Machine Learning at the Limit

John Canny*^  
* Computer Science Division  
University of California, Berkeley  
^ Yahoo Research Labs  
@GTC, March, 2015
My Other Job(s)

Yahoo [Chen, Pavlov, Canny, KDD 2009]*

Ebay [Chen, Canny, SIGIR 2011]**

Quantcast 2011-2013

Microsoft 2014

Yahoo 2015

* Best application paper prize
** Best paper honorable mention
Data Scientist’s Workflow

- Sandbox
- Digging Around in Data
- Hypothesize
- Model
- Customize
- Evaluate
- Interpret
- Production
- Large Scale Exploitation

[Diagram showing a workflow with arrows and data visualization]
Data Scientist’s Workflow

Sandbox

Digging Around in Data

Hypothesize

Model

Customize

Evaluate

Interpret

Production

Large Scale Exploitation

\[
\begin{bmatrix}
\cos \theta & \sin \theta \\
-\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2
\end{bmatrix}
= \begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix}
\]
Why Build a New ML Toolkit?

• **Performance**: GPU performance pulling away from other platforms for *sparse* and dense data. Minibatch + SGD methods dominant on Big Data,…

• **Customizability**: Great value in customizing models (loss functions, constraints,…)

• **Explore/Deploy**: Explore fast, run the same code in prototype and production. Be able to run on clusters.
Desiderata

**Performance:**
- Roofline Design (single machine and cluster)
- General Matrix Library with full CPU/GPU acceleration

**Customizability:**
- Modular Learner Architecture (reusable components)
- Likelihood “Mixins”

**Explore/Deploy:**
- Interactive, Scriptable, Graphical
- JVM based (Scala) w/ optimal cluster primitives
Roofline design establishes fundamental performance limits for a computational kernel.

Roofline Design (Williams, Waterman, Patterson, 2009)
A Tale of Two Architectures

Intel® CPU

Memory Controller

ALU  ALU  ALU  ALU
Core  Core  Core  Core

L3 Cache

NVIDIA® GPU

ALU  ALU  ALU  ALU  ALU  ALU  ALU

L2 Cache
**CPU vs GPU Memory Hierarchy**

**Intel® 8 core Sandy Bridge CPU**
- 4kB registers: 5 TB/s
- 512K L1 Cache: 1 TB/s
- 2 MB L2 Cache
- 8 MB L3 Cache: 500 GB/s
- 10s GB Main Memory: 20 GB/s

**NVIDIA® GK110 GPU**
- 4 MB register file (!): 40 TB/s
- 1 MB Constant Mem: 13 TB/s
- 1 MB Shared Mem: 1 TB/s
- 1.5 MB L2 Cache: 500 GB/s
- 4 GB Main Memory: 200 GB/s
Natural language parsing with a state-of-the-art grammar (1100 symbols, 1.7 million rules, 0.1% dense)

End-to-End Throughput (4 GPUs):
2-2.4 Teraflops (1-1.2 B rules/sec), 1000 sentences/sec.

This is more than $10^5$ speedup for unpruned grammar evaluation (and it’s the fastest constituency parser).

How: Compiled grammar into instructions, blocked groups of rules into a hierarchical 3D grid, fed many sentences in a queue, auto-tuned. Max’ed every resource on the device.
Roofline Design – Matrix kernels

- Dense matrix multiply
- Sparse matrix multiply

Operational Intensity (flops/byte)

Throughput (gflops)

GPU ALU throughput

GPU registers

GPU shared memory

GPU main memory

CPU main memory

CPU ALU throughput
A Rooflined Machine Learning Toolkit

Zhao+Canny
SIAM DM 13, KDD 13, BIGLearn 13

DataSource
(Memory)

DataSource
(JBOD disks)

DataSource
HDFS over network

CPU host code

Learner
data blocks

Model
Optimizer
Mixins

Model
Optimizer
Mixins

GPU 1 thread 1
30 Gflops to 2 Teraflops per GPU

GPU 2 thread 2

Compressed disk streaming at
~ 0.1-2 GB/s ≈ 100 HDFS nodes
Matrix + Machine Learning Layers

Written in the beautiful Scala language:

• Interpreter with JIT, scriptable.
• Open syntax +,-,* , °,●,⊗ etc, math looks like math.
• Java VM + Java codebase – runs on Hadoop, Yarn, Spark.
• Hardware acceleration in C/C++ native code (CPU/GPU).
• Easy parallelism: Actors, parallel collections.
• Memory management (sort of 😊).
• Pre-built for multiple Platforms (Windows, MacOS, Linux).

Experience similar to Matlab, R, SciPy
Benchmarks

Recent benchmarks on some representative tasks:

- **Text Classification** on Reuters news data (0.5 GB)
- **Click prediction** on the Kaggle Criteo dataset (12 GB)
- **Clustering** of handwritten digit images (MNIST) (25 GB)
- **Collaborative filtering** on the Netflix prize dataset (4 GB)
- **Topic modeling (LDA)** on a NY times collection (0.5 GB)
- **Random Forests** on a UCI Year Prediction dataset (0.2 GB)
- **Pagerank** on two social network graphs at 12GB and 48GB
Benchmarks

**Systems (single node)**
- BIDMach
- VW (Vowpal Wabbit) from Yahoo/Microsoft
- Scikit-Learn
- LibLinear

**Cluster Systems**
- Spark v1.1 and v1.2
- Graphlab (academic version)
- Yahoo’s LDA cluster
Benchmarks: Single-Machine Systems

RCV1: Text Classification, 103 topics (0.5GB). Algorithms were tuned to achieve similar accuracy.

<table>
<thead>
<tr>
<th>System</th>
<th>Algorithm</th>
<th>Dataset</th>
<th>Dim</th>
<th>Time (s)</th>
<th>Cost ($)</th>
<th>Energy (KJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDMach</td>
<td>Logistic Reg.</td>
<td>RCV1</td>
<td>103</td>
<td>14</td>
<td>0.002</td>
<td>3</td>
</tr>
<tr>
<td>Vowpal Wabbit</td>
<td>Logistic Reg.</td>
<td>RCV1</td>
<td>103</td>
<td>130</td>
<td>0.02</td>
<td>30</td>
</tr>
<tr>
<td>LibLinear</td>
<td>Logistic Reg.</td>
<td>RCV1</td>
<td>103</td>
<td>250</td>
<td>0.04</td>
<td>60</td>
</tr>
<tr>
<td>Scikit-Learn</td>
<td>Logistic Reg.</td>
<td>RCV1</td>
<td>103</td>
<td>576</td>
<td>0.08</td>
<td>120</td>
</tr>
</tbody>
</table>
Spark-XX = System with XX cores
BIDMach ran on one node with GTX-680 GPU

<table>
<thead>
<tr>
<th>System A/B</th>
<th>Algorithm</th>
<th>Dataset</th>
<th>Dim</th>
<th>Time (s)</th>
<th>Cost ($)</th>
<th>Energy (KJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark-72 BIDMach</td>
<td>Logistic Reg.</td>
<td>RCV1</td>
<td>1</td>
<td>30</td>
<td>0.07</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>103</td>
<td>14</td>
<td>0.002</td>
<td>3</td>
</tr>
<tr>
<td>Spark-64 BIDMach</td>
<td>RandomForest</td>
<td>YearPred</td>
<td>1</td>
<td>280</td>
<td>0.48</td>
<td>480</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>320</td>
<td>0.05</td>
<td>60</td>
</tr>
<tr>
<td>Spark-128 BIDMach</td>
<td>Logistic Reg.</td>
<td>Criteo</td>
<td>1</td>
<td>400</td>
<td>1.40</td>
<td>2500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81</td>
<td>0.01</td>
<td>16</td>
</tr>
</tbody>
</table>
### Benchmarks: Cluster Systems

