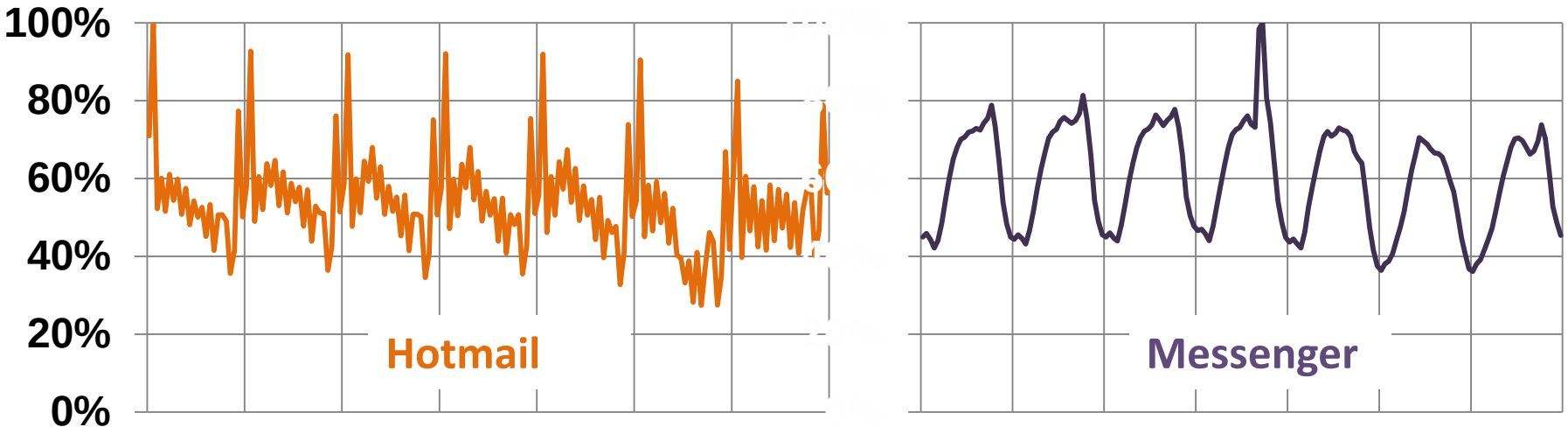
Sierra: practical power-proportionality for data center storage

Eno Thereska, Austin Donnelly, Dushyanth Narayanan

Microsoft Research Cambridge, UK

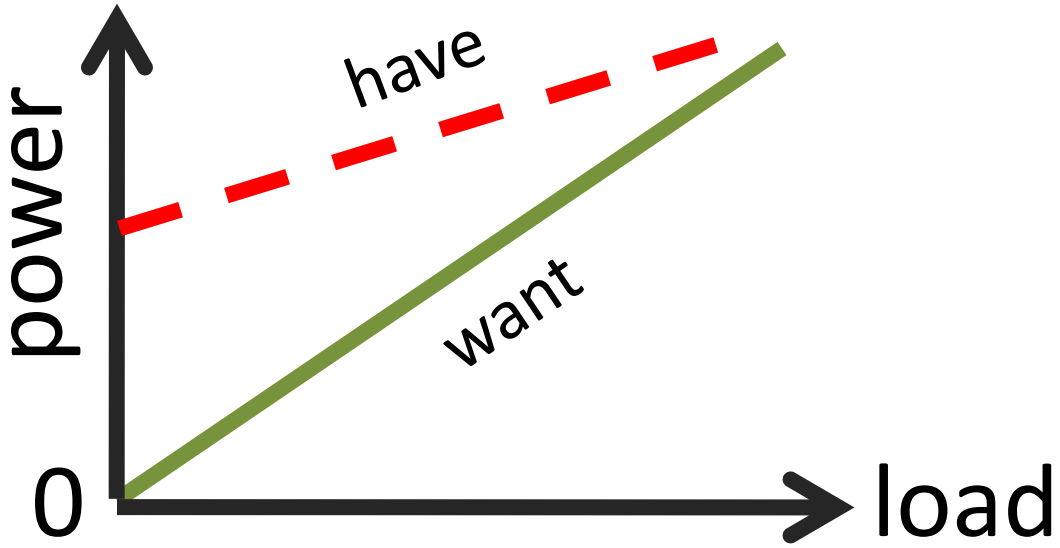
Our workloads have peaks and troughs



Servers not fully utilized, provisioned for peak

Zero-load server draws ~60% of power of fully loaded server!

Goal: power-proportional data center



- Hardware is not power proportional ☹
  - CPUs have DVS, but other components don't
- Power-proportionality in software
  - Turn off servers, rebalance CPU and I/O load

Storage is the elephant in the room

☺ CPU & network state can be migrated

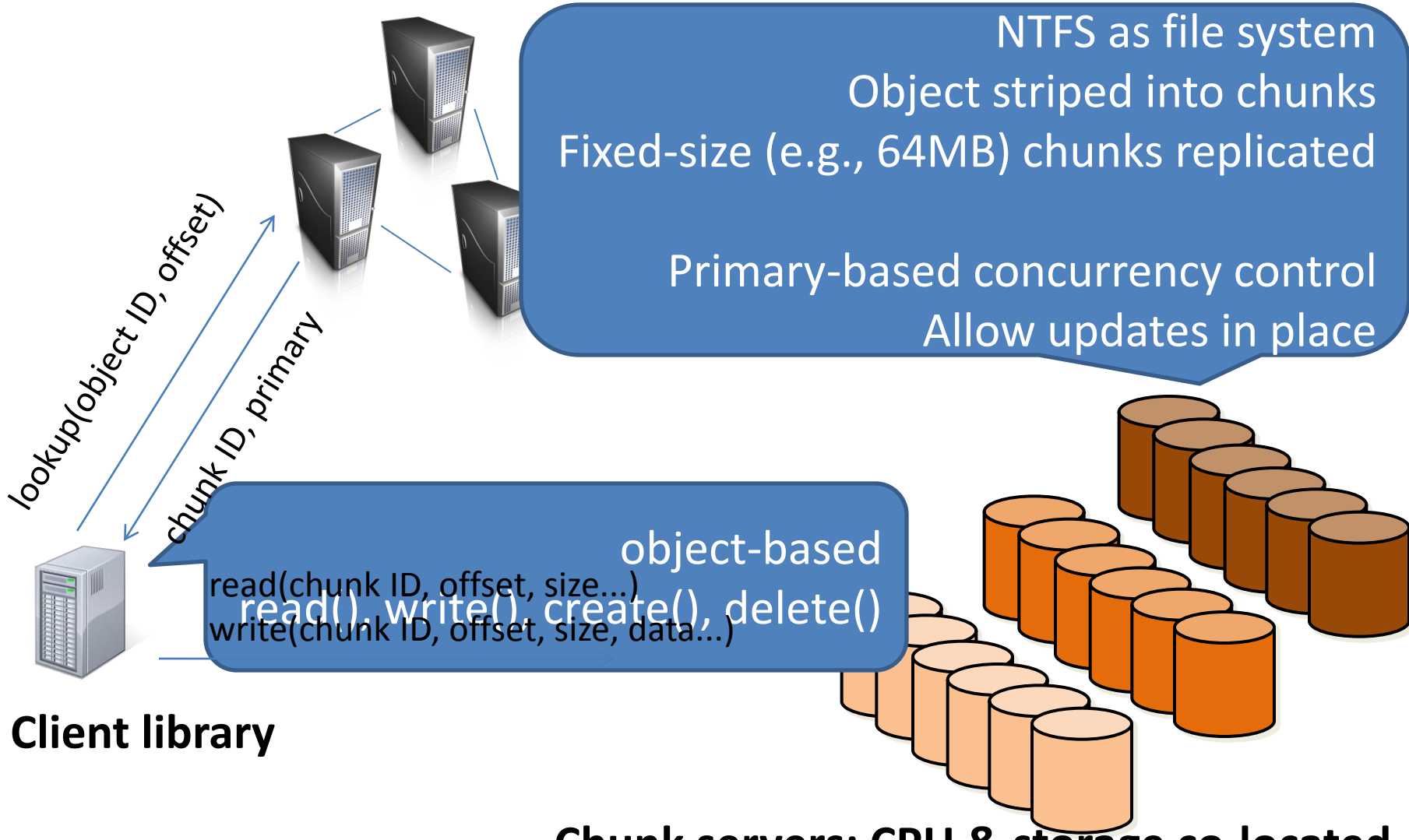
- Computation state: VM migration
- Network: Chen et al. [NSDI'08]

☹ Storage state can *not* be migrated

- Terabytes per server, petabytes per DC
- Diurnal patterns → migrate at least twice a day!

- Turn servers off, but keep data available?
  - and consistent, and fault-tolerant

Context: Azure-like system



Challenges

- Availability, (strong) consistency
- Recovery from transient failures
- Fast rebuild after permanent failure
- Good performance

- Gear up/down without losing any of these

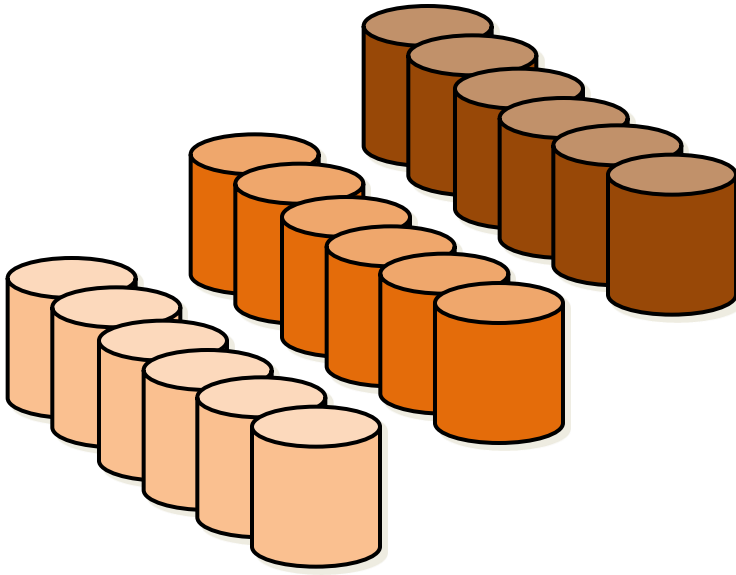
Sierra: storage subsystem with “gears”

- Gear level *g* → *g* replicas available
  - $0 \leq g \leq r = 3$
  - (r-g)/r of the servers are turned off
  - Gear level chosen based on load
  - At coarse time scale (hours)

Sierra in a nutshell

- Exploit R-way replication for read availability
- Careful layout to maximize #servers in standby
- Distributed virtual log for write availability & read/write consistency

- Good power savings
  - Hotmail: 23% - 50%



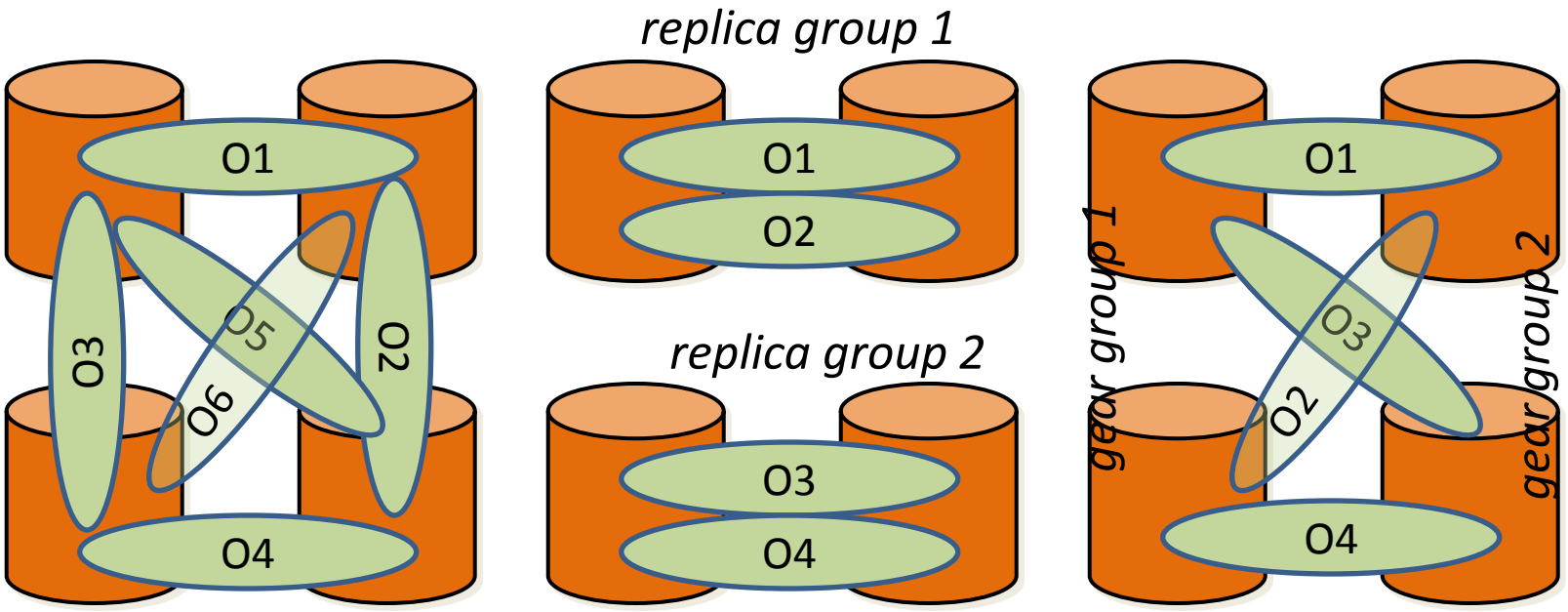
Outline

- Motivation
- Design
- Evaluation
- Future work and conclusion

Sierra design features

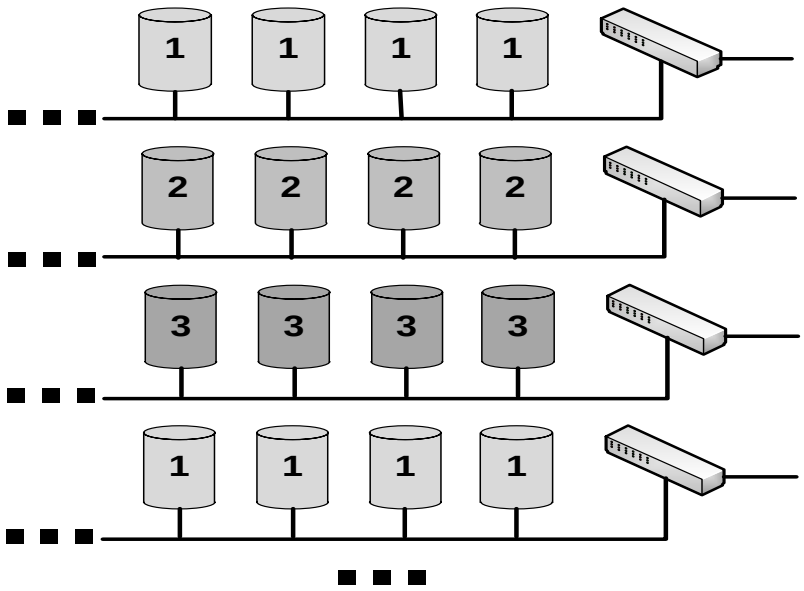
- Power-aware layout
- Distributed virtual log
- Load prediction and gear scheduling policies

Power-aware layout

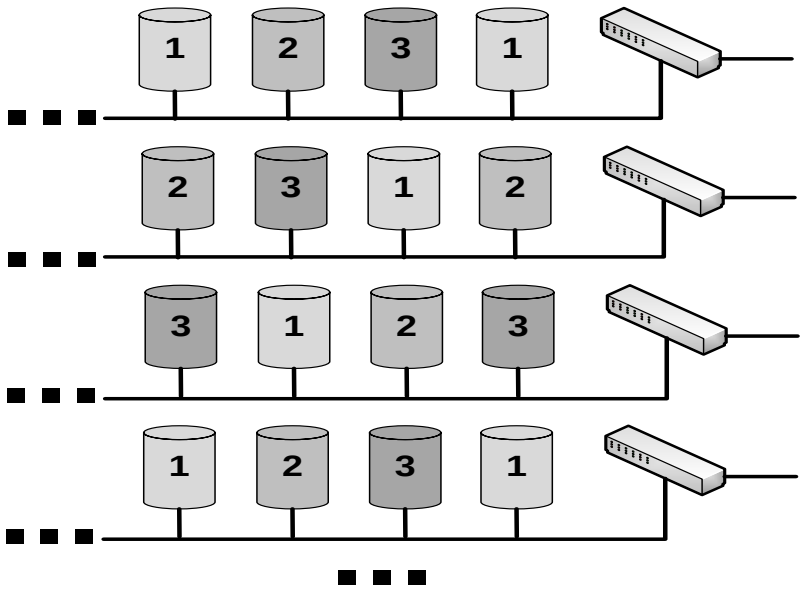


	Naïve random	***Naïve grouped***	***Sierra***
Power-down	r - g	N(r – g)/r	N(r – g)/r
Rebuild ||ism	N	1	N/r

Rack and switch layout



Rack - aligned

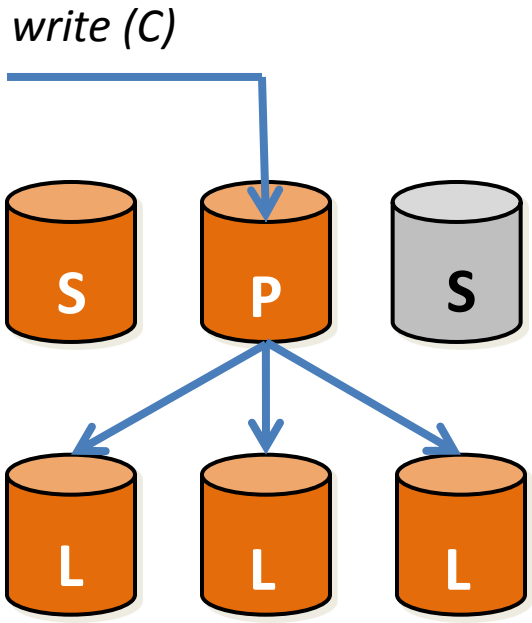


Rotated

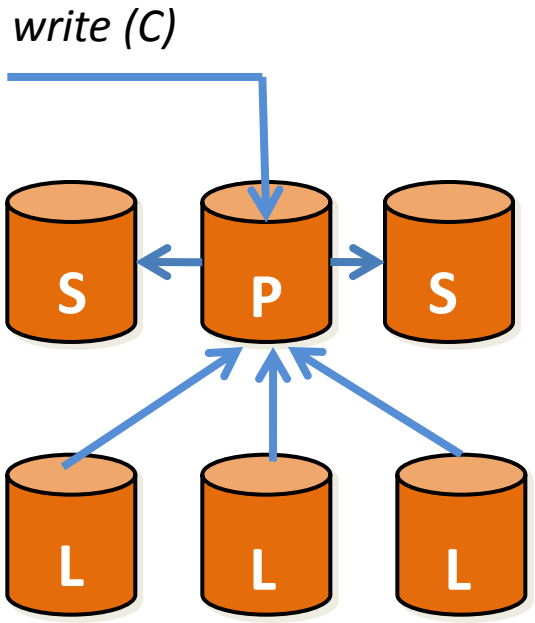
- Rack-aligned → switch off entire racks
- Rotated → better thermal balance

What about write availability?

- Distributed virtual log (DVL)



offloading mode (low gear)

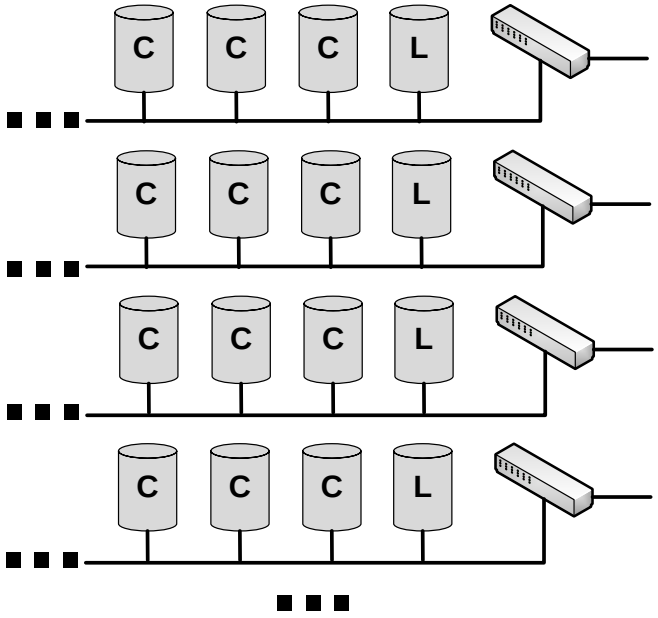


reclaim mode (highest gear)

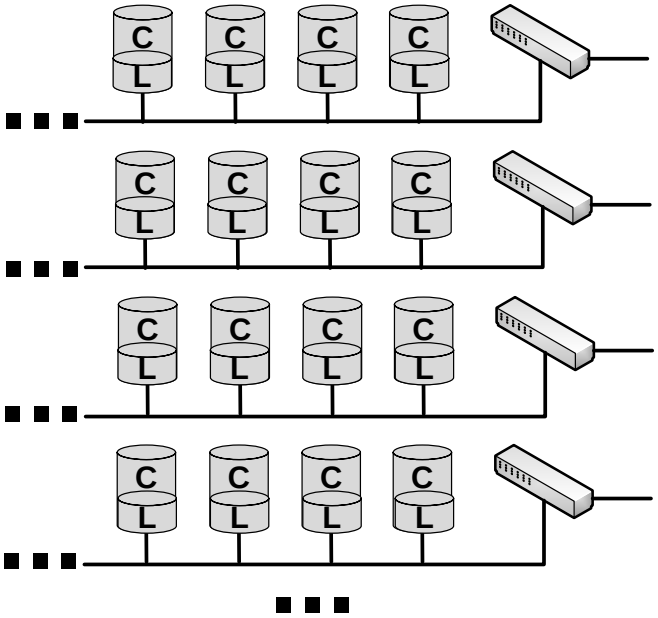
Distributed virtual log

- Builds on past work [FAST'08,OSDI'08]
- Evolved as a distributed system component
  - Available, consistent, recoverable, fault-tolerant
  - Location-aware (network locality , fault domains)
  - "Pick r closest loggers that are uncorrelated"
- All data eventually reclaimed
  - Versioned store is for short-term use

Rack and switch layout



Dedicated



Co-located

- Dedicated loggers → avoid contention
- Co-located loggers → better multiplexing

Handling new failure modes

- Failure detected using heartbeats
- On chunkserver failure during low-gear
  - MDS wakes up all peers, migrate primaries
  - In *g=1* there is short unavailability ~ O(time to wake up)
  - Tradeoff between power savings and availability using *g=2*
- Logger failures
  - Wake up servers, reclaim data
- Failures from powering off servers
  - Power off few times a day
  - Rotate gearing

Load prediction and gear scheduling

- Use past to predict future (very simple)
  - History in 1-hour buckets → predict for next day
  - Schedule gear changes (at most once per hour)
  - Load metric considers rand/seq reads/writes

- A hybrid predictive + reactive approach is likely to be superior for other workloads

Implementation status

- User-level, event-based implementation in C
- Chunk servers + MDS + client library = 11kLOC
- DVL is 7.6 kLOC
- 17 kLOC of support code (RPC libraries etc)
- +NTFS (no changes)
- MDS is not replicated yet

Summary of tradeoffs and limitations
(see paper for interesting details)

- New power-aware placement mechanism
  - Power savings vs. rebuild speed vs. load balancing
- New service: distributed virtual log
  - Co-located with vs. dedicated chunk servers
- Availability vs. power savings
  - 1 new failure case exposes this tradeoff
- Spectrum of tradeoffs for gear scheduler
  - Predictive vs. reactive vs. hybrid

Outline

- Motivation
- Design
- Evaluation
- Future work and conclusion

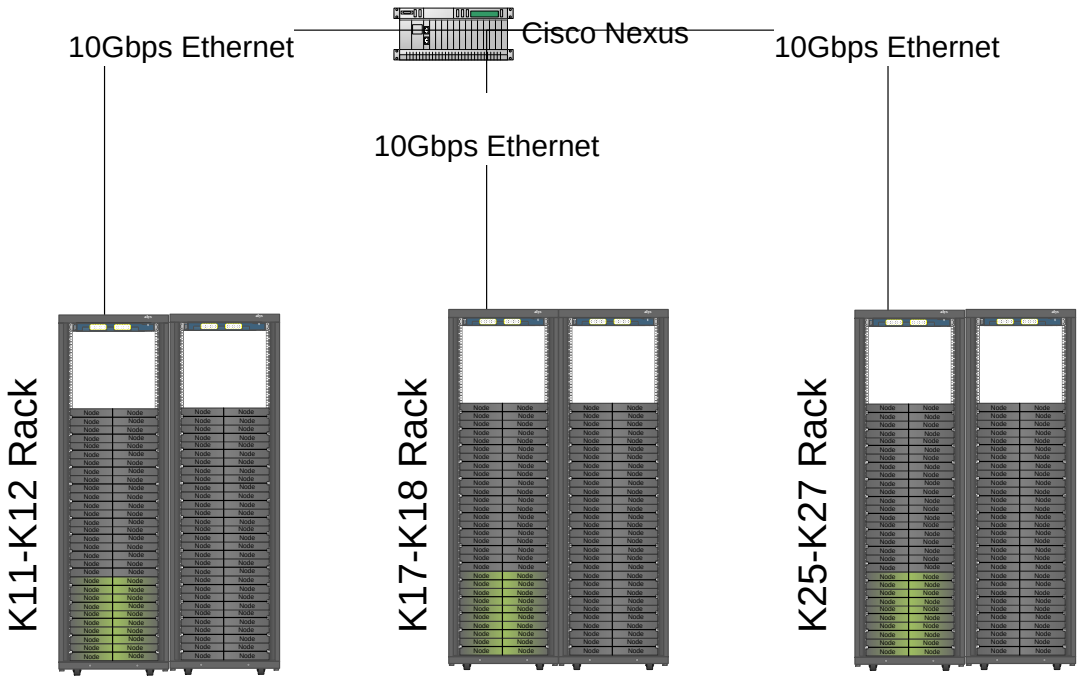
Evaluation map

- Analysis of 1-week large-scale load traces from Hotmail and Messenger
  - Can we predict load patterns?
  - What is the power savings potential?
- 48-hour I/O request traces + hardware testbed
  - Does gear shifting hurt performance?
  - Power savings (current and upper bound)

Hotmail I/O traces

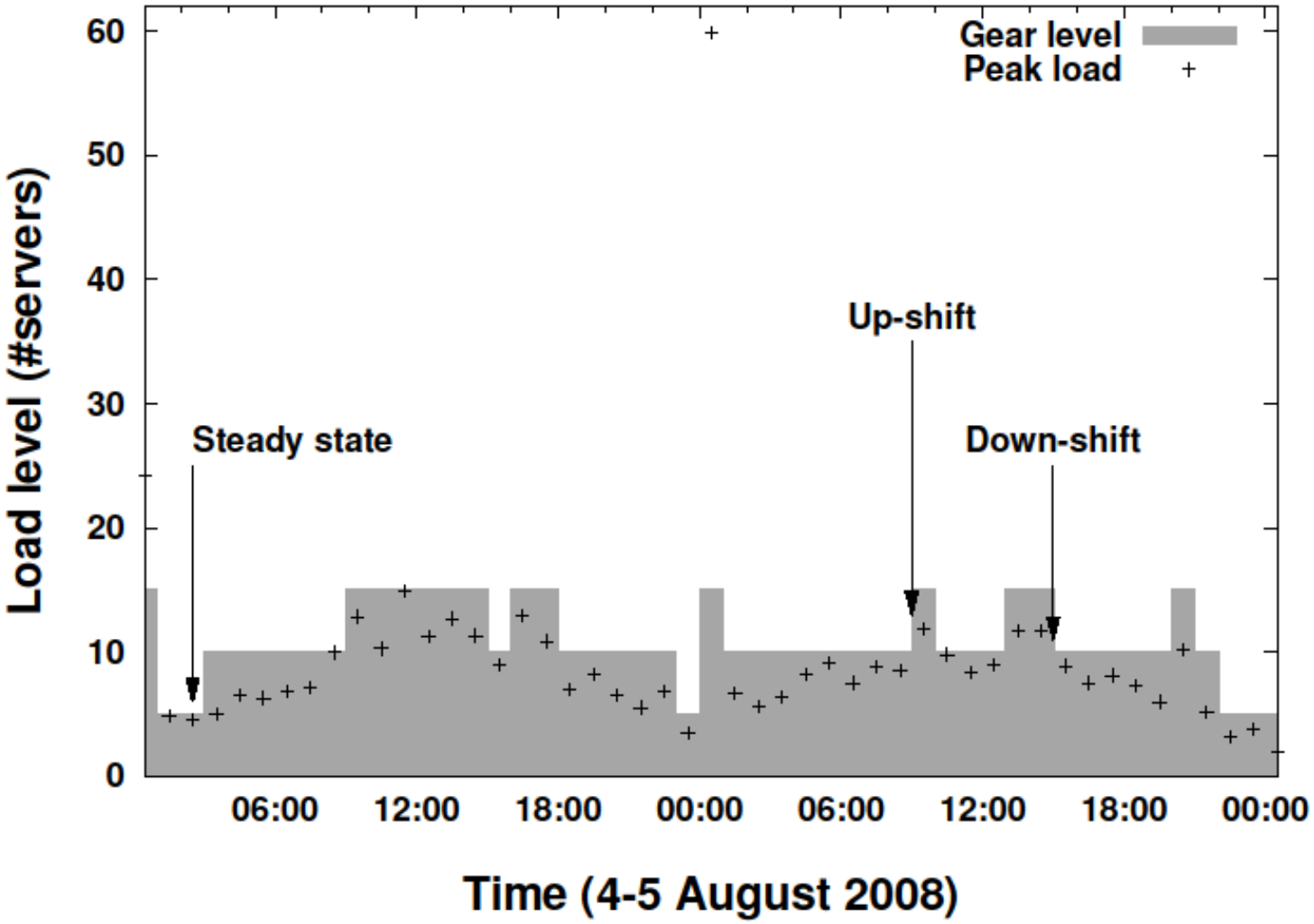
- 8 Hotmail backend servers, 48 hours
- 3-way replication
- Block I/O traces
- Data (msg files) accesses only
- 1 MB chunk size (to fit trace)

