Cloud Computing
CS 15-319

Dryad and GraphLab

Lecture 11, Feb 22, 2012

Majd F. Sakr, Suhail Rehman and

Mohammad Hammoud

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

Today…

- Last session
  - Pregel

- Today's session
  - Dryad and GraphLab

- Announcement:
  - Project Phases I-A and I-B are due today

Objectives

Discussion on Programming Models

- Why parallelism?
- Parallel computer architectures
- Traditional models of parallel programming
- Examples of parallel processing
- Message Passing Interface (MPI)
- MapReduce
- Pregel, Dryad and GraphLab

Last 3 Sessions

Dryad

Dryad

- In this part, the following concepts of Dryad will be described:

  - Dryad Model
  - Dryad Organization
  - Dryad Description Language and An Example Program
  - Fault Tolerance in Dryad

Dryad

- In this part, the following concepts of Dryad will be described:

  - Dryad Model
  - Dryad Organization
  - Dryad Description Language and An Example Program
  - Fault Tolerance in Dryad

Dryad

- Dryad is a general purpose, high-performance, distributed computation engine

- Dryad is designed for:
  - High-throughput
  - Data-parallel computation
  - Use in a private datacenter

- Computation is expressed as a directed-acyclic-graph (DAG)
  - Vertices represent programs
  - Edges represent data channels between vertices

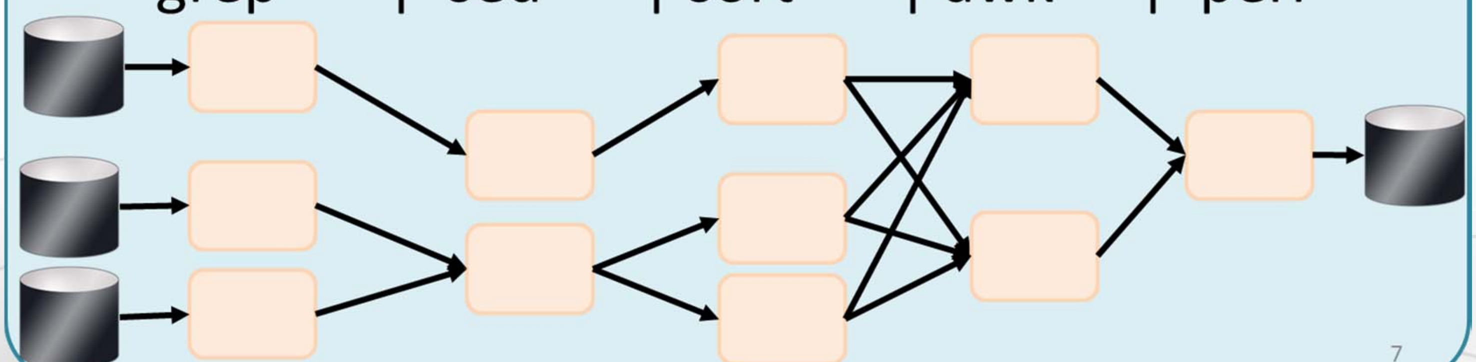
Unix Pipes vs. Dryad DAG

- Unix Pipes: 1-D

grep | sed | sort | awk | perl

- Dryad: 2-D

grep^{1000} | sed^{500} | sort^{1000} | awk^{500} | perl^{50}

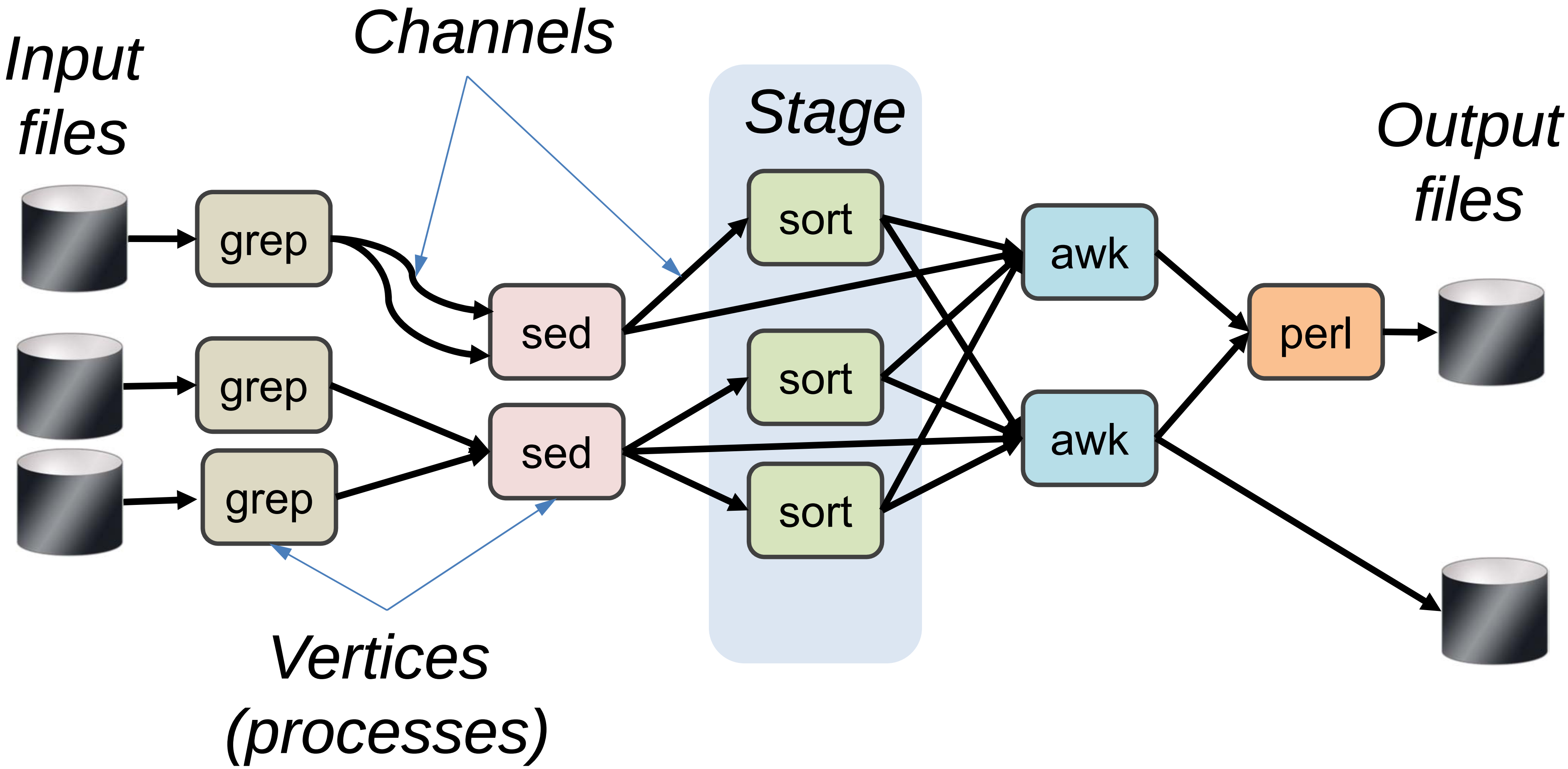


7

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

Dryad Job Structure

grep^{1000} | sed^{500} | sort^{1000} | awk^{500} | perl^{50}

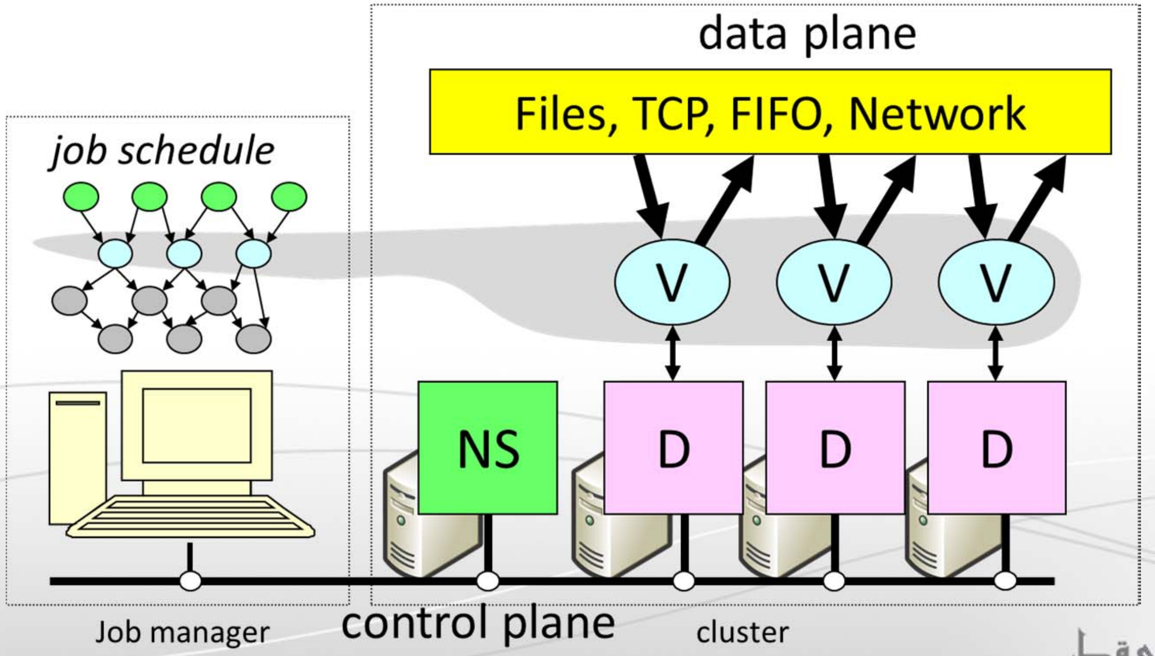


Dryad

- In this part, the following concepts of Dryad will be described:
  - Dryad Model
  - Dryad Organization
  - Dryad Description Language and An Example Program
  - Fault Tolerance in Dryad

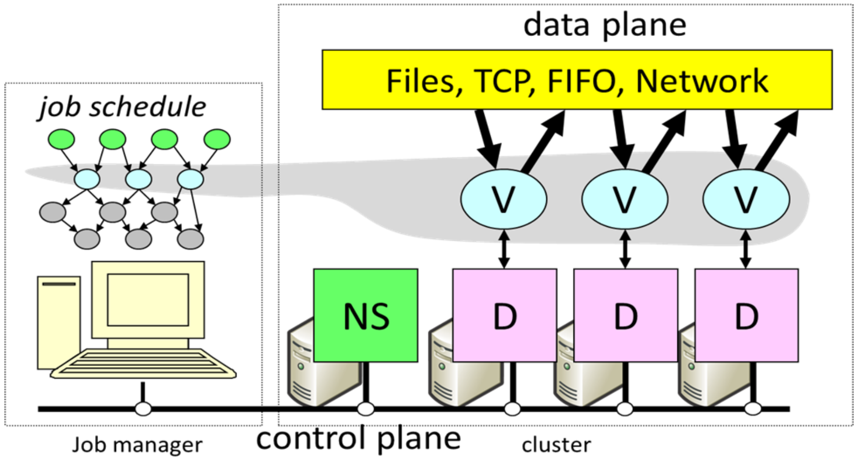
Dryad System Organization

- There are 3 roles for machines in Dryad
  - Job Manager (JM)
  - Name Server (NS)
  - Daemon (D)



Program Execution (1)

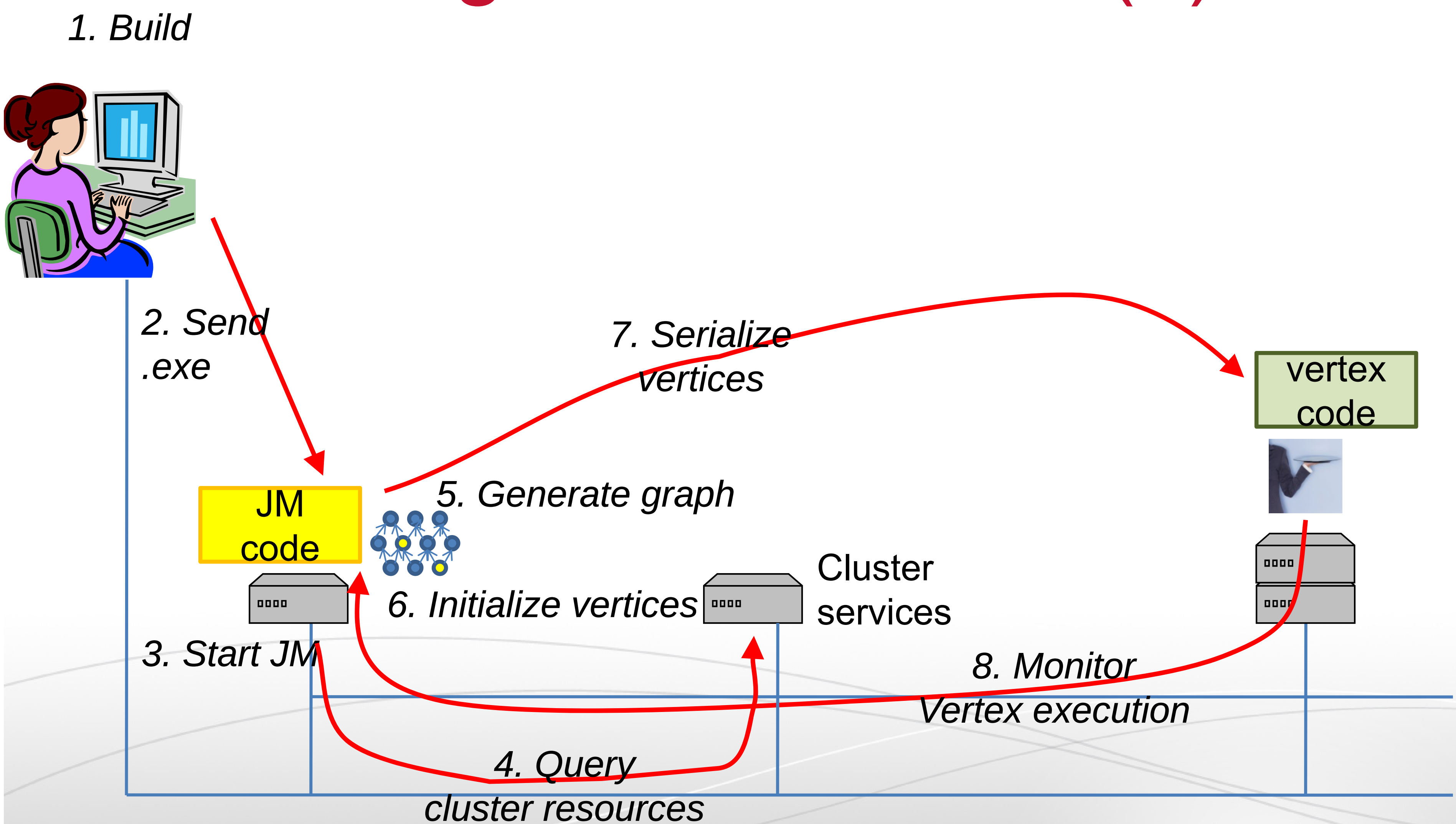
- The Job Manager (***JM***):
  - Creates the job communication graph (job schedule)
  - Contacts the ***NS*** to determine the number of ***D***s and the topology
  - Assigns ***V***s to each ***D*** (using a simple task scheduler- not described) for execution
  - Coordinates data flow through the data plane



