# A Parallel External-Memory Frontier Breadth-First Traversal Algorithm for Clusters of Workstations

*Robert Niewiadomski, José Nelson Amaral, and Robert C. Holte*

Department of Computing Science,

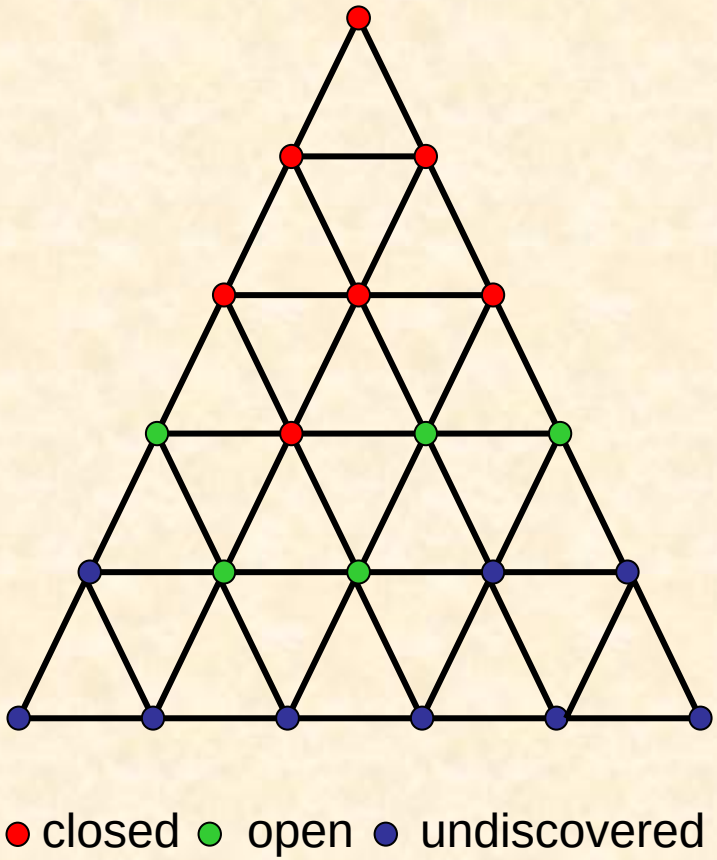
Edmonton, Alberta, Canada

# Overview

- A parallel algorithm for executing a breadth-first traversal algorithm of an implicit graph, *a.k.a.* state space
- The algorithm:
  - is based on the frontier breadth-first traversal algorithm
  - is secondary-storage oriented
  - is designed to run on a distributed-memory system
  - features:
    - bandwidth-bound secondary-storage access
    - bandwidth-bound communication
    - automated and adaptive workload distribution
- Traverse bigger graphs and traverse them faster

