On the Energy (In)efficiency of Hadoop: Scale-down Efficiency

Jacob Leverich
and Christos Kozyrakis

Stanford University

The current design of Hadoop precludes scale-down of commodity clusters.

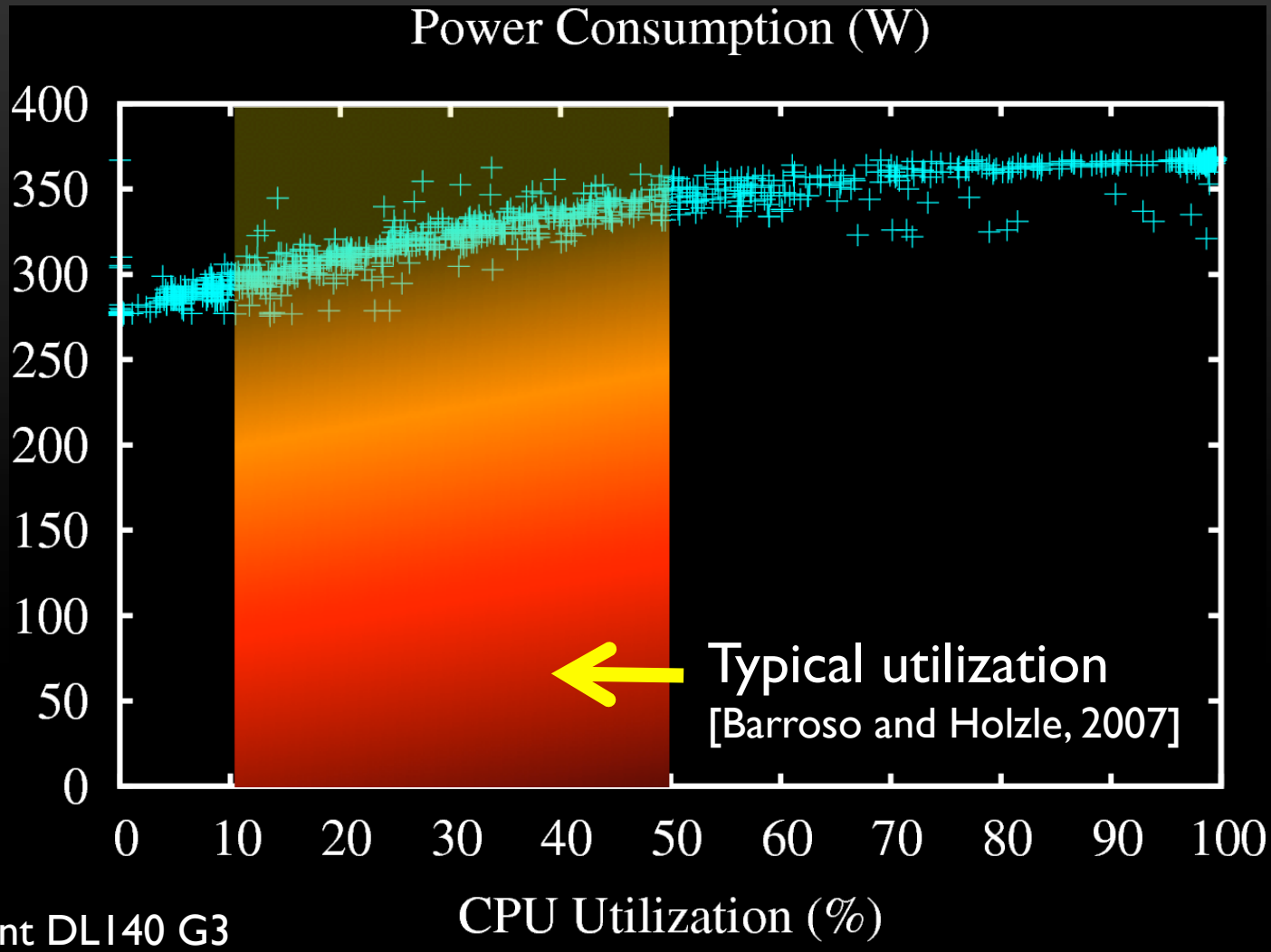
Outline

- Hadoop crash-course
- Scale-down efficiency
- How Hadoop precludes scale-down
- How to fix it
- Did we fix it?
- Future work

Hadoop crash-course

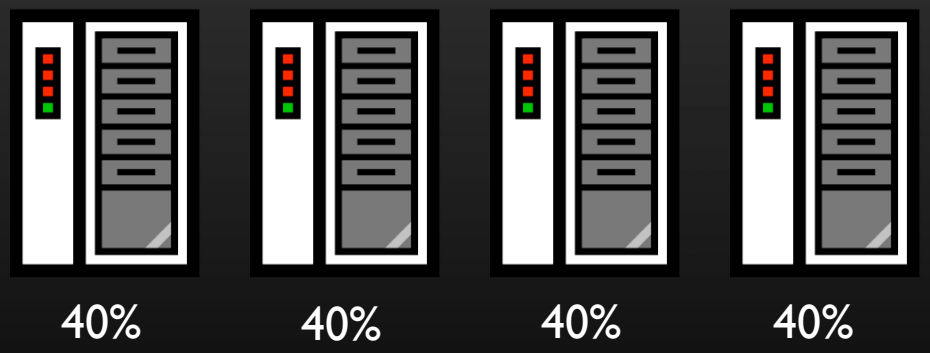
- Hadoop == Distributed Processing Framework
  - 1000s of nodes, PBs of data
- Hadoop MapReduce ≈ Google MapReduce
  - **Tasks** are automatically distributed by the framework.
- Hadoop Distributed File System ≈ Google File System
  - Files divided into large (64MB) blocks; amortizes overheads.
  - Blocks replicated for availability and durability management.

Scale-down motivation

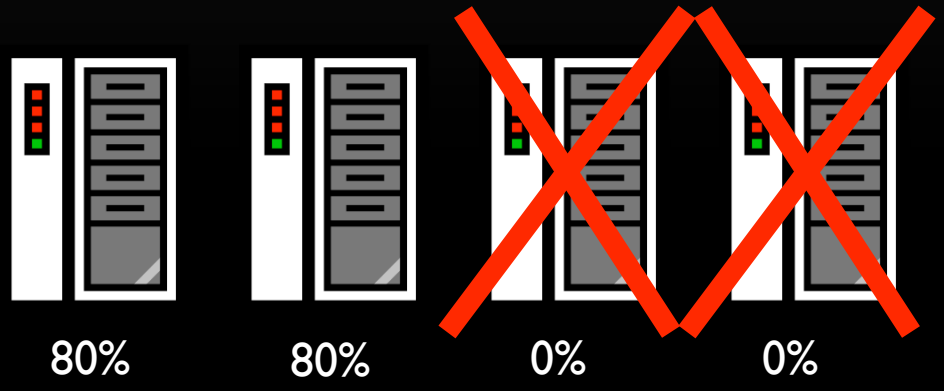


HP Proliant DL140 G3

Scale-down for energy proportionality

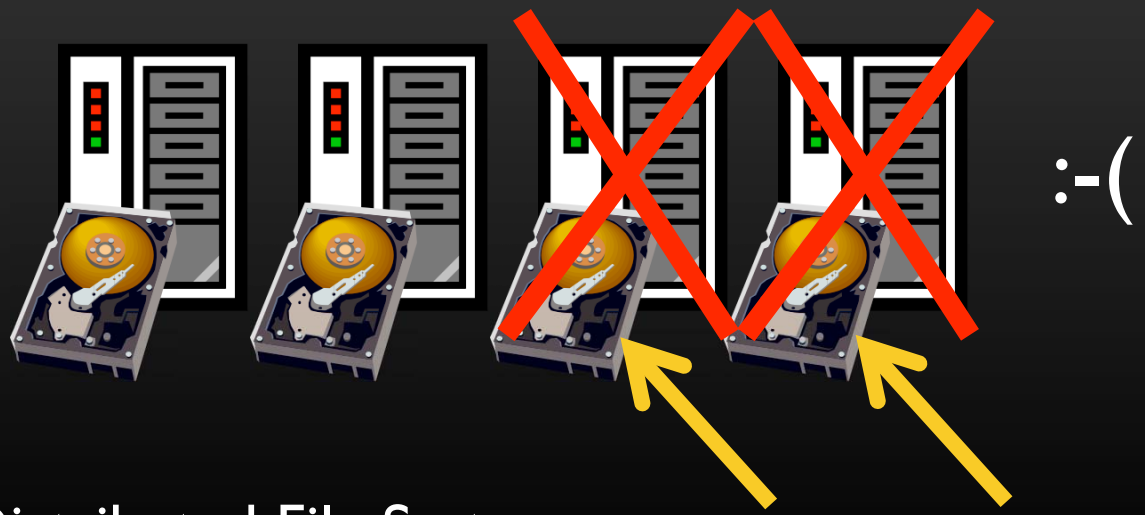


= 4 x P(40%)
= 4 x 325W = **1300W**



= 2 x P(80%)
= 2 x 365W = **730W**

The problem: storage consolidation



- Hadoop Distributed File System…
  - Consolidate computation? Easy.
  - Consolidate storage? Not (as) easy.

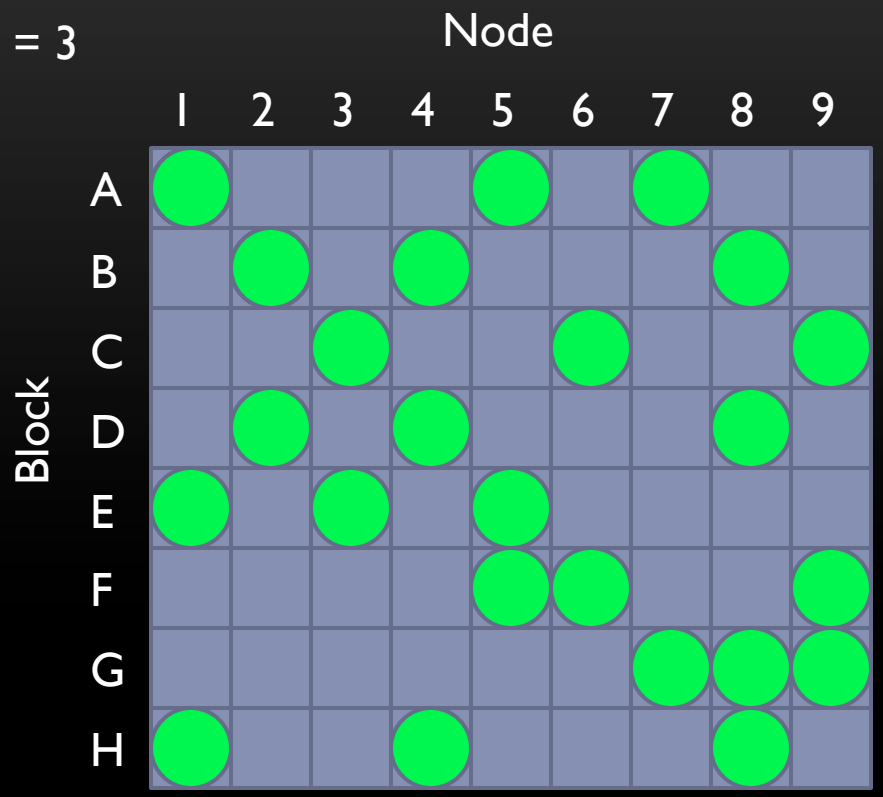
- "All servers must be available, even during low-load periods." [Barroso and Holzle, 2007]
  - Hadoop inherited this "feature" from Google File System

HDFS and block replication

= replica

Replication factor = 3

"block replication table"



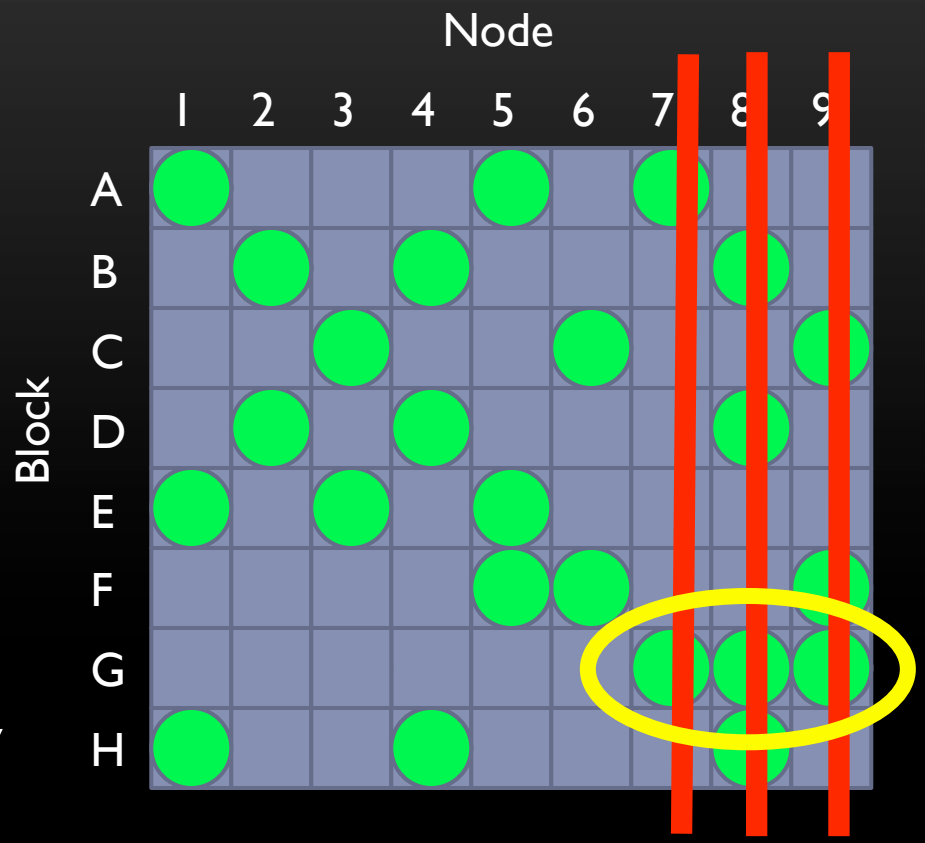
1st replica local, others remote.

Allocate evenly.

Attempted scale-down

Problems:

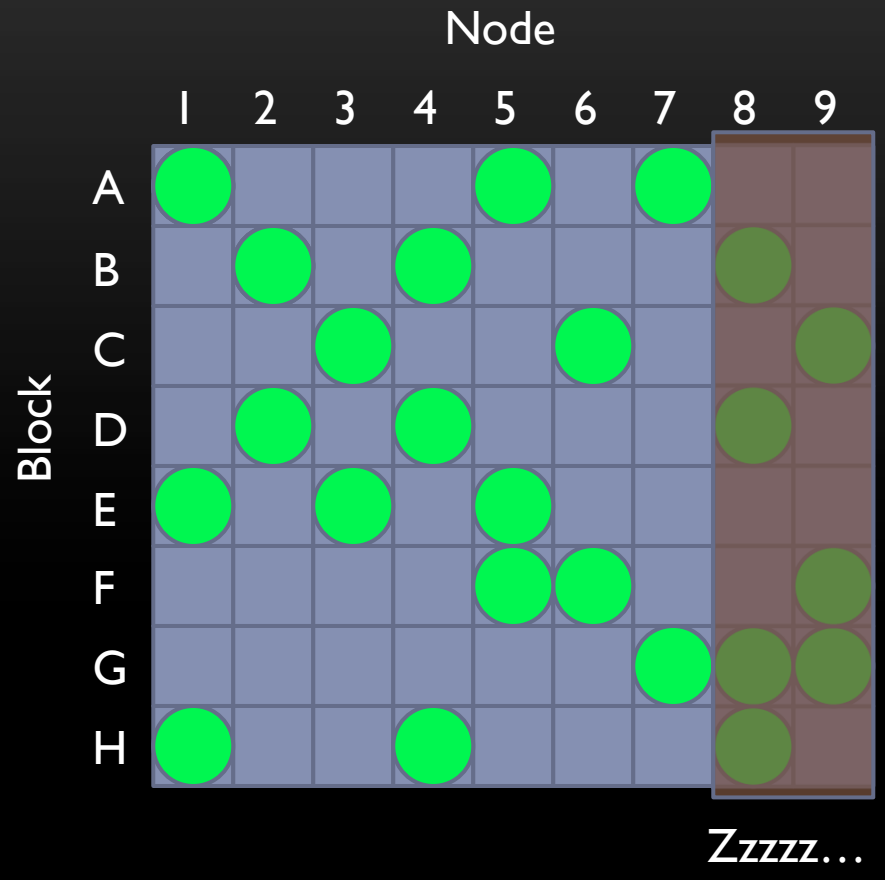
- Scale-down vs. Self-healing
  - Wasted capacity:
    sleeping replicas != lost replicas
  - Flurry of net & disk activity!
- Which nodes to disable?
  - Must maintain data availability



How to fix it

- Scale-down vs. Self-healing  Self-non-healing
  - Wasted capacity:
    sleeping replicas != lost replicas
  - Flurry of net & disk activity!
- Which nodes to disable?
  - Must maintain data availability

Self-non-healing



- Coordinate with Hadoop when we put a node to sleep
- Prevent block re-replications

New RPCs in HDFS primary node

- **`sleepNode(String hostname)`**
  - Similar to node decommissioning, but don't replicate blocks

    ```
    % hadoop dfsadmin –sleepNode 10.10.1.80:50020
    ```
  - Save blocks to a "sleeping blocks" map for bookkeeping
  - Ignore heartbeats and block reports from this node
- **`wakeNode(String hostname)`**
  - Watch for heartbeats, force node to send block report
  - Execute arbitrary commands (i.e. send wake-on-LAN packet)
- **`wakeBlock(Block target)`**
  - Wake a sleeping node that has a particular block

How to fix it

- Scale-down vs. Self-healing → Self-non-healing
  - Wasted capacity:
    sleeping replicas != lost replicas
  - Flurry of net & disk activity!
- Which nodes to disable? → "Covering Subset" replication invariant
  - Must maintain data availability

Replication placement invariants

- Hadoop uses simple invariants to direct block placement
- Example: Rack-Aware Block Placement
  - Protects against common-mode failures (i.e. switch failure, power delivery failure)
  - Invariant: Blocks must have replicas on at least 2 racks.
- Is there some energy-efficient replication invariant?
  - Must inform our decision on which nodes we can disable.

Covering subset replication invariant

- Goal:

  Maximize the number of servers that can simultaneously sleep.

- Strategy:

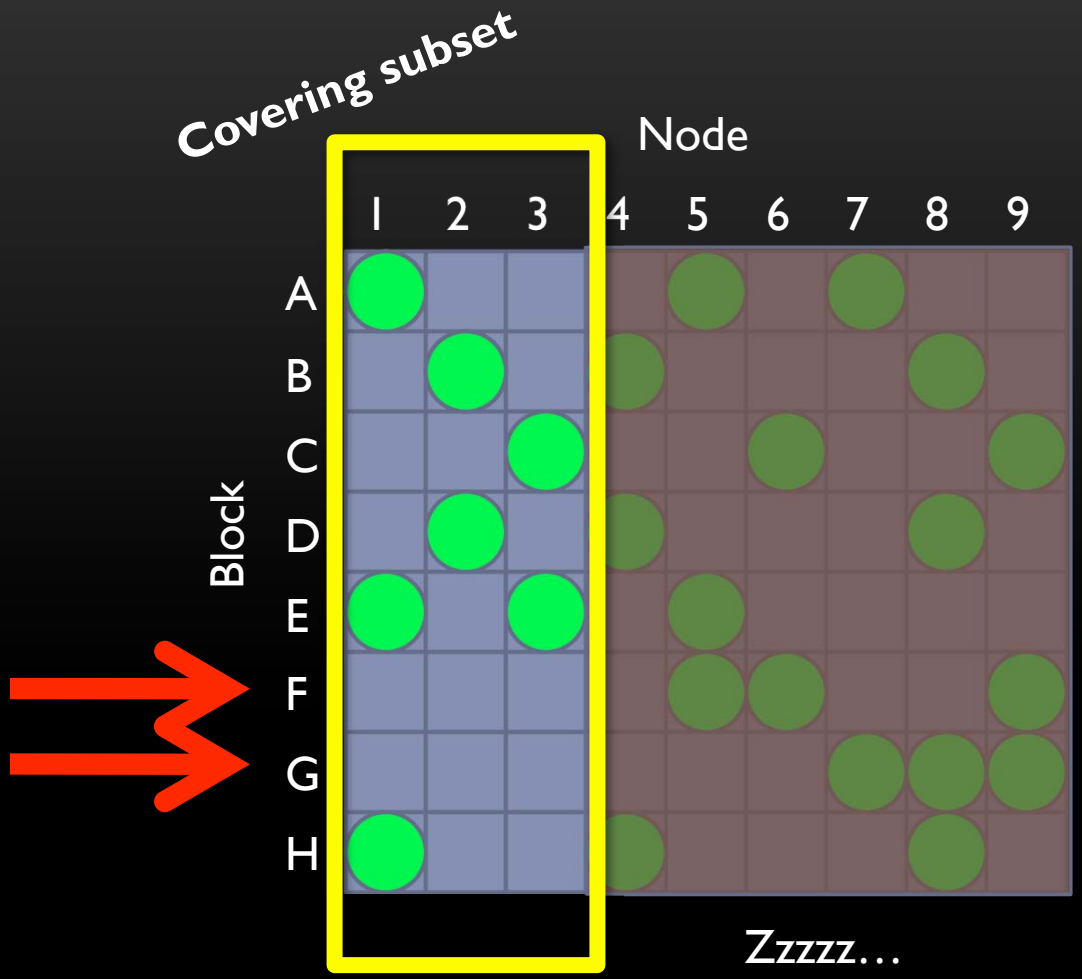
  Aggregate live data onto a "covering subset" of nodes.

  Never turn off a node in the covering subset.

- Invariant:

  Every block must have one replica in the covering subset.

Covering subset replication invariant



How to fix it

- Scale-down vs. Self-healing → Self-non-healing
  - Wasted capacity:
    sleeping replicas != lost replicas
  - Flurry of net & disk activity!
- Which nodes to disable? → "Covering Subset" replication invariant
  - Must maintain data availability

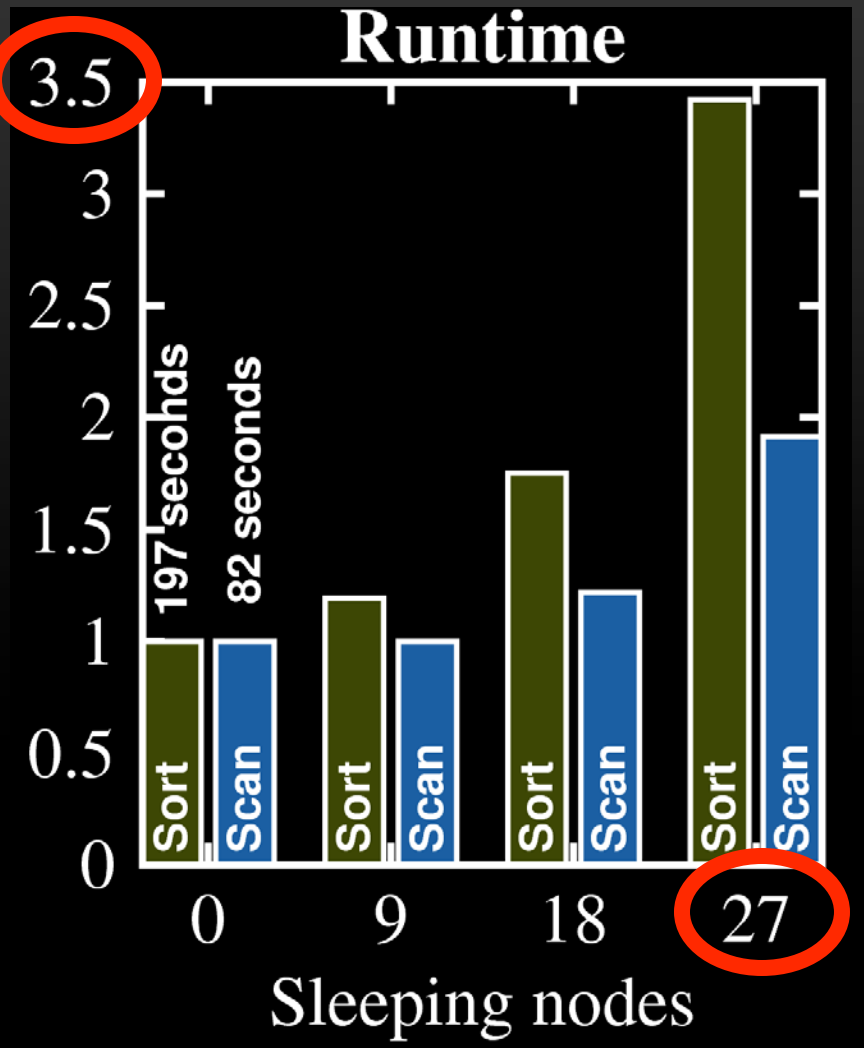
Evaluation

Methodology

- Disable *n* nodes, compare Hadoop job energy & perf.
  - Individual runs of webdata_sort/webdata_scan from GridMix
  - 30 minute job batches *(with some idle time!)*
- Cluster
  - 36 nodes, HP Proliant DL140 G3
  - 2 quad-core Xeon 5335s each, 32GB RAM, 500GB disk
  - 9-node covering subset (1/4 of the cluster)
- Energy model
  - Validated estimate based on CPU utilization
  - Disabled node = 0 Watts
  - Possible to evaluate hypothetical hardware

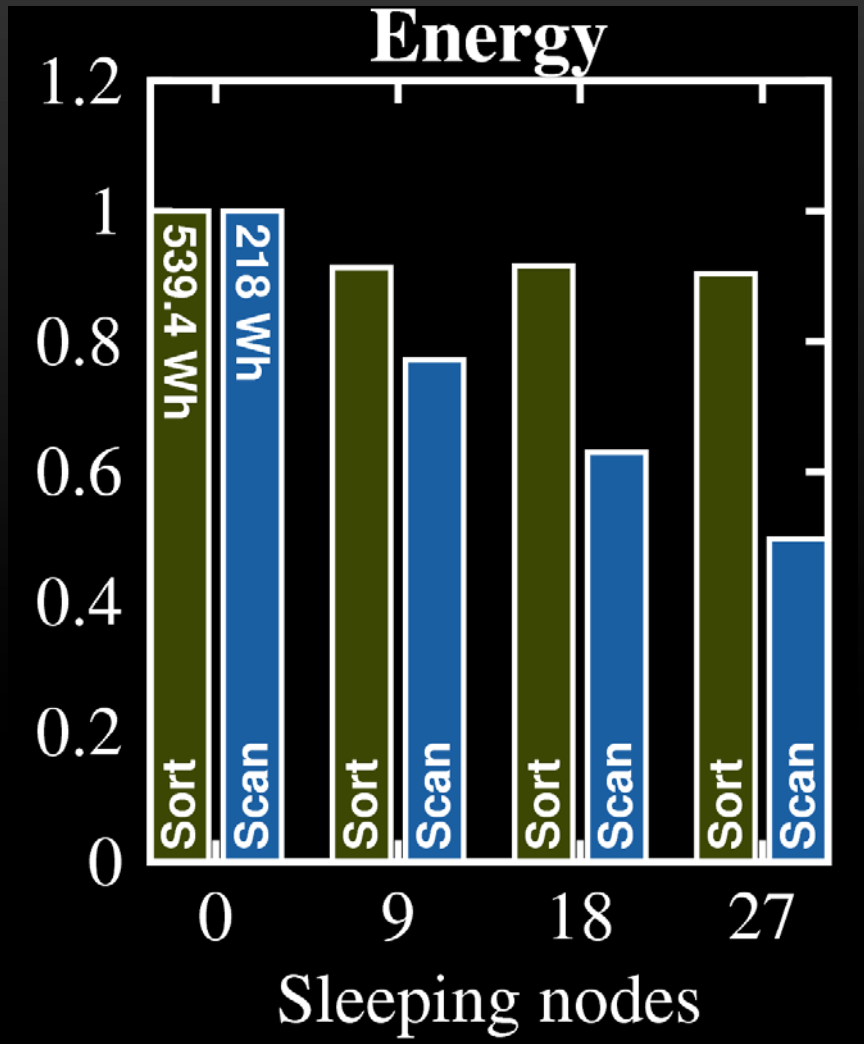
Results: Performance

- It slows down (obviously)
  - Peak performance benchmark
- Sort (network intensive) worse off than Scan
- Amdahl's Law



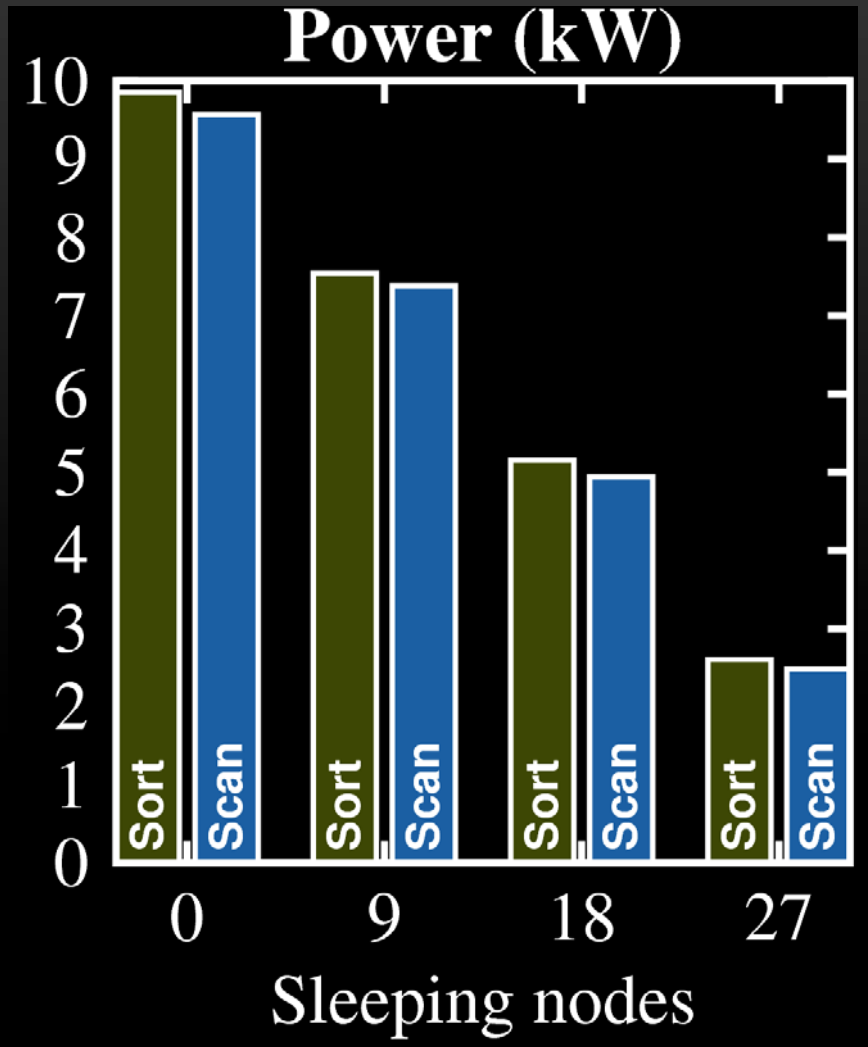
Results: Energy

- Less energy consumed for same amount of work
  - 9% to 51% saved
- Nodes consume energy more than they improve performance
- Slower systems usually more efficient;
  high performance is a **trade-off!**



Results: Power

- Excellent knob for cluster-level power capping
- Much larger dynamic range than tweaking frequency/voltage at the server level



Results: The Bottom Line

Operational Hadoop clusters can scale-down.

We reduce energy consumption

at the expense of single-job latency.

Continuing Work

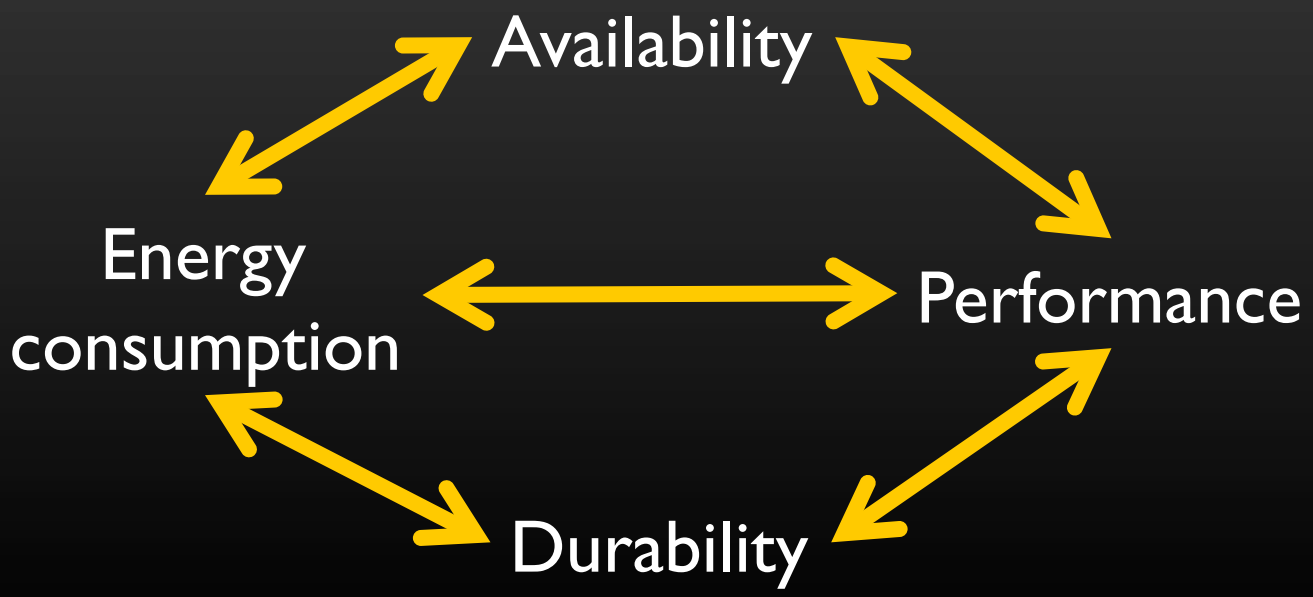
Covering subset: **mechanism vs. policy**

- The replication invariant is a mechanism.
- Which nodes constitute a subset is policy (open question).

- Size trade-off
  - Too small: Low capacity and **performance bottleneck**
  - Too large: Wasted energy on idle nodes
  - `1` / (replication factor) ← reasonable starting point

- How many covering subsets?
  - Invariant: Blocks must have a replica in ***each*** covering subsets.

Quantify Trade-offs



- Random Fault Injection experiments
  - What happens when a covering subset node fails?
- How much do you trust idle disks?

Dynamic Power Management

- Algorithmically decide which nodes to sleep or wakeup
- What signals to use?
  - CPU utilization?
  - Disk/net utilization?
  - Job Queue length?
- MapReduce and HDFS must cooperate
  - i.e. idle nodes may host transient Map outputs

Workloads

- Benchmarks
  - HBase/BigTable vs. MapReduce
    - Short, unpredictable data access vs. long streaming access
    - Quality of service and throughput are important
  - Pig vs. Sort+Scan
  - Recorded job traces vs. random job traces
- Peak performance vs. fractional utilization
- **What are typical usage patterns?**

Scale

- 36-nodes to 1000-nodes; emergent behaviors?
  - Network hierarchy
  - Hadoop framework inefficiencies
  - Computational overhead (must process many block reports!)

- Experiments on Amazon EC2
  - Awarded an Amazon Web Services grant
  - Can't measure power! Must use a model.

    Any Amazonians here? Let's make a validated energy model.