Integrating Globus Online With Condor

Zach Miller
Condor Project
Computer Sciences Department
University of Wisconsin-Madison

Overview

- What existed in 2010
- What's new in 2011
- Problems, Solutions, and Future Work

Changes to Condor

- Condor began supporting Globus Online in early 2010 (back when it was globus.org)
- Condor uses a general plugin architecture to support additional methods of file transfer

Changes to Condor

- Each data transfer plugin advertises its own capabilities to Condor:

```
% ./curl_plugin -classad
PluginType = "FileTransfer"
SupportedMethods = "http,ftp,file"

% ./go_plugin.pl -classad
PluginType = "FileTransfer"
SupportedMethods = "globusonline"
```

Changes to Condor

- Condor is then configured with a list of plugins it can use in the condor_config file:
- `FILETRANSFER_PLUGINS = \`

  `η /path/to/curl_plugin, \ /path/to/go_plugin.pl`

Changes to Condor

- Once the plugins are registered in the condor_config file, users can start using them in their submit files
- The first round of changes to Condor supported transferring job input files only
- This exists in the 7.4.X stable series of Condor

New Changes to Condor

- However, in Condor version 7.5.5 support was added to transfer output files using plugins as well
- Condor job input and output can now be entirely done using globusonline
- Condor 7.6.0 is due Any Day Now(TM) and will be the first stable release including the new features

Example Submit File

- `universe = vanilla`
- `should_transfer_files = ALWAYS`
- `when_to_transfer_output = ON_EXIT_OR_EVICT`
- `transfer_input_files = zkm.content, globusonline://zmiller@NEWBIO01:/scratch.1/blast/db/nr`
- `output_destination = globusonline://zmiller@NEWBIO01:/scratch.1/zmiller/tmp/`
- `executable = newz.sh`
- `arguments = 30`
- `x509userproxy = /tmp/x509up_u24842`
- `queue`

Assumptions

- In order to work, some things must already be in place:
  - User must have a valid X.509 proxy
  - User must already have a globusonline account
  - At least one endpoint must be configured in the user's globus online account

How the Plugin Works

- Starts a "personal" gridftp server using the user's proxy
- Registers the new server as a new globusonline endpoint
- Activates the endpoint with the user's proxy
- Performs the transfer to/from job sandbox
- Removes the endpoint
- Shuts down the gridftp server

More Changes to Condor

- Condor treats the job's proxy as an input file, but gives it special treatment (i.e. delegation over the wire)
- However, transferring other input files with plugins may require the proxy to already be in place
- Condor was modified to always move the proxy file first

More Changes to Condor

- Added support for the new attribute "output_destination" to the Condor submit file
- The default behavior is to then transfer all the job's output to the URL specified
- Can be used in conjunction with "transfer_output_files" to specify a subset of output files

Some Limitations

- By default, when Condor performs credential delegation, it creates "limited" proxies
- However, creating a dynamic endpoint requires delegating the credential again to the globusonline service
- This is not allowed with limited proxies
- You need to instruct Condor not to create limited proxies if you plan to use globusonline

Some Limitations

- Currently, input files can come from several different places, using different plugins
- All output files that are transferred must go to the same URL
- It would be great to be able to send different files using different plugins

Some Limitations

- Condor currently invokes file transfer plugins once for each file
- This creates more overhead than necessary because of standing up, registering, activating, and removing endpoints
- Particularily when using globusonline, Condor could batch all files into one transfer
- This would require modification to Condor's plugin API

Some Limitations

- Error propagation needs improvement
- gsissh doesn't fail just because the "scp" command issued to globusonline fails
- Condor relies on the globusonline deadline, which is difficult to select a default for in the context of running Condor jobs

Questions?

- http://www.cs.wisc.edu/condor/
- Email me: zmiller@cs.wisc.edu

THE INTEGRATION OF GLIDEINWMS WITH GLOBUSONLINE.ORG

Parag Mhashilkar
Computing Division, Fermi National Accelerator Laboratory

Overview

- Introduction
- CEDPS Activity at FNAL
- glideinWMS
- Interfacing glideinWMS with globusonline.org
- Asynchronous sandbox management
- Sandbox management at the application layer
- Sandbox management with Condor
- Ongoing & Future Work
- Acknowledgements

Introduction

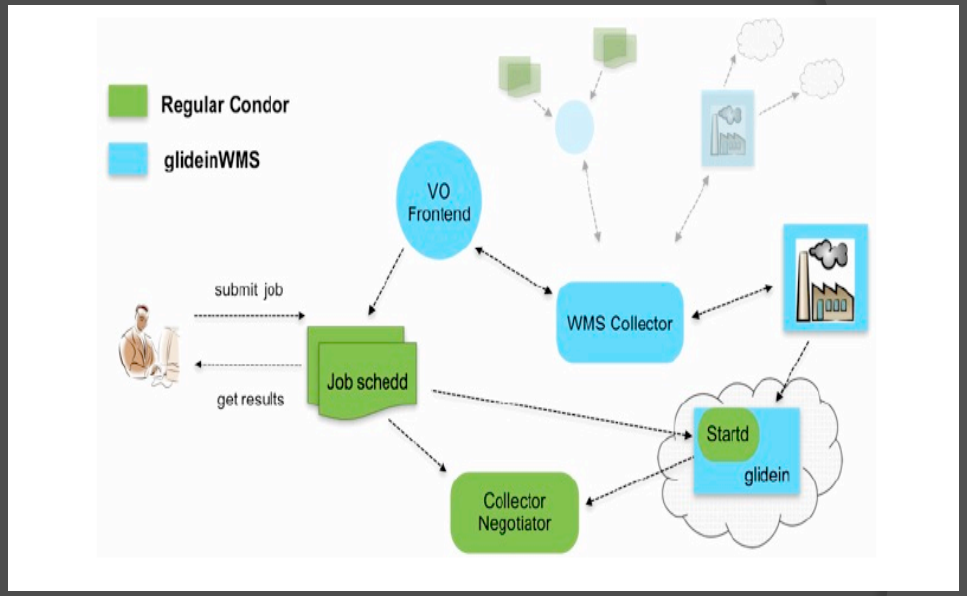
- CEDPS: The five year project started in 2006, funded by Department of Energy (DOE)
- Goals
  - Produce technical innovations for rapid and dependable **data placement** within a distributed high performance environment and for the construction of **scalable science services** for data and computing from many clients.
  - Address performance and functionality troubleshooting of these and other related distributed activities.
- Collaborative Research
  - Mathematics & Computer Science Division, Argonne National Laboratory
  - Computing Division, Fermi National Accelerator Laboratory
  - Lawrence Berkeley National Laboratory
  - Information Sciences Institute, University of Southern California
  - Dept of Computer Science, University of Wisconsin Madison

