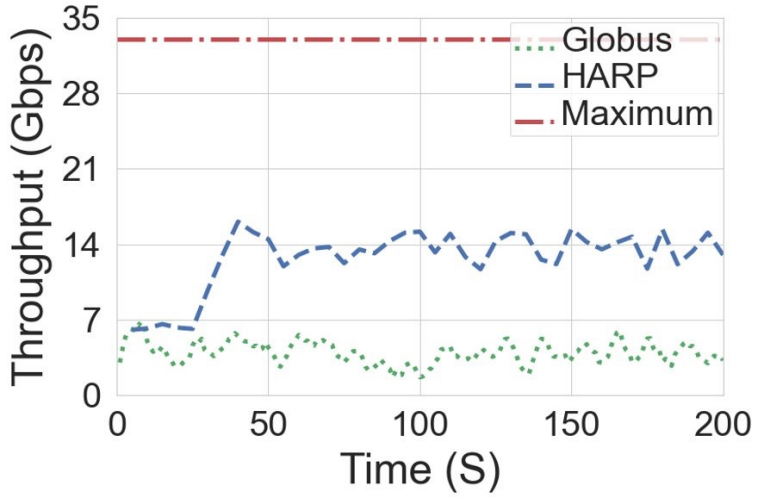
Online Optimization of File Transfers in High-Speed Networks

Md Arifuzzaman and Engin Arslan
University of Nevada, Reno

University of Nevada, Reno

Demand for High-Speed Transfers

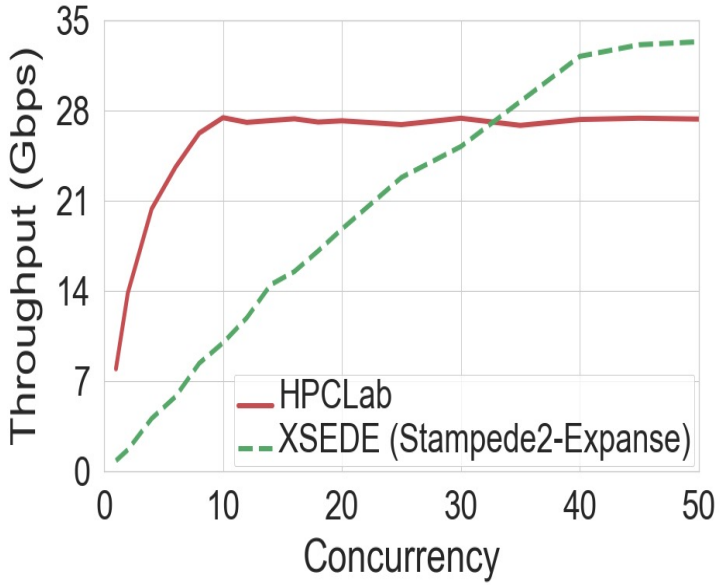
- Science applications are becoming increasingly data-intensive
  - Single high throughput genome sequencing data: From 5MB in 2006 to 5GB in 2018
- Produced data is often transferred in the wide-area for processing or archival
- Research networks (e.g., ESnet and Internet-2) offer high speed connectivity with up-to 100 Gbps bandwidth
- Yet, legacy transfer applications either fail to achieve high performance or cause high overhead and unfair resource allocation



Concurrency to the Rescue!

- Parallelism is key to increase system utilization
  - Single file read/write operations cannot yield more than 12 Gbps throughput
  - Single TCP connection is limited to around 30 Gbps throughput

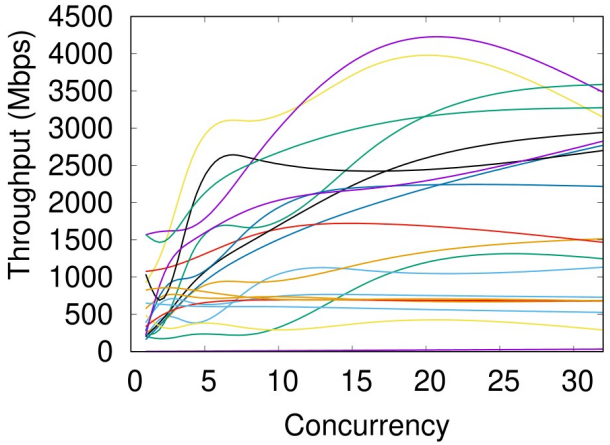
- Concurrent transfer of multiple files (Concurrency) improves transfer throughput by improving both I/O and network throughput
  - From 7 Gbps to 27 Gbps in local cluster (HPCLab)
  - From 1.5 Gbps to 33 Gbps in Stampede2 Cluster



How Much Concurrency is Enough?

- Optimal level of concurrency is not the same in all networks
  - I/O performance is dependent on file system settings, SSD vs HDD
  - High concurrency may be necessary if TCP buffer size is not tuned properly

- Optimal concurrency is not even same for the same network all the time
  - Background traffic, I/O interference, dataset characteristics, etc.



Related Work

Heuristic Models

- Tunes the level of concurrency using heuristic methods
  - Globus Online, Pehlivan et al.(JPDC'18), Arslan et al.(EuroPar'13)

- Advantages:
  - Simple, no need for developing complex models
  - Mostly fair resource allocation between competing transfers

- Limitations:
  - Sub-optimal performance due to using a fixed settings and lack of adaptability to dynamic network conditions

