

Chronicle: Capture and Analysis of NFS Workloads at Line Rate

Ardalan Kangarlou, Sandip Shete, and John Strunk
Advanced Technology Group

© 2015 NetApp, Inc. All rights reserved.

Motivation

Goal: To gather insights from customer workloads via **trace capture** and **real-time analysis**.

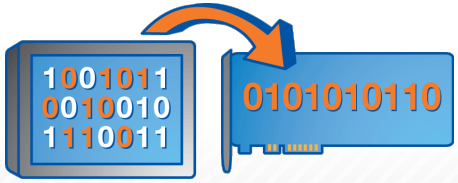
Design	• Evaluating new algorithms
Test	• Designing representative benchmarks
Support	• Diagnosing problems and misconfigurations
Sales	• Identifying appropriate storage platforms

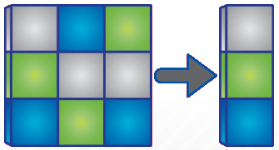
© 2015 NetApp, Inc. All rights reserved.

Use Case 1: Research

An I/O-by-I/O view of workloads is preferred for research in

- Data caching, prefetching, and tiering techniques
- Data deduplication analysis
- Creation of representative benchmarks for real-world workloads
- Studying data access and growth patterns over time









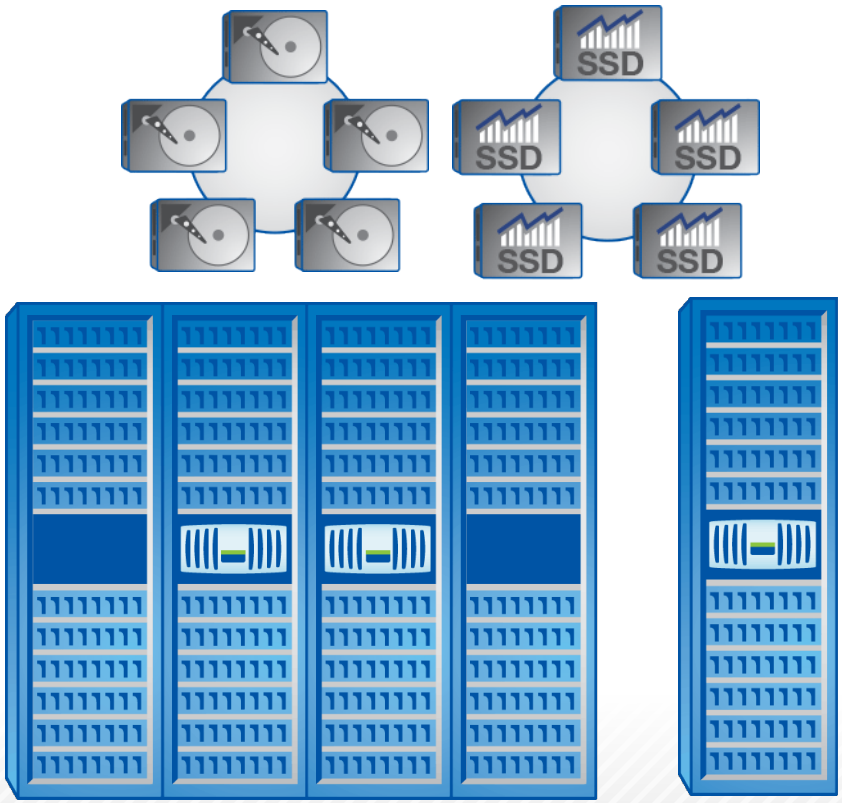
© 2015 NetApp, Inc. All rights reserved.

Use Case 2: Sales

Workload Sizing Questionnaire

- I/O rate?
- Number of clients?
- Random read working set size?
- Ratio of random reads vs. random writes?



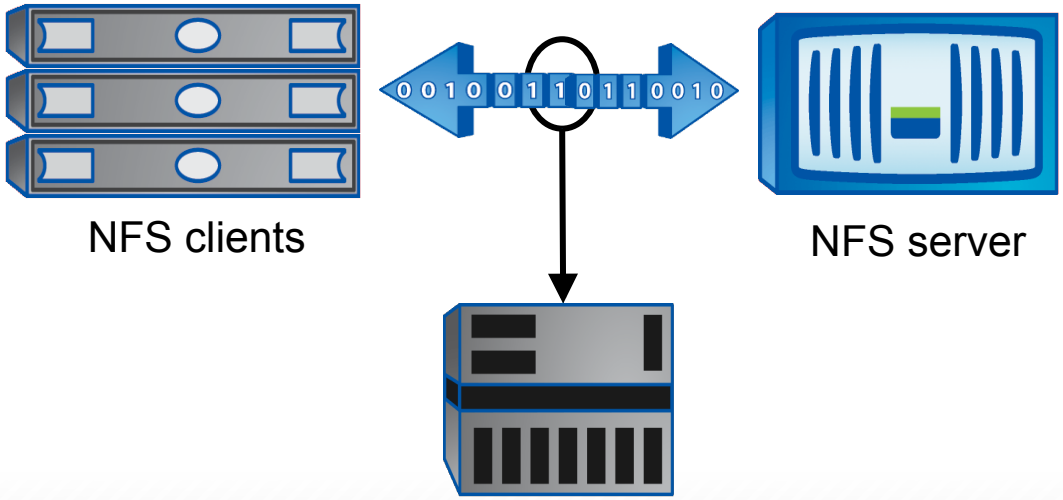


© 2015 NetApp, Inc. All rights reserved.

NetApp

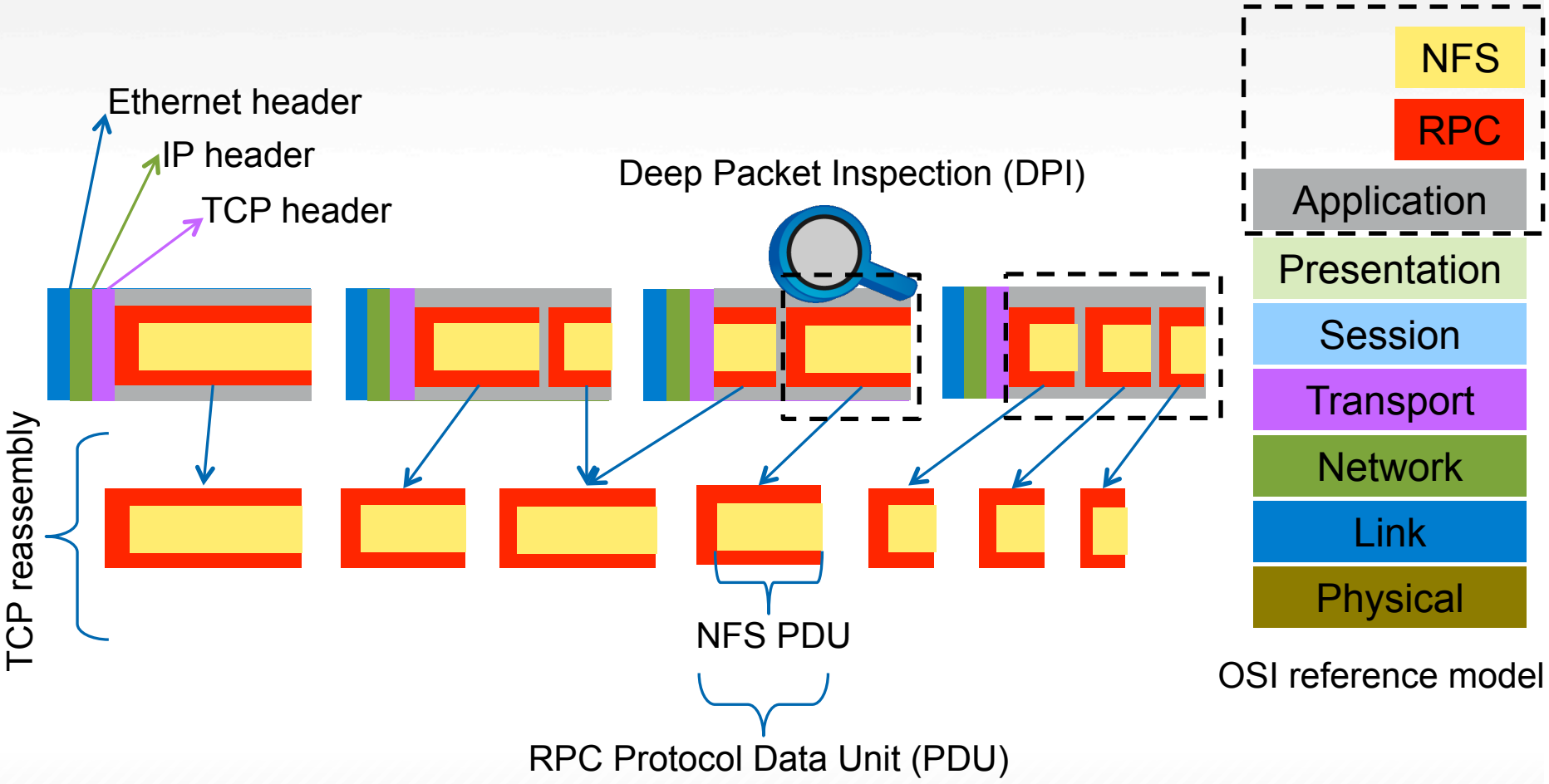
Chronicle Framework

- Capture and real-time analysis of NFSv3 workloads
- Rates above 10Gb/s
- Commodity hardware
- Passive network monitoring
- Runs for days to weeks
- Programmer-friendly and extensible API



© 2015 NetApp, Inc. All rights reserved.

Background – NFS Traffic



© 2015 NetApp, Inc. All rights reserved.

Background – NAS Tracing

- [Ellard-FAST'03] and [Leung-ATC'08]
  - NFS and CIFS workload analysis based on pcap traces
- Driverdump [Anderson-FAST'09]
  - The fastest software-only solution operating @ 1.4Gb/s
  - Network driver stores packets directly in the pcap format
- Main limitations for our use case:
  - Packet capture through the lossy pcap interface

```
root@chronicle02:/mnt/traces/tmp# tcpdump -ieth2 -w eth2.pcap -s0
tcpdump: WARNING: eth2: no IPv4 address assigned
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 65535 bytes
^C680108 packets captured
2507441 packets received by filter
1827319 packets dropped by kernel
root@chronicle02:/mnt/traces/tmp#
```

  - High storage bandwidth and capacity requirements
    - Trace capture @ 10Gb/s for a week requires 750TB of storage!
  - pcap traces require offline parsing of data
  - Stateless parsing: inability to parse fields that span packets

© 2015 NetApp, Inc. All rights reserved.

