ESE 532 Analysis Milestone 2

Sheil Sarda, Anthony Stewart, Shaokang Xia

**1. Identify major design space axes that could be explored for your implementation.**

CDC:

|  |  |
| --- | --- |
| **Axis** | Throughput; window size; parallelism; data size; II; memory |
| **Challenge** | Improving the throughput of CDC; applying pipelining strategy |
| **Opportunity** | Generating the chunks at multiple starting points |
| **Continuum** | Range from 1 to *Input\_size* *-Windows\_size* windows |
| **Equation for Benefit** | Running *N* match-ups in parallel: *T(N)=T(1)/N* |
| **Equation for Resources** | Resources(*R*)= *R\*single\_chunk\_resource* |

SHA:

|  |  |
| --- | --- |
| **Axis** | Throughput; chunk size; number of chunks; input size; II; memory |
| **Challenge** | Improving the throughput; optimizing the II |
| **Opportunity** | Parallelism with LZW; pipelining; multiple-threads parallelism |
| **Continuum** | Range from 1 to *number\_of\_chunks* hashes |
| **Equation for Benefit** | *N* threads parallelism: *T(N)=T(1)/N;*  Parallelism with LZW: *T=max(T\_SHA, T\_LZW)* |
| **Equation for Resources** | Resource(*R*)=*R\*single\_threshold\_resource* |

Deduplication:

|  |  |
| --- | --- |
| **Axis** | Throughput; input size; memory; chunk size; hashmap; hashes |
| **Challenge** | Improving the throughput; dependency on SHA256; |
| **Opportunity** | Parallelism |
| **Continuum** | Range from 1 to *number\_of\_chunks* hashes |
| **Equation for Benefit** | Running *N* storing in parallel: *T(N)=T(1)/N* |
| **Equation for Resources** | Resources(*R*)= *R\*single\_storage\_resource* |

LZW:

|  |  |
| --- | --- |
| **Axis** | Throughput; input size; memory; pointers; encoding; II |
| **Challenge** | Improving the throughput; optimizing the II |
| **Opportunity** | Parallelism with SHA; apply parallelism in LZW; pipelining |
| **Continuum** | Range from 1 to input\_size comparisons and memory lookup |
| **Equation for Benefit** | Running *N* comparison and lookup in parallel: *T(N)=T(1)/N* |
| **Equation for Resources** | Resources(*R*)= *R\*(single\_comparison\_resource+single\_lookup\_resource)* |

Communication/integration:

|  |  |
| --- | --- |
| **Axis** | Data rate among operations; input size; memory; hashmap; chunk size; pipeline |
| **Challenge** | Improving the data rate; data dependency |
| **Opportunity** | SHA and LZW may run in parallel |
| **Continuum** | All the data transmission processes |
| **Equation for Benefit** | Parallelism with LZW: *T=max(T\_SHA, T\_LZW)* |
| **Equation for Resources** | Resource = *Resource\_SHA*+*Resource\_LZW* |

**6. Document your design.**

1. Coding Resources:
   1. CDC Implementation:   
      <https://github.com/fanzhang312/Data-deduplication>
   2. LZW Algorithm and C++ Implementation: <https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/>
   3. SHA256 C Implementation: <https://github.com/amosnier/sha-2/blob/master/sha-256.c> <https://github.com/B-Con/crypto-algorithms/blob/master/sha256.c> <https://medium.com/a-42-journey/implementing-the-sha256-and-md5-hash-functions-in-c-78c17e657794>   
      <https://create.stephan-brumme.com/hash-library/>
2. Current compression ratio and breakdown of contribution from deduplication and from LZW compression.
   1. Contribution from deduplication (CDC)  
      How many chunks did we not have to re-encode?
   2. Contribution from LZW compression (including SHA-256)  
      (from LZW implementation)
   3. Compression Ratio (Overall)
3. **Overall throughput (Gb/s) of your current implementation.**

Works up till SHA

1. Description of all validation performed on your current functional implementation.
   1. Test each functions separately wherever possible (SHA, LZW, CDC)
   2. Run pipeline using known encoded output (e.g. Little\_Prince.txt)
2. **Report the raw ethernet speed measurements (Problem5) for all 3 partner's machines.**
   1. Anthony Stewart: 514 MBits/sec
   2. Sheil Sarda: 880 MBits/sec
   3. Shaokang Xia: 882 MBits/sec
3. **Description of who did what. How did your team collaborate on the design, implementation, and validation?**
   1. **Anthony Stewart:** LZW Implementation, Part of the SHA Implementation, Designing the pipeline interface
   2. **Sheil Sarda:** CDC Implementation, SHA Library Implementation, Designing the pipeline interface
   3. **Shaokang Xia:** Theory part: figuring out the axes, challenges, etc.

**7. Identify any challenges your group had in collaboration and design integration this week and how you plan to address them for future weeks.**

* Working around using std::map for LZW and implementing the same functionality using parallel std::vectors
* Changing our interface from what we proposed in the earlier milestone to accommodate more pipelined designs
* Move the hashmap to inside the LZW encoding loop in the pipeline so that we can easily check if a chunk has already been encoded before