End-to-end Data-flow Parallelism for Throughput Optimization in High-speed Networks

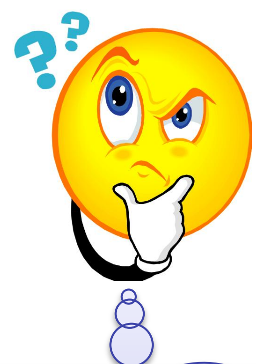
Esma Yildirim
Data Intensive Distributed Computing Laboratory
University at Buffalo (SUNY)

Condor Week 2011

Motivation

- Data grows larger hence the need for speed to transfer it
- Technology develops with the introduction of high-speed networks and complex computer architectures which are not fully utilized yet
- Still many questions are out in the uncertainty

I can not receive the speed I am supposed to get from the network



I have a 10G high-speed network and supercomputers connecting. Why do I still get under 1G throughput?

I want to get high throughput without congesting the traffic too much. How can I do it in the application level?

OK, may be I am asking too much but I want to get optimal settings to achieve maximal throughput

I can't wait for a new protocol to replace the current ones, why can't I get high throughput with what I have at hand?

Introduction

- Users of data-intensive applications need intelligent services and schedulers that will provide models and strategies to optimize their data transfer jobs
- Goals:
  - Maximize throughput
  - Minimize model overhead
  - Do not cause contention among users
  - Use minimum number of end-system resources

Introduction

- Current optical technology supports 100 G transport hence, the utilization of network brings a challenge to the middleware to provide faster data transfer speeds
- Achieving multiple Gbps throughput have become a burden over TCP-based networks
  - Parallel streams can solve the problem of network utilization inefficiency of TCP
  - Finding the optimal number of streams is a challenging task
- With faster networks end-systems have become the major source of bottleneck
  - CPU, NIC and Disk Bottleneck
- We provide models to decide on the optimal number of parallelism and CPU/disk stripes

Outline

- Stork Overview
- End-system Bottlenecks
- End-to-end Data-flow Parallelism
- Optimization Algorithm
- Conclusions and Future Work

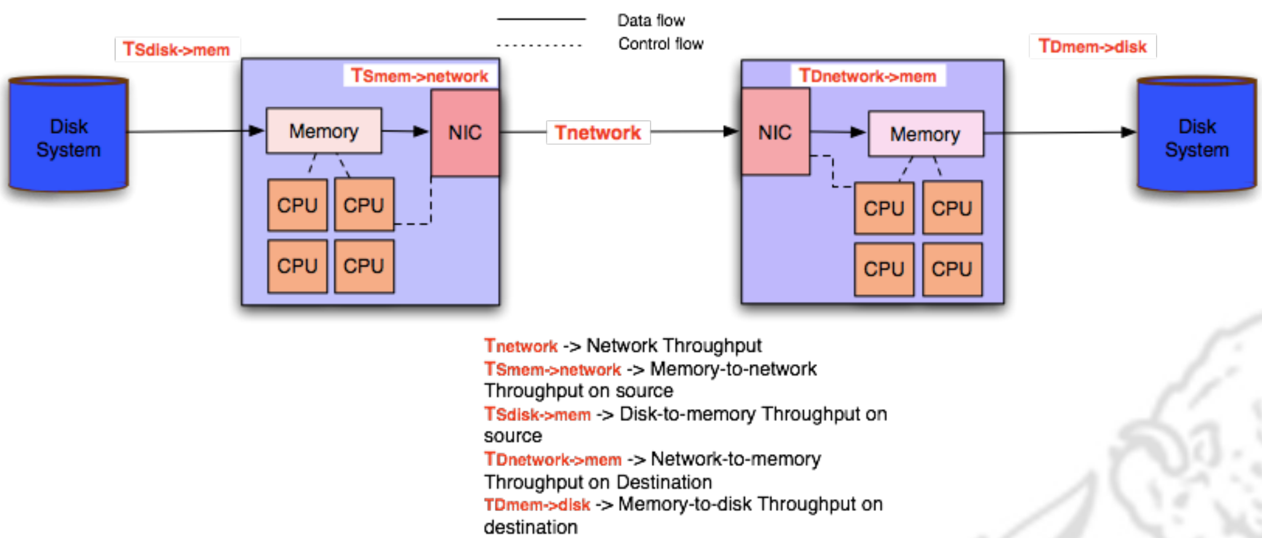
Stork Data Scheduler

- Implements state-of-the art models and algorithms for data scheduling and optimization
- Started as part of the Condor project as PhD thesis of Dr. Tevfik Kosar
- Currently developed at University at Buffalo and funded by NSF
- Heavily uses some Condor libraries such as ClassAds and DaemonCore

Stork Data Scheduler (cont.)

- Stork v.2.0 is available with enhanced features
  - http://www.storkproject.org
- Supports more than 20 platforms (mostly Linux flavors)
- Windows and Azure Cloud support planned soon
- The most recent enhancement:
  - Throughput Estimation and Optimization Service

End-to-end Data Transfer

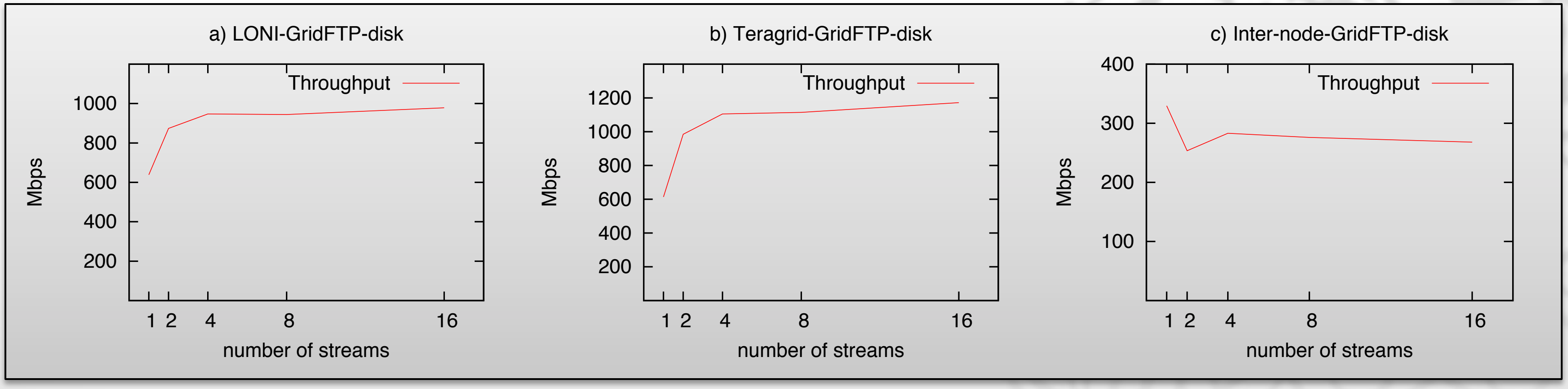


- Method to improve the end-to-end data transfer throughput
  - Application-level Data Flow Parallelism
    - Network level parallelism (parallel streams)
    - Disk/CPU level parallelism (stripes)

Network Bottleneck

Step1: Effect of Parallel Streams on Disk-to-disk Transfers

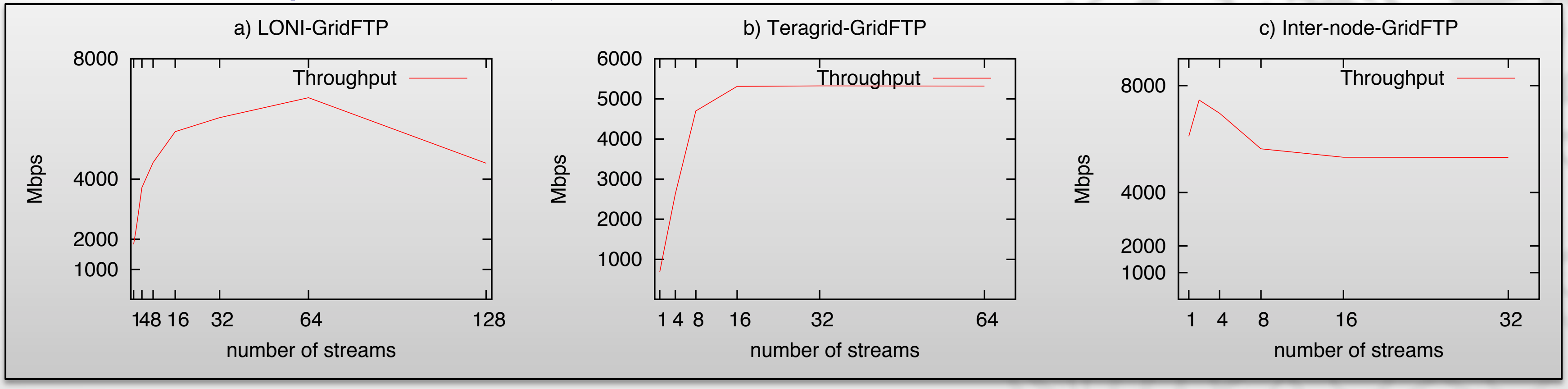
- Parallel streams can improve the data throughput but only to a certain extent
- Disk speed presents a major limitation.
- Parallel streams may have an adverse effect if the disk speed upper limit is already reached



Disk Bottleneck

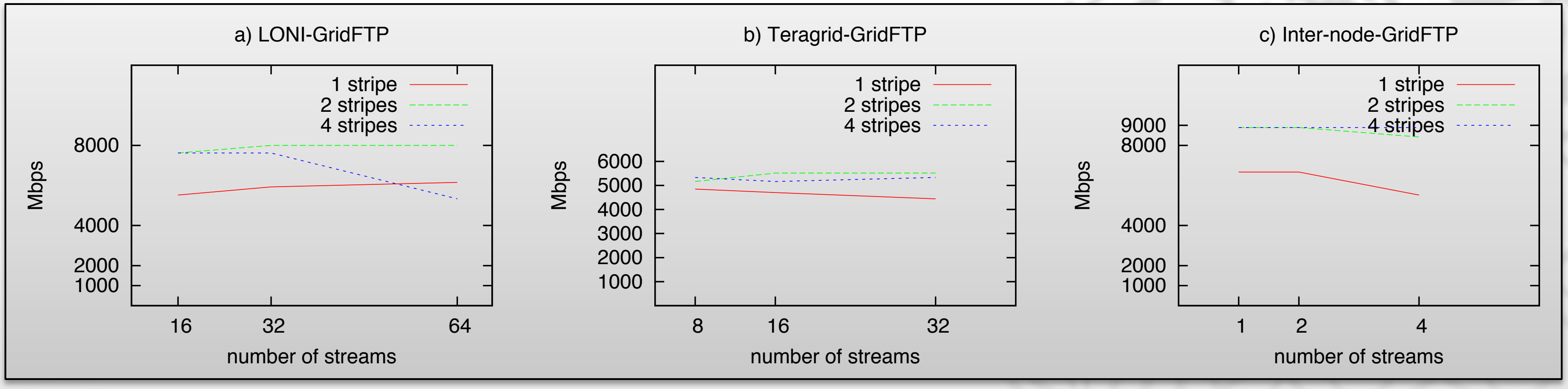
☑ Step2: Effect of Parallel Streams on Memory-to-memory Transfers and CPU Utilization

- Once disk bottleneck is eliminated, parallel streams improve the throughput dramatically
- Throughput either becomes stable or falls down after reaching its peak due to network or end-system limitations. Ex:The network interface card limit(10G) could not be reached (e.g. 7.5Gbps-internode)



CPU Bottleneck

- Step3: Effect of Striping and Removal of CPU Bottleneck
  - Striped transfers improves the throughput dramatically
  - Network card limit is reached for inter-node transfers(9Gbps)

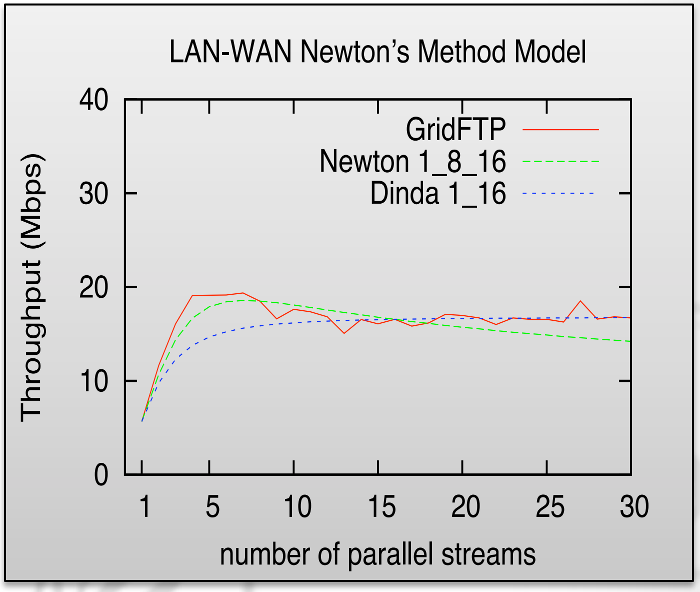


Prediction of Optimal Parallel Stream Number

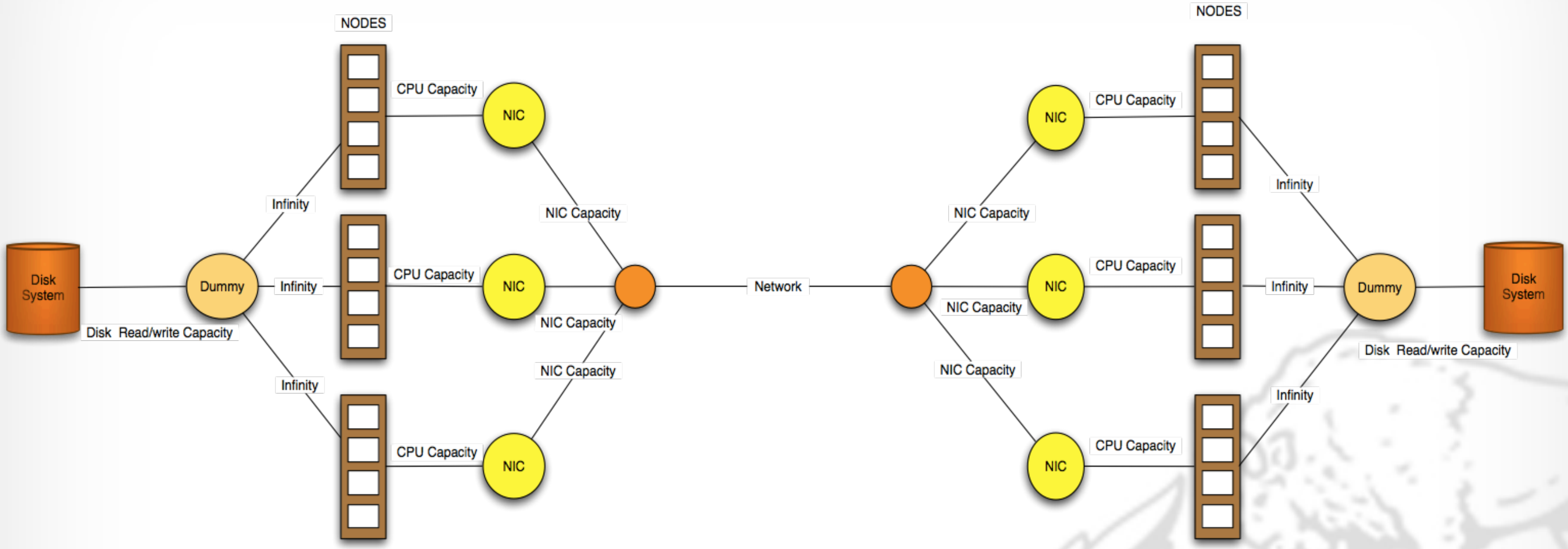
- Throughput formulation : Newton's Iteration Model

