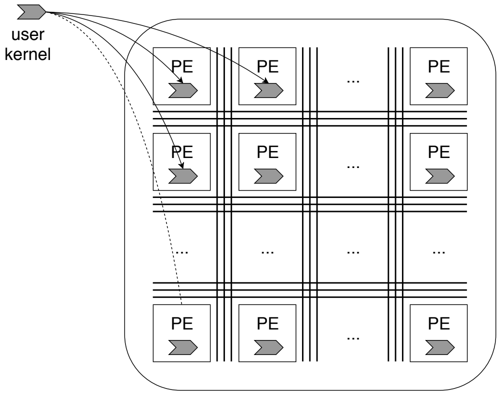
Performance-driven system generation for distributed vertex-centric graph processing on multi-FPGA systems

Nina Engelhardt
C.-H. Dominic Hung
Hayden K.-H. So

The University of Hong Kong

28th August 2018

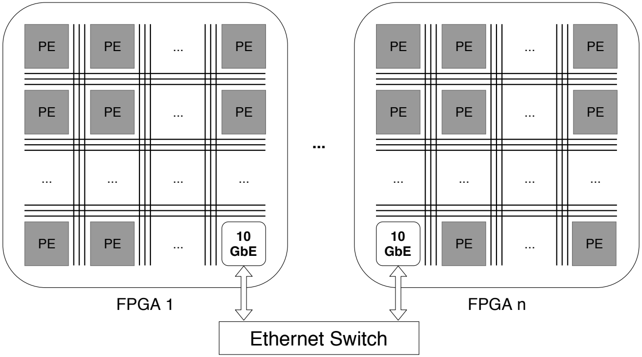
The GraVF graph processing framework



- Vertex-centric graph processing framework on FPGA
- User provides kernel, inserted in framework architecture
- PEs exchanging messages over on-chip network

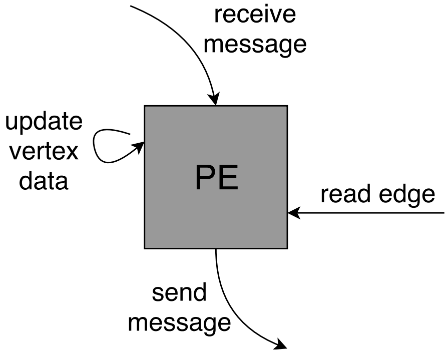
Now extended to multiple FPGA

- Allow PEs to exchange messages with PEs on different FPGAs
- Extend network by adding external interface and routing messages destined for PEs on other FPGAs over 10GbE



What's the performance?

- Vertex kernel has well-defined interface
- Can calculate the necessary resources to process one edge



- Build roofline-style performance model based on platform resources
- Use model to automatically pick configuration when generating

Limiting factors

4 limits considered:

- Processing element throughput
- Memory bandwidth
- Network interface bandwidth
- Total network bandwidth

Processing element throughput

$$T_{sys} \leq n_{FPGA} \times n_{PE/FPGA} \times f_{clk} / CPE_{PE} \qquad (L_{PE})$$

- Cycles per edge: analogous to processor CPI, used together with clock frequency to determine individual PE throughput
- CPE is affected by:
  - PE architecture
  - data hazards
  - kernel implementation
- multiplied by number of PEs in the system.

Memory bandwidth

$$T_{sys} \leq n_{FPGA} \times \frac{BW_{mem}}{m_{edge}} \qquad (L_{mem})$$

- Edges can be stored off-chip to increase processable graph size
- Can only be processed as fast as they can be loaded

Network interface bandwidth

$$T_{sys} \leq n_{FPGA} \times \frac{BW_{if}}{2\frac{n_{FPGA}-1}{n_{FPGA}} m_{message}} \qquad (L_{if})$$

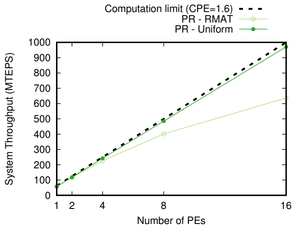
- When using multiple FPGAs, messages need to be transferred over external network interface
- Assuming equal distribution of vertices, a message has a $\frac{n_{FPGA}-1}{n_{FPGA}}$ chance to be sent to a different board
- Each message traverses an interface twice, sending and receiving
- Really a per-board limit, extra factor n_{FPGA} on both sides for system throughput

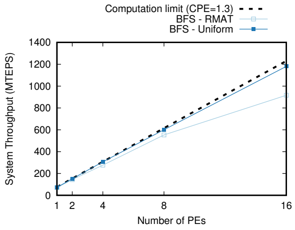
Total network bandwidth

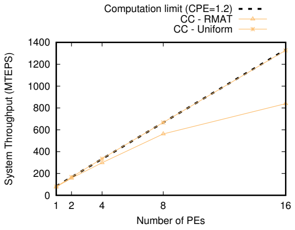
$$T_{sys} \leq \frac{BW_{network}}{\frac{n_{FPGA}-1}{n_{FPGA}} m_{message}} \qquad (L_{network})$$

- Total amount of messages transferrable by the external network
- Again, a fraction $\frac{n_{FPGA}-1}{n_{FPGA}}$ of messages needs to cross the external network

Results

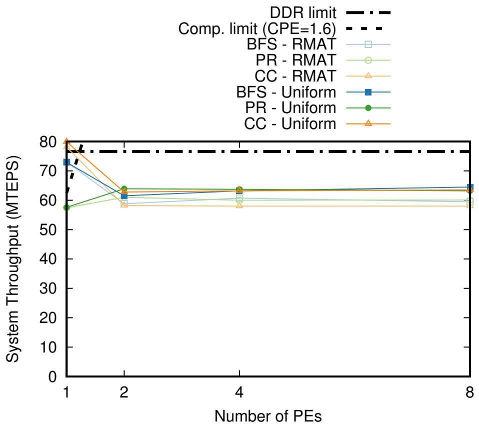






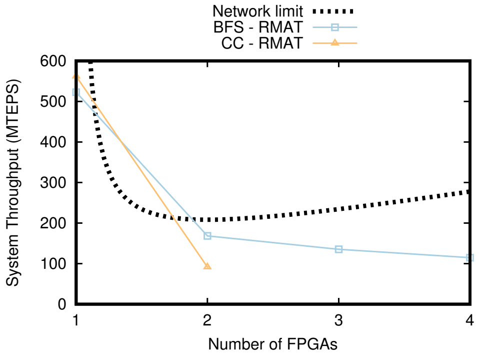
- everything on-chip: no memory, no network
- PE throughput is limiting
- close to limit for uniform graphs, slowdown due to imbalance for RMAT graphs

Results



- using external memory, but only one FPGA
- Xilinx MIG DDR3 controller's $BW_{mem} = 2.5$ Gbps random access performance is limiting
- better performance at 1 PE as accesses are more sequential

Results



- using 4 FPGAs, no memory
- network interface bandwidth $BW_{network} = 6.7$ Gbps is limiting
- imbalance has greater impact, further decays performance

Conclusion

- Graph algorithms are very communication-intensive
- need to optimize interfaces
- can predict performance reasonably accurately
- except for imbalance (depends on input properties)

Thank you for listening!

Questions? Visit poster board 8!