Parallelization of GISS ModelE
MPI etc.

Igor Aleinov

NASA GISS

CCSR, Columbia University

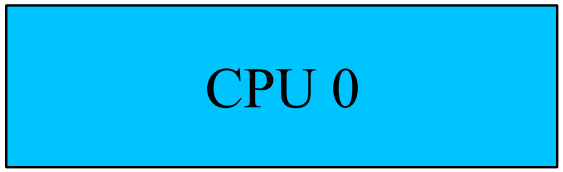
Developers

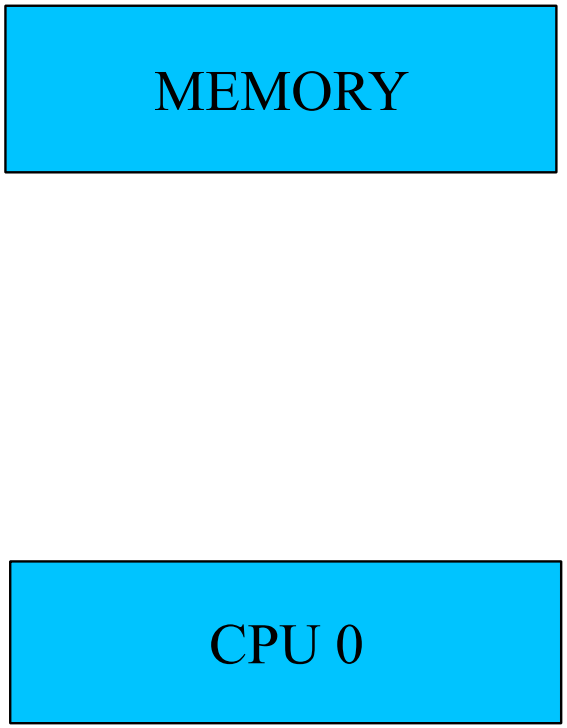
GISS
Igor Aleinov, Gavin Schmidt, Reto Ruedy,
Max Kelley, Nick Tausnev

