Supplementary File of Parallelizing Exact and Approximate String Matching via Inclusive Scan on a GPU

Yasuaki Mitani, Fumihiko Ino, *Member, IEEE,* and Kenichi Hagihara

8 PROOFS

Lemma 1. *The bitwise left shift operator* $\ll$: $\mathbb{F}^w \times \mathbb{Z} \to \mathbb{F}^w$ *satisfies* $(\mathbf{x} \ll u) \ll v = \mathbf{x} \ll (u+v)$, *where* $\mathbf{x} \in \mathbb{F}^w$ *is a bit-vector of size* w *and* $u, v \in \mathbb{Z}$ *are non-negative integers.*

Proof. $\mathbf{x} \ll u \Leftrightarrow \mathbf{x} \times 2^u$. Hence, $(\mathbf{x} \ll u) \ll v \Leftrightarrow (\mathbf{x} \times 2^u) \times 2^v \Leftrightarrow \mathbf{x} \times 2^{(u+v)} \Leftrightarrow \mathbf{x} \ll (u+v)$. □

Lemma 2. *The bitwise OR operator* $|$: $\mathbb{F}^w \times \mathbb{F}^w \to \mathbb{F}^w$ *is associative.*

Proof. When $w = 1$, $(x \mid y) \mid z = x \mid (y \mid z)$, for all $x, y, z \in \mathbb{F}$. When $w > 1$, $(\mathbf{x} \mid \mathbf{y}) \mid \mathbf{z} = \mathbf{x} \mid (\mathbf{y} \mid \mathbf{z})$, for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}^w$, is equivalent to $(x_i \mid y_i) \mid z_i = x_i \mid (y_i \mid z_i)$, for all $1 \le i \le w$, where $\mathbf{x} = (x_1, x_2, \cdots, x_w)$, $\mathbf{y} = (y_1, y_2, \cdots, y_w)$, and $\mathbf{z} = (z_1, z_2, \cdots, z_w)$. Thus, arbitrary bits can be computed independently, and thereby $|$ is associative. □

Lemma 3. $\ll$ *is right-distributive over* $|$.

Proof. Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}^w$ and $u \in \mathbb{Z}$. Then,

$$\begin{aligned}&(\mathbf{x} \mid \mathbf{y}) \ll u\\ &= (x_1 \,|\, y_1,\ x_2 \,|\, y_2,\ \cdots,\ x_w \,|\, u_w) \ll u\\ &= (\underbrace{0, \cdots, 0}_{u},\ x_1 \,|\, y_1,\ x_2 \,|\, y_2,\ \cdots,\ x_{w-u} \,|\, u_{w-u}). \end{aligned} \tag{1}$$

Conversely,

$$\begin{aligned}&(\mathbf{x} \ll u) \mid (\mathbf{y} \ll u)\\ &= (\underbrace{0, \cdots, 0}_{u}, x_1, \cdots, x_{w-u}) \mid (\underbrace{0, \cdots, 0}_{u}, y_1, \cdots, y_{w-u})\\ &= (\underbrace{0, \cdots, 0}_{u},\ x_1 \,|\, y_1,\ x_2 \,|\, y_2,\ \cdots,\ x_{w-u} \,|\, y_{w-u}). \end{aligned} \tag{2}$$

Equation (1) is equivalent to Eq. (2), thus $\ll$ is right-distributive over $|$. □

Lemma 7. $(\langle x_1, y_1\rangle \uplus \langle x_2, y_2\rangle) \uplus \langle x_3, y_3\rangle = \langle x_1, y_1\rangle \uplus (\langle x_2, y_2\rangle \uplus \langle x_3, y_3\rangle)$, *for all* $x_1, x_2, x_3, y_1, y_2, y_3 \in \mathbb{F}$, *where* $\mathbb{F}$ *is the set of a binary finite field.*

- *Y. Mitani is with the Development Head Office, DWANGO Corporation, Ltd., 4-12-15 Ginza, Chuo-ku, Tokyo 104-0061, Japan.*
- *F. Ino and K. Hagihara are with the Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan. E-mail: ino@ist.osaka-u.ac.jp*

Proof. Because $\langle x_1, y_1\rangle \uplus \langle x_2, y_2\rangle \stackrel{\text{def}}{=} \langle x_1 \,|\, x_2,\ (y_1 \,\&\, {\sim}x_2) \mid y_2\rangle$, we have

$$\begin{aligned}&(\langle x_1, y_1\rangle \uplus \langle x_2, y_2\rangle) \uplus \langle x_3, y_3\rangle\\ &= \langle (x_1 \,|\, x_2) \,|\, x_3,\ (((y_1 \,\&\, {\sim}x_2) \mid y_2) \,\&\, {\sim}x_3) \mid y_3\rangle. \end{aligned} \tag{3}$$

Conversely,

$$\begin{aligned}&\langle x_1, y_1\rangle \uplus (\langle x_2, y_2\rangle \uplus \langle x_3, y_3\rangle)\\ &= \langle x_1 \,|\, (x_2 \,|\, x_3),\ (y_1 \,\&\, {\sim}(x_2 \,|\, x_3)) \mid ((y_2 \,\&\, {\sim}x_3) \mid y_3)\rangle. \end{aligned} \tag{4}$$

With respect to the first component of Eqs. (1) and (2), we have $(x_1 \mid x_2) \mid x_3 \Leftrightarrow x_1 \mid (x_2 \mid x_3)$. On the other hand, the distributive property rewrites the second component of Eq. (1) as

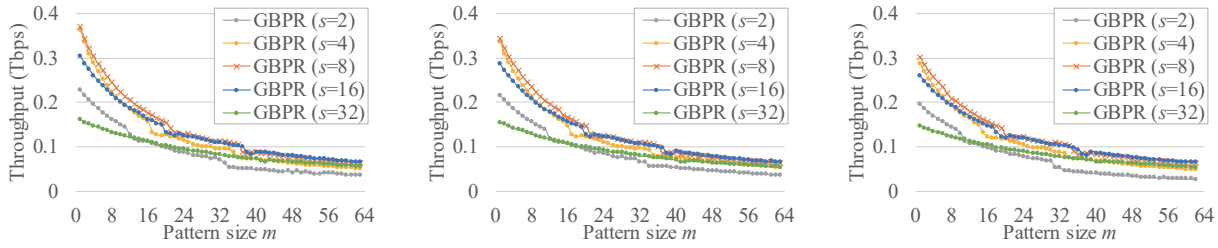
$$\begin{aligned}&(((y_1 \,\&\, {\sim}x_2) \mid y_2) \,\&\, {\sim}x_3) \mid y_3\\ &= ((y_1 \,\&\, {\sim}x_2) \,\&\, {\sim}x_3) \mid (y_2 \,\&\, {\sim}x_3) \mid y_3, \end{aligned} \tag{5}$$

which is equivalent to the second component of Eq. (2). □

9 PERFORMANCE ANALYSIS OF GBPR

The GBPR implementation is based on a segmentation-based scheme. Given segment size s, text of length n is divided into $\lceil n/s \rceil$ overlapping segments to allow $\lceil n/s \rceil$ threads to independently run the sequential bit-parallel algorithm for their responsible segments. Although GBPR uses shared memory for bit masks, off-chip memory is accessed when updating automata states; at each of these updates, threads fetch 1-byte text data from global memory and output the corresponding 1-byte search result to global memory. Consequently, each warp of 32 threads triggers s-strided accesses to off-chip memory. Increasing s reduces the redundancy in search, but at the same time drops the efficiency of off-chip memory accesses due to uncoalesced memory accesses.

Figure 12 shows how GBPR varies its search throughput depending on segment size s. In our sensitivity study on s, GBPR achieved the maximum performance on our experimental machine when $s = 8$. We think that this result is reasonable because $s = 8$ is also the best value reported in



(a) (b) (c)

Fig. 12: Search throughput of GBPR with different segment sizes s and pattern sizes m. Results are shown for (a) $k = 0$, (b) $k = 1$, and (c) $k = 2$. Text and alphabet sizes were fixed to $n = 2^{31}$ and $\sigma = 64$, respectively. s represents the segment size.

the original GBPR paper [1]. Similar to the results in Fig. 16 of [1], we found that large s dropped the performance due to the inefficient memory accesses mentioned above.

Finally, we compare the efficiency of our GBPR implementation with that of the original GBPR implementation [1] to show that ours is highly competitive. The original GBPR achieved search throughputs of 15–75 Gbps on a GeForce GTX 680 GPU [1], providing a memory bandwidth of 192.3 GB/s. Assuming that the search throughput is dominated by the memory performance rather than the arithmetic performance, the efficiency ranged from 1% to 5%. