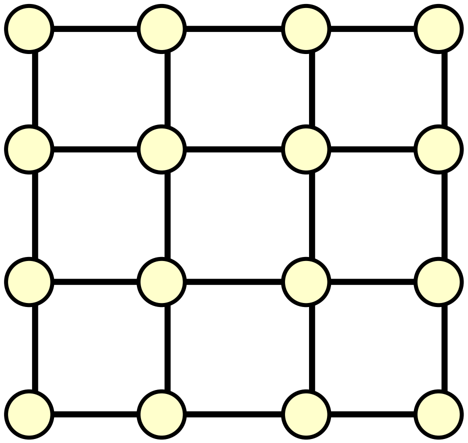
COMP 633 - Parallel Computing

Lecture 16
October 19, 2021

BSP (1)
Bulk-Synchronous Processing Model

Models of parallel computation

- Shared-memory model
  - Implicit communication
    - algorithm design and analysis relatively simple
    - but implementation issues shine through
      - caches, distribution of data in memories, consistency, synchronization costs, ….
    - limits to scaling in practice

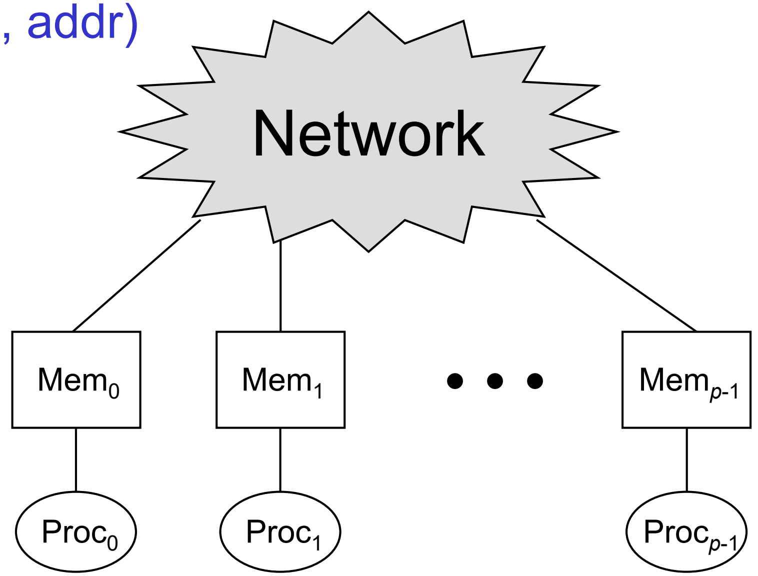


- Distributed-memory model
  - explicit communication (message passing)
    - design and analysis takes into account interconnection network and is complex
    - results not easily transferred between different networks

- "Bridging" model
  - simplified communication costs
    - balance realism with tractability of analysis
    - independent of detailed network characteristics (topology, routing, etc.)
    - cost model relies on average or "expected" network behavior

Bridging model of parallel computation

- p (processor-memory) pairs
  - p separate address spaces (distributed memory)
- Memory references
  - segregated into local and remote references
  - remote references
    - are explicit, typically in the form (proc, addr)
    - carry communication cost
- Global barrier synchronization
  - has large cost



BSP - Bulk Synchronous Parallel programming model

- BSP algorithm consists of a sequence of *supersteps*
- Superstep *i* consists of
  – local work: processors compute asynchronously
    - access values in local memory
    - record remote reads & writes to be performed
  – global communication
    - let Out_i^j be the set of values leaving proc j in step i
    - let In_{i+1}^j be the set of values arriving at proc j at the start of step $i+1$
    - the relation $Out_i \leftrightarrow In_{i+1}$ over all processors specifies the communication pattern
  – global synchronization
    - ensure communication phase is complete
    - ensure memory incorporates all updates (consistency)

proc j memory during step i

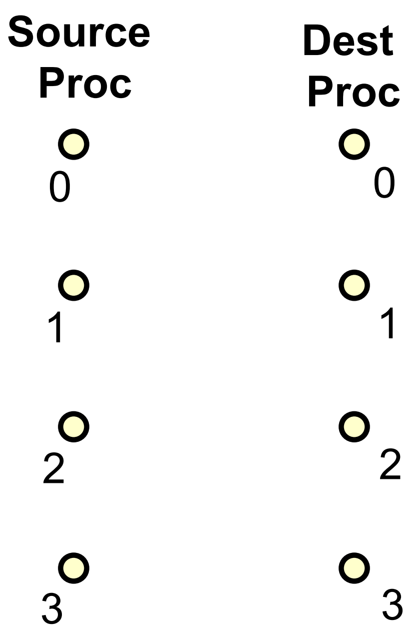
local memory	Out_i^j	In_{i+1}^j

BSP communication cost

- Definition
  - the *communication size* in step i (measured in 8-byte *words*) is

