Query-Directed Adaptive Heap Cloning for Optimising Compilers

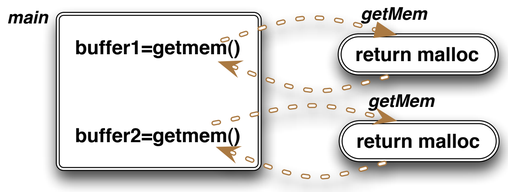
UNSW AUSTRALIA
Never Stand Still | Faculty of Engineering | School of Computer Science and Engineering

Heap Cloning

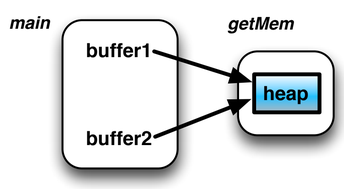
```
int main(){
    int* buffer1 = getMem();
    int* buffer2 = getMem();
}

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

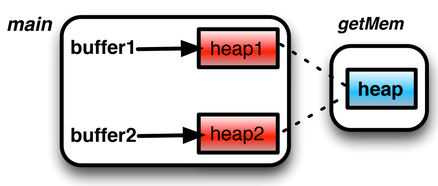
C program Code



Program execution

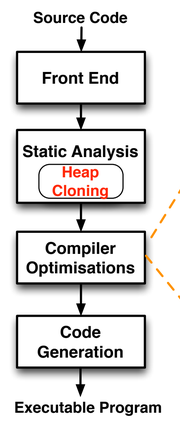


Static pointer analysis ***without*** heap cloning

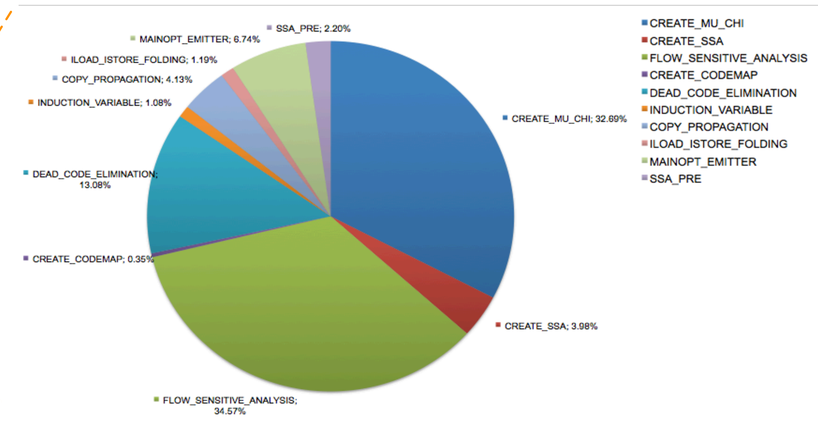


Static pointer analysis ***with*** heap cloning

Compiler Optimisations

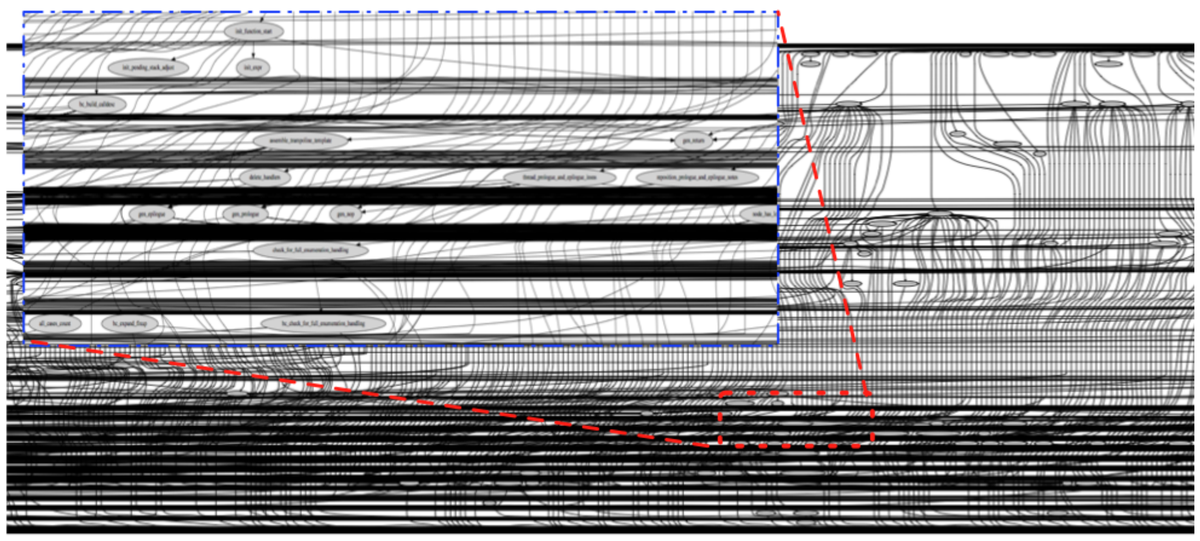


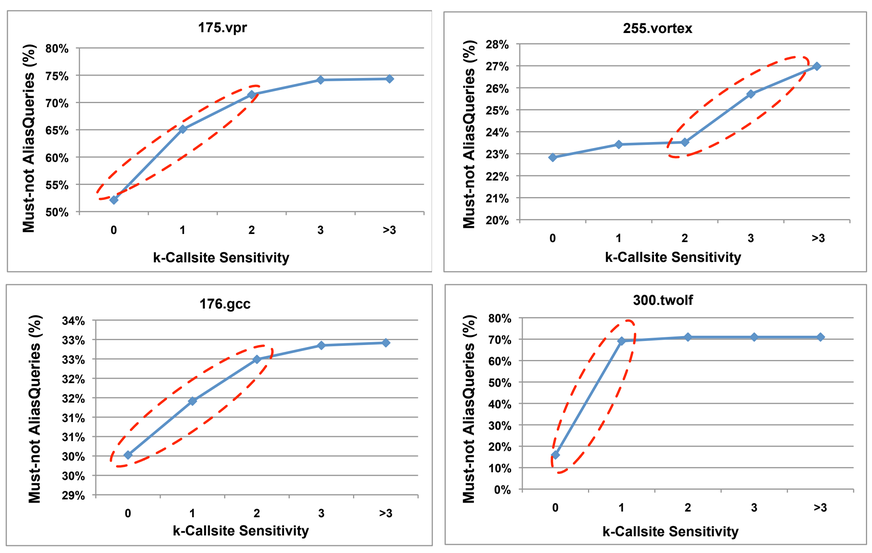
A Typical Compiler Workflow



A total of 496304 alias queries generated from optimisations in Open64 compiler for analysing CPU2000 C benchmarks

Motivation





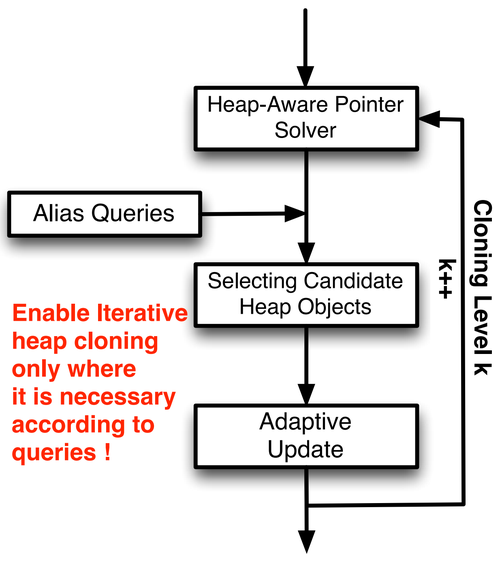
Full heap cloning is overkill !

and

Different programs require different levels of precision !

