第 41 卷 第 6A 期
2014 年 6 月

计 算 机 科 学
Computer Science

Vol. 41 No. 6A
June 2014

基于 MIC 平台的 offload 并行方法研究

沈 铂 张广勇 吴韶华 卢晓伟 张 清
(浪潮集团高效能服务器和存储技术国家重点实验室 济南 250101)

摘 要 随着并行计算的快速发展,开始出现了专用于并行计算加速的协处理器。通过对新推出的 MIC 架构协处理器的编程模式研究,描述了 MIC 平台下的应用模式,研究了不同应用模式各自的特点和适用范围,并深入研究了 offload 编程模式,提出了基于 MIC 平台 offload 编程模式的开发策略,为快速、高效地实现 MIC 并行程序的移植和加速提供了解决思路。

关键词 并行计算,MIC,offload 模式,协处理器
中图法分类号 TP391 **文献标识码** A

Research of Offload Parallel Method Based on MIC Platform

SHEN Bo ZHANG Guang-yong WU Shao-hua LU Xiao-wei ZHANG Qing
(National Key Laboratory for High-efficient Server and Storage Technology, Jinan 250101, China)

Abstract With the development of parallel computing, co-processor specializing in parallel computing appeared. Relevant knowledge about application model and offload programming method in the platform of MIC will be introduced, at the same time, different characteristics and environment according to different application model will be discussed. Above all, development strategy based on offload programming method is proposed, which can implement the transplanting and accelerating of MIC parallel program rapidly and effectively.

Keywords Parallel computing, Many integrated core, Offload mode, Co-processor

1 引言

近几年来,随着 CPU 制造工艺已经接近极限[1],频率难以进一步提升,制造厂商的策略转为在芯片中集成越来越多的计算核心。在高性能计算领域已有一定发展的并行计算应用随着多核 CPU 的普及而更具规模[2]。然而,人类对计算的需求是无止境的[3],CPU 的核心数从硬件上制约了应用的并发数量,严重限制了并行计算的效率[4],众核处理器的概念应运而生。众核处理器是指拥有众多计算核心的处理器,以协处理器方式存在。众核处理器单核心计算能力弱于 CPU,但并发能力远高于 CPU。一些图形处理芯片厂商发现,GPU 的硬件特性使之非常适合高并发度的并行计算[5]。因此 GPGPU 以协处理器的方式,为并行计算提供了重要的手段。然而,GPGPU 的架构和指令集与 CPU 完全不同,使用其进行并行计算时,需要对源代码进行大幅改写[6]。

2012 年底,Intel 公司推出基于 MIC(Many Integrated Core,集成众核)架构的 Intel Xeon Phi 协处理器,该协处理器针对并行计算而设计,主要特点为:形式为标准 PCI-E 接口的板卡;支持基本 x86-64 指令集和扩展的 512 位向量化指令,最大限度地兼容原有 CPU 代码,且优化方式与 CPU 多线程程序类似;拥有独立的内存、电源等设备;拥有 50 个以上核心,其中每个核心 4 个硬件线程,硬件线程有自己的硬件线程单元,性能优于超线程,为高并发应用场景提供了硬件支持;双精度性能达到 1TFlops 以上;拥有基于 Linux 的片上操作系统,提供 Linux 系统指令支持以及虚拟的以太网支持,MIC 架构设备拥有自己的 IP 地址,可以被视作网络中的一个节点等。与 GPGPU 不同的是,Intel Xeon Phi 以协处理器方式与 CPU 紧密结合,可以在不改动源代码或源代码改动很小的情况下使用该协处理器进行并行计算[7]。

MIC 架构推出时间还比较短,目前对其开发策略的介绍相对较少。本文首先描述了 MIC 应用模式并对各种模式进行分析和比较,提出了各个模式的适用范围;随后对 offload 编程模式的特点和语法进行了详细的说明,并提出了 offload 模式的开发策略,最后给出了结论。

2 MIC 应用模式分析与比较

MIC 架构协处理器由于既有协处理器的特点,又拥有独立的操作系统,因此拥有多种应用模式,如图 1 所示。

根据程序运行设备的不同,MIC 应用模式从左至右分为 5 种:CPU native 模式、MIC offload 模式、对等模式、CPU 协处理器模式和 MIC native 模式。

本文受国家 863 项目:浪潮亿级并发云服务器系统研制(2013AA01A208)资助。
沈 铂(1986—),男,工程师,主要研究领域为高性能计算,E-mail:shenbo@inspur.com;张广勇(1985—),男,硕士,工程师,主要研究领域为高性能并行计算;吴韶华(1982—),男,博士,工程师,主要研究领域为计算流体力学与高性能计算;卢晓伟(1984—),男,硕士,工程师,主要研究领域为高性能运算与数据流挖掘;张 清(1981—),男,硕士,工程师,主要研究领域为高性能计算。

1. CPU native 模式

CPU native 模式是仅使用 CPU 进行计算的模式，没有用到 MIC 协处理器设备。

2. MIC offload 模式

MIC offload 模式简称 offload 模式，即传统的协处理器模式，是通过改写源代码，使驱动程序能够将部分代码或函数卸载(offload)到协处理器中进行计算的模式。主函数及 MPI 通信函数(如果有)只存在于 CPU 端。其是 MIC 架构中最常用的模式之一。

3. 对等模式

对等模式是指将协处理器视为网络中的节点，并使用 MPI 或其它网络通信方式运行网络程序。因为 Intel Xeon Phi 协处理器拥有自己的操作系统，虚拟了以太网卡等网络设备，并实现了 TCP/IP 协议栈，因此可以将协处理器视作网络中的节点，通过 IP 地址与之通信。在 CPU 端和协处理器端分别运行一个或多个程序进程，以网络的方式进行进程间通信。

4. CPU offload 模式

与 MIC offload 模式相反，这是一种以 MIC 架构设备为主、以 CPU 为协处理器的模式。但是这种模式仅作为理论上的一种方式，目前还无法实际运行。

5. MIC native 模式

MIC native 模式简称 native 模式，与 CPU native 模式相反，是指程序仅运行在 MIC 架构协处理器上的模式。与对等模式类似，MIC native 模式的区别仅在于这种模式运行的是不涉及网络通信的单机程序。

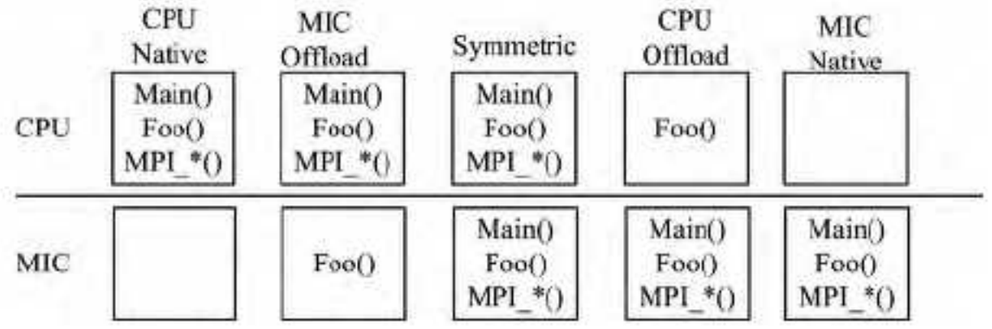

图 1 MIC 应用模式[8]

上述 5 种模式为 MIC 架构的应用模式，其中 CPU native 模式不涉及 MIC 架构协处理器，CPU offload 模式尚不能实际应用，其余 3 种模式中，native 模式与对等模式类似，区别在于 native 模式为对等模式的一个单节点、运行于 MIC 卡上的特例，因此有时也将二者统称为 native 模式。

综上，MIC 架构的应用模式常用的有 offload 模式和 native 模式两种。其中 native 模式无需修改代码，运行性能受代码可并行部分所占比例影响较大，适用于快速移植开发的场合。offload 模式更符合传统协处理器应用方式，可以指定部分代码使用协处理器，因而可以针对不同代码段，选择最适合的设备(CPU 或 MIC)、最适合的并行线程模型和线程数，并进行针对性的优化；CPU 与 MIC 之间采用线程间或线程内异步，充分利用计算资源，对代码改动量较小，且 CPU 和 MIC 版本之间共享源代码，减少了维护成本。

3 offload 编程模式

3.1 模式特点

offload 编程模式与传统协处理器编程模式类似，应用程序整体在主机端运行，内存控制、设备控制等功能在主机端控制。当遇到可以并行的热点函数时，在源代码中显式指定，通过驱动程序自动加载到协处理器中运行，并将结果传回主机端内存。一个典型的流程如图 2 所示。

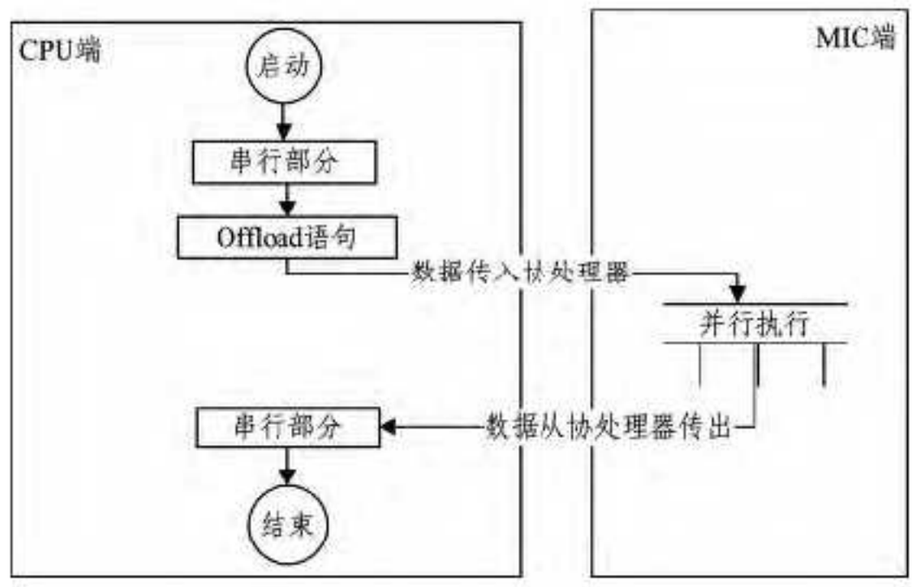

图 2 offload 流程图

应用程序的主要运行和控制仍在主机端，仅有热点部分被卸载(offload)到协处理器中，协处理器协助完成计算任务，一般不负责逻辑处理、设备管理等任务。

offload 模式有以下特点：

1)遵循传统协处理器模式，热点代码段被 offload 到协处理器中运行，MIC 协处理器负责计算，主机端负责流程控制。

2)数据在主机端和协处理器端之间的传输需要显式指定，但内存空间的开辟与控制不需要显式说明。

3)调用协处理器的时机需要显式指定。

4)程序代码需要做相应修改和添加，但对原始程序的改动较小。

5)不需要特殊编译选项。

3.2 编程与编译方式

offload 模式的编程语言支持 C/C++和 Fortran，并在其上扩展了编译指导语句，因此可以简单地在原有 CPU 程序的基础上添加编译指导语句，使程序使用 offload 模式运行。

例如，原始程序为：

```
for(int i=0;i<LEN;++i)
    a[i]=b[i]+c[i]
```

将其改写为 offload 模式为[9]：

```
# pragma offload target(mic)
    for(int i=0;i<LEN;++i)
      a[i]=b[i]+c[i]
```

offload 模式本身不提供并行编程模型，但 MIC 协处理器支持常见的并行编程模型，如 OpenMP、pThread、MPI 等，因此，在实际应用当中，需要将 offload 模型与某一种或多种编程模型结合使用，才能发挥 MIC 协处理器的性能优势。例如上例结合 OpenMP 改写为：

```
# pragma offload target(mic)
# pragma omp parallel for
for(int i=0;i<LEN;++i)
    a[i]=b[i]+c[i]
```

offload 模式的应用程序编译时不需要使用特殊的编译选项，只需要使用支持 MIC 架构的编译器进行编译即可，如果需要特殊的编译或优化手段，需参考相应编译器的手册。

offload 模式虽然语法相对简单，但在开发优化过程中，

与现有协处理器开发优化过程有一些不同之处。如果没有正确的开发方式，难免事倍功半，难以获得理想的性能。

4 offload 模式开发策略

基于 MIC 平台的 offload 编程模式的开发策略与基于加速器 GPU 的并行计算开发策略类似，主要选择并发度高的热点函数或代码段进行移植开发。通过使用编译指导语句的方式，指示代码段使用 MIC 协处理器运行，并通过传统的并行方式如 OpenMP 进行并行。

基于 MIC 平台的 offload 编程模式的开发策略如图 3 所示。

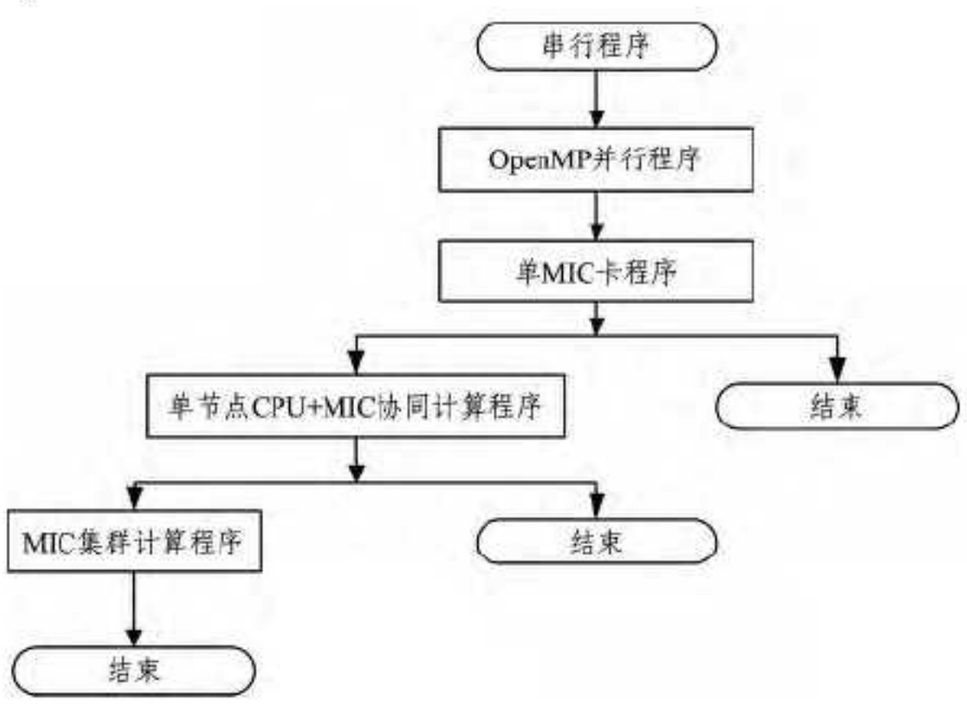

图 3 offload 编程模式开发策略

移植工作的基础是 CPU 版本的并行程序，本文采用 OpenMP 并行模型进行并行化，实际应用中可以使用 MIC 平台支持的任何并行编程模型。

1)热点分析

对原始程序进行热点分析，通过性能分析工具或手工添加时间函数的方式，检测程序中函数或代码段的运行时间，找到程序中耗时最多的一个或多个部分作为优化工作的重点。

2)热点部分算法分析

对热点部分进行算法分析，确定是否可以并行化，以及是否适合并行化。如果原始程序已是并行版本，则无需进行此步。

3)确定 MIC 平台内核用到的数组

根据对串行程序的分析，确定哪些模块需要移植到 MIC 平台上运行，对需要运行在 MIC 平台上的代码内的数据进行分析，确定数组在代码中的什么位置传递到 MIC 上，传递的方向是 in、out 还是 inout，以及每次传递时的大小等信息。

4)开发 CPU 端串行程序

与其它协处理器开发策略不同，MIC 平台的程序基于 CPU 平台程序，MIC 版本的代码可以直接用于 CPU 端。由于 CPU 端工具众多，且用户通常具有丰富的经验，因此使用传统 CPU 平台对程序进行开发、优化，再将 CPU 程序移植到 MIC 平台，可以取得事半功倍的效果。

一些 CPU 端程序从算法层面分析可以并行，但是在编写代码的过程中产生了一定的依赖，导致程序无法并行，甚至有些程序，串行版本的优化导致了一定的依赖。因此首先要对这种情况的代码进行修改，使程序能够尽可能地提高并发度。有些情况下，程序选择的算法无法并行，但存在可以替代的并行算法，此时需要实现可以等价替代的并行算法并替换串行程序。

5)CPU 端串行程序数组修改

根据步骤 3)中确定的数组，对需要修改的数组进行相应的调整。例如行列置换、多个数组合并、分割等，为移植到 MIC 上做先期准备。

6)CPU 端数据分块设计

由于 MIC 端卡上内存空间相对 CPU 端非常有限，因此需要针对这一特点，对程序可能出现的运行内存不足的情况进行针对性的前瞻设计，即对任务数据进行分块处理，使得一次处理的任务所需要的内存容量小于 MIC 平台所支持的运行内存容量。

7)CPU 端并行化

将根据上述步骤得到的串行程序进行并行化。通常采用 OpenMP 进行并行化，根据情况，也可以使用 pThread 等 MIC 平台支持的方式进行并行化。并行化后的程序需要进行全面测试以保证正确性。

在并行代码段的选择上，需要充分考虑 MIC 平台的特殊性，CPU 平台只有数个、至多数十个并发线程，但 MIC 平台需要数百个并发线程，因此选择并行代码段需要考虑该段代码是否支持大规模的并行，以及代码段的线程扩展性是否良好。对线程数的设定应采用 OpenMP 默认方式，或使用变量指定，并对 CPU 平台和 MIC 平台做出判断和区分，以保证灵活性和可移植性。

8)CPU 程序优化

因为 CPU 与 MIC 架构类似，且可以共享 MIC 版本的源码，因此对 CPU 平台程序的优化通常同样会对 MIC 平台程序产生影响。优化主要在以下方面进行工作：

a)并发度与向量化

MIC 平台的主要计算能力优势在于高并发线程和长向量化单元，对二者的利用程度直接决定了程序在 MIC 平台的计算性能。

MIC 平台支持同时并发数百个线程，并发度与 CPU 相比提高了一个数量级。因此需要尽量提高程序并发度，以满足 MIC 处理单元的计算能力。但是提高程序并发度，通常对 CPU 版本程序不会有较大的性能影响。

向量化从某种意义上说，是另一种并发处理数据的方式。MIC 平台拥有较宽的向量化处理单元，如果不能达到较好的向量化率，会对程序性能产生较大影响。提高向量化效率通常需要尽量保证数据的连续内存访问，以及尽量充分利用向量化处理单元的长度。由于 CPU 端向量化宽度较 MIC 端小，因此以 MIC 平台向量化宽度进行优化，可能对 CPU 程序没有性能提升，但是在绝大多数情况下，并不会导致性能下降。

并发度和向量化有一定重叠，部分情况下有一定制约。由于向量化仅能在最内层循环进行，因此需要调整循环嵌套方式，以及内外层循环的次数和顺序，以达到向量化和并发度的平衡。

需要注意的是，优化时应避免与硬件强相关和汇编指令级优化，例如，由于 CPU 与 MIC 向量化处理单元的宽度不

同,如果以确定的 CPU 的向量化宽度进行优化,则可能对 MIC 端程序性能产生负面影响。

b)访存方式

访问 cache 的速度远大于内存,因此需要利用局部性原理,尽量集中处理相关的数据。顺序访问的速度远大于随机访问,如果不能满足顺序访问,需要调整访问顺序或数据排布顺序,尽量达成顺序访问。计算与访存交叉,运行时间可以互相掩盖,减少总体运行时间。

9)开发 MIC 端并行版本

基于 CPU 端并行程序,开发 MIC 端并行版本。在需要移植到 MIC 端的代码段前添加 offload 语句,表示将代码段卸载到 MIC 协处理器中运行,并使用 in、out、inout 标示传输的数据、传输的长度、传输的方向,对 offload 代码中用到的函数使用__attribute__((target(mic)))进行标识。由于篇幅有限,具体语法请参照相关文档。

在此版本的测试过程中,由于基础版本已在 CPU 端通过测试,因此此处测试应偏重于 offload 语句以及传输的正确性。通过先期在 CPU 端的开发和测试,减少了此处的工作量,避免了使用协处理器调试程序的困难,提高了工作效率。

10)优化 MIC 端并行版本

MIC 设备与 CPU 设备虽然相似,但仍有一定的不同之处,因此 MIC 平台程序需要进行针对性的优化。主要优化点有:

a)并行线程数调整

根据实际测试情况,选择合适的并行线程数,由于 MIC 平台硬件设计,理论上每核心至少运行 2 个线程,通常将并行线程数设置为 MIC 每核心 3 或 4 个线程。

b)向量化调整

由于 MIC 平台向量化长度大于 CPU 平台,因此应针对 MIC 平台的 512bit 向量化宽度进行针对性的优化。虽然在 CPU 版本优化过程中已针对性地做了一定优化,但仍需在真实硬件环境下进行相应的测试和调整。

c)数据传输优化

对 CPU 和 MIC 之间的数据传输进行优化,包含减少不必要的数据传输,计算与传输异步执行等。对一部分应用而言,可能需要对某些非热点函数进行移植,虽然降低了该函数的计算效率,但减少了数据传输时间,进而提高了程序整体的运行效率。

如果热点部分的移植不能满足需求,可以采用 CPU+MIC 协同计算的方式。即采用多线程或多进程的方式,部分进程/线程使用 CPU 进行热点计算,部分进程/线程控制 MIC 协处理器进行热点计算,以充分利用计算资源。如果仍然无法达到要求,可以将程序改为集群版本。即采用 MPI 等通信方式,将程序扩展为多节点应用,每个节点内使用 CPU 与 MIC 协同计算的方式,充分利用多节点计算资源。

结束语 本文在介绍多核处理器和协处理器发展趋势和 MIC 架构的基础上,对 MIC 应用模式进行了介绍和比较,详细描述了 offload 模式,提出了 offload 模式的适用范围,并提出了一种基于 offload 模式的开发策略。依照本文提出的开发策略,可以快速移植开发基于 offload 模式的应用,减少调试、优化的成本,提高开发效率。

能够看出,MIC 作为新推出的并行计算架构,拥有并发度高、学习曲线平缓等特点。可以得出结论,基于 MIC 平台的 offload 编程模式可以与传统 CPU 平台并行方式互为补充,进一步提升硬件并发处理能力,为并行计算应用带来低迁移成本的性能提升。

参 考 文 献

[1] 王恩东,张清,等. MIC 高性能计算编程指南[M]. 北京:中国水利水电出版社,2012

[2] 唐天兵,谢祥宏,等. 多核 CPU 环境下的并行遗传算法的研究[J]. 广西大学学报:自然科学版,2009,8(34):546-550

[3] 陈国良. 并行计算:结构、算法、编程[M]. 北京:高等教育出版社,2003(8)

[4] Akher S,Roberts J. 多核程序设计技术:通过软件多线程提升性能[M]. 李宝峰,富弘毅,李韬,译. 北京:电子工业出版社,2007(3)

[5] Nvidia. NVIDIA CUDA programming guide[OL]. http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html,2013

[6] Sanders J, Kandrot E. CUDA by example: an introduction to general-purpose GPU programming[M]. Addison-Wesley,2011

[7] Jeffers J,Reinders J. Intel Xeon Phi Coprocessor High Performance Programming[M]. morgan kaufmann,2013

[8] Intel. Intel Xeon Phi system software developers guide[OL]. http://software.intel.com/sites/default/files/article/334766/intel-xeon-phi-systemssoftwaredevelopersguide-0.pdf,2013

[9] Intel. Intel C++ Compiler XE 13.1 User and Reference Guide [OL]. http://software.intel.com/sites/products/documentation/doclib/stdxe/2013/composerxe/compiler/cpp-mac/index.htm,2013

(上接第 462 页)

[9] 曾庆成,杨路燕. 基于动态数据驱动的港口水域溢油仿真模型与算法[OL]. http://www.paper.edu.cn/releasepaper/content/201301-475,2013-01-10

[10] Horowitz S R. Highway traffic state estimation using improved mixture kalman filters for effective ramp metering control[C]// Proceedings of 42th IEEE Conference on Decision and Control. 2003:6333-6338

[11] Liu J S, Chen R. Sequential Monte Carlo methods for dynamic systems[J]. Journal of the American statistical association, 1998,93(443):1032-1044

[12] Suh W, Henclewood D, Greenwood A, et al. Modeling pedestrian crossing activities in an urban environment using microscopic traffic simulation[J]. Simulation,2013,89(2):213-224

[13] Godbole V. Intelligent Driver Mobility model and traffic pattern generation based optimization of reactive protocols for vehicular ad-hoc networks[J]. International Journal of Information and Network Security (IJINS),2013,2(3):207-215