2018 年 3 月 5 日星期一　　第一周

微服务理念已经被越来越多的开发者所接受，逐渐应用在丰富的生产环境中，微服务理念将软件架构中复杂的逻辑关系进行解耦，拆分重叠的功能单元，使每一个独立的功能单元替换时不会影响到其他的功能单元，给开发者提供统一标准的开发环境单元，帮助开发者进行快速的版本迭代更新，有效的降低了开发成本与开发团队之间的沟通效率。本文中将使用微服务的设计理念，从底层架构入手改变传统的设计模式，基于 docker 微服务的实现方案完成机器学习算法的容器化分布式计算节点实现。

2018 年 3 月 7 日星期三　　第一周

将机器学习算法进行容器化，机器学习是现在的非常热点的研究领域，在人工智能、无人驾驶中都有广泛的应用，随着应用范围的增加，机器学习算法的复杂度也呈指数级增长，计算集群就变得越来越大，通过单台物理计算机完成任务越来越困难，急需一种可以解决单台物理机计算瓶颈的方案。

2018 年 6 月 13 日星期三　　第十五周

本次毕业设计中将机器学习算法容器化，并使用微服务的架构思想拆分具体的内部逻辑，实现了简单的容器化分布式计算节点，完整运行了机器学习算法，将计算节点性能提升为原来的2倍，取得了良好的结果。本次论文中的设计方案还有许多地方是可以进行完善的，比如微服务模型如何根据用户的需求进一步拆分，分布式分配任务时，如何合理的进行负载均衡，节点的选择算法哪种更合适，多用户共同使用的复杂场景时，如何规划混合 IP 网段使用与多租户场景模式，还可以结合容器集群的管理平台 kubernetes 完成 paas 云平台的改造等，由于时间原因不能在这里一一实现，但会在今后的工作学习中继续完善。

2018 年 3 月 12 日星期一　　第二周

使用虚拟化技术的 docker 容器化分布式的方案，将大规模问题拆分为小规模问题，分发到计算节点上，利用多计算节点来减少单台的任务压力，并封装机器学习的执行环境，让开发与执行环境统一标准，以此来降低占用资源的使用率，提高容器化的部署密度，保证一次封装，到处运行。

2018 年 3 月 14 日星期三　　第二周

网络设计方案选择 vxlan 隧道与 IP 路由相结合的方式，适合在复杂的多租户场景中使用，保证网络的健壮性与快速恢复性，并且不会受限于网络划分的个数。将底层的物理设备透明化，使跨宿主机容器之间能正常通信。经过功能验证和性能测试，可以完整的在容器化计算节点中运行机器学习算法，通过多容器化计算节点的分布式部署，使原本物理计算机的运算性能提高了2倍，将运算节点的启动速度压缩到秒级，随着容器化计算节点的增加，提升效果将会更加明显。

2018 年 6 月 11 日星期一　　第十五周

本次毕业设计中结合了工程的相关概念，基于工程相关背景知识进行合理分析，具体分析了不同模型带来的工程实现复杂度与可行性分析，评价计算机科学与技术专业工程实践和复杂工程问题解决方案对社会、健康、安全、法律以及文化的影响，并理解了应承担的责任。

2018 年 6 月 6 日星期三　　第十四周

作为技术人员不仅要在学校学习的阶段努力学习知识，更要在今后的工作之中继续学习提升自己的能力，做到可以自主学习和终身学习。随着开源的思想越来越浓厚，开源技术的迭代也越来越快，只有不断学习新的技术知识，提升自己才能在今后的工作中不被淘汰，这一点是我在本次毕设中最重要的收获。

2018 年 3 月 19 日星期一　　第三周

微服务理念是将工程项目中的复杂逻辑进行解耦，拆分为一个个独立可随时替换的功能单元，某一个功能单元的替换，不会影响其他单元的使用，这一理念可以帮助开发人员快速迭代项目，且不需要关心功能单元之间的耦合性，降低了沟通成本，节省了实际成本。当微服务理念与虚拟化技术相结合时又有了新的应用。将利用根据微服务理念，使用容器技术并改造其网络模型，将机器学习算法容器化并实现容器化分布式的计算节点部署。

2018 年 3 月 21 日星期三　　第三周

虚拟化是将一台物理计算机虚拟为多台逻辑计算机，在一台物理计算机上同时运行多个逻辑计算机，每个逻辑计算机可运行不同的操作系统，并且应用程序都可以在相互独立的空间内运行而互不影响[1-5]。随着虚拟化技术的快速发展，云的概念孕育而生，云将原本单独的计算节点的计算、存储、网络带宽等资源虚拟化，变成一个可按照粒度进行分配的资源池。将分散的计算节点集中起来，然后将硬件资源虚化成池，这样大大提高了硬件资源的利用率。容器技术是虚拟化技术发展的又一方向，容器是轻量级的操作系统级的虚拟化，将应用程序依赖的组件或环境打包成镜像，通过镜像产生容器实例，此时容器运行在一个资源隔离的环境中，即一个容器的变化无法影响其他容器。

2018 年 6 月 4 日星期一　　第十四周

既然有优势，主机模式也有它明显的劣势：

a）此时的容器也不再拥有隔离的独立的网络资源环境，容器会和物理计算机进行网络栈的竞争使用，一旦容器导致网络栈崩溃，那么物理计算机也可能崩溃。这种类似于硬编码的网络模式存在潜在的风险，因此在实际的生产环境中严禁被使用。

b）容器不再通过端口绑定的方式来接收流量，而是使用物理计算机的端口，这就使容器可利用的端口数大大下降，在物理计算机上本就有限的端口变的更加稀缺。

2018 年 5 月 30 日星期三　　第十三周

通过主机模式创建的容器在和外界进行通信时，是直接利用物理计算机的 IP 和端口。此时的流量转发也不会经过 iptables NAT 的转换和 docker0 网桥的转发，所以主机模式在效率和性能上高于网桥模式。

2018 年 3 月 26 日星期一　　第四周

互联网经过了几十年的发展，每一次跨阶段飞跃的本质都是数据量的飞速增长。20 世纪初，互联网开始变成社交的载体，人们通过使用 Facebook，Twitter、QQ 等社交工具进行交流沟通，实时互动，而每一个人此时也变成了数据的生产者。2012 年，手机交流应用软件推特仅仅一天时间就可以产生 80 亿个单词的信息量，而普通的一家纸质媒体在过去的 50 年时间里总共能产生 30 亿个单词的信息量[6-8]。由此看来，数据量成爆炸式指数级增长，粗略估算，现在一天产生的数据总量相当于普通的一家纸质媒体在 100 多年里产生的数据总量。如此巨大的信息量让物理计算机的运算负荷越来越重，人们只能通过升级 CPU 的工艺制程、增加逻辑运算单元的数量、加大缓存的带宽速率，在提升性能的同时降低功耗。

2018 年 3 月 28 日星期三　　第四周

起初通过升级硬件的方式解决了数据量剧增，运算速率太慢的问题，此时摩尔定律仍在发挥它的作用。摩尔定律是由 Inter 的创始人之一 Gordon Moore 所提出，定律指出当价格不变时，集成电路上可容纳的元器件的数目，约每隔 18-24 个月便会增加一倍，性能也将提升一倍。但随着 CPU 的工艺制程越来越小，硅晶体管就相当于几个分子的大小，材料的宽度正在逐渐逼近物理极限，这对硅晶体管的纯度和集成度的要求越来越高，带来最直接的影响就是成本呈指数级增长，摩尔定律的用在慢慢消失。

2018 年 5 月 28 日星期一　　第十三周

主机模式是指创建容器时不会根据 net namespace 隔离资源，此时容器已不再拥有自己独立的隔离的网络资源，而是和物理计算机共同使用一个网络命名空间，，容器的 IP 和物理计算机的 IP 保持一致。可以理解成为容器和宿主机属于同一类机器，容器可以使用物理计算机上的的所有网卡。

2018 年 5 月 23 日星期三　　第十二周

但显而易见的是，由于将容器端口和物理计算机的端口进行一一绑定，当容器过多时会影响到物理计算机端口资源，此时在物理计算机上的端口管理也变得异常复杂困难，同时根据 iptables NAT 选择的流量转发，带来的效率和性能上的降低也是可预见的。

2018 年 4 月 2 日星期一　　第五周

如今的时代是数据爆炸的时代，每个人作为独立的信息个体，每天都会产生巨大的信息流，但是串行计算方式的计算速率已经远远不能满足如此庞大的计算需求，急需新的信息流处理方式来解决这个问题。人们想到如果单台物理计算节点的性能不够强大，那如果把大量的物理计算节点聚合在一起，同时处理一个问题，能否大幅提升处理性能？此时分布式计算的概念被提出，分布式计算的最早形态出现在上世纪 80 年代末的 Inter 公司。随着 Internet 的迅速发展和普及，分布式计算的研究在 90 年代后达到了高潮。

