SDC

STORAGE DEVELOPER CONFERENCE

SNIA SANTA CLARA, 2015

Beyond Consistent Hashing and TCP:
Vastly Scalable Load Balanced Storage Clustering

Alex Aizman, Caitlin Bestler

Nexenta Systems

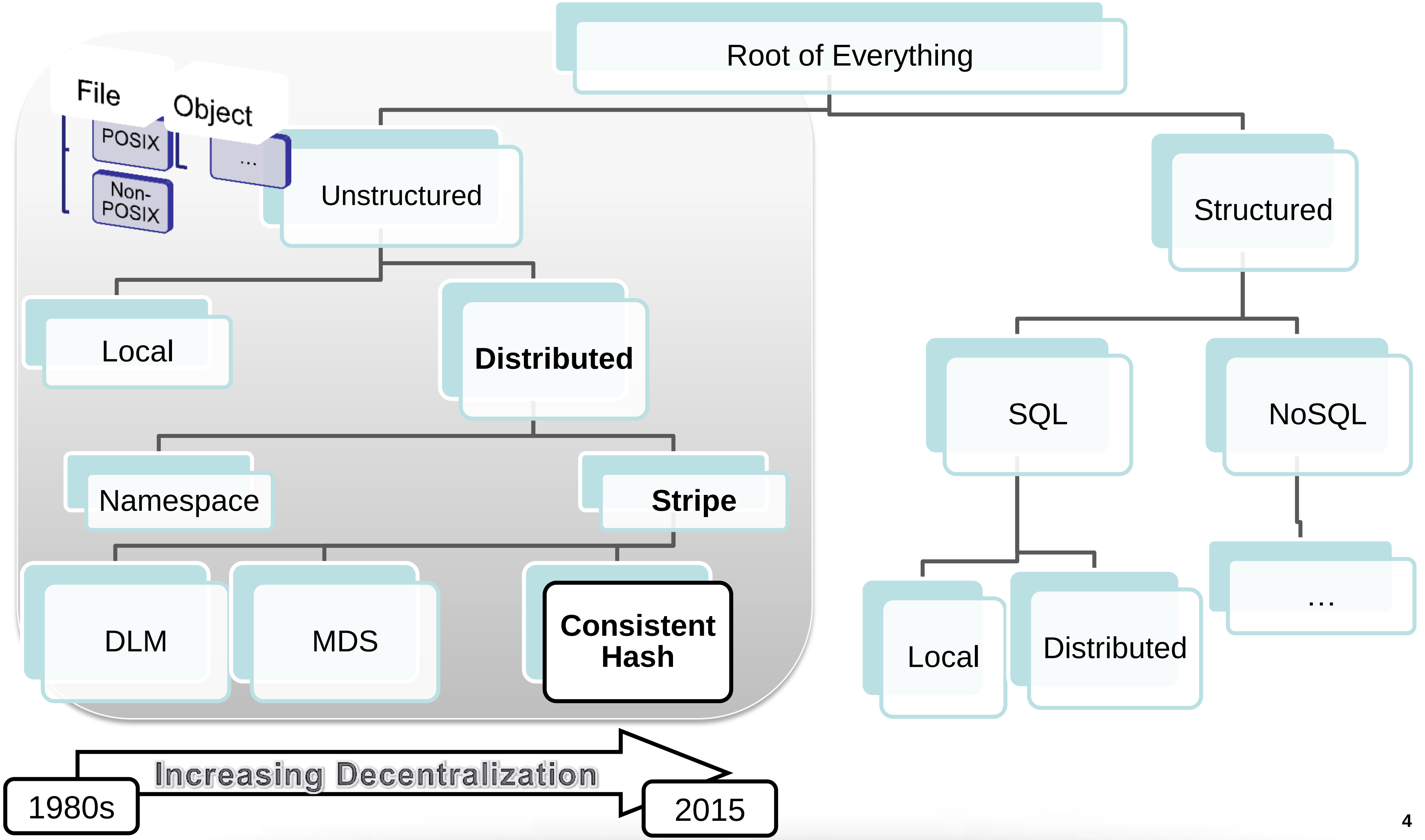
The Scale of Disruption

	Conventional	New
Data Distribution Mechanism	Consistent Hashing (CH)	Group Hashing (NGH)
Storage Transport	Unicast (TCP)	Multicast (Replicast™)

- This presentation:
  - Existing technology vs. massive scale
  - There are in fact better alternatives..

Distribution & Replication Mechanism: Data & Metadata

TOE (data storage)



