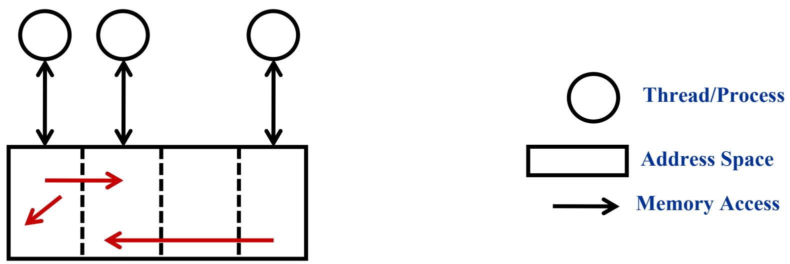
Partitioned Global Address Space (PGAS)

5-Dez-2017

PGAS Basics

- Programming models
  - Message passing
  - Shared memory (Global address space, includes DSM)
  - Partitioned Global Address Space
- Partitioned shared space
  - Global arrays have fragments in multiple partitions
  - A datum may reference data in other partitions
  - Examples: UPC, X10, Chapel, CAF, Titanium



PGAS Basics

- PGAS vs Others

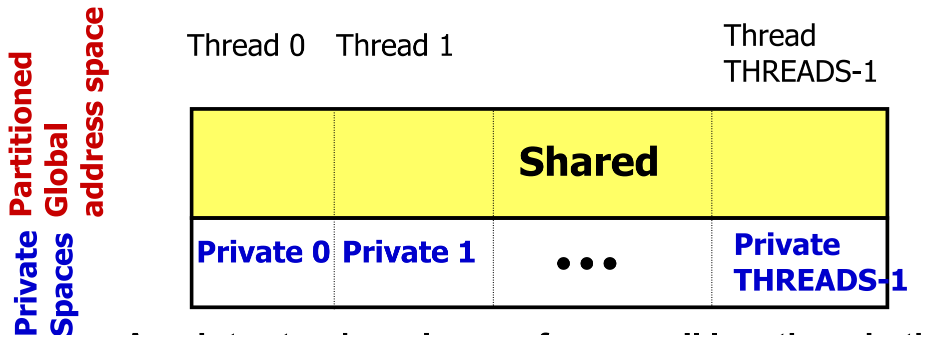
	UPC, X10, Chapel, CAF, Titanium	MPI	OpenMP
Memory model	PGAS (Partitioned Global Address Space)	Distributed Memory	Shared Memory
Notation	Language	Library	Annotations
Global arrays?	Yes	No	No
Global pointers/references?	Yes	No	No
Locality Exploitation	Yes	Yes, necessarily	No

UPC – Unified Parallel C

- A partitioned shared memory parallel programming language
- An explicit parallel extension of ISO C
- Execution model:
  - A number of threads working independently in a SPMD fashion
    - MYTHREAD specifies thread index
  - Synchronization when needed
    - Barriers, Locks, Memory consistency control
- Compiler implementations by vendors and others
- Consortium of government, academia, and HPC vendors including IDA CCS, GWU, UCB, MTU, UMN, ARSC, UMCP, U of Florida, ANL, LBNL, LLNL, DoD, DoE, HP, Cray, IBM, Sun, Intrepid, Etnus,

UPC memory model

- Private and shared memory:

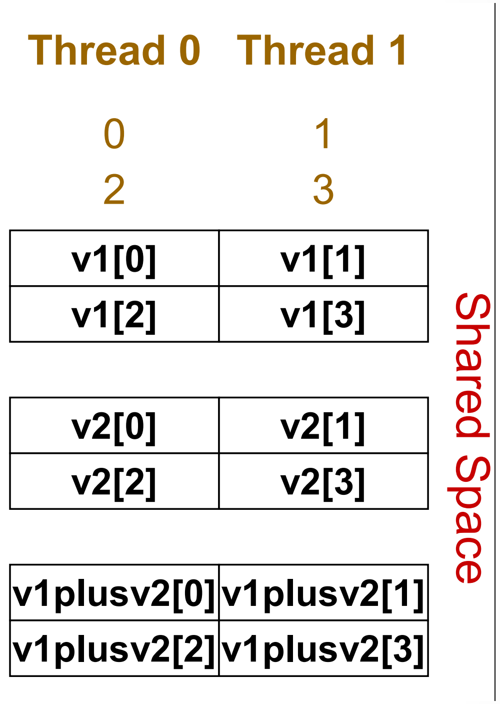


  - A **pointer-to-shared** can reference all locations in the shared space, but there is data-thread affinity
  - A **private pointer** may reference addresses in its private space or its local portion of the shared space
- Static and dynamic memory allocations are supported for both shared and private memory

