借助 IBM Cloud 提高大数据分析速度并提升响应性

相比 AWS 解决方案，IBM Cloud 配置能在更短的时间内完成大数据分析工作负载，同时吞吐量也更高

云计算提供弹性和灵活性，能够满足大数据相关项目的各种需求。尽管并非所有基础架构即服务 (IaaS) 提供商都一模一样，但是贵企业都需要通过性能和速度的优势来保证大数据项目成功实施。

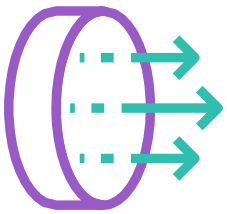
Principled Technologies 针对两家 IaaS 提供商，执行了基于 Hadoop 的大数据工作负载测试和一系列网络测试：IBM® Cloud 和 Amazon® Web Services (AWS®) Elastic Compute Cloud® (EC2®)。相比 AWS 解决方案，IBM Cloud 解决方案在处理大数据时，能够更快地将数据点聚合到虚拟机 (VM) 上，同时吞吐量也更高。相比 AWS，IBM Cloud 提供更短的大数据分析等待时间，快速提供数据支撑，以加速您企业的市场营销活动、应用或运营改进。

IBM Cloud 解决方案也能在数据中心之间更快地传输数据和发送 ping 值。如果您能在位于美国、英国和日本的数据中心之间更快地来回传输数据，这意味着，位于全球各地的员工能够更快地进行交流、计划和协作。

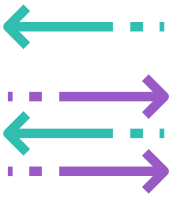
贵企业的 IaaS 提供商将影响您的大数据计划和全球通信。在大数据的性能和速度方面，我们的调查结果显示，相比 AWS 解决方案，IBM Cloud 解决方案能为贵企业提供更多收益。

IBM Cloud 解决方案能帮助您更快地从大数据中挖掘宝贵的洞察力，同时实现：

数据聚类分析的时间缩短 28%



每秒处理聚类分析的数据量增加 39%



数据传输时间最高缩短 3 倍

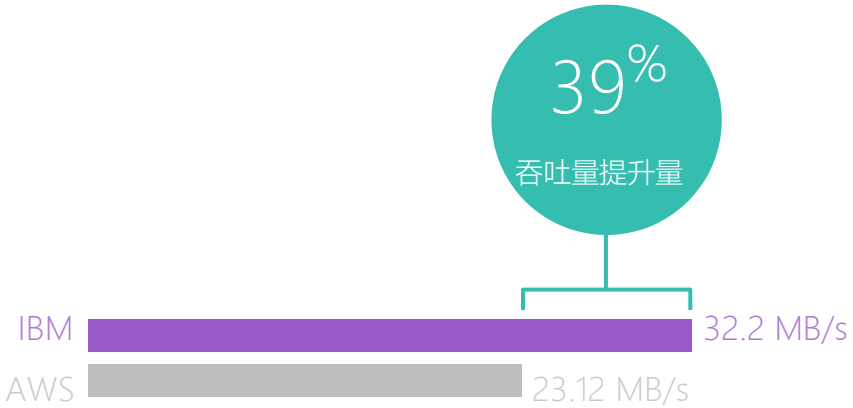
选择云为大数据实施带来灵活性和可扩展性

大数据分析能收集各种设备和用户生成的不断增长的数据集，并加以排序、分类和隔离，进而帮助您优化决策流程，协助改进客户服务和产品开发。这一切需要基础架构，来帮助数据分析师、开发人员和其他 IT 员工无缝扩展自己的工作，跟上数据的增长速度。云不仅能提供这种灵活性和可扩展性，还能提供其它优势，包括：

- 在预订流程中，除了选择配置外，设置工作很少。
- IT 管理员无需维护其它硬件。
- 如果企业无法承担或不想要其它内部计算资源，那么他们只需利用大数据分析技术，不需要投入硬件资本支出 (CapEx) 和运营支出 (OpEx)，如电力、冷却和管理等支出。
- 如果企业仅需要短期运行或仅在每隔一定的时间运行大数据分析，那么他们往往只需要为使用云的时间段付费。

