DAWN: Infrastructure for Usable Machine Learning

Peter Bailis, Kunle Olukotun, Chris Ré, Matei Zaharia

It's the Golden Age of Data*

Incredible advances in image recognition, natural language processing, planning, info retrieval

Society-scale impact: autonomous vehicles, personalized medicine, human trafficking

No end in sight for advances in ML

***for the best-funded, best-trained engineering teams**

Building ML Products is Too Hard

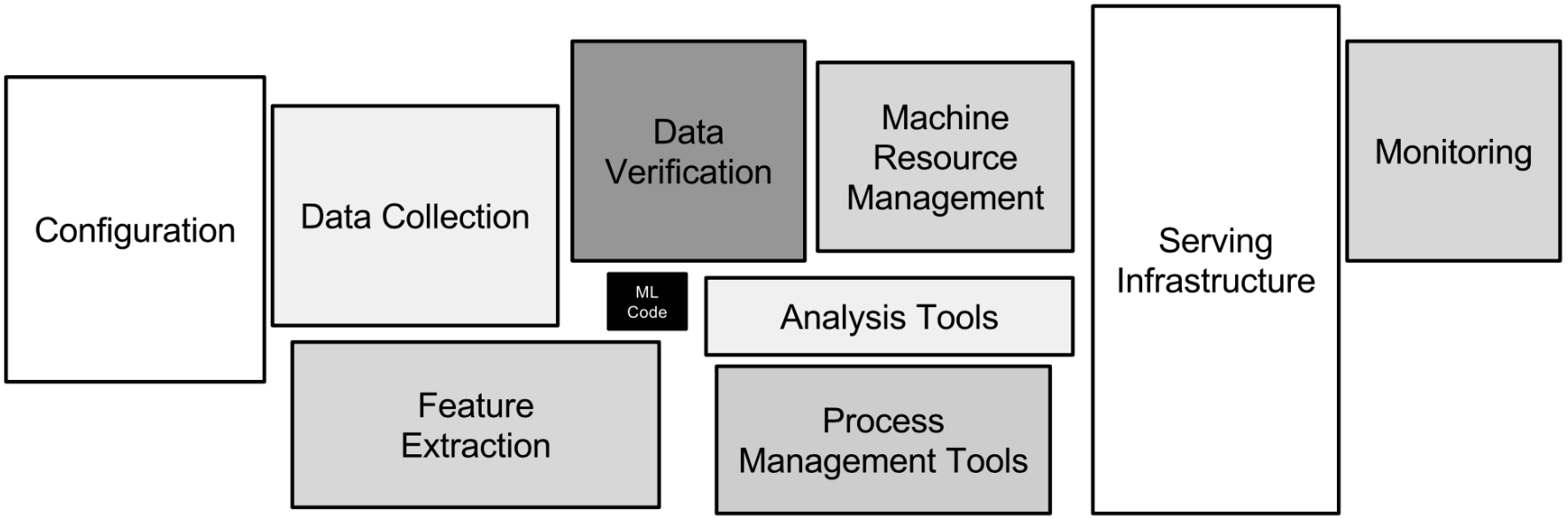
Major successes (e.g., AlphaGo, ImageNet) require hundreds to thousands of engineers

Huge effort in data preparation, model tuning, experimentation, and productionizing

Domain experts cannot easily or cheaply build ML products

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.



"Only a fraction of real-world ML systems is composed of ML code"

The DAWN Question

What if *anyone* with domain expertise could build their own production-quality ML products?

- Without a PhD in machine learning
- Without being an expert in systems
- Without understanding the latest hardware

It's happened before

It's happened before: Search

Before: Decades of research on information retrieval, indexes, ranking, etc

After: any developer can add search to an application by linking a library (e.g. Solr, Lucene); everyone (i.e., non-expert users) uses search

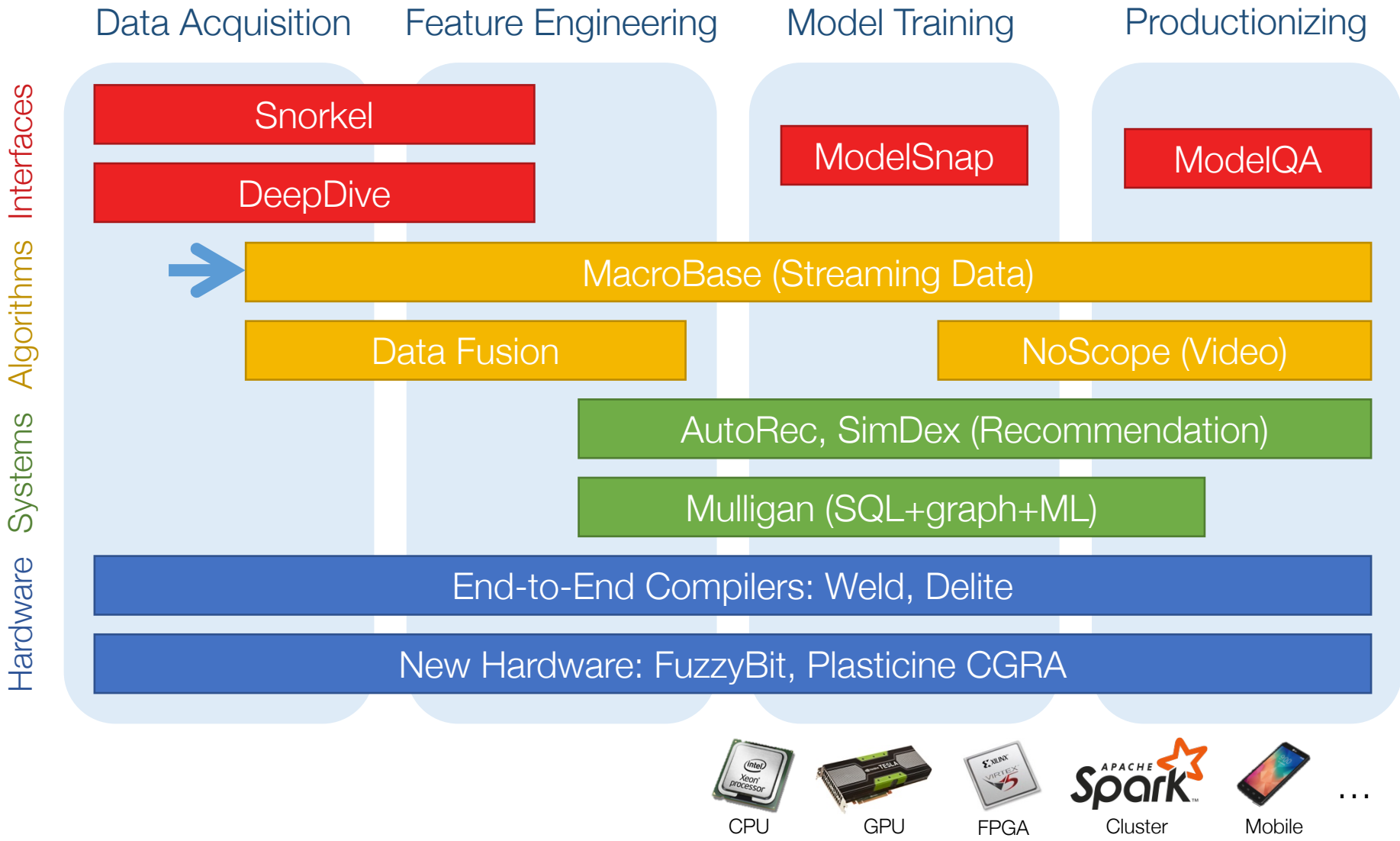
It's happened before: SQL

Before: raw access to disk, manual layout of records, network databases (CODASYL)

After: SQL forms basis for transactional engines, data warehousing, business intelligence tools

Key idea: end-to-end systems that tackle the barriers to access & production use

The DAWN Stack



Example: MacroBase for Continuous Analytics

End-to-end system to **prioritize user attention**

record_id	user_id	state	hw_make	hw_model	firmware_version	app_version	avg_temp	battery_drain	trip_time
131920	49e36c5b031141dd8cf240f7	CO	Lenovo	Lenovo_K910L	4.4.2	v21	79.252124	0.205834	40.910145
131921	a670eab2bc6d4e5991ea4269	WV	TCT (Alcatel)	4009A	7.1.1	v36	72.136380	0.184874	47.253076
131922	247c64e48a8743829c5f7199	UT	TCT (Alcatel)	4009A	7.1.1	v31	77.300103	0.230015	25.342140
131924	6bd9af7242ca480a96d75d0d	OH	HTC	HTC_M10u	6.0.1	v38	70.937014	0.454293	38.661611
131926	d449b12dcb6346d7af1021de	HI	HTC	HTC_Wildfire_S_A510b	6.0	v46	75.436764	0.151338	17.785555
131927	fff8907aa14e4a50ab76bd46	HI	bq	Aquaris_E4.5	4.4.1	v38	70.208187	0.286005	60.443799
131929	8226cd65bb1f4d61a6fcf455	MI	TCT (Alcatel)	ALCATEL_one_touch_97	6.0.1	v35	73.113370	0.249834	16.881133
131930	30e726fadc6744b2ace2d76b	LA	TCT (Alcatel)	ALCATEL_ONE_TOUCH_60	5.0	v40	77.918077	0.405417	51.163642
131931	569f35993da246f4afc83c2e	FL	Lava	S1	6.0.1	v44	76.558080	0.416760	42.252460
131932	9d2db241316c43378b8ec14c	AL	LGE	LG-D724	7.0	v29	76.760340	0.334446	37.922632
131933	484a1ced0a6a46468874861c	LA	Hisense	LED42K680X3DU	4.4.4	v49	77.138769	0.409485	23.345804
131934	d375d5a0e10d46cf9b91e343	MI	Techno	TECNO_P5S	6.0	v31	70.115019	0.179464	45.051123
131936	e4835a64d96e4e89997ce027	WI	ZTE	Z828	6.0.1	v35	71.615570	0.396389	47.662474
131937	cf00ae2105bb4e3cb43b64b2	FL	Spice	Spice_Mi-498H	5.0	v42	72.045184	0.327405	45.099422
131939	c94d264a846149f08f51c28e	RI	Infocus	InFocus_M320u	4.4.1	v49	73.543359	0.224504	19.069803
131940	c3c82947ab5a4d09b52afe21	MI							38.287879
131943	4e4566143b144be1809ad4d9	RI							28.306756
131944	0ee8ff83606b496392bedd49	NE							26.749918
131946	0ee8ff83606b496392bedd49	OK	LGE	LS670	4.0.4	v30	78.186519	0.381604	27.601968
131947	7a4bc4c56aa54d20b1119d42	NV	Oppo	F1f	4.3.1	v38	77.434095	0.436364	40.869723
131948	c3c82947ab5a4d09b52afe21	GA	Huawei	HUAWEI_Y320-U151	4.4.3	v42	77.715329	0.281726	23.077248
131949	2b0acf33a91f49b6aa70cbe5	MO	ZTE	KPN_Smart_300	4.4.4	v39	75.368614	0.371224	44.295975
131950	5aab0148ea794d2eacfc9a27	NV	Ketablet	TR10CS1	5.0	v49	79.459844	0.491424	37.653744

Too much data for manual inspection
Even harder when data is *streaming*

MACROBase

Database Configuration

Database URL: postgres submit

Base query: SELECT * from sensor_data; submit

Connected to postgres database!

