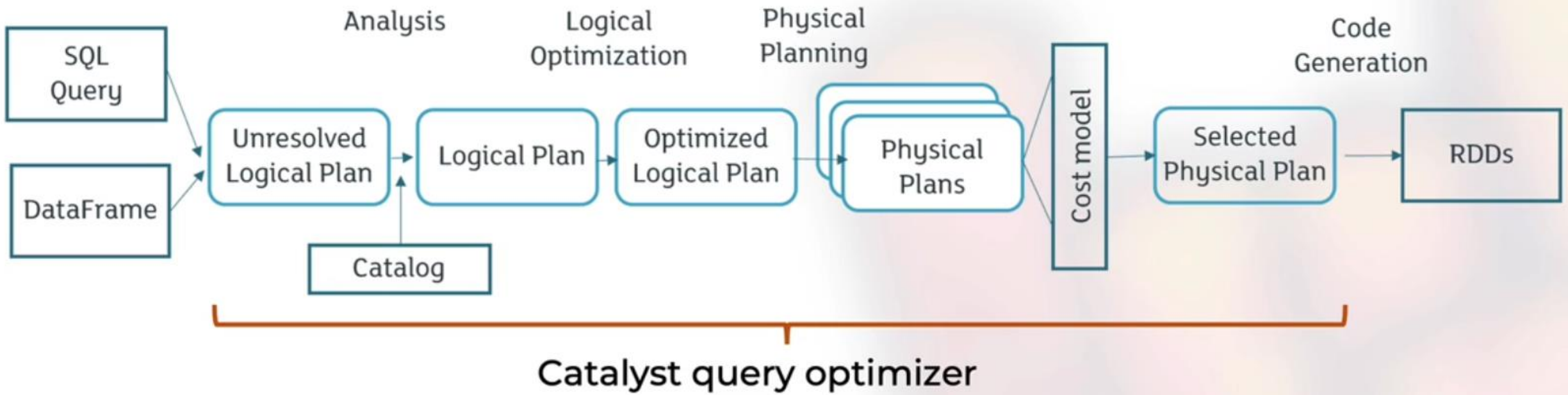
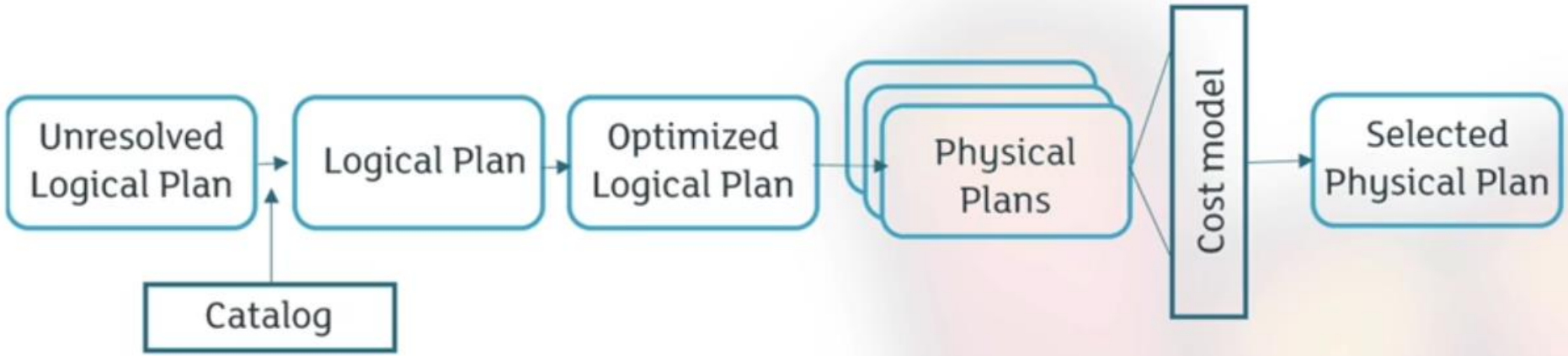
Catalyst



