OpenSHMEM over MPI-3 one-sided communication

Jeff Hammond, Sayan Ghosh
and Barbara Chapman

Argonne National Laboratory and University of Houston

6 March 2014

Background

Fundamental premises:

- MPI (community) is uncompromising w.r.t. portability.
- SHMEM (community) is uncompromising w.r.t. performance.

Historically, MPI and SHMEM had (mostly) non-overlapping feature sets.

- MPI-1 provided message passing.
- MPI-2 provided one-sided communication that was too restrictive for many applications due to the requirement that it run on the Earth Simulator (for example); atomics were missing and the memory model was challenging (even to understand).
- MPI-3 tried very hard to get it right w.r.t. one-sided communication.

New Features in MPI-3

- Designed to make it possible to use as a conduit for Global Arrays, ARMCI, SHMEM, UPC, CAF, etc.
- Defined new memory model (UNIFIED) for cache-coherent architectures.
- More flexible synchronization semantics (local completion).
- Real atomics (F&Op and C&S).
- Scalable memory allocation (potentially symmetric under-the-hood).
- Communicator creation that isn't collective on the parent group (not RMA).

Motivation

- Academic desire to verify the MPI Forum's belief that MPI-3 is a reasonable conduit for PGAS.
- `apt-get install openshmem`
- Keep vendors honest w.r.t. MPI-3 one-sided performance.
- Interoperability of OpenSHMEM and MPI.
- In the unlikely event that you have a supercomputer with MPI-3 but not SHMEM...

MPI-3 Details

`MPI_Win` are the objects against which one performs RMA...

```
MPI_Win_create_dynamic(info, comm, &win);
MPI_Win_create(buffer, size, disp, info,
               comm, &win);
MPI_Win_allocate(size, disp, info, comm,
                 &buffer, &win);
MPI_Win_allocate_shared(size, disp, info,
                        comm, &buffer, &win);
```

The symmetric heap is like an implicit window.

Using MPI windows in SHMEM

- Mapping symmetric heap to MPI windows is relatively easy.
- Mapping text+bss+data into MPI windows is OS-specific but otherwise easy.
- Mapping static into MPI windows is very hard (and not currently supported in OSHMPI).

Symmetric heap design

1. Allocate a single window and sub-allocate (standard approach).
2. Create a single dynamic window and attach all symmetric data to it (bad approach).
3. Allocate a window for every sheap allocation (ARMCI-MPI approach).

Only 1 avoids potentially expensive window lookup in every communication operation.

ARMCI usage is bandwidth-oriented and needs flexibility; SHMEM usage is latency-oriented and restrictive.

Implementation Details

```
void __shmem_put(MPI_Datatype type, int typsz, void *trg,
                 const void *src, size_t len, int pe)
{
    enum shmem_window_id_e win_id;
    shmem_offset_t offset;
    __shmem_window_offset(trg, pe, &win_id, &offset);
    if (world_is_smp && win_id==SHEAP) {
        void * ptr = smp_sheap_ptrs[pe] + (trg - sheap_base_ptr);
        memcpy(ptr, src, len*typsz);
    } else {
        MPI_Win win = (win_id==SHEAP) ? shpwin : txtwin;
        int n = (int)len; assert(len<(size_t)INT32_MAX);
        MPI_Accumulate(src, n, type, pe, offset,
                       n, type, MPI_REPLACE, win);
        MPI_Win_flush_local(pe, win);
    }
} /* This is condensed relative to original source. */
```

Implementation Details

```
void shmem_int_put(int *target, const int *source,
                   size_t len, int pe)
{
    __shmem_put(MPI_INT, 4, target, source, len, pe);
}
```

We encode the type size instead of making a function-call lookup in MPI.

We can and will support 64b count (via MPI datatypes) but right now we just assert if count exceeds 32b range.

SHMEM to MPI: Atomic Operations

SHMEM function	MPI function	MPI_Op
shmem_cswap	MPI_Compare_and_swap	-
shmem_swap	MPI_Fetch_and_op	MPI_REPLACE
shmem_fadd	MPI_Fetch_and_op	MPI_SUM
shmem_add	MPI_Accumulate	MPI_SUM

MPI requires two function calls because all RMA communication is nonblocking; we need a flush to complete AMOs.

It is natural to assume subcommunicators will be reused and thus the implementation should cache them; we have a partial implementation of this but don't use it.

