面向企业的服务器 CPU 采购咨询服务

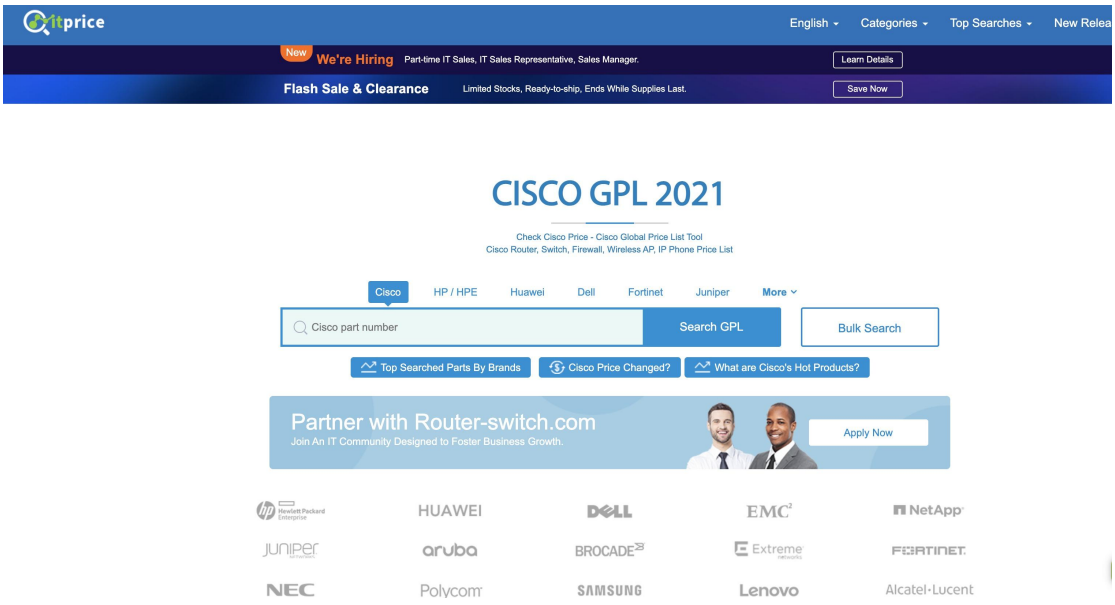
一、摘要

本项目的目标客户为准备采购服务器 CPU 的企业。而企业采购硬件的核心诉求是，在保证性能能够胜任日常业务的情况下，尽量压缩成本。所以我们关注的核心数据为 CPU 的性能及价格。

性能方面的数据，我们可以从 spec 网站上获取。考虑到当代硬件受摩尔定律影响，更新换代速度特别快，而 cpu_2006_result 只包含 2017 年以前的测试数据，所以我们只关注 cpu_2017_results 表的数据，我们只为客户推荐新 CPU。

价格方面的数据，我将从 itprice.com 上爬取。itprice 是一个来自于香港的 IT 设备及配件全球价格查询网站。（如下图所示）

有了这两个维度的数据，我们就能够为企业推荐高性价比的 CPU。



二、项目实现

一、项目开发工具

编程语言：Python
开发软件：Jupyter Notebook

数据库软件：MySQL
数据可视化软件：Tableau

二、数据汇聚整合，提纯加工，存储代码实现

1、spec 数据爬取

```
# 爬虫伪装
headers = {
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'zh-CN,zh;q=0.9',
    'Host': 'spec.org',
    'Referer': 'https://spec.org/cgi-bin/osgresults?conf=cpu2017',
    'Upgrade-Insecure-Requests': '1',
    'User-Agent': UserAgent().random
}
# 定义url
cpu_url_2017 = 'https://spec.org/cgi-bin/osgresults?conf=cpu2017;op=dump;format=csvdump'
cpu_url_2006 = 'https://spec.org/cgi-bin/osgresults?conf=cpu2006;op=dump;format=csvdump'
java_url_2015 = 'https://spec.org/cgi-bin/osgresults?conf=jbb2015;op=dump;format=csvdump'
java_url_2008 = 'https://spec.org/cgi-bin/osgresults?conf=jvm2008;op=dump;format=csvdump'
power_url_2008 = 'https://spec.org/cgi-bin/osgresults?conf=power_ssj2008;op=dump;format=csvdump'

# 数据爬取
cpu_2017_content = requests.get(cpu_url_2017).content
cpu_2017_data = pd.read_csv(io.StringIO(cpu_2017_content.decode('ISO-8859-1')))

cpu_2006_content = requests.get(cpu_url_2006).content
cpu_2006_data = pd.read_csv(io.StringIO(cpu_2006_content.decode('ISO-8859-1')))

java_2015_content = requests.get(java_url_2015).content
java_2015_data = pd.read_csv(io.StringIO(java_2015_content.decode('ISO-8859-1')))

power_2008_content = requests.get(power_url_2008).content
power_2008_data = pd.read_csv(io.StringIO(power_2008_content.decode('ISO-8859-1')))

print('spec数据爬取成功')
```

在爬虫伪装之中，我用到了 fake_useragent 模块中的 UserAgent().random 方法，此方法会生成随机的 User-Agent，让每次访问都不相同。

2、SPEC 数据入库

spec数据入库

```
# 连接数据库
engine = create_engine("mysql+pymysql://root:asd2288026@localhost:3306/spec?charset=utf8")

# 读取各表列名
sql4 = 'select * from power_2008'
sql1 = 'select * from cpu_2017'
sql2 = 'select * from cpu_2006'
sql3 = 'select * from java_2015'

power_2008_column = pd.read_sql_query(sql4, engine)
cpu_2017_column = pd.read_sql_query(sql1, engine)
cpu_2006_column = pd.read_sql_query(sql2, engine)
java_2015_column = pd.read_sql_query(sql3, engine)

# 将爬虫下来的spec数据列名与数据库中对应表的列名保持一致
power_2008_data.columns = pd.DataFrame(power_2008_column).columns
cpu_2017_data.columns = pd.DataFrame(cpu_2017_column).columns
cpu_2006_data.columns = pd.DataFrame(cpu_2006_column).columns
java_2015_data.columns = pd.DataFrame(java_2015_column).columns

# spec数据入库
power_2008_data.to_sql('power_2008',engine, if_exists='replace', index=False)
cpu_2017_data.to_sql('cpu_2017',engine, if_exists='replace', index=False)
cpu_2006_data.to_sql('cpu_2006',engine, if_exists='replace', index=False)
java_2015_data.to_sql('java_2015',engine, if_exists='replace', index=False)
print('入库成功')
```

3、CPU 价格数据爬取

CPU价格数据爬取

```
def get_price(row):
    headers = {
    'User-Agent': UserAgent().random}
    cpu_str = row['processor'].replace(' ','%20')
    url = 'https://itprice.com/dell-price-list/' + cpu_str + '.html'
    content = requests.get(url, headers = headers).content
    price_list = re.findall('\$[(\d)|(\,)]+\.', str(content))
    if price_list and len(price_list) > 0:
        price = int(max(set(price_list), key=price_list.count).replace(',', '').replace('.', '').replace('$', ''))
        row["price"] = price
    else:
        row["price"] = 0
    return row

cpu_2017_data = cpu_2017_data.apply(get_price, axis=1)

print('价格数据爬取成功')
```

4、应用数据清洗及入库

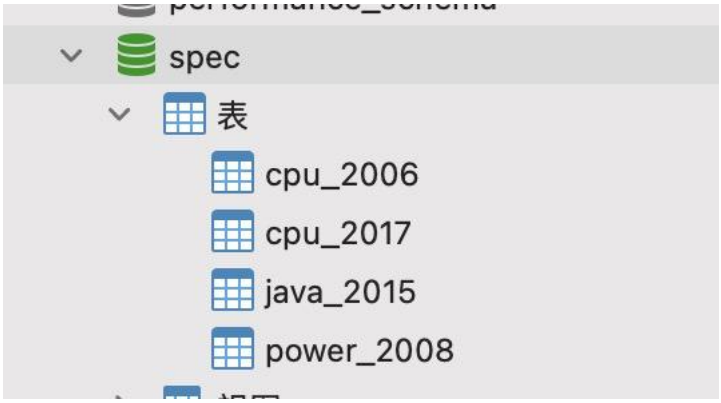
对应用数据进行数据清洗并入库

```
# 连接数据库
engine = create_engine("mysql+pymysql://root:asd2288026@localhost:3306/spec?charset=utf8")

# 定义应用数据列名
ads_cpu_2017 = cpu_2017_data[['processor', 'price', 'peak_result', 'base_result', 'cores', 'chips',
                              'enabled_threads _per_core', 'processor_mhz', 'parallelization', 'base_pointer_size',
                             'peak_pointer_size', 'lst_level_cache', '2nd_level_cache', '3rd_level_cache', 'test_date']]

# 数据清洗
result = ads_cpu_2017.groupby("processor", as_index =False)["base_result"].quantile(q=0.5, interpolation='nearest')
result = result.merge(ads_cpu_2017, how='left', on = ['processor', 'base_result'])
result = result.drop(b[b['processor']=='redacted'].index)
result = result.drop_duplicates(subset=['processor'], keep='first')
result = result.drop(b[b['price']==0].index)
result['test_date'] = pd.to_datetime(b['test_date'], format='%b-%Y')

# 应用数据入库
result.to_sql('cpu_2017_result',engine, if_exists='replace', index=False)
print('应用数据入库成功')
```

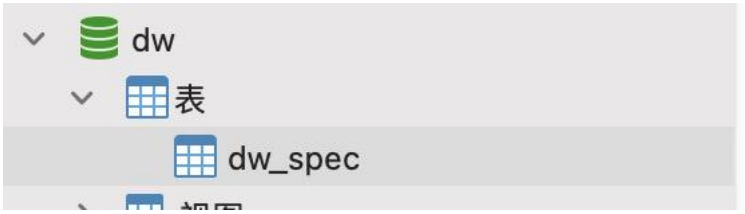
数据清洗方面，我删除了处理器名为“redacted”的，以及没有爬到价格的数据。我还对日期数据进行了格式化，把“Sep-2020”这类数据转为“2020-09”，让后续处理及分析更加方便。由于在 SPEC 数据中，几乎每款 CPU 都进行了多次测试，输出了多个测试报告，为了更加直观，可靠地看出每款 CPU 的性能表现，这里取 base_result 分数为中位数的测试数据。

三、数据资产化及数据体系建设

1、贴源数据 ODS

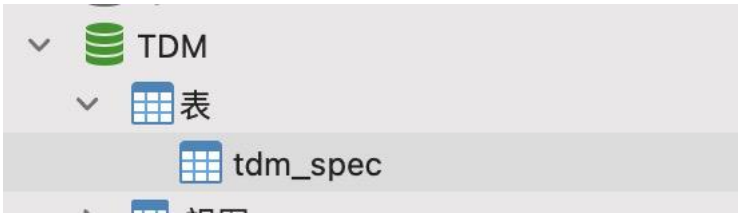


2、统一数仓 DW



此表属性为四个 ODS 表的公共属性，汇集了四个源表的数据。

3、标签数据 TDM

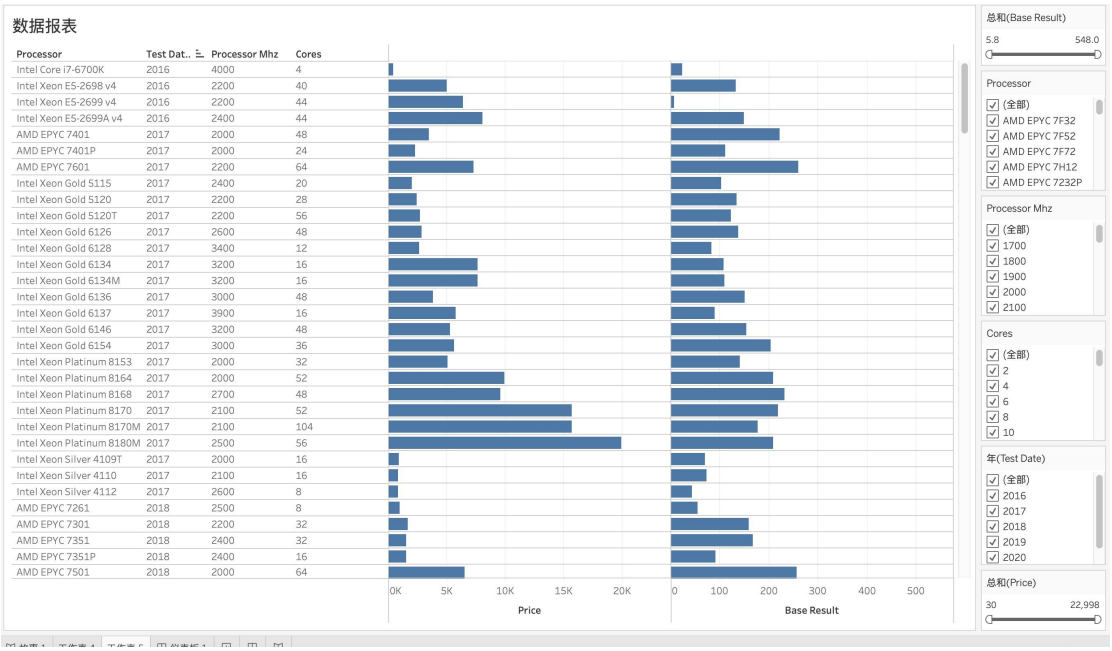


obj_id	1_level_lable	2_level_lable1	2_level_lable2	2_level_lable3	2_level_lable4	2_level_lable5	2_level_lable6
java_2015	服务器基本属性	hardware_vendor	system	(NULL)	(NULL)	(NULL)	(NULL)
java_2015	中央处理器	processor	processor_mhz	cores	chips	cores_per_chip	threads_per_core
java_2015	服务器基本属性	hardware_vendor	system	(NULL)	(NULL)	(NULL)	(NULL)
java_2015	中央处理器	processor	processor_mhz	cores	chips	cores_per_chip	threads_per_core
java_2015	其余硬件环境	memory	memory_details	disk	(NULL)	(NULL)	(NULL)
java_2015	软件环境	operating_system	file_system	compiler	(NULL)	(NULL)	(NULL)
cpu_2017	服务器基本属性	hardware_vendor	system	(NULL)	(NULL)	(NULL)	(NULL)
cpu_2017	中央处理器	processor	processor_mhz	cores	chips	cores_per_chip	threads_per_core
cpu_2017	其余硬件环境	memory	memory_details	disk	(NULL)	(NULL)	(NULL)
cpu_2017	软件环境	operating_system	file_system	compiler	(NULL)	(NULL)	(NULL)
cpu_2006	服务器基础属性	hardware_vendor	system	(NULL)	(NULL)	(NULL)	(NULL)
cpu_2006	中央处理器	processor	processor_mhz	cores	chips	cores_per_chip	threads_per_core
cpu_2006	其余硬件环境	memory	memory_details	disk	(NULL)	(NULL)	(NULL)
cpu_2006	软件环境	operating_system	file_system	compiler	(NULL)	(NULL)	(NULL)
power_2008	服务器基础属性	hardware_vendor	system	(NULL)	(NULL)	(NULL)	(NULL)
power_2008	中央处理器	processor	processor_mhz	cores	chips	cores_per_chip	threads_per_core
power_2008	其余硬件环境	memory	memory_details	disk	(NULL)	(NULL)	(NULL)
power_2008	软件环境	operating_system	file_system	compiler	(NULL)	(NULL)	(NULL)

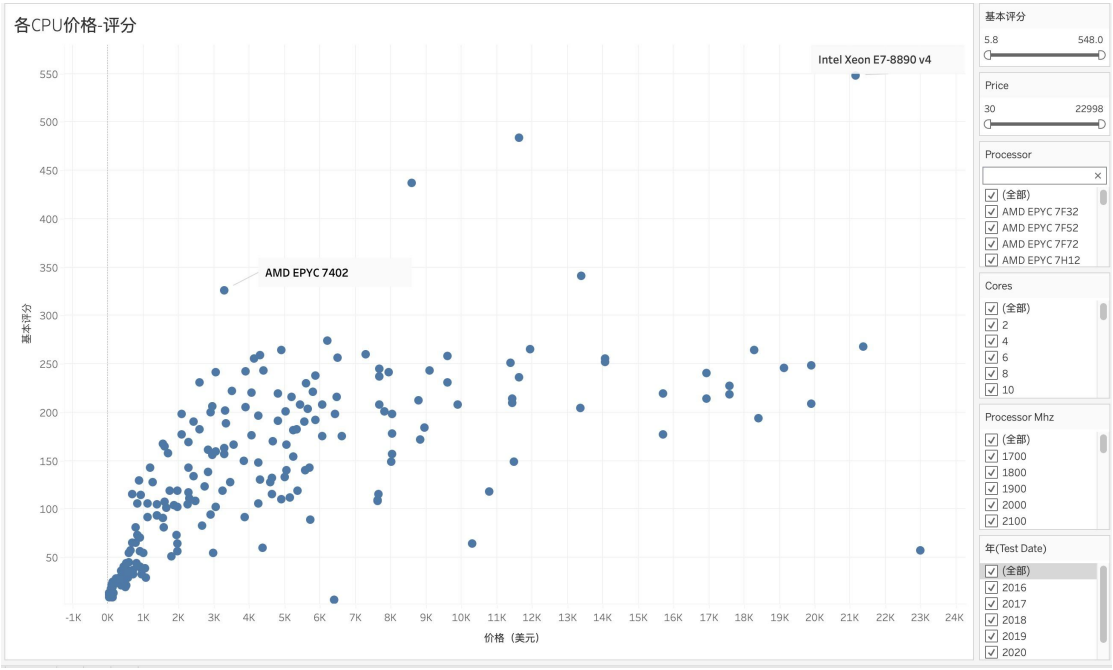
4、应用数据 ADS

processor	base_result	price	peak_result	cores	chips	enabled_threads _per_core	processor_mhz
AMD EPYC 7232P	54.9	589	0	8	1	2	3100
AMD EPYC 7251	56.1	899	57	8	1	2	2100
AMD EPYC 7252	115	689	122	16	2	2	3100
AMD EPYC 7261	54.9	999	51.8	8	1	2	2500
AMD EPYC 7262	81	1590	86.7	16	2	2	3200
AMD EPYC 7281	93.5	1399	94.8	16	1	2	2100
AMD EPYC 7282	106	1119	109	32	2	2	2800
AMD EPYC 7301	158	1699	173	32	2	2	2200
AMD EPYC 7302	143	2276	144	16	1	2	3000
AMD EPYC 7302P	117	2276	123	16	1	2	3000
AMD EPYC 7351	167	1549	178	32	2	2	2400
AMD EPYC 7351P	90.5	1549	0	16	1	2	2400
AMD EPYC 7352	166	3561	167	24	1	2	2300
AMD EPYC 7401	222	3499	0	48	2	2	2000
AMD EPYC 7401P	111	2299	121	24	1	2	2000
AMD EPYC 7402	326	3299	341	48	2	2	2800
AMD EPYC 7402P	163	3299	172	24	1	2	2800
AMD EPYC 7451	128	4599	134	24	1	2	2300
AMD EPYC 7452	182	5336	0	32	1	2	2350
AMD EPYC 7501	256	6499	0	64	2	2	2000

四、数据查询报表



五、数据可视化及价值变现



上图就是此项目的核心成果，根据此图，我们就可以根据客户预算，或者核数，频率等要求，为客户推荐 CPU。比如，A 企业预算是 3000 美元左右，上图的 AMD EPYC 7402 就是非常好的选择。