Cache-Adaptive Algorithms

Michael A. Bender†‡, Roozbeh Ebrahimi†, Jeremy T. Fineman*, Golnaz Ghasemiesfah†, Rob Johnson†, Samuel McCauley†.
Memory Fluctuates in Real Systems

Available memory to processes changes over time in real systems.

I/O bound processes:
- Joins, sorts in a DBMS.
- External memory algorithms running in time-sharing systems.
Real World vs. Theory

Most theoretical external-memory algorithms/analyses assume fixed memory.

- Disk Access Machine (DAM) model [AggarwalVitter'88].

There is one exception from 15 years ago:

- Barve and Vitter [98, FOCS'99] generalize the I/O model to allow memory to change size (sorting, FFT, matrix multiplication, etc).

- Little follow-up work because these algorithms are technically hard to design and analyze.
Point of This Talk

We utilize cache-oblivious technology to achieve cache-adaptivity.

- Cache-adaptive algorithms adapt when available memory changes during execution.
We offer an analysis toolbox for the cache-adaptive model.

- Piece-wise progress bounds, square memory profiles, inductive charging.
Results

- **Class of recursive CO algorithms** that are optimally cache-adaptive.
  - Matrix multiplication, matrix transpose, Gaussian elimination, all-pairs shortest paths.
  - **Lazy funnel sort** [Brodal+Fagerberg’02] is optimally cache-adaptive.

- Cache-replacement policies in the CA model:
  - 4-memory & 4-speed augmented **LRU** is competitive.
  - **Belady’s algorithm** is still optimal in CA model.
Disk Access Machine (DAM) Model

Data is transferred in blocks between memory and disk, and time is measured in terms of block transfers.

- Time bounds in block size $B$, memory size $M$, data size $N$.
- Goal: minimize block transfers.
Cache-Adaptive Model

Same metric as DAM, time \( \approx \) \#of block transfers.

- Memory size \( M(t) \) is known, but changes arbitrarily.
- Time is measured in I/Os.
  - If there are no block transfers time doesn’t progress (DAM). Hence, memory only changes on I/Os.
Cache-Oblivious Algorithms
[Frigo, Leiserson, Prokop, Ramachandran ’99]

- Ideal-cache model: DAM + automatic optimal cache-replacement.
- Parameters $B$ and $M$ are not known to the algorithm.
- An optimal CO algorithm is universal for all $B$, $M$, $N$.
- Optimal replacement can be simulated by LRU with extra memory [SleatorTarjan85].

- CA model also assumes automatic replacement.
Obliviousness vs. Adaptivity

\[ \text{cache-oblivious} \neq \text{cache-adaptive} \]

- Some optimal CO algorithms aren’t cache adaptive.
- For some problems, no optimal CA solution can be CO.

- CA and CO are Complementary notions.
  - CO: fixed, unknown \( B \) and \( M \).
  - CA: known fixed \( B \), known changing \( M(t) \).
We “want” **optimality** $\approx$ worst case running time of an algorithm is less than a constant factor of all other algorithms.

- **Issue:** If algorithm $A$ is **twice** slower than $\text{OPT}$ and $\text{OPT}$ finishes at $t_1$, $A$ can have exponentially more I/Os than $\text{OPT}$.

- **$K$-speed** augmentation $\approx$ An algorithm gets $k$ times as much I/O at each time step.

- **Optimality:** with **$O(1)$-speed** augmentation faster than all other algorithms.
**Example: Rec. Matrix Mult. (RMM)**

**RMM:** A $\sqrt{N} \times \sqrt{N}$ matrix multiplication can be done using 8 multiply of $\sqrt{N}/2 \times \sqrt{N}/2$ matrices.

\[
\begin{array}{c c}
A_{11} & A_{12} \\
A_{21} & A_{22} \\
\end{array}
\times
\begin{array}{c c}
B_{11} & B_{12} \\
B_{21} & B_{22} \\
\end{array}
=\begin{array}{c c}
A_{11}B_{11} & A_{11}B_{12} \\
A_{21}B_{11} & A_{21}B_{12} \\
\end{array}
+\begin{array}{c c}
A_{12}B_{21} & A_{12}B_{22} \\
A_{22}B_{21} & A_{22}B_{22} \\
\end{array}
\]

\[
T(N) = \begin{cases} 
0(N/B + 1) & \text{if } N \leq M, \\
8T(N/4) + O(1) & \text{otherwise.}
\end{cases}
\]

- RMM is an optimal cache-oblivious algorithm.
- The recursion **bottoms out** at a matrix multiplication of size $\Theta(\sqrt{M} \times \sqrt{M})$.
- Solving the bottomed-out cases costs just a linear scan.
Complications in the CA Analysis

The recursion **bottoms out** at **different** levels since the base case changes over time:

\[
T(N) = \begin{cases} 
0(N/B + 1) & \text{if } N \leq M(t), \\
8T(N/4) + 0(1) & \text{otherwise.} 
\end{cases}
\]
RMM is optimal in the CA

**Theorem:** Recursive matrix multiply is optimally cache-adaptive.

- We show that in each time interval:
  
  **Algorithm's progress** ~ max possible progress.

**Algorithm's progress:** Fraction of page references executed so far in the recursion tree.
**Piece-wise Progress Bounds**

**Maximum possible progress:**

- Matrix multiplication (DAM) \([\text{Irony+Toledo+Tiskin’04}]\) :
  Every time you have a full cache \((M)\), you can do \(O(M^{3/2})\) multiplications.

- We refer to these as **piece-wise progress bounds**.
  - History-less

- How to **port** these DAM bounds to use on arbitrary profiles?
Theorem: Limited to the inner squares, RMM performs optimally in the CA model.

Sudden drops in memory makes the squares bad!
Memory can increase by $\leq 1$ page per time step.

A 4-memory, 4-speed augmented RMM makes as much progress on the $S_i$ as an OPT does on the $S_{i+1}$.
Theorem: A \((4,4)\)-augmented RMM makes as much progress in \(U_{j=1}^{i} S_j\) as the OPT does on \(U_{j=1}^{i+1} S_j\) and is hence optimally cache-adaptive.

- The 4-memory aug. not needed for CO algorithms!
Other Applications of the Toolbox

**Theorem:** If a problem has a *piece-wise progress bound*, then an optimal **COR form** CO algorithm is optimal in the CA model.

- Matrix transpose, all pairs shortest paths, Gaussian elimination

**Also applicable to:**

- **Sorting progress bound** (not in COR form)
  
  **Theorem:** Lazy funnel sort [brodalfagerberg’03] is an optimal CA algorithm.
We say that an algorithm is in **Constant-Overhead Recursion (COR)** form if

- The algorithm just makes recursive sub-calls and besides that makes only $O(1)$ additional page references.

\[
T(N) = \begin{cases} 
O(N/B + 1) & \text{if } N \leq M, \\
 f(N)T(g(N)) + O(1) & \text{otherwise.} 
\end{cases}
\]

subject to: $f(N) \geq 2$, and $1 \leq g(N) < N$
Many cache-oblivious algorithms are optimally cache-adaptive
- But not all CO algorithms are optimally CA (see paper).

Cache obliviousness makes it easier to deal with changing memory size.

In future we would like to:
- Extend the class of cache-oblivious algorithms that are optimally cache-adaptive.
- Consider relaxed notions of optimality
  - Currently: optimal against algorithms that have foresight.