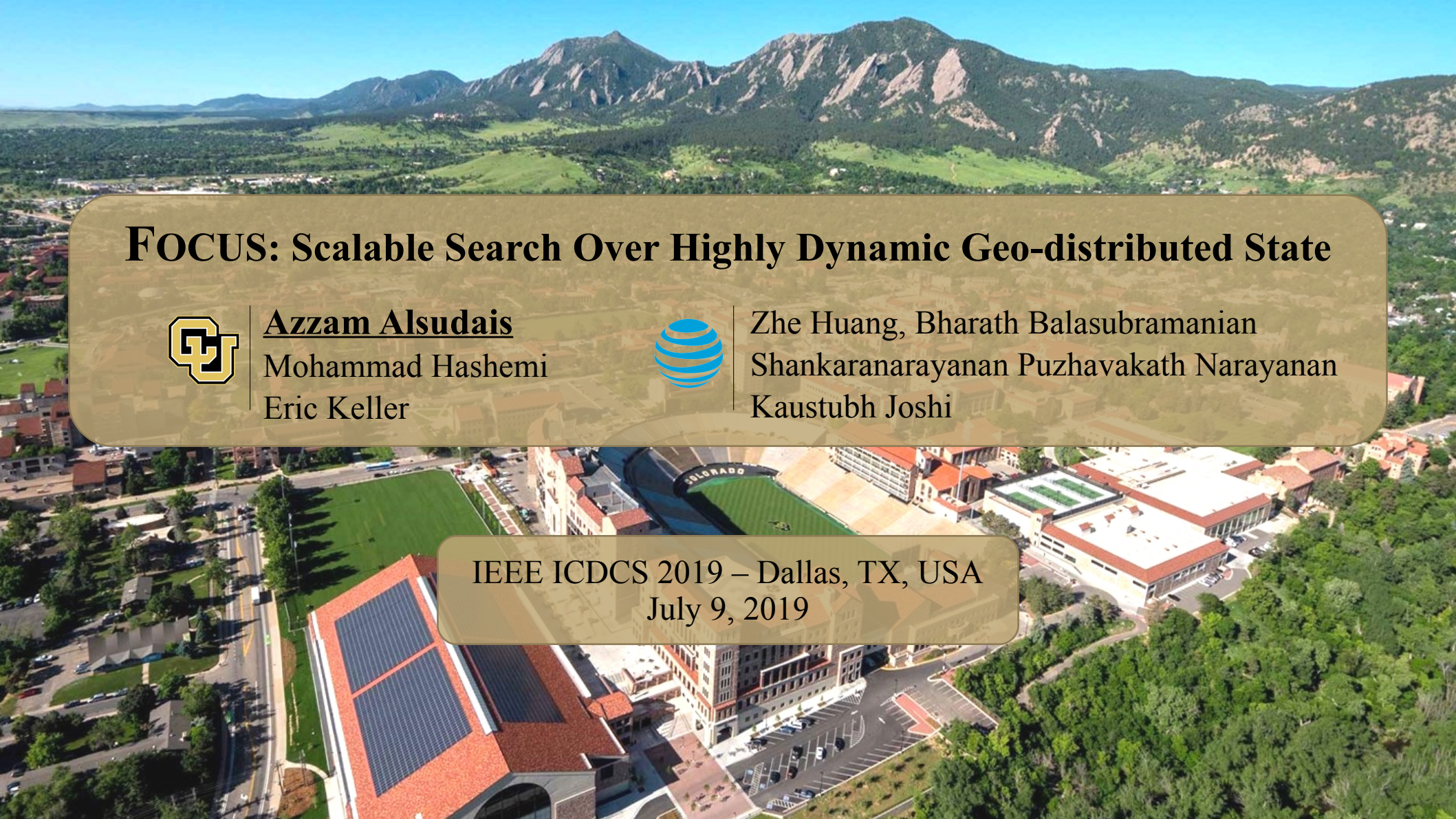


FOCUS: Scalable Search Over Highly Dynamic Geo-distributed State

Azzam Alsudais
Mohammad Hashemi
Eric Keller

Zhe Huang, Bharath Balasubramanian
Shankaranarayanan Puzhavakath Narayanan
Kaustubh Joshi

IEEE ICDCS 2019 – Dallas, TX, USA
July 9, 2019

Why do systems need to find nodes?

Use Cases

Cloud Management

Use Cases

Cloud Management

VM Provisioning

Use Cases

Cloud Management

VM Provisioning

VM Migration

Use Cases

Cloud Management

VM Provisioning

VM Migration

Monitoring

Use Cases

Cloud Management

VM Provisioning

VM Migration

Monitoring

NVF Automation

Use Cases

Cloud Management

VM Provisioning

VM Migration

Monitoring

NVF Automation

Geo-distributed VNF

Service Chain Placement

Use Cases

Cloud Management

VM Provisioning

VM Migration

Monitoring

NVF Automation

Geo-distributed VNF

Service Chain Placement

Required information is assumed available

Use Cases

Cloud Management

VM Provisioning

VM Migration

Monitoring

NVF Automation

Geo-distributed VNF

Service Chain Placement

Required information is assumed available

But **HOW** is node information collected?

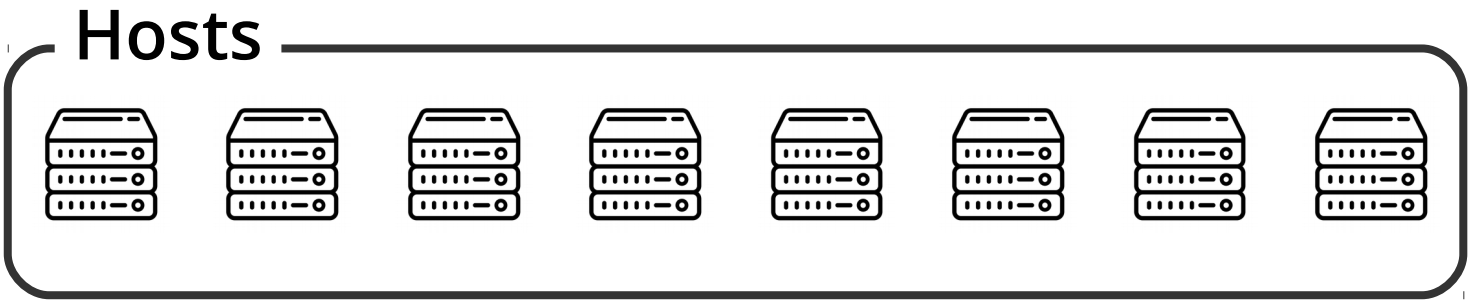
Outline

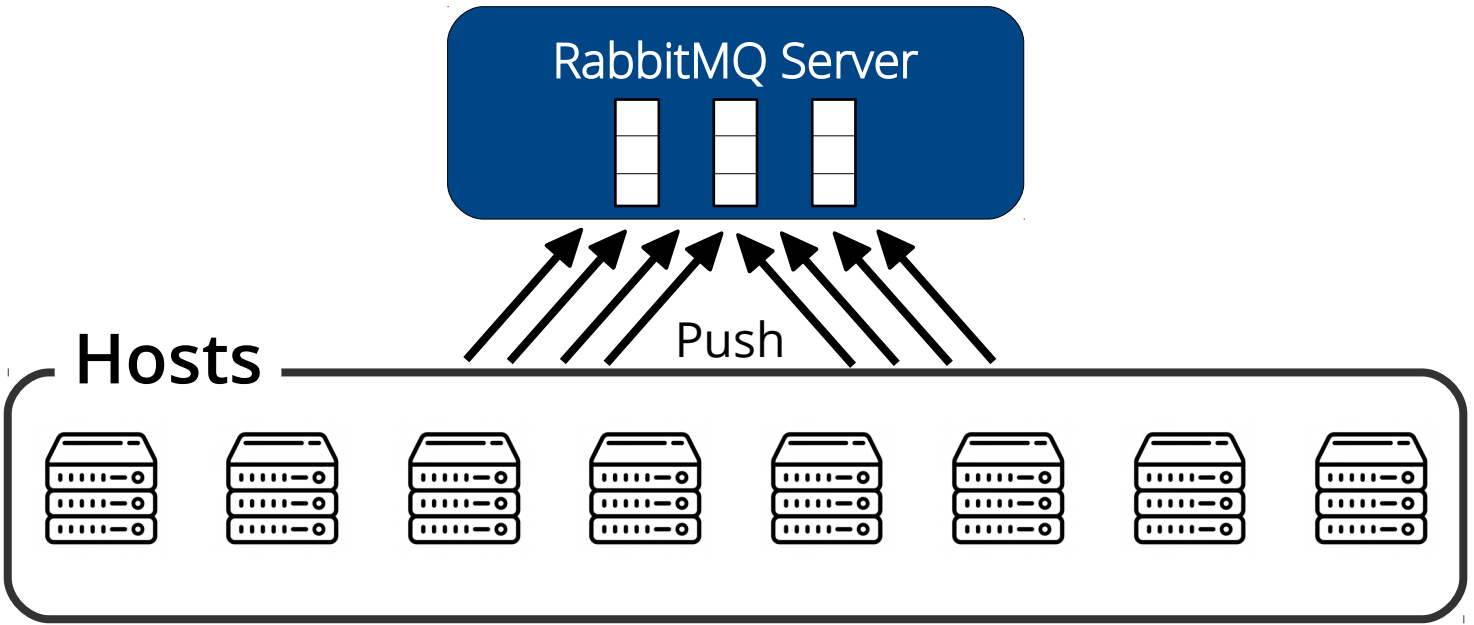
- How do systems find nodes?
- Limitations of current approaches
- FOCUS design
- Evaluation
- Conclusion

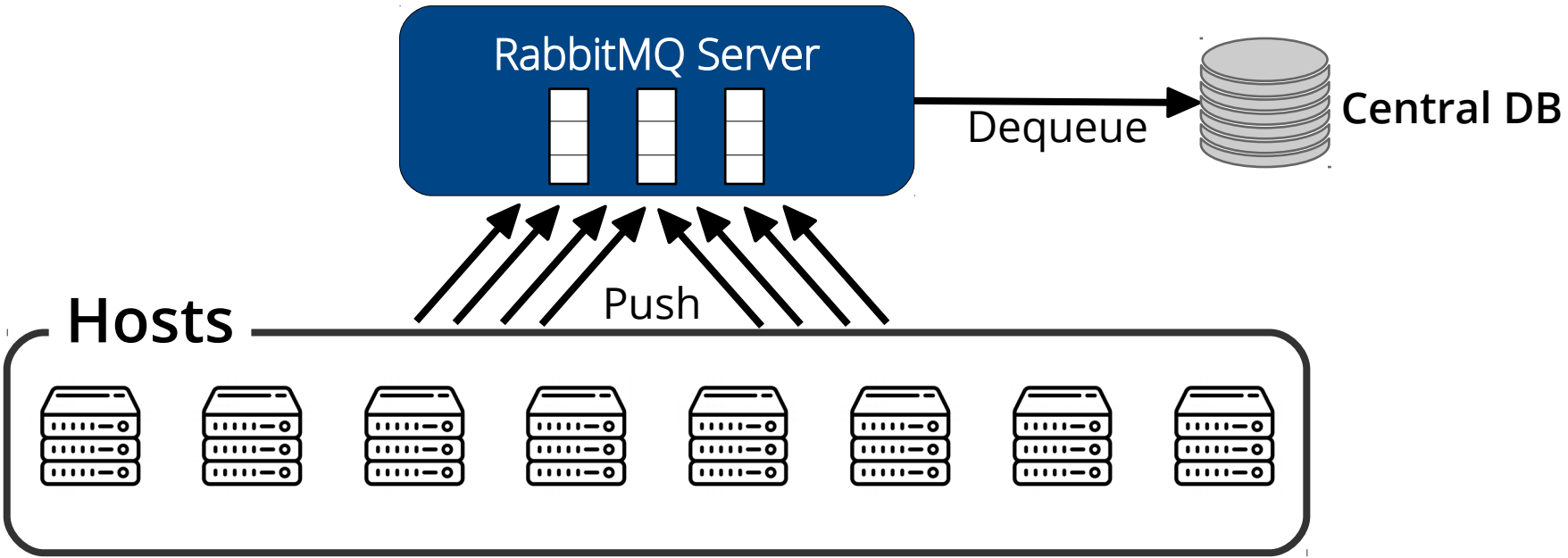
Looking Under The Hood

How do systems search for nodes?

