Hybrid MPI+OpenMP programming
Outline

- Underlying idea
- Hybrid programming styles
- Potential benefits
Hybrid programming

Process
- Independent execution units
- MPI launches N processes at application startup

Thread
- Threads share *memory space*
- Threads are created and destroyed (*parallel regions*)

Hybrid programming
Launch threads *within* MPI tasks
Hybrid programming

- Shared memory programming inside a node, message passing between nodes
- Matches well modern supercomputer hardware
- Usually one MPI task / node, one should experiment with the ratio
int main(int argc, char *argv[]){
    int my_id, omp_rank;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&my_id);
    #pragma omp parallel private(omp_rank)
    {
        omp_rank=omp_get_thread_num();
        printf("I'm thread %d in process %d\n", 
                omp_rank, my_id);
    }
    MPI_Finalize();
}

> aprun -n 4 -d 4 ./hybhello
I'm thread 0 in process 0
I'm thread 0 in process 1
I'm thread 2 in process 1
I'm thread 3 in process 1
I'm thread 1 in process 1
I'm thread 3 in process 0
I'm thread 1 in process 0
I'm thread 2 in process 0
I'm thread 0 in process 2
I'm thread 0 in process 3
I'm thread 1 in process 3
I'm thread 3 in process 3
I'm thread 2 in process 3
I'm thread 1 in process 2
I'm thread 3 in process 2
Hybrid programming styles: fine/coarse grained

- **Fine-grained**
  - Use `omp parallel do/for` on the most intensive loops
  - Possible to hybridize an existing MPI code with little effort and in parts

- **Coarse-grained**
  - Use OpenMP threads to replace MPI tasks
  - Whole (or most of) program within the same parallel region
  - More likely to scale over the whole node
  - Enables all cores to communicate
  - Enables dedicating cores to specific tasks
Hybrid programming styles: communication

- MPI only
- Hybrid MPI+OpenMP
- OpenMP only

- MPI only outside PARALLEL regions (Single)
- MPI also inside PARALLEL regions
- MPI on more than one thread

- MPI only on master thread (Funneled)
- No concurrent MPI calls (Serialized)
- No restrictions (Multiple)
Thread support in MPI

MPI provides an alternative initialization function for hybrid programs

```c
MPI_Init_thread(required, provided)
```

- **required**: requested thread support level
- **provided**: supported thread safety level

- **MPI_THREAD_SINGLE**: Only one thread allowed
- **MPI_THREAD_FUNNELED**: Only master thread allowed to make an MPI call
- **MPI_THREAD_SERIALIZED**: All threads allowed to make MPI calls, but not concurrently
- **MPI_THREAD_MULTIPLE**: No restrictions
Thread support in MPI

```c
int mpi_support;
MPI_Init_thread(&argc,&argv, MPI_THREAD_MULTIPLE, &mpisupport);
if (mpi_support != MPI_THREAD_MULTIPLE)
    printf("MPI_THREAD_MULTIPLE thread support required\n");
```

- Provided thread support may be determined by environment variables or special libraries
- E.g. Cray XT:

```bash
> cc -mp hyb.c -o hyb -lmpich_threadm
> setenv MPICH_MAX_THREAD_SAFETY multiple
> aprun -n 4 -d 4 ./hyb
```
Thread specific *tags* have to be used for distinguishing messages from different threads.

```fortran
!$omp parallel private(mythread, nthreads)
nthrds=OMP_GET_NUM_THREADS()
mythrd=OMP_GET_THREAD_NUM()

if (my_id==0) then
  call mpi_send(mythread, 1, MPI_INTEGER, 1, mythrd, MPI_COMM_WORLD, rc)
else
  call mpi_recv(j, 1, MPI_INTEGER, 0, mythrd, MPI_COMM_WORLD, status, rc)
end if

!$omp end parallel
```
Load balancing

- Good load balance is harder and harder to achieve as the number of MPI processes increases
- Hybrid approach decreases number of processes
- One can dynamically change the number of threads per process - can improve load balance
Reducing messages

Communication inside a node can be replaced by direct memory reads/writes
- Improved throughput and latency
- Decreased overhead from MPI library

In some algorithms the number of messages is reduced

Aggregated messages
- In many (data-parallel) algorithms the messages grow larger as the number MPI processes is decreased
- Increased throughput on inter-node communication
Overlapping computation and communication

While master thread communicate using send/recv other threads can calculate

Can be difficult to utilize properly - load balancing tricky
  – With enough threads, the master thread could be a dedicated communication thread
Improved memory use

The hybrid programming method can be used to decrease memory requirements

- Some algorithms have replicated data in all MPI processes
- In domain decomposition algorithms there are fewer boundary data points

Improved cache usage

- Shared data can reside in shared cache - decreased cache pressure
- OpenMP execution performs blocking
Some benefits of the hybrid approach are algorithm dependent, for example

- Limited parallelism in MPI parallelization
  - Additional levels of parallelism can be easier to implement with hybrid approach
  - Grid-based algorithm only parallelized in one dimension
  - Master-slave algorithms
- Embarrassingly parallel algorithms
  - Can be used to speed up single tasks and/or to increase system size
Parallel I/O

Hybrid approach provides a straightforward realization of the “subset of writers” parallel I/O scheme.

A dedicated thread/core doing I/O for shared data in each node.
All-to-all communication is often bottleneck

- Limits both strong and weak scalability
- Number of communication events increases as $N_{\text{task}}^2$
Hybrid implementation

- Number of communication events decreases as $N_{\text{thread}}^2$
- Size of messages increases
- Speed-up depends on the message size and number of cores but can be at least 4-5 times faster
Disadvantages of hybridization

- Increased overhead from thread creation/destruction
- More complicated programming
- Limited parallelism
- Thread support in MPI and other libraries
Summary

Hybrid programming maps well to modern hardware
- As number of cores inside a node increases, advantages of hybrid approach are likely to become more and more relevant

There are many ways for doing hybridization
- "Fine-grained" vs "coarse-grained"; communication design

In theory, hybrid programming offers several advantages
- In practice, it can be difficult to utilize these
Web resources

A more technical tutorial to hybrid programming
http://www.cac.cornell.edu/education/Training/Intro/Hybrid-090529.pdf