Dynamic Leader Election using

Metaheuristic Algorithm

Algorithm

```
Input: List<Job> jobs, List<Server> servers

Output: Selected Leader Election Algorithm with priority

if (time != scheduling_interval) return

flock, memory, mem_fitness, best = initialize(jobs, servers)

for iteration in range(MAX_ITERATION):

    p = selectFollowerCrow(flock)

    q = selectFollowedCrow(flock, p)
```

Algorithm

```
if rand() >= AWARENESS_PROBABILITY:
    crow = p + FL*r*(q-p) # described later
else:
    crow = p + FL*r*(rand_pos()-p)
update memory, mem_fitness and best if better fitness
if improvement is negligible for last N iterations:
    break
```

Algorithm

```
assignment = assignJob(servers, jobs, best)
assignPriorityId(assignment, best)
leader = mostPriority(assignment)


algorithm = selectAlgorithm(assignment)
submitJobs(assignment)
return leader
```

Fitness

cost = [0.1 0.3 0.2 0.4] . [usedProcessorElements ramUtilization bwUtilization usedThroughput]

fitness = 1 / cost

Leader Selection Cost/Weight might have different weights / resources to consider

Algorithm (Bully/Ring) will also be chosen using different criteria (eg. more priority to bandwidth / link between hosts)

Crow Search Algorithm - Mapping

Each Crow is Array<Assignment>, where size is same as # of jobs.

Each Assignment is a vector of dimension equals the size of server. There will be only one value of "1" and others will be "0".

"1" in j-th element of the assignment vector of i-th element in the array means -

i-th job is assigned to j-th server

Crow Search Algorithm - Arithmetic Operation

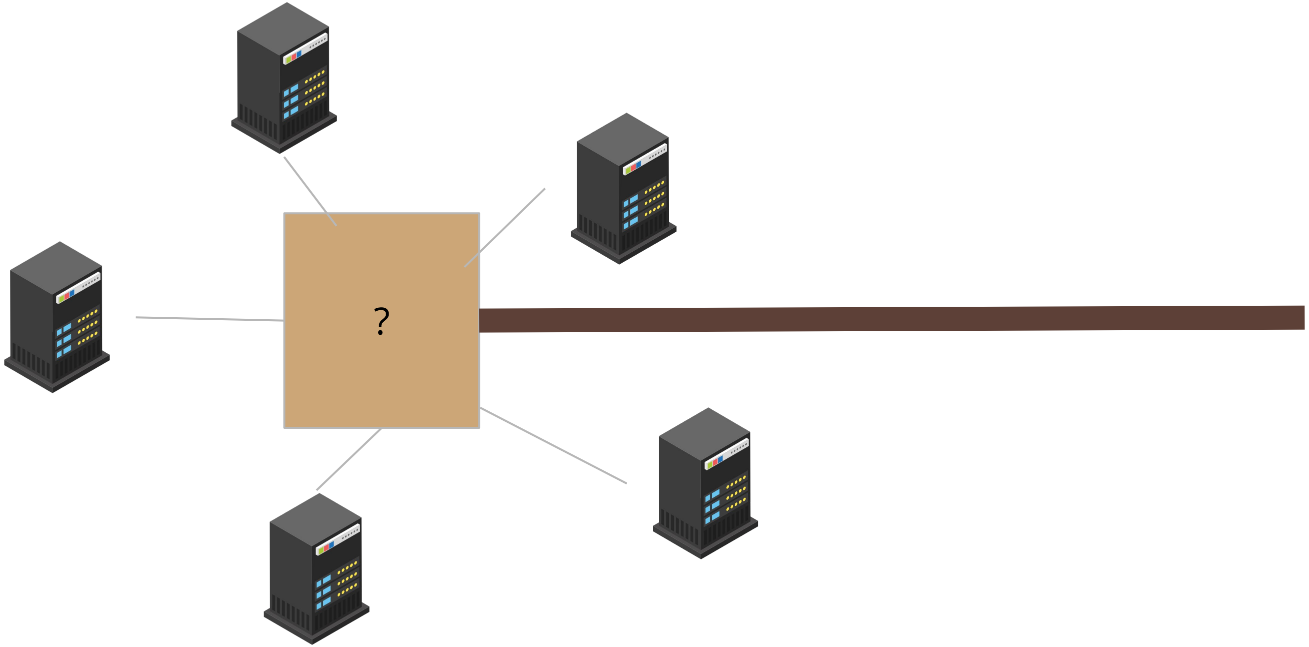
crow = p + FL * r * (q-p)

Here, p and q is crow, FL = Flight Length (Parameter) and **r is vector** of size with number of tasks with random number.

r is multiplied with (q-p) vector element wise

The resulting crow might not have only "0" and "1"s. So, in each assignment, we set **corresponding maximum value to 1** and others to 0.

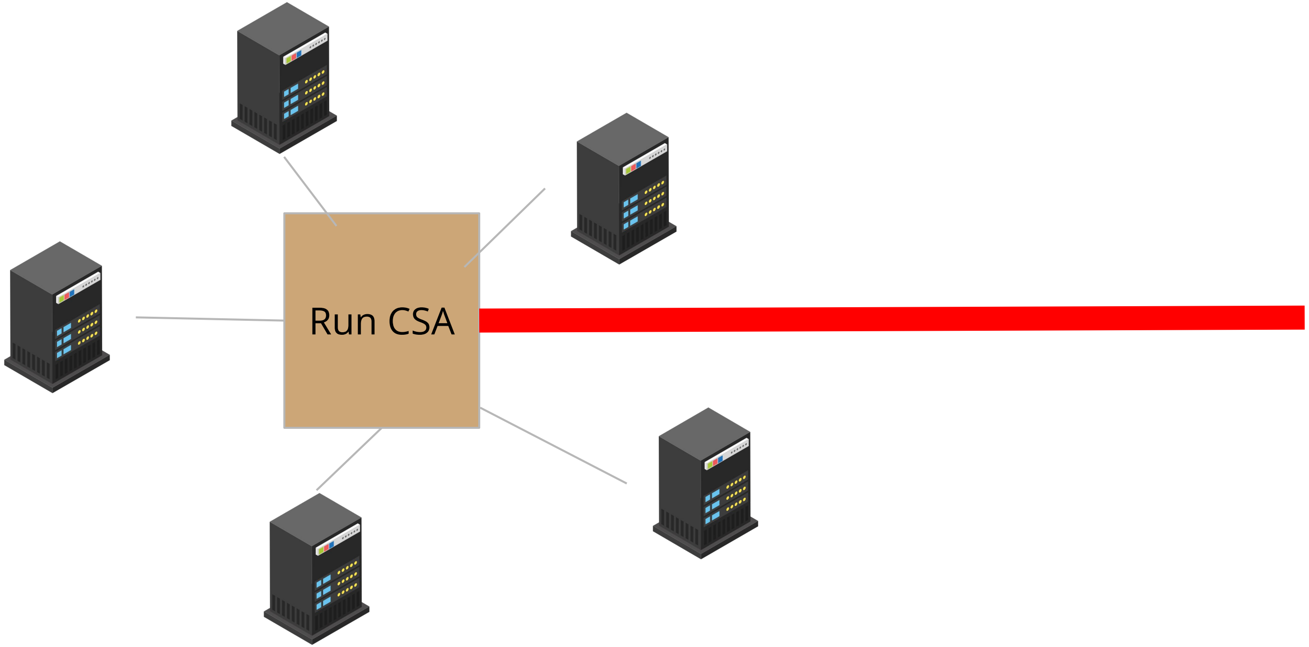
Data Center : Network Topology - Any



Data Center : Scheduling Interval

Metrics -

- Power Consumption
- Bandwidth
