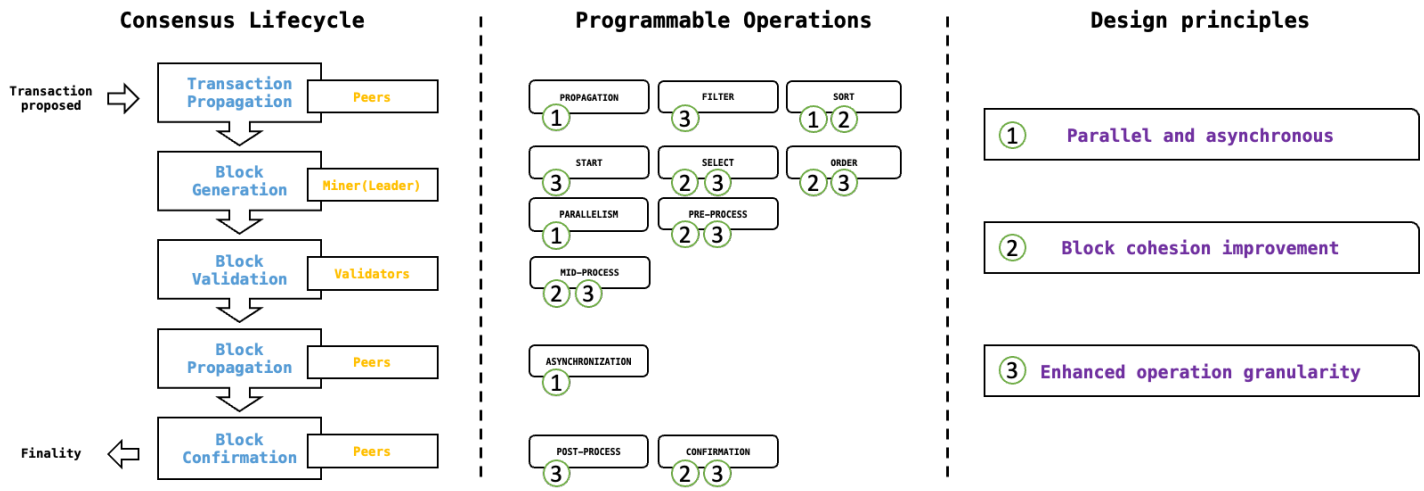
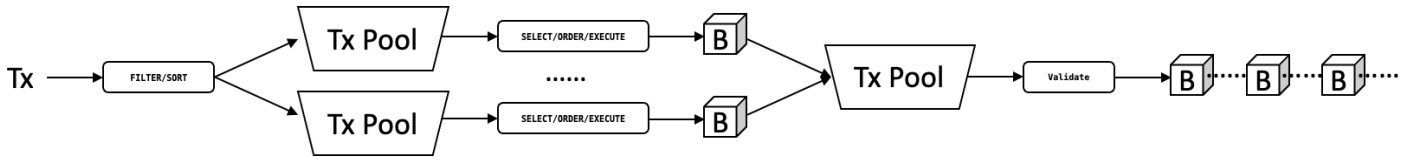
可编程共识

一般化交易流程分析



系统设计



可编程共识排队论分析

首先将一般化的联盟链区块链网络建模成为一个排队网络模型，主要执行步骤分解成为排序、执行、验证三个部分，由三个排队节点组成。我们通过分解简化网络模型，分别求解。

M/M/k系统的性能指标：

系统利用率：$U = \frac{\lambda}{k\mu}$

某阶段排队长度：$E_L = \frac{(kU)^k U}{k!(1-U)^2}\pi_0$

交易数量期望（队伍长度）：$E_N = E_L + kU$

响应时间期望：$E_R = \frac{E_N}{\lambda}$

停留时间期望：$E_T = E_R + \frac{1}{\mu}$

其中，$\pi_0 = \left[\sum_{i=0}^{k-1} \frac{(kU)^i}{i!} + \frac{(kU)^k}{k!(1-U)}\right]^{-1}$

0. 前提假设

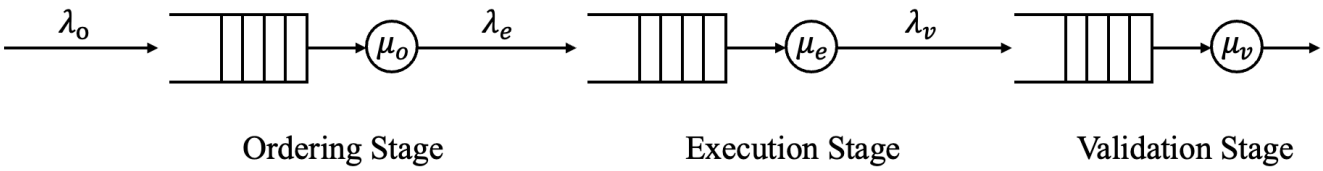
a. 交易有效概率

建模中考虑到当执行阶段和验证阶段存在交易冲突的时候，无效交易需要重新执行，因此假定这两个阶段存在**交易有效概率**P_e和P_v。因此，在执行阶段和验证阶段均有$1-P_e$和$1-P_v$的交易无效，执行结束后返回排序阶段重新执行。

b. 系统总服务率的优化提升

在构建我们的排队论模型时，核心的前提假设是系统总服务率可以通过交易分类和资源分配策略得到优化和提升。通过将交易基于其需求和特点进行分流，我们可以实现对服务资源的最大化利用，从而在不增加物理资源的前提下提升系统的处理能力。

1. Baseline 1：M/M/1+M/M/1+M/M/1单队列单窗口模型



a. 排序阶段

交易初始到达率为λ，考虑执行和验证阶段存在的交易无效概率，则

排序阶段交易到达率$\lambda_o = \lambda + (1-P_e)\lambda_e + (1-P_v)\lambda_v$，服务率为$\mu_o$

系统利用率：$U_o = \frac{\lambda_o}{\mu_o}$

交易数量期望：$E_{No} = \frac{\lambda_o}{\mu_o - \lambda_o}$

停留时间期望：$E_{T_o} = \frac{1}{\mu_o - \lambda_o}$

b. 执行阶段

执行阶段交易到达率$\lambda_e = \mu_o$，服务率为μ_e

系统利用率：$U_e = \frac{\lambda_e}{\mu_e}$

交易数量期望：$E_{Ne} = \frac{\lambda_e}{\mu_e - \lambda_e}$

停留时间期望：$E_{T_e} = \frac{1}{\mu_e - \lambda_e}$

c. 验证阶段

执行阶段交易到达率$\lambda_v = P_e\mu_e$，服务率为μ_v

系统利用率：$U_v = \frac{\lambda_v}{\mu_v}$

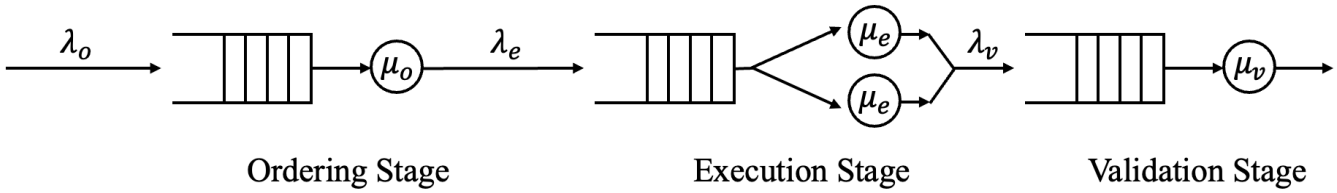
交易数量期望：$E_{Nv} = \frac{\lambda_v}{\mu_v - \lambda_v}$

停留时间期望：$E_{T_v} = \frac{1}{\mu_v - \lambda_v}$

d. 三阶段模型串联

交易总停留时间：$E_T = \frac{1}{\mu_o - \lambda_o} + \frac{1}{\mu_e - \lambda_e} + \frac{1}{\mu_v - \lambda_v}$

2. Baseline 2：M/M/1+M/M/k+M/M/1执行阶段多窗口模型



a. 排序阶段

交易初始到达率为λ，考虑执行和验证阶段存在的交易无效概率，则

排序阶段交易到达率$\lambda_o = \lambda + (1 - P_e)\lambda_e + (1 - P_v)\lambda_v$，服务率为$\mu_o$

系统利用率：$U_o = \frac{\lambda_o}{\mu_o}$

交易数量期望：$E_{No} = \frac{\lambda_o}{\mu_o - \lambda_o}$

停留时间期望：$E_{T_o} = \frac{1}{\mu_o - \lambda_o}$

b. 执行阶段

执行阶段交易到达率$\lambda_e = \mu_o$，服务率为$k\mu_e$，服务窗口数为k

系统利用率：$U_e = \frac{\lambda_e}{k\mu_e}$

交易数量期望：$E_{Ne} = \frac{(kU_e)^k U_e}{k!(1-U_e)^2}\pi_0 + \frac{\lambda_e}{\mu_e}$

停留时间期望：$E_{T_e} = \frac{(kU_e)^k U_e}{\lambda_e \cdot k!(1-U_e)^2}\pi_0 + \frac{1}{\mu_e}$

c.验证阶段

执行阶段交易到达率$\lambda_v = P_e\mu_e$，服务率为μ_v

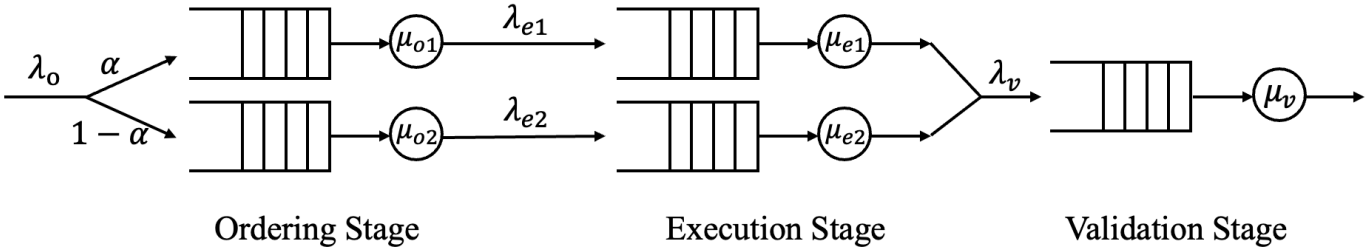
交易数量期望：$E_{Nv} = \frac{\lambda_v}{\mu_v - \lambda_v}$

停留时间期望：$E_{T_v} = \frac{1}{\mu_v - \lambda_v}$

d. 三阶段模型串联

交易总停留时间：$E_T = \frac{1}{\mu_o - \lambda_o} + \frac{(kU_e)^k U_e}{\lambda_e \cdot k!(1-U_e)^2}\pi_0 + \frac{1}{\mu_e} + \frac{1}{\mu_v - \lambda_v}$

3. Ours：(M/M/1+M/M/1)*k+M/M/1多队列多窗口队列模型



a. 排序阶段

交易初始到达率为λ，考虑执行和验证阶段存在的交易无效概率，则

排序阶段交易总到达率$\lambda_o = \lambda + (1 - P_e)\lambda_e + (1 - P_v)\lambda_v$，

通过交易分流操作，交易分流到k个独立的交易和执行队列，在交易打包成区块并执行后汇总到统一的验证队列进行验证上链操作。

为简化计算，设$k = 2$，两个队列具有不同的交易到达率、服务率μ_{o1}，μ_{o2}

系统利用率：$U_o = \frac{\alpha\lambda_o}{\mu_{o1}} + \frac{(1-\alpha)\lambda_o}{\mu_{o2}}$

交易数量期望：$E_{No} = \frac{\alpha\lambda_o}{\mu_{o1} - \alpha\lambda_o} + \frac{(1-\alpha)\lambda_o}{\mu_{o2} - (1-\alpha)\lambda_o}$

停留时间期望：$E_{T_o} = \frac{\alpha}{\mu_{o1} - \alpha\lambda_o} + \frac{(1-\alpha)}{\mu_{o2} - (1-\alpha)\lambda_o}$

b. 执行阶段

执行阶段队列对应对等的排序窗口，交易到达率$\lambda_{e1} = \mu_{o1}$，$\lambda_{e2} = \mu_{o2}$，服务率为μ_{e1}，μ_{e2}

系统利用率：$U_e = \frac{\lambda_{e1}}{\mu_{e1}} + \frac{\lambda_{e2}}{\mu_{e2}}$

交易数量期望：$E_{Ne} = \frac{\lambda_{e1}}{\mu_{e1}-\lambda_{e1}} + \frac{\lambda_{e2}}{\mu_{e2}-\lambda_{e2}}$

停留时间期望：$E_{T_e} = \frac{1}{\mu_{e1}-\lambda_{e1}} + \frac{1}{\mu_{e2}-\lambda_{e2}}$

c. 验证阶段

执行阶段交易到达率$\lambda_v = P_{e1}\mu_{e1} + P_{e2}\mu_{e2}$，服务率为$\mu_v$

系统利用率：$U_v = \frac{\lambda_v}{\mu_v}$

交易数量期望：$E_{Nv} = \frac{\lambda_v}{\mu_v-\lambda_v}$

停留时间期望：$E_{T_v} = \frac{1}{\mu_v-\lambda_v}$

d. 三阶段模型串联

交易总停留时间：$E_T = \frac{\alpha}{\mu_{o1}-\alpha\lambda_o} + \frac{(1-\alpha)}{\mu_{o2}-(1-\alpha)\lambda_o} + \frac{1}{\mu_{e1}-\lambda_{e1}} + \frac{1}{\mu_{e2}-\lambda_{e2}} + \frac{1}{\mu_v-\lambda_v}$

4. 可编程共识设计理念

a. 区块粒度处理：交易归类

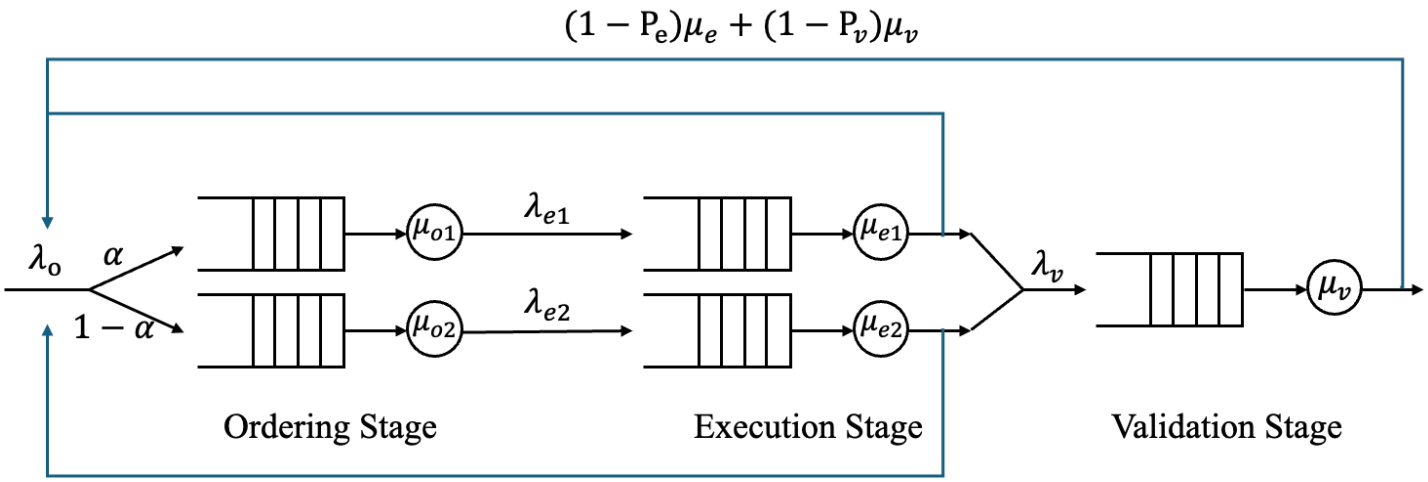
a1.冲突避免

传统的区块链不考虑交易的语义，因此交易处理趋向于无序。（在这样的场景下，区块为单位的批处理更多的是通信等资源开销的考量）因此会存在交易回滚情况，即$P < 1$。

当采用Baseline 2的并行执行策略，交易回滚可能性会进一步提升（乐观冲突并发控制）/各窗口服务率大幅下降（悲观冲突并发控制），即$P_2 < P_1$，$\mu_2 < \mu_1$。

可编程共识通过考虑交易语义，实现了以下的提升：

冲突避免：通过交易初期分流，在我们的模型中提升P（如何分析？）



a2.系统确定性提升

在信息论中，熵是一个度量系统不确定性的量。系统的熵越高，表示系统的不确定性越大，反之亦然。将系统的组件进行有效分类后，理论上我们减少了系统内部的不确定性，从而降低了系统的熵值。熵值的减少对于整个系统来说，意味着更高的效率、更好的预测性、优化的资源分配、提升的用户体验、增强的稳定性和促进的可扩展性，从而显著提高了系统的整体性能和可靠性。

在信息论中，熵是一个度量系统不确定性的量。系统的熵越高，表示系统的不确定性越大，反之亦然。当我们通过某种策略（如你提到的"分配"策略）将系统的组件进行有效分类后，理论上我们减少了系统内部的不确定性，从而降低了系统的熵值。

对于一个离散随机变量X，其熵$H(X)$定义为：
$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$
其中，$p(x_i)$是变量X取特定值x_i的概率，n是X可能的交易类型总数。

不分类单队列系统的熵值H：在不分类的情况下，所有交易都在一个队列中，假设交易类型的分布是均匀的，交易类型总数为 n。那么单个交易属于某一类型的概率是$\frac{1}{n}$。不分类单队列系统的熵值H可以表达为：

$$H = -\sum_{i=1}^{n} p_i \log_2(p_i) = -n\left(\frac{1}{n}\log_2\left(\frac{1}{n}\right)\right) = \log_2(n)$$

分类后双队列系统的熵值H'：分类后，假设队列 A 和队列 B 分别处理 n_A 和 n_B类交易。为了简化，我们假设每个队列内的交易类型分布仍然是均匀的。则队列 A 和队列 B 的熵值分别是：

$$H_A = \log_2(n_A), n_A = \alpha n$$

$$H_B = \log_2(n_B), n_B = (1-\alpha)n$$

$$H' = \alpha \log_2(\alpha n) + (1-\alpha)\log_2((1-\alpha)n)$$

熵值差异$H - H'$：分类策略对系统整体熵的影响，可以被视为信息增益，表示由于分类而获得的额外信息量。这额外的信息来源于我们通过分类减少的不确定性。在实践中，这意味着系统能够更有效率地处理交易，因为每个队列更专注于处理一类交易。

$$H - H' = \log_2(n) - [\alpha \log_2(\alpha n) + (1-\alpha)\log_2((1-\alpha)n)]$$

$$= \log_2(n) - [\alpha(\log_2(\alpha) + \log_2(n)) + (1-\alpha)(\log_2(1-\alpha) + \log_2(n))]$$

$$= \log_2(n) - [\alpha \log_2(\alpha) + (1-\alpha)\log_2(1-\alpha) + \log_2(n)]$$

$$= -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)$$

在该系统$k = 2$的假设当中，熵值差异取决于α，当 $\alpha = 0.5$,即交易类型被平均分配到两个队列）时，均匀分配最大程度减少了总体的不确定性，

$$H - H' = -0.5\log_2(0.5) - 0.5\log_2(0.5)$$

$$= -0.5(-1) - 0.5(-1) = 1$$

所以，当交易类型被平均分配时，信息增益是 1 比特。这表示，通过将交易类型平均分配到两个队列，系统的不确定性减少了一个量化单位。

b. 区块链整体服务率提升：并行处理

采用分类的多队列并行处理模型允许系统更有效地利用各种物理资源（如读写、计算、通信），通过为不同类型的交易分配到特定优化的队列中并行处理，因资源短板效应导致的性能瓶颈，从而实现了整体服务率的提升。这种策略不仅提高了各个资源的利用率，而且通过并行化处理显著提高了系统的处理能力。因此，我们的模型的服务率μ_{ours}超过了传统的串行处理基线模型的服务率μ_{baseline}，而整个系统的理论最大服务率μ_{total}则反映了在理想状态下资源完全利用时的服务能力，即$\mu_{\text{total}} > \mu_{\text{ours}} > \mu_{\text{baseline}}$，体现了通过分类策略提高系统性能的可能。