MPI Implementation

Kai Shen

Message Passing Interface

- De facto standard programming interface for message passing-based parallel programs
  - No shared memory
- Communications
  - Point-to-point: send/receive
  - Group communications: broadcast, gather, scatter, reduce, barrier

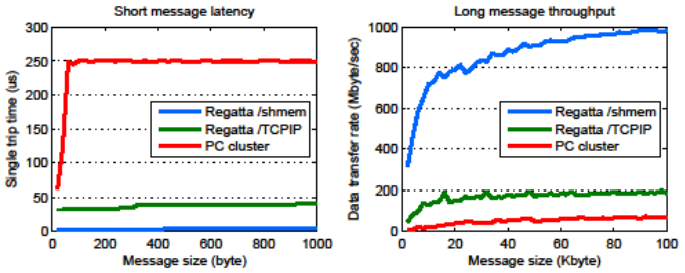
MPI Communications

- Cluster of machines ⇨ TCP/IP
- Performance issues with TCP/IP
  - Connection establishment
  - Congestion control
- UDP or raw IP with some error detection management

MPI Communications in Shared Memory

- MPI communications in shared memory multiprocessors
- TCP/IP
- Shared memory accesses
  - Shared message queue and buffer
  - Synchronization to protect shared accesses
  - Contention may create scalability bottleneck
- Number of memory copying operations
  - two
  - or one?
  - or zero?

Communication Performance



Custom Fast Communications

- Fast local area network (Myrinet, infiniband, …)
  - No need to support Internet communications
- Large multiprocessors from Cray, IBM, …
  - Each processor (or a sub-group of processors) has some local memory
  - Fast access to remote memory through fast system bus
  - No need to support cache coherence, or shared memory consistency

Pipelined Communications

- Heard it from Prof. Chen Ding
- Motivation
  - Large message filled over time at the sender
  - Incrementally useful at the receiver
  - Inefficient to send after the full message is filled
- Pipelined communications
  - Partial message is sent as soon as the data is available
  - When a part of the message arrives, the subsequent computation that only depends on this part of the data can proceed
- Problems
  - At the sender, how do we know that a part of the message is filled?
  - At the receiver, how do we know that a subsequent computation only depends on data that has already arrived?

Collective Communications: Broadcast

- Performance goal:
  - total bandwidth usage
  - latency to reach receivers (worst case or average)
- Many communication system supports native broadcast
  - ethernet broadcast
- Otherwise:
  - root sends to every receiver one by one
  - root sends to one other; both send to two more; then all reached nodes send to some unreached at each round; …
    - does the order matter?

Collective Communications: Reduce

- Approaches
  - all data sent to the root; reduced at root
  - tree-ordered parallel reduction (reduction op is associative)
  - adaptive order based on progress at each node (if op is commutative)

MPI without dedicated processors on Shared Memory Multiprocessor

- 4-processor machine; run 8 MPI processes
- What is going to happen?

- Benefit
  - better utilization despite load imbalance in static task assignment
- Drawbacks:
  - overhead of OS scheduling and context switches (particularly cache pollution)
  - terrible synchronization delays when the MPI implementation block waiting

Parallel Program Reliability

- Checkpointing and restart

- Checkpoint each parallel process
  - Synchronize the parallel checkpointing?

- How about outstanding messages?