Abstract
We develop several classes of graph algorithms using linear-algebraic (GraphBLAS) primitives. In-house combinatorial BLAS library enabled rapid development of bipartite graph matching, reverse Cuthill-McKee ordering, triangle counting, connected components and Markov clustering algorithms that scale to thousands of cores on modern supercomputers. These algorithms in turn empower key science applications including protein family detection and sparse linear solvers.

Motivation

Why graphs? Graph computation drives many applications in biology and scientific computing.

Expected impacts:
(a) better understanding of data and computational patterns, (b) rapid development of high-performance applications.

Approach

We develop graph and machine learning algorithms using linear-algebraic primitives. Two thrusts:

1. Develop communication-avoiding and work-efficient primitives (Aydin’s poster)

2. Design algorithms using optimized primitives

Results

1) Distributed-memory graph matching

- A suite of parallel algorithms developed. Sparse matrix-sparse vector multiply, inverted index used.
- Scales to several thousands of cores

Impact: Remove the sequential-ordering bottleneck from SuperLU and STRUMPACK

Motivation

Why GraphBLAS?
The diversity and rapid evolution of applications, architectures & algorithms motivates us to isolate a small number of graph kernels entrusted with delivering high-performance.

Areas in which we need help

- Biology & other domains to understand applications
- Programming language and libraries (UPC, GASNet)
- Efficient FILE I/O (HDF5)

Areas in which we can help

- Areas needing high-performance graph computation:
  - Machine Learning
  - Computational Biology
  - Scientific computing
  - Quantum computing

Areas in which we need help

- Areas needing high-performance graph computation:
  - Biology & other domains to understand applications
  - Programming language and libraries (UPC, GASNet)
  - Efficient FILE I/O (HDF5)

References