Map-Reduce for Machine Learning on Multicore

Original Paper at NIPS 2006 by –

C. Chu - chengtao@stanford.edu
S.K. Kim - skkim38@stanford.edu
Y. Lin - ianl@stanford.edu
Y.Y. Yu - yuanyuan@stanford.edu
G. Bradski - garybradski@gmail.com
A.Y. Ng - ang@cs.stanford.edu
K. Olukotun - kunle@cs.stanford.edu

Presented for CSE740 Requirement by - Tarun Gauba

Motivation

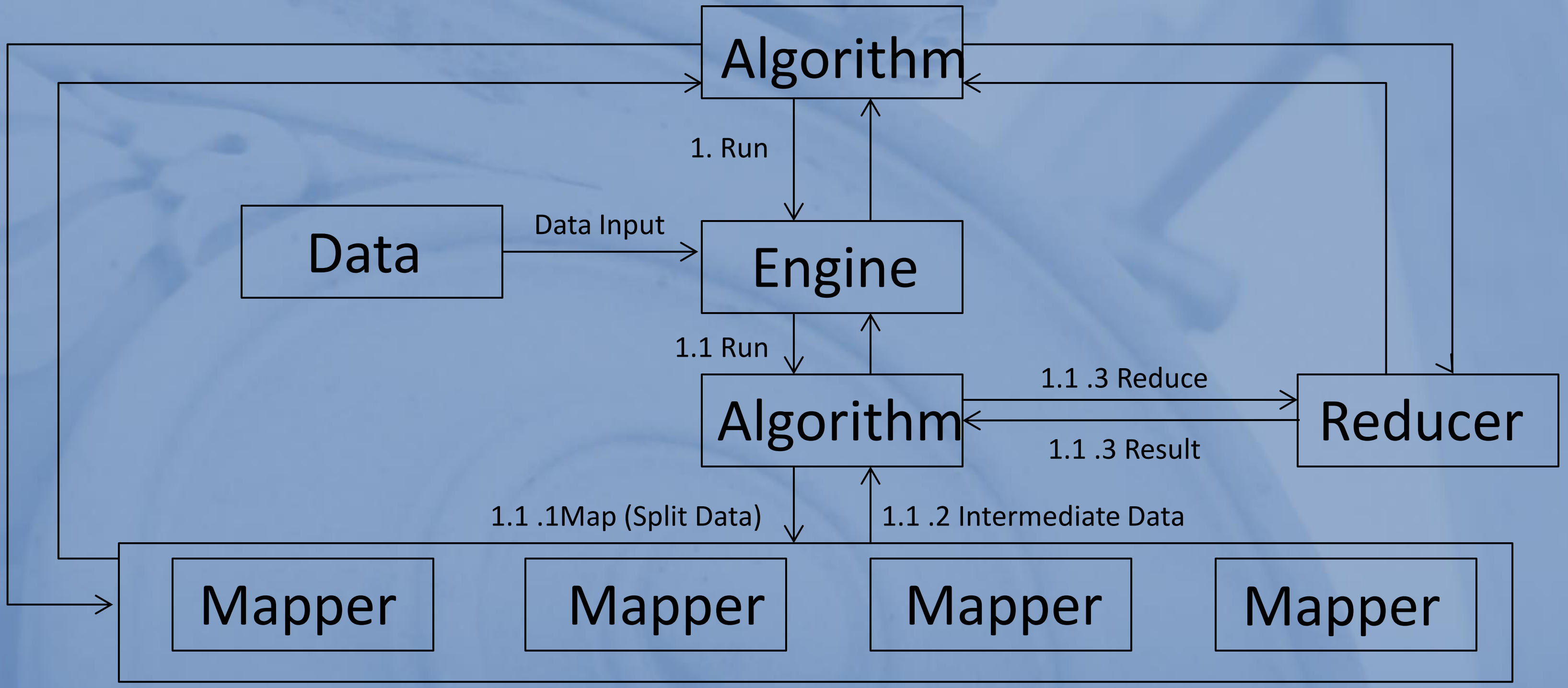
- Tremendous growth in multicore systems, yet no Machine Learning algorithms which harnesses such powerful machines.

- Goal: develop a general and exact technique for parallel programming of a large class of ML algorithms for multicore processors.

