Query-Directed Adaptive Heap Cloning for Optimizing Compilers

Yulei Sui, Yue Li, Jingling Xue

Programming Languages and Compilers Group
School of Computer Science and Engineering
University of New South Wales, Australia

February 25, 2013
Heap cloning

Statically distinguishing heap objects by call paths
Heap cloning

Statically distinguishing heap objects by call paths

```c
int main()
{
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}

int* getMem()
{
    return malloc(10);
}
```
Heap cloning

Statically distinguishing heap objects by call paths

```c
int main()
{
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}

int getMem()
{
    return malloc(10);
}
```

Program execution
Heap cloning

Statically distinguishing heap objects by call paths

```c
int main()
{
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}

int* getMem()
{
    return malloc(10);
}
```

Program execution
Heap cloning

Statically distinguishing heap objects by call paths

```c
int main()
{
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}

int* getMem()
{
    return malloc(10);
}
```

Program execution
Heap cloning

Statically distinguishing heap objects by call paths

```c
int main()
{
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}
```

```c
int* getMem()
{
    return malloc(10);
}
```

Program execution

Points-to relations without heap cloning
Heap cloning

Statically distinguishing heap objects by call paths

```c
int main()
{
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}

int* getMem()
{
    return malloc(10);
}

buffer1 = getMem();
buffer2 = getMem();
```

Program execution

Points-to relations without heap cloning

Points-to relations with heap cloning
Call graph of 176.gcc (230.4KLOC)

#Procedures: 2256  #Pointers: 134380  #Calling Contexts: $1.2 \times 10^5$

Context-sensitive heap cloning can be costly!
K-callsite-sensitive heap cloning

Call Graph with K-callsite-sensitive heap cloning

[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]
K-callsite-sensitive heap cloning

Call Graph with K-callsite-sensitive heap cloning
[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]
K-callsite-sensitive heap cloning

Call Graph with K-callsite-sensitive heap cloning

[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]
K-callsite-sensitive heap cloning

Call Graph with K-callsite-sensitive heap cloning

[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]
K-callsite-sensitive heap cloning

Call Graph with K-callsite-sensitive heap cloning

[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]
K-callsite-sensitive heap cloning

Call Graph with K-callsite-sensitive heap cloning

[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]
K-callsite-sensitive heap cloning

[Call Graph with K-callsite-sensitive heap cloning]

[Nystrom-SAS’04, Nystrom-PASTE’04, Lhotak-CC’06, Xu-ISSTA’08]

Analysis time of 176.gcc with Andersen-style context sensitive heap cloning
22x slower than Open64 "-O2" compile time
• Is full heap cloning overkill (relative to a client’s needs)?
• Is *full heap cloning* overkill (relative to a client’s needs)?

• Is *k-callsite sensitive cloning* the best solution?
Alias Query

- Whether two expressions may represent the same memory location.
Alias Query

- Whether two expressions may represent the same memory location.
- For example: \langle *buffer1,*buffer2 \rangle
  - Alias Without Heap Cloning
Alias Query

- Whether two expressions may represent the same memory location.
- For example: \( \langle *buffer1,*buffer2 \rangle \)
  - Alias Without Heap Cloning
  - Not-Alias Heap Cloning
Analysis precision for answering alias queries

Percentage of must-not aliases disambiguated among the queries issued by WOPT with k-callsite-sensitive heap cloning
A close look at 255.vortex’s call graph
Goal

- Can we enable heap cloning only where it is necessary?
- Can we achieve the same precision as full heap cloning according to a client’s needs?
Our QUDA framework
QUery-Directed Adaptive heap cloning
Query-directed adaptive heap cloning

Query-Directed Adaptive Heap Cloning

procedure
heap object
k=0
k=1
k=2
k=3
bb
<*p,*q>
Alias Query
p
p
q
q
must-not-alias

query

may-alias
not heap-related
heap-related

All done
Can't clone further
Query-directed adaptive heap cloning

Query-Directed Adaptive Heap Cloning

procedure
heap object
k=0
k=1
k=2
k=3
bb
<*p,*q>
Alias Query
must-not-alias
not heap-related
heap-related
May-alias
p
q
All done
Can't clone further
Query-directed adaptive heap cloning