快速处理更多数据

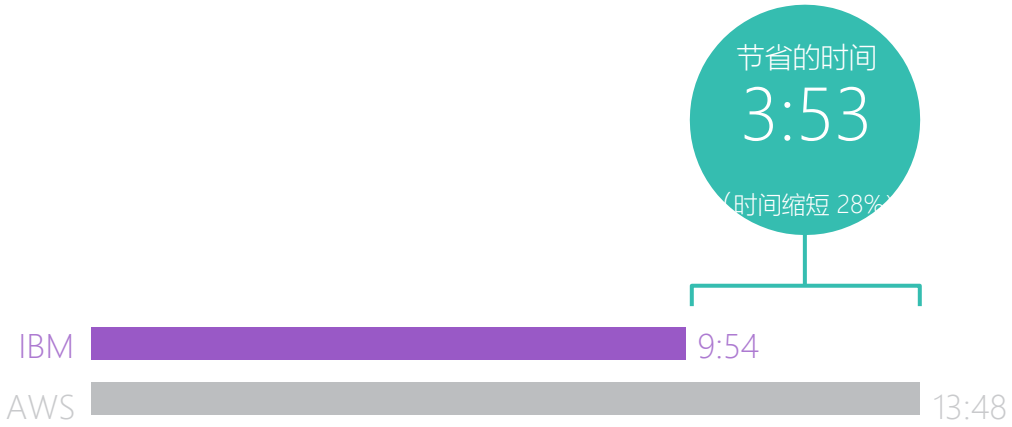
我们运行了大数据基准测试 HiBench 套件中的 Kmeans 工作负载。Kmeans 是一个机器学习基准测试工具，它能够聚集数据点，并报告工作负载的吞吐量和持续时间。当我们查看测试输出结果时发现，IBM Cloud 解决方案的吞吐量比 AWS 高 39%（参见附录 B，进一步了解我们如何开展测试）。如果您的虚拟化解决方案能提供极高的吞吐量，您可能需要较少的虚拟机即可处理大数据工作负载，同时还有可能减少获取分析技术所需的运营支出，快速处理大型数据集。



如果每秒能够分析更多数据，那么分析师就能处理更大规模的数据集，并且不需要大幅延长工作负载的持续时间。如果您需要分析的数据集不断增加，那么您可以扩展 IBM Cloud 解决方案，以便处理更大规模的大数据工作负载。

更快地聚合数据用于分析

不论您是希望开展有效的市场营销活动，还是尝试提高仓库的运营效率，解决方案越早将数据点进行聚类，数据分析师就能越早挖掘宝贵的洞察力和模式。在聚合 18.7GB 的数据用于 Kmeans 测试时，相比 AWS 解决方案，IBM Cloud 解决方案节省了 3 分钟 53 秒（用时缩短 28%）。如需进一步了解我们如何开展测试，请参见附录 B。



通过从数据集中（即使是小型数据集中）快速获取洞察力和模式，您能够将一次性客户转化成回头客，否则您将错失能为您创收的客户群。大数据分析对很多行业来说都很有用，并不只限于零售行业。即使是在医疗保健行业，节省时间也意味着医生能够正确诊断患者的病情或者改用另一种治疗方案。从运营的角度来看，云提供的速度能帮助贵企业即时完成工作负载，避免在虚拟机上堆积工作负载。

根据您的工作负载，选择 IBM Cloud 服务

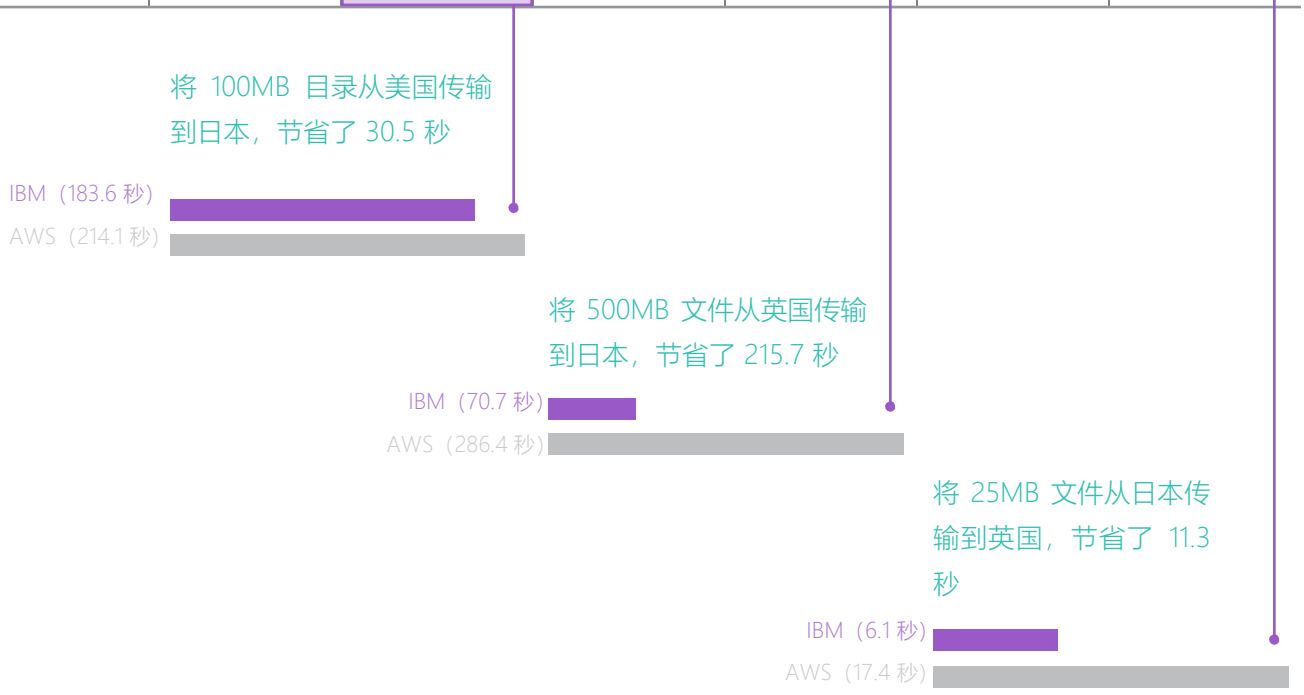
有些 IaaS 和云服务提供商只提供有限的服务，开发人员和其他 IT 员工只能考虑围绕这些服务开展工作。IBM Cloud 提供了 170 多项服务，包括裸机服务、公有云服务和容器服务，我们测试的 IBM 解决方案就包含在这些服务中。为了在成本和工作负载的使用之间找到一个平衡点，IT 部门和财务部门必须在按月定价和按小时定价之间选择一种定价模式。而且，IBM Cloud 在全球 19 个国家或地区有 60 个当地拥有并运营的数据中心。因此，管理员可以选择在特定的地区运行应用和工作负载，确保安全性与合规性。此外，IT 管理员还可以从多种 Intel® Xeon® 处理器中选择最适合特定工作负载的计算资源。有关更多信息，请访问：https://www.ibm.com/cloud/。

