BeeCluster: Drone Orchestration via Predictive Optimization

Presenter: Songtao He (songtao@mit.edu)
Present at ACM MobiSys 2020

Joint work with Favyen Bastani, Arjun Balasingam, Karthik Gopalakrishnan, Ziwen Jiang, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, Sam Madden
Aerial Drones Enable Many Applications

- Aerial Mapping
- Inspection
- Environmental Monitoring
- Agriculture
- Delivery
- Disaster Management
- Wildlife Monitoring
- Search and Rescue
Emerging Autonomous Drone Applications

Applications which require human operators

Autonomous Drone Applications

https://www.dronepilotgroundschool.com/full-time-drone-jobs/
https://www.information-age.com/ai-powered-drone-swarms-123479202/
Developing Autonomous Drone Applications is Challenging

Drone Application

- App Logic
- Optimization Logic
- Drone Fleet Control Logic
- Driver
- Drone

- Where to collect sensing data?
- How to process the data?
- How to react to the processing result?

- Task-to-drone scheduling.
- Routes planning.
- Drone monitoring, e.g., battery time

- Access raw sensors
- Drone control
- Telemetry
Drone Orchestration Comes to Help

Traditional Drone Applications

- App 1
  - App Logic
  - Optimization Logic
  - Drone Fleet Control Logic
  - Driver
  - Drone

- App 2
  - App Logic
  - Optimization Logic
  - Drone Fleet Control Logic
  - Driver
  - Drone

- App 3
  - App Logic
  - Optimization Logic
  - Drone Fleet Control Logic
  - Driver
  - Drone

Orchestrated Drone Applications

Drone Orchestration Platform

- App 1
  - App Logic
  - App-Independent Optimization & Drone Fleet Control Logic
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver

- App 2
  - App Logic
  - App-Independent Optimization & Drone Fleet Control Logic
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver

- App 3
  - App Logic
  - App-Independent Optimization & Drone Fleet Control Logic
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver
  - Driver

App 1
App 2
App 3

Drone Fleet
Control Logic

Driver

Drone
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
- Application-independent Optimization

Orchestrated Drone Applications
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
- Application-independent Optimization

Orchestrated Drone Applications
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options

- Application-independent Optimization
  - (3) Predictive Optimization
    - Forecast applications’ future behavior and use the forecast to speed up the execution of the app.
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- **Programming Interface**
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options

- **Application-independent Optimization**
  - (3) Predictive Optimization
    - Forecast applications’ future behavior and use the forecast to speed up the execution of the app.
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options

- Application-independent Optimization
  - (3) Predictive Optimization
    - Forecast applications’ future behavior and use the forecast to speed up the execution of the app.
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options

- Application-independent Optimization
  - (3) Predictive Optimization
    - Forecast applications’ future behavior and use the forecast to speed up the execution of the app.
BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

● Programming Interface
  ○ Dynamic Tasking
  ○ Rich Task-to-Drone Binding Options

● Application-independent Optimization
  ○ Predictive Optimization
    ■ Forecast applications’ future behavior and use the forecast to speed up the execution of the app.
Example of Predictive Optimization

1  A,B,C = initial locations
2  while True:
3      values = SenseAtLocations([A,B,C])
4      A,B,C = UpdateLocations(A,B,C,values)
Example of Predictive Optimization

1 A,B,C = initial locations
2 while True:
3    values = SenseAtLocations([A,B,C])
4    A,B,C = UpdateLocations(A,B,C,values)
Example of Predictive Optimization

1  A,B,C = initial locations
2  while True:
3    values = SenseAtLocations([A,B,C])
4    A,B,C = UpdateLocations(A,B,C,values)

A→ C→ B (Counter-Clockwise)
A→ B→ C (Clockwise)
Example of Predictive Optimization

1. A, B, C = initial locations
2. while True:
3.     values = SenseAtLocations([A, B, C])
4.     A, B, C = UpdateLocations(A, B, C, values)

A → C → B (Counter-Clockwise)
A → B → C (Clockwise)
Example of Predictive Optimization

1. A, B, C = initial locations
2. while True:
3.     values = SenseAtLocations([A, B, C]) # Second Iteration
4.     A, B, C = UpdateLocations(A, B, C, values)

