Introduction to Data Structure (Data Management) Lecture 10

Felipe P. Vista IV

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - University ID Num Name (no “( )“)
  - Ex: 202054321 Juan Dela Cruz
  -
  - Not changing your name to this format
    - you might be marked Absent

    * Ravshan → absent?

- NoSQL

- JSon and Semi-tructured Data

INTRO TO DATA STRUCTURE

NOSQL
(CH 11.1)

Motivation for NoSQL

- Motivated by Web 2.0 Applications
  - Web 2.0 allow anyone to create and share online information or material
  - Key element is allow people to create, share, collaborate & communicate
  - Hosted services (Google Maps), Web Apps (Google Docs, Flickr), vid sharing sites(YouTube), wikis, blogs, SNS(FB,IG), microblogging(Twitter)

Motivation for NoSQL

- Goal is to scale simple OLTP-style applications to millions or even billions of users
- OLTP (OnLine Transaction Processing)
  - capture, store, process data from transactions in real-time
  - typical size range from 100MB to 10GB
  - Ex: online banking, purchasing book online, booking ticket, send text message, call center staff view/update customer info

Motivation for NoSQL

- Facebook has 1.79B active users daily (Q2 2020)
  - use often correlated in time in each region
    - *correlated : one thing affects or depends on another*
  - more than 10M requests/sec if 25% users arrive w/in hour
  - SQL Server would crash under this workload
- Users doing both reads and updates

What is Problem?

- Single server DBMS too small for Web data

What is Problem?

- Single server DBMS too small for Web data
  - Solution→ scale out to multiple servers
    - *scale: resize a device, object or system*
    - *"scale up" or "scale vertically: expanding capability of a machine*
    - *"scale out" or ""scale horizontally": add more machines*

What is Problem?

- Single server DBMS too small for Web data
  - Solution→ scale out to multiple servers
    - *scale: resize a device, object or system*
    - *"scale up" or "scale vertically: expanding capability of a machine*
    - *"scale out" or ""scale horizontally": add more machines*
- This is hard for ***entire*** functionality of DBMS

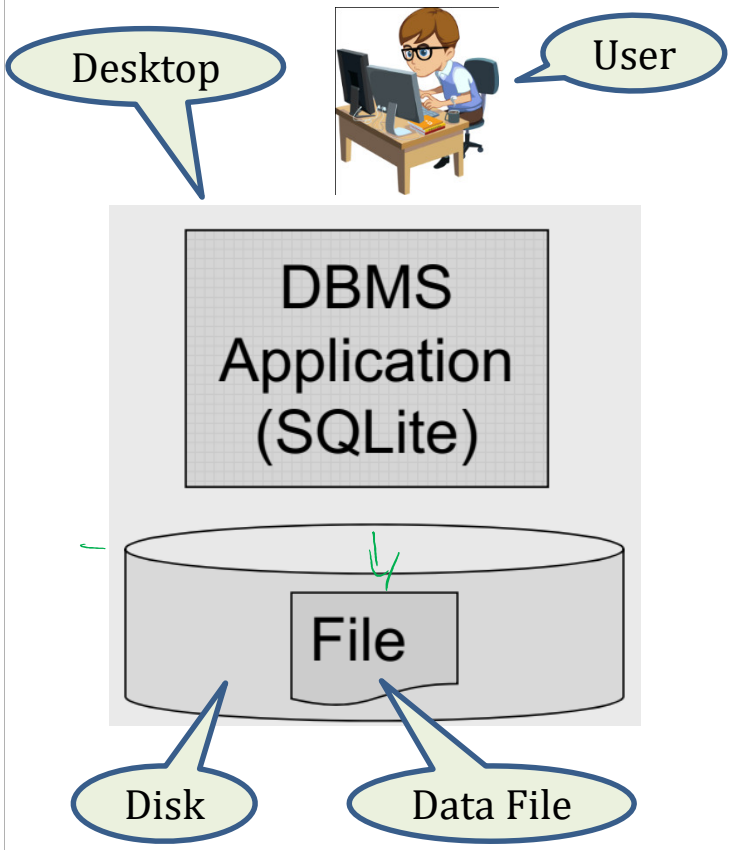
What is Problem?

- Single server DBMS too small for Web data
  - Solution→ scale out to multiple servers
    - *scale: resize a device, object or system*
    - *"scale up" or "scale vertically: expanding capability of a machine*
    - *"scale out" or ""scale horizontally": add more machines*
- This is hard for ***entire*** functionality of DBMS
- NoSQL: reduce functionality for easier scaling

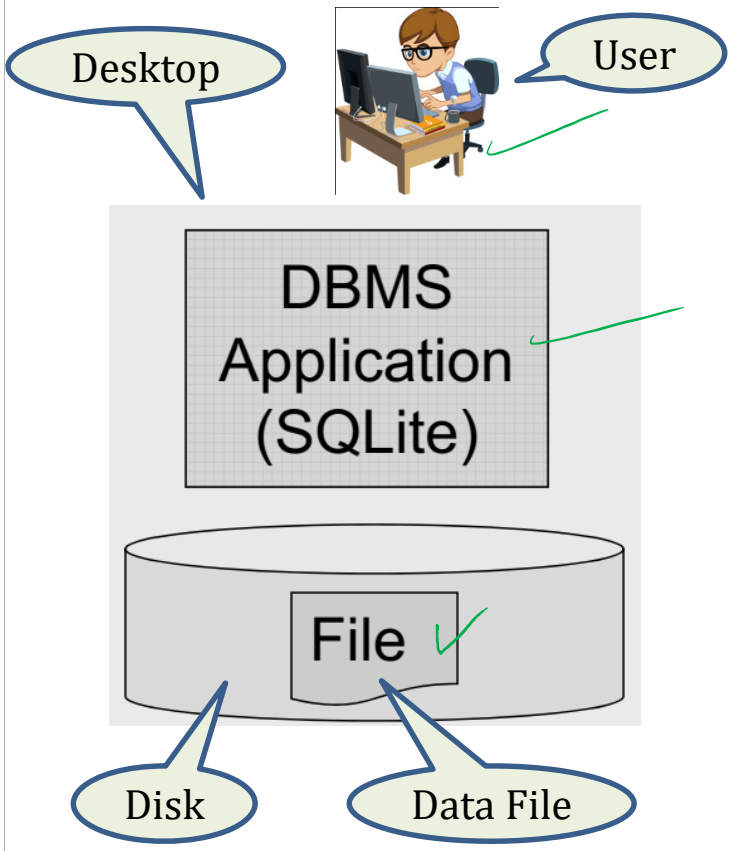
What is Problem?

- Single server DBMS too small for Web data
  - Solution→ scale out to multiple servers
    - *scale: resize a device, object or system*
    - *"scale up" or "scale vertically: expanding capability of a machine*
    - *"scale out" or ""scale horizontally": add more machines*
- This is hard for ***entire*** functionality of DBMS
- NoSQL: reduce functionality for easier scaling
  - simpler data model
  - fewer guarantees

Serverless Architecture



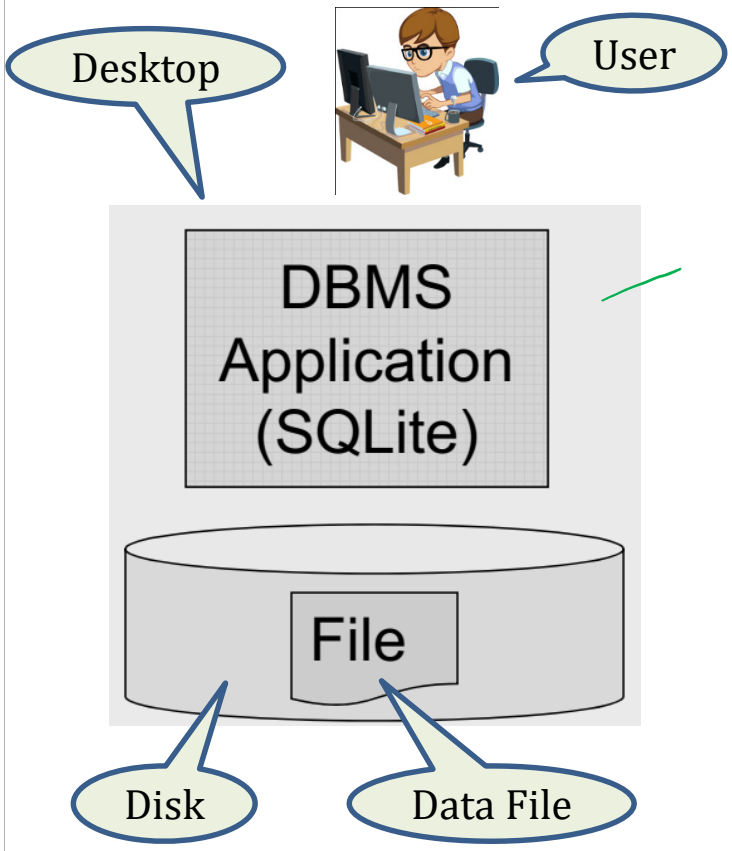
Serverless Architecture



SQLite

- One data file
- One user
- One DBMS application

Serverless Architecture

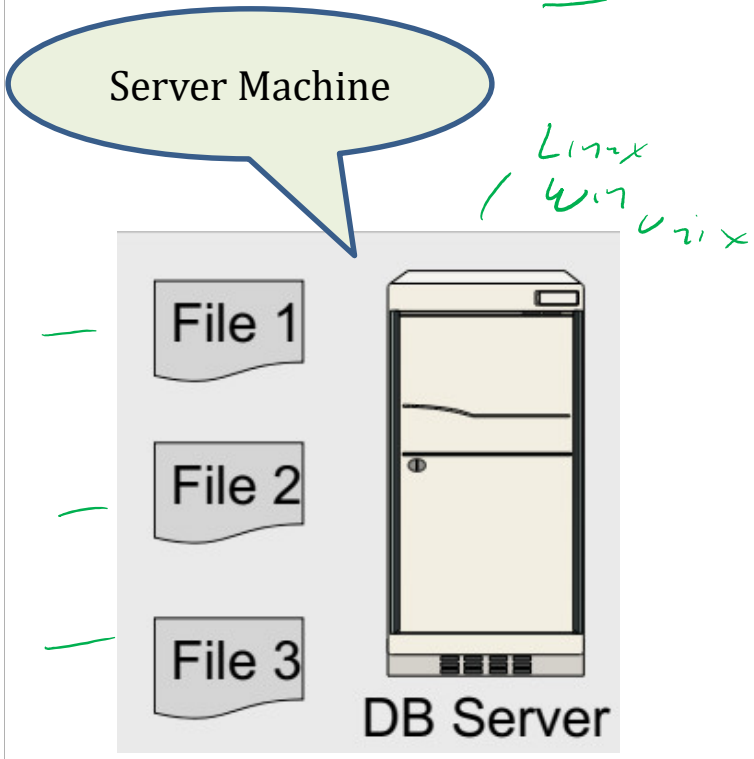


SQLite

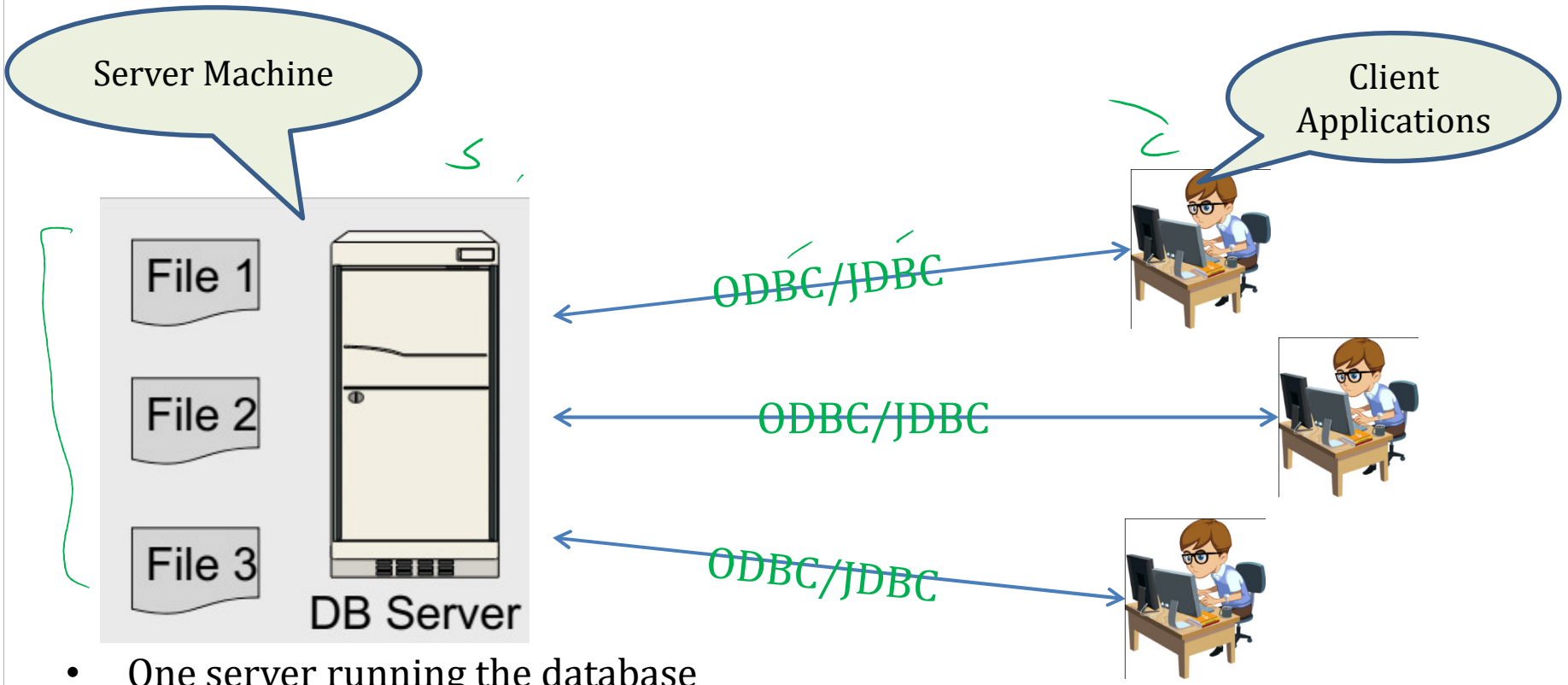
- One data file
- One user
- One DBMS application

- Scales well
- But only a limited number of scenarios work with such model
- Can be in browser/ phone

Client-Server Architecture

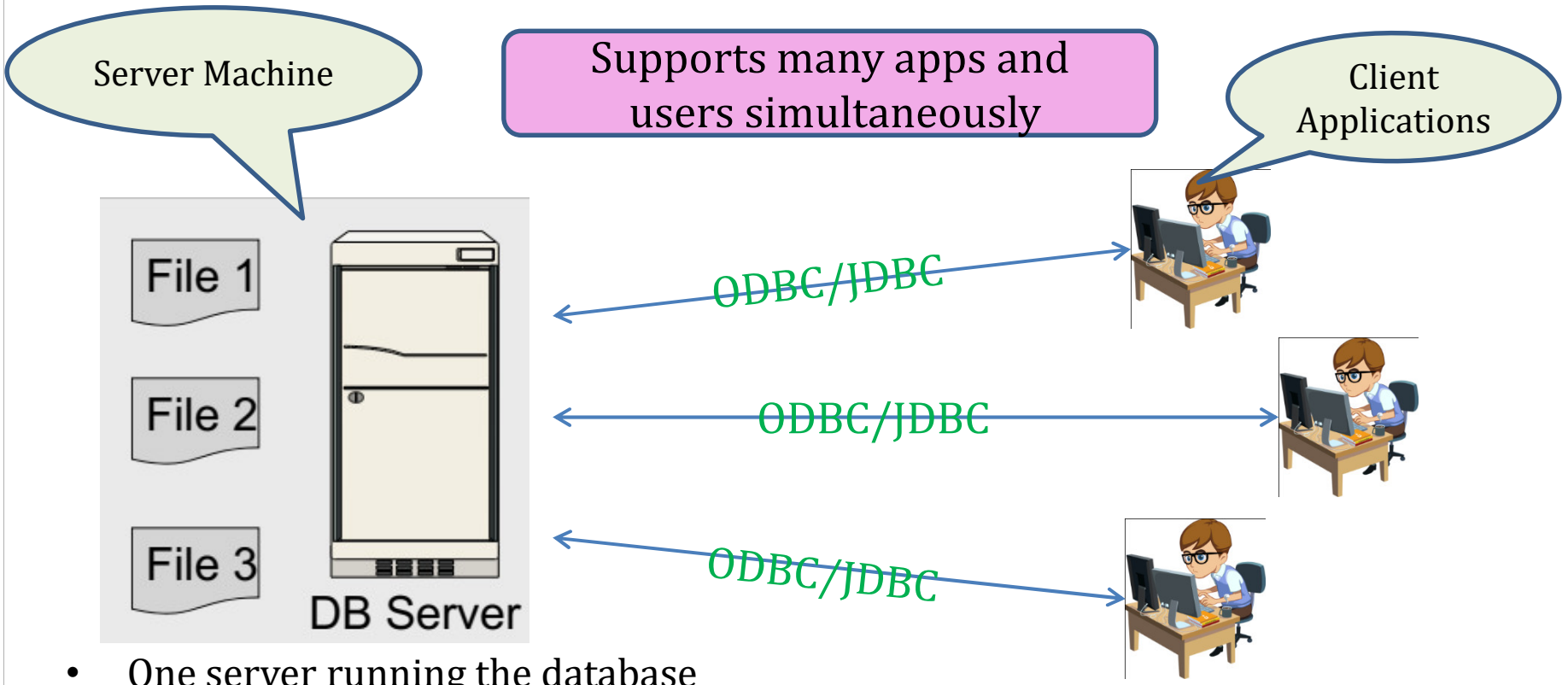


Client-Server Architecture



- One server running the database
- Many clients, connecting via ODBC (Open DB Connectivity / JDBC (Java DB Connectivity

Client-Server Architecture



- One server running the database
- Many clients, connecting via ODBC (Open DB Connectivity / JDBC (Java DB Connectivity

Client-Server

- One ***server*** runs DBMS (or RDBMS)
  - on own desktop
  - some beefy/powerful system
  - cloud service

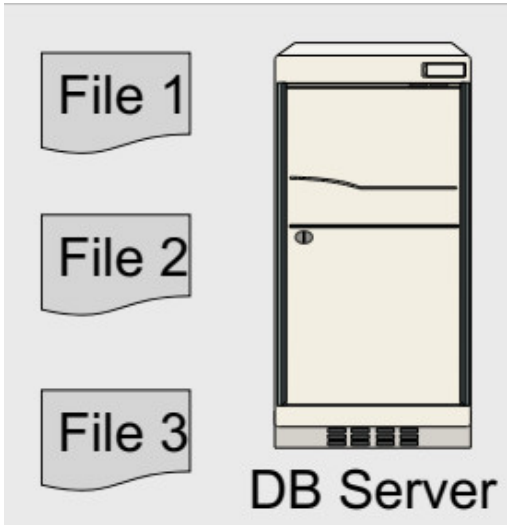
Client-Server

- One ***server*** runs DBMS (or RDBMS)
  - on own desktop
  - some beefy/powerful system
  - cloud service
- Many ***clients*** run apps and connect to DBMS
  - MS Management Studio (for SQL Server) or
  - pSQL (for postgres)
  - some Java program or C++ program

Client-Server

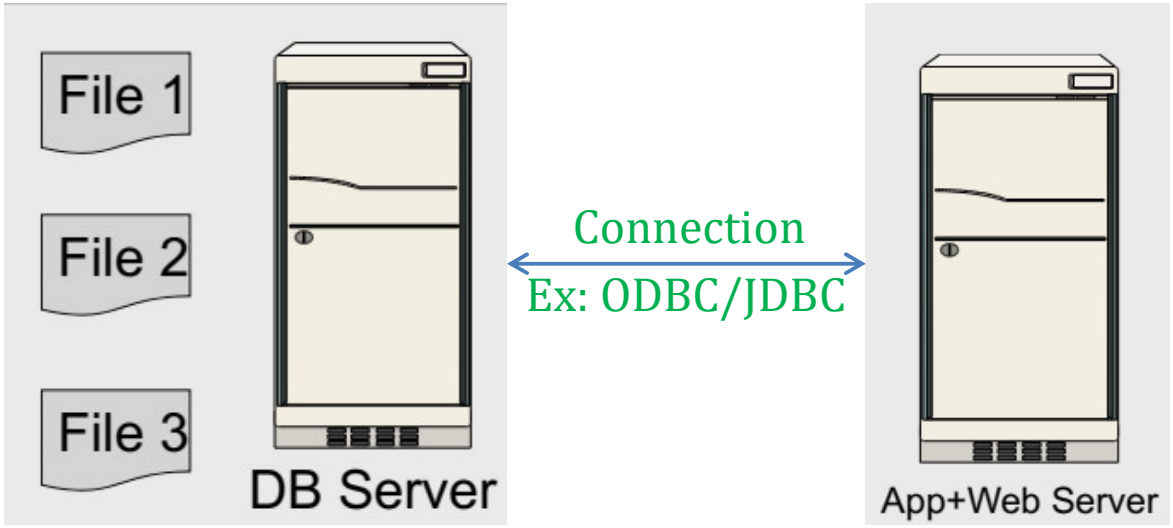
- One ***server*** runs DBMS (or RDBMS)
  - on own desktop
  - some beefy/powerful system
  - cloud service
- Many ***clients*** run apps and connect to DBMS
  - MS Management Studio (for SQL Server) or
  - pSQL (for postgres)
  - some Java program or C++ program
- Clients "talk" to server using ODBC/JDBC protocol

3-Tiered Architecture



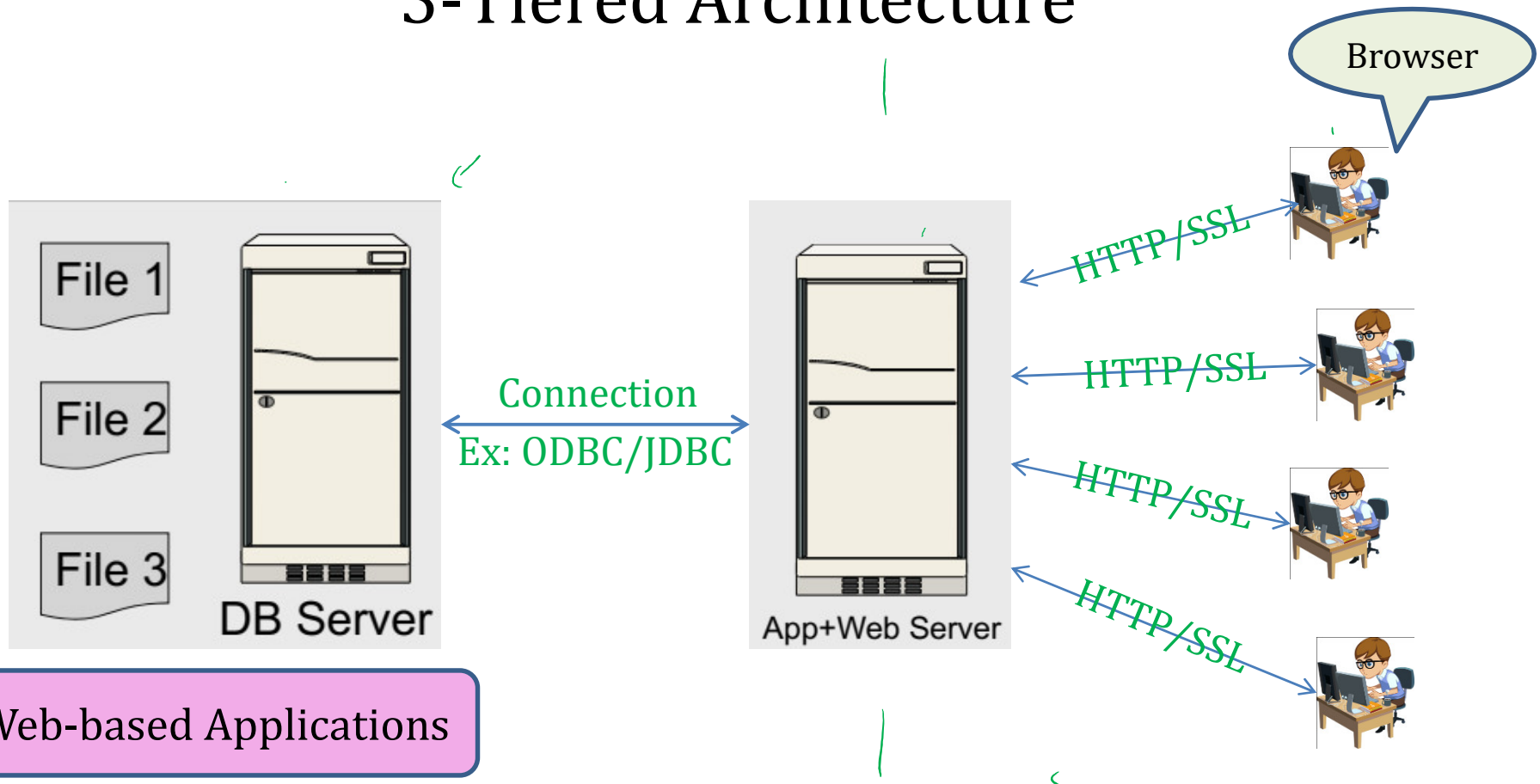
Web-based Applications

3-Tiered Architecture



Web-based Applications

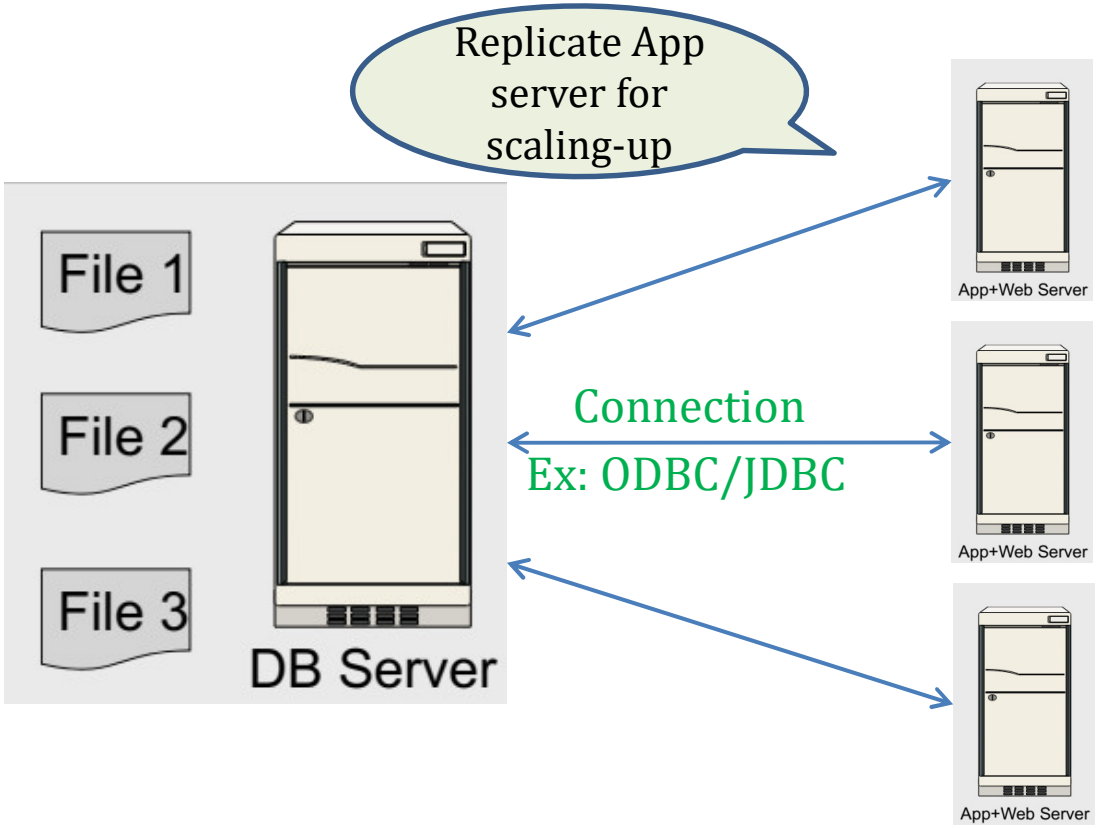
3-Tiered Architecture



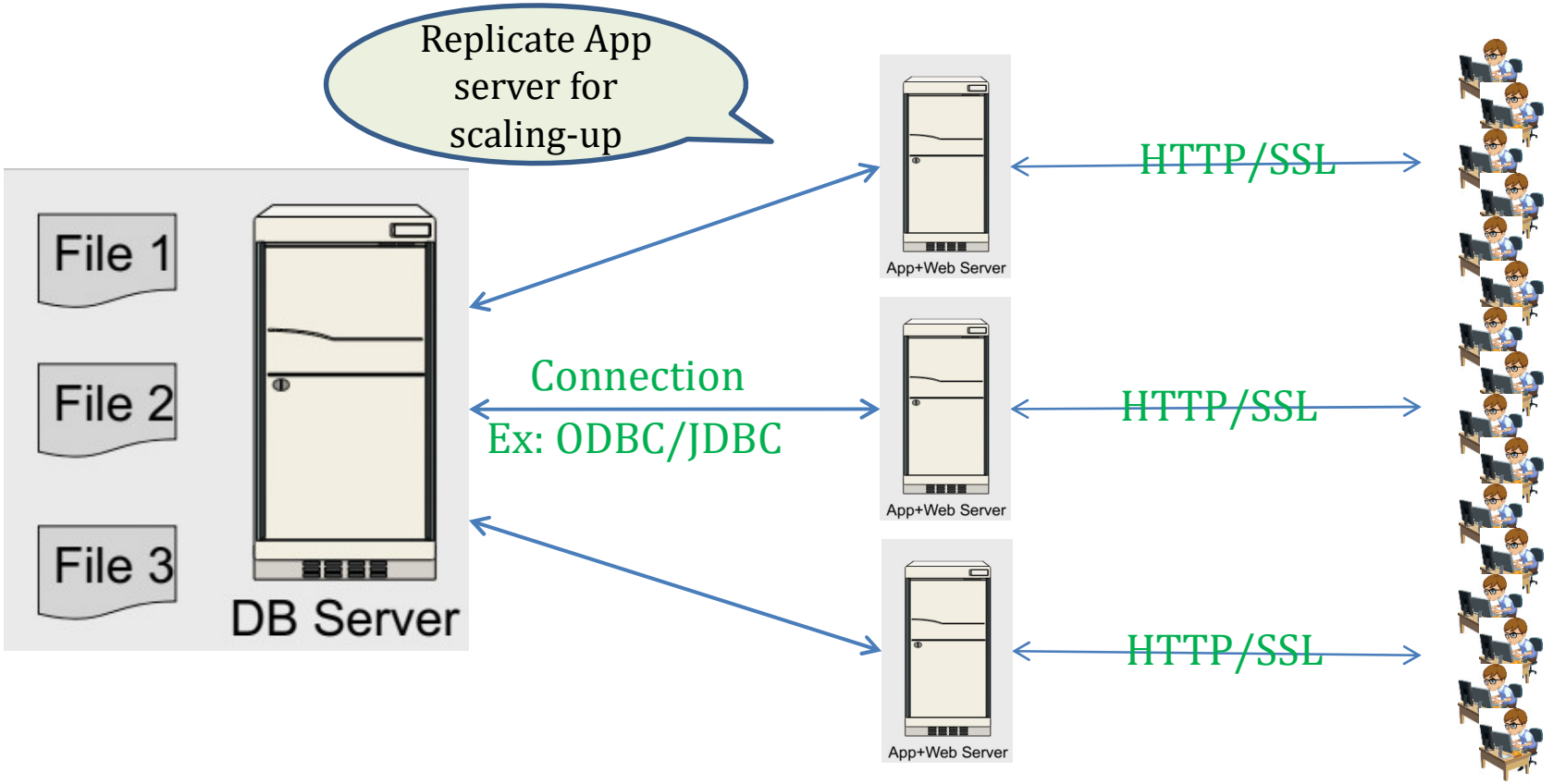
Web-based Applications

* HTTP = Hyper Text Transfer Protocol
* SSL = Secure Socket Layer

3-Tiered Architecture

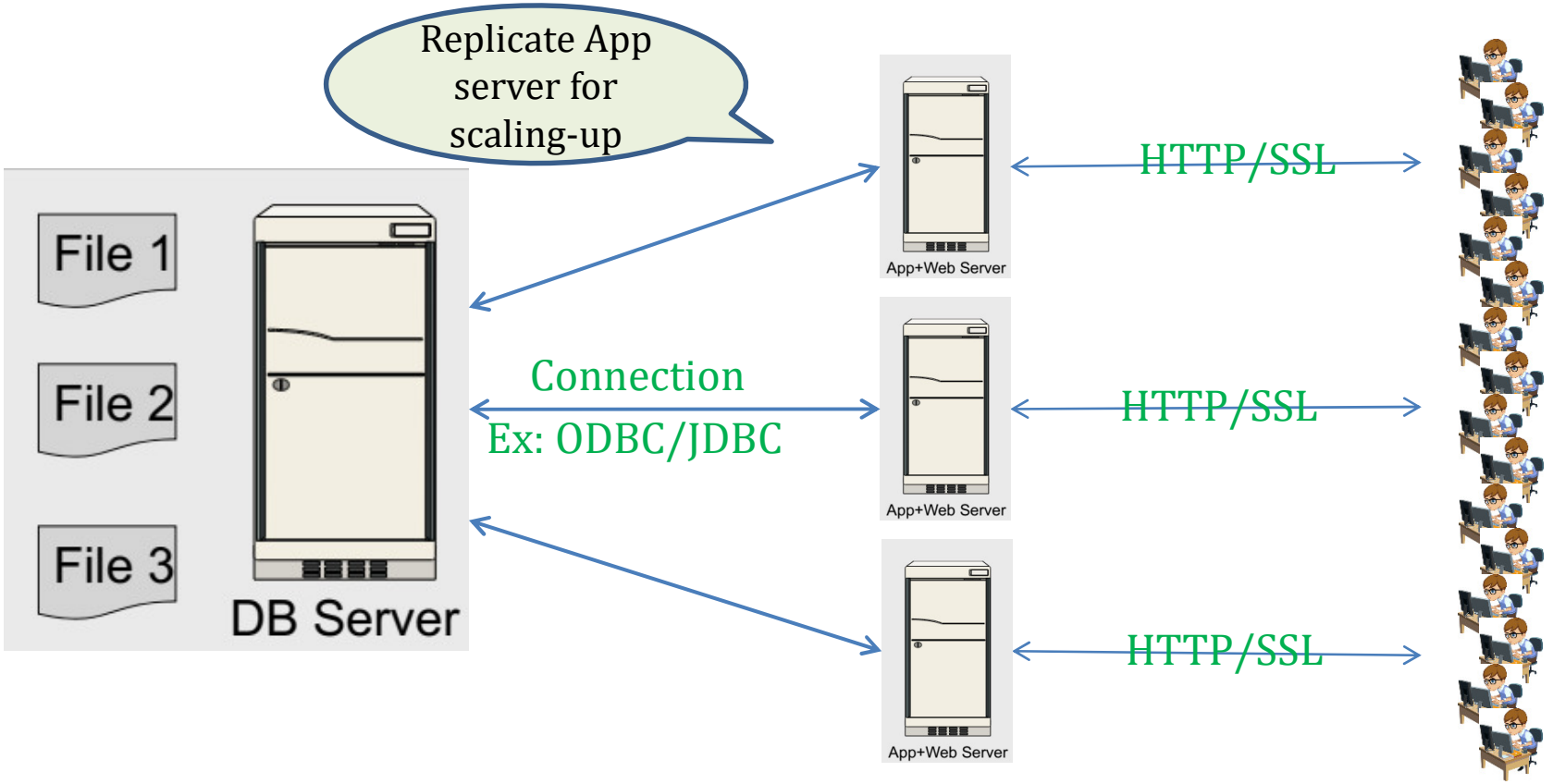


3-Tiered Architecture



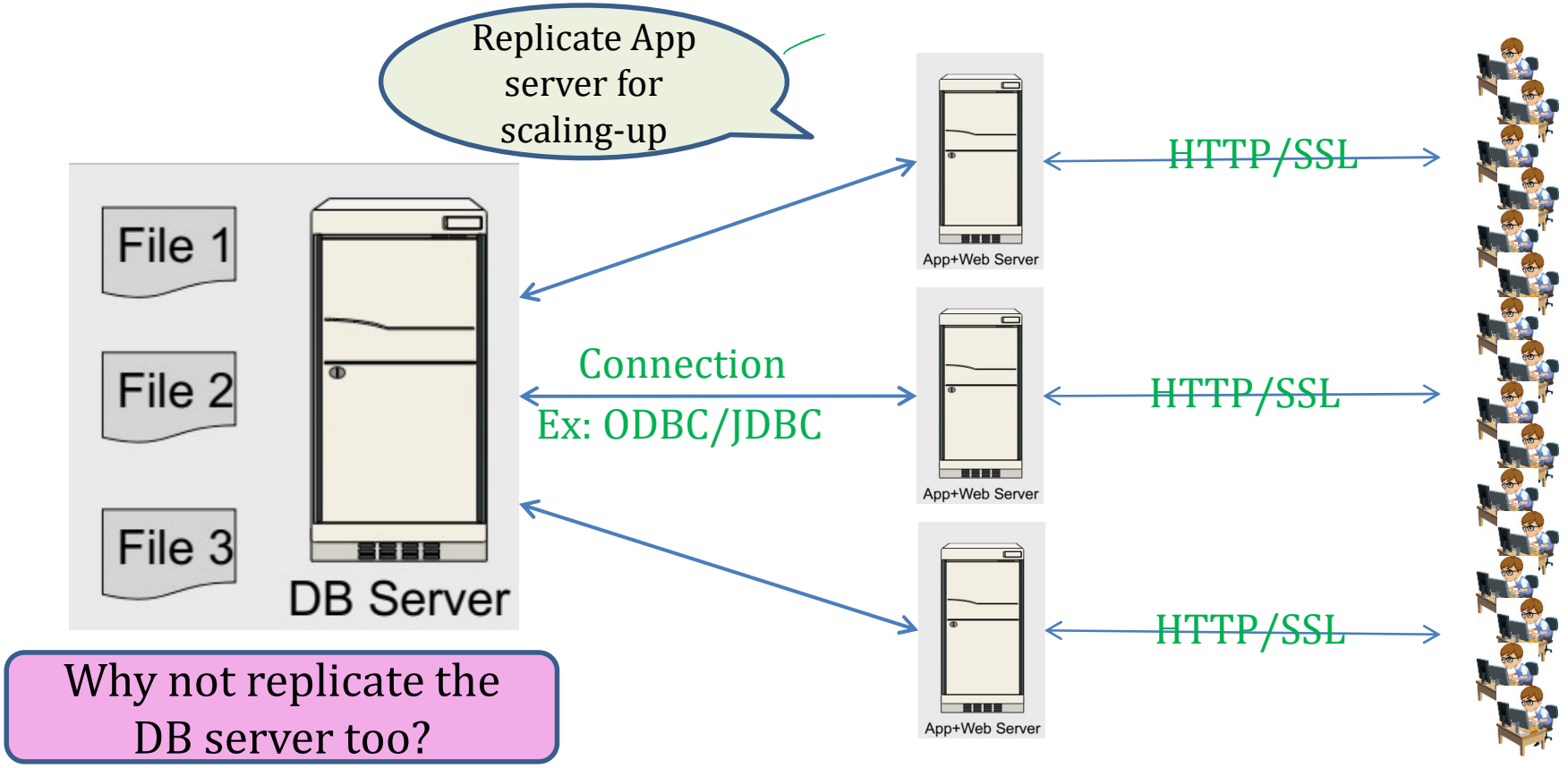
* HTTP = Hyper Text Transfer Protocol
* SSL = Secure Socket Layer

3-Tiered Architecture



* HTTP = Hyper Text Transfer Protocol
* SSL = Secure Socket Layer

3-Tiered Architecture



Why not replicate the DB server too?

* HTTP = Hyper Text Transfer Protocol
* SSL = Secure Socket Layer

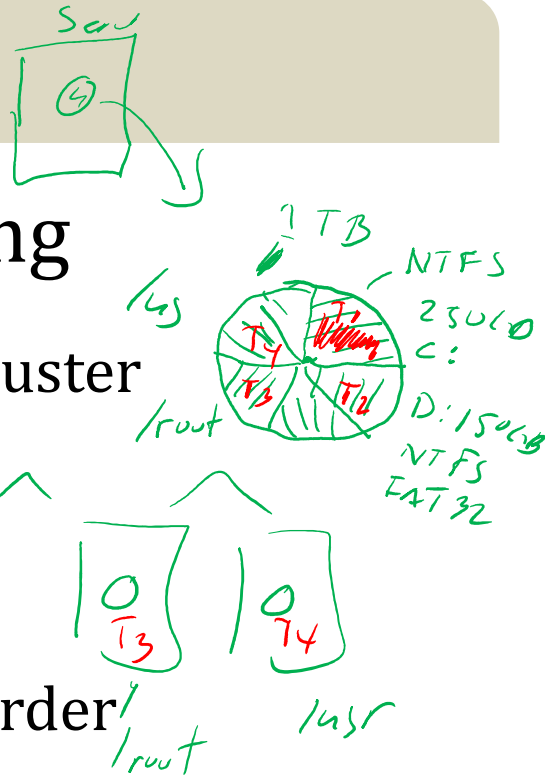
Replicating the Database

- Much harder because the state must be unique. In other words, database must act as a whole
  - Current DB instance must always be ***consistent***
    - Ex: Foreign keys must exist
    - as a result, some updates must occur simultaneously

Replicating the Database

- Much harder because the state must be unique. In other words, database must act as a whole
  - Current DB instance must always be ***consistent***
    - Ex: Foreign keys must exist
    - as a result, some updates must occur simultaneously
- Two basic approach:
  - Scale up by partitioning
  - Scale up by replication

Scale Through Partitioning



- Partition the DB across many machines in a cluster
  - Database could fit in main memory
  - Queries spread across these machines
- Can increase throughput
- Easy for (simple) writes but reads become harder

* throughput = amount of material/items passing through a system or process

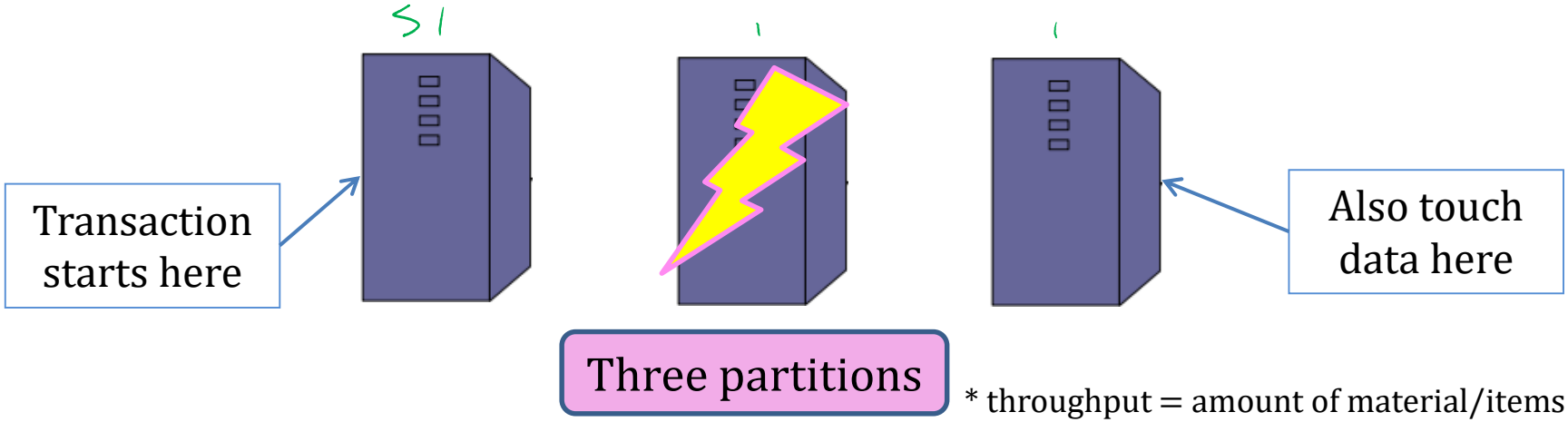
Scale Through Partitioning

- Partition the DB across many machines in a cluster
  - Database could fit in main memory
  - Queries spread across these machines
- Can increase throughput
- Easy for (simple) writes but reads become harder

* throughput = amount of material/items passing through a system or process

Scale Through Partitioning

- Partition the DB across many machines in a cluster
  - Database could fit in main memory
  - Queries spread across these machines
- Can increase throughput
- Easy for (simple) writes but reads become harder



* throughput = amount of material/items passing through a system or process