

一、测试目标

验证服务器在高并发场景下，能否满足“单次数据写入响应时间 $\leq 200\text{ms}$ ”的性能要求，同时确保服务稳定无故障。

二、核心测试思路

通过 独立测试表 + 极简接口 隔离业务数据库，避免测试数据干扰；

本地验证通后部署服务器

用 Locust 模拟多设备并发请求，精准测服务器接口承载能力。

三、详细实施步骤

步骤	操作内容	关键说明
1	本地开发：独立接口 + 测试表	脱离业务数据库设计，仅保留压测必需功能，本地快速跑通验证
2	服务器部署：配置 + 启动服务	修改数据库连接适配云端，用生产级服务器（Waitress）启动，开放端口
3	压力测试：Locust 本地远程压测	模拟多设备并发请求，监控核心性能指标，验证是否达标
4	结果验证 + 清理	核对压测指标与数据库记录，测试完成后删除独立测试表，无残留

四、核心设计细节

1. 独立测试表设计

为避免干扰原有业务数据库，新建独立测试表 `test_upload`，仅保留压测必需字段，结构极简：

```
class TestUpload(Base):
    __tablename__ = "test_upload" # 测试表名，测完直接删
    id = Column(Integer, primary_key=True, autoincrement=True)
    device_id = Column(String(50), nullable=False) # 模拟设备ID
    test_data = Column(Float, nullable=False) # 测试数据
    upload_time = Column(DateTime, default=datetime.now, nullable=False) # 上传时间
```

2. 数据库连接配置

复用现有数据库 `ystc_energy`，但通过独立表隔离，连接参数优化高并发支持：

```
# 数据库连接
# 格式: mysql+pymysql://用户名:密码@localhost:端口/数据库名?charset=utf8mb4
DATABASE_URI = "mysql+pymysql://root:root115@localhost:3306/ystc_energy?charset=utf8mb4"
```

```
# 数据库引擎
engine = create_engine(
    DATABASE_URI,
    pool_size=50,
    max_overflow=100,
    pool_pre_ping=True,
    echo=False # 关闭SQL打印，提升速度
)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

# 数据库初始化
def init_db():
    try:
        Base.metadata.create_all(bind=engine) # 自动创建test_upload表
        log.info("测试表test_upload创建成功（已存在则跳过）")
        # 验证连接
        db = SessionLocal()
        db.query(TestUpload).first()
        db.close()
        log.info("数据库连接成功")
    except Exception as e:
        log.error(f"数据库初始化失败: {str(e)}")
        raise e
```

3. 接口设计

设计 2 个接口，通过任务权重分配模拟真实请求比例，确保压测聚焦 “写入接口”：

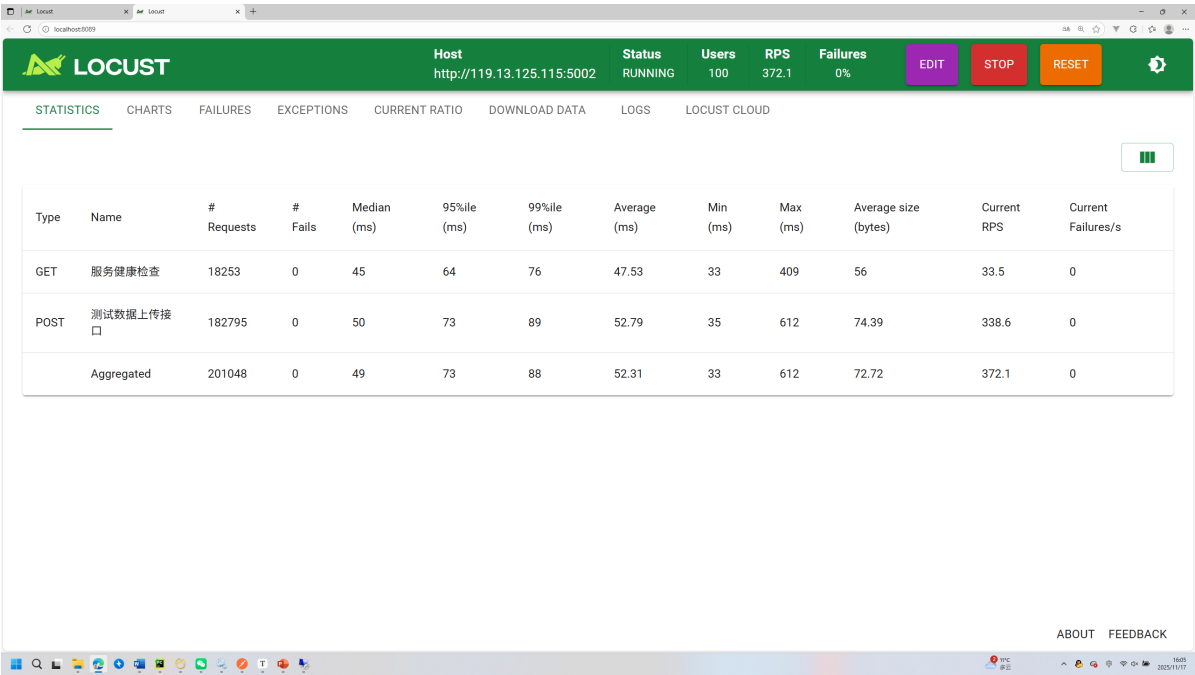
接口名称	接口路径	请求方式	核心功能	任务权重	备注
数据上传接口	/api/test/upload	POST	接收测试数据，写入 test_upload 表	10	核心压测接口（占 90%+ 请求量）
服务健康检查接口	/health	GET	验证服务与数据库连接状态	1	辅助接口（占少量请求，模拟真实监控场景）

