Green-Marl: A DSL for Easy and Efficient Graph Analysis

Hong, Chafi, Sedlar and Olukotun

Reviewed by Neil Satra (ns532)
OpenMP implementation

```c
#include "graphdefs.h"
#include "graph_metrics.h"
#include "utils.h"
#include "sprng.h"

void vertex_betweenness_centrality_parBFS(graph_t* G, double* BC, long numSrscs) {
    attr_id_t *S; /* stack of vertices in the order of non-decreasing 
                    distance from s. Also used to implicitly 
                    represent the BFS queue */
    plist_t* P; /* predecessors of a vertex v on shortest paths from s */
    double* sig; /* No. of shortest paths */
    attr_id_t* d; /* Length of the shortest path between every pair */
    double* del; /* dependency of vertices */
    attr_id_t* in_degree, *numEdges, *pSums;
    attr_id_t* pListMem;
    #if RANDSRCS
    attr_id_t* Srscs;
    #endif
    attr_id_t* start, *end;
    long MAX_NUM_PHASES;
```
Procedure Compute_BC(
    G: Graph, BC: Node_Prop<Float>(G)) {
    G.BC = 0;               // initialize BC
    For each (s: G.Nodes) {
        // define temporary properties
        Node_Prop<Float>(G) Sigma;
        Node_Prop<Float>(G) Delta;
        s.Sigma = 1;           // Initialize Sigma for root
        // Traverse graph in BFS-order from s
        InBFS(v: G.Nodes From s) (v!=s) {
            // sum over BFS-parents
            v.Sigma = Sum(w: v.UpNbrs) {w.Sigma};
        }
        // Traverse graph in reverse BFS-order
        InRBFS(v!=s) {
            // sum over BFS-children
            v.Delta = Sum (w:v.DownNbrs) {
                v.Sigma / w.Sigma * (1+ w.Delta)
            }
        }
        v.BC += v.Delta @s;     // accumulate BC
    }
}
Green-Marl needs way fewer Lines of Code
Green-Marl is a Domain Specific Language

For

Graph analysis algorithms

With

Intuitive high-level constructs

Which

Expose data-level parallelism inherent in the algorithm
High level constructs

- Graphs, nodes, edges
- Neighbours (in, out, up and down)
- Breadth-First and Depth-first search
Parsing and Checking

Target Independent Optimisation

Target Dependent Optimisation

Code Generation

In goes Green-Marl code

Out comes C++/OpenMP code
Objection: Performance

(a) RMAT

(b) Uniform
Objection: New Language

- Can interleave with C++ code
- Tutorial on Github
- Detailed language specs available online
Objection: Adoption

- Production ready – actively maintained on Github
- Built-in support for Giraph (in sequel to this paper)
In goes Green-Marl code

Out comes C++/OpenMP code
Objection: Adoption

- Production ready – actively maintained on Github
- Built-in support for Giraph (in sequel to this paper)
- Oracle adoption in their graph analytic framework, Oracle PGX
- No lock in
Advantages

- Easier to write graph algorithms*
- Algorithms perform better
- Don’t need to rewrite entire application
- Code is portable across platforms
Well Evaluated

- Tested on Random and Power-law graphs
- Individual optimisations tested
Weakness

Graph is immutable during the analysis
Summary

- Write graph analysis portion of software in Green-Marl

- Get human-readable output in target language

- With automatic optimisations

Reviewed by Neil Satra
ns532@cam.ac.uk