# GSOC 2017 PROPOSAL

## PERSONAL DETAILS


| | | |
|---|---|---|
| Name | : | Praveen velliengiri |
| Age | : | 18 |
| College | : | Psg college of technology |
| Location | : | Coimbatore,India(UTC + 5.30) |
| Course | : | Data Science |
| Degree | : | Msc |
| Email | : | praveenvelliengiri@gmail.com<br>kadyathack@gmail.com |
| Factors affects Availability | : | Exams in july |
| Intended dates | : | Start date - 19-May-2017<br>End date - 25-August-2017 |
| Time to spent | : | 25-30 hours per week(june)<br>25 hours per week(july)<br>25 hours per week(August) |
| Phone no | : | +91 7538888190 |
| Address | : | 5/223,Pachakattupalayam<br>Chengappalli,Uthukuli Tuluk,<br>Tamil nadu,India-638812 |
| Github | : | https://github.com/Praveenv98 |


## Background

I am studying Msc data science but I have more interest in High Performance Computing than Data Analytics so I decided to pursue a career in HPC.

Related Courses taken in college are

1. Computer Architecture
2. Operating Systems
3. Object Oriented Programming
4. Algorithms

I have started to learn Asynchronous Programming. Actually, I got interest in Open Source Projects and created an account in github. But as a beginner I found contributing to the open source projects is tedious task. Eventually I came to know about GSOC -" Where we can get mentoring from the Organizations for doing projects" so I decided to take part in gsoc 2017 and to gain experience in Open source community. This is my first attempt at gsoc.Then I found STE||AR Project and really like the execution model of hpx (ParallelX) and one of the core rules of hpx "Prefer moving work to the data than data to the work". I look Gsoc 2017 as a good opportunity to learn new things about HPC and also make good start in contributing open source projects.I am looking to be in touch with domain experts and to make good relationship with other contributors of hpx.

## Programming Background

I have worked on Object Oriented projects using c++ and python. Few projects are:

1. A Web Content Analysis Model - a c++ search engine with frontend provided by python ,extracting the content in webpages(limited to few websites) with the help of tags provided by the languages such as HTML etc and storing the content and web page id within a file index which helps in faster lookup's and assigning priorities to the contents based on their location in webpage.
2. A implement of n-ary tree and trie data structure for the faster searching of files within a directory using kernel modules.
3. Currently implementing the techniques of Hot Code Swapping in c++.

I recently gave a speech about Large Scale Computing with MPI in Google developers group,coimbatore. I have to agree that how worse

the MPI will land you when moving to exascale computing. I have also gone through some of the techniques that charm++ using in data distribution and load balancing.

I have not done any work in this area before but I will work as much as I can in completing this project.

Experience

| | |
|---|---|
| C++ | - 3.5 |
| C++ standard library | - 3.5 |
| Boost c++ libraries | - 3 |
| Git | - 1 |
| CUDA | - 0 |

Development tools that I'm using is Eclipse,Atom and editors such as vim,Emac.

I am somewhat similar with Doxygen documentation tool.I will rate myself in this as (1.5 out of 5).

NOTE : **Extension** in this proposal denote my future work in hpx as a contributor after this summer project.

