



Zoltan Tutorial: Hands-On Session

ACTS Workshop
August 20, 2010

Michael Wolf



Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



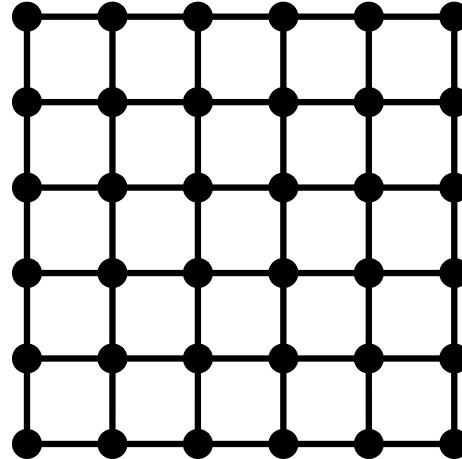


Objectives

- Learn how to partition a problem using Zoltan
- Understand the following
 - Basic process of partitioning with Zoltan
 - Setting Zoltan parameters
 - Registering query functions
 - **Writing query functions**
 - Zoltan_LB_Partition and its input/output
- Be able to integrate Zoltan into your own applications



Exercises



- Partition simple mesh structure into 4 parts
- Choice of three exercises
 - Exercise 1: Least difficult
 - Exercise 2: More difficult
 - Exercise 3: Most difficult



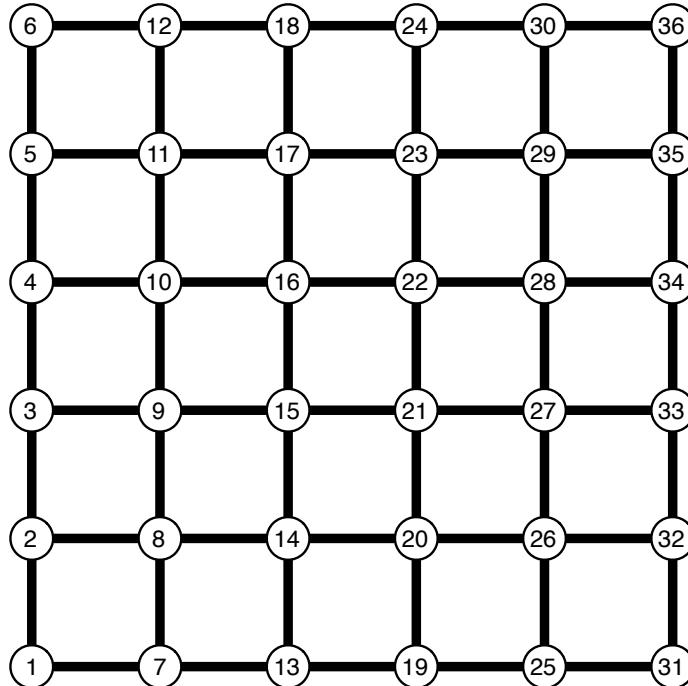
Exercises 1, 2: RCB Partitioning

(6) (0,5)	(12) (1,5)	(18) (2,5)	(24) (3,5)	(30) (4,5)	(36) (5,5)
(5) (0,4)	(11) (1,4)	(17) (2,4)	(23) (3,4)	(29) (4,4)	(35) (5,4)
(4) (0,3)	(10) (1,3)	(16) (2,3)	(22) (3,3)	(28) (4,3)	(34) (5,3)
(3) (0,2)	(9) (1,2)	(15) (2,2)	(21) (3,2)	(27) (4,2)	(33) (5,2)
(2) (0,1)	(8) (1,1)	(14) (2,1)	(20) (3,1)	(26) (4,1)	(32) (5,1)
(1) (0,0)	(7) (1,0)	(13) (2,0)	(19) (3,0)	(25) (4,0)	(31) (5,0)

- **RCB geometric partitioning**
- **Given x,y coordinates for each object**
 - Data stored in **MESH_DATA** struct
- **No connectivity information**



Exercise 3: Graph Partitioning



- **Connectivity information given**
- **Graph data stored in GRAPH_DATA struct**
 - List of global IDs on process (vertexGID)
 - Stores neighbors of local vertices (nborIndex, nborGID)
 - Processes owning neighbors (nborProc)



