



Finding needles in ever increasing haystacks

Overview of the building blocks for the scalability of Elasticsearch

Iraklis Psaroudakis

May 6th, 2023 at Devvxx Greece

Iraklis Psaroudakis



Principal Software Engineer
at [Elastic](#), focusing on distributed

Previously:

[Oracle Labs](#), graph-based analytics
PhD at [EPFL](#), scaling up analytics
ECE degree at [NTUA](#) in Athens

Agenda:

- What is Elastic
- What is Elasticsearch
- How Elasticsearch scales out with shards
- Distributed searches & aggregations
- Shard recovery
- Cluster state
- Scaling with data streams
- Scaling with ILM
- Future: scaling with Serverless

This talk contains personal views and is not officially endorsed

Meet Elastic










Elastic helps the world's leading organizations **accelerate results that matter** by putting data to work with the **power of search**.



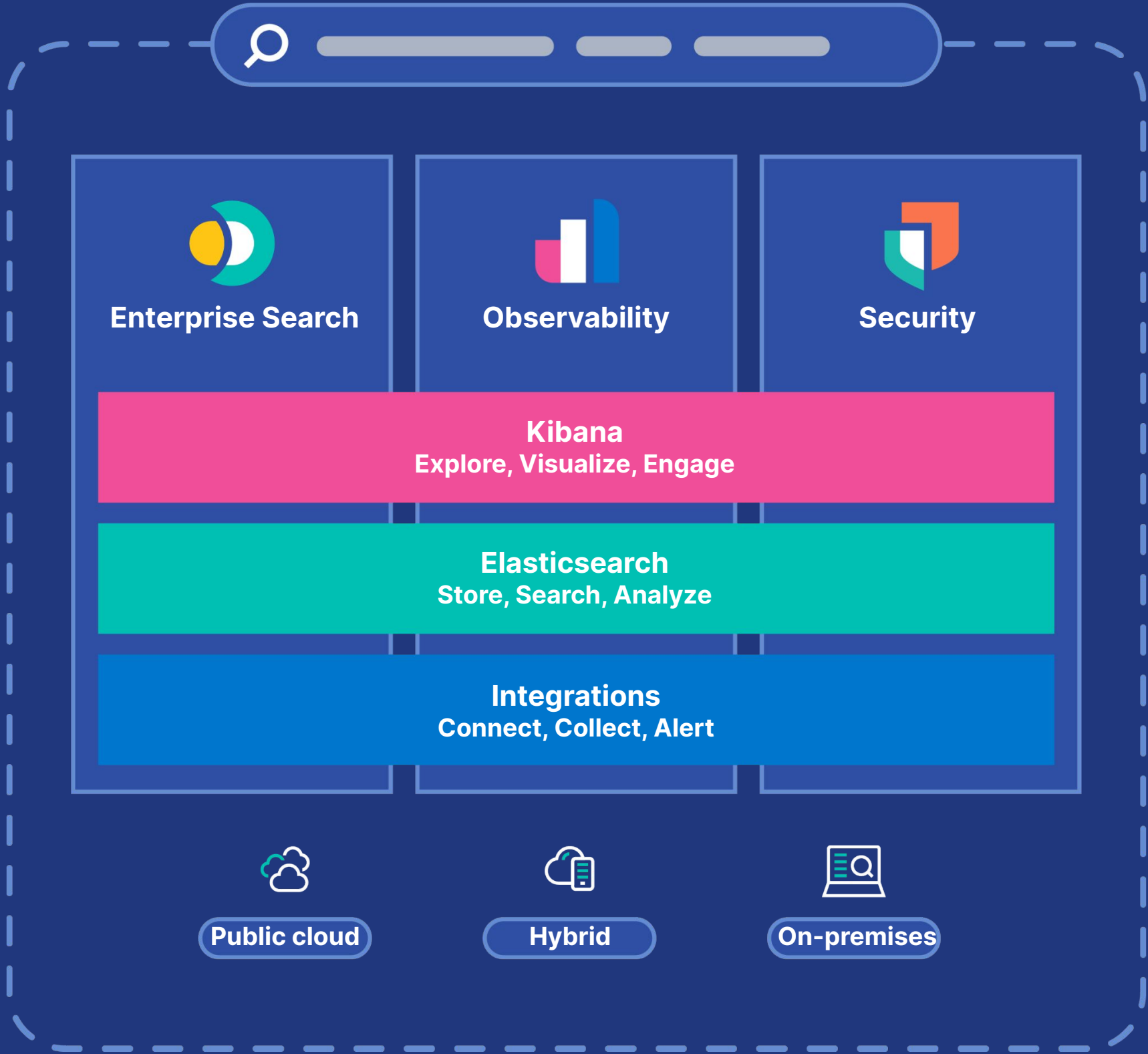
Founded in 2012
NYSE: ESTC



The Elastic Search Platform *is for everyone*

TECHNOLOGY	FINANCE	TELCO	CONSUMER	HEALTHCARE	PUBLIC SECTOR	AUTOMOTIVE / TRANSPORTATION	RETAIL
							
							
							
							
							


The Elastic Search Platform



Elasticsearch

- Distributed, scalable, highly available, resilient search & analytics engine
- HTTP based JSON interface
- Flexibility (index time vs. query time)
- Based on [Apache Lucene](#)
- Much more than grep or SQL's
LIKE = '%quick%'
 - Ranked results (BM25, recency, popularity), fuzzy matching
 - Complex search expressions
 - Spell correction, Synonyms, Phrases, Stemming
- Timeseries, geospatial, ML, vector search









github.com/elastic/elasticsearch

 elastic/elasticsearch
Free and Open, Distributed, RESTful **Search Engine**

java search-engine elasticsearch

☆ 63.4k ● Java Updated 3 hours ago 177 issues need help

db-engines.com/en/ranking

Rank			DBMS	Database Model
Apr 2023	Mar 2023	Apr 2022		
1.	1.	1.	Oracle +	Relational, Multi-model 
2.	2.	2.	MySQL +	Relational, Multi-model 
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model 
4.	4.	4.	PostgreSQL +	Relational, Multi-model 
5.	5.	5.	MongoDB +	Document, Multi-model 
6.	6.	6.	Redis +	Key-value, Multi-model 
7.	7.	↑8.	IBM Db2	Relational, Multi-model 
8.	8.	↓7.	Elasticsearch	Search engine, Multi-model 
9.	9.	↑10.	SQLite +	Relational
10.	10.	↓9.	Microsoft Access	Relational

Inverted index

Document 1: "The quick brown fox jumped over the lazy dog"

Document 2: "Quick brown foxes leap over lazy dogs in summer"

Map: sorted tokens → documents

Quick	2
The	1
brown	1,2
dog	1
dogs	2
fox	1
foxes	2
in	2
jumped	1
lazy	1,2
leap	2
over	1,2
quick	1
summer	2
the	1

~ Lucene segment

Queries

"lazy AND dog"

→ [1] AND [1,2] → [1]

"lazy OR dog"

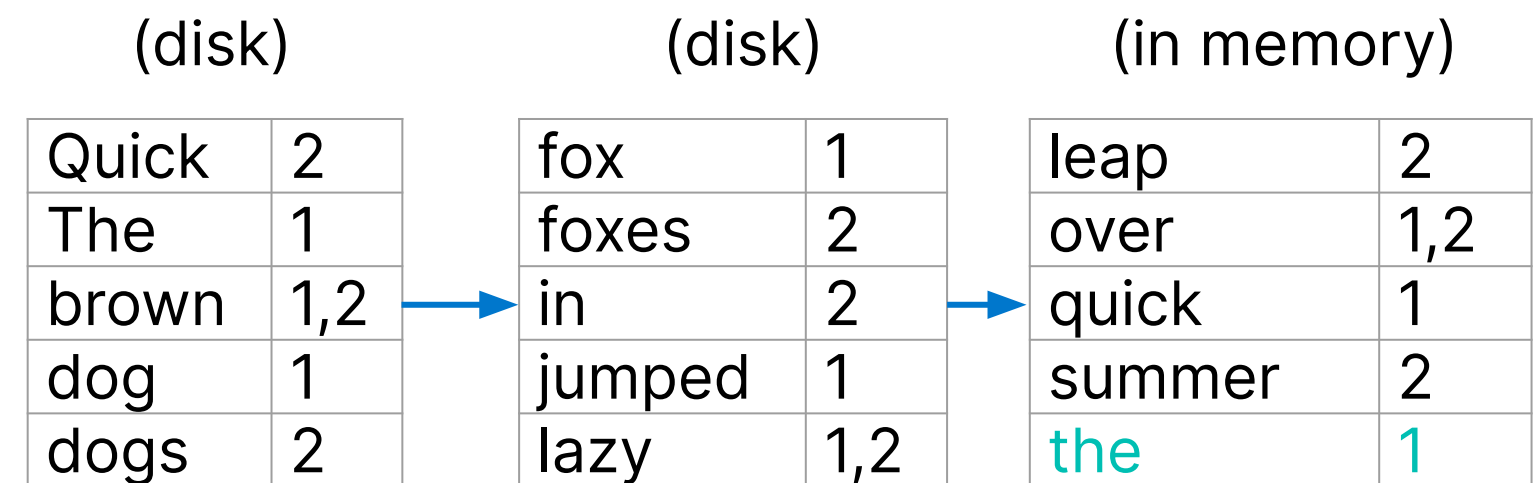
→ [1] OR [1,2] → [1,2]

and a document can have a higher score (TF/IDF based)

A Lucene instance comprises of segments

Each being a self sufficient inverted index

- Segments are immutable!
 - Pros: write-once, read efficient, file system cache
 - Cons: deletes (separate file) & updates (new segment), housekeeping (e.g., merging)
- Bulk insertions preferred
- What else can be in there?
 - Term frequencies: relevancy
 - Positions: positional queries
 - Offsets: highlighting
 - Stored fields: the original data

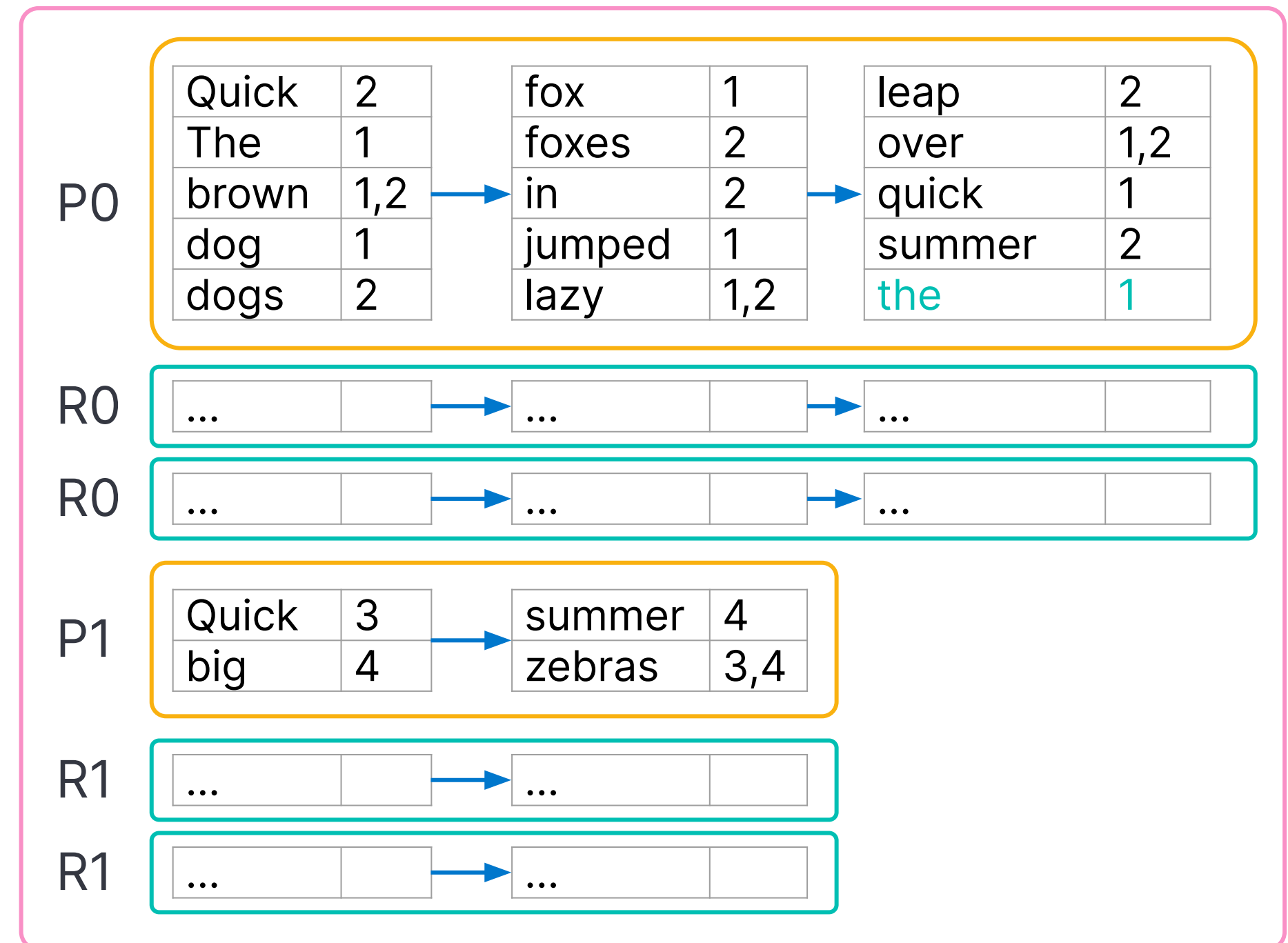


Elasticsearch scales with shards

An “index” consists of shards

- A shard is a Lucene instance
- Primary shards
 - Partitioning of data in the index (write/ingest scalability)
- Replica shard
 - Auto synced copy of a primary (query scalability)

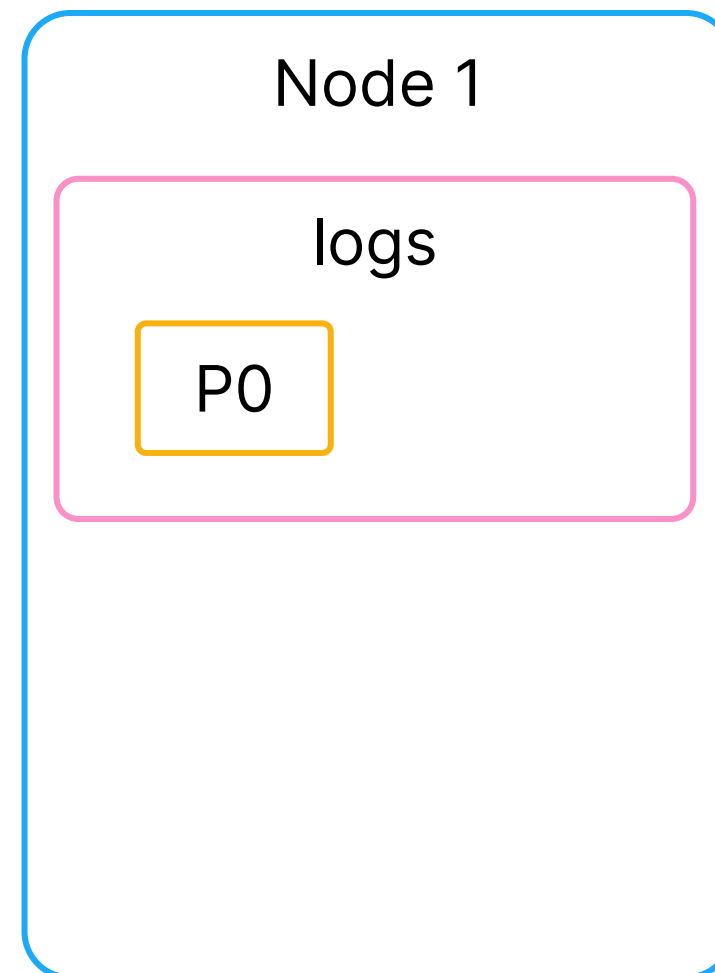
Index “children books”



Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

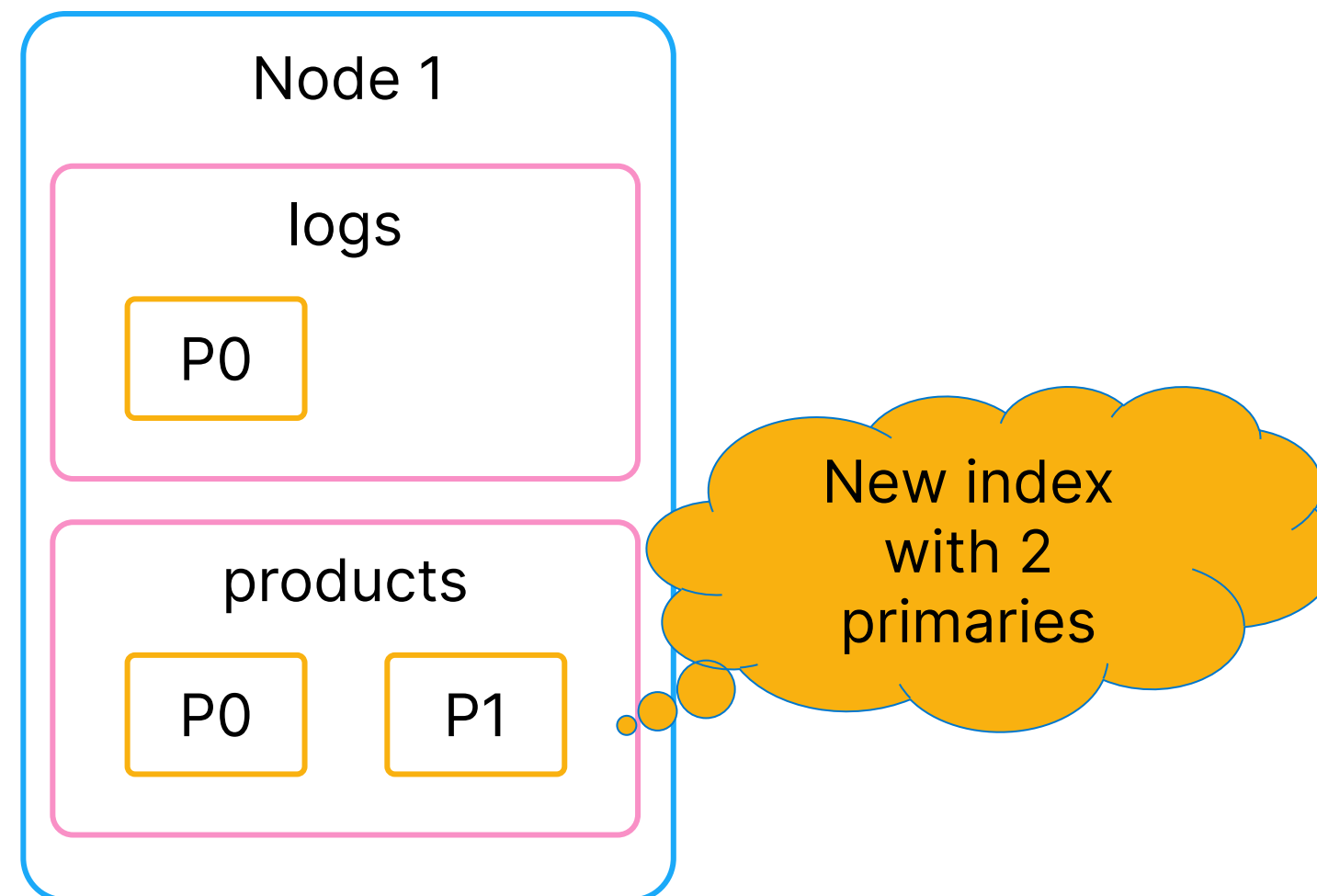
- In a balanced manner → ingestion and querying distribution
- Replicas are not colocated → fault tolerance



Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

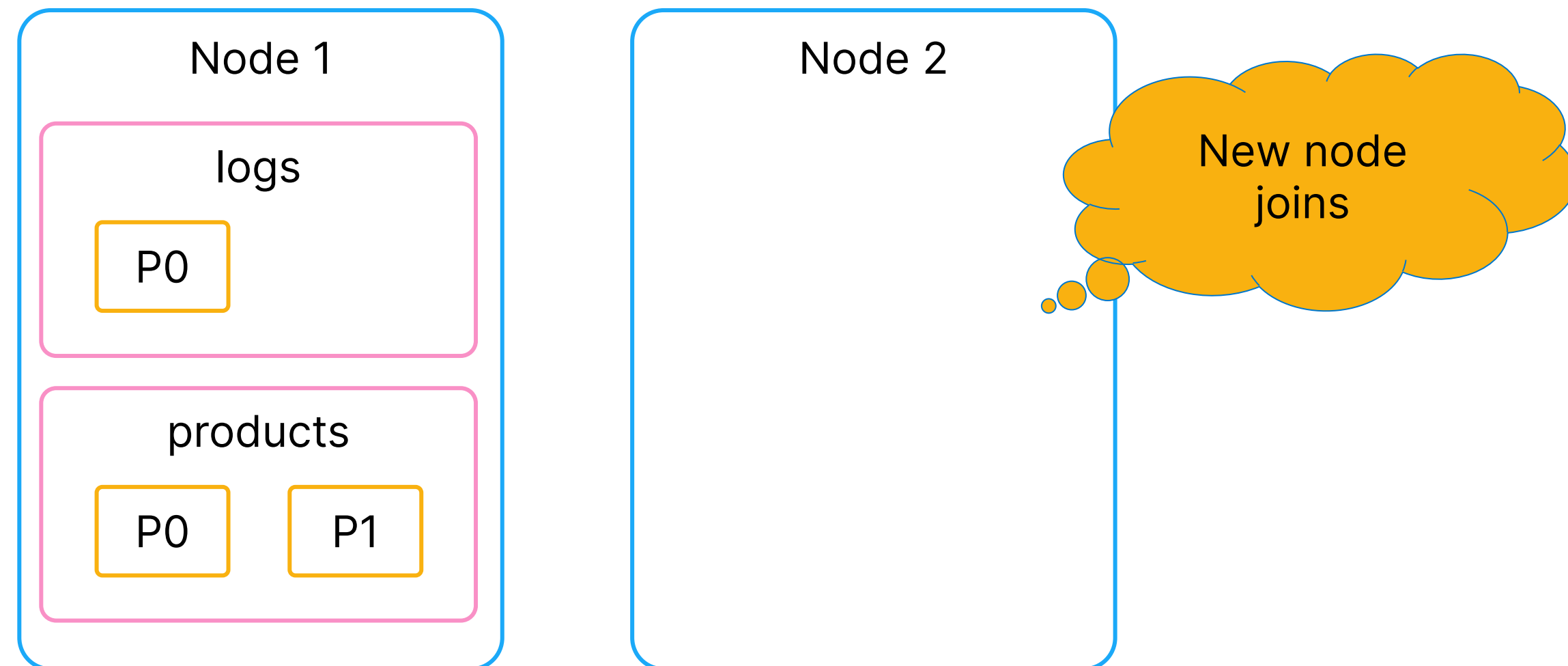
- In a balanced manner → ingestion and querying distribution
- Replicas are not colocated → fault tolerance



Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

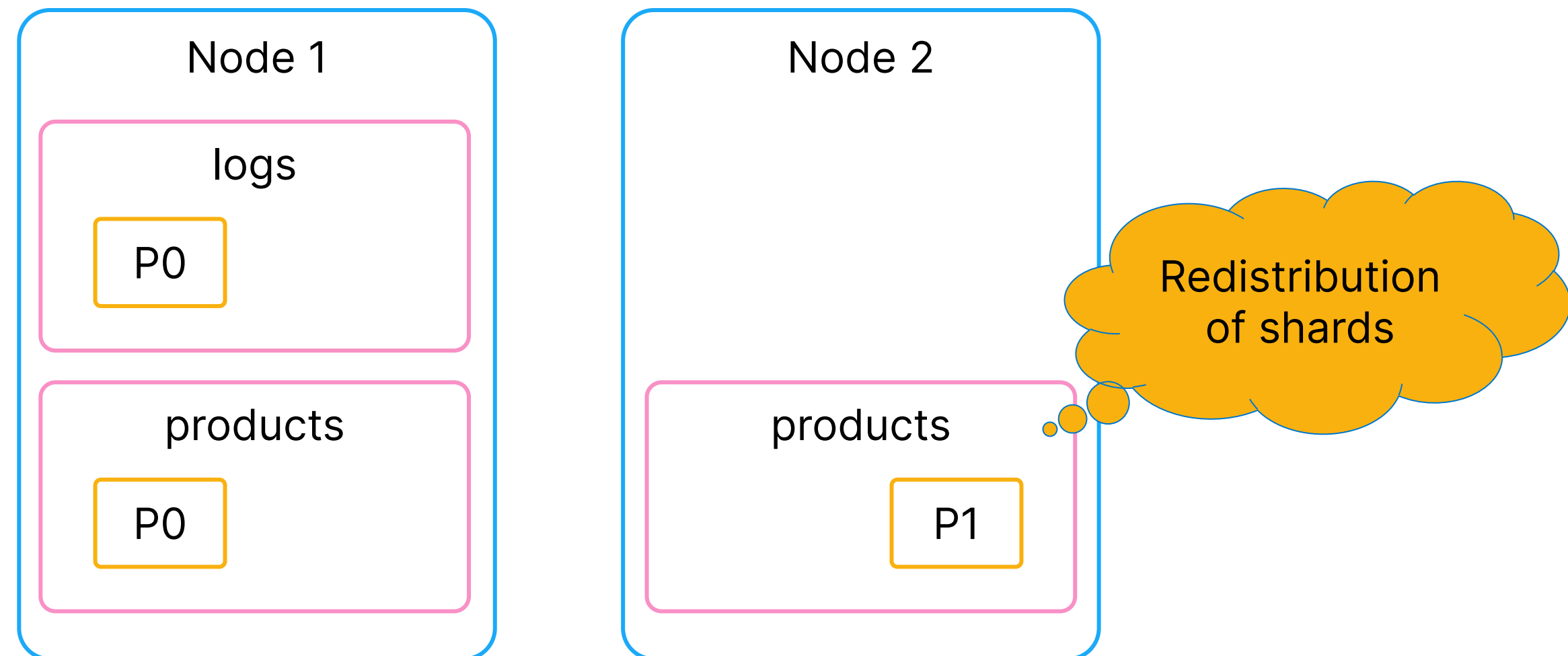
- In a balanced manner → ingestion and querying distribution
- Replicas are not colocated → fault tolerance



Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

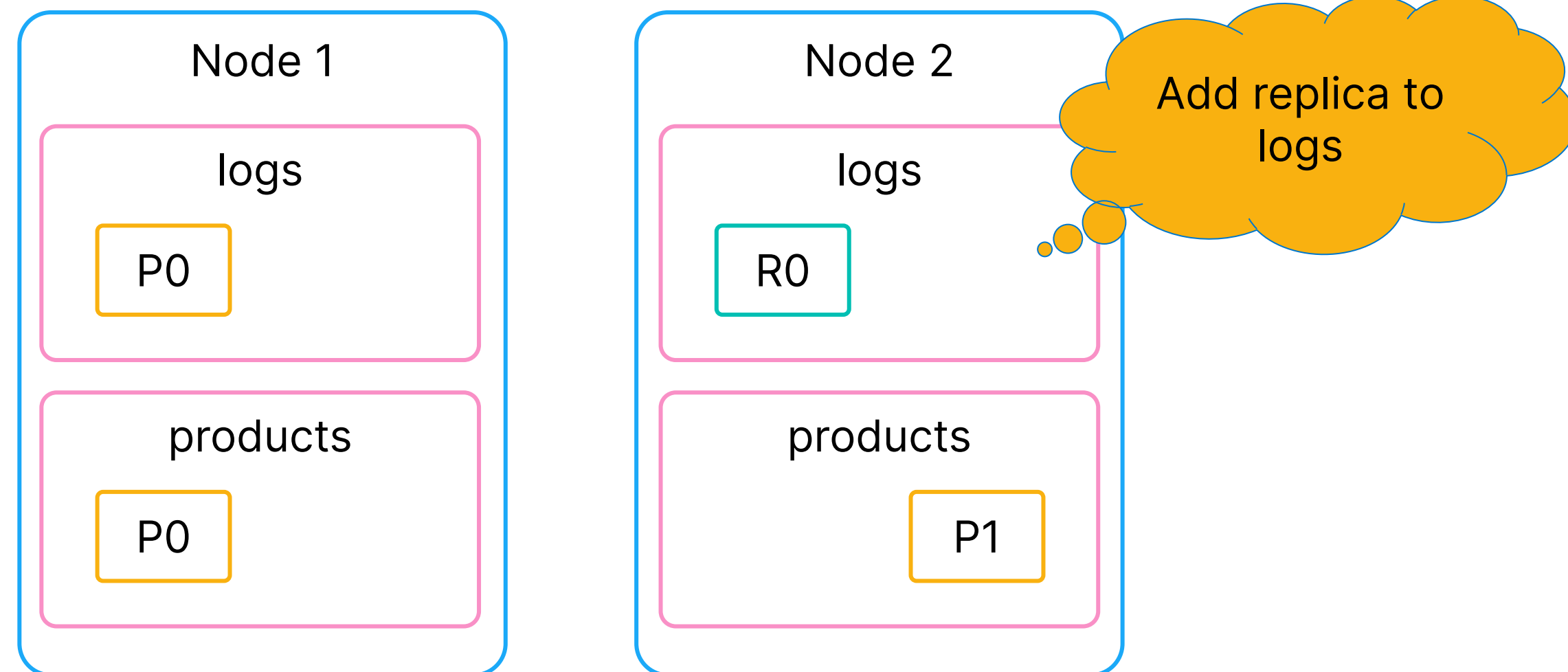
- In a balanced manner → ingestion and querying distribution
- Replicas are not colocated → fault tolerance



Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not colocated → fault tolerance

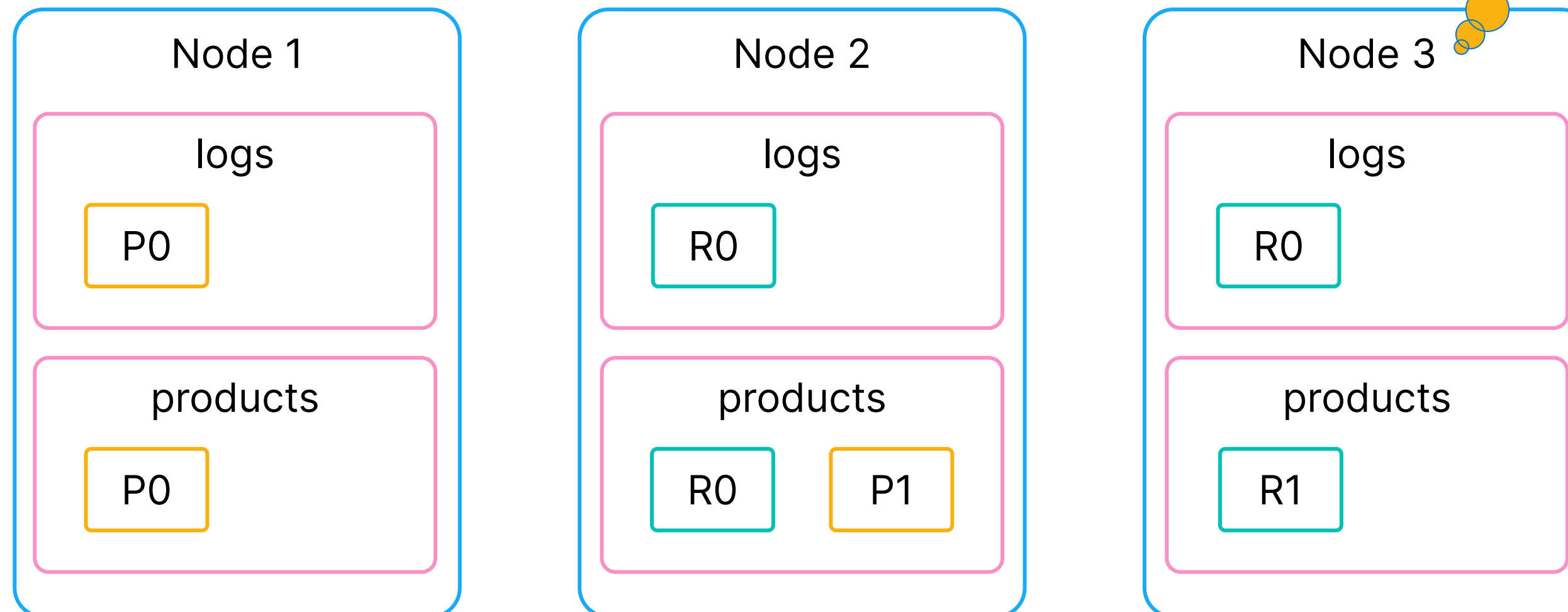


Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not colocated → fault tolerance

