Packet Capture in 10-Gigabit Ethernet Environments Using Contemporary Commodity Hardware

Fabian Schneider Jörg Wallerich Anja Feldmann
{fabian,joerg,anja}@net.t-labs.tu-berlin.de

Technische Universtität Berlin
Deutsche Telekom Laboratories

Passive and Active Measurement Conference
5th April 2007

Motivation

Example Scenario: Network security tool at the edge of your network

- need access to packet level data for application layer analysis
- High-speed networks $\Rightarrow$ high data and packet rate

Challenge: capture full packets without missing any packet

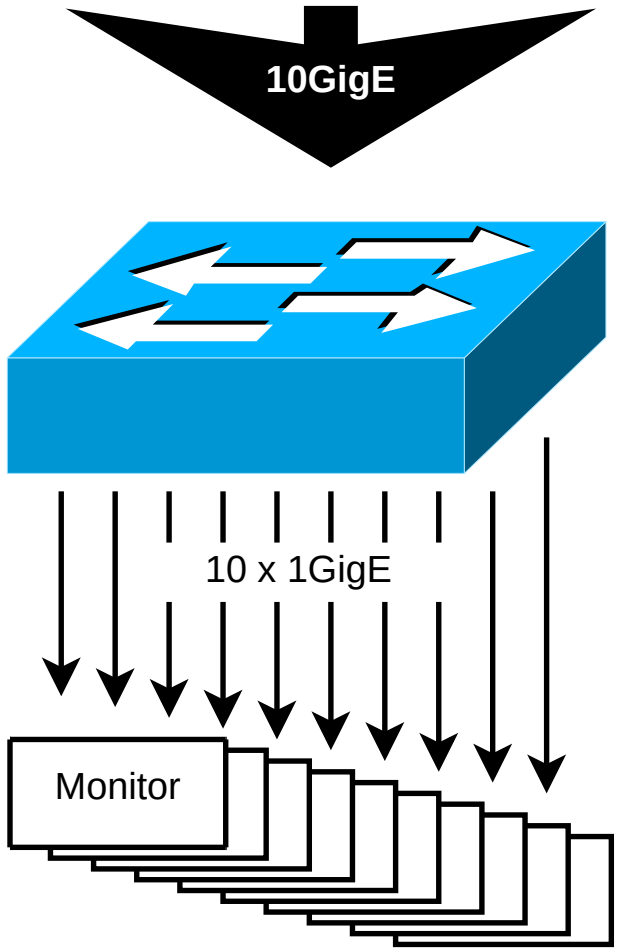
- One approach: specialized hardware
  - e.g. Monitoring cards from Endace
  - Drawbacks: high costs, single purpose

Question: Is it feasible to capture traffic with commodity hardware?

Outline

Approach for 10-Gigabit Monitoring

- *Problem:* No recent host bus or disk system can handle the bandwidth needs of 10-Gigabit environments
- *Solution:* split up traffic and distribute the load (e.g. 10-Gigabit on multiple 1-Gigabit links)
  - Use a switch: e.g. link bundling feature
  - Use specialized hardware
- Keep corresponding data together!



Link Bundling

Feasibility

Etherchannel (Cisco) feature enables link-bundling for:

- higher bandwidth, redundancy, ...
- or load-balancing e. g. for Webservers

Feasibility test:

- Tested on a Cisco 3750
- 1-Gigabit Ethernet link split on eight FastEthernet (100 Mbit/s) links.
- Assign packets to links based on both IP addresses.

$\Rightarrow$ It works with real traffic!

Link Bundling

Load-Balancing

- Simple switches use only MAC addresses
  $\Rightarrow$ Not useful for a router-to-router link
- On a Cisco 3750: any combination of IP and/or MAC addresses
  $\Rightarrow$ is sufficient for our example scenario
- On a Cisco 65xx: MAC's, IP's, and/or Port Numbers

Comparing 1-Gigabit Monitoring Systems

Methodology

- Comparable priced systems with
  - Different processor architectures
  - Different operating systems
- Task of those systems:
  - Capture full packets
  - Do not analyze them (Out-of-Scope)
- Workload:
  - All system are subject to identical input
  - Increase bandwidth up to a fully loaded Gigabit link
  - Realistic packet size distribution
- Measurement Categories:
  - Capturing Rate: number of captured packets (simple libpcap app)
  - System Load: CPU usage while capturing (simple top like app)

Systems under Test

Two examples of any of the systems:

- One installed with Linux
- The other with FreeBSD

First set of systems purchased in 2005:

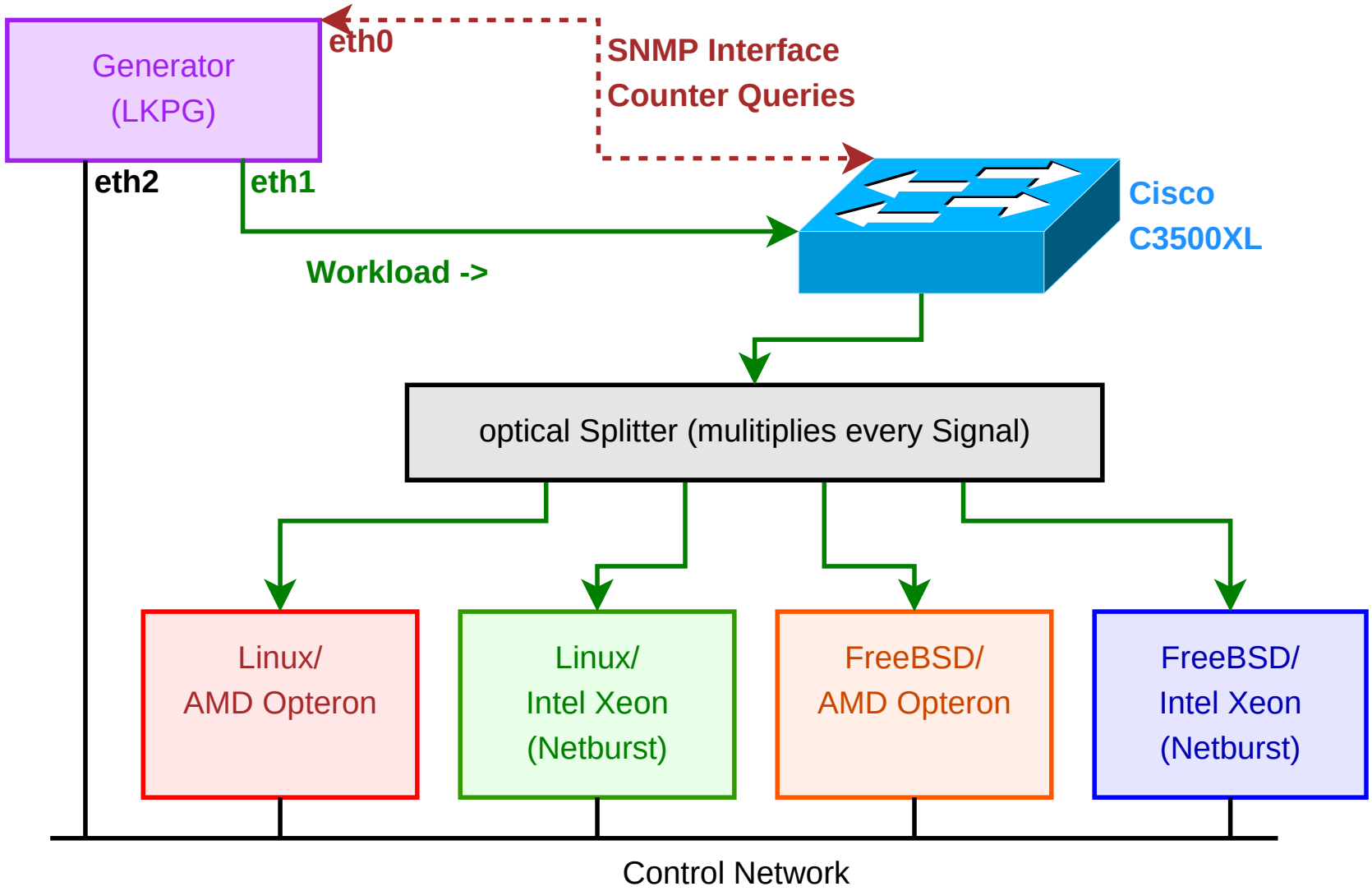
- 2x AMD Opteron 244 (1 MB Cache, 1.8 GHz),
- 2x Intel Xeon (Netburst, 512 kB Cache, 3.06 GHz),

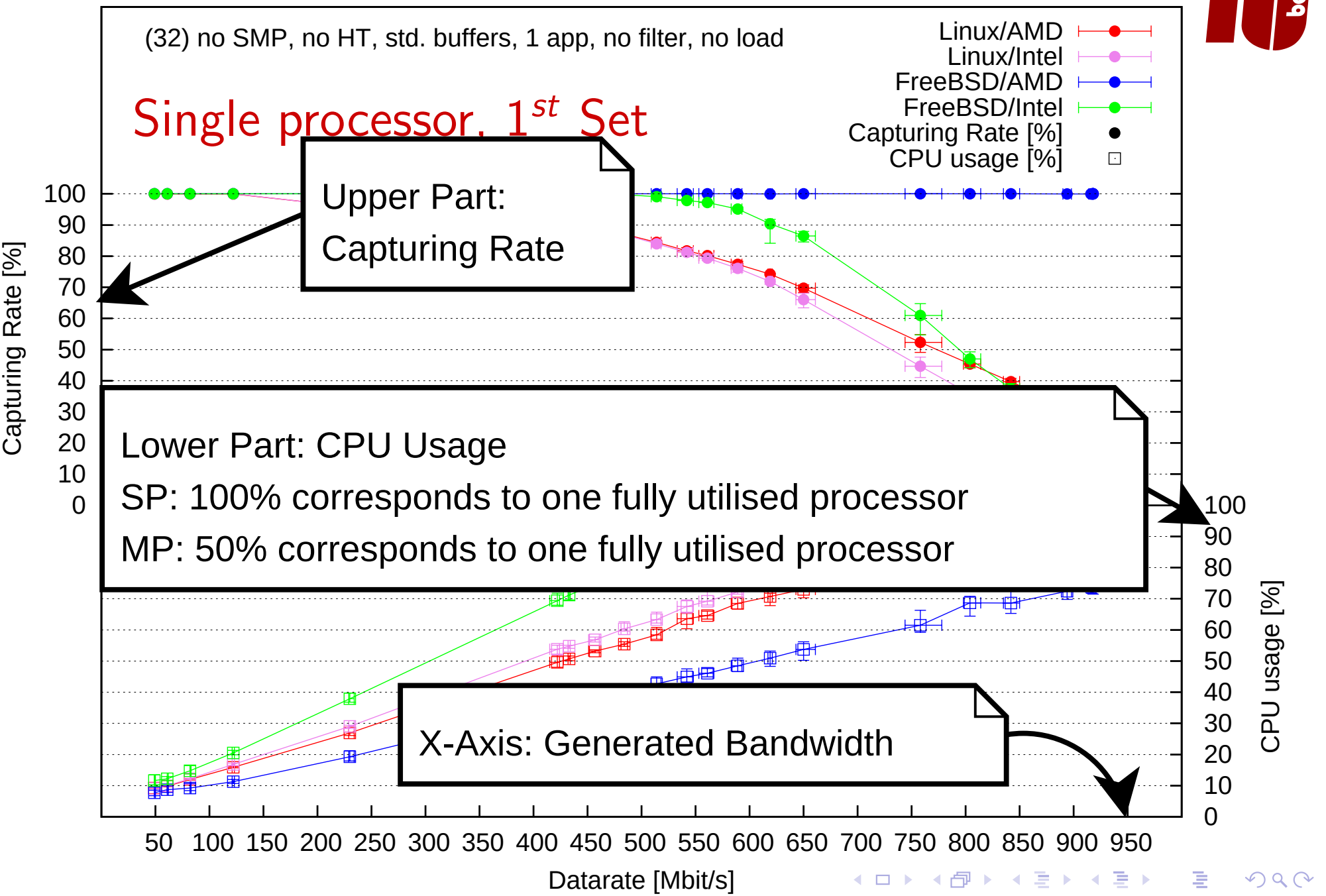
Second set purchased in 2006:

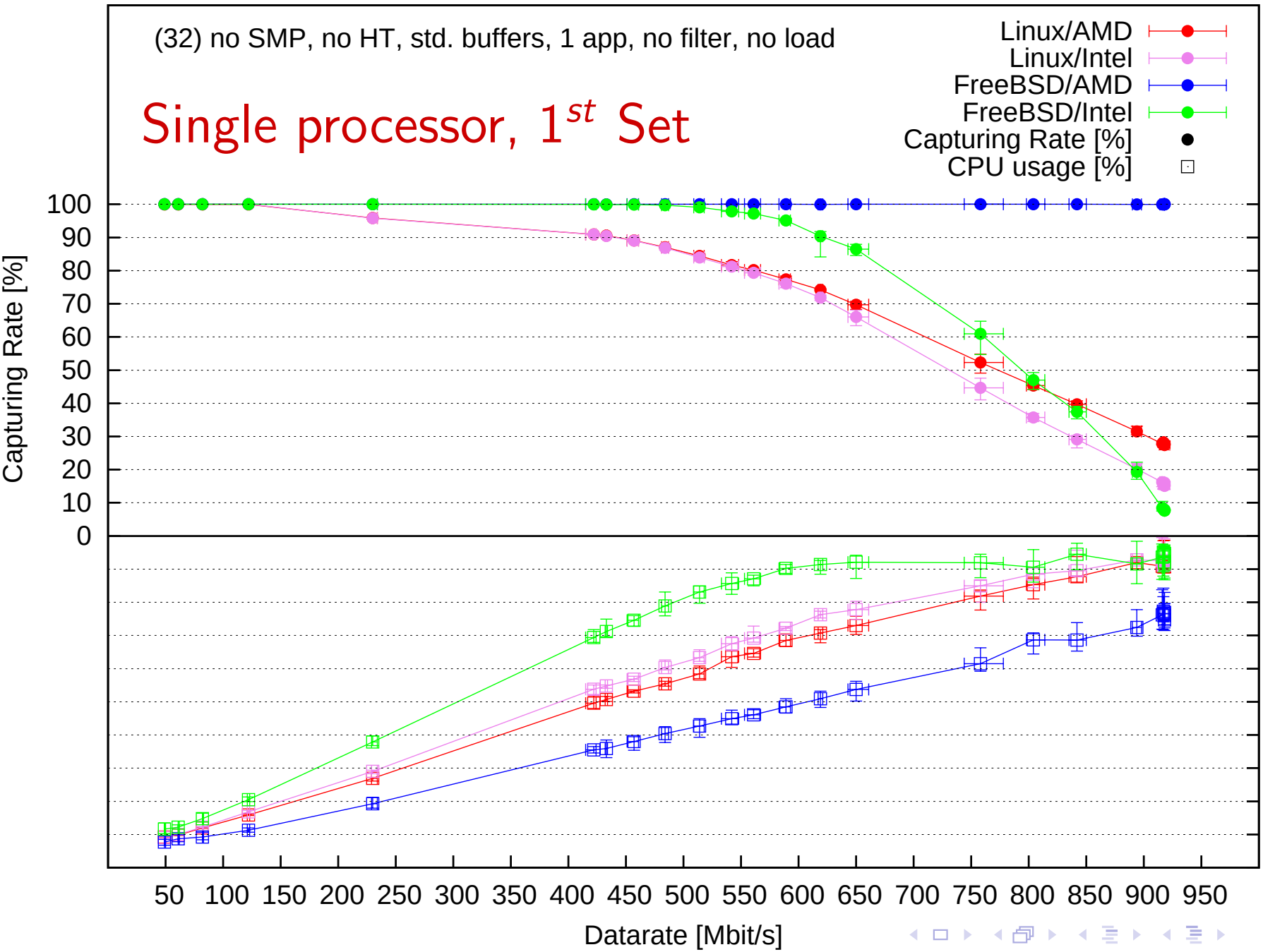
- 2x Dual Core AMD Opteron 270 (1 MB Cache, 2.0 GHz)

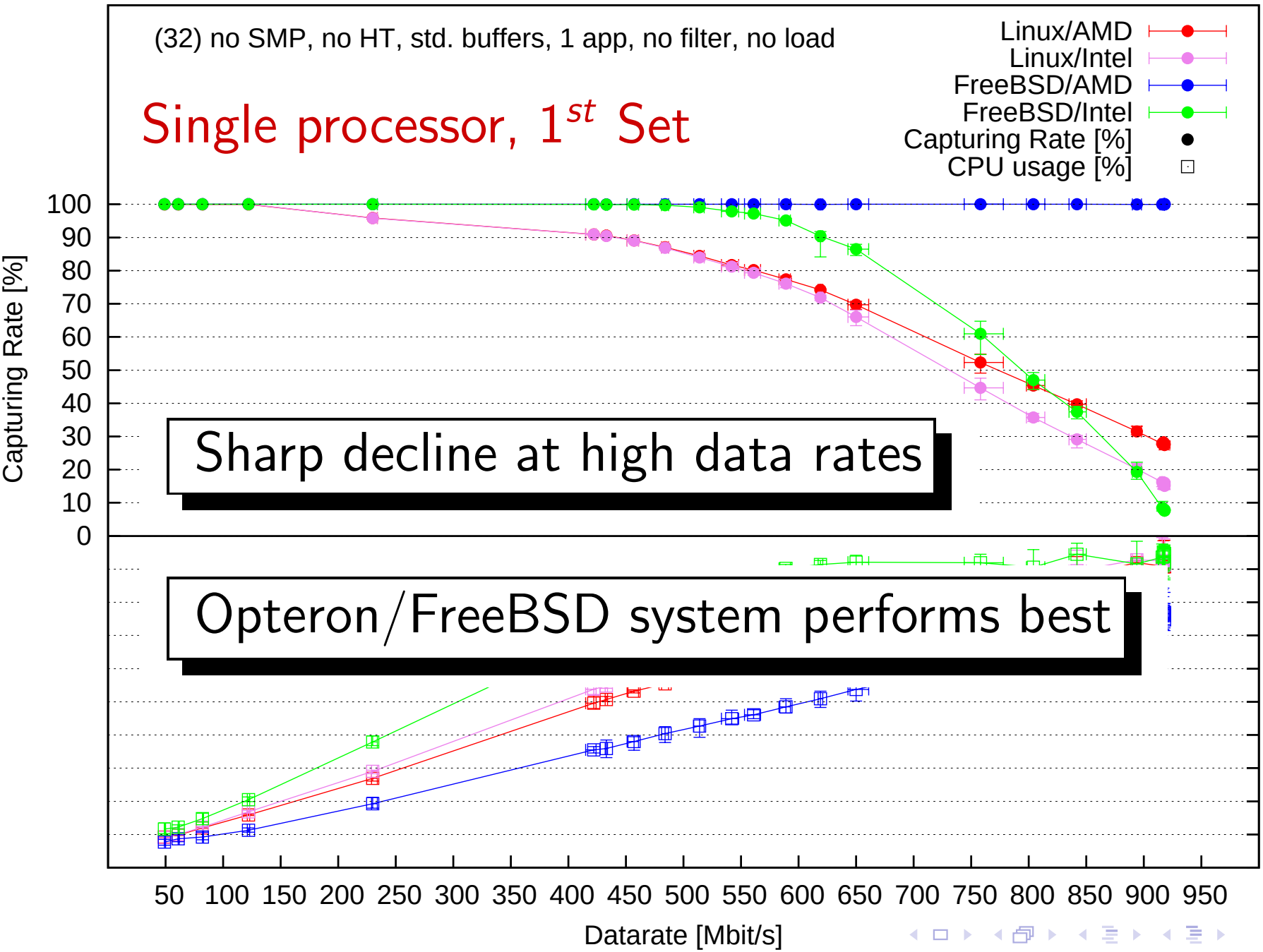
All: 2 Gbytes of RAM, optical Intel Gigabit Ethernet card, RAID array

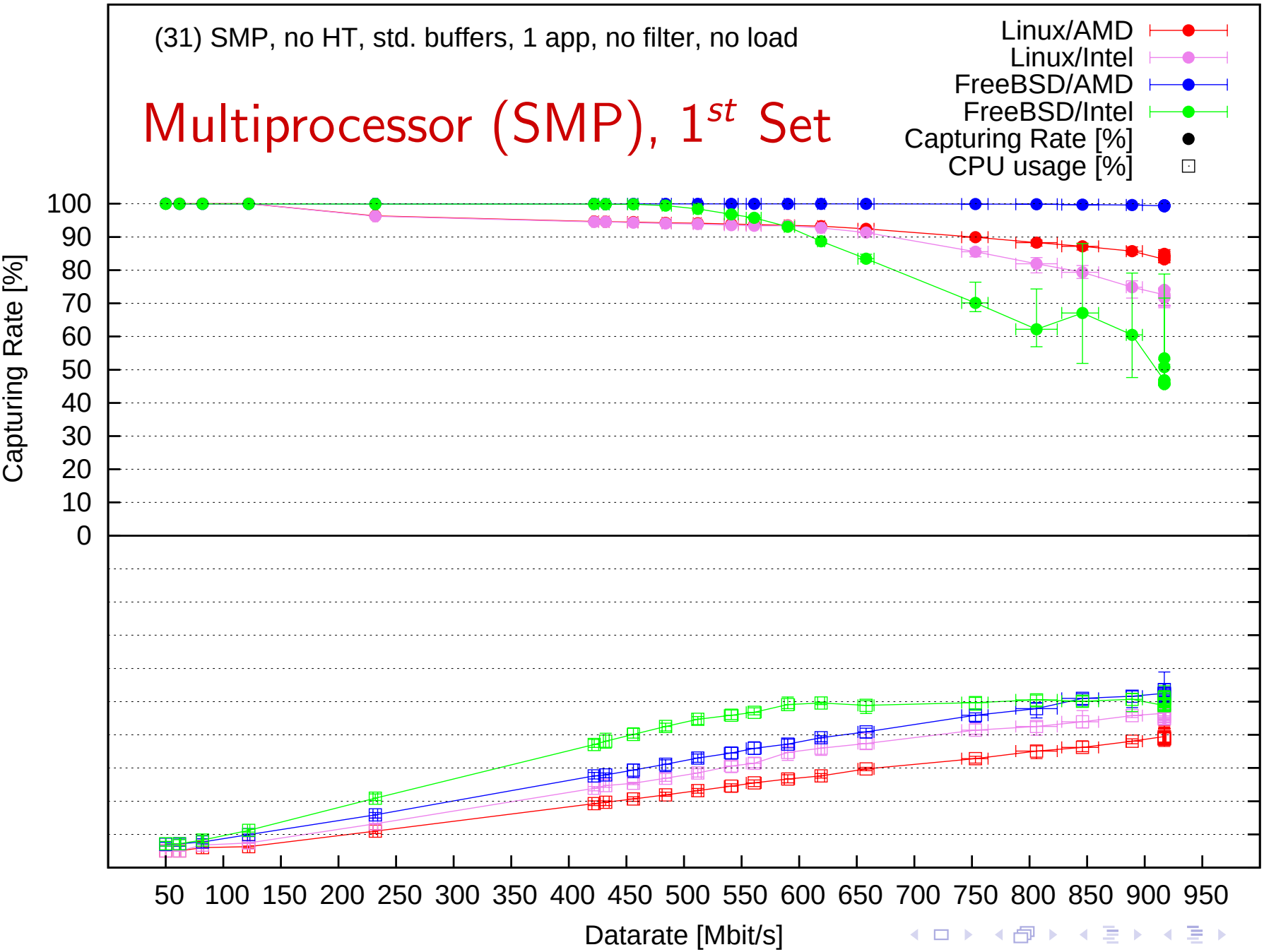
Measurement Setup

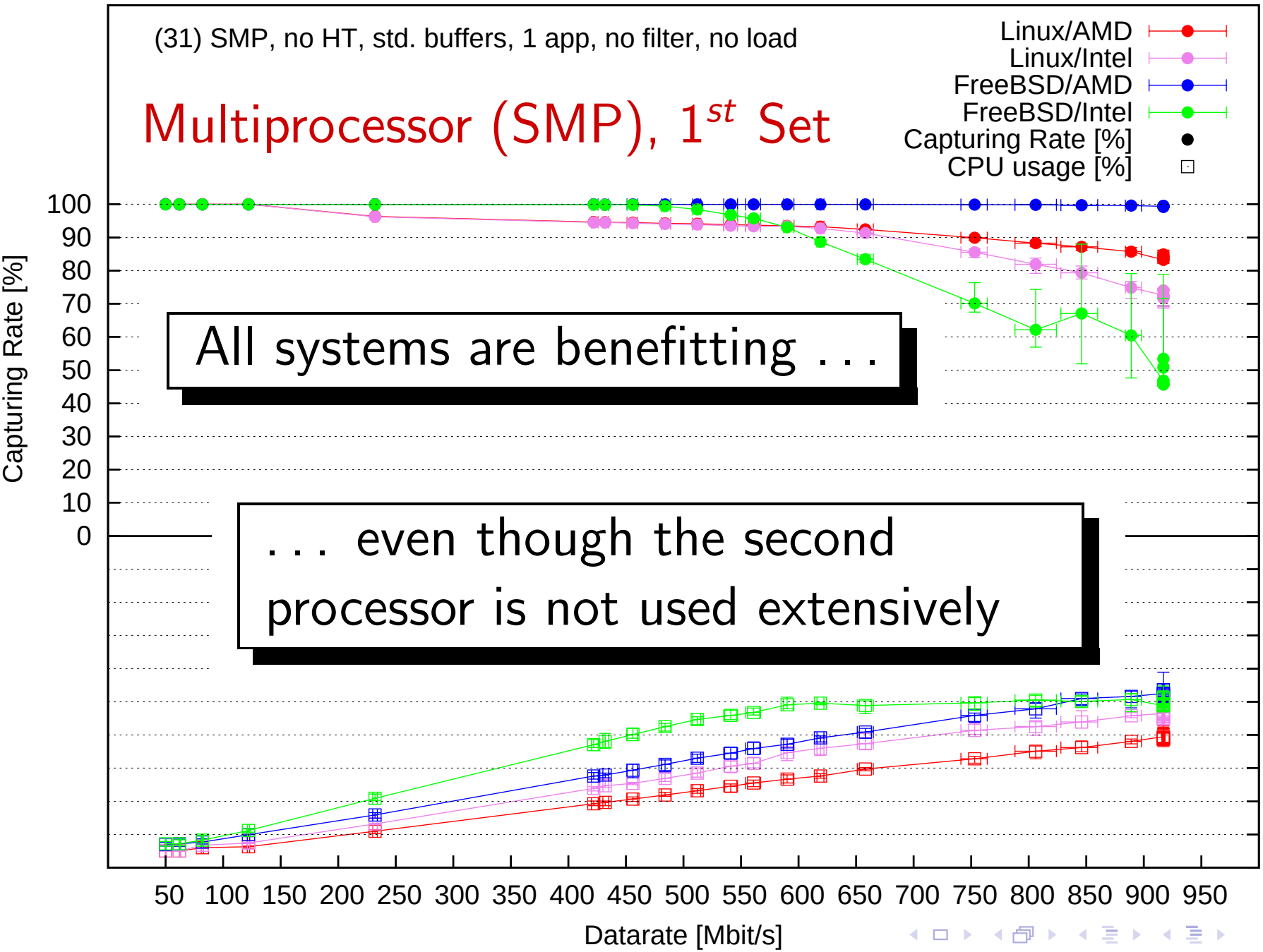



(32) no SMP, no HT, std. buffers, 1 app, no filter, no load
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
Single processor, 1st Set
Upper Part:
Capturing Rate
Lower Part: CPU Usage
SP: 100% corresponds to one fully utilised processor
MP: 50% corresponds to one fully utilised processor
X-Axis: Generated Bandwidth
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950


(32) no SMP, no HT, std. buffers, 1 app, no filter, no load
Single processor, 1st Set
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0
10
20
30
40
50
60
70
80
90
100
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950


(32) no SMP, no HT, std. buffers, 1 app, no filter, no load
Single processor, 1st Set
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
Sharp decline at high data rates
Opteron/FreeBSD system performs best
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950


(31) SMP, no HT, std. buffers, 1 app, no filter, no load
Multiprocessor (SMP), 1st Set
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0
10
20
30
40
50
60
70
80
90
100
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950


(31) SMP, no HT, std. buffers, 1 app, no filter, no load
Multiprocessor (SMP), 1st Set
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
All systems are benefitting . . .
. . . even though the second processor is not used extensively
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950

Increasing Buffer Sizes ?

Setup:

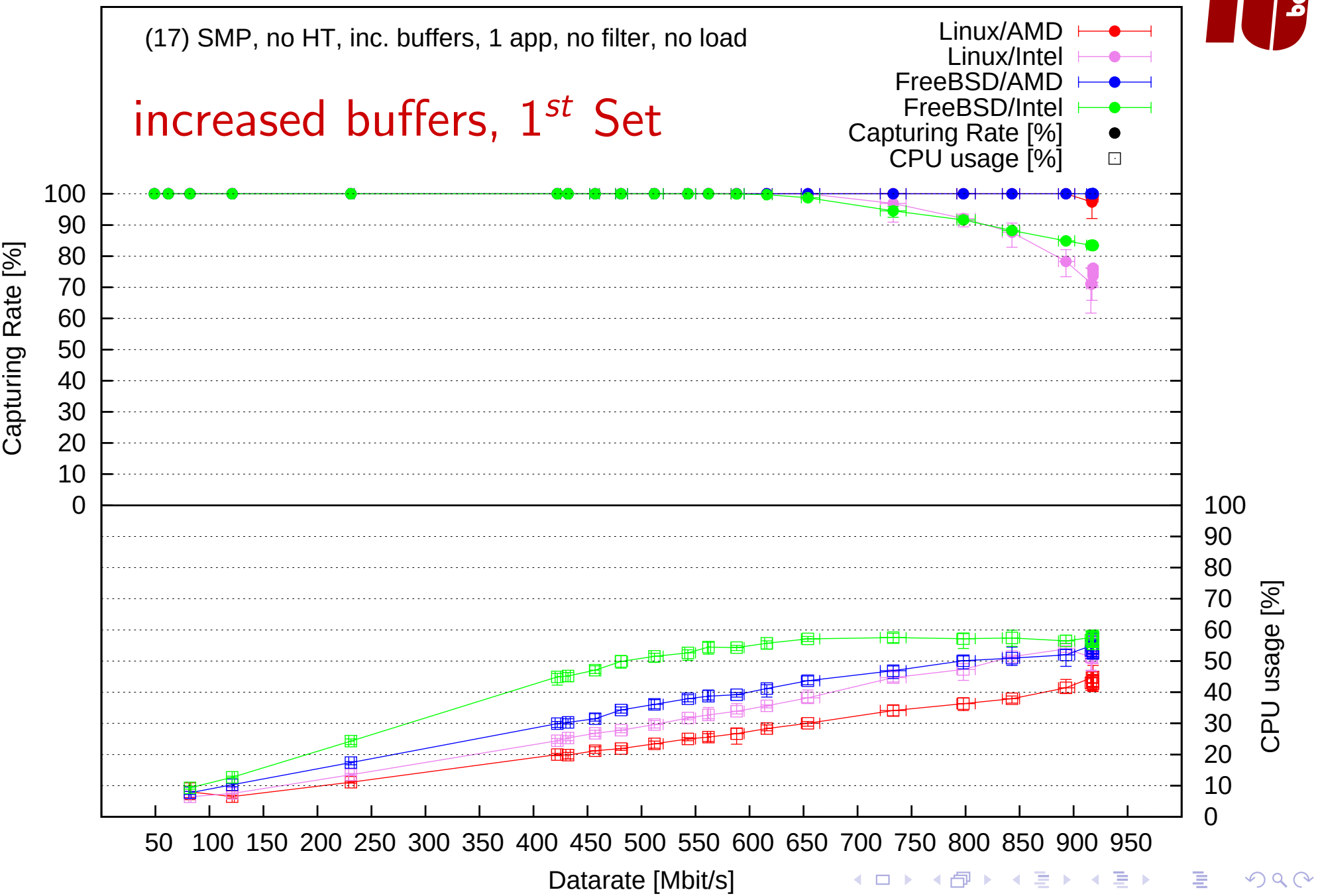
- First Set of systems
- Dual processor
- Increased buffer sizes

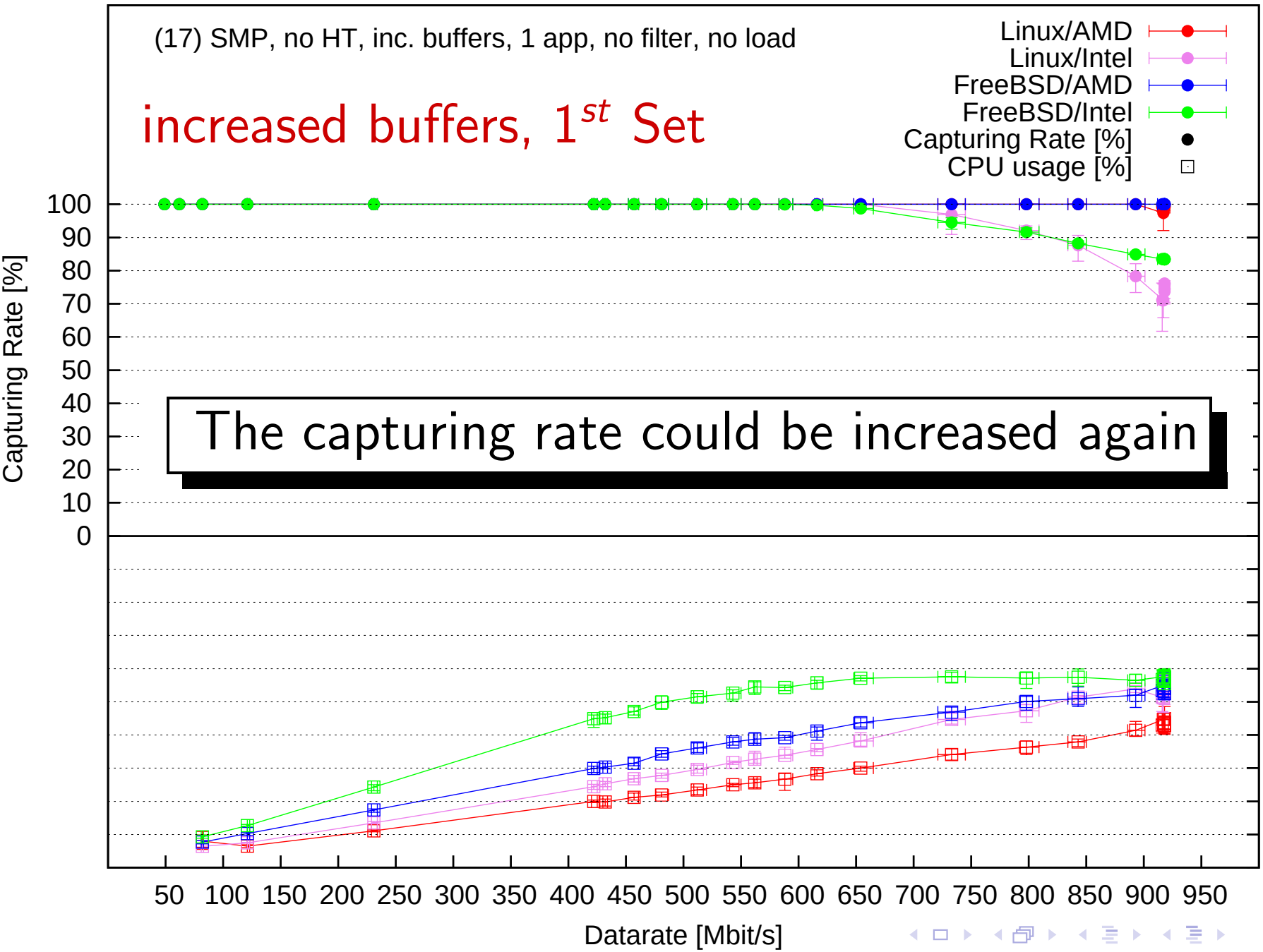
Operating system buffers:

FreeBSD 6.x: `sysctl`'s `net.bpf.bufsize` and `net.bpf.maxbufsize`

FreeBSD 5.x: `sysctl`'s `debug.bpf_bufsize` and `debug.maxbpf_bufsize`

Linux: `/proc/sys/net/core/rmem_default`, `/proc/sys/net/core/rmem_max`, and `/proc/sys/net/core/netdev_max_backlog`


(17) SMP, no HT, inc. buffers, 1 app, no filter, no load
increased buffers, 1st Set
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950


(17) SMP, no HT, inc. buffers, 1 app, no filter, no load
increased buffers, 1st Set
Linux/AMD
Linux/Intel
FreeBSD/AMD
FreeBSD/Intel
Capturing Rate [%]
CPU usage [%]
The capturing rate could be increased again
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950

Additional Insights

First set of measurements

- Filtering is cheap with respect to its benefit (reduced packet processing)
- Running multiple capturing applications concurrently leads to bad performance.
- Measurement with additional compression show some advantage for Intel Systems
- Intel Hyperthreading does not change the performance
- using the memory-map patch from Phil Woods (Linux only) does help

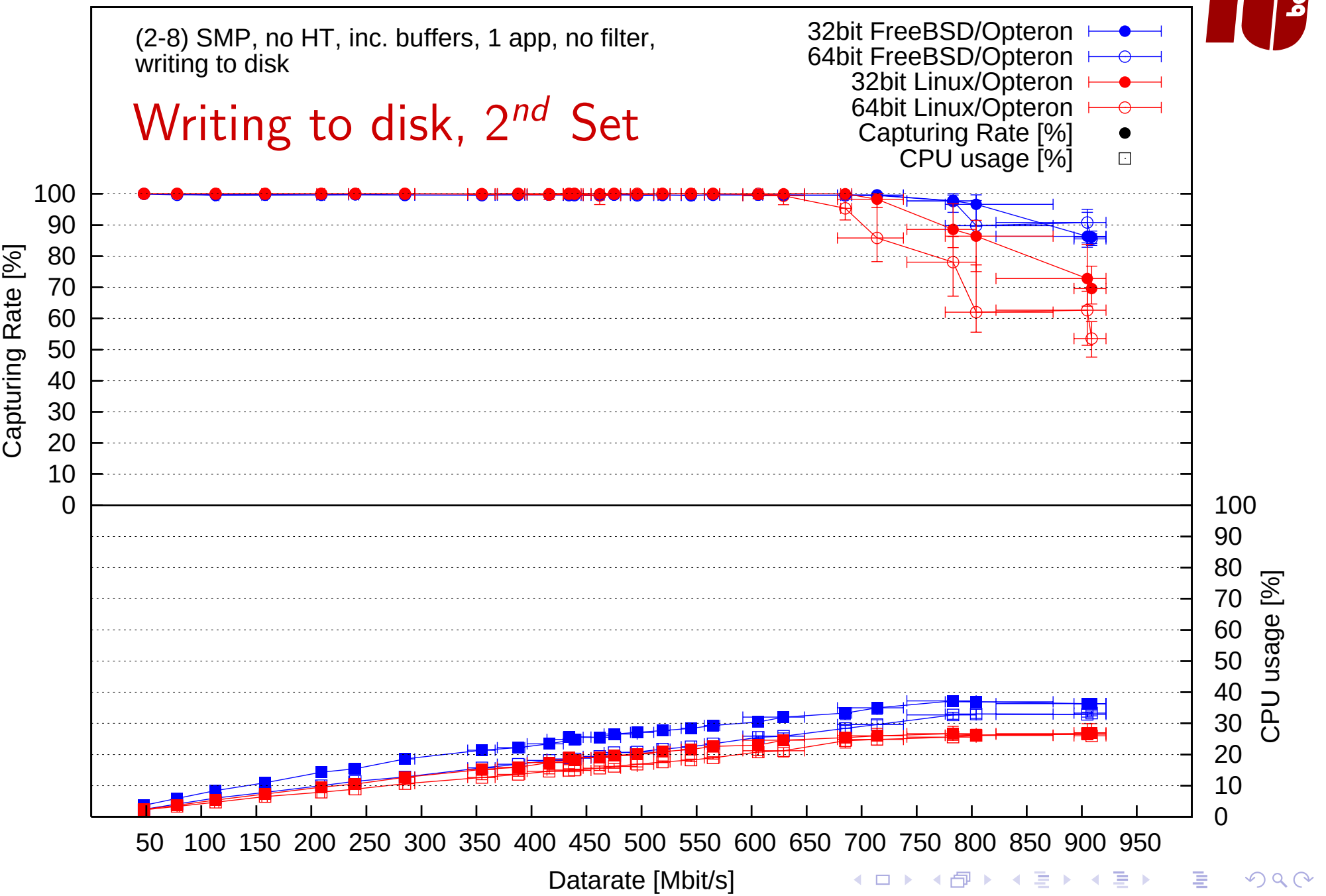
Writing packets to disk?

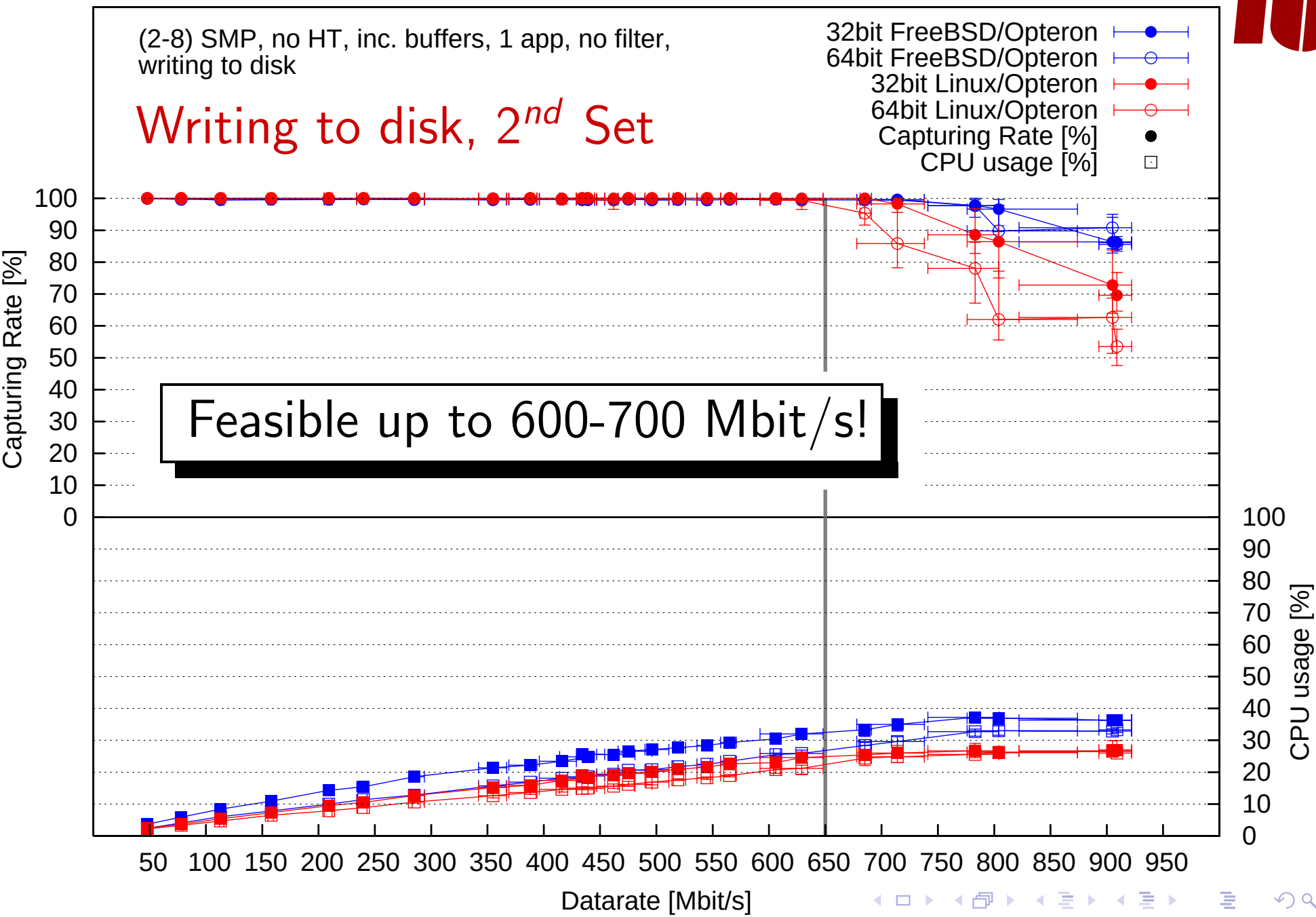
preliminary measurements have shown that

- newer system do not lose any packet: with buffers, SMP, etc.
- disk writing speed is not the bottleneck

Setup:

- Newer systems: 2x dual core AMD systems
  $\Rightarrow$ CPU usage: 25% correspond to one fully utilized processor
- Increased buffer sizes
- No filter
- Linux vs. FreeBSD
- 32bit vs. 64bit OS'es


(2-8) SMP, no HT, inc. buffers, 1 app, no filter, writing to disk
Writing to disk, 2nd Set
32bit FreeBSD/Opteron
64bit FreeBSD/Opteron
32bit Linux/Opteron
64bit Linux/Opteron
Capturing Rate [%]
CPU usage [%]
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950


(2-8) SMP, no HT, inc. buffers, 1 app, no filter, writing to disk
Writing to disk, 2nd Set
32bit FreeBSD/Opteron
64bit FreeBSD/Opteron
32bit Linux/Opteron
64bit Linux/Opteron
Capturing Rate [%]
CPU usage [%]
Feasible up to 600-700 Mbit/s!
Capturing Rate [%]
CPU usage [%]
Datarate [Mbit/s]
0 10 20 30 40 50 60 70 80 90 100
50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950

Additional Insights

Second set of measurements

- additional load (copying the packets in memory) shows significantly better performance for FreeBSD
- 64bit systems drop more packets
- Using 4 cores (2x Dual Core) is slightly better than 2 cores (1x Dual Core)

Summary

- Split up 10-Gigabit on multiple 1-Gigabit monitoring systems
- FreeBSD/AMD Opteron combination in general performs best
- Utilizing multiple processors proves to be benefitting
- Choosing large enough buffer size is important
- Capturing full traces to disk is feasible up to about 600-700 Mbit/s

For further information see: High Performance Packet Capture
http://www.net.t-labs.tu-berlin.de/research/hppc/