Node.js的一个远程服务调用客户代理v 0.5

——@tuniu/remote-client模块的应用

无线中心/网站研发部
作者：郭君龙,黄一君

@since 2016 / 12 / 01

实现背景

★ 途牛远程服务入出参个性化处理

★ 可配置化调用管理

★ 实现缓存机制

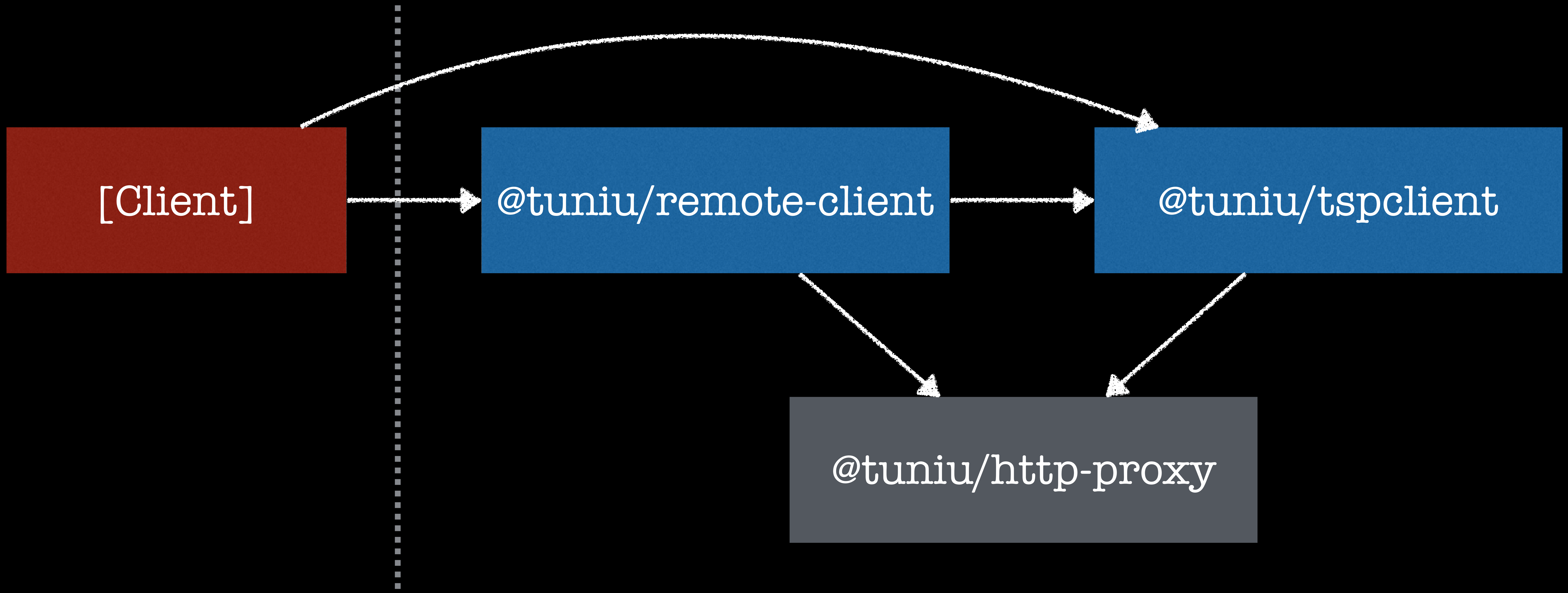
★ 集成TSP服务客户代理

★ 升级旧有实现进一步提高可用性

涵盖内容

- ★ 远程服务调用基础用法.
- ★ TSP客户端用法.
- ★ 远程服务调用与TSP客户端集成.
- ★ 集成到基于LightMvc的项目中.
- ★ 开发与调试.

