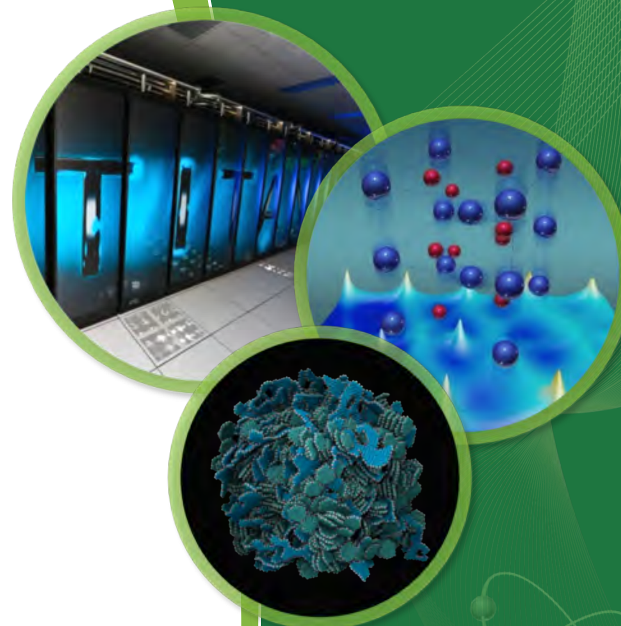
Scalable, Automated Characterization of Parallel Application Communication Behavior

Philip C. Roth
Computer Science and Mathematics Division
Oak Ridge National Laboratory

12th Scalable Tools Workshop



ORNL is managed by UT-Battelle
for the US Department of Energy

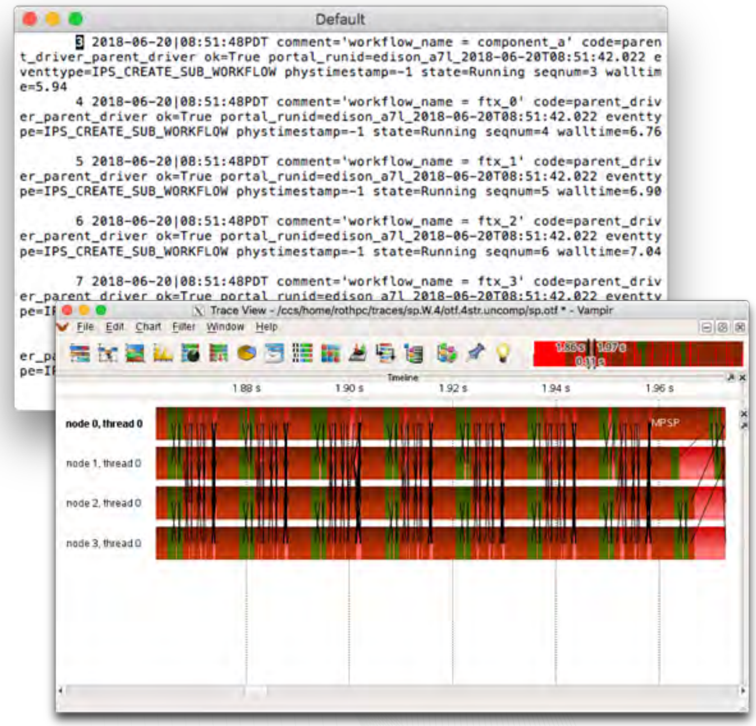
OAK RIDGE
National Laboratory

Motivation

- Often given unfamiliar application and asked to:
  - Describe how it works
  - Improve performance/scalability
- Helps to have high-level view of how processes communicate

Motivation

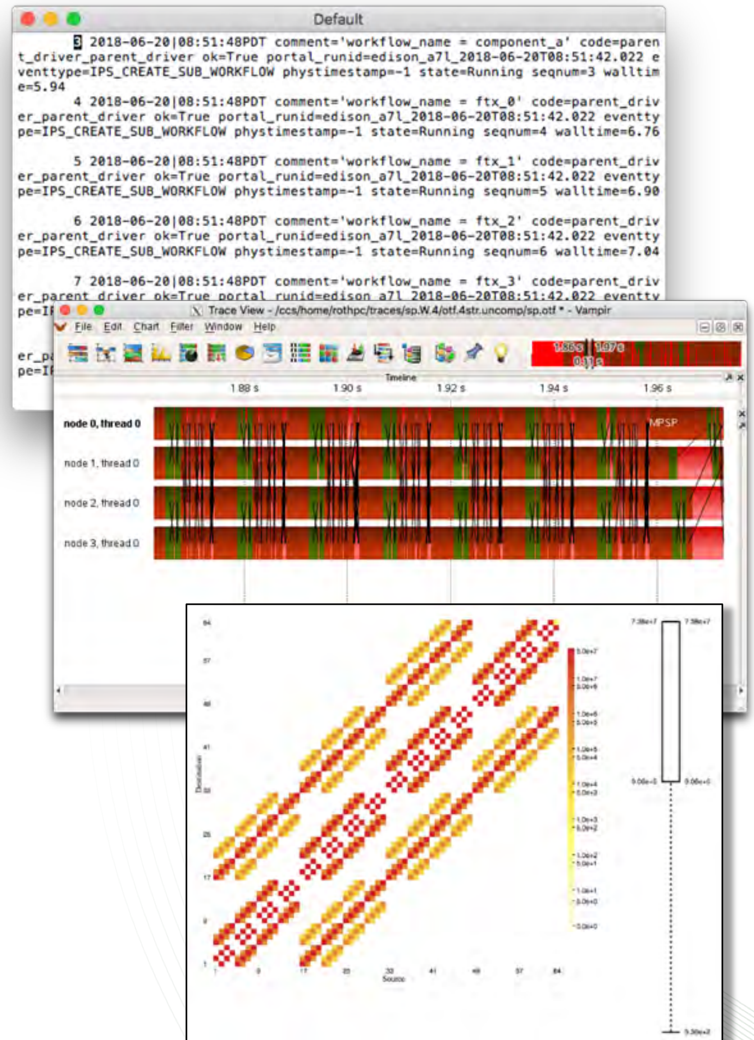
- Often given unfamiliar application and asked to:
  - Describe how it works
  - Improve performance/scalability
- Helps to have high-level view of how processes communicate
- Event traces and timeline visualizations → too much detail



