Building a Modern Data Infrastructure - Part 2

Building Out the Strategy

Conway's Law

HOW DO COMMITTEES INVENT?

by MELVIN E. CONWAY

That kind of intellectual activity which creates a useful whole from its diverse parts may be called the *design* of a *system*. Whether the particular activity is the creation of specifications for a major weapon system, the formation of a recommendation to meet a social challenge, or the programming of a computer, the general activity is largely the same.

conclusion

The basic thesis of this article is that organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations. We have seen that this fact has important implications for the management of system design. Primarily, we have found a criterion for the structuring of design organizations: a design effort should be organized according to the need for communication.

Dr. Conway is manager, peripheral systems research, at Sperry Rand's Univac Div., where he is working on recognition of continuous speech. He has previously been a research associate at Case Western Reserve Univ., and a software consultant. He has an MS in physics from CalTech and a PhD in math from Case.

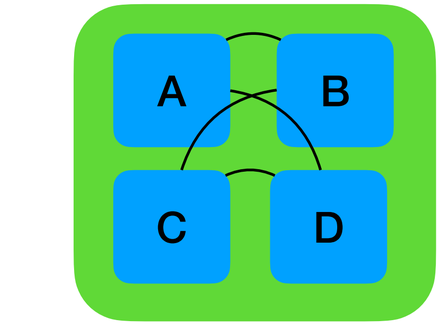
Product , Teams, Delivery

Low Cohesion, High Coupling

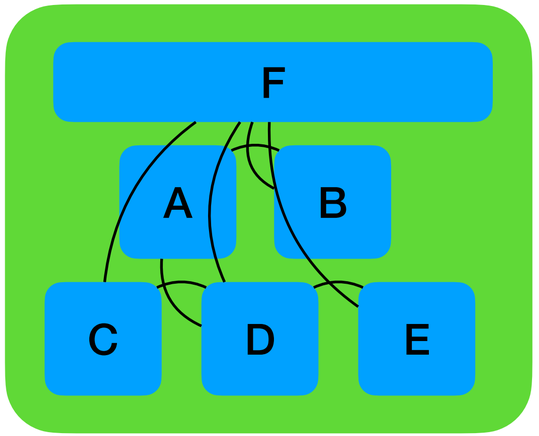
MVP

on-premise

Monolithical



Singular Cloud



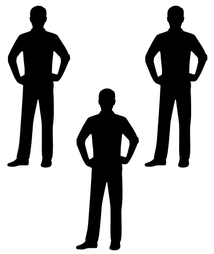
Distributed *

Multi-Cloud

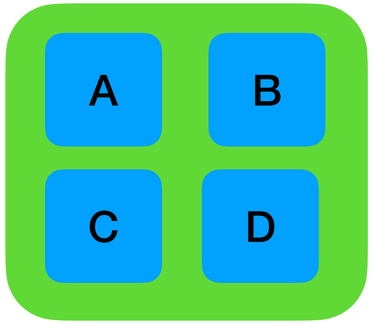
Good

Team

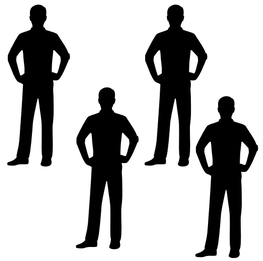
MVP



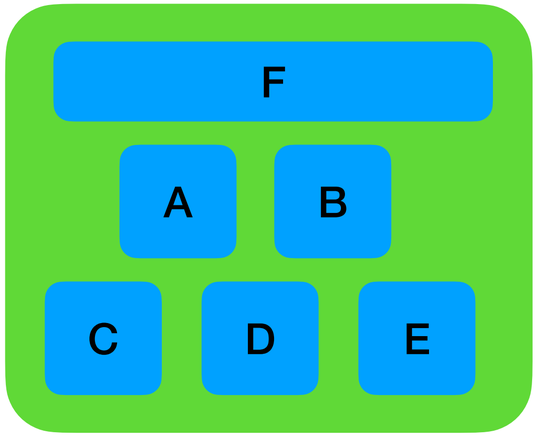
Bad



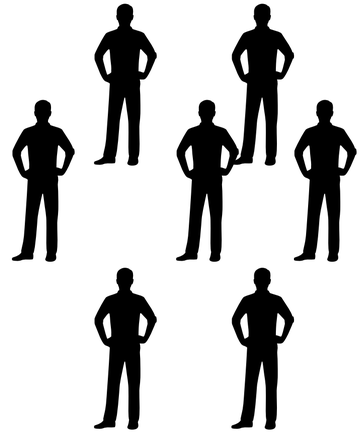
Team



Still Bad



Team

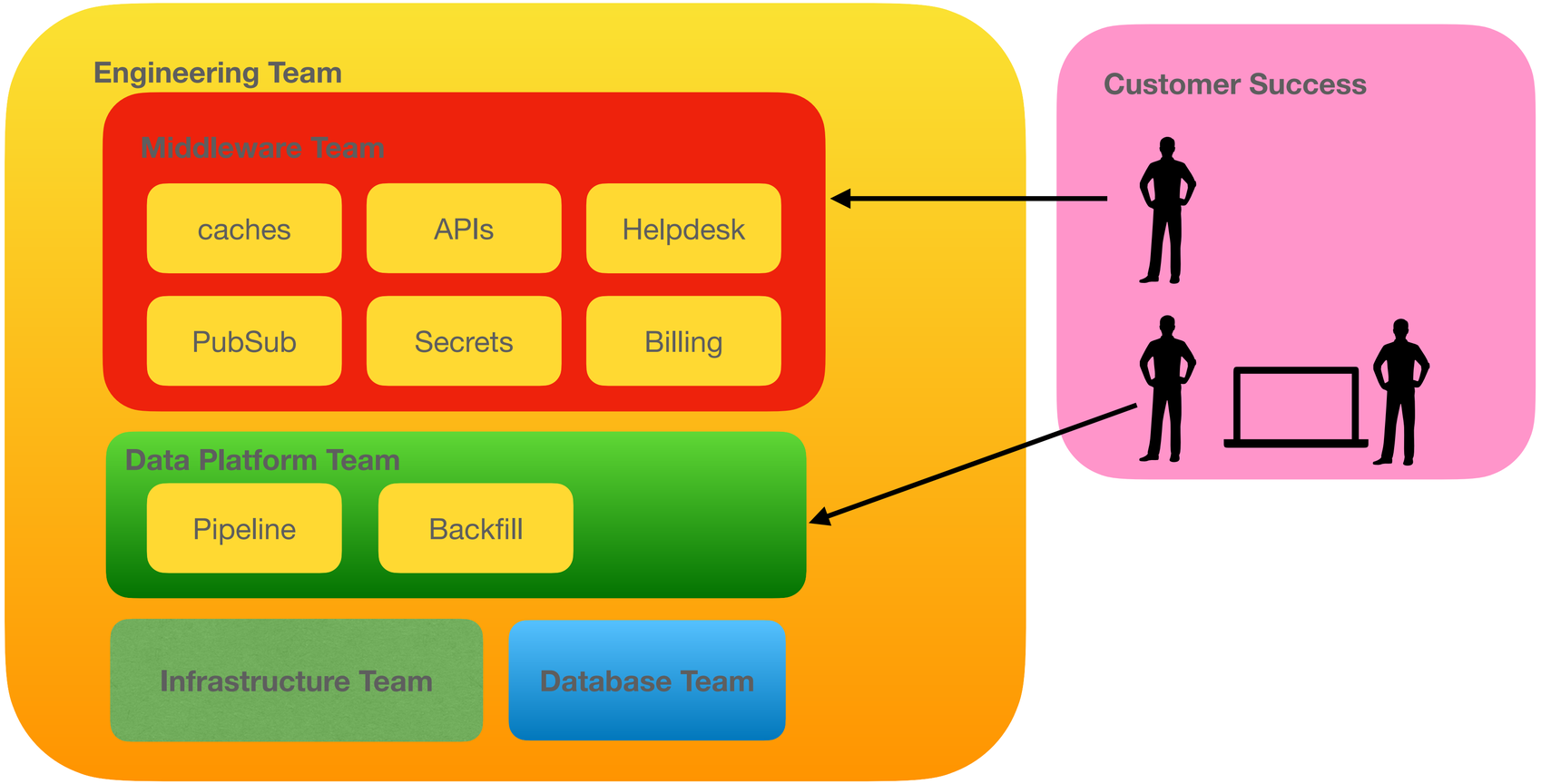


Time

Engineering

“State of Union”

