Weld: A Common Runtime for Data Analytics

Shoumik Palkar, James Thomas, Anil Shanbhag*, Deepak Narayanan,
Malte Schwarzkopf*, Holger Pirk*, Saman Amarasinghe*, Matei Zaharia

Stanford InfoLab, *MIT CSAIL

Motivation

Modern data apps combine many disjoint processing libraries & functions

» Relational, statistics, machine learning, …

» E.g. PyData stack

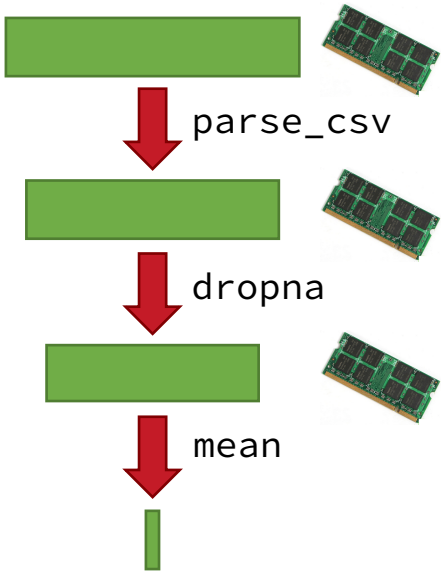
\+ Great results leveraging work of 1000s of authors

– No optimization across these functions

How Bad is This Problem?

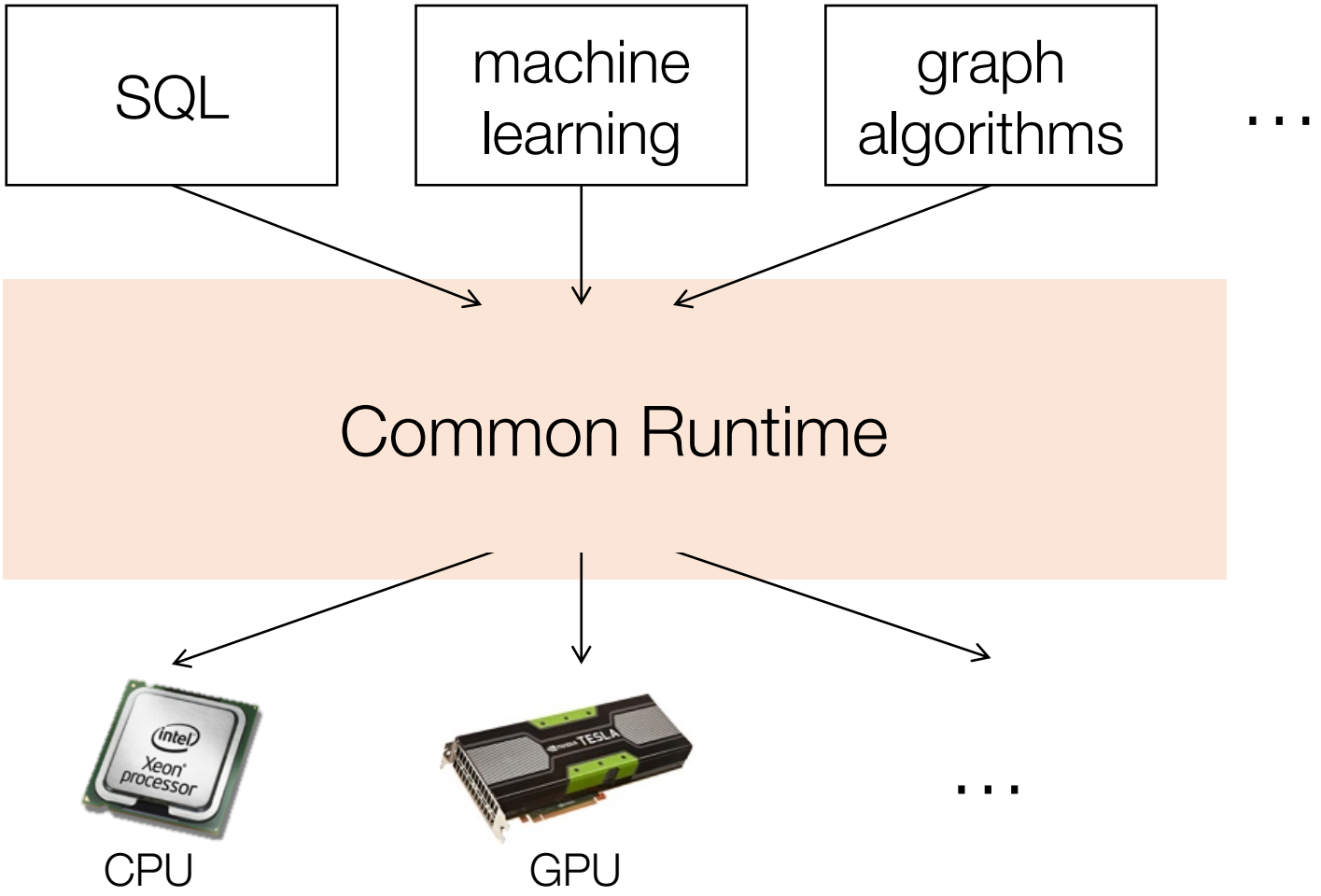
Growing gap between memory/processing makes traditional way of combining functions worse

```
data = pandas.parse_csv(string)

filtered = pandas.dropna(data)

avg = numpy.mean(filtered)
```

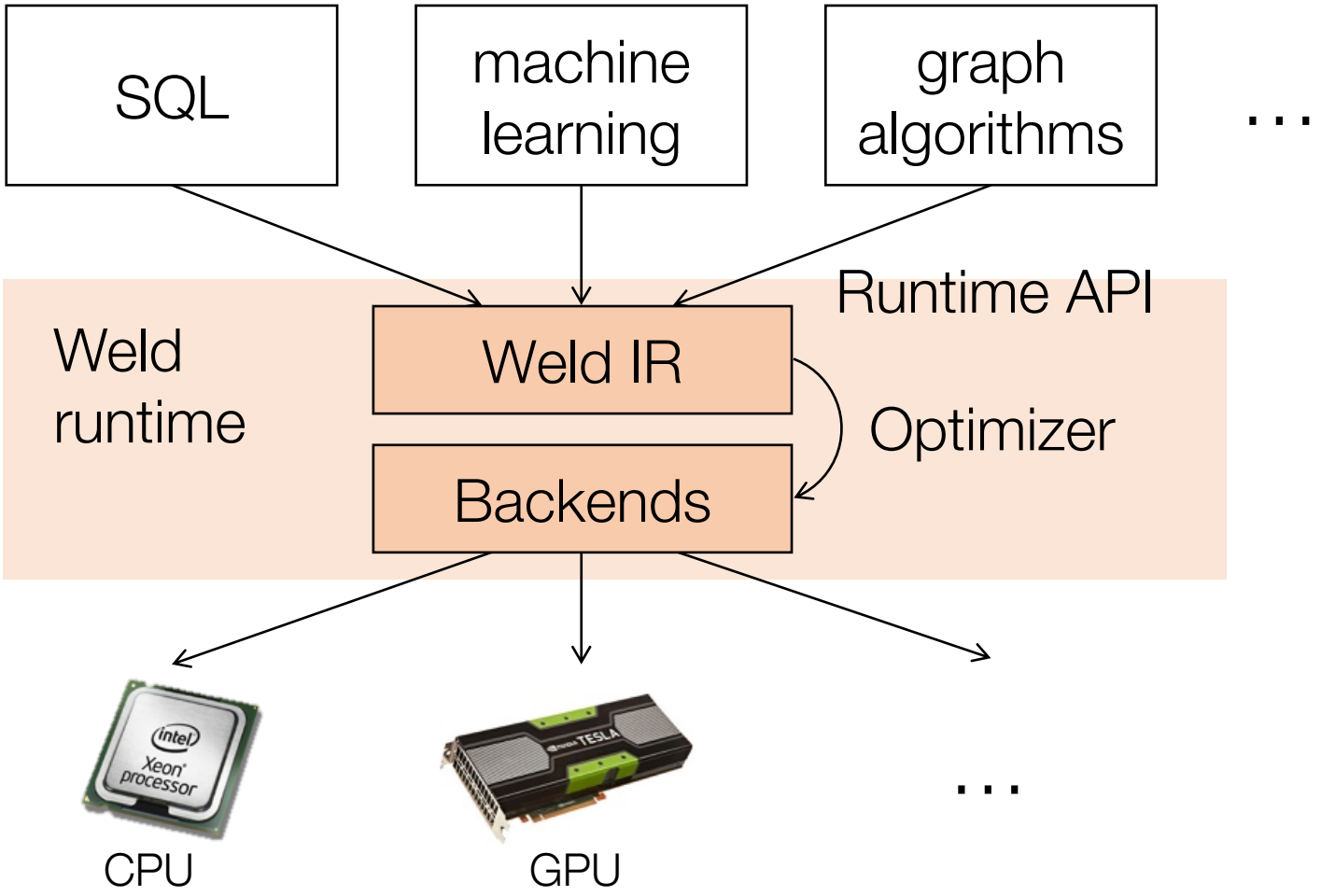


5-30x slowdowns in NumPy, Pandas, TensorFlow, etc

How We Solve This

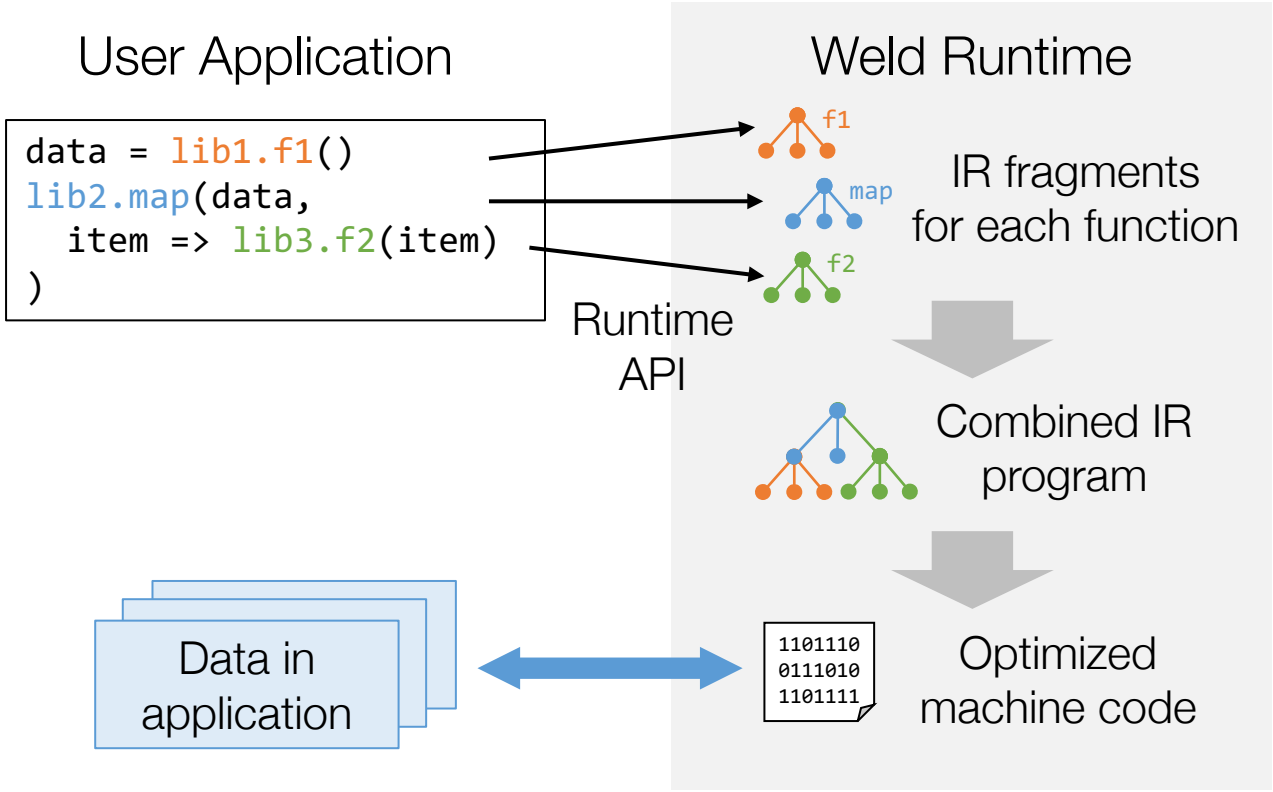


How We Solve This



Runtime API

Uses lazy evaluation to collect work across libraries



Weld IR

Designed to meet three goals:

1. **Library composition:** support complete workloads such as nested parallel calls

2. **Ability to express optimizations:** e.g. loop fusion, vectorization, loop tiling

3. **Explicit parallelism**

Weld IR

Small, powerful design inspired by “monad comprehensions”

Parallel loops: iterate over a dataset

