* I agree with Shawn's first post stating that the monolithic kernel is best suited for a distributed environment. He states that both kernels have advantages and the different kernels may be suited for different environments. In this case, we are looking for the kernel that is best suited for a distributed environment.

Last semester, I heard Dr. Rhoda Baggs present her research at the CSE 5500 Friday seminar class. She is doing performance testing on Ubuntu Linux and Windows. I was able to find the paper at:[http://proceedings.isecon.org/download/5xxdto3ejwwczwxzsymr (Links to an external site.)](http://proceedings.isecon.org/download/5xxdto3ejwwczwxzsymr) [1].

In the paper she includes the results of the tests they have been running. They have found that the Ubuntu experiments ran faster, but they have not analyzed why it runs faster. [1]

Researching Ubuntu Linux and Windows kernels, I found that:

Ubuntu is a monolythic kernel [2] and Windows is a hybrid kernel [3][4][5].

I also wanted to know which operating systems are microkernels. Researching microkernels, I found some examples on Indiana University’s webstite, stating “Mach and its derivatives, the most prominent examples of the microkernel architecture, are the foundations of systems such as Tru64 Unix, the [GNU (Links to an external site.)](https://kb.iu.edu/d/acuo) Hurd, and Mac OS X.” [6]

I also looked at Distributed networks examples. I found that an example of a Distributed network is the cloud. In a paper from IEEE Internet Computing,, M.D. Dikaiakos writes:

 Cloud computing is a recent trend in IT that moves computing and data away from   
desktop and portable PCs into large data centers. It refers to applications delivered as services over the Internet as well as to the actual cloud infrastructure — namely, the hardware and systems software in data centers that provide these services...

...A cloud comprises processing, network, and storage elements, and cloud architecture consists of three abstract layers. Infrastructure is the lowest layer and is a means   
of delivering basic storage and compute capabilities as standardized services over the network. Servers, storage systems, switches, routers, and other systems handle   
specific types of workloads, from batch processing to server or storage augmentation   
during peak loads. The middle platform layer provides higher abstractions and services to develop, test, deploy, host, and maintain applications in the same integrated   
development environment. The application layer is the highest layer and features a   
complete application offered as a service [7].

In a paper written by Judith Hurwitz, titled *The role of the Operating System in Cloud Environments*, she writes that,

One of the most important ways to support the underlying complexity of well-managed   
cloud computing resources is through the operating system. Operating systems such as   
Linux are designed to support these requirements so that cloud services and   
application services do not have to recreate underlying technologies tailored for each specific deployment. Users gain control, predictability, scalability, and security by having critical shared infrastructure at the operating system level. In addition, an   
operating system such as Linux supports important standards that enhance portability   
and interoperability across cloud environments. Operating system platforms are   
designed to hide much of the complexity required to support applications running in   
complex and federated environments. Much of the functionality required for the   
effcient operation of many applications is built in to the operating system. It needs to work competently in the background to ensure that all the right resources (such as processing power, required memory and storage) are available when needed. In addition, the operating system implements the level of security and quality of service to ensure that applications are able to access the resources needed to deliver an acceptable   
level of performance. [8]

 There role of the operating system in the cloud environment is important for its success. I think that a monolithic kernel is currently the best choice for distributed networks.

**References**

[1] Baggs, R., et. al., “Performance Analysis of I/O-Intensive & CPUIntensive Benchmarks on Windows 7, 8.1 & Ubuntu Linux 14.04LTS”, 2015 Proceedings of the Information Systems Education Conference Orlando, Florida USA, 2015.

Web. Jan 2016.  <[http://proceedings.isecon.org/download/5xxdto3ejwwczwxzsymr (Links to an external site.)](http://proceedings.isecon.org/download/5xxdto3ejwwczwxzsymr)>

[2]  Ask Ubuntu, [Is the Ubuntu kernel modular? (Links to an external site.)](http://askubuntu.com/questions/173578/is-the-ubuntu-kernel-modular) 2012.

Web. Jan 2016. [http://askubuntu.com/questions/173578/is-the-ubuntu-kernel-modular (Links to an external site.)](http://askubuntu.com/questions/173578/is-the-ubuntu-kernel-modular)

[3] Microsoft, User Mode and Kernel Mode, 2016.

Web. Jan 2016. [https://msdn.microsoft.com/en-us/library/windows/hardware/ff554836(v=vs.85).aspx (Links to an external site.)](https://msdn.microsoft.com/en-us/library/windows/hardware/ff554836(v=vs.85).aspx)

[4] Microsoft, Windows Kernel, 2014.

Web. Jan 2016. [https://social.technet.microsoft.com/Forums/windows/en-US/68a08c8d-1c28-4ab2-896b-5934823847f7/windows-kernel?forum=w8itprogeneral (Links to an external site.)](https://social.technet.microsoft.com/Forums/windows/en-US/68a08c8d-1c28-4ab2-896b-5934823847f7/windows-kernel?forum=w8itprogeneral)

 [5] Microsoft, Overview of Windows Components, 2016.

Web. Jan 2016. [https://msdn.microsoft.com/en-us/library/windows/hardware/ff558769(v=vs.85).aspx (Links to an external site.)](https://msdn.microsoft.com/en-us/library/windows/hardware/ff558769(v=vs.85).aspx)

[6] Indiana University, What are kernels and microkernels? 2013-11-15*.*

Web. Jan 2016. < [https://kb.iu.edu/d/agsv (Links to an external site.)](https://kb.iu.edu/d/agsv)>

[7] Dikaiakos, M. D., et. al., Cloud Computing: Distributed Internet Computing for IT and Scientific Research, 2009. [Internet Computing, IEEE  (Links to an external site.)](http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=4236) (Volume:13 ,  [Issue: 5  (Links to an external site.)](http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=5233600))

Web. Jan 2016. <[http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=5233607 (Links to an external site.)](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5233607)>

 [8] Hurwitz, J., Kaufman, M., The Role of the Operating System in Cloud Environments, 2011.

Web. Jan 2016. <[https://www.redhat.com/f/pdf/The\_Role\_of\_the\_OS\_in\_the\_Cloud.pdf (Links to an external site.)](https://www.redhat.com/f/pdf/The_Role_of_the_OS_in_the_Cloud.pdf)>

**Kernel**(Explain which kernel is better suited for distributed environment.)

A distributed environment is made up of loosely coupled computers. These systems may run different operating systems, but appear as one system to the user. They have their own clock and memory. The operating systems in a distributed environment may be a mixture of Windows, Linux, Unix, etc. These operating systems may have different kernels. For example, Linux is a monolithic kernel. Windows is now a hybrid kernel which combines advantages of both monolithic and microkernel but still has the same disadvantages as a monolithic kernel. Currently the monolithic kernel appears to be the best performer in distributed environments. According to Susmit Bagchi, who performed tests using a Linux monolithic kernel with distributed interprocess communication,

The efficient design of distributed IPC is a key to realize low latency remote tasking, distributed shared memory,   
distributed synchronization and message-based mobile computing.   
The monolithic kernel based design of distributed IPC architecture can offer high performance and low latency.   
The efficient distributed IPC systems can be used to implement event-based distributed computing systems [1].

## **The monolithic vs microkernel debate has been around for a while. In 1992 there was an interesting discussion known as the Tanenbaum/Linus "**[**Linux is obsolete (Links to an external site.)**](http://www.oreilly.com/openbook/opensources/book/appa.html)**" debates [2]. Linux is still around and performing well. In fact, Linux is a popular choice for Cloud Computing. According to IBM, “Linux and open source technologies play a huge role.” [3].**

**Interrupts**(Example of how an interrupt enhances the utilization of processor)

There is synchronous and asynchronous I/O. Asynchronous is more efficient because it enables the processor to working on other tasks while the I/O is in process. I/O is a slower process than the processor speed, so more processing can be done if the processor continues to process while waiting on the I/O. The OS keeps track of the I/O devices and their status in a table. The OS also has a wait queue for additional requests.

For example, the processor receives a request to print, while a document is printing the processor can continue working on other tasks. The user can also continue to type while the document is printing. This is another I/O that is also in process. If another print request is made while the printer is printing the first request, the second print request goes into the wait queue. When the printer is finished with the first print request, an interrupt is sent to the processor to let the processor know the printer finished, then the processor can interrupt its current task, finish processing the first printing task, begin the second print task, and then proceed with the task it was working on before the interrupt.

**I/O operations**(In respect to kernel, which is more efficient)

I think there are situations in which each can be efficient. And there are also times when they are not the most efficient choice.

Programmed I/O is also called polling. This can be very fast and efficient if both the device and controller are fast. It becomes inefficient if it repeatedly doesn’t find a device ready. It is stuck in a loop.

Interrupt driven I/O has a queue so the processor can continue to work while a device is busy and other requests are waiting in the queue. This requires efficient interrupt handling.

DMA is most efficient and useful for fast transfers of large blocks of data between memory and a device.

Operating Systems have an Application I/O interface so it can treat the different devices in a standard way, and an I/O subsystem to handle scheduling, buffering, caching and other services. There are also kernel data structures such as the device table that are used to keep track of the devices and their states.

**References**

[1]  Bagchi, S., Distributed IPC using Virtual Device Driver in Monolithic Kernel,

      IEEE International Conference on Embedded and Real-Time Computing Systems and

      Applications, IEEE, 2012.

[2]  DiBona, C., “Appendix A: The Tanenbaum-Torvalds Debate

      in Open Sources: Voices from the Open Source Revolution, 1st Edition Jan. 1999.

      Web. Jan. 2016. <[http://www.oreilly.com/openbook/opensources/book/appa.html (Links to an external site.)](http://www.oreilly.com/openbook/opensources/book/appa.html)>

 [3] Jones, M. T., Cloud Computing with Linux, 11 February 2009.

      Web. Jan. 2016. < [http://www.ibm.com/developerworks/library/l-cloud-computing (Links to an external site.)](http://www.ibm.com/developerworks/library/l-cloud-computing)/>

# DQ1

No unread replies.3636 replies.

**Kernel, Interrupts, Direct Memory Access and Memory Management**

**Kernel**

Most operating systems, until recently, featured a large **monolithic kernel**. A **monolithic kernel**is a large kernel containing virtually the complete operating system, including scheduling, file system, device drivers, and memory management. All the functional components of the kernel have access to all of its internal data structures and routines. Typically, a monolithic kernel is implemented as a single process, with all elements sharing the same address space. A **microkernel** is a small privileged operating system core that provides process scheduling, memory management, and communication services and relies on other processes to perform some of the functions traditionally associated with the operating system kernel.

**Interrupts**

Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal sequencing of the processor. Most common classes of interrupts.

Program: Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.

Timer: Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.

I/O: Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.

Hardware failure: Generated by a failure, such as power failure or memory parity error.

Interrupts are provided primarily as a way to improve processor utilization.

**Direct Memory Access**

When large volumes of data are to be moved, a more efficient technique is required: direct memory access (DMA). The DMA function can be performed by a separate module on the system bus or it can be incorporated into an I/O module. In either case, the technique works as follows. When the processor wishes to read or write a block of data, it issues a command to the DMA module, by sending to the DMA module the following information:

* Whether a read or write is requested
* The address of the I/O device involved
* The starting location in memory to read data from or write data to
* The number of words to be read or written

The processor then continues with other work. It has delegated this I/O operation to the DMA module, and that module will take care of it. The DMA module transfers the entire block of data, one word at a time, directly to or from memory without going through the processor. When the transfer is complete, the DMA module sends an interrupt signal to the processor. Thus the processor is involved only at the beginning and end of the transfer.

The DMA module needs to take control of the bus to transfer data to and from memory. Because of this competition for bus usage, there may be times when the processor needs the bus and must wait for the DMA module. Note that this is not an interrupt; the processor does not save a context and do something else. Rather, the processor pauses for one bus cycle (the time it takes to transfer one word across the bus). The overall effect is to cause the processor to execute more slowly during a DMA transfer when processor access to the bus is required.

**Memory Management**

Most operating systems have five principal storage management responsibilities.

**Process isolation:** The operating system must prevent independent processes from interfering with each other's memory, both data and instructions.

**Automatic allocation and management:** Programs should be dynamically allocated across the memory hierarchy as required. Allocation should be transparent to the programmer. Thus, the programmer is relieved of concerns relating to memory limitations, and the operating system can achieve efficiency by assigning memory to jobs only as needed.

**Support of modular programming:** Programmers should be able to define program modules, and to create, destroy, and alter the size of modules dynamically.

**Protection and access control:** Sharing of memory, at any level of the memory hierarchy, creates the potential for one program to address the memory space of another. This is desirable when sharing is needed by particular applications. At other times, it threatens the integrity of programs and even of the operating system itself. The operating system must allow portions of memory to be accessible in various ways by various users.

**Long-term storage:** Many application programs require means for storing information for extended periods of time, after the computer has been powered down.

Explain which kernel is better suited for distributed environment.

Write an example for which the use of interrupt is enhances the utilization of processor.

In respect to kernel, interrupt, explain which of three:

* Programmed I/O
* Interrupt-driven I/O
* Direct memory access (DMA)

I/O operations techniques is more efficient. Explain how operating systems meet these requirements.