L04f. Shared Memory Multiprocessor OS

Scan me

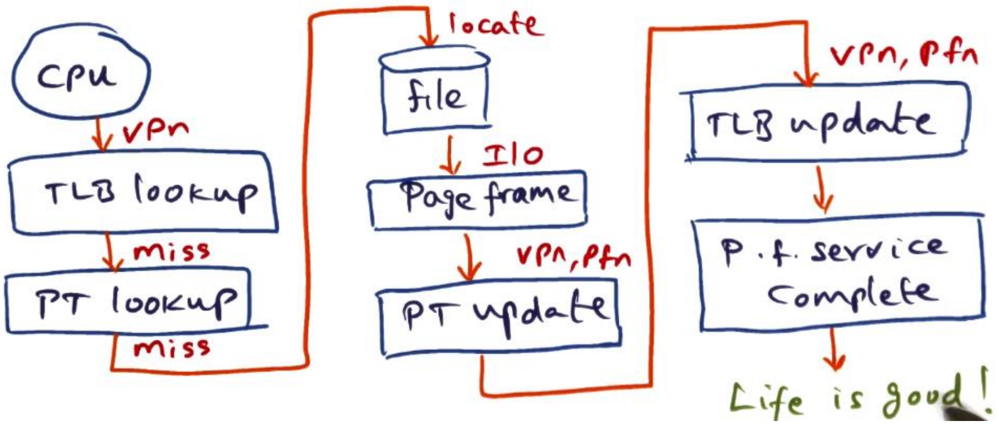
Challenges of Parallel Systems:

- The big size of the system results in bottlenecks for the global data structures.
- The memory latency is huge due to faster processors and more complex controllers.
- A non-uniform memory access (NUMA) architecture: Connecting all the nodes in the system through an Inter-connecting Network. The large distance between accessing processors and the target memory results in lower performance.
- Deep memory hierarchy (Multi-cache).
- False sharing: Sometimes the cache hierarchy (large cache lines) makes the memory addresses touched by different threads on different cores to be on the same cache block. This gives the illusion that these addresses are shared (without programmatic sharing). This particularly happens on modern processors because they tend to have larger cache blocks.

OS Design Rules:

- Cache decisions:
  - Pay attention to locality.
  - Exploit affinity of caches when taking scheduling decisions.
  - Limit the amount of sharing of data structures to reduce contention.
- Limit the amount of sharing of system data structures to reduce contention.
- Keep memory accesses as local as possible.

Page Fault Service:



- A page fault service consists of:
  - TLB and Page Table lookup: This is thread specific and can be done in parallel.

- Locating the data on desk and moving it to the page frame, then updating the Page Table: This is a bottleneck because these are OS functions and have to be done in series.
- TLB update: This is processor specific and can be done in parallel.

- There're two scenarios in parallel OSs:
  - Multi-process workload: If each thread is running independently on a specific CPU, we'll have distinct page tables and hence no serialization.
  - Multi-threaded workload: If the address space is shared between different threads, then the page tables will be shared as well. In this scenario, the OS should ensure that no or minimum serialization happens.

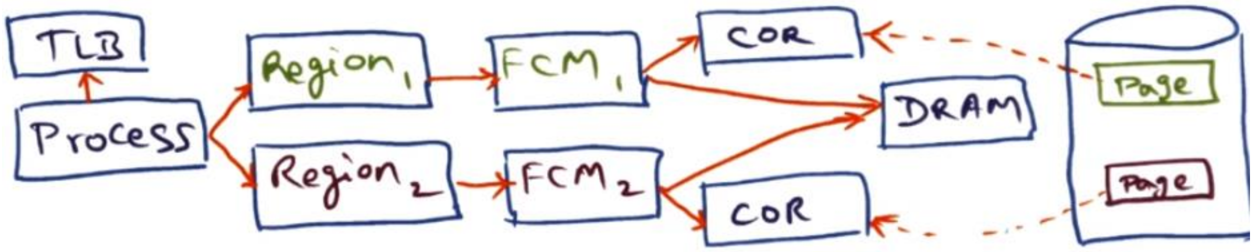
Designing Scalable Structures in Parallel OS:

- Determine what needs to be done for each service.
- To ensure concurrent executions of services, minimize shared data structures.
- Replicate/Partition data structures that have to be shared to reduce locking.

Tornado OS:

- Tornado uses an object-oriented approach, where every virtual and physical resource in the system is represented by an independent object. This ensures locality and independence for all resources.
- Trusted Object: Tornado uses a single object reference for all the OS parts.
- This single object reference, however, is translated into different physical representations.
- Degree of clustering (replicating a specific object):
  - Implementer choice.
    1. Single representation.
    2. One representation per core.
    3. One representation per CPU.
    4. One representation per group of CPUs.
  - The consistency of these representations is maintained through Protected Procedure Calls.
- Objectization of Memory Management:
  - The Address Space will be represented by the "Process Object", which will be shared by all the threads executing on the CPU.
  - The Address space will be broken into regions:
    → Each region will be backed by a "File Cache Manager – FCM" on the File System.
  - Another DRAM object will represent the Page Frame Manager, which is responsible for serving page frames for threads.
  - Another object called "Cached Object Representation – COR" will handle page I/O.

- Whenever a thread incurs a page fault:
  1. The Process Object will decide which region this page fault fall into, given the virtual page number.
  2. The Region Object will contact the File Cache Manager.
  3. The FCM will contact the DRAM Object to get the physical page frame.
  4. The FCM will pass the file and offset to COR, which will pull the data from the desk into the DRAM's page frame.
  5. FCM indicates to the Region Object that the physical page frame has been populated.
  6. The region Object will go through the Process Object to update the TLB.



- Objectization decisions:
  1. The Process Object can be one per CPU, since the TLB is one per CPU.
  2. The Region Object should be one per group of CPUs.
  3. The FCM should be one per group of CPUs.
  4. COR should have a single representation.
  5. DRAM Object can be one per physical memory.
- Advantage of clustered object: Same object reference on all nodes. We can have different replications of the same object, which decreases data structures locking.

• Implementation of Clustered Object:
- Each CPU has:
  → Translation Table: Maps an object reference to a representation in memory.
  → Miss Handling Table: If the object reference is not present in the Translation Table so far, the Miss Handling Table maps the object reference to Object Miss Handler that decides if this object reference should point to an already existing representation or a new representation should be created. Then, it maps this object reference to its representation and installs the mapping in the Translation Table.
  → If the Object Miss Handler is not local, a Global Miss Handler will be used. Every node has a Global Miss Handler, and it knows the partitioning of the Miss Handling Table. If an object reference is presented to the Global Miss Handler, it will resolve the location of the required replica, installs it locally, and populates the Translation Table.
- Non-Hierarchical Locking:
  → Hierarchical Locking: Whenever a thread is trying to execute a page fault, it locks the Process Object, the Region Object, the FCM, and the COR. This approach kills concurrency.
  → One way to resolve this is to use an Existence Guarantee and a Reference Count on the Process Object. This allows for concurrent operations on different regions.

- Dynamic Memory Allocation:
  - → Tornado OS breaks up the Heap space into multiple portions, each of which will be located on the physical memory of a specific node. That allows for scalable implementation of DMA.
  - → This also prevents false sharing across nodes.
- Inter-Process Communication – IPC:
  - → IPC is realized through Protected Procedure Calls (PPCs).
  - → If the communication is on the same processor, no context switch happens.
  - → If the communication is between different processors, full context switch happens.

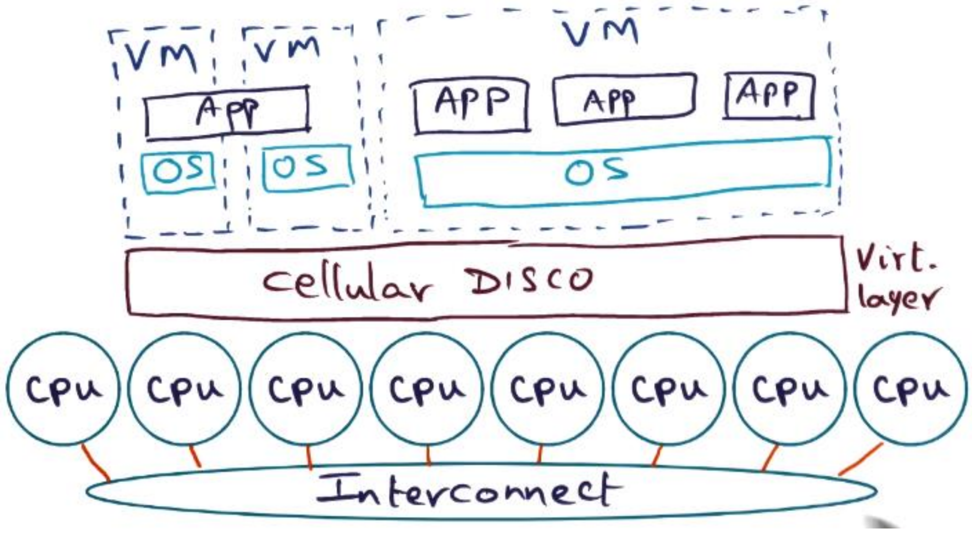
- Tornado OS summary:
  - Object oriented design for scalability.
  - Multiple implementations of OS objects.
  - Optimize for common case.
  - No hierarchical locking.
  - Limited sharing of OS data structures.

Corey OS:

- Address Ranges: Similar to Tornado's region concept. The difference is, instead of hiding the regions details from the application, on Corey, the address ranges are available to application so that it optimizes execution based on current thread accesses.
- Shares: A facility to be used by any process to communicate to the OS that the process will not share a specific data structure it's currently using.
- Dedicated cores for kernel activities.

Virtualization:

- Cellular Disco project explored the possibility of decreasing the virtualization overhead.
- The idea is to place a virtual layer between the guest OS and the I/O HW.


VM
VM
VM
App
APP
APP
APP
OS
OS
OS
cellular DISCO
Virt. layer
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
Interconnect