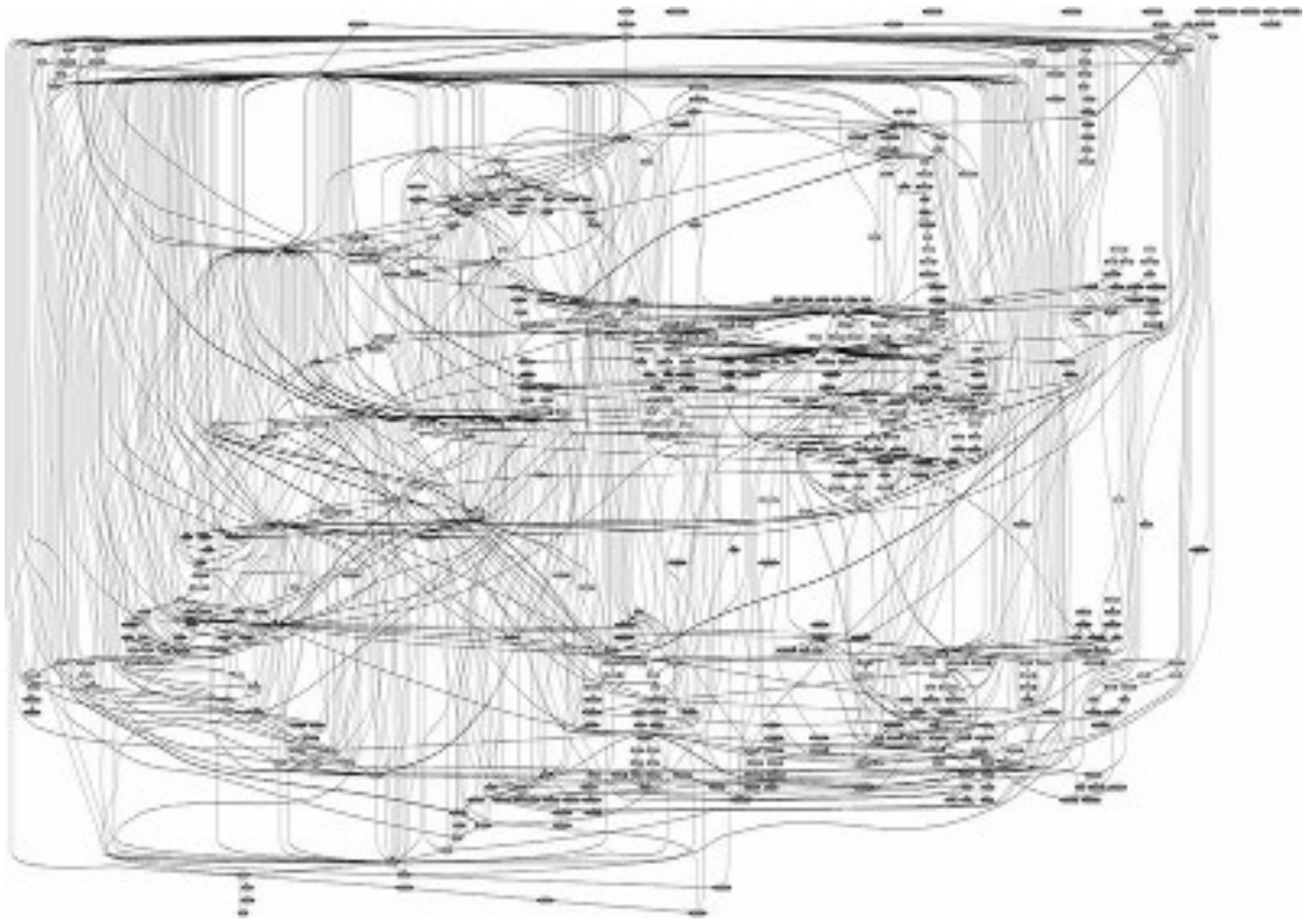
- Conway’s Law
- Cross-teams Interaction
- Engr Productivity
- Technical Debt
- Backlog
- Innovation



Monolithic Architecture

It’s not always a bad thing until …

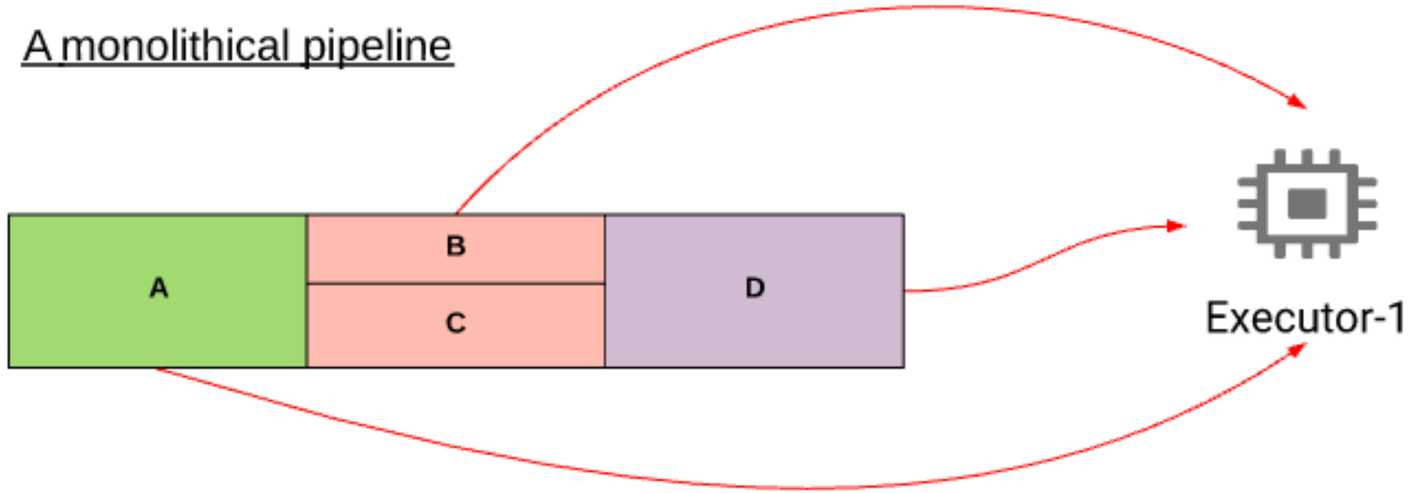
- Tight coupling
- Low Cohesion
- Deployment is akin to stop-the-world
- Unpredictable Runtime
- Unexplained Phenomena

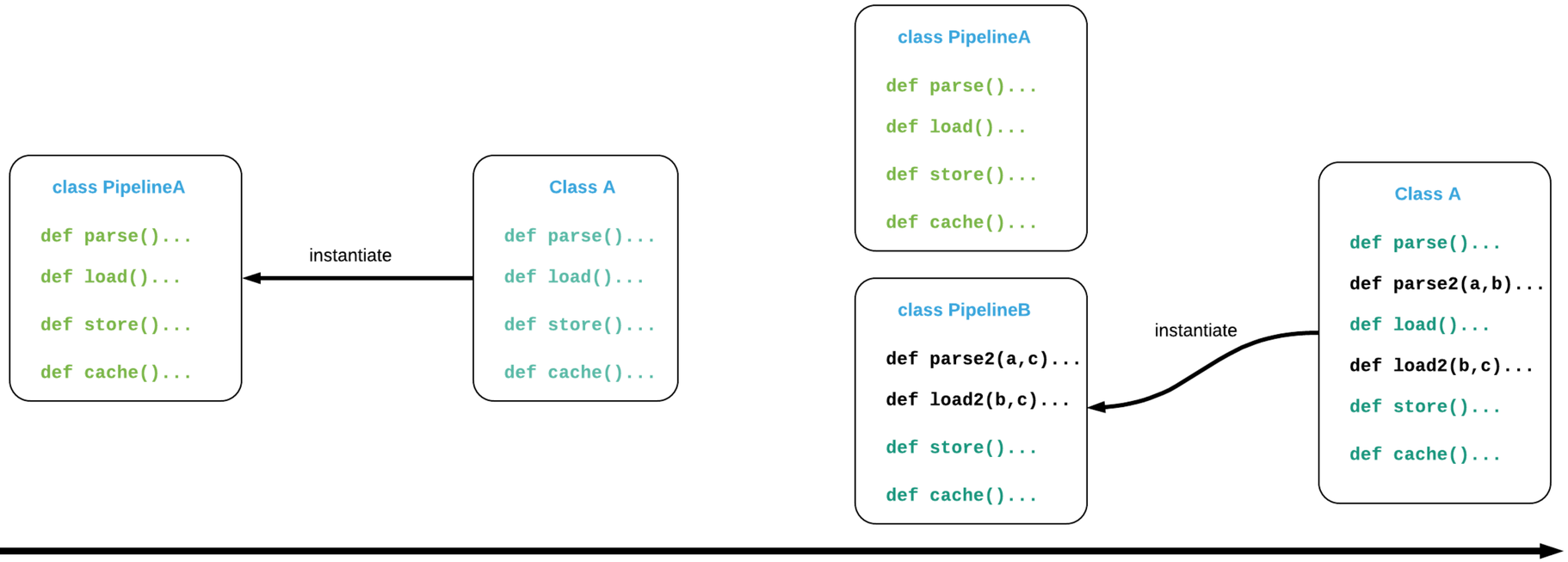


Monolithic Architecture (Abridged)

