Sistemas Operacionais

Working Set
(Conjunto de Trabalho)

Prof. Jose Paulo . de Oliveira
Eng. da Computação, UPE
jpgo@ecomp.poli.br

Working Set

Localidade de referências:

Processos normalmente acessam apenas uma pequena fração de suas páginas a cada instante

O conjunto de páginas acessadas na história recente de um processo é chamado *Conjunto de Trabalho* (*Working Set*) [Peter Denning]

Working Set

Definição:

- *É o menor conjunto de informação que deve estar presente na memória principal para garantir uma execução eficiente do processo*

Working Set

Definição:

- *É o menor conjunto de informação que deve estar presente na memória principal para garantir uma execução eficiente do processo*

- *Nem o programador nem o compilador conseguem determinar, a priori, que informações são essas*

Working Set

Definição:

- *É o menor conjunto de informação que deve estar presente na memória principal para garantir uma execução eficiente do processo*
- *Nem o programador nem o compilador conseguem determinar, a priori, que informações são essas*
- *Cabe ao SO, com base nos padrões de referência das páginas, determinar quais páginas estão em uso*

Working Set

Definição:

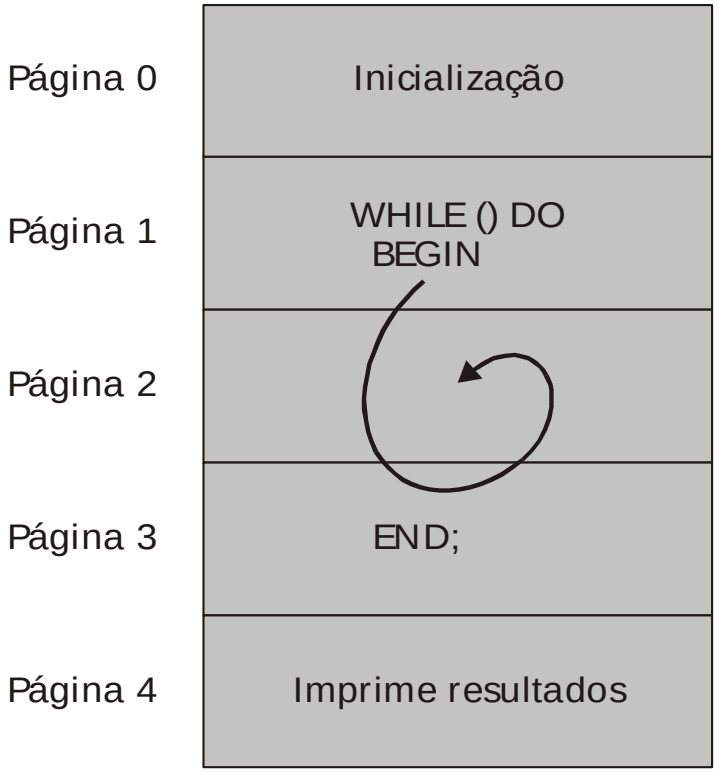
- *É o menor conjunto de informação que deve estar presente na memória principal para garantir uma execução eficiente do processo*
- *Nem o programador nem o compilador conseguem determinar, a priori, que informações são essas*
- *Cabe ao SO, com base nos padrões de referência das páginas, determinar quais páginas estão em uso*
- *O **conjunto de trabalho**, portanto, é formado pelas últimas páginas acessadas*

Working Set

- A composição do **conjunto de trabalho** é **dinâmica**
- Varia à medida que o processo executa e evolui seu comportamento, acessando novas páginas e deixando de acessar outras
- Ou seja, depende do instante de tempo de execução *t*

