Introduction to mpi^2

Rainer Gemulla

May 19, 2011

Message Passing Interface (MPI)

- A standardized API for interprocess communication
- Each process has a rank (0 to $n-1$)
- Message target a (rank, tag)-pair (tags are like ports)
- Example: Send/receive a buffer to (rank, tag)

```
1 MPI_Send(buf, BUFSIZE, MPI_CHAR, rank, tag,
      MPI_COMM_WORLD);
2 MPI_Recv(buf, BUFSIZE, MPI_CHAR, rank, tag,
      MPI_COMM_WORLD, &stat);
```

- boost provides a C++ frontend and serialization

```
1 send(rank, tag, value);
2 recv(rank, tag, &value);
```

mpi^2

- Message Passing Interface $\times$ Max-Planck-Institut
- Library built on top of MPI and `boost`
- Goals
  - Leverage aggregate memory
  - Leverage aggregate computational power
  - Low communication overhead
  - Easy to use for system's programming
  - Provide common primitives
  - Support both synchronous and asynchronous algorithms
- Non-goals
  - Replace MapReduce
- Features
  - Thread management (both spawning and communication)
  - Shared variables (across both threads and ranks)

Main Concepts

- Task
  - A piece of code to run
  - More technically, a function that takes a Channel (and a certain info object)
- Channel
  - Point-to-point communication
  - Thread-to-thread, not rank-to-rank
- TaskManager
  - Spawns (groups of) tasks
  - Sets up communication channels
- Env
  - Manages data stored at each rank
  - Provides remote access

Summary

- Use cases for mpi^2
  - Distributed matrix factorization
  - Distributed probabilistic inference
- mpi^2 users
  - Use mpi^2
  - Test mpi^2
  - Provide feedback
- mpi^2 developers
  - Lots of open issues
  - Help to develop mpi^2