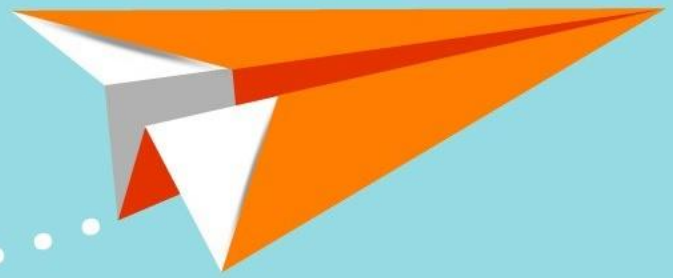
3rd October 2022



Elastic Heterogeneous Cluster and Heterogeneity-Aware Job Configuration

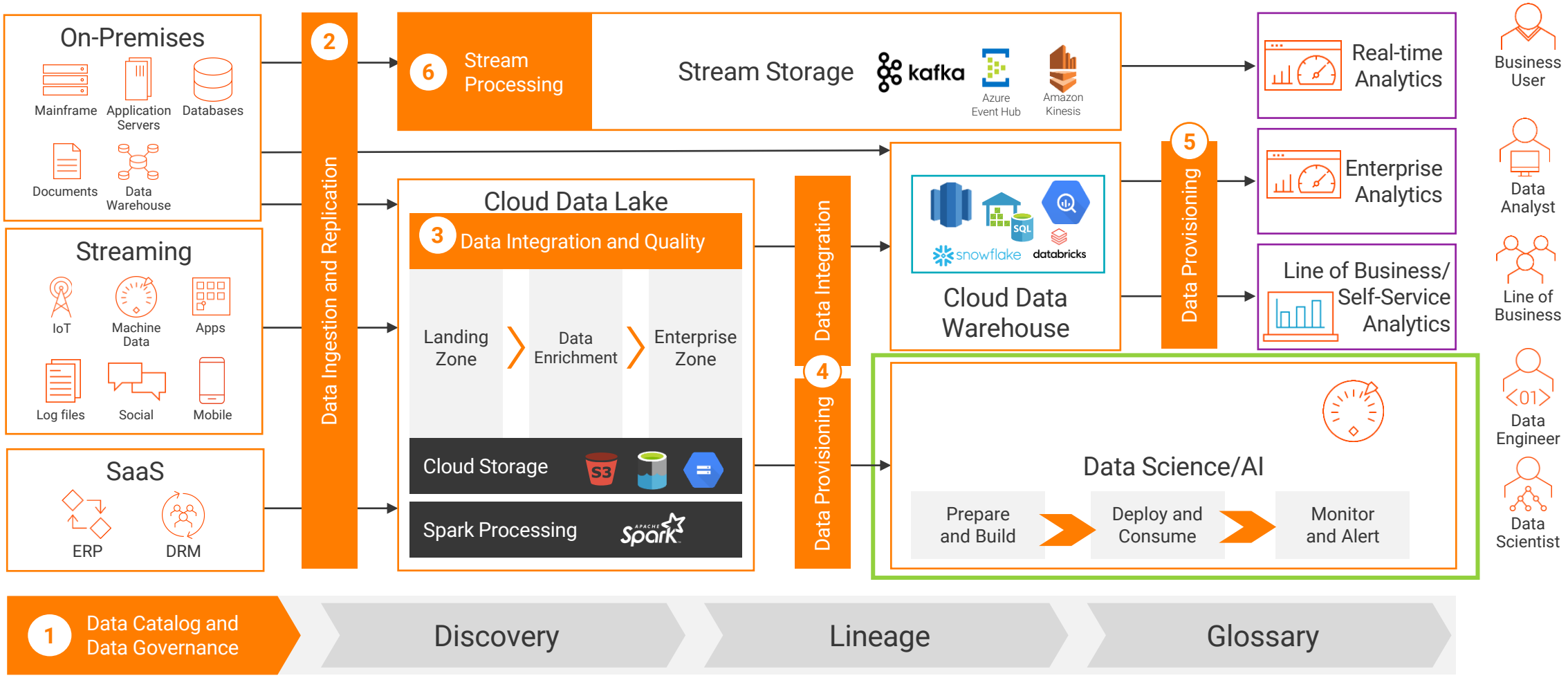
- Yongqin Xiao
- Atam Prakash Agrawal

Informatica

Agenda

- Context - Cloud Data Integration Elastic (CDI-E)
- Motivation for heterogeneous elastic computing
- Proposed solution
  - Auto-detection of preferred Instance type
  - Auto healing on instance type selection
  - Effective spark container scheduling & cluster autoscaling
- Impact

Informatica - Intelligent Cloud Data Management

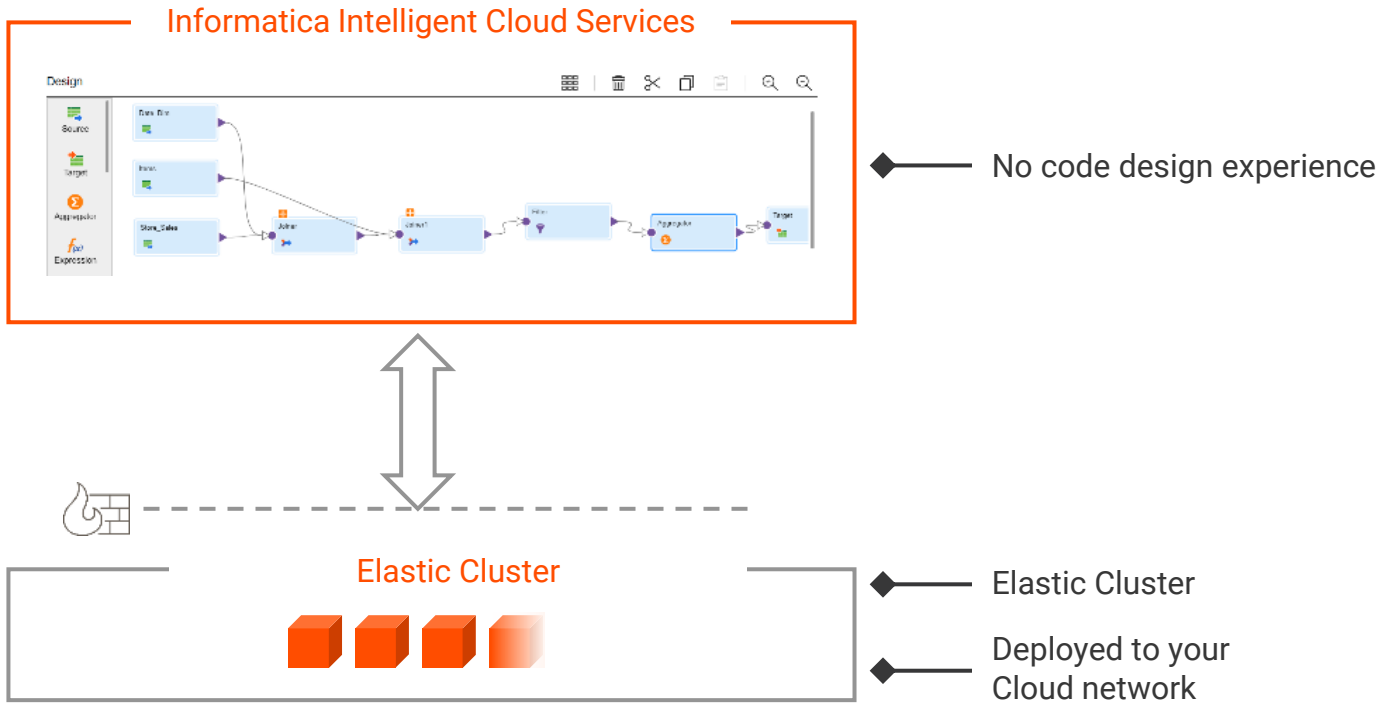


Cloud Data Integration Elastic – CDI-E

Build and maintain high performance data pipelines with Elastic infrastructure to match data processing needs.

CDI-Elastic Features

- Elastic Cloud infrastructure to process any data volumes and concurrency within your cloud subscription
- Support for modern use cases through structured, semi-structured data support, hierarchy processing, MLOps
- Developer productivity and Operational simplicity through dynamic mappings, Incremental File loader, Auto tuning and Auto Scaling.
- Up to 90% savings on Cloud Infrastructure using infa auto scaler and spot instances
- Up to 70% job perf improvement using Auto-Tuning
- Up to 5 times performance boost and 72% TCO savings running CDI-Elastic on NVIDIA GPU



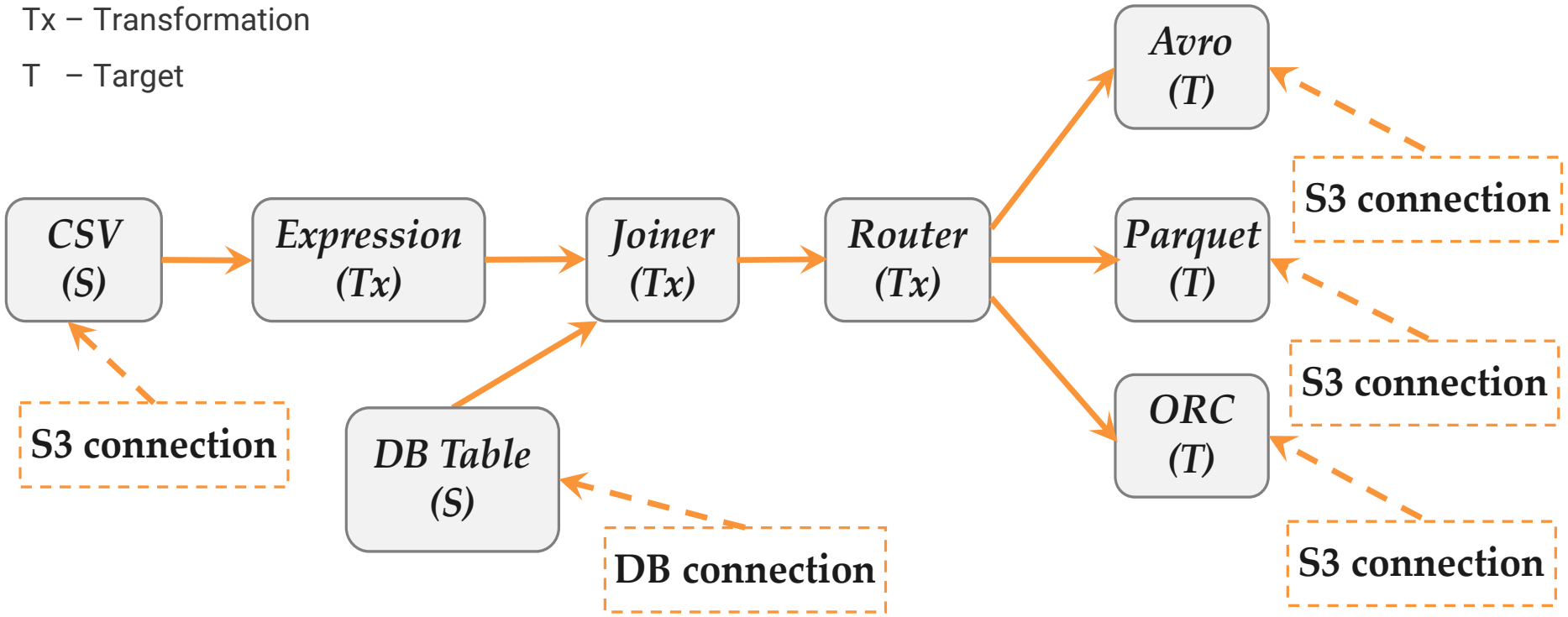
When to use CDI-Elastic

- Support for spot instances, arm64, GPU's
- Elastic infrastructure to match your data processing needs
- Support for all CDW/DL use cases
- Support for MLOps, Python Integration
- Be more productive with auto-scaling, auto-tuning

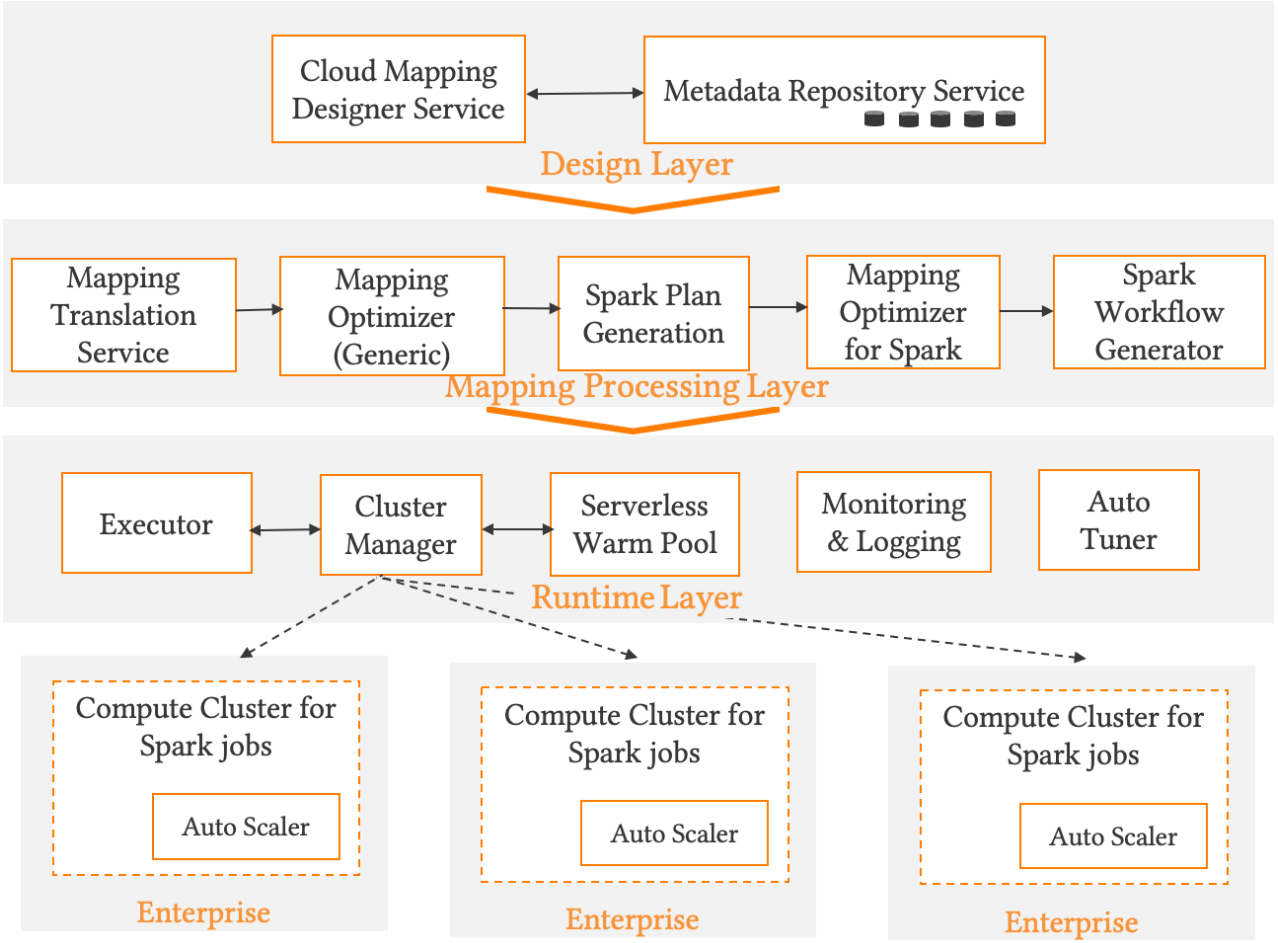
CDI-E Data Integration Job - A SQL like ETL job

Legend

- S – Source
- Tx – Transformation
- T – Target



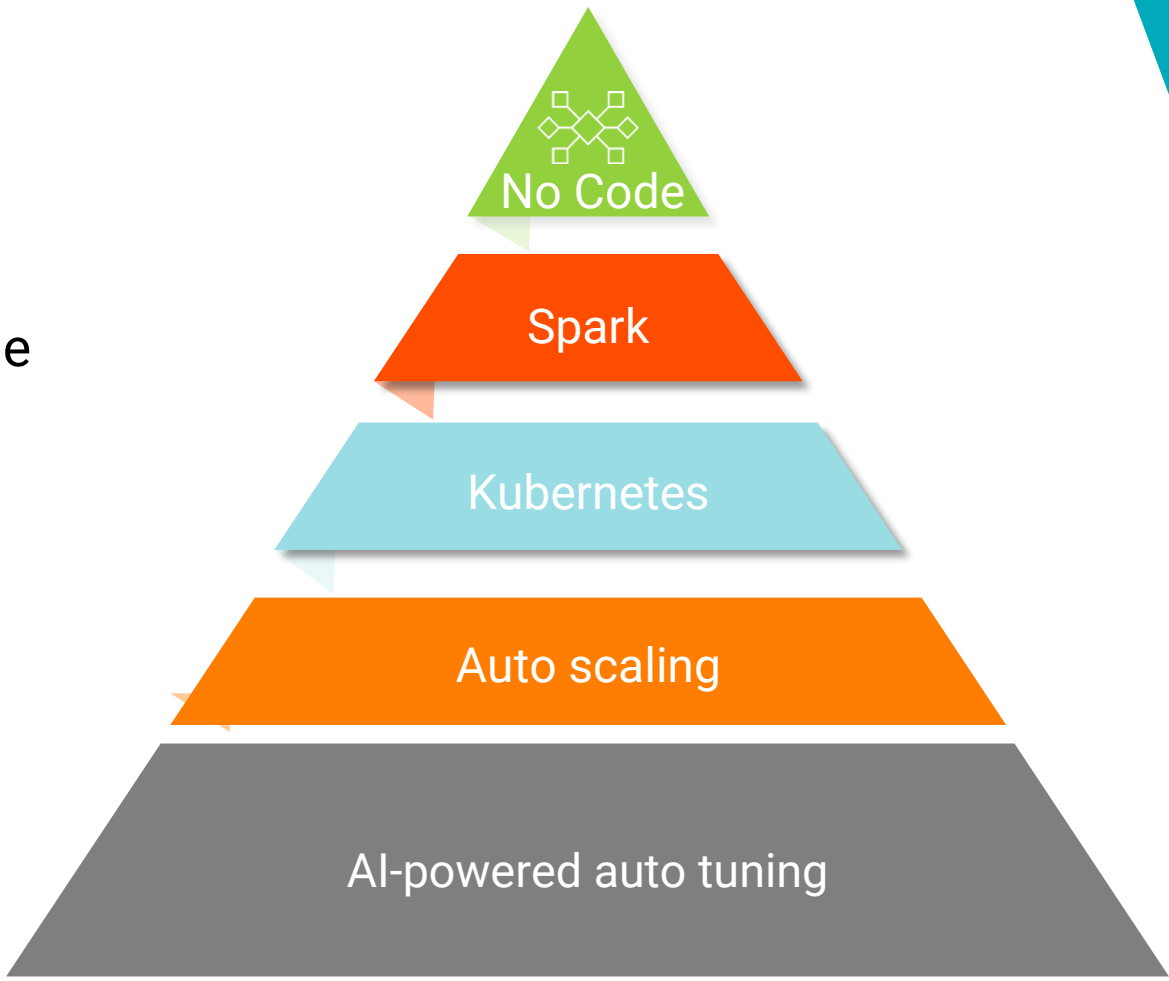
CDI-E Architecture



Informatica

CDI-E Elastic Compute Cluster

- A fully managed ephemeral Kubernetes Cluster.
- Shared by different users to run different types of jobs.
- Auto-scales up or down based on workload, within the resource boundaries user specifies.
- Executes spark job using native spark functionalities and a rich set of Informatica plugins.
- Spark job runs in cluster mode. Each is isolated by default.
- CLAIRE® - Informatica's AI engine, optimizes and auto-tunes the execution parameter for a spark job.



Motivation for Heterogeneous cluster

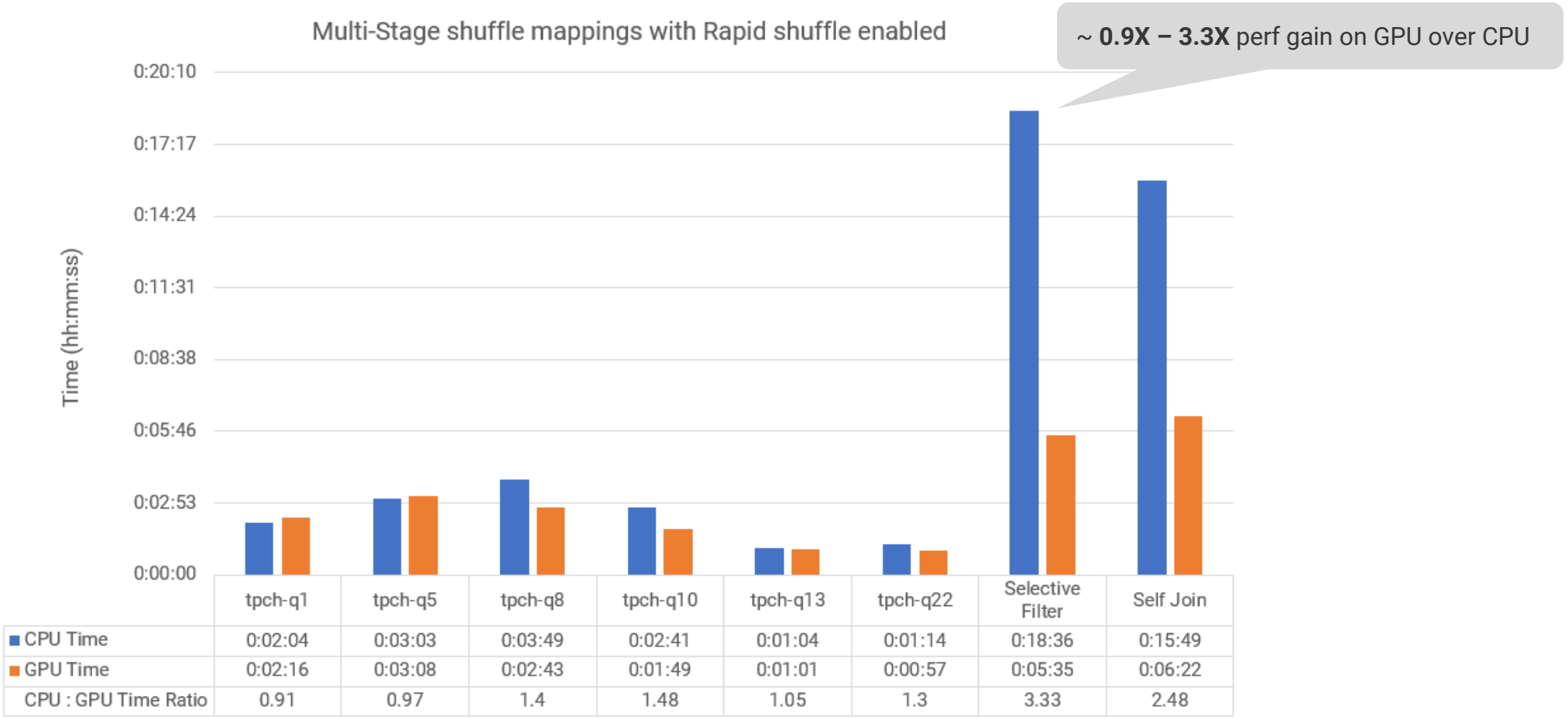
- The compute cluster serves a large variety of spark jobs with different resource requirements.
- Cloud provider offers a wide range of node instance types, with different resources and costs.
- Each instance type fits a different kind of job in terms of performance and cost.
- Traditional elastic cluster supports homogeneous worker nodes.
- No one-size-fits-all solution for choosing the instance type for a cluster.
- Users must configure different clusters and carefully assign jobs to the best fitting cluster.

CPU Node vs. GPU Node

- RAPIDS spark accelerator requires NVIDIA GPU
- General 1:4 vCPU to memory ratio
- Price: 1 GPU ~ 4 vCPU
- Job performance vs. cost
  - Cost - m5d : g4dn = 1 : 1.67
  - Job Time (CPU:GPU) -> overall cost
    - = 1.67 : 1 –> CPU cost same as GPU
    - > 1.67 : 1 –> save cost with GPU
    - < 1.67 : 1 –> save cost with CPU

Instance Type	GPU	vCPU	MEM(G)	Network (Gbps)	Price ($)
g4dn.2xlarge	1	8	32	25	**0.752**
g4dn.12xlarge	4	48	192	50	3.912
m5d.2xlarge	0	8	32	Up to 10	**0.452**
m5d.12xlarge	0	48	192	10	2.712

Job Performance on CPU Cluster vs. GPU Cluster



	tpch-q1	tpch-q5	tpch-q8	tpch-q10	tpch-q13	tpch-q22	Selective Filter	Self Join
CPU Time	0:02:04	0:03:03	0:03:49	0:02:41	0:01:04	0:01:14	0:18:36	0:15:49
GPU Time	0:02:16	0:03:08	0:02:43	0:01:49	0:01:01	0:00:57	0:05:35	0:06:22
CPU : GPU Time Ratio	0.91	0.97	1.4	1.48	1.05	1.3	3.33	2.48

Limitation of GPU Cluster

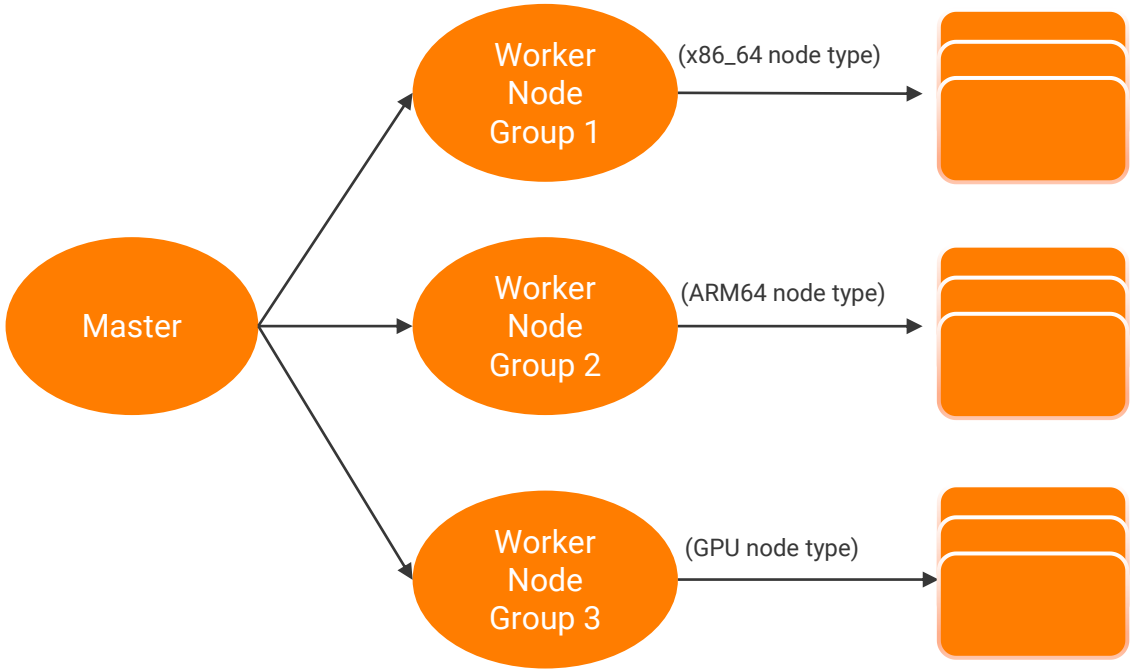
- **Resource on GPU node:** GPU node usually has GPU: CPU ratio from 1:8 to 1:64.
- All jobs are configured to request a GPU by default.
- **Spark Executor Sizing:** Only 1 executor can be created on a 1-GPU node because a single GPU can not be shared by multiple executors.
- **GPU-supported Spark job** - These jobs will not utilize enough CPU Resources. CPU Resources will be wasted.
- **Non-GPU supported Spark job -** GPU resources will be wasted.
- Requires user manual tuning at the job level to reduce such waste.

Proposed Solution – Heterogeneous Cluster and Heterogeneity-Aware Job Configuration

- A heterogeneous cluster environment, with mixed instance types for worker nodes, such as CPU node and GPU node.
- Auto decides the best suitable instance type for a job based on its computation and resource characteristics.
- Auto improves the above decision based on historical job execution statistics, for the next run.
- Effective k8s pod scheduling of the job to achieve better resource utilization within the cluster.
- Auto scales nodes of each type based on demands

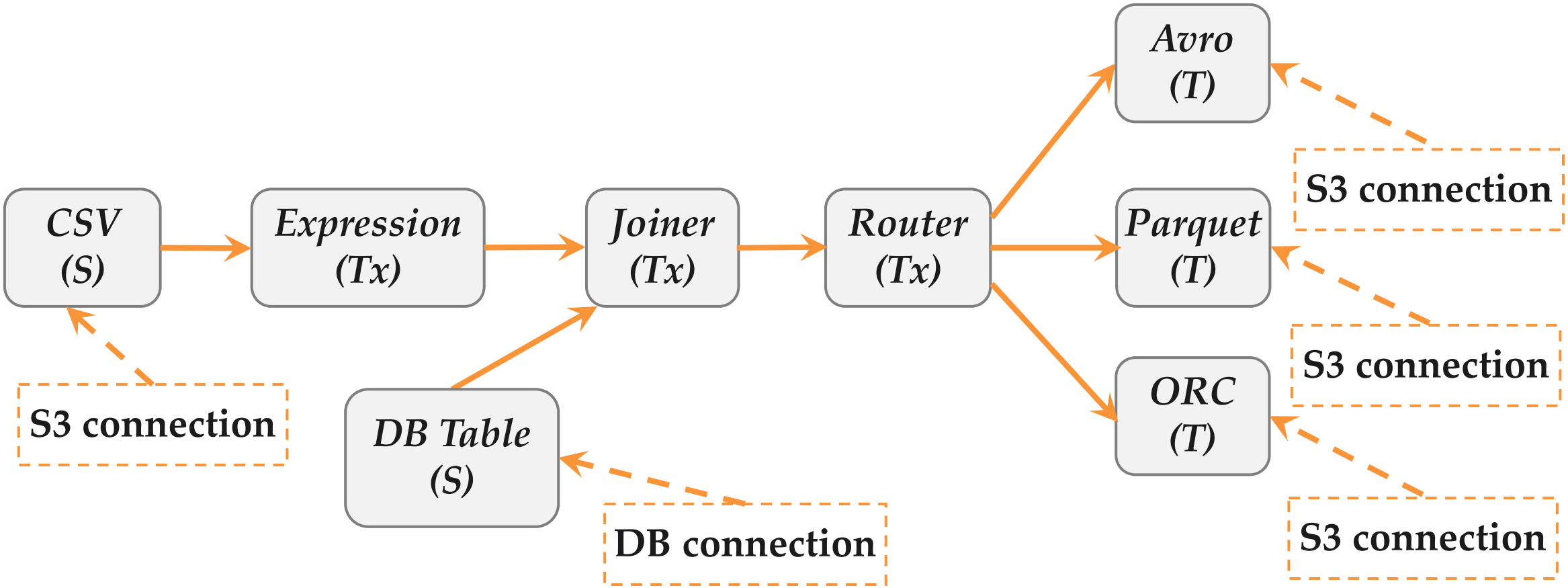
Heterogeneous Cluster

Sample cluster



- Mix of CPU, ARM-based CPU, and GPU nodes.
- GPU job runs on GPU node.
- CPU job favors either CPU node or ARM-based CPU node & Unused cores of GPU node.
- Nodes of the same type grouped together.
- Each node groups scale separately on demands.

Computation Units in Data Integration Job



Characteristic of Each Computation Units

- **Memory intensive**: parquet data connector, hierarchical data processing transformation, INFA transformations configured with large memory cache, …
- **IO intensive**: NFS-based file connector, transformations with cache overflow to disk, transformations requiring data shuffle, …
- **Network intensive**: transformations requiring data shuffle, data connectors for various cloud storage, …
- **CPU intensive**: expressions with heavy mathematical calculation, hierarchical data parser, Data Quality, and Data Masking transformations, …
- **GPU suitable**: expression, large sorter, file parser, …
- **ARM64 Support**: all except certain native-TPL dependent transformations

Other Characteristics to Consider

- Some connectors and transformations are genuinely slower than others due to **external resource restrictions** (e.g. concurrent DB connections) or their complexity
- **Data cardinality** – GPU is not suitable for small data
- **Computation complexity** of INFA-specific transformations

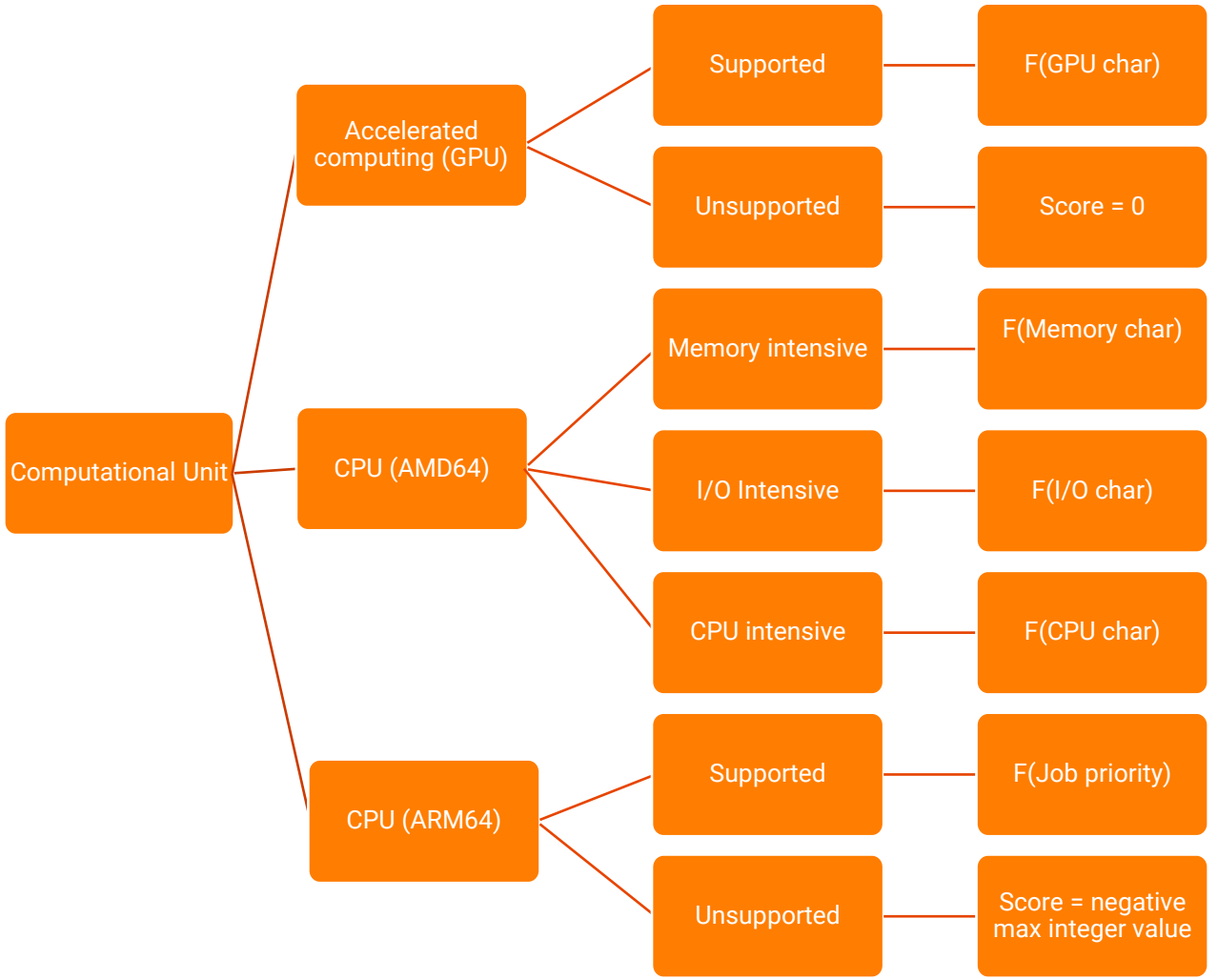
Auto-detection of Preferred Instance Type

- Called as node type preference matrix.
- Happens at Data Integration job compilation time - when translating DI job to Spark job in Scala
- A handler of the computation unit assigns one score to each node type.

Data-pipeline/ Worker node type	Computation Unit1	Computation Unit2	Computation Unit3	... ...	Computation UnitN
X86_64	s11	s12	s13	... ...	s1N
ARM64	s21	s22	s23	... ...	s2N
GPU	s31	s32	s33	... ...	s3N

Static score computation

- If a computation unit can't run on a certain instance type at all, the score will be negative infinity.
- If a computation unit doesn't support accelerated computing resources, the score is 0.
- The higher the score, the more preference for the node type.
- ARM64 node types are used as a cost-saving model for the low-priority job.



Overall Score for the Data Integration Job

- Final score of the data integration job for each node type:

  X86_64 score = (w1*s11 + w2*s12 + w3*s13 +……. + wn*s1N)/N

  ARM64 score = (w1*s21 + w2*s22 + w3*s23 +……. + wn*s2N)/N

  GPU score = (w1*s31 + w2*s32 + w3*s33 +……. + wn*s3N)/N

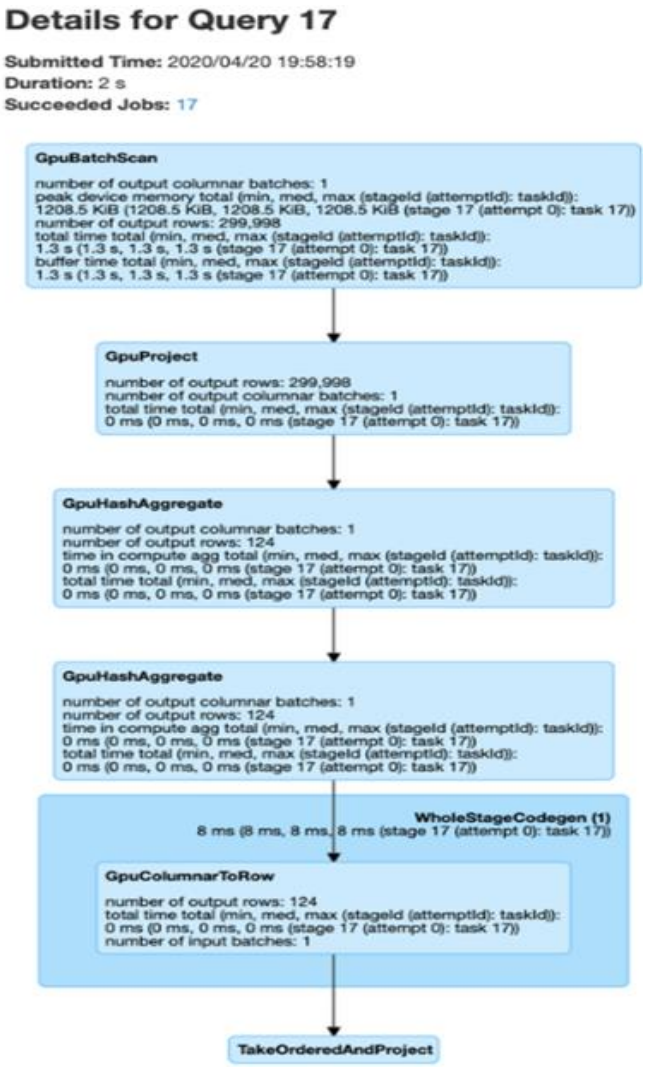
- The final score represents the node selection preference of the job.
- We call such scores static scores.
- A score-based job scheduler will schedule the job's executors to its preferred type of worker nodes as much as possible.

Instance Type Preference Score Assignment Summary

1. A handler of the computation unit assigns one score to each node type.
2. Score is defined based on the resource consumption nature and resource configuration of a computation unit.
3. Other factors like data cardinality and complexity of computation unit also weigh in.
4. AI training could be introduced to compute the weight of each computational unit.

Auto Heal the Static Score

- Spark's physical plan contains jobs, execution stages in each job, and phases in each stage. Each stage is scheduled separately to spark executors.
- Physical plan is available as part of Spark monitoring data.
- Spark monitoring data also contains execution statistics at the job, stage, or even phase level, including execution time, number of partitions, data (bytes and rows) being processed, amount of shuffle data, which hardware being used (e.g. CPU vs. GPU) at each phase, etc.
- Evaluated static score is adjusted based on this monitoring information.
- The adjusted score is called the dynamic score.



Informatica

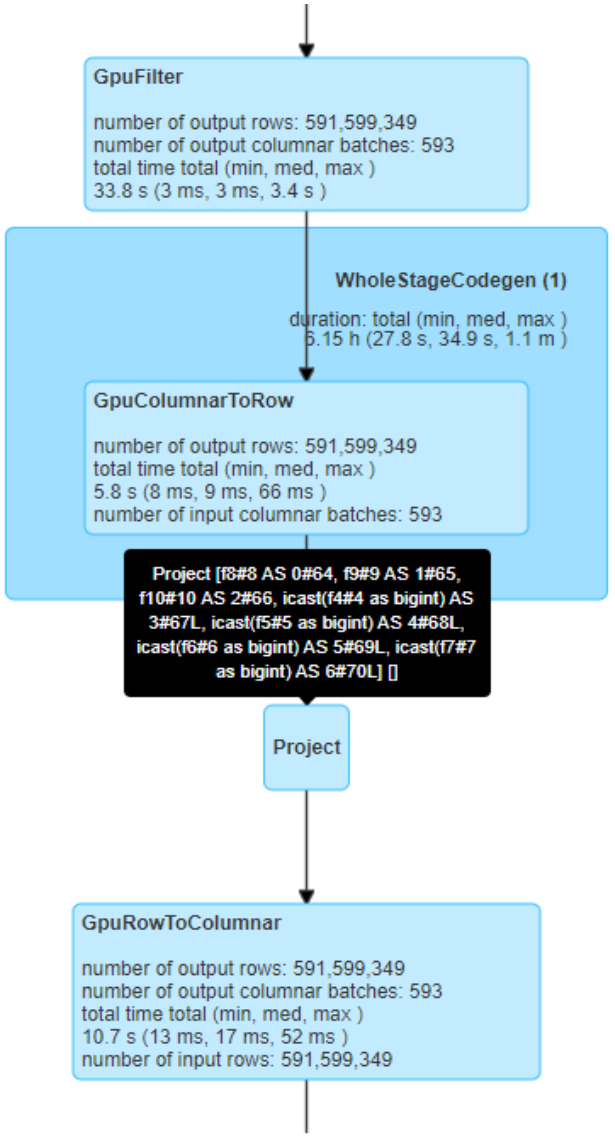
Spark Monitoring and Statistics Report

- Details of executor task metrics can be found at the following links-

  https://spark.apache.org/docs/latest/monitoring.html#executor-metrics
  https://spark.apache.org/docs/latest/monitoring.html#executor-task-metrics

- Spark qualification tool uses these stats to analyze Spark events generated from CPU-based Spark applications to help quantify the expected acceleration of migrating a Spark application or query to GPU.

App Name	App ID	App Duration	SQL DF Duration	GPU Opportunity	Estimated GPU Duration	Estimated GPU Speedup	Estimated GPU Time Saved	Recommendation
appName-01	app-ID-01-01	898429	879422	879422	273911.92	3.27	624517.06	Strongly Recommended
appName-02	app-ID-02-01	9684	1353	1353	8890.09	1.08	793.9	Not Recommended



Informatica

Persist and Leverage Dynamic Score

- Both dynamic and static scores persisted.
- The dynamic score is averaged across runs.
- The latest dynamic score is considered for the next run of the mapping.
- For a new product release, the new static score may overwrite an earlier dynamic score.
- If the data Integration job is modified, persisted scores will be erased.
- Allow users to explicitly choose GPU for better performance even if the overall cost might be higher.

