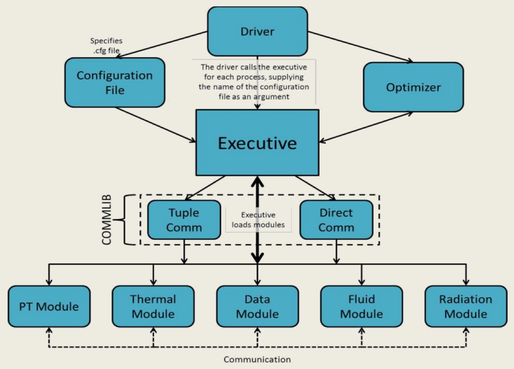
# OpenDIEL: A Parallel Open Source Workflow Engine

Students: Tristin Baker (Maryville College), Jordan Scott (Morehouse College),
Zac Trzil (UTK), Mentor: Dr. Kwai Wong (UTK)

## What is OpenDIEL?

OpenDIEL stands for **open Distributive Interoperable Executable Library**. It facilitates communication between loosely coupled modules leveraging the **MPI** (Message Parsing Interface) system. With a user-defined configuration file and driver, openDIEL can output a **single executable** managing the communication between multiple modules, allowing for efficient data sharing between unique, data-intensive modules.

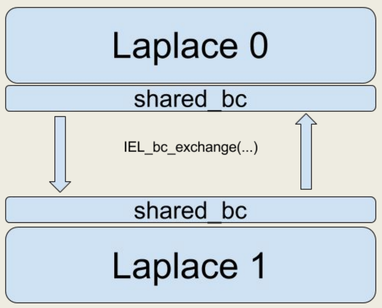



## Better Direct Communication Example

Previously, the examples showing off the direct communication aspect of the OpenDIEL did not adequately show the OpenDIEL's capabilities as well as it could have. Therefore, an example using **Laplace Transformation Matrices** was created.

```
/* Set the shared_bc write equal to the top row of t.
 * shared_bc write is specified by the user in the workflow.cfg file */
for(i = 500; i < 1000; i++) {
  exec_info->shared_bc[i] = t[0][i%500];
}

/* This function sends the shared_bc 'write' values to the
 * "laplace0" shared_bc 'read' values and 'reads' the shared_bc
 * 'write' values from the "laplace0" module.
 */
IEL_bc_exchange(exec_info, "laplace0", &request);

for(i = 1000; i < 1500; i++) {
  exec_info->shared_bc[i] = t[nr-1][i%1000];
}

/* This function sends the shared_bc 'write' values to the
 * "laplace2" shared_bc 'read' values and 'reads' the shared_bc
 * 'write' values from the "laplace2" module.
 */
IEL_bc_exchange(exec_info, "laplace2", &request);

/* Set the top row of t equal to what is received by the IEL_bc
 * exchange from laplace 0. */
for(i = 500; i < 1000; i++) {
  t[0][i%500] = exec_info->shared_bc[i];
}

for(i = 1000; i < 1500; i++) {
   t[nr-1][i%1000] = exec_info->shared_bc[i];
 }
```

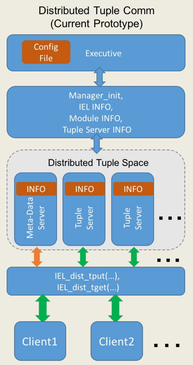
```
{
  function="laplace1"
  args=()
  libtype="static"
  shared_bc_read=([0,499],[1500, 2000])
  shared_bc_write=([500,999],[1000, 1499])
  size=1
},
```




This provided a clearer example of what was really going on in the direct communication.

## Distributed Tuple Space

**Overview:** Modules may use a distributed array of tuple servers to store data in system memory that other modules may access. The sender places the data using IEL_dist_tput() and a user-defined data tag as an argument of the function. The receiver, using the same tag and the IEL_dist_tget() function will be able to retrieve the data from the distributed array.



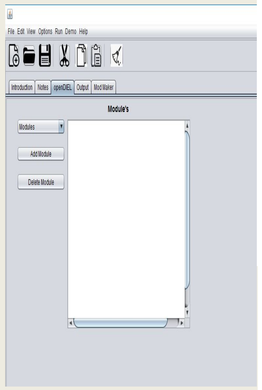

**Sending data:** A client can send data to the distributed array of tuple servers by calling IEL_dist_tput():

- Distributes data even among available tuple servers
- Sets up two arrays of meta data:
  - The server rank in the order used
  - The size of the data sent to that tuple server
- Stores the meta-data on the first tuple server

## User Interface Improvement

The User Interface serves to replace how OpenDIEL currently operates. It does this by allowing users to enter information directly into a single Interface as opposed to editing multiple files, and lines of code.

The Interface will allow for users to enter modules, groups and sets. This information will be used to make the "workflow.cfg" file necessary for running OpenDIEL.




Future works for the User Interface are

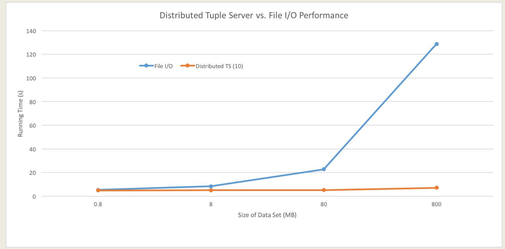
1. Fully inplament functinoallity listed above.
2. Allow users to convert their C or Fortran code into a OpenDIEL module.
3. To implement an open database of Modules.
4. Launch OpenDIEL executables on HPCs through the GUI via SSH.
5. Create runscripts by analyzing user code and determining how many processes the OpenDIEL executable needs.

## Distributed Tuple Space

**Receiving data:** A client can receive data stored on the distributed array of tuple servers by calling IEL_dist_tget():

- Queries the meta data server for the information corresponding to the tag the function was called with
- Uses the meta data to pull the data from the servers in the order in which is was stored
- Reconstructs the data into an array that the client passed to the function

**Results:** Distributing data across multiple tuple servers shows almost no increase in program running time while file I/O grows at an exponential rate.




The above graph shows the running time of two modules sharing data through openDIEL, one sending and one receiving data.

## Future Work

Taking advatage of the distributed tuple servers, data can be duplicated across the servers to provide RAID-like failure protection. A single or subset of tuple servers can be designated to backup to disk without interfering with the primary processes and their data.

Direct communication should take place in local a shared_bc rather than a global. In its current implementation, the shared_bc wastes memory and is not scalable.

## Acknowledgements

This project is made possible by funding provided by the NSF, facilities provided by the University of Tennessee, and computational resources provided by XSEDE.