FUNDAMENTAL CONCEPTS OF THE DISTRIBUTED OPERATING SYSTEM

Anup Kumar, Ranjeeta chauhan, Abhineet Sinha
Student, B.Tech, Department of Electronics and Computers Engineering
Dronacharya College of Engineering, Gurgaon, India

Abstract- **An operating system runs on computer hardware and serves as a platform for other software to run on the computer system .An operating system is a program which acts as an interface between user and computer hardware .As computer operating system have moved from essentially single-process or single –task system to single-processor, multiuser, multitasking system .The trend is of multiprocessor multitasking system. Distributed operating system is an operating system for a network of autonomous computers connected by a communication network. It is a collection of independent computers that appears to its users as a single coherent system A distributed operating system controls and manages the hardware and software resources of a distributed system such that its users view the entire system as a powerful monolithic computer system. In this paper all the aspects of distributed system, how the system has outsourced the centralized operating system, its merits, classification and their evolution.**

Index Terms- **Distinct service provisioners, Scalability Inherent Distribution, Message parsing, duplication, Explicit message-based.**

I. INTRODUCTION

When a program is executed in distributed environment, the user is unaware about the location of the resource accessed. The basic issues in the design distributed in operating system is similar as in traditional operating system viz. process synchronization ,deadlocks, scheduling, file system , enterprises communication, memory and buffer management , failure recovery etc. However ,several idiosyncrasies of a distributed system, namely the lack of both shared memory and physical global clock, and unpredictable communication delays, make the design of distributed operating system much more difficult.

The Distributed operating system extends the concept of resources management and user friendly interface for shared memory computers a step forward, encompassing a distributed computing system consisting of several autonomous computers connected by a communication network. Distributed operating system appears to its users as a centralized operating system for a single computer, but it runs on multiple-independent computers.An identical copy of the operating system may run at every computer. On the other hand, some computers in the system that serve a special purpose might run an extended version of him operating systems.

A **distributed operating system** is a software over a collection of independent, networked, communicating, and physically separate computational nodes. Each individual node holds a specific software subset of the global aggregate operating system. Each subset is a composite of two distinct service provisioners. The first is a ubiquitous minimal kernel, or microkernel, that directly controls that node's hardware. Second is a higher-level collection of system management components that coordinate the node's individual and collaborative activities. These components abstract microkernel functions and support user applications. The microkernel and the management components collection work together. They support the system's goal of integrating multiple resources and processing functionality into an efficient and stable system. This seamless integration of individual nodes into a global system is referred to as transparency, or single system image; describing the illusion to users of the global system's appearance as a single computational entity.

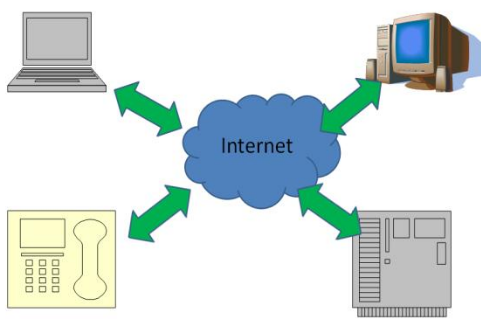


Fig: A Distributed Operating System

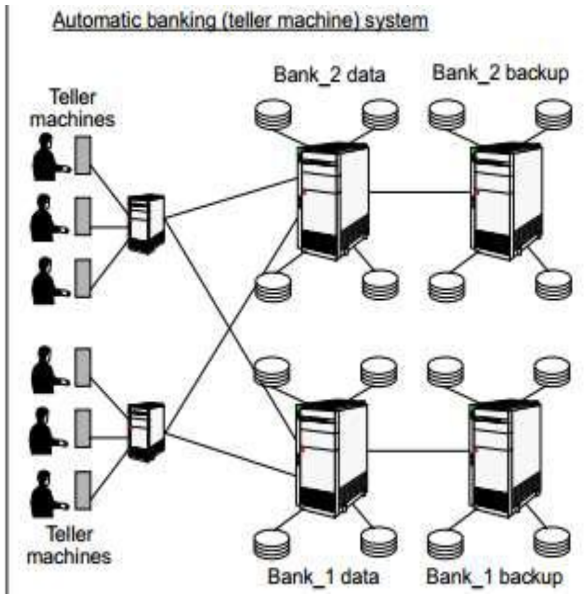
II. HISTORY

Research and experimentation efforts began in earnest in the 1970s and continued through 1990s, with focused interest peaking in the late 1980s. A number of distributed operating system were introduced during this period; however very few of these implementations achieved even modest commercial success.
In the mid-1970s, research produced important advances in distributed computing. These breakthroughs provided a solid, stable foundation for efforts that continued the 1990s
The accelerating proliferation of multi processor and multi-core processor systems research led to a resurgence of the distributed OS concept.

III. NEED OF DOS

1. Performance: Very often a collection of processors can provide higher performance along with better performance ratio. Multiple systems can decrease the work load and lead to better performance by reducing the workload.

2. Distribution: Many applications in today's world like banking, commercial automotive
System require spatially sepatated machines.



3. Communication: Disrtibuted operating system facilitates human to human communication which is an essential aspect of any technical or commercial procedure.
It covers the wide base of any successful work.

4. Resource Sharing: Resources such as hardware - disks and printers, software - files, windows, and data objects Hardware sharing for convenience reduction of cost Data sharing for consistency - compilers and libraries exchange of information – database cooperative work – groupware.
5. Transparency: It provides the transparency of

- Access
- Location
- Concurrency
- Replication
- Failure
- Scaling

6. Scalability: It is the factor that how the system handles growth.
It's of two cases:
Small system which consist two computers and a file server on a single network while the large systems contain current Internet.

IV. ADVANTAGES OF DISTRIBUTED SYSTEM OVER CENTRALIZED SYSTEMS

1. **Speed:** when used to implement parallel processing where only goal is to achieve maximum speed on a single problem, distributed systems can achieve very high speed as compared to the centralized ones.
2. **Inherent Distribution -** Another fact for building a distributed system is that some applications are inherently distributed. Banking and Airline reservation etc. are instances of the applications that are inherently distributed. When all the branches of a bank are joined, we have a commercial distributed system.
3. **Reliability-** Another potential importance of a distributed system over a centralized system is its higher reliability. By distributing the workload over many machines, a single chip failure will bring down at most one machine, leaving the rest intact. Most preferably, if 5 percent of the machines are down at any moment, the system should be able to continue to work with a 5 percent loss in performance. For critical applications, such as control of nuclear reactors or aircraft, using a distributed system to achieve high

reliability may be a dominant consideration.in essence if one machine crashes , the system as a whole can still servive.

4. **Incremental Growth:** Finally, incremental growth is also potentially a big plus. Often a company will buy a mainframe with the intention of doing all its work on it. If the company prospers and the workload grows, at a certain point the mainframe will no longer be adequate.
   The only solution is to either replace the mainframe with a larger one (if it exists), or add a second mainframe. Both of these can cause management difficulties with the company's operations. In contrast, with a distributed system, it may be possible to simply add more processors to the system, thus allowing it to expand gradually as the need arises. In one line we can say that computing power can be added in small increments.
5. **Economics:** microprocessors offer a better price than the mainframe ones thereby being affordable.10, 000 CPUs executing 50 MIPS yields executing 500,000 MIPS. Not possible for a single CPU to achieve

V. SIGNIFICANCE IN DOS

We now present some ideas, concepts, and principles from distributed systems which we feel are particularly related to OS design for modern machines. We draw parallels with existing ideas in operating systems, and where there is no corresponding notion, suggest how the idea might be applied. We have found that viewing OS problems as distributed systems problems either suggests new solutions to current problems, or fruitfully casts new light on known OS techniques.

VI. MESSAGE PASSING VS. SHARED MEMORY

Traversing a shared data structure in a modern cache-coherent system is equivalent to a series of synchronous RPCs to fetch remote cache lines. For a complex data structure, this means lots of round trips, whose performance is limited by the latency and bandwidth of the interconnect. In distributed systems, ``chatty" RPCs are reduced by encoding the high-level operation more compactly; in the limit, this becomes a single RPC or code shipping. When communication is expensive (in latency or bandwidth), it is more efficient to send a compact message encoding a complex operation than to access the data remotely. While there was much research in the 1990s into distributed shared virtual memory on clusters which is rarely used today. In an OS, a message-passing primitive can make more efficient use of the interconnect and reduce latency over sharing data structures between cores. If an operation on a data structure and its results can each be compactly encoded in less than a cache line, a carefully written and efficient userlevel RPC implementation which leaves the data where it is in a remote cache can incur less overhead in terms of total machine cycles. Moreover, the use of message passing rather than shared data facilitates interoperation between heterogeneous processors. Explicit message-based communication is amenable to both informal and formal analysis, using techniques such as process calculi and queueing theory. In contrast, although they have been seen by some as easier to program, it is notoriously difficult to prove correctness results about, or predict the performance of, systems based on implicit shared-memory communication.

VII. CONSISTENCY

Maintaining the consistency of replicas in current operating systems is a fairly simple affair. Typically an initiator synchronously contacts all cores, often via a global IPI, and waits for a
Conformation. In the case of TLB shootdown, where this occurs frequently, it is a well-known scalability bottleneck. Some existing optimizations for global TLB shootdown are
Familiar from distributed systems, such as deferring and coalescing updates. Uhlig's TLB shootdown algorithm appears to be a form of asynchronous single-phase commit. However, the design space for agreement and consensus protocols is large and mostly unexploited in OS design. Operations such as page mapping, file I/O, network connection setup and teardown, etc. have varying consistency and ordering requirements, and distributed systems have much to offer in insights to these problems, particularly as systems become more concurrent and diverse. Just as importantly, reasoning about OS state as a set of consistent replicas with explicit constraints on the ordering of updates seems to hold out more hope of assuring

correctness of a heterogeneous multiprocessor OS than low-level analysis of locking and critical sections.

VIII. NETWORK EFFECTS

Perhaps surprisingly, routing, congestion, and queueing effects within a computer are already an issue. Conway and Hughes document the challenges for platform firmware in setting up routing tables and sizing forwarding queues in a HyperTransport-based multiprocessor, and point out that link congestion (as distinct from memory contention) is a performance problem, an effect we have replicated on AMD hardware in our lab. These are classical networking problems. Closer to the level of system software, routing problems emerge when considering where to place buffers in memory as data flows through processors, DMA controllers, memory, and peripherals. For example, data that arrives at a machine and is immediately forwarded back over the network should be placed in buffers close to the NIC, whereas data that will be read in its entirety should be DMAed to memory local to the computing core.

IX. CONCLUSION

Modern computers are inherently distributed systems, and we miss chances to tackle the OS challenges of new hardware if we neglect insights from distributed systems research. Distributed and real-time systems are becoming powerful enough, in terms of processing capability, to support technologies for distributed architectures, which may enable fault-tolerance and load sharing for applications. However, for some of these technologies there are still shortcomings, especially with regard to other supports. Security also needs to be managed consistently on all levels of the system, and a common platform should be collaborative. The system needs all the necessary information to be available. More over research should work towards widely adopted standards, in order to accomplish the vision of cooperative distributed systems. Middleware or distributed operating systems need the essential capabilities for this to happen.

ACKNOWLEDGEMENT

The material is based upon work supported by the various papers, journals and substances created and presented online for references .The different online student guide various pdfs presentations etc have played a vital role .Sites like Wikipedia also provide a knowledgeable resources. The credit also goes to the search engine for briefing about the topic and the support which it has provided in the completion of this paper.

REFERENCES:

http://www.csc.villanova.edu/~schragge/CSC8530/Intro.html.

http://www.ida.liu.se/~TDDB37/lecture-notes/lect1.frm.pdf.

https://www.usenix.org/legacy/events/hotos09/tech/full_papers/baumann/baumann_html/index.html.

http://csnoteshelp.blogspot.in/2011/12/advantage-of-distributed-systems-over.html.

http://en.wikipedia.org/wiki/Distributed_operating_system.

http://google.o.in.