Extreme Memoization
Everything in a LUT!

Pratyush Patel
Luis Ceze
Extreme *Meme*-oization
Everything in a LUT!

Pratyush Patel
Luis Ceze
Extreme *Meme*-oization
Everything in a LUT!

Memoization  Meme-oization
CPUs aren't getting much faster…

Dennard Scaling
1974—2006

Moore's Law
1965—??
Memory capacity scaling

Capacity (GB)

Memory Technology

MRAM  DDR1  DDR2  HBM2  DDR3  DDR4  NVM
iPhone storage scaling

<table>
<thead>
<tr>
<th>iPhone Generation</th>
<th>Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>2² GB</td>
</tr>
<tr>
<td>2019</td>
<td>2⁹ GB</td>
</tr>
</tbody>
</table>

- **iPhone 1 (2007)**: 2² GB
- **iPhone 11 Pro (2019)**: 2⁹ GB
iPhone storage scaling

<table>
<thead>
<tr>
<th>iPhone Generation</th>
<th>Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3G</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
</tr>
<tr>
<td>6</td>
<td>400</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
</tr>
<tr>
<td>8</td>
<td>600</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>11P</td>
<td></td>
</tr>
</tbody>
</table>

2007
iPhone 1
$2^2$ GB

2019
iPhone 11 Pro
$2^9$ GB
At this rate, the 2055 iPhone 47 will store $\sim 2^{75}$ bytes: the total data *ever generated* thus far.

*assumes future iPhone generations are sequential natural numbers*
Datacenter and mobile networks scaling
Datacenter and mobile networks scaling

![Graph: X86 servers by Ethernet connection speed.](chart)

**Server# (millions)**

- **100M**
- **1G**
- **10G**
- **40G**
- **100G**

Years:
- 2000
- 2001
- 2002
- 2003
- 2004
- 2005
- 2006
- 2007
- 2008
- 2009
- 2010
- 2011
- 2012
- 2013
- 2014
- 2015
- 2016
- 2017
- 2018
- 2019
- 2020

Key:
- **10 years transitions for 1GbE**
- **5 years for 10GbE**
- **5 years for 40GbE**
Datacenter and mobile networks scaling

x86 Servers by Ethernet Connection Speed

- 100M
- 1G
- 10G
- 40G
- 100G

Server # (millions)

Bandwidth (Mbps)

- 1G
- 2G
- 3G
- 4G
- 5G

10 years transitions for 1GbE
5 years for 10GbE
5 years for 40GbE
So let’s memoize *almost* everything!
So let’s memoize *almost* everything!

**Extreme Memoization**: store most computation performed and share it globally rather than recomputing!
So let’s memoize *almost* everything!

**Extreme Memoization**: store most computation performed and share it globally rather than recomputing!

Compute $f(x)$
So let’s memoize *almost* everything!

**Extreme Memoization**: store most computation performed and share it globally rather than recomputing!
So let’s memoize *almost* everything!

**Extreme Memoization**: store most computation performed and share it globally rather than recomputing!
So let’s memoize *almost* everything!

**Extreme Memoization**: store most computation performed and share it globally rather than recomputing!
When will this make sense?

- CPU compute energy
- Memoization energy

Energy vs. Computational Complexity
When will this make sense?

- CPU compute energy
- Memoization energy

![Graph showing energy vs computational complexity]
When will this make sense?

- CPU compute energy
- Memoization energy

Energy vs. Computational Complexity graph.
When will this make sense?

- CPU compute energy
- Memoization energy
When will this make sense?

- CPU compute energy
- Memoization energy

Energy

Minimum access energy

Computational Complexity
When will this make sense?

When storage + access energy is lower than computation energy

- CPU compute energy
- Memoization energy

Minimum access energy

Breakeven point

Computational Complexity

Energy
Computation redundancy wastes energy
Computation redundancy wastes energy

# of worldwide Internet users

<table>
<thead>
<tr>
<th>Year</th>
<th>Internet Population (in billions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>0.00</td>
</tr>
<tr>
<td>2000</td>
<td>0.50</td>
</tr>
<tr>
<td>2005</td>
<td>1.00</td>
</tr>
<tr>
<td>2010</td>
<td>2.00</td>
</tr>
<tr>
<td>2015</td>
<td>3.00</td>
</tr>
<tr>
<td>2020</td>
<td>5.00</td>
</tr>
</tbody>
</table>
Computation redundancy wastes energy

# of worldwide Internet users

Potentially a lot of redundancy!
Computation redundancy wastes energy
We compute by memorization!! 😁

Source: Harvard University Youtube Channel
We compute by memorization!! 😁

Source: Harvard University Youtube Channel
Memoize function, inputs, and outputs

\[ f(x) \rightarrow v \]
Memoize function, inputs, and outputs

A global lookup table
f( ) implemented in any PL
Hash a language-agnostic IR!

\[ f(x) \rightarrow #(IR) \rightarrow v \]

A global lookup table
Many ways to write the same function

f(x) \rightarrow \#(IR) \rightarrow v

f(x) = g(x) = h(x) ...
Maybe use equivalence graphs?

$f(x) \rightarrow \#(\text{IR egraph?}) \rightarrow v$
Maybe use equivalence graphs?
Store **key-value** pair in the global LUT

\[ f(x) \rightarrow \#(\text{IR egraph?}) \rightarrow v \]
Too big a (lookup) table?
Too big a (lookup) table?

Worldwide compute capacity is $\sim 10^{20}$ FLOPS!
We might need a helluva lot of Hellabytes

1 Hellabyte = $10^{27}$ bytes
Approximation can help...
Approximation can help...

Coalesce entries based on output sensitivity
Approximation can help...

There's prior work on this!
Fuzzy memoization for floating-point multimedia applications [TC '05]
Temporal approximate function memoization [IEEE Micro '18]
Approximation can help...

There's prior work on this!
Fuzzy memoization for floating-point multimedia applications [TC ’05]
Temporal approximate function memoization [IEEE Micro ’18]

Coalesce entries based on output sensitivity

Still too big?
Cloud-scale DNA storage to the rescue!

Source: Max Willsey and Luis Ceze, Mega-Microfluidics, WACI 2019
Cloud-scale DNA storage to the rescue!

Storing a Hellabyte would require 3 trillion kg of SSDs, but only 2500 kg of DNA!

Source: Max Willsey and Luis Ceze, Mega-Microfluidics, WACI 2019
Cloud-scale DNA storage to the rescue!

Storing a Hellabyte would require 3 trillion kg of SSDs, but only 2500 kg of DNA!

Source: Max Willsey and Luis Ceze, Mega-Microfluidics, WACI 2019
Sharing is caring
Sharing is caring

Share $f(x)$ globally
Sharing is caring, *but is it safe?*

Share $f(x)$ globally
Sharing is caring, *but is it safe?*

- Share $f(x)$ globally
- Steal $x$ and $f(x)$!
Sharing is caring, *but is it safe?*

- **Share f(x) globally**
- **Steal x and f(x)!”**
Sharing is caring, *but is it safe?*

**Security and privacy implications**
Attacker could brute force inputs to perform timing attacks on sensitive data
What happens to open-source code and cryptography?

Share $f(x)$ globally

Steal $x$ and $f(x)$!
A sampling of new research challenges

- Cross-language program equivalence
- Security and privacy
- Efficient networks
- Global data delivery
- High-density storage
- Large-scale hash functions
- Fuzzy lookup tables
- Accurate energy models for functions
- Your idea here!
Extreme Memoization
Everything in a LUT!

Thanks!
Pratyush Patel — patelp1@cs.uw.edu