# HyLog: A High Performance Approach to Managing Disk Layout

Wenguang Wang  Yanping Zhao  Rick Bunt

Department of Computer Science
University of Saskatchewan
Saskatoon, Canada

# Background

- The write performance of a storage system is impacted by
  - the disk characteristics
    - Disk positioning time
    - Transfer bandwidth
  - the strategy for writing
    - Overwrite
    - LFS (Log-structured File System)

# Overwrite

- **Idea**: new data are overwritten on top of old data
- **Problems**: lots of time lost in disk arm positioning in workloads with small writes scattered over the disk

# LFS

- **Idea**: new data are accumulated and written to new disk locations in large sequential transfers
- **Assumptions of the disk characteristics**: large sequential transfers are more efficient than small block transfers

# LFS (cont.)

- **Advantages**
  - good write performance
  - no small write penalty on RAID-5
  - fast recovery
- **Problems**: segment cleaning is expensive
  - For a year 1991 disk, TPC-B workload, and 50% disk space utilization, cleaning overhead reduces overall system throughput by 33% (Seltzer et al. USENIX'95)

# Motivation

- **Observation:** disk sequential transfer bandwidth has improved 10x more than positioning time

| | DEC RZ26 (year 1991) | Cheetah X15 36LP (year 2003) | Diff. |
|---|---|---|---|
| Positioning time | 15ms | 5.6ms | 2.7x |
| Transfer B/w | 2.3MB/s | 61MB/s | 27x |

- **Question:** how are Overwrite and LFS affected by this trend?

# Objective

- Revisit the performance of LFS under modern and future disks
- Evaluate the performance of LFS under disk arrays and concurrent users
- Attempt to perform better than LFS and Overwrite

# Outline

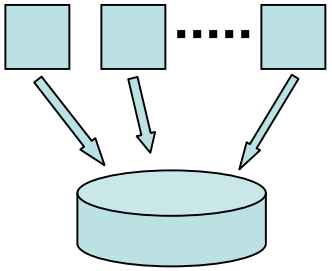
- Background, Motivation, and Objective
- The analysis of LFS and Overwrite
- The design of HyLog
- Experimental methodology and results
- Conclusions and future work

# Experimental Parameters

- Three SCSI disks
  - DEC RZ26 (year 1991)
  - Quantum atlas10k (year 1999)
  - Cheetah X15 36LP (year 2003)
- Page size: 8KB
- Workload: uniformly distributed random update (TPC-B)

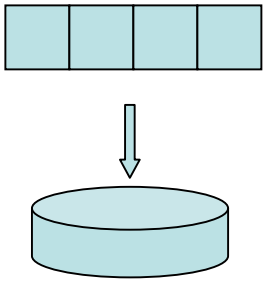
# Modeling Write Performance

- In Overwrite:



Time to write N pages is $T_1$

- In LFS:



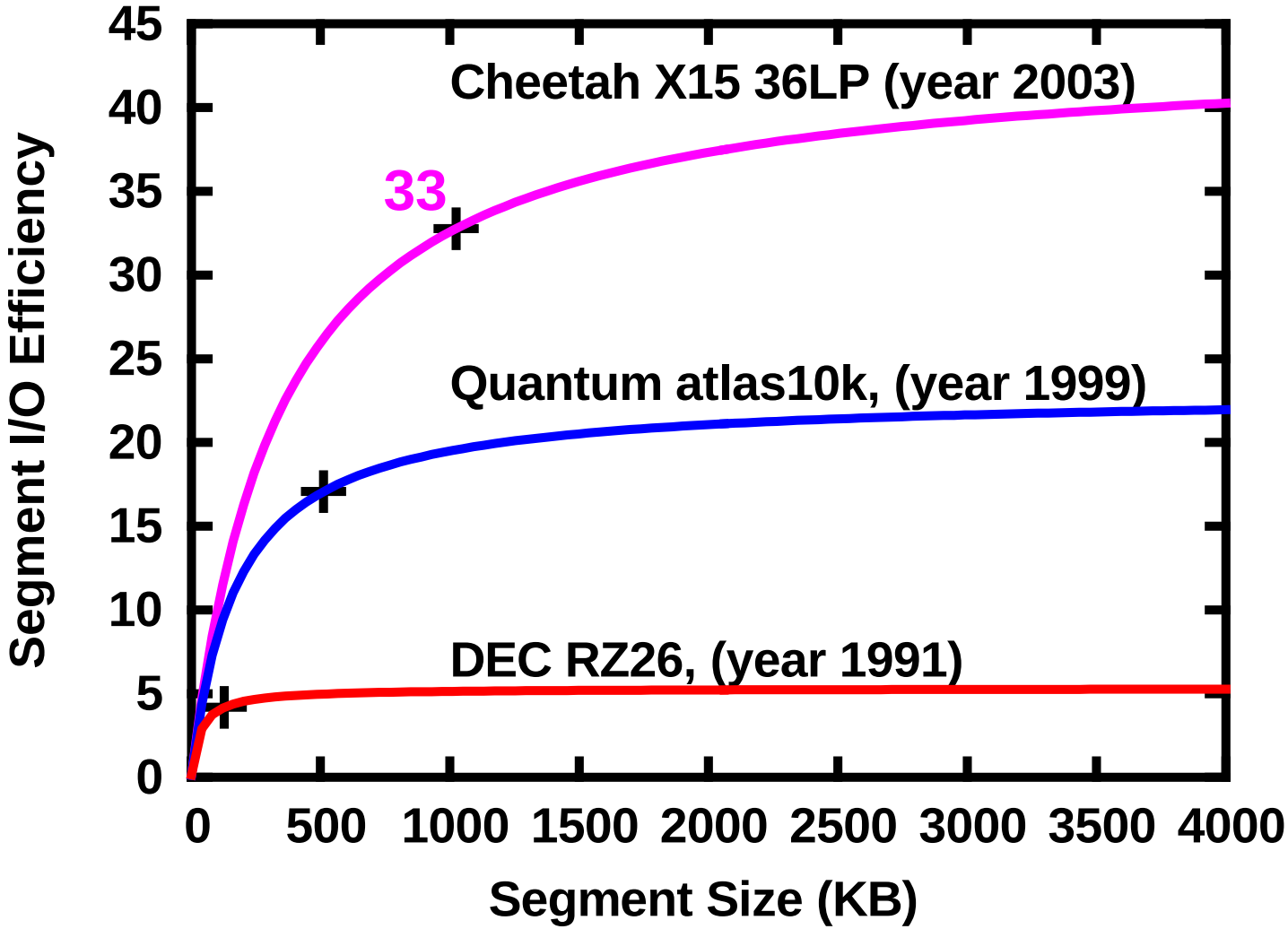
Time to write a segment containing N pages is $T_2$

