Parallel Graph Reduce Algorithm for Scalable Filesystem Structure Determination

John Emmons[1], Kirby Powell[2]

[1]Drake University [2]St. Edwards University

August 5, 2013

Overview

Overview

- Project Motivation

Overview

- Project Motivation
  - Increased high performance computing demand

Overview

- Project Motivation
    - Increased high performance computing demand
    - Current algorithm limitations

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design
  - MapReduce parallel framework

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design
  - MapReduce parallel framework
  - Graph reduce path generating algorithm

Overview

- Project Motivation
    - Increased high performance computing demand
    - Current algorithm limitations
- Hadoop and Algorithm Design
    - MapReduce parallel framework
    - Graph reduce path generating algorithm
- Algorithm Performance and Results

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design
  - MapReduce parallel framework
  - Graph reduce path generating algorithm
- Algorithm Performance and Results
  - Metrics

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design
  - MapReduce parallel framework
  - Graph reduce path generating algorithm
- Algorithm Performance and Results
  - Metrics
  - Time complexity

Overview

- Project Motivation
    - Increased high performance computing demand
    - Current algorithm limitations
- Hadoop and Algorithm Design
    - MapReduce parallel framework
    - Graph reduce path generating algorithm
- Algorithm Performance and Results
    - Metrics
    - Time complexity
    - Scalability

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design
  - MapReduce parallel framework
  - Graph reduce path generating algorithm
- Algorithm Performance and Results
  - Metrics
  - Time complexity
  - Scalability
- Conclusions

Overview

- Project Motivation
  - Increased high performance computing demand
  - Current algorithm limitations
- Hadoop and Algorithm Design
  - MapReduce parallel framework
  - Graph reduce path generating algorithm
- Algorithm Performance and Results
  - Metrics
  - Time complexity
  - Scalability
- Conclusions
- Future work

Project Motivation

Project Motivation

- Increased demand for High Performance Computing (HPC)

Project Motivation

- Increased demand for High Performance Computing (HPC)
    - Consequent, increased fileystem size

Project Motivation

- Increased demand for High Performance Computing (HPC)
  - Consequent, increased fileystem size
  - New back up procedure needed

Project Motivation

- Increased demand for High Performance Computing (HPC)
  - Consequent, increased fileystem size
  - New back up procedure needed
- Existing tree-walk algorithm

Project Motivation

- Increased demand for High Performance Computing (HPC)
  - Consequent, increased fileystem size
  - New back up procedure needed
- Existing tree-walk algorithm
  - Build filesystem paths

Project Motivation

- Increased demand for High Performance Computing (HPC)
  - Consequent, increased filesystem size
  - New back up procedure needed
- Existing tree-walk algorithm
  - Build filesystem paths
  - Substantial filesystem I/O

Project Motivation

- Increased demand for High Performance Computing (HPC)
  - Consequent, increased filesystem size
  - New back up procedure needed
- Existing tree-walk algorithm
  - Build filesystem paths
  - Substantial filesystem I/O
- **Project objective–design scalable algorithm**

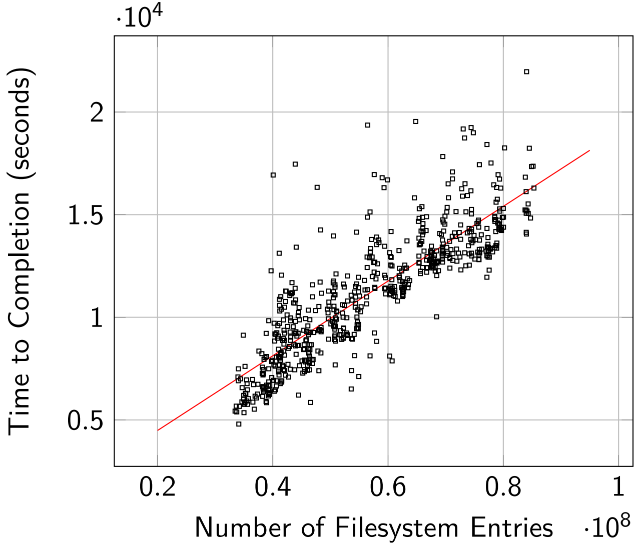
Project Motivation

- Increased demand for High Performance Computing (HPC)
  - Consequent, increased fileystem size
  - New back up procedure needed
- Existing tree-walk algorithm
  - Build filesystem paths
  - Substantial filesystem I/O
- **Project objective–design scalable algorithm**
  - Reduce filesystem access

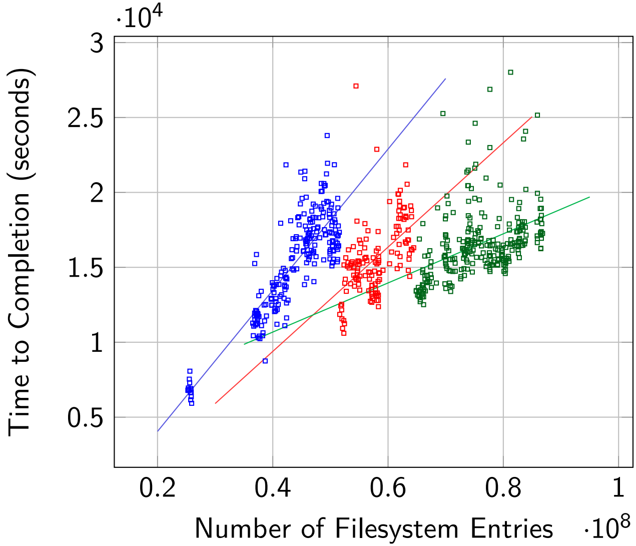
Project Motivation

- Increased demand for High Performance Computing (HPC)
    - Consequent, increased fileystem size
    - New back up procedure needed
- Existing tree-walk algorithm
    - Build filesystem paths
    - Substantial filesystem I/O
- **Project objective–design scalable algorithm**
    - Reduce filesystem access
    - Utilize Hadoop parallel framework

Project Motivation



Project Motivation



Hadoop and MapReduce

Hadoop and MapReduce

- MapReduce paradigm

Hadoop and MapReduce

- MapReduce paradigm
    - Mappers

Hadoop and MapReduce

- MapReduce paradigm
  - Mappers
  - Reducers

Hadoop and MapReduce

- MapReduce paradigm
  - Mappers
  - Reducers
- Scalability

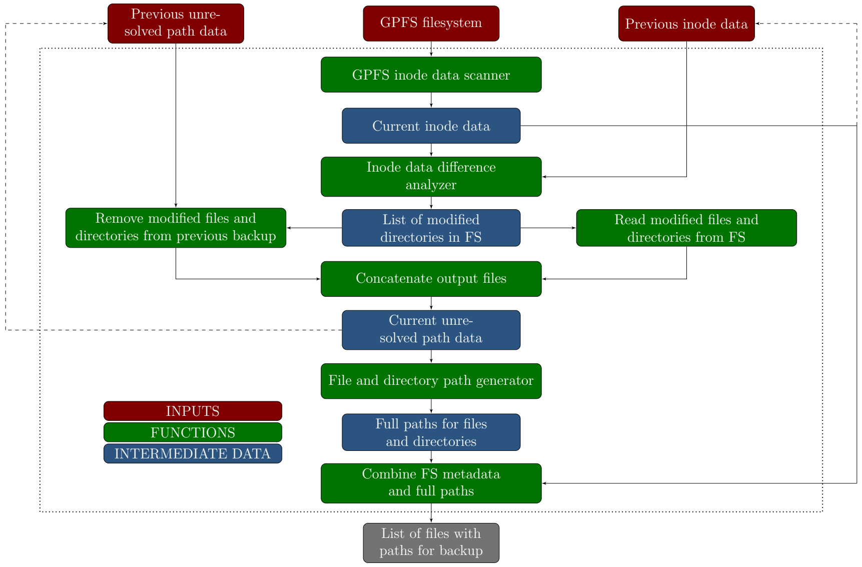
Hadoop and MapReduce

- MapReduce paradigm
    - Mappers
    - Reducers
- Scalability
    - Distributed Computing

Hadoop and MapReduce

- MapReduce paradigm
  - Mappers
  - Reducers
- Scalability
  - Distributed Computing
  - Parallelism

Algorithm Overview



Previous unresolved path data
GPFS filesystem
Previous inode data
GPFS inode data scanner
Current inode data
Inode data difference analyzer
Remove modified files and directories from previous backup
List of modified directories in FS
Read modified files and directories from FS
Concatenate output files
Current unresolved path data
File and directory path generator
INPUTS
FUNCTIONS
INTERMEDIATE DATA
Full paths for files and directories
Combine FS metadata and full paths
List of files with paths for backup

Algorithm Design

Algorithm 1 Path Generator Reducer

```
input: partitioned_mapper_output, filesystem_mount
 1: children_parent ← null
 2: for inode in partitioned_mapper_output do
 3:     if inode_flag is true then
 4:         children_parent ← inode_parent
 5:     else
 6:         inode_parent ← children_parent
 7:         write inode
 8:         if children_parent is not filesystem_mount then
 9:             report iterated_algorithm
10:         end if
11:     end if
12: end for
```

Algorithm Example

root/
parent/
child.txt

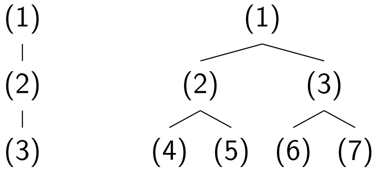
Algorithm Example

root/
root/parent/
parent/child.txt

Algorithm Example

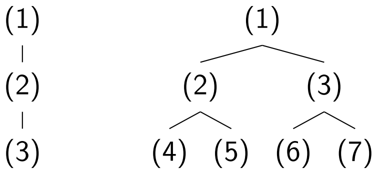
root/
root/parent/
root/parent/child.txt

Performance Metrics



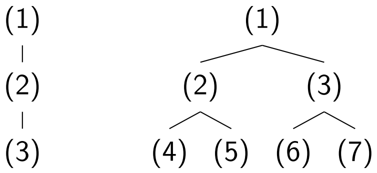
Performance Metrics

- Measuring success



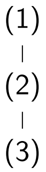
Performance Metrics

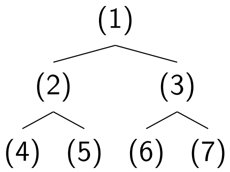
- Measuring success
    - Time to completion



Performance Metrics

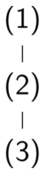
- Measuring success
  - Time to completion
  - Uniform artifical filesystems

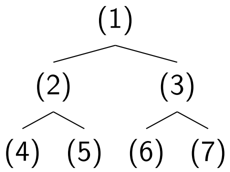




Performance Metrics

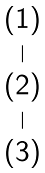
- Measuring success
  - Time to completion
  - Uniform artifical filesystems
- Limitations

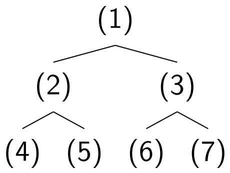




Performance Metrics

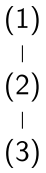
- Measuring success
  - Time to completion
  - Uniform artifical filesystems
- Limitations
  - Data noise

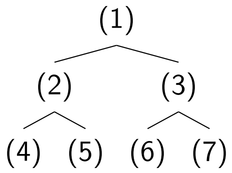




Performance Metrics

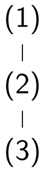
- Measuring success
  - Time to completion
  - Uniform artifical filesystems
- Limitations
  - Data noise

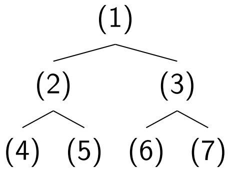




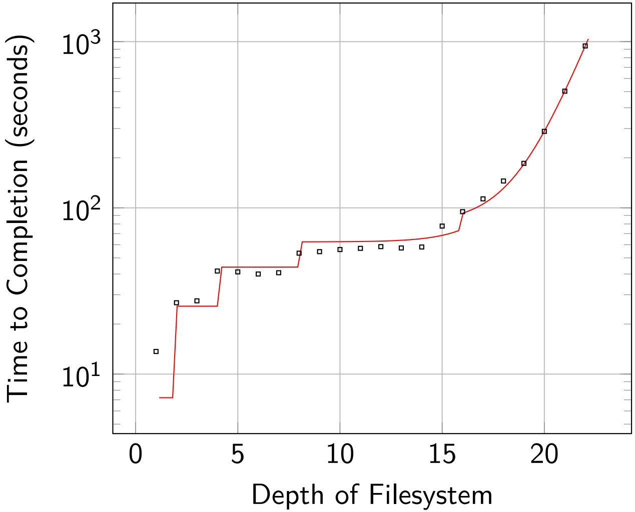
Performance Metrics

- Measuring success
  - Time to completion
  - Uniform artifical filesystems
- Limitations
  - Data noise
  - Machine dependent