Collective Operations - Communicator Setup

```
void __shmem_acquire_comm(int pe_start, int pe_logs, int pe_size,
                          MPI_Comm * comm, int pe_root, int * broot)
{
    if (pe_start==0 && pe_logs==0 && pe_size==shmem_world_size) {
        *comm  = SHCW /* SHMEM_COMM_WORLD */; *broot = pe_root;
    } else {
        MPI_Group strgrp;
        int * pe_list = malloc(pe_size*sizeof(int));
        int pe_stride = 1<<pe_logs;
        for (int i=0; i<pe_size; i++)
            pe_list[i] = pe_start + i*pe_stride;
        MPI_Group_incl(SHGW, pe_size, pe_list, &strgrp);
        MPI_Comm_create_group(SHCW, strgrp, pe_start, comm);
        if (pe_root>=0) /* Avoid unnecessary translation */
            *broot = __shmem_translate_root(strgrp, pe_root);
        MPI_Group_free(&strgrp);
        free(pe_list);
    }
} /* This is condensed relative to original source. */
```

SHMEM to MPI: Collective Operations

SHMEM	MPI
shmem_barrier	MPI_Barrier
shmem_broadcast	MPI_Bcast
shmem_collect	MPI_Allgatherv
shmem_fcollect	MPI_Allgather
shmem_<op>_to_all	MPI_Allreduce(op)

shmem_collect requires an MPI_Allgather on the counts into a temporary buffer prior to the MPI_Allgatherv.

Performance Results - Disclaimer

Do not attribute to malice what can be explained by stupidity.

We tried very hard to use every implementation properly but it is possible that we missed things. In some cases, we were unable to provide the best environment.

e.g. Portals-SHMEM should use XPMEM but we cannot install it.

Implementation effects

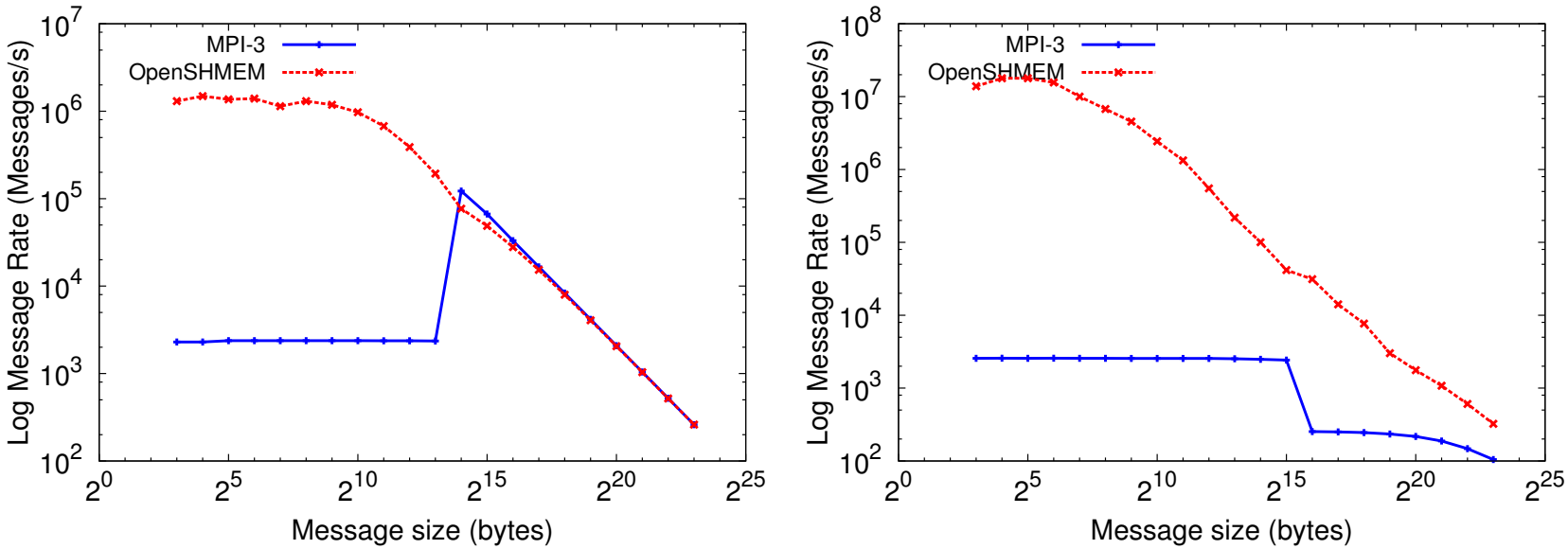


Figure: Internode and intranode (2 PEs) message rate (`Put+long`) with MPI-3 RMA and OpenSHMEM interfaces as implemented with MVAPICH2 and MVAPICH2-X.

Latency - Get

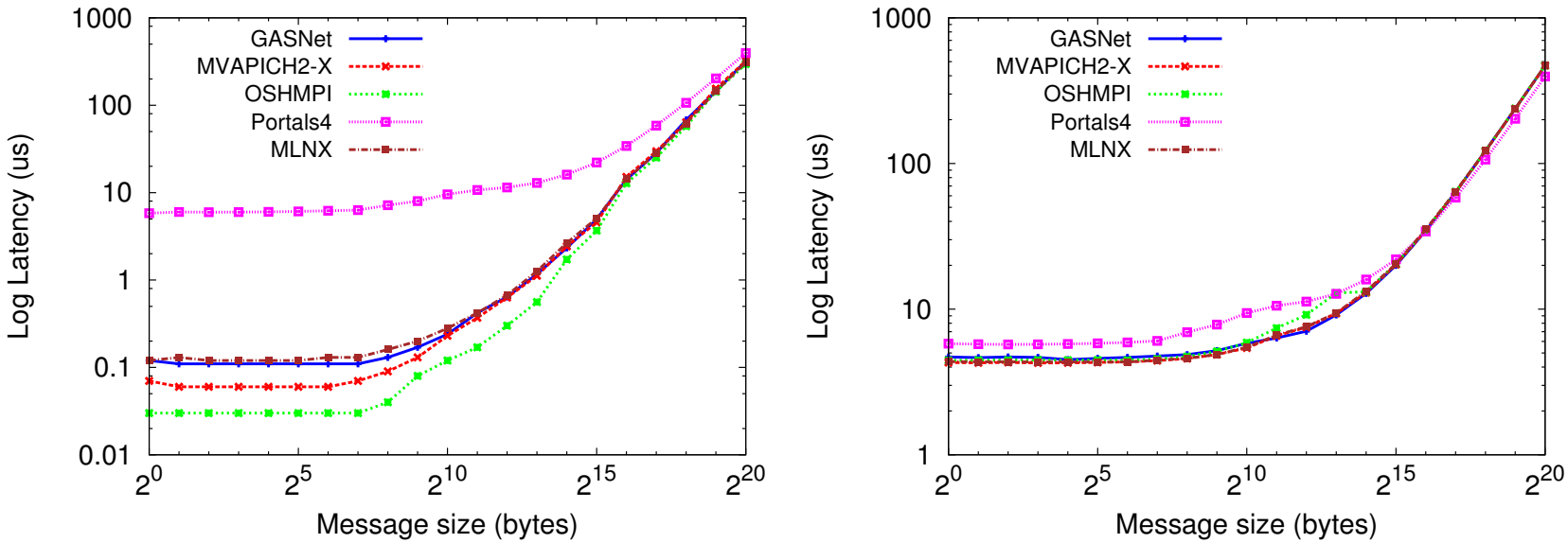


Figure: Intranode (left) and internode (right).

Latency - Put

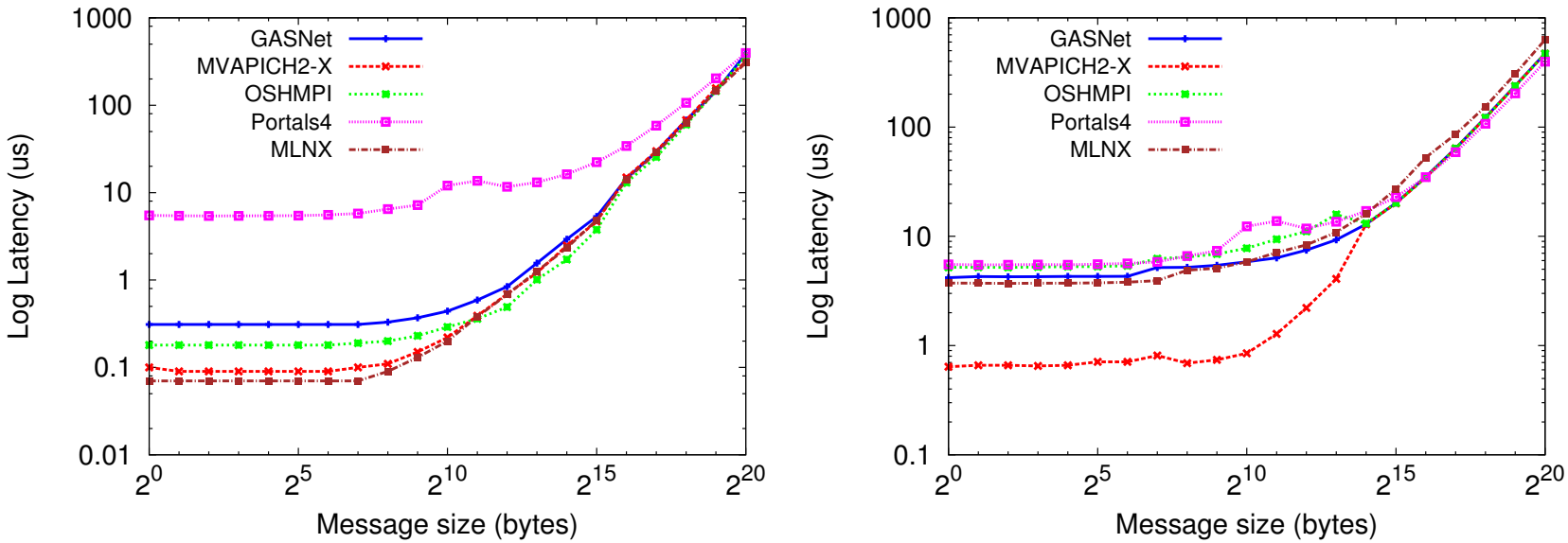


Figure: Intranode (left) and internode (right).

