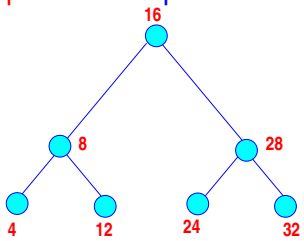
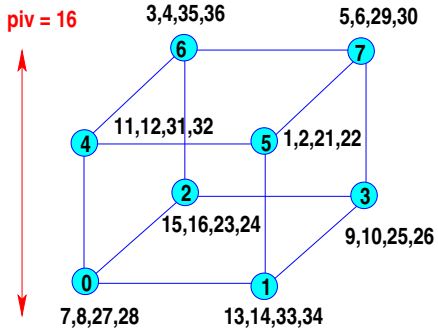
An Illustration: hypercube Quicksort

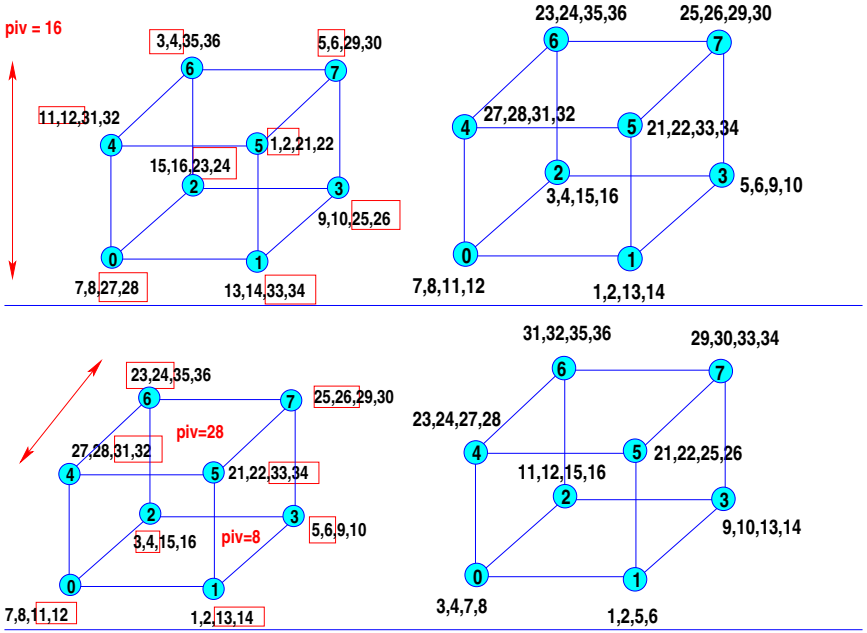
- Goal: to sort n numbers with parallel version of Quicksort.
- Example: The sequence of numbers is: 1,2 ..., 16, 21, 22,..., 36

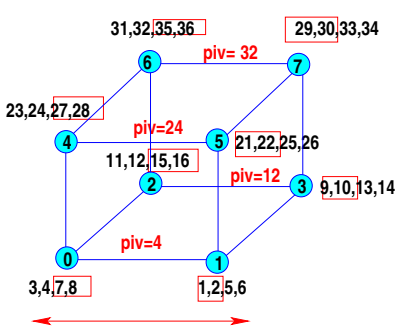
Binary tree of
pre-selected pivots:

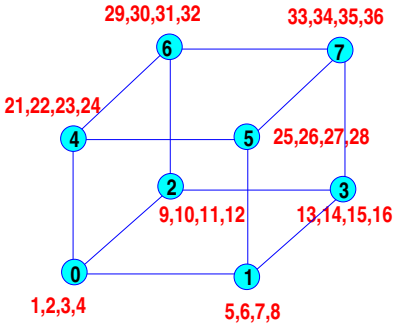




Initial setting







COMMUNICATION OPERATIONS AND MESSAGE PASSING

- Introduction to programming with message passing
- A preview of MPI interface
- Broadcast operations
- All-to-all broadcast and reduction operations
- Scatter and Gather operations
- All-to-all personalized communication

Introduction to message-passing

➤ Need to explicitly code the exchange of messages [data, control,..]