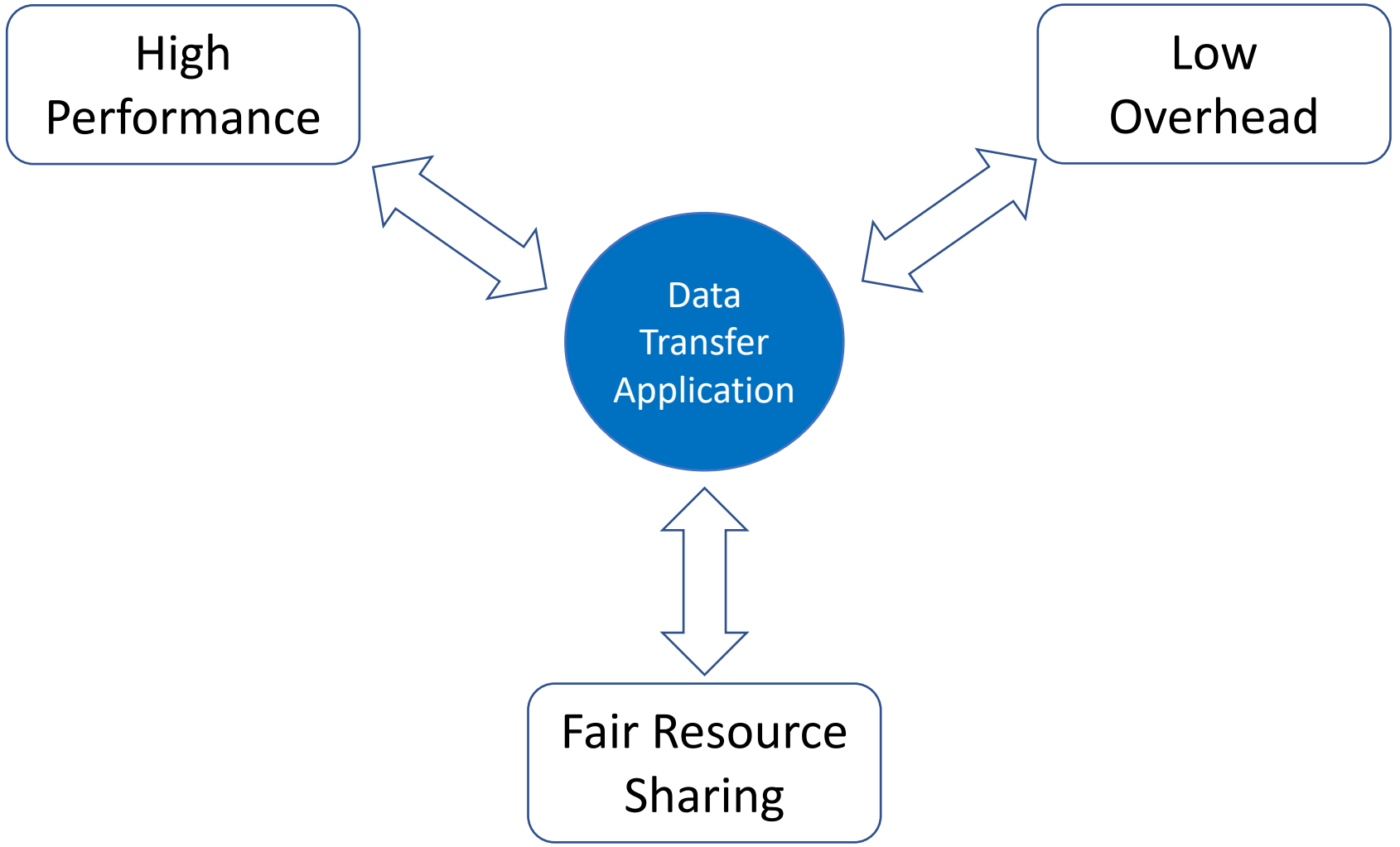
Supervised Models

- Train predictive models using historical logs to correlate transfer settings to throughput
  - Example: Nine et al.(TPDS'20), HARP (SC'16, TPDS'18)

- Advantages:
  - Mostly yields high-performance
  - Ability to react to changing transfer conditions through real-time sampling

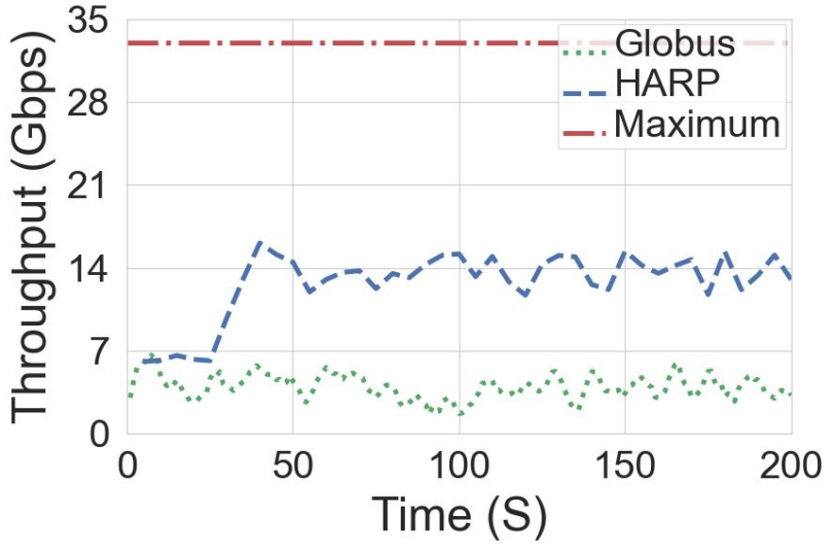
- Limitations:
  - Requires large volume of historical data to derive accurate models
  - Vulnerable to data drift (e.g., network upgrade, file system expansion, etc.)
  - Can cause high overhead on system resources to maximize transfer throughput
  - Potentially unfair resource allocation between competing transfers

Objectives


High Performance
Low Overhead
Data Transfer Application
Fair Resource Sharing

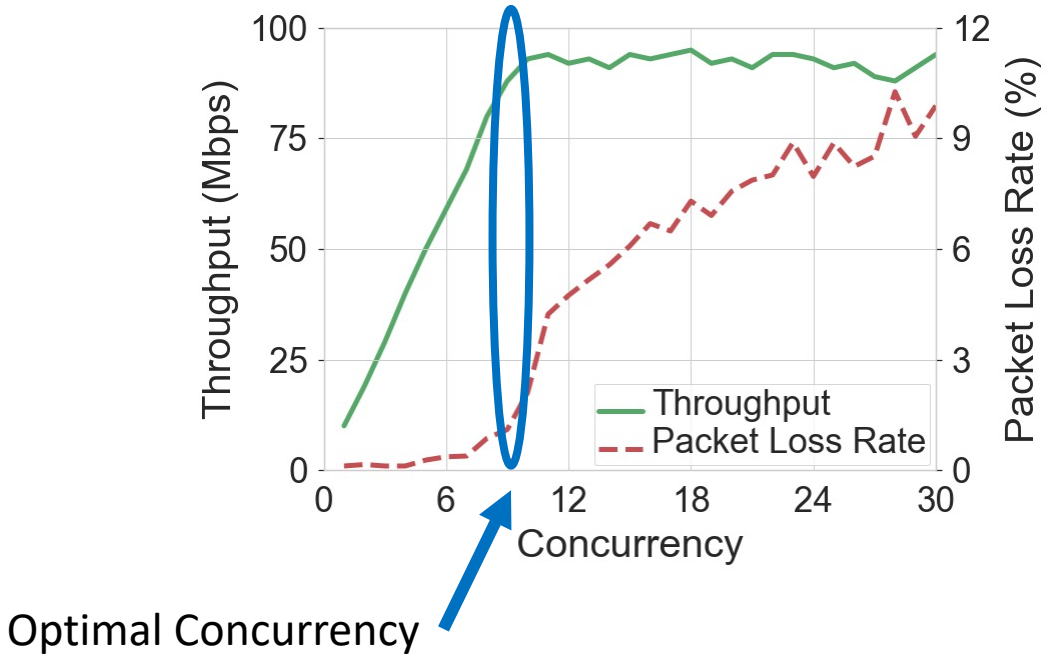
Objective 1: High Performance

Using a fixed and small concurrency levels leads to poor system utilization



Objective 2: Low Overhead

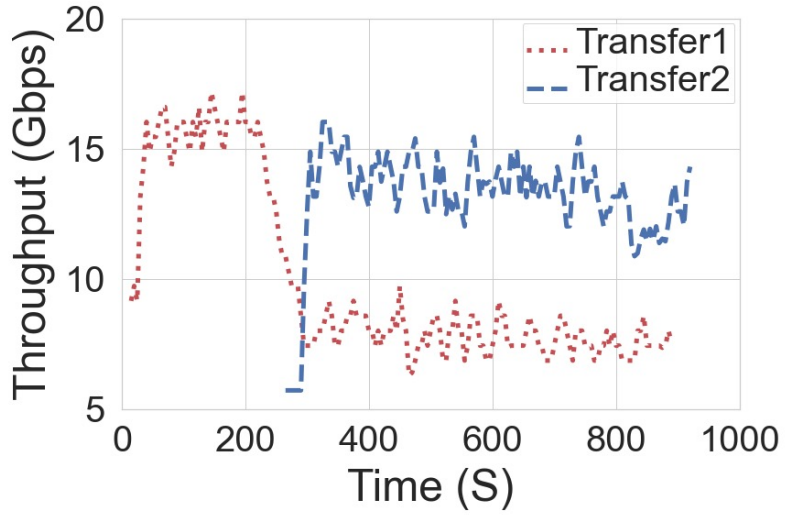
High concurrency levels can overwhelm network, storage, and end hosts



