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

Motivation and Objectives

- Newer functionalities are being developed over time with increasing popularity of SDN.
- Researchers are developing DSLs to make the task easier. However, ease of programming doesn't guarantee application performance.

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

Compilation Phases

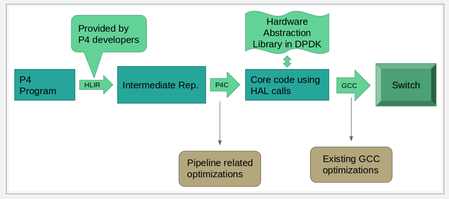

Figure: Compilation phases

Applications written in P4[1] are given as the input to P4C[2] compiler and the compiler generated DPDK[3] based Applications. DPDK[3] application is compiled using gcc to generate the target binary.

Packet Processing Pipeline

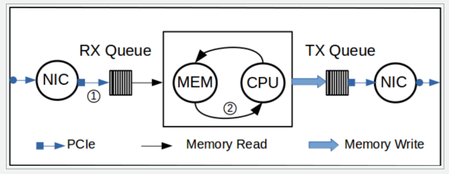

Figure: Packet Processing Pipeline

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

Compiler Transformations related to Batching, Sub-Batching and Prefetching

```
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]);
}
```

Loop Fission[4] Transformation for Batching

```
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]);
        ...
    }   }
```

Loop Fission[4] Transformation for Sub-Batching

```
sub     hash_lookup() {
        for( i=0; i<B; i++){
            key_hash[i] = hash_compute(key[i]);
            prefetch(bucket(key-hash[i]));
        }
        for( i=0; i < B; i++){
            val[j] = hash_lookup(key_hash[j]);
}       }
```

↑↓

```
sub hash_lookup() {
        for( i = 0; i < B; i += b){
            for( j = i; j < i + b; j++ ){
                key_hash[i] = hash_compute(key[i]);
                prefetch(bucket(key-hash[i]));
            }
            for( j = i; j < i + b; j++ ){
                val[j] = hash_lookup(key_hash[j]);
}       }   }
```

Use of Sub-Batching for Prefetching

Performance Model

- Let the service rate of CPU, CPU-Memory, I/O-Memory interface be **c, m, d**.
- **Throughput** of the application will be **min(c, m, d)**.
- The value of **m** will vary with change in **b** and will follow some $f_{mem}(m, b)$.
- DMA interface is not linearly scalable and **d** increases based on some function $f_{dma}(d, B)$.
- $Throughput = min(c, f_{mem}(m, b), f_{dma})(d, B))$

Experiments and Results

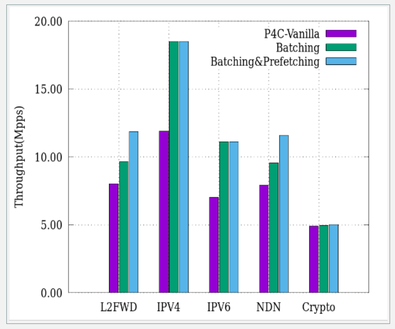

Figure: Effect of Batching and Prefetching

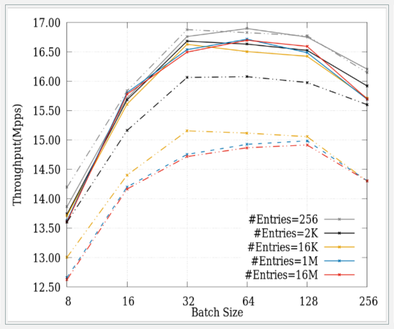

Figure: Sensitivity of Throughput to B. Solid line b = B and dotted lines b = 1.

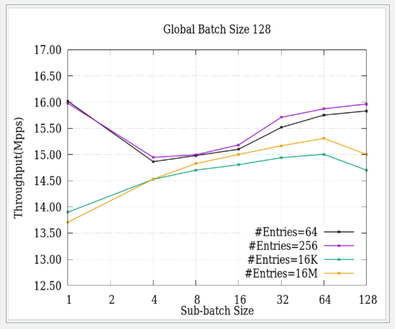

Figure: Sensitivity of Throughput to b. Batch Size B = 128 for these experiments.

Discussion

- Use of Batching(**B**) to exploit NIC-Memory Parallelism.
- Use of Prefetching to exploit MLP at Memory-CPU interface in the pipeline.
- Use of sub-batch(**b**) to adjust the prefetch distance.
- Run the applications to find optimal value of **b** and **B**.
- Input **b** and **B** to the compiler to generate optimized DPDK application.

Future Work

- Perform more fine-grained experiments to gain better understanding of underlying hardware.
- Explore optimizations other than scheduling and select the ones best suited to an application.

References

[1] P4: Programming protocol-independent packet processors.
SIGCOMM Comput. Commun. Rev., 44(3), July 2014.

[2] High speed packet forwarding compiled from protocol independent data plane specifications.
SIGCOMM '16.

[3] *Intel Data Plane Development Kit*.
`http://dpdk.org/`.

[4] *Optimizing supercompilers for supercomputers*.
PhD thesis, Univ. of Illinois,Urbana, IL, Jan 1982.

IIT DELHI