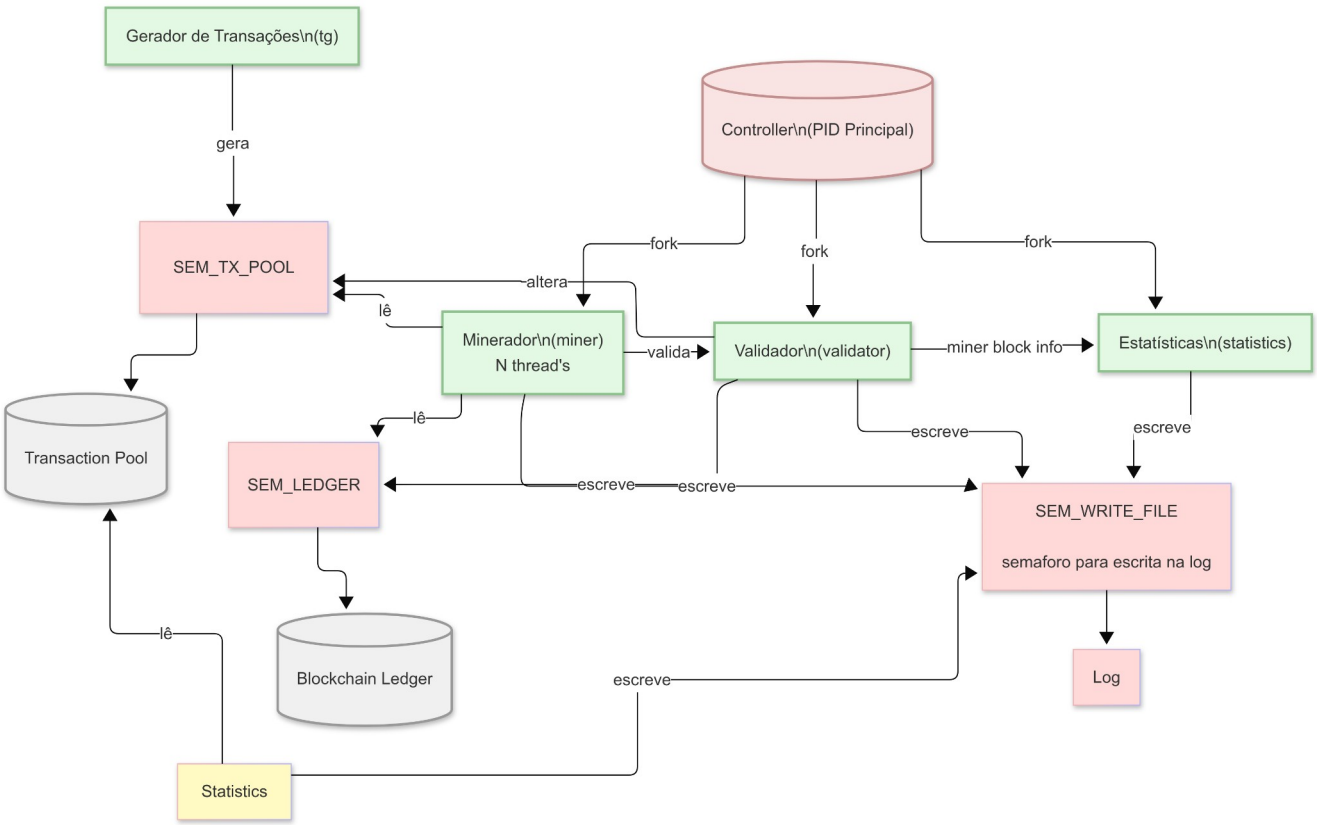

O diagrama acima representa a arquitetura do nosso sistema de processamento de transações em uma blockchain simplificada, com o uso de múltiplas threads, semáforos e componentes sincronizados para evitar problemas como race conditions e desperdício de CPU.

Componentes do Sistema

Gerador de Transações (tx)

Esse componente é responsável por gerar novas transações, que são enviadas para o **Transaction Pool**. Ele simula usuários ou eventos que geram transações na blockchain.

Transaction Pool

É a fila onde ficam armazenadas as transações que ainda não foram processadas pelos mineradores. Para acessar essa pool, usamos o semáforo `SEM_TX_POOL`, garantindo que as threads mineradoras acessem de forma segura.

Minerador (N threads)

O componente de mineração possui várias threads que competem para pegar transações do pool e montar blocos. Aqui foi adicionada a melhoria mencionada no texto: **um segundo semáforo de contagem no Transaction Pool**, com o objetivo de evitar **busy waiting**. Isso significa que a thread não fica ocupando a CPU desnecessariamente enquanto espera, o que otimiza bastante o desempenho.

Validador
Após o bloco ser minerado, ele passa para o validador, que checa se o bloco está correto e válido. Caso esteja tudo certo, o validador encaminha os dados para escrita no log e também para o ledger.

Blockchain Ledger
Após a validação, o bloco é registrado no **Blockchain Ledger**, representando o armazenamento permanente dos blocos válidos.

SEM_LEDGER e SEM_WRITE_FILE
Esses semáforos controlam o acesso ao Ledger e ao arquivo de log, respectivamente, garantindo que não haja corrupção nos dados ao serem escritos por múltiplas threads.

Estatísticas
Um componente que coleta informações do sistema, como quantidade de blocos minerados, tempo de processamento etc. Os dados são usados para análise de desempenho.

Controlador (PID Principal)
Esse processo principal é responsável por iniciar as threads do sistema, como os mineradores e validadores. Ele usa `fork` para criar os subprocessos.

Implementações após a meta intermediária

- **Semáforo adicional no Transaction Pool**: Para evitar que as threads fiquem checando continuamente se há novas transações (busy waiting), foi adicionado um semáforo de contagem que só libera as threads quando há algo novo no pool.
- **Uso de pausas**: Também foi implementado o uso de “pause” em locais estratégicos do código, o que evita o desperdício de CPU e melhora a eficiência geral do sistema.

Conclusão

Esse sistema simula de forma eficiente como funcionaria uma blockchain simplificada com controle de concorrência. As melhorias feitas, como o uso inteligente de semáforos e pausas, mostram como pequenas mudanças podem melhorar muito o desempenho e o consumo de recursos. Essa arquitetura é útil para entender como sistemas reais lidam com concorrência e sincronização.