Spark-XX or GraphLab-XX = System with XX cores
Yahoo-1000 had 1000 *nodes*

<table>
<thead>
<tr>
<th>System A/B</th>
<th>Algorithm</th>
<th>Dataset</th>
<th>Dim</th>
<th>Time (s)</th>
<th>Cost ($)</th>
<th>Energy (KJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark-384</td>
<td>K-Means</td>
<td>MNIST</td>
<td>4096</td>
<td>1100</td>
<td>9.00</td>
<td>22k</td>
</tr>
<tr>
<td>BIDMach</td>
<td></td>
<td></td>
<td></td>
<td>735</td>
<td>0.12</td>
<td>140</td>
</tr>
<tr>
<td>GraphLab-576</td>
<td>Matrix Factorization</td>
<td>Netflix</td>
<td>100</td>
<td>376</td>
<td>16</td>
<td>10k</td>
</tr>
<tr>
<td>BIDMach</td>
<td></td>
<td></td>
<td></td>
<td>90</td>
<td>0.015</td>
<td>20</td>
</tr>
<tr>
<td>Yahoo-1000</td>
<td>LDA (Gibbs)</td>
<td>NYtimes</td>
<td>1024</td>
<td>220k</td>
<td>40k</td>
<td>4E10</td>
</tr>
<tr>
<td>BIDMach</td>
<td></td>
<td></td>
<td></td>
<td>300k</td>
<td>60</td>
<td>6E7</td>
</tr>
</tbody>
</table>
BIDMach at Scale

Latent Dirichlet Allocation

BIDMach outperforms cluster systems on this problem, and has run up to 10 TB on one node.
Benchmark Summary

• BIDMach on a PC with NVIDIA GPU is at least 10x faster than other single-machine systems for comparable accuracy.

• For Random Forests or single-class regression, BIDMach on a GPU node is comparable with 8-16 worker clusters.

• For multi-class regression, factor models, clustering etc., GPU-assisted BIDMach is comparable to 100-1000-worker clusters. Larger problems correlate with larger values in this range.
• **Multilabel regression** problem (summer intern project):
  • Existing tool (single-machine) took ~ 1 week to build a model.
  • BIDMach on a GPU node takes 1 hour *(120x speedup)*
  • Iteration and feature engineering gave +15% accuracy.

• **Auction simulation** problem (cluster job):
  • Existing tool simulates auction variations on log data.
  • On NVIDIA 3.0 devices (64 registers/thread) we achieve a *70x speedup* over a reference implementation in Scala
  • On NVIDIA 3.5 devices (256 registers/thread) we can move auction state entirely into register storage and gain a *400x speedup*. 
Classification (cluster job):
- Cluster job (logistic regression) took 8 hours.
- BIDMach version takes < 1 hour on a single node.

SVMs for image classification (single machine)
- Large multi-label classification took 1 week with LibSVM.
- BIDMach version (SGD-based SVM) took 90 seconds.
Performance Revisited

- BIDMach had a **10x-1000x cost advantage** over the other systems. The ratio was higher for larger-scale problems.
- Energy savings were similar to the cost savings, at **10x-1000x**.

But why??

- We only expect about 10x from GPU acceleration?
- See our Parallel Forall post:

BIDMach ML Algorithms

1. Regression (logistic, linear)
2. Support Vector Machines
3. k-Means Clustering
5. Collaborative Filtering
6. NMF – Non-Negative Matrix Factorization
7. Factorization Machines
8. Random Forests
9. Multi-layer neural networks
10. IPTW (Causal Estimation)
11. ICA

● = Likely the fastest implementation available
SAME sampling accelerates standard Gibbs samplers with discrete + continuous data.

Our first instantiation gave a 100x speedup for a very widely-studied problem (Latent Dirichlet Allocation), and was more accurate than any other LDA method we tested:

SAME sampling is a general approach that should be competitive with custom symbolic methods.

Arxiv paper on BIDMach website.
Research: Rooflined cluster computing

Kylix (ICPP 2014)

• Near optimal model aggregation for sparse problems.
• Communication volume across layers has a characteristic Kylix shape:
Software (version 1.0 just released)

Code: github.com/BIDData/BIDMach

Wiki: http://bid2.berkeley.edu/bid-data-project/overview/
BSD open source libs and dependencies, papers

In this release:
• Random Forests, ICA
• Double-precision GPU matrices
• Ipython/IScala Notebook
• Simple DNNs

Wrapper for Berkeley’s Caffe coming soon…
Thanks

Sponsors:

Collaborators: