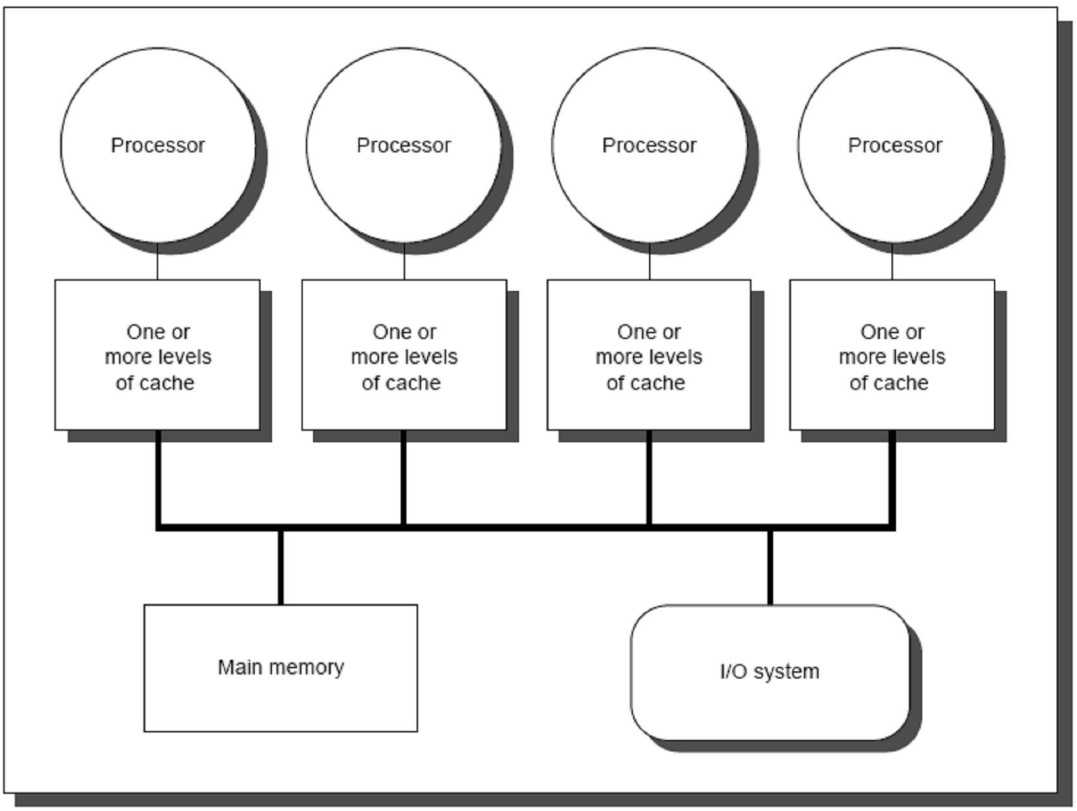
centralized shared-memory architectures

centralized shared-memory architectures , have at most a few dozen processors in 2000. For multiprocessors with small processor counts, it is possible for the processors to share a single centralized memory and to interconnect the processors and memory by a bus. With large caches, the bus and the single memory, possibly with multiple banks, can satisfy the memory demands of a small number of processors. By replacing a single bus with multiple buses, or even a switch, a centralized shared memory design can be scaled to a few dozen processors. Although scaling beyond that is technically conceivable, sharing a centralized memory, even organized as multiple banks, becomes less attractive as the number of processors sharing it increases. Because there is a single main memory that has a symmetric relationship to all processors and a uniform access time from any processor, these multiprocessors are often called *symmetric (shared-memory) multiprocessors* (*SMPs*), and this style of architecture is sometimes called *UMA* for *uniform memory access* . This type of centralized shared-memory architecture is currently by far the most popular organization. Figure below shows what these multiprocessors look like.



Basic structure of a centralized shared-memory multiprocessor.

Multiple processor-cache subsystems share the same physical memory, typically connected by a bus. In larger designs, multiple buses, or even a switch may be used, but the key architectural property: uniform access time o all memory from all processors remains.

Performance of Symmetric Shared-Memory Multiprocessors

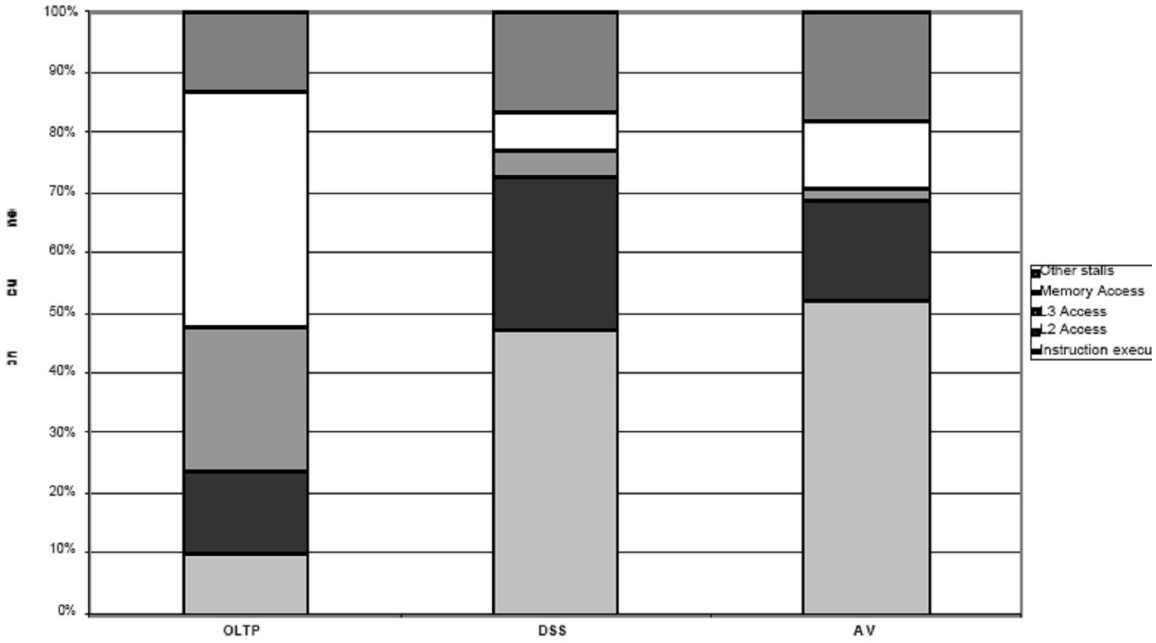
In a bus-based multiprocessor using an invalidation protocol, several different phenomena combine to determine performance. In particular, the overall cache performance is a combination of the behavior of uniprocessor cache miss traffic and the traffic caused by communication, which results in invalidations and subsequent cache misses. Changing the processor count, cache size, and block size can affect these two components of the miss rate in different ways, leading to overall system behavior that is a combination of the two effects. The misses that arise from interprocessor communication, which are often called coherence misses, can be broken into two separate sources.

The first source are the so-called true sharing misses that arise from the communication of data through the cache coherence mechanism. In an invalidation based protocol, the first write by a processor to a shared cache block causes an invalidation to establish ownership of that block. Additionally, when another processor attempts to read a modified word in that cache block, a miss occurs and the resultant block is transferred. Both these misses are classified as true sharing misses since they directly arise from the sharing of data among processors.

The second effect, called false sharing, arises from the use of an invalidation based coherence algorithm with a single valid bit per cache block. False sharing occurs when a block is invalidated (and a subsequent reference causes a miss) because some word in the block, other than the one being read, is written into. If the word written into is actually used by the processor that received the invalidate, then the reference was a true sharing reference and would have caused a miss independent of the block size or position of words. If, however, the word being written and the word read are different and the invalidation does not cause a new value to be communicated, but only causes an extra cache miss, then it is a false sharing miss. In a false sharing miss, the block is shared, but no word in the cache is actually shared, and the miss would not occur if the block size were a single word. The following Example makes the sharing patterns clear. True sharing and false sharing miss rates can be affected by a variety of changes in the cache architecture. Thus, we will find it useful to decompose not only the uniprocessor and multiprocessor miss rates, but also the true-sharing and false-sharing miss rates.

Performance Measurements of the Commercial Workload The performance measurements of the commercial workload, which we examine in this section, were taken either on a Alphaserver 4100, or using a configurable simulator modeled after the Alphaserver 4100.

The Alphaserver 4100 used for these measurements has four processors, each of which is an Alpha 21164 running at 300 MHz. Each processor has a three-level cache hierarchy:
* L1 consist of a pair of 8 KB direct-mapped on-chip caches, one for instruction and one for data. The block size is 32-bytes, and the data cache is write-through to L2, using a write buffer.
* L2 is a 96 KB on-chip unified 3-way set associative cache with a 32-byte block size, using write-back.
* L3 is an off-chip, combined, direct-mapped 2 MB caches with 64-byte blocks also using write-back.
The latency for an access to L2 is 7 cycles, to L3 it is 21 cycles, and to main memory it is 80 clock cycles (typical without contention). Cache to cache transfers, which occur on a miss to an exclusive block held in another cache, require 125 clock cycles. All the measurement shown in this section were collected by Barroso, Gharachorloo, and Bugnion We start by looking at the overall CPU execution for these benchmarks on the 4-processor system; as discussed on page 650, these benchmarks include substantial I/O time, which is ignored in the CPU time measurements. We group the six DSS queries as a single benchmark, reporting the average behavior. The effective CPI varies widely for these benchmarks, from a CPI of 1.3 for the Altavista web search to an average CPI of 1.6 for the DSS workload, to 7.0 for the OLTP workload. Figure shows how the execution time breaks down into instruction execution, cache and memory system access time, and other stalls (which are primarily pipeline resource stalls, but also include TLB and branch mispredict stalls). Although the performance of the DSS and Altavista workloads is reasonable, the performance of the OLTP workload is very poor, due to a poor performance of the memory hierarchy.



The execution time breakdown for the three programs (OLTP, DSS, and Altavista) in the commercial workload. The DSS numbers are the average across six different queries. The CPI varies widely from a low of 1.3 for Altavista, to 1.61 for the DSS queries, to 7.0 for Oracle. (Individually, the DSS queries show a CPI range of 1.3 to 1.9.) Other stalls includes: resource stalls (implemented with replay traps on the 21164), branch mispredict, memory barrier, and TLB misses. For these benchmarks resource-based pipeline stalls are the dominant factor.This data combines the behavior of user and kernel accesses. Only OLTP has a significant fraction of kernel accesses, and the kernel accesses tend to be better behaved than the user accesses!