GSFC
Tom Clune, Hamid Oloso, Maharaj Bhat

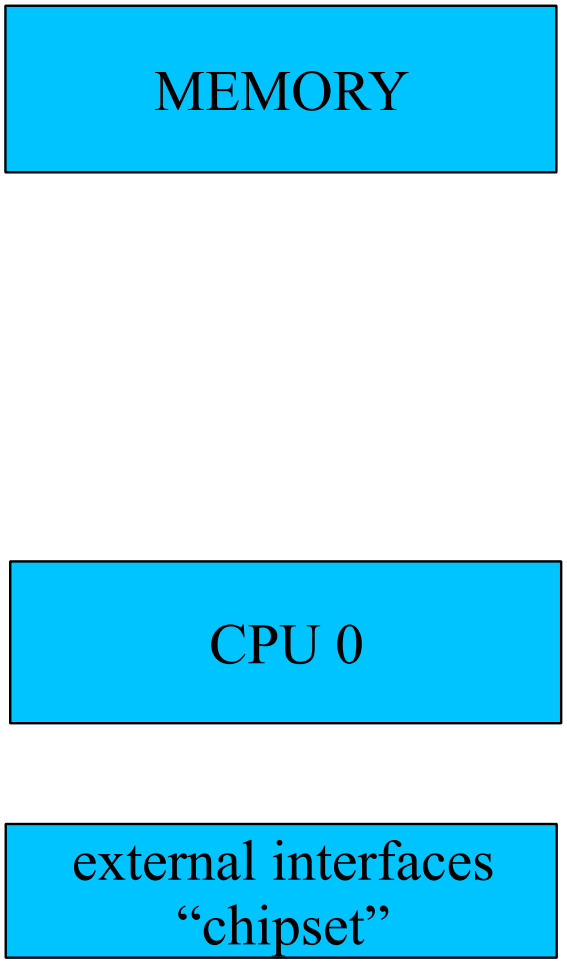
Hardware

Computer

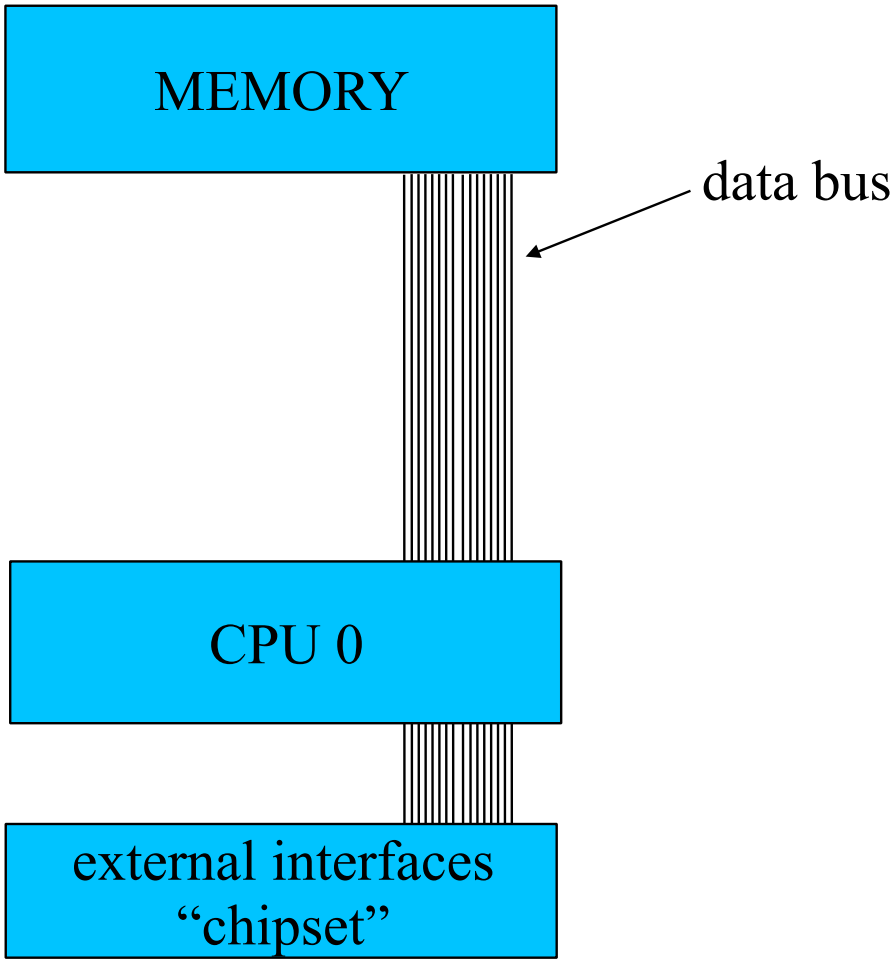



Computer
MEMORY
CPU 0

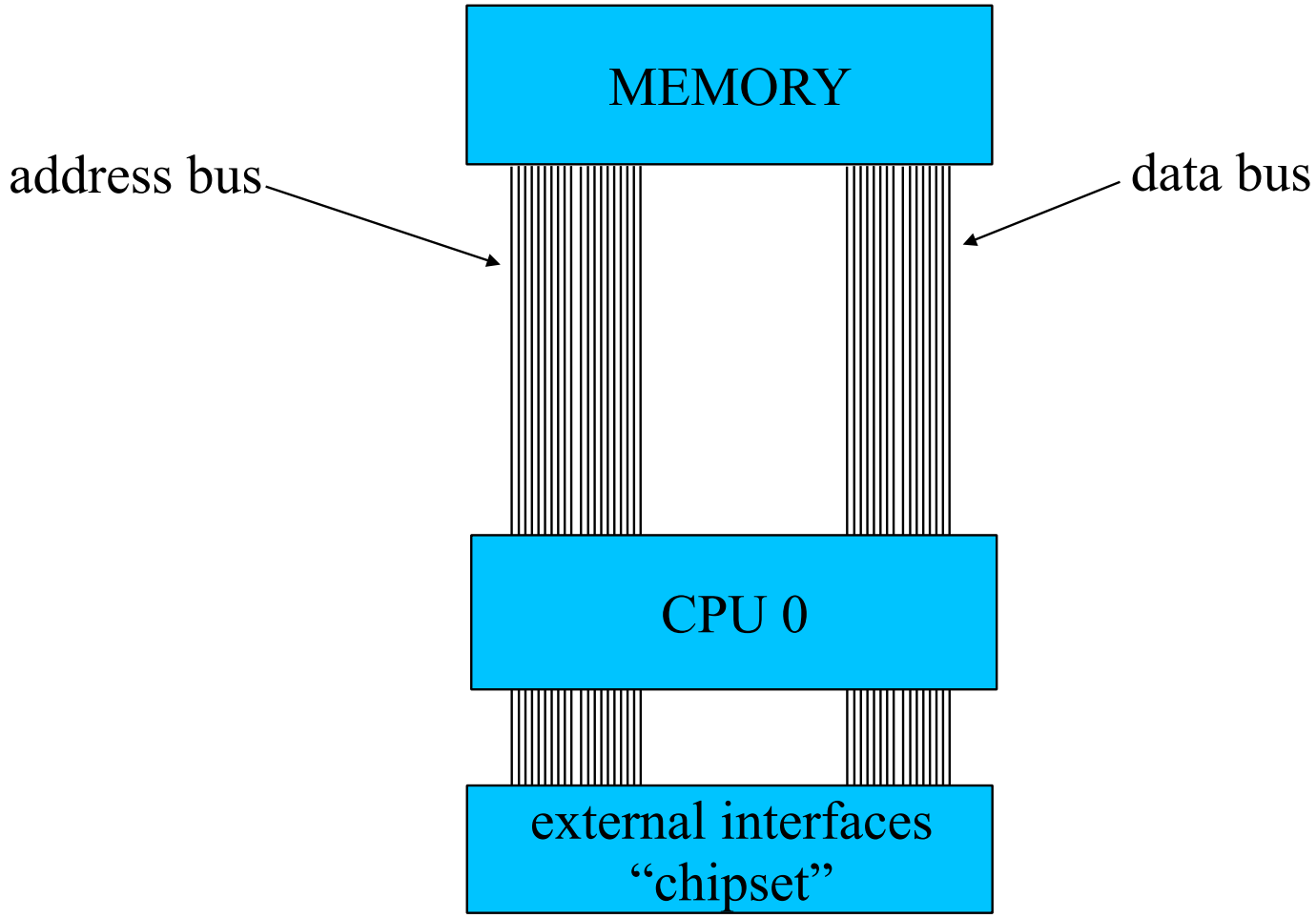
Computer



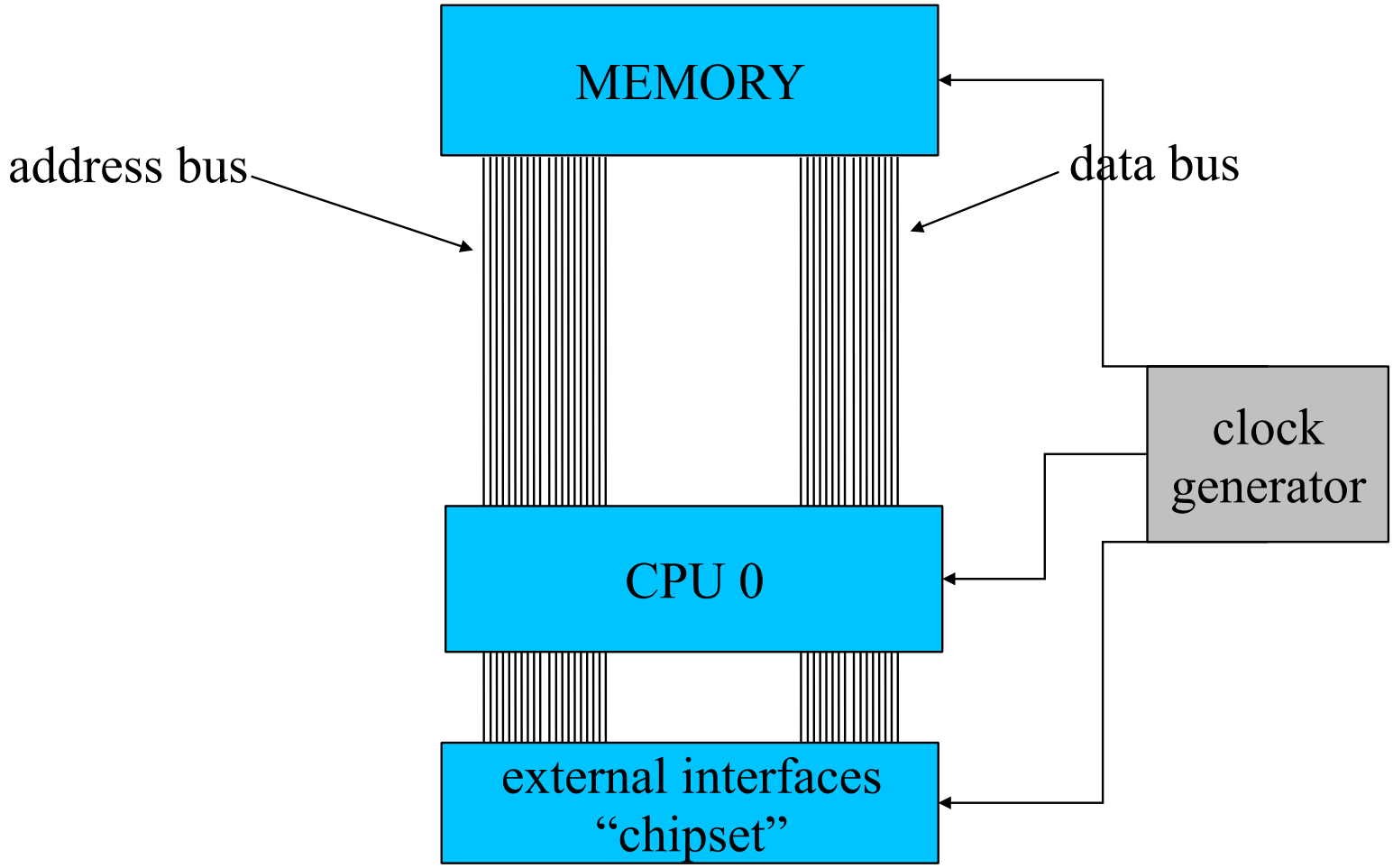
Computer



Computer


MEMORY
address bus
data bus
CPU 0
external interfaces
“chipset”

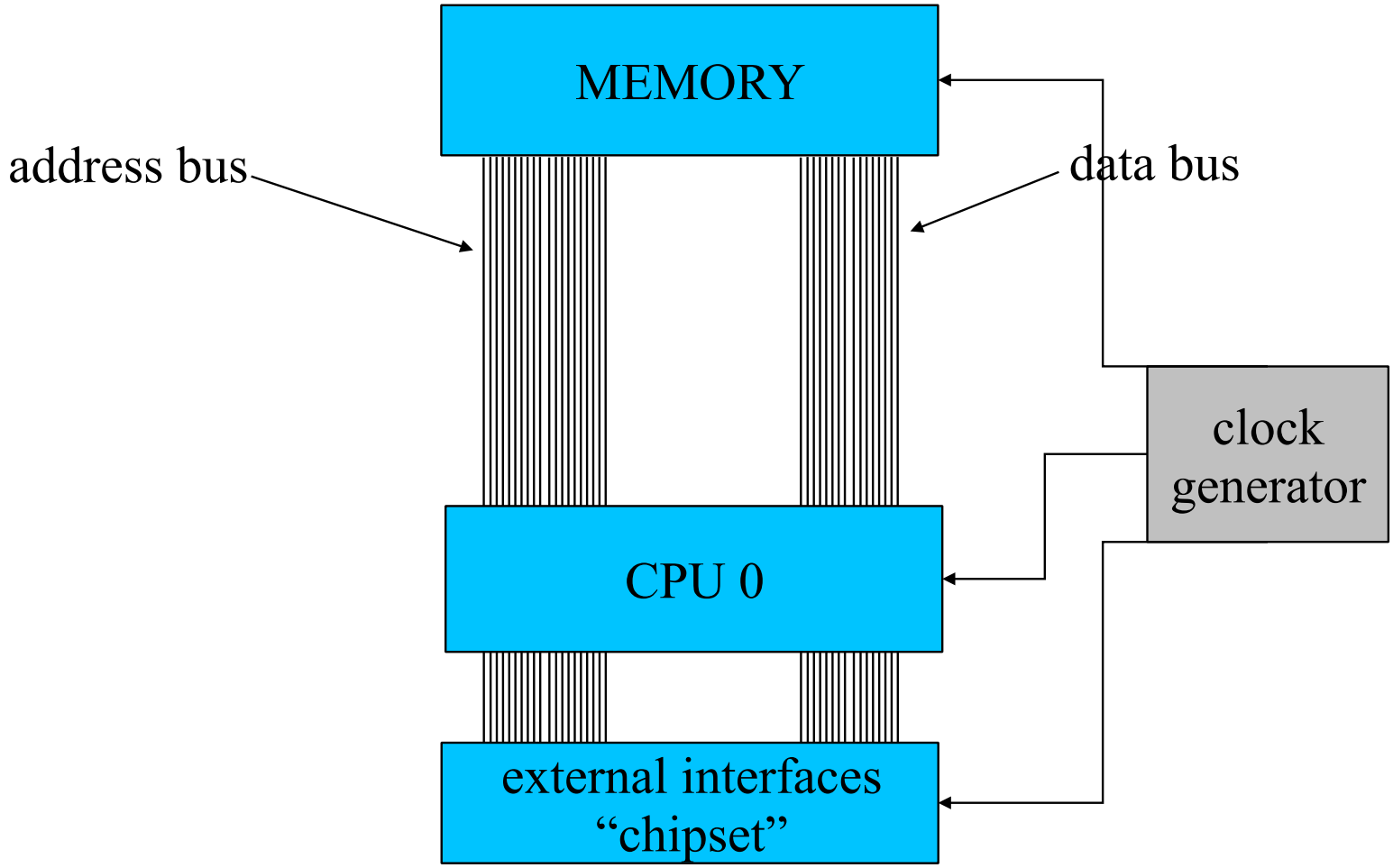
Computer


MEMORY
address bus
data bus
clock generator
CPU 0
external interfaces "chipset"

