Profiling call graph and Correction Algorithm

Jin-Woo Oh
Gwangju Institute of Science and Technology
(jinwoo0086@gist.ac.kr)

1. Introduction

- Software research environments normally include many large programs both for production use and for experimental investigation. The programs are typically modular, in accordance with generally accepted principles of good program design. Often they consist of numerous small routines that implement various abstractions. Sometimes such large programs are written by one programmer who has understood the requirements. But the program has had multiple authors and has evolved over time, changing the demands placed on the implementation of the abstractions without changing the implementation itself.

2. Goal

- Collects fairly accurate profiles
- Reduces collect profile time

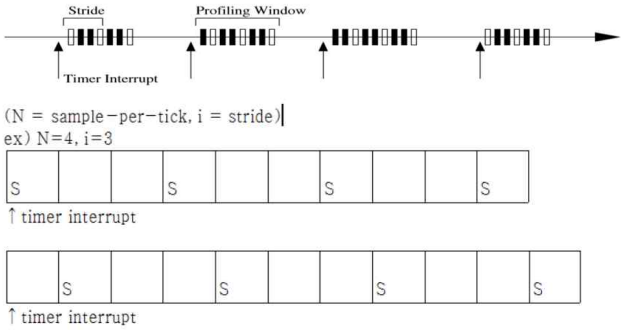
3. Background

- A call graph CG = (N,E)
- N is a set of nodes, and E is a set of edges. Each node in N represents a distinct method.
- E is triple (N1, CS, N2) that represents a call from call site CS in method N1 to method N2.
- A dynamic call graph(DCG) is a call graph that has frequencies associated with the call edges, and contains only those edges that are observed at runtime.

```
CallSite < BootstrapCL, Lorg/jikesrvm/compilers/opt/lir2mir/NormalBURS;, reachableChild, (Lorg/jikesrvm/ArchitectureSpecificOpt
$BURS_TreeNode;Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphNode;I)Z > 758 44 < BootstrapCL, Lorg/jikesrvm/compilers/opt/util/
SpaceEffGraphEdge;, getNextOut, ()Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphEdge; > 82 weight: 203194.0
CallSite < BootstrapCL, Lorg/jikesrvm/compilers/opt/lir2mir/NormalBURS;, reachableChild, (Lorg/jikesrvm/ArchitectureSpecificOpt
$BURS_TreeNode;Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphNode;I)Z > 758 34 < BootstrapCL, Lorg/jikesrvm/compilers/opt/lir2mir/NormalBURS;,
reachableRoot, (Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphNode;Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphNode;I)Z > 338 weight: 202830.0
CallSite < BootstrapCL, Lorg/jikesrvm/compilers/opt/lir2mir/NormalBURS;, reachableChild, (Lorg/jikesrvm/ArchitectureSpecificOpt
$BURS_TreeNode;Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphNode;I)Z > 758 26 < BootstrapCL, Lorg/jikesrvm/compilers/opt/util/
SpaceEffGraphEdge;, toNode, ()Lorg/jikesrvm/compilers/opt/util/SpaceEffGraphNode; > 82 weight: 202498.0
CallSite < BootstrapCL, Lorg/jikesrvm/classloader/UTF8Convert;, visitUTF8, ([BLorg/jikesrvm/classloader/UTF8Convert$UTF8CharacterVisitor;)V >
594 27 < BootstrapCL, Lorg/jikesrvm/classloader/UTF8Convert$StringHashCodeVisitor;, visit_char, (C)V > 110 weight: 162235.0
CallSite < BootstrapCL, Lorg/jikesrvm/classloader/UTF8Convert;, visitUTF8, ([BLorg/jikesrvm/classloader/UTF8Convert$UTF8CharacterVisitor;)V >
594 27 < BootstrapCL, Lorg/jikesrvm/classloader/UTF8Convert$ByteArrayStringEncoderVisitor;, visit_char, (C)V > 141 weight: 878.0
CallSite < BootstrapCL, Ljava/lang/VMSystem;, arraycopy, (Ljava/lang/Object;ILjava/lang/Object;II)V > 123 6 < BootstrapCL, Ljava/lang/
VMCommonLibrarySupport;, arraycopy, (Ljava/lang/Object;ILjava/lang/Object;II)V > 2365 weight: 123719.0
```

4. Sampling

- Combination of counter based and timer based sampling

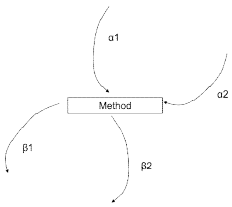


- Accuracy result

average sample per Timer Tick	Stride									
	1	3	7	15	31	63	127	255	511	1023
1	35.92	38.40	44.88	47.75	50.00	51.48	52.55	53.23	53.47	53.20
2	39.31	41.05	45.40	49.42	51.95	53.62	54.66	55.08	55.55	54.75
4	44.72	45.11	47.67	51.11	53.72	55.26	56.32	56.73	57.17	56.67
8	48.58	48.98	50.62	52.99	55.05	56.78	57.50	58.12	58.37	57.67
16	53.01	53.09	53.94	55.31	56.60	57.71	58.83	59.31	59.64	58.72
32	56.89	57.16	57.42	57.84	58.52	59.45	60.18	60.88	61.07	60.45
64	60.01	60.30	60.33	60.45	61.09	61.39	62.12	62.59	62.43	62.12
128	62.83	63.02	62.94	63.22	63.15	63.42	64.05	64.47	64.15	63.69

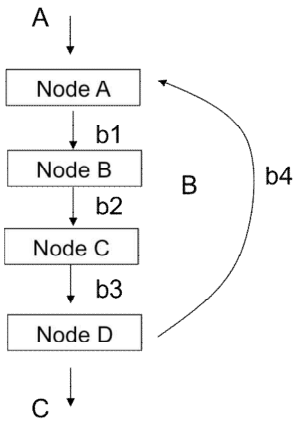
5. Correction Algorithm

- Correction1



$$\alpha 1 + \alpha 2 = \beta 1 + \beta 2$$

- Correction2



- Result