



操作台  
传智播客  
回收茅台酒

CSDN首页 > 云计算

[订阅云计算RSS](#)

## Project Tungsten：让Spark将硬件性能压榨到极限

发表于 2015-04-30 13:22 | 8293次阅读 | 来源 Databricks Blog | 12 条评论 | 作者 Reynold Xin、Josh Rosen

大数据    开源    Spark    Tungsten

摘要：对于Spark来说，通用只是其目标之一，更好的性能同样是其赖以生存的立足之本。北京时间4月28日晚，Databricks在其官方博客上发布了Tungsten项目，并简述了Spark性能提升下一阶段的RoadMap。

本文编译自Databricks Blog ([Project Tungsten: Bringing Spark Closer to Bare Metal](#))，作者 Reynold Xin ([@hashjoin](#))、Josh Rosen。由七牛云存储技术总监陈超 ([@CrazyJvm](#)) 友情审校。

以下为原文：

在之前的[博文](#)中，我们回顾和总结了2014年Spark在性能提升上所做的努力。本篇博文中，我们将为你介绍性能提升的下一阶段——Tungsten。在2014年，我们目睹了Spark缔造大规模排序的新世界纪录，同时也看到了Spark整个引擎的大幅度提升——从Python到SQL再到机器学习。

Tungsten项目将是Spark自诞生以来内核级别的最大改动，以大幅度提升Spark应用程序的内存和CPU利用率为目标，旨在最大程度上压榨新时代硬件性能。Project Tungsten包括了3个方面的努力：

- **Memory Management和Binary Processing:** 利用应用的语义（application semantics）来更明确地管理内存，同时消除JVM对象模型和垃圾回收开销。
- **Cache-aware computation**（缓存友好的计算）：使用算法和数据结构来实现内存分级结构（memory hierarchy）。
- 代码生成（**Code generation**）：使用代码生成来利用新型编译器和CPU。

之所以大幅度聚焦内存和CPU的利用，其主要原因就在于：对比IO和网络通信，Spark在CPU和内存上遭遇的瓶颈日益增多。详细信息可以查看最新的大数据负载性能研究（[Ousterhout](#)），而我们在为Databricks Cloud用户做优化调整时也得出了类似的结论。

为什么CPU会成为新的瓶颈？这里存在多个问题：首先，在硬件配置中，IO带宽提升的非常明显，比如10Gbps网络和SSD存储（或者做了条文化处理的HDD阵列）提供的高带宽；从软件的角度来看，通过Spark优化器基于业务对输入数据进行剪枝，当下许多类型的工作负载已经不会再需要使用大量的IO；在Spark Shuffle子系统中，对比底层硬件系统提供的原始吞吐量，序列化和哈希（CPU相关）成为主要瓶颈。从种种迹象来看，对比IO，Spark当下更受限于CPU效率和内存压力。

## 1. Memory Management和Binary Processing

在JVM上的应用程序通常依赖JVM的垃圾回收机制来管理内存。毫无疑问，JVM绝对是一个伟大的工程，为不同工作负载提供了一个通用的运行环境。然而，随着Spark应用程序性能的不断提升，JVM对象和GC开销产生的影响将非常致命。



**CSDN官方微信**  
扫描二维码,向CSDN吐槽  
微信号: CSDNnews



程序员移动端订阅下载

## 每日资讯快速浏览

## 微博关注



CSDN云计算 北京 朝阳区

领英称即将开源他们内部的应用软件WhereHow  
s, 一个企业级的数据挖掘软件。准确的说, 领英称  
它为“数据发现软件”。从商业角度讲, WhereHow  
的目标是从分布式的多种元数据中进行挖掘。ht  
p://t.cn/RGQBmLb

3月7日 17:15

转发(2) | 评论

Cisco今天宣布以 2.6 亿美元收购混合云环境应用管

一直以来，Java对象产生的开销都非常大。在UTF-8编码上，简单如“abcd”这样的字符串也需要4个字节进行储存。然而，到了JVM情况就更糟糕了。为了更加通用，它重新定制了自己的储存机制——使用UTF-16方式编码每个字符（2字节），与此同时，每个String对象还包含一个12字节的header，和一个8字节的哈希编码，我们可以从Java Object Layout工具的输出上获得一个更清晰的理解：

```
[java]  C  ?
```

```
1. java.lang.String object internals:
```

OFFSET	SIZE	TYPE	DESCRIPTION	VALUE
0	4		(object header)	...
4	4		(object header)	...
8	4		(object header)	...
12	4	char[]	String.value	[]
16	4	int	String.hash	0
20	4	int	String.hash32	0

```
9. Instance size: 24 bytes (reported by Instrumentation API)
```

毫无疑问，在JVM对象模型中，一个4字节的字符串需要48字节的空间来存储！

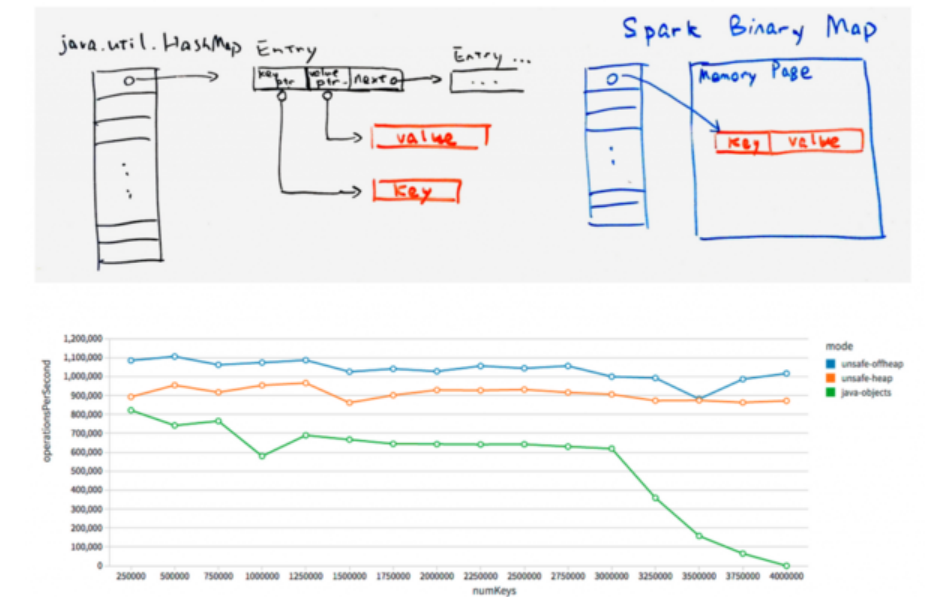
JVM对象带来的另一个问题是GC。从高等级上看，通常情况下GC会将对象划分成两种类型：第一种会有很高的allocation/deallocation（年轻代），另一种的状态非常稳定（年老代）。通过利用年轻代对象的瞬时特性，垃圾收集器可以更有效率地对其进行管理。在GC可以可靠地估算对象的生命周期时，这种机制可以良好运行，但是如果只是基于一个很短的时间，这个机制很显然会遭遇困境，比如对象忽然从年轻代进入到年老代。鉴于这种实现基于一个启发和估计的原理，性能可以通过GC调优的一些“黑魔法”来实现，因此你可能需要给JVM更多的参数让其弄清楚对象的生命周期。

然而，Spark追求的不仅仅是通用性。在计算上，Spark了解每个步骤的数据传输，以及每个作业和任务的范围。因此，对比JVM垃圾收集器，Spark知悉内存块生命周期的更多信息，从而在内存管理上拥有比JVM更具效率的可能。

为了扭转对象开销和无效率GC产生的影响，我们引入了一个显式的内存管理器让Spark操作可以直接针对二进制数据而不是Java对象。它基于sun.misc.Unsafe建立，由JVM提供，一个类似C的内存访问功能（比如explicit allocation、deallocation和pointer arithmetics）。此外，Unsafe方法是内置的，这意味着，每个方法都将由JIT编译成单一的机器指令。

在某些方面，Spark已经开始利用内存管理。2014年，Databricks引入了一个新的基于Netty的网络传输机制，它使用一个类jemalloc的内存管理器来管理所有网络缓冲。这个机制让Spark shuffle得到了非常大的改善，也帮助了Spark创造了新的世界纪录。