Our Approach – Efficient Trace Storage

- Instead of storing raw packets (i.e., pcap format)
  - Use DPI to identify fields of interests in packets
  - Checksum read and write data for data deduplication analysis
  - Leverage DataSeries [Anderson-OSR'09] as the trace format
    - Efficient storage of structured, serial data
    - Inline compression, non-blocking I/O, and delta encoding
    - Extents for storing RPC-, NFS-, and network-level information as well as read/write data checksums

Simplified RPC extent

record_id	request_ts	reply_ts	operation	client	server	transaction_id

- With above techniques, a single standard disk can handle the storage bandwidth requirements for tracing at line rate!

© 2015 NetApp, Inc. All rights reserved.

Background – Packet Processing

- Active area of research in software routing and network security:
  - Common techniques: partitioning and pipelining work across cores, judicious placement and scheduling of threads, minimizing synchronization overhead, batch processing, recycling allocated memory, zero-copy parsing, and bypassing kernel
- Some examples:
  - RouteBricks [Dobrescu-SOSP'09]:
    - Packet forwarding @ 10Gb/s, packet routing @ 6.4Gb/s, and IPsec @ 1.4Gb/s
    - Required "tedious manual tuning"
  - NetSlices [Marian-ANCS'12]:
    - Fixed mapping between packets and cores to support 9.7Gb/s routing throughput
  - Click [Kohler-TOCS'00, Chen-USENIXATC'01]
    - Kernel-mode (3-4 Mpps per core) and user-mode (490 Kpps)
  - netmap [Rizzo-USENIXATC'12, Rizzo-INFOCOM'12]:
    - Send/receive packets at line rate (14.88 Mpps @ 10Gb/s); 20ns/pkt vs. 500-1000ns/pkt for sockets
    - User-space Click on netmap resulted in the same throughput as kernel-mode Click (3.9Mpps)

© 2015 NetApp, Inc. All rights reserved.

Packet Processing Frameworks cont.

- Major limitations for our use case:
  - A network-centric view of packet processing:
    - No DPI, TCP reassembly, and stateful parsing across packets
    - Fixed, small per-packet processing cost
    - Maintaining low latency is as important as high throughput
  - Manual tuning for specific hardware platforms
  - Management of shared resources and state (e.g., locks, thread-safe queues, etc.)
  - Kernel implementations are hard to extend with custom libraries

Main Challenge:

To extend proven packet processing techniques to the application layer, for a more CPU-intensive use case, and in a programmer-friendly manner!

© 2015 NetApp, Inc. All rights reserved.

Our Approach – Packet Processing

- Libtask: A user-space actor model library
  - Performance:
    - Seamless scalability to many cores
    - Implicit batching of work to support high throughput
  - Flexibility and usability:
    - A pluggable, pipelined architecture
    - Portable software by hiding hardware configuration from users
    - Unburden application programmers of concurrency bugs
- Leverage netmap instead of libpcap for reading packets
  - Efficient framework for bypassing kernel based on modified network drivers
  - We extended netmap to support jumbo frames

© 2015 NetApp, Inc. All rights reserved.

Background – Actor Model Programming



Actor: A computation agent that processes tasks



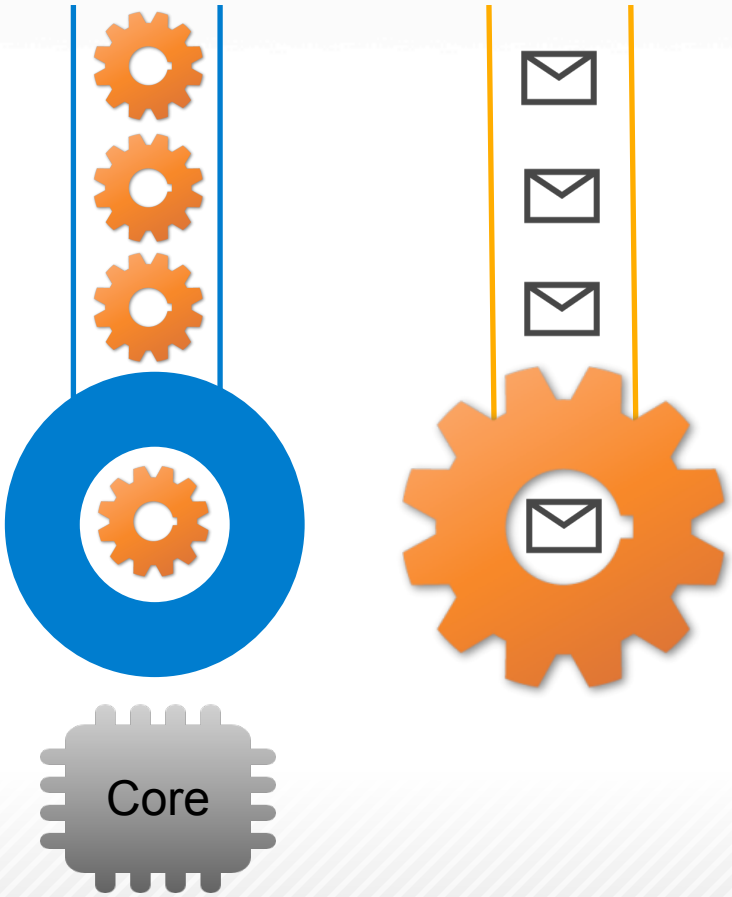
Message: Information to be shared with a target actor about a task or tasks



© 2015 NetApp, Inc. All rights reserved.

Libtask

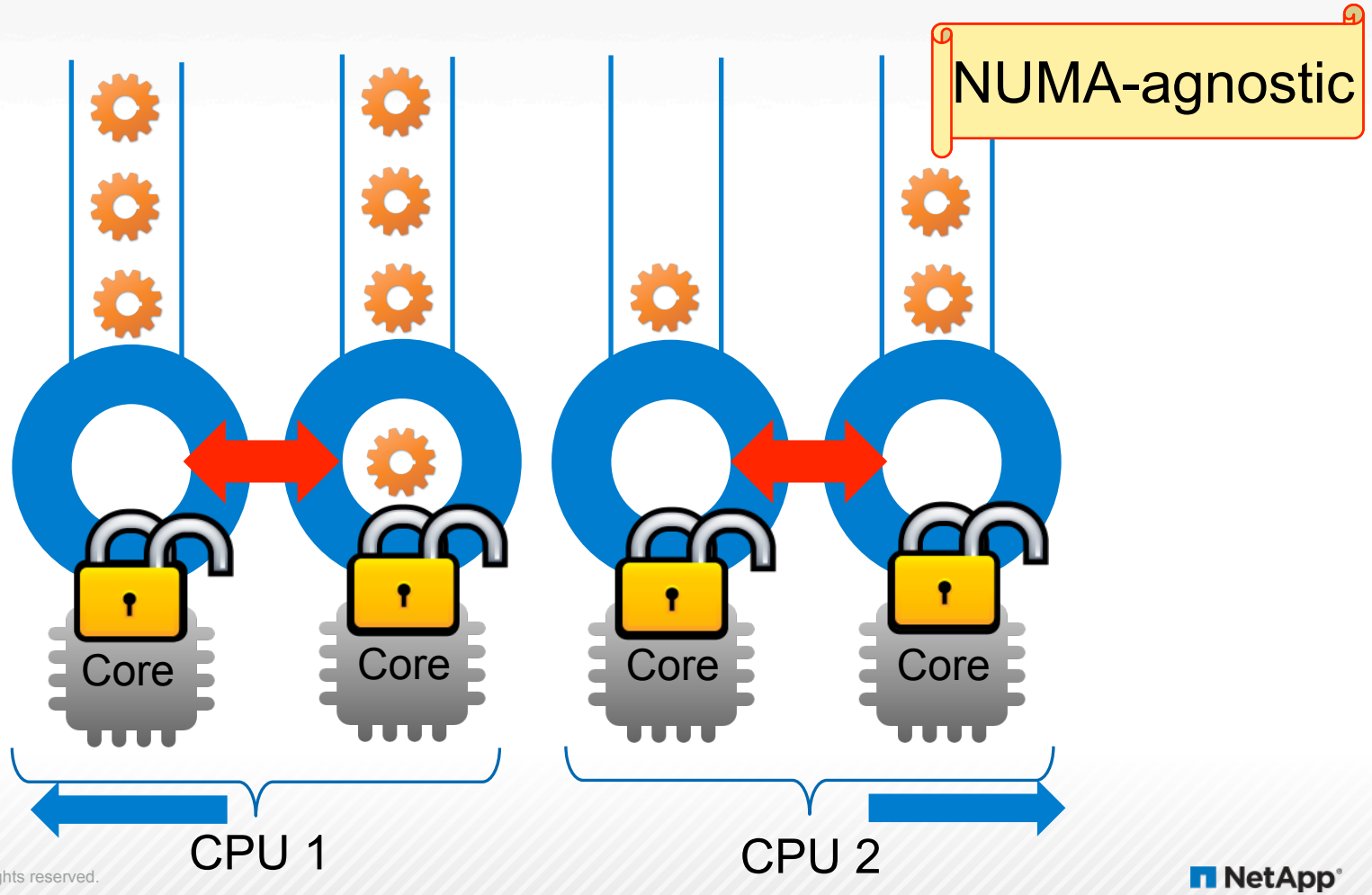
- A light-weight actor model library written in C++
- Three constructs: Scheduler, Process, and Message



© 2015 NetApp, Inc. All rights reserved.

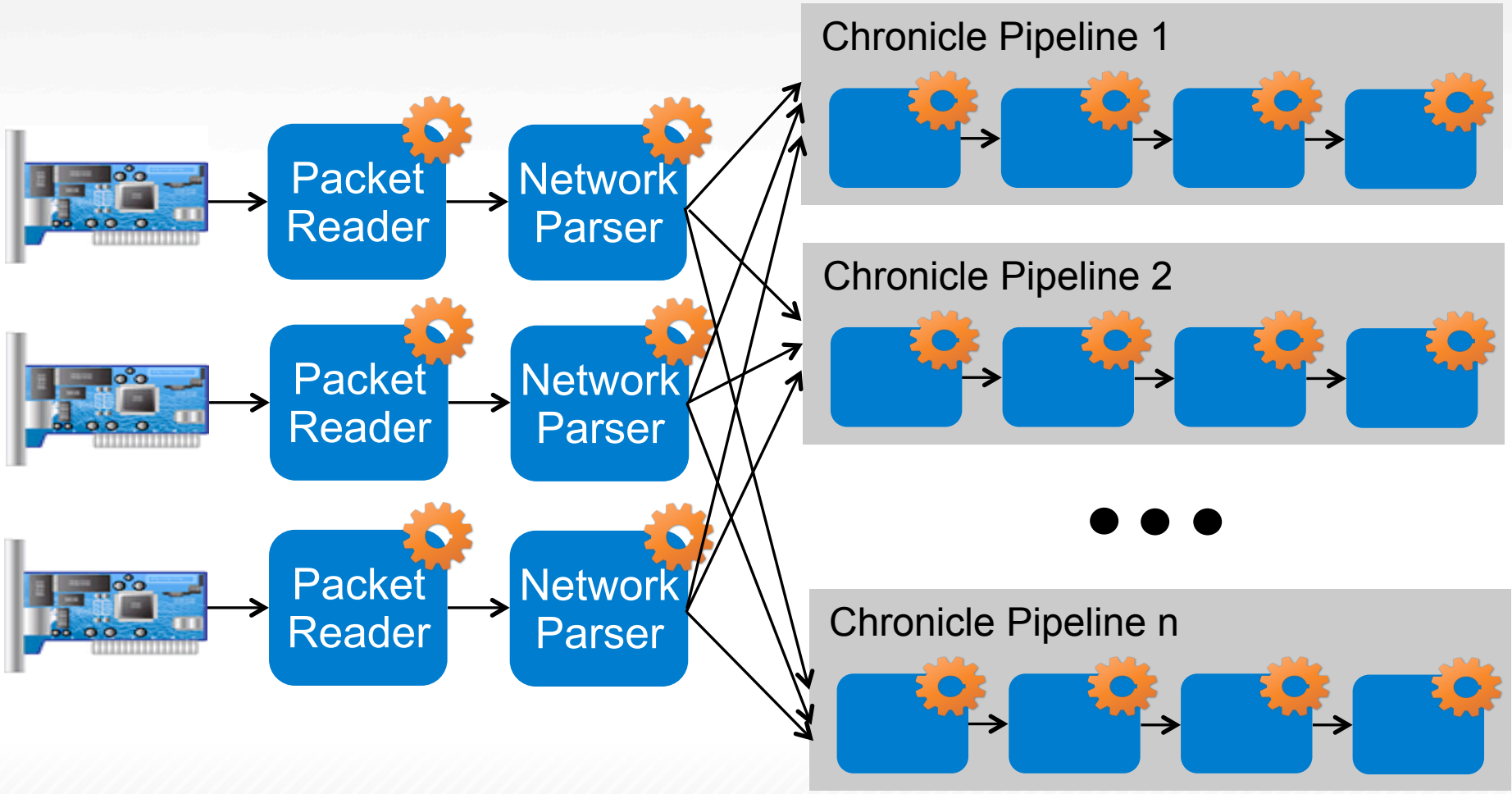
Libtask cont.

- Load balancing and seamless scalability
- Two versions of Libtask: NUMA-aware and NUMA-agnostic



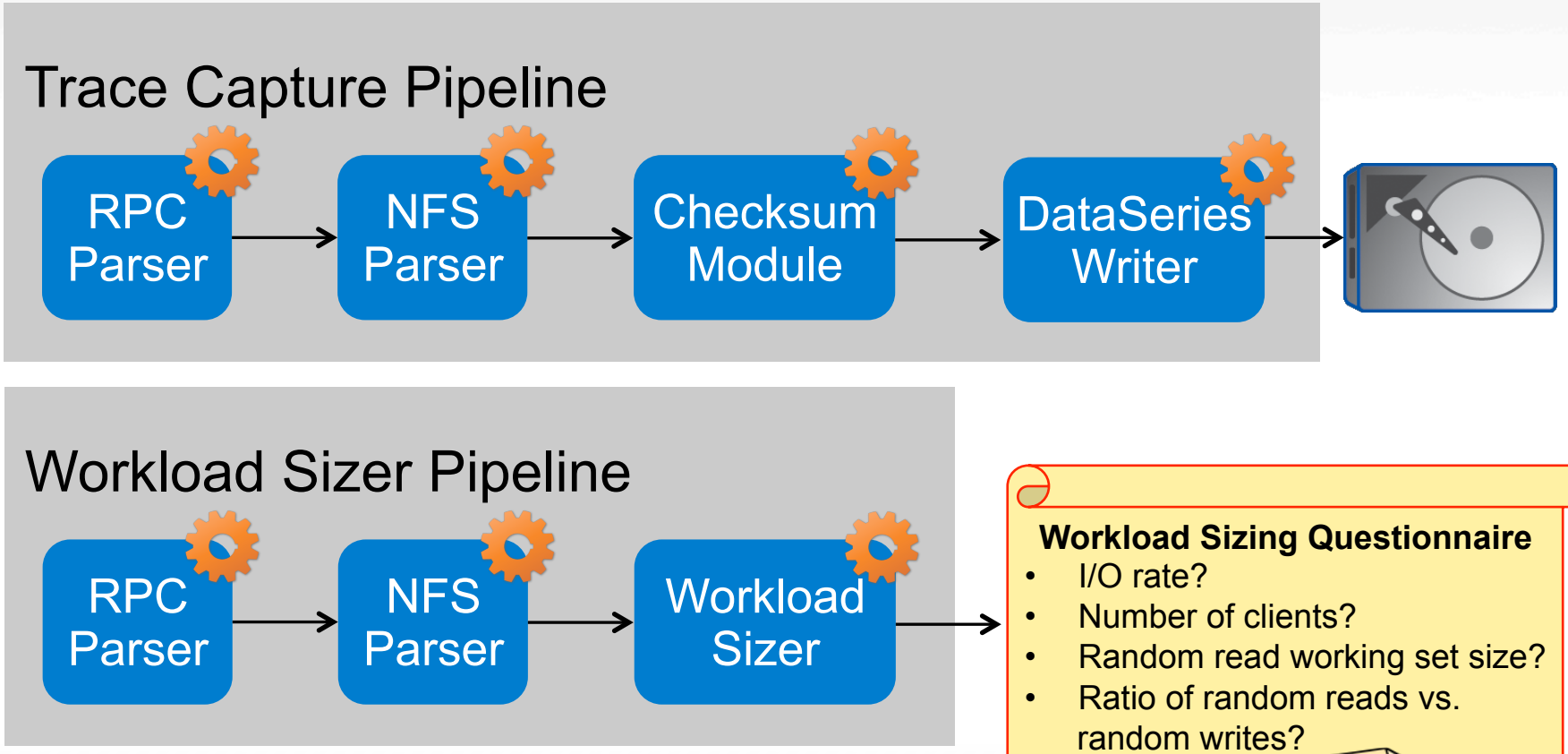
© 2015 NetApp, Inc. All rights reserved.

Chronicle Architecture



© 2015 NetApp, Inc. All rights reserved.

Chronicle Pipelines

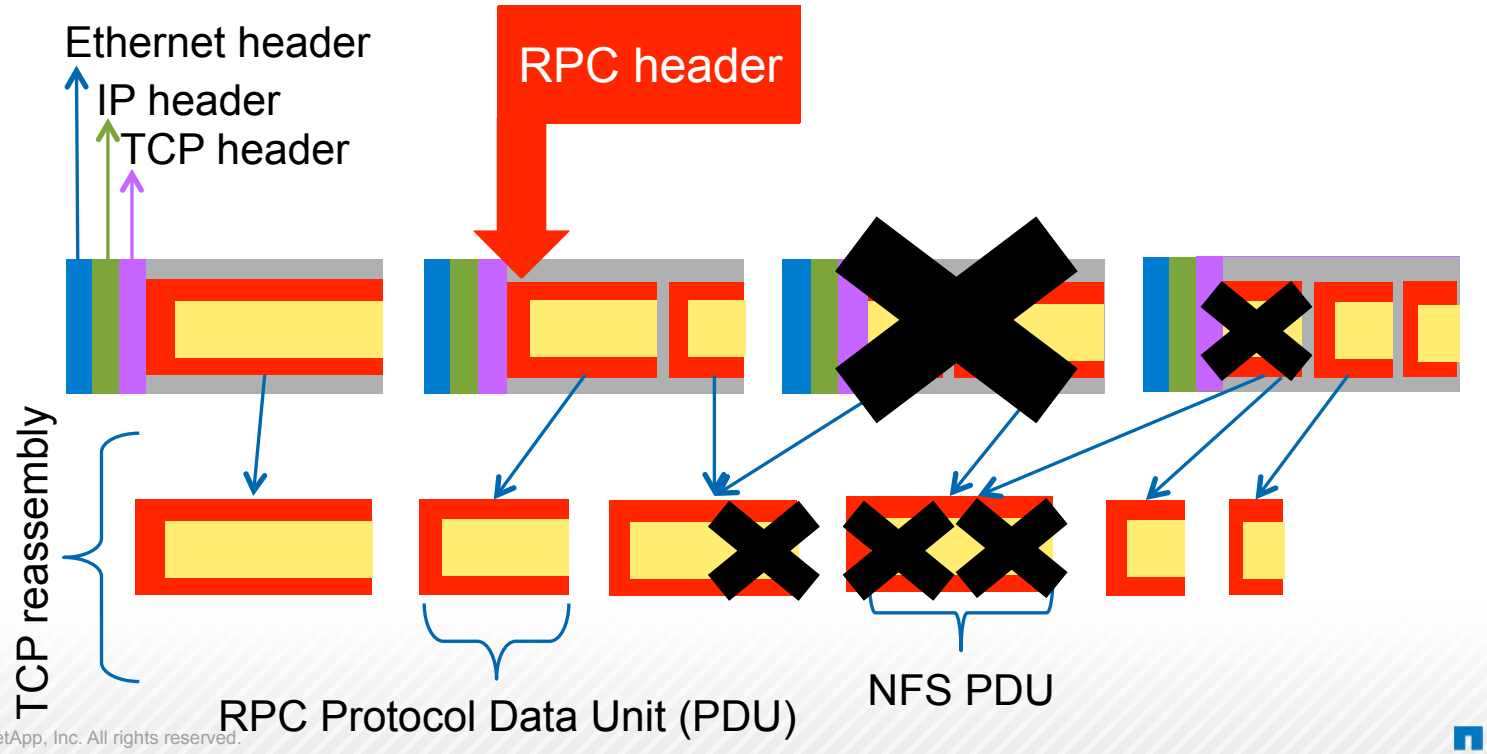


© 2015 NetApp, Inc. All rights reserved.

Chronicle Modules – RPC Parser

- Filtering of TCP and RPC traffic
- Detection and parsing of RPC header
- Matching RPC replies with the corresponding calls
- Reassembly of TCP segments
- Construction of RPC PDUs
- Two modes of operation

Slow mode



© 2015 NetApp, Inc. All rights reserved.
NetApp

More information on Chronicle

Please refer to the paper for more information on the following:

- The functions of each module in the pipeline
- The messages passed between modules
- Chronicle's novel, zero-copy application layer parsing approach
- A comprehensive comparison with other packet processing frameworks
- Insights from Chronicle that helped our customers

© 2015 NetApp, Inc. All rights reserved.

Evaluation Setup

- Chronicle server:
  - Two Intel Xeon E5-2690 2.90GHz CPUs (8 physical cores/16 logical cores per CPU)
  - 128GB of 1600MHz DDR3 DRAM memory (64GB per CPU)
  - Two dual-port Intel 82599EB 10GbE NICs
  - Ten 3TB SATA disks
  - 3.2.32 Linux kernel

$10,000

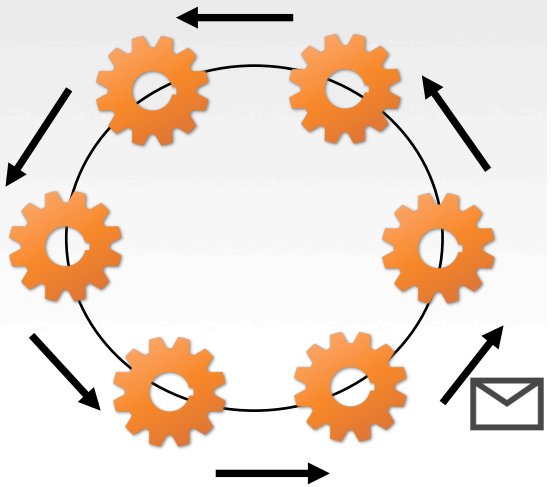


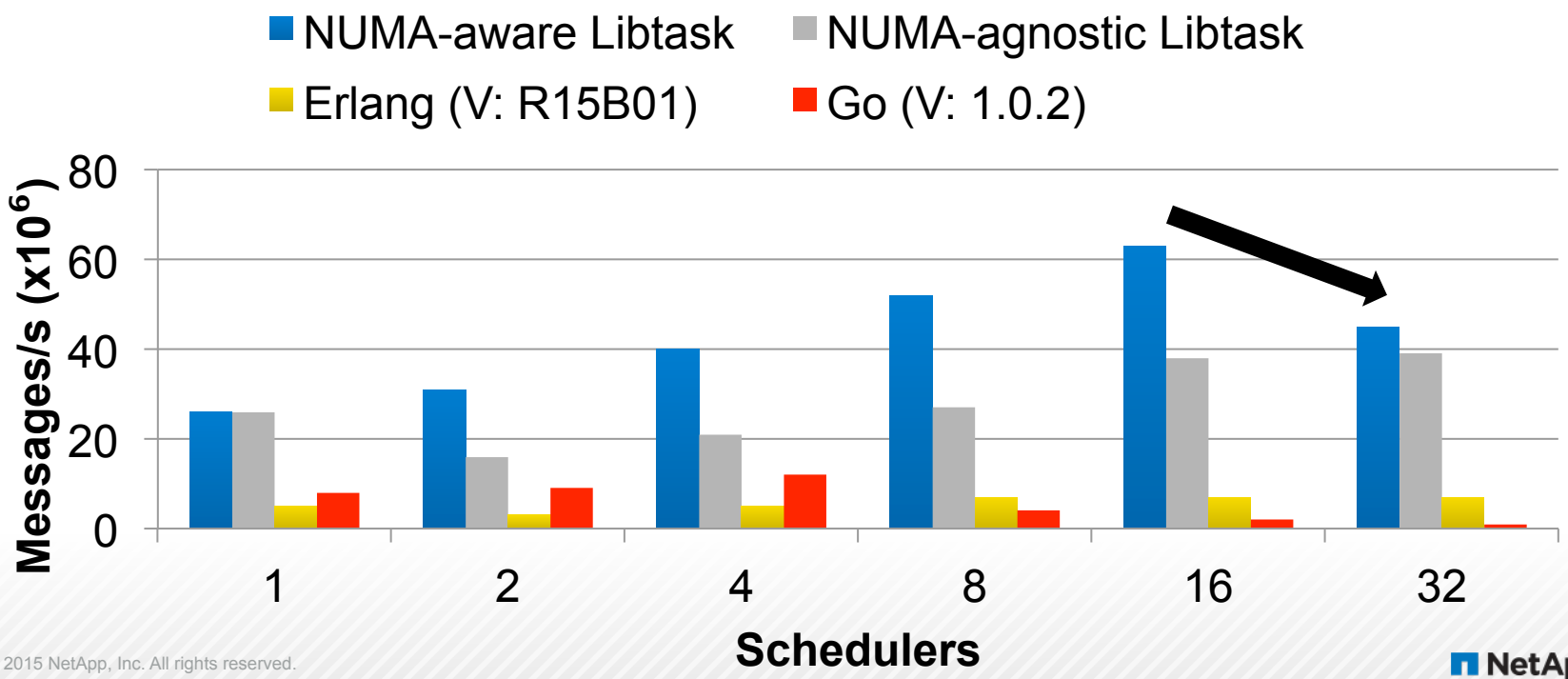
- A NetApp FAS6280 as the NFS server

© 2015 NetApp, Inc. All rights reserved.

Libtask Evaluation – Message Ring Benchmark

- 1000 Processes pass ~100M Messages in a ring
- 100 outstanding Messages at a given time
- Averages of 10 runs

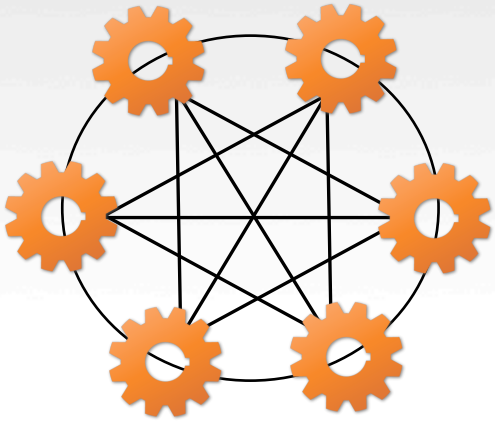


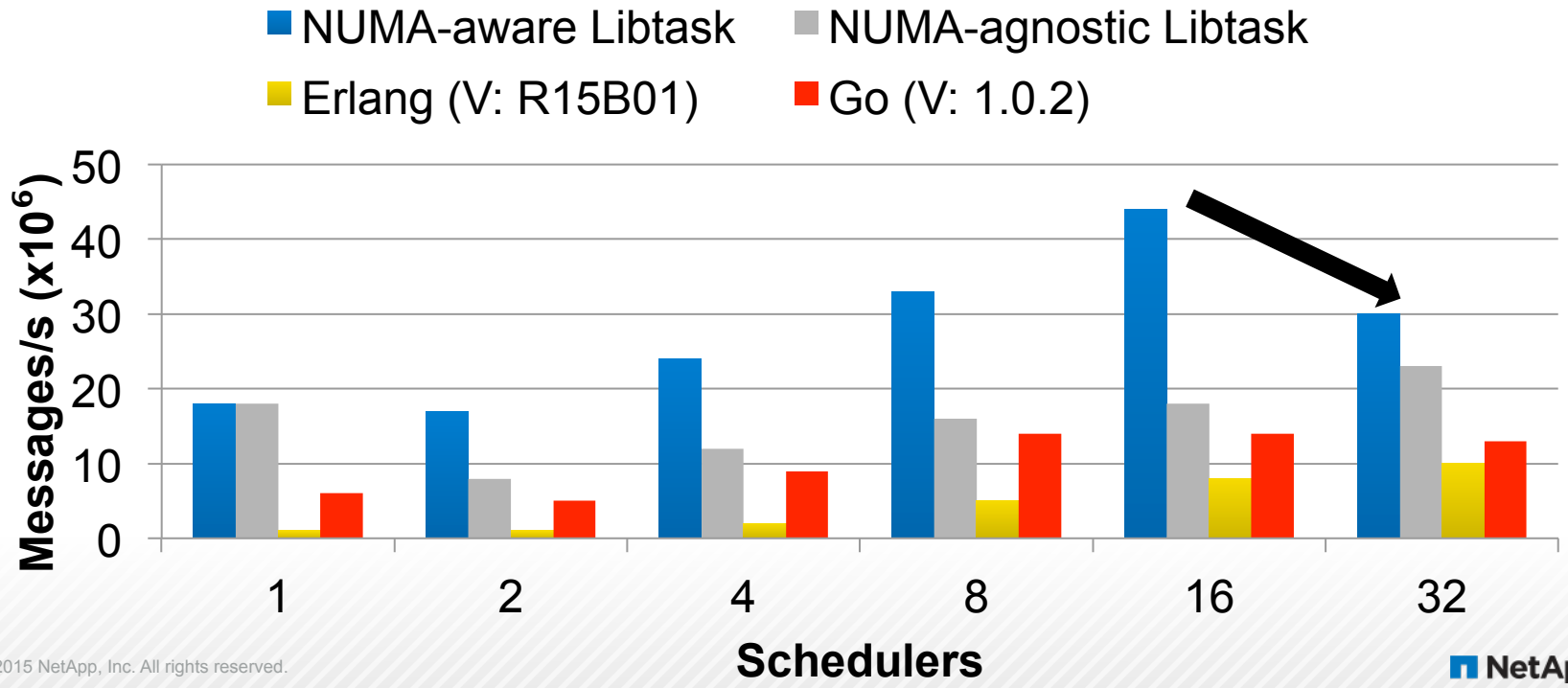


© 2015 NetApp, Inc. All rights reserved.
NetApp

Libtask Evaluation – All-to-All Benchmark

- 100 Processes pass ~100M Messages randomly
- 1000 outstanding Messages at a given time
- Averages of 10 runs

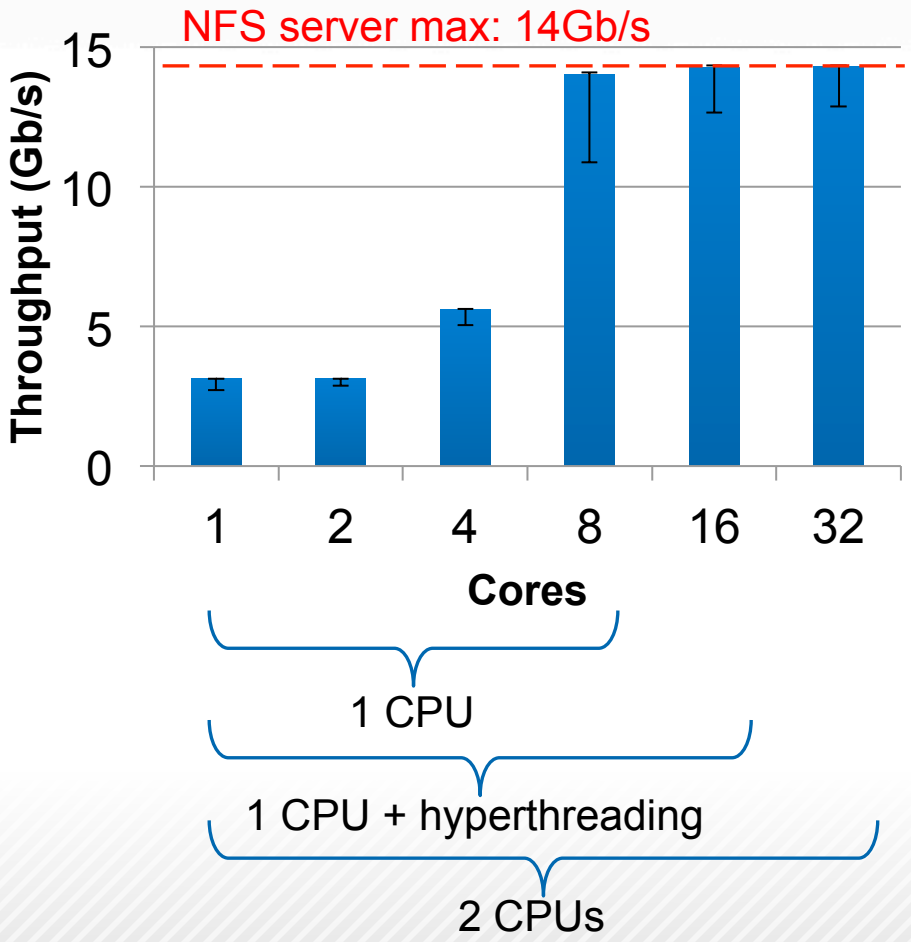


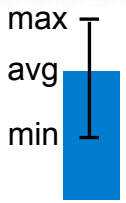


© 2015 NetApp, Inc. All rights reserved.
NetApp

Chronicle Evaluation – Maximum Sustained Throughput

- One client issuing 64KB sequential R/W ops across two 10Gb links using fio workload generator





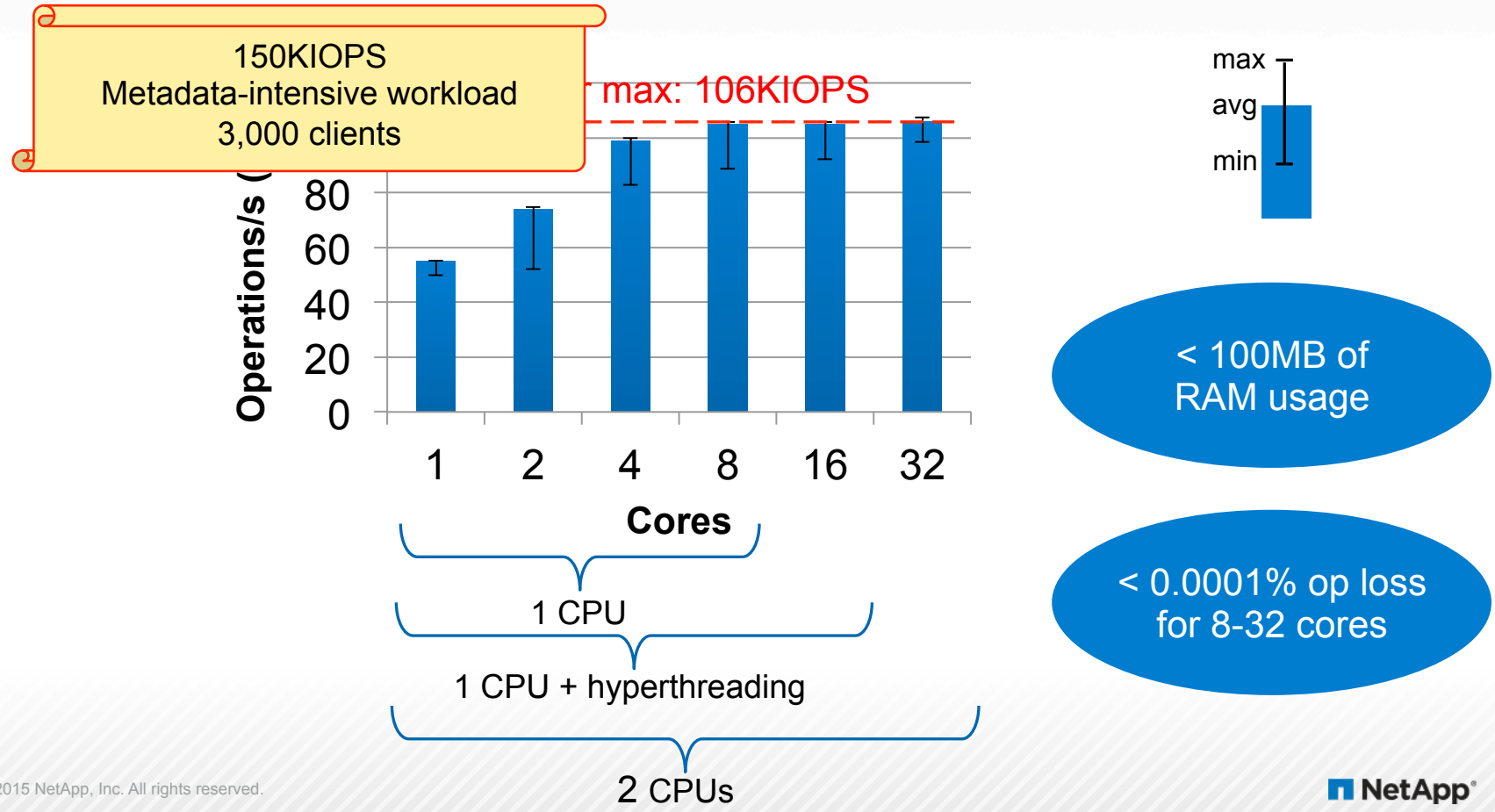
< 2.5GB of RAM usage

0.0002% op loss for 32 cores

NetApp

Chronicle Evaluation – Maximum Sustained IOPS

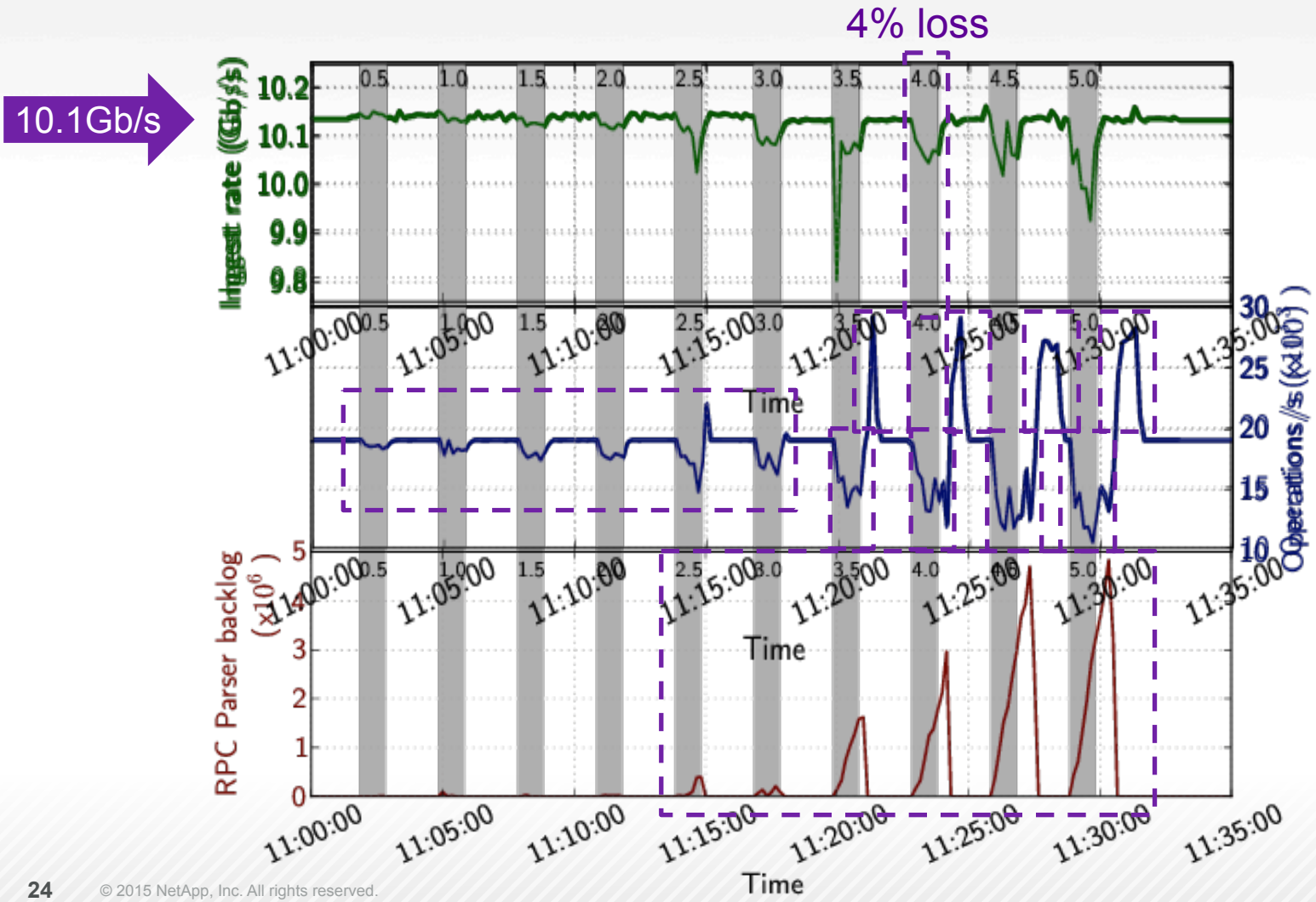
- One client issuing 1B sequential R/W ops across two 10Gb links using fio workload generator



© 2015 NetApp, Inc. All rights reserved.
NetApp

Chronicle Evaluation – Packet Loss

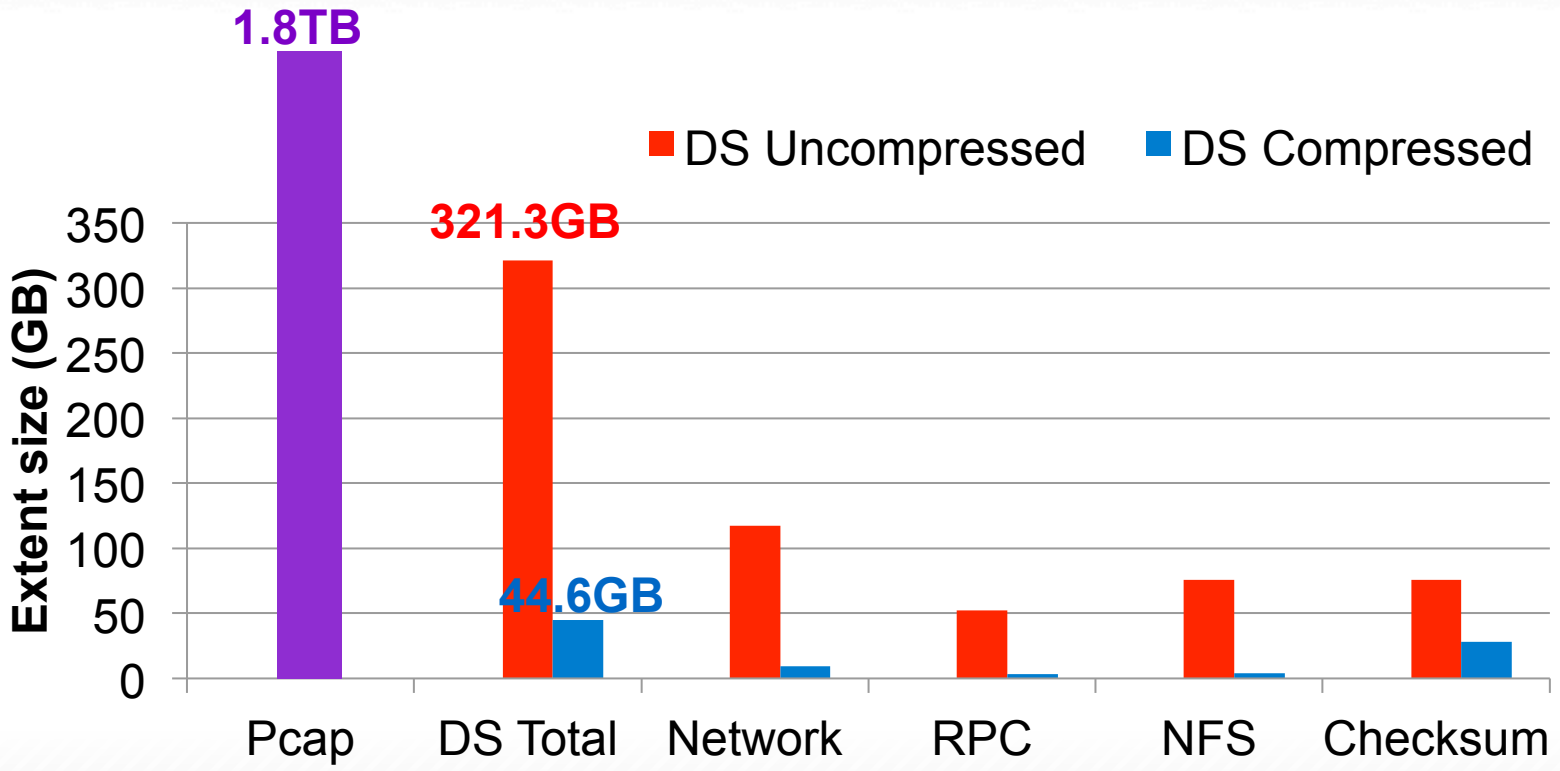
- Controlled experiment to study the impact of packet loss at 10Gb/s



© 2015 NetApp, Inc. All rights reserved.
NetApp

Chronicle Evaluation – Trace Storage Efficiency

- 7-hour-long trace of a production workload
- 40x reduction in trace size over pcap traces (1.8TB ➡ 44.6GB)!



© 2015 NetApp, Inc. All rights reserved.
NetApp

Conclusions

- Chronicle is an efficient framework for trace capture and real-time analysis of NFS workloads:
  - Operates at 14Gb/s using general-purpose CPUs, disks, and NICs
  - Based on actor model programming
  - Seamless scalability and a pluggable, pipelined
  - Programmer-friendly API
  - CPU-intensive operations like stateful parsing, pattern matching, data checksumming, and compression
  - Extensible to support other network storage protocols (e.g., SMB/CIFS, iSCSI, RESTful key-value store protocols)

© 2015 NetApp, Inc. All rights reserved.

Questions?

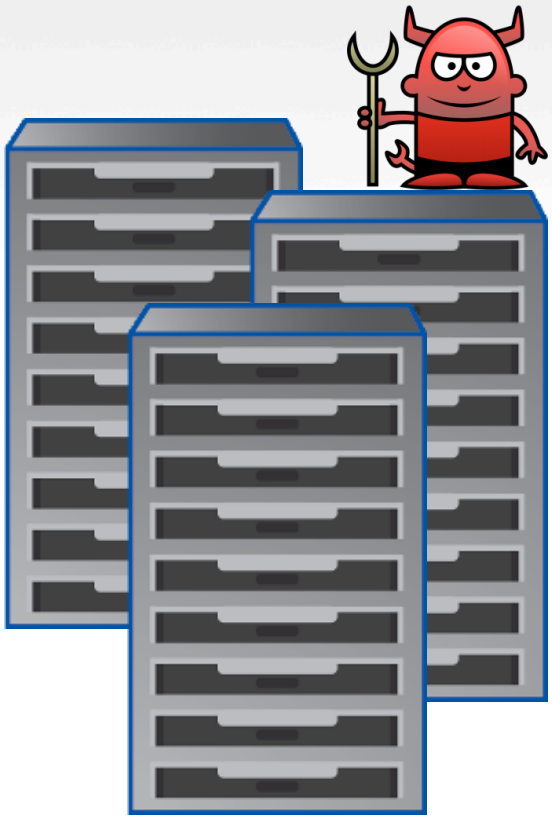
Thank You!

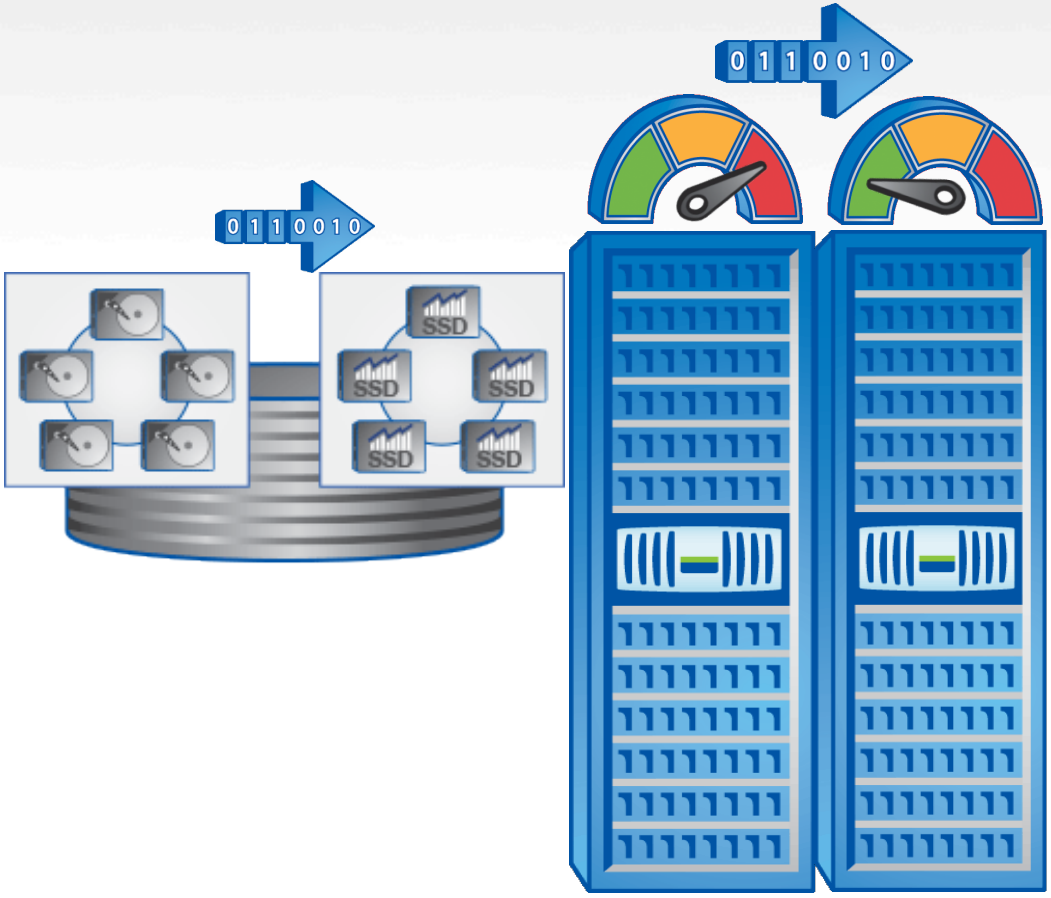
Chronicle's source code is available under an academic, non-commercial license:

https://github.com/NTAP/chronicle

© 2015 NetApp, Inc. All rights reserved.

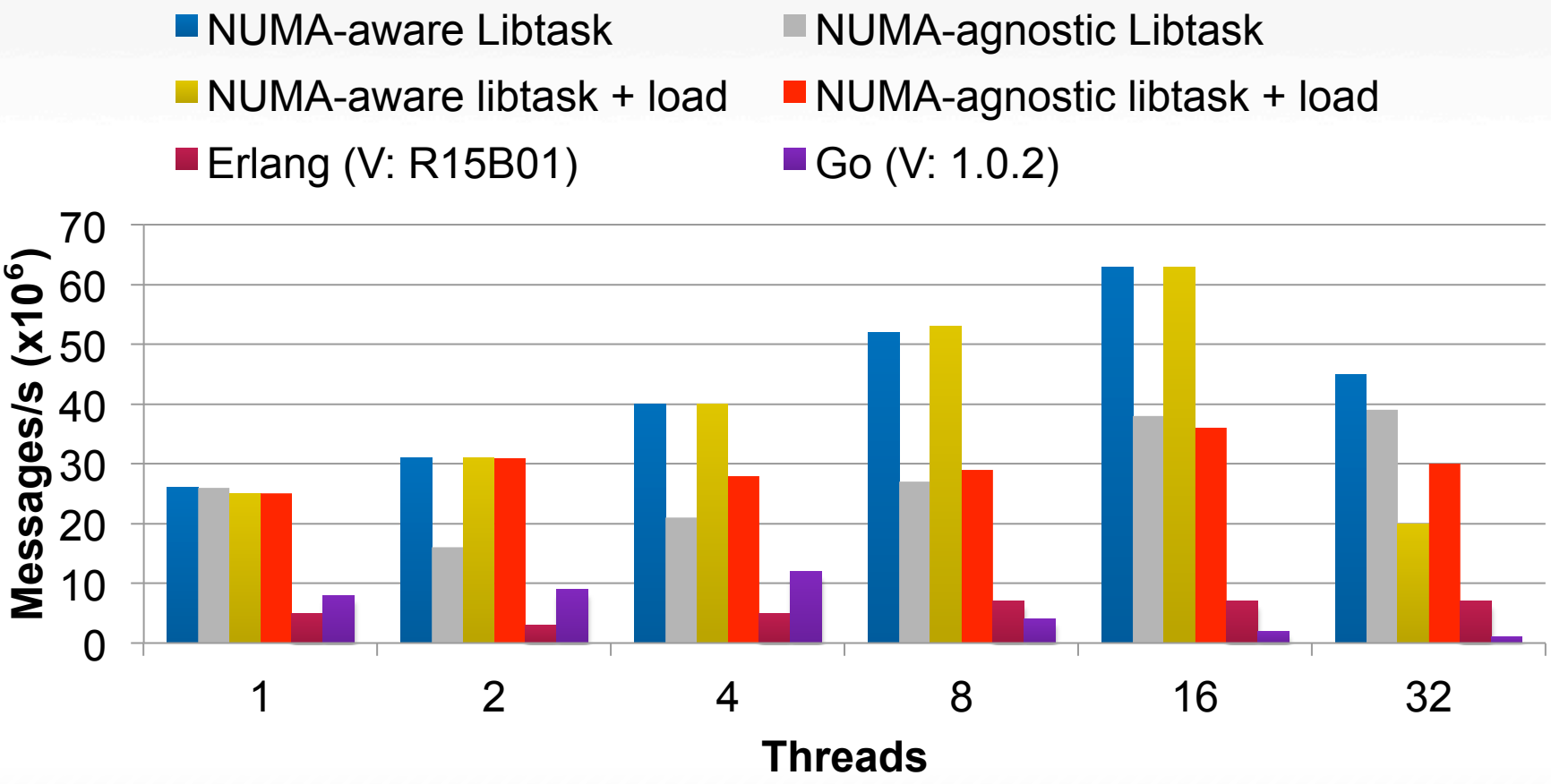
Use Case 3: Data-Driven Management





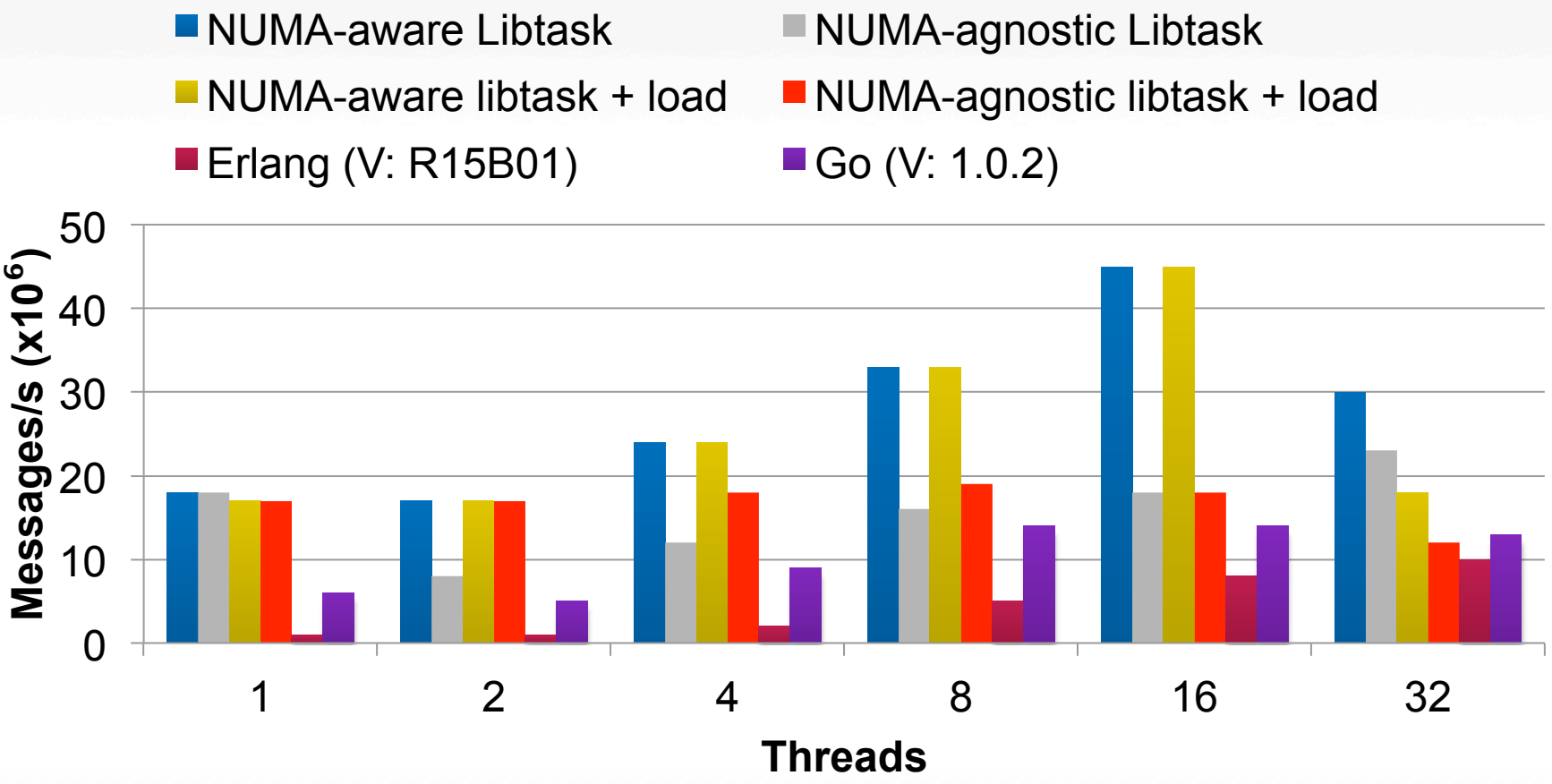
© 2015 NetApp, Inc. All rights reserved.
NetApp

Libtask Evaluation – Message Ring Benchmark



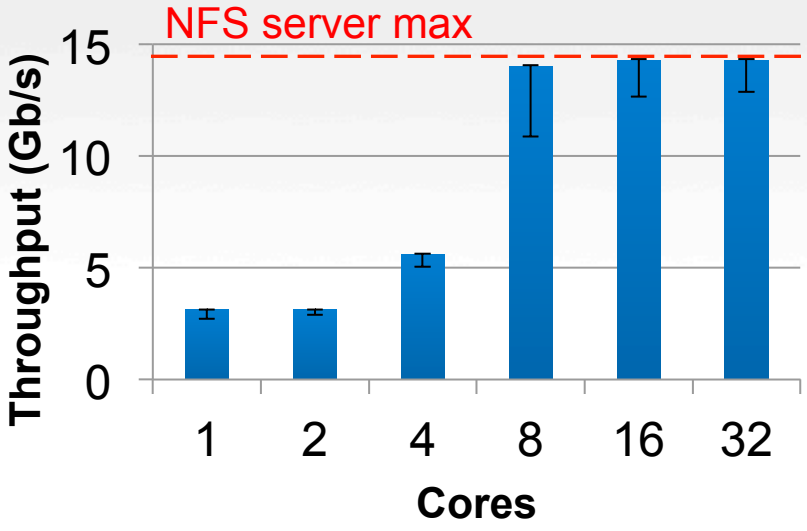
© 2015 NetApp, Inc. All rights reserved.

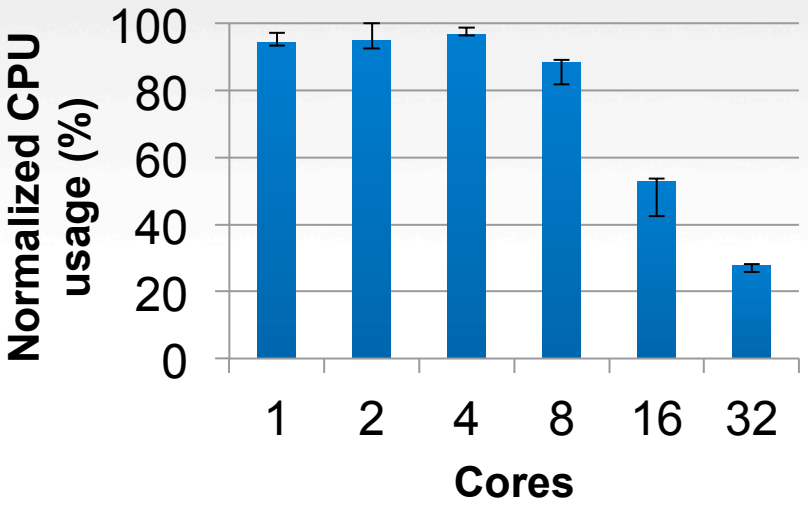
Libtask Evaluation – All-to-All Benchmark

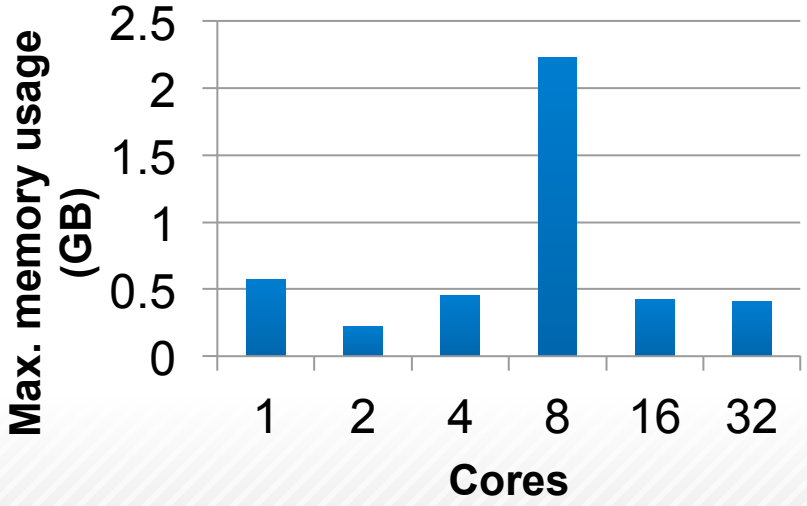


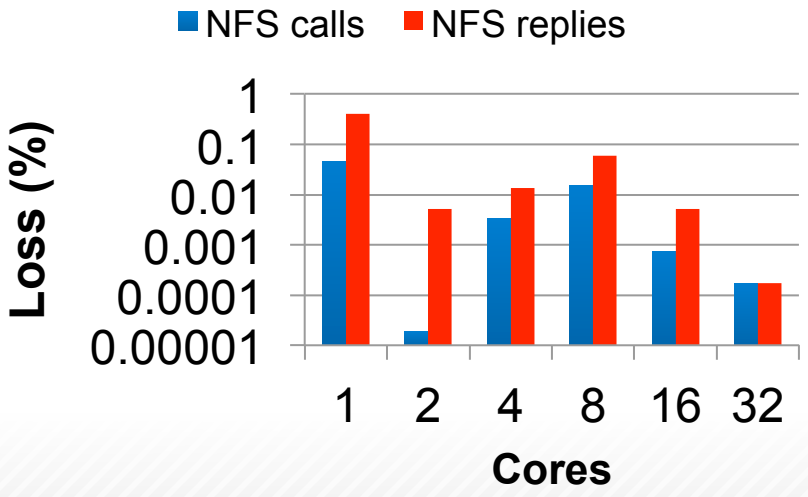
© 2015 NetApp, Inc. All rights reserved.

Chronicle Evaluation – Maximum Sustained Throughput



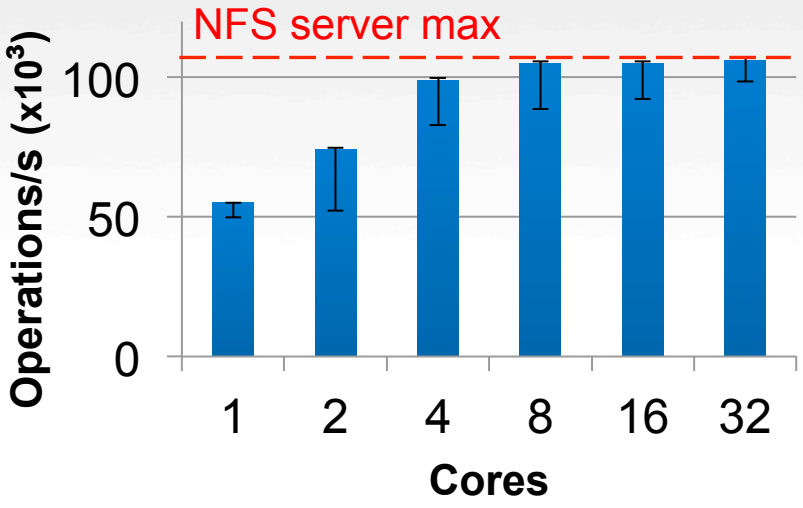


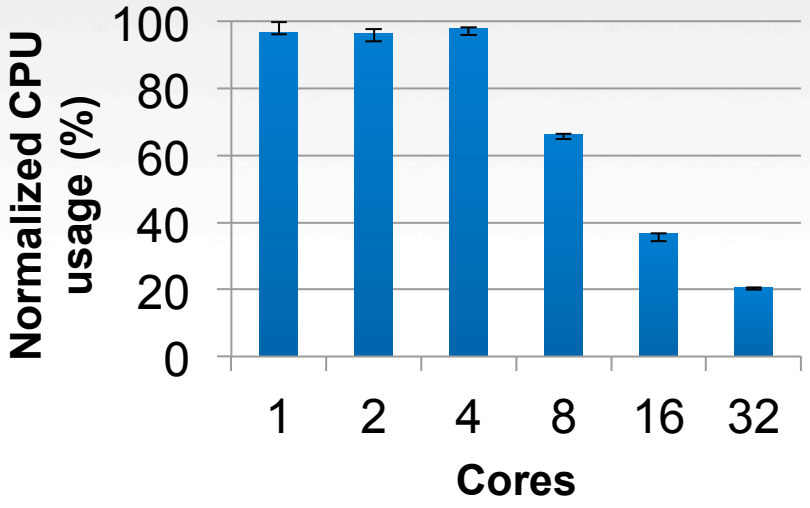


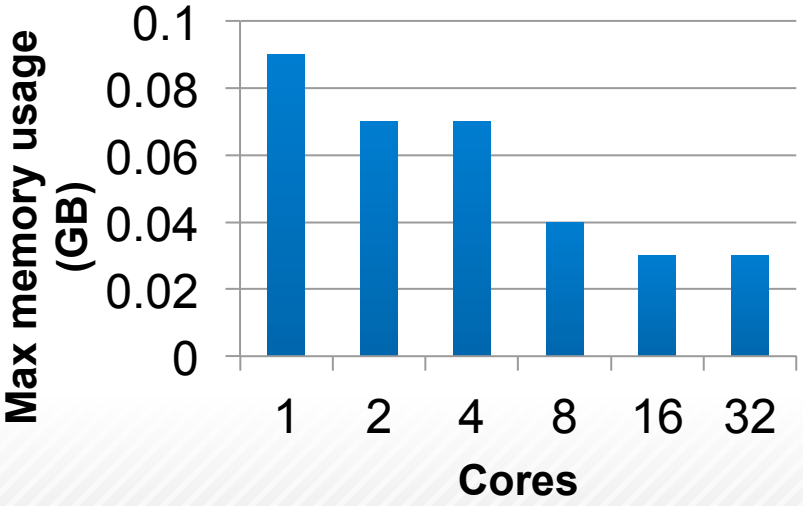


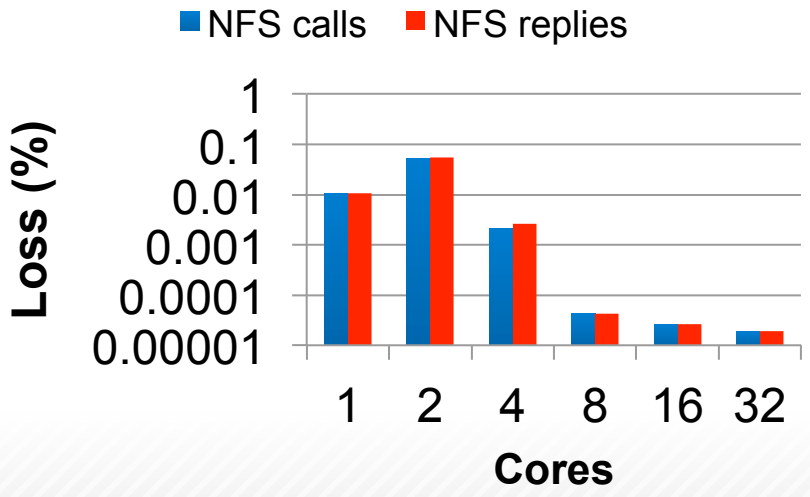
© 2015 NetApp, Inc. All rights reserved.

Chronicle Evaluation – Maximum Sustained IOPS









© 2015 NetApp, Inc. All rights reserved.