Functionality ready to land in optimizer and backend

Miguel Garcia http://lamp.epfl.ch/~magarcia LAMP.EPFL

2012-05-01

Outline

Status as of M3

Compilation speed test.stability under -optimize Not yet merged: Faster reaching-defs for ClosureElim Not yet merged: GenASM

<ロト</th>
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●
●

Ongoing work

Single-pass Type-flow analysis Open performance riddles around inlining

Next steps

- Status as of M3

Compilation speed

Stats compiling the compiler (excerpt, -Dscala.timings=true)								
phase	id	sec	share(%)					
inliner	22	54sec	30%					
typer	4	38sec	21%					
jvm	26	15sec	88					
erasure	13	14sec	88					
dce	25	12sec	7%					
icode	21	8sec	48					
closelim	24	6sec	3%					
uncurry	9	5sec	2%					
specialize	11	3sec	2%					
refchecks	8	3sec	2%					
mixin	19	3sec	18					
inlineExceptionHandlers	23	2sec	1%					

-Status as of M3

test.stability under -optimize

- Observed behavior, with makePublic() enabled :
 - test.stability fails under -optimize.
 - However, for *identical* compiler runs (but they really have to be *identical*) optimized output is stable (caveat: tested outside build.xml).
- Explanation:
 - During inliner, there's no topological sorting (over the call-graph relationship) of the methods being visited.
 - Different visit orders (across compiler runs) lead to different method bodies, with ripple effects.

Recent developments

- makePublic() was disabled the night before Scala Days.
- just found that postponing makePublic() until all other inlining conditions succeed, recovers test.stability.
- Another idea: sort IMethods by sym.id.

Details at: https://groups.google.com/d/topic/scala-internals/yGmQkBn9Gmk/discussion

24916

Status as of M3

test.stability under -optimize (2 of 2)

The only non-deterministic behavior that logs show is:

A (Base): iscar	d\stab\salc	ompbyq	uick.txt	B: temp\discar	d\s	tab\salc	ompbyst	rap.txt	C: \temp\disca	rd\stab\sal	compbyt	third.txt
Top line 15088	1æncoding	: UTF-{L	ine end style: L	Top line 15088	18	incoding	: UTF-EL	ine end style: L	Top line 15088	1æncoding	: UTF-EL	ine end style: L
1508818	[log	jvm]	(Lscala/H	1508818		[log	jvm]	(Lscala/I	1508818	[log	jvm]	(Lscala/B
1508819	[log	jvm]	(Lscala/H	1508819		[log	jvm]	(Lscala/I	1508819	[log	jvm]	(Lscala/E
1508820	[log	jvm]	(Lscala/r	1508820		[log	j∨m]	(Lscala/	1508820	[log	jvm]	(Lscala/r
				1508821		[log	jvm]	()Lscala,				
				1508822		[log	jvm]	()Lscala,				
1508821	[log	jvm]	No mirror	1508823		[log	jvm]	No mirror	1508821	[log	jvm]	No mirror
1508822	[log	jvm]	Lscala/co	1508824		[log	jvm]	Lscala/co	1508822	[log	jvm]	Lscala/co
1508823	[log	jvm]	Lscala/cc	1508825		[log	jvm]	Lscala/co	1508823	[log	jvm]	Lscala/co
1508824	[log	jvm]	()Lscala/	1508826		[log	jvm]	()Lscala,	1508824	[log	jvm]	()Lscala/
1508825	[log	jvm]	()Lscala/	1508827		[log	jvm]	()Lscala,	1508825	[log	jvm]	()Lscala/
1508826	[log	jvm]	()Lscala/	_					1508826	[log	jvm]	()Lscala/
1508827	[log	jvm]	()Lscala/	_					1508827	[log	jvm]	()Lscala/
1508828	[log	jvm]	Dumping n	1508828		[log	jvm]	Dumping r	1508828	[log	jvm]	Dumping n
1508829	[log	jvm]	Adding st	1508829		[log	jvm]	Adding st	1508829	[log	jvm]	Adding st
1												
A (Base): iscar	ı d\stab\salcı	ompbyq	uick.txt	B: temp\discar	¦ d∖s	tab\salo	ompbyst	rap.txt	C: (temp\disca	ı ard\stab\sal	compby	third.txt
A (Base): iscan Top line 150936	i d\stab\salci 59Encoding:	ompbyq : UTF-{L	uick.txt ine end style: L	B: <mark>temp\discare</mark> Top line 15093	d\s 69	tab\salo	ompbyst : UTF-{L	rap.txt ine end style: U	C: \temp\disca Top line 15093	i ard\stab\sal 369Encoding	compby : UTF-{I	third.txt
A (Base): iscard Top line 150936 1509369	i d\stab\salco 59Encoding: [log	ompbyq :UTF-&L jvm]	uick.txt ine end style: l (Lscala/r	B: temp\discard Top line 15093 1509369	d\s 69E	tab\salo incoding [log	ompbyst :UTF-&L jvm.]	rap.txt ine end style: U (Lscala/1	C: (temp\disca Top line 1509: 1509369	ard\stab\sal 369Encoding [log	compby : UTF-{I j vm]	third.txt ine end style: l (Lscala/r
A (Base): iscard Top line 150936 1509369 1509370	d\stab\salco 59Encoding: [log [log	ompbyq UTF-&L jvm] jvm]	uick.txt ine end style: l (Lscala/r (Lscala/c	B: temp\discard Top line 15093 1509369 1509370	d\s 69≘	tab\salc incoding [log [log	ompbyst :UTF-&L jvm] jvm]	rap.txt ine end style: U (Lscala/s (Lscala/s	C: (temp\disca Top line 15093 1509369 1509370	ard\stab\sal 369Encoding [log [log	compby :UTF-{i jvm] jvm]	third.txt ine end style: l (Lscala/r (Lscala/c
A (Base): iscard Top line 150936 1509369 1509370 1509371	d\stab\salco 69Encoding: [log [log [log	ompbyq :UTF-{L jvm] jvm] jvm]	uick.txt ine end style: l (Lscala/r (Lscala/c (Lscala/r	B: temp\discard Top line 15093 1509369 1509370 1509371	d\s 69:	tab\salo incoding [log [log [log	ompbyst UTF-{L jvm] jvm] jvm]	rap.txt ine end style: U (Lscala/s (Lscala/s (Lscala/s	C: (temp\disca Top line 15093 1509369 1509370 1509371	ard\stab\sal 369Encoding [log [log [log	compby :UTF-& jvm] jvm] jvm]	third.txt ine end style: l (Lscala/r (Lscala/c (Lscala/r
A (Base): iscard Top line 150930 1509369 1509370 1509371 1509372	d\stab\salca SEncoding: [log [log [log [log	ompbyq UTF-&L jvm] jvm] jvm] jvm]	uick.txt ine end style: ((Lscala/r (Lscala/c (Lscala/r No mirror	8: temp\discard Top line 15093 1509369 1509370 1509371 1509372	d\s 69	tab\salo ncoding [log [log [log [log	ompbyst UTF-{L jvm] jvm] jvm] jvm]	rap.txt ine end style: U (Lscala/s (Lscala/s (Lscala/s No mirros	C: temp\disca Top line 1509: 1509369 1509370 1509371	ard\stab\sal 369Encoding [log [log [log	compby :UTF-8 jvm] jvm] jvm]	third.txt ine end style: l (Lscala/r (Lscala/c (Lscala/r
A (Base): iscard Top line 150936 1509369 1509370 1509371 1509372 1509373	d\stab\salci 69Encoding: [log [log [log [log [log	ompbyq ;UTF-&L jvm] jvm] jvm] jvm] jvm]	uick.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r No mirror (Ljava/la	B: temp\discard Top line 150936 1509369 1509370 1509371 1509372 1509373	d\s 69	tab\salo incoding [log [log [log [log [log	ompbyst UTF-&L jvm] jvm] jvm] jvm] jvm]	rap.bxt ine end style: U (Lscala/s (Lscala/s (Lscala/s No mirros (Ljava/la	C: \temp\disca Top line 1509: 1509369 1509370 1509371	ard\stab\sal 36Œncoding [log [log [log	compby :UTF-81 jvm] jvm] jvm]	third.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r
A (Base): iscard Top line 150936 1509369 1509370 1509371 1509372 1509373 1509374	d\stab\salca SEncoding: [log [log [log [log [log [log	ompbyq jvm] jvm] jvm] jvm] jvm] jvm]	uick.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r No mirror (Ljava/la Adding st	B: temp\discard Top line 150936 1509370 1509371 1509372 1509373 1509374	d\s 69E	tab\salo ncoding [log [log [log [log [log	jvm] jvm] jvm] jvm] jvm] jvm] jvm]	rap.txt ine end style: U (Lscala/s (Lscala/s (Lscala/s No mirros (Ljava/la Adding st	C: \temp\disca Top line 1509: 1509369 1509370 1509371	ard\stab\sal 36Encoding [log [log [log [log	jvm] jvm] jvm] jvm]	third.txt ine end style: L (Lscala/r (Lscala/r (Lscala/r Adding st
A (Base): iscar Top line 150936 1509369 1509370 1509371 1509373 1509374 1509375	d \stab \salca 69Encoding: [log [log [log [log [log [log [log	ompbyq jvm] jvm] jvm] jvm] jvm] jvm] jvm]	uick.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r No mirror (Ljava/la Adding st Adding st	B: temp\discars Top line 15093 1509369 1509370 1509371 1509372 1509373 1509374 1509375	d <i>\s</i> 69E	tab\salo incoding [log [log [log [log [log [log [log	ompbyst jvm] jvm] jvm] jvm] jvm] jvm] jvm]	rap.txt ine end style: U (Lscala/c (Lscala/c (Lscala/c No mirror (Ljava/la Adding st Adding st	C: (temp/disca Top line 1509: 1509369 1509370 1509371 1509372 1509373	ard\stab\sal 36@ncoding [log [log [log [log [log	jvm] jvm] jvm] jvm]	third.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r Adding st Adding st
A (Base): iscan Top ine 15093(1509369 1509370 1509371 1509373 1509373 1509374 1509375	d\stab\salce 69Encoding: [log [log [log [log [log [log [log [log	ompbyq jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	uidk.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r No mirror (Ljava/la Adding st (Ljava/la	B: temp\discan Top ine 15093 1509369 1509370 1509371 1509373 1509373 1509374 1509375	d /s	tab\salo incoding [log [log [log [log [log [log [log [lo	ompbyst jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	nap.txt ine end style: U (Lscala/s (Lscala/s No mirror (Ljava/la Adding st (Ljava/la	C: temp\disca Top line 15093 1509369 1509370 1509371 1509372 1509373 1509374	rd\stab\sal 36Encoding [log [log [log [log [log [log	i UTF-{I jvm] jvm] jvm] jvm] jvm] jvm]	hird.txt ine end style: L (Lscala/r (Lscala/c (Lscala/r Adding st Adding st (Ljava/la
A (Base): iscan Top line 150936 1509370 1509370 1509372 1509373 1509373 1509375 1509376 1509377	d\stab\salce 69Encoding: [log [log [log [log [log [log [log [log	ompbyq jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	uidk.btt ine end style:L (Lscala/r (Lscala/r (Lscala/c (Lscala/r No mirror (Ljava/la Adding st (Ljava/la Adding st	B: temp\discarr Top line 15093 1509369 1509370 1509371 1509372 1509373 1509374 1509375 1509376	d\s 69	tab\salo ncoding [log [log [log [log [log [log [log [lo	jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	rap.bt ine end style: U (Lscala/) (Lscala/) (Lscala/) (Lscala/) (Ljava/) Adding st Adding st Adding st	C: temp\disca Top line 1509; 1509369 1509370 1509371 1509372 1509373 1509374 1509375	rd\stab\sal 36Encoding [log [log [log [log [log [log [log	jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	third.txt ine end style: L (Lscala/z (Lscala/d (Lscala/d (Lscala/d Adding st (Ljava/la Adding st