在不同位置的数据中心之间更快地传输数据

对于业务遍及全球各地的企业来说，如果能快速传输数据，他们的管理人员和高管就能及时制定明智的决策。我们在美国、日本和英国的数据中心之间传输数据，而 IBM Cloud 能比 AWS 更快地传输数据。在传输 500MB 的文件时，我们发现 IBM Cloud 解决方案节省的时间最多。将数据从英国的数据中心传输至美国的数据中心时，IBM Cloud 解决方案比 AWS 节省时间将近两分钟。将 500MB 的文件从英国的数据中心传输至日本的数据中心时，IBM Cloud 比 AWS 节省时间超过三分半钟。将数据从美国的数据中心传输至日本的数据中心时，IBM Cloud 解决方案比 AWS 节省时间超过两分半钟。下表比较了两款解决方案的用时。如需进一步了解我们如何开展测试，请参见附录 B。

文件和目录传输时间（秒）

25MB 文件						
	美国到英国	美国到日本	英国到美国	英国到日本	日本到美国	日本到英国
IBM Cloud	2.5	4.1	2.5	6.2	4.1	6.1
AWS	5.8	11.4	6.1	16.7	11.3	17.4
500MB 文件						
	美国到英国	美国到日本	英国到美国	英国到日本	日本到美国	日本到英国
IBM Cloud	24.7	46.2	24.5	70.7	46.2	70.9
AWS	104.0	208.6	143.3	286.4	199.5	281.4
100MB 目录						
	美国到英国	美国到日本	英国到美国	英国到日本	日本到美国	日本到英国
IBM Cloud	99.5	183.6	98.9	280.1	183.2	281.7
AWS	106.2	214.1	112.1	292.8	188.6	293



通过加速传输数据，跨国企业能够减少出现运营瓶颈的几率，提高生产力。比如，在美国、英国和日本都设有办事处的金融机构能够全天候 (24/7) 响应市场和客户活动。

除了文本传输外，我们还能在数据中心之间发送 ping 值。在 6 个 ping 测试中，IBM Cloud 解决方案赢得了其中 5 个测试。两款解决方案的最大区别在于，当我们在位于美国和日本的数据中心之间发送 ping 值时，IBM Cloud 解决方案节省了近 18 毫秒。发送 ping 值的速度更快意味着，数据中心之间的响应性更高。下面的图表比较了利用两款解决方案在测试的数据中心之间发送 ping 值的情况。

Ping 测试（毫秒）

	美国到英国	美国到日本	英国到美国	英国到日本	日本到美国	日本到英国
IBM Cloud	73.9	136.7	73.9	210.9	136.7	210.8
AWS	77.8	154.7	71.9	214.7	147.5	214.3

结论

大数据分析能够大幅改进企业运营。通过快速集群并分隔可用数据，分析师和决策者能够近乎实时地了解用户需求、部门如何有效运营，或者轰炸式发送电子邮件的最佳时机。我们对两款解决方案执行了基于 Hadoop 的机器学习工作负载和网络测试，结果发现相比 AWS 解决方案，IBM Cloud 解决方案能够更快地聚合数据，更快地传输数据，并且吞吐量也更高。如果贵企业需要通过一款 IaaS 平台来从大数据分析中受益，在位于美国、英国和日本的数据中心之间快速传输数据，那么 IBM Cloud 解决方案可能是您的理想选择。

1 “Bluemix is now IBM Cloud: Build confidently with 170+ services”. 2018 年 1 月 18 日. https://www.ibm.com/blogs/bluemix/2017/10/bluemix-is-now-ibm-cloud/

2017 年 8 月 4 日，我们最终确定了测试的硬件和软件配置。当前及近日发布的硬件和软件经常发布更新。因此，这些配置可能无法代表本报告中出现的最新软硬件版本，这种情况不可避免。对于更旧的系统，我们选择了大多数客户购买的系统配置。2017 年 8 月 13 日，我们完成了亲身实践测试。

