**GOAL: SCALE SYSTEM THROUGHPUT LINEARLY AS ADDING SERVERS**

**Observation:**
- Load balance is often workload dependent
- Example: 85-node FAWN key-value cluster
- 10K reqs/sec per node for key lookups
- Hash-based partition: nodeID = Hash(key)
- Uniformly access keys: tput scales linearly
- Biased access: underutilize system capacity

**Question:**
- Can we provide workload-independent load balance?

**SMALL CACHE, BIG EFFECT: PROVABLE LOAD BALANCING FOR RANDOMLY PARTITIONED CLUSTER SERVICES**

**Requirement:**
- Hash-based service partition
- Service partition opaque to clients
- Cacheable queries

**Intuition:**
- Skewed workload, but cache friendly
- Unfriendly to cache, but uniform workload

**Major Result:**
- If cache size = k * n * log n,
  - tput > (1-ɛ) * total capacity, regardless of workload and total number of items
- n: # nodes,
- k: a small and tunable constant factor

**Analytical vs Empirical**
- number nodes = 85

**Work distribution w/o cache**
- x: working set size

**Scalability w/ cache**
- cachesize = 8 n logn

**Analytical bound is accurate**

**EVALUATION: FRONTEND 900K REQ/S; BACKEND 10K REQ/S**

**Architecture**

**Queries**

**Client**

**Query director**

**Backend**

**Cache**

**Requirement**

Popularity based, small but fast

Larger working set yields better balanced load

Worst case tput very close to uniform case