

# Single-sided PGAS Communications Libraries

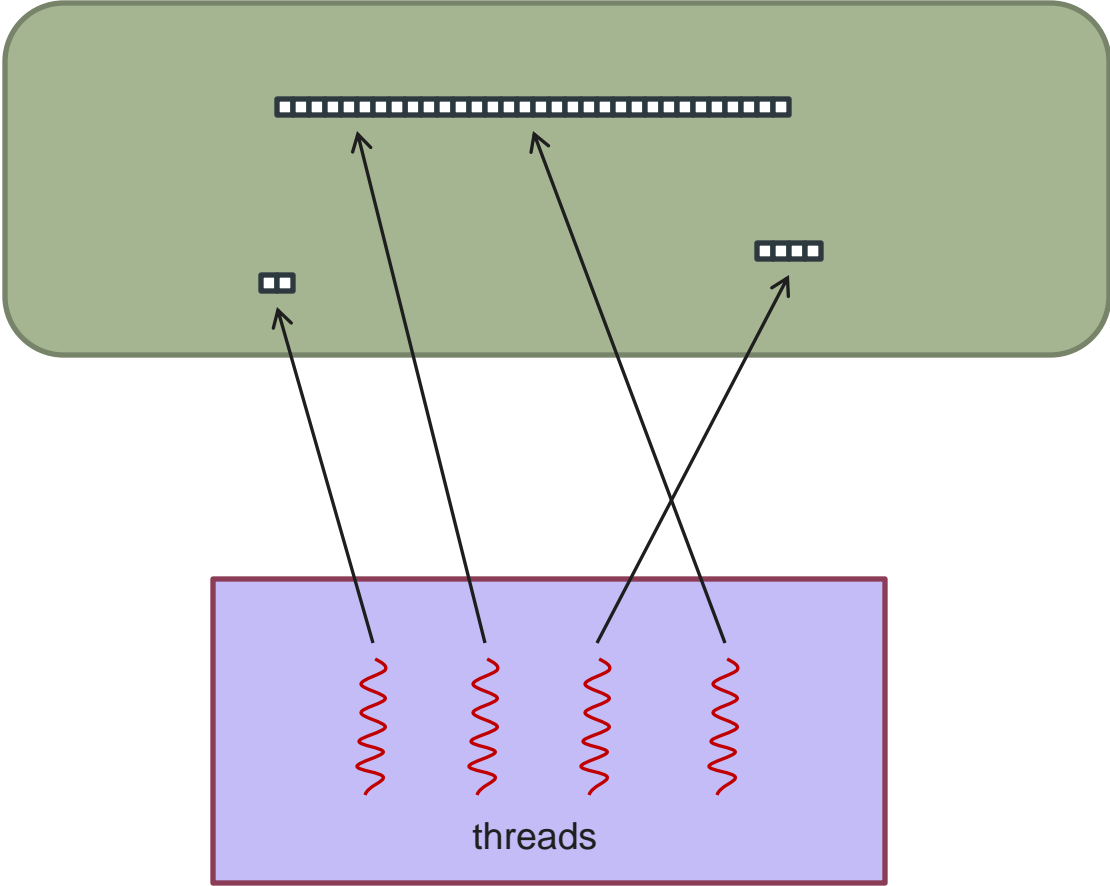
---

Overview of PGAS approaches

David Henty, Alan Simpson (EPCC)  
Harvey Richardson, Bill Long (Cray)

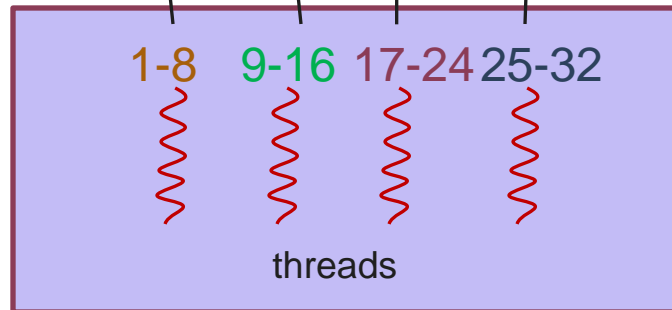
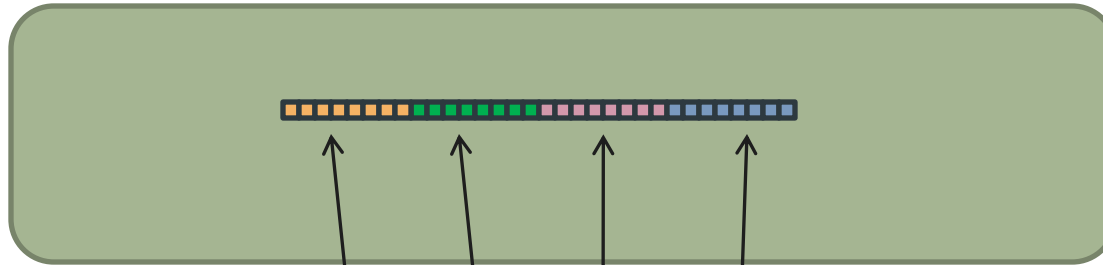
# Shared-memory directives and OpenMP

memory



# OpenMP: work distribution

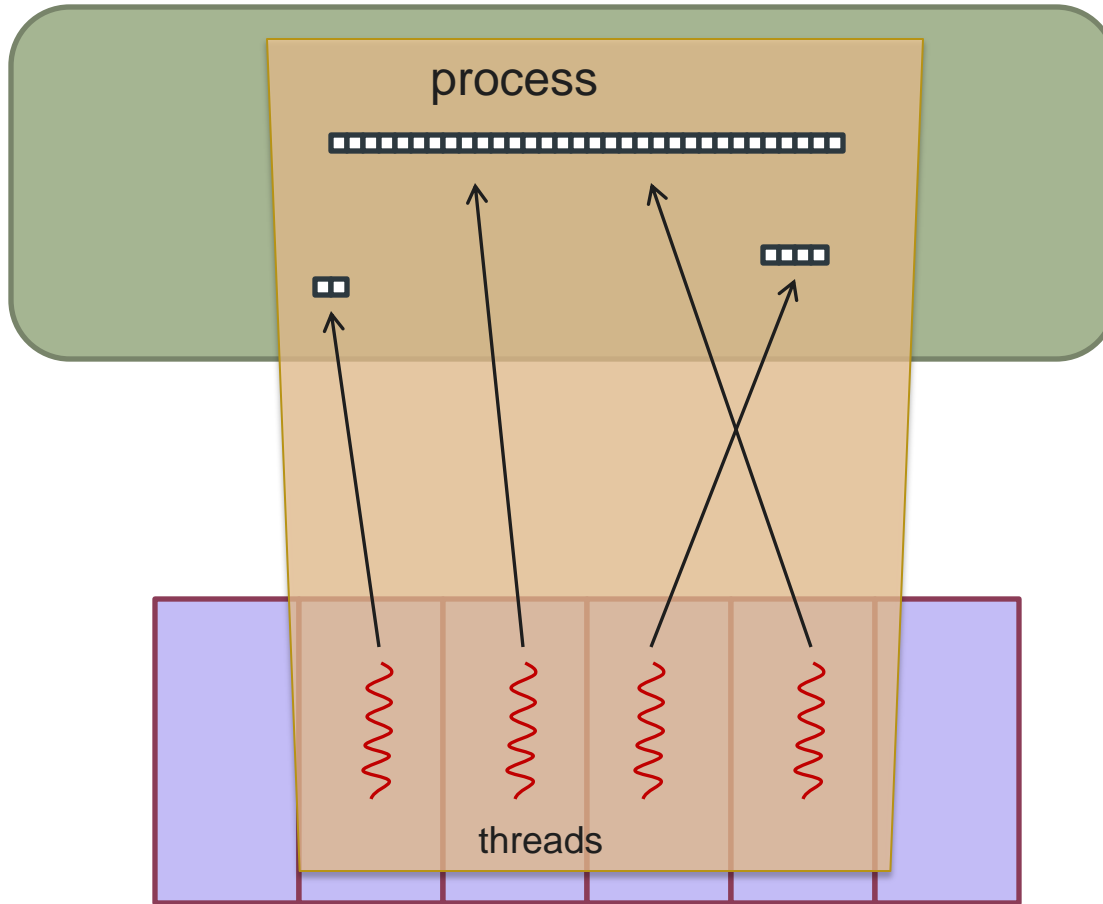
memory



```
!$OMP PARALLEL DO  
do i=1,32  
    a(i)=a(i)*2  
end do
```

# OpenMP implementation

memory

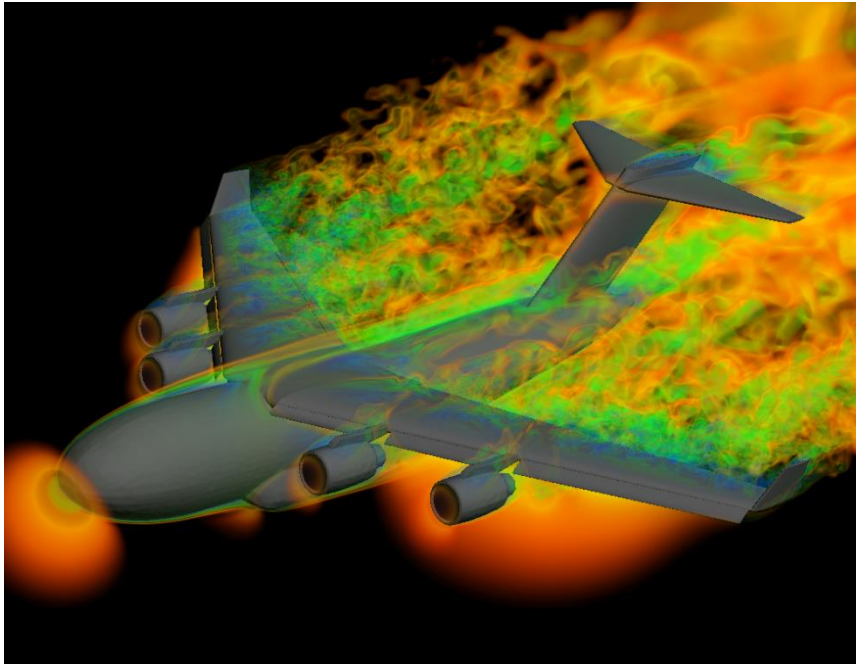


cpus

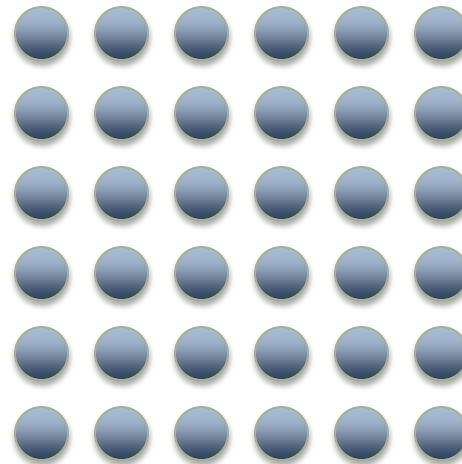
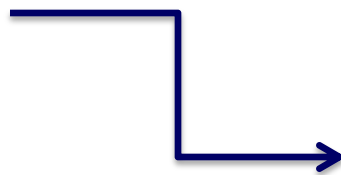
# Shared Memory Directives

- Multiple threads share global memory
- Most common variant: OpenMP
- Program loop iterations distributed to threads, more recent task features
  - Each thread has a means to refer to private objects within a parallel context
- Terminology
  - Thread, thread team
- Implementation
  - Threads map to user threads running on one SMP node
  - Extensions to distributed memory not so successful
- OpenMP is a good model to use within a node

