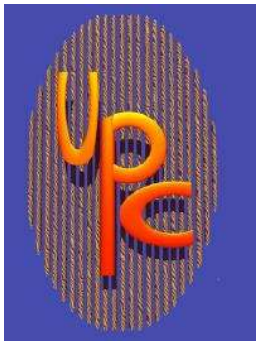
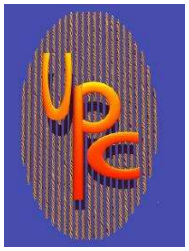


Introduction to Unified Parallel C: A PGAS C

Tarek El-Ghazawi



UPC Overview

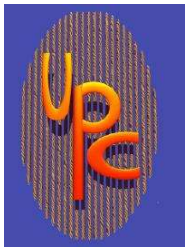
1) UPC in a nutshell
   - ❑ Memory model
   - ❑ Execution model
   - ❑ UPC Systems

2) Data Distribution and Pointers
   - ❑ Shared vs Private Data
   - ❑ Examples of data distribution
   - ❑ UPC pointers

3) Workload Sharing
   - ❑ upc_forall

4) Advanced topics in UPC
   - ❑ Dynamic Memory Allocation
   - ❑ Synchronization in UPC
   - ❑ UPC Libraries

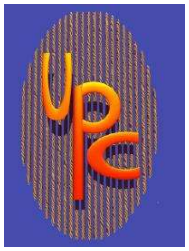
5) UPC Productivity
   - ❑ Code efficiency



Introduction

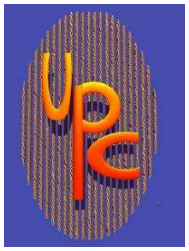
- UPC – Unified Parallel C
- Set of specs for a parallel C
  - v1.0 completed February of 2001
  - v1.1.1 in October of 2003
  - v1.2 in May of 2005
  - v1.3 in November of 2013
- Compiler implementations by vendors and universities
- Consortium of government, academia, and HPC vendors including IDA CCS, GWU, UCB, MTU, U of Florida, UMCP, ANL, LBNL, LLNL, DoD, DoE, HP, Cray, IBM, UMN, ARSC, Sun, Intrepid, Etnus, …

- http://upc.lbl.gov/publications/upc-spec-1.3.pdf



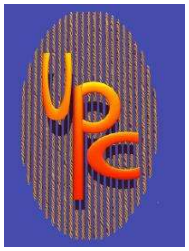
Introduction cont.

- UPC compilers are now available for most HPC platforms and clusters
  - Some are open source
- A debugger and a performance analysis tool are available
- Benchmarks, programming examples, and compiler testing suite(s) are available
- Visit www.upcworld.org or upc.gwu.edu for more information