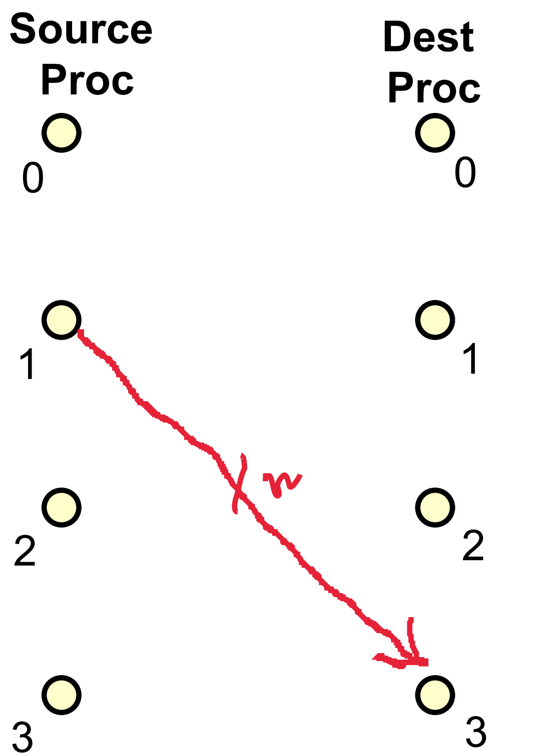
$$h_i = \max_{0 \le j < p} \left(\max \left(\left| Out_i^j \right|, \left| In_{i+1}^j \right| \right) \right)$$

  - the *communication cost* for superstep i is $h_i \cdot g + L$
    - g and L are machine-specific parameters of the cost model where
    - g (bandwidth^{-1} i.e. time per word) is the per-processor full-load permeability of the network
    - L (latency) is the transit time across the network plus any additional time for barrier synchronization of the processors



Basic communication operations (1)

– Send n values from proc 1 to proc 3

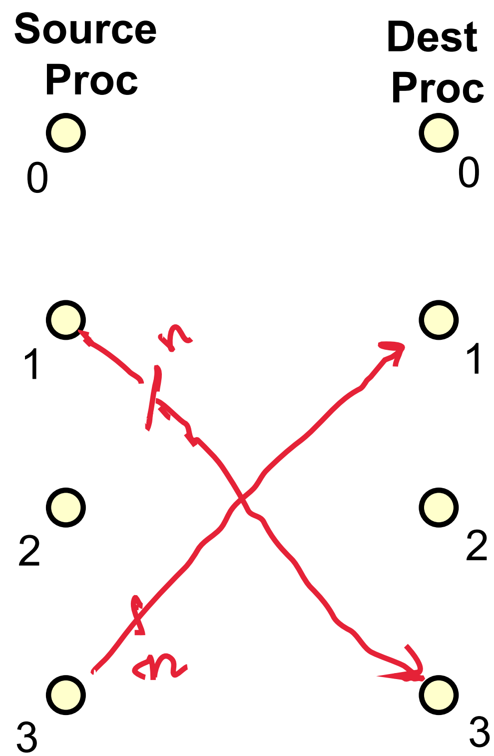


$h = n$

BSP communication cost $= n \cdot g + L$

Basic communication operations (2)

– Exchange n values between proc 1 and proc 3

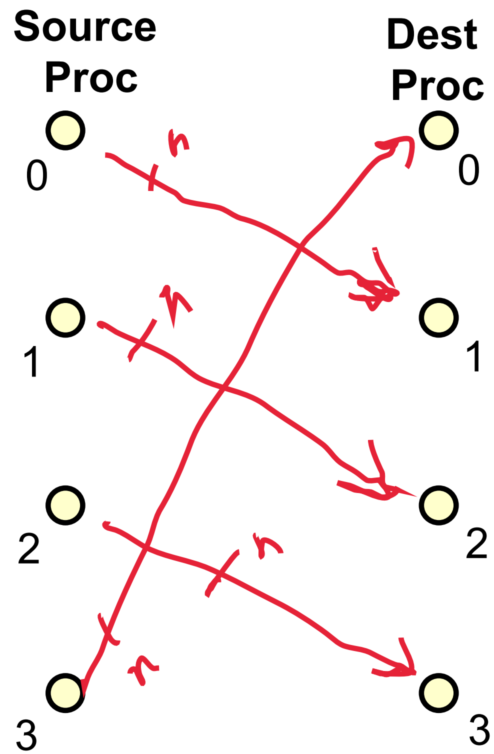


h = n

BSP communication cost = $n \cdot g + L$

Basic communication operations (3)

– Send n values between proc i and proc H(i) forall $0 \leq i < p$, with H a permutation of 0:p-1

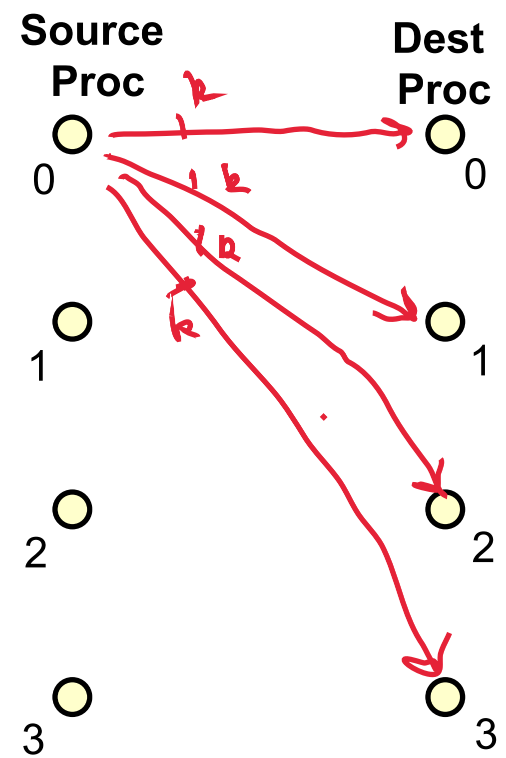


h = n

BSP communication cost = $ng + L$

Basic communication operations (4)

– Distribute n = kp values in proc 0 among p procs. Each proc receives k values from proc 0

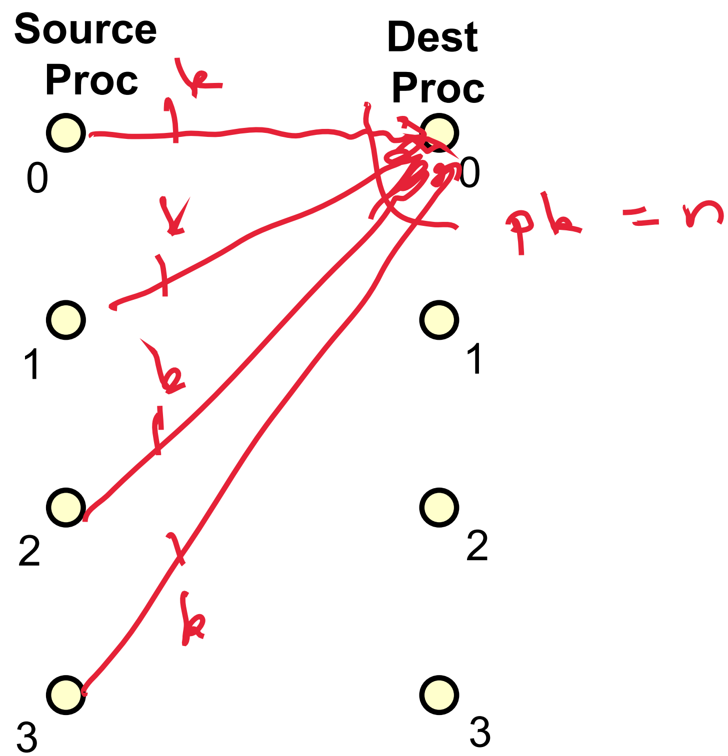


h = $kp = n$

BSP communication cost = $n \cdot g + L$

Basic communication operations (5)

– Combine n = kp values into proc 0. Each proc sends k values

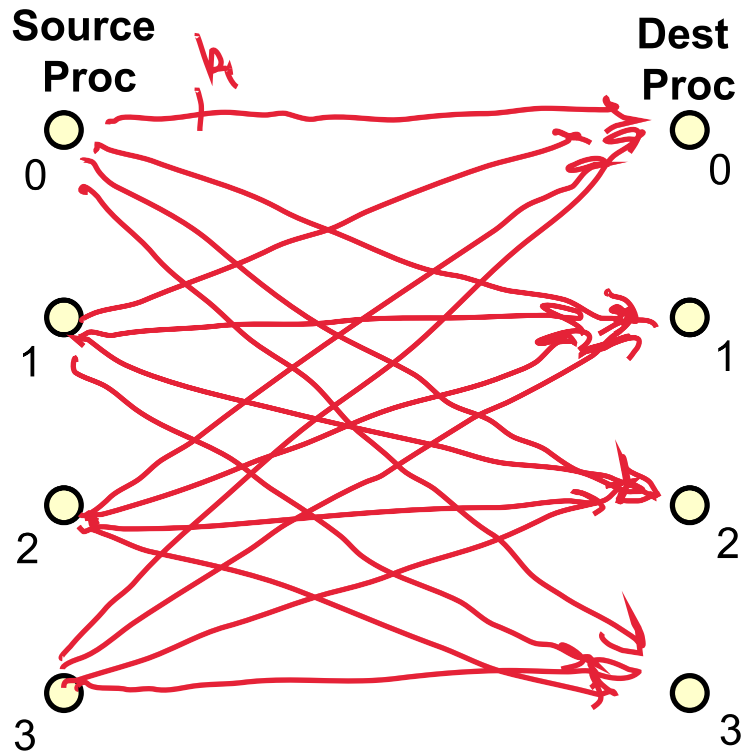


h = n

BSP communication cost = $n \cdot g + L$

Basic communication operations (6)

– Total exchange (all-to-all exchange) of $n = kp$ values among p processors. Each processor receives k values from every other processor

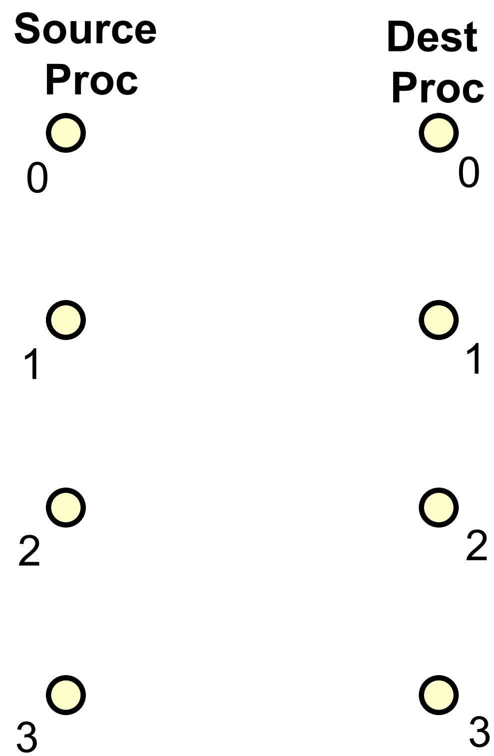


h = n

BSP communication cost = $ng + L$

Basic communication operations (7)

– Broadcast n values from proc 0 to all other processors



two steps

(a) distribute n values across p procs

(b) total exchange

$h_1 = ng + L \qquad h_2 = ng + L$

BSP communication cost $= 2(ng + L)$

BSP programs and execution model

- Basic presentation style is processor-centric
  - not like WT programs
    - number of processors p
    - explicit processor id j
- Single-Program Multiple-Datastream (SPMD) execution model
  - all processors execute same sequential program asynchronously
  - explicitly specify distribution of data over processors
  - specify supersteps
  - for each superstep specify
    - work to be performed by each processor
    - h-relation to be communicated

BSP cost

- Total cost of a BSP algorithm
  - let c be the number of supersteps
  - let p be the number of processors
  - Define

$$w_i = \max_{0 \le j < p} \left(\text{work done in FLOPS on superstep } i \text{ by processor } j\right)$$

$$h_i = \max_{0 \le j < p} \left(\max\left(\left|Out_i^j\right|, \left|In_{i+1}^j\right|\right)\right)$$

