Пошаговое руководство. Запуск отладчика MPI-кластера в Visual Studio 2010

Visual Studio 2010

В этом пошаговом руководстве описано, как настроить и запустить сеанс отладчика MPI-кластера на локальном компьютере и в кластере Microsoft Windows HPC Server 2008. Руководство также содержит процедуры и пример кода, которые необходимы для создания приложения, использующего интерфейс передачи сообщений (MPI) и API-интерфейс OpenMP (Open Multi-Processing).

Содержание руководства

Требования для использования отладчика MPI-кластера

- На компьютере разработчика должна быть установлена среда Visual Studio 2010 Professional Edition или более высокой версии (с удаленным отладчиком).
- У пользователя должны быть права администратора кластера.
- У Visual Studio должен быть доступ ко всем вычислительным узлам, на которых предполагается запускать сеансы отладки. Необходимый доступ обеспечивают сценарии, указанные ниже.
    - Приложение разрабатывается на головном узле кластера или на выделенном узле входа.
    - Используется кластер, в котором вычислительные узлы подключены к корпоративной сети (топология 2, 4 или 5), а компьютер разработчика присоединен к тому же домену или к домену, для которого настроены отношения доверия с доменом кластера.
- Для отправки приложения в кластер HPC с клиентского компьютера необходимо установить пакет Microsoft HPC Pack 2008.
- Для построения MPI-программ с использованием интерфейса передачи сообщений корпорации Майкрософт необходимо, чтобы на компьютере разработчика был установлен пакет Windows HPC Server 2008 SDK.

Создание образца проекта MPI на C++ в Visual Studio 2010

Пример кода, приведенный в этом разделе, предназначен для создания параллельного приложения, вычисляющего значение числа пи с использованием моделирования методом Монте-Карло.

Этот пример кода выполняет 50 000 000 итераций в каждом процессе MPI. При каждой итерации генерируются случайные числа в интервале [0;1] для определения набора координат *x* и *y*. Набор координат оценивается для определения того, попадает ли точка в область, ограниченную линией с уравнением *x*2 + *y*2 = *1*. Если точка попадает в эту область, значение переменной *count* увеличивается на единицу. Значения *count* из каждого процесса MPI суммируются, и результат присваивается переменной *result*. Общее количество точек, попавших в область (*result*), умножается на четыре и делится на общее количество итераций для получения приблизительного значения числа пи.

Приведенная ниже процедура содержит две реализации метода Монте-Карло.

- В первом примере используются интерфейсы MPI и OpenMP. Дополнительные сведения об интерфейсе OpenMP см. по ссылке OpenMP in Visual C++.

- Во втором примере используются интерфейс MPI и библиотека PPL (Parallel Patterns Library). Дополнительные сведения о библиотеке PPL см. по ссылке Parallel Patterns Library (PLL).

Создание образца проекта

1. Запустите Visual Studio 2010.
2. Создайте консольное приложение Win32 на C++ с именем **ParallelPI**. Используйте проект без предкомпилированных заголовков.
    1. В меню **Файл** выберите пункт **Создать**, а затем — **Проект**.

    2. В диалоговом окне **Новый проект** щелкните элемент **Установленные шаблоны** и выберите пункт **Visual C++**. (В зависимости от настроек Visual Studio пункт **Visual C++** может находиться под узлом **Другие языки**.)

    3. В списке шаблонов щелкните элемент **Консольное приложение Win32**.

    4. В качестве имени проекта введите **ParallelPI**.

    5. Нажмите кнопку **OK**. Откроется мастер создания консольного приложения Win32.

    6. Нажмите кнопку **Далее**.

    7. В разделе **Параметры приложения** в группе **Дополнительные параметры** снимите флажок **Предкомпилированный заголовок**.

    8. Нажмите кнопку **Готово**, чтобы закрыть окно мастера и создать проект.

3. Задайте дополнительные свойства проекта.
    1. В **обозревателе решений** щелкните правой кнопкой мыши элемент **Parallel PI** и выберите пункт **Свойства**. Откроется диалоговое окно **Страницы свойств**.

