

GA Hands-on

ACTS Workshop 2010

[Manoj Kumar Krishnan](#), Bruce Palmer,
Sriram Krishnamoorthy, Abhinav Vishnu,
Daniel Chavarria, Jeff Daily, Patrick Nichols

GA Hands-on

- Download Hands-on guide from here:
 - http://acts.nersc.gov/ga/hands-on/GA_hands_on.pdf
- Copy GA tutorial source codes from here:
 - `cp /usr/common/acts/GA/ga-tutorial.tgz ./`
 - `tar zxf ga-tutorial.tgz`
 - `cd ga-tutorial/`
- **module load ga/4.3.2i8**
- Compile: For example, **make transp1D.x**
- Modify the included job script file (tutorial.job) and submit the job as follows:

Tutorial Programs

- **TUTORIAL Programs: C and Fortran options available**
 - The tutorial source codes are as follows:
 - transp1D.tutorial.F transp1D-c.tutorial.c
 - matrix.tutorial.F matrix-c.tutorial.c
 - NOTE: For above tutorial codes, complete working codes are also available for reference
 - transp1D.F transp1D-c.c
 - matrix.F matrix-c
- Tutorial programs are incomplete. All you have to do is search file for comments marked with ###, and using the text as hints, replace the comments with subroutines or functions from the GA library to create a working code
- Pick any problem as explained on the hands-on guide, and complete the tutorial programs

Useful GA Functions (Fortran)

```
subroutine ga_initialize()  
subroutine ga_terminate()
```

```
integer function ga_nnodes()  
integer function ga_nodeid()
```

```
logical function nga_create(type,dim,dims,name,chunk,g_a)
```

```
integer type (MT_F_INT, MT_F_DBL, etc.)
```

```
integer dim
```

```
integer dims(dim)
```

```
character*(*) name
```

```
integer chunk(dim)
```

```
integer g_a
```

```
logical function ga_duplicate(g_a,g_b,name)
```

```
integer g_a
```

```
integer g_b
```

```
character*(*) name
```

```
logical function ga_destroy(g_a)
```

```
integer g_a
```

```
subroutine ga_sync()
```

Use GA Functions (Fortran)

```
subroutine nga_distribution(g_a, node_id, lo, hi)
  integer g_a
  integer node_id
  integer lo(dim)
  integer hi(dim)
subroutine nga_put(g_a, lo, hi, buf, ld)
  integer g_a
  integer lo(dim)
  integer hi(dim)
  fortran array buf
  integer ld(dim-1)
subroutine nga_get(g_a, lo, hi, buf, ld)
  integer g_a
  integer lo(dim)
  integer hi(dim)
  fortran array buf
  integer ld(dim-1)
```

Useful GA Functions (C)

```
void GA_Initialize()  
void GA_Terminate()
```

```
int GA_Nnodes()  
int GA_Nodeid()
```

```
int NGA_Create(type, dim, dims, name, chunk) Returns GA handle g_a  
    int type (C_INT, C_DBL, etc.)  
    int dim  
    int dims[dim]  
    char* name  
    int chunk[dim]
```

```
int GA_Duplicate(g_a, name) Returns GA handle g_b  
    int g_a  
    char* name
```

```
void GA_Destroy(g_a)  
    int g_a
```

```
void GA_Sync()
```

Useful GA Functions (C)

```
void NGA_Distribution(g_a, node_id, lo, hi)
    int g_a
    int node_id
    int lo[dim]
    int hi[dim]
void NGA_Put(g_a, lo, hi, buf, ld)
    int g_a
    int lo[dim]
    int hi[dim]
    void* buf
    int ld[dim-1]
void NGA_Get(g_a, lo, hi, buf, ld)
    int g_a
    int lo[dim]
    int hi[dim]
    void* buf
    int ld[dim-1]
```

Problem 1 (1D Transpose)*

- Transpose a distributed 1D vector containing N elements in the order $1, 2, \dots, N$ into a distributed vector containing N elements in the order $N, N-1, \dots, 2, 1$
- Fortran version of this problem is in the file `transp1D.F.tutorial`
- C version is in `transp1D.c.tutorial`
- Working versions of these codes are in

* From Global Arrays hands-on guide

Problem 2 (Matrix

- A simple matrix multiply algorithm that initializes two large matrices as GAs. It then multiplies a block of columns by a block or rows from the GAs locally on each processor and copies the result into a third global array
- Fortran version of this problem is in the file `matrix.F.tutorial`

* From Global Arrays hands-on guide

Problem 3*

- Both the codes in problems 1 & 2 initialize the data by initializing a local array on processor 0 with all the data and then copying it to a distributed global array. For real problems it is usually undesirable to have all the data located on one processor at any point in the calculation. Can you modify these codes (problem 1 and 2) so that each processor only initializes the data owned by that processor?
- **1D transpose (Problem 1)**
 - Modify code so that each processor only initializes the local array a() with the data owned by that processor and then copy that data to the global array g_a
 - Hint: Use nga_distribution and nga_put
 - You will also need to modify the result checking part of the code as well so that it also only uses smaller portions of the total GA
 - Hint: copy locally held part of result GA into local array b and corresponding part of original vector into local array a and compare (use arrays lo, hi, lo2, hi2 to get this data).
- **Matrix Multiply (Problem 2)**
 - Modify code so that each processor only initializes the local arrays a and b with the data held locally by that processor. Then copy that data to the global arrays g_a and g_b.
 - Hint: Use nga_distribution and nga_put

* From Global Arrays hands-on guide