KEDACOM

KEDACOM
www.kedacom.com

OSP应用程序重构-基于OSP的事务模型

讲师：范小钢

时间：2011年1月

苏州科达科技有限公司

培训目的

KEDACOM

- 通过本课程的学习，您将能够了解：
  - OSP应用程序设计的一种新思维：事务模型
  - 目前OSP应用程序中普遍存在的问题
  - 如何通过事务模型改善这些问题
  - 基于OSP事务模型的实战
- 适用对象：
  - 需要重构、优化OSP应用程序的开发人员

培训大纲

KEDACOM

- OSP程序存在的问题
- 如何构建基于OSP的事务模型
- 事务模型如何解决问题
- 事务模型的具体实现

现有程序中存在的问题：

- 业务处理： 缺乏事务和状态定义
- 类和函数： 存在巨大的类和函数
- 时间性能： 存在大量的遍历查找
- 调试界面： 打印信息控制不精准
- 代码质量： 难以维护和扩展 代码

OSP程序存在的问题

KEDACOM

- 业务处理：缺乏事务和状态定义
  - 处理各种异常情况的能力差，出现问题较难排查，故障恢复需要重新启动程序

举例 和讨论：

一个录像任务出现问题，既不能停止，也不能再度开启

OSP程序存在的问题

KEDACOM

- 类和函数：存在巨大的类和函数
  - 众多复杂的业务逻辑糅杂在一个函数或一个类

举例 和讨论：

巨大的CInstance类

巨大的CInstance::InstanceEntry函数

OSP程序存在的问题

KEDACOM

- 时间性能：存在大量的遍历查找
  - 对于大容量的平台服务器来说时间效率低下

举例 和讨论：

应该使用hashmap作为容器的地方使用了list

OSP程序存在的问题

KEDACOM

调试界面：打印信息控制不精准

举例和讨论：

总是有大量无关的调试信息不停出现

基于OSP的事务处理模型

- 事务和状态
  - 事务：系统中一次具体业务处理的整个过程
  - 状态：业务处理的各个阶段
  - 关系：一个事务有多个状态构成；状态由事务的特定事件去推动转换；状态模式
  - 功能：通过事务和状态去抽象系统中的各种业务的数据和逻辑；从概念上将各种业务进行分离，便于产生高内聚、低耦合的类。
  - 作用：事务横向分解系统业务的复杂性；状态纵向分解系统业务的复杂性

基于OSP的事务处理模型

KEDACOM

- 事务分类
  - CMD型事务(命令)
  - NTY型事务(通知)
  - REQ_RSP型事务(请求、应答)
  - REQ_RSP_NTY型事务(批量数据查询)
  - 复杂事务：由以上各种事务组成

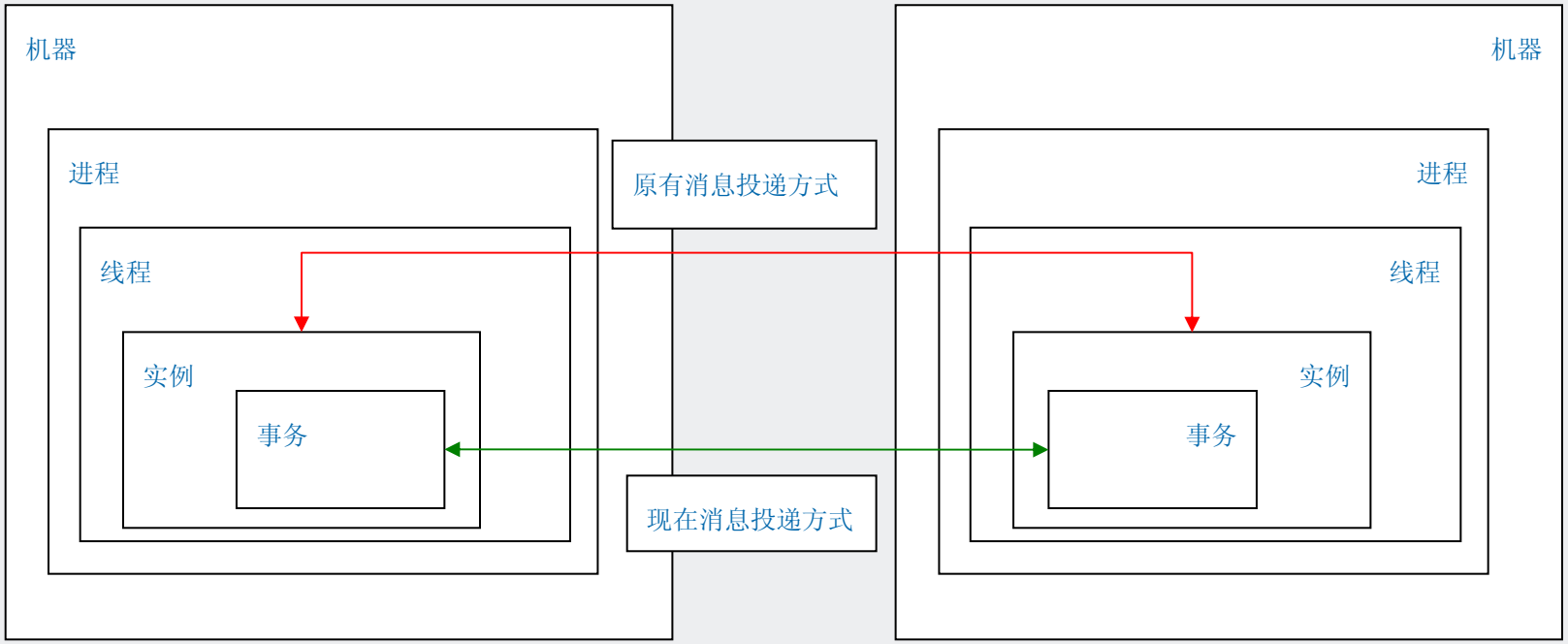
基于OSP的事务处理模型

KEDACOM

- 事务的消息模型：扩展的OSP消息模型

  原有模型：（node-app-instance）

  现有模型：（node-app-instance-task）



基于OSP的事务处理模型

- 事务一致性
  - 用事务号唯一标识事务对象
  - 事务具有消息集合
  - 消息集合中每条消息包含事务号
  - 事务号的类型(整形、字符串)
  - 事务号的产生(不能g_num++)
  - 根据事务号快速定位事务对象

基于OSP的事务处理模型

KEDACOM

- 事务/状态对象的产生和销毁
  - 内存需求：高速分配和释放的、不产生内存碎片的堆内存
  - 使用OspAllocMem、OspFreeMem 管理内存
  - 重载类的new、delete操作符
  - 内存管理基类COspObj
  - 事务/状态类都继承自COspObj
  - 直接使用new、delete产生和销毁事务/状态对象

基于OSP的事务处理模型

KEDACOM

- 事务的管理
  - 所有事务由事务管理器统一管理
  - 事务对象创建后向事务管理器注册
  - 事务管理器给事务分配全局唯一事务号
  - 使用CXMap(hashmap)保存事务号和事务指针
  - OSP实例根据事务号快速定位事务对象
  - 事务对象销毁后向事务管理器注销事务号

基于OSP的事务处理模型

KEDACOM

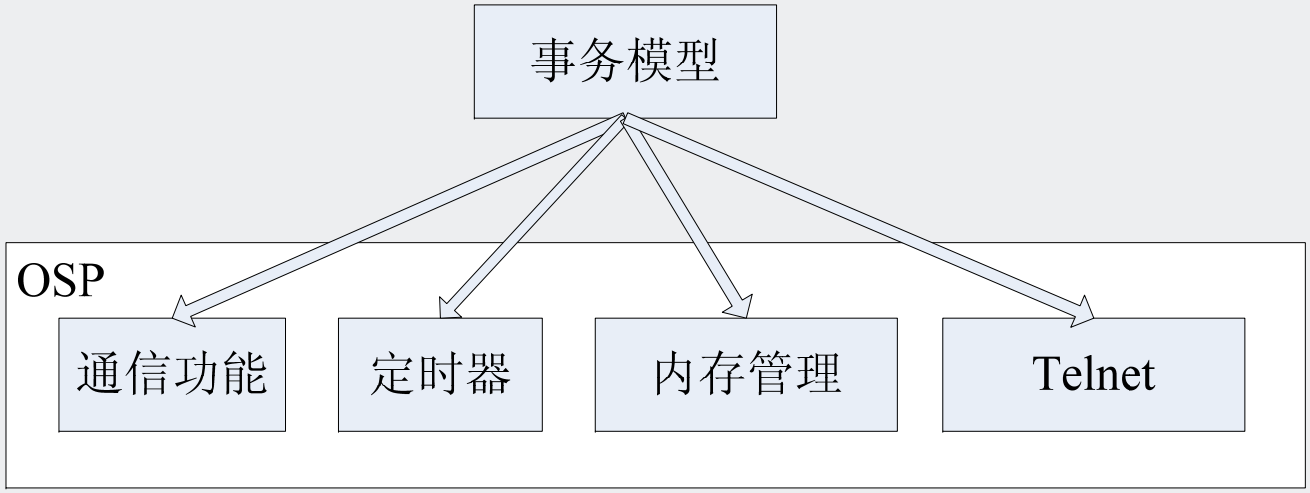
- 事务的轮询
  - 为什么需要轮询
    1. 业务需要，比如平台录像事务
    2. 每个事务中消息的超时处理
  - 使用OSP的定时器进行轮询
  - 轮询事件的最终处理者是状态
  - 同一事务不同状态的轮询结果不同
  - 事务分批进行轮询，避免一次处理大量事务

- 事务的telnet控制
  - 可以指定事务号输出调试信息
  - 可以指定事务类型输出调试信息
  - 可以只对一个新产生的事务输出调试信息
  - 增加定时消息级别和频繁通知消息级别
  - 可以通过telnet强行终止事务，进行故障恢复

基于OSP的事务处理模型

KEDACOM

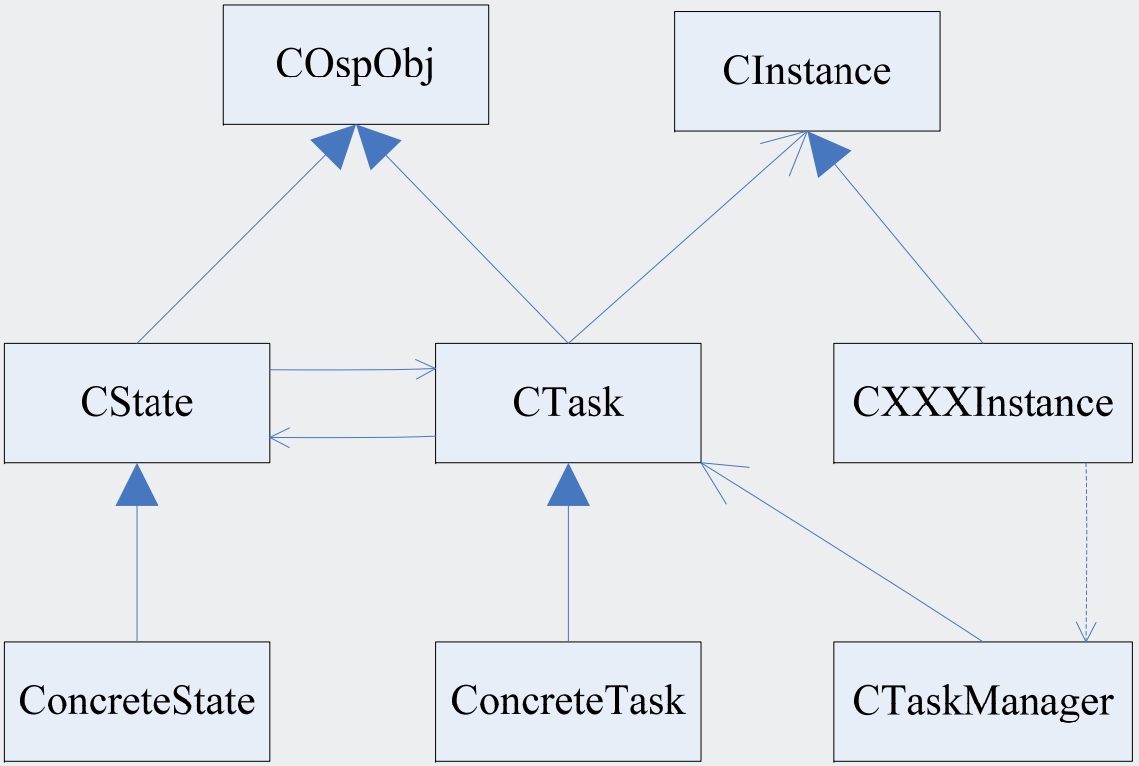
- 事务模型和OSP
  - 事务利用OSP通信功能进行消息收发
  - 事务利用OSP内存池进行内存管理
  - 事务利用OSP定时器进行轮询
  - 事务利用OSPtelnet功能进行调试、打印



基于OSP的事务处理模型

KEDACOM

- 事务模型的基本类图
  - 实箭头表示继承
  - 空箭头表示指针引用



- 改变编程模型
  - 原有模型：以CInstance为中心，各种业务的数据和逻辑都在CInstance中。
  - 现有模型：以CTask、CState为中心，数据和逻辑分散到CTask、CState类中； CInstance仅仅作为消息路由的句柄。

事务模型实战-平台录像

KEDACOM

- 基础构件
  - 内存管理类COspObj
  - 事务、状态的基类CTask、CState
  - 事务号生成器CTaskNOManager
  - Hash表类CXMap
  - 事务管理类CTaskManager

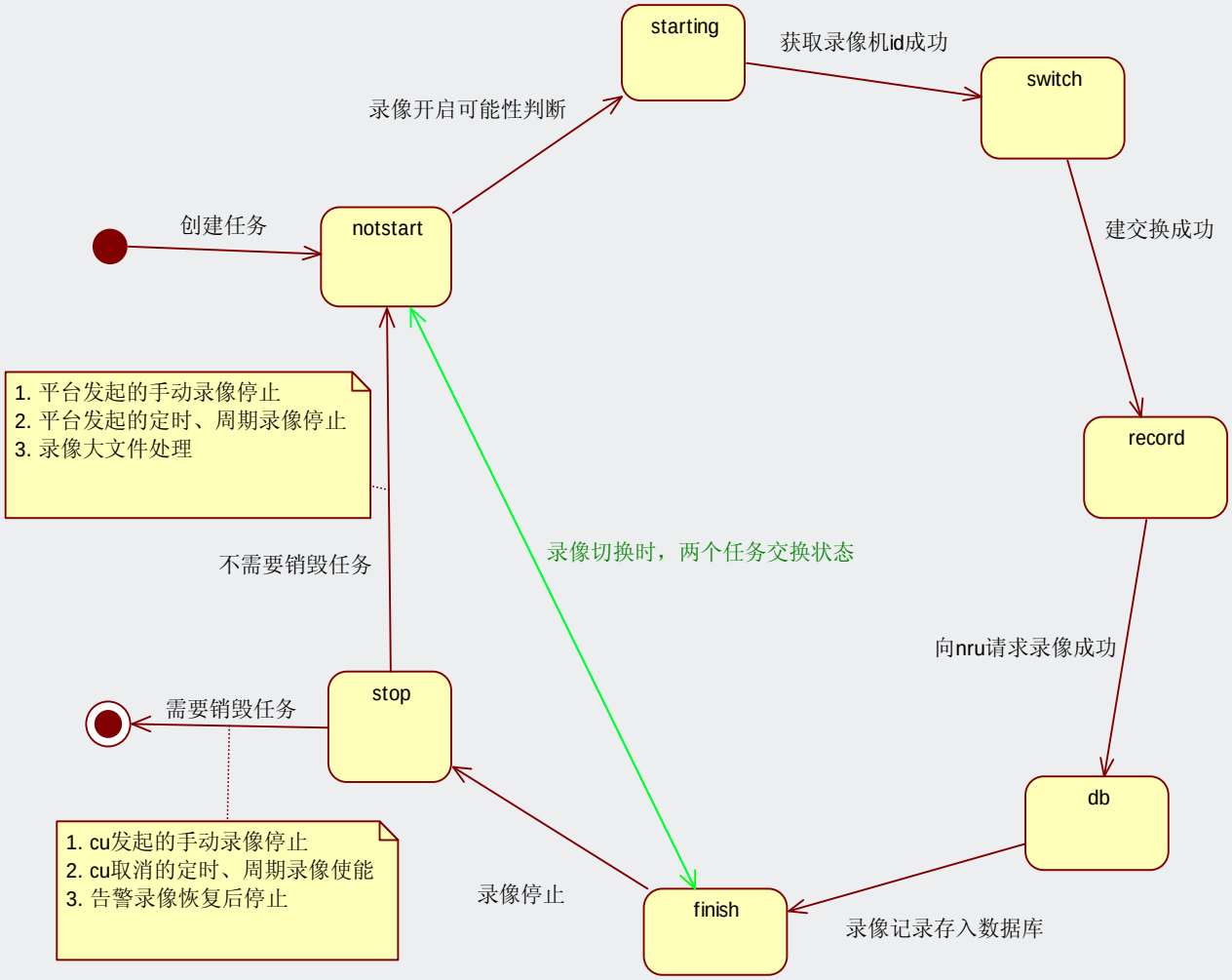
代码在这里：

Src.rar

事务模型实战-平台录像

KEDACOM

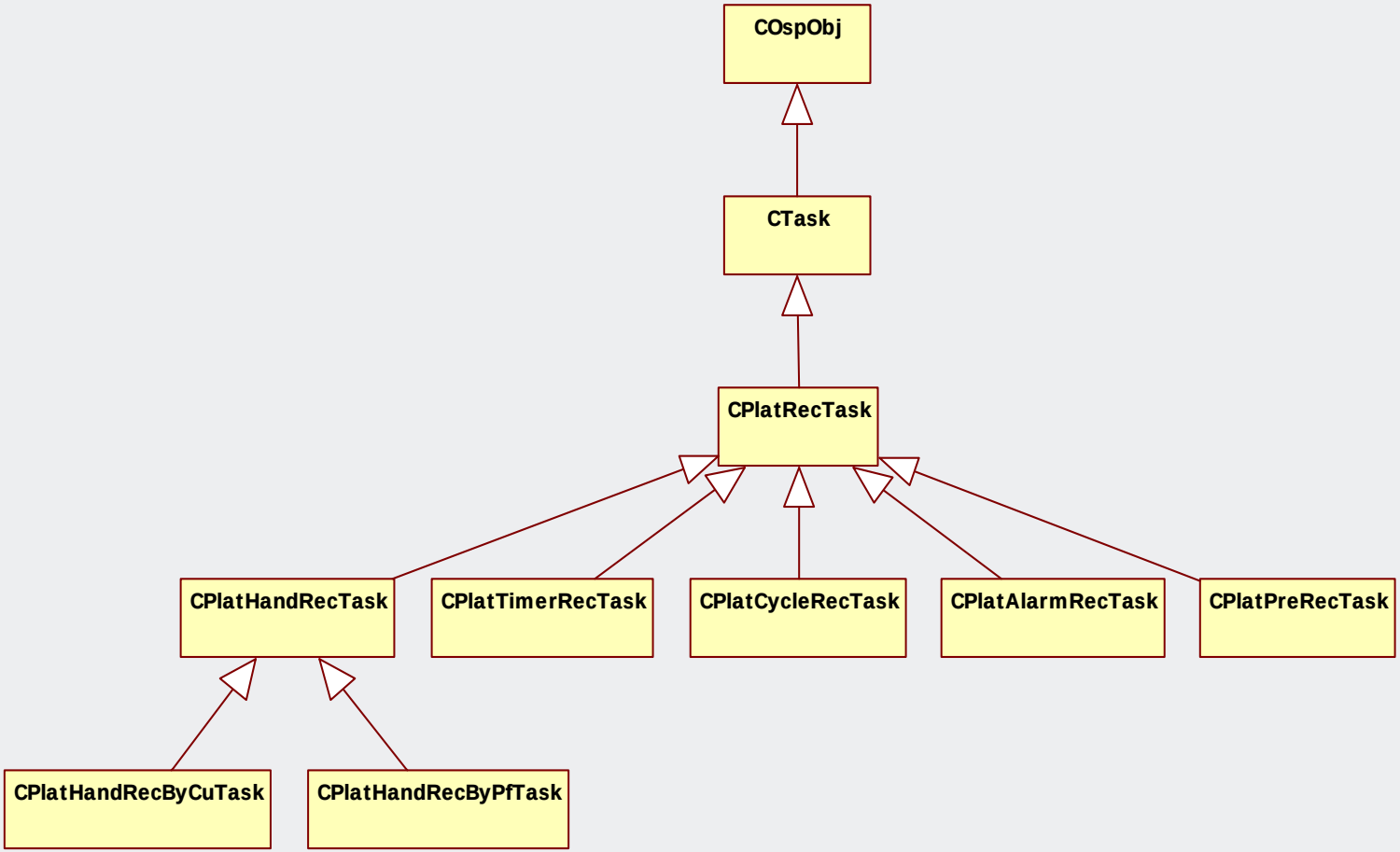
- 平台录像-状态图



事务模型实战-平台录像

KEDACOM

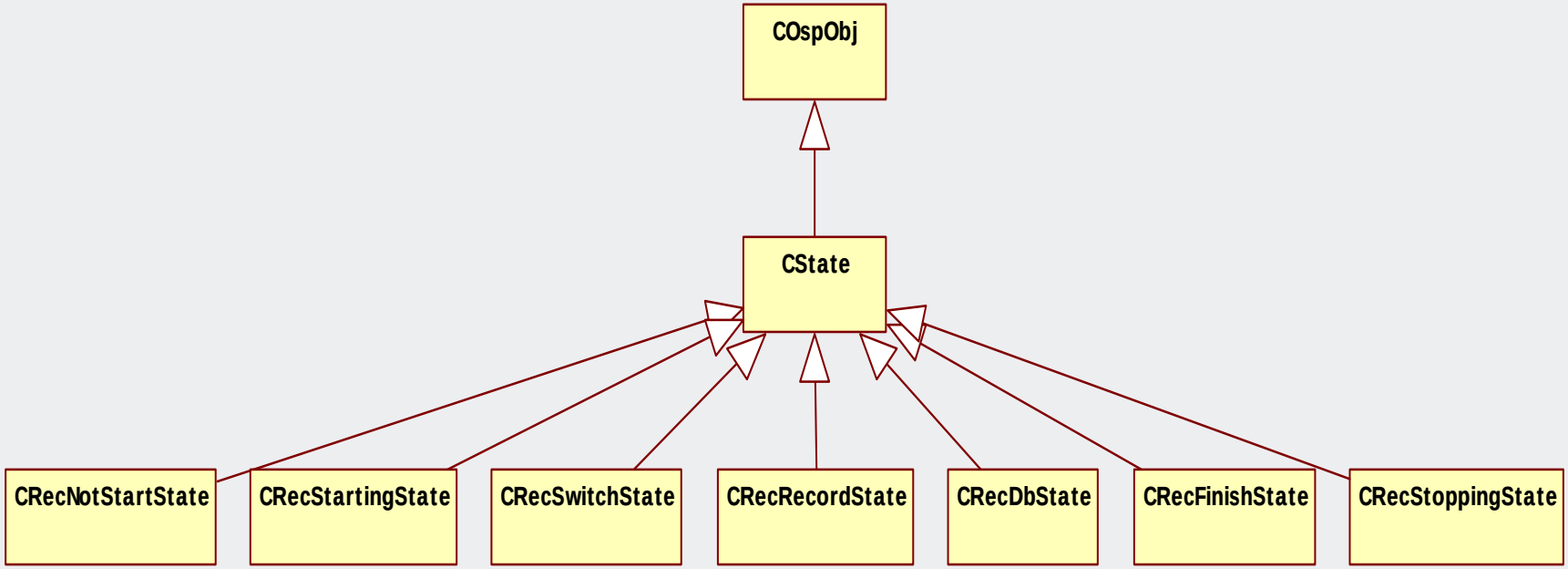
- 平台录像-事务类图



事务模型实战-平台录像

KEDACOM

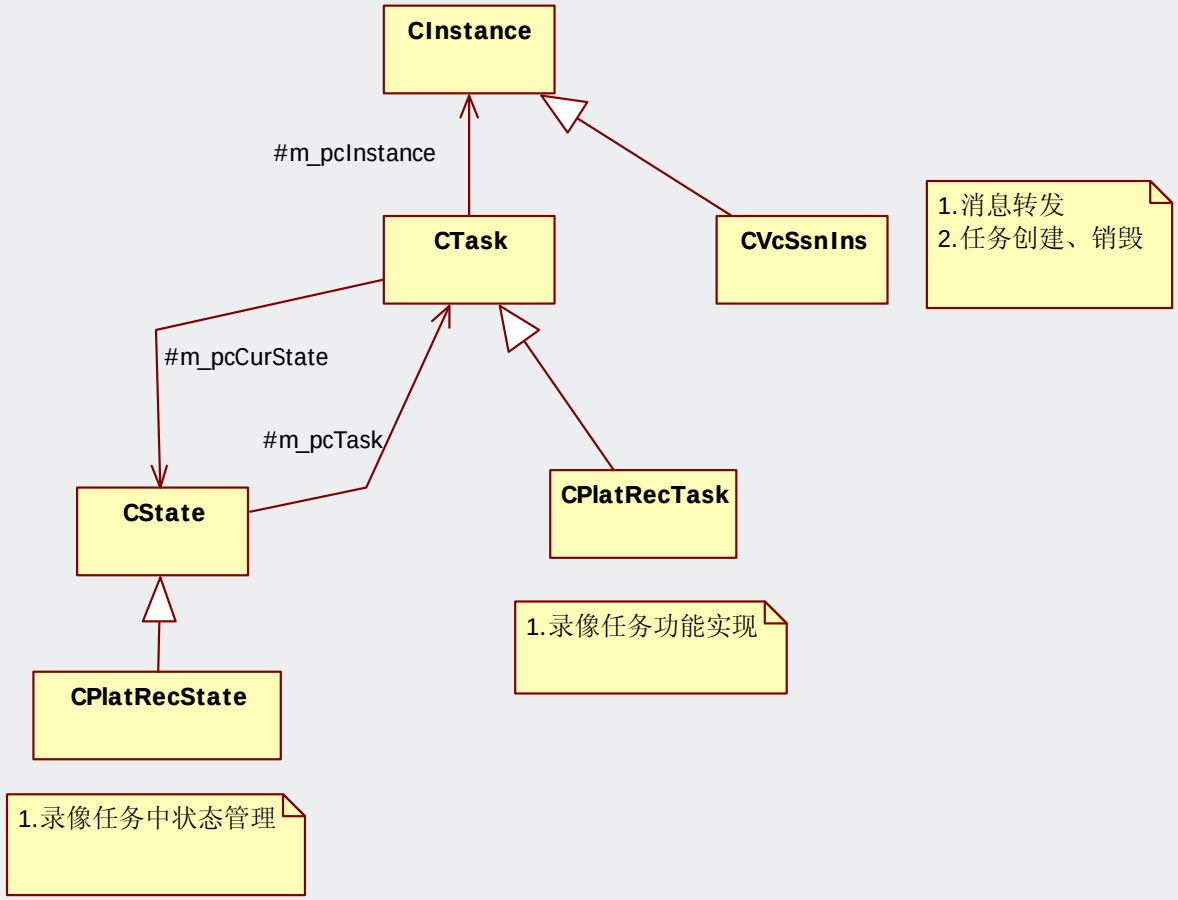
- 平台录像-状态类图



事务模型实战-平台录像

KEDACOM

- 平台录像-类的关系



事务模型-总结

- 优点
  - 将系统业务分解为各种事务
  - 清晰的类层次结构
  - 业务处理的时间性能提高
  - 更加灵活的调试接口
  - 易理解、易维护、易扩展的代码
  - 可以简化通信协议

事务模型-总结

- 缺点
  - 内存需求有一定程度上升
  - 类的数量可能急剧上升
  - 对开发人员的技术要求提高

谢　谢！