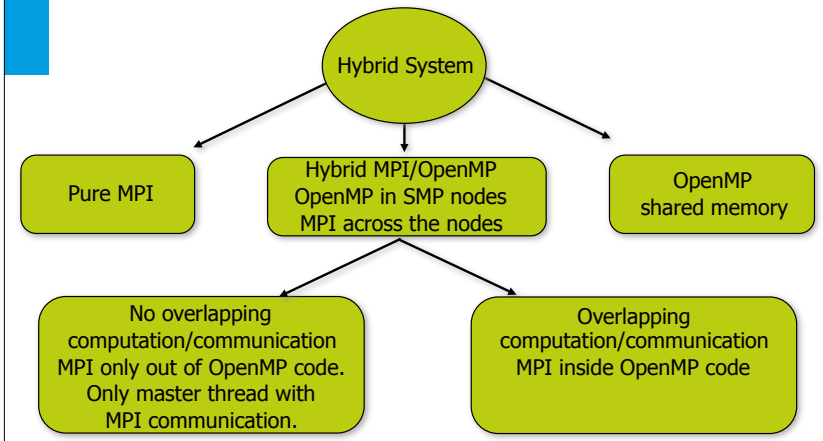
Programming with MPI

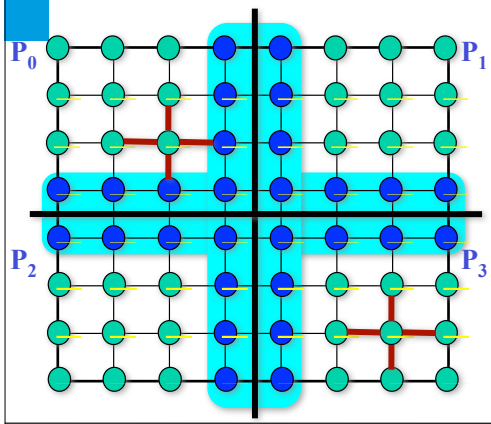
Hybrid MPI + OpenMP

Jan Thorbecke

Hybrid systems programming hierarchy



Overlapping computation/communication: Example



Suppose we wish to solve the PDE

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=f(x,y)$$

Using the Jacobi method: the value of u at each discretization point is given by a certain average among its neighbors, until convergence.

Distributing the mesh to SMP clusters by Domain Decomposition, it is clear that the GREEN nodes can proceed without any comm., while the Blue nodes have to communicate first and calculate later.

MPI/OpenMPI: Overlapping computation/ communication

Not only the master but other threads communicate. Call MPI primitives in OpenMP code regions.

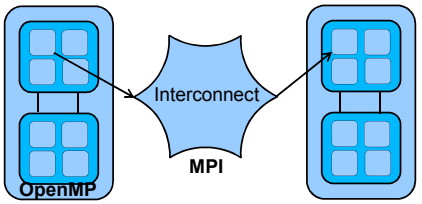
```
if (my_thread_id < # ){
    MPI_… (communicate needed data)
} else
    /* Perform computations that to not need
    communication */
    .
    .
}
/* All threads execute code that requires
    communication */
    .
    .
```

```
for (k=0; k < MAXITER; k++){
    /* Start parallel region here */
    #pragma omp parallel private(){
            my_id = omp_get_thread_num();

            if (my_id is given "halo points")
                    MPI_SendRecv("From neighboring MPI process");
            else{
                    for (i=0; i < # allocated points; i++)
                            newval[i] = avg(oldval[i]);
            }

            if (there are still points I need to do) /* Thi
                    for (i=0; i< # remaining points; i++)
                            newval[i] = avg(oldval[i]);

            }
            for (i=0; i<(all_my_points); i++)
                            oldval[i] = newval[i];
    }
    MPI_Barrier(); /* Synchronize all MPI processes here */
}
```


Hybrid programming

CSC

- Parallel programming model combining:
  - Parallelization over one SMP node with shared-memory parallelization
  - Parallelization over parallel computer with message passing
- Here: MPI + OpenMP



Hybrid programming

CSC

- Hybrid model closer to hardware model of SMP cluster, is it therefore always faster?
  - Not necessarily: in many case/architecture combinations it is slower
  - It is getting increasingly important, however
- Two programming models, "coarse grained" and "fine grained"

Expected benefits

CSC

- 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

Matrix vector OpenMP

```
!$OMP PARALLEL DO &
!$OMP & SHARED(Aloc,x,n,nloc)&
!$OMP & PRIVATE(yloc,i,j) &
!$OMP & REDUCTION(+:yloc)
  do j = 1, n
    do i = 1, nloc
      yloc(i)=yloc(i)+Aloc(i,j)*x(j)
    end do
  end do
!$OMP END PARALLEL DO
```

"Fine-grained"

```
#pragma omp parallel for \
  shared(A,y,x) private(i,j,asum) \
  schedule(guided,chunk)
  for (i=0;i<n;i++){
    asum=0.0;
    for (j=0;j<n;j++)
      asum += A[i][j]*x[j];
    y[i]=asum;
  }
```


Exercise: MatrixVector

- From directory Collectives/MatricVector
  - inserted MPI calls from previous exercise
- Same exercise: in directory MatrixVector
  - programmed more general with send- and recv-counts
  - insert OpenMP directives for local loop (see previous slide)
  - contains 2 OpenMP based solutions
  - check performance running pure MPI and Hybrid

Thread support in MPI

- MPI standard defines four levels of support
  - 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


```
MPI_Init_thread(&argc,&argv,
                MPI_THREAD_MULTIPLE,
                &mpisupport);
 printf("Supports level %d of %d %d %d %d\n",
          mpisupport,
          MPI_THREAD_SINGLE,
          MPI_THREAD_FUNNELED,
          MPI_THREAD_SERIALIZED,
          MPI_THREAD_MULTIPLE);
```

```
> cc -mp hyb.c -o hyb
> aprun -n 4 -d 4 ./hyb
Supports level 0 of 0 1 2 3
> setenv MPICH_MAX_THREAD_SAFETY serialized
> aprun -n 4 -d 4 ./hyb
Supports level 2 of 0 1 2 3
> cc -mp hyb.c -o hyb -lmpich_threadm
> setenv MPICH_MAX_THREAD_SAFETY multiple
> aprun -n 4 -d 4 ./hyb
Supports level 3 of 0 1 2 3
```