Designing ultra-fast algorithms

Jean-Louis Roch
Ensimag, INP, Grenoble University
INRIA / LIG Moais

• Multiple Access (SDMA / FDMA / CDMA ) provides concurrent access to memory in constant time :

  CRCW : Concurrent Read Concurrent Write

Algoritmic costs
Computing the maximum

• Designing the fastest circuit to compute the maximum

• Input : n elements $a_i$ of an ordered set $<$
  Output : the maximum element

• Available gates :
  \[ \text{And} \quad \text{Or} \quad < ? \quad \text{Max} \]

---

Basic serial circuit

\[ \text{Circuit}_{seq} \]

D=Depth= n  \quad W = \text{Work} = n
(NB Work and depth are in number of Compare gates)

\[
\begin{align*}
\text{res} & := a_1 ; \\
\text{For } i := 2 \ldots n \text{ do } \\
\text{res} & := \text{Max} ( \text{res}, a_n ) ; \\
\text{Return } \text{res};
\end{align*}
\]
Faster with Parallelism

\[ D = \log_2 n \quad \text{and} \quad W = n \]

**Circuit_{par}**

\[
\text{Function max2}(a_1 \ldots a_k) \{
\text{if } (i == k) \text{ return } a_i;
\text{else} \{ \\
\quad \text{PARALLEL} \{ \\
\quad \quad rl = \text{max2}(a_i \ldots a_{(i+k)/2}); \\
\quad \quad rh = \text{max2}(a_{(i+k)/2+1} \ldots a_k); \\
\quad \}\ \\
\quad \text{return Max}(rl, rh); \\
\}\}
\]

May Multiple Access help?

- Taking benefit of multiple access:
  logical or of n bits in constant time

\[
\text{Multiple Access OR}
\]
Ultrafast algorithm for testing the maximum

\[ a_k = \text{Max}(a_1 .. a_n) \iff a_k \geq a_i \text{ for } i \neq k \iff \text{AND}_{i \neq k}(a_k \geq a_i) \iff \text{NOT}(\text{OR}_{i \neq k}(a_k < a_i)) \]

Multiple Access OR

False iff \( a_k \) is the maximum

\[ D = 1 \quad \odot \quad W = n \]

Application: computing the maximum

Only one communication for the maximum element

\[ D = 1 \quad \odot \quad W = n^2 \quad \odot \]
A recursive ultrafast parallel algo

\[
D(n) = D(n^{0.5}) + 1 = \log\log n
\]

\[
W(n) = n^{0.5} \cdot \text{procs}(n^{0.5})n + n = n \log\log n
\]

Optimizing the number of units

• Take benefit of the parallel algorithm to minimize the number of units (could be the sequential one)
Conclusion: an ultrafast algorithm

- Final algorithm: $D = \text{depth} = 3 \cdot \log \log n$ 🤣
  $W = \text{work} = 1 \cdot n + O\left(\frac{n \log \log n}{\log n}\right)$ 😎

- Technique used: «cascading»
  mixing 3 algorithms to obtain an ultrafast one!

- Fundamental technique for parallel algorithms design
  and in software engineering too.

- Both theoretical and practical issues.

- Exercises:
  - Circuit for Merge on CREW PRAM in $D = \log \log n$ and $n$ ops
  - D&C Program for Prefix with depth=$O(\log n)$ (not $\log^2 n$!) (in Kaapi or Cilk or TBB or …)