Cache-Adaptive Analysis

Michael A. Bender¹  Erik Demaine⁴  Roozbeh Ebrahimi¹
Jeremy T. Fineman³  Rob Johnson¹  Samuel McCauley¹
Andrea Lincoln⁴  Jayson Lynch⁴

¹ Stony Brook University
³ Massachusetts Institute of Technology
⁴ College of William and Mary
Available Memory Can Fluctuate in Real Systems

Memory fluctuations are common
- Jobs starting and stopping
- Irregular parallel programs
- Any time-sharing system

Performance can be lost when algorithms can't adapt to changes in available memory
- Thrashing (when available memory shrinks)
- Underutilization (when available memory grows)
Adapting to Memory Changes: Empirical

Database papers on adaptive sorting or joins:

• *Empirical good, but not provably good.*
• *Rarely present in production systems, despite the need.*

[Pang, Carey, Livny, VLDB 93], [Zeller+Gray VLDB 90], [Zhang+Larson VLDB 97], [Zhang+Larson, CASCON 96], [Pang, Carey, Livny, SIGMOD/COMAD 93], [Graefe 13]
Barve and Vitter [98, FOCS 99] generalize the I/O model [Aggarwal+Vitter ’88] to allow RAM to change size.

- These are hard and technically sophisticated results (sorting, FFT, matrix multiplication, etc).
- There’s been little followup work over the last 15 years.

*It's hard to write memory-adaptive code and harder to prove bounds about it.*
We can design cache-adaptive algorithms using cache-oblivious algorithms.
Results

Tools for cache-adaptive analysis.
- Extension to external-memory and cache-oblivious models.
- Square profiles and inductive charging
- Worst-case profile analysis
- Machinery for porting progress bounds from DAM to CA model

Characterization theorem for when CO algorithm is CA
- Covers many Akra-Bazzi-style divide-and-conquer algorithms, e.g.
  - Matrix multiplication (two versions, one is CA, one is not)
  - Matrix transpose
  - Jacobi multi-pass filter
  - All-pairs shortest paths
  - Edit distance
  - Longest common substring

Typical Master-theorem-style CO algorithms are either optimal or $\log N$ off.

Cache-oblivious FFT is not CA, but is at most $\log \log N$ off.
Proof that Lazy Funnel Sort \cite{Brodal, Fagerberg 02} is cache adaptive.

Paging results when the cache changes sizes.
- Farthest-in-future is still optimal (cf. \cite{Belady 66}).
- LRU with 4-memory and 4-speed augmentation is competitive with OPT.
- LRU is constant-competitive even if cache hits are not free.
  - And even if OPT gets to perform prefetching.
Generalizes Disk Access Machine (DAM) model \cite{Aggarwal+Vitter '88}.

- Data is transferred in blocks between RAM and disk.
- Performance is measured in terms of block transfers.

Now size of internal memory is a function of time.

- Can change arbitrarily
- Can change without advance notice
Ideal-cache model: DAM model + automatic paging

• Contents of cache are managed by a separate paging algorithm.
• Time bounds are parameterized by $B$, $M$, $N$.
• Goal: Minimize # of block transfers $\approx$ time.

Beautiful restriction:

• Parameters $B$, $M$ are unknown to the algorithm or coder.
• An optimal CO algorithm is universal for all $B$, $M$, $N$. 
Example: Recursive Matrix Multiplication is Cache-Oblivious

$N \times N$ matrix multiplication: 8 multiply-adds of $N/2 \times N/2$ matrices:

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

\[
T(N) = \begin{cases}
O\left(\frac{N^2}{B}\right) & \text{if } N^2 = O(M) \\
8T\left(\frac{N}{2}\right) + O\left(\frac{N^2}{B}\right) & \text{otherwise}
\end{cases}
\]

\[
= O\left(\frac{N^3}{B\sqrt{M}}\right)
\]
A **progress bound** \( \rho(M) \) upper-bounds the amount of useful work that any algorithm can accomplish given \( M \) memory and \( M/B \) I/Os.

