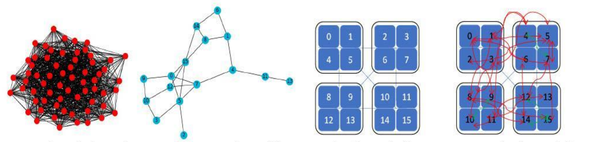
Simulating Application Agnostic Process Assignment for Graph Workloads on Dragonfly and Fat Tree topologies

[1]Md Nahid Newaz, [2]Sayan Ghosh, [3]Joshua Suetterlein, [4]Nathan R. Tallent, [5]Hua Ming

[1,5]Oakland University, [2,3,4]Pacific Northwest National Laboratory

OAKLAND UNIVERSITY
School of Engineering and Computer Science

Problem



- Iterating over vertices and edges of a graph is a common pattern
- Can lead to communication in the distributed-memory context as neighboring vertices can be owned by another process (some network hops away)
- Such communication is often considered “adversarial” due to incast flows (multiple messages converging towards the same endpoint)
- Real-world graphs are multifarious, process affinity changes as graphs are distributed over varied number of processes
- Relative affinity of the communicating processes affect overall performance
- Even with efficient partitioning, insufficient process mapping/assignment can lead to arbitrary communication variability
- Default strategy for process assignment to processor cores usually follows a shared-memory (SMP) style block placement, i.e., consecutively numbered MPI process ranks are placed sequentially on the same node
- Default block placement may not be optimal for irregular applications

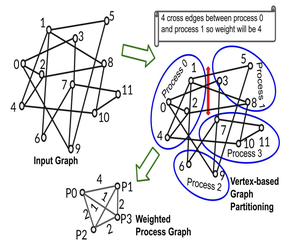
Contributions

- Existing solutions (e.g., CrayPAT) mostly caters to cartesian process topologies and not the graph topology
- We propose application and topology agnostic process remapping strategies for graph applications
- For two communication intensive distributed-memory graph applications (graph clustering and triangle counting), we demonstrate about 45% improvements
- Evaluated graph-traversal based process reorderings via SST-based packet-level simulations on Dragonfly and Fat Tree
- **Topology Agnostic:** One process order for multiple network topology
- **Application Agnostic:** One process order multiple application
- **One-time Rank Order calculation:** rank order can be generated one time and can be used multiple times as long as node*ppn = number of ranks
- Existing MPI implementations can adopt our heuristics (usually less than a second for thousands of processes, parallelized)
- Most useful for irregular memory intensive applications that use low PPN

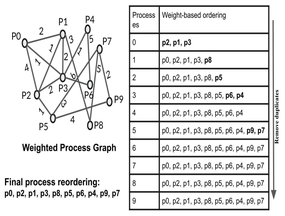
Approach

- We generate the rank order by exploring the consecutive vertex neighborhoods in the implicit process graph in parallel
- Scales to a large number of processes (return rank reorders under a second)
- Leverages process local neighborhood subgraph
- Number of cross edges in actual input graph considered as weights between adjacent vertices in the process graph
- New rank order is generated by traversing vertex neighbors according to the order of their weights and degrees
- We study weighted (natural order), weighted ascending and weighted descending order of degrees

Weighted Process Graph Generation



Process Reordering using Process Graph

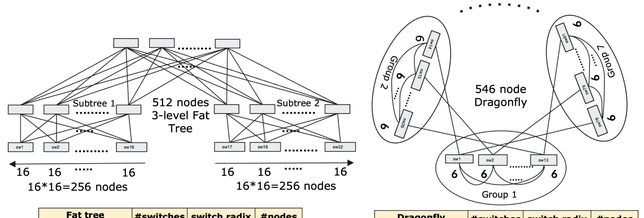


Process es	Weight-based ordering
0	**p2, p1, p3**
1	p0, p2, p1, p3, **p8**
2	p0, p2, p1, p3, p8, **p5**
3	p0, p2, p1, p3, p8, p5, **p6, p4**
4	p0, p2, p1, p3, p8, p5, p6, p4
5	p0, p2, p1, p3, p8, p5, p6, p4, **p9, p7**
6	p0, p2, p1, p3, p8, p5, p6, p4, p9, p7
7	p0, p2, p1, p3, p8, p5, p6, p4, p9, p7
8	p0, p2, p1, p3, p8, p5, p6, p4, p9, p7
9	p0, p2, p1, p3, p8, p5, p6, p4, p9, p7

Final process reordering: **p0, p2, p1, p3, p8, p5, p6, p4, p9, p7**

Remove duplicates

Simulation Setup

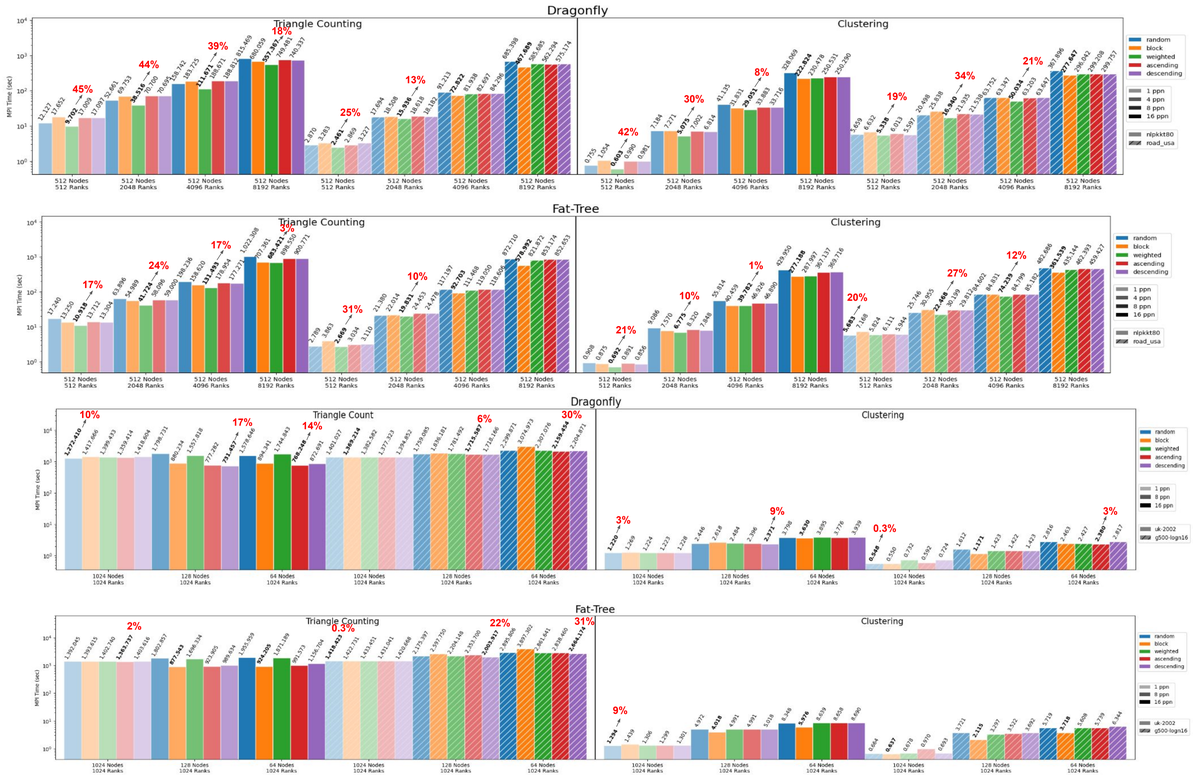


Fat tree	#switches	switch radix	#nodes
(3,32,2): 3-level pruned w 2 subtrees, 16 leaf and aggregate switches/subtree	80	32	512
(3,48,2): 3-level pruned w 2 subtree, 24 leaf and aggregate switches/subtree	120	48	1152

Dragonfly	#switches	switch radix	#nodes
(13,7,6,6): 13 switches/group, 7 groups, 6 nodes/switch, 6 global ports/switch	91	24	546
(17,9,8,8): 17 switches/group, 9 groups, 8 nodes/switch, 8 global ports/switch	153	32	1224

We used the packet-level parallel discrete event simulator Structural Simulation Toolkit, SST-Macro version 12.0

Results



Take-away messages and Future Work

- For smaller PPN our process graph property based remapping can perform better across multiple applications and network topologies
- With relatively small networks, increasing PPN keeping #processes fixed exploit only a small part of the topology which can be locally connected - in such cases performance will depend on the application and input graph characteristics
- Partitioning relatively small graphs on large #processes can thwart remapping heuristics (e.g., unspecific affinity, alltoall type pattern)
- Choice of PPN based on input graph and network topological properties critical for remapping performance
- Considering larger graphs and and topologies are relevant - must leverage distributed-memory simulation toolkits
- Results indicate that there is incentive for better heuristics (like separating high degree nodes by strides) to further improve the performance - also corroborate with actual performance on a supercomputer

U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

For additional information, contact:
Sayan Ghosh | (509) 375-6323 | sayan.ghosh@pnnl.gov

8/18/20 | IR # PNNL-SA-190071

Pacific Northwest NATIONAL LABORATORY
www.pnnl.gov