

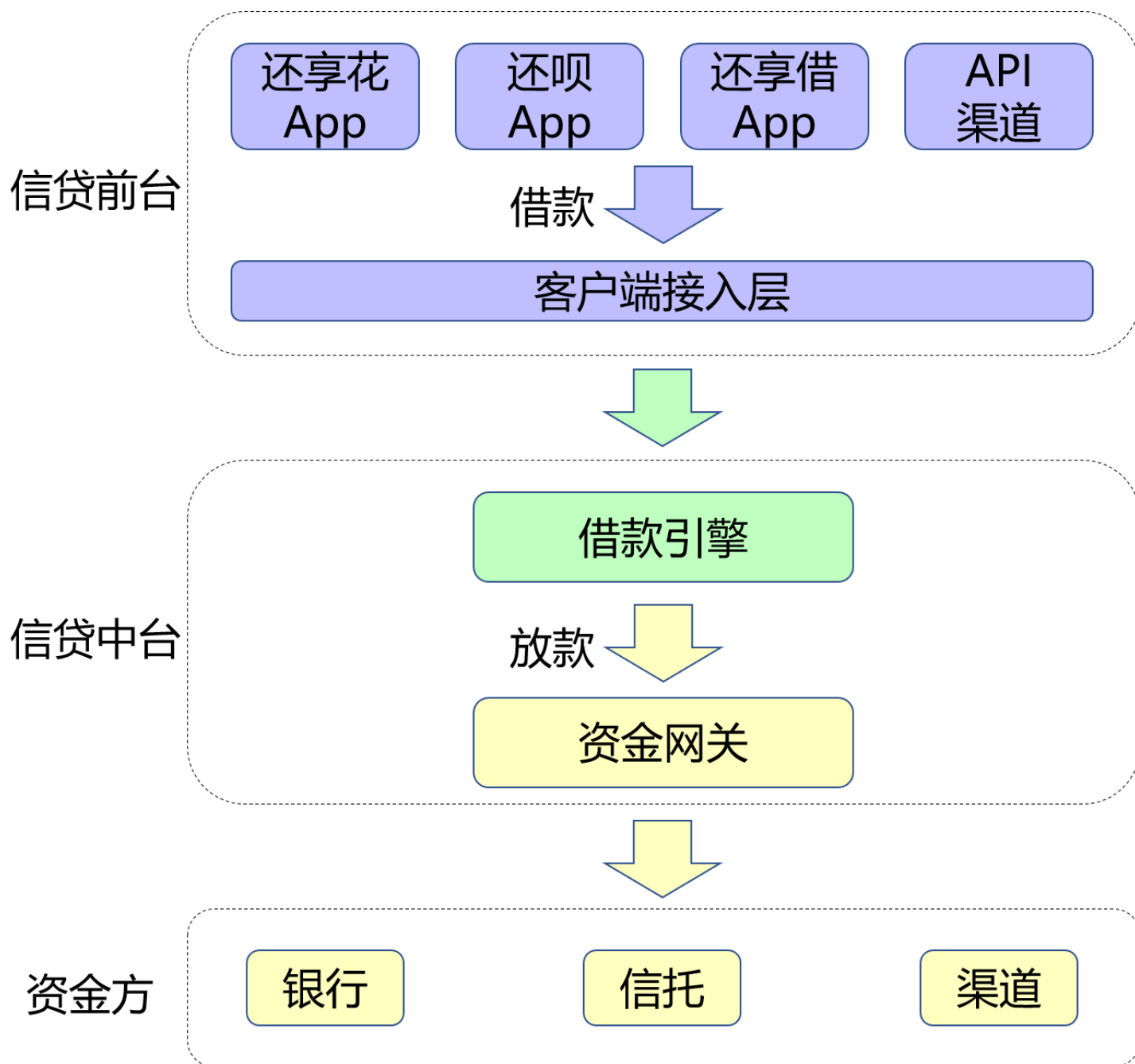
# 全面数字化建设之放款策略化的演进之路

## 1. 背景

还呗最初是一款信用卡账单分期 App，服务广大信用卡持卡人，帮用户解决信用卡难题。而后还呗衍生出了多种产品形态，面向年轻人提供账单分期和商品分期等多种服务。用户可在购物、消费、还款场景下，享受更灵活、更便捷、更高效的分期生活服务。

虽然产品形态多样，但本质上还呗为用户提供的是小额贷款服务。由于还呗并无自有资金，是一个助贷平台，并且在还呗几年的努力下，与还呗合作的资金方很多，作为借款人和资金方的中介，当用户在还呗平台发起借款请求时，如何为用户推荐最合适的资金方，是还呗的一项最基本也是最重要的能力。

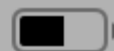
备注：「助贷业务」是指助贷机构利用自身掌握的获客、风控及贷后管理优势，向资金方(包括持牌金融机构、类金融机构)推荐借款人，经资金方风控终审后，完成发放贷款，并获取相关服务费的业务。



图：数禾放款业务架构精简版

## 1.1 用户借款流程

用户打开还呗 App 后，选择「借现金」。之后 App 跳转至借款方案选择页面，用户可以选择借款金额和分期数。



## 借现金

借款金额 (元)

1000

全部借出



### 选择借款期限

1个月



3个月



6个月



12个月



18个月





图：还呗 - 借款金额和分期选择

选择之后，系统会自动基于金额和期数，匹配最合适的资金方，并算出用户的还款计划（比如每一期要还多少钱，每一期还款的本金、利息是多少等）



## 借现金

借款金额 (元)

1000

全部借出

本次可借100至13000元，需100元的整数倍

还款方式


分期还款 &gt;

借款期限

6个月 &gt;

首期还款

¥163.15(11月26日) &gt;

收款储蓄卡  招商银行(1087) >  
该卡也将作为您的自动还款卡

优惠券

-30.03元利息优惠券 &gt;



提交借款

图：还呗 - 借款方案确认

用户点击「提交借款」后，就借款申请就提交成功了。



## 借款提交成功

借款资料审核中，最快10分钟到账。  
为提升还款成功率，建议您再次认证银行卡

[回到首页](#)[立即认证](#)

借款金额 1,000.00元

借款期数 6期

借款银行卡  招商银行(1087)

放款机构 分众小贷

借款资金由持牌机构提供，若逾期将上报央行征信、百行征信等。一处失信，处处受限。请按时还款，避免对个人信用及生活造成影响

元对人民币及人民币造成影响。

图：还呗 - 借款提交成功页

之后，系统会基于用户选择的金额、期数，以及系统匹配到的资金方，使用该资金方的资金完成对用户的放款。



用户



还呗App



借款引擎

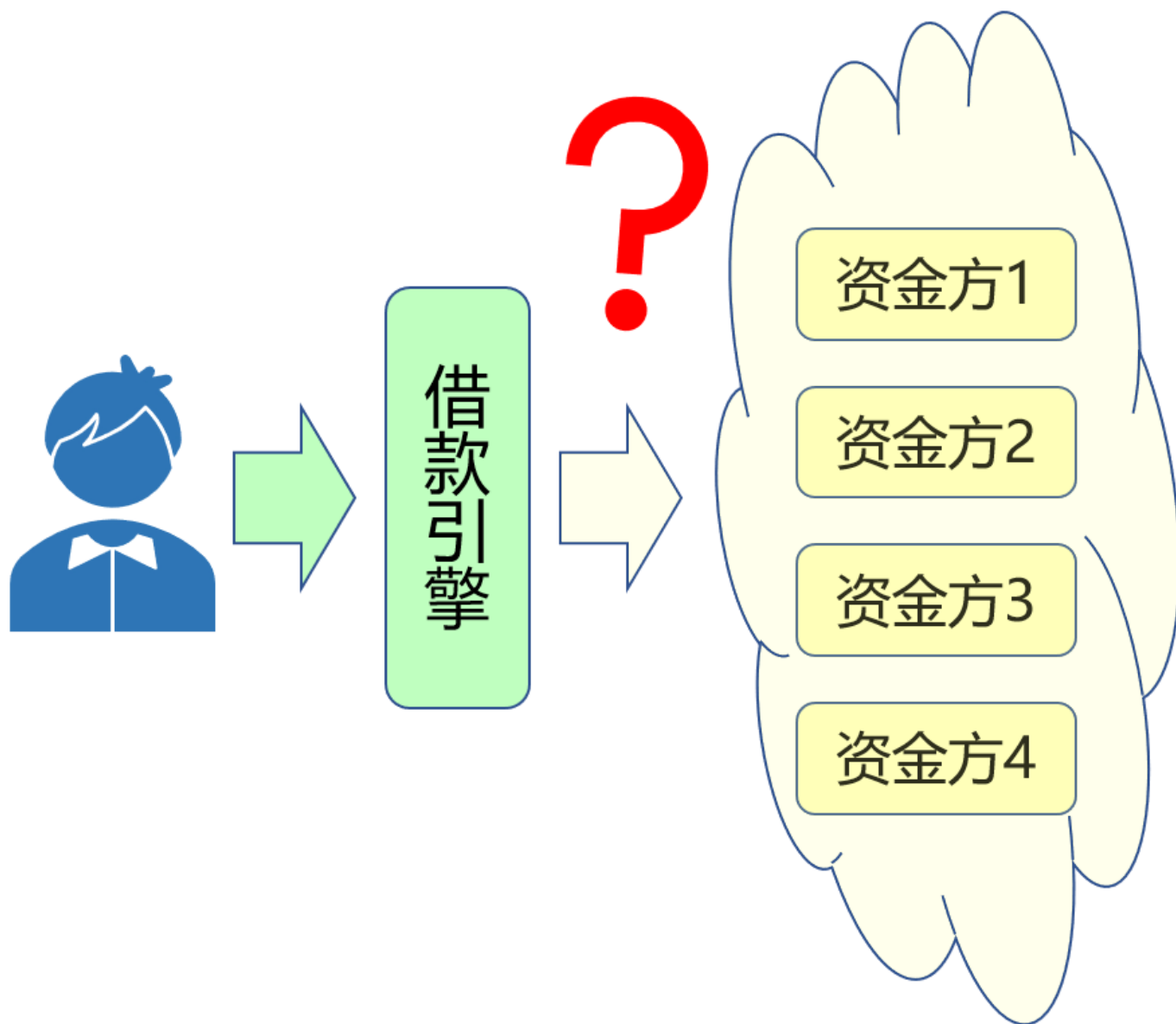


资金方



图：借款流程时序图（演示版）

因为是精简版的流程图，所以看起来很简单，但其中仍有一个要点值得探讨：与还呗合作的资金方很多，但是系统是怎么选择出最适合当前用户的资金方的呢？



图：如何匹配资金方

## 1.2 资金方匹配方案探讨

### 方案一：轮询方式随机指定资金方

第一次匹配时选择资金方1，第二次匹配时选择资金方2，第三次匹配时选择资金方3，第四次匹配时重新选择资金方1 .....

缺点：每个资金方支持的金额和分期并不一样；资金方每天放款的金额是有限制的，金额放完后就不能再放款了；有些资金方有特殊限制，如只支持年龄在 X 范围内的用户借款，其他资金方则不作此限制.....

由于每个资金方的放款规则都不同，所以不能采用轮询的方式随机指定资金方，我们需要根据用户的借款意愿、用户的个人信息、每日放款情况等等综合判断并为用户匹配最合适的资金方。

## 方案二：代码实现资金方匹配规则

比如通过代码实现复杂的匹配规则，基本思想参考下面的伪代码：

```
if (资金没放完(资金方1)) {  
    if (年龄 <= 40) {  
        return "资金方1";  
    }  
}  
if (资金没放完(资金方2)) {  
    if (年龄 <= 40) {  
        if (用户借款金额 < 1000) {  
            if (用户选择分期 in [3, 6, 12]){  
                return "资金方2";  
            }  
        }  
    }  
}  
if (资金没放完(资金方3)) {  
    ... ..  
}
```

这样的代码可以满足需求，甚至在精心对代码进行设计后，能够有较高的可读性、可扩展性等。

这样做有一个很大的问题：当商务同事隔几天就要求接入个新的资金方，我们就需要根据需求开发复杂的匹配规则；当商务同事觉得之前某个资金方的规则不满意，隔几天就想要变更匹配规则时，这时开发工作就成为了纯粹的需求翻译工作，体现不出创造性。由于此类需求无穷无尽，故此处始终需要占用可观的开发资源。

## 2. 演进1：借款路由 1.0

由于资方的匹配规则会经常性的变化，导致始终需要维持明显的开发成本，那么解决方案就呼之欲出了：将资金方的匹配规则及相关复杂逻辑从业务代码中剥离出去，转由独立的决策模块完成，决策模块可以提供友好的配置页面，方便业务同事独立完成配置工作，从而释放出相应的开发资源。

### 2.1 方案设计

每当借款引擎收到匹配资金方的请求时，借款引擎会准备足够的参数，并将这些参数输入决策路由，决策路由经过计算后会吐出一个值，该值即代表某一个资金方。

每当业务方需要变更匹配规则时，只需要根据借款引擎提供的丰富的参数，独立在决策路由的配置页面完成规则的配置工作即可。

基本信息

名称：

放款路由

Key：

LEND\_ROUTER

版本：

1.0.2

拷贝路由

校验

保存

移动模式

0

开始匹配

0

资金方1

0

资金没放完

0

年龄匹配

1

资金方2

0

资金没放完

0

用户借款金额匹配

2

资金方3

0

资金没放完

查看历史变更

显示输入项列表

操作指导

名称：

用户借款金额匹配

排序：

0

\* 类型：比较节点

展示更多

属性：

#{借款金额}

小于等于

1000

取值：

资金方2

保存节点

图：决策路由 - 借款路由规则（演示版）

决策路由的基本原理：决策路由配置完毕后，会形成一颗决策树，决策树上有多个决策节点，每个决策节点上均维护有独立的匹配规则。每次执行业务决策时，均会从头开始对整个决策树进行遍历，采用深度优先遍历（DFS）的思想，对遍历过程中访问的每一个决策节点执行相应的匹配规则，匹配失败时会跳过该决策节点，匹配成功时会检查其是否配置有决策子树，有子树时继续访问子树，无子树时直接返回当前决策节点的取值。

当然，有时候业务同事在配置决策树时，借款引擎默认给出的参数可能并不够，业务同事希望能够提供一些额外的参数，比如：用户使用的客户端版本、用户的手机系统类型、用户使用的借款产品类型等。当然这也没有问题，开发同事只需要基于需求开发相应的输入参数即可，增加通用的输入参数与变更复杂的匹配规则相比，开发工作量要小的多。

## 2.2 总结

借款路由演进至 1.0 版本后，具有以下特点：

1. 资金方匹配规则由业务同事独立设计与实现，对于开发同事和业务系统均无感知，有利于明确业务同事和开发同事各自的职责。
2. 将基本不变的业务流程代码与经常变动的资金方匹配规则剥离，两者独立发展，可保证通用业务流程的整体稳定性。
3. 释放了维护复杂匹配规则的开发资源，但增加了业务同事配置的工作量。

## 2.3 问题

该方案设计出来后，效果很好，稳定工作了一年，而后开始逐步暴露出问题了：

1. 资金方规则匹配的复杂逻辑完全从代码中抽离了出来，开发同事的工作量明显减少，但随着合作的资金方越来越多，决策路由的匹配规则也越来越复杂，配置经常出错，每天都有相关的问题产生且需要排查，总之决策路由的维护成本越来越高。
2. 每个资金方支持的分期范围和金额范围都有可能不同，且针对不同的用户，不同资金方的支持也都不同。然而用户借款时看到的是同一套可供选择的金额范围和分期范围方案，这套方案虽然经过了精心的人工设计，但仍然难以满足所有用户的需求，容易发生这样的问题：用户选择了某个不支持的金额和分期，导致无法匹配到可用的资方，从而无法成功提交借款请求。

## 3. 演进2：借款路由 2.0

针对以上问题，我们提出了新的设计方案。

### 3.1 演进方案

#### 3.1.1 通用资方配置及匹配规则单独维护

决策路由维护的规则过于繁杂，其中部分规则具有共性，虽然不同资金方匹配规则的具体取值不同，但是匹配的逻辑是一致的，没必要在决策路由中一遍一遍重复配置同一类规则，这类规则完全可以剥离出来由代码实现，从而明显减轻决策路由的负担。

自研资金配置中心：使用统一的维护页面，单独配置各个资金方共有的静态配置，且可以在代码中维护相应的匹配规则，从而减轻决策路由的负担。资方共有的业务配置，如：资金方支持的分期范围和金额范围、资金方支持的借款优惠券类型等。

资金平台 / 资金配置 / 资金配置中心 / 配置详情

资金方编号: FUND-1

资金方名称: 资金方 1

资方属性: 银行类

资产属性: 重资产

算费配置

新算费配置

放款环节

还款环节

账务配置

调账配置

准入路由

日志

借款金额、期数

3

500 元

~

5000 元

6

500 元

~

5000 元

12

500 元

~

5000 元

支持放款优惠券类型

☒ 利息优惠券

☒ 利率折扣券

☐ 新利息优惠券

☐ 新利率折扣券

☒ 免息券

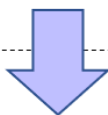
图：资金配置中心 - 资金方配置（演示版）

3.1.2 将单个决策路由拆为两个决策模块

对决策路由进行重构，将决策路由整体上分为两个决策模块：准入模块和策略模块。准入模块主要用于生成可借款的资金方列表（详细说明参加下一小节），进而生成用户可选择的借款方案。用户选择借款方案后，会经过策略模块，用于从刚刚生成的资金方列表中匹配最合适的资金方。两个决策模块职责分明，单个模块相比于原先的决策路由复杂性均大大减小。



用户发起借款



获取今日可放款的资金方列表

资金方排期

资金方1, 资金方2  
资金方3, 资金方4  
资金方5, 资金方6  
资金方7



基于通用过滤条件  
初步筛选资金方

通用条件过滤

资金方1, 资金方2  
资金方3, 资金方4  
资金方7



获取可对该用户  
放款的资金方列表

准入决策

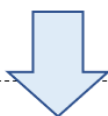
资金方1  
资金方2  
资金方3



匹配最优的  
一个资金方

策略决策

资金方2



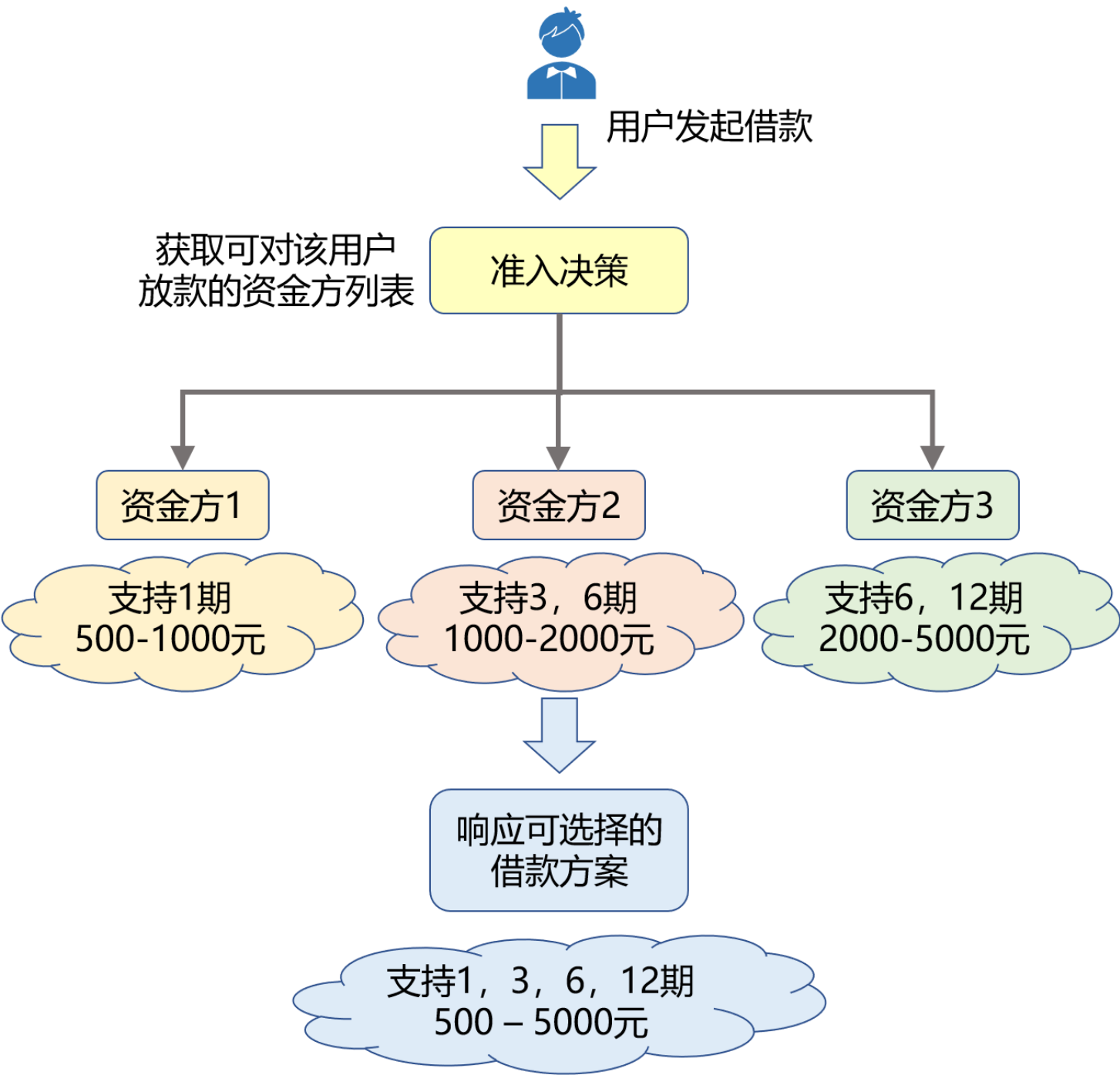
响应用户资金  
方及借款方案

资金方2 + 方案

图：借款路由 2.0 流程图（演示版）

### 3.1.3 实现用户可选择借款方案的动态生成

如上节所述，决策路由整体上分为准入模块和策略模块。用户要发起借款时，先过准入模块，得到一个资金方列表，须保证该列表中的资金方，对当前用户均可放款。将这些资金方支持的借款金额范围、支持的借款分期范围等取并集，即可得到用户可选择的借款金额、分期范围，根据不同的用户特征可生成最合适的可选借款方案。



图：动态生成用户可选择的借款方案

经过以上操作，动态生成的方案经过用户任意选择后，大概率能匹配到至少一个资金方。当然了，如果实在没有匹配成功，系统会自动向用户推荐最合适的借款方案，尽可能提升用户的借款成功率。

### 3.1.4 基于业务流引擎重构整个决策路由



通过前几小节的说明及配图，我们知道，借款路由 2.0 相比于借款路由 1.0，代码实现复杂了很多。为了减轻业务规则配置的复杂度，部分原本由决策路由实现的匹配规则转由业务代码实现；为了扩展更多的业务能力，借款路由新增了部分业务处理逻辑。

从流程图中可以发现，整个借款路由由多个逻辑分明的处理模块组成，相关的代码可基于上图的业务逻辑按照流程逐步进行实现，所以我们借助公司自研的业务流引擎，对借款路由进行了重构。

业务流：即工作流（Workflow），将业务流水线化，可将原先聚在一起较多耦合的处理逻辑拆分为相互关联的若干个处理模块，每个模块职责单一，具有较高的内聚性。业务数据在模块之间流转同时经由各个模块处理，业务逻辑清晰明确。

使用公司自研的业务流引擎重构借款路由，具有以下优势：

## 1. 业务流程可读性提升

通过代码实现原子化的业务模块后，可以通过配置页面将已实现的业务模块进行关联和展示，每个业务模块的职责清晰明确。单个业务模块开发的模版代码如下：

```
@BizFlowModule(name = BizFlowModuleEnum.DEMO)
public class BfLendApplyDemoModule extends AbstractBizFlowModule {

    @Override
    public ProcessResult process(ProcessContext<FundRouteContext> processContext) {
        return createSuccessProcessResult();
    }
}
```

只需要在模板方法中填入业务逻辑即可完成开发工作。可以在配置页面完成业务模块间的分支处理、异常处理，将相关逻辑从业务代码中抽离出来，有利于实现业务模块的单一职责，高内聚。

## 2. 业务流程可编排，可维护性提升

由于单个业务模块职责单一，且具有较高的内聚性，若需要模块内逻辑变更时，完全可以重新实现该模块，并在业务流层面直接完成新老模块的灰度替换。当在业务流程层面需要进行逻辑增减、业务调整时，则可以简单通过增减模块，调整模块顺序实现。以上措施有力保证了业务的整体稳定性和可维护性。

流程名称： 资金路由精简版「演示」

流程编号： PF-2021

保存流程

历史版本

复制

校验

全屏模式

撤销

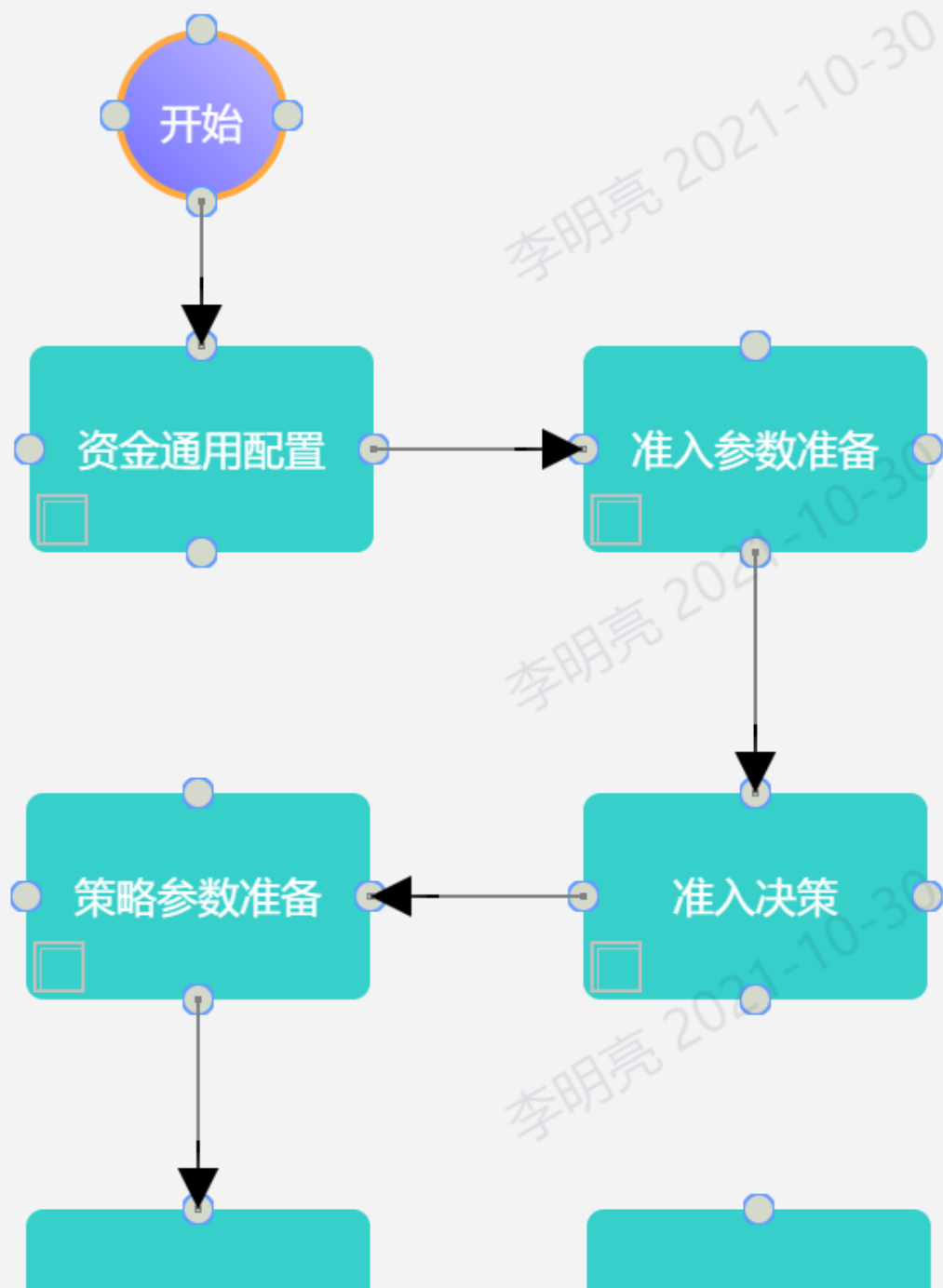
还原

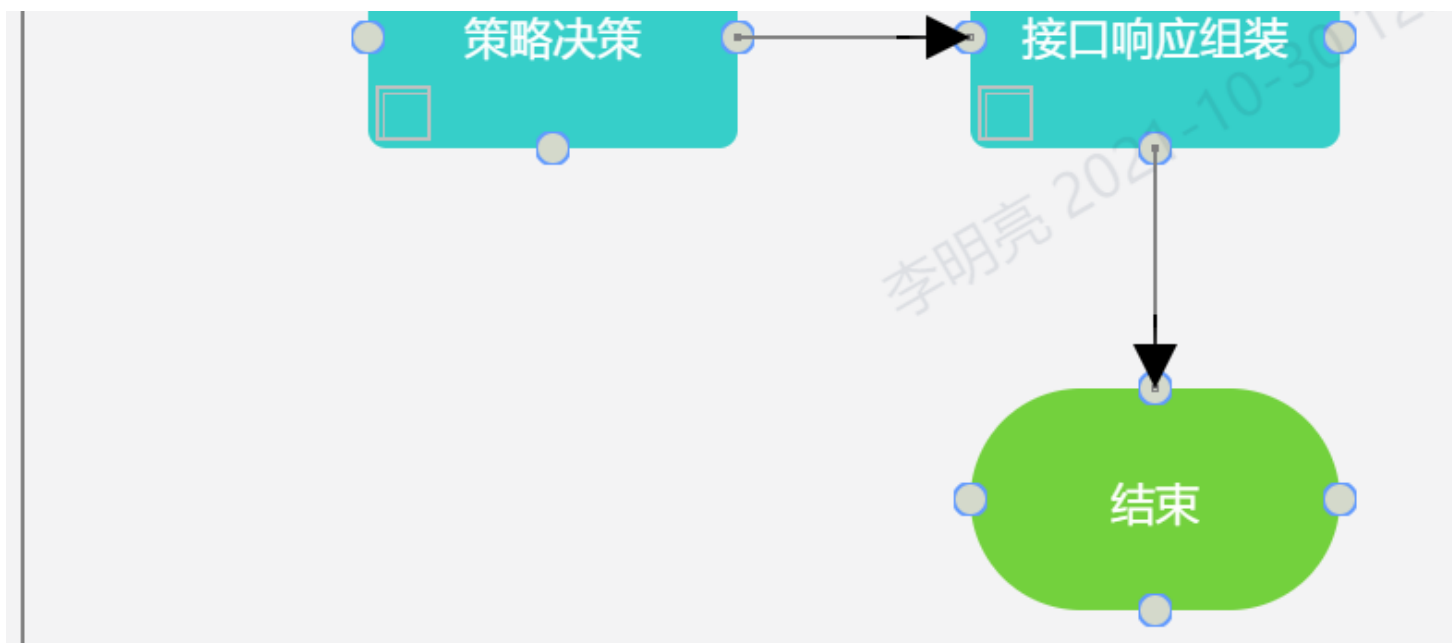
开始

执行

条件

结束





图：业务流引擎 - 借款路由 2.0（演示版）

## 3.2 总结

1. 对决策路由中复杂的匹配规则重新进行梳理，将通用的资金方配置单独维护，通用匹配规则（如通用的资金方匹配规则）抽离到业务代码中实现，并对决策路由进行拆分，简化单个决策模块内部匹配规则的复杂度，提升决策模块本身的可维护性，有力降低了匹配规则的维护成本。
2. 针对不同用户匹配可借款的资金方列表，进而展示最合适的可选借款方案供用户选择，明显增加了用户与资金方匹配的成功率。
3. 使用业务流引擎重构借款路由，提升了借款路由相关代码的可读性、可维护性和稳定性。

## 3.3 问题

基于前面的设计，借款路由整体已经趋于完善了，看起来已经没有了太多的改进空间，实际则不然。在一段时间的维护过程中，我们发现，借款路由还是有棘手的问题需要改进。

1. 随着决策输入参数的增加，取值耗时也在线性增加，导致有时用户与资金方的匹配耗时长达10秒，严重影响了用户体验，急需优化。
2. 随着时间的推移，与还呗合作的资金方越来越多，对产品的运营越来越精细化，借款引擎需要开发的决策输入参数也越来越多。导致原先并不受关注的决策输入参数开发工作量越来越重，给开发团队增加了不小的负担。输入参数过多过杂，又使得借款引擎耦合了过多且并不需要关心的业务系统以及参数获取逻辑。

针对以上问题，我们再次提出了新的设计方案。

## 4. 演进3：借款路由 3.0

## 4.1 演进方案

### 4.1.1 使用特征平台管理和配置特征

#### 1. 数据调研

对决策输入参数进行统计分析，我们逐渐发现，这些参数可以分为两类：

第一类是借款流程必知的参数，以下简称借款流程参数，如：本次用户借款的金额、期数、借款银行卡、还款银行卡等，这些参数借款引擎本身就需要知道，甚至有些参数可能就只有借款引擎知道，所以借款引擎开发这些输入参数无可厚非。

第二类需要借款引擎耦合其他业务系统并计算得到，以下简称外部依赖参数，如：用户是否绑定了某个银行的借记卡（需要耦合卡系统）、用户在某个特定的资金方是否已授信（需要耦合资金系统）、用户是否有借款订单还未还款完毕（需要耦合订单系统）、用户的真实姓名及注册手机号（需要耦合用户信息系统）。

我们使用公司自研的特征平台，接管外部依赖参数的获取及计算。

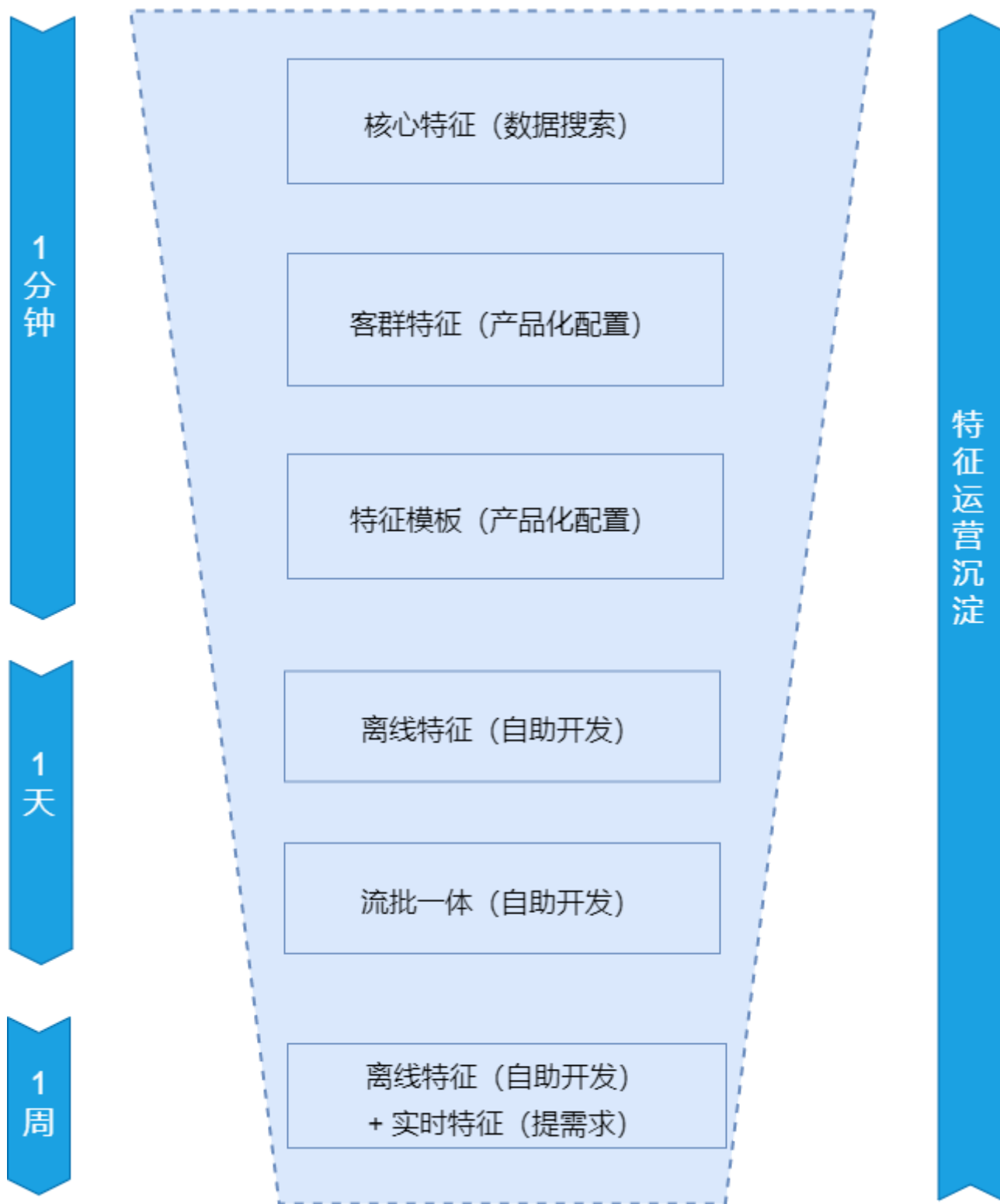
#### 2. 特征平台的使用

特征：有两种表述：1. 某个实体的某种属性值；2. 根据主题（主题，就是指某种实体，如用户实体、订单实体）调用的数据。在本文中，「特征」和「决策输入参数」两个概念含义一致。

特征平台管理并维护多种类型的特征，借款路由的常用特征如下：

1. 离线特征：基于 Hive 数据，编写 Hive SQL 生成，只能获取 T-1 数据。通常离线特征比较简单，业务线可以自行开发。离线特征会定期跑批刷新。
2. 实时特征：基于流式计算生成的特征。比离线特征有更高的时效性，是实时计算并生成的特征，通常在底层表增、删、改时生成。
3. 客群特征：是指基于部分属性或条件，对所有满足规则的用户进行归类，并生成相应的用户列表数据。

基于大数据部门开发、维护的特征平台，当用户（主要是业务同事）希望开发新的特征时，优先直接使用已有的特征；当无法满足要求时，可基于已有的模块化的特征，组装、计算并生成新的特征；如果仍未能满足要求时，可采用传统方式提需求让开发同事实现新的特征。



图：特征平台 - 特征“立体”交付体系

基于特征平台，当业务部门希望增加借款路由的输入参数时，绝大多数参数均可以自主高效实现，能够极大提升新参数上线的时效性。

借款路由需要执行决策时，可以基于特征平台高效获取所需的所有特征参数，可明显提升特征参数获取的时效性。

### 4.1.2 特征取值耗时优化

根据上一小节的方案进行改造，之前遇到的两个问题基本都得到了解决，用户与资金方的匹配耗时降到了2秒左右，已经达到了优化目标，然而业务同事又提出了新的要求，希望继续优化整体耗时。所以我们对借款路由的实现细节进行了进一步的梳理。

我们发现，有少量特征借款路由业务代码本身需要使用，如果特征平台和借款引擎均重复进行取值，不但会导致资源的浪费，更重要的是会影响到借款路由整体的时效性，进而影响用户体验，所以仅借款引擎自身做了取值工作。对此处的取值逻辑，我们可以继续优化：

1. 可以简单将借款引擎本身需要取值的特征参数取值逻辑，由串行转为使用线程池并行取值，即可大大减少特征参数取值的整体耗时。
2. 基于木桶原理，特征取值的整体时效性基本取决于耗时最久的那个特征的取值用时，所以在特征取值用时超出阈值时，果断进行降级处理，即可有力保证整体耗时的稳定性。

## 4.2 总结

1. 借款引擎将外部依赖参数进行剥离，并由特征平台统一管理，解除了借款引擎与多数业务系统的耦合，大大减轻了借款引擎的开发负担；借助特征平台的能力，明显提升了特征获取的时效性。
2. 借款引擎特征取值细节优化，采用了多种方法，如：串行转多线程并行取值，取值降级等，进一步提升了借款路由整体的时效性以及耗时的稳定性。

## 5. 借款路由演进总结

1. 公司较强的技术实力和丰富的技术工具能够有效为业务赋能。以上设计深度整合了公司自研的决策路由、资金配置中心、业务流引擎、特征平台等，若公司未提供上述技术组件、平台，借款路由难以优化到满意的水平。
2. 技术演进需要一步一步踏实实施，且每一步均目标明确，则大概率能达成较好的结果。最开始因为资金匹配规则复杂且需要持续占用开发资源，才引入了决策路由；而后因为规则复杂到难以维护才开始了借款路由的重构以及引入资金配置中心，因为流程化逻辑管理复杂才引入了业务流引擎；而后因为输入参数的开发需要持续占用开发资源且借款引擎与外部耦合越来越严重，才引入了特征平台管理外部特征，因为借款路由耗时需要进一步优化，才开始改造特征取值的实现细节。业务模块的每次演进均是了解决明确的问题，或对某业务场景能带来明显的优化提升，具有很强的目的性，最终基本都能够达成预定的目标。
3. 技术演进切忌直接设计大而全的目标，一步到位不可取。业务知识需要在具体的工作实践中逐步积累，业务问题也会逐步变得明确，在工作实践中一步步解决业务痛点，量变引起质变，稳健达成最终的目标。如果一开始直接设计一个大而全的目标，基本都会因为研发成本高、外部支持不足而难以推进。就算能够推进，因为未能了解真正的业务痛点，最终实现的结果极有可能与真实的需求相去甚远。