Windows HPC 講習会

- MS-MPIプログラミング演習 -

同志社大学生命医科学部　廣安 知之

同志社大学工学研究科　中尾 昌広

2009/9/25

目的とスライドの流れ

- 目的
  - MS-MPIの利用方法を習得して頂くこと
- スライドの流れ
  - MPI（Message Passing Interface）とは
  - MS-MPI（Microsoft-MPI）とMPI.NET
  - MS-MPIを利用した並列計算の概要
  - MS-MPIを用いた並列アプリケーションの作り方
  - MS-MPIのプログラミング演習

MPI（Message Passing Interface）とは

- 分散メモリプログラミングに必要なデータ通信のための標準仕様
- MPIの仕様に準じた実装が数多く存在する
- MPI実装を用いることで複数の計算機の協調動作が可能
- MPI-1とMPI-2があり、MPI-2の方が新しく機能も豊富

実装名	MS-MPI	MPI.NET	MPICH	MPICH2	LAM/MPI
対応Ver.	2	2	1	2	1
対応言語	C、C++、Fortan	.NET言語（C#など）	C、C++、Fortan		

MS-MPI（Microsoft-MPI）とは

- Microsoft社が提供するMPI-2の実装
- C，C++，Fortranなどで利用可能
- Windows HPC Server 2008で並列計算を行うための通信ライブラリ集
- MS-MPIはMPICH2と互換性を出来る限り保持した設計
- MPICH2のプログラムソースがあると、少しの変更で
  - Windowsクラスタのジョブスケジューラの機能
  - Microsoft社のInfinibandドライバを通した並列計算
  - NetworkDirectを用いた通信

  などを利用する事ができる。

MPI.NETとは

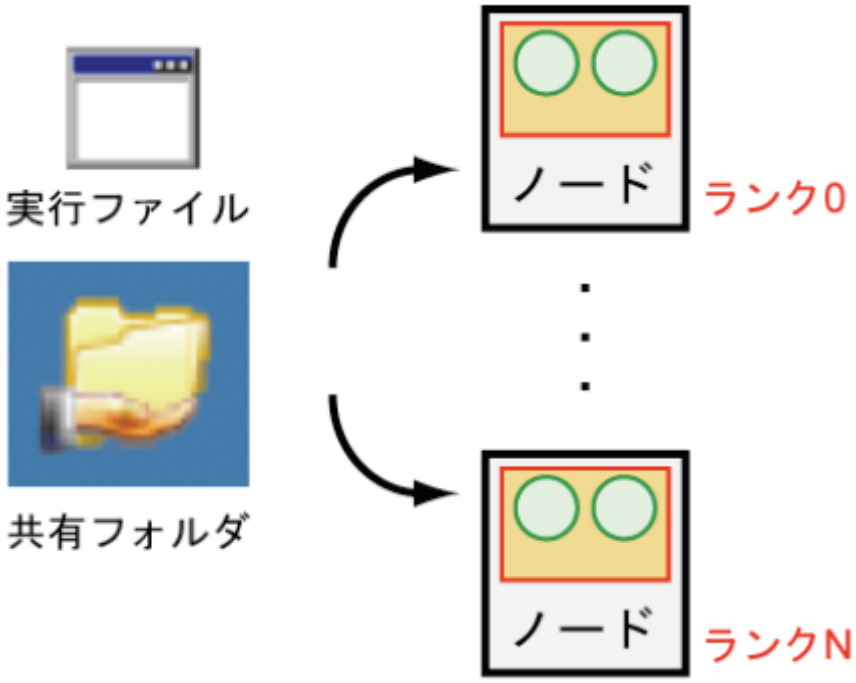
- .NET Framework上で動作する並列計算用ライブラリ
- ノード間の通信を簡易に行えるAPIを提供
- .NETで用いられている言語の全て（特にC#）をサポート
- C#とは，Microsoft社が開発したプログラミング言語
- javaに似たオブジェクト指向型言語であり、プロセッサに依存しない実行ファイルを生成可能

MS-MPIとMPI.NET

- MPI.NETの通信関数の方が，簡易に記述できる（後述）
- MS-MPIの方が，一般に実行は高速

MPIプログラムの概要

- 複数の計算機で動作させたい実行ファイルは1つのみ
- 実行ファイルはすべての計算機が参照できる場所に保存する
- if else文などを用いて、各計算機の処理を実行する

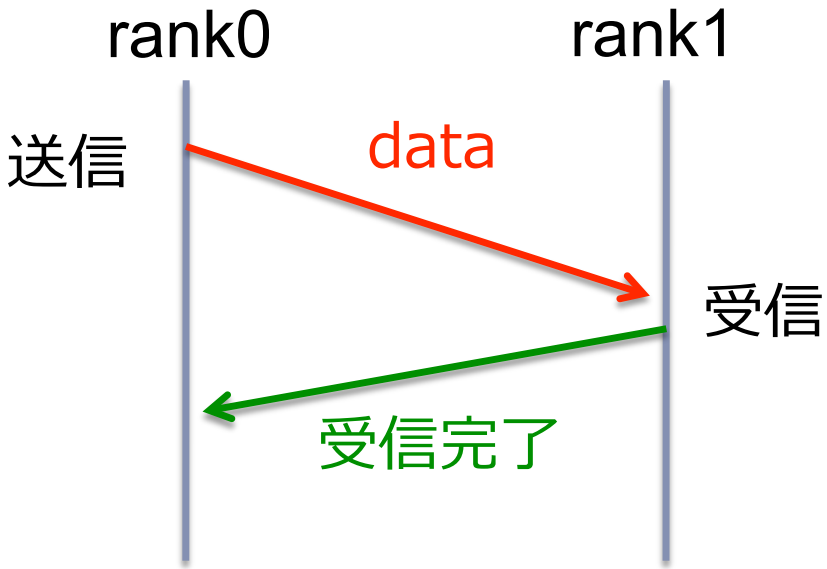



```
If ( rank == 0 ) {
 // ランク0にさせたい内容
} else if (rank == 1) {
 // ランク1にさせたい内容
}
```

通信関数（1対1通信）

	送信関数	受信関数
同期通信	MPI_Send()	MPI_Recv()
非同期通信	MPI_Isend()	MPI_Irecv()

送信関数と受信関数は組で用いる



MPI_Send関数（MS-MPI）

```
int MPI_Send(void *buf, int count, MPI Datatype datatype,
             int dest, int tag, MPI Comm comm)
```

送信バッファのデータを特定の受信先に送信する。

void *buf：送信バッファの開始アドレス

int count：データの要素数

MPI Datatype datatype：データタイプ

int dest：送信先（ランクを指定）

int tag：データ識別用のタグ

MPI Comm comm：コミュニケータ

Send関数（MPI.NET）

```
public void Send<T>(value, dest, tag)
```

送信バッファのデータを特定の受信先に送信する。

value ：送信したい値

int dest：送信先（ランクを指定）

int tag：データ識別用のタグ

MPI_Recv関数（MS-MPI）

```
int MPI_Recv(void *buf, int count, MPI Datatype datatype,
             int source, int tag, MPI Comm comm,
             MPI Status *status)
```

要求されたデータを受信バッファから取り出す。

void *buf：受信バッファの開始アドレス
（受け取ったデータの格納場所）

int source：送信元
（MPI ANY SOURCE で送信元を特定しない）

int tag：データ識別用のタグ
（MPI ANY TAG でメッセージ・タグを特定しない）

MPI Status *status：ステータス

Recv関数（MPI.NET）

```
public T Receive<T>(source, tag)
```

要求されたデータを受信バッファから取り出す。

int source：送信元（ランクを指定）

int tag：データ識別用のタグ

返り値が受信したデータになる

MPI_Isend関数とMPI_Irecv関数

（以下，すべてMS-MPI）

```
int MPI_Isend(void* buf, int count, MPI Datatype datatype,
        int dest, MPI Comm comm, MPI Request *request)
```

MPI Request : 非同期通信における送信，もしくは受信を
要求したメッセージに付けられる識別子.

通信の状況を調べるために用いる

```
int MPI_Irecv(void *buf, int count, MPI Datatype type,
        int source, int tag, MPI Comm comm,
        MPI request *request)
```

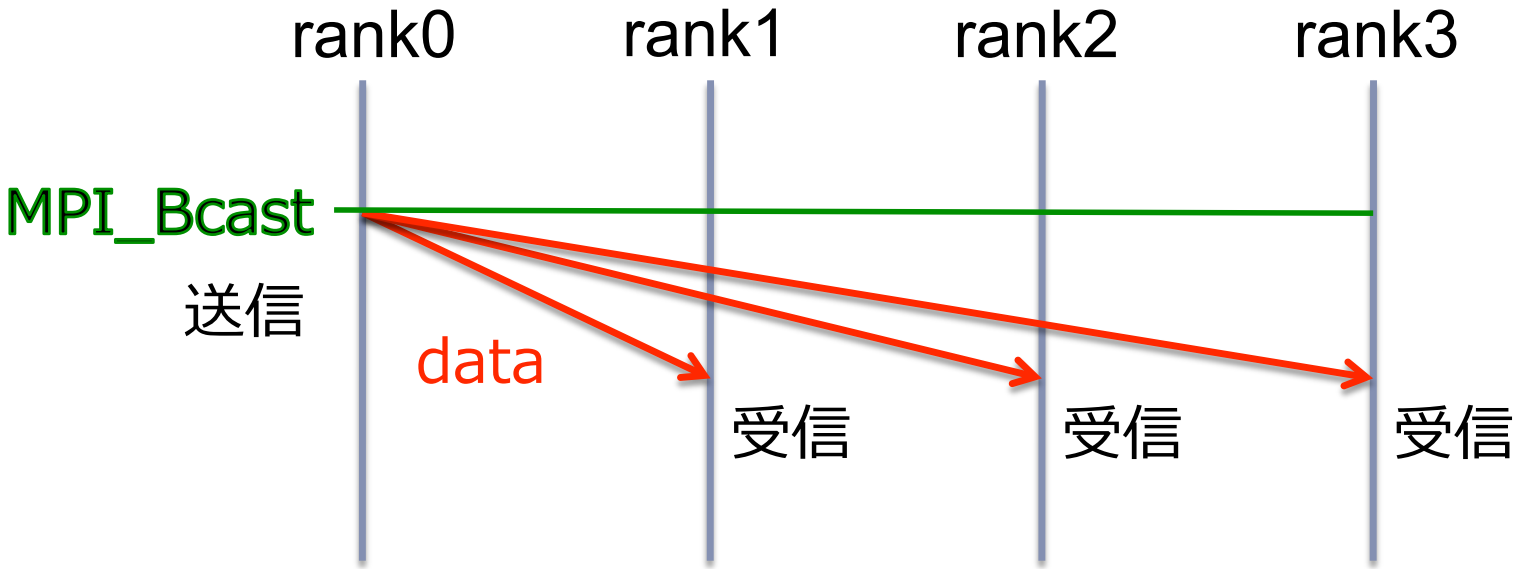
集合通信

- 集合通信は2台以上のプロセス間のデータの通信を行う
- すべてのプロセスが同じ関数を呼び出すことで、
  協調動作を行える

集合通信の例：one to all

```
int MPI_Bcast(void *buf, int count, MPI Datatype datatype,
              int root, MPI Comm comm)
```

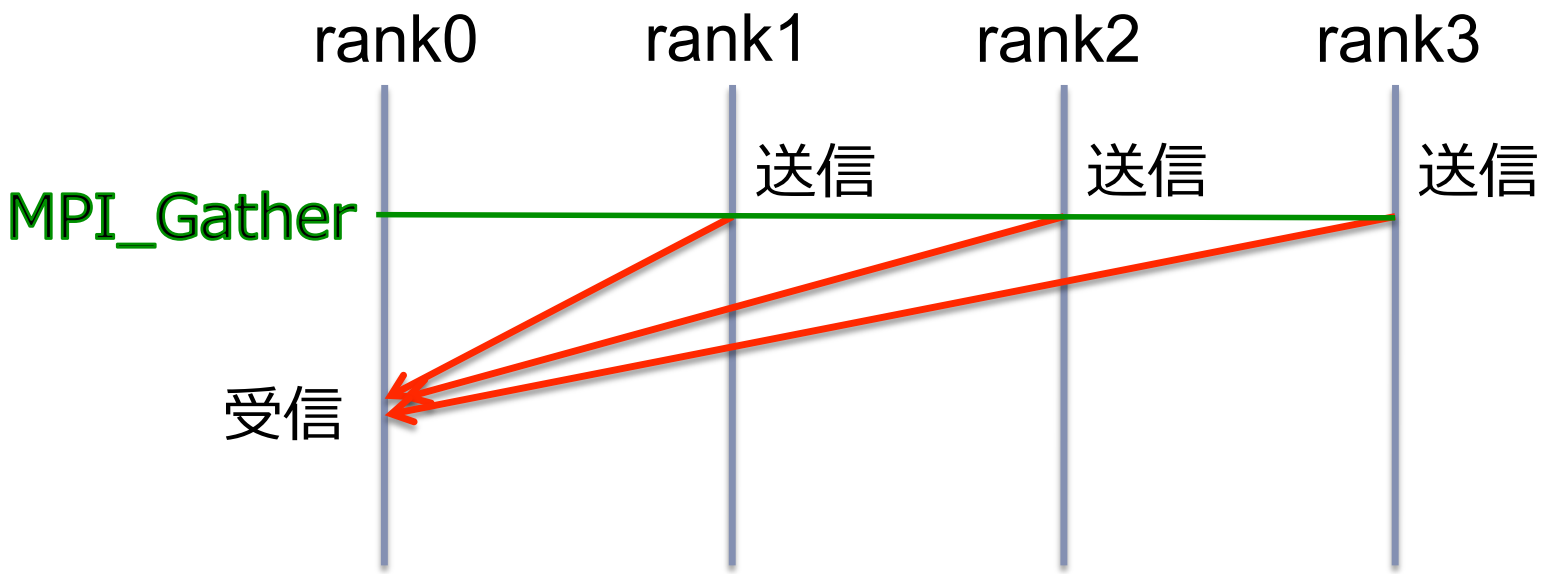
1 つのランク（引数のrootで指定）から全てのランクに
メッセージを一斉に送信する。



集合通信の例：all to one

```
Int MPI_Gather(void *sendbuf, int count,
                    MPI Datatype datatype, void *recvbuf,
                    MPI Datatype datatype, MPI Comm comm)
```

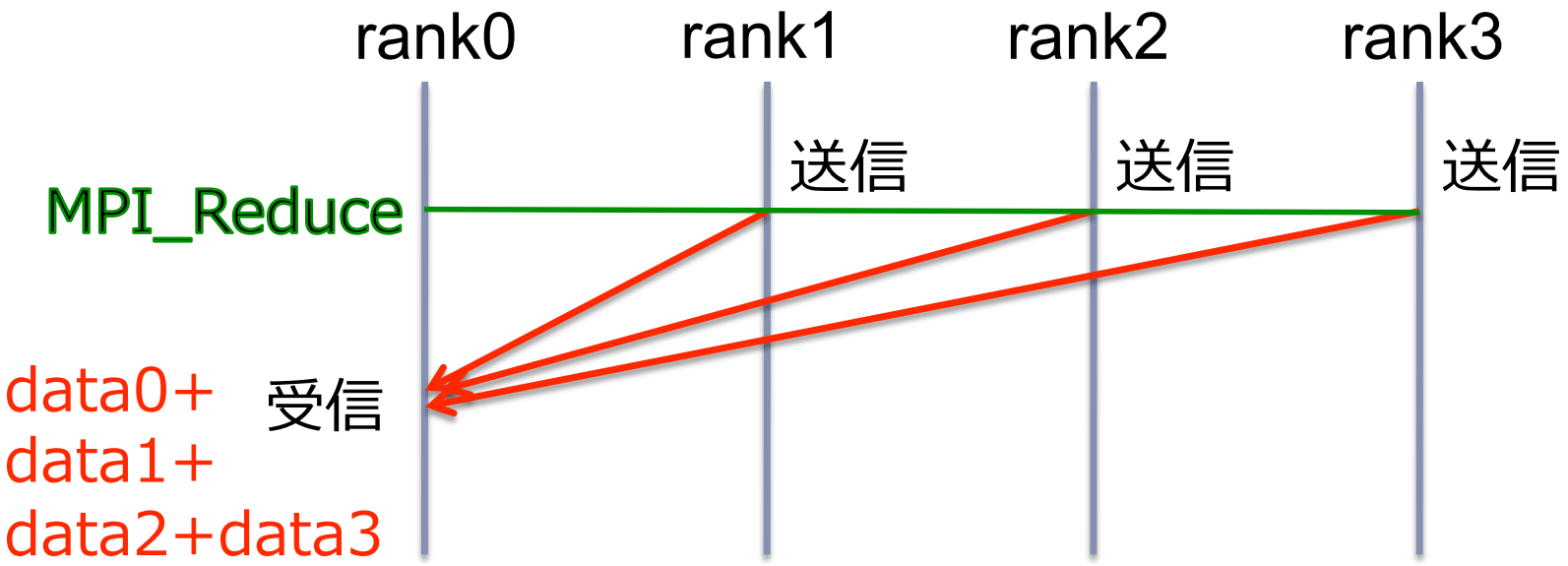
各ランクが持っている値を1つのランクに集める。



集合通信の例：all to one

```
int MPI_Reduce ( void *sendbuf, void *recvbuf, int count,
                 MPI Datatype datatype, MPI op op,
                 int dest,  MPI Comm comm )
```

全てのランクのデータをある１つのランクに集める。
同時に各データを足し合わせるなどの演算を行う。

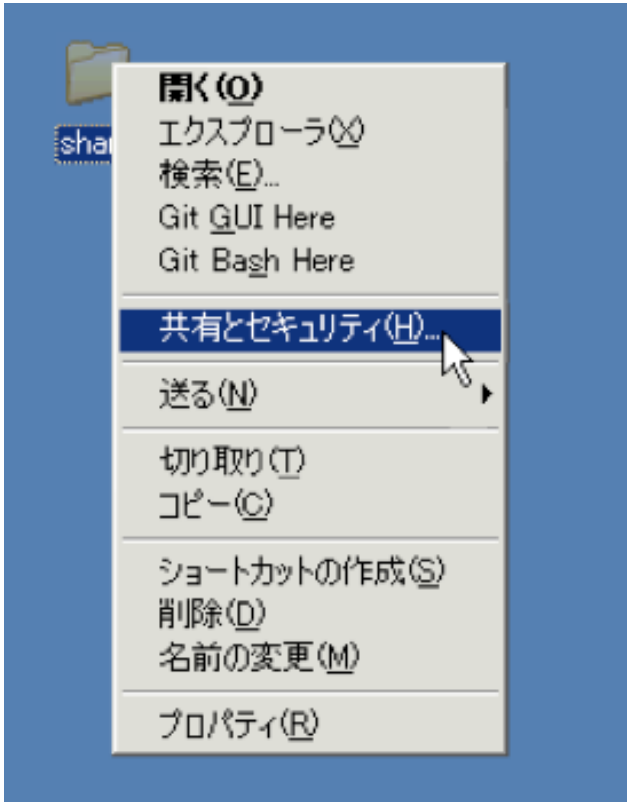


MS-MPIプログラミングの準備

- MS-MPIはHPC Pack 2008に同梱されている。
  そのため、HPC Pack 2008をインストールするのみで
  MS-MPIを利用できる
- 今回は開発環境として、Visual C++ 2008 Express Edition（無償）を用いる
- MS-MPIのアプリケーション開発はWindows Vista、XPでも可能である
- ただし、ノードを跨いだMS-MPIのアプリケーション実行には、Windows Serverが必要である
- すべてのマシンから見ることができる共有フォルダを準備する

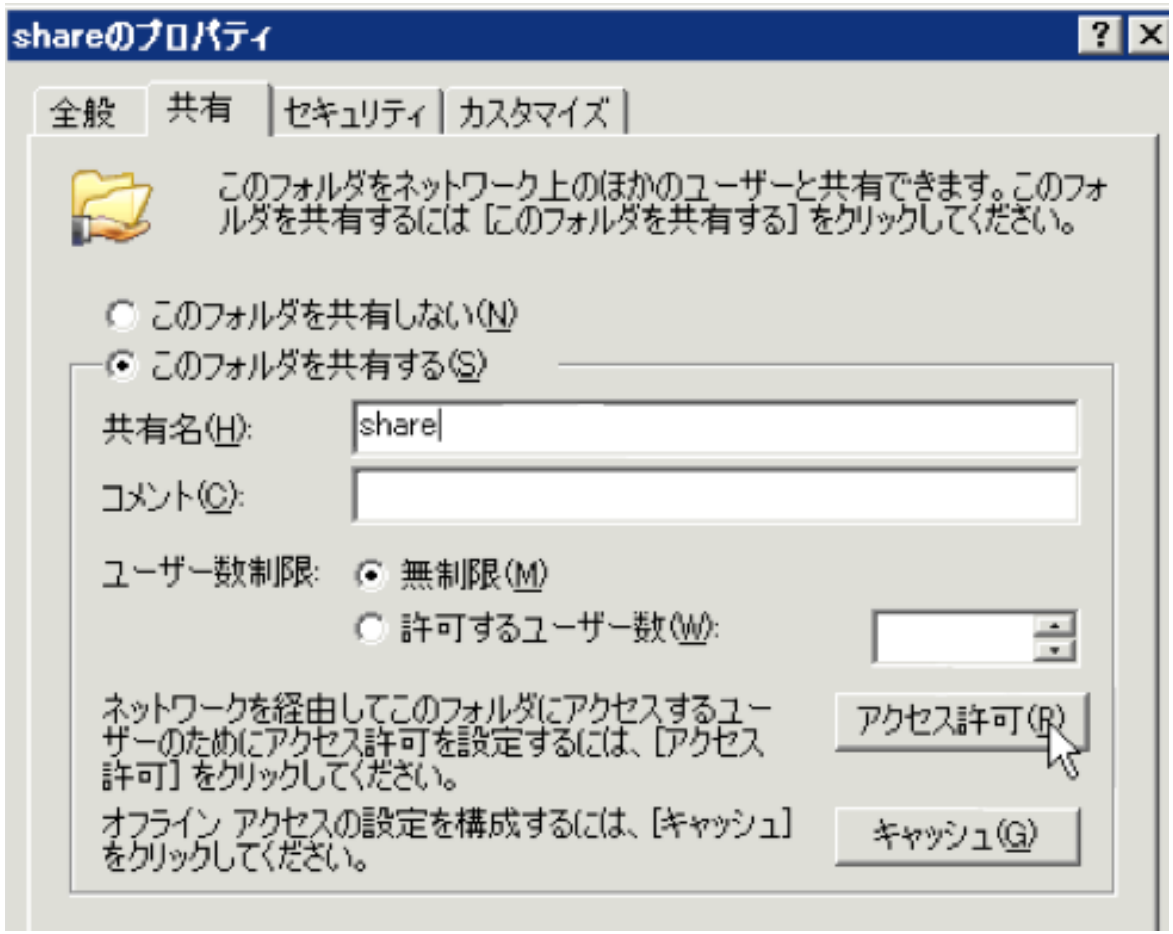
共有フォルダの作成

- デスクトップに適当な名前のフォルダを作成する
- 右クリックし，
  「共有とセキュリティ」を選択



共有フォルダの作成

- 適用な共有名を作成する（shareとして下さい）

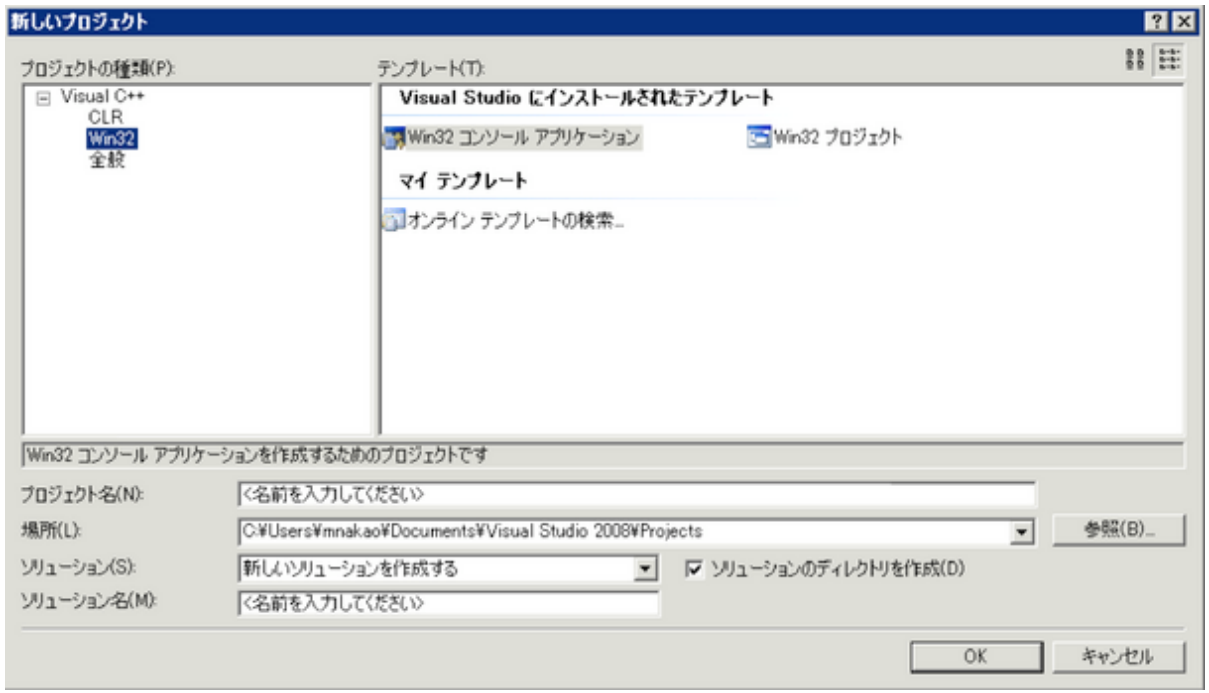


共有フォルダの作成

- アクセス許可で「フルコントロール」にチェック

MS-MPIプログラムの作成方法

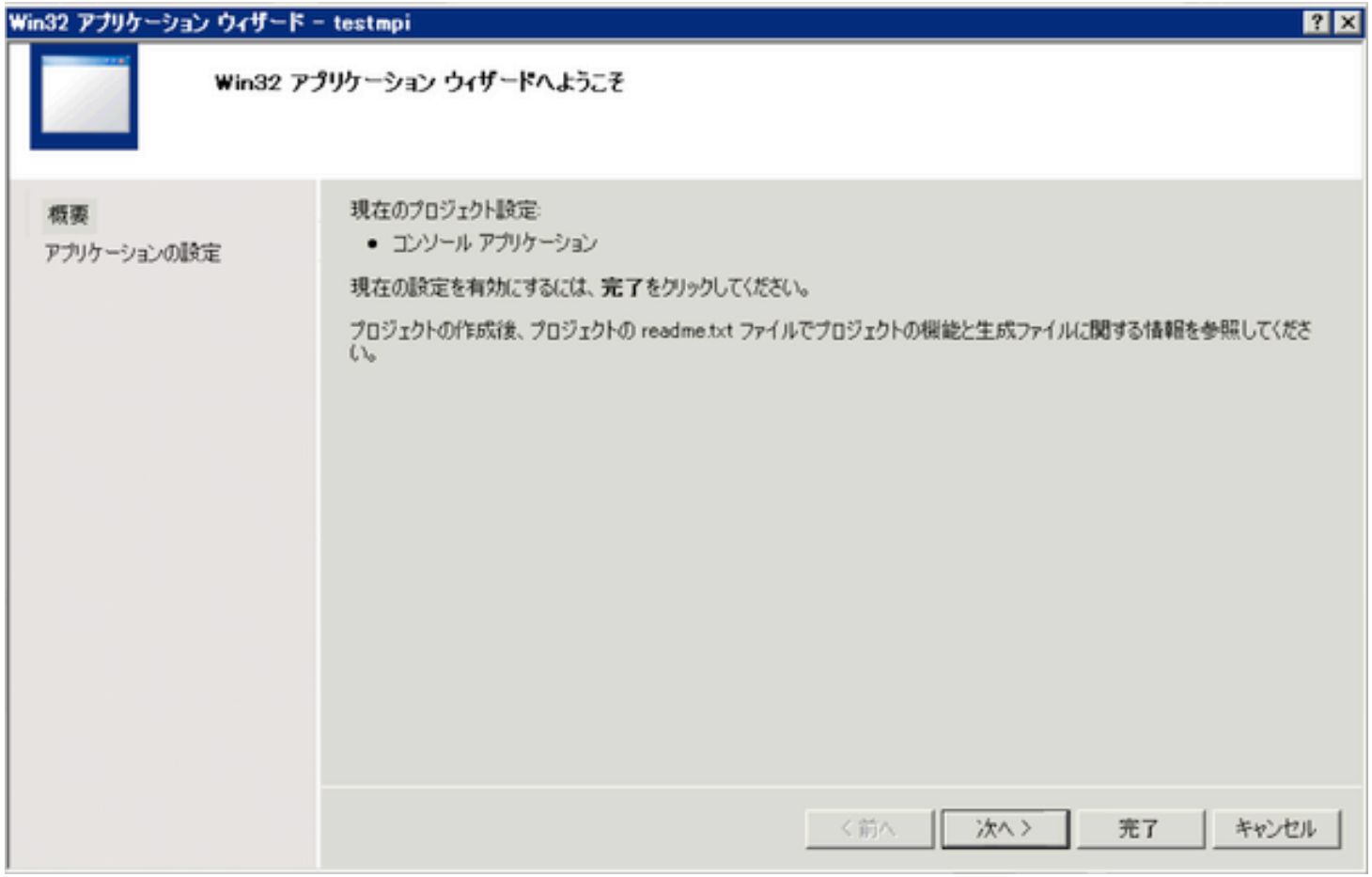
- Visual C++を起動し、「ファイル」→「新規作成」→「プロジェクト」を選択。



「Win32 コンソールアプリケーション」を選択し、プロジェクト名に適当な名前を入力する。

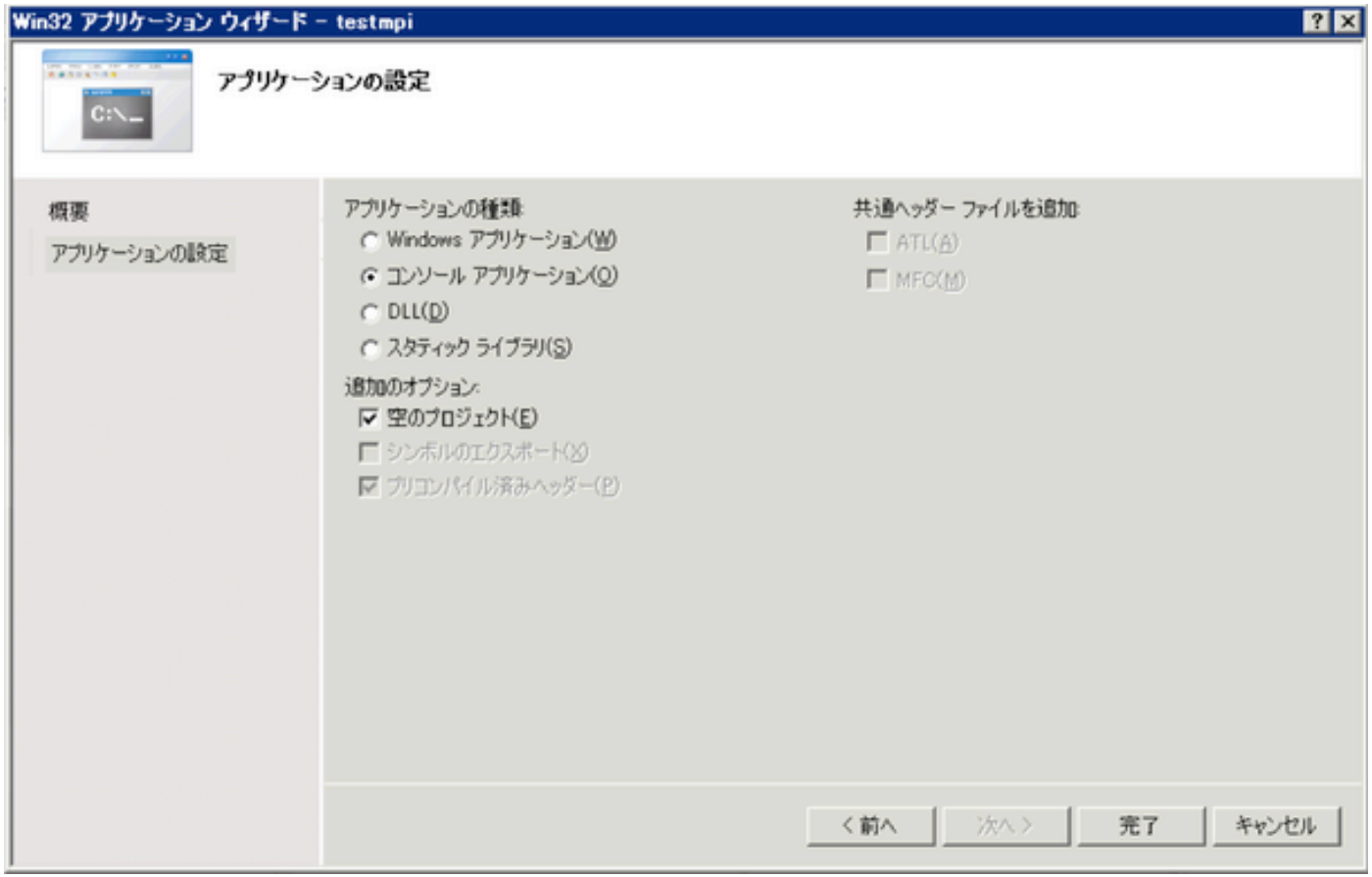
MS-MPIプログラムの作成方法

- 次へを選択



MS-MPIプログラムの作成方法

- 追加のオプションで「空のオブジェクト」を選択。
  「完了」をクリックする。

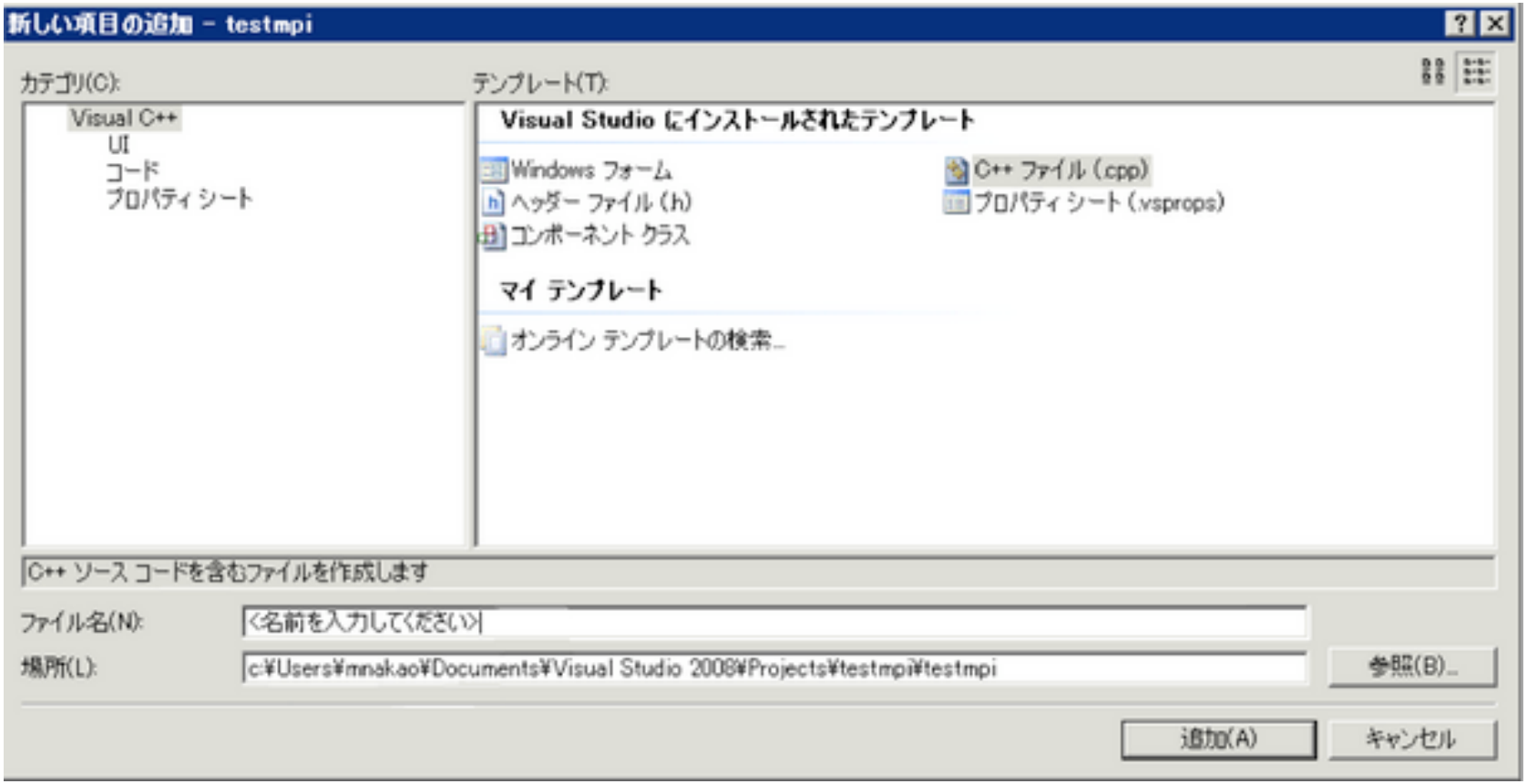


MS-MPIプログラムの作成方法

- C++ファイルをプロジェクトに追加。
  「ソリューションエクスプローラー」の作成した
  プロジェクトの「ソースファイル」を右クイックし、
  「追加」→「新しい項目」を選択。

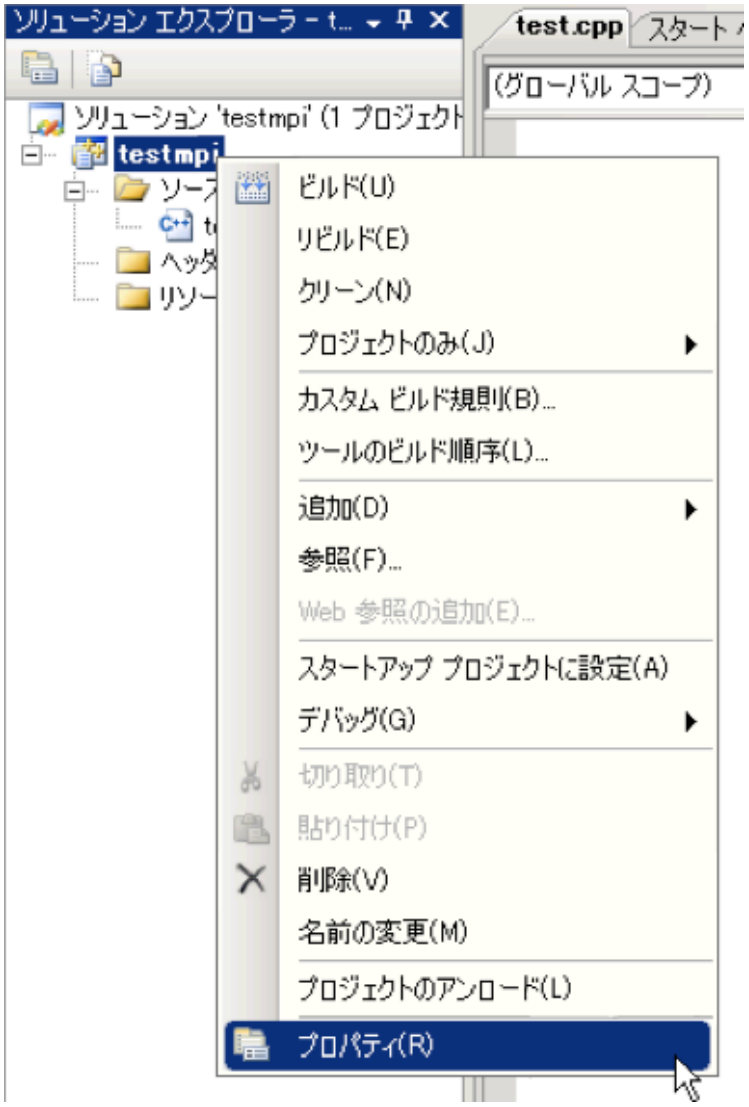
MS-MPIプログラムの作成方法

- テンプレートの「C++ファイル（.cpp）」を選択し、適当なファイル名をつけて作成する。



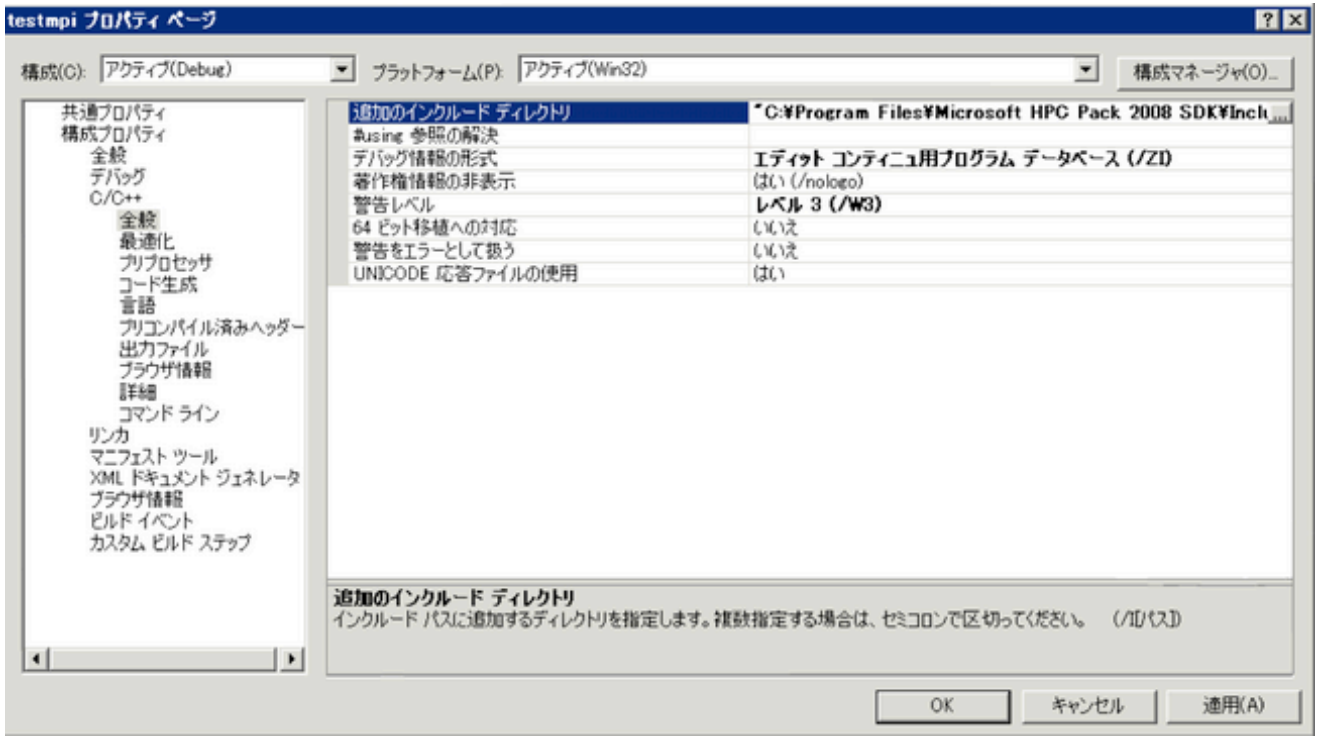
MS-MPIプログラムの作成方法

- インクルードパスの設定
  「ソリューション
  エクスプローラー」の作成した
  プロジェクトの
  「プロパティ」を選択。



MS-MPIプログラムの作成方法

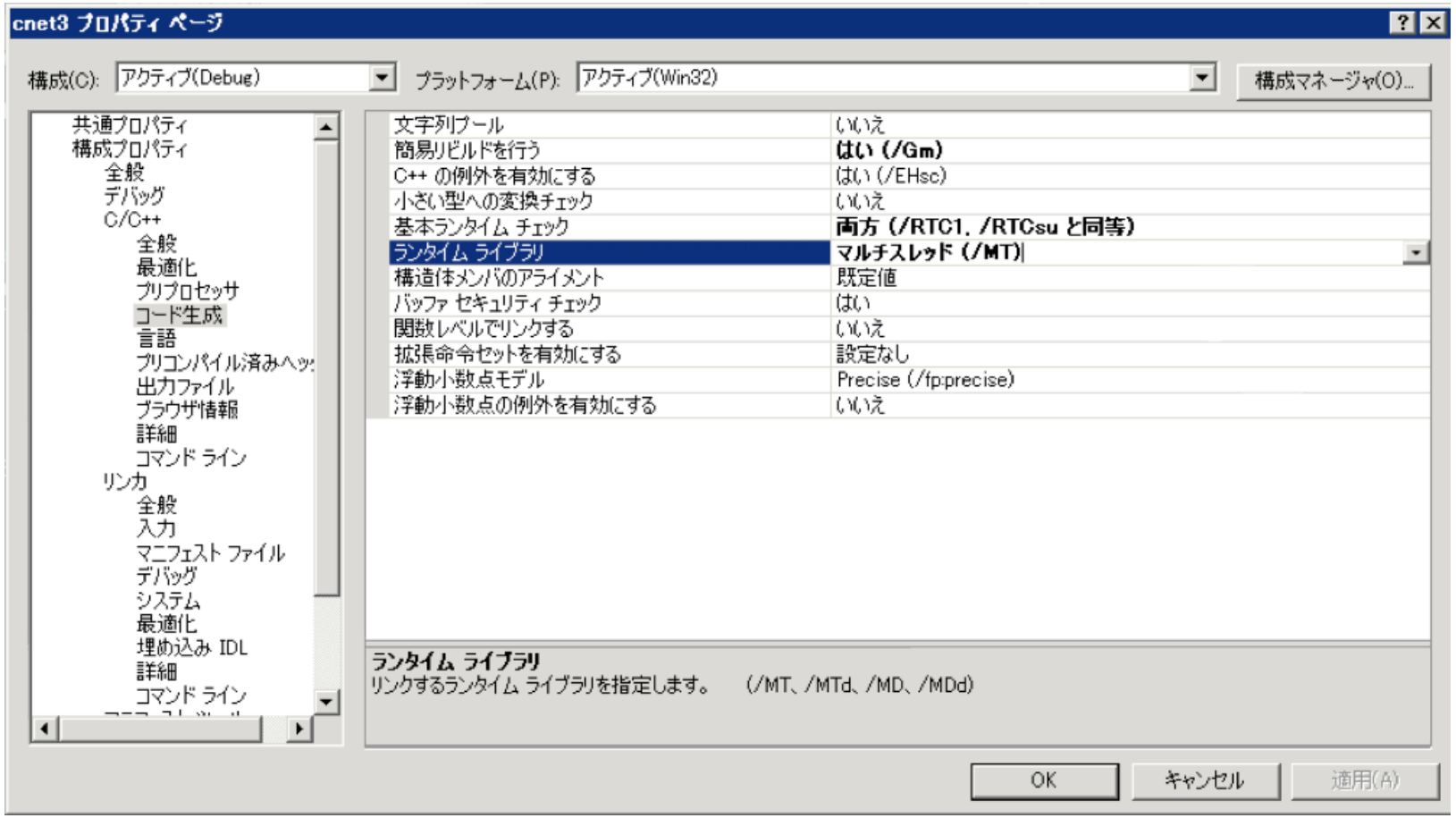
- 「構成プロパティ」→「C/C++」→「全般」→「追加のインクルードディレクトリ」にMS-MPIのインクルードパスを入力する。



今回は「C:\Program Files\Microsoft HPC Pack 2008 SDK \Include」と入力した。

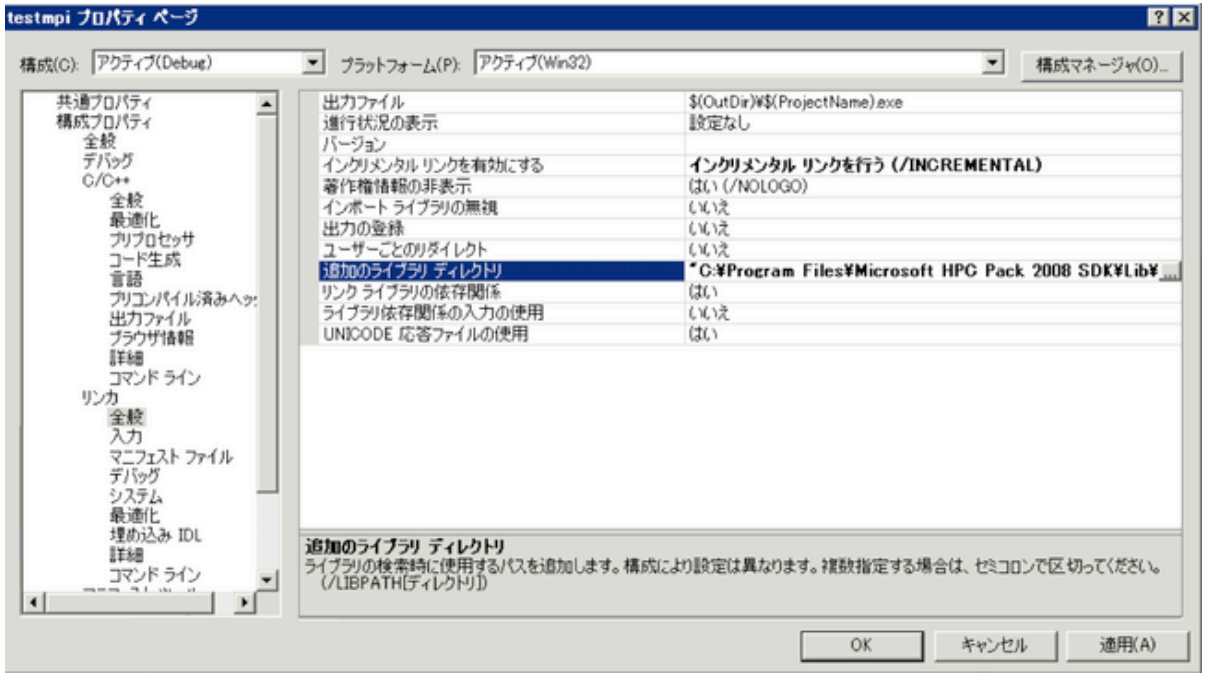
MS-MPIプログラムの作成方法

- 「構成プロパティ」→「C/C++」→「コード生成 」→
「ランタイム ライブラリ」→「マルチスレッド（/MT）」



MS-MPIプログラムの作成方法

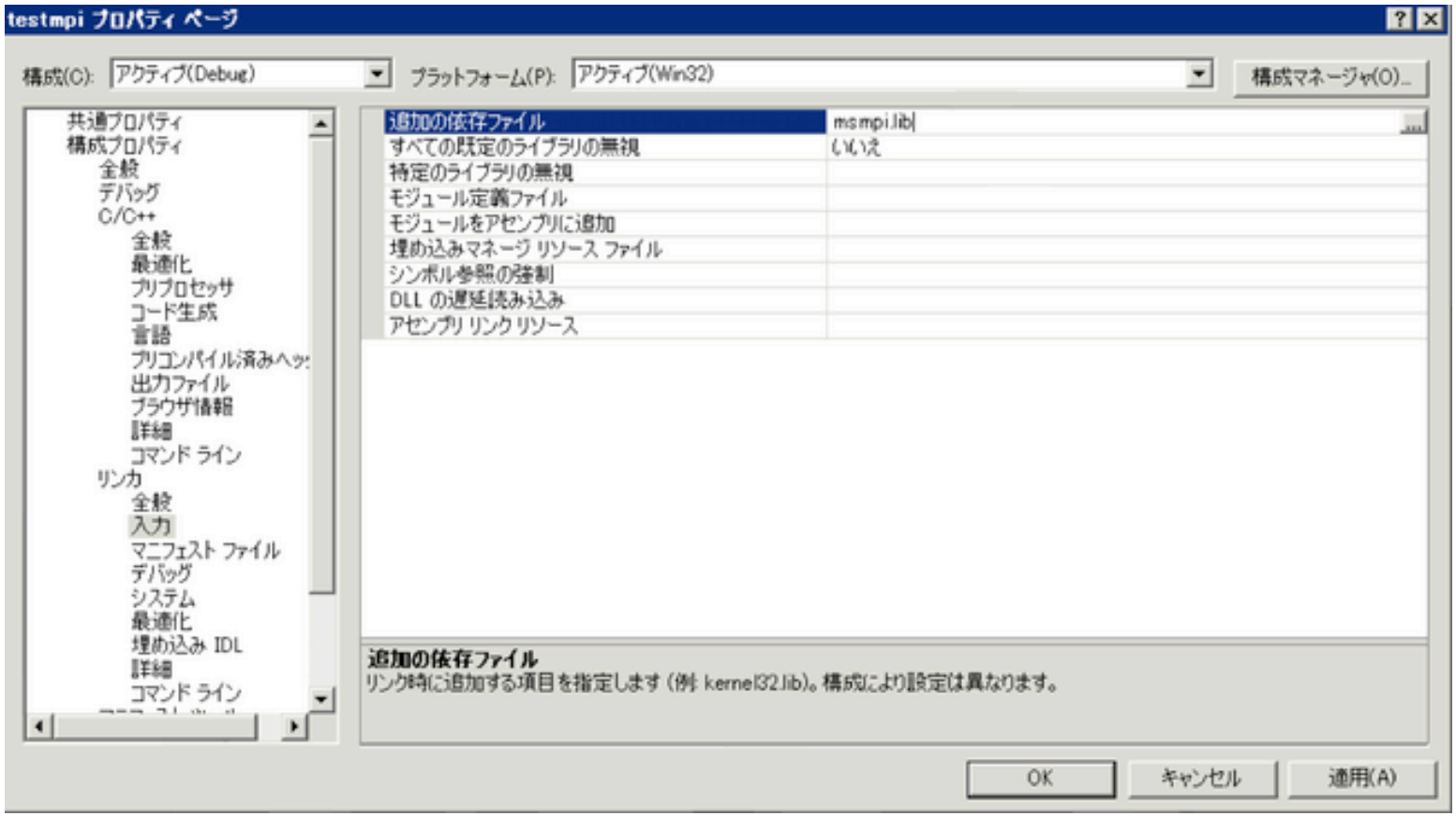
- リンカの設定
  「構成プロパティ」→「リンカ」→「全般」→「追加のライブラリディレクトリ」にリンクパスを追加する。



今回は「C:¥Program Files¥Microsoft HPC Pack 2008 SDK¥Lib¥i386」と入力した。

MS-MPIプログラムの作成方法

- 「リンカ」→「入力」→「追加の依存ファイル」に「msmpi.lib」と入力する。



MS-MPIプログラム例

- 次の3種類のMS-MPIアプリケーションを作成する
  - 通信を行わない並列アプリケーション
  - 1対1通信の並列アプリケーション
  - 集合通信を用いた並列アプリケーション

通信を行わない並列アプリケーション

- 自分自身のランクとマシン名を出力する

通信を行わない並列アプリケーション

```
#include <stdio.h>
#include “mpi.h”    // mpi用のヘッダファイルをインクルード

int main(int argc,char **argv){
   int rank, namelen;
   char hostname[MPI_MAX_PROCESSOR_NAME];

   MPI_Init(&argc, &argv); // 初期化
   MPI_Comm_rank(MPI_COMM_WORLD, &rank); // ランクを取得
   MPI_Get_processor_name(hostname,&namelen); // マシン名を取得

   printf("%d\t%s\n", rank, hostname);

   MPI_Finalize(); // 終了処理
   return 0;
}
```

通信を行わない並列アプリケーション

- ビルドが終了し、エラーがなければ、作成した並列アプリケーションの実行ファイルを共有フォルダにコピーする。
  - 実行ファイルはC:¥Documents and Settings¥（アカウント名）¥My Documents¥Visual Studio 2008¥Projects¥（プロジェクト名）¥Debug¥（プロジェクト名）.exeにある。
- Powershellを起動する。

通信を行わない並列アプリケーション

下記のようなコマンドを入力し、並列アプリケーションを実行する。

```
job submit /scheduler:(ジョブスケジューラ) /numcores:(計算に用いるコア数) /workdir:(実行ファイルのフォルダ)
/stdout:（出力ファイル）　 mpiexec （プロジェクト名）.exe
```

具体的な数値を入れた例は下記の通りである。

```
job submit /scheduler:192.168.0.1 /numcores:4
/workdir:\\192.168.0.1\share  /stdout:out.txt  mpiexec  test.exe
```

通信を行わない並列アプリケーション

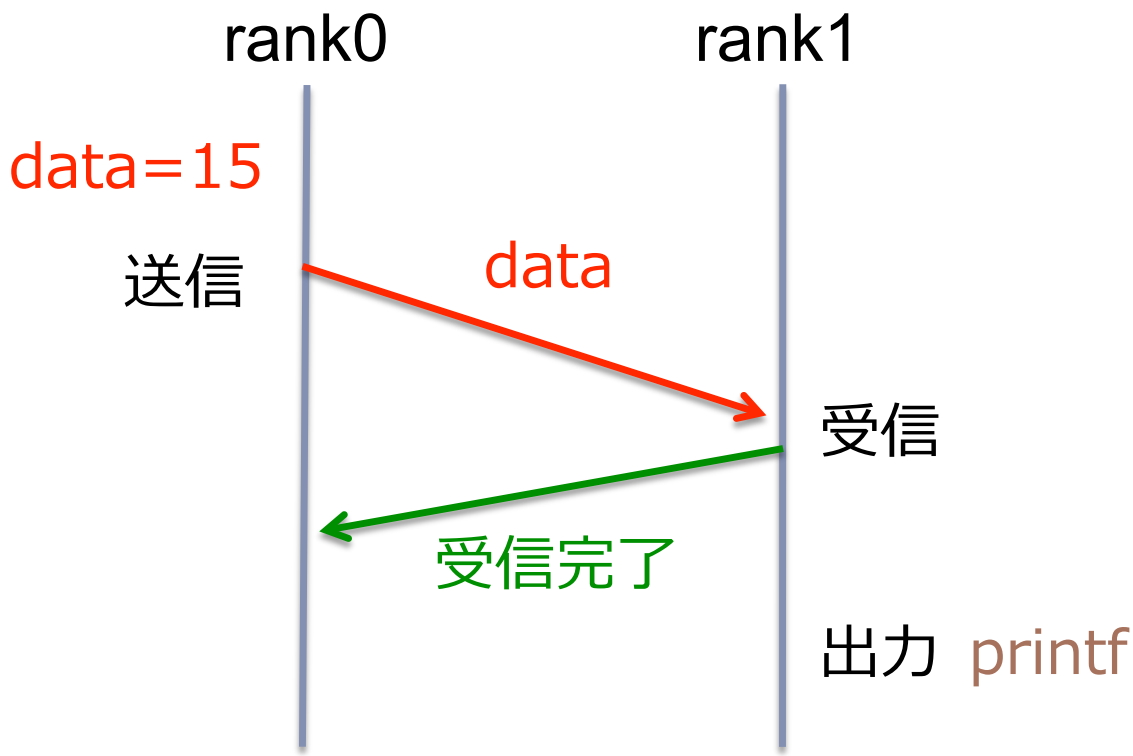
- 実行結果の例（出力ファイルに結果が入力される）

```
1       machine1.doshisha.ac.jp
3       machine3.doshisha.ac.jp
2       machine2.doshisha.ac.jp
0       machine0.doshisha.ac.jp
```

必ずしもランクの順に出力されないのは、
各プロセスが個別に実行されているからである。

1対1通信の並列アプリケーション

- 2台の計算機で実行する
- ランク0のデータをランク1に渡し、その値を出力する



1対1通信の並列アプリケーション

```
#include <stdio.h>
#include “mpi.h”    // mpi用のヘッダファイルをインクルード

int main(int argc,char **argv){
   int rank, tag = 999, data = 0;
   MPI_Status stat;

   MPI_Init(&argc, &argv); // 初期化
   MPI_Comm_rank(MPI_COMM_WORLD, &rank); // ランクを取得

   if( rank == 0 ){              // ランク0にだけ、dataの値を変える
            data = 15;
  }

   printf(“Before : %d\t%d\n”, rank, data); // 送信前のデータを出力
```

1対1通信の並列アプリケーション

```
 if( rank == 0 ){ // ランク0のデータをランク1に送信
    MPI_Send(&data, 1, MPI_INT, 1, tag, MPI_COMM_WORLD);
  } else if( rank == 1 ){
    MPI_Recv(&data, 1, MPI_INT, 0, tag, MPI_COMM_WORLD, &stat);
   }

 printf("After : %d\t%d\n", rank, data); // 送信後のデータを出力

 MPI_Finalize(); // 終了処理

 return 0;
}
```

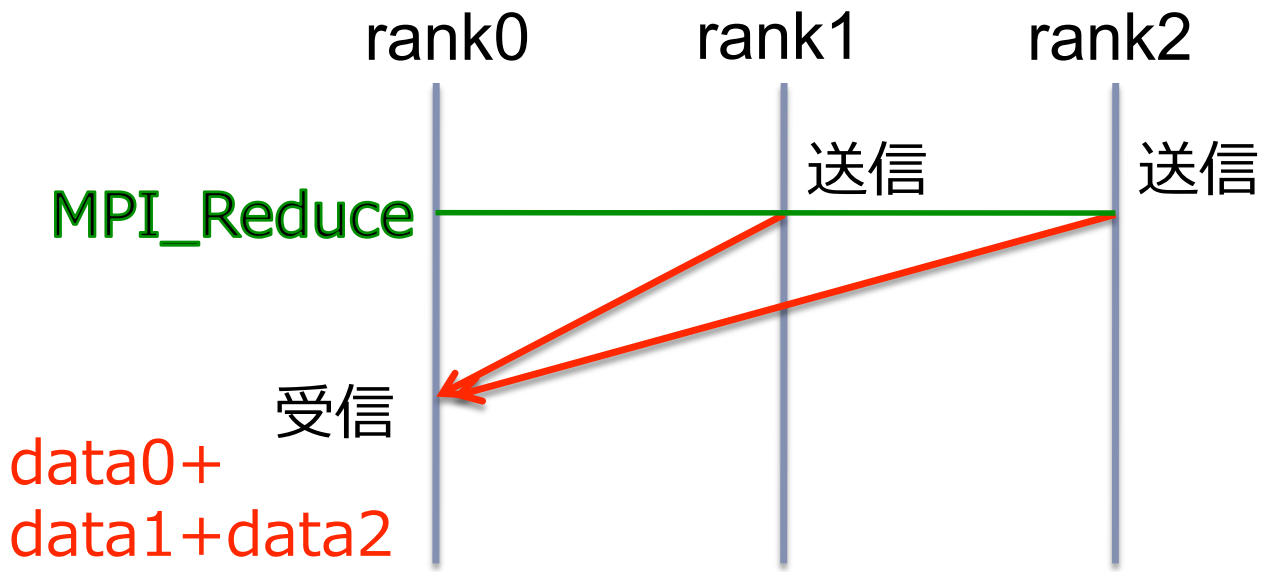
1対1通信の並列アプリケーション

- 実行結果の例

```
Before : 0       15
Before : 1       0
After : 0        15
After : 1        15
```

集合通信の並列アプリケーション

- 3台の計算機で実行する
- ランク1と2のデータをランク0に渡し、
  ランク0はすべてのデータの値を合計して出力する



集合通信の並列アプリケーション

```
#include <stdio.h>
#include “mpi.h”    // mpi用のヘッダファイルをインクルード

int main(int argc,char **argv){
   int rank, data = 0, sum = 0;
   MPI_Init(&argc, &argv); // 初期化
   MPI_Comm_rank(MPI_COMM_WORLD, &rank); // ランクを取得

   if( rank == 0 ){              // 各ランクにdataの値を設定
            data = 5;
  } else if (rank == 1){
            data = 3;
  } else if (rank == 2){
            data = 2;
 }
```

集合通信の並列アプリケーション

```
printf(“Before : %d\t%d\n”, rank, data); // 送信前のデータを出力

// ランク0にデータを送信する。その際に値を合計する
MPI_Reduce(&data, &sum, 1, MPI_INT, MPI_SUM, 0,
                    MPI_COMM_WORLD);

if(rank == 0){
  printf(“SUM : %d\n”, sum);  // 合計を出力
 }

 MPI_Finalize(); // 終了処理

 return 0;
}
```

集合通信の並列アプリケーション

- 実行結果の例（出力ファイルに結果が保存される）

```
Before : 0      3
Before : 1      2
Before : 2      5
SUM : 10
```

まとめ

- MS-MPIの概要説明
- Windows HPC Server上でのMS-MPIアプリケーションの作成方法の説明

参考URL

- 過去のWindowsHPC講習会の資料（MPI.NETなど）
  - http://pre-dawn.net/ja/?page_id=26
- MPI.NETの本家
  - http://www.osl.iu.edu/research/mpi.net/
- MS-MPIについて
  - http://technet.microsoft.com/ja-jp/library/cc967005.aspx