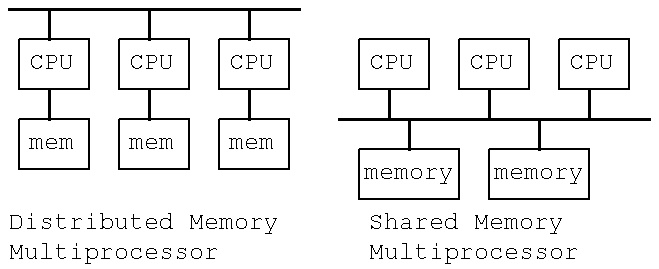
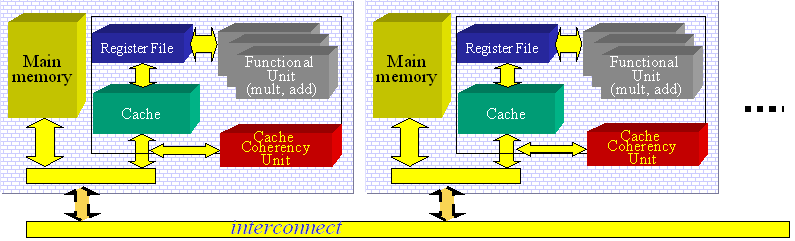
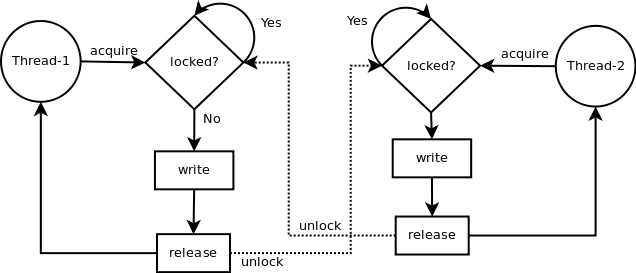
**Team Assignment 1**

1. There are three terms is confusing that we need to tell their differences. (1) Multiprocessor is the computer which runs multiple instruction stream simultaneously to cooperatively execute a single program. (2) When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute. (3) Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system. It also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them.

2. There are two main types of MIMD or multiple architecture: shared memory multiprocessor and distributed memory multiprocessor. Shared memory architecture (SMA) refers to a multiprocessing design where several processors access globally shared memory. Shared memory architectures may use: Uniform Memory Access (UMA), Non-Uniform Memory Access (NUMA) and Cache-only memory architecture (COMA). A Distributed-Memory Multiprocessor (DMM) is built by connecting nodes, which consist of uniprocessors or of shared memory multiprocessors (SMPs), via a network, also called Interconnection Network (IN) or Switch.

3. Since we’ve learn that distributed memory architecture and shared memory architecture, there is also a mixture of shared and distributed memory in we should know. Distributed shared memory (DSM) is a form of memory architecture where the (physically separate) memories can be addressed as one (logically shared) address space. It is not a single centralized memory but its address space is shared.



4. There is a common problem that we meet in parallel implementation is busy waiting. It is also called spinning in which a processor repeatedly checks to see if a condition is true. To reduce large amount busy waiting cost of the performance and resources of the machine, we find that semaphore is an effect solution. Semaphore is a type of generalized lock which defined by Dijkstra in the last 60s. There are two atomic operations in semaphore, P(S) waits for semaphore to become positive, then decrement it by 1; V(S) increments semaphore by 1 and wakes up a waiting thread at P(S). The former is called before a critical section, while the latter is called after a critical section.

**Team Assignment 2-1**

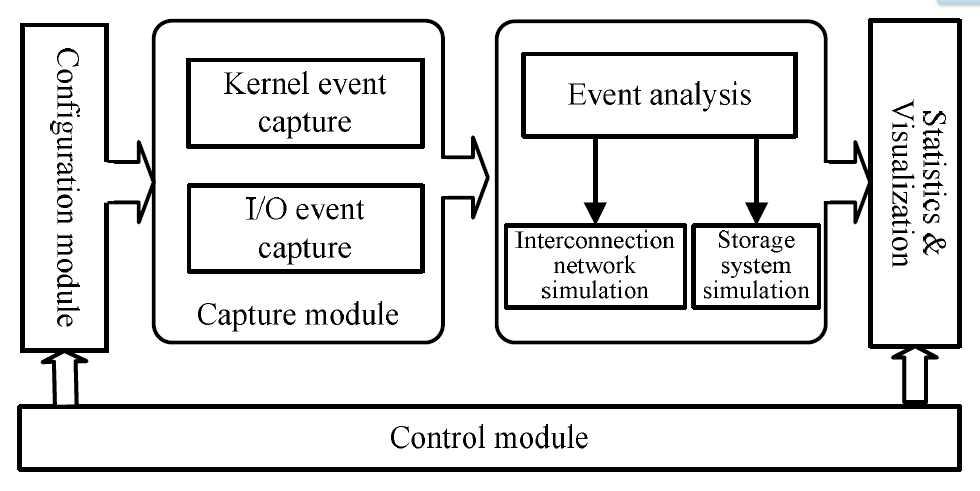
**Report on “SimNUMA: Simulating NUMA-Architecture Multiprocessor Systems Efficiently [1]”**

This article is a simulating argument for an execution-driven full-system simulator dedicated for Non-uniform memory access systems, called SimNUMA.

Compare with symmetric multi-processing (SMP) architecture, NUMA architecture is more scalable and more widely used in high-end servers and computing system. It can be scaled up/down flexibly in modular based on commercial interconnection technologies. The simulator uses the same type of processor with the target machine in the host system, and proposes a new method to capture remote-memory accesses.

The main specialty of NUMA system is that its memory are global addressed but distributed in each processing nodes, leading to different latencies for different memory locations. Therefore, memory accesses can be divided into two categories: local-memory and remote-memory accesses, furthermore, latencies of remote-memory accesses are also different, depending on the length of transport route and congestion status of the interconnection network.

There are six components in this system, including event capture, analysis, interconnection network & storage system simulation, control, configuration and statistic & visualization module. The architecture of SimNUMA is as below figure:



The event capture module are transmitted from kernel to the event analysis module which runs in user mode via pre-allocated exchange buffers. The event analysis module analyzes events one by one, invokes interconnection network and storage system simulation module when necessary, and constructs time-axes in the target system for each application process. The results are written into a log file, which will be used by statistics & visualization module. The system control module provides a command-line interface to users and supports startup, stop and configuration commands.

**Specific Q&A**

(a) Programming languages it supports: It supports C++ programming language.

(b) Compare and contrast the key features of the machine with what we covered in class   
Compare with symmetric multi-processing (SMP) architecture, NUMA architecture is more scalable and more widely used in high-end servers and computing system. It can be scaled up/down flexibly in modular based on commercial interconnection technologies.

(c) Date the machine was built: This simulator is built in 2013

(d) Is it still in use?   
Yes. The number of processor cores in NUMA systems increases rapidly with the development of multi-core processors. The simulation is still used in 64-way server with quad-core processor has 256 cores and a 128-way server with eight-core processor will have 1024 cores systems.

(e) What type of applications does it support?   
It supports bzip2, hmmer, sjeng, sphinx, dealII (SPEC CPU 2006) and LU(SPLASH-2).

(f) Operating systems (if information available)   
The simulator, the event capture module, the event analysis module and the system control module are implemented in Linux system (the authors use Ubuntu Server Linux 11.04 (kernel: 2.6.38) in this article). The statistics & visualization module run in Windows system and provides different kinds of statistics and graphs via a GUI interface on the basis of the log file output by the event analysis module.

(g) What type of interconnection network does it have?  
The simulator supports two interconnection techniques: Infiniband and Intel QuickPath Interconnect (QPI), and three kinds of topologies: full-connection/full-switch, 2DTorus and Hypercube. Other interconnection techniques (e.g. AMD HyperTransport) and network topologies (e.g. 3D-mesh/Torus) are supported as well.

**Reference**

[1] Yi Liu, Yanchao Zhu, Xiang Li, Zehui Ni, Tao Liu, “SimNUMA: Simulating NUMA-Architecture Multiprocessor Systems Efficiently”, *Special issue: Parallel and Distributed Systems (ICPADS), 2013 International Conference on* 15-18 Dec. 2013, Pages 341 - 348, doi> 10.1109/ICPADS.2013.55

URL: http://0-ieeexplore.ieee.org.skyline.ucdenver.edu/xpls/abs\_all.jsp?arnumber=6808192

**Team Assignment 2-2**

**Report on “WildFire: A Scalable Path for SMPs [1]”**

Researchers have been searching for scalable alternatives to the symmetric multiprocessor (SMP) architecture. Many people considered that the SMP will not scale over time. Compared to between SMP and NUMA, SMPs provide a simple model than Non-Uniform Memory Architecture (NUMA) for several architectural reasons. SMP’s uniform access time to shared memory provides a simple programing and performance model. With these various benefits, this paper proposed WildFire which is a distributed shared-memory (DSM) prototype implementation based on large SMPs.

The WildFire is an internal code name for a prototype shared memory multiprocessor developed by Sun Microsystems. It relied on two main techniques for creating application-transparent locality: Coherent Memory Replication (CMR), and Hierarchical Affinity Scheduling (HAS). These two techniques create extra node locality and they showed SMP-like performance can be achieved with no NUMA-specific optimizations. WildFire supports up to 112 UltraSPARC I or II processors, runs on Solaris 2.6, and is a application-binary-interface (ABI) compatible with Sun’s SMP multiprocessors. WildFire connects two to four unmodified Sun Enterprise E6500/E5500/E4500/E3500TM SMP servers. They experimented a performance study of a large OLTP benchmark running on DSMs built from various-sized nodes and with varying amounts of application-transparent locality. Sun’s WildFire prototype can implement the multiple SMP, it can be a viable architecture for OLTP workloads. The performance of evaluation at MSMP implemented WildFire prototype with Coherent Memory Replication showed 2.13 times that of a NUMA implementation with no optimizations.

**Specific Q&A**

1) WildFire is supporting the c++ language

2) Compare and contrast the key features of the machine  
- Cach Only Memory Architecture (COMA): the local memories (DRAM) at each nodes are used as cache.   
- Non-Uniform Memory Access (NUMA): CPUs of multiprocessor have each memory in NUMA systems. Therefore, memory access time depends on the memory location.  
- Hybrid NUMA-COMA: The pages start in NUMA mode and switch to COMA if appropriate, and this system is implemented in the Sun Microsystem’s WildFire.

3) The first version was published on 1997, and this is a version 2.0.

4) The operating systems is based on the Solaris OS and it is still in use.

5) WildFire’s interface has two different type: the Network Interface Address Controller (NIAC), which implements the coherence protocol, and the Network Interface Data Controller (NIDC), which provide a fat connection to/from the interconnect.

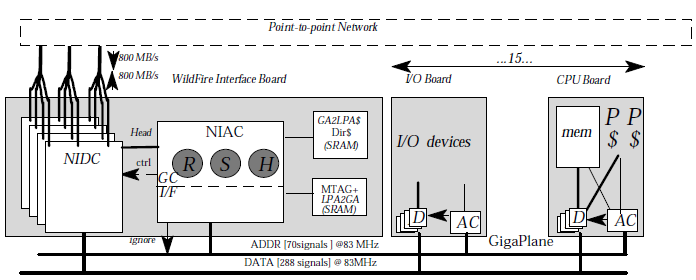


Figure 1. WildFire Architecture

**Reference**:

[1] Erik Hagersten and Michael Koster, “WildFire: A Scalable Path for SMPs”, *In Proc. High Performance Computer Architecture of IEEE, 1999,* Pages 172 - 181