Grid'BnB: A Parallel Branch and Bound Framework for Grids

Denis Caromel, Alexandre di Costanzo,
INRIA - CNRS - Université de Nice Sophia Antipolis, France

Laurent Baduel, and Satoshi Matsuoka
Tokyo Institute of Technology, Japan

December 21th, 2007
HiPC 2007 - Goa, India

The Big Picture

- Objective

  Solving NP-hard optimization problems

- Approach

  Parallel Branch and Bound & Grids

- Contributions

  1. *Branch and Bound framework for Grids*
  2. *Grid node localization*
  3. *Large-scale experiments*

Agenda

- Context, Problematic, Objectives
- Contributions
  - *Grid'BnB*: Parallel Branch and Bound for Grids
  - Grid Node Localization
  - Large-scale Experiments
- Conclusion & Perspectives

Context

- Search/Combinatorial Optimization/NP-Hard Problems
  - costly to solve (finding the best solution)
  - focus on combinatorial optimization problems (COPs)
- Branch and Bound (B&B)
  - well adapted for solving COPs [Papadimitriou 98]
  - relatively easy to provide parallel implementation
- Grid Computing
  - large pool of resources
  - challenges: latency, scalability, *etc.*

Branch and Bound

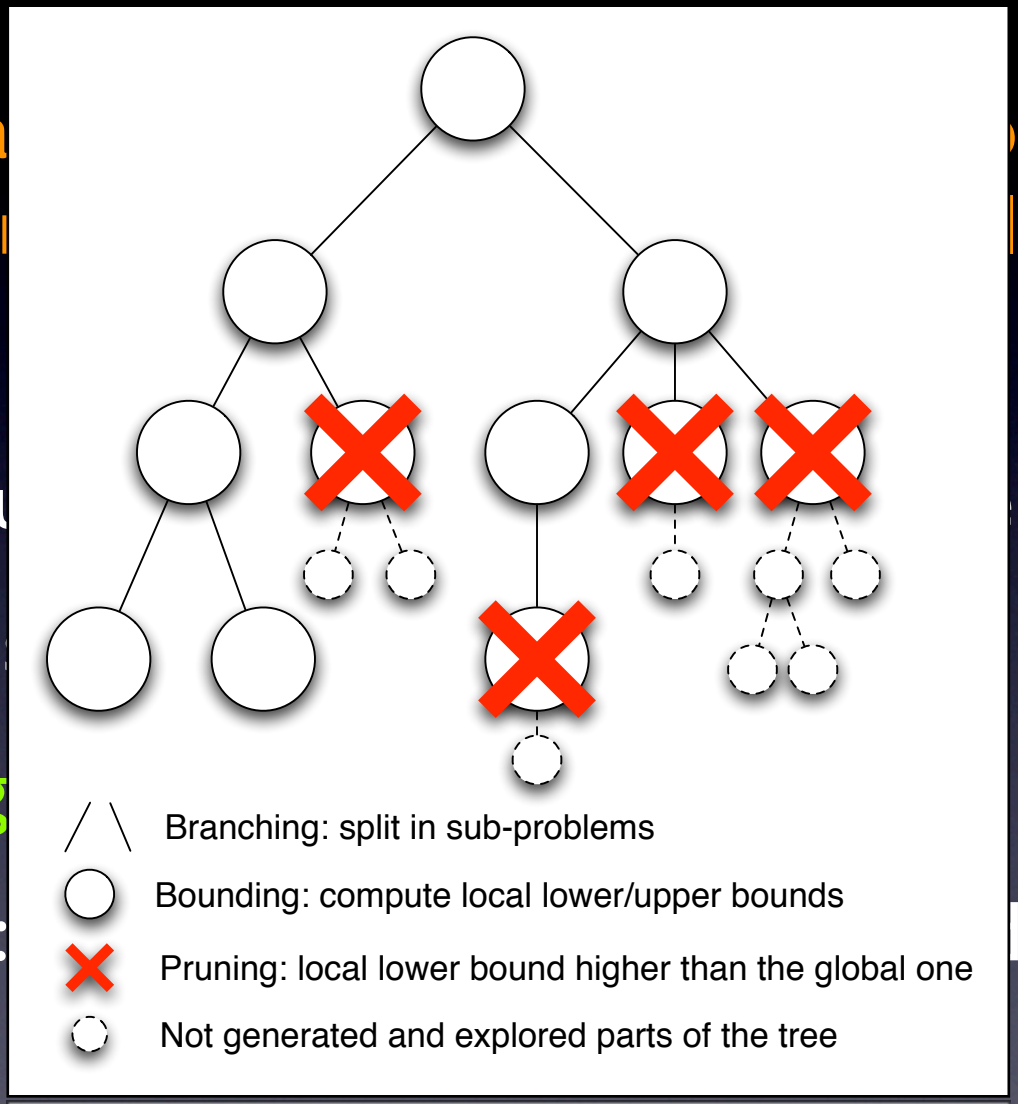
Consists of a partial enumeration of all feasible solutions and returns the guaranteed optimal solution

- Feasible solutions are organized as a tree: search-tree
- 3 operations:
  - Branching: split in sub-problems
  - Bounding: compute lower/upper bounds (objective function)
  - Pruning: eliminate bad branches

Branch and Bound

Consists of a pa…ble solutions and retur…lution

- Feasible solu…: search-tree
- 3 operation…
  - Branching…
  - Bounding:…ls (objective function)
  - Pruning: eliminate bad branches



Grid Computing

- Distributed shared computing infrastructure
  - multi-institutional virtual organization
- Provide large pool of resources
- Challenges
  - latency
  - deployment
  - scalability
  - communication
  - fault-tolerance
  - multiple administrative domains
  - heterogeneity
  - high performance
  - programming model
  - *etc.*

Parallel Branch and Bound

- COPs are difficult to solve
  - enumeration size & NP-hard class
- Many studies on parallel approach [Gendron 94, Crainic 06]
  - *node-based*: parallel bounding on sub-problems
  - *tree-based*: building the tree in parallel
  - *multi-search*: several trees are generated in parallel
- *Tree-based* is the most adapted to Grids
  - tree generated and explored in parallel
- Sharing global bounds for optimizing the prune operation

Parallel Branch and Bound

- COPs are difficult to solve
  - enumeration size & NP-hard class
- Ma
  -
  -
  -
- *Tree-based is the most adapted to Grids*
  - tree generated and explored in parallel
- Sharing global bounds for optimizing the prune operation

Proposition: Parallel B&B + Grid

Tree-based's Grid difficulties

- the solution tree is not known beforehand
- tasks are dynamically generated
- distributing issues (load-balancing & information sharing)

B&B on Grids: Problems and Solutions

Latency	Asynchronous communications
Scalability	Hierarchical master-worker [Aida 2003]
Solution tree size	Dynamically generated by splitting tasks
Share the best bounds	Efficient parallelism & communication
Faults	Fault-tolerance

B&B on Grids: Problems and Solutions

Latency	Asynchronous communications
S	ed
Share the best bounds	Efficient parallelism & communication
Faults	Fault-tolerance

Objective: hide Grid difficulties to users
Especially communication problems

Grid'BnB: B&B for Grids

- Context ProActive Java Grid middleware
  - latency ➟ asynchronous communication
  - underlying Grid infrastructure ➟ deployment framework (abstraction)

- Implement the tree-based parallel
- Master-worker architecture
  - Root Task
    - implemented by users
    - objective-function and splitting/branching operation
  - Master: Entry Point
    - splits the problem in tasks
    - collects partial-results ➟ the best solution
  - Sub-Master
    - intermediary between master and workers
  - Worker
    - computes tasks

Grid'BnB: B&B for Grids

- Context ProActive Java Grid middleware
  - latency ➟ asynchronous communication
  - underlying Grid infrastructure ➟ deployment framework (abstraction)
- Impl
- Mas
  - R
    -
    -
  - Master: Entry Point
    - splits the problem in tasks
    - collects partial-results ➟ the best solution
  - Sub-Master
    - intermediary between master and workers
  - Worker
    - computes tasks

Problem: workers need to share bounds

➟ Efficiently share bounds

Organizing Communication

- Solution 1: Master keeps the bound
  - previous work shows that it doesn't scale [Aida 2003]
- Solution 2: Message framework (Enterprise Service Bus)
  - Grid middleware dependent / Good for SOA
- Solution 3: Broadcasting
  - 1 to n communication cannot scale
  - ✓ hierarchical broadcasting scale [Baduel 05]

- clusters are high-performance communication environments

Idea: Grids are composed of clusters ➟ organizing Workers in groups

Workers are allocated to groups in regards of their localization

For hierarchical communications between groups

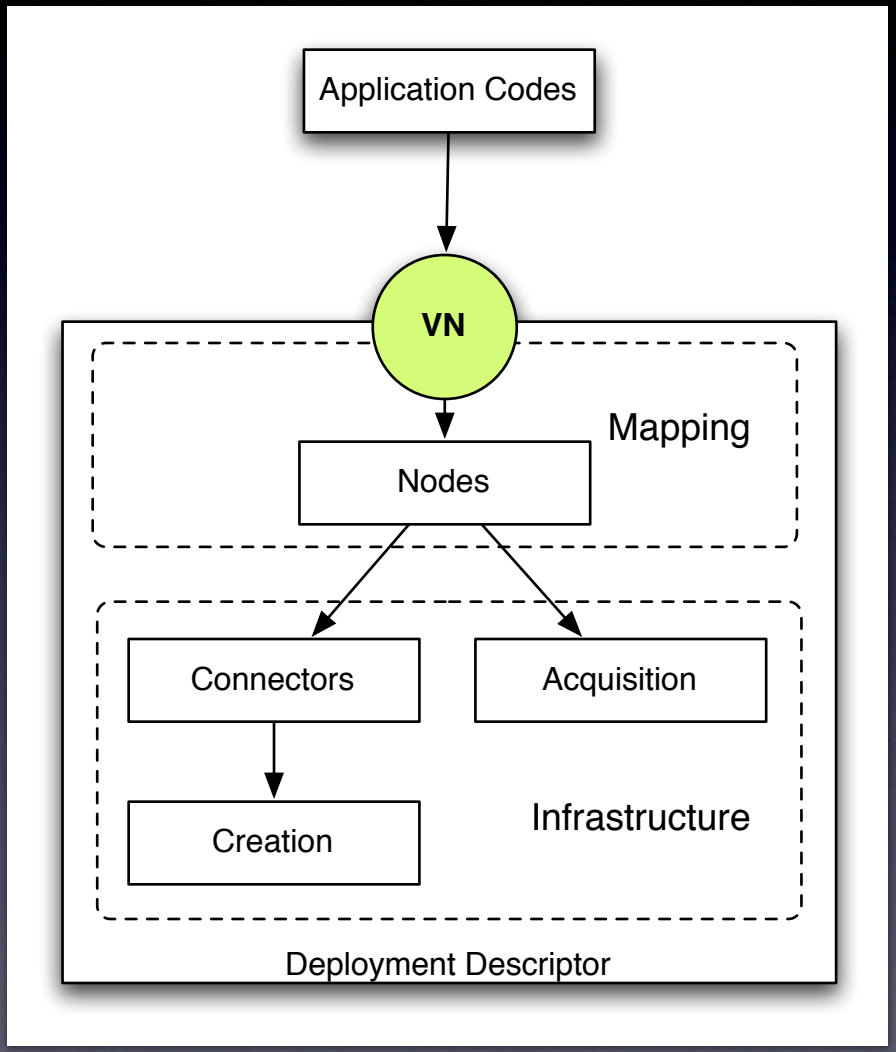
Problem: ProActive deployment is an abstraction

Grid Node Localization

- ProActive Deployment is an abstraction of the physical infrastructure

Grid Node Localization

- ProActive Deployment is an abstraction of the physical infrastructure



Grid Node Localization

- ProActive Deployment is an abstraction of the physical infrastructure

Grid Node Localization

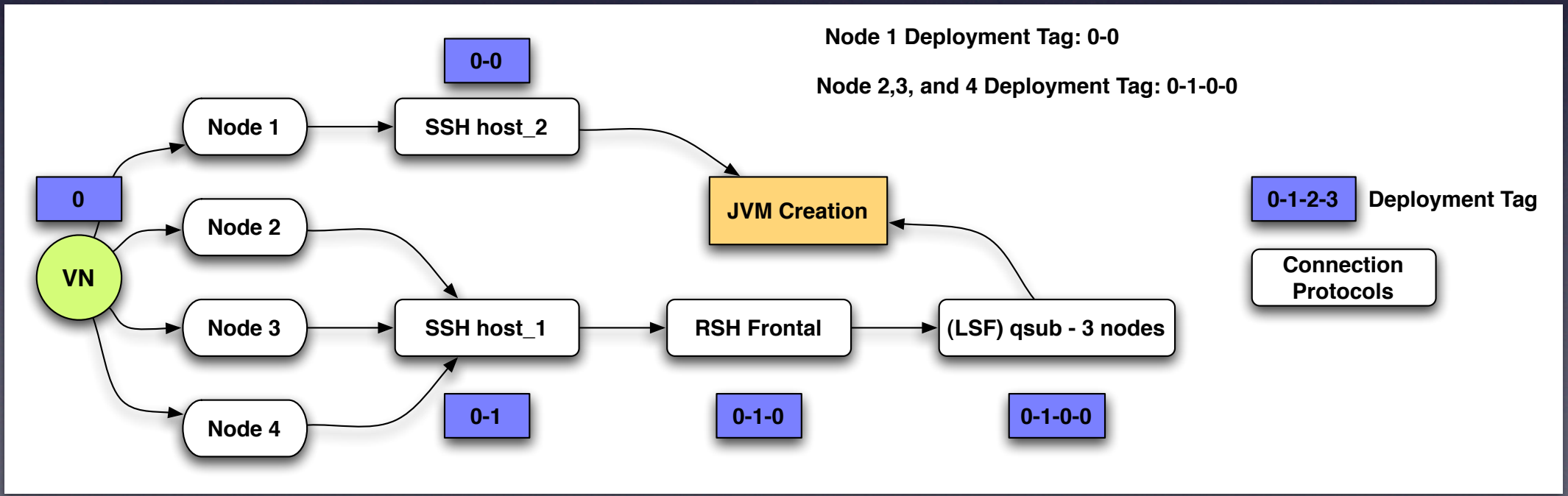
- ProActive Deployment is an abstraction of the physical infrastructure

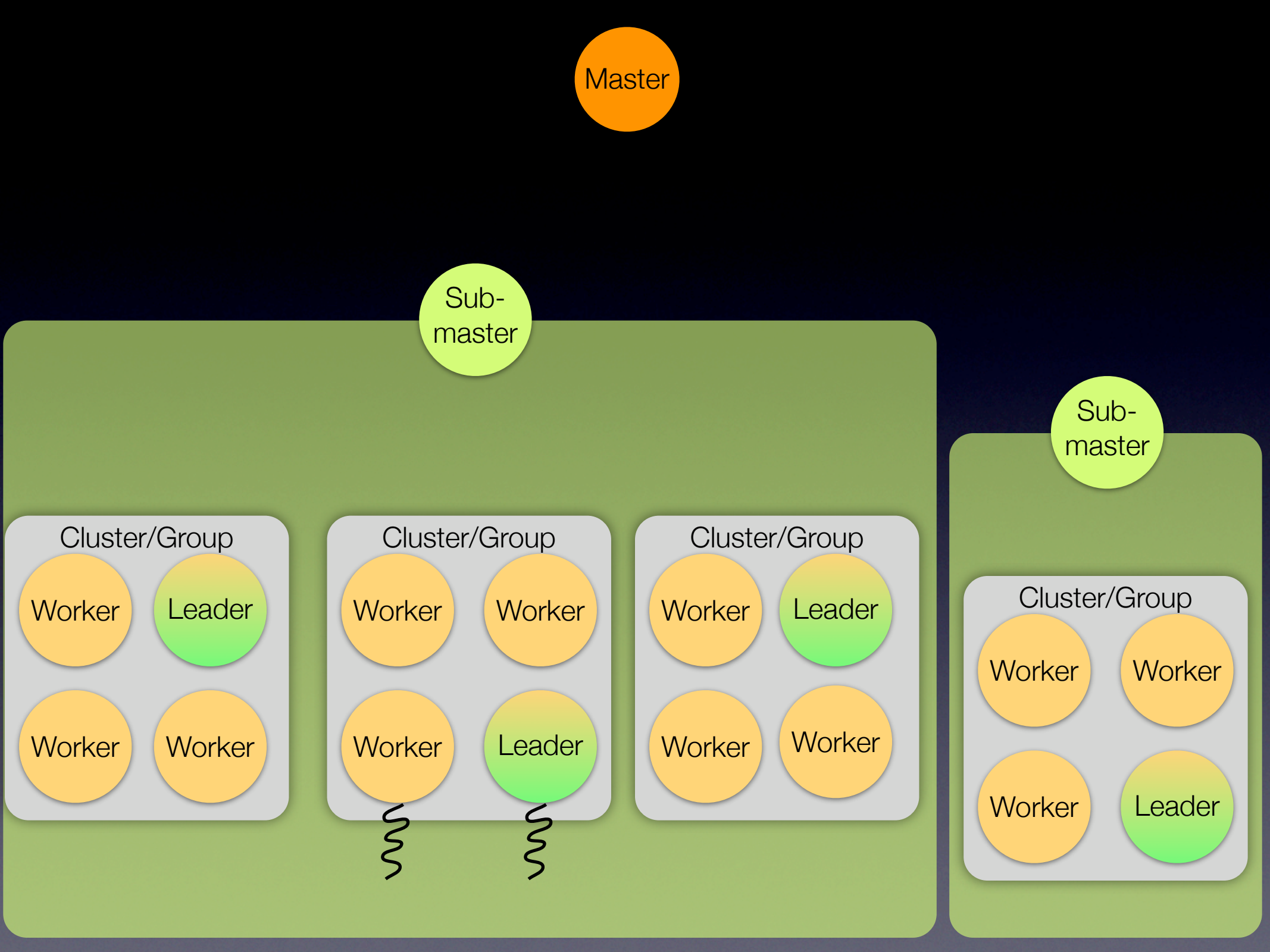
- Deployment is a DAG specified within an XML file
- Tagging each deployment level

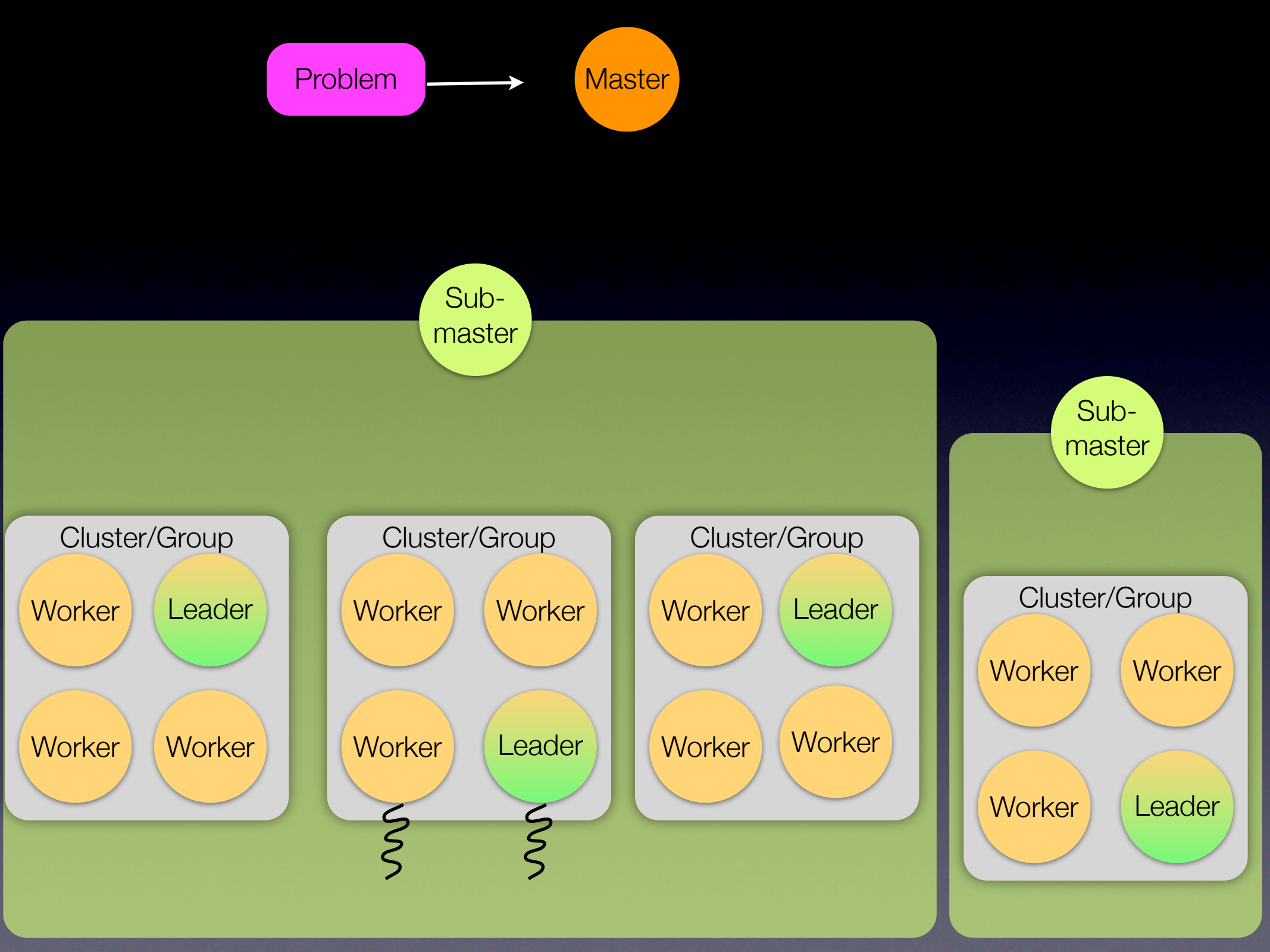
Grid Node Localization

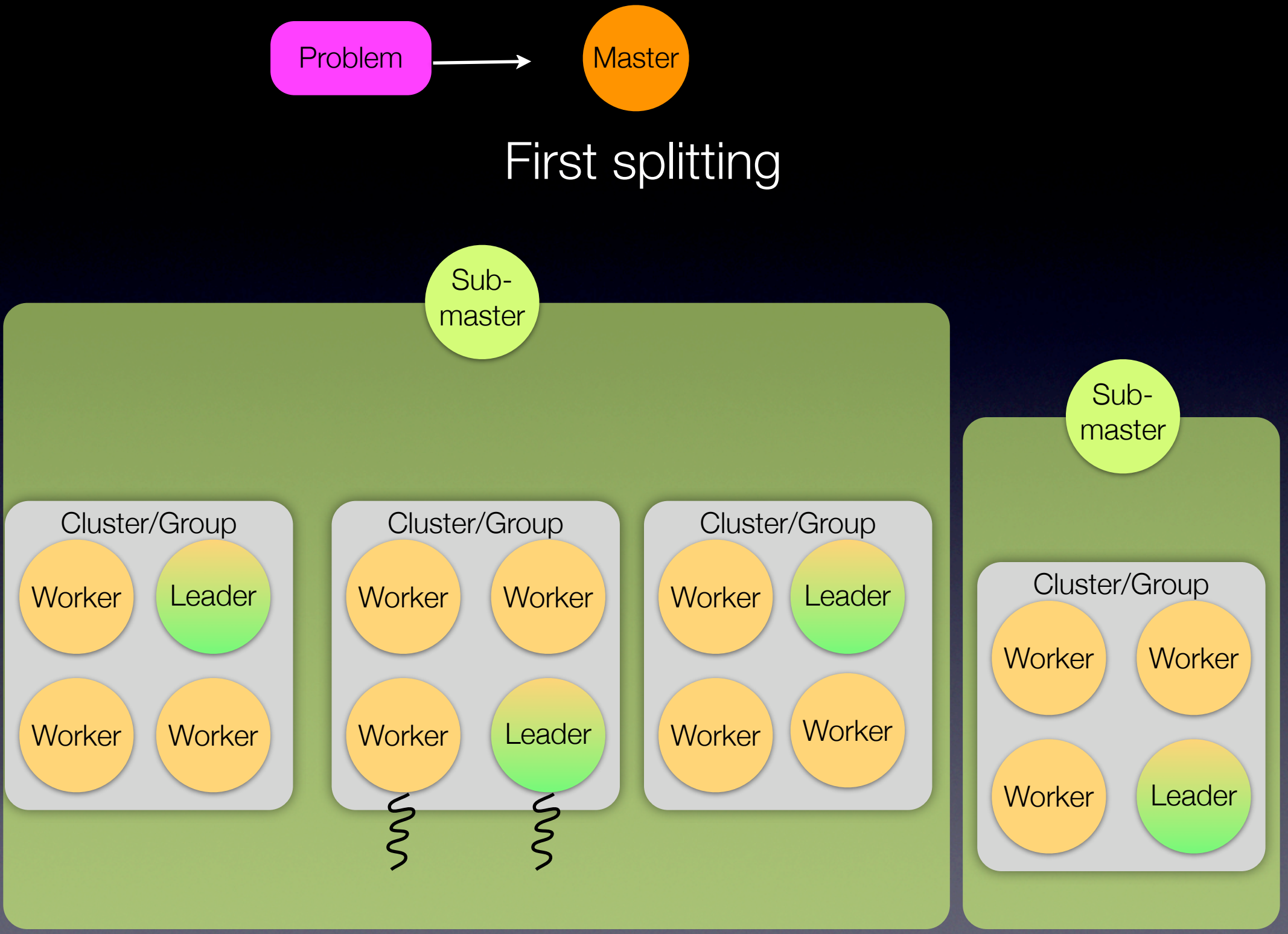
- ProActive Deployment is an abstraction of the physical infrastructure

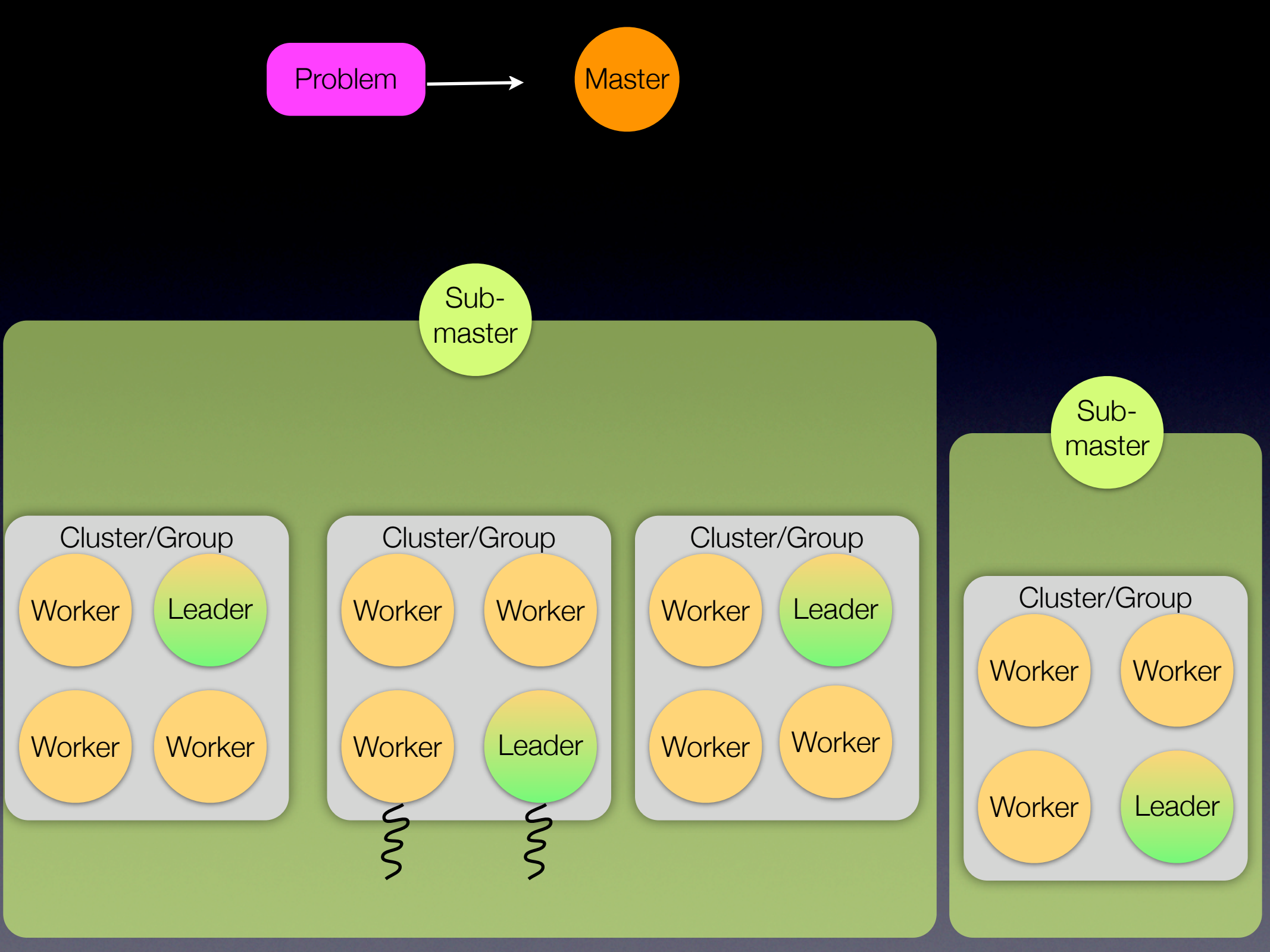
- Deployment is a DAG specified within an XML file
- Tagging each deployment level

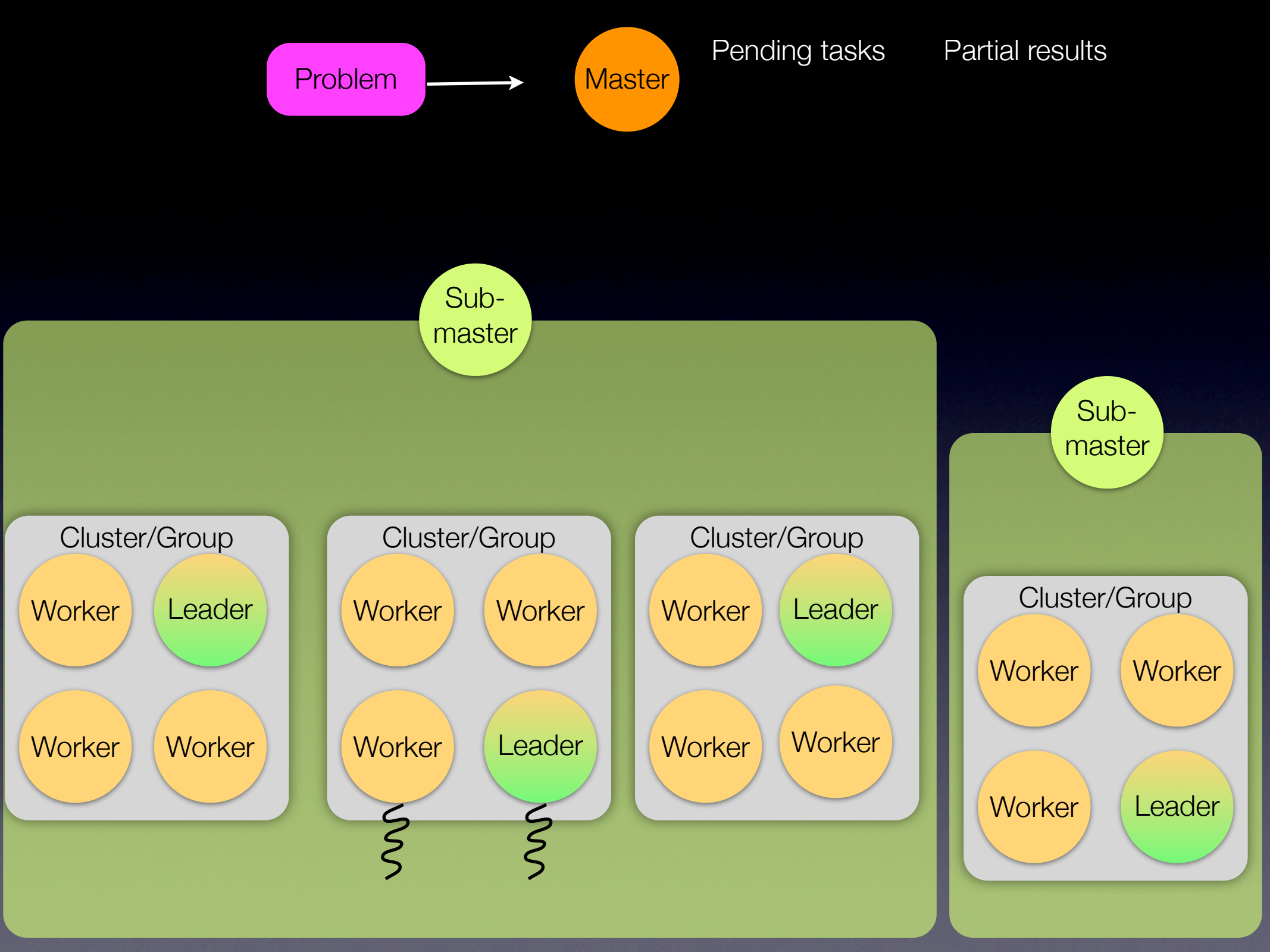


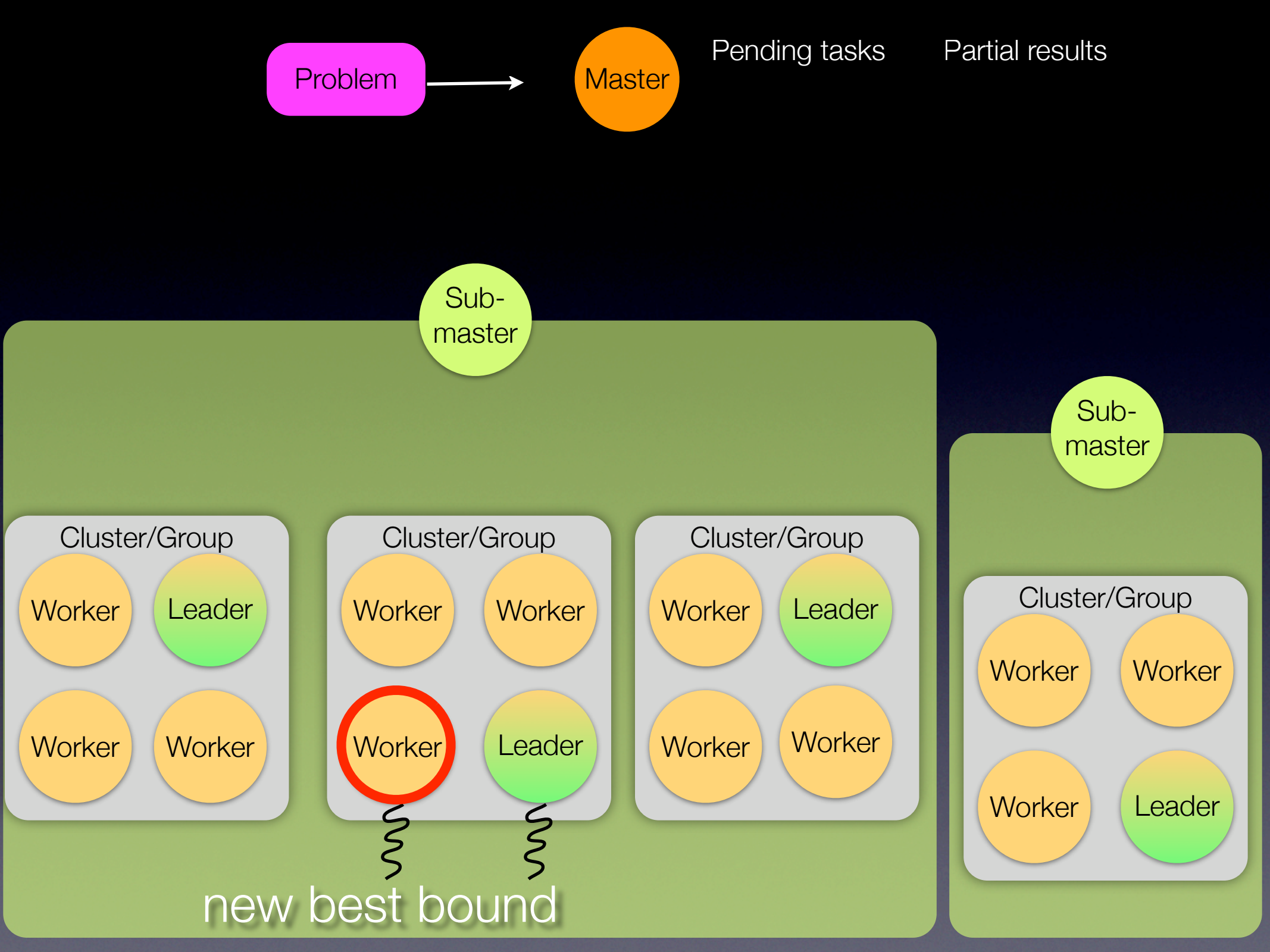

Master
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader

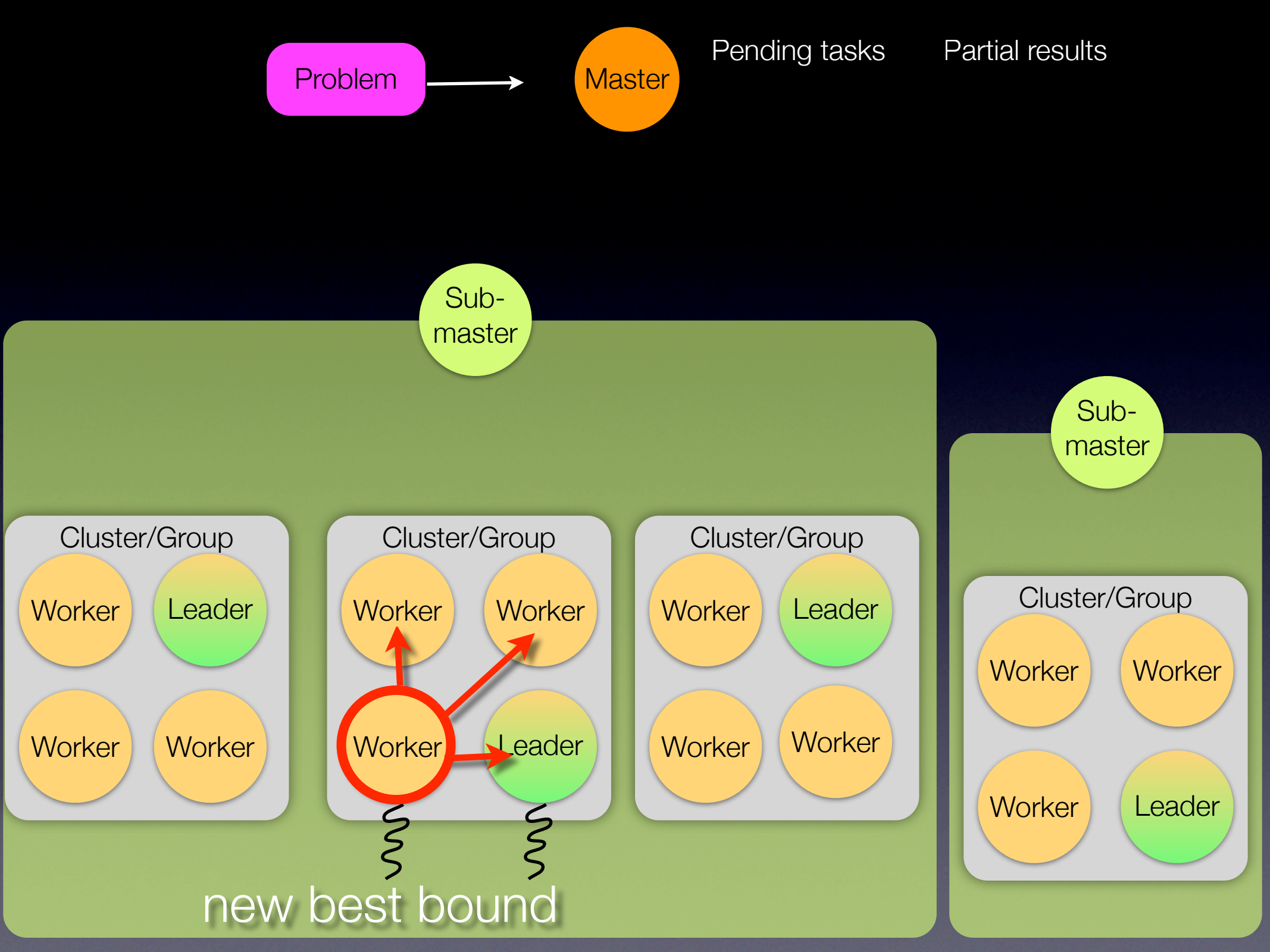

Problem
Master
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader

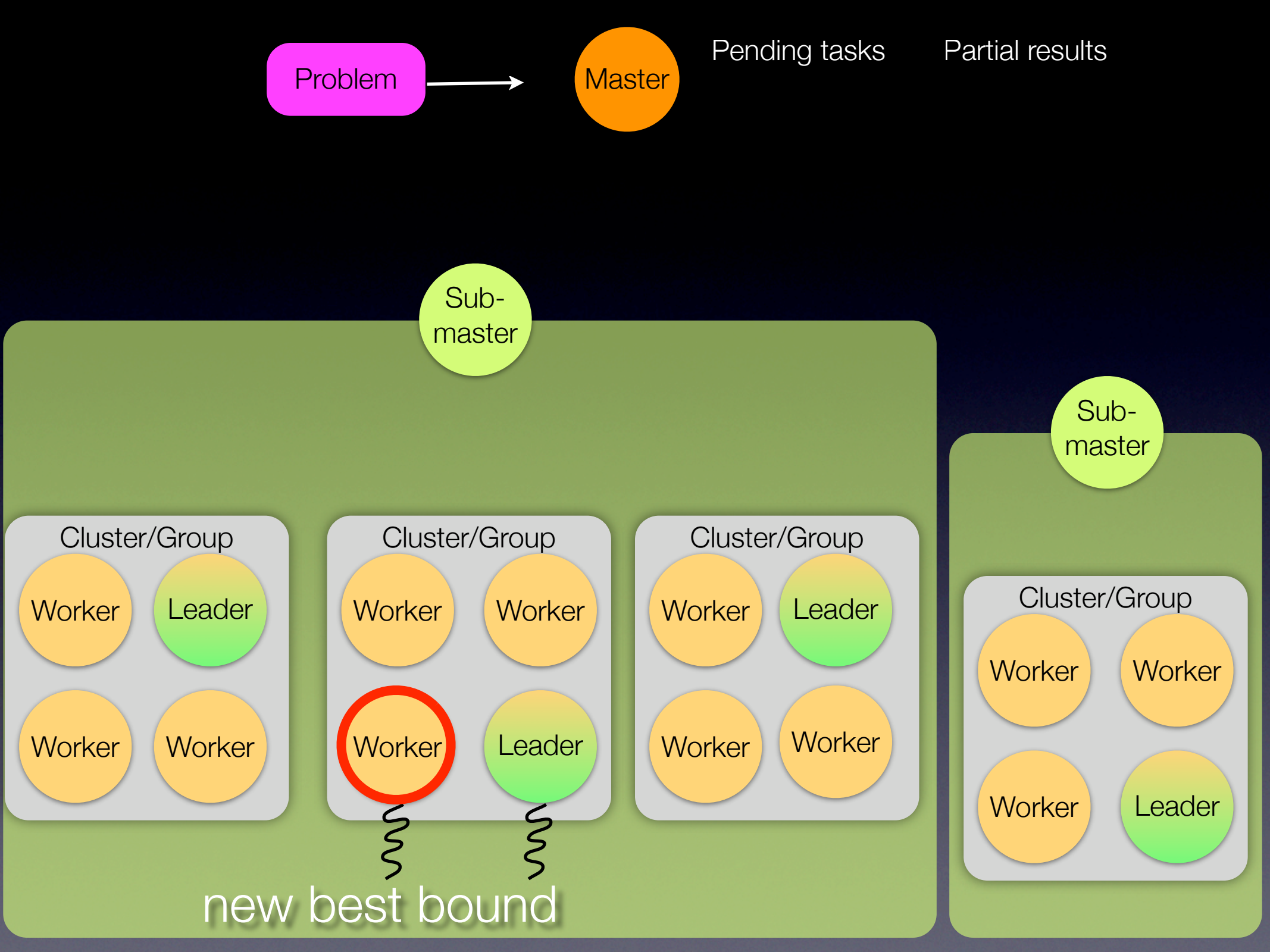

Problem
Master
First splitting
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader

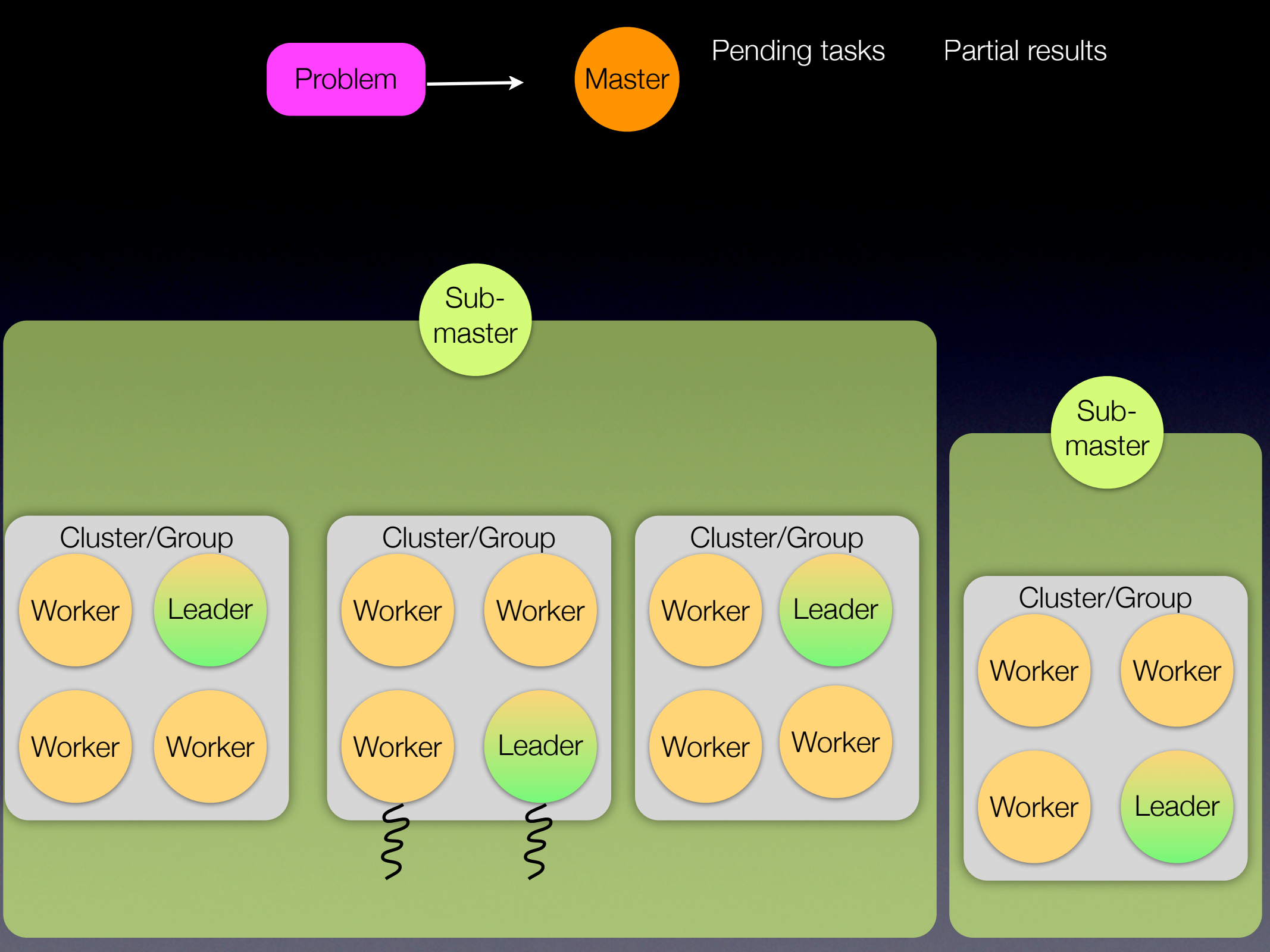

