MeerKAT & Mesos
ORCHESTRATING MEERKAT’S DISTRIBUTED SCIENCE DATA PROCESSING PIPELINES

PRESENTER: Anton Joubert
CO-AUTHOR: Bruce Merry

THBPA04
Correlator Beamformer

Science Data Processor

Archive

Control And Monitoring

2.2 Tb/s

600 Gb/s

20 Gb/s

Control flow ("small" data)

Data flow ("big" data)

www.ska.ac.za
Where we were
Where we were

configure(...)
Where we were
Where we were

**SDP CLUSTER**

- **mc1.sdp**
  - controller
  - dockerd
  - redis
  - cam2telstate
  - filewriter

- **ing1.sdp**
  - dockerd
  - ingest
  - disp

- **cal1.sdp**
  - dockerd
  - cal

**CAM**

configured

www.ska.ac.za
Where we were

Problems:
• Manual reconfiguration required if a host unavailable
• Host utilisation not well balanced
• Does not scale

How can we automate this?
Container orchestration tools

- Docker
- Swarm
- kubernetes
- Apache Mesos
- HashiCorp Nomad
- And many more...

www.ska.ac.za
Mesos Architecture

ZK: ZooKeeper
(distributed key-value store)

www.ska.ac.za
Mesos Architecture

ZK: ZooKeeper
(distributed key-value store)
Mesos Architecture

Agent (1)
Framework
Executor
Task

Agent (2)
Framework
Executor
Task

Agent (3)
Framework
Executor
Task

Agent (4)
Framework
Executor
Task

ZK: ZooKeeper
(distributed key-value store)

www.ska.ac.za
Why not use an existing framework?
The world according to Mesos

CPU
Memory
Disk
The real world

Machine (252GB total)

NUMANode P#0 (126GB)

- Package P#0
  - Core P#0
    - PU P#0
  - Core P#1
    - PU P#2
  - Core P#4
    - PU P#4
  - Core P#5
    - PU P#6

- PCI 1000:005d
  - sda
  - sdb

- PCI 10de:17c2
  - card0
  - renderD128

- PCI 14e4:168a
  - em1

- PCI 14e4:168a
  - em2

NUMANode P#1 (126GB)

- Package P#1
  - Core P#0
    - PU P#1
  - Core P#1
    - PU P#3
  - Core P#4
    - PU P#5
  - Core P#5
    - PU P#7

- PCI 15b3:1003
  - p4p1
  - p4p2
  - mlx4_0

NUMA: Non-Uniform Memory Architecture
NUMA support

E.g. run this task using 3 cores on the same NUMA node

**Cores & Topology**

Custom resources: cores: [0-7]

Custom attributes: node.numa: [[0,2,4,6], [1,3,5,7]]

**Scheduler**

Scheduler assigns cores from same node for NUMA-sensitive tasks

**Docker**

Use option:

--cpuset-cpus

Image from: http://en.community.dell.com/
GPU support

E.g. run this task on a Titan X GPU, using 25% compute and 3GB RAM, on same NUMA node as CPU cores

GPU details

Custom resources:  
node.gpu.0.compute: 1.0
node.gpu.0.mem: 8192

Custom attributes: NUMA node, GPU type, /dev entries, CUDA info, etc.

Scheduler

Scheduler assigns node with correct GPU and capacity, and cores from same NUMA node

Docker

E.g. pass in /dev/nvidia0 as Docker argument
Also have Docker images optimised for some GPUs

Note: Resource limits are not enforced by operating system
Network support

E.g. run this task on an ibverbs-capable NIC on the CBF network, allowing 27 Gb/s in-bound bandwidth, and using a single NUMA node

NIC details

Custom resources:
- node.interface.0.bandwidth_in: 40e9
- node.interface.0.bandwidth_out: 40e9

Custom attributes:
- IF name, /dev entries, network segment, NUMA node

Scheduler

Scheduler assigns node with correct NIC and capacity, and cores from same NUMA node

Docker

E.g. pass in /dev/em1 as Docker argument

Note: Resource limits are not enforced by operating system
The good

- *PyMesos* Python package hides complexity of the HTTP API
- Mesos cleans up after framework crashes
- Mesos UI handy for debugging
- Mesos developers are friendly and responsive
- Describing placement policies with code makes for ultimate extensibility
- Only 5k lines of Python code for our custom scheduler
The not so good

- Attribute values can only contain: A-Za-z0-9_/.- (so had to use base64 encoding)
- Changing attributes of an agent requires recovery step (kill tasks and restart)
- No GUI support for custom resources
- Mesos not as rock-solid as expected (during our development)
- Fault-tolerance is still hard
Conclusion

- SDP pipelines are now very scalable
- Easy to recover from a node failure
- Custom scheduler allows us to maintain optimal performance
- Could be improved for much higher availability
Questions?
Thank you

More details in the paper: THBPA04