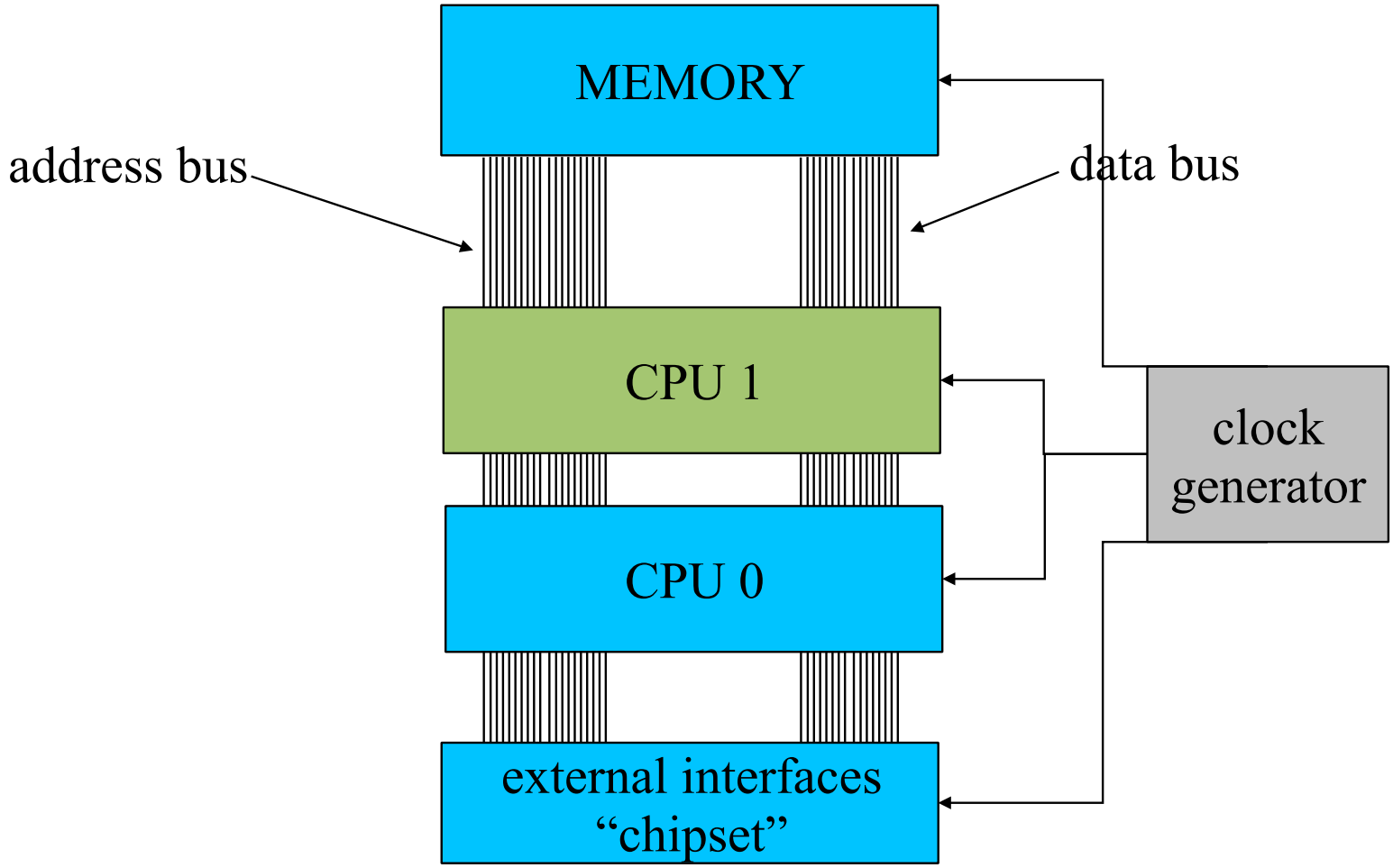
Computer

MEMORY
address bus
data bus
clock generator
CPU 0
external interfaces
“chipset”

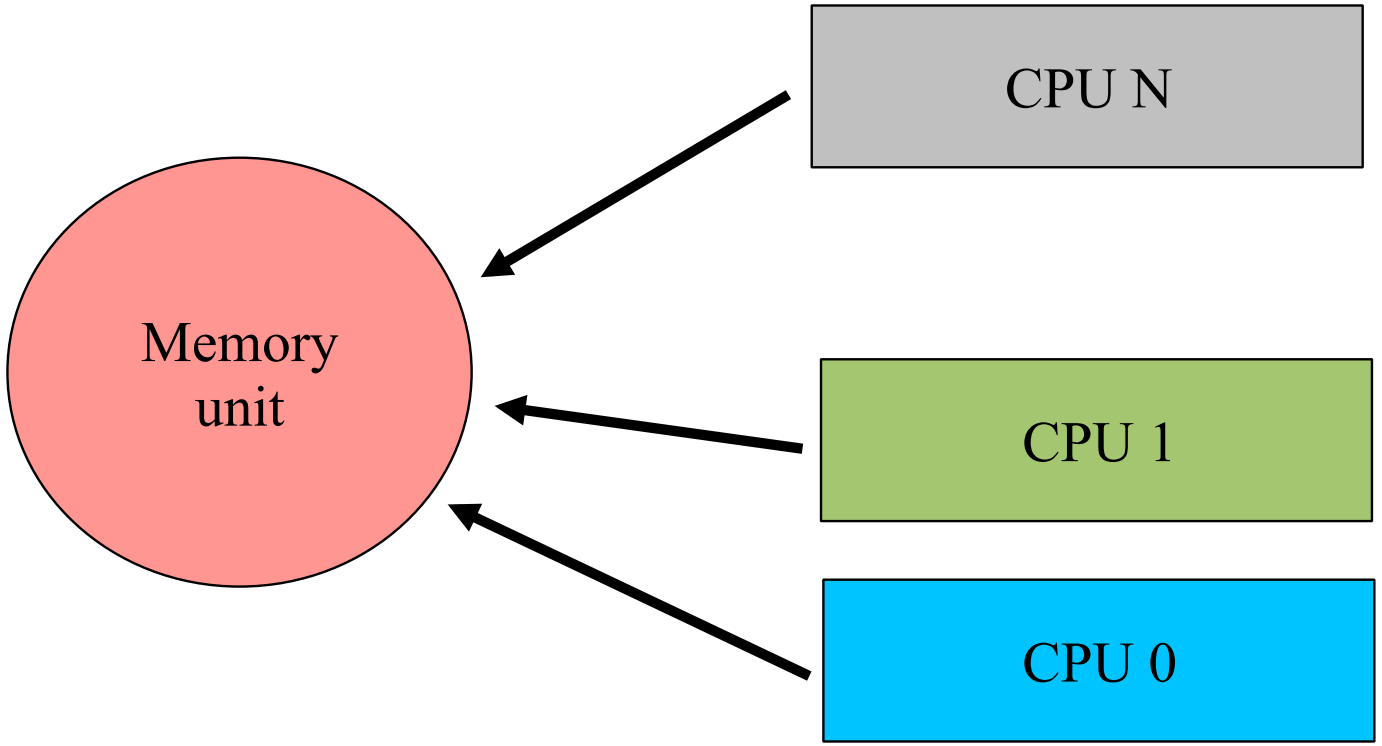
Computer


MEMORY
address bus
data bus
clock generator
CPU 0
external interfaces "chipset"

Computer


MEMORY
address bus
data bus
CPU 1
clock
generator
CPU 0
external interfaces
“chipset”