A (Base): iscan Top line 15093/ 1509369 1509370 1509371 1509373 1509374 1509375 1509377 1509377	d\stab\salce SEncoding: [log [log [log [log [log [log [log [log	ompbyq jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	uidk.bt ine end style:l (Lscala/r (Lscala/c (Lscala/r No mirror (Ljava/la Adding st Adding st Adding st	B: temp\discarr top line: 15093 1509369 1509370 1509371 1509373 1509373 1509374 1509376 1509377 1509378	d \s	tab\salo ncoding [log [log [log [log [log [log [log [lo	ompbyst UTF-EL jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	rap.bt (Lscala/: (Lscala/: (Lscala/: (Lscala/: (Ljava/la Adding st Adding st Adding st Adding st	C: temp\disca Top line 15093 1509369 1509370 1509371 1509372 1509373 1509374 1509375	rd\stab\sal 369Encoding [log [log [log [log [log [log [log [lo	jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	hird.bt Ine end style: L (Lscala/r (Lscala/r (Lscala/r Adding st Adding st Adding st Adding st
A (Base): iscan Top line 150937 1509369 1509370 1509371 1509373 1509374 1509375 1509375 1509377	d(stab)salcd SEncoding: [log [log [log [log [log [log [log [log	ompbyq UTF-{L jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	uidk.btt (Lscala/r (Lscala/r (Lscala/r (Lscala/r No mirroz (Ljava/la Adding st Adding st Adding st Adding st	B: temp\discar Top Ine 15093 1509369 1509370 1509371 1509373 1509373 1509374 1509375 1509377 1509377	d\s 69	tab\salo ncoding [log [log [log [log [log [log [log [lo	ompbyst UTF-&L jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	rap.bt ine end style: 1 (Lscala/2 (Lscala/2 (Lscala/2 No mirroo (Ljava/1 Adding st Adding st Adding st Adding st	C: temp\disca Top line 1509 1509370 1509370 1509371 1509372 1509373 1509374 1509376 1509376	rd\stab\sal 36\fraccoding [log [log [log [log [log [log [log [lo	jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	third.btt ine end style: L (Lscala/r (Lscala/r (Lscala/r Adding st Adding st Adding st Adding st Adding st
A (Base): iscan Top line 15093 1509369 1509370 1509372 1509373 1509374 1509375 1509376 1509377	d/stab/salce 59Encoding: [log [log [log [log [log [log [log [log	ompbyq UTF-{L jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	uidk.bt ine end style: ((Lscala/r (Lscala/c (Lscala/c (Ljava/la Adding st (Ljava/la Adding st Adding st Adding st	B: temp\discard Top Ine: 15093 1509369 1509370 1509371 1509372 1509373 1509374 1509376 1509377 1509378	d\s 69	tab\salo incoding [log [log [log [log [log [log [log [lo	jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	rap.bt ine end style: U (Lscala/: (Lscala/: (Lscala/: (Lscala/: (Ljava/ld Adding st Adding st Adding st Adding st	C: (temp\disca Top line 1509; 1509369 1509370 1509370 1509373 1509373 1509374 1509374 1509376 1509376	ard/stab/sal 36%Encoding [log [log [log [log [log [log [log [lo	<pre>compby: jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]</pre>	third.bt ine end style: L (Lscala/c (Lscala/c (Lscala/c (Lscala/r Adding st Adding st Adding st Adding st No mirror (Ljava/la
A (Base): iscan Top line 15093 1509369 1509370 1509371 1509372 1509374 1509374 1509376 1509377 1509378 1509379	d/stab/salce 65Encoding: [log [log [log [log [log [log [log [log	ompbyq : UTF-{L jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	wick.bt ine end style: ((Lscala/r (Lscala/r (Lscala/r No mirror (Ljava/la Adding st Adding st Adding st Adding st Adding st Adding st	B: temp\discarr Top line: 15093 1509369 1509370 1509371 1509372 1509373 1509374 1509376 1509377 1509378 1509379	d/s	tab\salo incoding [log [log [log [log [log [log [log [lo	jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]	No mirror	C: (temp\disca Top line 1509: 1509369 1509370 1509371 1509372 1509373 1509374 1509374 1509375 1509376 1509378 1509378	ard/stab/sal 36%Encoding [log [log [log [log [log [log [log [lo	<pre>compby: : UTF-{I jvm] jvm] jvm] jvm] jvm] jvm] jvm] jvm]</pre>	hird.bt ine end style: L (Lscala/z (Lscala/z (Lscala/z Adding st Adding st Adding st Adding st Adding st Adding st Adding st No mirror