Completing Exercises

- Refer to Zoltan User's Guide
 - http://www.cs.sandia.gov/Zoltan/ug_html/ug.html
- Two options
 - WebTrilinos
 - Work directly with source code
 - NERSC Magellan cluster or personal laptops



Completing Exercises with WebTrilinos

- <https://www.users.csbsju.edu/trilinos/WebTrilinosMPI-shared-10.4/c++/>
- From “Insert template” select your exercise
 - “Zoltan Tutorial exercise {1,2,3}” code templates
- Click “Insert”
- Note: code doesn’t work without modifications
- Modify the exercise
 - Complete TODO action items
- Click “Run Code” when finished
 - Attempt to compile and run code
- Solutions found in code templates
 - Exercise 1,2: “Using Zoltan RCB”
 - Exercise 3: “Using Zoltan Graph”



Completing Exercises with Source Code

- Enter directory for your exercise
 - E.g., `exercise1/`
- To build on Magellan
 - “`module load trilinos`”
 - “`make`”
 - `TRILINOS_INSTALL_DIR` in `Makefile` should be set to your Trilinos installation.
- Note: code doesn't work without modifications
- Modify the exercise
 - Edit only the `zoltTutorExercise{1,2,3}.cpp` file
 - Complete TODO action items
- To launch on Magellan:
 - `qsub zoltanExercise{1,2,3}.pbs`



Completing Exercises with Source Code

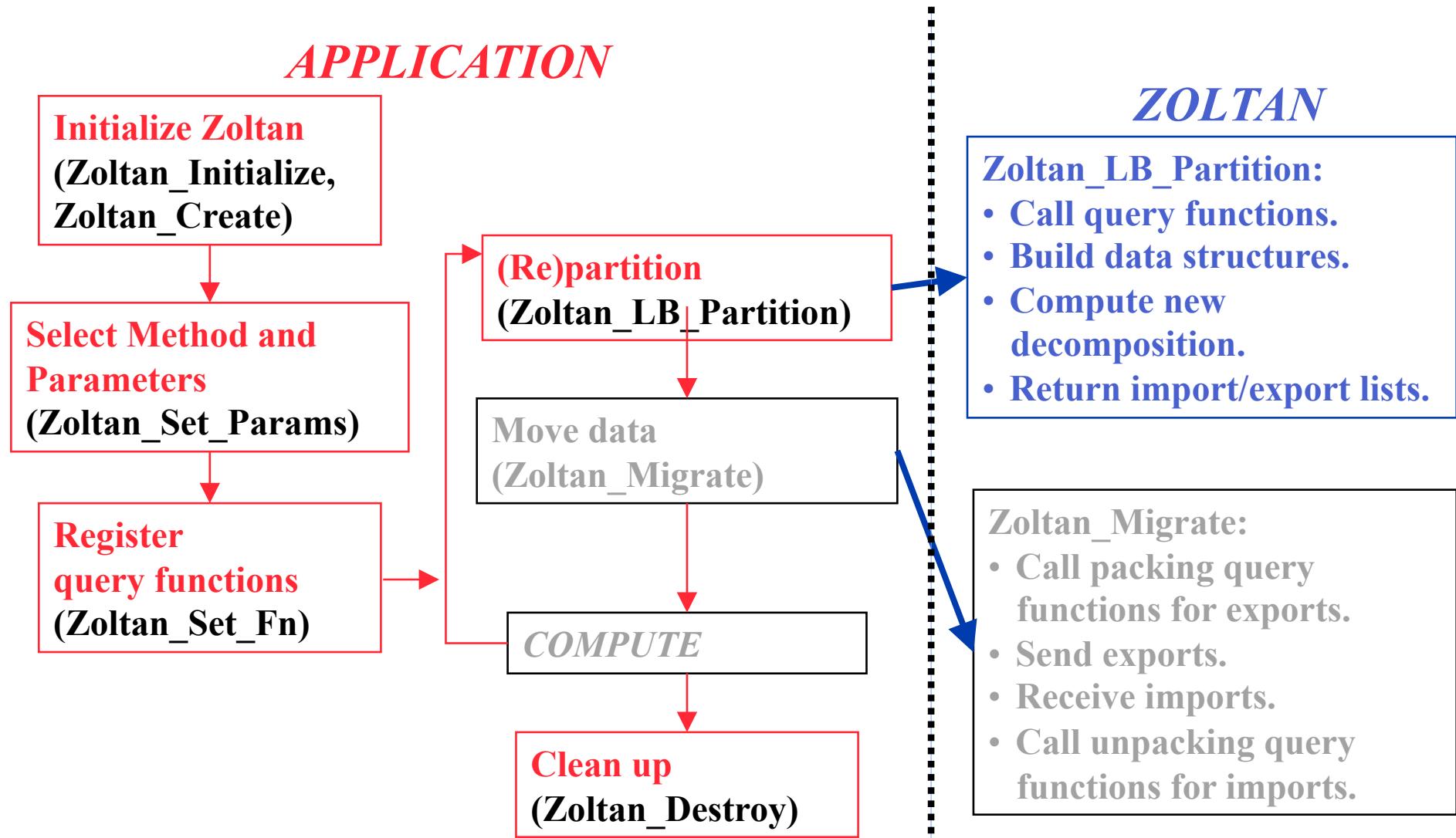
- **Solutions found in examples/**
 - **Exercise 1,2: simpleRCB.cpp**
 - **Exercise 3: simpleGRAPH.cpp**



