Google集群使用情况跟踪：格式+架构

查尔斯·赖斯，约翰·威尔克斯，约瑟夫·赫勒斯坦

跟踪版本2的2013-05-06版本。跟踪版本2.1的修订日期为2014-11-17。状态：已导出到Google外部。

版权所有©2011 GoogleInc。保留所有权利。

目录

介绍

通用技术和领域

混淆技术

时间和时间戳

唯一标识符

用户和工作名称

资源单位

数据表

机器

机器事件

机器属性

工作和任务

作业和任务生命周期以及事件类型

缺少事件记录

作业事件表

任务事件表

任务限制

资源使用

文件格式

下载数据

异常

数据收集

缺少信息

附录

文件纪录

介绍

本文档描述了Google计算单元的语义，数据格式和使用情况跟踪的架构。本文档描述了跟踪格式的2.1版，应与跟踪提供的详细信息一起阅读。

Google集群是一组计算机，打包成机架，并通过高带宽群集网络。单元是一组机器，通常都在单个群集中，它们共享一个公用的群集管理管理系统，该系统将工作分配给各个机器。工作以作业的形式到达一个单元。作业由一个或多个任务组成，每个任务都伴随着一组资源需求，这些资源需求用于将任务调度（打包）到机器上。每个任务代表一个Linux程序，可能在单个计算机上运行，​​该程序可能包含多个进程。根据下面描述的生命周期将任务和作业安排在计算机上。任务的资源需求和使用数据是从单元管理系统和单元中各个机器提供的信息中得出的。

单个使用情况跟踪通常描述这些计算单元之一上几天的工作量。跟踪由几个数据集组成。数据集包含一个表，该表由通常包含时间戳的主键索引。每个数据集打包为一组一个或多个文件，每个文件以压缩CSV格式提供。 （有关详细信息，请参见下文。）

发布此跟踪数据的目标之一是使影响Google工作量的许多调度复杂性可见，包括各种作业类型，某些作业的复杂调度约束，混合硬件类型以及用户对资源消耗的错误估计。我们希望研究人员会发现它有用。

常用技术和领域

本节描述了出现在多个表中的字段的表示形式和语义。请注意，跟踪可能不包括此处描述的所有数据类型。

混淆技术

出于保密原因，我们混淆了跟踪中的某些信息。特别是，大多数自由文本字段已被随机散列，资源大小已线性转换（缩放），并且某些值已映射到排序序列上。我们注意以一致的方式执行此操作，以便仍可将数据用于研究研究。

这是用于大多数数据类型的混淆转换：

●不变：不会混淆值。

●哈希：通过键控加密哈希将每个值转换为不透明值。

●已排序：将生成观察值（或可能的值）的列表并进行排序。然后，此列表中的项目被分配以0开头的连续数字（即，消除了空格），并且在将它们添加到迹线之前，将观测值映射到这些数字。

●重新缩放：通过除以该测量轨迹中任何地方出现的最大值来转换每个值。例如，在所有计算机上观察到的最大内存容量用于重新调整所有内存请求和使用值。 （最大值用于确保不被零除，并且不会有任何额外的信息泄漏到结果中。）然后，将此值舍入为max（1.0 / 210，max\_value / 220）的粒度-即，不超过10位二进制精度或最大值的一百万分之一，以较大者为准。几乎所有浮点值都以这种方式转换。

