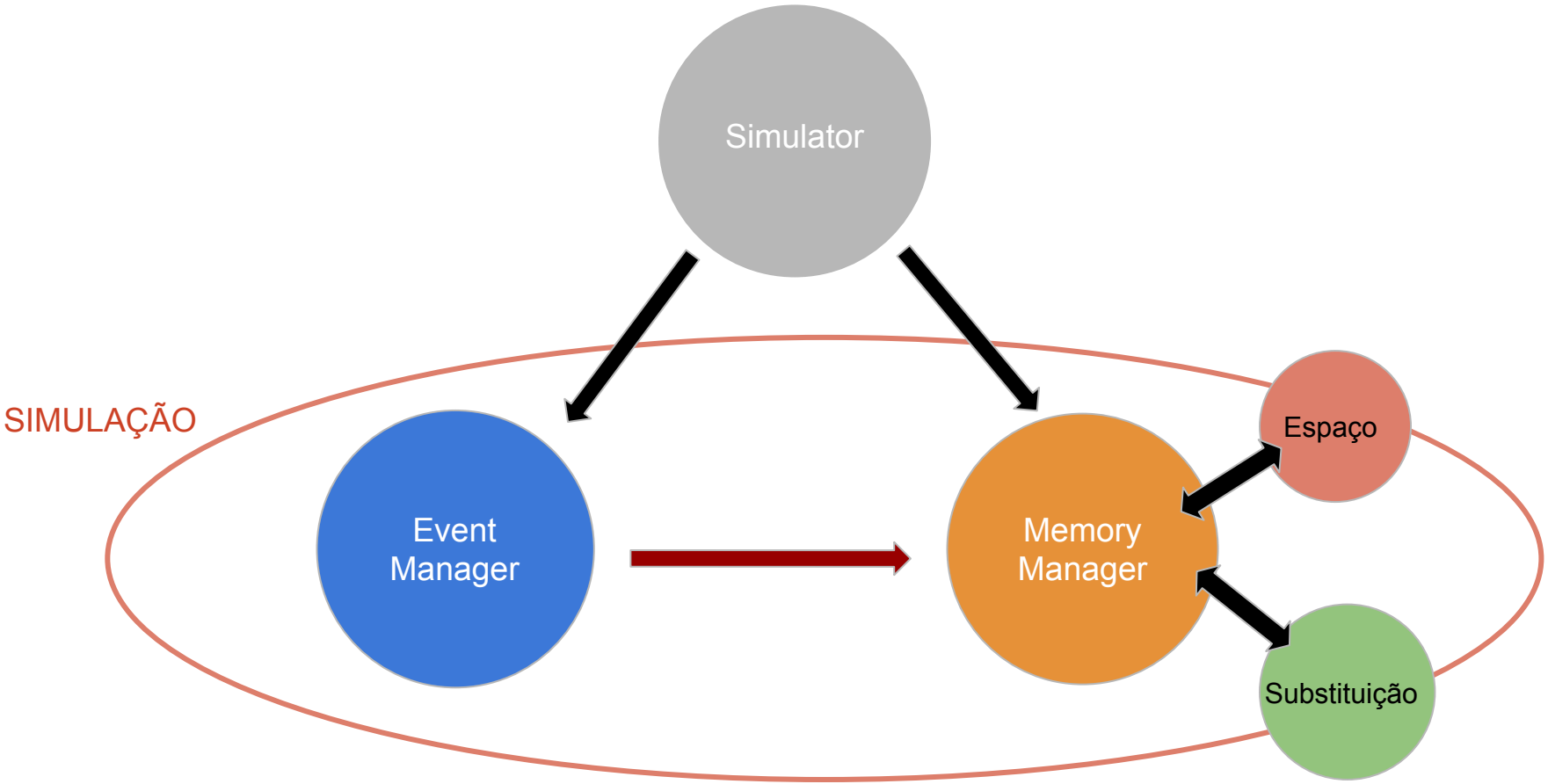
Simulador de Memória

EP2 - Sistemas Operarionais

Fellipe Souto
Pedro Scocco

Prof: Daniel Batista

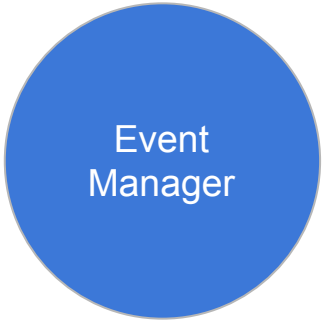
Estrutura do Simulador



Simulator

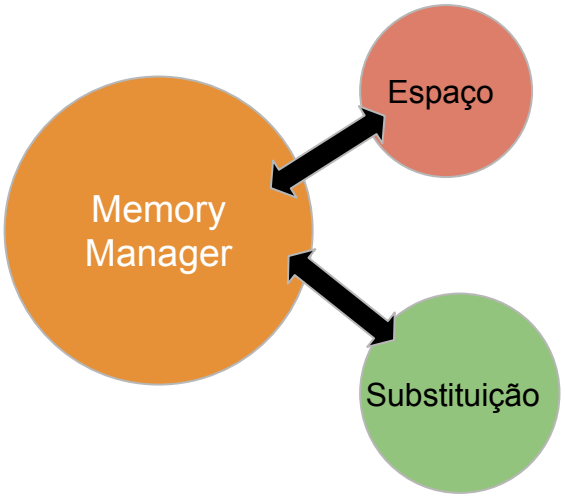
- Responsável por ler e interpretar os comandos do Shell
- Configura as classes de acordo com os parâmetros de entrada
- Inicia a simulação
- Permite repetir a simulação com novos parâmetros

Event Manager



- Interpreta o arquivo de Trace
- Cria eventos de Inicio de um Processo, Acesso à Memória e Fim de um Processo
- Cria também eventos de impressão com o tempo determinado pelo comando executa
- Dependendo do Algoritmo de Paginação cria eventos de Reset do Bit R
- Os Eventos são armazenados em ordem cronológica e enviados para o MemoryManager no momento certo para serem tratados

Memory Manager



- Implementa um método para cada tipo de evento
- A simulação ocorre basicamente nos eventos de chegada, acesso e finalização
- As classes de Espaço e Substituição são instanciadas de acordo com os algoritmos escolhidos, utilizando o Strategy pattern
- Os algoritmos retornam apenas a base para escrita nos arquivos (Offset)
- O memory manager realiza todas as leituras e escritas nos arquivos de memória

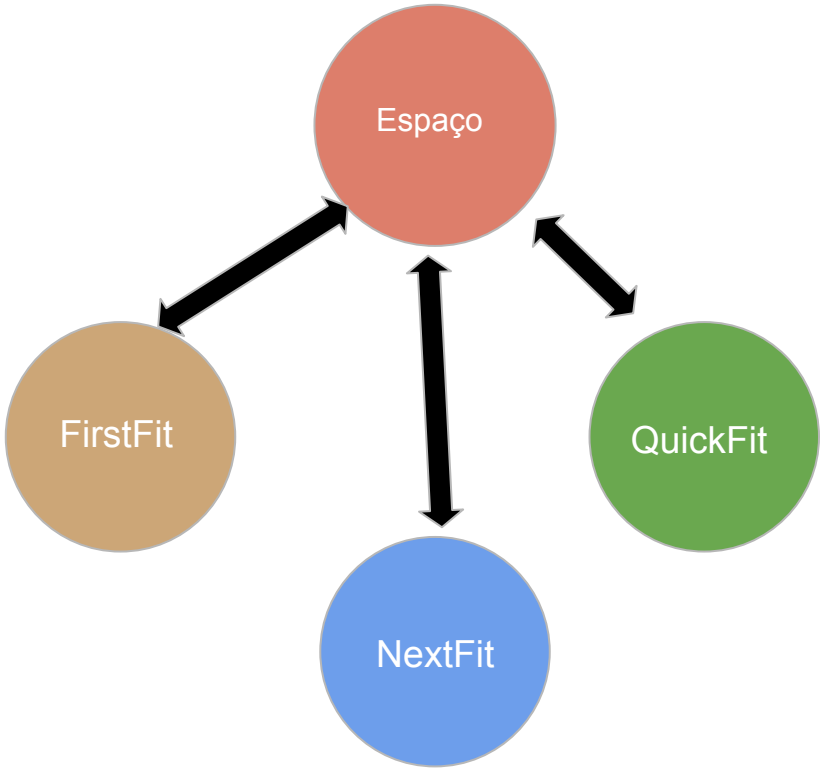
Memory Manager

Page Table entry (32 bits)

[unused_bits][M][R][P][page_frame_number]

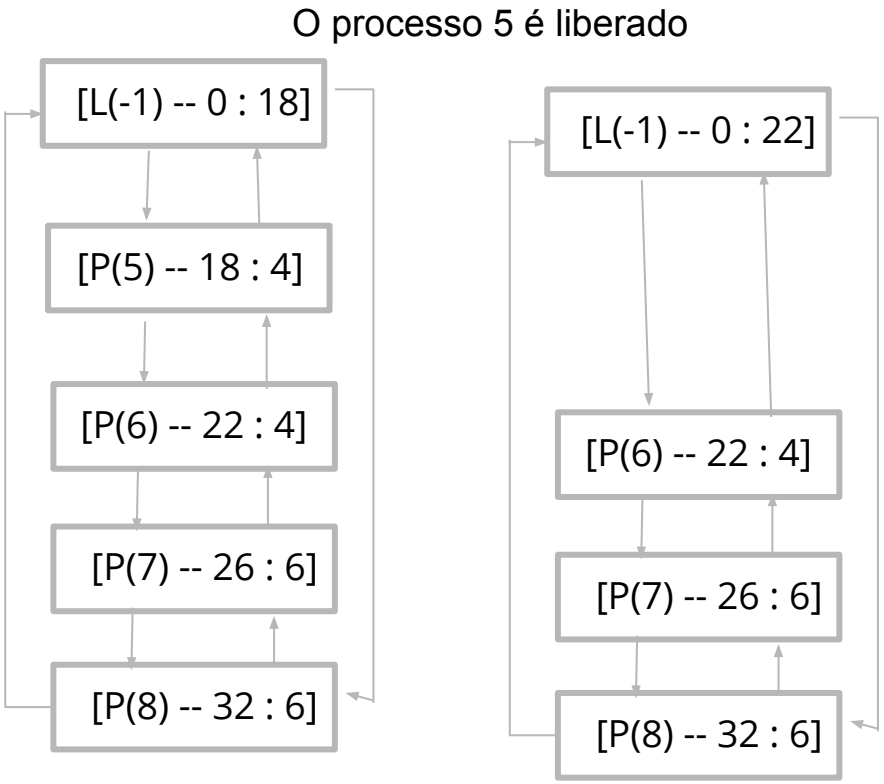
- Ele mantem uma tabela de processos, que guarda a base e o limite de cada processo
- A cada acesso da memória somamos o endereço local acessado com a base do processo
- Para descobrir o número da pagina fazemos 'virtual_addr << 4', onde 4 é log de 2 do tamanho da página, que é 16
- Usamos o número da página para acessar a tabela de páginas recuperando um inteiro de 32 bits
- Extraimos o frame, bit_p e bit_r desse inteiro através de máscaras binárias

Algoritmos de Espaço



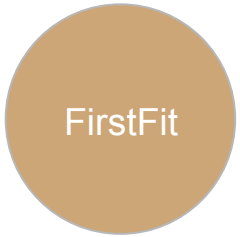
- Existe uma mesma interface nos três algoritmos :
  - find_free_space(process) : Integer offser
  - free_process(process) : Void
- Todos os algoritmos descendem da classe SpaceBase, a qual implementa certos métodos que são comuns a todos, como a inicialização.
- Certos métodos são sobrescritos pelas subclasses como 'release_process (pid)'
- Uma unidade de alocação consiste em 16 bytes, o tamanho de uma página

Lista ligada de espaço livre



- O algoritmo de espaço mantem uma representação abstrata do estado da memória virtual através de uma lista ligada.
- Quando um processo deixa de ser executado e tem uma posição contigua livre ocorre uma compactação do espaço livre

Algoritmo FirstFit



- O mais simples dos 3
- A lista ligada é percorrida desde o começo a cada solicitação de espaço

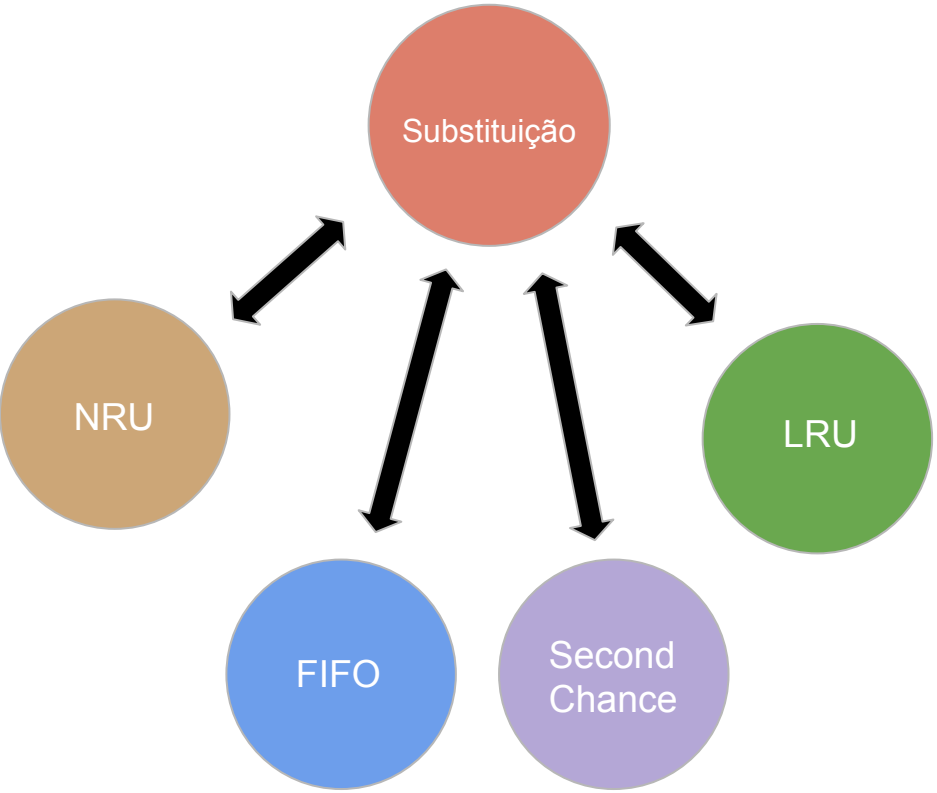
Algoritmo NextFit

- É guardado um ponteiro para a ultima posição alocada
- A cada solicitação de espaço a busca começa a partir do ponteiro para a ultima alocação

Algoritmo QuickFit

- Algoritmo com implementação mais complexa
- Um Vetor de listas ligadas guarda os espaços livres separados por seus tamanhos
- A cada solicitação de espaço percorremos a lista ligada que contem os menores espaços que comportam o processo

Algoritmos de Substituição



- Existe uma mesma interface nos três algoritmos :
  - find_free_space(process) : Integer offser
  - free_process(process) : Void
- Todos os algoritmos descendem da classe SpaceBase, a qual implementa certos métodos que são comuns a todos, como a inicialização.
- Certos métodos são sobrescritos pelas subclasses como 'release_process (pid)'
- Uma unidade de alocação consiste em 16 bytes, o tamanho de uma página

Algoritmo NRU

- Algoritmo sempre percorre a lista de paginas referenciadas inteira
- Caso encontre um quadro vaziu encerra o algoritmo e o retorna
- Separa as paginas em duas classes (Já que o bit M nunca é modificado)
- Tenta escolhe um quadro aleatóriamente primeiro da classe zero, se estiver vazia escolhe da classe um
- Necessita que o bit R seja resetado de tempo em tempo

Algoritmo FIFO

- Mantem uma lista ligada de quadros ocupados
- Mantem um contador de quadros livres para saber se deve procurar por um no vetor de quadros
- A cada page fault o primeiro elemento da lista é removido
- A cada nova pagina inserida em um quadro, ele é colocada no fim da fila

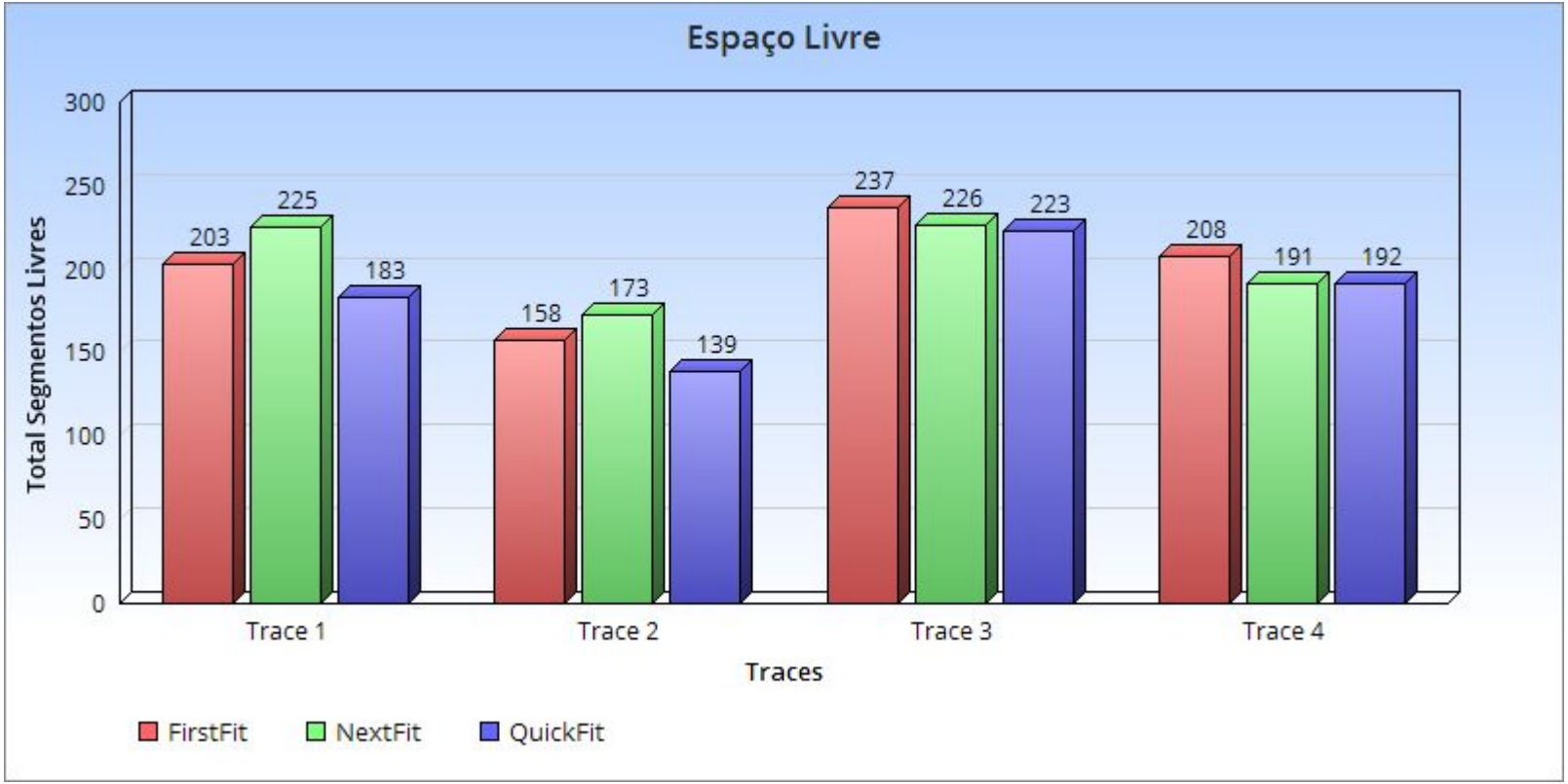
Algoritmo Second Chance

- Mantem uma lista ligada de quadros ocupados
- Mantem um contador de quadros livres para saber se deve procurar por um no vetor de quadros
- A cada page fault o bit R do primeiro elemento da lista é verificado.
- Se for 1 ele é inserido no final e o bit é zerado. Caso contrário ele é escolhido para substituição
- A cada nova pagina inserida em um quadro, ele é colocada no fim da fila

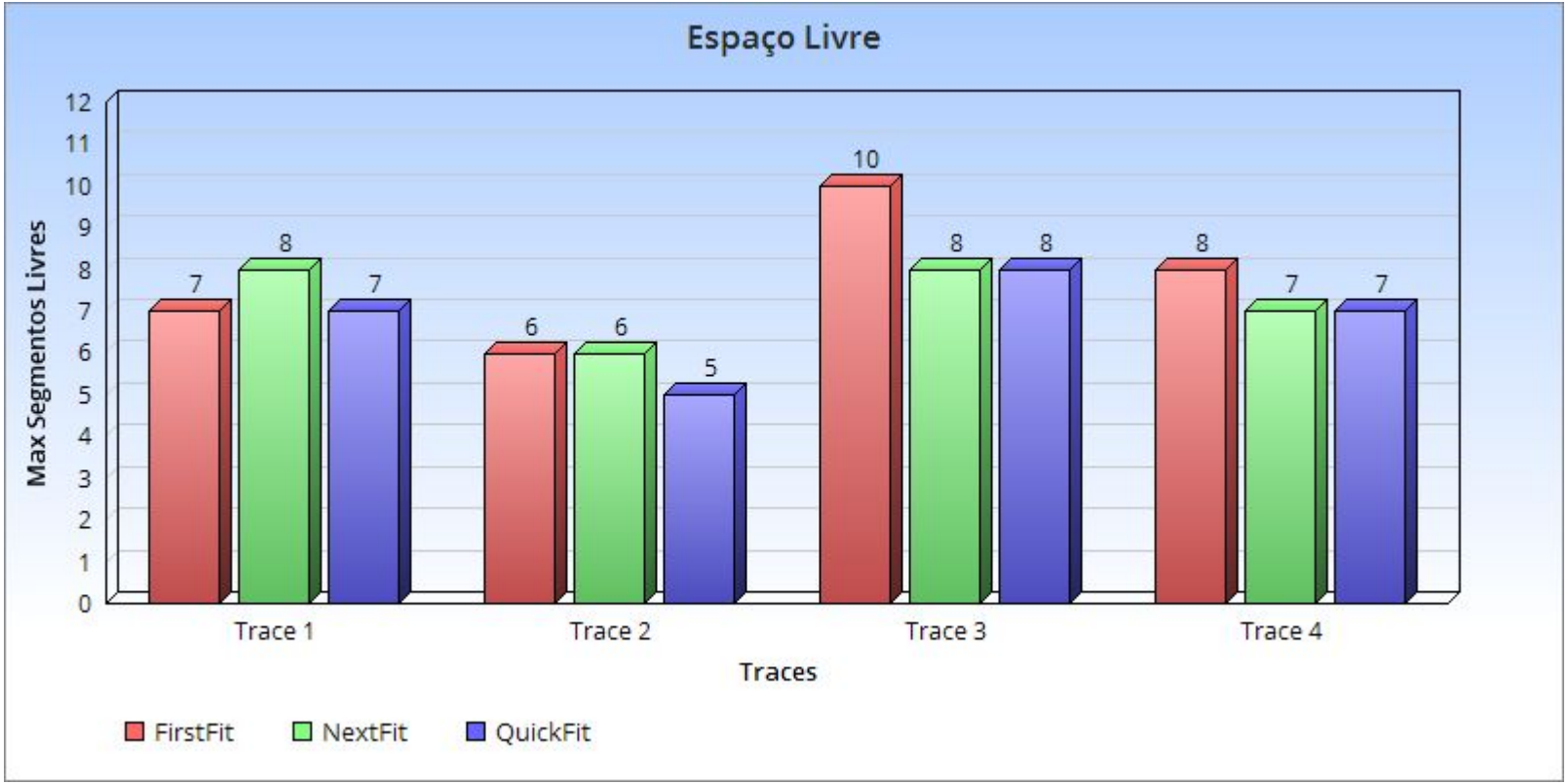
Algoritmo LRU

- Mantem um contador de quadros livres para saber se deve procurar por um no vetor de quadros
- É criada uma matriz de NxN onde N é a quantidade de quadros na memória física
- A cada acesso à memória, a linha com o indice do quadro acessado é preenchida com 1's e em seguida a coluna com 0's
- A cada paga fault procura-se a linha com o menor valor binário na matriz
- O quadro com o mesmo indice da linha é substituido

Comparação Algoritmos de espaço



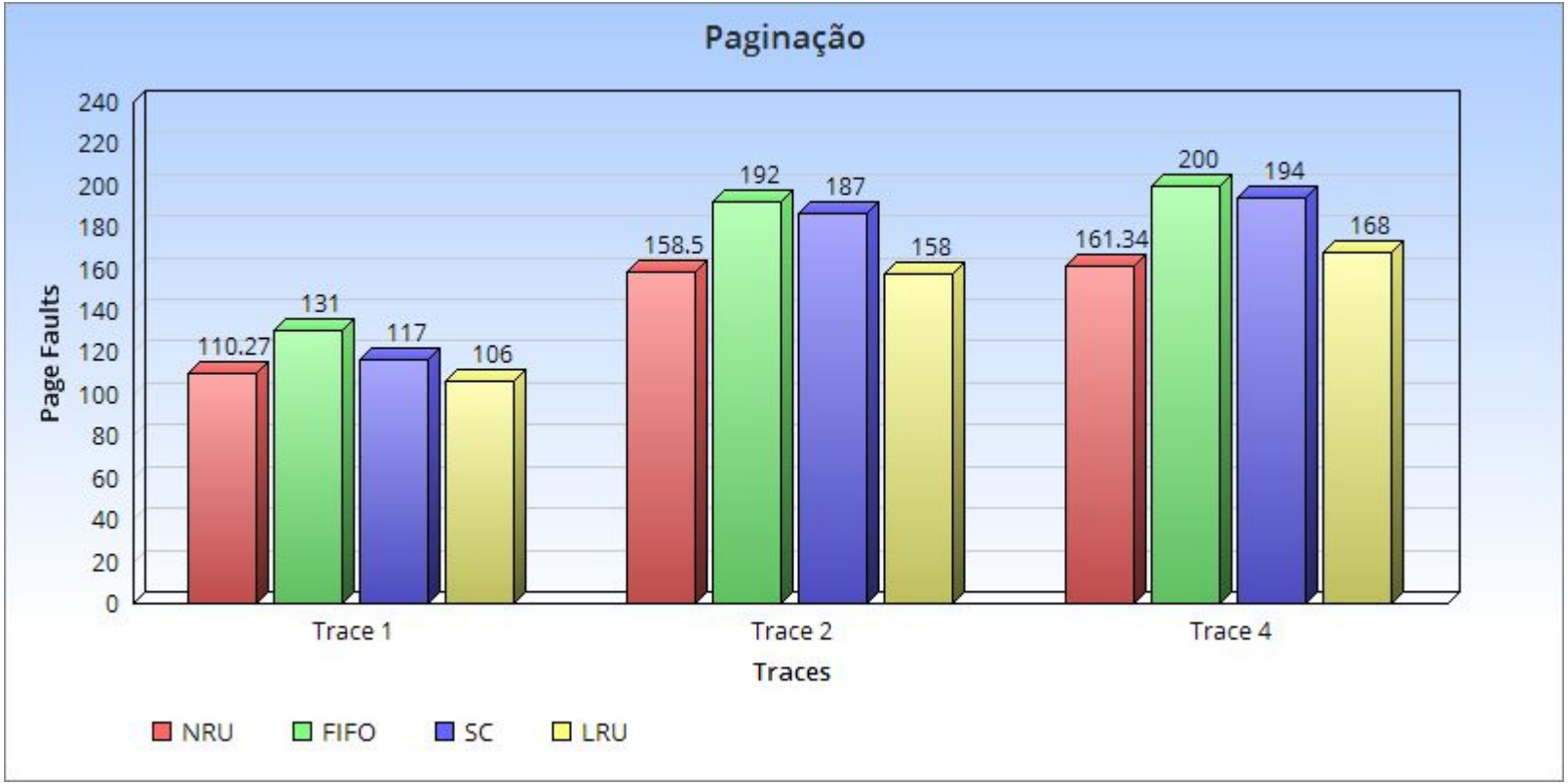
Comparação Algoritmos de espaço



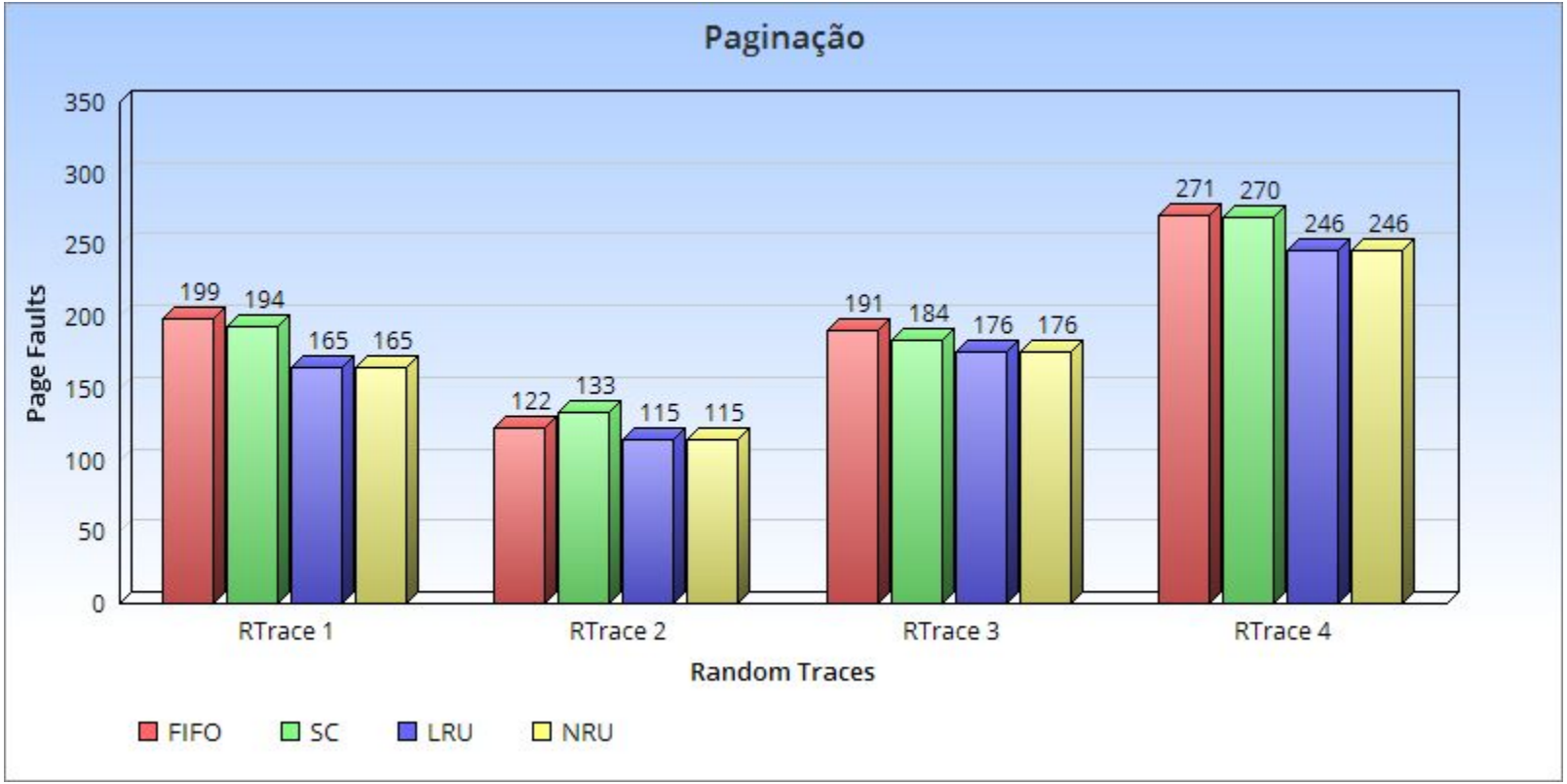
Observações

- Total é a soma da quantidade de nós livres na lista ligada de espaço, após cada alocação de um processo novo.
- Max é o máximo de nós livres que a lista ligada de espaços chegou a ter durante a execução
- Arquivos de Trace gerados aleatoriamente com algumas restrições:
  - 8000 bytes de memória virtual
  - 50 processos com começo e fim aleatórios dentro do intervalo de 20 segundos
  - Cada processo tinha de 16 a 240 bytes de tamanho aleatoriamente

Comparação Algoritmos de Paginação



Comparação Algoritmos de Paginação



Observações

- Arquivos de Trace gerados aleatoriamente com algumas restrições:
  - 160 bytes de memória física
  - 20 processos com começo e fim aleatórios dentro do intervalo de 20 segundos
  - Cada processo realizava 3 acessos à memória a cada segundo enquanto estava ativo
  - As páginas dos processos tinham pesos que determinavam a chance de ela ser acessada
- Os arquivos Trace Random eram iguais, porém sem os pesos para as páginas

Conclusões - Espaço

- Como era esperado o algoritmo QuickFit teve o melhor desempenho, porém seu consumo de memória e sua complexidade são piores comparados aos outros
- O NextFit, com alguns arquivos de Trace, teve desempenho melhor que o FirstFit, o que não era esperado. Suspeitamos que seja devido ao arquivo de trace ter grande aleatoriedade.

Conclusões - Substituição

- Para nossa surpresa o algoritmo NRU teve o melhor desempenho dentre os demais
- Testamos alguns valores para o intervalo de resetar o bit R no algoritmo NRU, e encontramos que com os nosso arquivos de trace, valores maiores melhoravam a performance
- O Algoritmo LRU implementado com matriz ficou em segundo lugar, porém muitas vezes bem próximo ao NRU

OBRIGADO