When you run a SQL job

- Spark knows the DF dependencies in advance – unresolved logical transformation plan
- Catalyst resolves references and expression types – resolved logical plan
- Catalyst compresses and pattern matches on the plan tree – optimized logical plan
- Catalyst generates physical execution plans

Catalyst



Steps

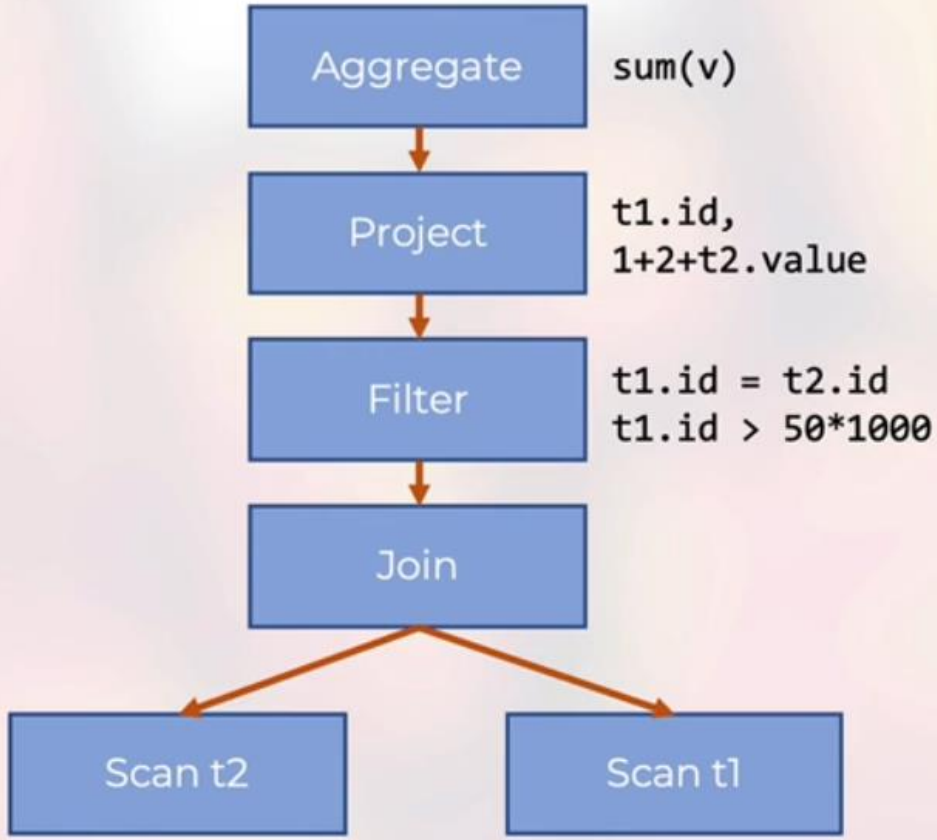
- Analysis: unresolved plan => resolved plan
  - use Catalog to find where DataFrames, columns are coming from
  - resolves column types
- Optimization: resolved plan => optimized plan
  - processes the transformation tree
  - column pruning, predicate pushdown etc
- Physical planning: optimized (logical) plan => physical execution plans
- Code generation: generate Scala code from the execution plan

Catalyst

```
t1 = (DF with numerical id and value)
t2 = (DF with numerical id)
```

```
t1.join(t2, "id")
  .where(col("id") > 50 * 1000)
  .select(col("id"), (lit(1) + lit(2) + col("value")).alias("v"))
  .agg(sum(col("v")))
```

```
SELECT sum(v)
FROM
  SELECT t1.id, 1 + 2 + t1.value as v
  FROM t1 join t2
  WHERE t1.id = t2.id AND t1.id > 50 * 1000
```

