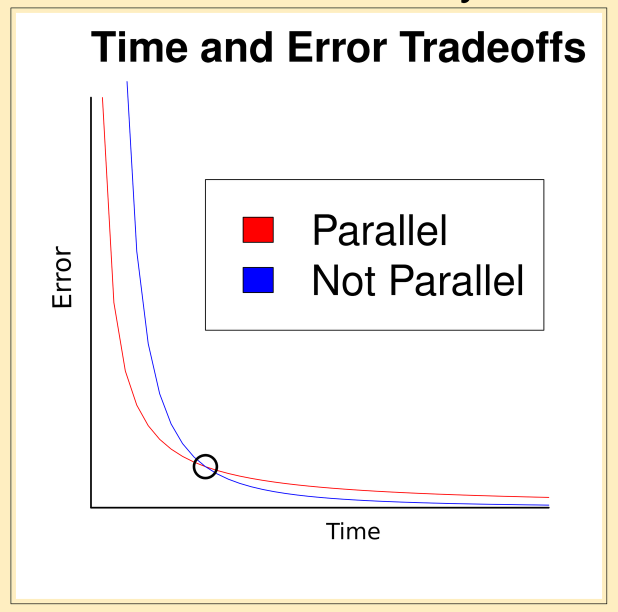
An Optimization Layer for Distributed Matrix Computations

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.



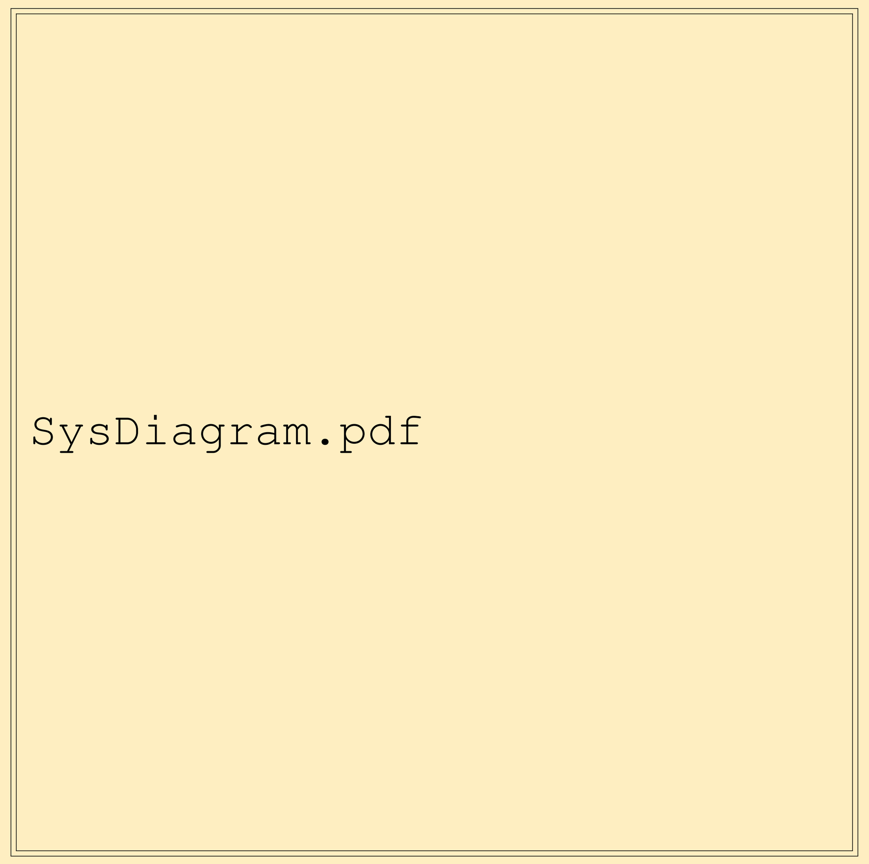
  - Human operators manually tweak parameters and partitioning
  - Humans are prone to error and costly to hire!
- Solution: Build an optimization layer to automatically tweak and manage these computations
  - Learn to adjust parameters from past computations
  - Incoming jobs come with budgets of time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

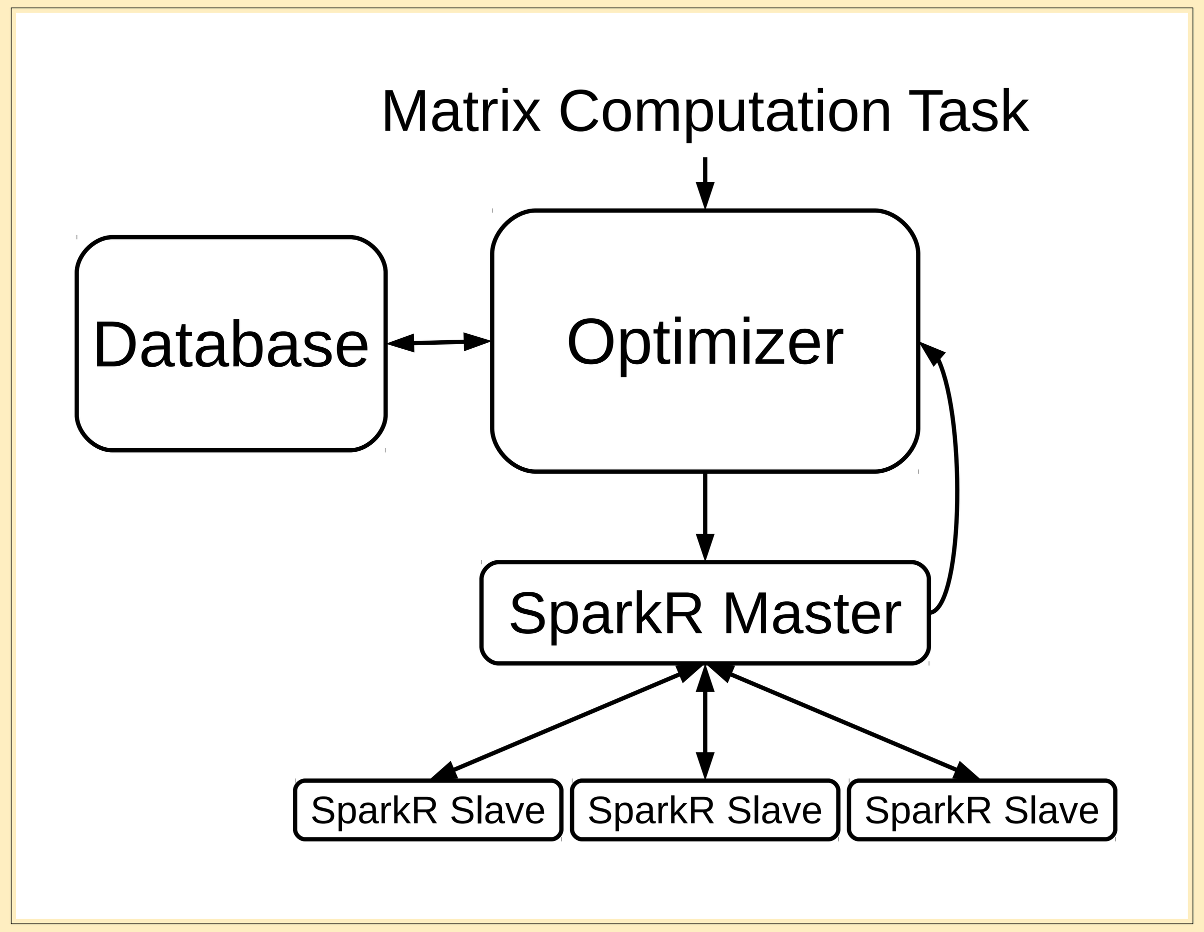
Framework

- 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
  - Stores statistics from previous jobs on instances with similar distribution, and improves predictions with time.
  - sdf
- Local-optimum Avoiding
  - In “explore mode” we choose the instance parameters randomly, according to a distribution specified by the
  - sdf



