August 2014

Framework Evaluation - ASM vs. BCEL in static analysis setting

For my experiment, I used 27 open-source Java applications of different sizes. As shown in the table below. The size of each application is mentioned in terms of types and bytecode instructions including the accompanied external libraries*. I intentionally left the analysers to parse external libraries to run on the largest possible number of bytecode instructions. In the table, the number of all visited methods and the count of interface method invocations are declared. In the last column, the speed ratio of ASM vs BCEL based on the time cost of both analysers on each run is computed. Finally, I concluded that for my static analysis, ASM is in average 3.65 times faster than BCEL. All of my static analyses were done on an Intel Core i7-3630QM machine with 16 GB of memory running x64 Windows 7 Professional. The time cost logged for each experiment run does not include the time spent on file stream reading (i.e., reading the benchmark program from disk, extract the .class files in each .jar or .zip file and provide the parser a raw java.io.InputStream).

	Benchmark	Analysed Types	Analysed Bytecode instructions (ASM count)	all visited methods	Interface method invocations Count	Static analysis time (ms)		ASM Speed vs BCEL ratio
						ASM	BCEL	
1	perf4j-0.9.16	87	9,648	707	278	32	281	778.13%
2	Commons-io-2.4	110	20,716	1,169	173	47	374	695.74%
3	cglib-2.2.2	228	22,957	1,288	615	63	421	568.25%
4	Slf4j-1.7.5	358	34,346	2,534	737	95	581	511.58%
5	commons-lang-3-3.1	153	49,844	2,358	591	78	515	560.26%
6	sablecc-3.2	286	61,596	2,277	864	105	561	434.29%
7	bcel-5.2	383	61,617	2,911	1,208	156	702	350.00%
8	fidocadj	173	61,904	1,162	177	80	530	562.50%
9	Jmock-2.6.0	685	99,066	4,411	1,471	188	797	323.94%
10	javacc-5.0	154	116,889	2,149	1,073	93	531	470.97%
11	Proguard-4.11	592	122,898	5,643	1,389	173	861	397.69%
12	Hamcrest-1.3	1,532	263,294	10,327	7,562	313	1,343	329.07%
13	findbugs-1.3.9	1,735	286,709	11,233	7,411	375	1,454	287.73%
14	pmd-5.1.0	1,288	294,051	9,536	5,921	453	1,327	192.94%
15	Saxon-9.1.0.8	1,141	353,997	10,624	5,674	376	1,472	291.49%
16	swt	897	398,503	10,453	335	297	1,331	348.15%
17	truezip	3,188	587,575	20,971	6,157	578	1,911	230.62%
18	tomcat-8	3,217	954,463	30,434	18,873	954	3,168	232.08%
19	Groovy-all-1.8.6	3,715	997,227	41,266	23,022	937	3,222	243.86%
20	fanchallenge-1.3.3	5,595	1,088,556	39,713	13,153	1,046	2,872	174.57%
21	Aspectj-1.7.4	5,376	1,764,277	56,219	35,936	1,093	5,242	379.60%
22	jdownloader.zip	8,221	2,151,198	65,603	24,476	1,476	5,162	249.73%
23	Hibernate-4.3.4	21,204	3,005,295	159,479	165,895	3,058	10,500	243.36%
24	Sonarqube-4.1.2	43,957	8,786,271	322,524	169,210	5,866	18,893	222.08%
25	jdk1.7.0_10	44,665	9,618,575	355,034	137,111	6,023	23,276	286.45%
26	jboss-as-7.1.1.final	60,741	10,698,548	457,764	278,613	7,910	31,123	293.46%
27	eclipse_rcp_kepler_sr2	80,856	13,770,171	568,554	443,147	11,545	34,103	195.39%
	Avg							3.65

*To save space I did not include a description on each application.