Swap Space

- Swap space - espaço reservado no disco para trocas de páginas entre a memória e o disco
  - Quando combinado com o espaço de memória física temos o nº de páginas naveis no sistema
- Uma entrada PTE deve ter um present bit que indica se a página está em memória ou no disco. O ato de acessar uma página que não está em memória chama-se page fault e o OS Deve invocar um page fault handler que deve:
  - encontrar um frame livre em memória
  - ler a página do disco (I/O request) e colocá-la em memória → atualiza o PTE
  - voltar a tentar aceder à página em memória → pode ser um TLB miss que deve ser tratado
- Quando a página está a ser lida do disco o processo state está bloqueado e o OS pode correr outro programa → otimiza o uso de hardware (overlapping I/O and CPU time)

Swap daemon

- Swap daemon - thread que corre no background e liberta memoria quando a quantidade de frames disponiveis está abaixo de um determinado numero (low watermark - LK), liberta tantos páginas quantos necessarios para atingir o highwatermark HW

- Ter um swap deamon altera o comportamento do page fault handler e permite duas coisas:
  - o page fault handler deixa de ser responsável por substituir as páginas ~~uma~~ apenas informa o daemon se não houver frames livres e aguarda por uma notificação dele
  - o daemon pode agrupar varias páginas e remove-las (flush) num unico pedido

- Estes mecanismos são feitos de forma transparente, Cada processo tem a sua memoria privada e não se preocupa com a página na memoria ou no disco - o OS trata de tudo - abstração da memoria virtual

Page Replacement Policies

- Com a memória virtual, a memória física pode ser vista como uma cache de páginas no sistema (os restantes estão em disco)
- O objetivo é minimizar o número de cache misses, maximizar o nº de cache hits

- Average Memory Access Time (AMAT) = $T_M + (P_{miss} \times T_D)$
  - T_M (custo de aceder à memória)
  - T_D (custo de aceder a disco)
  - P_{miss} (probabilidade de não encontrar os dados na memória)

- Optimal Replacement Policy - substitui a página que será usada mais longe no futuro, isto é, aquela que não será usada por mais tempo. Resulta no mínimo possível de cache misses mas não é aplicada na prática porque não conhecemos o futuro

- FIFO (first in, first out) - remove a página mais antiga em memória independentemente do seu uso; é fácil de usar mas pode remover páginas ativamente usadas só porque estão no fim da queue

- Random - remove uma página escolhida aleatoriamente; é simples e evita padrões previsíveis, mas como depende da sorte tem um desempenho instável e não leva em consideração a frequência ou a recência de uso

Algoritmo do Relógio

- Tanto LRU como LFU exigem demasiado do OS então sistemas modernos aproximam o comportamento do LRU. Um bit de referência na entrada da tabela de páginas (PTE) é definido pelo hardware como 1 quando a página correspondente é acedida
  - Imagine-se todas as páginas do sistema organizadas numa lista circular e um ponteiro que aponta para uma das páginas. Quando uma página tem de ser substituída, o OS olha para essa página e
    - se o bit de referência for 1 → página usada recentemente → define o bit como 0 e avança o ponteiro do relógio para a próxima página
    - se o bit de referência for 0 → a página é substituída

- LRU (Least Recently Used) - remove a página que não é usada há mais tempo é uma boa aproximação do algoritmo ótimo, mas é caro em termos de implementação

- LFU (Least Frequently Used) - remove a página que foi usada com menos frequência, favorece as páginas mais frequentemente usadas mesmo que não tenham sido usadas recentemente. Pode manter páginas antigas que foram muito usadas mas já não são

Algoritmo do relogio (melhorado)

1. Primeiro tenta substituir as páginas que não foram acedidas recentemente nem modificadas
2. Depois, escolhe as páginas que não foram acedidas recentemente mas que estão sujas (foram modificadas)

- Uma melhoria é considerar se o conteúdo da página foi modificado depois de ter sido trazida (swapped) para a memória: adiciona um dirty bit que é definido para 1 se a página ~~foi trocada~~ foi modificada
- Algumas páginas são trocadas (swapped) para dentro e para fora da memória várias vezes. Se houver espaço suficiente na swap area, o OS pode deixar o conteúdo das páginas lá e assim:
  - substituir uma página não modificada não requer uma operação I/O extra porque o conteúdo mais recente da página já está na swap area
  - Substituir uma página modificada requer escrever o conteúdo de volta para disco (atualizar o conteúdo da página na área de swap).

Thrashing

- Thrashing acontece quando o OS passa mais tempo a tratar de page faults e a trocar processos do que a executar processos
  - Acontece quando existe pouca memória disponível para os processos; os processos acessam muitas páginas num curto intervalo de tempo ou a memória não consegue manter as páginas ativas o que gera um ciclo de page faults
  - À medida que o sistema está a sofrer thrashing a utilização da CPU diminui porque mais I/O está a ser feito → o escalonador observa a baixa utilização do CPU e decide executar mais processos o que acentua o thrashing
  - Soluções: ajustar o grau de multiplicidade, gerenciar o conjunto de trabalho, políticas de substituição mais eficientes, aumentar a memória física (RAM)

Hard Disk Drivers (HDDs)

- HDDs são dispositivos de armazenamento magnético usado para guardar dados. São compostos por setores que podem ser lidos ou escritos, para um disco com n setores estes vão de 0 até n-1.
- Os dados são armazenados num platter que tem duas faces chamadas de superfícies. Um disco pode ter um ou mais platters presos num eixo (spindle) que é ligado a um motor e os roda a uma velocidade fixa
- Um braço mecânico com uma cabeça de leitura/gravação que se move sobre os pratos
- O acesso a dados envolve tempo de busca (mover a cabeça até a trilha certa) e latência rotacional (esperar que o setor pare sobre a cabeça

Nearest block first

Shortest Seek time first

- Nearest block first - atende os pedidos por ordem de chegada
  - simples mas ineficiente se os pedidos estiverem muito distantes entre si
  - pode gerar grandes deslocamentos

- Shortest Seek time first - atende o pedido mais proximo da posição atual da cabeça
  - reduz o tempo médio de busca, mas pode causar starvation (pedidos distantes podem esperar muito tempo)

SCAN

C-SCAN

- **SCAN** - a cabeça do disco move-se numa direção e atende os pedidos que aparecem até ao fim do disco. Se surgir um pedido num setor que já foi passado, este é colocado em fila de espera. Ao chegar ao fim a cabeça inverte o sentido
  - mais justo que SSTF mas pode demorar a atender pedidos que estão na direção oposta

- **C-SCAN** - varre as pistas do disco em apenas uma direção. Quando chega ao fim, a cabeça volta ao início (sem atender pedidos) e recomeça
  - mais uniforme (latência previsível)
  - um pouco mais lento que o SCAN

Shortest Positioning time first (SPTF)

- Shortest Positioning Time First - considera dois fatores para decidir qual pedido atender: - distância (seek time)
  - latência rotacional (quanto tempo vai demorar até o setor desejado girar até à cabeça

É escolhido o pedido com menor tempo total de posicionamento não só de distância linear

- Pode ser mais eficiente que SSTF
- Complexo de implementar

Redundant Array of Inexpensive Disks (RAID)

RAID - tecnologia que combina multiplos discos fisicos numa unica unidade para aumentar a confiabilidade e melhorar o desempenho
Pode ser implementado de duas formas: Hardware RAID através de um controlador dedicado ou Software RAID gerido pelo OS

- RAID level 0 - blocos distribuidos por vários discos, blocos na mesma linha formam um stripe
  - a capacidade total é a soma dos 4 discos
  - acesso mais rapido a dados contiguos devido ao paralelismo entre discos
  - dificuldade em escolher o stripe size (blocos pequenos → + paralelismo → + custos em HDDs)
  - se um disco falha todos os dados são perdidos, não oferece tolerância a falhas.

- RAID level 1 - cada bloco é copiado para dois ou mais discos
  - se um disco falha os dados são recuperados a partir do disco espelho
  - possibilidade de paralelismo na leitura e balanceamento de carga entre discos espelhados
  - capacidade reduzida
  - escritos duplicados (se for feita em paralelismo a escrita é limitada pelo disco + lento
- RAID level 4 - adiciona um bloco de paridade que guarda um bloco de paridade (XOR) por cada stripe
  - suporta a falha de um disco. A paridade permite reconstruir os dados perdidos
  - leituras podem ser feitas em paralelo exceto no disco de paridade
  - menor overhead de armazenamento, apenas é necessário um disco adicional para a paridade
  - atualizações lentas (atualizar um bloco implica atualizar o de paridade
  - O disco de paridade torna-se um ponto de congestão.

- RAID level 5 - os blocos de paridade são distribuídos por todos os discos do RAID
  - suporta a falha de 1 disco
  - melhor desempenho em leitura (podem ser feitas em paralelo em todos os discos uma vez que os blocos de paridade são espalhados
  - menos gargalo na paridade (como os blocos de paridade estão espalhados evita sobrecarga num único disco
  - recalculo a paridade ainda tem um custo
  - mais complexo de implementar

Translation Look-Aside Buffer (TLB)

- TLB (Translation Look-Aside Buffer) - cache de hardware que guarda traduções de endereços popular. É mais rápido de acessar do que a memória principal.
  - Quando uma tradução de virtual-adress é pedida:
  1. Extraino VPN do virtual adress
  2. Verifica se o VPN está na TLB
     - TBL hit: a tradução está no TLB (obtem-se o PFN, junta-se com o offset e temos o endereço físico sem ter de aceder à tabela de páginas o que poupa um acesso à memória)
     - TLB miss: a tradução não está no TLB: devemos consultar a tabela de páginas na memória) encontrar a tradução e atualizar o TLB com uma entrada e reexecutar a instrução.

- O escalonador de processos tenta manter um equilíbrio entre processos intensivos do CPU e de I/O porque enquanto um processo está à espera do disco, por ex, o processador pode ser usado por outro processo que precise dele. Assim o sistema consegue manter o processador e o disco ocupados, o que evita que eles fiquem parados ou sobrecarregados. Isto maximiza a utilização dos recursos do sistema, melhora o desempenho real e evita starvation.
- reduz o tempo que o processador fica parado à espera
- melhora o throughput
- melhora a responsividade do sistema
- Evita sobrecarga.

Base and Bounds

- Base and Bounds vai permitir colocar o adress space em qualquer lugar na memória física e ao mesmo tempo assegurar que o processo apenas pode aceder ao seu próprio space
  - Cada programa é escrito e compilado como se fosse carregado no endereço 0. O SO decide onde carrega o endereço na memória física define o registo base como o endereço físico onde o programa foi colocado
  - O registo bounds ajuda a proteger os limites do espaço de memória de cada processo. Se um acesso à memória não estiver dentro dos limites é gerada uma exceção out-of-bounds

- Physical Adress = Virtual adress + base

Segmentação

- Segmentação - técnica implementada que usa vários registos base and bound, um por cada segmento lógico.
  - permite que o OS coloque cada segmento em diferentes partes da memória física
  - é possível partilhar segmentos. Se dois registos base A e B apontarem para o mesmo endereço físico temos um segmento partilhado
  - bits de proteção indicam o que cada processo pode fazer com cada segmento: ler, escrever ou executar
  - uma free list contem os endereços e o tamanho dos chunck livres e memoria física
  - Existem vários protocolos para quando é feito um pedido
  - Pode sofrer de fragmentação externa e interna

- Fragmentação externa - quando ao colocar segmentos na memória física o SO a deixa cheia de pequenos buracos

- Fragmentação interna - se um alocador entrega chunks de memória maiores do que o requisitado o que estiver a mais é considerado fragmentação interna

- Splitting - quando um bloco de memória é maior do que o requisitado, o SO pode dividi-lo fica com um bloco do tamanho pedido e outro com o restante que se mantem livre → ajuda a diminuir a fragmentação interna

- Coalescing - quando um bloco de memória é liberado, o sistema tenta fundi-lo com blocos livres adjacentes para formar blocos livres maiores, reduzindo a fragmentação externa e melhorando a eficiência de futuras alocações

- Embedding - em vez de manter uma estrutura separada com apontadores para blocos livres, os próprios blocos contêm os ponteiros da lista isto evita alocar memória extra só para a lista livre; fácil de percorrer e gerir os blocos

First fit

Next fit

- First fit - devolve a memoria requisitada do primeiro bloco grande o suficiente que encontrar, o restante do bloco mantem-se na free list
  - é rapido mas polui o inicio da free list com blocos pequenos (fragmentação)
  - se usar mos order-based ordering - manter a lista ordenada pelos endereços de espaço livre o coalescing torna-se fácil e a fragmentação tende a ser reduzida

- Next fit - mantem um extra pointer do ultimo ponto onde parou e continua daí a ideia é espalhar a pesquisa pela lista o mais uniformemente possivel para evitar splinterings
  - equilibrado em grandes listas, mas pode ignorar bons blocos no inicio da lista

Best fit

Worst fit

- Best fit - procura pela free list os chunks de memoria livres que são grandes ou maiores que o pedido e devolve o menor desses
  - Reduz a fragmentação interna, mas é lento e pode deixar fragmentos muito pequenos que são difíceis de utilizar depois

- Worst fit - procura o maior chunk e retorna a memoria requisitada. A ideia é que o restante do chunk seja util pra ainda manter-se na free list
  - é bastante lento, tem má performance e pode causar fragmentação interna

CPU scheduling

- Turnaround time - tempo que um processo demora a ser completo depois de chegar ao sistema
  - Turnaroundtime = Tcompletion - Tarrival

- Convoy Effect - um processo que demora pouco tempo fica à espera de um processo que demora muito tempo.

- Response time - tempo desde a submissão de um processo até ao primeiro momento em que o processo é executado
  - Tresponse = Tfirstrun - Tarrival

First Come, First Served (FCFS)

- First Come, First Served - corre o processo que chega primeiro
  - sofre de convoy effect - um processo que demora pouco tempo fica à espera de um que demora muito tempo
  - Turnaround time alto/medio
  - Response time alto para os ultimos processos
  - não aceita interrupção forçada

Shortest Job First (SJF)

- Shortest Job First - corre o processo que demora menos tempo e depois o mais pequeno a seguir a esse e vice-versa
  - injusto com processos longos
  - Turnaround time é bom em média
  - Response time alto para jobs longos
  - não aceita interrupção forçada

Shortest time-to-completion First (STCF)

- Shortest-time-to-completion-first - Cada vez que um novo processo entra no sistema, para tudo, e corre o processo com runtime mais pequeno no momento
  - Turnaround time é ótimo
  - Response time é muito bom para jobs curtos
  - aceita interrupção forçada

Round

- Round-Robing - corre o processo por um determinado período de tempo (scheduling quantum) e depois troca para o próximo processo na queue
  - Um pequeno scheduling quantum pode ser problemático. O custo de um context switching pode dominar a performance. A duração do scheduling quantum deve armotizar o custo de context switching
  - Turnaround time médio
  - Response time baixo (resposta rápida)
  - aceita interrupção forçada

Multi Level FeedBack Queue

- Multi-Level Feedback Queue - tem um número de queues distintos cada uma com diferentes graus de prioridade. MLFQ usa a prioridade para saber que job correr a seguir. Se existirem vários jobs com a mesma prioridade:
  1. If Priority(P1) > Priority(P2), P1 runs P2 não
  2. If Priority(P1) = Priority(P3), P1 e P3 correm em Round Robin
  3. Quando um processo entra no sistema é colocado na queue com maior prioridade
  4. Quando um processo usa o seu allotment a sua prioridade é reduzida
  5. Após um determinado período de tempo todos os jobs são movidos para a queue + prioridade
- job allotment - quantidade de tempo que um job pode passar num determinado nível de prioridade
  - Turnaround time balanceado
  - Response time muito bom para jobs interativos
  - aceita interrupção forçada