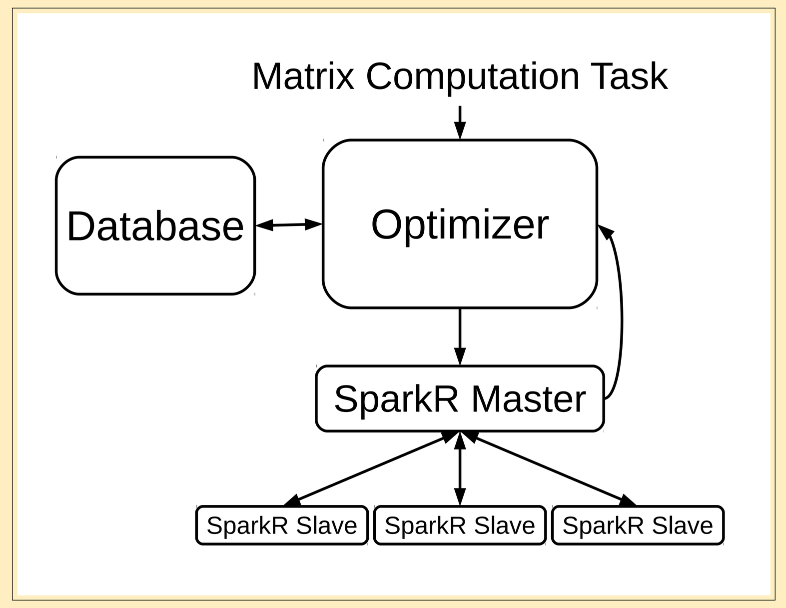
Implementation

- 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):

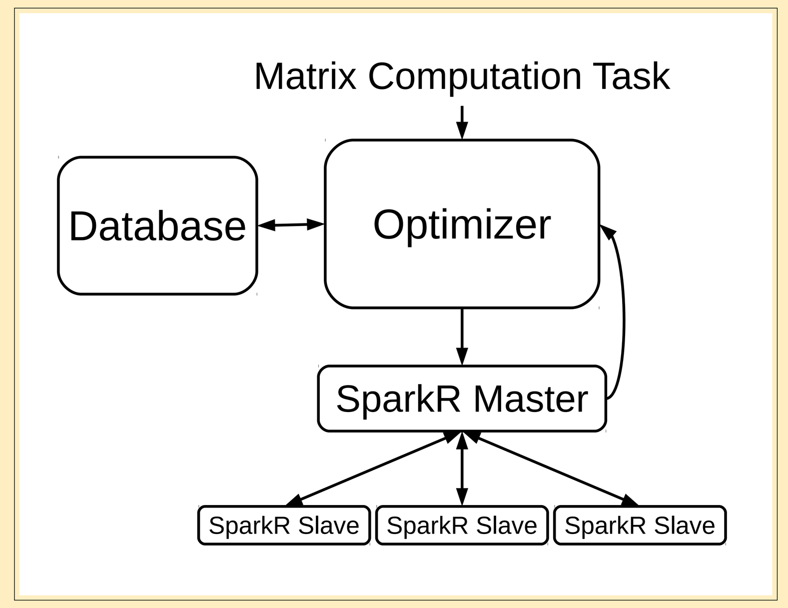
Evaluation

Can the AI determine the correct memory of a player?

- Generated data for players with restricted memory.
  - Extremely large number of samples required for AIC.
  - Can only implement model with memory parameters 1 through 3.
- Computed (corrected) AIC and BIC values
  - AIC consistently overestimates.
  - BIC consistently underestimates.
- Failure of AIC due to information available to player not captured by the model.



(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

Future Work

Achievements:

- Can learn a player’s strategy assuming restricted memory.
- Can choose words that are hard for that player.
- Compiled a list of hard words for a dictionary-using frequency player.
  - Also hard for regular humans.

Future Work:

- Online Learning.
- Foiling an Adaptive Player.
  - Can we learn an adaptive strategy quickly and counter it?
  - Is there a Nash Equilibrium to the responses?

Acknowledgements

We would like to thank:

- Anthony.
- UC Berkeley and the NSF for providing funds and resources.