CEDPS Activities at FNAL

- Investigating data movement mechanisms for data stage-out on Grids
  - globusonline.org needs integration with SRM interface for OSG
- Supporting the integration of data movement mechanisms with scientific DH frameworks
  - Supporting the integration of globusonline.org with Dark Energy Survey (DES) data handling system
- Integration of asynchronous data stage-out mechanisms in overlay workload management systems (…this talk…)
  - Release resources at job termination. Delegate data stage-out to external agents.
  - Integrate support for globusonline.org into glideinWMS

glideinWMS

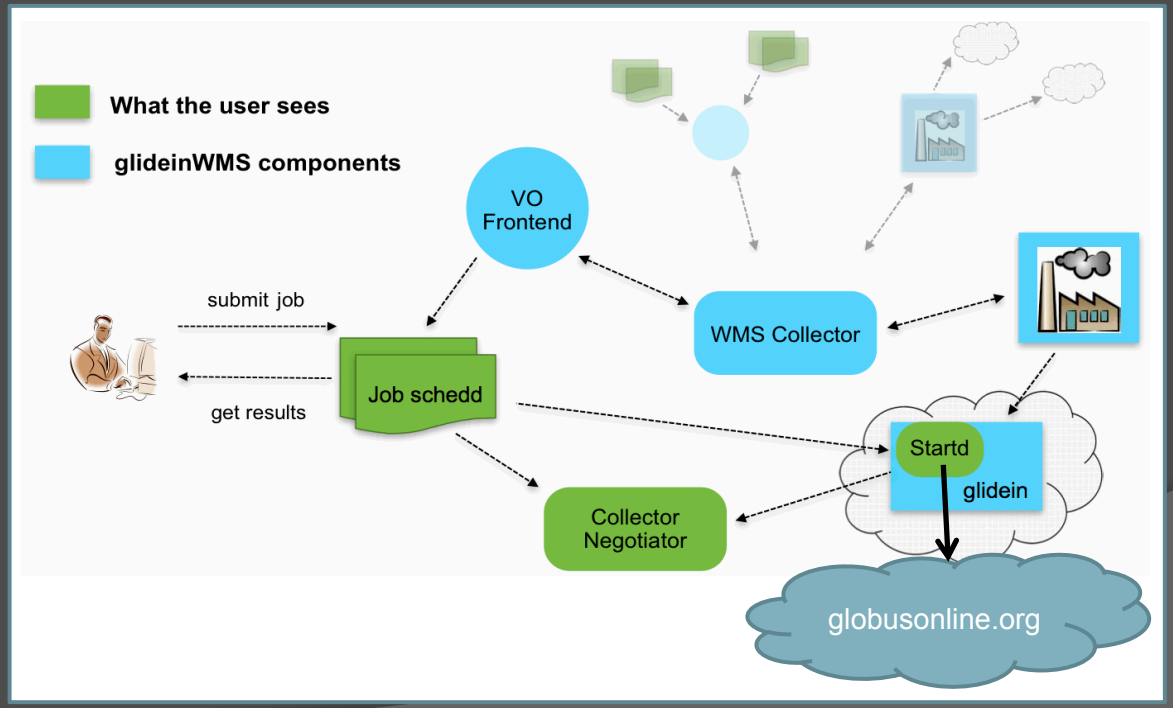
- Pilot-based WMS that creates on demand a dynamically-sized overlay condor batch system on Grid resources to address the complex needs of VOs in running application workflows
- Components
  - WMS Collector
  - Glidein Factory
  - User Pool Collector
  - User Scheduler
  - VO Frontend



- Factory knows about the sites and how to submit glideins to the sites
- VO frontend knows about the user job details
- WMS Collector acts as a dashboard for Factory - VO Frontend communication.

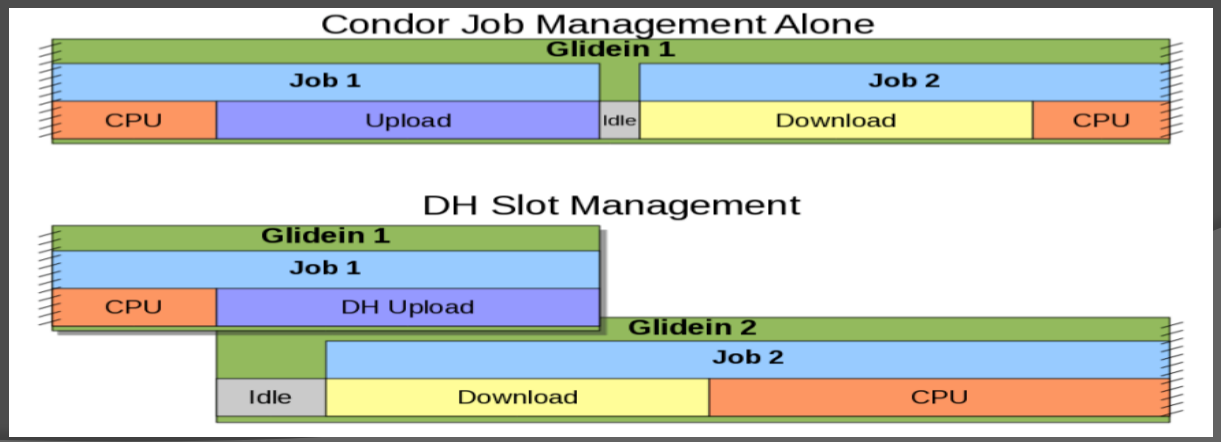
Interfacing glideinWMS with globusonline.org

- glideinWMS is the workload management system for user jobs
- globusonline.org services are responsible for data management
- glideinWMS interfaces with the globusonline.org services through condor's globusonline.org transfer plug-in
  - Work done in collaboration with the Condor team



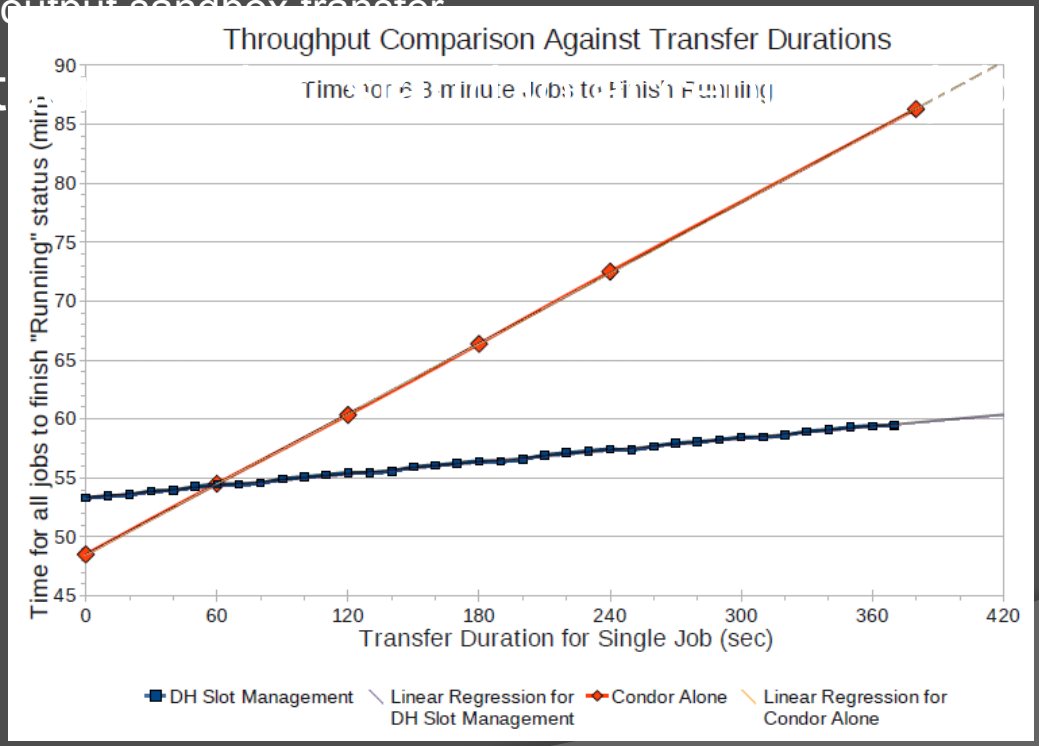
Asynchronous Sandbox Management

- Enhance glideinWMS by
  - Increasing the CPU utilization of Condor-managed resources in a wide area environment through CPU and network I/O overlap enabled by asynchronous transfers of sandboxes – Miron Livny
- What does this mean?
  - Pipeline the transfer of asynchronous sandboxes in Condor using globusonline.org
  - Multiple transfers can take place via transfer slots
  - New job can start running if the previous job has entered stage-out state
  - Reattempt failed transfers as needed
  - Support multiple transfer protocols using transfer Plug-ins



First Prototype: Condor Hooks

- Support at the Application Layer implemented by Evan Boldt.
- Use Condor Hooks
  - To identify end of CPU stage
  - Initiate the output sandbox transfer
- Start anot



Asynchronous Sandbox Management in Condor

- Rather than the application, let condor transfer the output sandboxes asynchronously
  - Generic
  - Robust
  - Reliable
  - Scalable
- Collaborative Effort
  - Condor team working on output sandbox management
  - Fermi team is implementing the Sandbox Manager
    - A repository for sandboxes
    - Condor_startd interfaces with the sandbox manager to keep track of output sandboxes

Ongoing & Future Work

- Sandbox Manager & condor
  - **Milestone 1 (3/11):** Startd maintains the slot manager
    - Interface the sandbox manager with the condor_startd
  - **Milestone 2 (3/24): Bind CPU slot to data slot**
    - Introduce the concept of the data slot
  - **Milestone 3 (4/25): Handle sandbox transfer semantics**
    - Make scheduler aware of the new sandbox transfer semantics
  - **Milestone 4+: Improve scalability & robustness**
- Provide an end-to-end solution for VO applications
  - Involve potential Virtual Organizations (VOs) like Intensity Frontiers (IF) and any VO interested in end-to-end solution
  - Possible deployment on OSG Integration testbed for new VOs
- Testing sandbox manager and integration with glideinWMS
  - Summer student

Acknowledgments

- The work is being done as a part of the CEDPS project
- The GlideinWMS project was developed and is supported by Fermilab Computing Division and the CMS Experiment
- This work is an ongoing collaboration with the Condor team at University of Wisconsin-Madison
- Currently used in production by CMS, CDF and DZero, MINOS, ICECube with several other VO's evaluating it for their use case.
- Fermilab is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.
- The Open Science Grid (OSG) collaborates with CEDPS for solutions on data movement and stage-out

End-to-end Data-flow Parallelism for GridFTP Throughput Optimization

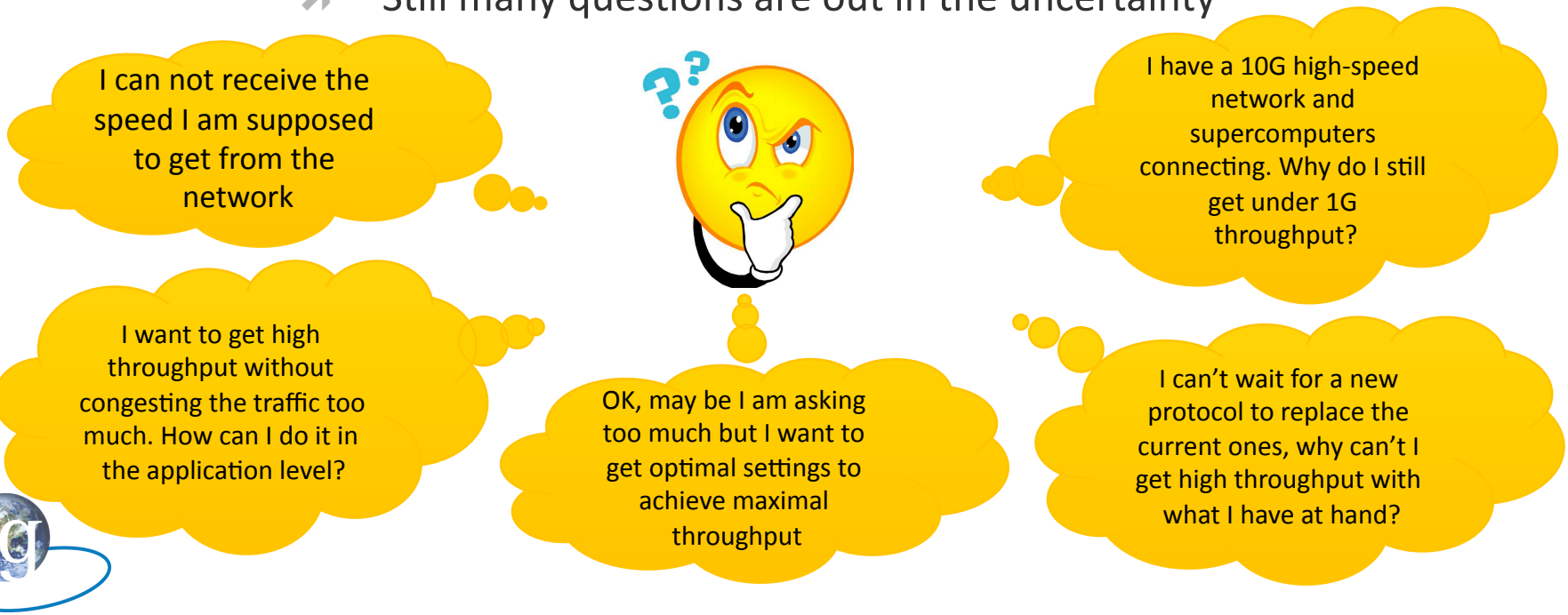
Esma Yildirim

University at Buffalo (SUNY)

GlobusWORLD 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



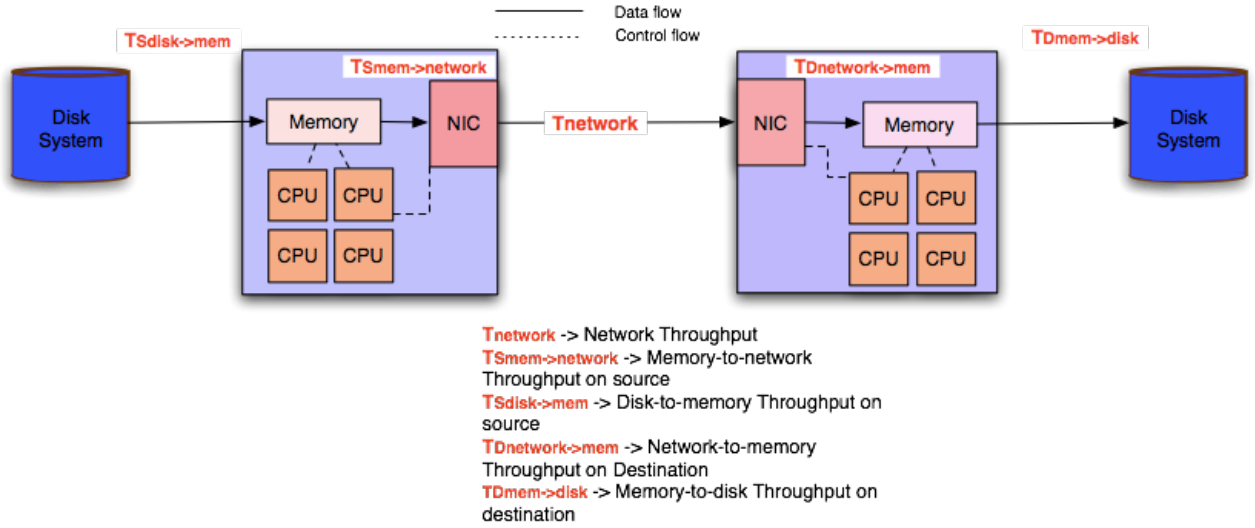
Outline

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

Introduction

- The emerging data management of scientific applications is only possible with end-to-end data transfer capabilities that will support their Petascale problems
- 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 the development of high-speed networks, end-systems have become the major point of bottleneck
  - CPU, NIC and Disk Bottlenecks

End-to-end Data Transfer

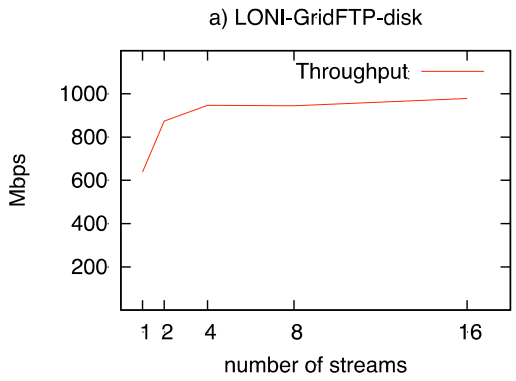


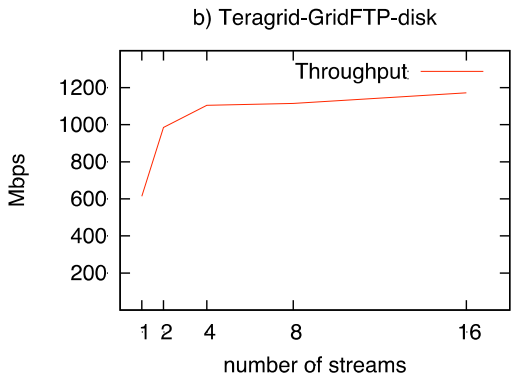
- Method to improve the end-to-end data transfer throughput
  - Application-level Data Flow Parallelism
    - Network level parallelism (parallel streams)
    - Disk/CPU level parallelism (stripes)

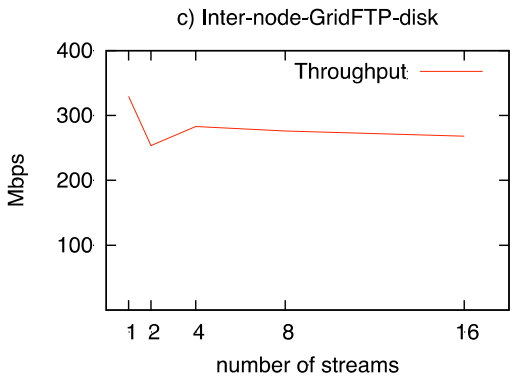
Network Bottleneck

☑ 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



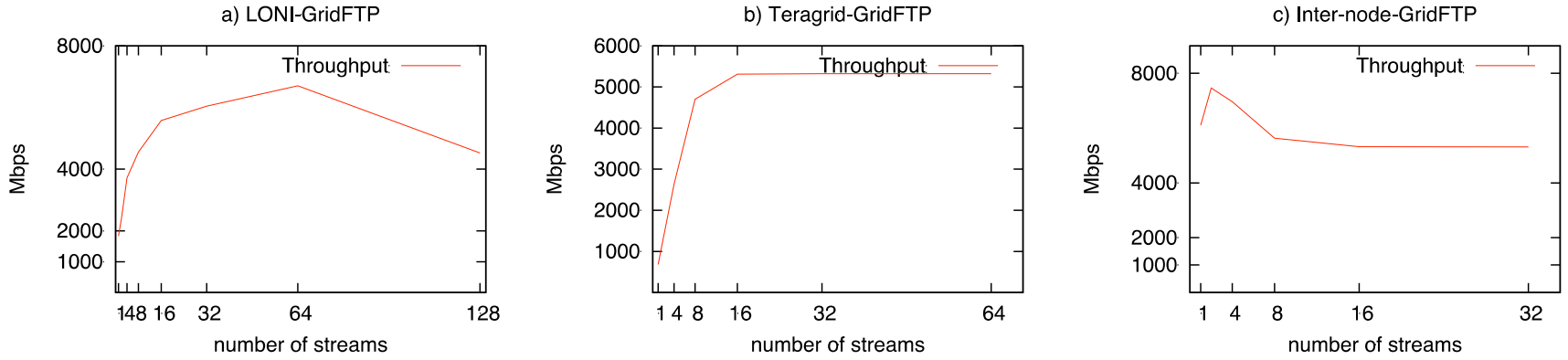




Disk Bottleneck

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

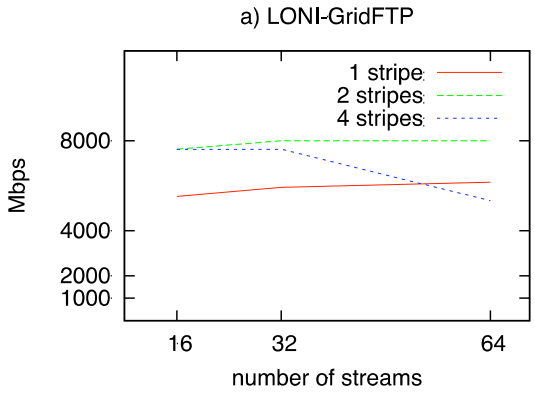
- Once disk bottleneck is eliminated, parallel streams improve the throughput dramatically
- Throughput either becomes stable or falls down after reaching its peak due to network or end-system limitations. Ex:The network interface card limit(10G) could not be reached (e.g.7.5Gbps-internode)

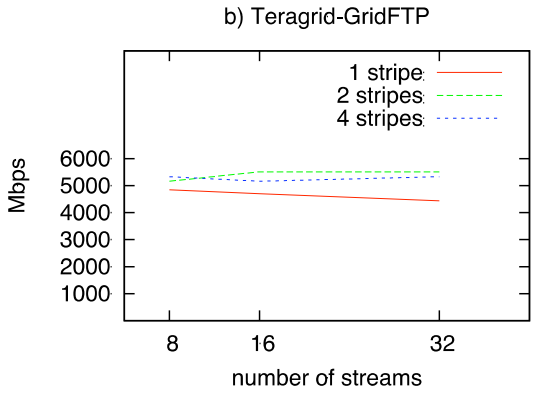


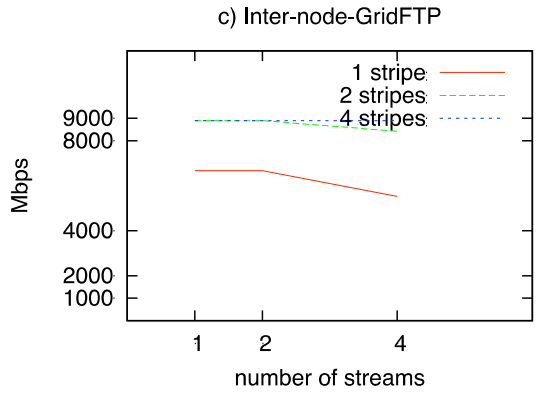
CPU Bottleneck

☑ Step3: Effect of Striping and Removal of CPU Bottleneck

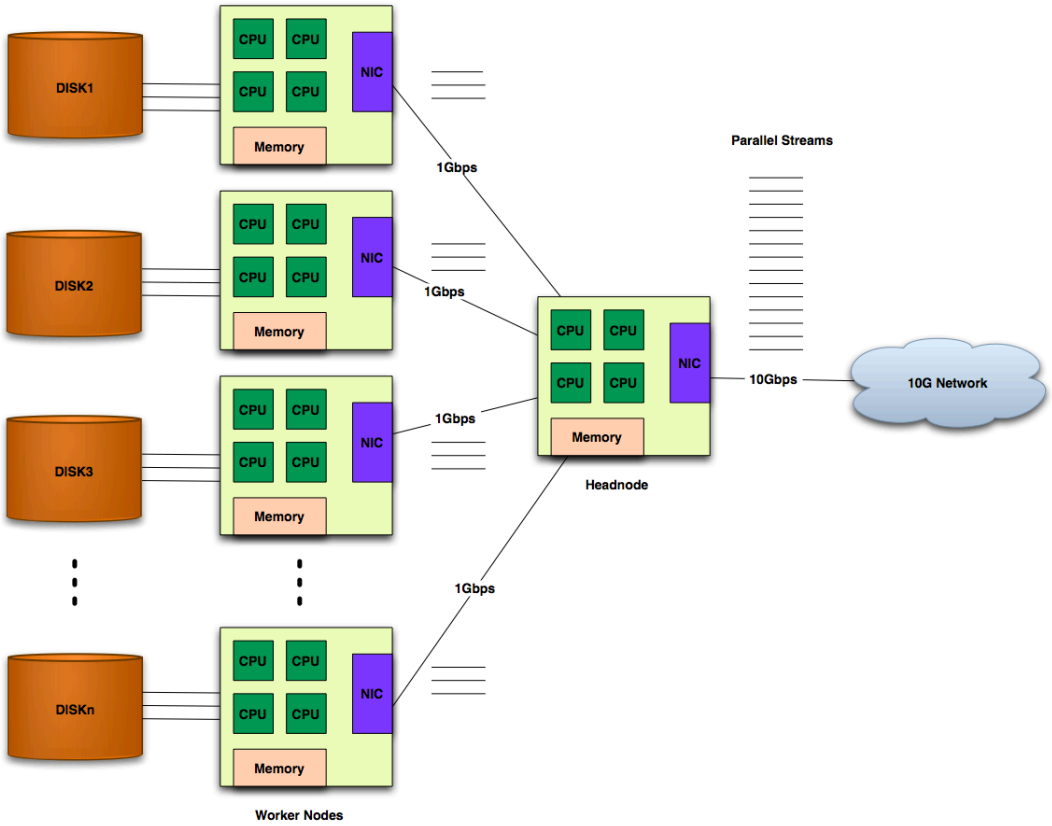
- Striped transfers improves the throughput dramatically
- Network card limit is reached for inter-node transfers(9Gbps)







Data Flow Parallelism

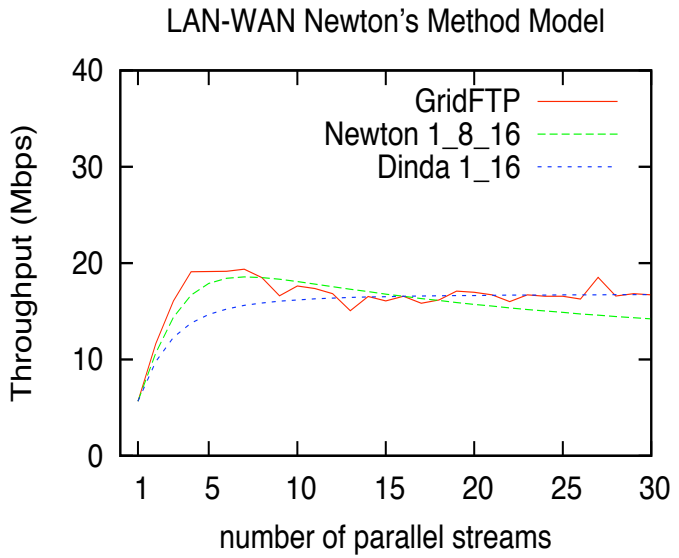


Prediction of Optimal Parallel Stream Number

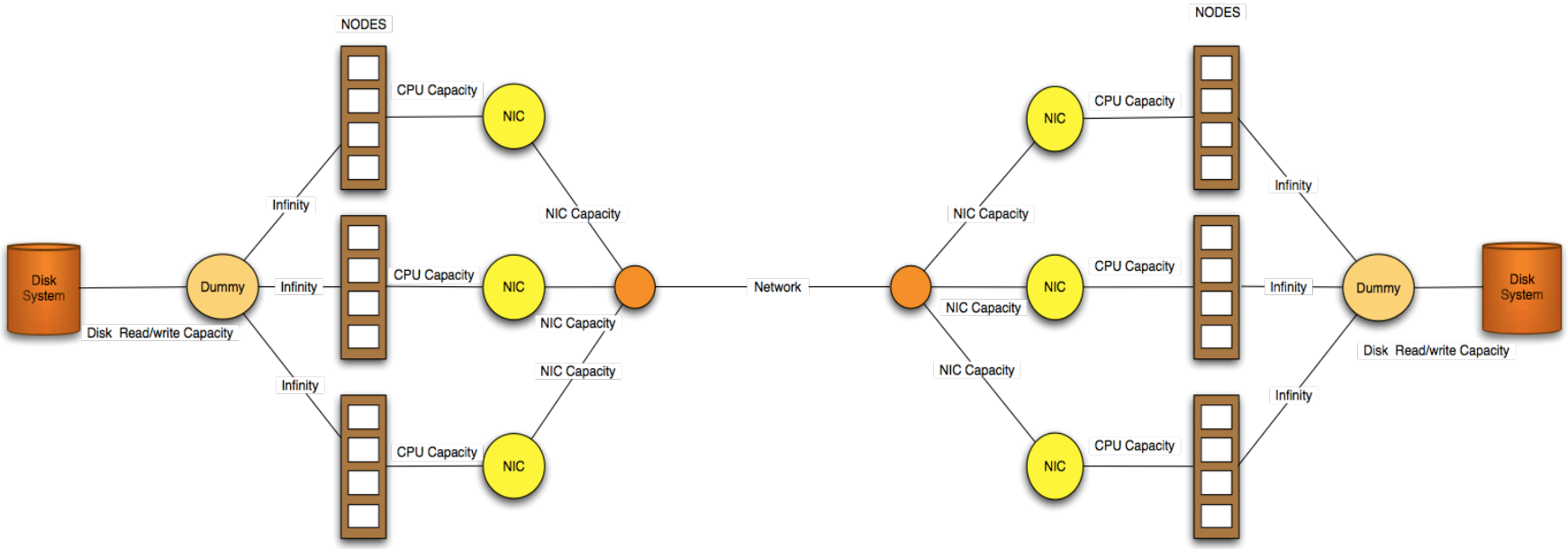
- Throughput formulation : Newton's Iteration Model

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

- 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

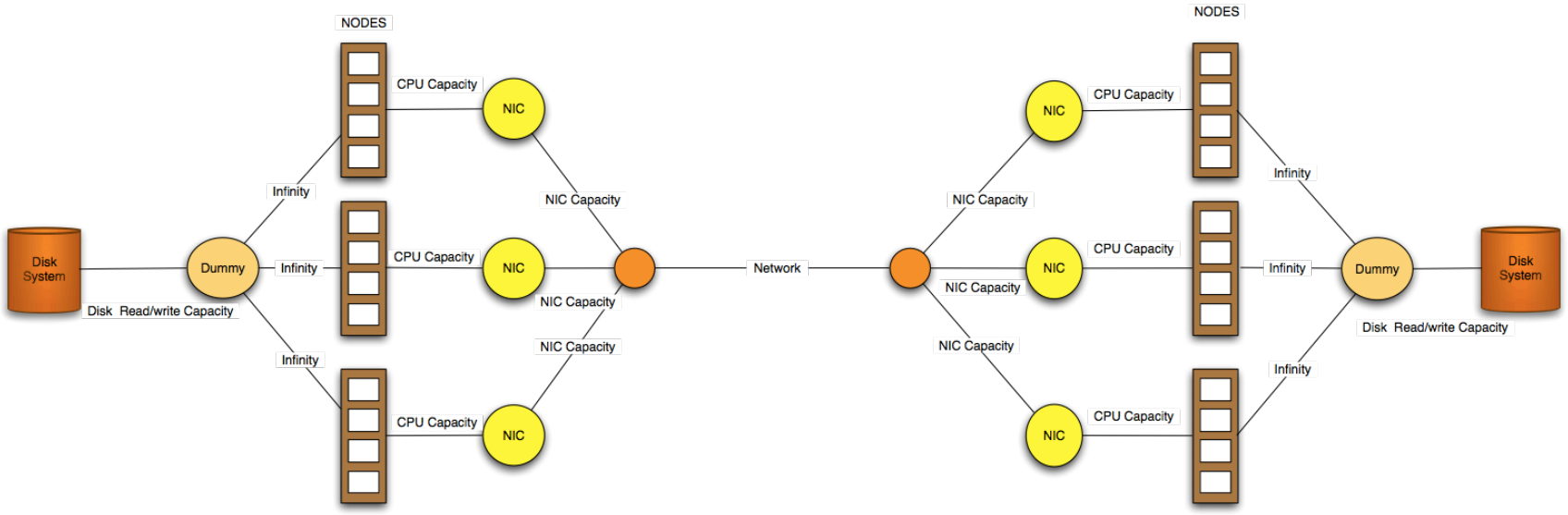


Flow Model of End-to-end Throughput



- 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: $0 \le X_{ij} \le U_{ij}$ $0 \le X_{fk} \le U_f$
- 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 :

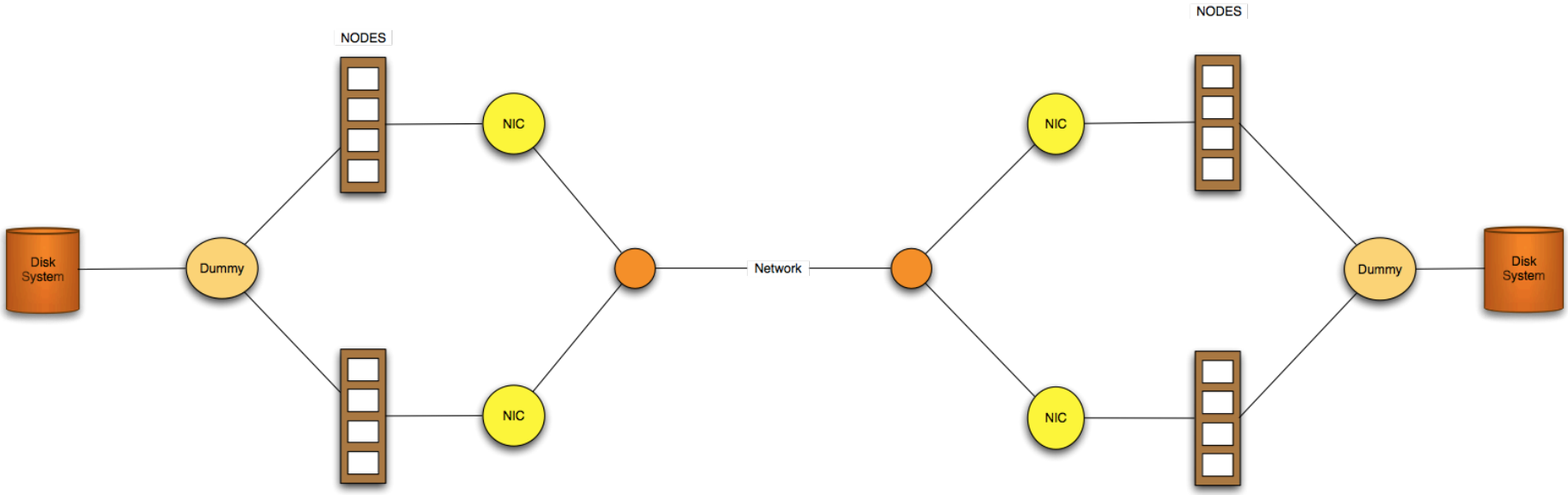
$$Ucpu = a + b \times Th$$

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

OPT_B Algorithm for Homogeneous Resources

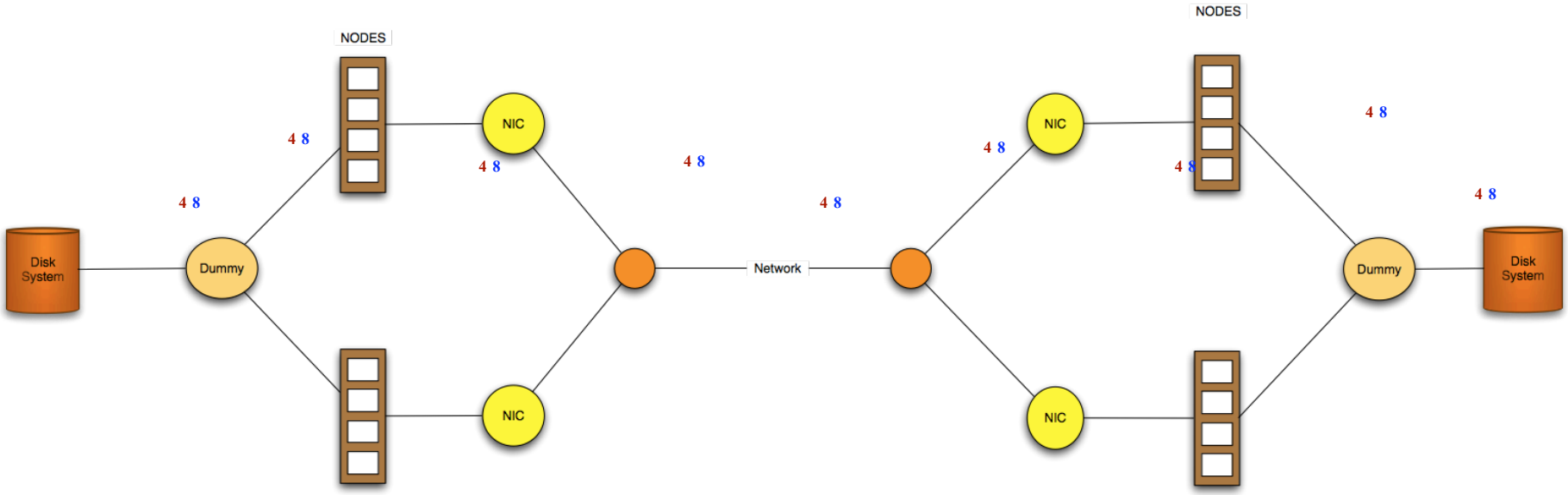
- This algorithm finds the best parallelism values for maximal throughput in homogeneous resources
- Input parameters:
  - A set of sampling values from sampling algorithm (Th_N)
  - Destination CPU, NIC capacities (U_{CPU}, U_{NIC})
  - Available number of nodes (N_{avail})
- Output:
  - Number of streams 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

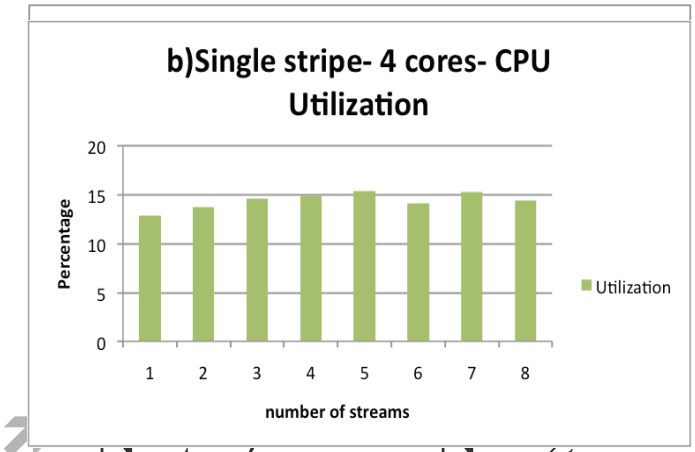


- Systems: Oliver, Eric
- Network: LONI (Local Area)
- Processor: 4 cores
- Network Interface: 10GigE Ethernet
- Transfer: Disk-to-disk (Lustre)
- Available number of nodes: 2

OPT_B-Application Case Study

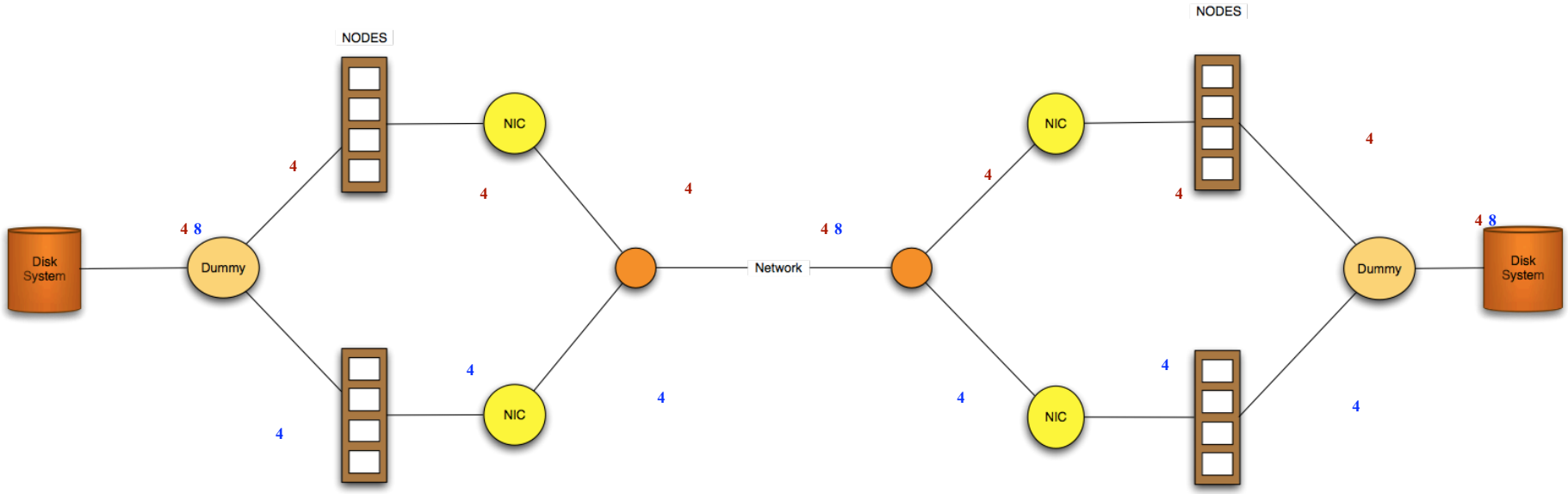


- Th_{Nsi}=903.41Mbps p=1
- Th_{Nsi}=954.84 Mbps p=2
- Th_{Nsi}=990.91 Mbps p=4
- Th_{Nsi}=953.43 Mbps p=8

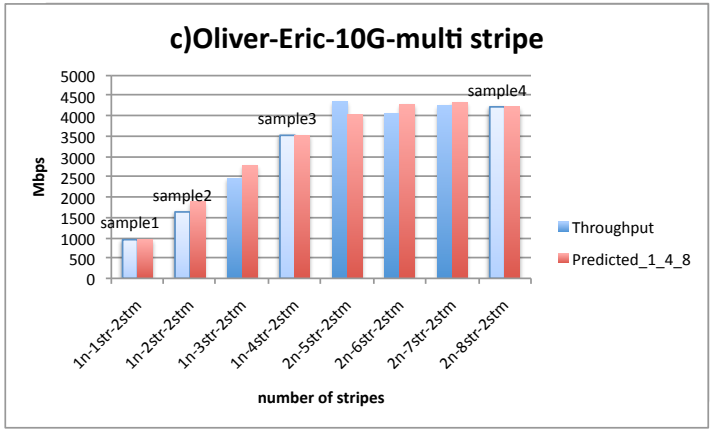


- N_{opt} [illegible] N_{si} 2

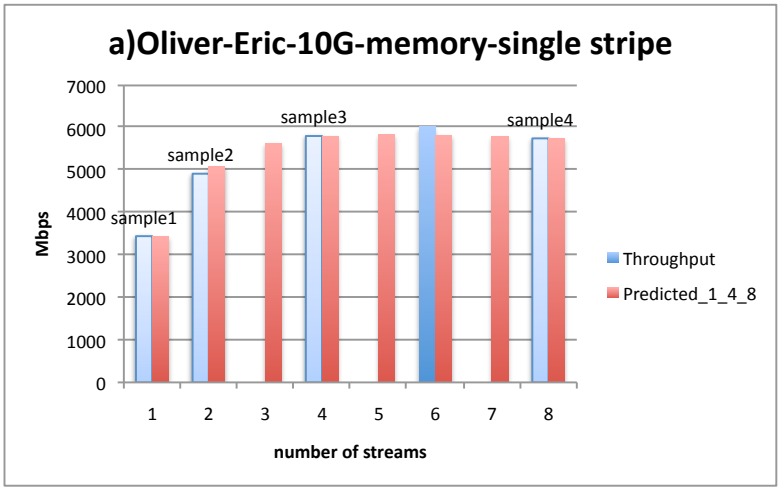
OPT_B-Application Case Study

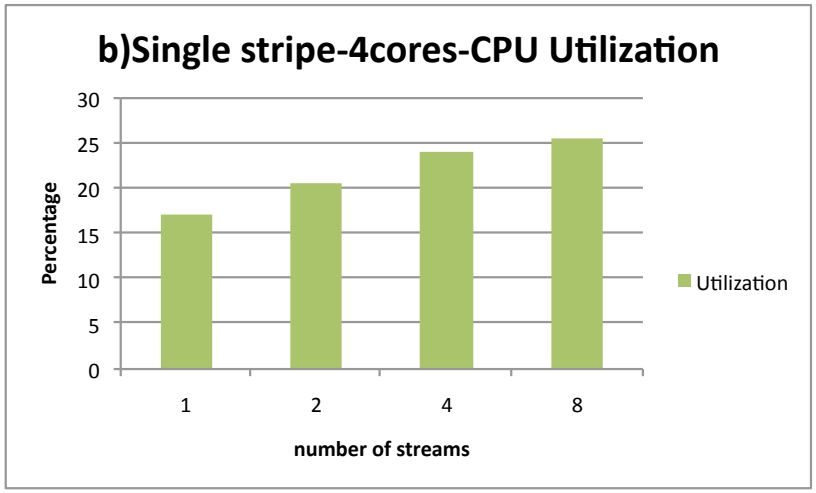


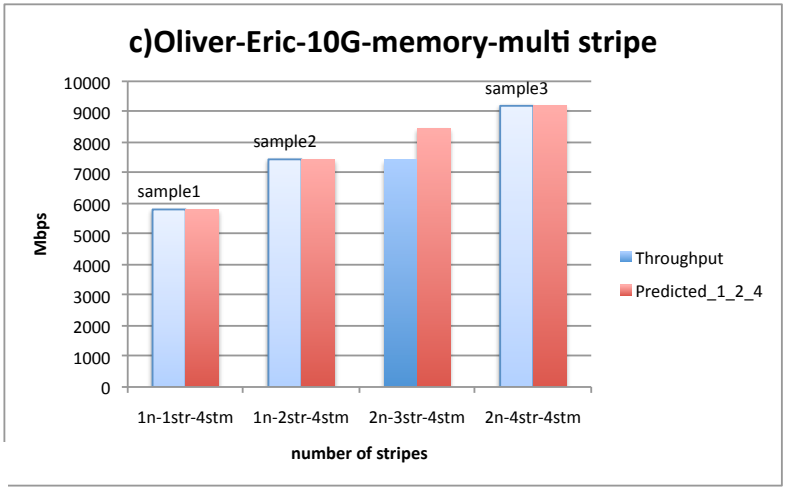
- S_x=2 $Th_{Sx1,2,2}$=1638.48
- S_x=4 $Th_{Sx1,4,2}$=3527.23
- S_x=8 $Th_{Sx2,4,2}$=4229.33

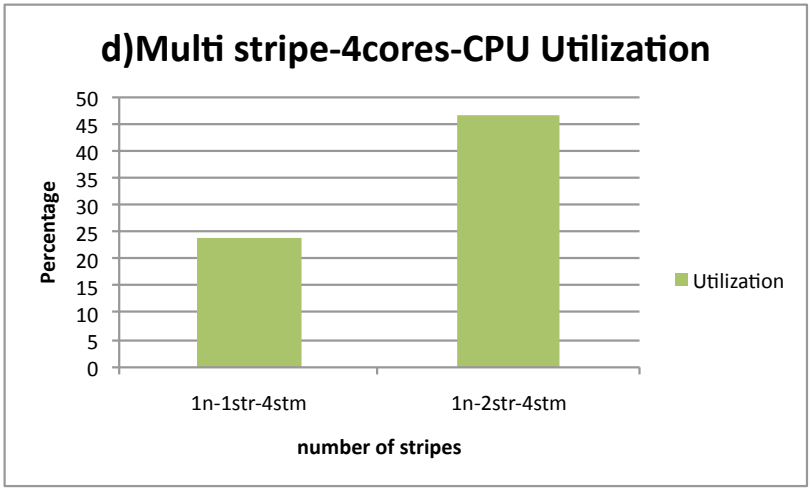


OPT_B-LONI-memory-to-memory-10G



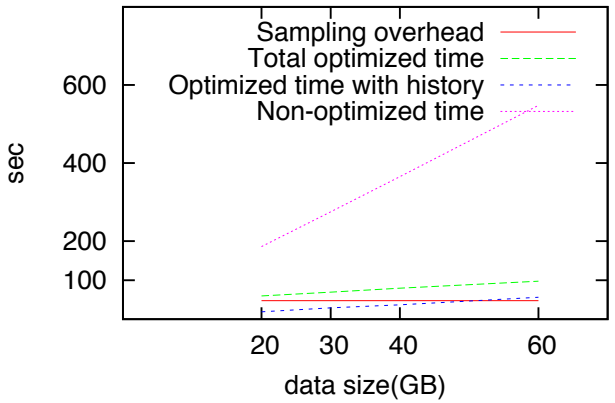




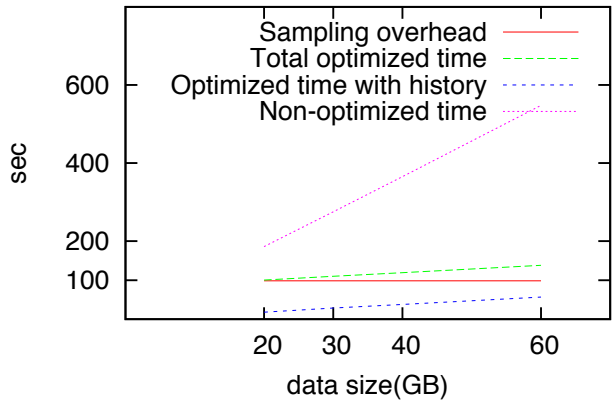


OPT_B-LONI-memory-to-memory-1G-Algorithm Overhead

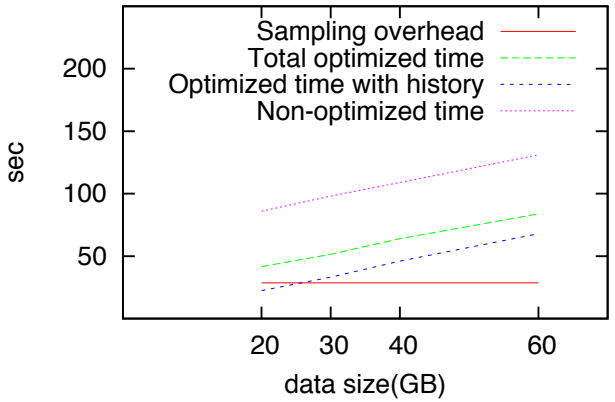
a) Oliver-Eric-1G NIC-1GB sampling size



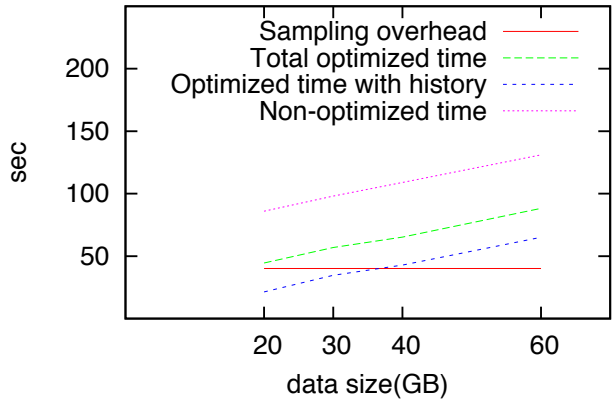
b) Oliver-Eric-1G NIC-2GB sampling size



c) Oliver-Eric-10G NIC-1GB sampling size



d) Oliver-Eric-10G NIC-2GB sampling size



Estimation & Optimization Service in Stork Scheduler

- For a specific data transfer Stork estimates
  - optimal parallel stream number
  - throughput
  - transfer time
- Estimation jobs can be submitted for
  - Memory-to-memory transfers
  - Disk-to-disk transfers

- Sample job submission

```
[
    dap_type = "Estimation";
    src_url  = "gsiftp://$src.loni.org/";
    dest_url = "gsiftp://$dest.dsl-stork.org/";
    x509proxy = "default";
    filesize = "2GB";
]

[
    dap_type = "Estimation";
    src_url  = "gsiftp://poseidon1.loni.org/work/siva/siva";
    dest_url = "gsiftp://eric.loni.org/work/siva/igb";
    disk_transfer ="yes";
    x509proxy = "default";
    arguments = "-s 50M"
]
```

- Sample Output

```
[
  file_size_in bytes = 756;
   optimization_cost = 2.100000000000000E+01;
   max_throughput = 3.521927022370290E+01;
   status = "request_completed";
   dap_id = 17;
   est_time = "0 hours, 3 mins, 1 secs";
   optimal_streams = 1;
   timestamp = absTime("2010-09-22T11:09:15-0500")
]
```

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-0846052 (CAREER)
  - OCI-0926701 (Stork)
- http://www.storkproject.org

Questions