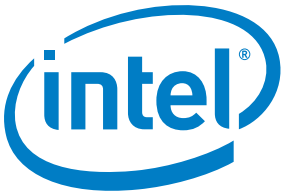

Intel® heteroStreams 0.9

Release Notes for Intel® MPSS 3.6 release of hStreams

September 2015

US

Revision: 0.9

World Wide Web: http://www.intel.com

Legal Disclaimer

Contents

§

1 Introduction

Intel® heteroStreams is a library that supports task concurrency on heterogeneous platforms.

The concurrency may be across nodes; within a node for small matrix operations; and in the overlapping of computation and communication, particularly for tiled solutions. It relieves the user of complexity in dealing with thread affinitization, offloading, memory types, and memory affinitization. The formal name of this product is heteroStreams, and the casual name is hStreams.

2 Changes

2.1 New features

2.1.1 Using host as the sink for streaming operations

Users may now create logical domains and streams in the host physical domain, HSTR_SRC_PHYS_DOMAIN. Streams created in the host physical domain and the streams created on Intel® Xeon Phi™ are fully interoperable.

Please note that with the Intel® MPSS 3.6 release, this feature is provided as a preview on the Linux* operating system only. Further support and enhancements are envisioned. Please refer to Known issues and limitations for details.

2.1.2 Buffer instantiations

hStreams buffers now have separate, independent instantiations per each logical domain. Previously, a buffer instance was shared between all logical domains belonging to the same physical domain.

Moreover, when a logical domain is created after a buffer is created, the buffer will not have an instantiation in that logical domain.

2.1.3 Explicit control over buffer instantiations and buffer properties

Users may now exercise precise control over which logical domains a buffer should be instantiated for (refer to the documentation of hStreams_Alloc1DEx()) and query, add and remove buffer's instantiations after the buffer had been created (refer to the documentation of hStreams_AddBufferLogDomains(), hStreams_RmBufferLogDomains(), hStreams_GetBufferNumLogDomains() and hStreams_GetBufferLogDomains()).

Buffers can now be described using extended properties, such as the aliasing, source-pinned or incremental property. Refer to the documentation of hStreams_Alloc1DEx(), hStreams_GetBufferProps() for details). Also, please note that some buffer properties are not currently implemented – please refer to Known issues and limitations for details.

2.1.4 Buffer transfer between arbitrary logical domains

Users may use a new API, hStreams_EnqueueDataXDomain() to enqueue in a stream a data transfer action which involves buffer instantiations on arbitrary logical domains.

2.1.5 Partially overlapping streams

Users are now free to create logical streams which partially overlap each other. Two logical streams which partially overlap will map to a different physical stream.

To ease the management and monitoring of a physical domain's usage, a new function to query the oversubscription of individual hardware threads is provided, hStreams_GetOversubscriptionLevel().

2.1.6 Updated library format string

The version string returned by the hStreams_Version() function is now simplified and takes the format "MAJOR.MINOR[.MICRO]", where individual components correspond to heteroStreams library version. The ".MICRO" part is omitted if MICRO component of the version equals 0.

A companion API to query the length of the version string is now provided, hStreams_GetVersionLen().

Additionaly, three new preprocessor definitions are exposed, HSTR_VERSION_MAJOR, HSTR_VERSION_MINOR and HSTR_VERSION_MICRO.

2.1.7 Thread safety

All the APIs have been updated with the description of how they behave in multithreaded usage scenario. Great care has been taken to make the APIs thread safe while retaining a high degree of concurrency allowed in the usage of the API.

2.1.8 Unification of app API function signature

The interface of app API functions pertaining to performing an operation in a selected logical stream now obey the following convention regarding argument order: (stream_id, parameters, return value (if any), output event). The affected APIs are:

- hStreams_app_xfer_memory()
- hStreams_app_invoke()
- hStreams_app_memset()
- hStreams_app_memcpy()
- hStreams_app_sgemm()
- hStreams_app_dgemm()
- hStreams_app_cgemm()
- hStreams_app_zgemm()

§

3 Known issues and limitations

3.1 Using host as a target

The feature of creating logical domains and streams in the host physical domain as well as using those streams for executing actions is currently limited to the Linux* operating system only. Affected API functions return a HSTR_RESULT_NOT_IMPLEMENTED error code on the Windows* operating system.

Support for the following convenience functions is not provided on the host physical domain:

- hStreams_app_memset()
- hStreams_app_memcpy()
- hStreams_app_sgemm()
- hStreams_app_dgemm()
- hStreams_app_cgemm()
- hStreams_app_zgemm()
- hStreams_memset_sink()
- hStreams_memcpy_sink()
- hStreams_sgemm_sink()
- hStreams_dgemm_sink()
- hStreams_cgemm_sink()
- hStreams_zgemm_sink()

The following app initialization APIs do not yet automatically make use of resources on the host. Use of host resources is supported with the core APIs.

- hStreams_app_init()
- hStreams_app_init_domains()

3.2 Extended buffer properties

Support for a memory type other than HSTR_MEM_TYPE_NORMAL is not provided and the corresponding API – hStreams_Alloc1DEx() – will return a HSTR_RESULT_NOT_IMPLEMENTED error code if provided with an erroneous memory type in the buffer properties structure.

Further, support for the HSTR_BUF_PROP_AFFINITIZED is not provided and the corresponding API – hStreams_Alloc1DEx() – will return a HSTR_RESULT_NOT_IMPLEMENTED error code if provided with buffer properties structure with this flag set.

§

4 Deprecated functionality

4.1 libhstreams_sink.so

The Intel® Xeon Phi™ coprocessor library, libhstreams_sink.so is now considered deprecated and will be removed in the upcoming release.

This library exposes the following functions, all of them being marked for deprecation in the Intel® Xeon Phi™ coprocessor library:

- _hStreams_EmitMessage
- hStreams_GetCurrentOptions
- hStreams_GetOptions__hStreams_EmitMessage
- hStreams_GetOptions__hStreams_FatalError
- hStreams_GetOptions_dep_policy
- hStreams_GetOptions_kmp_affinity
- hStreams_GetOptions_openmp_policy
- hStreams_GetOptions_phys_domains_limit
- hStreams_GetOptions_time_out_ms_val
- hStreams_GetOptions_verbose
- hStreams_GetVerbose
- hStreams_GetVersionStringLen
- hStreams_ResetDefaultOptions
- hStreams_ResultGetName
- hStreams_SetOptions
- hStreams_SetVerbose
- hStreams_Version

4.2 Functionality related to emitting messages

hStreams_EmitMessage_Prototype_Fptr, the interface of the message emitting function is expected to change in a future release. As such, the setting HSTR_OPTIONS._hStreams_EmitMessage is expected to be replaced with an updated setting. Moreover, the default implementation of the message emitting interface, _hStreams_EmitMessage function will be removed in the future.

4.3 Signature of hStreams_GetLogStreamDetails()

The signature of the hStreams_GetLogStreamDetails() is expected to change in that the in_LogDomainID parameter will become an output parameter out_pLogDomainID.

§

5 Removed functionality

5.1 ConvertToUint64_t

The ConvertToUint64_t class was removed from the hStreams_common.h header. A reference implementation is still provided in the reference codes, in the ref_code/common/type_converter.h header file.

§