Testbed: MSR cluster

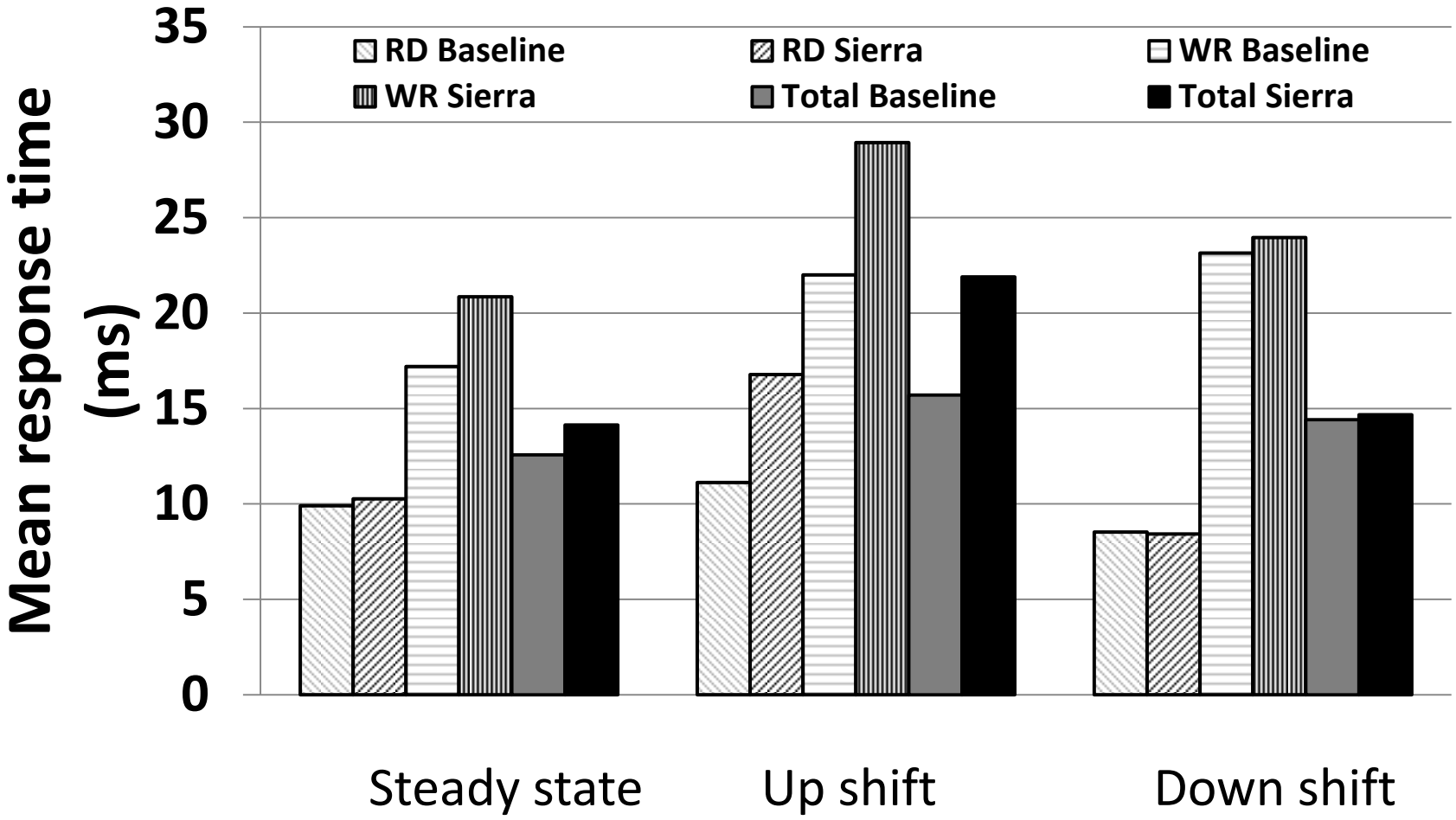


- Lots of effort spent on provisioning correctly (see paper for details)
- 6x3 chunk servers in baseline, 5x3 chunk servers + 1x3 loggers in Sierra; clients on other servers; 1 MDS

Load, gears, interesting periods

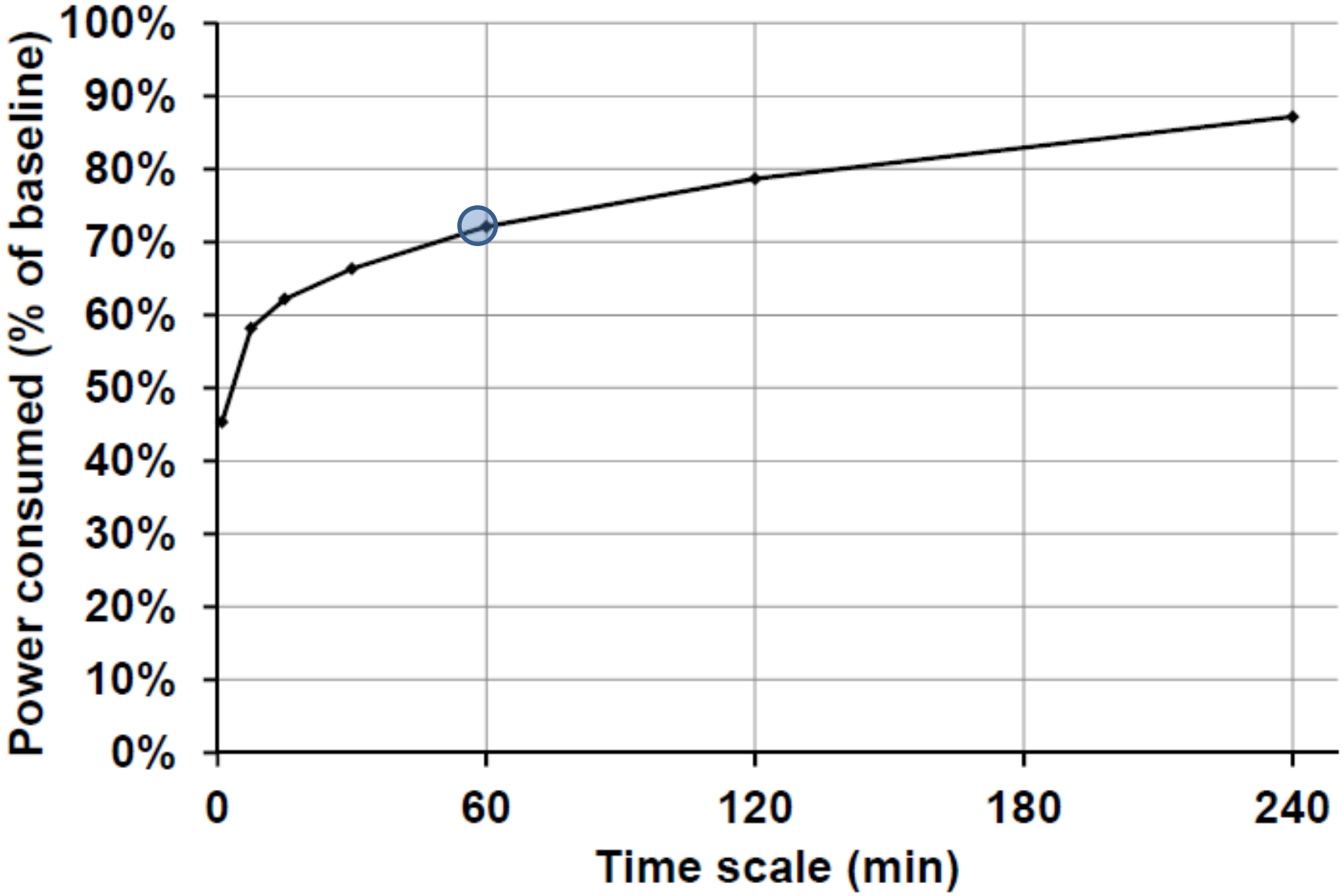


Performance during key stages



One power savings curve

(more curves in paper)



Summary

- A step towards better power-proportionality
  - In software, no reliance on HW-based approaches
  - Storage state as main challenge (vs. computation)
- Several challenges addressed
  - Layout, availability, consistency, performance
- Working prototype
  - Lots of interesting technical details in the paper

Future work

- Filling in troughs with useful work
- Interactions with CPU/Network-based consolidation
- Quorum-based & Byzantine fault tolerant systems & erasure codes

Related work

- Power savings in RAID arrays
  - Write off-loading [FAST'08], PARAID[FAST'07]
  - Multi-speed disks in Hibernator [SOSP'05]
- A possible improvement over our layout for read-only workloads [SOCC'10]
- Hot vs. Cold data
  - Popular data concentration [ISC'04]
- Power savings for other resources
  - CPU voltage scaling & VM migration
  - Network considerations [NSDI'08]

Thank you

- http://research.microsoft.com/sierra/