UPC example: Vector addition

```
shared int v1[N], v2[N], v1plusv2[N];

for(i=MYTHREAD; i<N; i+=THREADS)
    v1plusv2[i]=v1[i]+v2[i];
```



- Implementation with upc_forall

```
upc_forall(i=0; i<N; i++; i)
    v1plusv2[i]=v1[i]+v2[i];
```

Shared and private data

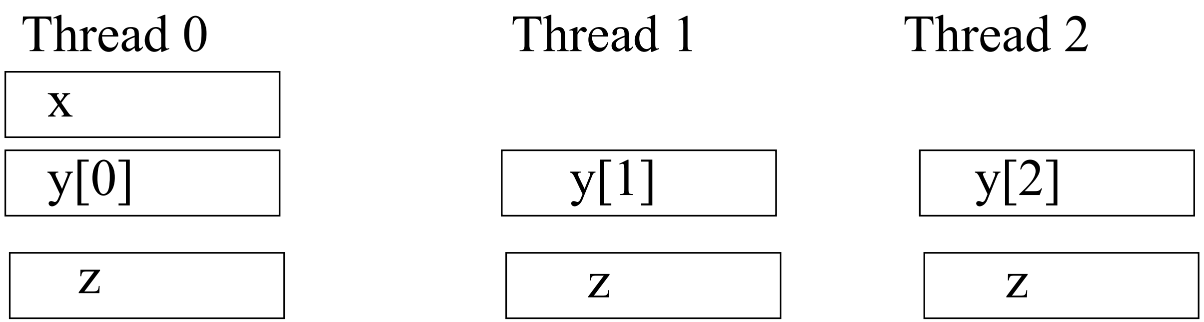
- **Assume THREADS = 3**

  shared int x; /*x will have affinity to thread 0 */

  shared int y[THREADS];

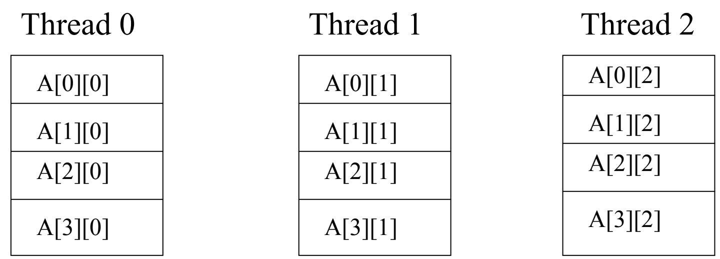
  int z;

- **will result in the layout:**



Shared and private data

shared int A[4][THREADS];



Blocking of Shared Arrays

- Default block size is 1
- Shared arrays can be distributed on a block per thread basis, round robin with arbitrary block sizes.
  - E.g., shared [4] int a[16];
- Element i of a blocked array has affinity to thread:

$$\left\lfloor \frac{i}{blocksize} \right\rfloor \bmod THREADS$$

- Special Operators
  - upc_localsizeof - size of the local portion of a shared object
  - upc_blocksizeof - blocking factor associated with the argument

String functions

- library functions to move data to/from shared memory
- equivalent of memcpy
  - upc_memcpy(dst, src, size) - copy from shared to shared
  - upc_memput(dst, src, size) - copy from private to shared
  - upc_memget(dst, src, size) - copy from shared to private

Worksharing with upc_forall

- Distributes independent iterations across threads
  - typically used to boost locality exploitation in a convenient way
- upc_forall(init; test; loop; affinity)
  - Affinity could be an integer expression or a reference to (address of) a shared object
- Examples:

```
shared int a[100],b[100], c[100];
int i;
upc_forall (i=0; i<100; i++; &a[i])  // equivalent to upc_forall (i=0; i<100; i++; i)
  a[i] = b[i] * c[i];

upc_forall (i=0; i<100; i++; (i*THREADS)/100)   // distribution by chunks
```

Collective Operations

- Global Memory Allocation
  - **called by all threads; all the threads will get the same pointer**
    - **shared void *upc_all_alloc (size_t nblocks, size_t nbytes);**
    - **Equivalent to :**
      shared [nbytes] char[nblocks * nbytes]
- upc_barrier
- upc_all_broadcast
- upc_all_prefix_reduce
- Etc..

Memory consistency models

- Consistency can be *strict* or *relaxed*
  - Under the relaxed consistency model, operations on shared data can be reordered by the compiler / runtime system
  - The strict consistency model enforces sequential ordering of shared operations

- UPC provides a fence construct
  - **upc_fence**

  - UPC implementation ensures that all references to shared data are consistent before the upc_fence is completed