    2. Разверните элемент **Свойства конфигурации** и выберите пункт **Каталоги VC++**.

В области **Каталоги включения** поместите курсор в начало списка, который отображается в текстовом окне, и укажите местоположение файлов заголовков MS MPI C, после чего введите точку с запятой (;). Пример.

Копировать

```
C:\Program Files\Microsoft HPC Pack 2008 SDK\Include;
```

3. В области **Каталоги библиотек** поместите курсор в начало списка, который отображается в текстовом окне, и укажите местоположение файла библиотеки Microsoft HPC Pack 2008 SDK, после чего введите точку с запятой (;).

   Например, если необходимо построить и отладить 32-разрядное приложение, введите следующее:

   Копировать

   ```
   C:\Program Files\Microsoft HPC Pack 2008 SDK\Lib\i386;
   ```

   Например, если необходимо построить и отладить 64-разрядное приложение, введите следующее:

   Копировать

   ```
   C:\Program Files\Microsoft HPC Pack 2008 SDK\Lib\amd64;
   ```

4. В разделе **Компоновщик** выберите элемент **Ввод.**

   В разделе **Дополнительные зависимости** поместите курсор в начало списка, который отображается в текстовом окне, и введите следующее:

   ```
   msmpi.lib;
   ```

5. При использовании примера кода с интерфейсом OpenMP выполните указанные ниже действия.

   Разверните узлы **Свойства конфигурации** и **C/C++** и выберите элемент **Язык.**

   В качестве значения свойства **Поддержка Open MP** выберите **Да (/openmp)**, чтобы включить поддержку интерфейса Open MP компилятором.

6. Нажмите кнопку **OK**, чтобы закрыть страницы свойств.

4. В главном исходном файле выделите весь код и удалите его.
5. Вставьте в пустой исходный файл один из приведенных ниже примеров кода. В первом примере используются интерфейсы MPI и OpenMP, а во втором — интерфейс MPI и библиотека PPL.

В приведенном ниже примере кода используются интерфейсы MPI и OpenMP. Функция `ThrowDarts` использует параллельный цикл `for` интерфейса OpenMP для задействования процессоров с несколькими ядрами, если они доступны.

C++

Копировать

```
// ParallelPI.cpp : Defines the entry point for the MPI application.
//
#include "mpi.h"
#include "stdio.h"
#include "stdlib.h"
#include "limits.h"
#include "omp.h"
#include <random>

int ThrowDarts(int iterations)
{
std::tr1::uniform_real<double> MyRandom;
std::tr1::minstd_rand0 MyEngine;


double RandMax = MyRandom.max();
int count = 0;
omp_lock_t MyOmpLock;

omp_init_lock(&MyOmpLock);
//Compute approximation of pi on each node
#pragma omp parallel for
for(int i = 0; i < iterations; ++i)
{
double x, y;
x = MyRandom(MyEngine)/RandMax;
y = MyRandom(MyEngine)/RandMax;

if(x*x + y*y < 1.0)
{
omp_set_lock(&MyOmpLock);
count++;
omp_unset_lock(&MyOmpLock);
}
}

omp_destroy_lock(&MyOmpLock);

return count;
}

int main(int argc, char* argv[])
{
int rank;
int size;
int iterations;
int count;
int result;
double time;
MPI_Status s;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&size);
```

```
if(rank == 0)
{
//Rank 0 asks the number of iterations from the user.
iterations = 50000000;
if(argc > 1)
{
iterations = atoi(argv[1]);
}
printf("Executing %d iterations.\n", iterations);
fflush(stdout);
}
//Broadcast the number of iterations to execute.
if(rank == 0)
{
for(int i = 1; i < size; ++i)
{
MPI_Ssend(&iterations, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
}
}
else
{
MPI_Recv(&iterations, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &s);
}

//MPI_Bcast(&iterations, 1, MPI_INT, 0, MPI_COMM_WORLD);

count = ThrowDarts(iterations);

//Gather and sum results
if(rank != 0)
{
MPI_Ssend(&count, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
}
else
{
for(int i = 1; i < size; ++i)
{
int TempCount = 0;
MPI_Recv(&TempCount, 1, MPI_INT, i, 0, MPI_COMM_WORLD, &s);
count += TempCount;
}
}
result = count;

//MPI_Reduce(&count, &result, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

if(rank == 0)
{
printf("The value of PI is approximated to be: %16f",
4*((float)result/(float)(iterations*size)));
}

MPI_Barrier(MPI_COMM_WORLD);

MPI_Finalize();
return 0;
}
```

