OpenMP
HYBRID
ZOMG
KTHXBAI
(Okay, Thanks, Bye)
LOL
Acknowledgement

- Moses Schwartz
Hybrid Computing

- MPI (Message Passing Interface)
  - Distributed memory
- OpenMP (Open Multi-Processing)
  - Shared memory
MPI

- Advantages
  - Portability (shared and distributed memory machines)
  - Scales beyond one node
- Disadvantages
  - Complex implementation
  - High latency
  - Explicit communication
  - Difficult load balancing
OpenMP

- **Advantages**
  - Ease of implementation
  - High bandwidth, low latency
  - Implicit communication
  - Dynamic load balancing

- **Disadvantages**
  - Only on shared memory machines
  - Scale within one node
  - No specific thread order
  - Possible placement issues
Hardware Considerations?
Hardware Considerations

- Nodes
- CPUs / node
# Job Submission

- `#!/bin/bash`
- `#PBS -q standard`
- `#PBS -l select=8:ncpus=1:node_type=4way`
- `#PBS -l scatter:excl`
- `#PBS -l walltime=8:00:00`
- `#PBS -j oe`

- `cd $PBS_O_WORKDIR`

- `mpirun -np 8 OMP_NUM_THREADS=4 ./a.out`
Hybrid Sample Code

- call MPI_INIT (ierr)
- call MPI_COMM_RANK (…)
- call MPI_COMM_SIZE (…)
- … some computation and MPI communication
- #pragma omp parallel for private(y) schedule(static,chunk)
- for (x = 0; x < MSIZE; x++){
  - for (y = 0; y< MSIZE; y++){
    - … computation
  }
- }
- //End of OMP PARALLEL DO
- … some computation and MPI communication
- call MPI_FINALIZE (ierr)
MPI_Init(&argc, &argv) ; MPI_Comm_Rank(MPI_COMM_WORLD, &mpi_id) ;
// a whole bunch of initializations
#pragma omp parallel
{
    #pragma omp for
    for (I=0;I<N;I++) U[I] = big_calc(I);
    #pragma master
    {
        MPI_Send (U, BUFF_SIZE, MPI_DOUBLE, neigh, tag, MPI_COMM_WORLD);
        MPI_Recv (incoming, count, MPI_DOUBLE, neigh, tag, MPI_COMM_WORLD,
                   &stat);
    }
    #pragma omp barrier
    #pragma omp for
    for (I=0;I<N;I++) U[I] = other_big_calc(I, incoming);
    #pragma omp master
    consume(U, mpi_id);
}
Is Hybrid Worth It?

- MPI alone sometimes better
- Data replication of MPI costly
- Less communication / overhead
- Cache efficiency benefit from shared caches
- Implements well with multiple SMP (Symmetric Multi-Processing)
References

• Parallel Programming in C with MPI and OpenMP, Michael J. Quinn, 2003
• Hybrid MPI and OpenMP Parallel Programming, Rolf Rabenseifner, Georg Hager, Gabriele Jost, Rainer Keller, 2007