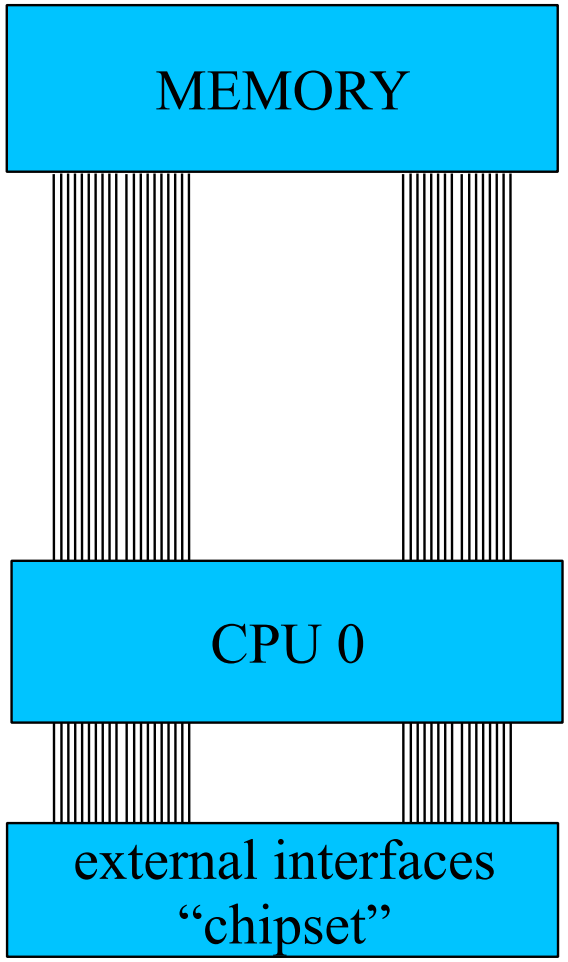
Shared memory architecture



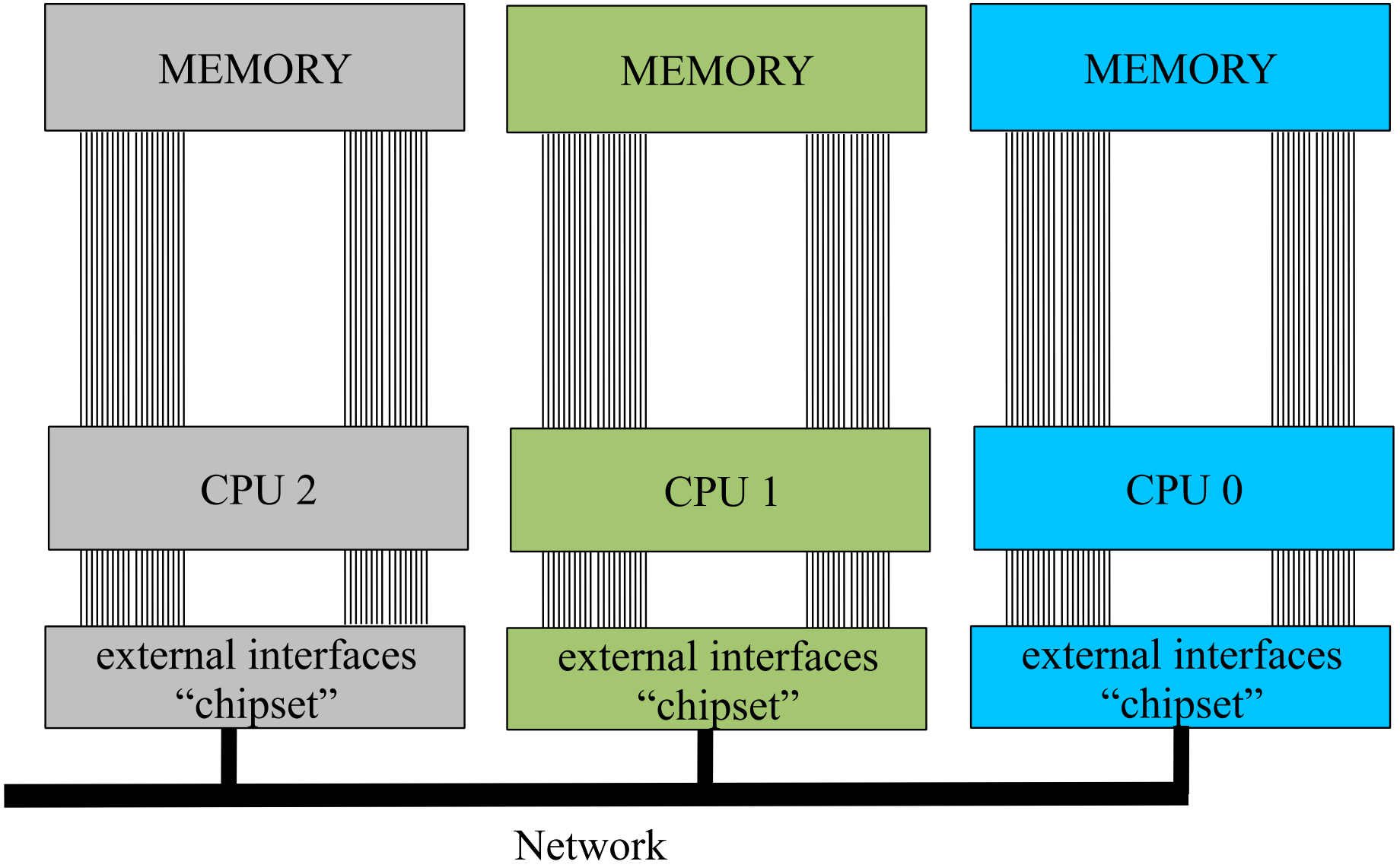
bus frequency = 800 MHz

distance = 300E6 / 800E6 = 0.37 m

