# **ADVANCE OPERATING SYSTEM**

# 

# **Name: M. Hassan Sattar**

# **Roll No: 22F-3773**

# **Submitted to: Dr. Bilal Khan**

# **National University of Computer and Emerging Sciences**

# **Microkernel on Real-Time Shared Memory in Operating System and transmission techniques**

Abstract:

IoT devices are increasingly using multi-core CPUs to suit their demand for advanced processing capabilities. Multi-core CPUs are not properly supported by existing iot systems. Inter-core interaction, often known as data transfer between cores, is frequently necessary for tasks. It has a big effect on the processor speed of multi-core operating systems. Here we are going to present a shared memory and signal transmission-based approach of inter-core communication. The microkernel operating system, called Mginkgo, we implement it and tested this microkernel in to this survey. According to the data averaging 74 cycles, is the time it takes to initiate an inter-core interrupt. The typical processing time for inter-core interrupts is 3157 cycles. In this study, we suggested a shared memory and signal transmission inter-core communication technique. which is flexible and different types of data can be transferred efficiently. This achieves very low latency with nearly minimal performance consumption and the overall system runs quite efficiently.

In order to address the demands of IoT devices for high processing capabilities, multi-core processors have started to be deployed frequently.

The microkernel architecture is a kernel structure that is rising in popularity. Many microkernel’s system services are user-mode processes. A microkernel OS supports the kernel's basic OS features. The basic functionality of the microkernel OS is assessed in this survey. Microkernel ensures the set of real-time scheduling policies. System calls reducing the complexity of the kernel and keeping the most fundamental resources through shared memory and address-mapping method to achieve inter-process communication. We conducted experiments our own microkernel called Mginkgo. According to the data averaging 74 cycles, is the time it takes to initiate an inter-core interrupt. The typical processing time for inter-core interrupts is 3157 cycles. In this study, we suggested a shared memory and signal transmission inter-core communication technique. which is flexible and different types of data can be transferred efficiently. This achieves very low latency with nearly minimal performance consumption and the overall system runs quite efficiently.

Introduction:

Single-core processors cannot satisfy the demands of the expanding market in terms of performance and power consumption due to the industry's ongoing growth in chip production.

Increasing the number of transistors on the chip boosts the processor's performance. however heat, switching frequency, and other factors will also affect the CPU. Therefore, multi-core processors come into scene.

As multicore technology is going advances, multicore CPUs are being used in embedded devices. Each CPU on a multiprocessor chip shares L2 cache, front-end bus, memory, and I/O devices, all of which are connected by a high-speed bus.

The high-speed bus allows CPU cores to communicate with one another, and speeds communication between parallel tasks.

There are now three major common kernel architecture: monolithic kernel, microkernel, and a hybrid kernel . The first kernel with a hierarchical structure to appear was the monolithic kernel. The monolithic kernel exposed several points of vulnerability and had poor security since it ran a lot of extra service operations in kernel space. the monolithic kernel includes too much unpruneable code, making it challenging to follow to the demands of utilizing relatively little resources on embedded devices. The microkernel only offers the most fundamental kernel services as a means of overcoming the drawbacks of the monolithic kernel. Microkernels vastly minimize the number of kernel coding and security issues. Currently, Android, iOS, and Windows are the most popular multi-core operating systems for embedded devices.

The majority of the kernels in these operating systems have monolithic kernel designs because they are intended to be desktop operating systems. The quantity of kernel modules and code constantly grows as the functionality of these kernels continues to advance.

For embedded platform, the microkernel is ideal. Because of its secure and lightweight design, the microkernel regard as the most popular operating system for embedded computers in the future. However, the majority of microkernel operating systems do not offer enough support for multi-core processor scenarios. Therefore, future growth of the embedded area depends on research on multi-core support for microkernel operating systems. A multi-core operating system's performance can be influenced by a variety of other factors, it does not simply get better as the number of cores increases. The architecture of inter-core communication, which is how data is transferred between various cores, is a crucial aspect that determines how well a multi-core operating system performs.

**Multi-Core Operating System**

The first level of software installed on top of the hardware is the operating system. There are two primary categories of multi-core operating systems today:

Symmetric multi-processing (SMP) and asymmetric multi-processing (AMP) modes

In the AMP approach, the entire system is running multiple Operating system instances on one or more CPU cores. These operating Operating system instances each have their own unique independent resources, including memory, peripherals, etc.

Typically, the user sets up these resources during the boot process, and they are not modified once the computer has started. Because each of these Operating system instances has autonomous resources of its own, they communicate with one another by sharing a block of memory.

In the SMP approach . All of the multicore processor's resources, including memory and peripherals, are managed by a single Operating system instance, which is the sole Operating system instance in the entire system.

In this approach, a design concept like that of homogeneous multi-core CPUs is utilized, i.e.

The processing cores are merely utilised as a computational resource and are only required to complete the tasks that have been allocated to them. The cores are treated as equals and

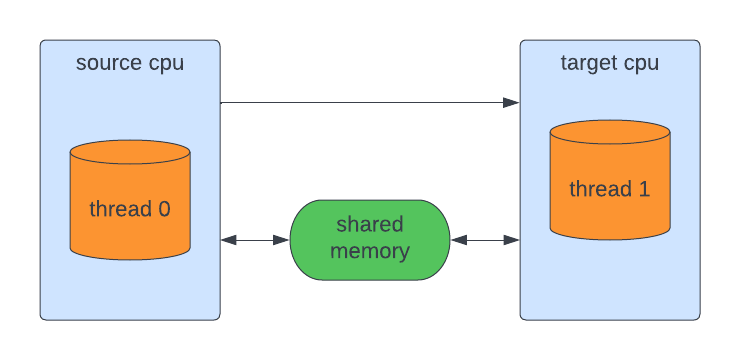
exchange resources

SMP mode operating systems, as opposed to AMP mode, have the benefits of a simple structure, high communication speed, and flexible load balancing, allowing them to more effectively utilise the performance and power benefits of multi-core processing platforms. SMP mode may be supported by a wide variety of operating systems, including Linux, Windows.

Because of its single format, the SMP mode is better suited for embedded systems. It can benefit from the power and performance of modern multi-core CPUs.In this survey i observe to create an SMP-compatible inter-core communication system. This mechanism's design fully utilises the benefits of resource sharing and core equality in SMP mode.

**Methadology**

In this work, we propose an inter-core communication mechanism based on signal transmission and shared memory. We implement this mechanism on the i.MX6Q chip platform on our own microkernel operatingsystem named Mginkgo. We also design some test cases to test this mechanism. The test results show that the average time to trigger an inter-core interrupt is about 0.093 microseconds. The average time to process an inter-core interrupt is about 3.986 microseconds. And the communication time of multi-core IPC is about 18us. which is the same as that of single-core IPC.



**Mginkgo Microkernel**

**A group of developer developed the Mginkgo microkernel based on the third-generation microkernel seL4, which was created to address the industry's growing demands for operating systems for industrial control in terms of security, real-time, and dependability.**

**The  task management, memory management, IPC module,  interrupt management and capability subsystem are the five primary functional modules that make up the core functional modules of the microkernel.**

**Result:**

colusion