RAPIDS OAK RIDGE National Laboratory

Motivation

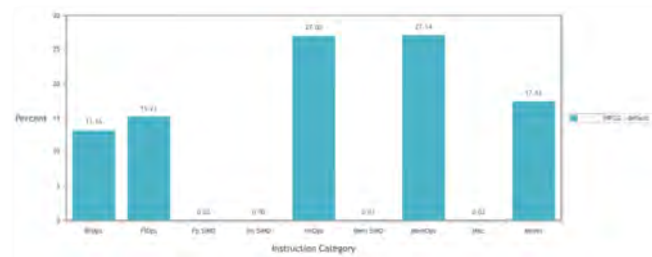
- Often given unfamiliar application and asked to:
  - Describe how it works
  - Improve performance/scalability
- Helps to have high-level view of how processes communicate
- Event traces and timeline visualizations → too much detail
- Communication matrix visualization → hard to interpret



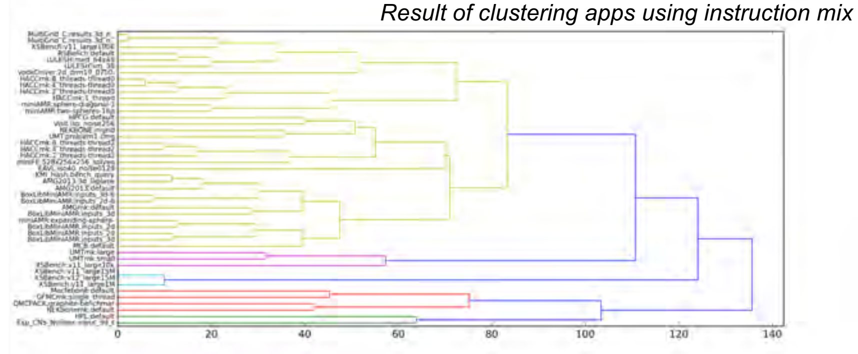
Background: Oxbow

- Characterize application demands independent of performance
  - System design
  - Representativeness of proxy apps
- Characterization on several axes:
  - Computation (instruction mix)
  - Memory access (reuse distance)
  - *Communication (topology, volume)*
- Online database for results with web portal including analytics support
- Project is dormant

Instruction Mix, HPCG, 64 processes



Result of clustering apps using instruction mix



AChax: Automated Communication Pattern Characterization

- Goal: capture communication pattern recognition expertise in an automated tool
- Given data describing application communication behavior, recognize communication pattern(s) and scale(s) that best account for observed data
- Express recognized patterns as parameterized expression

$$
\begin{aligned}
C_{LAMMPS} = {} & 13354 \cdot Broadcast(root:0)+ \\
& 700 \cdot Reduce(root:0)+ \\
& 19318888 \cdot 3DNearestNeighbor(\\
& \quad dims:(4,4,6), \\
& \quad periodic:True)
\end{aligned}
$$

Inspiration I: Paradyn's Performance Consultant

- **Automated search** through a **space** to find "point" that best explains observed performance
- **Hypothesize, test,** and **refine**
- Record results in a **search tree**

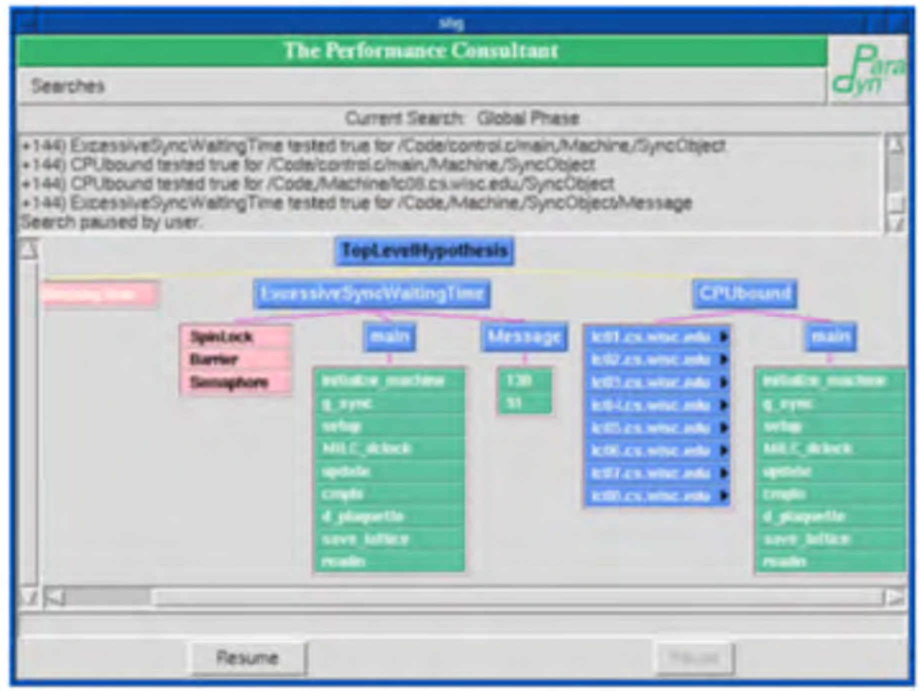
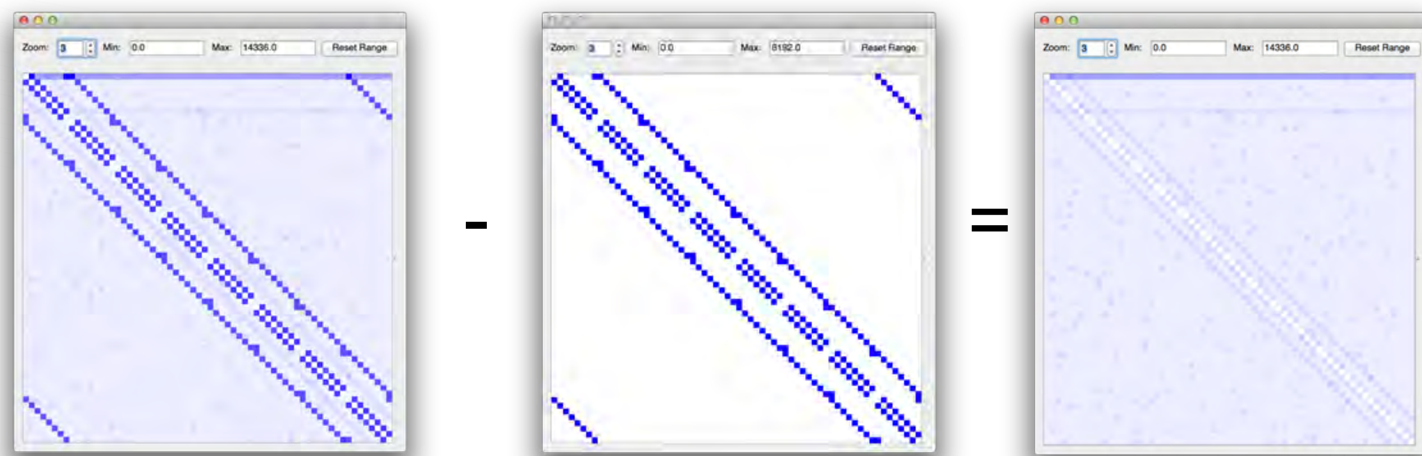


Figure 1.2 A Search History Graph display.

Inspiration II: Sky Subtraction

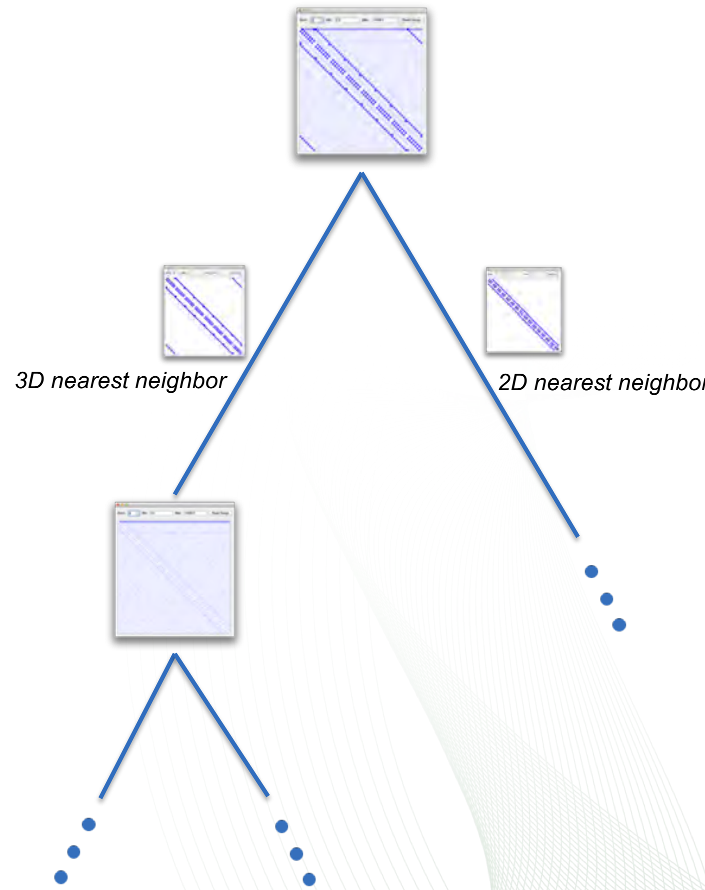
- Given an image of the sky, remove the known to make it easier to recognize the unknown



*Recognizing and removing the contribution of a 2D nearest neighbor pattern in a synthetic communication matrix. This represents **one step** in a search-based approach.*

