 Writes

Writes

Top Previous Next

This parameter specifies the number of writes performed to the destination address in memory before any Reads are performed. No data or offsets are changed between write operations (of Window size amount), so additional write operations are meant to be redundant. A user may want to increase this value if, for example, they are concerned about cache operations not working as expected.

Optional
Default: 1
Range: 1 - 4294967295 (2^{32} - 1), 0 produces error

3.19 Memory Percent Variation

Test Memory Variation

Top Previous

Test Memory Variation is the percentage difference allowed between a OS detected memory and package configuration memory. This parameter is only necessary when a pre-configured memory size parameter is used to test system without running lxmem "AutoConfig". The Linux OS may report available memory size slightly different from system to system, even with an identical DIMM configuration.

Optional
Default: 0.0%
Range: 0.0 (double precision) - 100.0

Index

Endnotes 2... (after index)

Back Cover