Analyze

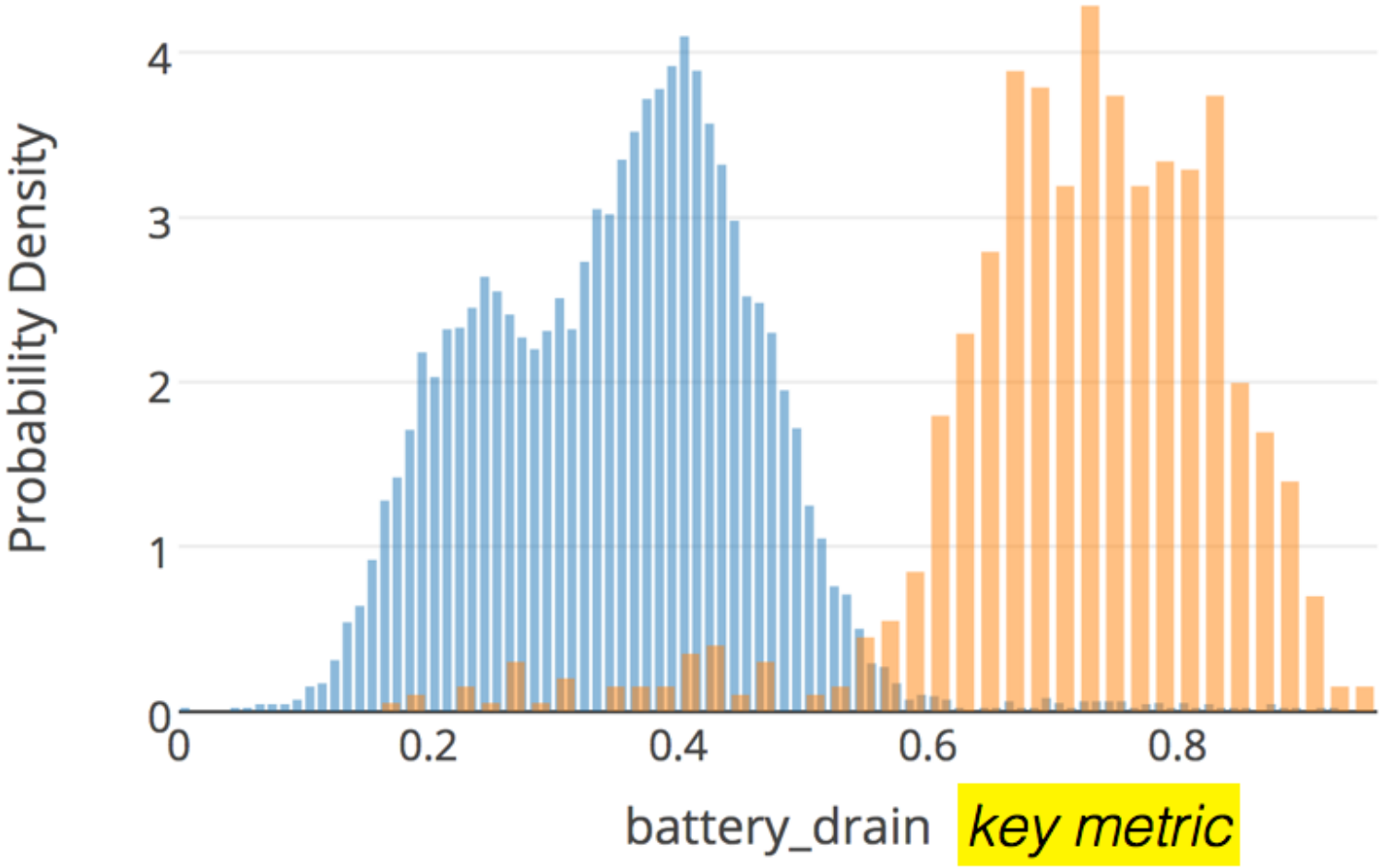
Schema Information and Selection sample reset clear

Clustering Attribute?	Target Metric? Lo/Hi	Name	Type
+	↓ ↑	reading_id	int8
+	↓ ↑	device_id	int8
+	↓ ↑	state	varchar
+	↓ ↑	model	varchar
+	↓ ↑	firmware_version	varchar
+	↓ ↑	temperature	numeric
+	↓ ↑	power_drain	numeric

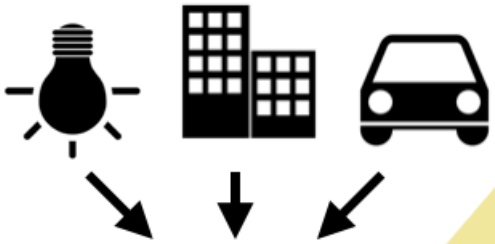
Column	Value
app_version	v50
hw_model	em_i8180
hw_make	Emdoor

correlated attributes

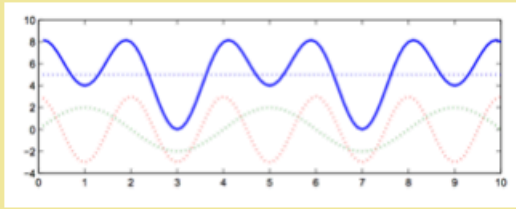
Support:	0.8226
Ratio Out/In:	472.92
Records:	849



MacroBase Query Architecture

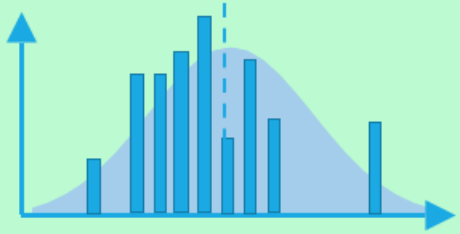


FEATURE TRANSFORM



extract domain-specific signals

OUTLIER DETECTION



identify data in tails

RESULT EXPLANATION

find disproportionately correlated attributes

Outliers	Inliers
{iPhone6, Canada}	{iPhone6, USA}
{iPhone6, USA}	{iPhone6, USA}
{iPhone5, Canada}	{iPhone5, USA}

ALERTS, REPORTS

MacroBase Summary

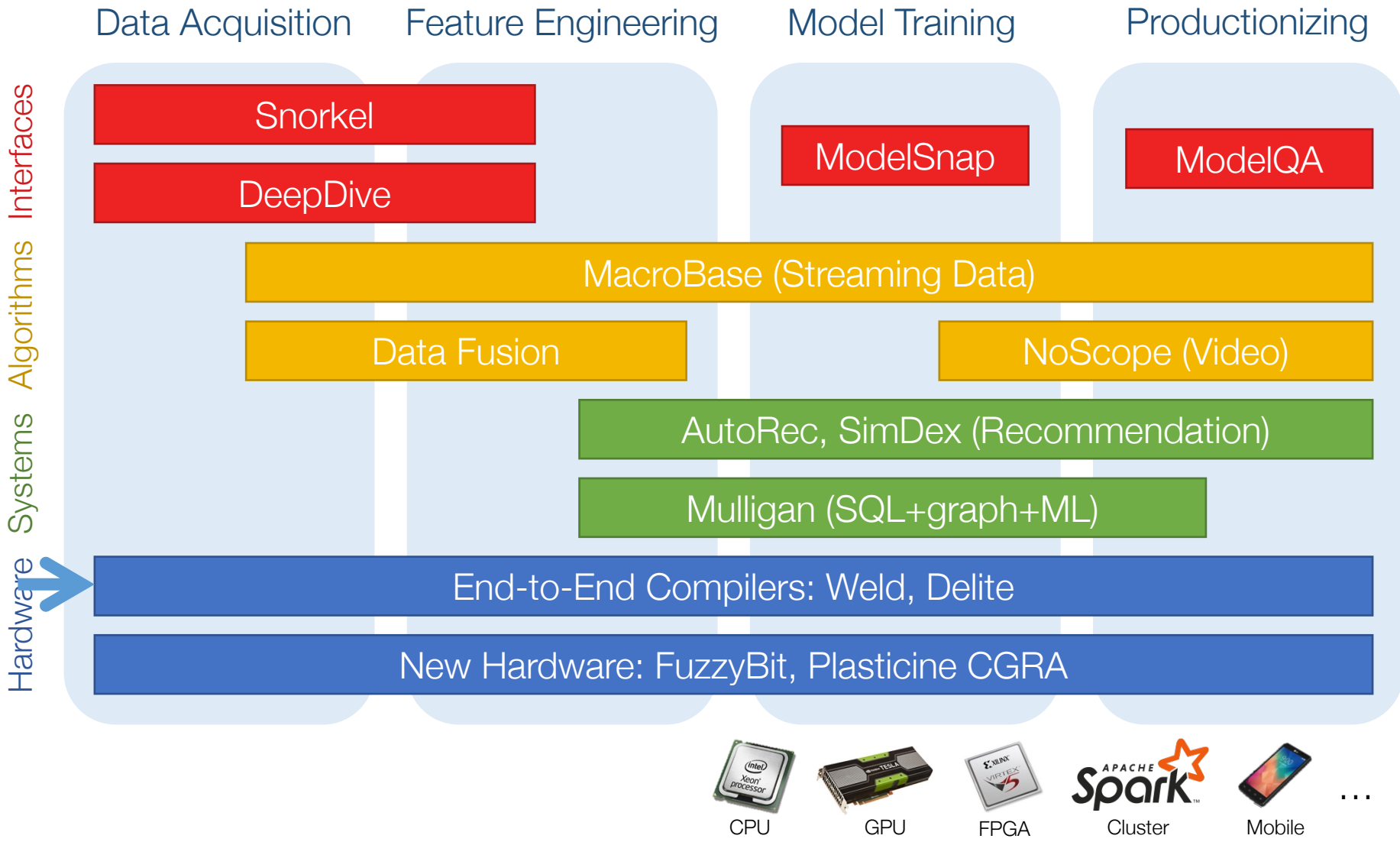
End-to-end system to prioritize user attention

- No ML expertise needed: MacroBase uses general models and tunes them automatically
- No separate step for production use
- Co-design from algorithms to HW

Early users: automotive, cloud, mobile apps, manufacturing

Open source: github.com/stanford-futuredata/macrobase

The DAWN Stack

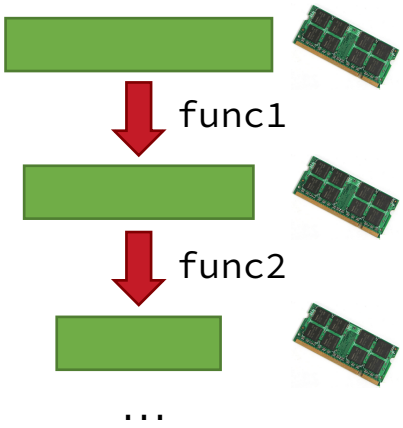


Weld: Rethinking the Interface to Data Analytics Libraries

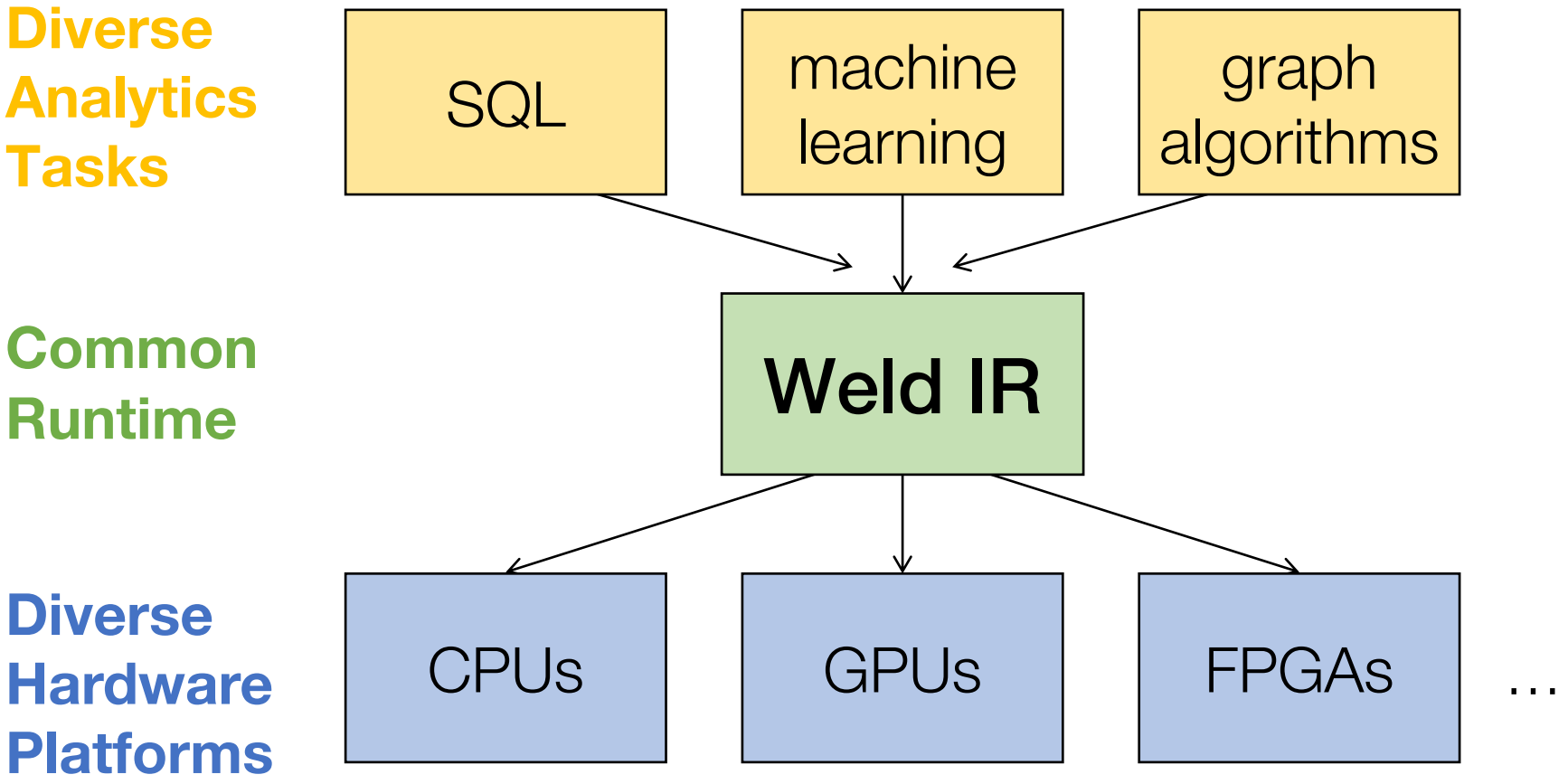
Standard approach: users combine libraries using function calls that pass data via memory

Problem: for data-intensive apps, data movement cost dominates on modern hardware!

