Goal

- A Distributed Operating System Backbone
  - Information collection and management
- Core functionality of large distributed systems
  - Monitor, query and react to changes in the system
  - Examples:
    - System administration and management
    - Sensor monitoring and control
    - Distributed Denial-of-Service attack detection

Benefits

- Ease development of new services
- Facilitate deployment
- Avoid repetition of same task by different services
- Optimize system performance

Outline

- SDIMS: a distributed operating system backbone
- Aggregation abstraction
- Our approach
  - Design
  - Prototype
- Simulation and experimental results
- Conclusions

Contributions – SDIMS

- Provides an important basic building block
  - Information collection and management
- Satisfies key requirements
  - Scalability
    - With both nodes and attributes
  - Leverage Distributed Hash Tables (DHT)
- Flexibility
  - Enable applications to control aggregation
    - Provide flexible API-metil, flush and probe
  - Autonomy
    - Enable administrators to control flow of information
    - Build Autonomous DHTs
- Robustness
  - Handle failures gracefully
  - Perform re-aggregation upon failures
    - Lazy (by default) and On-demand (optional)

Outline

- SDIMS: A Scalable Distributed Information Management System
- Courtesey of Praveen Yalagandula (UT Austin)

Varun Singhal

January 23, 2007
Scalability with nodes and attributes

- To be a basic building block, SDIMS should support:
  - Large number of machines
    - Enterprise and global-scale services
  - Trend: Large number of small devices
  - Applications with a large number of attributes
    - Example: File location system
      - Each file is an attribute
      - Large number of attributes
- Challenges: Build aggregation trees in a scalable way
  - Build multiple trees
  - Single tree for all attributes → load imbalance
  - Ensure small number of children per node in the tree
    - Reduces maximum node stress

Building aggregation trees: Exploit DHTs

- A DHT can be viewed as multiple aggregation trees
- Distributed Hash Tables (DHTs)
  - Each node assigned an ID from large space (160-bit)
  - Routes from all nodes to a particular key form a tree
  - Different trees for different keys

DHT trees as aggregation trees

- Keys 111 and 000

API – Design Goals

- Expose scalable aggregation trees from DHT
- Flexibility: Expose several aggregation mechanisms
  - Attributes with different read-to-write ratios
  - "CPU load" changes often: a write-dominated attribute
  - Aggregate on every write: too much communication cost
  - "NonCPU" changes rarely: a read-dominated attribute
  - Aggregate on reads: streamlining latency
  - Spatial and temporal heterogeneity
  - Non-uniformity and changing read-to-write rates across tree
  - Example: a multicast session with changing membership
- Support sparse attributes of same functionality efficiently
  - Examples: file location, ms/second, etc.
  - Not all nodes are interested in all attributes

Design of Flexible API

- New abstraction: separate attribute type from attribute name
  - Attribute = (attribute type, attribute name)
- Example: type="fileLocation", name="fileFoo"
- Install an aggregation function for a type
  - Amortize installation cost across attributes of same type
  - Arguments up and down control aggregation or update
- Update the value of a particular attribute
  - Aggregation performed according to up and down
  - Aggregation along tree with key=hash(Attribute)
  - Probe: for an aggregated value at some level
  - If required, aggregation done to produce this result
  - Two modes: one-shot and continuous
Install time aggregation and propagation strategy
- Applications can specify *up and down*
- Examples:
  - Update-Local
  - Update-All
  - Update-Up
  - Update-Down

Spatial and temporal heterogeneity
- Applications can exploit continuous mode in probe API
- To propagate aggregate values to only interested nodes
- With expiry time to propagate for a fixed time
- Also implies scalability with sparse attributes

Also implies scalability with sparse attributes

Applications can exploit continuous mode in probe API
- To propagate aggregate values to only interested nodes
- With expiry time to propagate for a fixed time
- Update-All
- Update-Up
- Update-Down

Scalability and Flexibility

Spatial and temporal heterogeneity
- Applications can exploit continuous mode in probe API
- To propagate aggregate values to only interested nodes
- With expiry time to propagate for a fixed time
- Also implies scalability with sparse attributes

Systems spanning multiple administrative domains
- Allow a domain administrator control information flow
- Prevent external observer from observing the interaction in the domain
- Prevent external failure from affecting the operation in the domain
- Support for efficient domain wise aggregation
- DHT trees might not conform
  - Autonomous DHT
  - Two properties:
    - Path Locality
    - Path Convergence
    - Reach domain root first

Autonomy
Outline

- SDIMS: a distributed operating system backbone
- Aggregation abstraction
  - Our approach
    - Leverage Distributed Hash Tables
    - Separate attribute type from name
    - Flexible API
    - Prototype and evaluation

Prototype and Evaluation

- SDIMS prototype
  - Built on top of FreePastry [Druschel et al, Rice U.]
  - Two layers
    - Bottom: Autonomous DHT
    - Top: Aggregation Management Layer

Methodology

- Simulation
  - Scalability
  - Flexibility
- Prototype
  - Aqlon-Benchmarks on real networks
  - PlanetLab
  - CS Department

Simulation Results - Scalability

- Methodology
  - Sparse attributes: multicast sessions with small size membership
  - Node Stress: sum of incoming and outgoing info
- Two points
  - Max Node Stress an order of magnitude less than Astrolabe
  - Max Node Stress decreases as the number of nodes increases

Simulation Results - Flexibility

- Simulation with 4096 nodes
- Attributes with different up and down strategies

Prototype Results

- CS department: 180 machines (283 SDIMS nodes)
- PlanetLab: 70 machines

Conclusions

- SDIMS - basic building block for large-scale distributed services
  - Provides information collection and management
- Our Approach
  - Scalability and Flexibility through
    - Leverage DHTs
    - Separating attribute type from attribute name
  - Providing flexible API: install, update, and probe
  - Autonomy through
    - Building Autonomous DHTs
  - Robustness through
    - Default lazy aggregation
    - Optional on-demand aggregation