# Cooperating Processes Models



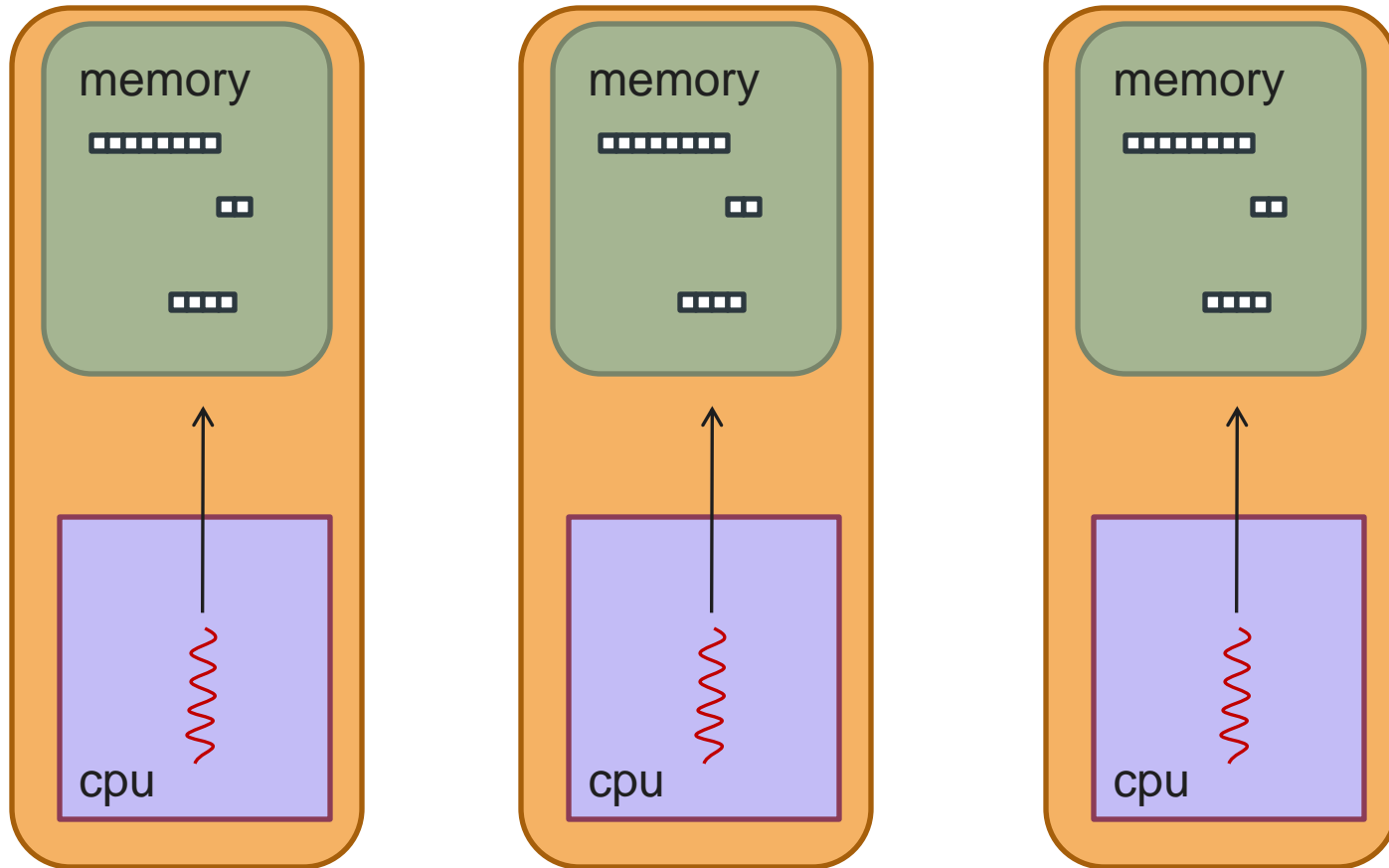
PROBLEM



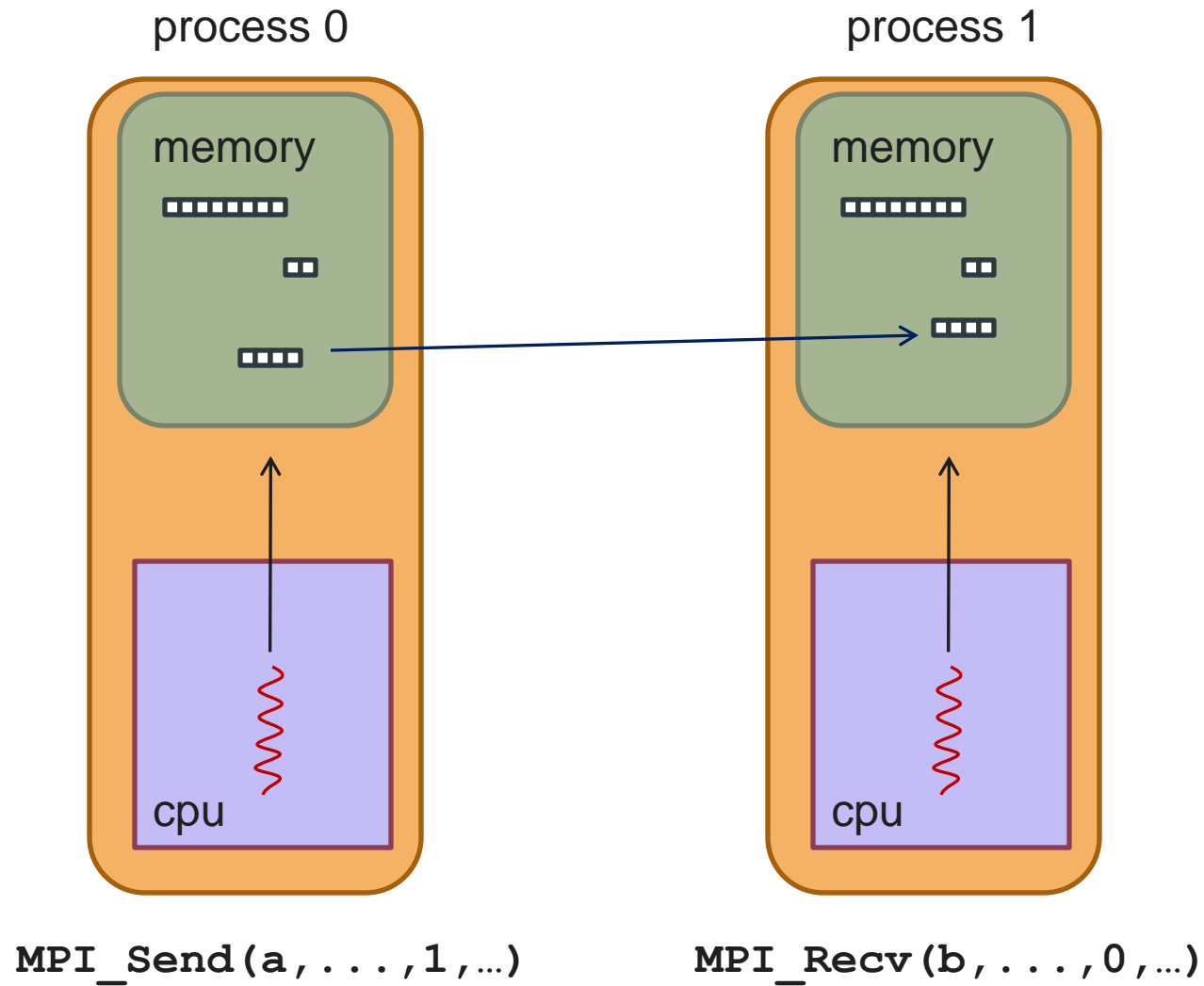
processes

# Message Passing, MPI

process



# MPI

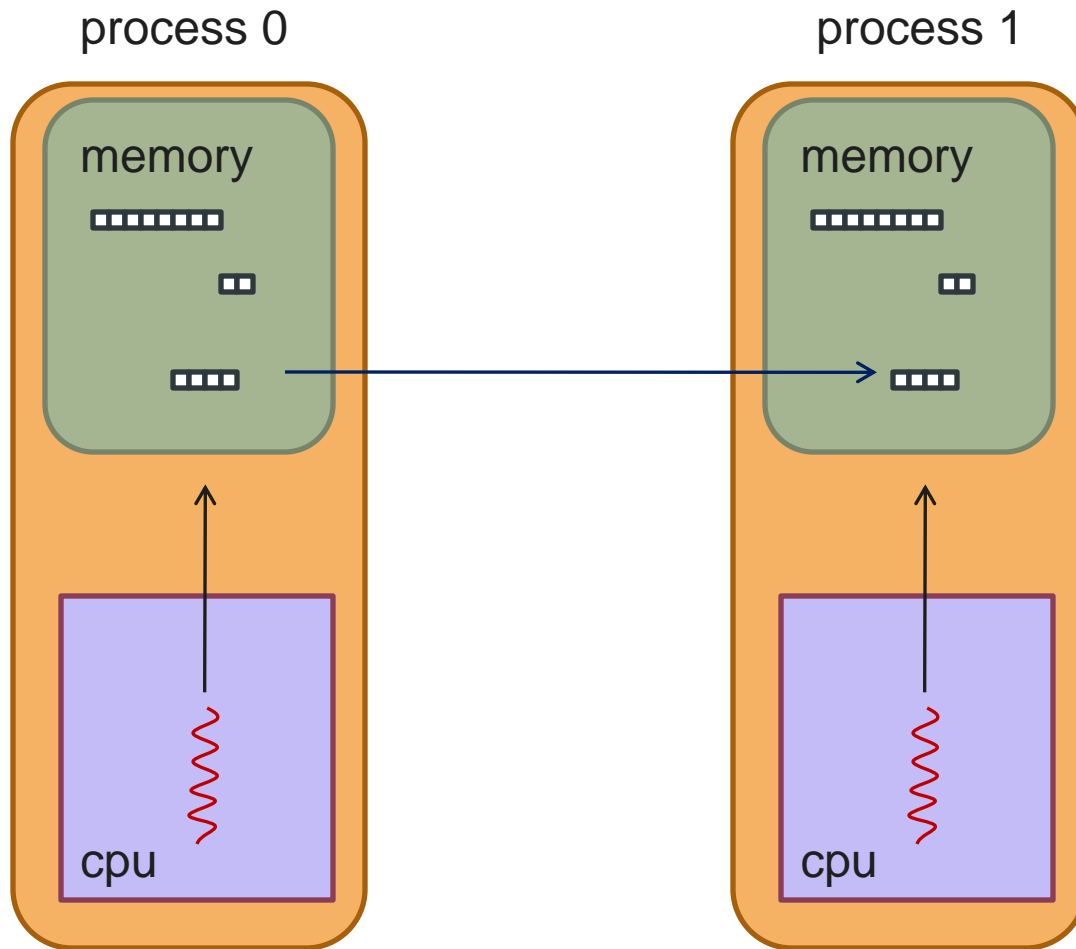




# Message Passing

- Participating processes communicate using a message-passing API
- Remote data can only be communicated (sent or received) via the API
- MPI (the Message Passing Interface) is the standard
- Implementation:  
MPI processes map to processes within one SMP node or across multiple networked nodes
- API provides process numbering, point-to-point and collective messaging operations
- Mostly used in two-sided way, each endpoint coordinates in sending and receiving

# SHMEM



`shmem_put(a, b, 1, ...)`

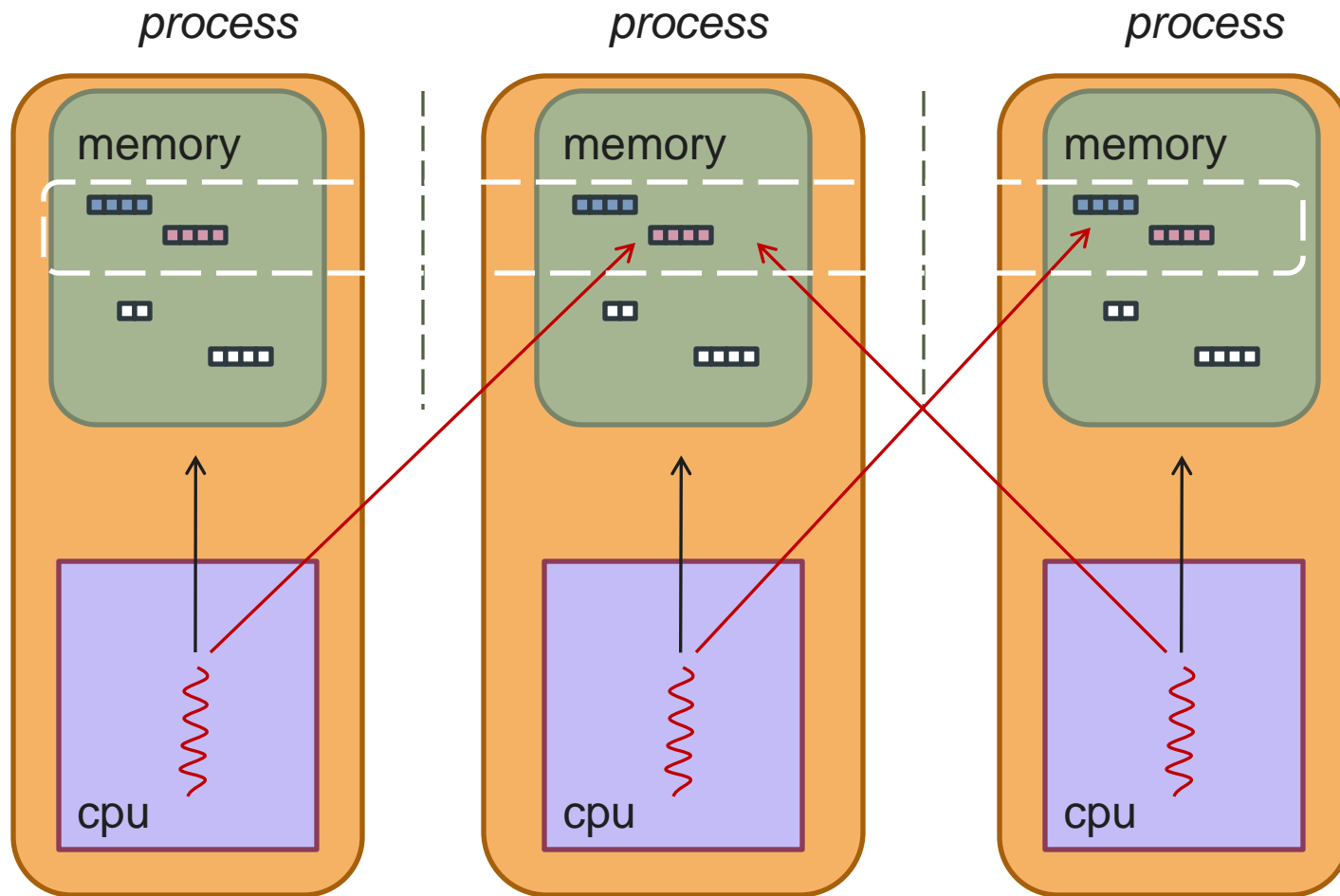
# SHMEM

- Participating processes communicate using an API
- Fundamental operations are based on one-sided PUT and GET
- Need to use symmetric memory locations
- Remote side of communication does not participate
- Can test for completion
- Barriers and collectives
- Popular on Cray and SGI hardware, also Blue Gene version
- To make sense needs hardware support for low-latency RDMA-type operations

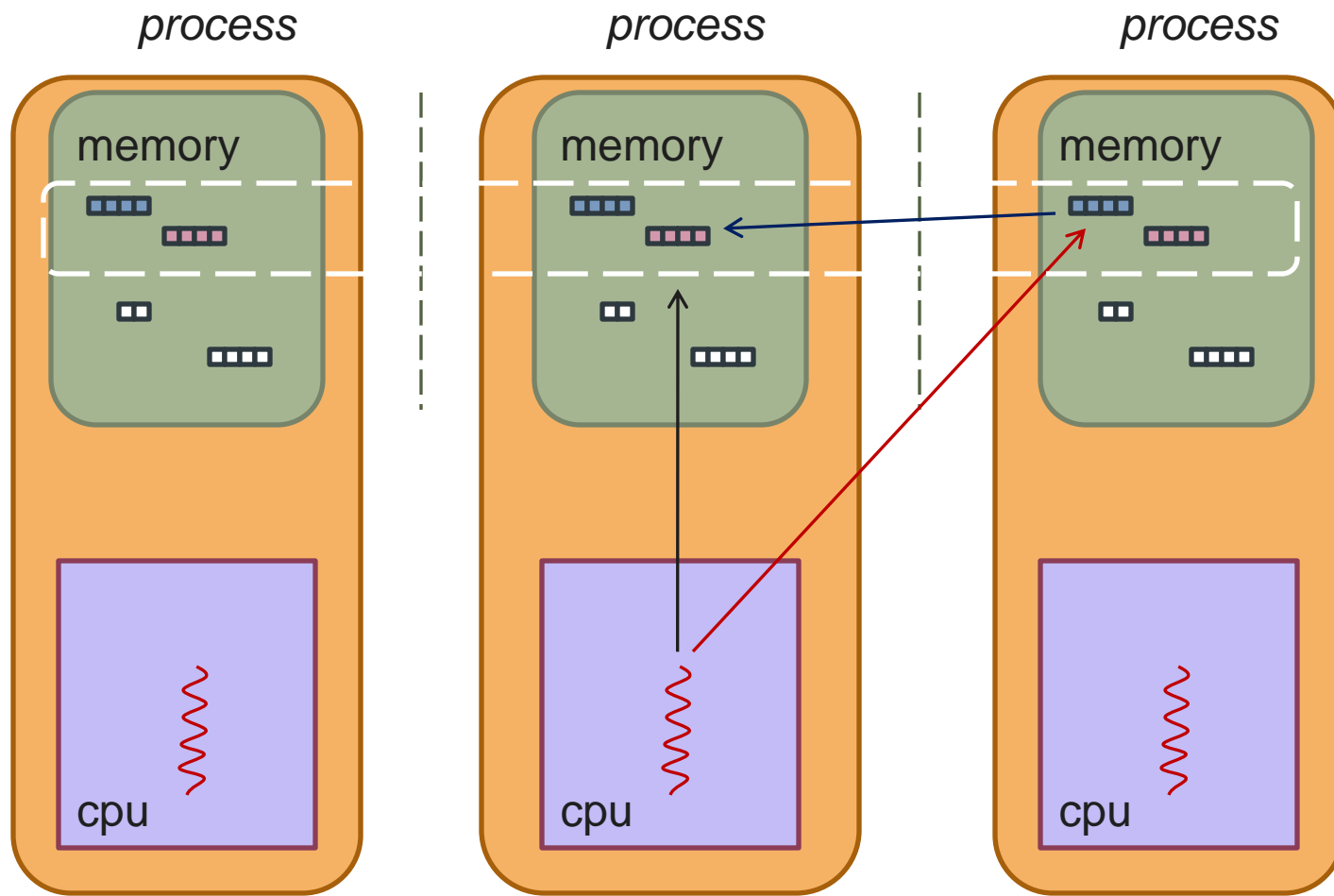
# Fortran 2008 coarray model

- Example of a Partitioned Global Address Space (PGAS) model
- Set of participating processes like MPI
- Participating processes have access to **local memory** via standard program mechanisms
- Access to **remote memory** is directly supported by the language

# Fortran coarray model



# Fortran coarray model



`a = b[3]`

# Fortran coarrays

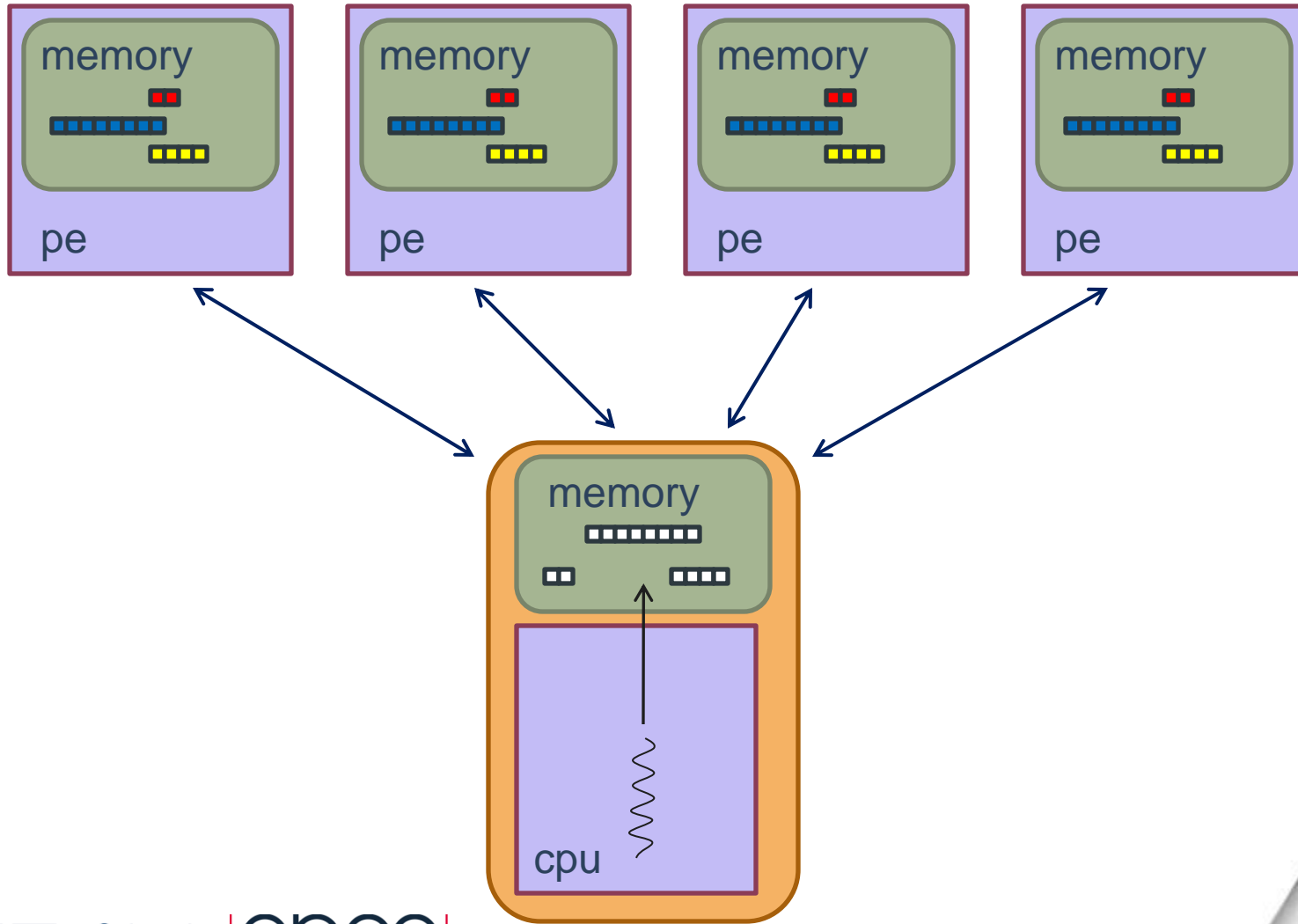
- Remote access is a full feature of the language:
  - Type checking
  - Opportunity to optimize communication
- No penalty for local memory access
- Single-sided programming model more natural for some algorithms
  - and a good match for modern networks with RDMA