В приведенном ниже примере кода используются библиотека PPL вместо интерфейса OpenMP и групповые операции MPI вместо операций "точка-точка".

C++

Копировать

```
// ParallelPI.cpp : Defines the entry point for the MPI application.
//
#include "mpi.h"
#include "stdio.h"
#include "stdlib.h"
#include "limits.h"
#include <ppl.h>
#include <random>
#include <time.h>

using namespace Concurrency;

int ThrowDarts(int iterations)
{

combinable<int> count;

int result = 0;


parallel_for(0, iterations, [&](int i){

std::tr1::uniform_real<double> MyRandom;
double RandMax = MyRandom.max();
std::tr1::minstd_rand0 MyEngine;
double x, y;

MyEngine.seed((unsigned int)time(NULL));

x = MyRandom(MyEngine)/RandMax;
y = MyRandom(MyEngine)/RandMax;

if(x*x + y*y < 1.0)
{
count.local() += 1;
}
});

result = count.combine([](int left, int right) { return left + right;
});

return result;
}

void main(int argc, char* argv[])
{
int rank;
int size;
int iterations;
int count;
int result;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&size);

if(rank == 0)
{
//Rank 0 reads the number of iterations from the command line.
```

```
//50M iterations is the default.
iterations = 50000000;
if(argc > 1)
{
iterations = atoi(argv[argc-1]);
}
printf("Executing %d iterations on %d nodes.\n", iterations, size);
fflush(stdout);
}
//Broadcast the number of iterations to execute.
MPI_Bcast(&iterations, 1, MPI_INT, 0, MPI_COMM_WORLD);

count = ThrowDarts(iterations);

//Gather and sum results
MPI_Reduce(&count, &result, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

if(rank == 0)
{
printf("The value of PI is approximated to be: %16f",
4*((double)result/(double)(iterations*size)));
}

MPI_Barrier(MPI_COMM_WORLD);

MPI_Finalize();

}
```

6. В меню **Файл** выберите пункт **Сохранить все**.
7. В меню **Построение** выберите пункт **Перестроить решение**.

Настройка и запуск отладчика MPI-кластера

После построения приложения можно настроить и запустить отладчик. В этом разделе описаны три варианта отладки.

- Отладка одного процесса MPI на локальном компьютере
- Отладка нескольких процессов MPI на локальном компьютере
- Отладка одного или нескольких процессов MPI в кластере

Замечание.

Начать работу с отладчиком MPI-кластера без запуска отладки невозможно. Сочетание клавиш Ctrl+F5 (или выбор пункта **Запуск без отладки** в меню **Отладка**) также запускает отладку.

Отладка одного процесса MPI на локальном компьютере

Для отладки на локальном компьютере с применением только одного процесса MPI используйте тот же процесс, с помощью которого отлаживали бы любое другое приложение. Установите точку останова в необходимом месте программы и нажмите клавишу F5, чтобы запустить отладчик.

Программы MPI взаимодействуют по протоколу IP через порты. При первом запуске программы MPI может быть выведено предупреждение брандмауэра, уведомляющее об открытии порта. Ознакомьтесь с сообщением и разберитесь с тем, какие изменения будут внесены в систему. Для продолжения отладки на локальном компьютере необходимо разблокировать брандмауэр.

Отладка нескольких процессов MPI на локальном компьютере

Ниже описано, как запустить локальный сеанс отладки для приложения **ParallelPI**.

Запуск отладчика MPI-кластера с четырьмя процессами MPI, запущенными на локальном компьютере

1. В **обозревателе решений** щелкните правой кнопкой мыши элемент **Parallel PI** и выберите пункт **Свойства**. Откроется диалоговое окно **Страницы свойств**.
2. Разверните узел **Свойства конфигурации** и выберите элемент **Отладка**.
3. В группе **Загружаемый отладчик** выберите элемент **Отладчик MPI-кластера**.
4. В раскрывающемся списке **Среда выполнения** выберите элемент **Правка узла Hpc**. Откроется диалоговое окно **Выбор узла**.
5. В раскрывающемся списке **Головной узел** выберите элемент **localhost**.
6. В поле **Количество процессов** выберите значение **4**.
7. Нажмите кнопку **ОК**, чтобы сохранить изменения и закрыть диалоговое окно **Выбор узла**.
8. Приложение **ParallelPI** принимает один аргумент, который определяет количество выполняемых итераций. По умолчанию выполняется 50 000 000 итераций. Для локального сеанса отладки уменьшите количество итераций до 5 000, выполнив указанное ниже действие.

   В поле **Аргументы приложения** введите значение **5000**.

9. Нажмите кнопку **ОК**, чтобы сохранить изменения и закрыть окно **Страницы свойств**.
10. Установите точку останова в теле параллельного цикла `for`.
11. Нажмите клавишу F5, чтобы запустить отладчик.
12. Появятся пять окон консоли: одно окно **cmd.exe** и четыре окна **ParallelPI.exe** (по одному для каждого запущенного процесса). В окне консоли, соответствующем процессу с рангом 0, будут выведены количество итераций и вычисленное приблизительное значение числа пи.
13. В меню **Отладка** выберите пункт **Окна**, а затем — **Процессы**.
14. Выберите активный процесс для отладки, дважды щелкнув его в окне **Процессы**.

Замечание.

По умолчанию при отладке нескольких процессов точка останова влияет на все отлаживаемые процессы. Чтобы избежать прерывания процессов в непредвиденном месте, снимите флажок **Прерывать все процессы при прерывании одного**. (В меню **Сервис** выберите пункт **Настройки**, а затем — **Отладка**). Дополнительные сведения об изменении поведения точки останова см. в разделе Инструкции. Прерывание выполнения.

Отладка одного или нескольких процессов MPI в кластере

При запуске отладчика MPI в кластере отладчик отправляет приложение в кластер как задание. Среды выполнения Visual C, соответствующие проекту (x86 или x64, отладка или выпуск), должны находиться в рабочем каталоге на вычислительных узлах. Если необходимые среды выполнения отсутствуют на вычислительных узлах, необходимо включить их в развертывание отладчика, указав их с помощью свойства **Дополнительные файлы для развертывания**. Приведенная ниже процедура включает действие по развертыванию отладочной библиотеки DLL OpenMP времени выполнения. По умолчанию библиотека времени выполнения C (CRT) развертывается при запуске отладчика MPI-кластера. Если необходимые библиотеки времени выполнения недоступны, при попытке запустить приложение возникнут ошибки. Если библиотека времени выполнения OpenMP не подключена, точки останова не сработают.

Запуск отладчика MPI в кластере

1. В **обозревателе решений** щелкните правой кнопкой мыши элемент **Parallel PI** и выберите пункт **Свойства**. Откроется диалоговое окно **Страницы свойств.**
2. Разверните узел **Свойства конфигурации** и выберите элемент **Отладка.**
3. В группе **Загружаемый отладчик** выберите элемент **Отладчик MPI-кластера.**
4. В раскрывающемся списке **Среда выполнения** выберите элемент **Правка узла Hpc**. Откроется диалоговое окно **Выбор узла.**
5. В раскрывающемся списке **Головной узел** выберите имя головного узла кластера, который необходимо использовать.

   Список головных узлов заполняется с контроллера домена Active Directory. Список содержит только кластеры, входящие в домен пользователя. Если нужный головной узел не отображается, введите имя или адрес IPv4 головного узла в поле свойства.

6. В поле **Количество процессов** выберите значение **4.**
7. В поле **Назначить процесс для** выберите способ выделения процессов. Можно выделить один процесс на **ядро**, **сокет** или **узел.**
8. Нажмите кнопку **ОК**, чтобы сохранить изменения и закрыть диалоговое окно **Выбор узла.**
9. В поле **Каталог развертывания** укажите общий каталог на головном узле. Если каталог развертывания не существует и у пользователя имеются права на запись в указанный корневой каталог, то каталог развертывания будет создан автоматически.

   Общий каталог **CcpSpoolDir** создается при установке пакета HPC Pack 2008 на головном узле. Например, введите следующую строку, где <*myHeadNode*> — имя используемого кластера:

   \\<*myHeadNode*>*CcpSpoolDir*\

10. В поле **Рабочий каталог** укажите локальный рабочий каталог на каждом вычислительном узле. Например, введите следующую строку, где <*myUserName*> — ваше имя пользователя:

    C:\Users\<*myUserName*>*ParallelPI*

11. Если используется пример кода с интерфейсом OpenMP, добавьте файл отладочной DLL-библиотеки OpenMP (**Microsoft.VC100.DebugOpenMP\vcomp100d.dll**).

1. В группе **Дополнительные файлы для развертывания** выберите пункт **<Изменить файл…>**. Откроется диалоговое окно **Выбор файла и папки**.

2. Нажмите кнопку **Добавить файл** , перейдите к файлу **Microsoft.VC100.DebugOpenMP\vcomp100d.dll**, выберите его и нажмите кнопку **Открыть**.

   Например, на компьютерах с архитектурой x86 и 64-разрядной версией системы Windows Server 2008 этот файл по умолчанию расположен по следующему пути:

   C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\redist\Debug_NonRedist\x86\Microsoft.VC100.DebugOpenMP\vcomp100d.dll

3. Нажмите кнопку **ОК**, чтобы добавить файл и закрыть диалоговое окно **Выбор файла и папки**.

12. Нажмите кнопку **ОК**, чтобы сохранить изменения и закрыть окно **Страницы свойств**.
13. Установите точку останова в теле параллельного цикла `for`.
14. Нажмите клавишу F5, чтобы запустить отладчик.
15. Поскольку задание отправляется в кластер, отобразится окно ввода пароля для подключения к кластеру. Введите пароль и нажмите клавишу ВВОД.
16. После запуска отладчика проверьте в окне процессов расположение процессов. Узнайте для каждого процесса вычислительный узел, на котором запущен процесс. Эти сведения содержатся в столбце **Квалификатор транспорта**.

Приложение. Файлы, развертываемые Visual Studio в дополнение к двоичным файлам приложения (и CRT по запросу)

- DebuggerProxy.dll
- DebuggerProxy.dll.manifest
- Delete_from_workdir.bat: скрипт для удаления развернутых файлов
- Deploy_to_workdir.bat: скрипт для копирования файлов из каталога развертывания в рабочий каталог
- dbghelp.dll
- mcee.dll
- Mpishim.bat: скрипт для запуска удаленного отладчика
- Mpishim.exe: программа, организующая взаимодействие между интегрированной средой разработки и файлом Msvsmon.exe
- Msvsmon.exe: удаленный отладчик

- Msvsmon.exe.config
- PfxTaskProvider.dll
- symsrv.dll
- symsrv.yes
- vbdebug.dll
- 1049\msdbgui.dll
- 1049\vbdebugui.dll

См. также

Понятия

Свойства конфигурации отладчика MPI-кластера
Отладка приложений MPI в кластере HPC

Другие ресурсы

Основы отладки
mpiexec Command Reference