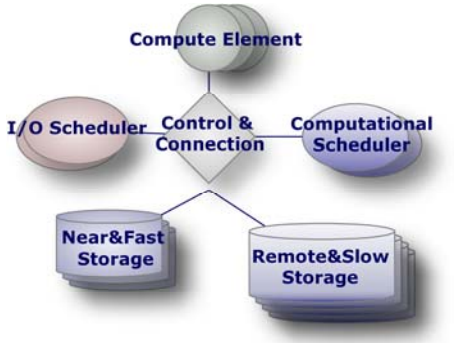
From Micro- to Macro-processing: A Generic Data Management Model

Mehmet Balman and Tevfik Kosar
{balman,kosar}@cct.lsu.edu
Center for Computation & Technology, and Computer Science Department
Louisiana State University



Generic Data Management Model

Data management has been one of the crucial problems in every stage of computer engineering. Accessing data in a transparent and efficient manner is also a major issue both in operating system design and in microprocessor architecture.

We first start with microprocessors, the basic component of computer systems, and study data management in processor architecture. Then, we study disk scheduling and I/O scheduling techniques in operating systems. We concentrate on mapping traditional operating system concepts to widely distributed environments in terms of scheduling and I/O management. We focus on problems on accessing and storing data that have been encountered, and also analyze how those issues have been resolved.

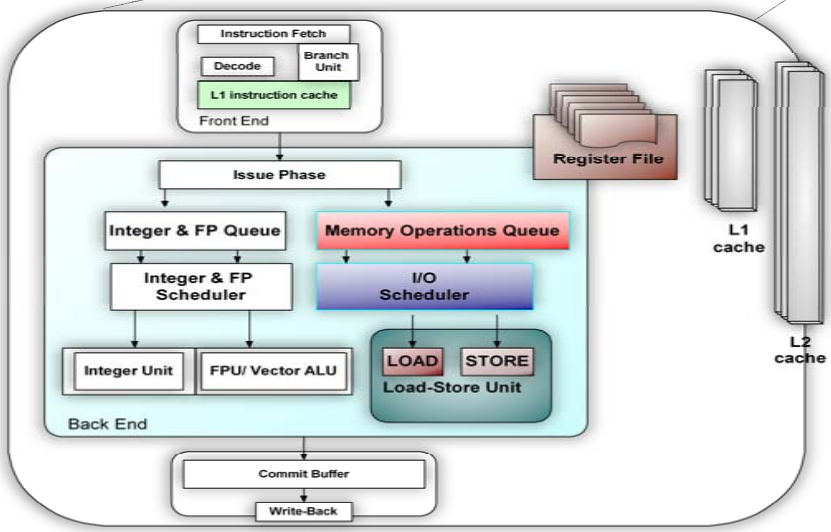
Since small and large scale systems have similar problems, we are planning to utilize approaches in microprocessor and operating system kernel architecture to come up with a broader perspective in which we can extend known methodologies to be used in distributed systems.

Scheduling and control of I/O requests:

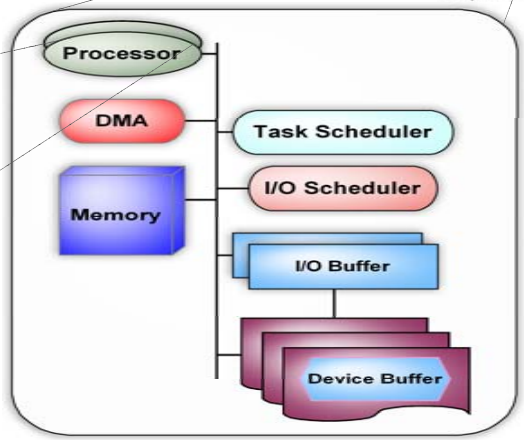
❖**Disk Request Scheduling** – seek time access time (FIFO, SSTF, SCAN, F-SCAN)
❖The I/O scheduler – a crucial component of the operating system

❖The Linus **Elevator** I/O Scheduler
❖The **Anticipatory** I/O scheduler

At some level the issues in processor and operating system I/O are similar to those in distributed systems, but there are also quite different issues at the storage layer, name spaces, data organization, provenance, etc.

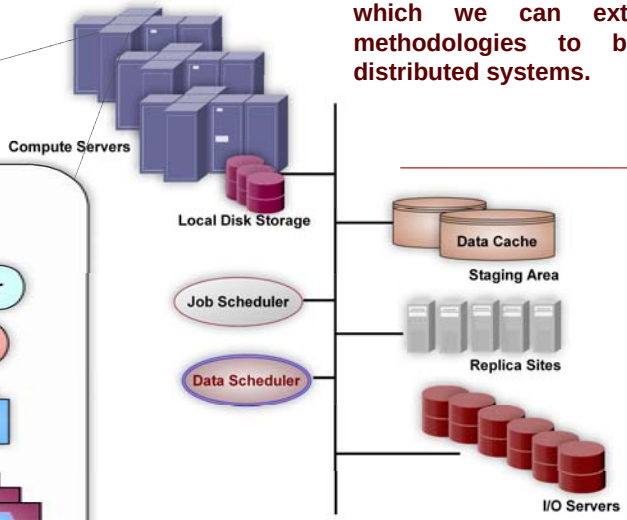


❖Memory access and arithmetic instruction are buffered and ordered such that their executions are overlapping to improve overall performance and throughput. Same technique is used in workflow management systems such that different kinds of jobs are processed at the same time while preserving the task sequence and dependencies.



❖ Most of the I/O accesses are repetitive; therefore, data can be fetched to a faster location beforehand. Caching – **staging data** - is one of the main improvements reducing the waiting time in microprocessors, also in operating systems.
❖ **I/O Buffering** – Staging Area- Operating system assigns a buffer in main memory for an I/O requests as a staging area.
❖ Multi-level cache systems **– hierarchically storage** technique in which data moves between high-cost and low-cost storage media.

❖ In microprocessors, load/store operations are dealt with different segments in the overall system. Separate **memory-access units** have been used to exploit the superscalar design and increase the performance.
❖**Direct Memory Access** (DMA) - data exchange between main memory and the I/O device is controlled by a separate module.



STORK: A Scheduler for Data Placement Activities

Data placement activities as **"first class citizens"** in the Grid just like the computational jobs.

- Support for heterogeneously
- Protocol translation using Stork memory buffer/Disk Cache
- Flexible Job Representation and Multilevel Policy Support
- Run-time adaptation
- Dynamic protocol selection, Run-time Protocol Auto-tuning
- Failure Recovery and Efficient Resource Utilization

PetaShare
www.petashare.org
An innovative distributed data archival, analysis and visualization instrument for data intensive collaborative research.

Distributed System Laboratory
www.dsl.csc.lsu.edu