$$Th_n = \frac{n}{\sqrt{a' n^{c'} + b'}}$$

- ***a'*** , ***b'*** and ***c'*** are three unknowns to be solved hence 3 throughput measurements of different parallelism level (***n***) are needed
- Sampling strategy:
  - Exponentially increasing parallelism levels
    - Choose points not close to each other
    - Select points that are power of 2: 1, 2, 4, 8, … , 2^k
    - Stop when the throughput starts to decrease or increase very slowly comparing to the previous level
- Selection of 3 data points
  - From the available sampling points
    - For every 3-point combination, calculate the predicted throughput curve
    - Find the distance between the actual and predicted throughput curve
    - Choose the combination with the minimum distance

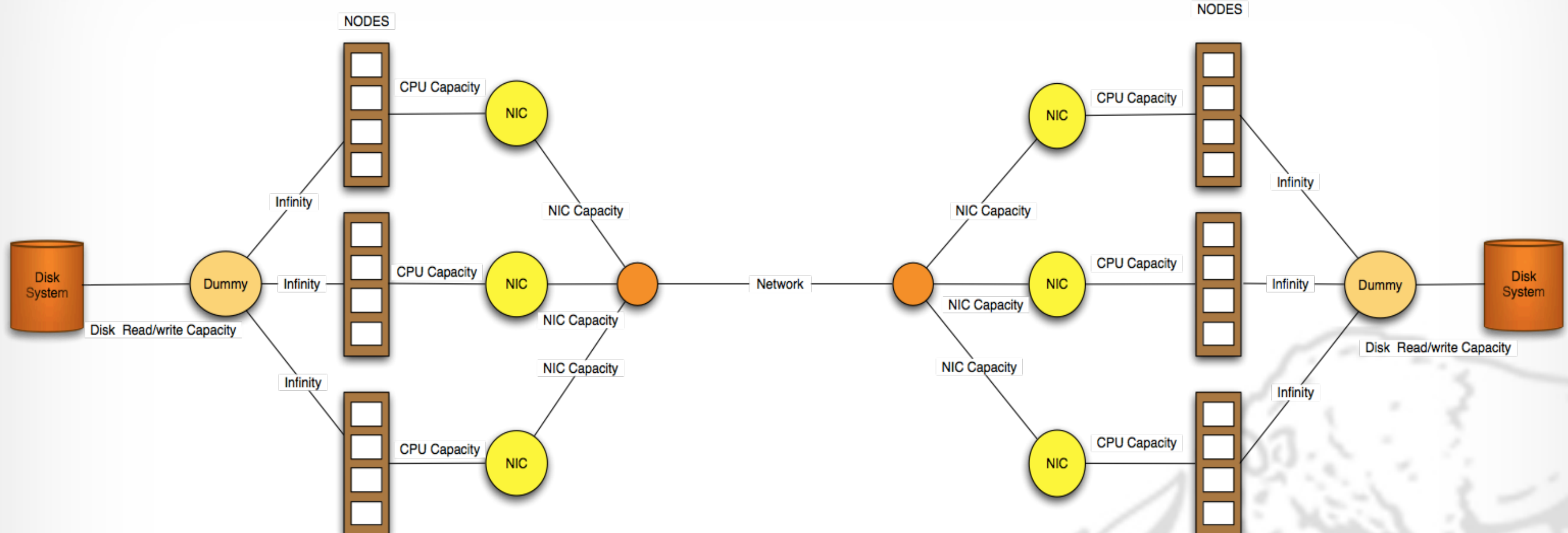


Flow Model of End-to-end Throughput



- CPU nodes are considered as nodes of a maximum flow problem
- Memory-to-memory transfers are simulated with dummy source and sink nodes
- The capacities of disk and network is found by applying parallel stream model by taking into consideration of resource capacities (NIC & CPU)

Flow Model of End-to-end Throughput



- Assumptions
  - Parameters not given and found by the model:
    - Available network capacity (***$U_{network}$***)
    - Available disk system capacity (***U_{disk}***)
  - Parameters given
    - CPU capacity (100% assuming they are idle at the beginning of the transfer) (***U_{CPU}***)
    - NIC capacity (U_{NIC})
    - Number of available nodes (***N_{avail}***)

- Convert the end-system and network capacities into a flow problem
- Goal: Provide maximal possible data transfer throughput given real-time traffic (maximize (*Th*))
  - Number of streams per stripe (***N_{si}***)
  - Number of stripes per node (***S_x***)
  - Number of nodes (***N_n***)

Flow Model of End-to-end Throughput

- Variables:
  - Uij = Total capacity of each arc from node i to node j
  - Uf= Maximal (optimal) capacity of each flow (stripe)
  - N_{opt} = Number of streams for U_f
  - X_{ij} = Total amount of flow passing i –> j
  - X_{fk} = Amount of each flow (stripe)
  - N_{Si}= Number of streams to be used for X_{fkij}
  - Sx_{ij}= Number of stripes passing i– > j
  - N_n = Number of nodes
- Inequalities:
- There is a high positive correlation between the throughput of parallel streams and CPU utilization
  - The linear relation between CPU utilization and Throughput is presented as :

$$0 \le X_{ij} \le U_{ij} \qquad 0 \le X_{fk} \le U_f$$

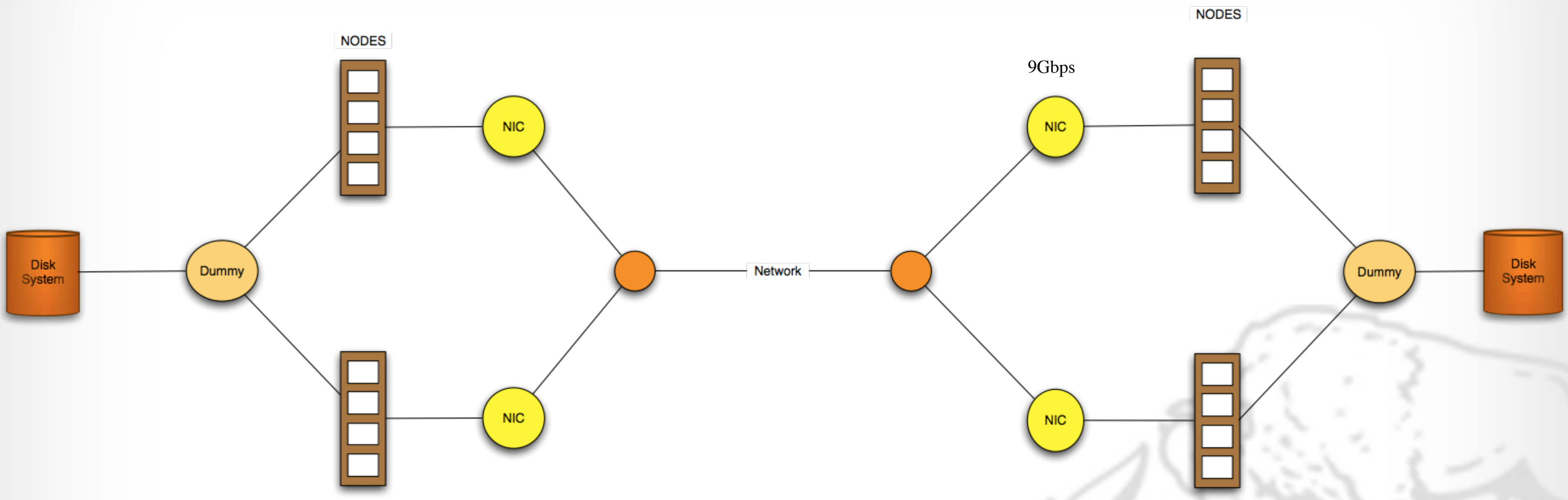
  - ***a*** and ***b*** variables are solved by using the sampling throughput and CPU utilization measurements in regression of method of least squares

$$Ucpu = a + b \times Th$$

OPT_B Algorithm for Homogeneous Resources

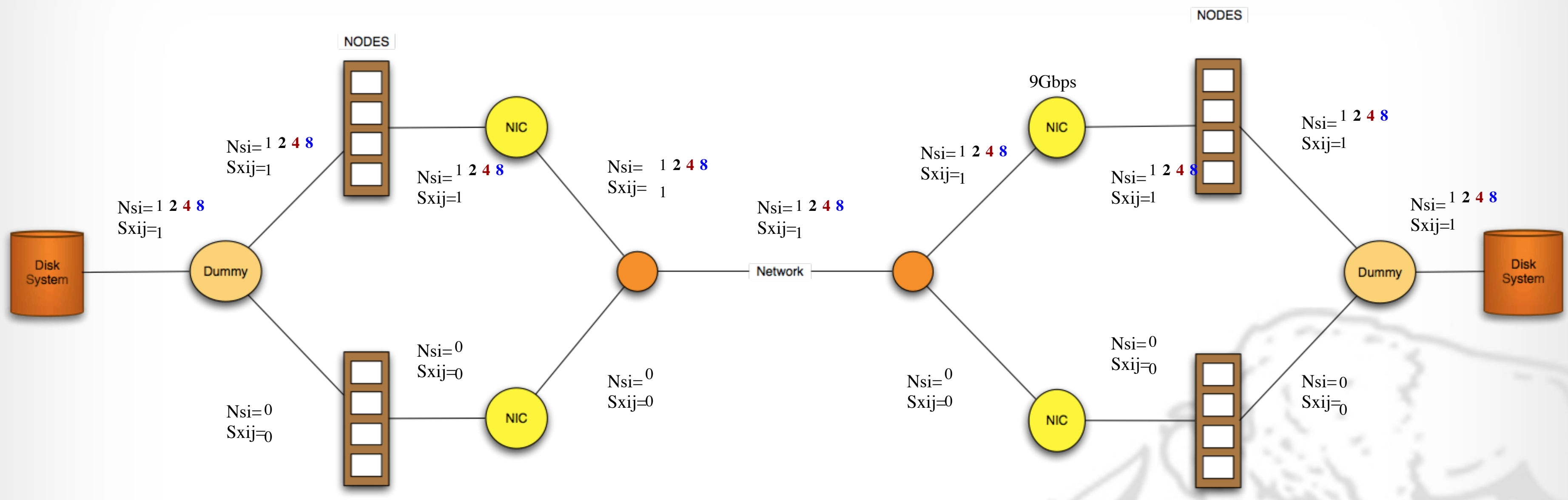
- This algorithm finds the best parallelism values for maximal throughput in homogeneous resources
- Input parameters:
  - A set of sampling values from sampling algorithm (Th_N)
  - Destination CPU, NIC capacities (U_{CPU}, U_{NIC})
  - Available number of nodes (N_{avail})
- Output:
  - Number of streams per stripe (N_{si})
  - Number of stripes per node (S_x)
  - Number of nodes (N_n)
- Assumes both source and destination nodes are idle

OPT_B-Application Case Study

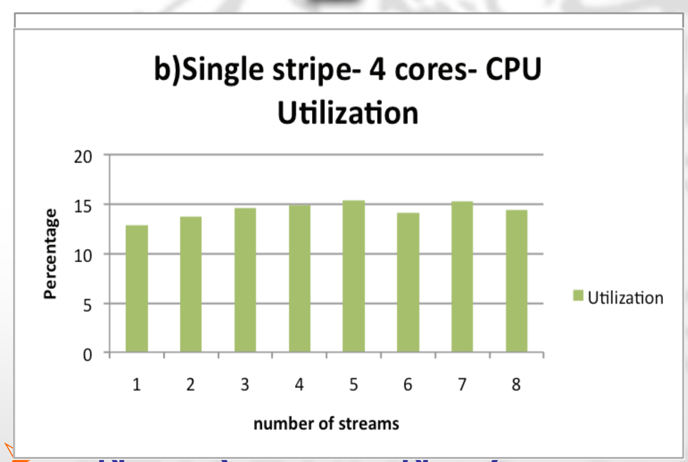


- Systems: Oliver, Eric
- Network: LONI (Local Area)
- Processor: 4 cores
- Network Interface: 10GigE Ethernet
- Transfer: Disk-to-disk (Lustre)
- Available number of nodes: 2

OPT_B-Application Case Study

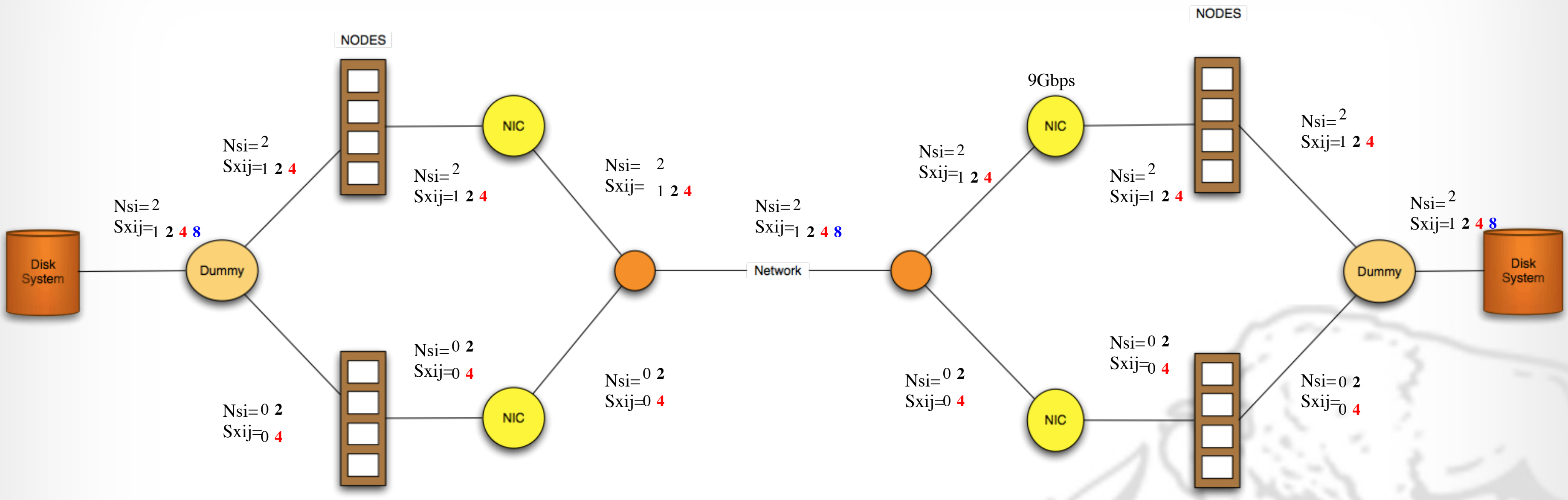


- Th_{Nsi}=903.41Mbps p=1
- Th_{Nsi}=954.84 Mbps p=2
- Th_{Nsi}=990.91 Mbps p=4
- Th_{Nsi}=953.43 Mbps p=8

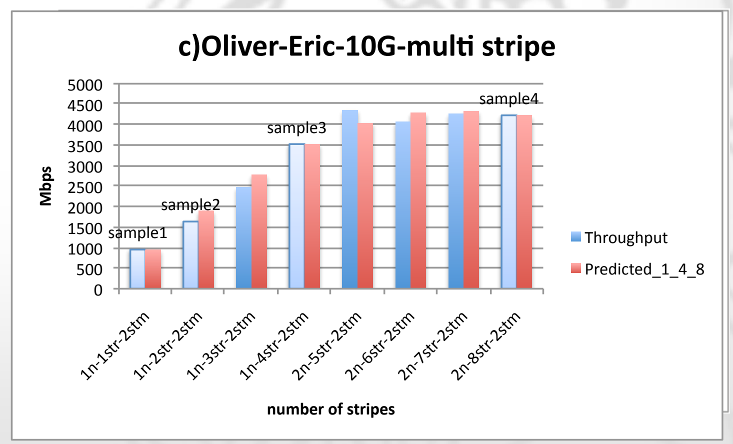


- $N_{opt}=5$ $N_{si}=2$

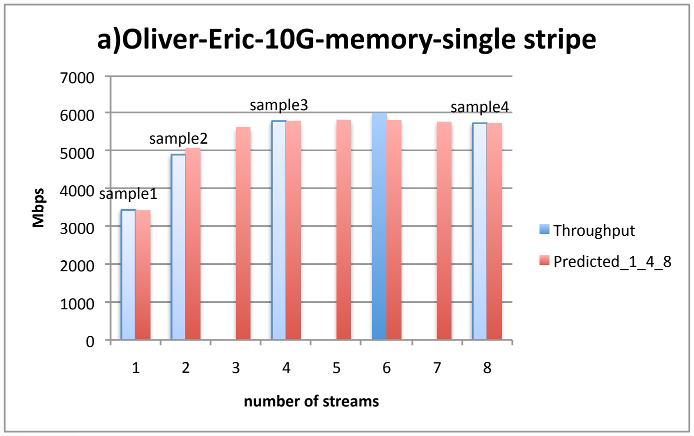
OPT_B-Application Case Study

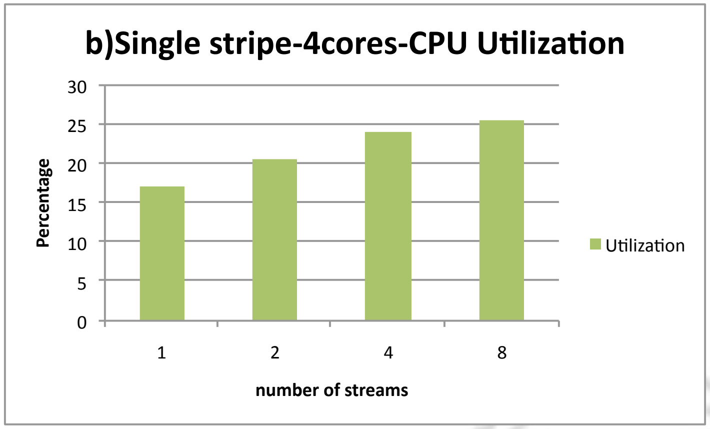


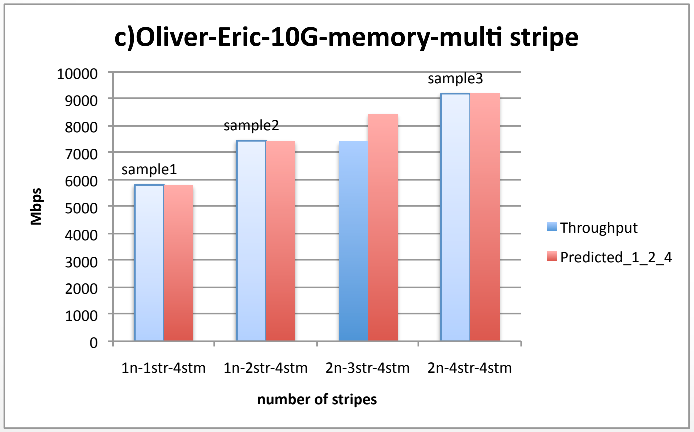
- S_x=2 $Th_{Sx1,2,2}$=1638.48
- S_x=4 $Th_{Sx1,4,2}$=3527.23
- S_x=8 $Th_{Sx2,4,2}$=4229.33

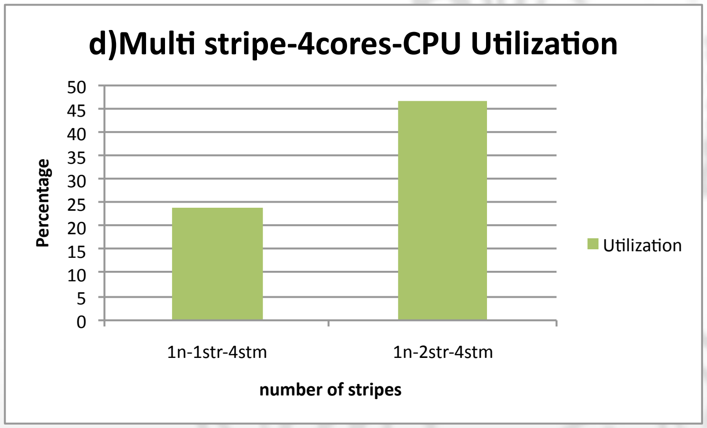


OPT_B-LONI-memory-to-memory-10G

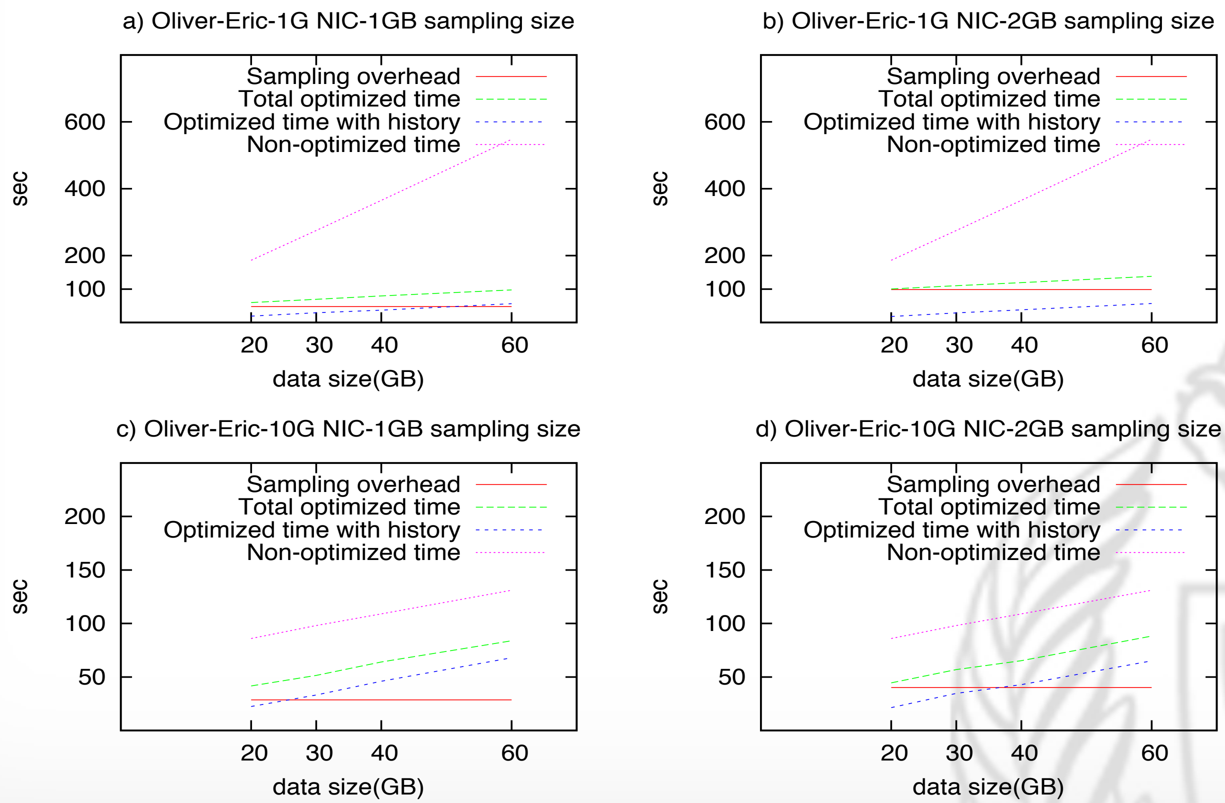








OPT$_B$-LONI-memory-to-memory-1G-Algorithm Overhead



Conclusions

- We have achieved end-to-end data transfer throughput optimization with data flow parallelism
  - Network level parallelism
    - Parallel streams
  - End-system parallelism
    - CPU/Disk striping
- At both levels we have developed models that predict best combination of stream and stripe numbers

Future work

- We have focused on TCP and GridFTP protocols and we would like to adjust our models for other protocols
- We have tested these models in 10G network and we plan to test it using a faster network
- We would like to increase the heterogeneity among the nodes in source or destination

Acknowledgements

- This project is in part sponsored by the National Science Foundation under award numbers
  - CNS-1131889 (CAREER) – *Research & Theory*
  - OCI-0926701 (Stork) – SW Design & Implementation
  - CCF-1115805 (CiC) – Stork for Windows Azure
- We also would like to thank to Dr. Miron Livny and the Condor Team for their continuous support to the Stork project.
- http://www.storkproject.org