Problem
Master
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader

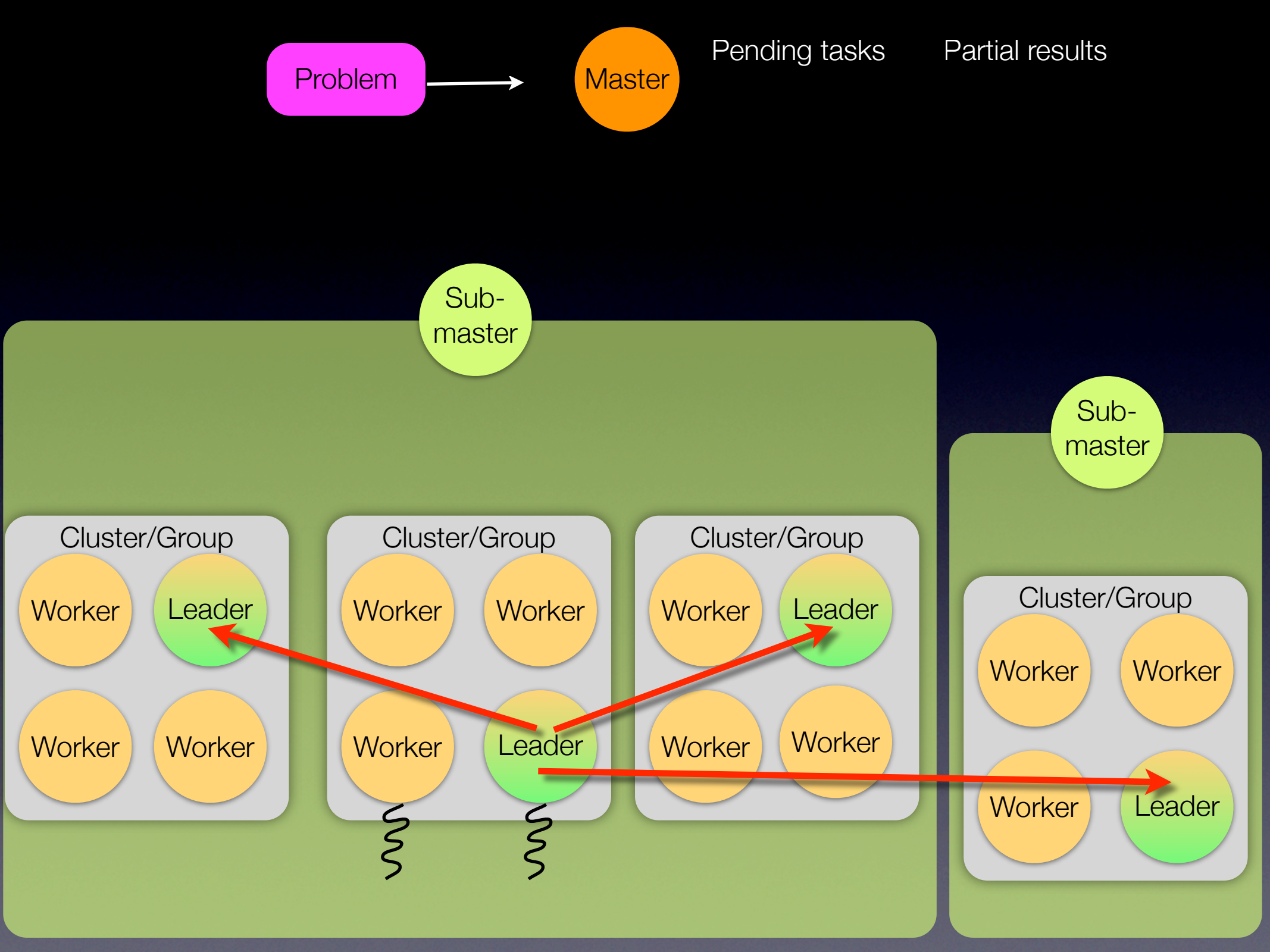

Problem
Master
Pending tasks
Partial results
Sub-master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-master
Cluster/Group
Worker
Worker
Worker
Leader

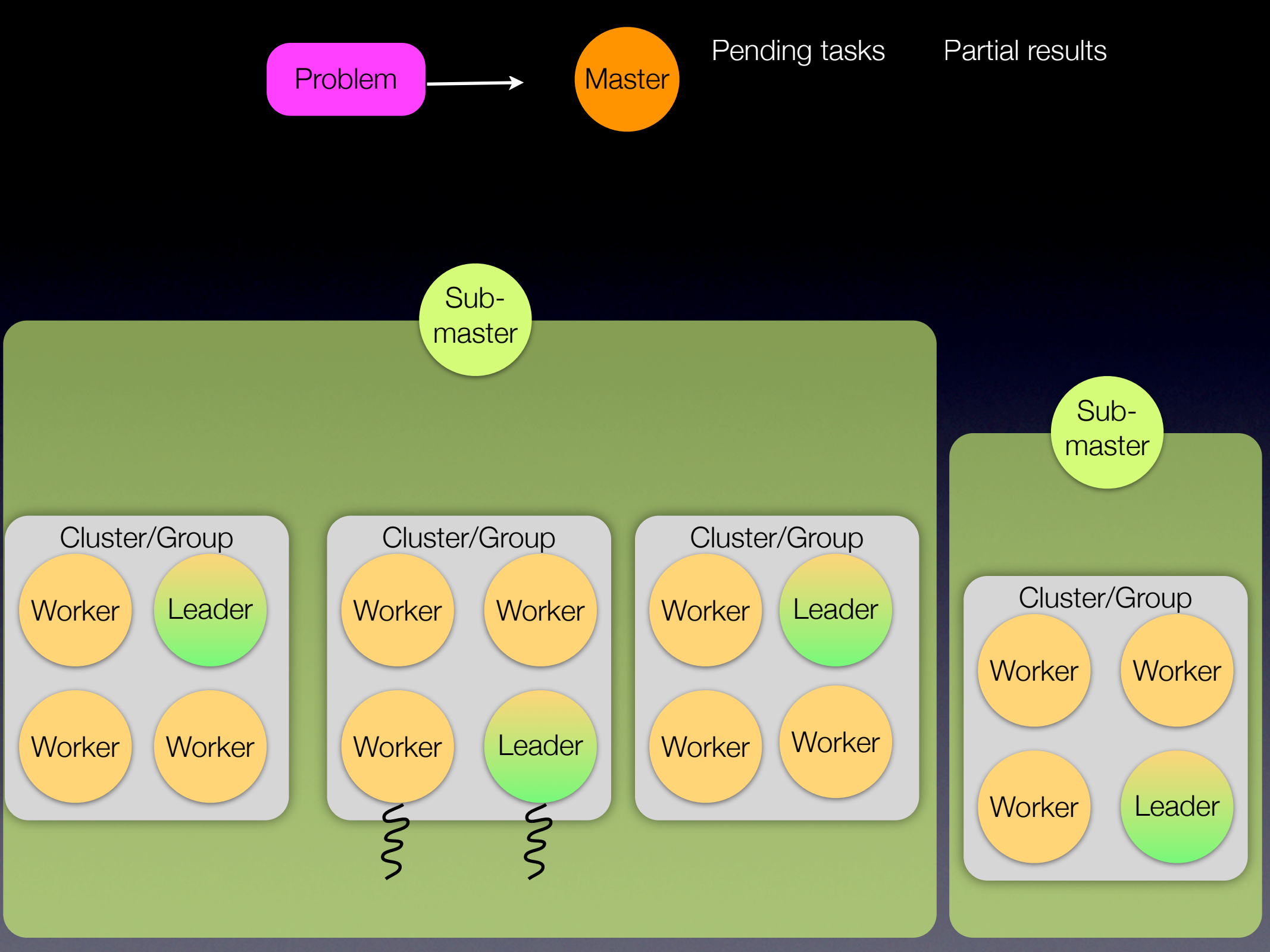

Problem
Master
Pending tasks
Partial results
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader
new best bound

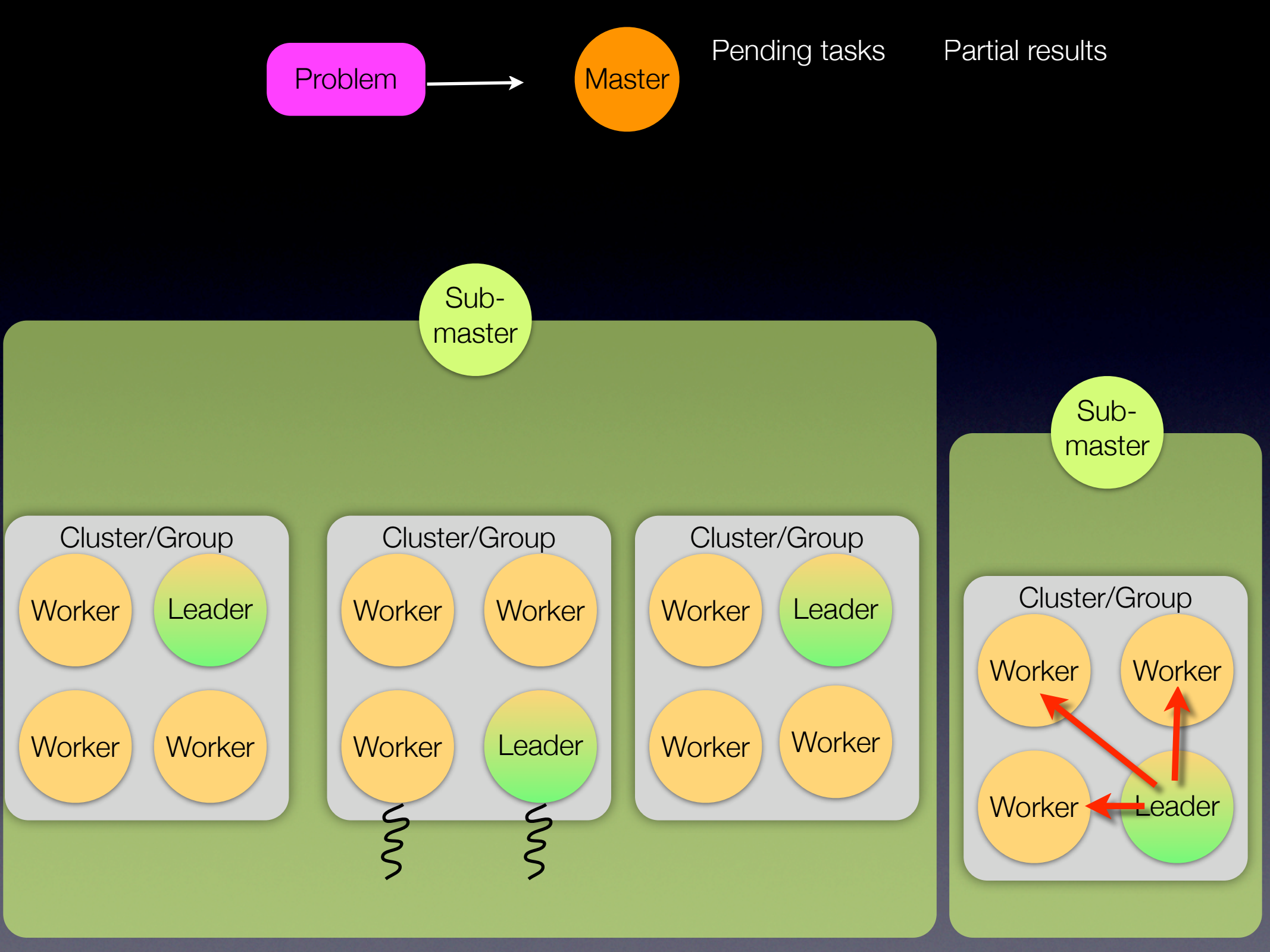

Problem
Master
Pending tasks
Partial results
Sub-master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-master
Cluster/Group
Worker
Worker
Worker
Leader
new best bound

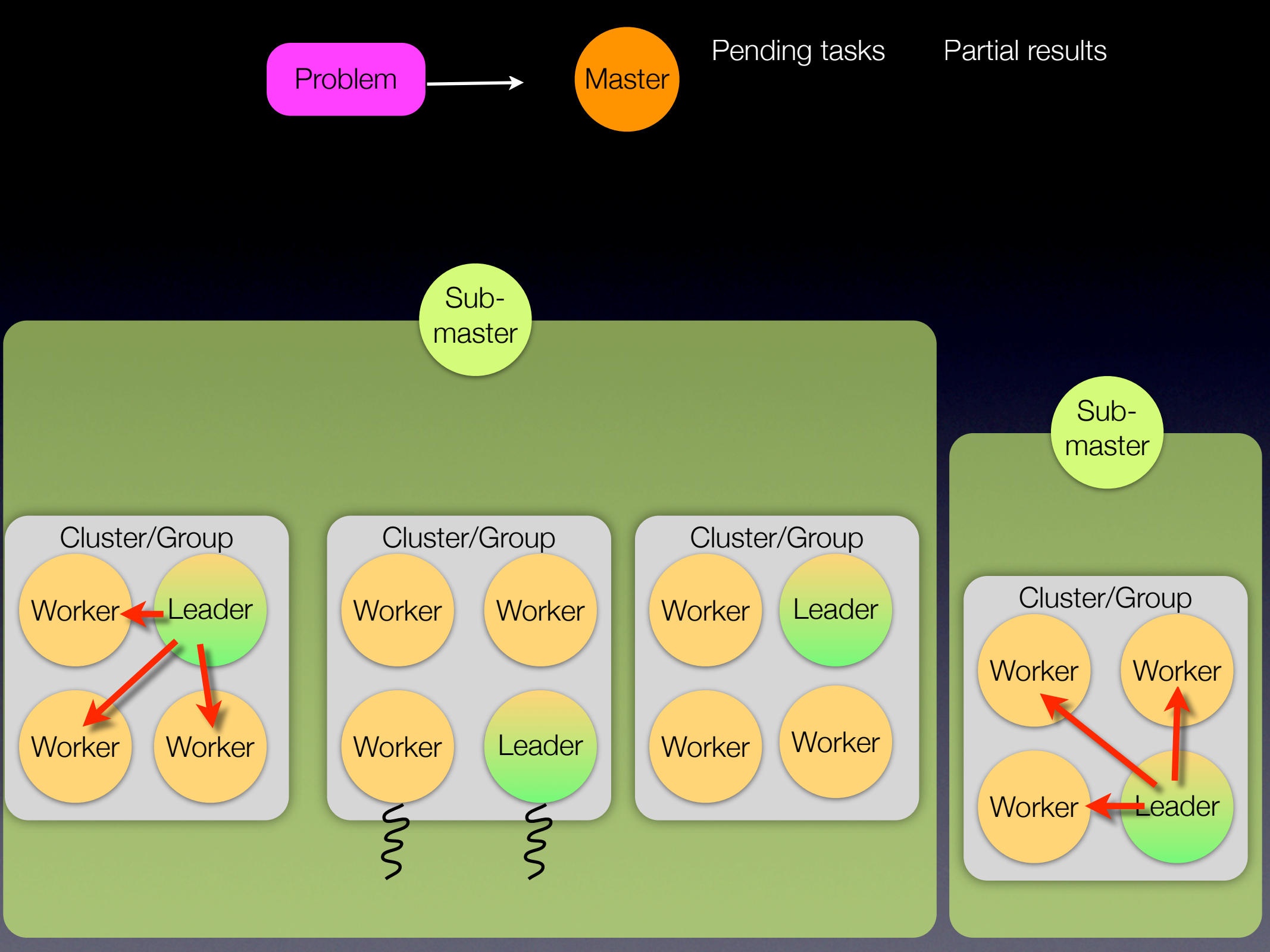

Problem
Master
Pending tasks
Partial results
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader
new best bound

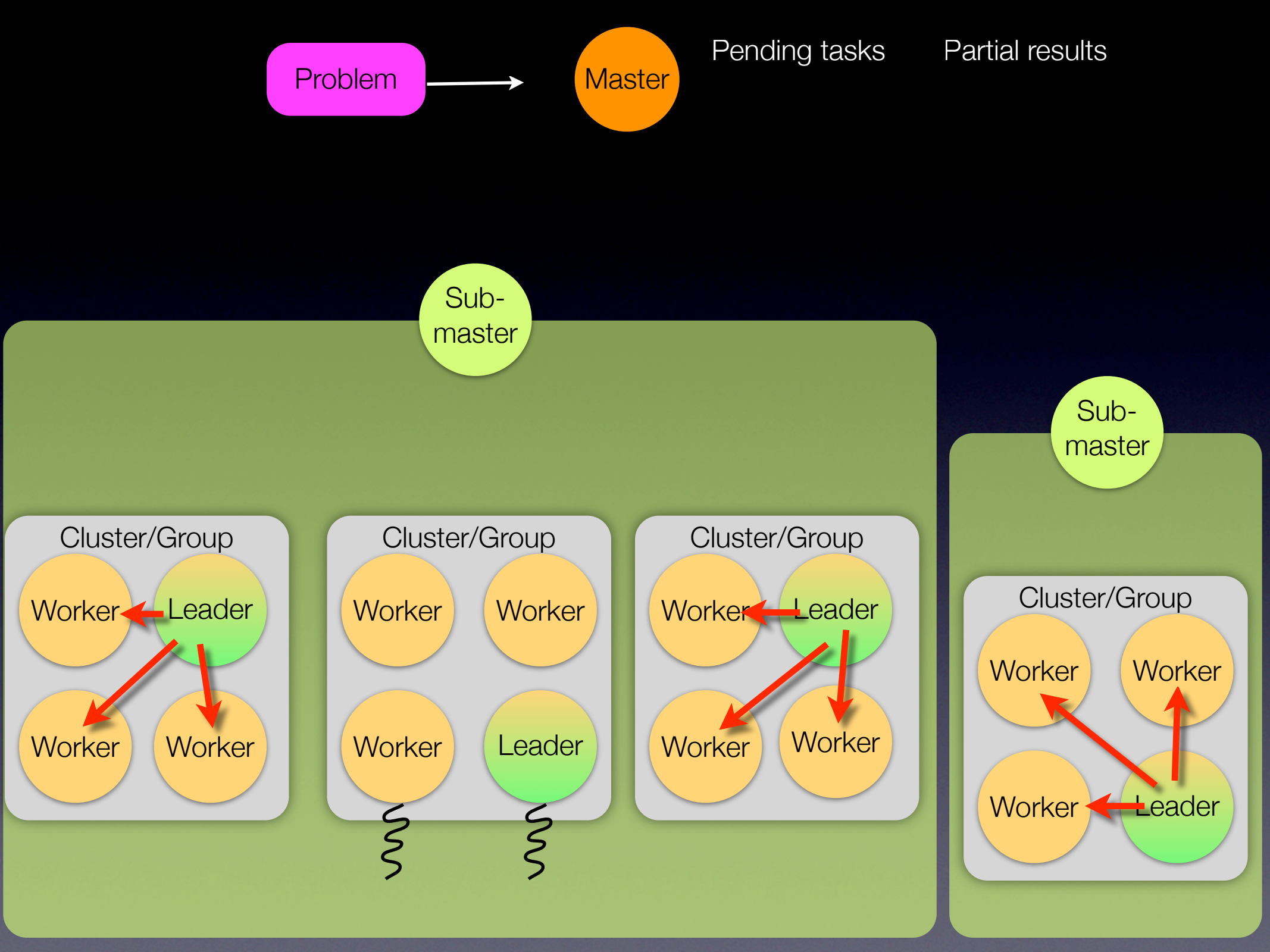

Problem
Master
Pending tasks
Partial results
Sub-master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-master
Cluster/Group
Worker
Worker
Worker
Leader


Problem
Master
Pending tasks
Partial results
Sub-master
Sub-master
Cluster/Group
Cluster/Group
Cluster/Group
Cluster/Group
Worker
Leader
Worker
Worker
Worker
Worker
Worker
Leader
Worker
Worker
Worker
Leader
Worker
Worker
Worker
Leader


Problem
Master
Pending tasks
Partial results
Sub-master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-master
Cluster/Group
Worker
Worker
Worker
Leader


Problem
Master
Pending tasks
Partial results
Sub-master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-master
Cluster/Group
Worker
Worker
Worker
Leader


Problem
Master
Pending tasks
Partial results
Sub-
master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-
master
Cluster/Group
Worker
Worker
Worker
Leader


Problem
Master
Pending tasks
Partial results
Sub-master
Cluster/Group
Worker
Leader
Worker
Worker
Cluster/Group
Worker
Worker
Worker
Leader
Cluster/Group
Worker
Leader
Worker
Worker
Sub-master
Cluster/Group
Worker
Worker
Worker
Leader

Grid'BnB: Search Strategies

- The improvement of the bounds
  1. depends on how it is shared (communication)
  2. depends on how the search-tree is generated
     - Classical
       - Depth-First Search
       - Breadth-First Search
     - Contribution
       - First-In-First-Out (FIFO)
       - Priority
       - open API ...