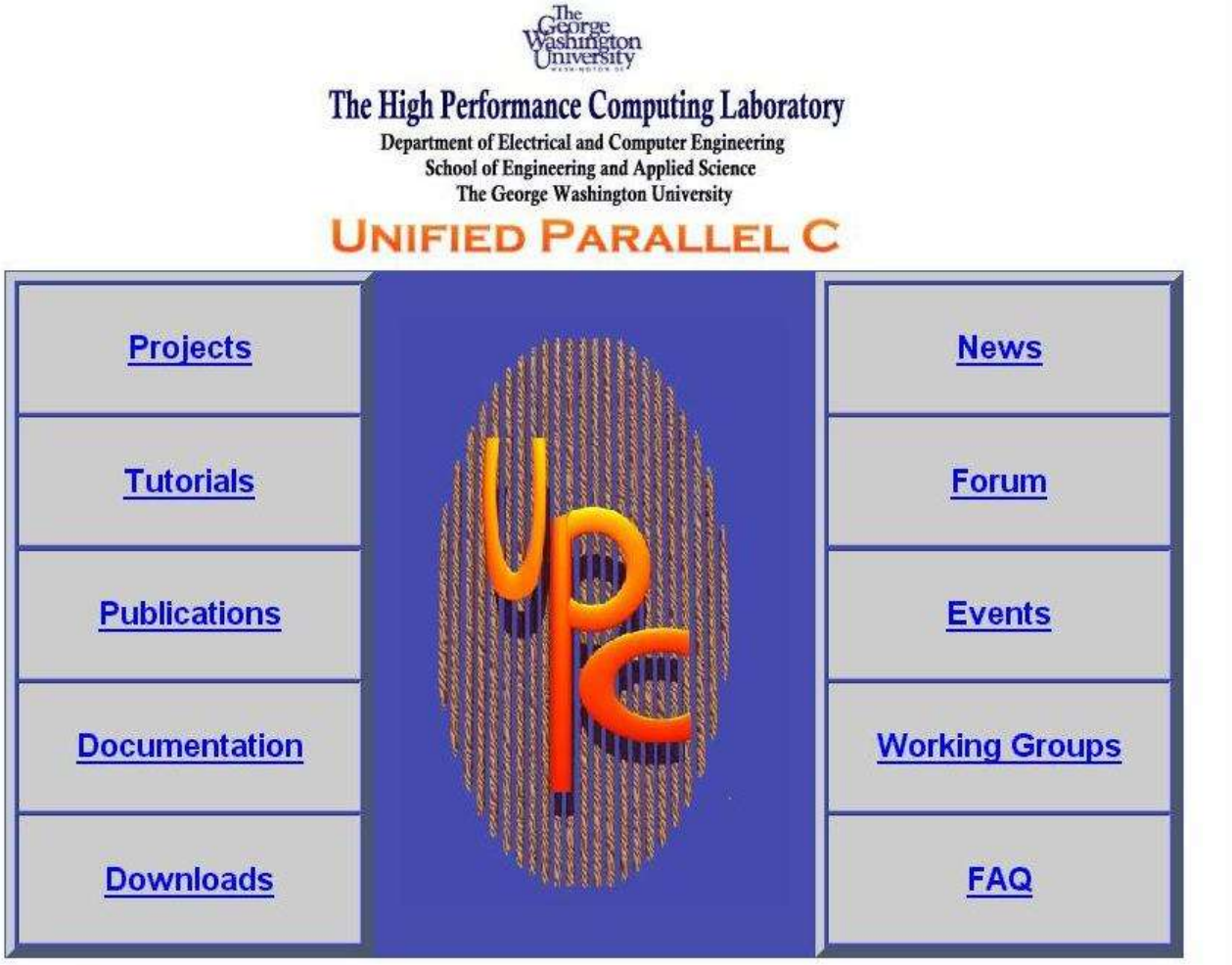
UPC Systems

- **UPC Compilers**
  - Cray
  - Hewlett-Packard
  - Berkeley
  - Intrepid
  - IBM
  - MTU
- **UPC Tools**
  - Totalview
  - PPW from UF
  - TAU

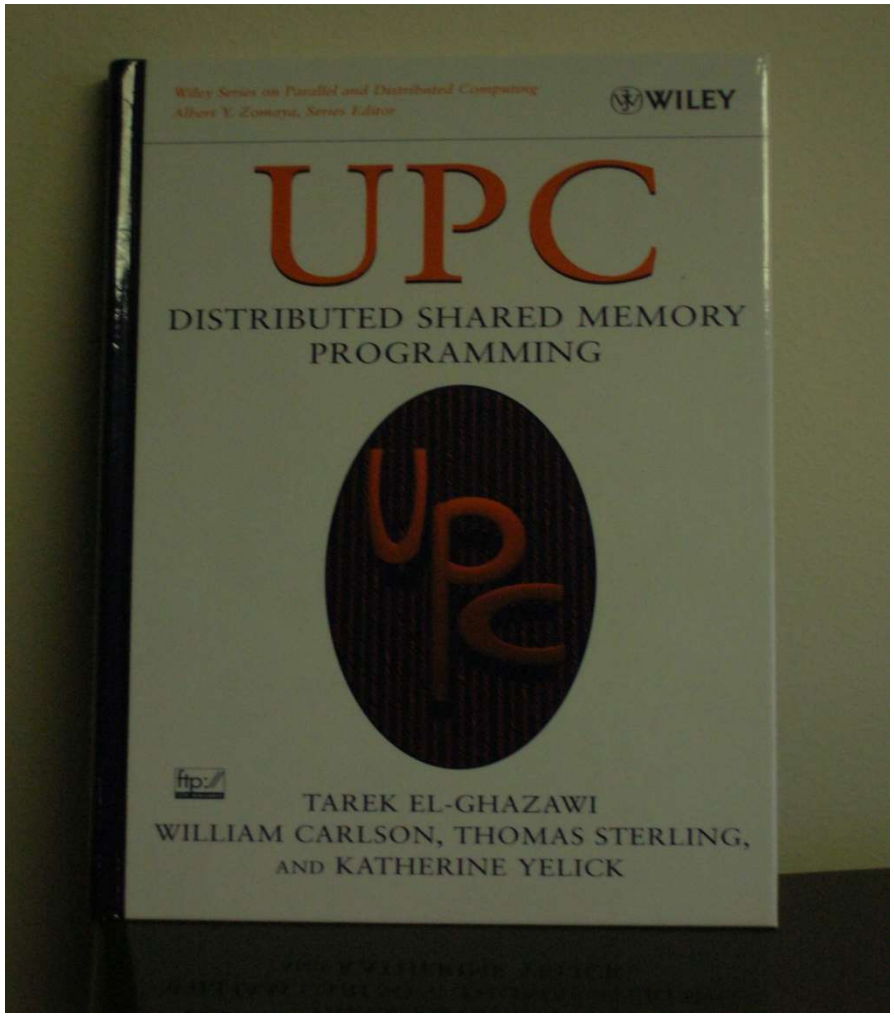


UPC Home Page

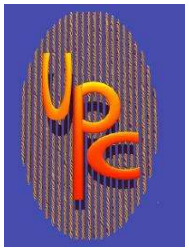
http://upc.gwu.edu



UPC textbook now available

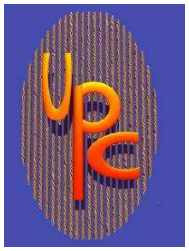


- *UPC: Distributed Shared Memory Programming*
  - Tarek El-Ghazawi
  - William Carlson
  - Thomas Sterling
  - Katherine Yelick
- Wiley, May, 2005
- ISBN: 0-471-22048-5



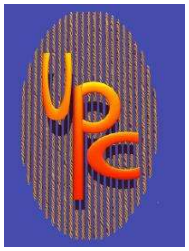
What is UPC?

- Unified Parallel C
- An explicit parallel extension of ISO C
- PGAS parallel programming language

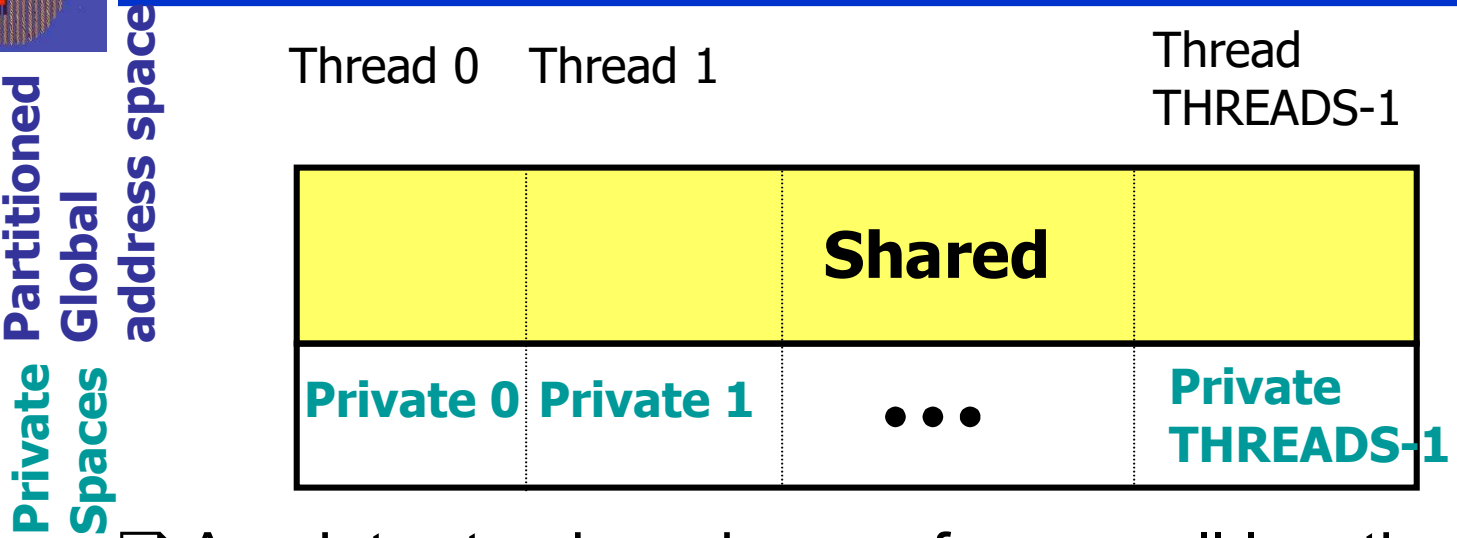


UPC Execution Model

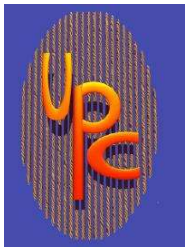
- A number of threads working independently in a SPMD fashion
  - MYTHREAD specifies thread index (0..THREADS-1)
  - Number of threads specified at compile-time or run-time
- Synchronization when needed
  - Barriers
  - Locks
  - Memory consistency control



UPC Memory Model

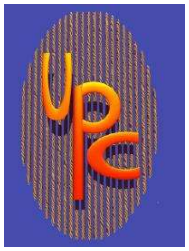


- A pointer-to-shared can reference all locations in the shared space, but there is data-thread affinity
- A private pointer may reference addresses in its private space or its local portion of the shared space
- Static and dynamic memory allocations are supported for both shared and private memory



User's General View

A collection of threads operating in a single global address space, which is logically partitioned among threads. Each thread has affinity with a portion of the shared address space. Each thread has also a private space.



UPC Overview

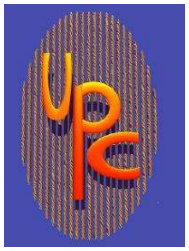
1) UPC in a nutshell
   - ❑ Memory model
   - ❑ Execution model
   - ❑ UPC Systems

2) Data Distribution and Pointers
   - ❑ Shared vs Private Data
   - ❑ Examples of data distribution
   - ❑ UPC pointers

3) Workload Sharing
   - ❑ upc_forall

4) Advanced topics in UPC
   - ❑ Dynamic Memory Allocation
   - ❑ Synchronization in UPC
   - ❑ UPC Libraries

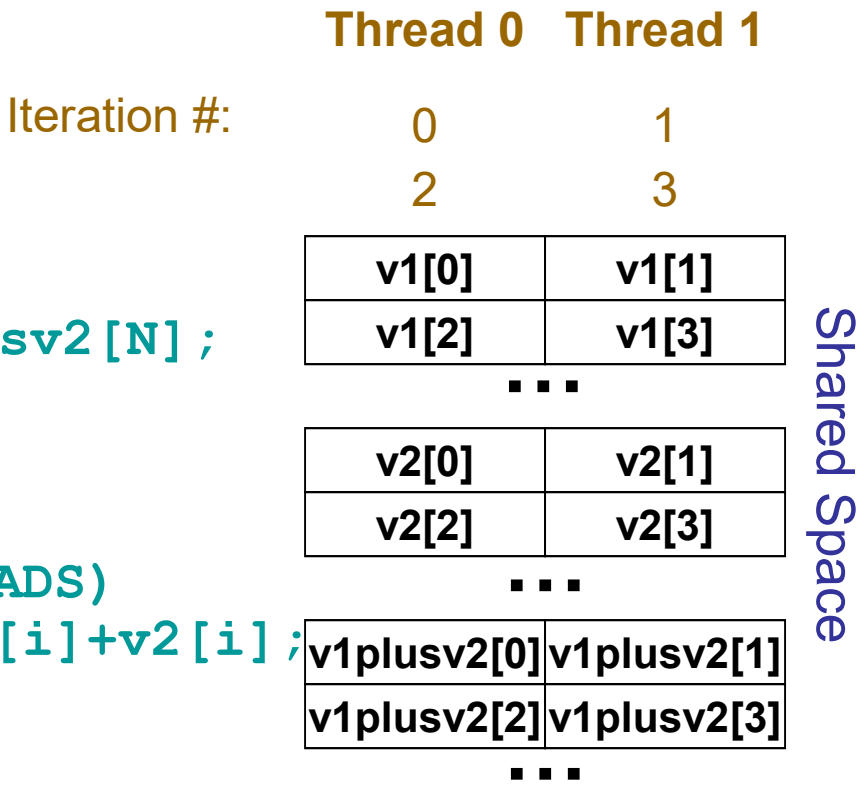
5) UPC Productivity
   - ❑ Code efficiency

