OPENFABRICS ALLIANCE

14th ANNUAL WORKSHOP 2018

OPENSHMEM AND OFI: BETTER TOGETHER

James Dinan, David Ozog, and Kayla Seager

Intel Corporation

[April 11, 2018]

intel

NOTICES AND DISCLAIMERS

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/benchmarks .

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

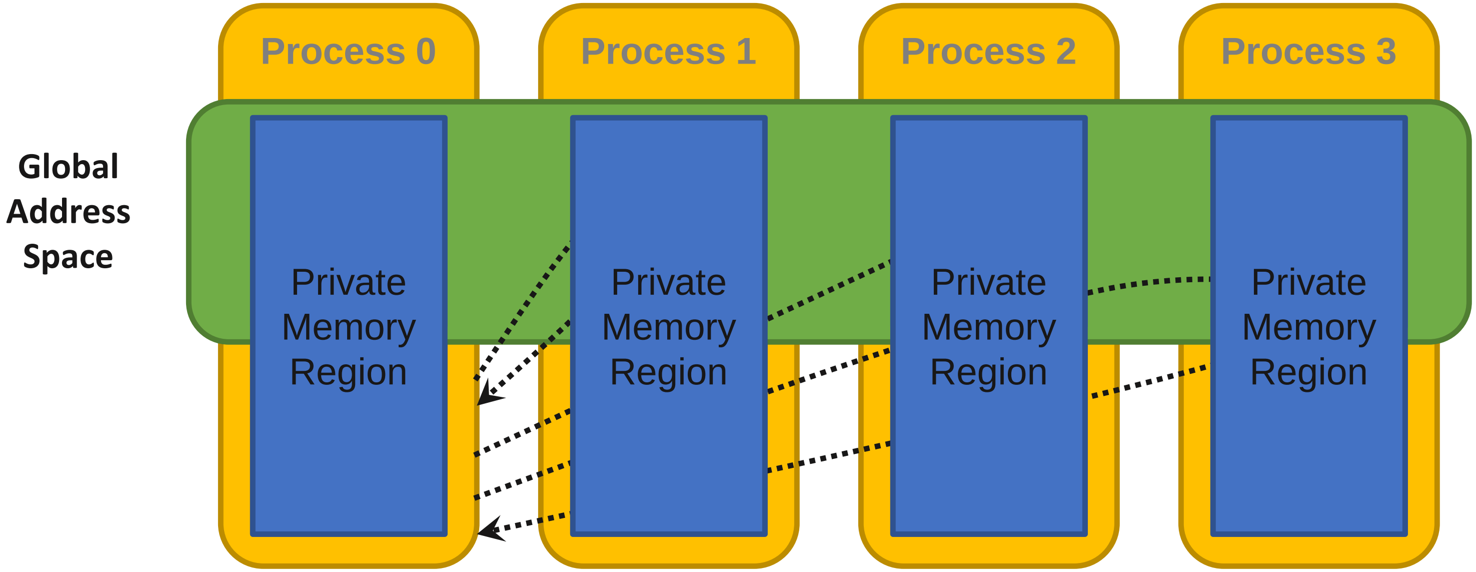
Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

WHAT IS OPENSHMEM?



