

MPICH2



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt "Nowoczesne metody i techniki kształcenia w UMCS. Wzmocnienie potencjału dydaktycznego Wydziału MFiI" współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego Program Operacyjny Kapitał Ludzki 2007-2013 Priorytet IV, Poddziałanie 4.1.1 "Wzmocnienie potencjału dydaktycznego uczelni"

Człowiek - najlepsza inwestycja

MPICH2

Programowanie współbieżne i rozproszone
A. Baran, 2010

LINK: <http://kft.umcs.lublin.pl/baran/prir/index.html>



MPICH2

[home](#) [about](#) [downloads](#) [documentation](#) [publications](#) [support](#) [release information](#)

MPICH2 is a **high-performance**
and **widely portable** implementation
of the **Message Passing Interface (MPI)**
standard (both MPI-1 and MPI-2).



The goals of MPICH2 are:

1. to provide an MPI implementation that efficiently supports different computation and communication platforms including **commodity clusters** (desktop systems, shared-memory systems, multicore architectures), **high-speed networks** (10 Gigabit Ethernet, InfiniBand, Myrinet, Quadrics) and **proprietary high-end computing systems** (Blue Gene, Cray).
2. to enable cutting-edge research in MPI through an **easy-to-extend** modular framework for other derived implementations.

NEWS & EVENTS

MPICH2-1.3 released

A new stable release of MPICH2, 1.3, is

RELEASES

The current stable release for MPICH2 is 1.3.
It was released on October 22, 2010.

PARTNERS/COLLABORATORS

MPICH2 collaborates with a number of
partner institutes. Several of these are

<http://www.mcs.anl.gov/research/projects/mpich2/>



MPICH2 + ...



PRZYKŁADY ...



hello.f90

FORTRAN - plik hello.f90

```
program hello
  integer nthreads, myid, omp_get_thread_num, omp_get_num_threads
!make the values of nthreads and myid private to each thread
!$omp parallel private(nthreads, myid)
  myid = omp_get_thread_num()
  print *, ' Hello I am thread ', myid
! If I am the master node print number of threads
  if (myid .eq. 0) then
    nthreads = omp_get_num_threads()
    print *, ' Number of threads = ', nthreads
  end if
!$omp end parallel
end
```



hello.c

```
#include <omp.h>
int main() {
    int nthreads, myid;
    /* make the values of nthreads and myid private to each thread */
    #pragma omp parallel private (nthreads, myid)
    {
        myid = omp_get_thread_num();
        printf("Hello I am thread %d\n", myid);
        /* only master node print the number of threads */
        if (myid == 0) {
            nthreads = omp_get_num_threads();
            printf("Number of threads = %d\n", nthreads);
        }
    }
    return 0;
}
```



Przykład. Fortran.

```
PROGRAM EXAMPLE
    INTEGER A(0:1)
    INTEGER AA(1000), B(1000), C(1000)
    INTEGER omp_get_thread_num
    A(0) = -1
    A(1) = -1
!$OMP PARALLEL
    A(omp_get_thread_num()) = omp_get_thread_num()
!$OMP MASTER
    PRINT *, "YOU SHOULD ONLY SEE THIS ONCE"
!$OMP END MASTER
!$OMP DO
    DO I=1, 1000
        AA(I) = B(I) + C(I) + 10000 *J
    ENDDO
!$OMP END DO
!$OMP END PARALLEL
    PRINT *, "A(0)=",A(0), " A(1)=",A(1)
END
```



Przykład. Fortran.



```
include "mpif.h"
  double precision startt, stopt
  double precision PI25DT
  parameter (PI25DT = 3.141592653589793238462643d0)
  double precision mypi, pi, h, sum, x, f, a
  integer n, myid, numprocs, i, rc, ierr
c function to integrate
  f(a) = 4.d0 / (1.d0 + a*a)
  call MPI_INIT( ierr )
  call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
  call MPI_COMM_SIZE( MPI_COMM_WORLD, numprocs, ierr )
  print *, "Process ", myid, " of ", numprocs, " is alive"
10  if ( myid .eq. 0 ) then
    write(6,98)
98  format('Enter the number of intervals: (0 quits)')
    read(5,99) n
99  format(i10)
    endif
  startt = MPI_WTIME(ierr)
  call MPI_BCAST(n,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
c check for quit signal
  if ( n .le. 0 ) goto 30
c calculate the interval size
  h = 1.0d0/n
  sum = 0.0d0
```



Przykład. Fortran.



```
!$OMP PARALLEL DO
!$OMP& SHARED(sum,n,h,myid,numprocs)
!$OMP& PRIVATE(i,x)
      do i = myid+1, n, numprocs
          x = h * (dble(i) - 0.5d0)
!$OMP CRITICAL
          sum = sum + f(x)
!$OMP END CRITICAL
      end do
!$OMP END PARALLEL DO
      mypi = h * sum
c collect all the partial sums
      call MPI_REDUCE(mypi,pi,1,MPI_DOUBLE_PRECISION,MPI_SUM,0,
$          MPI_COMM_WORLD,ierr)
      stopt = MPI_WTIME(ierr)
c node 0 prints the answer.
      if (myid .eq. 0) then
          write(6, 97) pi, abs(pi - PI25DT)
97      format(' pi is approximately: ', F18.16,' Error is: ', F18.16)
          write(*,*) 'Total Time = ', stopt-startt
          endif
          goto 10
30      call MPI_FINALIZE(rc)
          stop
          end
```




Problemy...?

MPICH2

