Frequently asked questions from the previous class survey

- Can Spark's modules (SQL, MLib) work on the same RDD's in parallel?
- During scheduling, does Spark take into account machines with larger memory?
- Kafka simultaneously write data to Spark streams and a DB?
- Does Spark provide Tuple's in Java for operations such as join?
- Is repartitioning good if you add many machines to your cluster? Are these costs tracked in the industry?
- Does window size impact accuracy?
- What are the intervals at which results are produced?
- Spark Streaming not suitable for hard real-time constraints?
- Receiver receives from multiple sources at different rates, how are the intervals set?
- Can this be dynamic?
- Thread.join() instead of awaitTermination()?
- Does Storm use microbatches?
- Is there additional caching internal to Spark Streaming?
- Does Spark keep a few RDDs in memory? Or is it strictly checkpointing for recovery?

Topics covered in this lecture

- Naming in Distributed Systems
- Name resolution
- Structured naming

Names and entities in distributed systems

- **Name**
  - String of bits or characters used to refer to an entity

- **Entity**
  - Resource: Hosts, printers, disks, files
  - Processes, users, mailboxes
  - Web pages, network connections

Entities can be operated on ...

- Printer interfaces include
  - Printing and status of the print job

- Network connection
  - Send/receive data
  - Set QoS parameters

- Disk
  - List files
  - Read/write to files
But to operate on an entity

- You need to be able to access it
  - Access point
- Access point is a type of entity
- The name of an access point
  - Address

The telephone (landline) analogy

- Telephone
  - Access point for a person
- Telephone number?
  - Address
- There can be multiple access points for an entity
  - Multiple telephones

An address is a special kind of name

- Refers to an access point for an entity
  - Close association
- How about using the address as a regular name for the entity?
  - Let's call Bob; 970-491-2466

The rationale for separating access points from entities: Server example

- Machine on which a server was running
  - May be assigned to a different server
- Entity may change an access point
  - Access point may be assigned to a different entity

Multiple access points: A service is often distributed across multiple servers

- If we use address of server as reference for service
  - Which address would you use?
- SOLUTION: Single name for the service
  - Independent of the server addresses

Identifier

- Type of name
- Used to *uniquely identify* an entity
Properties of a true identifier
- Refers to at most one entity
- Each entity is referred to by at most one identifier
- An identifier always refers to the same entity
  - Never reused

Identifiers allow us to unambiguously refer to an entity
- Check if two processes are referring to the same entity
  - Simple equality test
- Addresses may be reassigned to a different entity
  - Telephone numbers are reassigned
  - Not so with identifiers

Some more about identifiers and addresses
- Represented in machine-readable form
  - Ethernet addresses: 48 bits
  - Memory addresses: 32-bit or 64-bit string
  - UUIDs: 128-bits
- Very tedious to remember

Human friendly names
- Represented as character strings
- Easy to remember
  - Files in UNIX: 255 characters, case-sensitive
  - DNS names: Case-insensitive character strings

Central problem in Naming schemes
- How do we resolve names and identifiers to addresses?
- Simple solution:
  - Centralized table of (name, address) tuples
  - Not scalable

NAME RESOLUTION
Central problem in Naming schemes

- How do we resolve names and identifiers to addresses?
- Simple solution:
  - Centralized table of \{name, address\} tuples
  - Not scalable

Name resolution often happens in a distributed fashion

- Name is decomposed into parts
- Name resolution by looking up those parts
  - Recursively
  - Iteratively

Naming Approaches

- Flat naming
  - Random bit strings
    - E.g. identifiers
  - Contains no information about how to locate the access point

Flat naming

- Random bit strings
  - E.g. identifiers
  - Contains no information about how to locate the access point

Using broadcast to resolve names

1. Message created with identity
2. Message is broadcast to each machine
3. Each machine checks if it has entity
4. Only machine that offers access point for entity responds
   - With address of the access point

Broadcast is used in the Internet Address Resolution Protocol

- Find the data-link address of a machine when given only the IP address
- Broadcast packet on local network
  - Message: Who is the owner of a given IP?
- Receivers check whether they should listen to the requested IP address
- Valid receiver responds with Ethernet address
Problems with broadcast based name resolution

- **Inefficient** when the network grows
- Poor bandwidth utilization
- Too many hosts interrupted
  - By requests that they cannot answer!

Multicast can minimize some of the problems with broadcasts

- **Restricted group** of hosts receive the multicast
- Locating a mobile computer A
  - A is assigned a dynamic address (DHCP)
  - A joins a multicast group G
  - Process interested in IP_A sends “IP of A?” to G
  - If A is connected, it responds with IP_A

Locating mobile addresses: Forwarding pointers

- When an entity moves from A to B
  - Leave forwarding pointer in A
  - To new location at B
- Lookup current address by following chain of forwarding pointers

Problems with forwarding pointers?

- Chain for a very mobile entity can be long
  - Locating the entity becomes very expensive
- All intermediate locations must maintain their parts for as long as needed
- Vulnerable to broken link

Coping with issues in forwarding pointers

- Keep chains relatively short
- Ensure that forwarding pointers are robust

We previously looked at Mobile IPv6

- A mobile node has a home-network
  - This node has a home-address
  - The node has a home agent
    - Takes care of traffic to the mobile node while it is away
Forwarding addresses in Mobile IPv6
- When a mobile node attaches to a foreign network
  - Gets a temporary care-of address
- Care-of address reported to the home-agent
  - Home-agent forwards all traffic to the mobile node

Problems with the home-based approach
- Home location is fixed and must always exist
  - Contacting entity is not possible otherwise
- What if a long-lived entity decides to move permanently?
  - Shouldn’t its home also move?

A possible solution to migration problem
- Register home at a naming service
- Client should first discover the location of home

Hierarchical location scheme
- Network is divided into a collection of domains
  - Subdivided into multiple, smaller domains
- Single top-level domain spans entire network
- Lowest domain is called the leaf domain
  - LAN in a computer network
  - Cell in a Mobile network

Hierarchical organization of location service

Tracking the whereabouts of an entity
- Each entity located in domain D represented by a location record
  - Directory node dir(D)
- Trickled upwards to higher domains
  - Pointers point to the directory node in the lower domain from whence the record came
Managing entries in hierarchical organizations

Looking up an entity in a hierarchically organized service

Hierarchical location searches exploit locality

Structured naming

In structured naming, the names are organized into a namespace
**The leaf node**
- Represents a named entity
- Has **no outgoing edges**
- Stores information (e.g., address) about the entity it represents
- Can also store state of the entity
  - E.g., files

**Directory Node**
- Has a **number** of outgoing edges
  - Each edge has a name
  - Each node has an associated identifier
- Stores a **directory table**
  - Each outgoing edge is represented
    - (edge label, identifier)

---

**An example naming graph**

```
<table>
<thead>
<tr>
<th>Node</th>
<th>Data Stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>n0</td>
<td>home</td>
</tr>
<tr>
<td>n1</td>
<td>alice, bob, eve</td>
</tr>
<tr>
<td>n2</td>
<td>alice</td>
</tr>
<tr>
<td>n3</td>
<td>bob</td>
</tr>
<tr>
<td>n4</td>
<td>eve</td>
</tr>
<tr>
<td>n5</td>
<td>keys</td>
</tr>
<tr>
<td>n6</td>
<td>/home/eve/keys</td>
</tr>
</tbody>
</table>
```

**Paths in a naming graph are a sequence of labels corresponding to the edges**

- \( N: \langle \text{label}_1, \text{label}_2, ..., \text{label}_n \rangle \)
- \( N \) first node in the path
- **Path name**
  - if (first node in path == root) {
    - **Absolute** path name
  } else {
    - **Relative** path name
  }

---

**In file systems, edge labels are separated by a “/”**

- Instead of using \( n0: \langle \text{home, eve, mbox} \rangle \)
- Use: /home/eve/mbox

**Name resolution is the process of looking up a name**

- \( N: \langle \text{label}_1, \text{label}_2, ..., \text{label}_n \rangle \)
- Resolution starts at node \( N \)
  - label\(_1\) is looked up in the directory table
    - Returns identifier of node that label\(_1\) refers to
  - Resolution **continues** at the identified node
    - Lookup label\(_2\)
For name resolution to take place we must know when and where to start

- This is called the **closure mechanism**
  - Partly implicit
  - Difficult to compare different mechanisms
- 0019704912466
  - Resolution process starts by dialing the number
  - Telephone switch network does the rest

Resolving a file name requires some mechanism where the process can start

- OS hard codes information about the organization of the **superblock**
  - **inode** of the root directory is the first node of the logical disk
  - Computed from values in other fields of the superblock

**Alias** is another name for the same entity

- Environment variables
- Naming graphs
  - Hard links
  - Symbolic links

**Hard links**: Node n5 can be referred to by two path names

```
Data Stored in n1
n2: alice
n3: bob
n4: eve

Data Stored in n6
/home/eve/keys
```

```
/home/eve/keys
```

**Symbolic links**: Node stores an absolute path name that leads to another node

```
/home/eve/keys
```

```
/home/eve/keys
```

**A simple distributed name space**

```
/home/eve/keys
```

```
/home/eve/keys
```
The contents of this slide-set are based on the following references: