CSCI-UA.0480-003
Parallel Computing

Lecture 11: MPI: Last Touch

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

Many slides of this lecture are adopted and slightly modified from:
- Gerassimos Barlas
- Peter S. Pacheco

Questions

Suppose we have p processes, and we need to compute a vector sum. If we ignore the I/O time, can we get more than p speedup over sequential version?

Questions

Assume we have p processes and we need to implement a binary tree search. Can we get more than p speedup, also ignoring I/O delay?

The Communicator(s)

- We are familiar with the communicator MPI_COMM_WORLD
- A communicator can be thought of a handle to a group of an ordered set of processes
- For many applications maintaining different groups is appropriate
- Groups allow collective operations to work on a subset of processes

MPI_Comm_split

```
int MPI_Comm_split(
    MPI_Comm comm,
    int          color,
    int          key,
    MPI_Comm * newcomm);
```

Called by all processes in comm

Must be non-negative

Rank of the process in newcomm

The original communicator does not go away!

MPI_Comm_split

- Partitions the group associated with comm into disjoint subgroups
- Processes with the same color will be in the same group
- Within each subgroup, the processes are ranked in the order defined by the value of the argument key
  - with ties broken according to their rank in the old group

MPI_Comm_split

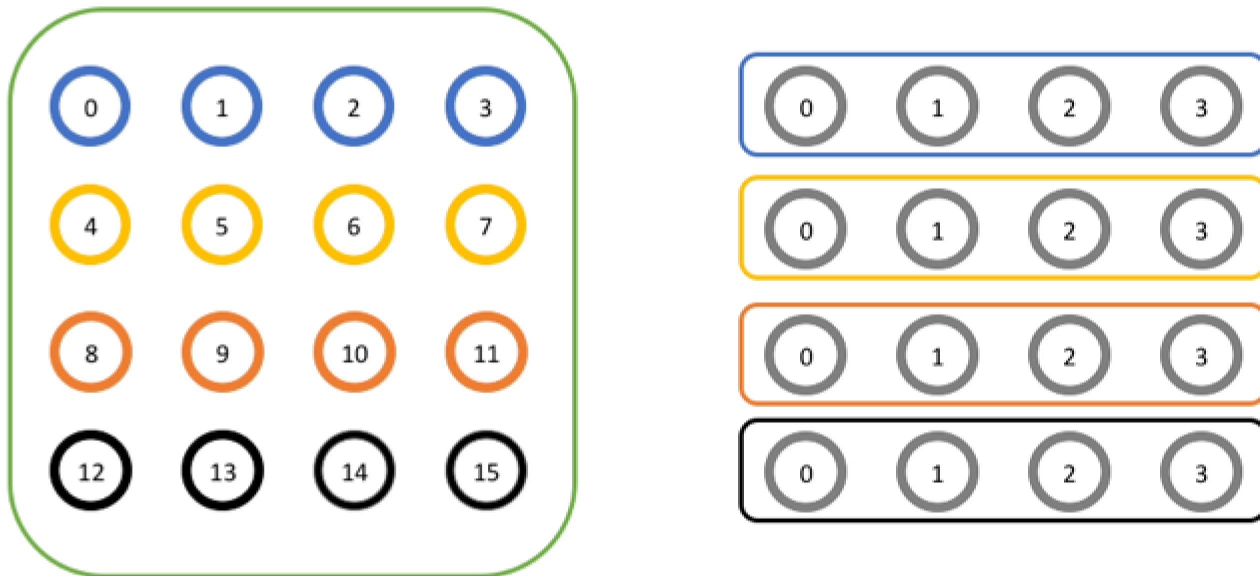
- If a process uses the color MPI_UNDEFINED it won't be included in the new communicator.

MPI_Comm_free

```
int MPI_Comm_free(
  MPI_Comm * newcomm);
```

- Deallocation of created communicator
- Better do it if you are not using the comm again.

Example



Source: http://mpitutorial.com/tutorials/introduction-to-groups-and-communicators/

Example

```
// Get the rank and size in the original communicator
int world_rank, world_size;
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
int color = world_rank / 4;

// Determine color based on row
// Split the communicator based on the color and use the
// original rank for ordering
MPI_Comm row_comm;
MPI_Comm_split(MPI_COMM_WORLD, color, world_rank, &row_comm);

int row_rank, row_size;
MPI_Comm_rank(row_comm, &row_rank);
MPI_Comm_size(row_comm, &row_size);
printf("WORLD RANK/SIZE: %d/%d \t ROW RANK/SIZE: %d/%d\n",
      world_rank, world_size, row_rank, row_size);

MPI_Comm_free(&row_comm);
```

Source: http://mpitutorial.com/tutorials/introduction-to-groups-and-communicators/

Example

Output:

```
WORLD RANK/SIZE: 0/16     ROW RANK/SIZE: 0/4
WORLD RANK/SIZE: 1/16     ROW RANK/SIZE: 1/4
WORLD RANK/SIZE: 2/16     ROW RANK/SIZE: 2/4
WORLD RANK/SIZE: 3/16     ROW RANK/SIZE: 3/4
WORLD RANK/SIZE: 4/16     ROW RANK/SIZE: 0/4
WORLD RANK/SIZE: 5/16     ROW RANK/SIZE: 1/4
WORLD RANK/SIZE: 6/16     ROW RANK/SIZE: 2/4
WORLD RANK/SIZE: 7/16     ROW RANK/SIZE: 3/4
WORLD RANK/SIZE: 8/16     ROW RANK/SIZE: 0/4
WORLD RANK/SIZE: 9/16     ROW RANK/SIZE: 1/4
WORLD RANK/SIZE: 10/16    ROW RANK/SIZE: 2/4
WORLD RANK/SIZE: 11/16    ROW RANK/SIZE: 3/4
WORLD RANK/SIZE: 12/16    ROW RANK/SIZE: 0/4
WORLD RANK/SIZE: 13/16    ROW RANK/SIZE: 1/4
WORLD RANK/SIZE: 14/16    ROW RANK/SIZE: 2/4
WORLD RANK/SIZE: 15/16    ROW RANK/SIZE: 3/4
```

Source: http://mpitutorial.com/tutorials/introduction-to-groups-and-communicators/

MPI_Comm_dup

```
int MPI_Comm_dup(
  MPI_Comm comm,
  MPI_Comm * newcomm);
```

- Creates an exact copy of comm in newcomm

Groups and Communicators

- In reality, processes are ordered in groups
- Communicators are the mean by which processes communicate
- A process can belong to more than one group, with different rank in each.
- … But we will not got deeper than that here!

Words of Wisdom!

Don't Forget!

- MPI is a library
  - → Any MPI operation requires one or more function calls.
  - → Not very efficient for very short data transfers.
  - → Communication should be aggregated as much as possible.
- Avoid unnecessary synchronizations.

When to use MPI

- Portability and Performance
- Irregular Data Structures
- Building Tools for Others
  - Libraries
- Need to Manage memory on a per process basis

When *not* to use MPI

- Programs that have irregular communication patterns are often difficult to express in MPI's message-passing model.
- Domain-specific applications with an API tailored to that application
- Require Fault Tolerance

Strengths of MPI

- **Small**
  - Many programs can be written with only 6 basic functions
- **Large**
  - MPI's extensive functionality (MPI-1 contains about 125 API, let alone MPI-2 and MPI-3)
- **Scalable**
  - Point-to-point communication
- **Flexible**
  - Don't need to rewrite parallel programs across platforms

Conclusions

- You now know enough to use MPI in many problem solving
- We have not studied all APIs though.
- It is fairly easy to understand the rest of APIs.
- The main rules:
  - Reduce communication
  - Ensure load-balancing
  - Increase concurrency