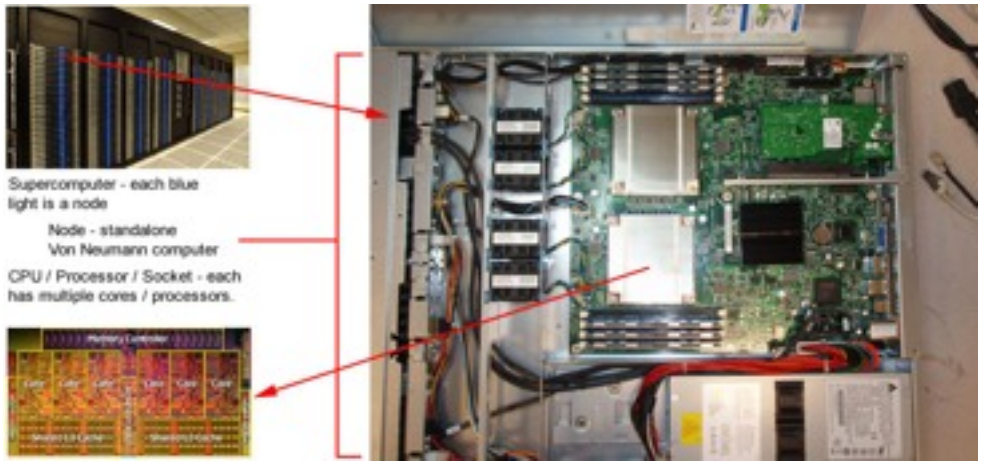
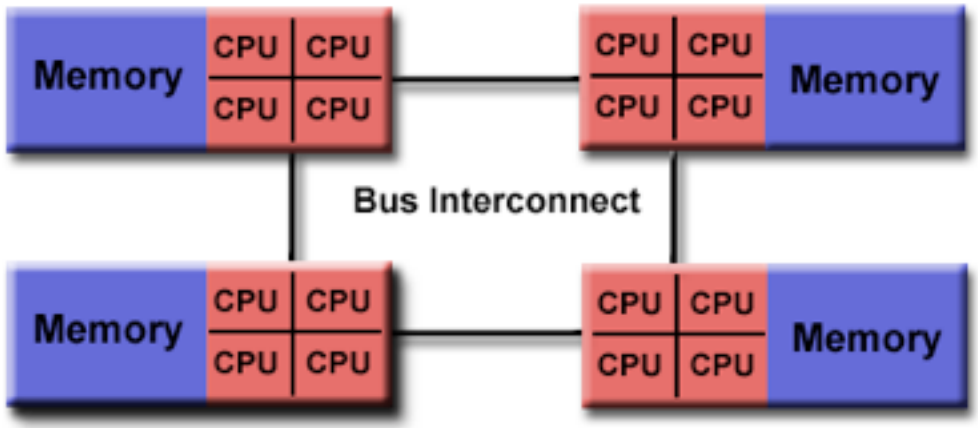
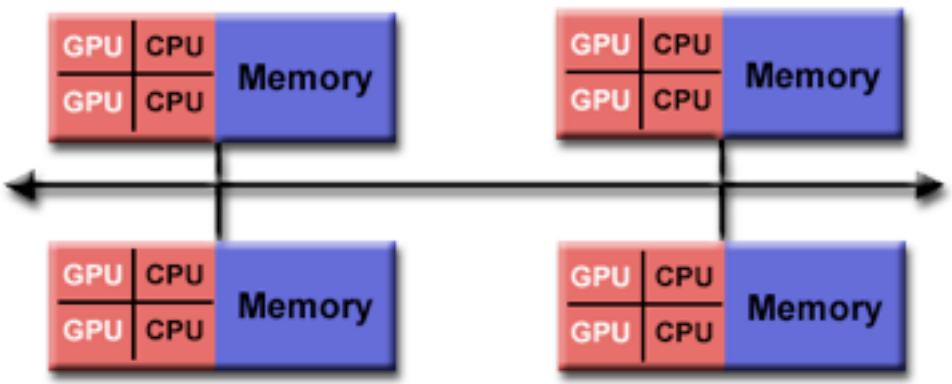
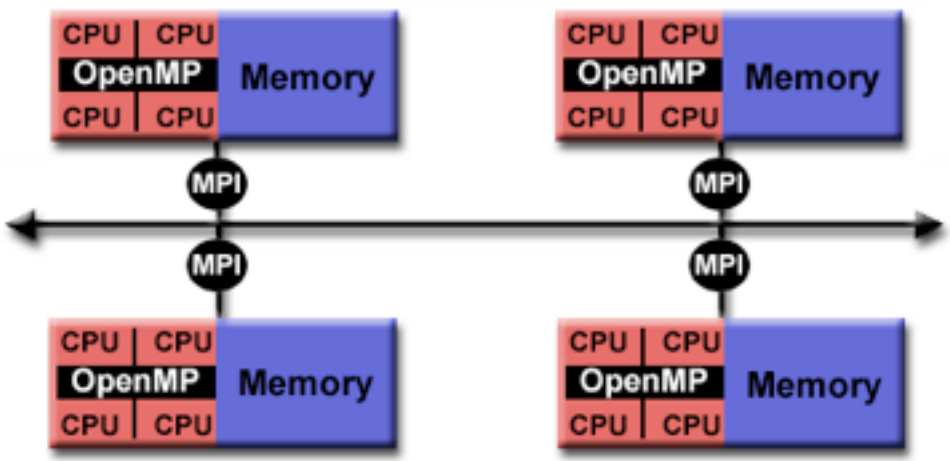
GPU a szuperszámítógépben, CUDA Fortran wrapper









"ACCelerator"=PPC,MIC, GPU, Cell, FPGA
Compute Node Linux
Scheduler

computing.llnl.gov/linux/slurm

```
sudo apt-get install slurm
```

slurm.conf

```
# U36
NodeName=gpu  Weight=104 RealMemory=34000
Sockets=2 CoresPerSocket=4 Gres=gpu:2
Feature=gpu,compute,all
```

gres.conf

```
Name=gpu File=/dev/nvidia0 CPUs=0-3
Name=gpu File=/dev/nvidia1 CPUs=4-7
```

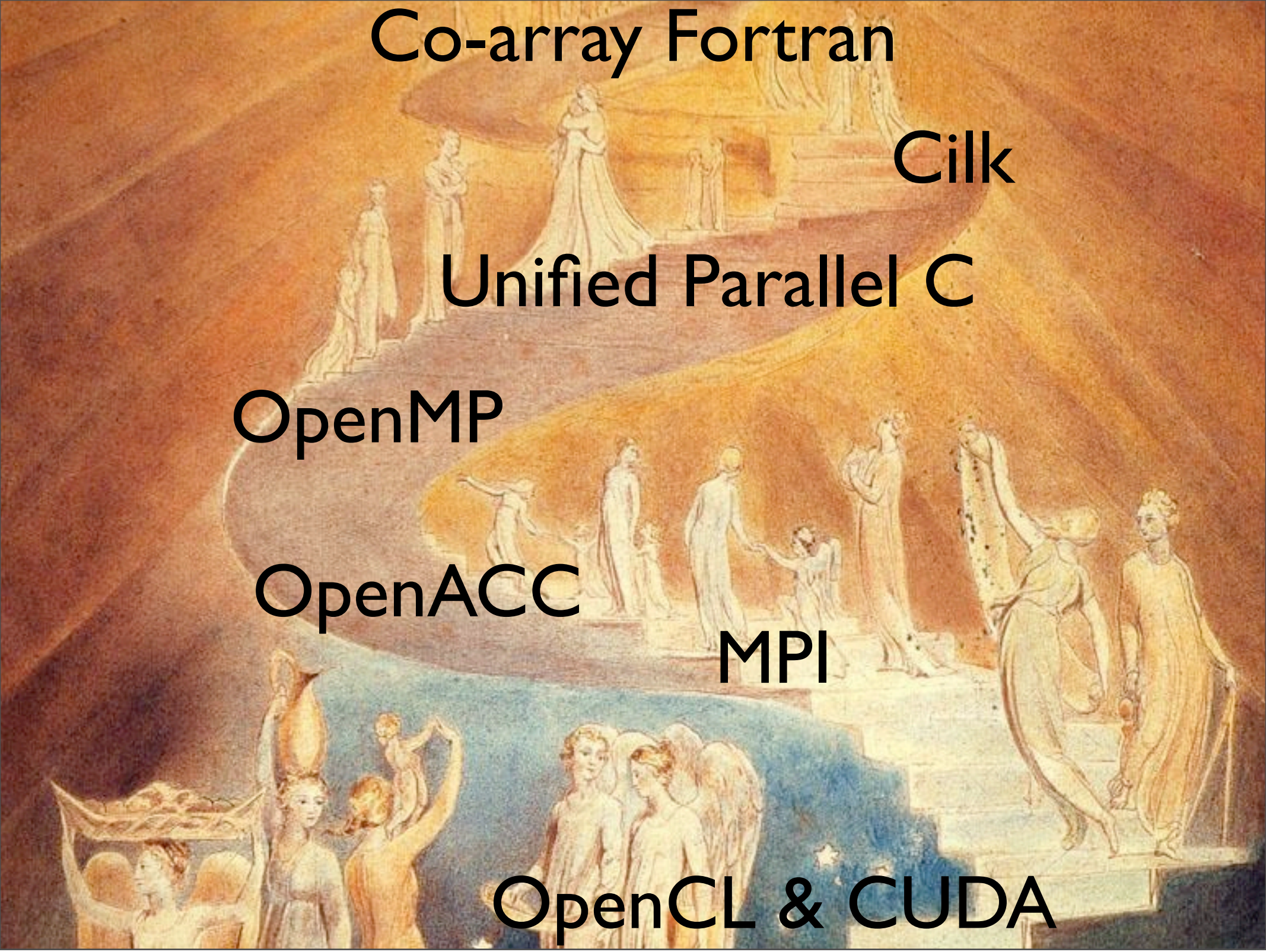
```
NodeName=gpu Arch=x86_64 CoresPerSocket=4
   CPUAlloc=0 CPUErr=0 CPUTot=8 Features=gpu,compute,all
   Gres=gpu:2
   OS=Linux RealMemory=34000 Sockets=2
   State=IDLE ThreadsPerCore=1 TmpDisk=0 Weight=104
   BootTime=2012-05-31T10:04:28 SlurmdStartTime=2012-06-23T15:09:28
   Reason=(null)
```

$ srun -p devel -n 1 -B 2:2 --gres=gpu:2 ./sanity_check.sh

```
CUDA_VISIBLE_DEVICES=0,1
[gpu][0]:Device name=Tesla M2075, global memory size=5636554752
[gpu][1]:Device name=Tesla M2075, global memory size=5636554752
```

```
4085     ****         4 devel     R                0:05  gpu
```

```
| Compute processes:                                                GPU Memory |
|  GPU  PID     Process name                                        Usage      |
|=============================================================================|
|  0.  3316     ...***********/gpu/cuda_memtest-1.2.3/./cuda_memtest 10662MB  |
|  1.  3316     ...***********/gpu/cuda_memtest-1.2.3/./cuda_memtest 10662MB  |
```


Co-array Fortran
Cilk
Unified Parallel C
OpenMP
OpenACC
MPI
OpenCL & CUDA

Makefile
arch.make.gnu
arch.make.intel
cufu
fuda_test.F
fuda_test.sh
fudadevice.F
fudadevice.c
fudaerror.F
fudaerror.c

```
C_SRC   = fudadevice.c fudaerror.c
C_OBJ   = $(C_SRC:.c=.o)
LIB     = fuda.a
LDFLAGS = $(LIB) -L$(CUDA_LIB) -lcudart

F_SRC  = $(C_SRC:.c=.F)
F_MOD  = $(F_SRC:.F=.mod)

all: $(LIB) fuda_test

$(LIB): $(C_OBJ)
     ar vq $(LIB) $(C_OBJ)

fuda_test: $(F_MOD) fuda_test.o
     $(FC) -o fuda_test fuda_test.o $(LDFLAGS)

.c.o:
     @echo "C Object:" $@
     $(CC) -c $(CFLAGS) $< -o $@

.F.o:
     @echo "F Object:" $@
     $(CPP)
     $(FC) $(FFLAGS) -c $*.f90 -o $@

.F.mod:
     @echo "F Module:" $@
     $(CPP)
     $(FC) $(FFLAGS) -c $*.f90
```

fudadevice.F

```
module fudadevice
  use iso_c_binding
  include "cudadevice.h"
  interface
    subroutine fudaChooseDevice(device,prop,ierr)
      use iso_c_binding
      import cudaDeviceProp
      integer(c_int) :: device,ierr
      type(cudaDeviceProp)   :: prop
    end subroutine
...
    subroutine fudaSetValidDevices(device_arr,len,ierr)
      use iso_c_binding
      integer(c_int), dimension(*) :: device_arr
      integer(c_int) :: len,ierr
    end subroutine
```

fudadevice.c

```
#include <driver_types.h>

extern cudaError_t cudaChooseDevice ( int *, const
struct cudaDeviceProp * );
void fudachoosedevice_( int *device, const struct
cudaDeviceProp *prop, int *ierr ) {
  *ierr = cudaChooseDevice( device, prop );
}
```

fudaerror.F

```
    subroutine fudaGetErrorString(buf,ierr)
      use iso_c_binding
      integer(c_int) :: ierr
      character(*,c_char),intent(out) :: buf
    end subroutine
```

```
extern const char *cudaGetErrorString (cudaError_t);
void fudageterrorstring_( char *buf, int *ierr ) {
  const char *err=cudaGetErrorString(*ierr);
  strcpy(buf, err);
}
```

```
#include "cudaerror.h"

program fuda_test
  use fudadevice
  implicit none
  integer :: count,ierr,i
  integer :: device
  character(len=128) :: error = ''
  character(len=512) :: fmt_prop
  type(cudaDeviceProp) :: prop

  call fudaGetDeviceCount(count,ierr)
  if (ierr.ne.CUDA_SUCCESS) then
    call fudaGetErrorString(error,ierr)
    print*,trim(error)
    stop
  end if

...

  do device=0,count-1
    call fudaGetDeviceProperties(prop,device,ierr)
    if (ierr.ne.CUDA_SUCCESS) then
      call fudaGetErrorString(error,ierr)
      print*,error
      stop
    end if
    print(fmt_prop),device,prop%name,&
                    prop%regsPerBlock,prop%warpSize,&
                    prop%maxThreadsPerBlock,(prop%maxThreadsDim(i),i=1,3),&
                    prop%pciBusID,prop%pciDeviceID,prop%pciDomainID
  end do

end program fuda_test
```

github.com/hornos/fuda

github.com/hornos/fmp

github.com/hornos/shf3