An Evaluation of the Zoltan Parallel (Hyper-)Graph Partitioners

Erik G. Boman and Siva Rajamanickam

10th DIMACS Challenge,

Atlanta, Feb. 2012

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.

NNSA National Nuclear Security Administration

Sandia National Laboratories

Models and Objectives

- Graph edge cut (EC)
  - Classic model, lots of software, but has limitations
  - Inaccurate representation of communication in parallel computing
  - Requires symmetric/undirected graph
- Graph communication volume (CV)
  - CV-sum: total comm. volume
  - CV-max: max comm. volume for any part (process)
- Hypergraph edge cut
  - One hyperedge for each vertex, includes all nbors
  - Exactly CV-sum

Partitioning Software

Lots of good partitioners available.

Focus on software used in scientific computing.

Software	Graph	Hypergraph	Parallel
Chaco	X		
(P)Jostle	X		X
(PT)Scotch	X		X
(Par)Metis	X		X
Patoh		X	
Mondriaan		X	
Zoltan	X*	X	X
Ka(FF)PPa	X		X

Sandia National Laboratories

Zoltan

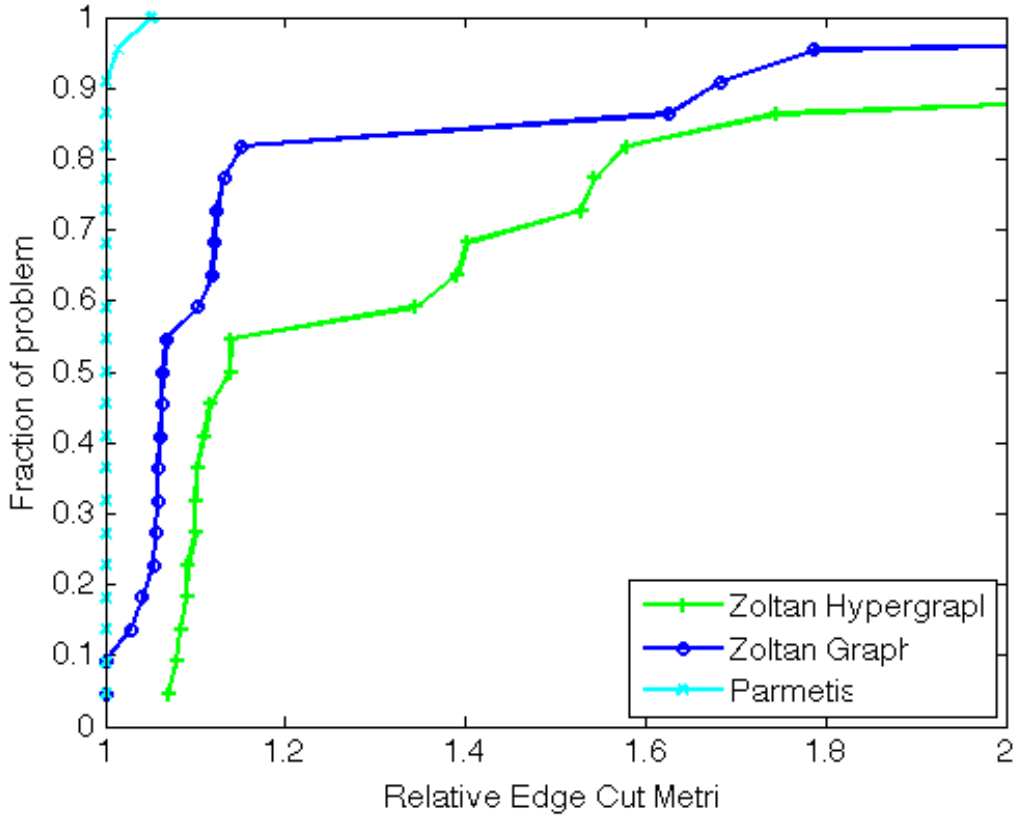
- Parallel toolkit for load balancing and combinatorial scientific computing
  - Also a Trilinos package
- Contains several native partitioning algorithms
  - And interfaces to others as 3rd party libraries
- We focus on PHG (Parallel Hypergraph and Graph Partitioner)
  - Was designed as a hypergraph partitioner
  - Graph partitioning is supported by treating each edge as an hyperedge of size two

Sandia National Laboratories

Empirical Study

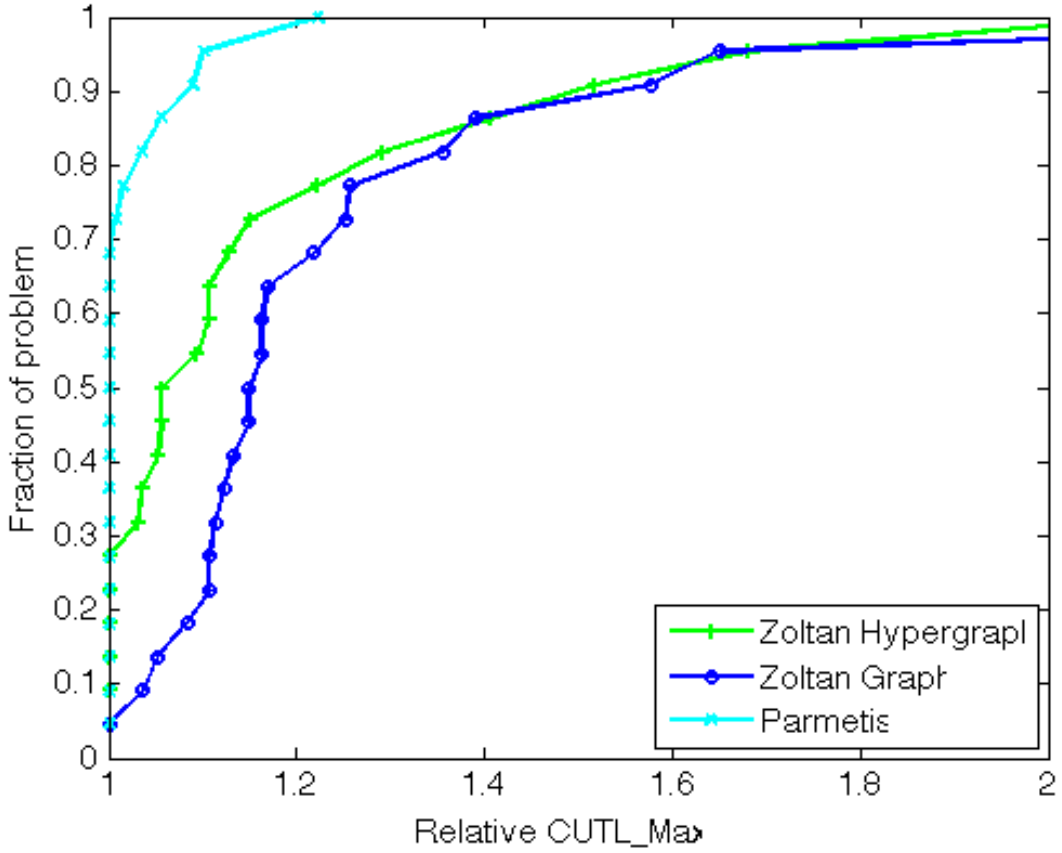
- Evaluate performance of Zoltan both as graph and hypergraph partitioner
  - Default parameters, “out-of-box”
- Use subset of DIMACS challenge data
  - Selected 22 graphs that we deemed relevant to scientific computing
  - 7 families represented
  - Excluded random and scale-free graphs
  - Picked large graphs suitable for parallel computing
- Parallel test platform
  - Hopper Cray XE6, 24 cores per node (#8 in Top500)

Edge cut (EC)