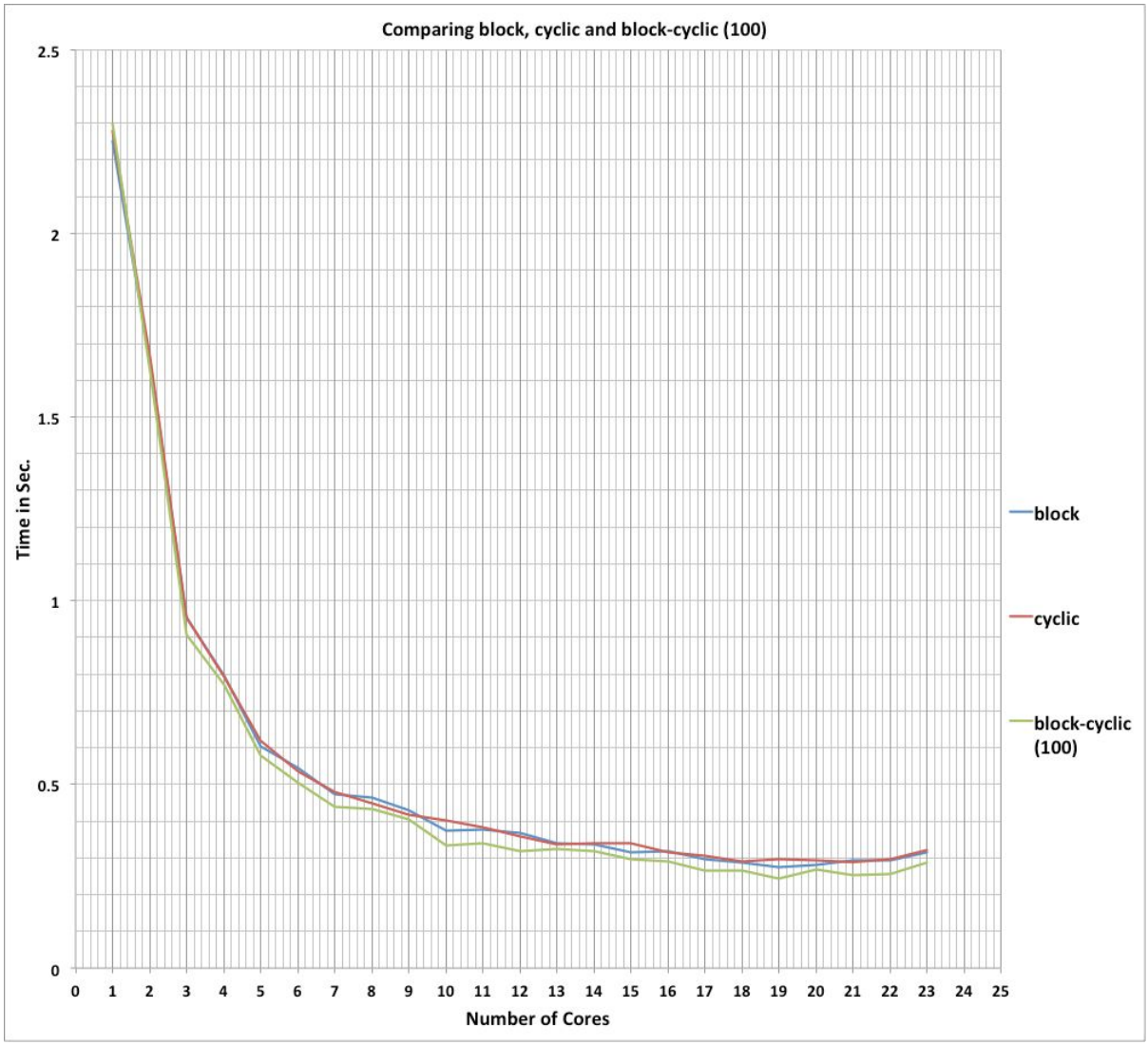
## *why data distribution is important?*

The specification of data partitioning is the crucial & most critical part in the parallelization process requires a detailed Knowledge of algorithm,Architecture and cost of communication involves. The penalty of choosing a wrong distribution will result in the chance getting worst behaviour if even the system uses low latency network.So, the method(policy) in which data is distributed among locales will have a larger impact in the Performance and the Scalability of the Application.

One such situation is data scientists crunch and analyze the large amount of data by applying machine learning algorithms on it to get information from data. The parallel processing is the key for many ML algorithms to give good performance. So If we do some hard work

in distributing the data among the locales then we can achieve the better Scalability and Performance even in the Exascale computing Systems.

Another situation is in calculating the Mandelbort pixel value the block-cyclic distribution gives a good performance over other two policies.




It is taken from data distribution in hpx by Bibek Ghimire thesis as a reference.

So, we can strongly argue that the data distribution policy will definitely affect the performance of Large Scale Systems.

## *Why hpx need data distribution?*

In MPI model, each process has it's own address space and communicates with other process to access their address space in a

MPI_SEND and MPI_RECEIVE pattern. It obviously leads to some blocking manner between process.MPI provides interfaces for data distribution but it is very difficult to program with send & receive interface in MPI. In MPI deadlocks may be occur when exchanging between nodes. But HPX supports AGAS so we can easily support these distribution policies with good elegant API's so that programmer can opt HPX over the traditional solutions for scalability.

***Looking for Solutions?***

I have referred many research papers to address this issue in the most optimal way. One such solution is Chapel domain maps. Data distribution among the nodes can be solved by domain maps in a optimal way so I have used domains map idea from chapel to address this issue.

While designing associative domains , I found out dynamic distributed dimensional data model(D4M) database and computation system use a similar kind of distribution policies for data distribution across their databases. And some complex database queries can be modified into mathematical operations on the associative arrays.This results in faster access of data in database.

# Distributed Component Placement

## Abstract

Implementation of Embedded Domain Specific Language to specify the placement policies for components in Distributed Systems by implementing domain maps in hpx. And also building allocators over domain maps to use with c++ standard library containers.

# Scope

Allow Programmers to code Parallel Algorithms quickly without much concentration on initial placement of data. To overcome the inability of Partitioned Global Address Space(PGAS) systems in Scalability of Large Scale Applications. To give a chance to dynamic data distribution techniques in Scaling Large scale applications which is not possible in today PGAS systems.

**AGAS vs PGAS**

The benefit in using AGAS over PGAS is we can move remote objects in physical space without changing their virtual name.

