Beyond processor-centric operating systems

Summary:
Submitted by: Sharanya Kamath

In this paper, a vision is articulated for a memory-centric OS that moves traditional OS functionality like memory allocation, protection, synchronization and error handling out of the compute node and closer to the memory system. As data becomes the new currency, architectural changes are necessary to overcome the limitations of the traditional compute-centric model. Data-centric designs call for an architectural approach that minimizes data duplication and redundant motion, enables ubiquitous and heterogeneous computing resources, and deals with resilience and security from the ground up.

The paper states the following challenges to Processor-Centric OS:

- Shared pool of memory: Traditional memory management functionality is expected to move into memory side controllers, accelerators and more novel computational elements to form distributed cluster-wide services.
    - Memory resource management
    - Protection
    - Memory error handling

- Memory at large scale - The sheer size of the memory system poses challenges in addressing, coherency, synchronization, big memory operations, and error.
    - Addressing memory
    - Coherency
    - Synchronisation
    - Big memory operations
    - Memory error at scale

- Memory non-volatility - The non-volatile nature of the memory presents more challenges in abstraction, caches, error recovery, and encryption.
    - Abstraction for persistent data
    - Volatile caches
    - Recovery from errors
    - Encryption

- Additional issues - Memory-centric I/O, and memory-centric application runtimes, requiring a deeper study of distributed application runtimes that effectively use distributed persistent memory.
    - Memory centric I/O
    - Memory centric application runtime

Questions:

1. What can be done to run memory-centric OS in a distributed environment?
2. How is paging affected by processor-centric OS?