Message Passing Interface
Lecture 4

Manfred Liebmann
Technische Universität München
Chair of Optimal Control
Center for Mathematical Sciences, M17
manfred.liebmann@tum.de

November 30, 2015

One-Sided Communication

Only one process is involved actively in the communication. Put and Get operations must be synchronized with a fence. Data is written to or read from the foreign address space defined by the window.

- One-sided communication
  - MPI_Win_create, MPI_Win_free (window into remote address space)
  - MPI_Win_fence (synchronize)
  - MPI_Put (writes into remote memory)
  - MPI_Get (reads from remote memory)
  - MPI_Accumulate (updates remote memory)

Active target synchronization: MPI_Win_start, MPI_Win_complete and MPI_Win_post, MPI_Win_wait. Post/Start/Complete/Wait synchronization model.

Passive target synchronization: MPI_Win_lock and MPI_Win_unlock. Lock/Unlock synchronization model.

Partitioning of a Vector

Let $a \in \mathbb{R}^N$ be a vector of length N, then the count c_p and the displacement d_p for the MPI rank p, $0 \leq p < P$ for an mostly even distribution of the vector to P processes with the remainder added to the first $N \mod P$ ranks is defined as

$$r := N \mod P, \quad q := \lfloor N/P \rfloor \tag{1}$$

$$c_p := \begin{cases} q+1, & p < r \\ q, & p \geq r \end{cases}, \quad d_p := \begin{cases} pq+p, & p < r \\ pq+r, & p \geq r \end{cases} \tag{2}$$

C/C++ code fragment for the count and displacement calculation:

```
int r = N % P, q = N / P;
int c = q, d = p * q;
c += p < r ? 1 : 0;
d += p < r ? p : r;
```

Determining the Local Vector Index and Rank within a Global Vector

Let $a \in \mathbb{R}^N$ be a global vector of length N with a mostly even distribution of the vector to P processes with the remainder added to the first $N \mod P$ MPI ranks.

Given a global vector index i within the index range $0 \leq i < N$, then the local vector index l_i and rank r_i are given by

$$r := N \mod P, \quad q := \lfloor N/P \rfloor \tag{3}$$

$$l_i := \begin{cases} \lfloor i/(q+1) \rfloor, & i - r < rq \\ \lfloor (i-r)/q \rfloor, & i - r \geq rq \end{cases}, \quad r_i := \begin{cases} i \mod (q+1), & i - r < rq \\ (i-r) \mod q, & i - r \geq rq \end{cases} \tag{4}$$

C/C++ code fragment for the local vector index and rank calculation:

```
int r = N % P, q = N / P;
int loc = i - r < r * q ? i / (q + 1) : (i - r) / q;
int rnk = i - r < r * q ? i % (q + 1) : (i - r) % q;
```

MPI Parallel File I/O

```
#include <mpi.h>
int main(int argc, char** argv)
{
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    MPI_File fh;
    int n = 733, r = n % size, q = n / size;
    int len = rank < r ? q + 1 : q;
    int off = rank < r ? q * rank + rank : q * rank + r;
    int buf[len];

    MPI_File_open(MPI_COMM_WORLD, "data.bin", MPI_MODE_CREATE | MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
    MPI_File_write_at(fh, off * sizeof(int), buf, len, MPI_INT, MPI_STATUS_IGNORE);
    MPI_File_close(&fh);

    MPI_File_open(MPI_COMM_WORLD, "data.bin", MPI_MODE_RDONLY, MPI_INFO_NULL, &fh);
    MPI_File_read_at(fh, off * sizeof(int), buf, len, MPI_INT, MPI_STATUS_IGNORE);
    MPI_File_close(&fh);

    MPI_Finalize();
    return 0;
}
```

MPI Parallel File I/O

- MPI-IO functions
  - MPI_File_open
  - MPI_File_close
  - MPI_File_seek
  - MPI_File_set_view
  - MPI_File_get_view
  - MPI_File_read
  - MPI_File_write
  - MPI_File_read_at
  - MPI_File_write_at
  - MPI_File_read_shared
  - MPI_File_write_shared

The MPI-IO functions enable parallel file I/O to read and write data files.

Network Technology: Ethernet

Networking technology for local area networks and metropolitan area networks. Commercially introduced in 1980 and standardized in IEEE 802.3 for wired connections and in IEEE 802.11 for wireless connections. *Ethernet is the foundation of the Internet.*

- Ethernet
  - Socket based programming
  - Internet protocol suite: TCP/IP
    - Link layer: ARP, NDP, OSPF, L2TP, PPP, MAC, ..
    - Internet layer: IP (IPv4, IPv6), ICMP, ...
    - Transport layer: TCP, UDP, ...
    - Application layer: DHCP, DNS, FTP, HTTP, IMAP, LDAP, NTP, POP, RTP, ..
  - Speeds: 10/100Mb/s, 1/10Gb/s (Copper cable)
  - IPoIB (IP over InfiniBand): 10/40/56/100 Gigabit Ethernet (Fiber cable)