Glaring Mistakes

- Homogeneity “golden hammer”
- Async vs Sync
- High coupling
- God Object

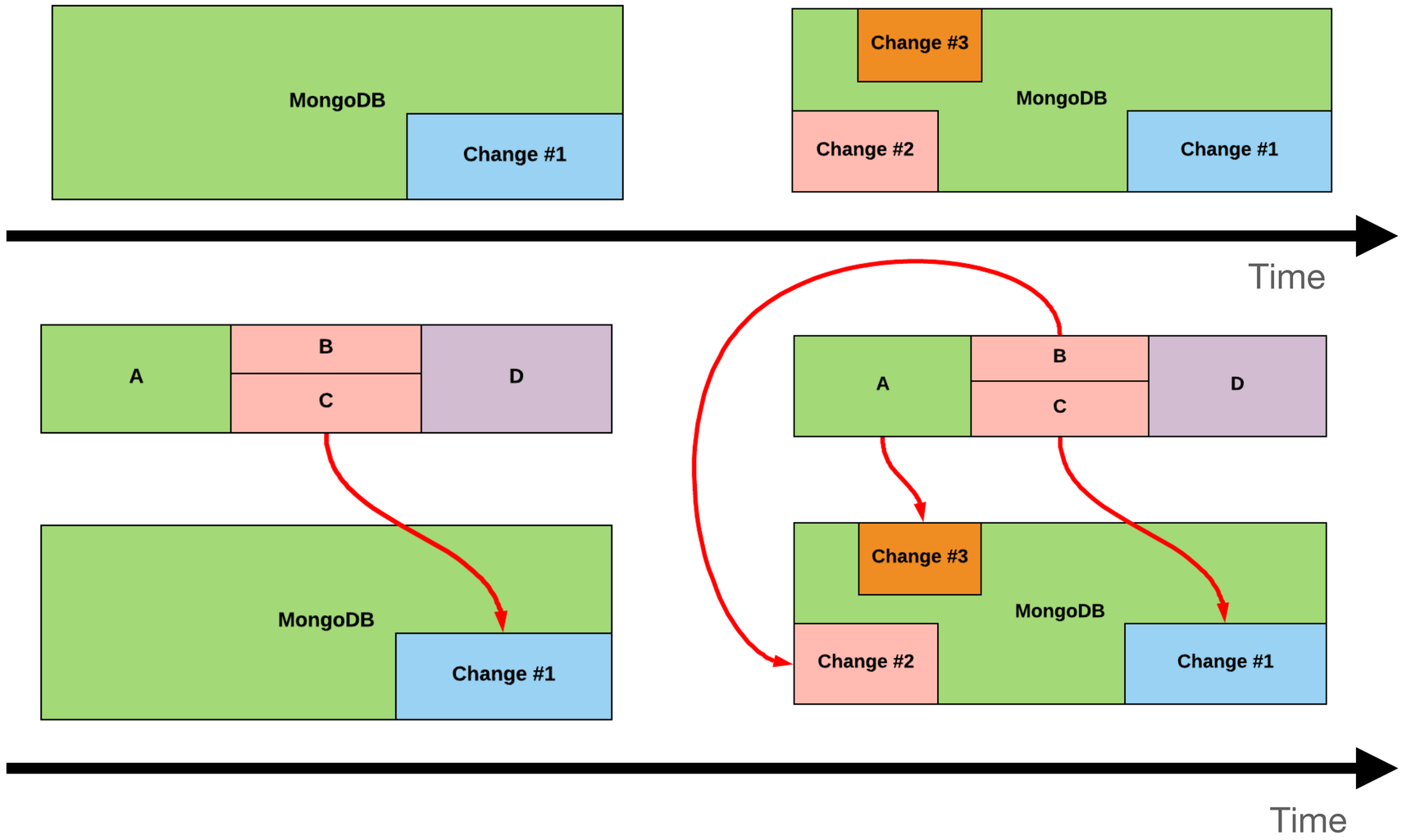




Monolithic Architecture (Abridged)

Glaring Mistakes

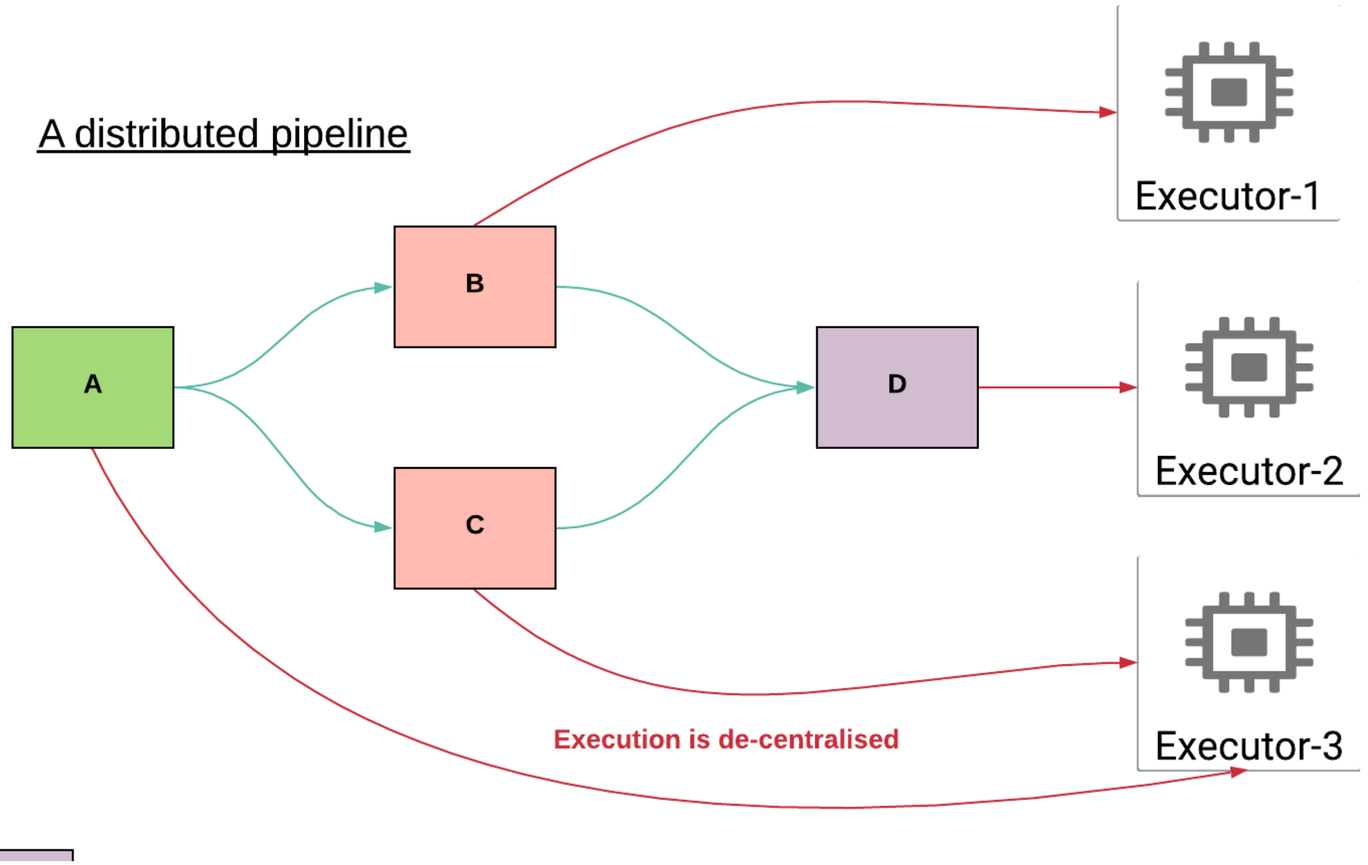
- Lack of engineering patterns, best-practices
- “Techno Push” vs “Value Driven”
- Non-portable code; increased coupling



The Approach …

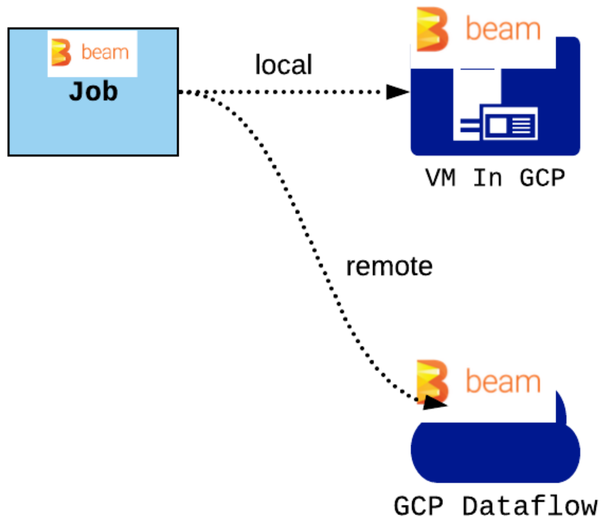
Desirable Architecture (Not Perfect)

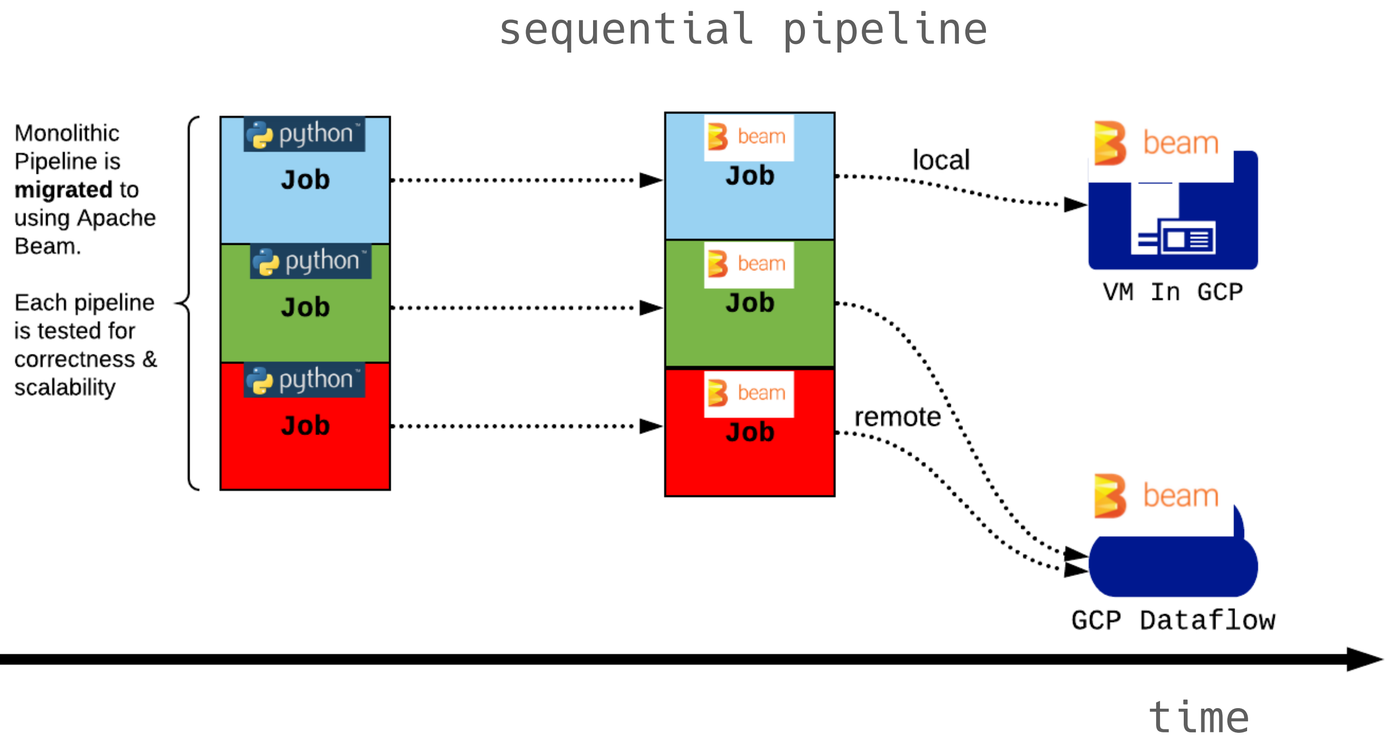
- Multi-Cloud
- Multi-Tenancy
- Lower TCO
- Heterogenous
- Distributed Compute
- Scaling To Diverse Workloads
- Open Protocols



MVP

- A small team (4 pax) was assembled
- “The Lean Startup” - Eric Ries
- Storming
- Migration

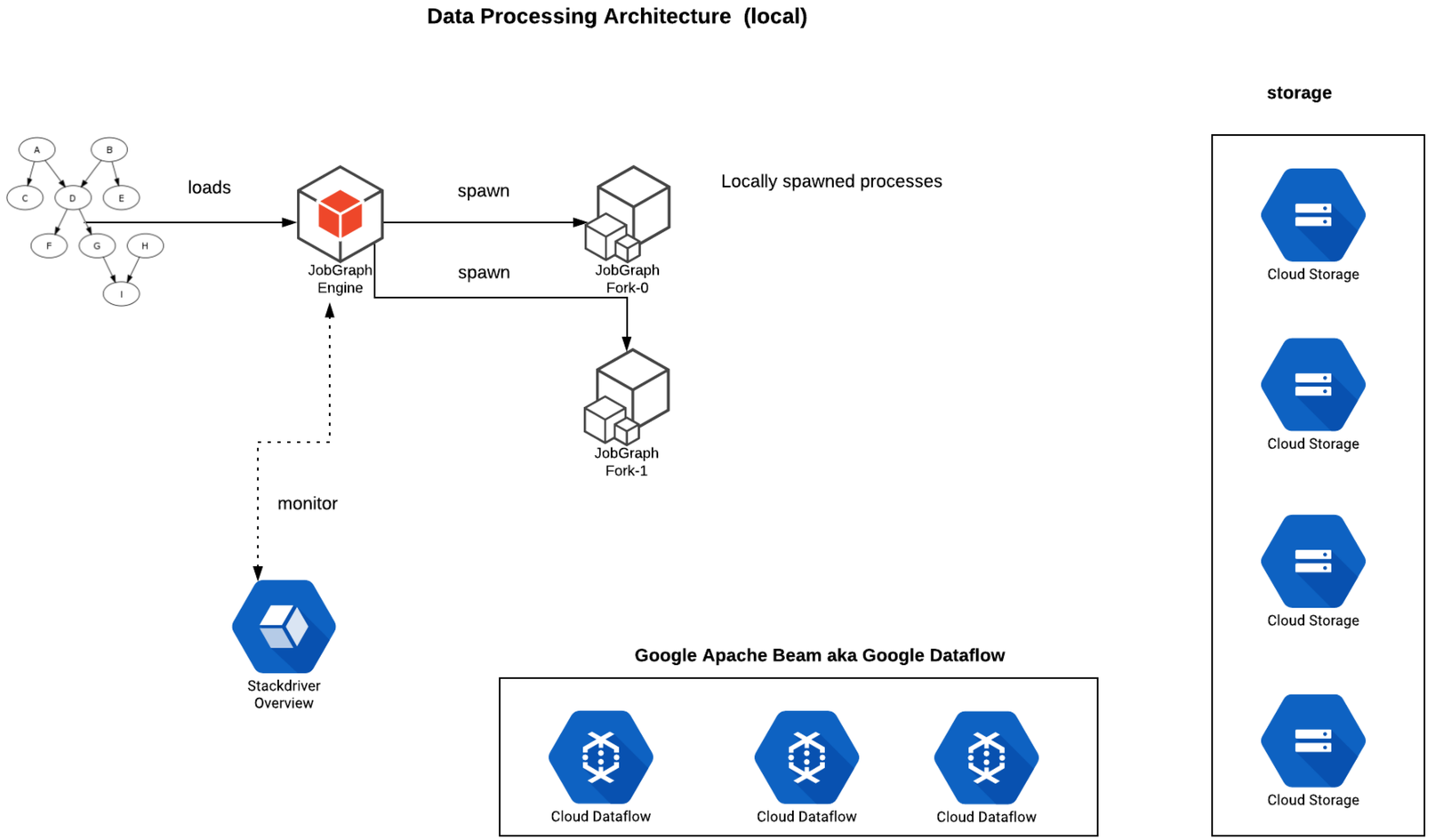




MVP

GCP Architecture

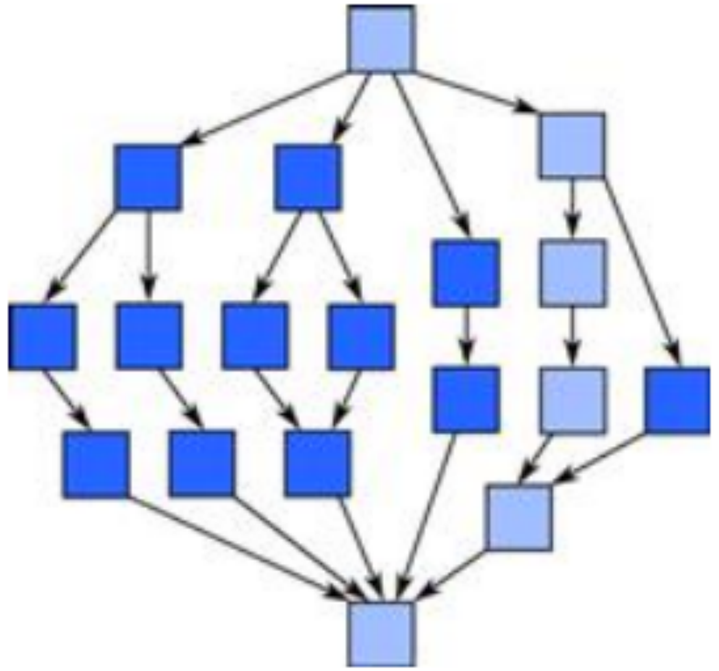
- A small team (4 pax) was assembled
- “The Lean Startup” - Eric Ries
- Storming
- Migration



Modelling Job Topology

Multi-Graph Algorithms

- Arrows captures dependencies
- DAG exposes parallelism
  - **work-span** model/analysis is preferred over Amdahl's Law (graph's speedup is 18/6 = 3)
- Martin Erwig's FGL is captured in the paper "Inductive Graphs and Functional Graph Algorithms"
  - FGL adapted by Verizon's Quiver (JVM) library



We believe that this leaves a very negative impression of the usability of functional languages in general.

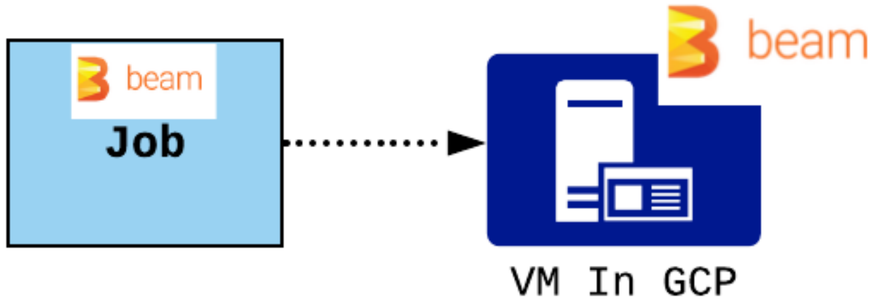
Why do so few functional data structure textbooks exist? We believe that one reason is that the treatment of graphs in functional languages has been rather weak so far.

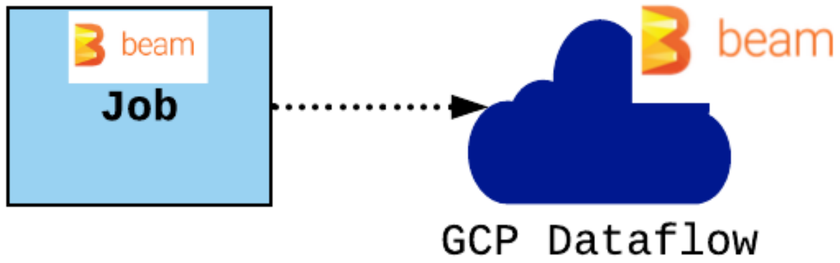
Hence, the goal of this paper is twofold. First, we want to demonstrate that it is possible to define graph algorithms in a distinctive functional style and that these algorithms are at the same time often competitive in terms of efficiency with typical imperative implementations. Second, by giving a collection of algorithms typically found in courses on algorithms and data structures, we try to close a gap in functional algorithms and data structure textbooks. In Section 2 we review related

Modelling Job Topology

Apache Beam - refresher

- Each Beam-job has ***states*** which allows external parties to query and manage the lifecycle
- Each Beam-job can execute ***locally*** or ***remotely***.
- Beam jobs can be written in Python or Java*
- Beam cannot express heterogenous pipelines

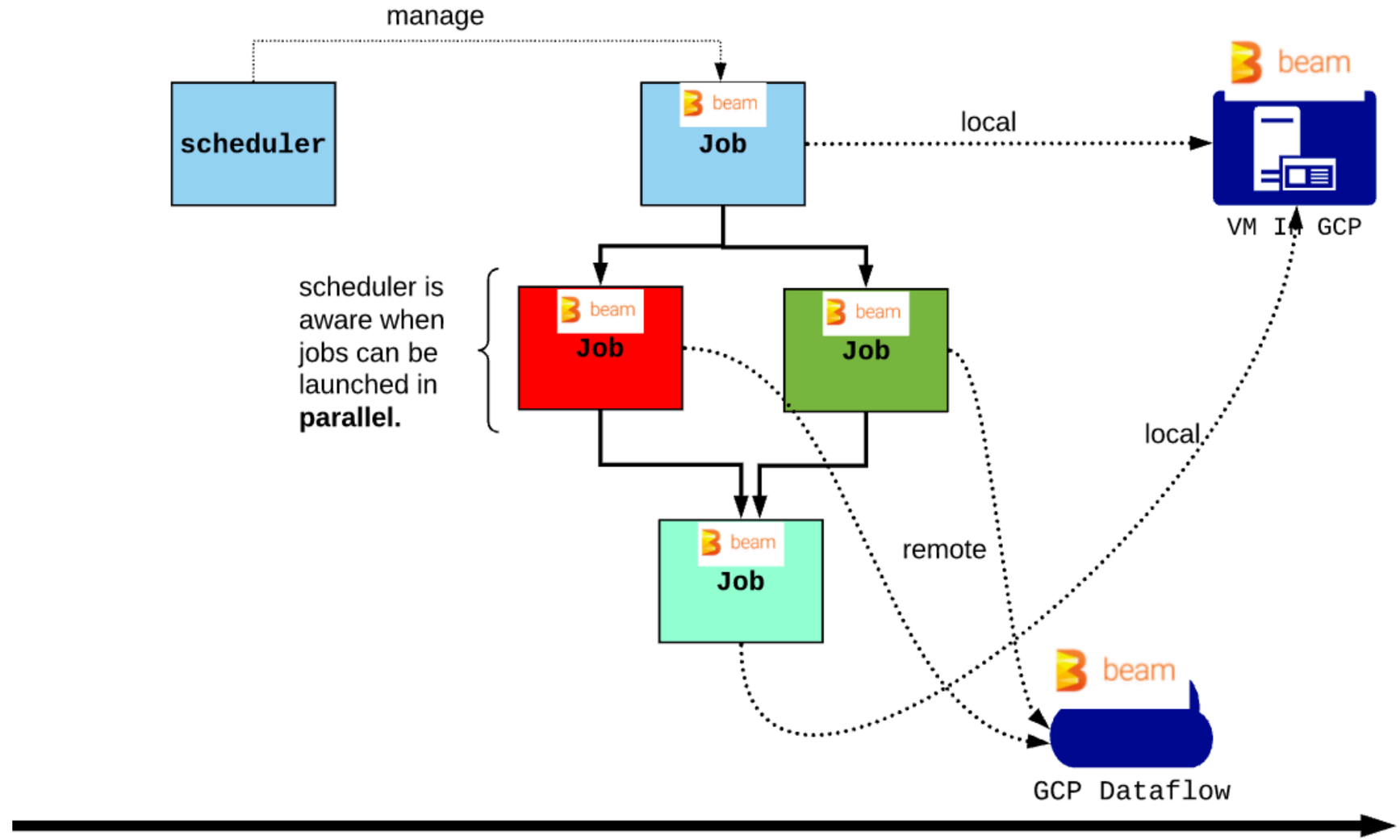




Modelling Job Topology

Scheduler Engine

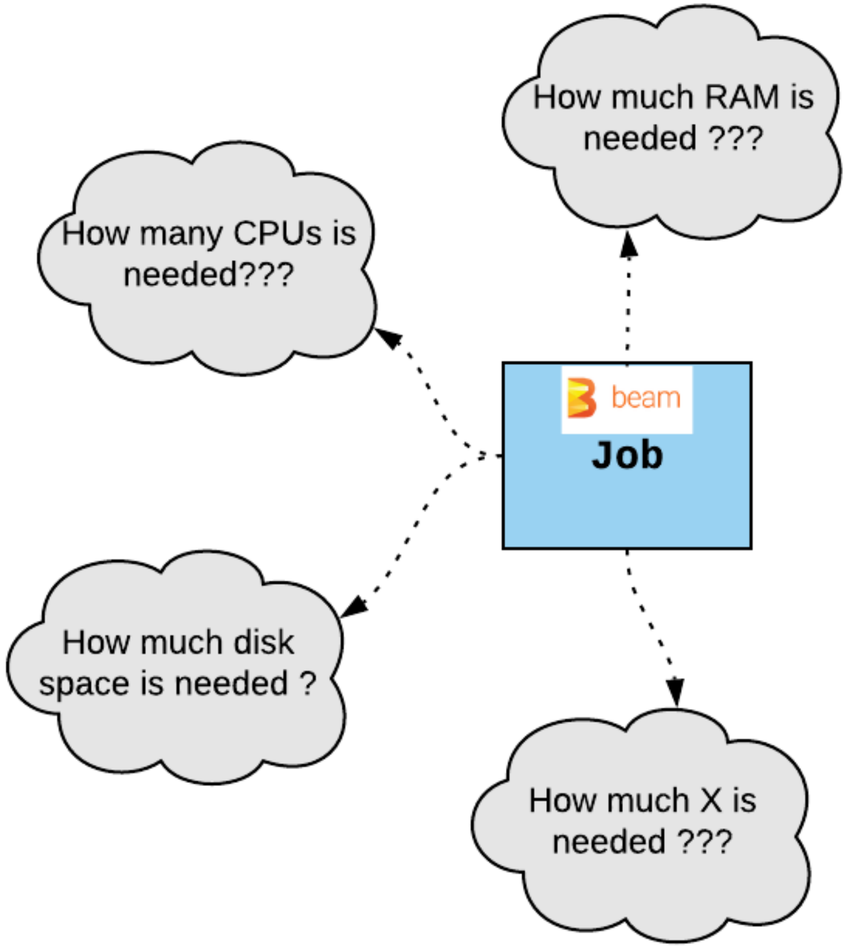
- Each Job has a status which mirrors Apache Beam's status codes
- Each Job's status is monitored by the Engine and the engine will decide to trigger parallel jobs when applicable
- Engine triggers next step depending on (a) when the previous job has completed; (b) previous job timed out or unknown*



Modelling Job Topology

Job Dependencies Modelling

- A job needs CPU, 100% of the time
- A job also needs GPU for heavy-lifting ML work
- Each job's ingress/egress dependencies are in cloud storage e.g. postgresql, blob storage etc
- Each job's dependencies is modelled by *`params`* in the ETL *descriptor*
- Each job can be written in various programming languages i.e. heterogenous



Modelling Job Topology

Job Dependencies Modelling

- A *descriptor* describes all that a job needs e.g. its unique identifier, its working director, the input it expects, the output it writes to etc
- Descriptor is validated by the Scheduler Engine
- Descriptor is programming language agnostic

```
jobs : [
  {
    id : 0
    name : "Read from Data File."
    description : "blah blah blah reading a file."
    workdir : ${?HOME}"/working_dir"
    workdir : ${?JOB_WORK_DIR}
    sessionid : "TEST_ID_0"
    sessionid : ${?SESSION_ID}
    timeout : 5
    restart.max = 2

    # python -m apache_beam.examples.wordcount
    #        --jobName bloody-google-dataflow
    #        --input gs://hicoden/README.md
    #        --output gs://hicoden-test-1/XX
    #        --runner DataflowRunner
    #        --project hicoden
    #        --temp_location gs://hicoden-test-1/

    runner {
      module : "apache_beam.examples.wordcount"
      runner : "Dataflow:python"
      cliargs : ["--jobName raymond-jobgraph-test-0",
                 "--input gs://hicoden/README.md",
                 "--output gs://hicoden-test-1/XX",
                 "--runner DataflowRunner",
                 "--project hicoden",
                 "--temp_location gs://hicoden-test-1/temp/"
                ]
      cliargs : ${?CLI_ARGS}
    }

  },
```

Modelling a Job Topology

Scheduler Engine

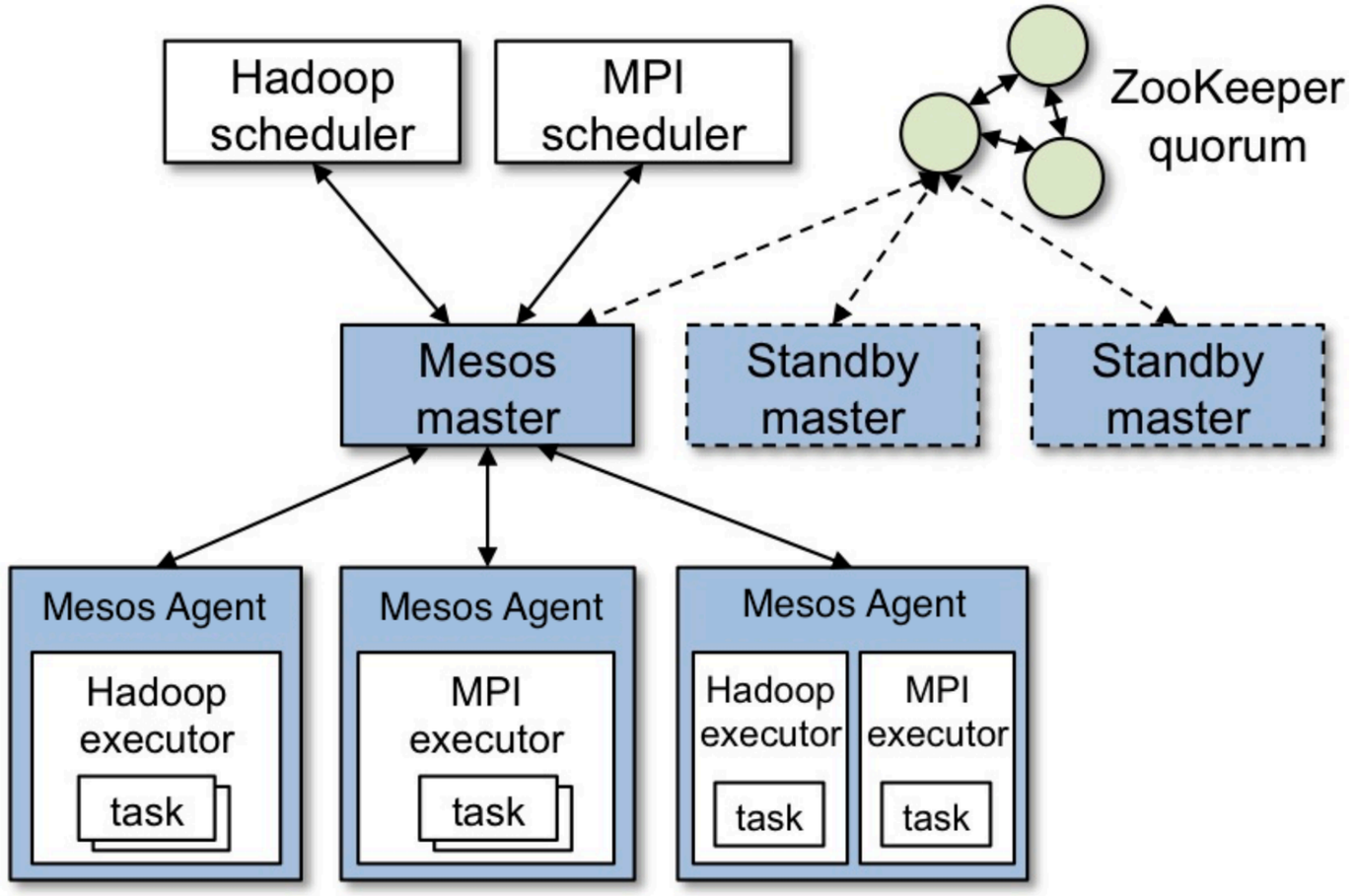
- Each job can be *chained* to create a Pipeline
- The Engine validates
- The Engine manages the cycle of each Pipeline
- Engine interprets the jobgraph and launches jobs in parallel, when possible.

```
workflows : [

  {
    id : 0
    name : "A sample process"
    description : "Reads from file, processed them and finally read them"
    steps : [1,2,3,4]
    jobgraph : [
      "1 -> 2",
      "1 -> 3",
      "2 -> 4"
    ]
  },

  {
    id : 1
    name : "A DFP workflow"
    description : "Extracts the standard report, extract the creatives and line items & sanitize data for final CSV."
    steps : [4,5,6]
    jobgraph : [
      "4 -> 5",
      "5 -> 6"
    ]
  },

  {
    id : 2
    name : "A partial DFP workflow"
    description : "Extract the creatives and line items & sanitize data for final CSV."
    steps : [5,6]
    jobgraph : [
      "5 -> 6"
    ]
  }
```

Common Resource Sharing in Data Center

Apache Mesos

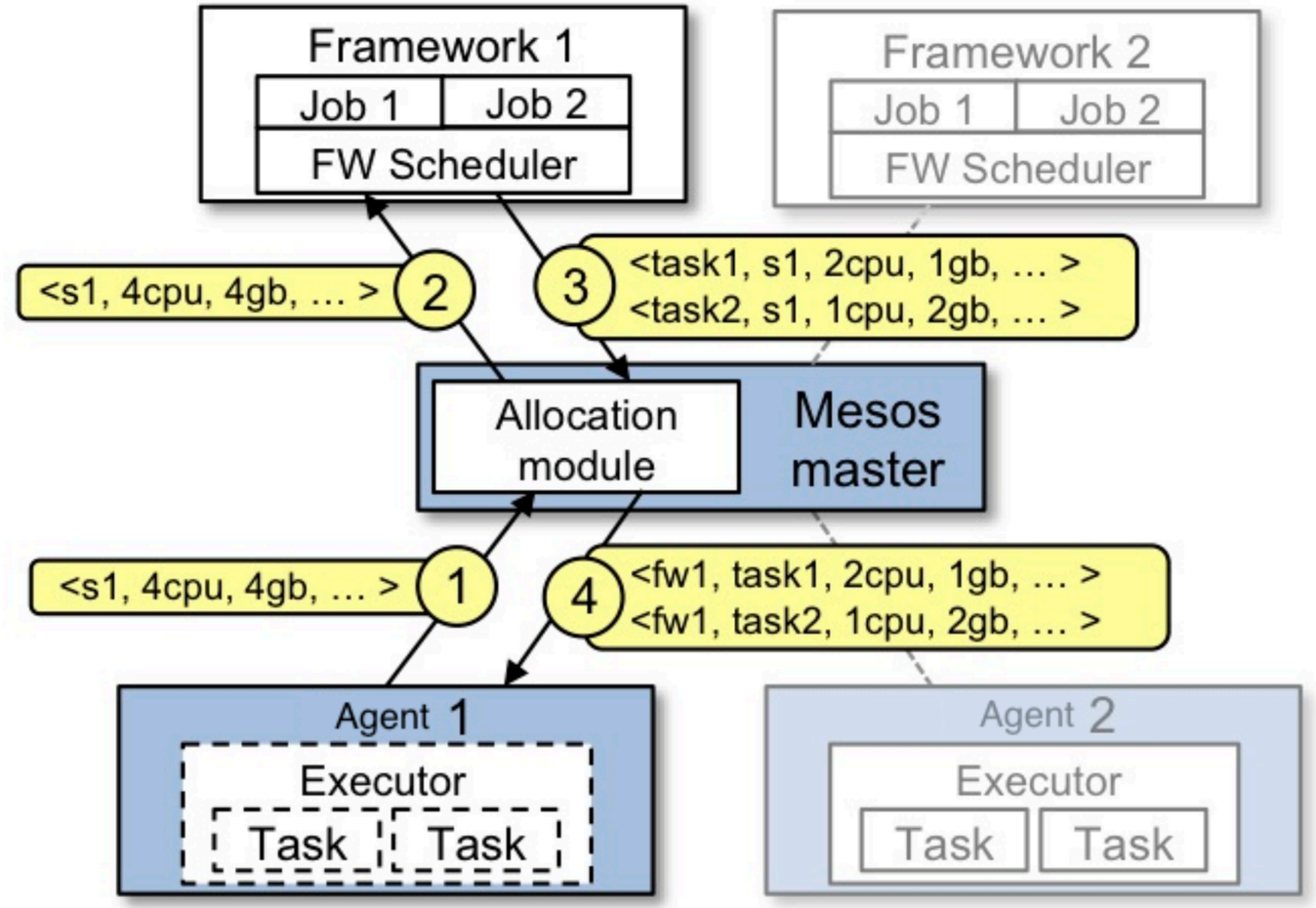
- A *multitenant cloud* architecture describes a single cloud instance and infrastructure purpose-built to support multiple customers
- Resource-aware data centers Apache Mesos
- Mesos Agents are deployed in the data center environment and maintain comms with the Mesos Masters



Common Resource Sharing in Data Center

Apache Mesos Resource Offering

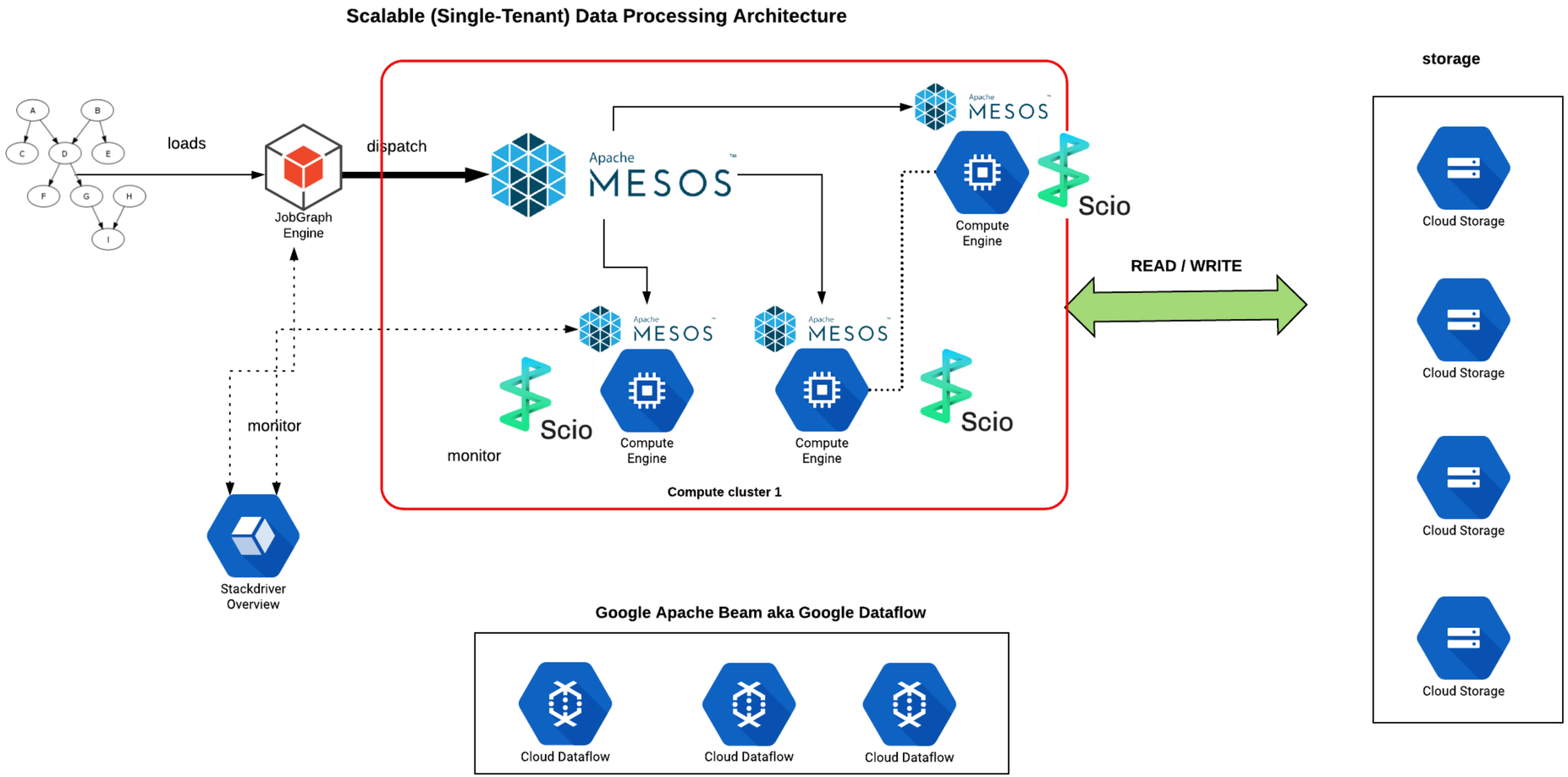
- Whenever a task requires resources from the data center, Mesos would figure out how to assign those resources
- Built-in resource allocators like CPU, DRAM, GPU, HDD
- Mesos allows YOU to develop inject new **resource** with your own **resource allocator**



Modelling a Job Topology

Job Topology Architecture in GCP

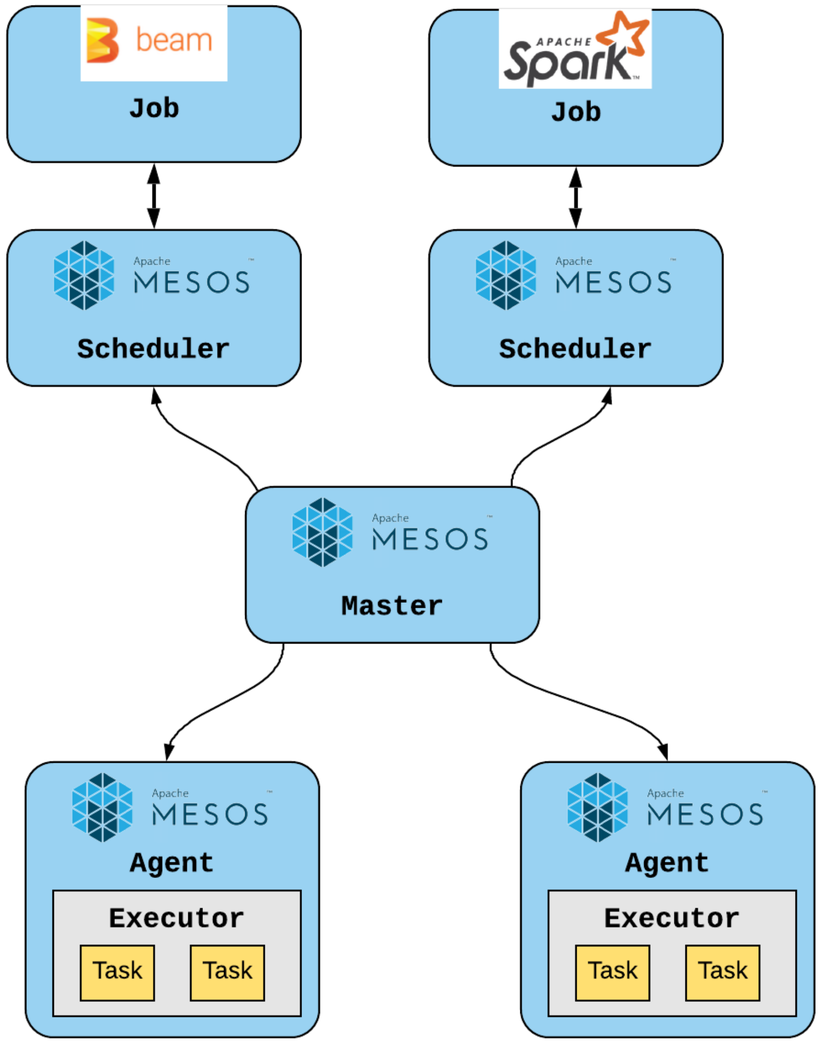
- Single-tenant compute cluster for running our pipelines
- All cloud resources are in GCP



