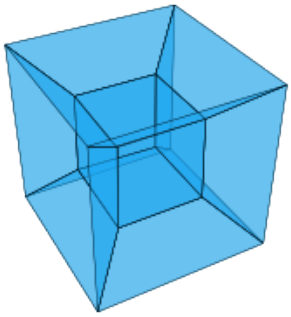
GISpark Documentation

发布 ***0.1***

openthings

6月 02, 2016



Blaze

CHAPTER 1

索引

前言

我们正面临一个数据快速增长的时代。

数据获取的手段、获取的频度、数据的存储量、数据的类型、存储的方式都以前所未有的速度快速增加，而数据处理的手段和数据理解、利用的方法却相对落后，以至于陷入“信息爆炸”和“信息焦虑”的困境。对于地理空间信息领域来说，现在拥有智能手机的每个人都可以随时获取位置信息，以及位置相关的视频、图像、声音、温湿度、速度、姿态等各种信息，而卫星和无人机等设备获取的空间图像数据更是快速增长，各种物联网传感器也在每时每刻发送大量数据，迫切需要新的信息技术方法来对数据进行处理、分析和利用，为资源利用、环境保护、城市发展、灾难应急等提供有效、及时、准确的支持。

传统的GIS软件以（已知结构的）数据获取、存储、管理功能为主，并添加了各种时间维度和空间维度的专题图展示功能，具有空间统计、地理分区、路径分析、选址优化等空间分析方法，在城市建设、交通运输、气象预测、区域发展、决策支持等方面发挥了非常重要的作用，但总体上来说，仍然以数据的搬运和直接展示为主，缺乏对数据的深度理解和多维度的分析方法。而且，现有的数据管理和分析算法也难以扩展到大规模的分布式并行计算系统之上，无法满足对非结构化的未知数据的探索性分析的需要，无法满足大规模数据快速处理和复杂时空关系的分析需要，无法满足大规模的数据挖掘和高级复杂模型的研究需要，也无法让更多的人参与进来，顺利开展跨领域的交叉性研究工作。

GISpark-地理空间信息分布式计算平台设计为主要满足这类新的需求的平台。GISpark基于最新的虚拟化计算基础设施（如OpenStack/Docker）和分布式计算架构（如Spark/Hadoop）技术构建，集成Anaconda科学计算环境、Jupyter Notebook等数据探索分析工具和GIS的时空数据管理和分析系统，将大规模计算能力运用于时空信息分析，不仅为地理空间信息领域提供大数据处理能力，同时为其他数据应用领域提供强大的时空分析模型和高级空间可视化工具。

这里介绍GISpark的组成、体系架构和详细的部署方式，并通过案例帮助使用者逐步进入时空大数据分析的殿堂，领略数据之美、世界之妙。

- 本文档在线发布：http://gispark.readthedocs.io ，欢迎访问和提出宝贵意见。
- 本文档提供源码，点击此处获取。注意: 需要安装Jupyter Notebook和NBSphinx, 使用Sphinx编译。
- 本站点文档许可：GPL, 授权任何非商业用途使用，但需提到：内容来自于 http://gispark.readthedocs.io 。

GISpark简介

GISpark为数据工程专家、大数据科学家、地理与空间研究的学者而设计，将理论、方法、模型、算法与数据链接为一个开放的平台，使任何对时空数据和地理建模感兴趣的人和各领域研究都能从中获益。GISpark提供完整的时空数据处理与建模分析工具集和运行环境，从而让使用者能够减少大规模IT系

统复杂性的影响，将更多精力投入到关注所要解决的问题本身。GISpark集GIS方法、科学计算环境和云计算、分布式计算技术之大成，将大数据技术运用于地理空间领域问题的解决，同时将GIS的时空分析能力带入大数据行业，促进时空一体化的大数据分析技术和大数据解决方案的发展。

GISpark是什么?

GISpark 是用于大规模地理空间信息处理与建模分析的分布式计算平台。

- GISpark基于OpenStack云计算环境和Docker容器技术进行开发，用于构建多用户托管的云计算基础设施。
- GISpark采用HDFS/Ceph/MongoDB等大规模虚拟化存储系统进行时空信息的存储，能够处理大规模的矢量、影像和流式数据，能够兼容已有的GIS系统，并拓展GIS的数据服务和空间分析能力到整个IT基础设施。
- GISpark采用最新的Spark分布式并行计算框架，融合GIScript地理空间信息脚本引擎和多种开源的GIS模型库，可以快速集结数百台服务器进行大规模的时空信息处理和地理模型分析。
- GISpark集成Anaconda科学计算环境和Jupyter notebook交互计算Web界面，包含大量的经典算法和模型库，可以为时空大数据处理和时空一体化分析提供持续发展的模型系统。
- GISpark集成TensorFlow和Orange等多种机器学习算法库，包含深度学习、对象识别、网络分析、自主分类、降维分析、随机森林、支持向量机等多种算法，为地理空间模型与机器学习的进一步融合打下了坚实的基础。
- GISpark是一个高度可扩展的架构，全部采用开源软件搭建，同时可以集成SuperMap GIS的强大地理空间数据处理和制图、可视化与地图发布服务、空间分析服务等重要能力，并能集成其它各种商业软件模块。
- GISpark集成GitLab源代码托管、Jenkins自动化构建、Maven软件仓库等持续集成软件，充分利用Git的分布式版本管理能力，可以为用户构建一个持续交付的IT基础设施。以及通过SuperMap Portal/iServer/Express等服务器系统和iDesktop/iClient/iMobile等客户端工具的综合运用，快速建立时空大数据处理和在线分析的GIS云平台。

GISpark易于学习，并且随着社区不断演进，是一个自我进化的时空大数据平台。

GISpark 体系架构

GISpark并非一个固定的框架或模式，采用开源框架并保持开放，让各种模型和算法可以非常容易集成进来，甚至于改变GISpark本身。

GISpark通过OpenStack等云计算环境建立计算基础设施，采用Docker等容器技术为多用户建立独立的运行环境。GISpark通过虚拟存储技术建立统一的大规模存储系统和访问接口，保障所有用户可以按照一致的方式访问需要的数据。GISpark为每一个用户建立Anaconda科学计算环境，可以在线运行Python/R以及其他语言建立的模型，并通过Github或类似的方式进行模型和计算结果的版本控制、即时协作和在线共享。

GISpark通过运行于大规模计算资源的Spark或iPyParallel等分布式调度服务来按需提供计算能力，以最快的速度获得计算结果，并通过网络发布出来，供所有连接的终端和系统进行使用。

GISpark 实施路线图

可以使用基于GISpark建立的IT服务设施，或者搭建自己的GISpark运行环境。

如果使用GISpark服务资源，只需要登入服务系统，就能获得需要的基础服务，但仍然需要掌握数据分析的基本方法，如Python及其相关的NumPy/SciPy/Pandas/Spark等基础知识，对于时空数据的格式和模型不需要

精确地了解，但数据的特性、使用的方法、模型的适应性和专业领域的知识的深入掌握是必须的能力，是驾驶大数据获得正确结论和解决领域问题的必由之路。

如果搭建自己的GISpark运行环境，这里提供了完整的指南。所采用的软件都已经广泛使用并且提供源代码，而且可以从开放文档获得经验，以及从社区获得帮助，或者通过其生态系统获得技术支持和定制的服务。搭建自己的GISpark运行环境具有强大的灵活性，可以自行决定配置的组件和规模，以及通过定制更好地与其他业务连接，实现自动化的处理流程。不过，这需要对Linux/Dokcer/Spark/Jupyter/GitLab等的安装和配置有足够的了解，并且对网络运维和安全配备一定的人员进行管理和维护，对资源的使用进行调度以达到更高的使用效率。

将公共GISpark服务和专有的GISpark环境进行对接，迁移和混合使用，也是非常简单的事情，因为二者采用了相同的软件接口和技术栈。因此，我们也建议尽量保持自己的环境与公有设施尽量保持一致，对不一致的地方进行隔离以便单独处理。

可持续的信息基础设施

GISpark 的目标是提供一个可持续的地理空间信息基础设施。

信息系统的持续性是一件极具挑战性的事情。因为信息系统的运行过程中有很多因素导致原先设定的目标无法达成，或在运行过程被中断，导致所积累的知识和付出的投资无法得到回报，问题也无法得到最终的解决。

影响信息系统持续性主要有五大挑战性因素，包括：

1. **业务变化。**往往由需求变化引起，然后系统设计的目标将随着发生大的变化，从而引发功能、软件形态和运行形式的巨大调整，进一步引起业务流程和管理制度的改变和要素的重组，原来的架构被重新开发，模块的重用也成为泡影。
2. **人员变动。**人员频繁变化是IT领域面临的通病，这不仅包括人员组织的变化，也包括与其相关的知识结构、文档体系、培训教育和知识的传承。即便“人”全部保持不变，也会面临由于新技术的引入需要学习新知识和采用新流程的问题，导致难以应对新的挑战和不断解决新的问题。
3. **数据变化。**业务运行中，数据规模将不断增长，不断引入新的数据类型，以及建立新的数据获取手段、为了新功能而进行数据结构的调整，都会导致原来的数据基础发生变化，引发效率降低、系统崩溃、数据错误，甚至变得不再可用。而数据的更新往往需要花费巨大的代价和时间成本。
4. **系统扩容。**随着系统的投入运行，往往规模会变得越来越大。当现有实施无法满足业务增长需要时，就需要进行扩容和部署的变更。遗憾的是，大部分系统并不能线性地进行规模扩展，效率会急剧下降甚至进入瓶颈阶段，甚至引发大面积的宕机现象，让业务步入泥潭。
5. **软件更新。**由于功能调整、性能优化或者仅仅是Bug修复，软件总是处于不断的优化、升级的过程中。并非每次升级都能解决问题，有时候还会引入新的破坏性因素。可怕的是，如果对升级的影响缺乏足够的评估和测试、验证，可能会带来一系列的连锁反应，导致升级的结果无法控制、成本超出预期。

GISpark 采用虚拟化和微服务架构提供灵活的基础设施，可以实现资源的动态调度和服务的动态编排、负载均衡、故障转移等能力。

通过基于Git的分布式版本控制，可以实现多版本共存和网络协同，从而降低需求变化带来的不利影响，以及应对数据和软件版本的兼容性问题。

通过Maven软件仓库和Jekins持续集成和持续交付的功能，将软件开发、自动化测试、快速部署过程集成到一个预定的流程之中，实现软件“从摇篮到坟墓”的持续交付过程的规范化和自动化，可以将原有的组件和代码实现更高程度的复用，从而提高运营的效率，减少人为因素在大规模部署中的影响。

通过软件技术与开源社区的融合，为人才培养提供社会化的储备，而且可以从社区获得知识、经验和更好的解决方案，通过向社区共享成果，构建共赢、共享、互利的开放社区，使这一过程进入良性循环。

云计算基础设施

这里介绍云计算的基本概念，虚拟存储，弹性计算，容器技术，资源调度等内容，涉及到OpenStack，Docker，Mesos，Hadoop，Ceph等云计算相关的软件，帮助使用者建立一个基于虚拟大数据存储、弹性动态调度的高性能分布式计算平台。

云计算概述

云计算基础设施提供动态调度的计算资源池可以快速满足用户的计算需要。计算资源池通常包括存储、网络、镜像、实例以及相关的管理、调度、计费和监控、预警等工具。

云计算基础设施的分层

云计算基础设施一般分为IaaS(基础设施作为服务，如虚拟机)、PaaS(平台作为服务，如Web服务器)、SaaS(服务作为服务，如API接口)等几层，也有人提出 DaaS(数据作为服务，如NoSQL服务)和AaaS(应用作为服务，如Spark)。但在实际部署和应用中，这些层是完全融合在一起的，部分软件往往也可以同时完成多层的功能。

构建自己的IaaS基础设施

构建自己的Iaas云环境并将其提供给用户，需要提供以下几个特性:

1. 允许应用用户注册云服务、查看使用情况以及账单。
2. 允许开发商和开发人员创建和存储自定义的镜像。
3. 允许开发商和开发人员启动、监控、停止虚拟机实例。
4. 允许操作人员配置和操作云基础设施，提供API可编程接口。

可以使用在线的云计算服务如Amazon和Aliyun，也可以自己搭建专用的云计算平台。搭建专用云计算平台，可以使用开源的OpenStack或者商用平台。GISpark本着开放、易用的原则，主要选用OpenStack，但其中的各种服务也可以兼容主流的商用平台，或者将其部署到在线的云服务系统之上。

云计算向应用发展的趋势

目前的趋势是，云计算逐步向应用端、为最终用户提供直接服务发展，而不仅仅提供基础设施。这就要求云计算平台不仅提供计算、存贮等资源，还需要提供数据、应用软件、流程与模型等应用层能力，如面向机器学习的应用服务平台。

OpenStack的安装

OpenStack包含块存储系统、对象存储系统、虚拟机镜像管理系统、弹性计算系统以及集成管理控制台等多个子系统(查看 OpenStack软件与体系架构)，采用Python编写，使用RabbitMQ作为服务调度总线。OpenStack是一个社区项目，提供开放源码软件，帮助企业运行虚拟计算或者存储云。

OpenStack当前包括三个子项目，三个项目相会独立，可以单独安装(查看OpenStack官方文档，中文安装攻略参考这里，系统架构可以参考这里)。

- Swift: 提供对象存储。这是大致类似于Rackspace云文件（从它派生）或亚马逊S3。
- Glance: 提供OpenStack Nova虚拟机镜像的发现，存储和检索。
- Nova: 根据要求提供虚拟服务。这与Rackspace云服务器或亚马逊EC2类似。

  提示：**OpenStack在快速演进，一些文档(尤其是中文)可能已经过时，请注意版本差别。**

对象存储服务-Swift

OpenStack对象存储是一个在具有内置冗余和容错的大容量系统中按照对象方式存储和检索数据的服务系统，在OpenStack内部用来存储虚拟机镜像。对象存储还有各种应用，如备份或存档数据，存储图形或视频，储存二级或三级静态数据。通过开发与数据存储集成的应用程序，创造弹性和灵活的云存储Web应用程序。

虚拟机镜像服务-Glance

OpenStack镜像服务是一个查找和虚拟机图像检索系统，支持镜像的存储、索引、上传、下载、复制、删除等各种操作。它可以配置三种方式：使用OpenStack对象存储来存储镜像文件，使用亚马逊S3直接存储，或使用S3对象存储作为S3访问中间存储。

弹性计算服务-Nova

OpenStack计算服务是一个云控制器，用来启动一个或一组虚拟实例。Nova从Glance中获取镜像的存储，然后将该镜像投入，然后对该实例的运行状态进行监控、管理。Nova也用于配置每个实例运行期的参数配置，如包含多个实例为某个特定项目的联网(查看OpenStack VLan的部署)。

管理控制服务-Horizon

OpenStack的主要组件如镜像、存储、计算都实现了命令行和API接口(查看中文解释 命令行参数，API的使用)，可以非常容易地通过`Shell`或者`Python`脚本进行控制。同时，为了方便系统监控和可视化管理，还提供了基于Web的管理工具－Horizon，可以交互地管理系统中的主要资源（该工具还在发展之中，部分用得较少的功能和参数调节还不能通过Web界面操作），对于网管新手和小型团队还是很方便的，适合快速入门使用(安装参考在Ubuntu16.04上安装Horizon)。

自动部署和管理OpenStack

OpenStack体系非常庞大，涉及到大量的软件，安装过程相对繁琐和复杂，而且容易引起版本的兼容性问题。因此，很多社区都发展出了一些自动化的部署和管理工具，可以提升OpenStack基础设施建立和管理的效率。

OneStack一键自动化部署OpenStack工具。OneStack目前能够完整而正确在Ubuntu12.04（precise）安装部署OpenStack。一键完整部署OpenStack，可以自定义配置，无需交互；安装过程不需要等待提示和输入配置，mysql密码可以自行配置，也可以使用默认的，不需要等待mysql等程序安装的提示；数据库密码可以自行配置，全部完整安装和部署；网络配置可以自行定义；配置文件和依赖关系已经处理；设置变量配置kvm或者虚拟机配置qemu；默认安装一个Ubuntu12.04的操作系统镜像，并启动一个实例，通过运行状态可以查看是否正确部署和运行；通过dashboard进行web管理和查看，或者nova命令管理。

FuelWeb-OpenStack自动化部署工具。FuelWeb中文版是由TryStcack社区联合Mirantis公司开发的OpenStack部署工具。支持Red Hat® 企业版Linux® 和Grizzly版OpenStack®平台。最新特性包括简单快速部署OpenStack云平台(查阅Fuel文档)：(1)部署和HA配置支持，支持最新的RedHat企业版本；(2)OpenStack健康检查，部署后会对云平台进行相关测试，并对OpenStack核心组件和Fuel提供的部署配置工具使用的额外的包进行验证；(3)单一的集成包,包括Fuel图形控制界面和命令行脚本，可以通过单独的ISO或IMG获取；(4)提升HA弹性，通过Pacemaker提供的高可靠性服务，以及Clusterlabs开发的可扩展的集群资源管理器提供；

Solum-云服务应用和集成到用户的工具。Solum是专为OpenStack设计的，使OpenStack云服务能更简单的应用和集成到用户的应用开发过程中。Solum利用现有OpenStack服务，比如 Heat, Nova, Glance, Keystone, Neutron等等，通过一个`push`按钮就可以转换代码到一个OpenStack云上的管理应用程序中。Solum 是一个社

区驱动的项目。这个项目源于各大社区的合作讨论，主要贡献者来自：Rackspace, eBay, dotCloud, Cumulogic 和 CloudSoft等。

Tempest为云计算平台OpenStack提供集成测试。Tempest为云计算平台OpenStack提供集成测试的开源项目，基于 unittest2 和 nose 建立的灵活且易于扩展及维护的自动化测试框架，使得 OpenStack 相关测试效率得到大幅度提升。

openstack-ops-tools管理Openstack的业务工具和实用程序。openstack-ops-tools用于管理 Openstack 的业务工具和实用程序。是一个Python编写的工具包，用于执行OpenStack管理上的自动化操作。

数据虚拟化存储系统

虚拟化的存储系统将传统的硬件磁盘集群通过软件来实现，并提供跨多台机器的多个硬盘的统一访问接口，并提供并发访问、冗余存储和多设备容错、分布式内存缓存等分布式存储的高级特性。虚拟存储系统有的提供POSIX文件接口可以像本机文件一样访问，有的提供专用编程接口API和SDK，有的提供REST网络访问接口，有的提供SQL引擎的查询接口。虚拟存储系统从实现的接口层次上可以分为三种：1、块存储，提供类似于“裸盘”磁盘块的访问方式；2、对象存储，提供类似“文件”或者“数据包”的访问接口；3、列存储，提供类似于数据库的“表”和“记录”的访问接口。

虚拟块存储设备

iSCSI网络存储设备

iSCSI设备通过IP网络连接磁盘，并虚拟为一个统一的物理设备。技术已经相当成熟，并形成了配套的硬件体系。用户可以直接购买硬件化的存储设备，也可以安装相应的软件将一台Linux服务器转化为一个网络存储设备。iSCSI在云计算概念出现之前就已存在，是一个底层的网络磁盘协议实现，旨在构建一个网络无关的“大容量物理磁盘”，磁盘必须使用专用的格式进行管理。

Cinder虚拟存储服务

Cinder (中文介绍)虚拟存储服务是一个框架，可以将多种存储融合为一个统一的“卷”进行管理和使用，因此可以充分利用各种现有的存储系统资产。当然，强大兼容性的代价是性能会有所影响。Openstack从Folsom开始使用Cinder替换原来的Nova-Volume服务，为OpenStack云平台提供块存储服务。

Cinder服务的组成

- API service：接受和处理Rest请求，将请求放入RabbitMQ队列。提供Volume API V2。
- Scheduler service: 处理任务队列的任务，并根据预定策略选择合适的Volume Service节点来执行任务。目前版本的cinder仅仅提供了一个Simple Scheduler, 该调度器选择卷数量最少的一个活跃节点来创建卷。
- Volume service: 该服务运行在存储节点上，管理存储空间。每个存储节点都有一个Volume Service，若干个这样的存储节点联合起来可以构成一个存储资源池。

Cinder服务的驱动

为了支持不同类型和型号的存储，当前版本的Cinder为Volume Service如下drivers。当然在Cinder的blueprints当中还有一些其它的drivers，以后的版本可能会添加进来。

```
本地存储：LVM, Sheepdog
网络存储： NFS, RBD (RADOS)
IBM: XIV, Storwize V7000, SVC storage systems
Netapp: NFS存储
ISCSI存储则需要OnCommand 5.0和Data ONTAP 7-mode storage systems with installed iSCSI licenses。
```

```
EMC: VNX, VMAX/VMAXe
Solidfire: Solidfire cluster
```

Cinder服务的部署

上述的Cinder服务都可以独立部署，cinder同时也提供了一些典型的部署命令:

```
cinder-all: 用于部署all-in-one节点，即API, Scheduler, Volume服务部署在该节点上。
cinder-scheduler: 用于将scheduler服务部署在该节点上。
cinder-api: 用于将api服务部署在该节点上。
cinder-volume: 用于将volume服务部署在该节点上。
```

虚拟对象存储系统

对象存储提供类似“文件”或者“数据包”的访问接口，一般建立在操作系统的文件系统之上，让访问者可以存取一个“巨大的文件仓库”，而不用关心具体的文件如何组织、存放在何处。对象存储系统一般还提供冗余存储、并行访问、故障转移、动态备份等特性，可以通过专用软件、SDK或者文件系统接口、Web应用等来进行访问。

由于对象存储的多份拷贝需要处理一致性问题，往往写入时较慢，而读取时可以并行访问速度可以很快，适合于写入少、读写多的场合，比如互联网站的日志分析、数据挖掘等场合使用。对于高速写入的实时数据，则需要设计数据分区策略和采用适当的并行写入措施，才能更好地适应流数据的处理模式。

HDFS 分布式文件系统

HDFS(Hadoop Distributed File System)是Hadoop的基础存储系统，其HBase列存储系统和MapReduce分布式处理系统都建立在HDFS之上。HDFS通过建立Master和Slave节点，将多个节点的磁盘整合为一个统一的资源，并将数据目录(相当于单机的“文件分配表”)保存在Master节点上，从而建立一个跨越多个节点和存储设备的虚拟化的“文件系统”。HDFS将数据划分为“块”，在多个节点保存至少三个拷贝，读取时可以从多个节点同时读取。因此，HDFS分布式文件系统具有较高的“读文件”吞吐率，可以获得较高的读访问带宽。HDFS也支持通过Web接口和文件接口进行访问(效率不算高)。

HDFS已经发展多年，已经在互联网领域大量使用，相对来讲比较稳定可靠。但也存在一些缺点，一直以来未得到有效的改进，包括：由于HDFS保存多个拷贝，导致写入数据时也需要写入多个备份而导致花费较多的时间，因此不适合需要数据大量写入的场合；虽然数据冗余存储，但数据目录保存在Master，如果Master节点失效，将导致整个存储系统失效；如果有大量小文件写入，将导致Master节点的内存占用和查找时CPU占用急剧增长，性能大幅下降。对于快速变动、时效性强而单次数据量小的流式数据处理，不太适合采用HDFS进行存储，而对于存档(不再修改)的图像、视频、打包的日志等大文件较为适合，能提供较大的带宽。

Ceph 分布式文件系统

Ceph在一个统一的系统中同时提供了对象、块、和文件存储功能。Ceph是加州大学Santa Cruz分校的Sage Weil（DreamHost的联合创始人）博士论文期间设计的分布式文件系统。自2007年毕业之后，Sage开始全职投入到Ceph开发之中，使其能适用于生产环境。Ceph的主要目标是设计成基于POSIX的没有单点故障的分布式文件系统，使数据能容错和无缝的复制。这篇文章探讨了Ceph的架构，它的容错实现和简化海量数据管理的功能。

Ceph在标准POSIX文件接口之上提供大规模分布式文件管理能力，使任何操作系统可以像访问本地文件系统一样访问网络存储。2010年3月，Linus Torvalds将Ceph client合并到内核2.6.34中。2016年，在Ubuntu16.04版本也将Ceph纳入了其发行版中。Ceph的中文文档较为完整，Ceph中文社区也有大量的资源和使用经验分享。

Ceph的相关工程包括(使用OSChina搜索):

- Calamari 是 Ceph 的管理和监控服务，输出REST API。访问Calamari通过客户端代码。
- ceph-dash 是用 Python 开发的一个 Ceph 的监控面板，用来监控Ceph的运行状态，同时提供REST API来访问状态数据。
- Inkscope 是一个 Ceph 的管理和监控系统，依赖于Ceph提供的API，使用MongoDB来存储实时的监控数据和历史信息。
- autobuild-ceph包含一组脚本和一个fabric文件 (fabfile.py) 用来远程部署Ceph以及Ceph的自动构建。
- ceph-deploy 是一个Ceph的简易部署工具，可以非常方便的部署Ceph集群存储系统。
- ceph-zabbix 是一个 Zabbix 插件，用来监控 Ceph 集群文件系统。

FastDFS 分布式文件系统

FastDFS是一个开源的分布式文件系统(中文社区)，它对文件进行管理，包括文件存储、文件同步、文件访问(文件上传、文件下载)等，解决了大容量存储和负载均衡的问题。特别适合以文件为载体的在线服务，如相册网站、视频网站等等。

- FastDFS同时对文件的meta data进行管理。
- 文件的meta data就是文件属性，以键值对（key value pair）方式表示：如：width=1024，其中的key为width，value为1024。
- 文件meta data是文件属性列表，可以包含多个键值对。

FastDFS服务端有两个角色：跟踪器（tracker）和存储节点（storage）。跟踪器主要做调度工作，在访问上起负载均衡的作用；存储节点存储文件，完成文件管理的所有功能，如存储、同步和提供存取接口。跟踪器和存储节点都可以由一台或多台服务器构成。跟踪器和存储节点中的服务器均可以随时增加或下线而不会影响线上服务。其中，跟踪器中的所有服务器都是对等的，可以根据服务器的压力情况随时增加或减少。

FastDFS支持大容量存储，存储节点（服务器）采用了分卷（或分组）的组织方式。FastDFS中的文件标识分为两个部分，卷名和文件名，二者缺一不可。存储系统由一个或多个卷组成，卷与卷之间的文件是相互独立的，所有卷的文件容量累加就是整个存储系统中的文件容量。一个卷可以由一台或多台存储服务器组成，一个卷下的存储服务器中的文件都是相同的，卷中的多台存储服务器起到了冗余备份和负载均衡的作用。在卷中增加服务器时，同步已有的文件由系统自动完成，同步完成后，系统自动将新增服务器切换到线上提供服务。当存储空间不足或即将耗尽时，可以动态添加卷。只需要增加一台或多台服务器，并将它们配置为一个新的卷，这样就扩大了存储系统的容量。

SeaFile 分布式文件系统

Seafile 是新一代的开源云存储软件(访问SeaFile源码)。你可以把Seafile想象成是面向团队的开源Dropbox，SeaFile提供更丰富的文件同步和管理功能，以及更好的数据隐私保护和群组协作功能。Seafile支持 Mac、Linux、Windows 三个桌面平台，支持 Android 和 iOS 两个移动平台。Seafile 是由中国团队开发的国际型项目，目前已有10万左右的用户，以欧洲用户为多。典型的机构用户包括比利时的皇家自然科学博物馆，德国的Wuppertal气候与能源研究所等。

NoSQL存储系统

广义的NoSQL存储系统包括除了传统的关系数据库和文件系统之外的各种存储系统，包括K-V存储、文档数据库、分布式内存系统以及XML数据库、JSON数据库等。XML数据库、JSON数据库更类似于增强的文件系统，近年来的成熟产品不多，使用范围较小，而且能比较容易被其它类型存储系统取代。这里重点介绍以K-V存储为代表的NewSQL存储系统（虽然体系架构不同于传统数据库，但其支持传统的SQL查询，现在更多地被称为NewSQL而不是NoSQL）。

MongoDB

MongoDB是典型的NewSQL产品(文档)，是非关系数据库当中功能最丰富，最像关系数据库的。MongoDB支持的数据结构非常松散，类似json的bjson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

MongoDB的特点是高性能、易部署、易使用，存储数据非常方便。主要功能特性有:

```
1.面向集合存储，易存储对象类型的数据。模式自由。
2.支持查询，支持动态查询。支持完全索引，包含内部对象。
3.使用高效的二进制数据存储，包括大型对象（如视频等）。文件存储格式为BSON（一种JSON的扩展）。
4.自动处理碎片，以支持云计算层次的扩展性。
5.支持RUBY，PYTHON，JAVA，C++，PHP等多种语言。
6.可通过网络访问，支持复制和故障恢复。
```

所谓"面向集合"（Collection-Oriented），意思是数据被分组存储在数据集中，被称为一个集合（Collection)。每个集合在数据库中都有一个唯一的标识名，并且可以包含无限数目的文档。集合的概念类似关系型数据库（RDBMS）里的表（table），不同的是它不需要定义任何模式（schema)。模式自由（schema-free)，意味着对于存储在mongodb数据库中的文件，我们不需要知道它的任何结构定义。如果需要的话，你完全可以把不同结构的文件存储在同一个数据库里。存储在集合中的文档，被存储为键-值对的形式。键用于唯一标识一个文档，为字符串类型，而值则可以是各中复杂的文件类型。我们称这种存储形式为BSON（Binary Serialized dOcument Format）。

MongoDB服务端可运行在Linux、Windows或OS X平台，支持32位和64位应用，默认端口为27017。MongoDB把数据存储在文件中（默认路径为：/data/db），为提高效率使用内存映射文件进行管理。推荐运行在64位平台，因为MongoDB在32位模式运行时支持的最大文件尺寸为2GB。

- Mongodb Management Studio 功能如下:

```
1.服务器管理功能，添加服务器,删除服务器
2.服务器,数据库,表,列,索引,树形显示和状态信息查看
3.查询分析器功能，支持select,insert,Delete,update
4.索引管理功能，支持列名的显示,索引的创建,查看,删除.
5.数据库Profile管理，可以设置Profile开关,查看Profile信息.自定义分页大小.
6.master/slave信息显示。
```

MongoDB的其它工程可以使用OSChina搜索查看。

Redis

Redis是一个高性能的k-v数据库(查看源码)，提供简单快速的键-值访问机制，支持网络共享访问，支持持久化，非常适合存储程序中用到的词典/列表类信息，如用户信息等等。Redis虽然是一个轻量级的数据服务系统，但也已经发展成了一个完整的社区生态系统，具有集群、监控、管理等完整的功能(搜索OSChina)。

Hbase

HBase基于Hadoop的列存储服务系统，建立在HDFS分布式文件系统之上，支持MapReduce分布式处理。

Docker计算容器系统

Docker提供应用运行的"容器"，能够提供类似于虚拟机的环境隔离的效果，但是通过cGroup技术直接在内核上执行，因此性能上有较好的表现。Docker的运行代码也称为"镜像"，但与虚拟机不同的是，镜像是由多个"层"组合在一起构成的，底层的"层"可以在应用镜像间共享，因此会大大减少需要下载的数据量和启动一

个镜像的运行时间。Docker是完全开源的(源代码。目前，类似于Docker的“容器技术”已经有多种实现，包括Ubuntu上面的LXD和CoreOS的Rocket等。

Docker的安全性仍然是一个值得注意的问题。由于部分宿主操作系统不支持资源配额，可能引起部分容器资源占用大量资源而导致其它服务出现问题，可以通过监控以及Mesos之类的进行管理；如果导入镜像时引入了含有木马等不恰当的软件，将很难发现和清除；软件出现Bug时将导致容器内进程从沙箱逃逸出来，从而获得主机权限并可以进而控制整个系统，包括其他容器。因此，使用Docker时应只下载值得信任的镜像，建议尽可能使用软件官方镜像，或者自行制作镜像，尽可能建立私有的镜像存储系统，能提供更好的安全性和更快的下载速度。

Docker的核心组件包括Docker Engine、Machine、Compose、Registry、Swarm等，分别执行运行、环境管理、服务编排、镜像存储和集群部署等功能。Docker.IO提供基于Docker的云平台和镜像管理的Docker Hub，通过Registry和Harbor可以搭建私有的Docker镜像库。通过Rancher搭建自己的基于Docker的PaaS平台，实现资源调度、运维、管理等功能。

Docker Engine

Docker Engine是整个Docker生态系统的运行时核心引擎，包括镜像的创建、管理和操作，镜像运行时称为容器，容器可以启动、停止或重新连接。Docker Engine是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化，而且不依赖于任何语言、框架或包装系统。

在Ubuntu上安装Docker Engine:

```
wget -c https://get.docker.com/ -O docker-setup.sh
sudo chmod +x docker-setup.sh
sudo ./docker-setup.sh
```

在Aliyun上安装后无法启动，主要因为缺省的网段被阿里云系统预分配占用了，手动分配即可:

```
docker --bip 192.168.100.1/24 daemon &
```

Docker Engine提供丰富的命令行操作，参见：Docker简明手册。 Docker Engine常见的操作包括:

```
docker pull ubuntu，下载ubuntu镜像。
docker run -i -t ubuntu /bin/bash，启动一个控制台。
docker images，列出本机的镜像文件。
docker ps，列出运行的容器。
docker ps -a，列出所有的容器。
docker commit <container> <some_name>，将容器存为一个镜像。
```

Docker可以通过Dockerfile进行自动构建，类似于Makefile，详细参考这里。以下是一个简单的Dockerfile，可以创建在github.com上，提交到Docker hub时，可以自动构建镜像。

```
FROM ubuntu:14.04
 RUN apt-get update
 RUN apt-get install -y curl
```

Docker Machine

Docker Machine在Mac OS X和Windows中创建和维护Docker运行环境(Linux下使用Docker系统服务，不需使用Docker Machine)。

安装参考下面的脚本（Ubuntu系统）:

```
wget https://github.com/docker/machine/releases/download/v0.7.0-rc3/docker-machine-Linux-x86_64
    -O docker-machine
```

```
sudo cp docker-machine /usr/local/bin/docker-machine
sudo chmod +x /usr/local/bin/docker-machine
```

这个文件存在Amazon上的，很多时候无法访问，Aliyun上也访问不了，解决办法参见阿里云安装Docker。

Docker Compose

Docker Compose组合多个Docker容器为一个服务链，如同时启动MySQL实例和Ngnix提供Web服务。

安装Docker-compose:

```
wget https://github.com/docker/compose/releases/download/1.7.0-rc2/docker-compose-`uname
    -s`-`uname -m` -O docker-compose
cp docker-compose /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

启用一个Compose服务(参见教程)，包括：

```
使用Dockerfile来生成容器的镜像。
在docker-compose.yml文件中定义服务，该服务将运行在隔离的环境。
运行`docker-compose up`自动启动所有关联的容器，提供服务。
```

docker-compose.yml文件如下所示(文件格式详细参考):

```
version: '2'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Docker Registry

Docker的公共镜像存储Docker hub具有大量的制作好的镜像，可以下载来直接使用(需要注意适用性和安全性)。也可以安装Registry服务来建立自己的私有镜像存储系统。建立专用镜像服务站可以使用Harbor，这是一个VMWare开源的镜像管理系统，提供WebUI，采用Nginx作为前端Web服务器，后台也是采用Registry进行管理。

Docker Swarm

Docker Swarm是Docker开发的集群管理软件，可以通过简单的配置将Docker在一个计算机集群上投入运行。除了Swarm，Docker还支持YARN，Mesos，Kubernetes的集群管理和调度机制。

Rancher 容器PaaS

Rancher是一个开源的容器管理平台，帮助构建企业私有容器服务，相当于KVM里的Openstack。Rancher是一个完善的开箱即用的容器管理平台。当企业开始去在生产环境部署 Docker 容器时，面临的首个巨大挑战是如何把一组数量众多的开源技术集成在一起。容器管理涉及到的问题领域包括：存储、网络、监控、编排和调度。Rancher开发、集成和贡献了在生产环境中运行容器所需的所有必要技术。

在Rancher Labs，集成和发布容器编排和调度框架，如：Docker Swarm 和 Kubernetes，同时还开发了常用的应用目录，企业用户管理，访问控制，容器网络和持久存储的技术模块。为开发者和运维者同时提供了完整的功能和优异的用户体验。使用 Rancher，企业不需要再为追赶和集成那些，存在于快速发展的容器生态系统中的无穷无尽技术而忧虑。与之相反，他们可以在一次部署 Rancher 之后就把精力都放到如何快速开发应用，和对业务的提高改善上。

详情参考Rancher的安装与配置。中国的云舒网络提供基于Docker的容器云商业服务和技术支持。

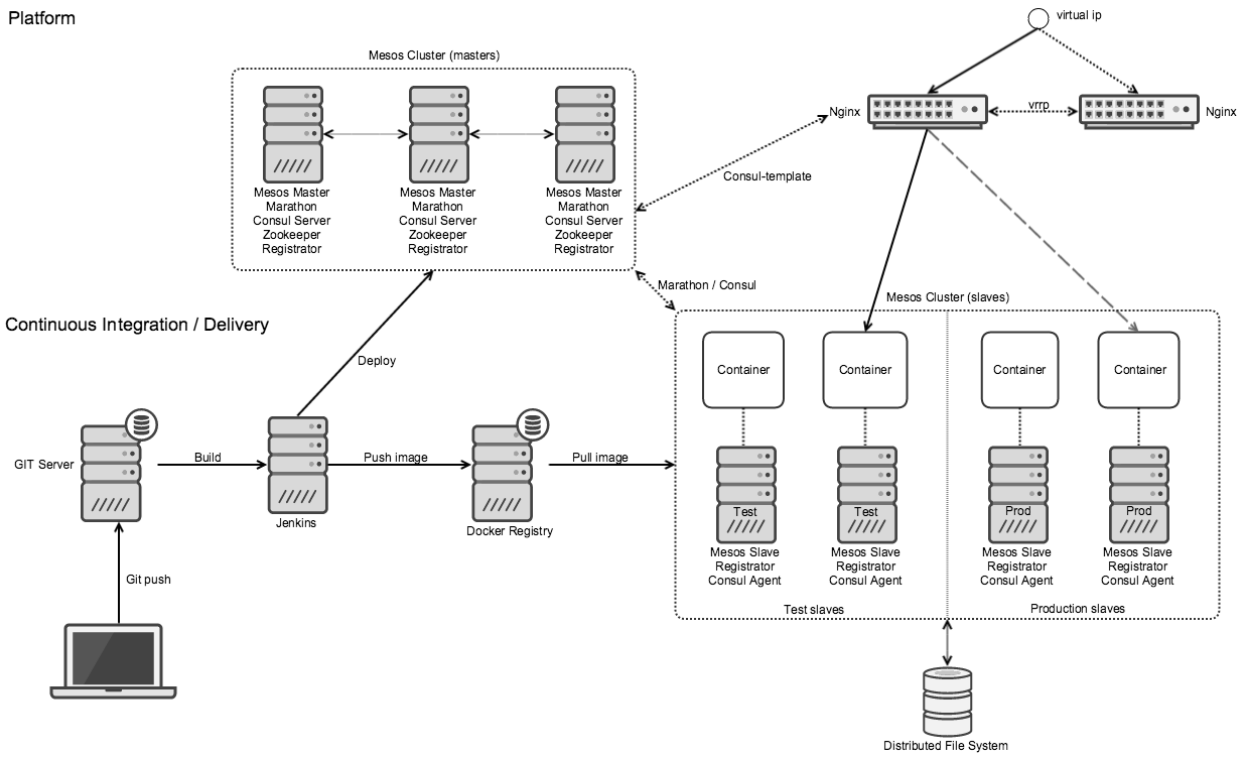
Mesos与自动化执行

Mesos集群资源管理

Apache Mesos是一个集群管理器，提供了有效的、跨分布式应用或框架的资源隔离和共享，可以运行Hadoop、MPI、Hypertable、Spark。开源中国组织翻译的Apache Mesos中文文档是非常好的资料。

Mesos的商业版由mesosphere提供，开发了一些生产级应用的工具，并提供了预先配置好的Docker镜像，可以快速投入运行。

自行安装和配置Mesos参考 在生产环境中使用Apache Mesos和Docker，部署架构如下。



Mesos主要特性:

- Master节点容错，使用ZooKeeper。

- 运行规模支持伸缩到上万个节点。
- 任务之间通过Linux容器技术隔离。
- 基于内存和CPU感知的多资源调度。
- Java, Python 和 C++ APIs。
- Web UI进行整个集群状态的管理。

Mesos可以对节点上的资源统一管理，在不同的应用间进行资源信息收集和多节点任务执行的调度。还可以通过Marathon和Chronos管理调度任务的执行，如日志收集、磁盘清理、数据整理、数据同步、数据挖掘等各种定期工作，并可以通过Docker的容器来执行，还可以调度定期的一个Spark计算任务或者对Spark的计算工作进行排队等等。

Marathon 服务编排

Mesos仅仅是适用于集群的管理，这意味着它可以隔离不同的任务负载。但是仍然需要额外的工具来帮助工程师查看不同系统上运行的工作负载。不然的话，如果某些工作负载消耗了所有资源，那么重要的工作负载可能就难以及时地获得资源。

Marathon（马拉松）是一个全新的框架，可以在单一的集群上运行不同的应用程序。它的设计宗旨就是让用户在同一组服务器之上，更智能地运行多种应用程序和服务——Hadoop、Storm，甚至一个标准的Web应用。Marathon出自于一家初创公司Mesosphere之手，这家公司主要就是想构建一个数据中心操作系统，不过这个系统是运行在Apache Mesos集群管理软件之上，这也是Twitter基础设施的重要组成部分。该公司的联合创始人是前Airbnb的工程师Florian Leibert（也曾在Twitter工作过）和Tobias Knaup。

特性:

- HA – 高可用支持。可以运行多个Marathon schedulers，但同一时间只有被选举出来作为主导者调度执行过程。
- Constraints - 约束，每一个机架、节点只运行一个应用实例等等。
- 服务发现和负载均衡，通过HAProxy或events API。
- 健康检查，检查应用运行的健康状况，通过 HTTP 或 TCP 检查。
- 事件订阅，使你通过 HTTP endpoint 接收通知，例如与外部的负载均衡器接入。
- Web UI，基于浏览器的管理界面。
- JSON/REST API，易于整合和编程控制。
- 基本 Auth 认证和 SSL 加密支持。
- 计量，基于JSON格式的计量数据。

Marathon是一个“元架构”，它可以让Mesos和Chronos变得更好用(参考 Mesos+Marathon+Docker部署)，随着Mesos一起运行，并且在运行工作负载的同时提供了更高的可用性，让用户可以添加资源以及自动的故障转移。

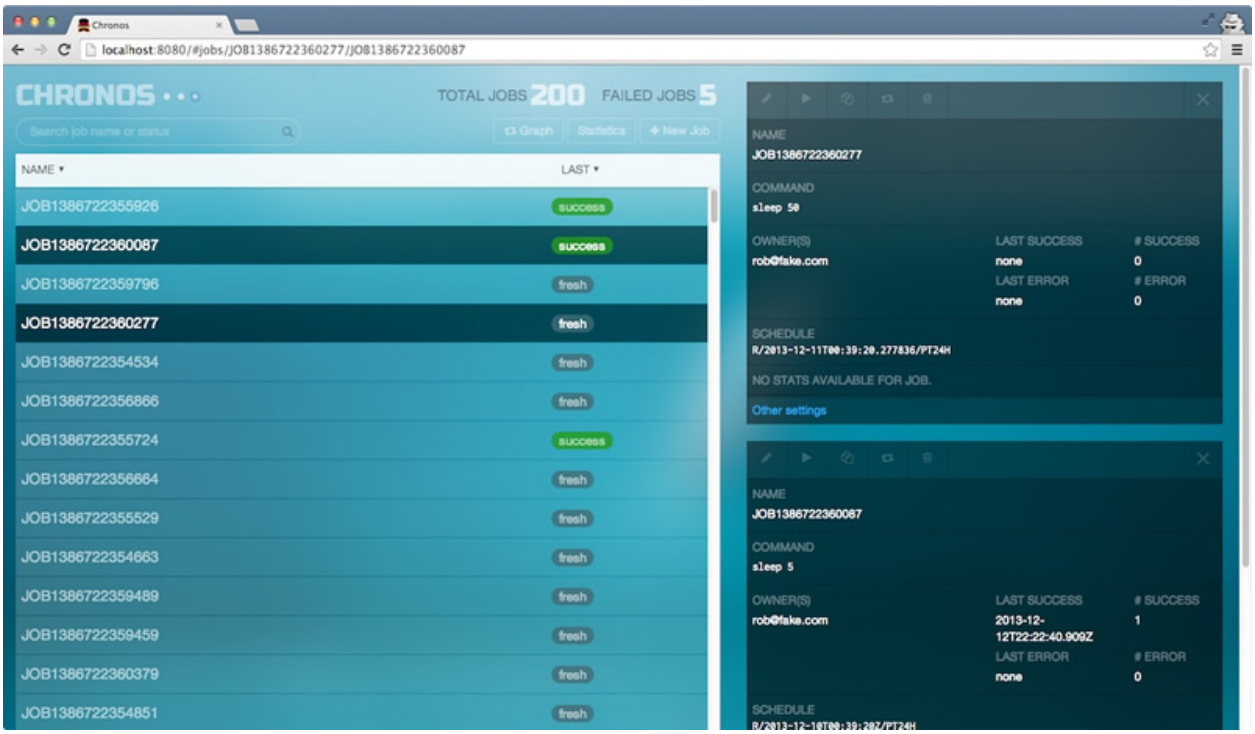
Chronos 任务管理

Chronos 是一个具备容错特性的作业调度器，可处理依赖性和基于ISO8601的调度。Chronos是由Airbnb公司推出的用来替代cron的开源产品。你可以用它来对作业进行编排，支持使用Mesos作为作业执行器，支持和Hadoop/Spark进行交互。可定义作业执行完成后的触发器。支持任意长度的依赖链。

