Vertex-centric Graph Processing on FPGA

Nina Engelhardt, Hayden Kwok-Hay So

Department of Electrical and Electronic Engineering
The University of Hong Kong

Abstract

- Past work has shown that FPGAs are efficient at processing many graph algorithms, but they are notoriously hard to program
- We propose a vertex-centric framework for graph processing on FPGAs, providing a base execution model and distributed architecture so that developers need only write very small application kernels.

Programming Model

We adopt a Pregel[1]-like vertex-centric programming model. Users specify a *vertex kernel* that is executed in parallel for each vertex in the graph. Vertices can exchange messages only along edges, and can access only private local data. To enable optimization of data storage in our system, we ask users to split their implementation in two parts, based on availability of data:

- *Apply* is called for each message that a vertex receives. It may access and modify the vertex' local data, and optionally produce an update to be broadcast to the vertex' neighbors.
- For each update, *Scatter* is called on each outgoing edge of the vertex. It can access the edge's data to finalize the message to be sent to each neighbor.

Example Code: PageRank

Apply(vertex, message):

```
vertex.sum += message.weight
vertex.nrecvd += 1
if vertex.nrecvd == vertex.indegree and superstep < 30 then
    update(sender=vertex.id,
    weight=0.15 / num_vertices + 0.85 * vertex.sum)
    vertex.sum = 0
    vertex.nrecvd = 0
end if
return vertex
```

Scatter(update, edge):

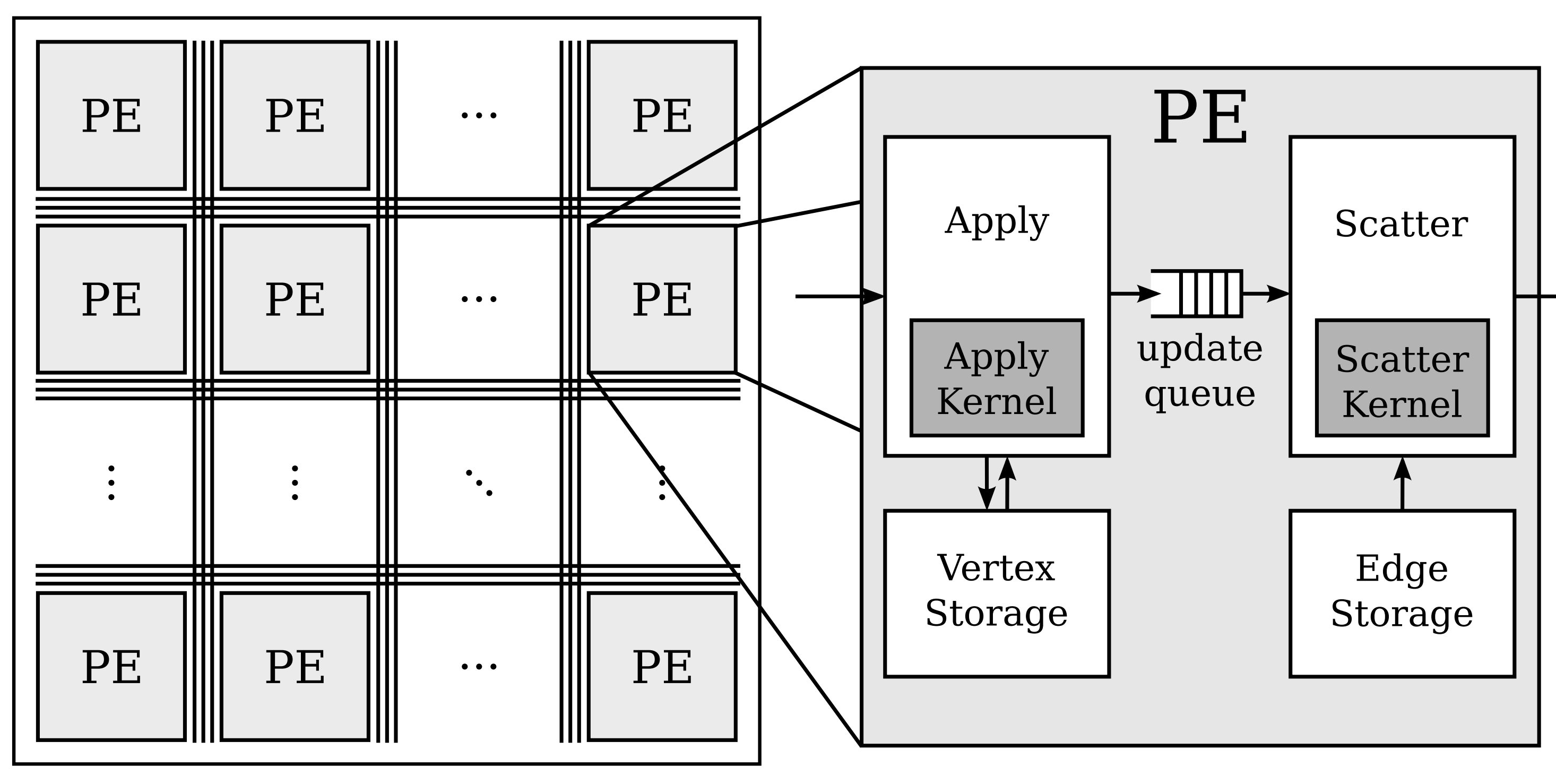
```
message(sender=update.sender, destination=edge.target,
weight=update.weight/sender.outdegree)
```

Floating Barrier

- Synchronization by Floating Barrier [2]: 2 supersteps concurrently in execution, improving load balance
- Apply-Scatter split + Floating Barrier allows moving storage to the middle of the pipeline, where data is most condensed – reducing the storage requirements to guarantee deadlock-free execution from $|V|^2$ to $2\,|V|$

Architecture

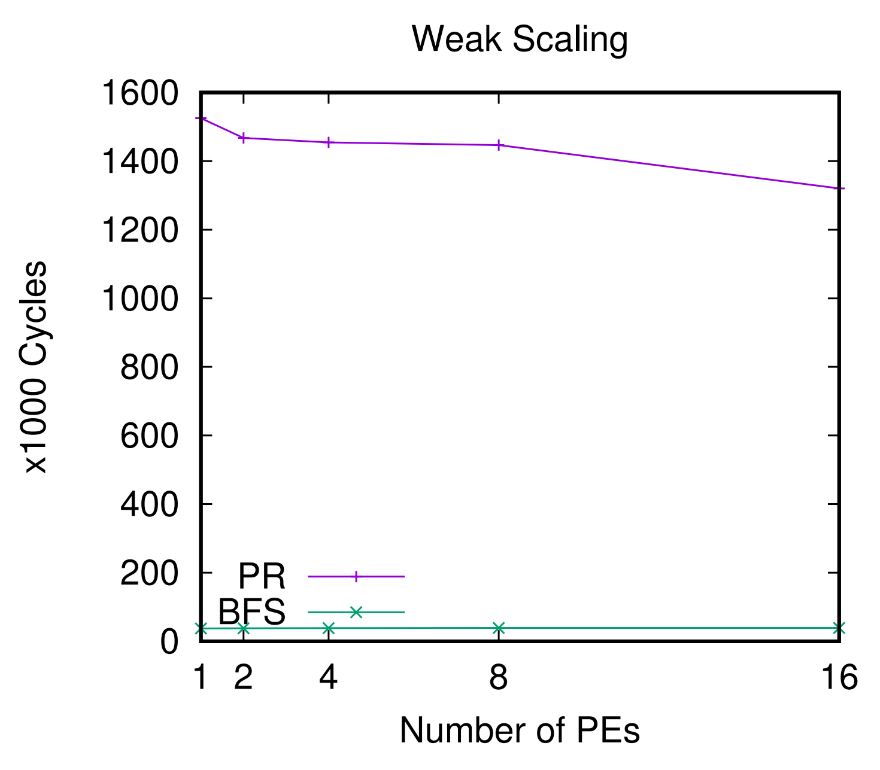
Architecture Overview

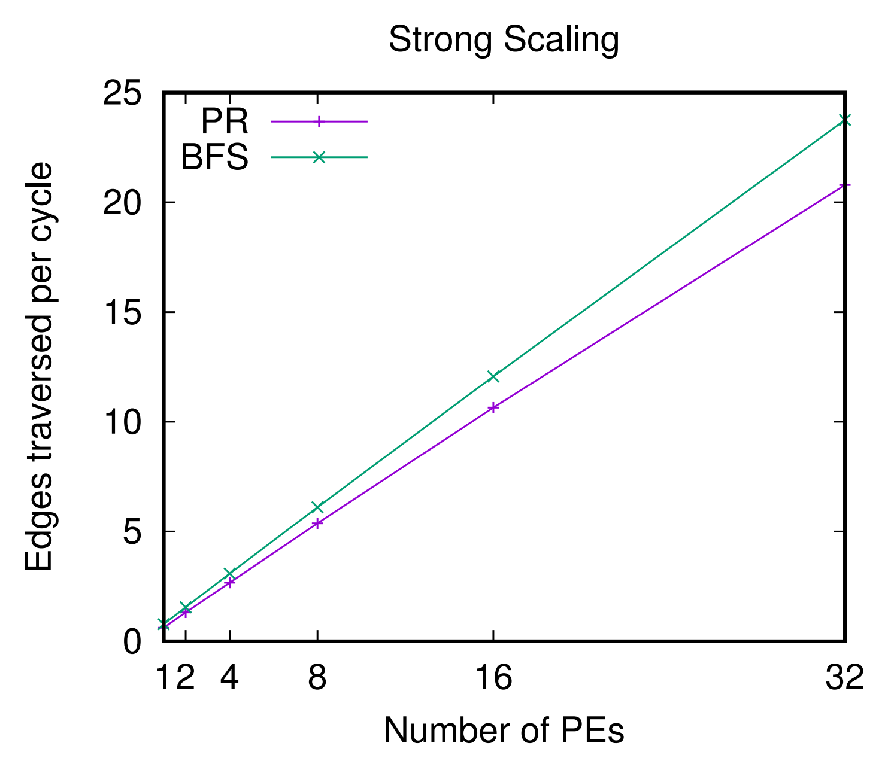


- Network of identical processing elements (PEs)
- PE and network architecture are fixed, only user-provided Apply and Scatter kernels (shown shaded) switched out for each algorithm

Results

Weak scaling – 8192 vertices per PE, graph size grows: Slightly improving performance with larger size due to additional buffer resources





Strong scaling – Fixed size graph of 128k vertices divided over growing number of PEs: Linear scaling

Conclusion

An FPGA-based vertex-centric framework for graph processing can improve development speed to the point where even casual users may take advantage of the platform's massive parallelism and energy efficiency.

Bibliography

[1] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. ACM, 2010, pp. 135–146.

[2] Q. Wang, W. Jiang, Y. Xia, and V. Prasanna, "A message-passing multi-softcore architecture on FPGA for breadth-first search," in *Field-Programmable Technology (FPT), 2010 International Conference on*, Dec 2010, pp. 70–77.

LaTeX TikZposter