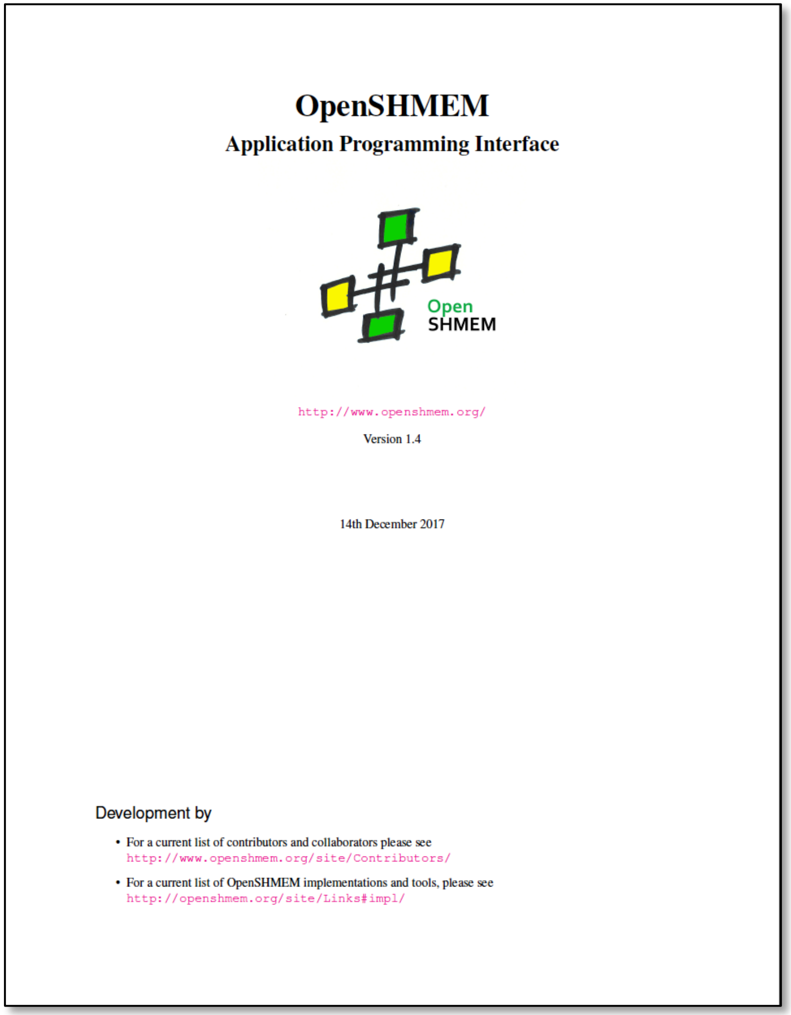
- **Open standard for SHMEM programming model**
- **Partitioned Global Address Space memory model, SPMD execution**
  - Part of the memory in a process is exposed for remote access
  - Asynchronous read (get), write (put), and atomic update operations
- **Fence (ordering), quiet (remote completion), barrier/wait (sync)**

OPENSHMEM 1.4

- **Specification ratified Dec. 14, 2017**
  - Thread safety
  - Communication management API (contexts)
  - Test, sync, calloc
  - Bitwise atomic operations
  - Updated C11 generic selection bindings

- **Committee actively working on 1.5**
  - Happy to have you join us!

- **Intel is engaged in the Sandia OpenSHMEM implementation effort**
  - SOS v1.4.1 release candidate out
  - Open source, supports OFI and Portals
  - Req.: FI_RMA, FI_ATOMICS, FI_EP_RDM
  - First open source implementation to support OpenSHMEM 1.3 and 1.4
  - https://github.com/Sandia-OpenSHMEM/SOS



OpenSHMEM

Application Programming Interface

http://www.openshmem.org/

Version 1.4

14th December 2017

Development by

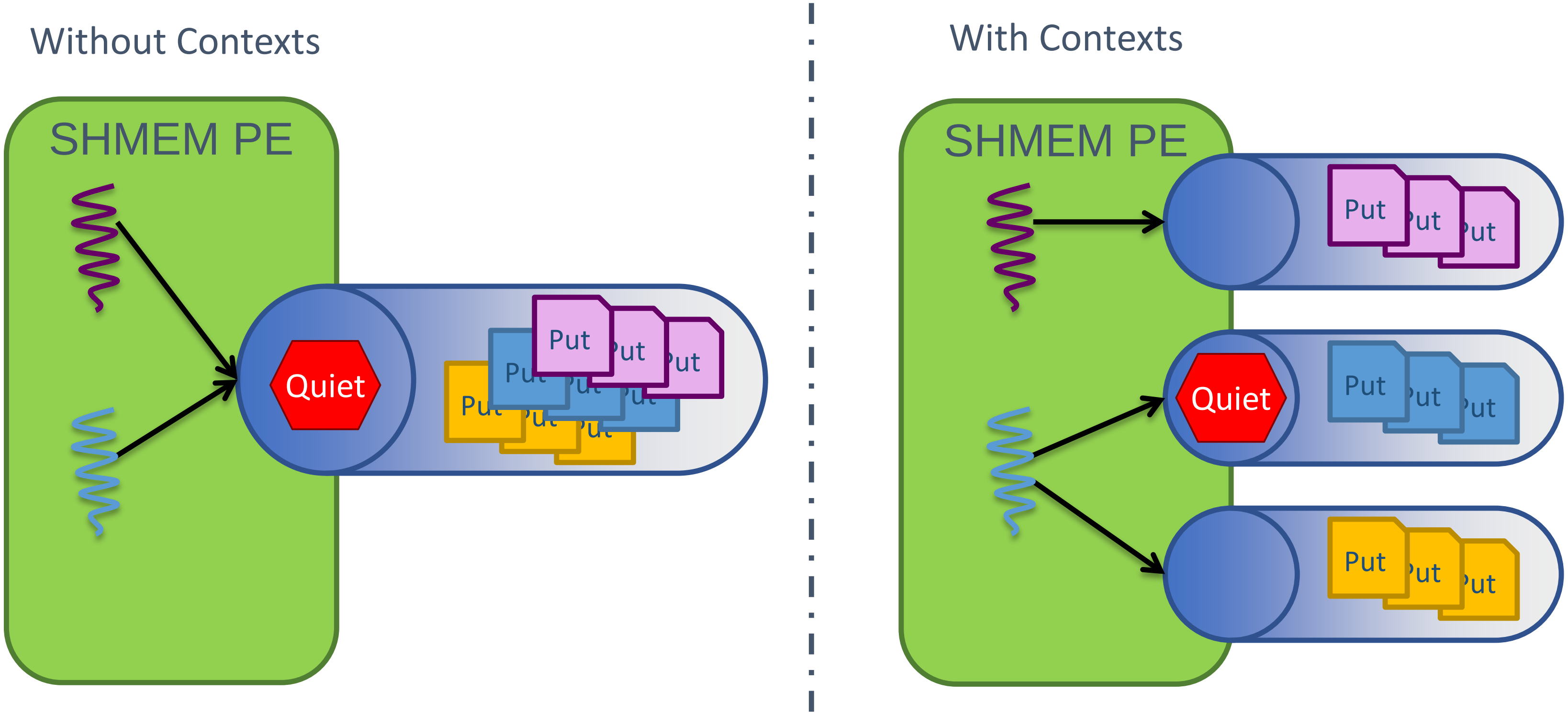
- For a current list of contributors and collaborators please see http://www.openshmem.org/site/Contributors/
- For a current list of OpenSHMEM implementations and tools, please see http://openshmem.org/site/Links#impl/

OPENSHMEM 1.4 THREAD SAFETY

```
int  shmem_init_thread(int requested, int *provided);
void shmem_query_thread(int *provided);
```

- **Defines semantics of threads and OpenSHMEM routines**
- **Threading level selected at initialization:**
  - SHMEM_THREAD_SINGLE: No threading
  - SHMEM_THREAD_FUNNELED: Master thread calls SHMEM API
  - SHMEM_THREAD_SERIALIZED: Any thread calls SHMEM API, but serialized
  - SHMEM_THREAD_MULTIPLE: Any thread calls SHMEM API, concurrently
- **Sandia OpenSHMEM supports FI_THREAD_SAFE and COMPLETION**
  - FI_THREAD_SAFE: SOS-level atomics, no mutexes
  - FI_THREAD_COMPLETION: SOS-level mutexes, but can be eliminated with user-provided hints

OPENSHMEM CONTEXTS: ISOLATION AND OVERLAP



- **Programmer chooses which operations are completed by quiet**
  - Control communication/computation overlap
  - Eliminate interference between threads

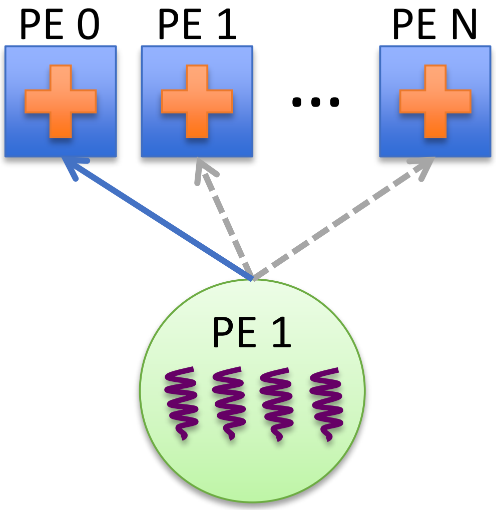
OPENSHMEM 1.4 CONTEXTS API

```
int  shmem_ctx_create(long options, shmem_ctx_t *ctx);
void shmem_ctx_destroy(shmem_ctx_t ctx);

void shmem_ctx_putmem(shmem_ctx_t ctx, void *dest,
                      const void *source, size_t nbytes, int pe);

void shmem_ctx_fence(shmem_ctx_t ctx);
void shmem_ctx_quiet(shmem_ctx_t ctx);
```

- **SHMEM_CTX_DEFAULT: Created during initialization**
  - Legacy SHMEM API operations are performed on the default context
- **Context options:**
  - **SHMEM_CTX_SERIALIZED:** The given context will not be used by multiple threads concurrently
  - **SHMEM_CTX_PRIVATE:** The given context will be used only by the thread that created it
- **Options enable thread synchronization optimizations**
  - Need a way to pass hints to OFI in FI_THREAD_SAFE mode to relax synchronization