附录 A：系统配置信息

服务器配置信息	AWS 服务器 - Xen HVM domU
BIOS 名称和版本	Xen 4.2
操作系统名称与版本/构建号	Ubuntu 16.04.3 LTS
最近一次 OS 更新/补丁修复日期	2017 年 7 月 20 日
处理器	
处理器数量	1
供应商和型号	Intel Xeon 处理器 E5-2676 v3
每个处理器的内核数	40
内核频率 (GHz)	2.40
步进	2
内存模块	
系统总内存 (GB)	160
内存模块数量	10
内存大小 (GB)	16
存储控制器	
供应商和型号	Intel 82371SB PIIX3 IDE
本地存储	
驱动器 #1 容量 (GB)	100
驱动器 #2 容量 (GB)	500
网络适配器	
供应商和型号	Intel 82599 Ethernet Controller

服务器配置信息	IBM Cloud server
操作系统名称与版本/构建号	Ubuntu 16.04.3 LTS
最近一次 OS 更新/补丁修复日期	2017 年 7 月 20 日
处理器	
处理器数量	1
供应商和型号	Intel Xeon 处理器 E5-2683 v4
每个处理器的内核数	32
内核频率 (GHz)	2.10
步进	1
内存模块	
系统总内存 (GB)	128
存储控制器	
供应商和型号	Intel 82371SB PIIX3 IDE
本地存储	
驱动器 #1 容量 (GB)	350
驱动器 #2 容量 (GB)	500

附录 B：测试方法

对比场景设计

对于本次分析的对比研究部分，我们分别在同等配置的 IBM Cloud 和 AWS 中测量了 CPU 密集型工作负载的直接、真实性能。测量时，我们使用了来自 Intel HiBench Hadop 基准测量套件中的 K-means Clustering 工具。我们之所以选择该工具，其目的在于凸显两个云解决方案在平均计算吞吐量方面的差异，因此并没有选择综合性的基准测量工具，因为后者只能在较窄的任务范围内测量 CPU 性能，所以测量结果的客观性稍差。

我们通过配置使这两个云解决方案达到类似的性能，以便公正地评估它们各自的计算吞吐量。由于供应商的配置产品存在局限性，因此，我们无法做到两者的配置完全一致。以下各节汇总了通过测量得出的两款云平台的共性和差异。

通用配置

在每个云平台中，节点无外乎扮演四种角色，分别是客户端节点（基准测量机器）、配置节点（配置管理节点）、基础架构节点（Hadoop 名称节点，又称作主节点），以及从节点（Hadoop 数据节点，即受测系统）。我们构建了具有 4 个内核、16 GiB 内存的虚拟机，以满足客户端节点、配置节点及基础架构节点的需求。对于每个虚拟机，我们分配了足够的存储容量，以容纳基准测量基础架构和 Hadoop 基础架构。

在 ESXi 中创建第一个虚拟机

所有的从节点均使用 Intel Xeon 处理器 E5-2600 v3（虚拟化处理器）。

IBM Cloud

- 每个从节点均按照 32 个线程（1 个 CPU，每个 CPU 32 个内核，每个内核 1 个线程）、128 GiB RAM 进行配置。
- 该款云解决方案配置了 4 个节点，共 128 个线程，总内存是 512 GiB。

AWS

- 每个从节点均按照 40 个线程（1 个 CPU，每个 CPU 40 个内核，每个内核 1 个线程）、160 GiB RAM 进行配置。
- 该款云解决方案配置了 3 个节点，共 120 个线程，总内存是 480 GiB。

网络

所有节点都尽可能地共置，尽可能地接近供应商的允许配备选项，同时确保最快的网络速度。对于 IBM Cloud 解决方案，所有的节点均部署在 Dallas 06 中，虚拟机采用 1 Gbps 的网络。对于 AWS 解决方案，我们指定了一个通用放置组，用于共置所有节点，而且为每个节点分配了 10Gbps 的网络。

存储

对于客户端节点、配置节点和基础架构节点，我们为其配备了 25GB 的普通存储主驱动器。IBM Cloud 解决方案采用的是基于存储局域网 (SAN) 的驱动器；AWS 解决方案采用的则是 EBS，具体来说就是通用 SSD (GP2)。

对于从节点，我们为每个节点都配备了一个大容量的主驱动器（操作系统驱动器），还另外配备了一个大容量驱动器，用于支持 Hadoop HDFS。IBM Cloud 虚拟节点的驱动器采用的是 SAN 驱动器（操作系统驱动器的容量是 100 GB， HDFS 驱动器的容量是 500 GB）。对于 AWS 解决方案的从节点，操作系统驱动器的容量是 100 GB，HDFS 驱动器的容量是 500 GB，均部署在 3,000 IOPS 的 IO1 存储上。

方法

这两款解决方案的测试步骤如下所述：

1. 使用 Linux Ubuntu 16.04 操作系统配备虚拟机。
2. 在基础架构节点和从节点上，使用 Ambari 安装并配置 Hadoop。
3. 在客户端节点上，安装并配置 Intel HiBench。
4. 在客户端节点上，安装并配置 IBM LPCPU 监控实用程序。
5. 准备 HiBench 工作负载。
6. 对从节点进行测试前调整。
7. 在从节点上，启动 LPCPU 监控。
8. 运行 HiBench Benchmark 并捕获结果。
9. 在从节点上，停止 LPCPU 监控并捕获结果。
10. 每次重复测试时，重复操作步骤 5 至步骤 9。