模块调用关系


[Client]
@tuniu/remote-client
@tuniu/tspclient
@tuniu/http-proxy

第一个远程调用

```
'use strict';
const RemoteClient = require('@tuniu/remote-client');
const co = require('co');
let config = {
    services: {
        'TBS.city.CityContoller.queryBookingCity': {
            serverInfo: {
                baseUrl: 'http://10.10.30.109:8080',
                path: '/tbs/city/booking-city/query'
            }
        }
    }
}
let client = RemoteClient(config);
co(function* () {
    let resp = yield client.invoke('TBS.city.CityContoller.queryBookingCity');
    console.log(resp);
});
```

带参数的远程调用

```
// GET或POST参数
let anotherSrvName = 'another-service';
let params = {
    name: 'simple-life',
};
let resp = yield client.invoke(anotherSrvName, params);
console.log(resp);

// 传递headers（也可在配置里设置静态值）
let options = {
   headers: { 'mobile-device': 'HTC 850' }
};
let resp = yield client.invoke(anotherSrvName, params, options);
console.log(resp);
```

入出参数的编解码

```
let config = {
  services: {
    'TBS.city.CityContoller.queryBookingCity': {
      serverInfo: {
        baseUrl: 'http://10.10.30.109:8080',
        path: '/tbs/city/booking-city/query'
      }
      inputEncoding: 'rest',
      outputEncoding: 'base64'
}}};
```

- ★ **inputEncoding:**
  - ★ base64
  - ★ encodeURIComponent
  - ★ rest
  - ★ querystring #不指定时默认值
- ★ **outputEncoding:**
  - ★ **base64** #不指定时不做任何处理

① 假设总是返回**json**或者**base64**编码过的**json.**

② 不指定，不做**base64**解码预处理**.**

③ 如果返回**http**头**tn-res-data-type == json/base64**，自动**base64**解码**.**

GET入参编码详解

* **base64**

http://api.tuniu.org/customer?eyJuYW1lIjoiVG9tSGFua3MiLCJ3ZWlnaHQiOiI3OCJ9

* **encodeURIComponent**

http://api.tuniu.org/customer?d=%7B%22name%22%3A%22TomHanks%22%2C%22weight%22%3A%2278%22%7D

* **rest**

http://api.tuniu.org/customer/{"name":"TomHanks","weight":"78"}
http://api.tuniu.org/customer/{name}/{weight} => http://api.tuniu.org/customer/TomHanks/78

* **json**

http://api.tuniu.org/customer/{"name":"TomHanks","weight":"78"}

* **querystring**

http://api.tuniu.org/customer?name=TomHanks&weight=78

```
let path = 'http://api.tuniu.org/customer';
let params = {
   name: 'TomHanks',
   weight: '78'
};
```

POST及其他HTTP方法入参

- POST方法只接受inputEncoding=base64或不填.

  ① 不填：默认处理params的方式和通常网页表单一致.

  ② base64：params对象做base64编码作为http body传入

- 有限支持DELETE和PUT，但未久经考验.

Restful风格URL

- 无论GET/POST请求，URL都可能要求遵从Restful风格
- 前文提到的rest，是粗糙的参数处理，正规方法是为调用方法options（第3个参数）提供pathParams属性

```
let config = {
    services: {
        'another-service': {
            serverInfo: {
                baseUrl: 'http://api.t.org',
                path: '/phone/query/{brand}/{model}'
            }}}};
… …
let options = {
    pathParams: {model: 'xiaomi5', brand: 'MI'}
}
let resp = yield client.invoke(anotherSrvName,
params, options);
```

上面示例实际发起调用的目标URL是：http://api.t.org/phone/query/MI/xiaomi5

带缓存的配置

```
let config = {
   services: {
      'TBS.city.CityContoller.queryBookingCity': {
         serverInfo: {
            baseUrl: 'http://10.10.30.109:8080',
            path: '/tbs/city/booking-city/query'
         }
         cache: {
              shortTermCache: {
                 expired: 10 * 60 * 1000, // 单位：ms
                 enabled: false
              },
              longTermCache: {
                 expired:  0,  // 单位：ms, 0表示永久保存.
                 enabled: true
              }
            }
         }
}}};
```

★ 仅GET缓存

★ shortTermCache：通常的缓存

★ longTermCache：仅在网络错误或500返回使用的缓存

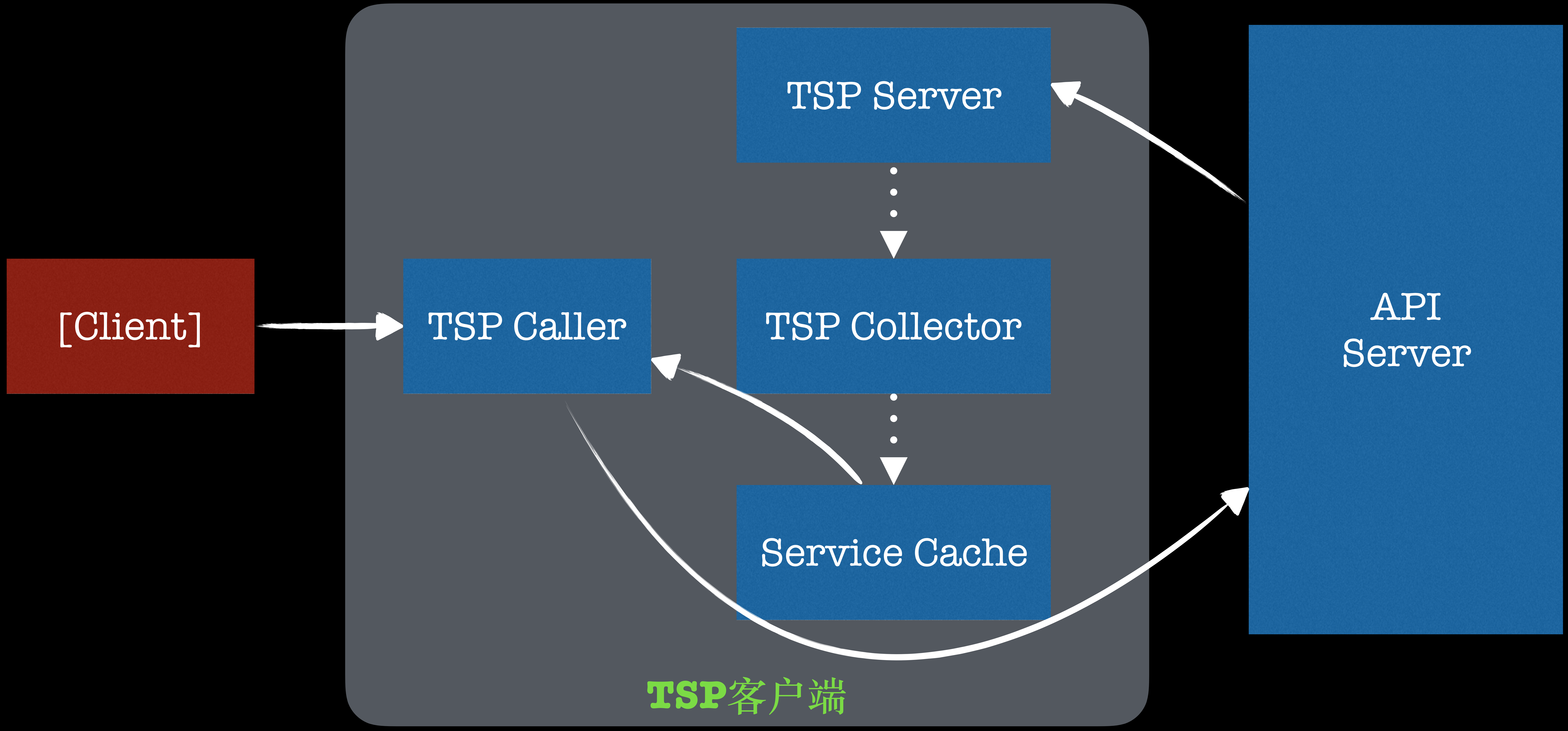
Service全配置

```
let config = {
  services: {
    'TBS.city.CityContoller.queryBookingCity': {
       channel: 'TSP', // 默认为非TSP, 可选
      tspName: 'TBS.city.CityContoller.queryBookingCity', //默认取key为tspName
       serverInfo: { // 当channel==TSP, 该节点无效.
           baseUrl: 'http://10.10.30.109:8080',
           path: '/tbs/city/booking-city/query'
      }
       inputEncoding: 'rest',
       outputEncoding: 'base64’,
       method: 'GET', // 默认为GET
       timeout: 89000 // 默认取全局配置.
       headers: { from: 'mobile' },
       cache: {
          shortTermCache: {
              expired: 10 * 60 * 1000,  // 单位ms, 此例为10分钟.
              enabled: false}
          longTermCache: {
             expired:  0, // 单位ms, 0表示永久保存.
             enabled: true}
      }
  }}};
```

全局完整配置

```
let config = {
    cache: {
        storageType: 'redis',
        storage: {
            redis: {
                host: 'hx-ares-master.redis.tuniu.org',
                port: 26514,
            }
        },
        shortTermCache: {
            expired: 100 * 60 * 1000, // 单位ms
            enabled: false
        },
        longTermCache: {
            expired: 24 * 60 * 60 * 1000, // 单位ms, 0表示永久保存
            enabled: false
        },
        keyPrefix: 'ACHILLES:REMOTE_SERVICES:CACHE:'
    },
    httpTimeout: 20 * 1000,
    services: { /* 此处省略..... */}
  }}};
```

TSP客户端


TSP Server
[Client]
TSP Caller
TSP Collector
API Server
Service Cache
TSP客户端

TSP客户端初始化

```
let remoteServices = {
   'HTE.Traffic.StrategyController.lowprice': { channel: 'TSP' },
   'HTE.Traffic.StrategyController.query': { channel: 'TSP' }
};
let tspConfig = {
   registerServers: ['bj-tspregister-master.tuniu.org:9989'],
   monitor: {
      servers: [ // TSP监控中心UDP服务器主备IP端口信息配置
         {ip: 'bj-tsp-flume-03.tuniu.org', port: 11311}
      ],
   },
   subSystem: 'MOB',              // 当前使用者隶属子系统
   module: 'MOB_ACHILLES',  // 当前子系统模式
   storage: {
      redis: {
         host: 'hx-ares-master.redis.tuniu.org',
         port: 26514,
      }
   },
   services: [
      'MOB.PORTAL.ChannelController.getChannelIndex',
      'TBS.city.CityContoller.queryBookingCity'
   ]
};
```

```
let tspClient = require('@tuniu/tspclient')(tspConfig);
tspClient.collector = tspClient.ServiceCollector();
if (remoteServices) {
     tspClient.collector.
                     addRemoteServices(remoteServices);
}
tspClient.collector.start(); // 常驻内存

// 调用tsp服务句柄
tspClient.caller = tspClient.ServiceCaller();
```

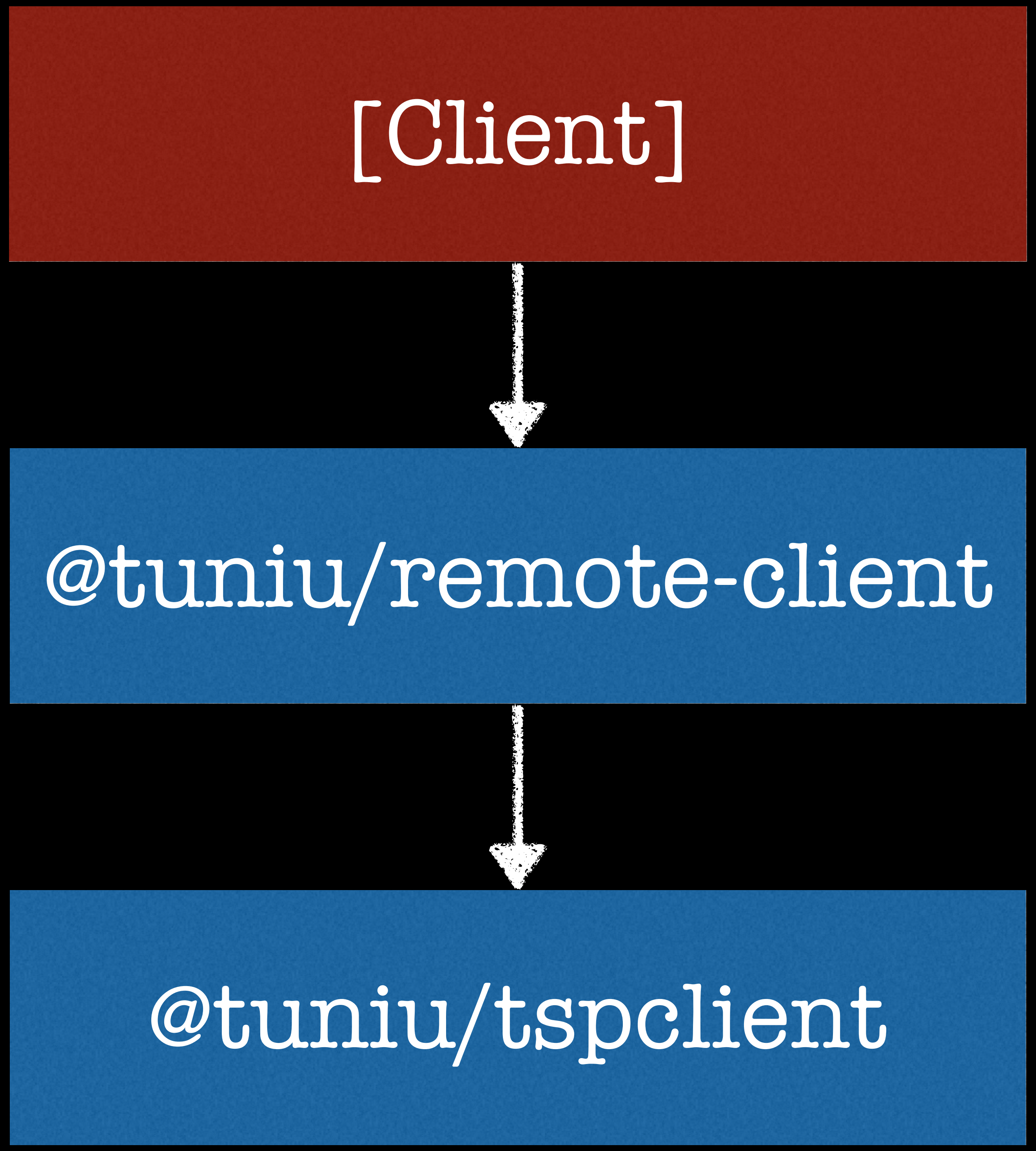
```
 _____              ___ _ _            _   
/__   \___ _ __    / __| (_) ___ _ __ | |_ 
  / /\/ __| '_ \  / /  | | |/ _ \ '_ \| __|
 / /  \__ \ |_) |/ /___| | |  __/ | | | |_ 
 \/   |___/ .__/\____/|_|_|\___|_| |_|\__|
          |_|                             
```

第一个TSP调用

```
let tspName = 'TBS.city.CityContoller.queryBookingCity';
let params = {
    name: 'Someone'
}
let options = {
    inputEncoding: 'base64',
   // outputEncoding: ‘base64',
   // headers: { }
}
let response = yield tspClient.caller.invoke(tspName, params, options);

console.log(response);
```

远程服务客户代理调用tspclient



```
'use strict';
const RemoteClient = require('@tuniu/remote-client');
let config = {
   services: {
      'TBS.city.CityContoller.queryBookingCity': {
         channel: 'TSP',
      }
   }
}
let srvName = ‘TBS.city.CityContoller.queryBookingCity';
client = RemoteClient(config);
let resp = yield client.invoke(srvName);
console.log(resp);
```

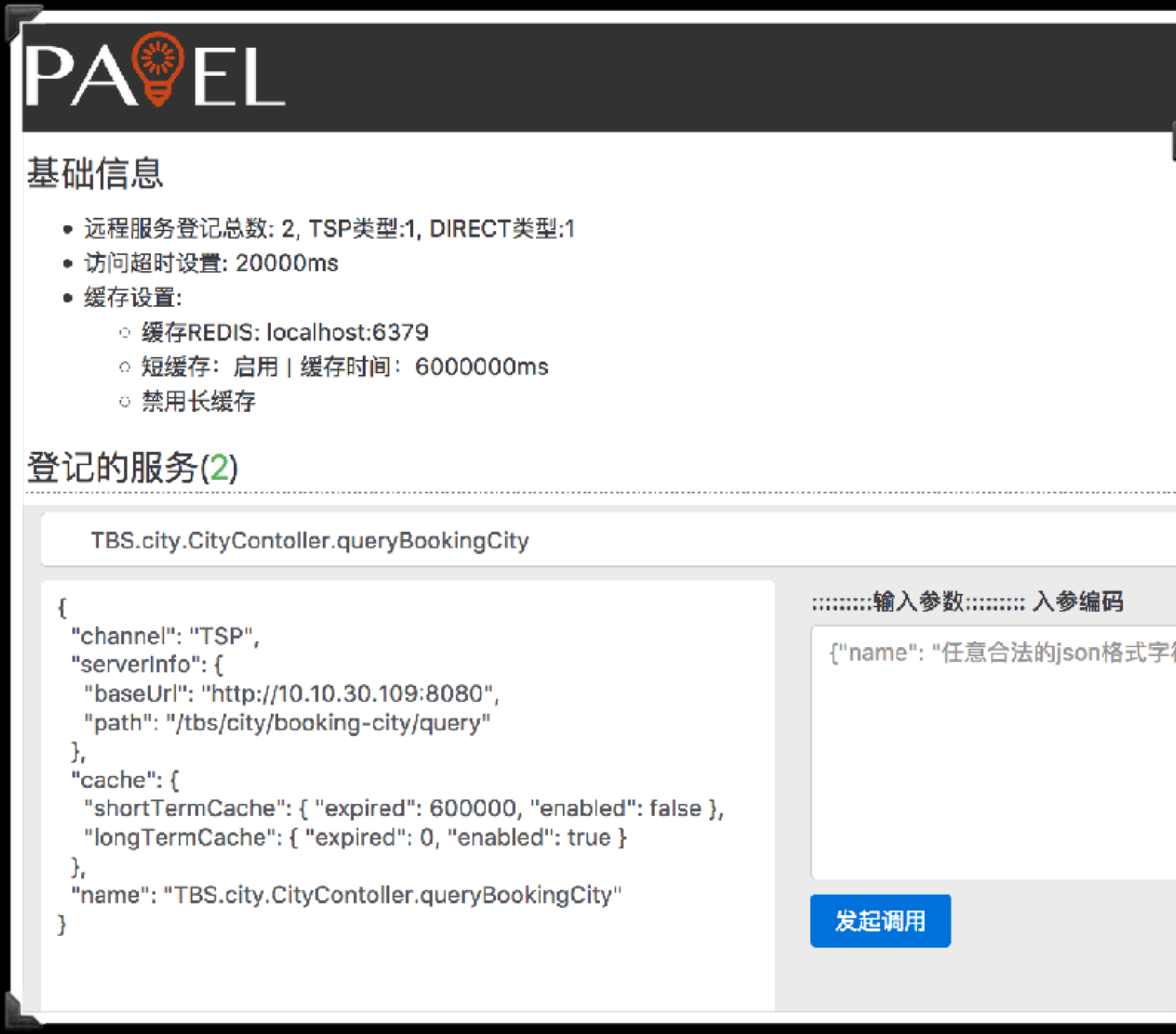
- ★ channel=tsp时，走TSP通道

集成到LIGHT

```
// RemoteService.js 模块文件
'use strict';
const RemoteClient = require('@tuniu/remote-client');
module.exports = class {
  __construct() {
    let options = {
      logLevel: this.ctx.config.params.logLevel
    }
    if(this.ctx.tspClient && this.ctx.tspClient.caller) {
      options.tspCaller = this.ctx.tspClient.caller;
    }
    this.remoteClient = RemoteClient(this.ctx.config.remote, options);
  }

  requestP(service, params, options) {
    return this.remoteClient.invoke(service, params, options);
  }
}
```

开发与调试



PANEL

基础信息

- 远程服务登记总数: 2, TSP类型:1, DIRECT类型:1
- 访问超时设置: 20000ms
- 缓存设置:
  - 缓存REDIS: localhost:6379
  - 短缓存：启用 | 缓存时间：6000000ms
  - 禁用长缓存

登记的服务(2)

TBS.city.CityContoller.queryBookingCity

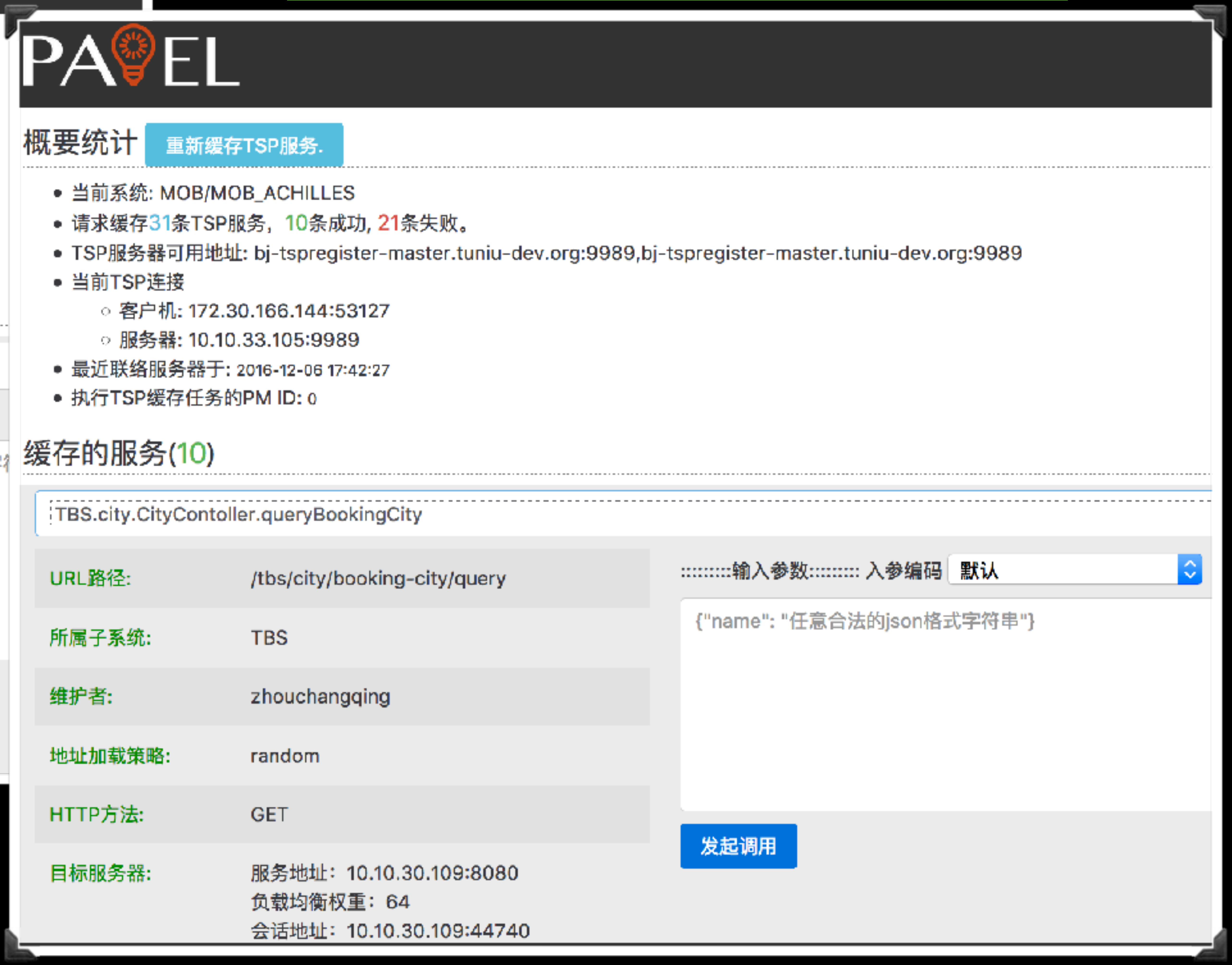
```
{
  "channel": "TSP",
  "serverInfo": {
    "baseUrl": "http://10.10.30.109:8080",
    "path": "/tbs/city/booking-city/query"
  },
  "cache": {
    "shortTermCache": { "expired": 600000, "enabled": false },
    "longTermCache": { "expired": 0, "enabled": true }
  },
  "name": "TBS.city.CityContoller.queryBookingCity"
}
```

::::::::::输入参数:::::::::: 入参编码

{"name": "任意合法的json格式字

http://localhost:5000/panel/remote

http://localhost:5000/panel/tsp



PANEL

概要统计

- 当前系统: MOB/MOB_ACHILLES
- 请求缓存31条TSP服务，10条成功, 21条失败。
- TSP服务器可用地址: bj-tspregister-master.tuniu-dev.org:9989,bj-tspregister-master.tuniu-dev.org:9989
- 当前TSP连接
  - 客户机: 172.30.166.144:53127
  - 服务器: 10.10.33.105:9989
- 最近联络服务器于: 2016-12-06 17:42:27
- 执行TSP缓存任务的PM ID: 0

缓存的服务(10)

TBS.city.CityContoller.queryBookingCity

URL路径:	/tbs/city/booking-city/query
所属子系统:	TBS
维护者:	zhouchangqing
地址加载策略:	random
HTTP方法:	GET
目标服务器:	服务地址：10.10.30.109:8080 负载均衡权重：64 会话地址：10.10.30.109:44740

::::::::::输入参数:::::::::: 入参编码 默认

{"name": "任意合法的json格式字符串"}

资源参考

@tuniu/remote-client
http://wiki.tuniu.org/pages/viewpage.action?pageId=68871311

@tuniu/tspclient
http://wiki.tuniu.org/pages/viewpage.action?pageId=65378359

TODO

- ★ 更好的管理、授权、测试工具
- ★ 二进制文件处理.
- ★ 完整的HTTP方法支持.
- ★ 更好的缓存数据结构.
- ★ redis以外的的缓存.
- ★ 支持HTTPS.
- ★ HTTP 2.0 或非HTTP协议.

加入开发请联系: guojunlong@tuniu.com

THE END