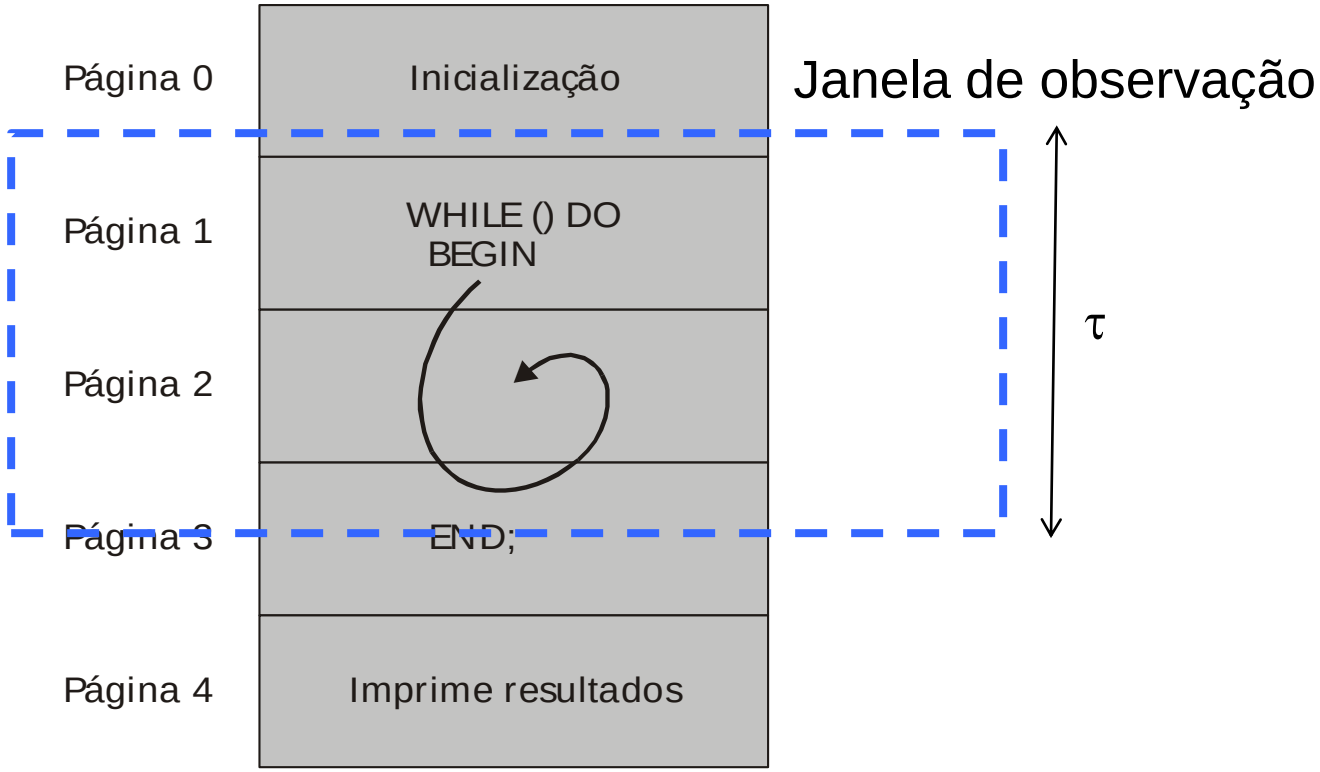
Working Set

Conceito de localidade



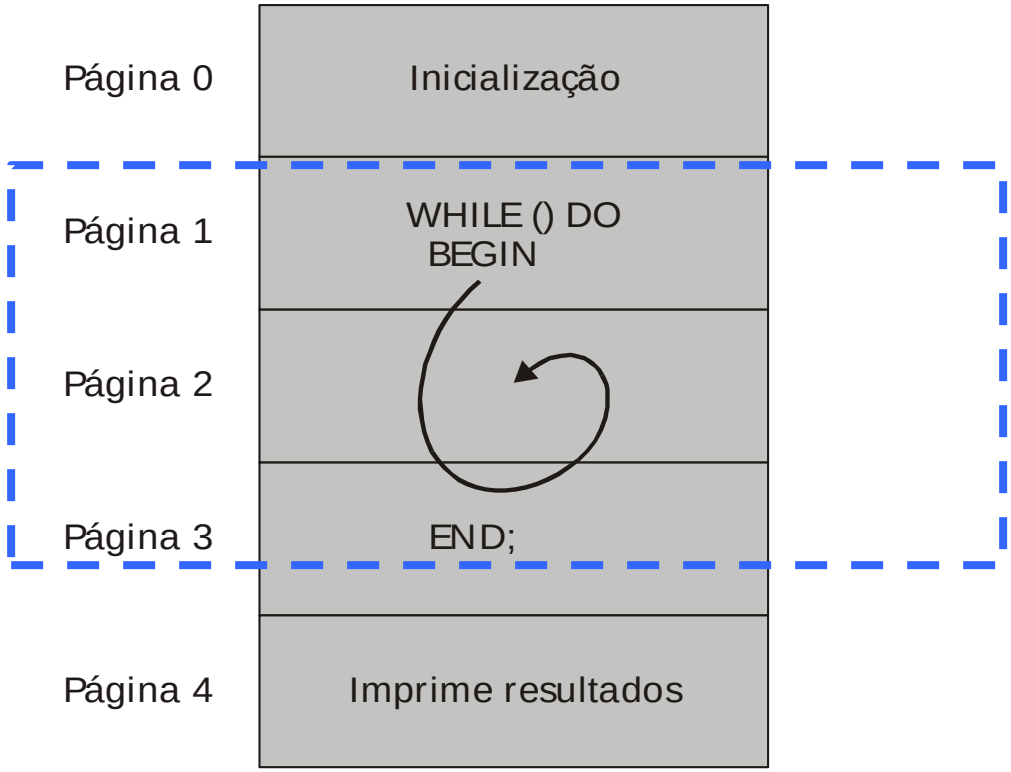
Working Set

Conceito de localidade



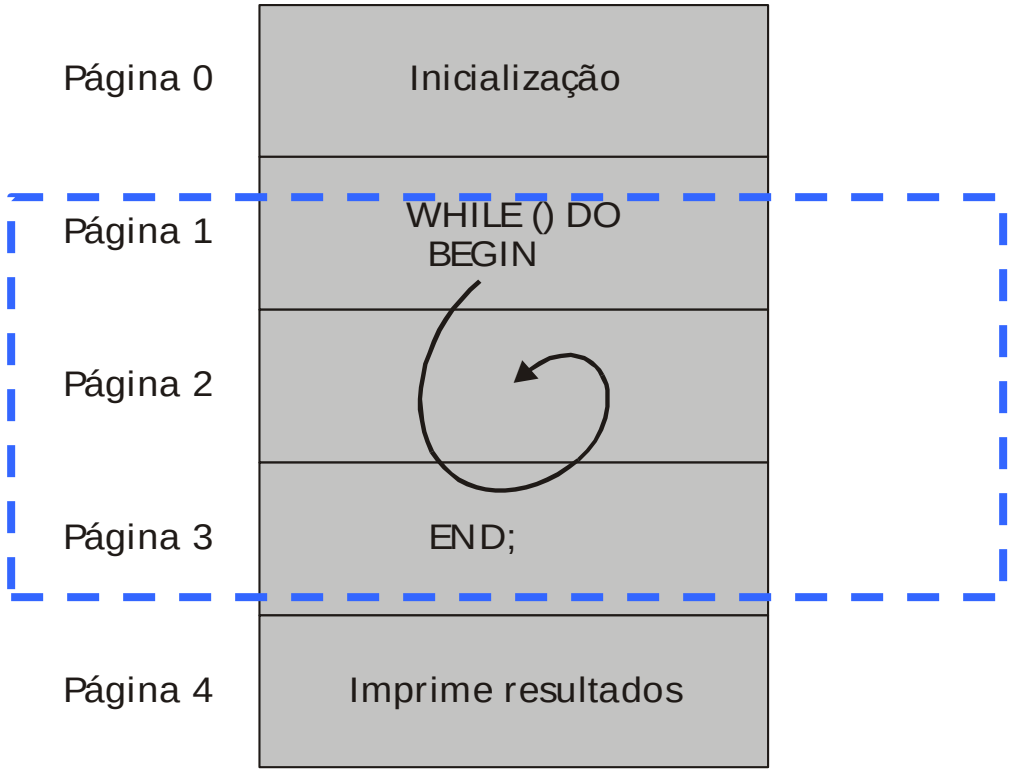
Working Set

Conceito de localidade



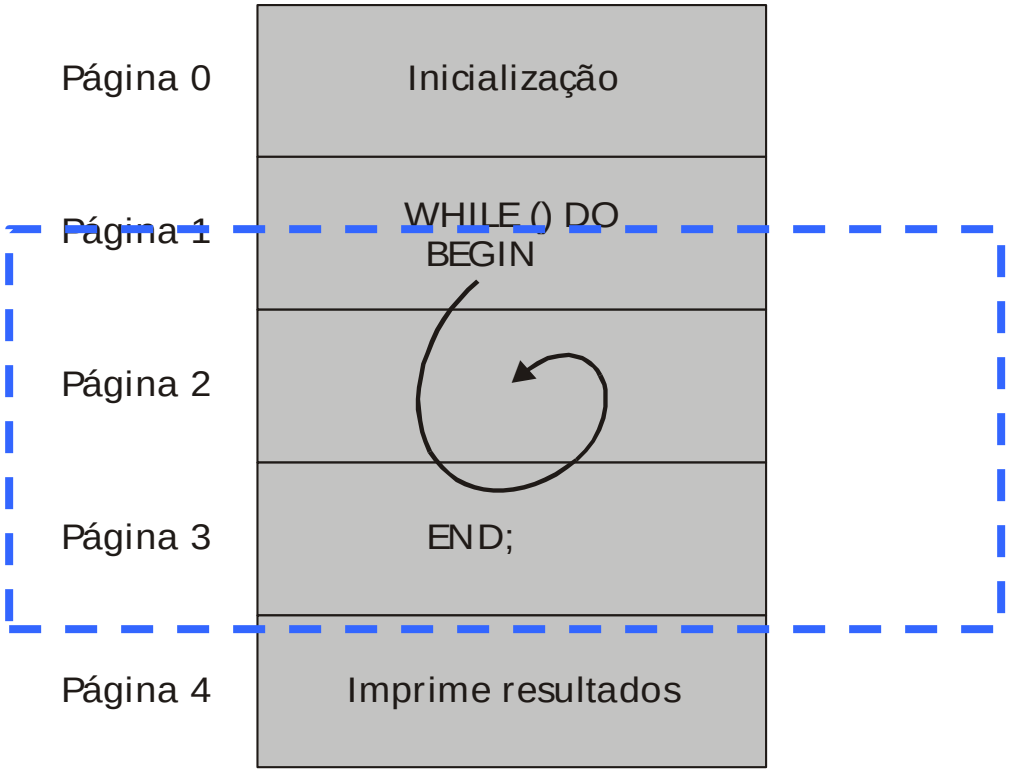
Working Set

Conceito de localidade



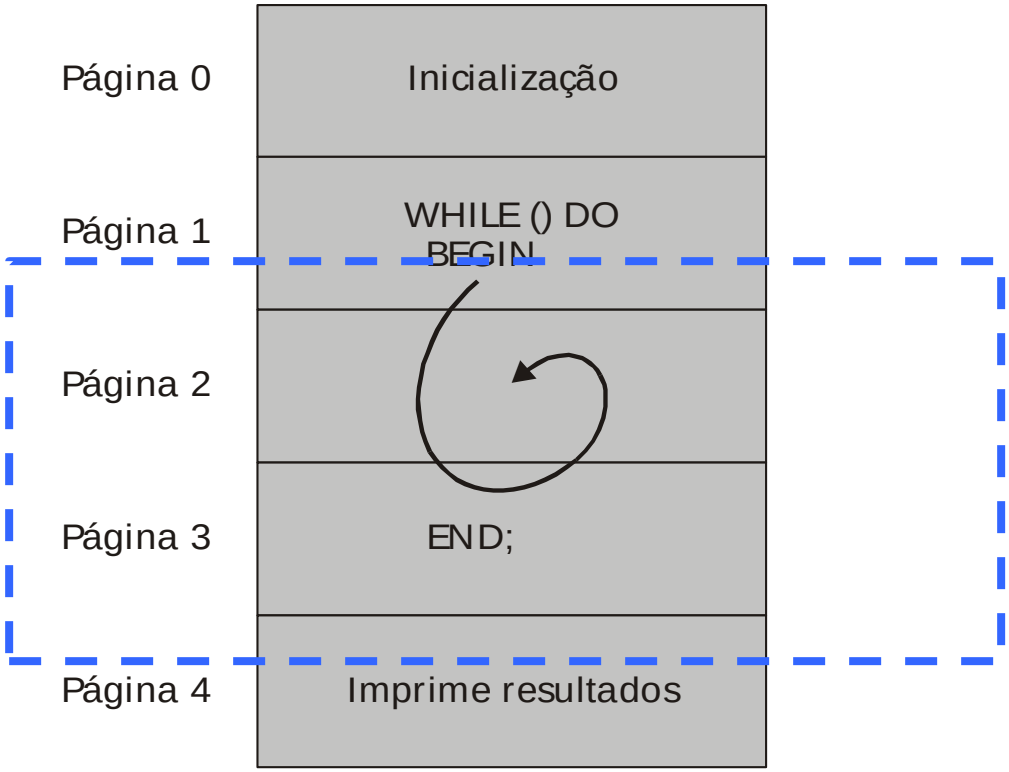
Working Set

Conceito de localidade



Working Set

Conceito de localidade



Working Set

- O tamanho e a composição do conjunto de trabalho também dependem do número de páginas consideradas em sua história recente

Working Set

- O tamanho e a composição do conjunto de trabalho também dependem do número de páginas consideradas em sua história recente
- Ou, similarmente, ao período de observação τ da história recente

Working Set

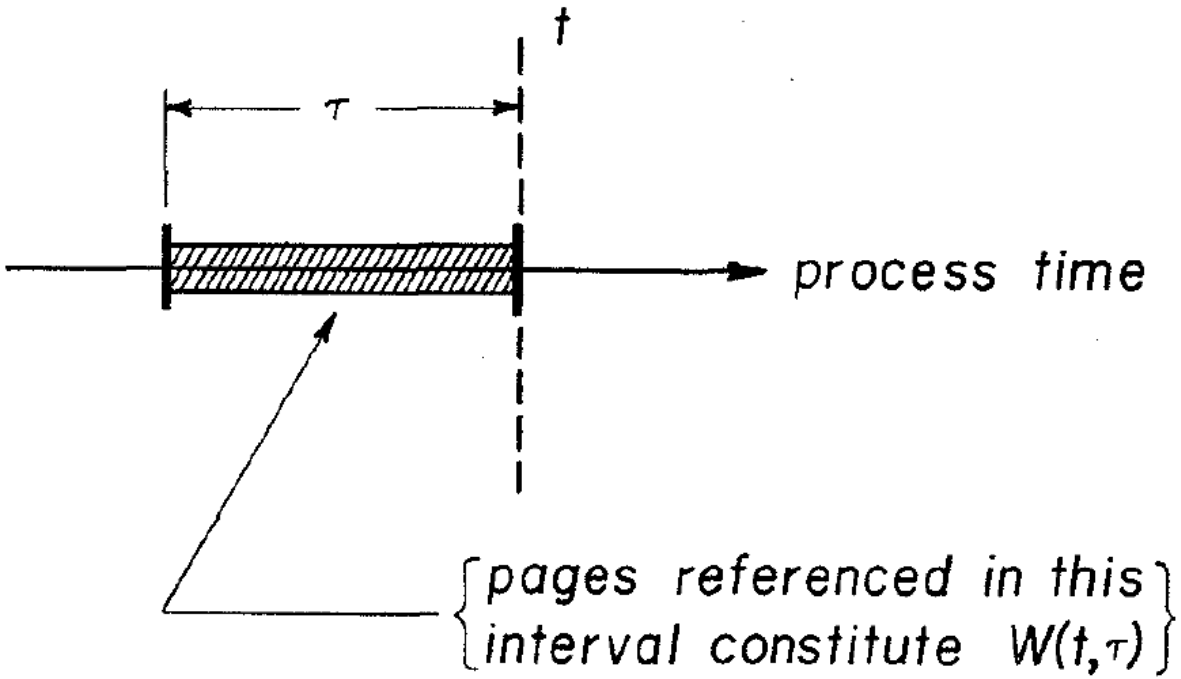


FIG. 2. Definition of $W(t, \tau)$

Working Set

- O tamanho e a composição do conjunto de trabalho também dependem do número de páginas consideradas em sua história recente
- Ou, similarmente, ao período de observação τ da história recente
- Similarmente: depende da quantidade de acessos observados durante $\boldsymbol{\tau}$

Exemplo:

t	página	$ws(n = 3)$
1	0	{0}
2	2	{0, 2}
3	1	{0, 2, 1}
4	3	{2, 1, 3}
5	5	{1, 3, 5}
6	4	{3, 5, 4}
7	6	{5, 4, 6}
8	3	{4, 6, 3}
9	7	{6, 3, 7}
10	4	{3, 7, 4}
11	7	{4, 7}
12	3	{4, 7, 3}
13	3	{7, 3}
14	5	{3, 5}
15	5	{3, 5}
16	3	{5, 3}
17	1	{5, 3, 1}
18	1	{3, 1}
19	1	{1}
20	7	{1, 7}
21	1	{7, 1}
22	3	{7, 1, 3}
23	4	{1, 3, 4}
24	1	{3, 4, 1}

Cadeia de referências

Exemplo:

Cadeia de referências

t	página	$ws(n = 3)$	$ws(n = 4)$
1	0	{0}	{0}
2	2	{0, 2}	{0, 2}
3	1	{0, 2, 1}	{0, 2, 1}
4	3	{2, 1, 3}	{0, 2, 1, 3}
5	5	{1, 3, 5}	{2, 1, 3, 5}
6	4	{3, 5, 4}	{1, 3, 5, 4}
7	6	{5, 4, 6}	{3, 5, 4, 6}
8	3	{4, 6, 3}	{5, 4, 6, 3}
9	7	{6, 3, 7}	{4, 6, 3, 7}
10	4	{3, 7, 4}	{6, 3, 7, 4}
11	7	{4, 7}	{3, 4, 7}
12	3	{4, 7, 3}	{4, 7, 3}
13	3	{7, 3}	{4, 7, 3}
14	5	{3, 5}	{7, 3, 5}
15	5	{3, 5}	{3, 5}
16	3	{5, 3}	{5, 3}
17	1	{5, 3, 1}	{5, 3, 1}
18	1	{3, 1}	{5, 3, 1}
19	1	{1}	{3, 1}
20	7	{1, 7}	{1, 7}
21	1	{7, 1}	{7, 1}
22	3	{7, 1, 3}	{7, 1, 3}
23	4	{1, 3, 4}	{7, 1, 3, 4}
24	1	{3, 4, 1}	{3, 4, 1}

Exemplo:

Cadeia de referências

t	página	$ws(n = 3)$	$ws(n = 4)$	$ws(n = 5)$
1	0	{0}	{0}	{0}
2	2	{0, 2}	{0, 2}	{0, 2}
3	1	{0, 2, 1}	{0, 2, 1}	{0, 2, 1}
4	3	{2, 1, 3}	{0, 2, 1, 3}	{0, 2, 1, 3}
5	5	{1, 3, 5}	{2, 1, 3, 5}	{0, 2, 1, 3, 5}
6	4	{3, 5, 4}	{1, 3, 5, 4}	{2, 1, 3, 5, 4}
7	6	{5, 4, 6}	{3, 5, 4, 6}	{1, 3, 5, 4, 6}
8	3	{4, 6, 3}	{5, 4, 6, 3}	{5, 4, 6, 3}
9	7	{6, 3, 7}	{4, 6, 3, 7}	{5, 4, 6, 3, 7}
10	4	{3, 7, 4}	{6, 3, 7, 4}	{6, 3, 7, 4}
11	7	{4, 7}	{3, 4, 7}	{6, 3, 4, 7}
12	3	{4, 7, 3}	{4, 7, 3}	{4, 7, 3}
13	3	{7, 3}	{4, 7, 3}	{4, 7, 3}
14	5	{3, 5}	{7, 3, 5}	{4, 7, 3, 5}
15	5	{3, 5}	{3, 5}	{7, 3, 5}
16	3	{5, 3}	{5, 3}	{5, 3}
17	1	{5, 3, 1}	{5, 3, 1}	{5, 3, 1}
18	1	{3, 1}	{5, 3, 1}	{5, 3, 1}
19	1	{1}	{3, 1}	{5, 3, 1}
20	7	{1, 7}	{1, 7}	{3, 1, 7}
21	1	{7, 1}	{7, 1}	{3, 7, 1}
22	3	{7, 1, 3}	{7, 1, 3}	{7, 1, 3}
23	4	{1, 3, 4}	{7, 1, 3, 4}	{7, 1, 3, 4}
24	1	{3, 4, 1}	{3, 4, 1}	{7, 3, 4, 1}

Working Set

- O tamanho e a composição do conjunto de trabalho também dependem do número de páginas consideradas em sua história recente

- Em sistemas reais, essa dependência não é linear, mas segue uma **proporção logaritmica**, devido à localidade de referências

Working Set

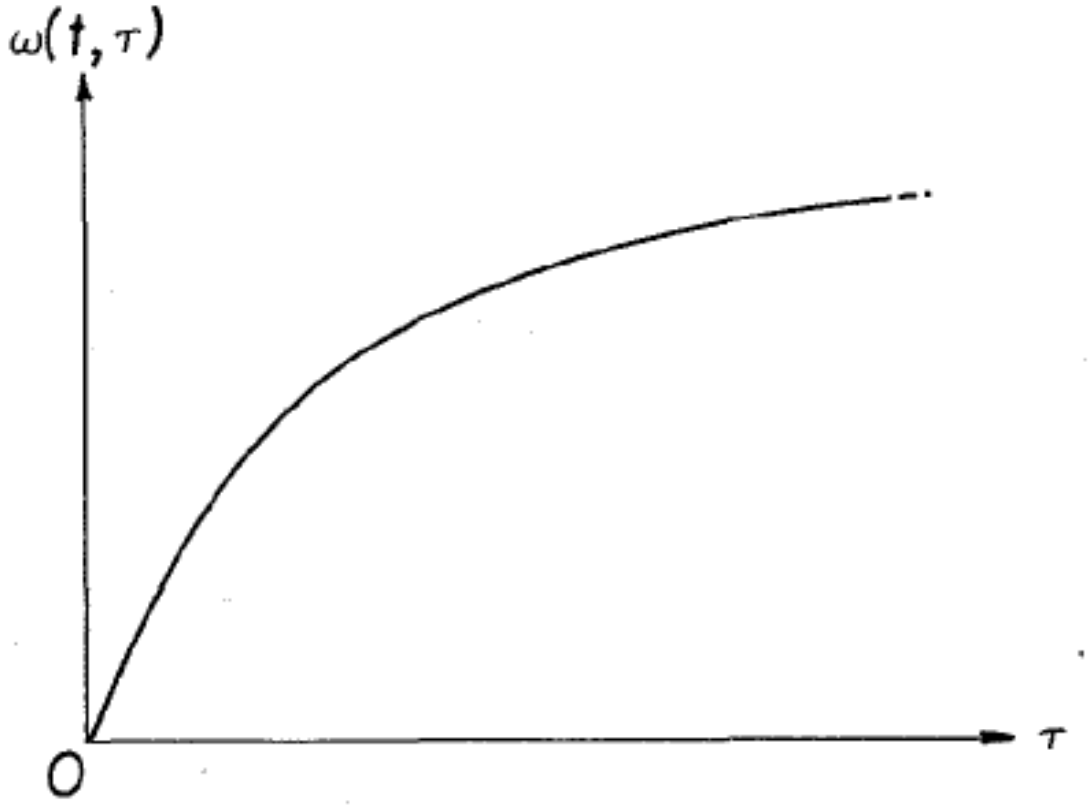


FIG. 3. Behavior of $\omega(t, \tau)$

Working Set

Isso pode ser observado na tabela:

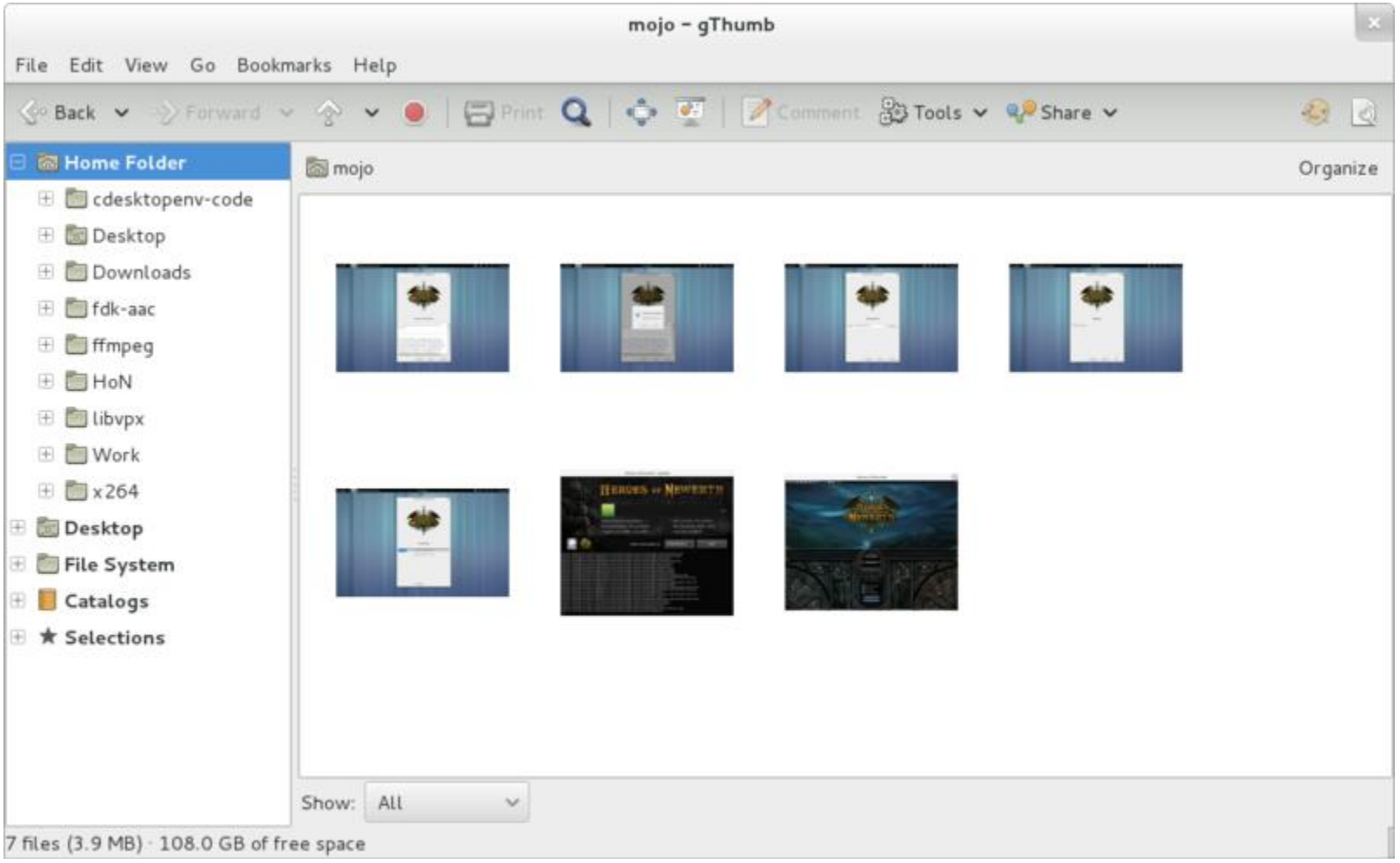
t	página	$ws(n = 3)$	$ws(n = 4)$	$ws(n = 5)$
1	0	{0}	{0}	{0}
2	2	{0, 2}	{0, 2}	{0, 2}
3	1	{0, 2, 1}	{0, 2, 1}	{0, 2, 1}
4	3	{2, 1, 3}	{0, 2, 1, 3}	{0, 2, 1, 3}
5	5	{1, 3, 5}	{2, 1, 3, 5}	{0, 2, 1, 3, 5}
6	4	{3, 5, 4}	{1, 3, 5, 4}	{2, 1, 3, 5, 4}
7	6	{5, 4, 6}	{3, 5, 4, 6}	{1, 3, 5, 4, 6}
8	3	{4, 6, 3}	{5, 4, 6, 3}	{5, 4, 6, 3}
9	7	{6, 3, 7}	{4, 6, 3, 7}	{5, 4, 6, 3, 7}
10	4	{3, 7, 4}	{6, 3, 7, 4}	{6, 3, 7, 4}
11	7	{4, 7}	{3, 4, 7}	{6, 3, 4, 7}
12	3	{4, 7, 3}	{4, 7, 3}	{4, 7, 3}
13	3	{7, 3}	{4, 7, 3}	{4, 7, 3}
14	5	{3, 5}	{7, 3, 5}	{4, 7, 3, 5}
15	5	{3, 5}	{3, 5}	{7, 3, 5}
16	3	{5, 3}	{5, 3}	{5, 3}
17	1	{5, 3, 1}	{5, 3, 1}	{5, 3, 1}
18	1	{3, 1}	{5, 3, 1}	{5, 3, 1}
19	1	{1}	{3, 1}	{5, 3, 1}
20	7	{1, 7}	{1, 7}	{3, 1, 7}
21	1	{7, 1}	{7, 1}	{3, 7, 1}
22	3	{7, 1, 3}	{7, 1, 3}	{7, 1, 3}
23	4	{1, 3, 4}	{7, 1, 3, 4}	{7, 1, 3, 4}
24	1	{3, 4, 1}	{3, 4, 1}	{7, 3, 4, 1}

Working Set

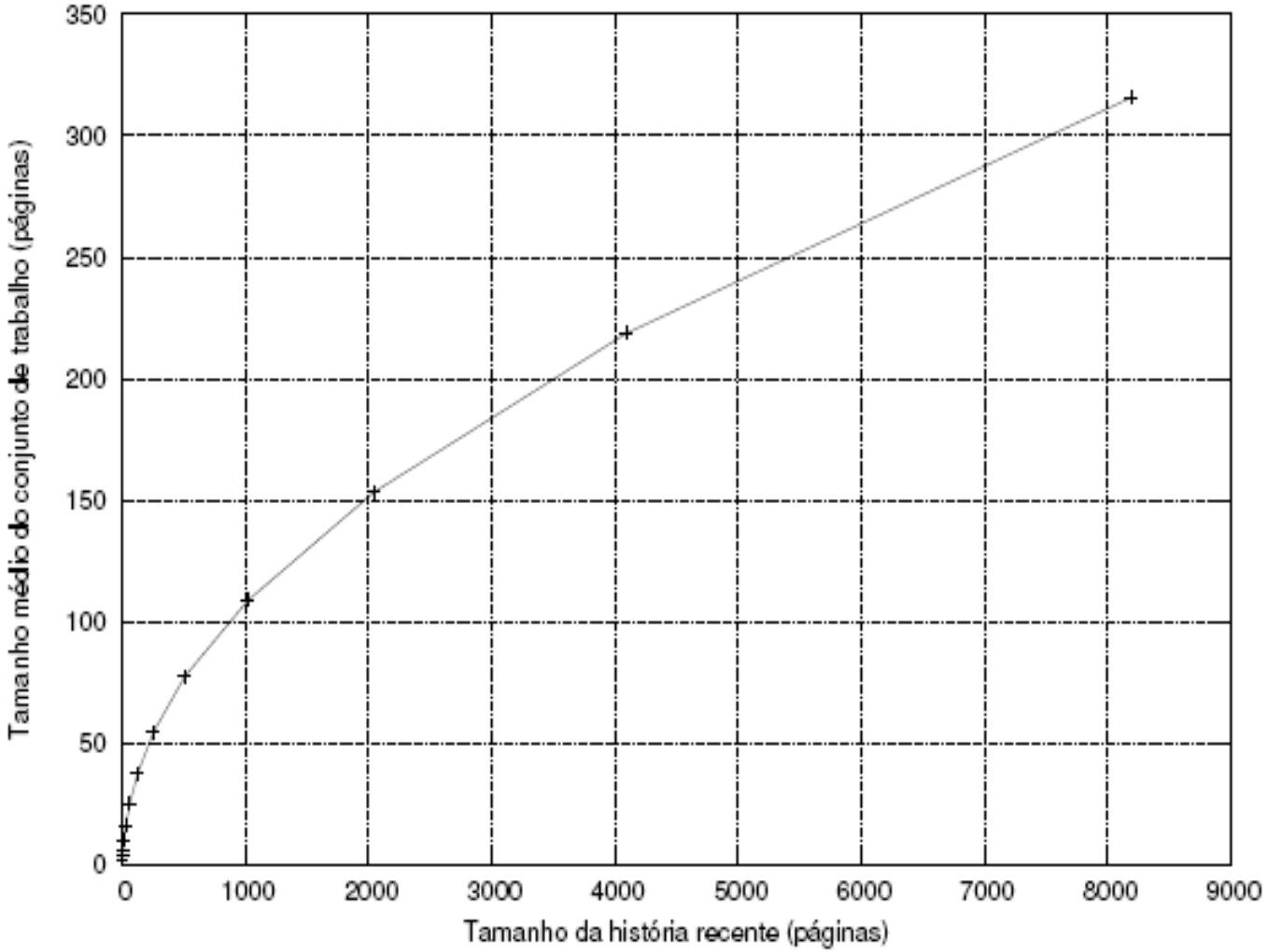
t	página	$ws(n = 3)$	$ws(n = 4)$	$ws(n = 5)$
1	0	{0}	{0}	{0}
2	2	{0, 2}	{0, 2}	{0, 2}
3	1	{0, 2, 1}	{0, 2, 1}	{0, 2, 1}
4	3	{2, 1, 3}	{0, 2, 1, 3}	{0, 2, 1, 3}
10	4	{3, 7, 4}	{6, 3, 7, 4}	{6, 3, 7, 4}
11	7	{4, 7}	{3, 4, 7}	{6, 3, 4, 7}
12	3	{4, 7, 3}	{4, 7, 3}	{4, 7, 3}
13	3	{7, 3}	{4, 7, 3}	{4, 7, 3}
14	5	{3, 5}	{7, 3, 5}	{4, 7, 3, 5}
15	5	{3, 5}	{3, 5}	{7, 3, 5}
16	3	{5, 3}	{5, 3}	{5, 3}
17	1	{5, 3, 1}	{5, 3, 1}	{5, 3, 1}
18	1	{3, 1}	{5, 3, 1}	{5, 3, 1}
19	1	{1}	{3, 1}	{5, 3, 1}
20	7	{1, 7}	{1, 7}	{3, 1, 7}
21	1	{7, 1}	{7, 1}	{3, 7, 1}
22	3	{7, 1, 3}	{7, 1, 3}	{7, 1, 3}
23	4	{1, 3, 4}	{7, 1, 3, 4}	{7, 1, 3, 4}
24	1	{3, 4, 1}	{3, 4, 1}	{7, 3, 4, 1}

Assim que a localidade de referências se torna mais forte (a partir de $t = 12$), os três conjuntos de trabalho ficam muito similares

Exemplo - gThumb

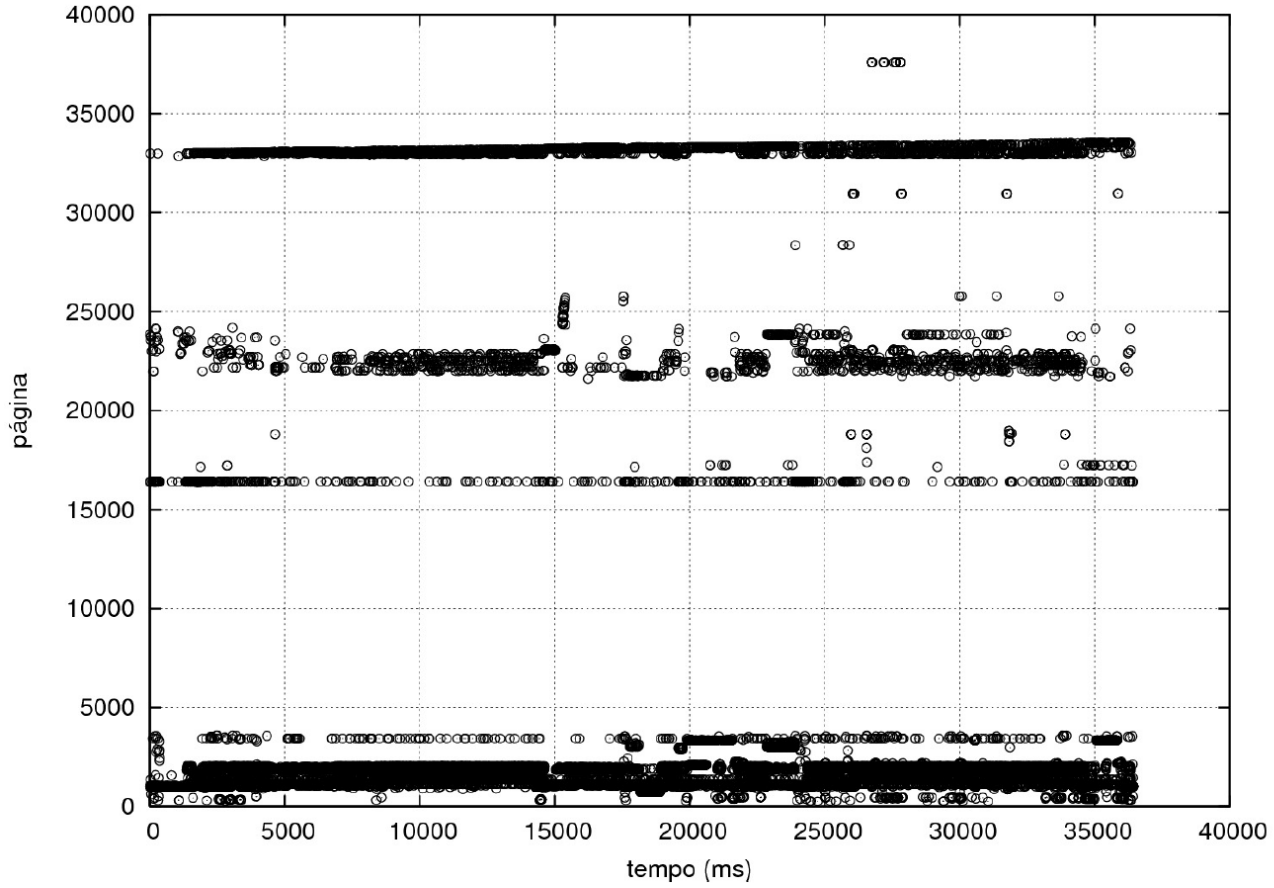


Exemplo - gThumb



Localidade de referências

Exemplo de aplicativo real (*gThumb*):