使用 Linux Ubuntu 16.04 配备虚拟机

不同供应商配备机制稍有差异，但都遵循着同一通用范例：

1. 登录到 Web 门户。
2. 点击设备或实例。
3. 选择以下各项的值：
   - 数量
   - 区域或放置组
   - 操作系统
   - CPU
   - 内存
   - 网络
   - 磁盘
4. 确认预订信息，然后等待虚拟机启动。
5. 使用 SSH 登录到虚拟机，并配置支持 Hadoop、Ambari、HiBench、LPCPU 所需的软件条件，或配置任何配置管理软件。
6. 对于集群中的每个主机，执行以下步骤：
   a. 连接至主机 <XXX>：
      > ssh root@XXX
   b. 为根用户创建一个 RSA 密钥对：
      > ssh-keygen -t rsa -b 8192 -N "" -f ~/.ssh/id_rsa
   c. 对于集群中的每个其他主机 YYY，将 root@XXX 的 SSH 密钥复制到 YYY：
      > ssh-copy-id -i ~/.ssh/id_rsa.pub root@YYY
   d. 对集群中的每个主机重复步骤 6c。
7. 对集群中的每个主机重复步骤 6。

安装并配置 Hadoop

1. 在所有节点上，安装 Ambari REPO 和密钥：
   > wget -O /etc/apt/sources.list.d/ambari.list http://public-repo-1.hortonworks.com/ambari/ubuntu16/2.x/updates/2.5.1.0/ambari.list
   > apt-key adv --recv-keys --keyserver keyserver.ubuntu.com B9733A7A07513CAD
2. 更新软件包缓存并进行升级：
   > apt-get update -y
   > apt-get upgrade -y
3. 在配置节点上，安装 Ambari Server：
   > apt-get install -y ambari-server
   > ambari-server setup -j PATH_TO_JAVA_HOME -s
   > ambari-server restart
4. 在其他节点上，安装 Ambari Agent：
   > apt-get install -y ambari-agent
   > nano /etc/ambari-agent/conf/ambari-agent.ini
      set 'hostname' to the IP/FQDN of the 'config' node
   > ambari-agent restart
5. 在配置节点上，按照所示值计算或设置以下加粗的变量：
   - 集群中的从节点数量：
     NUMBER_OF_SLAVES = X
   - 任何节点的最小内存 (MiB)：
     MIN_MEM = 131072
   - 操作系统预留内存（如 28 GB）：
     OS_RESERVE_MB = 31072
   - 每个节点的线程数：
     NUM_CORES = num_cpus * cores/cpu * threads/core
   - 计算允许的内核数：
     YARN_NUM_CORES = NUM_CORES - 2
   - 允许的 Yarn 内存：
     YARN_MEMORY_MB = MIN_MEM - OS_RESERVE_MB = 100000

- Yarn 内存分配器：
  YARN_MIN_ALLOC_MB = 8192
  YARN_MAX_ALLOC_MB = 32768
  YARN_INCREMENT_MB = 512
- Yarn vcpu 分配器：
  YARN_MIN_CORES = 1
  YARN_MAX_CORES = 30
  YARN_INCREMENT_CORES = 1
- MapReduce 内存：
  MAPRED_MAP_MEMORY_MB = 4096
  MAPRED_REDUCE_MEMORY_MB = 8192

a. 创建"主机映射"，用于将主机映射到 Ambari 中的 Hadoop 角色：

```
> nano ~/ambari-hostmap.json
 {
    "blueprint" :                "my_hadoop_cloud",
    "default_password" :"top-secret", "host_groups" :[
       { "name" :"master1", "hosts" :[ { "fqdn" :"master1.my_cloud.org" } ] },
       { "name" :"master2", "hosts" :[ { "fqdn" :"master2.my_cloud.org" } ] },
       { "name" :"master3", "hosts" :[ { "fqdn" :"master3.my_cloud.org" } ] },
       { "name" :"clients", "hosts" :[ { "fqdn" :"client.my_cloud.org" } ] },
       { "name" :"slaves", "hosts" :[
             { "fqdn" :"slave1.my_cloud.org" },
             { "fqdn" :"slave2.my_cloud.org" },
             { "fqdn" :"slave3.my_cloud.org" },
             ...
             { "fqdn" :"slaveN.my_cloud.org" }
          ]
       }
    ]
 }
```

b. 创建 Ambari 蓝图：

```
> nano ~/ambari-blueprint.json
        "configurations" :[
       { "hive-site" :{
             "javax.jdo.option.ConnectionUserName" :"$HIVE_USER",
             "javax.jdo.option.ConnectionPassword" :"$HIVE_PASS" } },
       { "hdfs-site" :{
             "dfs.datanode.data.dir" :"$DISK_1_MOUNTPOINT, $DISK_2_MOUNTPOINT, ...$DISK_N_MOUNTPOINT" } } },
       { "yarn-site" :{ "properties" :{
             "yarn.nodemanager.resource.cpu-vcores"          :$YARN_NUM_CORES,
             "yarn.nodemanager.resource.memory-mb"           :$YARN_MEMORY_MB,
             "yarn.scheduler.minimum-allocation-mb"          :$YARN_MIN_ALLOC_MB,
             "yarn.scheduler.maximum-allocation-mb"          :$YARN_MAX_ALLOC_MB,
             "yarn.scheduler.increment-allocation-mb"        :$YARN_INCREMENT_MB,
             "yarn.scheduler.minimum-allocation-vcores"      :$YARN_MIN_CORES,
             "yarn.scheduler.maximum-allocation-vcores"      :$YARN_MAX_CORES,
             "yarn.scheduler.increment-allocation-vcores" :$YARN_INCREMENT_CORES } } },
       { "mapred-site" :{ "properties" :{
             "mapreduce.map.memory.mb"          :$MAPRED_MAP_MEMORY_MB,
             "mapreduce.reduce.memory.mb"       :$MAPRED_REDUCE_MEMORY_MB } } }
    ],
    "host_groups" :[
       { "components" :[
             { "name" :"SECONDARY_NAMENODE"},
             { "name" :"HST_AGENT"},
             { "name" :"SLIDER"},
             { "name" :"SPARK_CLIENT"},
             { "name" :"HDFS_CLIENT"},
             { "name" :"ZOOKEEPER_SERVER"},
```

```
        { "name" :"HISTORYSERVER"},
        { "name" :"METRICS_MONITOR"},
        { "name" :"TEZ_CLIENT"},
        { "name" :"APP_TIMELINE_SERVER"},
        { "name" :"RESOURCEMANAGER"}
    ],
    "configurations" :[ ],
    "name" :"master2",
    "cardinality" :"1" },
{ "components" :[
        { "name" :"SPARK_CLIENT" },
        { "name" :"YARN_CLIENT" },
        { "name" :"HDFS_CLIENT" },
        { "name" :"HST_SERVER" },
        { "name" :"STORM_UI_SERVER" },
        { "name" :"METRICS_MONITOR" },
        { "name" :"INFRA_SOLR_CLIENT" },
        { "name" :"NAMENODE" },
        { "name" :"TEZ_CLIENT" },
        { "name" :"ZEPPELIN_MASTER" },
        { "name" :"NIMBUS" },
        { "name" :"KNOX_GATEWAY" },
        { "name" :"SPARK2_JOBHISTORYSERVER" },
        { "name" :"ACTIVITY_ANALYZER" },
        { "name" :"KAFKA_BROKER" },
        { "name" :"HST_AGENT" },
        { "name" :"MAPREDUCE2_CLIENT" },
        { "name" :"ZOOKEEPER_SERVER" },
        { "name" :"INFRA_SOLR" },
        { "name" :"SPARK_JOBHISTORYSERVER" },
        { "name" :"DRPC_SERVER" },
        { "name" :"METRICS_GRAFANA" }
    ],
    "configurations" :[ ],
    "name" :"master1",
    "cardinality" :"1" },
{ "components" :[
        {"name" :"MAHOUT" },
        { "name" :"SPARK_CLIENT" },
        { "name" :"YARN_CLIENT" },
        { "name" :"HDFS_CLIENT" },
        { "name" :"SPARK2_CLIENT" },
        { "name" :"METRICS_MONITOR" },
        { "name" :"INFRA_SOLR_CLIENT" },
        { "name" :"TEZ_CLIENT" },
        { "name" :"ZOOKEEPER_CLIENT" },
        { "name" :"HCAT" },
        { "name" :"PIG" },
        { "name" :"HST_AGENT" },
        { "name" :"MAPREDUCE2_CLIENT" },
        { "name" :"SLIDER" },
        { "name" :"HIVE_CLIENT" }
    ],
    "configurations" :[ ], "name" :"clients",
    "cardinality" :"1"
},
{ "components" :[
        { "name" :"YARN_CLIENT" },
        { "name" :"HDFS_CLIENT" },
        { "name" :"HIVE_SERVER" },
        { "name" :"MYSQL_SERVER" },
        { "name" :"METRICS_MONITOR" },
        { "name" :"HIVE_METASTORE" },
        { "name" :"TEZ_CLIENT" },
        { "name" :"ZOOKEEPER_CLIENT" },
        { "name" :"PIG" },
```

```
            { "name" :"WEBHCAT_SERVER" },
            { "name" :"HST_AGENT" },
            { "name" :"MAPREDUCE2_CLIENT" },
            { "name" :"ZOOKEEPER_SERVER" },
            { "name" :"HIVE_CLIENT" },
            { "name" :"METRICS_COLLECTOR" }
          ],
          "configurations" :[ ],
          "name" :"master3",
          "cardinality" :"1" },
        { "components" :[
            { "name" :"NODEMANAGER" },
            { "name" :"HST_AGENT" },
            { "name" :"DATANODE" },
            { "name" :"METRICS_MONITOR" },
            { "name" :"SUPERVISOR" }
          ],
          "configurations" :[ ], "name" :"slaves",
          "cardinality" :$NUMBER_OF_SLAVES }
      ],
      "settings" :[
        { "recovery_settings" :[ { "recovery_enabled" :"true" } ] },
        { "service_settings" :[
            { "name" :"HIVE",                    "credential_store_enabled" :"true" },
            { "name" :"AMBARI_METRICS", "recovery_enabled" :"true" }
          ] },
        { "component_settings" :[
            { "name" :"METRICS_COLLECTOR", "recovery_enabled" :"true" }
          ] }
      ],
      "Blueprints" :{
        "blueprint_name":"my_hadoop_cloud",
        "stack_name"          :"HDP",
        "stack_version" :"2.6"
      }
    }
```

c. 注册蓝图：

```
> BLUEPRINT=`cat ~/ambari-hostmap.json`
> AMBARI_SERVER=ambari.my_cloud.org
> BLUEPRINT_NAME=my_hadoop_cloud
> URL="http://${AMBARI_SERVER}:8080/api/v1/blueprints/${BLUEPRINT_NAME}?validate_topology=false"
> curl -H "X-Requested-By: ambari-script" -d "$BLUEPRINT" -X POST -u admin:admin $URL
```

注：您也可以通过向 http://${AMBARI_SERVER}:8080/api/v1/ blueprints/${BLUEPRINT_NAME} 发送 GET CURL 请求的方式来确认您已经成功注册了蓝图。

d. 注册集群（主机映射）

```
> HOSTMAP=`cat ~/ambari-blueprint.json`
> AMBARI_SERVER=ambari.my_cloud.org
> CLUSTER_NAME=my_hadoop_cloud
> URL="http://${AMBARI_SERVER}:8080/api/v1/clusters/${CLUSTER_NAME}?validate_topology=false"
> curl -H "X-Requested-By: ambari-script" -d "$HOSTMAP" -X POST -u admin:admin $URL
```

6. 打开浏览器，导航至 AMBARI_SERVER:8080，登录并监控运行状态，直至集群部署完成为止。

注：您也可以通过向 http://${AMBARI_SERVER}:8080/api/v1/clusters/${CLUSTER_ NAME}/requests/1 发送 GET CURL 请求的方式来监控部署状态。

安装 HiBench

在客户端节点上，执行以下步骤：

1. 安装先决条件：

   > apt-get install -y maven git python-numpy libblas-common libblas-dev \ libblas3 liblapack-dev liblapack3 liblapacke \ liblapacke-dev bc

2. 克隆 HiBench：

   > mkdir ~/HiBench

   > cd ~/HiBench

   > git clone https://github.com/intel-hadoop/HiBench.git .

3. 构建 HiBench：

   > cd ~/HiBench

   > mvn -Dspark=2.1 -Dscala=2.11 clean package

4. 禁用 SSH StrictHostKeyChecking：

   > nano ~/.ssh/config

   Host *

   StrictHostKeyChecking no

   [...]

   注：在步骤 4 中，您可以生成监控/利用率报告。

安装 LPCPU

在从节点上，执行以下步骤：

1. 安装先决条件：

   > apt-get install -y sysstat

2. 下载 LPCPU 压缩包 (Tarball)：

   > wget -o ~/lpcpu.tar.bz2 “https://www.ibm.com/developerworks/community/wikis/form/anonymous/ api/wiki/26579cc3-66fe-42b8-baf9-1fcc88445848/page/4a7d2717-05c8-4b78-886e-5e4f9fdf07c1/ attachment/7018590b-7fbe-4852-8d44-79af2d83fffa/media/lpcpu.tar.bz2”

3. 提取 LPCPU：

   > cd ~

   > tar -xjvf lpcpu.tar.bz2

准备 HiBench

1. 计算或记下以下加粗的变量值：

   - HDFS 用户名

     HDFS_USER = XXX

   - 第一个主节点完全合格域名

     MASTER_1_FQDN = master1.my_cloud.org

   - 所有主节点的完全合格域名（用空格隔开）

     MASTER_FQDNS = master1.my_cloud.org master2.my_cloud.org master3.my_cloud.org

   - 所有从节点的完全合格域名（用空格隔开）

     SLAVE_FQDNS = slave1.my_cloud.org slave2.my_cloud.org slave3.my_cloud.org

   - 工作负载的大小（小型、大型、超大型、大数据等）

     HIBENCH_SCALE = gigantic

   - 从节点计数

     S = 从节点的总数量

   - 线程计数

     C = 所有从节点的 CPU/线程的总数量

   - 任何从节点的最小内存 (GiB)

     U = 128

   - 内存占用百分数，推荐值为 0.9 (90%)

     W：0

   - 主节点数量

     M：3

- Spark 内核数（推荐值为 5）
  EXECUTOR_CORES = 5
- Spark 执行器数量
  EXECUTOR_NUM = floor((C-S) / EXECUTOR_CORES)
- Spark 内存
  EXECUTOR_MEMORY = floor((U*S) / EXECUTOR_NUM)
- Spark 内存
  DRIVER_MEMORY = floor((U*S) / EXECUTOR_NUM)
- Spark 映射并行化
  MAP_PARALLELISM = C-S
- Spark 归约并行化
  SHUFFLE_PARALLELISM = C-S

2. 设置 HiBench Hadoop 配置：

```
> cp ~/HiBench/conf/hadoop.conf.template ~/HiBench/conf/hadoop.conf
> nano ~/HiBench/conf/hadoop.conf [...]
hibench.hadoop.home /usr/hdp/current/hadoop-client hibench.hdfs.master
hdfs://${MASTER_1_FQDN}:8020/user/${HDFS_USER}"

[...]
```