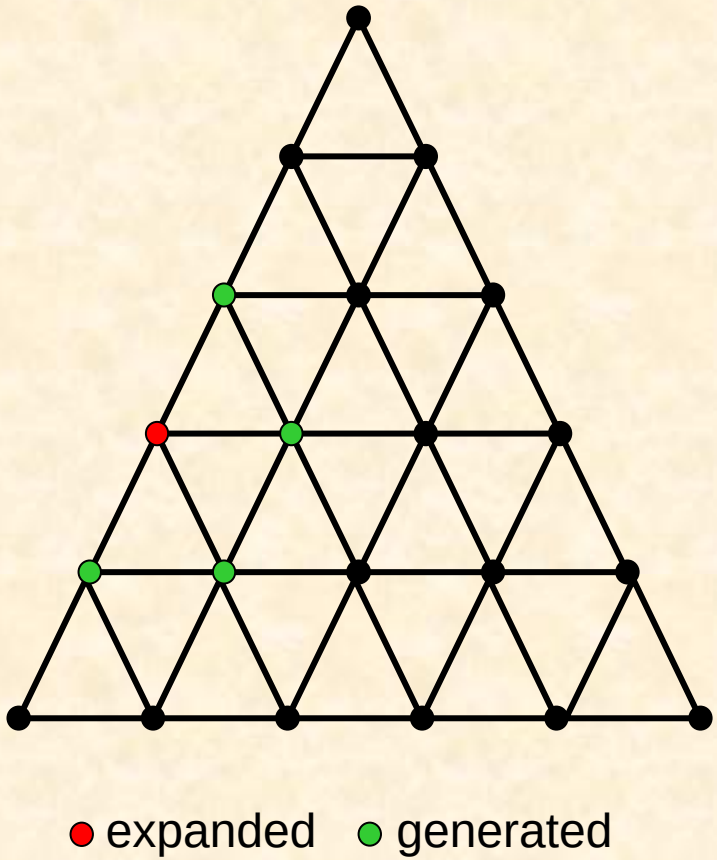
# Search-Algorithm Terminology

- During a breadth-first traversal each vertex is in one of three states
  - **closed** : a visited vertex
  - **open** : an unvisited vertex that is a neighbour of at least one visited vertex
  - **undiscovered** : an unvisited vertex and that is not a neighbour of at least one visited vertex



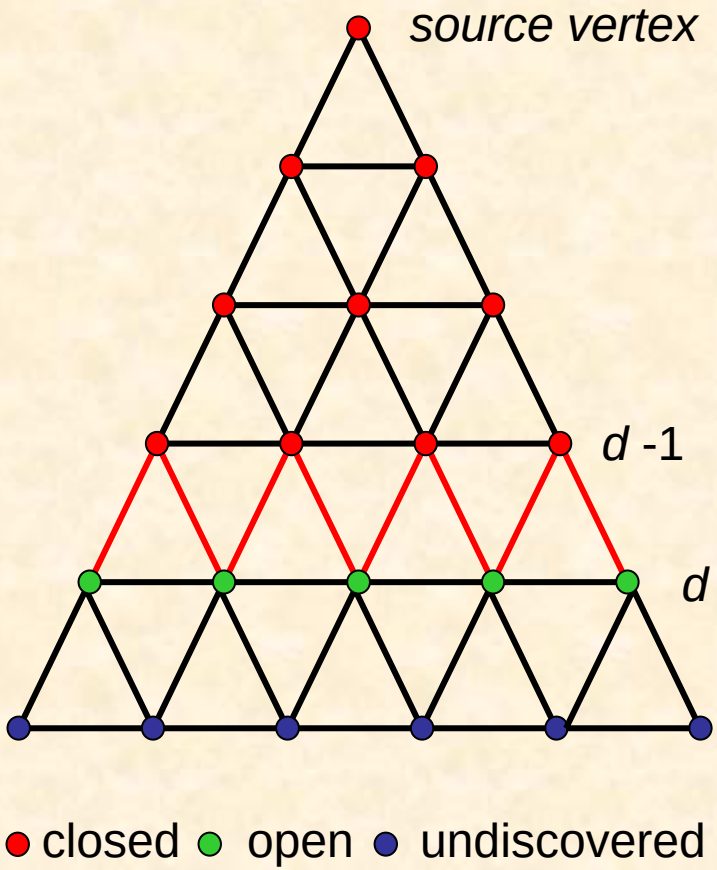
# Search-Algorithm Terminology

- We **expand** a vertex by computing each neighbour of a vertex and refer to each computed neighbour as a vertex **generated** in the expansion



# Sequential-Algorithm Structure

- Maintains
  - $Open_d$ : set of all open vertices at distance $d$ from the source vertex
  - $ClosedIn_d$ : set of all edges from open vertices at distance $d$ to closed vertices at distance $d - 1$ from the source vertex
- Computes $Open_d$ for successive values of $d$ starting at $d = 0$



# Sequential-Algorithm Structure

- For $d = 0$:
  - Compute $Open_d$ as set of vertices consisting of the source vertex and compute $ClosedIn_d$ as empty set of edges
- For $d \geq 1$:
  - Compute $Generated_{d\text{-}1}$ as set of all vertices that are generated in the expansion of the vertices in $Open_{d\text{-}1}$ and are not end-vertices of edges in $ClosedIn_{d\text{-}1}$
  - Compute $Open_d$ as set of all vertices in $Generated_{d\text{-}1}$ that are not in $Open_{d\text{-}1}$ and compute $ClosedIn_d$ as the set of all edges from vertices in $Open_d$ to vertices in $Open_{d\text{-}1}$
  - Delete $Open_{d\text{-}1}$, $ClosedIn_{d\text{-}1}$, and $Generated_{d\text{-}1}$

# Parallel-Algorithm Structure

- Given a range-$n$ vertex-mapping function $F$:
  - the $i$-th subset of a set of vertices $A$ defined by $F$ is the set of all vertices in $A$ that map to $i$ according to $F$
  - the $i$-th subset of a set of edges $A$ defined by $F$ is the set of all edges in $A$ whose start-vertices map to $i$ according to $F$
- For $d = 0$
  - In parallel, for each $0 \leq i \leq n - 1$, given a range-$n$ vertex-mapping function $F$, compute $Open_{d,i}$ as $i$-th subset of $Open_d$ defined by $F$ and $ClosedIn_{d,i}$ as $i$-th subset of $ClosedIn_d$ defined by $F$

# Parallel-Algorithm Structure

- Represents
  - $Open_d$ with $n$ sets of vertices $Open_{d,0}$, $Open_{d,1}$, …, $Open_{d,n-1}$
  - $ClosedIn_d$ with $n$ sets of edges $ClosedIn_{d,0}$, $ClosedIn_{d,1}$, …, $ClosedIn_{d,n-1}$
  - $Generated_d$ with $n$ sets of vertices $Generated_{d,0}$, $Generated_{d,1}$, …, $Generated_{d,n-1}$
- Uses range-$n$ vertex-mapping functions to partition sets of vertices and sets of edges
- For $d = 0$:
  - In parallel, for each $0 \leq i \leq n - 1$, given a range-$n$ vertex-mapping function $F$, compute $Open_{d,i}$ as $i$-th subset of $Open_d$ defined by $F$ and $ClosedIn_{d,i}$ as $i$-th subset of $ClosedIn_d$ defined by $F$

# Parallel-Algorithm Structure

- For $d \geq 1$:
  - In parallel, for each $0 \leq i \leq n - 1$, compute $Generated_{d\text{-}1,i}$ as set of all vertices that are generated in the expansion of the vertices in $Open_{d\text{-}1,i}$ and are not end-points of edges in $ClosedIn_{d\text{-}1,i}$
  - In parallel, given range-$n$ vertex-mapping function $F$, for each $0 \leq i \leq n - 1$, logically partition $Open_{d\text{-}1,i}$ into the $n$ subsets defined by $F$ and logically partition $Generated_{d\text{-}1,i}$ into the $n$ subsets defined by $F$

# Parallel-Algorithm Structure

- For $d \geq 1$: (continued)
  - In parallel, for each $0 \leq i \leq n-1$, compute $Open_{d,i}$ as the set of all vertices in the $i$-th subsets of $Generated_{d\text{-}1,0}$, $Generated_{d\text{-}1,1}$, …, $Generated_{d\text{-}1,n\text{-}1}$ that are not in the $i$-th subsets of $Open_{d\text{-}1,0}$, $Open_{d\text{-}1,1}$, …, $Open_{d\text{-}1,n\text{-}1}$ and compute $ClosedIn_{d,i}$ as the set of all edges from vertices in $Open_{d,i}$ to vertices in the $i$-th subsets of $Open_{d\text{-}1,0}$, $Open_{d\text{-}1,1}$, …, $Open_{d\text{-}1,n\text{-}1}$
  - In parallel, for each $0 \leq i \leq n-1$, delete $Open_{d\text{-}1,i}$, $ClosedIn_{d\text{-}1,i}$, and $Generated_{d\text{-}1,i}$