Case 1 (Better)

Case 2
Example of Predictive Optimization

1. \(A, B, C = \text{initial locations}\)
2. \(\textbf{while True:}\)
3. \(\text{values} = \text{SenseAtLocations}([A, B, C]) \# \text{First Iteration}\)
4. \(A, B, C = \text{UpdateLocations}(A, B, C, \text{values})\)
Outline

● Introduction
● Motivation
● **Design of BeeCluster**
  ○ Programming Interface
    ■ Programming model (dynamic tasking)
    ■ Task-to-drone mapping options
  ○ Predictive Optimization
● Implementation
● Case Studies
Outline

- Introduction
- Motivation
- Design of BeeCluster
  - Programming Interface
    - Programming model (dynamic tasking)
    - Task-to-drone mapping options
  - Predictive Optimization
- Implementation
- Case Studies
BeeCluster Programming Model

- Programming Model
  - An imperative programming model based on dynamic task graph (DTG)
BeeCluster Programming Model

- Programming Model
  - An imperative programming model based on dynamic task graph (DTG)

- Programming Primitives (Building Blocks)
  - Actions - E.g., fly to a location, and take a photo.
  - Tasks - Sequence of actions, multiple tasks run concurrently.
BeeCluster Programming Model

● Programming Model
  ○ An imperative programming model based on dynamic task graph (DTG)

● Programming Primitives (Building Blocks)
  ○ Actions - E.g., fly to a location, and take a photo.
  ○ Tasks - Sequence of actions, multiple tasks run concurrently.

● Basic API
  ○ Non-blocking creations of new actions and new tasks:
    ■ handler = act(args)
    ■ handler = newTask(function_entry, args, ...)
  ○ Retrieve values from the handler (blocking): result = handler.value
  ○ Cancel tasks: cancelTask(task_handler)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking
    loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
    values = [task0.value, task1.value, task2.value, task2.value]
    # blocking
    loc = GradientDescent(locs, values)
```
```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ]  # blocking
    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)

        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [ task0.value, task1.value,
                   task2.value, task2.value ]
                   # blocking

    loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking

    loc = GradientDescent(locs, values)
```
```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [task0.value, task1.value, task2.value, task2.value]
        # blocking
    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)

        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

```python
1 def sense(loc):
2     act("flyto", loc)
3     handle = act("measure_WIFI_signal")
4     return handle.value  # blocking
5
6 if __name__ == "__main__":
7     loc = initial_loc  # e.g., (0,0)
8     for i in range(10):
9         locs = four_corners(loc)
10     task0 = newTask(sense, locs[0])
11     task1 = newTask(sense, locs[1])
12     task2 = newTask(sense, locs[2])
13     task3 = newTask(sense, locs[3])
14
15     values = [ task0.value, task1.value,
16                 task2.value, task2.value ]
17     loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [task0.value, task1.value, task2.value, task2.value]
        # blocking
    loc = GradientDescent(locs, values)
```
```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [task0.value, task1.value, task2.value, task2.value]  # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent
```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [task0.value, task1.value, task2.value, task2.value]
        # blocking
    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent
Example: Locate WiFi Hotspot Using Gradient Descent

```python
1 def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking

5 if __name__ == "__main__":
6     loc = initial_loc # e.g., (0,0)
7     for i in range(10):
8         locs = four_corners(loc)
9
10     task0 = newTask(sense, locs[0])
11     task1 = newTask(sense, locs[1])
12     task2 = newTask(sense, locs[2])
13     task3 = newTask(sense, locs[3])
14
15     values = [ task0.value, task1.value, task2.value, task2.value ] # blocking
16     loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act(“flyto”, loc)
    handle = act(“measure_WiFi_signal”)
    return handle.value # blocking

if __name__ == “__main__”:
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ] # blocking
        loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [task0.value, task1.value, task2.value, task2.value]  # blocking
    loc = GradientDescent(locs, values)
```
def sense(loc):
    act("flyto", loc)
    handle = act("measure WiFi signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [task0.value, task1.value, task2.value, task2.value]  # blocking
    loc = GradientDescent(locs, values)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto",loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]  # blocking

    loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking
    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

![Diagram showing the process of locating a WiFi hotspot using gradient descent](image-url)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [task0.value, task1.value, task2.value, task2.value]
        # blocking
        loc = GradientDescent(locs, values)
```

Synchronization (line 16)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking
    loc = GradientDescent(locs, values)
```

Synchronization (line 16)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)

        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [ task0.value, task1.value, task2.value, task2.value ] # blocking

    loc = GradientDescent(locs, values)
```

- **Root**
  - Sense @ locs[0]
    - 1 flyto
    - 2 measure
  - Sense @ locs[1]
    - 1 flyto
    - 2 measure
  - Sense @ locs[2]
    - 1 flyto
    - 2 measure
  - Sense @ locs[3]
    - 1 flyto
    - 2 measure

Synchronization (line 16)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking
    loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto",loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)

        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [ task0.value, task1.value,
                    task2.value, task2.value ]  # blocking

    loc = GradientDescent(locs, values)
```

Synchronization (line 16)
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == '__main__':
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])

        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking

    loc = GradientDescent(locs, values)
```
Example: Locate WiFi Hotspot Using Gradient Descent

```python
def sense(loc):
    act("flyto",loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc)
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [task0.value, task1.value, task2.value, task3.value]
        # blocking
        loc = GradientDescent(locs, values)
```

Multi-threading
1  
2    for i in range(4):
3      newTask(foo, locs[i])

Conditional Branch
1  
2    v = newTask(sense).value
3  
4    if v > 100:
5      handler = newTask(foo)
6    else:
7      handler = newTask(bar)
8    handler.wait()

Dynamic Loop
1  
2    v = 0, loc = [0,0]
3  
4    while v < 50:
5      h = newTask(sense, loc)
6    v = h.value
7    loc[0] += 10

Recursive Graph Search
1  
2    def BFS(loc):
3      act("flyto", loc)
4      v = act("sense").value
5      locs = nextLocs(loc, v)
6      for newLoc in locs:
7        newTask(BFS, newLoc)

Frontend (E.g., Python, Golang)  Intermediate Representation (IR)  Backend (Different Optimizations)
Outline

● Introduction
● Motivation
● Design of BeeCluster
  ○ Programming Interface
    ■ Programming model (dynamic tasking)
    ■ Task-to-drone mapping options
  ○ Predictive Optimization
● Implementation
● Case Studies
Task-to-Drone Mapping Options

Timeline

Task

Action 1  Action 2  Action 3

Drone 1

Drone 2
Task-to-Drone Mapping Options

![Diagram showing task-to-drone mapping with actions and drones.]

- **Task**: Action 1, Action 2, Action 3
- **Drone 1**
- **Drone 2**

Timeline
Task-to-Drone Mapping Options

Timeline

Task

Action 1  Action 2  Action 3

Drone 1  Drone 2
Task-to-Drone Mapping Options

Timeline

Task

Action 1  Action 2  Action 3

Drone 1  Drone 2
Task-to-Drone Mapping Options

- **Same-Drone Flag**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.

- **Interruptible Flag**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).
Task-to-Drone Mapping Options

- **Same-Drone Flag**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.

- **Interruptible Flag**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).
Task-to-Drone Mapping Options

- **Same-Drone Flag = False**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.

- **Interruptible Flag = False**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).

![Diagram showing task timeline and drone assignments](image)
Task-to-Drone Mapping Options

- **Same-Drone Flag = False**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.

- **Interruptible Flag = False**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).
Task-to-Drone Mapping Options

- **Same-Drone Flag = True**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.

- **Interruptible Flag**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).
Task-to-Drone Mapping Options

- **Same-Drone Flag = True**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.

- **Interruptible Flag = True**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).

![Timeline diagram with tasks]

**Task**
- Pickup Package At A
- Take Photos Between A and B
- Drop Package At B

*From other Task*
Four Task-to-Drone Mapping Options

**Package Delivery**
- Same-Drone Flag = True
- Interruptible Flag = True
- Pickup Package at Location A
- Drop Package at Location B

**Recording Videos**
- Same-Drone Flag = True
- Interruptible Flag = False
- Fly to Location A
- Start Video Recording
- Fly to Location B
- Stop Video Recording

**Repeated Measures**
- Same-Drone Flag = False
- Interruptible Flag = True
- Fly to Location A
- Measure WiFi Signal
- Measure WiFi Signal
- Measure WiFi Signal

**Object Tracking**
- Same-Drone Flag = False
- Interruptible Flag = False
- Locate and Move
- Locate and Move
- Locate and Move
- Locate and Move
Outline

- Introduction
- Motivation
- Design of BeeCluster
  - Programming Interface
    - Programming model (dynamic tasking)
    - Task-to-drone mapping options
  - Predictive Optimization
- Implementation
- Case Studies
Predictive Optimization - Application Prediction

From , what will happen next?

or

or

Cancel
Predictive Optimization - Application Prediction

Current Dynamic Task Graph

Historical Dynamic Task Graphs
(Include the early portion of the current DTG)

Predictions
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model
Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model
Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model
Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Predicted Task
Active Task

Assignment

Drone
Drone
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
● Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Predictive Optimization - Scheduling

Scheduling Model
- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm
Outline

● Introduction
● Motivation
● Design of BeeCluster
  ○ Programming Interface
  ○ Predictive Optimization
● Implementation
● Case Studies
BeeCluster Implementation

- **Software**
  - 15K lines of code.
    - 80.8% in Golang for the BeeCluster core system.
    - 5.3% in python for the python wrapper of BeeCluster API.
    - 13.9% in C++ for the drone driver.

- **Hardware**
Drone Endpoint
- Drone-Specific Drivers
- Software/Hardware Boundary

Centralized
- BeeCluster Runtime
  - API Layer
  - Drone Hub
  - Simulator

Onboard
- Drone Endpoint
  - Hardware Abstraction Layer
    - Drone-Specific Drivers
    - Software/Hardware Boundary

Applications

DJI N3 Flight Controller
DJI F450 Drone Frame
5000 mAh 4S Battery (15-17 mins flying time)
Wide-Angle Camera
Raspberry Pi 3 (Onboard Computer)
WiFi Antenna

DJI N3 Flight Controller
DJI F450 Drone Frame
5000 mAh 4S Battery (15-17 mins flying time)
Wide-Angle Camera
Raspberry Pi 3 (Onboard Computer)
WiFi Antenna
Case Study 1 - WiFi Hotspot Localization
Case Study 1 - WiFi Hotspot Localization

```python
def sense(loc):
    act("flyto",loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc, "10 meters")
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value,
                   task2.value, task2.value ]  # blocking
    loc = GradientDescent(locs, values)
```

```python
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
    loc = initial_loc # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc, "10 meters")

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```
def sense(loc):
    act("flyto", loc)
    handle = act("measure_WiFi_signal")
    return handle.value  # blocking

if __name__ == "__main__":
    loc = initial_loc  # e.g., (0,0)
    for i in range(10):
        locs = four_corners(loc, "10 meters")
        task0 = newTask(sense, locs[0])
        task1 = newTask(sense, locs[1])
        task2 = newTask(sense, locs[2])
        task3 = newTask(sense, locs[3])
        values = [ task0.value, task1.value, task2.value, task2.value ]
        # blocking
    loc = GradientDescent(locs, values)
Case Study 1 - WiFi Hotspot Localization
Case Study 1 - WiFi Hotspot Localization

Local optimal
but not global optimal
Case Study 1 - WiFi Hotspot Localization

Local optimal but not global optimal

Global optimal with predictive optimization
Case Study 2 - Mapping Wifi Coverage
Case Study 3 - Mapping New Roads
(a) Without Speculative Execution

Drone 1

Drone 2

(b) Speculative Execution with 2 drones

Drone 1

Drone 2

Flying  Sensing  Flying  Sensing  Flying  Sensing

Flying  Sensing  Flying  Sensing  Flying  Sensing
BeeCluster Drone Orchestration Platform

Goal:
- Make it easy to build cross-platform and optimized drone applications.

Key Features:
- Dynamic Tasking
- Rich Task-to-Drone Mappings
- Predictive Optimization

Open-Source Project
Website: beecluster.csail.mit.edu
GitHub: github.com/songtaohe/BeeCluster