2018 年 4 月 4 日星期三　　第五周

分布式计算的原理是将一台计算节点无法在短时间内处理完成的计算量大的问题，拆解成多个计算量小的问题，然后将这些计算量小的问题分发到多个计算节点上，计算节点计算处理完成后把结果进行集合，就得到了原本计算量大的问题的结果。成千上万的计算节点通过网络连接起来，组成一台虚拟的超级计算机，完成单台计算机无法完成的超大规模问题求解。通过开源社区开发者的持续贡献，各大分布式计算平台不断发展，功能与特性日趋完善。其中以 Hadoop 和 Spark 为首的分布式计算平台发展快速，在众多互联网公司中应用广泛。

2018 年 5 月 21 日星期一　　第十二周

使用桥接模式时，创建的容器会为其分配独立的网络资源，拥有独立的网络栈，保证容器内部的网络和物理计算机的网络相互隔离。下图为桥接模式的模型，在同一物理计算机上的容器都会连接到相同的 docker0 网桥上，docker0 网桥在这里起到了虚拟交换机的作用。因为物理计算机的 IP 与容器所分配的 IP 不在同一个网段，所以仅通过 net namespace 和 veth pair 还不能让容器和本物理计算机以外的网络进行通信。所以还需要在 docker0 和物理计算机上的物理网卡之间使用 NAT 技术，并将物理计算机上的端口和容器某一端口进行绑定，保证端口流量能正确的转发到容器端口上。

2018 年 5 月 16 日星期三　　第十一周

桥接模式是指容器创建时通过 DHCP 动态主机配置协议获取一个和已经存在的 docker0 网桥同网段的 IP 地址，并默认连接到 docker0 网桥，以此实现容器与宿主机的网络互通。桥接模式是容器的默认网络设置，在创建容器时如果没有额外指定网络模式，则为桥接模式。

2018 年 4 月 9 日星期一　　第六周

随着互联网时代的发展，快速开发迭代和弹性伸缩的要求越来越高，应用开发的传统三层架构（界面层、业务逻辑层、数据访问层）已不能满足需求，取而代之采用微服务的方式，实现软件系统的低耦合，从而达到跨部门开发和快速交付的目的[4]。应用开发不仅在架构上遇到了瓶颈，开发体量的不断增大也带来了新的问题。开发的体量越大，投入到项目中的开发人员就越多，其中开发、测试、运维、产品、运营人员之间的沟通成本越来越高，协同工作的效率变得很低，互相之间的职责关系愈发模糊。此时出现了 DevOps 的概念，人们迫切需要一种便捷、自动化、统一的开发运营平台。

2018 年 4 月 11 日星期三　　第六周

Docker 的出现加速了这种愿景的实现。Docker 是一款开源的轻量级容器管理工具，提供针对分布式应用构建、交付、运行的功能。Docker 利用 Linux 的内核机制，例如 cgroups、namespace、selinux 等来实现容器之间的隔离。Docker 容器可以类比集装箱，将应用使用的各种组件封装起来，一次封装多次部署，做到随时迁移，而不需要关注底层环境。

2018 年 5 月 14 日星期一　　第十一周

分布式计算节点由之前的物理计算机转变为容器，这就带来了新的问题：容器计算节点和外界进行通信时，必须要通过物理计算机，那么该如何选择容器的网络模式才能保证和容器外界的正常通信？Linux 内核资源分配相关的众多特性中有一个非常重要，就是根据 namespace 的设置将物理计算机有限的资源隔离，可以为每一个命名空间分配独立的资源环境。对于网络资源，就可以根据 net namespace 的设置，为容器创建隔离的网络环境，在隔离的网络环境中，容器具有完全独立的网络栈，与物理计算机的网络资源进行隔离。或让容器和物理计算机、容器和容器之间使用相同的 net namespace，以共享网络资源。

2018 年 4 月 16 日星期一　　第七周

互联网发展如今迎来了新的阶段，人工智能的浪潮又一次将人们的想象定格在未来，“千人千面”、“无人汽车”、“虚拟现实”，这些未来应用的背后意味着数据计算量的又一次爆炸式指数级增长。数据的产生来源从每个独立个体，变成每个可以连接到互联网的设备，万物互联的场景即将到来。同时，数据信息的来源越是复杂，对数据进行分析的难度就越大，分析的结果变得非常模糊，有可能已经不再具有代表性，所以还要再数据分析前，对数据集合进行提前的预处理，把无用的数据剔除掉，保证数据具有代表性。这些新需求对分布式计算技术提出了新的挑战——实时计算，实时计算不仅要求将大数据量的问题解决，而且要求能根据问题规模的大小自适应的选择计算节点数量，快速将问题处理完成。