Search Overview

- Associate application's communication matrix with root node
- At root node, for each pattern in **pattern library**
  - Attempt to recognize pattern in node's matrix
  - If recognized, subtract scaled pattern from node's matrix to get child matrix
  - Add child node with new matrix and edge to **search result tree**
  - Recursively apply search starting at child node

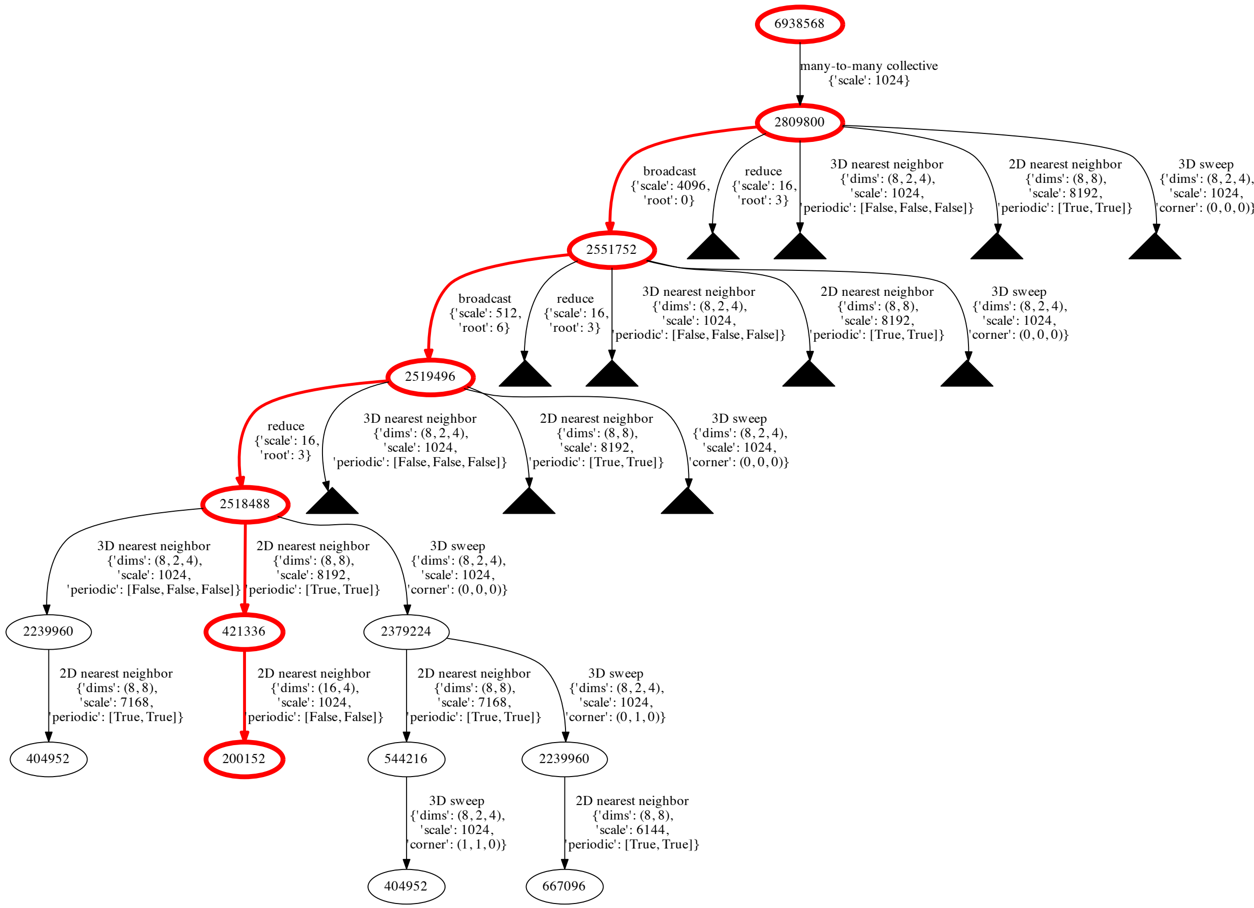


Pattern Recognition

- Library of scale-independent pattern generators and recognizers
- When attempting to recognize a pattern in a matrix
  - Determines number of processes
  - Determines dimension sizes for multidimensional patterns
  - Determines **scale** of the pattern
  - Determines root process for rooted collectives
  - Detects origin corner for wavefront patterns
- Heuristics for lightweight checks when possible

Search Result

- **Residual**: total communication volume in a communication matrix
- When search finishes, path between root and leaf with smallest residual indicates patterns that best explain original communication matrix

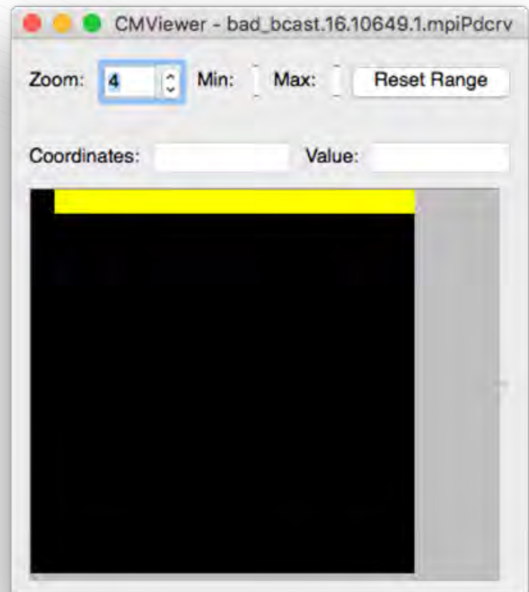


Three Problems

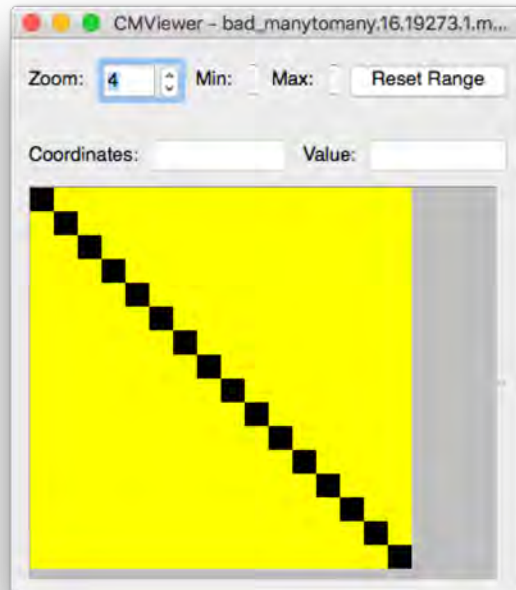
- Ambiguity in pattern recognition
- Greedy recognition approach can be too greedy
- Inefficient implementation

Problem 1: Pattern Recognition Ambiguity

- Representing communication data using traditional communication matrix leads to ambiguity, especially with collectives



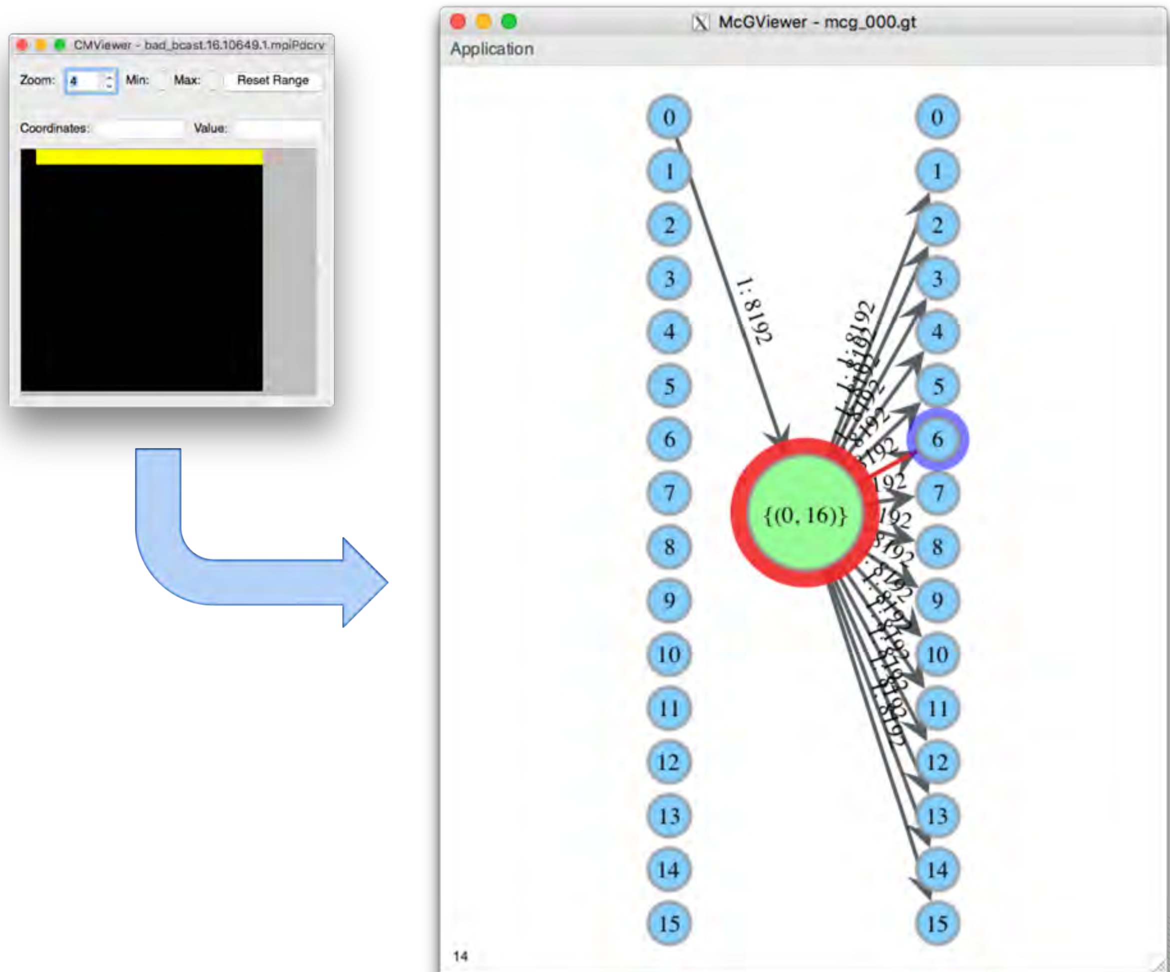
Broadcast or multiple point-to-point?



Worst case

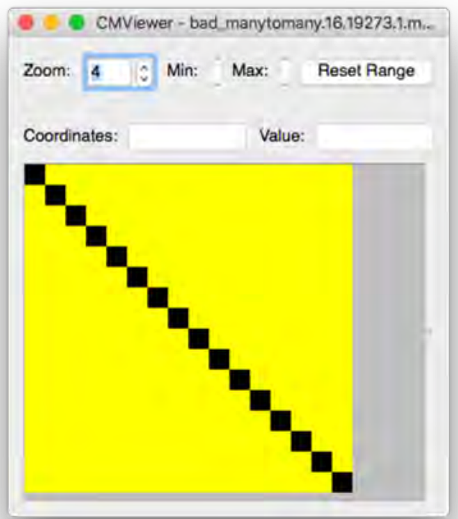
Augmented Communication Graphs (ACGs)

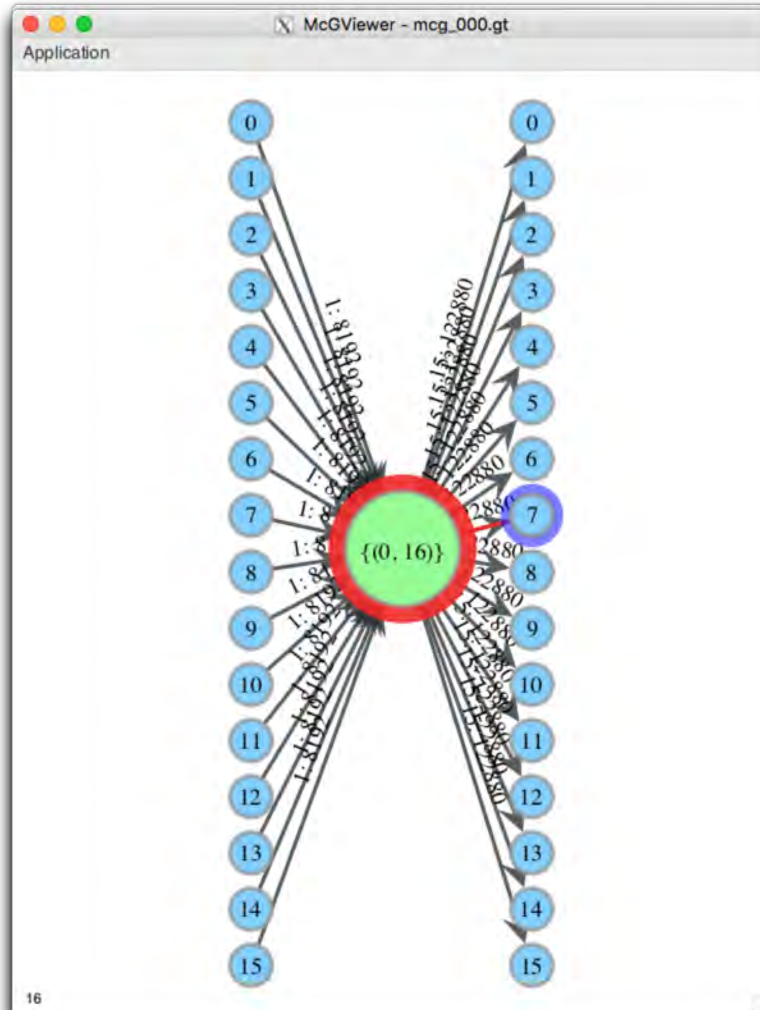
- Instead of traditional communication **matrix,** represent communication data as a graph
- Vertices for processes
  - Separate sender/receiver roles
- Edges denote communication occurred
  - Labeled with operation count and message volume
- To make it easier to discern collective operations, augment the graph with vertices representing communicators



And That Worst Case?

- As presented so far, better but not ideal
- May need to label communicator vertices with collective operation or operation type

