Efficient On-Demand Operations in Distributed Infrastructures

Steve Ko and Indranil Gupta
Distributed Protocols Research Group
University of Illinois at Urbana-Champaign

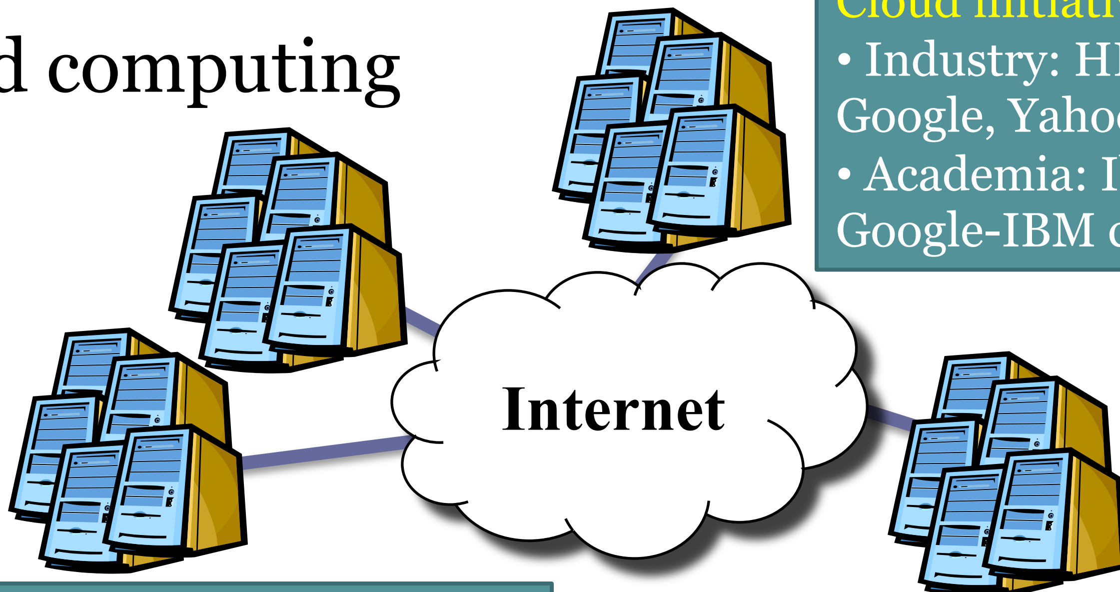
One-Line Summary

We need to design responsive datacenters that handle massive data.

- Will use monitoring (querying) as a main example

The Need

- Cloud computing



Cloud initiatives
- Industry: HP, IBM, Amazon, Google, Yahoo, MS, …
- Academia: Illinois CCT, Google-IBM cluster, etc

Apps in the Cloud
- Social networking sites: facebook, LiveJournal, etc
- In-Cloud desktop software: Google docs, Photoshop express, Apple's MobileMe, etc
- Data-intensive apps: log analysis, scientific computing, etc

Massive data in the Cloud
- Arise in many domains, e.g., scientific computing, social networking sites, web search engines, etc
- TB to PB (e.g., Yahoo 12TB/day)

The Requirement

Responsiveness while handling massive data

- Web 2.0 apps: users expect desktop-quality responsiveness.
- Data-intensive processing: long, long time to finish

Importance

- Amazon: every 100ms of latency cost them 1% in sales.
- Google: an extra .5 seconds in search page generation time dropped traffic by 20%.
- “Users really respond to speed” – Google VP Marissa Mayer
- Coadd took 70 days to complete with over 30 sites and 4,500 CPUs.
  - Coadd? a spatial processing application with 44,000 tasks accessing 588,900 files in total

Not Easy to Do Both

Datacenter monitoring example

- Monitoring responsiveness: quick query results regarding the state of the datacenter

