An Energy Aware Model of Computation, *and* Computation with Energy-Time Trade-Offs: Models, Algorithms and Lower-Bounds

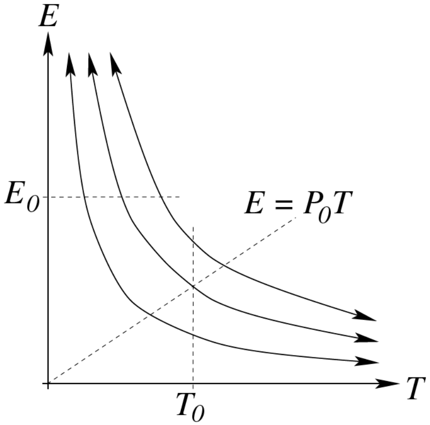
Brad D. Bingham Mark R. Greenstreet

Presenter: Brian Wongchaowart
April 21, 2010

Objective: Minimize ET^α

- Main thesis of this paper: ET^α is a measure of the goodness of an algorithm.
- Suppose that for some input, an algorithm requires E units of energy and T units of time.
- If power as a function of speed is $P(s) = s^\beta$, then running the algorithm c times faster increases energy by a factor of c^β / c ($E = Pt$).
- In other words, T^α decreases by a factor of $1/c^\alpha$, while E increases by a factor of c^α, where $\alpha = \beta - 1$. Thus ET^α remains constant under speed scaling, but depends on the algorithm and input size.

Knowing ET^α is useful whether the objective is time, energy, or power.



Machine Model

- A machine is modeled as a network of processing elements (PEs) on a plane.
- A PE has $O(1)$ input bits, $O(1)$ output bits, and stores $O(1)$ bits of state. It can set its outputs to an arbitrary function of its inputs and state using e units of energy and t units of time, where $et^{\alpha} = 1$.
- The input to an algorithm is initially stored at designated input PEs and the output of the algorithm must be stored at designated output PEs when computation terminates.
- Sometimes all the input PEs are required to lie along a line, and all the output PEs are required to lie along a possibly different line (*perimeter I/O*).

Lower-Bound Model

- Lower bounds on ET^α can be proven by examining the minimum communication cost that must be incurred by any algorithm for a problem.
- A PE can have at most d^2 other PEs within distance d of itself, since the PEs lie on a plane.
- The time t and energy e required to send a bit between two PEs distance d apart must satisfy $et^\alpha = d^{\alpha+1}$.
- Justification: A wire of length d can be thought of as a chain of d PEs. Since each PE can copy its input bit to its output bit using 1 unit of time and 1 unit of energy, sending a bit through d PEs requires d units of time and d units of energy, giving $et^\alpha = d \cdot d^\alpha = d^{\alpha+1}$. Scaling the transmission speed t does not alter et^α because of the corresponding increase in energy e.

Upper-Bound Model

- An upper bound on ET^α for a problem can be demonstrated by giving a concrete layout of PEs on a plane and an algorithm for how the PEs are used.
- To ensure that the design can be implemented, PEs are required to occupy a unit square and can only communicate with the four PEs directly adjacent to it.
- Wires are just chains of PEs that copy their inputs to their outputs.

Binary Addition: Lower Bound

- We prove a lower bound on ET^α for binary addition of two n-bit input words from the fact that a carry generated by the least significant bit can affect all the bits of the sum ($00000001 + 01111111 = 10000000$).
- Thus one bit of information must be propagated from the input PE with the least significant bit to all of the output PEs.
- With the perimeter I/O constraint, the $n+1$ output PEs lie along a line, so one bit of information must travel over a distance of at least $n/2$. Since the communication cost alone is at least $et^\alpha = d^{\alpha+1} = (n/2)^{\alpha+1}$, the total cost of any perimeter I/O binary addition algorithm must be $\Omega(n^{\alpha+1})$.
- Without the perimeter I/O constraint, one bit of information must travel over a distance of at least $\sqrt{n}$, since it must reach $n+1$ output PEs. The ET^α complexity of any binary addition algorithm is therefore $\Omega((\sqrt{n})^{\alpha+1}) = \Omega(n^{(\alpha+1)/2})$.

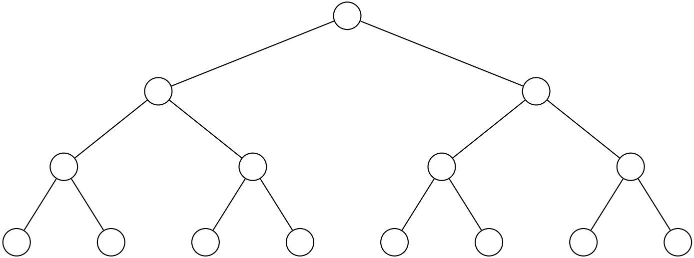
Binary Addition: Perimeter I/O Upper Bound

- A ripple carry adder can be constructed from a chain of n 1-bit adders.
- The ith adder is given bit i of each operand as input, as well as the carry in c_i from the previous adder ($c_0 = 0$).
- It produces bit i of the sum as output—$s_i = a_i \oplus b_i \oplus c_i$—and computes the carry out as $c_{i+1} = (a_i \cdot b_i) + (a_i + b_i) \cdot c_i$.
- The 1-bit adders operate sequentially, and each adder only needs to perform computation during one time step, so the addition requires time $O(n)$ and energy $O(n)$, giving $ET^\alpha = O(n^{\alpha+1})$. This matches the lower bound of $\Omega(n^{\alpha+1})$.

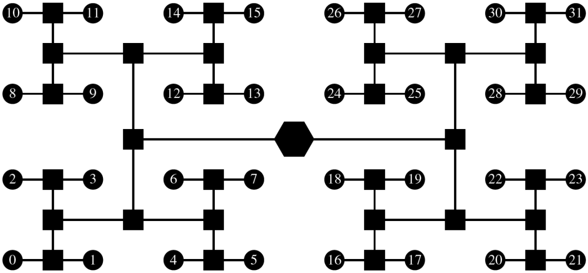
Binary Addition: Planar I/O Upper Bound

- A carry-lookahead adder can be implemented as a binary tree.
- The *i*th leaf node receives bit i of each operand as input and computes bit i of the sum as output: $s_i = a_i \oplus b_i \oplus c_i$.
- The carry in c_i comes from the parent of the leaf node. In order for this to be computed, each leaf node first provides its parent with a carry-generate bit $g_i = a_i \cdot b_i$ and a carry-propagate bit $p_i = a_i \oplus b_i$.
- Each internal node receives g_l and p_l from its left child and g_r and p_r from its right child. It calculates the carry-generate bit for the entire subtree as $g_t = g_r + (g_l \cdot p_r)$ and a carry-propagate bit for the entire subtree as $p_t = p_l \cdot p_r$, and sends these values to its parent.

- The root node provides 0 as the carry in for its left child, g_l as the carry in for its right child, and sets bit $n+1$ of the sum to $g_r + (g_l \cdot p_r)$. Every other internal node copies the carry in from its parent to the carry in for its left child, and sets the carry in for its right child to $g_l + (c_{\text{parent}} \cdot p_l)$.



- The binary tree carry-lookahead adder can be laid out on the plane as an H-tree.
- Setting the time for PEs at level k and the PEs in the chain from level k to level $k+1$ to $2^{k/(\alpha+1)}$ yields $E = O(\sqrt{n})$, $T = O(\sqrt{n})$, and $ET^\alpha = O((\sqrt{n})^{\alpha+1}) = \Omega(n^{(\alpha+1)/2})$ for $\alpha > 1$, matching the lower bound for ET^α.



Sorting: Lower Bound

- Sorting corresponds to matching each input position with the correct output position.
- To derive a lower bound on the communication cost, consider the input PEs one by one and permute the input values in such a way that each input PE must send its input to the unmatched output PE that is farthest from it.
- The ith input PE can be matched with $n - i + 1$ output PEs. For perimeter I/O (all the output PEs lie along a line), this means that the distance from input PE i to the farthest unmatched output PE is at least $(n - i)/2$. For planar I/O, the distance to the farthest unmatched output PE is at least $\sqrt{n - i}$.

- Suppose that the entire sort is completed in T time units.
- For perimeter I/O, the energy e_i used to send the ith input value to the correct output PE in time t_i must satisfy $e_i t_i^\alpha = d^{\alpha+1} \geq ((n-i)/2)^{\alpha+1}$. Since $t_i \leq T$, $e_i \geq ((n-i)/2)^{\alpha+1} T^{-\alpha}$.
- The total energy must be at least

$$E = \sum_{i=1}^{n} e_i \geq \sum_{i=1}^{n} ((n-i)/2)^{\alpha+1} T^{-\alpha} = 2^{-(\alpha+1)} T^{-\alpha} \sum_{i=1}^{n} (n-i)^{\alpha+1}.$$

  Thus $ET^\alpha = \Omega(n^{\alpha+2})$.
- For the planar I/O case, $e_i \geq (\sqrt{n-i})^{\alpha+1} T^{-\alpha}$. Thus

$$E = \sum_{i=1}^{n} e_i \geq T^{-\alpha} \sum_{i=1}^{n} (n-i)^{(\alpha+1)/2},$$

  and $ET^\alpha = \Omega(n^{(\alpha+3)/2})$.

Sorting: Perimeter I/O Upper Bound

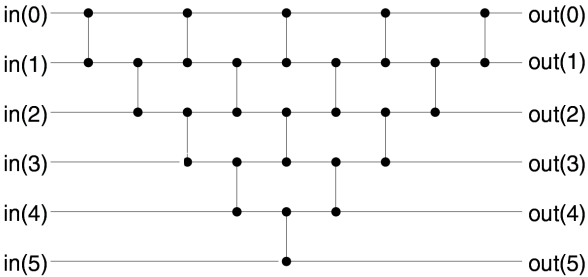
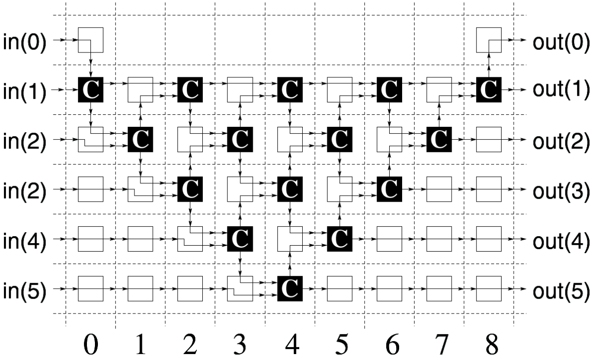


Figure: Sorting network for bubble sort.

In the implementation of bubble sort shown below, allocating unit time to all PEs results in $T = O(n)$ and $E = O(n^2)$. Thus $ET^{\alpha} = O(n^{\alpha+2})$, matching the lower bound for perimeter I/O.



Summary

- ET^{α} as a measure of algorithm quality
- Lower-bound versus upper-bound models
- Perimeter I/O versus planar I/O
- Matching lower and upper bounds for addition, multiplication, and sorting

Questions/Comments