Additional Slides: Using Zoltan



Zoltan Application Interface





Basic Zoltan Interface

- **Initialize MPI for Zoltan**

```
int err = Zoltan_Initialize(argc, argv, &version);
```

Arguments:

<i>argc</i>	The number of command-line arguments to the application.
<i>argv</i>	An array of strings containing the command-line arguments to the application.
<i>ver</i>	Upon return, the version number of the library.

- **Create Zoltan structure**

```
struct Zoltan_Struct *zz = int Zoltan_Create(communicator);
```

- **Destroy Zoltan structure**

```
Zoltan_Destroy(&zz);
```

Arguments:

<i>zz</i>	struct Zoltan_Struct **, created by Zoltan_Create , to be destroyed.
-----------	---



Setting Zoltan Parameters

- Set Zoltan parameters using the following:

```
err = Zoltan_Set_Param(zz, param_name, new_val)
```

zz Zoltan data structure
param_name char *, corresponding to the parameter name
new_val char *, new value for parameter

- For Example:

```
Zoltan_Set_Param(zz, "LB_METHOD", <meth>);
```

where <meth> = “RCB”, “GRAPH”, “HYPERGRAPH”, ...



Zoltan Query Functions

General Query Functions (2 required)

ZOLTAN_NUM_OBJ_FN	Number of items on processor
ZOLTAN_OBJ_LIST_FN	List of item IDs and weights.

Geometric Query Functions (2 required)

ZOLTAN_NUM_GEOM_FN	Dimensionality of domain.
ZOLTAN_GEOM_MULTI_FN	Coordinates of objects (all objects)
ZOLTAN_GEOM_FN	Coordinates of objects (1 object)

Graph Query Functions (2 required)

ZOLTAN_NUM_EDGES_MULTI_FN	Number of graph edges (all vertices)
ZOLTAN_NUM_EDGES_FN	Number of graph edges (1 vertex)
ZOLTAN_EDGE_LIST_MULTI_FN	List of graph edges (all vertices)
ZOLTAN_EDGE_LIST_FN	List of graph edges (1 vertex)

Or [

Or [

Or [



Setting Query Functions

- Register query functions using the following:

```
err = Zoltan_Set_<zoltan_fn_type>_Fn(zz, <zoltan_fn_type> (*fn_ptr)(),
                                         void *data);
```

zz Zoltan data structure

fn_ptr Ptr to query function that is to be registered

data Ptr to user defined data that will be passed, as an argument,
 to the function pointed to by fn_ptr

- For Example:

```
Zoltan_Set_Num_Obj_Fn(zz, get_number_of_objects, &myData);
```



ZOLTAN_NUM_OBJ_FN Prototype

```
typedef int ZOLTAN_NUM_OBJ_FN (void *data, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data
<i>ierr</i>	Error code to be set by function

Returned Value:

int	Number of objects that are assigned to the process
-----	--



ZOLTAN_OBJ_LIST_FN Prototype