●特殊：一些值以特殊方式处理。

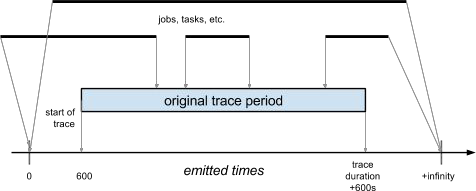
时间和时间戳

每条记录都有一个时间戳记，该时间戳记距跟踪周期开始前600秒以毫秒为单位，并记录为64位整数（即，跟踪记录开始后20秒的事件的时间戳记为620s）。

此外，还有两个特殊的时间值表示未在跟踪窗口内发生的事件：

●时间0表示在跟踪窗口开始之前发生的事件。示例：跟踪开始时存在的机器；在跟踪开始之前已提交（或已计划）的任务或作业。

●时间2^63-1（MAXINT）表示在跟踪窗口结束之后发生的事件（例如，如果在跟踪窗口结束附近发生数据收集失败）。



将原始时间映射到跟踪中发出的时间

作为例外，使用时间测量的处理方式略有不同，因为最大测量长度为300秒。我们对这些时间应用相同的开始时间偏移，就像对跟踪窗口内的事件所做的一样；这足以确保在跟踪之前的事件与其他事件之间实现清晰的分隔。注意，测量间隔可能与轨迹的开始或结束重叠。此外，使用情况测量的最佳精度是精确到秒。为了保持一致性，我们仍然以微秒为单位报告这些时间。

时间戳是近似的，尽管相当准确。它们来自多台机器，因此时间戳上的细微差异可能反映了集群中机器之间的时钟漂移，而不是任何实际的调度或利用率问题。

唯一标识符

每个作业和每台机器都分配有唯一的64位标识符。这些ID永远不会重用；但是，从群集中删除计算机并返回计算机时，计算机ID可能保持不变。极少数情况下，停止，重新配置和重新启动同一作业时，作业ID可能保持不变。 （我们在第一个跟踪中仅知道一个这样的示例。通常，重新启动作业将生成一个新的作业ID，但保持相同的作业名称和用户名。）

任务由其任务的任务ID和任务中基于0的索引来标识。具有相同作业ID和任务索引的任务可以（并且经常被停止）并重新启动，而无需分配新的作业ID或索引。

用户和工作名称

用户名和作业名经过哈希处理，并提供为不透明的base64编码字符串，可以对其进行相等性测试。

逻辑作业名称是一种作业的启发式规范化名称，该名称结合了来自多个内部名称字段的数据，并对结果进行哈希处理以构建不透明的base64编码的字符串（例如，逻辑名称中的大多数数字都被固定的字符串替换）。在生成唯一作业名称以避免冲突的情况下，由同一程序的不同执行生成的作业名称通常将具有相同的逻辑名称。 MapReduce是一个经常生成唯一作业名称的Google系统示例。

此跟踪中的用户名代表Google工程师和服务。由相同用户名运行的生产作业可能属于相同的外部或内部服务。每当单个程序运行多个作业（例如，MapReduce程序同时生成主作业和辅助作业）时，这些作业几乎总是以同一用户身份运行。

资源单位

大多数资源利用率度量和请求都已标准化，包括：

●CPU（核心计数或核心秒数/秒）

●内存（字节）

●磁盘空间（字节）

●磁盘时间分数（I / O秒/秒）

对于上述每一项，我们计算单独的归一化。标准化是相对于跟踪中任何计算机上资源最大容量的缩放比例（为1.0）。即使对于我们在此版本中未提供单个计算机容量的磁盘空间和磁盘时间分数，也是如此。

数据表

下面我们描述将要提供的表格。表键以斜体显示；请注意，某些键是从多个字段构造的。提醒：并非所有跟踪都将包含此处描述的所有数据类型。这些列可能以不同的顺序出现，或者名称与此处报告的名称不同：可以在schema.csv文件中找到此类详细信息的权威性说明。

机器

机器由两个表描述：机器事件表和机器属性表。

机器事件

每台机器由机器事件表中的一个或多个记录描述。大多数记录描述了在跟踪开始时存在的计算机。

机器事件表：

1.时间戳

2.机器ID

3.事件类型

4.平台编号

5.容量：CPU

6.容量：内存

机器事件分为三种：

●ADD（0）：一台计算机可用于群集-跟踪中的所有计算机都将发生ADD事件。

●删除（1）：一台计算机已从集群中删除。可能由于故障或维护而导致拆卸。

●更新（2）：群集可用的计算机更改了其可用资源。

机器容量反映了每个机器沿每个维度的标准化物理容量。每个维度（CPU内核，RAM大小）均独立标准化。请注意，并非所有这些功能都可用于任务。例如，集群调度程序需要为其自身和操作系统保留一些资源。

平台ID是不透明的字符串，表示计算机的微体系结构和芯片集版本。微体系结构（例如，供应商的芯片代号，如Nehalem或Bulldozer）与内存技术（例如，DDR2与DDR）的不同组合将产生不同的平台ID。具有相同平台ID的两台计算机可能具有完全不同的时钟速率，内存速度，内核数等。

机器属性

机器属性表：

1.时间戳

2.机器ID

3.属性名称：不透明的字符串

4.属性值：不透明字符串或整数

5.属性删除：一个布尔值，指示属性是否已删除

机器属性是代表机器属性的键/值对，例如内核版本，时钟速度，外部IP地址的存在。任务可以指定对计算机属性的约束。

如果该属性名称的值数量很少，则将计算机属性值提供为整数，否则提供为模糊字符串。在第一种情况下，记录并分类所有机器上观察到的属性值（如果全部为数字，则以数字顺序）。第一个值映射为1，第二个映射为2，依此类推。这种策略与先前在Google发布一些相关数据时使用的策略相同。1有些属性没有关联的值，因为它们的存在与否是信号（例如，机器是否运行GFS块服务器）；如果存在，这些属性的值将设置为1。

工作和任务

这些表描述了作业和任务：

●工作事件表

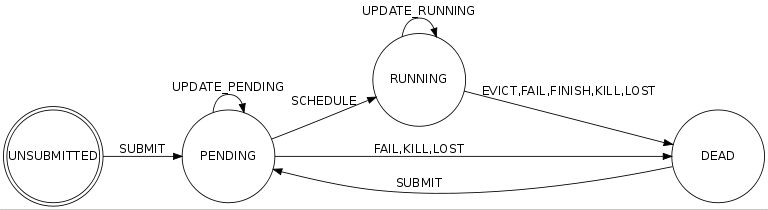
●任务事件表

●任务约束表

这两个事件表描述了作业/任务及其生命周期。约束表描述了任务放置约束，这些约束限制了可以在其上调度任务的计算机。

通常，作业中的所有任务都使用完全相同的选项和资源请求执行完全相同的二进制文件。需要运行具有不同资源需求的不同类型任务的程序通常会作为多个作业执行。例如，MapReduce在不同的作业中运行母版和工人。

作业和任务的生命周期以及事件类型



工作和任务的状态转换。

上图显示了工作或任务进行的状态简化模型。作业和任务事件指示这些状态之间的转换。

基本上有两种类型的事件：影响调度状态的事件（例如，作业已提交，或者已调度并变为可运行，或者其资源请求已更新），以及反映任务状态变化的事件（例如，任务退出）。

每个作业和任务事件都有一个代表事件类型的值。事件后作业或任务的状态始终可以通过此事件类型确定。对于工作或任务死亡，事件类型还包含有关死亡原因的信息。以下是事件（转换）代码：

●提交（0）：一个任务或作业符合计划的资格。

●时间表（1）：已在计算机上计划了作业或任务。 （由于代码运送时间等原因，它可能不会立即开始运行。）对于作业，这是第一次在计算机上计划作业的任何任务。

●EVICT（2）：由于优先级较高的任务或作业而调度了任务或作业，原因是调度程序超负荷使用且实际需求超出了计算机的容量，因为正在运行的计算机变得无法使用（例如，脱机进行维修） ），或因为保存任务数据的磁盘丢失。

●FAIL（3）：由于任务失败，对任务或作业进行了调度（或者在极少数情况下，在挂起时不再符合调度的条件）。

●结束（4）：任务或作业正常完成。

●KILL（5）：某个任务或作业被用户或驱动程序取消，或者因为该任务所依赖的另一个作业或任务死亡。

●LOST（6）：一个任务或作业可能已终止，但是在我们的源数据中缺少一条记录，表明其终止。

●UPDATE\_PENDING（7）：任务或作业的调度类，资源要求或约束在等待调度时已更新。

●UPDATE\_RUNNING（8）：计划任务或作业的计划类，资源要求或约束条件时进行了更新。

上图的顶部路径显示了最简单的情况：作业已提交，并被放入待处理队列；此后不久，它被调度到机器上并开始运行；一段时间后，它成功完成。

如果刚刚被EVICT，FAIL或KILLed的任务或作业仍可运行，则在调度事件之后将立即出现“ SUBMIT”事件。 （即，系统尝试重新启动失败的作业和任务。）尽管作业非正常终止，但有关可以重新安排任务多少次的实际策略有所不同。例如，尽管

罕见的是，某些事件可能在调度程序未给任务提供其请求的内存而引起的EVICT事件后无限期地进行调度。

对于作业或任务，跟踪可能同时包含SUBMIT和SCHEDULE事件，时间完全相同。这意味着作业或任务已提交并立即计划，或者，如果事件的时间戳为0，则作业或任务在跟踪开始之前已提交并计划了。

不幸的是，我们无法精确确定所有任务死亡的原因；在不确定的情况下，我们将其映射到KILL事件，其中还包括外部程序或开发人员采取显式操作来终止它的情况。**在任何一种情况下，都没有有关作业或任务是否正常执行的信息。**

请注意，这是我们实际使用的调度系统的简化版本，一些更复杂的功能已映射到该模型中。从资源调度研究的角度来看，我们认为在此过程中不会损失任何明显的忠诚度。

缺少事件记录

我们的数据源包含作业和任务更改的时间序列以及作业和任务状态的定期快照，因此我们可以检查它们之间的一致性。看来

我们缺少事件记录，我们合成了一个替换记录。同样，我们在跟踪时间窗口的末尾查找活动的每个作业或任务的记录，如果找不到，则合成丢失的记录。

合成记录具有一个数字（称为“缺少信息”字段），以表示为什么将它们添加到跟踪中：

●SNAPSHOT\_BUT\_NO\_TRANSITION（0）：我们找不到代表给定事件的记录，但是作业或任务状态的更高快照表示必须已经发生过渡。合成事件的时间戳是快照的时间戳。

●NO\_SNAPSHOT\_OR\_TRANSITION（1）：我们找不到代表给定终止事件的记录，但是作业或任务从群集状态的更高快照中消失了，因此它必须已终止。假设事件可能从一个快照中合理丢失，则合成事件的时间戳是其实际终止时间的悲观上限。

●EXISTS\_BUT\_NO\_CREATION（2）：我们找不到代表给定任务或作业创建的记录。在这种情况下，我们可能缺少有关作业或任务的元数据（作业名称，资源请求等），并且我们可能将SCHEDULE或SUBMIT事件放置在实际位置之后。

没有丢失数据的记录在“缺少信息”字段中没有值（空字符串）。

工作事件表

作业事件表包含以下字段：

1.时间戳

2.缺少信息

3.工作编号

4.事件类型

5.用户名

6.排课

7.工作名称

8.逻辑工作名称

作业/任务事件表包括在跟踪中的任何点处于活动状态（正在运行）或有资格运行但等待调度（正在等待）的任何作业。对于跟踪中的每个作业，我们将为其所有任务至少包含一条记录，其中将包括其调度约束。

所有作业和任务都有一个调度类，可以大致表示其对延迟的敏感性。调度类别由一个数字表示，其中3代表一个

延迟敏感型任务（例如，服务于产生收入的用户请求），0表示非生产任务（例如，开发，非关键业务分析等）。请注意，调度类不是优先级，尽管对延迟敏感的任务较多，但任务优先级较高。调度类影响机器本地的资源访问策略。优先级确定任务是否在计算机上安排。

作业名称经过哈希处理，并提供为不透明的base64编码字符串，可以对其进行相等性测试。有时，自动化系统会生成唯一的作业名称，以避免冲突。 （MapReduce是执行此操作的Google系统的示例。）

逻辑作业名称是不透明的名称，它组合了来自多个内部名称字段的数据（例如，逻辑名称中的大多数数字都用固定的字符串替换）。逻辑作业名称会部分补偿自动工具生成的唯一作业名称；在这种情况下，同一程序的不同执行通常将具有相同的逻辑名称。

任务事件表

任务事件表包含以下字段：

1.时间戳

2.缺少信息

3.工作编号

4.任务索引-工作中

5.机器ID

6.事件类型

7.用户名

8.排课

9.优先

10. CPU内核的资源请求

11. RAM的资源请求

12.对本地磁盘空间的资源请求

13.不同机器的约束

machineID（如果存在）指示计划任务的计算机。每个任务都有一个优先级，它是一个小的整数，在此处被映射为一组排序的值，

最低优先级为0（最不重要）。具有较高优先级数字的任务通常比具有较低优先级数字的任务更喜欢资源。

有一些特殊的优先级范围：

●“免费”优先级：这些是最低优先级。以这些优先级请求的资源几乎不会产生内部费用。

●“生产”优先级：这些是最高优先级。集群调度程序试图防止由于机器资源的过度分配而驱逐这些优先级上对延迟敏感的任务。

●“监视”优先级：这些优先级适用于监视其他优先级较低的工作的运行状况

我们可能会在文档中指出哪些优先级范围代表哪些类别

每条痕迹；由于我们规范了提供的优先级，因此这些版本之间的范围可能会有所不同。

如果存在“不同机器”约束字段，并且为true，则表明必须安排任务与该作业中任何其他当前正在运行的任务一起在不同机器上执行。这是一种特殊的约束。

资源请求代表允许任务使用的最大CPU，内存或磁盘空间量（我们称其为限制）。超出限制的任务可能被限制（对于CPU等资源）或被杀死（对于内存等资源）。请注意，调度程序可能会过量使用计算机上的资源。结果，可能发生的情况是，没有足够的资源来满足任务中的所有运行时请求，即使每个任务使用的资源少于其限制。如果发生这种情况，可能会杀死一个或多个低优先级任务。

此外，运行时环境有时允许任务使用超出其请求的范围。例如，允许任务使用计算机上的可用CPU容量，因此具有短暂的对延迟不敏感的CPU突发的任务可能会在请求的CPU分配为0时有用地运行。

此跟踪包括一些支持Google分布式文件系统的作业，但这些作业不通过集群调度程序请求文件系统的永久存储的磁盘空间-它们依赖于外部分配​​机制。这些资源请求和使用情况不包含在跟踪数据中。

任务约束

一个任务可能具有零个或多个任务放置约束，这限制了可以在其上运行任务的计算机。每个约束记录仅对应一个任务事件记录。 （如果仅通过更新机器约束来更新任务，那么将有一个任务事件记录指示更新。）单个属性可能有多个约束，例如，一个属性可能同时具有小于和大于单个任务的约束。

任务约束表包含以下字段：

1.时间戳

2.工作编号

3.任务索引

4.属性名称-对应于机器属性表

5.属性值-不透明字符串或整数或空字符串

6.比较运算符

比较运算符为：

●小于（2），大于（3）：将机器属性表示为整数（如果不存在该属性，则为0），然后将其与提供的属性值进行比较。这些比较严格小于和严格大于；

●等于（0），不等于（1）：将机器属性表示为字符串（如果不存在，则表示为空字符串），然后将其与提供的属性值进行比较；

资源使用

我们的集群使用Linux容器进行资源隔离和使用情况统计。每个任务都在其自己的容器中运行，并且可以在该容器中创建多个进程。

我们报告每个测量周期的使用值，通常为5分钟（300秒），尽管某些测量记录的时间较短，这通常是因为任务在该时间段内进行了更新。当任务被杀死时，测量时间段可能会比任务终止时延长长达数十秒，并且一些测量记录会在任务终止后的几分钟内延续。

在每个测量周期内，通常以1秒的间隔进行测量，尽管系统负载有时会阻止以所需的速率对资源信息进行采样，因此测量可能会超过1秒。为了帮助识别这些情况，我们提供了样本部分（又称为采样率），即预期样本数（例如300 s测量窗口中为300）与观察样本数之比。

在每个测量周期的长度上聚合1秒样本，以提供测量周期的平均值以及该周期内任何1秒样本中看到的最大值。

在某些情况下，容器的总价值是根据从多个子容器中获得的测量值构建的：尽管我们知道每个子容器的最大使用量，但我们的测量系统可能无法报告外部容器的真实最大值。在这种情况下，我们报告的最大值是子容器中最大值的总和，以这种方式聚合的最大使用量的记录将以聚合类型标记1进行标记；所有其他记录的聚合类型将为0。此外，即使平均使用量度包括其使用量，我们也可能没有所有子容器的最大值（例如，如果它们很小和/或寿命很短） 。