Data Architecture

Distributed Compute Cluster

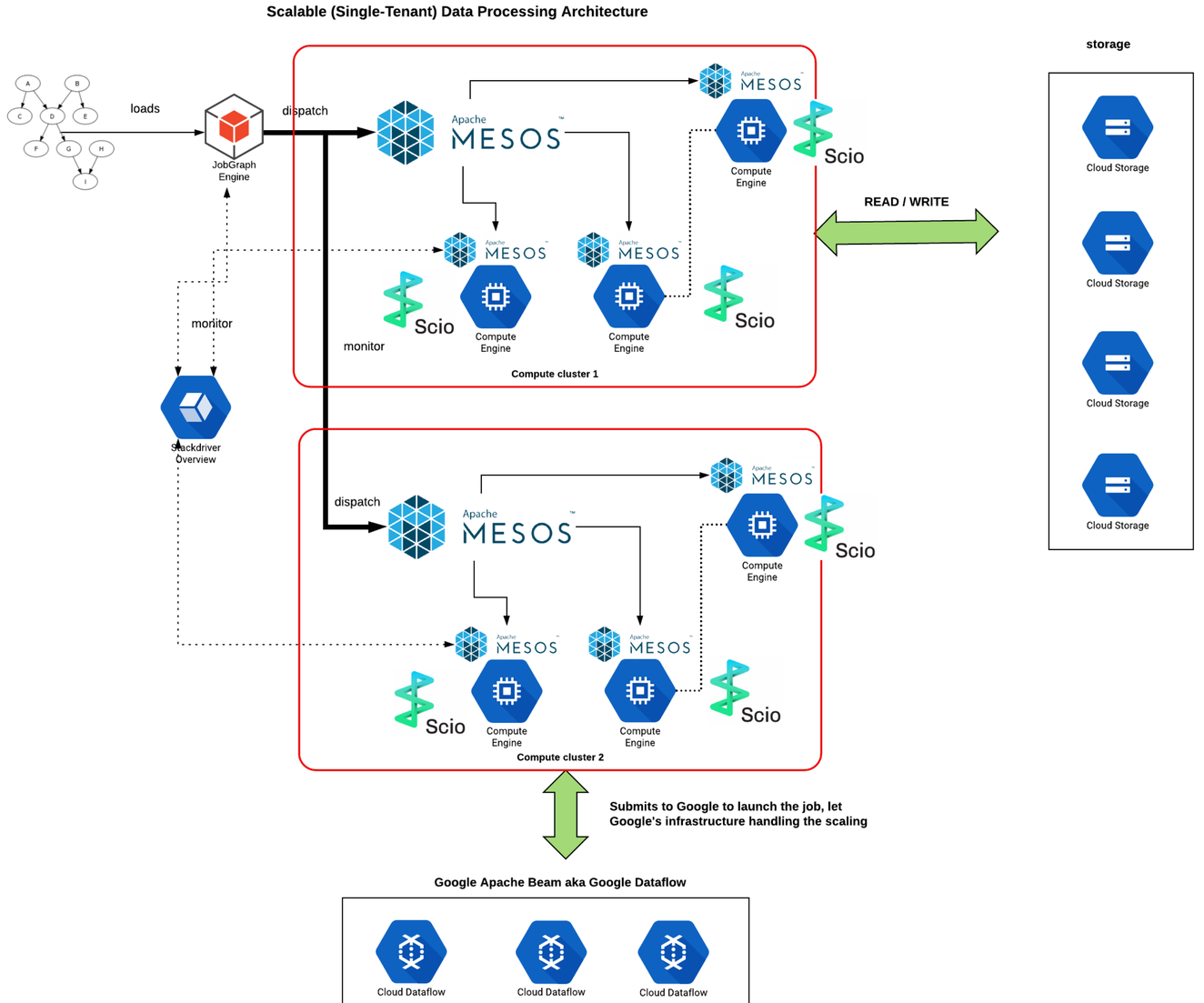
- Data Team migrated from “old” ETL to the new ETL
- Data Team saw increased demands to run Apache Spark jobs - great news!
- Data Team is able to spin new Mesos Compute Clusters to run experiments



Data Architecture

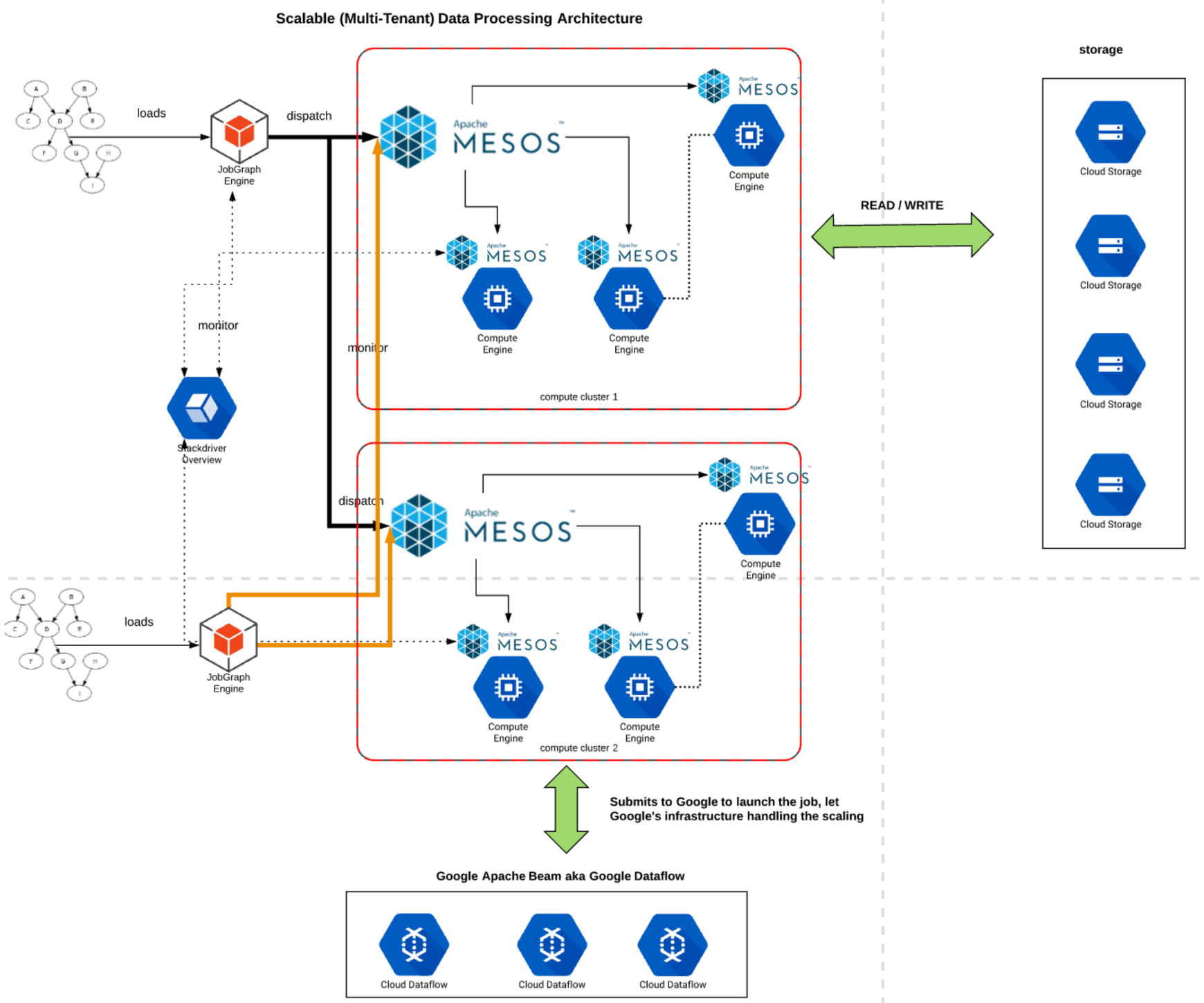
Multiple Compute Clusters

- Enhancement that allows *production* to co-exist with *non-production*



Data Architecture

Multi-Tenant



References

- Dataflow / Apache Beam - Eugene Kirpichov
- The Dataflow Model - Tyler Akidau, Sam Whittle et al
- MillWheel : Fault-Tolerant Stream Processing at Internet Scale - Tyler Akidau, Sam Whittle et al
- FlumeJava : Easy, Efficient Parallel Data Pipelines - Craig Chambers, Nathan Weizenbaum et al
- Mesos - Matei Zahari et al
- Why Curiosity Matters - Harvard Business Review September 2018
- Spotify Scio - Spotify's Scala API around Apache Beam
- Typelevel Cats Lightweight, modular, and extensible library for functional programming in Scala
- Verizon Quiver : A reasonable library for modelling multi-graphs in Scala
- Scala - The Scala Programming Language

References

- Apache Beam VLDB paper - Tyler Akidau et al @ Google
- Streaming 101
- Streaming 102
- Beam vs Spark
- Hierarchical scheduling in diverse data center workloads : Battacharya, Ali Ghodsi, Ion Stoica et al
- Beam Comparison : Data Artisans
- Dataflow/Beam & Spark : A programming model comparison : Tyler et al @ Google
- Dataflow Pipeline Execution Parameters