school of | Computer Science & Statistics

# *Forgetful*: Eliminating Small, Transient Memory Allocations

**Student:** Dário Tavares Antunes

**Supervisor:** Dr. David M. Abrahamson

## Pattern

```
3  int main(void) {
⋮
8      size_t arrLen = (rand() % 10) + 1;
9
10     int* arr = malloc(arrLen);
⋮
17     free(arr);
18     return result;
19 }
```

Limited size

Limited lifetime

## Solution

```
3  int main(void) {
4      int stackArr[5];
⋮
9      size_t arrLen = (rand() % 10) + 1;
10
11     int* arr;
12     if (arrLen <= sizeof(stackArr)) {
13         arr = &stackArr[0];
14     } else {
15         arr = malloc(arrLen);
16     }
⋮
23     free(arr);
24     return result;
25 }
```

Determined by profiling

Conditional stack use

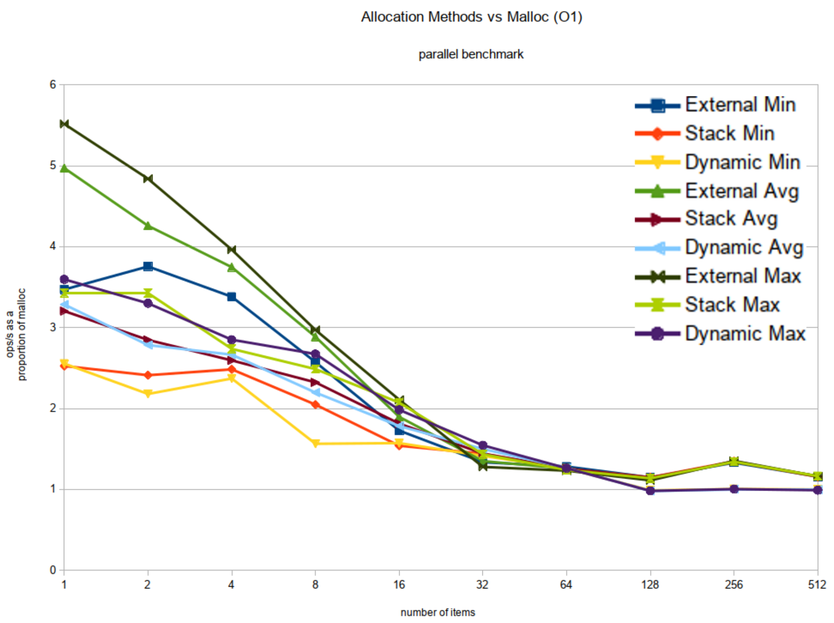
## Plugin Output

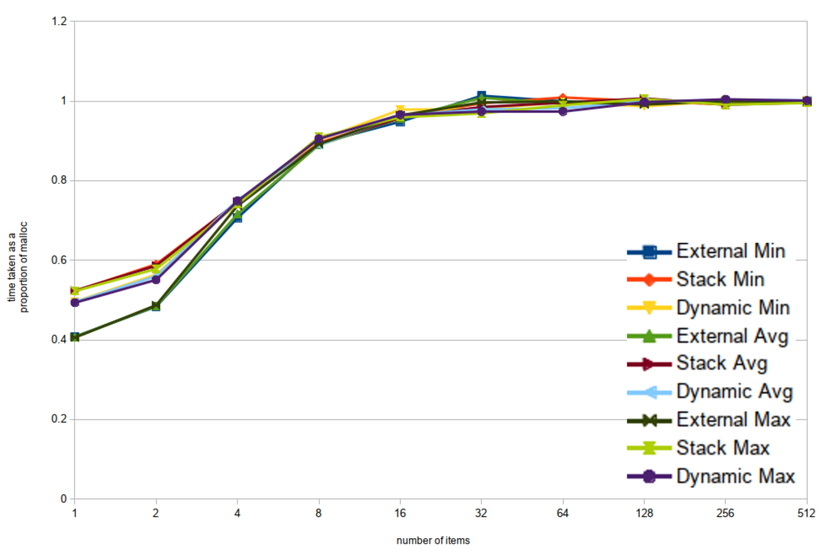
```
[value:final-states] Values at end of function main:
  __fc_random_counter ∈ [--..--]
  __fc_heap_status ∈ [--..--]
  arrLen ∈ [1..10]
  arr ∈ {{ NULL ; (int *)&__malloc_main_l10 }} or ESCAPINGADDR
  result ∈ {0}
[forgetful] Candidate for replacement in main: `free((void *)ar
r);` (test.c:17) frees base allocated at `
int *arr = malloc(arrLen);` (test.c:10)
```

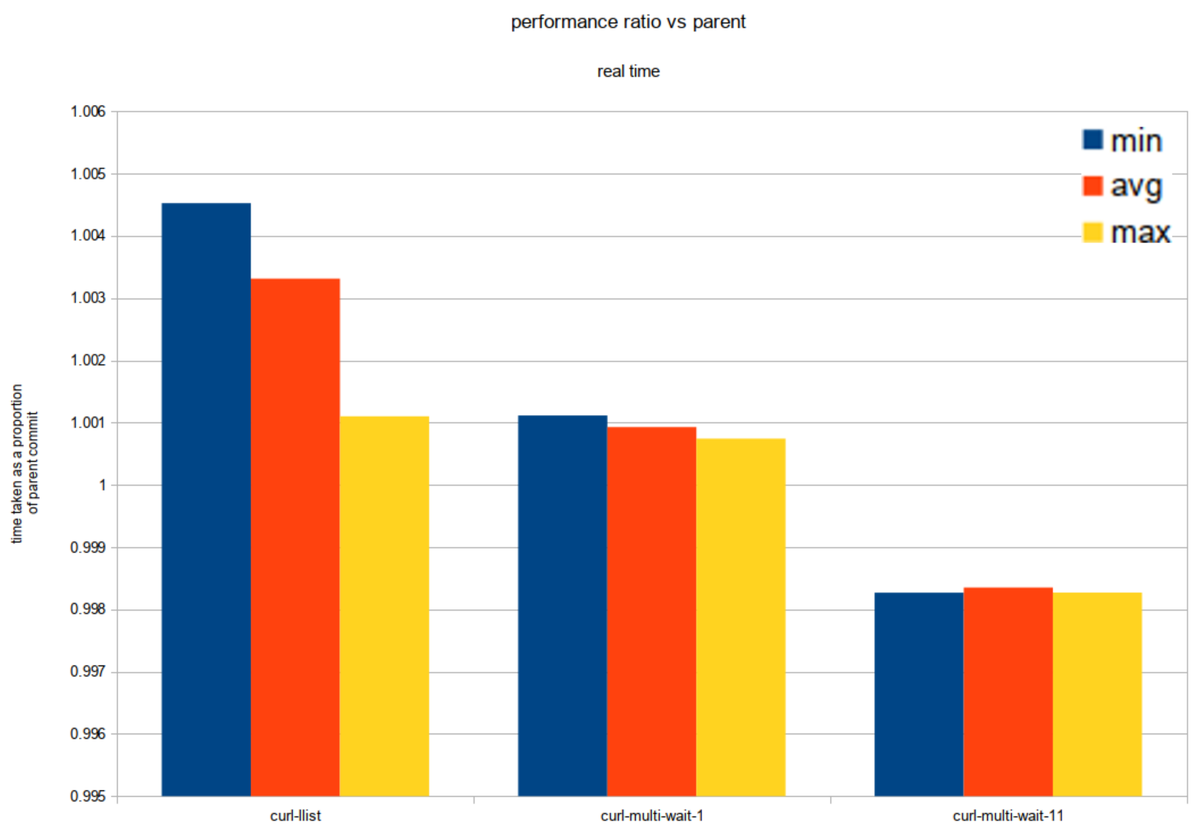
frama C

Software Analyzers

## Results




**B.A.(Mod.) Computer Science**

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin