ChipOS based Grid Computing

Jijun MA
Computer Science College
Zhejiang University
Hangzhou, Zhejiang
CHINA

Man Cao
Computer Science College
Zhejiang University
Hangzhou, Zhejiang
CHINA

Wei Ma
Computer Science College
Zhejiang University
Hangzhou, Zhejiang
CHINA

Tianzhou Chen
Computer Science College
Zhejiang University
Hangzhou, Zhejiang
CHINA

Abstract: Grid computing technology has become the most popular and mature way of distributed computing recently. At the same time, embedded system and System-on-Chip(SoC) architecture are becoming commonly used in calculation area, because of the low design cost and extensible architecture. These new technology provides a noval way to construct grid computing infrustructure with embedded system on SoC. This paper presents a dual OS architcture to establish a grid computing infrustructure, with a on-chip OS named ChipOS to provide grid computing tasks dispatching and the general purpose operating system to deal with normal computing tasks. This dual OS grid computing architecture is proved an effective task dispatching platform for grid computing.

Key–Words: Grid Computing, System-on-Chip, Embedded System, Dual OS, Scratch Pad Memory

1 Introduction

Grid computing technology has become the most popular and mature way of distributed computing. Computational Grids enable the sharing, selection, and aggregation of a wide variety of geographically distributed computational resources, such as supercomputers, computer clusters, storage systems, and data sources. The grid computing technology presents all these resources as a single, unified resource for solving large-scale compute and data intensive computing applications (e.g, molecular modeling for drug design, brain activity analysis, and high energy physics).

The modern grid architecture is often regarded as a three-layer (sometimes four-layer) model. In both case the key to success of grid is the middleware, the software that organizes and integrates the disparate computational facilities belonging to a Grid, which lies between OS and grid application.

Embedded systems have become a large part of computational resources recently, but they are seldom used in grid computation. With the development and improvement of the computing power of embedded hardware, especially the emergence of system-on-chip(SoC) technology in embedded domain, it is more and more possible to use embedded systems in grid computing architecture.

SoC is an idea of integrating all components of a computer or other electronic system into a single integrated circuit(chip). A typical SoC chip consists of the following components: one or more microprocessor cores, memory devices such as ROM, RAM and Flash, reconfigurable components such as FPGA, etc.[1]

A typical SoC architecture with reconfigurable logic and multiple processing cores can share a common memory hierarchy such as L2 cache and memory. The on-chip memory called Scratch Pad SRAM(SPM) can be accessed by these cores. SPM is a kind of memory device with the characteristics of high performance of reading-writing speed and low power consumption. Generally, the size of SPM is 256 KB to 768 KB.[2] [3]

As the increasing needs of the Grid Computing, SoC technology provides a method to conquer the limitation of the current Grid architecture. Since the advantage of on-chip memory SPM, we describe an architecture of Grid computing based on the SoC of dual OS.

Since middleware is pure software above OS, the performance of computational grid is somehow affected by the inefficiency of middleware, especially if the computational resource is small-scaled such as embedded system. There is possibility to improve the performance if we embed some basic function of middleware into lower level.

Thus this paper suggests a ChipOS based grid computing framework to make use of SoC architecture and describes how to construct the grid computing architecture base on the SoC with dual OS of a ChipOS and a GPOS.

The rest of this paper describes the design and implementation of the ChipOS based grid computing framework. Section 2 describes the design of chip os,

and the grid computing related works. The coordination of ChipOS and GPOS is described in Section 3. Section 4 is about the implementation of our grid computing architecture. Section 5 offers the conclusion of our research and future work.

2 Related Works

The ChipOS is a small operating system for SoC architecture developed by the research group of authors' laboratory. The basic system contains five modules, task management, memory management, network communication and interrupt routines. The task management is the main work of the task management module. It is designed to be able to schedule tasks by different policies, such as FCFS(first come, first serve) or SJF(shortest job first). The memory management splits the memory into three type of regions – kernel region, real-time region and user region – for different type of processes.

It is important to mention how is the two operating system loaded into the board. ChipOS is firstly loaded into the on-chip memory and then the ChipOS loads the GPOS into off-chip memory. At last, after the loading process, the ChipOS establishes the infrastructure for both OS running on-chip or off-chip.

The jobs of ChipOS and GPOS are divided into several parts according their properties.

The jobs of the GPOS:

1. Schedules the processes arrived to GPOS and dispatches them between GPOS and ChipOS.
2. Manages off-chip resources, such as SDRAM management, off-chip hardware driver management.
3. Provides the basic infrastructure for applications execution environment.

The jobs of the ChipOS:

1. Manages on-chip memory SPM.
2. Provides the real-time API for applications dispatched to ChipOS by GPOS.
3. Manages on-chip computing resources such as on-chip process units and reconfigurable logic.

The dual OS architecture containing ChipOS and GPOS is an extensive architecture for several uses. This architecture is used in the following applications:

1. ChipOS can be used as a system monitor to monitor the GPOS's resources usage, such as CPU usage, memory usage, and file system usage. It will make use of the profile to figure out execution policies for GPOS.
2. In security domain, ChipOS is used as a platform for anti-virus program to do virus detection works on GPOS.
3. The dual OS architecture can be also used in computation-intensive system, where the ChipOS can monitor the work overload of the GPOS and schedule the tasks between several parallel computer nodes.

A simple architectural description about modern computational grid is given here. There are three necessary components(layers) to form a grid:

1. Grid fabric:

   Fabric is the interfaces to local control[4]. This is the lowest level that provides the access to the actual resources.

2. Grid middleware:

   Middleware is the core component of the Grid architecture, which provides communication protocols, authentication protocols, resource discovery and management, storage access, task scheduling and other services necessary to build a grid[4][6]. Sometimes this layer is divided into two layers: resources and connectivity protocols, and collective services[4].

3. Grid applications:

   This is the topmost layer which compromises user applications. Applications are constructed in terms of services defined at middleware layer. They often needs access to remote processing element and resources. An application programmer will see APIs defined to exchange protocol messages with the appropriate service(s) to perform desired actions[4].

Most of the research on embedded system with grid computing is to utilize the portability of embedded system, such as collecting and sending information. Few have thought of embedded system as a sort of computing resources in grid technology.

Danny Hughes, Phil Greenwood et al. [7] proposed a novel Grid-based approach to supporting flood prediction through the use of embedded sensor nodes equipped with wireless networking technology.

Chen-Khong Tham, Rajkumar Buyya et al. [8] discussed the integration of sensor networks and grid computing in sensor-grid computing, which enables the construction of real-time models and databases of the environment and physical processes as they unfold, from which high-value computations like decision-making, analytics, data mining, optimization

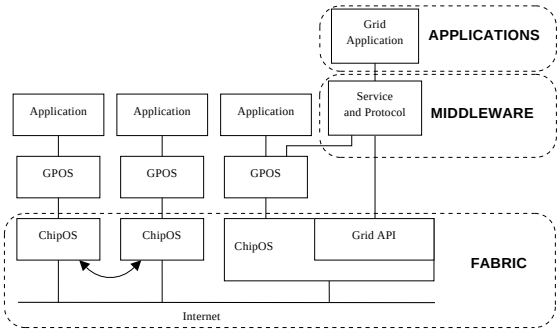

Figure 1: Layers of Grid SoC and Grid Network

and prediction can be carried out to generate 'on-the-fly' results.

Embedded systems currently used in grid computing are just a kind of real-time infomation collection interfaces. The full computing resource and the effective manipulation of low level hardware are not digged out from embedded domain. Thus, it is possile to use the ChipOS based grid framework to full utilize the ever-growing embedded system resources.

3 ChipOS based Grid Framework

3.1 Architecture of the ChipOS based Grid

The on-chip SPM in SoC architecture has fast write and read speed compare to off-chip memory. The CPU cycles for accessing SPM data are only about 3 cycles. This proved to be an effective memory usage for applications. The basic tasks – package dispatching, connectivity check and transportation – can be done with the ChipOS for the sake of efficiency. Thus, this paper introduces the ChipOS based grid architecture to make use of the effective SoC hardware.

The three layers of Grid SoC is depicted in figure 1. Grid SoC is a three-layer model. Fabric is the Internet, ChipOS and the processors. Some basic function of middleware is provided by Grid API, which is part of ChipOS. The nodes in grid network are same in structure but different in function during the process of running a grid application.

The change between traditional SoC and Grid SoC is that we integrate basic function of Grid middleware directly into the ChipOS, so that we can take advantage of the SoC. ChipOS, as part of the fabric layer, is in charge of some basic functions of the middleware. These mainly include package management, connectivity check and transportation, and necessary communication among ChipOSes. Other functions of middleware are implemented by the service and protocols above GPOS.

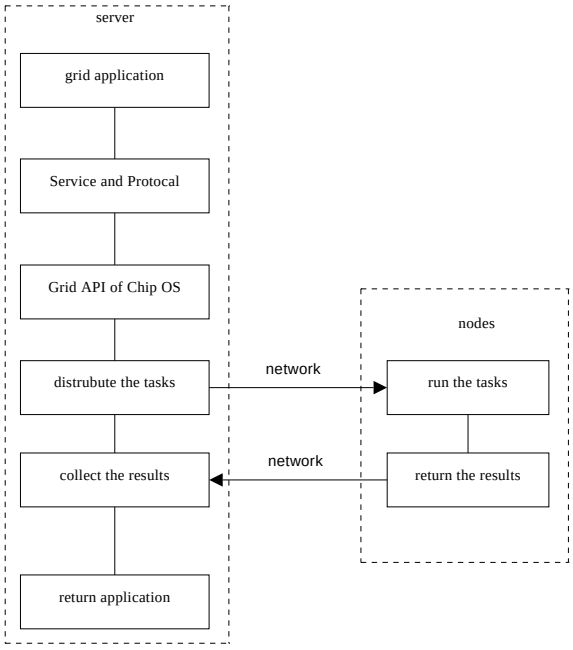

Figure 2: Network Architecture,Communications and Task distribution of Our Grid Computing

Figure 1 depicts three ChipOSes. In the process of running a grid application, we regard the one with Grid API on the right as server node, the other two as ordinary nodes.

The basic function of the server node is to divide the grid application into small parts and distribute to other ordinary nodes in the grid. The function of the ordinary nodes is to run and coordinate the given task, and communicate with each other. The logical connections between ChipOSes actually are the physical Internet connections. All the nodes can simultaneously run the non-grid applications above GPOS.

In fact, all the nodes in the grid are the same in structure and in ChipOS. They all have Grid API, and necessary grid service and protocols above GPOS. This means that any node can be the server node. The words "server" and "ordinary" are only to distinguish the different roles of nodes in the process of running a single grid application.

3.2 Task Processing

Figure 2 depicts the network architecture, communications and task distribution in grid computing.

When ChipOS receives a network package, it first tests the package to see whether it is a grid package. If it is, ChipOS sends the package to ChipOS network interface, loads balance through the ChipOS grid API, and runs the grid task. Otherwise, it delivers the package to GPOS network driver, pushes into the network

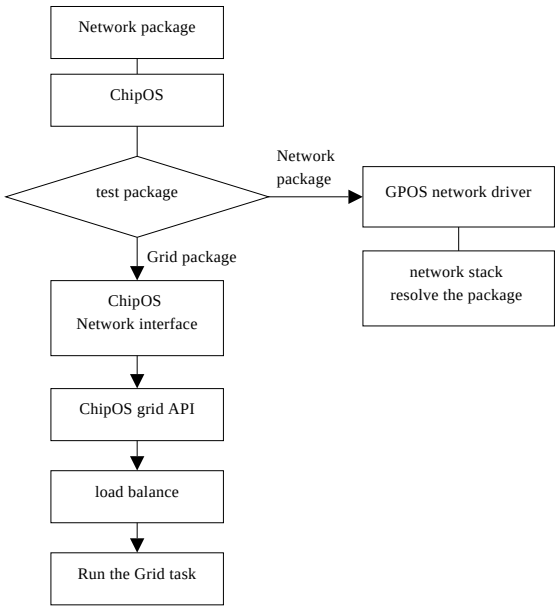

Figure 3: Task distribution process of Our Grid Computing

stack and then let GPOS resolve the package.

Figure 3 describes task distribution process of our grid computing.

The task distribution process is described as following:

1. The middleware of grid application running on the server's GPOS calls the grid API to notify the ChipOS on the server.

2. When ChipOS on the server is ready, it starts to collect the data from grid application, and distributes the task to various nodes on the internet through the task-distribution modularity. Task scheduling and load balance is also needed here.

3. The result is returned to the server when each node complete it's task. The server is responsible for collecting the results.

4. Server returns the result to the grid application.

4 Implementation of ChipOS based Grid Computing

The Intel XScale PXA is used as the SoC architecture in our research. Intel PXA27x processor is the product of Intel Company and it is an integrated system-on-a-chip microprocessor designed for mobile devices. The frequency of PXA27x development board is 520

Table 1: Extra cache mapped to registers

Address	Name	Description
0x58000000 - 0x58FFFFFC	–	reserved
0x5C000000 - 0x5C00FFFC	Memory Bank0	64-KByte SRAM
0x5C010000 - 0x5C01FFFC	Memory Bank1	64-KByte SRAM
0x5C020000 - 0x5C02FFFC	Memory Bank2	64-KByte SRAM
0x5C030000 - 0x5C03FFFC	Memory Bank3	64-KByte SRAM
0x5C040000 - 0x5C7FFFFC	–	reserved
0x5C800000 - 0x5FFFFFFC	–	reserved

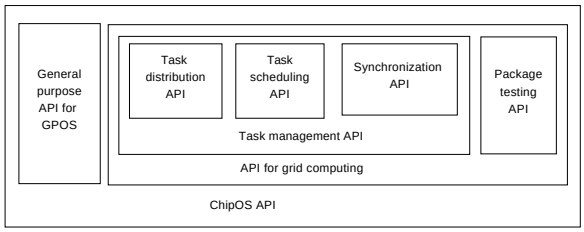

Figure 4: The componets of ChipOS API

HZ. PXA27x provides extra 256K cache which is considered as internal memory. This internal memory-mapped SRAM consists of four banks with the capacity of 64K. The SRAM array module consists of four banks of 8K x 64bit memory arrays. Each memory bank has a dedicated single-entry queue and 8K x 64 bits for data storage. If a memory bank is in standby mode, the access request is stored in the queue while the memory bank is placed in run mode. The access is completed when the memory bank has entered run mode. If a memory bank is in run mode and the queue does not contain any pending access requests, the queue is bypassed and the memory is accessed normally. Table 1 describes the division of SRAM.

Package dispatching, connectivity check and transportation of the computation module is movde from middle ware of grid computing architecture into ChipOS through the invokong of ChipOS API.

4.1 ChipOS API

The architecture of our ChipOS API is indicated in figure 4.

GPOS and the middleware all communicate with ChipOS through ChipOS API. So first of all,we must design and implement a series of ChipOS APIs to pro-

vide this function. The ChipOS API are divided into two parts:

1. General purpose API, used by the GPOS to perform normal tasks.
2. Grid API, called by grid applications and is responsible for the whole process of grid computation.

We pay more attention to the GPOS API since it is the base of our grid architecture.The upper level GPOS and grid applications all need these APIs to complete their task. We also divide them into different parts via thier different functions.Here are some more detailed descriptions of each part:

1. Task management API

   These API are designed to perform task distribution, task scheduling and synchronization between nodes.

2. Task distribution API

   The server node calls task distribution API to do the task distribution.

3. Task scheduling API

   The server node calls task scheduling API to do the task scheduling.

4. Synchronization API

   The nodes on the system calls synchronization API to communicate with other nodes,to exchanges data,information.

The ChipOS is responsible for handling the incoming network packages. There are two categories of package, one for GPOS applications while the other for grid applications. We modify the network protocol stack of our ChipOS, make it able to detect which category of an incoming network package belongs to. We also add several bytes of identification in the network packages of grid application. Thus before delivering each package to the upper applications, ChipOS must look at the package header and then decide where to send package.

4.2 Design of Grid Computing

An ideal grid environment requires many features to be realized. These include secure access to resources, single resources sharing and multiple resources coordinating. Due to the limited capacity of ChipOS, we cannot provide these functions in ChipOS. The services and protocols above GPOS handle these issues.

Table 2: Basic data of ChipOS

ChipOS load time	3 ms
ChipOS kernel size	8K
switch time	4.307 μs
watchdog code size	390 lines of arm assembly

Instead, ChipOS Grid API helps the middleware to complete its tasks. Package testing API processes part of the metadata in a network package, task management API directly deal with the basic function of task distribution and coordination. The communication among ChipOSes is also done by API. In essence, ChipOS Grid API offers great convenience to middleware.

We also modify the GPOS to make it fully cooperate with our grid architecture. The middleware in the upper level is running on GPOS, since in our architecture it sometimes need to call the ChipOS API directly, the GPOS must provide such interfaces that middleware can directly call the ChipOS API safely and efficiently.

5 Result and Future works

As it is shown in table 2, the size of ChipOS image is 8K; the load time of ChipOS is less then 3 ms and the average task switch time is 4.307 μs and the code size of the switch routine is 390 lines of arm assembly language. As the swtich procedure involves registers modification and memory accesses, there are some degree of latency. It proves that the The ChipOS based gird computing makes use of the effective hardware resources to provide improved coputing power. With the ever-growing embedded system domain, more and more computing resources of embedded system can be used in scientific, engineering and society grid computing projects. This architecture will be used and improved in near future.

Acknowledgements: The research was supported by the National Natural Science Foundation of China under Grant No. 60673149 and the National High-Tech Research and Development Plan of China 'Hardware-software Codesign of virtual FPGA' No. 2007AA01Z105.

References:

[1] III Mooney, V.J. and D.M. Blough, A hardware-software real-time operating system framework for SoCs. *Design & Test of Computers, IEEE*. Vol. 19, No. 6, Pages 44–51, 2002

[2] O. Ozturk, M. Kandemir, and I. Kolcu. Shared scratch-pad memory space management. In Quality Electronic Design, 2006. ISQED '06. 7th International

[3] Banakar, R. and Steinke, S. and Bo-Sik Lee and Balakrishnan, M. and Marwedel, P. Scratch-pad memory: a design alternative for cache on-chip memory in embedded systems. *Hardware/Software Codesign, 2002. CODES 2002. Proceedings of the Tenth International Symposium on*, Pages 73–78 6-8 May, 2002

[4] Ian Foster, Carl Kesselman, Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, Vol. 15, No. 3, 200-222, 2001.

[5] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002

[6] Mark Baker1, Rajkumar Buyya, Domenico Laforenza. Grids and Grid technologies for wide-area distributed computing. John Wiley & Sons, Ltd, 2002

[7] Danny Hughes, Phil Greenwood, Geoff Coulson, Gordon Blair. GridStix: Supporting Flood Prediction using Embedded Hardware and Next Generation Grid Middleware. *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, 2006

[8] Chen-Khong Tham,Rajkumar Buyya. SensorGrid: Integrating Sensor Networks and Grid Computing. *CSI Communications*, 2005 Our architectural description

[9] Open Grid Forum, www.ogf.org

[10] Grid Computing Info Centre, www.gridcomputing.com