Six Principles of Pipeline Design, Taken From The Apollo Missions

Israel Herraiz, Paul Balm
Israel Herraiz

Paul Balm

Google Cloud
“Light Years Ahead: The 1969 Apollo Guidance Computer” – Robert Wills

youtube.com/watch?v=VYI0Kf_1wgk
An Eventful Journey
Similarities to pipelines in the cloud

Launching a pipeline in the cloud is like launching a spacecraft (...almost)

1. Once launched, control is limited
2. Observability depends on preparation
3. Mistakes can be expensive
Similarities to pipelines in the cloud

Launching a pipeline in the cloud is like launching a spacecraft (...almost)
Principle: Use a high-level language

Orbit of command module

Principle: Use a high-level language
Principle: Use a high-level language

[Java][Python][Go]

Cross-language pipeline support

beam.apache.org/documentation/programming-guide/#multi-language-pipelines
2022.beamsummit.org/sessions/beam-cross-language-transforms/
Principle: Divide your program into jobs
Principle: Divide your program into jobs
Principle: Divide your program into jobs
Principle: Divide your program into jobs

Write reusable PTransforms, and structure your pipeline as PTransforms

beam.apache.org/contribute/ptransform-style-guide/
Principle: Restart on failure
**Principle: Restart on failure**

1. Design jobs for gapless processing  
   (error handling, dead letter queue)

2. Considering draining vs. canceling a pipeline*

3. Run a parallel updated pipeline

---

*runner specific feature

cloud.google.com/architecture/building-production-ready-data-pipelines-using-dataflow-deploying

2022.beamsummit.org/sessions/error-handling-asgarde/

github.com/tosun-si/asgarde
Principle: Checkpoint good state
Principle: Checkpoint good state

1. Reshuffles trigger a checkpoint and interacts with I/O
2. Behaviour is Runner dependent: Checkpoint in Dataflow and Flink
Principle: Hardware monitors software
Principle: Hardware monitors software
Principle: Hardware monitors software

cloud.google.com/architecture/building-production-ready-data-pipelines-using-dataflow-planning
sre.google/resources/book-update/data-processing-pipelines/
Principle: Send Telemetry
Principle: Send Telemetry
Principle: Send Telemetry

Beam Metrics*:
→ Counter
→ Distribution
→ Gauge

- Low-level metrics
- Business-level metrics

*Not all metrics are supported by all runners
[beam.apache.org/documentation/runners/capability-matrix/what-is-being-computed/]
Conclusions
Recap: The Six Principles

1. High-level language: leverage cross lang pipelines if necessary
2. Divide and conquer: write reusable PTransforms, compose pipelines
3. Restart on failure: write fault tolerant, gapless, resilient pipelines
4. Checkpoint: reshuffle/shuffling to create backtracking barriers
5. Monitor: define SLOs from the planning phase, monitor accordingly
6. Telemetry: produce business level metrics, use them SLOs too
Conclusions

Pipelines and aircrafts are not exactly the same, but both have to land successfully.

Don't hope for the best. Prepare. Hope is not a strategy.

Further reading:

Building production-ready data pipelines using Dataflow

Design your pipeline
Create your pipeline
Test your pipeline

SRE Data Processing Pipelines
Thank you!