

Prototyping a New Ingest Pipeline System for SDP

Drew Devereux

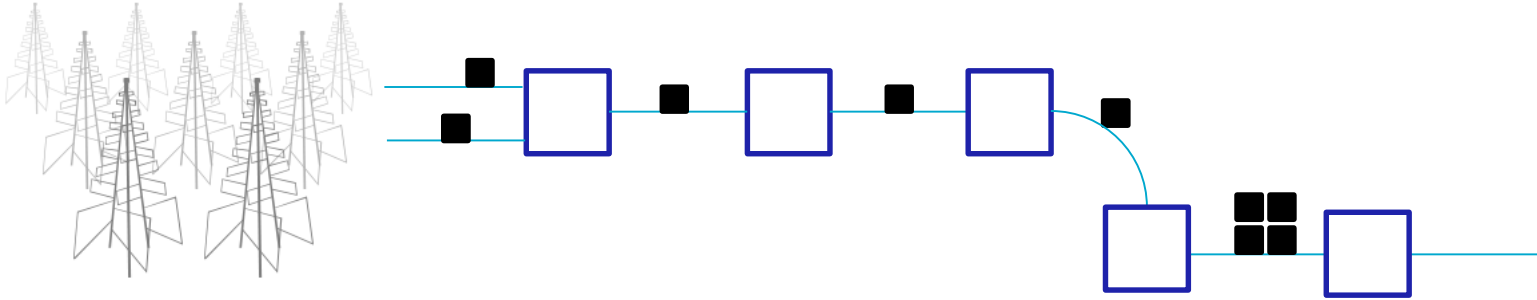
16 February 2018

CSIRO ASTRONOMY AND SPACE SCIENCE

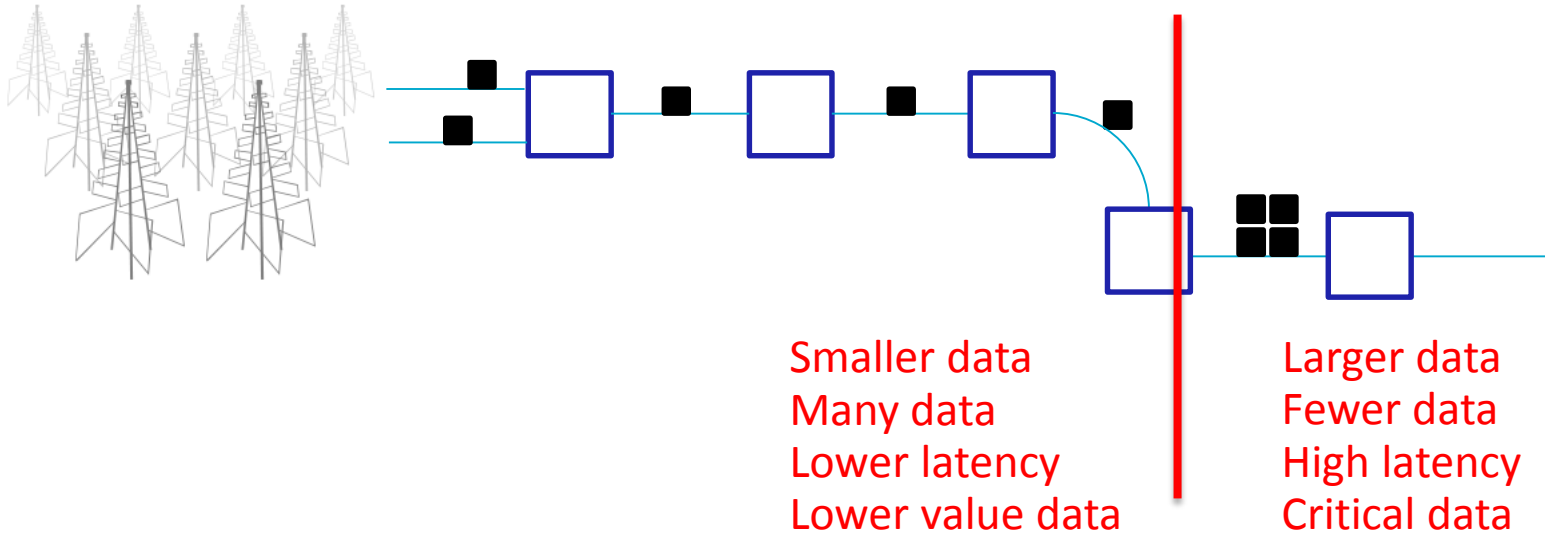
www.csiro.au



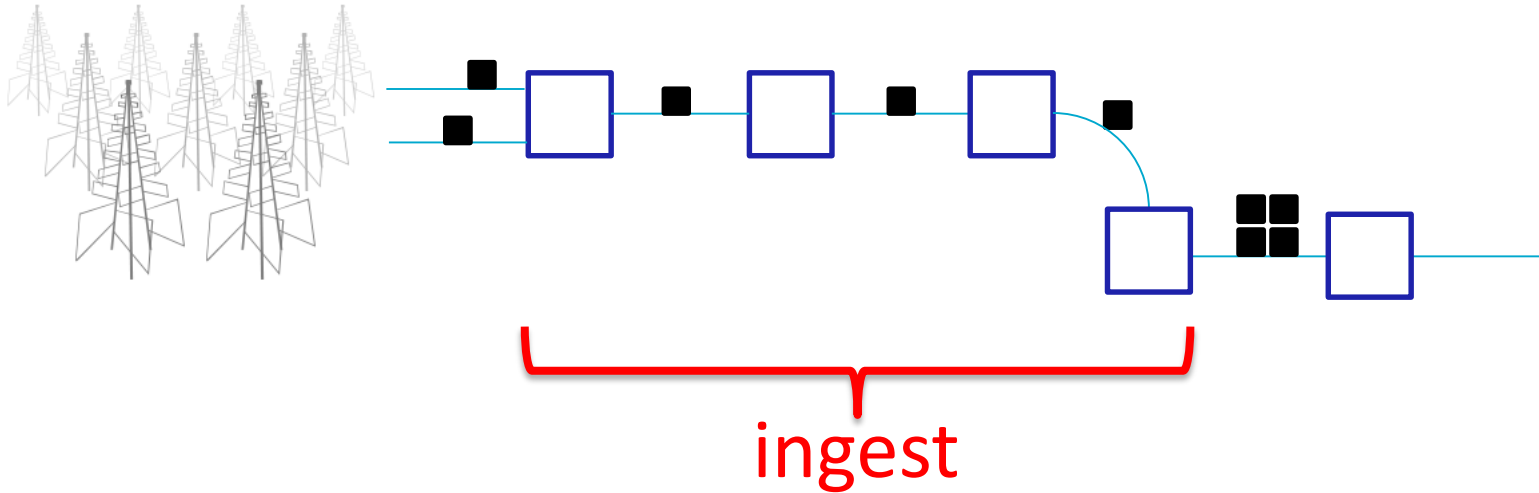
Problem definition



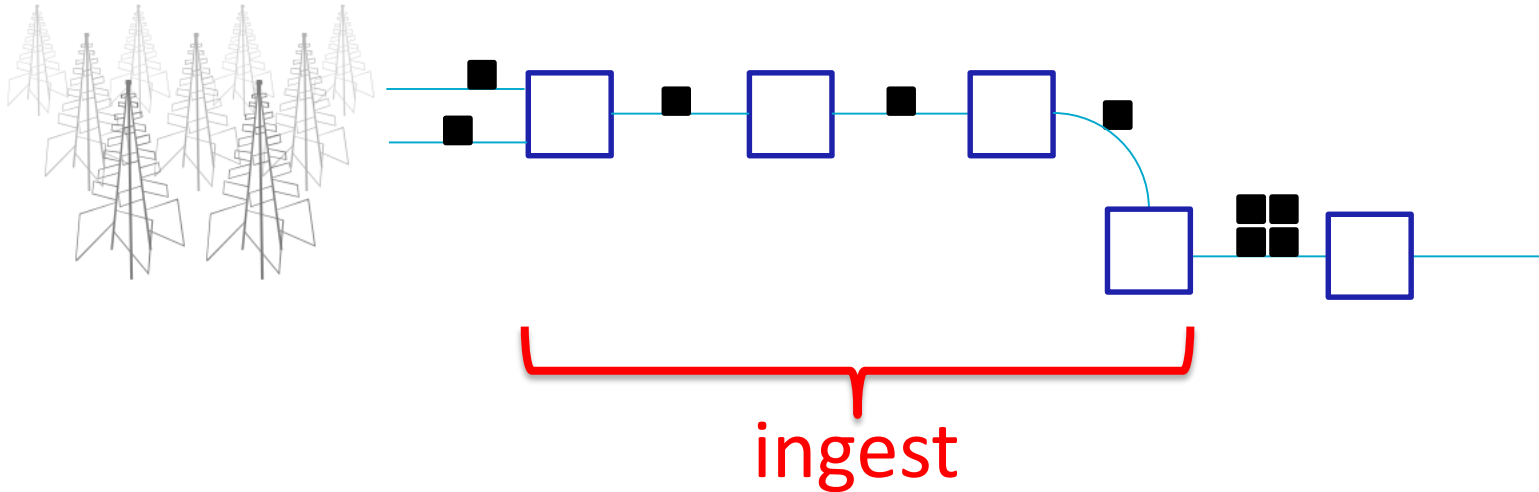
Problem definition



Problem definition



Problem definition



- Define the requirements of the ingest stage
- Determine upon an architectural style
- Design, implement and test a prototype

Requirements

- High throughput
 - Not necessarily real-time latencies, but must be able to keep up
 - We need ingest to finish soon enough after the observation that the rest of the system has time to do its processing

Requirements

- High throughput
 - Not necessarily real-time latencies, but must be able to keep up
 - We need ingest to finish soon enough after the observation that the rest of the system has time to do its processing
- Available / reliable / fault-tolerant
 - Dropping an observation because a buffer is full – no problem
 - Crashing because a buffer is full – big problem

Requirements

- High throughput
 - Not necessarily real-time latencies, but must be able to keep up
 - We need ingest to finish soon enough after the observation that the rest of the system has time to do its processing
- Available / reliable / fault-tolerant
 - Dropping an observation because a buffer is full – no problem
 - Crashing because a buffer is full – big problem
- Configurable / composable
 - The configuration of the observation determines the configuration of the processing system.

Requirements

- High throughput
 - Not necessarily real-time latencies, but must be able to keep up
 - We need ingest to finish soon enough after the observation that the rest of the system has time to do its processing
- Available / reliable / fault-tolerant
 - Dropping an observation because a buffer is full – no problem
 - Crashing because a buffer is full – big problem
- Configurable / composable
 - The configuration of the observation determines the configuration of the processing system.
- Manageable
 - Ability to monitor the system and intervene if necessary

Architectural approach

- Service oriented architecture (SOA)
 - “Services” are independent, isolated components that interact only via a communication protocol
 - Communication is usually asynchronous to maximise decoupling e.g. request/response, publish/subscribe
 - Isolation of services makes SOA systems configurable, reliable, fault-tolerant, deployable.

