Falcon A Graph Manipulation Language for Heterogeneous systems

Unnikrishnan C, Department of CSA,IISc, email:- unni_c@csa.iisc.ernet.in

Rupesh Nasre, Department of CSE, IIT Madras

Y N Srikant, Department of CSA,IISc

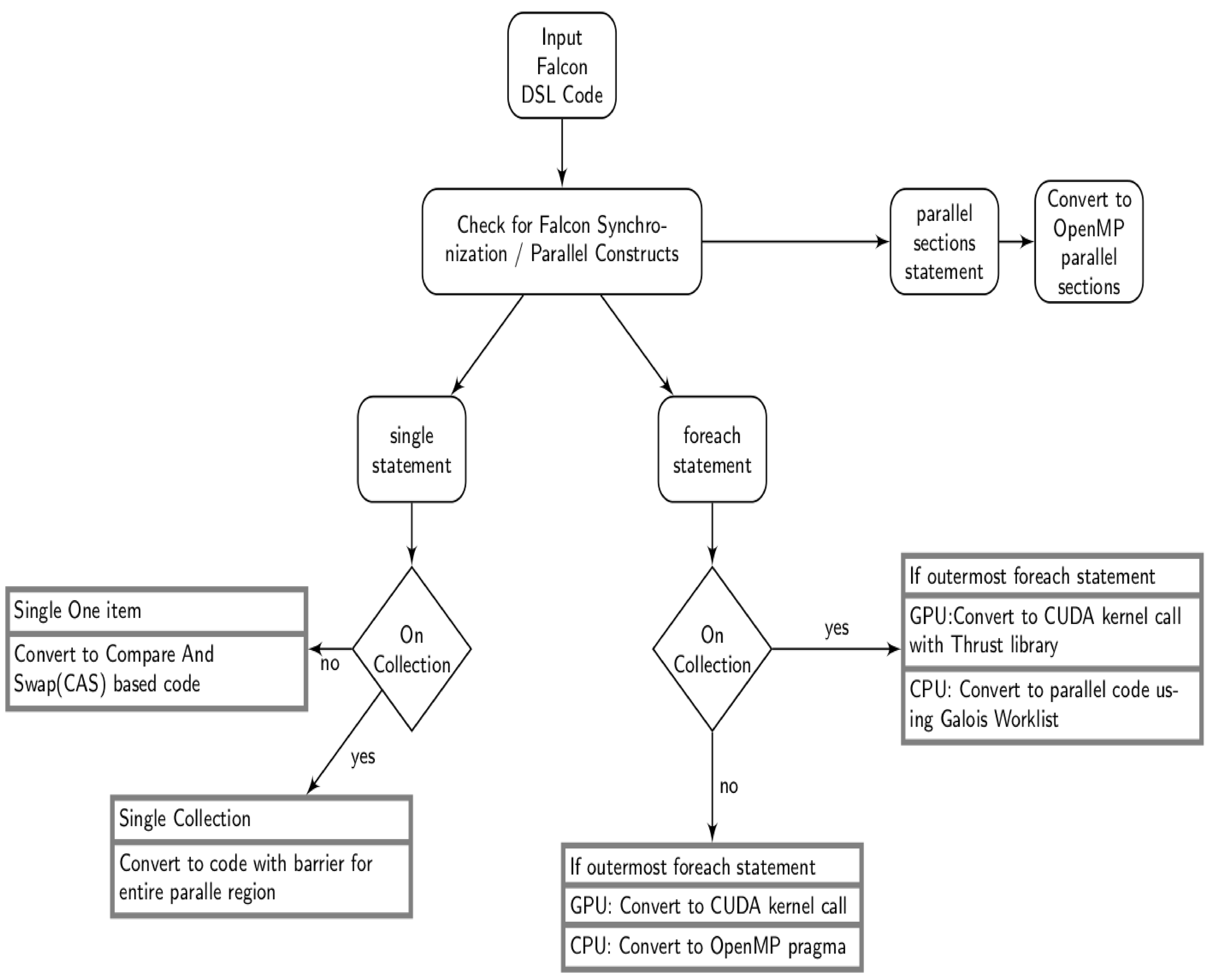
Abstract

Graph algorithms have been shown to possess enough parallelism to keep several computing resources busyeven hundreds of cores on a GPU. Unfortunately, tuning their implementation for efficient execution on a particular hardware configuration of heterogeneous systems consisting of multicore CPUs and GPUs is challenging, time consuming, and error prone. To address these issues, we propose a domain-specific language (DSL),`Falcon`, for implementing graph algorithms that (i) abstracts the hardware, (ii) provides constructs to write explicitly parallel programs at a higher level, and (iii) can work with general algorithms that may change the graph structure (morph algorithms). We illustrate the usage of our DSL to implement local computation algorithms (that do not change the graph structure) and morph algorithms such as Delaunay mesh refinement, survey propagation, and dynamic SSSP on GPU and multicore CPUs. Using a set of benchmark graphs, we illustrate that the generated code performs close to the state-of-the-art hand-tuned implementations.