Value Addition

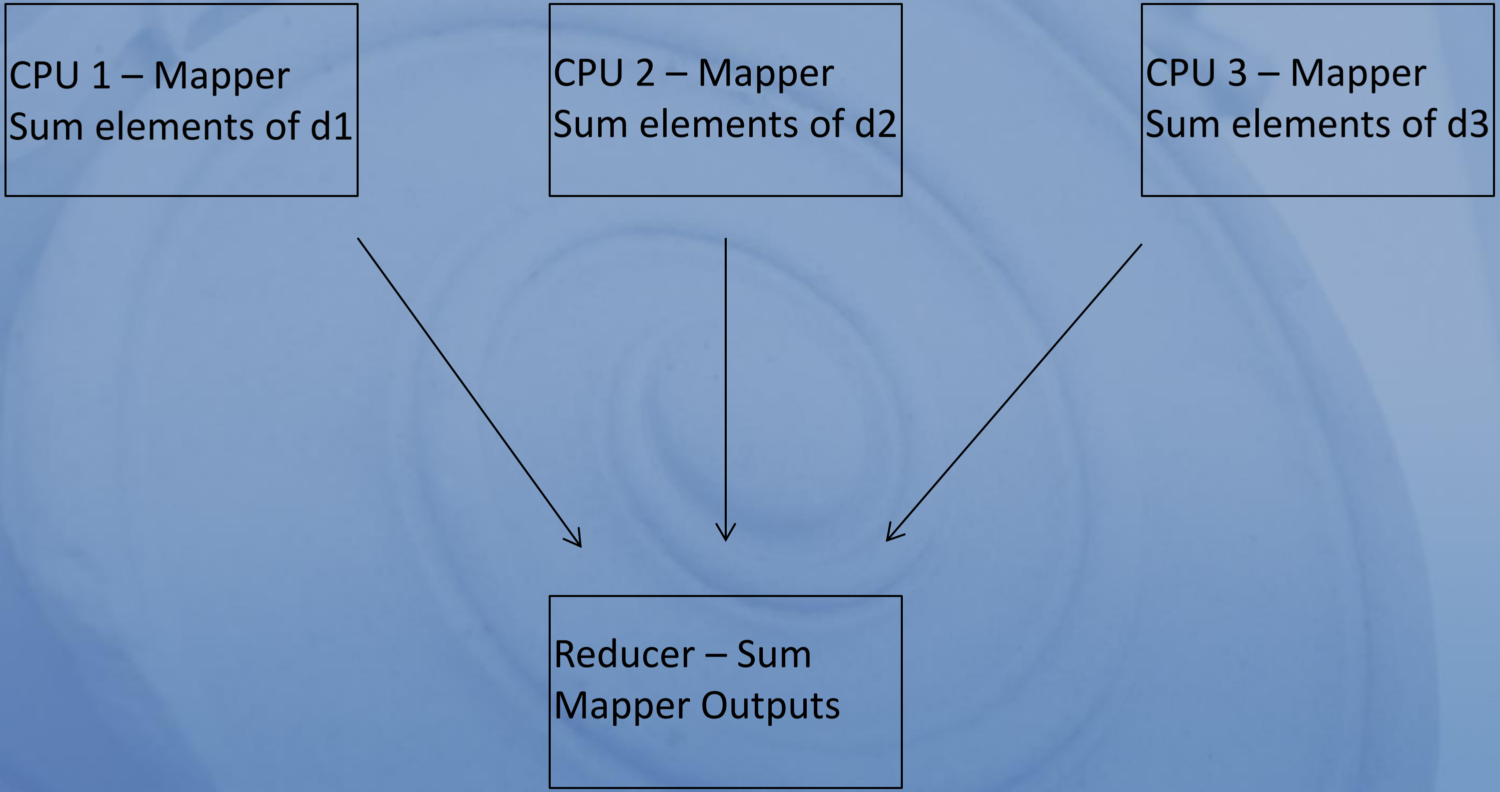
- General idea but still exact and useful.
- Used Google’s map-reduce paradigm to demonstrate parallel speed up technique – a generic paradigm for distributed, scalable computing.
- Original Goal of Apache Mahout project.

Map Reduce Architecture



Map Reduce Simple Addition Example

- Data set D divided into d1, d2, ... dn (each a list of things that can be summed)
- Mappers running on CPU 1, CPU 2, ... CPUn. Each mapper forms a sum over its piece of the data and emits the sum s1, s2, ... sn



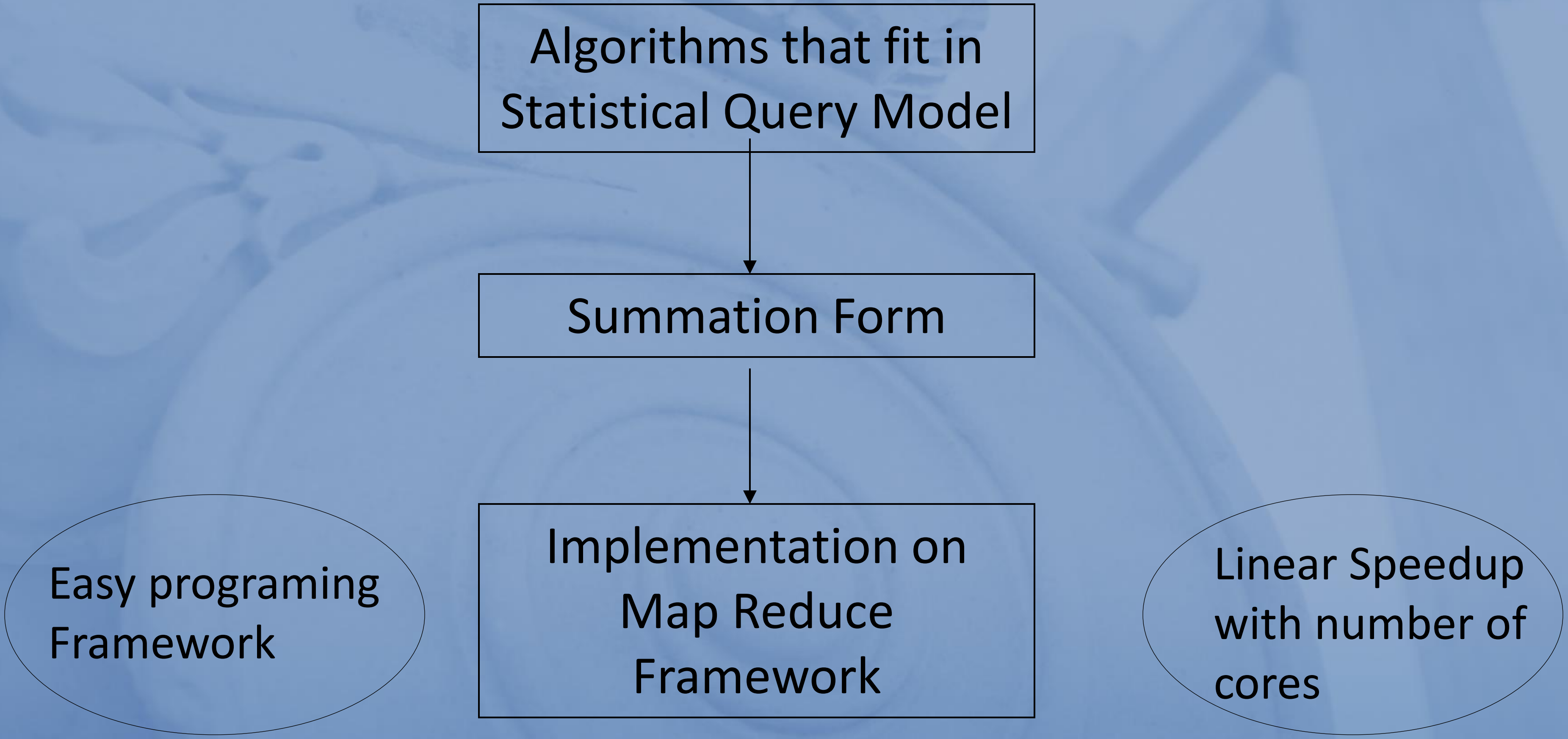
Statistical Query Model

A two step process for summation form -

1. Compute sufficient statistics by summing some functions over the data
2. Perform calculation on sums to yield data mining model.

- When an algorithm does sum over the data, we can easily distribute the calculations over multiple cores: We just divide the data set into as many pieces as there are cores, give each core its share of the data to sum the equations over, and aggregate the results at the end. We call this form of the algorithm the “summation form.”

Statistical Query Model


Algorithms that fit in Statistical Query Model
Summation Form
Implementation on Map Reduce Framework
Easy programing Framework
Linear Speedup with number of cores

Linear Regression using Least Squares

- Given m outputs y_i (also called labels, observations, etc.)
- And m corresponding attribute vectors x_i (also called regressors, predictors, etc.)

$$\mathrm{J}(\theta) = \frac{1}{2N}\left\| \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ Y_N \end{bmatrix} - \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1K} \\ 1 & x_{21} & x_{22} & \cdots & x_{2K} \\ & & \cdots & & \\ & & \cdots & & \\ & & \cdots & & \\ 1 & x_{N1} & x_{N2} & \cdots & x_{NK} \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_K \end{bmatrix} \right\|^2$$

Linear Regression using Least Squares

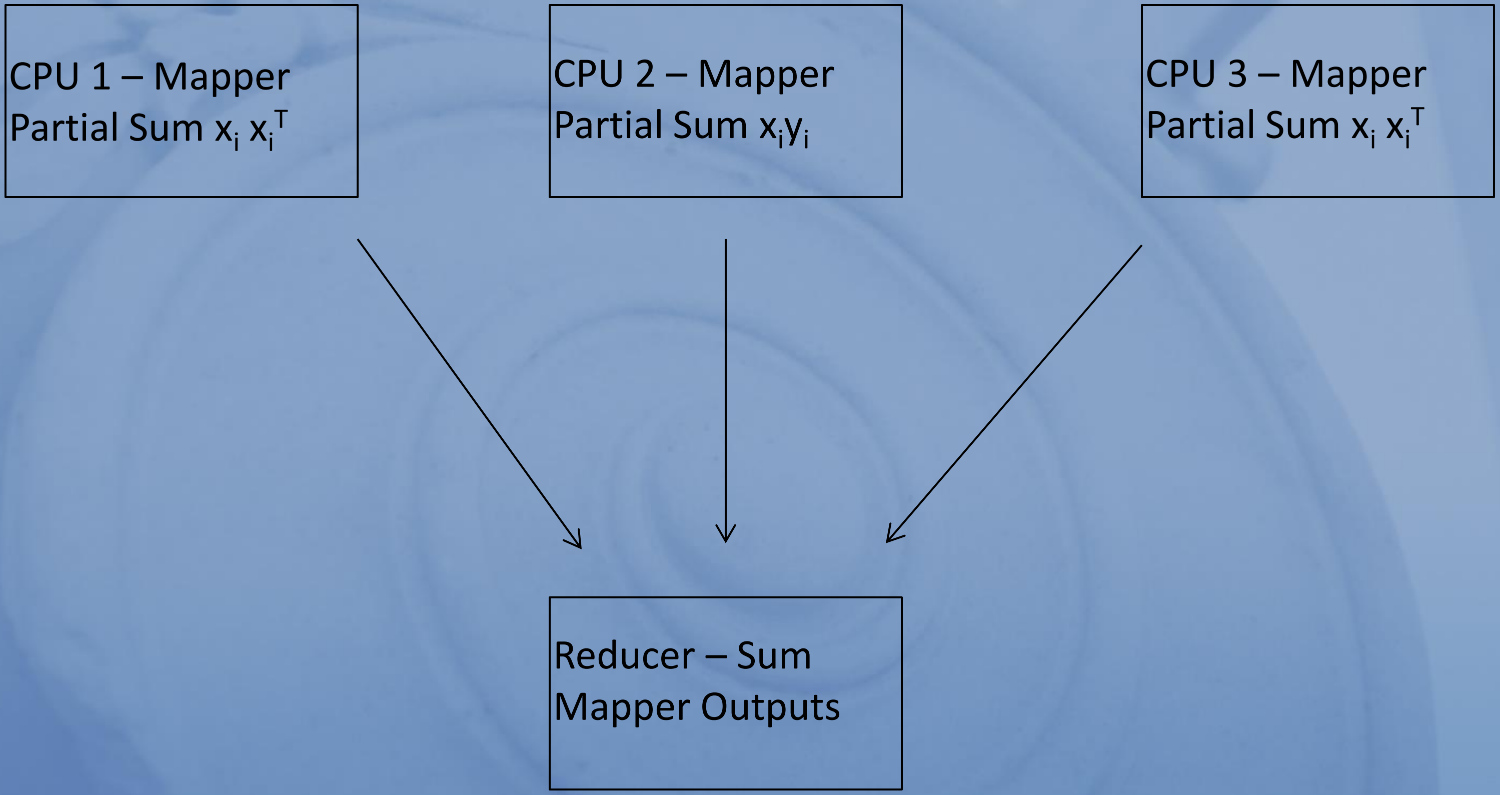
- Fit a model of the form $y = \theta^T x$ by solving –

