设计文档

所有跟LEPD交互的逻辑，都在backend文件夹的文件里面。

* LepDClient, 是唯一跟LEPD直接通信的类.

它其中的一个方法SendRequest(), 封装了跟LEPD之间的JSONRPC通信，目前我们应该不需要对这个方法有改动，除非我们需要支持带参数的RPC call.

它其他的方法都是通过SendRequest()来调用LEPD的接口函数，来获取相应的结果，并加以基本的解析。

以getCmdFree()为例，它通过调用相应的LEPD方法，从response中拿到result，如下：

total used free shared buffers cached

Mem: 1017788 750688 267100 728 157364 368288

这是原始数据，但是还没有结构化，所以我们对这份数据做基本的处理，得到如下的JSON数据并返回：

{

'total': '1017788',

'shared': '728',

'free': '267100',

'buffers': '157364',

'cached': '368288',

'used': '750688'

}

【问题】：是不是把值的部分转为long型数据更有意义？

LepDClient只负责跟LEPD通信，对原始返回数据做最基本的解析和加工，不做任何业务逻辑的处理。

* MemoryMonitor/CPUMonitor/etc., 这些类处理对应的业务逻辑，他们都是通过LepDClient获取数据，根据业务逻辑的需要对数据进行加工整理，供上层调用。

以MemoryMonitor.getCapacity()为例，这个方法获取内存的配置信息，它通过调用LepDClient.getCmdFree(), 得到如下数据：

{

'total': '1017788',

'shared': '728',

'free': '267100',

'buffers': '157364',

'cached': '368288',

'used': '750688'

}

从中拿total的值，除以1000， 并加上单位的信息，返回如下JSON:

{

‘name’: ‘memory’,

‘capacity’: 1018,

‘unit’: ‘MB’

}

这个数据会被用在前端页面， 至于如何显示，接下来开会的时候我再给大家介绍一下。

几个要点：

1. 每个Python类都被加上了main函数，你们不需要配置Django的环境，可以直接用Python3来跑，可以看到结果。
2. 请注意用python3.
3. 需要大家一起来完善backend的这些类，现在LEPD中已经支持但是LEPDClient中还没有支持的，请大家往LepDClient中添加代码。
4. 我会再创建一个Excel表格，让我们来追踪我们支持的方法。