2018 年 5 月 9 日星期三　　第十周

分布式计算的核心是将规模大的问题分解成规模小的问题，然后将规模小的问题分发到各个计算节点上，之后再将计算节点处理完成的结果进行汇总，得到规模大的问题的结果，因此各个计算节点之间如何进行通信就显得十分重要。

2018 年 4 月 18 日星期三　　第七周

如果能解决这一技术问题，那么对于互联网厂商来说，不仅可以将对开发、测试、运营等人员提供一个统一、透明、高效的 DevOps 开发平台，不用关心底层的具体实现，关注自己的工作任务即可，而且节约的成本将是巨大的。利用容器技术，结合分布式计算的思想理念，基于 Docker 微服务的机器学习算法容器化的实现，提高整体算法容器化的性能，使用容器解决虚拟机占用资源多、启动慢的缺点，可以更快的实现交付和弹性伸缩、做到一次封装多次部署。

2018 年 5 月 7 日星期一　　第十周

考虑到环境有可能给不同需求的业务方使用的场景或者要频繁更新环境的场景，因此采用镜像分层制作的设计思想。最底层为基础镜像层，即系统镜像层，本方案中使用官方提供的 CentOS 镜像，但是系统镜像中原始的 yum 源和 pip 源下载速度过慢，因此更改为清华大学的源。基础镜像层的上层是语言镜像层，这一层封装了本文中机器学习算法运行的必需语言环境。第三层为工具镜像层，封装了各个容器化计算节点之间通信所需要的所有插件工具与 IP 路由配置。最上层为容器运行时临时创建的镜像层，用户在容器化计算节点中的操作和改变都会记录在此层。除了临时创建的容器运行层是可读可写层，其他镜像层都是只读不可写的。

2018 年 4 月 23 日星期一　　第八周

毕业论文一共有五章。第一章为绪论，主要介绍本文的研究内容，根据查阅的文献资料与阅读的技术文档，阐述研究背景，分析在实际生产环境中的实际意义，总结研究意义；第二章为微服务模型与容器化分布式模型的介绍，主要介绍本文中设计的总体架构设计，包括微服务模型设计、容器设计架构与镜像设计思路，并阐述以容器为基础的分布式网络架构和容器的基本原理；第三章为跨宿主机的多容器网络模型选型的介绍，详细阐述二层 vlan 模型与 vxlan 隧道模型，两种模型方案的内容，介绍每一个方案的特点与适用场景，并对方案进行对比，选出最适合本方案的模型；第四章为分布式计算节点模型选型的介绍，详细阐述虚拟机模型与容器模型，两种模型方案的内容，介绍每一种模型的特点。

2018 年 5 月 2 日星期三　　第九周

本方案的计算节点采用了容器化的方式。主节点存在一个主进程，此主进程控制相关镜像的制作、下载、上传等镜像相关的任务，并且控制容器计算节点的启动与暂停。主进程与计算机节点中的 Docker 守护进程进程进行通信，传递相关指令。Docker 守护进程会一直在后台运行，等待接收指令，在收到相关指令后会根据指令的具体内容调取计算节点本地的镜像存储仓库，选择仓库中的某一个镜像来产生容器实例。当本地镜像仓库没有需要的镜像时，就需要从远端景象仓库查找镜像，查询有此镜像时会将镜像下载到本地，然后执行上述的操作。

2018 年 4 月 25 日星期三　　第八周

随着软件开发规模越来越大，快速迭代的需求变得难以满足，开发、测试与生产环境的不同也导致研发成本提高。本节将介绍微服务模型的架构，利用微服务的设计理念，从底层设计架构的方向入手，将复杂的软件内部逻辑解耦，提高开发、维护与测试的效率。

2018 年 4 月 30 日星期一　　第九周

微服务模型总体架构将建立任务与执行任务的逻辑关系拆分。客户端在这里是抽象的概念，主要向主节点发送需要处理的任务。独立的主节点主要进行接收任务、创建任务、响应任务与回复任务结果的功能。主节点分解任务规模，将小规模的任务分发到每一个计算节点上。分布式部署的计算节点可以与主节点进行稳定高效的通信，计算节点集群之间也实现了互相通信。