Very costly to reason about full heap cloning on large size program with high-density call graph **Compiler optimisations of diverse programs benefit in different precision levels of heap cloning**

Approach

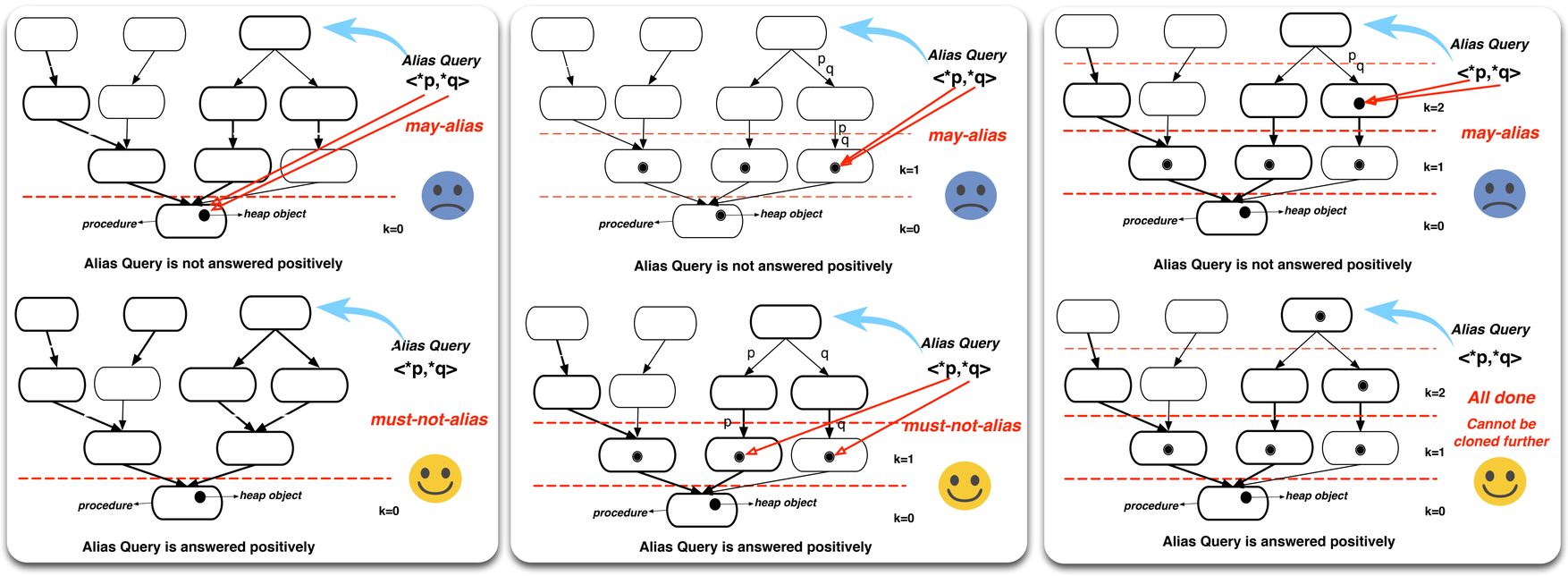


Enable Iterative heap cloning only where it is necessary according to queries !

QUDA: **QU**ery-**D**irected **A**daptive heap cloning

QUDA performs heap cloning on parts of the program according to clients' need in a demand-driven fashion

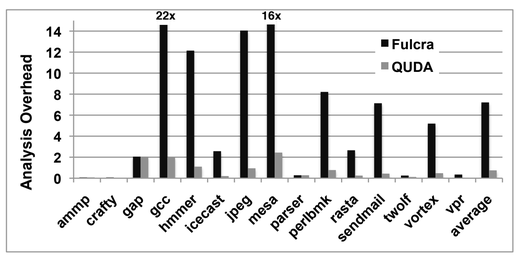
An Example



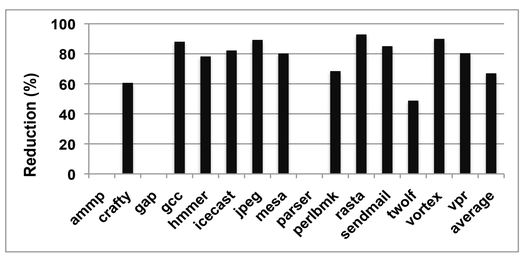
QUDA heap cloning (1st Iteration) *QUDA heap cloning (2nd Iteration)* *QUDA heap cloning (3rd Iteration)*

QUDA performs k-callsite-sensitive heap cloning iteratively, starting with k = 0 (without heap cloning), so that an abstract heap object is cloned at iteration k = i + 1 only if some alias queries that are not answered positively at iteration k = i may now be answered more precisely.

