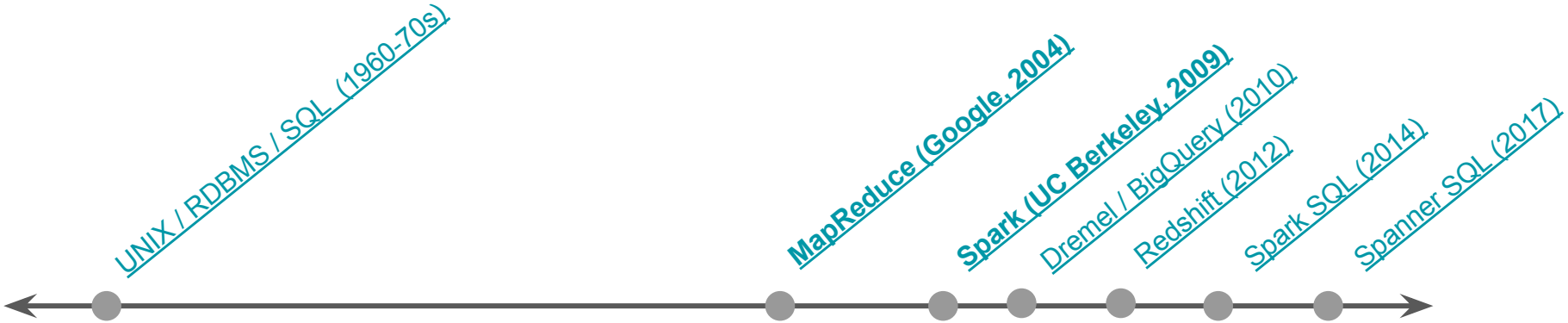
# CSC 369

## Introduction to Distributed Computing

# An Abridged Time~~line~~*loop*



# UNIX Philosophy

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new “features.”

- Expect the output of every program to become the input to another, as yet unknown, program.

Doug McIlroy, Elliot Pinson and Berk Tague, 1978

# UNIX Pipes

“We should have some ways of coupling programs like a garden hose — screw in another segment when it becomes necessary to massage data in another way.”

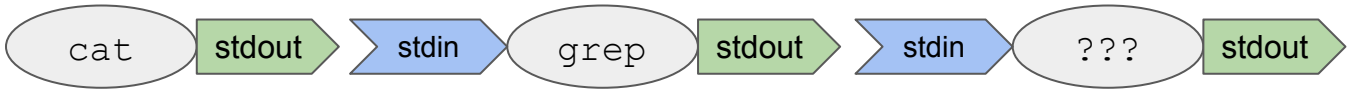
M. D. McIlroy

October 11, 1964

http://doc.cat-v.org/unix/pipes/

# Composing UNIX Tools

```
cat | grep | ??? …
```




UNIX Text Processing Tools (Linux Documentation Project)

GNU coreutils Documentation

# HTTP Access Log (Apache Format)

```
10.0.0.153 - - [10/Mar/2004:15:10:10 -0800]
"GET /robots.txt HTTP/1.0" 200 68
```

1. Client IP / Hostname
2. Client ID
3. Username
4. Timestamp
5. HTTP Request
6. HTTP Status Code
7. Response Size (bytes)

(based on the default Apache 2.4 format)

# Find Top Visitors

```
awk '{print $1}' access.log | # Split by whitespace, extract client address (field #1)
  sort     | # Occurrences of the same client address will appear consecutively
  uniq -c  | # Collapse repeated client address into a count
  sort -nr | # Sort (numeric), in reverse (descending) order
  head -n 10 # Show top 10 client addresses (preceded by count of log lines)
```

Example adapted from: [Unix philosophy of distributed data, Martin Kleppmann]

# Find Top Visitors: SQL Edition

```
awk '{print $1}' access.log | # Split by whitespace, extract client address (field #1)
  sort     | # Occurrences of the same client address will appear consecutively
  uniq -c  | # Collapse repeated client address into a count
  sort -nr | # Sort (numeric), in reverse (descending) order
  head -n 10 # Show top 10 client addresses (preceded by count of log lines)
```

```
SELECT COUNT(*) AS c, client_address
FROM access_log
GROUP BY client_address
ORDER BY c DESC
LIMIT 10
```

# UNIX Utilities and Pipes

## The Good

- Each utility designed for one specific purpose
- Composable, standard interface
- Immutability (`grep` does not modify file contents)
- Simplicity output

## The Not-So-Good

- Single-node
- Single input / single output
- Input/output format & structure (parsing, escaping, etc.)
- Lack of fault tolerance

can't recover when an operation is failed. what happened if one operation take 5 hours?

# Limiting Factor: CPU or Data

## CPU-Intensive

CPU power is the limiting factor, for example:

- Video processing
- Graphics / Rendering
- Cryptography

## Data-Intensive

Primary challenge is not CPU power, but the amount of data, complexity of data, or speed of data change.

- Large online systems
- Bio/Physics/Chem

Other Examples?

# Why Distributed Data?

- Scalability
  - Data volume, read/write load beyond what a single machine can handle
  - Cost of vertical scaling is not linear: 2x CPU, RAM > 2x $
- High availability, fault tolerance
- Latency
  - Geographically distributed clients

# Shared Nothing Architecture

- Each server or virtual machine (*node*) has independent CPU, RAM, and disk
- Coordination occurs over conventional network
- Individual machines can be configured for best price/performance, located in multiple geographic locations

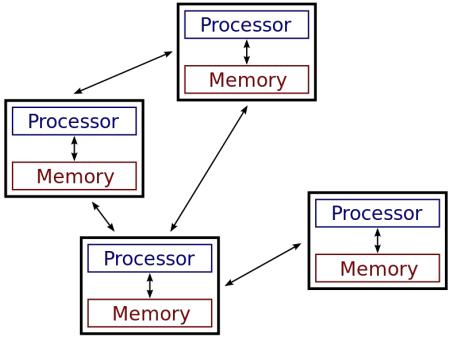



Image from Wikipedia

# Shared Nothing (What About the Data?)

Two approaches to distributing data across nodes:

- Replication
  - Keep an exact copy of data on multiple nodes
- Partitioning
  - Split a large dataset into smaller subsets, store each subset on a different node

Separate mechanisms, but often used together