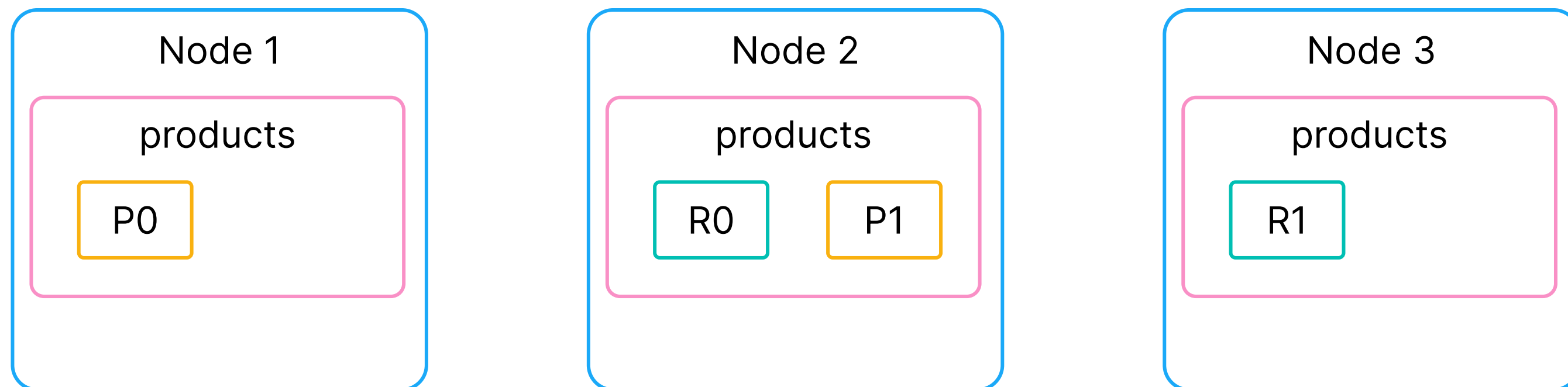
- New node
- logs: 1 shard, 2 replicas
- products: 2 shards, 1 replica



Distributed search

Two phase approach

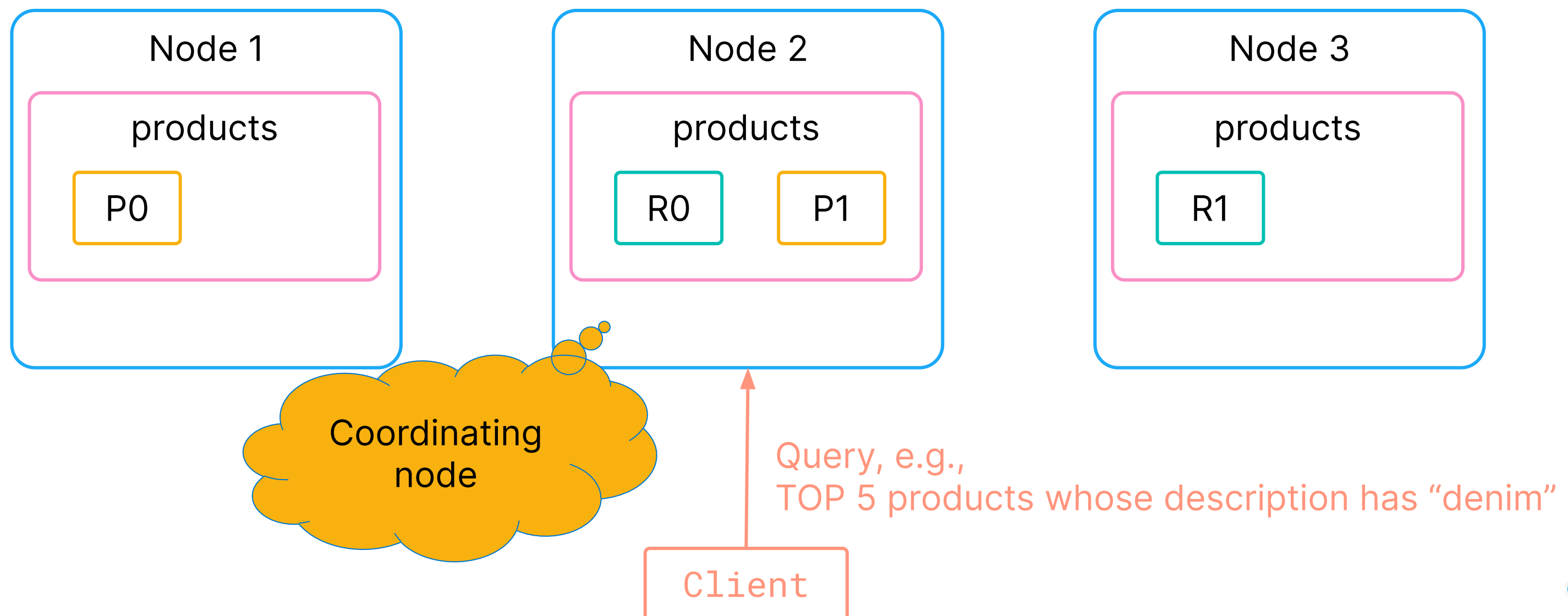
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed search

Two phase approach

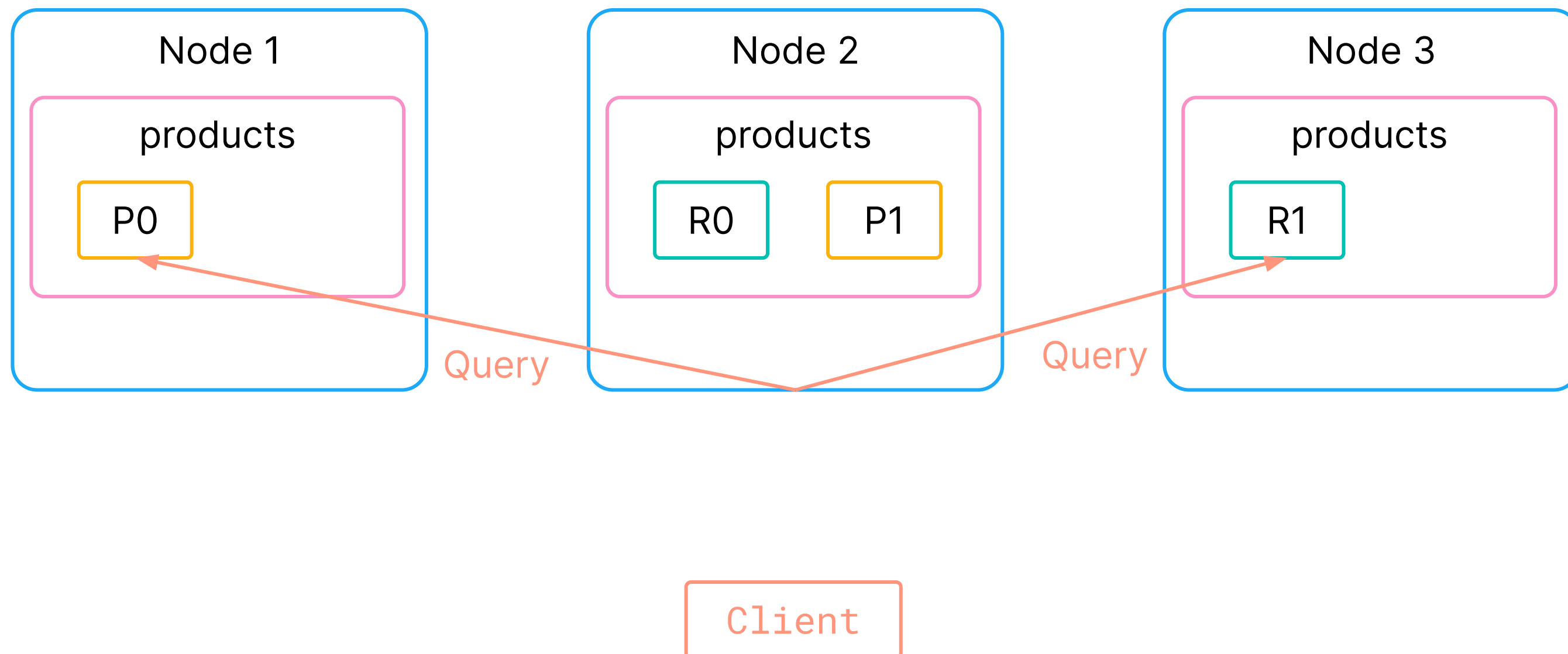
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed search

Two phase approach

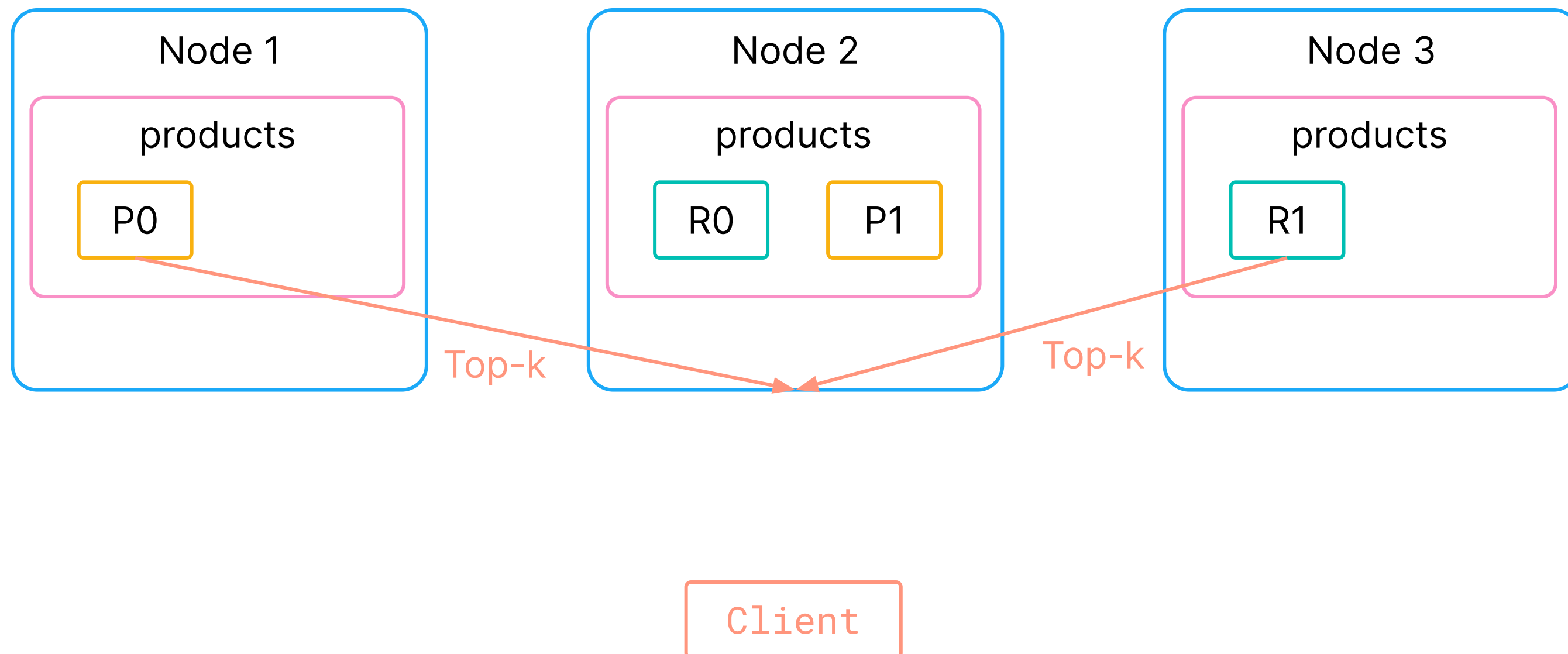
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed search

Two phase approach

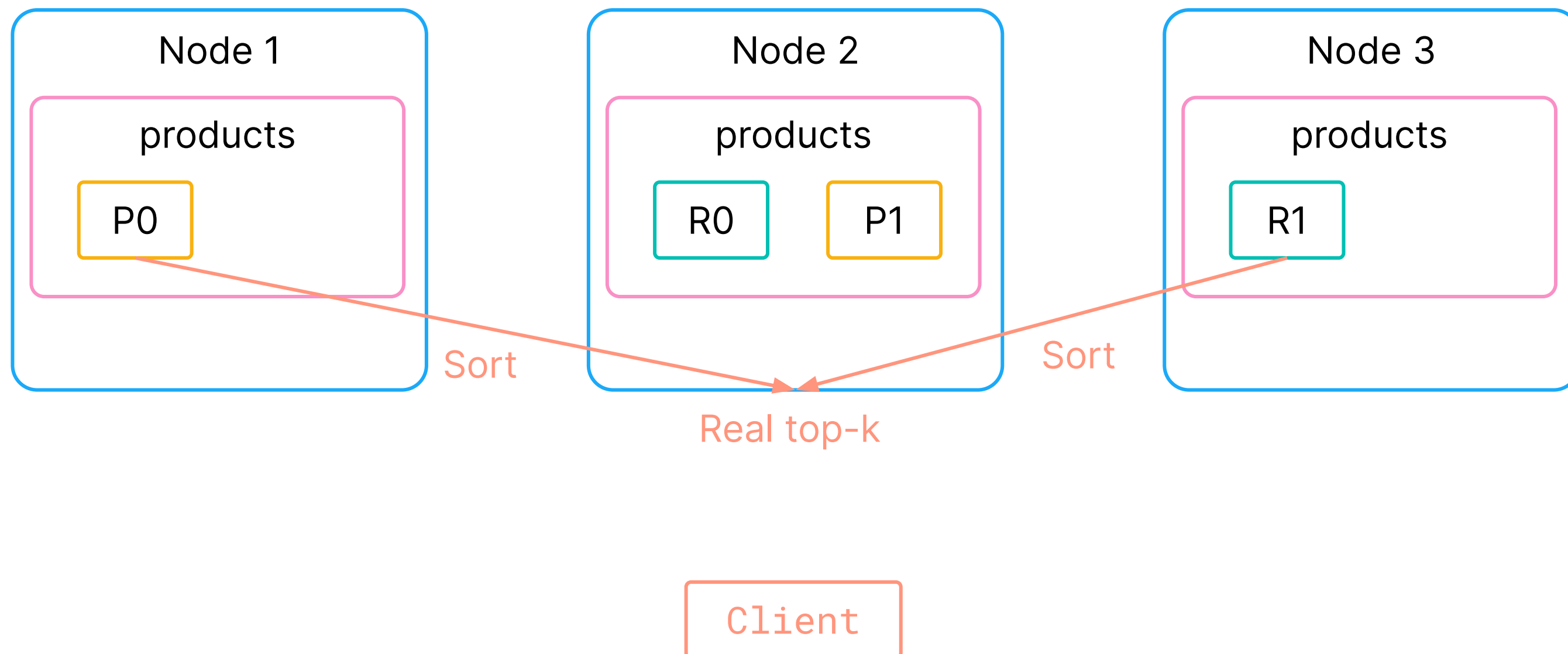
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed search

Two phase approach

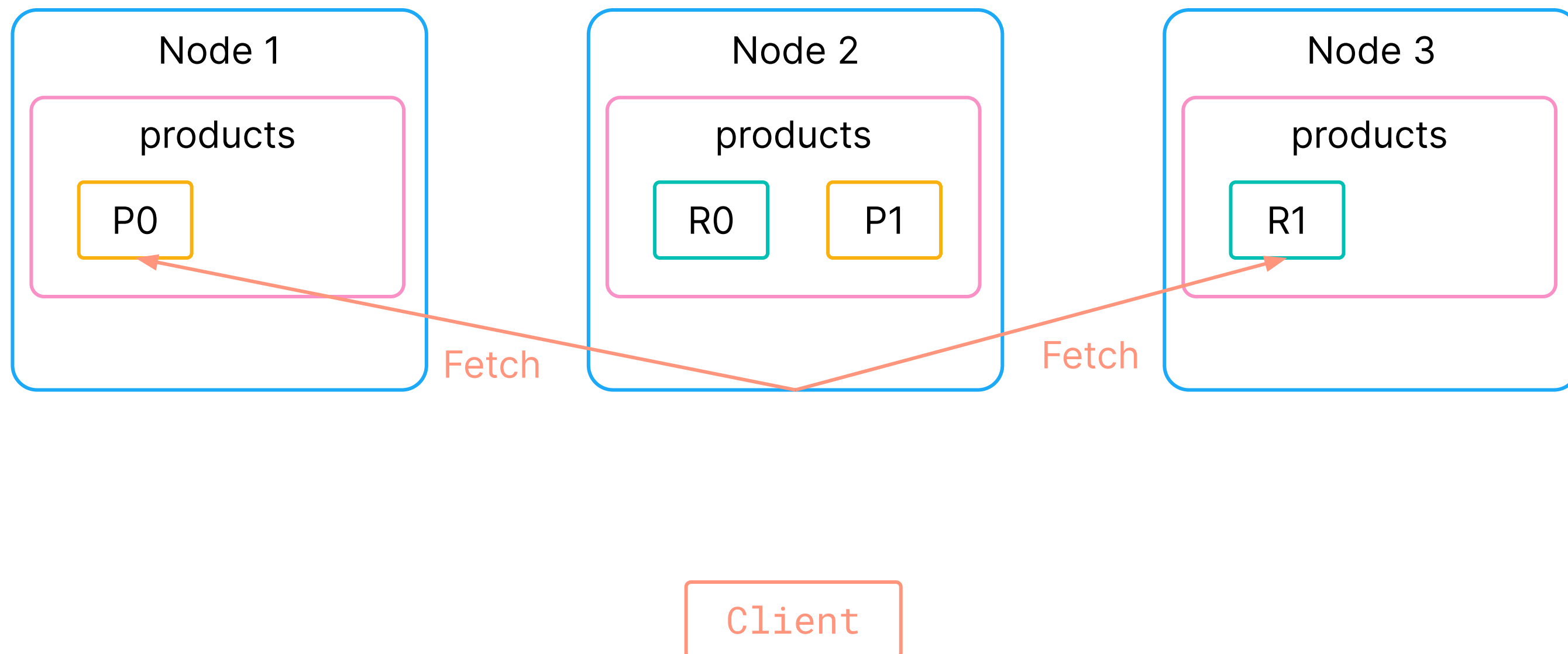
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed search

Two phase approach

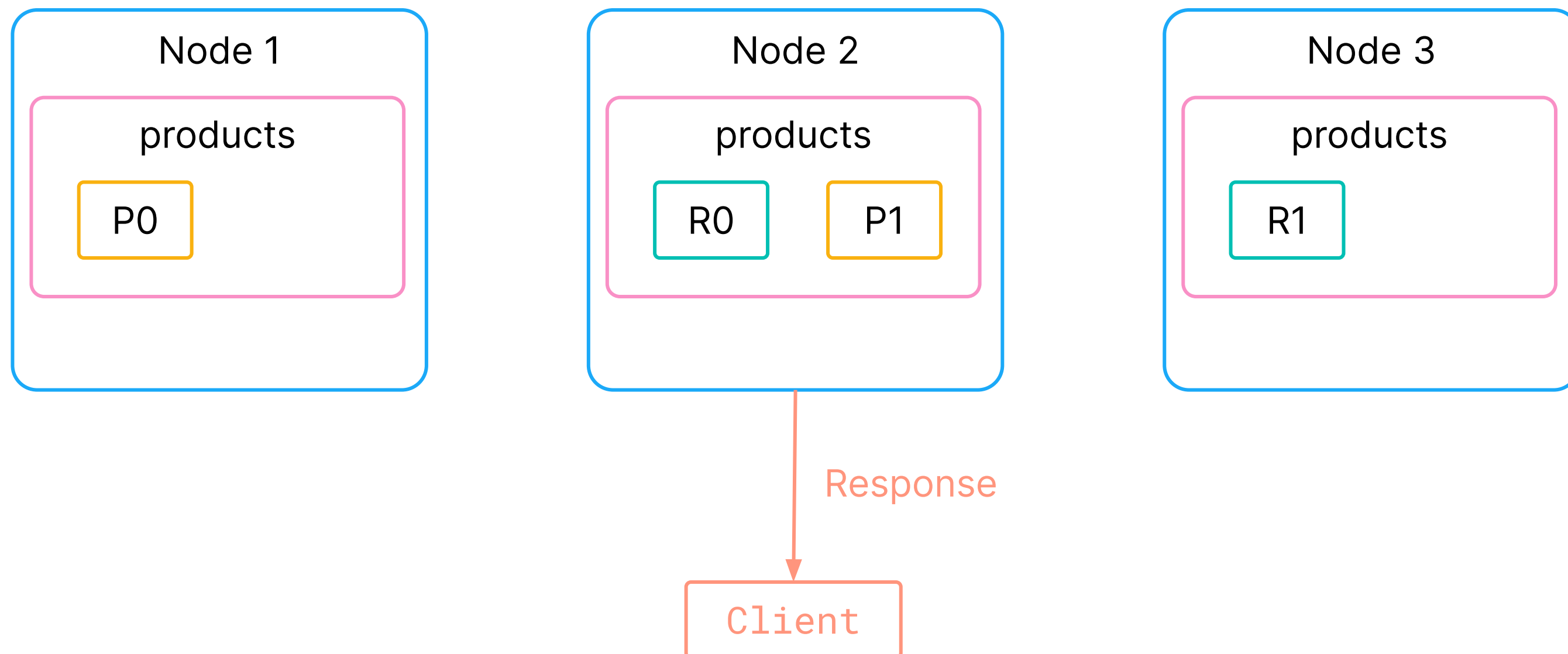
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed search

Two phase approach

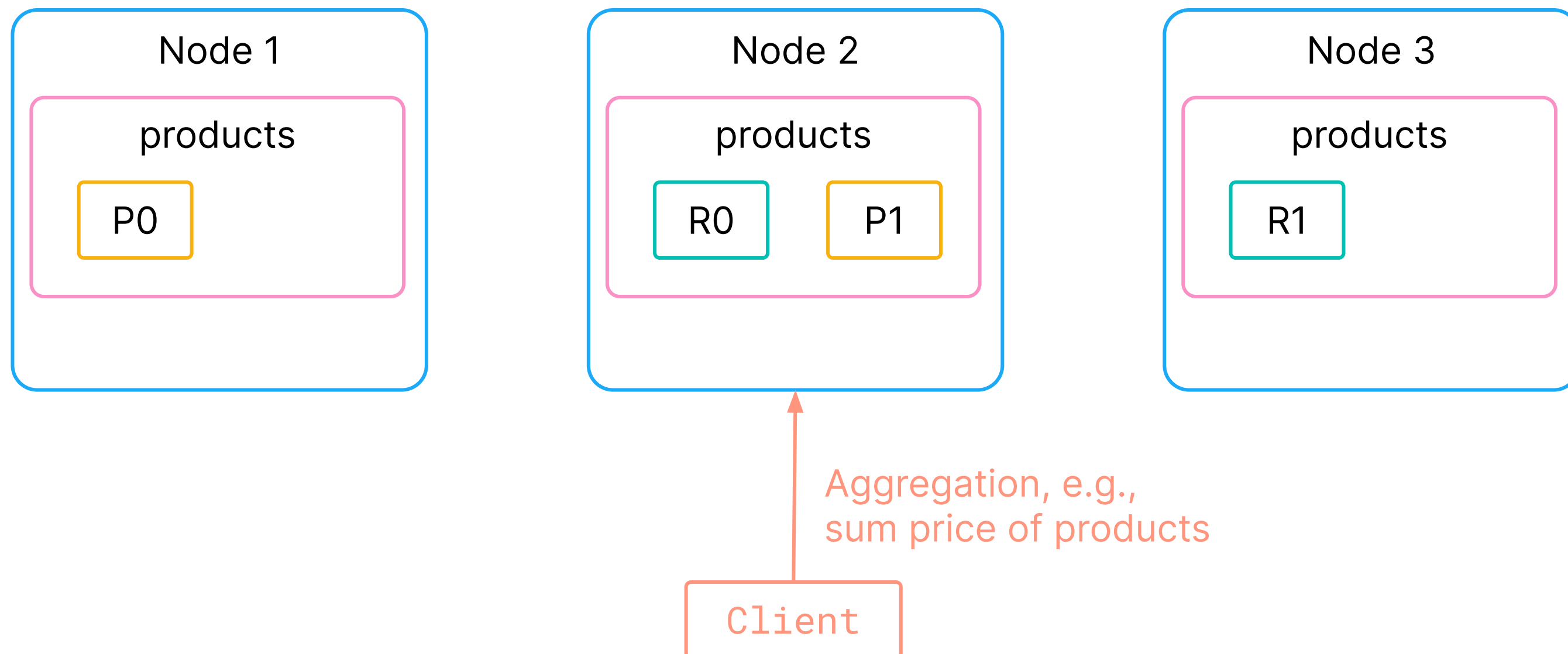
- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)



Distributed aggregations

Slice, dice and combine data to get insights

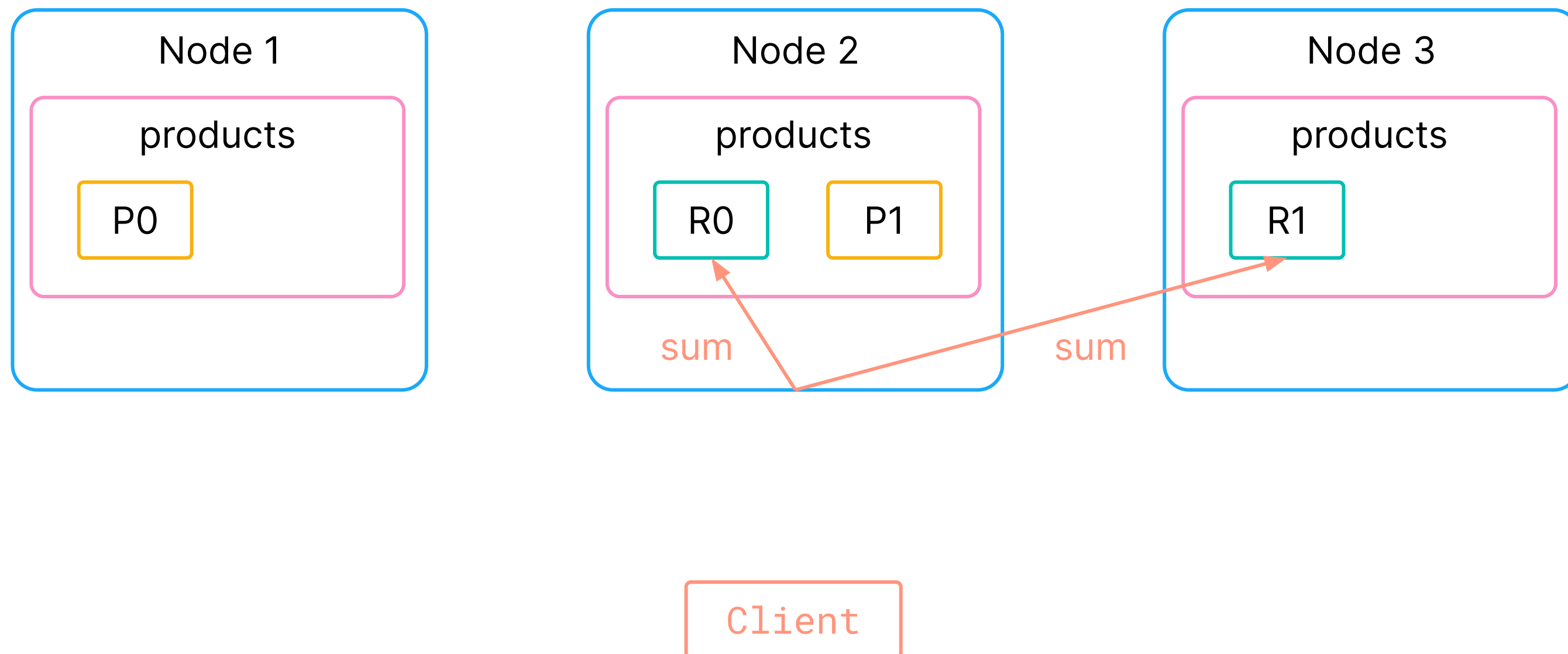
- Can be run on top of a result set of a query
- Some aggregations require your data to be central, e.g., cardinality → efficient estimations



Distributed aggregations

Slice, dice and combine data to get insights

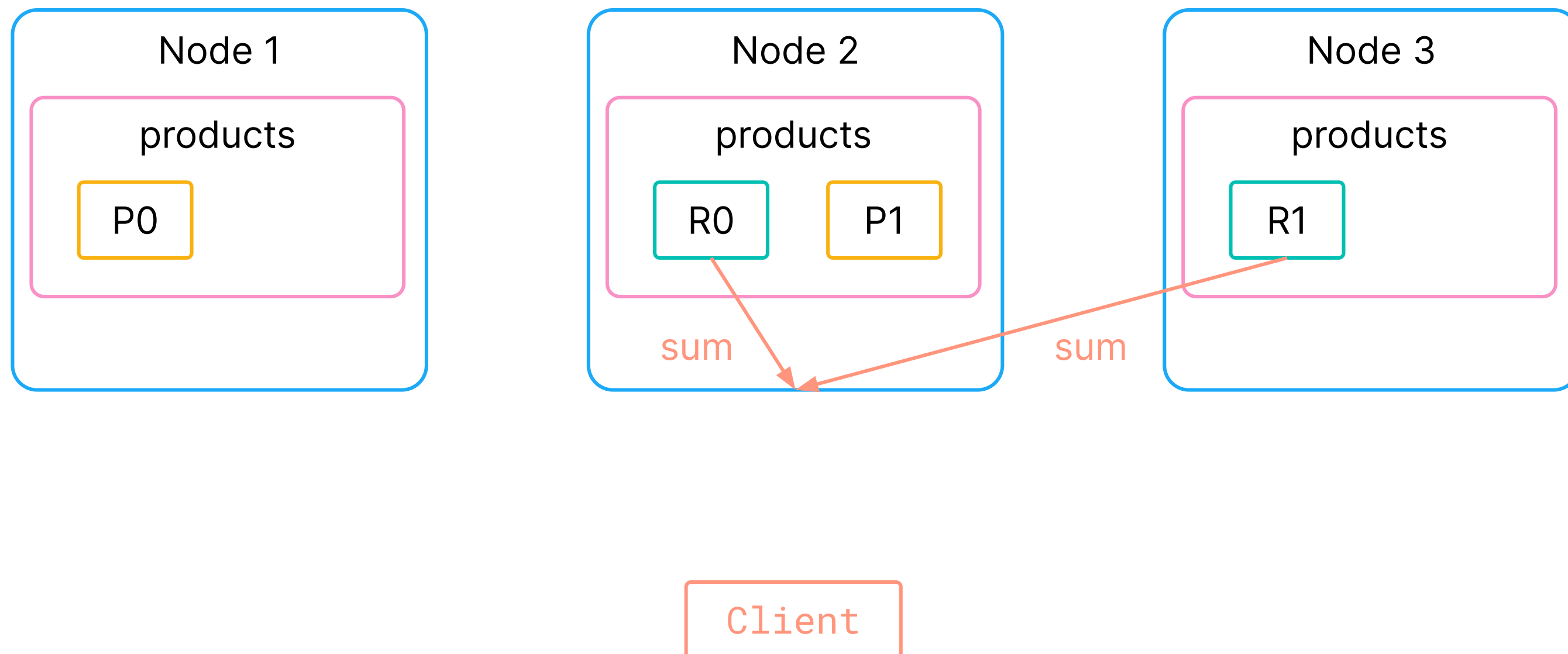
- Can be run on top of a result set of a query
- Some aggregations require your data to be central, e.g., cardinality → efficient estimations



Distributed aggregations

Slice, dice and combine data to get insights

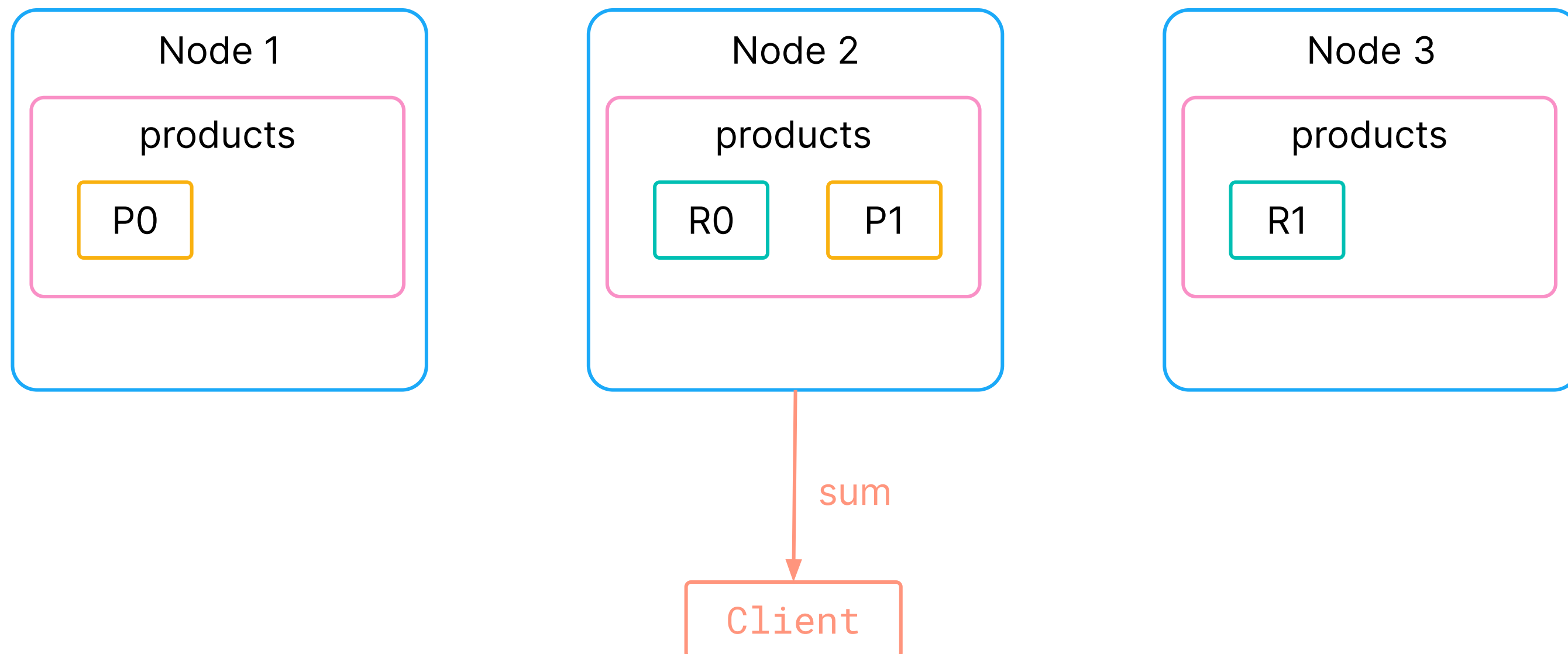
- Can be run on top of a result set of a query
- Some aggregations require your data to be central, e.g., cardinality → efficient estimations



Distributed aggregations

Slice, dice and combine data to get insights

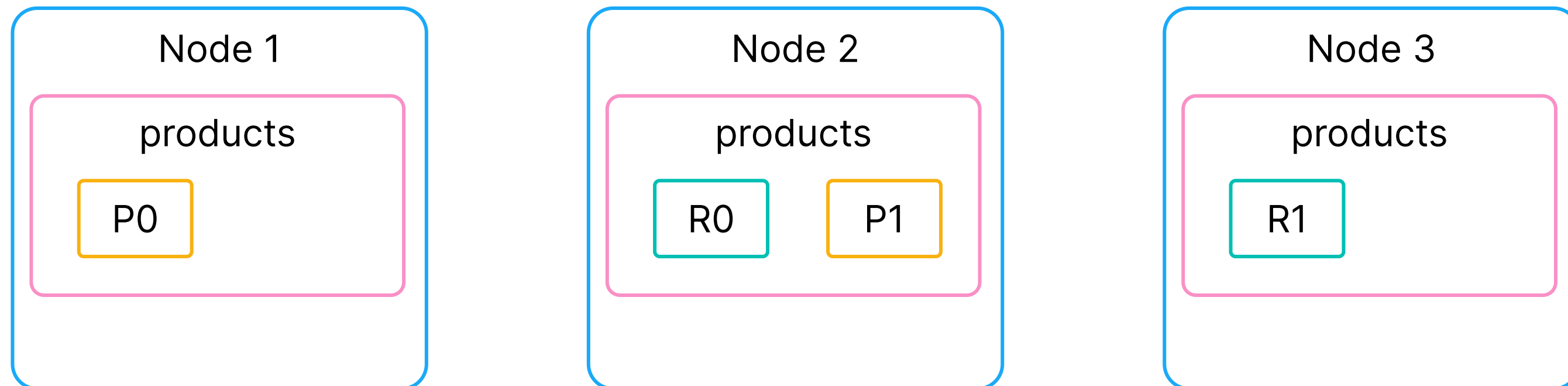
- Can be run on top of a result set of a query
- Some aggregations require your data to be central, e.g., cardinality → efficient estimations



Shard recovery

What happens to shards when a node fails?

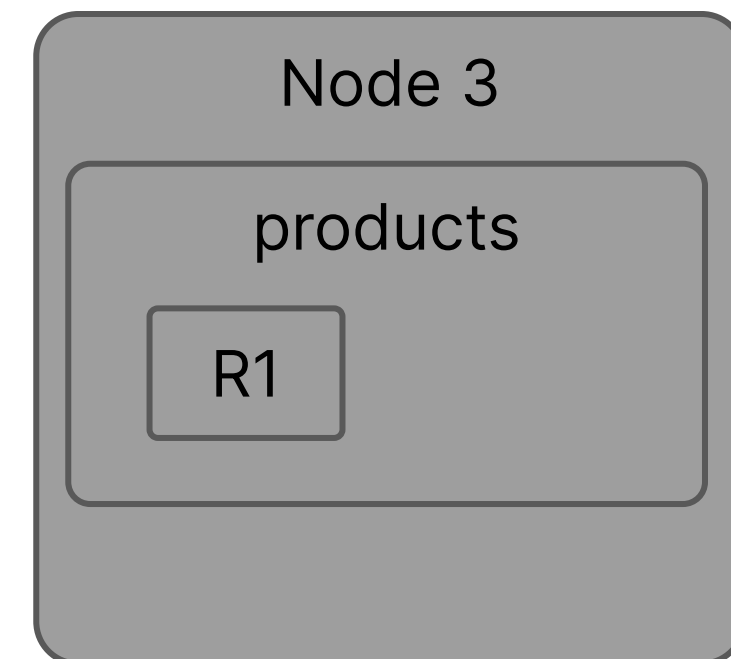
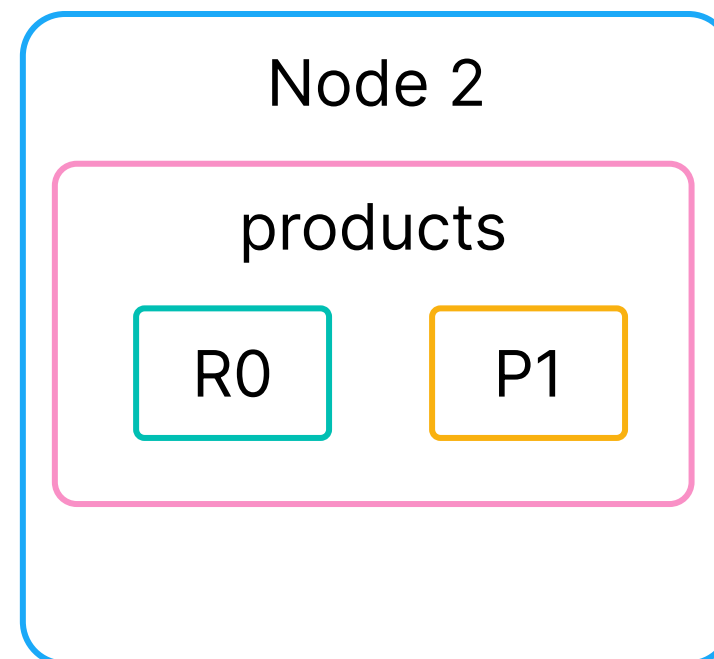
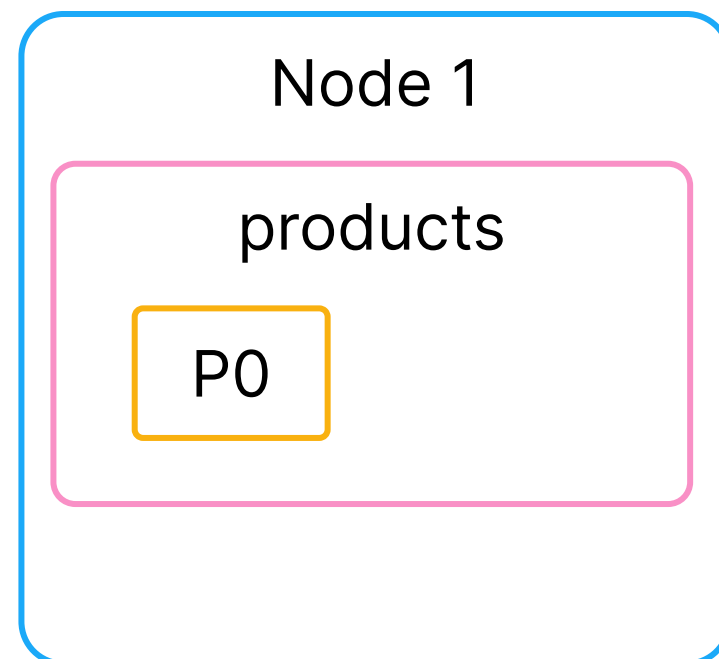
- If primary is lost → a replica is promoted to primary
- If replica is lost → it is copied from the primary to another node



Shard recovery

What happens to shards when a node fails?

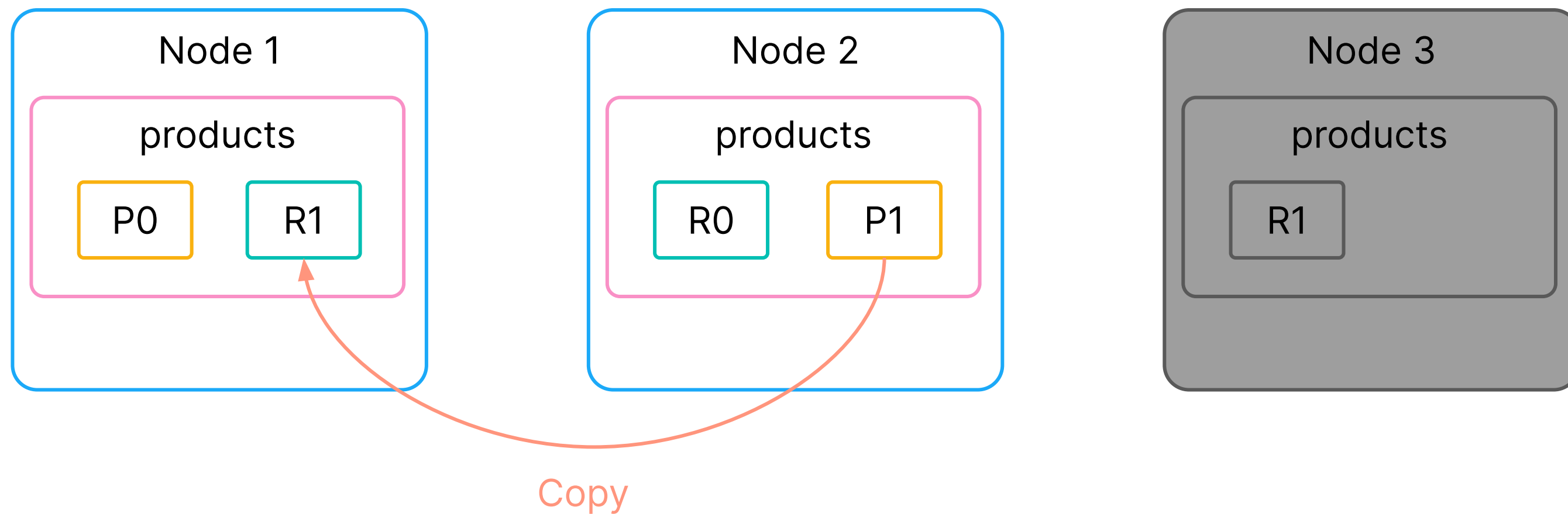
- If primary is lost → a replica is promoted to primary
- If replica is lost → it is copied from the primary to another node



Shard recovery

What happens to shards when a node fails?

- If primary is lost → a replica is promoted to primary
- If replica is lost → it is copied from the primary to another node

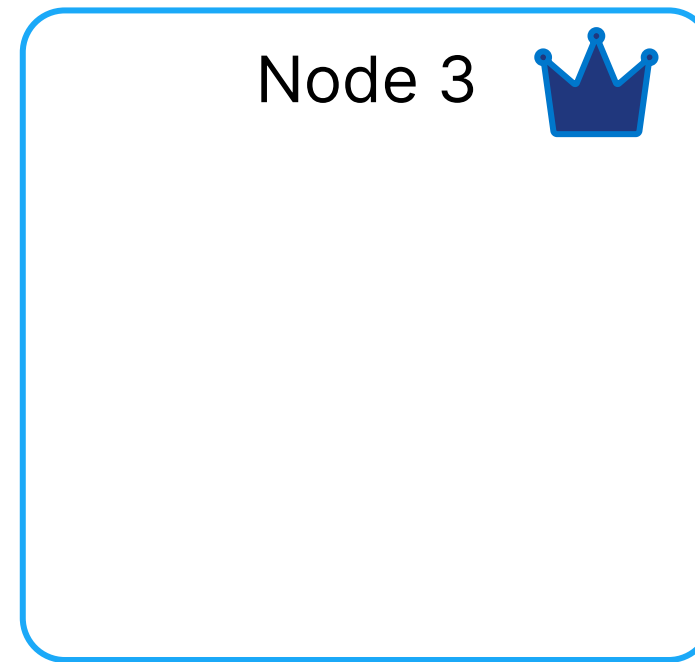
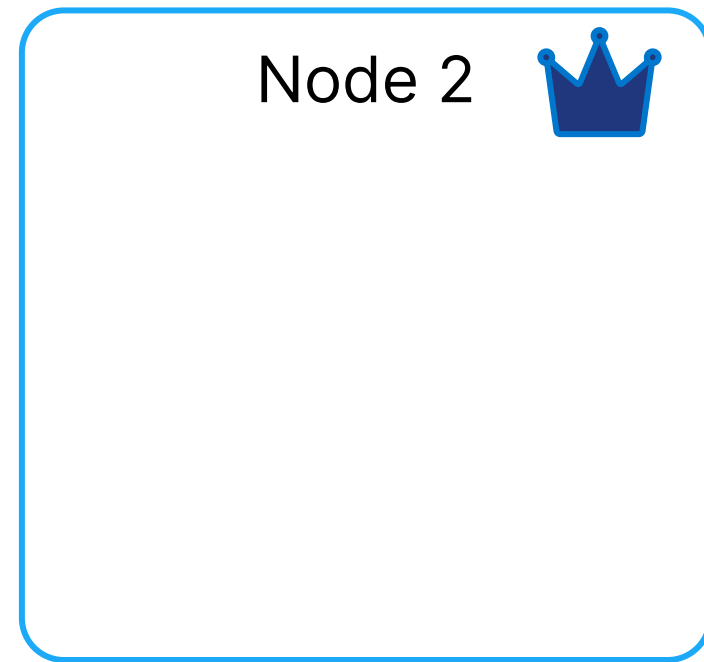
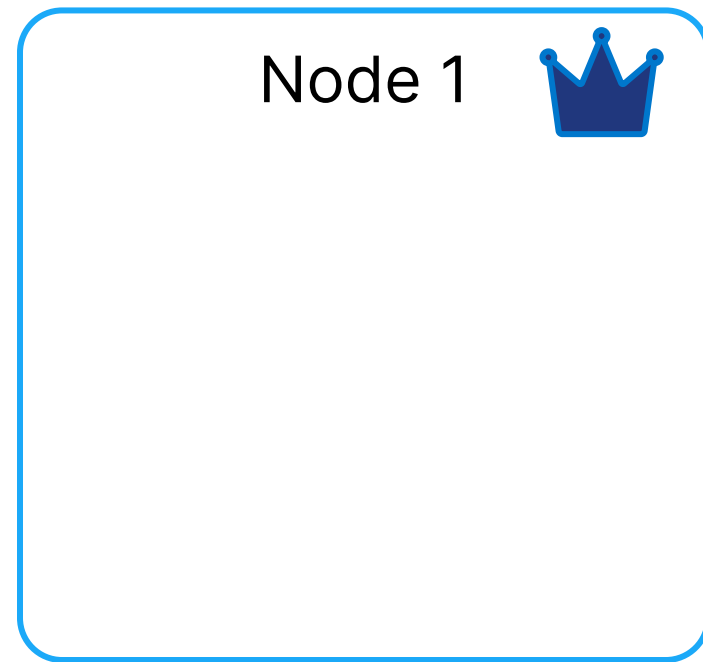


Cluster state

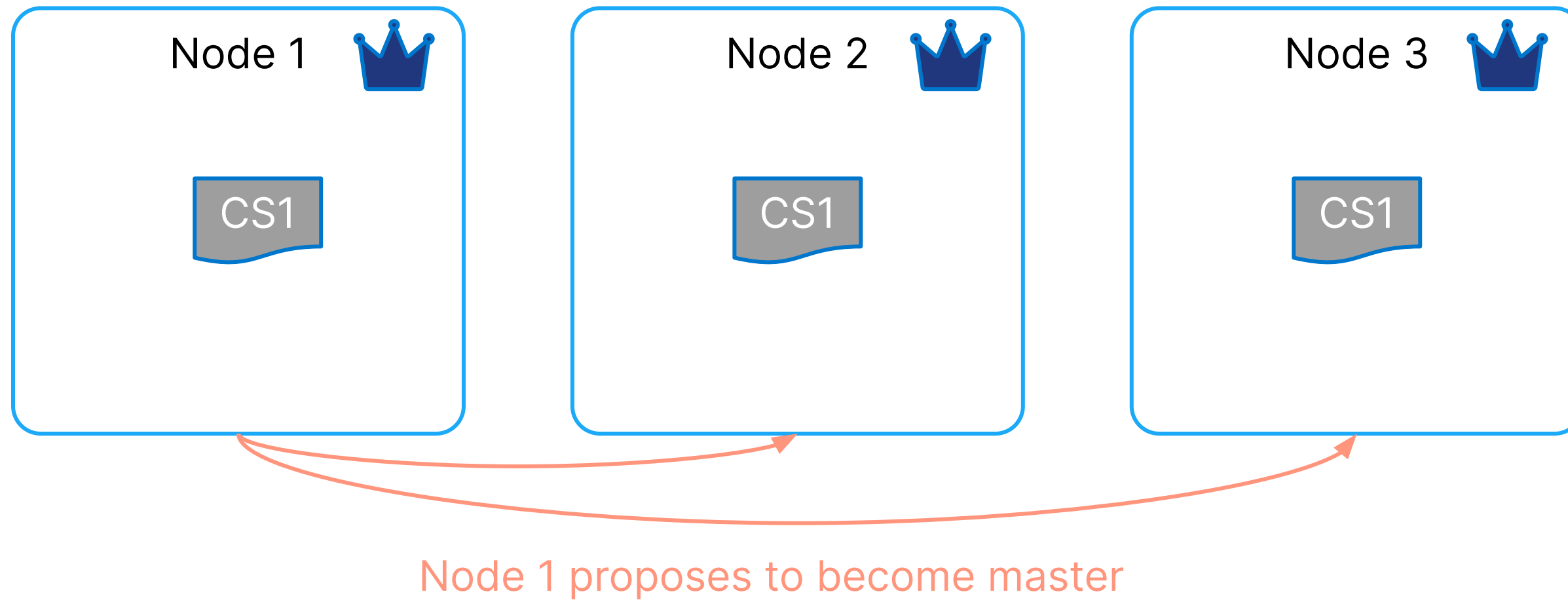
Contains the core information on the cluster membership and the indices

- One elected master node
 - Proposes updates of the cluster state based on events (e.g., node leaves, index created) and distributes it to all nodes
 - Decides data placement, where shards are moved and replicated
 - Node health checks
 - Not needed for reading/writing
- Consensus is used across a small set of master-eligible nodes
 - To establish the cluster state update proposals (quorum required)
 - To re-elect a master node on failure

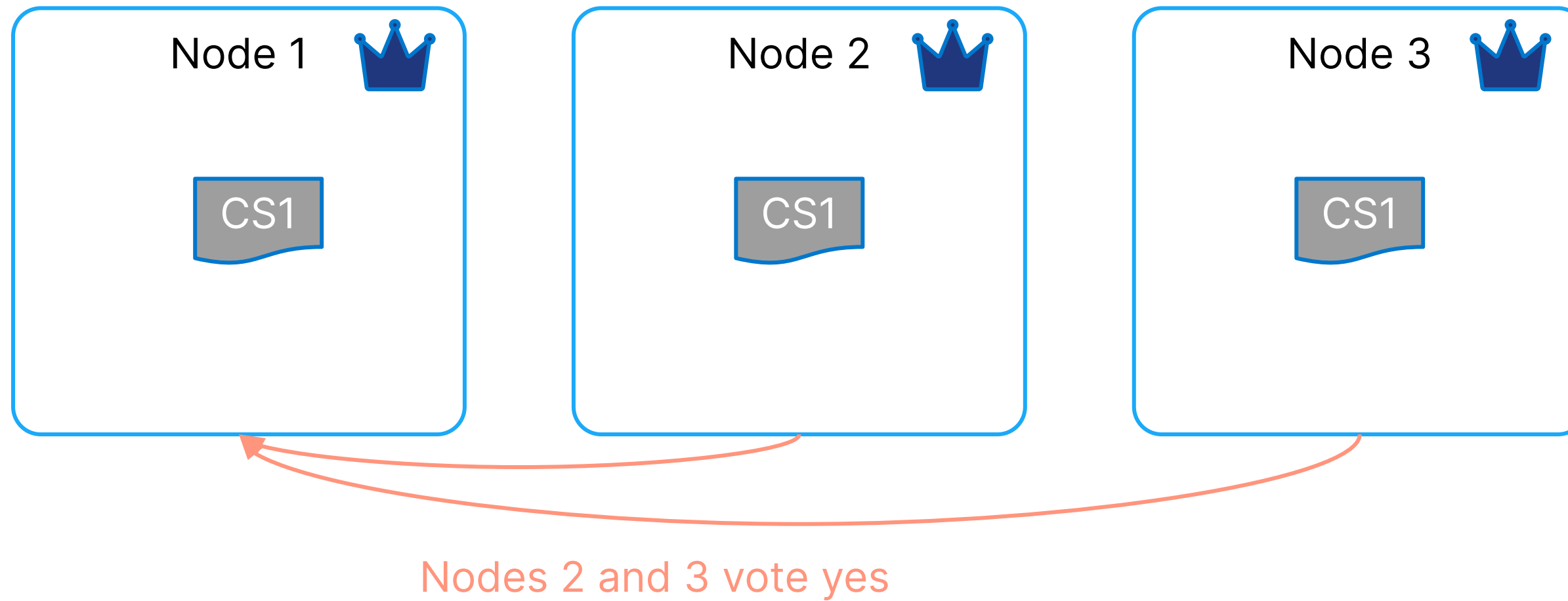
Cluster state – bootstrapping



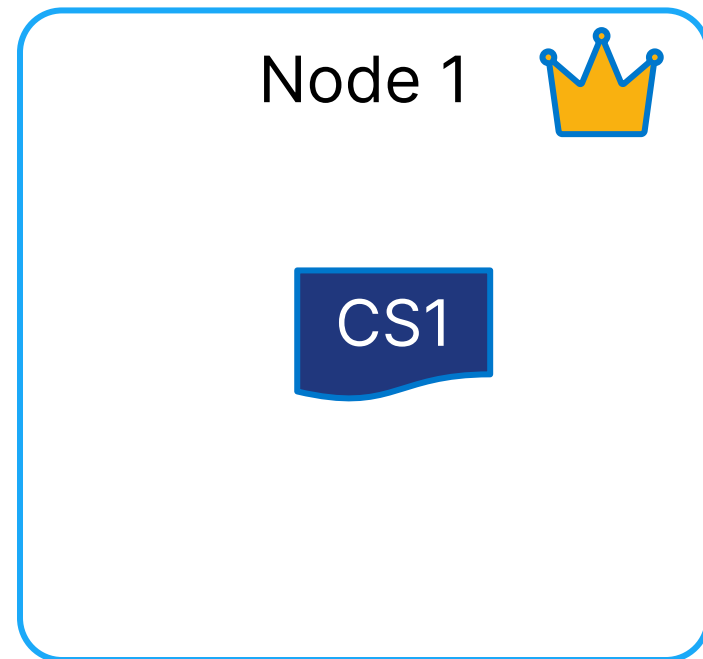
Cluster state – election



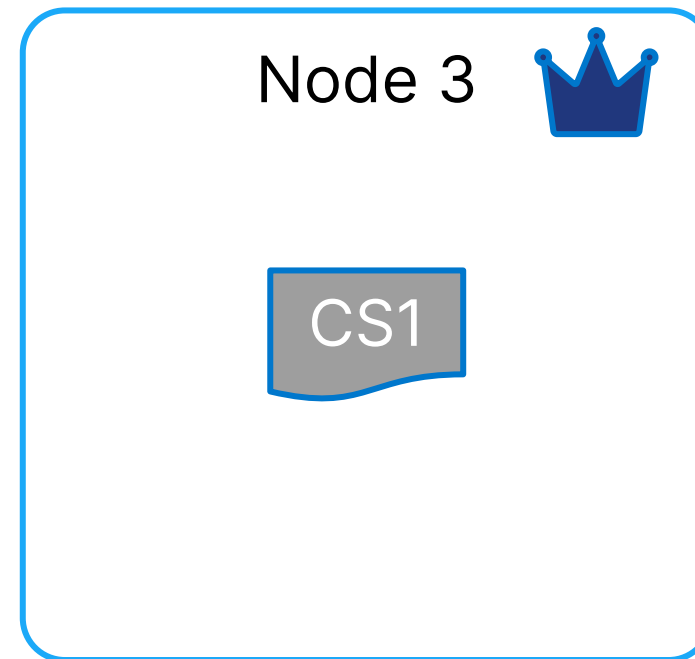
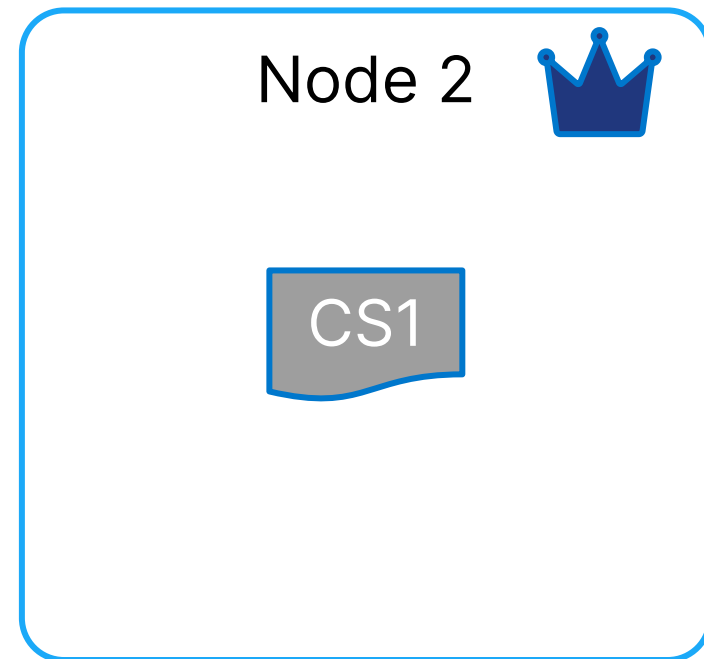
Cluster state – election



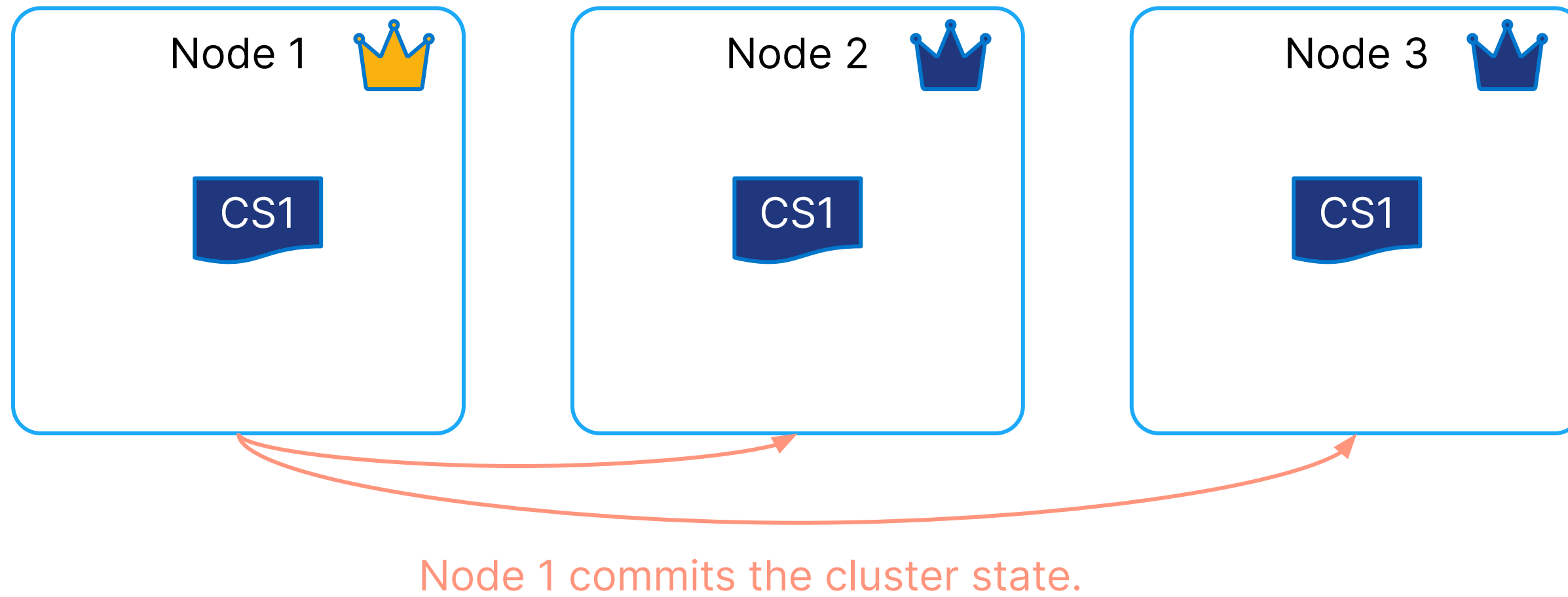
Cluster state – election



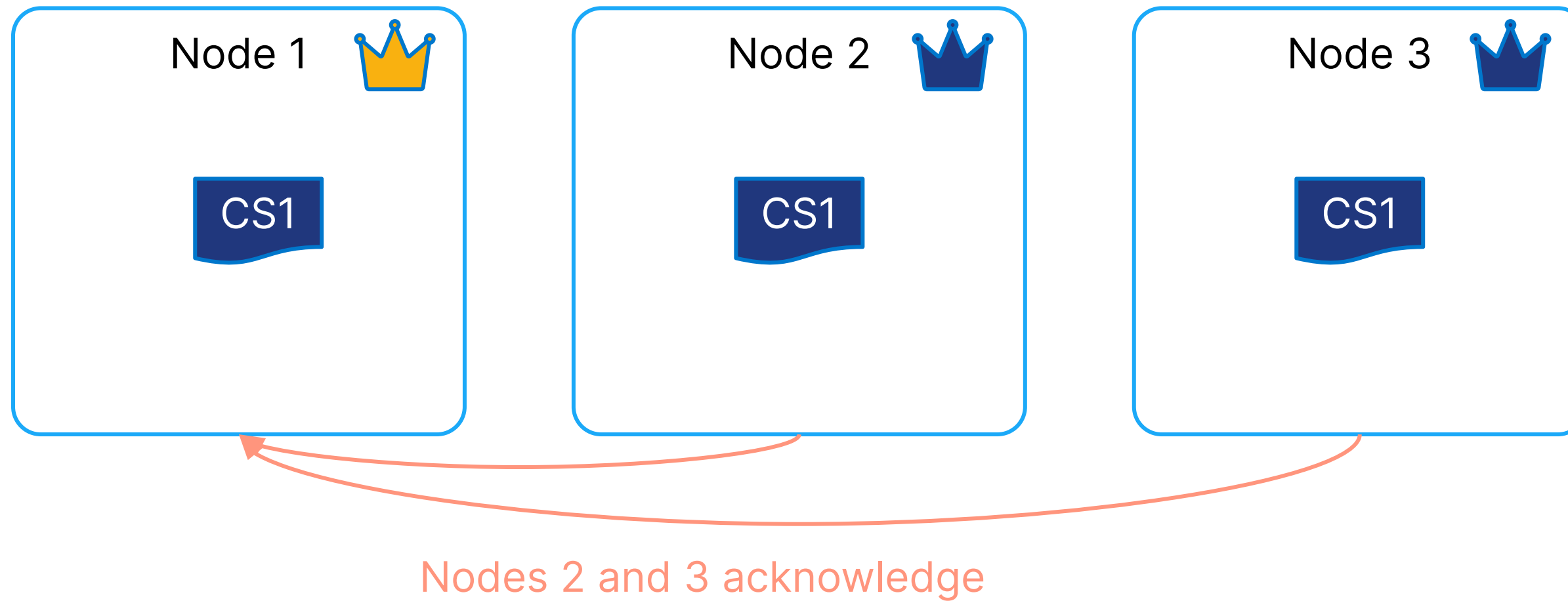
Node 1 becomes master



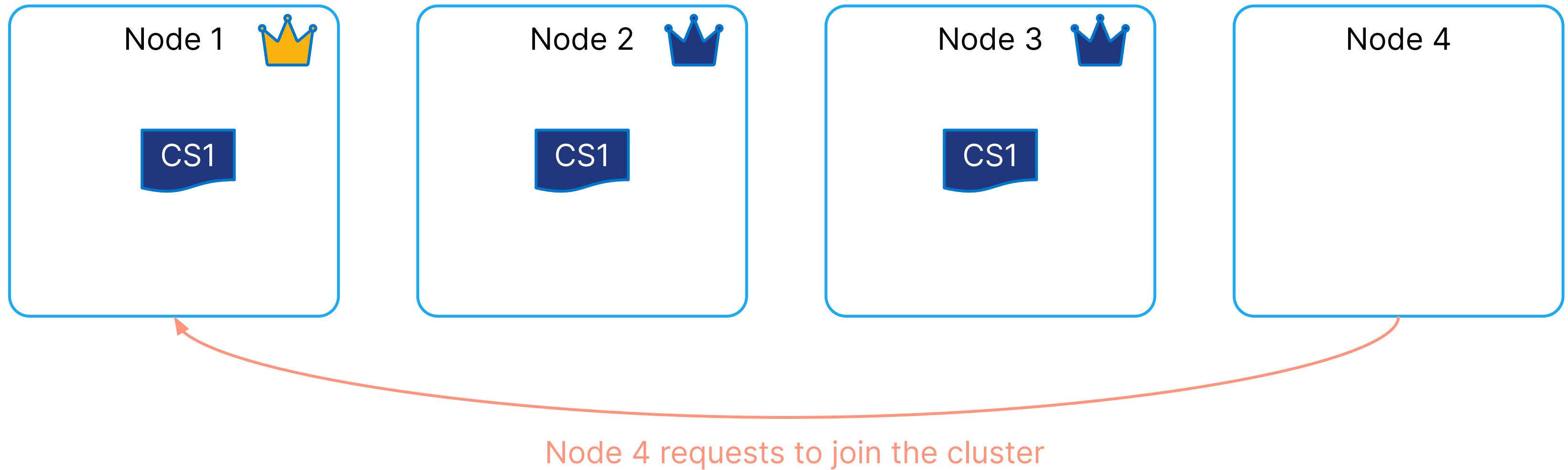
Cluster state – election



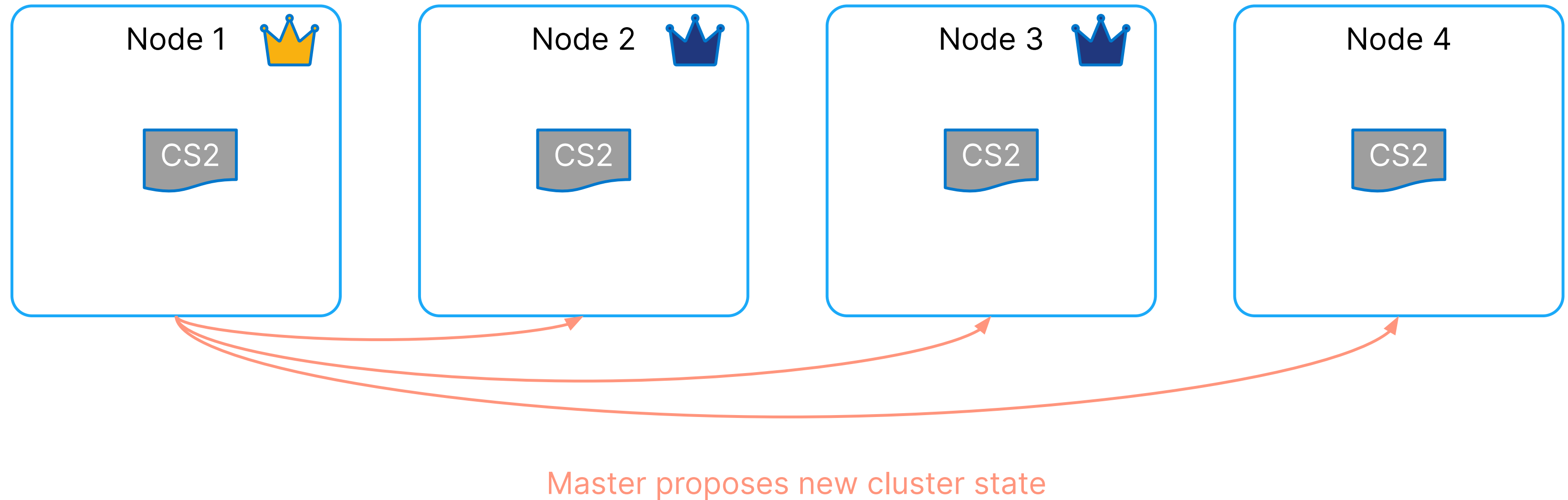
Cluster state – election



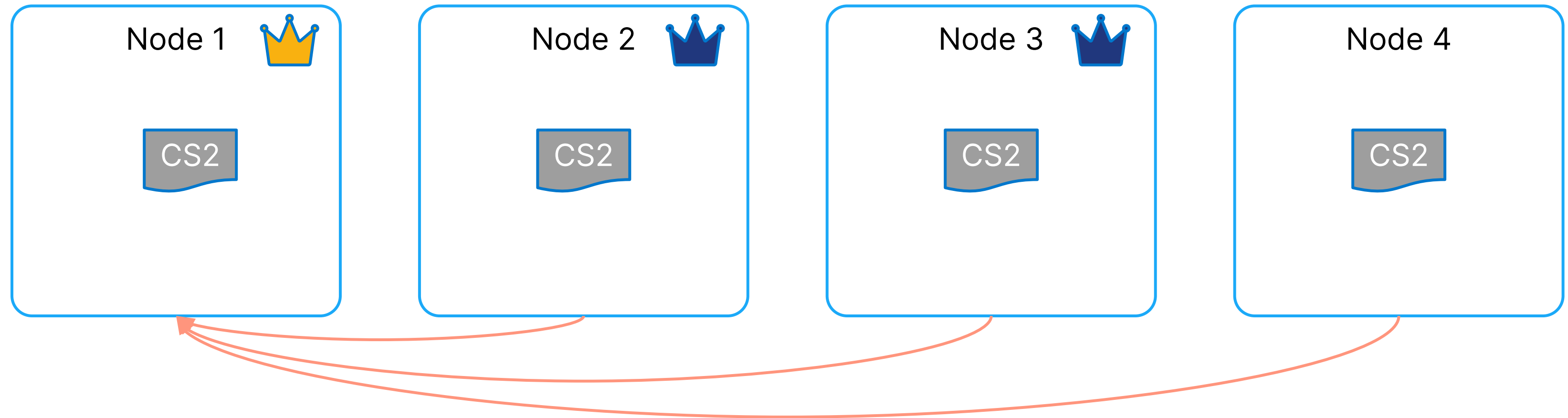
Cluster state – node joins



Cluster state – node joins

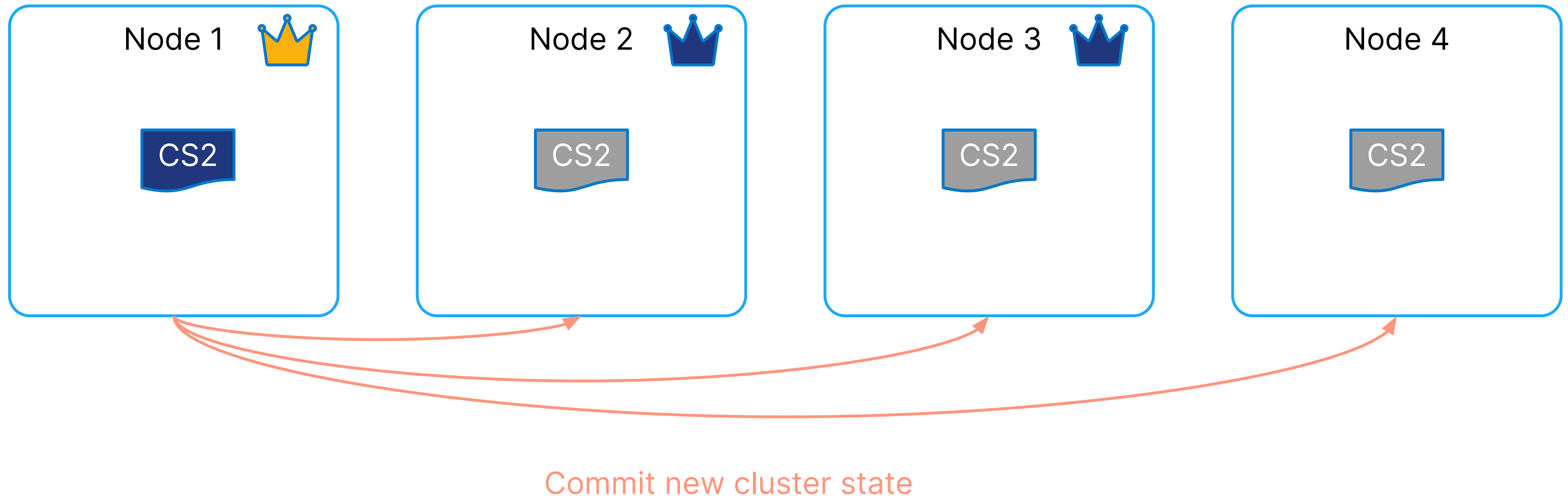


Cluster state – node joins

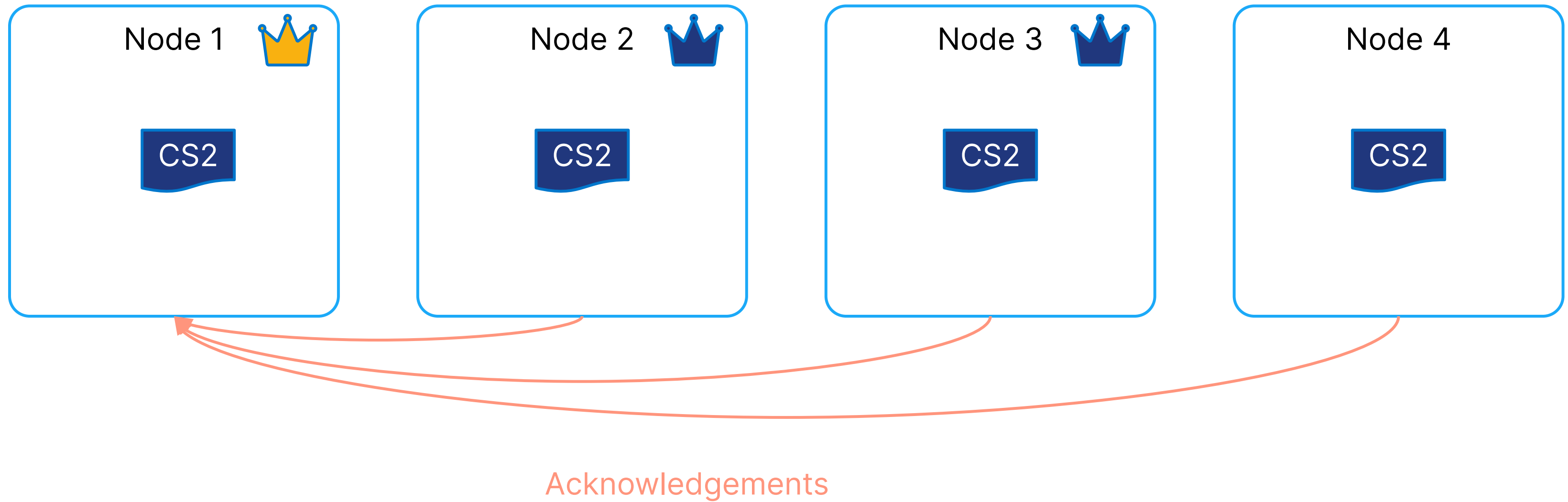


Acknowledgements (including yes votes from master-eligible nodes)

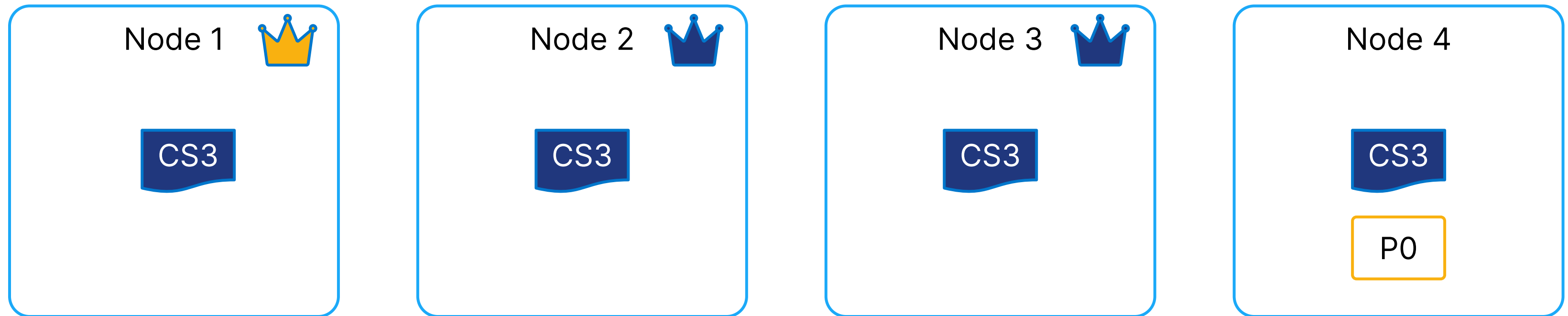
Cluster state – node joins



Cluster state – node joins



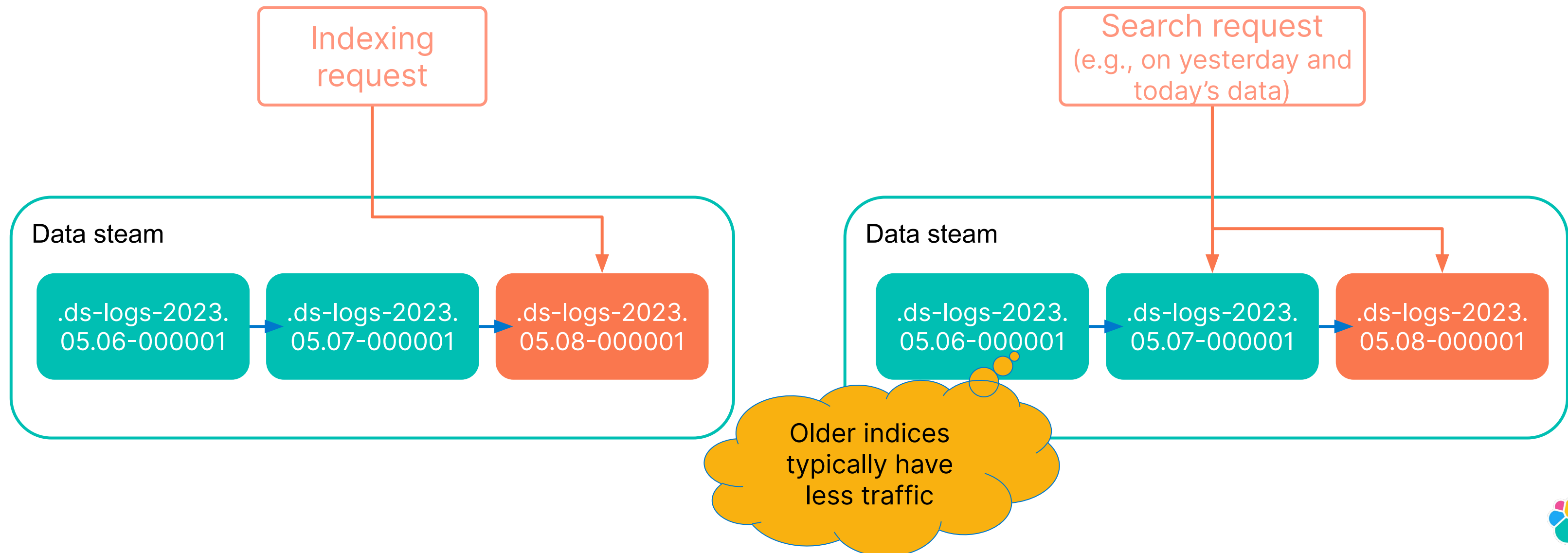
Cluster state – index created (fast forwarded)



Scaling with data streams

Store append-only time series data across multiple indices

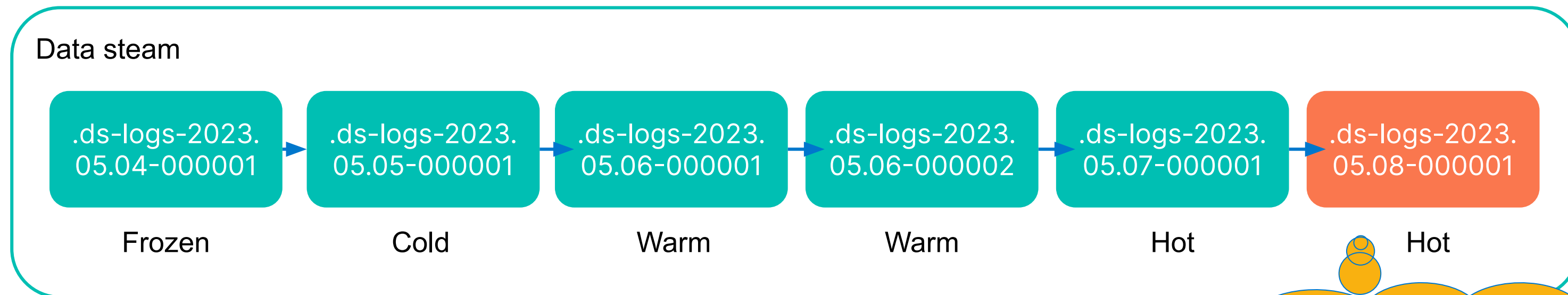
- Gives a single named resource for requests
- Rollover indices based on age and/or size



Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

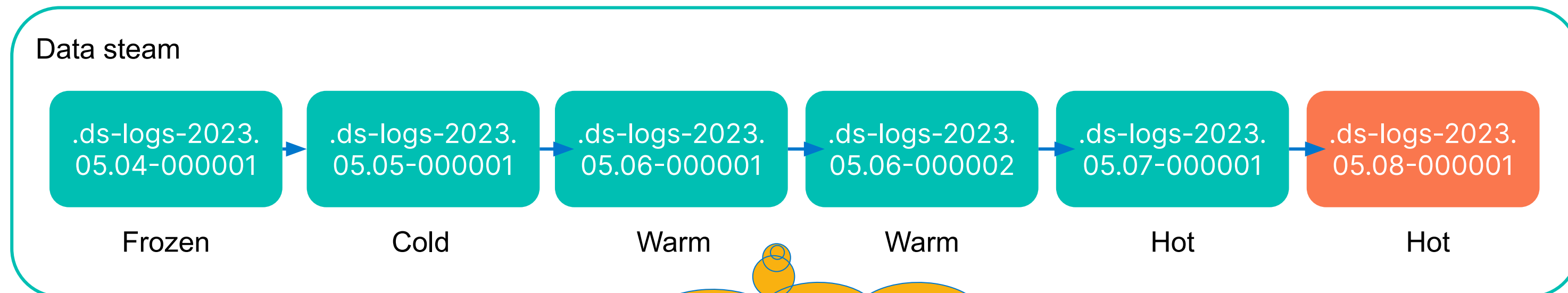


Most frequently accessed read and written data. E.g., 2 replicas, SSD disks, beefier machines.

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

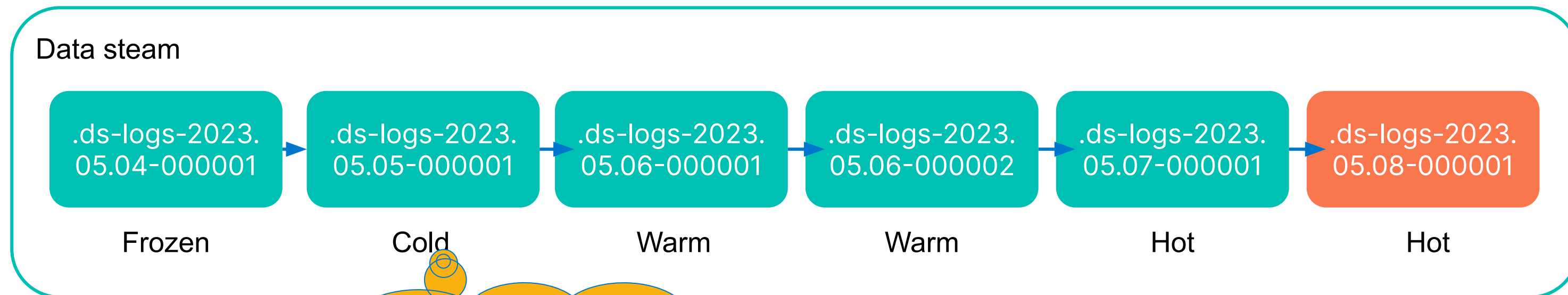


Less frequently accessed
read and written data. E.g.,
1 replica, HDD disks,
cheaper machines.

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

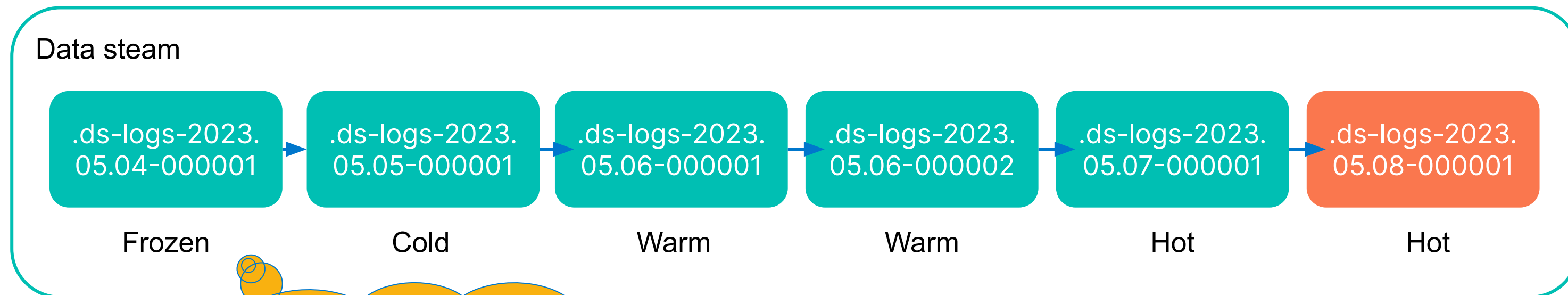


Read-only data. Searchable snapshots stored on a cloud object store (e.g., S3) and fully cached on disk. No replicas.

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

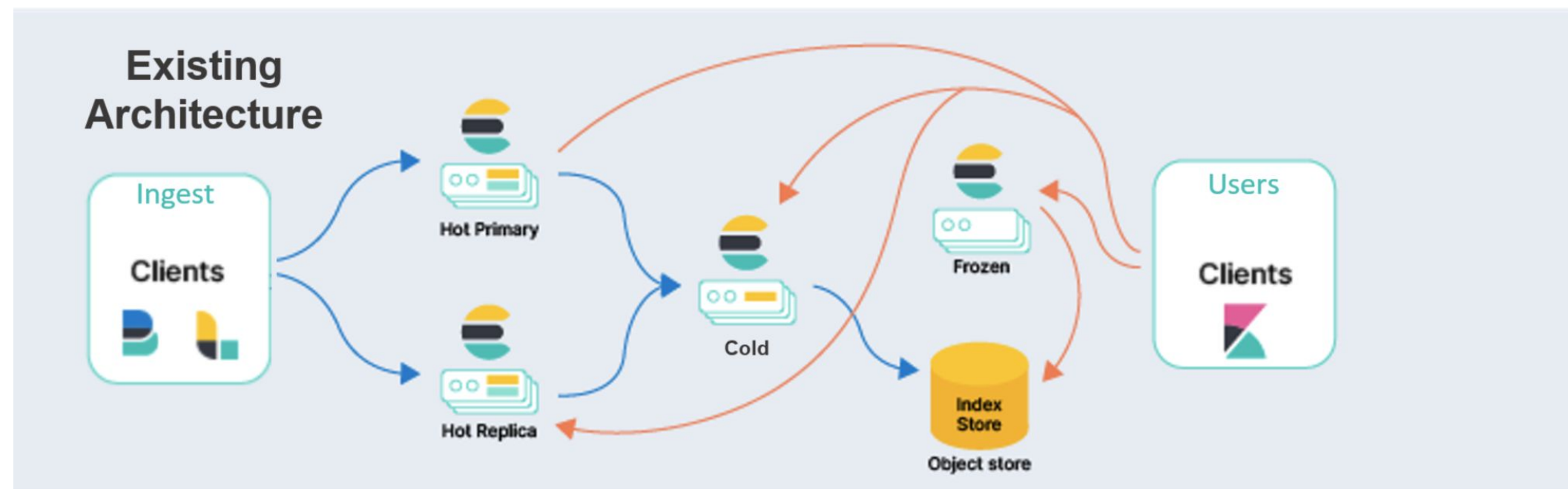


Read-only data with slower queries. Partially cached (on disk) searchable snapshots from cloud object store.


Future: Scaling with Serverless

Stateless: store indices or cluster state on a cloud object store

- Fault tolerance via cloud → no replicas
- Just two data tiers: indexing and search, which can be scaled independently



75% indexing throughput improvement



Summary

- Elasticsearch is an unparalleled scalable search & analytics engine:
 - Scaling Lucene instances with multiple shards
 - Distributed searches and aggregations over multiple nodes
 - Master node directs the cluster state updates, via an efficient consensus protocol
 - Time based data streams are scaled with Index Lifecycle Management
 - Future: serverless vision based on keeping state on cloud object store
- Example of scalability
 - Adobe ([2018](#)): 400 VMs, 10B docs, 600q/sec, 6000docs/sec
 - Elastic Internal Observability Clusters ([2022](#)): 207 clusters (through [Cross-Cluster Search](#)), 1.2 trillion docs, 300TB events/day, 4 cloud providers (53 regions)
 - [Elastic Stack & Cloud: Start from AWS in 3 clicks, learn about Elastic's serverless vision, Benchmark-driven optimizations, A new era for cluster coordination in Elasticsearch, Autoscale your Elastic Cloud data and machine learning nodes, How many shards should I have in my Elasticsearch cluster?](#)

Community, culture and careers

- Vibrant community
 - Community portal → www.elastic.co/community
 - Elastic Community on Slack → ela.st/slack
 - Community videos → ela.st/community-youtube
 - Discussion forums → discuss.elastic.co
 - Elastic Contributor Program (e.g., earn training) → elastic.co/community/contributor
 - Community events & groups across the globe → community.elastic.co
 - Newsletter. News like [AI-ready vector search](#) with exact match and approximate [kNN](#) search, or [Integrate with ChatGPT](#).
- Careers → elastic.co/about/careers
 - [Elastic Source Code](#), remote, 2600+ employees across 40+ countries
- Subscribe for next meetup.com/greece-elastic event
 - Which is expected in Athens in **June!**

Thank you! Questions?

Iraklis Psaroudakis
www.kingherc.com