# 1、、Windows系统中实施MPI：与UNIX系统下的并行计算做对比

HPC系统常用Unix系统，Windows系统也被用于高端的HPC集群，例如上海超算中心的曙光500A集群，具有30720个核心，使用Windows Server 2008操作系统，LINPACK测试，达到200TF/s，2008年11月Top500排名第10。

本文介绍如何在Windows系统上实施MPI，如何利用Windows系统的特殊功能，并与相同硬件的Unix系统并行计算做对比。

# 2、MPI背景知识

MPICH2是实施MPI并行的一种标准库，还有MS-MPI，Open-MPI。。。



图1 MPICH2架构

MPICH2 is a scalable, multi-network communication subsystem called Nemesis. Nemesis offers very low-latency and high-bandwidth communication by using efficient shared memory operations, lock-free algorithms, band optimized memory-copy routines.

An MPI library typically requires thread services (e.g., thread creation, mutex locks), shared-memory services (for intranode communication), internode communication services (e.g., TCP/IP sockets), and OS process-management services.

MPICH2 also includes a portable library for atomic operations, called OPA (Open Portable Atomics).

# 3、在Windows和Unix系统上实施MPI

Windows和Unix操作系统存在一些差异，介绍如下：

(1) Asynchronous progress

Windows系统支持通信的异步操作，用户启动命令，操作系统确保操作的进度，提醒用户操作完成。

Unix系统上的MPI一般使用nonblocking进度，通过TCP/IP sockets实施节点间通信。

总之，Windows的异步操作效率比Unix系统的高，看不懂，Balabala....

"*Nonblocking progress is generally inefficient compared with asynchronous progress because of deficiencies in poll. It also requires the MPI implementation to do more work than with asynchronous progress where some work is offloaded to the operating system. The poll system call requires the set of socket descriptors to be polled to be contiguous in memory. This restriction increases bookkeeping and reduces scalability of libraries that allow for dynamic connections or that optimize memory allocated for the socket descriptors by dynamically expanding it. When poll returns, indicating the occurrence of an event, the user must search through the entire set of descriptors to find the one with the event.* "

（2）进程管理

启动MPI工作：Unix系统使用fork启动当地MPI进程，使用SSH网络协议远程启动MPI进程。

Windows系统中，进程管理器(process-manager daemon)做这个工作。MPICH2使用SMPD实施进程管理。On Windows, the standalone SMPD daemon impersonates the user launching the job by using the user's credentials. Where available, SMPD can also use technologies such as Active Directory and the job scheduler in Windows HPC Server 2008 to manage user credentials and launch MPI jobs.

管理MPI进程：SMPD通过通信协议提供输入输出、报错等信息，与集群中每个计算节点的无关。因此，这允许用户在安装Unix和Windows系统的计算上运行MPI。

（3）节点间通信

Windows provides an OS service for directly accessing the address space of a specified process, provided the process has appropriate security privileges.

（4）进程

The MPI standard clearly defines the interaction between user threads and MPI in an MPI program. Unix platforms typically use a POSIX threads (Pthreads) library, whereas Windows has its own version of threads.

# 4、试验评估与分析

试验在NCSA的Abe集群上执行，安装Unix和Windows节点。每个节点为2核 Intel 64 2.33GHz处理器，2X4MB L2缓存，4X32 L1缓存，8GB内存。Unix节点安装Linux 2.6.18，使用Intel C/C++ 10.1编译器。Windows节点安装Windows Server 2008 HPC SP2和Visual Studio 2008编译器。网络连接使用千兆以太网。

4.1 Asynchronous Progress

时间循环中执行nonblocking receive, a blocking send and an MPI\_Wait()。

如图2，使用非阻塞进程的效率更好。



图2 Windows执行MPI函数的时间（使用异步进程iocp和非阻塞进程select）

4.2 节点内部（intranode）通信

对比使用lock-free shared-memory queues和direct remote-memory access的大数据（>16KB）通信效率。也对比在Windows和Unix上MPICH2的节点通信效率，测试延迟和带宽。

•延迟：分组从信息源发送到目的地所需的时间。

•带宽：逻辑或物理通信路径最大的吞吐量。

如图3，与L1和L2缓存容量有关。



图3 Windows系统下使用shm和rrvm的节点通信带宽

如图4，小数据通信，2种操作系统的表现都很好，240ns on Unix, 275ns on Windows，仅相差35ns；大数据通信：Unix系统下，进程不共享缓存时，带宽明显下降，而Windows系统使用直接复制方法，2种情况下的性能表现都很好。



图4 Windows和Unix系统下节点内部通信的延迟与带宽

4.3 节点间(internode)通信

对比发现，Unix下MPICH2的通信延迟时间和带宽的表现均较Windows的要好。此测试是点对点的通信的结果，

We are investigating the cause of the difference, but we expect that further tuning and optimization of the Nemesis TCP module for Windows will eliminate the performance gap.

overall application performance depends also on the scalability of the underlying communication subsystem and the ability to overlap communication with computation. We expect the Windows version to have good overall application performance because the asynchronous model with completion ports is scalable and supports overlap.



图5 Windows和Unix系统下节点间通信的延迟与带宽

4.4 支持线程安全的成本

MPI库支持多线程时会发生附加成本(overhead)。

Unix使用Pthread mutex locks确保线程安全，Windows使用intraprocess locks (critical sections)。图6所示，Windows下的附加成本比Unix下的明显要低。



图6 使用多线程后Windows和Unix下节点内部MPI通信延迟的附加成本

# 5、相关工作

微软公司和Intel公司都基于MPICH2发展了自己的MPI库;还有DeinoMPI, OpenMPI, MPI.NET都增加了对Windows系统的支持和优化。

MS-MPI的相关信息见下面链接：

https://docs.microsoft.com/en-us/message-passing-interface/microsoft-mpi

Microsoft MPI (MS-MPI) is a Microsoft implementation of the [Message Passing Interface standard](https://www.mpi-forum.org/) for developing and running parallel applications on the Windows platform.

MS-MPI offers several benefits:

* Ease of porting existing code that uses [MPICH](https://www.mpich.org/).
* Security based on Active Directory Domain Services.
* High performance on the Windows operating system.
* Binary compatibility across different types of interconnectivity options.

# 6、结论

1. Windows和Unix系统下的MPICH2表现相当。Windows下的计算节点间通信表现要差一些。
2. MPICH2 takes advantage of the asynchronous communication features in Windows, which enable applications to overlap communication with computation.
3. Windows HPC Server 2008 introduced a new low-latency RDMA network API, called Network Direct, that enables applications and libraries to use the advanced capabilities of modern high-speed networks, such as InfiniBand. We plan to implement a Nemesis module for Network Direct and study the performance of MPICH2 with high-speed networks on Windows.
4. We also plan to evaluate application-level performance with MPICH2 on Windows, including commercial MPI applications at large scale.