**测量记录可能会丢失。缺少记录并不一定表示任务没有运行。**

当任务的容器中没有属于该任务的进程正在运行时，会有一段时间的任务使用情况度量。例如，将二进制文件复制到计算机时可能会发生这些测量。在这种情况下，任务的内存，CPU和磁盘使用率可能合法为0。在某些情况下，任务可能长时间没有任何进程运行。

任务资源使用情况表包含以下字段：

1.测量周期的开始时间

2.测量期的结束时间

3.工作编号

4.任务索引

5.机器ID

6.平均CPU使用率

7.规范的内存使用

8.分配的内存使用情况

9.未映射的页面缓存的内存使用情况

10.页面缓存总内存使用量

11.最大内存使用

12.平均磁盘I / O时间

13.平均使用的本地磁盘空间

14.最大CPU使用率

15.最大磁盘IO时间

16.每条指令周期（CPI）

17.每个指令的内存访问（MAI）

18.样品部分

19.聚合类型（如果将子容器的最大值相加，则为1）

20.采样的CPU使用率：在测量周期内随机1s采样期间的平均CPU使用率（仅在v2.1及更高版本中）

CPU使用率（也称为CPU速率）以每秒CPU核心秒数为单位进行度量：如果一个任务一直使用两个核心，则将其反映为2.0核心/秒的使用率。

●仅在2.1版和更高版本中提供了“ CPU使用率-采样”字段。从一秒钟的平均CPU使用率是从用于该行中其他数据的5分钟的测量周期中随机抽取的。对于长时间运行的任务，可以使用此数据来构建合理准确的CPU使用率随机模型。

由于内存隔离是通过Linux容器实现的，因此代表该任务的某些内核内存使用情况将计入该任务。任务必须请求足够的内存以包括此类分配。包括以下内存使用情况数据：

●内存使用情况：规范的内存使用情况度量；用户可访问页面的数量，包括页面缓存，但不包括标记为陈旧的某些页面。

●未映射的页面缓存内存：Linux页面缓存（文件支持的内存）未映射到任何用户空间进程中；

●页面缓存内存：Linux页面缓存总数（文件支持的内存）

●分配的内存：基于实际分配给容器的内存使用量（但不一定使用）

●最大内存使用量：在测量间隔内观察到的规范内存使用量测量的最大值。该值不适用于某些任务。

使用blkio子系统（用于Linux容器）测量磁盘I / O时间。使用率度量是计算机上所有磁盘的总和，以每秒磁盘时间秒为单位。

记录在此跟踪中的磁盘空间表示运行时本地磁盘容量使用情况。不包括二进制文件和其他只读的，预转换的运行时文件所需的磁盘使用情况。

此外，此跟踪未考虑分布式持久存储（例如GFS，Colossus）使用的大多数磁盘空间。

CPI和MAI统计信息是从处理器性能计数器中收集的；并非所有机器都收集此数据。内存访问基于对最后一级缓存未命中的测量。

文件格式

跟踪是一组数据集，每个数据集对应一个上述的不同表。每个数据集由一个或多个文件组成，每个文件都使用gzip压缩。

每个文件都是CSV格式，使用Unix样式的行尾（ASCII LF）。 CSV文件没有标题。行按事件时间戳排序，该时间戳始终存储在第一列中。

所使用的文件名与以下模式匹配（?????表示填充零的五位数字，例如00123）。

自述文件MD5SUM

SHA1SUM SHA256SUM

schema.csv

job\_events / part-?????-of-??????。csv.gz

task\_events / part-?????-of-?????。csv.gz

machine\_events / part-?????-of-?????。csv.gz

machine\_attributes / part-?????-of-??????。csv.gz

task\_constraints / part-?????-of-?????。csv.gz

task\_usage / part-?????-of-??????。csv.gz

字段格式如下：

1.如果一个字段没有值，则该字段将为空（无引号）。

2.数字字段表示为十进制整数（用于时间戳，ID，事件类型等）或浮点值（用于资源利用编号），可以使用C printf格式“％g”打印出来。

3.布尔字段表示为0（假）或1（真）。

4.字符串字段是SHA256哈希的base-64编码值，因此具有恒定的长度。

所有字段均已标准化，因此不包含逗号，换行符或引号。这意味着CSV文件的每一行都可以通过用逗号分割来解析。

大表分为几个连续的文件，按时间顺序编号。 CSV记录永远不会在文件中拆分，但是特定作业，任务或机器的记录可能会在文件中拆分。相同的时间戳范围将用于不同数据集中具有相同文件号的所有文件（例如，task\_events-01234-of-05678.csv.gz将包含与machine\_events-01234-of-05678.csv相同的时间戳范围.gz）。

下载数据

数据存储在Google Storage for Developers中。每个跟踪使用其自己的存储桶。 （以下示例为例，我们将使用2011年的v2.1跟踪，称为clusterdata-2011-2。）您无需具有Google帐户或注册Google Storage即可下载数据。

每个跟踪中都包含一个SHA256SUM文件，可以使用来自GNU coreutils的sha256sum命令（例如，使用以下命令）来验证下载的完整性

sha256sum –check SHA256SUM

有三种下载数据的方法：使用GSUtil实用程序（这是推荐的方法）；通过boto库，或仅使用HTTP。请注意，整个跟踪可能会适度大（几十GB）。有关更多信息，请参见单个跟踪描述。

命令行：GSUtil

首先，下载GSUtil软件。然后您使用类似的命令

gsutil ls gs：// clusterdata-2011-2 /

列出存储桶的内容和类似的命令

gsutil cp -R gs：// clusterdata-2011-2 /目标目录

将存储桶的内容复制到本地磁盘。您还可以使用目录名称或模式匹配（例如，

“ gs：// clusterdata-2011-2 / task\_usage / \*”。如果连接快速，使用-m选项并行下载多个文件可能会有所帮助。如果您的下载被中断，如果您尝试在部分传输的文件上进行下载，则gsutil将恢复该下载。

以编程方式：boto库

您可以使用boto库以编程方式从Google存储空间下载数据。

网址：HTTP

此外，Google存储空间中的每个文件都可以通过HTTP使用。例如，https://commondatastorage.googleapis.com/clusterdata-2011-2/SHA256SUM是第一条跟踪的SHA256SUM文件的URL。不幸的是，没有目录索引。

异常现象

本节列出了您应该了解的一些怪异之处。

数据采集

此数据主要来自监视数据，该监视数据是通过定期远程过程调用（RPC）收集的。**当监视系统或群集过载时，可能不会收集数据。**此外，我们特意删除了一些使用情况数据，这些数据似乎由于监视系统中的错误而被贴错标签。对于调度程序事件，我们已经合成记录以使数据如上所述一致，但是跟踪中仍然可能会丢失一些数据。

请参阅跟踪附带的详细说明，以了解缺少多少事件记录和资源使用情况度量。

丢失的信息

我们的集群调度程序支持跟踪中未表示的一些功能。这些包括：

●作业之间的依存关系：用户可以提交作业，以指定他们只能在另一个作业完成之前才能运行，或者只能在另一个作业正在运行时才能运行。当满足任何此类作业依赖关系时，我们的跟踪就会显示提交的作业。永远不会满足其依赖关系的作业不存在。由于相关作业死亡而杀死的作业或任务显示为“杀死”事件。

●其他资源：一些其他资源类型及其用法。

●其他约束：一些非资源调度约束（例如，限制可以在同一机架上的作业中的任务数量）； “首选”约束，任务将请求在具有特定属性的计算机上运行，​​但仍愿意在其他地方进行调度。

●不受调度程序管理的工作负载：任务使用情况和计算机容量度量不考虑计算机上的某些工作负载，因为某些系统在群集调度程序的控制范围之外运行。

附录

文件纪录

日期 备注

2014-11-17添加了有关cluster-2011-2的信息。

2014-09-02版本2.1：在任务使用表中添加了“ CPU使用率（aka速率）-随机1s示例”

2013.05.06增加页码突出显示的文本，指出schema.csv是列的权威参考；解释说CPU使用率和CPU速率是同一回事。

2012.03.20添加了有关由schema.csv文件定义的列顺序的说明；修复了任务约束表以匹配数据。

2011.11.08添加了有关gsutil“最大打开文件数”限制的说明。

2011.10.27首次对外发布版本。