Builders: declarative objects for producing results

» E.g. append items to a list, compute a sum

» Can be implemented differently on different hardware

Captures relational algebra, functional APIs like Spark, linear algebra, and composition thereof

Examples

Implement functional operators using builders

```
def map(data, f):
    builder = new vecbuilder[int]
    for x in data:
        merge(builder, f(x))
    result(builder)


def reduce(data, zero, func):
    builder = new merger[zero, func]
    for x in data:
        merge(builder, x)
    result(builder)
```

Example Optimization: Fusion

```
squares = map(data, x => x * x)
sum = reduce(data, 0, +)
```

```
bld1 = new vecbuilder[int]
bld2 = new merger[0, +]
for x in data:
    merge(bld1, x * x)
    merge(bld2, x)
```

Loops can be merged into one pass over data

Implementation

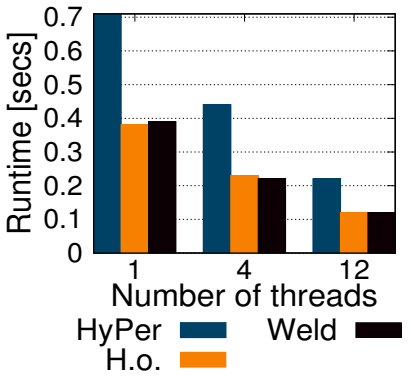
Prototype with APIs in Scala and Python

» LLVM and Voodoo for code gen

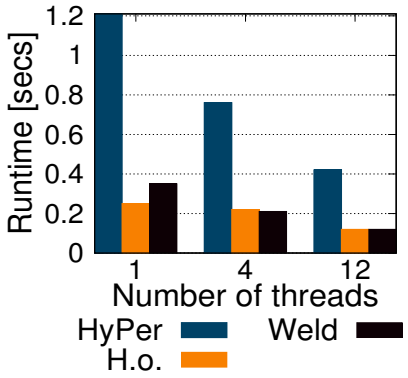
Integrations: TensorFlow, NumPy, Pandas, Spark

Results: Individual Workloads

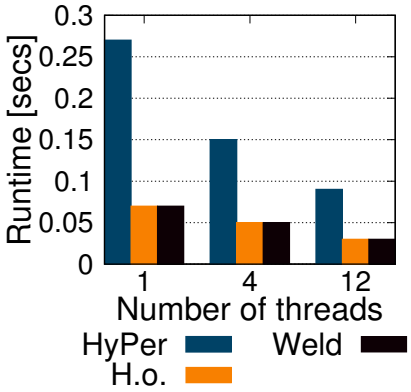
SQL (TPC-H)



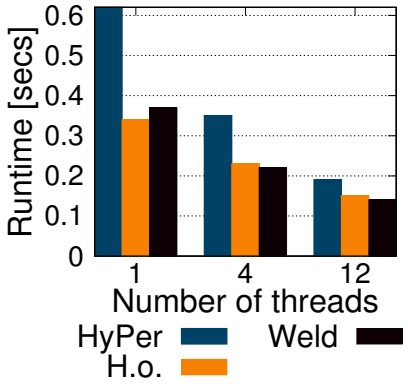
Q1



Q3

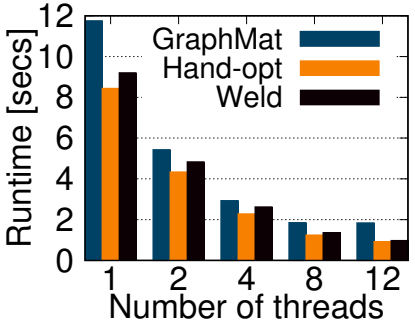


Q6

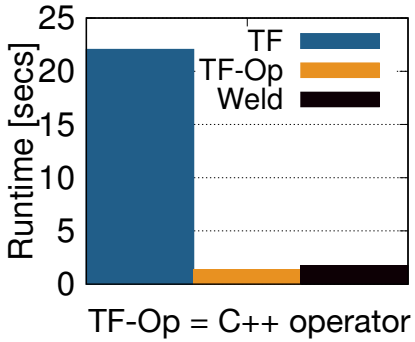


Q12

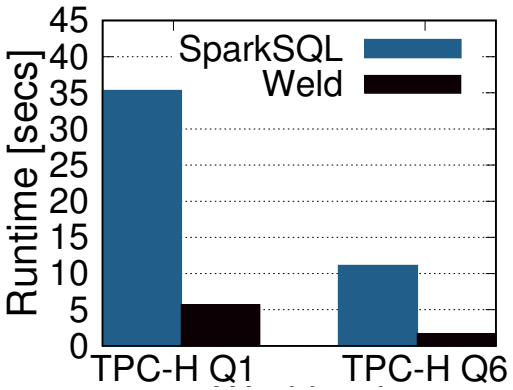
PageRank

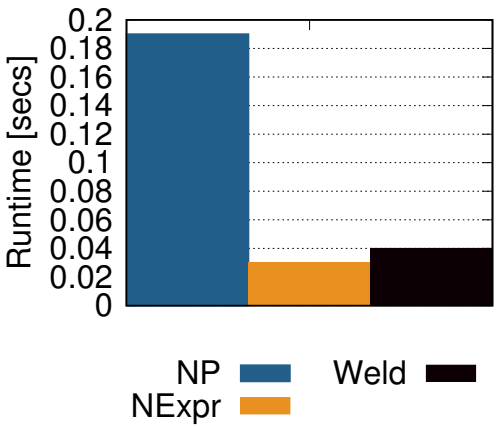


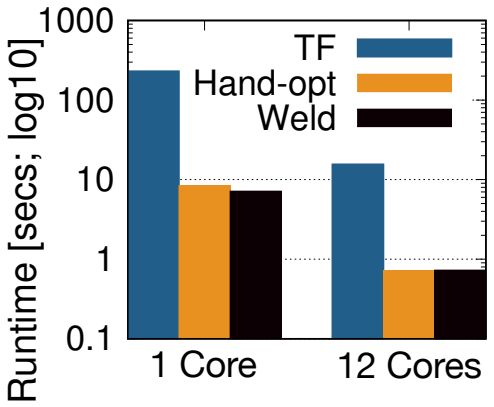
Word2Vec



Results: Existing Frameworks







Spark SQL

TPC-H

NumPy

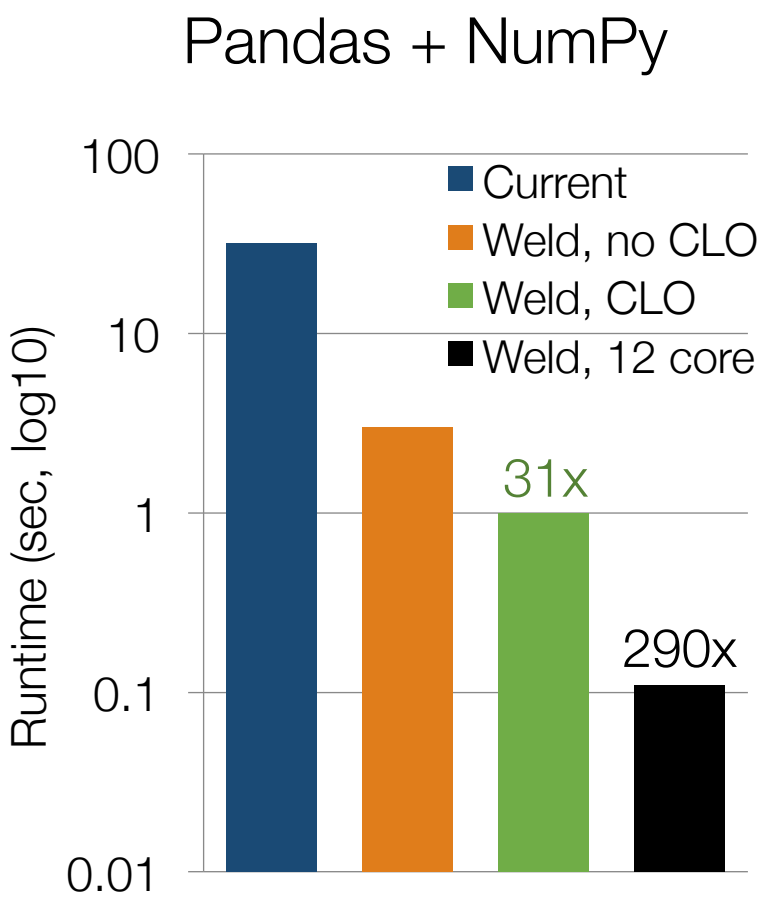
Vector Sum

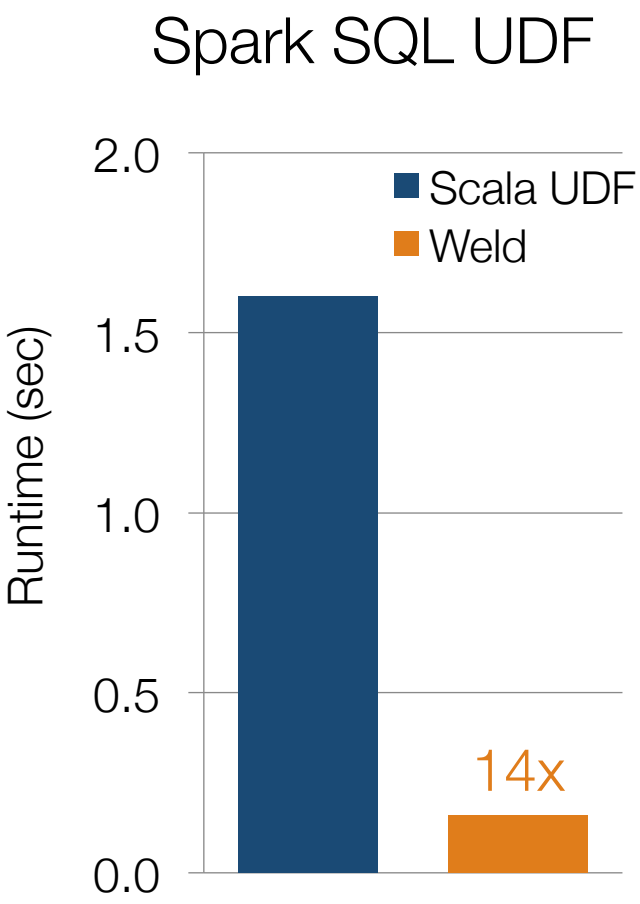
TensorFlow

Logistic Regression

Integration effort: 500 lines glue, 30 lines/operator

Results: Cross-Library Optimization





Conclusion

The way we compose software will have to change to efficiently use modern hardware

Weld is our first attempt at such a design – lots of open questions!

» Optimization, specialized hardware, domain info, …

Open source: this spring

We're hiring! (postdocs)