AMOEBA

Sanjana Singh[1], Surbhi Bhardwaj[2], Vinay Mathur[3]
Dronacharya College of Engineering, Gurgaon, Khentawas, Farrukh Nagar

I. INTRODUCTION

Amoeba is a complete distributed operating system design which includes all the basic facilities that one would expect from a conventional operating system. In other words Amoeba is an Operating System that performs all the standard functions of any OS, but it performs them with a collection of machines. In this operating system users effectively log into the system as a whole, and not to a specific machine. One of the main goals of the Amoeba development team was to design a transparent distributed system that allows users to log into the system as a whole. Its design and implementation started in 1981 and is developing at the VrijeUniversiteit in Amsterdam but previously it was developed by Centrum voorWiskunde, also in Amsterdam.

The main components for the architecture of amoeba are the processor pool, workstations (SUN-3s and VAX stations are supported), X-terminals, servers and gateways. The assumptions being made for the architecture of amoeba was that the memory and processors are sufficiently cheap that each user will be allocated multiple processors, and each processor will have plenty of memory to run applications, without the need for backing store. The amoeba distributed operating system was built up on the popular paradigm or assumption of client processes communicating with services via message transition. Amoeba uses capabilities to access services and the objects these services implement. The amoeba's capabilities is a 256 bit reference to an object. The first 64 bits are known as the port. Refer to the service managing the object. The next 64 bits are available to the system for use as a location hint. The rest 128 bits are allocated by the services to identify the objects. A capabilities is generated in such a way and contains sufficient bits that the probability of an unauthorized user guessing an objects capabilities in negligible. Its function is process management and inter-process communication. There is thus also a minimum amount of state that has to migrate when a cluster migrates. The researchers believe that this was one of the essential choices that made this mechanisms work. Things would have been much more difficult if we had to deal with things like 'open file state,' 'controlling terminals' or the complicated connection state of a sliding-window protocol. The Amoeba inter-process communication mechanism has also been vital to the success of our design. Firstly the communicating entities are named using aoccasion-independent naming mechanism that uses an underlying locate service to find out dynamically where the packets have to be sent. None of the migration apparatus has to worry about rerouting messages, no forwarding addresses have to be left behind and the hosts can forget about the existence of a cluster immediately after migration is complete. Then comes the simplicity of the Amoeba protocols contribute enormously to the portability of clusters. The protocol has only a few states in which it can stay for arbitrary lengths of time and it is relatively easy to migrate a cluster in these states using the "I'm frozen, don't bother me" messages described earlier. When the protocol is in any of the other states, the Amoeba Kernel can wait until the protocol. The most important conclusion that the researchers have drawn from this design—1.which is trdll being implemented and 2. is that it is possible to build a simple mechanism that is sufficient to realize downloading, migration, exception handling, check-point hag, emulation and debugging. Although the implementation is not complete at the time of writing this paper, we expect to finish soon enough to present performance information at the SOSP conference. reaches a 'migratable' one.

II. FEATURES

- The Amoeba architecture is designed as a collection of micro-kernels.
- An Amoeba system consists of four principle components: user workstations, pool processors, specialized servers, and gateways.
- Amoeba is built upon a microkernel architecture.
- The microkernel supports the basic process, communications, and object primitives. It also handles device I/O and memory management.
- Each machine in the Amoeba system runs a small identical software program - called the microkernel.
- The function of the kernel is to allow efficient communication between client processes, which run application programs, and server processes, such as the Bullet File server or the directory server.
- Amoeba implements a standard distributed client / server model, where user processes and applications (the clients) communicate with servers that perform the kernel operations.
- Threads
- Each process has its own address space and contains multiple threads.
- These threads have their own stack and program counter, but share the global data and code of the process.
- Remote Procedure Calls
- RPC is the basic communication mechanism in Amoeba. Communication consists of a client thread sending a message to a server thread, then blocking until the server thread sends back a return message, at which time the client is unblocked.
- Amoeba uses stubs to access remote services which

hide the details of the remote services from the user. A special language in Amoeba called the Amoeba Interface Language (AIL) generates these stubs automatically. The stubs will then marshal parameters and hide the communication details from the user.

- Group Communication: Amoeba provides a mechanism that allows all receivers in a one-to-many configuration to receive a transmitted message in the same order. This simplifies parallel processing and distributed programming problems.

III. HISTORY

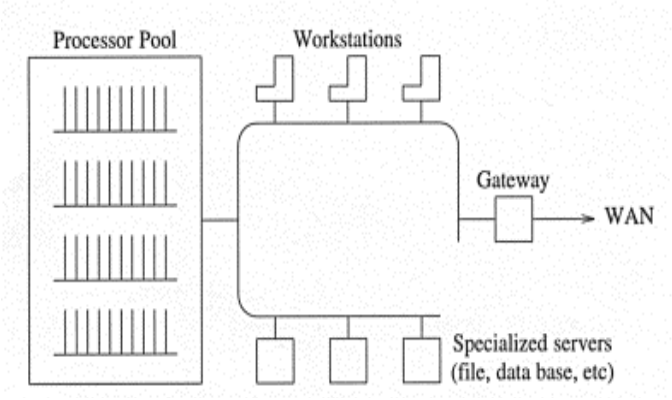
Amoeba is a complete distributed operating system design, including all the basic facilities that one would expect from a conventional operating system. It is currentlybeing developed at the VrijeUniversiteit in Amsterdam, where its design and implementation were begun in 1981, and it was previously developed jointly with the Centrum voorWiskunde, also in Amsterdam. The Amoeba system model is an example of the processor pool model. The main components makingup the architecture are the processor pool,X-terminals, workstations (SUN-3s and VAXstations are supported), servers and gateways. The gateways arebasically used for connecting Amoeba sites over WANs. The assumptions behind this architecture are that memory and processors are sufficiently cheap that each user will be allocated multiple processors, and each processor will have plenty of memory to run applications, without the need for backing store. Rather than allocating a multiprocessor to each user, processing power is largely concentrated in the processor pool, where it can be shared more flexibly among users and more economically housed and interconnected. First proto was release in 1983 (V1.0), last official release 1996 (V5.3). The current computer environment started arriving in the early 1980s. It was at this time that technology allowed computers to become smaller and cheaper, allowing for each individual to have their own personal computer. These personal computers began replacing the bulky computers that employees would have to share its use time with one another. These personal computers then started being networked together, allowing communication and sharing of resources. In fact this time period is where the basis of the current networking was born, and out of this period came a need for multiple computers to be connected together but act as one. This concept of multiple computers acting as one is known as parallelism and one of the things Amoeba was designed to handle. Amoeba was first developed in the early 1980s at the VrijeUniversiteit in Amsterdam, Netherland, under the guidance of Andrew Tanenbaum and with the cooperation of Centrum voorWiskundeenInformatica. The purpose of Amoeba was to create a simplistic way to handle the needs of distributed systems and parallels while maintaining the transparency for its users. The transparency that is being maintained is that the user has no clue how the underlying system works, where the files are stored or on which machine the process is running. Traditional distribution systems have workstations that users can log into, and while on these workstations they can tell that the process they run, only run on the local machine and other calls must occur for the process to run on a different machine. In this, users can tell that there are different machines. With Amoeba the user doesn't know where the process is running, where files are stored or even if certain files are stored in the same physical memory. The last official release of Amoeba came in 1996 with version 5.3. It came 13 years after the prototype was released in 1983. This doesn't mean that others have not done their own work; just that Tanenbaum and VrijeUniversiteit have not done work. One example of this is Fireball Amoeba by Fireball Software Distribution.

IV. ARCHITECTURE

Amoeba is designed as a collection of micro kernels. Thus the Amoeba system consists of many CPU's connected over a network. Each CPU owns his own local Memory in the range from 2MB to several 100MB. A huge number of Processor's build the so called Processorpool. This group of CPUs can be dynamically allocated as needed by the system and the users. Specialized servers, called Run server, distribute processes in a fair manner to these machines.
Many different Processor architectures are supported: i80386(Pentium), 68k, SPARC. Today, only the i80386 architecute is significant for building an Amoeba system (cheap!!!). Workstations allow the users to gain access to the Amoeba system. There is typically one workstation per user, and the workstation are mostly diskless; only a workstation kernel must be booted (from floppy, via tftp, burned in Flash-EEPROM). Amoeba supports X-Windows and UNIX-emulation. At heart of the Amoeba system are several specialized servers that carry out and synchronize the fundamental operations of the kernel. Amoeba has a directory server (called SOAP) that is the naming service for all objects used in the system. SOAP provides a way to assign ASCII names to an object so it's easier to manipulate(by humans). The directory server can replicate files without fearing their change. Amoeba has of course a file server (called the Bullet Server) that implements a stable high speed file service. High speed is achieved by using a large buffer cache. Since the files are first created in cache, and are only written to disk when they are closed, all the files can be stored contiguously. The underlying idea behind immutable files is to prevent the replication mechanism from undergoing race conditions. And file server crashes normally don't result in an inconsistent file system! The Bullet server uses the virtual disk server to perform I/O to disk, so it's possible that the file server run as a normal user program!
The Boot server controll all global system servers (outside the kernel): start, check and poll, restart if crashed. All Amoeba objects (files, programs, memory segments, servers) are protected anddiscribed with so called Capabilities. In Other words, Amoeba implements a universell distributed Client-Server-Modell. In fact, basically the whole system needs only three Functions to do all the work: The transaction call from the Client, and the GetRequest and PutReply functions on the Server side.

An Amoeba System consists of four principle components:

- Workstations
- Pool Processors
- Specialized Servers (File server...)
- Gateways



Objects

- Abstract data types with data and behaviors.
- Amoeba primarily supports software objects, but hardware objects also exist.
- Each object is managed by a server process to which RPCs can be sent. Each RPC specifies the object to be used, the operation to be performed, and any parameters to be passed.

Capabilities

- 128-bit value object description created and returned to the caller when the object is created.
- Capabilities are encrypted to prevent tampering.
- Subsequent operations on the object require the user to send its capability to the server to both specify the object and prove the user has permission to manipulate the object.

V. GOLES OF AMOEBA

Amoeba was designed with four main goals:

- Distribution – Connecting together many machines
- Parallelism – Allowing individual jobs to use multiple CPUs easily
- Transparency – Having the collection of computer act like a single system
- Performance – Achieving all of the above in an efficient manner

- Distribution: Connecting together many machines so that multiple independent users can work on different projects. The machines need not be of the same type, and may be spread around a building on a LAN.
- Parallelism: Allowing individual jobs to use multiple CPUs easily. For example, a branch and bound problem, such as the TSP, would be able to use tens or hundreds of CPUs. Or, one user playing chess where the CPUs evaluate different parts of the game tree.
- Transparency: Having the collection of computers act like a single system. So, the user should not log into a specific machine, but into the system as a whole.
- Performance: Achieving all of the above in an efficient manner. The basic communication mechanism should be optimized to allow messages to be sent and received with a minimum of delay. Also, large blocks of data should be moved from machine to machine at high bandwidth.

REFERENCES

[1] http://fsd-amoeba.sourceforge.net/amoeba.html
[2] http://fsd-amoeba.sourceforge.net/
[3] http://www.cdk5.net/oss/Ed2/Amoeba.pdf
[4] [Tanenbaum 1990] Tanenbaum, A.S., Renesse, R. van, Staveren, H. van., Sharp, G.J., Mullender, S.J., Jansen, A.J., and Rossum, G. van: "Experiences with the Amoeba Distributed Operating System," Commun. ACM, vol. 33, pp. 46-63, Dec. 1990
[5] Ramsay, M., Keigel, T., Memmer, H. "Ameoba Distributed Operating System" Online http://csserver.evansville.edu/~mr56/CS470/Final_Draft.pdf
[6] https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0CDsQFjAE&url=http%3A%2F%2Fwww.pcs.cnu.edu%2F~mzhang%2FCPSC450_550%2FCase%2520Study%2F2007%2FJamesSchultz_CaseStudy.doc&ei=6hIYVMXPLtP68QXW34FY&usg=AFQjCNE9LV0vyeCPDPVp-r53E1jnBpAoHg
[7] https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&cad=rja&uact=8&ved=0CEEQFjAF&url=http%3A%2F%2Fweb.iiit.ac.in%2F~kalyan_s%2Famoeba.ppt&ei=JBIYVNjVE8X_8QXa04LACQ&usg=AFQjCNHQSDcI9_MHf_LT8ndE-4e_Zko9iQ
[8] http://oai.cwi.nl/oai/asset/18386/18386A.pdf
[9] https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=17&cad=rja&uact=8&ved=0CEIQFjAGOAo&url=http%3A%2F%2Fwww.ics.uci.edu%2F~cs237%2Flectures%2Fold%2Famoeba-peter.ppt&ei=tBUYVKbeCoyk8AWgr4HgAQ&usg=AFQjCNFKsgVPqloIo8T66fWOibMi2qyWbw

