A Generic API for Load Balancing in Structued P2P Systems

Maeva Antoine, Laurent Pellegrino, Fabrice Huet and Françoise Baude

University of Nice Sophia-Antipolis (France), CNRS, I3S, UMR 7271

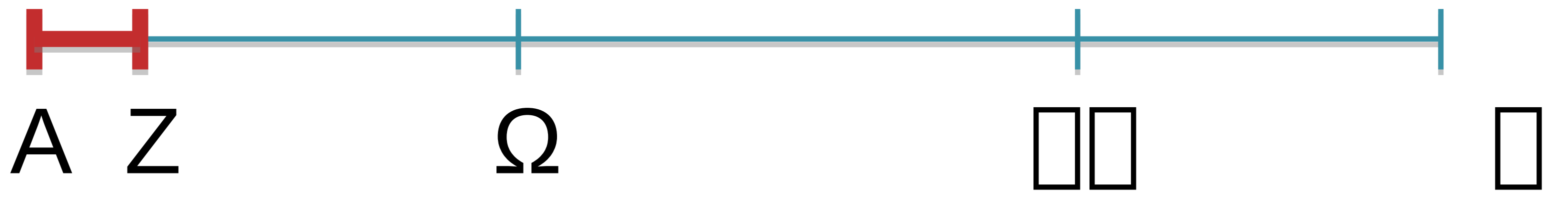
Motivation

- P2P: large scale solution for Big Data management systems (Cassandra, CouchDB…)
- However, key issue with distributed systems: Load Balancing

Load Imbalance Issues

When managing real world datasets:

- Very biased data (ex: Unicode)



- Large workloads sent to very few nodes
- Churn
- Heterogeneity between peers

(bandwidth, CPU, storage capacities)

Load Balancing Solutions

- Plenty of existing load balancing strategies
- Hard to anticipate the most efficient strategy for a particular system
- Many parameters to take into account

Summary

- Criteria to choose a load balancing strategy
- How existing papers match our criteria
- API for load balancing
- Experiments on our own storage system

How to Build a Strategy

- How is load information exchanged?
- How to trigger load balancing?
- What should be balanced?

How to Build a Strategy

How is load information exchanged?

- Load information exchange? *What, how and when*
- Load information recipients? *Who informs who*

How to Build a Strategy

How to trigger load balancing?

- Load criteria? *Resource (CPU, disk space, ...) & operation (item lookup, insertion, ...)*
- Load state estimation? *How to estimate load*
- Load balancing decision? *When to trigger rebalance*

How to Build a Strategy

What has to be moved?

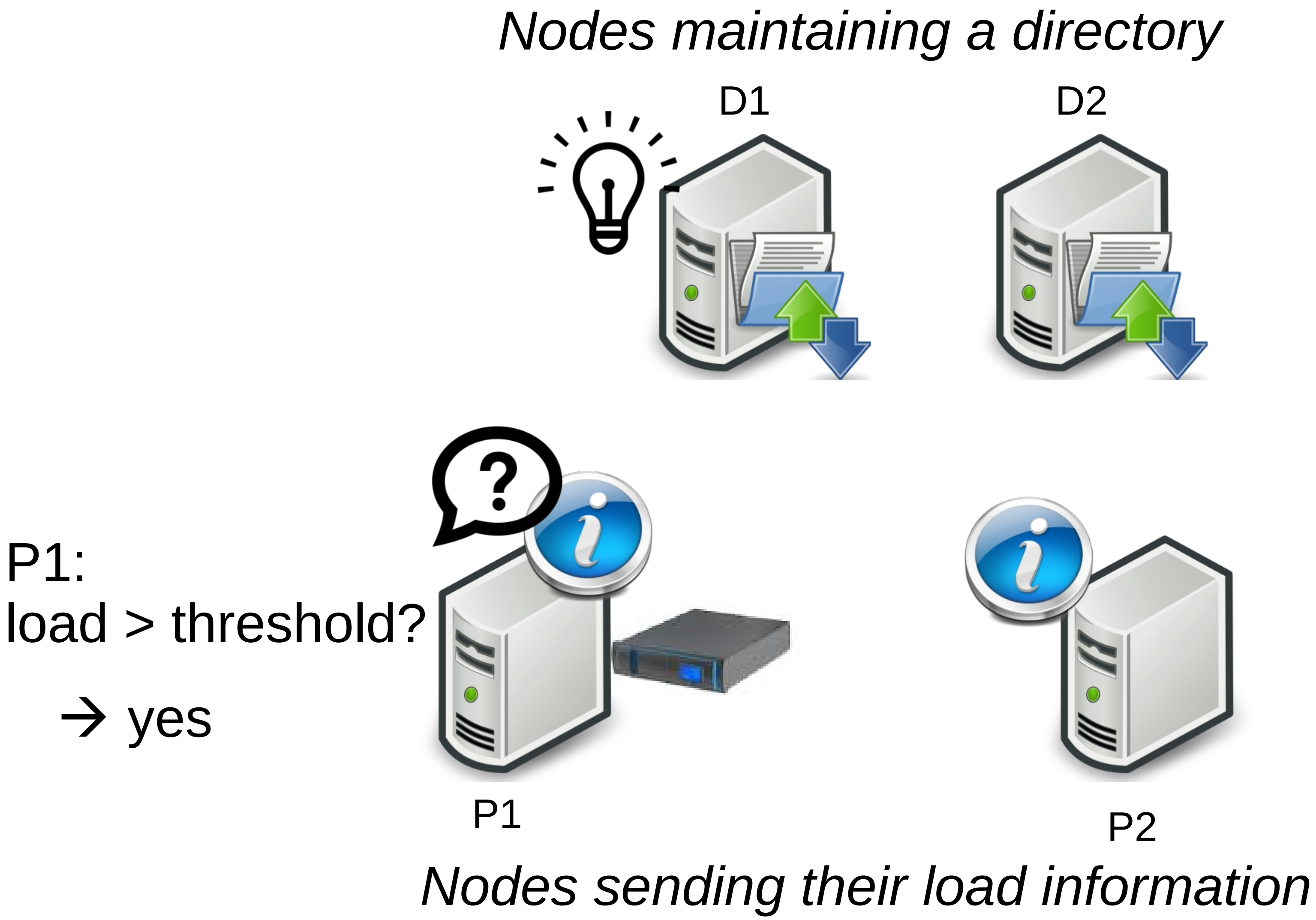
- Load balancing method? *How to balance load*
- Load to move? *What and how much to move*
- Target? *Who will receive the load to move*

Existing Load Balancing Strategies

- 3 different strategies
- Among the most cited for this topic
- Differences:
  - Load balancing triggered after various events
