A Bridging Model for Multi-Core Computing

L. G. Valiant

Journal of Computer and System Sciences, 2011

袁 晨（180949）
陈 恳（180958）

November 8, 2018

Contents

- **Problem**: It's hard to write program for parallel systems
  - resources are lacking
- **Method:** Multi-BSP Model
  - Bridging model
  - Multi-level model
  - Designing efficient and portable
- **Discussion:** parameter-aware portable algorithms
- Standard matrix multiplication
- Fast Fourier Transform
- Comparison Sorting

Challenges for multi-core architectures

Comparing with sequential system

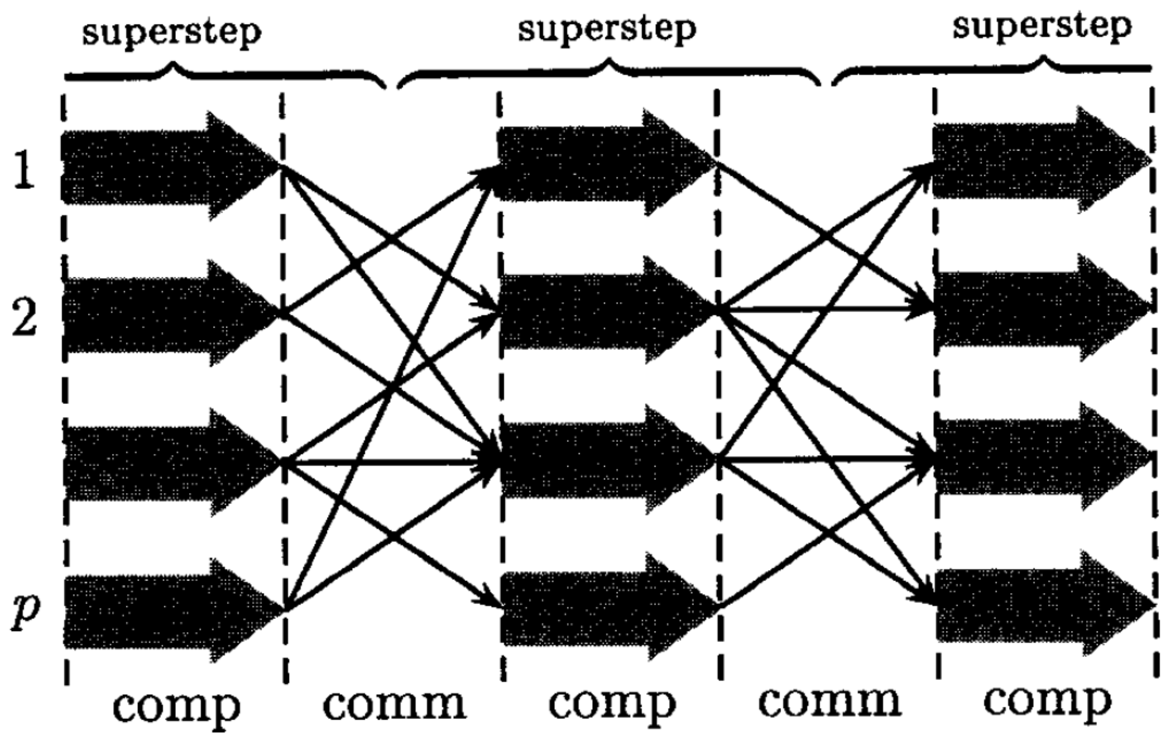
- Underlying computational substrate is much more complex than conventional sequential computing
- Sequential algorithms are much better understood and highly optimized
- As machines differs, one optimized algorithm for one machine may not work on another
- Acceleration is limited (at best a speedup of a constant factor)

Portable Parallel Algorithms

- Portable parallel algorithms are parameter-aware
- Have to expressed as a bridging model
  - Make a bridge between programming language and computer architecture
- Necessary performance parameters should be defined
  - a prerequisite for portable parallel algorithms to be possible

From BSP* model to multi-BSP model

- **B**ulk **S**ynchronous **P**arallel model for parallel computation:

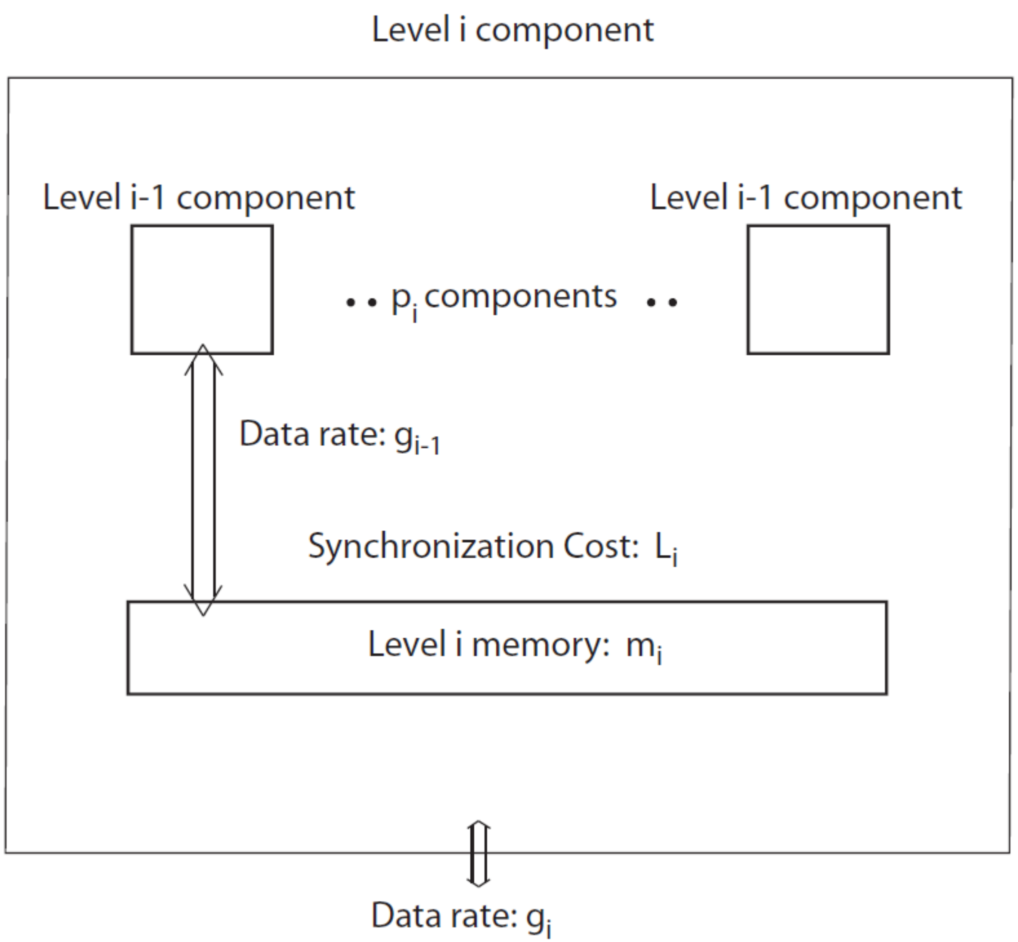


* A. Tiskin, “The bulk-synchronous parallel random access machine,” Theoretical Computer Science, vol. 196, no. 1, pp. 109–130, Apr. 1998.

From BSP* model to multi-BSP model

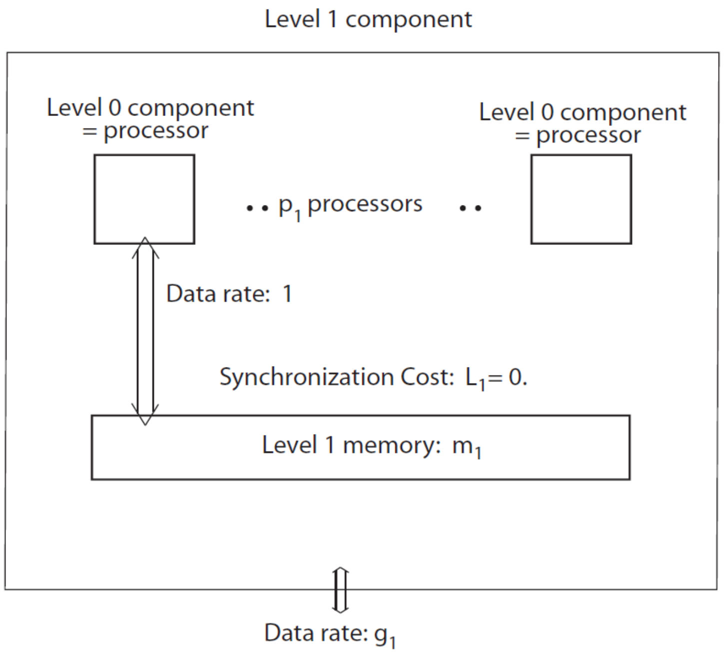
- **Multi-BSP model**
  - Hierarchical model
    - Arbitrary number of levels
    - Modeling all levels of an architecture together
  - At each level, multi-BSP contains memory size as a further parameter
    - Physical limitation on the amount of memory that can be accessed in fixed time from physical location of a processor ⇒ multiple levels

Multi-BSP model: Parameters



- p_i: number of i-1 level component
- g_i: communication bandwidth
  - (Operations a processor can do per second) / (words can be transmitted between level i component memories and level $i + 1$ per second)
- L_i: barrier synchronization cost for superstep i
- m_i: number of words in i^{st}component that is not in any $i - 1$ component

Multi-BSP model: Structure



- Tree structure
  - Leaf components are processors
  - Other level contains storage capacity
  - Doesn't distinguish memory from cache
- Processor number in level i component
  - $P_i = p_1 \cdots p_i$
- Number of level i components in a level j component
  - $Q_{i,j} = p_{i+1} \cdots p_j$, for whole system($j = d$): $Q_{i,d} = p_{i+1} \cdots p_d$
- Total memory available in a level i component
  - $M_i = m_i + p_i m_{i-1} + p_{i-1} p_i m_{i-2} + \cdots + p_2 \cdots p_{i-1} p_i m_1$
- Communication cost(level 1 to i) : $G_i = g_i + g_{i-1} + g_{i-2} + \cdots g_1$

Multi-BSP model: Assumption and Discussion

- To simplify our analysis, make assumption that for i:
  - $m_i \geq m_{i-1}$
  - $m_i \geq M_i/i$
- **Definition:**

$F_1 \lesssim F_2$: $\forall \varepsilon > 0, F_1 < (1 + \varepsilon)F_2$

$F_1 \lesssim_d$: for constant c_d, $F_1 < (1 + c_d)F_2$, c_d depending on d

For multi-BSP algorithm A*:

- Comp(A*): parallel costs of **comp**utation
- Comm(A*): parallel costs of **comm**unication
- Synch(A*): parallel costs of **synch**ronization

Multi-BSP model: Algorithm & Parallel Costs

- To quantify the efficiency of A^*, defining a *baseline* algorithm A, multi-BSP machine H,
  - $Comp_{seq}(A)$: total number of operations of A
    - $Comp(A) = Comm_{seq}(A)/P_d$, P_d is processor number in H
  - $Comm(A)$: minimal comm cost for A on H
  - $Synch(A)$: minimal synch cost of A on H
- A^* is optimal with respect to A, if :
  - $Comp(A^*) \lesssim_d Comp(A)$ & $Comp_{seq}(A^*) \lesssim_d Comp_{seq}(A)$
  - $Comm(A^*) \lesssim_d Comm(A)$
  - $Synch(A^*) \lesssim_d Synch(A)$

Multi-BSP model: Architecture Requirements

- Barrier synchronization needs to be supported efficiently
  - Literatures shows that this can be done already with current architectures*
- Model should control storage explicitly
- Machines support the features of this model

* N. Vachharajani, M. Iyer, C. Ashok, M. Vachharajani, D. I. August, and D. Connors, “Chip Multi-processor Scalability for Single-threaded Applications,” SIGARCH Comput. Archit. News, vol. 33, no. 4, pp. 44–53, Nov. 2005.

Work Limited Algorithms: Definition

- **Definition**

Straight-line program A is $w(S)$-limited:

- Every subset of its operations
  - uses at most S inputs
  - produces at most S outputs
  - consists of no more than w(S) operations

Work Limited Algorithms: Propositions

- For associative composition task AC(n):
  - A: linear array of n elements from a set X
  - $\otimes$: an associative binary operation on X
  - A specific set of disjoint contiguous subarrays of A

Proposition 1

For any n and S, any algorithm for associative composition $AC(n)$ is $(S - 1)$ limited

Work Limited Algorithms: Propositions

- Problem $MM(n \times n)$ for multiplying two $n \times n$ matrices by standard algorithm

Proposition 2

For any n and S, the standard matrix multiplication algorithm $MM(n \times n)$ is $S^{3/2}$-limited

Work Limited Algorithms: Propositions

- For $FFT(n)$ the standard binary recursive algorithm for computing the one-dimensional Fast Fourier Transform on n points (n is power of 2)

Proposition 3

For any n and S the standard Fast Fourier transform algorithm $FFT(n)$ is $2S\ \log_2 S$ -limited

Lower Bounds : Lemma

Lemma

Suppose W computation steps are executed of a $w(S)$-limited straight-line program on a Multi-BSP machine. Then for any j the total number of words transmitted between level j components and the level $j+1$ components to which they belong is at least

$$M_j(W/w(2M_j) - Q_j)$$

And the total number of level j component supersteps at least

$$W/w(M_j)$$

Lower Bounds：Theorem

Theorem

Suppose W(n) operations are to be performed of a $w(m)$- limited straight-line program A on input size n on a depth d Multi-BSP machine. Then the communication cost over the whole machine is at least

$$Comm(n,d) \gtrsim_d \sum_{i=1\cdot\cdot d-1} (W(n)/Q_i w(2M_i)) - 1)M_i g_i$$

The synchronization cost at least:

$$Synch(n,d) \gtrsim_d \sum_{i=1\cdot\cdot d-1} W(n)L_{i+1}/(Q_i w(M_i))$$

Lower Bounds : Application (I)

- For associative composition:

$$\text{AC-Comm}(n,d) \gtrsim_d \sum_{i=1 \cdot\cdot d-1} (n/(M_i Q_i) - 1)\, M_i g_i$$

$$\text{AC-Synch}(n,d) \gtrsim_d \sum_{i=1 \cdot\cdot d-1} n L_{i+1} / (Q_i M_i)$$

- For Matrices Multiplication

$$\text{MM-Comm}(n \times n, d) \gtrsim_d \sum_{i=1 \cdot\cdot d-1} (n^3/Q_i M_i^{3/2} - 1)\, M_i g_i$$

$$\text{MM-Synch}(n \times n, d) \gtrsim_d \sum_{i=1 \cdot\cdot d-1} n^3 L_{i+1} / (Q_i M_i^{3/2})$$

Lower Bounds : Application (II)

- For Fast Fourier Transform algorithm:

  FFT-Comm$(n, d)\ \gtrsim_d\ \sum_{i=1..d-1}(n\log n\,/(Q_i M_i \log M_i) - 1)M_i g_i$

  FFT-Synch$\gtrsim_d\ \sum_{i=1..d-1} n\log n\, L_{i+1}/(Q_i M_i \log M_i)$

- For Sorting

  Sort-Comm$(n, d) \gtrsim_d\ \sum_{i=1..d-1}(n\log n\,/Q_i M_i \log M_i - 1)M_i g_i$

  Sort-Synch$(n, d)\ \gtrsim_d\ \sum_{i=1..d-1} n\log n\, L_{i+1}/(Q_i M_i \log M_i)$

Optimal Algorithms

$$\text{AC-Comm}(n,d) \precsim_d \sum_{i=1\cdot\cdot d-1} n g_i / Q_i$$

$$\text{AC-Synch}(n,d) \precsim_d \sum_{i=1\cdot\cdot d-1} n L_{i+1} / (Q_i m_i)$$

$$\text{MM-Comm}(n \times n,d) \precsim_d \sum_{i=1\cdot\cdot d-1} (n^3 g_i / Q_i) \sum_{k=i\cdot\cdot d-1} (1/m_k^{1/2})$$

$$\precsim_d n^3 \sum_{j=1\cdot\cdot d-1} g_j m_j^{-1/2} / Q_j$$

$$\text{MM-Synch}(n \times n,d) \precsim_d n^3 \sum_{j=1\cdot\cdot d-1} L_{j+1} m_j^{-3/2} / Q_j$$

Optimal Algorithms

$$\text{FFT-Comm}(n,1,d) \precsim_d \sum_{i=1\cdot\cdot d-1} (n\log n) g_i/(Q_i \log m_i)$$

$$\text{FFT-Synch}(n,1,d) \precsim_d \sum_{i=1\cdot\cdot d-1} (n\log n) L_{i+1}/(Q_i m_i \log m_i)$$

$$\text{Sort-Comm}(n,d) \precsim_d \sum_{i=1\cdot\cdot d-1} (n\log n) g_i/(Q_i \log m_i)$$

$$\text{Sort-Synch}(n,d) \precsim_d \sum_{i=1\cdot\cdot d-1} (n\log n) L_{i+1}/(Q_i m_i \log m_i)$$

Thank You