SKA SDP-COMP Middleware: The intersect with commodity computing

Piers Harding // February, 2017

catalyst
open source technologists
Overview

- SDP Middleware – why is this important
- What are the options
- Middleware – where is industry heading
- What are NZA doing
Quick recap of SKA Context for SDP

Murchison Region, AU; Karoo Desert, SA

Two independent SDPs

Ref. J Taylor - 2016
Middleware: Where is it?

- **App1**, **App2**, **App2.1**, **App2.2**, **App3**, **App4**

**Container Management** – metrics, logging, monitoring

**Container Orchestration**

**Container Engine**

**Shared Services**

- **OS Management** – metrics, logging, monitoring, security, accounting
- **Common Components** - Message Queue, *DBMS, Key/Value Store*

**Base OS**

**Software Defined Infrastructure**

- **Compute**
- **Storage**
- **Network**

**Platform Management** – hardware inventory, bootstrap, monitoring

- **Containerised Apps**
- **Containerised utilities**
- **Container scheduler**

**Managed resources**
SDP COMP Middleware:
why is this important?
SDP COMP Middleware: opportunities

- Deal with the uncertainty and pain of growth
- The opportunities to do things differently
- Adopt modern software architecture and management
- Less about jobs (batch) and more about services
- Decouple bespoke software from hardware and platform (as much as possible)
- Guard against becoming a single purpose platform
- Position to take advantage of future innovation
Middleware: Project aspirations

• Commodity computing – COTS
• Reduce investment in bespoke development - “let others do as much as possible”
• Control costs – initial and ongoing
• Openness – preference for open source and open standards – enable participation
Middleware: SDP developer infrastructure

- External entities will need to write code to insert in the pipelines
- Must define APIs and interfaces, publish and give reference implementations
- Provide development tools
- Testing environments
- “encapsulate in tools and environments so I can run at home”
Middleware: what are the options?
Middleware: Focus on containerisation—a modern software paradigm
Containerisation: how it (should!) works

- In kernel virtualisation using cgroups, and namespaces
- Containers launched from immutable images – share layers
- Packaging and dependency encapsulation
- Philosophy:
  - 1 container == 1 service (preferably 1 process)
  - Immutability – IO to services, external config
  - Cattle not Pets
- Efficient – operational density increased - no OS boot, small images
- Enables cohabitation – heterogeneous hosts, and container versions
Containerisation: Why should you care?

- Changes the way systems are architected and managed - SDLC
- Focuses on delivering services that are:
  - Robust (self-healing)
  - Scalable – resource aware, and scheduling capabilities
  - High availability – continuous operation
- Developers closer to the platform – environmental consistency
- Delegate all but specific operational functions to the platform
Container Orchestration: Why should you care?

• Centralises core functions such as:
  • Telemetry
  • Monitoring
  • Logging
  • Scheduling
  • Scaling
• Focuses on resources as services
Middleware and developers:

- Pipeline software developers interact with the middleware
- It becomes their API
- It defines the application process flow and their design limitations
- And their workflow (SDLC) – dev, test, prod, packaging, sharing, debugging
SKA: could be Service oriented by design

- Many characteristics of a service:
  - Soft real-time
  - Tight performance requirements
  - Scalability and scheduling key – service flavours
  - Continuous operation is an aspiration
- Unknown future processing requirements
SKA-SDP: But, there is a problem!

- Data rates are vast – 11Tbps
  - Per node – 254MB/s, 6TB temp, 3TB shared*
- The buffer storage
- Critical process overlap
- We cannot terminate nodes without:
  - Load balancing the ingest
  - Using shared storage for the buffer
  - Service recovery/resume

* 6 hour observation - 51GB Grids * 52 max
Observation flow & overlap
Focus on Imaging Pipeline (biggest)

Continuous calibration - soft real-time

Observation 1 – 6 hours
Data Ingestion
Image Pipeline
Observation 2 – 6 hours
Data Ingestion
Image Pipeline
Observation 3 – 6 hours
Data Ingestion
Image Pipeline
Observation 4 – 3 hours
Ingest
Image
Observation 3 – 6 hours
Data Ingestion
Image Pipeline
Ingest without processing!
Middleware: where is the industry heading?
Traditional HPC

- Is batch
- generally doesn’t have real-time considerations – Mesos
- Infrastructure Down time is OK (generally not considered)
Emerging HPC Technologies

- Approaching real-time
- Aligned with Advanced Analytics
- Focus on operational efficiency
- Container based technologies – isolation, density, replication
- Service oriented – Spark, ImpalaDB, Kubernetes, Docker Swarm
- Evolving fast – Google GCE with GPUs, AWS ECS with GPUs
- Coming: serverless architecture, FaaS - AWS Lambda (exists but no GPU), OpenWhisk
- Resurgence in compiled languages – Go + GPU
- Not there yet
Middleware: what are NZA doing?
Investigating design options

- Centred on Containerisation, Storage, Telemetry & Logging – allocated tasks from the SKAO

- Also looking into:
  - Platform Management – Software defined Infrastructure from the SysAdmin and DevOps point of view
  - Orchestration & Scheduling
  - Solution architecture
Thank you

- Piers Harding – piers@catalyst.net.nz