Node finding in  openstack™

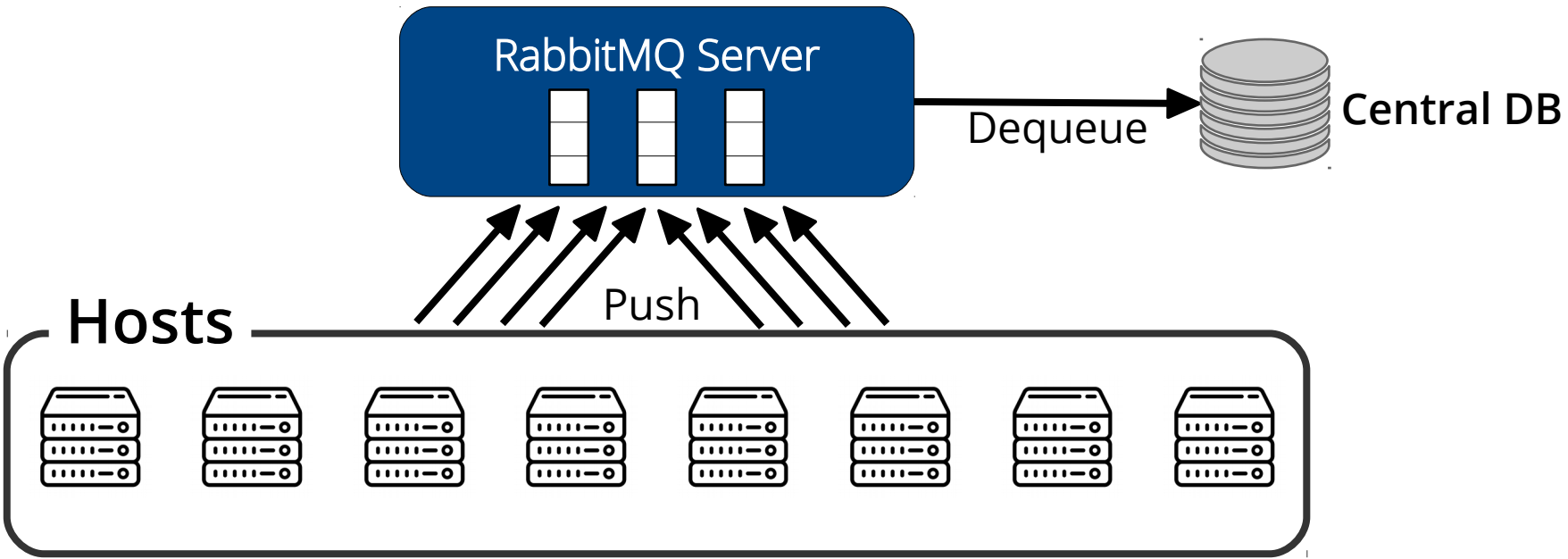

Hosts

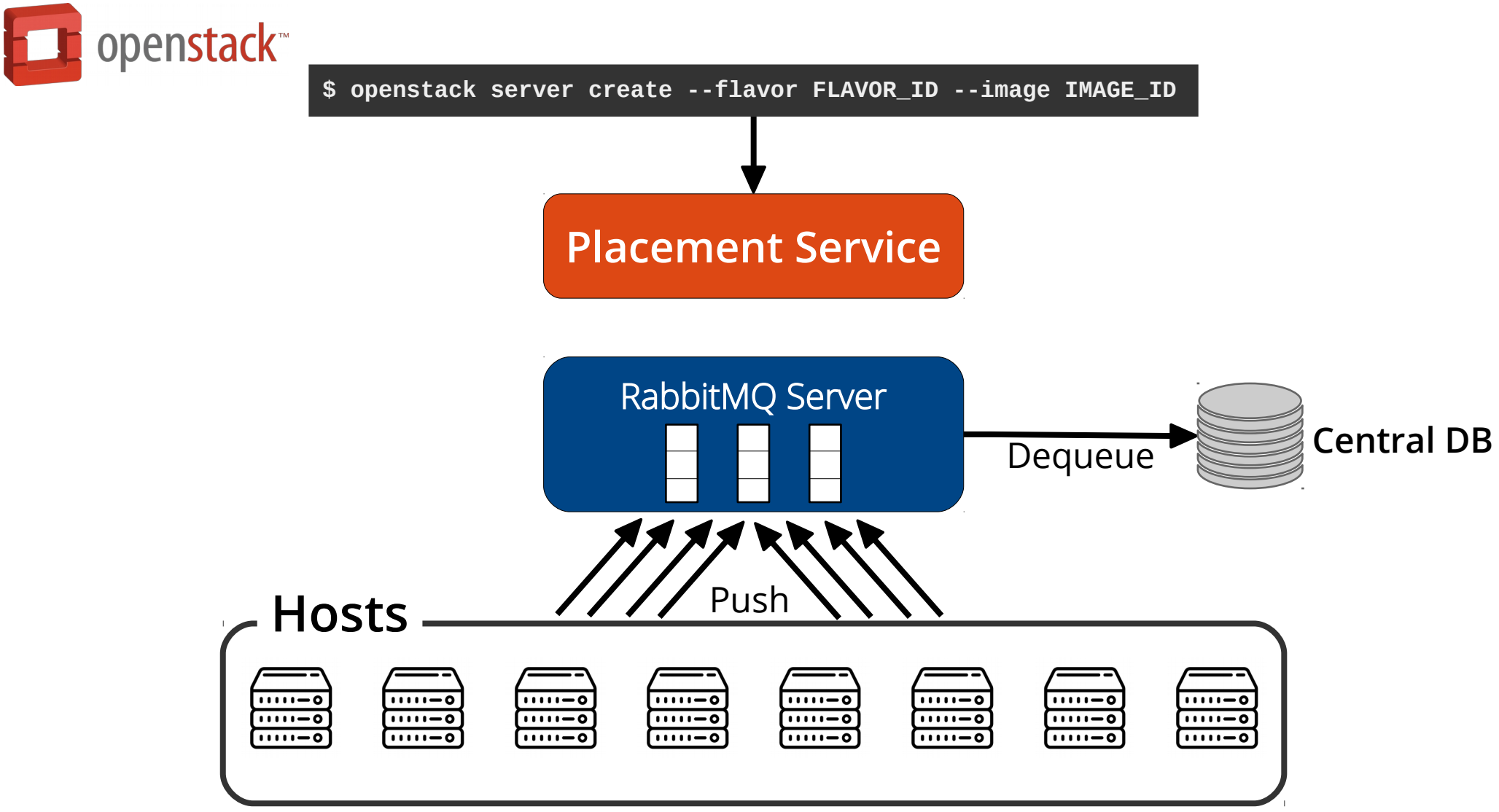

RabbitMQ Server
Push
Hosts

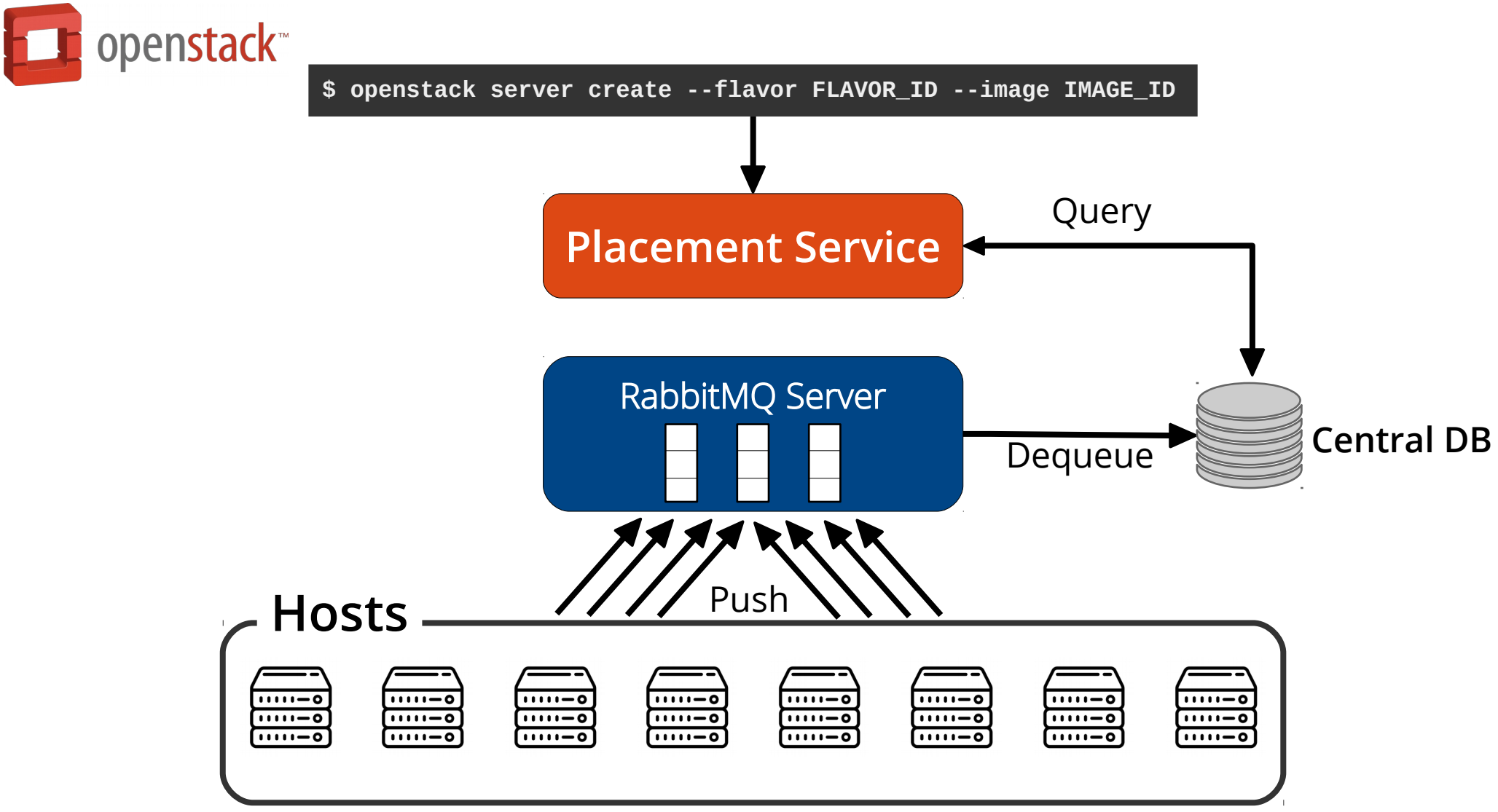

RabbitMQ Server
Dequeue
Central DB
Push
Hosts

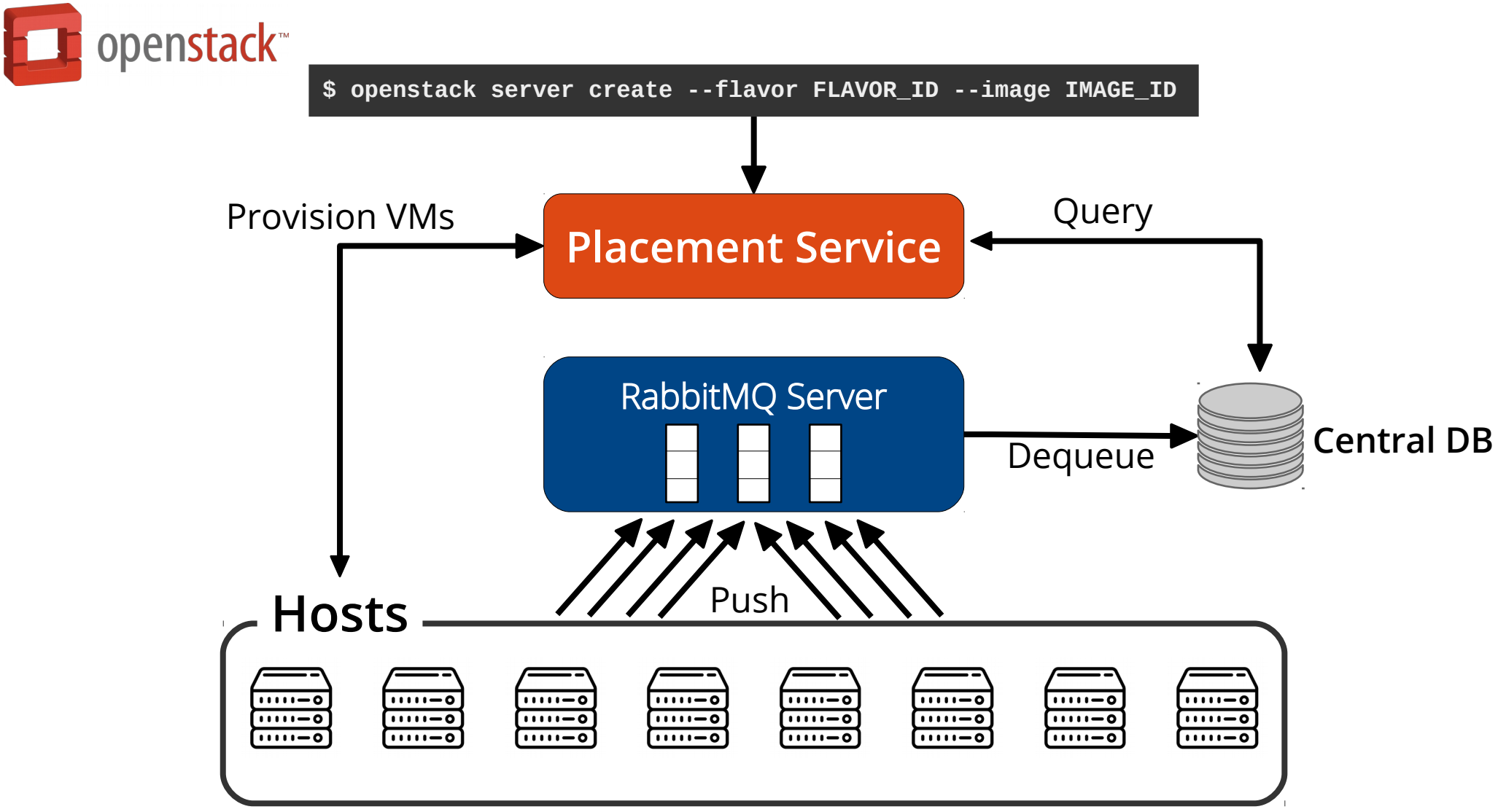
openstack™

```
$ openstack server create --flavor FLAVOR_ID --image IMAGE_ID
```

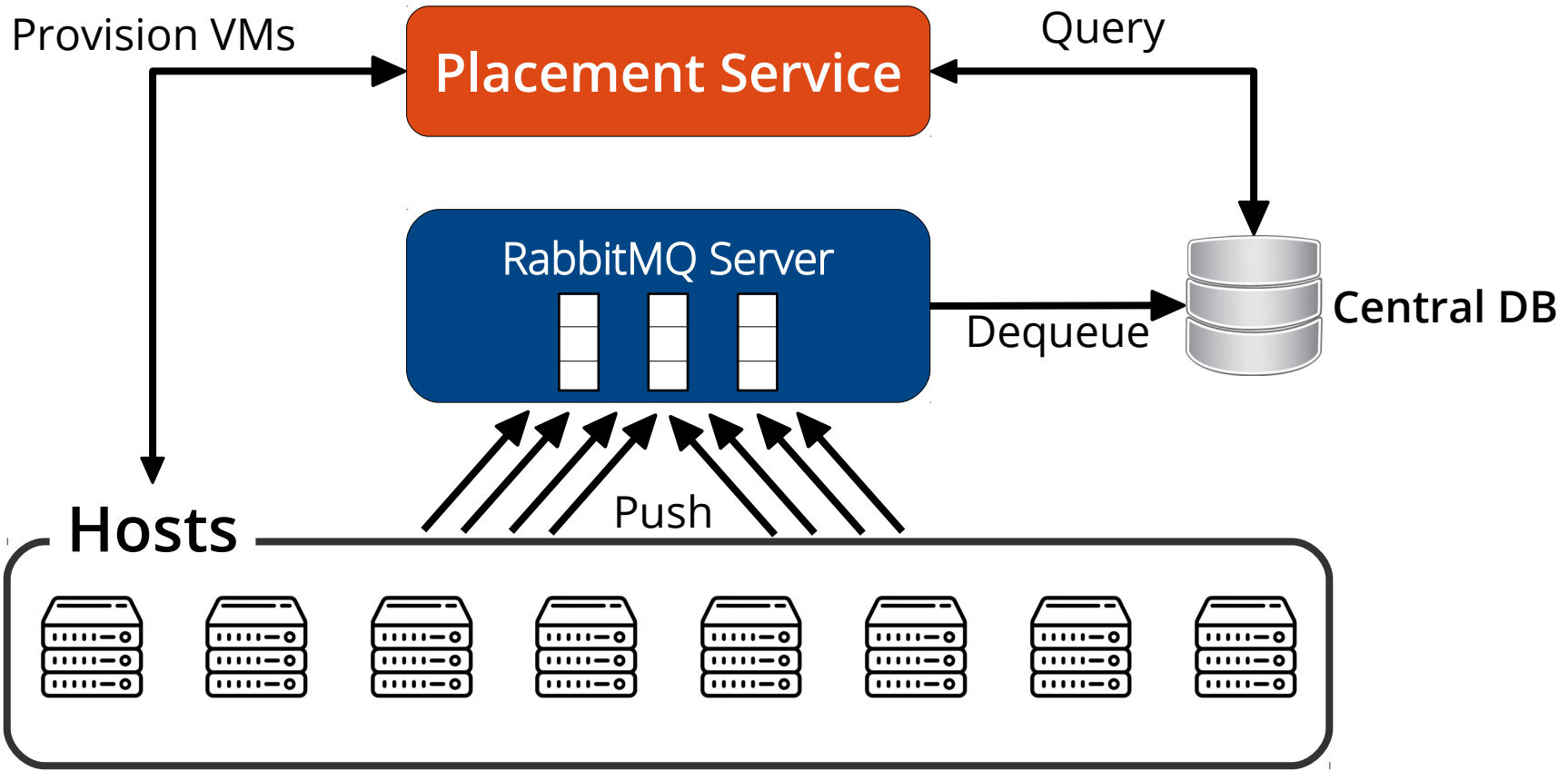



openstack™
$ openstack server create --flavor FLAVOR_ID --image IMAGE_ID
Placement Service
RabbitMQ Server
Dequeue
Central DB
Push
Hosts


openstack™
$ openstack server create --flavor FLAVOR_ID --image IMAGE_ID
Placement Service
Query
RabbitMQ Server
Dequeue
Central DB
Push
Hosts

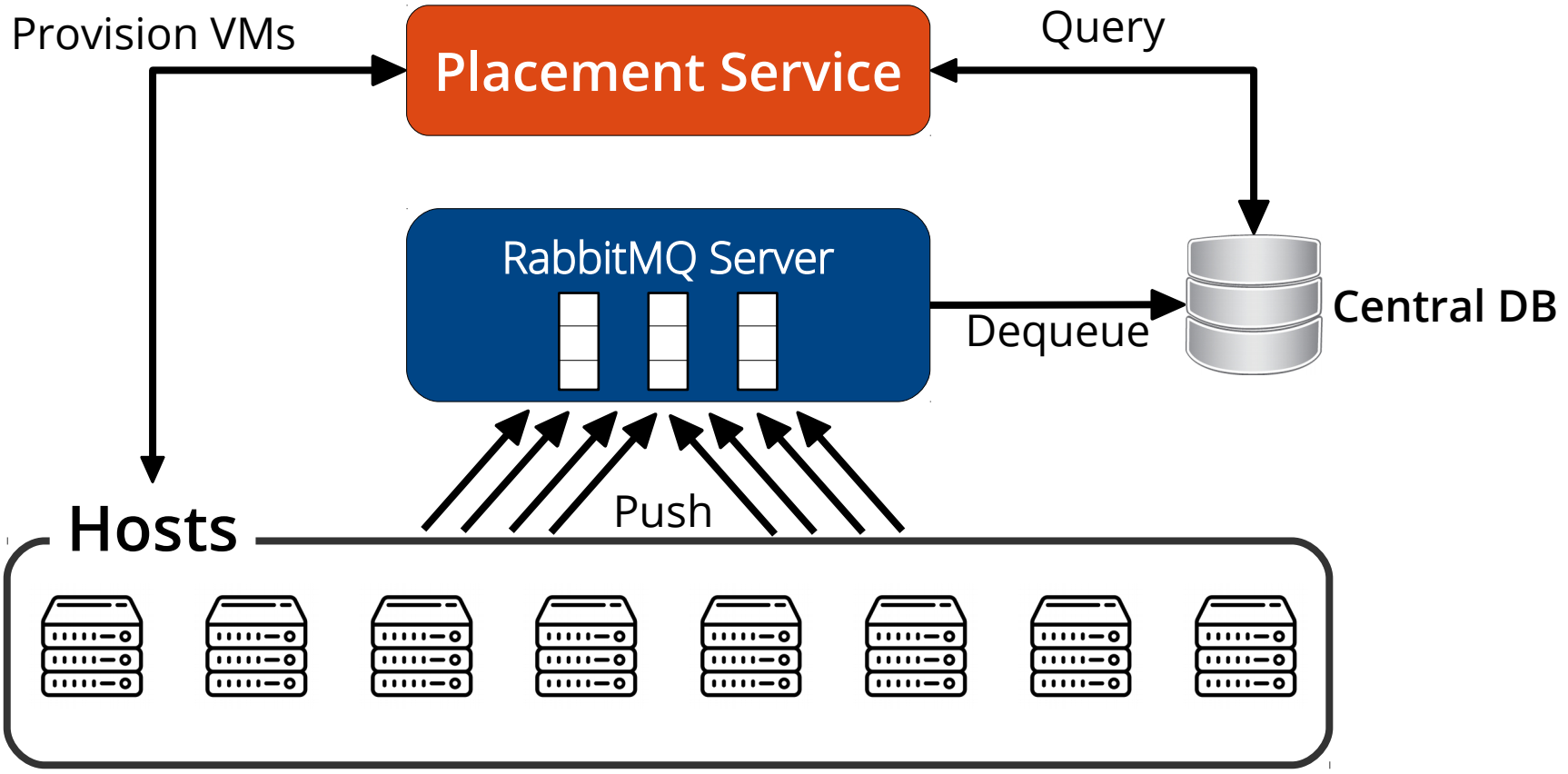

openstack™
$ openstack server create --flavor FLAVOR_ID --image IMAGE_ID
Provision VMs
Placement Service
Query
RabbitMQ Server
Dequeue
Central DB
Push
Hosts

Limitations of Current Approaches


Provision VMs
Placement Service
Query
RabbitMQ Server
Dequeue
Central DB
Push
Hosts

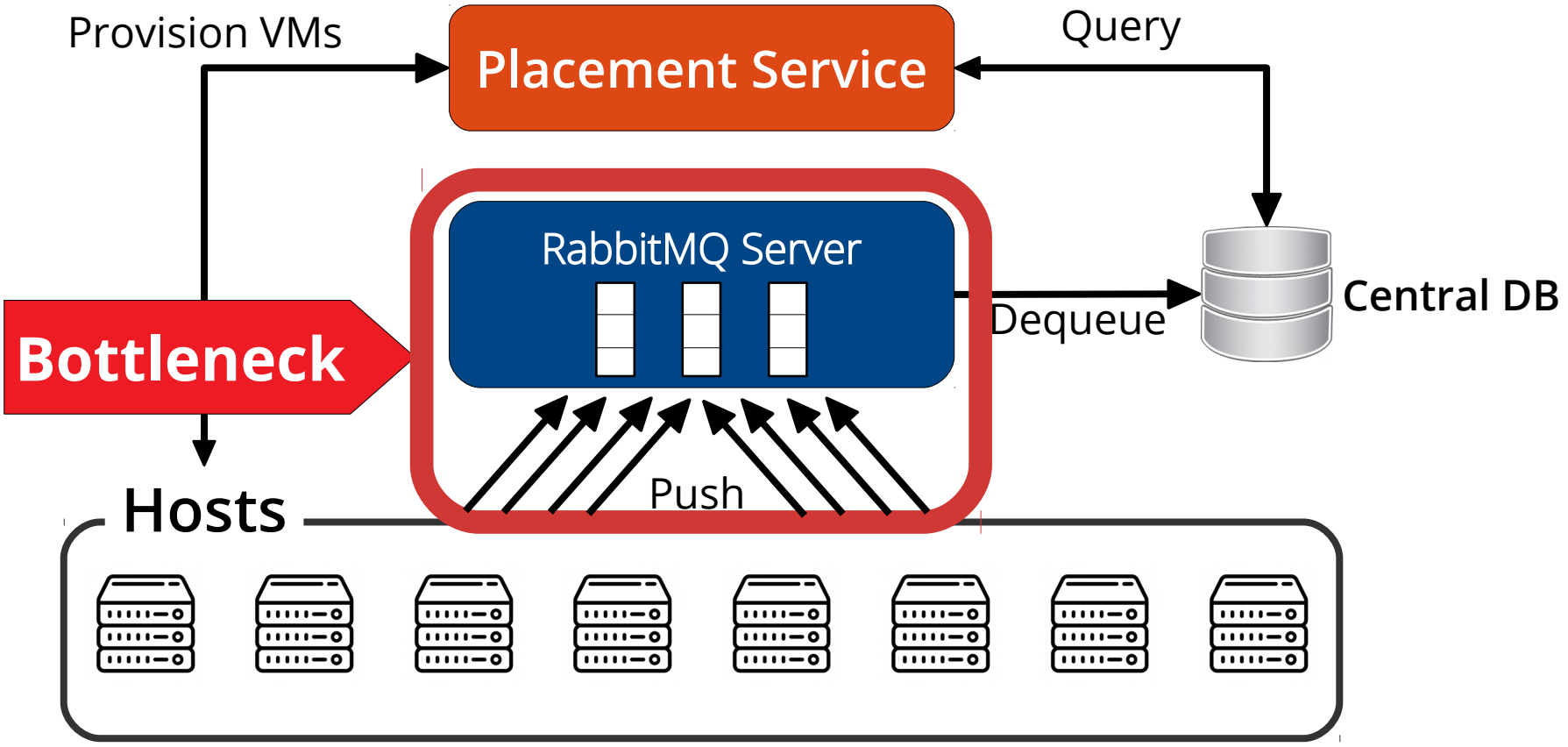
openstack™

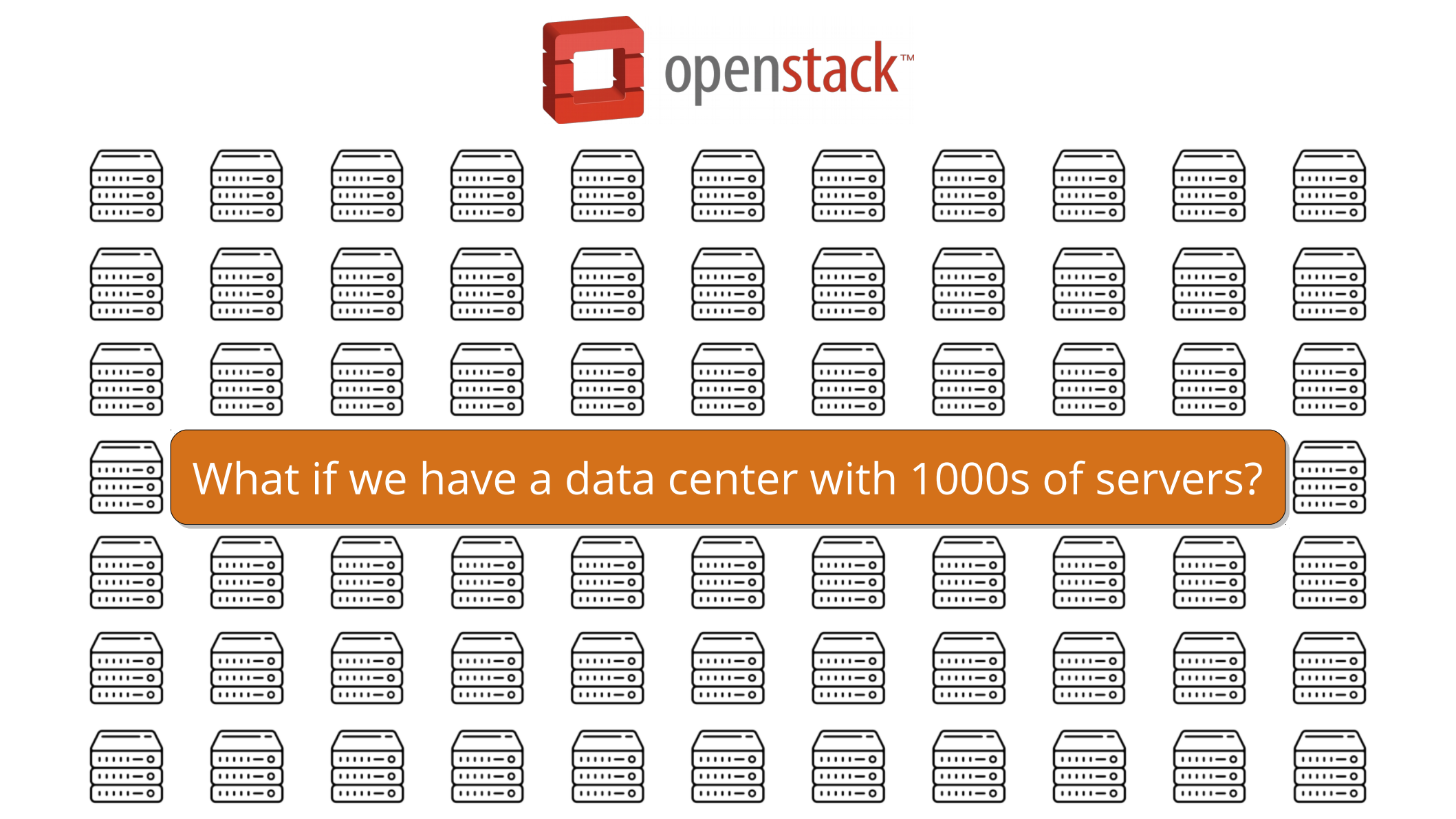
Hard to scale > 100s of nodes!

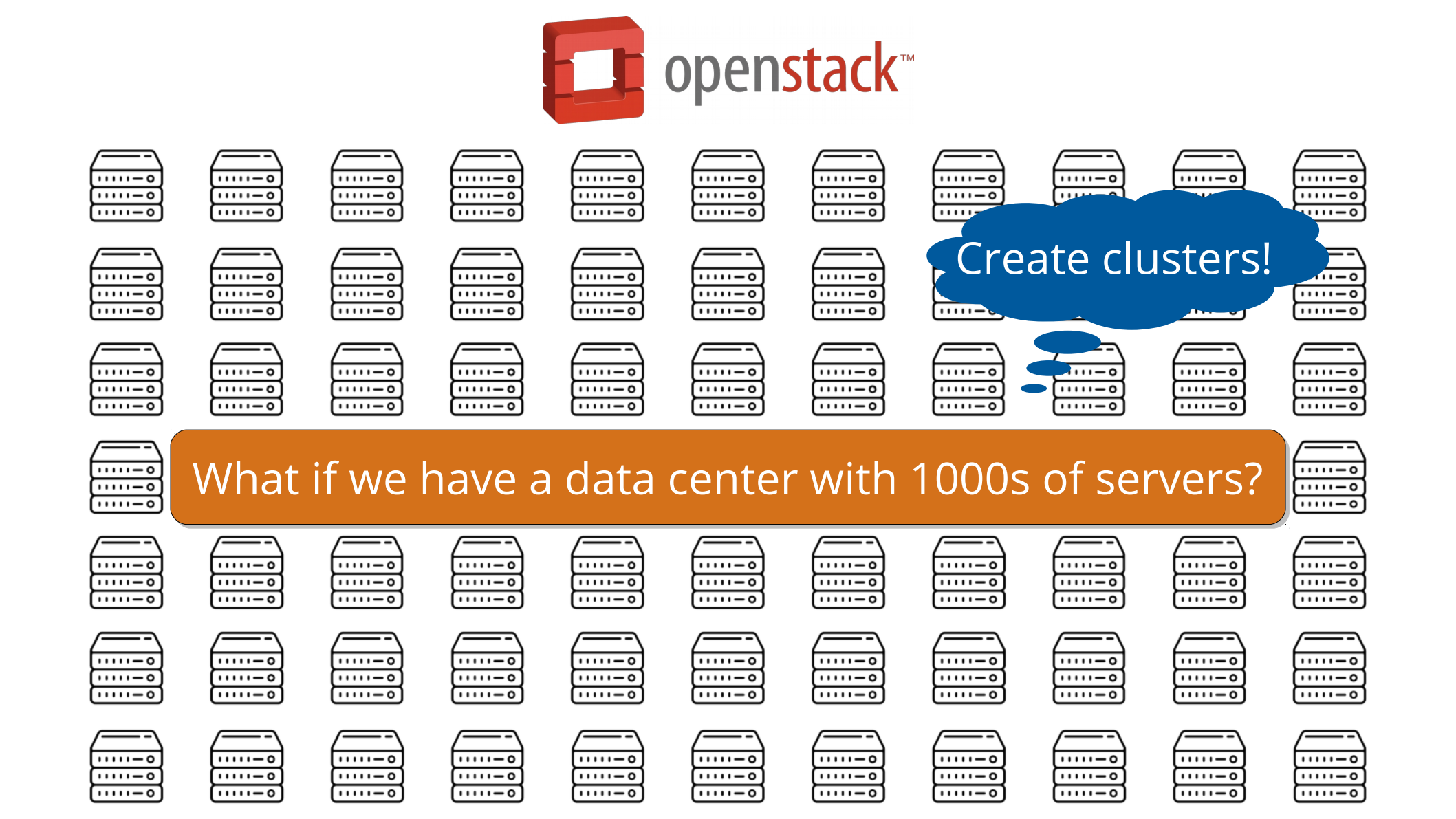


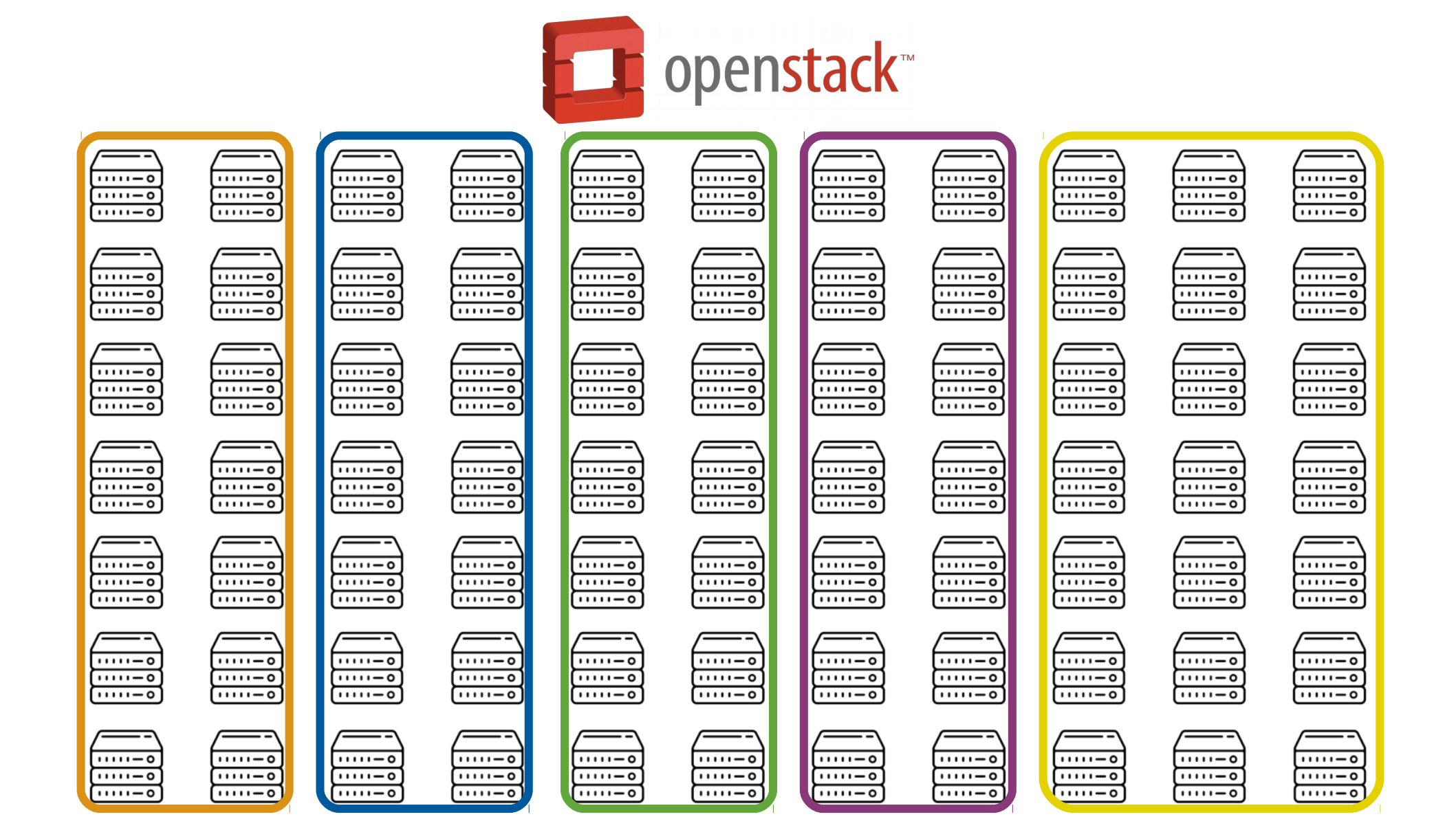
openstack™

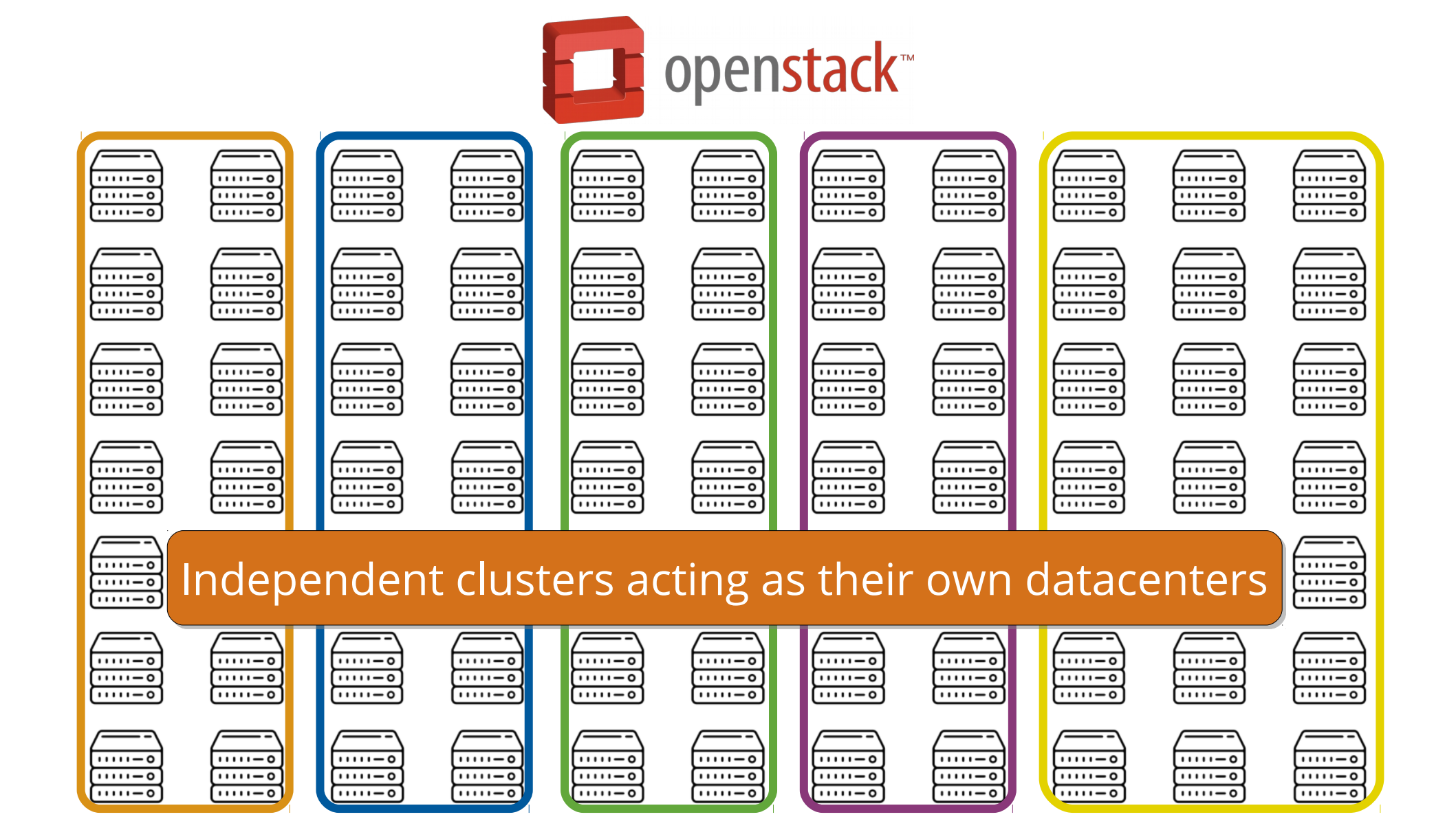
Hard to scale > 100s of nodes!

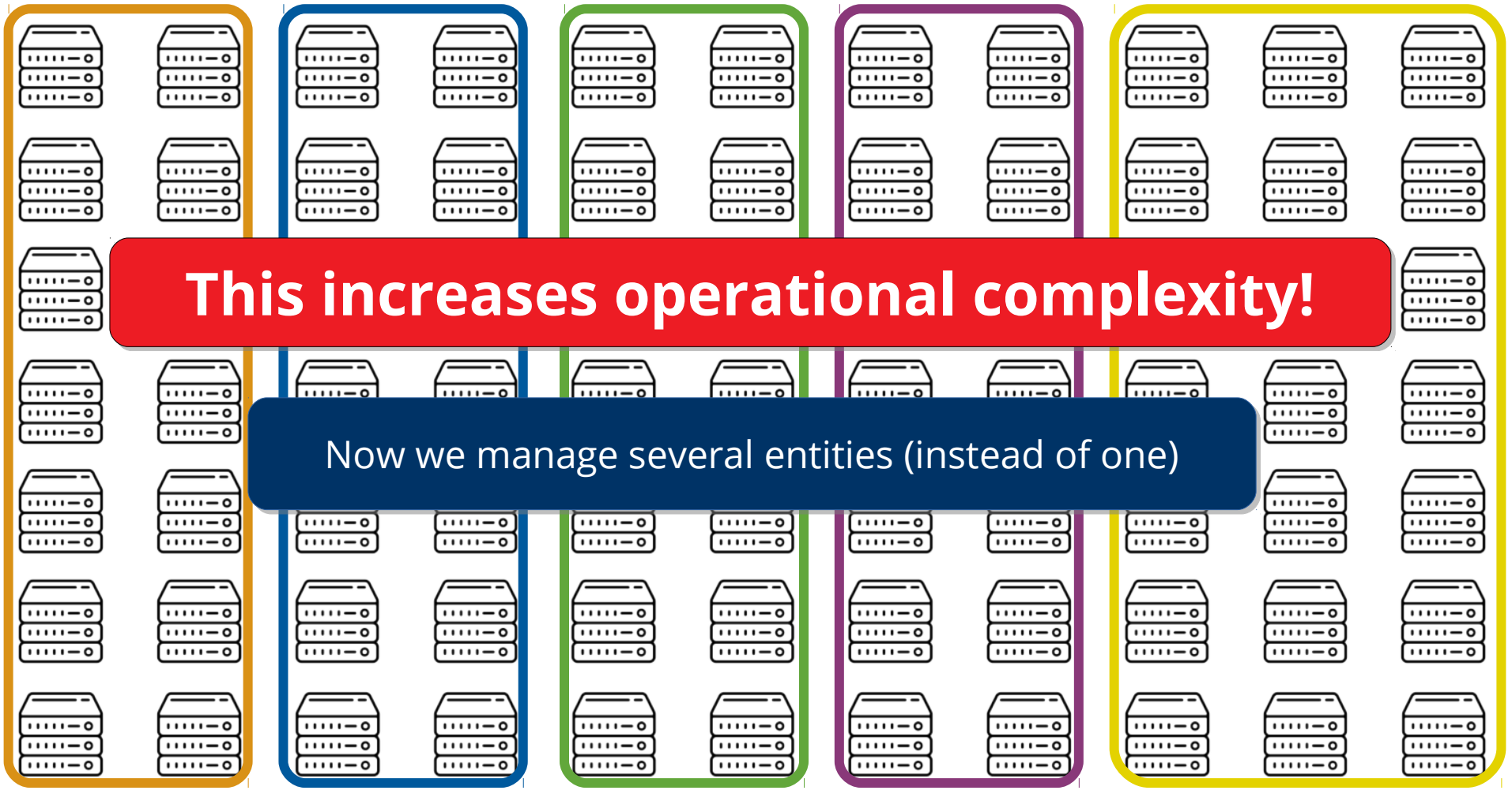



openstack™
What if we have a data center with 1000s of servers?

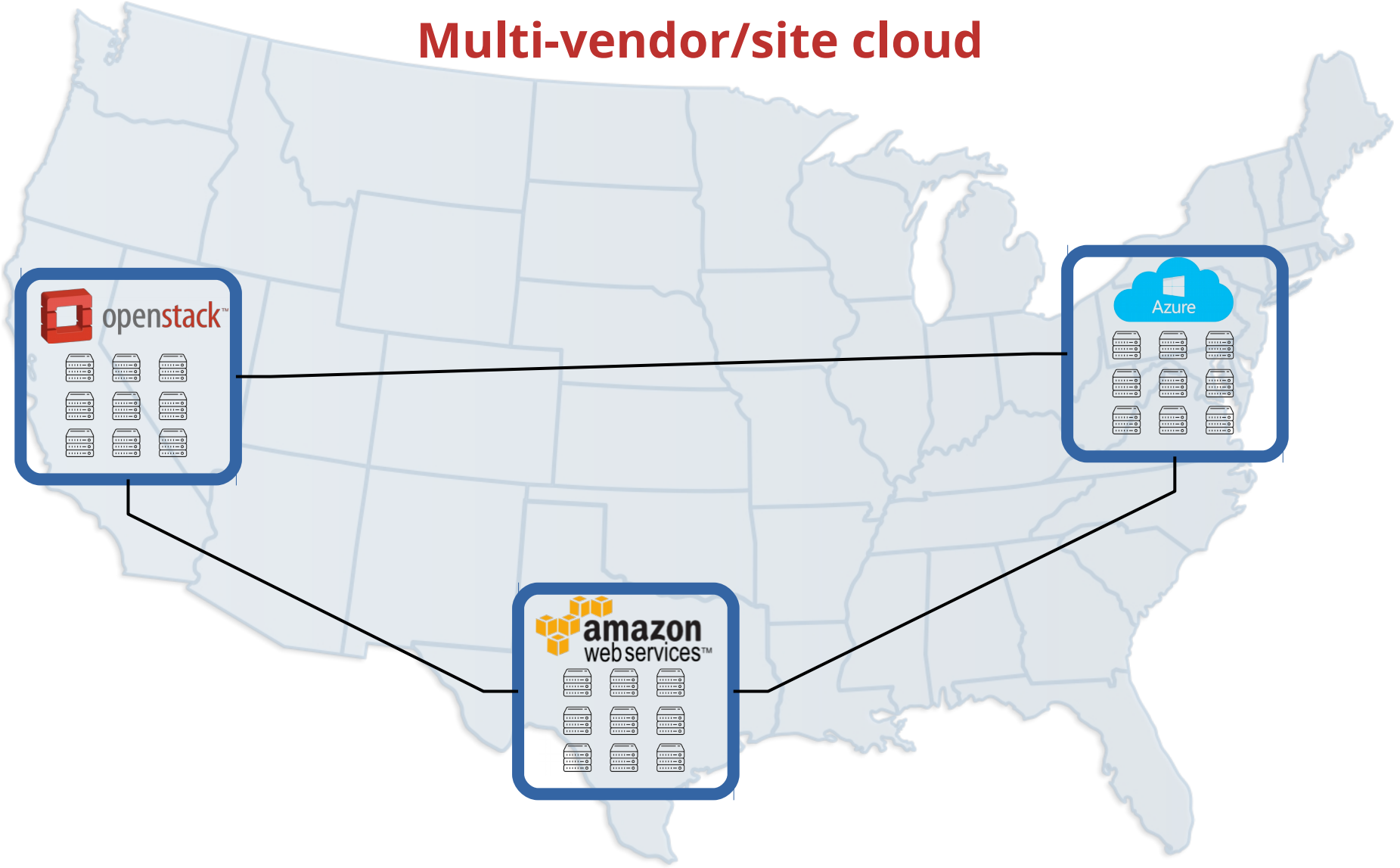

openstack™
Create clusters!
What if we have a data center with 1000s of servers?


openstack™

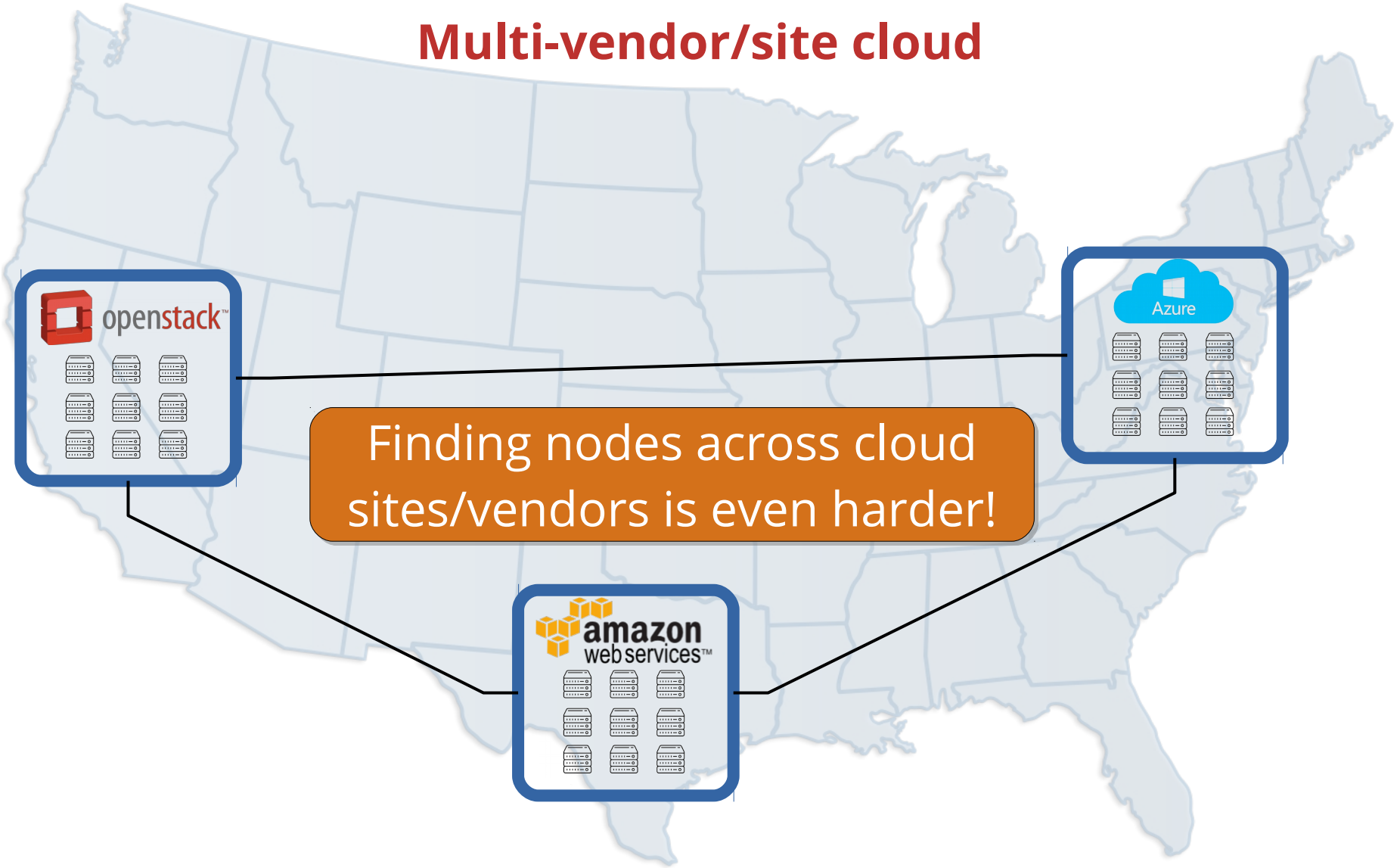

openstack™
Independent clusters acting as their own datacenters


openstack™
This increases operational complexity!
Now we manage several entities (instead of one)

Multi-vendor/site cloud



Multi-vendor/site cloud



Scalable and generic search service for distributed systems

Main Components

Main Components

Query Processing with Directed Pulling

Main Components

Query Processing with Directed Pulling

Gossip-based Node Coordination

Main Components

Query Processing with Directed Pulling

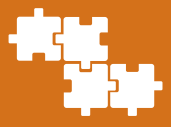
Gossip-based Node Coordination

Easy-to-integrate Query Interface

Main Components

Query Processing with Directed Pulling

Gossip-based Node Coordination



Easy-to-integrate Query Interface

Abstractions

Abstractions

Node Attributes

Abstractions

- **Node Attributes**
  - Static
  - Dynamic

Abstractions

- **Node Attributes**
  - Static
    - Never change
  - Dynamic

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change
      - Usage: *cpu, ram, disk, bandwidth, etc*

