中国科学技术大学超级计算中心

Intel MIC高性能计算服务器使用指南

李丰

fli186@mail.ustc.edu.cn

2014年9月23日

目录

1 前言

本用户使用指南主要将对在中国科学技术大学超级计算中心Intel MIC高性能服务器上进行编译以及运行作业做一基本介绍，详细信息请参看相应的指南。

为了便于查看，主要排版约定如下：

- 命令：*command parameters*
- 文件名：*/path/file*
- 环境变量：*MKLROOT*
- 脚本文件内容或长命令：

```
export OPENMPI=/opt/openmpi-1.6.4_intel-13.1.0.146
export PATH=$OPENMPI/bin:$PATH
export MANPATH=$MANPATH:$OPENMPI/share/man
```

- 命令输出：

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
serial	50	Open:Active	–	16	–	–	0	0	0	0
long	40	Open:Active	–	–	–	–	0	0	0	0
normal	30	Open:Active	–	–	–	–	0	0	0	0

由于受水平和时间所限，错误和不妥之处在所难免，欢迎指出错误和改进意见，本人将尽力完善。请从超算中心主页上下载更新后的手册。

2 Intel MIC高性能服务器简介

中国科学技术大学超级计算中心Intel MIC高性能计算服务器，挂载于ChinaGrid高性能计算集群，共计2个计算节点，32个CPU计算核心，244个Intel MIC计算核心，总双精度峰值计算能力为每秒4.96万亿次。具体参数为：

- 管理和用户登录节点：
  - 用户主登录节点，可以进行编译与通过作业调度系统提交作业。禁止直接在此节点上运行作业。
  - 浪潮NF5260M3机架式服务器一台，两颗64位主频2.60GHz的Intel Xeon E5-2670 x86_64 8核CPU，共16核，32GB内存。
  - 节点名为chinagrid。
- MIC计算节点：
  - 节点配置：浪潮NF5588M3机架式服务器，每台含两颗64位主频2.6GHz的Intel Xeon E5-2650 v2 x86_64 8核CPU（共16核）、两块Intel Xeon Phi 7110P MIC加速卡（每块61核，主频1.1GHz，双精度峰值计算能力为1073GFLOPS，8GB内存）、64GB内存及600GB SAS硬盘。
  - 节点名：node45、node46
- 存储节点：浪潮SA5212H2机架式存储服务器，6块2TB SATA硬盘，可用空间9TB。
- 计算网络：56Gbps InfiniBand高速网。
- 管理网络：千兆以太网。
- 操作系统：x86_64架构的64位CentOS 6.5 Linux。
- 编译器：Intel和GNU等C/C++ Fortran编译器。
- 数值函数库：Intel MKL。
- 并行环境：Intel MPI和Open MPI等，支持MPI并行程序；各节点内的CPU共享内存，节点内既支持分布式内存的MPI并行方式，也支持共享内存的OpenMP并行方式；同时支持在节点内部共享内存，节点间分布式内存的混合并行模式。
- 资源管理和作业调度：IBM Platform LSF。
- 常用公用软件安装目录：*/opt*。请自己查看有什么软件，有些软件需要在自己*~/.bashrc*等配置文件中设置后才可以使用。

3 作业调度管理系统

本服务器属于ChinaGrid集群的计算节点,作业调度管理系统的说明亦参见ChinaGrid高性能计算集群使用指南。

node45用于直接编译MIC程序以及直接运行小规模的测试程序。

另外，native模式的MIC程序只允许在node45上运行。

大规模长时间的程序请在专门的mic队列（node46）上提交。

运行作业时，需要利用*bsub␣−q␣mic*选项将任务提交至mic队列即可，如下：

bsub␣−q␣mic␣−o␣%J.log␣−e␣%J.err␣executable

4 Intel MIC介绍

4.1 Intel MIC支持的软件列表

以下是目前Intel官方支持的可以在MIC架构上运行的软件项目，另外也有许多非官方人士在从事MIC程序的开发，请自行搜索。

1. Amber：生物分子动力学模拟
2. BOPM：二叉树定价模型
3. BWA* ALN 0.5.10：生物信息学
4. Embree：Intel开发的高性能光线追踪内核
5. GEMM、STREAM、Linpack：Benchmark
6. GROMACS：利用牛顿运动方程执行分子动力学的多功能包
7. GTC-P：陀螺动力学环形代码-普林斯顿
8. Hogbom Clean基准：天文学
9. LAMMPS：大规模原子/分子大型并行模拟器
10. LBS3D：基于自由能的离散格子玻尔兹曼方法(LBM)的多相流体的模拟工具，可以模拟类不可压缩的二相流体，并使用允许大密度比的多相模型。
11. Mantevo MiniFE：隐式有限元法
12. Mathworks MATLAB
13. 蒙特卡罗欧洲期权定价（带RNG界面）：统计抽样
14. MPI-HMMER：用于蛋白质序列分析的隐马尔可夫模型
15. NAMD：分子动力学，用于大生物分子系统的高性能模拟。
16. Quantum ESPRESSO：用于纳米级别的电子结构计算及材料建模
17. SHOC：Scalable Heterogeneous Computing Benchmark Suite（可扩展异质计算基准指标套件）
18. WRF：Weather Research and Forecasting，气象研究和预报

此处参考下面连接，此链接含有以上软件的下载链接并且会有软件的持续更新：

Code Recipes for Intel Xeon Phi Coprocessor：
https://software.intel.com/en-us/articles/code-recipes-for-intelr-xeon-phitm-coprocessor

4.2 MIC结构简要介绍

虽然作为用户可以不需要了解具体的结构，但是多了解一些还是更有利于对MIC的编程及使用，有利于提升程序应用效率。

Intel MIC(Many Integrated Core)架构，是一种新型的由许多x86核集成在一起的加速设备，这些处理器核心跟历史上的奔腾处理器设计相似，但添加了64位硬件支持，更多的硬件线程（每个核四个硬件线程），512位的SIMD向量指令支持，Intel MIC已经远远超越了20年前的奔腾处理器。

4.2.1 Intel MIC硬件配置

表 1: CPU和MIC的指标对比

Intel Xeon	CPU	MIC
型号	Xeon E5-2650v2	Xeon Phi 7110P
核心数目	8	61
线程数目	16	244
核心频率	2.6GHz	1.238GHz
L1数据cache	8*32KB	61*32KB
L1指令cache	8*32KB	61*32KB
L2 cache	8*256KB	61*512KB
LLC Cache	20MB	/
内存	外置DDR3	8GB GDDR5
设计最大功耗	95W	300W

4.2.2 Intel MIC硬件微结构

如图1所示，所有的核心、内存控制器(GDDR MC)和PCIe接口都由一个高带宽、双向环连接在一起。其中PCIe接口提供了到host系统的内存访问支持，内存控制器提供了到MIC卡自身的内存访问支持。

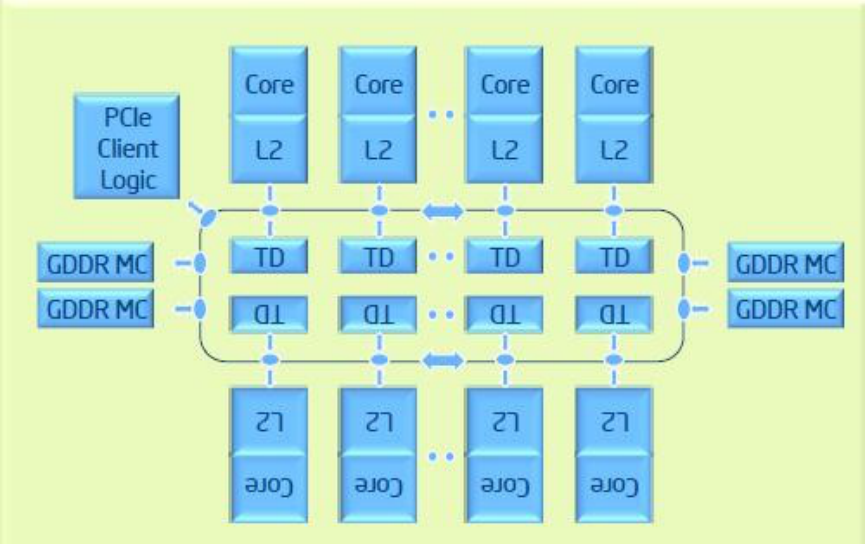

图 1: Intel Xeon Phi

4.3 Intel MIC使用方法

4.3.1 使用MIC OS

MIC可被视为一个独立的系统，其上拥有自己的Linux系统，能够通过ssh协议登录。但是要注意其存储空间依然是易失性存储，与内存是同一个性质，不仅主机重启，即使重启MPSS服务后也会造成文件丢失。所以拷贝到其上的文件一定要自行保存好，做好备份，以免丢失。

中心的服务器上将主机系统的文件系统挂载到了MIC服务器上，用户在登录MIC卡后可以看到用户的自己的主目录，就可以运行程序，省去了拷贝文件到MIC的步骤。

中心服务器上每个节点有两块MIC卡，分别命名为mic0和mic1。

有些时候需要从host系统端登录到MIC卡上，运行native模式的程序，只允许在node45上执行，可以使用如下命令登录mic0（mic1类似）：

ssh mic0

或

ssh node45−mic0

4.3.2 MIC上常用命令

MIC上具有Linux操作系统，大多数Linux基本命令都能使用，如*top*，*ls*等

需要注意其编辑器命令为*vi*，不是*vim*。

4.3.3 Intel MIC开发环境

开发MIC应用时，建议使用Intel的开发环境，如Intel编译器和Intel MPI。

Intel C/C++/Fortran编译器，是一种主要针对Intel平台的高性能编译器，可用于开发复杂且要进行大量计算的C/C++、Fortran程序。

系统已经安装的是Intel Parallel Studio XE 2013，并设置默认使用64位的13.1.0.146版本的Intel编译器和4.1版本的Intel MPI，安装目录为*/opt/intel/composer_xe_2013.2.146*，用户直接使用即可，无需自己设置。

另有最新版的Intel编译器：Intel Parallel Studio XE 2015，安装目录为*/opt/intel/composer_xe_2015*。推荐使用该环境，其设置方法为：

1. 在终端输入以下命令：*mpi−selector−menu*

```
Current system default : intel −mpi−4.1.0.030_intel−compiler−13.1.0.146
Current user default : <none>

    "u" and "s" modifiers can be added to numeric and "U"
    commands to specify "user" or "system−wide".

1. intel −mpi−4.0.3.008_intel−compiler−13.0.1.117
2. intel −mpi−4.1.0.030_intel−compiler−13.0.1.117
3. intel −mpi−4.1.0.030_intel−compiler−13.1.0.146
4. intel-mpi-5.0.1.035_intel-compiler-2015.0.090
5. mvapich2−2.0rc1
6. openmpi−1.6.3_intel−compiler−13.0.1.117
7. openmpi−1.6.3_intel−compiler−13.1.0.146
8. openmpi−1.6.4_gcc−4.4.7
9. openmpi−1.6.4_intel−compiler−13.1.0.146
10. openmpi−1.6.4_pgi−10.6
11. openmpi−1.6.5
12. openmpi−1.8.1_intel−compiler−13.1.0.146
13. openmpi−1.8.2
U. Unset default
Q. Quit

Selection (1−13[us], U[us], Q):
```

2. 在出现的选项中选择，intel-mpi-5.0.1.035_intel-compiler-2015.0.090，序号为4，因此在终端中输入*4u*，回车；
3. 随后输入*Q*，回车；
4. 之后退出当前终端，重新登录用户即可。

Intel MPI库是一种多光纤消息传递接口(MPI)库，当前安装的最新版本为5.0.1（默认使用4.1版本的Intel MPI）。Intel MPI 库可以使开发者采用新技术改变或升级其处理器和互联网络而无需改编软件或操作环境成为可能。主要包含以下内容：

- Intel MPI库运行时环境(RTO)：具有运行程序所需要的工具，包含多功能守护进程(MPD)、Hydra及支持的工具、共享库(.so)和文档。
- Intel MPI库开发套件(SDK)：包含所有运行时环境组件和编译工具，含编译器命令，如*mpiicc*、头文件和模块、静态库(.a)、调试库、追踪库和测试代码。

5 Intel MIC程序设计

5.1 MIC程序的运行模式

MIC程序的运行模式有native模式和offload模式两种。下面一节将举例详细描述MIC应用的基本编程方法。

5.1.1 native模式

整个程序从启动到结束都在MIC上运行，与CPU端无关，故需登录MIC卡使用。

可以将MIC卡视为平时所用的CPU，编程方法也通用，原有CPU代码只需在编译时添加-mmic参数，即可在MIC上运行。

注意MIC卡上不能进行编译操作，故以上编译仍需在host系统上编译，然后登录到MIC卡系统上运行。下一节的代码例子中会体现这一步骤。

5.1.2 offload模式

类似于使用GPU作为计算设备的程序，即程序在CPU端启动并运行，中间遇到需要加速计算的部分则转移到MIC上运行，运行结束后再返回CPU端，不需要登录MIC卡即可使用。

使用的编程方法类似OpenMP，添加一些指定MIC作为加速卡的语句，如以下：

```
#pragma offload target(mic)
```

Offload模式通常可以用于进行更细粒度的并行性化编程，因此更常用，使用更广泛。大型项目基本都是通过此模式构建的。因此我也推荐初学者重点学习此部分。

5.2 Intel MIC编译开源程序

本节参考链接见：https://software.intel.com/en-us/articles/compiling-open-source-for-intel-xeon-phi-coprocessors

MIC协处理器的一个优势是以最小的改变将现有的软件运行起来（在大多数情况下，不需要改变代码）。

当软件只是用make编译（甚至只是调用Intel编译器直接编译）时，只需要在编译和链接选项中加入”-mmic”即可。

当软件是通过GNU Autotools(automake/autoconf)，SCons或CMake编译构建时，此过程就会复杂一些，以下以cmake为例说明。

5.2.1 为Intel MIC编译使用cmake构建的程序

本节参考链接见：Cross-compilation for Intel Xeon Phi Coprocessor with CMake: https://software.intel.com/en-us/articles/cross-compilation-for-intel-xeon-phi-coprocessor-with-cmake

CMake是一个支持跨平台的开源构建程序系统。

需要一个toolchain文件定义所有的编译工具(编译器、链接器、库等)。使用CMake时调用toolchain文件的命令行选项如下：

```
–DCMAKE_TOOLCHAIN_FILE=<path_to_toolchain_file>
```

toolchain文件可以参考下面链接提供的模版：（需根据Intel MPSS的安装路径来修改CMAKE_FIND_ROOT_PATH变量）

cmake-files.tar.gz (769 Bytes)

如果程序需要使用MPI，以上链接中还提供了FindMPI补丁。

以下变量（编译器和编译选项）应在编译构建程序前利用*export*命令定义：

```
export CC=icc
export CXX=icpc
export FC=ifort
export CFLAGS="-mmic"
export CXXFLAGS=$CFLAGS
export FFLAGS=$CFLAGS
export MPI_C=mpiicc
export MPI_CXX=mpiicpc
```

6 Intel MIC代码范例

6.1 π的计算

目前最广泛的并行编程方法为OpenMP、MPI,我们以计算π为例,分别介绍OpenMP、MPI在MIC上使用native模式和offload模式的编程方法。

已知:

$$\int_0^1 \frac{1}{1+x^2}\,dx = \arctan(x)|_0^1 = \arctan(1) - \arctan(0) = \arctan(1) = \frac{\pi}{4}$$

令函数:

$$f(x) = 4/(1+x^2)$$

则有:

$$\int_0^1 f(x)\,dx = \pi$$

其数值化为:

$$\pi \approx \sum_{i=1}^{N} f\left(\frac{2\times i-1}{2\times N}\right) \times \frac{1}{N} = \frac{1}{N} \times \sum_{i=1}^{N} f\left(\frac{i-0.5}{N}\right)$$

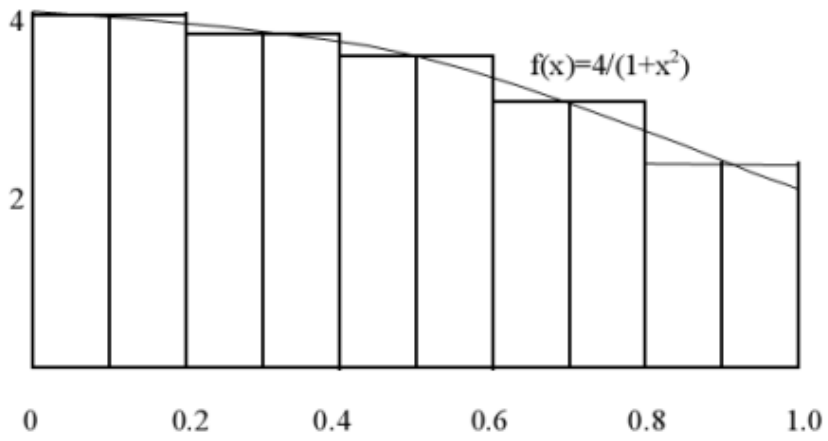

图 2: π的计算

6.2 CPU串行代码

```
#include <stdio.h>
#include <math.h>
```

```
int main(int argc, char *argv[])
{
    int n, i;
    double pi, dx, x;
    n = 10000;
    dx = 1.0 / (double) n;
    pi = 0.0;
    for(i=0; i<n; i++)
    {
        x = dx * ((double)i + 0.5);
        pi += 4.0*dx/(1.0+x*x);
    }
    printf("pi is approximately %.16f\n", pi);
    return 0;
}
```

文件名：*pi-serial.c*

编译：*icc -o pi-seria pi-serial.c*

运行：*./pi-serial*

6.3 并行OpenMP代码

6.3.1 CPU并行OpenMP代码

```
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[])
{
    int n, i;
    double pi, dx, x;
    n = 10000;
    dx = 1.0 / (double) n;
    pi = 0.0;
    #pragma omp parallel for reduction(+:pi)
    for(i=0; i<n; i++)
    {
        x = dx * ((double)i + 0.5);
        pi += 4.0*dx/(1.0+x*x);
    }
    printf("pi is approximately %.16f\n", pi);
```

```
    return 0;
}
```

文件名：*pi-omp.c*

编译：*icc␣-openmp␣-o␣pi-omp␣pi-omp.c*

运行：*./pi-omp*

6.3.2 MIC Native模式OpenMP程序

程序代码与之前完全相同，只需编译时添加"-mmic"参数，即：

icc␣-mmic␣-openmp␣-o␣pi-omp-native␣pi-omp.c

运行方法：

1. 登录到MIC节点上：*ssh␣mic0*
2. 将编译器安装目录下的*libiomp5.so*文件（必需是MIC版本的）拷贝到*$HOME/lib*目录下：*cp␣/opt/intel/composer_xe_2013.2.146/compiler/lib/mic/libiomp5.so␣$HOME/lib*
3. 设置环境变量：*export␣LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/lib*
4. 进入工作目录（可执行文件所在目录）：*cd␣/your/working/directory*
5. 运行：*./pi-omp-native*

6.3.3 MIC offload模式OpenMP程序

```
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[])
{
    int n, i;
    double pi, dx, x;
    n = 10000;
    dx = 1.0 / (double) n;
    pi = 0.0;
    #pragma offload target(mic)
    #pragma omp parallel for reduction(+:pi)
    for(i=0; i<n; i++)
    {
        x = dx * ((double)i + 0.5);
```

```
        pi += 4.0*dx/(1.0+x*x);
    }
    printf("pi is approximately %.16f\n", pi);
    return 0;
}
```

上述程序只是在OpenMP编译指导语句前增加了一行MIC编译指导语句#pragma offload target(mic)。

文件名：*pi-omp-offload.c*

编译：*icc -openmp -o pi-omp-offload pi-omp-offload.c*

运行：*./pi-omp-offload*

6.4 并行MPI代码

6.4.1 CPU并行MPI代码

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
int main(int argc,char *argv[])
{
    int i, n, ntmp, ifr, ito;
    int myid, nprocs;
    double mypi, pi, dx, x;
    double startwtime, endwtime;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&nprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(processor_name,&namelen);
    printf("Process %d of %d is on %s\n", myid, nprocs, processor_name);
    n = 1000;
    if (myid == 0) startwtime = MPI_Wtime();
    n = (int)(n/nprocs) * nprocs;
    dx = 1.0 / (double)n;
    mypi = 0.0;
    ntmp = n / nprocs;
    ifr = myid * ntmp;
```

```
    ito = (myid + 1) * ntmp
    for(i=ifr; i<ito; i++)
    {
        x = dx * ((double)i + 0.5);
        mypi += 4.0*dx/(1.0+x*x);
    }
    MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, PI_COMM_WORLD);
    if (myid == 0)
    {
        endwtime = MPI_Wtime();
        printf("pi is approximately %.16f\n", pi);
        printf("wall clock time = %f seconds\n", endwtime-startwtime);
    }
    MPI_Finalize();
    return 0;
}
```

文件名：*pi-mpi.c*

编译：*mpiicc -o pi-mpi pi-mpi.c*

运行：*mpirun -np 12 ./pi-mpi*

6.4.2 MIC Native模式MPI程序

程序代码与前面的*pi-mpi.c*完全相同，编译时添加-mmic参数，即：

mpiicc -mmic -o pi-mpi-native pi-mpi.c

运行：

- 在MIC节点上的Intel MPI环境已配置好，无需操作。
- 登录到MIC节点上：*ssh mic0*
- 设置环境变量：

```
MPI4MIC=/opt/intel/impi_5.0.1/mic
export PATH=$MPI4MIC/bin:$PATH
export LD_LIBRARY_PATH=$MPI4MIC/lib:$LD_LIBRARY_PATH
```

- 进入工作目录（可执行文件所在目录）：*cd /your/working/directory*
- 运行：*mpirun -np 244 ./pi-mpi-native*

6.4.3 MIC offload模式MPI程序

在CPU版本MPI代码基础上仅添加以下两行指导语句：

```
#pragma offload target(mic:myid)
#pragma omp parallel for reduction(+:mypi)
for(i=ifr; i<ito; i++)
{
  x = dx * ((double)i + 0.5);
  mypi += 4.0*dx/(1.0+x*x);
}
```

文件名：*pi-mpi-offload.c*

编译：*mpiicc -openmp -o pi-mpi-offload pi-mpi-offload.c*

运行：*mpirun -np 2 ./pi-mpi-offload*

每个进程各使用一块MIC卡进行计算

7 参考文献

1. https://software.intel.com/en-us/articles/code-recipes-for-intelr-xeon-phitm-coprocessor
2. http://software.intel.com/mic-developer
3. MPSS Boot Config Guide
4. Intel Xeon Phi Coprocessor Developer’s Quick Start Guide
5. Intel Xeon Phi Coprocessor System Software Developers Guide
6. MIC高性能计算编程指南，王恩东等编著，中国水利水电出版社，2012