

A Preliminary Performance Model for Optimizing Software Packet Processing Pipelines

Ankit Bhardwaj, Atul Shree, Bhargav Reddy V, and Sorav Bansal

Department of Computer Science and Engineering, Indian Institute of Technology Delhi

Objectives and Motivation

Goal: Develop a compiler that can automatically map a high-level specification to an underlying machine architecture in an optimal way.

Motivation:

- Increasing popularity of Software Defined Networks(SDN).
- SManual Optimizations are time-consuming and error prone.
- DSLs are not very helpful until we develop good compilers.

Compilation Phases

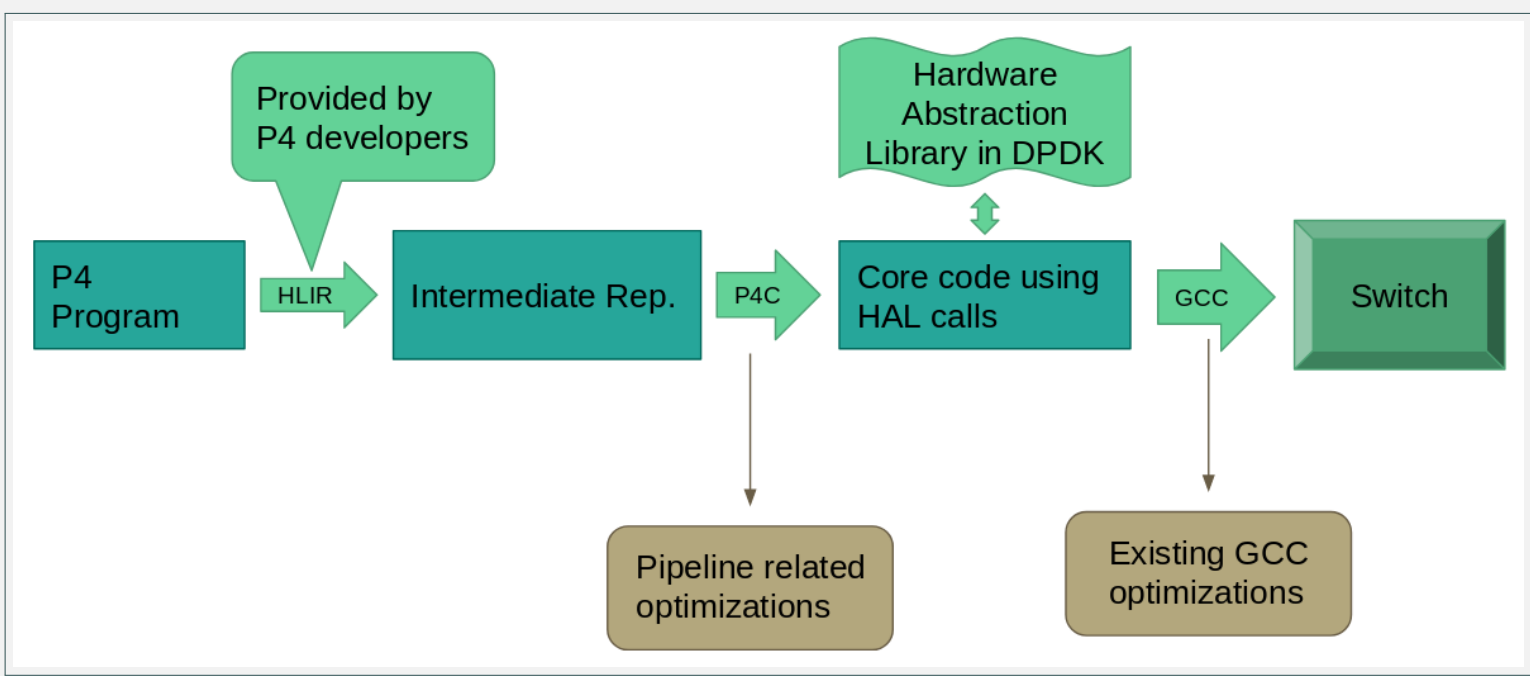


Figure: Compilation phases

Packet Processing Pipeline

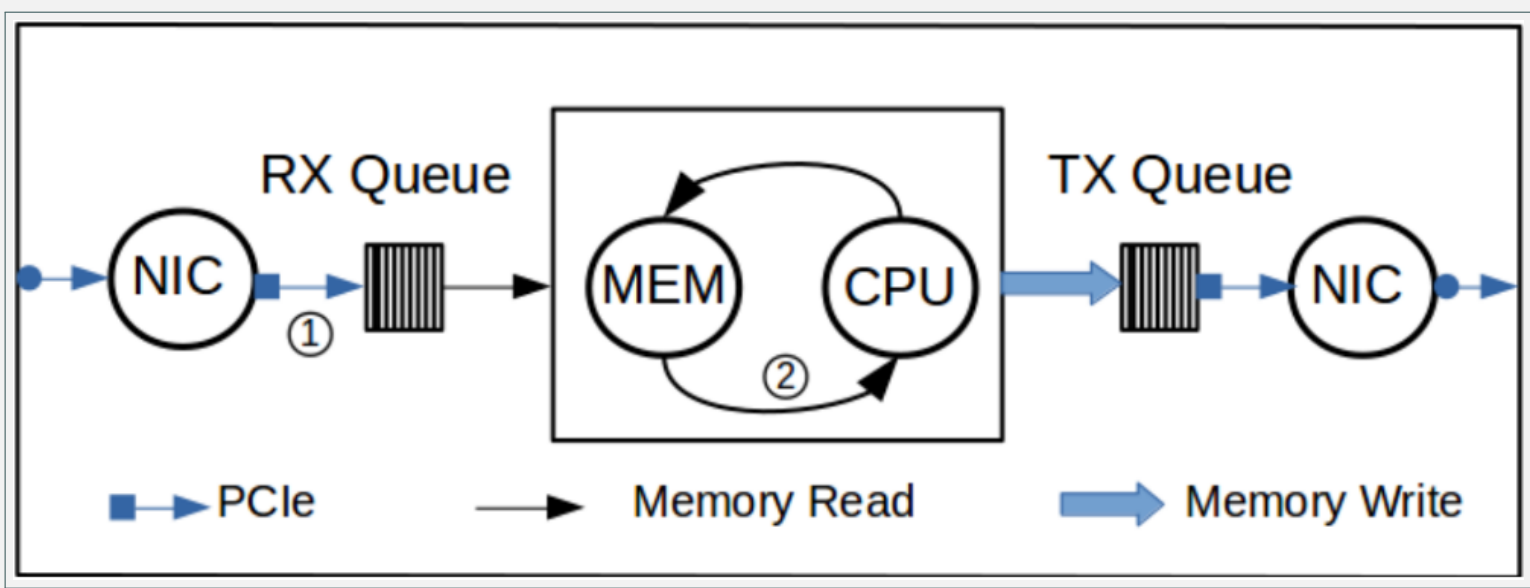


Figure: Packet Processing Pipeline

- * Exploit DMA bandwidth between NIC and Main Memory labeled ① in Figure
- * Exploit Memory Level Parallelism between CPU and Memory labeled ② in Figure

Optimizations

```
sub app {
  for (i = 0; i < B; i++) {
    p = read_from_input_NIC();
    p = process_packet(p);
    write_to_output_NIC(p);
  }
}
sub app {
  for (i = 0; i < B; i++)
    p[i] = read_from_input_NIC();
  for (i = 0; i < B; i++)
    p[i] = process_packet(p[i]);
  for (i = 0; i < B; i++)
    write_to_output_NIC(p[i]);
}
```

```
sub process_packet(p) {
  for (i = 0; i < B; i++) {
    t1 = lookup_table1(p[i]);
    t2 = lookup_table2(p[i], t1);
    ...
  }
}
sub process_packet(p) {
  for (i = 0; i < B; i+=b) {
    for (j = i; j < i+b; j++)
      t1[j-i] = lookup_table1(p[j]);
    for (j = i; j < i+b; j++)
      t1[j-i] = lookup_table2(p[j], t1[j-i]);
    ...
  }
}
```

Performance Model

IO Subsystem, Memory Subsystem and CPU are working in a pipeline fashion. Let the service rate of CPU, CPU-Memory, I/O-Memory DMA interface be c, m, d .

$$Throughput = \min(c, m, d)$$

The value of b will vary with change in the parameters related to memory $f_{mem}(m, b)$. DMA interface is not linearly scalable and rate increases based on some function $f_{dma}(d, B)$.

$$Throughput = \min(c, f_{mem}(m, b), f_{dma}(d, B))$$

Experiments and Results

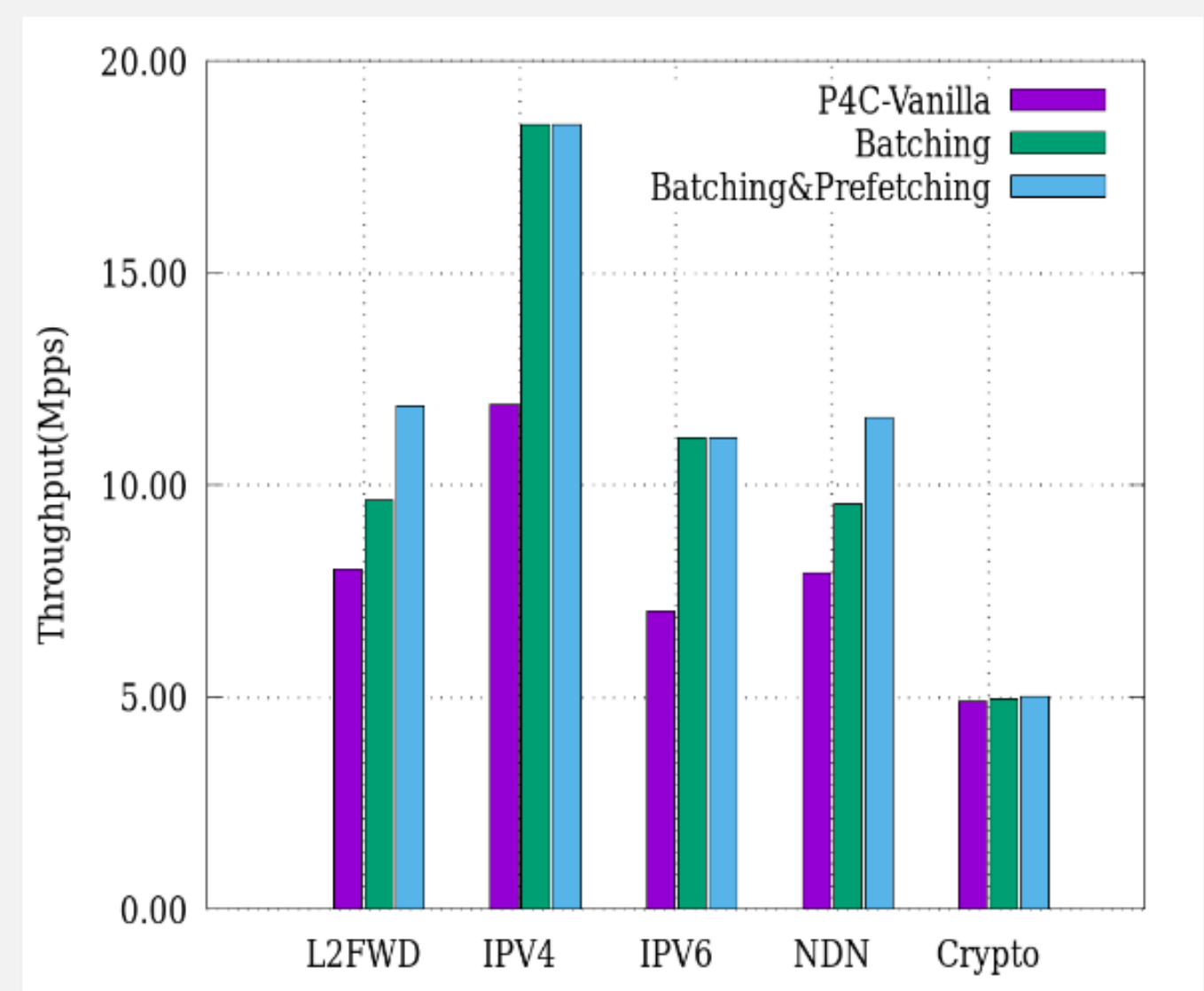


Figure: Effect of Batching and Prefetching

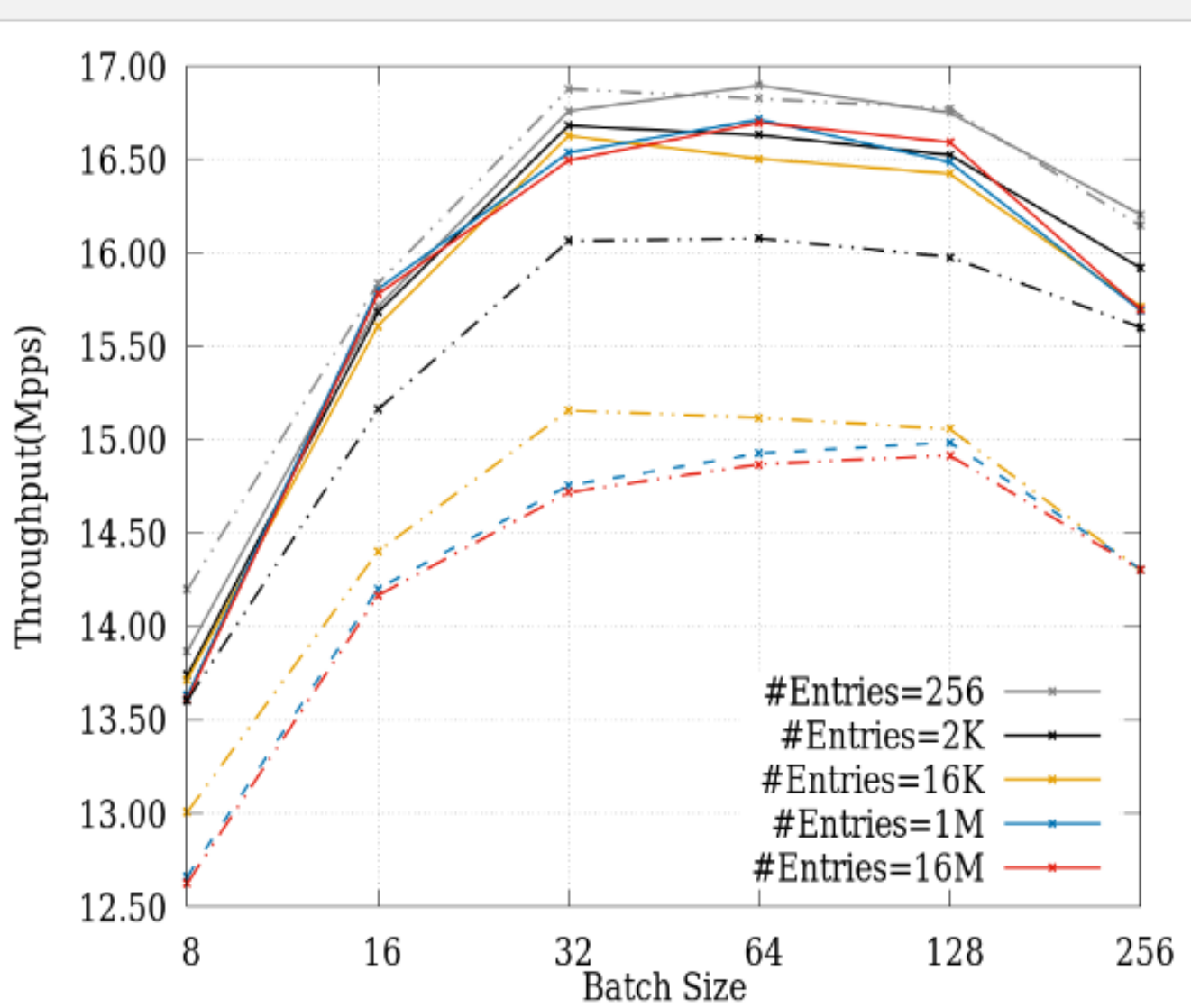


Figure: Sensitivity of Throughput to B

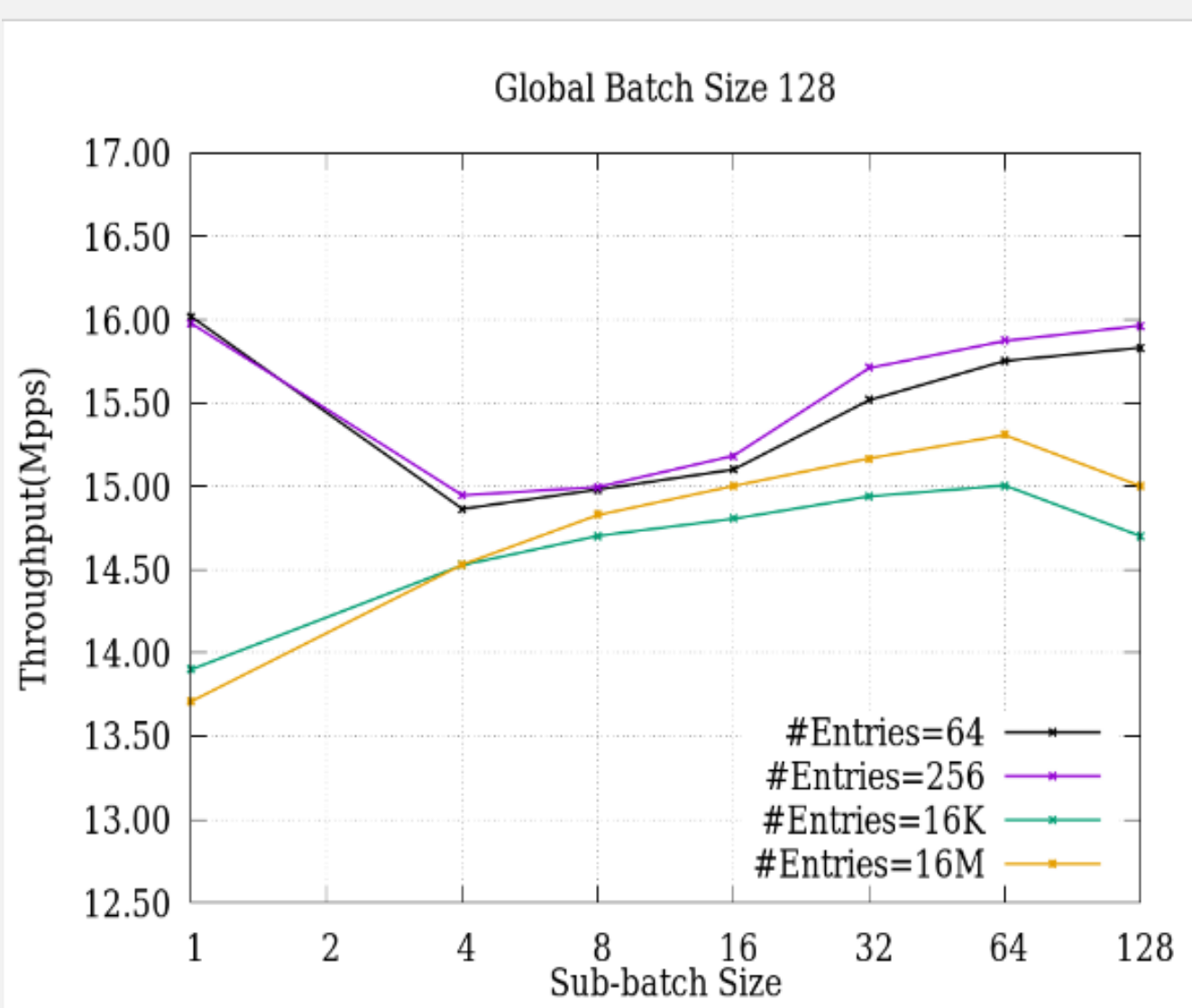


Figure: Sensitivity of Throughput to b

Conclusion

We are using Batching(B) to exploit available parallelism between NIC and Memory interface, and prefetching to exploit the memory level parallelism between CPU and Memory interface. Sub-Batching(b) is being used to adjust the prefetch distance to take full advantage from software prefetching. The model to get the optimal value of b and B is preliminary and not fully automatic. We are running the applications to get the value of b and B and feeding the compiler to generate the optimal code. These optimizations can be used for wide variety of network and streaming applications.

Future Work

- Get more understanding of NIC-Memory interface and CPU-Memory interface.
- Make the model robust to be used for various kind of applications.
- Use the model for various architecture and do the appropriate changes.

References

- [1] Intel Data Plane Development Kit. <http://dpdk.org/>.
- [2] P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