- $T_1 > T_2$
- Segment I/O Efficiency = $T_1 / T_2$

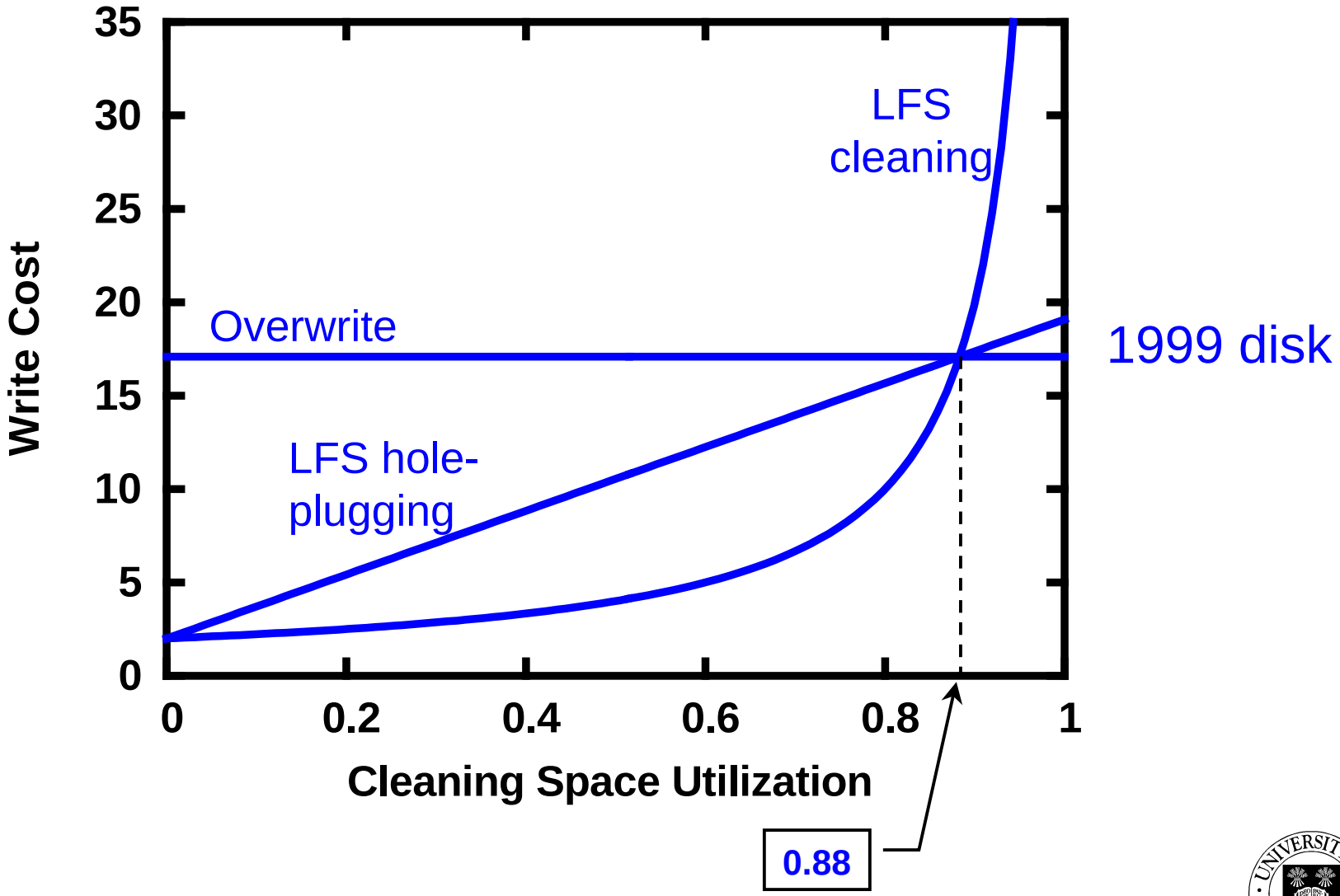
# A Simple Scenario

- Assume the segments to be cleaned are always 80% utilized (cleaning space utilization = 80%)
- LFS requires 5 seg. reads and 4 seg. writes to reclaim a free segment
- LFS requires 10 seg. I/Os (9 seg. for cleaning, 1 seg. for new data) to write a segment
- If Segment I/O Efficiency > 10, LFS is still faster than Overwrite!

# Segment I/O Efficiency



# Overwrite vs. LFS



# Overwrite vs. LFS

- The crossing point where LFS has the same performance as Overwrite

| Year of Disk | Cleaning Space Utilization | Disk Space Utilization |
|---|---|---|
| 1991 | 0.52 | 0.74 |
| 1999 | 0.88 | 0.94 |
| 2003 | 0.94 | 0.97 |