特性:

- Web UI 管理器。

- ISO8601 重复任务执行规范。
- 处理依赖性，可以指定任务执行顺序。
- Job执行统计(如：时间占用百分比，成功／失败)。
- 容错(Hot Master)。
- 配置重试－Configurable Retries。
- 多个执行器 (i.e. Mesos Slaves)。



分布式计算与持续交付

大数据技术需要形成生产力，一个强大的持续集成和持续交付的软件工程环境和收缩性强的支持大规模分布式计算的基础设施是必不可少的。其中的软件是大数据分析项目中的重要组成部分，一方面提供硬件设施之上的基础运行环境，另一方面，软件又是数据发挥价值的“大脑”。

这里介绍将多种开源工具和在线服务组合形成持续交付系统实现分布式计算的架构和方法。GISpark集成了软件工程持续集成环境的系列工具，包括代码管理、代码审查、模块仓库、系统构建、即时交付、在线文档和Anaconda科学计算环境、Jupyter Notebook交互计算WebUI系统以及Spark和Hadoop大数据处理平台。使用GISpark的软件工具集，可以实现从代码到组件、集群化部署和分布式运行的全生命期过程，提供强大的智能计算能力，动态地、自动化地产出有效的分析结果。

持续集成与持续交付

持续集成与持续交付系统包括源代码版本管理、代码审查系统、软件仓库系统、持续集成系统、自动化测试与Bug跟踪、在线文档持续交付以及服务总线等子系统。

Git版本管理系统

Git是一个开源的分布式的版本管理系统，由Linux的创始人Linus发起和实现，现在由开源社区进行发展。与传统的版本管理不同的是，Git中是以开发者为中心而不是以管理者为中心的，每一个参与者都可以拥有一份完整的代码版本，可以进行离线编辑、在线提交、多分支合并等操作，适合用于支持大型的、多分支、多个节点的分布式软件开发模式，不仅可用于管理源代码，也可以用于管理版本化文档和小型的测试数据集等。

Git已经成为开源软件版本管理的事实标准，Linux的内核即采用Git进行管理，支持基于互联网的全球范围的多级软件协作。基于Git的在线代码管理服务包括Github.com和git.oschina.net等，也可以使用GitLab搭建自己专属的代码托管服务系统。通过将本地库、局域仓库和在线的代码库结合起来，开发者可以与全球各地的同行紧密协作，创建出超大规模的软件系统。

GitLab软件代码托管

Github是全球开源软件开发者的宝库，可以在线地获取各种功能强大的软件的源代码，加以修改并提交更新，使开源软件得以持续不断地改进。不仅得到大量的软件爱好者的拥趸，而且也得到了大量的知名软件企业的大力支持。

GitLab源代码托管与版本管理系统提供与Github类似的强大功能。但不同的是，GitLab让使用者可以搭建自己的源代码仓库服务系统，可通过Web界面进行访问公开的或者私人项目。能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，非常易于浏览提交过的版本并提供一个文件历史库。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

GitLab的客户端仍然使用Git进行操作，可以与Github等各种服务进行代码和文档共享。GitLab基于工程(Project)的源代码和相关资源的版本化管理Web服务站点，底层采用Git，可以实现分布式的代码编辑和合并，在各种内部团队和企业软件开发中得到越来越多的使用。

Sonar与代码审查

在编制大型软件的时候，需要大量的开发者参与，由于不同的软件开发者的知识、经验、习惯等方面的不同，代码往往出现大量的不一致、不规范的情况，从而导致缺陷和维护的困难，从而引发质量问题和严重后果。因此，代码审查(Code Review)和代码的相互评审是提高代码质量的重要手段（由于代码的关联性，到了测试阶段再去修复的时候往往已经为时过晚）。

传统的代码审查通过专家评审和相互评审方式进行，工作量大、成本很高。ReviewBoard可以帮助交互式地进行代码审查，提高代码审查的效率。它提供了在diffs里进行语法彩色编码，使得代码阅读变得简便（使用参考ReviewBoard代码评审实践总结）。此外，它还实现了基于Lucene的搜索来帮助管理较大的diffs组。Review Board在审查补丁（Patch）方面表现完美。一个叫做“提交审查”的工具被用来和SCM系统进行连接（当前支持SVN、CVS、Perforce、Git和Mercurial等），可以允许你请求一个将被提交的修改的审查。用户基础页面将这个过程描述如下：

1. 你在本地检出的代码上做了些可怕的修改。

- 你通过公布diff、编写描述和选择一些审查者来创建一个审查请求。
- 你在审查请求中点击“发布”并等待你的审查者看到它。
- 其他人看了你的审查请求，说道“这太可怕了，除非什么东西坏掉了。”
- 你根据他们的评论更新了你的代码。
- 你公布了更新后的diff，以及对他们评论的解答以指明你修改了什么（或者你要说明为什么你不打算按照他们的建议修改代码）。
- 大家看了你更新后的代码，并请你继续。
- 你将修改提交到仓库中。

- 你在审查请求中点击“设置为已提交”来从其他人的面板中移除你的审查请求。

Sonar是一个自动化的代码审查工具，可以帮助改善源代码的质量，内置了大量的语法、格式等错误检查的规则。开源中国集成了Sonar在线服务系统，其上托管的的项目可以进行代码质量的在线检查。Sonar通过预定义的编码规则，检查常见的编码错误和促进源代码编写的一致性，形成代码质量的数据报告。Sonar支持的语言包括：Java、PHP、C#、C、Cobol、PL/SQL、Flex 等。主要特点包括：(1)代码覆盖，通过单元测试，将会显示哪行代码被选中；（2）改善编码规则；（3）搜寻编码规则，按照名字，插件，激活级别和类别进行查询；（4）项目搜寻，按照项目的名字进行查询；（5）对比数据，比较同一张表中的任何测量的趋势。

Maven软件仓库

Maven软件仓库管理软件构建(Build)中用到的组件，可以集中管理、按需获取。Maven提供一个中心化的软件仓库服务，也可以搭建自己的私有软件仓库。Maven基于project object model (POM)的工程依赖模块定义，可以自动地获取需要的模块进行软件的自动化构建而无需预先安装。由于Maven中心仓库访问量大、速度较慢，可以使用 Sonatype Nexus 搭建私有组件仓库和中心仓库的本地镜像，参考安装方法。

Jenkins持续集成

随着软件开发复杂度的不断提高，团队开发成员间如何更好地协同工作以确保软件开发的质量已经慢慢成为开发过程中不可回避的问题。尤其是近些年来，敏捷（Agile） 在软件工程领域越来越红火，如何能再不断变化的需求中快速适应和保证软件的质量也显得尤其的重要。持续集成正是针对这一类问题的一种软件开发实践。它倡导团队开发成员必须经常集成他们的工作，甚至每天都可能发生多次集成。而每次的集成都是通过自动化的构建来验证，包括自动编译、发布和测试，从而尽快地发现集成错误，让团队能够更快的开发高质量的软件。

Jenkins持续构建系统将代码构建、持续集成与持续交付集成为一个整合性服务，实现从源码到运行的完整过程。最新的版本的Jenkins可以实现DevOps(开发运维一体化)的自动化部署，从而实现持续交付。采用Jenkins实现将分布式系统的组件部署到多个节点之中，这一过程可以通过虚拟机和Docker来完成，Jenkins的代码构建结果将被输出到Docker的构建环节，通过Dockerfile实现以Docker容器形式的最终交付件，并继续调用Docker Compose将一组服务立即投入运行。详细使用方法参考基于 Jenkins 快速搭建持续集成环境。

自动化测试与Bug跟踪

交付软件是容易的，交付高质量的产品和服务是非常困难的。建立自动化测试平台、测试用例库和Bug跟踪体系，通过自动化测试流程将软件交付过程形成闭环，并度量和跟踪其中的改变，进行变更控制，从而持续提升交付成果的质量。由于测试的设备和语言、环境的多样性，需要根据需求选择测试软件，可以参考这里。

文档的持续交付

软件具备清晰、详细的文档对于使用者和软件的升级维护都具有重要的意义，但是文档的编写需要消耗巨大的工作量。随着软件的修改，原来的文档可能很快不再适用，是软件开发中存在的普遍问题。基于Git + Sphinx + ReadTheDocs的工具组合，可以形成一套版本化的在线文档发布系统，简化文档更新的步骤，加快最新版本文档发布的过程。

Git可以提供版本化的文档管理，支持多人同时协作，可以采用Markdown(*.md)*格式或者*IPython Notebook*的.ipynb格式。

- 可以将*.md文件组装为Gitbook的文档系统。
- 通过NBviewer进行Notebook文档预览。或者，

- 通过Sphinx的NBSphinx插件将ipynb动态地转换为rst文档，并组装到Sphinx文档系统。
- 然后通过ReadTheDocs在线发布出来。

Sphinx是一个文档组装系统(中文使用手册)，可以将多种类型文档(rst,md,ipynb...)组装为一个文档，并自动产生索引。Sphinx默认使用的格式是rst (reStructuredText)，采用类似于Markdown的格式，但支持更多的精细排版操作。通过扩展插件，可以支持更多的格式，并输出为Latex/PDF等电子文档格式，适合多种设备上的阅读。Sphinx支持多种风格模版，最初用于Python开发的自动化文档生成，其本身亦采用Python开发，非常容易通过插件扩展其功能，或者通过配置文件调整文档生成的各种参数。

ReadTheDocs是一个免费的在线文档发布系统，可以将Sphinx的文档项目直接发布到“文档云”上。大量的软件使用手册已经采用该平台发布（如Jupyter）。只需要注册一个账号，使用Sphinx编写文档项目，并提交到Github，再将ReadTheDocs的项目与Github账号和Git库连接，系统即可自动获取文档内容、编译并产生最终的HTML发布文档。发布的文档将自动给出*.readthedocs.io的域名，可以在浏览器中直接访问，或者将其重定向到自己的域名。如果编译中需要安装其它组件，在根目录下创建一个requirement.txt文件，将组件名称写入再提交即可，与Python的安装模式是完全一样的。

企业服务总线

随着信息化程度的提升，组织机构中的各种服务将会快速增加，需要进行树立和统一管理，企业服务总线就应运而生。各种资源和服务需要连接起来，方便查找、使用和维护，同时可以用于负载均衡、故障转移、失效恢复等工作。

企业服务总线主要有三种技术形式，SOAP、REST和消息总线。基于SOAP的服务总线主要用于传统的服务集成，正在被基于REST的服务架构所取代，很多企业服务总线软件可以同时这两种协议，这两种协议主要用于中心式的集中管理架构。

基于消息总线的服务总线提供对等的体系架构，非常适合松散耦合的大规模、分布式系统，如物联网、社交软件中数据流处理以及多种软件之间分布式的即时协同等等，近年来得到快速发展，正在逐步取代传统的中心式架构。消息协议主要有已经成为国际标准的AMQP/MQTT，以及JMS或者自定义的格式(如，coAP是基于UDP的轻量级物联网协议，不过目前使用还不够广泛)。消息服务器可以采用RabbitMQ/ActiveMQ/Mosquitto等，RabbitMQ可以同时支持AMQP和MQTT协议。Paho是基于MQTT的客户端，支持多种开发语言和小型设备通过MQTT协议进行互联。

目前Amazon/Aliyun等云服务商都推出了基于云的消息总线服务，可以支持多种消息协议，只需要申请一个账号即可立即开通服务，可以在整个互联网上使用。大多数时候，需要将云服务和局域网服务结合起来。原则上，本地服务应通过网关连接云服务，从而避免本地端口暴露在互联网上带来的安全性问题。

Anaconda科学计算软件包

Anaconda是一个集成了Jupyter Notebook以及NumPy、SciPy、Pandas、Matplotlib等科学计算与数据处理的Python分发包（使用参考）。continuum公司还开发了企业版，可以让研究人员共享数据和运行环境，从而实现基于网络的团队协同分析。

Anaconda采用conda进行软件包的管理，可以实现安装、列表、搜索等各种功能，也可以在Anaconda Cloud中创建自己的软件包。Conda的功能类似于pip和Virtualenv的组合，可以创建和管理运行的虚拟环境从而避免不同版本引起的冲突问题（安装在Anaconda3/envs目录下），conda也提供了完整的安装和卸载运行组件的功能，参考conda的使用，同时Anaconda也支持通过pip进行软件组件的安装。

Anaconda支持Spark分布式计算环境和TensorFlow机器学习引擎，参考Anaconda上安装TensorFlow/Spark，实现jupyter远程访问以及使用Anaconda集成IPython、Spark和TensorFlow、Orange。Orange是支持可视化流程设计的多算法机器学习计算框架，可以通过Conda中安装和使用，其核心算法也可以在Jupyter Notebook中调用。

Jupyter Notebook是Anaconda的重要组件，源于**‘IPython Notebook <>‘__**项目。Jupyter可以创建出“可运行”的Web文档，非常便于Python算法学习和研究交流使用，已被大量的科研、数据分析、金融分析人员使用，并发展出了大量的增强工具。在最新的Jupyter版本中，不仅可以运行python和R代码，

还可以通过Magics操作符（%和%%）同时运行shell、ruby等代码。Jupyter Notebook文档可以被转换为HTML/Latex/PDF等格式，在多种设备上浏览。

创建Jupyter NoteBook Server的步骤如下：

```
#创建jupyter配置文件，原来的IPython的一些教程要退休了，按这个：
jupyter notebook --generate-config

#修改jupyter配置文件，在~/.jupyter/下。
gedit ~/.jupyter/jupyter_notebook_config.py

#设置访问密码。
#需要使用加密的字符串，参考jupyter_notebook_config.py文件中的说明。

#设置服务器的IP地址和端口绑定。
#搜到这一行：c.NotebookApp.ip = 'LOCALHOST'
#将LocalHost改为自己的机器名、域名或者IP地址。
#不知道IP地址的，用ifconfig查看（linux上）如果设为"*"，则允许所有地址。

#按照上面类似的方法，修改端口。

#如果通过不同机器进行访问(无GUI的服务器，如云服务器必须设置)。
#将browser允许一栏设为Disable，否则启动时可能导致挂起。

#其它参数，根据需要设置。
#保存，退出gedit。

#启动Jupyter
jupyter notebook

#打开浏览器，远程访问
http://myhost_ip:8888
```

