SwarmNetwork [WIP]

The `SwarmNetwork` class is a powerful tool for managing a pool of agents, orchestrating task distribution, and scaling resources based on workload. It is designed to handle tasks efficiently by dynamically adjusting the number of agents according to the current demand. This class also provides an optional API for interacting with the agent pool, making it accessible for integration with other systems.

Key Features

- **Agent Pool Management**: Dynamically manage a pool of agents.
- **Task Queue Management**: Handle tasks through a queue system.
- **Agent Health Monitoring**: Monitor the health of agents.
- **Agent Pool Scaling**: Scale the agent pool up or down based on workload.
- **API**: Interact with the agent pool and task queue through a simple API.
- **Agent Deployment Options**: Run agents on threads, processes, containers, machines, or clusters.

Parameters

Parameter	Type	Default Value	Description
name	str	None	The name of the swarm network.
description	str	None	A description of the swarm network.

| agents | List[Agent] | None | A list of agents in the pool.

|

| idle_threshold | float | 0.2 | The idle threshold for the agents.

|

| busy_threshold | float | 0.7 | The busy threshold for the agents.

|

| api_enabled | Optional[bool] | False | A flag to enable/disable the API.

|

| logging_enabled | Optional[bool] | False | A flag to enable/disable logging.

|

| api_on | Optional[bool] | False | A flag to enable/disable the FastAPI instance.

|

| host | str | "0.0.0.0" | The host address for the FastAPI instance.

|

| port | int | 8000 | The port number for the FastAPI instance.

|

| swarm_callable | Optional[callable] | None | A callable to be executed by the swarm network.

|

| *args | tuple | | Additional positional arguments. |

| **kwargs | dict | | Additional keyword arguments.

|

Attributes

| Attribute | Type | Description |

|---|---|---|

task_queue	queue.Queue	A queue for storing tasks.
idle_threshold	float	The idle threshold for the agents.
busy_threshold	float	The busy threshold for the agents.
agents	List[Agent]	A list of agents in the pool.
api_enabled	bool	A flag to enable/disable the API.
logging_enabled	bool	A flag to enable/disable logging.
host	str	The host address for the FastAPI instance.
port	int	The port number for the FastAPI instance.
swarm_callable	Optional[callable]	A callable to be executed by the swarm network.
agent_dict	dict	A dictionary of agents for easy access.
lock	threading.Lock	A lock for synchronizing access to shared resources.

Methods

Description

Initializes a new instance of the `SwarmNetwork` class.

Parameters

- `name` (str): The name of the swarm network.
- `description` (str): A description of the swarm network.
- `agents` (List[Agent]): A list of agents in the pool.
- `idle_threshold` (float): The idle threshold for the agents.
- `busy_threshold` (float): The busy threshold for the agents.
- `api_enabled` (Optional[bool]): A flag to enable/disable the API.
- `logging_enabled` (Optional[bool]): A flag to enable/disable logging.
- `api_on` (Optional[bool]): A flag to enable/disable the FastAPI instance.

- `host` (str): The host address for the FastAPI instance.

- `port` (int): The port number for the FastAPI instance.

- `swarm_callable` (Optional[callable]): A callable to be executed by the swarm network.

- `*args`: Additional positional arguments.

- `**kwargs`: Additional keyword arguments.

`add_task`

```python
def add_task(self, task)
```

Description

Adds a task to the task queue.

Parameters

- `task` (_type_): The task to be added to the queue.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

agent = Agent()
swarm = SwarmNetwork(agents=[agent])
```

```
swarm.add_task("task")
```

`async_add_task`

```python
async def async_add_task(self, task)
```

Description

Adds a task to the task queue asynchronously.

Parameters

- `task` (_type_): The task to be added to the queue.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

agent = Agent()
swarm = SwarmNetwork(agents=[agent])
await swarm.async_add_task("task")
```

`run_single_agent`

```python
def run_single_agent(self, agent_id, task: Optional[str], *args, **kwargs)
```

Description

Runs a task on a specific agent by ID.

Parameters

- `agent_id` (_type_): The ID of the agent.
- `task` (str, optional): The task to be executed by the agent.
- `*args`: Additional positional arguments.
- `**kwargs`: Additional keyword arguments.

Returns

- `_type_`: The output of the agent running the task.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent = Agent(
```

```
agent_name="Financial-Analysis-Agent",
llm=model,
max_loops="auto",
autosave=True,
dashboard=False,
verbose=True,
streaming_on=True,
interactive=True,
# interactive=True, # Set to False to disable interactive mode
saved_state_path="finance_agent.json",
# tools=[Add your functions here# ],
# stopping_token="Stop!",
# interactive=True,
# docs_folder="docs", # Enter your folder name
# pdf_path="docs/finance_agent.pdf",
# sop="Calculate the profit for a company.",
# sop_list=["Calculate the profit for a company."],
user_name="swarms_corp",
# # docs=
# # docs_folder="docs",
retry_attempts=3,
# context_length=1000,
# tool_schema = dict
context_length=200000,
# agent_ops_on=True,
# long_term_memory=ChromaDB(docs_folder="artifacts"),
```

)

swarm = SwarmNetwork(agents=[agent])

result = swarm.run_single_agent(agent.id, "task")

```

### `run_many_agents`

```python

def run_many_agents(self, task: Optional[str] = None, *args, **kwargs) -> List

```

Description

Runs a task on all agents in the pool.

Parameters

- `task` (str, optional): The task to be executed by the agents.

- `*args`: Additional positional arguments.

- `**kwargs`: Additional keyword arguments.

Returns

- `List`: The output of all agents running the task.

Example

```python

```
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent = Agent(
    agent_name="Financial-Analysis-Agent",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here#],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
```

```
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)


# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
```

```
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)



swarm = SwarmNetwork(agents=[agent1, agent2])
results = swarm.run_many_agents("task")
```

`list_agents`

```python
def list_agents(self)
```

Description

Lists all agents in the pool.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
```

```
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)

swarm = SwarmNetwork(agents=[agent])
swarm.list_agents()
```

`get_agent`

```python
def get_agent(self, agent_id)
```

Description

Gets an agent by ID.

Parameters

- `agent_id` (_type_): The ID of the agent to retrieve.

Returns

- `_type_`: The agent with the specified ID.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
```

```
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)

swarm = SwarmNetwork(agents=[agent])
retrieved_agent = swarm.get_agent(agent.id)
```

`add_agent`

```python
def add_agent(self, agent: Agent)
```

Description

Adds an agent to the agent pool.

Parameters

- `agent` (_type_): The agent to be added to the pool.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
```

```
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)

swarm = SwarmNetwork(agents=[])
swarm.add_agent(agent)
```

`remove_agent`

```python
def remove_agent(self, agent_id)
```

Description

Removes an agent from the agent pool.

Parameters

- `agent_id` (_type_): The ID of the agent to be removed.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
```

```
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)

swarm = SwarmNetwork(agents=[agent])
swarm.remove_agent(agent.id)
```

`

async_remove_agent`

```python
async def async_remove_agent(self, agent_id)
```

Description

Removes an agent from the agent pool asynchronously.

Parameters

- `agent_id` (_type_): The ID of the agent to be removed.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
```

```
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)

swarm = SwarmNetwork(agents=[agent])
await swarm.async_remove_agent(agent.id)
```

`scale_up`

```python
def scale_up(self, num_agents: int = 1)
```

Description

Scales up the agent pool by adding new agents.

Parameters

- `num_agents` (int, optional): The number of agents to add. Defaults to 1.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
    verbose=True,
```

```
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)

swarm = SwarmNetwork(agents=[agent])
swarm.scale_up(2)
```

`scale_down`

```python
def scale_down(self, num_agents: int = 1)
```

Description

Scales down the agent pool by removing agents.

Parameters

- `num_agents` (int, optional): The number of agents to remove. Defaults to 1.

Example

```python
from swarms.structs.agent import Agent
from swarms.structs.swarm_net import SwarmNetwork

# Initialize the agent
agent2 = Agent(
    agent_name="ROTH-IRA-AGENT",
    system_prompt=ESTATE_PLANNING_AGENT_SYS_PROMPT,
    llm=model,
    max_loops="auto",
    autosave=True,
    dashboard=False,
```

```
    verbose=True,
    streaming_on=True,
    interactive=True,
    # interactive=True, # Set to False to disable interactive mode
    saved_state_path="finance_agent.json",
    # tools=[Add your functions here# ],
    # stopping_token="Stop!",
    # interactive=True,
    # docs_folder="docs", # Enter your folder name
    # pdf_path="docs/finance_agent.pdf",
    # sop="Calculate the profit for a company.",
    # sop_list=["Calculate the profit for a company."],
    user_name="swarms_corp",
    # # docs=
    # # docs_folder="docs",
    retry_attempts=3,
    # context_length=1000,
    # tool_schema = dict
    context_length=200000,
    # agent_ops_on=True,
    # long_term_memory=ChromaDB(docs_folder="artifacts"),
)



swarm = SwarmNetwork(agents=[agent])
swarm.scale_down(1)
```

```

### `run`

#### Description

Runs the swarm network, starting the FastAPI application.

#### Example

```python

import os

from dotenv import load_dotenv

# Import the OpenAIChat model and the Agent struct

from swarms import Agent, OpenAIChat, SwarmNetwork

# Load the environment variables

load_dotenv()

# Get the API key from the environment

api_key = os.environ.get("OPENAI_API_KEY")

# Initialize the language model

llm = OpenAIChat(

```
    temperature=0.5,
    openai_api_key=api_key,
)

Initialize the workflow
agent = Agent(llm=llm, max_loops=1, agent_name="Social Media Manager")
agent2 = Agent(llm=llm, max_loops=1, agent_name=" Product Manager")
agent3 = Agent(llm=llm, max_loops=1, agent_name="SEO Manager")

# Load the swarmnet with the agents
swarmnet = SwarmNetwork(
    agents=[agent, agent2, agent3],
)

# List the agents in the swarm network
out = swarmnet.list_agents()
print(out)

# Run the workflow on a task
out = swarmnet.run_single_agent(
    agent2.id, "Generate a 10,000 word blog on health and wellness."
)
print(out)
```

```
# Run all the agents in the swarm network on a task
out = swarmnet.run_many_agents("Generate a 10,000 word blog on health and wellness.")
print(out)
```

Additional Information and Tips

- **Error Handling**: Make use of try-except blocks to handle potential errors when adding tasks, running tasks, and managing agents.
- **Logging**: Enable logging to track the activity and status of the swarm network.
- **API**: The provided API allows for easy interaction with the swarm network and can be extended as needed.
- **Asynchronous Operations**: Utilize the asynchronous methods for non-blocking operations, especially in a production environment.
- **Scaling**: Adjust the scaling thresholds (`idle_threshold` and `busy_threshold`) based on the specific needs and workload patterns.

References and Resources

- [Python Queue Documentation](https://docs.python.org/3/library/queue.html)
- [Threading in Python](https://docs.python.org/3/library/threading.html)
- [FastAPI Documentation](https://fastapi.tiangolo.com/)
- [Tenacity Documentation](https://tenacity.readthedocs.io/en/latest/)

By following this documentation, users can effectively manage and utilize the `SwarmNetwork` class to handle dynamic workloads and maintain an efficient pool of agents.