- Data is distributed using a distributed storage system that shares with the Google File System some properties (e.g., data are split into chunks and replicated across machines)

- Dryad also supports the use of NTFS for accessing files locally

Program Execution (2)



جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

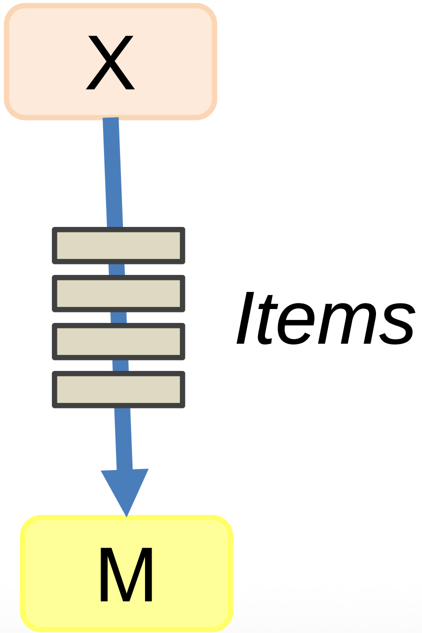
Data Channels in Dryad

- Data items can be *shuffled* between vertices through *data channels*

- Data channels can be:
  - Shared Memory FIFOs (intra-machine)
  - TCP Streams (inter-machine)
  - SMB/NTFS Local Files (temporary)
  - Distributed File System (persistent)

- The performance and fault tolerance of these mechanisms vary

- Data channels are abstracted for maximum flexibility



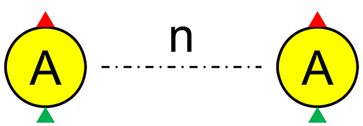
Dryad

- In this part, the following concepts of Dryad will be described:
  - Dryad Model
  - Dryad Organization
  - Dryad Description Language and An Example Program
  - Fault Tolerance in Dryad

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

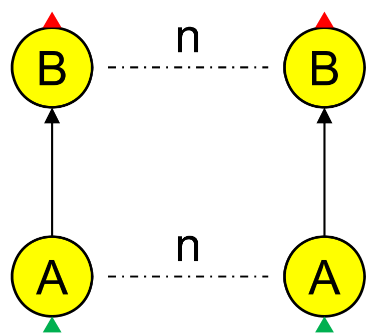
Dryad Graph Description Language

- Here are some operators in the Dryad graph description language:



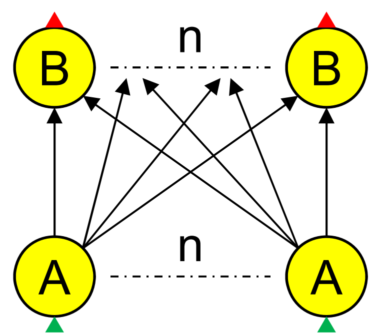
AS = A^n

(Cloning)



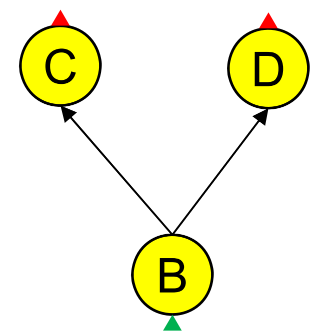
AS >= BS

(Pointwise Composition)



AS >> BS

(Bipartite Composition)

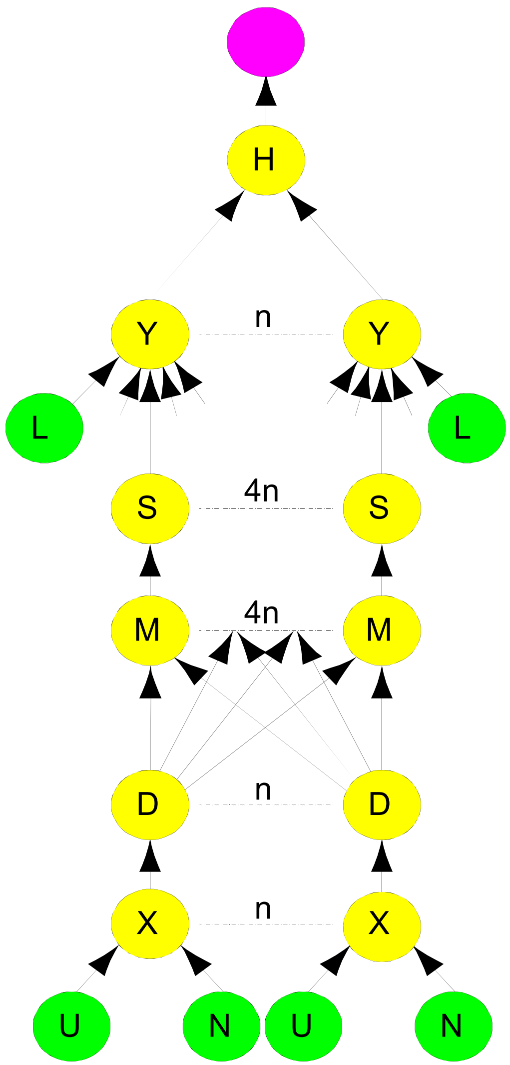


(B>=C) || (B>=D)

(Merge)

Example Program in Dryad (1)

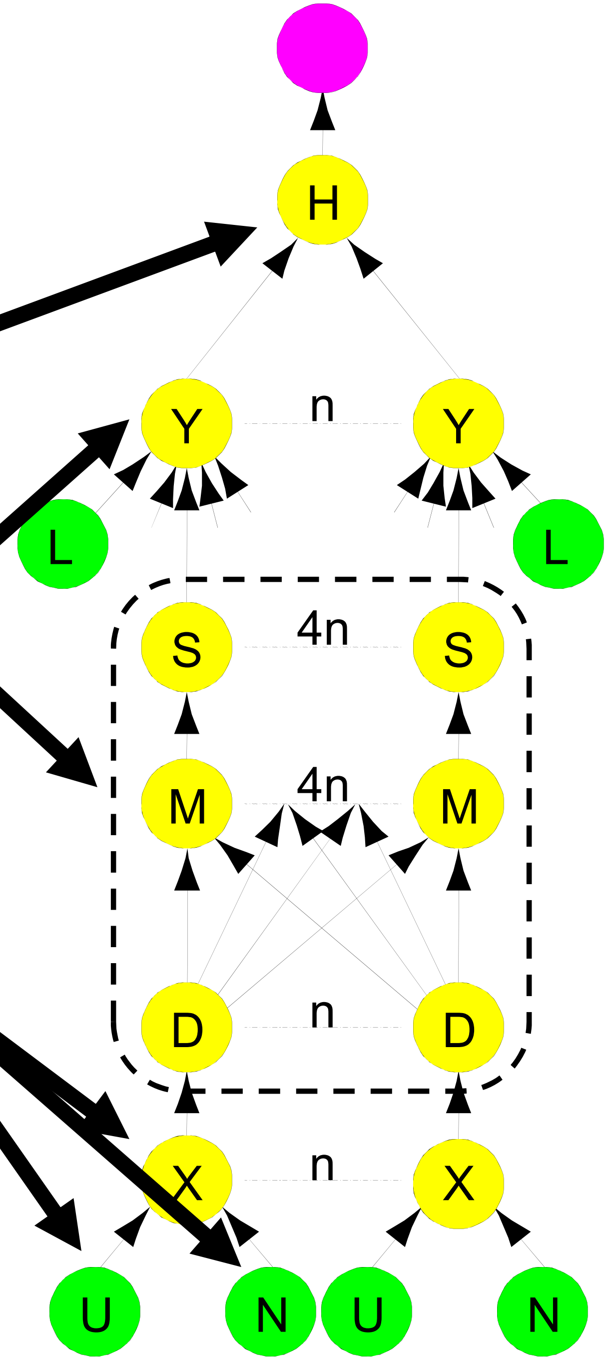
- Skyserver SQL Query (Q18):
  - Find all the objects in the database that have neighboring objects within 30 arc seconds such that at least one of the neighbors has a color similar to the primary object's color
  - There are two tables involved
    - photoObjAll and it has 354,254,163 records
    - Neighbors and it has 2,803,165,372 records
  - For the equivalent Dryad computation, they extracted the columns of interest into two binary files, "ugriz.bin" and "neighbors.bin"



