PARALLEL COMPUTING IN LATTICE QCD

施羅斯

國立台灣大學

Basic parallel operations

* Setup machine configuration
* Distribute work (MIMD/SIMD)
* Distribute data, collect (gather) data
* Blocking/non-blocking communications
* Reduction operations on data
* Barriers
* (Parallel I/O and/or FS) (I don‘t count that as basic)

PVM

* Supports all basic parallel operations
* Dynamical change machine configuration, heterogeneous clusters, high portability
* Limited support for topology-dependent communication
* Fault tolerance
* Limited use in lattice QCD

MPI

- Supports all basic parallel operations
- Static machine configuration, most massively parallel machines are homogeneous
- Dynamical topologies (powerful, more than we need)
- No fault tolerance
- Support for all major programming languages, available on most machines

MPI-2

- Dynamic process management
  - Cannot be done by most batch systems
- Parallel I/O
- Very complex middleware
- Few programs use all the power (or could possible *need* any of it - Lattice QCD does not)
- Implementations not available on all systems

OpenMP

* Supports only subset of all operations
* Only on shared memory systems (no topologies)
* Does not require changes to source code (except compiler/preprocessor directives)
* Very simple to learn and employ
* Can be combined with PVM or MPI

Special-purpose systems

* Portable libraries (e.g. QMP)
* Special-purpose computers (e.g. Quadrics/APE)
* Lattice QCD uses only subset of com. operations
* Almost always fixed topology, known at compile time
* Communication only of elementary data types
* No IEEE bit compliant operations needed

What we actually need

- Static machine size (sometimes known at compile time)
- Static topology (always known at compile time)
- Nearest-neighbor communication
- Communicate float, double, int (maybe char[] for I/O)
- Reductions (usually only global sums, **but** with high accuracy)
- Possibly utilize non-blocking communication, then we also need barriers

Example “Hello world“

```
c  Fortran example
   program hello
   include 'mpif.h'
   integer rank, size, ierror, tag, status(MPI_STATUS_SIZE)

   call MPI_INIT(ierror)
   call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierror)
   call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierror)
   print*, 'node', rank, ': Hello world'
   call MPI_FINALIZE(ierror)
   end
```

But: Lack of synchronization of output

Commands we need

- Cannot go into too much detail, list of commands:
  - MPI_init / MPI_finalize
  - MPI_comm_size / MPI_comm_rank
  - MPI_(i)send/(i)recv/(test)/sendrecv / MPI_barrier
  - MPI_scatter/gather / MPI_bcast
  - (MPI_type_*/MPI_type_commit)

Optimization

- Latency
- Data rate
- Minimize # of communications or amount of data?
- Topology & machine size dependent
- Machine (hardware) dependent

Summary

- Parallel computing introduces new paradigms
- Takes a little getting used to, but very rewarding
- Learning-by-doing is best
- Different approaches, but similar basic concepts
- MPI well-suited for lattice QCD, only limited set of commands needed