CSCI 251: Concepts of Parallel and Distributed Systems

Alvin Lin

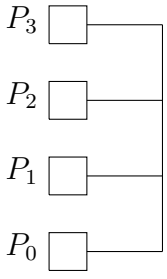
September 25th, 2017

Topics

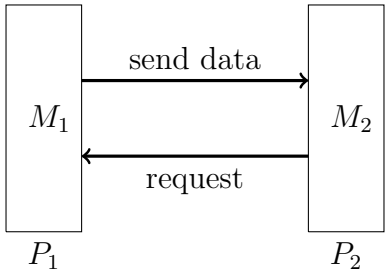
- Message Passing
- Distributed Memory
- Send/receive
- Exercise problem

Message Passing

Every processor has its own memory and the processors may be connected together. They may be connected via a local area network. Every processing unit has its own local array. Memory is distributed.

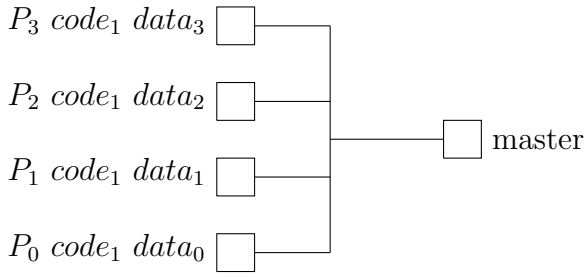


Processes executing on different processors cannot directly access other processors, they must request data from other processors. There is communication and coordination among processes executing in different physical processing units.

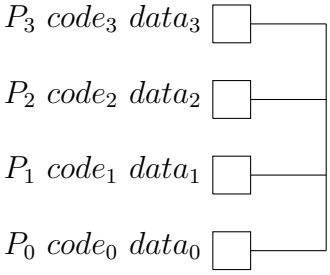


There are two paradigms for data distribution:

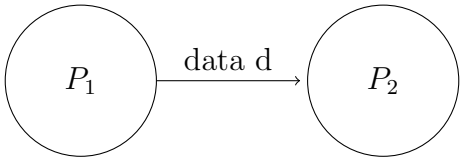
- **Single Program Multiple Data** (SPMD): Identical programs run on all the different processors on different data values. In the early days, SPMD was called SIMD (Single Instruction Multiple Data).



- **Multiple Program Multiple Data** (MPMD): Each processor runs a separate program. MPMD does not need a master process to divide and handle the data.



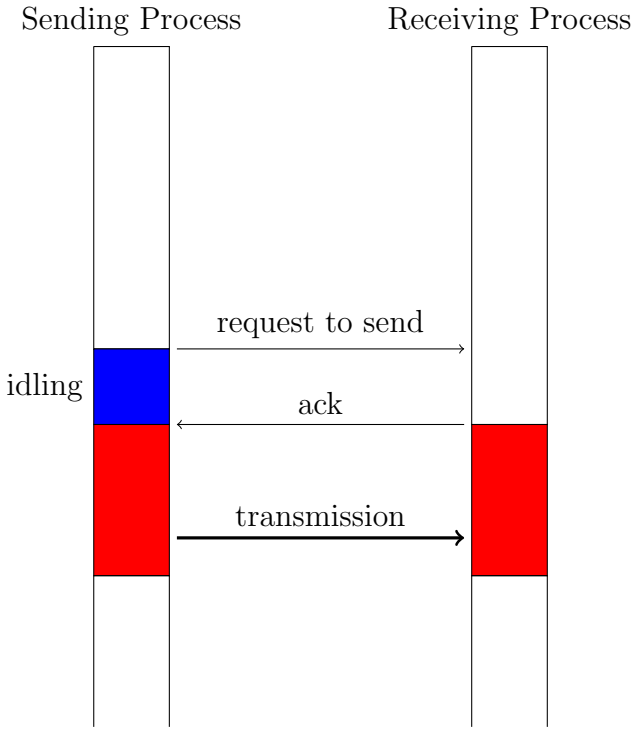
Send and Receive



Suppose in the above example, data d is generated by processor P_1 and sent to P_1, which requires the data before it can proceed. There are four types of data sending which we are concerned with:

- **Blocking and non-buffered**
- **Blocking and buffered**
- **Non-blocking and non-buffered**
- **Non-blocking and buffered**

As the name suggests, blocking means that the execution is blocked until the send and receive is finished. Buffered means that there is a separate memory segment designated for holding the data when it is created by process P_1.



DMA

"**Direct memory access** (DMA) is a feature of computer systems that allows certain hardware subsystems to access main system memory (Random-access memory), independent of the central processing unit (CPU).

Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU first initiates the transfer, then it does other operations while the transfer is in progress, and it finally receives an interrupt from the DMA controller when the operation is done. This feature is useful at any time that the CPU cannot keep up with the rate of data transfer, or when the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer. Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards and sound cards. DMA is also used for intra-chip data transfer in multi-core processors. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without DMA channels. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel." -*Wikipedia*

Send and Receive Building Blocks

```
send(void* sendbuf, int n_elems, int dest);
receive(void* recvbuff, int n_elems, int source);
```

where `sendbuf` is a pointer to a buffer at the sender's memory that stores the data to be sent, `recvbuf` is a pointer to the buffer at the receiver's memory that stores the received data, `n_elems` are the number the data elements, and `dest` and `source` are identifiers.

Reminders

The midterm is on October 11th. Refer to MyCourses for details on Project 1.

Professor Mohan Kumar:
`mjkvcs@rit.edu`
`https://cs.rit.edu/~mjk`

Rahul Dashora (TA):
`rd5476@mail.rit.edu`

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech