

Simulations between Models

BSP – LogP

Situation & Questions

- BSP (g, l, p) nicer to program
- LogP (L, O, G, P) more realistic when comparing measurements on real machines
- Can we find simulations or even translations between the models?
- Are there principle (regarding computational complexity) differences between them?
- What's the delay of simulations.

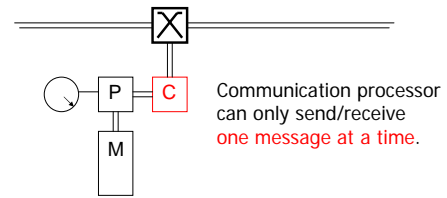
2

Discussion of LogP (Bilardi et al.)

- $G \geq O$
 - Assumption w.l.o.g., if $G < O$ we could safely set $G = O$ since only every O time units a message is produced/consumed
- $G \geq 2$
 - Assume $G = 1$ and $\lceil L/1 \rceil = L$ messages sent simultaneously to a single processor
 - Should be delivered after (at most) L time units as capacity constraints are not violated
 - Requires the delivery of individual the messages after 1, 2, 3, ..., L time units and, hence, routing from a message in 1 time unit
 - $G \geq 2$ instead would lead to deliver times at least $\Theta(L)$
- $L \geq G$
 - Assumption $G > L$ could lead to incoming message rate $> 1/G$
 - $L \geq G$ permits bounded input buffer size