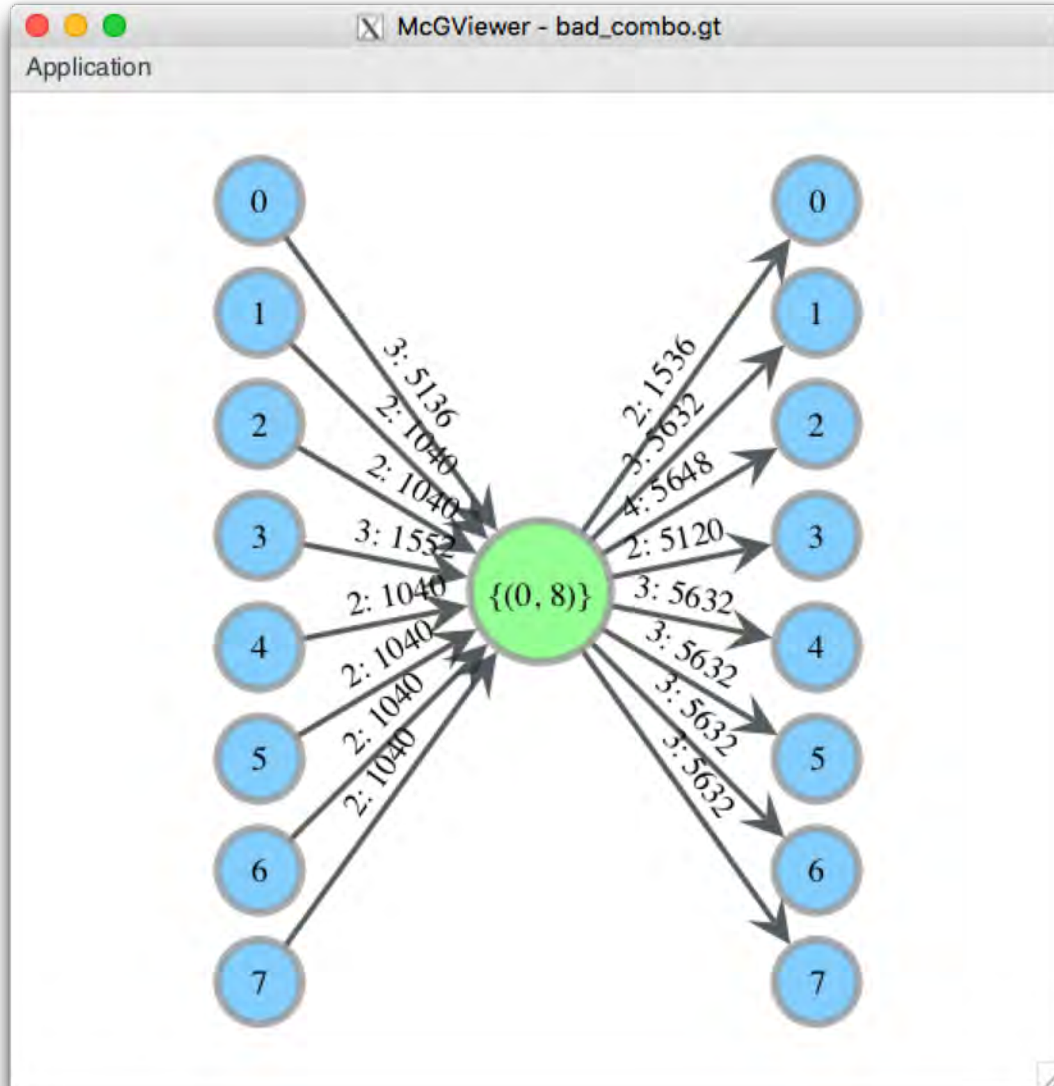




Problem 2: Too Greedy

- When recognizing a pattern, AChax recognizes as much data as possible for that pattern
- Can cause automated search to fail to recognize some pattern combinations

– broadcast: {’scale’: 4096, ’root’: 0}
– broadcast: {’scale’: 512, ’root’: 3}
– reduce: {’scale’: 16, ’root’: 2}
– many-to-many: {’scale’: 1024}

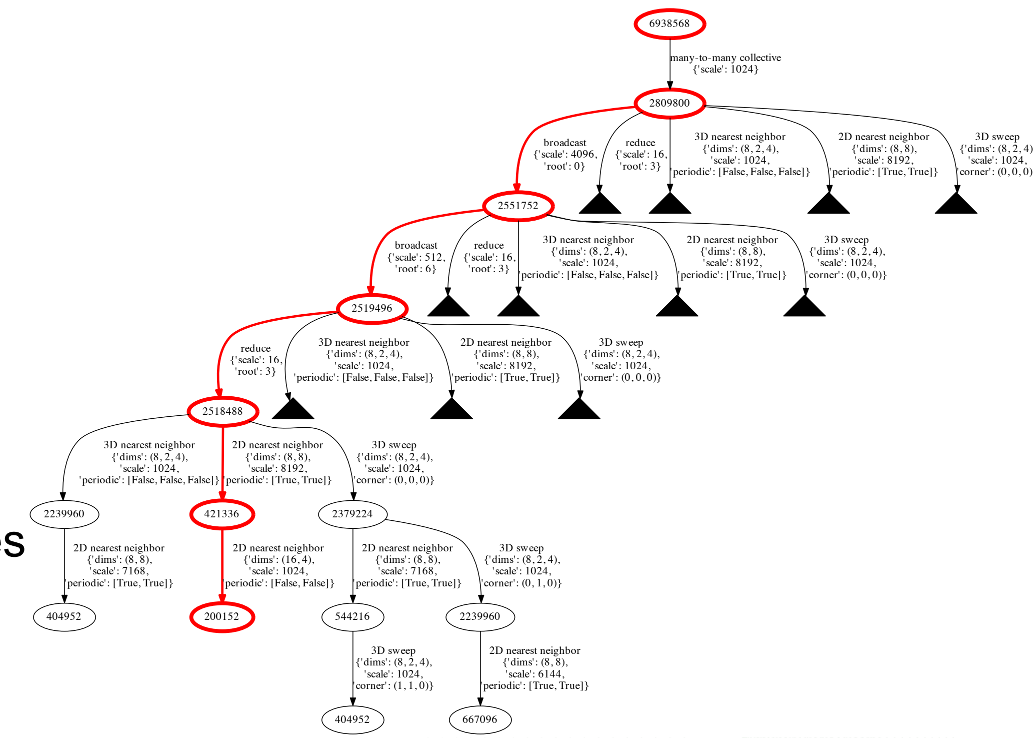


Non-Greedy Pattern Recognition

- If pattern recognized, check if removing pattern with maximum scale will result in invalid ACG
- If so, find smaller scale(s) and refine search at each
- Problem: if pattern recognized at maximum scale *S*, can be recognized for every integer scale between 0 and *S*
  - *Search space explosion*
- Instead, find “interesting” scale values
- Heuristic based on communication count differences on ACG edges
  - *Current implementation may still refine at large number of scales*