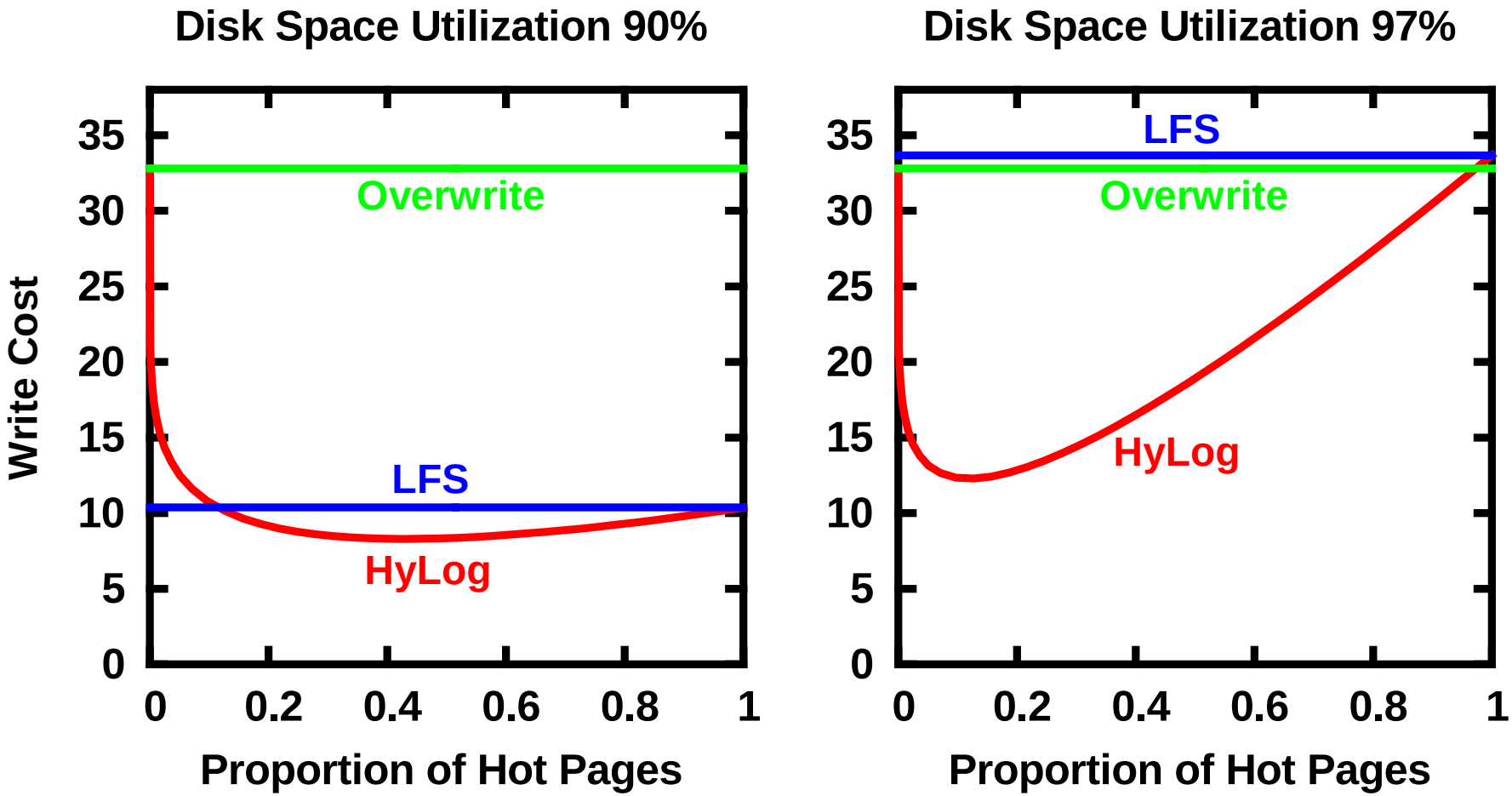
# Disk Access Characteristics

- In most workloads, most writes are to a small number of pages (hot pages)
- Impact of skewness on LFS performance
  - Most of the cleaning cost comes from cold pages
  - Most of the good write performance comes from accumulating the writes to hot pages

# HyLog (Hybrid Log-structured Approach)

- Separates the disk into two partitions: hot partition and cold partition
- Uses log-structured approach to manage the hot partition
- Uses overwrite to manage the cold partition

# Performance Potential of HyLog




Disk: year 2003, workload: 80% references are in 20% pages

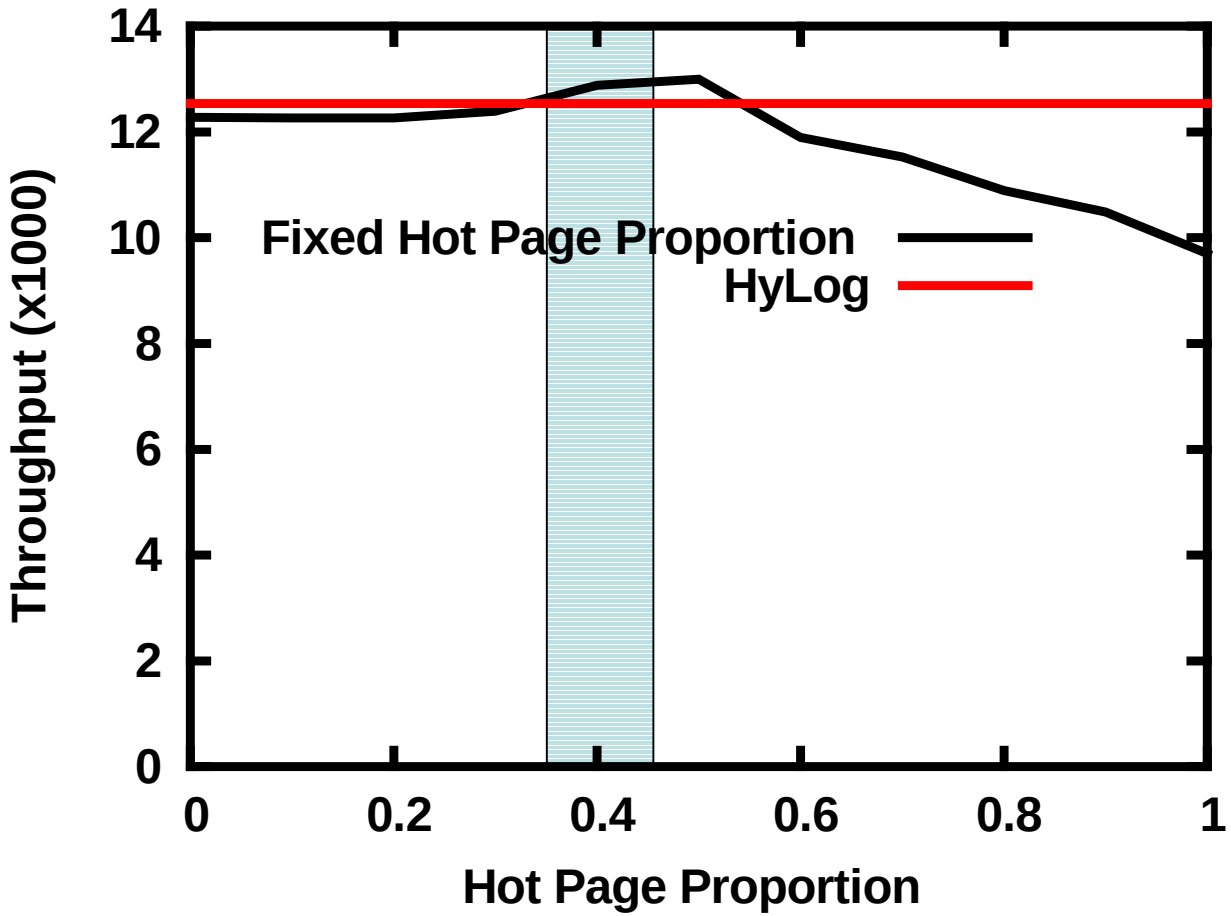
# Design of HyLog

- **Key design issue:** page separating algorithm
  - Collects page write frequencies
  - Finds the hot page proportion to minimize expected write cost
  - Determines the threshold of write frequency from the desired hot page proportion
  - Uses the threshold to distinguish hot pages from cold pages

# Evaluation Methodology

- Trace driven simulation
  - Overwrite, LFS, WOLF, and HyLog are implemented
  - TPC-C, Email, and OLTP traces
  - Year 1999, 2003, and 2008 disk models
  - No think time between requests
- Metrics
  - Throughput: # I/O requests finished per second

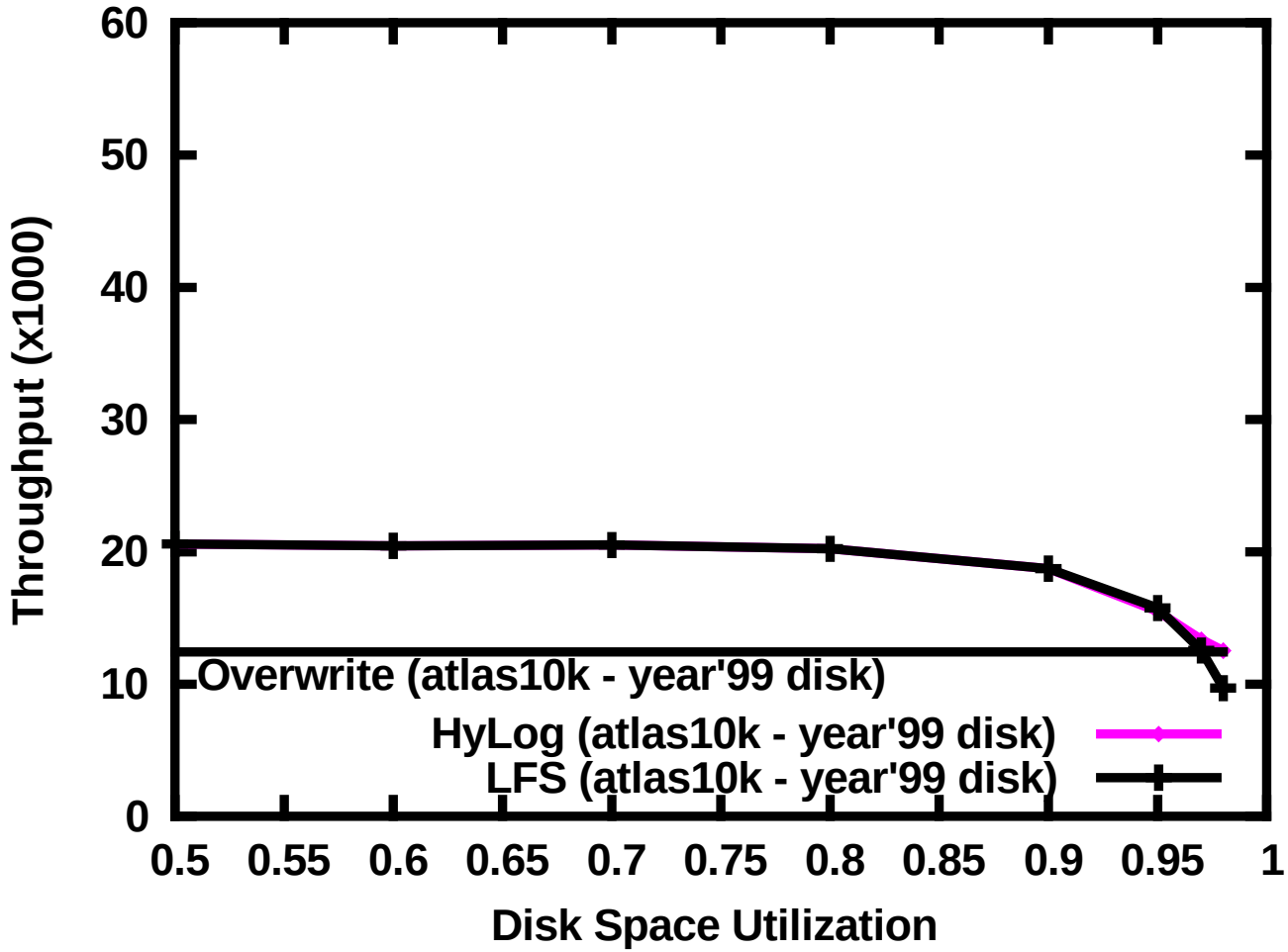
# Results – HyLog Page Separating Algorithm




HyLog adjusts the hot page proportion between 35-45%

TPC-C trace with 20 users and 4 disks, 98% disk space utilization

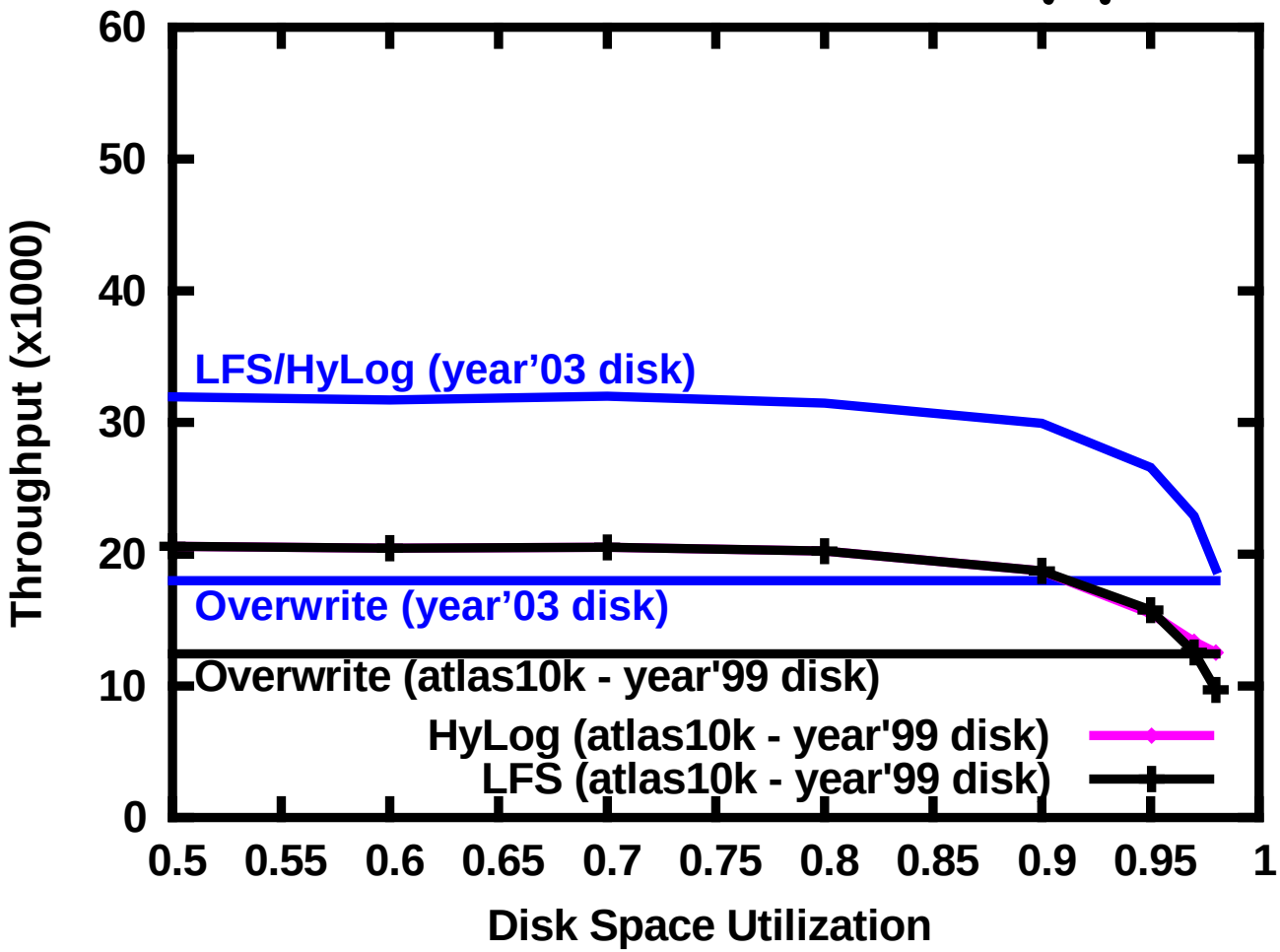
# Results – Disk Space Utilization




1999 disk

TPC-C trace with 20 users and 4 disks

# Results - Disk Type

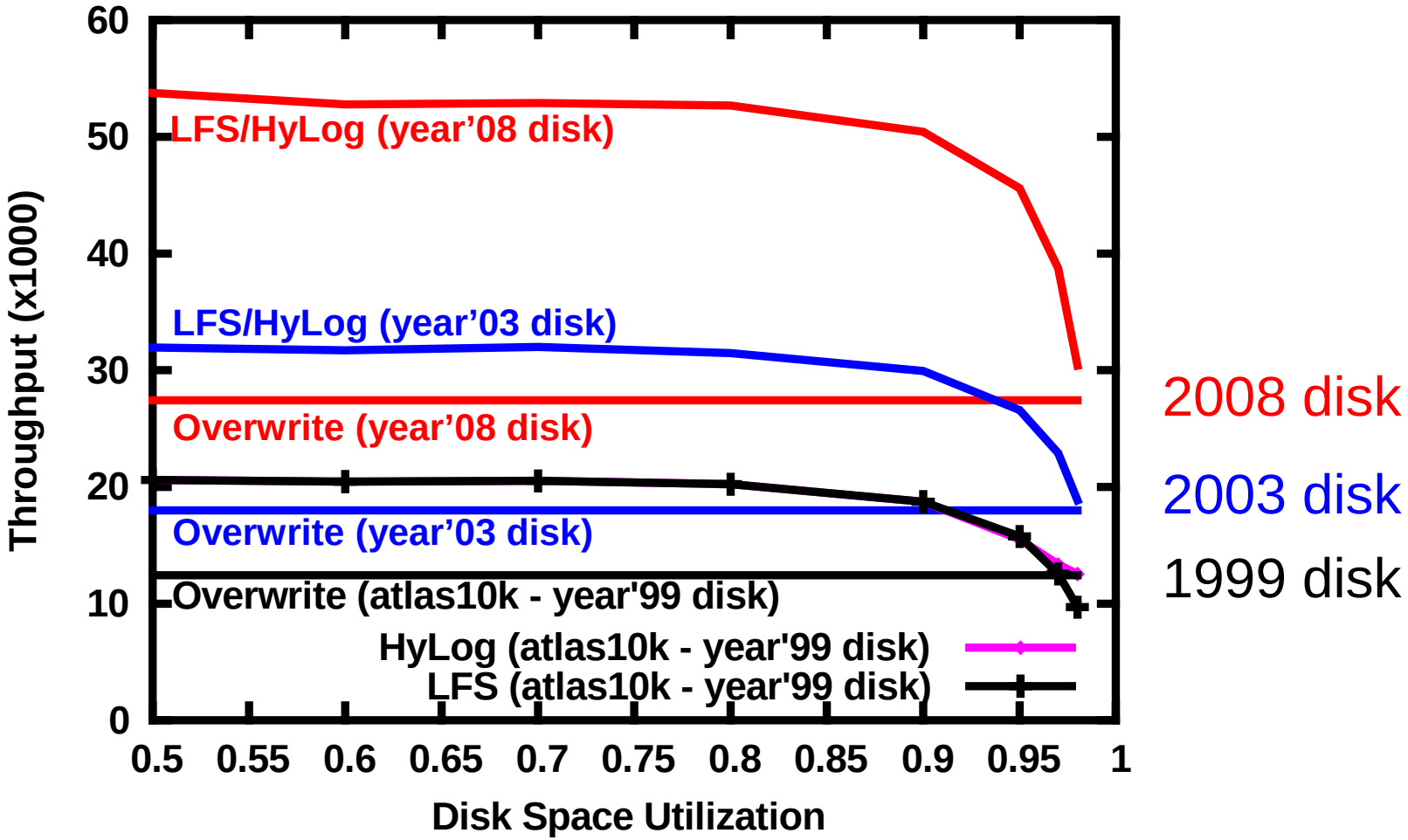



2003 disk

1999 disk

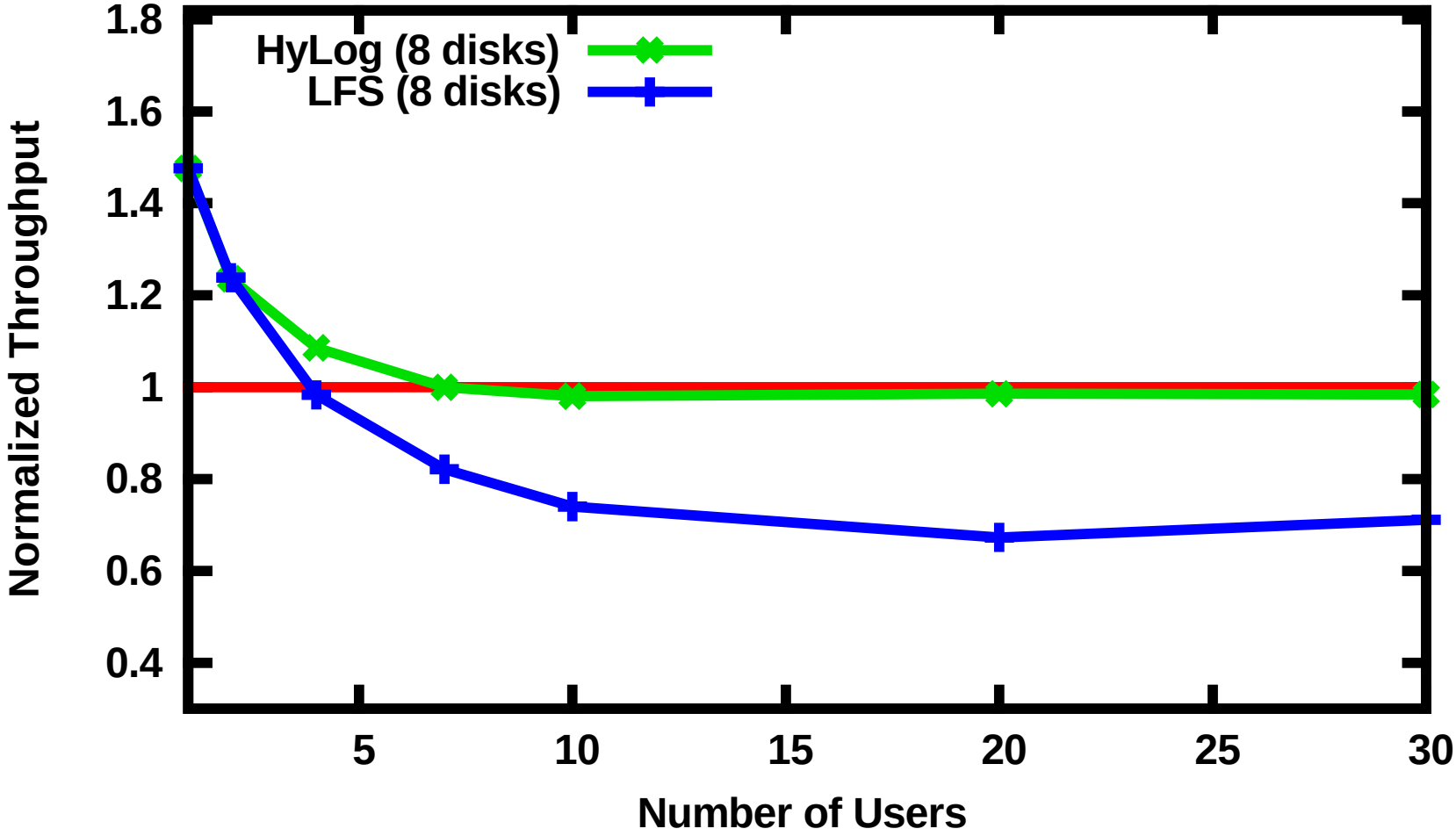
TPC-C trace with 20 users and 4 disks

# Results - Disk Type (cont.)



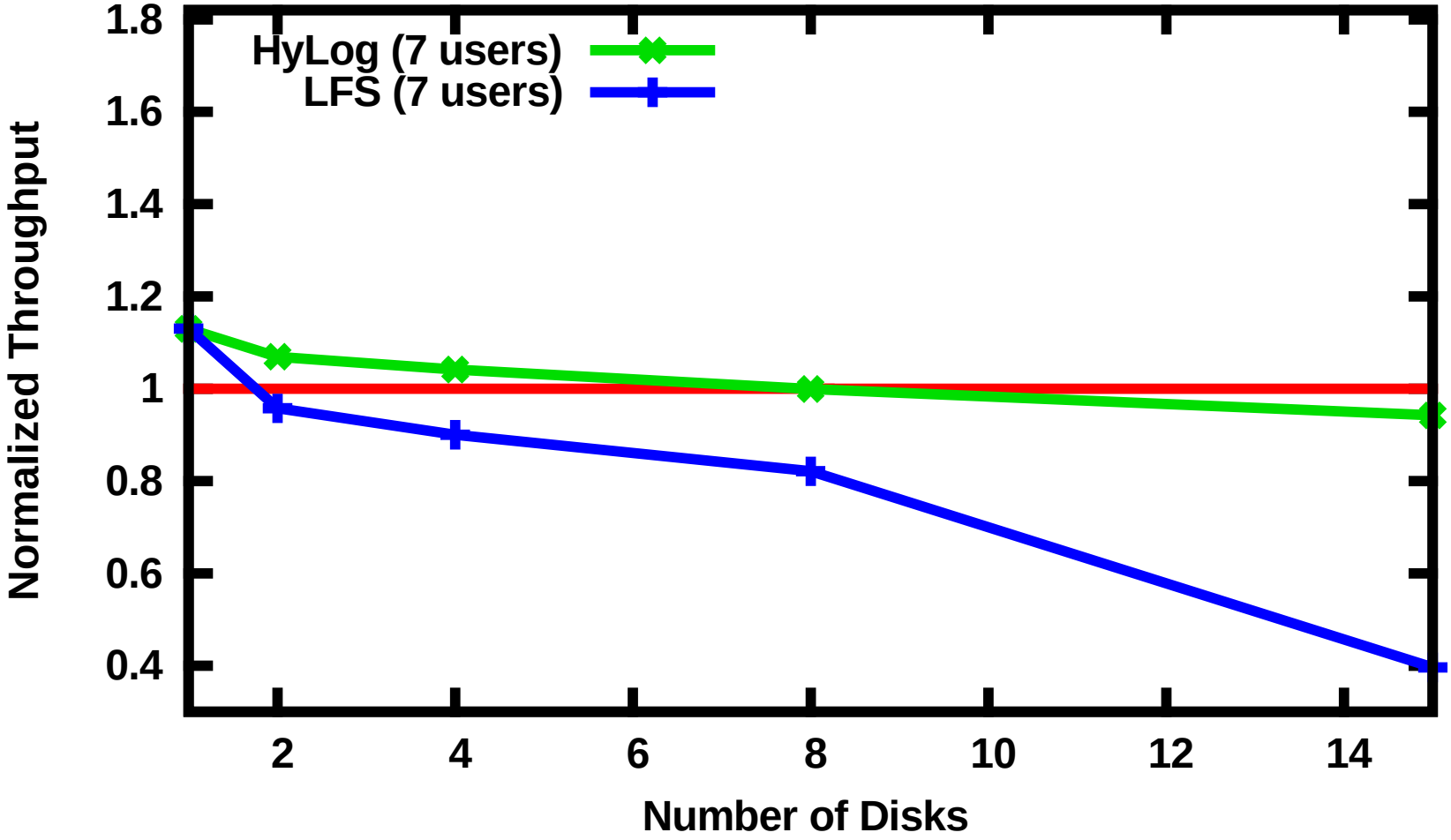

TPC-C trace with 20 users and 4 disks

# Results - Number of Users



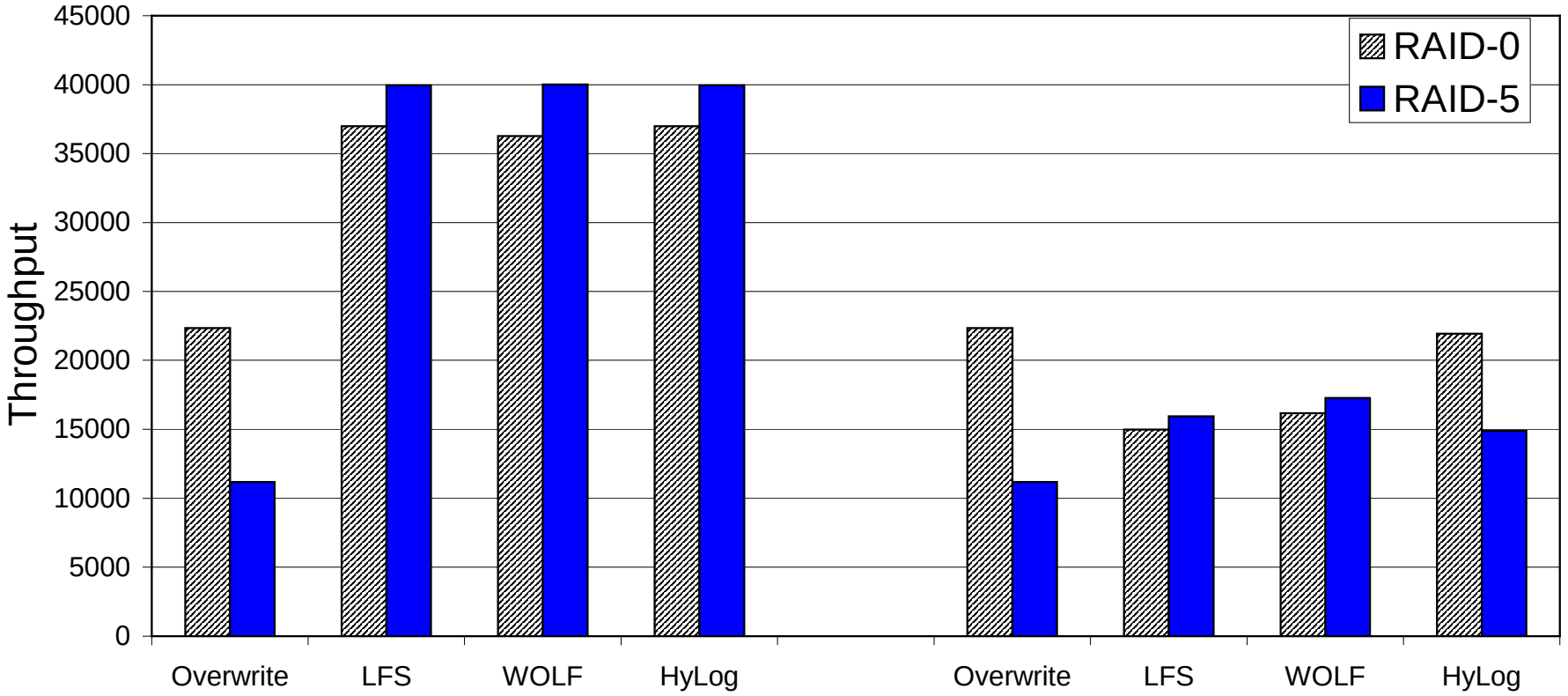

TPC-C trace, disk space utilization 98%, year 1999 disk

# Results - Number of Disks




TPC-C trace, disk space utilization 98%, year 1999 disk

# Results – RAID-5

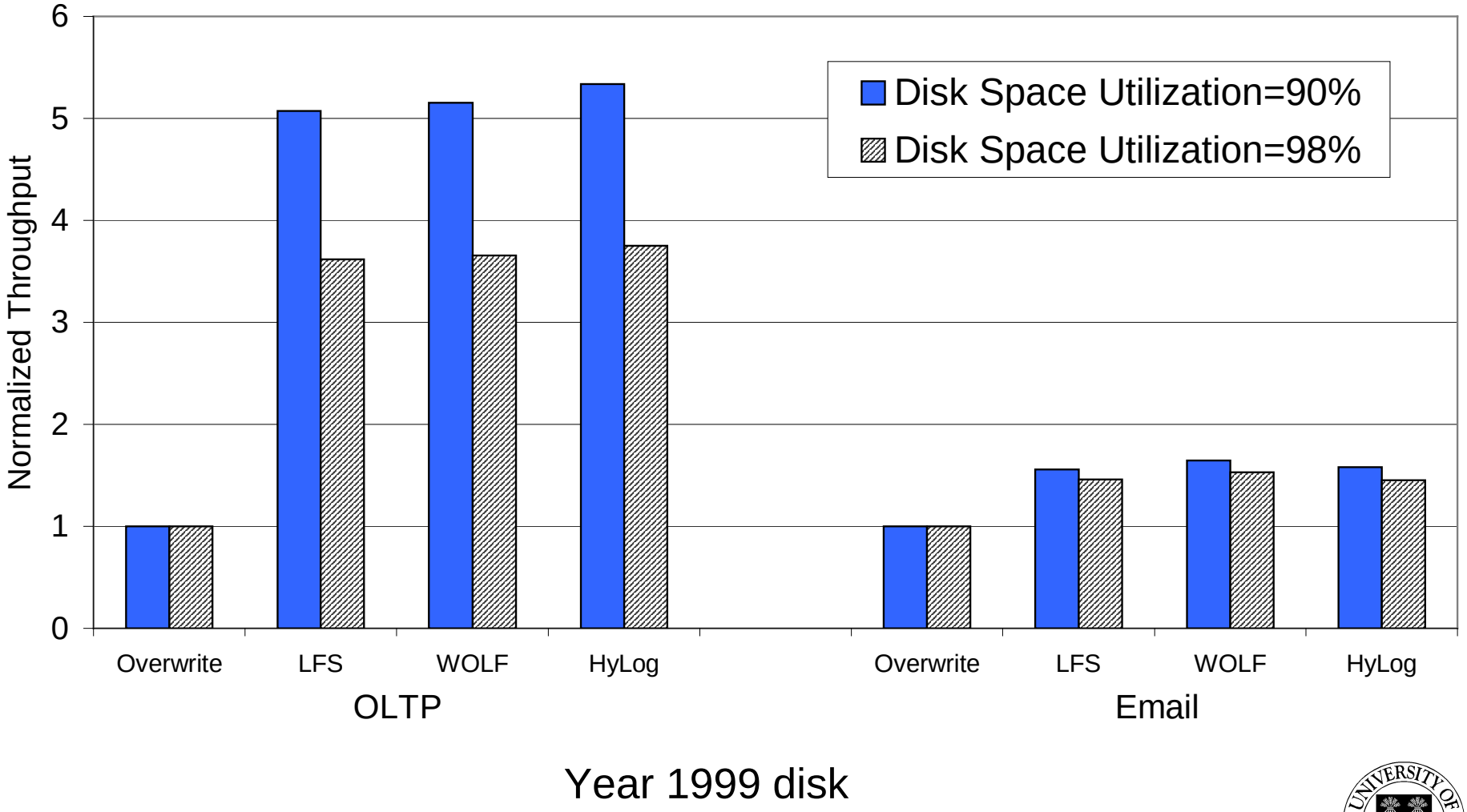



Disk Space Utilization 80%

Disk Space Utilization 98%

TPC-C trace, year 1999 disk, 8-disk RAID-0, 9-disk RAID-5

# Results – Other Traces




Year 1999 disk

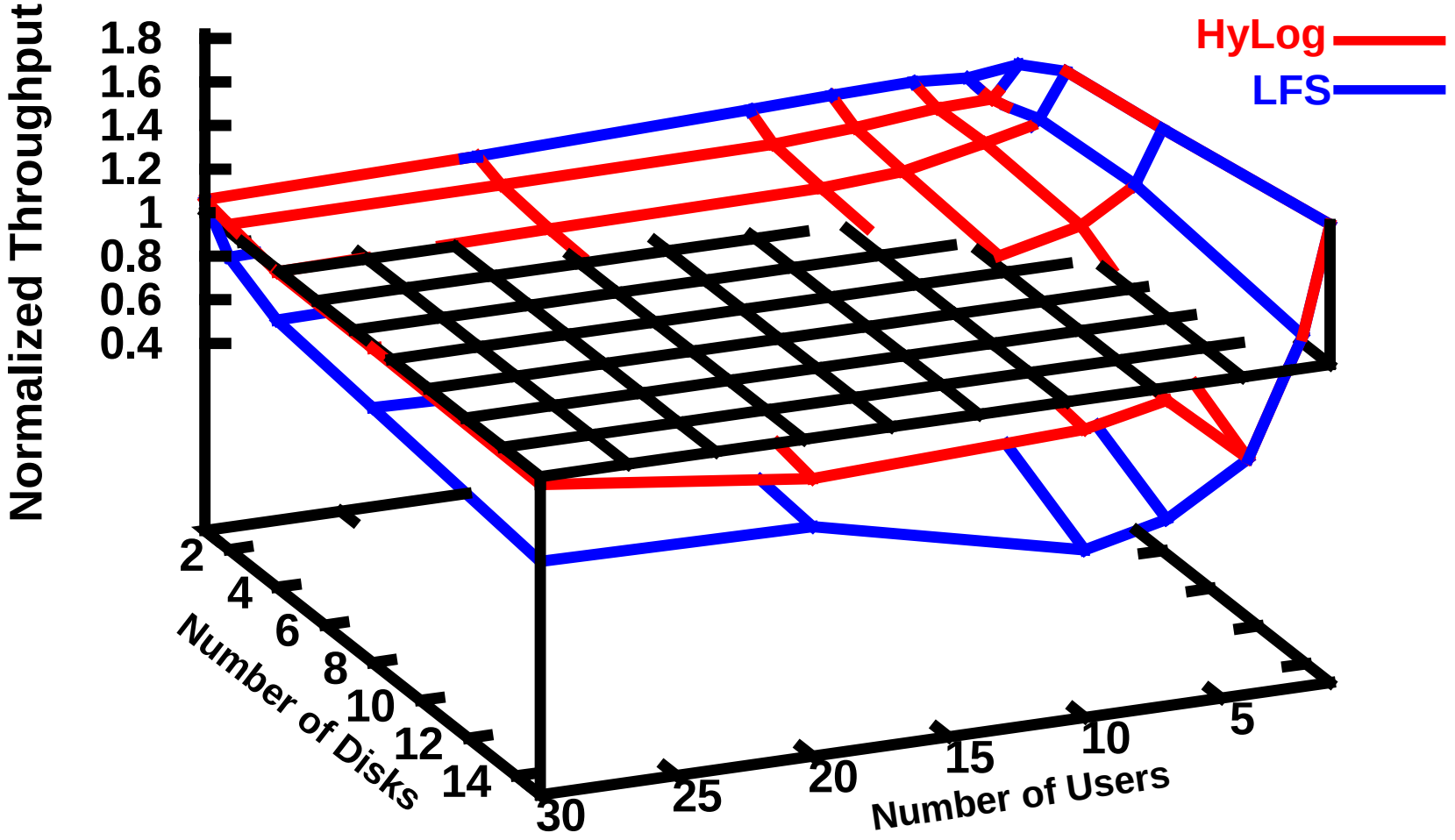
# Conclusions

- On modern and future disks, LFS significantly outperforms Overwrite unless the disk space utilization is very high
- HyLog performs more robustly than LFS and Overwrite

# Future Work

- Add fast recovery support in HyLog
  - All meta-data are considered as hot pages
- Stabilize NetBSD LFS implementation and measure its performance
- Implement and evaluate HyLog in NetBSD

# Results – # Users and Disks




TPC-C trace, disk space utilization 98%, year 1999 disk