Drone Mapping: Hyperscaling a service with Kubernetes on ARM

LEADING COLLABORATION IN THE ARM ECOSYSTEM

March 2017

Linaro TECHNOLOGIES

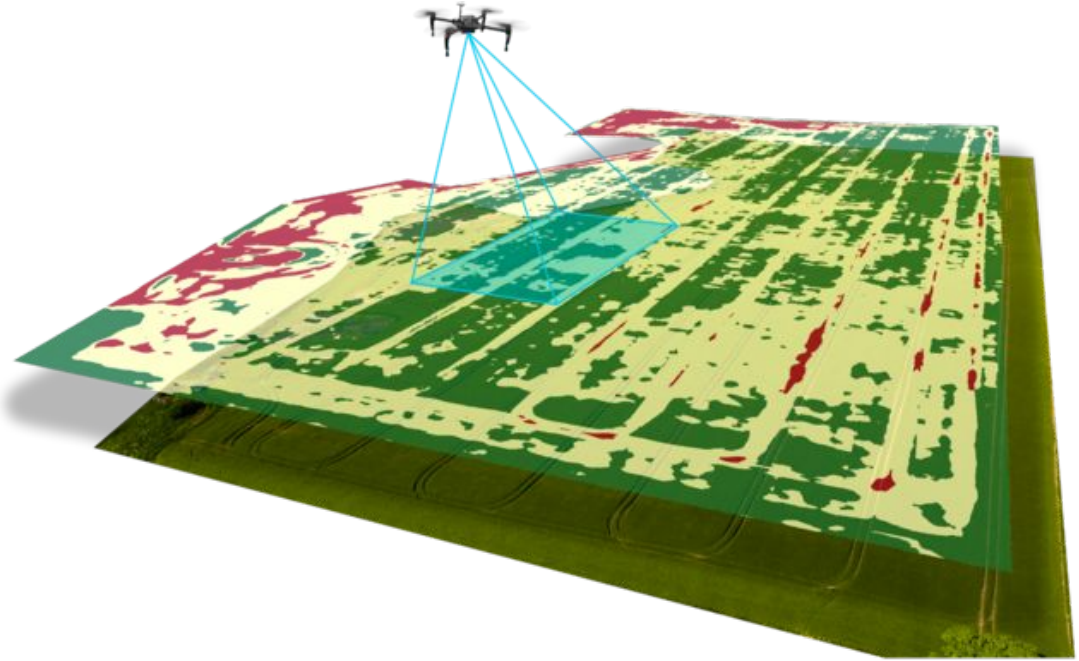
Introduction

- Tyler Baker
  - Principal Software Engineer
  - Linaro Technologies Division
    - A small team within Linaro focusing on open source end-to-end solutions
      - Firmware to Cloud
      - IoT to Enterprise
  - Piloting model aircraft since 2004
    - Fixed wing, quadcopters, tricopters
    - Long range FPV systems
    - 3D printed aircraft

- How to reach me
  - tyler.baker@linaro.org
  - Freenode IRC: tyler-baker

Drone Mapping Service

A user provides aerial images which are processed by the service to create a 2D geolocated orthomosaic image and a 3D textured model



Kubernetes (k8s)

Kubernetes (commonly referred to as "K8s") is an open source container cluster manager originally designed by Google and donated to the Cloud Native Computing Foundation. It aims to provide a "platform for automating deployment, scaling, and operations of application containers across clusters of hosts". It usually works with the Docker container tool and coordinates between a wide cluster of hosts running Docker.

Source: https://en.wikipedia.org/wiki/Kubernetes

Why Kubernetes?

- Load Balancers for Services
  - Creates a virtual IP which other services use
  - Routes traffic to underlying pods (containers) seamlessly
  - Allows the deployment to scale a service across N number of nodes
- Autoscaler (Horizontal Scale)
  - Automagically scales your load balanced service and/or replica set
    - User defines a maximum/minimum number of pods for a service
    - Metrics used to scale out or in
      - Number of requests per second
      - CPU load percentage
      - Custom metrics can be defined
- Health Checks
  - Checks if services are responding
  - Moves pods and restart if necessary
- Storage Abstraction
  - NFS/CEPH/S3/FLOCKER
- Pod Networking
  - Tons of options like Weave (L2/L3), Calico (L3), etc

Kubernetes on ARM64

There were a few missing pieces to the puzzle…

The biggest one was the lack of an ARM64 pod network. So I rebuilt all the weave net daemon set containers from scratch.

kubectl apply -f https://raw.githubusercontent.com/EmbeddedAndroid/weave-arm/master/weave-kube.yml

Now we are routing packets!

The ARM64 Kubernetes Cluster

- Three physical nodes
  - 32 ARM64 Cores with 64GB of system memory
  - 1TB 7200RPM Hard Disk
  - Upstream Tianocore UEFI booting with ACPI
  - Enterprise Reference Platform
    - v4.9 RPK
    - Debian and CentOS 7 packages available, so this isn't necessary
- Kubernetes v1.5
  - Weave Pod Networking
- NFS Server providing persistent storage for pods
  - Not recommended for large scale deployments
  - CEPH would be a good choice for larger deployments
- Docker v1.13 using the overlay2 storage driver
  - IPv6 support

Open Drone Map

- Load Balanced HTTP Front End
  - 1 replication controller
  - 6 replica pods running NGINX
  - Traffic flows through the replication controllers virtual IP and is distributed amongst the pod replicas
  - Pods forward traffic to the back end service
- Load Balanced UWSGI Back End
  - 1 replication controller
  - 6 replica pods running Open Drone Map Django application serving UWSGI via Gunicorn
- Load Balanced and Replicated Database
  - pgpool-II middleware for brokering client/server interactions
  - 6 pods running PostgreSQL
  - Deployed with helm!

Open Drone Map Processing Nodes

- Processing nodes do the real work
  - Pull images, inspect, compile, stitch, and output the final product
  - Provides a swagger API for automation
- They are not stateless :(
  - Stateless services should not be put behind a load balancer
    - Requests are not always forwarded to the same node
    - For example request for status updates cannot be guaranteed to hit the same node every time
  - This means they cannot autoscale, another service must be in charge of the number of processing nodes for true horizontal scale
- The good news
  - They can self register with the backend via the swagger API
  - Once a processing node is started it registers itself with a the backend automatically

With Kubernetes it’s fairly straightforward to create a highly scalable and available Django web application.

However, Open Drone Map is a very large, complex software project with many dependencies that requires large amounts of compute that brings anything but the highest powered servers to a crawl.

Open Drone Map Image Processing Components

- CMVS-PMVS
  - Takes the output of a structure-from-motion (SfM) software as input, then decomposes the input images into a set of image clusters of manageable size.
- Ceres Solver
  - Open source C++ library for modeling and solving large, complicated optimization problems.
- Ecto
  - Is a dynamically configurable Directed Acyclic processing Graph (DAG) framework.
- OpenCV
  - Open Source Computer Vision Library
- OpenGV
  - Is a collection of computer vision methods for solving geometric vision problems
- OpenSfM
  - Open source Structure from Motion pipeline
- PDAL
  - Is a C++ BSD library for translating and manipulating point cloud data.

Questions?