Algoritmo de substituição baseado no Working Set

- Se um processo tiver todas as páginas de seu conjunto de trabalho carregadas na memória sofrerá poucas faltas de página

Algoritmo de substituição baseado no Working Set

- Se um processo tiver todas as páginas de seu conjunto de trabalho carregadas na memória sofrerá poucas faltas de página

- Essa constatação permite delinear um **algoritmo simples** para substituição de páginas:

  – Substituir páginas que não pertençam ao conjunto de trabalho de nenhum processo ativo

Algoritmo de substituição baseado no Working Set

- Contudo, esse algoritmo é difícil de implementar
  - Exigiria manter atualizado o conjunto de trabalho do processo a cada acesso à memória → custo computacional proibitivo

Algoritmo de substituição baseado no Working Set

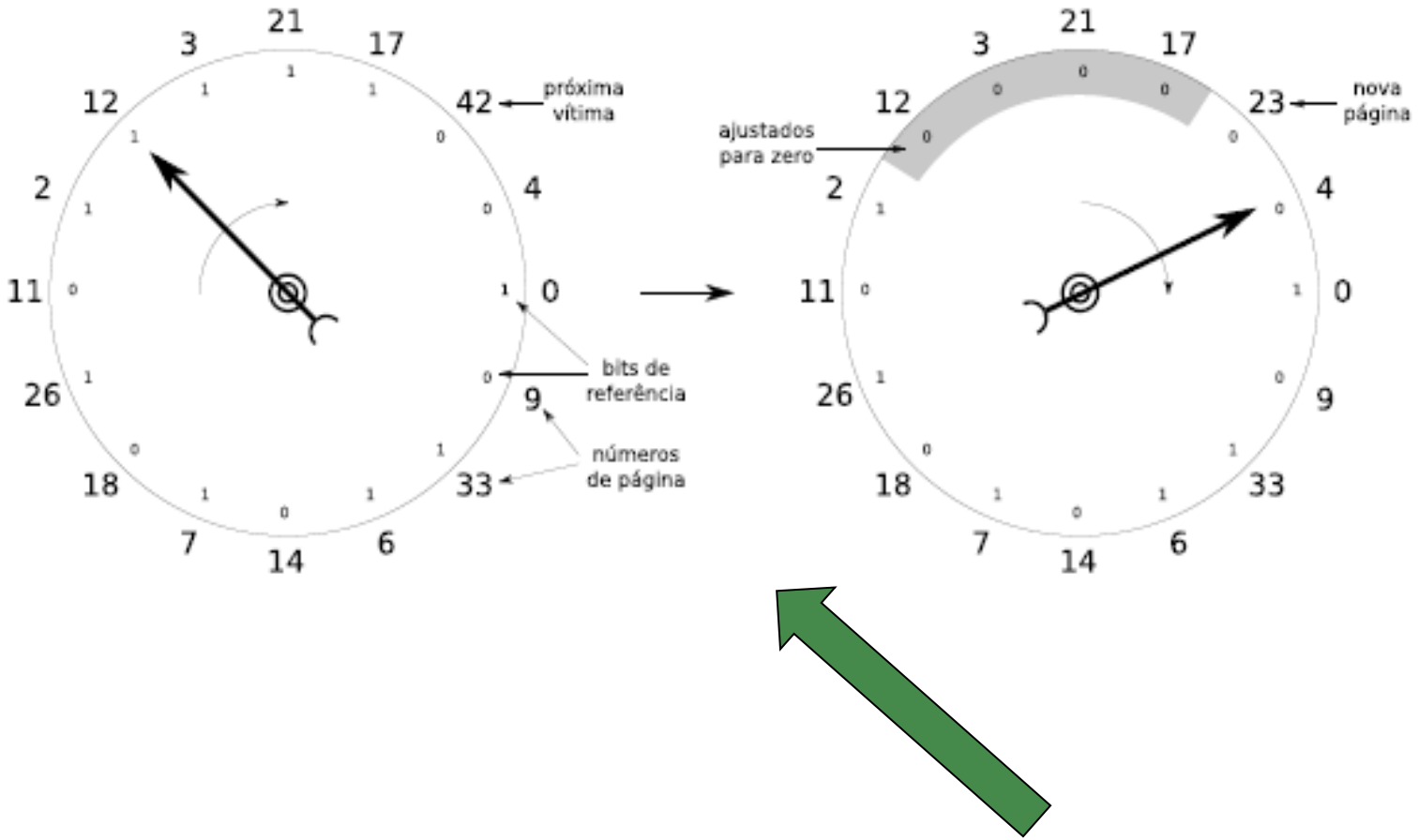
- Contudo, esse algoritmo é difícil de implementar
  - Exigiria manter atualizado o conjunto de trabalho do processo a cada acesso à memória → custo computacional proibitivo
- Uma alternativa mais simples e eficiente de implementar seria verificar que páginas cada processo acessou recentemente, usando a informação dos respectivos bits de referência (R)

Algoritmo de substituição baseado no Working Set

- Contudo, esse algoritmo é difícil de implementar
  - Exigiria manter atualizado o conjunto de trabalho do processo a cada acesso à memória → custo computacional proibitivo
- Uma alternativa mais simples e eficiente de implementar seria verificar que páginas cada processo acessou recentemente, usando a informação dos respectivos bits de referência (R)

Algoritmo WSClock (*Working Set Clock*)

Algoritmo WSClock (*Working Set Clock*)



Similar ao **algoritmo do relógio (*segunda chance*)**

Algoritmo WSClock (*Working Set Clock*)

1. Uma data de último acesso $t_a(p)$ é associada a cada página p

2. Define-se um prazo de validade τ para as páginas

3. A “idade” $i(p)$ de uma página p no instante atual t_c é:

$$i(p) = t_c - t_a(p)$$

Algoritmo WSClock (*Working Set Clock*)

4. Quando há necessidade de substituir páginas:

4.1 Ao encontrar uma página referenciada ($R = 1$):

- Sua data de último acesso é atualizada com o valor corrente do tempo ($t_a(p) = t_c$)
  - $\boldsymbol{i(p) = t_c - t_a(p) = 0}$
- Seu bit de referência é *ressetado* ($R = 0$)
- O ponteiro do relógio avança, ignorando aquela página

Algoritmo WSClock (*Working Set Clock*)

4. Quando há necessidade de substituir páginas:

4.2 Ao encontrar uma página não referenciada ($R = 0$):

- Se sua idade for menor que τ, a página está no conjunto de trabalho
  - Não é substituída
  - t_a não é atualizado
- Caso contrário, ela é considerada fora do conjunto de trabalho e é **substituída**

Relembrando o *Working Set*

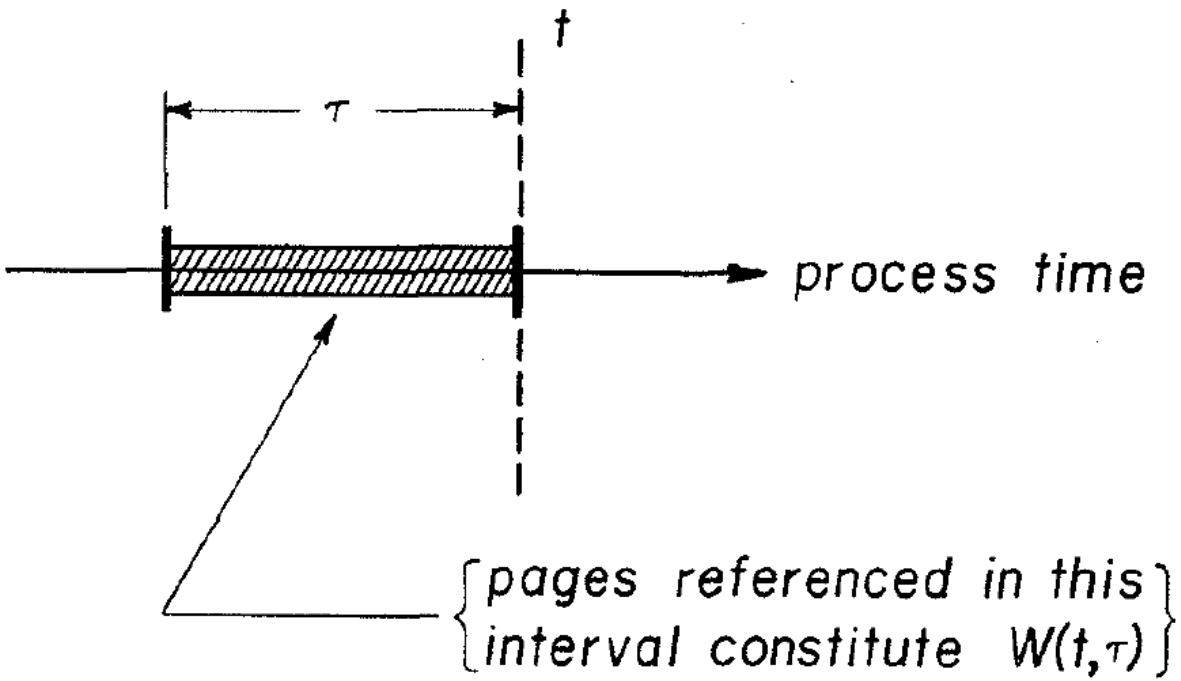
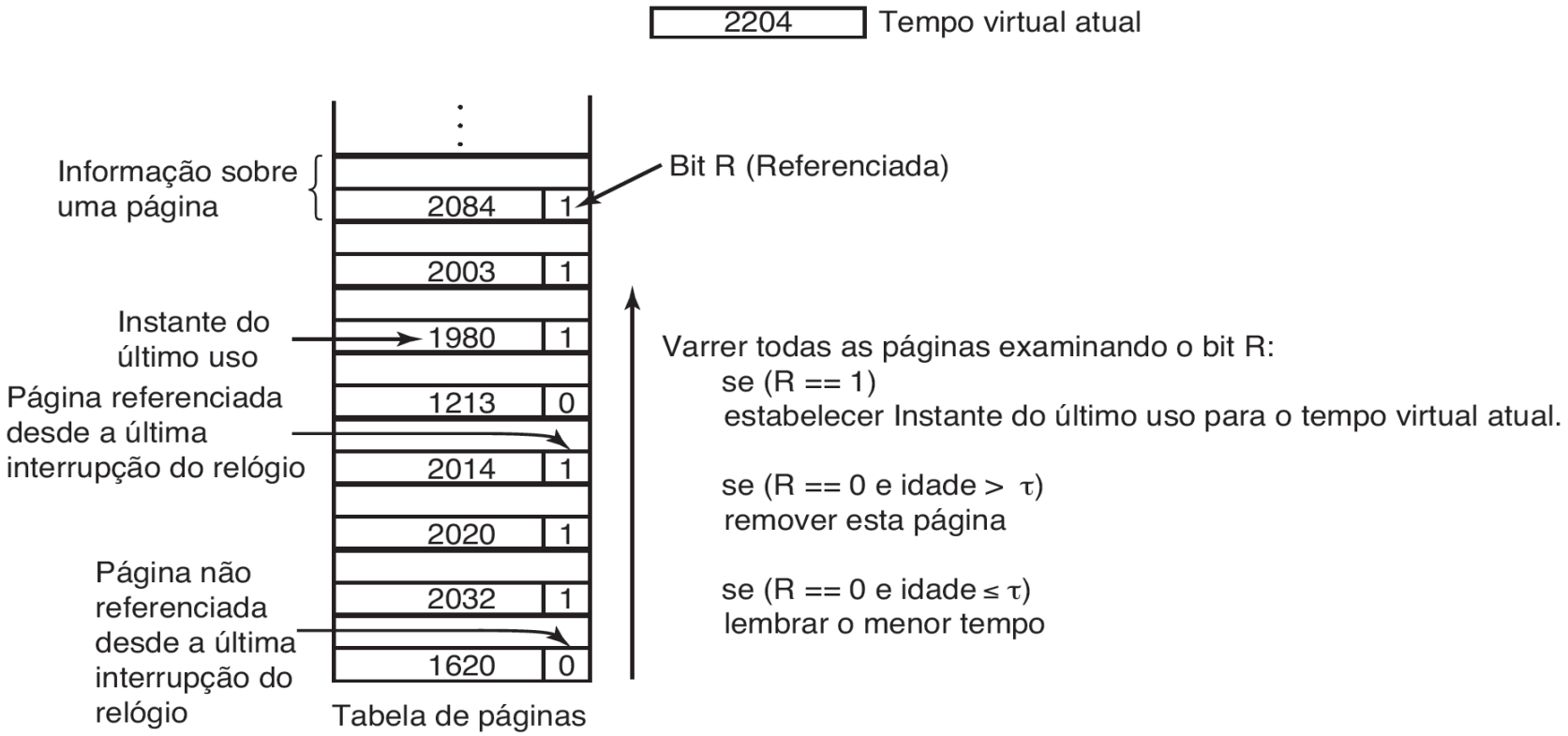
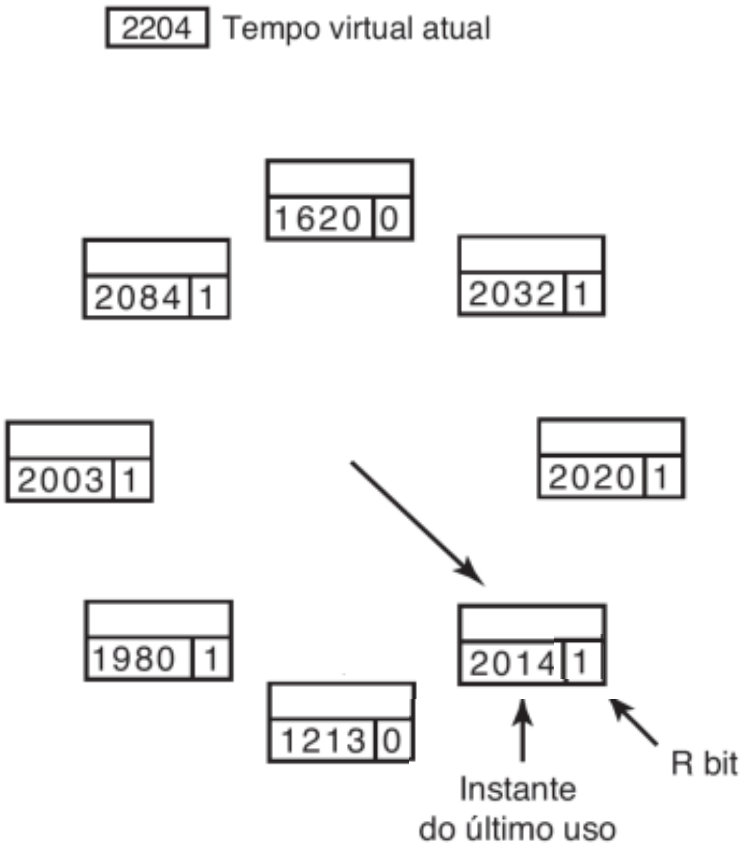


FIG. 2. Definition of $W(t, \tau)$

Algoritmo WSClock - implementação

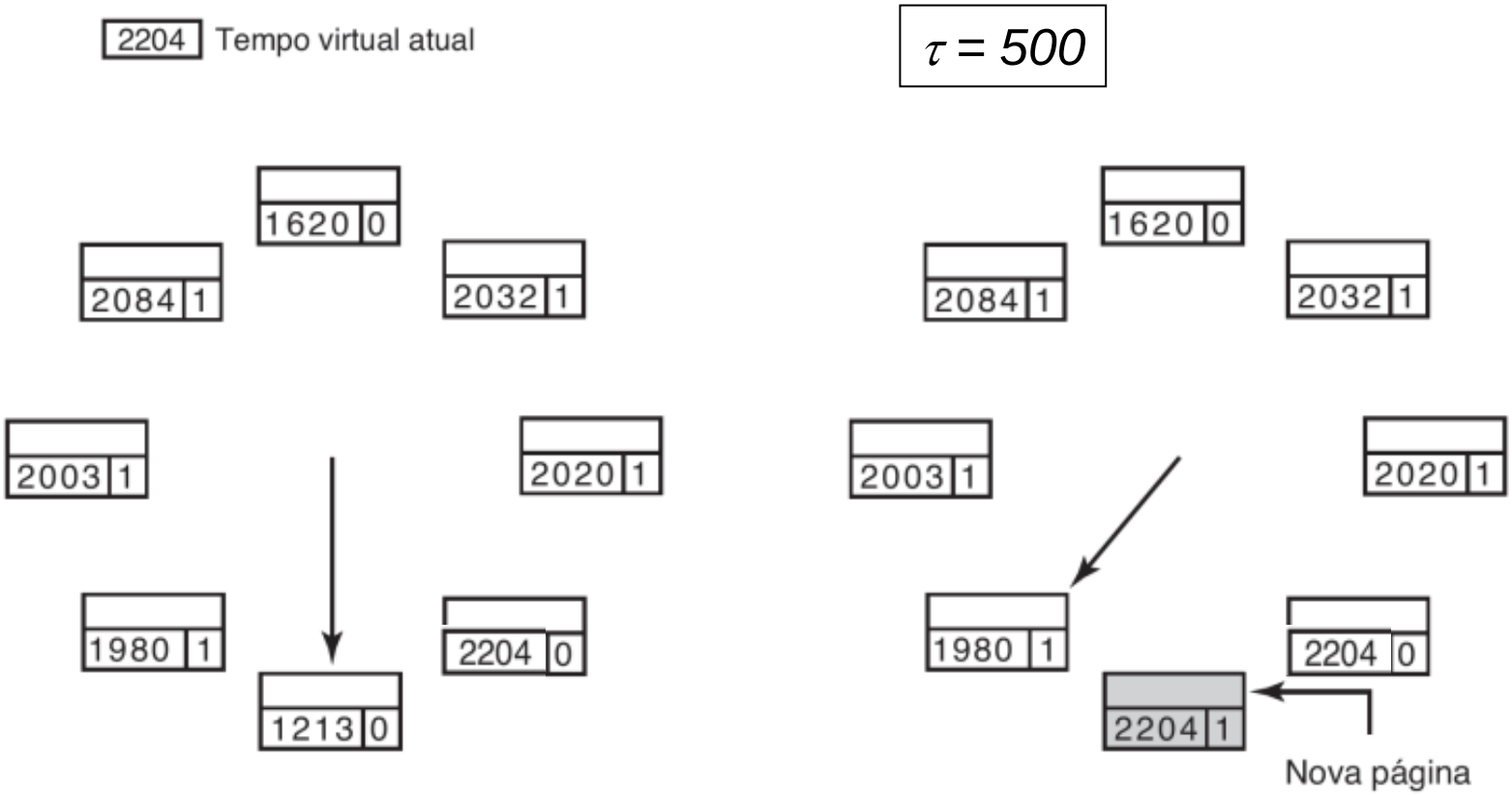


Algoritmo WSClock - exemplo



$\tau = 500$

Algoritmo WSClock - exemplo



2204 – 1213 = 991 > τ

Algoritmo WSClock - exemplo

5. Caso o ponteiro dê uma volta completa na fila e não encontre páginas com idade maior que τ, a página mais antiga [menor $t_a(p)$] encontrada na volta anterior é substituída

6. **Em todas as escolhas, dá-se preferência a páginas não-modificadas (*M* = 0), pois seu conteúdo já está salvo no disco**

Algoritmo WSClock - exemplo

O algoritmo *WSClock* pode ser implementado de forma eficiente

- A data do último acesso de cada página não precisa ser atualizada a cada acesso à memória
- Somente quando a referência da página na fila circular é visitada pelo ponteiro do relógio (caso $R = 1$)

WS: composição de conceitos de vários algoritmos

- **FIFO e segunda-chance**

  estrutura ordenada e percurso do relógio

- **Conjuntos de trabalho**

  divisão das páginas em dois grupos conforme a idade

- **LRU**

  escolha das páginas com datas de acesso mais antigas

- **NRU**

  preferência às páginas não modificadas

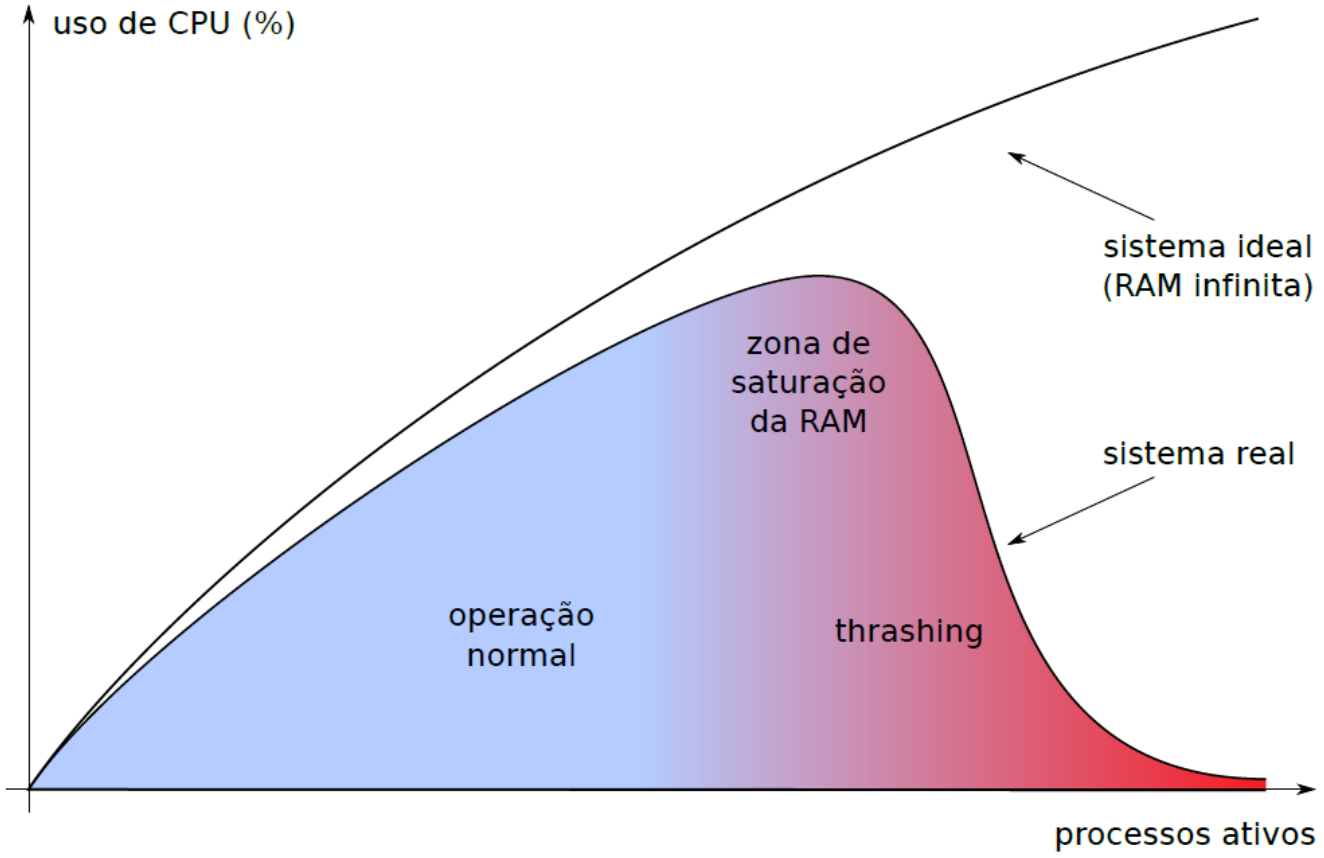
Thrashing

Thrashing

"Transferência excessiva de páginas ou segmentos entre a memória principal e a secundária."

Thrashing

"Transferência excessiva de páginas ou segmentos entre a memória principal e a secundária."



Tipos de Thrashing

1 - *Thrashing* de processo

***2 - Thrashing* de sistema**

1 - *Thrashing* de processo

- Devido ao número elevado de falta de páginas

$\Rightarrow$ O processo fica mais tempo esperando a página do que em execução

1 - *Thrashing* de processo

- Devido ao número elevado de falta de páginas

$\Rightarrow$ O processo fica mais tempo esperando a página do que em execução

Causas:

- Dimensionamento incorreto do limite máximo de páginas – muito menor que seu *working set*
  - *Causando muitas faltas de páginas*
- Ausência do princípio de localidade de referências

2 - *Thrashing* de sistema

- Ocorre quando existem mais processos do que memória disponível

Soluções:

  - Redução do limite máximo de páginas de cada processo
    - Entretanto, causaria *Thrashing* de processo
  - Swapping – transferência de processos (inteiros) da memória principal para a secundária

2 - *Thrashing* de sistema

- Ocorre quando existem mais processos do que memória disponível

Soluções:

  - Redução do limite máximo de páginas de cada processo
    - Entretanto, causaria *Thrashing* de processo
  - Swapping – transferência de processos (inteiros) da memória principal para a secundária
  - A solução adequada, neste caso, é disponibilizar mais memória