Example Program in Dryad (2)

[distinct]
[merge outputs]

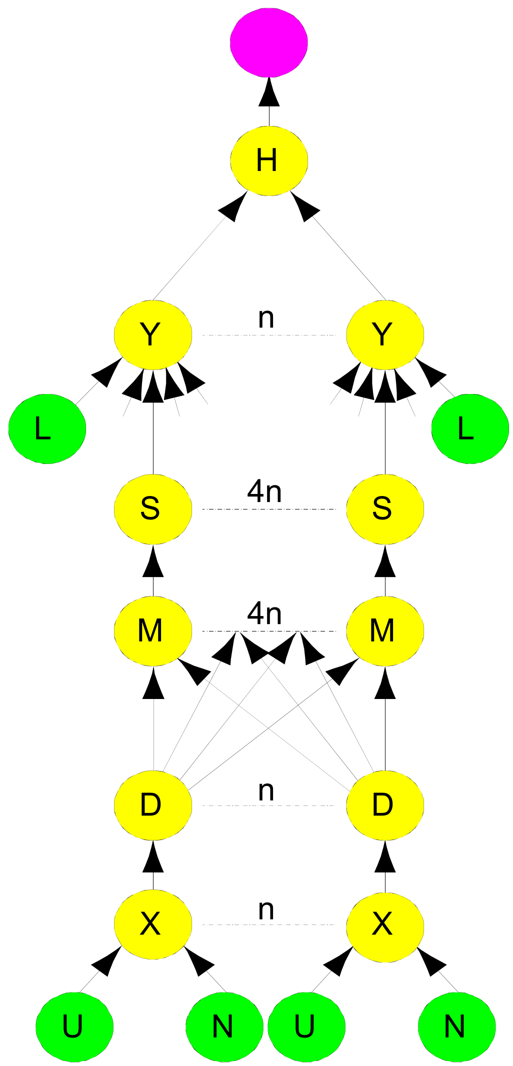
select
  u.objid
from u join <temp>
where
  u.objid = <temp>.neighborobjid and
  |u.color - <temp>.color| < d



Example Program in Dryad (3)

- Here is the corresponding Dryad code:

```
GraphBuilder XSet = moduleX^N;
GraphBuilder DSet = moduleD^N;
GraphBuilder MSet = moduleM^(N*4);
GraphBuilder SSet = moduleS^(N*4);
GraphBuilder YSet = moduleY^N;
GraphBuilder HSet = moduleH^1;

GraphBuilder XInputs = (ugriz1 >= XSet) || (neighbor >= XSet);

GraphBuilder YInputs = ugriz2 >= YSet;
GraphBuilder XToY = XSet >= DSet >> MSet >= SSet;

for (i = 0; i < N*4; ++i)
{
XToY = XToY || (SSet.GetVertex(i) >= YSet.GetVertex(i/4));
}

GraphBuilder YToH = YSet >= HSet;
GraphBuilder HOutputs = HSet >= output;

GraphBuilder final = XInputs || YInputs || XToY || YToH ||
HOutputs;
```



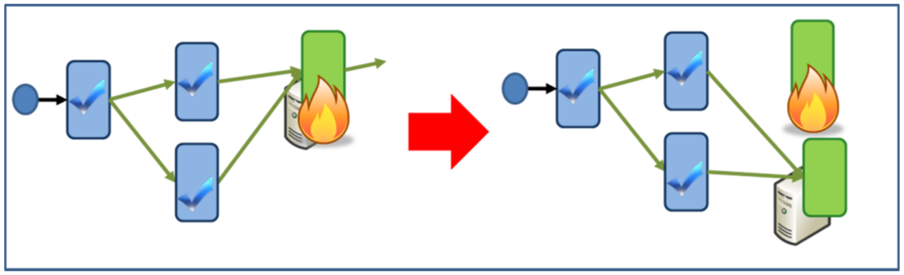
Dryad

- In this part, the following concepts of Dryad will be described:
  - Dryad Model
  - Dryad Organization
  - Dryad Description Language and An Example Program
  - Fault Tolerance in Dryad

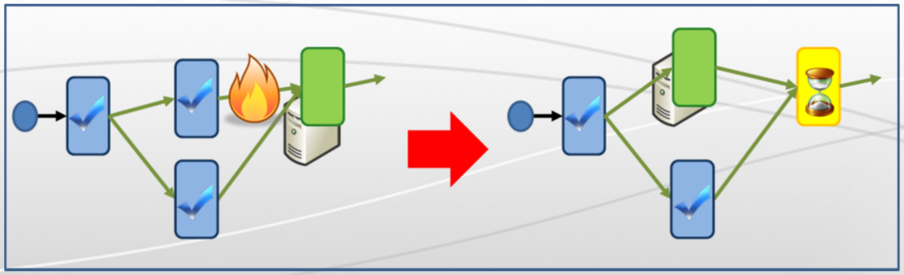
Fault Tolerance in Dryad (1)

- Dryad is designed to handle two types of failures:
  - Vertex failures
  - Channel failures

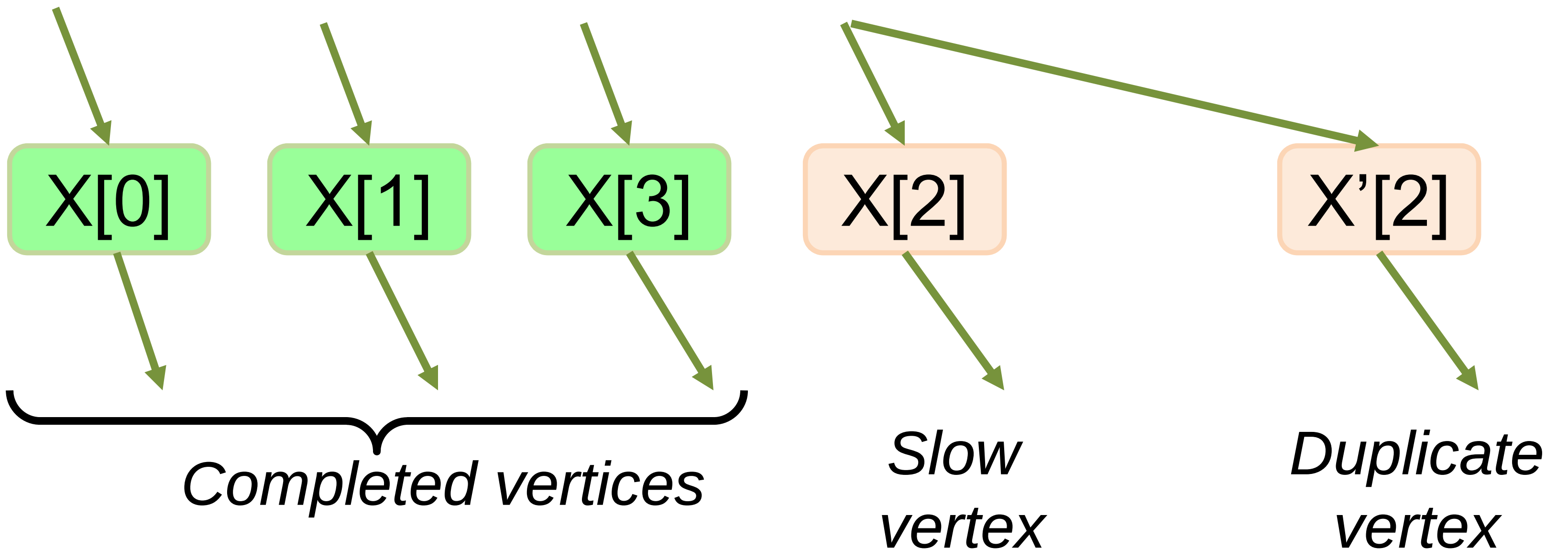
- Vertex failures are handled by the JM and the failed vertex is re-executed on another machine



- Channel failures cause the preceding vertex to be re-executed



Fault Tolerance in Dryad (2)



Duplication Policy = f(running times, data volumes)

GraphLab

GraphLab

- In this part, the following concepts of GraphLab will be described:
  - Motivation for GraphLab
  - GraphLab Data Model and Update Mechanisms
  - Scheduling in GraphLab
  - Consistency Models in GraphLab
  - PageRank in GraphLab

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

GraphLab

- In this part, the following concepts of GraphLab will be described:

  - Motivation for GraphLab
  - GraphLab Data Model and Update Mechanisms
  - Scheduling in GraphLab
  - Consistency Models in GraphLab
  - PageRank in GraphLab

Motivation for GraphLab

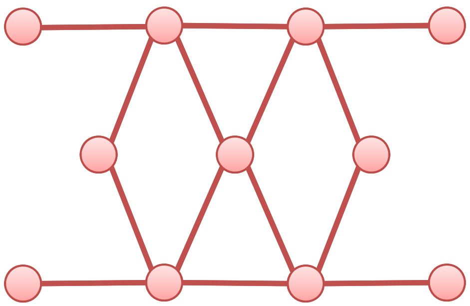
- Shortcomings of MapReduce
  - Interdependent data computation difficult to perform
  - Overheads of running jobs iteratively – disk access and startup overhead
  - Communication pattern is not user definable/flexible

- Shortcomings of Pregel
  - BSP model requires synchronous computation
  - One slow machine can slow down the entire computation considerably

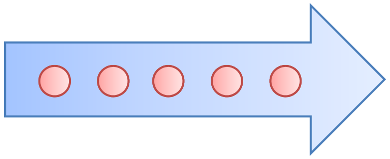
- Shortcomings of Dryad
  - Very flexible but steep learning curve for the programming model

GraphLab

- GraphLab is a framework for parallel machine learning

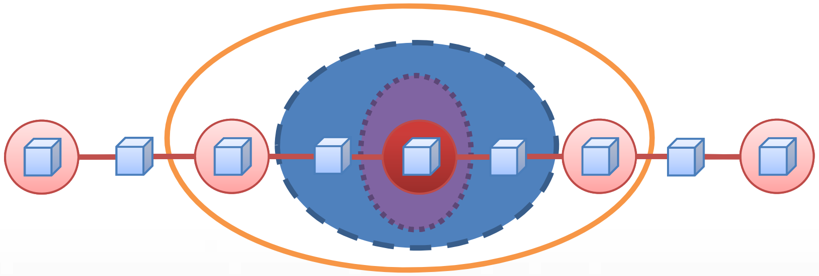


Data Graph



Scheduling

Shared Data Table



Update Functions and Scopes

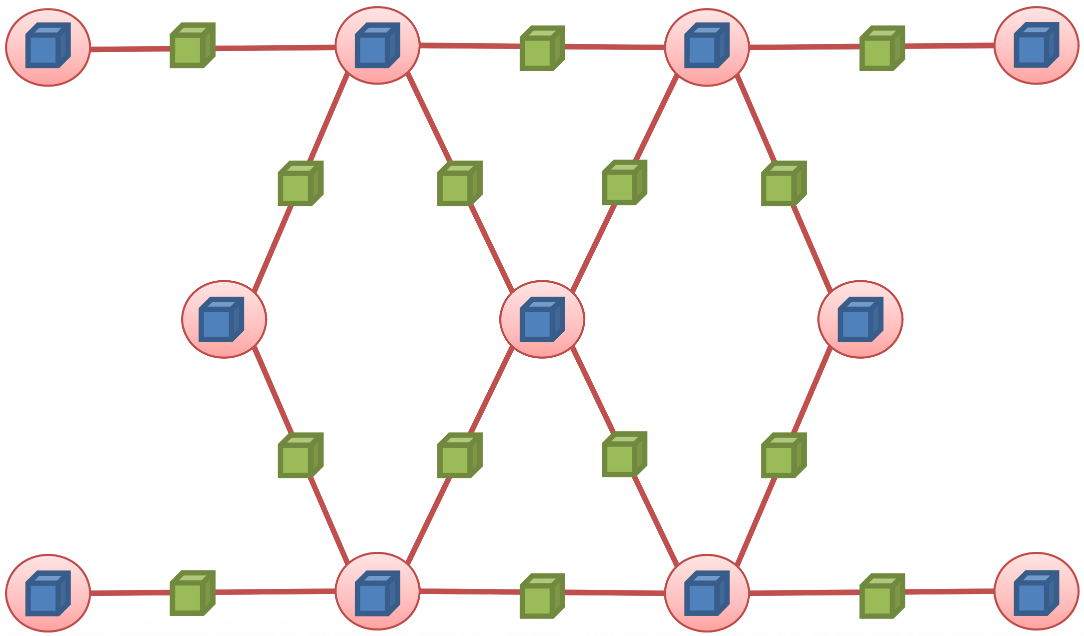
جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

GraphLab

- In this part, the following concepts of GraphLab will be described:
  - Motivation for GraphLab
  - GraphLab Data Model and Update Mechanisms
  - Scheduling in GraphLab
  - Consistency Models in GraphLab
  - PageRank in GraphLab