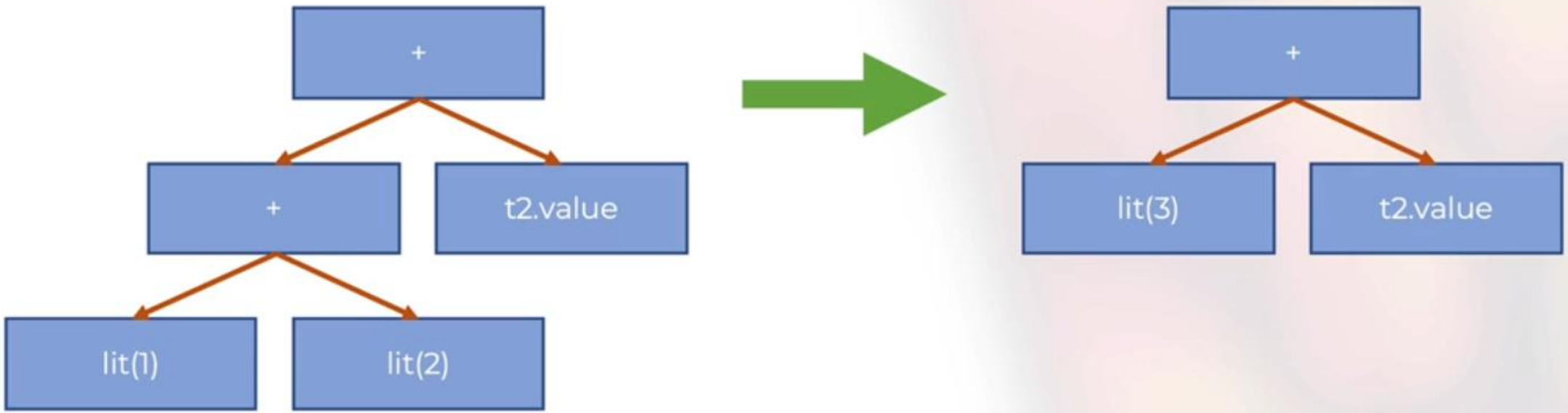


Logical plan: describes computations without how to execute them

Catalyst

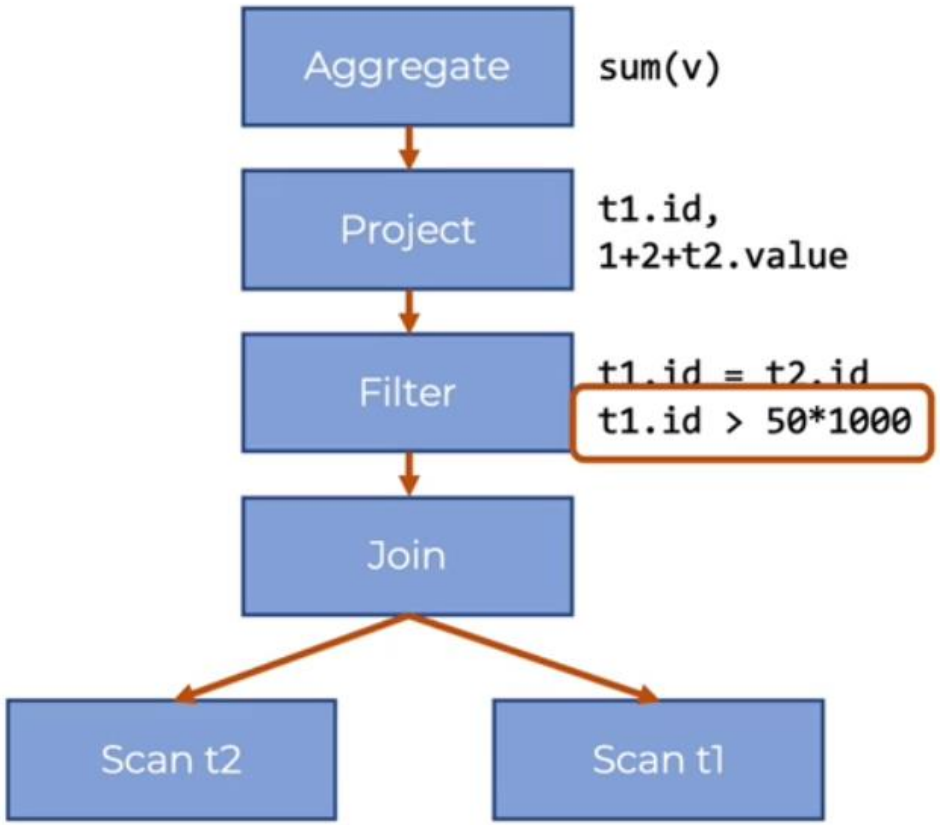
Optimization: pre-compute as much as possible

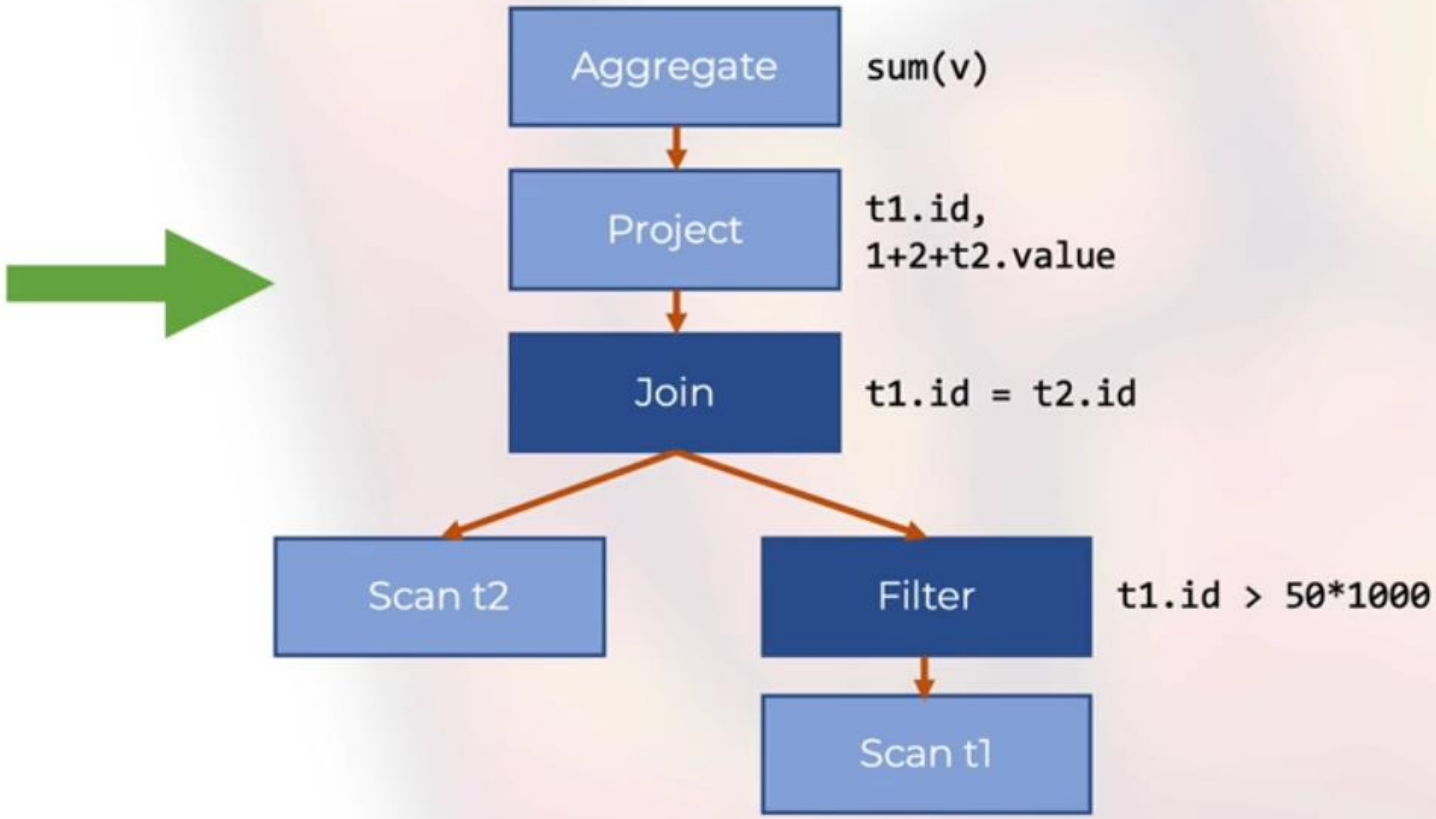
```
1 + 2 + t2.value
```



Catalyst

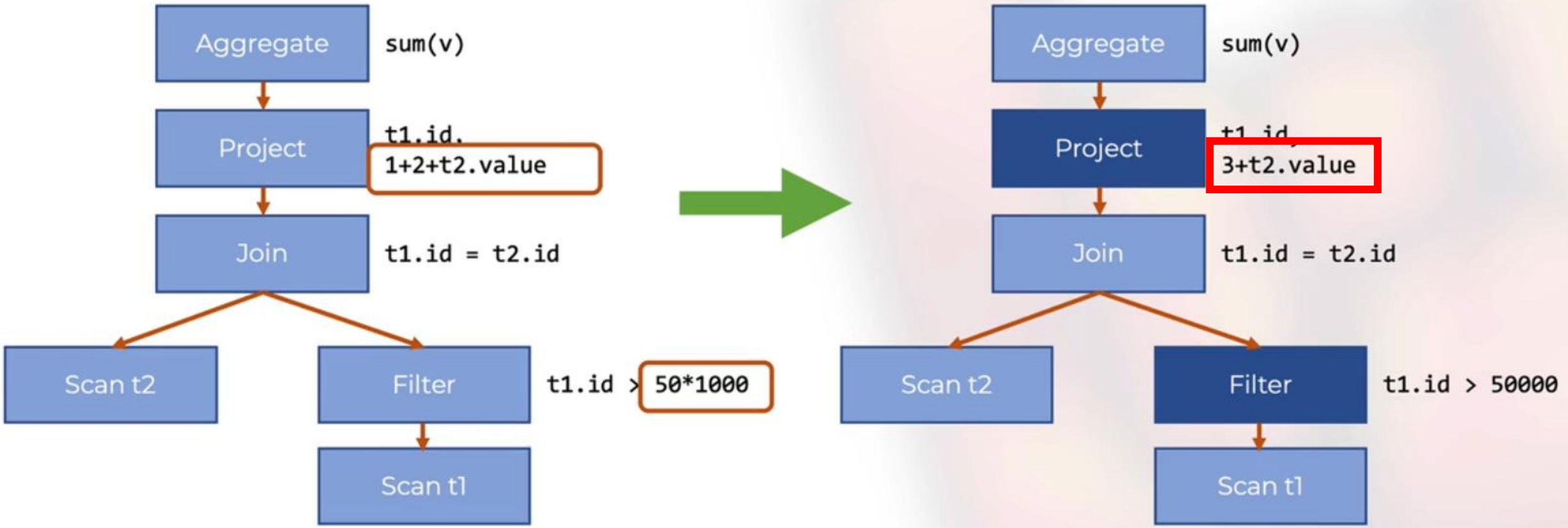
Optimization: filter pushdown





