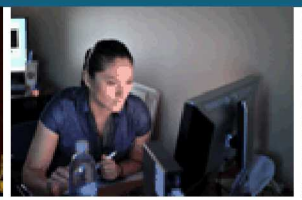
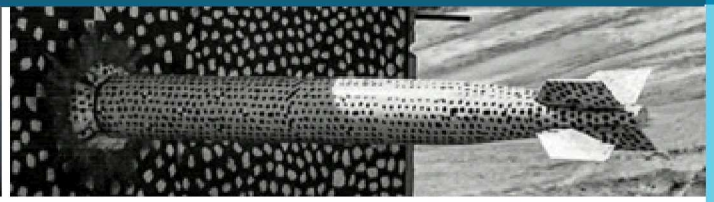
SAND2020-1350PE

Some Linear Algebra-based Graph Analytics using Chapel





Richard Barrett

Presented to the Cray Chapel team

February 5, 2020

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline

Project overview

Some graph preliminaries

Triangle enumeration

Mean hitting time

Chapel design and implementation ideas

- In comparison to existing C++ with serial, OpenMP, kokkos and MPI.

La Posta project

Chapel productivity analysis, with an emphasis on runtime performance, of some linear algebra based graph analytic algorithms.

Productivity is some measure of expressiveness, performance, portability, and robustness.

Staff capabilities: applications, programming models, performance, runtime systems, architecture.

One year project, aiming to expand to multi-year.

Some graph preliminaries

Graphs capture relationships (e.g. physical domains, data science)

Triangles show relationships between vertices.

Cliques combine those relationships.

Hitting time: expected time for a random walk between (possibly sets of) vertices.

Adjacency matrix: $a_{i,j} = 1$ when vertex i is adjacent to vertex j, i.e. connected by an edge; else $a_{i,j} = 0$.

Incidence matrix: $b_{i,j} = 1$ when vertex i and edge j are incident; else $b_{i,j} = 0$.

miniTri

is an application proxy for a class of triangle based data analytics, used in dense subgraph detection, characterizing graphs, improving community detection, and generating graphs.

Enumerates triangles with a calculation of specific vertex and edge properties.

“A Task-Based Linear Algebra Building Blocks Approach for Scalable Graph Analytics”, M. Wolf, J. Berry, and D. Stark, Proc. of 19th Annual IEEE High Performance Extreme Computing Conference, 2015.

https://github.com/Mantevo/miniTri

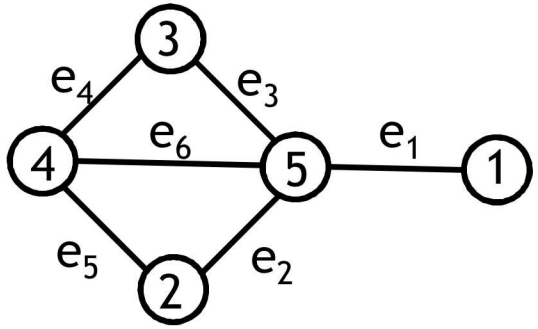
Triangle enumeration steps in miniTri

1) Create a list of triangle vertices: C = A * B, for adjacency matrix A and incidence matrix B.

2) Compute triangle vertex and edge degrees
   a) Number of triangle vertices, i.e. row sum(C): $t_v = C * 1$
   b) Number of triangle edges, i.e. column sum (C): $t_e = C^T * 1$

3) kcount (C, t_v, t_e), for constructing cliques (binomial formula)

Triangle enumeration in miniTri



$$A * B = C = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}$$

- Adjacency matrix: $a_{i,j}$ = 1 when vertex i is adjacent to vertex j.
- Incidence matrix: $b_{i,j}$ = 1 when vertex i and edge j are incident.
- Don't actually form matrix C. Instead, if $C_{i,j}$= 2, add the triangle vertices to list.

Triangle Degrees

$$C = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}$$

$$t_v = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix} * \mathbf{1} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}$$

Triangles with v_4

"Overloaded" SpMV to count triangles in rows

$$t_e = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{2,4,5\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{3,4,5\} \\ \emptyset & \{4,2,5\} & \{4,3,5\} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \{5,3,4\} & \{5,2,4\} & \emptyset \end{pmatrix}^T * \mathbf{1} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

Triangles with $e_{4,5}$

Triangle degree calculation

- Each triangle represented once in C for each of its edges and vertices
- Triangle vertex and edge degrees are number of nz in rows and columns

Computing mean hitting times using Grafiki

Sandia developed code for analyzing graphs on distributed memory parallel processing computers.

(formerly known as Tridata)

C++, with Kokkos for intra-node and MPI for inter-node data sharing.

Linear algebra via Trilinos.

Mean hitting time

Average time to traverse an undirected connected graph from one subset vertices to another subset of vertices, i.e. expected time for a random walk between vertices.

MHT reference implementation

Our exemplar

Configure as combinatoric graph Laplacian, solve using Jacobi preconditioned ***conjugate gradient method,*** to find the mean hitting times, stddev, skew, and kertosis.

- Based on Grafiki project research code written in MATLAB.
- Combinatoric graph Laplacian L(G)= D – A = row_sums (A) – A, i.e. D are vertex degrees.
- Four consecutive solves using Jacobi-preconditioned Conjugate Gradient method.
  - Operating on integers.
- Post-process solution approximations using lots of array notation and element-wise operations, resulting in mean, standard deviation, skew, and kertosis.

Conjugate gradient method solving A*x=b

For symmetric positive definite matrix A in R^{nxn}, x and b in R^{nx1}

```
r = b - A*x;
error = norm( r ) / bnrm2;
if ( error < tol ) return, end

for iter = 1:max_it

   z   = M \ r; Preconditioning.     Ax=b => M^-1Ax = M^-1b; Jacobi: M = diag(A)
   rho = (r'*z); inner product

   if ( iter > 1 ),
      beta = rho / rho_1;
      p = z + beta*p; vector update (daxpy)
   else
      p = z;
   end

   q = A*p; Matrix-vector product
   alpha = rho / (p'*q ); inner product
   x = x + alpha * p;  vector update (daxpy)

   r = r - alpha*q;    vector update (daxpy)
   error = norm( r ) / bnrm2;
   if ( error <= tol ), break, end

   rho_1 = rho;

end
```

Array notation operations

```
HTM(:,3) = HTM(:,3) - HTM(:,1).*( v2 + 2*HTM(:,2) );

HTM(:,4) = HTM(:,4) - HTM(:,1).*( 4*v3 - 3*HTM(:,1).* (v2 + HTM(:,2)) );

HTM(:,2) = sqrt( HTM(:,2) );

for j=3:4
   HTM(:,j) = HTM(:,j)./(HTM(:,2).^j);
```

Related work

The Stellar group@LSU, using HPX RTS.

Arkouda, python interface to Chapel, making use of numpy.

Chapel Hyper-graph library

GraphBLAS

Graph500

- Chapel implementation or other Chapel graph library?

GPU-connect?

Summary

Investigating the use of Chapel for some linear algebra based graph analytics algorithms.

Linear algebra uses meaningfully different from that used in traditional CS&E applications.

Performance analysis layers.

One year project, hoping to continue.

- Investigate different configurations, runtime options, algorithms, computing environments, …

References

A Task-Based Linear Algebra Building Blocks Approach for Scalable Graph Analytics, Wolf, Berry, and Stark, Proc. of 19th Annual IEEE High Performance Extreme Computing Conference, 2015.

Advantages to modeling relational data using hypergraphs versus graphs, Wolf, Klinvex, and Dunlavy, IEEE High Performance Extreme Computing Conference (HPEC), 2016. (Grafiki then named Tridata.)