Example: Revisit the sum example seen earlier

Parallel Sum of n numbers

```
  ...
  for  (j=0;  j<p; j++)  { // Parallel Loop
    tmp[j]=0;
//---------- compute partial sums
    for (i=j*m; i<(j+1)*m; i++)
    tmp[j] += + x[i];
  }
//---------- sum-up partial sums
  s=0;
  for (j=0; j<p; j++)  // Sequential loop
    s+=tmp[j];
```


➤ Let "root" = 'master' node where the sum ends up. Recall: $m = n/p$

Parallel sum with communication

```
{  ...
  if (myid == root)  {
    read/ generate array x ;
    for (j=0; j<p && j != root ; j++)
    send x[j*m:(j+1)*m-1] to proc. j
  }
  else {
    receive xloc[1:m] from root ;
  }
  tmp=0;
  for (i=0 ; i< m; i++)
  tmp+= xloc[i];
}
  REDUCE(sum, tmp,'ADD')   //  Reduction oper.
```


➤ REDUCE($sum, tmp,'ADD'$) adds 'tmp' from each PE into 'sum'

➤ Can do reductions with add, multiply, max, min, etc..

➤ More on reductions later.

➤ Next: we will see some of the common communication fucntions used –

➤ On occasion we will see their implementation with MPI

➤ MPI will be covered in more detail later

Communication 'kernels'

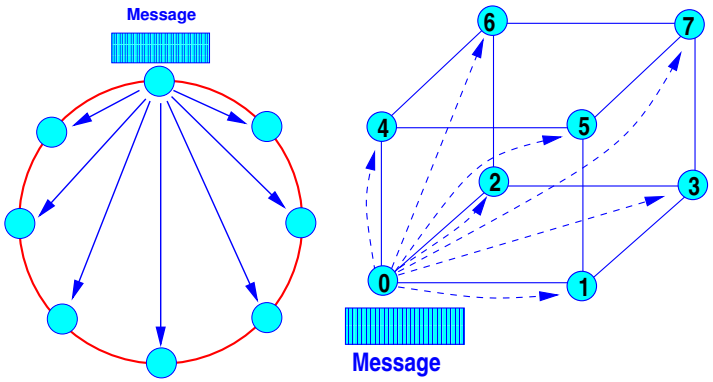
Typical questions addressed:

1. Identify the important communication operations
2. Find effective algorithms for performing these on distributed memory computers
3. Analyze their cost

➤ A by-product: some framework for generic algorithms

Example: Broadcast operation

➤ Sending a message from a 'root' node to all nodes is a Broadcast

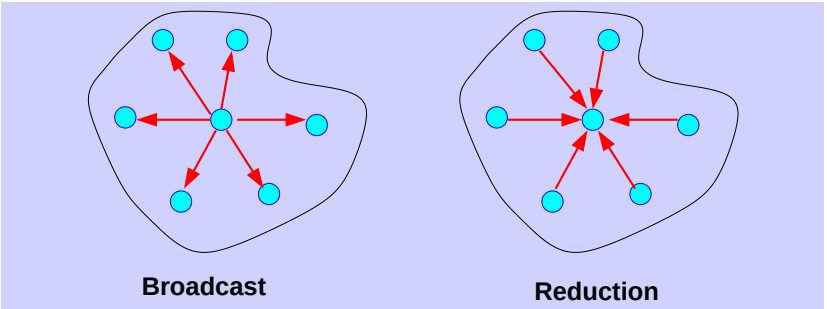


Questions: Best way to broadcast a message from a root node to all others in a ring? In hypercube?

Standard broadcast and reduction operations

➤ Reduction does a global operation (e.g. a sum) on items located on all processors onto a 'root' processor

➤ Can be viewed as a sort of inverse of the broadcast



➤ In parallel sum example, could replace the sends of $x(j*m : (j+1)*m-1)$ from root to all others by a broadcast of all x from root of the vector x. Lines 1 – 6 replaced by:

1. broadcast(x,root)

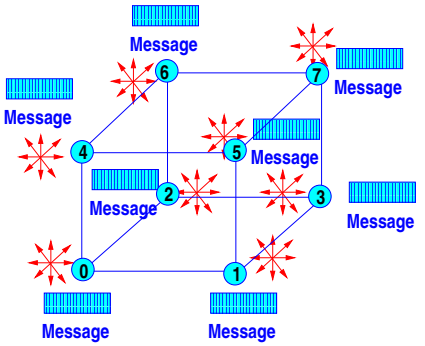
➤ Note however that each PE will get the whole vector.

➤ Corresponding MPI code provided in class web-site

All-to-all broadcast and reduction

➤ All-to-all broadcast can be viewed as p broadcasts, one from each node.

➤ Similarly: All-to-all reduction is a reduction to each node (different for each node).



Note: All-reduce ($\neq$ all-to-all reduce) is a reduction operation in which the result of reduction is available in each processor

➤ All-reduce achievable by a reduce followed by a broadcast [not best way]

➤ Important application of all-reduce: testing if an algorithm has "converged".

Example: Test would be something like:

$$\text{if} \quad \max_{i=0,\ldots,p-1} |x_k^i - x_{k+1}^i| < \quad \text{then stop}$$

➤ Variable i = processor, variable k= iteration number

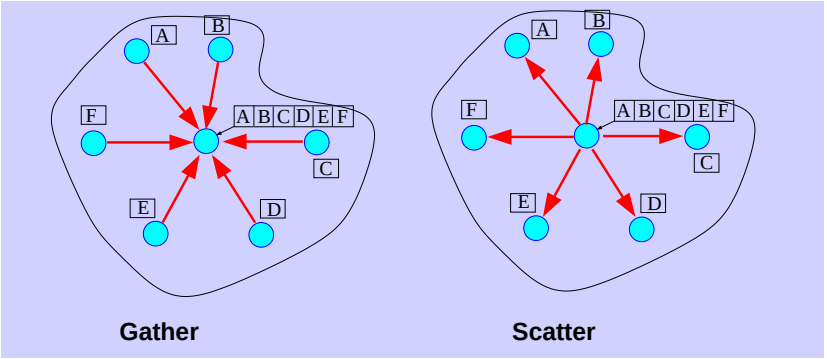
➤ Need to know $\max_i |x_k^i - x_{k+1}^i|$ in each processor.

➤ See text for algorithms on linear array, ring, and hypercubes

Gather and scatter operations

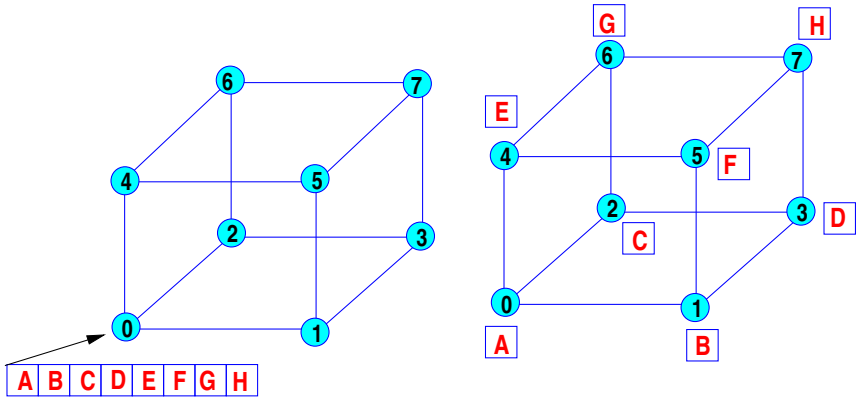
➤ Scatter is similar to a broadcast – but a different item is sent to each processor -

➤ Gather does the inverse operation.



✍1 How would you implement a Scatter operation on a hypercube?

✍2 Cost?



Example: For the parallel sum example – we can "scatter" the subvectors to be summed up in each processors.

➤ In parallel sum algorithm, the lines

```
for(j=0;  j<p && j != root ; j++) {
   send  x[j*m : (j+1)*m-1]  to process  j
else
   receive  xloc[1:m]  from  root  ;
}
```

are replaced by

```
  scatter(x)
```

All-to-All personalized communication

➤ Can be viewed as a scatter from each node: each node sends a distinct message to every other node.

P_0	A_0	A_1	A_2	A_3	$\longrightarrow$	P_0	A_0	B_0	C_0	D_0
P_1	B_0	B_1	B_2	B_3		P_1	A_1	B_1	C_1	D_1
P_2	C_0	C_1	C_2	C_3		P_2	A_2	B_2	C_2	D_2
P_3	D_0	D_1	D_2	D_3		P_3	A_3	B_3	C_3	D_3

➤ Equivalent to p gathers too (0ne to each node)

➤ Notice : operation amounts to transposing a $p \times p$ array!

✍D3 How would you code an all-to-all communication on a hyper-cube?