新内存管理的首次亮相将出现在Spark 1.4版本，它包含了一个由Spark管理，可以直接在内存中操作二进制数据的hashmap。对比标准的Java HashMap，该实现避免了很多中间环节开销，并且对垃圾收集器透明。



当下，这个功能仍然处于开发阶段，但是其展现的初始测试行能已然令人兴奋。如上图所示，我们在



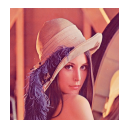
## 相关热门文章

关于云上大数据应用开发，IBM Bluemix工程师...

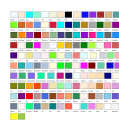
## 热门标签

Hadoop	AWS	移动游戏
Java	Android	iOS
Swift	智能硬件	Docker
OpenStack	VPN	Spark
ERP	IE 10	Eclipse
CRM	JavaScript	数据库
Ubuntu	NFC	WAP


## 下载专辑




图像拼接外文论文原著



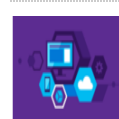
DIV-CSS



HTML5精品书籍



2016HTML5+CSS3专辑



C#设计模式+ASP.NET MVC框架开发教程

3个不同的途径中对比了聚合计算的吞吐量——开发中的新模型、offheap模型、以及java.util.HashMap。新的hashmap可以支撑每秒超过100万的聚合操作，大约是java.util.HashMap的两倍。更重要的是，在没有太多参数调优的情况下，随着内存利用增加，这个模式基本上不存在性能的衰弱，而使用JVM默认模式最终已被GC压垮。

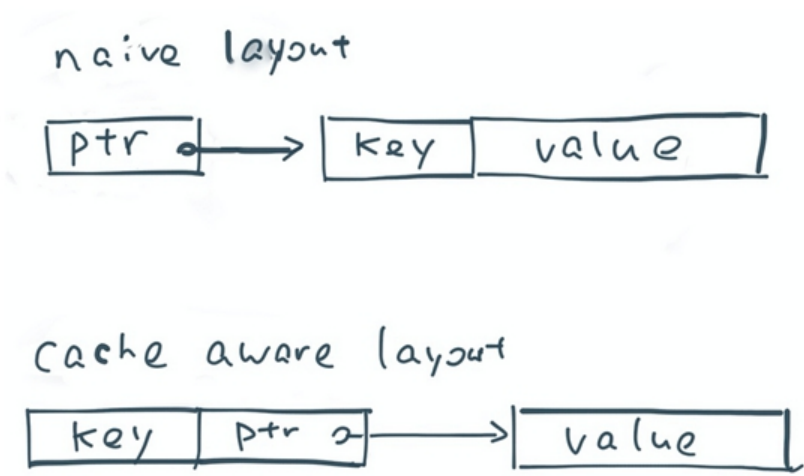
在Spark 1.4中，这个hashmap可以为DataFracmes和SQL的聚合处理使用，而在1.5中，我们将为其他操作提供一个让其利用这个特性的数据结构，比如sort和join。毫无疑问，它将应用到大量需要调优GC以获得高性能的场景。

2. Cache-aware computation（缓存友好的计算）

在解释Cache-aware computation之前，我们首先回顾一下“内存计算”，也是Spark广为业内知晓的优势。对于Spark来说，它可以更好地利用集群中的内存资源，提供了比基于磁盘解决方案更快的速度。然而，Spark同样可以处理超过内存大小的数据，自动地外溢到磁盘，并执行额外的操作，比如排序和哈希。

类似的情况，Cache-aware computation通过使用 L1/ L2/L3 CPU缓存来提升速度，同样也可以处理超过寄存器大小的数据。在给用户Spark应用程序做性能分析时，我们发现大量的CPU时间因为等待从内存中读取数据而浪费。在 Tungsten项目中，我们设计了更加缓存友好的算法和数据结构，从而让Spark应用程序可以花费更少的时间等待CPU从内存中读取数据，也给有用工作提供了更多的计算时间。