Procedure

Heap object

k=0

k=1

k=2

k=3

bb

Alias Query

<\*p,\*q>

Alias Query

p

p

q

q

must-not-alias

may-alias

may-alias

not heap-related

heap-related

All done

Can't clone further

Query-Directed Adaptive Heap Cloning

procedure

heap object
Query-directed adaptive heap cloning

Procedure

Heap object

k=0

k=1

k=2

k=3

bb

Alias Query

< * p, * q >

must-not-alias

procedure

heap object

Query-Directed Adaptive Heap Cloning

k=0

must-alias

heap-related

All done

Can't clone further

Query-Directed Adaptive Heap Cloning
Query-directed adaptive heap cloning

Alias Query

<*p,*q>

may-alias

All done
Can't clone further

Query-Directed Adaptive Heap Cloning
Query-directed adaptive heap cloning

Alias Query

<*p,*q>

may-alias
not heap-related

All done
Can't clone further
Query-directed adaptive heap cloning

Alias Query
\(<*p,*q>\)

may-alias
not heap-related

Query-Directed Adaptive Heap Cloning

procedure
heap object
k=0
k=1
k=2
k=3

bb

<*p,*q>

Alias Query

must-not-alias

may-alias
heap-related

All done
Can't clone further

k=0

Query-Directed Adaptive Heap Cloning

procedure
heap object

<*p,*q>

Alias Query

must-not-alias

may-alias
not heap-related

may-alias
heap-related

All done
Can't clone further

k=0

Query-Directed Adaptive Heap Cloning
Query-directed adaptive heap cloning

Query-Directed Adaptive Heap Cloning

Procedure
Heap Object

k=0
k=1
k=2
k=3

bb

<\*\*p,\*q>

Alias Query

may-alias

heap-related

procedure

heap object

k=0

Query-Directed Adaptive Heap Cloning

All done
Can't clone further
Query-directed adaptive heap cloning

Alias Query

<*p,*q>

may-alias

heap-related

procedure

heap object

k=0

All done
Can't clone further

Query-Directed Adaptive Heap Cloning

@ Yulei Sui, CGO 2013, 25/02/2013
PLC, UNSW
Query-directed adaptive heap cloning

Query-Directed Adaptive Heap Cloning

procedure
heap object
k=0
k=1
k=2
k=3
bb
<*p,*q>
Alias Query
must-not-alias
heap-related
may-alias
not heap-related

All done
Can't clone further
Query-directed adaptive heap cloning
Query-directed adaptive heap cloning

Query-Directed Adaptive Heap Cloning

Alias Query
<p, q>

may-alias
heap-related

All done
Can't clone further
Query-directed adaptive heap cloning

Alias Query
<*p,*q>

may-alias
heap-related

All done
Can't clone further

Query-Directed Adaptive Heap Cloning

procedure
heap object
k=0
k=1
k=2
k=3

*p,*q

must-not-alias

not heap-related

may-alias

heap-related

procedure
heap object
k=0
k=1

© Yulei Sui, CGO 2013, 25/02/2013
PLC, UNSW
Query-directed adaptive heap cloning

Procedure heap object

k=0
k=1
k=2
k=3

Alias Query

<*p,*q>

May-alias

heap-related

Must not alias

All done

Can't clone further
Query-directed adaptive heap cloning

Alias Query

<*>p,*q>

may-alias
heap-related

procedure
heap object

k=0
k=1
k=2

All done
Can't clone further

Query-Directed Adaptive Heap Cloning
Query-directed adaptive heap cloning

Procedure

Heap object

k=0

k=1

k=2

k=3

Alias Query

<*p,*q>

p

q

must-not-alias

may-alias

heap-related

not heap-related

All done
Can't clone further

Query-Directed Adaptive Heap Cloning
QUDA: QUery-Directed Adaptive heap cloning

Alias Queries

Heap-Aware Pointer Solver

Selecting Candidate Heap Objects

Adaptive Update

Cloning Level K

K++
Heap-aware pointer analysis

\[
x \rightarrow g \\
p \rightarrow o
\]
Heap-aware pointer analysis

\[ x \rightarrow (\text{true, } g) \]
\[ p \rightarrow (h_0, \ o) \]
Heap-aware pointer analysis

\[
x \rightarrow (\text{true}, \; g) \\
p \rightarrow (h_o, \; o) \\
q \rightarrow (h_o, \; o)
\]

