GPU 计算平台

设备管理协议(udp 通信)

1.第一次请求信息(将管理器信息传递给计算设备)

```
{"managerinfo":[
      {"ip":"192.168.0.1"},
      {"port":"8088"}]
}
```

协议说明:

ip:　　管理器 IP 地址

port:　　管理器端口

2.设备列表反馈协议(只有在设备发生变更才反馈)

```
{"txdevlist":[
      {"txid":"tx00001"},
      {"txid":"tx00002"}
     ]
}
```

协议说明:

txid:　　设备 id

3.设备状态反馈协议(间隔一直反馈)

```
{"txdevlistinfo":[
      {"txid":"tx00001","state":1,"cpu":50,"gpu":60,"mem":30},
      {"txid":"tx00002","state":0,"cpu":53,"gpu":68,"mem":80}
     ]
}
```

协议说明:

txdevlistinfo(节点列表包含开机和非开机 state 0 的)

txid　　设备 id(数据来源设备端定义)

state为设备显示状态(1 开电 0 关机) state 为 0 时 后面 cpu 等参数可随意填

cpu　　cpu 消耗值　%比

gpu　　gpu 消耗值　%比

mem　　内存消耗值 %比

4.管理设备工作协议

```
{"txdevwork":{"txid":"tx00001","state":1}}
```

协议说明:

txid　　设备 id(数据来源设备端定义)

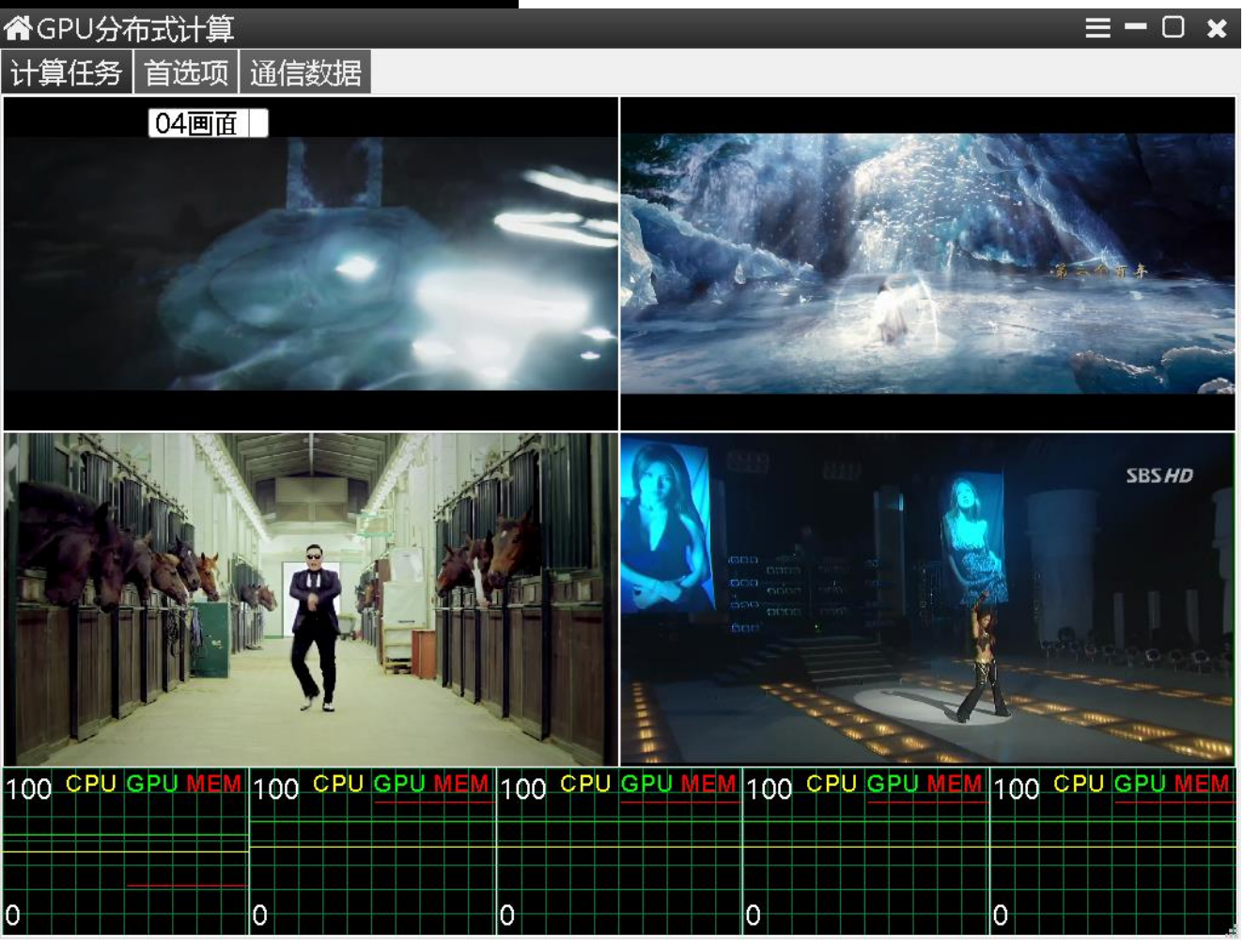
state1 执行开电/0 执行关电

5.计算服务状态的系统配置

在界面首选项-计算模块设备 UDP 状态表-启动 UDP 服务

配置好首选项后建立好通信后下面状态面板对应的 CPU/GPU/MEM 会有实时状态变化。

CPU-黄色 GPU-绿色 内存-红色



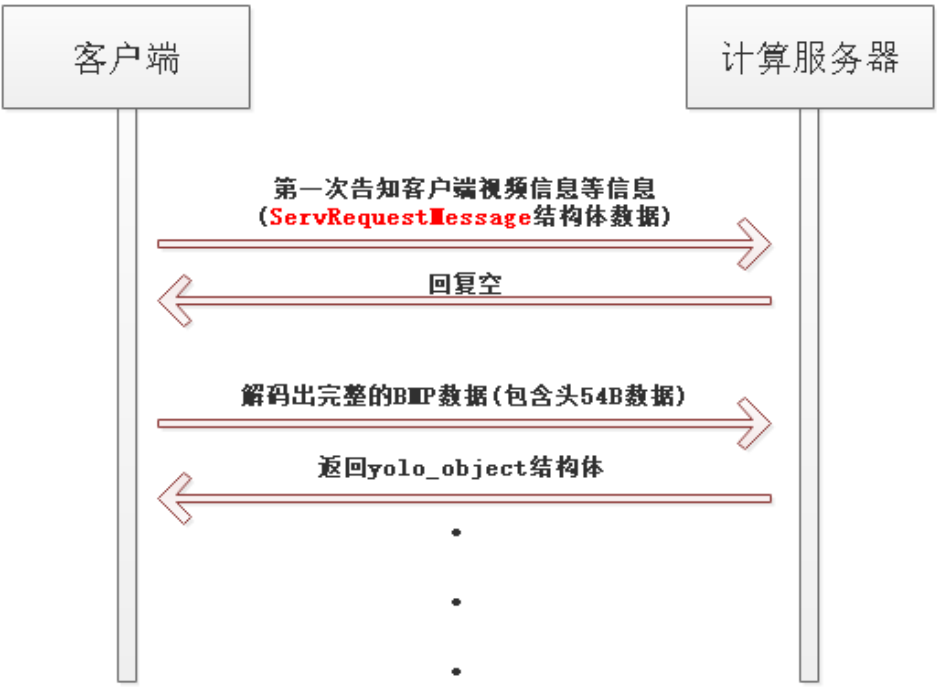
任务作业协议(tcp 通信)

1. 待定格式

```
**********请求服务**********
typedef enum
{
        CPU_UNIT = 0,
        GPU_UNIT,
        CPU_GPU_UNIT
}COMPUTE_UNIT_TYPE;

typedef enum
{
        OBJECT_DETECTION = 0,
        OBJECT_CLASSIFY
} COMPUTE_SERV_TYPE;

/* 请求服务帧格式 */
typedef struct ServRequestMessage
{
        COMPUTE_UNIT_TYPE compute_unit_type;
        COMPUTE_SERV_TYPE compute_serv_type;
        int frame_width;
        int frame_height;
}SERV_REQUEST_MESSAGE;

/**********计算结果回传**********/
#define YOLO_MAX_OBJ_NUM    64

typedef struct
{
        int class_id;
        int x, y, w, h;
        float score;
}yolo_rect;

/*计算结果帧格式*/
typedef struct
{
        int num;
        yolo_rect obj_rect[YOLO_MAX_OBJ_NUM];
}yolo_object;
```

2. 图传 TCP 通信过程

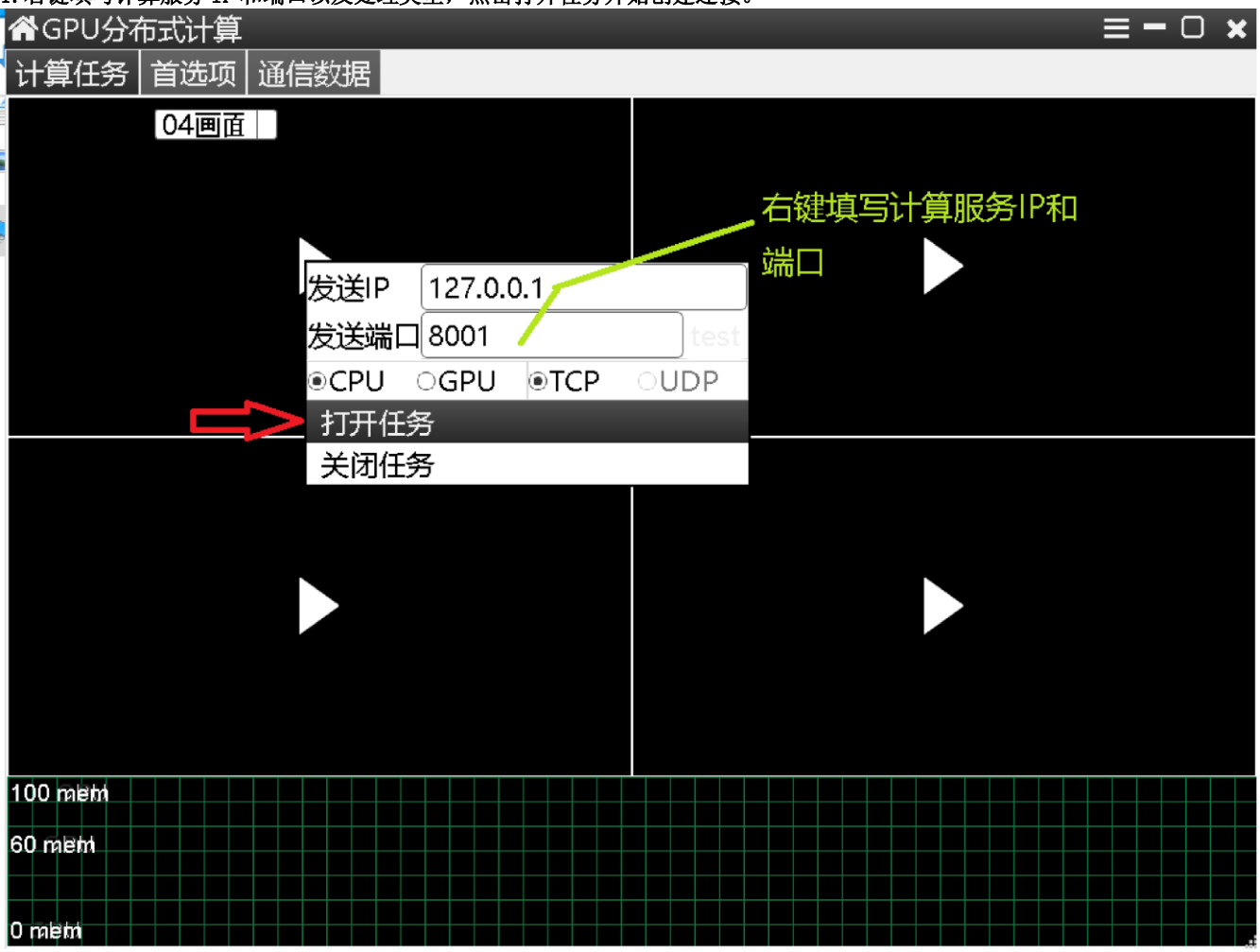
数据发送流程如下：



计算服务器接收端 recv 到的全部数据 datas (`recv(Sock, (char*)&datas, len , 0);`)
可参考 stb_image 库的 `stbi_load_from_memory(datas, len, &w, &h, &c, 0);`
把 BMP 图片解析出来。

下面是客户端演示例子

1. 右键填写计算服务 IP 和端口以及处理类型，点击打开任务开始创建连接。



2. 连接成功后解码视频帧画面通过协议传输给计算服务，返回识别矩形坐标会绘制在显示区域。