

When Table meets AI: Build Flink AI Ecosystem on Table API

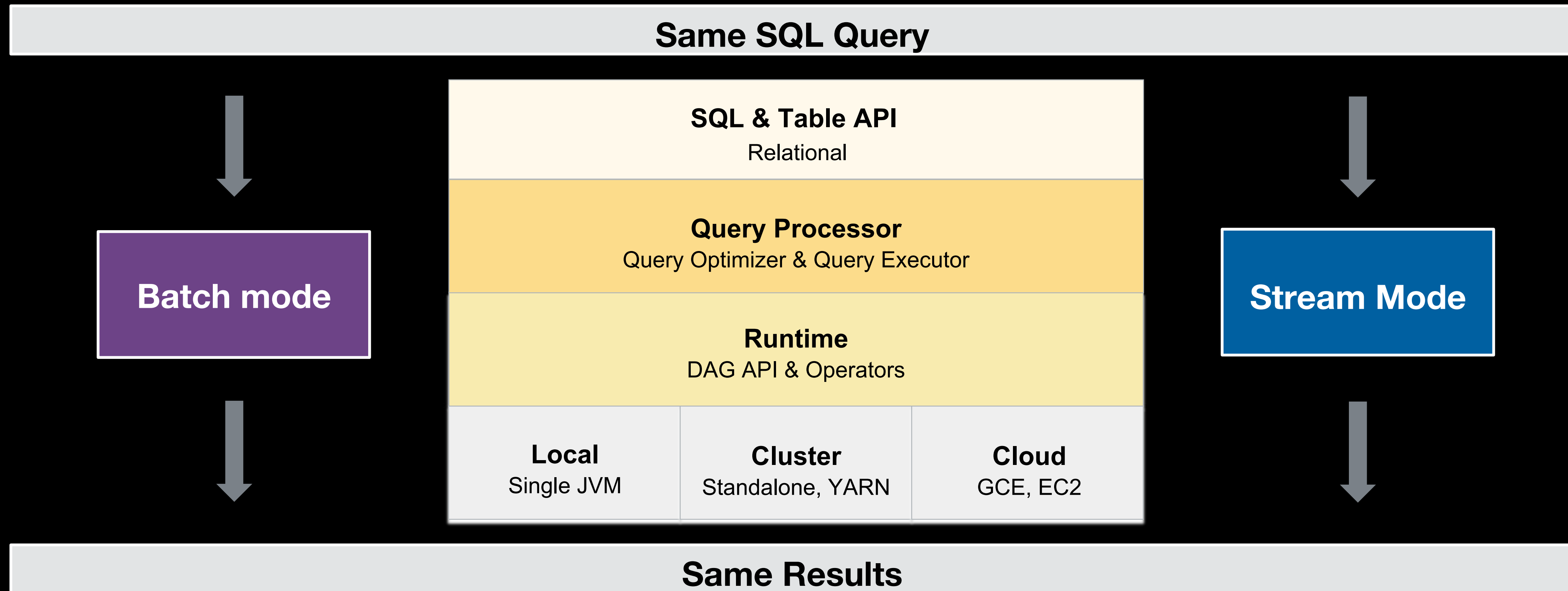
Jiangjie (Becket) Qin / 秦江杰
Staff Engineer at Alibaba

2019/6/28, Apache Flink Meetup Beijing



- **Why TableAPI + AI?**
- Build an AI Ecosystem on TableAPI
 - TableAPI enhancement
 - Iteration
 - Machine Learning Pipeline & ML Libs
 - Deep Learning on Flink (TensorFlow, PyTorch)

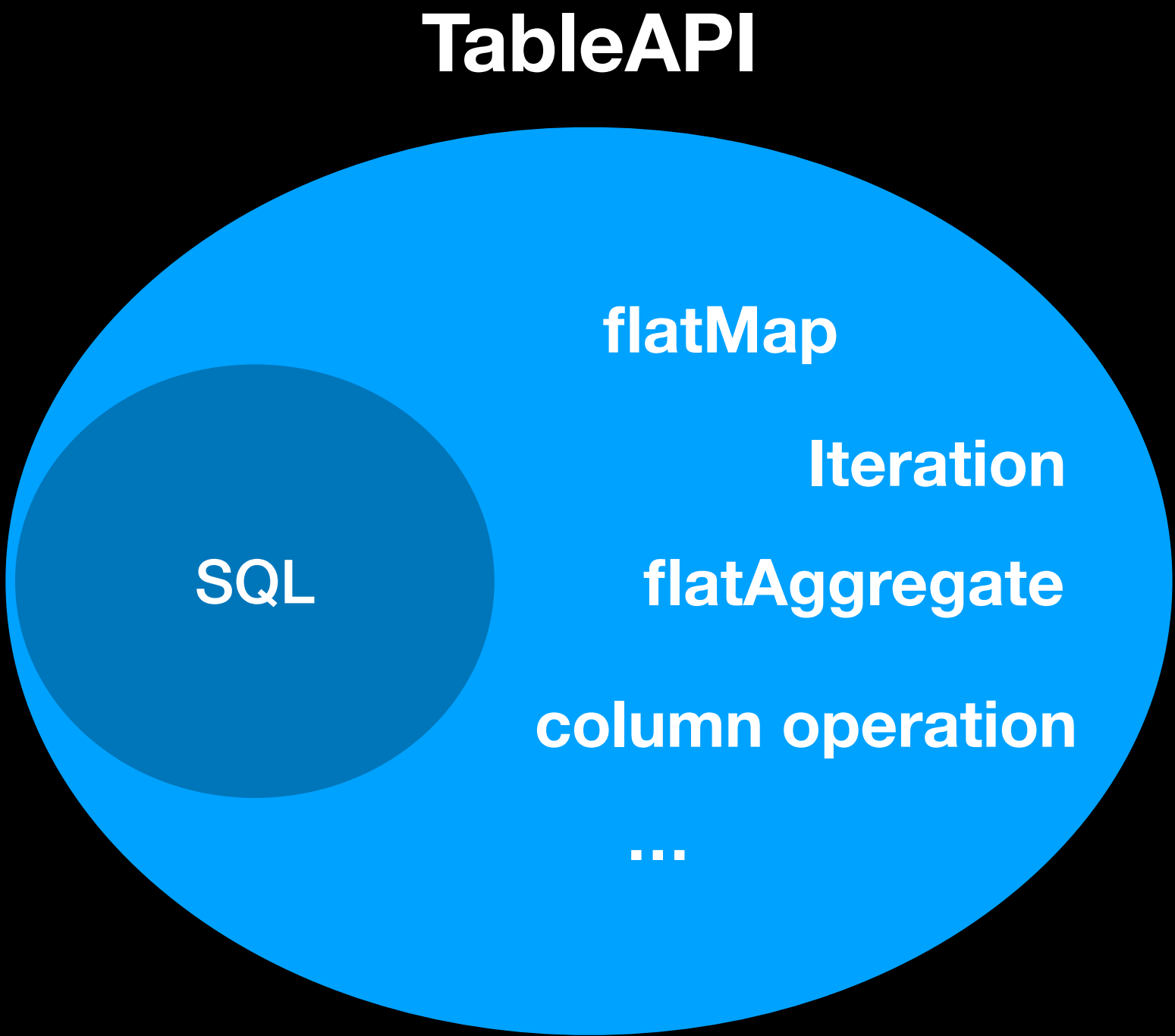
TableAPI: Declarative API with Optimization



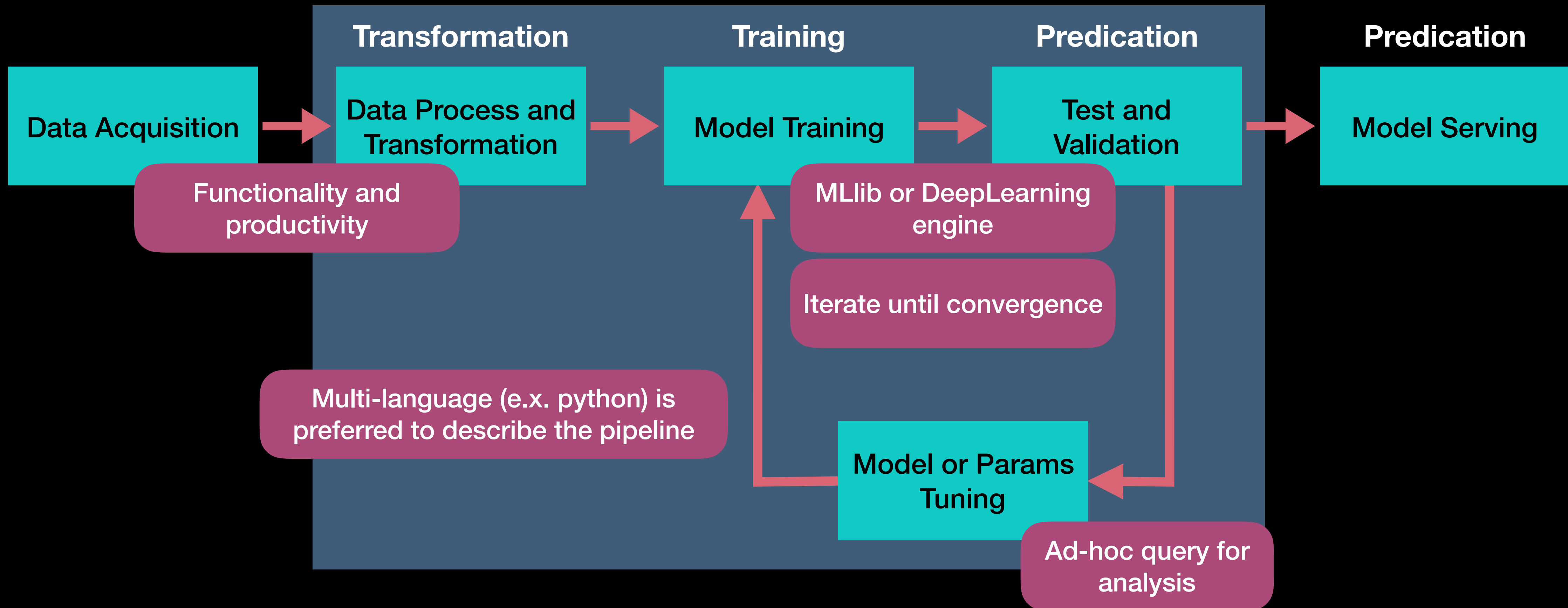
similar as SQL, batch&stream unified, declarative API with nature optimization framework

TableAPI is more than SQL

	TableAPI	SQL	e.g.
Stream and batch unified	Y	Y	SELECT/AGG/ WINDOW etc.
Functional scalability	Y	N	flatAgg/ Column operations etc.
Rich expressive	Y	N	map/flatMap/ intersect etc.
Compile check	Y	N	Java/Scala IDE



AI Computing Pipeline



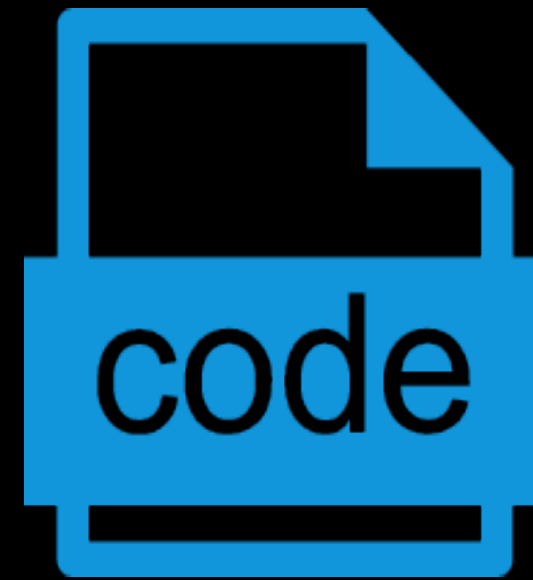
AI Requirements



**Functionality
and Productivity
Enhancement**



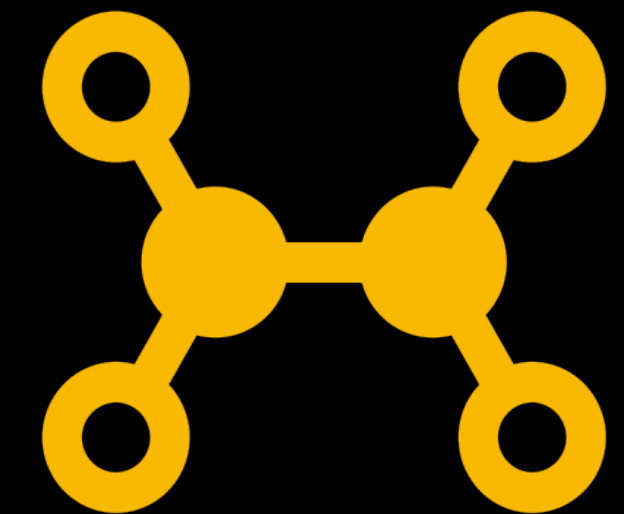
**Interactive
programming**



Multi-language



Iteration



**Execute MLlib
and DL engine**



- Why Table API + AI
- Build an AI Ecosystem
 - TableAPI enhancement
 - Iteration
 - Machine Learning Pipeline & ML Libs
 - Deep Learning on Flink (TensorFlow, PyTorch)

Introducing Row-based processing APIs

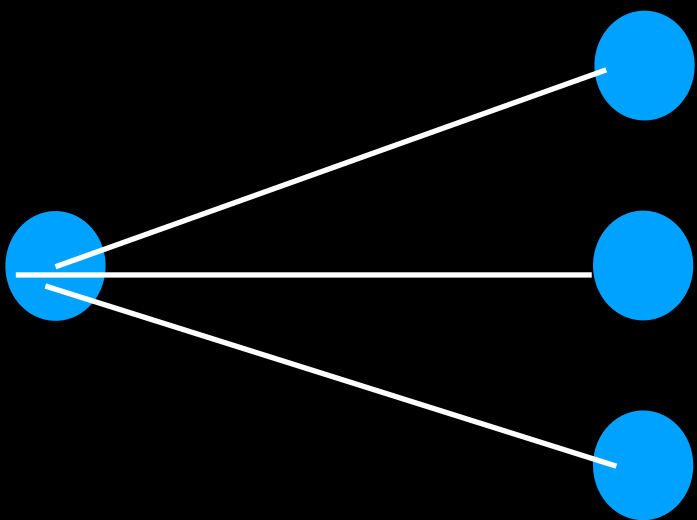


Apache Flink

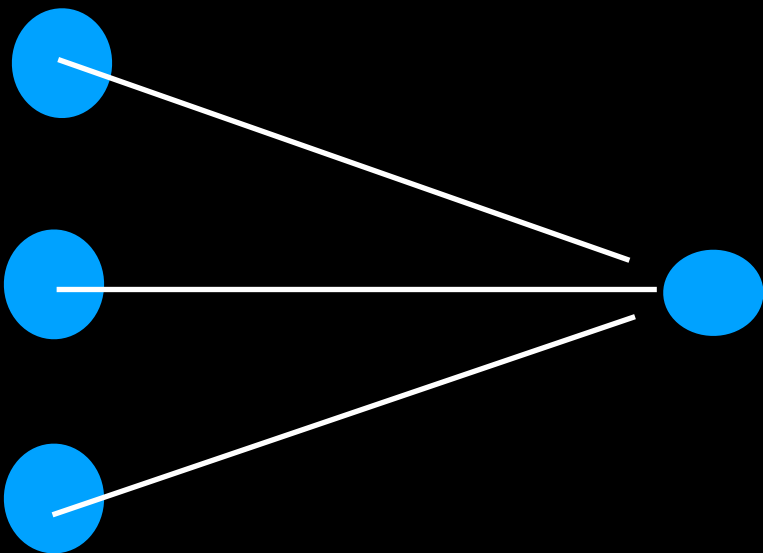
	Single Row Input	Multiple Row Input
Single Row Output	ScalarFunction (select/ map)	AggregateFunction (select/ aggregate)
Multiple Row Output	TableFunction (cross join/ flatmap)	TableAggregateFunction (flatAggregate)



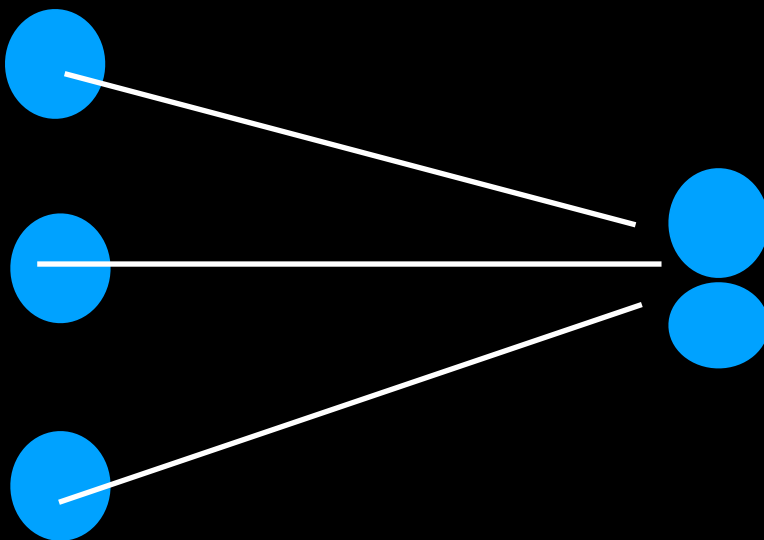
ScalarFunction
(select/map)



TableFunction
(cross join/flatmap)

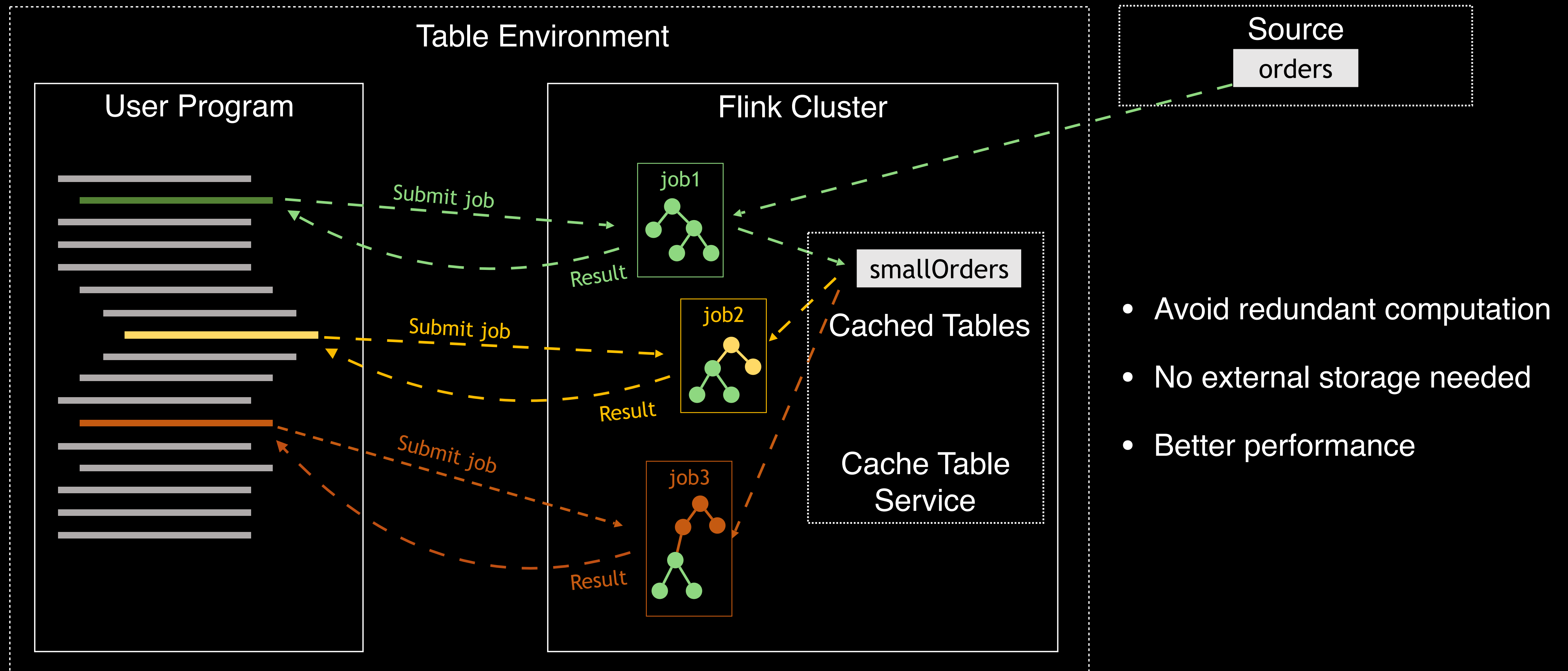


AggregateFunction
(select/aggregate)

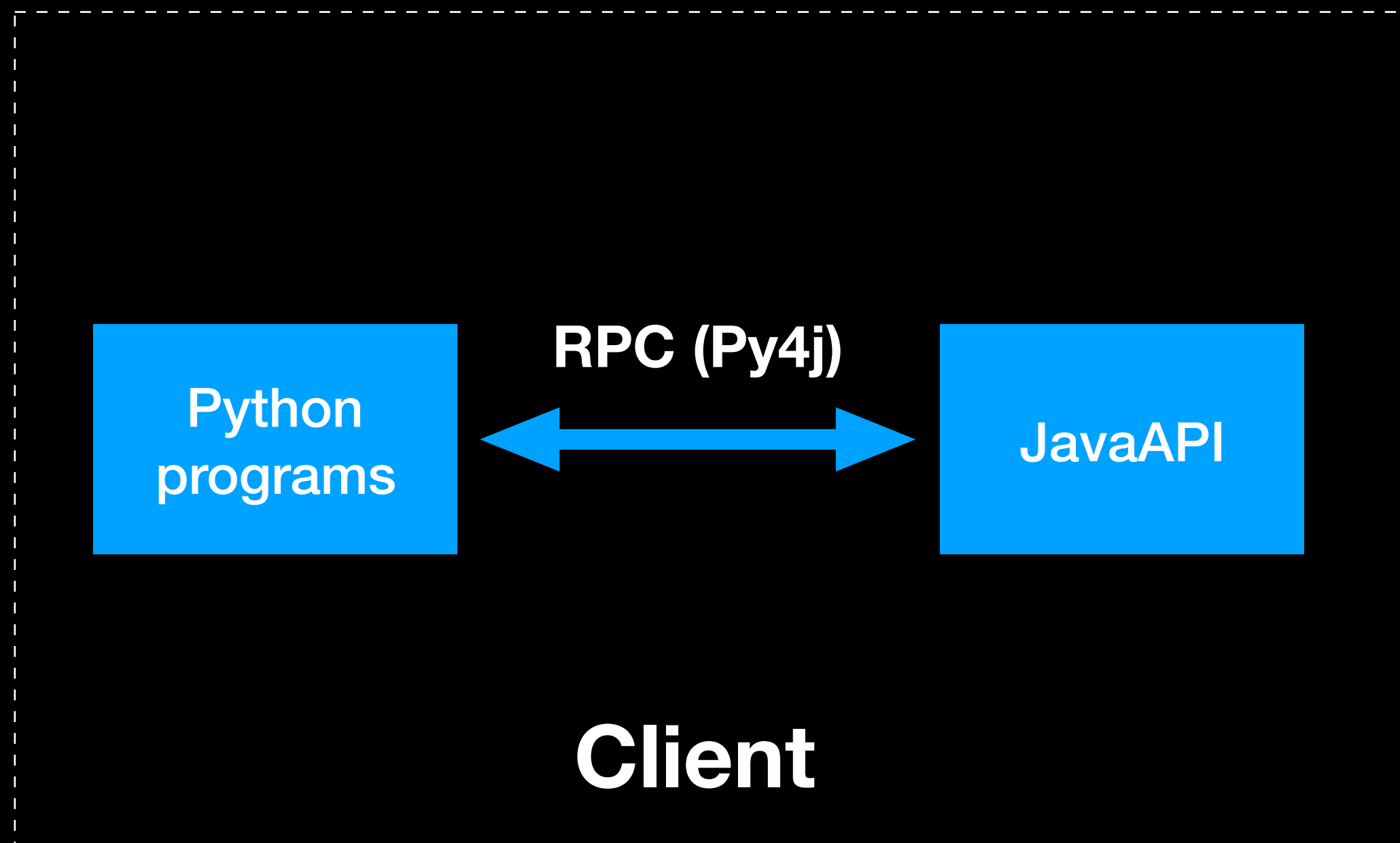


TableAggregateFunction
(flatAggregate)

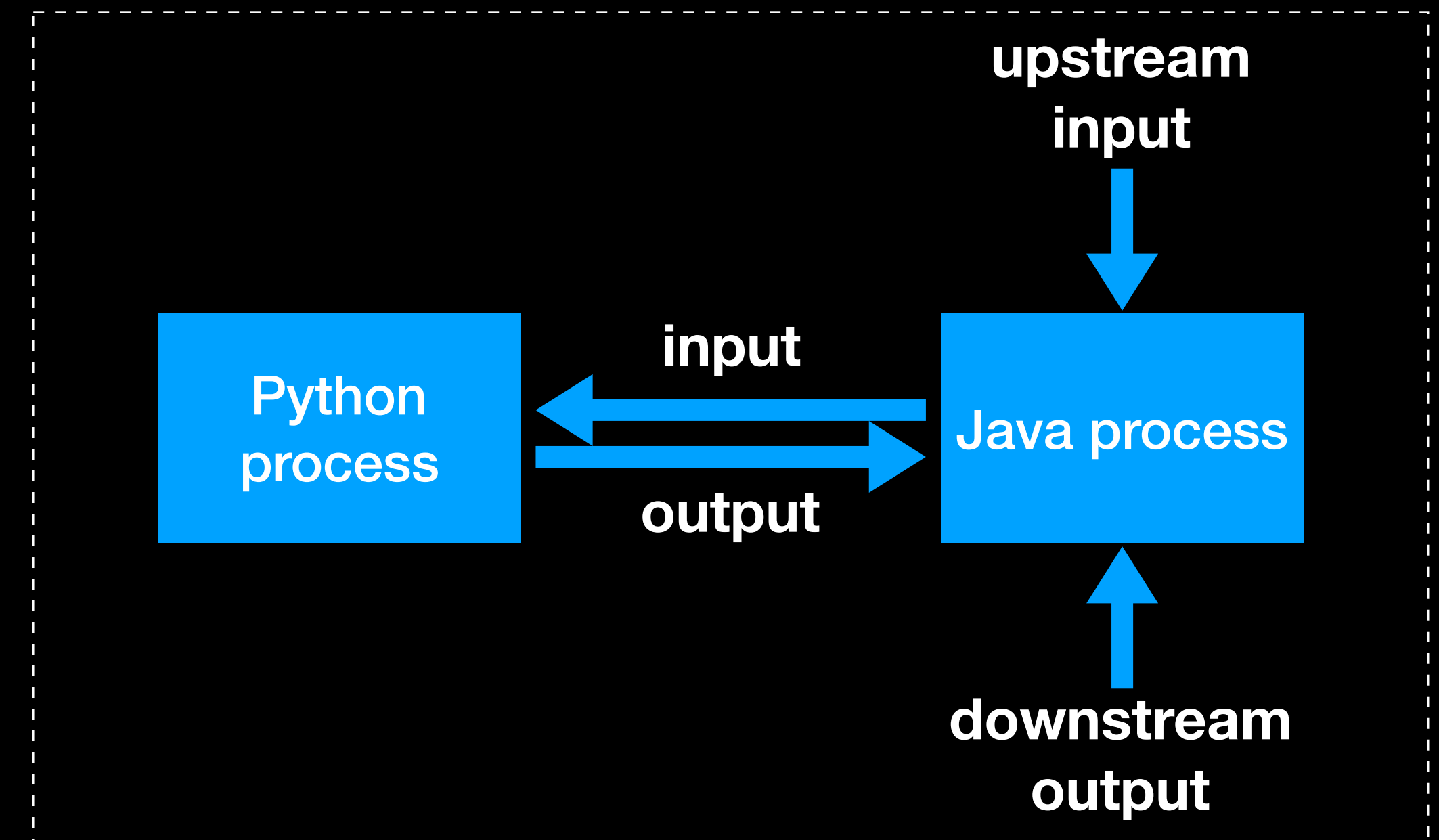
Introducing Table.cache() for Interactive Programming Apache Flink



Python TableAPI



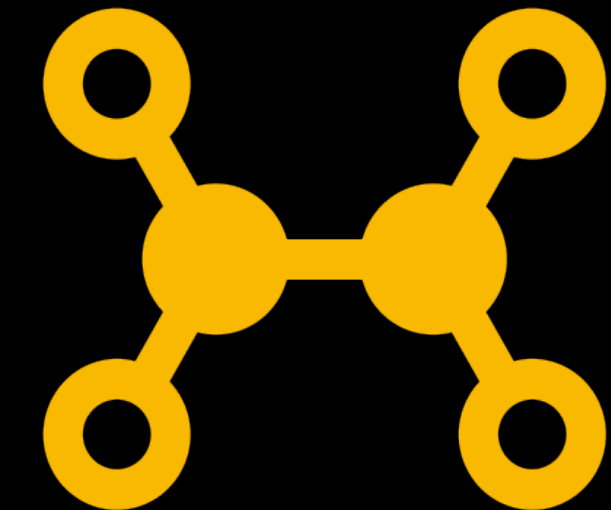
Python UDF



- Why Table API + AI
- Build an AI Ecosystem
 - TableAPI enhancement
 - **Iteration**
 - Machine Learning Pipeline & ML Libs
 - Deep Learning on Flink (TensorFlow, PyTorch)

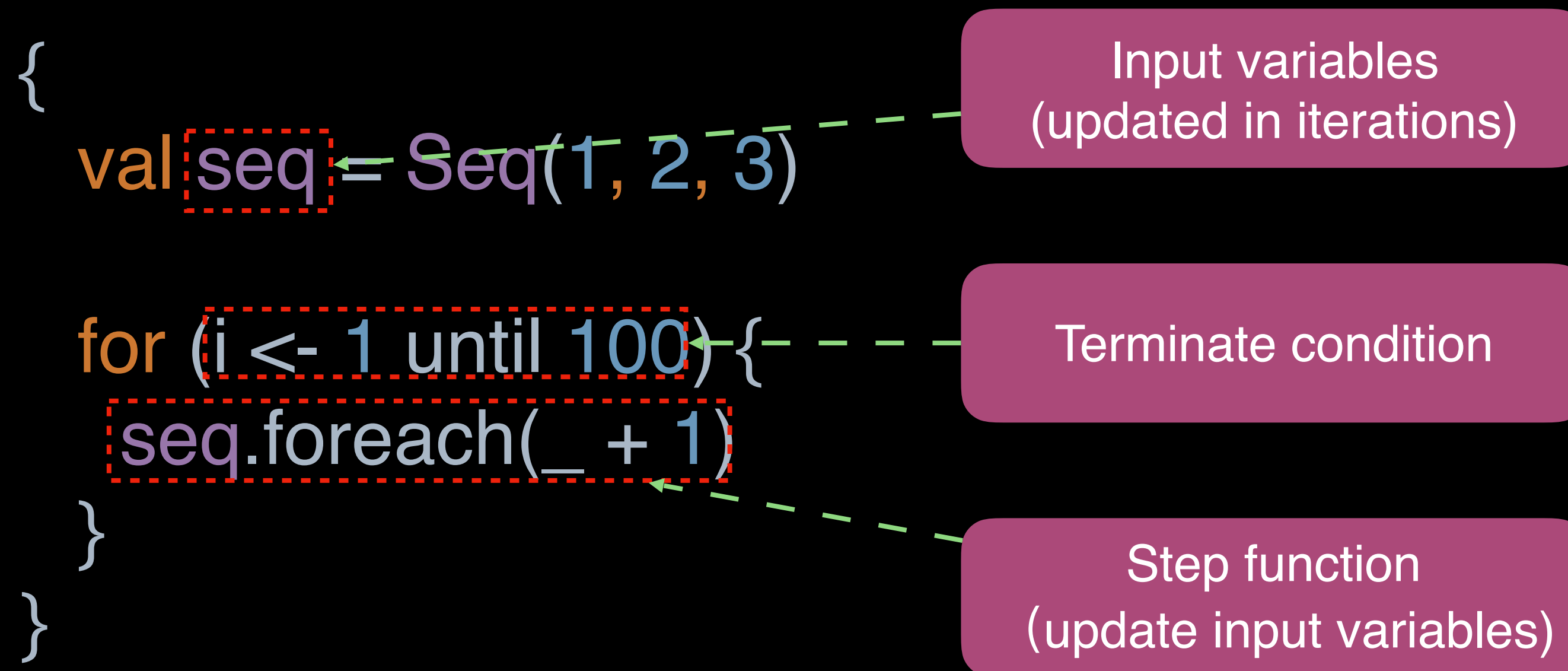


Iteration



**Execute MLlib
and DL engine**

- Execute same processing logic repeatedly
- Input of each round is the output of the previous round





Without any new API:

Leveraging interactive programming

With a new tableAPI:

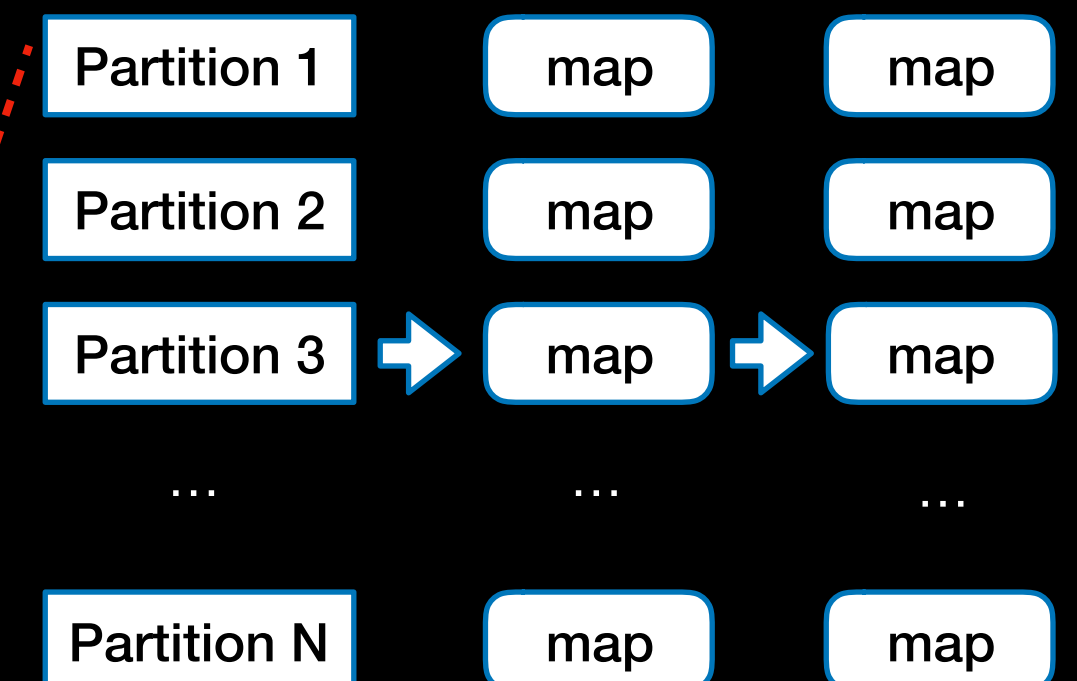
Introducing `table.iterate()`

Approach 1: Iteration with interactive programming Apache Flink

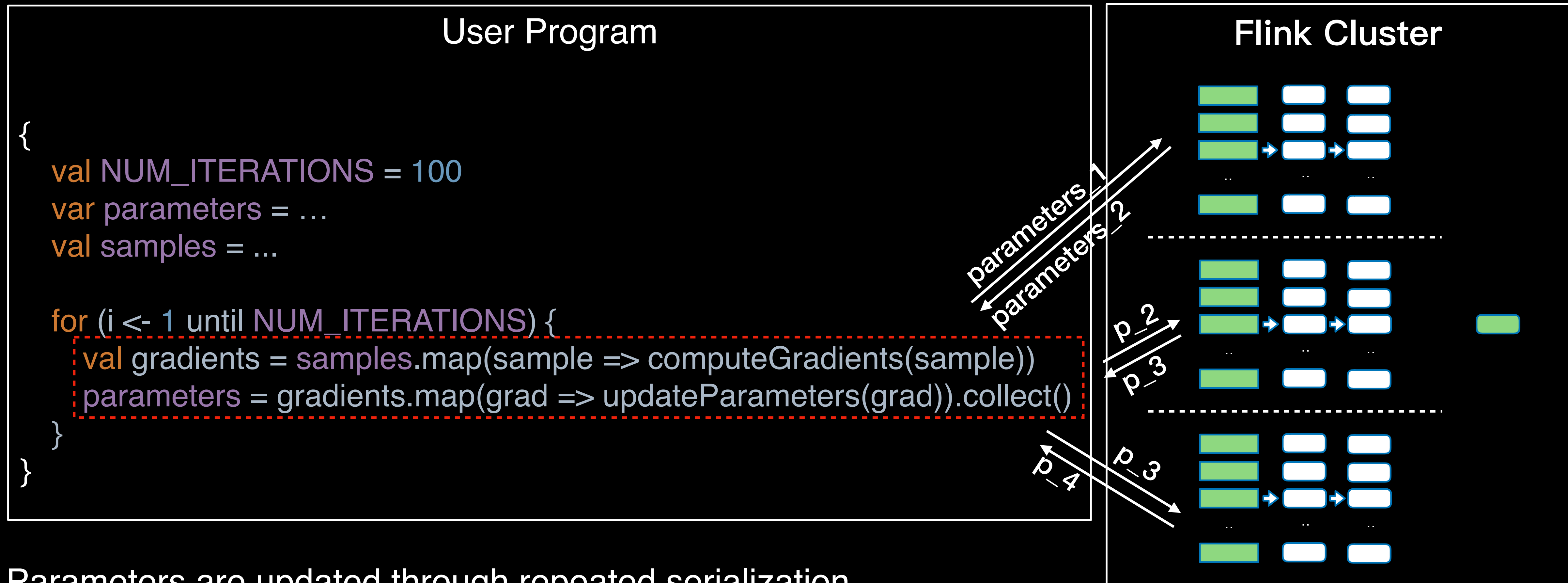
User Program

```
{  
  val NUM_ITERATIONS = 100  
  var parameters = ...  
  val samples = ...  
  
  for (i <- 1 until NUM_ITERATIONS) {  
    val gradients = samples.map(sample => computeGradients(sample))  
    parameters = gradients.map(grad => updateParameters(grad)).collect()  
  }  
}
```

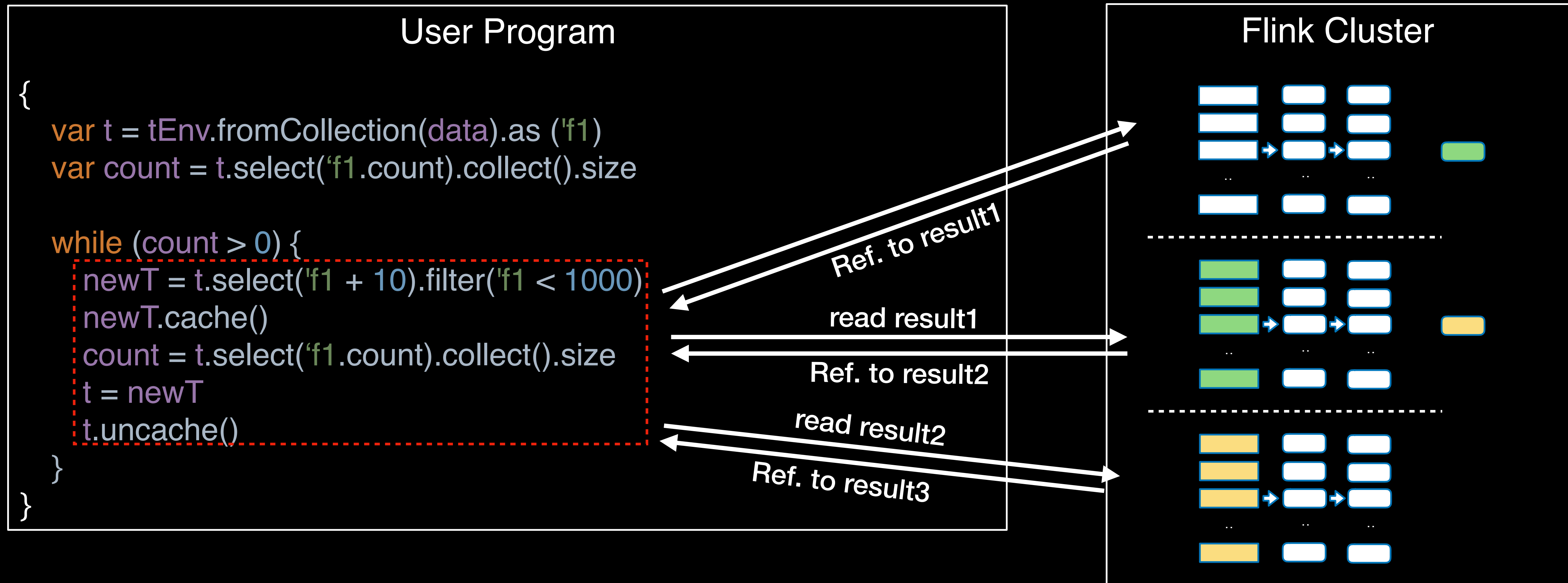
Flink Cluster



Approach 1: Iteration with interactive programming Apache Flink



Approach 1: Iteration with interactive programming Apache Flink



Variables updated with cached result from last round

Approach 1: Iteration with interactive programming **Apache Flink**

- logic of iteration is implemented by the users
- have to submit a job for each round of iteration
- have to manage the lifecycle of the cached tables

Pros

Simple and intuitive

Good flexibility in code logic

Cons

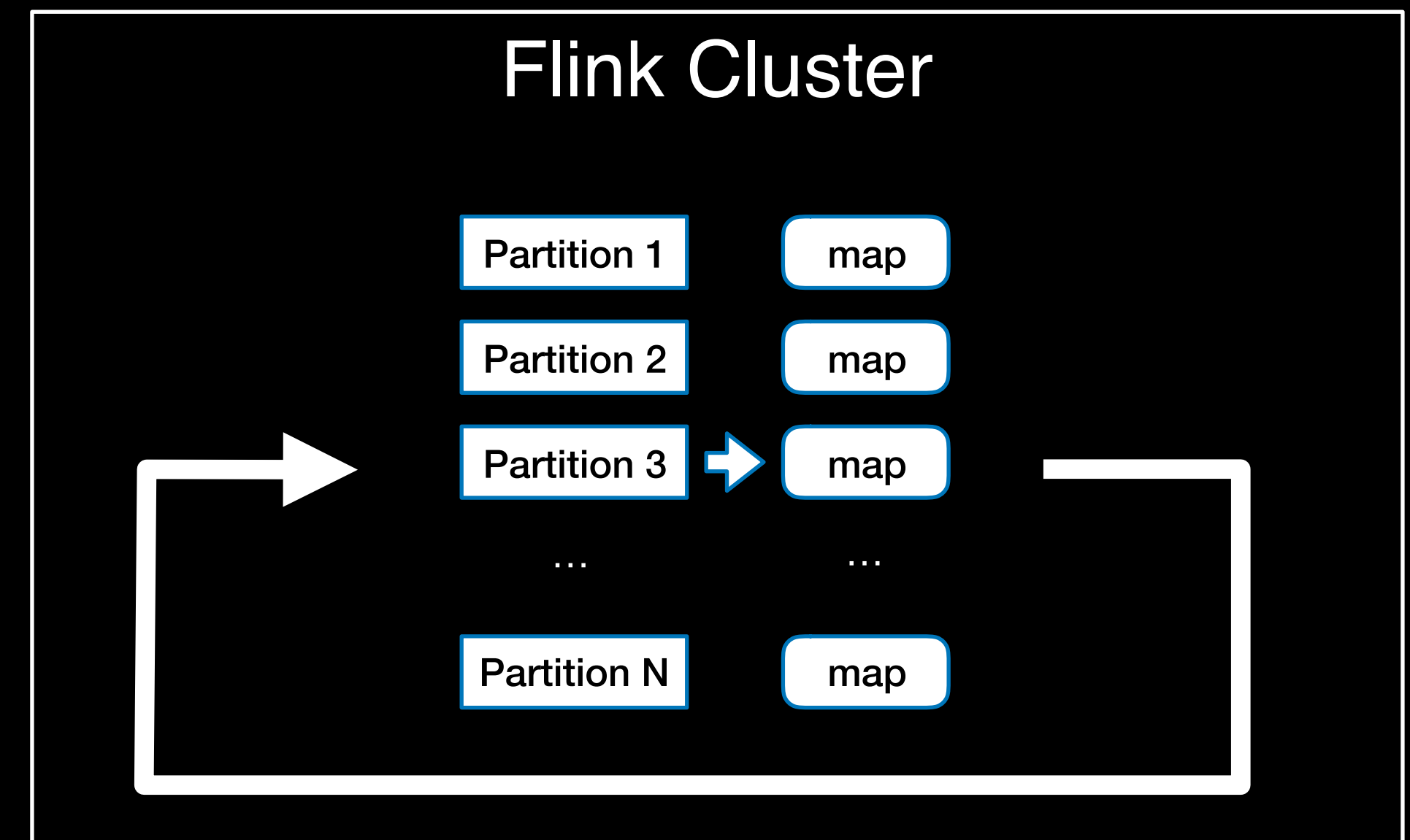
Overhead of job submission

Synchronization after each round

Approach 2: Introducing Iterate on TableAPI



- Native iteration implemented by the processing engine
- Feedback edge on the processing DAG
- Improve the caveats in DataSet and DataStream iterations
 - Support multi-variable iteration
 - Support nested iteration





```
{  
  val a: Table = ...  
  val b: Table = ...  
  val resultSeq = Table.iterate(a, b) {  
    val next_a = b.select('v_b + 1 as 'v_a)  
    val next_b = next_a.select('v_a * 2 as 'v_b)  
    Seq(next_a, next_b)  
  }.times(10)  
}
```

Iteration variables

Step function

Termination condition

Table.iterate - Nested Iterations



Apache Flink

```
{  
  val a: Table = ...  
  val b: Table = ...  
  val resultSeq = iterate(a, b) {  
    val next_a = iterate(a) {  
      Seq(a.select('v_a + 1 as 'v_a))  
    }.times(100).head  
  
    val next_b = next_a.select('v_a * 2 as 'v_b)  
  
    Seq(next_a, next_b)  
  }.times(10)  
}
```



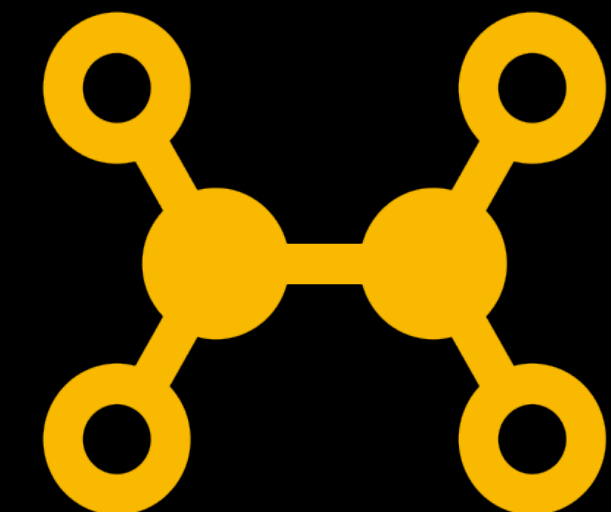
- About to contribute to Apache Flink
- Future Work
 - Native Support for SGD
 - Native Support for Online Learning Scenarios



- Why Table API + AI?
- Build an AI Ecosystem
 - TableAPI enhancement
 - Iteration
 - Machine Learning Pipeline & ML Libs
 - Deep Learning on Flink (TensorFlow, PyTorch)



Iteration

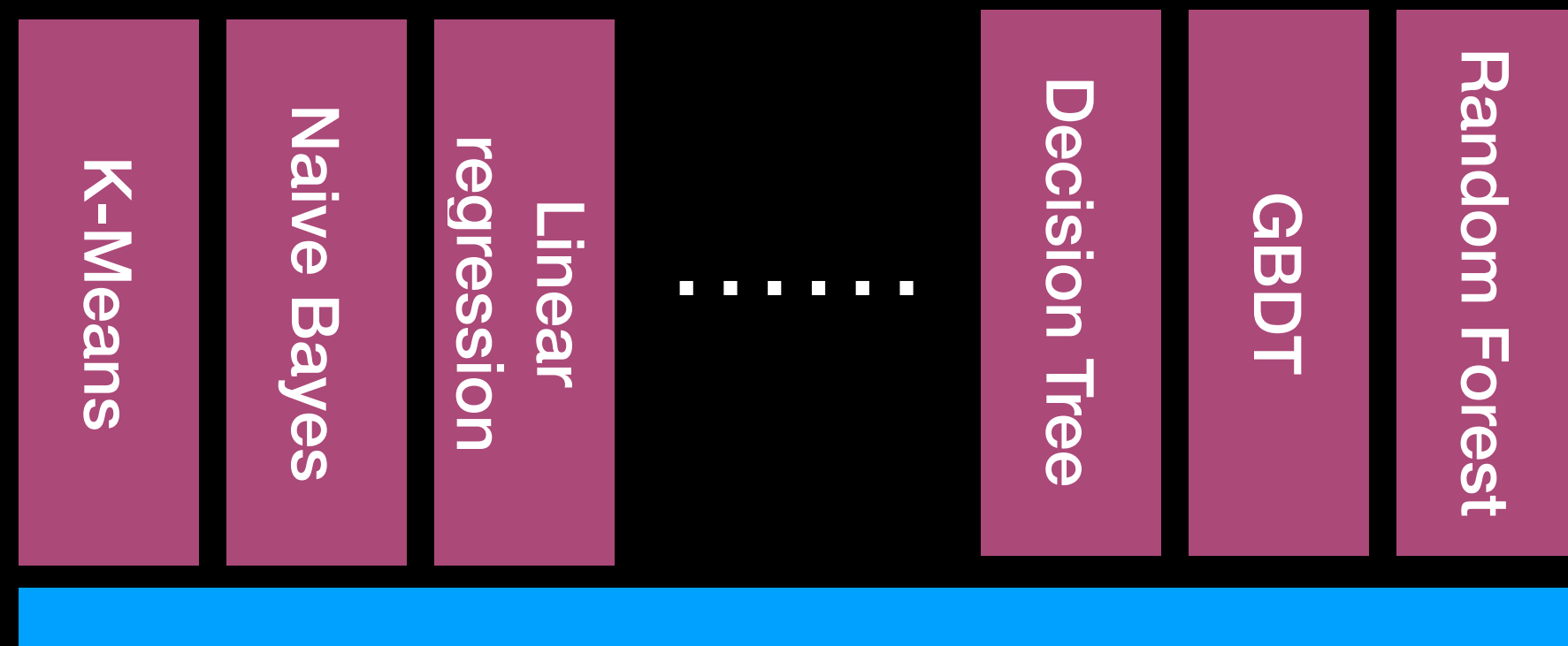


Execute MLlib
and DL engine

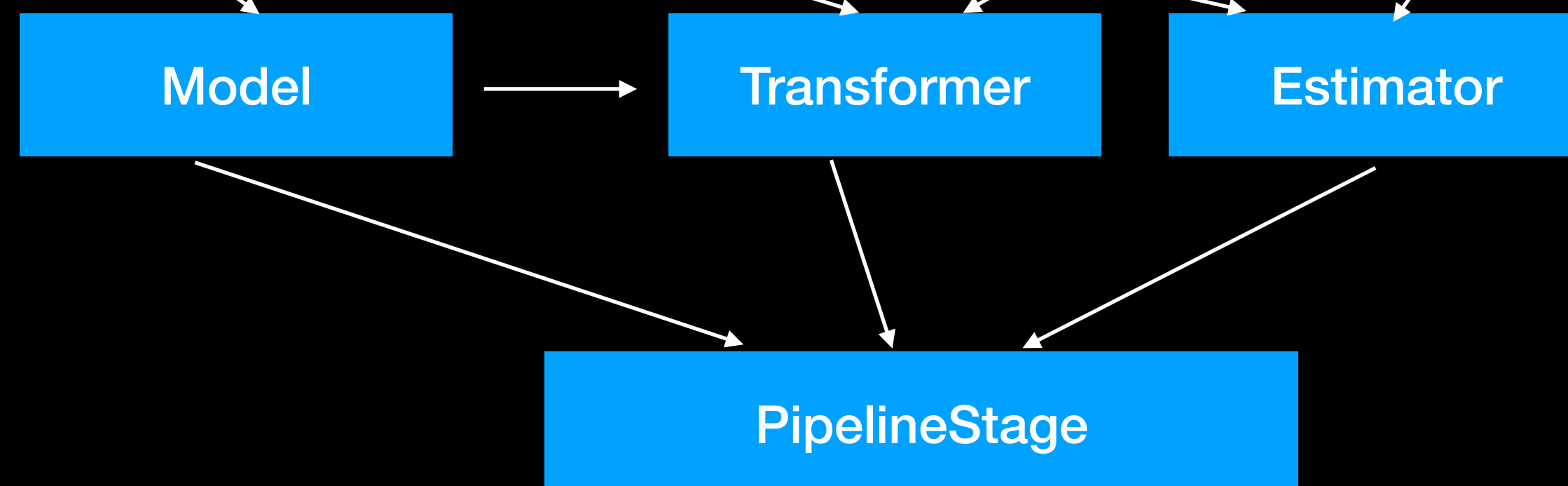
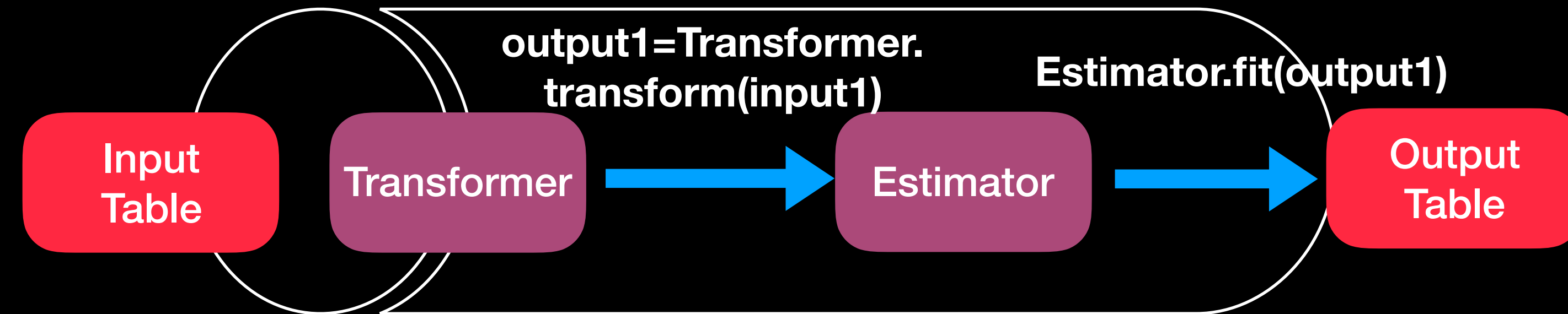
ML Pipeline - Overview

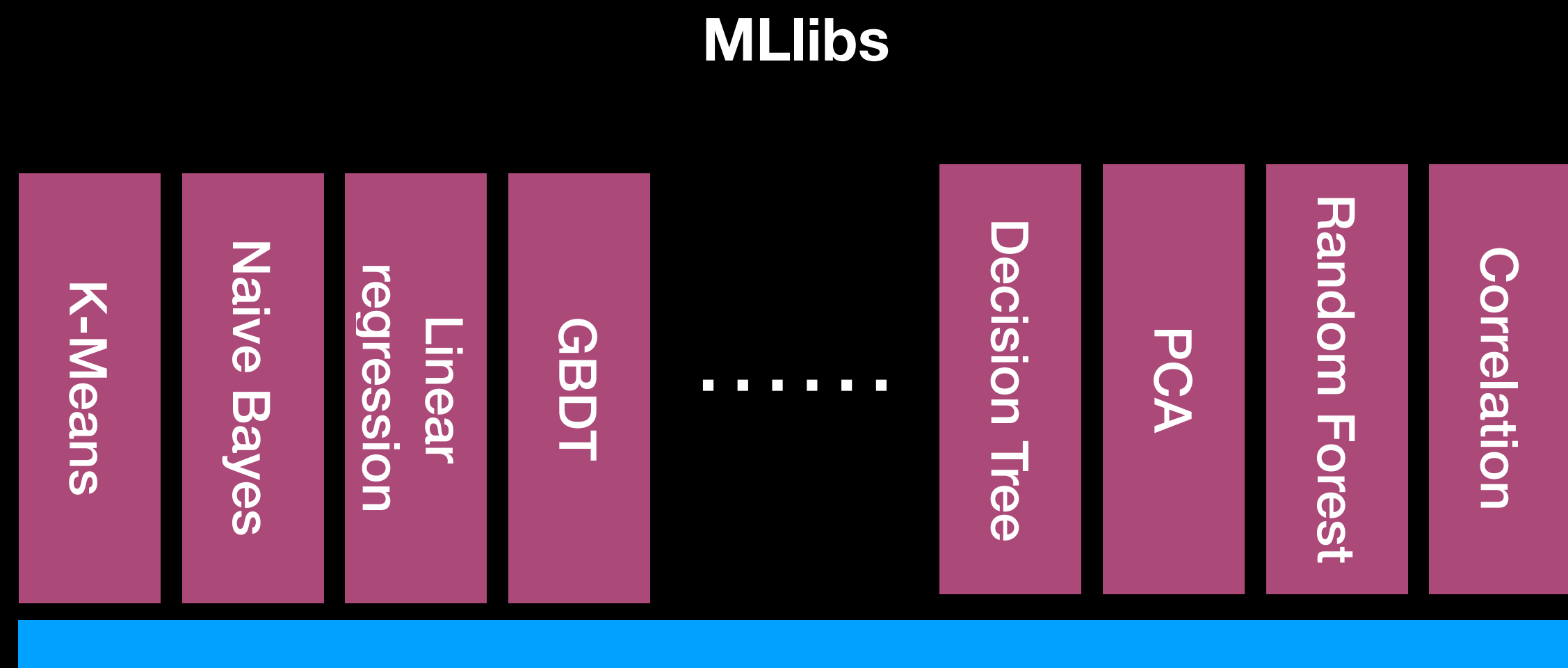


ML Lib Developers



ML Lib Users

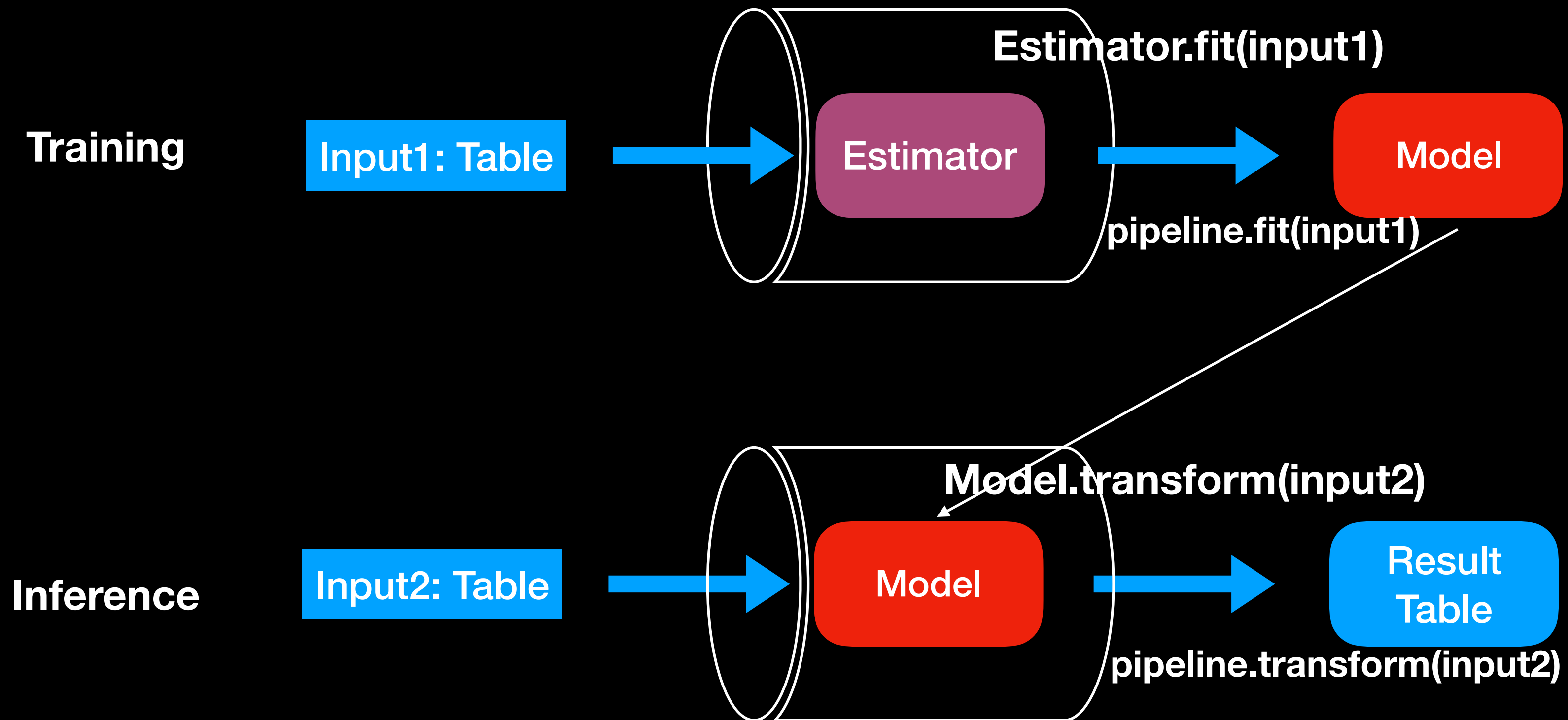




Rewrite Flink ML Libs

- ML pipeline based
- Table API based
- Battle tested algorithm

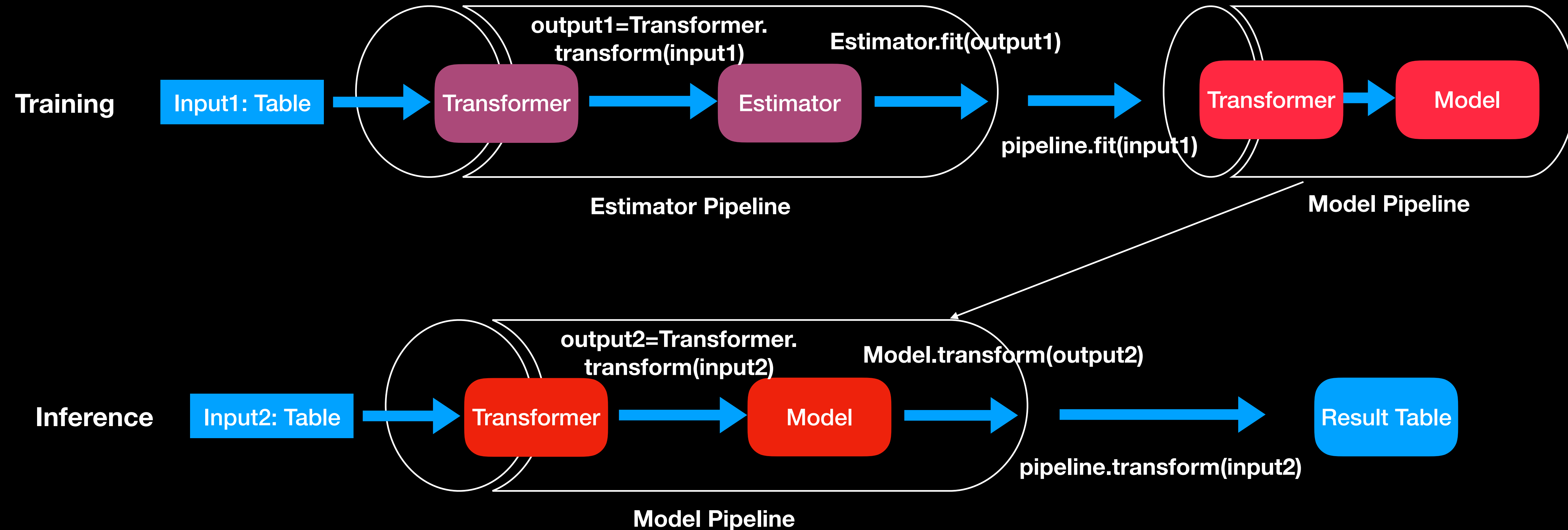
ML Pipeline - Simple Case



ML Pipeline



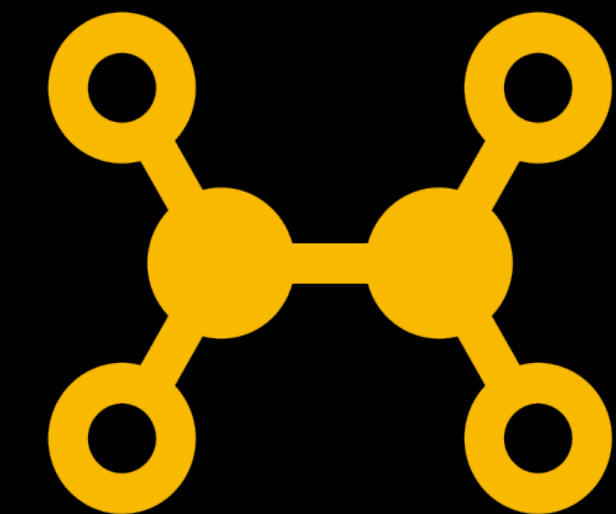
Apache Flink



- Why Table API + AI?
- Build an AI Ecosystem
 - TableAPI enhancement
 - Iteration
 - Machine Learning Pipeline & ML Libs
 - Deep Learning on Flink (TensorFlow, PyTorch)



Iteration

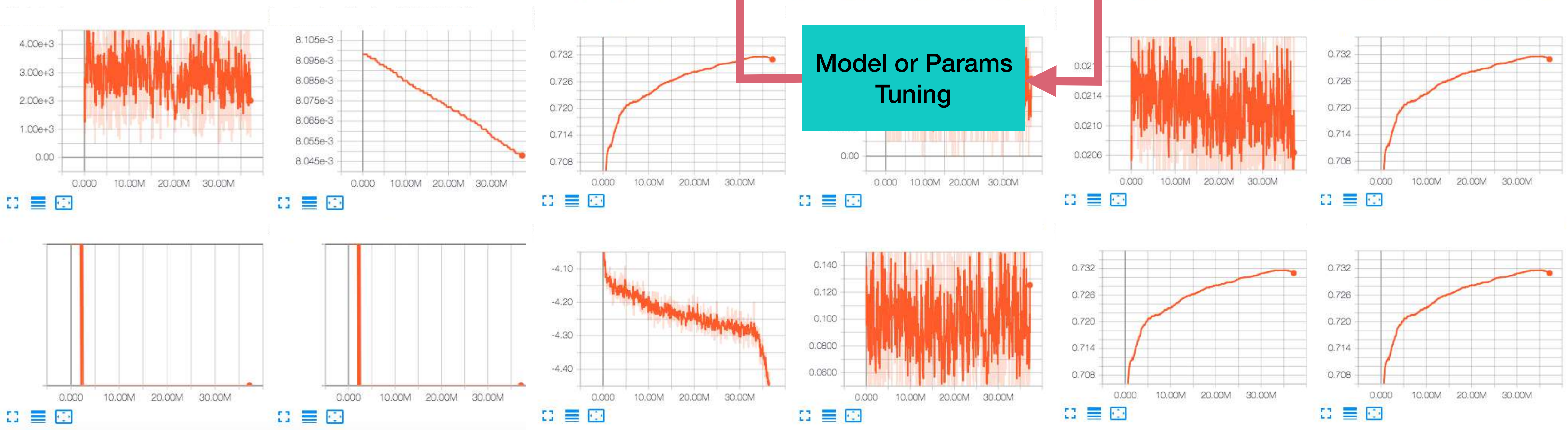


**Execute MLlib
and DL engine**

Deep Learning Pipeline



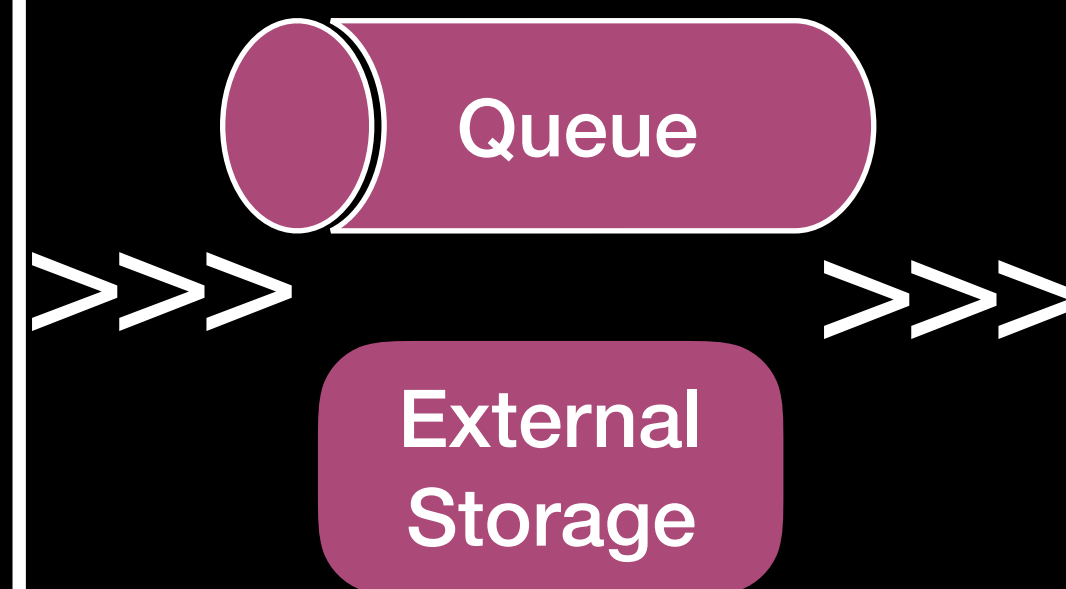
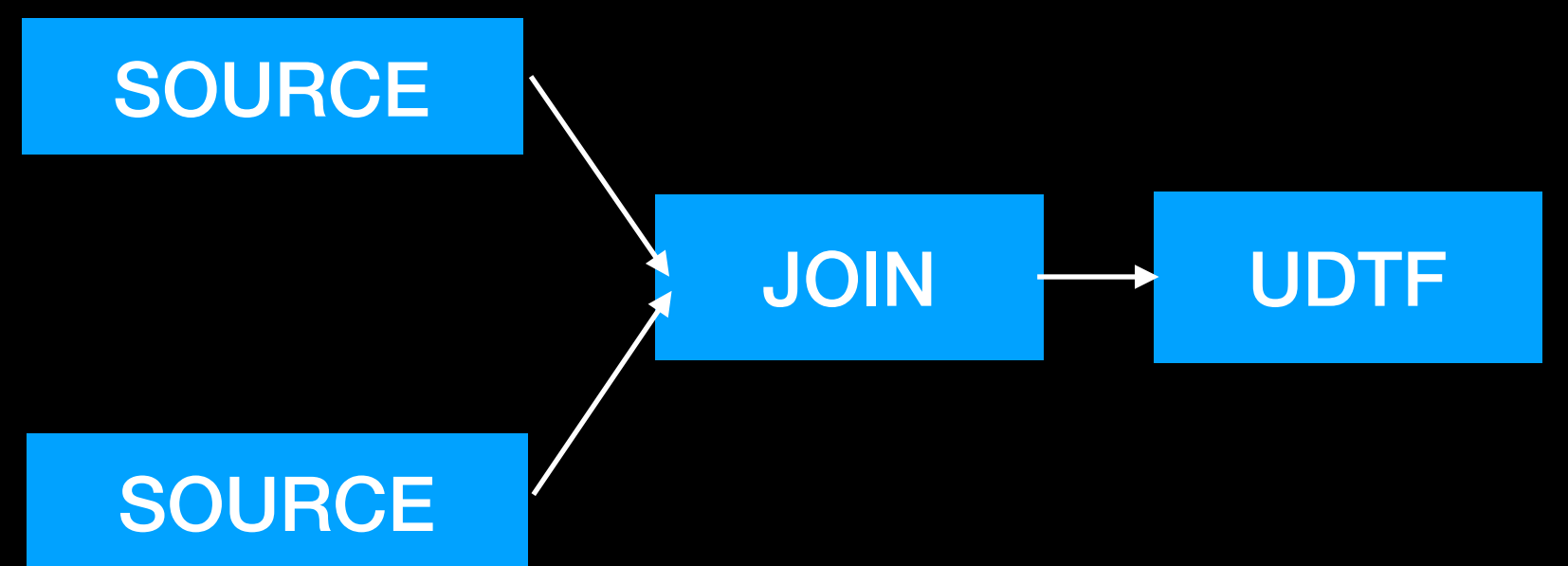
Model or Params
Tuning