Spark分布式计算平台

Spark是一个高性能分布式计算平台，通过基于内存的分布式数据对象RDD的使用，大幅度提高了MapReduce算法的性能，可以增强Hadoop等大数据系统的性能，扩展了流处理、拓扑分析、机器学习等分布式算法框架，同时能够兼容各种已有的存储系统。参见更多介绍。

Spark通过PySpark模块支持基于Python的数据分析，同时也可以与Jupyter Notebook一起使用，创建优雅的数据探索工具和易于理解的数据分析教程，本文档的大部分教程亦使用该工具和方法创建。更多内容，请查看Spark学习资源。

Spark大数据分析框架的组成

Spark大数据分析框架的核心部件包含RDD内存数据结构、Streaming流计算框架、GraphX图计算与网状数据挖掘、MLlib机器学习支持框架、Spark SQL数据检索语言、Tachyon文件系统、SparkR计算引擎等主要部件。这里做一个简单的介绍。

** (1)RDD内存数据结构 **

大数据分析系统一般包括数据获取、数据清洗、数据处理、数据分析、报表输出等子系统。Spark为了方便数据处理、提升性能，专门引入了RDD数据内存结构，这一点与R的机制非常类似。用户程序只需要访问RDD的结构，与存储系统的数据调度、交换都由提供者驱动去实现。RDD可以与Haoop的HBase、HDFS等交互，用作数据存储系统，当然也可以通过扩展支持很多其它的数据存储系统。因为有了RDD，应用模型就与物理存储分离开来，而且能够更容易地处理大量数据记录遍历搜索的情

况，这一点非常重要。因为Hadoop的结构主要适用于顺序处理，要翻回去反复检索数据的话效率就非常低下，而且缺乏一个统一的实现框架，由算法开发者自己去想办法实现。毫无疑问，这具有相当大的难度。RDD的出现，使这一问题得到了一定程度的解决。但正因为RDD是核心部件、实现难度大，这一块的性能、容量、稳定性直接决定着其它算法的实现程度。从目前看，还是经常会出现RDD占用的内存过载出问题的情况。

** (2)Streaming流计算框架 **

流是现在推特、微博、微信、图片服务以及物联网、位置服务等等的重要数据形态，因此流计算正显得前所未有的重要。流计算框架是所有互联网服务商的核心基础架构，Amazon、Microsoft都已经推出了Event消息总线云服务平台，而facebook:raw-latex:*twitter*等更是将自己的流计算框架开源。Spark Streaming专门设计用于处理流式数据。通过Spark Streaming，可以快速地将数据推入处理环节，犹如流水线一样进行快速的加工，并在最短的时间反馈给使用。

** (3)GraphX图计算与网状数据挖掘 **

物理网络的拓扑结构，社交网络的连接关系，传统数据库的E-R关系，都是典型的图（Graph）数据模型。Hadoop主要适用于“数据量”很大的场合，对于关系的处理几乎没有支持，Hbase也是非常弱的关系处理能力。图数据结构往往需要快速多次对数据进行扫描式遍历，RDD的引入使Spark可以更高效地处理基于图的数据结构，从而使存储和处理大规模的图网络成为可能。类似的专用于图的系统还有neo4j等。GraphX相对于传统数据库的关系连接，可以处理更大规模、更深度的拓扑关系，可以在多个集群节点上进行运算，确实是现代数据关系研究的利器。

** (4)MLlib机器学习支持框架 **

通过把机器学习的算法移植到Spark架构上，一方面可以利用底层的大规模存储和RDD的数据快速访问能力，还可以利用图数据结构和集群计算的处理能力，使机器学习的运算可以在大规模的集群系统上展开，即大力拓展了机器学习算法的应用能力。

** (5)Spark SQL数据检索语言 **

这个跟基于Hive的实现有些类似，但是基于RDD理论上能提供更好的性能，同时能更方便处理如join和关系检索等操作。这个被设计为与用户交互的一个标准化入口。

** (6)Tachyon文件系统 **

Tachyon是基于内存的文件系统，可以使用HDFS等多种文件系统做持久化存储。已更名为Alluxio，通过将数据缓存到内存，从而大幅度提高了大尺寸文件的访问性能。

** (7)SparkR计算引擎 **

将R语言的能力应用到Spark基础计算架构上，为其提供算法引擎。

Spark安装与部署

安装Spark的工作，包括：安装Scala语言支持环境、安装SBT编译环境、安装Hadoop(可选)、按照需求配置Spark的运行模式。使用虚拟机、Docker来运行Spark或者直接使用Amazon/DataBricks的云端集群环境，可以大幅度简化安装和配置工作，快速建立自己的分布式计算集群。

** Spark安装配置 **

Spark的运行依赖于Scala。首先下载Scala，然后配置 scala 环境变量：

```
vim /etc/profile
export SCALA_HOME=/home/hadoop/software/scala-2.11.4
export PATH=$SCALA_HOME/bin:$PATH
```

从http://spark.apache.org/downloads.html 下载 Spark的版本，解压缩、配置Spark环境变量，运行即可。详细的安装方法参考Spark 伪分布式 & 全分布式 安装指南。

** PySpark安装配置 **

注意：如果要用pyspark，则需要设置 python 相关的 spark 包路径：

```
vi .bashrc
export PYTHONPATH=$SPARK_HOME/python/:$SPARK_HOME/python/lib/py4j-0.8.2.1-src.zip:$PYTHONPATH
```

请根据你的机器环境，设置上面的路径的环境变量。

** Spark源码编译 **

Spark能同Hadoop进行交互，而Hadoop的厂商比较多有很多商业版。Spark官方提供的安装包不一定和我们的Hadoop集群版本相同，如果不相同就有可能出现莫名其妙的错误。这时，我们手工指定相应版本进行编译是最好选择，请参阅详细参考，详细的编译过程参阅Spark官网指南。

Spark编译有三种方式：SBT、MAVEN、make-distribution.sh。SBT、MAVEN两种方式打出来的包比较大，不适合部署使用，通常使用第三种方式打包。

- 直接进Github，复制源码：https://github.com/apache/spark 。
- SBT编译：`sbt/sbt clean assembly`
- MAVEN编译。
  - 由于MAVEN工具默认的内存比较小，需要先调大其占用的内存上限：
    `export MAVEN_OPTS="-Xmx2g -XX:MaxPermSize=512M -XX:ReservedCodeCacheSize=512m"`
  - 打包: `mvn clean assembly:assembly`
- make-distribution.sh构建安装包。该脚本会使用MAVEN进行编译，然后打成一个tgz包。
  - 脚本的使用方法：`./make-distribution.sh --help`
  - 打包：`./make-distribution.sh --tgz --with-tachyon`

Spark的运行模式

Spark本地运行模式

控制台：pyspark –master local[4]，打开浏览器，访问 http://localhost:4040 可查看运行状态。

PySpark缺省使用的Python环境是2.7，如果希望使用python3，可以设置环境变量`PYSPARK_PYTHON=python3`。在控制台首先运行：`export PYSPARK_PYTHON=python3`，或者将上述语句加入pyspark的脚本中，或者加入~/.bashrc文件中，或者加到.profile。控制台窗口需要重启才有效。

Spark集群运行模式

这里介绍Spark的`Standalone`模式，即只有或主要运行Spark的集群。

** A.启动Master **

启动Master: start-master.sh，打开浏览器进入Master管理页面： http://localhost:8080 ，可以看到启动Master的监听URL在：

```
URL:          spark://supermap:7077
REST URL:     spark://supermap:6066(cluster mode)
```

** B.启动Slave **

启动Slave: start-slaves.sh –master spark://supermap:7077 上面的–master参数为启动的master服务地址。刷新Master的管理网页，可以看到新增加的worker。

** C.启动控制台客户端 **

连接集群: pyspark –master http://supermap:7077/

```
刷新Master的管理网页，可以看到新增加的Application.
点击PySparkShell，进去可以看到所运行的job等信息。
```

** D.提交批处理任务 **

使用spark-submmit，具体参考：http://spark.apache.org/docs/latest/submitting-applications.html 。

注意，提交任务既可以进入交互模式，也可以使用deploy-mode。交互模式下，客户端一直保持连接，并即时获得运行的信息。而deploy-mode提交后由后台运行，关闭客户端不影响任务的执行，运行信息需要通过日志文件去查看。

Spark 协同运行模式

如果在集群上会同时运行很多任务，Spark的调度器与别的调度器会竞争资源，就需要更高级的调度器来进行协调，一般使用mesos或YARN。这里不再详述，可参考：http://spark.apache.org/docs/latest/running-on-mesos.html 。

Spark支持部署在虚拟机或者Docker上运行。

Spark 使用入门

Spark支持多种方式调用，包括：开发Spark扩展组件、使用Scala控制台创建计算任务、使用PySpark进行交互式数据探索、SQL接口，通过Shell提交计算任务等等。Spark所提供的SQL接口以及DataFrame API，可以让拥护像传统的数据库一样进行操作大型分布式数据集，参见Spark SQL和DataFrame指南。

Spark还在快速发展之中，Spark 2.0架构预计在2016年中推出，将带来更好的SQL支持、更好的性能和更好的软件生态。GISpark亦将为Spark注入先进的时空分析方法和地图可视化方法，拓展Spark的空间数据处理和分析能力。更多的Spark快速入门安装方法，访问Spark 入门（Python、Scala 版），以及Spark学习资源。

Python 使用教程

在大数据分析中，Python已经成为必不可少的工具。Python因为其简单、灵活、易用、兼容性好，具有非常丰富的各种各样的工具软件库，在数据探索、金融分析、科学研究、编程教学等方面得到了广泛的应用，Hadoop和Spark等都具有Python接口，大量的在线服务也都提供了Python API，可以方便地进行编程、实现自动化操作，直接进行数据获取和在线分析。

Python(查看全部源码)完全由社区贡献的模式推动，发展非常迅速。参与社区和参加PyCon大会、PyData全球技术会议等活动都是了解最新的信息和提高编程和数据分析技能的捷径。欢迎查看PyCon 2016 in Portland的信息。

Python

使用Python进行数据分析不需要专业的编程知识和长期的逻辑训练，可以通过这里的Python基础教程快速学习。在Notebook资源大全可以获得大量的从Python基础使用到数据处理、数据建模、并行计算、机器学习、深度学习等大量资源，采用notebook编写，也可以直接下载到本地进行练习，通过下载Git资源到本地可以快速将其下载到本地存储中慢慢学习。

Jupyter

Jupyter是一个优雅的Web编程环境，将Shell、Python、Ruby等多种脚本语言集成到一个执行环境中，可以混合在一起执行、设置，可以在多语言运行环境之间交换变量，使用Jupyter魔法操作符即可。Jupyter可以通过Anaconda快速安装，参考安装方法，以及阿里云服务器安装Jupyter及conda/Spark技巧。

Jupyter的核心组件是Notebook，源于IPython项目(参见'IPython源码 <https://github.com/ipython>'__，IPython文档)，可以通过Web界面进行Python程序的编写，并可以转换为PDF等文档格式，可以产生漂亮的"可运行"文档，便于通过网络共享和协作式研究。详情参阅Jupyter文档。

NumPy和SciPy

NumPy和SciPy是著名的基于Python的科学计算软件库，提供"大数变量"、快捷矩阵运算和大量的科学函数库，在数学、物理、化学、生物等基础科学和医疗图像、基因研究、引力波探测、信号处理等科研活动中都得到了大量的应用，也是Anaconda的基础软件包。

Pandas

数据分析常用的是Python列表类对象，用于进行各种数据对象的处理。用于时间序列、二维表、数据立方分析的Pandas内存数据表功能非常强大，可以在内存中进行常规的数据库表的查询、修改、运算等操作。目前的Pandas主要是单机版本，在金融领域已经得到广泛的使用，一些社区正在开发可用于集群运算的Pandas版本，Spark的DataFrame以及提供了类似的分布式计算能力。

Blaze

Blaze系列软件(源码)提供在集群上运行类似于NumPy和Pandas的功能。包括Blaze数据访问接口、DataShape数据类型系统、Odo数据迁移系统、DyND内存动态数组、Dask.array多核/磁盘上NumPy arrays、Dask.DataFrame多核/磁盘上的Pandas DataFrame实现等子项目。Blaze与NumPy和Pandas不同的是，Blaze直接将相应操作映射到多种数据存储系统，提供动态的数据访问能力。Blaze系列软件面向分布式的内存数据处理，目前仍然在快速的发展中。

Python并行处理

Python支持多种并行处理方法，包括MPI接口、ParallelPython、iPyParallel、Cuda、OpenCL以及PySpark(见本文下一节)。这里主要介绍ipyparallel，可以在Jupyter中直接启用，现在是IPython工程的一部分。

** 安装ipyparallel **

ipyparallel安装方法: `pip install ipyparallel`

在Jupyter Notebook的IPython Clusters页面上启用ipyparallel: `ipcluster nbextension enable`

关闭ipyparallel功能: `ipcluster nbextension disable`

配置notebook server，查看 `jupyter_notebook_config.py` 相应的参数配置。

** 运行ipyparallel **

启动集群: `ipcluster start`

在Python中使用:

```
import os
import ipyparallel as ipp

rc = ipp.Client()
ar = rc[:].apply_async(os.getpid)
pid_map = ar.get_dict()
```

更多的信息参考：https://ipyparallel.readthedocs.io/en/latest/

PySpark使用指南

PySpark是Spark的客户端访问库，可以支持通过Python编写脚本，然后通过Spark在分布式环境中执行。使用方法参考 快速体验Spark，PySpark教程，Spark快速入门，Spark SQL和DataFrame使用，PySpark SQL API参考，使用GraphFrames进行飞一般的图计算。PySpark的安装和配置参考 Spark 安装配置，与Jupyter Notebook一起使用Spark，Spark配置参数。

地理空间信息服务平台

应用服务系统搭建

iCloudManager,iPortal,iServer,iExpress.

客户端应用及开发

iClient iMobile iDesktop

地理空间信息数据处理

本章介绍地理空间数据的GeoJSON格式规范和使用方法，Python中进行地理几何数据处理的GeoPandas、Shapely、GDAL库，以及专业GIS的脚本引擎-GIScript。最后，介绍通过使用这些技术进行OpenStreetMap、SRTM和TM等开放的免费数据的处理过程和方法。

地理空间数据格式

GeoJSON 规范

GeoJSON是一种对各种地理数据结构进行编码的格式。GeoJSON对象可以表示几何、特征或者特征集合。GeoJSON支持下面几何类型：点、线、面、多点、多线、多面和几何集合。GeoJSON里的特征包含一个几何对象和其他属性，特征集合表示一系列特征。

GeoJSON 编程

Python的词典和列表、元组等数据结构可以直接与JSON映射。GeoJSON完全遵循JSON规范，因此GeoJSON很容易通过Python编程操作，并且与Shapeply、Fiona、GeoPandas组合使用，完成地理空间对象和文件的I/O操作。GDAL和很多商业GIS软件也都支持GeoJSON，可以非常方便地进行数据交换，尤其是在网络服务中，有逐渐取代其它格式的趋势。

GeoCSV 格式

Hadoop和Spark为了便于分布式处理中数据切分，都主要按照分行处理方式设计。因此，空间数据经常保存为空间数据CSV格式，方便在分布式环境下使用。CSV格式数据域一般以"分隔和表格符\t两种常见格式，而几何对象一般采用WKT(well known text)、WKB(WKT的二进制编码)或GeoJSON格式存储，每行以回车结束。

几何对象与地图绘制

Matplotlib绘制地图

Matplotlib是Python上广泛使用的绘图库，支持在Jupyter Notebook中绘图，安装basemap包后，可以直接使用Matplotlib绘制地图。

GeoPandas内存数据集

GeoPandas内存数据集派生于Pandas数据类，增加GeoSeries对象，可以存储Geometry地理空间几何对象。GeoPandas的Geometry对象采用GeoJSON格式存储，非常便于在Python中操作，详细请参考GeoPandas.org。

Shapely几何对象运算

Shapely提供地理几何对象数据结构、编解码及对象之间的几何代数运算。Shapely工程源代码可以直接下载、编译、安装，文档参见：http://toblerity.org/shapely/project.html。

Fiona几何对象操作

Fiona库提供几何对象操作和Shape文件的读写功能，基于OGR实现的简洁的Python API对象接口，详情可以查看文档和源代码。

GDAL库安装与使用

GDAL是一个广泛使用的地理空间数据对象和文件读写的支持库，其本身完全开源而且被整合到了很多开源软件和商业GIS软件中，详情查看GDAL库安装与使用，可以查看和下载GDAL的Geometry使用教程进行研究和练习。

GIScript安装与使用

GIScript是专业GIS脚本引擎，提供数据交换、数据管理、制图和分析等丰富的功能，支持在Jupyter notebook和Spark环境下使用，支持Linux和Windows操作系统，请参考GIScript安装与使用。

OSM开源地图数据处理

OSM是OpenStreetMap的众包开放地图，原始数据采用xml格式进行存储。这里提供的教程包括：OSM数据下载与结构分析、Overpass在线API使用、OSM流式转为行存储JSON,所生成的行式存储数据可以在PySpark中数据查询构建Geometry和进行更高级的组合分析。

SRTM地形高程数据处理

这里介绍SRTM地形高程数据处理方法，将实现数据的下载、转为GeoTif、设置投影等基本操作，以及进行渲染立体图、解译等高线等高级功能。

影像数据处理与挖掘

Python中有丰富的影像处理模块，这里以TM影像处理为例，进行影像处理方法的介绍。影像的高级处理、土地分类、对象识别、机器学习等将在后续的专题中另行介绍。

Rasterio是读写地理空间信息栅格数据集的工具。Rasterio采用GDAL进行文件I/O和栅格文件格式的操作。典型功能是接受和返回Numpy的ndarray数据对象。Rasterio设计为提高地理空间栅格文件处理的生产率，让其处理变得更有趣。

地理制图与空间可视化

静态制图与分块输出

动态Web地图

高级三维可视化

高级空间分析

地理选址与优化

快速地理编码

地理空间网络

地理网络分析

社交网络分析

复杂网络分析

实时流处理

框架与流程

处理与存储

算法与过滤

多维时空分析

空间图谱

时间序列

时空演变

机器学习框架

TensorFlow

Keras

模型构建与共享

科研的大数据基础设施

基于云计算的协同研究

究的宝库。因为数据提供者可能会变更数据和提供的许可政策，请注意鉴别数据的有效性和适用性，以免违反许可条款或得到不准确的结论。

- **本列表仅供参考，供技术研究使用，不对其有效性、准确性以及合法性承担任何责任。**

开放数据源（一），全球地理、自然、经济数据等。

- 世界银行,数据网站: data.worldbank.org
- gapminder,编译数据源,包括世界卫生组织和世界银行覆盖经济、医疗和社会统计数据。
- 美国中央情报局,世界概况,包含267个国家的历史信息、人口、经济、政府、基设施和军事等信息。
- 国家气候数据中心,巨大的环境、气象和气候数据集来自美国国家气候数据中心。世界上最大的气象数据的归档。
- OpenStreetMap,全球范围的免费地图数据，每周动态更新。
- 亚马逊网络服务公共数据集,巨大的公共数据资源,包括1000基因组计划,试图构建人类遗传信息的最全面的数据库和NASA的地球的卫星图像的数据库。
- 谷歌趋势,统计搜索量(搜索)总额的比例对于任何给定的期限,自2004年以来。
- Facebook Graph 虽然大部分的信息用户的Facebook的个人资料是私人的,很多不是——Facebook提供图形API的方式查询大量的信息,其用户乐于分享与世界(或无法隐藏,因为他们没有了隐私设置工作)。
- DBPedia,维基百科是由数百万块的数据,在每个主题在阳光下结构化和非结构化。DBPedia是一个雄心勃勃的项目目录,并创建一个公共、自由可分配的数据库允许任何人来分析这些数据。
- 社交媒体数据,Topsy公共微博提供了一个可搜索的数据库2006年以及几个工具来分析对话。
- Likebutton,矿山Facebook的公共数据——在全球范围内,从您自己的网络给人们“喜欢”的概述。
- 纽约时报,索引归档的新闻文章回到1851年。
- 开放数据库,community-compiled数据库结构化数据的人,地方和事情,与超过4500万个条目。
- 百万歌曲数据集,元数据超过一百万首歌曲和乐曲,亚马逊网络服务的一部分。
- 欧盟开放数据门户,基于来自欧盟机构的数据。
- SDMX,欧洲央行,
- 美联储
- 谷歌财经,40年的股票市场数据,实时更新。
- 金融量化:pydatastream
- 中国金融,TuShare
- NASDAQ Data Store：提供市场数据。
- 美国、欧盟、加拿大、CKAN以及其他的公开政府数据
- 英国,英国政府的数据,其中包括英国国家参考书目元数据(自1950年以来英国所有书籍和刊物)。
- Google Books ngram,搜索和分析全文的数以百万计的图书数字化,作为谷歌图书项目的一部分。
- 美国联邦政府数据网站, 美国政府承诺可以免费在线所有政府数据,第一阶段已有各种数据。
- 美国人口普查局,丰富对美国公民的生活人口地理数据和教育的信息。
- Healthdata.gov,125年的美国医疗数据包括claim-level医疗数据、流行病学和人口统计数据。
- 国民健康和社会保健信息中心,来自英国的国家卫生服务的健康数据集。

开放数据源（二），科研、论文，生物、网址、机器学习等。

编译自http://www.bigdata-madesimple.com/70-websites-to-get-large-data-repositories-for-free/

- Wikipedia:Database： 向感兴趣的用户提供所有可用的内容的免费副本。可以得到多种语言的数据。内容连同图片可以下载。
- Common crawl：建立并维护一个所有人都可以访问的开放的网络。这个数据保存在亚马逊s3bucket中，请求者可能花费一些钱来访问它。
- EDRM File Formats Data Set：由381个文件夹200种文件格式组成。
- Apache Mahout TLP：项目创建一个可扩展的机器学习算法。Mahout有许多免费的和付费的语料库语料。
- StatLib,卡内基梅隆大学数据档案。
- STATOO Datasets part 1和 STATOO Datasets part 2，Time Series Data Library。
- Visual Analytics Benchmark Repository。
- UCI KDD Database Repository：适用于机器学习和知识发现研究的大数据集。
- UCI Machine Learning Repository.
- UCR Time Series Data Archive：提供数据集、论文、链接和代码。
- EDRM Enron Email Data Set v2：由安然公司邮件信息和附件组成，存在两组可下载的压缩文件中：XML和PST。
- ClueWeb09：用来支持信息检索和相关人类语言技术研究的资料库。它包含了从2009年1月到2月间收集的大约10亿个网页，包含10种语言。资料库被若干TREC会议的追踪检测使用。
- DMOZ：最大的、最全面的人工编辑的开放式网站目录。它收集了不同类型的网站链接。Dmoz是互联网搜索引擎的一个主要来源。
- theinfo.org：这是一个大数据集网站，在这里学者、设计师、艺术家等可以交流技巧和窍门，一起开发和共享工具，并开始整合他们独有的项目。
- Project Gutenberg：提供超过36000免费电子书的下载，可以下载到个人电脑、Kindle, Android, iOS or 或其他便携式设备。
- Million song data set：与tracks 和艺术家有关的数据。
- AWS (Amazon Web Services) Public Data Sets：提供了可以无缝融入AWS（亚马逊网络服务）云应用的公共数据集的集中存储库。
- GeoDa Center：地理和空间数据。

BigML big list of public data sources.

- Bioassay data：研究文章“生物测定数据的虚拟筛选”，由Amanda Schierz编写，有21个生物测定数据集（活性/非生理活性成分），可以下载。
- Bitly 1.usa.gov data：匿名点击链接。
- Canada Open Data：有许多政府和地理空间的数据集的试点项目。
- Causality Workbench：数据存储库。
- Corral Big Data repository：在德克萨斯高级计算中心，提供以数据为中心的技术。

Data Source Handbook:公开数据指南。

- Data.gov/Education：对于教育数据资源的主要指南，包括高价值的数据集、数据可视化、课堂资源、创建自公开数据的应用程序以及其他。
- DataMarket：可视化的世界经济、社会、自然和工业，拥有来自联合国，世界银行，欧盟统计局和其他重要数据提供者的一亿时间序列。
- Datamob：可以很好利用的公开数据。
- DataSF.org：可向City & County of San Francisco, CA.购买的数据集信息交流中心。
- DataFerrett：一个用来访问和使用The Data Web的数据挖掘工具，许多网上美国政务数据集的集合。
- EconData：大量经济学的时间序列，由许多美国政府机构编制。
- Enron Email Dataset：来自大约150个用户的数据，这些用户大多数是安然公司高级管理人员。
- Europeana Data：包含2000万文字，图片，视频开放的元数据，以及由欧洲数位图书馆收集的声音，对于欧洲文化遗产内容值得信赖的、全面的资源。
- Europeana Data：FEDSTATS：一个美国统计资料的综合资源以及更多。
- FIMI repository for frequent itemset mining：工具和数据集。
- Financial Data Finder at OSU：大型财务数据集目录。
- GDELT:关于事件、位置和音调的全球数据，被英国卫报形容为“生命、宇宙和一切的大数据历史”。
- GEO (GEO Gene Expression Omnibus)：一个支持MIAME兼容数据提交的基因表达/分子丰度信息库，用于基因表达数据的浏览，查询和检索。
- Google ngrams datasets：来自数Google扫描的百万书籍文本。
- Grain Market Research：财务数据，包括股票、期货等。
- Hilary Mason research-quality Big Data sets: 收集许多文本和图片数据集.
- HitCompanies Datasets：HitCompanies随机取样的1万个英国公司全面的数据，采用人工智能/机器学习进行自动更新。
- ICWSM-2009 dataset：包含2008年8月1日到10月1日之间的4400万个博文。
- Infochimps：一个数据开放的目录和集合，允许分享、出售和下载关于任何内容的数据。
- Investor Links：包含财物数据。
- KDD Cup center：数据、工作表和结果。
- Kevin Chai list of datasets：文本、SNA和其他领域。
- KONECT：科布伦茨网络收集，拥有大量各种类型的网络数据集，以便在网络挖掘领域进行研究。
- Linking Open Data 工程，免费向所有人提供数据。
- MIT Cancer Genomics gene expression datasets and publications：来自麻省理工Whitehead Center用于基因组研究。
- ML Data：欧盟Pascal2网络数据储存库。
- National Government Statistical Web Sites：来自大约70个网站的数据、报告、统计年鉴、新闻和其他，包括非洲、欧洲、亚洲和拉丁美洲的国家。
- National Space Science Data Center (NSSDC)：美国国家航空航天局的数据集，包含行星探索、空间和太阳物理学、生命科学、天体物理学以及其他方面。
- Open Data Census：评估世界各地的开放数据的状态。

- OpenData from Socrata：允许访问超过10000个数据集，包括商业、教育、政府和娱乐。
- Open Source Sports：大量运动数据库，包括棒球、足球、篮球和曲棍球。
- Peter Skomoroch dataset Bookmarks PubGene(TM) Gene Database and Tools：基因组有关的出版物数据库。
- Quandl, a collaboratively curated portal to millions of financial and economic time-series datasets.
- qunb：一个用来发现和可视化的数据资料的平台。
- Robert Schiller data：住房建筑、股票市场和更多的来自于他的书 Irrational Exuberance的数据
- SMD: Stanford Microarray Database,存储来自微阵列实验的原始的和标准的数据。
- Jerry Smith dataset collection：财经、政府、机器学习、科学和其他数据。
- SourceForge.net Research Data：包含大约10万个项目和超过100万注册用户的活动的历史和现状的统计数据的项目管理网站。
- Wikiposit：一个（虚拟的）融合了来自许多不同网站的数据（大多数是金融的），允许用户合并来自不同来源的数据。
- Wolfram Alpha disease and patient level dat.
- Yahoo Sandbox datasets：语言、图表、评级、广告与营销、竞赛。
- Yelp Academic Dataset：30家大学的250个最接近商业的所有数据和评论，为学生和学者来探讨和研究。

开放数据源（三），中国，经济为主。

一、政府类

- 1、国家统计局,如果你需要一应俱全的最新宏观经济数据，一个宝贵的来源是国家统计局提供的《进度统计数据》，网址是http://www.stats.gov.cn/tjsj/。 如果想要从数据收集之日起的完整国民经济核算资料，权威的来源是国家统计局国民经济核算司出版的《中国国内生产总值核算历史资料》（1952-1995） 和《中国国内生产总值核算历史资料》（1996-2002）。在这两本年鉴里，提供了核算中国GDP的详实数据。特别是《中国国内生产总值核算历史资料》 （1996-2002）提供了电子版，电子版数据不仅提供1996-2002年的详实数据，还大致回溯了1952-1995年间的数据，非常好用。如果你想要从数据收集之日起的较为完整的宏观经济数据，《新中国五十年统计资料汇编》和《新中国55年统计资料汇编》是一个不错的选择。遗憾的是，它们都没有提供电子版，但后者可以在中国资讯行下载。
- 2、工业和信息化部, 较多数据在此发布，尤其是有关工业运行及信息化相关数据。
- 3、中国人民银行, 中国金融市场政策及运行相关数据。
- 4、银监会,银行金融相关数据。
- 5、中国海关,中国进出口相关数据。
- 6、国家知识产权局,专利相关查询。
- 7、中国证监会,相关政策及招股书披露平台，以及拟上市公司排队每周披露。
- 8、上海市政府数据服务网, 上海市政府数据服务网（dataShanghai）集中发布政府部门及第三方机构的数据产品以及数据应用，数据将涉及经济、教育、卫生、交通、地理、法 律、规划等。上海市政府数据服务网（dataShanghai）中，政府部门提供的数据产品目前都是免费的，保留收费的权利。所有的数据与服务都是无需注册可以直接使用的。搜索到需要的数据标题后，点击进入详细页面，可以看到下载图标。就可以按照需求来下载。
- 9、上海公共研发平台,可以注册，人工审核，内包含较多数据库。