# Implementation

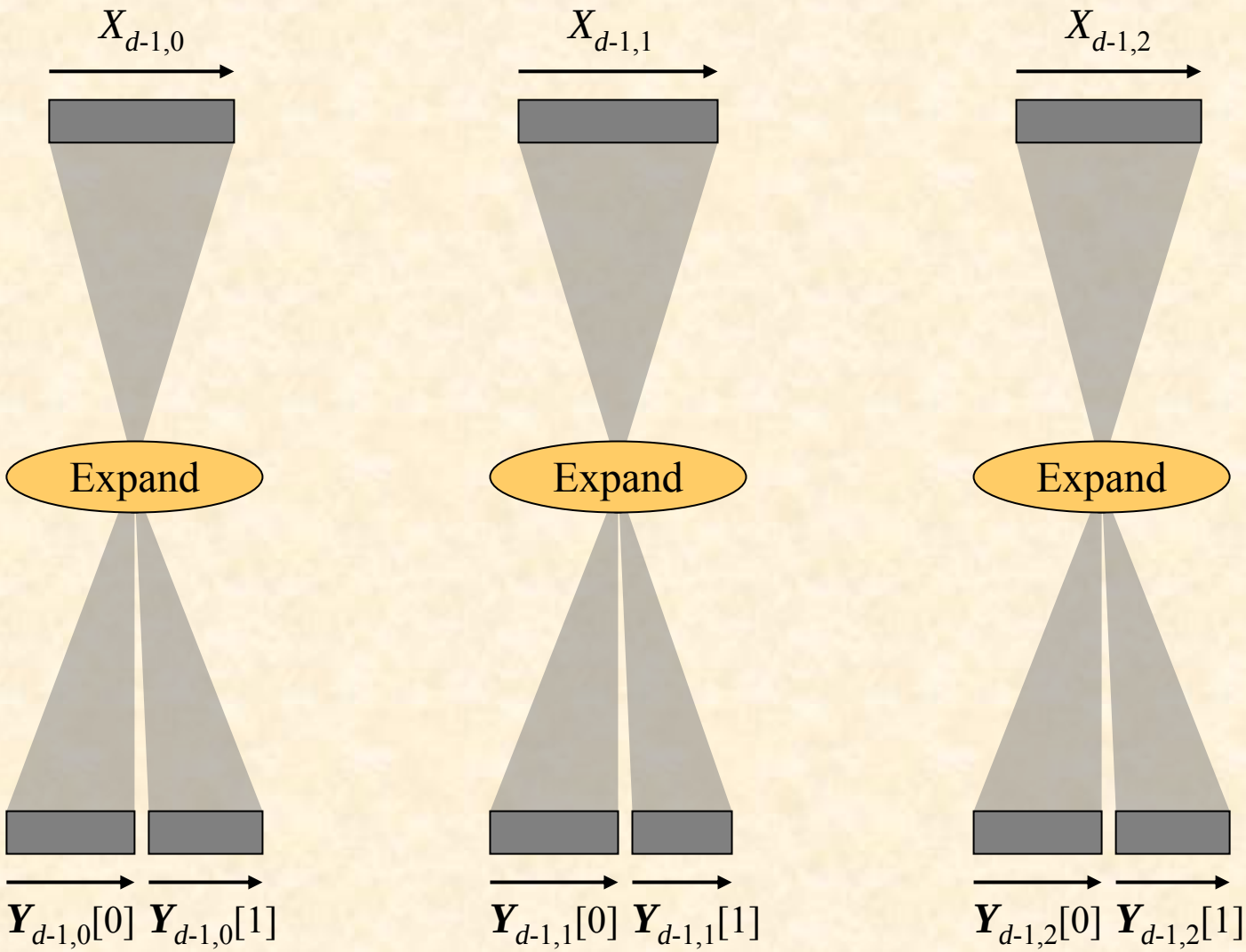
- Uses record runs and record sub-runs
  - a record consists of a vertex and of a subset of edges that start at the vertex
  - a run is a list of records where records appear in a non-decreasing order of their vertices
  - a sub-run maps a sub-list of a run
- Encapsulates
  - $Open_{d,i}$ and $ClosedIn_{d,i}$ with run $X_{d,i}$
  - $Generated_{d,i}$ and set of all vertices in $Generated_{d,i}$ to vertices in $Open_{d,i}$ with list of runs $\boldsymbol{Y}_{d,i}$

