Colorado State University

CS 575 Parallel Processing
Lecture 5: Ch 4 (GGKK)

Sanjay Rajopadhye
Colorado State University

Basic Communication Ops

- PRAM, final thoughts
- Quiz 3
- Collective Communication
  - Broadcast & Reduction
  - All-to-All Broadcast & Reduction
  - All-Reduce & Prefix Sum
  - Scatter & Gather
  - All-to-All Personalized Communication

PRAM Quiz: Matrix mult

- What is the most work efficient sequential algorithm to multiply two *N*x*N* matrices?
  - Strassen O($N^{2.7}$) (but ignore this)
  - Standard: O(N^3)
- What is the ***fastest*** PRAM algorithm
  - Time = ?
  - Work = ?
- What is the ***fastest*** EREW PRAM algorithm?
  - Time?
  - Work optimality?

Quiz 3

- Classification:
  - EREW < [CREW, ERCW] < CRCW
  - Middle two are incomparable
  - Section 3.2 (relative power of concurrent writes)
- Algorithm question:
  - List Rank is a special case of suffix "sum"

Collective Communications

- Patterns of communication that occur very often in may algorithms
  - e.g., a scatter occurs in (the I/O part of) all pairs shortest path (Ch 6 of the 475 text)
  - Useful to build a library (e.g, MPI collectives)
- Understand the algorithms behind these algorithms on various topologies
- Duality: some patterns are duals of others
  - Reverse the communication
  - Reverse the steps of the algorithm

The Collectives

- Broadcast/reduction
- All-reduce = reduce + broadcast
- All-to-all broadcast or multi-broadcast (same as all-reduce with concatenation)
- Gather = reduction with concatenation
- Scatter = dual of gather
- All-scatter, All-gather and All-to-All

Broadcast and reductions

- 1-to-all & all-to-1
  - Discussion 3
  - Linear array, Ring, meshes & tori, hypercube & fully connected
- Reduction is dual of broadcast
  - already seen in PRAM classification
- All-to-all broadcast
  - Every processor has a distinct data packet that is to be broadcasted – P independent broadcasts
    - but don't want to do them one after the other
- All-reduce (not exactly the dual)
  - Each PE has one data element, they get reduce-ed, and the (single) result is broadcast to everyone
    - but don't want to do them one after the other (but this "optimization" yields only a constant factor savings at best)

Scatter & Gather

What is a scatter?

- *p* processors, P_1, P_2, P_3 … P_p
- P_1 has an array (view as a column vector) of messages, m_1, m_2, m_3 … m_p
- Interpret message subscripts as desired destinations
  - P_1 ends up with m_1
  - P_2 has m_2 … P_p has m_p
- Messages have be "scattered" by P_1 to all the other processors
  - Variant: source may be an arbitrary processor

Scatter & Gather …

- Gather is the dual
  - Initially every processor has a message, and all messages have to end up in processor P_1
- Scatter and gather can be viewed as "transposing:"
  - Make a column vector into a row vector and vice versa
    - rows are message id's/memory locations
    - columns are processors
- All-scatter: each processor does an independent scatter
  - Also called All-to-All, personalized communication
  - P_i has a column vector $[m_{i,1}, m_{i,2}, \ldots m_{i,p}]^T$
  - Two subscripts: first is source, second is destination
  - Think matrix transposition
- So what is an all-gather?
  - An all-scatter? Or a single gather + broadcast

Machine Assumptions

- Distributed memory machine P processors
- Topology is (mostly) important
- Transfer of m words between any pair of processors takes $t_s + mt_w$ time (3 cases)
  - If processors are connected by a link
  - If there is a congestion free path between them (cut-through routing)
  - Topology independent (close to reality, but t_w is now effective transfer rate (network is over-engineered)

The three models

- Topology oblivious
  - Like the PRAM, but distributed memory
  - a processor can send a message of size m to any other processor, provided only one partner
- Topology aware (w cut-through)
  - A processor can send a message to any other provided paths do not conflict
- Topology aware (store & forward)
  - A processor can send a message to any other provided links do not conflict
- Communication time = $t_s + t_w m$

Topology oblivious machines

- Broadcast/reduction
- All-reduce = reduce + broadcast
- All-to-all broadcast or multi-broadcast (same as all-reduce with concatenation)
- Gather = reduction with concatenation
- Scatter = dual of gather
- All-scatter, All-gather and All-to-All

Topology oblivious Broadcast

- d = lg (n) rounds:
  - Assume node 0 is the source
  - In round # i (starting with i=0):
    - 2^i messages (copies)
    - Who sends?
    - To whom?

Round:	0	1	2	3
Sender(s):	{0}	{0, n/2}	{0, n/4, n/2, 3n/4}	{0, n/8, n/4, 3n/8, n/2, 5n/8, 3n/4, 7n/8}
Receiver(s):	{n/2}	{n/4, 3n/4}	{n/8, 3n/8, 5n/8, 7n/8}	

Colorado State University 13

In round # i

- 2^i senders
- For x = 0 … 2^1-1:
  - the x^{th} sender is: $s = \frac{x}{2^i} n$
  - and its receiving partner is $r = s + \frac{n}{2^{i+1}}$

Round:	0	1	2	3
Sender(s):	{0}	{0, n/2}	{0, n/4, n/2, 3n/4}	{0, n/8, n/4, 3n/8, n/2, 5n/8, 3n/4, 7n/8}
Receiver(s):	{n/2}	{n/4, 3n/4}	{n/8, 3n/8, 5n/8, 7n/8}	

Colorado State University 14

Arbitrary Source

What if node *a* was the original sender?

- Re-label the nodes and reuse the same algorithm:
- Some properties of the XOR function:
  - For any constant *c*, let $f(x) = \text{XOR}(x, c)$

Claim: *f* is a bijection

  - Prove it
  - So, what's its inverse?

All Reduce

First do a reduce, then do a broadcast

Can do better (divide and conquer)

- Divide the the machine into two equal halves (each node has a partner in the other half [conceptual step]
- Recursively do all reduce in each half (in //)
- Pairwise exchanges with partner

```
1   for (i=0; i < lg(n); i++)
2       exchange message with XOR(myid, 1<<i);
```

Scatter-Gather and their ilk

- Gather = reduce with concatenation
- Scatter = ?
- What about multi-scatter, multi-gather

All2All Personalized Comm