Escape Analysis Opportunities in C2

Roberto Castañeda Lozano, Daniel Lundén, Saranya Natarajan, Antón Seoane Ampudia, Daniel Skantz, Jesper Wilhelmsson

2025-10-20

Current State of Escape Analysis in C2

- C2 implements Escape Analysis (EA) and Scalar Replacement (SR):
  - EA follows the **flow-insensitive** algorithm from Choi [1].
  - SR happens “naturally” in the IR after EA is carried out.
    - Some IR preparation at the end of EA.

 Copyright © 2025, Oracle and/or its affiliates

Current State of Escape Analysis in C2

- C2 implements Escape Analysis (EA) and Scalar Replacement (SR):
  - EA follows the **flow-insensitive** algorithm from Choi [1].
  - SR happens “naturally” in the IR after EA is carried out.
    - Some IR preparation at the end of EA.
- Additionally, current EA in C2 is **not partial** (i.e., does not follow an approach such as Stadler’s [2]).

 Copyright © 2025, Oracle and/or its affiliates

Current State of Escape Analysis in C2

- C2 implements Escape Analysis (EA) and Scalar Replacement (SR):
  - EA follows the **flow-insensitive** algorithm from Choi [1].
  - SR happens “naturally” in the IR after EA is carried out.
    - Some IR preparation at the end of EA.
- Additionally, current EA in C2 is **not partial** (i.e., does not follow an approach such as Stadler’s [2]).

[1] Jong-Deok Choi et al. “Escape Analysis for Java”. In: Denver, CO, USA: ACM, 1999, pp. 1–19. DOI: `10.1145/320384.320386`.

[2] Lukas Stadler, Thomas Würthinger, and Hanspeter Mössenböck. “Partial Escape Analysis and Scalar Replacement for Java”. In: Grenoble, France: Springer, 2014, pp. 68–87. DOI: `10.1007/978-3-642-54807-9_4`.

 Copyright © 2025, Oracle and/or its affiliates

Objectives

- Expand Escape Analysis in C2 so we can benefit from a more refined analysis:
  - Flow sensitivity.
  - Partiality.

 Copyright © 2025, Oracle and/or its affiliates

Objectives

- Expand Escape Analysis in C2 so we can benefit from a more refined analysis:
  - Flow sensitivity.
  - Partiality.
- Analyze the current limitations to SR in C2 and address them.
  - Cesar Soares' work attempted something similar before, introducing improvements across control merges.

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis

- Empirical evaluation to analyze current challenges to EA/SR in C2.

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis

- Empirical evaluation to analyze current challenges to EA/SR in C2.
- Manual assessment of why C2 has failed to scalar replace each of the "hottest" allocation sites through a series of benchmarks.
  1. *(Throughout all benchmarks).* Instrumented JFR to obtain high-level allocation statistics.
  2. *(Per-benchmark).* Determining "hot" sites and inspecting the Java code.

Results shown from SPECjvm, SPECjbb, and DaCapo in the following slides are for internal reference only. Benchmarks were not executed in the official publication mode permitted for public disclosure of results.

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis: DaCapo

benchmark	allocations	allocated (MB)	duration (s)	allocs/s	alloc. rate (MB/s)	array alloc. (MB)	inst. alloc.
avrora	594345	3433	181	3266	19	847	75%
batik	39090	87543	517	76	169	51150	41%
biojava	280770	563348	415	676	1357	258608	54%
eclipse	318395	81467	209	1520	389	58947	27%
fop	18669	20141	36	506	546	9240	54%
graphchi	284074	309652	542	523	570	166623	46%
h2	188135	1248875	575	327	2169	435281	65%
jme	612	188	408	1	0	127	32%
jython	172905	347558	364	475	955	166616	52%
kafka	152585	102365	264	577	387	55713	45%
pmd	234443	181262	77	3025	2339	25660	85%
spring	1484482	1155359	181	8179	6365	944760	18%
sunflow	423821	845603	185	2279	4546	20126	97%
tomcat	204435	153115	220	926	693	128595	16%
xalan	515680	443860	56	9169	7892	296905	33%
zxing	128185	31140	25	4993	1213	30196	3%

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis: SPECjbb

benchmark	allocations	allocated (MB)	duration (s)	allocs/s	alloc. rate (MB/s)	array alloc. (MB)	inst. alloc.
specjbb2005	55262	80994	41	1335	1957	41229	49%
specjbb2015	34861	167527	94	370	1780	95993	42%

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis: SPECjvm

benchmark	allocations	allocated (MB)	duration (s)	allocs/s	alloc. rate (MB/s)	array alloc. (MB)	inst. alloc.
Compress	21935	11741	55	396	212	11324	3%
Crypto.aes	101577	74189	55	1835	1340	74169	0%
Crypto.rsa	506419	108598	55	9179	1968	84926	21%
Crypto.signverify	294854	295509	55	5328	5340	291946	1%
Derby	276465	499347	55	5005	9041	113179	77%
FFT.large	532	13123	55	10	238	13119	0%
FFT.small	185810	90812	55	3363	1644	90766	0%
LU.large	38371	6912	55	695	125	6909	0%
LU.small	359696	236328	55	6518	4283	236233	0%
MonteCarlo	3676	141	55	67	3	128	9%
MPEG	330165	37707	55	5979	683	37670	0%
Serial	357461	478024	55	6477	8662	281037	41%
SOR.large	16451	543	54	302	10	525	3%
SOR.small	3512	80	55	64	1	75	6%
Sparse.large	945	11118	55	17	202	11014	0%
Sparse.small	55798	19102	55	1008	345	18857	1%
XML.transform	311604	341589	55	5641	6184	212748	37%
XML.validation	202152	348890	55	3666	6327	156078	55%

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis: Summary

- Through a simple analysis we can see several benchmarks presenting high allocation numbers.
- Narrowing those numbers down to the highest allocation pressure sites, we can understand why we cannot scalar replace those objects.
- Common reasons:
  - “True” escaping allocations (nothing to do).
  - Insufficient inlining.
  - Suboptimal merge handling.
  - Limitations of EA algorithm.

 Copyright © 2025, Oracle and/or its affiliates

Measuring the Impact of Escape Analysis: Summary

- Through a simple analysis we can see several benchmarks presenting high allocation numbers.
- Narrowing those numbers down to the highest allocation pressure sites, we can understand why we cannot scalar replace those objects.
- Common reasons:
  - “True” escaping allocations (nothing to do).
  - **Insufficient inlining.**
  - **Suboptimal merge handling.**
  - **Limitations of EA algorithm.**

 Copyright © 2025, Oracle and/or its affiliates

A Significant Example: specjbb2005

- `BigDecimal.java:1321` (`BigDecimal::valueOf(long, int)`).

 Copyright © 2025, Oracle and/or its affiliates

A Significant Example: specjbb2005

- `BigDecimal.java:1321` (`BigDecimal::valueOf(long, int)`).
- Multiple occurrences within the compilation unit:
  - No escape: 4.
  - Arg escape: 1.
  - Global escape: 2.

 Copyright © 2025, Oracle and/or its affiliates

A Significant Example: specjbb2005

- `BigDecimal.java:1321` (`BigDecimal::valueOf(long, int)`).
- Multiple occurrences within the compilation unit:
  - No escape: 4.
  - Arg escape: 1.
  - Global escape: 2.
- No scalar replacement possible.

Copyright © 2025, Oracle and/or its affiliates

A Significant Example: specjbb2015

- The “no escape” allocation sites cannot be scalar replaced due to control merge limitations. Example:

```
private static BigDecimal add(long xs, long ys, int scale) {
  long sum = add(xs, ys);
  if (sum != INFLATED)
    return BigDecimal.valueOf(sum, scale);
  return new BigDecimal(BigInteger.valueOf(xs).add(ys), scale);
}
```

  - Even if all methods with `BigDecimal` are inlined, C2 cannot scalar replace through such allocation merges.

 Copyright © 2025, Oracle and/or its affiliates

A Significant Example: specjbb2015

- The “no escape” allocation sites cannot be scalar replaced due to control merge limitations. Example:

```
private static BigDecimal add(long xs, long ys, int scale) {
  long sum = add(xs, ys);
  if (sum != INFLATED)
    return BigDecimal.valueOf(sum, scale);
  return new BigDecimal(BigInteger.valueOf(xs).add(ys), scale);
}
```

  - Even if all methods with `BigDecimal` are inlined, C2 cannot scalar replace through such allocation merges.
- The “arg escape” allocation escapes to a non-inlined method.
  - Forcing inlining leads to the allocation being detected as “no escape” but not scalar replaceable due to merges.

 Copyright © 2025, Oracle and/or its affiliates

Circling Back: Current Obstacles

- C2 can benefit from a more complete escape analysis.

 Copyright © 2025, Oracle and/or its affiliates

Circling Back: Current Obstacles

- C2 can benefit from a more complete escape analysis.
- However, there are two significant current limitations that reduce the potential of an improved escape analysis:
  - Insufficient inlining.
  - A suboptimal scalar replacement algorithm.

 Copyright © 2025, Oracle and/or its affiliates

Circling Back: Current Obstacles

- C2 can benefit from a more complete escape analysis.
- However, there are two significant current limitations that reduce the potential of an improved escape analysis:
  - Insufficient inlining.
  - A suboptimal scalar replacement algorithm.
- Addressing (individually or together) these limitations is what will reduce allocations the most.

 Copyright © 2025, Oracle and/or its affiliates

Circling Back: Summary

- There are many allocation-heavy examples that can benefit from improvements in EA and SR.

 Copyright © 2025, Oracle and/or its affiliates

Circling Back: Summary

- There are many allocation-heavy examples that can benefit from improvements in EA and SR.
- Improving Escape Analysis:
  - Roberto will next present a “frequency-aware” escape analysis algorithm.

Copyright © 2025, Oracle and/or its affiliates

Circling Back: Summary

- There are many allocation-heavy examples that can benefit from improvements in EA and SR.
- Improving Escape Analysis:
  - Roberto will next present a “frequency-aware” escape analysis algorithm.
- Improvements in EA must come with more effective scalar replacement.

 Copyright © 2025, Oracle and/or its affiliates