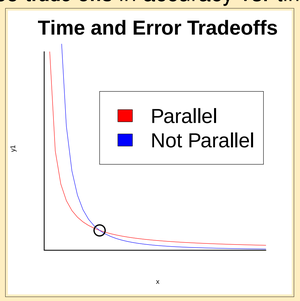
An Optimization Layer for Distributed Matrix Computations

Jonah Brown Cohen, Tselil Schramm, and Ben Weitz

Motivation

- Big data companies like Facebook, Netlix, or Google perform large-scale distributed matrix computations
  - Computations experience **trade-offs** in accuracy vs. time or money.



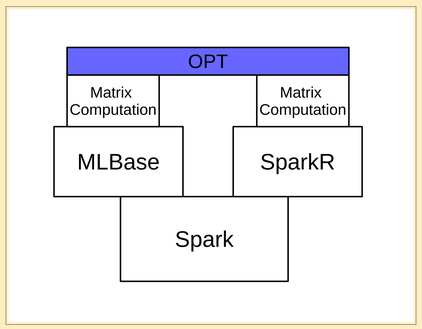
  - Human operators manually tweak parameters and partitioning
  - Humans are prone to error and costly to hire!
- Solution: Build an optimization layer to automatically tweak and manage these computations
  - Learn to adjust parameters from past computations
  - Incoming jobs come with budgets of time or accuracy that must be met

Objective

- **Create an optimizer that automatically picks algorithm parameters and the degree of data partitioning to meet budget specifications**

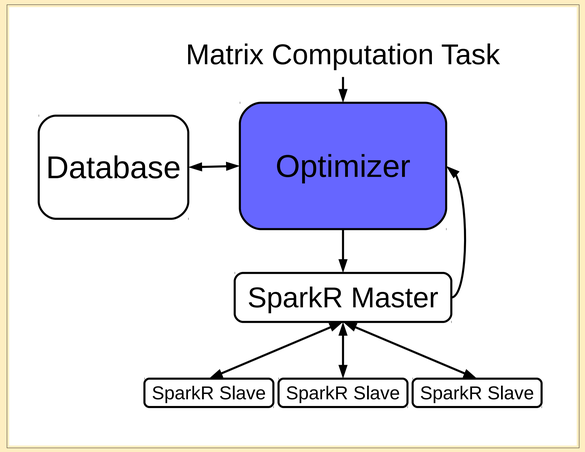
Framework

- Optimizer built on Python
- Interfaces with matrix algorithms implemented in SparkR or MLBase
- Optimizer interfaces directly with algorithms, all parameters hidden from user



Optimizer Design

- Architecture-independent
  - Chooses parameters based on statistics from prior jobs
  - sdflksad
- Adaptive
  - sdfad
  - sdf
- Local-optimum Avoiding
  - sdfad
  - sdf



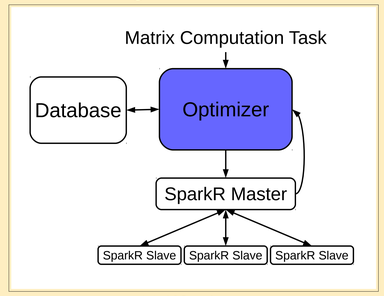
Implementation

- The words chosen by the adversary are hard,
  - But once you know theyre difficult it's easy to adjust.
- Top 12 words (probability of losing in 6 turns):

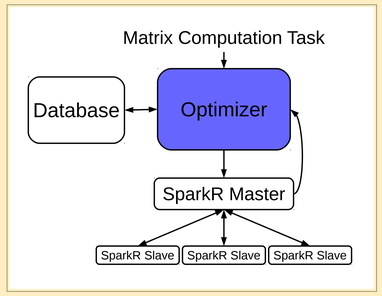
Evaluation

Tested on Synthetic and Real-world Data

- Gaussian Random Matrices
  - Trained on eight random matrices and tested on a ninth



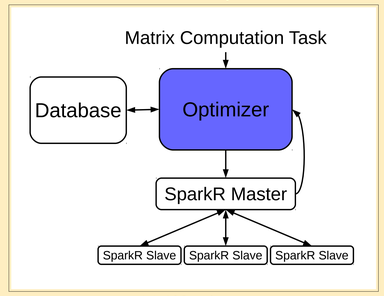
(a) AIC values.



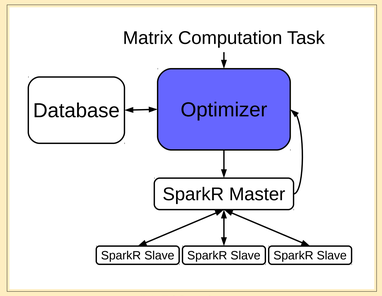
(b) BIC values.

Figure : Plots of AIC and BIC Values against the Memory Parameter for a Dictionary Player with True Memory 2

- MovieLens 10M Dataset
  - Partitioned dataset into 7 parts
  - Trained on all but one partition and tested on the remainder



(a) AIC values.



(b) BIC values.

Figure : Plots of AIC and BIC Values against the Memory Parameter for a Dictionary Player with True Memory 2

- Optimizer performs as well as manually setting the parameter!

Future Work

- Optimize over space of algorithms in addition to space of parameters
- Avoid RAM bottlenecks by distributing collect step
- Handle novel or different jobs

Acknowledgements

We would like to thank:

- Shivaram Venkataraman, Ameet Talwalkar, Prof. Anthony Joseph, and Prof. John Kubiatowicz
- UC Berkeley and the NSF for providing funds and resources.