Kubernetes Scheduling

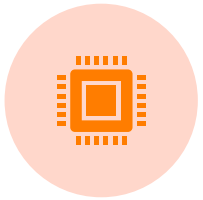
K8s Node Affinity Feature

```
spec:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - preference:
          matchExpressions:
          - key: node.kubernetes.io/instancegroup
            operator: In
            values:
            - GPU
        weight: 20
      - preference:
          matchExpressions:
          - key: node.kubernetes.io/instancegroup
            operator: In
            values:
            - CPU
        weight: 100
  containers:
```

Kubernetes Scheduling

Scheduling Spark Driver/Executor Containers

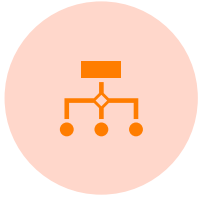
Cluster is started with a minimal set of general-purpose CPU nodes.



Spark driver will prefer general-purpose CPU node.

GPU executors are scheduled to GPU node only

CPU executor has higher node affinity for CPU node than GPU and ARM64 node. Schedule CPU containers to unused CPU core of GPU node.



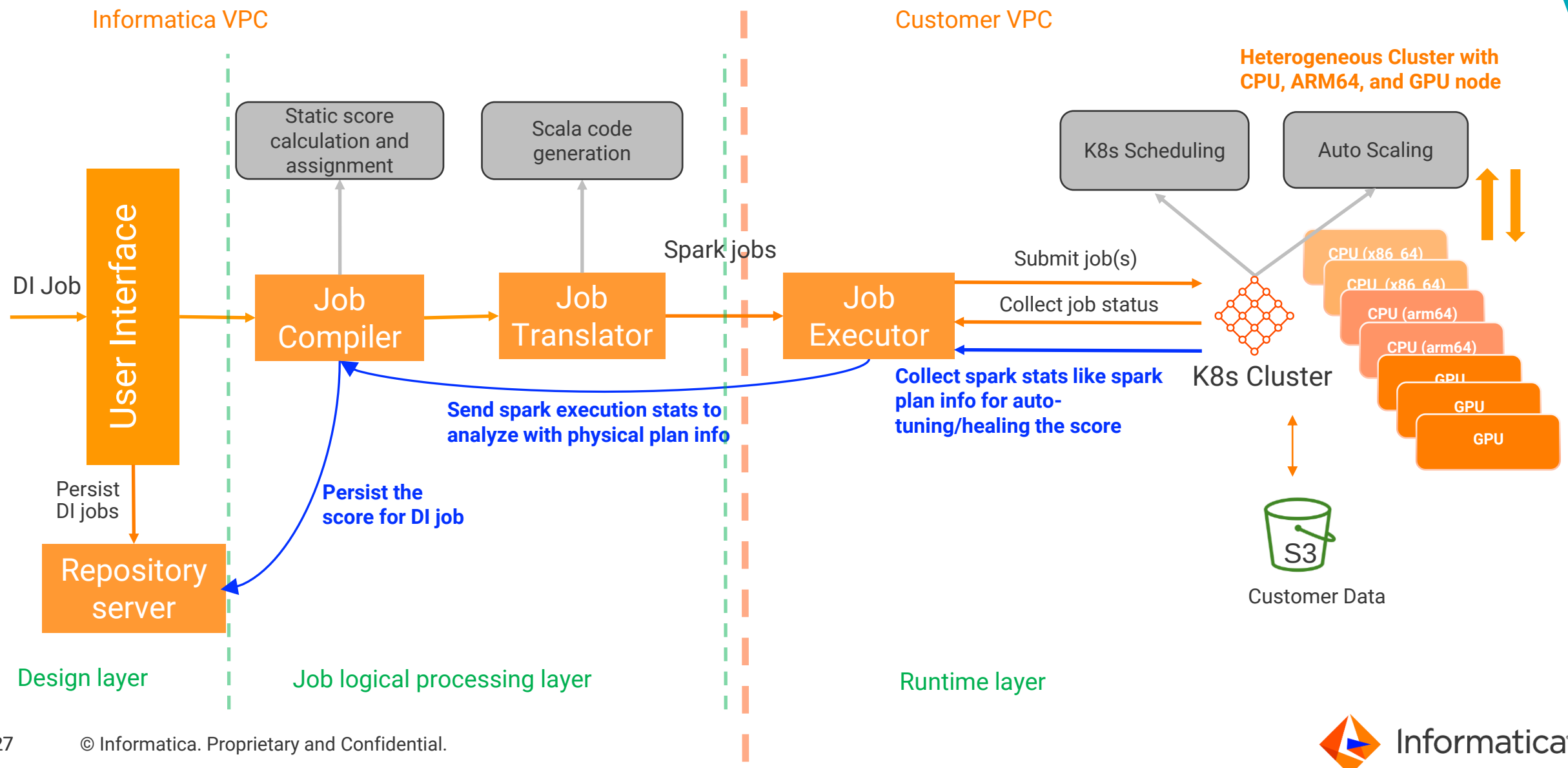
The scheduling could also depend on application priority or user preference.

The low-priority job will run on cheaper instances like an ARM64 node.

Kubernetes Cluster Auto Scaling

- User defines max node count N for the heterogeneous cluster.
- Default quota for different node groups
  - Max GPU node count: 70% of N.
  - Max CPU node count: 50% of N.
  - Max ARM64 node count: 50% of N.
- Node group of each instance type scales separately based on needs.
- Cluster auto scaler controls the overall node count.
- Schedule CPU containers to GPU node only if there are already some GPU containers running on the node – to allow GPU nodes to scale down.

Design Flow



Informatica

Benefits after Solution

Simplicity and intelligence

- Simple cluster type configuration: hetero vs. homo.
- No worry about quota among node groups by default.
- No need to manual configure GPU vs. CPU for job.
- Job properties are auto adjusted based on GPU vs. CPU – partition size, memory, cpu.

Performance and cost

- Different types of jobs can share one cluster.
- Up to 5x performance improvement for GPU suitable job.
- Up to 72% Significant cost saving.

Q & A

Informatica

Thank You