我的Intel MPI 筆記

Table of Contents

起步

使用流程

digraph intel_mpi_flowchart { compile [shape=round, label="(只對程式開發者) 編譯\n並連結應用程式"]; mpd_setup [shape=Mrecord, label="{設定MPD\n守護程式}"]; net_setup [shape=Mrecord, label="{選擇網路\n結構或裝置}"]; run [shape=record, label="運行你的MPI 程式"]; compile -> mpd_setup -> net_setup -> run; }

安裝

依手冊安裝Intel MPI 函式庫，保正庫、腳本和公用程式安裝正常。

起步

1. 載入 `mpivars.[c]sh` 腳本(或以其它類似方式載入Intel MPI 函式庫的環境)。
2. 建立文字文件 `mpd.hosts` ，其中保存有叢集的節點列表，每行一個名字
3. (只針對開發者) 確保環境變數PATH 中包含有對應的編譯器，例如icc。
4. (只針對開發者) 使用適當的編譯驅動編譯測試程序，例如 `mpiicc`

```
mpiicc -o test test.c
```

5. 使用`mpirun`運行測試程序

```
mpirun -r ssh -f mpd.hosts -n <# of processes> ./test
```

編譯和連結

(只針對開發者) 編譯與連結Intel MPI 函式庫：

1. 保證在`PATH`環境變數中編譯器設定正確。使用Intel 編譯器，注意`LD_LIBRARY_PATH`環境變數中含有編譯函式庫的路徑。

2. 透過對應的`mpi`命令編譯MPI 程式。例如呼叫`mpicc`使用GNU C 編譯器：

```
mpicc <path-to-test>/test.c
```

所以支援的編譯器都有對應的以`mpi`開頭的指令，像是Intel Fortran (`ifort`) 對應的為`mpiifort`.

設定MPD 守護程式

Intel MPI 函式庫使用Multi-Purpose Daemon (MPD) 任務排程機制。要執行使用`mpiicc` (或類似) 編譯的程序，首先需要設定好MPD 守護程式。

與系統管理員為系統中所有使用者啟動一次MPD 守護程式不同，使用者需要啟動和維護自己的一組MPD 守護程式。這種設定增強了系統安全性，並為控制可執行程式的環境提供了更強的靈活性。

設定守護進行的步驟如下：

1. 設定對應的環境變數和目錄。例如，在 `.cshrc` 或 `.bashrc` 文件中：
    - 保證 `PATH` 變數中包含有 `<installdir>/bin` 或Intel 64 位元架構對應的 `<installdir>/bin64` 目錄，其中 `<installdir>` 指的是MPI 的安裝路徑。可使用Intel MPI 庫中帶有的 `mpivars.[c]sh` 來設定此變數。
    - 確保 `PATH` 中包含的Python 至少為2.2 或以上版本。
    - (只對開發者) 如果使用Intel 編譯器，確保 `LD_LIBRARY_PATH` 變數包含有編譯器的函式庫目錄。可使用編譯器中帶有的 `{icc,ifort}*vars.[c]sh` 腳本來設定。
    - 設定應用程式所需的其它環境變數。
2. 建立 `$HOME/.mpd.conf` 文件，設定MPD 密碼，需要在文件中寫入一行：

```
secretword=<mpd secret word>
```

不要使用Linux 登陸密碼。 `<mpd secret word>` 可為任意字串，它僅在不同的群集使用者對MPD 守護程序進行控制時有用。

3. 使用 `chmod` 設定 `$HOME/.mpd.conf` 檔案的權限，使得它只能被你自己讀寫：

```
chmod 600 $HOME/.mpd.conf
```

4. 保證你在叢集的所有節點上 `rsh` 指令看到同樣的 `PATH` 和 `.mpd.conf` 內容。例如在叢集的所有節點上執行下面的命令：

```
rsh <node> env
rsh <node> cat $HOME/.mpd.conf
```

保證每個節點都能夠與其它任意節點連接。可使用安裝中提供的`sshconnectivity` 腳本。此腳本使用提供所有節點清單的檔案作為參數，每個節點一行：

```
sshconnectivity.exp machines.LINUX
```

或集群使用的是`ssh`而不是`rsh`, 可參考後面的註釋[1]作相應的命令調整。

5. 建立文字檔案`mpd.hosts`, 其中列出了叢集中所有的節點，每行一個主機名稱。比如：

```
$ cat > mpd.hosts
node1
node2
…
<ctrl>D
```

6. 使用`mpdallexit`指令關閉上一次的MPD 守護程式。

```
mpdallexit
```

7. 使用`mpdboot`[2]指令啟動MPD 守護程式。

```
mpdboot -n <#nodes>
```

如果檔案`$PWD/mpd.hosts`存在，則會被用作預設參數。如果沒有主機名稱文件，啟用`mpdboot`只會在本機上執行MPD 守護程式。

8. 使用`mpdtrace`指令檢查MPD 守護程式的狀態：

```
mpdtrace
```

其輸出結果應該為目前進行MPD 守護程序的節點清單。該列表應該與`mpd.hosts`文件中節點列表符合。

選擇網路結構

Intel MPI 函式庫會動態選擇大部分適用的網路結構以便MPI 程序之間進行通訊。要選擇特定的網路結構，需要設定環境變數`I_MPI_DEVICE`為下表中的某個值：

I_MPI_DEVICE 值	支持的結構
sock	TCP/Ethernet/sockets
shm	Shared memory only (no sockets)
ssm	TCP + shared memory [3]
rdma[:<provider>]	InfiniBand, Myrinet (via specified DAPL provider)
rdssm[:<provider>]	TCP + shared memory + DAPL [4]

運行MPI 程式

運行使用Intel MPI 庫連接的程序，使用`mpiexec`命令：

```
mpiexec -n <# of processes> ./myprog
```

使用`-n`參數設定進程數，這是`mpiexec`唯一需要明顯指定的選項。

如果使用的網路結構與預設的不同，則需要使用`-genv`選項來提供一個可以賦給`I_MPI_DEVICE`變數的值。

例如使用`shm`結構來運行MPI 程序，可執行以下命令：

```
mpiexec -genv I_MPI_DEVICE shm -n <# of processes> ./myprog
```

例如使用`rdma`結構來運行MPI 程序，可執行以下命令：

```
mpiexec -genv I_MPI_DEVICE rdma -n <# of processes> ./myprog
```

可以透過指令選擇任何支援的設備。

如果應用程式運行成功，可將其移至使用不同結構的叢集中，不需要重新連結程式。

除錯

安裝

MPD 設定

編譯運行

參考

- Intel MPI Support Resources

Footnotes:

[1]

注意: 如果叢集中使用`ssh`而不是`rsh`, 需要確保任一節點與其它節點連接時都不需要密碼。這需要參考系統管理手冊。

2

注意: 如果叢集中使用`ssh`而非`rsh`, 在啟動`mpdboot`時需要加上呼叫參數`-r ssh`或`--rsh=ssh`.

3

for SMP clusters connected via Ethernet

4

for SMP clusters connected via RDMA-capable fabrics

要確保所選的網路結構可用。例如，使用`shm`只有當所有進程可以透過共享記憶體進行通訊時才行；使用`rdma`只有當所有進程可以透過單一的DAPL 相互通訊時才行。

版權所有©2012-2018: Vivodo Lio | 日期: 2011-10-14 五00:00

Generated by Emacs 25.3.1 (Org mode 9.1.7), Validate