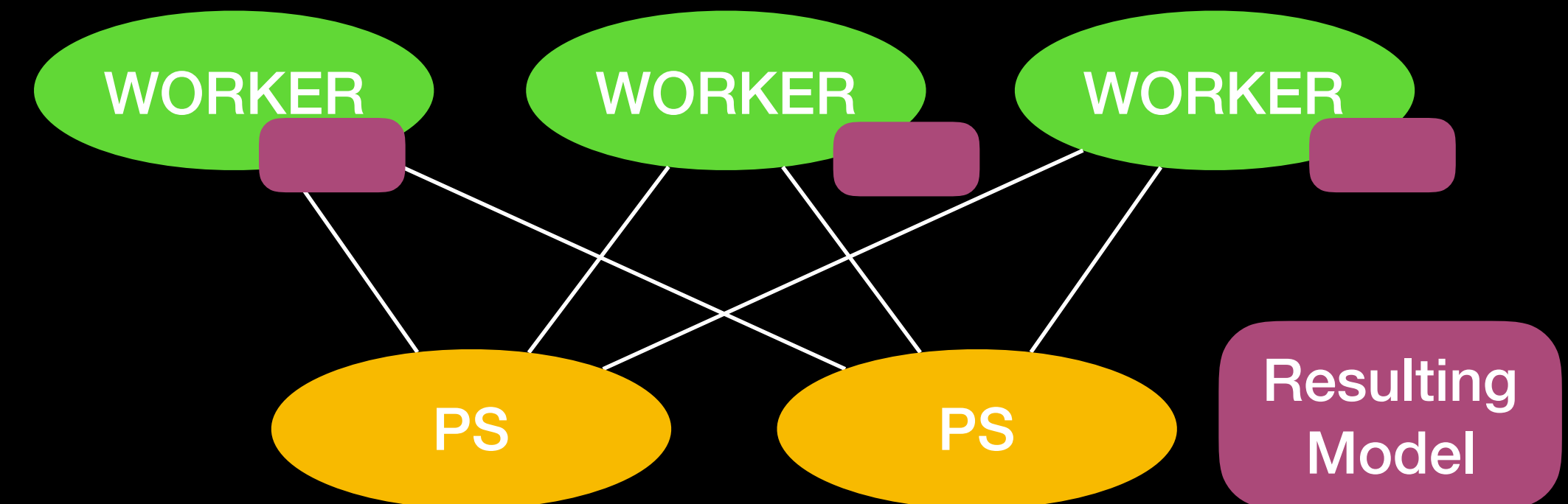
Deep Learning Pipeline



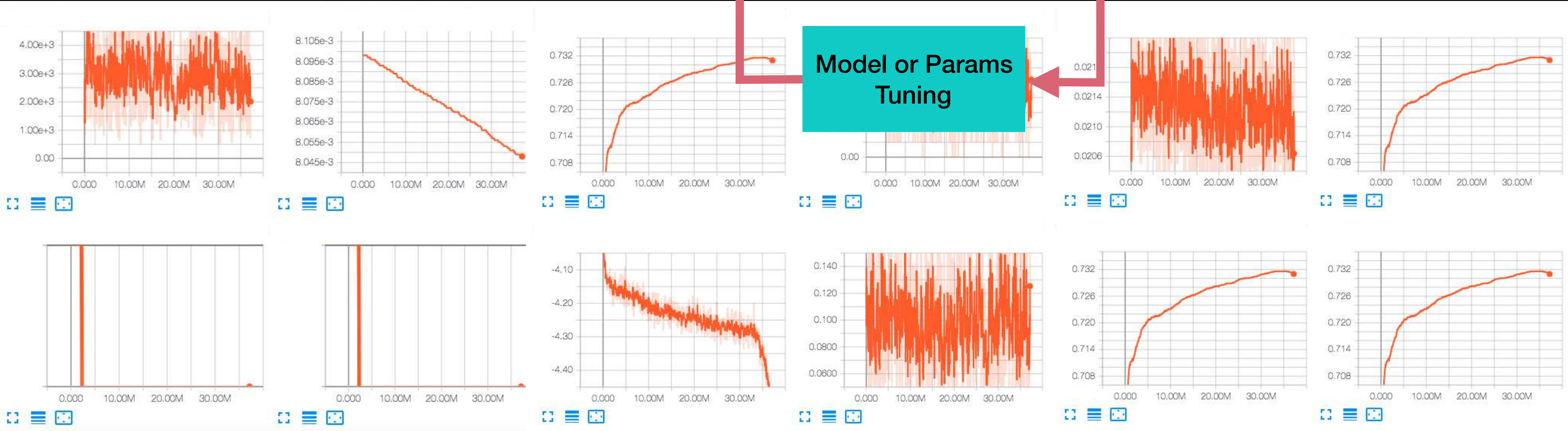
One Flink job in Cluster/Environment



Distributed TF framework in a Cluster/Environment

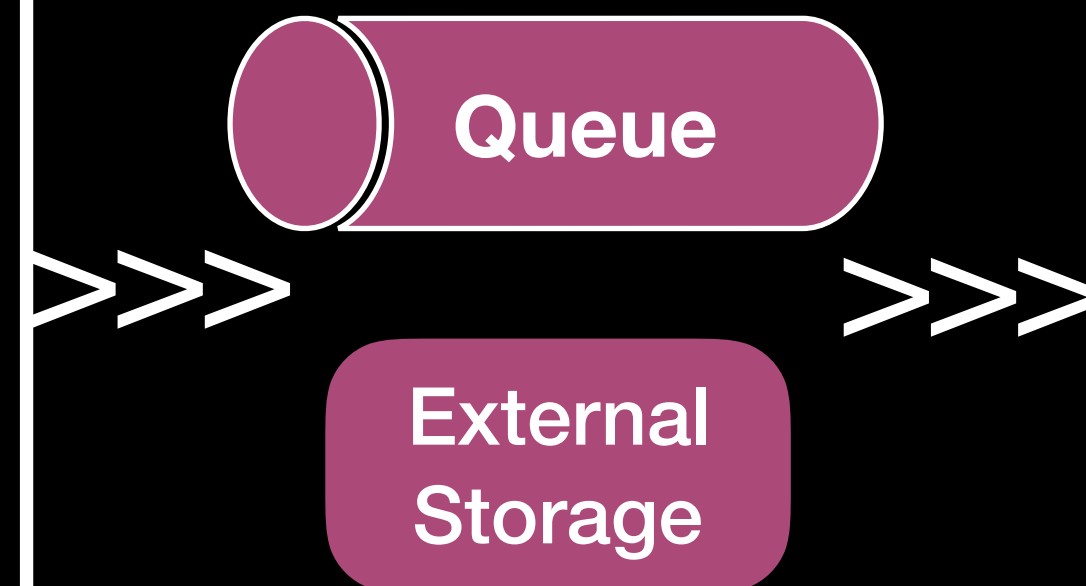
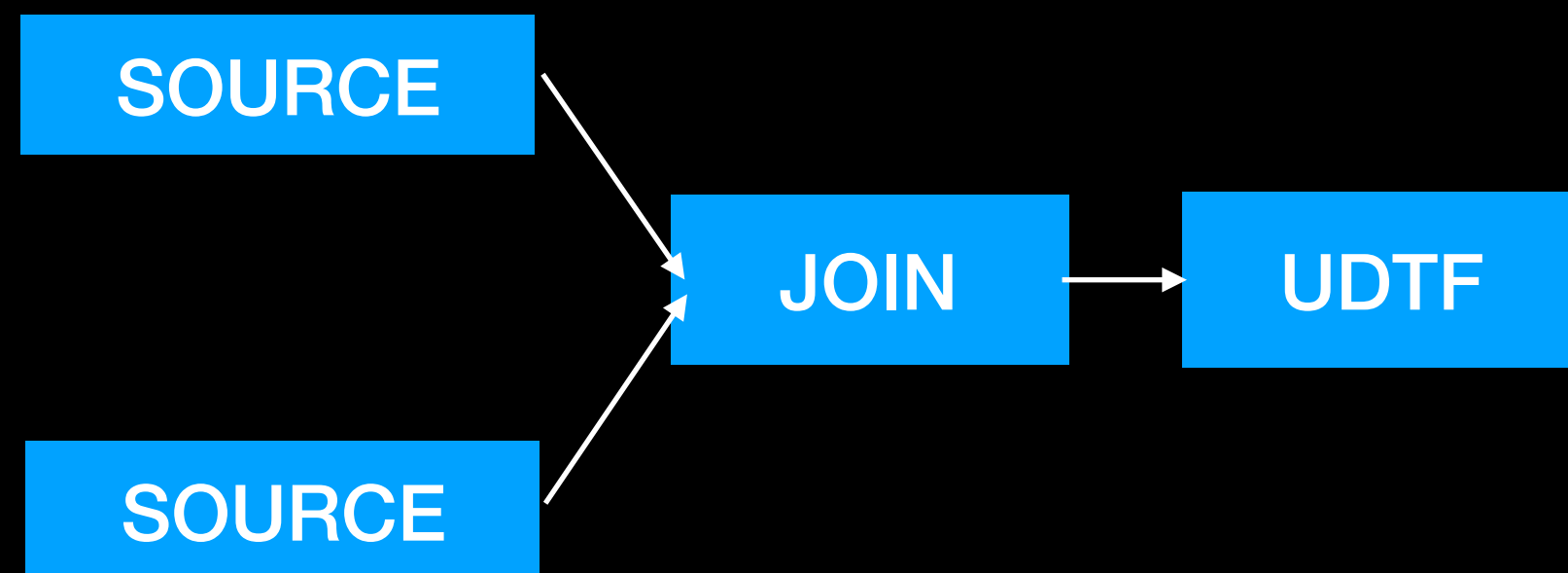


Deep Learning Pipeline

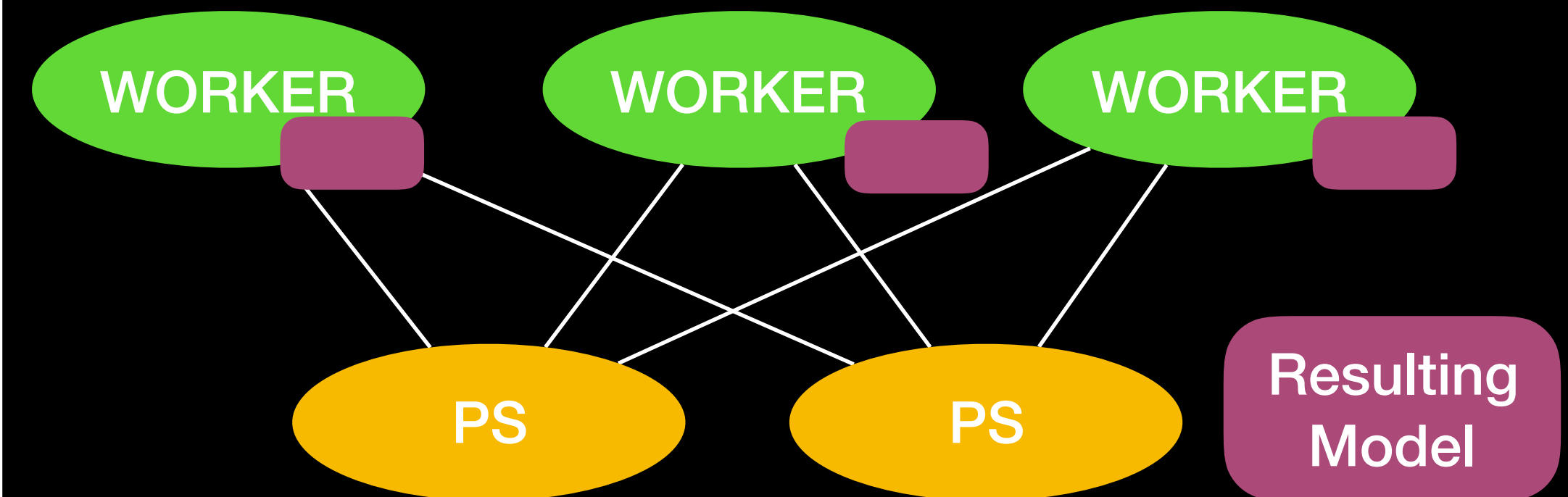


TensorFlow-Flink Integration

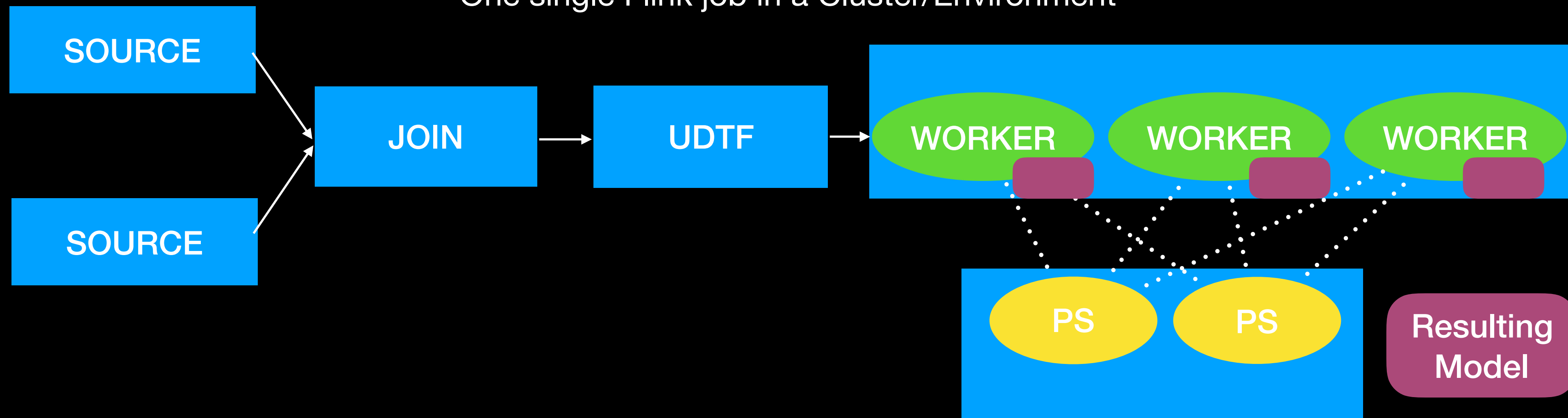
One Flink job in Cluster/Environment



Distributed TF framework in a Cluster/Environment



One single Flink job in a Cluster/Environment





- Why Table API + AI?
- Build an AI Ecosystem
 - TableAPI enhancement (FLIP29 etc.)
 - Iteration (Will contribute to Flink)
 - Machine Learning Pipeline & ML Libs (Flink1.9, FLIP-39)
 - Deep Learning on Flink (TensorFlow, PyTorch) (Will open source soon)

THANKS

Q & A

