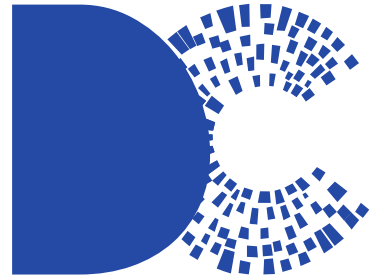


Paralelismo de software

Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

Introdução

- Desempenho em hardware
  - *Pipeline*
    - Aumento da taxa de execução
    - Melhor aproveitamento do hardware

Introdução

- Desempenho em hardware
  - *Pipeline*
    - Aumento da taxa de execução
    - Melhor aproveitamento do hardware
  - *Superescalar*
    - Paralelismo entre instruções
    - Aumento do desempenho de execução

Introdução

- Desempenho em hardware
  - *Pipeline*
    - Aumento da taxa de execução
    - Melhor aproveitamento do hardware
  - *Superescalar*
    - Paralelismo entre instruções
    - Aumento do desempenho de execução
  - Multiprocessamento
    - Paralelismo de processo e *thread*
    - O software precisa explorar o paralelismo

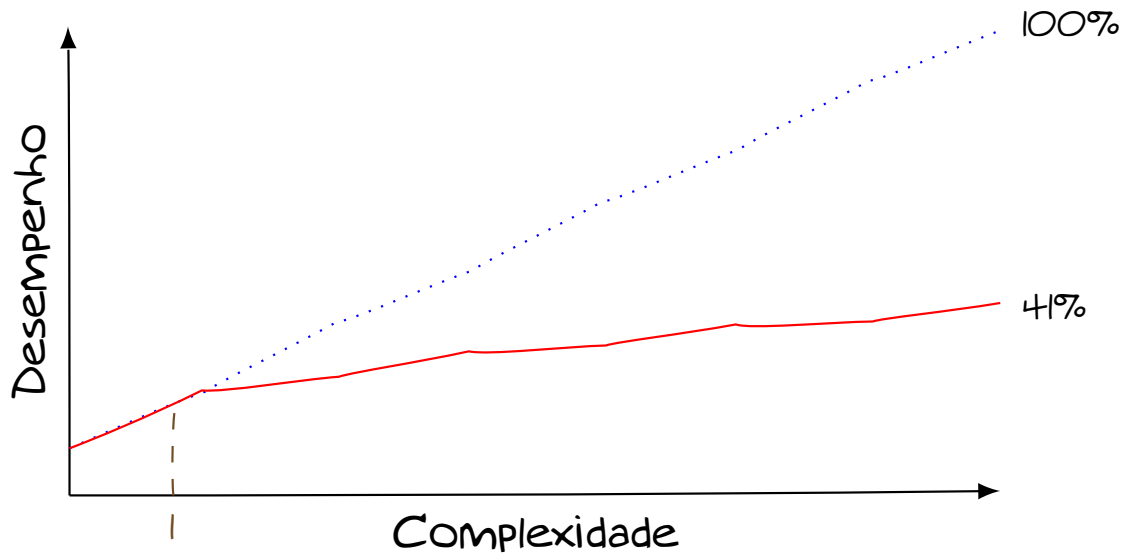
Introdução

- Desempenho em hardware
  - Aumento da complexidade do projeto

Introdução

- Desempenho em hardware
  - Aumento da complexidade do projeto

  - Regra de Pollack
    - O aumento de desempenho em processadores é aproximadamente proporcional à raiz quadrada do incremento de complexidade do projeto



Introdução

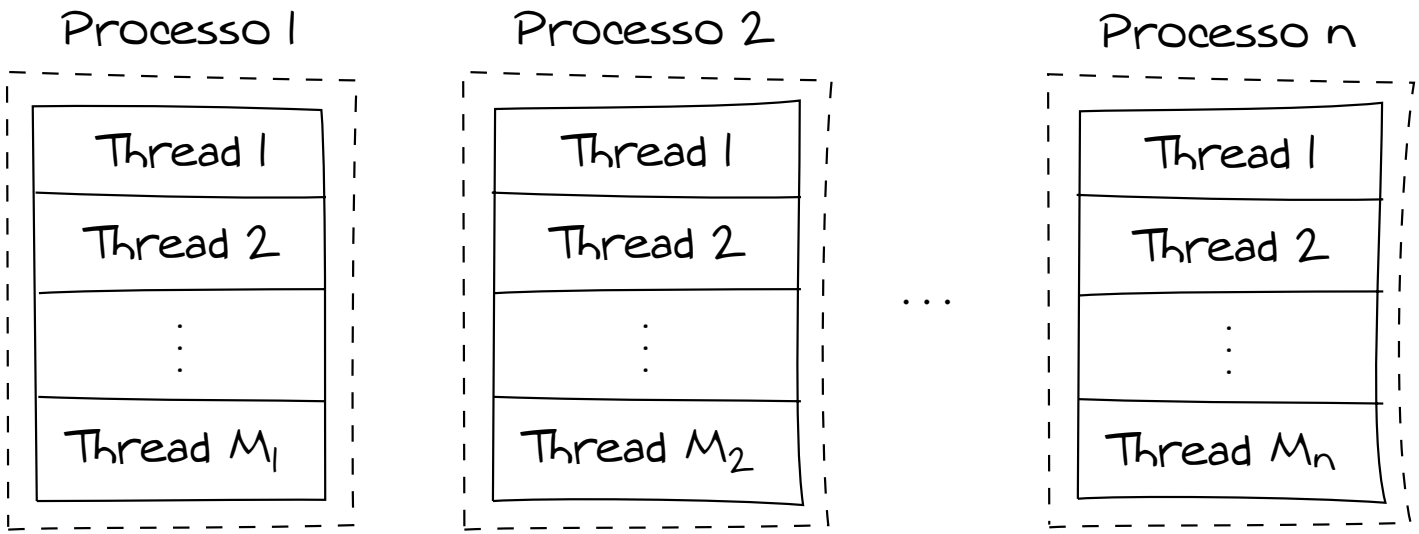
- Qual a diferença entre processo e *thread*?
  - Processo
    - É uma instância independente de uma aplicação que executa com escalonamento feito pelo SO
    - Contexto + Memória Virtual + Recursos alocados

Introdução

- Qual a diferença entre processo e *thread*?
  - Processo
    - É uma instância independente de uma aplicação que executa com escalonamento feito pelo SO
    - Contexto + Memória Virtual + Recursos alocados
  - *Thread*
    - Só existe como parte de um processo e seu escalonamento pode ser feito pelo programador
    - Utiliza os mesmos recursos do processo

Introdução

- Qual a diferença entre processo e *thread*?
  - Processo
    - É uma instância independente de uma aplicação que executa com escalonamento feito pelo SO
    - Contexto + Memória Virtual + Recursos alocados
  - *Thread*
    - Só existe como parte de um processo e seu escalonamento pode ser feito pelo programador
    - Utiliza os mesmos recursos do processo



Introdução

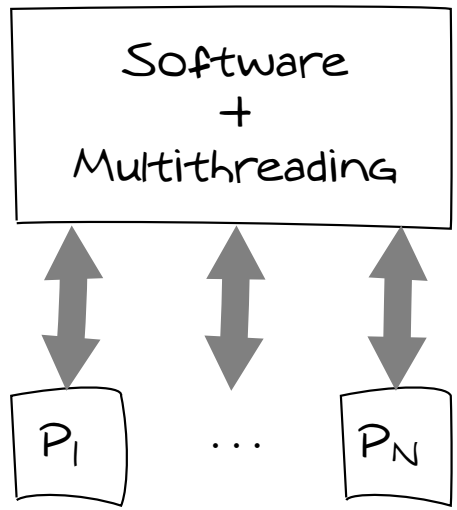
- Desempenho em software
  - Multiprogramação
    - Permite a execução concorrente de múltiplos processos durante um mesmo período de tempo
    - Em plataformas multiprocessadas, os processos podem ser executar paralelamente em cada processador

Introdução

- Desempenho em software
  - Multiprogramação
    - Permite a execução concorrente de múltiplos processos durante um mesmo período de tempo
    - Em plataformas multiprocessadas, os processos podem ser executar paralelamente em cada processador
  - *Multithreading*
    - Cria um ambiente de execução concorrente dentro do processo para maximizar o uso dos recursos
    - Um processo com *multithreading* pode ser paralelizado entre os núcleos de processamento

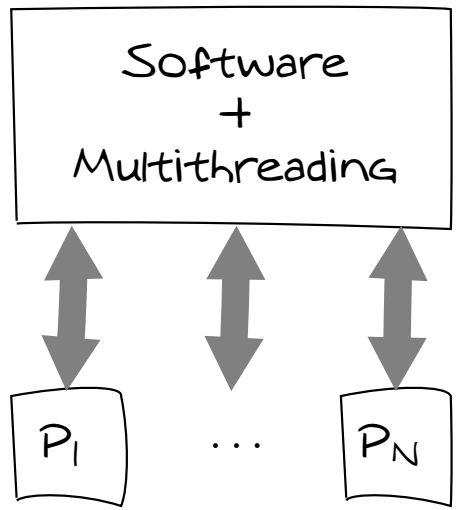
Introdução

- Qual é o limite de aumento do desempenho?
  - Hardware $\times$ Software



Introdução

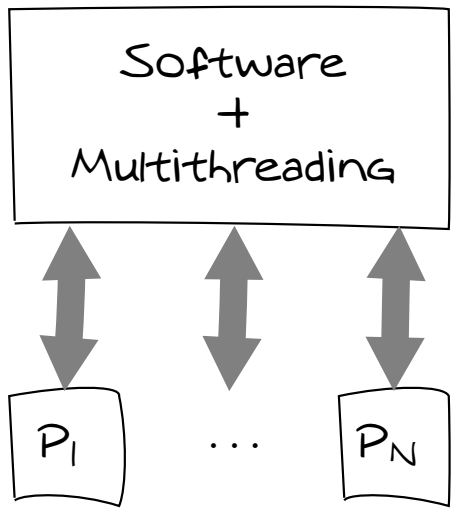
- Qual é o limite de aumento do desempenho?
  - Hardware $\times$ Software



$$\uparrow \textit{Processadores} \xrightarrow{?} \uparrow \textit{Desempenho}$$

Introdução

- Qual é o limite de aumento do desempenho?
  - Hardware $\times$ Software



$$\uparrow \textit{Threads} \xrightarrow{?} \uparrow \textit{Desempenho}$$

Introdução

- Lei de Amdahl
  - A melhoria de desempenho está limitada a parte S do software que é sequencial e não a parte $P = 1 - S$ que pode ser paralelizada em N processadores

$$Amdahl(N) = \frac{Uniprocessador}{Multiprocessador}$$

Introdução

- Lei de Amdahl
  - A melhoria de desempenho está limitada a parte S do software que é sequencial e não a parte $P = 1 - S$ que pode ser paralelizada em N processadores

$$\begin{aligned} Amdahl(N) &= \frac{Uniprocessador}{Multiprocessador} \\ &= \frac{S + P}{S + \frac{P}{N}} \end{aligned}$$

Introdução

- Lei de Amdahl
  - A melhoria de desempenho está limitada a parte S do software que é sequencial e não a parte $P = 1 - S$ que pode ser paralelizada em N processadores

$$
\begin{aligned}
Amdahl(N) &= \frac{Uniprocessador}{Multiprocessador} \\
&= \frac{S + P}{S + \frac{P}{N}} \\
&= \frac{S + (1 - S)}{S + \frac{P}{N}}
\end{aligned}
$$

Introdução

- Lei de Amdahl
  - A melhoria de desempenho está limitada a parte S do software que é sequencial e não a parte $P = 1 - S$ que pode ser paralelizada em N processadores

$$
\begin{aligned}
Amdahl(N) &= \frac{Uniprocessador}{Multiprocessador} \\
&= \frac{S + P}{S + \frac{P}{N}} \\
&= \frac{S + (1 - S)}{S + \frac{P}{N}} \\
&= \frac{1}{S + \frac{P}{N}}
\end{aligned}
$$

Introdução

- Lei de Amdahl
  - Na análise de código de um software, foi detectado que 1% de seu fluxo de execução é sequencial
  - Para a execução do software podem ser utilizados um número infinito de unidades de processamento

$$\lim_{N \to \infty} Amdahl(N) = \frac{1}{(1 - P) + \frac{P}{N}}$$

Introdução

- Lei de Amdahl
  - Na análise de código de um software, foi detectado que 1% de seu fluxo de execução é sequencial
  - Para a execução do software podem ser utilizados um número infinito de unidades de processamento

$$\lim_{N \to \infty} Amdahl(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

$$= \frac{1}{0,01 + \frac{0,99}{N}}$$

Introdução

- Lei de Amdahl
  - Na análise de código de um software, foi detectado que 1% de seu fluxo de execução é sequencial
  - Para a execução do software podem ser utilizados um número infinito de unidades de processamento

$$\lim_{N \to \infty} Amdahl(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

$$= \frac{1}{0,01 + \frac{0,99}{N}}$$

$$= \frac{1}{0,01 + \frac{0,99}{N} \to 0}$$

Introdução

- Lei de Amdahl
  - Na análise de código de um software, foi detectado que 1% de seu fluxo de execução é sequencial
  - Para a execução do software podem ser utilizados um número infinito de unidades de processamento

$$\begin{aligned}
\lim_{N \to \infty} Amdahl(N) &= \frac{1}{(1-P) + \frac{P}{N}} \\
&= \frac{1}{0,01 + \frac{0,99}{N}} \\
&= \frac{1}{0,01 + \frac{0,99}{N} \to 0} \\
&= 100
\end{aligned}$$

Paralelismo de dados

- Processamento vetorial (SIMD)
  - Operações com vetores

```
1 // Multiplicação escalar de vetor
2 void mult(int32_t k, int32_t V[], uint32_t n) {
3     // Controle iterativo
4     for(uint32_t i = 0; i < n; i++) {
5         // Multiplicação escalar
6         V[i] = k * V[i];
7     }
8 }
```

Paralelismo de dados

- Processamento vetorial (SIMD)
  - Operações com vetores

```
1  // i = 0
2  mov r1, k
3  mov r2, n
4  mov r3, 0
5  loop:
6      // i < n
7      cmp r3, r2
8      bae 5
9      // V[i] = k * V[i]
10     l32 r5, [V + r4]
11     mul r5, r5, r2
12     s32 [V + r4], r5
13     // i++
14     addi r4, r4, 1
15     bun -7
```

Escalar

```
1  // V[i] = k * V[i]
2  mov r1, k
3  mov r2, V
4  mov r3, n
5  l32v r2, r3
6  mulvs r1
7  s32v r2, r3
```

Vetorial

Paralelismo de dados

- Processamento escalar x vetorial
  - Repetição de operações sem utilização de laços, diminuindo a busca e decodificação de instruções

Paralelismo de dados

- Processamento escalar x vetorial
  - Repetição de operações sem utilização de laços, diminuindo a busca e decodificação de instruções
  - Não existem conflitos de dados em operações vetoriais, devido a independência dos dados

Paralelismo de dados

- Processamento escalar x vetorial
  - Repetição de operações sem utilização de laços, diminuindo a busca e decodificação de instruções
  - Não existem conflitos de dados em operações vetoriais, devido a independência dos dados
  - Como não existem laços, não existem conflitos de controle durante na predição de desvio

Paralelismo de dados

- Processamento paralelo (MIMD)
  - Programação com *threads*

```
1  // POSIX thread
2  #include <pthread.h>
3  // Multiplicação escalar de vetor
4  void mult_pt(int32_t k, int32_t V[], uint32_t n,
       uint32_t ID, uint32_t NP) {
5      // Índices
6      uint32_t N = n / NP, I = ID * N;
7      // Controle iterativo
8      for(uint32_t i = I; i < I + N; i++) {
9          // Multiplicação escalar
10         V[i] = k * V[i];
11     }
12 }
```

Programação paralela com organização particionamento explícito dos dados

Paralelismo de dados

- Processamento paralelo (MIMD)
  - Programação com OpenMP

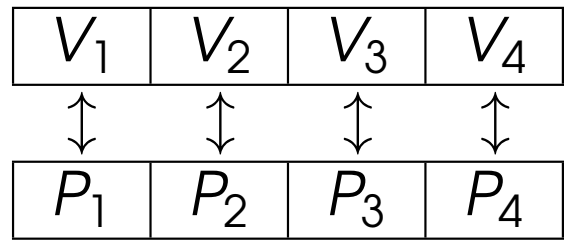
```
1  // OpenMP
2  #include <omp.h>
3  // Multiplicação escalar de vetor
4  void mult_omp(int32_t k, int32_t V[], uint32_t n) {
5      // Controle iterativo
6      #pragma omp parallel for
7      for(uint32_t i = 0; i < n; i++) {
8          // Multiplicação escalar
9          V[i] = k * V[i];
10     }
11 }
```

Programação paralela de alto nível
utilizando diretivas de compilação

Paralelismo de dados

- Processamento paralelo (MIMD)
  - Plataforma com 4 processadores
  - São criadas threads para execução das operações

$$V = \begin{bmatrix} V_1 & \cdots & V_1 \\ V_2 & \cdots & V_2 \\ V_3 & \cdots & V_3 \\ V_4 & \cdots & V_4 \end{bmatrix} \longrightarrow$$



Os dados da matriz são particionados e processados paralelamente por cada um dos núcleos

Paralelismo de *thread*

- Para que uma plataforma multiprocessada com *N* processadores tenha seus recursos devidamente aproveitados, devem existir pelo menos *N* processos ou *threads* em execução no sistema
  - Uniprocessamento: pseudo paralelismo de processos
  - *Superescalar*: paralelismo em nível de instrução
  - Multiprocessamento: paralelismo em nível de *thread*

Paralelismo de *thread*

- Paralelismo em nível de *thread* (TLP)
  - A principal vantagem da *thread* sobre o processo está na troca de contexto muito mais rápida

Paralelismo de *thread*

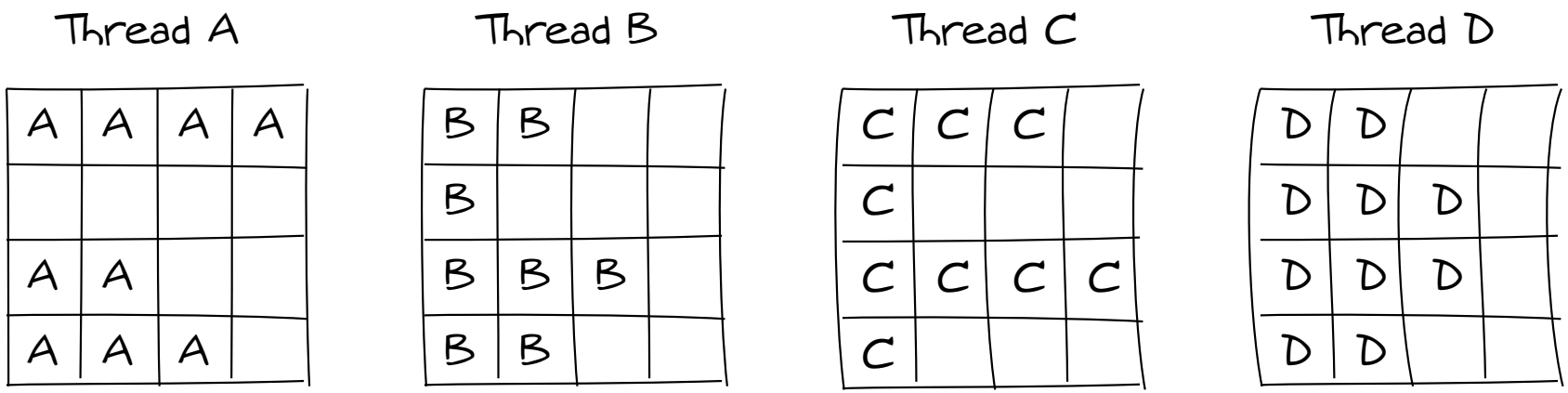
- Paralelismo em nível de *thread* (TLP)
  - A principal vantagem da *thread* sobre o processo está na troca de contexto muito mais rápida
  - Com o suporte de *multithreading* em hardware, cada *thread* possui sua própria cópia dos registradores e mecanismos para otimizar seu escalonamento

Paralelismo de *thread*

- Paralelismo em nível de *thread* (TLP)
  - A principal vantagem da *thread* sobre o processo está na troca de contexto muito mais rápida
  - Com o suporte de *multithreading* em hardware, cada *thread* possui sua própria cópia dos registradores e mecanismos para otimizar seu escalonamento
  - A forma como as instruções das *threads* são executadas no processador é definida pela granularidade de execução

Paralelismo de *thread*

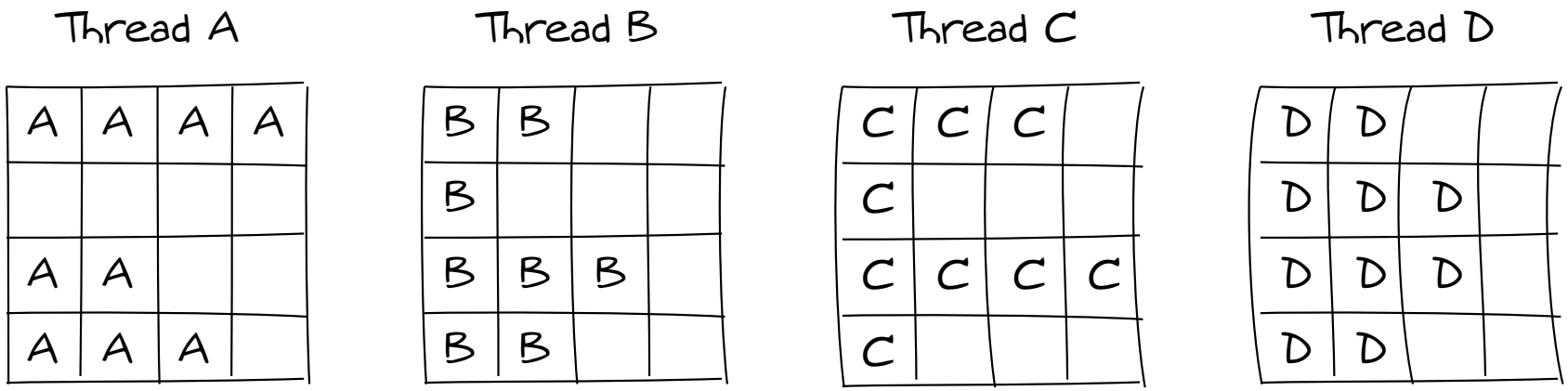
- *Fine-grained Multithreading*
  - Nesta abordagem as *threads* do sistema são escalonadas com alta granularidade de execução
  - As instruções das diversas *threads* são intercaladas pelo processador a cada busca de instrução
  - Aquelas *threads* que estão em estado de espera não são executadas pelo processador

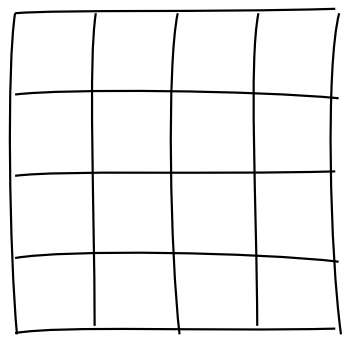


Execução das threads em um superescalar de 4 instruções

Paralelismo de *thread*

- *Fine-grained Multithreading*

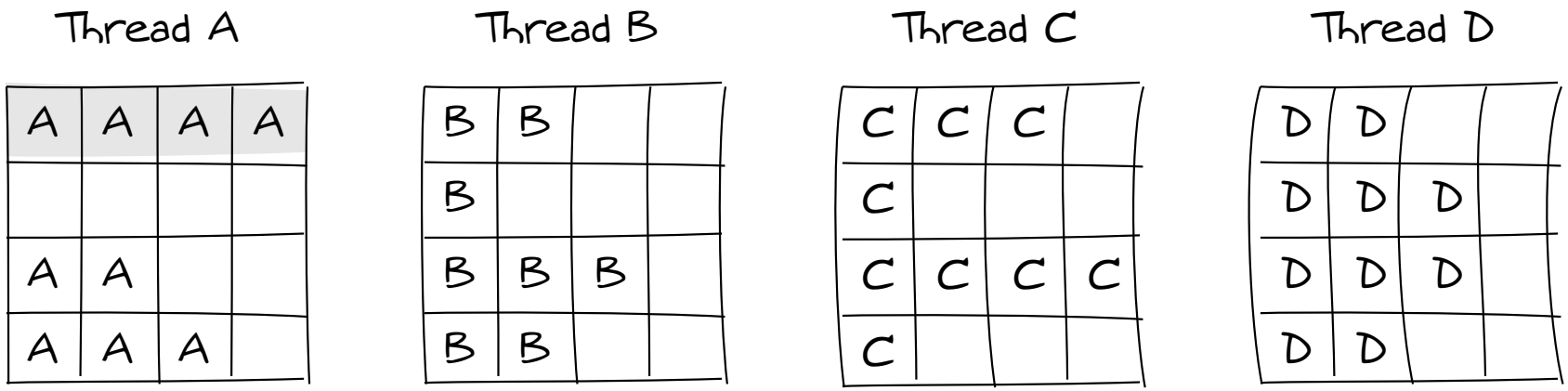


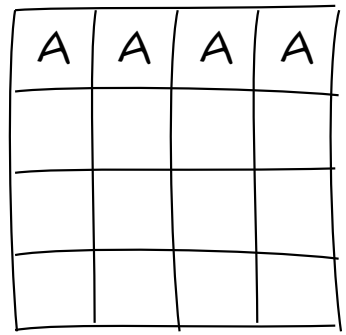


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

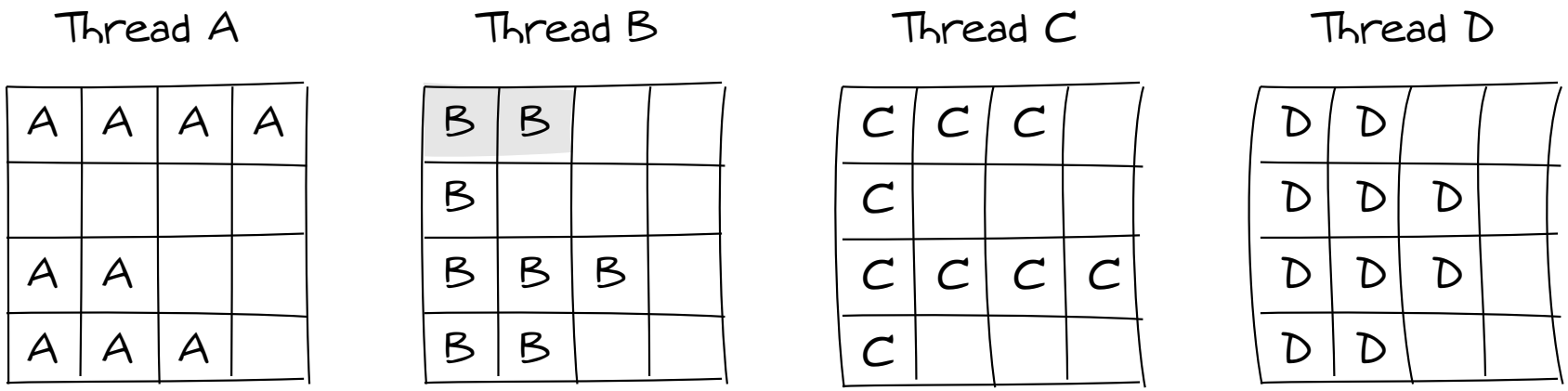


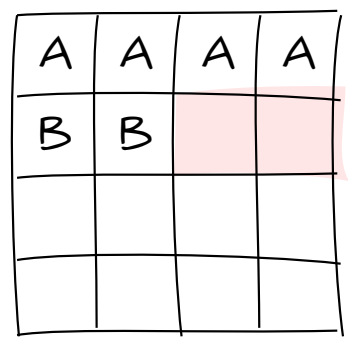


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

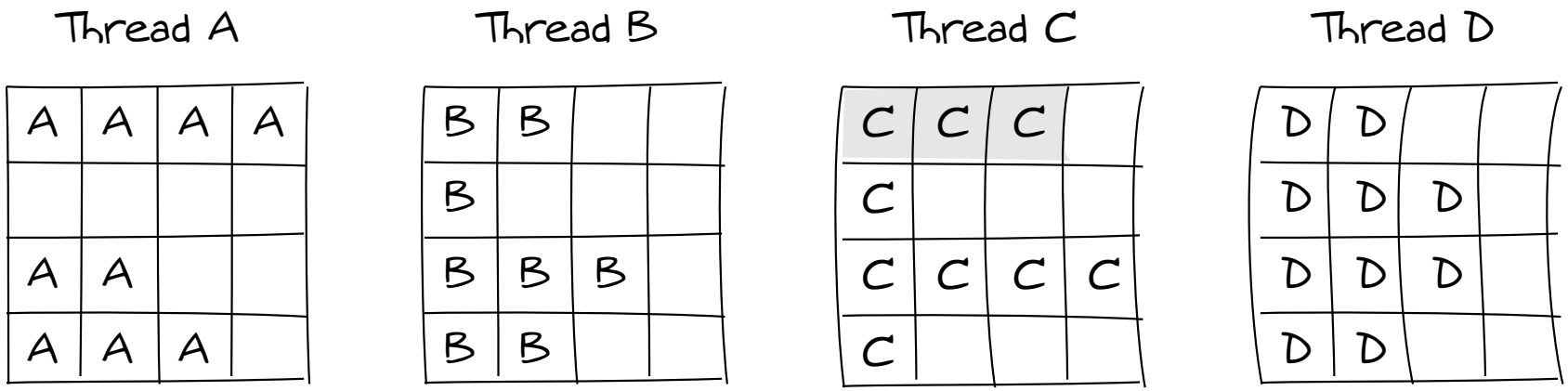


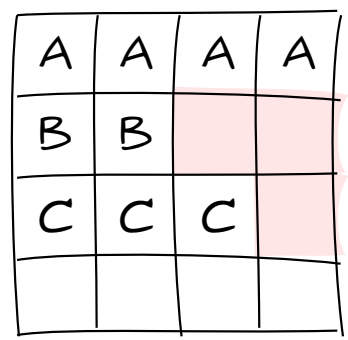


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

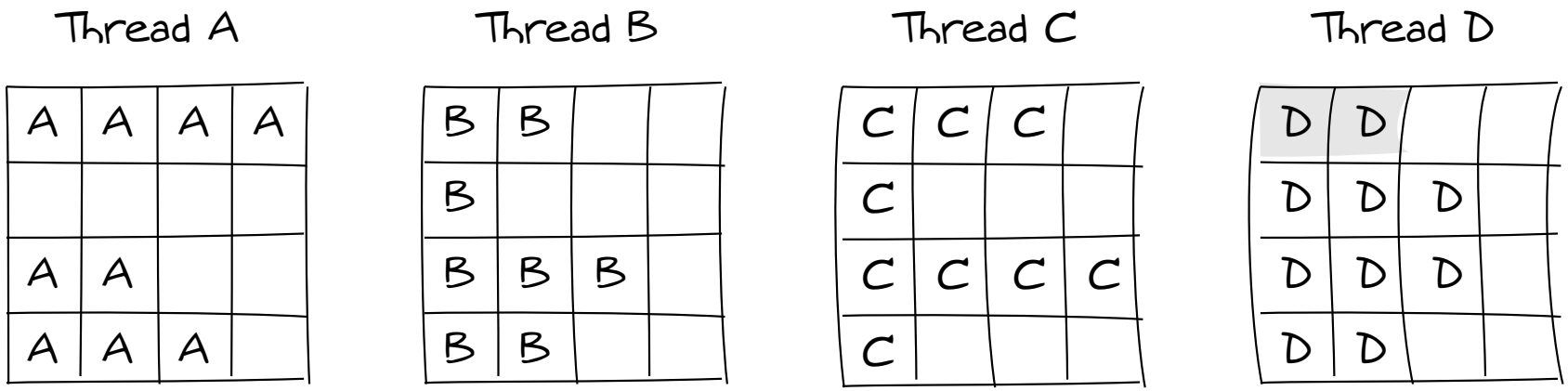


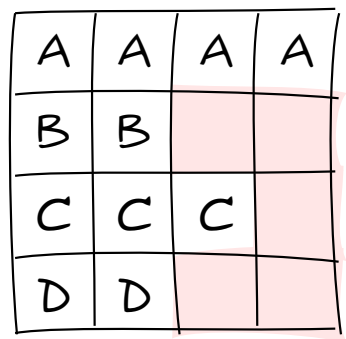


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

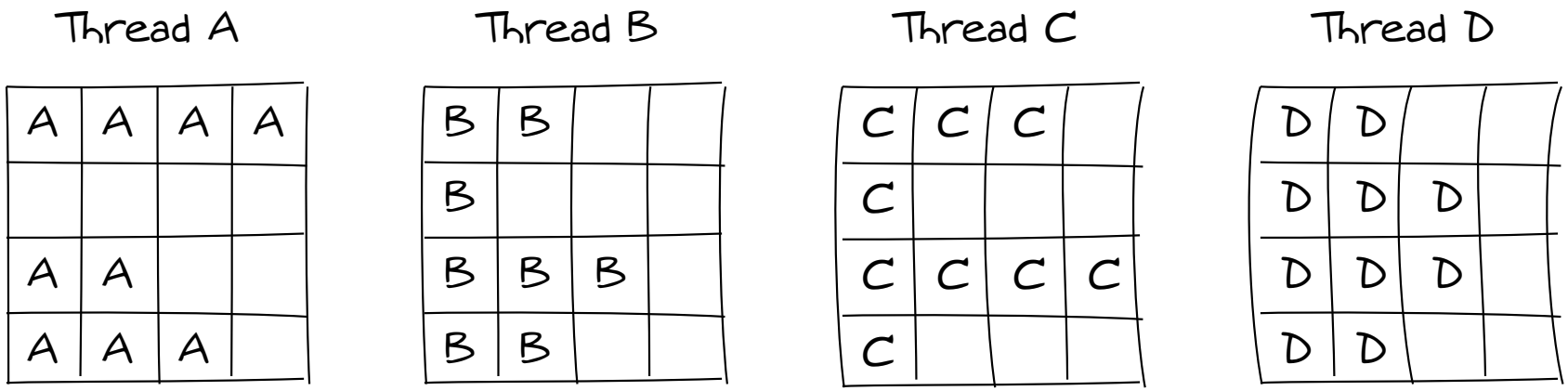


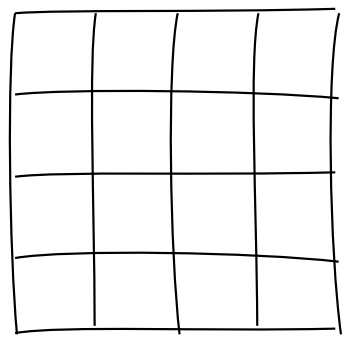


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

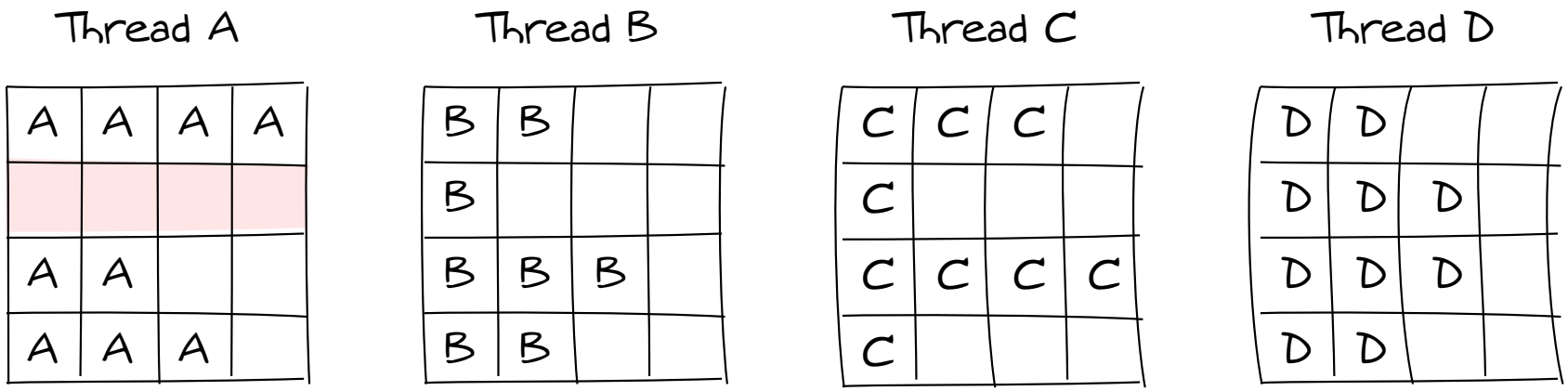


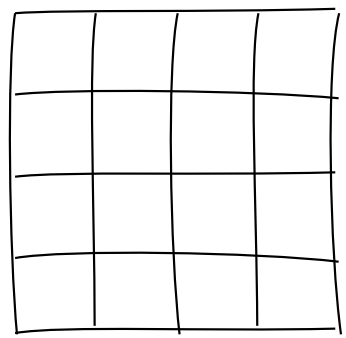


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

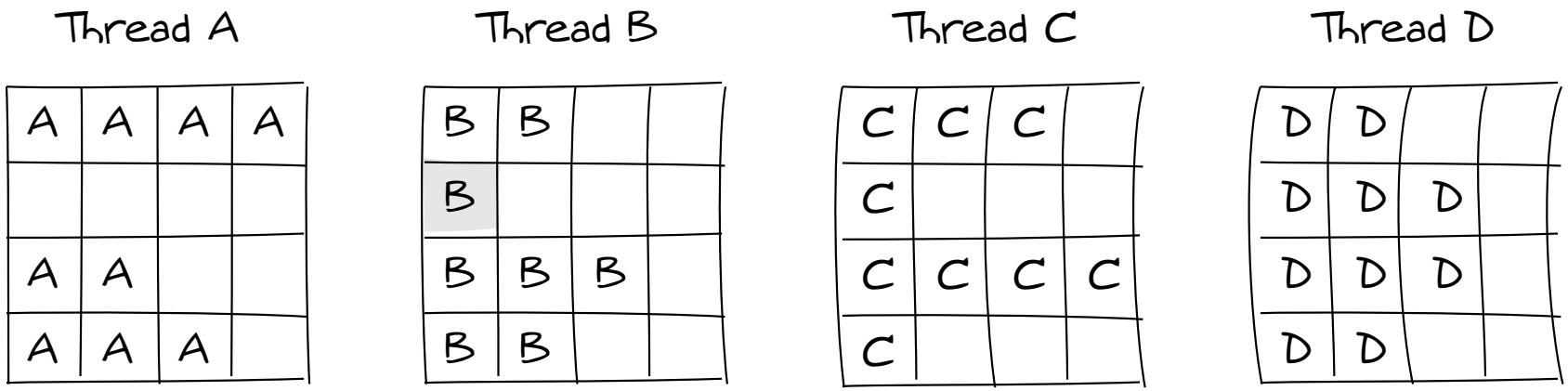


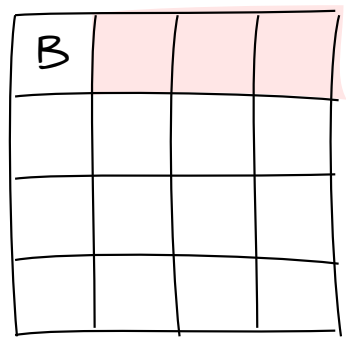


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

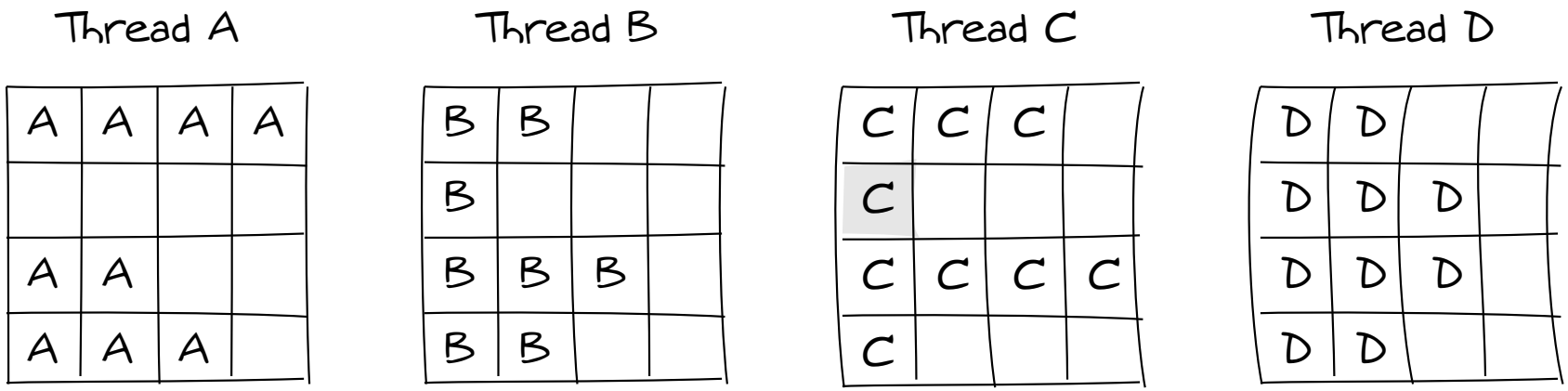


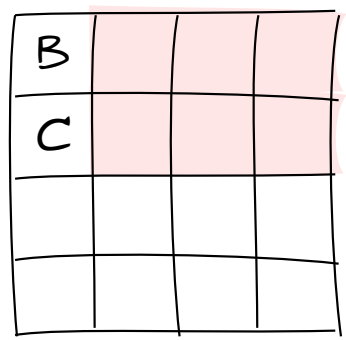


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

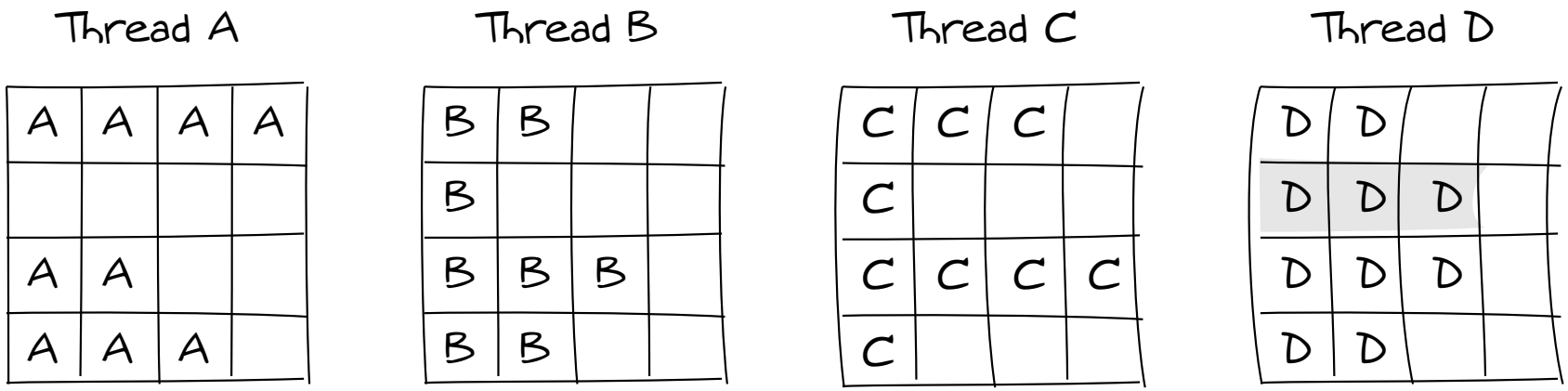


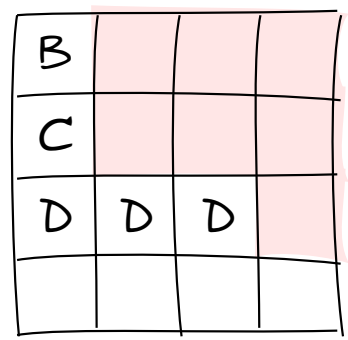


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

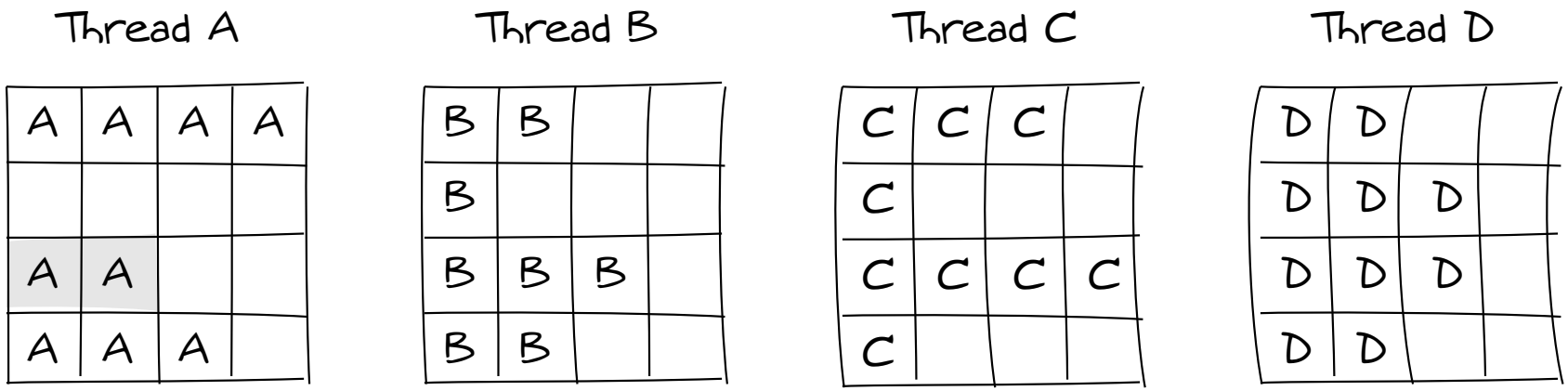


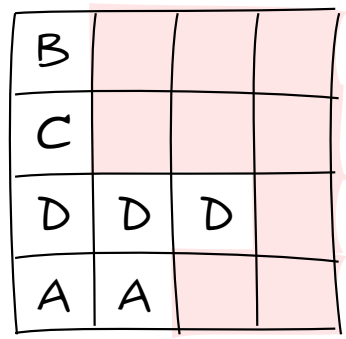


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

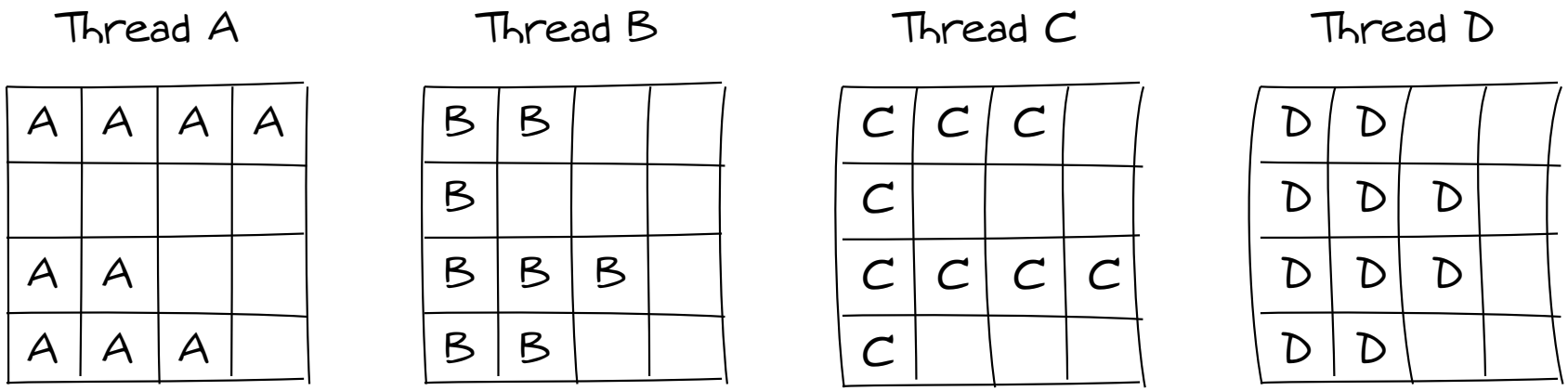


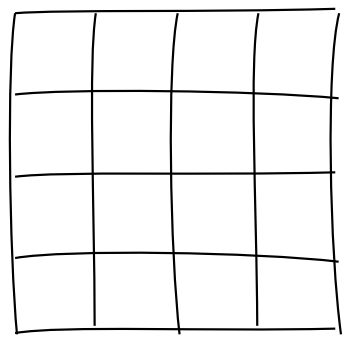


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

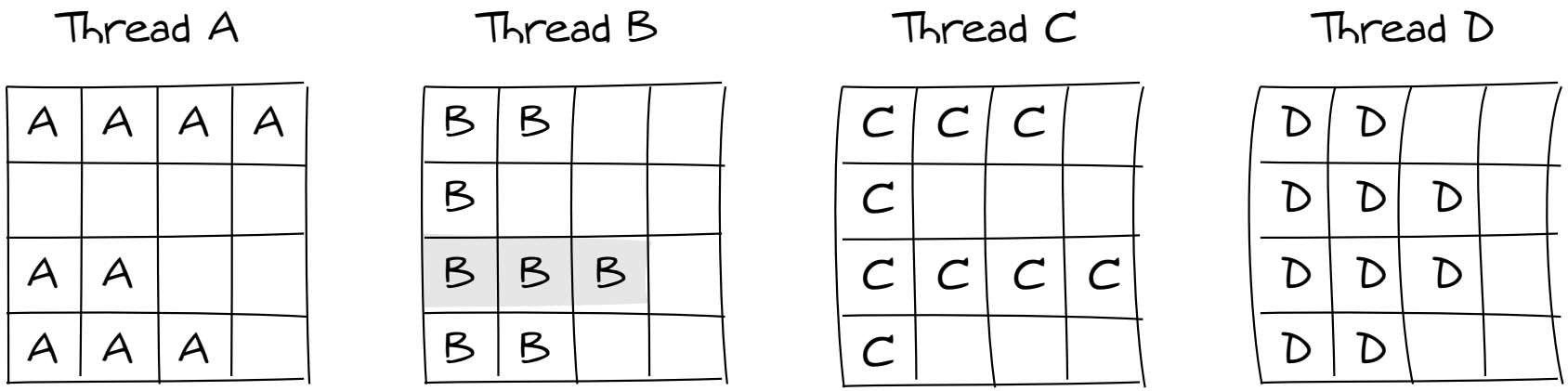


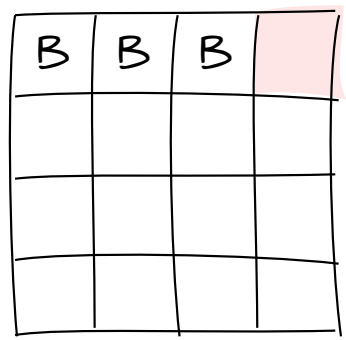


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

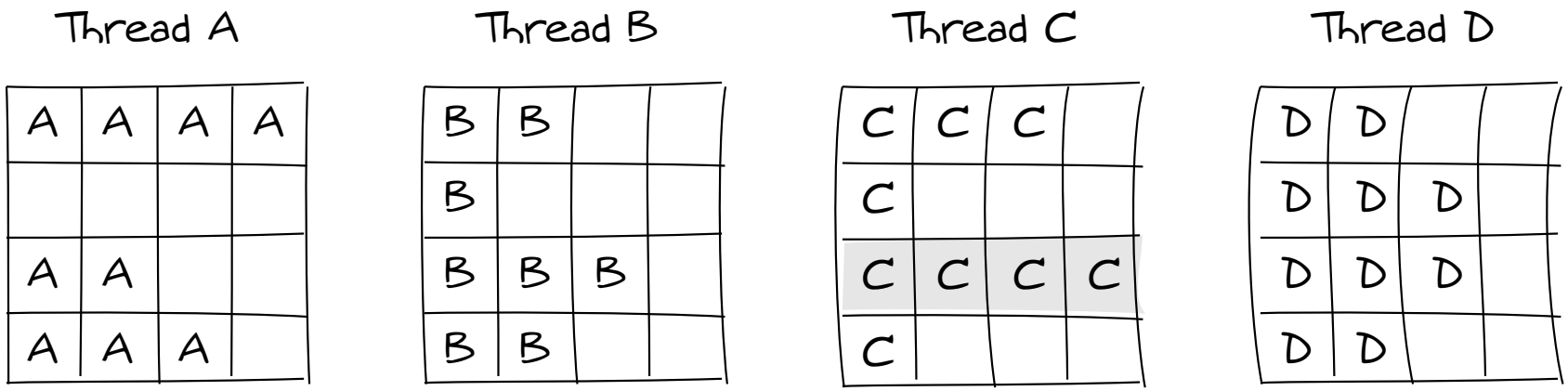


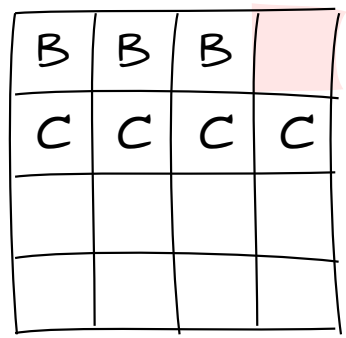


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

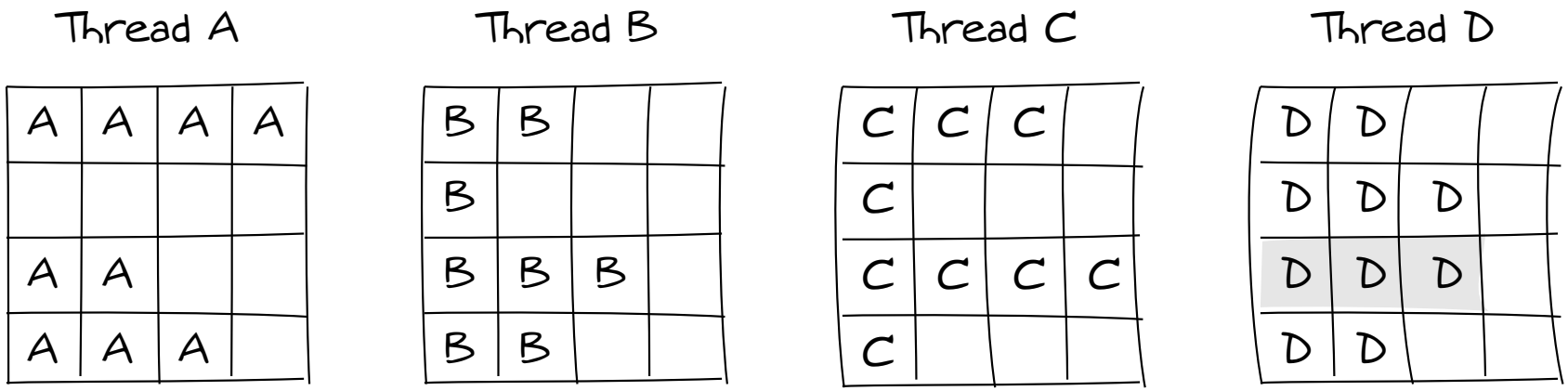


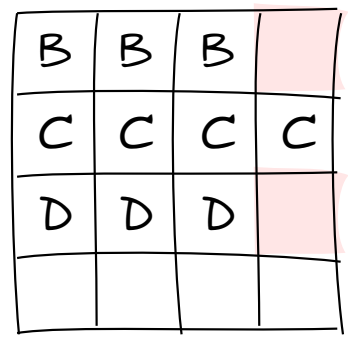


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

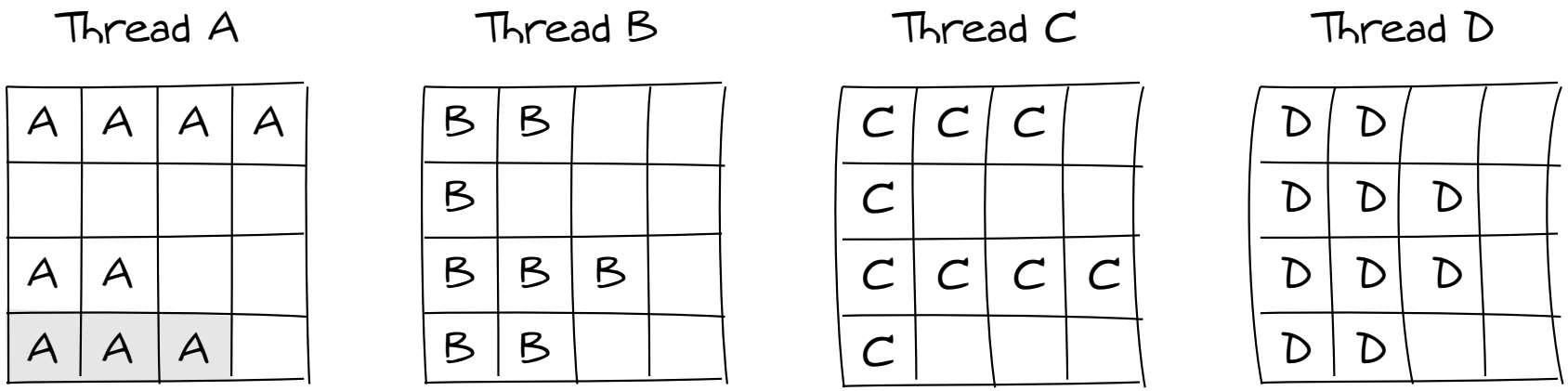


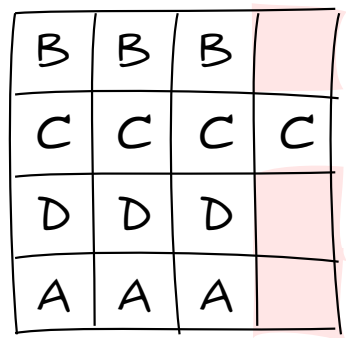


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

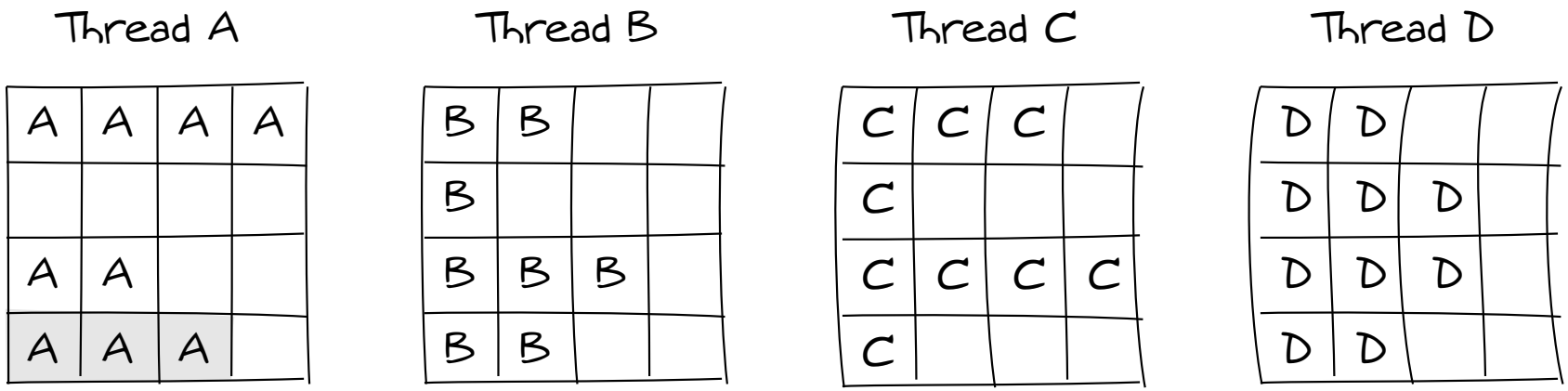


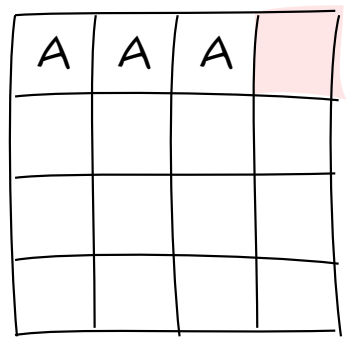


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

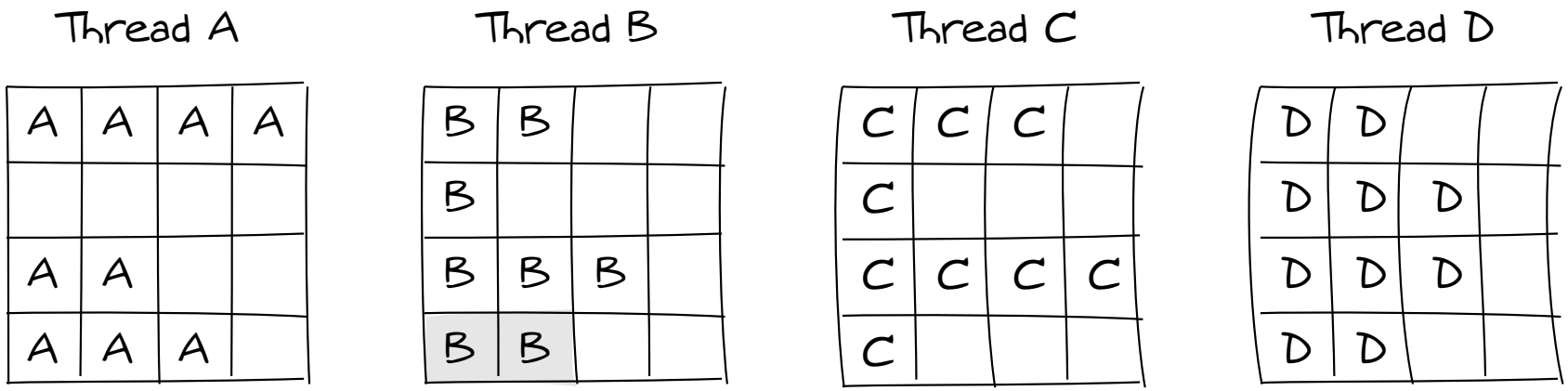


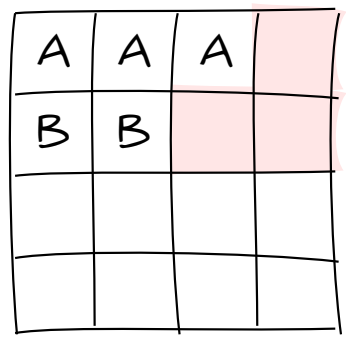


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

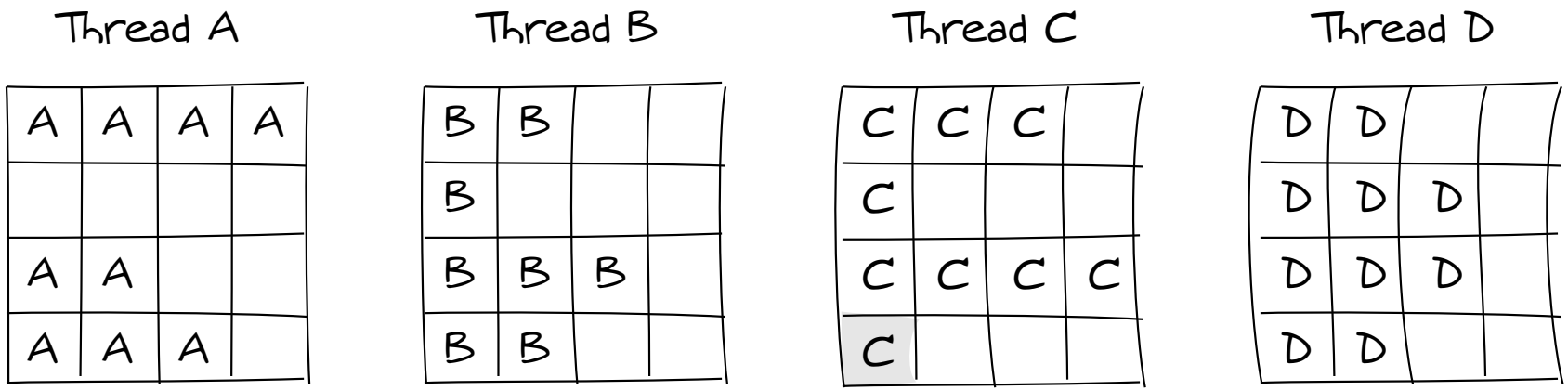


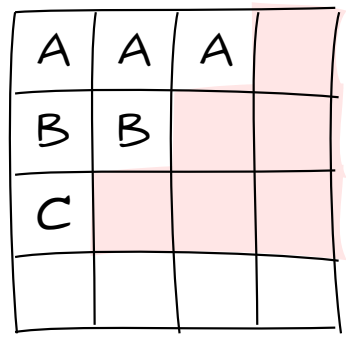


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

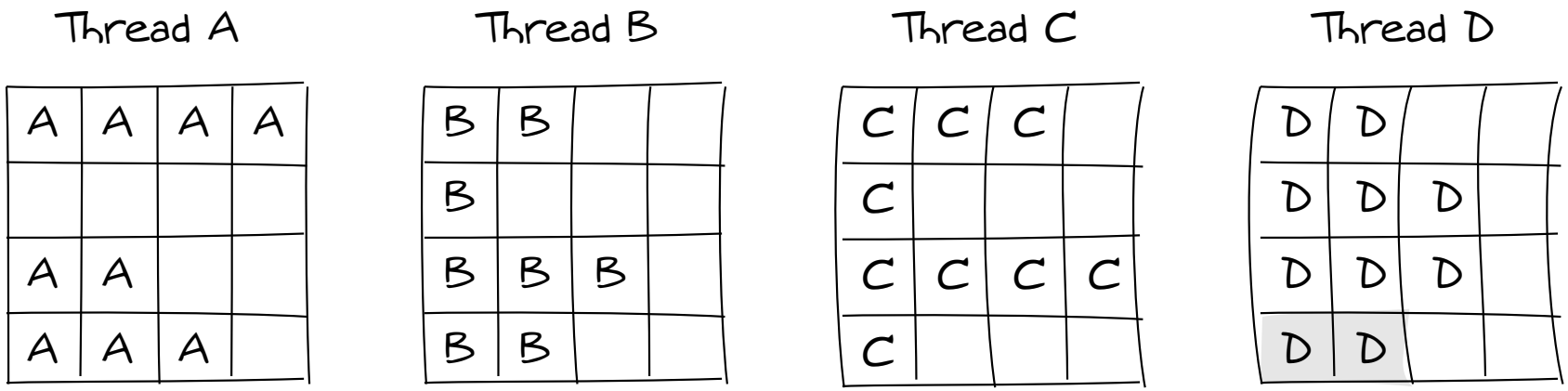


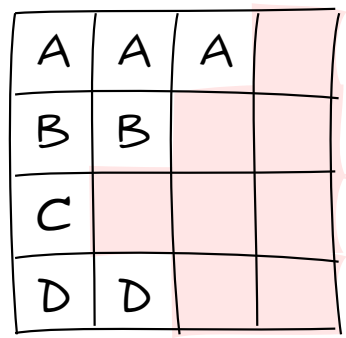


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*

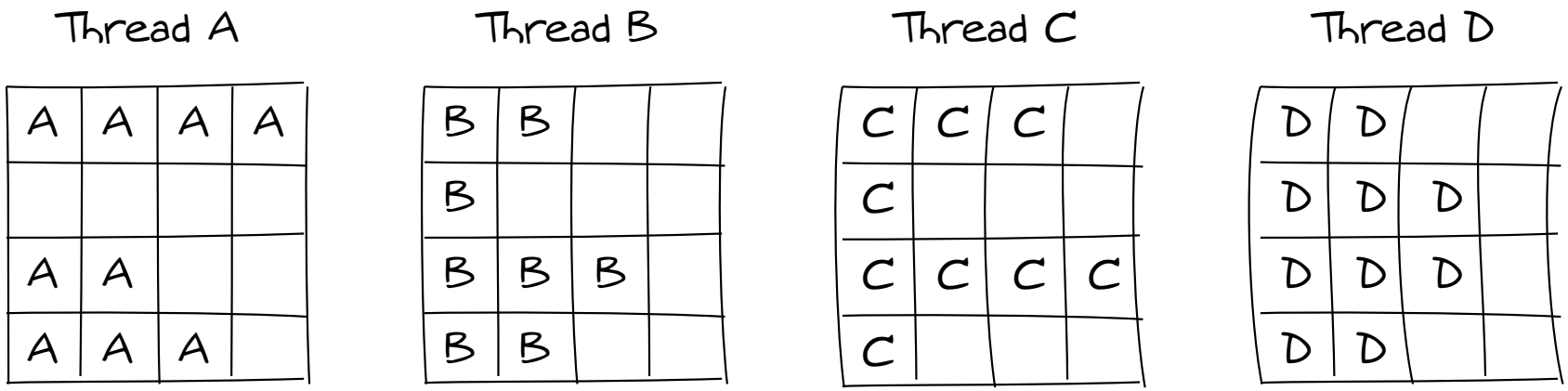


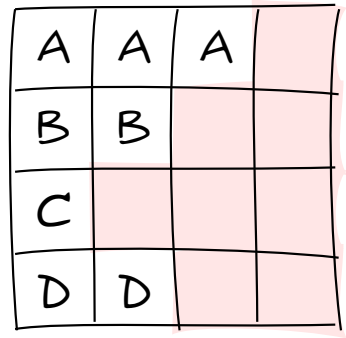


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*





Média de 62,5%
de uso da CPU

Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Fine-grained Multithreading*
  ✓ Manutenção da taxa de execução, apesar das paralisações de algumas *threads* em execução

Paralelismo de *thread*

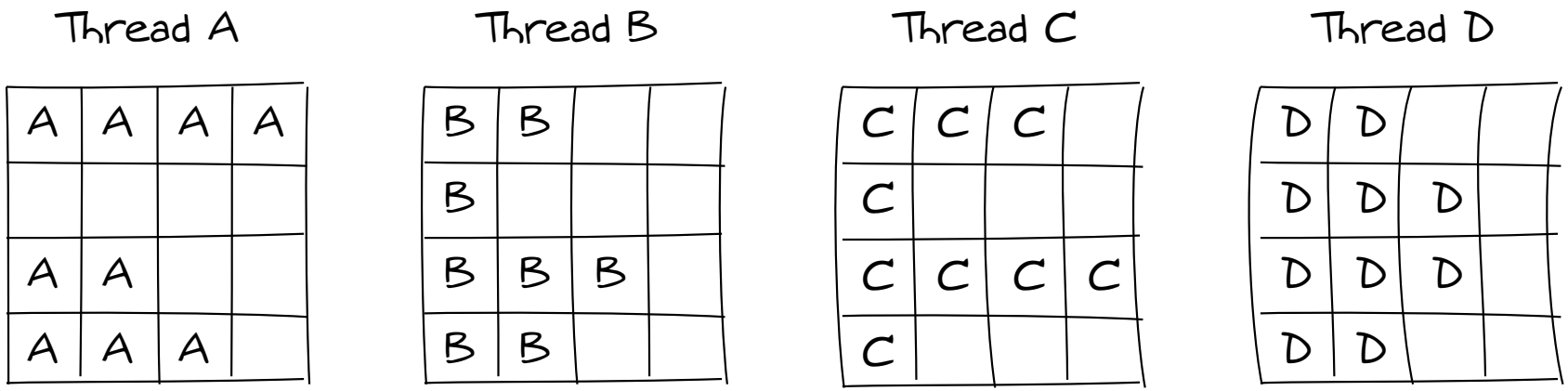
- *Fine-grained Multithreading*
  ✓ Manutenção da taxa de execução, apesar das paralisações de algumas *threads* em execução
  ✓ Maximização do aproveitamento das unidades de processamento, evitando a ociosidade

Paralelismo de *thread*

- *Fine-grained Multithreading*
  ✓ Manutenção da taxa de execução, apesar das paralisações de algumas *threads* em execução
  ✓ Maximização do aproveitamento das unidades de processamento, evitando a ociosidade
  ✗ Cada *thread* é executada de forma mais lenta, devido a intercalação com outras *threads*

Paralelismo de *thread*

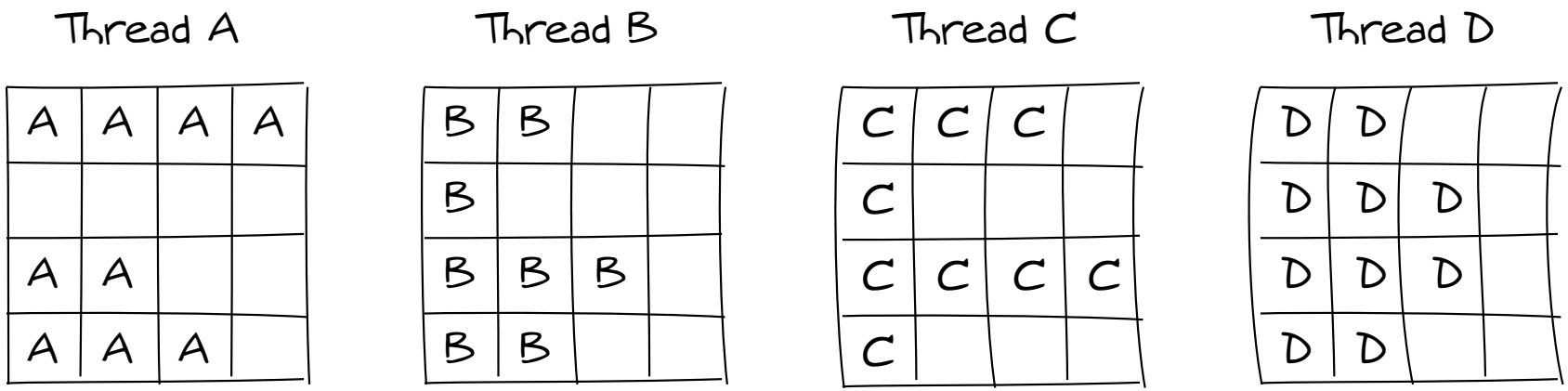
- *Coarse-grained Multithreading*
  - Nesta abordagem o escalonamento das *threads* do sistema é feito com baixa granularidade
  - Cada *thread* é executada até que ocorra sua paralisação devido a eventos internos ou externos
  - Quando uma paralisação ocorre, outra *thread* é alocada para execução no processador

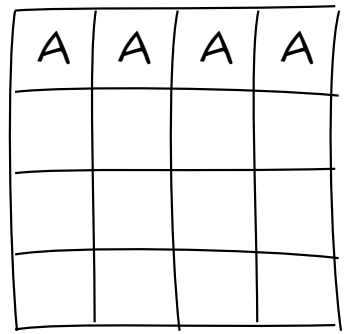


Execução das threads em um superescalar de 4 instruções

Paralelismo de *thread*

- *Coarse-grained Multithreading*

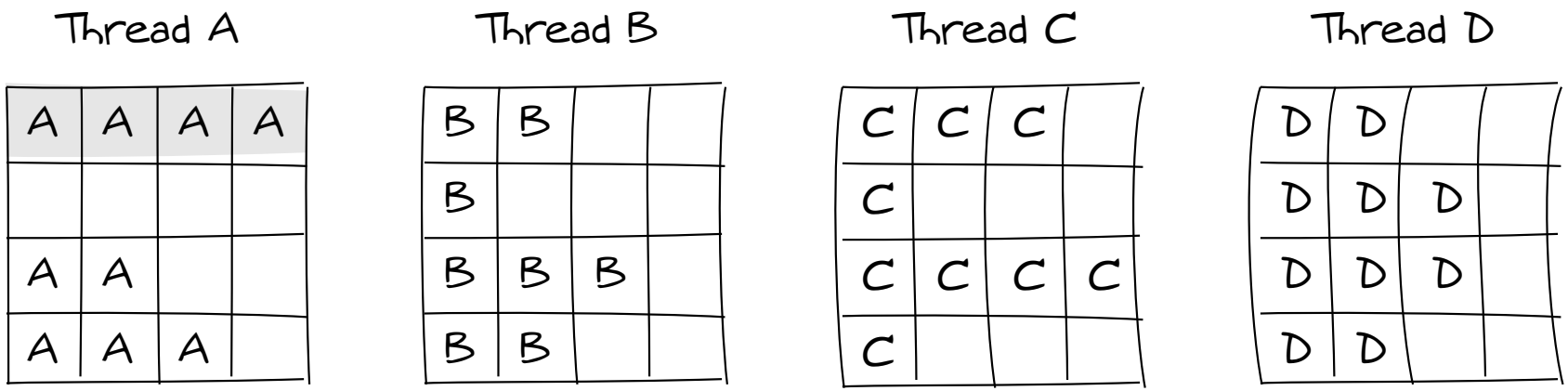


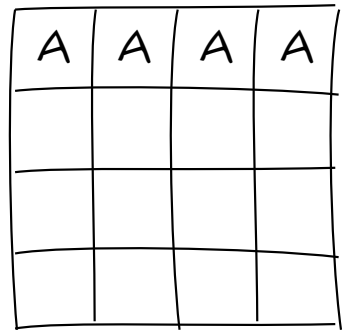


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*

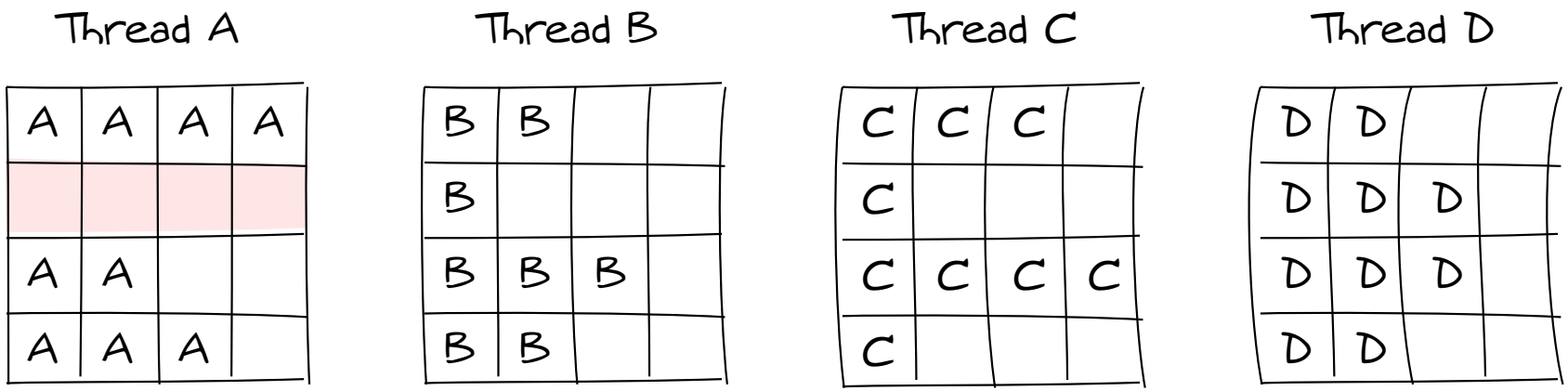


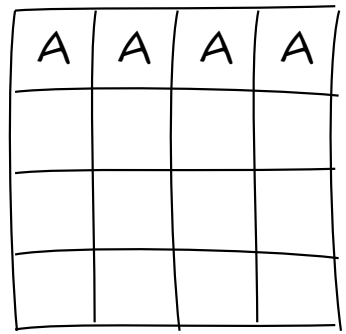


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*

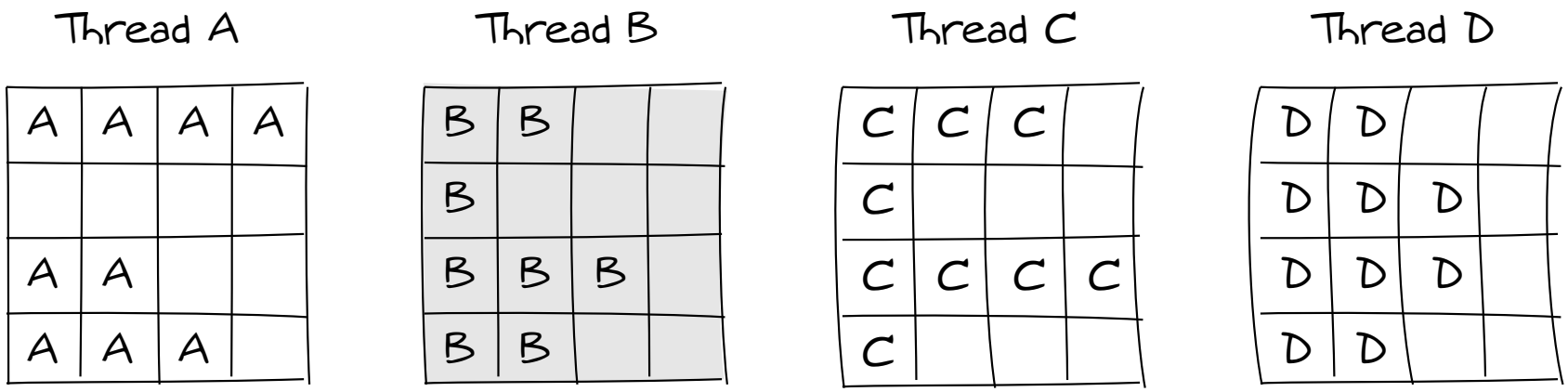


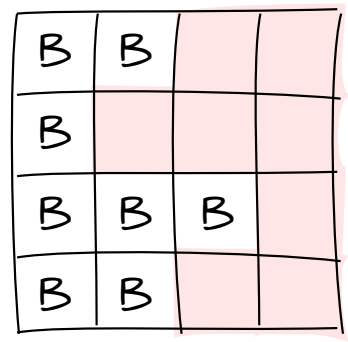


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*

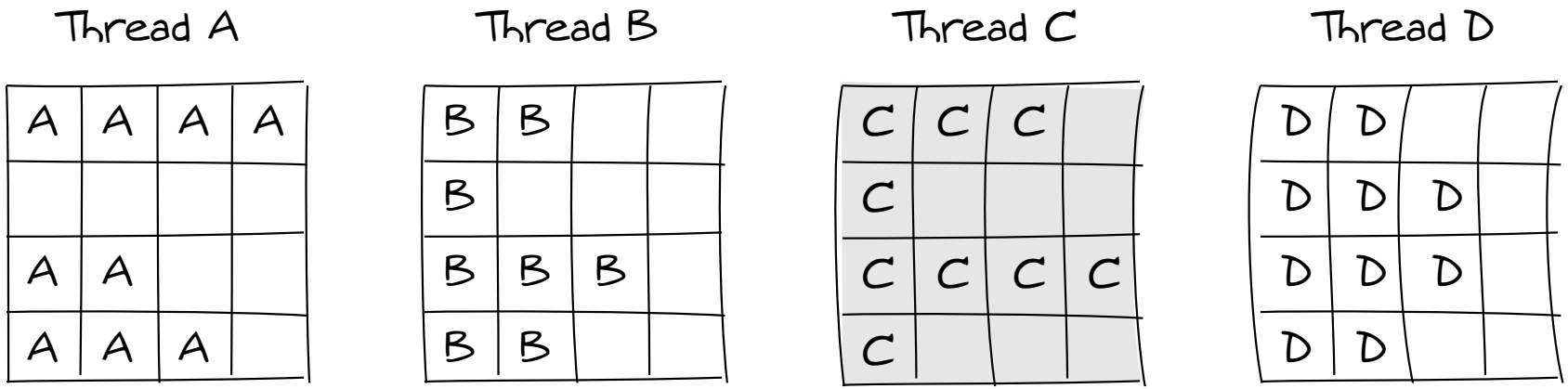


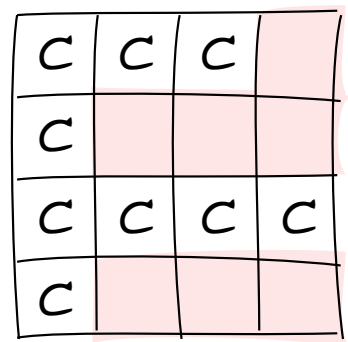


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*

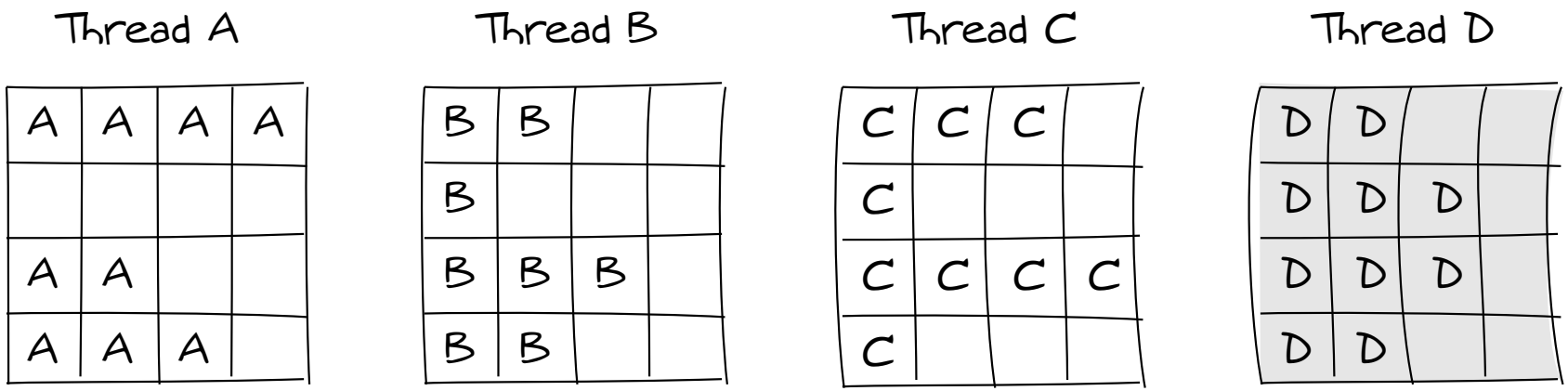


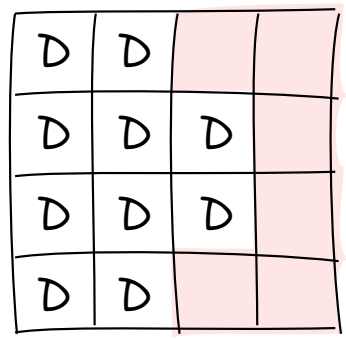


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*

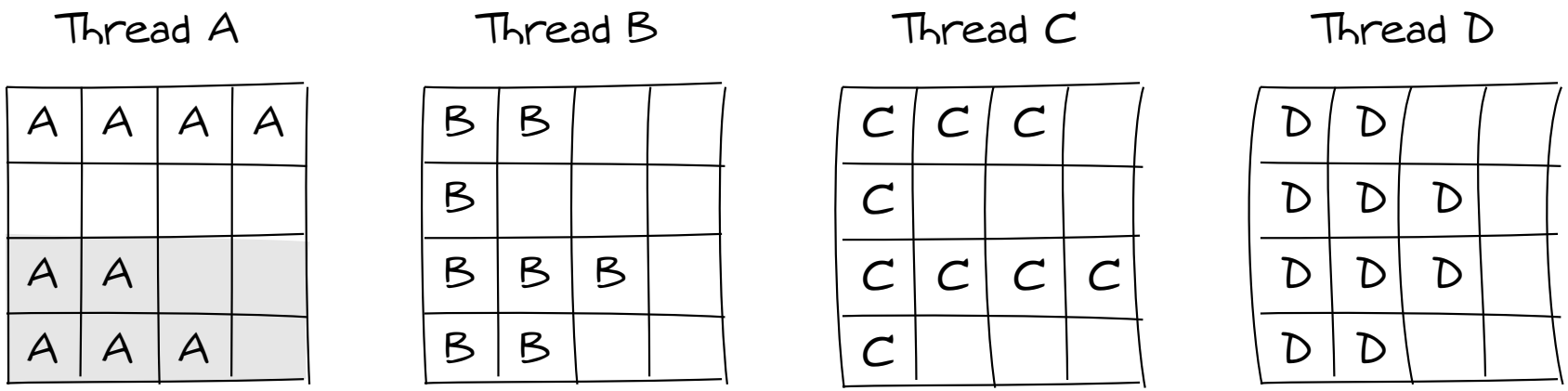


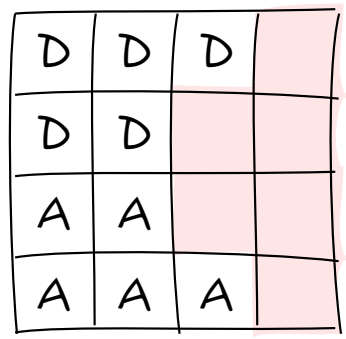


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*

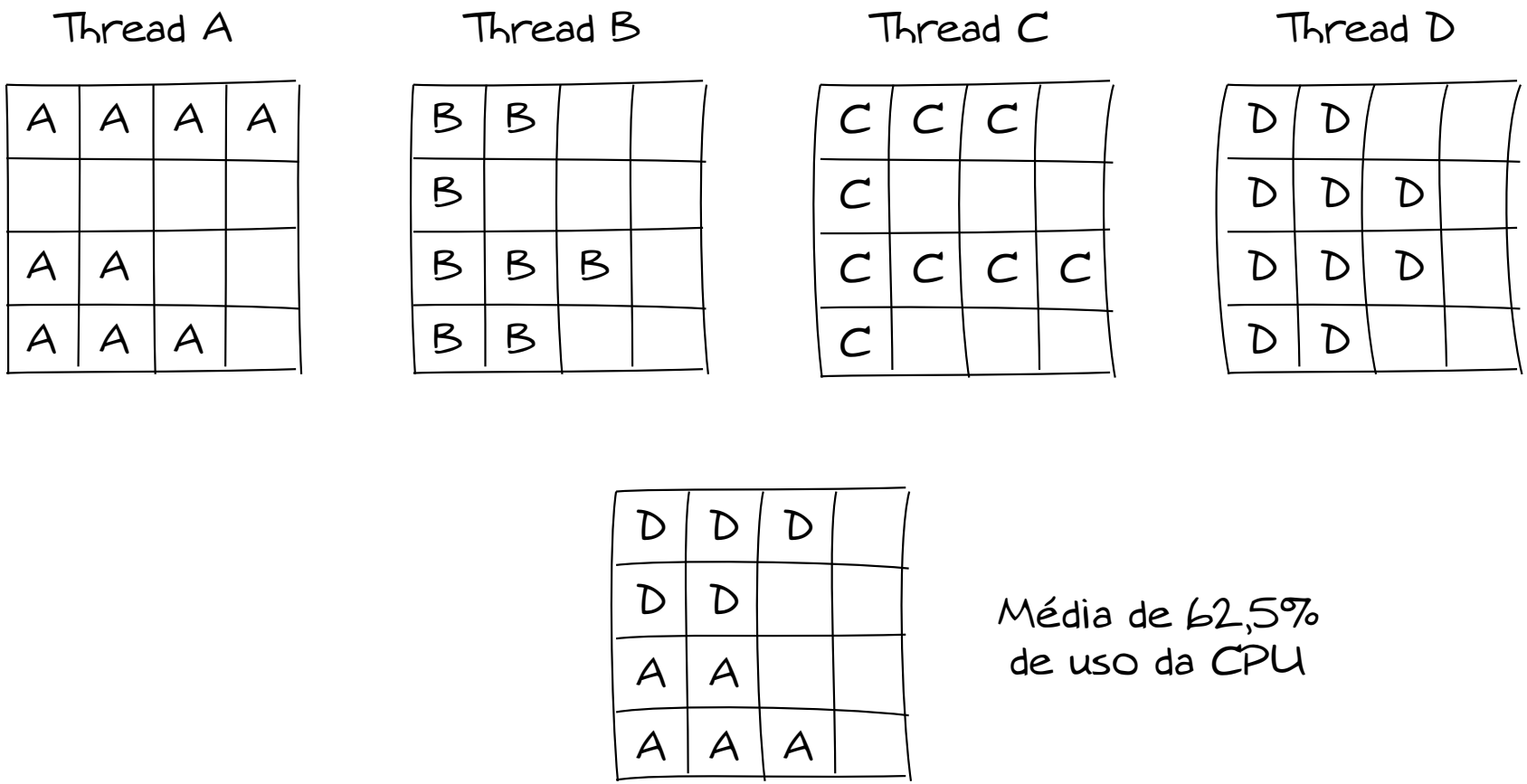




Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*



Média de 62,5% de uso da CPU

Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Coarse-grained Multithreading*
  ✓ Cada *thread* é executada mais rapidamente, considerando o tempo individual de execução

Paralelismo de *thread*

- *Coarse-grained Multithreading*
  ✓ Cada *thread* é executada mais rapidamente, considerando o tempo individual de execução
  ✓ Ocorre a redução do escalonamento das *threads*

Paralelismo de *thread*

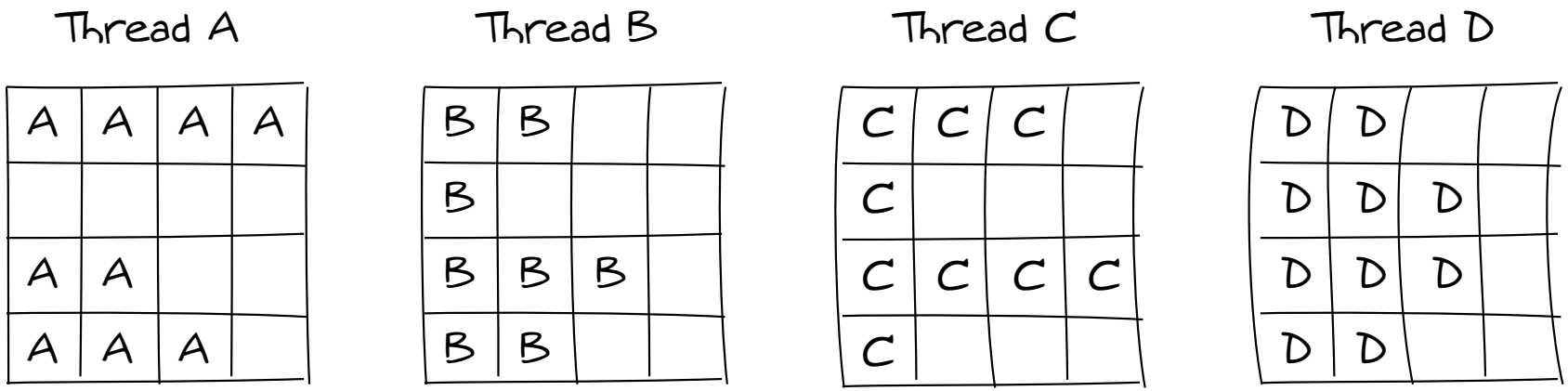
- *Coarse-grained Multithreading*
  ✓ Cada *thread* é executada mais rapidamente, considerando o tempo individual de execução
  ✓ Ocorre a redução do escalonamento das *threads*
  ✗ Caso ocorra uma paralisação, outra *thread* precisa ser executada, esvaziando o *pipeline*

Paralelismo de *thread*

- *Coarse-grained Multithreading*
  ✓ Cada *thread* é executada mais rapidamente, considerando o tempo individual de execução
  ✓ Ocorre a redução do escalonamento das *threads*
  ✗ Caso ocorra uma paralisação, outra *thread* precisa ser executada, esvaziando o *pipeline*
  ✗ Como apenas uma *thread* está executando, pode reduzir o desempenho por faltas na cache

Paralelismo de *thread*

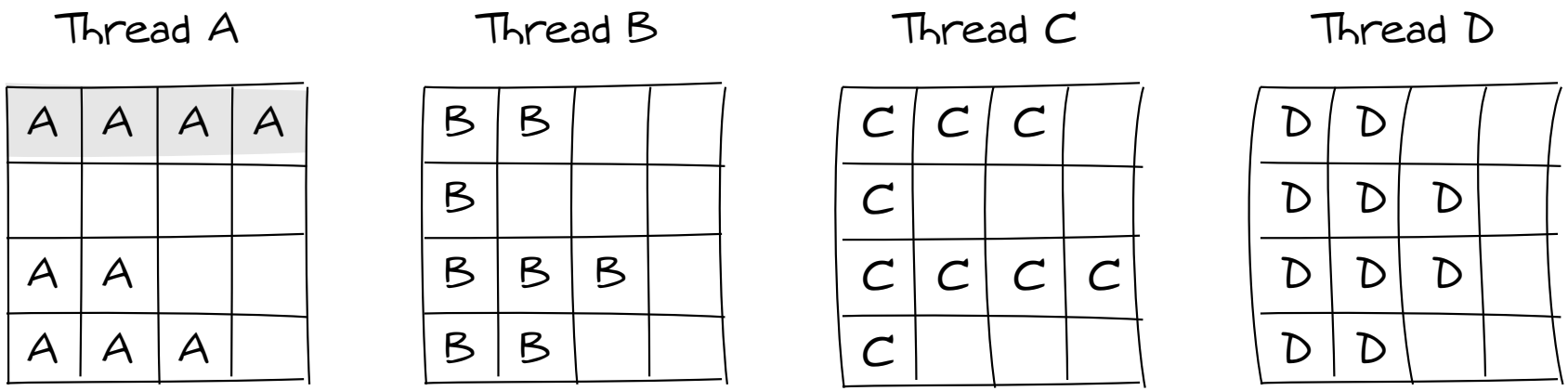
- *Simultaneous Multithreading* (SMT)
  - É uma técnica de escalonamento que combina o paralelismo de *thread* com o paralelismo de instrução
  - As instruções das *threads* são buscadas e executadas independentemente, utilizando recursos dedicados
  - Várias *threads* podem estar executando ao mesmo tempo, sem escalonamento por granularidade

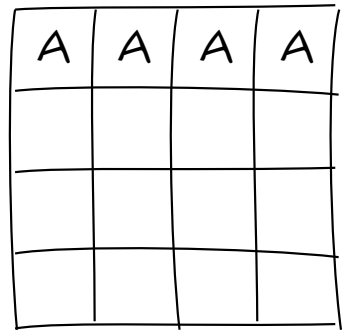


Execução das threads em um superescalar de 4 instruções

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

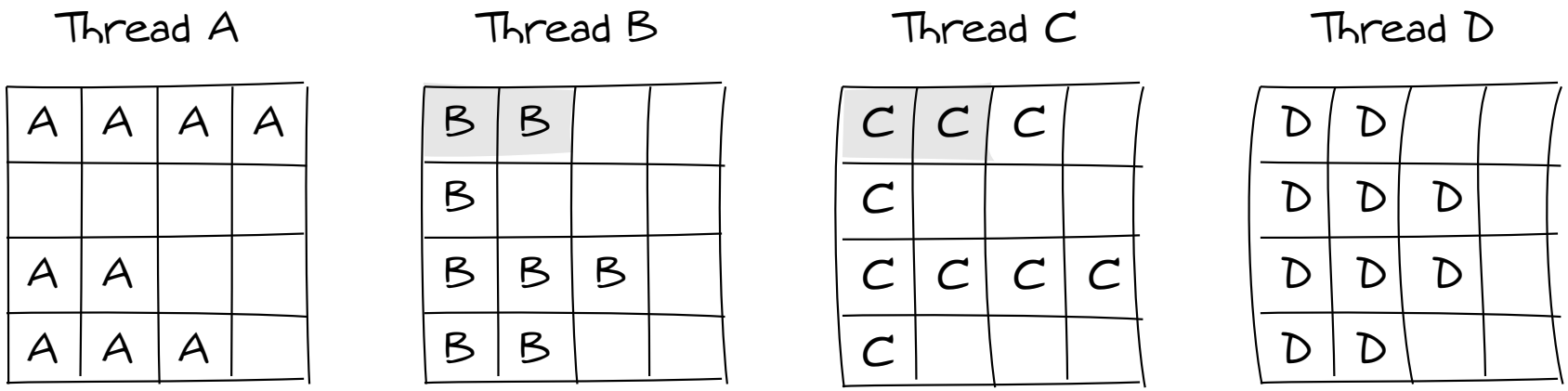


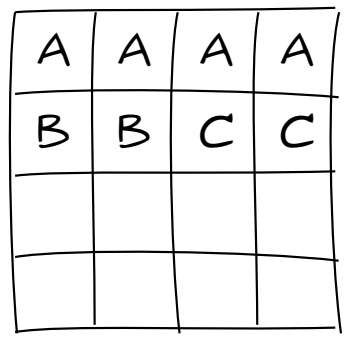


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

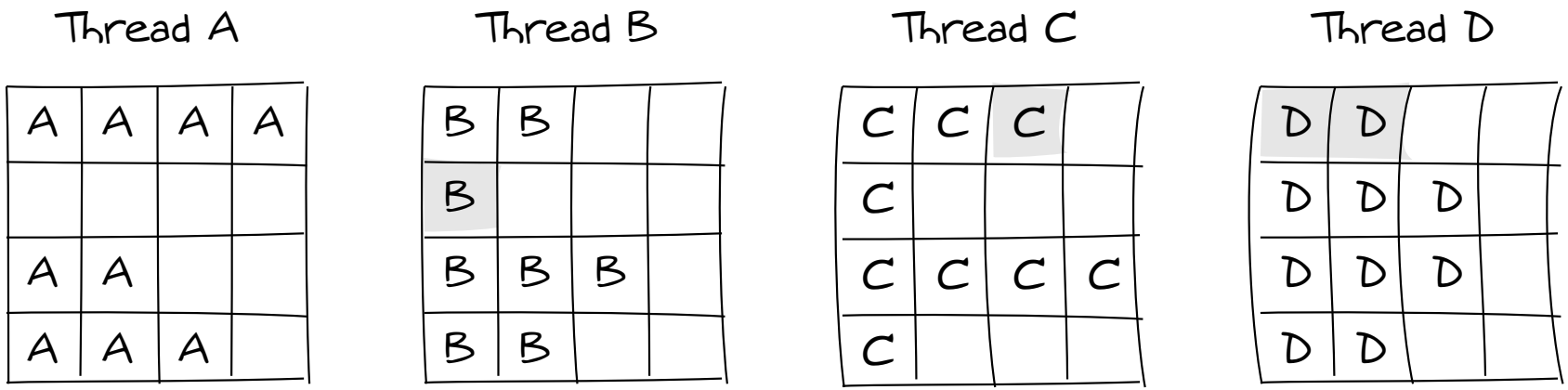


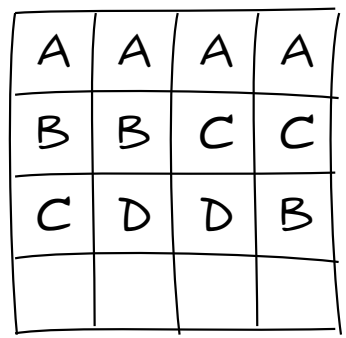


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

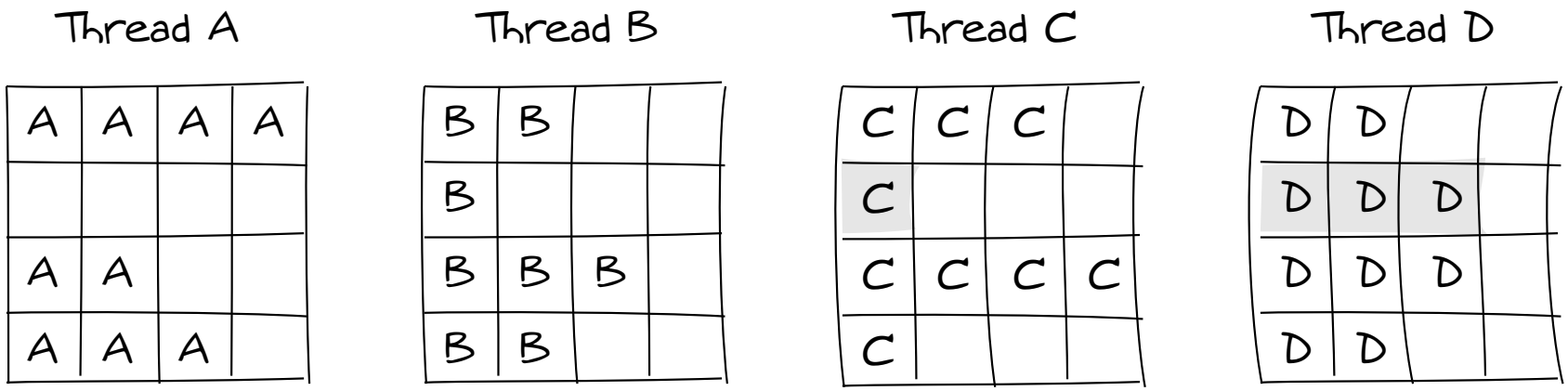


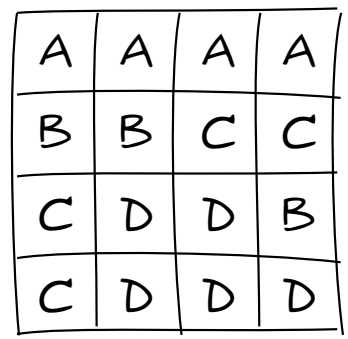


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

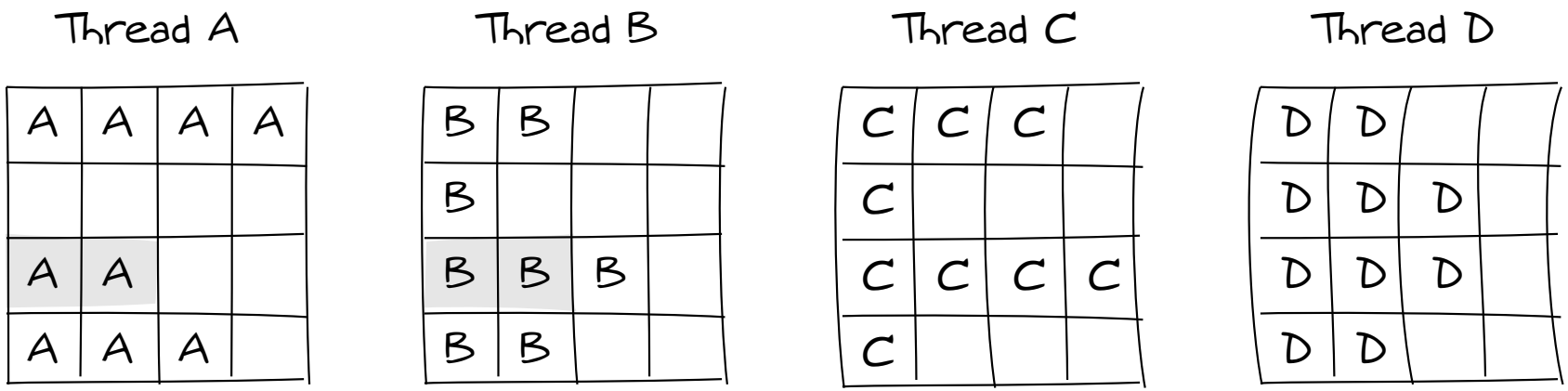


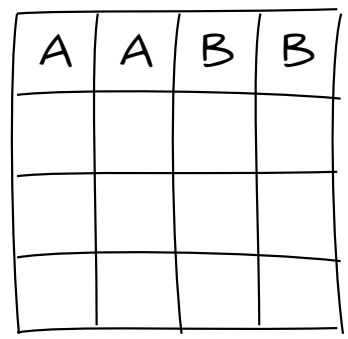


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

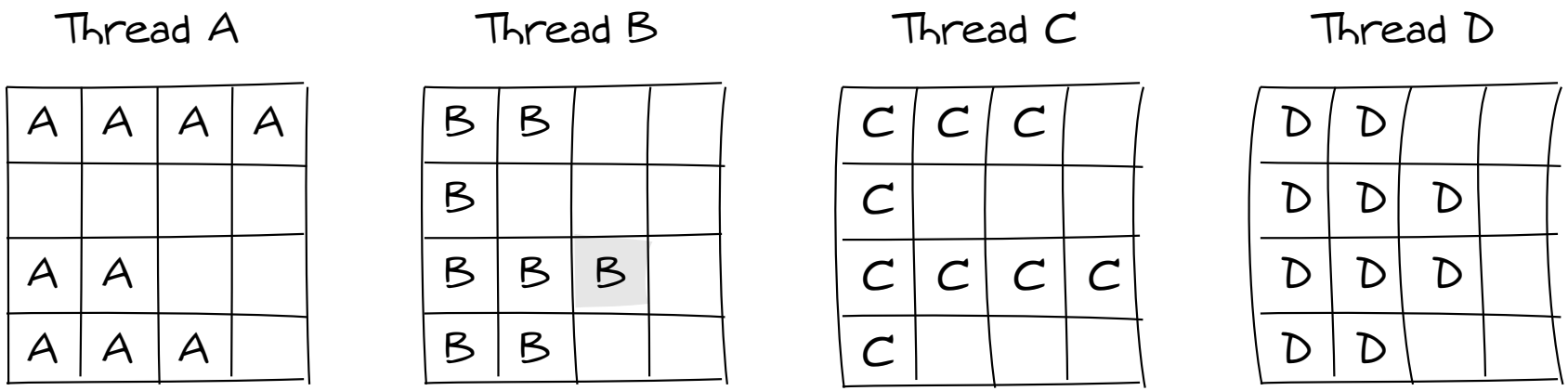


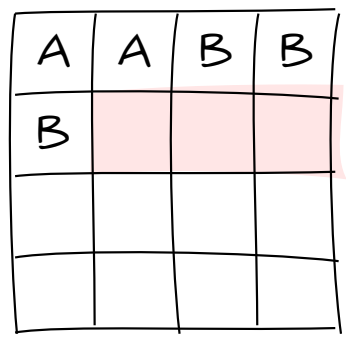


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

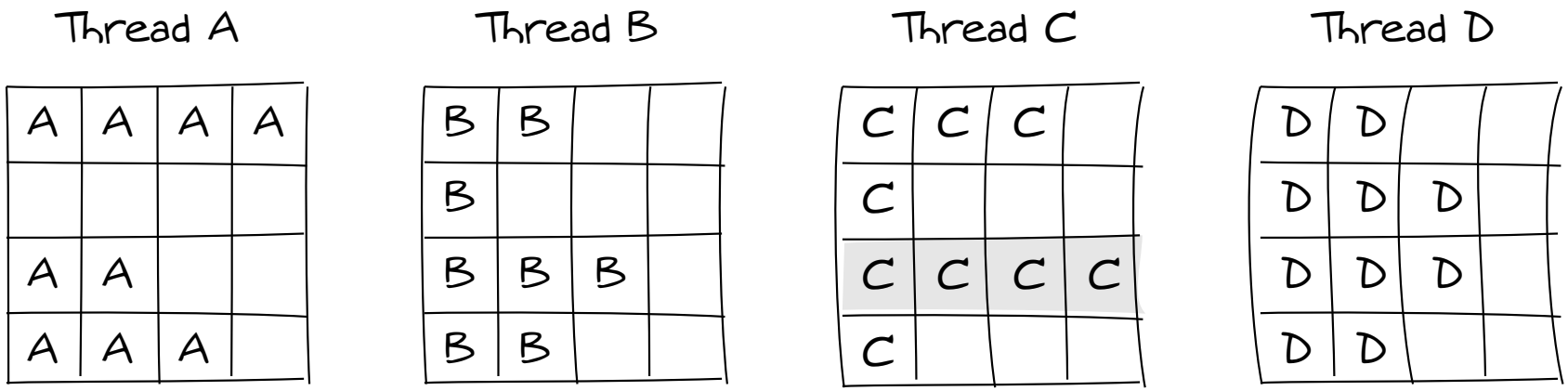


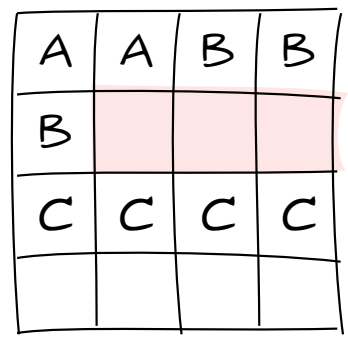


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

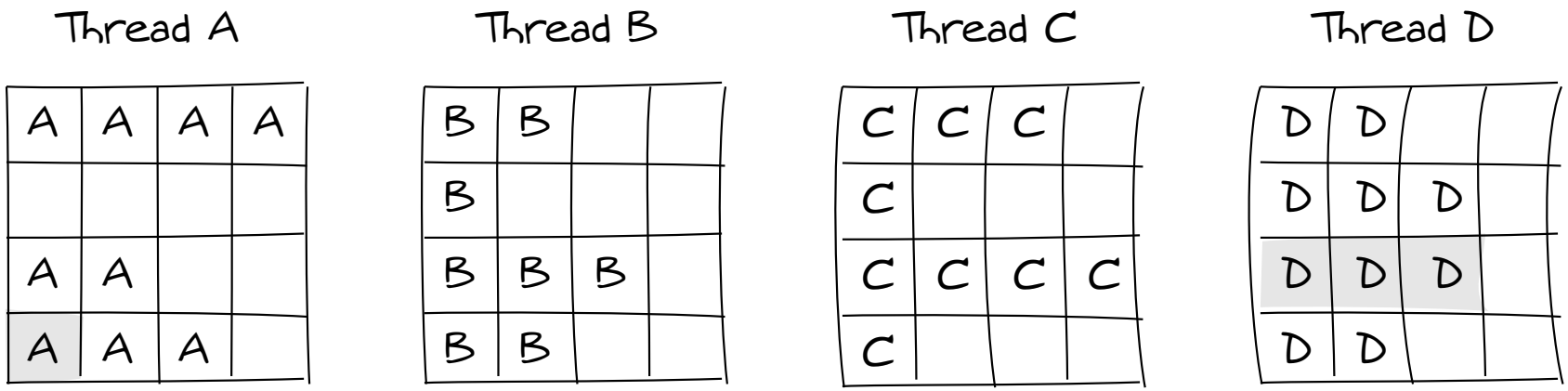


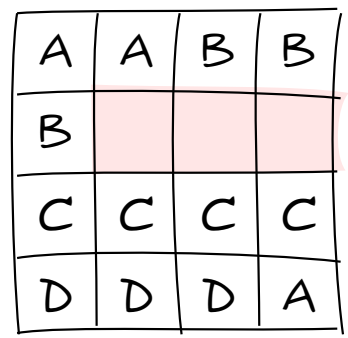


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

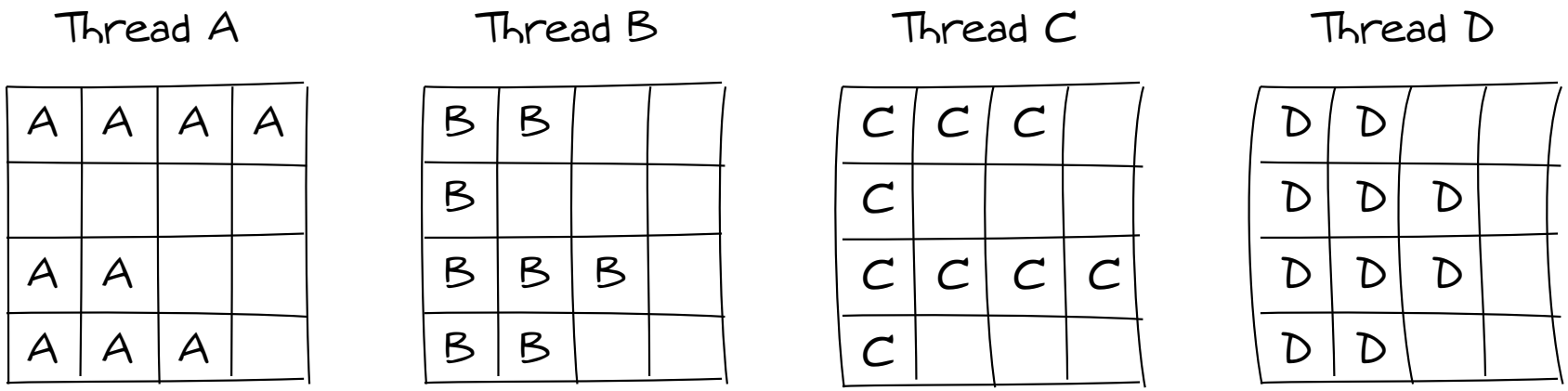


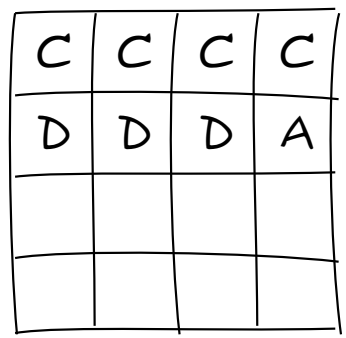


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

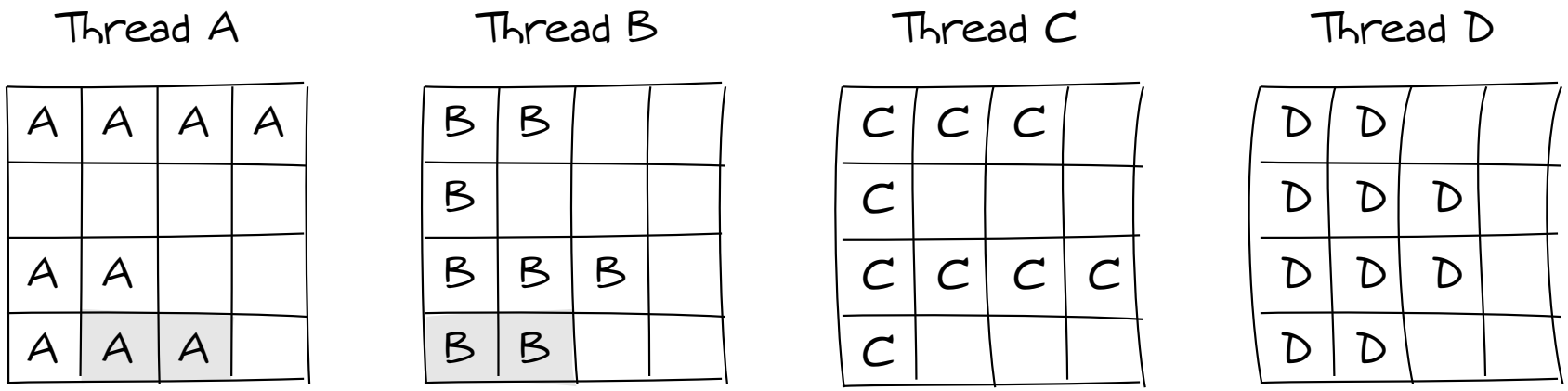


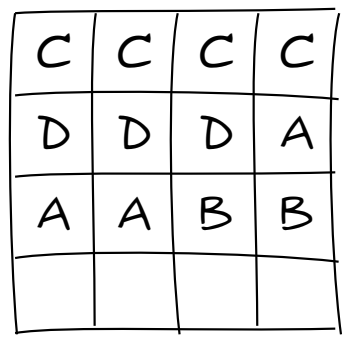


Intercalação das threads em
um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)

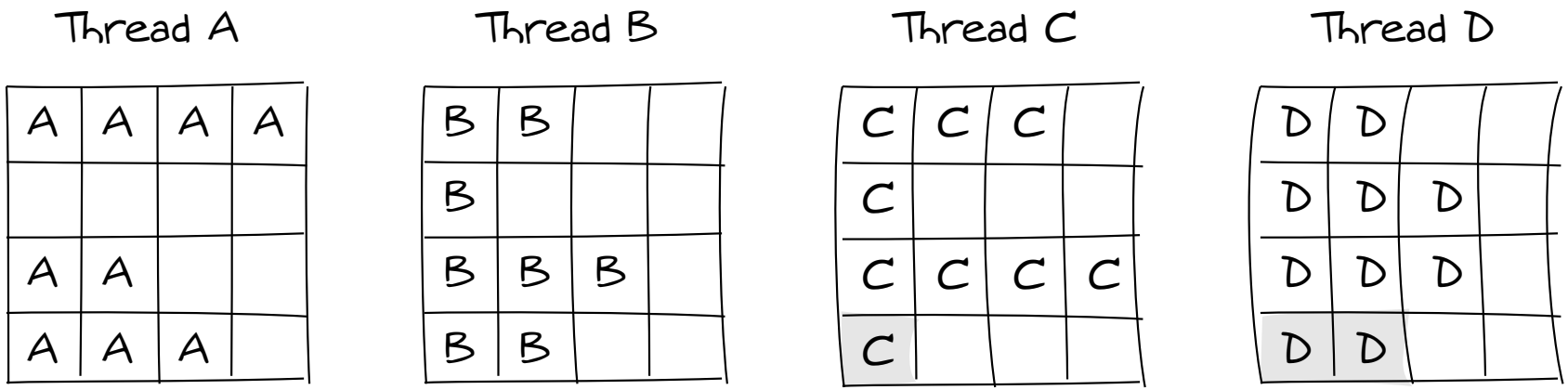




Intercalação das threads em um processador superescalar

Paralelismo de *thread*

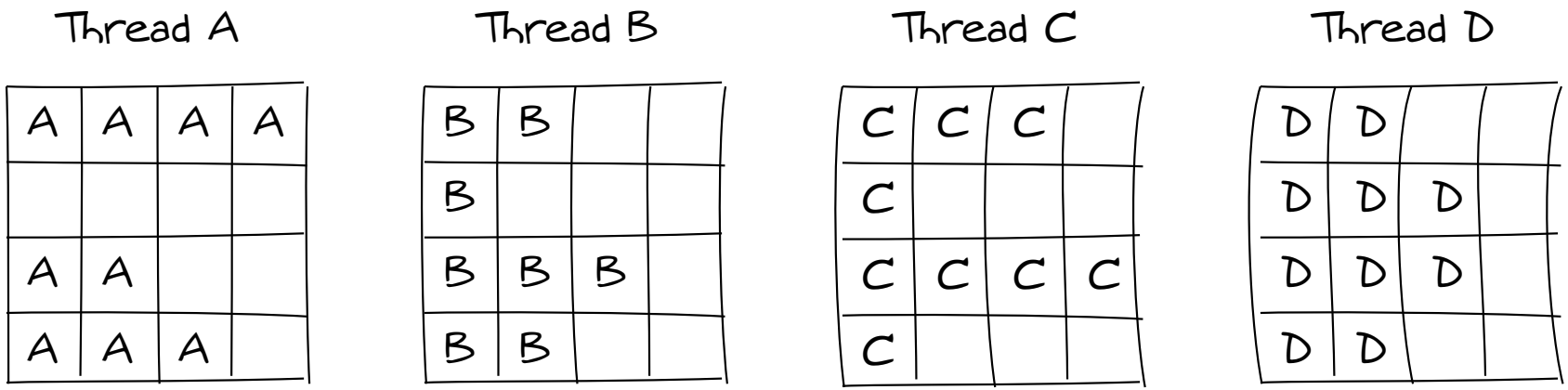
- *Simultaneous Multithreading* (SMT)

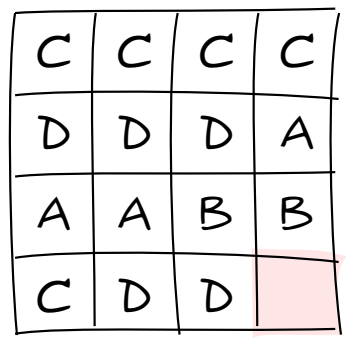


Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)





Média de 90% de uso da CPU

Intercalação das threads em um processador superescalar

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)
  ✓ Aumento da taxa de execução de instruções e de utilização dos recursos da plataforma

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)
  ✓ Aumento da taxa de execução de instruções e de utilização dos recursos da plataforma
  ✓ Paralelismo de *thread* e de instrução combinados

Paralelismo de *thread*

- *Simultaneous Multithreading* (SMT)
  ✓ Aumento da taxa de execução de instruções e de utilização dos recursos da plataforma
  ✓ Paralelismo de *thread* e de instrução combinados
  ✗ Maior complexidade no projeto de processador, com unidades dedicadas para cada *thread*