Parallel Programming

Communication cost

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

WS 18/19

High Performance and Automatic Computing

Collective Communication: Lower Bounds

Cost of communication: $\alpha + n\beta$

Cost of computation: $\gamma \, \#\text{ops}$

$\alpha =$ "latency", "startup" $\quad$ $\beta =$ 1/"bandwidth"

$n =$ size of the message $\quad$ $\gamma =$ cost of 1 flop

$p =$ # of processes $\quad$ $\alpha \gg \beta$

Objective: Establish lower bounds for the cost of the primitives.

Why? A lower bound is a hard reference again which to evaluate algorithms.

Primitive	Latency	Bandwidth	Computation
`Broadcast`			
`Reduce`			
`Scatter`			
`Gather`			
`Allgather`			
`Reduce-scatter`			

- `Broadcast`: The full array (size n) needs to leave the root.
- `Reduce`: The arrays have to be combined.
  Total number of ops = $n * (p-1)$. If perfectly parallel: $n * (p-1)/p$.
- `Scatter`: $(p-1)$ chunks —each of size n/p— have to leave the root.
- `Gather`: $(p-1)$ chunks —each of size n/p— have to reach the root.
- `Allgather`: Since every process ends up in the same condition as that of a `Gather`, the cost is at least that of a `Gather`.
- `Reduce-scatter`: Every process has to send at least $(p-1)$ chunks —each of size n/p— (these are the chunks whose reduction will end up in a different process), and has to receive at least one chunk —of size n/p— (to reduce the local chunk). Since data can be sent and received at the same time, the lower bound is $\frac{(p-1)n}{p} \times \beta$.

Collective Communication: Lower Bounds

Primitive	Latency	Bandwidth	Computation
`Broadcast`	$\lceil \log_2(p) \rceil \alpha$	$n\beta$	-
`Reduce`	$\lceil \log_2(p) \rceil \alpha$	$n\beta$	$\frac{p-1}{p} n\gamma$
`Scatter`	$\lceil \log_2(p) \rceil \alpha$	$\frac{p-1}{p} n\beta$	-
`Gather`	$\lceil \log_2(p) \rceil \alpha$	$\frac{p-1}{p} n\beta$	-
`Allgather`	$\lceil \log_2(p) \rceil \alpha$	$\frac{p-1}{p} n\beta$	-
`Reduce-Scatter`	$\lceil \log_2(p) \rceil \alpha$	$\frac{p-1}{p} n\beta$	$\frac{p-1}{p} n\gamma$

Implementation of Bcast and Reduce

- IDEA: recursive doubling / “Minimum Spanning Tree” (MST)
  At each step, double the number of active processes.

- How to map the idea to the specific topology?
  - ring: linear doubling
  - (2d) mesh: 1 dimension first, then another, then another ...
  - hypercube: obvious, same as mesh

- Cost?
  - # steps: $\log_2 p$
  - cost(step): $\alpha + n\beta$
  - total time: $\log_2(p)\alpha + \log_2(p)n\beta$ lower bound: $\log_2(p)\alpha + n\beta$
  - note: cost(p^2) = 2 cost(p) $\Rightarrow$ not optimal, but not by much

- Reduce
  Bcast in reverse; cost(computation) ?

Implementation of Scatter (and Gather)

- IDEA: MST again
  At step i, only $\frac{1}{2^i}$-th of the message is sent
- # steps: $\log_2 p$
- cost(step$_i$): $\alpha + \frac{n}{2^i}\beta$
- total time: $\sum_{i=1}^{\log_2(p)} \alpha + \frac{n}{2^i}\beta = \log_2(p)\alpha + \frac{p-1}{p}n\beta$
- lower bound: $\log_2(p)\alpha + \frac{p-1}{p}n\beta$ optimal!

A different implementation of Bcast

- IDEA: Scatter + cyclic algorithm (e.g., pass to the right)
- Cost?

Implementation of Allgather (and Reduce-scatter)

- IDEA: "Recursive-doubling" (bidirectional exchange)
  Recursive allgather of half data + exchange data between disjoint nodes.

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]			
	v[1]		
		v[2]	
			v[3]

⇓

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[0]		
v[1]	v[1]		
		v[2]	v[2]
		v[3]	v[3]

⇓

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[0]	v[0]	v[0]
v[1]	v[1]	v[1]	v[1]
v[2]	v[2]	v[2]	v[2]
v[3]	v[3]	v[3]	v[3]

- # steps: $\log_2 p$
- cost(step$_i$): $= \alpha + \frac{n2^{i-1}}{p}\beta$
- total time:

$$\sum_{i=1}^{\log_2(p)} \alpha + \frac{n}{2^i}\beta = \log_2(p)\alpha + \frac{p-1}{p}n\beta$$

- lower bound: $\log_2(p)\alpha + \frac{p-1}{p}n\beta$

Another implementation of Allgather

- IDEA: Cyclic algorithm

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]			
	v[1]		
		v[2]	
			v[3]

$\Downarrow$

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[0]		
	v[1]	v[1]	
		v[2]	v[2]
v[3]			v[3]

$\Downarrow$

$Node_0$	$Node_1$	$Node_2$	$Node_3$
v[0]	v[0]	v[0]	
	v[1]	v[1]	v[1]
v[2]		v[2]	v[2]
v[3]	v[3]		v[3]

- # steps: $p-1$
- cost(step$_i$): $\alpha + \frac{n}{p}\beta$
- total time:

$$\sum_{i=1}^{p-1} \alpha + \frac{n}{p}\beta = (p-1)\alpha + \frac{p-1}{p}n\beta$$

- lower bound: $\log_2(p)\alpha + \frac{p-1}{p}n\beta$