Data Graph

- A graph in GraphLab is associated with data at every vertex and edge

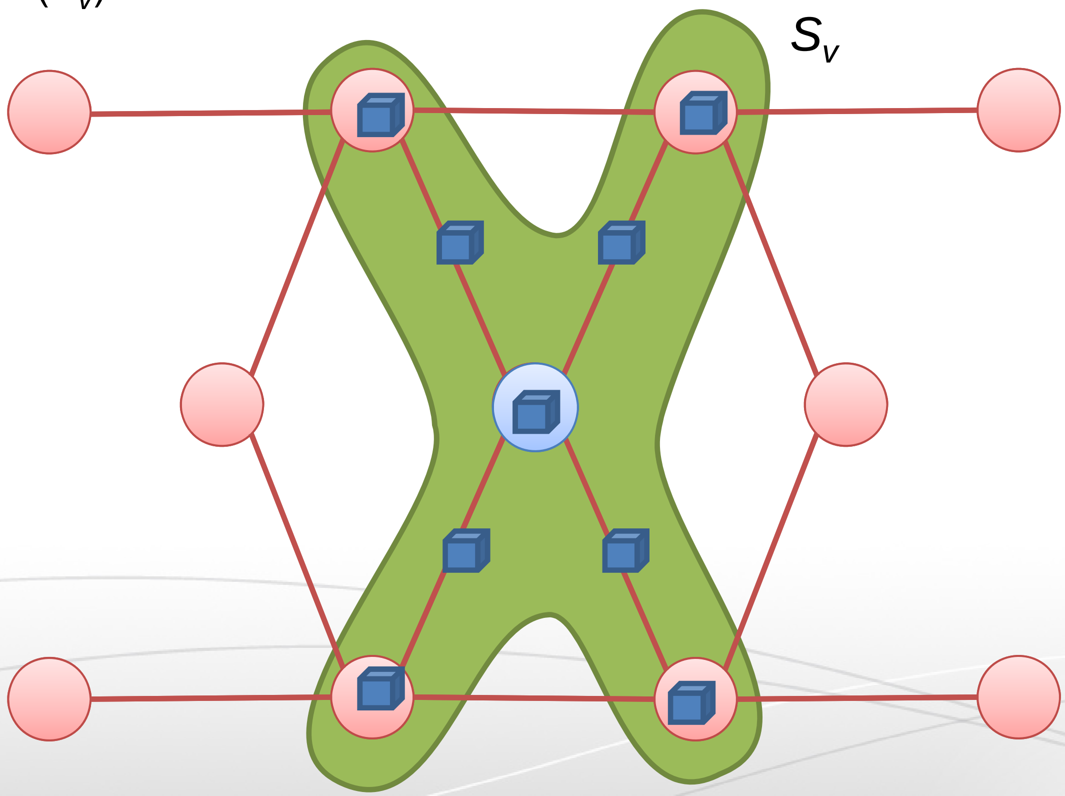


Data Graph

- Arbitrary blocks of data can be assigned to vertices and edges

Update Functions

- The data graph is modified using update functions
  - The update function can modify a vertex ***v*** and its neighborhood, defined as the scope of ***v*** (S_v)



Shared Data Table

- Certain algorithms require global information that is shared among all vertices (Algorithm Parameters, Statistics, etc.)
  - GraphLab exposes a Shared Data Table (SDT)

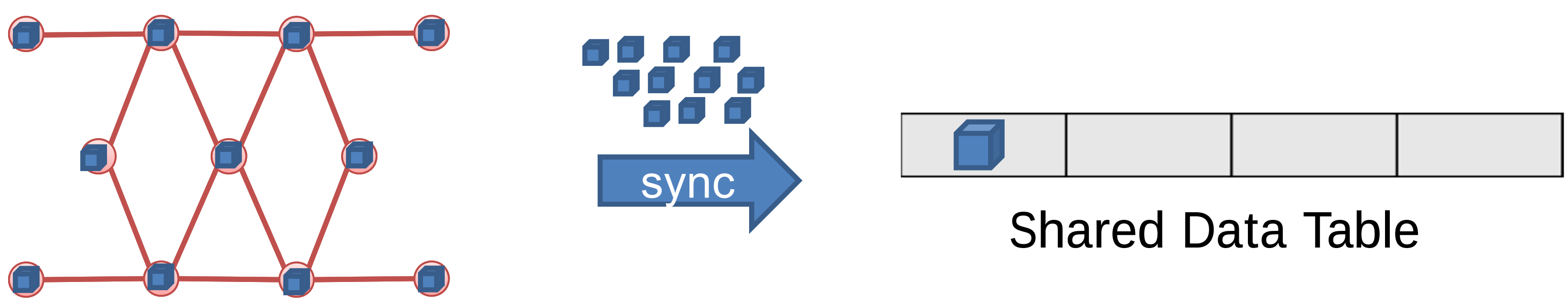
- SDT is an associative map between keys and arbitrary blocks of data
  - T[Key] → Value

Shared Data Table

- The shared data table is updated using the *sync mechanism*

Sync Mechanism

- Similar to Reduce in MapReduce
  - User can define fold, merge and apply functions that are triggered during the global sync mechanism
- Fold function allows the user to sequentially aggregate information across all vertices
- Merge optionally allows user to perform a parallel tree reduction on the aggregated data collected during the fold operation
- Apply function allows the user to finalize the resulting value from the fold/merge operations (such as normalization etc.)



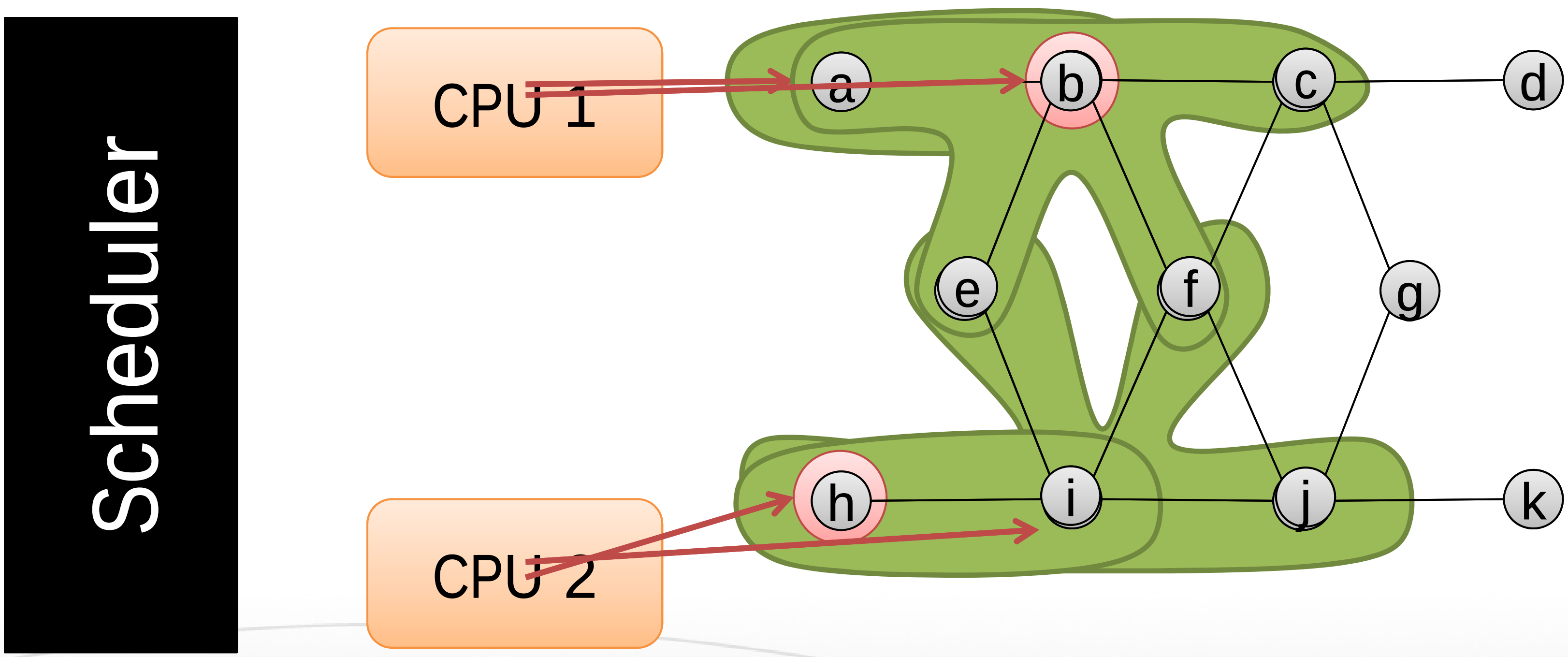
GraphLab

- In this part, the following concepts of GraphLab will be described:
  - Motivation for GraphLab
  - GraphLab Data Model and Update Mechanisms
  - Scheduling in GraphLab
  - Consistency Models in GraphLab
  - PageRank in GraphLab

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

Scheduling in GraphLab (1)

- The scheduler determines the order that vertices are updated



The process repeats until the scheduler is empty

Scheduling in GraphLab (2)

- An update schedule defines the order in which update functions are applied to vertices
  - A parallel data-structure called the scheduler represents an abstract list of tasks to be executed in Graphlab
- Base (Vertex) schedulers in GraphLab
  - Synchronous scheduler
  - Round-robin scheduler
- Job Schedulers in GraphLab
  - FIFO scheduler
  - Priority scheduler
- Custom schedulers can be defined by the set scheduler
- Termination Assessment
  - If the scheduler has no remaining tasks
  - Or, a termination function can be defined to check for convergence in the data

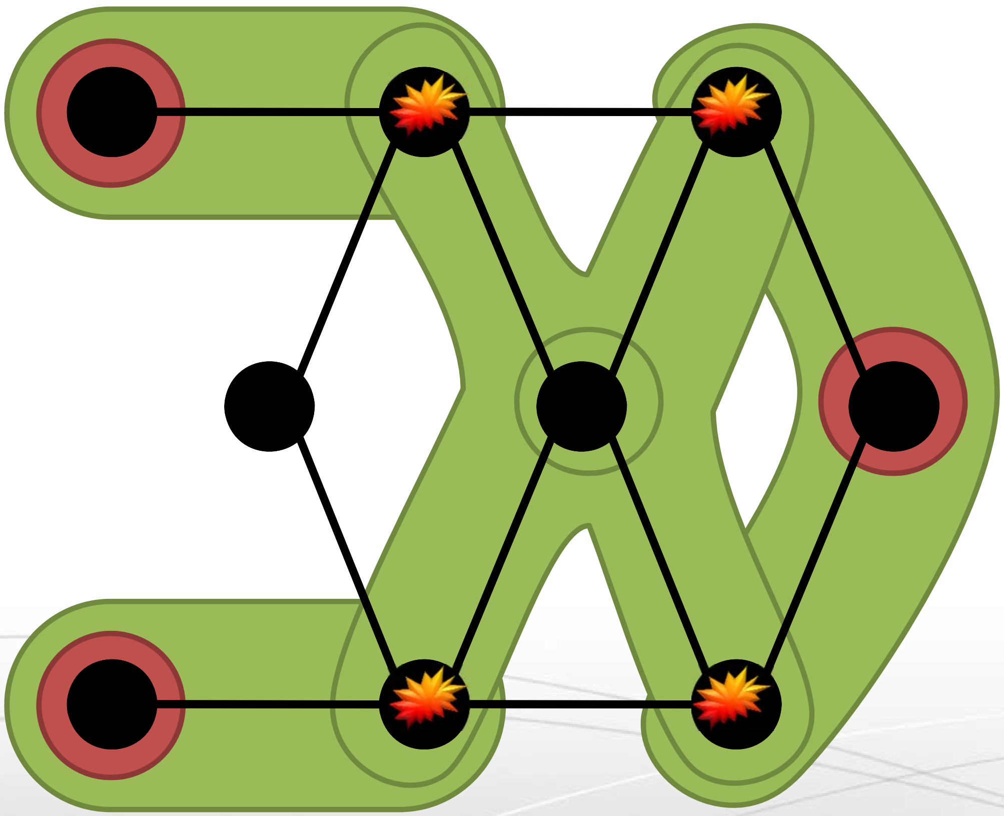
جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

GraphLab

- In this part, the following concepts of GraphLab will be described:
  - Motivation for GraphLab
  - GraphLab Data Model and Update Mechanisms
  - Scheduling in GraphLab
  - Consistency Models in GraphLab
  - PageRank in GraphLab

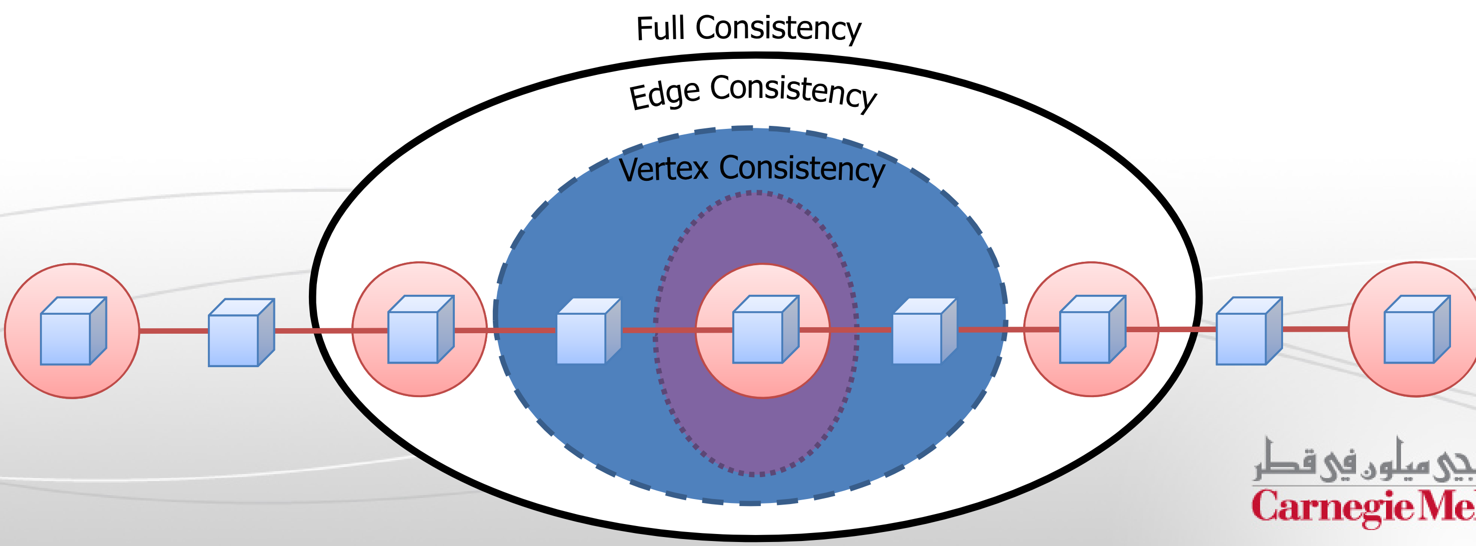
Need for Consistency Models

- How much can computation overlap?