Grid'BnB Features

Design

- Asynchronous communications
- Hierarchical master-worker with communications
- Dynamic task splitting
- Efficient communications with groups
- Fault-tolerance

Users

- Hidden parallelism and Grid difficulties
- API for COPs
- Ease of deployment
- Principally tree-based
- Implementing and testing search strategies
- Focus on objective function

Grid'BnB Features

- Asynchronous communications
- Hierarchical master-worker with communications
- Dynamic task splitting

Design

Validate and Test Grid'BnB by experiments

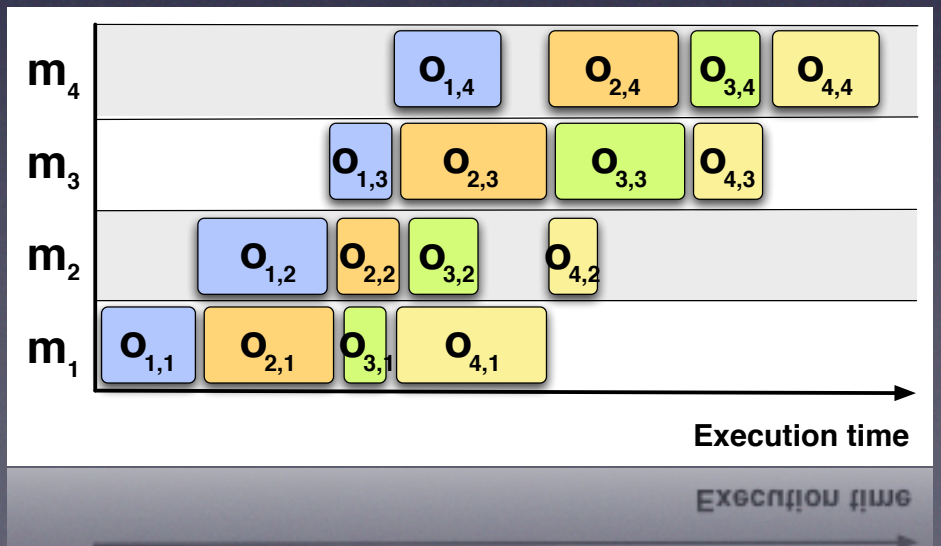
- API for COPs
- Ease of deployment
- Principally tree-based
- Implementing and testing search strategies
- Focus on objective function

Users

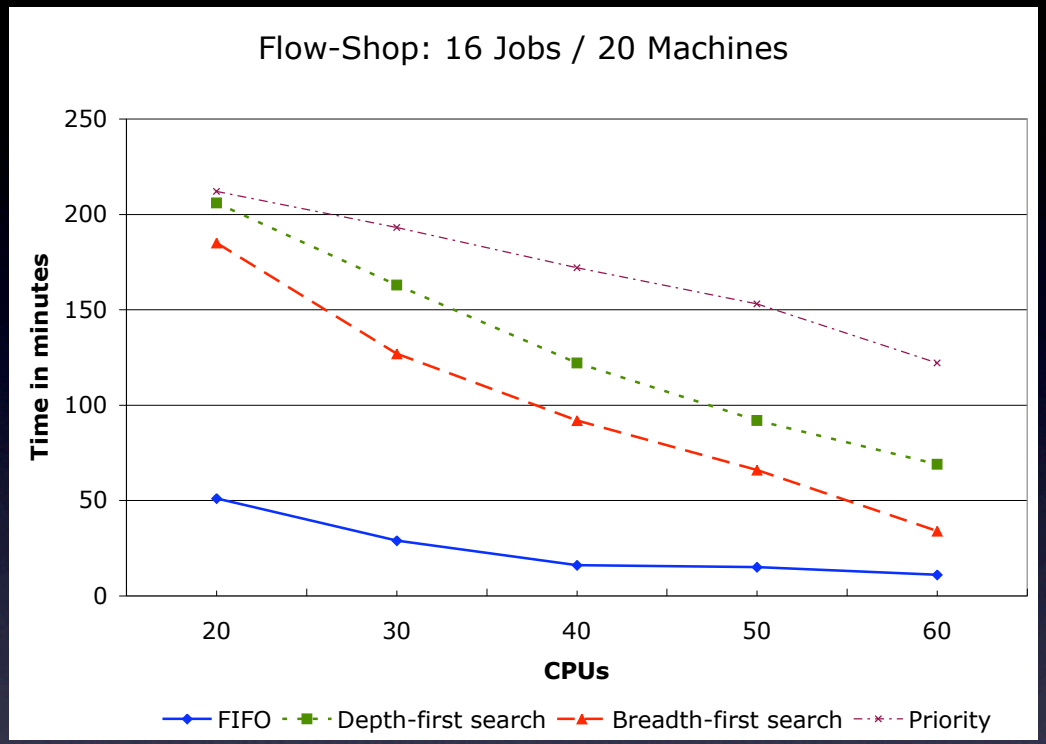
Flow-Shop Experiments

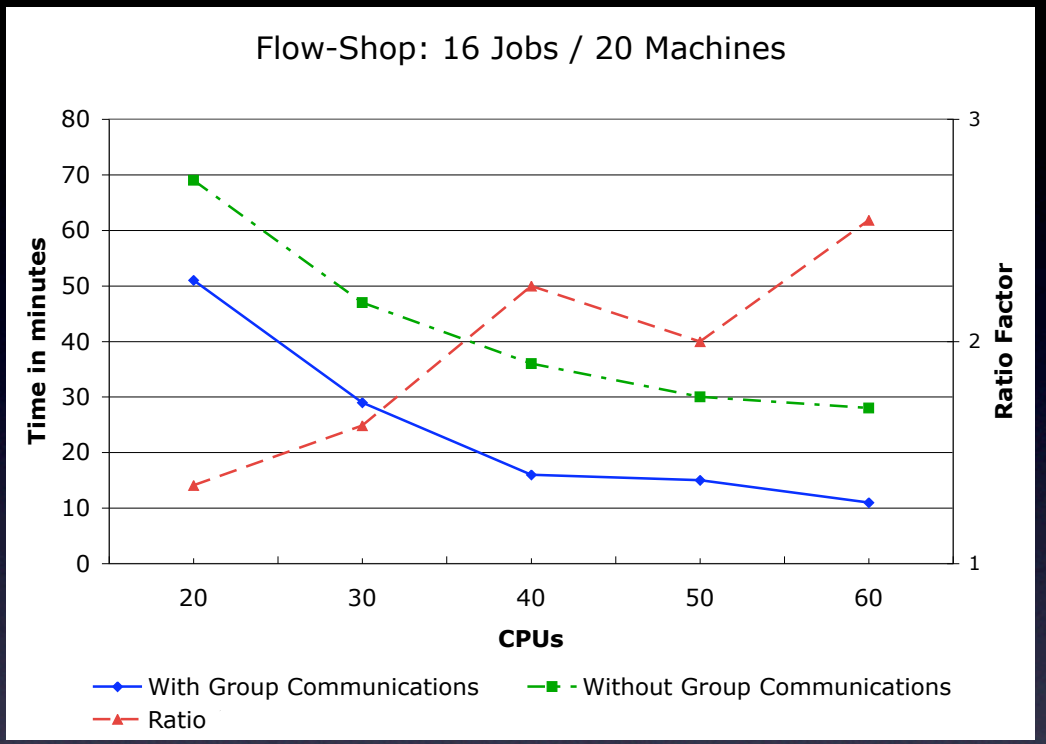
- NP-hard permutation optimization problem
- The flow-shop problem consists in finding the optimal schedule of n jobs on m machines.

The set of jobs: $J = \{j_1, j_2, \ldots j_n\}$
Each j_i is a set of operations: $j_i = \{ o_{i1}, o_{i2}, \ldots o_{im} \}$
The set of machines: $M = \{m_1, m_2, \ldots m_m\}$



Single Cluster Experiments



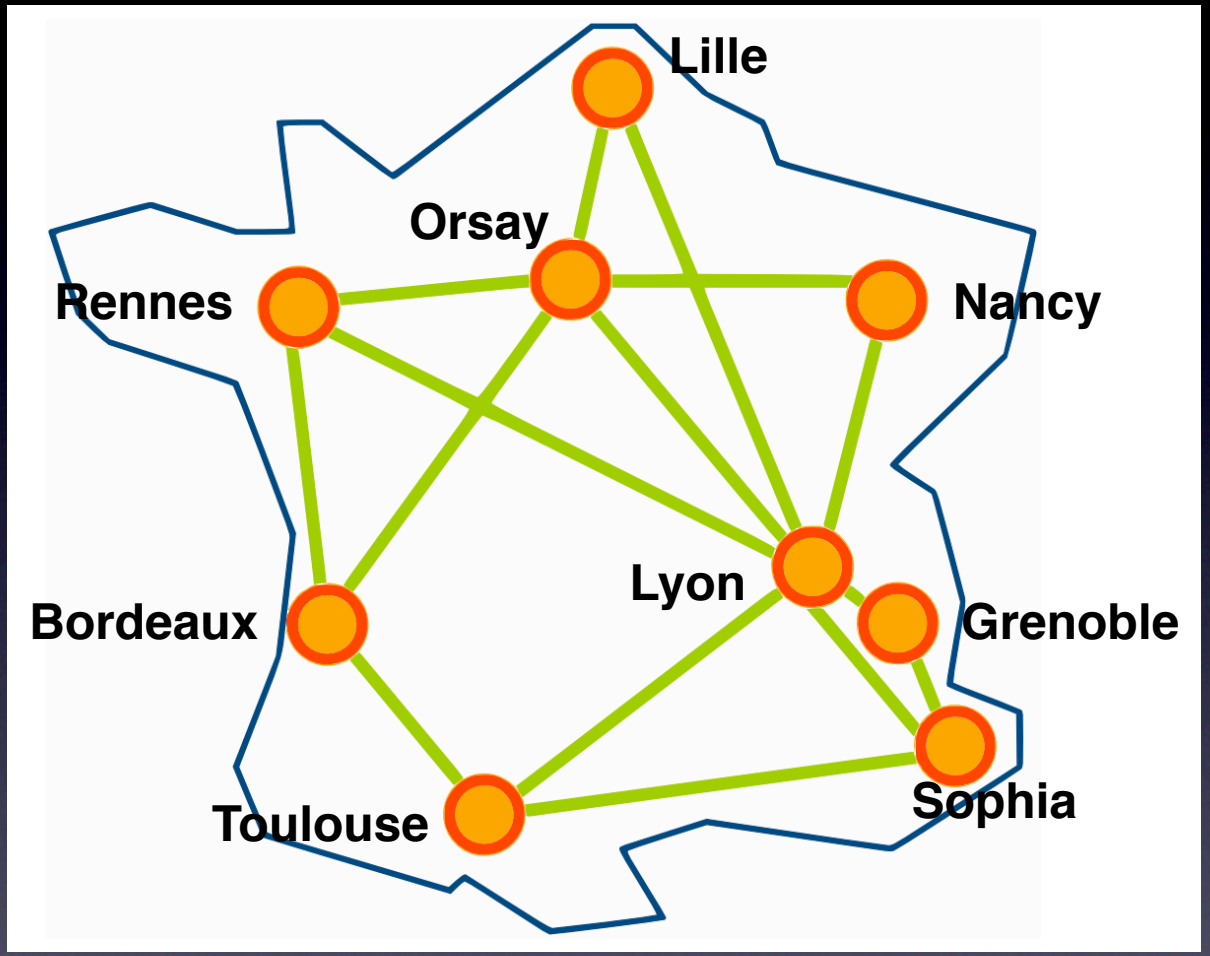


Search Strategies

Communication

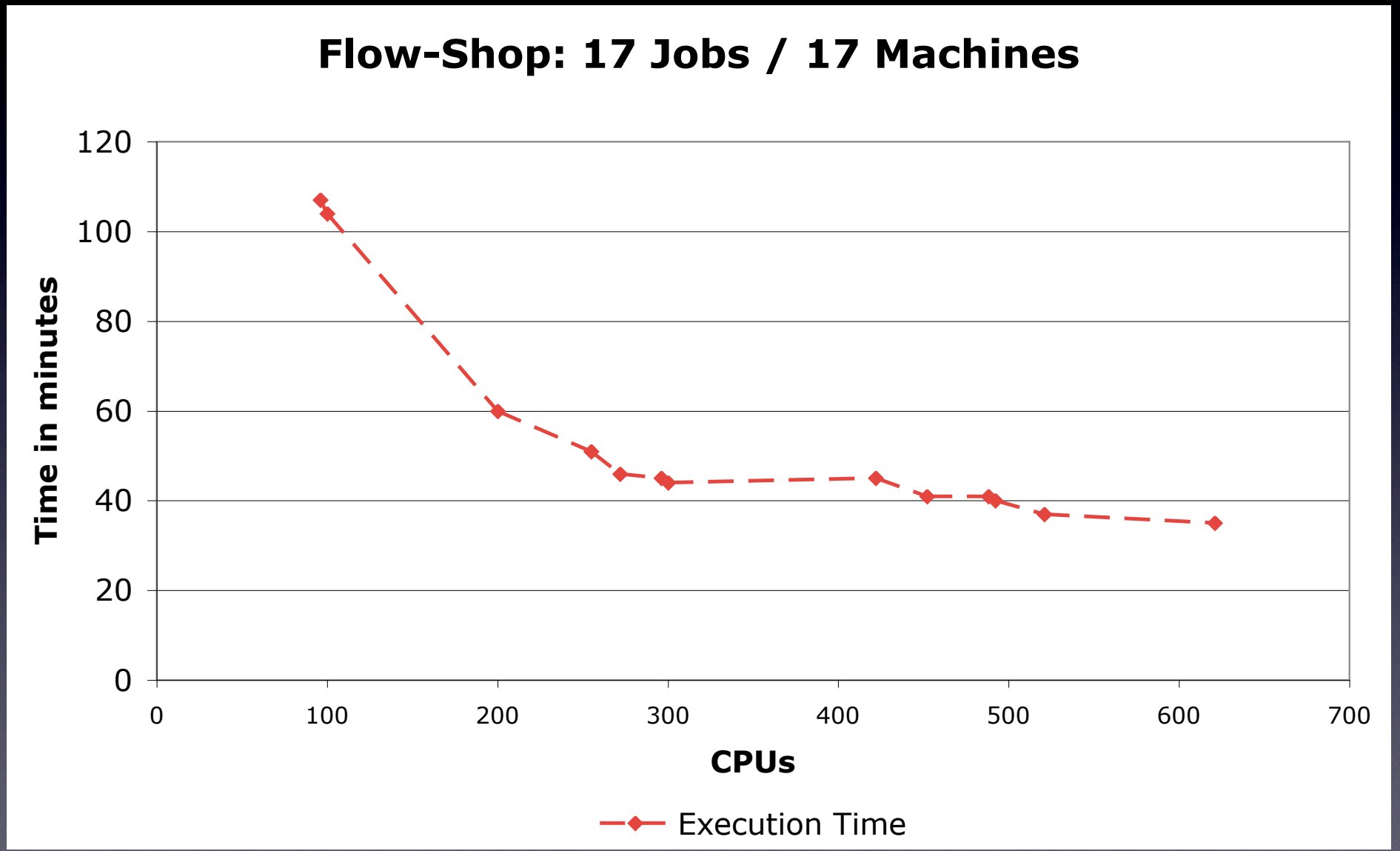
- Best execution time with FIFO search strategy
- Ratio No Communication / Communication ≈ 2.5

Grid'5000: the French Grid

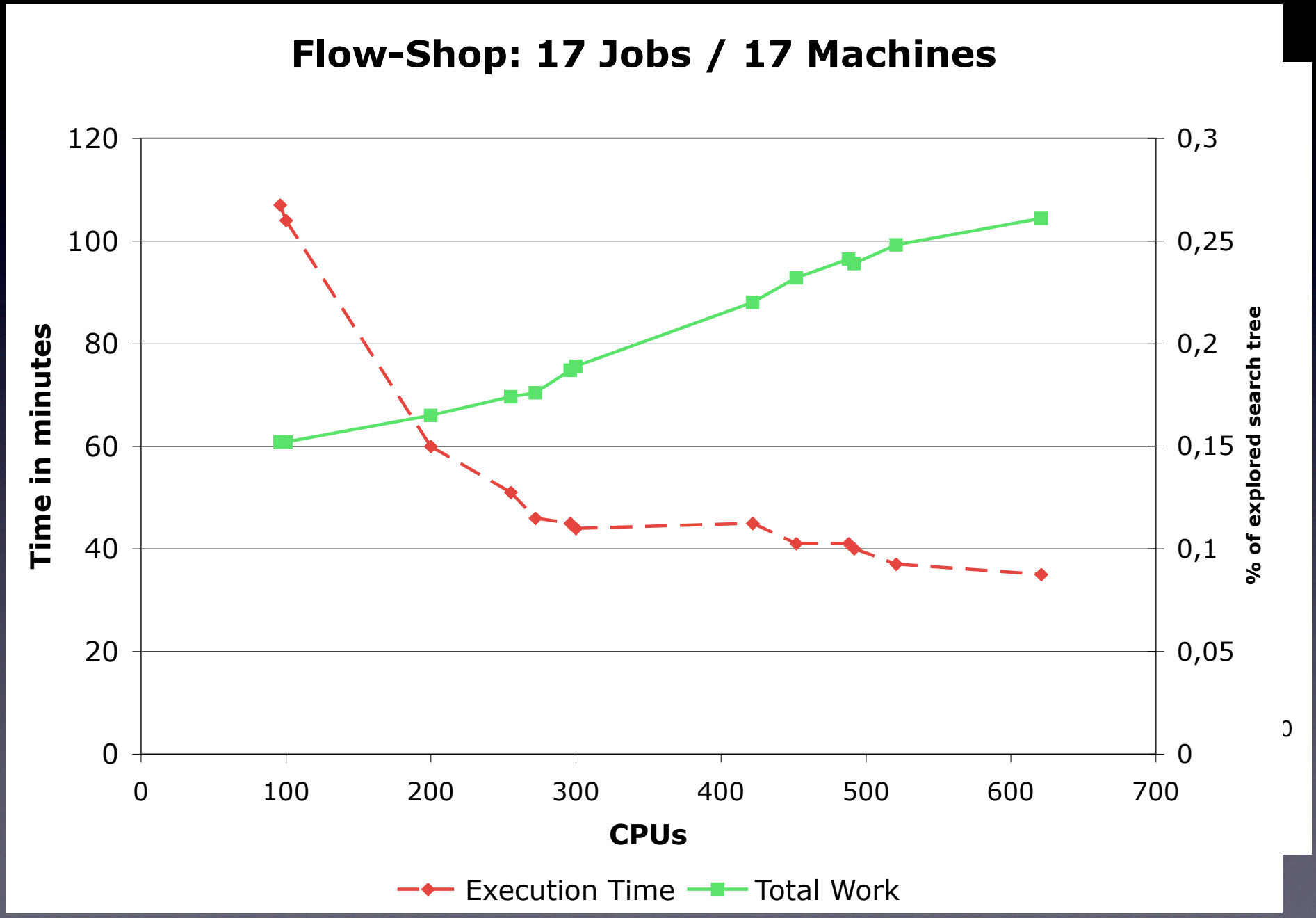


9 sites - 13 clusters - 4422 cores

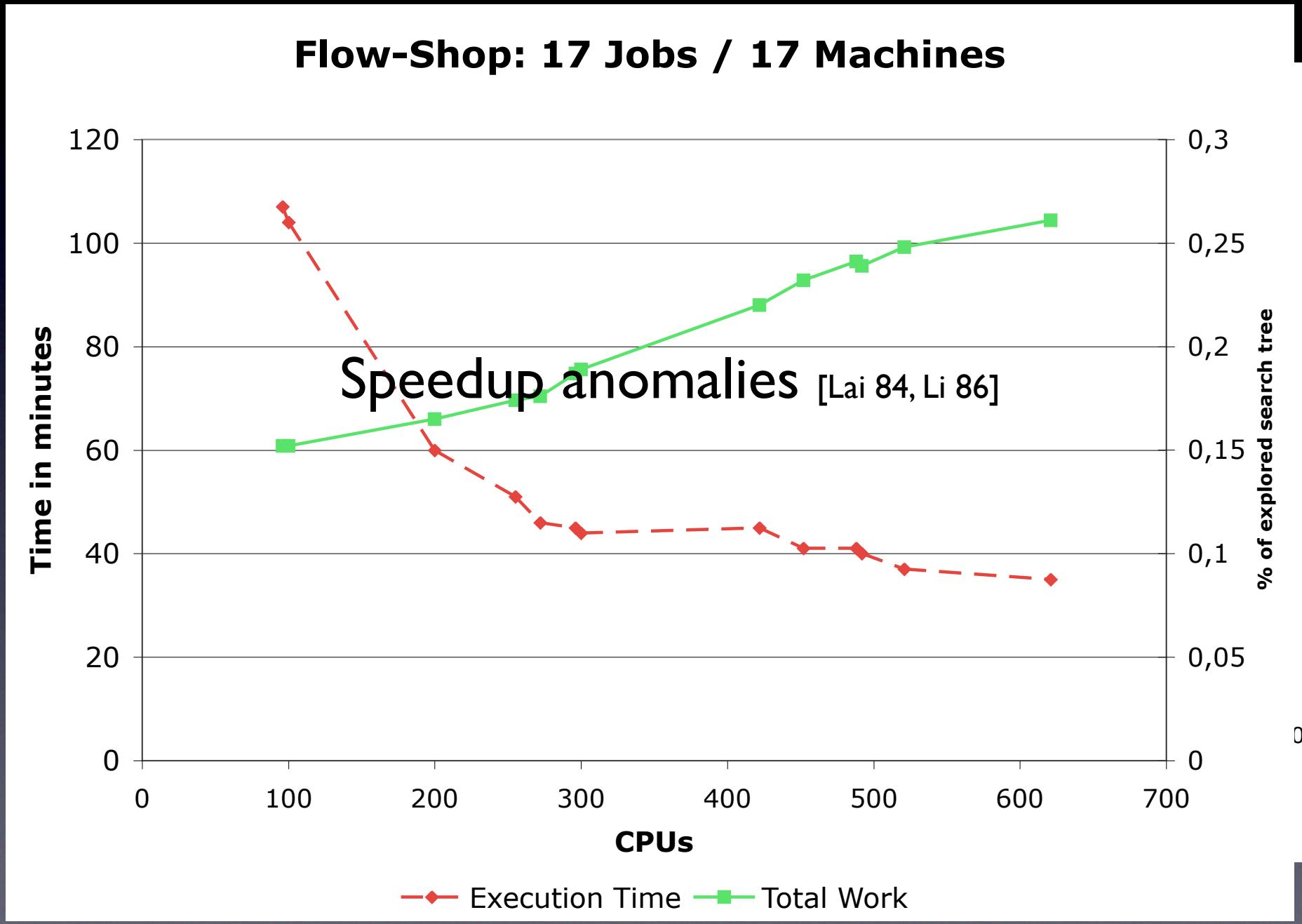
Grid Experimentations



Grid Experimentations



Grid Experimentations

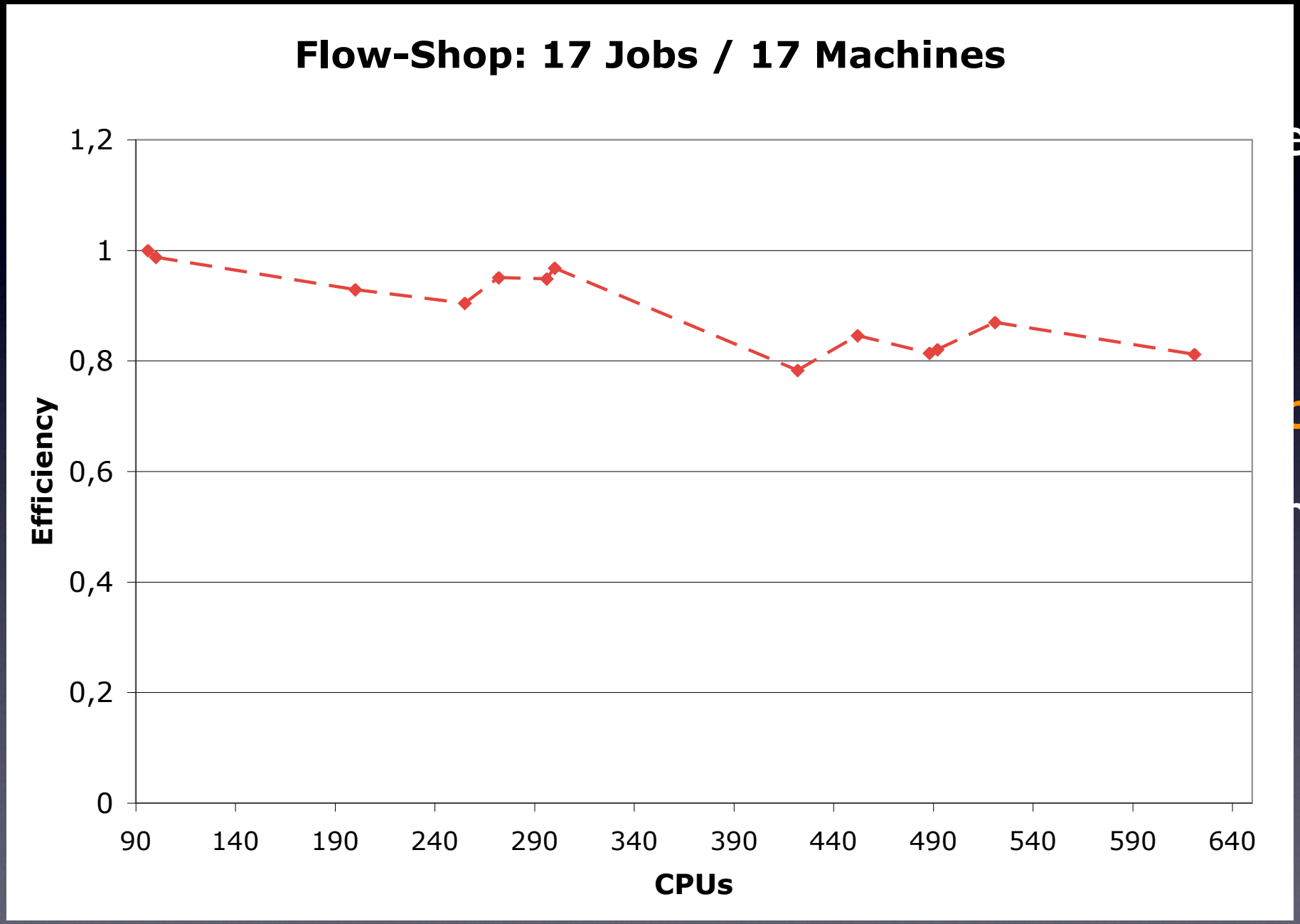


Speedup anomalies [Lai 84, Li 86]

Speedup Anomalies & Efficiency

- Parallel tree-based speedup may be sometimes quite spectacular (> or < linear) [Mans 95]
- Speedup Anomalies in BnB [Roucairol 87, Lai 84, Li 86]
  - speedup depends on how the tree is dynamically built
- Efficiency (E) estimates how CPUs are utilized for the computation
  - $0 \leq E \leq 1$
  - $E = 1$ is equivalent to linear-speedup

Speedup Anomalies & Efficiency



Conclusion & Perspectives

- Branch and Bound for Grids
  - solving optimization problems
  - hiding Parallel & Grid difficulties
  - using communication between workers
- Experimentally validate *Grid'BnB*
  - validity of organizing communications
  - scalability on Grid (up to 621 CPUs on 5 sites)
- Future Work: improving framework with experiments on international Grids

Thanks!