Message Rate - Put

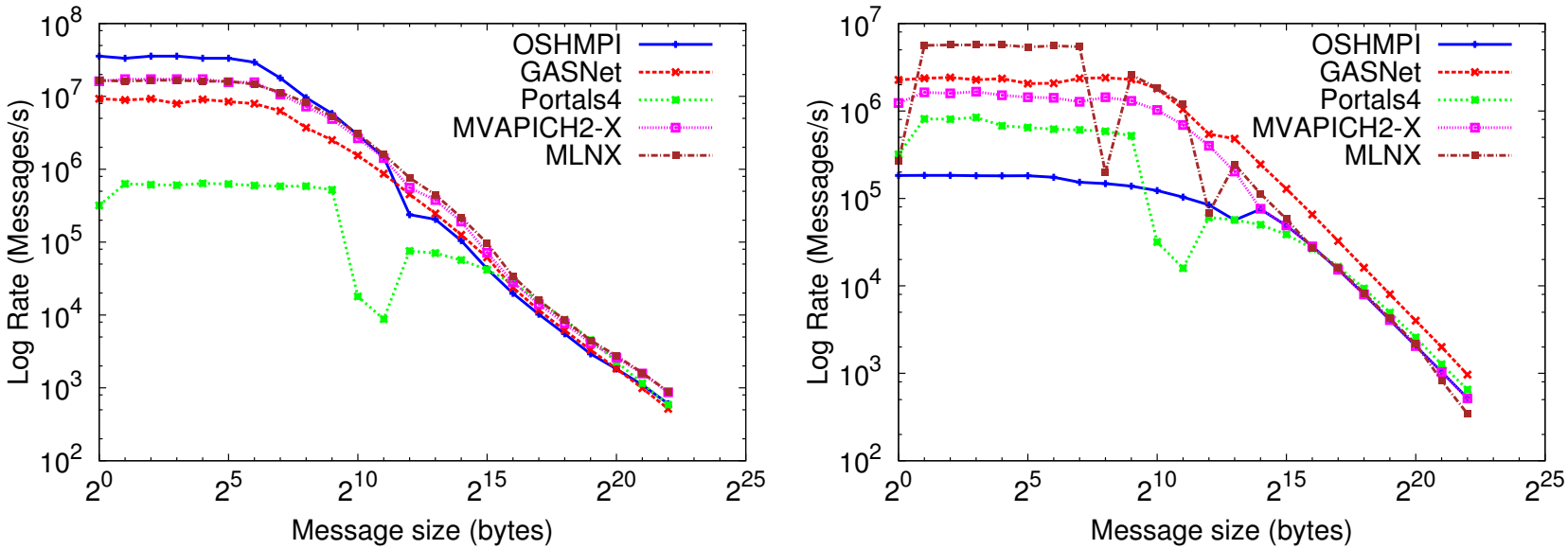
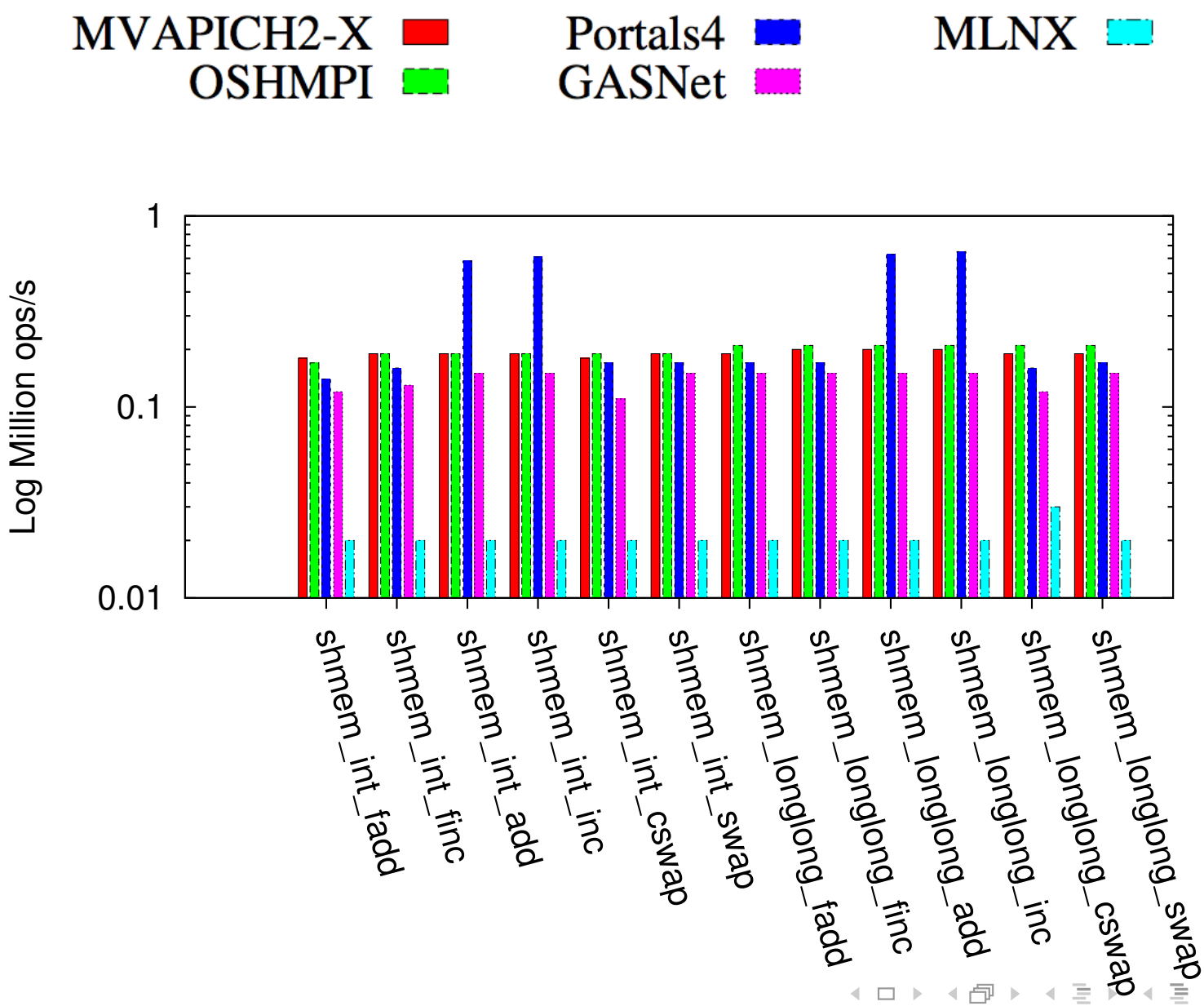


Figure: Intranode (left) and internode (right).

Message Rate - Atomics (internode)



Conclusions and Future Work

- MPI-3 is a reasonable conduit for OpenSHMEM.
- Shared memory performance is (naturally) good.
- MPI implementation quality is (obviously) the limiting factor in internode performance.
- Looking at MPI-3 might help one reason about future extensions to OpenSHMEM.

We would very much like to have users and their feedback.

Software hardening and performance tuning is ongoing.

Acknowledgments

Pavan Balaji and Jim Dinan for MPI-3 expertise.

SHMEM-Portals team (esp. Brian Barrett and Keith Underwood).

Tony Curtis for encouragement.

`https://github.com/jeffhammond/oshmpi`