二、综合类

- 1、中国经济数据库,司尔亚司数据信息有限公司（CEIC）成立于1992年，由经济学家和分析师组成，提供有关世界发达经济和发展中经济的最广泛、最精确的信息。作为欧洲货币机构投资公司的一个产物，我们已经成为世界各地经济学家、分析师、投资者、企业以及院校经济和投资研究的首选。
- 2、中国经济信息网,行业研究报告，宏观数据较全。中国经济信息网简称中经网，是国家信息中心组建的、以提供经济信息为主要业务的专业性信息服务网络。
- 3、中国资讯行数据库,收费宏观经济数据。
- 4、国研网, 数据较为权威，有些报告可以一看。
- 5、中国国家图书馆

三、金融类

证券

- 1、上海证券交易所,其中研究出版栏目中有些研究报告.
- 2、深圳证券交易所,其中研究/刊物中有研究报告
- 3、全国中小企业股份转让系统（新三板）,新三板挂牌公司的转让及信息披露.
- 4、香港证券交易所
- 5、台湾证券交易所
- 6、新加坡证券交易所
- 7、纽约证券交易所
- 8、纳斯达克证券交易所

金融

- 1、万德数据库（金融） 中国大陆领先的金融数据、信息和软件服务企业，Wind资讯的客户包括超过90%的中国证券公司、基金管理公司、保险公司、银行和投资公司等金融企业；在 国际市场，已经被中国证监会批准的合格境外机构投资者（QFII）中75%的机构是Wind资讯的客户。同时国内多数知名的金融学术研究机构和权威的监管 机构也是我们的客户，大量中英文媒体、研究报告、学术论文等经常引用Wind资讯提供的数据。 定位：高端机构客户 机构市场占有率：80%优势： （1）数据表结构还是比较科学，而且还有很多不同工具，例如WACC计算小插件、贝塔计算小插件、另外还有直接在EXCEL估值的模版。 （2）用户体现非常好，界面体验一流，符合中国人的使用习惯。 （3）特色数据库有中国A\B股数据、基金数据、债券数据和期货数据都非常突出。 （4）资讯内容结构严重模仿BLOOMBERG （5）支持API插件 缺点 （1）其实我想突出数据质量只是一般，有一些还是很多错误、例如指数的数据库的错误和雅虎Finance几乎是一样的。 （2）世界指数等国际数据库还是一般。 （3）主要是提供资讯，下单通道没有Bloomberg没有那么强大。 （4）行业数据严重缺乏，而且质量真的不太好。
- 2、恒生聚缘（金融） 这个数据库其实也是定位为机构的，还有一套完全的信息技术系统解决方法。但是这个数据库不太出名，但是这是我用过价格便宜然后质量非常高的数据库。 优点： （1）界面设计虽然没有万德那么花哨，但是非常实在，非常实用，而且很方便。数据结构也科学，不会出现过多冗余的状况。 （2）价格比万德便宜，但是性价比挺高的。 （3）A\B股数据是强项 （4）研究报告更新速度比较快，比较全面、质量比万德好。 （5）数据质量过硬。
- 3、CSMAR数据库（金融） 定位：中国80%的学术机构和香港高校都是使用CSMAR，美国大部分的大学例如沃顿等是使用CSMAR数据库 优点 （1）公司金融数据是强项，非常强大和齐全，我经常使用哈哈。 （2）数据库做学术还是比较全面的。年份比较早的数据都会有收录。 （3）高频数据是全国第

二好。 （4）公司治理数据比较好，详细，包括公司控制链图均有收录。 缺点： （1）由于是学术数据库关系，更新速度不够快。机构是绝对不会使用的。 （2）数据结构有些设计是有问题。 （3）缺乏资讯类的数据。 （4）行业数据是更新速度是所有数据库中最慢的，建议不要使用行业数据库。

- 4、锐思数据库（金融） 定位：学术机构 特点： 基本上是Copy外国的数据库结构，而且数据字段不够丰富，建议不要使用。
- 5、巨潮数据库（金融） 深交所旗下的一个数据库公司，有这个得天独厚的优势. （1）交易所的公告、董事会决议总是最快可以知道。 （2）异动数据库中的异动记录肯定不止前十名，获取还能看到前15名，哈哈！ 缺点 （1）数据结构太老的了，严重有问题，见过5个字段来做表主键的，无语。 （2）好像异动数据库，把所有的债券、股票、衍生证、涡轮全部放在一起，结果有一次踩地雷，把债券和股票都提了出来，原因是股票的代码=债券的代码，真的死了。 （3）异动数据中的计算方法严重不正确，如果你查阅交易所对涨跌幅偏离值的计算方法，你会发现在2006年8月4日前后会非常不同。结果又一次让我踩到地雷。 （4）数据质量一般，算不上好。
- 6、清科数据库（金融） 清科研究数据库包含风险投资,私募股权,创业者相关投资,私募,并购,上市数据库,范围涉及投资机构,企业,投资人物相关TMT、传统行业、清洁技术、生技健康等行业市场事件用的比较少，专做Pe,风险投资数据的。
- 7、人大经济论坛 http://bbs.pinggu.org/forum-55-1.html 有许多数据叫卖，提供大量的可供下载的经济学资源，而且还有许多有用的连接。当然，这是一个免费的网站，但下载某些资源时，说不定要求一定的所谓积分限制。这个强力推荐~~

四、互联网类

- 1、淘宝指数
- 2、互联网TMT数据
- 3、百度指数（综合）

五、自然卫生类

- 1、中国气象局
- 2、中国气象科学数据共享服务网 在http://cdcNaNa.gov.cn注册为用户后（密码会发送至你的邮箱）登录，选择数据种类（共14大类），在每类中选择你所关心的数据集，这时弹出每个数据集的元数据信息页面。页面正中有检索方式，选台站或空间、时间就可得到检索结果，点击下载即可。CDC网站的数据只要是共享的数据，就是免费的。
- 3、公共卫生科学数据中心

房地产

- 克尔瑞（房地产）,中国最大、最先进的房地产数据库，易居中国旗下。

其他

- 1、数据堂
- 2、数据熊猫（导航）

开放数据源（四），遥感、地理信息

- 中国国家海洋卫星应用中心
- 中国资源卫星应用中心
- SRTM全球地形高程测量,http://dds.cr.usgs.gov/srtm/,http://www.cgiar-csi.org/data/srtm-90m-digital-elevation-database-v4-1,http://dwtkns.com/srtm/,http://elevation.bopen.eu/en/stable/quickstart.html
- TM全球15米存档多波段专题影像,貌似已无法访问。
- German Aerospace Center (DLR)

其它TM和MSS:

- http://earthexplorer.usgs.gov/
- http://edcsns17.cr.usgs.gov/earthexplorer
- http://glovis.usgs.gov/
- http://landsat.datamirror.csdb.cn/, 这是中科院的国际科学数据服务平台。

多光谱卫星影像:

- http://speclab.cr.usgs.gov
- http://asterweb.jplnasa.gov

遥感数据免费下载网址:

- 1.http://www.landsat.org/ortho/index.htm
- 2.http://edcdaac.usgs.gov/datapool/datatypes.asp
- 3.modis L1B 1km,免费注册，免费下载，daily data.
- 4.http://edcimswww.cr.usgs.gov/pub/imswelcome/
- 5.http://glovis.usgs.gov/
- 6.landsat etm+ and tm images for
- http://www.landsat.org/ortho/index.htm

全球DEM、遥感图像、矢量图像免费下载

- 全球各国shape数据下载，包括矢量要素、dem数据、遥感图片，免费，精度不知。http://biogeo.berkeley.edu/bgm/gdata.php
- 全球各国eoo格式数据下载，包括矢量要素、dem数据、遥感图片，需付费，也有部分类型数据免费，精度不知道。 http://data.geocomm.com/
- 公开的DEM数据,SRTM3/SRTM,数据主要是由美国太空总署(NASA)和国防部国家测绘局(NIMA)

Sphinx 快速入门

Sphinx 简介

Sphinx 令人可以轻松撰写出专业、优雅的文档, 由Georg Brandl在BSD许可证下创造。

Sphinx 采用reStructureText-新结构化文本格式编写文档，可以同时支持HTML、Markdown等流行的格式，并且通过nbsphinx插件可以支持Jupyter Notebook的.ipynb格式。Sphinx支持文档目录自动创建、搜索等功能，可以输出HTMl、PDF等多种格式，还可以通过Github来协同编写版本化的文档，然后在ReadTheDocs.org在线共享生成的文档，或者通过ReadTheDoc开源服务端软件来搭建自己的文档系统。

Sphinx 特性

Python的文档采用Sphinx创建和管理，已支持最新版Python的文档生成, 已经成为Python相关项目的首选文档工具，同时也对C/C++以及其它语言的工程有较好的支持。

Sphnix具有下列特性，并已在Python官方文档中体现:

- 丰富的输出格式: HTML (包括M$帮助), LaTeX (为PDF输出), manual pages(man), 纯文本
- 完备的交叉引用: 语义化的标签,并对 函式,类,引文,术语以及类似片段消息可以自动化链接
- 明晰的分层结构: 轻松定义文档树,并自动化链接同级/父级/下级文章
- 美观的自动索引: 可自动生成美观的模块索引
- 精确的语法高亮: 基于 Pygments 自动生成语法高亮
- 开放的扩展: 支持代码块的自动测试,自动包含Python 的模块自述文档,等等。

Sphinx 文档

Sphinx中文版是改进的中文版，使用了Sphinx-rtd-theme更易于阅读，增加了中文字符集的支持和PDF输出选项。可以在这里查看以前版本，需要注意其中的变更，可能新版本的Sphinx中已经不再适用。

Sphinx 项目

大量的软件项目使用了Sphinx文档系统，下面是其中一些比较有影响的项目。

缺省theme

- APSW: http://apidoc.apsw.googlecode.com/hg/index.html
- ASE: https://wiki.fysik.dtu.dk/ase/
- boostmpi: http://documen.tician.de/boostmpi/
- Calibre: http://calibre-ebook.com/user_manual/
- CodePy: http://documen.tician.de/codepy/
- Cython: http://docs.cython.org/
- C:raw-latex:‘C‘++ Python language binding project: http://language-binding.net/index.html
- Cormoran: http://cormoran.nhopkg.org/docs/
- Director: http://packages.python.org/director/

- Dirigible: http://www.projectdirigible.com/documentation/
- Elemental: http://elemental.googlecode.com/hg/doc/build/html/index.html
- F2py: http://f2py.sourceforge.net/docs/
- GeoDjango: http://geodjango.org/docs/
- Genomedata: http://noble.gs.washington.edu/proj/genomedata/doc/1.2.2/genomedata.html
- gevent: http://www.gevent.org/
- Google Wave API: http://wave-robot-python-client.googlecode.com/svn/trunk/pydocs/index.html
- GSL Shell: http://www.nongnu.org/gsl-shell/
- Heapkeeper: http://heapkeeper.org/
- Hands-on Python Tutorial: http://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/
- Hedge: http://documen.tician.de/hedge/
- Kaa: http://doc.freevo.org/api/kaa/
- Leo: http://webpages.charter.net/edreamleo/front.html
- Lino: http://lino.saffre-rumma.net/
- MeshPy: http://documen.tician.de/meshpy/
- mpmath: http://mpmath.googlecode.com/svn/trunk/doc/build/index.html
- OpenEXR: http://excamera.com/articles/26/doc/index.html
- OpenGDA: http://www.opengda.org/gdadoc/html/
- openWNS: http://docs.openwns.org/
- Paste: http://pythonpaste.org/script/
- Paver: http://paver.github.com/paver/
- Pyccuracy: https://github.com/heynemann/pyccuracy/wiki/
- PyCuda: http://documen.tician.de/pycuda/
- Pyevolve: http://pyevolve.sourceforge.net/
- Pylo: http://documen.tician.de/pylo/
- PyMQI: http://packages.python.org/pymqi/
- PyPubSub: http://pubsub.sourceforge.net/
- pyrticle: http://documen.tician.de/pyrticle/
- Python: http://docs.python.org/
- python-apt: http://apt.alioth.debian.org/python-apt-doc/
- PyUblas: http://documen.tician.de/pyublas/
- Quex: http://quex.sourceforge.net/doc/html/main.html
- Scapy: http://www.secdev.org/projects/scapy/doc/
- Segway: http://noble.gs.washington.edu/proj/segway/doc/1.1.0/segway.html
- SimPy: http://simpy.sourceforge.net/SimPyDocs/index.html
- SymPy: http://docs.sympy.org/

- WTForms: http://wtforms.simplecodes.com/docs/
- z3c: http://docs.carduner.net/z3c-tutorial/

缺省theme 修改版

- Advanced Generic Widgets: http://xoomer.virgilio.it/infinity77/AGW_Docs/index.html
- Bazaar: http://doc.bazaar.canonical.com/en/
- Chaco: http://code.enthought.com/projects/chaco/docs/html/
- Djagios: http://djagios.org/
- GetFEM++: http://home.gna.org/getfem/
- GPAW: https://wiki.fysik.dtu.dk/gpaw/
- Grok: http://grok.zope.org/doc/current/
- IFM: http://fluffybunny.memebot.com/ifm-docs/index.html
- LEPL: http://www.acooke.org/lepl/
- Mayavi: http://code.enthought.com/projects/mayavi/docs/development/html/mayavi
- NOC: http://redmine.nocproject.org/projects/noc
- NumPy: http://docs.scipy.org/doc/numpy/reference/
- Peach^3: http://peach3.nl/doc/latest/userdoc/
- PyLit: http://pylit.berlios.de/
- Sage: http://sagemath.org/doc/
- SciPy: http://docs.scipy.org/doc/scipy/reference/
- simuPOP: http://simupop.sourceforge.net/manual_release/build/userGuide.html
- Sprox: http://sprox.org/
- TurboGears: http://turbogears.org/2.0/docs/
- Zentyal: http://doc.zentyal.org/
- Zope: http://docs.zope.org/zope2/index.html
- zc.async: http://packages.python.org/zc.async/1.5.0/

Sphinxdoc theme

- Fityk: http://fityk.nieto.pl/
- MapServer: http://mapserver.org/
- Matplotlib: http://matplotlib.sourceforge.net/
- Music21: http://mit.edu/music21/doc/html/contents.html
- MyHDL: http://www.myhdl.org/doc/0.6/
- NetworkX: http://networkx.lanl.gov/
- Pweave: http://mpastell.com/pweave/
- Pyre: http://docs.danse.us/pyre/sphinx/

- Pysparse: http://pysparse.sourceforge.net/
- PyTango: http://www.tango-controls.org/static/PyTango/latest/doc/html/index.html
- Reteisi: http://www.reteisi.org/contents.html
- Satchmo: http://www.satchmoproject.com/docs/dev/
- Sphinx: http://sphinx.pocoo.org/
- Sqlkit: http://sqlkit.argolinux.org/
- Tau: http://www.tango-controls.org/static/tau/latest/doc/html/index.html
- Total Open Station: http://tops.berlios.de/
- WebFaction: http://docs.webfaction.com/

内置专题－Builtin theme

- C/C++ Development with Eclipse: http://eclipsebook.in/ (agogo)
- Distribute: http://packages.python.org/distribute/ (nature)
- Jinja: http://jinja.pocoo.org/ (scrolls)
- jsFiddle: http://doc.jsfiddle.net/ (nature)
- pip: http://pip.openplans.org/ (nature)
- Programmieren mit PyGTK und Glade (German): http://www.florian-diesch.de/doc/python-und-glade/online/ (agogo)
- Spring Python: http://springpython.webfactional.com/current/sphinx/index.html (nature)
- sqlparse: http://python-sqlparse.googlecode.com/svn/docs/api/index.html (agogo)
- Sylli: http://sylli.sourceforge.net/ (nature)
- libLAS: http://liblas.org/ (nature)

定制theme－网站集成

- Blender: http://www.blender.org/documentation/250PythonDoc/
- Blinker: http://discorporate.us/projects/Blinker/docs/
- Classy: classy: http://classy.pocoo.org/
- Django: http://docs.djangoproject.com/
- e-cidadania: http://e-cidadania.readthedocs.org/en/latest/
- Flask: http://flask.pocoo.org/docs/
- Flask-OpenID: http://packages.python.org/Flask-OpenID/
- Gameduino: http://excamera.com/sphinx/gameduino/
- GeoServer: http://docs.geoserver.org/
- Glashammer: http://glashammer.org/
- MirrorBrain: http://mirrorbrain.org/docs/
- nose: http://somethingaboutorange.com/mrl/projects/nose/

- ObjectListView: http://objectlistview.sourceforge.net/python
- Open ERP: http://doc.openerp.com/
- OpenLayers: http://docs.openlayers.org/
- PyEphem: http://rhodesmill.org/pyephem/
- German Plone 4.0 user manual: http://www.hasecke.com/plone-benutzerhandbuch/4.0/
- Pylons: http://pylonshq.com/docs/en/0.9.7/
- PyMOTW: http://www.doughellmann.com/PyMOTW/
- pypol: http://pypol.altervista.org/ (celery)
- qooxdoo: http://manual.qooxdoo.org/current
- Roundup: http://www.roundup-tracker.org/
- Selenium: http://seleniumhq.org/docs/
- Self: http://selflanguage.org/
- Tablib: http://tablib.org/
- SQLAlchemy: http://www.sqlalchemy.org/docs/
- tinyTiM: http://tinytim.sourceforge.net/docs/2.0/
- tipfy: http://www.tipfy.org/docs/
- Werkzeug: http://werkzeug.pocoo.org/docs/
- WFront: http://discorporate.us/projects/WFront/

网站－Homepages

- Applied Mathematics at the Stellenbosch University: http://dip.sun.ac.za/
- A personal page: http://www.dehlia.in/
- Benoit Boissinot: http://bboissin.appspot.com/
- lunarsite: http://lunaryorn.de/
- Red Hot Chili Python: http://redhotchilipython.com/
- The Wine Cellar Book: http://www.thewinecellarbook.com/doc/en/
- VOR: http://www.vor-cycling.be/

Sphinx－书籍

- The repoze.bfg Web Application Framework
- A Theoretical Physics Reference book
- Simple and Steady Way of Learning for Software Engineering(in Japanese)
- Expert Python Programming(Japanese translation)
- Pomodoro Technique Illustrated(Japanese translation)

“nbsphinx-toctree”: { “maxdepth”: 3, “numbered”: true }

安装并启动jupyter

安装并启动jupyter

安装 Anaconda 后, 再安装 jupyter

```
pip install jupyter
```

- 设置环境

```
ipython --ipython-dir= # override the default IPYTHONDIR directory, ~/.ipython/ by default
ipython profile create foo # create the profile foo
ipython profile locate foo # find foo profile directory, IPYTHONDIR by default,
ipython --profile=foo # start IPython using the new profile
```

启动jupyter的几个命令, 启动后, 默认还将启动一个浏览器进入 notebook 环境 ipython notebook # 启动 jupyter notebook服务器, 使用默认端口8080 ipython notebook –ip=0.0.0.0 –port=80 # 启动 jupyter notebook服务器, 指定端口 ipython notebook –profile=foo # 使用 foo profile 启动 jupyter notebook服务器 ipython notebook –pylab inline # 启用 PyLab graphing support ipython notebook 是老版本的命令组合, 新版是jupyter notebook命令组合, 如果使用Anaconda的发布包, 直接使用jupyter-notebook这个工具.

更多jupyter使用信息, 见 http://nbviewer.jupyter.org/github/ipython/ipython/blob/3.x/examples/Notebook/Notebook%20Basics.ipynb

定制Jupyter

[root#]./jupyter-notebook –generate-config 将生成一个jupyter的配置文件, 比如 /root/.jupyter/jupyter_notebook_config.py , 在其中可配置Notebook App的基本信息文件名为: /root/.jupyter/jupyter_notebook_config.py

```
c = get_config()
c.IPKernelApp.pylab = 'inline'
c.NoteBookApp.ip = '0.0.0.0'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 8880 # or whatever you want
```

为 notebook 的 cell增加line number

在 ~/.ipython/profile_foo/static/custom/custom.js 增加下面几行

define(['base/js/namespace', 'base/js/events'], function(IPython, events) { events.on("app_initialized.NotebookApp", function () { require("notebook/js/cell").Cell.options_default.cm_config.lineNumbers = true; }); });

更改jupyter的主题

https://github.com/transcranial/jupyter-themer 更改命令 jupyter-themer -c monokai 访问远端的notebook server

pip install jupyter-themer

Jupyter服务器

如果是在远端开启了notebook server, 在本机浏览器直接访问, 将不能打开, 这应该是notebook server为安全考虑, 有意屏蔽非本机的访问. 当然, 我们总不能一直通过x-windows到远端打开浏览器使用notebook吧.

1.最简单的做法是,启动notebook sever时, 加上参数–ip=0.0.0.0, 即: ./jupyter-notebook –port=7777 –ip=0.0.0.0 2.另一个方法是:在本机使用ssh软件建立一个连接远端的ssh tunnel, 本机浏览器通过ssh tunnel就可以访问远端的notebook server. 比如, 我习惯使用putty, 方法是: putty-> Connection > SSH > Tunnels Under Add new forwarded port:, enter the following information: Source port: local_port Destination: remote_host:remote_port Click Add.然后, 使用这个配置ssh连接远端server. 访问远端的ipython

jupyter是使用tab能进行代码补全, 但在浏览器中有时并不起作用, 使用远端的ipython就没有这个问题, 当然ipython在EDA(探索式数据分析)文档化方面比notebook弱多了, 鱼和熊掌不能皆得. 我更喜欢ipython qtconsole方式. 连接远端ipython的步骤:

```
远端开启ipython host, 命令为 ipython kernel -f ~/ipython-connect-info.json # 这是一个没有前端的ipython进程
将远端的kernel-1234.json 文件复制到本机, 修改其中的ip地址为远端真实的地址
本机上使用putty为kernel-1234.json 文件中的5个port都开启ssh tunnel
本机执行 ipython qtconsole --existing c:\kernel-connect-info.json
```

与PySpark集成

IPython和普通的Python interpreter相比, 优点在于对交互性支持更好, 所以PySpark只有在需要更好交互性的情形下, 才有集成IPython的必要, 显然只有 pyspark shell 才需要集成IPython. Jupyter和PySpark shell集成方式有好几种, 比如:

```
先启动IPython, 然后调用pyspark\shell.py启动spark.
启动IPython后, 我们可以手动调用pyspark\shell.py, 将调用脚本加到IPython profile目录中自动启动, 自动启动python程序
00-pyspark-setup.py的写法可参考 https://github.com/harisekhon/pytools/blob/master/.ipython-notebook-pys

采用IPython 这个高级 interpreter 来启动pyspark
例子1, 在 spark master server 上以notebook的形式启动 pyspark shell.
spark_master_node$ PYSPARK_DRIVER_PYTHON=/usr/python_anaconda/bin/ipython PYSPARK_DRIVER_PYTHON_OPTS=
```

例子2, 在 spark master server 上以ipython kernel的形式启动 pyspark shell, 以便其他机器连入 spark_master_node$ PYSPARK_DRIVER_PYTHON=/usr/python_anaconda/bin/ipython PYSPARK_DRIVER_PYTHON_OPTS=”kernel -f ~/ipython-connect-info.json” pyspark –master spark://quickstart.cloudera:7077

例子3, 在 spark master server 上以notebook的形式启动 pyspark shell, 并加载额外的package, 指定内存参数. spark_master_node$ PYSPARK_DRIVER_PYTHON=ipython PYSPARK_DRIVER_PYTHON_OPTS=”notebook –no-browser –port=7777 –profile=foo” pyspark –packages com.databricks:spark-csv_2.10:1.1.0 –master spark://spark_master_hostname:7077 –executor-memory 6400M –driver-memory 6400M

设置 PYSPARK_DRIVER_PYTHON 和 PYSPARK_DRIVER_PYTHON_OPTS 环境变量后, 之后调用pyspark将采用这两个环境变量指定的Python 解释器配置来运行python 版spark 应用.

注意不应该export 这两个环境变量, 因为export后, 非shell的pyspark spark应用也将使用IPython运行, 容易造成滥用. 虽然我不推荐在Linux profile将 PYSPARK_DRIVER_PYTHON 设置为IPython, 但我推荐将 PYSPARK_DRIVER_PYTHON 设置为 Anaconda的 python, 因为这将省去安装额外科学计算包的麻烦, 当然, 环境变量PYSPARK_DRIVER_PYTHON_OPTS不应该再带上notebook或kernel参数.

最好在 spark-env.sh 增加下面4个参数,

```
# worker和driver 的python版本应该是一致的, 否则spark会报错的, 当然driver可以设置为ipython.
export PYSPARK_PYTHON=/usr/bin/python3
export PYSPARK_DRIVER_PYTHON=/usr/bin/ipython3
# 为了简化提交pyspark 应用的提交,可以预先设置一个 PYSPARK_SUBMIT_ARGS 环境变量.
export PYSPARK_SUBMIT_ARGS='--master local[2]'
export PYSPARK_SUBMIT_ARGS='--master yarn --deploy-mode client --num-executors 24 --executor-memory
```

- 参考文章

1. How-to: Use IPython Notebook with Apache Spark
2. http://blog.cloudera.com/blog/2014/08/how-to-use-ipython-notebook-with
3. https://www.dataquest.io/blog/installing-pyspark/
4. Configuring IPython Notebook Support for PySpark
5. Using Jupyter on Apache Spark: Step-by-Step with a Terabyte of Reddit Data
6. 如何自定义jupyter notebook的主题
7. jupyter cell 增加 line number
8. Spark编程环境搭建(IPython)
9. 如何使用Docker快速配置数据科学开发环境(搭建Docker + Jupyter环境)

Python数据操作

在Python中有list、dictionary、Tuple等多种类似于数组的数据结构实现，可以帮助快速操作数据。

by openthings@163.com, 2016-04.

List，使用[a,b,c,...]方式声明。

列表是基础的数据结构。

```
In [9]: alist = [0,1,2,3,4]
        print("总计: ",len(alist))
        print("元素:", alist)
```

```
总计:  5
元素: [0, 1, 2, 3, 4]
```

字符串的列表。

```
In [20]: colours = ["red","green","blue"]
         for colour in colours:
             print(colour)
```

```
red
green
blue
```

列表的传统方式遍历。

```
In [24]: for i in range(0,len(alist)):
             print(alist[i])
```

```
0
1
2
3
4
```

列表的递归方式遍历。

```
In [25]: for i in alist:
             print(i)
```

```
0
1
2
3
4
```

可以直接调用一个列表。

```
In [26]: for obj in [0,1,2,3,4]:
             print(obj)
```

```
0
1
2
3
4
```

** 创建一个矩阵。**

```
In [63]: olist = [[11,12,13],[21,22,23],[31,32,33]]
         for row in olist:
             print(row)
```

```
[11, 12, 13]
[21, 22, 23]
[31, 32, 33]
```

** 生成一个数据序列。在做数值检验时很有用。**

```
In [43]: for obj in range(5):
             print(obj)
```

```
0
1
2
3
4
```

** 生成一个数据序列：range(开始值，结束值，步长)**

```
In [27]: for obj in range(5,10,2):
             print(obj)
```

```
5
7
9
```

** String as a list of char. 字符串是字符数组。**

```
In [7]: name='BeginMan'
        for obj in range(len(name)):
            print('(%d)' %obj,name[obj])
```

```
(0) B
(1) e
(2) g
(3) i
(4) n
(5) M
(6) a
(7) n
```

Dictionary，词典，{ key:value,... }

词典数据就是一系列k:v值对的集合。

```
In [30]: dict = {'name':'BeginMan','job':'pythoner','age':22}

         print("Dict Length: ",len(dict))
         print(dict)
```

```
Dict Length:  3
'age': 22, 'job': 'pythoner', 'name': 'BeginMan'
```

注意：上面的词典数据的输出与json表示是完全一致的，后面在json会专门介绍。

词典的遍历：

```
In [71]: dict["name"]
```

```
Out[71]: 'BeginMan'
```

```
In [41]: print("Key","\t Value")
         print("=================")
         for key in dict:
             print(key,"\t",dict[i])
```

```
Key      Value
=================
age      BeginMan
job      BeginMan
name     BeginMan
```

dict的每一个item(obj)是一个二元组（下面介绍元组）。

```
In [31]: for obj in dict.items():
             print(obj)
```

```
('age', 22)
('job', 'pythoner')
('name', 'BeginMan')
```

```
In [46]: for k,v in dict.items():
             print(k,v)
```

```
age 22
job pythoner
name BeginMan
```

** Json的简单演示，将dict转为典型的json字符串表示。**

```
In [72]: import json

         j = json.dumps(dict)
         print(repr(j))
```

```
'"age": 22, "job": "pythoner", "name": "BeginMan"'
```

** 从JSON字符串载入一个dict对象。**

```
In [53]: d = json.loads('{"age": 22, "job": "pythoner", "name": "BeginMan"}')
         print("Type of d: ", type(d))
         print(d)
```

```
Type of d:  <class 'dict'>
'name': 'BeginMan', 'job': 'pythoner', 'age': 22
```

tuple，(obj1,obj2,...)，元组

一个元组可包含多种类型的对象，不可修改。

```
In [59]: tup = 'abc',1,2,'x',True

In [60]: len(tup),tup

Out[60]: (5, ('abc', 1, 2, 'x', True))
```

** 元组的赋值和取值。**

```
In [56]: x,y =1,2

In [10]: x,y

Out[10]: (1, 2)
```

一个字典、元组构成的复合列表对象。

```
In [67]: ao = [{"k1":"key","k2":2},(3,"element")]
         ao

Out[67]: [{'k1': 'key', 'k2': 2}, (3, 'element')]
```

** 访问这个符合对象的值。** len(ao),ao[0]["k1"],ao[1][0],ao[1][1] 从上面可以看出，python的数据结构是非常灵活的，是数据探索和分析、处理的利器。

CHAPTER 2

Indices and tables

- genindex
- modindex
- search