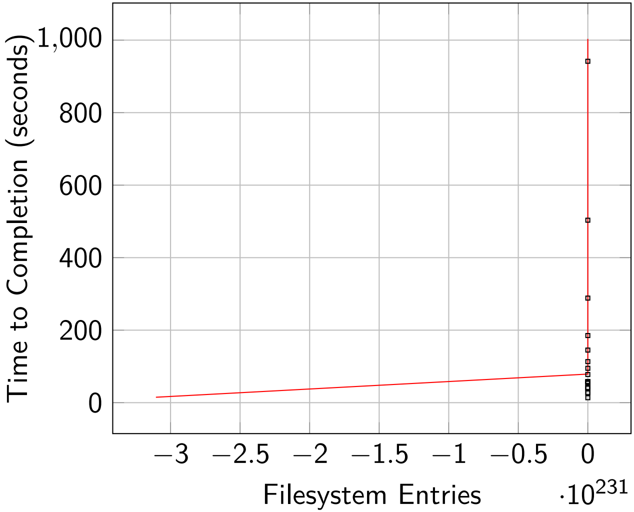




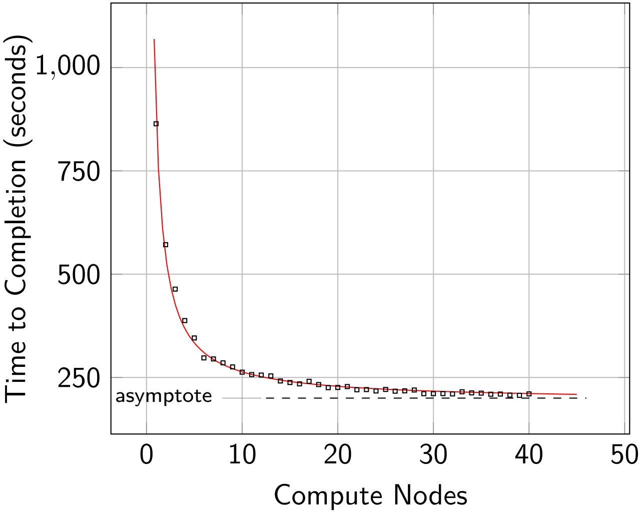
Time Complexity



Time Complexity



Algorithm Scaling



Conclusions

Conclusions

- Scalability

Conclusions

- Scalability
    - Expands to multiple nodes

Conclusions

- Scalability
    - Expands to multiple nodes
    - Inverse effect

Conclusions

- Scalability
  - Expands to multiple nodes
  - Inverse effect
- Cost effectiveness

Conclusions

- Scalability
  - Expands to multiple nodes
  - Inverse effect
- Cost effectiveness
  - Uses utilized resourses

Conclusions

- Scalability
  - Expands to multiple nodes
  - Inverse effect
- Cost effectiveness
  - Uses utilized resourses
  - Resources can be yielded back

Conclusions

- Scalability
  - Expands to multiple nodes
  - Inverse effect
- Cost effectiveness
  - Uses utilized resourses
  - Resources can be yielded back
- User interference

Conclusions

- Scalability
  - Expands to multiple nodes
  - Inverse effect
- Cost effectiveness
  - Uses utilized resourses
  - Resources can be yielded back
- User interference
  - Minimal filesystem access

Future Work

Future Work

- Refactor code in Java

Future Work

- Refactor code in Java
- Run Filter and Read Directories concurrently

Future Work

- Refactor code in Java
- Run Filter and Read Directories concurrently
- Hama implementation of File Path Generator

Acknowledgements

- Team for Research in Ubiquitous Secure Technology (TRUST)
- National Science Foundation (NSF award number CCF-0424422)
- Matthew Andrews–NERSC (Mentor)
- Jason Hick–NERSC (Mentor)