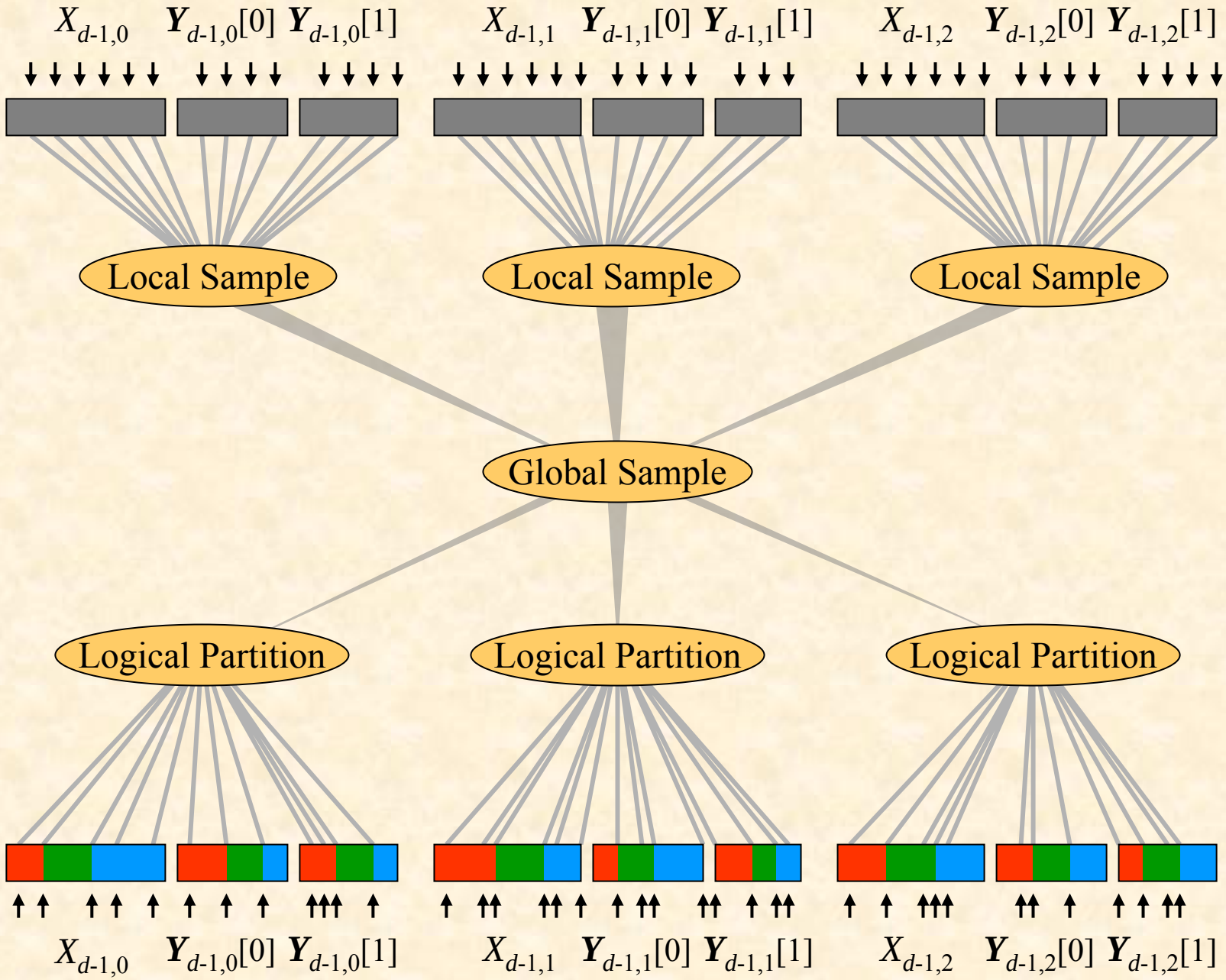
# Implementation

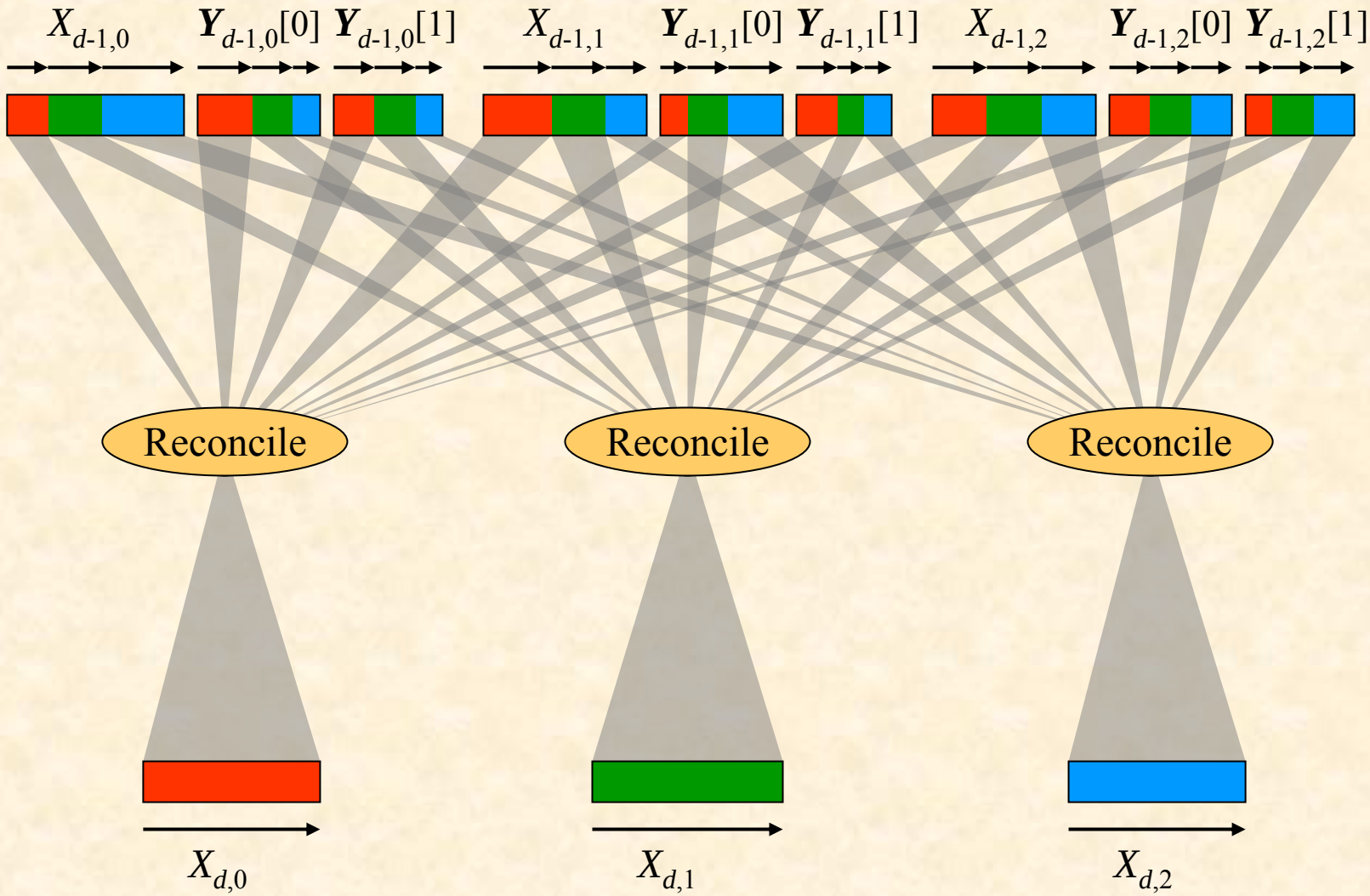
- Range-*n* vertex-mapping functions
  - *n* vertex intervals that split vertex interval from -∞ to ∞
    - map vertex to *i* if it falls into *i*-th interval
    - map edge to *i* if its start vertex falls into *i*-th interval
  - the *n* vertex intervals are computed via a sampling-based mechanism
    - sample vertices of records in each $Xd,i$ and in each $\mathbf{Y}d,i$ at a regular stride, collect samples, and compute $n - 1$ splitting points
  - use binary search to compute sub-runs of $X_{d,i}$ and of runs in $\mathbf{Y}_{d,i}$ that correspond to sub-sets defined by range-*n* vertex-mapping function

# Implementation

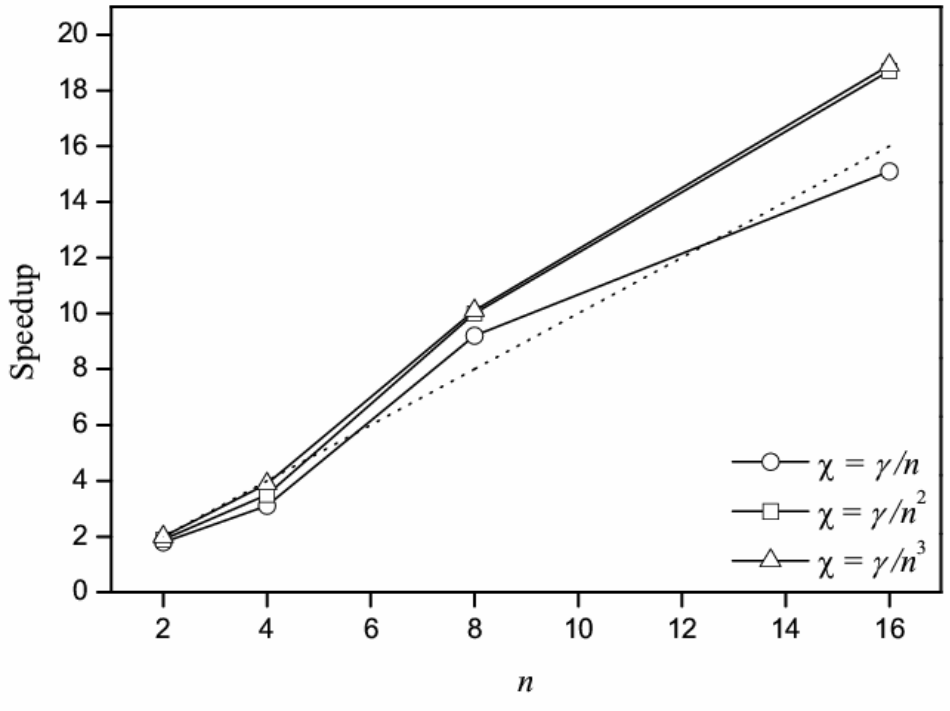
- Each $X_{d,i}$ and $\boldsymbol{Y}_{d,i}$ resides in secondary storage of i-th workstation
- Each workstation *i* executes two processes
  - worker : performs algorithm
  - server : facilitates streaming-access to $X_{d,i}$ and each run in $\boldsymbol{Y}_{d,i}$ to remote workers


$X_{d\text{-}1,0}$
Expand
$\boldsymbol{Y}_{d\text{-}1,0}[0]$ $\boldsymbol{Y}_{d\text{-}1,0}[1]$
$X_{d\text{-}1,1}$
Expand
$\boldsymbol{Y}_{d\text{-}1,1}[0]$ $\boldsymbol{Y}_{d\text{-}1,1}[1]$
$X_{d\text{-}1,2}$
Expand
$\boldsymbol{Y}_{d\text{-}1,2}[0]$ $\boldsymbol{Y}_{d\text{-}1,2}[1]$

