# todo - outline

- (Horace) Short (very very short) problem description
- Solution description
  - (Kevin) High level overview
  - (Sweet) Technical details
  - (Horace) Results (i.e. numbers they want to see, perhaps in graph form)
  - (Sweet) Testing (custom files)
- (Kevin) Lingering issues?
- (Sweet) Q&A

# CS 5300: Final Project Presentation

Horace Chan (hhc39), Kevin Chen
(kfc35), Sweet Song (ss2249)

May 2, 2013

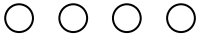
# Agenda

Introduction

Solution Overview

Lingering Issues

Q & A



# Problem Description

## Fast Convergence PageRank in Hadoop

- compute PageRank for a reasonably large Web graph (685230 nodes, 7600595 edges)
- Blocked Matrix Multiplication instead of a direct node-by-node approach

# Algorithm Overview - Simple PR

<nodeID, (pageRank numOuts (List: outNodes)>

MAP

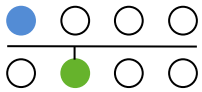
<nodeID, (incomingPR) OR (pageRank numOuts (List: outNodes)>

REDUCE

<nodeID, (newPR numOuts (List: outNodes)>

Reducer receives either an incomingPR value (from a diff. node) to sum or "initial data"

High Level Overview



Solutions Overview

# In Depth Description - Simple PR

- Mapper:
  - Parses the value and sends its rank to its outNodes or to the counter for no outNodes
- Reducer:
  - Adds its residual to the counter (multiple the number by 10E12)
  - Output the files in the exact same format as mapper receives them

# Algorithm Overview - Block PR

Block PR MR

<nodeID, (pageRank numOuts (List: outNodes)>

MAP

<blockID, (typeOfValue ...)>

REDUCE

<nodeID, (newPR numOuts (List: outNodes)>

typeOfValue indicates whether the value is of initial data, an edge within the block, or a boundary PR for a certain node.

# In Depth Description - Block PR

- Mapper:
  - Parses the value and sends the value to its node
  - If it does have any out edges, add its pagerank to the counter for no out edges
  - Calculates its outrank to each outNode
    - If its outNode is in the same block, send its nodeID to it.
    - Else send its outrank to that node's block.

# In Depth Description - Block PR

- Reducer:
  - Process all edges to create various structures and pageranks.
  - Iterate at most 10 times or until the block residual is less than 0.01. (Keep the iteration average)
  - Find the biggest nodeID in each block and pass it to its corresponding counter

# Average Residual Per Iteration

| Iteration Number | Average Residual |
|---|---|
| 0 | 2.338671828254877 7 |
| 1 | 0.3229799562618682 6 |
| 2 | 0.1918914214682807 3 |
| 3 | 0.0940770135730922 4 |
| 4 | 0.0628498339289581 7 |

# Biggest Node in Block PageRank Values

# Testing Our Solution

- Testing was incremental.
  - 4 node sample graphs
  - Hundreds of randomly generated nodes
  - Counter/sanity checks
  - Local Hadoop cluster for simple computations

- & comparative
  - Computed smaller graphs by hand
  - Wrote another program for verification
  - Iterative results vs Block results.

# Problems?

None! :)

Questions?