Docker 容器迁移技术在水利监测系统中的应用研究

Application Research of Docker Container Migration Technology in Water Monitoring System

刘基建

Jijian Liu

华北水利水电大学　中国·河南 郑州 450000

North China University of Water Resources and Electric Power, Zhengzhou, Henan, 450000, China

摘　要：随着国民经济的高速发展，水利工程在国民经济中所起的作用越来越大。在水利工程中水利监测必不可少。为了保障水利监测系统的实时稳定性，可以将水利监测系统部署在 Docker 容器中。当部分服务节出现故障时，Docker 容器的在线迁移技术可以保障服务的正常运行，维持系统的稳定监测。

Abstract: With the rapid development of national economy, water conservancy project plays an increasingly important role in national economy. Water conservancy monitoring is indispensable in water conservancy projects. In order to ensure the real-time stability of the water conservancy monitoring system, the water conservancy monitoring system can be deployed in the Docker container. When some service sections fail, the online migration technology of Docker container can guarantee the normal operation of the service and maintain the stable monitoring of the system.

关键词：水利监测；Docker 容器；在线迁移；稳定监测

Keywords: water conservancy monitor; Docker container; online migration; stability monitoring

DOI: 10.12346/etr.v4i6.6235

1 相关背景与研究现状

中国是一个自然灾害多发国家，并且河流、湖泊众多。随着国民经济的高速发展，水利工程在国民经济中所起的作用越来越大，而在水利工程中影响最大、最广泛的是防汛问题，而水文水利监测报警系统通过在线实时监测，能够及时反映各水域的水文特征，以便相关部门了解状况，做出安排，防范各大灾害事故的发生 [1]。

水利监测系统的稳定性关乎监测结果的稳定正确。如何保证水利监测系统长时间的稳定运行，在系统需要升级改进时如何保障监测服务的不中断是个至关重要的问题。

随着虚拟化技术和云技术的发展，容器作为一种资源占用少，启动速度快、不受平台限制的虚拟化技术，正在受到广泛使用。Docker 容器就是最典型的代表。2008 年，Linux 容器（Linux Containers,LXC）项目启动，通过把 CGroups、Namespace 以及 chroot 等技术融合，提供了一套完整的容器方案。将水利监测系统部署在 Docker 容器集群服务中，可以保证系统环境的稳定。在系统服务出现故障或者需要升级的时候，使用 Docker 容器的在线迁移技术可以保障服务的持续稳定。

根据容器迁移过程中是否有明显宕机时间，将容器的迁移分为离线迁移和在线迁移。国内外针对 Docker 容器的在线迁移提出了不少解决方案。最早由 Virtuozzo 团队提出商用的容器迁移解决方案，但其需要对内核进行修改来支持容器相关进程的迁移，缺乏普遍性，无法推广使用。后来启用了 CRIU 项目，其被开发的主要目的是用来取代在内核态的 checkpoint/restore，可以在用户空间（UserSpace）内实现进程的迁移。

CRIU（Checkpoint/Restore In Userspace）是一个在 Linux 用户空间上实现了检查点创建 / 恢复功能的软件。CRIU 可以用在应用程序的热迁移、快照，加速程序的进程等。P.haul 团队也是基于 CRIU 技术，根据 OCI 标准来实现容器的在线迁移。Voyager 也提出了一种借助网络文件挂载

【作者简介】刘基建（1996-），男，中国江苏盐城人，硕士，从事大数据与智能信息处理研究。

（NFS）的方式，将源主机的 rootfs 挂载到目的主机上 [3]。在对目的主机上的容器进行恢复时，直接通过 NFS 从源端拉取 rootfs 所需的数据。Ma 等针对边缘计算及移动应用场景，通过 Docker 容器迁移实现跨边缘服务器的高效服务切换。该系统借助 Docker Hub 完成容器镜像层和容器层的迁移，使用 CRIU 创建检查点完成运行时环境迁移，最后重载 Docker Daemon 恢复容器运行状态。

已有的容器在线迁移方案虽然可以完成容器的迁移，但还存在一定的问题。比如会丢失容器的镜像分层特征，不利于迁移完成后的进一步开发。论文从实际出发，通过分析水利监测系统的实际运行需求，设计了一套可以实现容灾备份的水利监测系统的预迁移方案。

2 水利监测系统与 Docker 容器在线迁移相关分析

2.1 水利监测系统与 Docker 容器的适应性分析

水利监测系统作为水利建设工程中的重要部分，它的持续稳定性尤为重要。传统的水利监测系统由信息采集、信息传输、信息管理和信息服务四部分组成。其中信息管理和信息服务需要系统的持久稳定 [2]。Docker 容器可以为水利监测系统提供稳定的系统环境。只需要配置一次，即可使用安全稳定的隔离环境。监测系统所需的大量数据也可以通过 Docker 容器的数据卷完美处理。既可以保证数据的持久性，也可以解决数据扩容的问题。

Docker 容器的数据卷可以在容器之间共享和重用。对数据卷的操作是即时性的，可以立即生效。即使容器停止，数据卷中的数据也会持久存储。这对于水利监测系统的稳定可靠必不可少。

通过将水利监测系统部署在 Docker 容器集群中，可以在某个节点出现故障或时，将其状态迁移到新的节点上。并且监测的所有数据可以通过数据卷进行持久化存储，之后可以挂载在需要的监测系统容器上。

2.2 Docker 镜像

Docker 镜像是 Docker 的核心组件。容器的所有运行时环境和依赖都被保存在相关的镜像文件中。通过这个特征，容器可以跨平台部署应用，不需要每次都配置系统环境，实现“一处构建，处处运行”。

Docker 镜像文件是一种层级结构，通过联合挂载和父子间的层级关系，Docker 可以在不同的镜像文件中共享同一个镜像层。这极大地减少了存储空间的消耗。Docker 容器镜像由多个镜像层组成，在所有只读镜像层的最上面是可读写的容器层。每个镜像层只会保存被修改过增量数据。新增的镜像层会被放在最上面的读写层中。当需要删除层文件时，由于它们都是只读的，aufs 使用 whiteout 机制，通过在读写层建立对应的 whiteout 来隐藏下面的镜像层。同样，当需要修改镜像层文件时，Docker 会将对应的镜像层内容拷贝到可读写的容器层进行修改。这种技术就叫做写时拷贝 [4]。

2.3 Docker 容器管理架构

Docker 容器之所以受到广泛关注，不仅在于其镜像文件的写时复制和联合挂载，还有一个重要的原因是它对 Docker 容器丰富且强大的管理能力。在早期 Docker 版本中，Docker 通过 libcontainer 来实现对容器的管理。Libcontainer 通过 namespace，cgroup 及文件系统来进行容器控制。Namespaces 负责容器之间的隔离。Cgroups 负责管理容器使用的资源，目前 Docker 独立出两个核心组件：Containerd 和 Runc。Containerd 是满足了 OCI 标准的容器管理组件，其主要职责是镜像管理和容器执行。Docker Daemon 通过 grpc 接口与 Containerd 通信，屏蔽 Docker Daemon 对下面结构变化的感知。Runc 则是基于 libcontainer 实现对容器资源管理等功能 [5]。

Driver 是 Docker 架构中的驱动模块。通过 Driver 驱动，Docker 可以控制容器内各项功能的正常运行。如 Graph 负责镜像的存储，execdriver 则负责容器的执行。networkdriver 负责容器和主机的网络通信，最常用的网络驱动有 bridge，overlay，macvlan 这三种类型。

3 水利监测系统容器的在线迁移方案

通过将水利监测系统部署在 Docker 容器中，可以保证监测系统稳定的运行环境。但当其所在节点需要升级更新时，我们需要将迁移到新的节点上并保持迁移之前状态的连续性。根据 OCI 规范，每个容器镜像都存储在一个文件系统包中，在解压后成为主机文件系统上的另一个目录 rootfs。在容器实例化时，所有运行时环境更改和数据更改默认都保留在 rootfs 中。因此，迁移水利监测系统容器涉及发现容器的所有数据端点，并且除了其内存状态之外，将它们的状态从源主机一致地转移到目标主机。可以通过 CRIU 在用户空间中实现监测系统内存中状态的迁移。此外，监测系统的数据存储都可以通过从源主机卸载并在目标主机上安装来迁移。