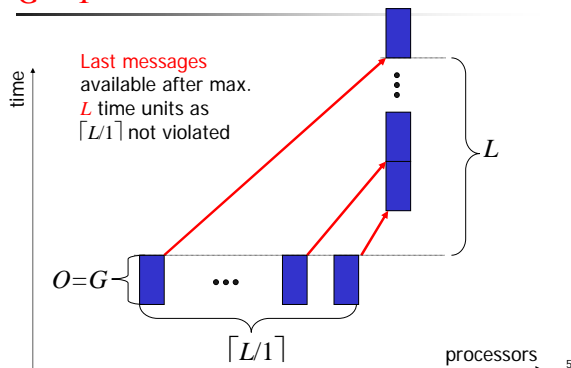
3

Congestion



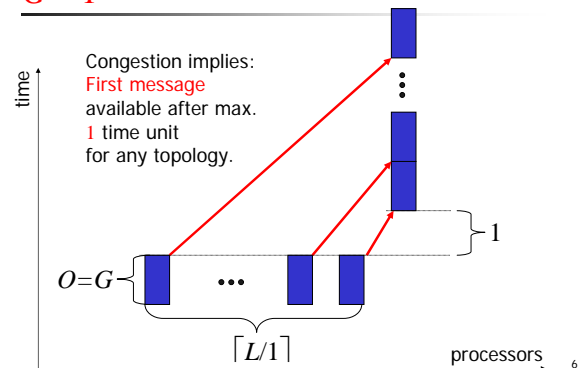
4

$G = 1$



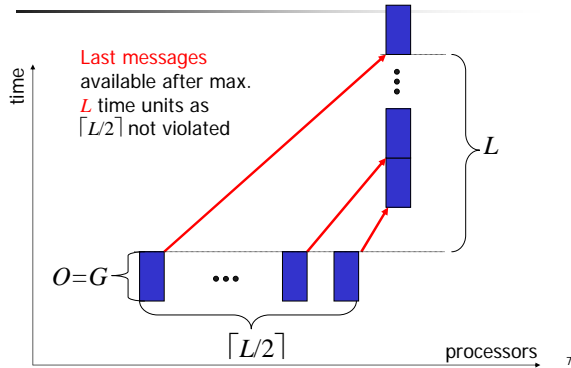
5

$G = 1$

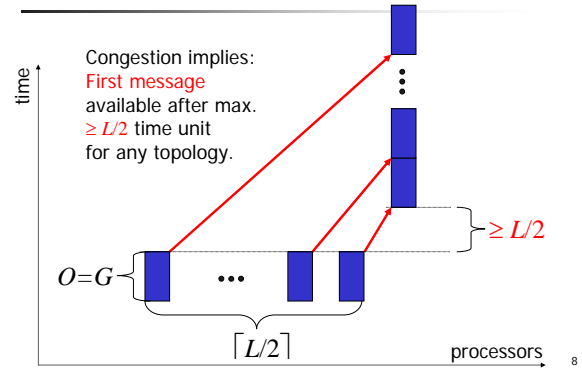


6

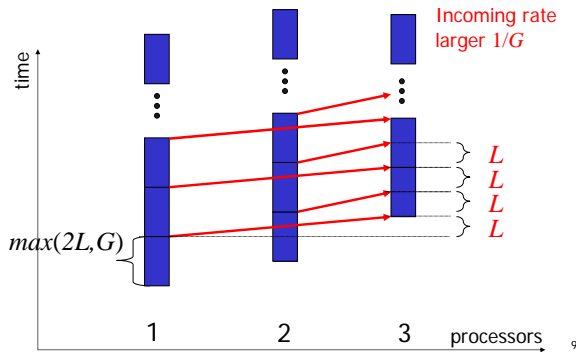
$G \geq 2$



$G \geq 2$



$G > L$



Stall free LogP on BSP (Bilardi et al.)

- Each super-step simulates cycles of $L/2$ consecutive LogP instructions
- Communication is postponed to end of cycle
- Then routing of a $h \leq \lceil LG \rceil$ relation
 - No more than $\lceil L/2 / G \rceil \leq \lceil LG \rceil$ messages sent by a processor (maximum send rate in $L/2$ steps)
 - No more than $\lceil LG \rceil$ messages sent to a processor
 - All messages sent within cycle of $L/2$ are still in transit at its end
 - Cannot be more than $\lceil LG \rceil$ in a stall free program
- For a super-step: $T \leq L/2 + \lceil LG \rceil g + l$
- Slowdown: $O(1 + g/G + l/L)$ i.e. constant if $l = \Theta(L)$ and $g = \Theta(G)$

10

Comparison BSP and LogP (revisited)

- IBM SP-2, small messages (< 16 bytes), $P=16$
 - BSP: $l = 502 \mu\text{s}$ $g = 30.1 \mu\text{s}$
 - LogP: $L = 17.1 \mu\text{s}$ $O = 9.0 \mu\text{s}$ $G = 9.8 \mu\text{s}$

BSP on LogP (Bilardi et al.)

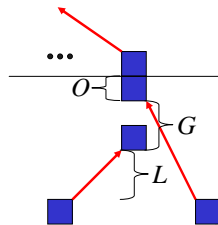
- Each super-step is simulated
 - Local computations on LogP processors
 - Routing the h -relation
 - Synchronization of all LogP processors to start the next super-step (or terminate computations, otherwise)
- How to synchronize LogP processors?
- How to route the h -relation?

11

12

Synchronize LogP Processors

- Synchronize = Combine & broadcast
 - Use inverse and direct optimal broadcast
 - No closed expression of time bound in L, O, G, P
- Binary tree:
 - Heights: $\log P$
 - Time per step: $2O+G+L$
- Synchronization bound: $T_{\text{sync}} = 2(2O+G+L) \log P$



Comparison BSP and LogP (revisited)

- IBM SP-2, small messages (< 16 bytes), $P=16$
 - BSP: $l = 502 \mu\text{s}$ $g = 30.1 \mu\text{s}$
 - LogP: $L = 17.1 \mu\text{s}$ $O = 9.0 \mu\text{s}$ $G = 9.8 \mu\text{s}$
- LogP: Optimal broadcast $T_{\text{OptBC}} \leq 100 \mu\text{s}$
- LogP: Binary tree broadcast $T_{\text{BinBC}} \leq 220 \mu\text{s}$
- LogP: Synchronization $T_l \leq 440 \mu\text{s}$

14

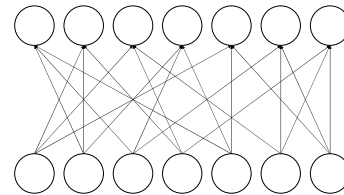
Routing the h -relation

(Eisenbiegler, Löwe, Zimmermann)

- Send at: $0, G, 2G \dots (h-1)G$
- Compute the communication graph
 - Bipartite graph $H=(U,V,E)$
 - $(u,v) \in E$ iff processor u sends to v in that h -relation
 - Degree of each node is at most h
- Compute edge coloring of graph H
 - For bipartite graph in at most $O(|E| \log(U+V))$
 - Guarantees a h -coloring
 - Offline for oblivious predictable communications
- Same color same sending time

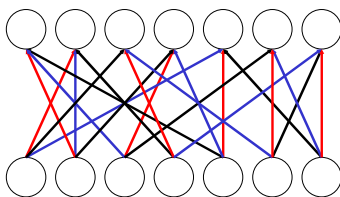
15

Example: routing a 3-relation



16

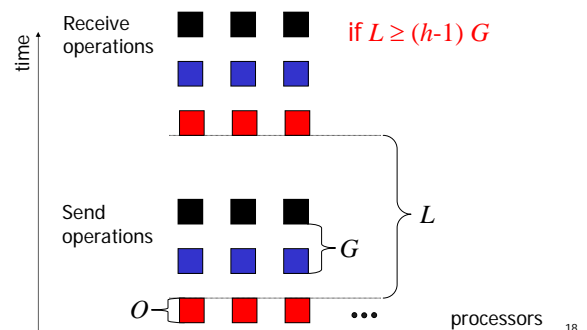
Coloring of the 3-relation



No 2 edges of same color start / end in same node

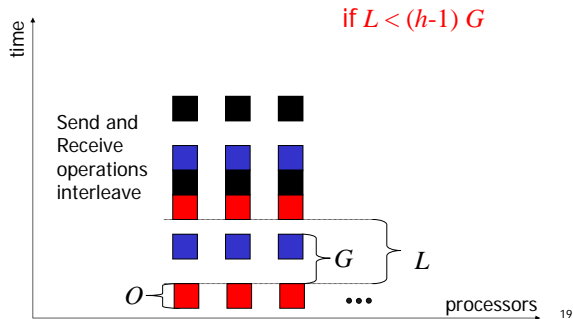
17

Routing the 3-relation



18

Routing the 3-relation



19

Routing oblivious h -relation

- If $L \geq (h-1)G$
 $L + 2O + (h-1)G$
- If $L < (h-1)G$
 $2O + 2(h-1)G$
- Approx: $\max(L, (h-1)G) + 2O + (h-1)G$
- Assumes that
 - $h \leq \lceil L/G \rceil$
 - h -relations are known in advance (oblivious)
- In general
 - It could be that $h > \lceil L/G \rceil$
 - h -relations are data dependent (non-oblivious)

20

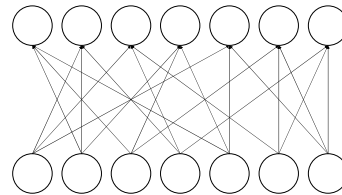
Routing general h -relation

(Bilardi et al.)

1. Compute the max. number r of received messages, broadcast it to all processors
2. Sort all messages by destination and compute a *rank* of each message
3. Compute the max. number s of send messages, broadcast it to all processors
4. $h = \max(r, s)$, send messages in routing cyclic ($\text{rank mod } h$) + 1

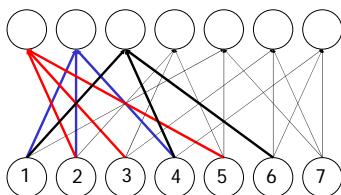
21

Example: routing a 3-relation



22

Sorting the messages

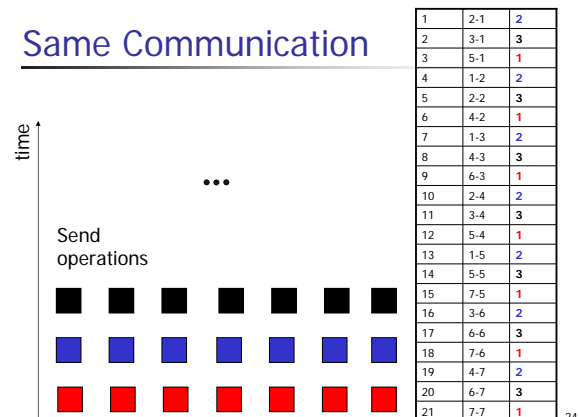


First – Second – Third – ...

1	2-1	2
2	3-1	3
3	5-1	1
4	1-2	2
5	2-2	3
6	4-2	1
7	1-3	2
8	4-3	3
9	6-3	1
10	2-4	2
11	3-4	3
12	5-4	1
13	1-5	2
14	5-5	3
15	7-5	1
16	3-6	2
17	6-6	3
18	7-6	1
19	4-7	2
20	6-7	3
21	7-7	1

23

Same Communication



24

Routing general h -relation

$$T \leq r + T_{\text{binTree}} + s + T_{\text{binTree}} + T_{\text{sort}} + T_{\text{OblivRout}}$$

$$\leq 2T_{\text{binTree}} + h + T_{\text{sort}} + T_{\text{OblivRout}}$$

1. Max received: $r + T_{\text{binTree}}$
 $T_{\text{binTree}} = (L + 2O + G) \log P$
2. E.g. AKS net: $T_{\text{sort}} = O((L + rG) \log P)$
3. Max sent: $s + T_{\text{binTree}}$
4. Routing – same as oblivious: $T_{\text{OblivRout}}$
 $T_{\text{OblivRout}} = \max(L, (h-1)G) + 2O + (h-1)G$

25

Overall time for a super-step

- $T \leq w +$ (compute)
- $2T_{\text{binTree}} + h + T_{\text{sort}} + T_{\text{OblivRout}}$ (route)
- $2T_{\text{binTree}}$ (synchronize)
- $\leq w +$
- $h + T_{\text{sort}} + T_{\text{OblivRout}} +$
- $4T_{\text{binTree}}$
- $T = O(w + \max(L, (h-1)G) + 2O + (h-1)G) \times$
- **Slow**(L, O, G, P, h)
- **Slow**(L, O, G, P, h) = $O(1)$ for sufficiently large h
- **Slow**(L, O, G, P, h) = $O(\log P)$ otherwise

26

Special case oblivious programs

- Routing much easier
 - No sorting as the routing can be predetermined off-line
 - $\max(L, (h-1)G) + 2O + (h-1)G$
 - No synchronization required at all
 - As all data dependencies are known in advance
 - Overall time then:
- $$w + \max(L, (h-1)G) + 2O + (h-1)G$$
- Slow**(L, O, G, P, h) = $O(1)$ always

27

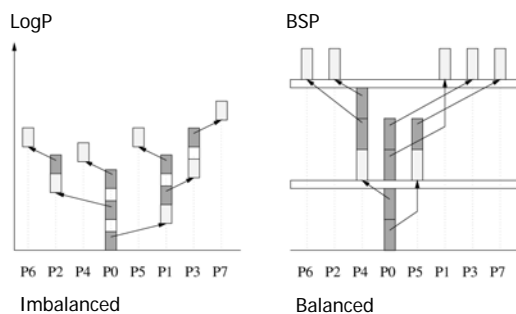
Distinguish Oblivious BSP Programs

- MPMD programs
 - Not natural partition into super-steps
 - Translate to LogP pays off
- Sparse SPMD programs
 - Local communication more efficient than global barrier
 - Translate to LogP pays off
 - Benefit decreases with increasing problem size
- Dense SPMD programs
 - BSP barriers are the natural way of parallelization

28

Example MPMD programs

Broadcast



29

Conclusion

- BSP, LogP interesting machine models for designing algorithms
 - LogP more realistic / efficient on real machines
 - BSP more convenient for programming
- Simulation
 - LogP on BSP – No serious slow down
 - BSP on LogP
 - No serious slowdown for oblivious programs and for programs with sufficient h
 - Slowdown logarithmic in P otherwise
- To do: look at compilation in details

30