Made runtime=compiled the default

Merged **Mats-SX** merged 4 commits into `neo4j:2.3` from `pontusmelke:2.3-compiled-default` May 1, 2015

Conversation 28 Commits 4 Files changed 78 +593 −224

pontusmelke commented Apr 23, 2015 Collaborator

Fixed some bugs in compiled plans showing up when running all tests

- Explain in compiled methods
- Proper error handling of user specified exceptions and `KernelExceptions`
- Fixed statement handling on when closing iterators
- Rewrote `dumpToString` such that it uses `accept` instead of `toList`
- Forced some tests to use interpreted runtime which uses `PROFILE`

Labels
2.3
cypher

Milestone
No milestone

Assignee
No one assigned

3 participants

pontusmelke added cypher 2.3 labels Apr 23, 2015

pontusmelke changed the title from **Made CompiledPlans the default** to **Made runtime=compiled the default** Apr 23, 2015

davidegrohmann commented on an outdated diff Apr 23, 2015 Show outdated diff

Mats-SX commented on an outdated diff Apr 30, 2015 Show outdated diff

Mats-SX commented on the diff Apr 30, 2015

...g/neo4j/cypher/internal/compiler/v2_3/birk/Javac.java View full changes

```
@@ -99,13 +98,13 @@ public CompilationError( String message )
 99   98              return clazz;
100   99          }
101  100
102        -    public static InternalExecutionResult newInstance( Class<InternalExecut
     101   +    public static InternalExecutionResult newInstance( Class<InternalExecut
```

Mats-SX added a note Apr 30, 2015 Collaborator

This is going to conflict with the profiling PR. I say we merge this one first, and I can rebase the other one.

Add a line note

Mats-SX referenced this pull request Apr 30, 2015
Profiling support for compiled runtime #4549 Merged

Mats-SX commented on the diff Apr 30, 2015

...compiler/v2_3/birk/codegen/LogicalPlanConverter.scala View full changes

```
@@ -182,7 +182,9 @@ object LogicalPlanConverter {
182  182         (methodHandle, ProjectProperties(projectionInstructions, action))
183  183       }
184  184
185       -    private def createProjectionInstruction(expression: Expression, context
     185  +    private def createProjectionInstruction(expression: Expression, context
```

Mats-SX added a note Apr 30, 2015 Collaborator

Return type annotation should not be needed here (given by name anyway).

pontusmelke added a note May 1, 2015 Collaborator

it is needed, called recursively

Add a line note

Mats-SX commented on the diff Apr 30, 2015

...pher/internal/compiler/v2_3/birk/il/Instruction.scala View full changes

```
((6 lines not shown))
32  34      def generateInit(): String
33  35
34  36      def methods: Seq[Method] = {
35  37        (allLeafs :+ this).flatMap(_._method)
36  38      }
37  39
    40  +   def exceptions: Set[ExceptionCodeGen] = allLeafs.flatMap(_._exceptions()
```

Mats-SX added a note Apr 30, 2015 Collaborator

Make final

pontusmelke added a note Apr 30, 2015 Collaborator

Having a final method an a trait seems a bit weird. However I do agree that this needs to be cleaned up but let's do that as separate PR.

Mats-SX added a note Apr 30, 2015 Collaborator

Fair enough.

Add a line note

Mats-SX commented on the diff Apr 30, 2015

...pher/internal/compiler/v2_3/birk/il/Instruction.scala View full changes

```
@@ -58,5 +64,7 @@ object Instruction {
58  64        override def fields(): String = ""
59  65
60  66        override def generateInit(): String = ""
    67  +
    68  +     override def exceptions: Set[ExceptionCodeGen] = Set.empty
```

Mats-SX added a note Apr 30, 2015 Collaborator

It should override the underscore version, shouldn't it? I understand that `f` is the aggregating function and `_f` is the "local" function, is this correctly interpreted?

pontusmelke added a note Apr 30, 2015 Collaborator as above let's clean this up in a later PR, if we want to change this we should change all the methods in this trait

Add a line note

Mats-SX commented on the diff Apr 30, 2015

`...pher/internal/compiler/v2_3/birk/il/Instruction.scala` View full changes

```
((14 lines not shown))
38  42      // Generates import list for class
39  43      protected def _importedClasses(): Set[String] = Set.empty
40  44
    45  +   protected def _exceptions(): Set[ExceptionCodeGen] = Set.empty
```

Mats-SX added a note Apr 30, 2015 Collaborator

We could make this abstract to clarify the difference between it and the non-underscore variant, but that would on the other hand mean duplicated `Set.empty` implementations in subclasses... I'm torn on this one.

pontusmelke added a note Apr 30, 2015 Collaborator

I agree but let's not do this now.

Add a line note

Mats-SX commented on an outdated diff Apr 30, 2015 Show outdated diff

Mats-SX commented on the diff Apr 30, 2015

`...nal/compiler/v2_3/birk/il/ProjectionInstruction.scala` View full changes

```
@@ -150,6 +163,8 @@ case class ProjectSubtraction(lhs: ProjectionInstructio
150 163
151 164     def fields() = ""
152 165
    166 +   override def children = Seq(lhs, rhs)
```

Mats-SX added a note Apr 30, 2015 Collaborator

Ah, you found that bug too :)

Add a line note

Mats-SX commented on an outdated diff Apr 30, 2015 Show outdated diff

Mats-SX commented on an outdated diff Apr 30, 2015 Show outdated diff

Mats-SX commented on the diff Apr 30, 2015

`...iler/v2_3/executionplan/CompiledExecutionResult.scala` View full changes

((38 lines not shown))

```
     174  +
     175  +    val keyValStrings = properties.
     176  +      map(pkId => readOperations.propertyKeyGetName(pkId) + ":" + text(prop
     177  +
     178  +    keyValStrings.mkString("{", ",", "}")
     179  +  }
     180  +
     181  +  def text(a: Any): String = a match {
     182  +    case x: Node              => x.toString + props(x)
     183  +    case x: Relationship      => ":" + x.getType.name() + "[" + x.getId + "]"
     184  +    case x if x.isInstanceOf[Map[_, _]] => makeString(x.asInstanceOf[Map[S
     185  +    case x if x.isInstanceOf[java.util.Map[_, _]] => makeString(x.asInstanc
     186  +    case IsCollection(coll) => coll.map(elem => text(elem)).mkString("[", '
     187  +    case x: String            => "\"" + x + "\""
     188  +    case v: KeyToken          => v.name
     189  +    case Some(x)              => x.toString
```

Mats-SX added a note Apr 30, 2015 — Collaborator

We match on `Some(x)` , but not on `None` . Is this correct?

pontusmelke added a note Apr 30, 2015 — Collaborator

None will be matched on 192, `case x => x.toString`

Mats-SX added a note Apr 30, 2015 — Collaborator

Great!

Add a line note

Mats-SX commented on an outdated diff Apr 30, 2015 — Show outdated diff

Mats-SX commented on an outdated diff Apr 30, 2015 — Show outdated diff

Mats-SX commented on an outdated diff Apr 30, 2015 — Show outdated diff

Mats-SX commented on the diff Apr 30, 2015

`...r/internal/compiler/v2_3/birk/CodeGeneratorTest.scala` — View full changes

((22 lines not shown))

```
437  436        }
438  437
439       -  private def compile(plan: LogicalPlan, params: Map[String, AnyRef] = Map.
     438  +  test("should throw the same error as the user provides") {
     439  +    // given
     440  +    val plan = ProduceResult(List.empty, List.empty, List("a"), Projection(
     441  +
     442  +    // when
     443  +    val closer = mock[TaskCloser]
     444  +    val compiled = compile( plan, taskCloser = closer )
     445  +
     446  +    // then
     447  +    val visitor = mock[ResultVisitor[RuntimeException]]
     448  +    val exception = new scala.RuntimeException()
     449  +    when(visitor.visit(any[ResultRow])).thenThrow(exception)
```

Mats-SX added a note Apr 30, 2015 — Collaborator

I don't understand this test. Why are we testing that the mocking framework throws the correct exception for us? Shouldn't we send bad input in and let the function throw by itself?

pontusmelke added a note Apr 30, 2015 Collaborator

we are testing that if a user sends in a `ResultsVisitor<MyAwesomeException>` and that visitor throws we should propagate that exception to the user and not hide it. The outer (java) signature is something like

```
<E extends Exception> void accept( ResultVisitor<E> visitor )
            throws E;
```

thus we must propagate `E`

Mats-SX added a note Apr 30, 2015 Collaborator

I see. Thanks!

Add a line note

Mats-SX commented on an outdated diff Apr 30, 2015 Show outdated diff

pontusmelke added some commits Apr 23, 2015

`Made CompiledPlans the default …`	40ff6cd
`Rewrote dumpToString so that is using accept instead of toList`	ef9b0b6
`Added ability to call both toList and dumpToString on CompiledExecuti… …`	e2e1079
`Improved error handling, handle user errors and KernelExceptions prop… …`	0ded4de

Mats-SX merged commit **c6329f2** into `neo4j:2.3` May 1, 2015 View details

1 check passed

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment