Sistemas Operacionais

Gestão de memória - Alocação de memória

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Abril de 2019

Conteúdo

1. Alocação de memória
2. Fragmentação
   - Estratégias de alocação
   - Desfragmentação
   - Fragmentação interna
3. O alocador Buddy
4. O alocador Slab
5. Alocação no espaço de usuário

Alocação de memória

Alocar memória

Reservar memória para uso do SO ou dos processos

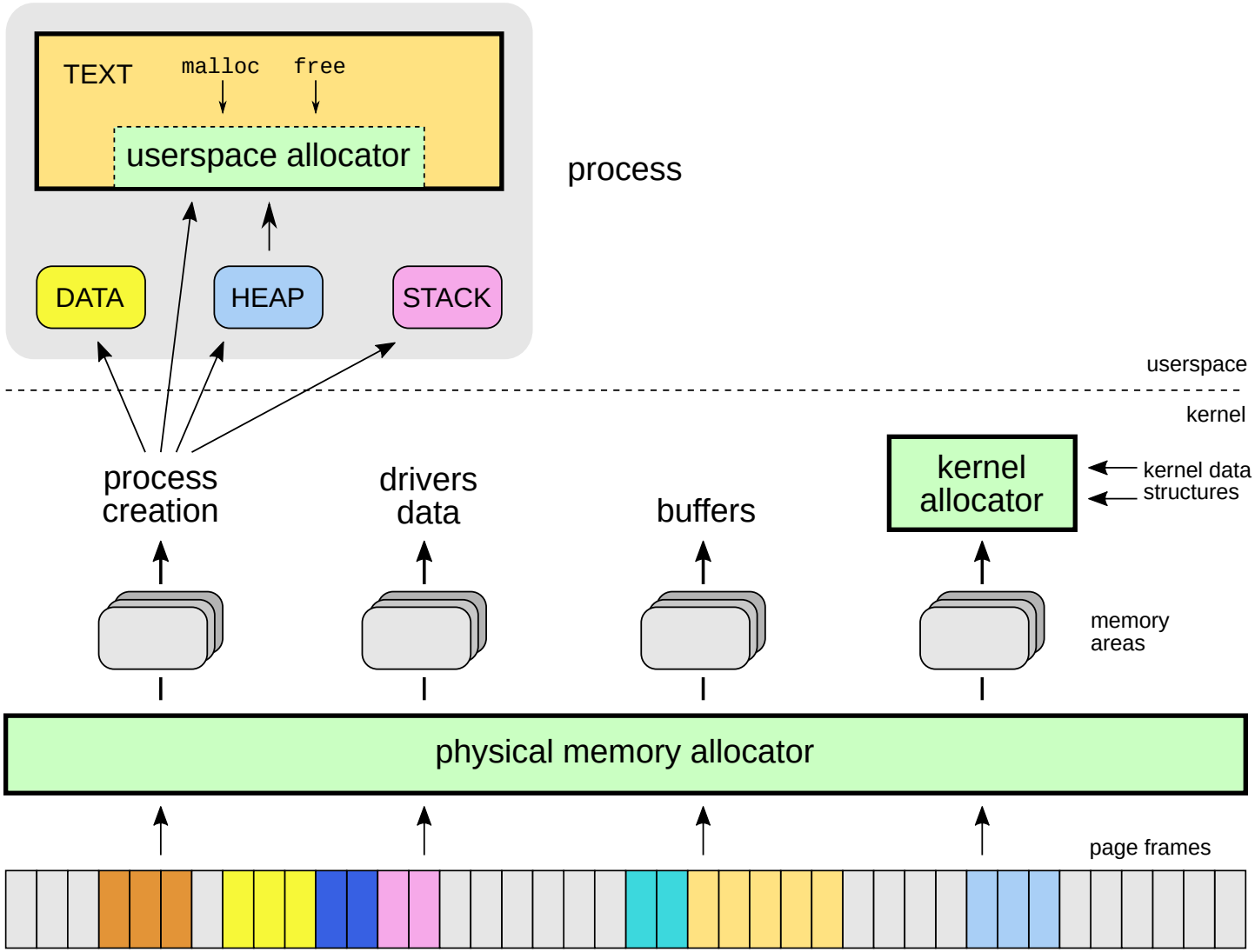
O **alocador** de memória:

- atende solicitações do núcleo ou de processos
- aloca e libera áreas de memória
- gerencia que áreas estão livre ou ocupadas
- rapidez e pouco desperdício de memória

Níveis de alocação

- **Memória física**:
  - Organiza a memória RAM
  - Aloca áreas para carregar processos e para o núcleo
- **No espaço de núcleo**:
  - Obtém áreas do alocador físico
  - Aloca/libera estruturas de dados do núcleo
  - Sockets, semáforos, descritores de processos, ...
- **No espaço de usuário**:
  - Implementado em bibliotecas fora do núcleo
  - Gerencia a área *Heap* do processo
  - Atende requisições dos processos (`malloc`, `free`)

Alocação de memória



Alocação básica

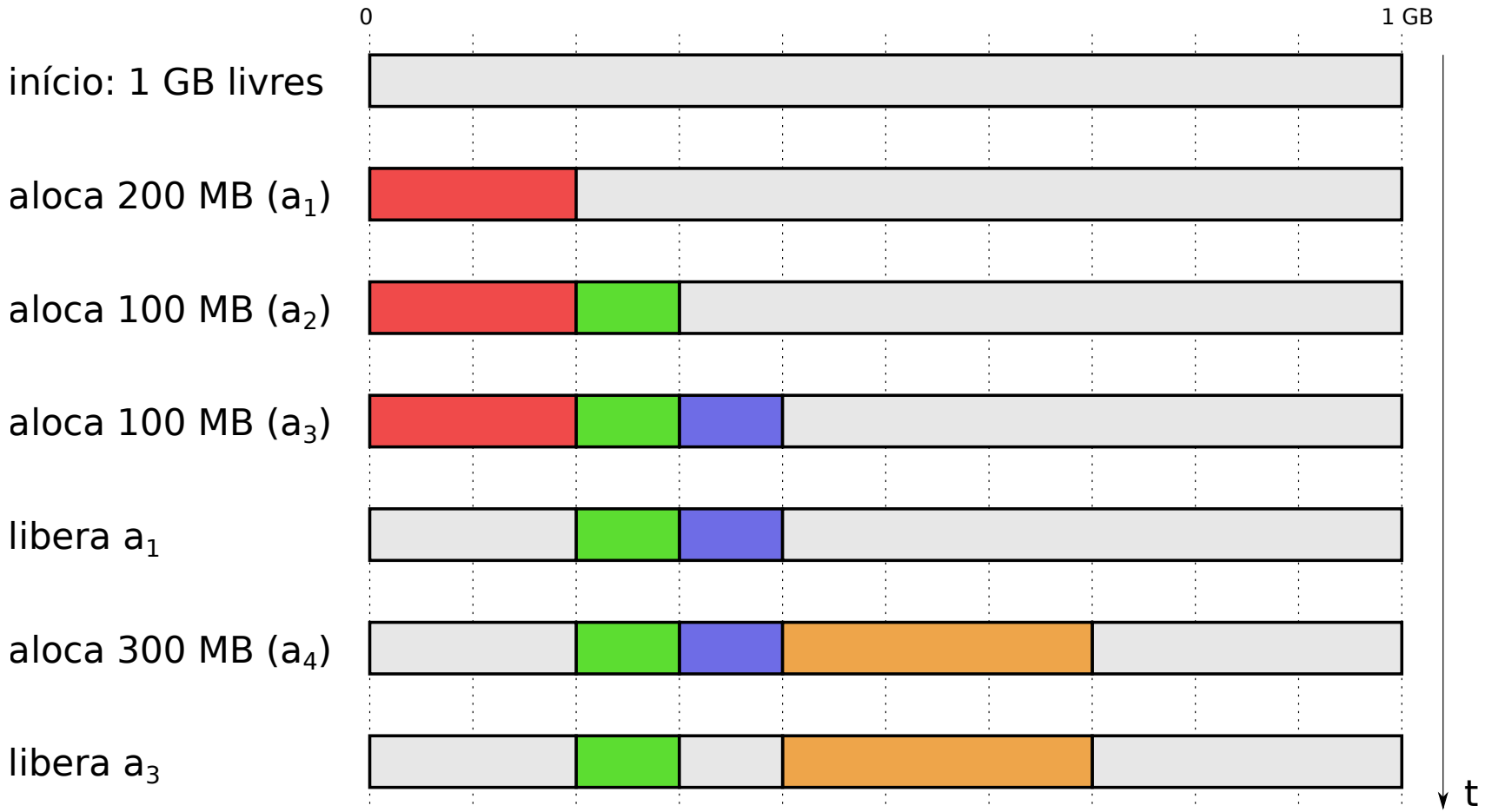
Problema:

- manter uma ou mais áreas de memória
- atender pedidos de alocação e liberação de blocos
- otimizar o uso da memória
- evitar/minimizar a fragmentação

Estratégia:

- manter listas de áreas livres e ocupadas

Alocação de memória



Fragmentação da memória

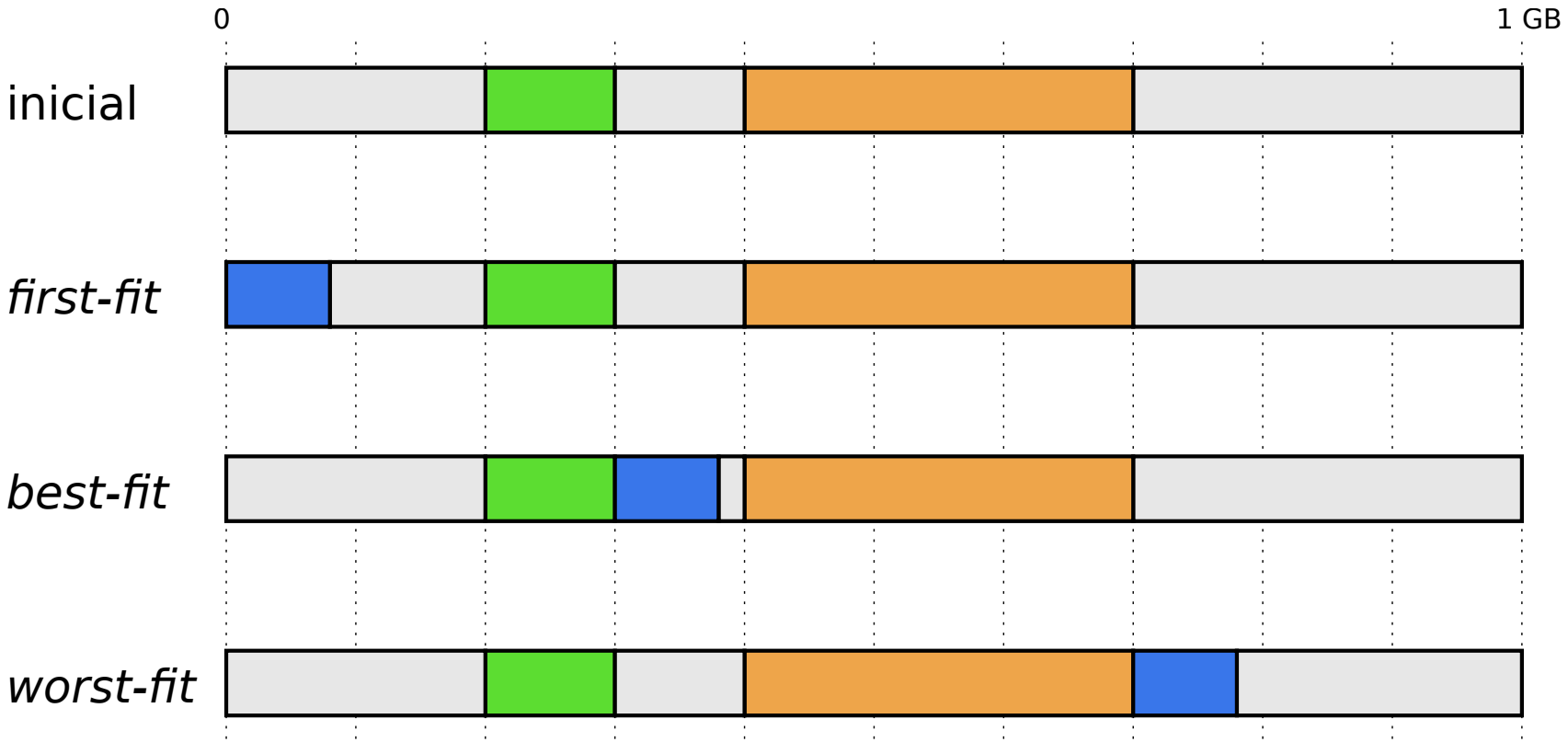
Fragmentação **externa**:

- “Buracos” entre áreas de memória alocadas
- Gerados pelas alocações e liberações de memória
- Reduz a capacidade de alocação
- Exige mais esforço de gerência

Estratégias de alocação de novos blocos

- ***Best-Fit***:
  - Escolher a **menor área** livre onde o bloco couber
  - Pode gerar fragmentos pequenos e inúteis
- ***Worst-Fit***:
  - Escolher sempre a **maior área** livre
  - Pode gerar escassez de áreas grandes
- ***First-Fit***:
  - Escolher a **primeira área** livre onde o bloco couber
  - Solução rápida
- ***Next-Fit***:
  - Variante da *first-fit*, para distribuir as alocações

Fragmentação externa



Qual a melhor estratégia?

Desfragmentação

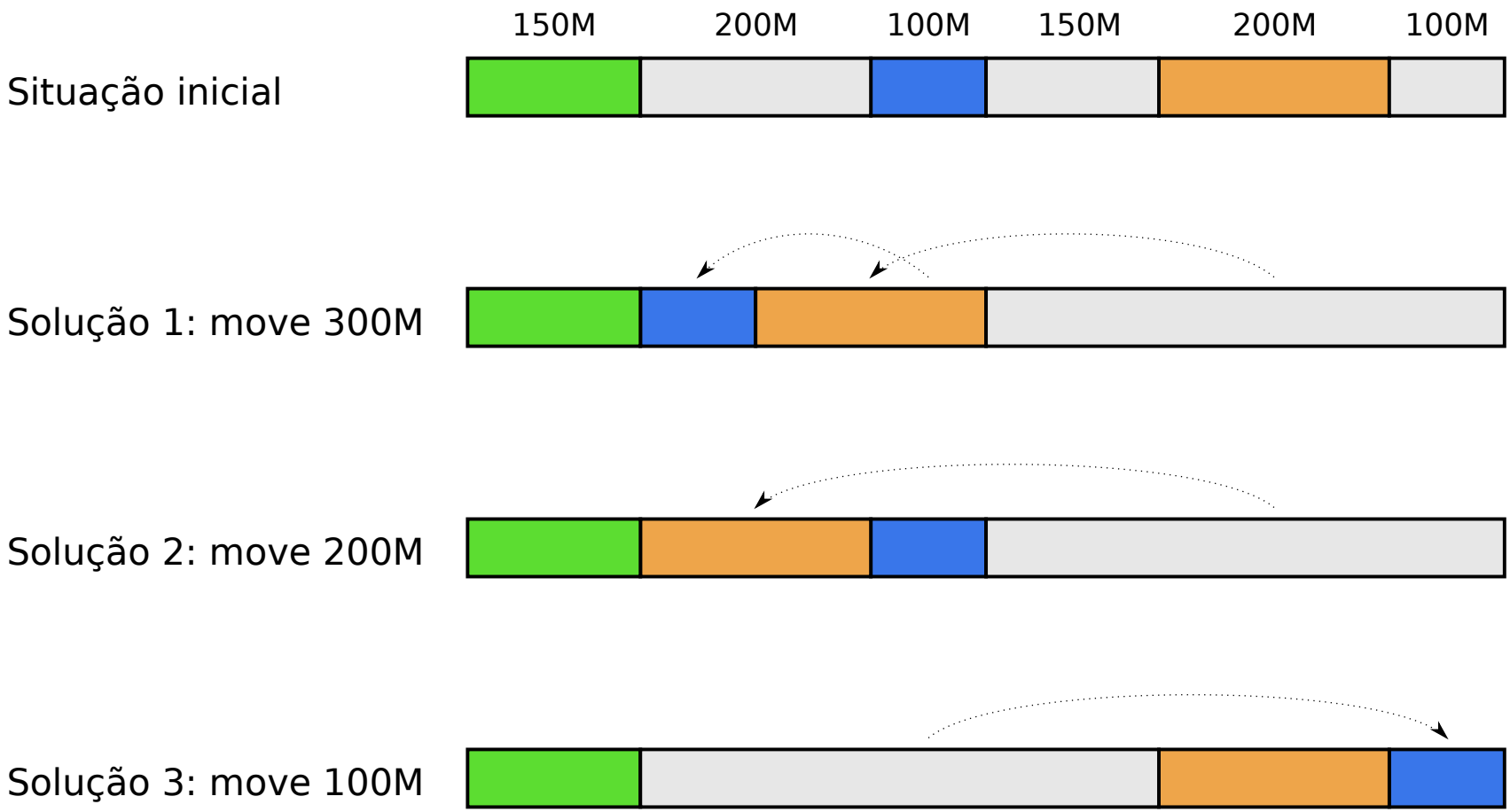
Ideia:

- Periodicamente, desfragmentar a memória
- Mover os blocos para concatenar fragmentos
- Liberar áreas maiores de memória

Problemas:

- Endereços de memória dos blocos serão alterados
- Só pode ser aplicado à memória física (MMU)
- Deve ser rápido para não atrapalhar o sistema
- Algoritmos de otimização combinatória

Desfragmentação



Fragmentação interna

Arredondamento das solicitações de blocos de memória:

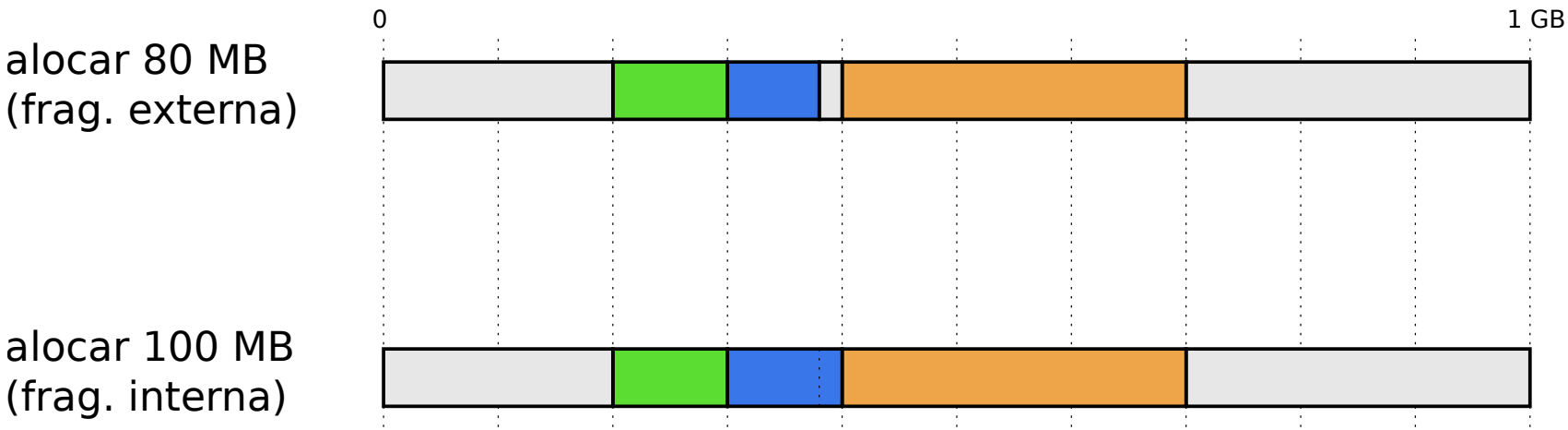
- Para evitar o surgimento de fragmentos muito pequenos
- Diminuir custo de gerência
- Inevitável em sistemas paginados

Processos recebem mais memória que o solicitado:

- Essa memória não será usada
- Perda média: 1/2 página por processo

Fragmentação interna

Exemplo: dada uma requisição de 80 MB de memória:



O alocador Buddy

Princípio:

- Alocar blocos de memória de tamanho 2^n
- Exemplo: p/ pedido de 85 KB, alocar 128 KB (2^7 KB)
- Bloco mínimo: entre 1 KB e 64 KB

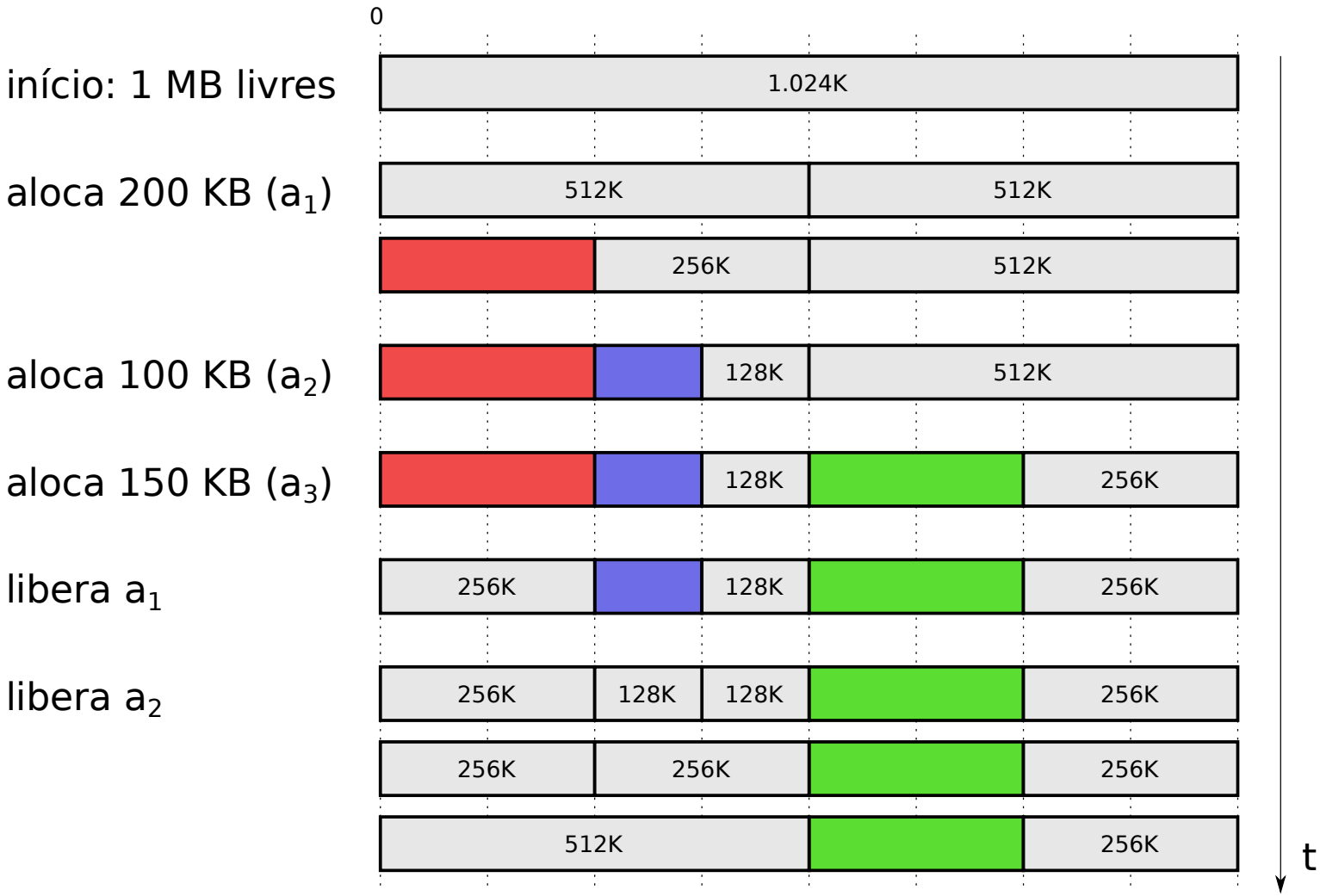
Exemplo: ao alocar um bloco de 40 KB:

1. procura bloco de 64 KB e o aloca
2. se não houver, procura bloco de 128 KB e divide em dois
3. se não houver, procura bloco de 256 KB e ...

Exemplo: ao liberar um bloco:

1. se o "par" (*buddy*) do bloco estiver livre, funde-os
2. se o par do novo bloco estiver livre, funde-os

O alocador Buddy



O alocador Buddy

Variantes do alocador Buddy:

- Buddy binário (sempre dois buddies iguais)
- Buddy binário com pesos (64 KB = 48KB + 16 KB)
- Buddy com Fibonacci

Uso: alocador de memória física no Linux:

```
> cat /proc/buddyinfo

Node 0, zone      DMA    1      0      1      2      3    2    0    0   1   1    3
Node 0, zone    DMA32   12      8      9      7      6   10    6    6   4   3  551
Node 0, zone   Normal  237  15443  11954   3798    984  444  289  133  70  37  126
```

O alocador Slab

Proposto para o sistema SunOS 5.4, amplamente usado hoje.

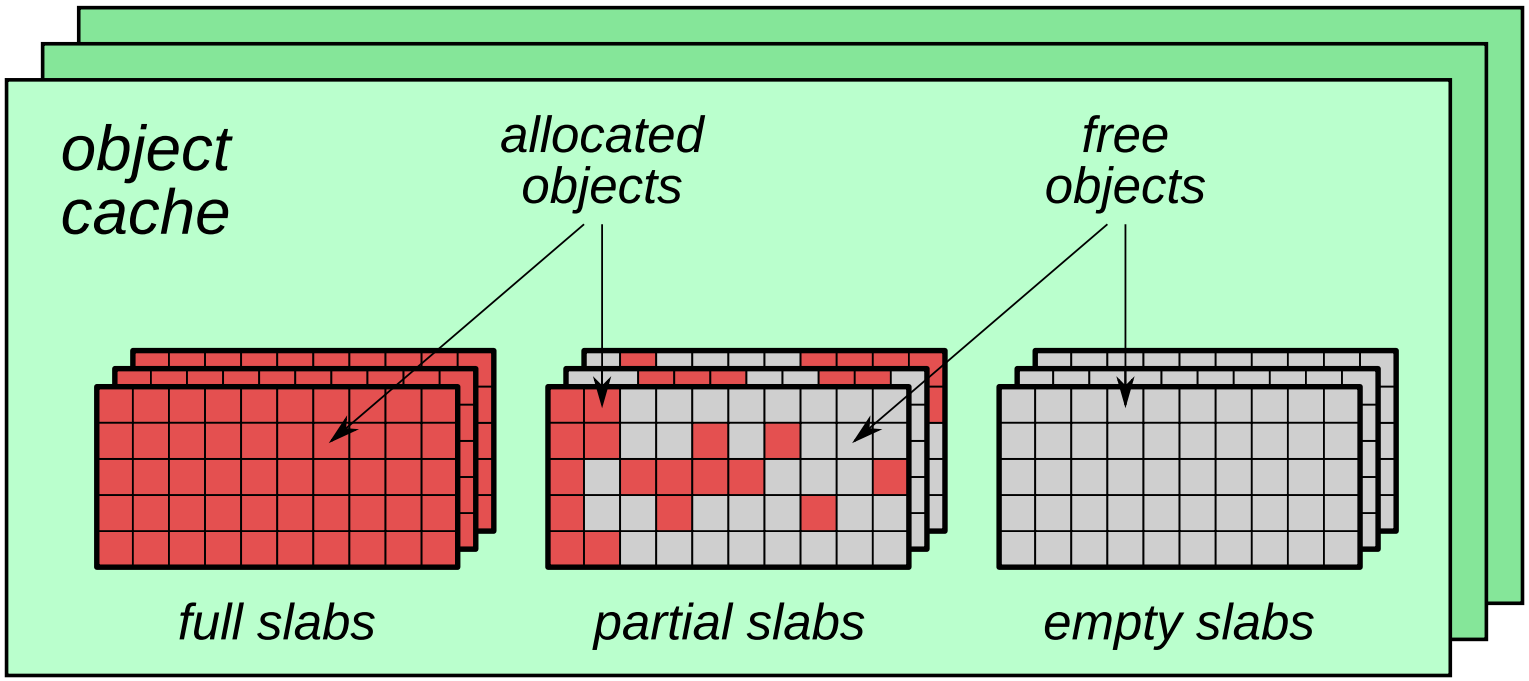
Especializado na alocação de “objetos” do núcleo:

- descritores de arquivos, processos, sockets, etc
- pequenos (10s-100s bytes) e com tamanhos padronizados
- continuamente criados/destruídos na operação do núcleo
- podem ser reutilizados ao invés de alocados/liberados

Estratégia básica: caching de objetos

- Um cache para cada tipo de objeto
- Cada cache é dividido em *slabs* (placas)
- Um *slab* pode estar cheio, vazio ou parcial

O alocador Slab



Funcionamento do alocador Slab

1. Ao receber um pedido de objeto, o alocador analisa o cache adequado e entrega um objeto livre de um *slab* parcial;
2. se não houver *slab* parcial, entrega um objeto livre de um *slab* vazio e muda o status dele para parcial;
3. se não houver *slab* vazio, solicita RAM ao alocador físico para um novo *slab*, ajusta seu status para vazio, inicializa seus objetos e os marca como livres;
4. ao liberar um objeto, este é marcado como livre;
5. se todos os objetos de um *slab* estiverem livres, ele é marcado como vazio;
6. *slabs* vazios podem ser devolvidos ao alocador físico se houver necessidade de RAM no sistema.

O alocador Slab no Linux

```
> cat /proc/slabinfo

# name            <active_objs> <num_objs> <objsize> <obj/slab> <pg/slab>
RAWv6                   252          252        1152          28          8
UDPv6                   104          104        1216          26          8
fat_inode_cache          44           44         744          22          4
fat_cache                 0            0          40         102          1
pid_namespace            76           76         208          19          1
posix_timers_cache      374          374         240          17          1
request_sock_TCP          0            0         304          26          2
TCP                     224          224        2048          16          8
sock_inode_cache       2783         2783         704          23          4
file_lock_cache         160          160         200          20          1
inode_cache           18902        18902         608          26          4
mm_struct               207          225        2112          15          8
files_cache             230          230         704          23          4
signal_cache            346          368        1024          16          4
task_struct            1141         1185        5824           5          8
...
```

Alocação no espaço de usuário

Userspace allocator:

- Gerencia o uso da área *Heap* do processo
- Alocação dinâmica de variáveis
- Implementado em biblioteca (LibC)

Abordagens:

- Mais simples: *best-fit*
- DLmalloc (usado no Linux)
- Solução ad-hoc para aplicações específicas
- Memory pool: prealocar vetor de blocos iguais