5-30x slowdowns in NumPy, Spark, TensorFlow, …

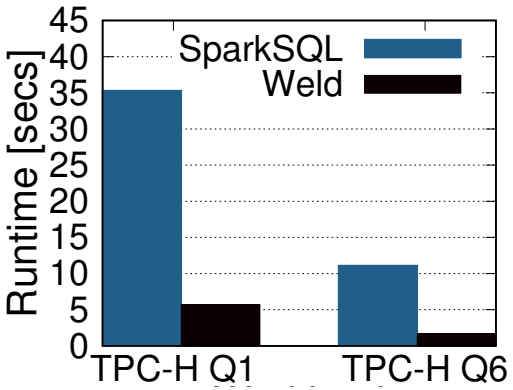


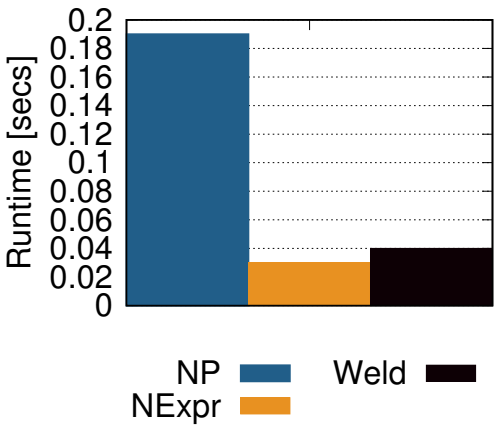
Weld's Approach

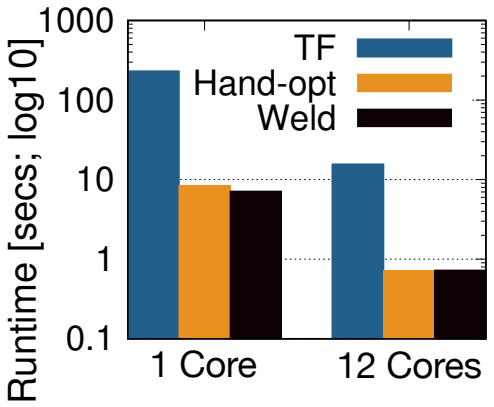


Open source: weld.stanford.edu

Results: Existing Frameworks







Spark SQL

TPC-H

NumPy

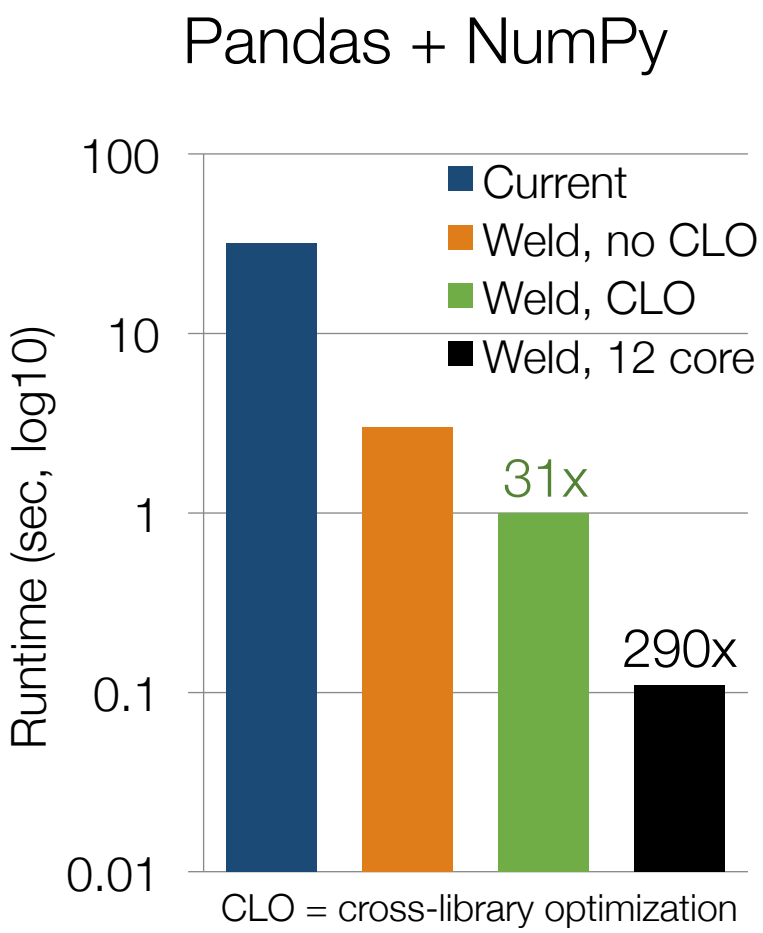
Vector Sum

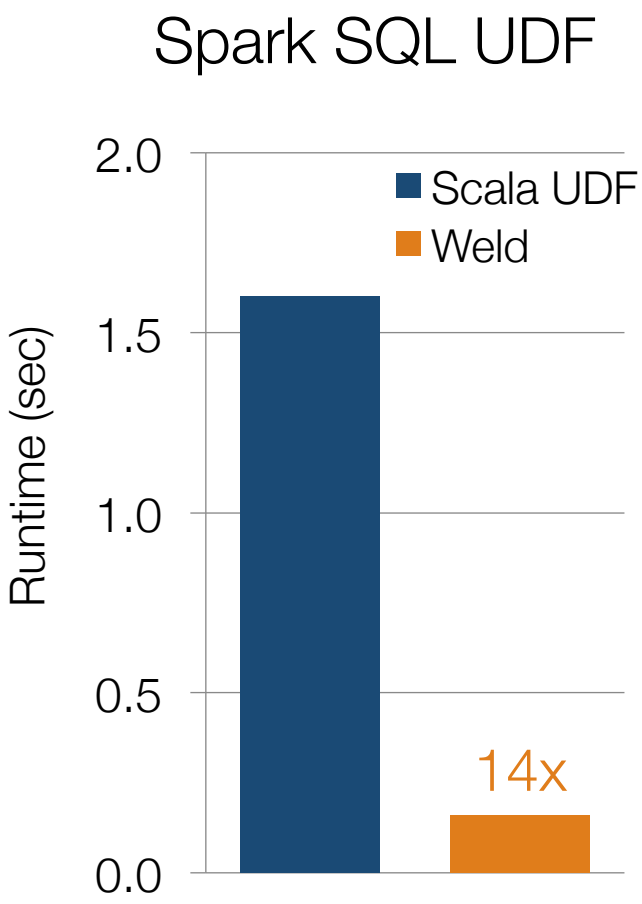
TensorFlow

Logistic Regression

Integration effort: 500 lines glue, 30 lines/operator

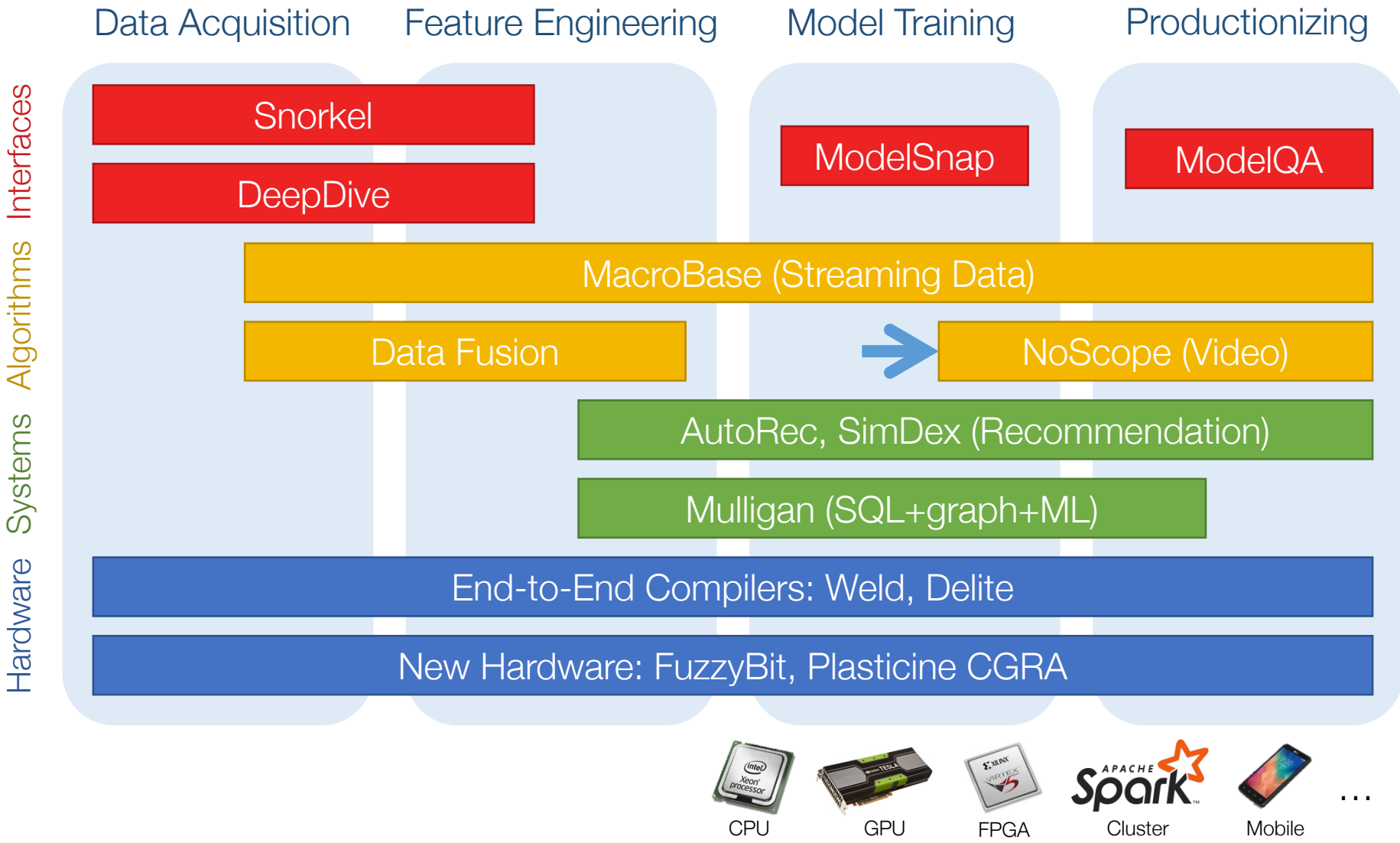
Results: Cross-Library Optimization





Open source: weld.stanford.edu

The DAWN Stack



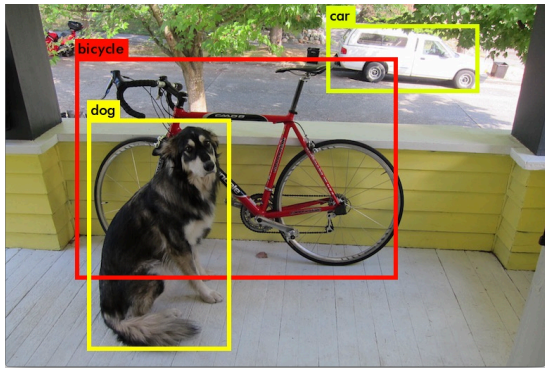
NoScope: Fast CNN-Based Video Queries

Opportunity: CNNs allow more accurate queries on visual data than ever

Challenge : processing 1 video in real time requires a $1000 GPU

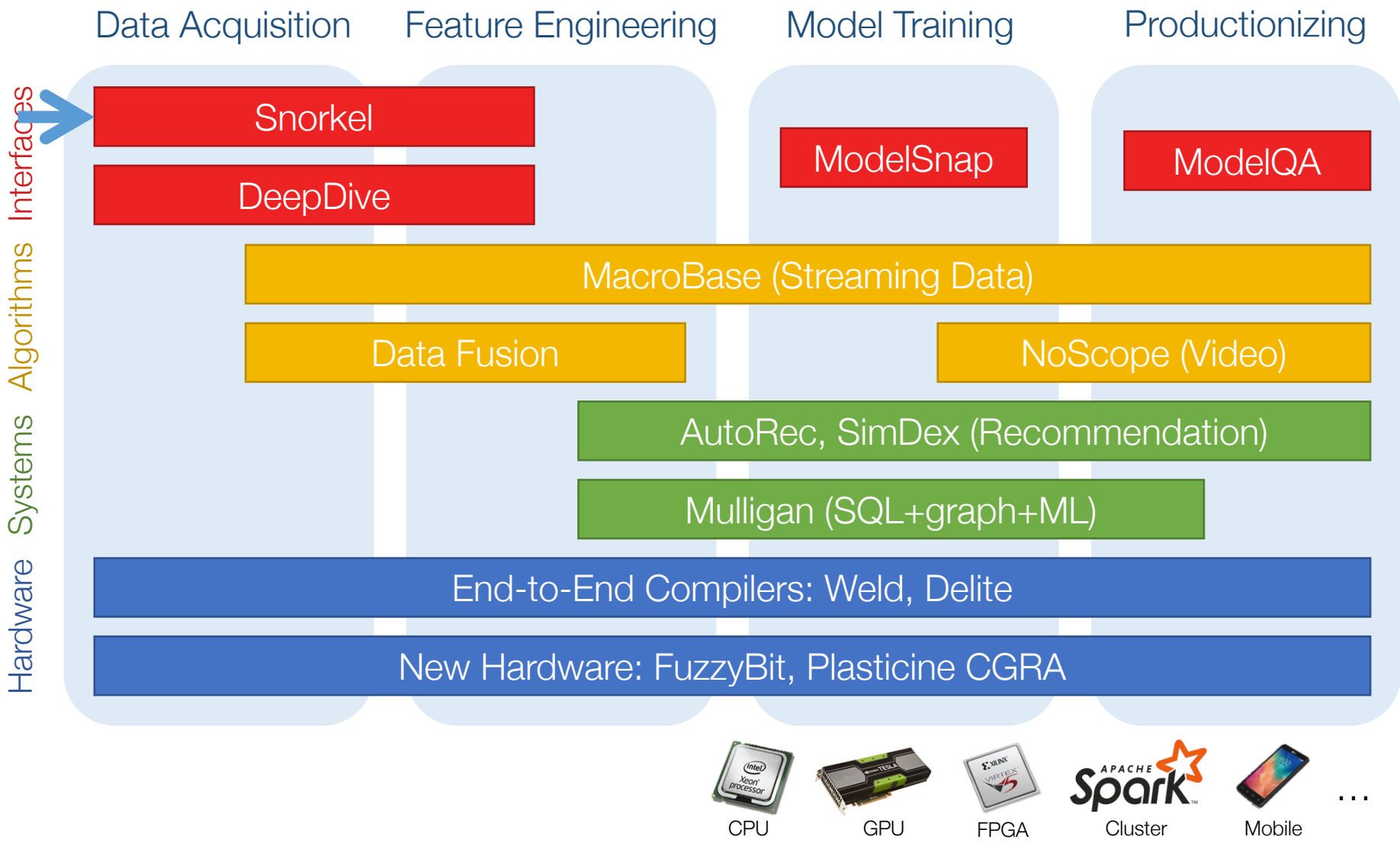
Result: same accuracy but 100-3000x faster through:

- Scene-specific distillation
- Temporal + spatial locality



bit.ly/NoScopeArxiv

The DAWN Stack



Training data is key enabler, barrier to entry

The *New* New Oil

How can we leverage data that's expensive to label at scale?

Snorkel's Approach: Weak Supervision

1) User writes *labeling functions*: short programs that may not always give right label
   - E.g. regex to search in text

2) Snorkel simultaneously learns *noise* in LFs and a *noise-aware* target model (e.g. LSTM)

System	NCBI Disease (F1)	CDR Disease (F1)	CDR Chem. (F1)
TaggerOne (Dogan, 2012)*	**81.5**	79.6	**88.4**
Snorkel: Logistic Regression	79.1	79.6	**88.4**
Snorkel: LSTM + Embeddings	79.2	**80.4**	88.2

github.com/HazyResearch/snorkel

DAWN: machine learning for everyone via novel techniques and interfaces that span hardware, systems, and algorithms

Find out more at dawn.cs.stanford.edu


Peter Bailis


Chris Ré


Kunle Olukotun


Matei Zaharia

STANFORD INFOLAB