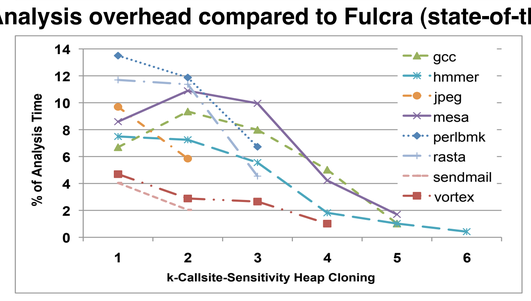
Results



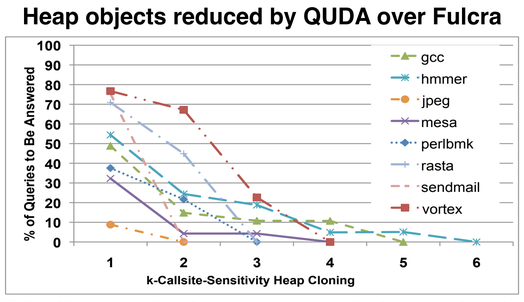
Analysis overhead compared to Fulcra (state-of-the-art)



Heap objects reduced by QUDA over Fulcra

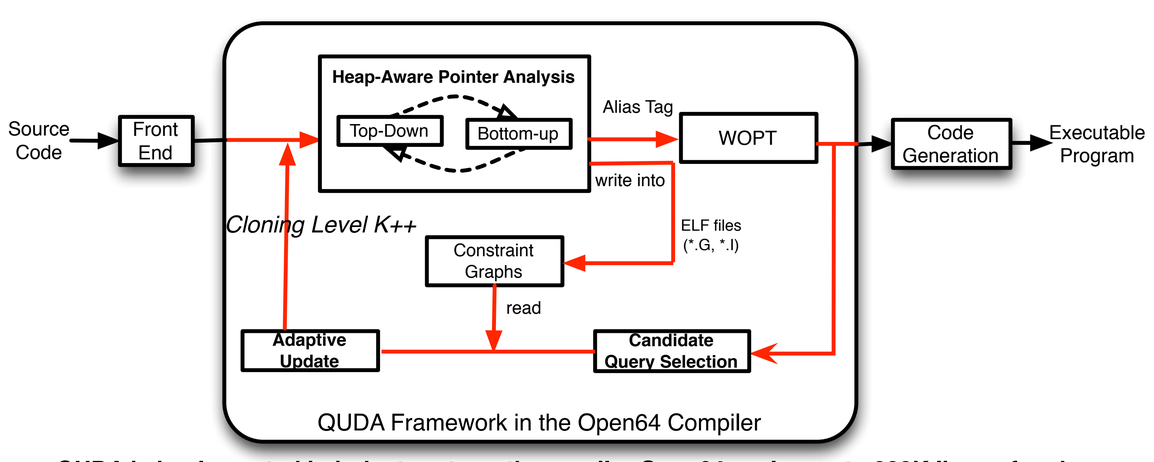


QUDA analysis time per iteration over the total



QUDA alias queries to be answered at each iteration

Tool Framework



QUDA is implemented in industry-strength compiler Open64 scales up to 200K lines of code.
It has the same precision as the state-of-the-art, but is significantly more scalable.
For 10 SPEC2000 C benchmarks and 5 C applications (totalling 840 KLOC) evaluated,
QUDA takes only 4+ minutes but exhaustive heap cloning takes 42+ minutes to complete.