  - Context: pub/sub, virtual servers, data storage

Strategy #1

- Rao et al. (Berkeley)



Nodes maintaining a directory
D1
D2
P1:
load > threshold?
→ yes
P1
P2
Nodes sending their load information

Strategy #1

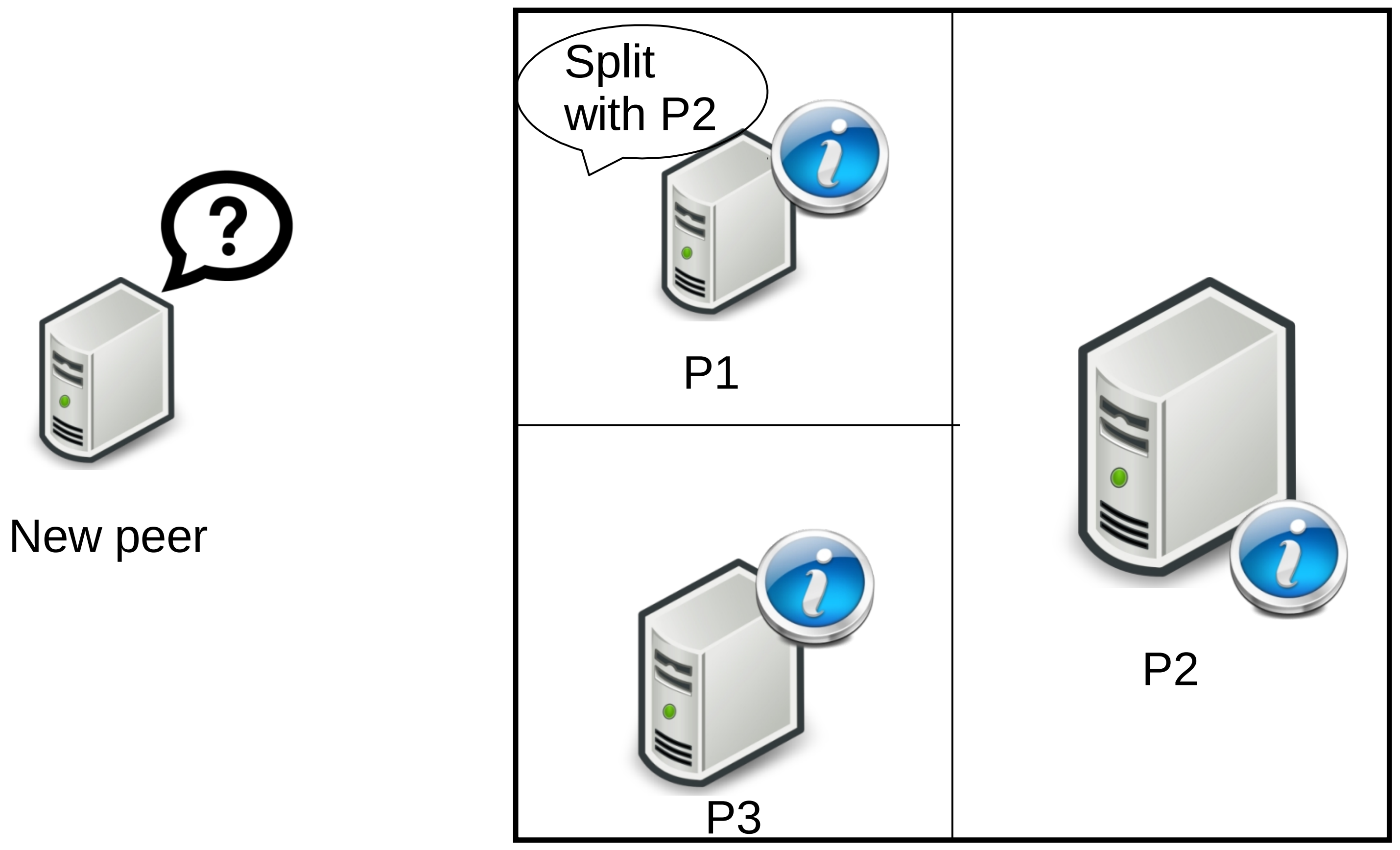
- ***How is load information exchanged?***
  - *Periodic push and pull calls from peers to directories.*

- ***How to trigger load balancing?***
  - *Periodically compare virtual servers load with internal threshold.*

- ***What has to be moved?***
  - *Transfer a virtual server to a light node.*

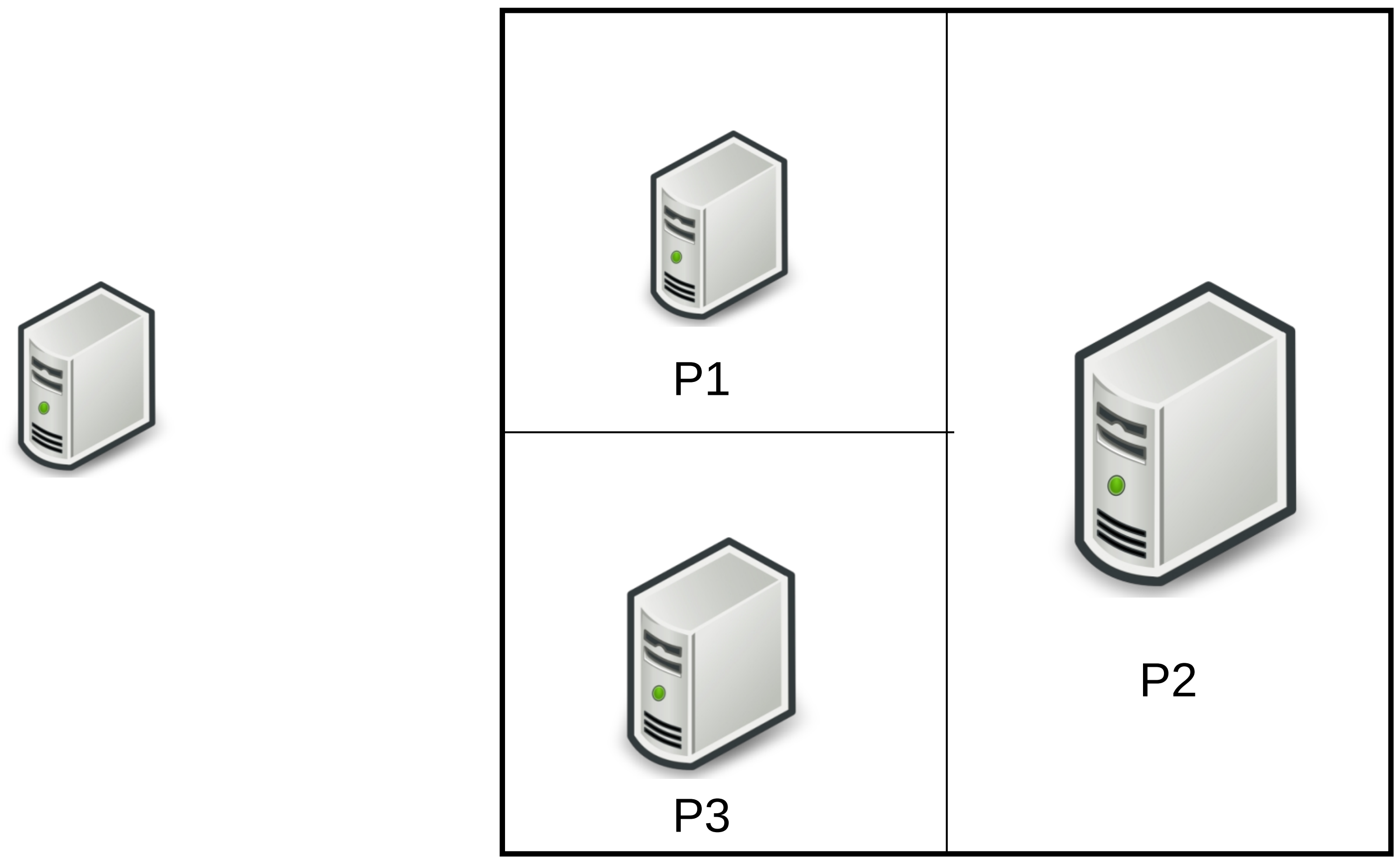
Strategy #2

- Gupta et al. (University of California)



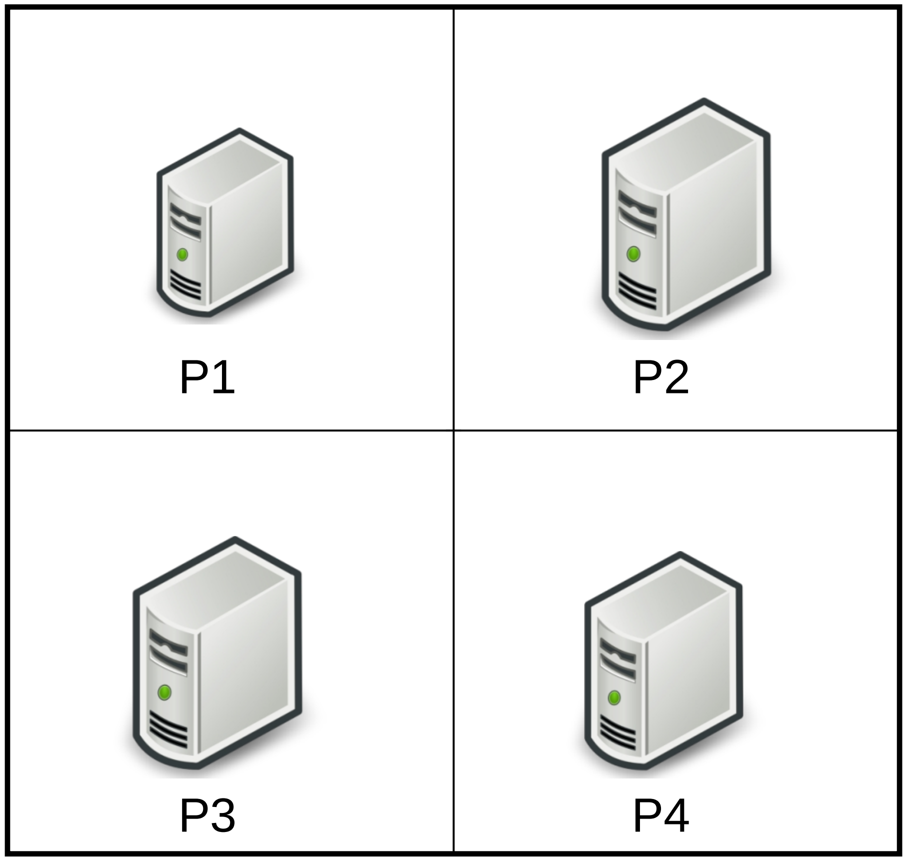
Strategy #2

- Gupta et al. (University of California)



Strategy #2

- Gupta et al. (University of California)



Strategy #2

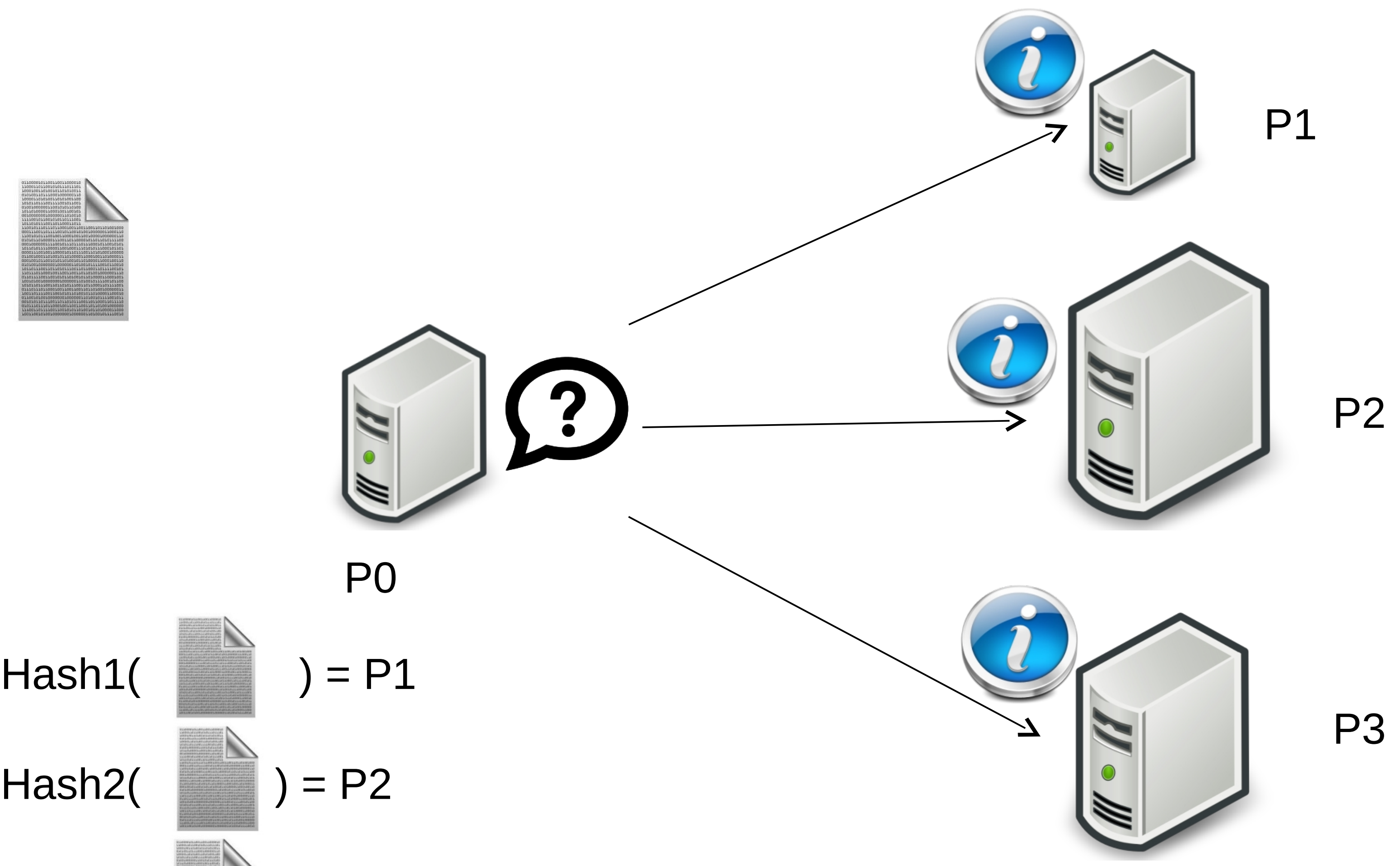
- ***How is load information exchanged?***
  - *Periodic push calls between peers.*

- ***How to trigger load balancing?***
  - *When a new peer joins the system: find the most loaded with subscriptions.*

- ***What has to be moved?***
  - *Half of the heavy peer's area to the new peer.*

Strategy #3

- Byers et al. (Boston & Harvard University)



Hash1( ) = P1

Hash2( ) = P2

Hash3( ) = P3

Strategy #3

- ***How is load information exchanged?***
  - *Hash_n(item) to contact n peers.*

- ***How to trigger load balancing?***
  - *When inserting an item: find the least loaded peer among n.*

- ***What has to be moved?***
  - *The item to insert to the lightest node.*

Load Balancing Implementation

Many different criteria

=

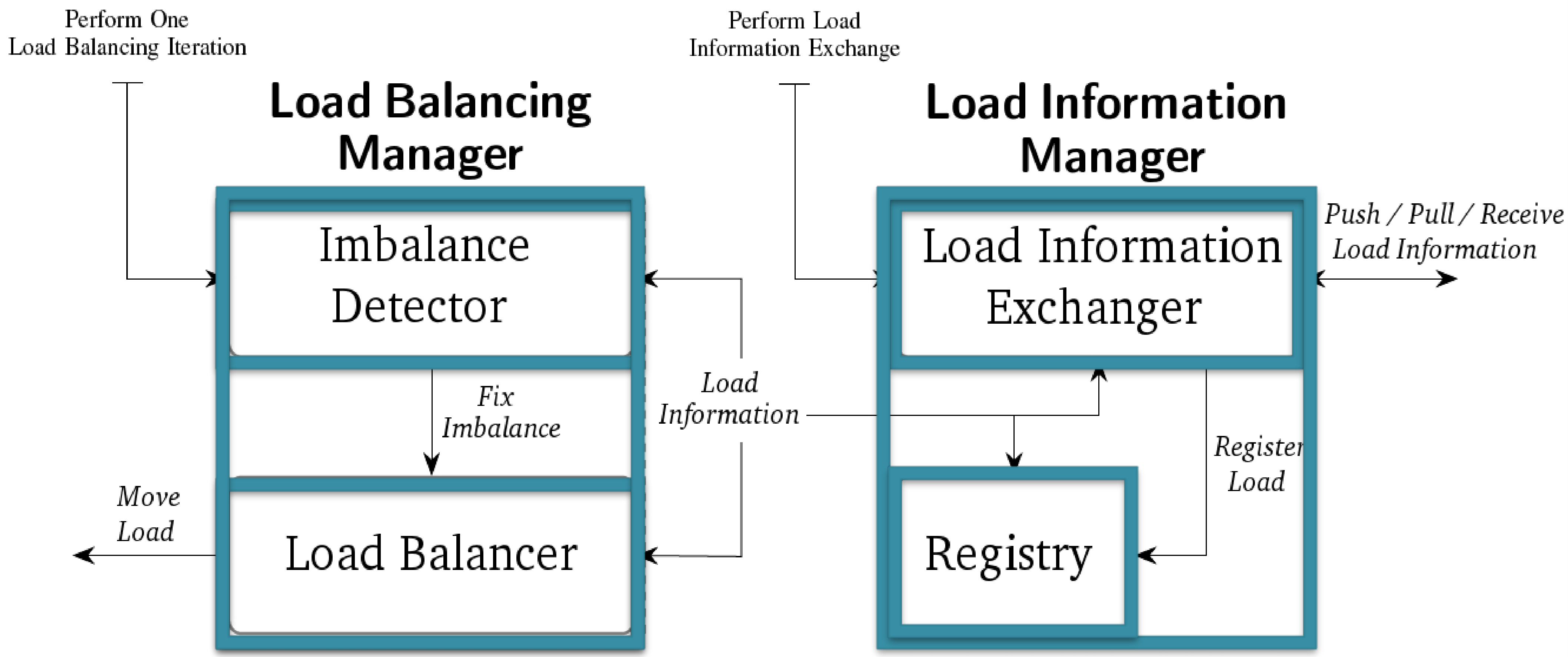
Many strategies possible

=

Many specific implementations

- Identify key points for a generic API to implement any strategy

Generic API Components



What has to be moved?

How to trigger load balancing?

How is information exchanged?

Use Case: Event Cloud

- Continuous storage and retrieval in a Big Data environment

- Distributed RDF quadruple store (Semantic web)

- RDF term = set of URIs = biased data

Implementation on Event Cloud

- Load information exchange:
  - None (internal threshold)
  - With neighbors
- Load criteria:
  - Number of items per peer
  - CPU used for subscription matching

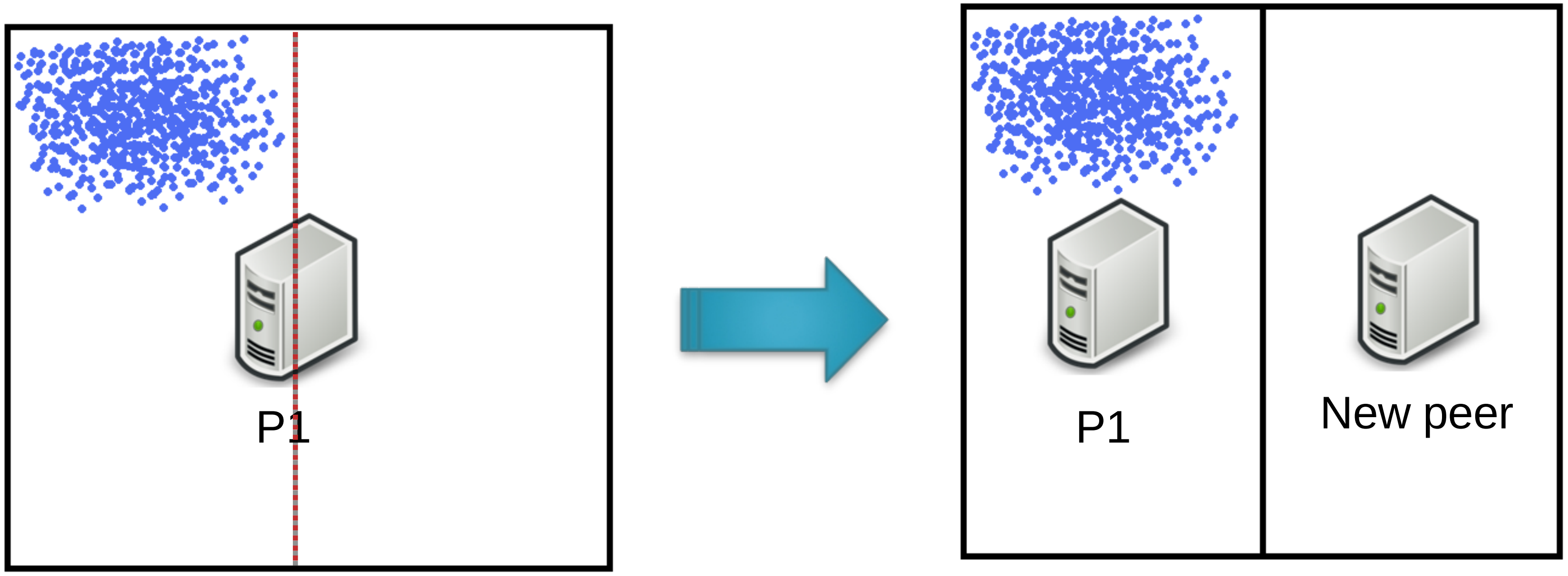
Imbalance Detector

```
make_decision() {
if (load > threshold)
get_neighbors_load())
    }…
}
```

10 lines of code required to modify strategy

Implementation on Event Cloud

Middle vs. Centroid Split:



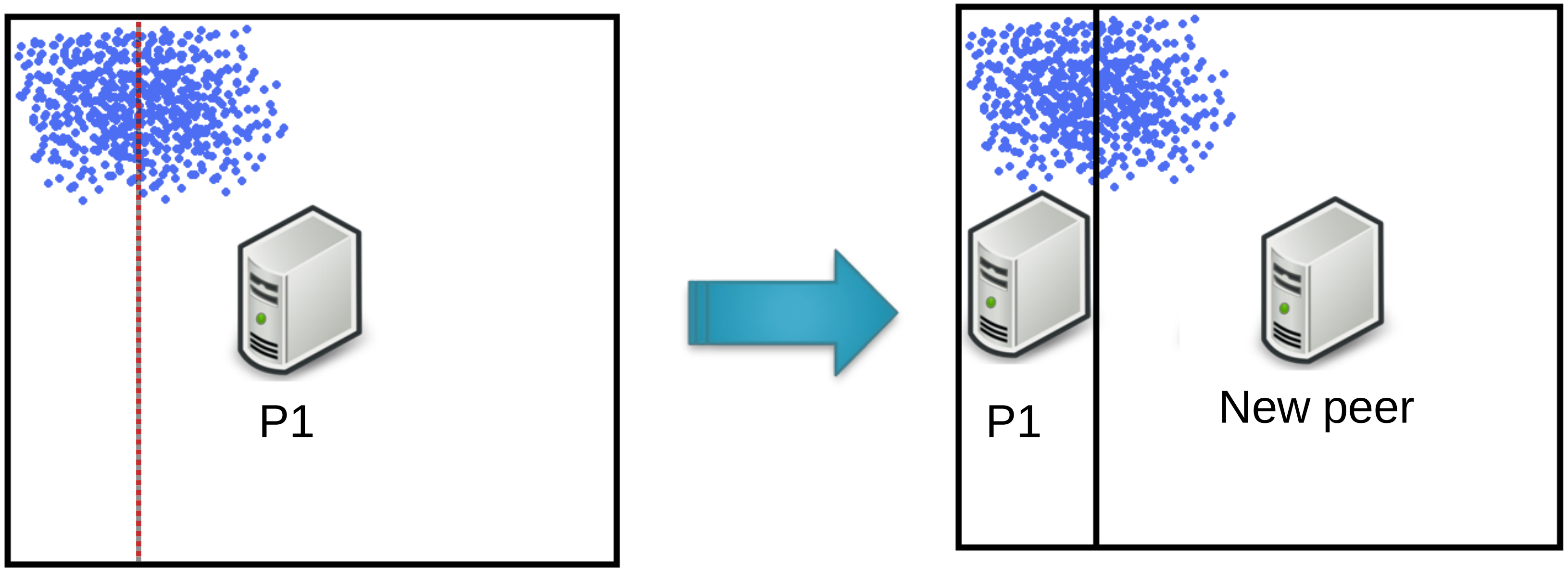
Peer managing data (blue dots)

New peer joins at Middle value

```
P1.select_load_to_move(){
→get_data_from_middle()
}
```

Implementation on Event Cloud

Middle vs. **Centroid** Split:



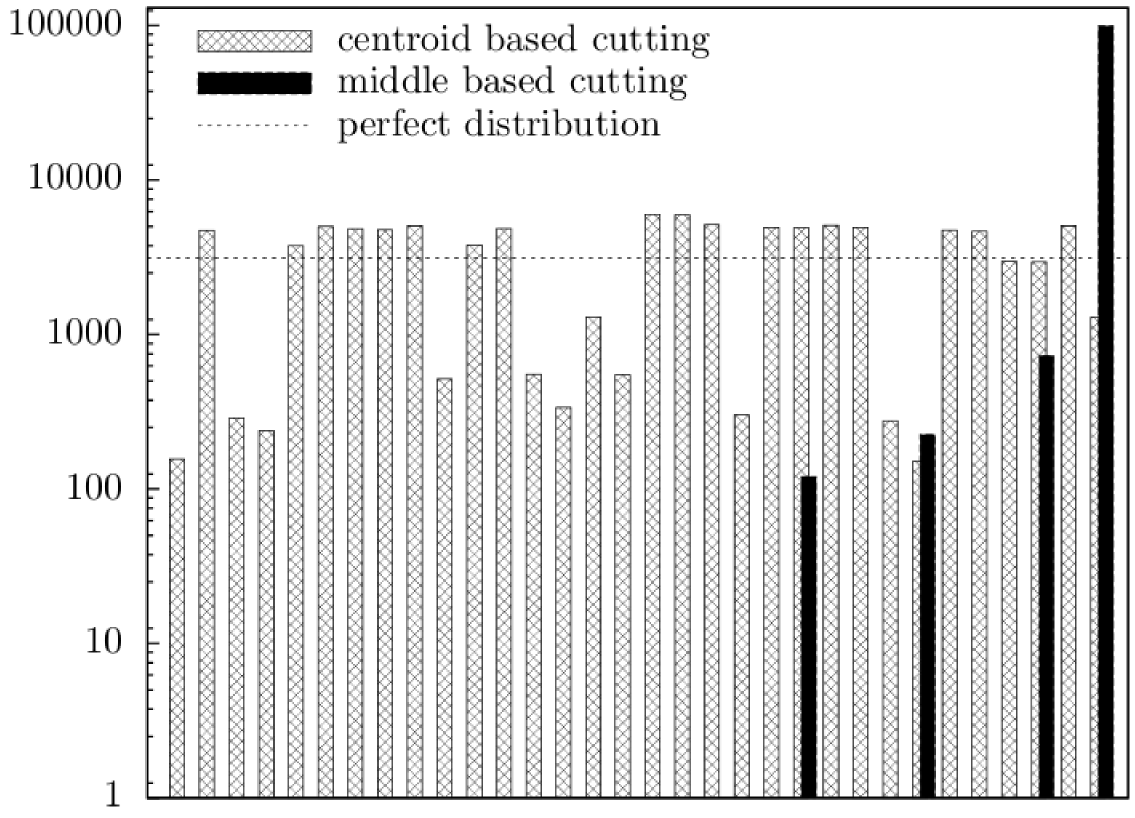
Peer managing data (blue dots)

New peer joins at Centroid value

```
P1.select_load_to_move(){
→get_data_from_centroid()
}
```

Implementation on Event Cloud

Number of Quadruples per peer



Distribution among 32 peers

Conclusion

- Flexible API
- Separation with the rest of the code
- Implemented on our storage system
- Compatible with famous existing strategies
- Principles applicable on non P2P systems

The End

- Thank you!
- Questions?