3. Set HiBench Spark Config:

```
> cp ~/HiBench/conf/spark.conf.template ~/HiBench/conf/spark.conf
> nano ~/HiBench/conf/spark.conf [...]
hibench.spark.home                  /usr/hdp/current/spark2-client
hibench.yarn.executor.num           ${EXECUTOR_NUM}
hibench.yarn.executor.cores         ${EXECUTOR_CORES}
spark.executor.memory               ${EXECUTOR_MEMORY}
spark.driver.memory                 ${DRIVER_MEMORY}

[...]
```

4. 设置 HiBench Spark 配置：

```
> cp ~/HiBench/conf/hibench.conf ~/HiBench/conf/hibench.conf.orig
> nano ~/HiBench/conf/hibench.conf

[...]
hibench.scale.profile                   ${HIBENCH_SCALE}
hibench.streambench.kafka.home          /usr/hdp/current/kafka-broker
hibench.default.map.parallelism         ${MAP_PARALLELISM}
hibench.default.shuffle.parallelism ${SHUFFLE_PARALLELISM}
hibench.masters.hostnames               '${MASTER_FQDNS}'
hibench.slaves.hostnames                '${SLAVE_FQDNS}'
[...]
```

5. 初始化存储：

   a. 如果之前已进行了存储初始化，则清除初始化：

```
> sudo -u hdfs hdfs dfs -rmr /HiBench
> sudo -u hdfs hdfs dfs -rmr /user/${HDFS_USER}
> sudo -u hdfs hdfs dfs -expunge
> sudo -u hdfs hdfs dfs -rmr '/user/*/.Trash'
```

   b. 创建所需目录：

```
> sudo -u hdfs hdfs dfs -mkdir /HiBench
> sudo -u hdfs hdfs dfs -chown -R ${HDFS_USER}:hadoop /HiBench
> sudo -u hdfs hdfs dfs -mkdir /user/${HDFS_USER}
> sudo -u hdfs hdfs dfs -chown ${HDFS_USER} /user/${HDFS_USER}
```

6. 运行准备脚本：

   > cd ~/HiBench

   > ~/HiBench/bin/workloads/ml/kmeans/prepare/prepare.sh

调整从节点

在所有从节点上，执行以下步骤：

1. 禁用 swap：

   > swapoff -a

2. 重新启用 swap：

   > swapon -a

3. 刷新页面缓存：

   > [-f /proc/sys/vm/drop_caches] && echo 3 > /proc/sys/vm/drop_caches

   注：您可以对所有节点进行该步操作，但在测试中，我们仅针对从节点。

启动 LPCPU 监控

在所有从节点上，执行以下步骤：

1. 启动 LPCPU 监控：

   > cd ~/lpcpu

   > ./lpcpu.sh duration=99999999 output_dir=output interval=${INTERVAL} dir_name=data

   注：在整个测试期间，应保持 SSH 会话处于打开状态、LPCPU 处于运行状态（或打包到 start-stop-daemone 中，使其在后台运行）。

运行 HiBench K-means 基准测试工具

在客户端节点上，执行以下步骤：

1. 运行基准测试工具：

   > cd ~/HiBench

   > ./bin/workloads/ml/kmeans/hadoop/run.sh

2. 收集 ~/HiBench/report 中的内容。

停止 LPCPU 监控

在所有从节点上，执行以下步骤：

1. 按下 [CTRL] + C，中断 SSH 会话中的 lpcup.sh。
2. 等待创建压缩包 (output/data.tar.bz2)。
3. 提取压缩包并运行 postprocess.sh 脚本。
4. 收集 output/data 中的内容。

测量带宽

我们使用基于 TCP 的 ping 测试工具及安全拷贝工具，测量了美国东海岸、英国及日本这三个地区的虚拟机之间的网络性能，了解了网络延迟情况。我们还通过 PaPing 测试了两个平台在这些方面的性能。

我们通过连接在目标云环境中运行的 Web 服务器进行了 TCP-ping 测试，并使用 ping 实用程序测量了端口 80 的平均延迟，持续时间为 30 秒，然后取三次测试的平均值。

我们使用 scp 测量了文件拷贝性能，还分别测试了传输 25 MB 文件、500 MB 文件以及包含多个小文件的 100 MB 文件夹所需的时间。

云实例配置

每个云平台的三台虚拟机分别位于美国东海岸、欧洲西部、亚太地区（日本），配置如下：

区域

- AWS：北弗吉尼亚、爱尔兰、亚太地区（东京）
- IBM Cloud：华盛顿特区、伦敦、东京

虚拟机实例类型

- 所有的虚拟机均运行 Ubuntu 14

虚拟机配置

- AWS EC2：“m4.xlarge”通用镜像，配备 4 个 vCPU、16GB RAM、EBS 存储及高速网络
- IBM Cloud：计算虚拟服务器配置为 4 个 2.0GHz 内核、16GB RAM、1 Gbps 公用和私有网络上行链路

本项目受 IBM 的委托进行。