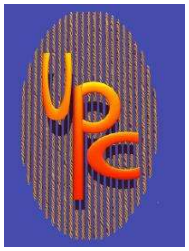


A First Example: Vector addition

```
//vect_add.c
#include <upc_relaxed.h>
#define N 100*THREADS

shared int v1[N], v2[N], v1plusv2[N];
void main() {
    int i;
    for(i=0; i<N; i++)
        if (MYTHREAD==i%THREADS)
            v1plusv2[i]=v1[i]+v2[i];
}
```

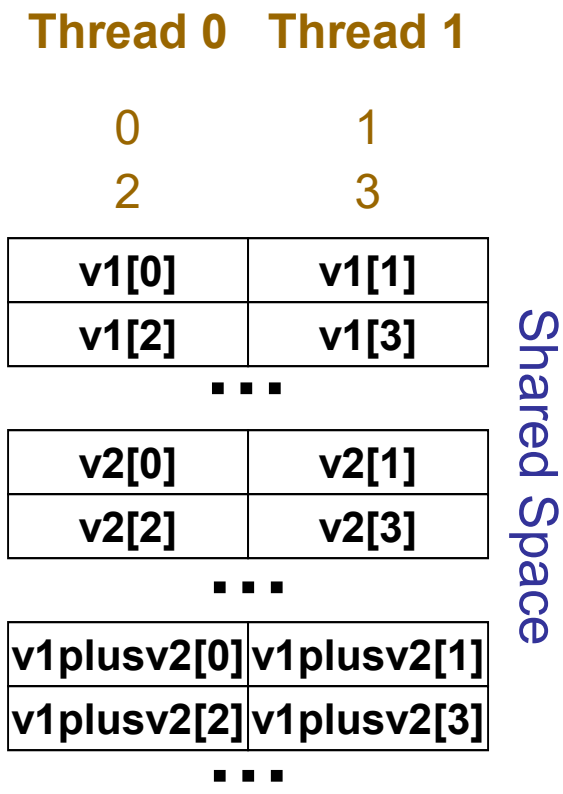


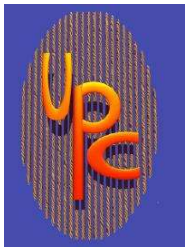


2nd Example: A More Efficient Implementation

```
//vect_add.c
#include <upc_relaxed.h>
#define N 100*THREADS

shared int v1[N], v2[N], v1plusv2[N];
void main() {
    int i;
    for(i=MYTHREAD; i<N; i+=THREADS)
          v1plusv2[i]=v1[i]+v2[i];
}
```



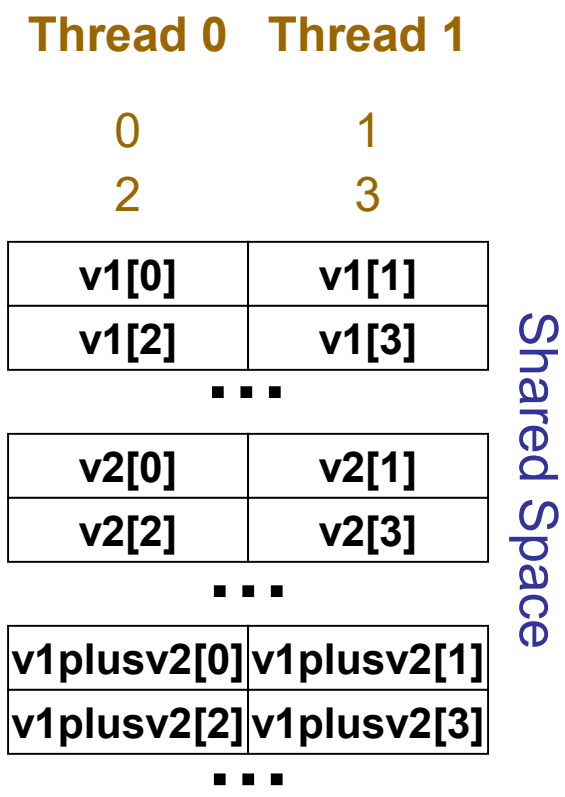


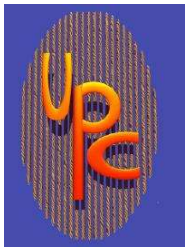
3rd Example: A More Convenient Implementation with upc_forall

```
//vect_add.c
#include <upc_relaxed.h>
#define N 100*THREADS

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

void main()
{
    int i;
    upc_forall(i=0; i<N; i++; i)
        v1plusv2[i]=v1[i]+v2[i];
}
```

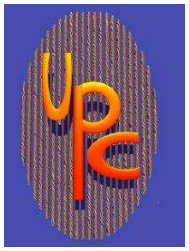




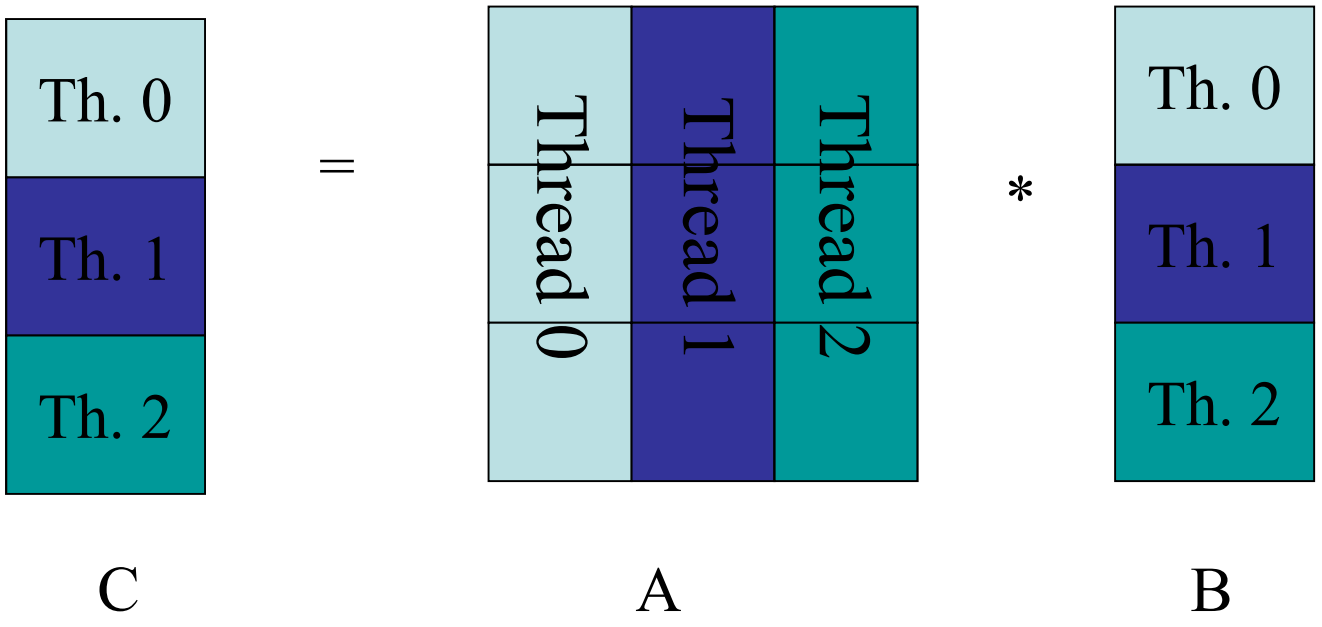
Example: UPC Matrix-Vector Multiplication- Default Distribution

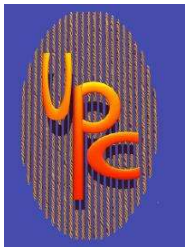
```
// vect_mat_mult.c
#include <upc_relaxed.h>

shared int a[THREADS][THREADS] ;
shared int b[THREADS], c[THREADS] ;
void main (void)
{
      int i, j;
      upc_forall( i = 0 ; i < THREADS ; i++; i){
            c[i] = 0;
            for ( j= 0 ; j < THREADS ; j++)
                  c[i] += a[i][j]*b[j];
      }
}
```

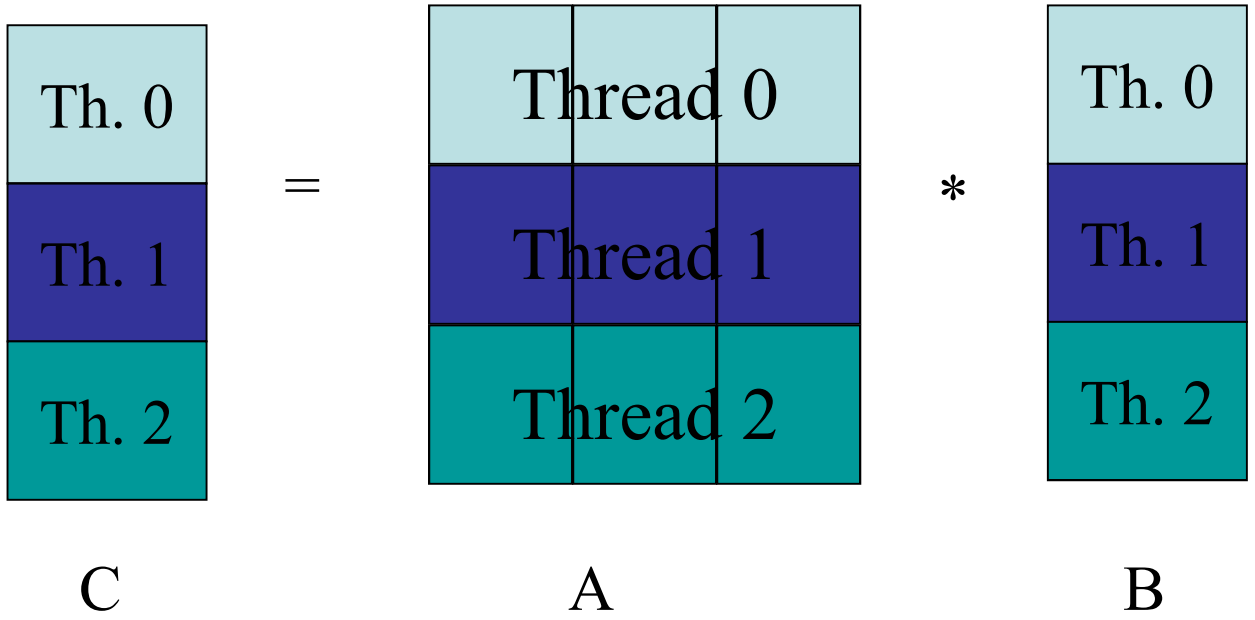


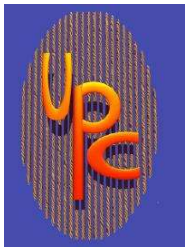
Data Distribution





A Better Data Distribution





Example: UPC Matrix-Vector Multiplication- The Better Distribution

```
// vect_mat_mult.c
#include <upc_relaxed.h>

shared [THREADS] int a[THREADS][THREADS];
shared int b[THREADS], c[THREADS];

void main (void)
{
      int i, j;
      upc_forall( i = 0 ; i < THREADS ; i++; i){
            c[i] = 0;
            for ( j= 0 ; j< THREADS ; j++)
                  c[i] += a[i][j]*b[j];
      }
}
```