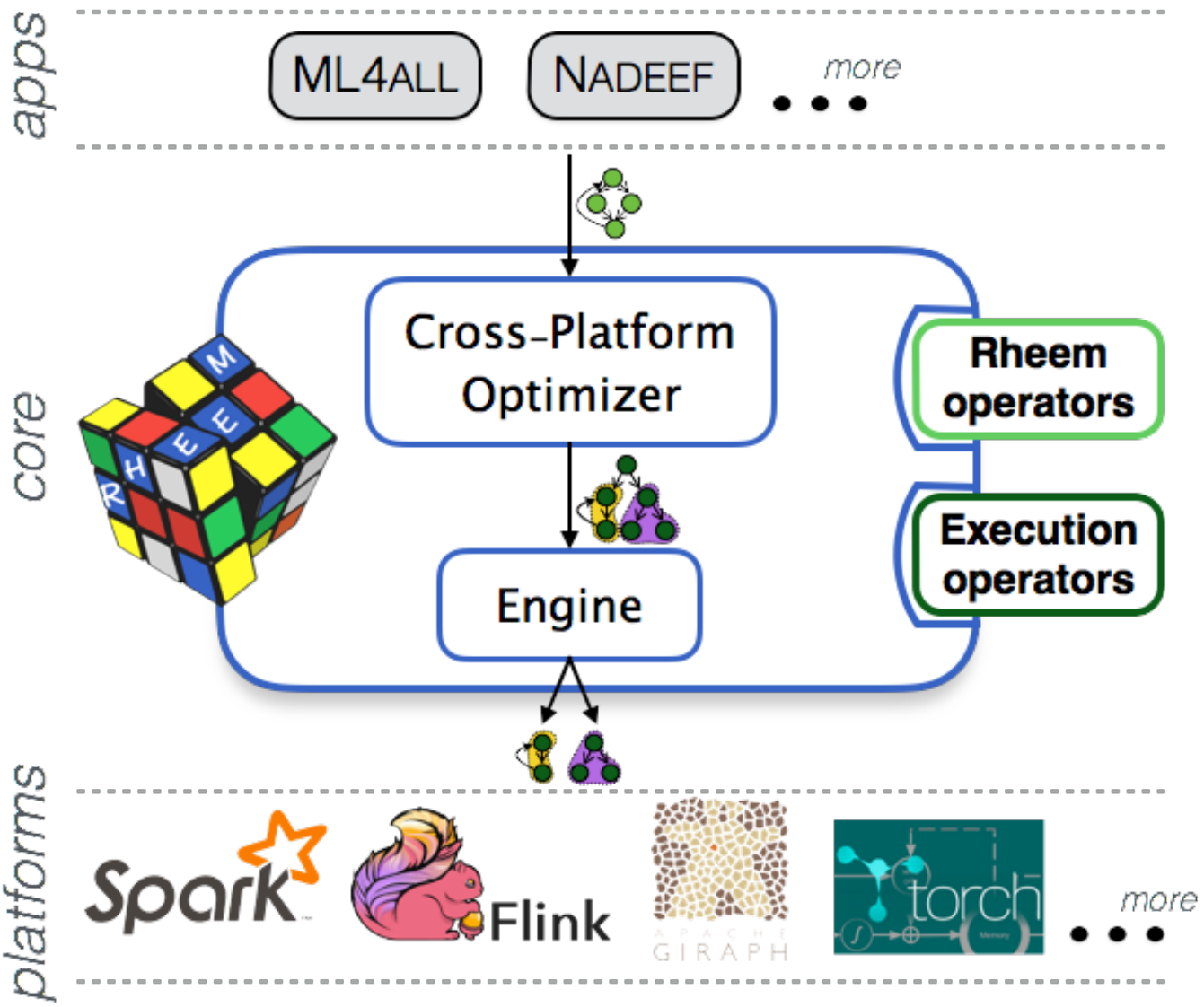
Rheem：可扩展且易于使用的跨平台大数据分析系统

RHEEM是一个可扩展且易于使用的跨平台大数据分析系统，它在现有的数据处理平台之上提供了一个抽象。它允许用户使用易于使用的编程接口轻松地编写数据分析任务，为开发者提供了不同的方式进行性能优化，编写好的程序可以在任意数据处理平台上运行，这其中包括：Postgre SQL, Spark, Hadoop MapReduce或者Flink等；Rheem将选择经典处理框架的最佳配置来运行这些程序。RHEEM抽象完全基于用户定义函数（UDF），允许用户专注于其应用程序逻辑而不是物理细节。这就使得数据工程师和软件开发人员可以不去了解不同数据处理系统的API、优缺点以及不同平台之前通信的复杂性等繁琐工作。从上面的特点可以看出，其目标和去年Google开源的Apache Beam很类似。直到目前，Rheem内置支持以下的数据处理平台：

- Java 8 Streams
- Apache Spark
- GraphChi
- Postgres
- SQLite

rheem的体系结构如下：



RHEEM Architecture

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

如何使用Rheem

Rheem需要我们安装好Java8，然后根据自己的需求在pom.xml文件里面引入下面的依赖：

```
<dependency>
  <groupId>org.qcri.rheem</groupId>
  <artifactId>rheem-***</artifactId>
  <version>0.2.1</version>
</dependency>
```

注意上面的 ***，因为Rheem包含了很多个模块，我们需要根据自己的需求选择不同的模块，主要模块介绍如下：

- rheem-core: 提供了核心数据结构和优化器，这个模块必须引入；
- rheem-basic: 提供了通用的运算符和数据类型；
- rheem-api: 提供了Java和Scala语言的API供大家使用；
- rheem-java, rheem-spark, rheem-graphchi, rheem-sqlite3, rheem-postgres: 适用于各种平台的适配器
- rheem-profiler: provides functionality to learn operator and UDF cost functions from historical execution data

下面介绍如何使用RHEEM编写一个WordCount程序。这里以Scala API进行介绍：

```
import org.qcri.rheem.api._
import org.qcri.rheem.core.api.{Configuration, RheemContext}
import org.qcri.rheem.java.Java
import org.qcri.rheem.spark.Spark

object WordcountScala {
  def main(args: Array[String]) {

    // Settings
    val inputUrl = "file:/tmp.txt"

    // Get a plan builder.
    val rheemContext = new RheemContext(new Configuration)
      .withPlugin(Java.basicPlugin)
      .withPlugin(Spark.basicPlugin)
    val planBuilder = new PlanBuilder(rheemContext)
      .withJobName(s"WordCount ($inputUrl)")
      .withUdfJarsOf(this.getClass)

    val wordcounts = planBuilder
      // Read the text file.
      .readTextFile(inputUrl).withName("Load file")

      // Split each line by non-word characters.
      .flatMap(_.split("₩₩W+"), selectivity = 10).withName("Split words")

      // Filter empty tokens.
      .filter(_.nonEmpty, selectivity = 0.99).withName("Filter empty words")
```

```
    // Attach counter to each word.
    .map(word => (word.toLowerCase, 1)).withName("To lower case, add counter")

    // Sum up counters for every word.
    .reduceByKey(_._1, (c1, c2) => (c1._1, c1._2 + c2._2)).withName("Add counters")
    .withCardinalityEstimator((in: Long) => math.round(in * 0.01))

    // Execute the plan and collect the results.
    .collect()

  println(wordcounts)
 }
}
```

从上面的代码可以看出，这个代码的函数和处理过程和使用Spark或者Flink开发程序流程很类似，然后我们可以使用下面命令运行这个程序：

```
java  com.iteblog.WordcountScala
```

然后就
可以在Spark上
运行这个程序。更多关于RHE
EM的介绍可以参见期官方文档介绍：https://github.com/daqcri/rheem