Problem 3: Inefficient Search

- Original AChax implementation susceptible to doing lots of redundant work
- E.g., pattern combination from original AChax paper
  - Search results tree has 506 nodes
  - 180 leaves ("best" for given search refinement)
  - Only ***3*** distinct residual values in leaves
- Instead, prune search when root→node path is permutation of another root→node path

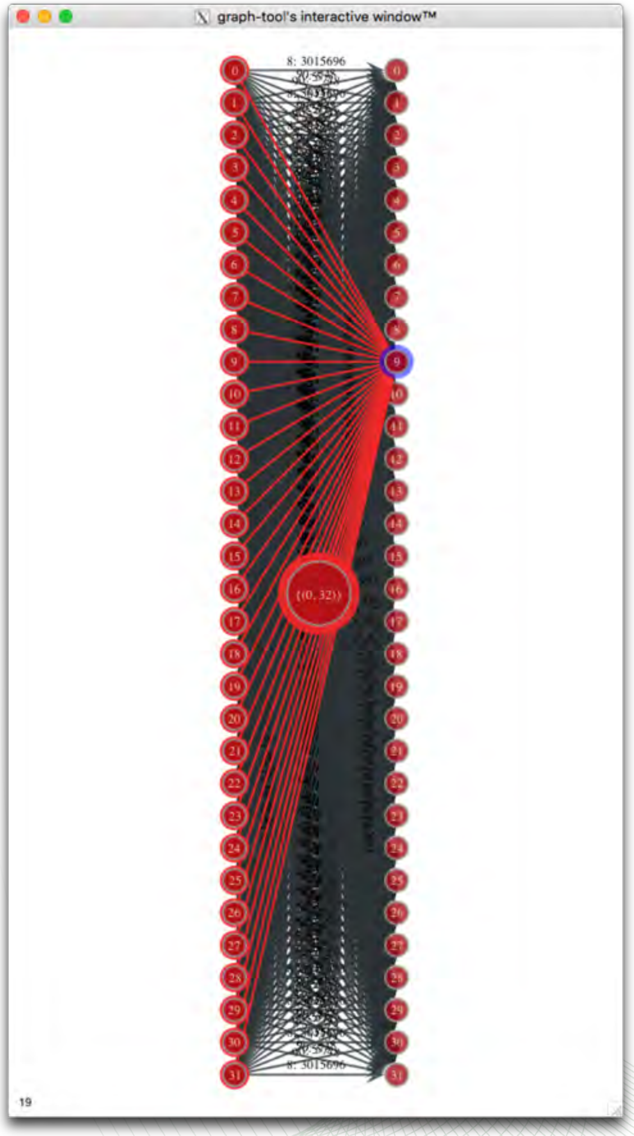


Implementation

- Original AChax tool
  - Python, using NumPy and SciPy for matrix ops and I/O
  - MatrixMarket format for communication matrix files
- AChaxG – ACG-based tool
  - Still Python
  - Graph-tool module for I/O, analysis, and visualization of ACGs
  - VERY slow ⇒ recently back to MatrixMarket representation of ACG
- Simple ACG viewer
  - Interactive, highlights edges to/from selected nodes
- Grabber: MPI communications data capture library
  - C++ with Boost and Todd Gamblin's MPI wrapper generator

Case Study: Xolotl

- Plasma surface interactions model
  - C++, MPI, PETSc
- Ran on OLCF Eos Cray XC30
  - 1D problem, 2048 grid points
  - 32 processes, 5 time steps
- AChaxG recognized broadcast, reduce, and 1D nearest neighbor patterns – didn't account for much
- Interactive visualization exposed point-to-point collectives (eventually found within PETSc)



Lots Left to Do

- Handle patterns whose communication volume depends on specific sender/receiver pair
  - Statistical distributions instead of constant scales?
- Handle sub-communicators and tightly-coupled MPMD apps
  - Two-stage pattern recognition (identify subcommunicators then original search)?
- Handle apps that re-number ranks
- Explore alternative approaches
  - Optical pattern recognition with machine learning
  - Matrix optimization problem using traditional solver techniques
- Improve recognition performance (parallelization)
- Scalable graph viewer

Acknowledgements

- This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under contract number DE-AC05-00OR22725.

- This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

RAPIDS OAK RIDGE National Laboratory

Summary

- Developing automated communication pattern recognition to support debugging, optimization, system choice, system design
- Recently augmented automated communication pattern recognition approach to use:
  - Communication graphs augmented with information about collectives
  - Aggressive search space pruning
- Exploring alternatives: using statistical distributions, machine learning, optical pattern recognition, parallelization
- Publications
  - P.C. Roth, J.S. Meredith, J.S. Vetter, “Automated Characterization of Parallel Application Communication Patterns,” HPDC’15
  - P.C. Roth, “Improved Accuracy for Automated Communication Pattern Characterization Using Communication Graphs and Aggressive Search Space Pruning,” ESPT’17. Published as LNCS 11027 (to appear)

- For more information: rothpc@ornl.gov

RAPIDS OAK RIDGE National Laboratory