- Ram
- Utilization
- CPU

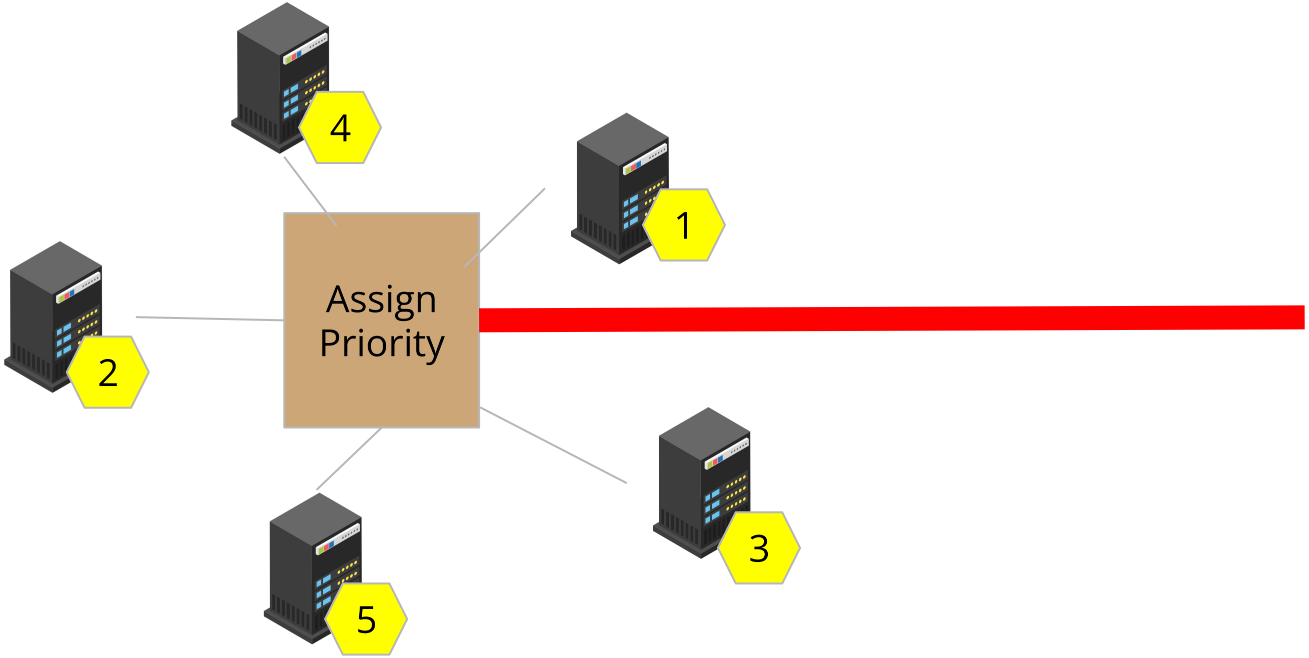


Run CSA

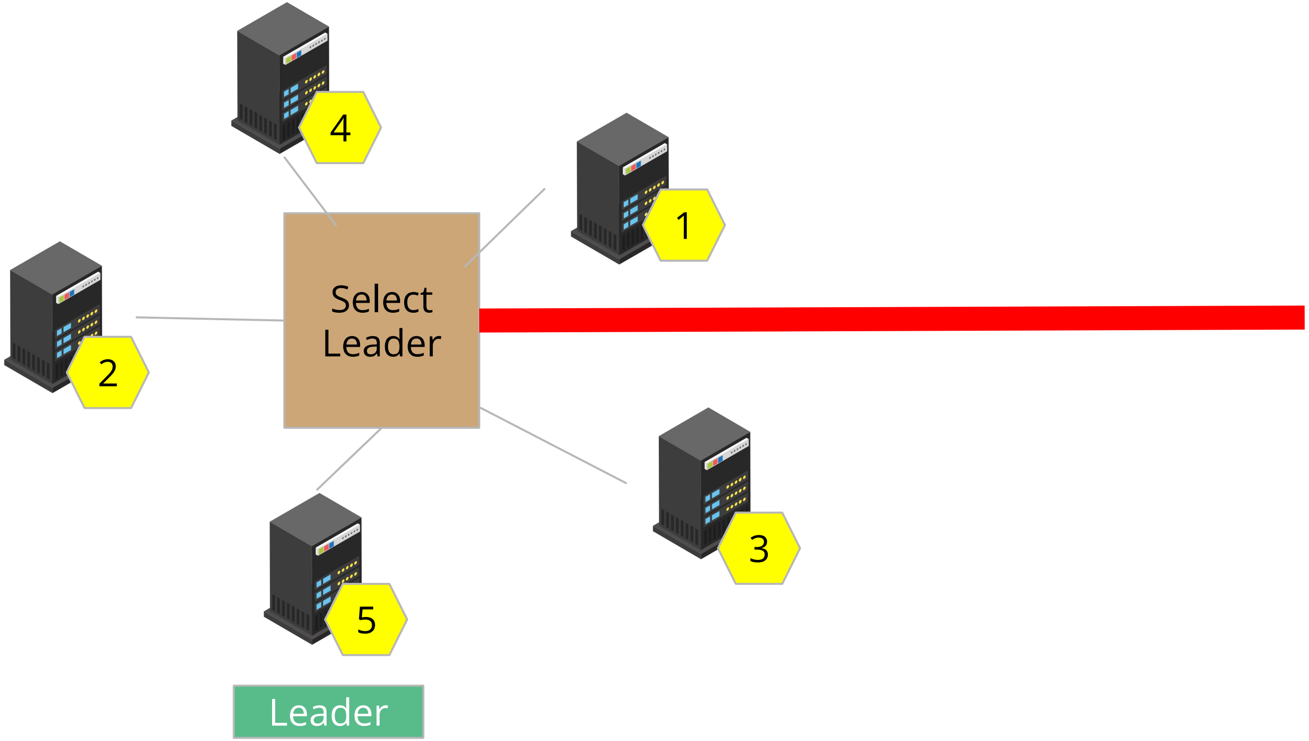
Data Center : Scheduling Interval

Metrics -

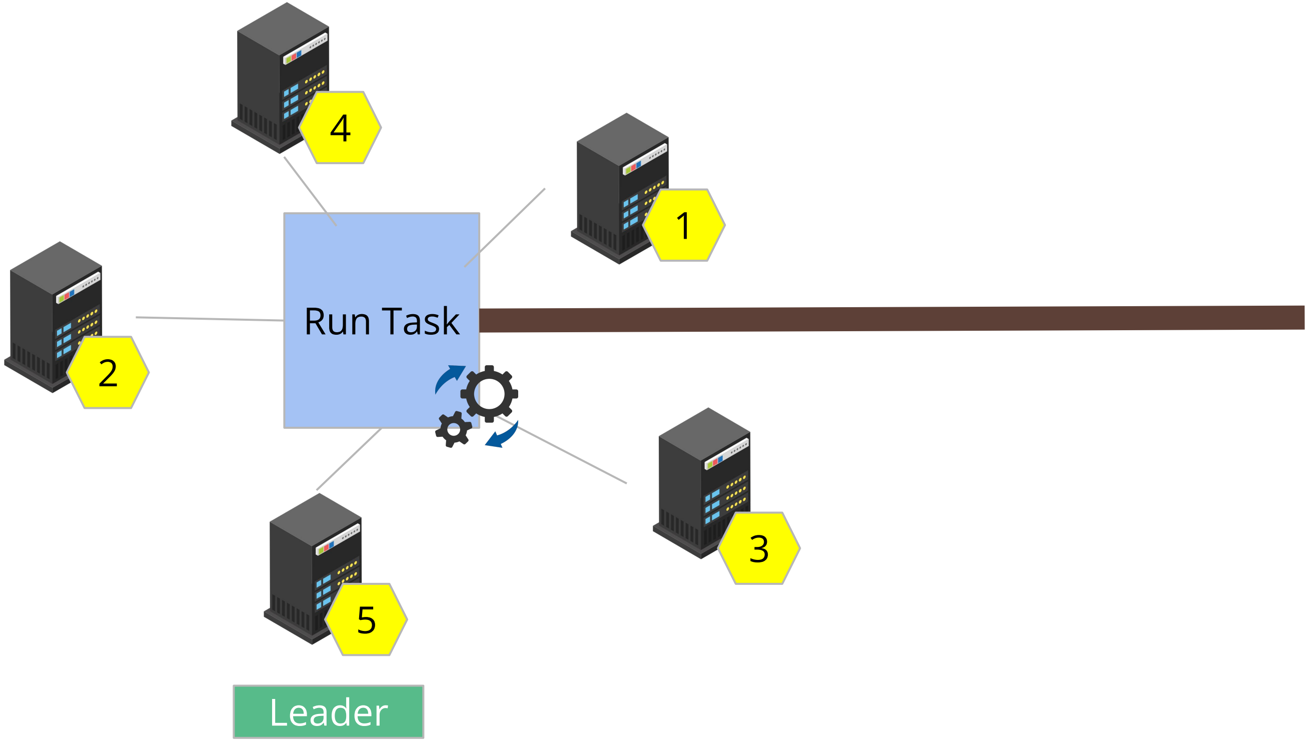
- Bandwidth
- CPU
- Ram



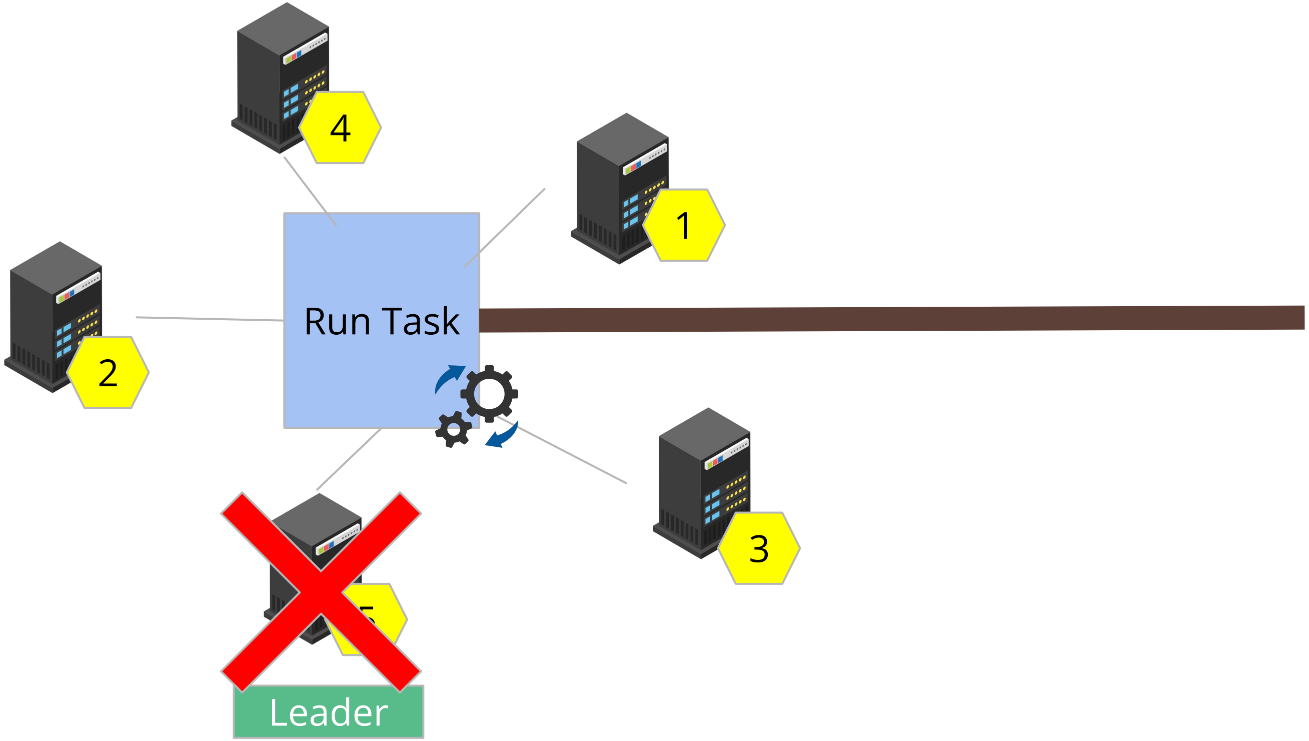
Data Center : Scheduling Interval



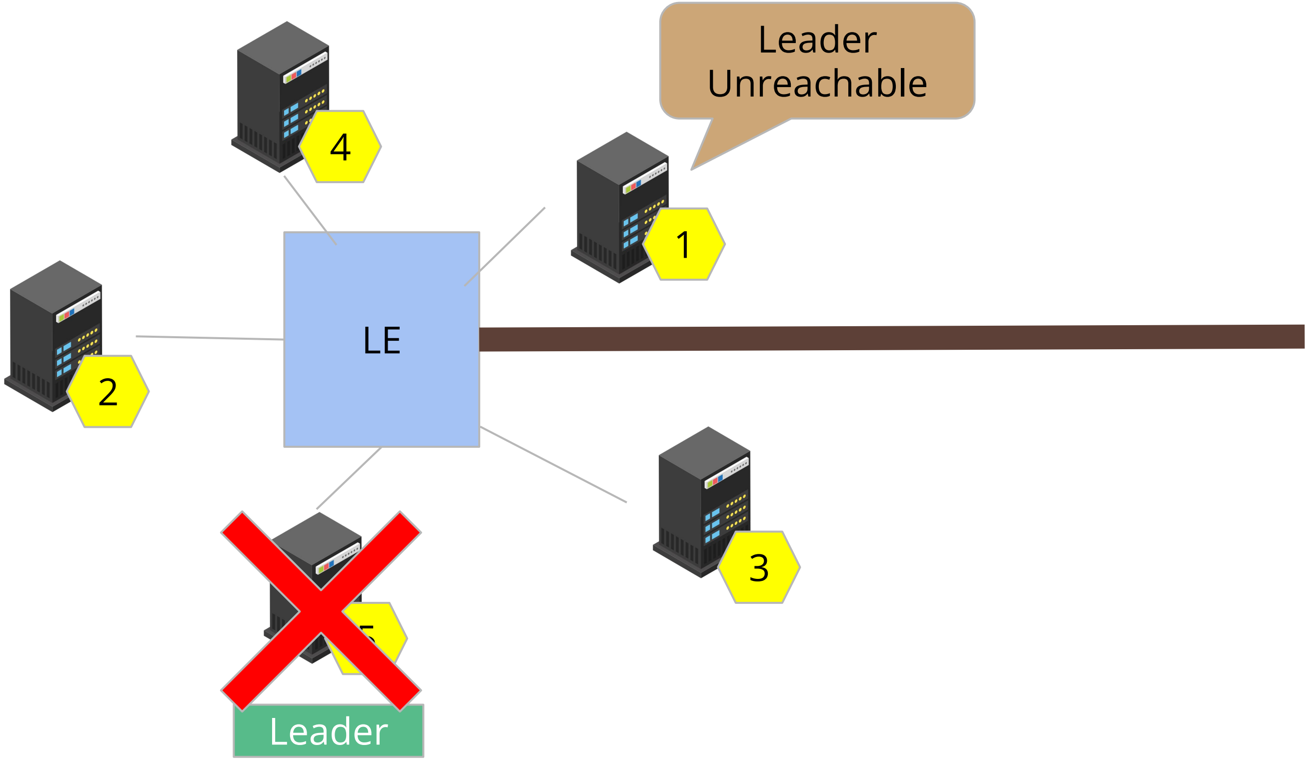
Data Center : Execution



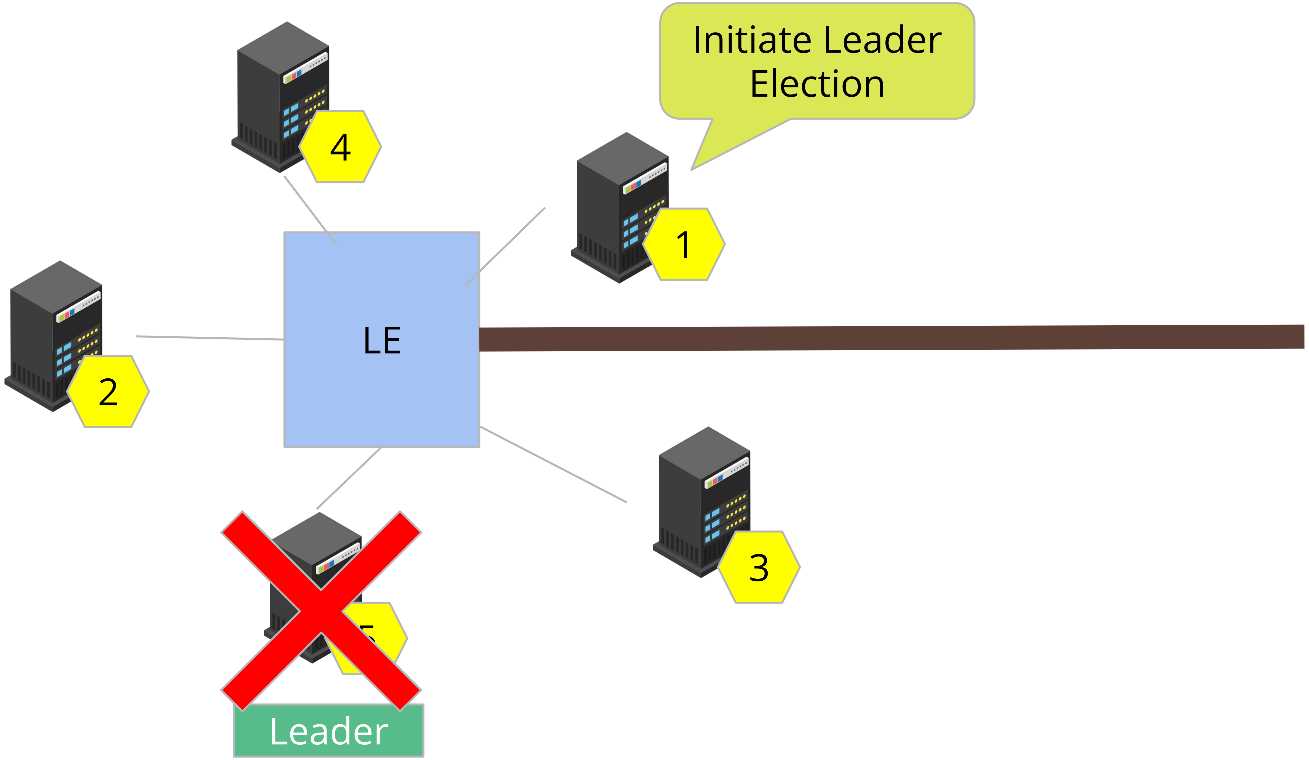
Data Center : Leader Failed



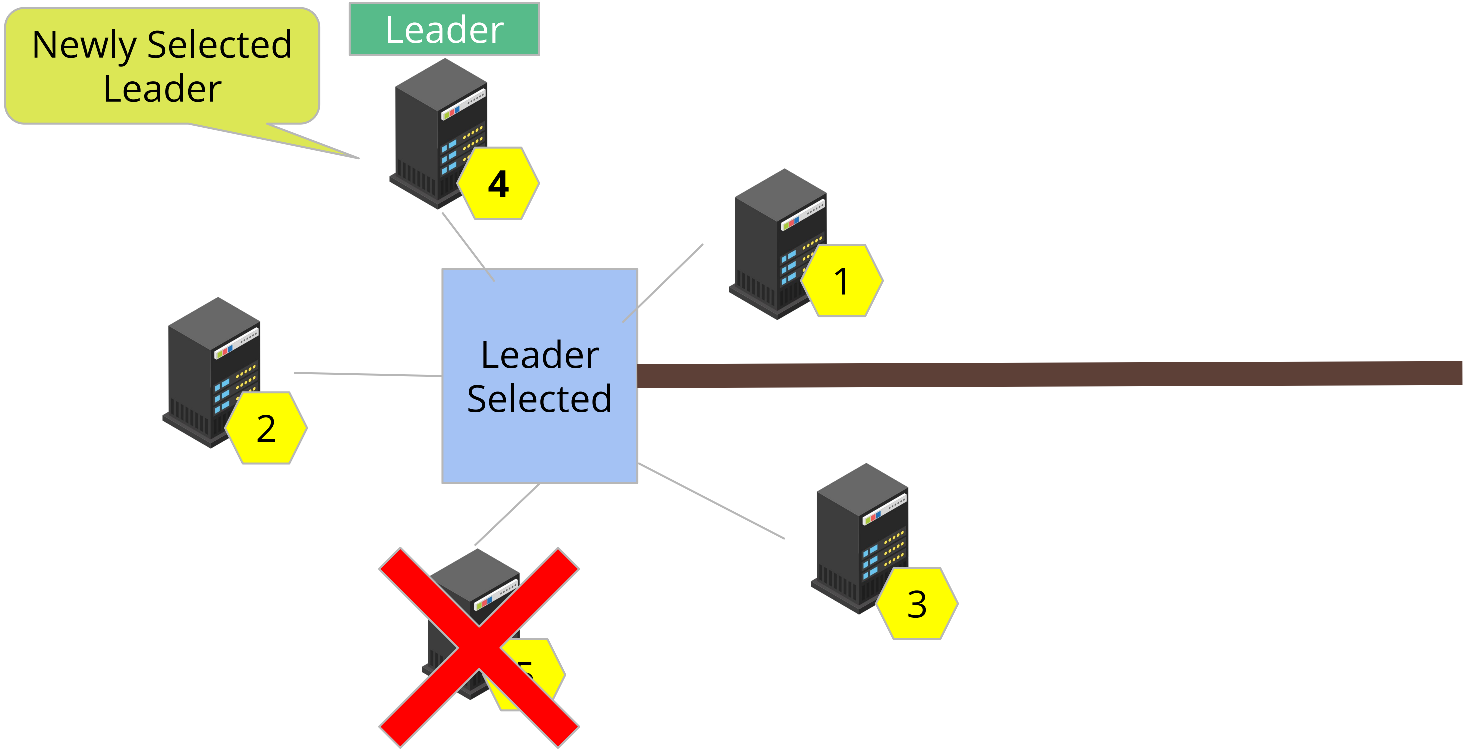
Data Center : Leader Failed Identified



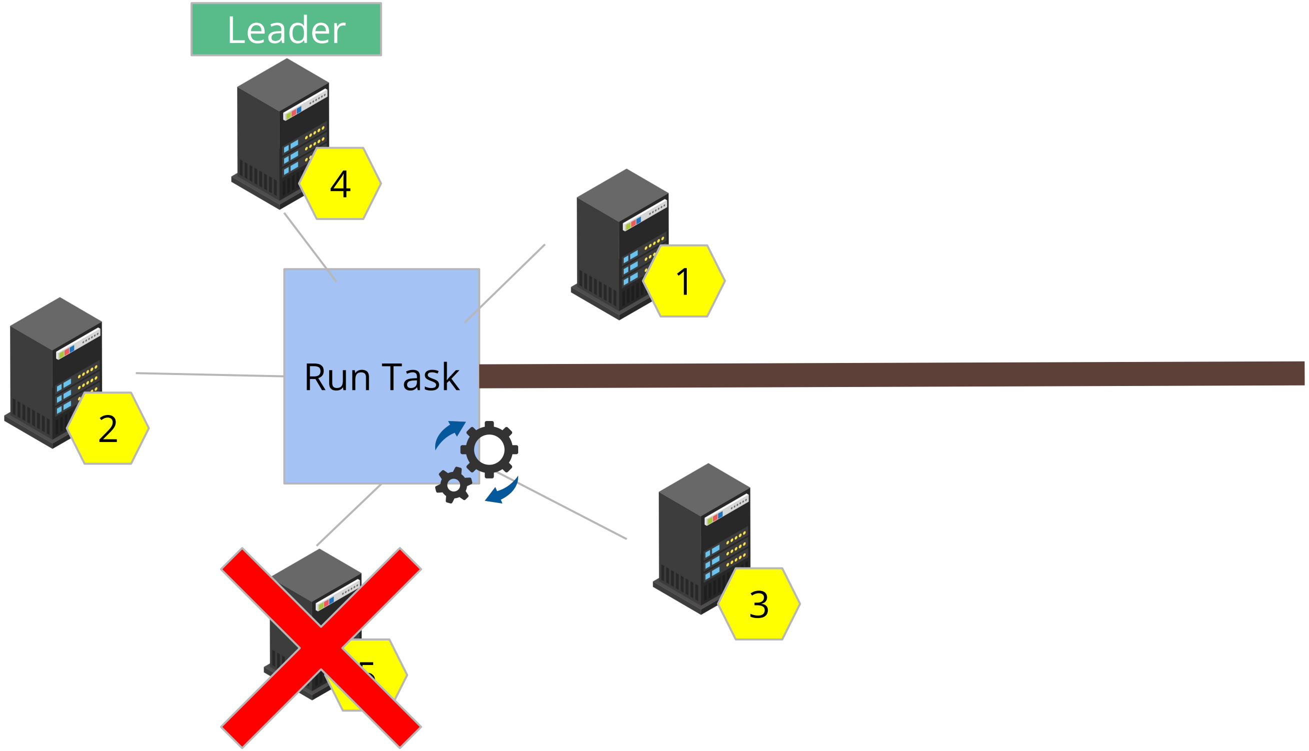
Data Center : Bully Algorithm or other variant



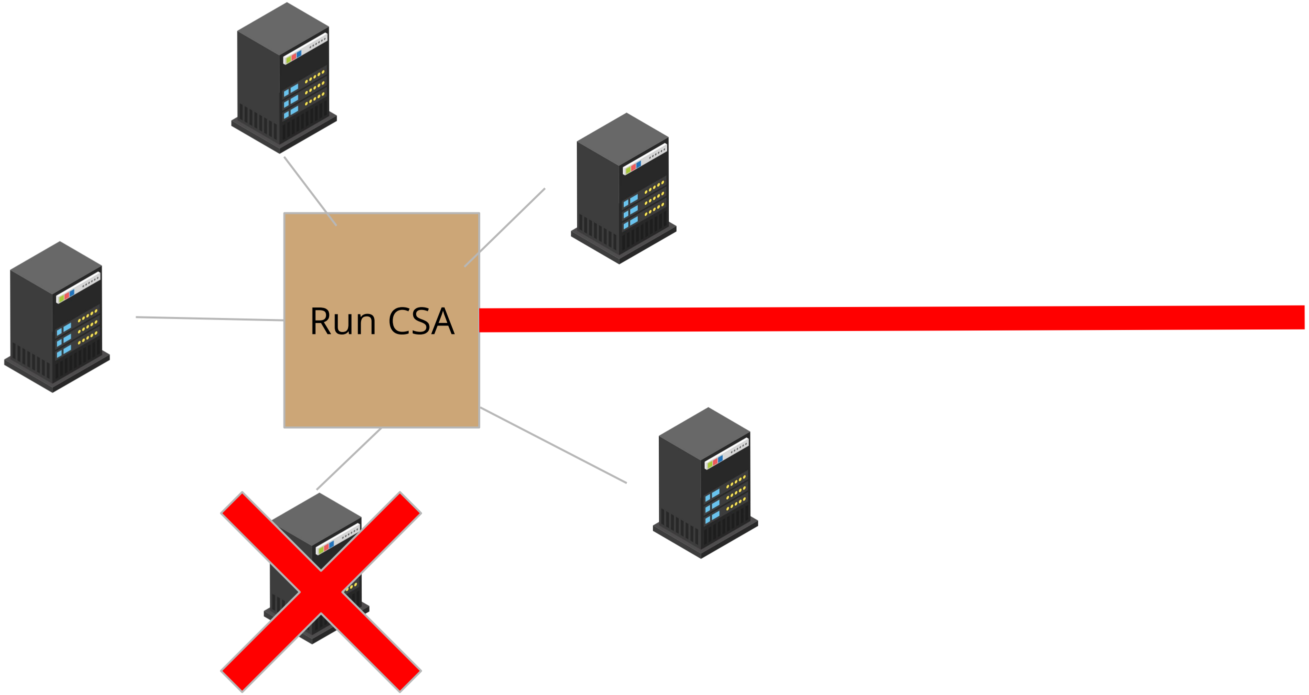
Data Center : Leader Selected



Data Center : Leader Selected



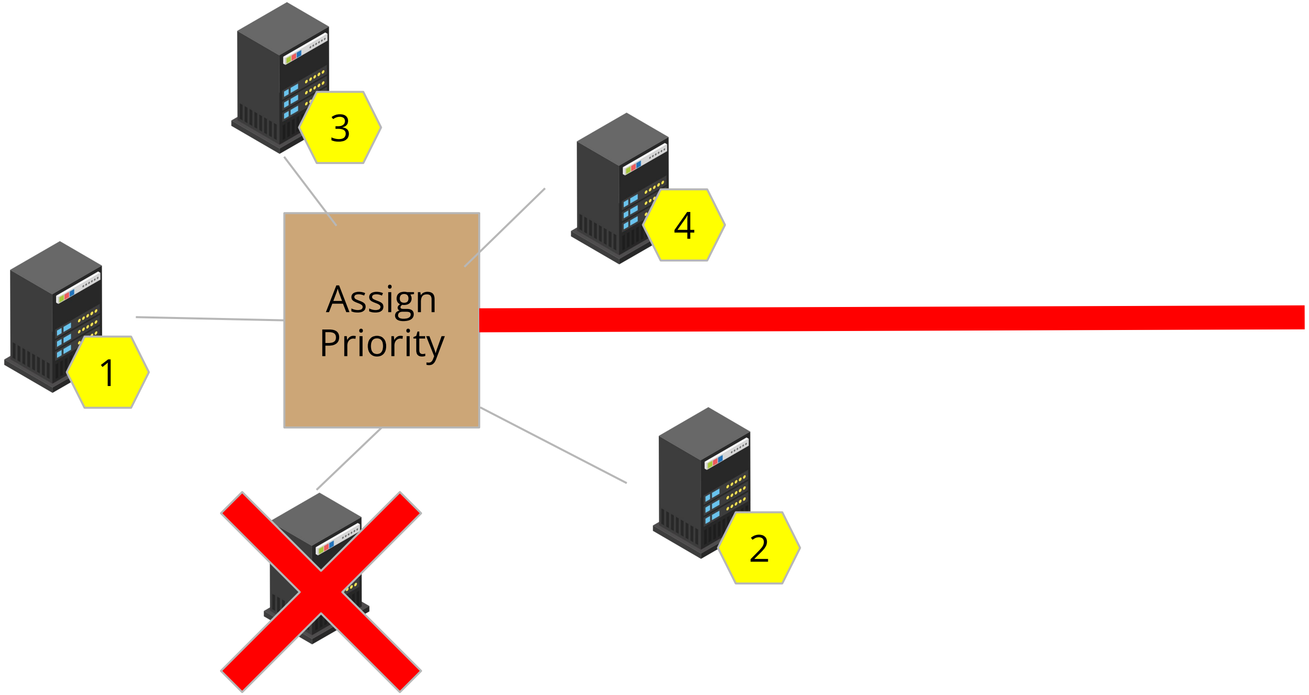
Data Center : Scheduling Interval



Data Center : Scheduling Interval

Metrics -

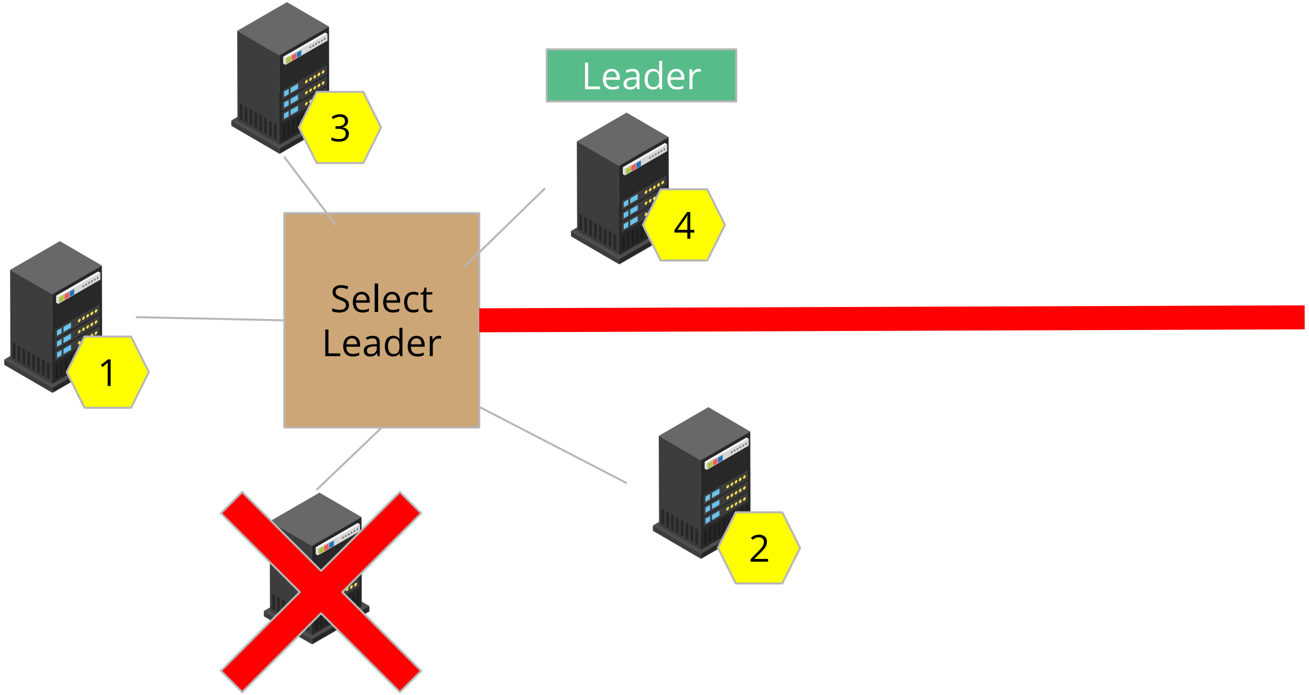
- Power Consumption
- Bandwidth
- Ram
- Utilization
- CPU



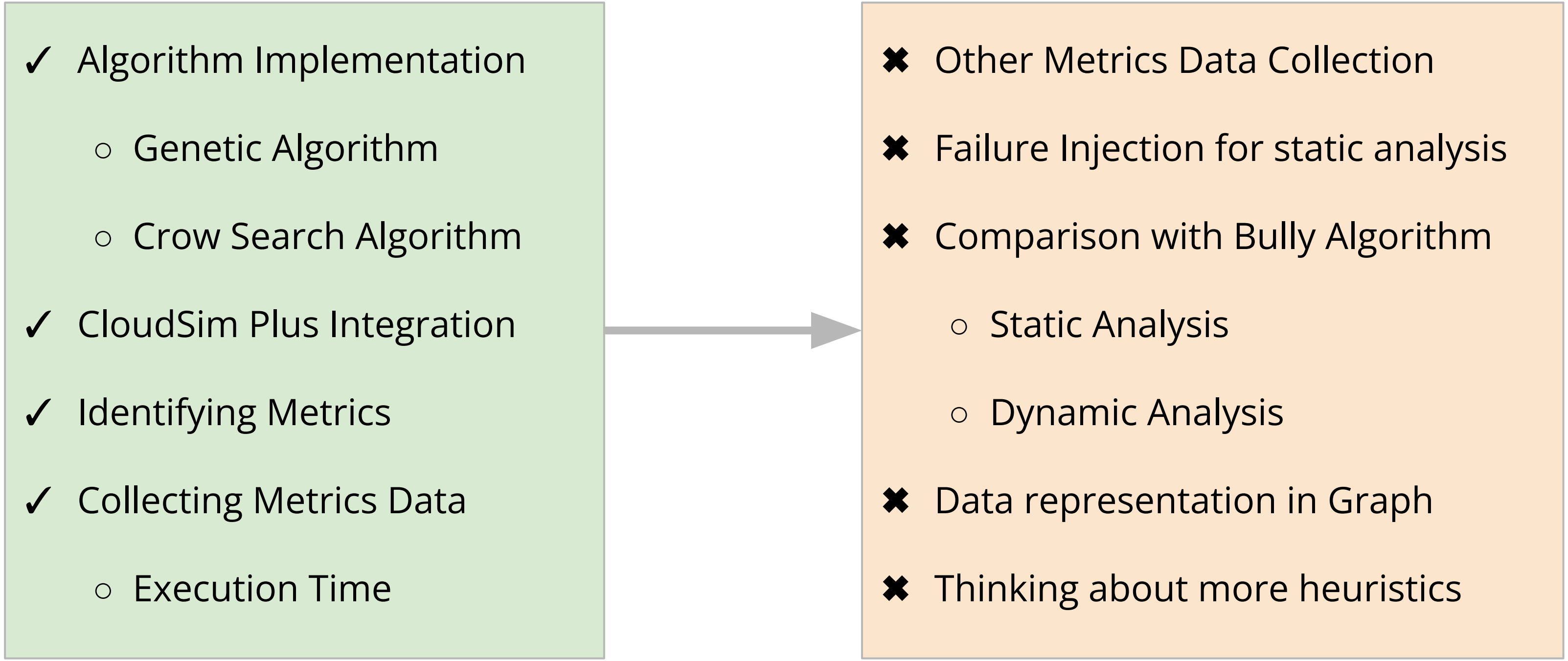
Data Center : Scheduling Interval

Metrics -

- Bandwidth
- CPU
- Ram



Thesis Experiment Schedule



- ✓ Algorithm Implementation
  - ○ Genetic Algorithm
  - ○ Crow Search Algorithm
- ✓ CloudSim Plus Integration
- ✓ Identifying Metrics
- ✓ Collecting Metrics Data
  - ○ Execution Time

- ✖ Other Metrics Data Collection
- ✖ Failure Injection for static analysis
- ✖ Comparison with Bully Algorithm
  - ○ Static Analysis
  - ○ Dynamic Analysis
- ✖ Data representation in Graph
- ✖ Thinking about more heuristics

Thesis Experiment Schedule (Upto Mid-break)

This Week (W1):

Finalizing Schedule, Collecting and plotting data -

Execution Time and Fitness

Week 2 (W2):

Collecting and plotting data -

~~SLA Violation~~, Memory Utilization, CPU Utilization, Power Consumption and others

Exploring fault injection for bully comparison

Thesis Experiment Schedule (Upto Mid-break)

Week 3 (W3):

- Automating data collection and plotting
- Variable # of VM, task; End of dynamic analysis code
- *Exploring fault injection for bully comparison*

Week 4 (W4):

- **Exploring fault injection for bully comparison**
- **Static analysis** for bully

Thesis Experiment Schedule (Upto Mid-break)

Week 5 (W5):

Finalizing data collection variables, parameters and experiment

Ready for collecting final data for experiment

Start collection of final data

Mid-break:

Finish data collection