A **progress requirement function** \( R(N) \) lower bounds the amount of work required to solve all problems of size \( N \).

\[
\rho(M) = O\left(M^{3/2}\right)
\]

\[
R(N) = O\left(N^3\right)
\]

Example: Hong and Kung's progress bound for matrix multiplication [Hong and Kung 81]
Why Recursive Matrix Multiply is Optimal in the DAM Model

So no algorithm can have running time less than

\[ \Omega \left( \frac{R(N)}{\rho(M)} \times \frac{M}{B} \right) = \Omega \left( \frac{N^3}{M^{3/2}} \times \frac{M}{B} \right) = \Omega \left( \frac{N^3}{B \sqrt{M}} \right) \]

\[ \rho(M) = O(M^{3/2}) \]

\[ R(N) = O(N^3) \]

CO matrix multiply running time: \[ T(N) = O \left( \frac{N^3}{B \sqrt{M}} \right) \]
What Can Go Wrong in the CA Model?

\[ A \ast B \]

\[
R_1 = \begin{pmatrix}
A_{11} \ast B_{11} & A_{11} \ast B_{12} \\
A_{21} \ast B_{11} & A_{21} \ast B_{12}
\end{pmatrix}
\]

\[
R_2 = \begin{pmatrix}
A_{12} \ast B_{21} & A_{12} \ast B_{22} \\
A_{22} \ast B_{21} & A_{22} \ast B_{22}
\end{pmatrix}
\]

return \( R_1 + R_2 \)

\[ 8 \text{ recursive calls} \]
\[ \Theta \left( \frac{N^2}{B} \right) \text{ I/Os} \]

**No matter how much memory is available.**
What Can Go Wrong in the CA Model?

We can recursively construct a “bad” profile $W_N$ that

- Has lots of memory when algorithm doesn't need it
- Little memory when algorithm could use it

$$8 \text{ copies of } W_{N/2} \ldots W_{N/2} W_{N/2}$$

$$\rho\left(\frac{N^2}{B}\right)$$

$$A_{11}B_{11} \quad A_{11}B_{12} \quad \ldots \quad A_{22}B_{22} \quad R_1 + R_2$$

$$\Theta\left(\frac{N^2}{B}\right)$$

$N^2/B$ IOs

$N^2$
What Can Go Wrong in the CA Model?

$W_N$ supports a lot of progress:

$$\rho(W_N) = 8 \rho(W_{N/2}) + \Theta(\rho(N^2))$$
$$= 8 \rho(W_{N/2}) + \Theta(N^3)$$
$$= \Theta(N^3 \log N)$$
What Can Go Wrong in the CA Model?

$W_N$ supports a lot of progress:

$$
\rho(W_N) = 8 \rho(W_{N/2}) + \Theta(\rho(N^2))
= 8 \rho(W_{N/2}) + \Theta(N^3)
= \Theta(N^3 \log N)
$$

CO matrix multiply makes only $O(N^3)$ progress, so it is not optimal.
Write down recurrence relation for the algorithm:

\[ T(N) = aT(N/b) + \Theta(N^c/B) \]

Derive new recurrence by replacing additive terms with progress bound \( \rho \):

\[ S(N) = aS(N/b) + \Theta(\rho(N^c)) \]

If \( S(N) = O(R(N)) \), then the algorithm is optimally progressing.
This Recipe is General

Covers many different divide-and-conquer forms

- Master Theorem
- Akra-Bazzi
- Mutually recursive functions
- Plus others (e.g. cache-oblivious FFT)

Can answer several different questions

- Is an algorithm optimal?
- Is it not optimal?
- How far is it from optimal?

And it's easy!

- Just manipulating and solving recurrence relations
Conclusions

The CA model works.

- It is general enough to describe real systems.
- It is easy to work with.

Cache-oblivious algorithms are a good way to make CA algorithms.

- Many cache oblivious algorithms are CA.
- And are pretty close to optimal otherwise.