Emulab Testbed
I/O limit: *10Mbps/thread*
Bandwidth: *100Mbps*

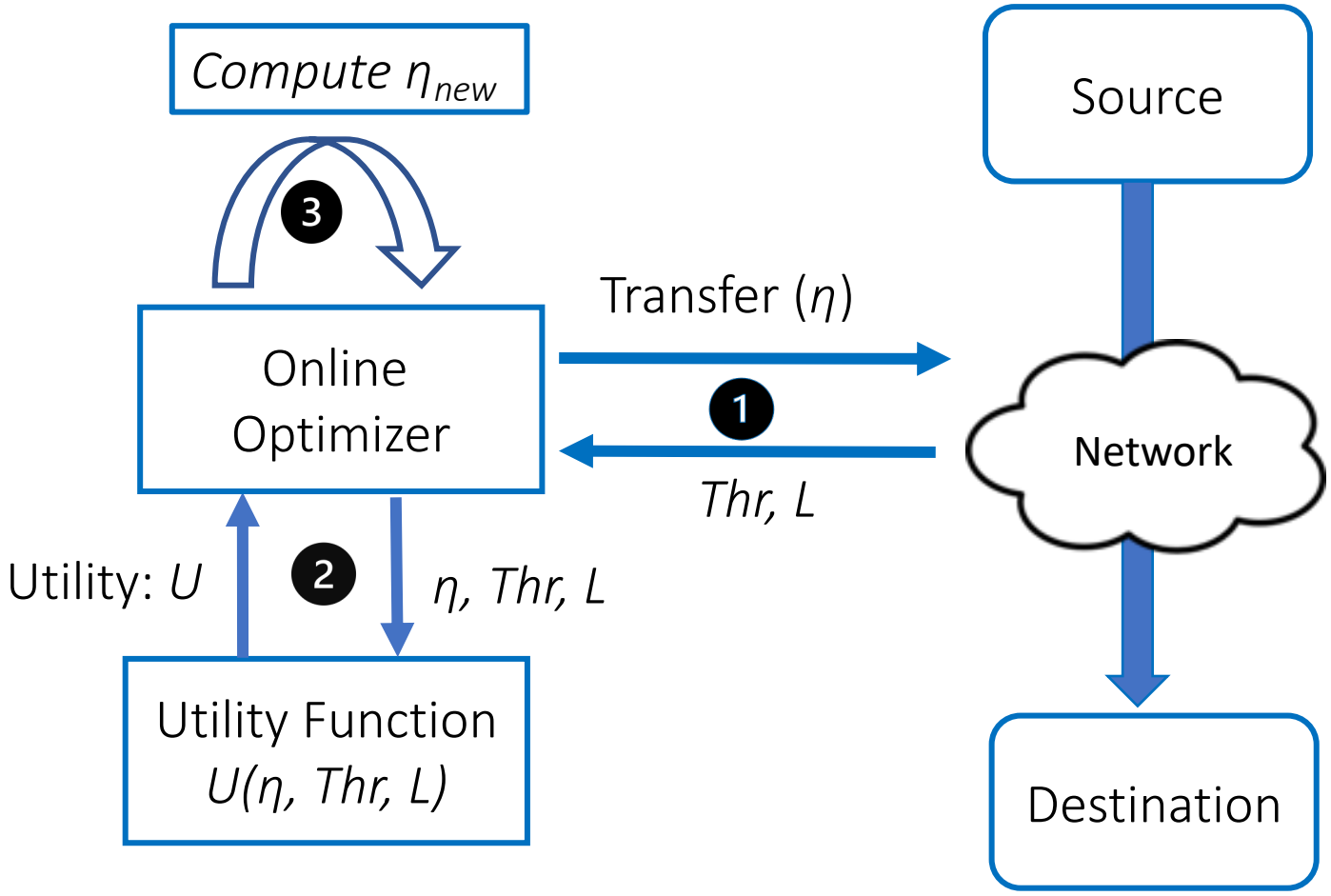
Objective 3: Fair Resource Sharing

Lack of penalty term in optimization functions lead to aggressive behaviors to maximize the gain



Falcon - Online Transfer Optimizer

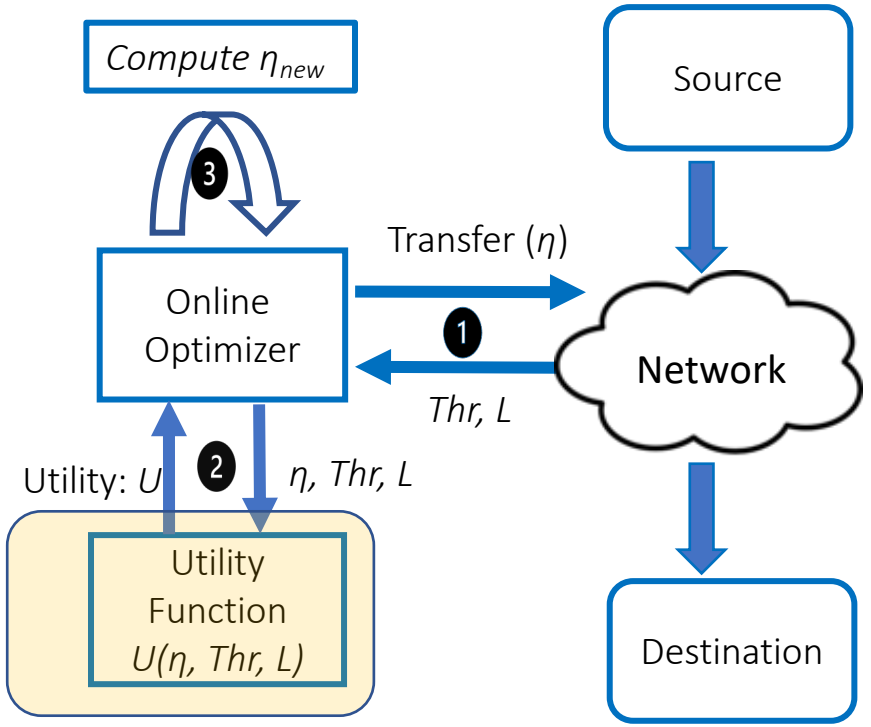
- Online, decentralized, and black-box optimization
- Two major components:
  - Optimization algorithm
  - Utility function



Black-box Optimization

- Poor network visibility
  - Wide-area file transfer travels through multiple independently-controlled domains.
- Avoid making assumptions about hosts, networks, or storage systems
  - Instantly deployable
- Optimization is relatively simple due to being restricted to few observable metrics
  - Throughput, packet loss rate, concurrency value

Utility Function

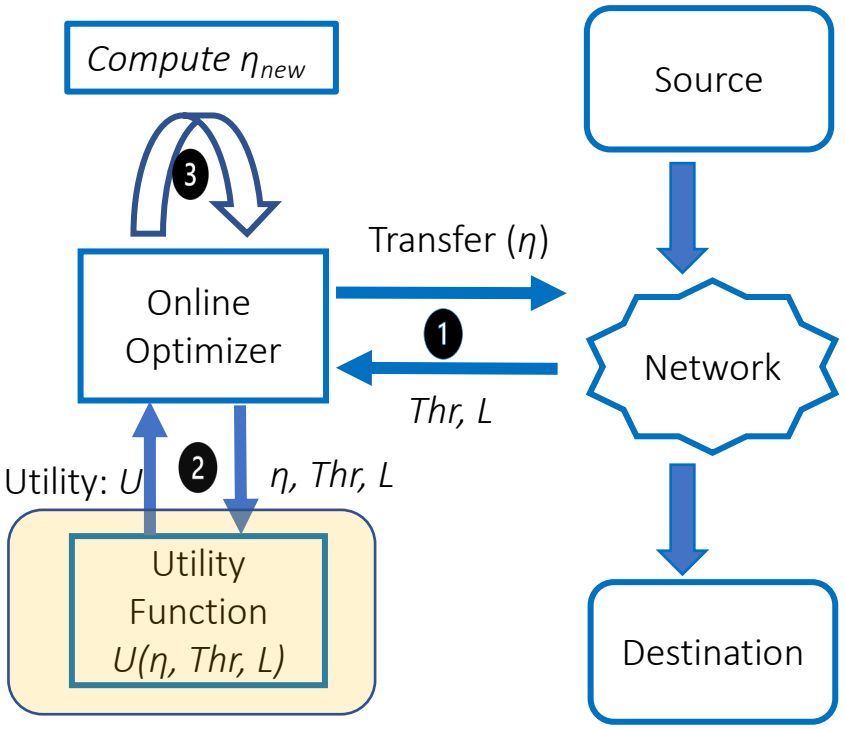


Utility Function – Version 1

- Throughput-oriented model; i.e., no penalty term

$$u(n,t) = \sum_{i=1}^{n} t = n * t$$

n = *concurrency*
t = *throughput per concurrent transfer*



- No-regret learning, aims to maximize throughput
- Overburdens both networks and storage systems by choosing solutions with high system overhead
- Potentially unfair resource allocation between transfers

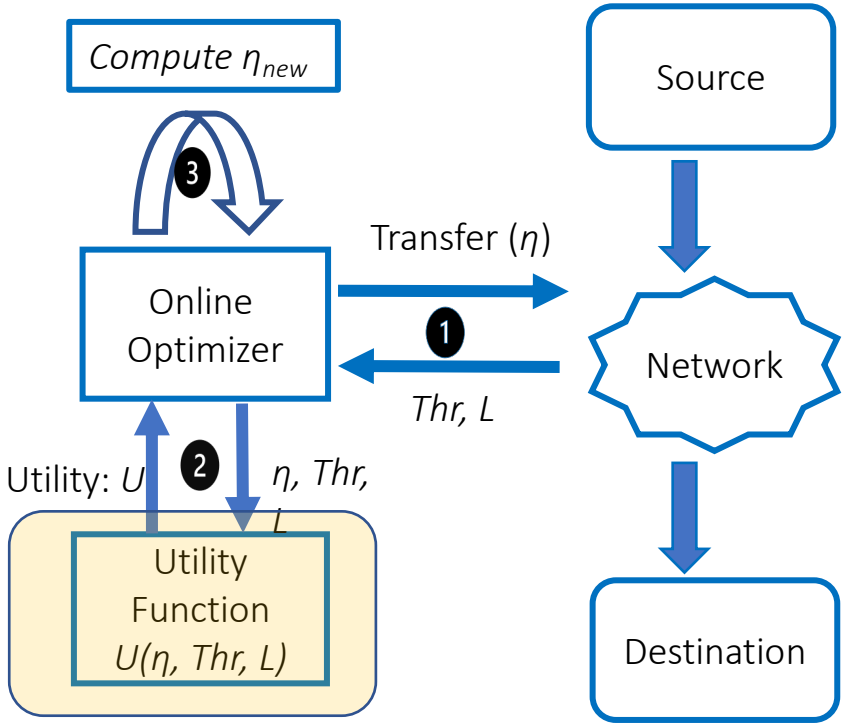
Utility Function - Version 2

- Introduces a penalty term for increased packet loss rate

$$u(n, t, L) = n * t - n * L * B$$

L = *Packet loss rate*
B = *constant coefficient for packet loss*

- Stops increasing the concurrency when packet loss rate increases more than throughput
- Fair resource sharing among competing transfer agents
  - Nash Equilibrium
- Does not work if a transfers are sender limited; i.e., no packet loss



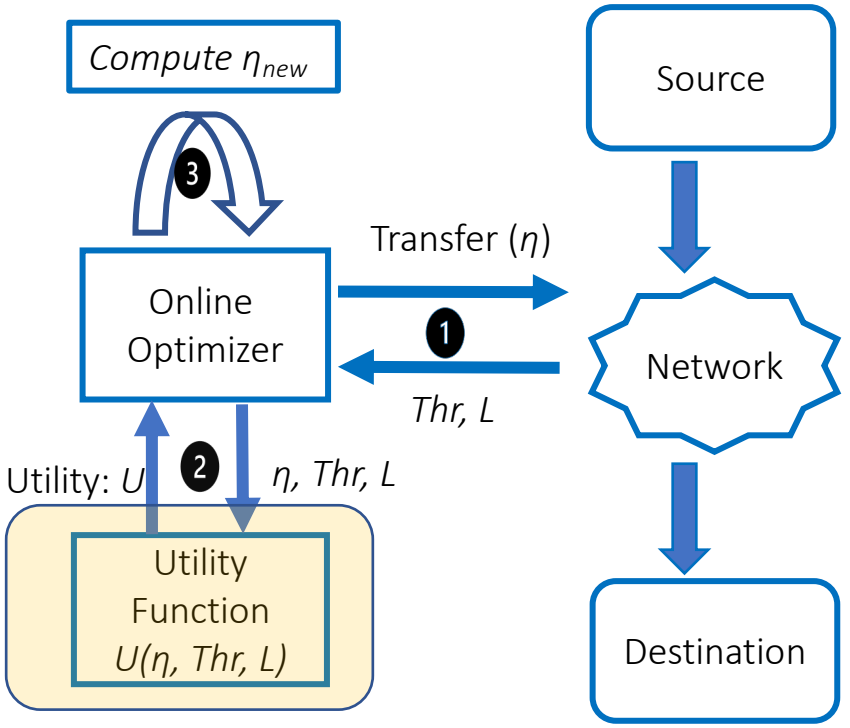
Utility Function – Version 3

- Adds a penalty term for increased concurrency value

$$u(n,t,L) = n * t - n * L * B - n * C$$

L = *Packet loss rate*
B = *constant coefficient for packet loss*
C = *constant coefficient for concurrency*

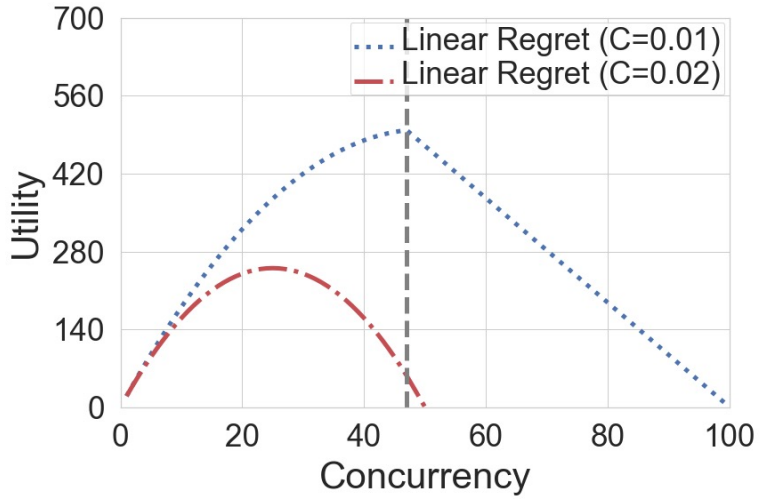
- Works in the absence of packet loss
- Throughput gain must offset concurrency punishments to increase concurrency
- Linear form of regret can lead to convergence to suboptimal solution

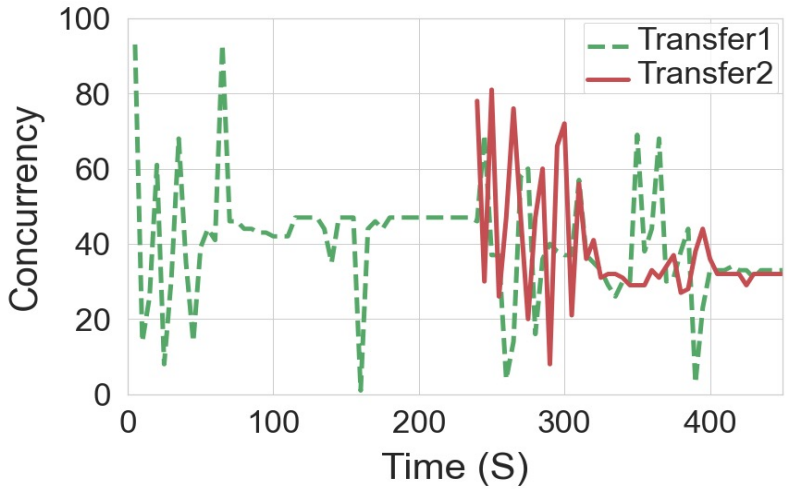


Limitations of Linear Regret

Utility falls sharply with increasing concurrency; leads to suboptimal solution unless punishments set to very low

Low punishment is sensitive to measurement fluctuations; leads to longer search phase and higher than required concurrency





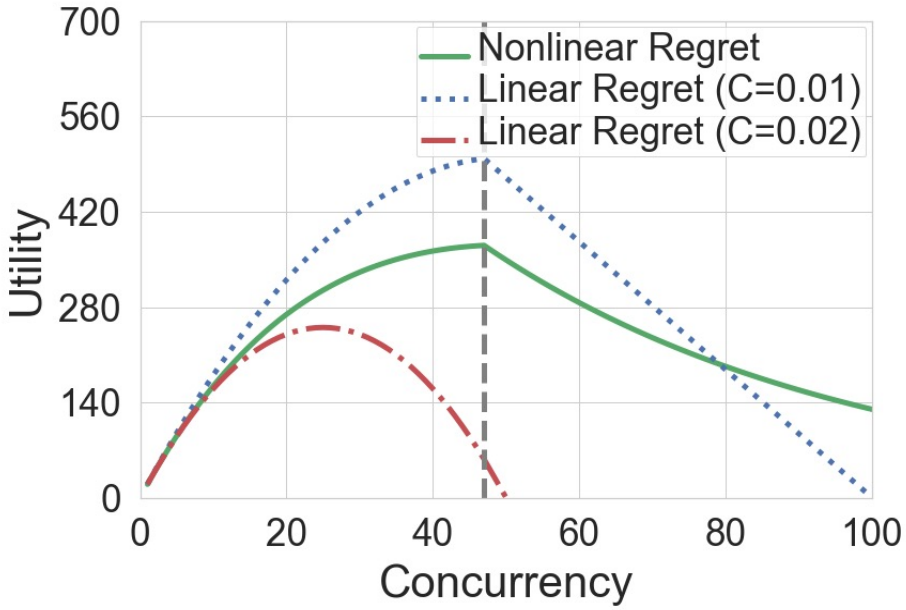
Grey vertical line represent target concurrency

Utility Function – Version 4

- Introduces a penalty term for concurrency in non-linear form

$$u(n,t,L) = \frac{n * t}{C^n} - n * L * B$$

Default: C = 0.02, B = 10



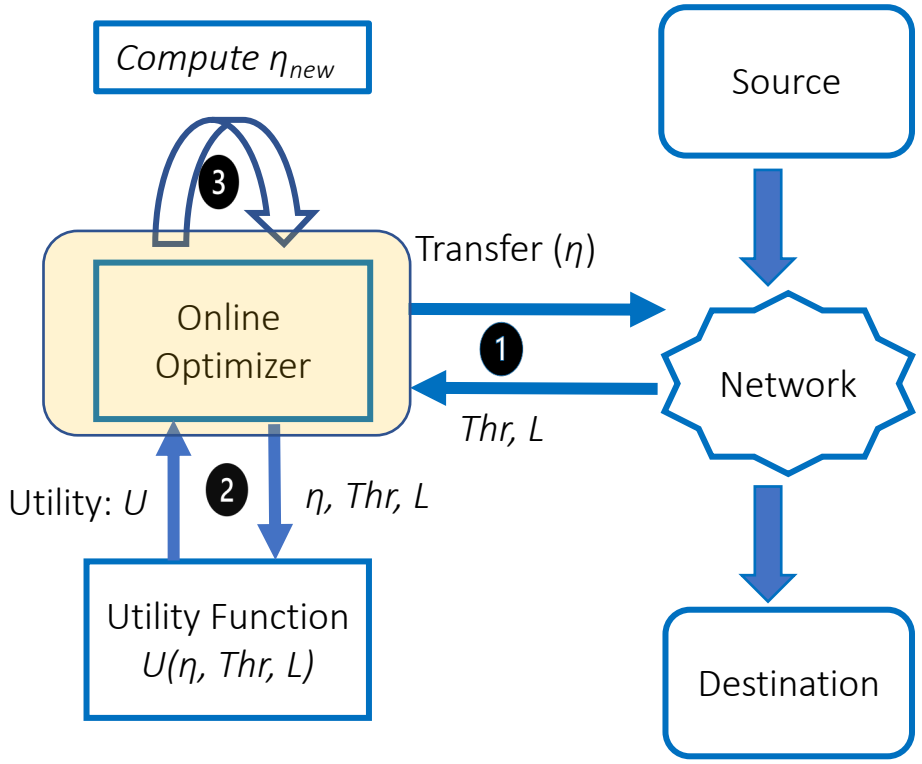
Grey vertical line represent target concurrency

- What's next?
  - How to scan search space efficiently?

Online Search

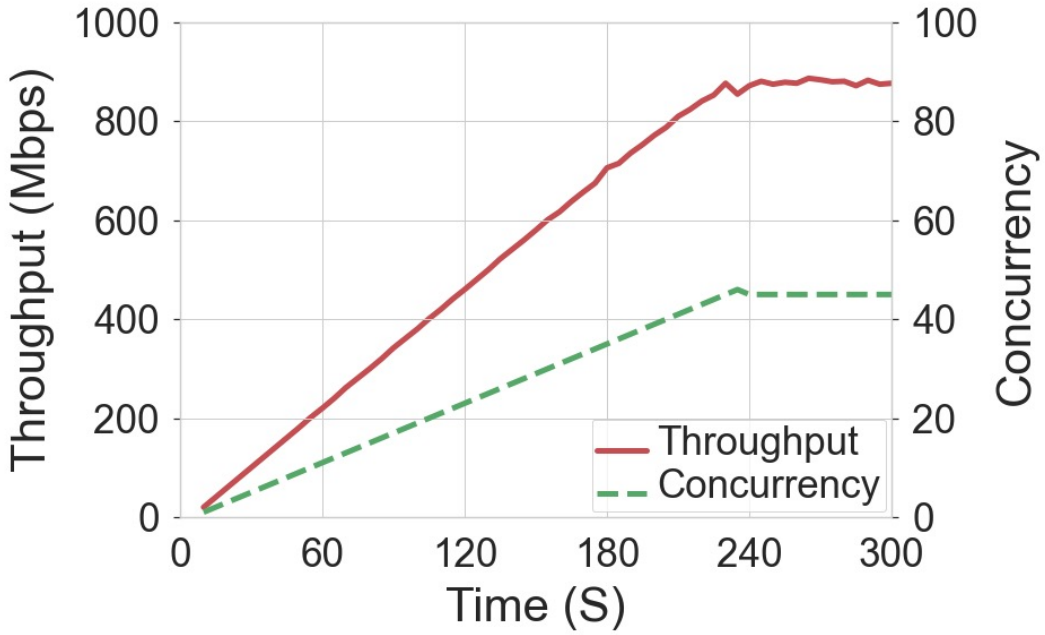
Observations come in sequentially, so sequential optimization algorithms are more suitable!

Expectation: Quick convergence to the optimal solution to maximize the gain for overall transfer



Hill-Climbing Algorithm

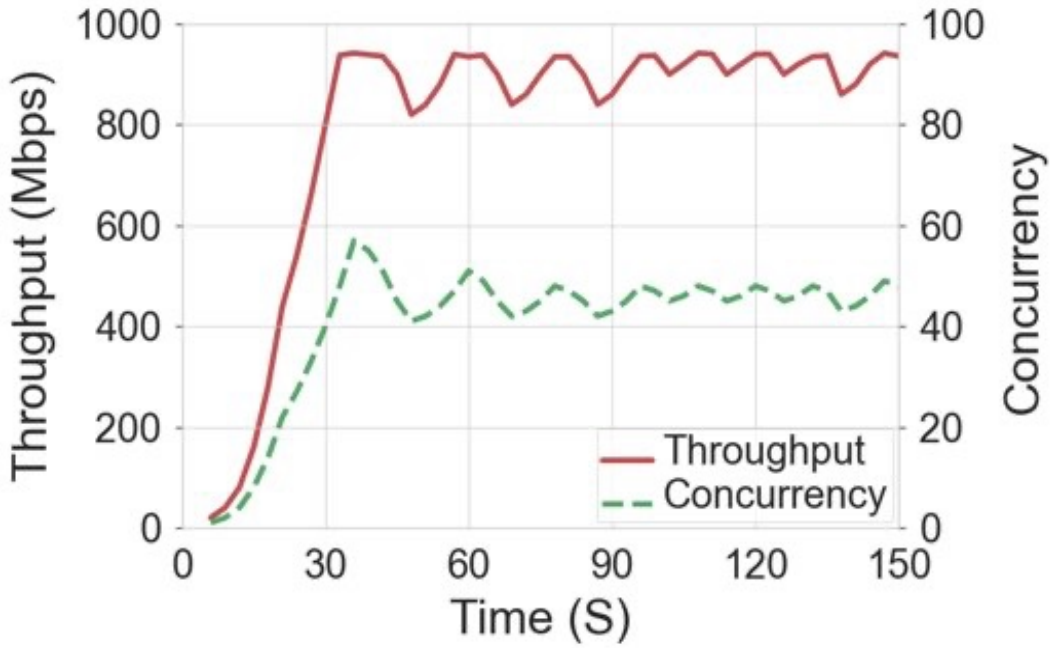
- Incrementally goes up, down based on utility
- Slow convergence to optimal solution



Emulab: 20Mbps/thread I/O limits
Target Concurrency: 47

Gradient Descent

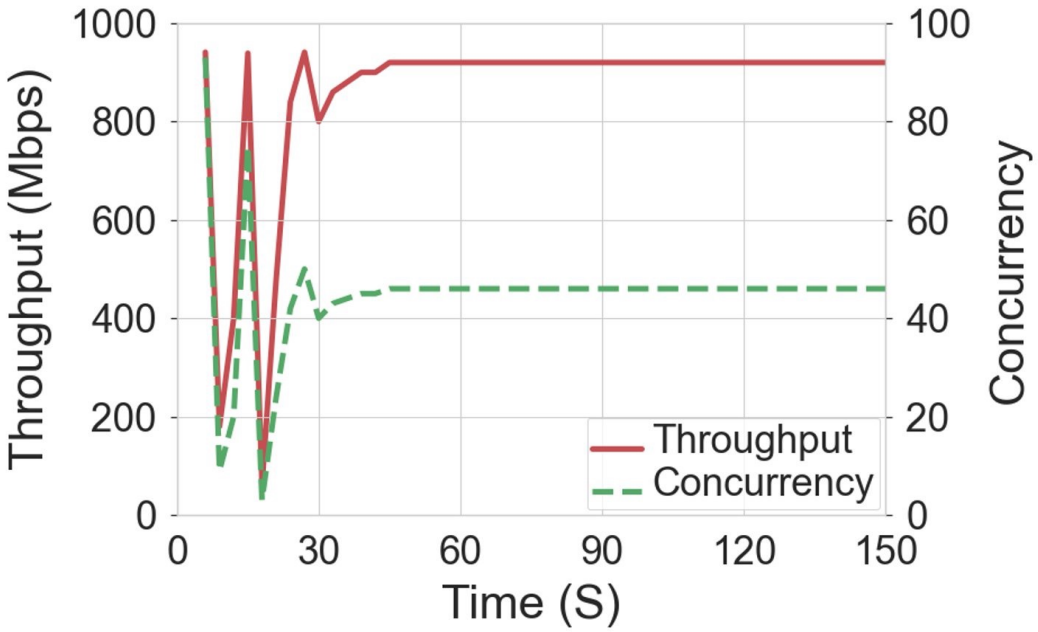
- Calculate gradient from last two observations
- Use gradient for search direction
- Quickly narrows down search space
- Never settles to any fixed solutions
- Keep searching around the minimum cost regions



Emulab: 20Mbps/thread I/O limits
Target Concurrency: 47

Bayesian Optimization

- Sequential Model Based Optimization (SMBO)
- Build a probabilistic model, Gaussian Process (*GP*) based on observations
- Predict, evaluate and update *GP* model at each iteration
- Repeat until the ends of the transfer

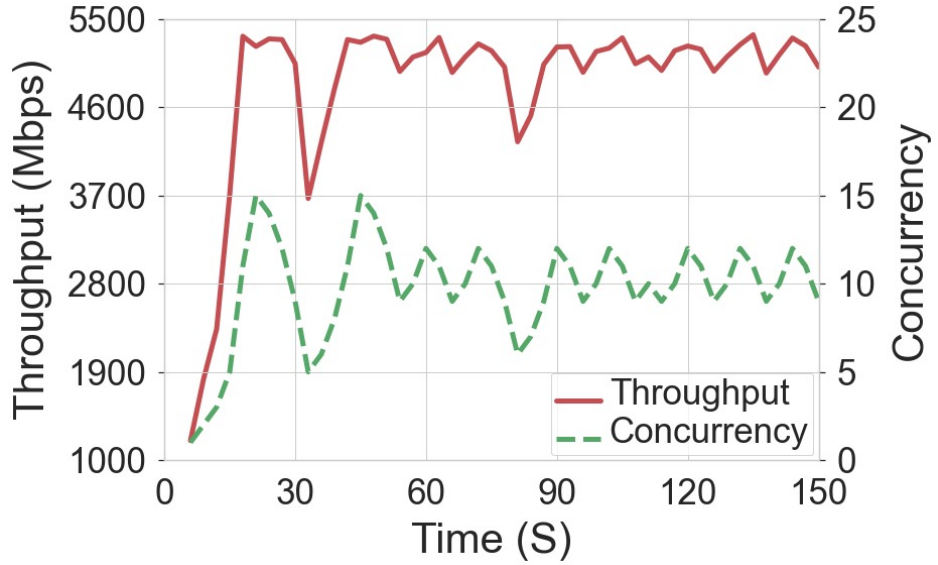


Emulab: 20Mbps/thread I/O limits
Target Concurrency: 47

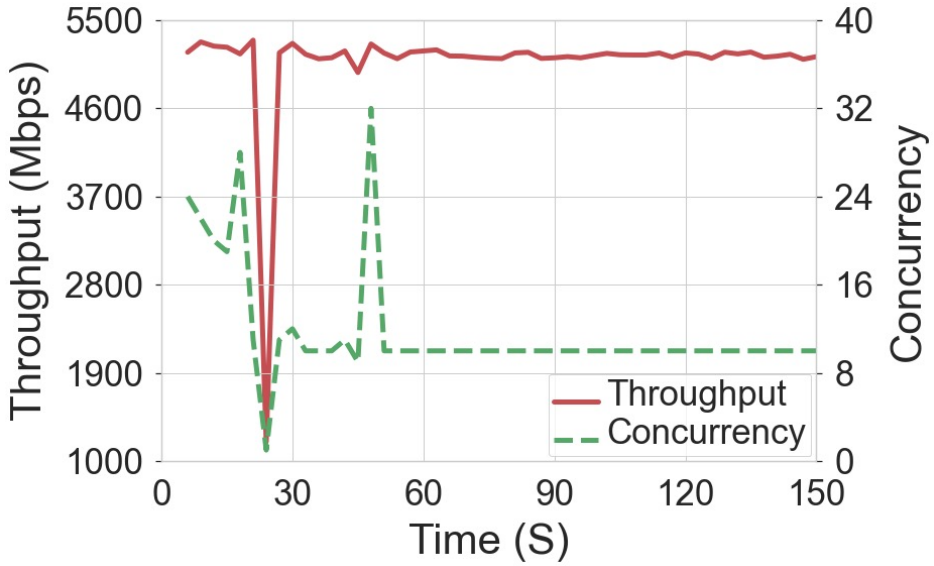
Evaluation

High Performance

Falcon agents achieve high throughput in OSG to Expanse transfer while keeping concurrency values at minimum (Link *Bandwidth is 10GB; Sender I/O limited*)



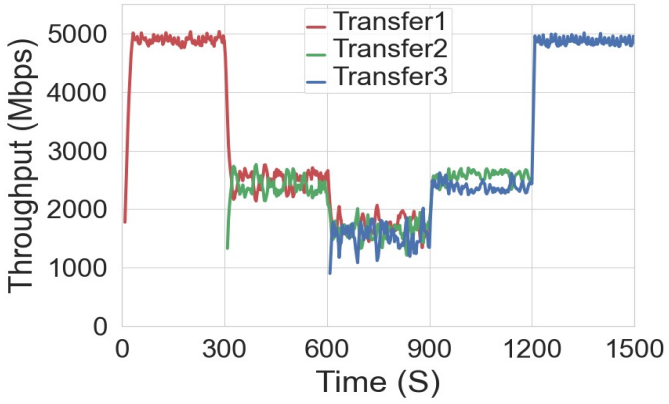
Gradient Descent



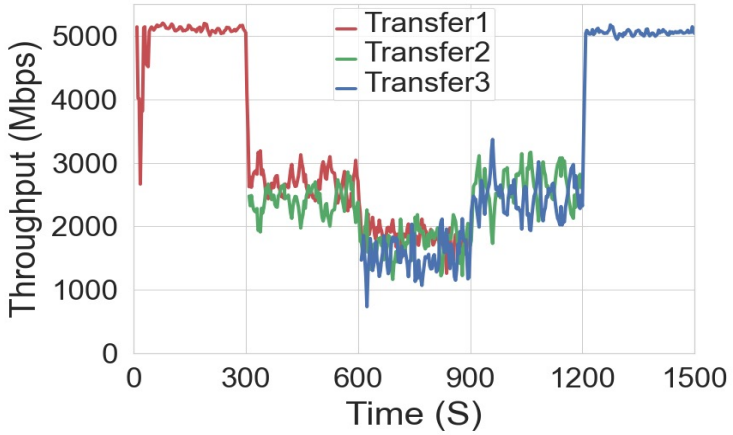
Bayesian Optimization

Fair Resource Sharing

- Falcon agents fairly share bandwidth
- When one of the competing agents leave, the others grab the available bandwidth



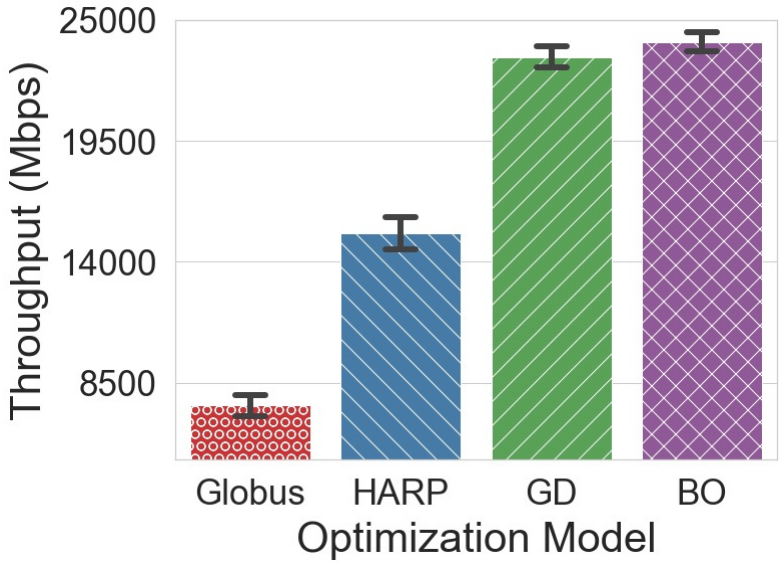
Gradient Descent



Bayesian Optimization

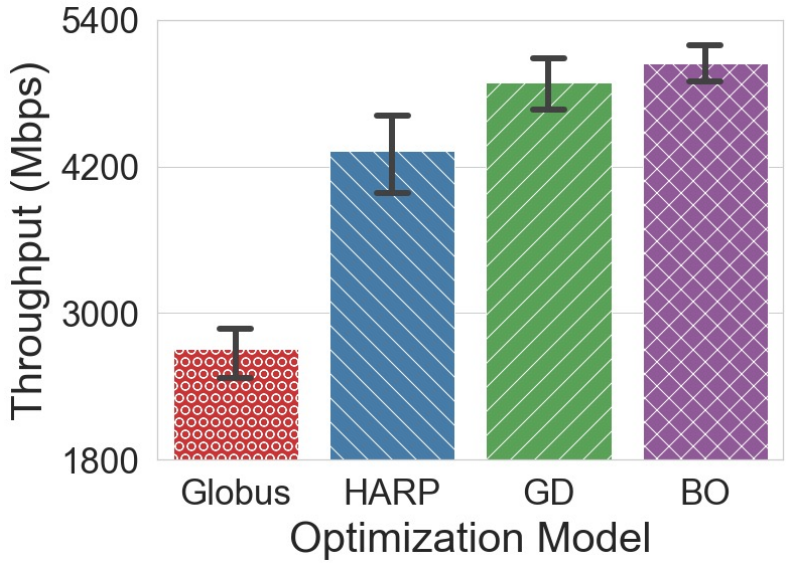
Comparison to State-of-the-Art

HPCLab



- 220% higher throughput than Globus
- 55% higher throughput than HARP

OSG to Expanse



- 85% higher throughput than Globus
- 20% higher throughput than HARP

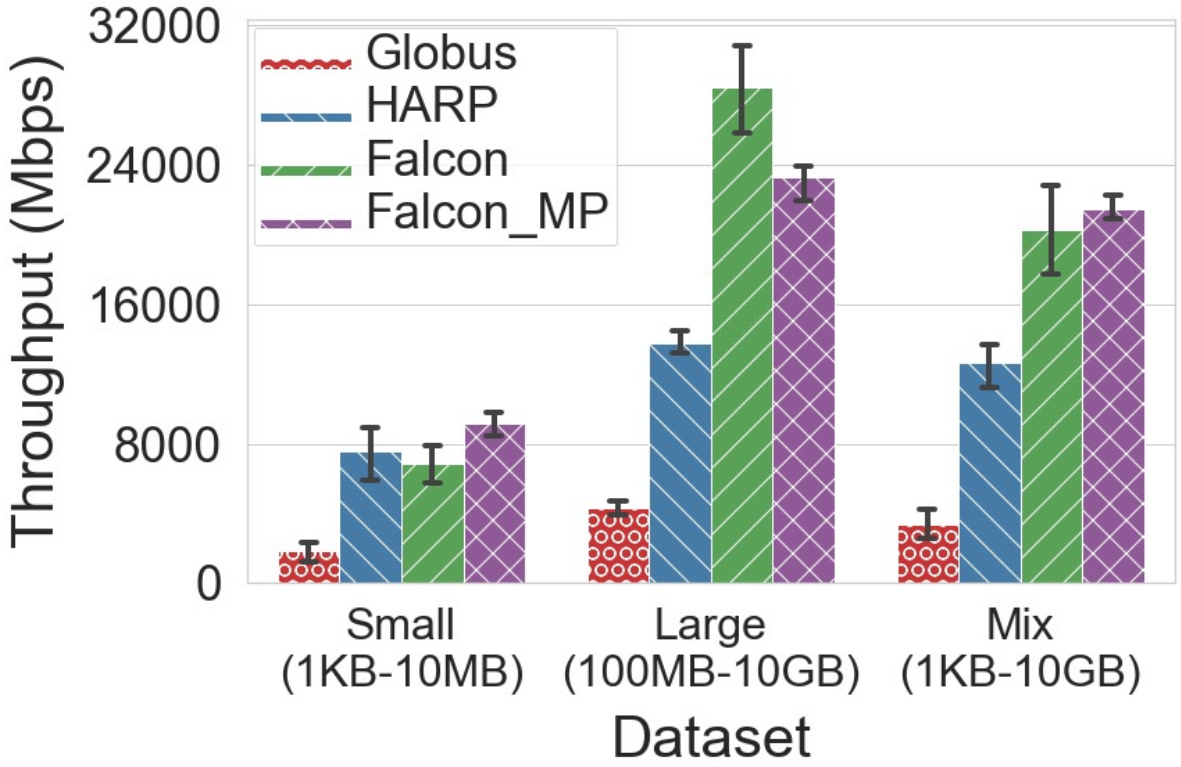
*GD – Gradient Descent; BO – Bayesian Optimization

Multi-parameter Optimization

- Searching for optimal of concurrency (n), parallelism (p), and command pipelining (cp)

$$u(n,\ p,\ t,\ L) = \frac{n * p * t}{C^{n*p}} - n * L * B$$

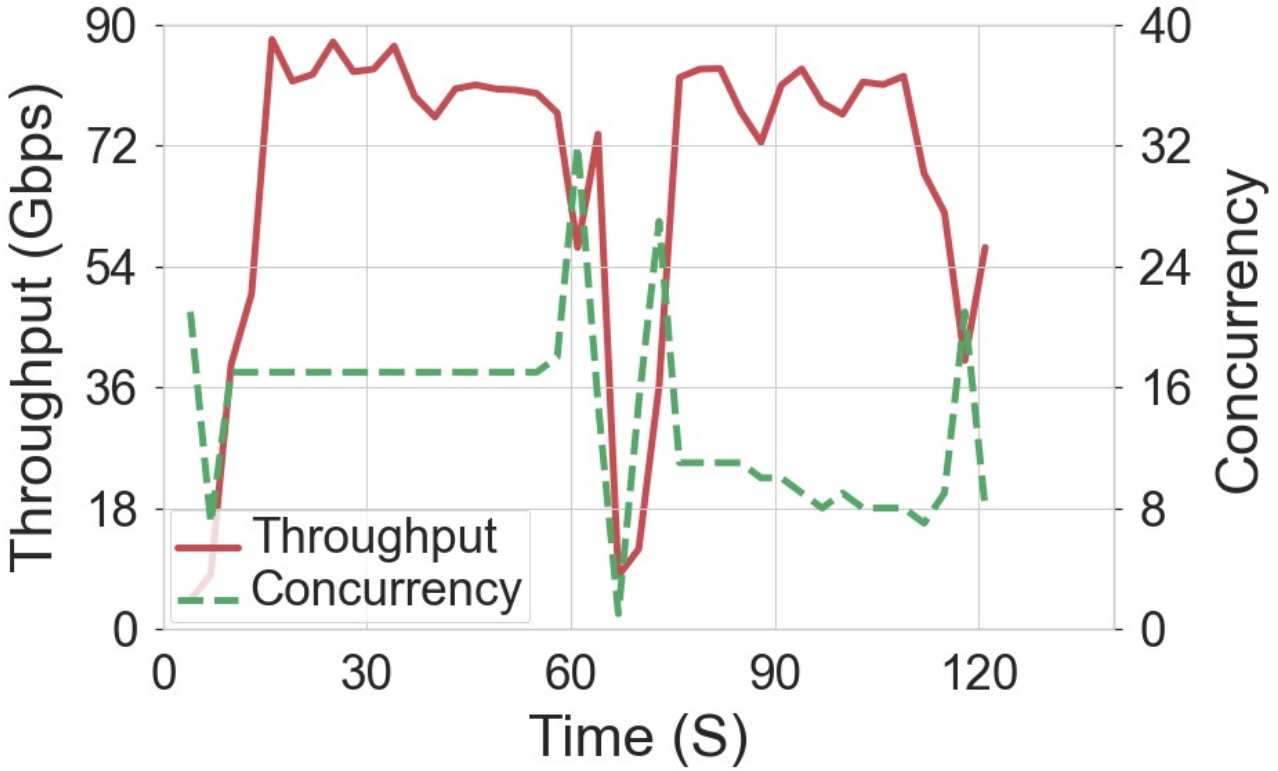
- Additional nonlinear cost added for parallelism
- No penalty added for using command pipelining since it does not incur overhead



Pipelining bundle acknowledgments for small files to mitigate TCP slow-start problem, while parallelism speeds up transferring large files by splitting them

High Bandwidth-Delay Product

- Between two ESnet testbeds
  - *100 Gbps bandwidth, 88ms RTT*
- 1TB data transfer
  - Average finish time is 129 seconds using Falcon
  - On average 117 seconds with fixed concurrency value 8



Summary

While transfer parallelism is key to improve performance, it is challenging to find the optimal that obtains high performance and incurs low overhead

Falcon combines game-theory inspired utility function with state-of the-art online optimization algorithms to discover optimal parallelism

Falcon outperforms the existing file transfer algorithms by 1.2 to 5x in both local and production cluster

Thanks!