NetStitcher

Delay Tolerant Bulk Data Transfers on the Internet

Xinghong Fang

March 6, 2012

Introduction

- Large datacenter deployed at multiple location across globe
  - Latency affects revenue (mirroring)
  - Need for fault tolerance (backup)
- Large amount of data transferred between datacenters
  - Usage follows strong diurnal patterns
- With flat or 95 percentile pricing
  - Low utilisation for paid off-peak hours

Outlines

Outlines

The Problem

- Dimensioning of site determined by peak usage
  - Peak usage follows diurnal pattern
  - Poor utilisation of site's resource during off-peak hours
  - Constant upgrading
- Inter-datacenter bulk data transfer
  - Wide-area transit bandwidth cost
  - Backup, bulky updates, data migration applications
  - Leftover bandwidth appears at limited time-window (3am-6am)
  - Time difference between datacenters (no overlapping time-window)

Outlines

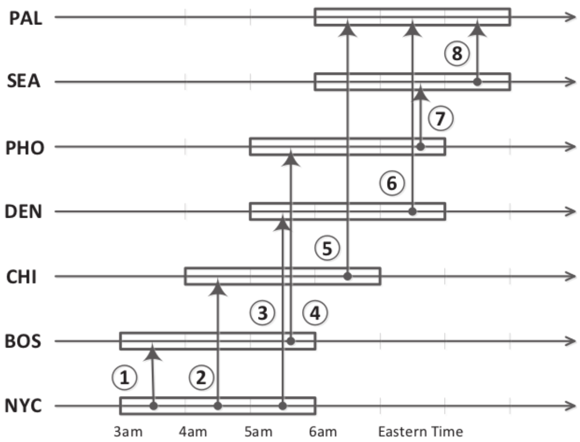
1. The Problem
2. Approach
   - NetStitcher
   - Multipath and Multi-hop Forwarding
   - Fault Tolerance
3. Scheduling
4. Implementation
5. Evaluation
6. Future Works

NetStitcher

- Postpone bulk transfer to non-peak time
- Store-and-forward (SnF) algorithm
  - Store data at intermediate nodes and forward later
  - Network cost $->$ storage cost (much lower)
- Schedule transfers by predicting availability of leftover bandwidth
- Adapt to estimation errors and failure
- Outperforms previous methods
  - End-to-end
  - Multipath overlay routing
  - Simple SnF
  - BitTorrent (greedy SnF)

Multi-hop and Multipath Forwarding

- **Multi-hop**: Enable end-to-end links overtime
- **Multipath**: Improve efficiency and bypass problematic links



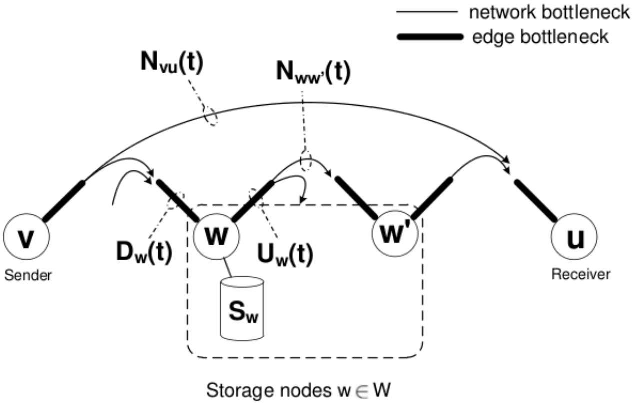
Fault Tolerance

- Estimation error or failure can occur
- NetStitcher revises the transfer schedule periodically
- Optimal flow algorithm
- End-Game Mode (EGM)

Outlines

Scheduling with Perfect Knowledge

- Storage and network constraints are known at any given time T
- Minimum Transfer Time (MTT) problem $M(F, v, u, W)$
- Find the minimum time T_{max} to transfer the file F



Scheduling with Imperfect Knowledge

- Periodically recompute transmission schedule
- Need to consider parts already delivered to intermediate nodes
- Multi-source max-flow problem
- Super nodes S

Cast multi-source to single-source max-flow problem

- Demand at sender: F_v (unsent part)
- Demand at intermediate nodes: F_w (volume of data store)
- Total flows: $F_v + \sum_{w \in W} F_w$

+'s and -'s of the scheduling

- +'s
  - Stateless
  - Simplifies the admission of multiple jobs
- -'s
  - Fallback to multipath overlay routing when only short-term prediction is available

Outlines

Implementation

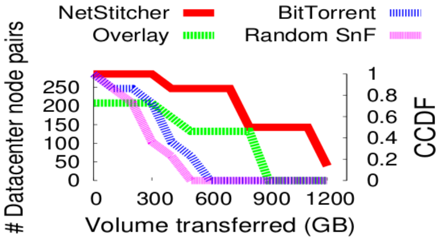
- Overlay management
  - Run at broker process
  - Add/remove node to/from overlay
- Volume prediction
  - Run at broker process and peer process
  - Predict maximum volume of data to be forwarded
- Scheduling
  - Run at the sender's scheduler process
  - Calculate initial transmission schedule and update it periodically
- Transmission management
  - Run at peer process
  - Throttling transmission according to schedule

Outlines

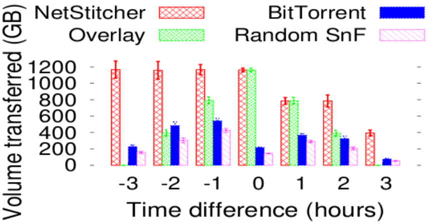
North American Transfers - Experiment Setting

- One datacenter in each location (different time-zone)
- 1Gbps up/down link
- 3am-6am time-window
- Backbone constraint not considered
- Attempt to send a 1223GB file between each pair

North American Transfer - Transferred Volume

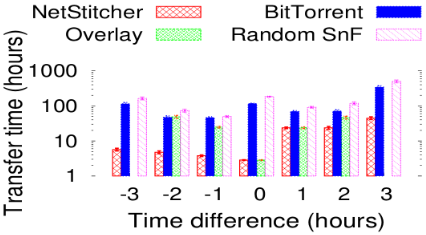


(a)

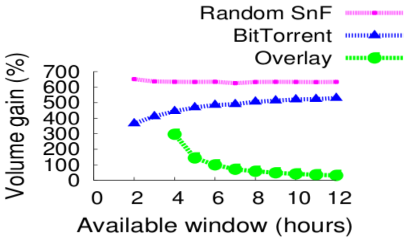


(b)

North American Transfer - Transfer Time



(a)



(b)

International Transfer

Conditions	Volume Transfered	Transfer Time
No backbone constraints	2720GB	16h
Backbone constraints and constraint-aware scheduler	1553GB	36.4h
Backbone constraints and constraint-unaware scheduler	799GB	∞
Backbone constraints, constraint-aware scheduler and *NetStitcher* node in Madrid	2720GB	16h

Table 1: Volumes transferred during 24h and the total time it takes to transfer 2720GB between the Frankfurt and Palo Alto Equinix datacenters. We repeat the experiment 5 times and report mean values. We consider as the start time of a transfer the moment the window of Frankfurt becomes available.

Live CDN Deployment

- Satisfactory approximation ($error < 0.18$)
- Lower cost
  - 86% lower cost than overlay
  - 90% lower cost than BitTorrent
  - Storage cost ($33) much lower than median bandwidth cost ($438)
- Greatest saving with negative time difference

Outlines

1. The Problem
2. Approach
3. Scheduling
4. Implementation
5. Evaluation
6. **Future Works**
7. Conclusion

Future Works

- Bootstrapping deployment
- Intermediate nodes security
- Dealing with higher estimation error or failure rate
- Optimisation on positive time difference
- Optimise multiple transfers

Conclusion

- $+$'s
  - Utilise non-overlapping leftover bandwidth
  - Use bandwidth prediction to optimise transmission schedule
  - Adapt to estimation errors and failures
  - Stateless recomputation and multiple jobs admission support
  - No changes needed to the network devices
  - Larger transferred volume and less transfer time
  - Cheaper bulk transfer in CDN
- -'s
  - Short-term prediction causes NetStitcher to fallback to multipath overlay routing
  - Efficiency on positive time difference

Thank you!