我们不妨看向对记录排序的例子。一个标准的排序步骤需要为记录储存一组的指针，并使用quicksort来互换指针直到所有记录被排序。基于顺序扫描的特性，排序通常能获得一个不错的缓存命中率。然而，排序一组指针的缓存命中率却很低，因为每个比较运算都需要对两个指针解引用，而这两个指针对应的却是内存中两个随机位置的数据。



那么，我们该如何提高排序中的缓存本地性？其中一个方法就是通过指针顺序地储存每个记录的sort key。举个例子，如果sort key是一个64位的整型，那么我们需要在指针阵列中使用128位（64位指针，64位sort key）来储存每条记录。这个途径下，每个quicksort对比操作只需要线性的查找每对pointer-key，从而不会产生任何的随机扫描。希望上述解释可以让你对我们提高缓存本地性的方法有一定的了解。

这样一来，我们又如何将这些优化应用到Spark？大多数分布式数据处理都可以归结为多个操作组成的一个小列表，比如聚合、排序和join。因此，通过提升这些操作的效率，我们可以从整体上提升Spark。我们已经为排序操作建立了一个新的版本，它比老版本的速度快5倍。这个新的sort将会被应用到sort-based shuffle、high cardinality aggregations和sort-merge join operator。在2015年底，所有Spark上的低等级算法都将升级为cache-aware，从而让所有应用程序的效率都得到提高——从机器学习到SQL。

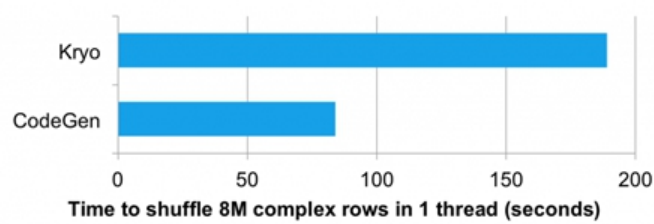
3. 代码生成

大约在1年前，Spark引入代码生成用于SQL和DataFrames里的表达式求值（expression evaluation）。表达式求值的过程是在特定的记录上计算一个表达式的值（比如age > 35 && age <

40)。当然，这里是在运行时，而不是在一个缓慢的解释器中为每个行做单步调试。对比解释器，代码生成去掉了原始数据类型的封装，更重要的是，避免了昂贵的多态函数调度。

在之前的博文中，我们阐述了代码生成可以加速（接近一个量级）多种TPC-DS查询。当下，我们正在努力让代码生成可以应用到所有的内置表达式上。此外，我们计划提升代码生成的等级，从每次一条记录表达式求值到向量化表达式求值，使用JIT来开发更好的作用于新型CPU的指令流水线，从而在同时处理多条记录。

在通过表达式求值优化内部组件的CPU效率之外，我们还期望将代码生成推到更广泛的地方，其中一个就是shuffle过程中将数据从内存二进制格式转换到wire-protocol。如之前所述，取代带宽，shuffle通常会因数据序列化出现瓶颈。通过代码生成，我们可以显著地提升序列化吞吐量，从而反过来作用到shuffle网络吞吐量的提升。



上面的图片对比了单线程对800万复杂行做shuffle的性能，分别使用的是Kryo和代码生成，在速度上后者是前者的2倍以上。

Tungsten和未来的工作

在未来的几个版本中，Tungsten将大幅度提升Spark的核心引擎。它首先将登陆Spark 1.4版本，包括了Dataframe API中聚合操作的内存管理，以及定制化序列化器。二进制内存管理的扩展和cache-aware数据结构将出现在Spark 1.5的部分项目（基于DataFrame模型）中。当然如果需要的话，这个提升也会应用到Spark RDD API。

对于Spark，Tungsten是一个长期的项目，因此也存在很多的可能性。值得关注的是，我们还将考察LLVM或者OpenCL，让Spark应用程序可以利用新型CPU所提供的SSE/SIMD指令，以及GPU更好的并行性来提升机器学习和图的计算。

Spark不变的目标就是提供一个单一的平台，让用户可以从中获得更好的分布式算法来匹配任何类型的数据处理任务。其中，性能一直是主要的目标之一，而Tungsten的目标就是让Spark应用程序达到硬件性能的极限。更多详情可以持续关注Databricks博客，以及6月旧金山的Spark Summit。

本文为CSDN原创文章，未经允许不得转载，如需转载请联系market#csdn.net(#换成@)

顶

8

踩

0

推荐阅读相关主题：

- 相关文章
- 最新报道
- 旧金山Spark Summit 2015，见证中国的技术力量

BDTC PPT集萃（一）：IBM、Intel、Hortonworks...