`Falcon`- Introduction

1. i) extends C programming language.
   ii) provides additional data types for Graph processing.
   iii) constructs for writing explicitly parallel graph algorithms.
2. Support for heterogeneous backends(CPU and GPU).
3. Supports parallel execution of different algorithms on multiple devices.
4. Supports partitioning of Graph objects and execution of a single algorithm using multiple devices. Used when graph object does not fit in a single device. Supports mutation of Graph object.
5. Allows viewing Graph in different way(say collection of triangles).

Compiler Overview



Data Types and Iterators

Data Type	Iterator	Description
Graph	points	iterate over all points in graph
Graph	edges	iterate over all edges in graph
Graph	pptyname	iterate over all elements in new ppty.
Point	nbrs	iterate over all neighboring points
Point	outnbrs	iterate over dst point of outgoing edges (Directed `Graph`)
Edge	nbrs	iterate over neighbor edges
Set	item	iterate over all items in Set
Collection	item	iterate over all items in Collection

Table 1: Iterators for `foreach`(parallelization) statement in `Falcon`

Sample `Falcon` DSL codes

a)Multi-GPU SSSP and BFS

```
int <GPU>changed;
SSSPBFS(char *name) { //begin SSSPBFS
Graph hgraph;//Graph object on CPU
hgraph.addPointProperty(dist,int);
hgraph.addProperty(changed,int);
hgraph.getType() <GPU>graph0;//Graph on GPU0
hgraph.getType() <GPU>graph1;//Graph on GPU1
hgraph.addPointProperty(dist1,int);
hgraph.read(name);//read Graph from file to CPU
graph0=hgraph;//copy entire Graph to GPU0
graph1=hgraph;//copy entire Graph to GPU1
foreach(t In graph0.points)t.dist=1234567890;
foreach(t In graph1.points)t.dist=1234567890;
graph0.points[0].dist=0;
graph1.points[0].dist=0;
parallel sections { //do in parallel
  section {//compute BFS on GPU1
    while(1){
      graph1.changed[0]=0;
      foreach(t In graph1.points)BFS(t,graph1);
      if(graph1.changed[0]==0) break;
    }
  }//end section BFS
  section {//compute SSSP on GPU0
    while(1){
      graph0.changed[0]=0;
      foreach(t In graph0.points)SSSP(t,graph0);
      if(graph0.changed[0]==0) break;
    }
  }//end Section SSSP
}//end Sections
}//end SSSPBFS
```

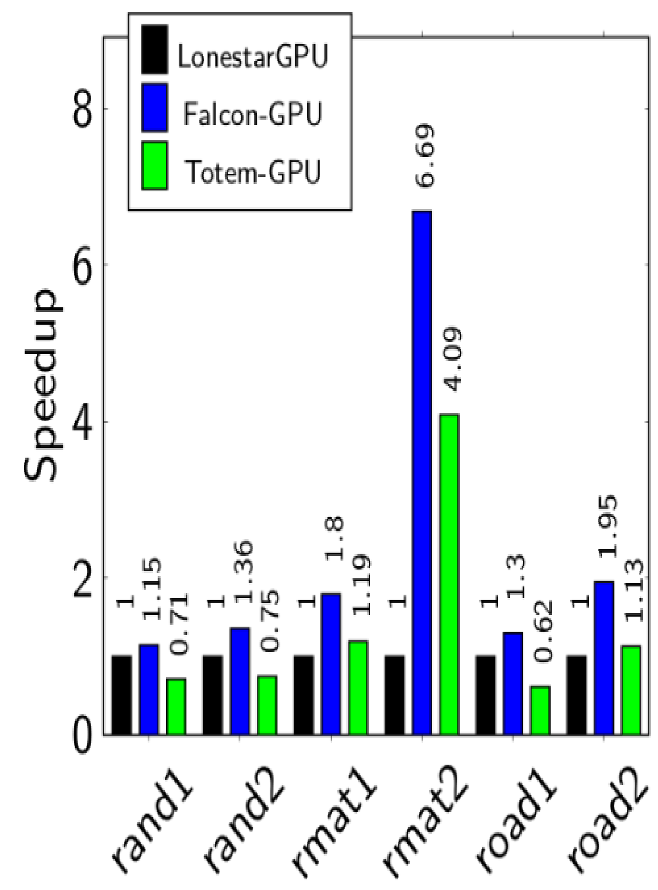
b) Heterogeneous Execution(CPU and GPUs)

```
fun1(Point ori, Point incom){
  if(orig.dist >incom.dist)
    orig.dist=incom.dist
}
relaxgraph(Point p, HGraph hgraph){
  foreach(t in p.outnbrs)
    MIN(t.dist,p.dist+hgraph.getWeight(p,t),
  hgraph.changed[0]);
}
main(int argc, char *argv[]){
HGraph hgraph;
hgraph.addPointProperty(dist, int);
  hgraph.addProperty(changed, int);
hgraph.read(argv[1]);
  hgraph.makePartition(1,1,ORDERED);
  hgraph.updateFunction(fun1);
foreach(t In hgraph.points) t.dist=1234567890;
hgraph.points[0].dist=0;
while( 1 ){
    hgraph.changed[0]=0;
    foreach(t In hgraph.points)relaxgraph(t,hgraph);
    if(hgraph.changed[0]==0)break;
    hgraph.updatePartition();
}
for(int i = 0;i <hgraph.npoints; i++)
  printf("%d", hgraph.points[i].dist); }//end main
```

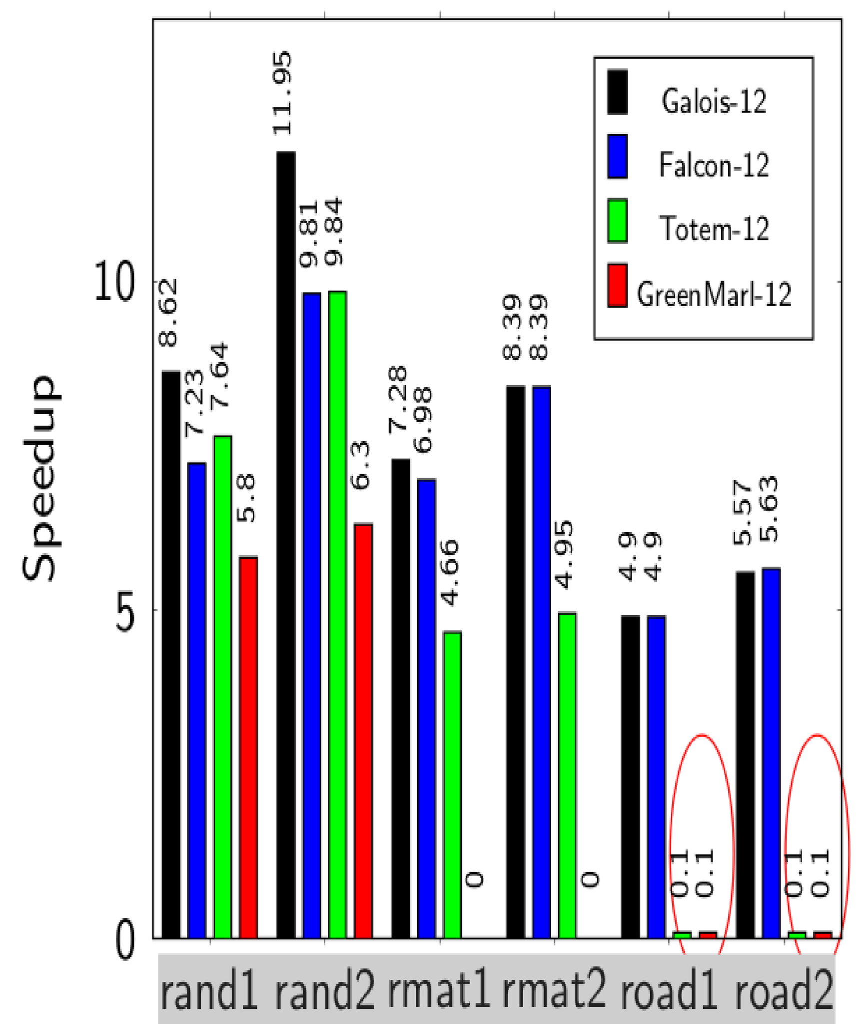
Results

1. Using `Falcon` compiler we wrote algorithms like BFS, SSSP and Boruvka's-MST.
2. We wrote dynamic algorithms like Survey Propagation(SP), Delaunay Mesh refinement(DMR) and Dynamic-SSSP in `Falcon`.
3. Performance of `Falcon` codes were compared with
   i) **LonestarGPU**-(ISS group at the University of Texas at Austin)
   ii) **Galois**- (ISS group at the University of Texas at Austin)
   iii)**Green-Marl**- DSL (PP Laboratory, Stanford University)
   iv)**Totem**(NetSysLab, University of British Columbia)
4. **Totem**- for comparing Performance on CPU, GPU and heterogeneous execution.
5. **Galois** and **Green-Marl** for comparing Performance on CPU.
6. **LonestarGPU** for comparing Performance on GPU.
7. We were able to get performance close to and some times better than above systems.
8. Tested on 12-core CPU and 4-GPU machine(1 Kepler and 3 Tesla).

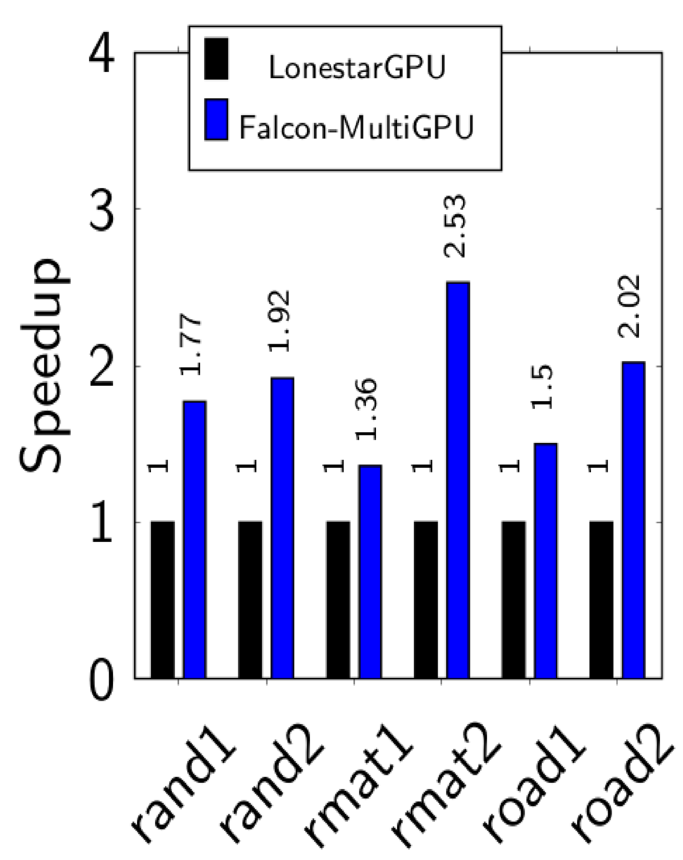
(Points,Edges) rand1(16M,64M),rand2(32M,128M)
rmat1(10M,100M),rmat2(20M,200M)
road1(14M,34M) road2(23M,58M)



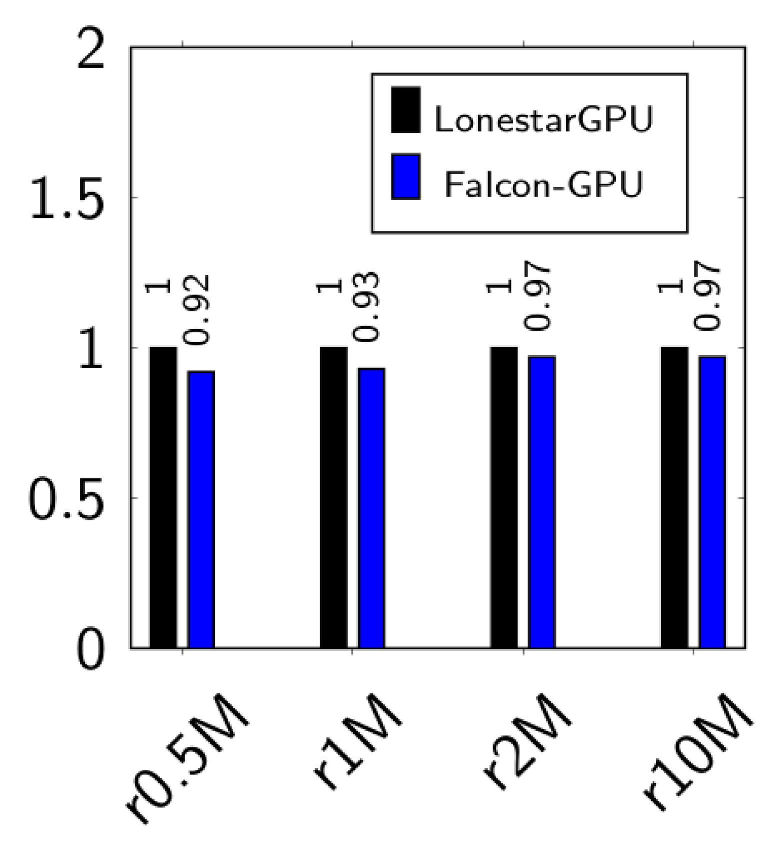
a) Speedup of SSSP over LonestarGPU



b) Speedup of SSSP over Galois single



c) `Falcon` multi-GPU speedup



d)DMR speedup over LonestarGPU

Future works

- OpenCL extension and Support for CPU and GPU cluster.
- Automatic partitioning and removal of <GPU>tag.
- For queries email to unni_c@csa.iisc.ernet.in.

Publications

[1] Unnikrishnan C , Rupesh Nasre and Y N Srikant. `Falcon`: A Graph Manipulation Language for Heterogeneous Systems. ACM TACO, December 2015.