  $\theta^* = \min_\theta \sum_{i=1}^m (\theta^T x_i - y_i)^2$

  $\Theta^* = A^{-1} b$ where $A = \sum_{i=1}^m (x_i x_i^T)$ and $b = \sum_{i=1}^m x_i y_i$

- We can see natural division into mapper-reducer by $\sum$

Linear Regression using Least Squares



Locally Weighted Linear Regression

Model: $y = \Theta^T X$

Goal: $\Theta* = \min_\Theta \sum_{i=1}^{m} (\Theta^T X_i - Y_i)$

Solution: Given m examples: $(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)$. We write a matrix X with $x^1, \ldots, x^m$ as rows, and row vector $Y=(y^1, y^2, \ldots y^m)$. Then the solution is –

$$\Theta* = (X^T X)^{-1} X^T Y$$

Parallel computation: $\Theta* = A^{-1}B$

Cut to m/num processor pieces

$A = (X^T X)$ $\quad\quad$ $A = \sum_{i=1}^{m} w_i (x_i x_i^T)$

$B = X^T Y$ $\quad\quad$ $B = \sum_{i=1}^{m} w_i (x_i y_i)$

Mappers: one sets compute A, the other set compute b

Two reducers for computing A and b

Finally compute the solution

Naive Bayes

Goal: estimate $P(x_j = k|y = 1)$ and $P(x_j = k|y = 0)$

Ex – P(Y=mal | X*={circ, sm, li}) = P(Y=mal) * P(x_1=circ|Y=mal) * P(x_1=sm|Y=mal) * P(x_1=li|Y=mal).

Computation: count the occurrence of (x_j=k, y=1) and (x_j=k, y=0), count the occurrence of (y=1) and (y=0), then compute division

Mappers: count a subgroup of training samples (Just like the word count example).

Reducer: aggregate the intermediate counts, and calculate the final result.

Logistic Regression

Another generalized linear model (GLM) procedure using the same basic formula but has Categorical outcome.
The equation for the probability of Y looks like this:

$$P(Y) = 1+e^{-(\Theta 0+\sum(\Theta i\, Xi))}$$

Learning is done by fitting θ to the training data where the likelihood function can be optimized by using Newton-Raphson to update $\theta := \theta - H^{-1}\nabla_\theta l(\theta)$.

$\nabla_\theta l(\theta)$ is the gradient, which can be computed in parallel by mappers summing up $\sum_{subgroup} (y^{(i)} - h_\theta(x^{(i)}))x^{(i)}{}_j$ each Newton-Raphson step i.

The computation of the hessian matrix can be also written in a summation form of $H(j,k) := H(j,k) + h_\theta(x^{(i)})(h_\theta(x^{(i)})-1)x^{(i)}{}_j\ x_{(i)}{}^k$ for the mappers.
The reducer will then sum up the values for gradient and hessian to perform the update for θ.

K-means

Mapper – given K initial guesses, run through local data and for each point, determine which centroid is closest, accumulate vector sum of points closest to each centroid, combiner emits <centroidi , (sumi , ni)> for each of the i centroids.

Reducer – for each old centroid(i) , aggregate sum and n from all mappers and calculate new centroid.
This map-reduce pair completes an iteration of Lloyd's algorithm.

Neural Network (NN)

Focus on Back-propagation.
3-layer network, Input, middle, 2 output nodes

Goal: compute the weights in the NN by back propagation

$$E = \frac{1}{2} \sum_{i=1}^{m} (w_i a_i - y_i)^2$$

Mapper: propagate its set of training data through the network, and propagate errors to calculate the partial gradient for weights

Reducer: sums the partial gradients and does a batch gradient descent to update the weights

Principal Components Analysis (PCA)

Map: Compute Principle eigenvectors of the covariance matrix , $\sum = 1/m * (\sum_{i=1}^{m} x_i x_i^T) - \mu\mu^T$

and Mean Vector, $\mu = 1/m \sum_{i=1}^{m} x_i$

Clearly, we can compute partial summation form using map.

Reducer: Sum up partial results to produce final empirical covariance matrix.

Gaussian Discriminative Analysis (GDA)

Goal: Classification of x into classes of y
assuming each class is a Gaussian Mixture model with different means but same covariance.

Computation: P(y), u_0, u_1 and $\sum$.

Mappers: Each mapper will handle the summation (i.e. Σ 1{yi = 1},Σ 1{yi = 0},Σ 1{yi = 0}xi, etc) for a subgroup of the training samples.
Reducer will aggregate the intermediate sums and calculate the final result for the parameters.

Other Algorithms

There are few other algorithms which can be parallelized in similar summation form -

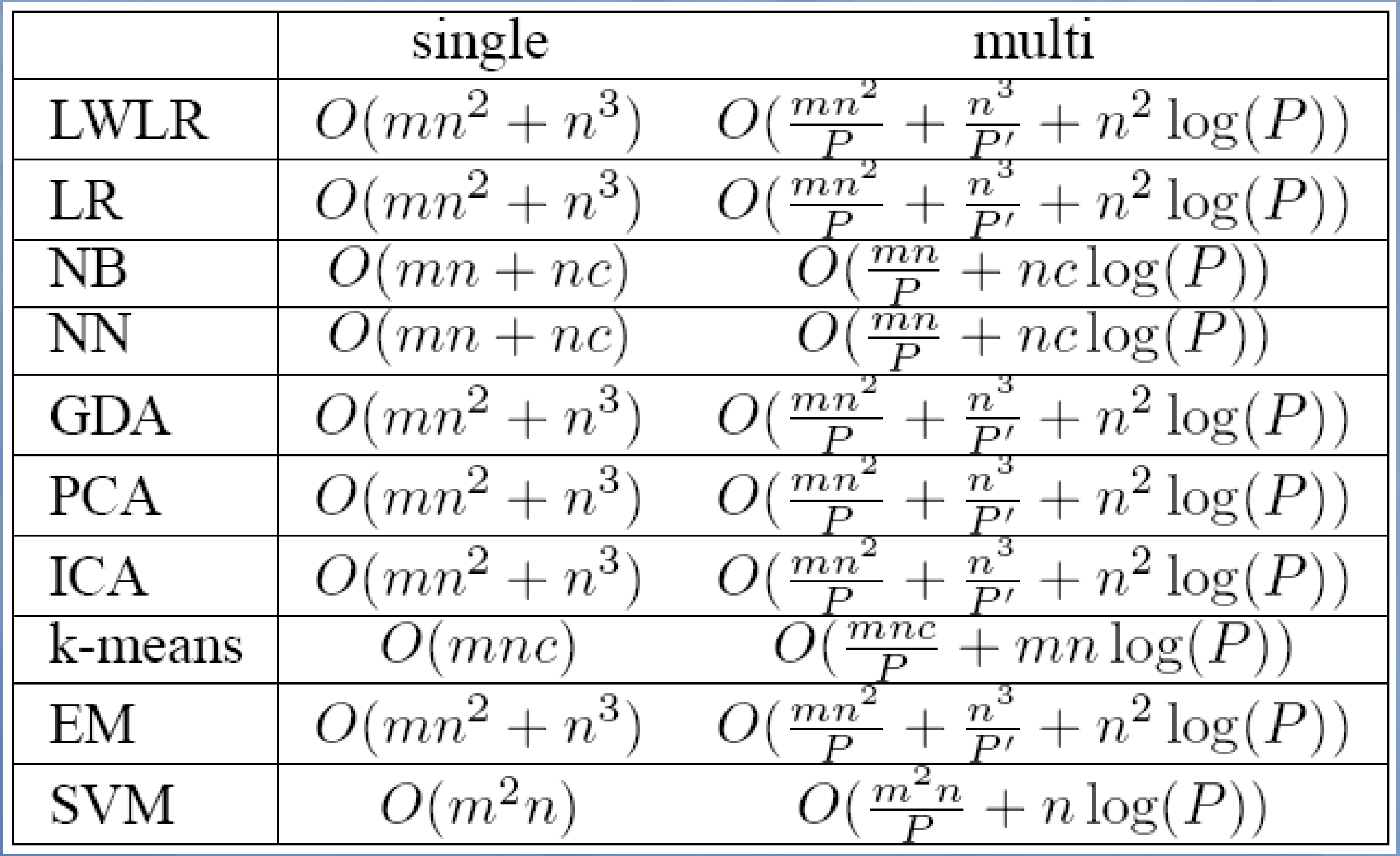
- SVM
- Independent Component Analysis

Mahout, MLib.

Experiment Setup

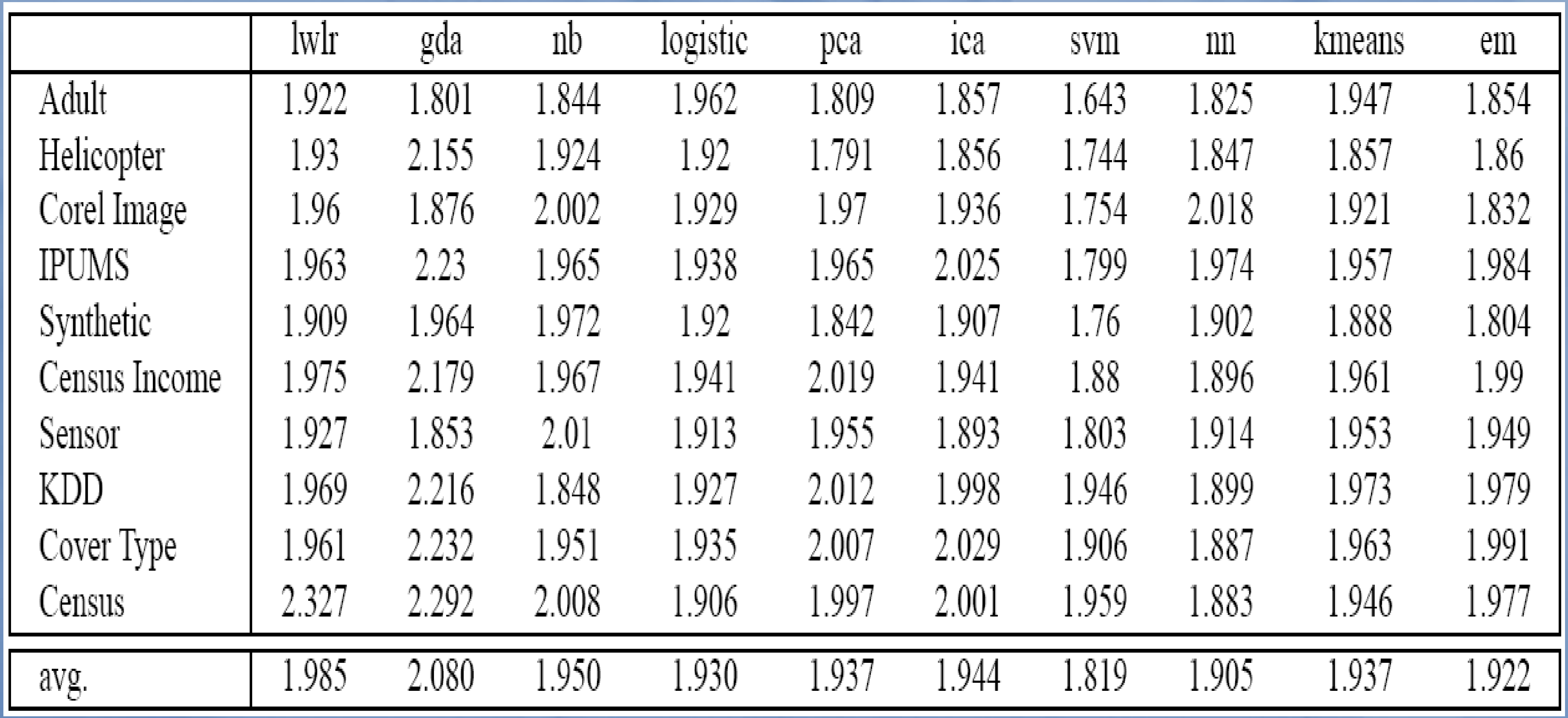
- Compare map-reduce version and sequential version
- 10 data sets (8 - UCI Machine Learning Repository and 2 – Helicopter Control and Sensor Data)
- Machines:
  - X-86, Dual-processor Pentium-III 700MHz, 1GB RAM
  - Linux RedHat 8.0 Kernel 2.4.20
  - 16-way Sun Enterprise 6000 using Solaris 10 using 1, 2, 4, 8, 16 cores.

Time Complexity



	single	multi
LWLR	$O(mn^2 + n^3)$	$O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2 \log(P))$
LR	$O(mn^2 + n^3)$	$O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2 \log(P))$
NB	$O(mn + nc)$	$O(\frac{mn}{P} + nc \log(P))$
NN	$O(mn + nc)$	$O(\frac{mn}{P} + nc \log(P))$
GDA	$O(mn^2 + n^3)$	$O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2 \log(P))$
PCA	$O(mn^2 + n^3)$	$O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2 \log(P))$
ICA	$O(mn^2 + n^3)$	$O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2 \log(P))$
k-means	$O(mnc)$	$O(\frac{mnc}{P} + mn \log(P))$
EM	$O(mn^2 + n^3)$	$O(\frac{mn^2}{P} + \frac{n^3}{P'} + n^2 \log(P))$
SVM	$O(m^2n)$	$O(\frac{m^2n}{P} + n \log(P))$

Dual-Processor SpeedUps



	lwlr	gda	nb	logistic	pca	ica	svm	nn	kmeans	em
Adult	1.922	1.801	1.844	1.962	1.809	1.857	1.643	1.825	1.947	1.854
Helicopter	1.93	2.155	1.924	1.92	1.791	1.856	1.744	1.847	1.857	1.86
Corel Image	1.96	1.876	2.002	1.929	1.97	1.936	1.754	2.018	1.921	1.832
IPUMS	1.963	2.23	1.965	1.938	1.965	2.025	1.799	1.974	1.957	1.984
Synthetic	1.909	1.964	1.972	1.92	1.842	1.907	1.76	1.902	1.888	1.804
Census Income	1.975	2.179	1.967	1.941	2.019	1.941	1.88	1.896	1.961	1.99
Sensor	1.927	1.853	2.01	1.913	1.955	1.893	1.803	1.914	1.953	1.949
KDD	1.969	2.216	1.848	1.927	2.012	1.998	1.946	1.899	1.973	1.979
Cover Type	1.961	2.232	1.951	1.935	2.007	2.029	1.906	1.887	1.963	1.991
Census	2.327	2.292	2.008	1.906	1.997	2.001	1.959	1.883	1.946	1.977
avg.	1.985	2.080	1.950	1.930	1.937	1.944	1.819	1.905	1.937	1.922

Multicore Simulator Results

- Results are better than multicore machines than multiprocessors.
  - Could be because of less communication cost
  - Results are in line with the complexity analysis.
  - Speedups achieved involved no special optimizations of the algorithms themselves.