The PGAS is less well-suited for event-driven and active messages execution where short-lived computations is performed on global data.The static nature of PGAS restricts the system ability to load balance computation and communication

# Design

## Embedded Domain Specific Language

I have decided to design a EDSL for hpx library by creating a standard set of API's that can be integrated with hpx library by end of this summer project.

Rules followed while designing API's

1. Unambiguousness
2. User friendliness
3. Make one task per API.

Because the user need only that functionality to be done without interfering anything else. Of course, it hides the actual implementation of domain map class and functionality that does data distribution from the user and also make them integrate well with the distributed data structures. It gives the programmer ease of writing Parallel Algorithms by

using these standard api's. So the programmer concentrate in implementing the algorithm without worrying about data placement in Active Global Address Space(AGAS).

## *what are domains?*

I have derived the concept of domain maps and domain from Chapel programming language.Domains describes a **collection of names** for data.These names are referred to as indices of the domain. All these indices are themselves values with same type.
Types of these index set includes primitive types and class references, Unions and User defined data types.The distribution is specified over domains those index set is distributed over the target locales. The computations can be done asynchronously over locales if needed.

I have planned to implement these three types of domains during this summer:

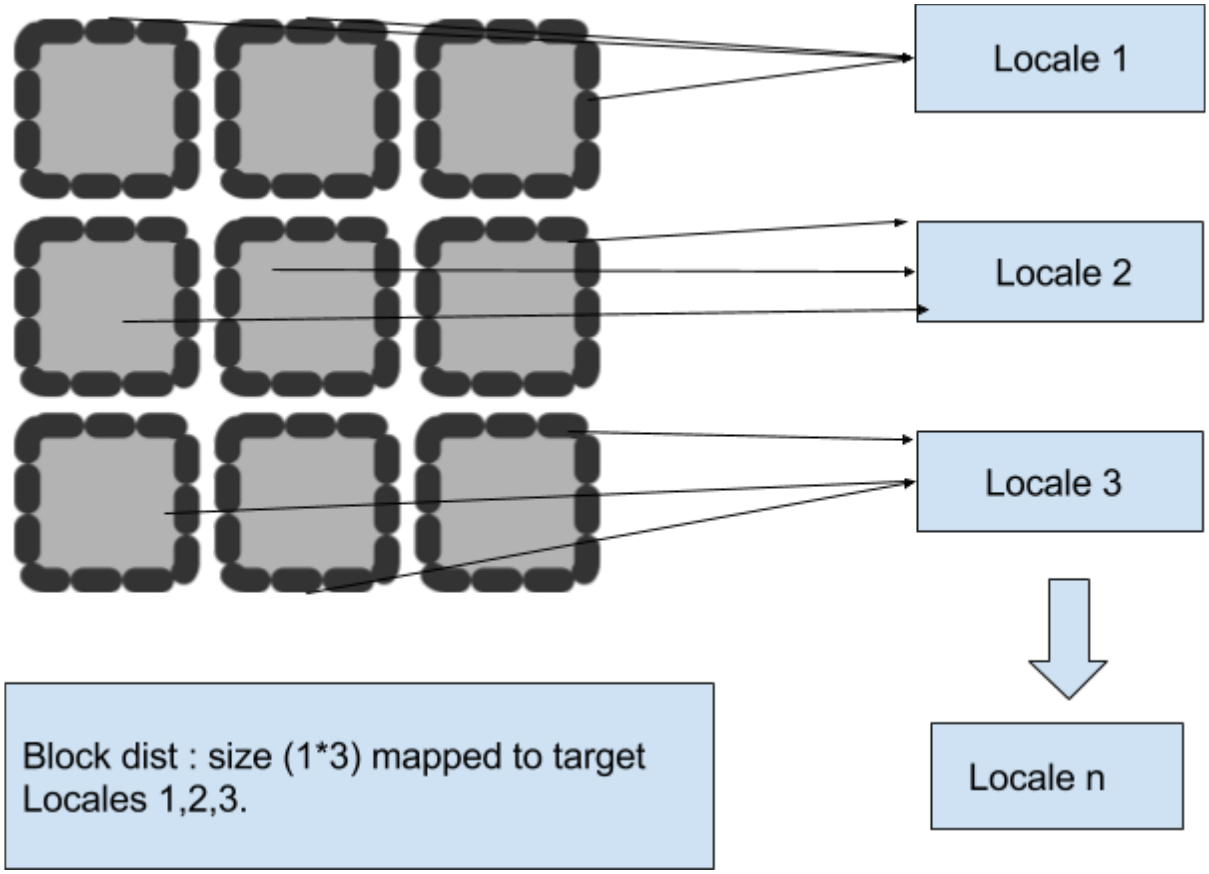
Arithmetic Domains - a rectilinear set of cartesian indices.
Sparse Domains - a Subdomain useful in handling large sparse matrix with minimum storage.
Associative Domains - set of indices those type can be any allowed types.(Key,Value) pairs.