\[
*p = x; \\
y = *q;
\]
Heap-aware pointer analysis

Constraint Graph:

\[
\begin{align*}
    x &\rightarrow (\text{true, } g) \\
p &\rightarrow (h_o, o) \\
q &\rightarrow (h_o, o) \\
\end{align*}
\]

\[
\begin{align*}
    *p &\rightarrow x; \\
y &\rightarrow *q;
\end{align*}
\]
Heap-aware pointer analysis

\[ x \rightarrow (\text{true, } g) \]
\[ p \rightarrow (h_0, \ o) \]
\[ q \rightarrow (h_0, \ o) \]

\[ *p = x; \]
\[ y = *q; \]
Heap-aware pointer analysis

Constraint Graph:

\[ x \rightarrow (\text{true, g}) \]
\[ p \rightarrow (h_o, o) \]
\[ q \rightarrow (h_o, o) \]

\[ *p = x; \]
\[ y = *q; \]
Candidate heap objects selection

Constraint Graph:

Copy ← Store ← Load

Alias Query

\(\langle \ast x, \ast y \rangle\)

\[
\begin{align*}
\text{pts}(x) &= \{\text{true}, g\} \\
\text{pts}(y) &= \{h_0, g\}
\end{align*}
\]
Candidate heap objects selection

Constraint Graph:

Alias Query

\[ \langle *x, *y \rangle \]

\[ \text{pts}(x) = \{ \text{true, } g \} \]
\[ \text{pts}(y) = \{ h_0, g \} \]

Candidate Heap Object \( \{ o \} \)
Adaptive update

Candidate Heap Object \( \{ o \} \)

Constraint Graph:

Copy \( \longleftrightarrow \) Store \( \leftarrow \) Load \( \rightarrow \)

\[
\begin{align*}
\text{Candidate Heap Object} & \quad \{ o \} \\
\text{Constraint Graph:} & \\
\text{Copy} & \leftarrow \text{Store} \leftarrow \text{Load} \\
\end{align*}
\]
Adaptive update

Constraint Graph:

Candidate Heap Object \{o\}
Next round resolution

Constraint Graph:

Copy ← Store ← Load

Alias Query
⟨∗x, ∗y⟩

Not-alias!
QUDA framework in Open64

**Diagram: QUDA Framework**

1. **Program** → **FE** → **IPL**
2. **IPA (Heap-Aware Pointer Analysis)**
   - **Top-Down**
   - **Bottom-up**
3. **Candidate Query Selection** → **Adaptive Update**
4. **Constraint Graphs**
5. **Alias Tag** → **WOPT** → **CG**
6. **ELF files (*.G, *.I)** → **write into**
7. **read**

**Additional Notes:**
- Cloning Level K++
- The framework involves multiple stages of analysis and transformation, with integration points for adaptive update and candidate query selection.
- The use of ELF files is highlighted, indicating a focus on binary or executable files in the context of program analysis and optimization.
Analysis times of FULCRA and QUDA

Analysis time normalized with respect to Open64’s compile times (-O2)
Heap objects reduced by QUDA over FULCRA

Number of heap objects reduced by QUDA over FULCRA in percentage terms
Heap distribution with full heap cloning (175.vpr)
Heap distribution with QUDA (175.vpr)
Alias queries to be answered at each iteration

% of Queries to Be Answered vs. k-Callsite-Sensitivity Heap Cloning

- gcc
- hmmer
- jpeg
- mesa
- perlbmk
- rasta
- sendmail
- vortex

% of Analysis Time

@ Yulei Sui, CGO 2013, 25/02/2013
PLC, UNSW
Analysis time per iteration over the total

% of Analysis Time

k-Callsite-Sensitivity Heap Cloning

gcc
hmmer
jpeg
mesa
perlbmk
rasta
sendmail
vortex

% of Queries to Be Answered

k-Callsite-Sensitivity Heap Cloning

% of Analysis Time
Conclusion

Novel heap cloning approach: same precision as full heap cloning but significantly more scalable

- Heap-aware analysis
- Query-directed
- Adaptive

Challenges and opportunities:

- Iterative compilation (prioritising queries in hot functions)
- Bug detection (scaling precise pointer analysis)