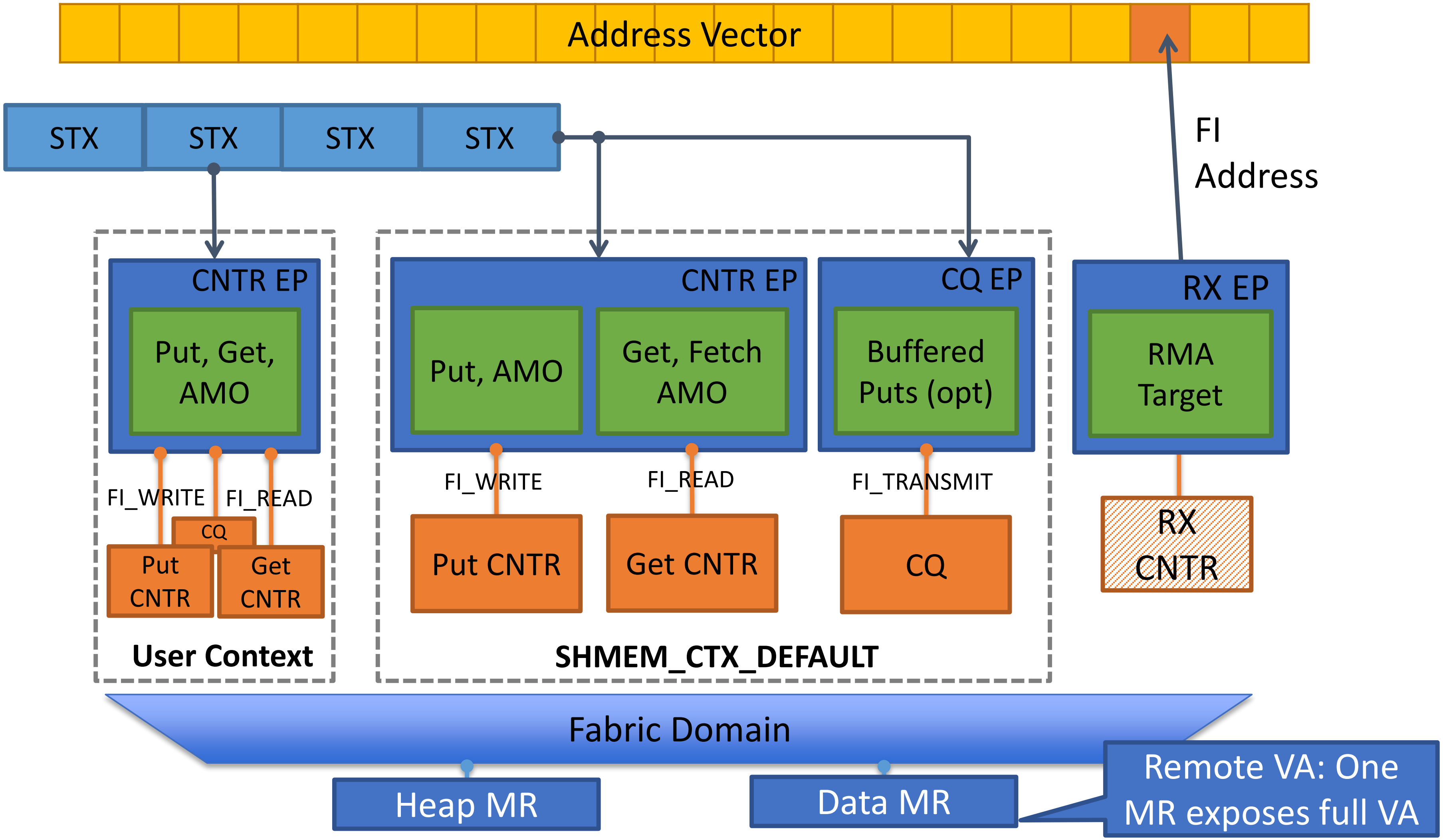
CONTEXTS AND THREADS EXAMPLE

```
long task_cntr = 0; /* Next task counter */

int main(int argc, char **argv) {
  long ntasks = 1024; /* Total tasks per PE */
  ...

#pragma omp parallel
  {
    shmem_ctx_t ctx;
    int task_pe = shmem_my_pe(), pes_done = 0;
    shmem_ctx_create(SHMEM_CTX_PRIVATE, &ctx);

    while (pes_done < npes) {
      long task = shmem_atomic_fetch_inc(ctx, &task_cntr, task_pe);
      while (task < ntasks) {
        /* Perform task (task_pe, task) */
        task = shmem_atomic_fetch_inc(ctx, &task_cntr, task_pe);
      }
      pes_done++;
      task_pe = (task_pe + 1) % shmem_n_pes();
    }

    shmem_ctx_destroy(ctx);
  } /* End parallel section */
}
```



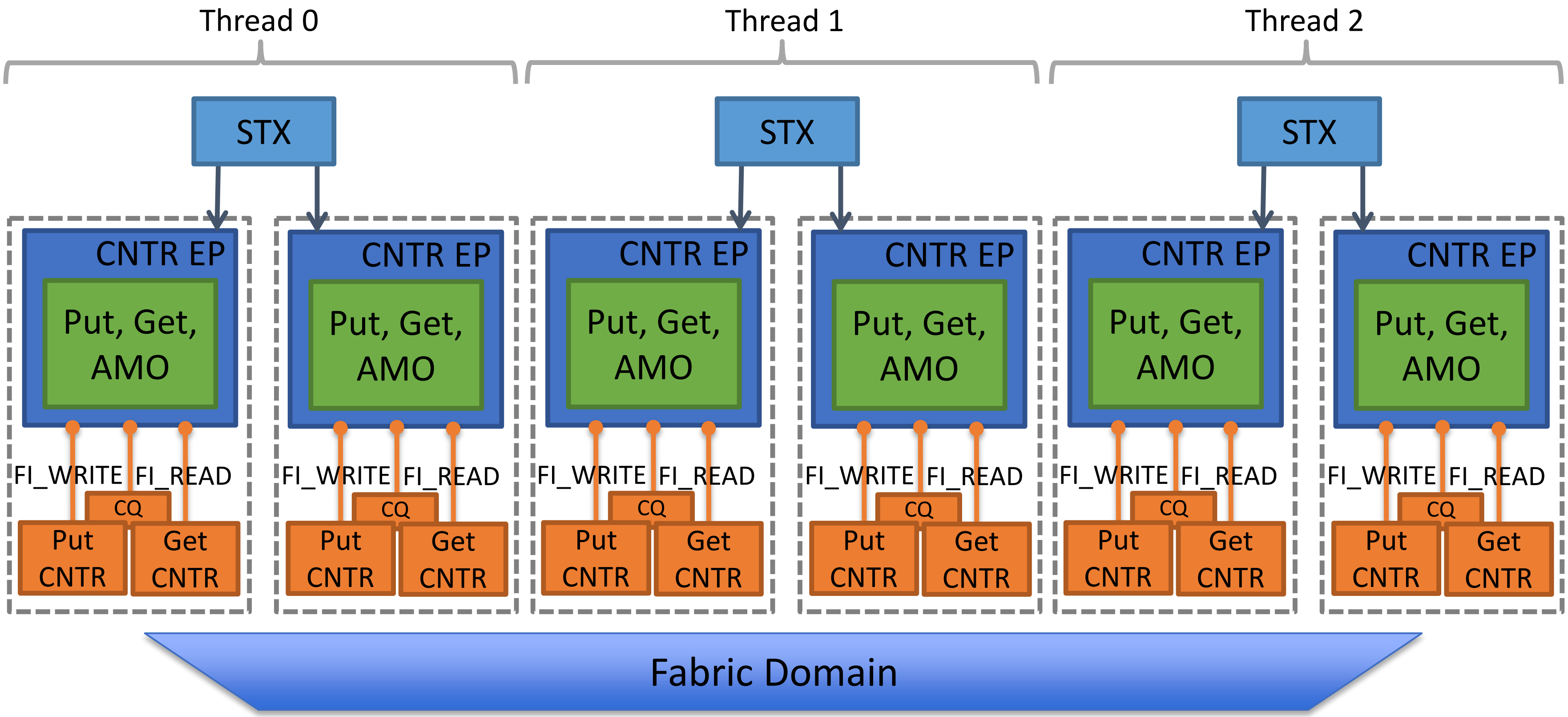
- **Dynamic load balancing**
  - Threads process local tasks
  - Proceed to help round-robin
- **Contexts isolate threads**
  - Fetch-inc completion waits on event counter
  - Threads share counter
  - Leads to interference

SOS 1.4.X OFI TRANSPORT ARCHITECTURE



Remote VA: One MR exposes full VA

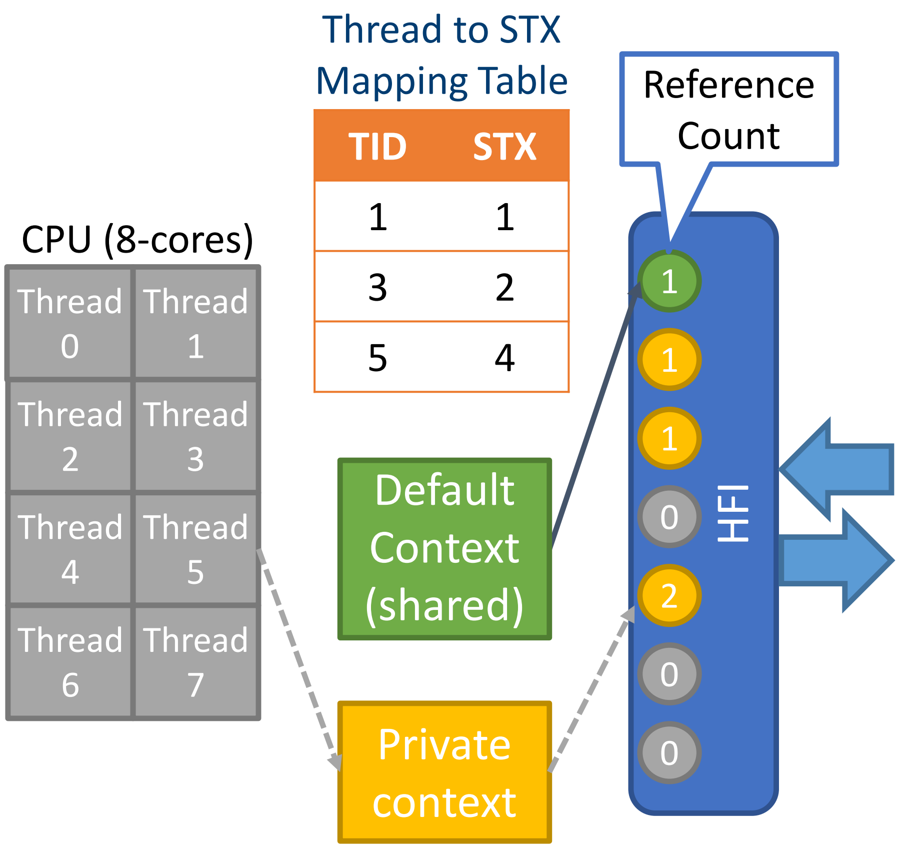
THREAD-AWARE RESOURCE PRIVATIZATION



- **Use shareable transmit context (STX)**
  - Leverage thread-context mapping hints to optimize STX assignment
  - Scalable endpoints TX resource is automatic, can't optimize for usage model

SHAREABLE TRANSMIT CONTEXT MANAGEMENT

- **STX allocator controls assignment of STX to contexts**
  - STXs are in shared, private, or free state
  - Default context is created first and claims 0th STX as shared
- **Private contexts**
  - Check TID-to-STX table for given thread
  - If no STX, attempt to allocate a private STX to the calling thread
  - If none available, treat as shared
- **Shared contexts**
  - Allocate according to policy: round-robin, random, least used, etc.
  - Set low water mark to favor private usage or disable private to favor sharing