It simplifies reasoning about the program by keeping the implementation and relative alignment of arrays separate.

Arrays associate the variables or elements with set of indices of domain i.e arrays must be declared over the domain.

### *How to distribute Domains to locales?*

So, here our Key concept of our project **Domain Map** -a mapping from domain index set to the locales




## Data Distribution Policies

This design is very much concerned about the core rules of hpx should not be violated. Mainly focused on hiding latencies over eliminating it , supports Adaptive locality control instead of static data distribution and moving work to data over data to work.

Chapel supports huge number of real time applications using its own constructs so deriving from chapel and modifying according to our need is easier and productive over designing a new approach for data distribution job from scratch.

In Static data distribution, data is distributed among the nodes at compile time,whereas in dynamic data distribution the data is

distributed at runtime.But chapel provides static data distribution as it follows partitioned global address space.The scope of this project is to elegant way programming the data distribution policies among the nodes and also add some mechanisms which can form the basis for dynamic data distribution. The current implementation of distributed vector in hpx works good for regular data distribution but in case of irregular data distribution this will be cumbersome. So Domains and domain maps will be a better solution.This results ease of programming i.e hey I have a domain (arithmetic,sparse,associative and more) and distribute it according to the maps I have provided (storage layouts,policies etc) and now I can created any number of arrays over the domain and the system will take care of its distribution over locales.

For design we want to have three ***global descriptors*** namely :

Domain Map Descriptor
Domain Descriptor
Array Descriptor

## Global Domain Map Descriptor class

We can implement the domain map descriptor as a template class and at any point of time domain map descriptor stores required things to characterize the domain map as a whole.

Domain Map standard Interface is must implemented for the support of data distribution among Locales.

This class must derived from the distribution class(Base Class) because in many applications the traditional distribution policies like block,cyclic,block-cyclic will have little to do. So the user define their own distribution policy and inherit domain map class from their distribution (user-defined class) class and add functionalities for their use cases. The built-in classes and user defined classes should follow the same method to avoid performance cliff.

class Global_domain_map : distribution_class

start_index // cyclic distribution
block_size // block distribution
Start_index,block_size // block-cyclic distribution
storage_layout
distribution_policy // three above specified
target_locales // target node's
… … .

Functionalites:

Global_domain_map :: Global_domain_map() //constructor

Global_domain_map :: Global_domain_map(*/ argument_list /*)

// implementation of move,copy constructors for this to be // discussed with mentors because we can create as many // domains under single domain map.

Global_domain_map :: bool Equality(domain_map &t1,domain_map &t2) -returns true if distribution policies are same. Operator == overloads includes

// Almost similar message-passing interface should be provided to communicate between domain descriptor in case of insufficient memory for allocating components in locale node

Global_domain_map::add_locales()

Global_domain_map::index_ownership(index)

It takes index has a argument and returns the global address (GID) of locality in which the index resides.

// adding some functionalities to achieve better performance when integrated with hpx

**Creating domain descriptor class**

It also serves as a ***factory class*** that creates new domain descriptors for each ***domain value*** created using its domain map.

***Extension :*** After summer I would like to add some more methods for dealing irregular domains such as graph which has many practical uses

## Global Domain Descriptor Class

It has important job to store the representation of domain's index set and it should distribute the indexes over the locals according to the distribution policy provided in Domain map descriptor class.It can be only done in this class.

```
class  global_domain_descriptor
      {
        dimension      // dimension of domain
        T  Index_type    //  type of index of domain
       bool  stridable   // Set to 1 if stridable
        dist              // reference to distribution
       ……
      }
```

Supported Functions

Global_domain_descriptor :: Dist_policy( ) // returns distribution policy

Global_domain_descriptor :: Get_indices() // get the associated indices of domains

Global_domain_descriptor :: Set_indices() // set the indices for domain

Global_domain_descriptor :: modify_indices() // Array is associated with the domain so changing the domain will lead to losing the array values

Global_domain_descriptor :: iter() // returns a parallel iterator that points to locales in which data is residing ,that introduces parallelism between locales having the data. so parallel task can be made asynchronous.

Global_domain_descriptor :: subdomain()  // ability to create subdomain for given domain with provided dimension of subdomain

Global_domain_descriptor :: sparse()  // ability to create sparse domain from given domain

Global_domain_descriptor ::make_stridable()  // to create a stridable domain from another domain.