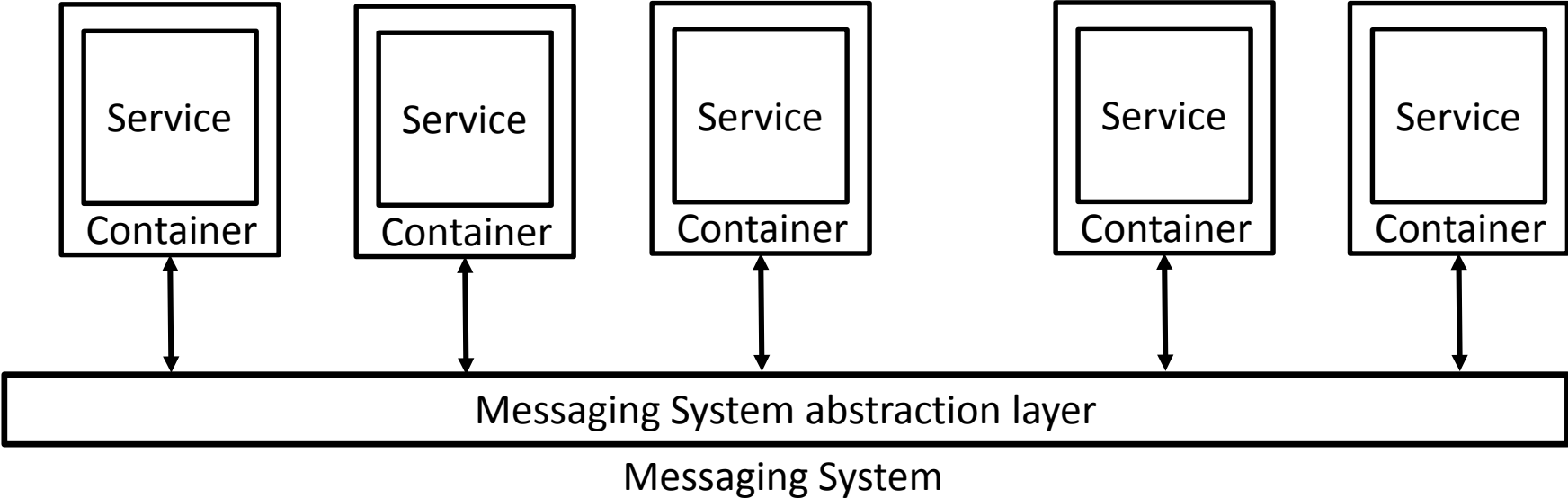
Architectural approach

- Service oriented architecture (SOA)
 - “Services” are independent, isolated components that interact only via a communication protocol
 - Communication is usually asynchronous to maximise decoupling e.g. request/response, publish/subscribe
 - Isolation of services makes SOA systems configurable, reliable, fault-tolerant, deployable.
- Microservices
 - Each service does one thing well



Architectural approach

- Service oriented architecture (SOA)
 - “Services” are independent, isolated components that interact only via a communication protocol
 - Communication is usually asynchronous to maximise decoupling e.g. request/response, publish/subscribe
 - Isolation of services makes SOA systems configurable, reliable, fault-tolerant, deployable.
- Microservices
 - Each service does one thing well
- Streaming

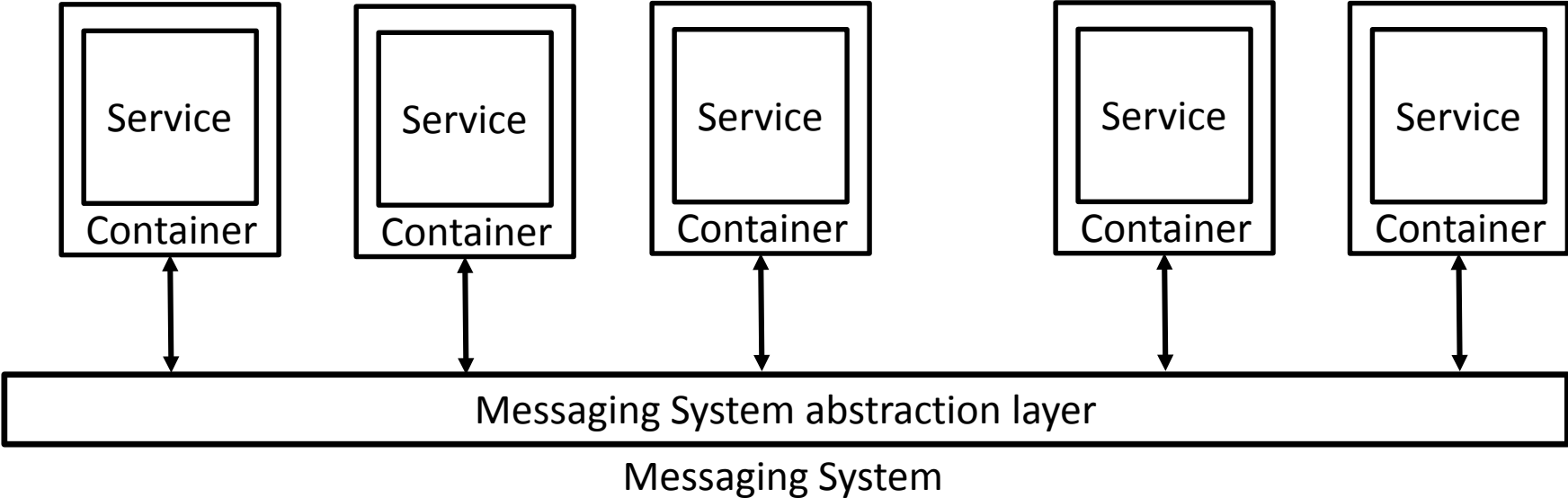
Architectural approach



Streaming engines

												
	Flume	NiFi	Gearpump	Apex	Kafka Streams	Spark Streaming	Storm	Storm + Trident	Samza	Flink	Ignite Streaming	Beam (*GC DataFlow)
Current version	1.6.0	0.6.1	incubating	3.3.0	0.9.0.1* [available in 0.10]	1.6.1	1.0.0	1.0.0	0.10.0	1.0.2	1.5.0	incubating
Category	DC/SEP	DC/SEP	SEP	DC/ESP	ESP	ESP	ESP/CEP	ESP/CEP	ESP	ESP/CEP	ESP/CEP	SDK
Event size	single	single	single	single	single	micro-batch	single	single	single	single	single	single
Available since (incubator since)	June 2012 (June 2011)	July 2015 (Nov 2014)	(Mar 2016)	Apr 2016 (Aug 2015)	Apr 2016 (July 2011)	Feb 2014 (2013)	Sep 2014 (Sep 2013)	Sep 2014 (Sep 2013)	Jan 2014 (July 2013)	Dec 2014 (Mar 2014)	Sep 2015 (Oct 2014)	(Feb 2016)
Contributors	26	67	19	53	160	838	207	207	48	159	56	60
Main backers	Apple Cloudera	Hortonworks	Intel Lightbend	Data Torrent	Confluent	AMPLab Databricks	Backtype Twitter	Backtype Twitter	LinkedIn	dataArtisans	GndGain	Google
Delivery guarantees	at least once	at least once	exactly once at least once (with non-fault-tolerant sources)	exactly once	at least once	at least once (with non-fault-tolerant sources)	at least once	exactly once	at least once	exactly once	at least once	exactly once*
State management	transactional updates	local and distributed snapshots	checkpoints	checkpoints	local and distributed snapshots	checkpoints	record acknowledgements	record acknowledgements	local snapshots distributed snapshots (fault- tolerant)	distributed snapshots	checkpoints	transactional updates*
Fault tolerance	yes (with file channel only)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes*
Out-of-order processing	no	no	yes	no	yes	no	yes	yes	yes [but not within a single partition]	yes	yes	yes*
Event prioritization	no	yes	programmable	programmable	programmable	programmable	programmable	programmable	yes	programmable	programmable	programmable
Windowing	no	no	time-based	time-based	time-based	time-based	time-based	time-based	time-based	time-based	time-based	time-based
Back-pressure	no	yes	yes	yes	N/A	yes	yes	yes	yes	yes	yes	yes*
Primary abstraction	Event	FlowFile	Message	Tuple	KafkaStream	DStream	Tuple	TridentTuple	Message	DataStream	igniteDataStreamer	PCollection
Data flow	agent	flow (process group)	streaming application	streaming application	process topology	application	topology	topology	job	streaming dataflow	job	pipeline
Latency	low	configurable	very low	very low	very low	medium	very low	medium	low	low (configurable)	very low	low*
Resource management	native	native	YARN	YARN	Any process manager (e.g. YARN, Mesos, Chef, Puppet, Salt, Kubernetes, ...)	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN	YARN Mesos	integrated*
Auto-scaling	no	no	no	yes	yes	yes	no	no	no	no	no	yes*
In-flight modifications	no	yes	yes	yes	yes	no	yes (for resources)	yes (for resources)	no	no	no	no
API	declarative	compositional	declarative	declarative	declarative	declarative	compositional	compositional	compositional	declarative	declarative	declarative
Primarily written in	Java	Java	Scala	Java	Java	Scala	Clojure	Clojure	Scala	Java	Java	Java
API languages	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	Scala Java Python	Java Python Scala	Java	Java Scala Python	Java NET C++	Java*
Notable users	Meebo Sharethrough SimpleGeo	N/A	Intel Levi's Honeywell	Capital One GE Predix PubMatic	N/A	Kelkoo Localytics AsialInfo Opentable Fairdata Guavus	Yahool Spotify Croupon Flipboard The Weather Channel Alibaba Baidu Yelp WebMD	Klout GumGum CrowdFlower	LinkedIn Netflix Intuit Uber	King Otto Group	GndGain	N/A

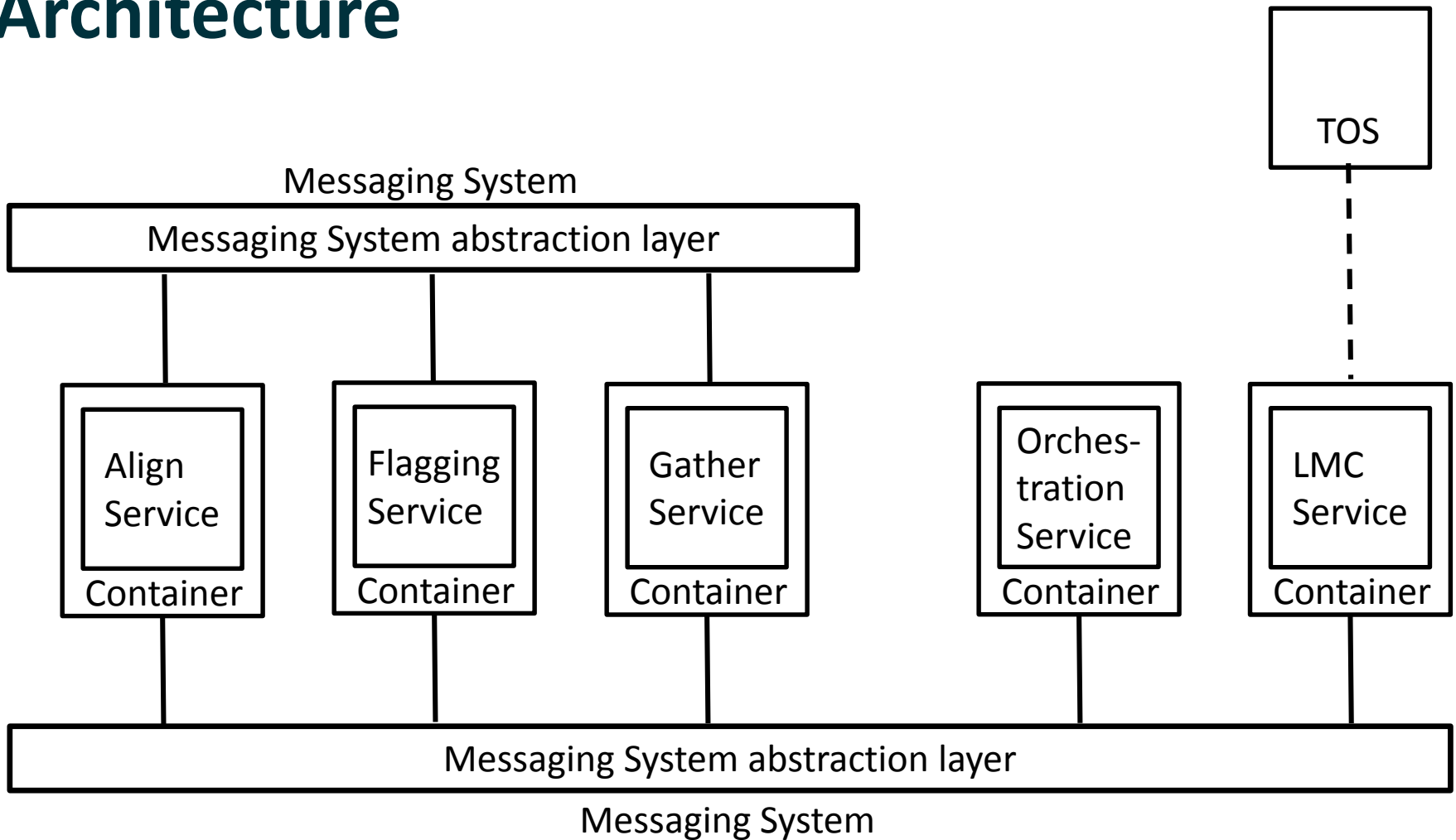
Architectural approach



Elements of the system

- Messaging system
 - publish / subscribe
 - Multiple messaging systems
- Service container
 - Instantiates and runs a service
- Services for
 - Data processing
 - Monitoring and control
 - Orchestration

Architecture



Status and future work

- We have a working design
- We have a prototype, with services that run, process data and communicate over a messaging system
- We have basic monitoring and control
- Configuration-driven orchestration is next

Thank you

CSIRO Astronomy and Space Science
Drew Devereux
Research Scientist

t +61 8 6436 8878
e drew.devereux@csiro.au

CSIRO Astronomy and Space Science
JC Guzman
ATNF Software & Computing Group
Leader

t T +61 8 6436 8569
e juan.guzman@csiro.au

ADD BUSINESS UNIT/FLAGSHIP NAME
www.csiro.au

