

FAWN: A Fast Array of Wimpy Nodes

David G. Andersen, Jason Franklin, Michael
Kaminsky, Amar Phanishayee, Lawrence Tan, Vijay
Vasudevan

Presented By,
Elangovan KN

The Requirement

- **Key-value** storage system are growing in size and importance.
- Some examples include Amazon's **Dynamo**, LinkedIn's **Voldemort**, Facebook's **memcached** storage systems.
- These are data intensive, **I/O operations** with large amount of **parallel** and independent **random access**.
- Their high loads require large clusters to support them and the clusters must provide high performance, low cost operation.

Existing Solutions

- **Conventional disk-based clusters**

- The traditional storage disk such as the magnetic disk might be used in the clusters.
- However, the conventional disks perform poorly during random-access workloads.
- Hence this method becomes inefficient in terms of performance.

- **DRAM-based clusters**

- DRAMs have high performance even for random access workloads.

Existing Solutions (cntd.)

- However, storage of terabytes of data will be very expensive and power consuming.
- Two 2GB Dual Inline Memory Module (DIMMs) consume the **same amount of power** as a 1TB hard disk.
- These clusters power consumption can account up to **50%** of the total cost.
- In the future, the datacenters that house them may need a dedicated substation to operate properly.
- Hence there is a serious tradeoff in existing solutions between power and performance.

The Problem

- Looking at these problems, the question arises,
Can a cost-effective cluster be built that reduces the power consumption while meeting the performance requirements of these data intensive applications?
- The authors address this question with their architecture, the **Fast Array of Wimpy Nodes** (FAWN).
- A Wimpy node can be defined as a low power, low-cost CPU coupled with flash storage.

How FAWN Solves the Problem?

- Flash memory provides low power, efficient way for fast random reads.
- By providing proper cache and mapping, the write performance of the flash can be improved.
- By reducing the number of random writes, the performance can be improved further.
- FAWN combines these properties to produce low power, high performance architecture.
- Furthermore, FAWN architecture is designed to balance computation and I/O capabilities to ensure efficient and parallel access to data.

Questions?