Consistency Models in GraphLab

- GraphLab guarantees sequential consistency
  - Guaranteed to give the same result as a sequential execution of the computational steps
- User-defined consistency models
  - Full Consistency
  - Vertex Consistency
  - Edge Consistency



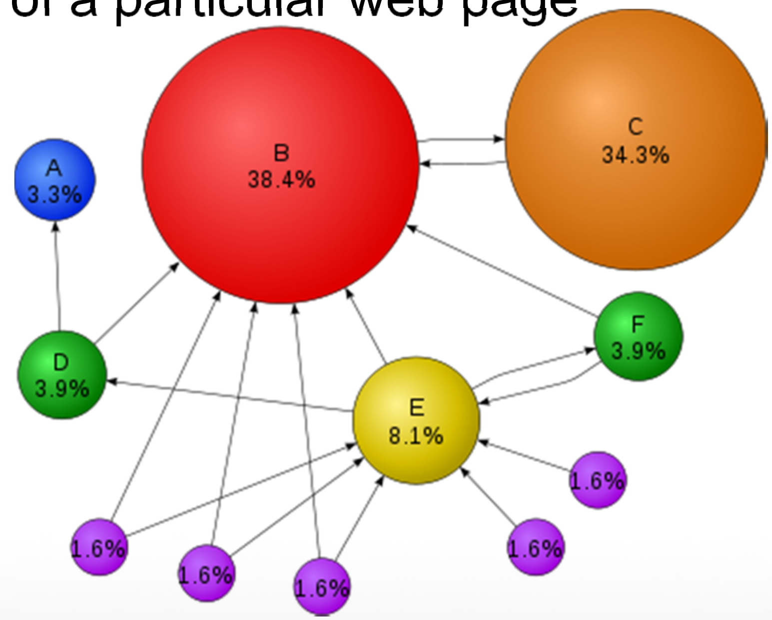
GraphLab

- In this part, the following concepts of GraphLab will be described:
  - Motivation for GraphLab
  - GraphLab Data Model and Update Mechanisms
  - Scheduling in GraphLab
  - Consistency Models in GraphLab
  - PageRank in GraphLab

جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

PageRank (1)

- PageRank is a link analysis algorithm
- The rank value indicates an importance of a particular web page
- A hyperlink to a page counts as a vote of support
- A page that is linked to by many pages with high PageRank receives a high rank itself



- A PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to the document with the 0.5 PageRank

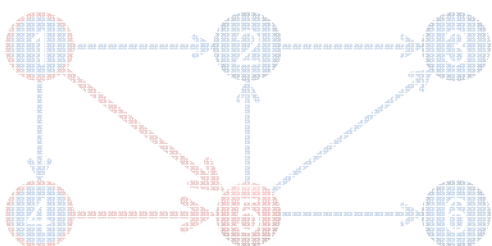
جامعة كارنيجي ميلون في قطر
Carnegie Mellon Qatar

PageRank (2)

- Iterate:

$$R[i] = \alpha + (1-\alpha) \sum_{(j,i)\in E} \frac{1}{L[j]} R[j]$$

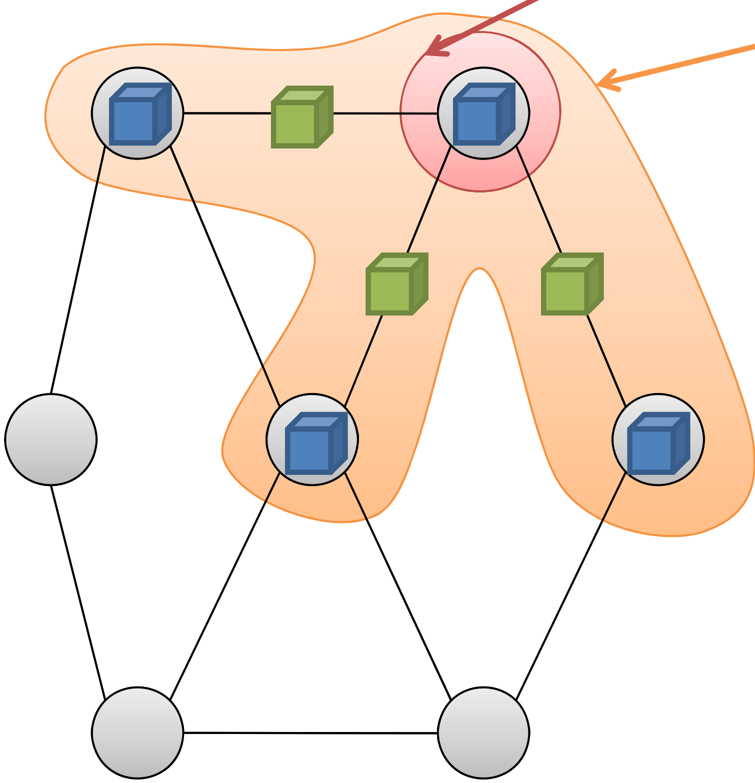
- Where:
  - α is the random reset probability
  - $L[j]$ is the number of links on page j



$$R[5] = \alpha + (1-\alpha)\left(\frac{1}{3}R[1] + \frac{1}{1}R[4]\right)$$

PageRank Example in GraphLab

- PageRank algorithm is defined as a per-vertex operation working on the scope of the vertex



```
pagerank(i, scope){
  // Get Neighborhood data
  (R[i], W_ij, R[j]) ←scope;

  // Update the vertex data
```

$$R[i] \leftarrow \alpha + (1-\alpha)\sum_{j \in N[i]} W_{ji} \times R[j];$$

```
  // Reschedule Neighbors if needed
  if R[i] changes then
    reschedule_neighbors_of(i);
}
```

Dynamic computation

How MapReduce, Pregel, Dryad and GraphLab Compare Against Each Other?

Comparison of the Programming Models

	MapReduce	Pregel	Dryad	GraphLab
Programming Model	Fixed Functions – Map and Reduce	Supersteps over a data graph with messages passed	DAG with program vertices and data edges	Data graph with shared data table and update functions
Parallelism	Concurrent execution of tasks within map and reduce phases	Concurrent execution of user functions over vertices within a superstep	Concurrent execution of vertices during a stage	Concurrent execution of non-overlapping scopes, defined by consistency model
Data Handling	Distributed file system	Distributed file system	Flexible data channels: Memory, Files, DFS etc.	Undefined – Graphs can be in memory or on disk
Task Scheduling	Fixed Phases – HDFS Locality based map task assignment	Partitioned Graph and Inputs assigned by assignment functions	Job and Stage Managers assign vertices to available daemons	Pluggable schedulers to schedule update functions
Fault Tolerance	DFS replication + Task reassignment / Speculative execution of Tasks	Checkpointing and superstep re-execution	Vertex and Edge failure recovery	Synchronous and asychronous snapshots
Developed by	Google	Google	Microsoft	Carnegie Mellon

References

- This presentation has elements borrowed from various papers and presentations:

- Papers:
  - Pregel: http://kowshik.github.com/JPregel/pregel_paper.pdf
  - Dryad: http://research.microsoft.com/pubs/63785/eurosys07.pdf
  - GraphLab: http://www.select.cs.cmu.edu/publications/paperdir/uai2010-low-gonzalez-kyrola-bickson-guestrin-hellerstein.pdf

- Presentations:
  - Dryad Presentation at Berkeley by M. Budiu: http://budiu.info/work/dryad-talk-berkeley09.pptx
  - GraphLab1 Presentation: http://graphlab.org/uai2010_graphlab.pptx
  - GraphLab2 Presentation: http://graphlab.org/presentations/nips-biglearn-2011.pptx

Next Class

Distributed File Systems