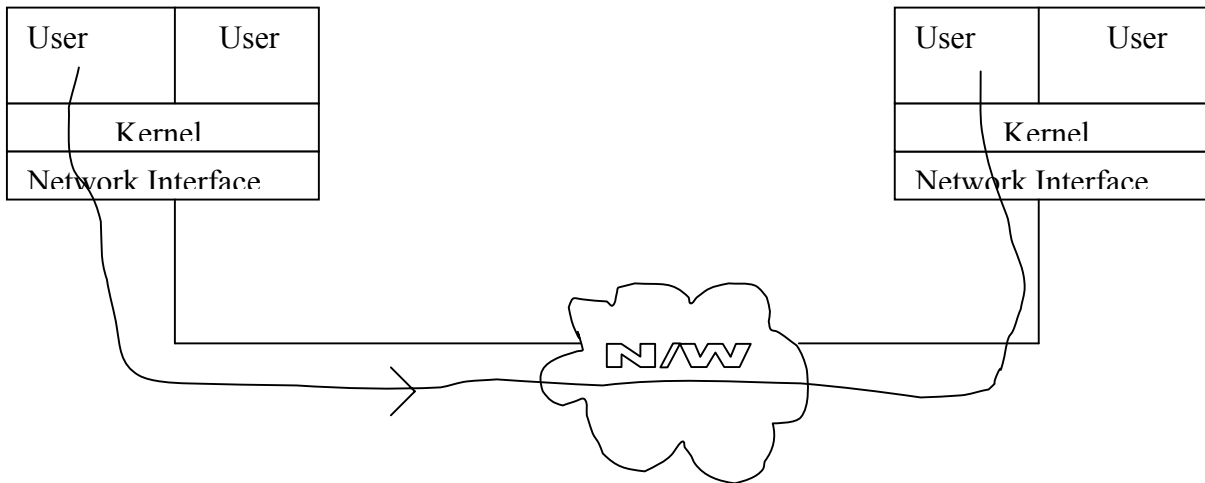


CS8803j - High Performance Communication
18th Feb 2002

Network Interfaces:



Network Interface Issues:

- Metrics:

Infinite Bandwidth, 0 latency → not achievable

- Bandwidth is an important parameter when the message is large.
 - Latency is important for small messages.
 - Buffering in Network interfaces ?
 - DMA; DMA vs VM.
 - Notification : how do you notify that some message just arrived.
 - Mux / demux
 - Protocol accelerators.
 - Verification.
-

Message model:

1. RPC:
 - Synchronous
 - Variable data size
 - Driven by functionality rather than performance.
 2. Streams:
 - Big data
 3. Datagrams:
 - Small data
 4. Send/Recv (MPI)
 - Usually synchronous but need not be.
 - Implicit sync.
 5. Remote write
 - Shared memory
 - Small/Big
 - Asynchronous
 6. Active Messages
 - Small messages.
 - Run handler on the recv side.
Handler can decide where to put the msg or do sync if needed.
 7. Remote read
-

Jacobi relaxation

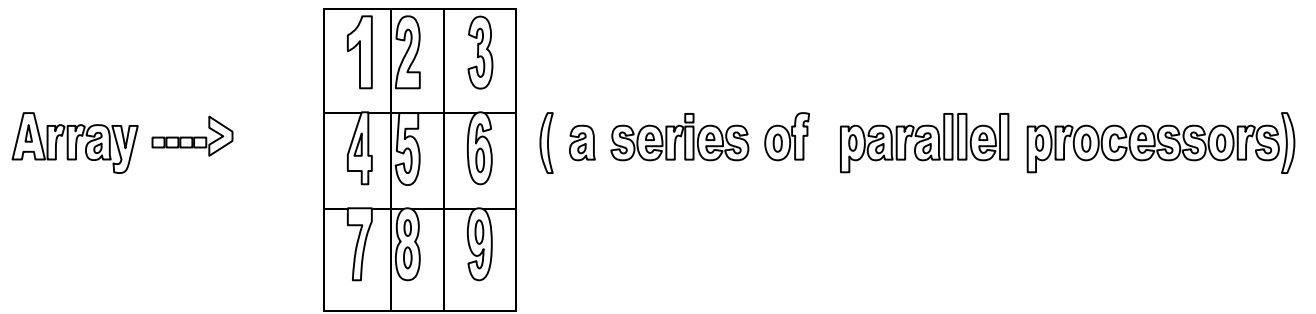
(Not a yoga method!!)

Given: an array

$$X_{i+1} [i,j] = (X_i [i+1,j] + X_i [i-1,j] + X_i [i,j+1] + X_i [i,j-1]) / 4;$$

How to parallelize this?

Using shared memory (easiest way!)

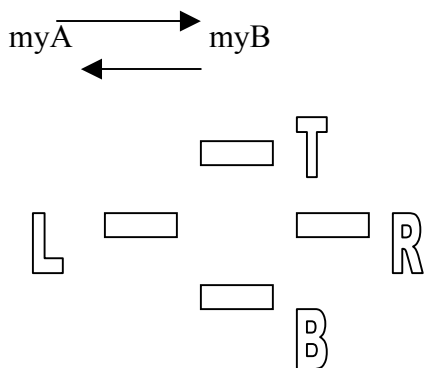


While (1)

```
{  
    subset(a)  $\rightarrow$  subset(b);  
    barrier();  $\rightarrow$  to make sure everybody's is done.  
  
    subset(b)  $\rightarrow$  subset(a);  
    barrier();  
}
```

Using Send/Recv:

Everyone keeps their own tile.



While(1)

```
{  
    a  $\rightarrow$  b (ugly)
```

```
send( North, b – toprow);  
send( South, b – bottomrow);  
send( West, b – leftrow);  
send( East, b – rightrow);
```

```
recv( South, B);           → Sync implicit, buffering implicit.  
recv(North,T);  
recv(East, R );  
recv(West, L);
```

```
b → a (ugly)
```

```
}
```