Conclusion

- Parallelize summation forms
- Use map-reduce on a single machine

Questions

Map-Reduce for Machine Learning on Multicore

Aniket Santosh Deshpande
ad78@buffalo.edu

Strengths:
- It takes advantage of summation in MapReduce framework to parallelize a lot of machine learning algorithms.
- The linear speedup achieved did not involve ANY optimizations on algorithms themselves

Weaknesses:
- Not all machine learning algorithms can be fit into the "summation" mould that the paper talks about
- The size of data plays a significant role in gauging performance of any framework. The size of data set is not mentioned in the paper.

Nikhil Singhal
nsinghal@buffalo.edu

Weakness:

- Separate engine required for every algorithm to parallelize it.
- Under experiments section there is no mention of the size of the data set used.
- Comparison against serial execution of the algorithm is biased towards boosting the results shown in this paper.
- Comparison against MapReduce on distributes systems would have been much more informative.

Strengths:

- If this solution can be incorporated with the MapReduce, it may provide a linear increase in the speed, bringing along the ability to manage terabytes of data.

Note:

- This work can be extended by performing calculations on CUDA platform, where instead of a few cores, calculations can be spread over thousands of cores.

Ladan Golshanara
ladangol@buffalo.edu

Strength:
The idea is novel and interesting. Almost all of the machine learning algorithms are iterative so their architecture can handle that. They had a good observation on algorithms that can be put in their summation form. I think overall they achieve their goal which was a general framework for paralleling machine learning algorithms on multi-core.
Weakness:
In my opinion, two things are not justified well in their paper. They mentioned that multi-cores do not have unreliable communication as in clusters. But what are some specific things related to multi-core that is ignored by map-reduce, if there are any. For example, in terms of memory management. In other words, to me the authors should have elaborate more on why they thought map reduce model which was originally proposed on clusters is still good for multi-cores. Are there any characteristics specific to multi-cores that clusters do not have?
Also, in their architecture in Figure 1, they use many mappers but one reducer, I could not find reasons why a unique reducer is necessary or enough? can we have more or not?

Pragna Basu
pragnaba@buffalo.edu

Weakness:
- Multiple reducers could have been used for faster execution.
- Insufficient data provided. Results that have been used to display the 'speedup' are on multi processors and not multi cores.

Strength:
Idea that any algorithm fitting the Statistical Query Model may be written in a certain “summation form” and thus be divided into separate tasks on separate cores thereby reducing the idle time of the processor is simple and interesting.

Simranjot Singh
simranjo@buffalo.edu

Strength : The results show a linear speedup with increase in the number of processors for almost all the algorithms.

Weakness : The paper was only able to cover exact ML algorithms while study of their parallel approximations using the MR technique were left out.

Gaurav Kshirsagar
gkshirsa@buffalo.edu

Strengths:
Parallel execution of tasks to compute the individual values (e.g. matrix multiplications) is a good way of speeding up any Machine learning algorithm since it has a lot of computations and iterations.

Weaknesses:
Time complexity calculation is based on just one iteration of the algorithms.
This method is weakly related to MapReduce. Typical mapping (transforms, modification or extraction) and reducing (summation, sorting) operations are not performed on the data in those phases respectively.
MapReduce was run on a bunch of machines and the experiments for this were run on a single machine. For better understanding of how it compares with MapReduce, it should be run on cluster.

Pratik Pramod Chavan
pchavan2@buffalo.edu

Strengths:
The architecture takes advantage of the fact that original algorithms do not utilize all the cpu cycles
efficiently, but do better when we distribute the tasks to separate threads/processes. Also, they have conducted quite a lot of experiments with variety of algorithms achieving great results.

Weaknesses:

The algorithms themselves haven't been tweaked. Even though good in a way (just focussing on the number of cores), it would have been nicer to see if some minor changes would improve the performance significantly.

Daniel Frederick Nazareth
dnazaret@buffalo.edu

Strengths: Directly utilizes multicore machines for true parallelism. Many multicore distributed systems confuse parallelism for concurrency and achieve no measurable speedup. This is certainly not the case for this mechanism looking at the significant speedup for each algorithm.
-There is an impressive amount of variety in terms of algorithmic approaches,datasets and machine learning techniques which gives a lot of weight to the authors assertion that having an effective parallel programming approach for map reduce can greatly increase speedup

Weaknesses
-The paper somewhat restricts itself to "parallel prefix sum" or as the author puts it the "summation form" of the parallel algorithm. Some other parallel prefix operations could have been explored.
-The algorithm complexity is extremely broad, makes a large number of mathematical assumptions and does not explain what data was used to test the iterative form of the algorithms for comparison.

XIN MA
xma24@buffalo.edu

Strength:
Instead of adding more computers to the distributed system to reinforce the computing power, the authors focus on the potential computing power of individual machines, i.e., multicore architecture, and then propose a new programming framework.