Case of HP OpenView

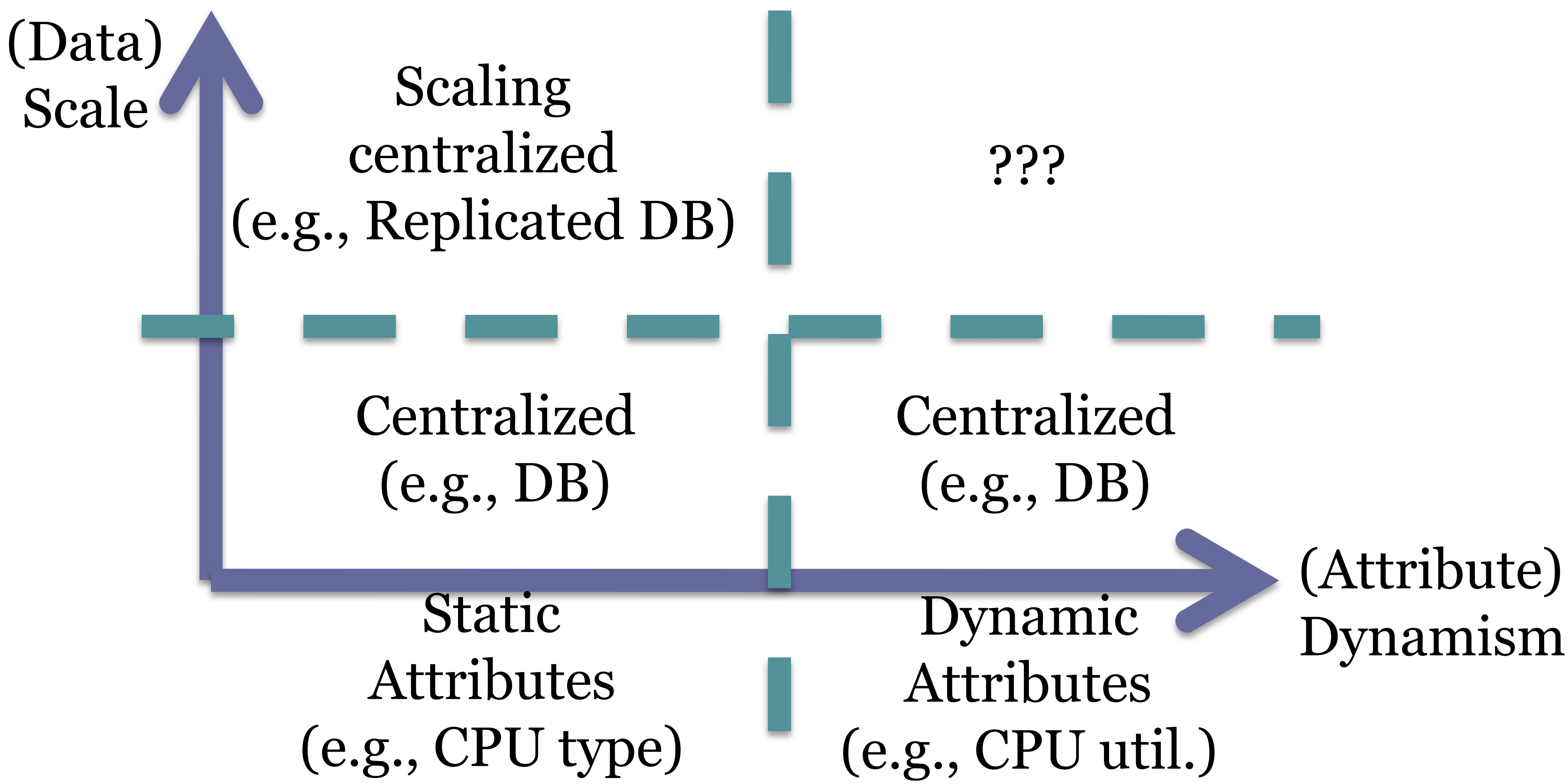
- 144 attributes, store them in a database
- Data scale: # of attributes * # of machines
- First-cut attempt in 2006: “6 ½ hours for 6,200 machines, and 12 hours for 11,500 machines”
- Not possible to query the most up-to-date state

More than merely scale

- Static vs. dynamic attributes

Challenges – Scale & Dynamism

Monitoring example



On-Demand Operations

On-demand operations

- Operations that are responsive in spite of massive data
- Challenges: scale & dynamism