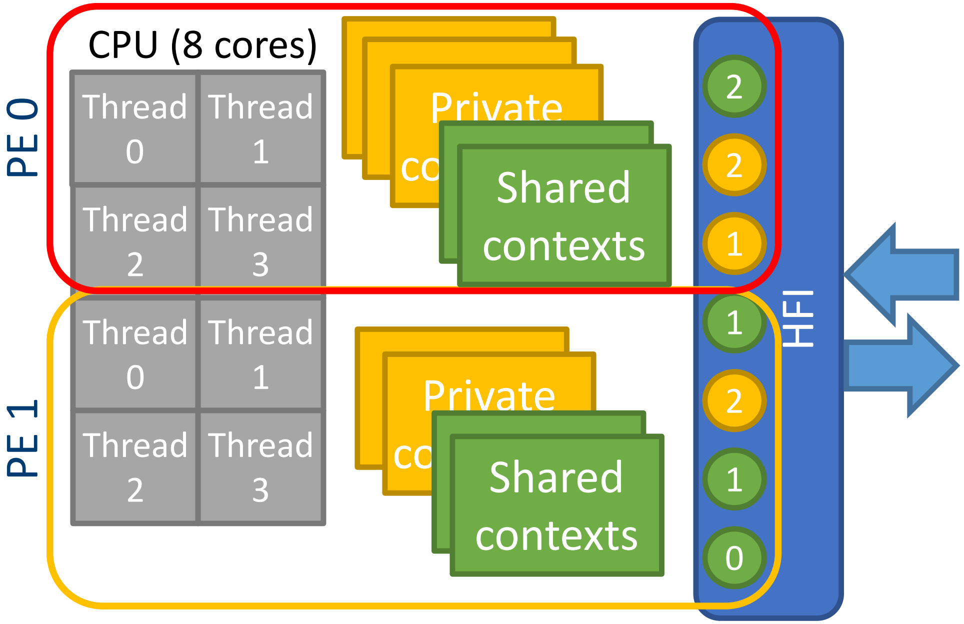
Thread to STX Mapping Table

TID	STX
1	1
3	2
5	4

STX PARTITIONING

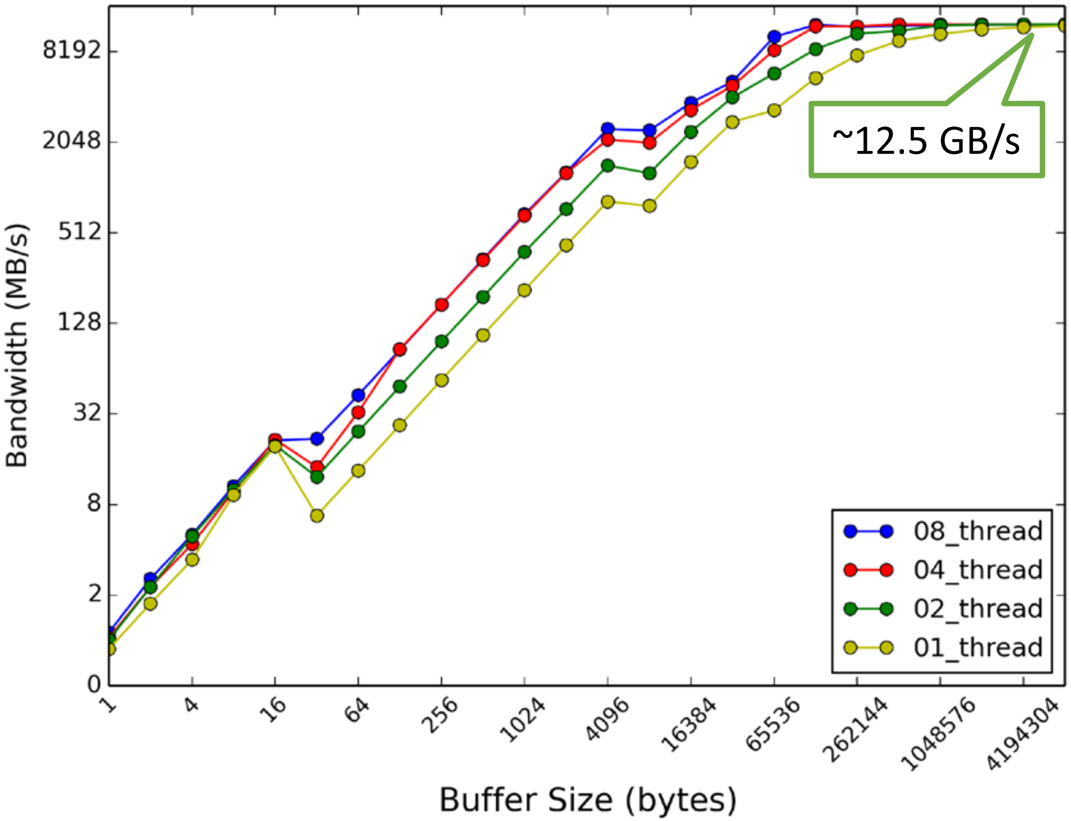
- **Multiple PEs per node**
  - Query maximum number of STX and automatically partition
  - Or manually Set maximum STX per PE: SHMEM_OFI_STX_MAX

- **OpenSHMEM threading introduces new and interesting resource management challenges**
  - Exposes threads to middleware enabling optimizations
  - Good solutions critical for realizing full performance potential



BLOCKING PUT BANDWIDTH

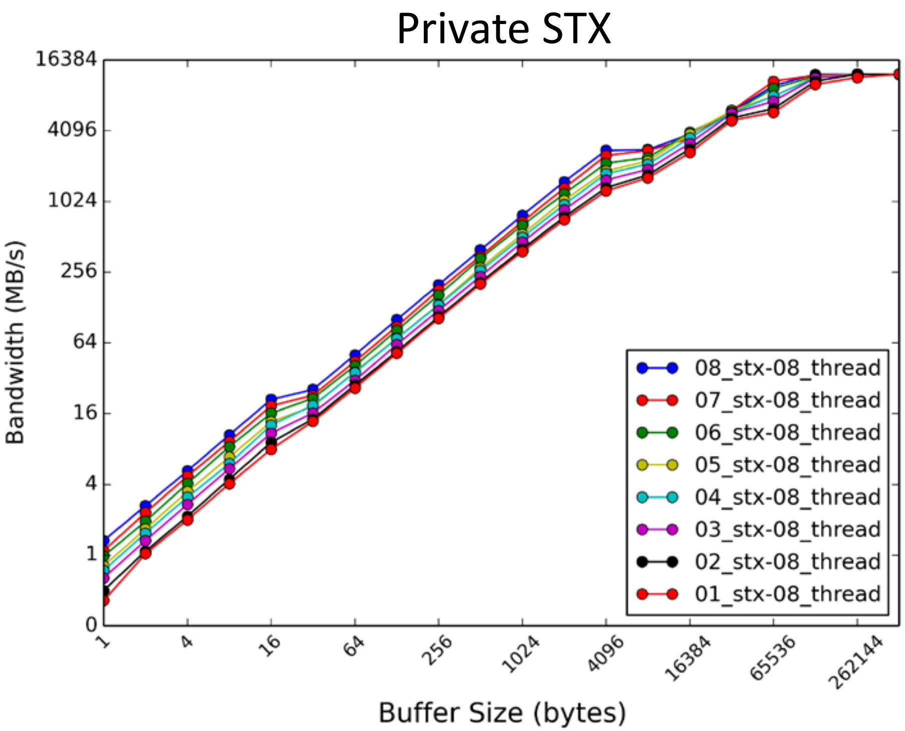
- **Early results subject to change**
- **Multithreaded point-to-point unidirectional bandwidth test**
  - Each thread has a separate context
- **Two nodes, 1 PE per node:**
  - Dual socket Intel® Xeon® CPU E5-2699 v3 (Haswell) 2.30GHz
    - 18 cores, 36 threads
  - Intel® Omni-Path Architecture
  - Nodes connected via single switch
  - 64 GB RAM
  - Libfabric v1.6.0, PSM2 provider
  - CentOS* Linux release 7.3.1611
- **Sandia OpenSHMEM v1.4.1rc1**
  - Manual progress and thread completion support enabled

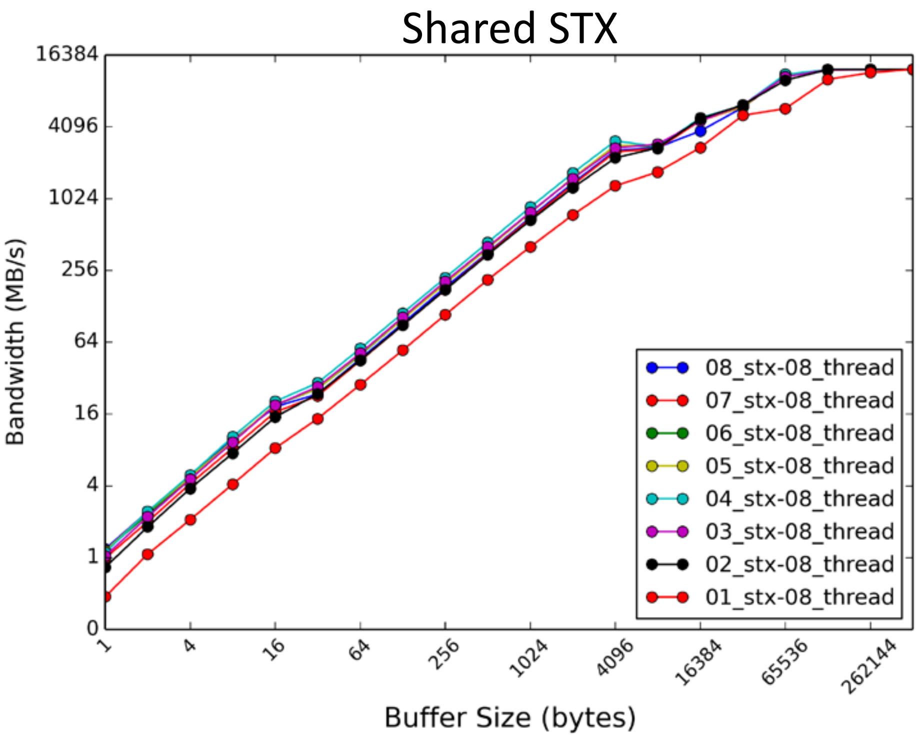


~12.5 GB/s

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance.

COMPARISON OF STX ALLOCATION POLICIES





- **Experiment: 8 threads per PE, increase STX from 1 to 8**
  - Always at least one shared STX (default context); how we assign the rest?
  - E.g., Private @2 STX, 1 private, 7 threads 1 STX. Shared @2 STX, 8 threads share 2 STX.
- **Application usage model determines best method for using available resources**

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance.

OPENFABRICS ALLIANCE

14th ANNUAL WORKSHOP 2018

THANK YOU

James Dinan

Intel Corporation

intel