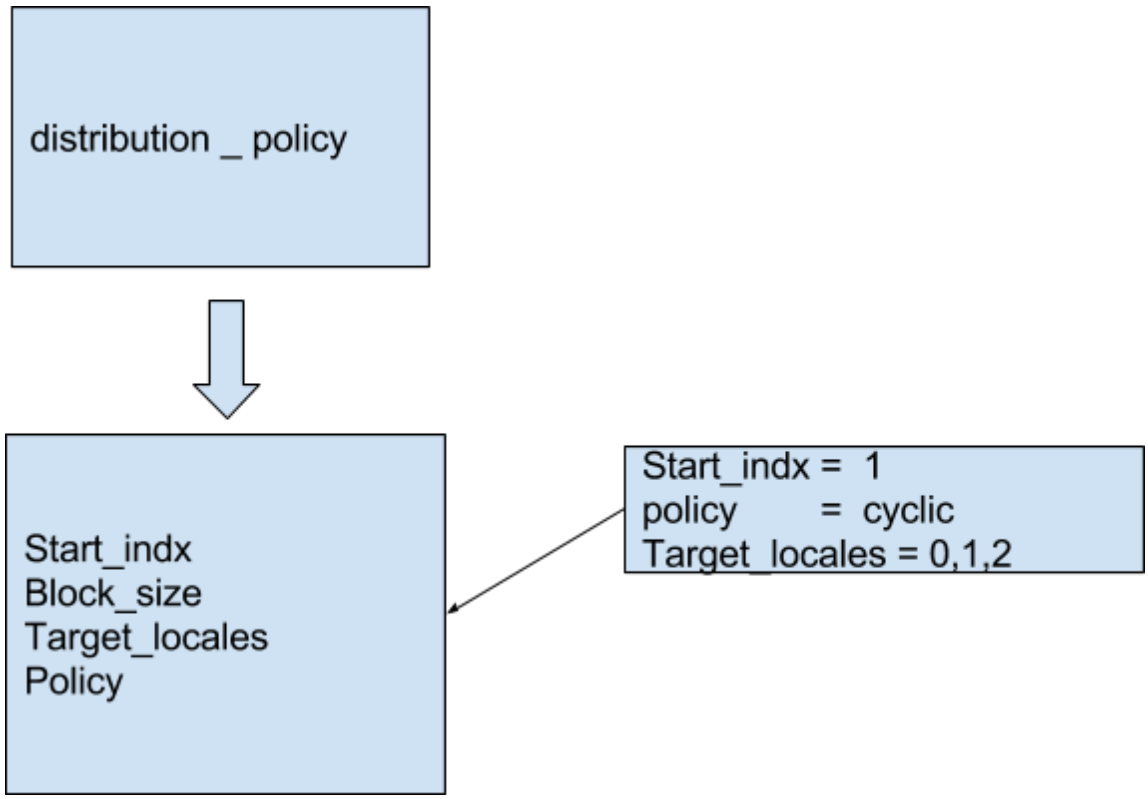
Global_domain_descriptor :: Is_rectangular()

Global_domain_descriptor :: Is_sparse()

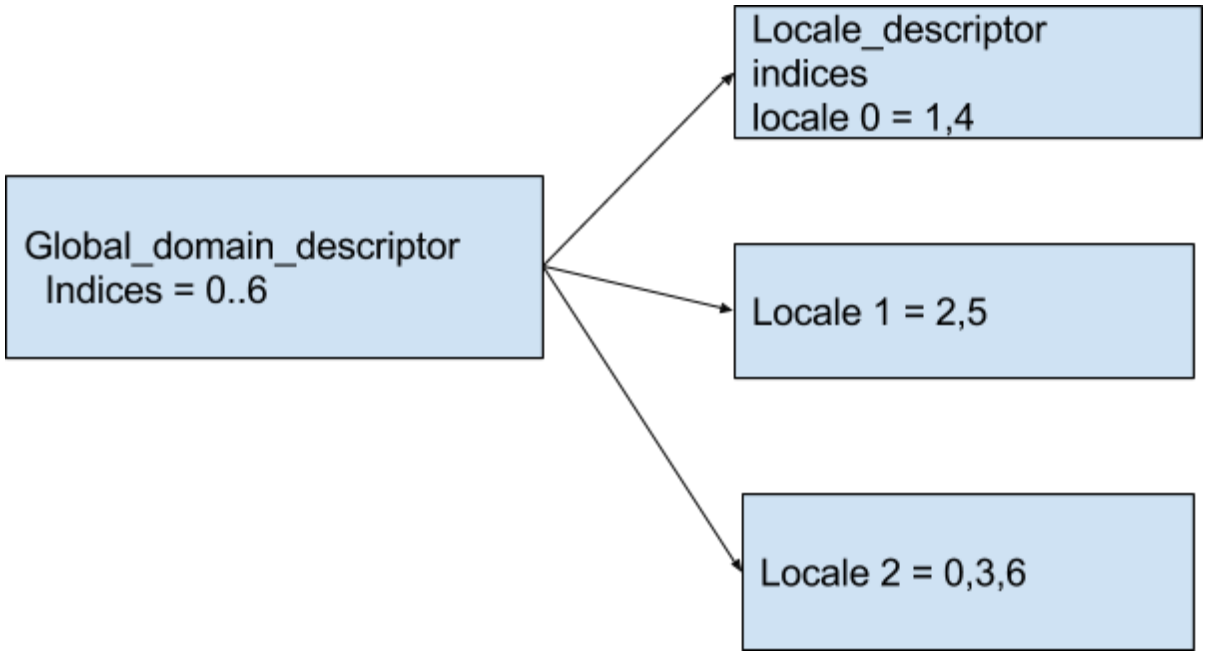
Global_domain_descriptor :: Is_associative()

We can create local descriptor in each locales they store their respective indices.

Like domain map descriptor it is also a factory class that creates a Array Descriptor.




This creates the instance of domain_map_class ,then




***Extension*** : I would like add some more functionalities for what I have added in domain_map class.For eg : creating an iterator for irregular domains.

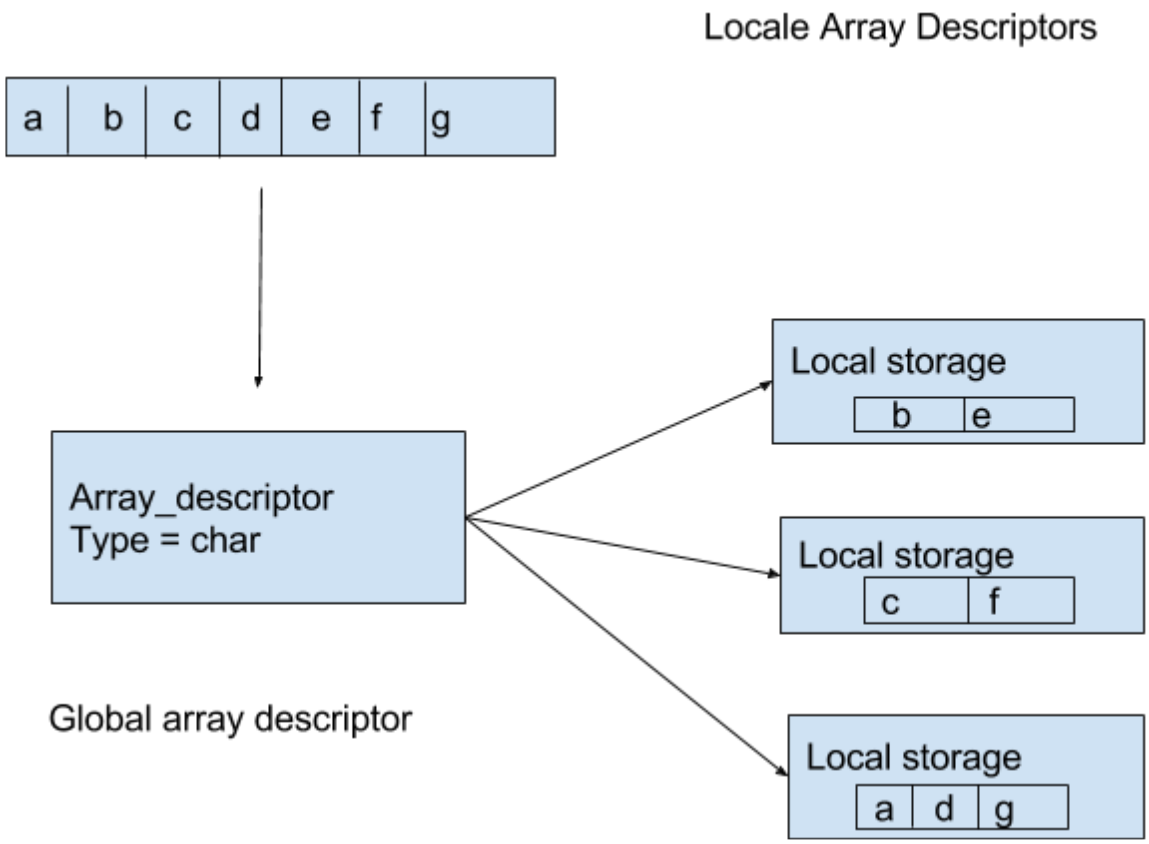
## Global Array Descriptor Class

```
Class global_array
        {
          T  eletype        // type of array element
          refDomain         // reference to global
domain which created this array
          …...
        }
```

We want our API compliant with c++11 because HPX is strictly conforming to c++11 standard.

global_array::iter() // returns the iterator i.e iterator points to the array stored in the particular locale.

We must provide a suitable interface for accessing the values of the array like copying and slicing etc.



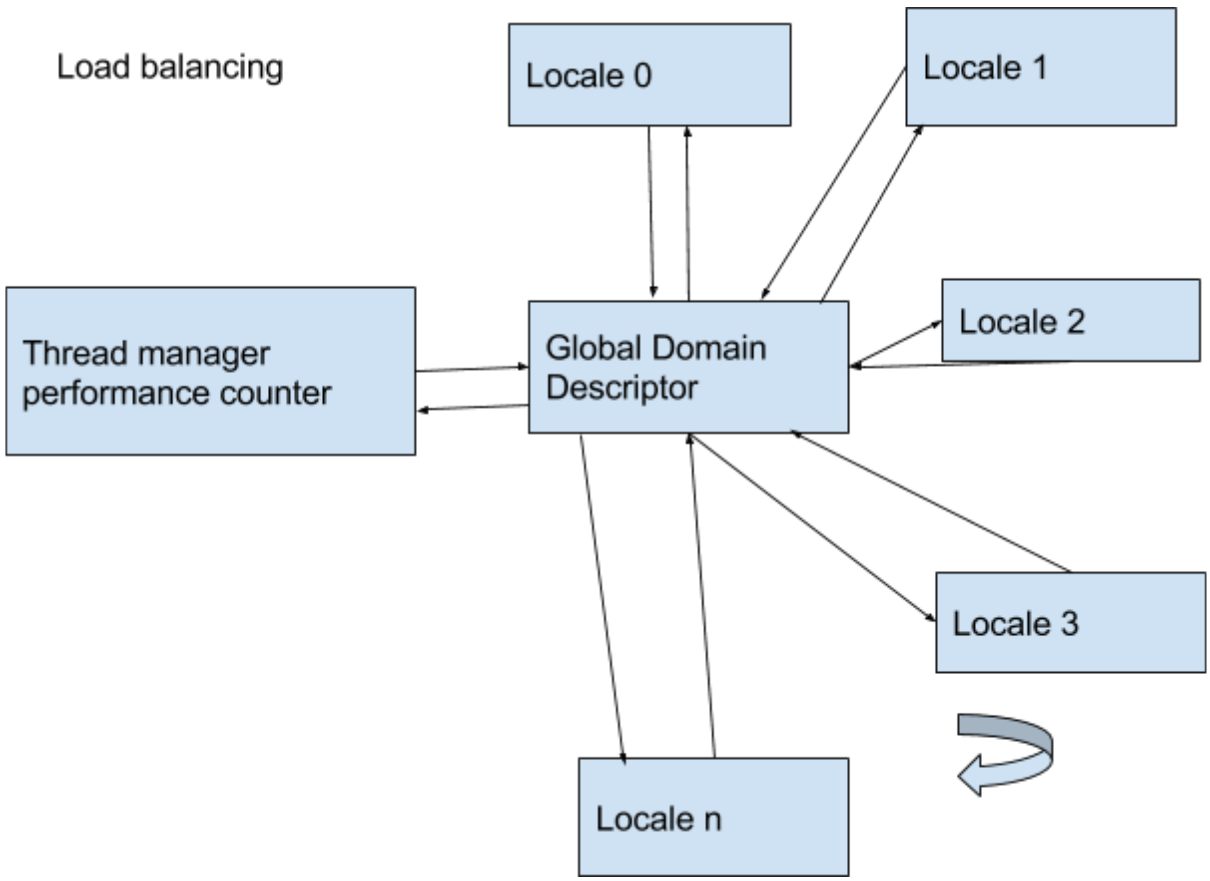

Must supported Functionalities:

- index_type() - returns the type of the index
- rank() - returns the dimension of array, I will also try to design for irregular domains.
- Size() - returns size of the array
- Target_locale() - returns the locale (i.e stored locale)
- Is_empty () - returns true if empty else returns false
- Push_back(),Pop_back() // some preconditions should be met
- Push_front(),Pop_front() // domain non-stridable,non-sharable

- remove(index) - removes element // same precondition
- find(value) - return index
- reshape(domainType) - reshape based on the domain only if no:of:elements are equal.

Additionally functions can be implemented after discussion with mentors.

I would like integrate load balancing with domain maps. I will do the following if possible.




It involves some form Periodic Querying over Locales by the Global Domain Descriptor about the loads on each locales and it can decide how to allocate domains over these locales to achieve better performance. This should be discussed with the mentors.

I currently not sure if this can be done if it is possible I will happy to do so.

Finally,for single domain map distribution we have multiple domains and for each domains we can have multiple arrays.

**Deliverables**

- A good implementation of Global Domain map,Domain,Array descriptor classes.

- Support for above mentioned Domains and their functionalities.

- Testing each implementation with some benchmark algorithms and analyze their performance and rebuilt if not satisfied.

- Standard set of API's over the domain maps,domains.

- A Clear documentation.

**Future Work**

At first I have decided to work for c++ runtime replacement but I cannot understand things at that moment without knowing HPX well and then moved to c++ resumable functions it makes me wondered but later it seems impossible to me to do this implementation. Then I moved to Distributed Component Placement.So after summer I will look for implementing resumable functions (await) I believe I will do it after getting some experience in hpx. As I mentioned above I liked 'Move work to data than data to Work' principle I will work in parcel transport layer.

Work Chart:

I have decided to split 3-months as group of (7 or 8) days.In which 6 or 7 days meant for development and remaining one or two days for analyzing performance , writing blogs and documentation.If time constraint allows comparing performance with OpenMP or MPI and optimizing.

Development includes designing modules and getting approved by mentors and coding with hpx standards.

I will push changes to git every 2 days to avoid loss.

I need mentors to analyze the performance of the code in clusters if possible during review.

| | |
|---|---|
| May 4 - May 23 2017 | Community bonding.<br>Get to know about mentors and their respective work in hpx and getting similar with codebase and start digging deeper about domain maps and domains.<br>***Gain*** : A clear understanding of hpx and domain maps. |
| May 25 - May 31 | Implementation of block distribution policy by implementing distribution class.<br><br>Gain: At start we have one distribution policy forehand |
| June 1-8 | Implementation of block-cyclic distribution policy and writing simple programs based on two distributions and check their correctness and performance |
| June 9-10 | Implementation of Cyclic policy Distribution and integrating the distributions with HPX |

| | |
|---|---|
| June 12 - 19 | 1. Implementation of Arithmetic domains and Sparse domains with their supported functionalities.<br>2. Checking their abilities with simple arrays.<br>3. Optimizing it. |
| June 19 - 26 | 1. Implementing Array descriptor class.<br>2. Making it work well with Arithmetic domains and sparse domains under the distribution policies.<br>3. Adding some API's over the classes and functions. |
| June 26 - 30 | 1. Submitted for first evaluation And review.<br>2. Meantime, Looking to get expertise in different subsystems of hpx and study to integrate with Distributed data structures.<br>3. Implementing some data structures with the help of domains. |
| July 1-8 | 1. Building associative domains and their respective arrays and their functionalities. |

| | |
|---|---|
| July 9 -16 | 1. Implementing useful distributed data structures with the above defined mechanisms and integrating with hpx.<br>2. Start to define Standard set of API's. |
| July 16 - 24 | 1. Implementing a Standard Set of |

| | |
|---|---|
| | API's and integrating to hpx.<br>2.Solving remaining works in all areas if any. |
| July 24 - 28 | 1. Submitted code for second evaluation and review.<br>2. Looking for possibility for extending the set of features over HPX subsystems. |
| July 29 - August 4 | Defining and Implementing allocators over domains and arrays and to support the user defined allocators over the domain maps. |

| | |
|---|---|
| August 5 - 13 | 1. Correcting errors and to know which parts want to optimized by running on clusters if possible.<br>2. Providing tests and use the benchmark algorithms to cross-check the efficiency. |
| August 14 - 21 | 1. Adding all possible requirements and additional functionalities to gain performance<br>2.If time allows implementing another widely used domains such as opaque etc. |

| | |
|---|---|
| August 22 - 28 | Submitting the work to google and providing a good documentation. |

Note : Exams will be in July but i don't know exact dates so I design work chart without mentioning exams.(Exams will cover - 3 days).
If some unhealthy things happen during the time-line and time will be taken from last two weeks. I sure that I will work to my full potential to

complete the project. And I sure that I will not leave the project under any circumstances.

**Potential Risks and Mitigation Strategy**

Weather / Internet connectivity:

Summers is extremely hot and rains towards August it will not disrupt the internet connectivity and my availability.

Laptop crash:

I will work with my DELL inspiron PC throughout the program if any crash happens I have a sparse laptop in home.I commit once every 3 days to my git.

Additional information:

For any reason if I'm not available you can contact my brother


Name : Elangopalakrishnan v

Email : elangopalakrishnan@gmail.com , epkmca@gmail.com

Ph.no : +91 9715228948, +91 9092645724


All information stated above are true to my best of knowledge.

praveenv

REFERENCES:


Authoring user-defined domain maps in chapel Bradford L. Chamberlain,Sung-Eun Choi,Steven J. Deitz- chapel Inc.

SC12-6 domain maps.pdf

Data distribution in hpx thesis by Bibek Ghimire

HPX – A Task Based Programming Model in a Global Address Space -Hartmut Kaiser,Thomas Heller,Bryce Adelstein-Lelbach

HPX -documentation

Chapel -documentation

Active Global Address Space (AGAS):
Global Virtual Memory for Dynamic Asynchronous Many-Tasking (AMT) Runtimes -Abishek Kulkarni