<ロト</th>
 ・< 目 > < 目 > < 目 > 5/10

-Status as of M3

Not yet merged: Faster reaching-defs for ClosureElim

Reaching Definitions

- Status quo: 12sec (but with mismatched stack sizes)
- Fixing that: 25sec
- With some improvements (two Ints in a Long, etc): 7sec
- Details at https://github.com/magarciaEPFL/scala/tree/fasterRDef

47c96c22	i magarciaEPFL	no more mismatched stack sizes in reaching def analysis	2 months ago
dcac1d87	💓 magarciaEPFL	lattice meet operation is associative and commutative (at least on te	2 months ago
32d5d099	magarciaEPFL	towards replacing list scanning with hash operations	2 months ago
ded71a7b	imagarciaEPFL	just one iteration over instructions in rdef.init()	2 months ago
72581324	magarciaEPFL	no separate set needed for kill(b), it's always == gen(b).keySet	2 months ago
aac6070f	magarciaEPFL	map replacements doing away with list scans (more to come)	2 months ago
2c7f3d92	imagarciaEPFL	using persistent data structures	2 months ago
e55cee9b	💓 magarciaEPFL	dawn of alt reaching defs	2 months ago
57d9320d	💓 magarciaEPFL	one iteration fewer over all instructions rdef.interpret()	2 months ago
628a7ca1	magarciaEPFL	replaced old rdef also in ICodeReader	2 months ago
8a87893e	imagarciaEPFL	changes in the management of the worklist	2 months ago
274dd501	💓 magarciaEPFL	stop passing Tuple2[BasicBlock, Int] around	2 months ago

-Status as of M3

-Not yet merged: GenASM

ASM-based backend:

- twice as fast as GenJVM, even faster with "Instruction.emit (asm.MethodVisitor)"
- bootstraps, passes all tests, including:

```
ant quick.clean -Dscalac.args.quickonly="-target:jvm-1.5"
test.stability
```

```
similarly for jvm-1.5-asm and jvm-1.6
```

Alternatives for build.xml:

- Download asm.jar and asm-util.jar from Maven, re-namespace on-the-fly viar JarJar
- 2. Distribute ASM sources "re-namespaced" by us

Details at https://groups.google.com/d/topic/scala-internals/7gecxktTUWs/discussion

・ロト・日本・モン・モン・モン・アン10

-Ongoing work

- Single-pass Type-flow analysis

Why bother making inlining faster? Currently:

inliner 22 54sec 30%

Gist of Single-pass Type-Flow Analysis (TFA)

- 1. BasicBlock instructions are scanned at most once, collecting its *net effect* on the output lattice elem ("single-pass")
- 2. Afterwards, the iterative dataflow uses the net-effects.
- 3. On average, twice faster as MethodTFA.

What can be done with SinglePassTFA:

- integrate the solution repair approach used in inliner (extra 5x speedup)
- more scalable concurrency (lower contention on typer as compared to MethodTFA)

Morale: avoid repeated pattern matching over ICode instructions.

Details at https://github.com/magarciaEPFL/scala/tree/SinglePassTA () + () + ()

- Ongoing work

- Open performance riddles around inlining

Riddle 1:

```
def isMonadicMethod(sym: Symbol) = {
    nme.unspecializedName(sym.name) match {
        case nme.foreach |
            nme.filter | nme.withFilter |
            nme.map | nme.flatMap => true
        case _ => false
    }
}
```

Any other method "m"

(1) not explicitly marked <code>@inline</code> (2) taking a closure as last arg; won't have its callsites inlined thus preventing <code>ClosureElim</code> from eliminating anonymous-closure-classes ref'ed at m callsites.

Riddle 2:

```
/* TODO
 * Do we really want to inline inside exception handlers?
 * Seems counterproductive
 * (larger methods less likely to be JITed). */
```

-Next steps

Next steps

- Phasing into trunk the improvements described above.
- ▶ Yes, but *which ones* aiming for *which release*? (M4, RC1, 2.11)

<ロ><日><日><日><日</th><日><日><日</td><10/10</td>