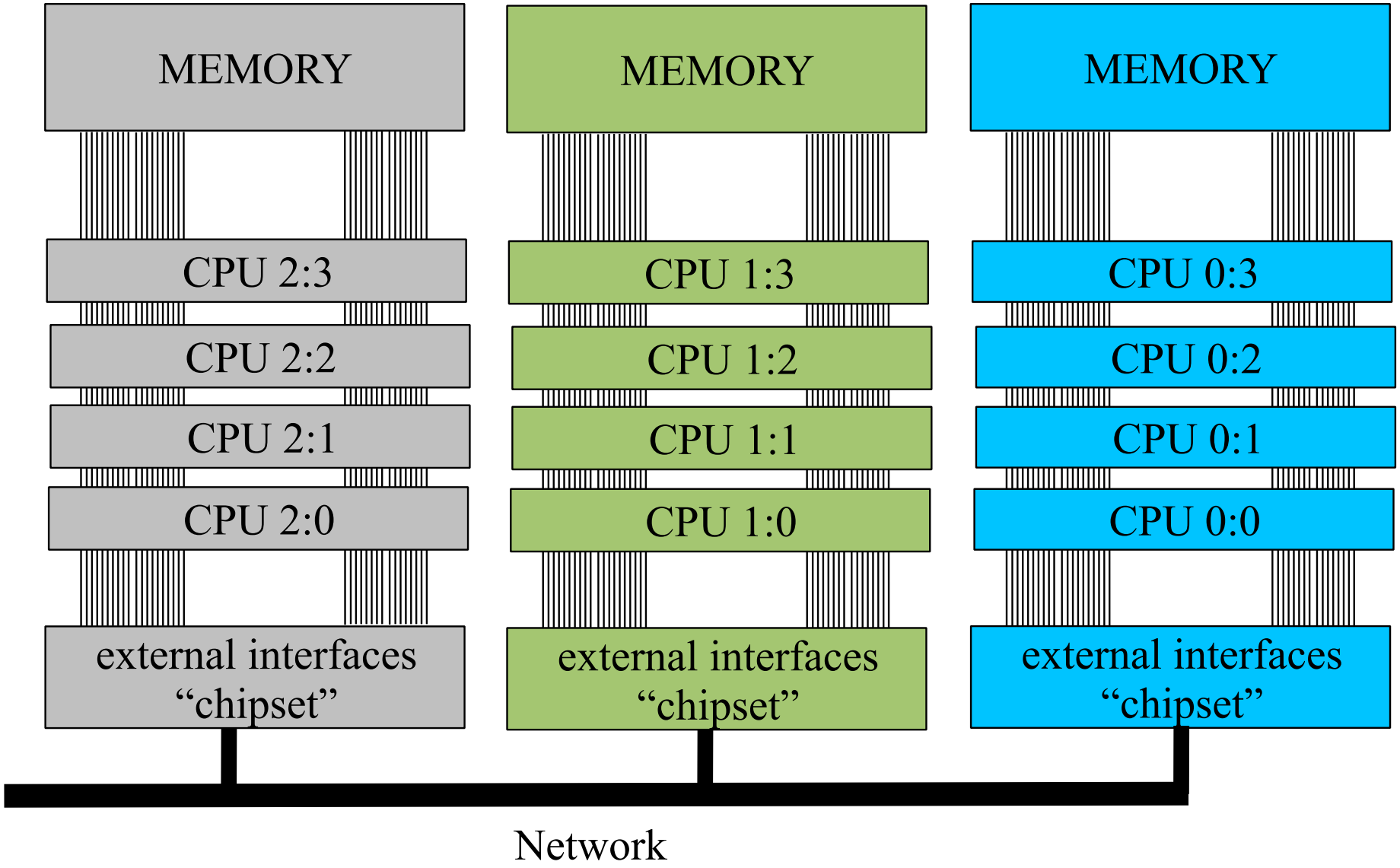
Distributed memory architecture



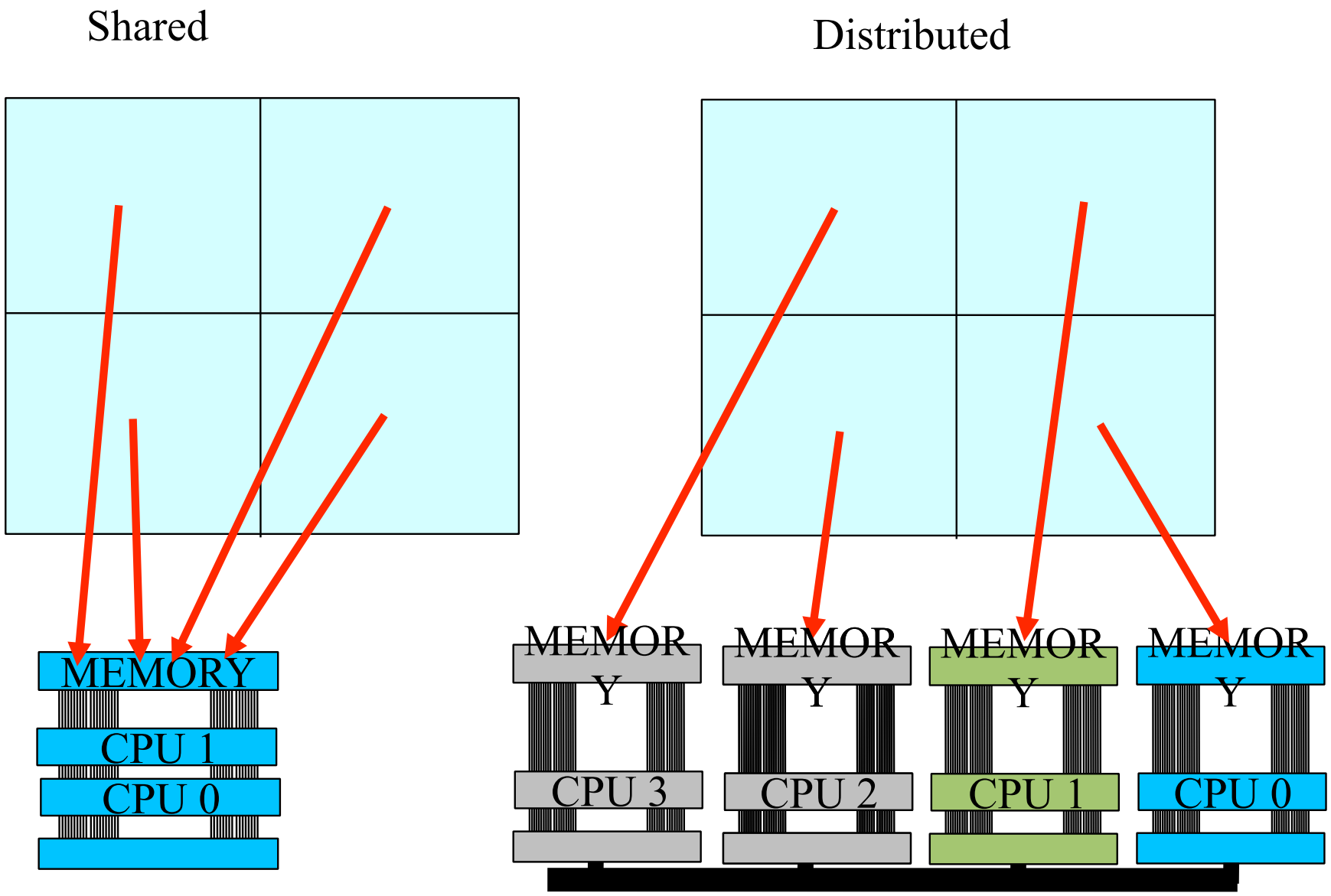
Distributed memory architecture

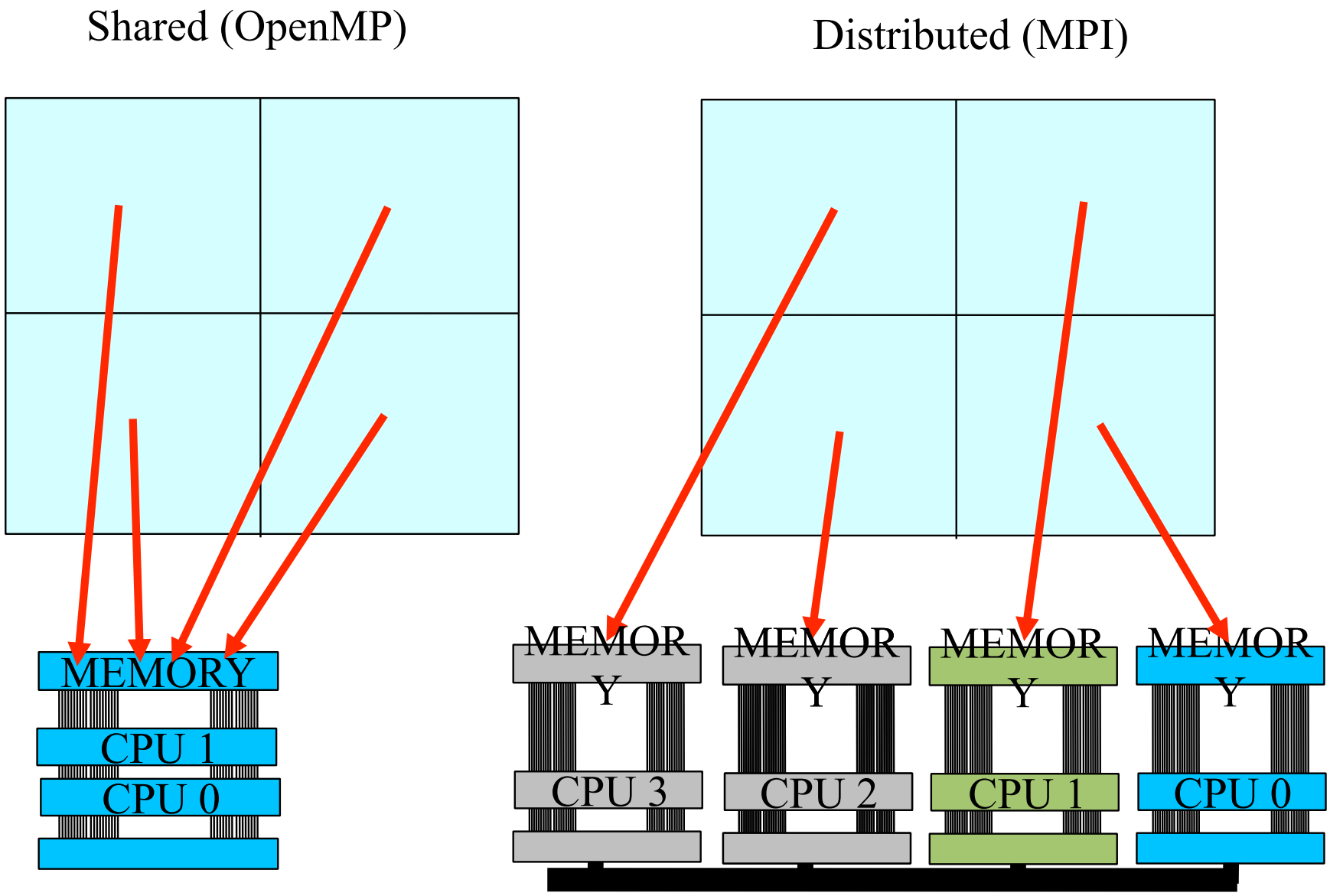


“Discover”

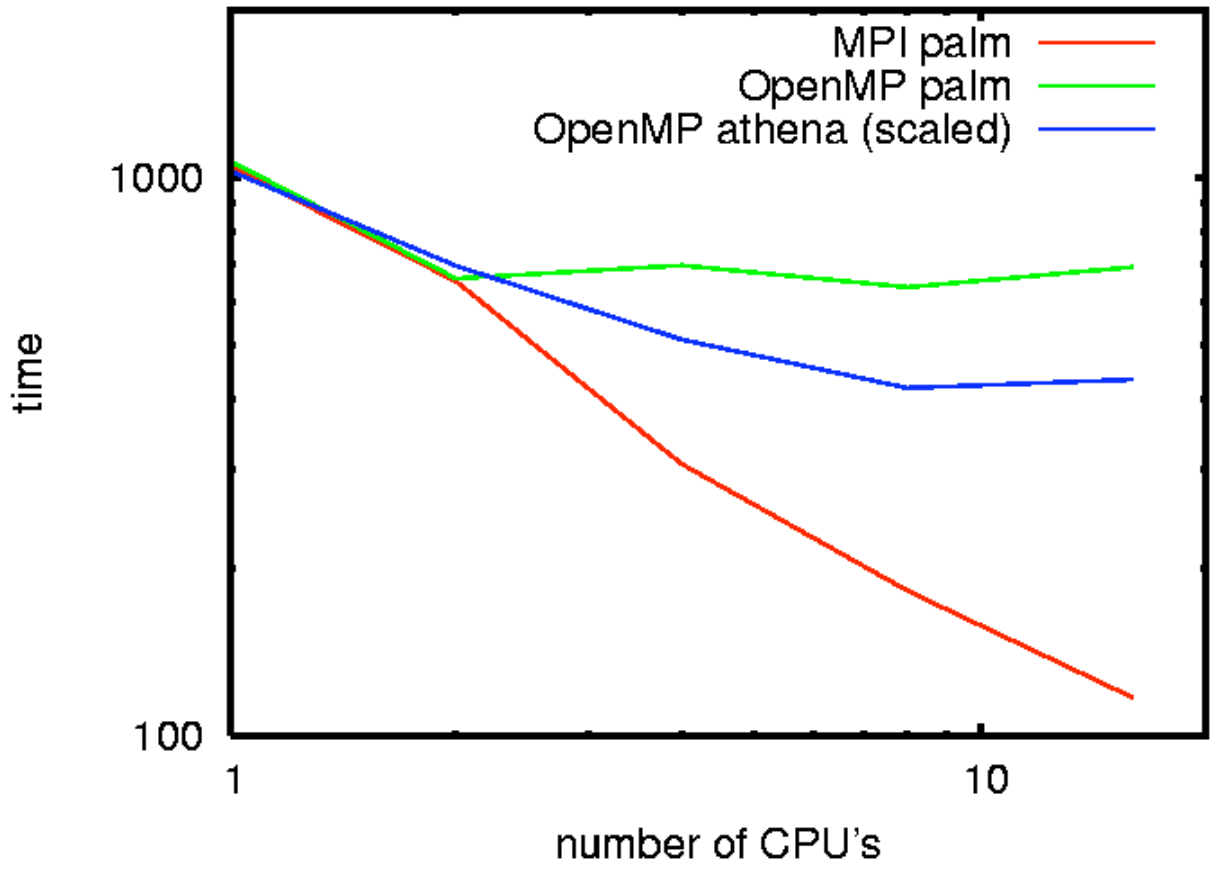

MEMORY
MEMORY
MEMORY
CPU 2:3
CPU 2:2
CPU 2:1
CPU 2:0
CPU 1:3
CPU 1:2
CPU 1:1
CPU 1:0
CPU 0:3
CPU 0:2
CPU 0:1
CPU 0:0
external interfaces “chipset”
external interfaces “chipset”
external interfaces “chipset”
Network

“Discover”

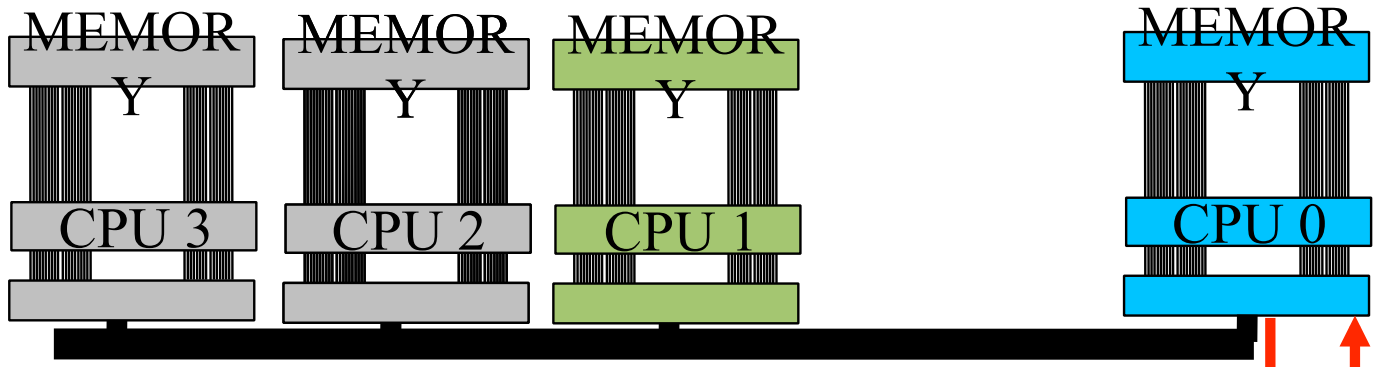

Shared
Distributed
MEMORY
CPU 1
CPU 0
MEMOR Y
MEMOR Y
MEMOR Y
MEMOR Y
CPU 3
CPU 2
CPU 1
CPU 0


Shared (OpenMP)
Distributed (MPI)
MEMORY
CPU 1
CPU 0
MEMOR Y
MEMOR Y
MEMOR Y
MEMOR Y
CPU 3
CPU 2
CPU 1
CPU 0

MPI vs OpenMP (E1M20)



Message Passing Interface (MPI)

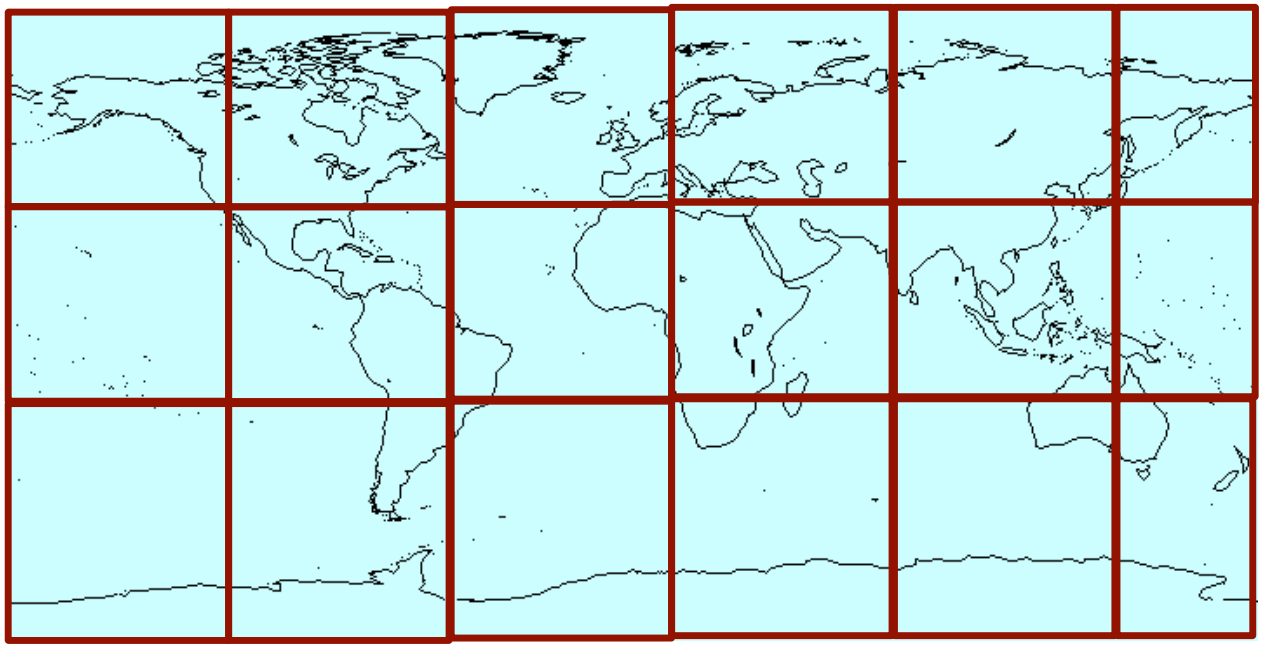


MPI_SEND(BUF, COUNT, DATATYPE, DEST,
TAG, COMM, IERR)

MPI_RECV(BUF, COUNT, DATATYPE, SOURCE,
TAG, COMM, STATUS, IERR)

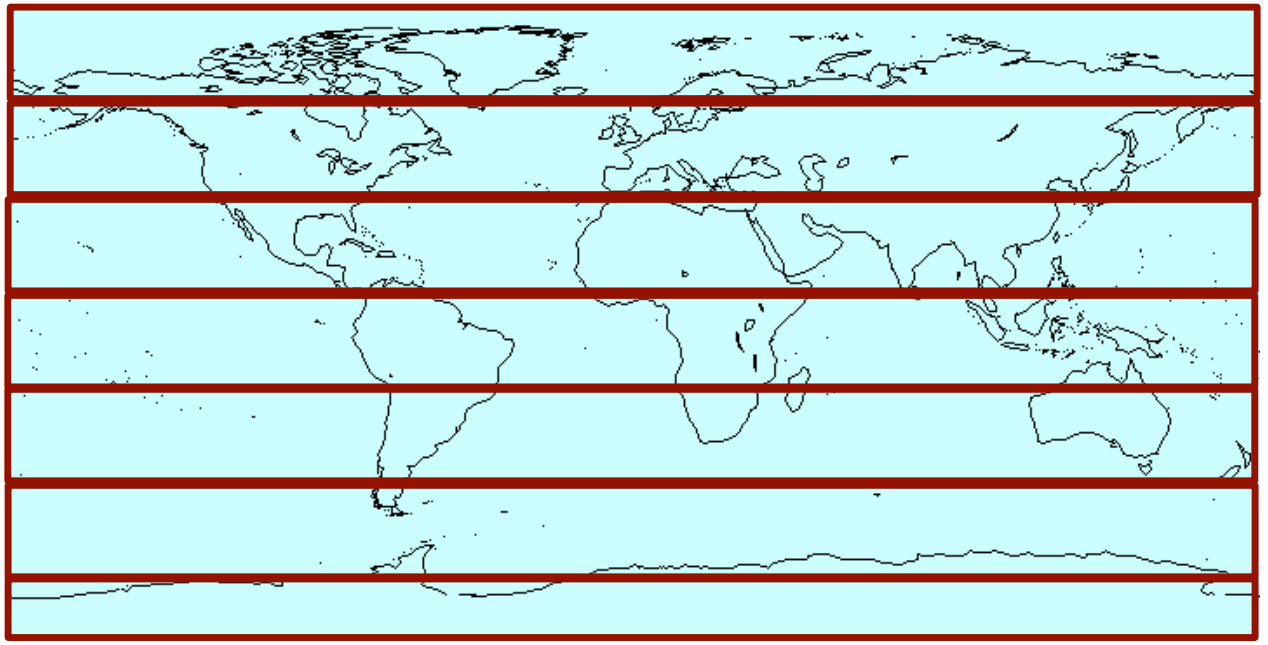
Domain Decomposition

2-Dimensional



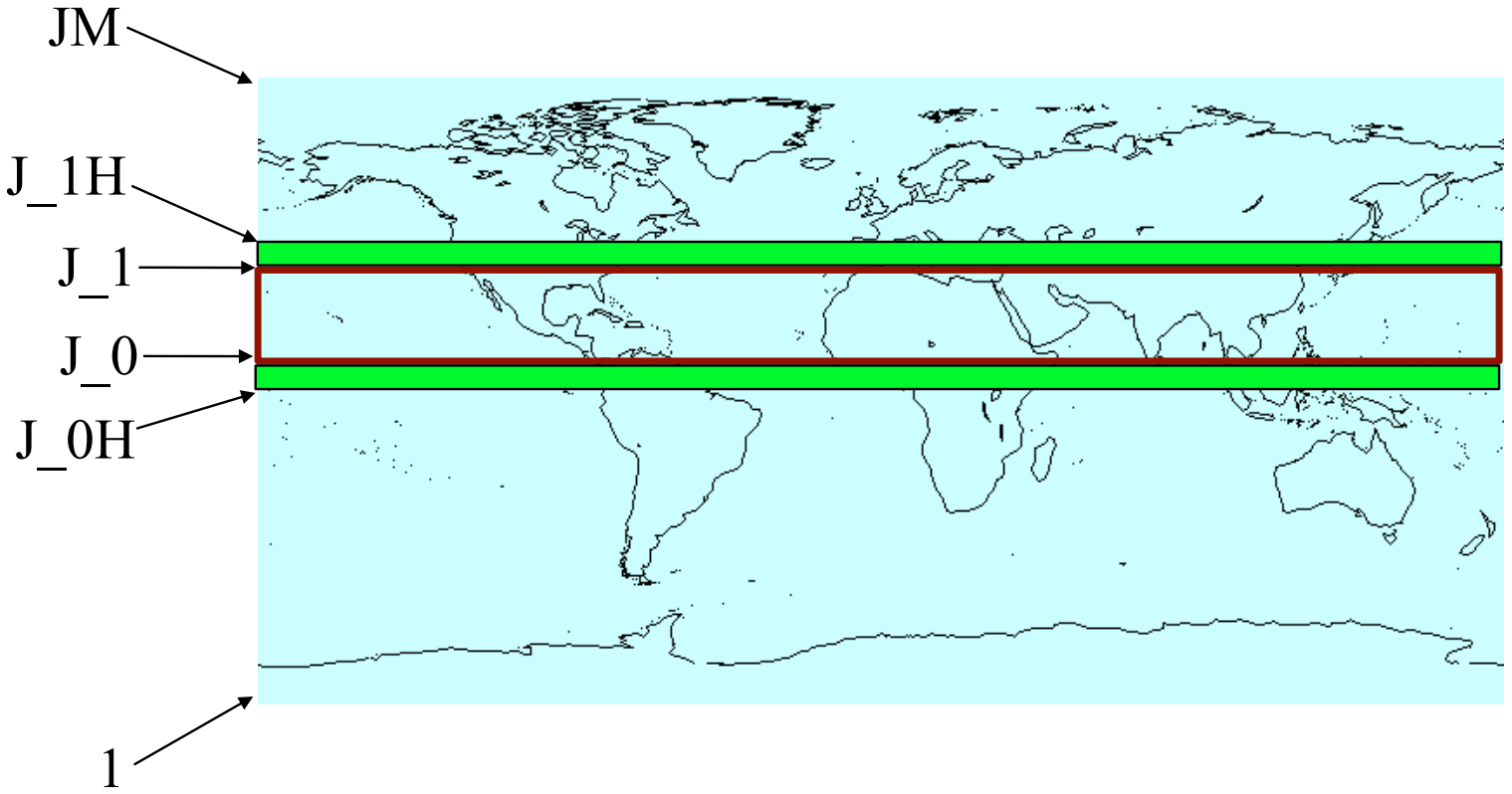
Domain Decomposition

1-Dimensional



Halos (Ghost Cells)

We use 1-cell halos : J_0H = J_0 – 1
J_1H = J_1 + 1

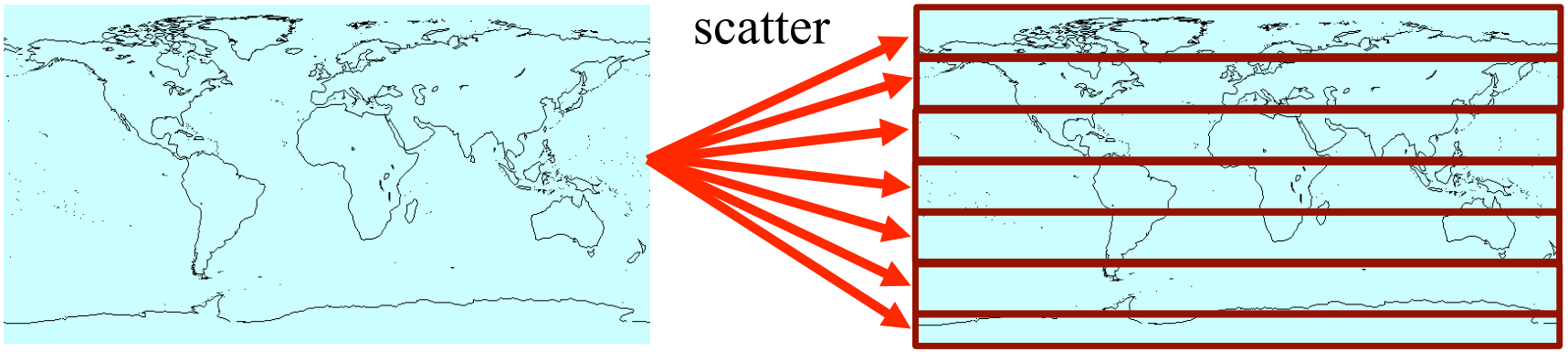


CALL HALO_UPDATE(grid, V, from=NORTH)

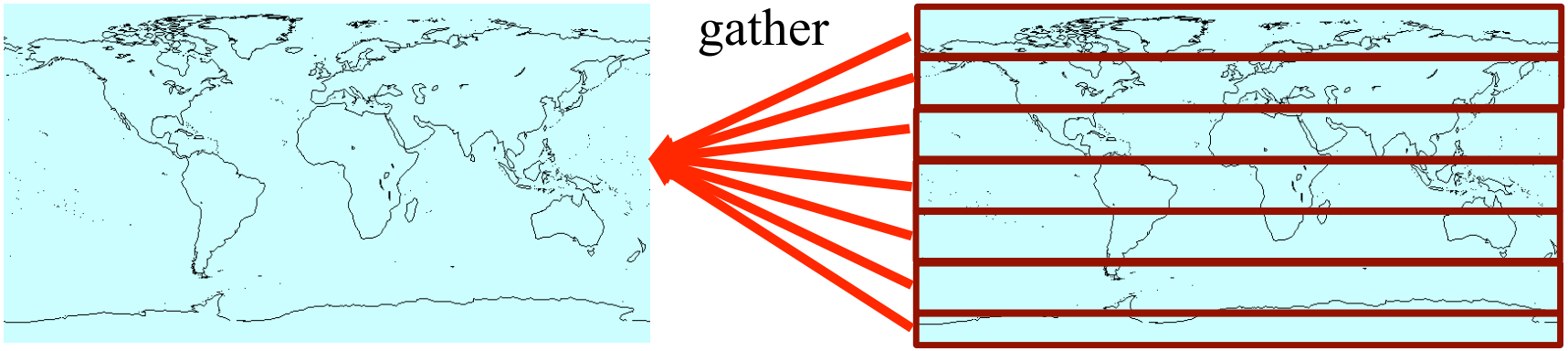
Gather/Scatter

I/O, diagnostics, coupling

computations



CALL UNPACK_DATA(grid, TEARTH_glob, TEARTH)



CALL PACK_DATA(grid, TEARTH, TEARTH_glob)

DOMAIN_DECOMP.f

```
TYPE DIST_GRID       ! - contains all info about domain decomposition
  TYPE (ESMF_Grid) :: ESMF_GRID
  ! Parameters for Global domain
  INTEGER :: IM_WORLD       ! Number of Longitudes
  INTEGER :: JM_WORLD       ! Number of latitudes
  ! Parameters for local domain
  INTEGER :: I_STRT         ! Begin local domain longitude index
  INTEGER :: I_STOP         ! End   local domain longitude index
  INTEGER :: J_STRT         ! Begin local domain latitude  index
  INTEGER :: J_STOP         ! End   local domain latitude  index
  INTEGER :: J_STRT_SKP     ! Begin local domain exclusive of S pole
  INTEGER :: J_STOP_SKP     ! End   local domain exclusive of N pole
  ! Parameters for halo of local domain
  INTEGER :: I_STRT_HALO    ! Begin halo longitude index
  INTEGER :: I_STOP_HALO    ! End   halo longitude index
  INTEGER :: J_STRT_HALO    ! Begin halo latitude  index
  INTEGER :: J_STOP_HALO    ! End   halo latitude  index
  ..........................................................................
END TYPE DIST_GRID
```

DOMAIN_DECOMP.f

Initialization routines:
INIT_APP, INIT_GRID, FINISH_APP, DESTROY_GRID

Scatter routines:
ESMF_BCAST, UNPACK_DATA, UNPACK_DATAj,
UNPACK_COLUMN, UNPACK_BLOCK, UNPACK_J

Gather routines:
PACK_DATA, PACK_DATAj, PACK_COLUMN, PACK_BLOCK,
PACK_J, GLOBALSUM, GLOBALMIN, GLOBALMAX

Halo routines:
HALO_UPDATE, HALO_UPDATE_COLUMN

I/O routines:
DREAD_PARALLEL, MREAD_PARALLEL, READT_PARALLEL,
READ_PARALLEL, WRITE_PARALLEL,WRITEI_PARALLEL

Polling routines:
GET, AM_I_ROOT

Main

```
PROGRAM GISS_MODELE
USE DOMAIN_DECOMP

CALL INIT_APP(GRID, IM, JM, LM)
CALL ALLOC_DRV(GRID)

CALL INIT_GRID(OGRID, IMO, JMO, LMO)
CALL ALLOC_OCEAN(OGRID)

! MAIN CODE
.....

CALL STOP_MODEL("OK", 13)

END
```

Memory Allocation

```
SUBROUTINE ALLOC_GHY_COM(GRID)
USE DOMAIN_DECOMP, ONLY : DIST_GRID, GET
IMPLICIT NONE
TYPE (DIST_GRID), INTENT(IN) :: GRID

INTEGER :: J_1H, J_0H
INTEGER :: IER

CALL GET(GRID, J_STRT_HALO=J_0H, J_STOP_HALO=J_1H)

ALLOCATE(      SNOWE(  IM,J_0H:J_1H),
*              TEARTH(    IM, J_0H:J_1H),
*              WEARTH(    IM, J_0H:J_1H),
*             AIEARTH(    IM, J_0H:J_1H),
*              SNOAGE(3, IM, J_0H:J_1H),
*         STAT=IER)
  .....................................
 END SUBROUTINE ALLOC_GHY_COM
```

“Single Column” code

```
SUBROUTINE A(GRID)
USE DOMAIN_DECOMP, ONLY : DIST_GRID, GET
IMPLICIT NONE
TYPE (DIST_GRID), INTENT(IN) :: GRID

INTEGER :: J_1, J_0

CALL GET(GRID, J_STRT=J_0, J_STOP=J_1)



DO J=J_0,J_1
  DO I=1,IM
    X(I,J) = Y(I,J) + Z(I,J)
   ENDDO
 ENDDO

 ..............................................
 END SUBROUTINE A
```

“Advection” code

```
SUBROUTINE B(GRID)
USE DOMAIN_DECOMP, ONLY : DIST_GRID, GET, NORTH
IMPLICIT NONE
TYPE (DIST_GRID), INTENT(IN) :: GRID

INTEGER :: J_1, J_0

CALL GET(GRID, J_STRT=J_0, J_STOP=J_1)
CALL HALO_UPDATE(GRID, Z, FROM=NORTH)

DO J=J_0,J_1
  DO I=1,IM
    X(I,J) = Y(I,J) + Z(I,J+1)
   ENDDO
 ENDDO

 .....
 END SUBROUTINE B
```

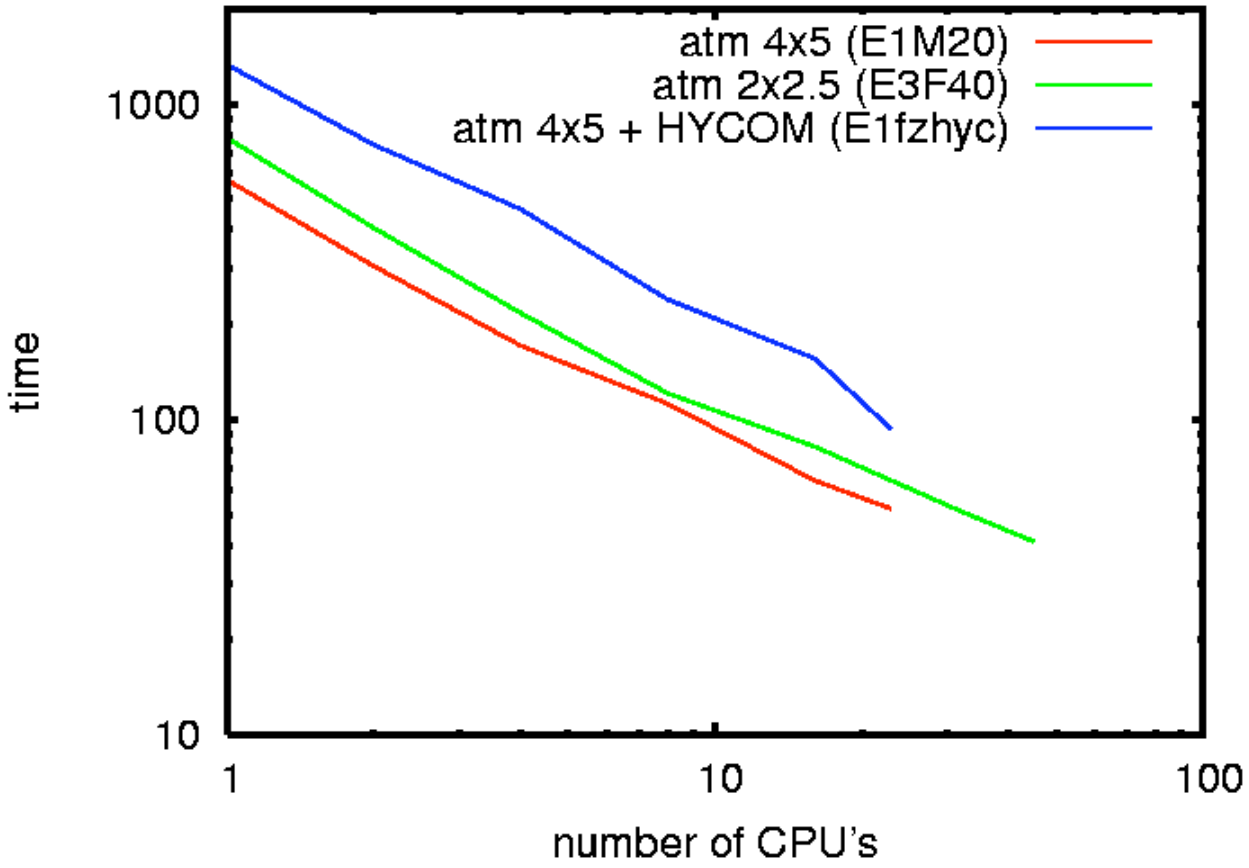
I/O

```
SUBROUTINE READ_SOMETHING(GRID)
USE SOME_MODULE, ONLY : TEARTH
USE DOMAIN_DECOMP, ONLY : DIST_GRID
IMPLICIT NONE
TYPE (DIST_GRID), INTENT(IN) :: GRID
REAL*8 TEARTH_glob(IM,JM)

 IF( AM_I_ROOT() ) THEN
  READ( KUNIT ) TEARTH_glob
 ENDIF

 CALL UNPACK_DATA( GRID, TEARTH_glob, TEARTH )
 ..........................................
 END SUBROUTINE READ_SOMETHING
```

Local arrays and arguments

```
SUBROUTINE C( GRID, X )
USE DOMAIN_DECOMP, ONLY : DIST_GRID, GET
IMPLICIT NONE
TYPE (DIST_GRID), INTENT(IN) :: GRID
REAL*8 X(:, GRID%J_STRT_HALO:)
REAL*8 Y(IM, GRID%J_STRT_HALO:GRID%J_STOP_HALO)

.....
END SUBROUTINE C
```

Current performance of the model



Some statistics (atmospheric part)

Times for a 10 day model run on 15 CPU's:

Global (excluding initialization):	97 s
MPI_SendRecv (i.e. halo updates):	12 s

The number of MPI_SendRecv called: ~23000/day

Conclusions

- OpenMP is ineffective on current architectures should be abandoned
- Current MPI implementation demonstrates decent scaling in all tests
- Though the model may profit from switching to 2D decomposition, there is no evidence that 1D decomposition is currently a bottleneck
- In any case, the model needs cleaner interfaces to domain decomposition library