Cork: Dynamic Memory Leak Detection for Garbage-Collected Languages

Maria Jump and Kathryn S. McKinley
POPL ’07 January 17-19, 2007, Nice, France.

Cork: Dynamic Memory Leak Detection for Java

Maria Jump and Kathryn S. McKinley
Technical Report TR-06-07 January 2006

概要

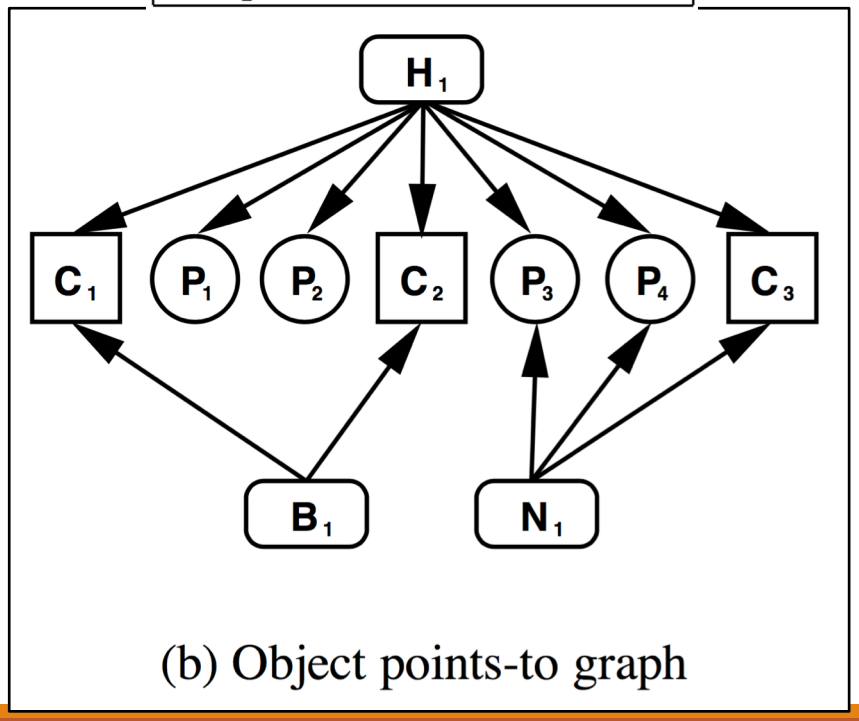
- オブジェクトの型単位でグループ化し、GC毎にサイズの成長を計測
- 成長率を集約し、特定の閾値を超えた型をリークとして報告
- TPFGから原因箇所の特定に役立つ情報をレポート化
  - TPFG: Type Points from Graph
- 低い空間オーバーヘッド(Ave:+0.145%, Max:+0.5%)を達成
- 低い実行オーバーヘッド(Total:+1.9%~+4.0%)を達成
- 致命的なリークを特定する精度を達成

3行モチベ

- 1つ1つのオブジェクトの判定なんかやっていられるか！
- 致命的なメモリリークは特定の型のオブジェクトであふれる
  - 致命的＝メモリが枯渇してクラッシュする、極端に性能が落ちる
- 増加していくオブジェクト型に注目すればいいんだ！

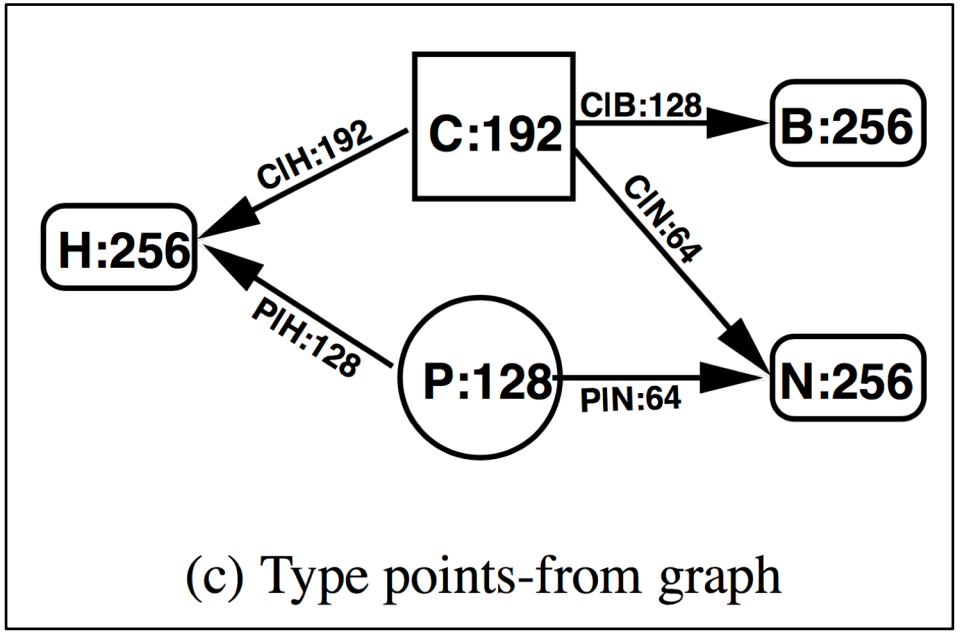
提案手法
TPFG(Type points-from graph)

Type	Symbol	Size
HashTable	H	256
Queue	N	256
Queue	B	256
Company	C	64
People	P	32



(b) Object points-to graph

このグラフに残っている型がリーク候補



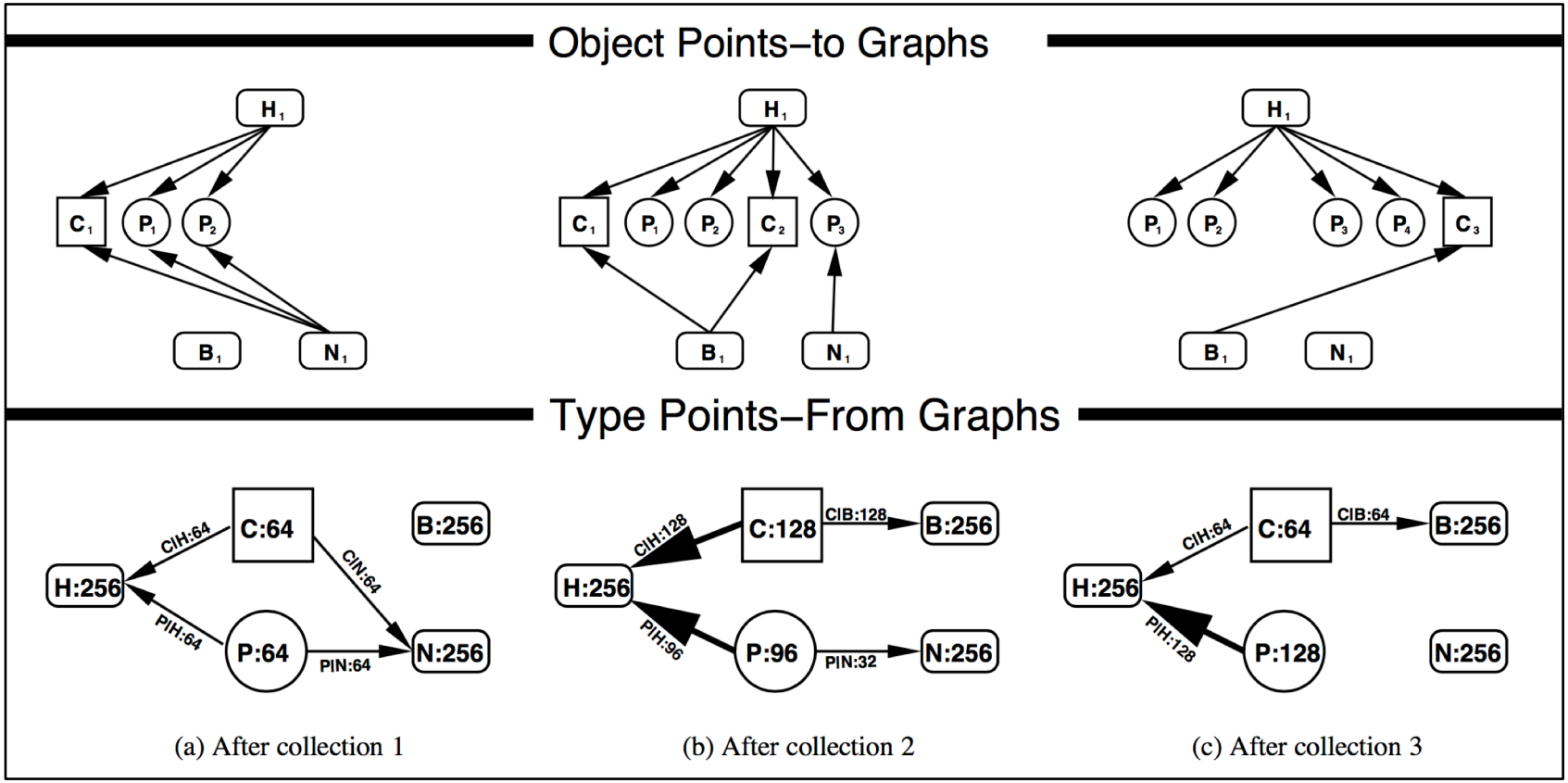
(c) Type points-from graph

提案手法
RRT（Ratio Ranking Technique）

連続するGC間でオブジェクト型の成長を計測

- V : ある型のサイズ合計(Volume)
- Q : 成長率 $Q_{Ti} = \frac{V_{Ti}}{V_{Ti-1}}$
  - i : i回目のGC
- p : 経過GC数
- g : 評価成長率 $g_{Ti} = p_{Ti} * (Q_{Ti} - 1)$
- r : 成長レート $r_{Ti} = r_{Ti-1} + g_{Ti}$
- 評価式1 : $V_{Ti} > (1 - f)V_{Ti-1}$ → 満たせばTPFGに型を残す
  - f : 減衰率
- 評価式2 : $r_{Ti} > R_{thres}$ → 満たせばリーク判定
  - R_thres : リーク判定の閾値

提案手法
RRT（Ratio Ranking Technique）



(a) After collection 1 (b) After collection 2 (c) After collection 3

提案手法

	V_{T1}	Q_{T1}	p_{T1}	g_{T1}	r_{T1}	state1	state2
H	256	1	0	0	0	T	F
N	256	1	0	0	0	T	F
B	256	1	0	0	0	T	F
C	64	1	0	0	0	T	F
P	64	1	0	0	0	T	F

Type	Symbol	Size
HashTable	H	256
Queue	N	256
Queue	B	256
Company	C	64
People	P	32

	V_{T2}	Q_{T2}	p_{T2}	g_{T2}	r_{T2}	state1	state2
H	256	1	1	0	0	T	F
N	256	1	1	0	0	T	F
B	256	1	1	0	0	T	F
C	128	2	1	1	1	T	F
P	96	1.5	1	0.5	0.5	T	F

	V_{T3}	Q_{T3}	p_{T3}	g_{T3}	r_{T3}	state1	state2
H	256	1	2	0	0	T	F
N	256	1	2	0	0	T	F
B	256	1	2	0	0	T	F
C	64	0.5	2	-1	0	T	F
P	128	1.33	2	0.66	1.16	T	F

実験結果(省略)

	(b) Type Points-From Statistics							(c) Space Overhead				
	# of types		# edges per type		# edges per TPFG		% pru-	TIB		TIB+Cork		
Benchmark	avg	max	avg	max	avg	max	ned	MB	%H	MB	%H	Diff
Eclipse	667	775	2	203	4090	7585	42.2	0.53	0.011	0.70	0.015	0.167
fop	423	435	3	406	1559	2623	45.2	0.36	0.160	0.55	0.655	0.495
pmd	360	415	3	121	967	1297	66.0	0.30	0.031	0.44	0.186	0.155
ps	314	317	2	93	813	824	66.3	0.28	0.029	0.39	0.082	0.053
javac	347	378	3	99	1118	2126	45.8	0.28	0.071	0.43	0.222	0.151
jython	351	368	2	114	928	940	66.2	0.28	0.041	0.39	0.112	0.071
jess	318	319	2	89	844	861	66.0	0.27	0.049	0.38	0.143	0.094
antlr	320	356	2	123	860	1398	55.8	0.27	0.016	0.39	0.282	0.266
bloat	345	347	2	101	892	1329	50.6	0.26	0.017	0.38	0.064	0.047
jbb2000	318	319	2	110	904	1122	59.0	0.26	**	0.38	**	**
jack	309	318	2	107	838	878	66.2	0.26	0.042	0.37	0.131	0.089
mtrt	307	307	2	91	820	1047	57.5	0.26	0.081	0.37	0.258	0.177
raytrace	305	306	2	91	814	1074	56.1	0.26	0.085	0.37	0.272	0.187
compress	286	288	2	89	763	898	60.9	0.25	0.105	0.36	0.336	0.231
db	289	289	2	91	773	787	66.1	0.25	0.160	0.35	0.467	0.307
Geomean	342	357	2	116	1000	1303	57.4	0.29	0.048	0.41	0.168	0.145

レポート例[TR'06]

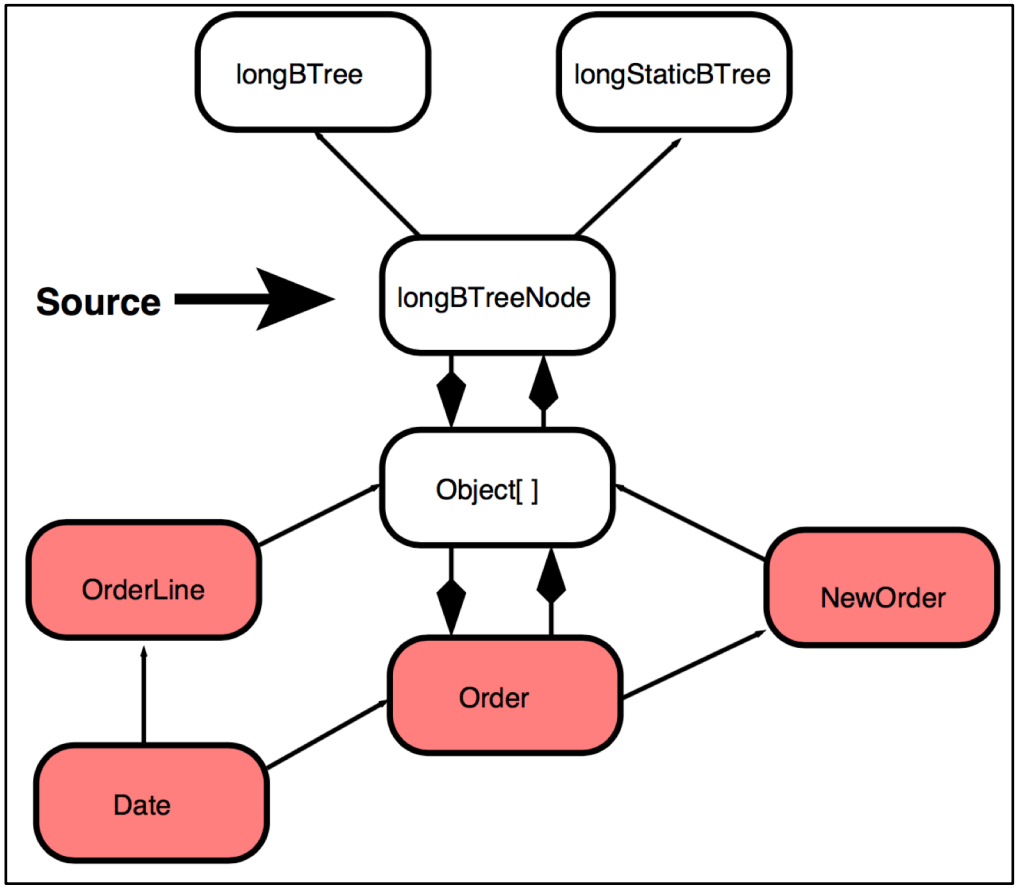
GC No.	Rank	Type
15	2104.17	Ljava/util/ArrayList;
	135.26	Lorg/apache/fop/fo/LengthProperty;
	115.64	Lorg/apache/fop/datatypes/AutoLength;
18	2106.28	Ljava/util/ArrayList;
21	2108.24	Ljava/util/ArrayList;

Method	bcidx
Ljava/util/ArrayList;	
too numerous to be useful (45)	
Lorg/apache/fop/layout/inline/WordArea;	
.../render/pdf/PDFRenderer;.renderWordArea...	355
.../render/pdf/PDFRenderer;.renderWordArea...	450
.../render/pdf/PDFRenderer;.renderWordArea...	555
.../render/pdf/PDFRenderer;.renderWordArea...	811
.../render/pdf/PDFRenderer;.renderWordArea...	820
Lorg/apache/fop/layout/LineArea;	
.../layout/BlockArea;.getCurrentLineArea...	26
.../layout/BlockArea;.createNextLineArea...	45

←リーク候補
↙アロケーションサイト
↓関連ポインタ

Type
Ljava/util/ArrayList;
← Lorg/apache/fop/layout/inline/WordArea;
← Ljava/lang/Object; []
← Ljava/util/ArrayList;
← Lorg/apache/fop/layout/LineArea;
← Ljava/lang/Object; []
← Ljava/util/ArrayList;
← Lorg/apache/fop/layout/inline/WordArea;
← Ljava/lang/Object; []
← Ljava/util/ArrayList;
← Lorg/apache/fop/layout/inline/InlineSpace;
← Ljava/lang/Object; []
← Ljava/util/ArrayList;
...

レポート例[POPL'07]



関連ポインタをグラフ化

小泉的まとめ

・グループ化してその成長を計測するのはグッドアイデア

・ただ、本論文の手法では危険性の低いリークは見逃している

・また、リーク判定に直接関わる閾値R_thresなどは集計の頻度(本論文内のGC頻度)に大きく左右される