# High Performance Fortran (HPF)

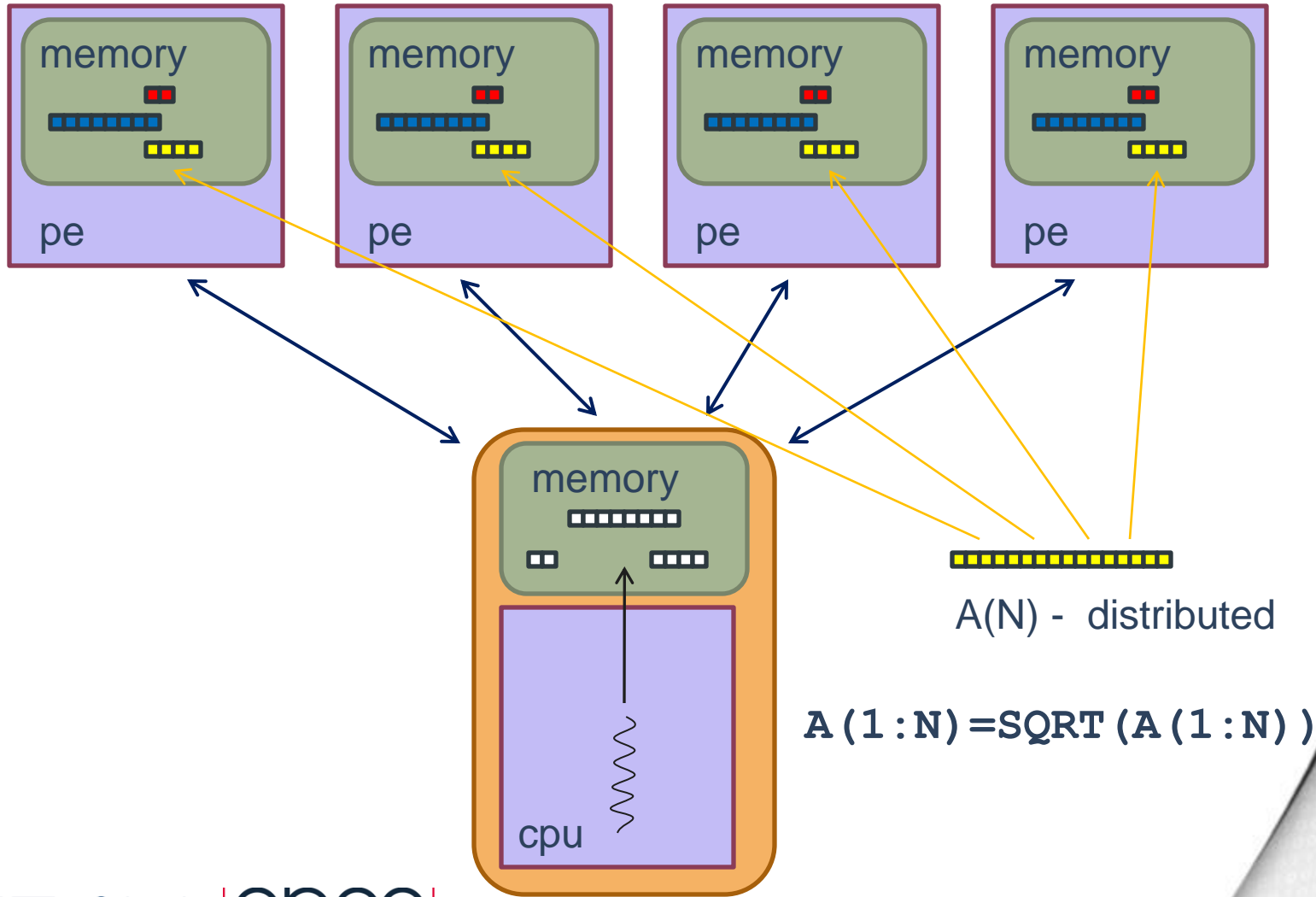
- Data Parallel programming model
- Single thread of control
- Arrays can be distributed and operated on in parallel
- Loosely synchronous
- Parallelism mainly from Fortran 90 array syntax, FORALL and intrinsics.
- This model popular on SIMD hardware (AMT DAP, Connection Machines) but extended to clusters where control thread is replicated



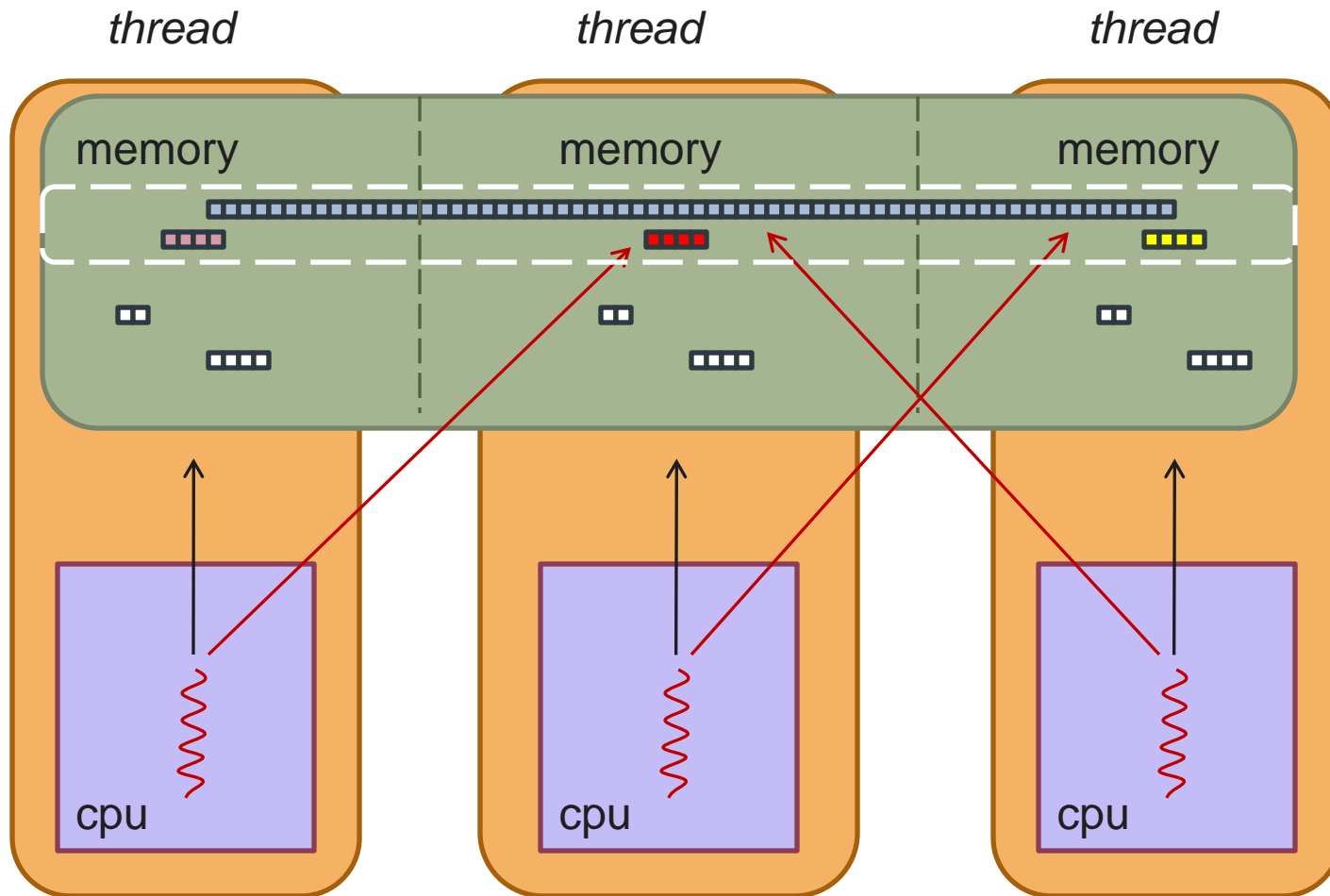
# HPF



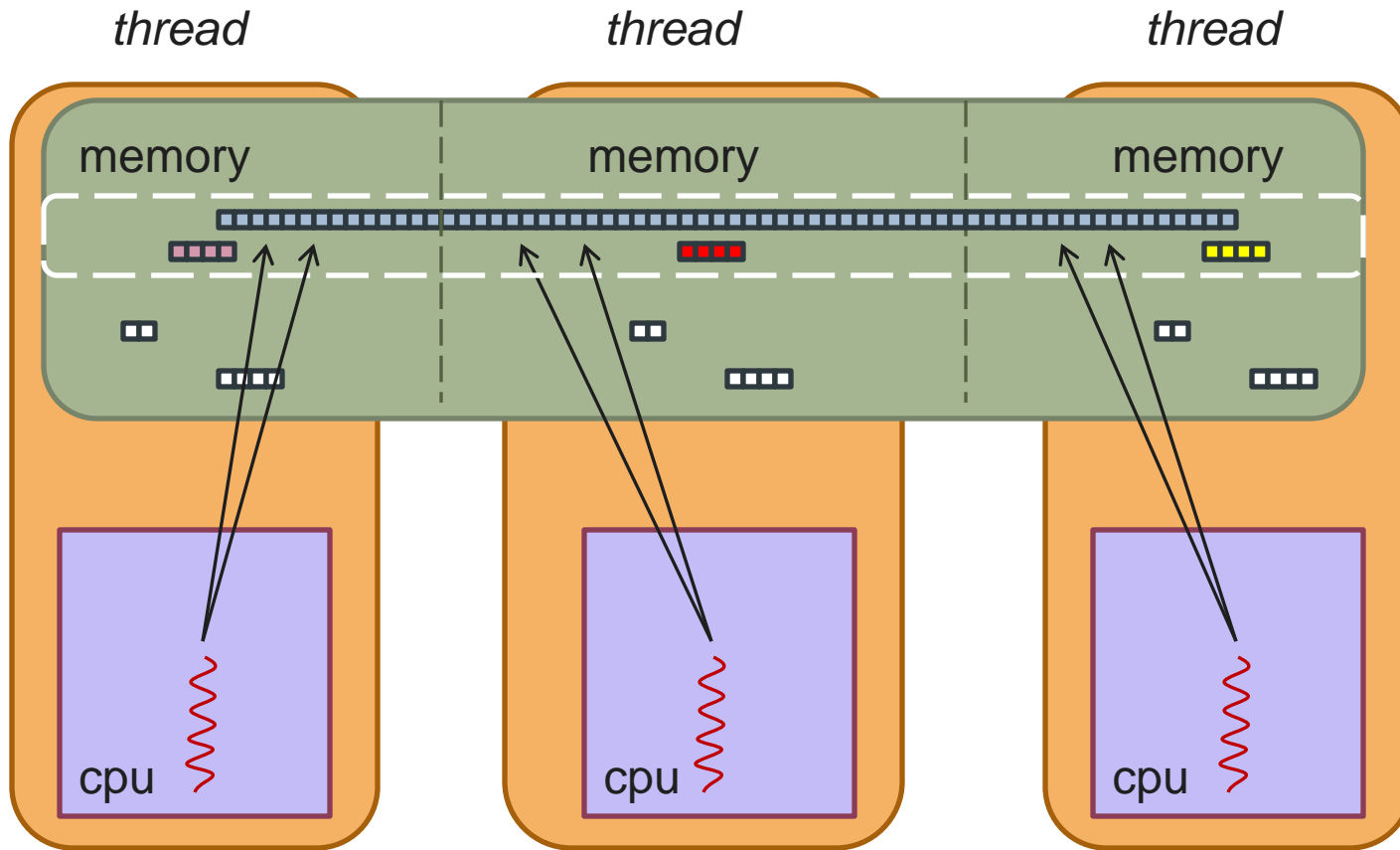
# HPF



# UPC



# UPC



```
upc_forall(i=0;i<32;i++;affinity) {  
    a[i]=a[i]*2;  
}
```

# UPC

- Extension to ISO C99
- Participating “*threads*”
- New *shared* data structures
  - shared pointers to distributed data (block or cyclic)
  - pointers to shared data local to a thread
  - Synchronization
- Language constructs to divide up work on shared data
  - `upc_forall()` to distribute iterations of `for()` loop
- Extensions for collectives
- Both commercial and open source compilers available
  - Cray, HP, IBM
  - Berkeley UPC (from LBL), GCC UPC



**CRAY**  
THE SUPERCOMPUTER COMPANY

**epcc**