3.1 镜像相关文件迁移

容器迁移过程中镜像文件的处理非常重要。根据上文分析的镜像文件结构，镜像可以通过镜像层以及一定的层级关系恢复。基于论文提出的容灾备份的容器迁移方案，使用一个节点服务器作为安全备份的数据中心。在容器服务空闲时，通过 Rsync 将水利监测系统所在容器镜像层文件同步到数据中心的镜像仓库中，并将该容器镜像文件的层级关系同时保存在数据中心。这样可以做到一个容灾备份的效果。

镜像文件的层级关系可以通过 layerdb 下的 mounts 目录和 sha256 目录进行回溯，通过栈的数据结构保存镜像层的 chainID。数据中心根据容器 ContainerID 对镜像层文件建立索引。当需要对容器进行在线迁移时，将目的主机挂载在作为数据中心的服务器上。通过读取对应 ContainerID 镜像文

件的层级关系，将 /var/lib/Docker/aufs/diff 中的镜像层关联，从而构建出所需的镜像文件。

3.2 容器运行时迁移

Docker 容器的运行时包括根文件系统 rootfs，内存数据等。根文件系统 rootfs 是 Docker 容器的存储驱动（aufs、overlay 等）通过联合挂载镜像层，以及结合写时拷贝技术，对外集中展示的全局文件系统。由于 Docker 容器数据的改动最终都体现在容器的读写层上。因此对根文件系统 rootfs 的迁移转化为对水利监测系统容器的读写层进行迭代迁移。

Docker 容器的内存数据是通过 CRIU 来进行迭代迁移的。论文所提出的容灾备份迁移方案是通过计划迁移，将内存文件使用 pre-copy 机制，迭代同步到目标服务器上，并存入相关容器 ID 文件夹下。预拷贝内存（Pre-copy）是指通过不断的迭代，将源主机的内存数据发送至目的主机。首次迭代发送所有内存数据，接下来每一轮迭代只发送上一轮之后修改的内存页面数据。最后一轮是停机拷贝阶段。源主机上的进程被中断，防止内存更新，然后将最后一轮改变的内存数据拷贝到目的主机上。

在实现上，采取有限循环，对循环的次数和结果进行控制。当内存传输完成后，开始同步水利监测系统容器层的数据文件，断开源主机对数据源的连接，并在目的端主机上恢复其监测数据卷的挂载。至此，水利监测系统容器运行时的所有相关数据就完成了迁移。

3.3 恢复容器运行

当完成镜像文件和容器运行时相关文件迁移后，水利监测系统并不能立刻恢复运行，因为目的端的 Docker 失去了对监测系统容器的上下文信息。

为了恢复水利监测系统的正常运行，需要重载 Docker Daemon，恢复其管理能力。而传统的重载非常耗时，不利于水利监测系统的稳定持久性。本方案通过对 Docker Daemon 源码的分析，使用原有的接口方法，动态加载迁移后目的端水利监测系统容器的上下文信息，使之可以快速恢复运行。首先在目的端通过以迁移的镜像信息验证该容器是否具备所需启动的所有数据，当所需数据完备后，则触发 Docker Daemon 的动态加载。验证所有镜像层文件，并建立其父子关系，将内存文件加载到对应的容器中。只要以正确的顺序将容器信息加载到对应的驱动中，即可恢复目的端 Docker 对容器的管理能力。

4 研究结论

通过将水利监测系统部署在 Docker 容器中，系统在保证稳定运行的同时，可以将其相关数据进行备份容灾。并且在系统需要升级时，通过 Docker 容器在线迁移相关技术保证了监测系统的不间断服务，加强了水利监测系统的实时性和健壮性。这对水利工程建设提升施工效率、保证施工安全与质量具有积极影响。

参考文献

[1] 胡全舟,张华安,李毅男.水利工程的安全监测分析[J].工程技术研究,2017(11):178+192.

[2] 吕中东.水利水电工程施工中的新技术应用模式[J].科学技术创新,2020(6):2

[3] Shripad N, Sahil S, Nilton B, et al. Voyager: complete container state migration. In Proceedings of th 26th ACM Symposium on Operating Systems Principles(SOSP’17)[J]. Shanghai, China: ACM,2017(3):2137-2142.

[4] Biederman E W, Networx L. Multiple instances of the global linux namespace[J]. Proceedings of the Linux Symposium,2006,1(1):101-112.

[5] Kozhirbayev Z, Sinnott R, Richard. Aperformance comparison of container-based technologies for the cloud[J].Future Generation Computer Systems,2017,68(2):175-182.