Project Tungsten: 让Spark将硬件性能压榨到极限


搭建高可用的MongoDB集群（上）：MongoDB的...

Spark SQL深度理解篇：模块实现、代码结构及执行...

轻松搞定TB级数据，开源GraphLab突破人类图计算...

已有12条评论

还可以再输入500个字



有什么感想，你也来说说吧！

yxydde

欢迎您！

发表评论

最新评论

最热评论

- 

BigData大数据

2015-05-01 10:09 来自 新浪微博

回复@石宣化:冲剂一篇A 别等伯克利发了.....

回复
- 

BigData大数据

2015-05-01 09:50 来自 新浪微博

你们团队总是走在技术最尖端[赞][赞]

回复
- 

BigData大数据

2015-05-01 04:57 来自 新浪微博

这个介绍还是比较粗粒度！要了解这个项目深入，还是六月份直接来三藩参加 SPARK SUMMIT吧！注册：<http://t.cn/zR0cMAI> @hashjoin @CrazyJvm //@冠诚: 我咋这么喜欢看到"压榨性能极限"这几个字呢？[嘻嘻] //@hashjoin:翻译成中文了。

回复
- 

BigData大数据

2015-05-01 04:57 来自 新浪微博

这个介绍还是比较粗粒度！要了解这个项目深入，还是六月份直接来三藩参加 SPARK SUMMIT吧！注册：<http://t.cn/zR0cMAI> @hashjoin @CrazyJvm //@冠诚: 我咋这么喜欢看到"压榨性能极限"这几个字呢？[嘻嘻] //@hashjoin:翻译成中文了。

回复
- 

hashjoin

2015-05-01 01:48 来自 新浪微博

回复@Penguini:哈哈多年没变

回复
- 

crazyjvm

2015-04-30 18:59 来自 新浪微博

的确是仅仅指bare metal

回复
- 

crazyjvm

2015-04-30 17:27 来自 新浪微博

回复@明风Andy:浏览过。其实也没帮他们看什么，就说了两个三个明显有问题的地方。

回复
- 

牛客网

2015-04-30 16:04 来自 新浪微博

//@老师木: 这个目前是处在“目标”的阶段吧

回复
- 

牛客网

2015-04-30 16:04 来自 新浪微博

//@老师木: 这个目前是处在“目标”的阶段吧

回复
- 

hashjoin

2015-04-30 15:46 来自 新浪微博



我在白板上画完拍的照片。

回复

 可口可乐 2015-04-30 14:33  
Cool

回复

 hashjoin 2015-04-30 13:58 来自 新浪微博  
翻译成中文了。

回复

共1页

首页

上一页


1

下一页

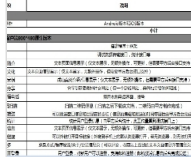
末页

请您注意

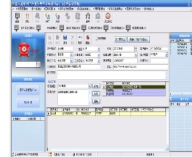
- 自觉遵守：爱国、守法、自律、真实、文明的原则
- 尊重网上道德，遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规
- 严禁发表危害国家安全，破坏民族团结、国家宗教政策和社会稳定，含侮辱、诽谤、教唆、淫秽等内容的作品
- 承担一切因您的行为而直接或间接导致的民事或刑事责任
- 您在CSDN新闻评论发表的作品，CSDN有权在网站内保留、转载、引用或者删除
- 参与本评论即表明您已经阅读并接受上述条款




微信源码




app开发报价单




呼叫中心系统




舆情监控系统



毕业论文代做



新西兰移民条件



澳大利亚买房

项目管理软件

在线考试系统

青鸟学校

ios学习路线

在职研究生报名

linux学..

智能玻璃

客户管理系统

app外包