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change
      - Usage: *cpu, ram, disk, bandwidth, etc*

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change
      - Usage: *cpu, ram, disk, bandwidth, etc*

Query Structure

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change
      - Usage: *cpu, ram, disk, bandwidth, etc*

- **Query Structure**
  - Attribute List

Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change
      - Usage: *cpu, ram, disk, bandwidth, etc*

- **Query Structure**
  - Attribute List
    - *name (string)*
    - *upper bound (int)*
    - *lower bound (int)*

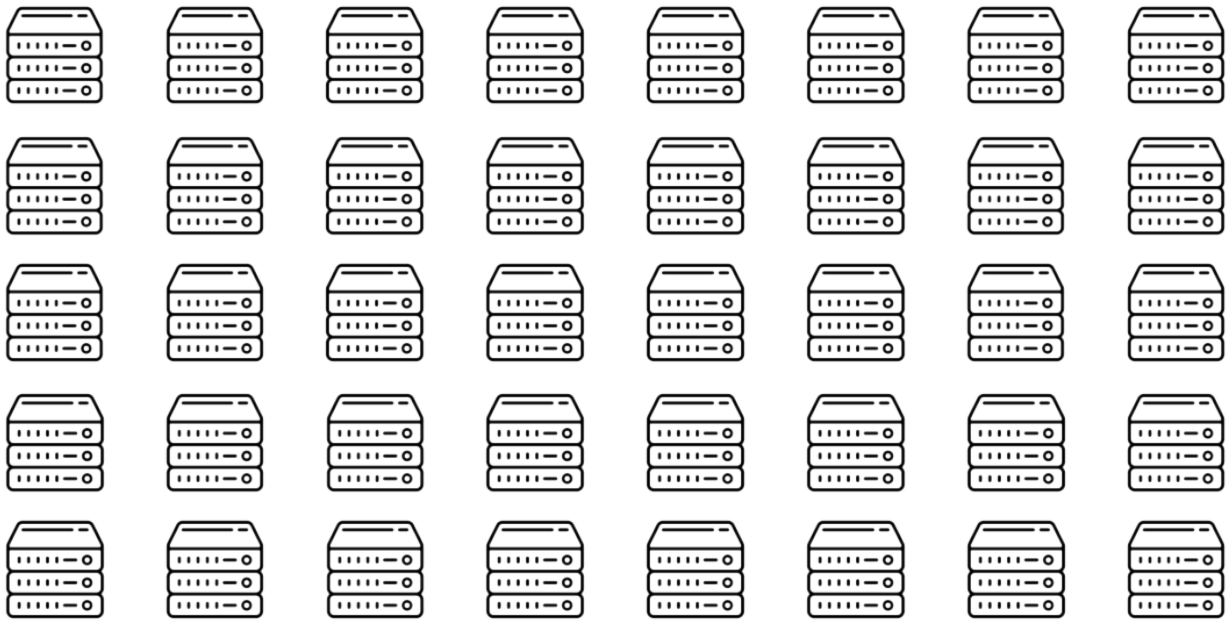
Abstractions

- **Node Attributes**
  - Static
    - Never change
      - *# cpu cores, arch, etc*
  - Dynamic
    - Frequently change
      - Usage: *cpu, ram, disk, bandwidth, etc*

- **Query Structure**
  - Attribute List
    - *name (string)*
    - *upper bound (int)*
    - *lower bound (int)*
  - *limit (int)*
  - *freshness (int)*

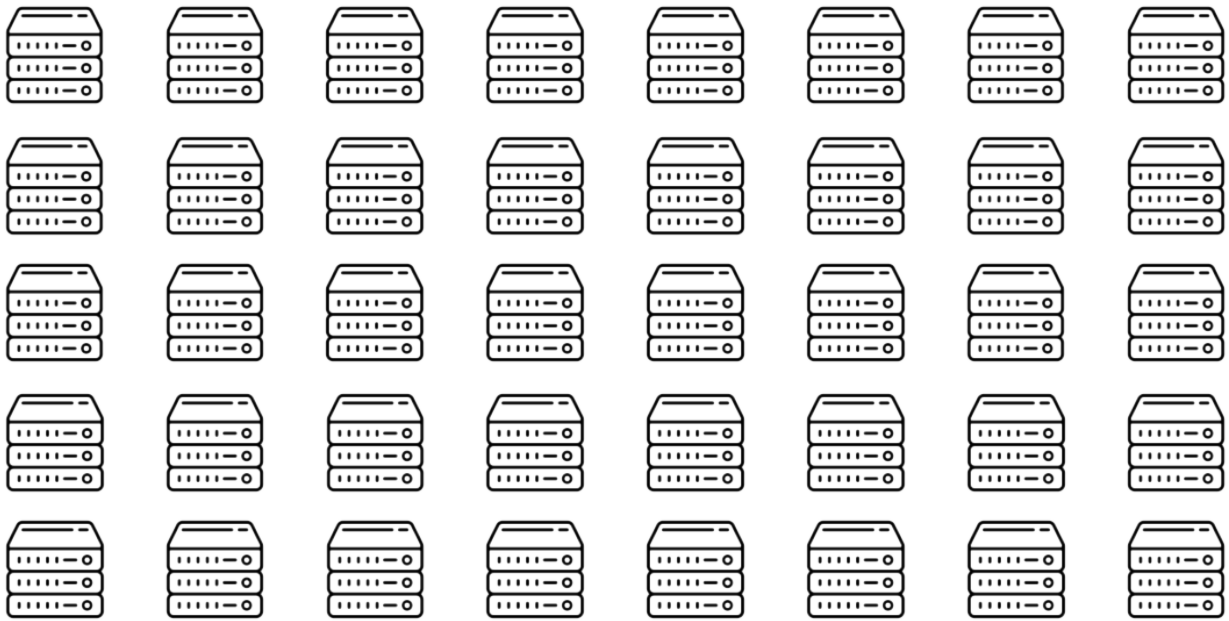
Query Processing with Directed Pulling

Attribute-based Grouping



Attribute-based Grouping

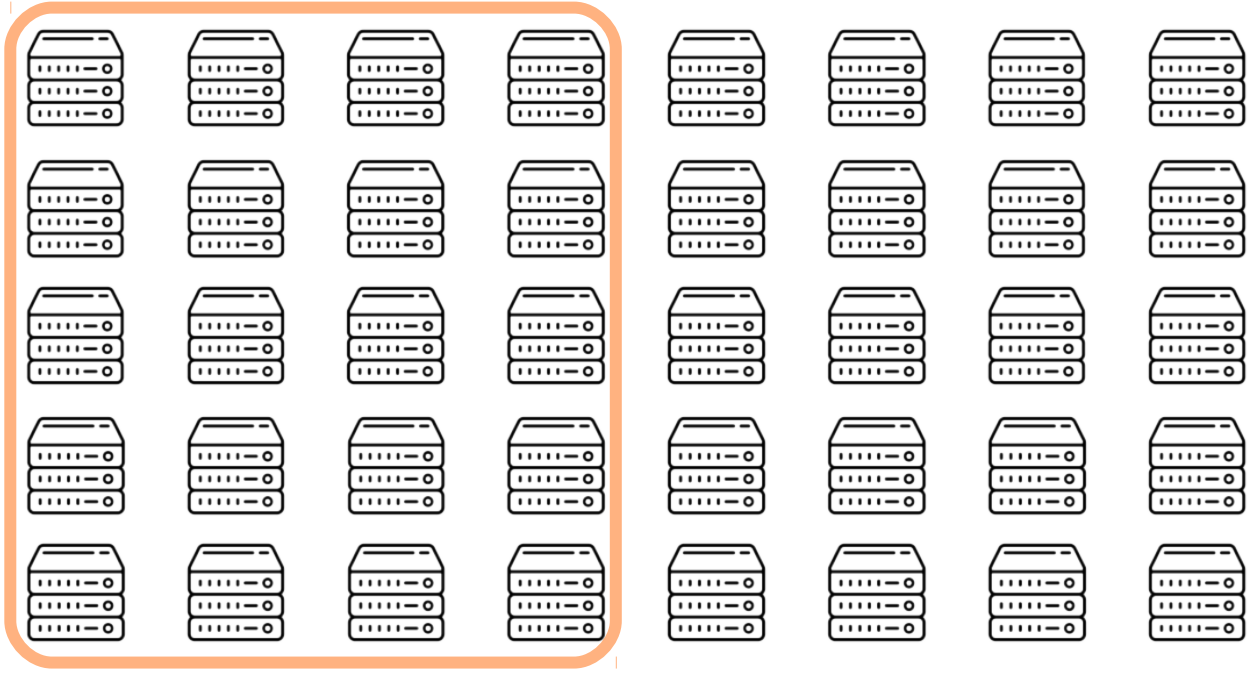
Group nodes according to their attribute values



Attribute-based Grouping

cpu_usage {50-100}%

Group nodes according to their attribute values

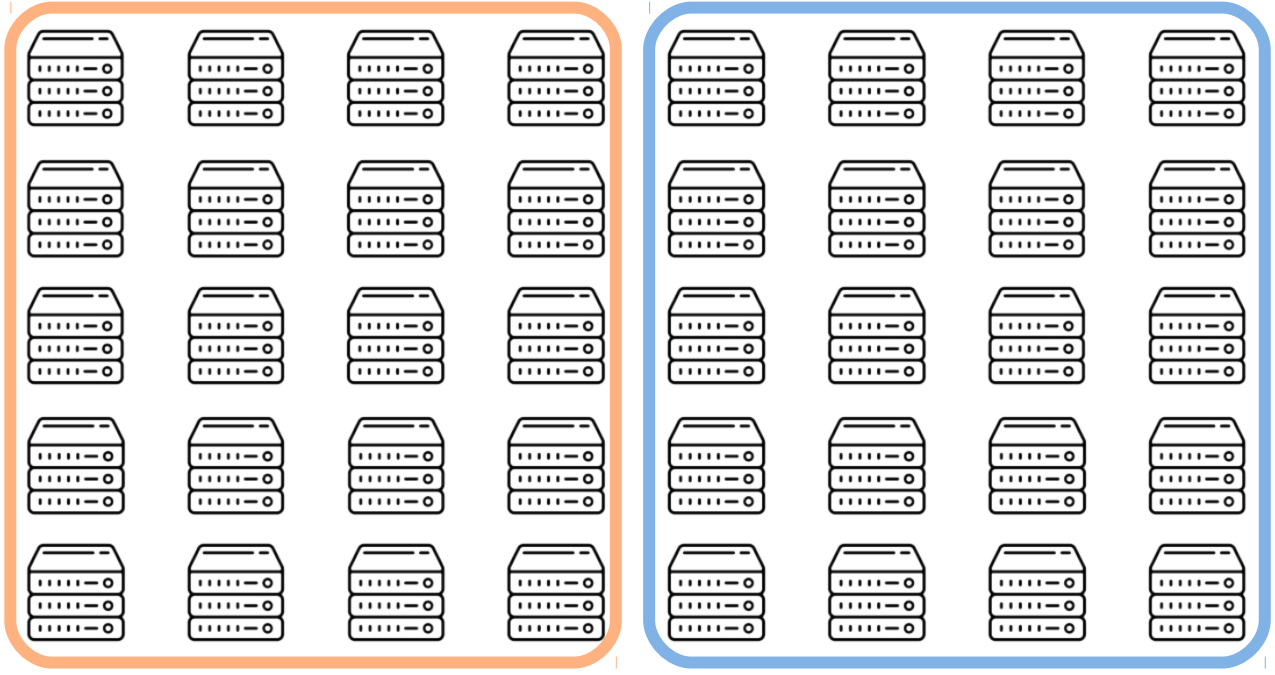


Attribute-based Grouping

cpu_usage {50-100}%

cpu_usage {0-50}%

Group nodes according to their attribute values



Attribute-based Grouping

cpu_usage {50-100}%

cpu_usage {0-50}%

avail_RAM {4-8}GB

Group nodes according to their attribute values

Attribute-based Grouping

cpu_usage {50-100}%

cpu_usage {0-50}%

avail_RAM {4-8}GB

cpu_cores {8-12}

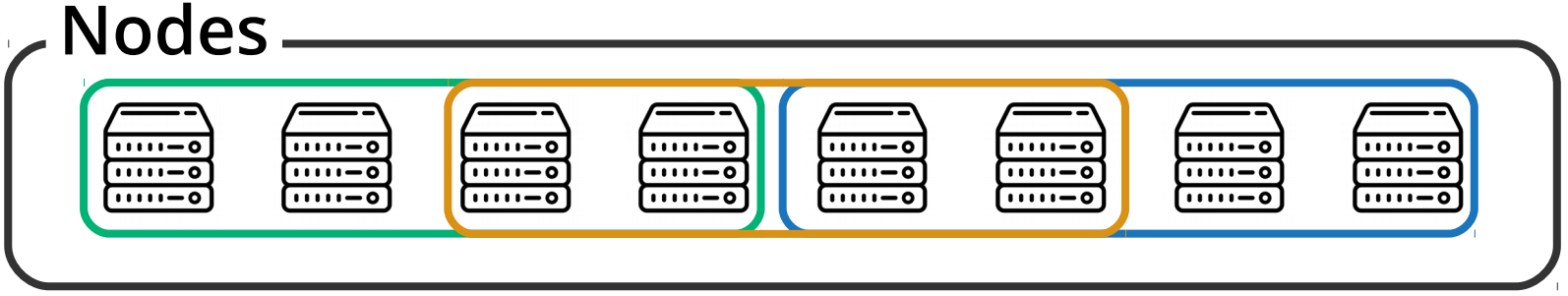
Group nodes according to their attribute values

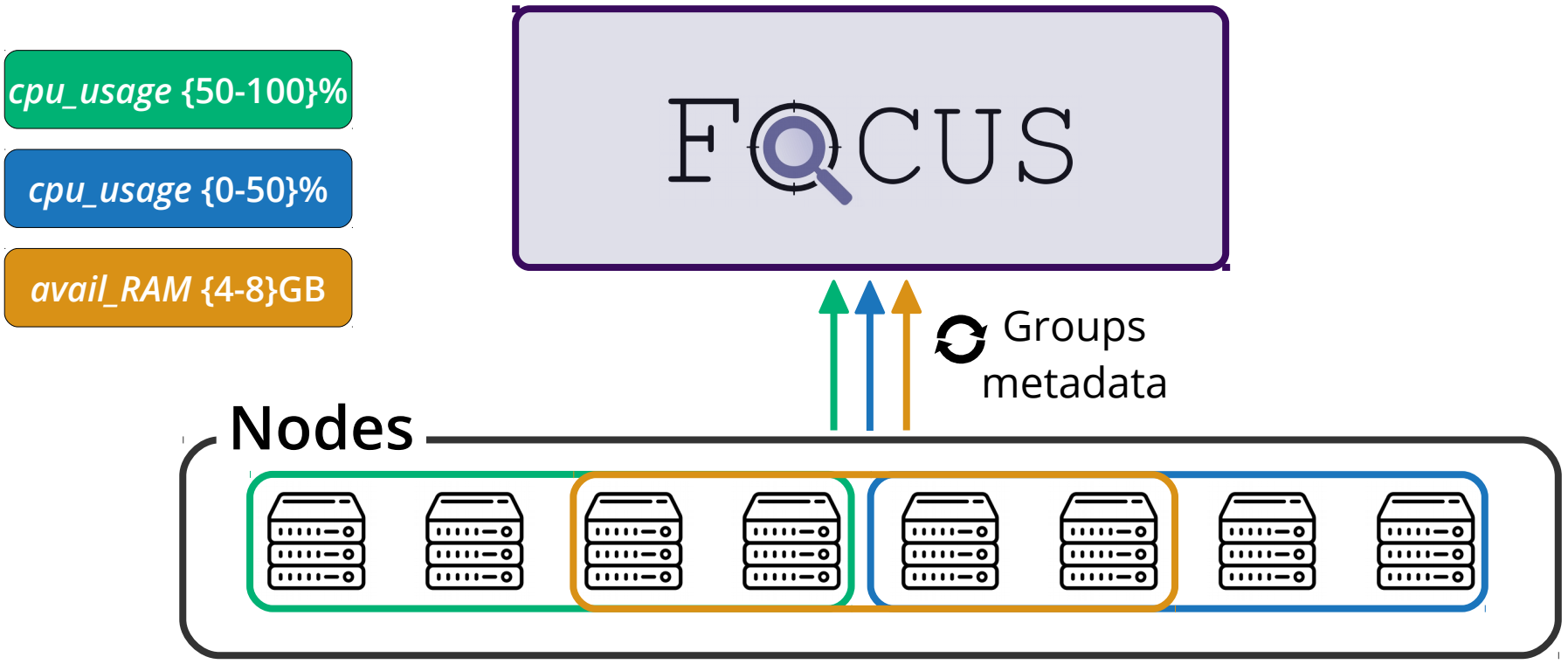
cpu_usage {50-100}%

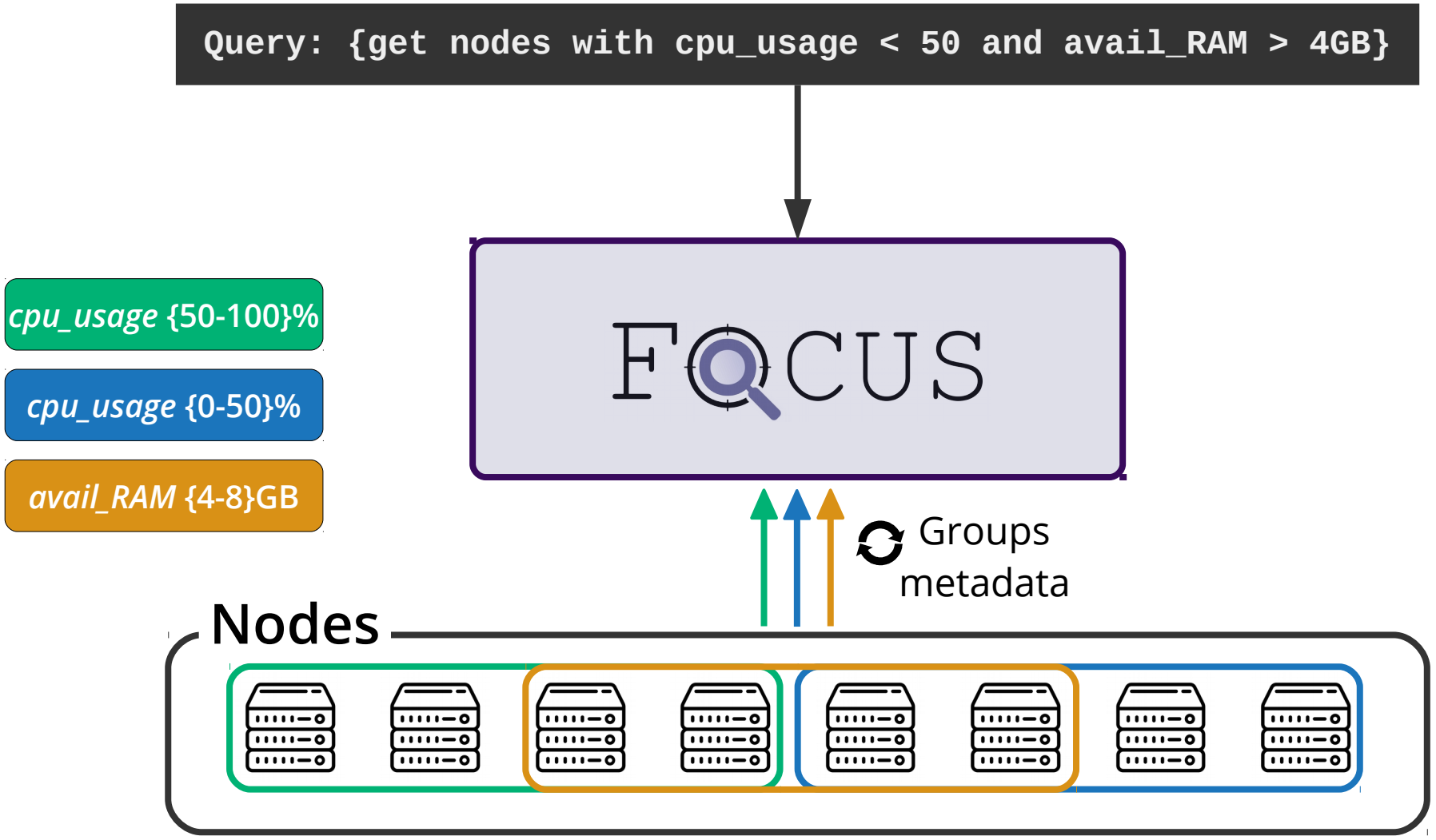
cpu_usage {0-50}%

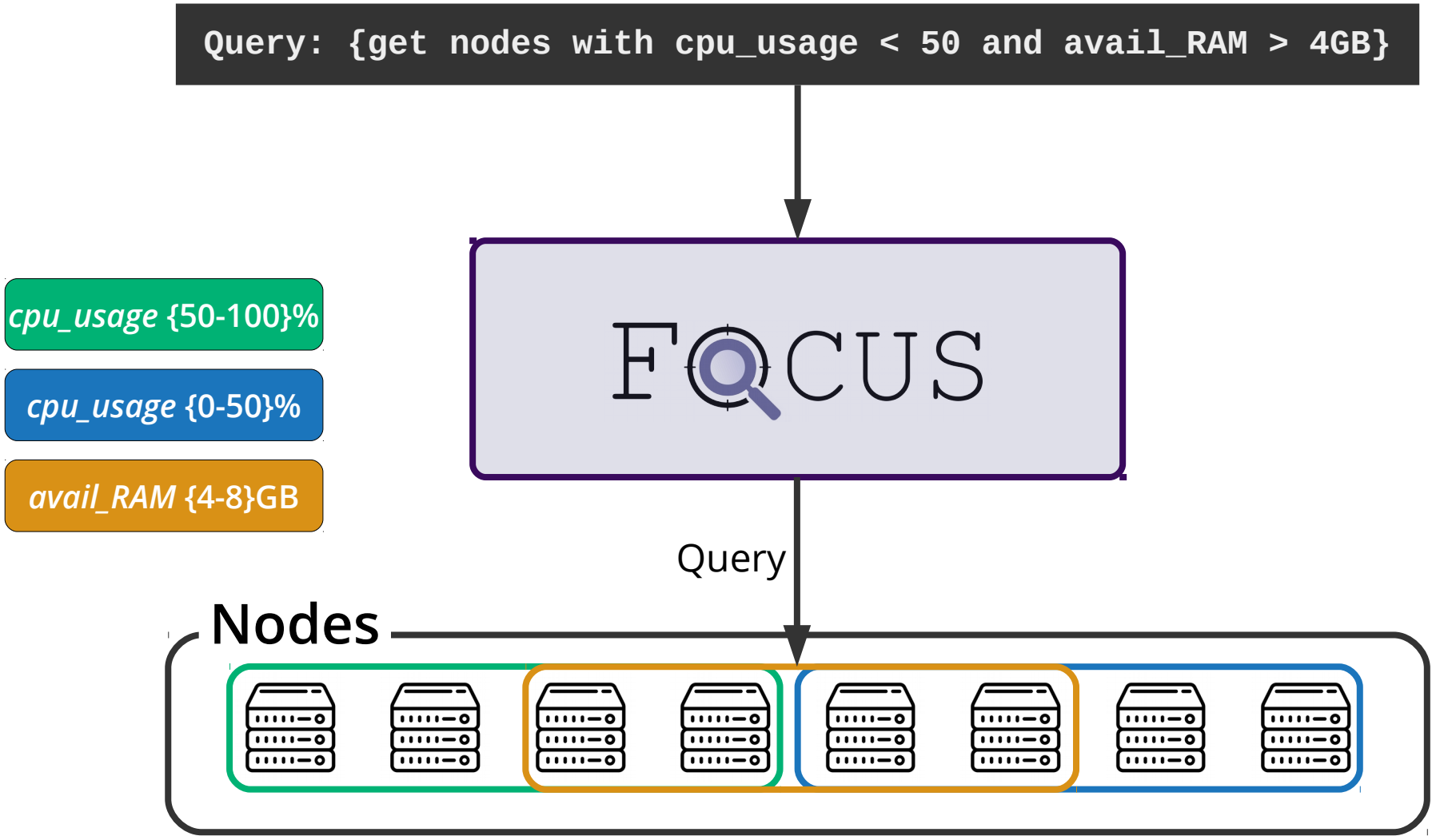
avail_RAM {4-8}GB

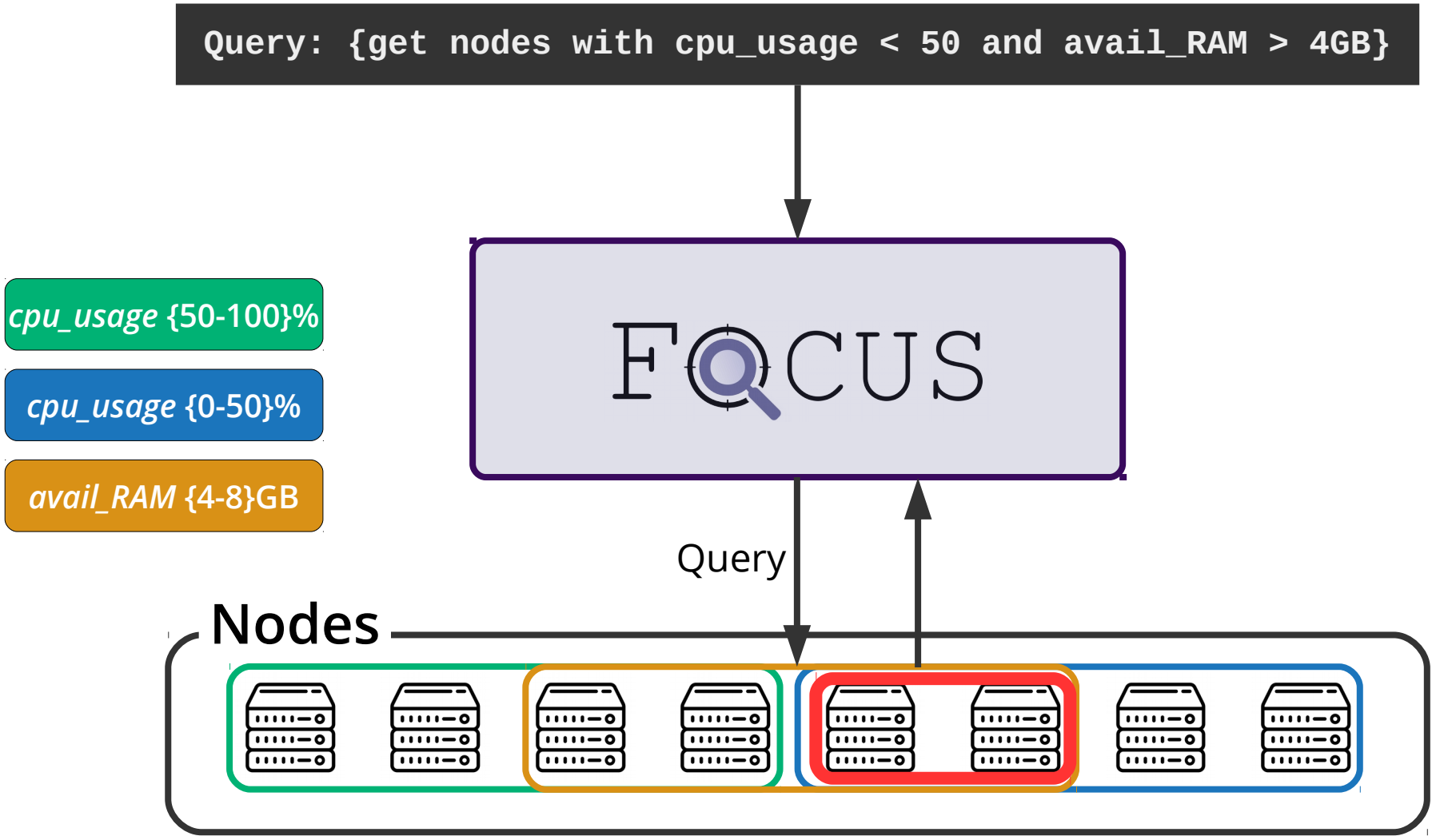
FOCUS

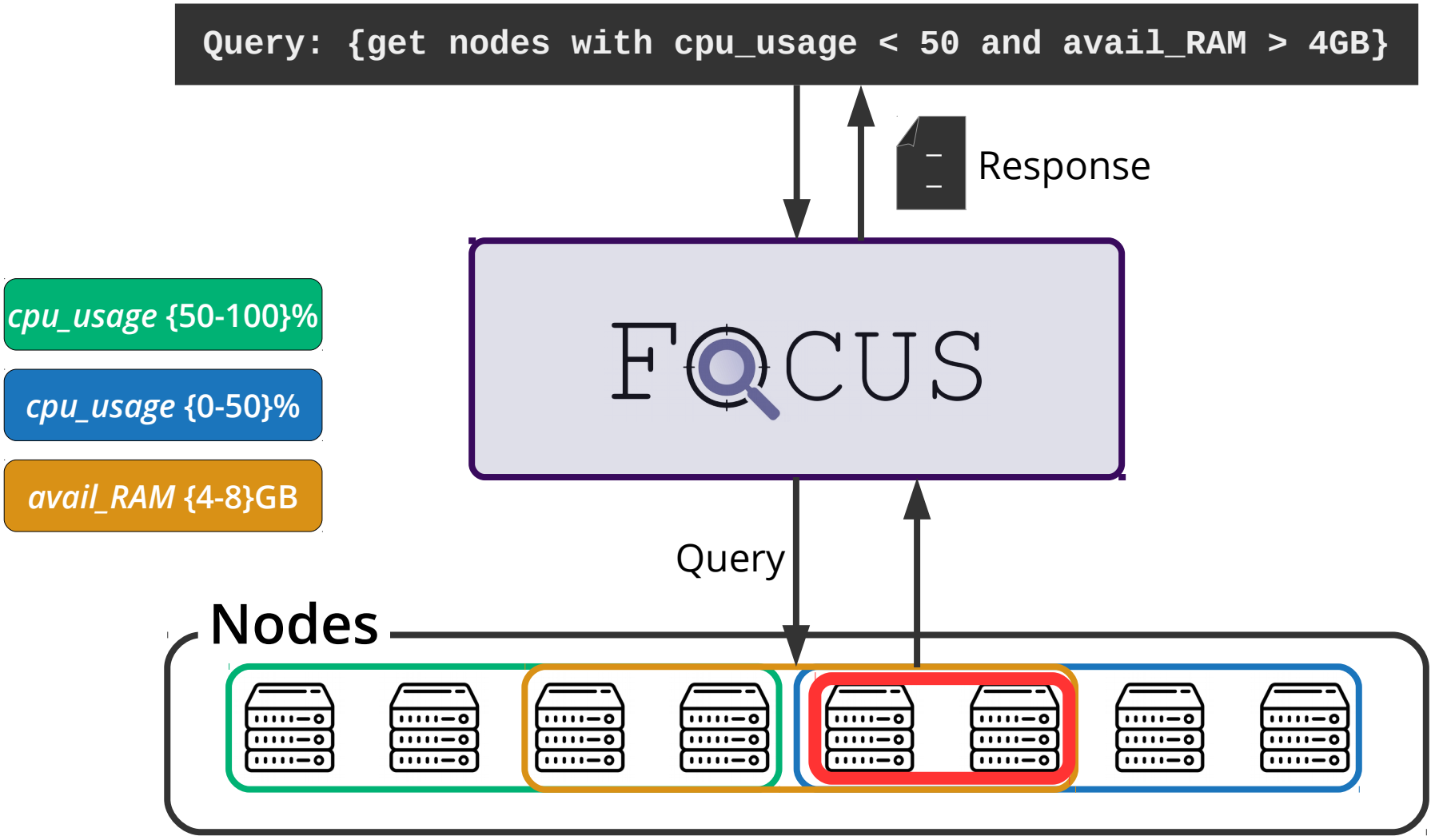



cpu_usage {50-100}%
cpu_usage {0-50}%
avail_RAM {4-8}GB
FOCUS
Groups
metadata
Nodes


Query: {get nodes with cpu_usage < 50 and avail_RAM > 4GB}
cpu_usage {50-100}%
cpu_usage {0-50}%
avail_RAM {4-8}GB
FOCUS
Groups
metadata
Nodes


Query: {get nodes with cpu_usage < 50 and avail_RAM > 4GB}
cpu_usage {50-100}%
cpu_usage {0-50}%
avail_RAM {4-8}GB
FOCUS
Query
Nodes

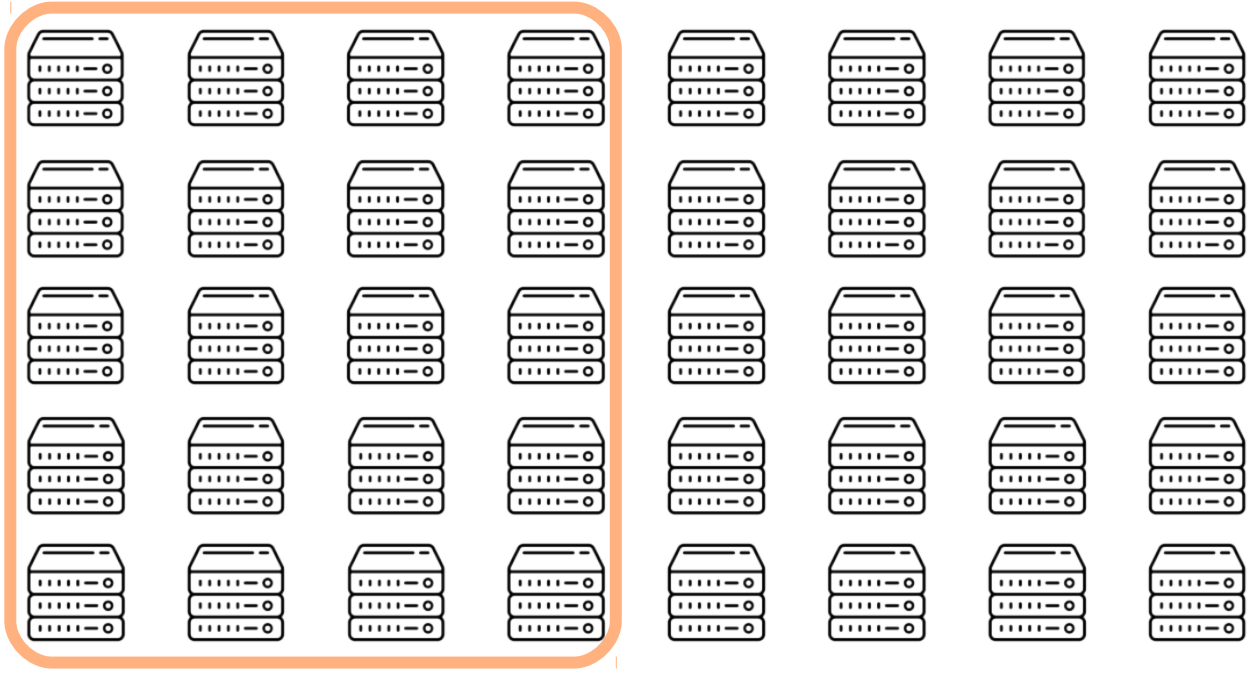

Query: {get nodes with cpu_usage < 50 and avail_RAM > 4GB}
cpu_usage {50-100}%
cpu_usage {0-50}%
avail_RAM {4-8}GB
FOCUS
Query
Nodes


Query: {get nodes with cpu_usage < 50 and avail_RAM > 4GB}
Response
FOCUS
cpu_usage {50-100}%
cpu_usage {0-50}%
avail_RAM {4-8}GB
Query
Nodes

Gossip-based Node Coordination

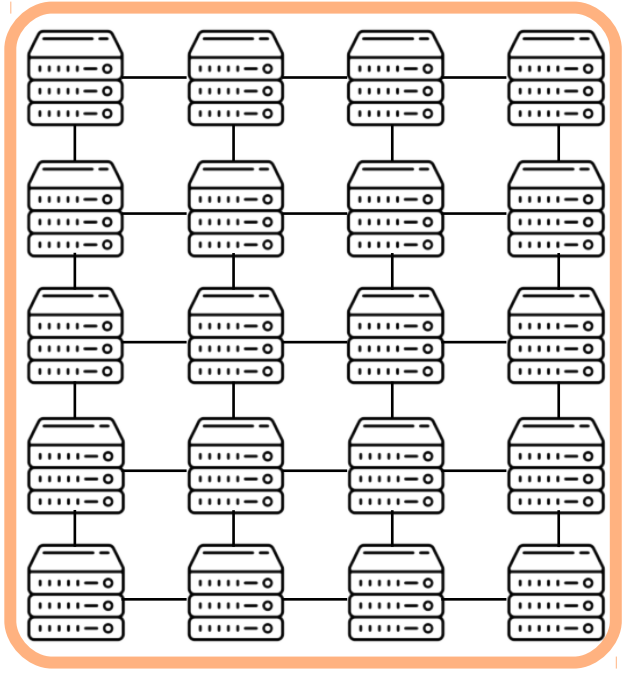
Gossip-based Coordination

cpu_usage {50-100}%



Gossip-based Coordination

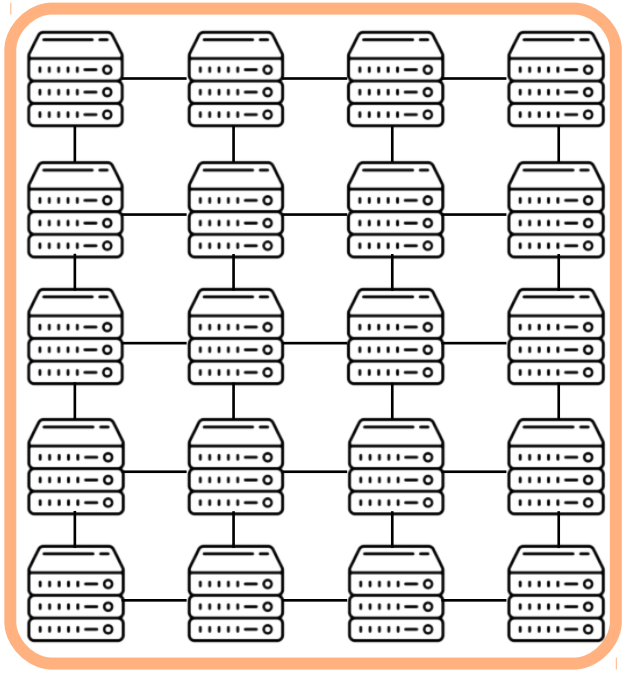
cpu_usage {50-100}%



Nodes in a group are connected through a **p2p *gossip*** channel

Gossip-based Coordination

cpu_usage {50-100}%

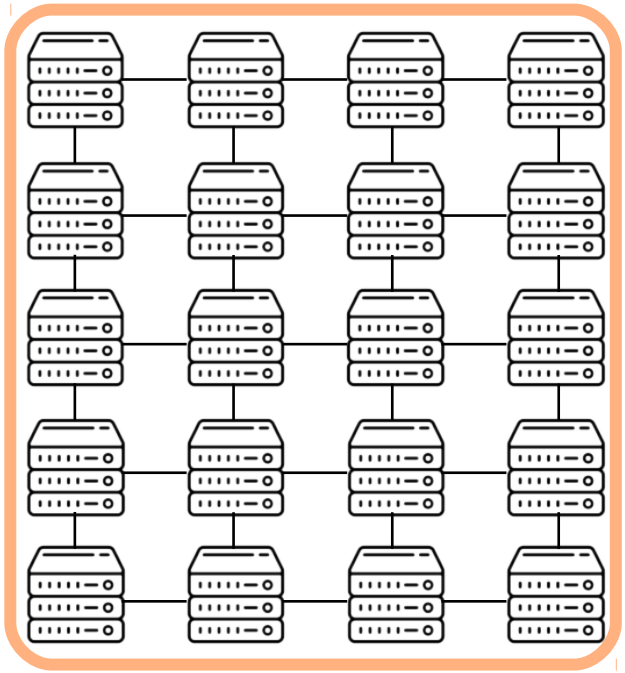


Nodes in a group are connected through a **p2p *gossip*** channel

Nodes exchange membership information

Gossip-based Coordination

cpu_usage {50-100}%



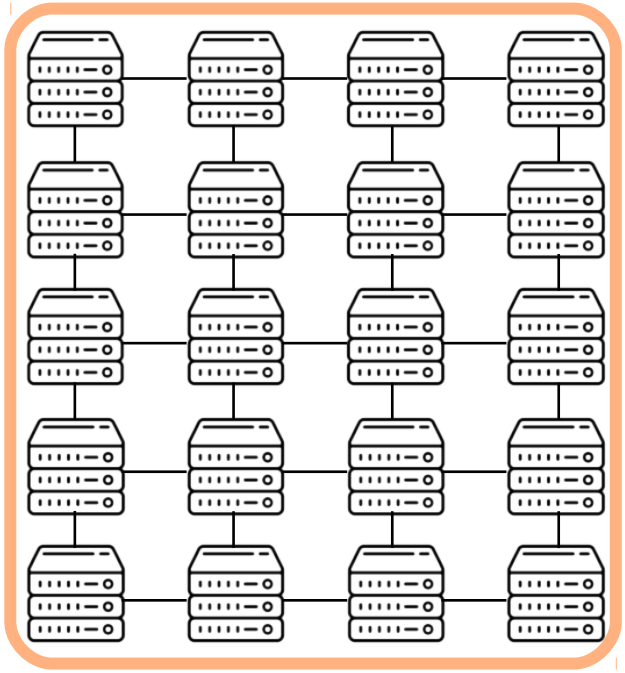
Nodes in a group are connected through a **p2p *gossip*** channel

Nodes exchange membership information

One node pushes group info to the FOCUS server

Gossip-based Coordination

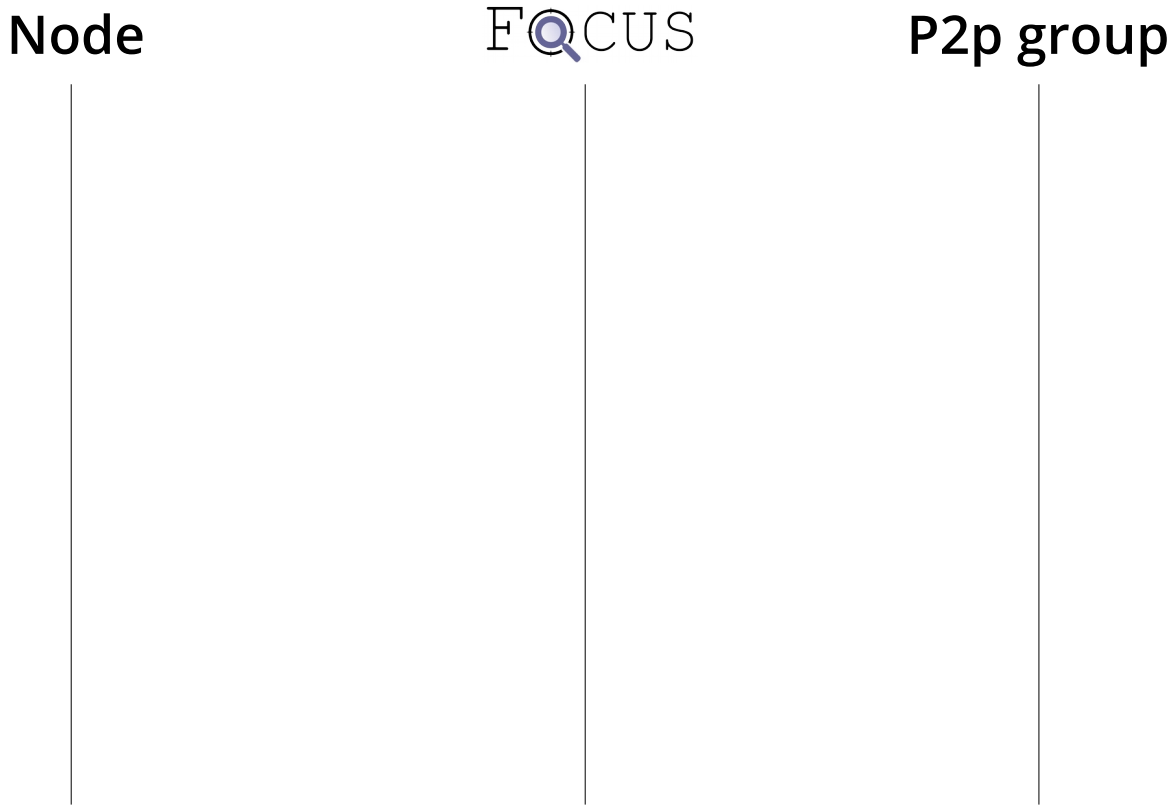
cpu_usage {50-100}%



- Nodes in a group are connected through a **p2p *gossip*** channel
- Nodes exchange membership information
- One node pushes group info to the FOCUS server
- Queries are propagated via *gossip* channel

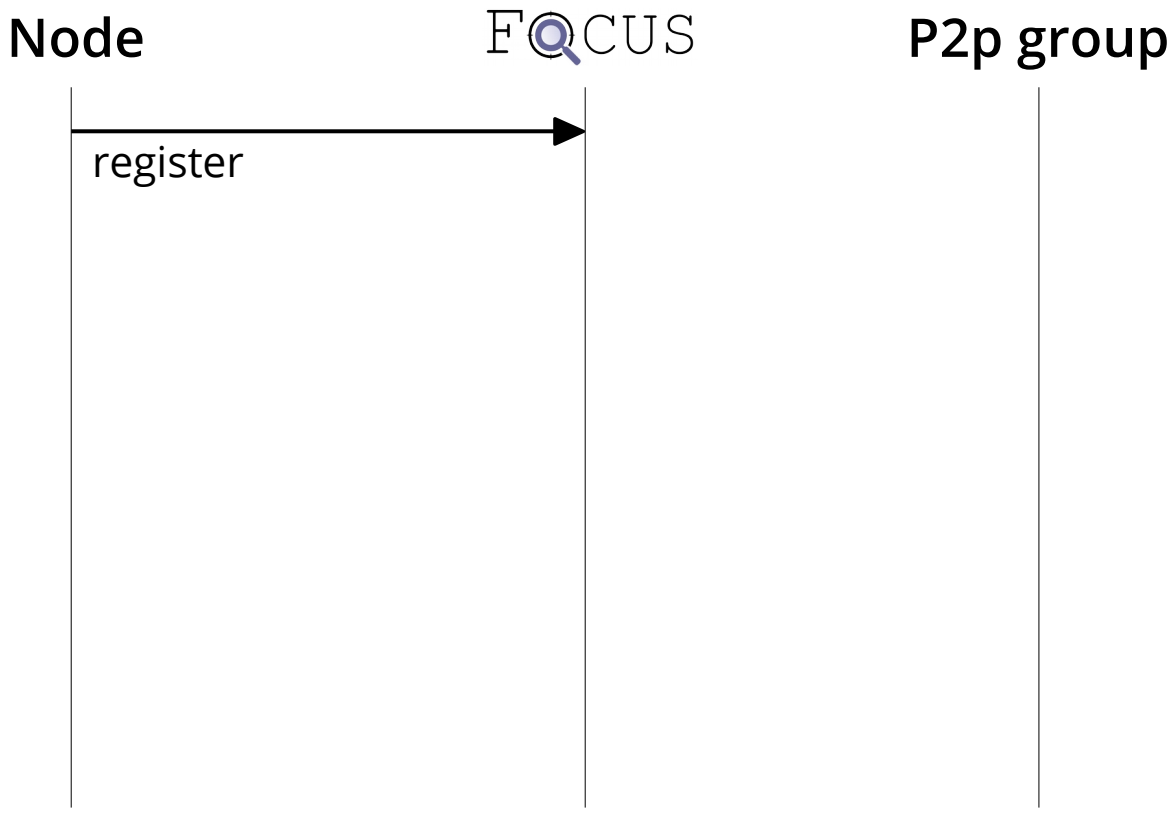
Dynamic Groups Management

Operations flow in FOCUS



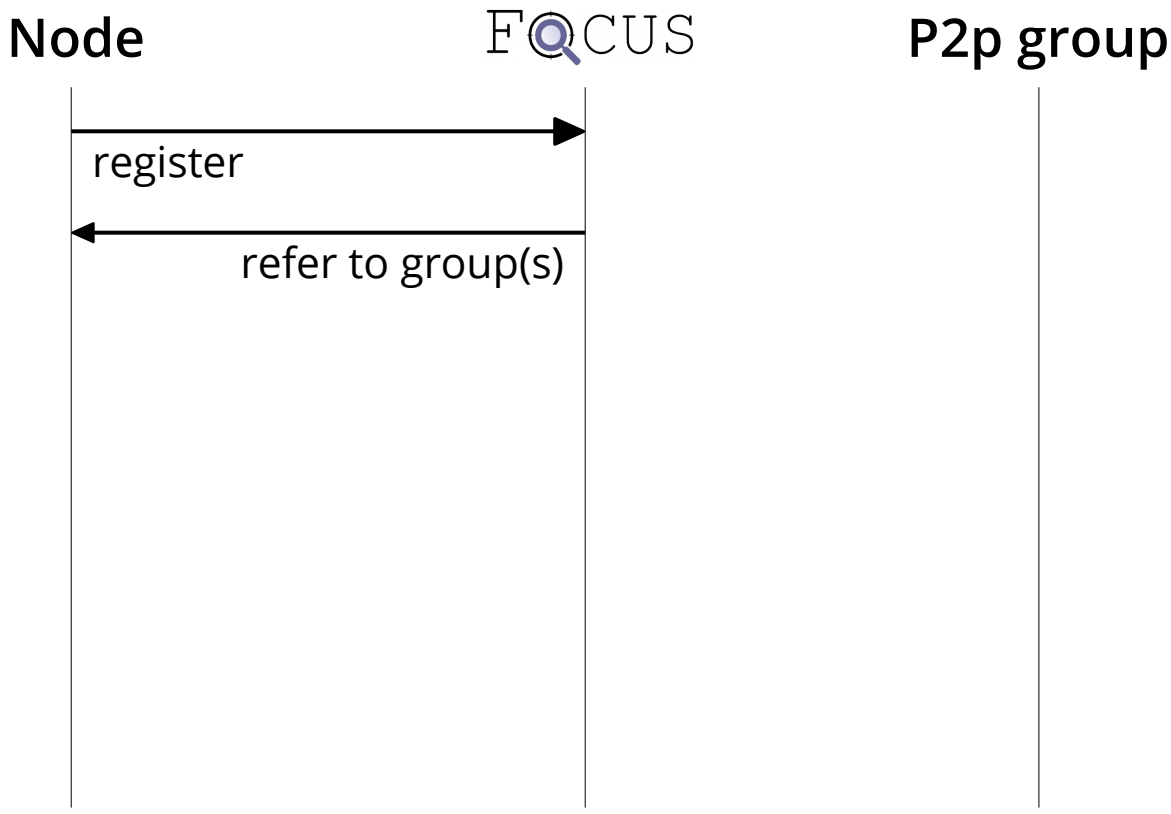
Dynamic Groups Management

Operations flow in FOCUS



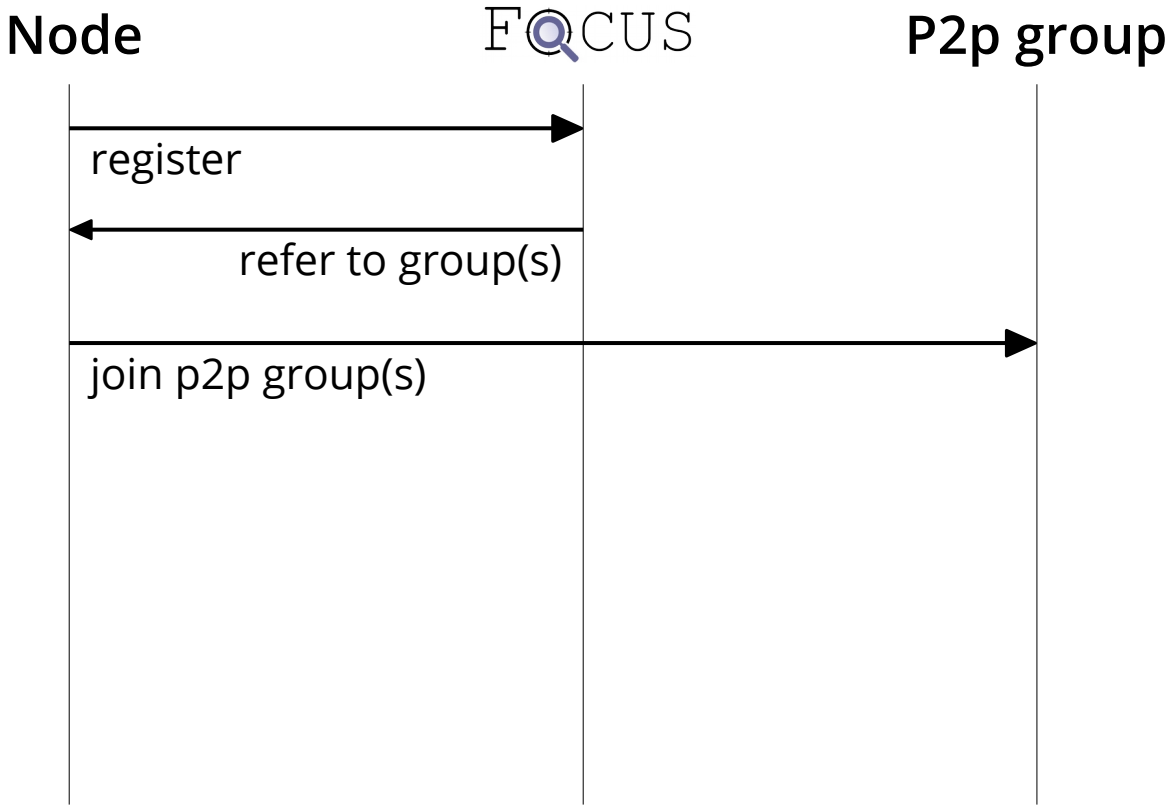
Dynamic Groups Management

Operations flow in FOCUS



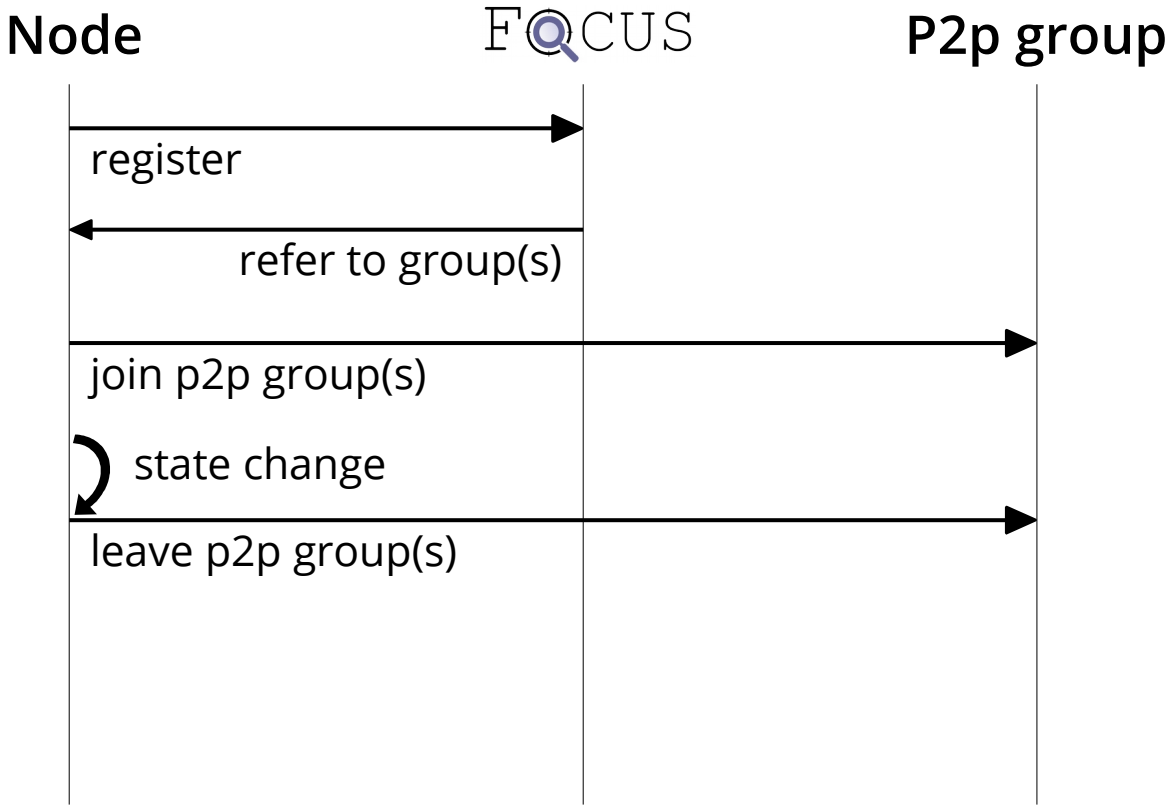
Dynamic Groups Management

Operations flow in FOCUS



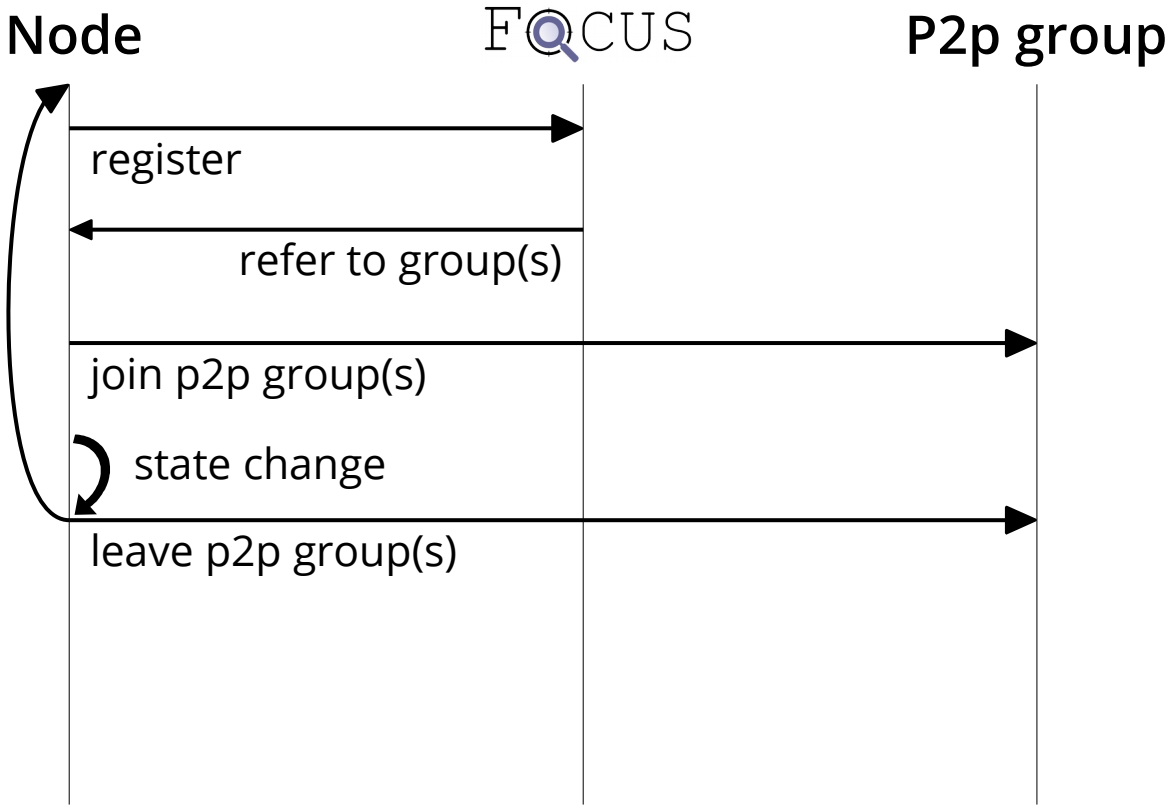
Dynamic Groups Management

Operations flow in FOCUS



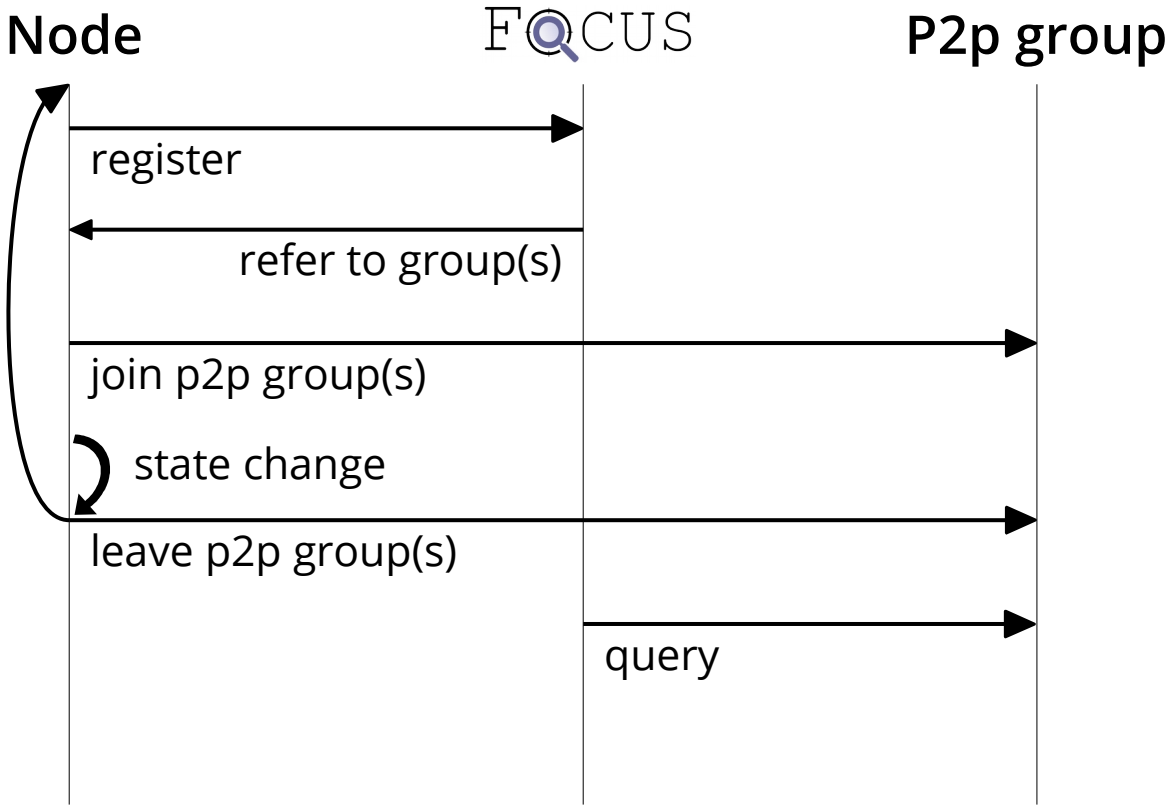
Dynamic Groups Management

Operations flow in FOCUS



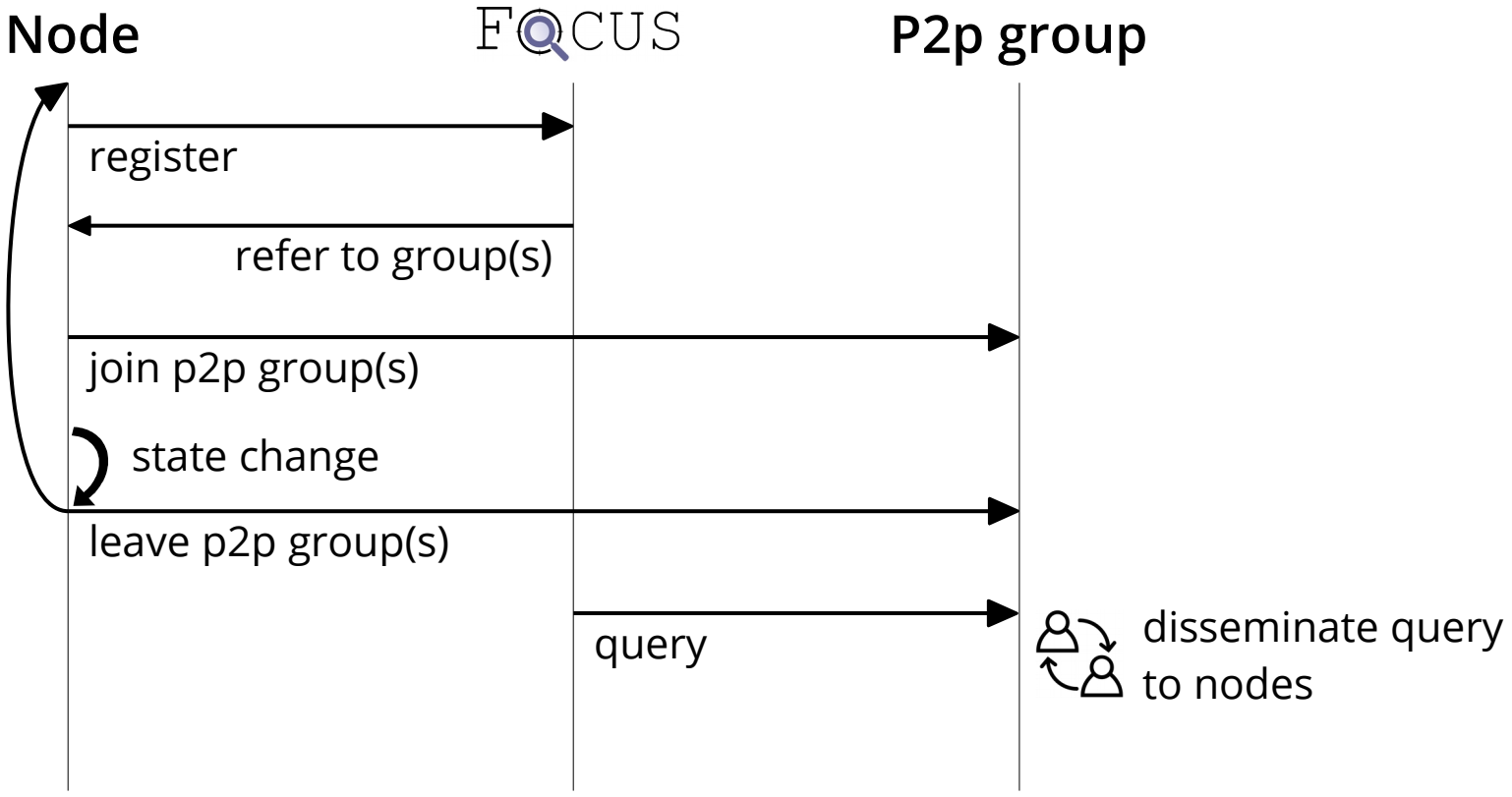
Dynamic Groups Management

Operations flow in FOCUS



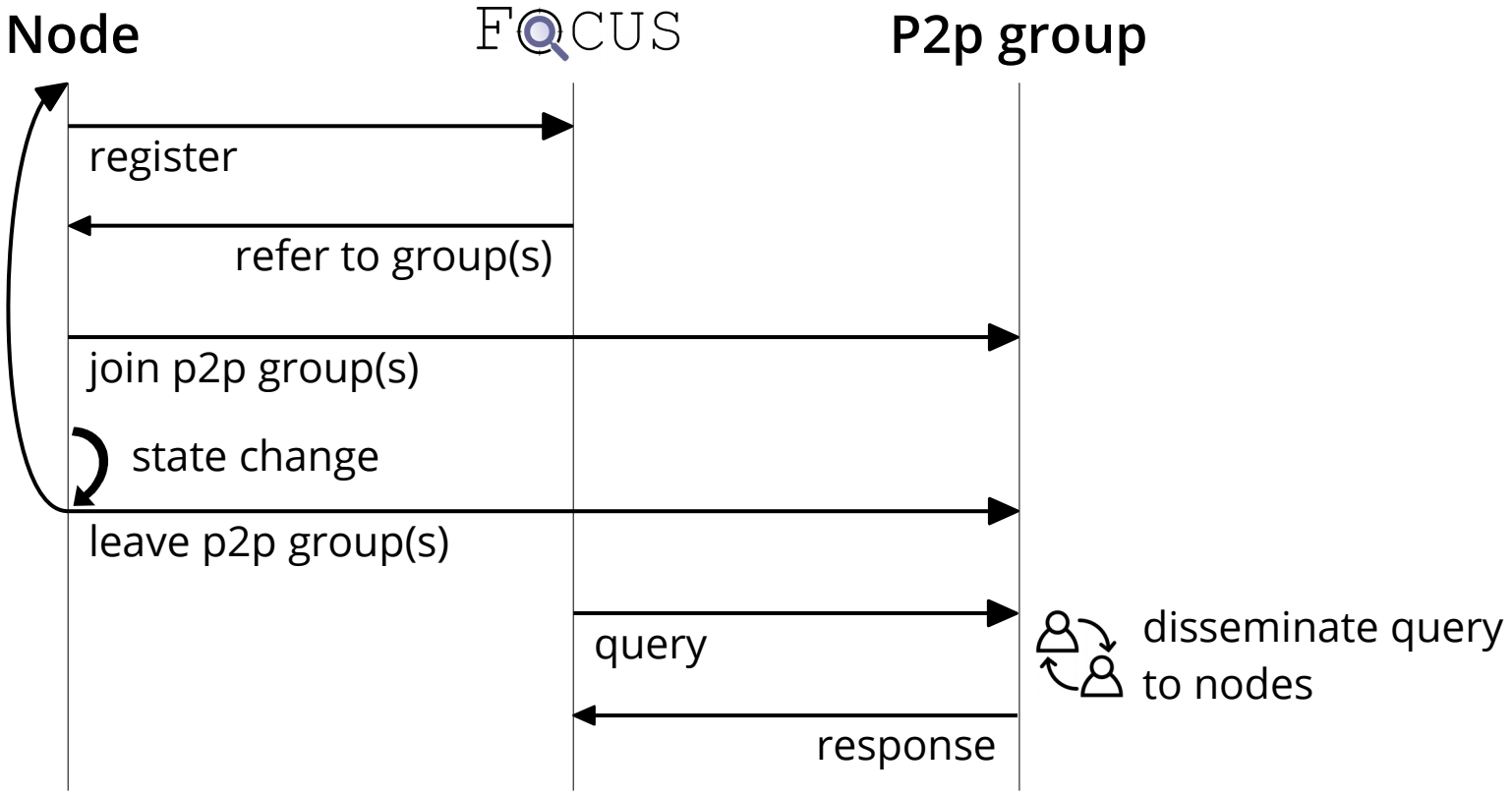
Dynamic Groups Management

Operations flow in FOCUS



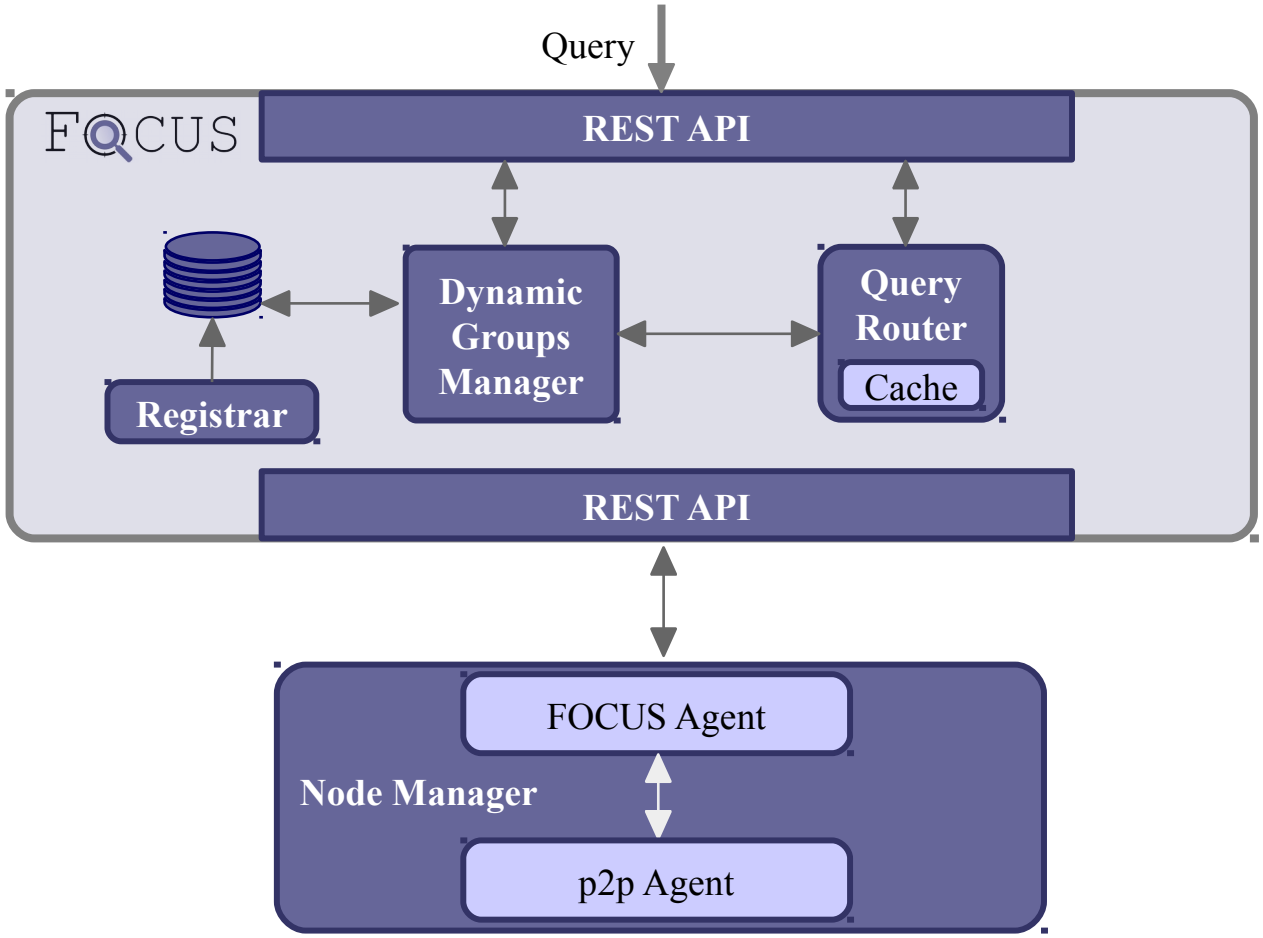
Dynamic Groups Management

Operations flow in FOCUS

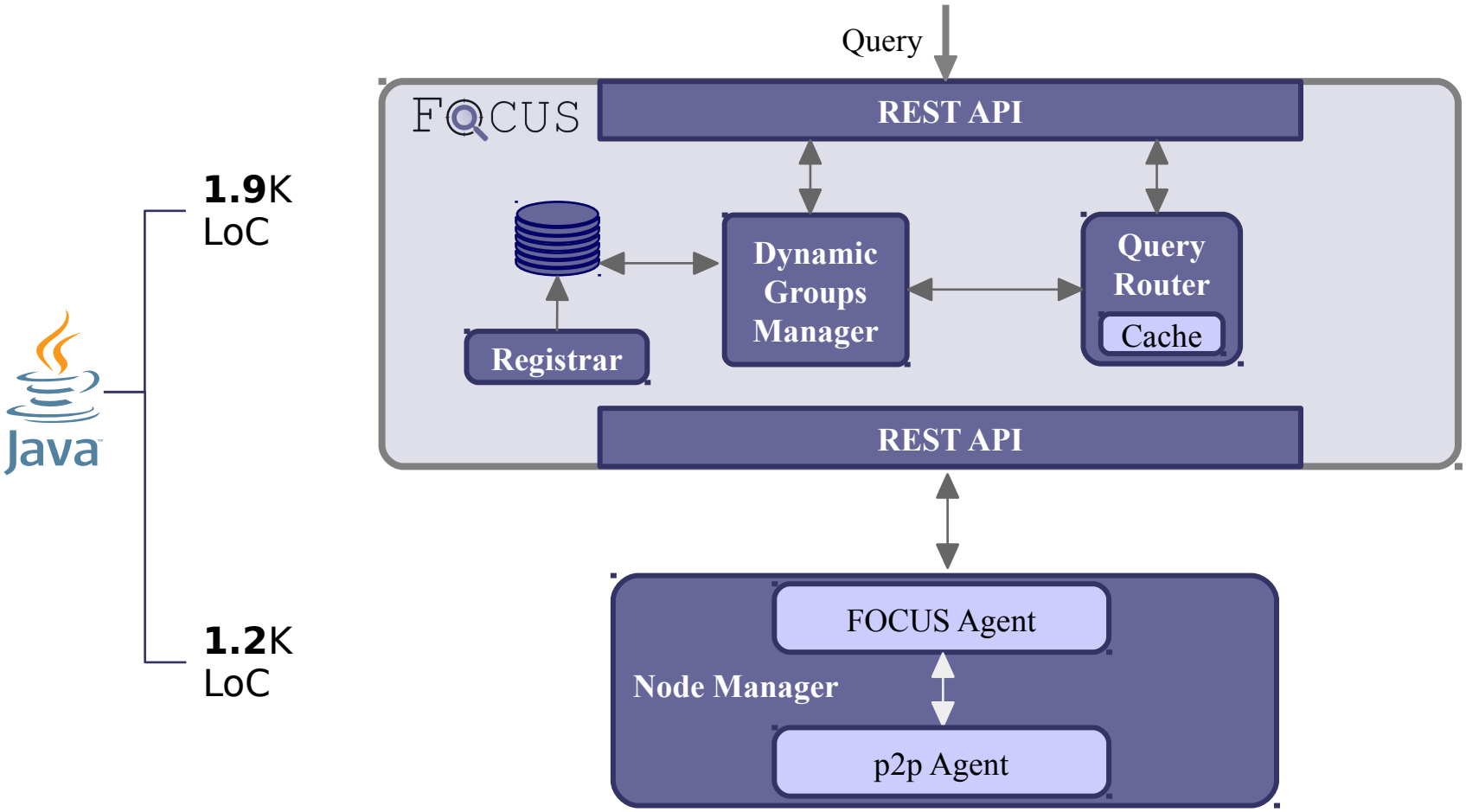


Implementation & Evaluation

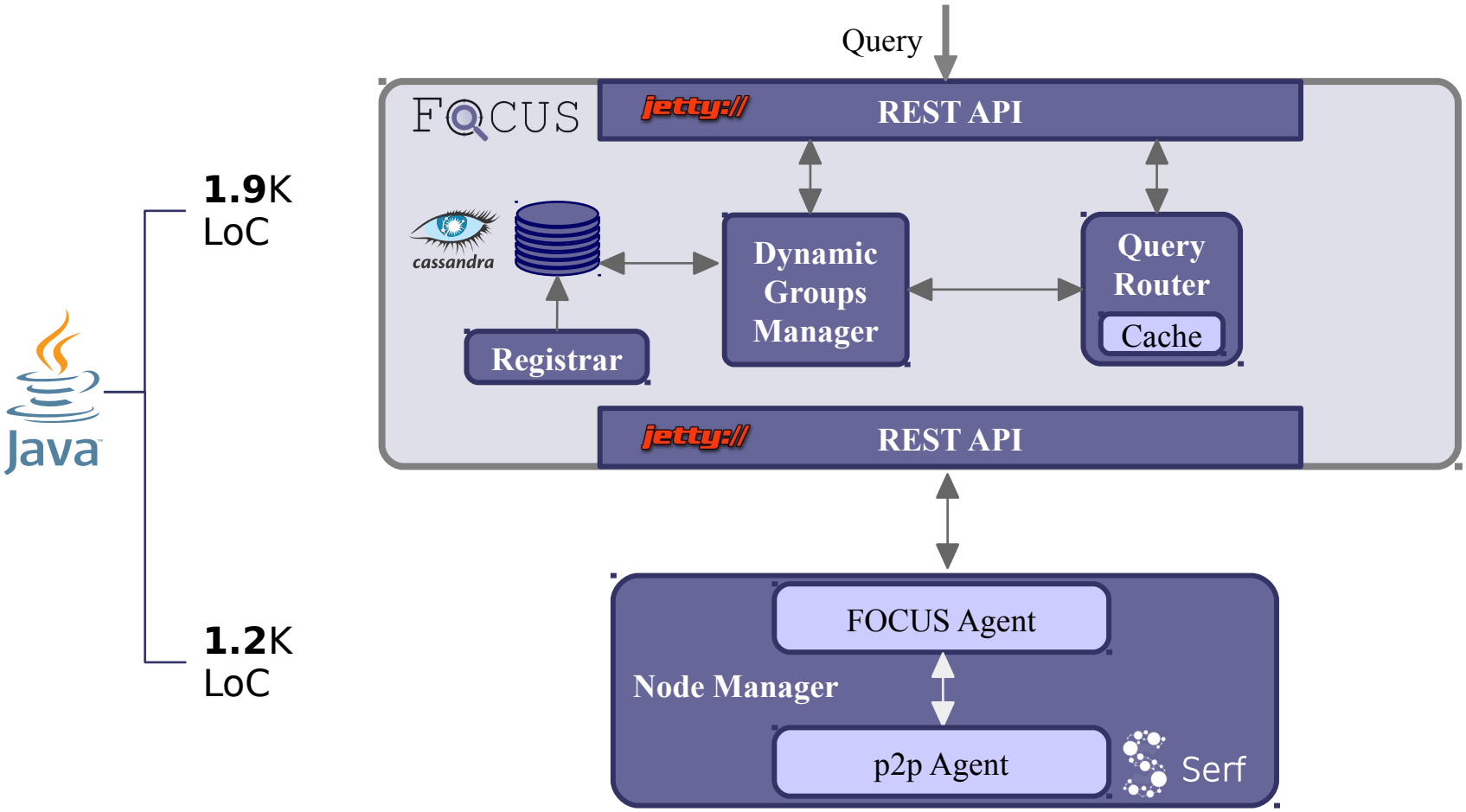
Implementation



Implementation



Implementation



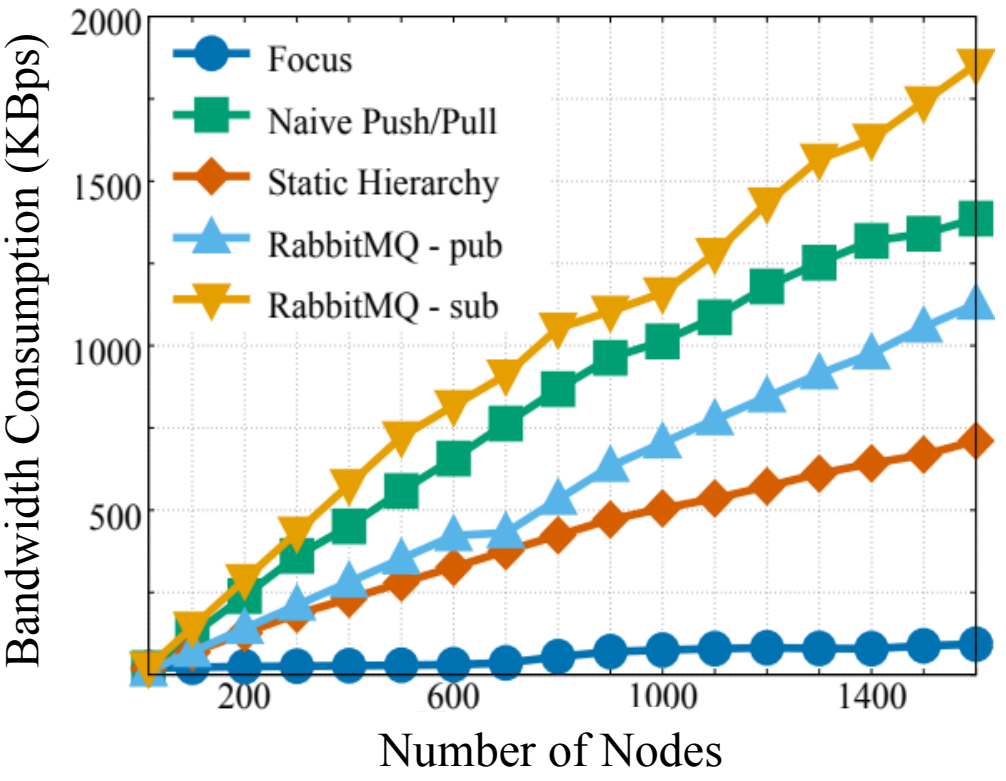
Evaluation

- Deployed in Amazon EC2
- 4 regions: Canada, California, Ohio, Oregon
- In each region: 8 VMs (4 vCPUs, 16GB RAM)
- FOCUS server running in California (same VM config)
- Testing up to 1600 simulated node agents

FOCUS vs. Other Approaches

Measuring BW Consumption at the Query Server
(frequency = 1 query/update per second)

- Naive Push/Pull
- Static Hierarchy
- RabbitMQ (Publish)
- RabbitMQ (Subscribe)

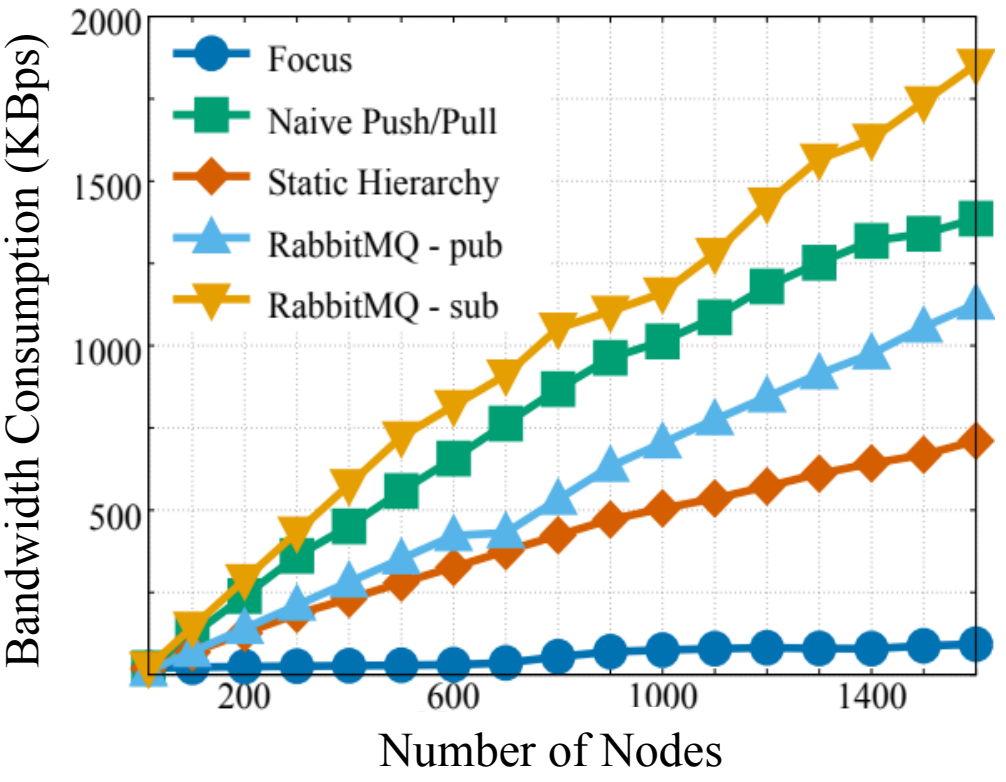


FOCUS vs. Other Approaches

Measuring BW Consumption at the Query Server
(frequency = 1 query/update per second)

- Naive Push/Pull
- Static Hierarchy
- RabbitMQ (Publish)
- RabbitMQ (Subscribe)

Adding a layer of intermediate nodes acting as aggregators

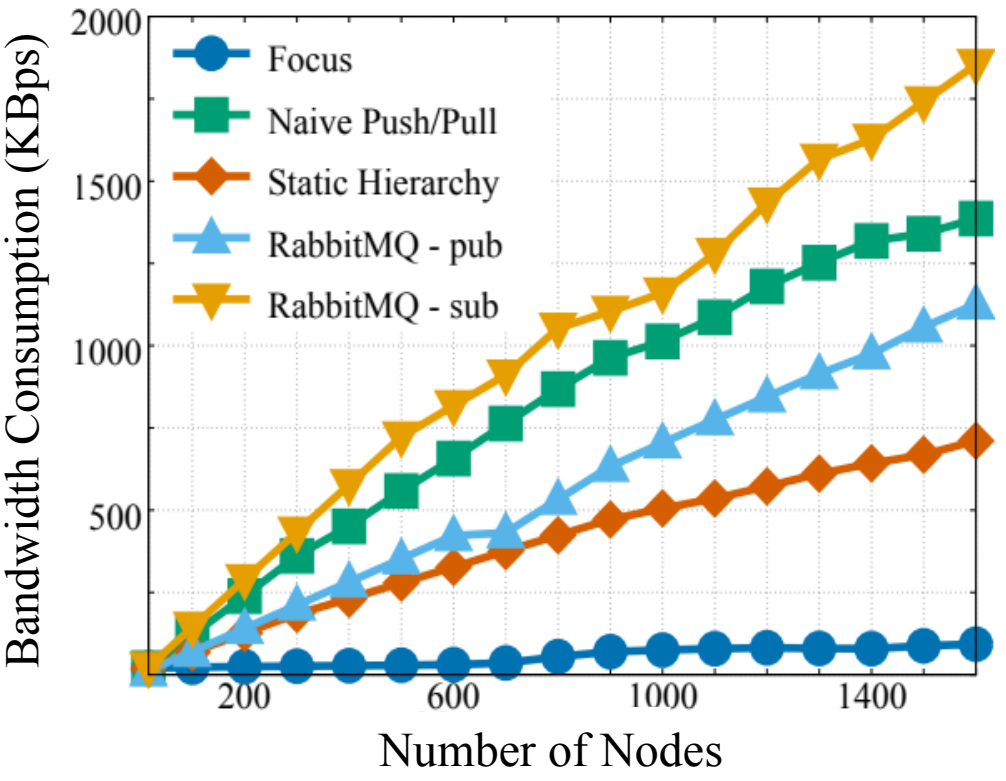


FOCUS vs. Other Approaches

Measuring BW Consumption at the Query Server
(frequency = 1 query/update per second)

- Naive Push/Pull
- Static Hierarchy
- RabbitMQ (Publish)
- RabbitMQ (Subscribe)

Nodes publish their state (i.e.,fancy push)

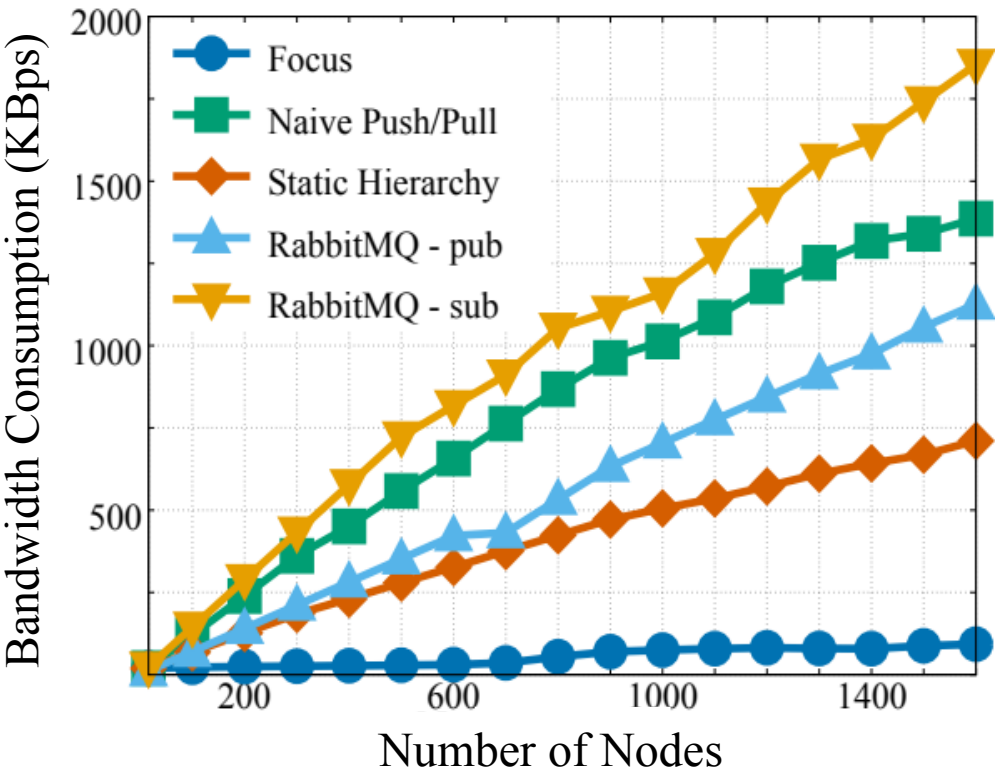


FOCUS vs. Other Approaches

Measuring BW Consumption at the Query Server
(frequency = 1 query/update per second)

- Naive Push/Pull
- Static Hierarchy
- RabbitMQ (Publish)
- RabbitMQ (Subscribe)

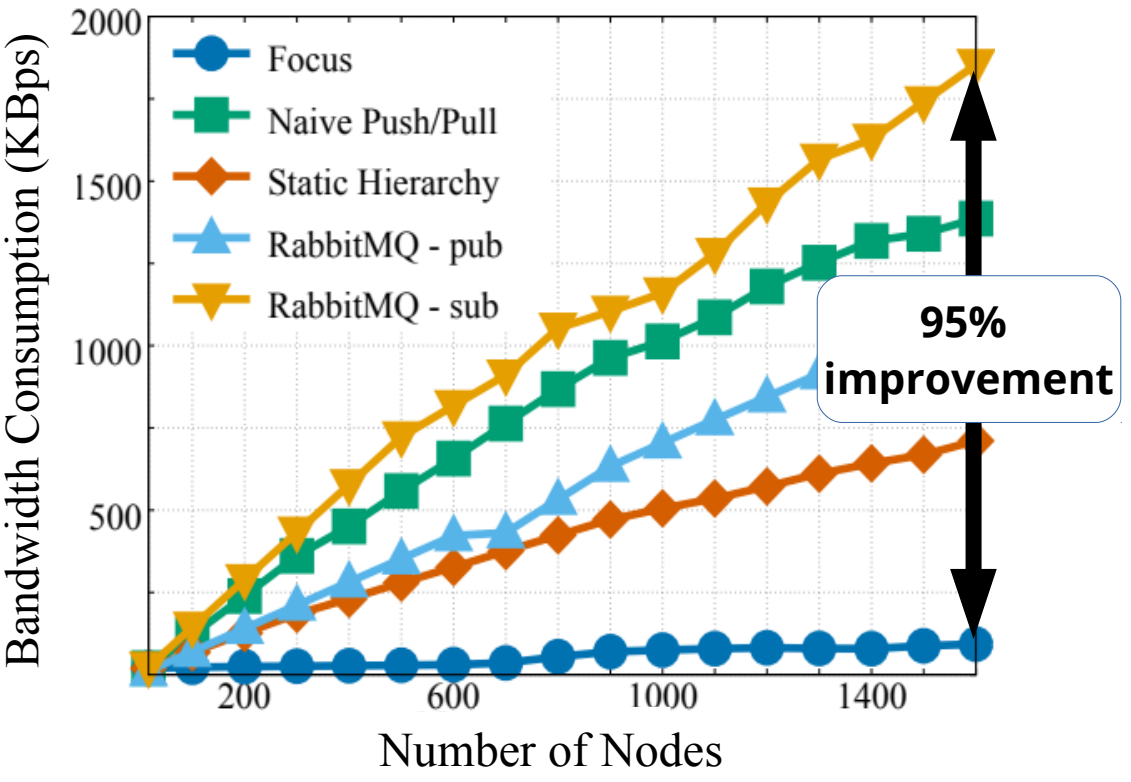
Nodes subscribe for queries



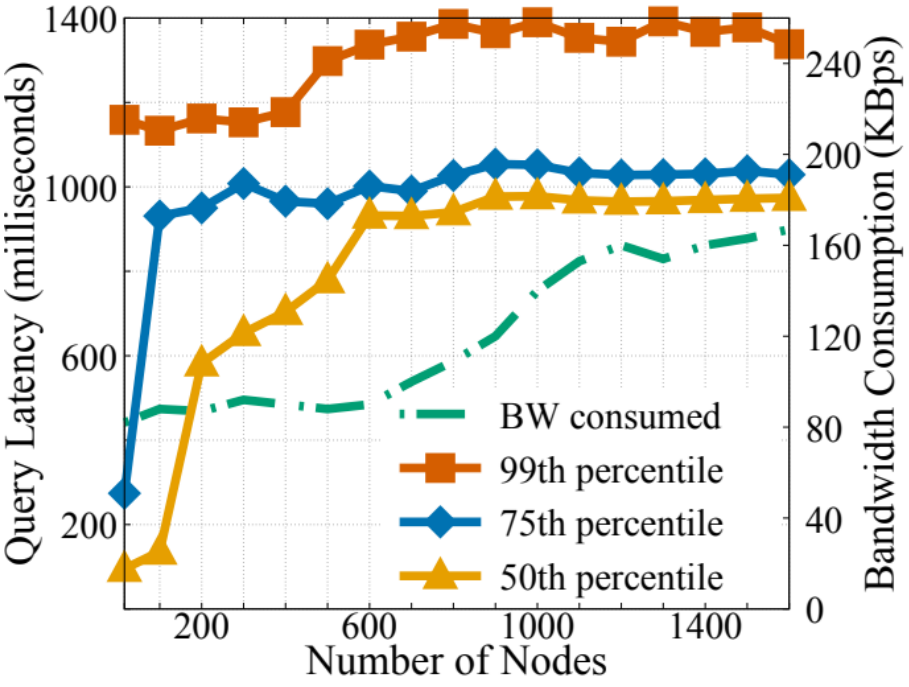
FOCUS vs. Other Approaches

Measuring BW Consumption at the Query Server
(frequency = 1 query/update per second)

- Naive Push/Pull
- Static Hierarchy
- RabbitMQ (Publish)
- RabbitMQ (Subscribe)



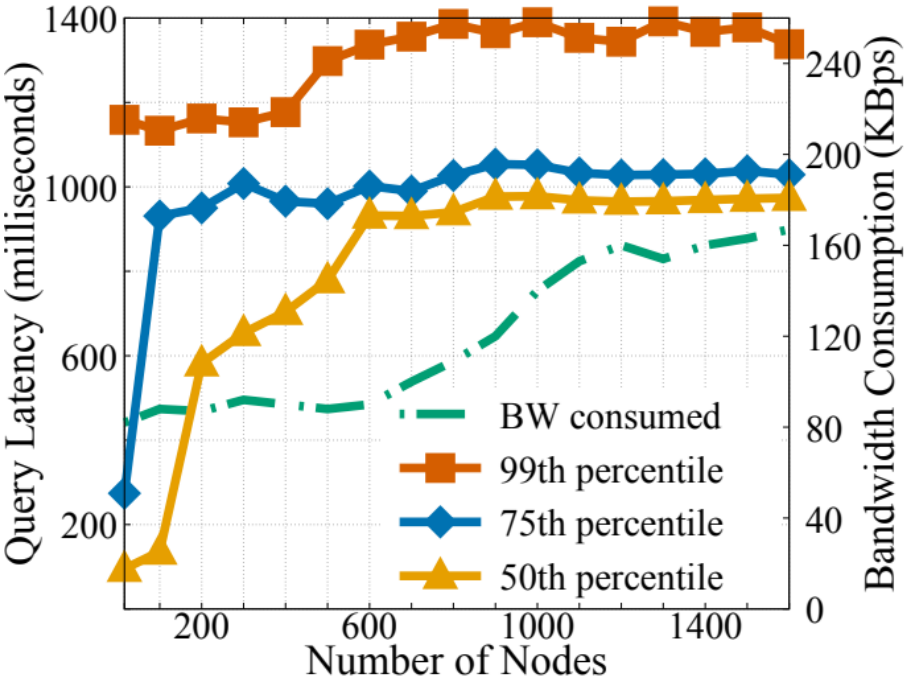
FOCUS with Real-world Cloud Traces*



* “Chameleon Cloud: A configurable experimental environment for largescale cloud research,” https://www.chameleoncloud.org/.

FOCUS with Real-world Cloud Traces*

75K OpenStack VM placement requests

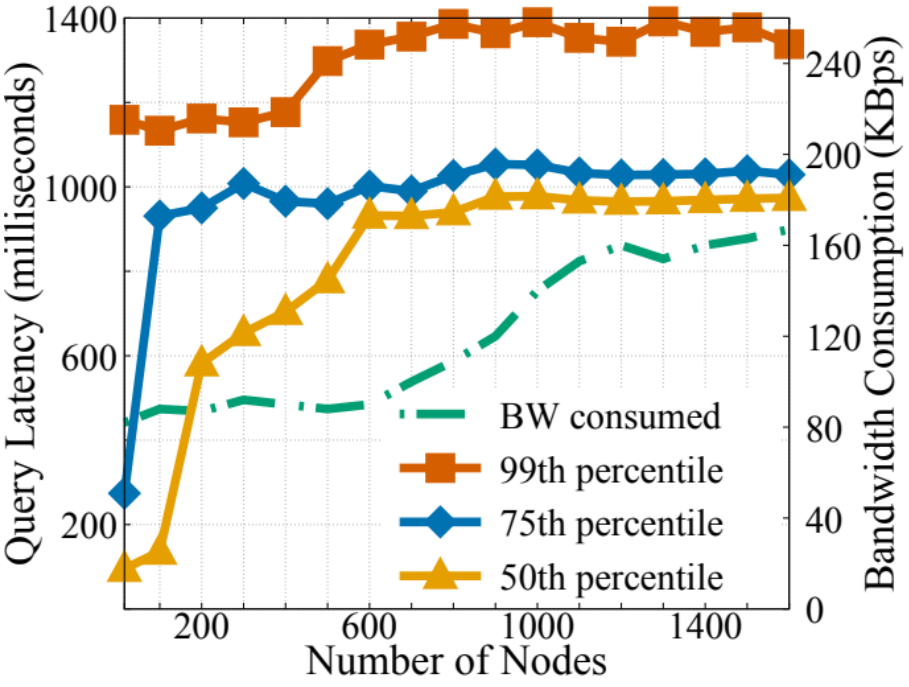


* “Chameleon Cloud: A configurable experimental environment for largescale cloud research,” https://www.chameleoncloud.org/.

FOCUS with Real-world Cloud Traces*

75K OpenStack VM placement requests

Replayed at accelerated rate (15,000x)



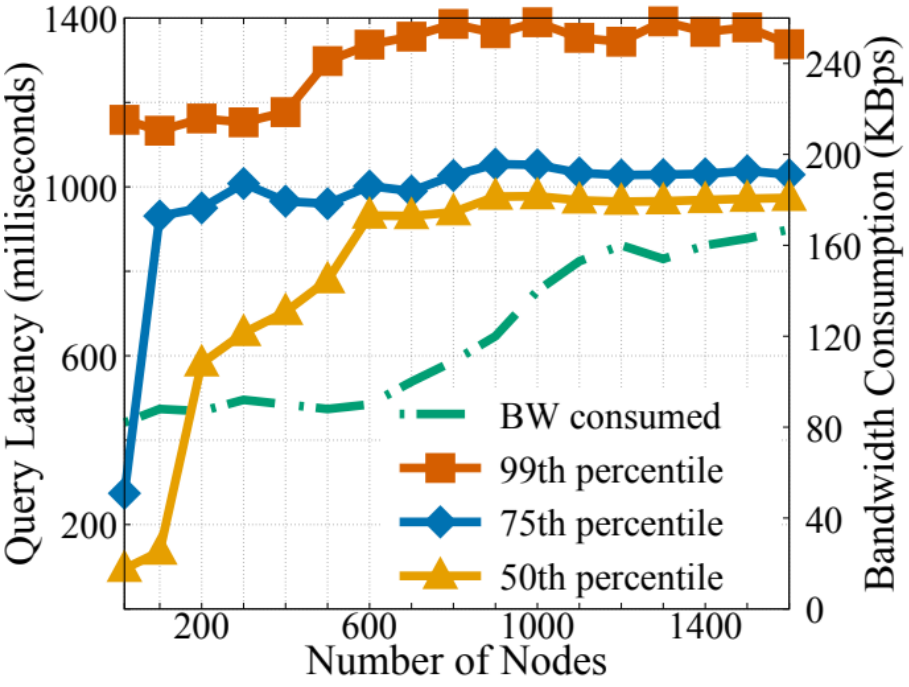
* “Chameleon Cloud: A configurable experimental environment for largescale cloud research,” https://www.chameleoncloud.org/.

FOCUS with Real-world Cloud Traces*

75K OpenStack VM placement requests

Replayed at accelerated rate (15,000x)

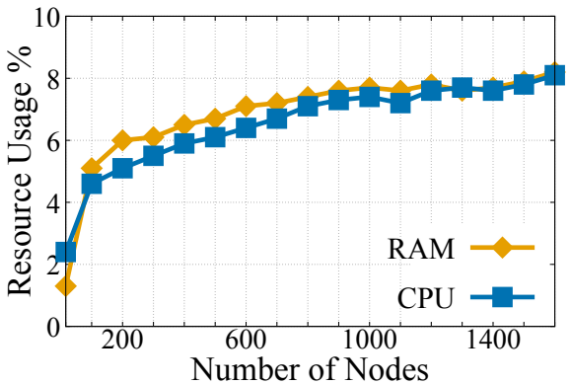
Latency stabilizes after 600 nodes
→ because group size is capped (~150 nodes per group)



* “Chameleon Cloud: A configurable experimental environment for largescale cloud research,” https://www.chameleoncloud.org/.

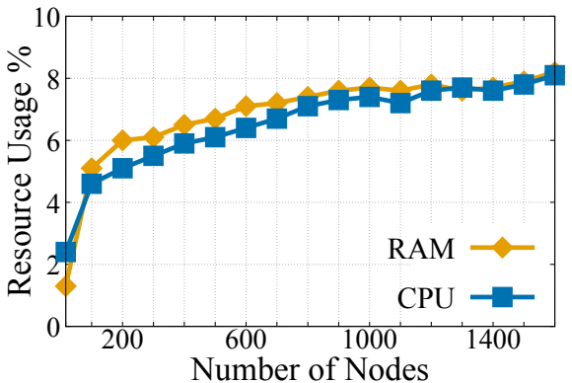
Microbenchmarks

Resource usage of the FOCUS server (40 queries/s)

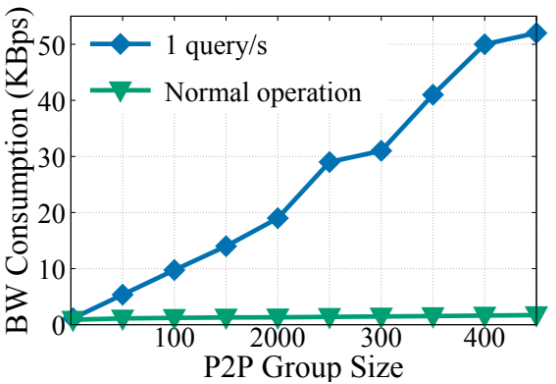


Microbenchmarks

Resource usage of the FOCUS server (40 queries/s)

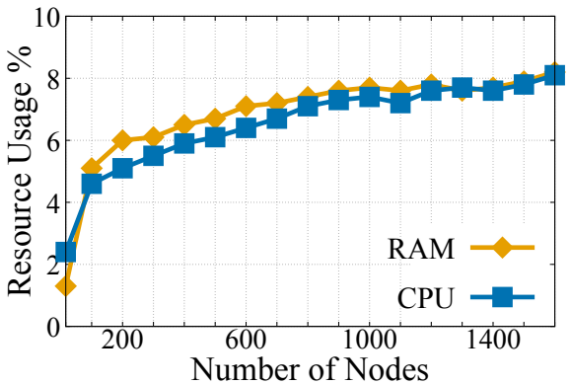


Overhead imposed by node agent (KBps)

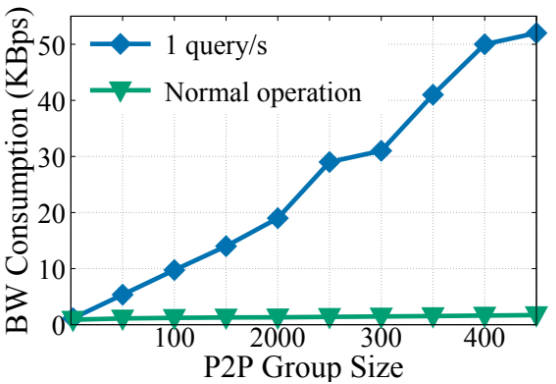


Microbenchmarks

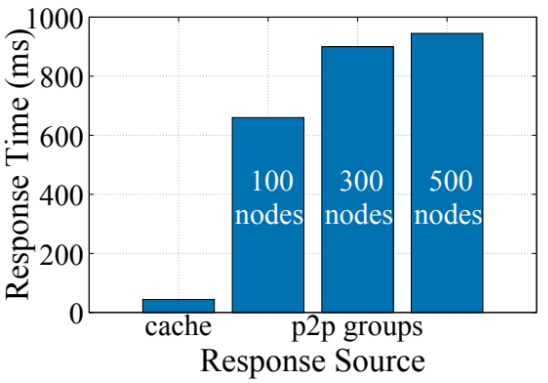
Resource usage of the FOCUS server (40 queries/s)



Overhead imposed by node agent (KBps)



Query response time for different group sizes



Conclusion

- **Current systems' scalability is limited**
  - This is due to tightly-coupled node management

Conclusion

- **Current systems' scalability is limited**
  - This is due to tightly-coupled node management
- **FOCUS is scalable search service**
  - Employs a *loosely-coupled* node management (p2p)
  - *Scales* better than current approaches (15x improvement)
  - Imposes *minimal* overhead on nodes
  - *Integrates* well with current systems

Thank You!

Questions?