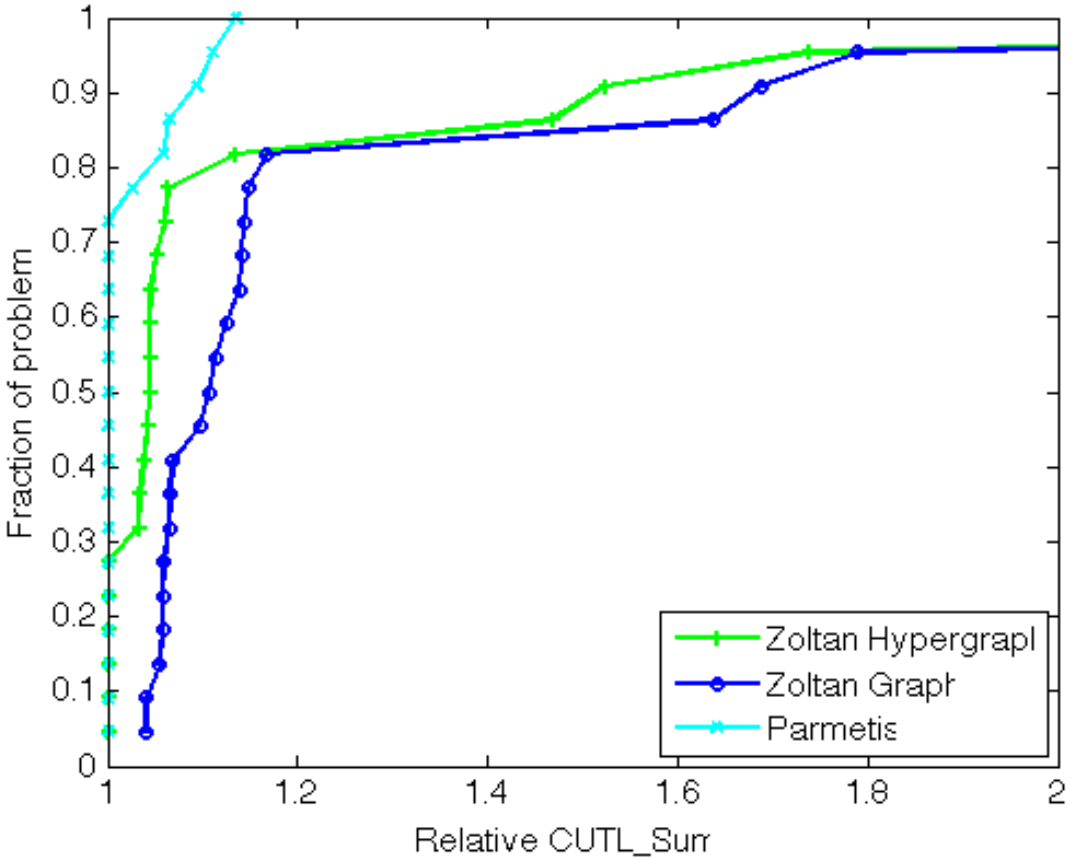
Graph partitioner (ParMetis) wins, as expected

Comm. volume (CV-max)



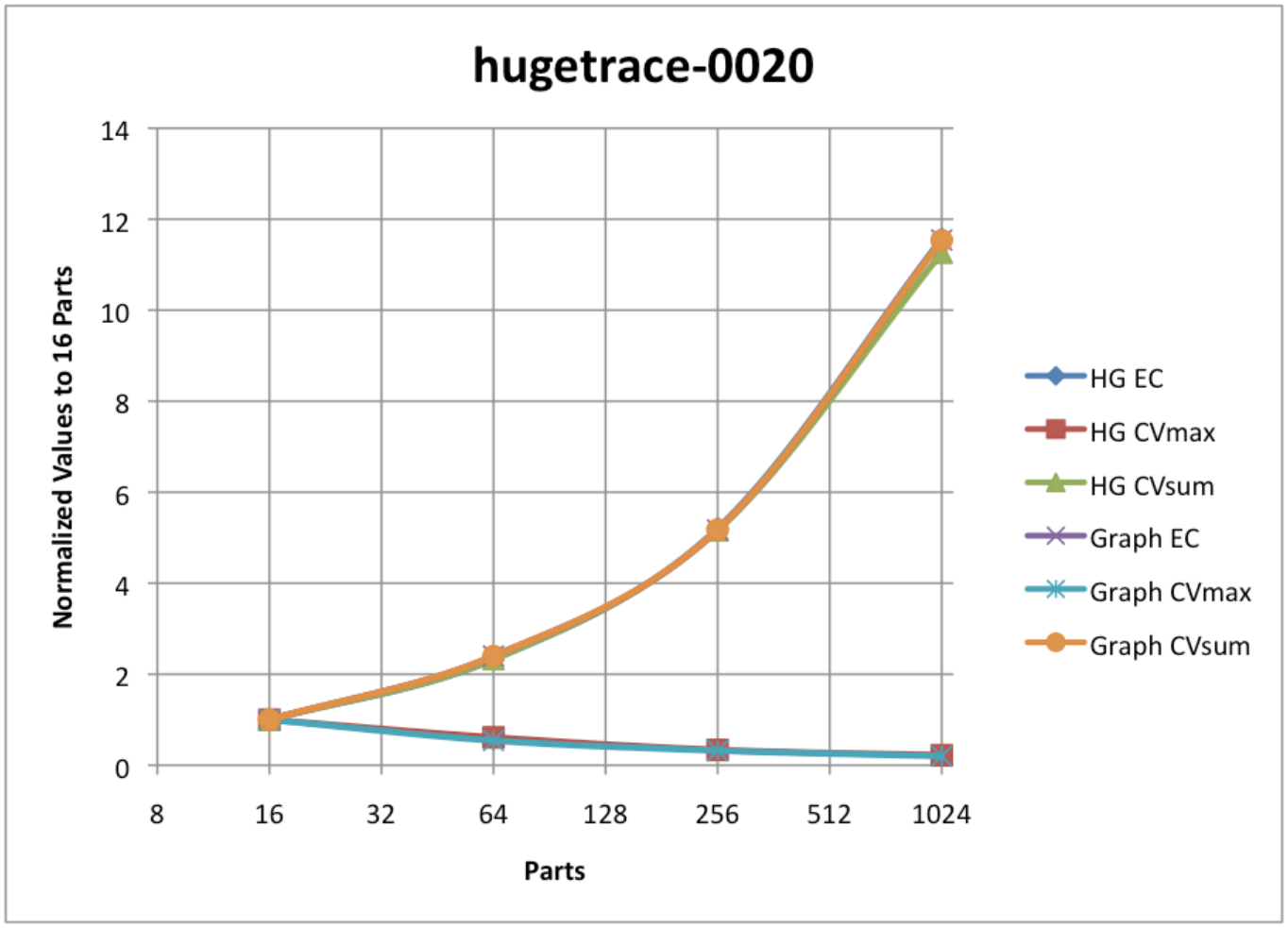
No partitioner optimizes this objective.
A bit surprisingly, ParMetis wins again.

Comm. volume (CV-sum)

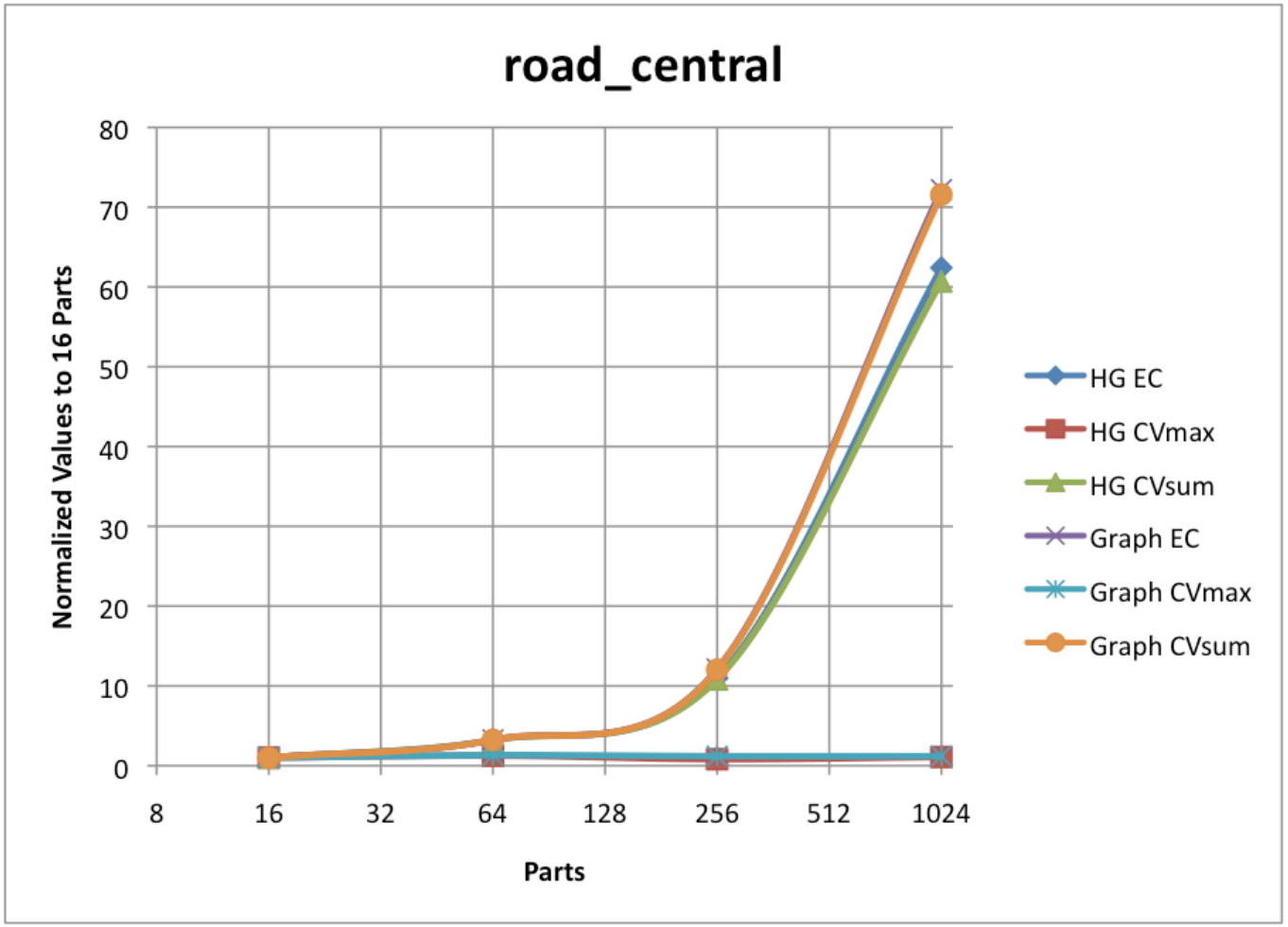


This time, a close race .Right metric for hypergraphs, but ParMetis takes advantage of symmetry.

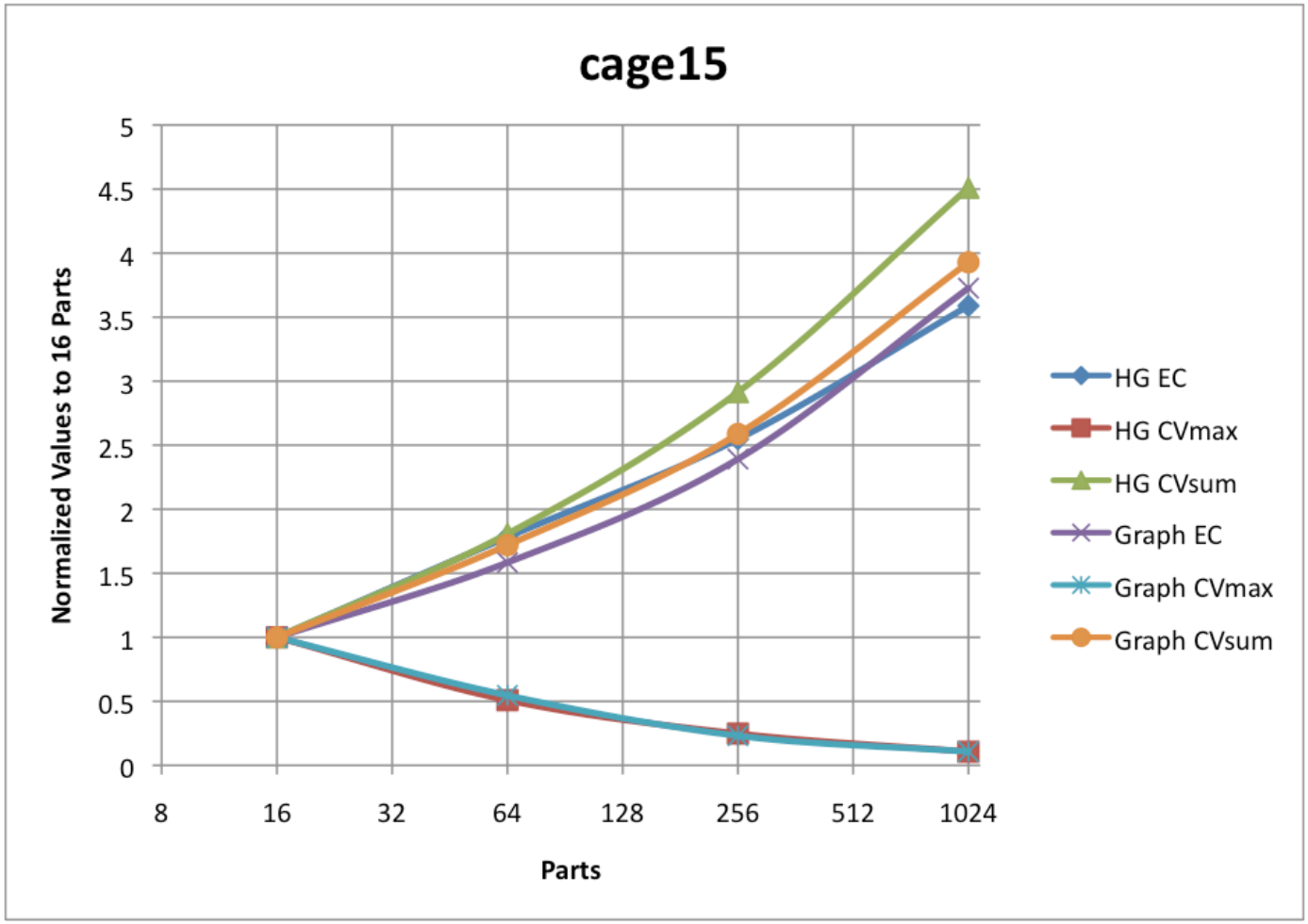
Cut Quality Scalability



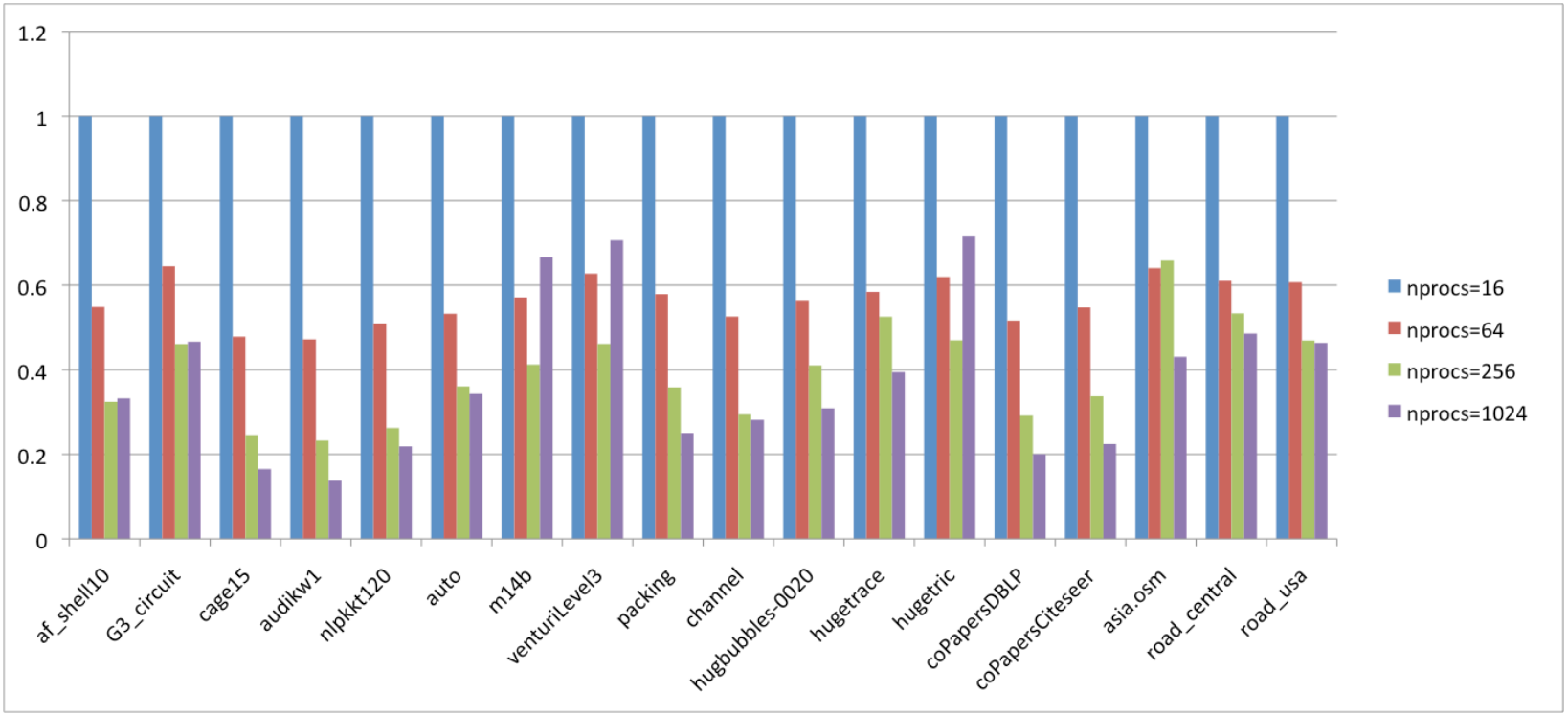
Cut Quality Scalability



Cut Quality Scalability



Zoltan HG Scalability (time)



Impact on Parallel Codes

- GMRES in Trilinos
  - Epetra distributed matrix
  - Mat-vecs dominate
  - No preconditioner
- Choice of metric/ partitioner makes little difference
  - Any partitioner better than block will do
  - Bad news for partitioning research?

Matrix	Block	ParMetis	Zoltan HG
audikw1	8.81	3.22	3.01
nlpkkt120	8.45	6.18	6.00
G3_circuit	2.76	1.59	1.55
af_shell10	2.61	2.74	2.79

Real data is NOT always symmetric

- Nonsymmetric A
  - Symmetrize for graph partitioners
  - Use A+A’ (square case)
  - Use AA’ or A’A (rectangular case)
- Usually better to partition A directly
  - Never need to form A+A’ or A’A
  - Need hypergraph partitioner

Nonsymmetric Ex. 1

- A is nonsymmetric
- Compare graph partitioning on (A+A') and hypergraph on A
- Measure cuts on A

Web-Stanford	H.P. on A	G.P. on A+A'
CV-max (A)	267	2,768
CV-sum (A)	2,020	21,858

Nonsymmetric Ex. 2

- A is rectangular term-document matrix
- Suppose we want a row partitioning of A
- Compare two options:
  - Graph partitioning on AA'
  - Hypergraph partitioning on A
- Ex: tbdlinux (113k x 20k, 2M nonzeros)
  - Note: AA' is too dense, so must filter it

tbdlinux	H.P. on A	G.P. on AA'
CV-max (A)	5,063	17,080
CV-sum (A)	38,970	132,210

Sandia National Laboratories

Conclusions

- Zoltan is a good general-purpose partitioner
  - CV close to ParMetis on symmetric problems
  - Real benefit is for nonsymmetric problems
- Zoltan is decent also as a graph partitioner
  - Does not exploit symmetry, so not as good as ParMetis
- Zoltan shows good scalability up to ~1K cores
  - Cut quality degrades slowly
  - Partitioning time decreases (though not linearly)
- In real apps, any reasonable partitioner will do
  - Optimizing the “right” metric isn’t necessary
- Suggestion for future DIMACS partition challenge:
  - Please include nonsymmetric problems (digraphs or hypergraphs)