# MapReduce in MPI for Large-scale Graph Algorithms

Steven J. Plimpton, Karen D. Devine
Sandia National Laboratories
Albuquerque, NM
sjplimp@sandia.gov

Keywords: MapReduce, message-passing, MPI, graph algorithms, RMAT matrices

**Abstract**

new MR library: freely available as open-source in-core or out-of-core C++ library (also callable from C, Fortran, Python) runs on top of MPI how to do MR operations in distributed-memory MPI context how to handle out-of-core issues to enable large data sets how to express various graph algs as MR algorithms side issue: how to generate artificial problems at scale via RMAT performance/scalability for range of data set sizes where possible, compare performance to: other distributed-memory algs other machines Hadoop

# 1 Introduction

what is MR why implement in MPI our motivation = graph algorithms, large data sets but lib can also be used for any MR algorithm, data agnostic novel features of our MR implementation C++ lib, open-source, MPI, out-of-core no fault tolerance

# 2 MapReduce in MPI

keys and values, KV and KMV data structures map and reduce are naturally parallel on distributed KV, KMV user owns data, just set of bytes passed to library flavors of map() collate is equivalant to all2all() on distributed hash table how we do irregular communication big difference is we are doing the comm in syncronous manner data is reorganized locally, once it is all on-proc other related operations: compress, gather, collapse, sort

# 3 Out-of-core Issues

what part needs to be out-of-core in MPI context each proc does it's own disk IO basic idea: modified data structures for KV and KMV map() operation aggreate() operation convert() operation reduce() operation

# 4 Graph Algoriths in MapReduce

RMAT generation, pagerank, CC, triangle-finding, Luby, SSSP, other? how many variants of something like CC do we want to include? I would vote for just one discuss possible inefficiency due to a giant CC in CC algorithm

# 5 Performance Results

ideally, for each problem: define chosen characteristics of graph, e.g. edges/vertex choose 3 sizes: small = fits on 1 proc) medium = out-of-core on 1 proc, but not on P procs huge = impressive size show performance plots for each size running on 1 to P procs would like to see kink in plots as go out-of-core be able to judge and explain basic scaling results what machine: Odin or Tbird or RStorm? something that maps to a vanilla cluster where we have data: show multiple machines alternate non-MR algorithms discuss alternate CC algorithms via Trilinos and hybrid alternate MR implementations (Hadoop)

# 6 Acknowledgements

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Hadoop folks.