Examples

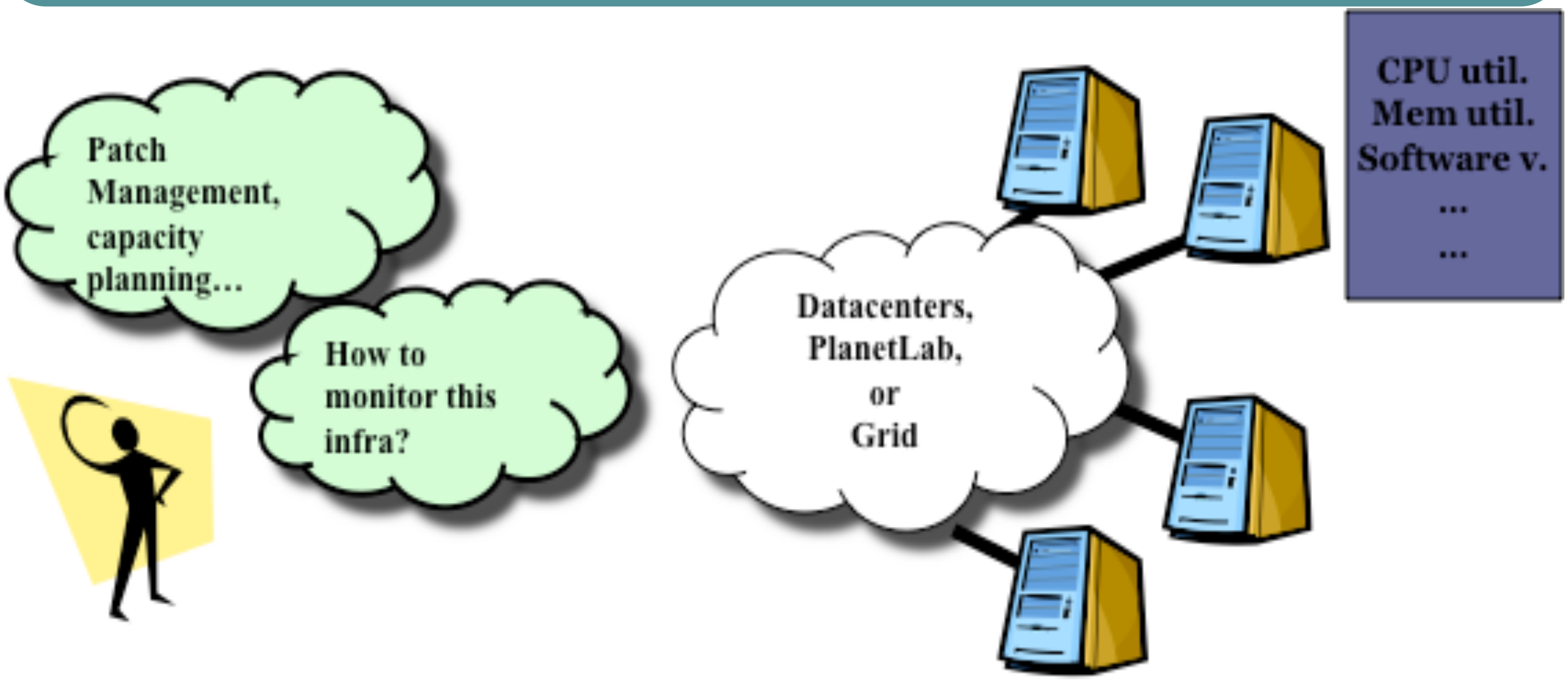
- On-demand monitoring: quick response even when querying the most up-to-date state of the infrastructure
- On-demand scheduling for data-intensive workload: quick scheduling decisions reflecting the current resource availability

On-Demand Group Querying

Steve Ko (UIUC),
Praveen Yalagandula (HP Labs),
Indranil Gupta (UIUC),
Vanish Talwar (HP Labs),
Dejan Milojicic (HP Labs),
and Subu Iyer (HP Labs)

Problem

Group monitoring (querying) in large-scale infrastructures



Necessity

Groups are formed naturally

- A set of machines running a service in a datacenter, PlanetLab Slices, Grid sites, etc

Users want to query attributes from groups

- Avg. CPU utilization of a slice
- Top-3 loaded machines running Linux and executing a MapReduce task
- List of machines in rack R with either CPU util. < 10% or Mem util. < 5% running less than 2 VMs on either Xen or VMware

Requirements

Efficient group support with expressive queries

- Single groups & multiple groups
  - Queries targeting unions and intersections of groups
  - List of machines in rack R with CPU util. < 10% or Mem util. < 5% running less than 2 VMs on Xen or VMware
  - *A and (B or C) and D and (E or F*)
- Static groups & dynamic groups
  - Static groups: Linux machines, web servers, etc
  - Dynamic groups: CPU > 90%, # of VMs < 2, etc
- Query: (Aggregation function, Query attribute, Group predicate)

Responsiveness while handling massive data

- Quick query results even for the most up-to-date state
- Massive data (many attributes over many machines)

Previous Solutions

Centralized DB-based systems

- Collect data to a DB (HP OpenView, IBM Tivoli, …)
- Advantage: very expressive (SQL)
- Disadvantage: can't deliver freshness & scalability at the same time (e.g., every 5 min for CoMon on PL) – gathering all attributes every time

Distributed aggregation systems

- Build end-host aggregation trees (Astrolabe, SDIMS, Seaweed, MON, etc)
- Advantage: scalability with responsiveness
- Disadvantages (next slide)

Previous Solutions

No support for groups

- One group = the entire system
- They all collect data from the entire system

E.g.,

- Just to monitor problematic 3 machines in your PlanetLab slice with 400 machines
- => Need to get data from all 400 machines

Moara Overview

A group-based querying system

- Allows users to target specific groups of machines
- Fast query resolution & expressive group predicates

Group trees

- Moara builds per-group trees & uses these for aggregation
- Different groups can share a tree for scalability.
- Leverages SDIMS (a Pastry-based aggregation framework)

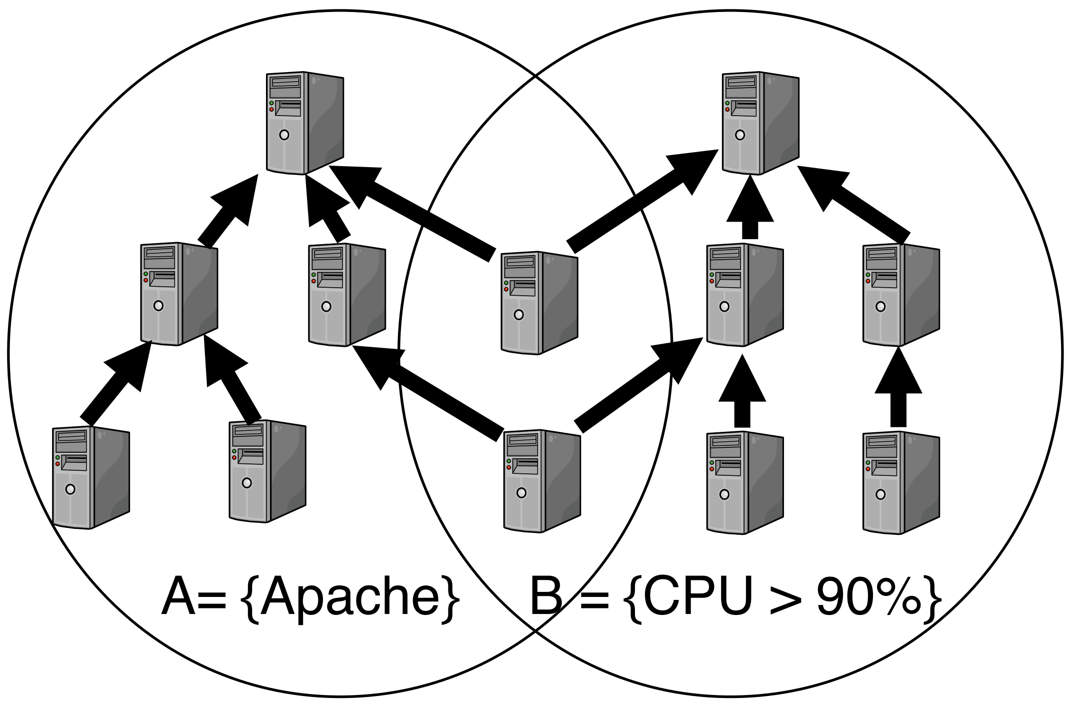
Two optimization techniques

- Multi-group optimization
- Single-group optimization for dynamic groups

Multi-Group Aggregation

Goal

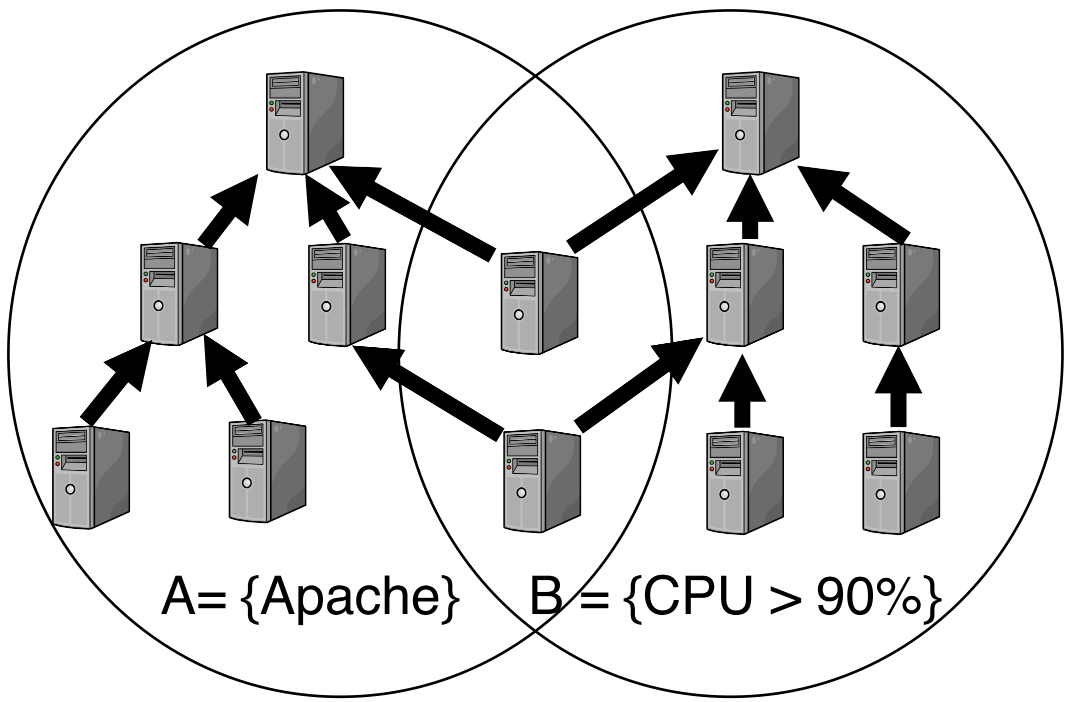
- Quick results with minimum B/W when querying union (*or*) & intersection (*and*) of groups
- E.g., {(*A or B*) *and C*} *or* (*D and E*)?
- Naïve, but correct approach: send a query to every group and collect the data



Multi-Group Aggregation

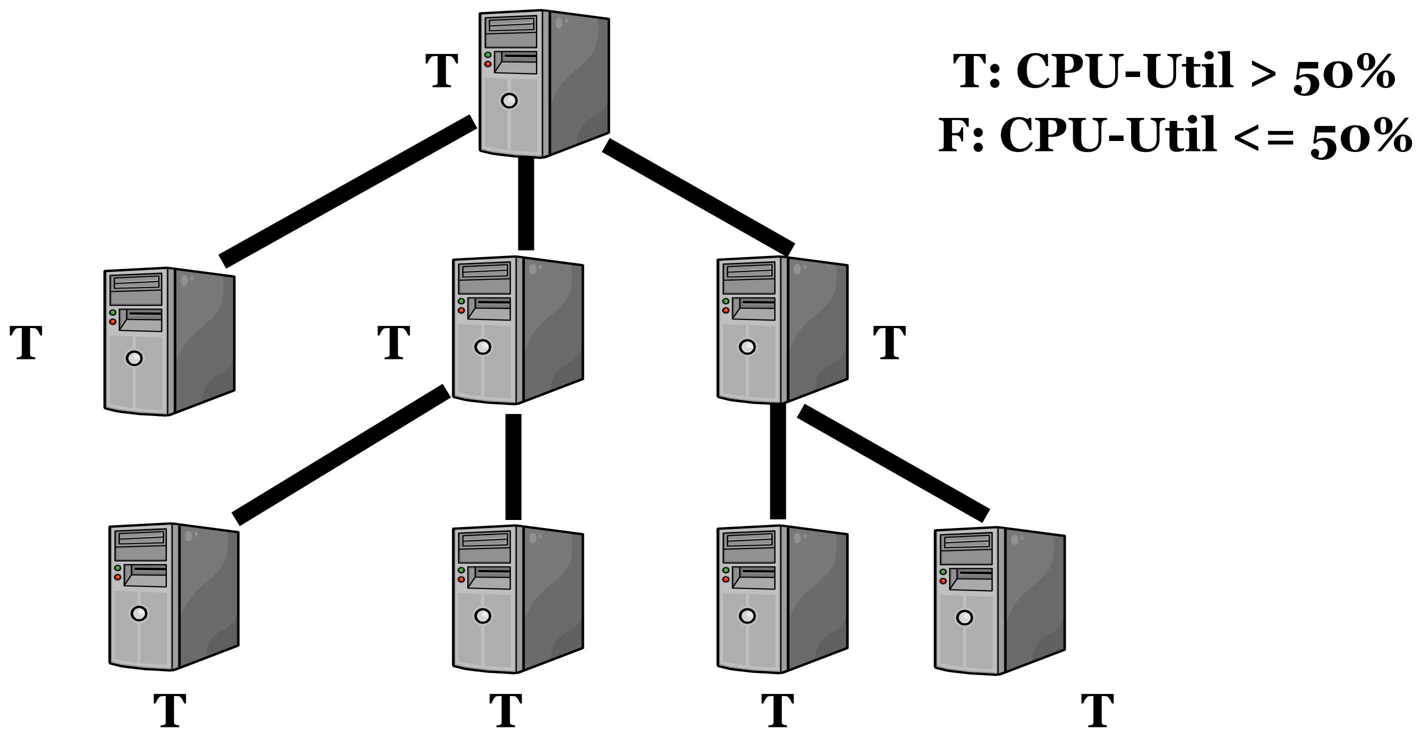
Optimization opportunity for latency and bandwidth

- Querying (*A and B*): need to send queries to only one (smaller) of the two
- What about complex expressions? e.g., {(*A or B*) *and C*} *or* (*D and E*)?
- Solution: CNF transformation & a few more tricks



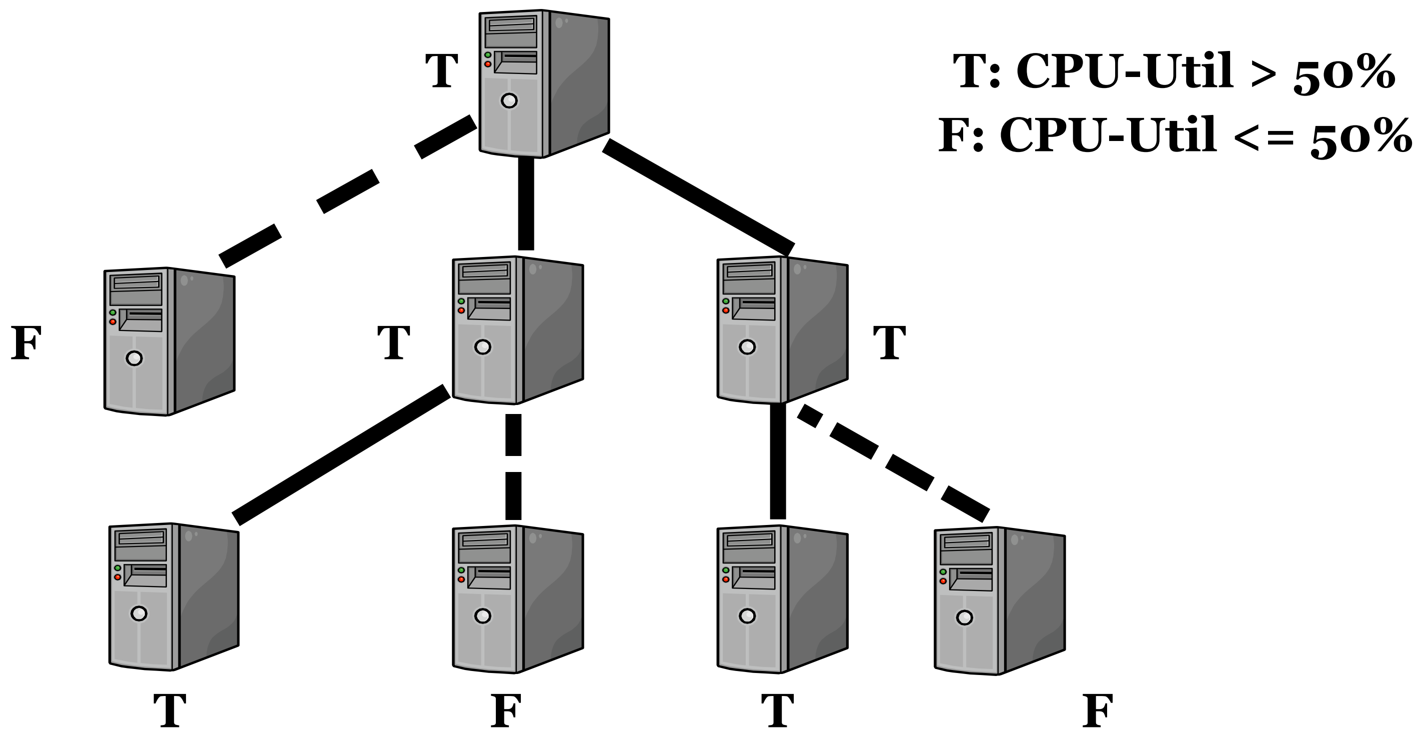
Efficient Dynamic Group Management

Group tree management



Efficient Dynamic Group Management

Group tree management



- Two choices: keep or cut

Efficient Dynamic Group Management

Tradeoff

- Aggressive group tree management: saves per-query cost, but increases management cost
- Lazy group tree management: increases per-query cost, but saves management cost

Adaptive group tree management

- Moara strikes the balance between aggressive and lazy tree management (best of both worlds)
- Adaptively adjust aggressiveness of management by continuously tracking query cost and management cost

Evaluation

PlanetLab: 200 nodes

- Represents wide-area infrastructures

Emulab: 500 instances on 50 machines

- Represents medium-size datacenters

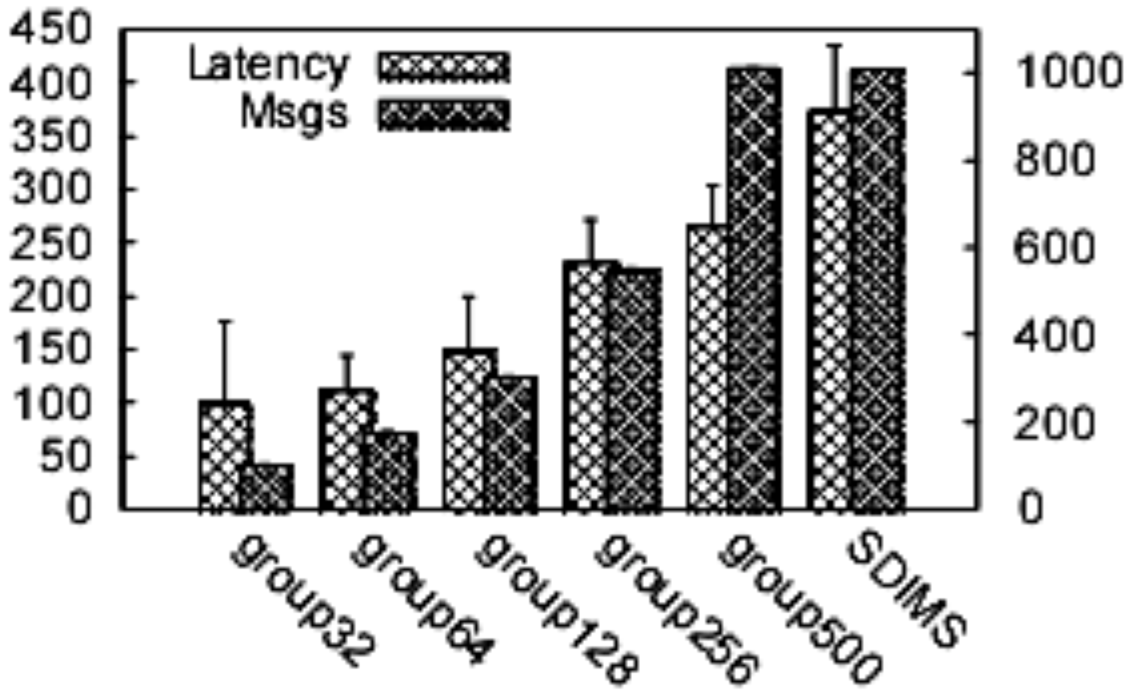
Simulation: up to 10K nodes

- Don't care about latency
- Only measure # of msgs

Emulab

Latency & Bandwidth (static group, 500 instances)

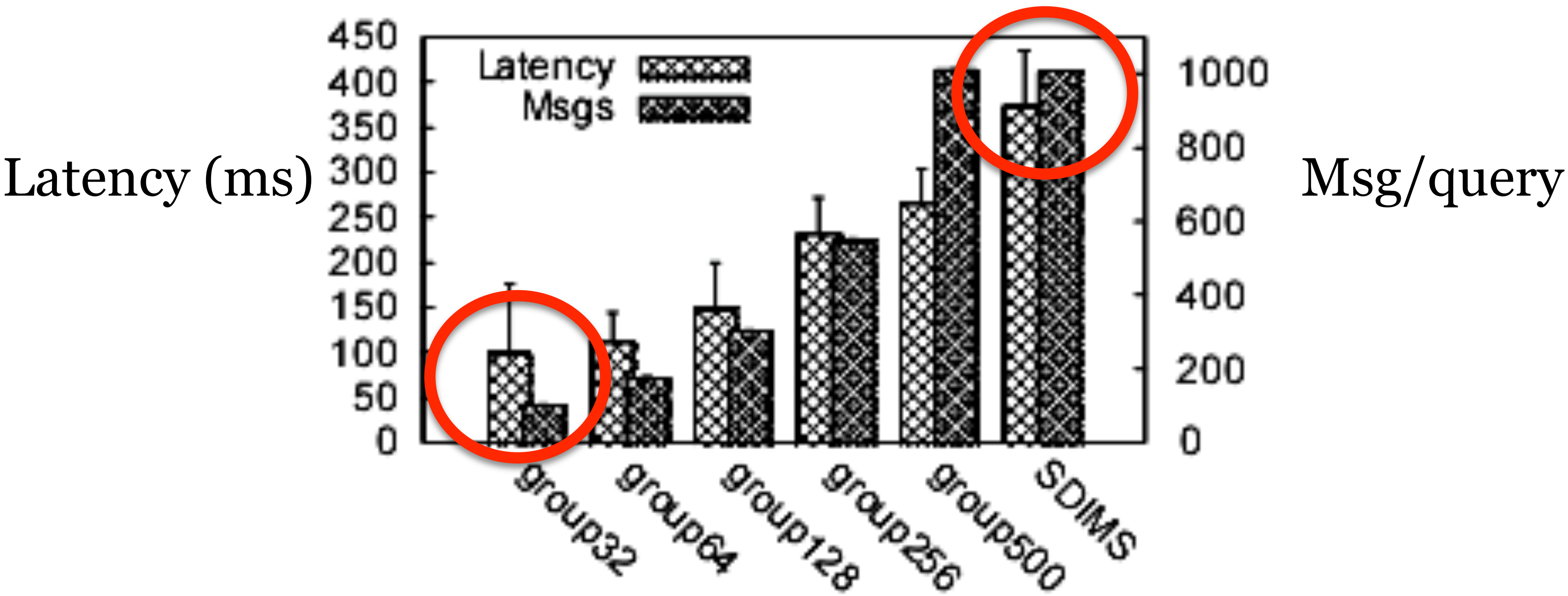
Latency (ms)



Msg/query

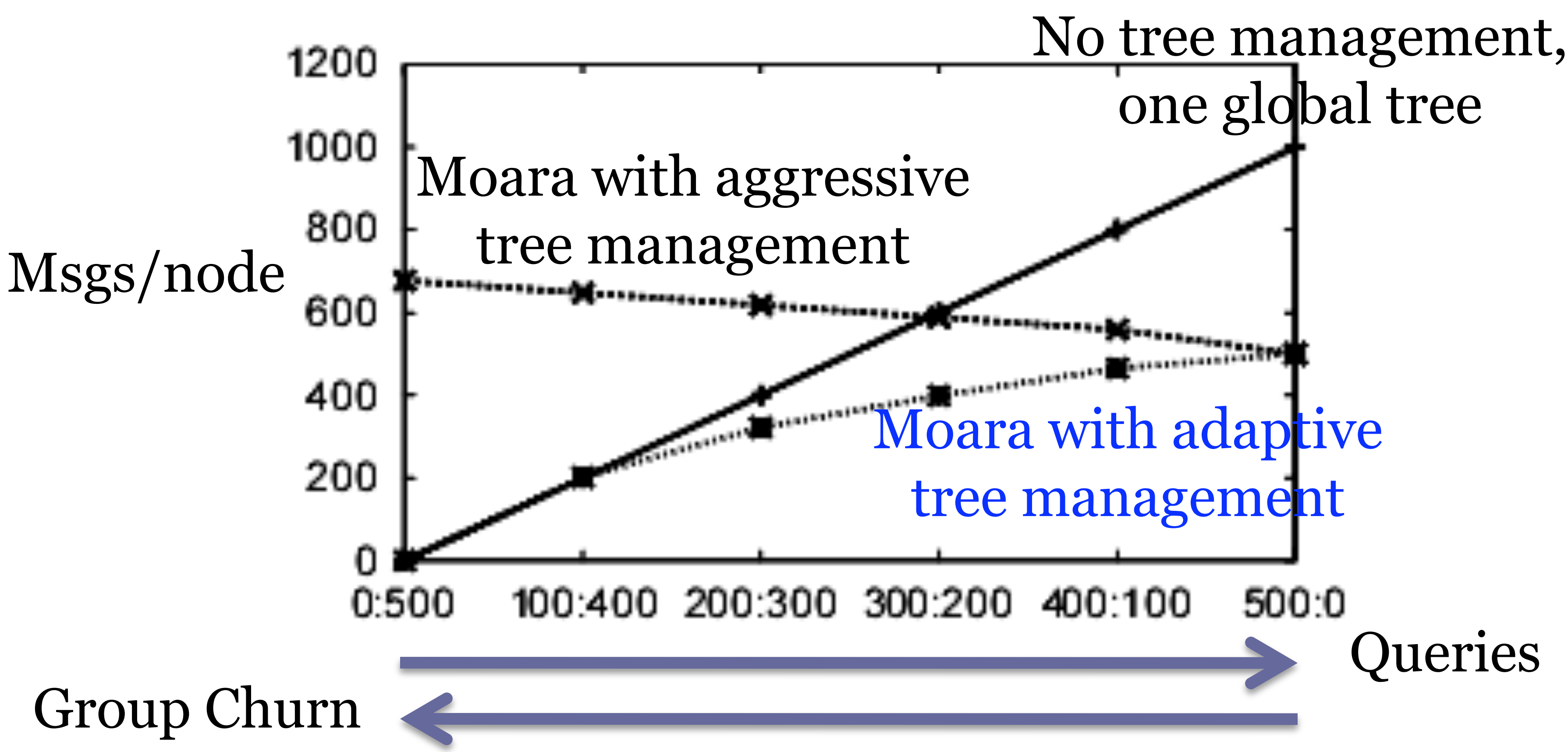
Emulab

Latency & Bandwidth (static group, 500 instances)



Simulation

Adaptive tree mechanism (10K nodes)



Beyond On-Demand Operations

Beyond On-Demand Operations

Data-intensive apps in the Cloud

- Apps access & generate massive amount of data (TB to PB)

Need new solutions to old problems

- E.g., scheduling for data-intensive apps (Hadoop scheduler, worker-centric scheduler – Middleware '07, etc)

New problems

- How to coordinate data management and scheduling
- How to handle intermediate data
  - Data that exists only during the lifetime of an app
  - E.g., data from the Map phase

Beyond On-Demand Operations

Web 2.0 apps in the Cloud

- Problem: responsiveness
- Users expect desktop-quality responsiveness

Best practices

- Make each component as fast as possible
- E.g., multi-layer caching: DB in-memory cache, in-datacenter distributed cache (e.g., memcached), CDN, etc

What's missing? Coordination

- Possibility: Design components that are "aware" of one another.
- To achieve maximum possible performance

Summary

We need to design responsive datacenters that handle massive data.

- Web 2.0 apps in the Cloud
- Data-intensive apps in the Cloud

On-demand operations

- Operations that achieve the goal
- Scale & dynamism are the challenges.