4. 压测配置

通过任务权重控制请求比例，模拟多设备高并发：

- 核心压测任务：调用 /api/test/upload 接口（权重 10）
- 辅助任务：调用 /health 接口（权重 1）
- 模拟设备数：初始 100 台，逐步加压至峰值
- 请求间隔：0.05-0.1 秒 / 次（单设备 10-20 次 / 秒请求）

五、测试结果



- 核心指标：
 - 并发用户数（Users）：100 → 模拟 100 台设备同时发请求；
 - 每秒请求数（RPS）：372.1 → 服务器每秒能处理 372.1 次请求，吞吐量优异；
 - 失败率（Failures）：0% → 所有请求均成功，服务稳定性拉满；
- 接口响应时间：
 - 测试数据上传接口（核心压测接口）：中位数 50ms、平均 52.79ms，95% 分位 73ms → 远低于“200ms 内写入”的目标；
 - 服务健康检查接口：中位数 45ms、平均 47.53ms → 辅助接口响应也极快；

结论：在 100 并发压力下，服务器接口响应速度、吞吐量、稳定性均远超预期，完全满足性能要求。

2025-11-17 15:12:03,794	Serving on http://0.0.0.0:5002
2025-11-17 15:44:08,697	Task queue depth is 1
2025-11-17 15:44:08,702	Task queue depth is 1
2025-11-17 15:44:08,704	Task queue depth is 1
2025-11-17 15:44:43,213	Task queue depth is 1
2025-11-17 15:45:03,497	Task queue depth is 1
2025-11-17 15:45:03,502	Task queue depth is 1
2025-11-17 15:45:28,770	Task queue depth is 1
2025-11-17 15:45:28,777	Task queue depth is 1
2025-11-17 15:45:40,189	Task queue depth is 1
2025-11-17 15:45:40,197	Task queue depth is 1
2025-11-17 15:45:40,203	Task queue depth is 1
2025-11-17 15:46:00,333	Task queue depth is 1
2025-11-17 15:46:00,336	Task queue depth is 2
2025-11-17 15:46:00,342	Task queue depth is 2
2025-11-17 15:46:00,343	Task queue depth is 2
2025-11-17 15:46:00,347	Task queue depth is 2
2025-11-17 15:46:17,992	Task queue depth is 1
2025-11-17 15:46:18,186	Task queue depth is 1
2025-11-17 15:46:18,192	Task queue depth is 1
2025-11-17 15:46:42,721	Task queue depth is 1
2025-11-17 15:46:42,735	Task queue depth is 2
2025-11-17 15:46:45,120	Task queue depth is 1
2025-11-17 15:46:45,128	Task queue depth is 2
2025-11-17 15:46:45,131	Task queue depth is 3
2025-11-17 15:46:54,393	Task queue depth is 1
2025-11-17 15:46:54,395	Task queue depth is 2
2025-11-17 15:47:07,353	Task queue depth is 1
2025-11-17 15:47:23,206	Task queue depth is 1
2025-11-17 15:47:34,865	Task queue depth is 1
2025-11-17 15:47:34,867	Task queue depth is 1
2025-11-17 15:47:42,333	Task queue depth is 1
2025-11-17 15:47:42,343	Task queue depth is 1
2025-11-17 15:48:00,669	Task queue depth is 1
2025-11-17 15:48:00,671	Task queue depth is 2
2025-11-17 15:48:00,673	Task queue depth is 3
2025-11-17 15:48:05,380	Task queue depth is 1
2025-11-17 15:48:05,380	Task queue depth is 2
2025-11-17 15:48:08,995	Task queue depth is 1

- 数值范围：始终在 1-3 之间波动 → 代表“等待服务器处理的请求数量”；
- 意义：队列深度越小，说明请求处理越及时。1-3 的极低数值，意味着服务器处理请求的速度**远大于**请求进来的速度，几乎没有请求积压；

结论：结合 Locust 的指标，进一步证明服务器在当前压力下“处理能力充足、无性能瓶颈”，还有很大的性能余量。

id	device_id	test_data	upload_time
82001	locust_device_00022	445.82	2025-11-17 15:49:30
82002	locust_device_00004	527.67	2025-11-17 15:49:30
82003	locust_device_00002	772.5	2025-11-17 15:49:30
82004	locust_device_00002	813.01	2025-11-17 15:49:30
82005	locust_device_00005	392.09	2025-11-17 15:49:30
82006	locust_device_00024	686.83	2025-11-17 15:49:30
82007	locust_device_00003	673.3	2025-11-17 15:49:30
82008	locust_device_00015	715.02	2025-11-17 15:49:30
82009	locust_device_00025	347.95	2025-11-17 15:49:30
82010	locust_device_00004	333.34	2025-11-17 15:49:30
82011	locust_device_00005	878.4	2025-11-17 15:49:30
82012	locust_device_00015	537.2	2025-11-17 15:49:30
82013	locust_device_00026	519.26	2025-11-17 15:49:30
82014	locust_device_00026	310.17	2025-11-17 15:49:30
82015	locust_device_00012	689.14	2025-11-17 15:49:30
82016	locust_device_00025	886.16	2025-11-17 15:49:30
82017	locust_device_00026	52.83	2025-11-17 15:49:30
82018	locust_device_00013	664.16	2025-11-17 15:49:30
82019	locust_device_00021	490.9	2025-11-17 15:49:30
82020	locust_device_00011	29.78	2025-11-17 15:49:30
82021	locust_device_00011	411.69	2025-11-17 15:49:30