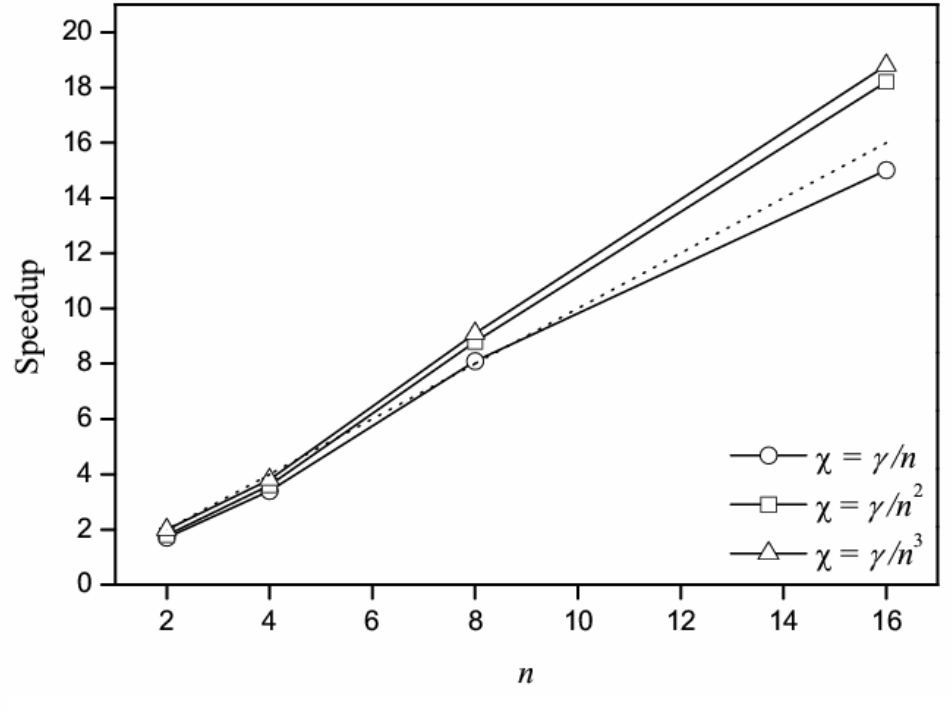

$X_{d\text{-}1,0}$ $\mathbf{Y}_{d\text{-}1,0}[0]$ $\mathbf{Y}_{d\text{-}1,0}[1]$
$X_{d\text{-}1,1}$ $\mathbf{Y}_{d\text{-}1,1}[0]$ $\mathbf{Y}_{d\text{-}1,1}[1]$
$X_{d\text{-}1,2}$ $\mathbf{Y}_{d\text{-}1,2}[0]$ $\mathbf{Y}_{d\text{-}1,2}[1]$
Local Sample
Local Sample
Local Sample
Global Sample
Logical Partition
Logical Partition
Logical Partition
$X_{d\text{-}1,0}$ $\mathbf{Y}_{d\text{-}1,0}[0]$ $\mathbf{Y}_{d\text{-}1,0}[1]$
$X_{d\text{-}1,1}$ $\mathbf{Y}_{d\text{-}1,1}[0]$ $\mathbf{Y}_{d\text{-}1,1}[1]$
$X_{d\text{-}1,2}$ $\mathbf{Y}_{d\text{-}1,2}[0]$ $\mathbf{Y}_{d\text{-}1,2}[1]$


$X_{d\text{-}1,0}$
$\boldsymbol{Y}_{d\text{-}1,0}[0]$
$\boldsymbol{Y}_{d\text{-}1,0}[1]$
$X_{d\text{-}1,1}$
$\boldsymbol{Y}_{d\text{-}1,1}[0]$
$\boldsymbol{Y}_{d\text{-}1,1}[1]$
$X_{d\text{-}1,2}$
$\boldsymbol{Y}_{d\text{-}1,2}[0]$
$\boldsymbol{Y}_{d\text{-}1,2}[1]$
Reconcile
Reconcile
Reconcile
$X_{d,0}$
$X_{d,1}$
$X_{d,2}$

# Experimental Evaluation




Sliding Tile Puzzle : 2x7




Four-Peg Towers of Hanoi : 18-disk

# Some Observations

- Approach extends to other breadth-first traversal algorithms
  - Divide-and-Conquer Breadth-First Search
  - Breadth-First Heuristic-Search and Divide-and-Conquer Breadth-First Heuristic-Search
- Additional things that can be done:
  - partition workload in expansion : *trivial*
  - work stealing:
    - work stealing in expansion : *trivial*
    - work stealing in reconciliation : *not trivial*