Increasing Decentralization

Increasing Decentralization

1980s

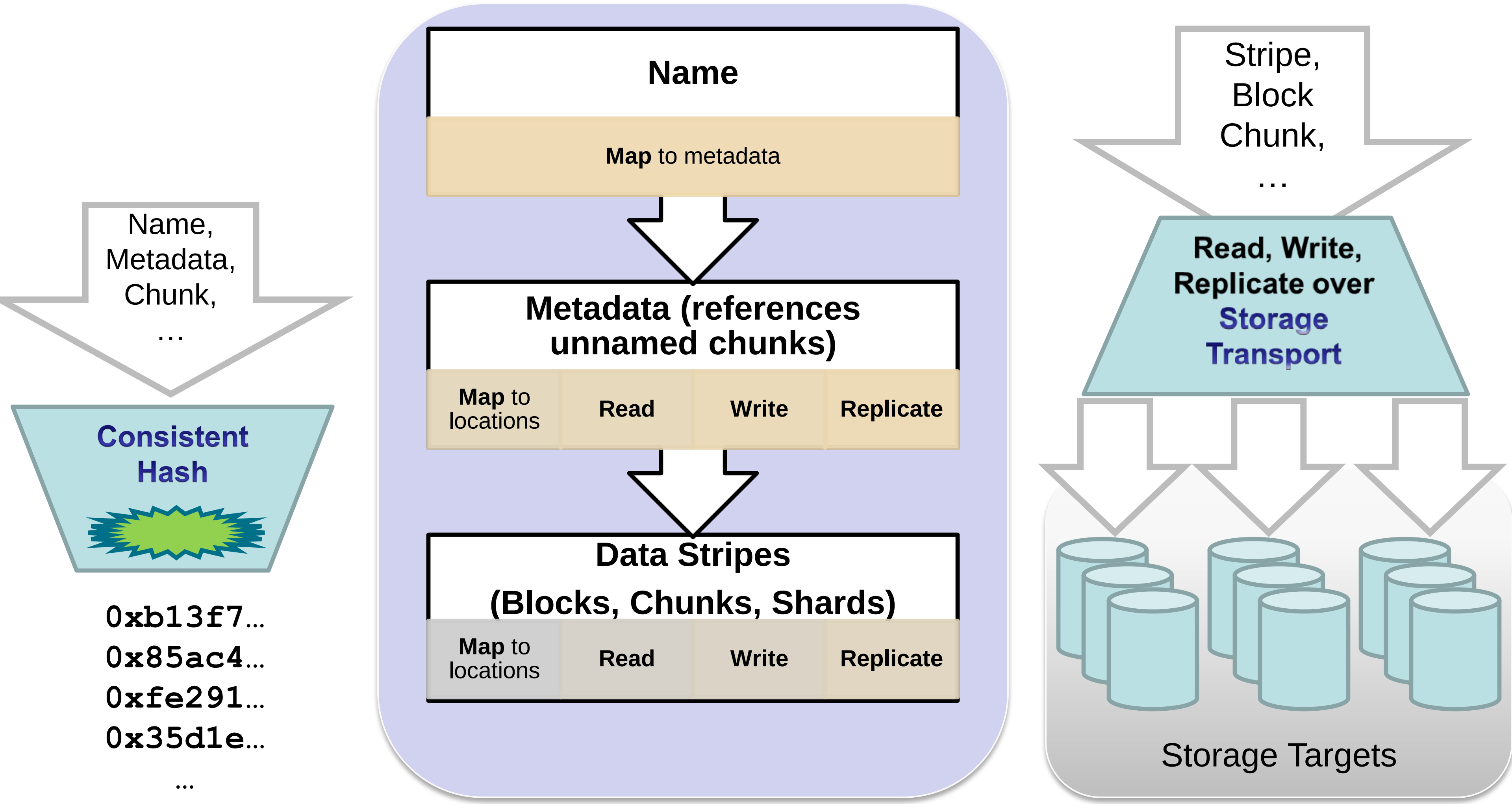
1. Allocate block or stripe
2. Record the location in the parent's table
3. Write…

2010s

1. Map name or chunk to target
2. Let the target take care of the rest

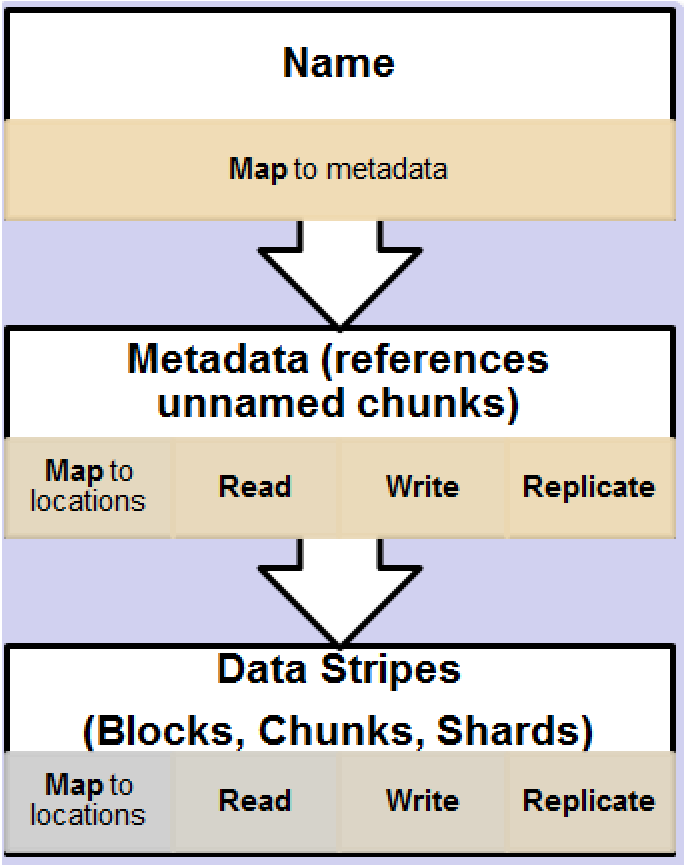
Increasing Scale

I/O pipeline(*): unstructured, distributed and striped



Quick Recap

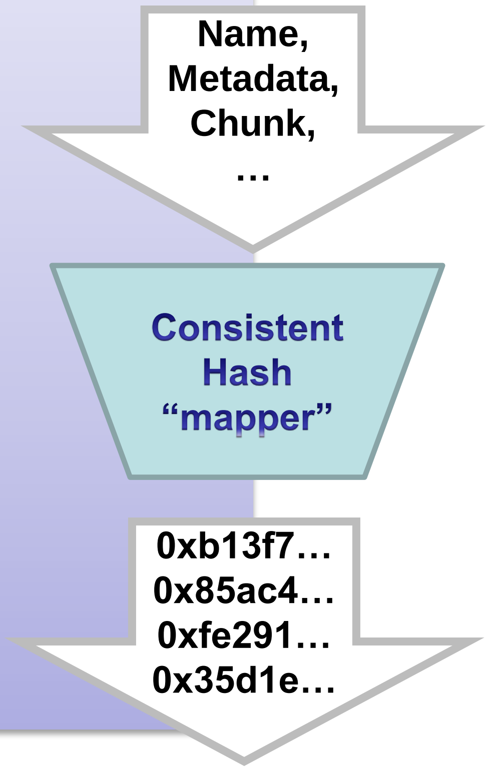
	Conventional	New
Data Distribution Mechanism	Consistent Hashing	Group Hashing (NGH)
Storage Transport	TCP	Multicast (Replicast™)



- Next:
  - CH vs. NGH
  - TCP/Unicast vs. Replicast
  - Simulation: model, results, charts

Consistent Hashing

- Openstack's Object Storage Service Swift
- Amazon's storage system Dynamo
- Apache Cassandra
- Voldemort
- Riak
- GlusterFS
- Ceph*
- …



❖ Trademarks listed here and elsewhere in this presentation are property of their respective owners

Ceph vs. Consistent Hashing

- Input:
  - CRUSH MAP aka Cluster Map
    - Placement Groups
      - OSDs aka devices
      - OSD weights
  - Rules: data redundancy
  - Replica
- Output:
  - Pseudo-random uniform hash across failure domains – binomial(n, p)
- Problems:
  - Same as CH – see next

- **Simplified CRUSH: HASH(node, replica)**
- **Rendezvous hashing: HASH(node, replica)**
- **Consistent hashing: HASH(replica)**

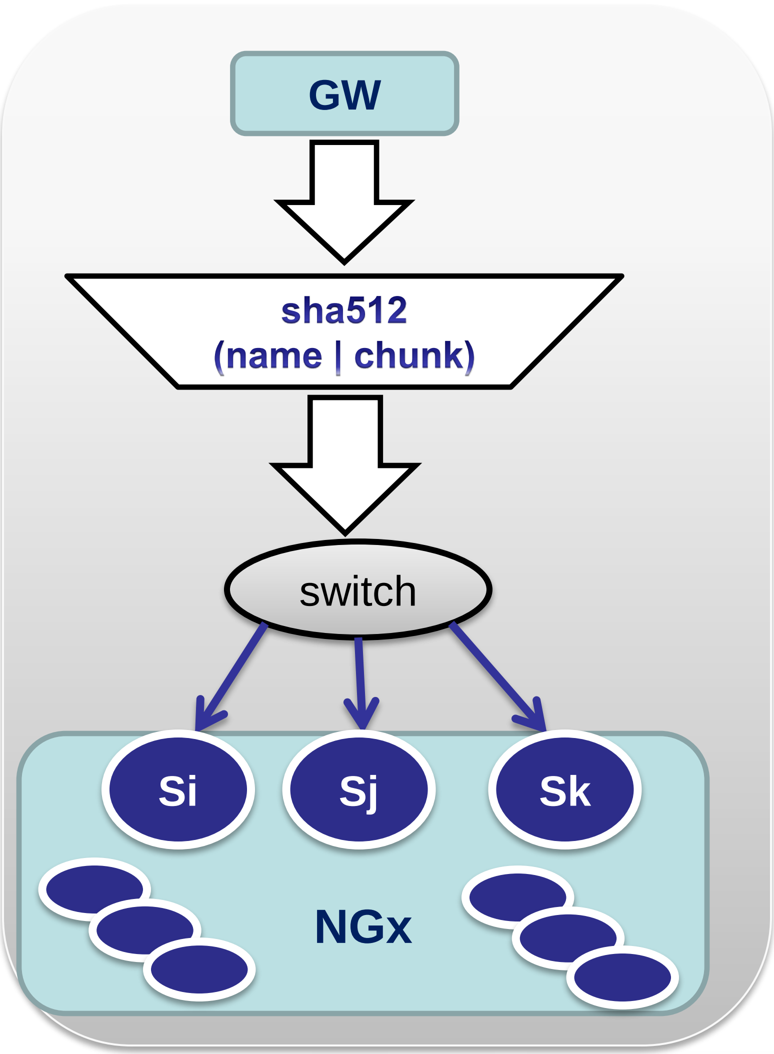
Consistent Hashing: the good, the bad…

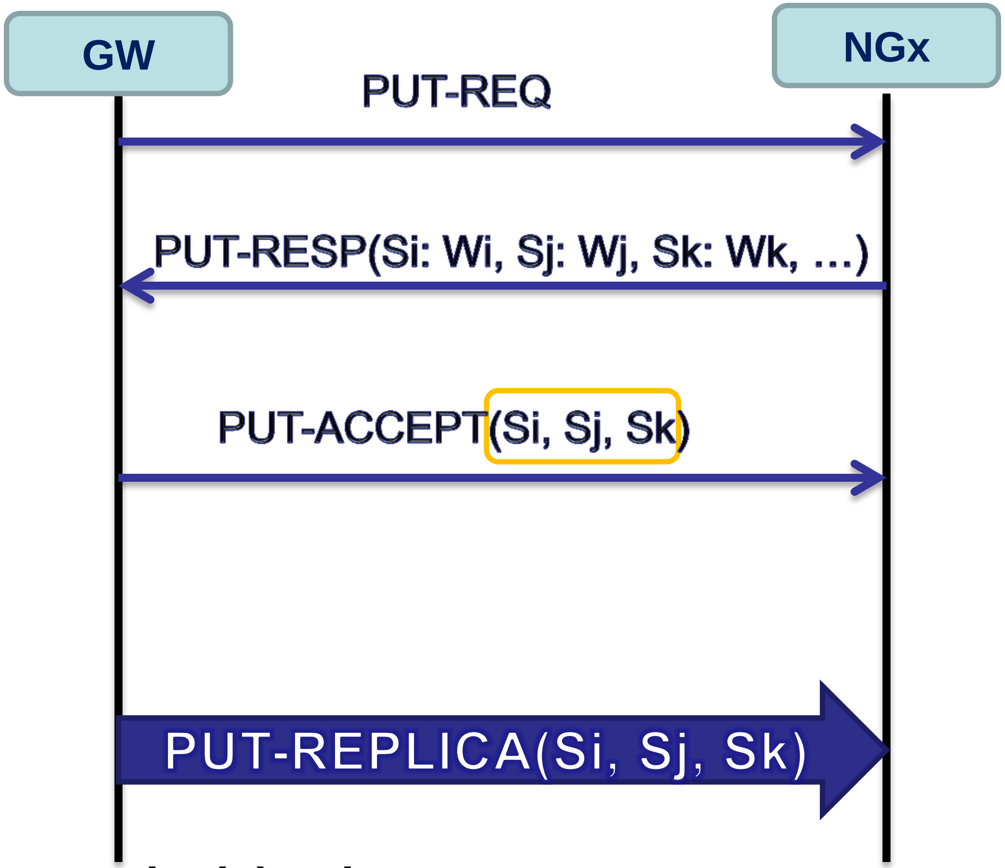
- The good
  - better than hash(chunk) % n-servers
  - even better when used with virtual nodes (partitions)
- The bad
  - Balls-into-Bins: load ➔ $\log(n)$: congestion “ripples”
  - Subject to binomial spikes

$$\frac{\log n}{\log \log n} \cdot (1 + o(1))$$

- The ugly
  - Hashing determinism vs. Load balancing: either/OR
  - Worse for larger clusters
  - Much worse at a high utilization
  - Is **too** consistent..

NGH: How it works





- Min(*) size NG = 6
- Simulating 9 targets per NG
- Must select 3 best bids

SDC 15

Why NGH

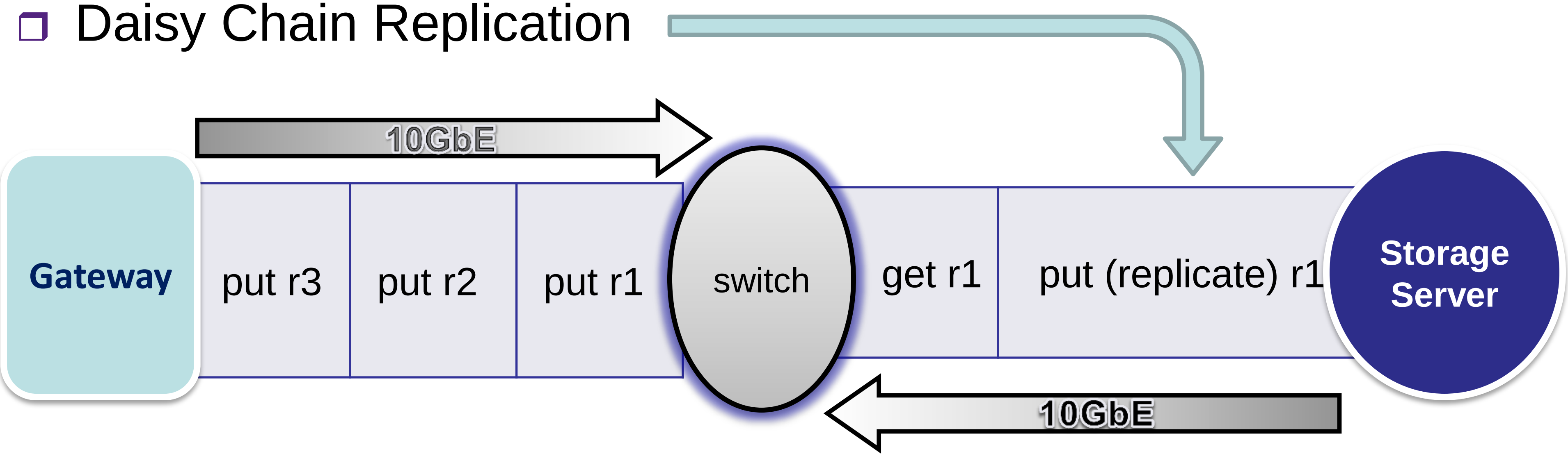
- Admit: we simply don't know enough about the state on the edge
- Accept it.
- Main principles and motivations:
  - Serial transmit: one chunk at a time
  - Per-disk dedicated thread: one chunk at a time
  - Edge-based load balancing as part of the datapath
  - Edge-based reservation of link bandwidth:

- Storage Target ➔ to Gateway:
  - begin = NOW + F(queue-depth, cache-size, link bw)
  - end = begin + G(chunk-size)

Storage Transport:
Data & Control

Unicast vs. Persistent Redundancy

- Typical: put(…) **blocks** until 3 copies get persistently stored on 3 storage servers (variations: RAIN, EE, 2 cached, etc.)
- IO pipeline: redundancy – first, completion – second
- Daisy Chain Replication



- Sustained get() rate G
- + Sustained single-replica put() rate P
- + Daisy chain

- New get() rate (%): G * 100/ (G + P)

TCP: the good, the bad…

- The good:
  - Predominant, totally integrated and reliable
- The bad:
  - Was never designed for scale-out storage clusters
- The ugly:
  - Unicast – previous slide
  - TCP incast – "*is a catastrophic TCP throughput collapse*…"
    - TCP "outcast" – converged and hyper-converged clusters
  - Time to converge >> RTO
- However:
  - DCB helps
  - Custom/proprietary TCP kernels will help as well..

TCP Simulation

	Conventional	New
Data Distribution Mechanism	Consistent Hashing (CH)	Group Hashing (NGH)
Storage Transport	Unicast (TCP)	Multicast (Replicast™)

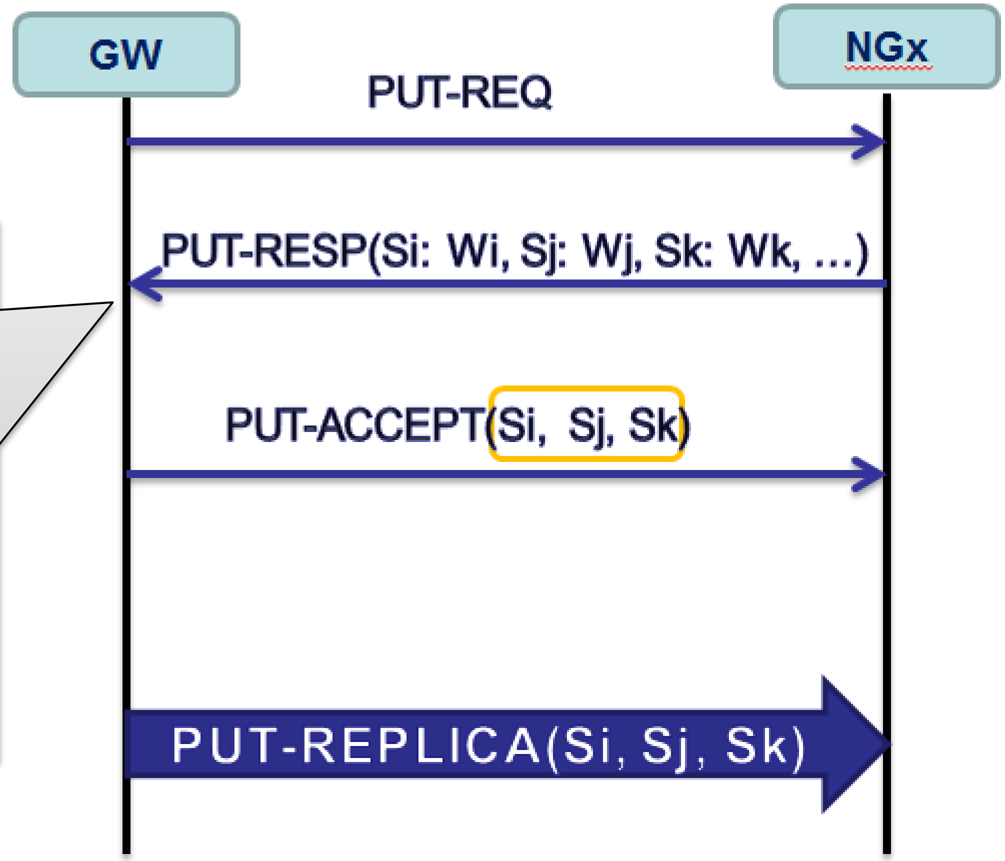
- Each chunk put: 3 successive TCP transmissions
- ACK every received replica
  - ACK delivery time = chunk reception time + ½ * RTT
- Future-perfect TCP Congestion Protocol: Instant Convergence
  - N flows to one target, each instantly adjusting to use 1/Nth of the 10GE bandwidth
- TCP receive window and Tx buffer sizes unlimited
- Zero per-chunk overhead

- *Simulating TCP as a quintessential perfect Unicast*

Replicast Simulation

The (simulated) penalty is on Replicast:

- +RTT per-chunk overhead
- Note: TCP is benchmarked with (0) zero overhead
- Replicast still wins..

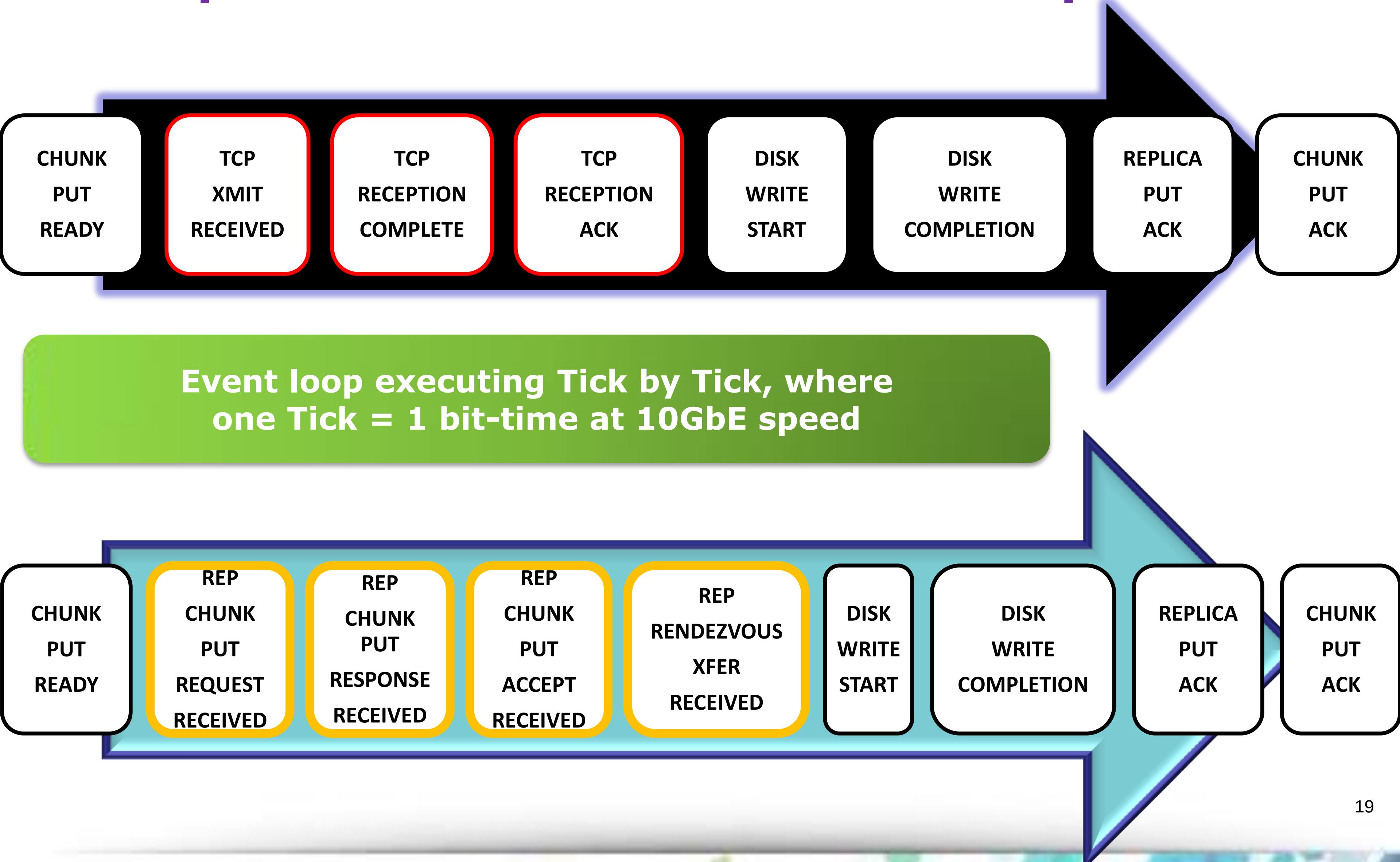


Tire Kicking

- https://github.com/Nexenta/nedge-sim
  - ./build-ndebug
  - ./do_benchmarks.sh
- Example:
  - `./test ngs 128 chunk_size 128 gateways 512 duration 100`
- Extensive comments inside
- Detailed log tracking of each transaction
- Logs for the cited runs can be reproduced at ff5fbda

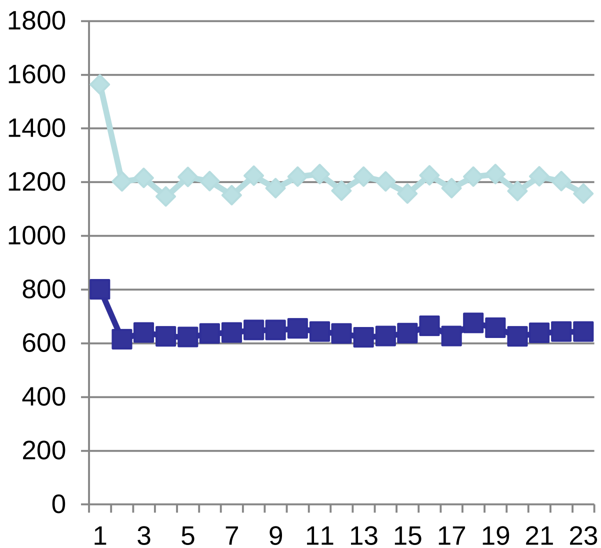
Default		Comment
#define CLUSTER_TRIP_TIME	10000	// approximately 1us; RTT = 2us
#define N_REPLICAS	3	// replicas of each stored chunk
#define MBS_SEC_PER_TARGET_DRIVE	400	// 400MB/s (eMLC)

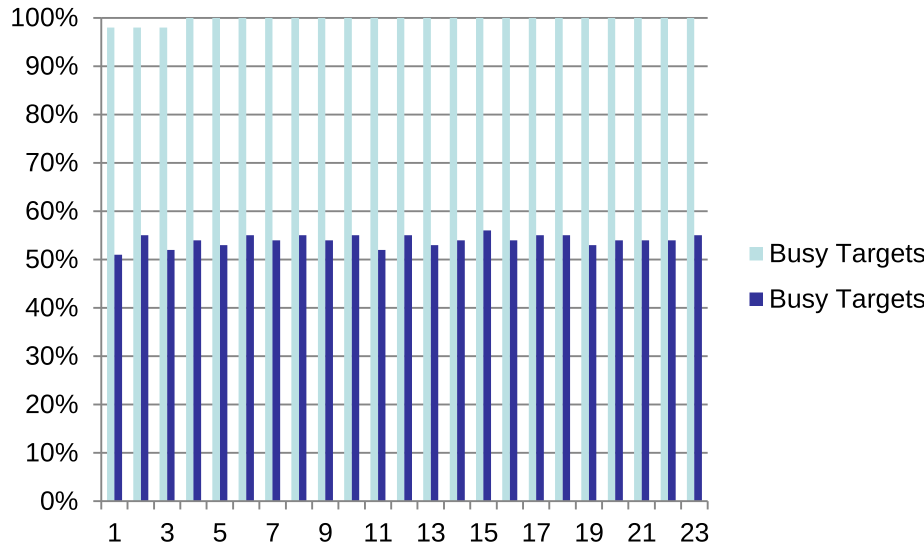
I/O Pipelines: CH/TCP and NGH/Replicast



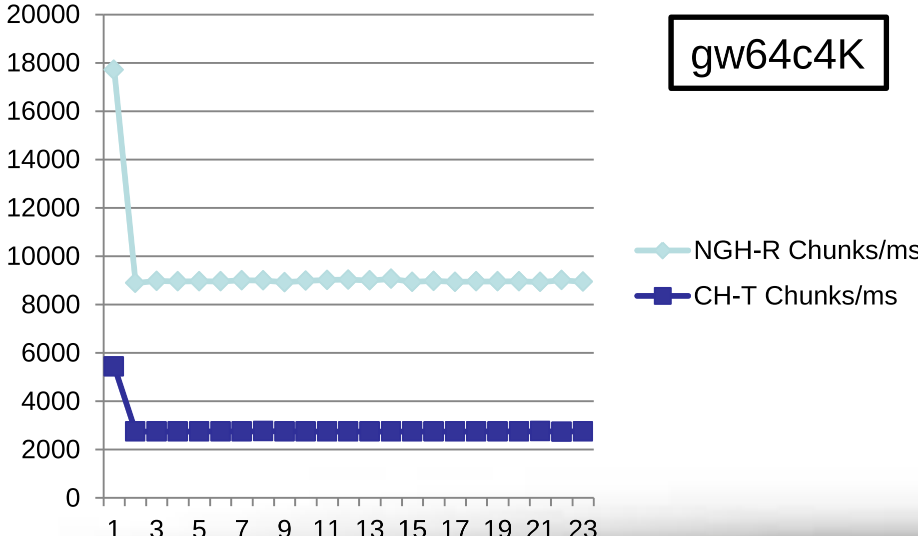
Simulation Results

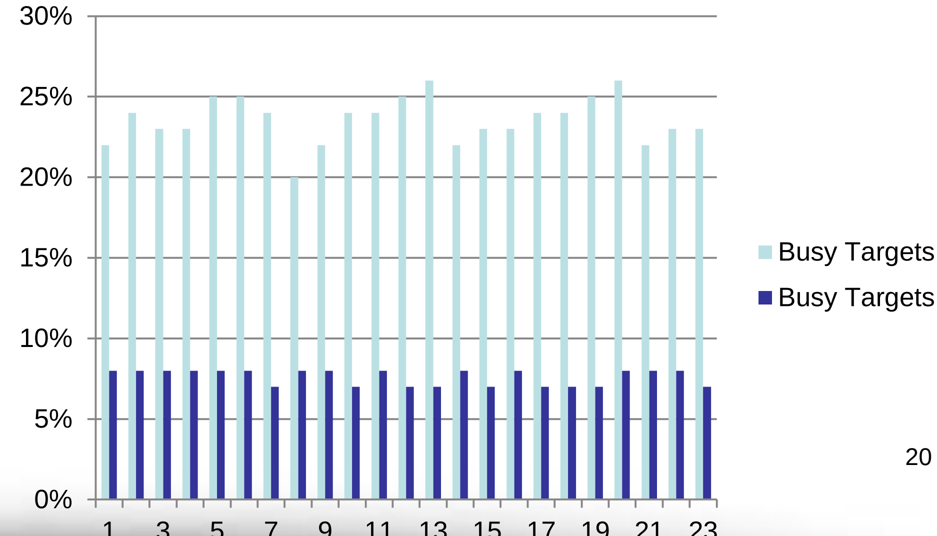
gw256c128K





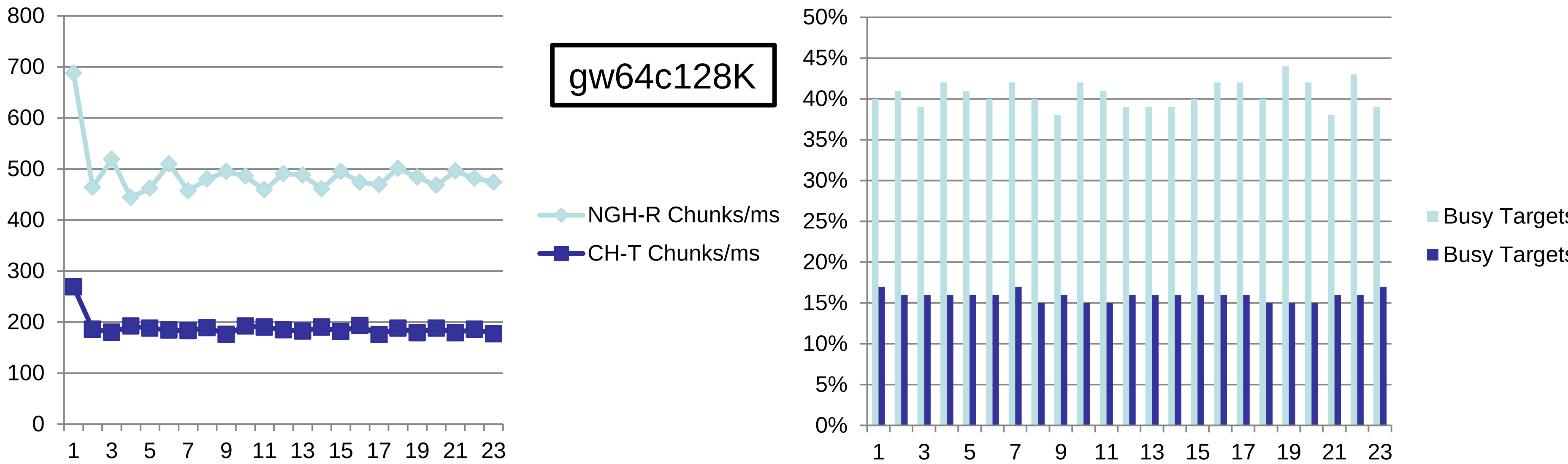
gw64c4K



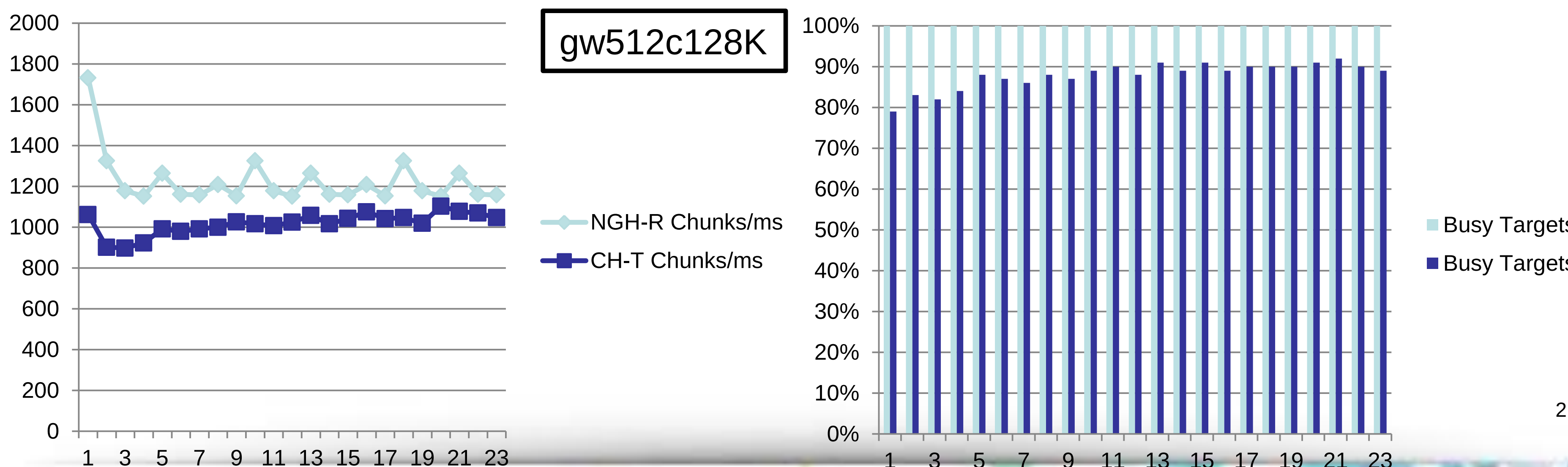


Simulation Results (cont-d)

gw64c128K

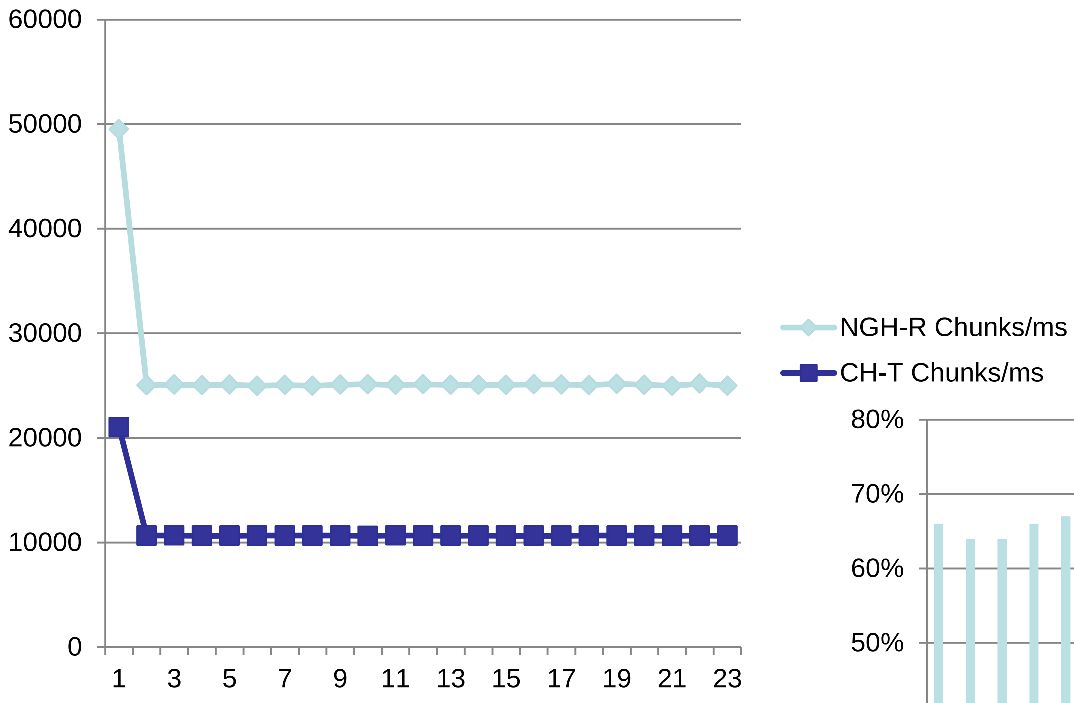


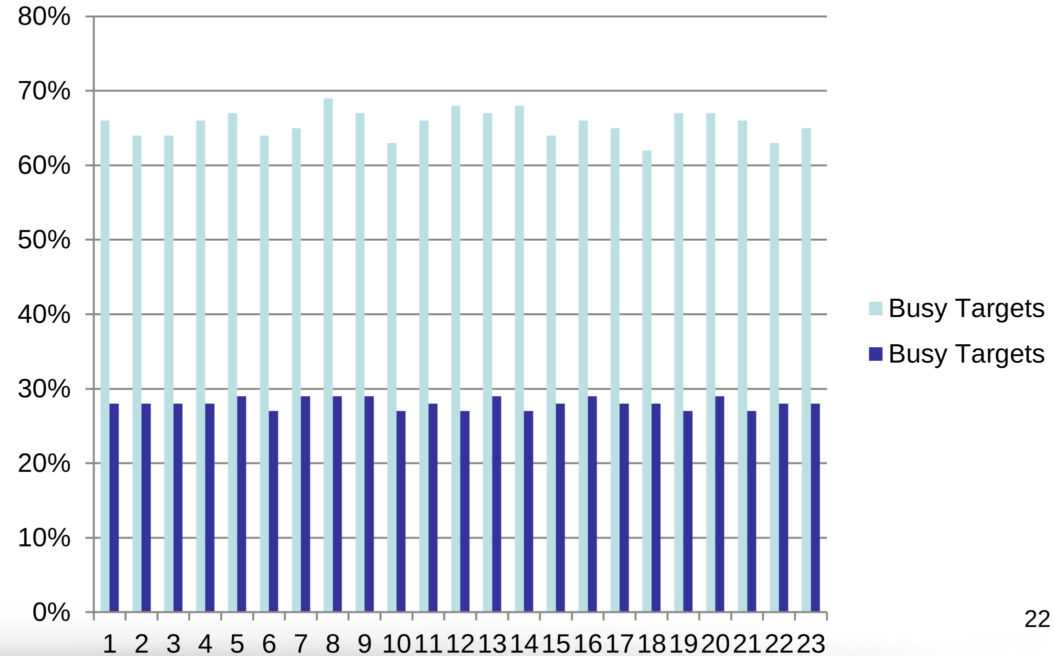
gw512c128K



Simulation Results (last)

gw256c4K

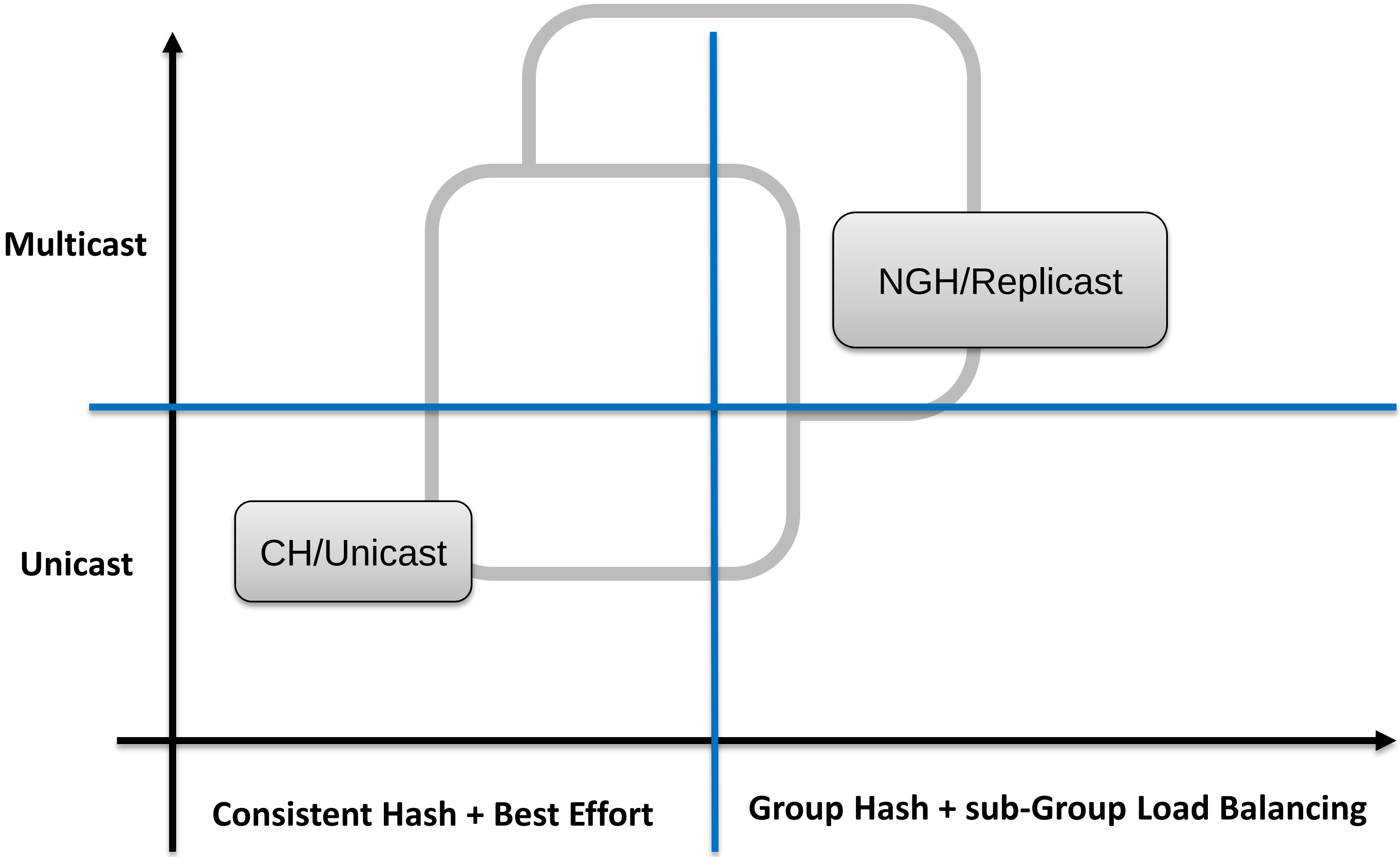




Conclusions

- Distributed storage clusters require technology that can simultaneously optimize:
  - available IOPS (budget) of the backend
  - storage capacity
  - network utilization
- To address **all** of the above, the mechanism **must** load balance at runtime
  - Consistent Hashing = blind selection
- Storage transport must be designed for scale-out and replication (3 replicas, etc.)
- Hence, NGH/Replicast
- Source on the github..

What's Next


Multicast
Unicast
NGH/Replicast
CH/Unicast
Consistent Hash + Best Effort
Group Hash + sub-Group Load Balancing

Q & A