

Scheduler and Ray Basics

Scheduler and Ray Basics

Два минимальных скрипта для проверки, что инфраструктура (Ray-кластер, планировщик) вообще работает — без них имеет смысл начинать отладку любого другого примера, если что-то не запускается.

`examples/ray_example.py` —
????????? ?????? ? Ray-???????????

```
ray.init(address="ray://localhost:10001")

print(f"{len(ray.nodes())} nodes, {ray.cluster_resources()['CPU']} CPU total")

@ray.remote
def f():
    time.sleep(0.001)
    return socket.gethostbyname(socket.gethostname())

object_ids = [f.remote() for _ in range(10000)]
ip_addresses = ray.get(object_ids)

for ip_address, num_tasks in Counter(ip_addresses).items():
    print(f"{num_tasks} tasks on {ip_address}")
```

Подключается к Ray Client Server по адресу `ray://localhost:10001` (порт `HEADNODEPORT`), печатает число узлов и суммарные CPU-ресурсы кластера, затем запускает 10 000 тривиальных удалённых задач и показывает, сколько из них выполнилось на каждом IP-адресе — наглядно демонстрирует, что worker'ы реально участвуют в распределении нагрузки, а не всё выполняется на одном узле.

Запуск: `python examples/ray_example.py` (с хоста, где проброшен порт 10001 — по умолчанию так и есть в `docker-compose.yml`).

examples/scheduler/test.py — ???????? ???? ???? ??????????

```
import time
for i in range(30):
    print(i)
    time.sleep(1)
```

Ничего специфичного для mlumina — просто скрипт на ~30 секунд, который печатает числа. Предназначен не для прямого запуска, а как `entrypoint` для сервиса scheduler (`mlumina.scheduler`, порт 8264, см. книгу «Документация MLumina»): удобная минимальная нагрузка, чтобы проверить весь путь постановки задачи в очередь, её запуска планировщиком и завершения, не тратя время на реальное обучение модели.

Пример отправки этой задачи в планировщик (аналогично `ray_wrapper/examples/*/schedule_*.py`, см. следующие страницы книги):

```
import requests

requests.post(
    "http://127.0.0.1:8264/add_task",
    json={
        "name": "scheduler_smoke_test",
        "entrypoint": "python examples/scheduler/test.py",
        "runtime_env": {"working_dir": "/home/ray/src/mlumina"},
        "priority": 1,
        "cpu": 1,
        "gpu": 0,
        "memory": 1,
        "timeout": 60,
    },
    timeout=60,
)
```

После постановки в очередь прогресс можно отслеживать через `GET /get_running_tasks_list` или веб-дашборд планировщика (`http://<host>:8264/`).

Revision #1

Created 2026-09-23 06:54:45 UTC by Маслов Алексей

Updated 2026-09-23 06:54:46 UTC by Маслов Алексей