  - then total cost (~ running time) $C(n, p)$ of a BSP algorithm is

$$C(n,p) = \sum_{i=1}^{c} (w_i + h_i \cdot g + L)$$

$$= \sum_{i=1}^{c} w_i + \sum_{i=1}^{c} h_i \cdot g + c \cdot L$$

BSP algorithm: Vector summation

- Problem: given V^n distributed evenly over p processors, find s = Sum(V)
    - for simplicity, assume $p = 2^k$ and p divides n
    - let $0 \leq j < p$ be the processor id
    - initially processor j holds $r = n/p$ values: $V[j \cdot r : (j+1) \cdot r - 1]$
    - on completion, each processor holds the value of s
- Algorithm
    - Superstep 1
        - s := Sum ($V[j \cdot r : (j+1) \cdot r - 1]$)
        - read s from proc $(j+1) \bmod p$ into s'
    - Superstep i = 2 to lg p
        - s := s + s'
        - read s in proc $(j + 2^{i-1}) \bmod p$ into s'
    - Superstep 1+ lg p
        - s := s + s'
- BSP cost

BSP algorithm: Vector summation

- Problem: given V^n distributed evenly over p processors, find s = Sum(V)
    - for simplicity, assume p divides n
    - initially processor i holds r = n/p values: V[i•r: (i+1)•r-1]
    - on completion, each processor holds the value of s
- Algorithm
  - Let $0 \leq i < p$ be processor id
    - Superstep 1

      $$\boxed{w_1 = \frac{n}{p} - 1, \quad h_1 = 0}$$

      - s := Sum (V[i•r: (i+1)•r-1])
      - read s in proc (i+1) mod p into s'
    - Superstep j in 2 .. 1 + lg p

      $$\boxed{w_j = 1, \quad h_j = 1}$$

      - s := s + s'
      - read s in proc (i + 2^{j-1}) mod p into s'
- BSP cost

$$C^{\text{sum}}(n,p) = \sum_{j=1}^{1+\lg p} \left(w_j + h_j g + L\right) = \left(\frac{n}{p} - 1 + \lg p\right) + (1 + \lg p) \cdot (g + L)$$

$$\approx \frac{n}{p} + (\lg p) \cdot (g + L)$$

BSP alternate vector summation algorithm

- Problem: given V^n distributed evenly over p processors, find s = Sum(V)
    - for simplicity, assume p divides n
    - initially processor i holds r = n/p values: $V[i \cdot r : (i+1) \cdot r-1]$
    - on completion, each processor holds the value of s
- Algorithm

BSP algorithm: Matrix * Vector

- Problem: given M^{nxn}, V^n distributed evenly over p processors, compute $R = M \bullet V$
    - for simplicity, assume p divides n
    - initially each processor holds n^2/p values of M, and n/p values of V
    - on completion, each processor should hold n/p values of R

- BSP algorithm
  – Let $0 \le j < p$ be processor id, and let $r = n/p$
    - Superstep 1
      – get elements of M from other processors so that local $M' = M[j \bullet r: (j+1) \bullet r-1, \ :]$
      – get elements of V from other processors so that local $V' = V$
    - Superstep 2
      – perform local computation of $R' = M' \bullet V'$ and observe that $R' = R[j \bullet r: (j+1) \bullet r-1]$
      – therefore each processor holds $r = n/p$ elements of the result

- BSP cost

BSP algorithm: Matrix * Vector

- Problem: given M^{nxn}, V^n distributed evenly over p processors, compute R = M•V
    - for simplicity, assume p divides n
    - initially each processor holds n^2/p values of M, and n/p values of V
    - on completion, each processor should hold n/p values of R

- BSP algorithm
  – Let $0 \le j < p$ be processor id, and let r = n/p
    - Superstep 1 $\boxed{w_1 = 0, \quad h_1 = nr + n}$
      – get elements of M from other processors so that local M' = M[j•r: (j+1)•r-1, :]
      – get elements of V from other processors so that local V' = V
    - Superstep 2 $\boxed{w_2 = \frac{2n^2}{p}, \quad h_2 = 0}$
      – perform local computation of R' = M' • V' and observe that R' = R[j•r: (j+1)•r-1]
      – therefore each processor holds r = n/p elements of the result

- BSP cost

$$C^{\mathrm{MV}}(n,p) = \frac{2n^2}{p} + \left(\frac{n^2}{p} + n\right) \cdot g + 2 \cdot L$$

BSP algorithm: Matrix * Matrix

- Problem: given A, B $\in \Re^{nxn}$ distributed evenly over p processors, compute C = A•B
  - assume $p^{1/2}$ integral and divides n
  - initially each proc holds n^2/p values of A and B
  - on completion, each proc should hold n^2/p values of C

- BSP algorithm
  – Let (i,j) in (0.. $p^{1/2}$ -1, 0.. $p^{1/2}$ -1) be the processor id, and let s = $n/p^{1/2}$
  - Superstep 1
    – get elts of A from other processors so that A' = A[i•s: (i+1)•s-1 , :]
    – get elts of B from other processors so that B' = B[: , j•s: (j+1)•s-1]
  - Superstep 2
    – perform local computation of C' = A' • B' to compute s $\times$ s portion of C

- BSP cost

BSP algorithm: Matrix * Matrix

- Problem: given A, B $\in \Re^{n \times n}$ distributed evenly over p procs, compute C = A•B
  - assume $p^{1/2}$ integral and divides n
  - initially each proc holds n^2/p values of A and B
  - on completion, each proc should hold n^2/p values of C

- BSP algorithm
  – Let (i,j) in (0.. $p^{1/2}$ -1, 0.. $p^{1/2}$ -1) be the processor id, and let s = $n/p^{1/2}$
  - Superstep 1 $\boxed{w_1 = 0, \quad h_1 = 2(n/\sqrt{p})n = \frac{2n^2}{\sqrt{p}}}$
    – get elts of A from other processors so that A' = A[i•s: (i+1)•s-1 , :]
    – get elts of B from other processors so that B' = B[: , j•s: (j+1)•s-1]
  - Superstep 2 $\boxed{w_1 = (2n)\left(\frac{n}{\sqrt{p}}\right)^2 = \frac{2n^3}{p}, \quad h_1 = 0}$
    – perform local computation of C' = A' • B' to compute s $\times$ s portion of C

- BSP cost $$C^{\mathrm{MM}}(n,p) = \frac{2n^3}{p} + \left(\frac{2n^2}{\sqrt{p}}\right) \cdot g + 2 \cdot L$$

BSP cost model: units

- Goal: architecture-independent performance analysis
  - g and L are expressed in FLOPS
  - h is expressed in words (8 bytes)
    - g = 10 means 10 FLOPS can be performed for every word communicated
- Relating BSP cost to running time
  - $T_p(n,p) = s \cdot C(n,p)$
    - parallel running time $T_p(n,p)$
    - BSP cost $C(n,p)$
    - s is a processor-specific constant in units of seconds per flop
      - typically s = 1/(peak MFLOPS per second)
      - tends to substantially underestimate true time on many machines

g, L, s values for some (old) machines

Machine	Network topology	p_{max}	Bisection b/w B (MB/s)	Peak rate r (Mflops)	g = 8r/B (flops/wd)	L (flops)	s (sec/flop)
PC	bus	4	250	250p	8p	1200	4×10^{-9}
SGI O2000	hypercube	128	250p	500p	16	800	2×10^{-9}
Cray T3E	3D Torus	1024	$600p^{2/3}$	900p	$12p^{1/3}$	500	1.1×10^{-9}
NEC SX-5	crossbar	16	64000p	8000p	1	400	0.13×10^{-9}

- Notes
  - Bisection bandwidth is for the complete network and is measured in megabytes per second
  - Peak computing rate is total for p processor machine and is measured in megaflops per second

BSP metrics: normalized cost

- Normalized BSP cost
  - ratio of BSP cost to optimal parallel execution

$$\overline{C}(n,p) = \frac{T_P^{BSP}(n,p)}{W(n)/p}$$
$$= a + b \cdot g + c \cdot L$$

  - work efficiency goal
    - a ~ 1
  - communication efficiency goal
    - b << 1/g
    - c << 1/L

More BSP metrics: asymptotic efficiency

- Recall $C(n,p) = \sum_{i=1}^{c} w_i + \sum_{i=1}^{c} (h_i \cdot g + L)$
- Asymptotic efficiency
  - work efficiency π
    - also measures load-balance
    - goal π close to 1

$$\pi = \lim_{n \to \infty} \left(\frac{\sum_{i=1}^{c(n,p)} w_i}{W(n)/p} \right)$$

  - communication overhead μ
    - goal $\mu < 1$

$$\mu = \lim_{n \to \infty} \left(\frac{\sum_{i=1}^{c(n,p)} (h_i \cdot g + l)}{W(n)/p} \right)$$

- Examples
  - Matrix * Vector
    - $\pi = 1$, $\mu = g/2$
    - highly dependent on network performance at all problem sizes
  - Matrix * Matrix
    - $\pi = 1$, $\mu = 0$
    - insensitive to network performance, for sufficiently large problems