Grid'BnB: Fault-Tolerance & Load-Balancing

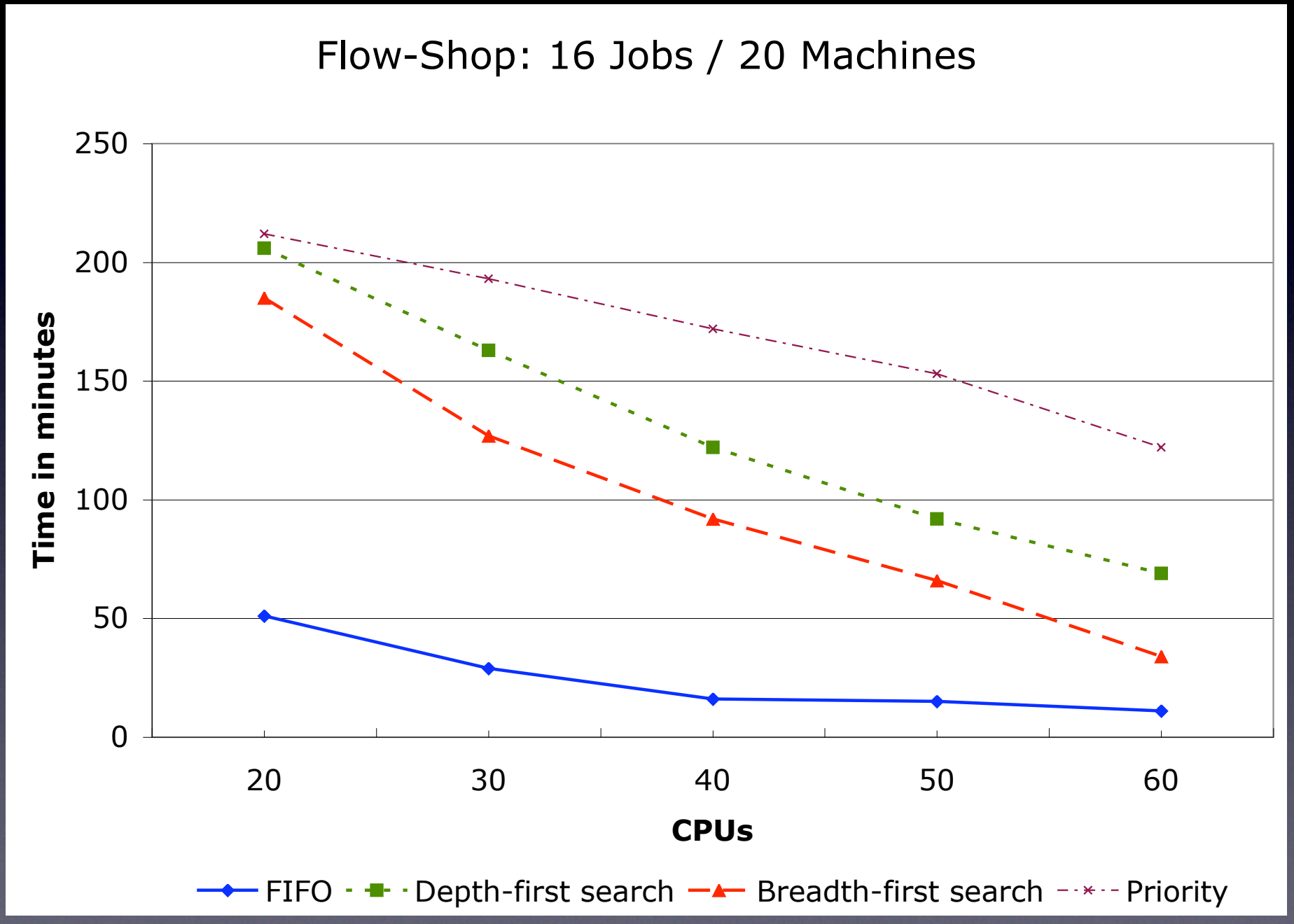
- Manage user exception ➟ computation stopped

  For us, a fault is a Failed Stop

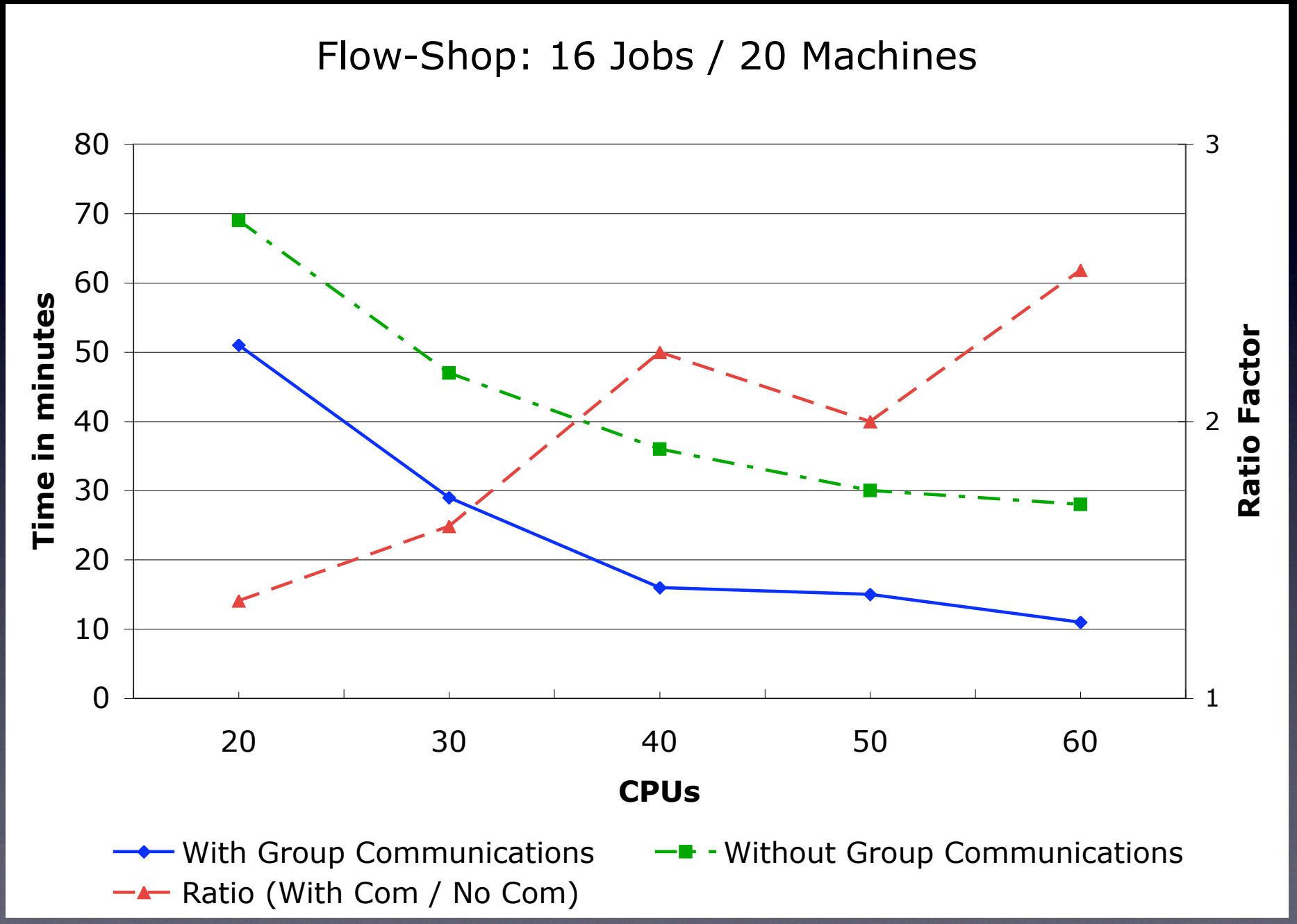
- Worker fault ➟ handled by (sub-)master
- Leader fault ➟ master chooses a new one
- Sub-master fault ➟ master turns a worker to sub-master
- Master fault ➟ restart from last saved checkpoint

- Load-Balancing is natural with master-worker
  - the framework provides a function to get the number of free Workers ➟ users use it to decide branching

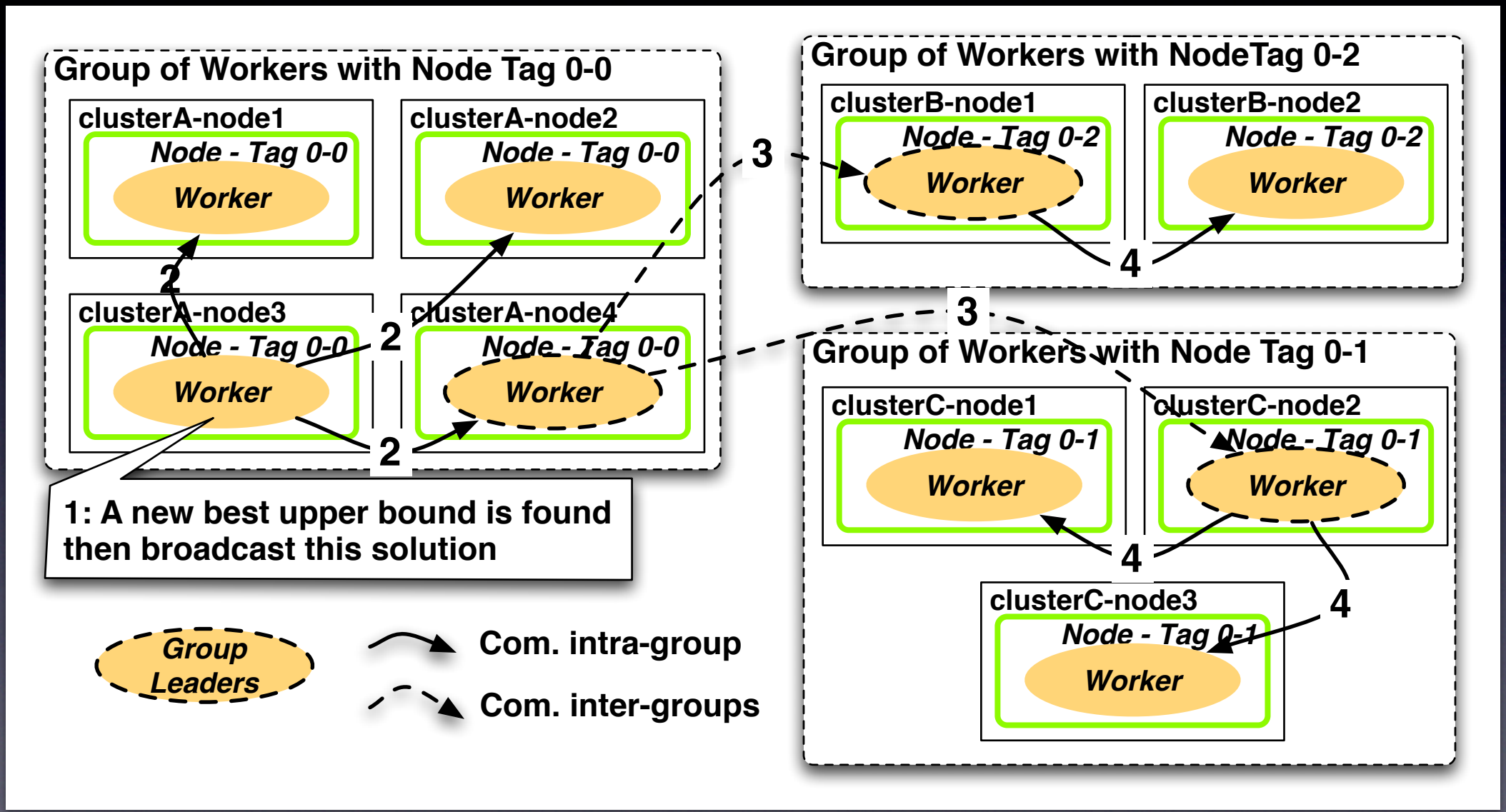
Single Cluster: Search Strategies



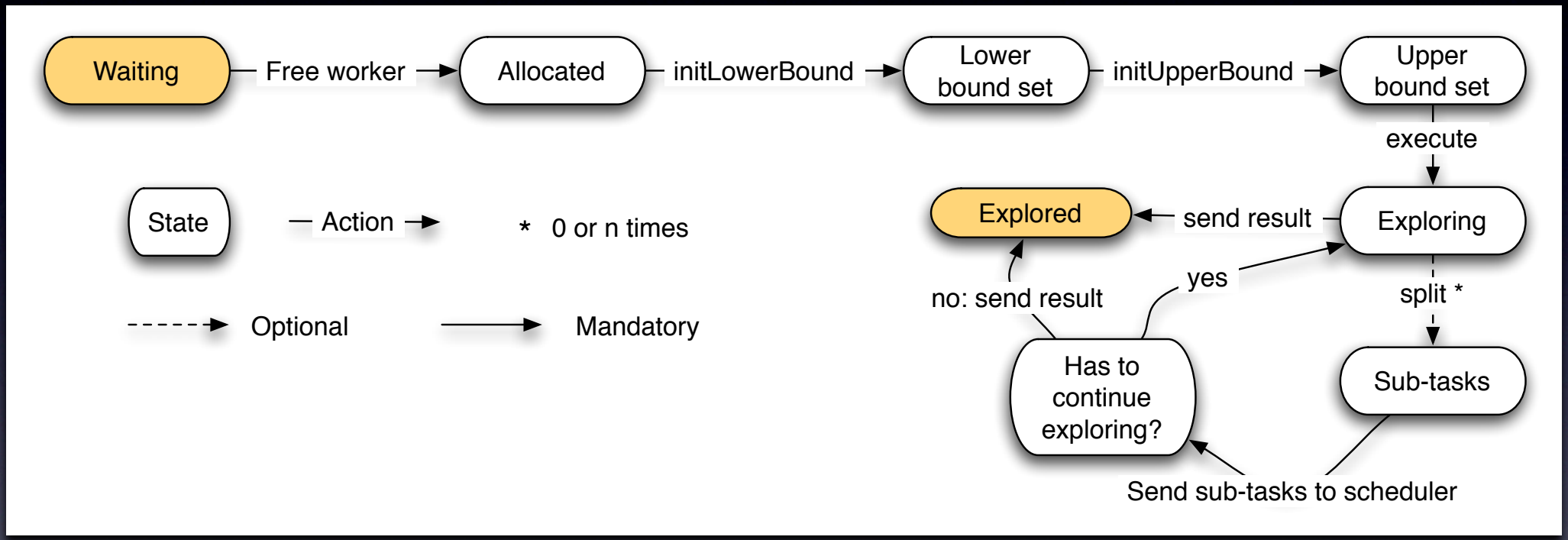
Single Cluster: Communications



Grid Node Localization



Task State Diagram



Grid'BnB: API

```
public abstract class Task<V> {
    protected V GUB;
    protected Worker worker;

    public abstract V explore(Ob ject[] params);
    public abstract ArrayList<?extends Task<V>> split();
    public abstract void initLowerBound();
    public abstract void initUpperBound();
    public abstract V gather(V[] values ) ;
}
```

```
public interface Worker <T extends Task, V>{
    public abstract IntWrapper availableWorkers();
    public abstract void sendSubTasks(ArrayList<T> subTasks);
    public abstract void newBestBound(V betterBound)
}
```