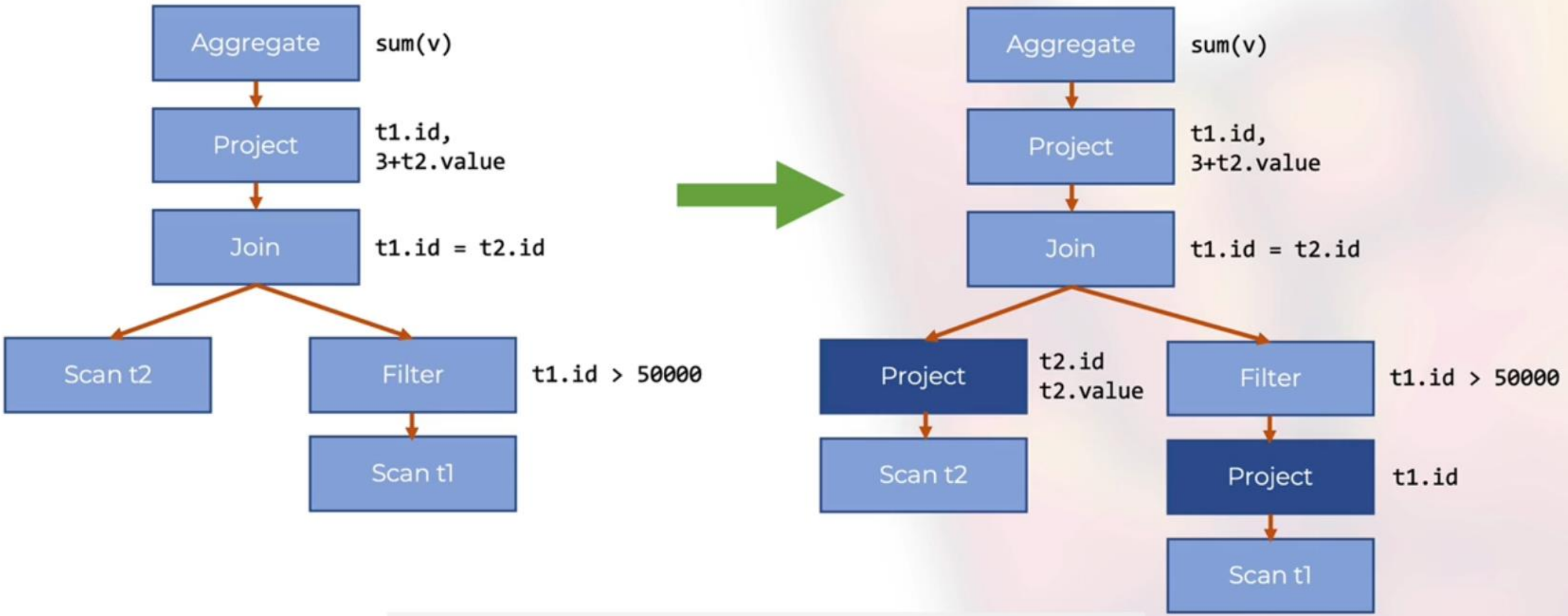
Catalyst

Optimization: combining rules



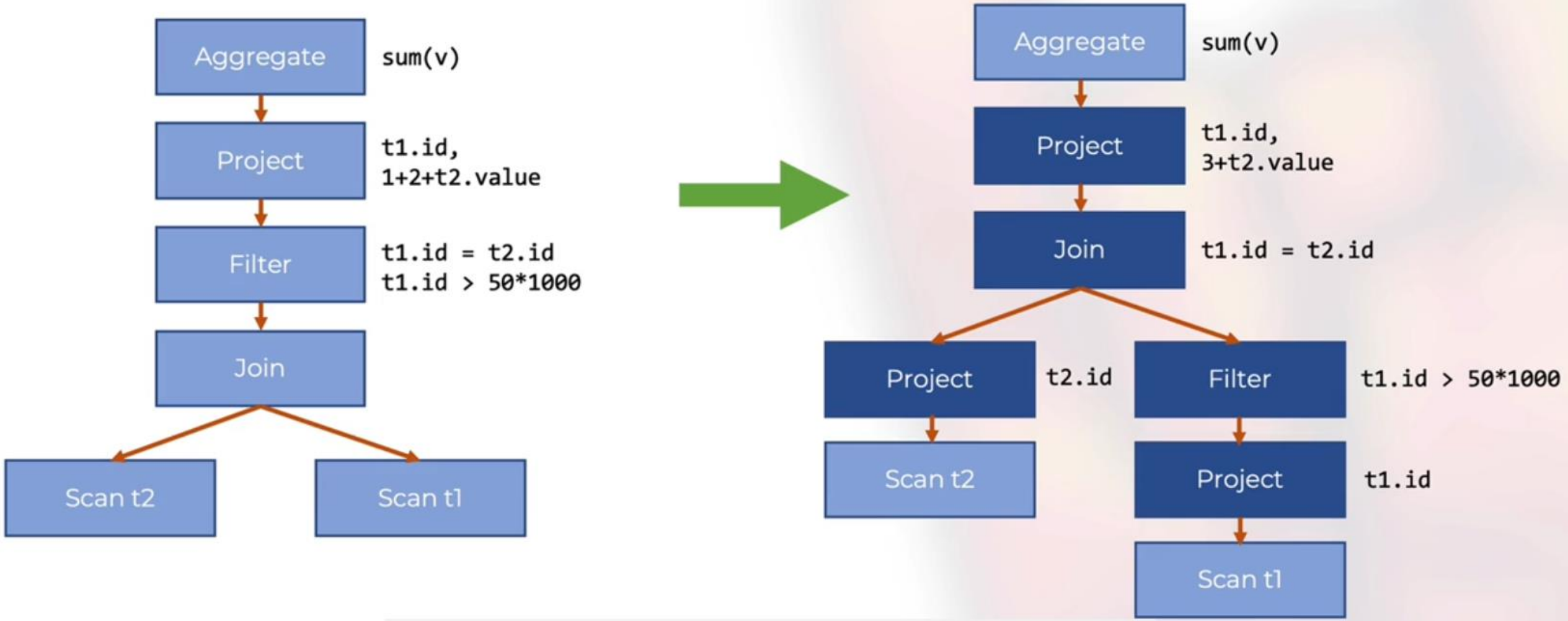
Catalyst

Optimization: column pruning



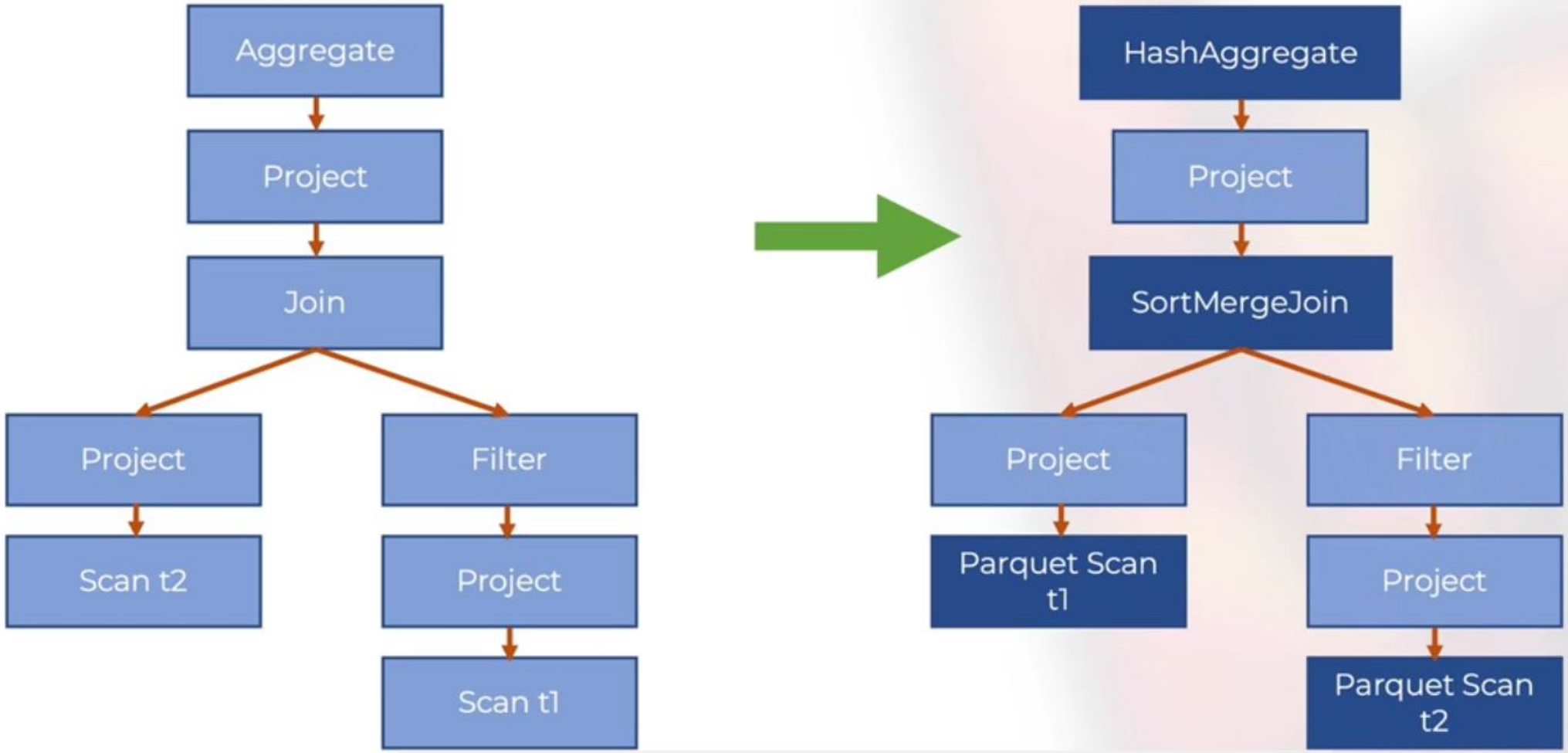
Catalyst

Combine all:



Catalyst

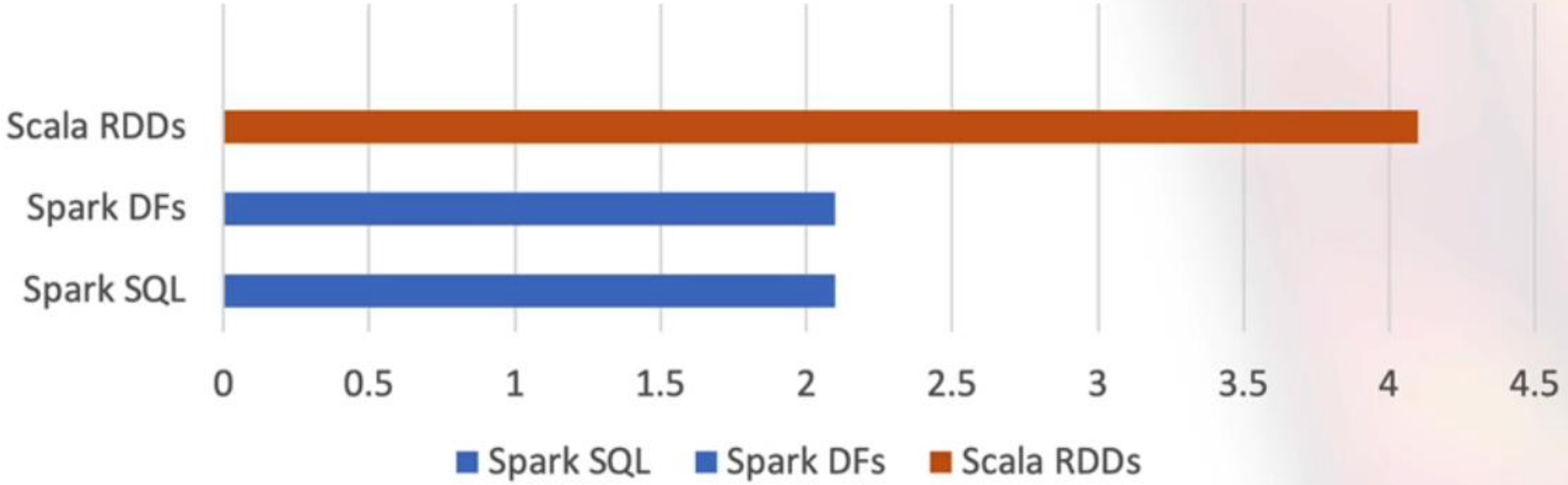
Physical plan: known implementation of all operations



Catalyst

Results

- extra structure (SQL) limits what can be expressed (vs RDDs)
- however, we can express most computations
- expressions are more concise
- structure allows for optimizations



What Catalyst can't do

- can't optimize lambdas