robots

在广告交易市场中发现价值，按需采买广告流量，最大化收益与花费。

用途

1. 发现价值。
2. 利用价格杠杆竞取广告位。

出价公式

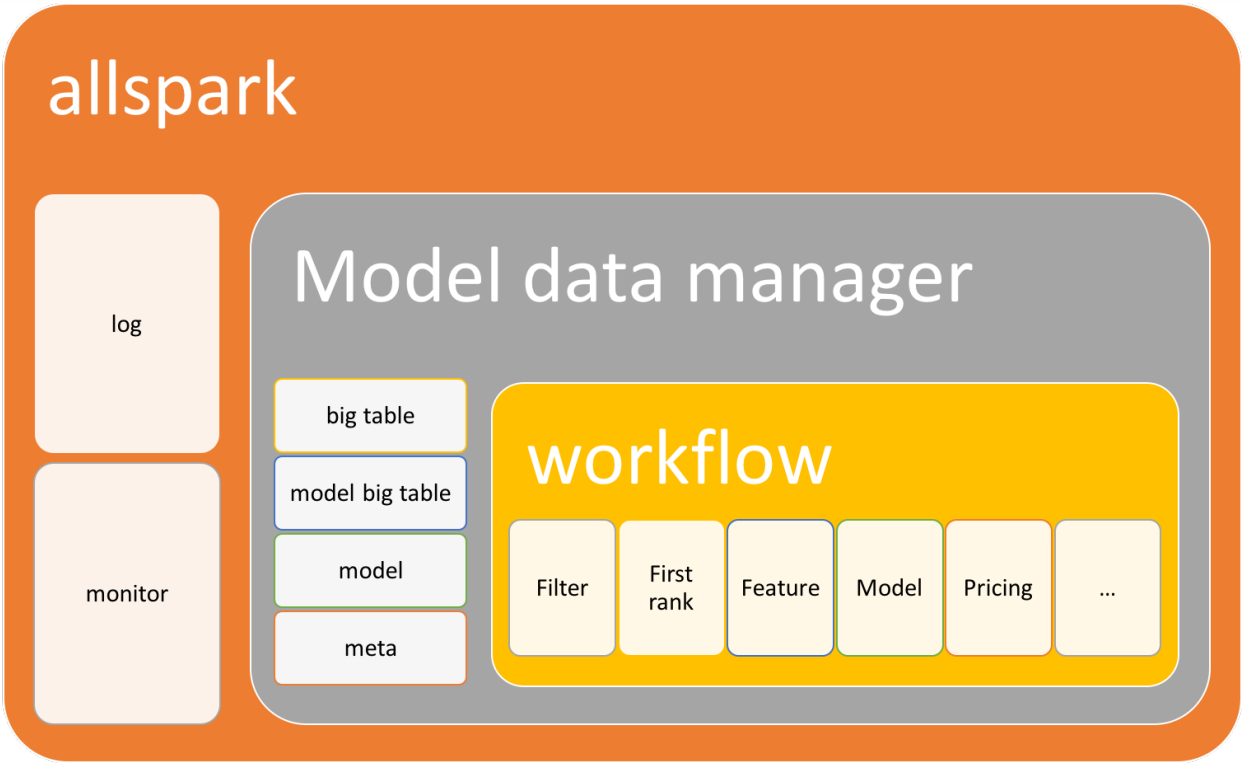
```
price = f(a, u, c|model)
a: 广告主
u: 用户
c: 媒体上下文
model: 历史数据、模型规则等
```

公式分解

$$CPM = CPI * 1000 = CPC * CTR * 1000$$

> - CPM 与平台结算
> - CPC 与广告主结算

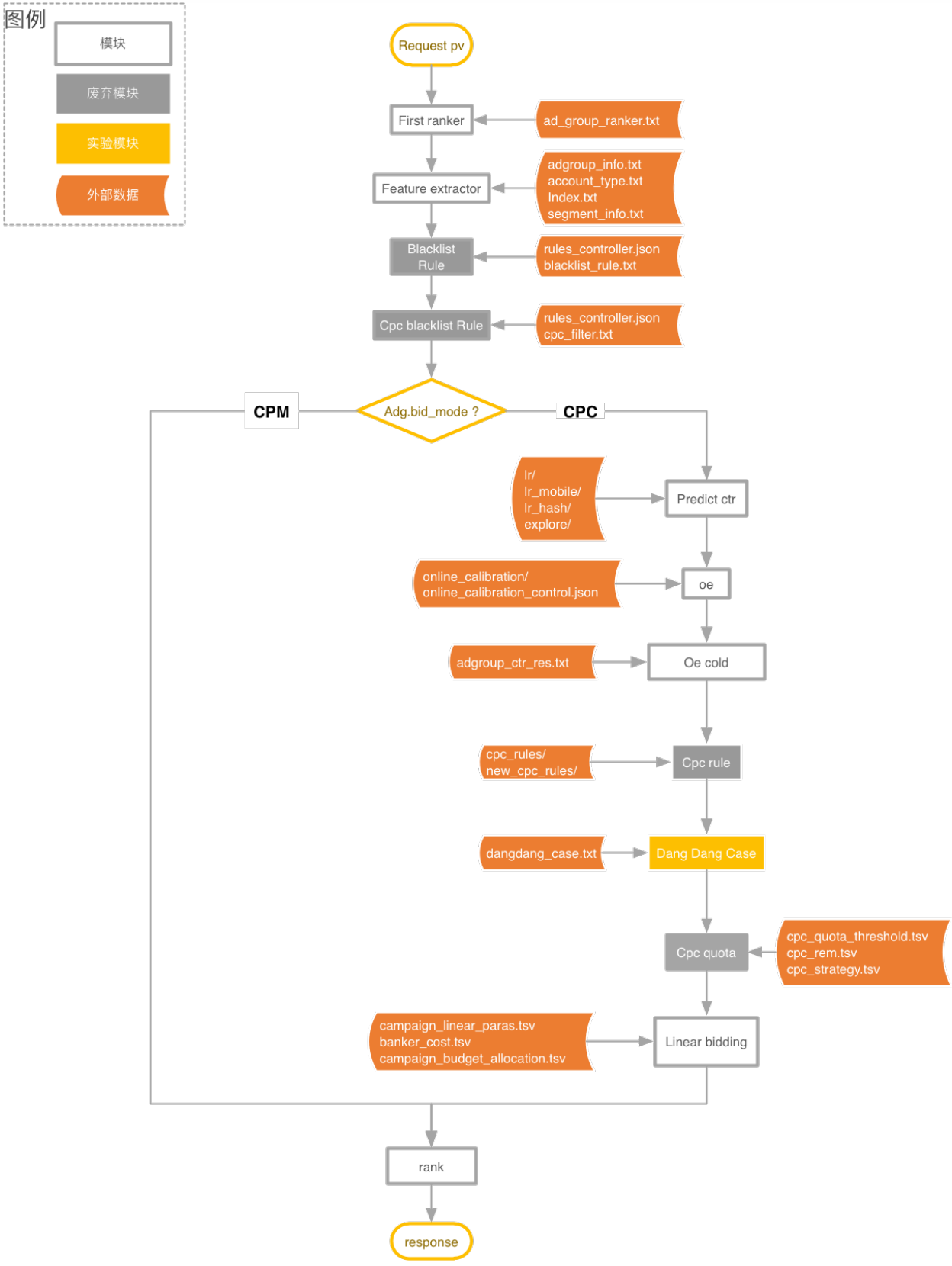
架构



数据类别

type	location	usage
big table	/var/log/data_update/output/bigtable/zampda_v3	获取adgroup信息（出价方式，价格，segment...）
model big table	/var/log/data_update/output/model	模型策略使用的数据
model	/var/lib/robots/model	模型包数据
meta	/usr/local/robot_data_builder/data/robot_data	模型包版本，全局价格限制，adgroup排名表

流程

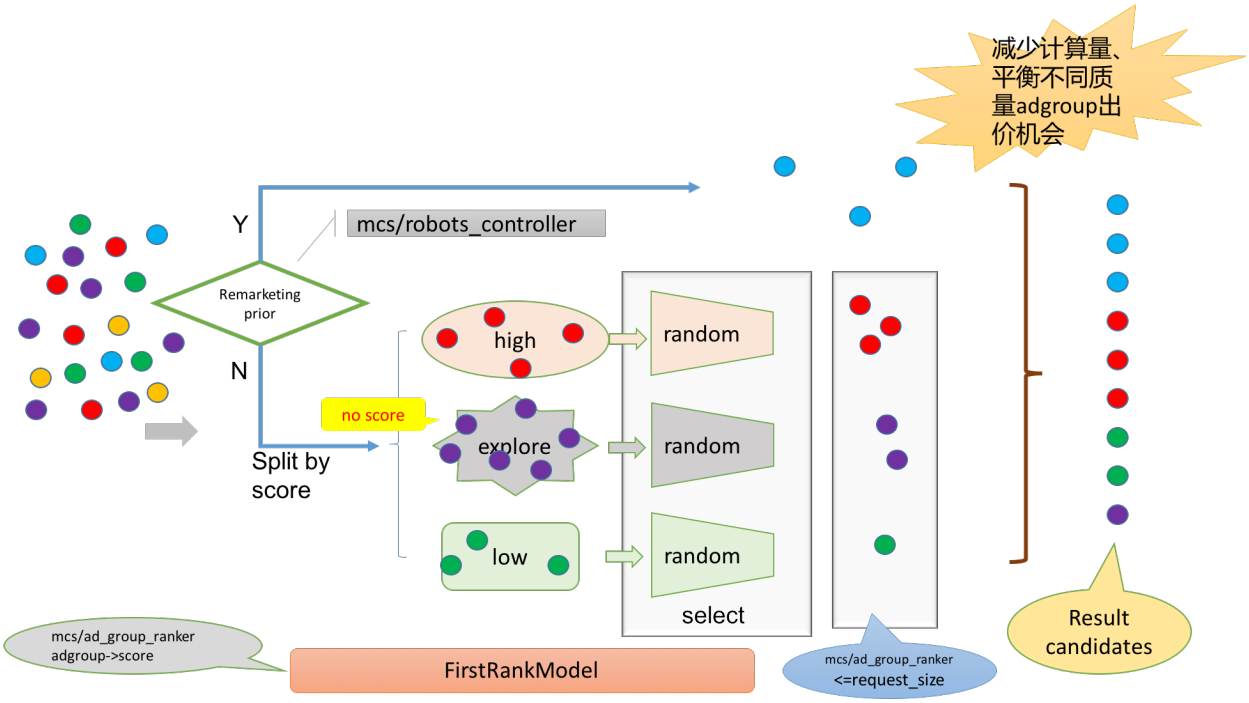


模块

first rank

针对候选adgroup过多超过robots处理能力时，平衡不同质量adgroup的出价机会，让质量越高的adgroup参与竞价的几率越大

流程




```
def first_ranker(adgroups):
    adgroups.sort(by='score', assending)

    # split high, low, explore adg into bins
    high_score = adgroups[:result_count]
    low_score = adgroups[result_counts+1:]
    explore = adgroups[adgroups.score.isnull()]

    # randam select adg from each bins by corresponding ratio
    result = []
    result.append(high_score.select(by=high_score_ratio))
    result.append(explore.select(by=explore_ratio))
    result.append(low_score.select(count=result_count - len(result)))
    return result
```

数据

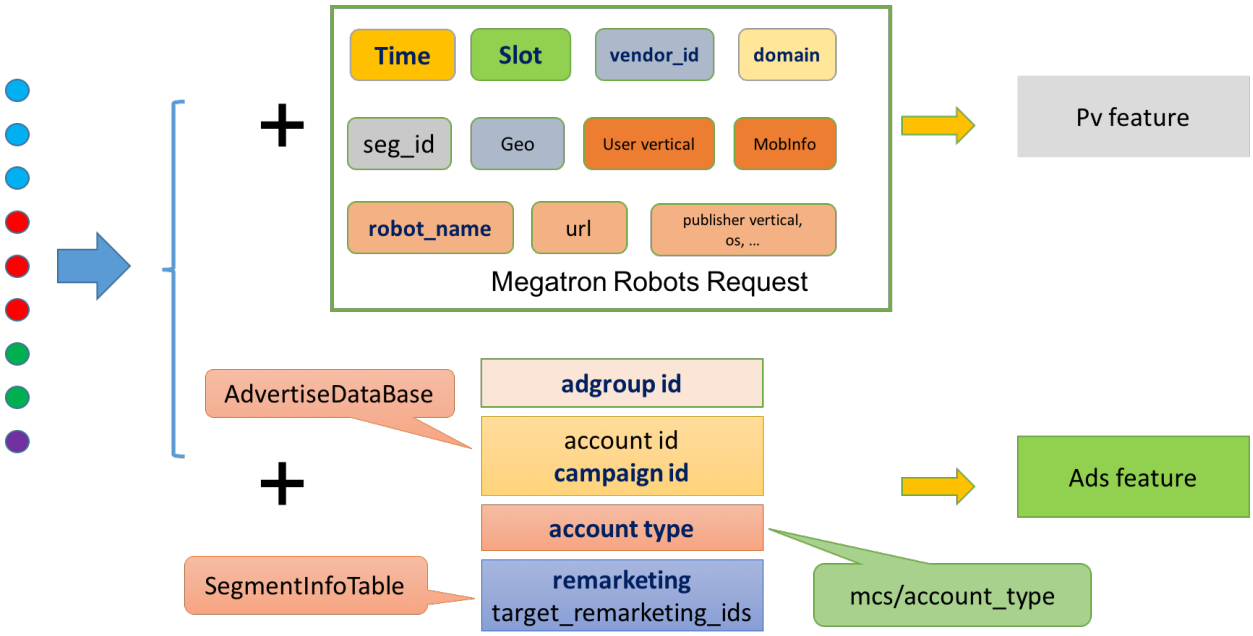
root	file	usage
meta	ad_group_ranker.txt	adgroup分数，筛选概率

ad_group_ranker.txt

```
${result_count}     # number of adgroup selected for bidding
${high_score_ratio} # select high score ratio
${explore_ratio}    # select no score ratio
${adgroup_id} \t ${score}
...
```

Feature extract

抽取每个adgroup相关特征为模型预测准备数据



数据

- 大表数据

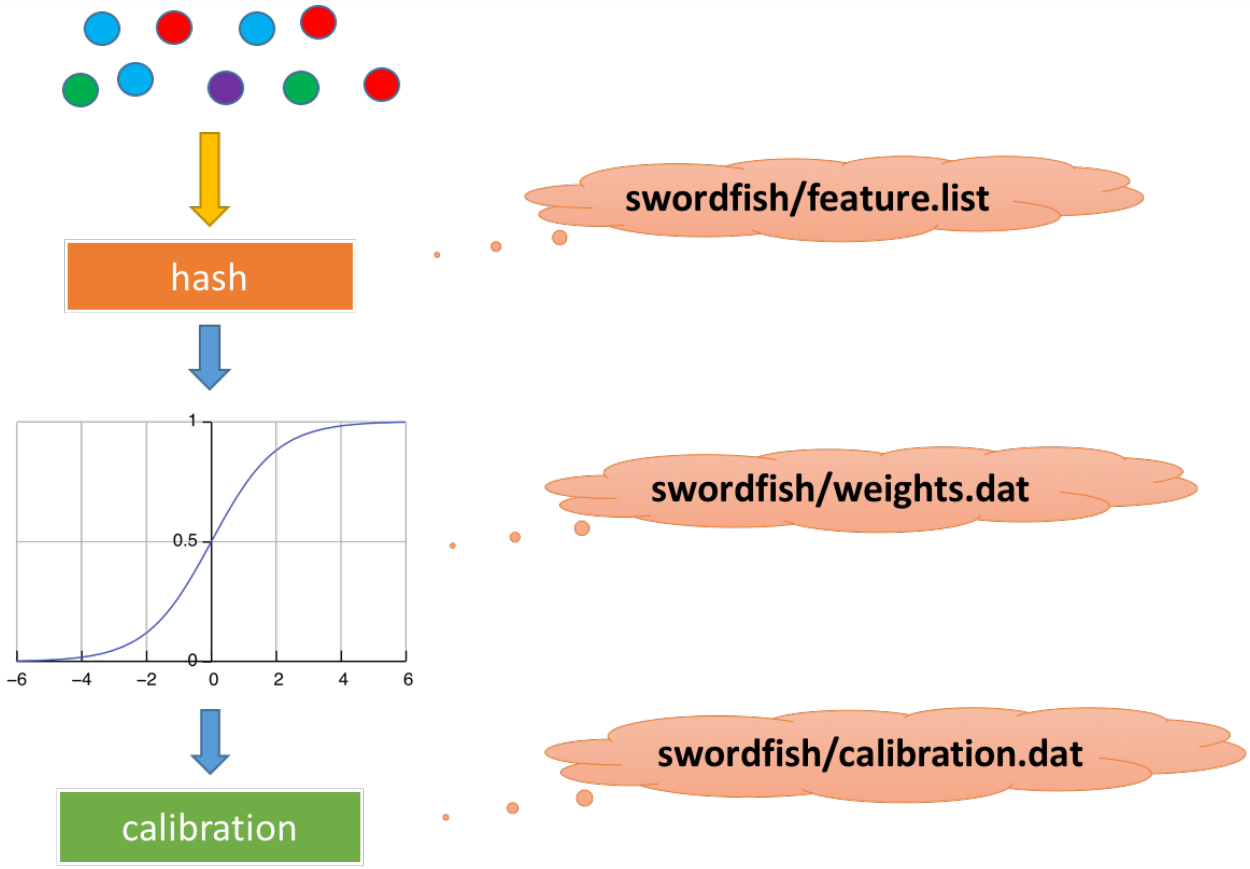
root	file name	usage
big table	adgroup_info.txt	adgroup出价类型，价格
big table	index.txt	adgroup定向（结合segment info判断是否rmkt）
big table	segment_info.txt	获取segment id的类型

- 流量数据

  直接来自megatron传递过来的流量特征

pCTR

通过logistic regression（LR）模型预测每个adgroup的pCTR



公式

$$f(x) = \frac{1}{1 + e^{-\sum wx}}$$

计算流程

```
def calibration(score):
    for low_score, high_score, factor in calibrator:
        if low_score <= score < high_score:
            return score * factor
    return score

def predict(adg):
    # lr
    w = 0.0
    for feature_name, start, end in feature.list:
        v = feature_name + '_' + feature_value
        h = hash(v, start, end)
        wi = weight[h]
        w += wi
    w = -1 * w
    score = 1.0 / (1 + e ** w)

    # calibration
    ctr = calibration(score)
    return ctr
```

模型包

root	file name	usage
model	swordfish_${version}.tar.gz	pc平台模型包
model	swordfish_mobile_${version}.tar.gz	移动平台模型包

数据格式

```
swordfish_${version}
├── ${vendor}
|   ├── ${is_rmkt}
|   |   ├── calibration.dat
|   |   ├── feature.list
|   |   └── weights.dat
...
```

> 示例

```
swordfish_${version}
├── 1
|   ├── 0
|   |   ├── calibration.dat
|   |   ├── feature.list
|   |   └── weights.dat
|   └── 1
|       ├── calibration.dat
|       ├── feature.list
|       └── weights.dat
...
```

feature.list

模型使用的特征列表

```
${feature_name}\t${bin_start}\t${bin_end}
...
```

字段	类型	说明
feature_name	string	使用的特征名
bin_start	int	哈希分桶的起始值
bin_end	int	哈希分桶的终止值

> 示例

```
is_rmkt 1   5000
vendor_id   1   5000
account_id  1   5000
campaign_id 1   5000
adgroup_id  1   5000
domain  5001    35000
host    5001    35000
refer_domain    5001    35000
browser_id_adgroup_id   36001   120000
city_adgroup_id 36001   120000
domain_adgroup_id   36001   120000
host_adgroup_id 36001   120000
slot_width_adgroup_id   36001   120000
weekday_adgroup_id  36001   120000
adgroup_id_refer_domain 36001   120000
adgroup_id_hour 36001   120000
slot_id_weekday 36001   120000
```

weights.dat

```
${offset}
${hash_value}\t${weight}
...
```

字段	类型	说明
offset	double	逻辑回归中便宜参数
hash_value	int	特征哈希后的值
weight	double	特征的权重

> 示例
>
> ```
> 0.000
> 0   -0.000000
> 1   -0.000000
> 2   -0.000000
> 3   -0.000000
> ...
> ```

calibration.dat

```
${score_start}\t{ctr_start}
...
```

字段	类型	说明
score_start	double	逻辑回归计算值
ctr_start	double	对应的真实ctr值

> 示例
>
> ```
> 0.000000    0.000000
> 0.012080    0.000000
> 0.013837    0.000000
> 0.015500    0.000000
> 0.017037    0.000000
> ...
> ```

OE（Observation Over Expectation）

修正模型ctr预估不准确

公式

$$OE = \frac{\sum click}{\sum pCTR}$$

流程

```
def oe_calibrator(ctr, model_name):
    model = calibrator_map[model_name]
    for data in model.ranges:
        if data.start < ctr < data.end:
            return data.factor
    raise 'not found'
```

数据

root	file name	usage
model big table	online_calibration/	oe矫正数据，每个adgroup单独一个数据文件

目录结构

```
online_calibration
├── file_list.txt
├── data_${vendor}_${adgroup_id}.dat
...
```

> 示例
>
> ```
> online_calibration
> ├── file_list.txt
> ├── data_1_1159395429071217836.dat
> ├── data_1_1159395429071217837.dat
> ...
> ```

文件格式

file name	usage
file_list.txt	目录包含的列表
data*${vendor}*${adgroup_id}.dat	oe矫正数据

data*${vendor}*${adgroup_id}.dat

```
${model_name}\t${ctr_start}\t${ctr_end}\t${factor}
...
```

> 示例

```
startree    0    1    1
startree_mobile 0    1    1
coldtree    0    1    1
swordfish    0    0.0034   0.948466391634
swordfish    0.0034   0.0051   1.3849977217
swordfish    0.0051   0.0061   2.69992424836
swordfish    0.0061   0.0077   1.93357829422
swordfish    0.0077   0.0101   1.94765843314
swordfish    0.0101   1    1.14225993914
swordfish_mobile    0    1    1
```

linear bidding

低成本获取大量点击

出价公式

$$price = base\_bid * \frac{score}{base\_ctr}$$

流程

```
def linear_bidding(campaign, campaign_linear_paras, bank_cost,
budget_allocation, current_hour):
    # check current budget
    actual_cost = bank_cost[campaign.id]
    total_budget = campaign.budget
    current_ratio = budget_allocation[(campaign.id, current_hour)]
    if actual_cost >= total_budget * current_ratio:
        raise 'no money'

    # check if this campaign need to bid
    params = campaign_linear_paras[(campaign.id, current_hour)]
    base_bid, base_ctr, bid_probability = params
    if rand() > bid_probability:
        raise 'give up bidding'

    price = base_bid * score / base_ctr
    return price
```

数据

root	file name	usage
model big table	campaign_linear_paras.tsv	linear bidding 参数配置文件
model big table	campaign_budget_allocation.tsv	campaign预算
banker_minute_cost_log	banker_cost.tsv	campaign当前花费

campaign_linear_paras.tsv

```
${campaign_id}\t${hour}\t${base_bid}\t${base_ctr}\t${bid_probability}
```

banker_cost.tsv

```
${campaign_id}\t${current_cost}
```

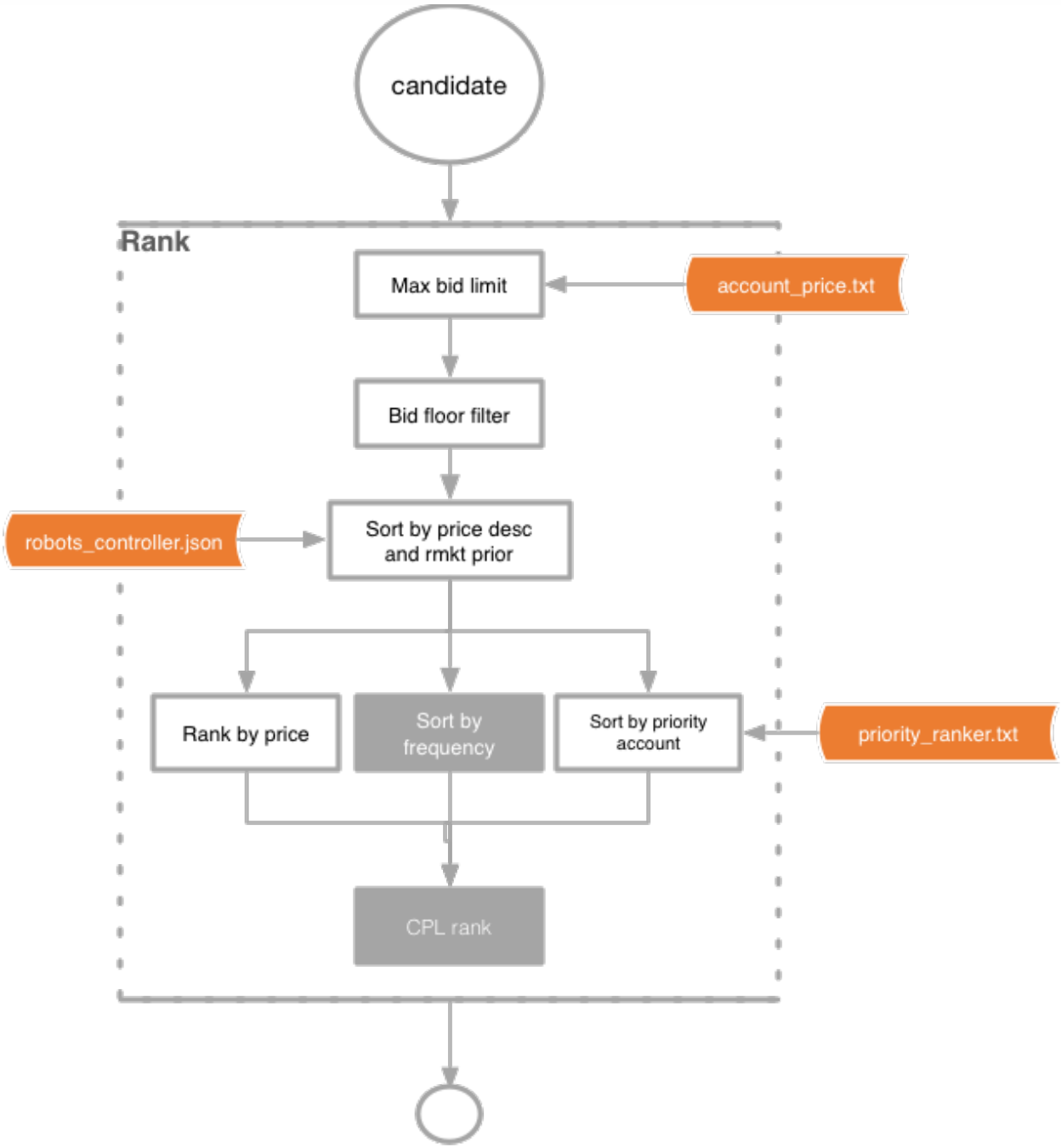
campaign_budget_allocation.tsv

```
${campaign_id}\t${hour}\t${ratio}
```

rank（精排）

根据adgroup的出价、策略等因素平衡各个adgroup的竞价机会，最大概率的赢下流量

流程



数据

root	file name	usage
model big table	priority_ranker.txt	优先广告配置

priority_ranker.txt

```
${enable_option}
${ratio} # if random_value < ratio then apply prior account)
${exchange}\t${account}\t${weight}
...
```

> 示例

```
0
0.95
-1  20  10
-1  458 30
-1  104 25
-1  13  20
```

生产部署

代码

field	value
仓库	http://git.dev.zamplus.com/projects/BID/repos/robots/browse?at=refs%2Fheads%2Fzampda3_release
分支	zampda3_release

服务器

四台实例对等部署

ip	hostname
172.22.56.8	SERV08
172.22.57.31	SERV5731
172.22.57.32	SERV5732
172.22.57.39	ROB02

数据分发

类别	内容
ftp机器	172.22.41.102
目录	/home/zamplusftp/model

数据推送

大表数据

通过同机部署的data-update-server定时从ftp机器上拉取

mcs数据

通过同机部署的robot_data_builder 定时向mcs数据库拉数据

模型数据

由172.22.40.251上服务主动将模型生成的模型包推送到每台robots机器

策略数据

通过同机部署的data-update-server定时从ftp机器上拉取

监控

http://grafana.dev.zamplus.com:8813/#/dashboard/db/30-robots