Mining Hot Calling
Contexts in Small Space

Joint work with:

Camil Demetrescu
Dept. Computer and System Sciences
Sapienza University of Rome

Irene Finocchi
Dept. Computer Science
Sapienza University of Rome
A **calling context** is a sequence of routine calls that are currently active on the run-time stack (=> leads to a program location)
### Space required to maintain whole CCT?

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>N (stream length)</th>
<th># routines</th>
<th># distinct contexts (CCT nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oocalc</td>
<td>551 472 065</td>
<td>30 807</td>
<td>48 310 585</td>
</tr>
<tr>
<td>firefox</td>
<td>625 133 218</td>
<td>6 756</td>
<td>30 294 063</td>
</tr>
<tr>
<td>gimp</td>
<td>805 947 134</td>
<td>4 146</td>
<td>26 107 261</td>
</tr>
<tr>
<td>inkscape</td>
<td>675 915 815</td>
<td>6 454</td>
<td>13 896 175</td>
</tr>
<tr>
<td>audacity</td>
<td>924 534 168</td>
<td>6 895</td>
<td>13 131 115</td>
</tr>
<tr>
<td>ark</td>
<td>216 881 324</td>
<td>9 933</td>
<td>8 171 612</td>
</tr>
<tr>
<td>vlc</td>
<td>125 436 877</td>
<td>5 692</td>
<td>3 295 907</td>
</tr>
</tbody>
</table>

921 MB

Just for a few minutes session!!
What fraction of the CCT is really useful?

Top 10% of hottest CCT nodes cover 97% of calls

Benchmark:
A more space-efficient approach

Our work aims to identify most frequent contexts

Space-efficient approach

Keep information about most frequent contexts only

Ignore on the fly info about contexts with low frequency
Hot Calling Context Tree

HCCT

CCT

Hot nodes corresponding to frequent calling contexts

Cold nodes that must be stored anyway
Cold nodes require little additional space

HCCT representing top 8% hottest nodes saves 90% of space of whole CCT

Benchmark:
Building the HCCT on-the-fly

Running program

Stream of call/return events

HCCT analysis tool

Track current path + add nodes to HCCT

Stream of node IDs

Counting algorithm

Stream of node IDs dropping below threshold

On-line pruning of HCCT
Frequent items in a data streaming model

Given a stream of \( N \) items, find those that occur most frequently

E.g., find all items occurring more than 1% of the time

Formally hard in small space, so find approximate solution
Approximate counting approaches

\(\epsilon\)-deficient frequent items algorithms:

**Input:**
- stream of \(N\) items
- frequency threshold \(0 \leq \varphi \leq 1\)
- error parameter \(0 \leq \epsilon \leq 1\) \((\epsilon \ll \varphi)\)

**Output:**
- all items with frequency \(\geq \varphi N\)
- none with frequency \(< (\varphi - \epsilon) N\)
- estimated frequencies with error \(\pm \epsilon N\)

no false negatives
Building the \((\varphi-\epsilon)\)-HCCT on the fly

Running program

\textbf{Stream of call/return events}

\((\varphi-\epsilon)\)-HCCT analysis tool

Track current path + add nodes to \((\varphi-\epsilon)\)-HCCT

\textbf{Stream of node IDs}

Frequent items streaming algorithm

\textbf{Stream of node IDs}

On-line pruning of \((\varphi-\epsilon)\)-HCCT

\textbf{Stream of node IDs dropping below threshold}
The whole picture

Mining Hot Calling
Contexts in Small Space

False positives

(φ-ε)-HCCT

HCCT

CCT
Approximate counting algorithms

Space saving

Metwally, Agrawal, and El Abbadi
[ACM Trans. on Database Systems (TODS), 2006]

Space = $O\left(\frac{1}{\varepsilon}\right)$ (provably optimal)

Efficient engineered implementation

Lossy counting

Manku and Motwani [VLDB 2002]

Space = $O\left(\frac{1}{\varepsilon \log(\varepsilon N)}\right)$

O(1/ε) in our experiments
Space statistics

Mining Hot Calling
Contexts in Small Space

Space comparison with full CCT

Size of (φ-ε)-HCCT as a percentage of full CCT size. For most benchmarks space occupancy reduced to 1/100.

Percentage of cold, hot and false positive nodes

Benchmarks (from left to right): ark, audacity, inkscape, oocalc, vlc. For each benchmark two bars, related to LC (left) and SS (right).
Mining Hot Calling
Contexts in Small Space

Accuracy statistics

**Avg counter error**
Average error among hot elements (% of the true frequency).

**Max counter error**
Maximum error among hot elements (% of the true frequency).
Final remarks

Calling context analysis:

calling context trees may be very large and difficult - sometimes impossible - to analyze in several applications

Our work:

- provides an actual space-efficient approach
- exploits theoretical guarantees of streaming algorithms, achieving high precision in counters
- can be easily integrated with other techniques, since it's orthogonal to performance aspects
  e.g., sampling, adaptive bursting (see paper)