```
typedef void ZOLTAN_OBJ_LIST_FN (void *data, int num_gid_entries, int num_lid_entries,  
                                ZOLTAN_ID_PTR global_ids, ZOLTAN_ID_PTR local_ids, int wgt_dim,  
                                float *obj_wgts, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data.
<i>num_gid_entries</i>	The number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.
<i>num_lid_entries</i>	The number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (It should be zero if local ids are not used.)
<i>global_ids</i>	Upon return, an array of unique global IDs for all objects assigned to the processor.
<i>local_ids</i>	Upon return, an array of local IDs, the meaning of which can be determined by the application, for all objects assigned to the processor. (Optional.)
<i>wgt_dim</i>	The number of weights associated with an object (typically 1), or 0 if weights are not requested. This value is set through the parameter OBJ_WEIGHT_DIM.
<i>obj_wgts</i>	Upon return, an array of object weights. Weights for object <i>i</i> are stored in <i>obj_wgts[(i-1)*wgt_dim:i*wgt_dim-1]</i> . If <i>wgt_dim</i> =0, the return value of <i>obj_wgts</i> is undefined and may be NULL.
<i>ierr</i>	Error code to be set by function.



ZOLTAN_NUM_GEOM_FN Prototype

```
typedef int ZOLTAN_NUM_GEOM_FN (void *data, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data
<i>ierr</i>	Error code to be set by function

Returned Value:

int	Number of values needed to express the geometry of an object
-----	--



ZOLTAN_GEOM_MULTI_FN Prototype

```
typedef void ZOLTAN_GEOM_MULTI_FN (void *data, int num_gid_entries, int num_lid_entries, int num_obj,
                                    ZOLTAN_ID_PTR global_ids, ZOLTAN_ID_PTR local_ids,
                                    int num_dim, double *geom_vec, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data.
<i>num_gid_entries</i>	Number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.
<i>num_lid_entries</i>	Number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (Zero if local ids not used.)
<i>num_obj</i>	Number of object IDs in arrays <i>global_ids</i> and <i>local_ids</i> .
<i>global_ids</i>	Array of global IDs of objects whose geometry values should be returned.
<i>local_ids</i>	Array of local IDs of objects whose geometry values should be returned. (Optional.)
<i>num_dim</i>	Number of coordinate entries per object (typically 1, 2, or 3).
<i>geom_vec</i>	Upon return, an array containing geometry values. For object <i>i</i> (specified by <i>global_ids</i> [<i>i</i> * <i>num_gid_entries</i>] and <i>local_ids</i> [<i>i</i> * <i>num_lid_entries</i>], <i>i</i> =0,1,..., <i>num_obj</i> -1), coordinate values should be stored in <i>geom_vec</i> [<i>i</i> * <i>num_dim</i> :(<i>i</i> +1)* <i>num_dim</i> -1].
<i>ierr</i>	Error code to be set by function.



ZOLTAN_GEOM_FN Prototype

```
typedef void ZOLTAN_GEOM_FN (void *data, int num_gid_entries, int num_lid_entries,
                             ZOLTAN_ID_PTR global_id, ZOLTAN_ID_PTR local_id, double *geom_vec,
                             int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data.
<i>num_gid_entries</i>	Number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.
<i>num_lid_entries</i>	Number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (Zero if local ids are not used.)
<i>global_id</i>	Global ID of object whose geometry values should be returned.
<i>local_id</i>	Local ID of the object whose geometry values should be returned.
<i>geom_vec</i>	Upon return, an array containing geometry values.
<i>ierr</i>	Error code to be set by function.



ZOLTAN_NUM_EDGES_MULTI_FN

```
typedef void ZOLTAN_NUM_EDGES_MULTI_FN(void *data, int num_gid_entries, int num_lid_entries,  
    int num_obj, ZOLTAN_ID_PTR global_ids,  
    ZOLTAN_ID_PTR local_ids, int *num_edges, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data
<i>num_gid_entries</i>	Number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.
<i>num_lid_entries</i>	Number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (Zero if local ids are not used.)
<i>num_obj</i>	Number of object IDs in arrays <i>global_ids</i> and <i>local_ids</i> .
<i>global_ids</i>	Array of global IDs of objects whose number of edges should be returned.
<i>local_ids</i>	Array of local IDs of objects whose number of edges should be returned. (Optional.)
<i>num_edges</i>	Upon return, an array containing numbers of edges. For object <i>i</i> (specified by <i>global_ids</i> [<i>i</i> * <i>num_gid_entries</i>] and <i>local_ids</i> [<i>i</i> * <i>num_lid_entries</i>], <i>i</i> =0,1,..., <i>num_obj</i> -1), the number of edges should be stored in <i>num_edges</i> [<i>i</i>].
<i>ierr</i>	Error code to be set by function.



ZOLTAN_NUM_EDGES_FN

```
typedef int ZOLTAN_NUM_EDGES_FN (void *data, int num_gid_entries, int num_lid_entries,  
                                ZOLTAN_ID_PTR global_id, ZOLTAN_ID_PTR local_id, int *ierr);
```

Arguments:

data Pointer to user-defined data.

num_gid_entries Number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.

num_lid_entries Number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (It should be zero if local ids are not used.)

global_id Global ID of the object for which the number of edges should be returned.

local_id Local ID of the object for which the number of edges should be returned.

ierr Error code to be set by function.

Returned Value:

int Number of edges for the object identified by *global_id* and *local_id*.



ZOLTAN_EDGE_LIST_MULTI_FN

```
typedef void ZOLTAN_EDGE_LIST_MULTI_FN (void *data, int num_gid_entries, int num_lid_entries, int num_obj,
                                         ZOLTAN_ID_PTR global_ids, ZOLTAN_ID_PTR local_ids,
                                         int *num_edges, ZOLTAN_ID_PTR nbor_global_id,
                                         int *nbor_procs, int wgt_dim, float *ewgts, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data.
<i>num_gid_entries</i>	Number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.
<i>num_lid_entries</i>	Number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (Zero if local ids are not used.)
<i>num_obj</i>	The number of object IDs in arrays <i>global_ids</i> and <i>local_ids</i> .
<i>global_ids</i>	Array of global IDs of objects whose edge lists should be returned.
<i>local_ids</i>	Array of local IDs of objects whose edge lists should be returned. (Optional.)
<i>num_edges</i>	Array containing numbers of edges for each object in <i>global_ids</i> . For object <i>i</i> (specified by <i>global_ids</i> [<i>i</i> * <i>num_gid_entries</i>] and <i>local_ids</i> [<i>i</i> * <i>num_lid_entries</i>], <i>i</i> =0,1,..., <i>num_obj</i> -1), the number of edges is stored in <i>num_edges</i> [<i>i</i>].
<i>nbor_global_id</i>	Upon return, array of global IDs of objects sharing edges with the objects specified in <i>global_ids</i> . For object <i>i</i> (specified by <i>global_ids</i> [<i>i</i> * <i>num_gid_entries</i>] and <i>local_ids</i> [<i>i</i> * <i>num_lid_entries</i>], <i>i</i> =0,..., <i>num_obj</i> -1), edges are stored in <i>nbor_global_id</i> [<i>sum</i> * <i>num_gid_entries</i>] to <i>nbor_global_id</i> [<i>(su+num_edges[i])*num_gid_entries-1</i>], where <i>sum</i> = the sum of <i>num_edges[j]</i> for <i>j</i> =0,1,..., <i>i</i> -1.
<i>nbor_procs</i>	Upon return, an array of processor IDs that identifies where the neighboring objects reside. For neighboring object <i>i</i> (stored in <i>nbor_global_id</i> [<i>i</i> * <i>num_gid_entries</i>]), the processor owning the neighbor is stored in <i>nbor_procs</i> [<i>i</i>].
<i>wgt_dim</i>	Number of weights associated with an edge (typically 1), or 0 if edge weights are not requested. This value is set through the parameter EDGE_WEIGHT_DIM.
<i>ewgts</i>	Upon return, array of edge weights, where <i>ewgts</i> [<i>i</i> * <i>wgt_dim</i> :(<i>i</i> +1)* <i>wgt_dim</i> -1] corresponds to the weights for the <i>i</i> th edge. If <i>wgt_dim</i> =0, the return value of <i>ewgts</i> is undefined and may be NULL.
<i>ierr</i>	Error code to be set by function.



ZOLTAN_EDGE_LIST_FN

```
typedef void ZOLTAN_EDGE_LIST_FN (void *data, int num_gid_entries, int num_lid_entries, ZOLTAN_ID_PTR global_id,  
                                 ZOLTAN_ID_PTR local_id, ZOLTAN_ID_PTR nbor_global_id, int *nbor_procs,  
                                 int wgt_dim, float *ewgts, int *ierr);
```

Arguments:

<i>data</i>	Pointer to user-defined data.
<i>num_gid_entries</i>	Number of array entries used to describe a single global ID. This value is the maximum value over all processors of the parameter NUM_GID_ENTRIES.
<i>num_lid_entries</i>	Number of array entries used to describe a single local ID. This value is the maximum value over all processors of the parameter NUM_LID_ENTRIES. (It should be zero if local ids are not used.)
<i>global_id</i>	Global ID of the object for which an edge list should be returned.
<i>local_id</i>	Local ID of the object for which an edge list should be returned.
<i>nbor_global_id</i>	Upon return, an array of global IDs of objects sharing edges with the given object.
<i>nbor_procs</i>	Upon return, an array of processor IDs that identifies where the neighboring objects reside.
<i>wgt_dim</i>	Number of weights associated with an edge (typically 1), or 0 if edge weights are not requested. This value is set through the parameter EDGE_WEIGHT_DIM.
<i>ewgts</i>	Upon return, an array of edge weights, where <i>ewgts</i> [<i>i</i> * <i>wgt_dim</i> :(<i>i</i> +1)* <i>wgt_dim</i> -1] corresponds to the weights for the <i>i</i> th edge. If <i>wgt_dim</i> =0, the return value of <i>ewgts</i> is undefined and may be NULL.
<i>ierr</i>	Error code to be set by function.



Partitioning Interface

Zoltan computes the **difference** (Δ) from current distribution

Choose between:

- a) Import lists (data to import **from** other procs)
- b) Export lists (data to export **to** other procs)
- c) Both (the default)