Weakness:
Actually, when we use a multicore computer, its performance is just there. From my perspective, instead of comparing the performance of a single core computer with a multicore computer or a multicore computer with small number of cores with a multicore computer with large number of cores, the optimization among those cores in the multicore computer is much more significant.

Madhur Gupta
madhurgu@buffalo.edu

Strengths : Map-reduce utilizes true parallelism. Original algorithms do not utilize all the CPUS cycles efficiently, but Map reduce distributes the tasks to separate threads/processes. Hence achieves true parallelism.

Weakness : Time based analysis is based on single iteration of algorithms. Comparison of the algorithms should have been done on distributed systems.

krishnakant Chavali
kchavali@buffalo.edu

Strength:
Achieve linear speed up using multiple cores.
Write any algorithm fitting the Statistical Query Model into its summation form which can then be easily expressed as Map/Reduce functions.

Weakness:
Implementation only optimizes a few types of mathematical operations which have a summation form.
Complexity is not calculated accurately because it was only calculated over one iteration.
Size of the data sets used for experiments is not mentioned which kinda not explains the performance of the implementation on big or small data sets.

Yan Sun
ysun27@buffalo.edu

Strengths:
By adapting "summation form", many machine learning algorithms can easily implemented on MapReduce paradigm.

Weakness:
The model is based on multicore. One major difference between multicore and multiprocessor is that the tasks deployed on mulitcore communicate much easier. But the paper does not take advantage of it.

Karthik Kakarla
kkakarla@buffalo.edu

Strengths:
1)Many machine learning algorithms can be written in summation form and hence can be easily parallelised using this model

Weaknesses:
1)Having multiple reducers could have been more effective. The need for a single reducer is not explained well

Sai Srinath Sundar
saisrina@buffalo.edu

Strength:
The model provides horizontal scalability with linear speedup which is great as this problem was not addressed effectively previously.This translates to effective CPU utilization as well.
Weakness:
Most of the parallelizations seem to revolve around implementing the parallel prefix algorithm wherever it can be implemented(typically in summations) There is nothing specifically unique about this being a specific optimization for machine learning techniques. This can be implemented for any summation related algorithm.
It is mentioned that the dataset must be split across the cores by the user in many of the applications. When working with large datasets to be processed across a large amount of cores, asking to user to do such tasks may not be a practical solution. This creates a new problem by itself.
While the mentioned optimizations improve the performance for Machine Learning applications, it does not effectively address the fundamental problem related with Map Reduce on Machine Learning algorithms namely that an iterative model does not fit too well with Map Reduce. Even with multi core systems that problem will still persist and will still be the primary problem.

Qiuling Suo
qiulings@buffalo.edu

The paper makes use of the summation form in a map-reduce framework and proposes a potentially applicable technique for parallel programming of various

machine learning algorithms. The idea to speed up the machine learning applications is to use more cores instead of search for the optimizations. Through testing various machine learning algorithms, they have shown that their model is able to parallelize them and achieve a visible speedup. It may provide a simple and powerful way to make machine learning applications attainable and efficient.
Weaknesses:
-It is not sure that this framework is a general model to satisfy all the machine learning algorithms. But I don't think it is necessary to prove this because the machine learning algorithm is ever changing.
-The scales of data sets are missed in the experiments.

Prashanth Balaboina
pbalaboi@buffalo.edu

Strengths:
Like Map and Reduce is known for its simplicity. The more the number of machines the parallel the tasks can be. Failure of a machine can be recovered by including proper back up tasks on other machines.

Weaknesses:
The number of instances that perform the map and reduce functions need to be controlled. Finally the efficiency of the algorithm is also equally important. Many ML algorithms requires few tasks sometimes and iterative frequencies often. This could be a problem however.

Liuyi Yao
liuyiyao@buffalo.edu

Strengths:
Multicore process achieves linear speed up on machine learning applications.

Weakness:
Though in the paper, many Machine Learning algorithm can be adopted to this framework. However, It uncertain that all Machine Learning algorithms can be done like this.

Hongfei Xue
hongfeix@buffalo.edu

Strength:
The author gave out a plausible parallel solution for many kinds of machine learning algorithms.
MapReduce, which has been proved to be highly efficient is leveraged to implement this method.

Weakness:
The prerequisite for a solvable problem by this method is that the problem can be formulated as summation form. So the method is useless to those problems which have no form of summation.

Utkarsh Srivastava
u2@buffalo.edu

Strength :
Machine learning algorithms listed here follow Two different phases : They need matrix/parameter calculation followed by summation. Therefore, In the first phase, there is a huge scope for parallel processing will dramatically increase the thoroughput as there is scope for independent processing without bothering for co-ordination between the processors. This voids the communication overhead; An ideal scenario for delegating work to various mappers Summation and final computation is the second and final phase, which can always be done during the reducer phase in a similar manner.
Weakness
I think the main drawback here is that in many cases, the map reduce algorithm inhibits our ability to take advantage of parallel processing properly at every stage by forcing us to do certain complex computations at only certain stages. For instance, hessian matrix and gradient is a rather complex calculation and but we are forced to calculate it only during the mapper stage and the reducers are comparitively less loaded than mappers . We also fail to fit in, the huge number of tools and libraries that are available for ML algorithms into map reduce

---

Stephen Gung
sgung@buffalo.edu

Strengths:
Results show increase in speedup as more cores are used
Weakness:
Additional cores doesn't necessarily solve the fundamental situation Machine Learning algorithms are tasked with, rather a just a "short cut"