

# [544] BigQuery Cost Management

Tyler Caraza-Harter

# Learning Objectives

- describe the relationship between BigQuery's two billing models (capacity and on-demand)
- manage and inspect BigQuery costs

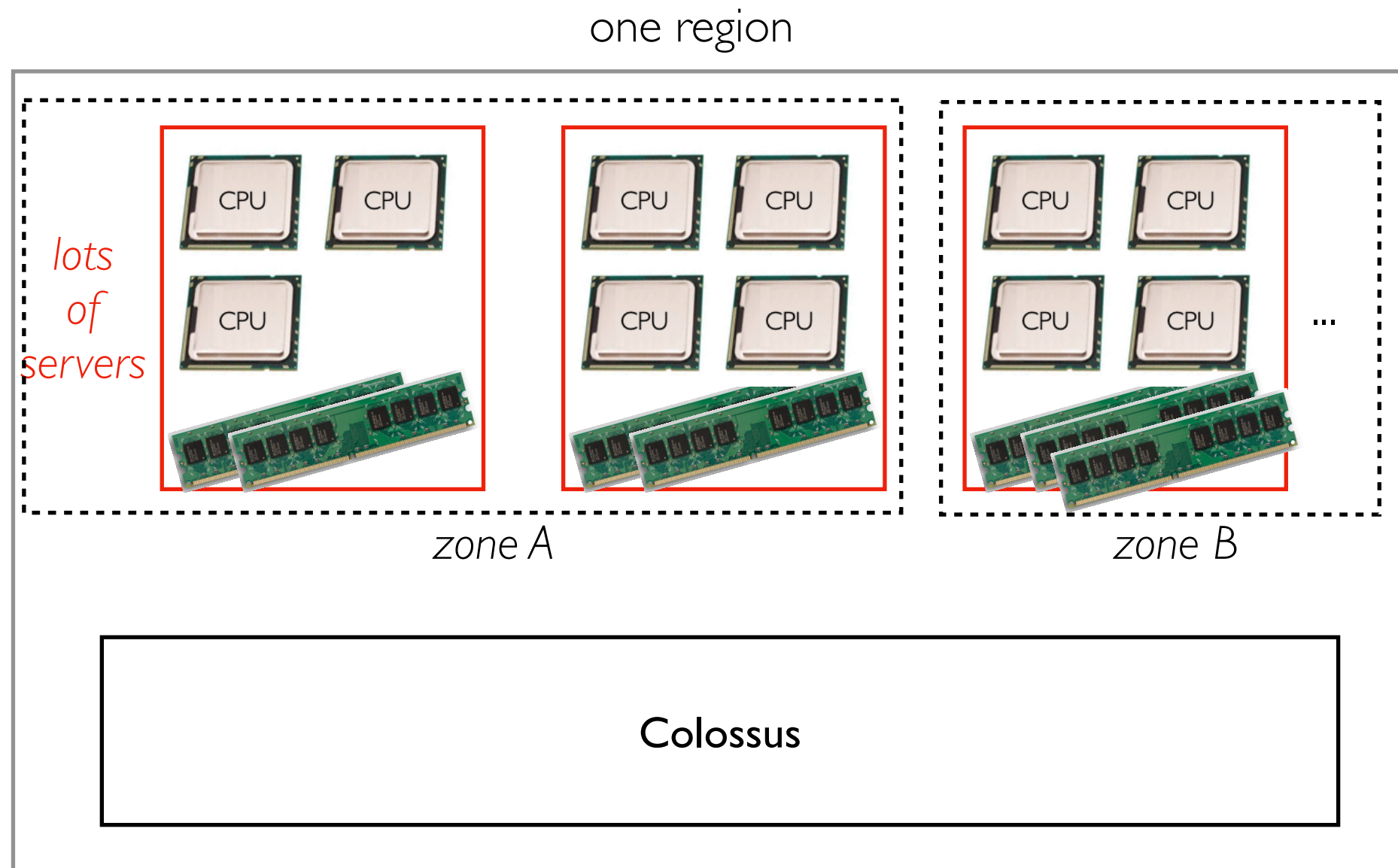
# Outline

Billing Models

Optimization

# Resources

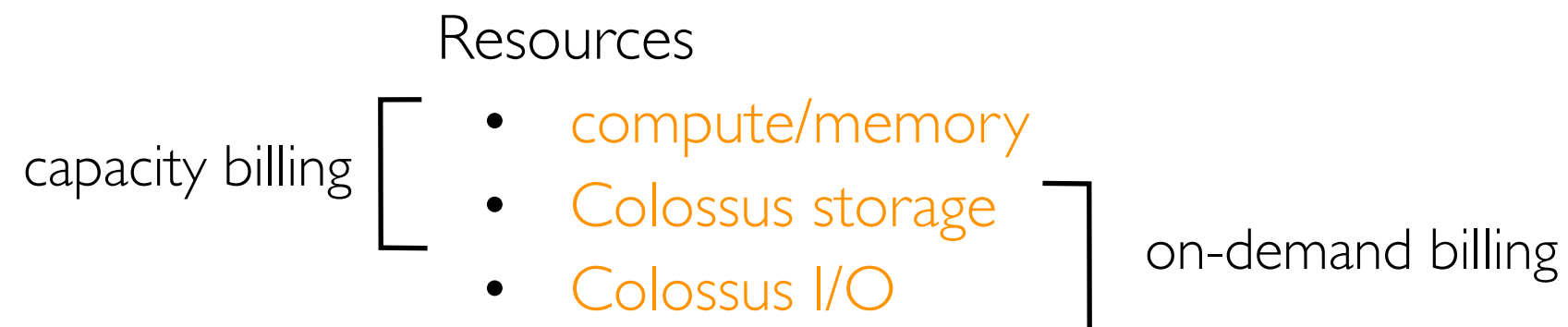
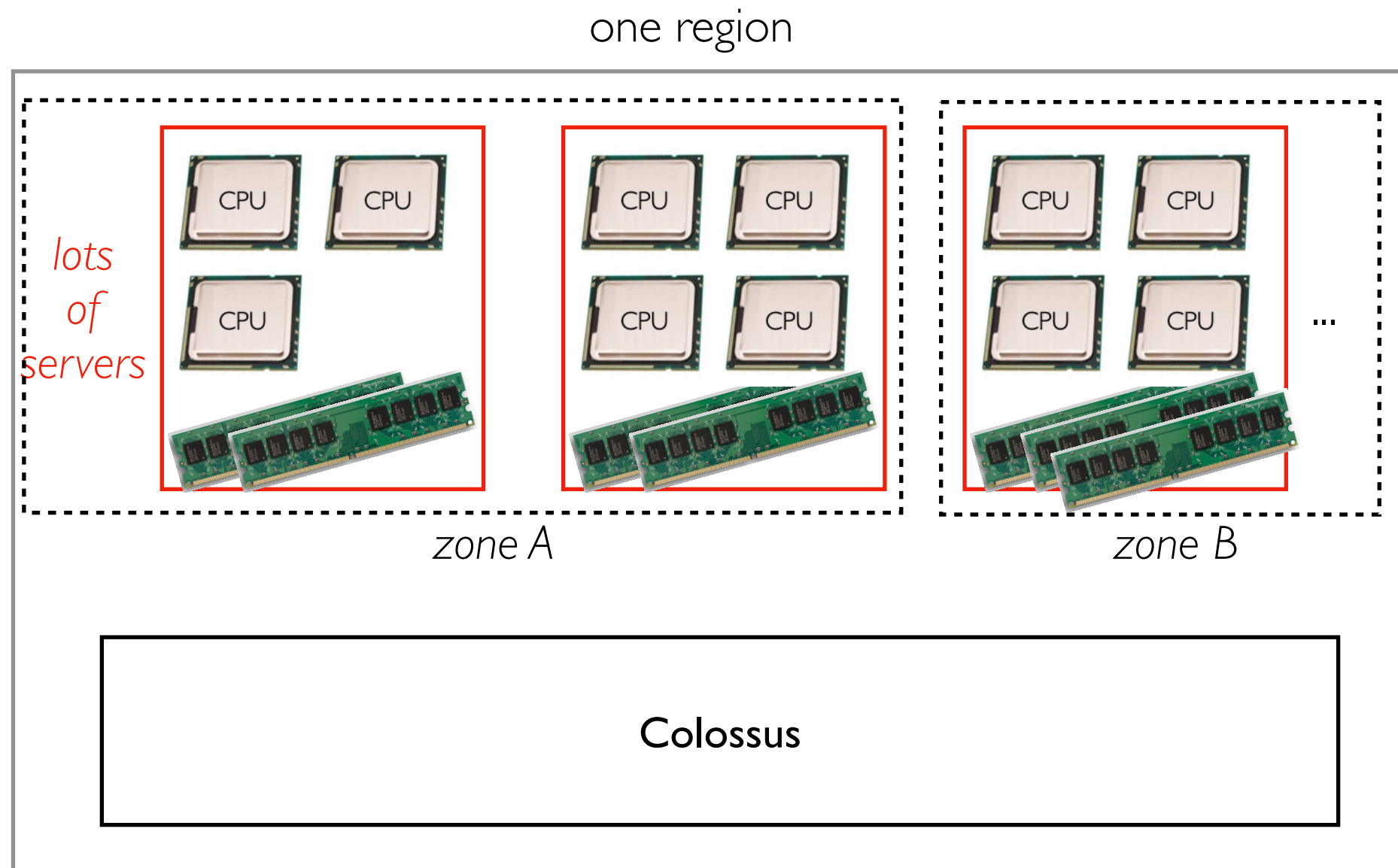
other regions...



- **Query engine:** Dremel running on many servers with lots of CPU+RAM
- **Storage engine:** Capacitor files in Colossus file system  
(not clear if Dremel+Colossus servers are co-located on same machines)

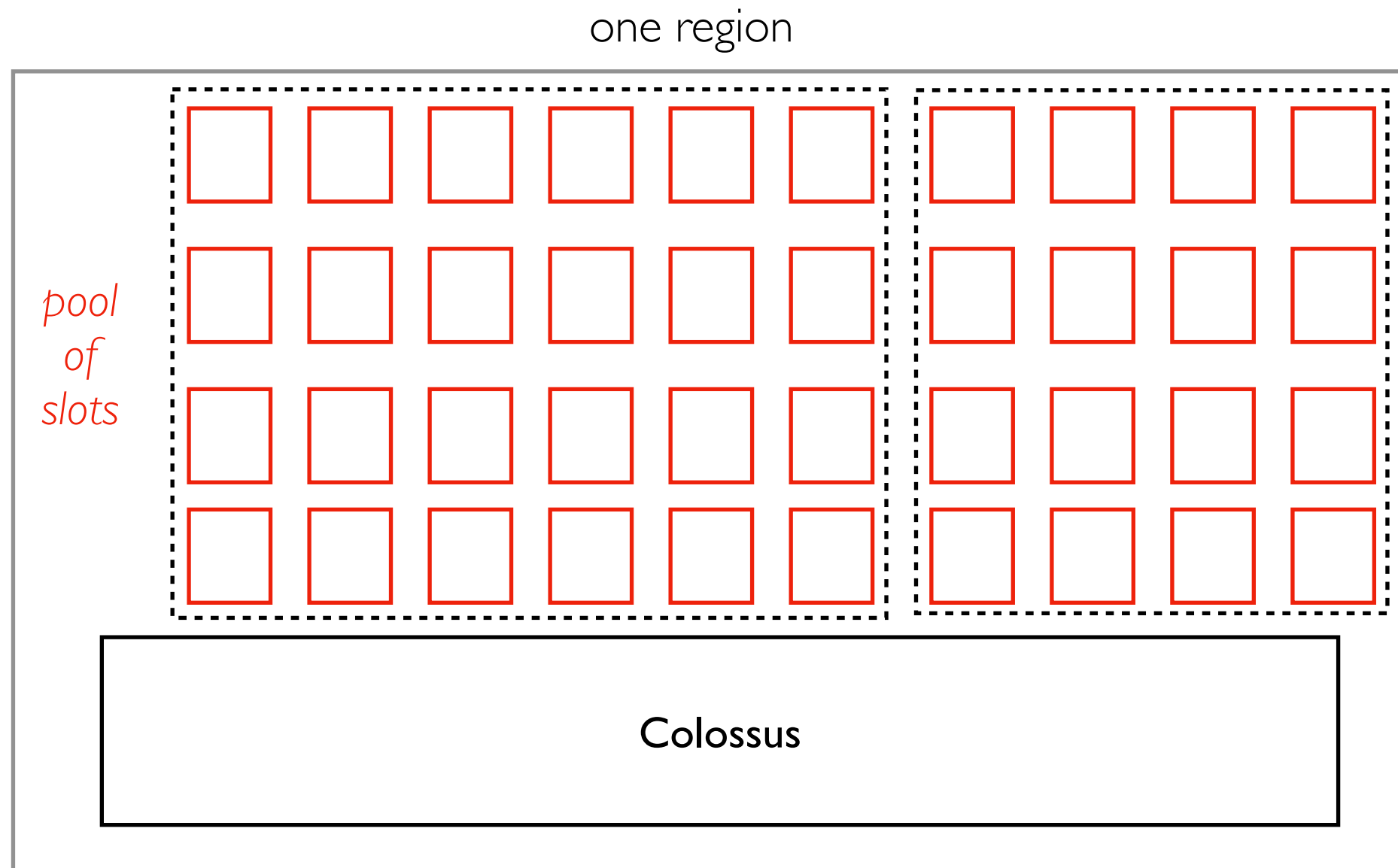
# Resources

other regions...



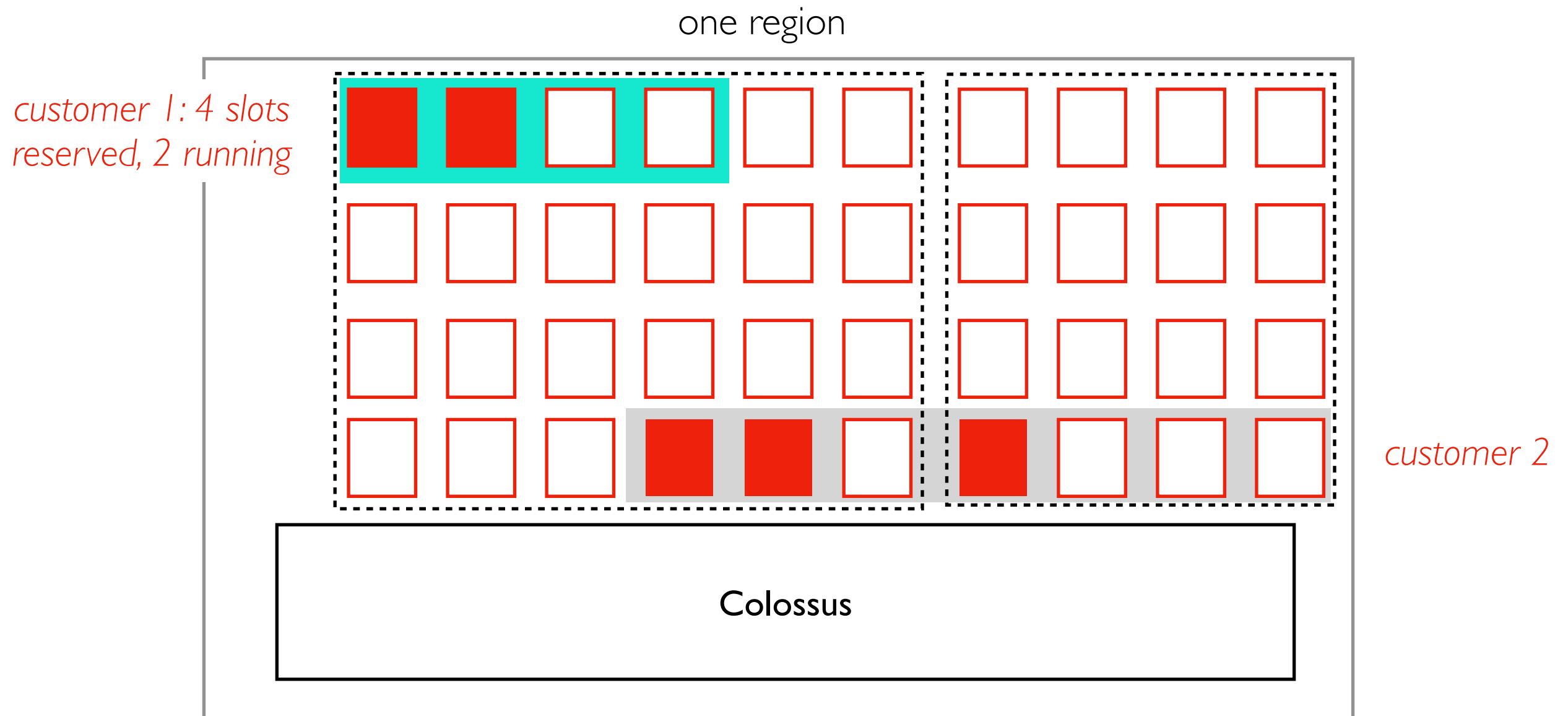
# BigQuery Slots

other regions...



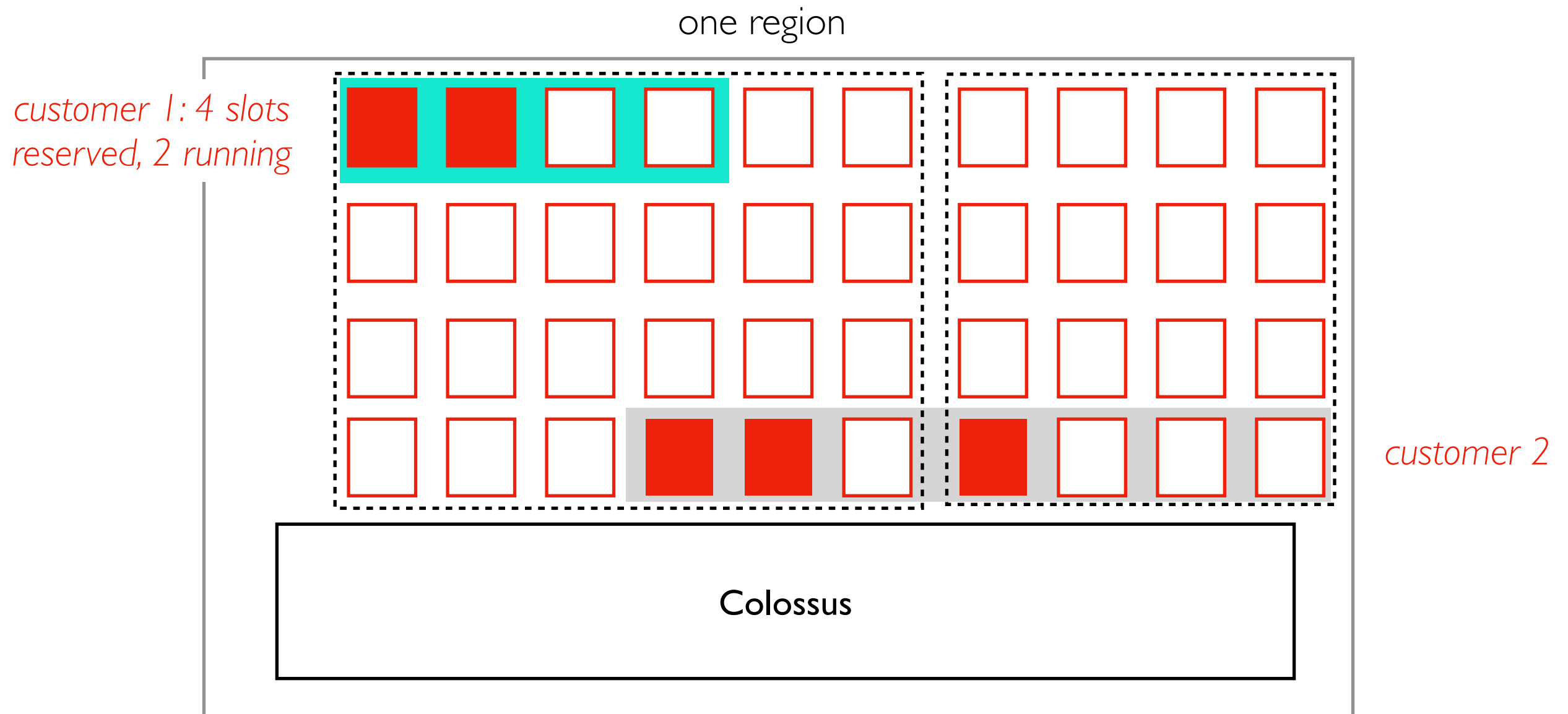
- the compute and memory resources of the servers are broken down into a pool of "slots"
- a slot has approximately  $\frac{1}{2}$  cores and 1 GB of RAM
- if newer servers get added with faster CPUs or different core/memory ratios, the exact resources can change a bit

# Billing Model I: Capacity Pricing (compute based)



- customers can pay a fixed rate for slot capacity (about \$0.96 for 1 slot day)
- whether or not they use the slot does not affect the cost
- reservations aren't fixed to one location (execution will ideally happen near the data).
- *slightly more expensive than the e2-medium instances we used this semester, which have 2x compute and 4x memory resources (but not free Colossus I/O). But VMs are IaaS and BigQuery is PaaS.*

# Billing Model 1: Capacity Pricing (compute based)



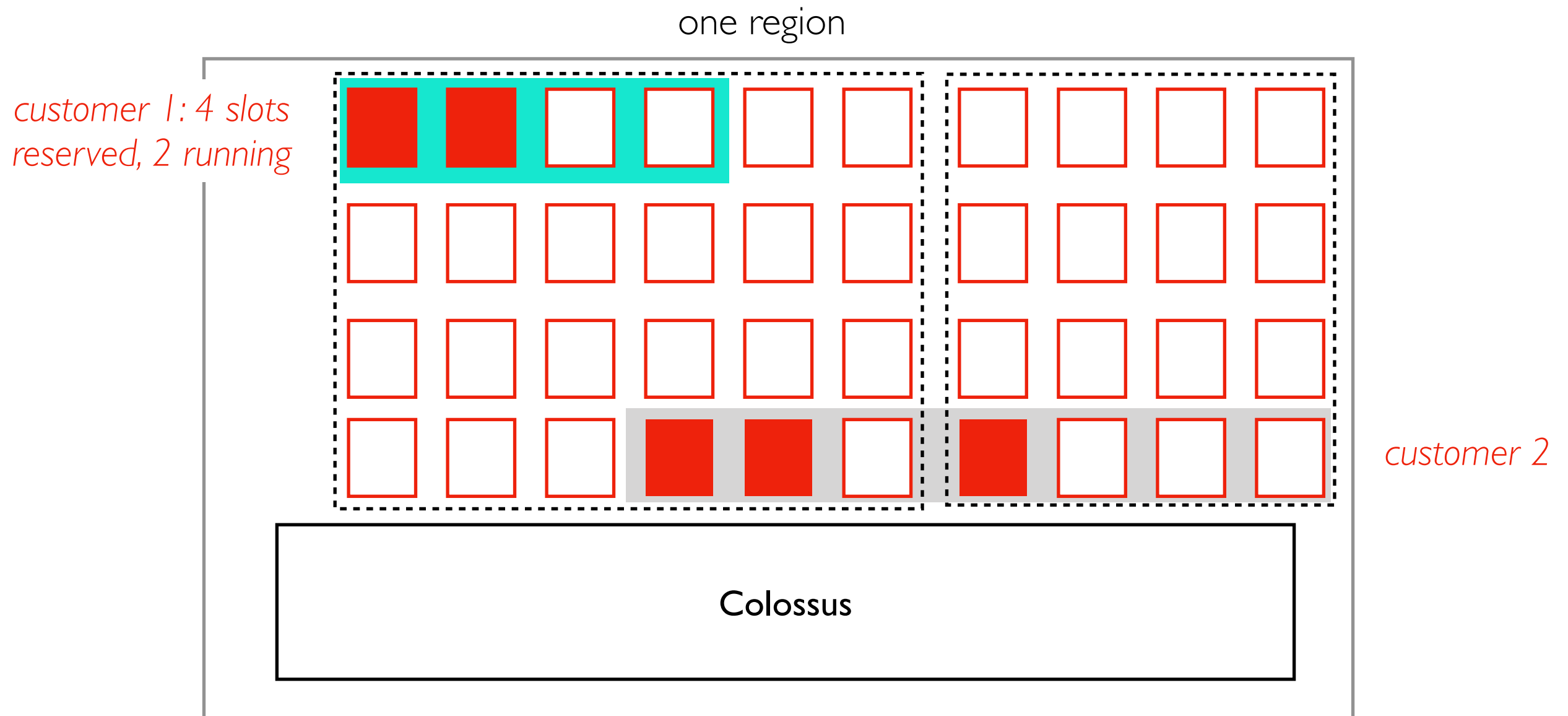
Excess capacity cases:

- not reserved
- reserved, but not currently used

Billing Model 2 (On-Demand) draws from this excess...



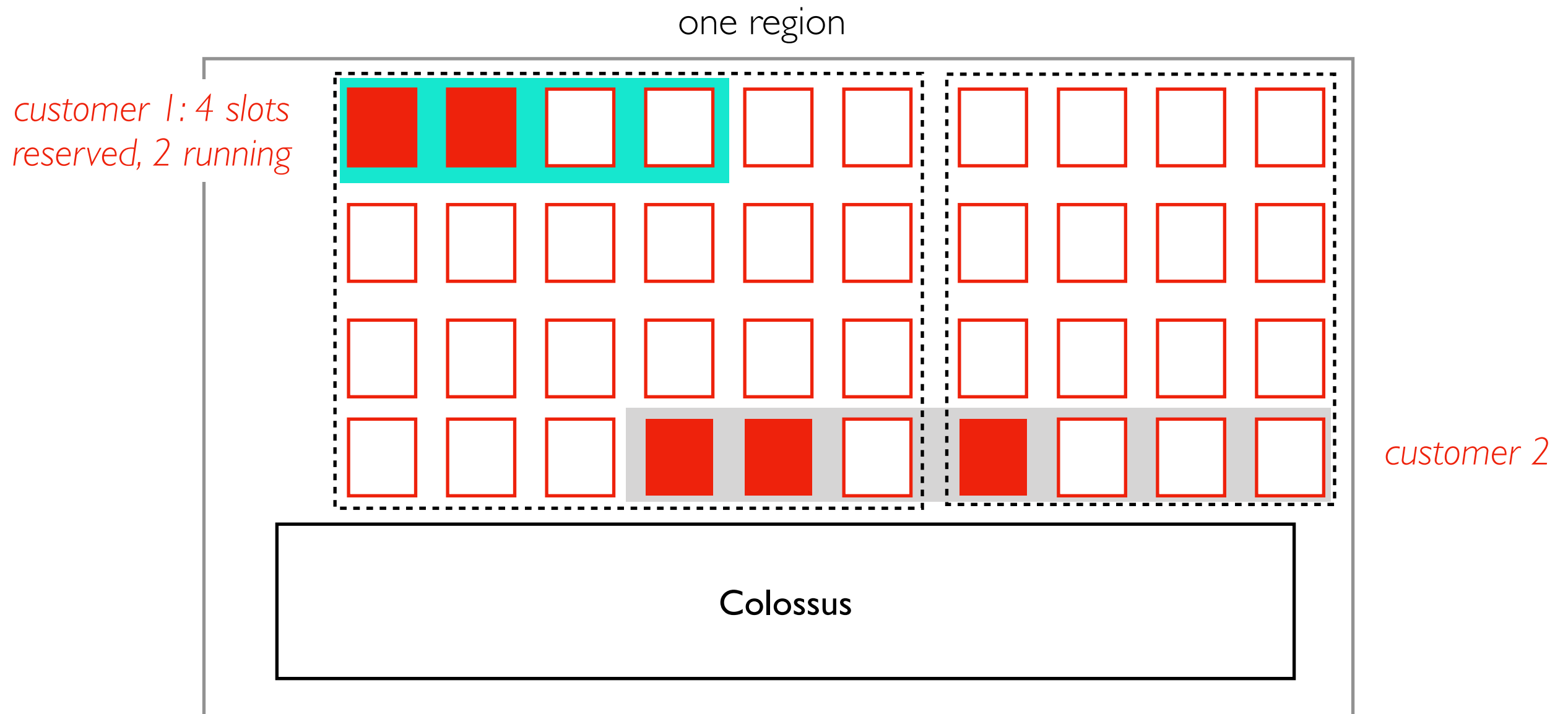
# Billing Model 2: On-Demand Pricing (I/O based)



## Pricing:

- pay for Colossus I/O after free tier (about \$6.25/TB)
- slots (compute/memory) are free
- use whatever is left over from capacity-based usage (up to 2000 slots!)
- **preemptible**: a task running in a slot can be interrupted (if a reservation is suddenly needed or new on-demand jobs start -- want to share capacity between these fairly)

# Billing Model 2: On-Demand Pricing (I/O based)



Pricing:

- pay
  - slots
  - use
  - preemptible: a task running in a slot can be interrupted (if a reservation is suddenly needed or new on-demand jobs start -- want to share capacity between these fairly)
- BigQuery tasks are **atomic** and **idempotent** so we have **exactly-once semantics**. Don't want interrupted and restarted tasks to cause duplicate results.**
- 2000 slots!)

# Comparison


## Capacity Billing

- very predictable costs
- very predictable performance (other customers don't affect you)
- discounts if commit to buying lots of cores for a long time (e.g., a year)
- pay when using nothing
- can't use lots of resources for a short while

## On-Demand Billing

- pay-as-you-go: use nothing, pay nothing
- if resources are available, you can use 1000 cores at once -- very fast!
- how to make sure you don't accidentally spend more than intended?

# Estimating/Capping On-Demand Costs

Filter Metric : bigquery.googleapis.com/quota/query/usage  Enter property name or value					
<input type="checkbox"/>	Quota	Dimensions (e.g. location)	Limit	Current usage percentage ↓	Current usage
<input type="checkbox"/>	Query usage per day		1,048,576 MiB (1 TiB) ⓘ	<div><div></div></div> 0%	0 MiB
<input type="checkbox"/>	Query usage per day per user		Unlimited	– ⓘ	

Options:

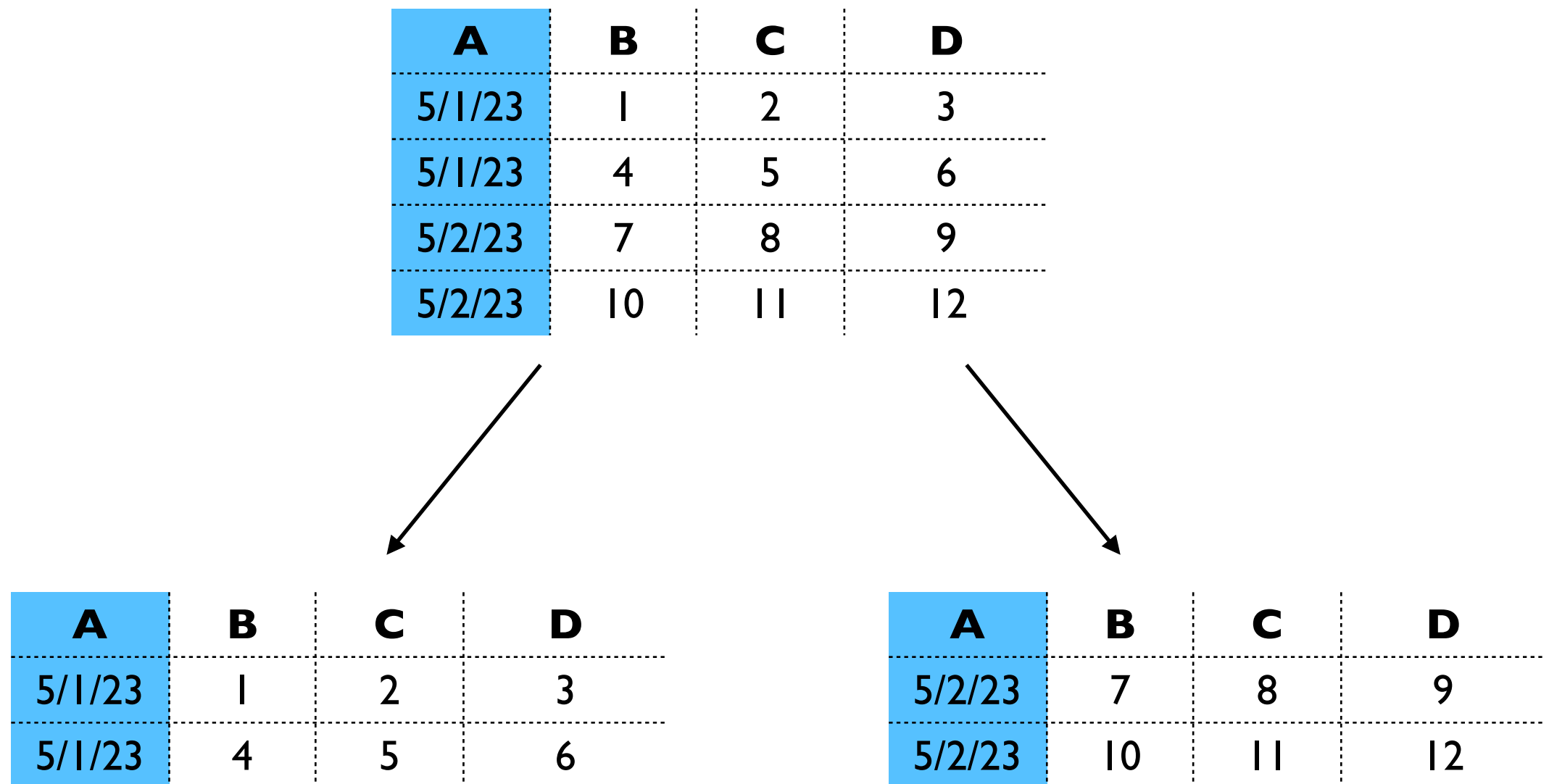
- Limit per day:  
<https://console.cloud.google.com/iam-admin/quotas>
- Estimate before run:  
`job_config=bigquery.QueryJobConfig(dry_run=True)`
- Set max per query:  
`bigquery.QueryJobConfig(maximum_bytes_billed=200*1024**2)`
- See most expensive queries:  
`cs320-f21.region-us.INFORMATION_SCHEMA.JOBS_BY_PROJECT`

# Outline

Billing Models

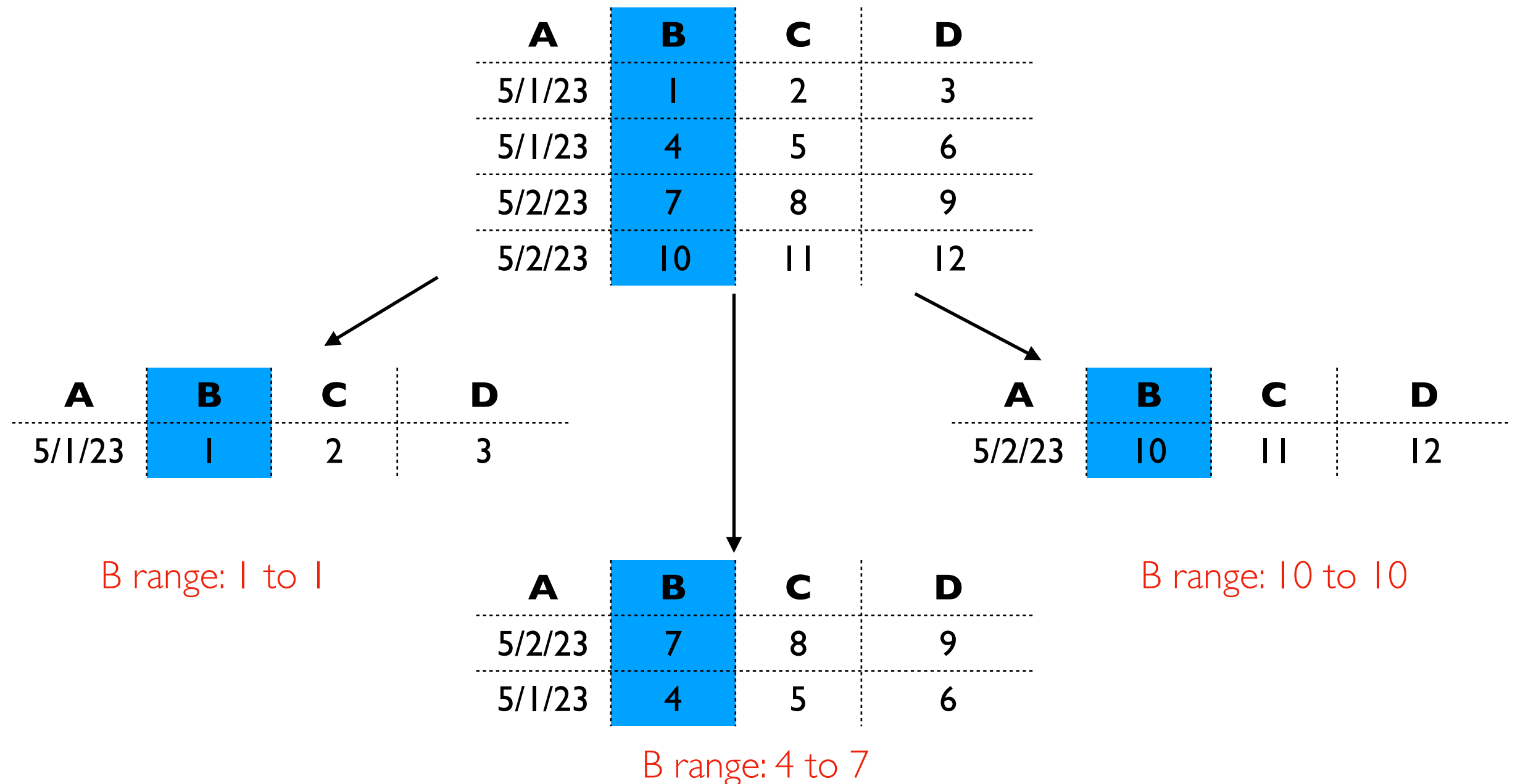
Optimization

# Partitioning



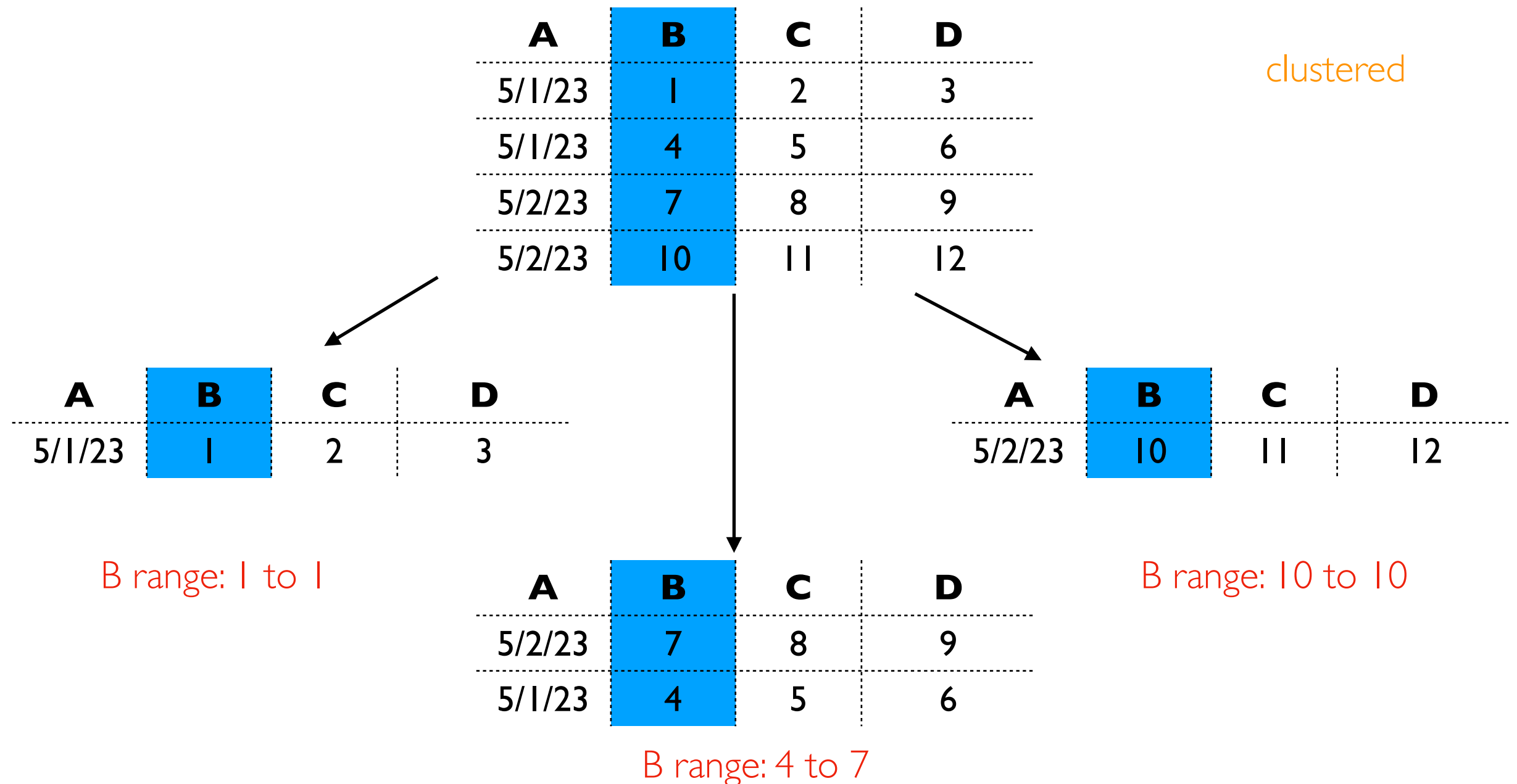
- each unique value in a partition column corresponds to a partition (basically a mini table)
- WHERE filters can limit which mini tables need to be read (saving I/O cost)
- limited options for types (e.g., ints, dates)
- only works well when substantial data per partition

# Clustering



- semi sorted: sub files are non overlapping on cluster key, but no order within file
- all types, combinations of columns possible
- some queries will be cheaper because they can look at subset of files

# Clustering



- some min ratio of data is clustered
- don't want few new rows to force total reorg

A	B	C	D
5/2/23	5	1	2
5/1/23	12	3	4

unclustered



Demos