```
err = Zoltan_LB_Partition(zz,
    &changes, /* Flag indicating whether partition changed */
    &numGidEntries, &numLidEntries,
    &numImport, /* objects to be imported to new part */
    &importGlobalGids, &importLocalGids, &importProcs, &importToPart,
    &numExport, /* # objects to be exported from old part */
    &exportGlobalGids, &exportLocalGids, &exportProcs, &exportToPart);
```



Solution Output for 4 Parts



RCB (Exercises 1 and 2)

6 (0,5)	12 (1,5)	18 (2,5)	24 (3,5)	30 (4,5)	36 (5,5)
5 (0,4)	11 (1,4)	17 (2,4)	23 (3,4)	29 (4,4)	35 (5,4)
4 (0,3)	10 (1,3)	16 (2,3)	22 (3,3)	28 (4,3)	34 (5,3)
3 (0,2)	9 (1,2)	15 (2,2)	21 (3,2)	27 (4,2)	33 (5,2)
2 (0,1)	8 (1,1)	14 (2,1)	20 (3,1)	26 (4,1)	32 (5,1)
1 (0,0)	7 (1,0)	13 (2,0)	19 (3,0)	25 (4,0)	31 (5,0)

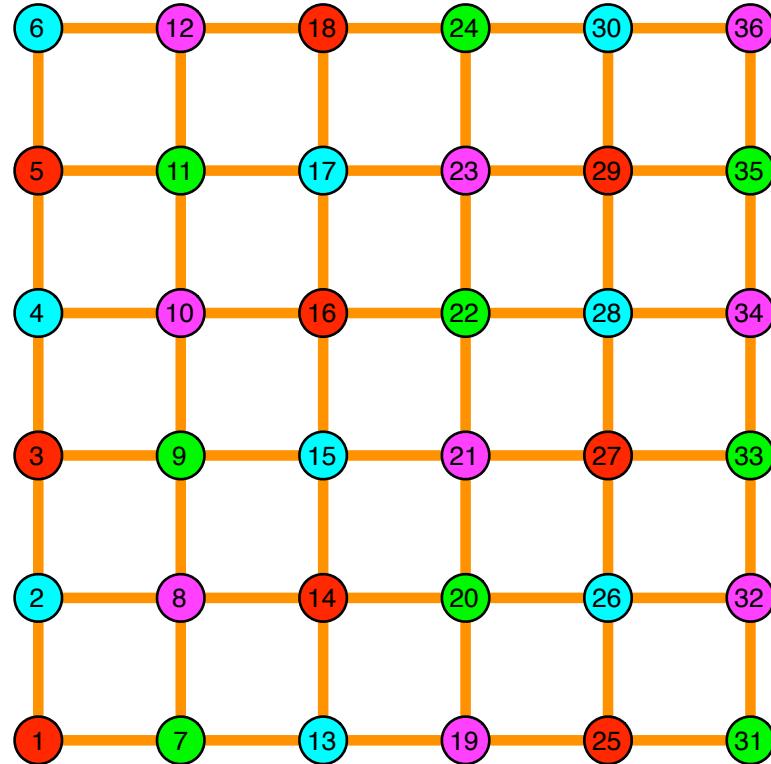
Initial Partition

6 (0,5)	12 (1,5)	18 (2,5)	24 (3,5)	30 (4,5)	36 (5,5)
5 (0,4)	11 (1,4)	17 (2,4)	23 (3,4)	29 (4,4)	35 (5,4)
4 (0,3)	10 (1,3)	16 (2,3)	22 (3,3)	28 (4,3)	34 (5,3)
3 (0,2)	9 (1,2)	15 (2,2)	21 (3,2)	27 (4,2)	33 (5,2)
2 (0,1)	8 (1,1)	14 (2,1)	20 (3,1)	26 (4,1)	32 (5,1)
1 (0,0)	7 (1,0)	13 (2,0)	19 (3,0)	25 (4,0)	31 (5,0)

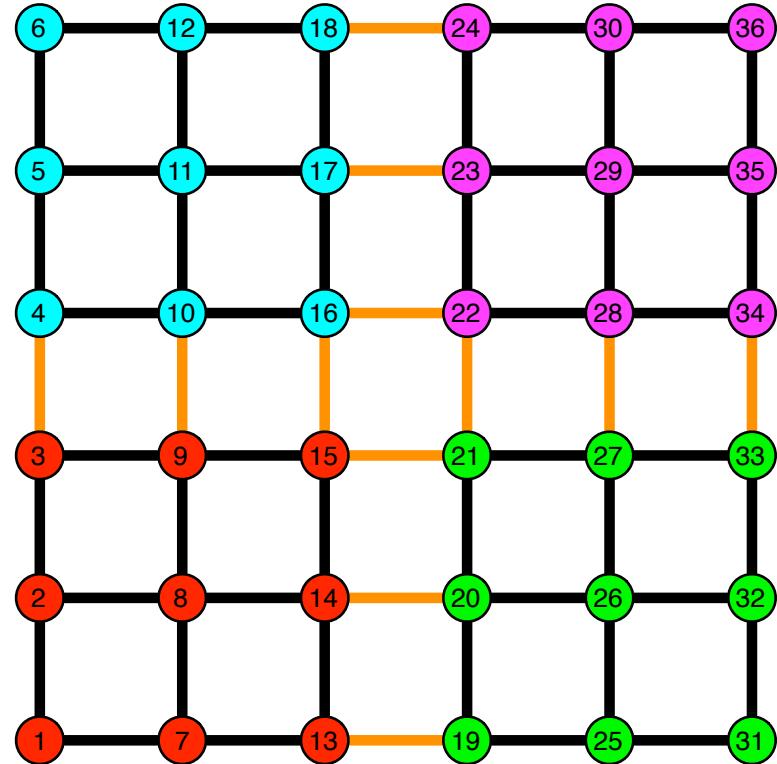
RCB Partition



Graph (Exercise 3)



Initial Partition (cut = 60)



Graph Partition (cut = 12)

Cut edges highlighted in orange



Time Permitting....

- Additional things to try
 - Other exercise
 - For those completing exercise 1 or 2, try exercise 3.
 - For those completing exercise 3, try exercise 1 or 2.
 - Weighting objects
 - How does this affect the resulting partition?
 - Weighting edges (for exercise 3)
 - How does this affect the resulting partition?
 - Different geometric methods (for exercises 1, 2)
 - Try RIB, HSFC
 - May need to vary number of parts to see difference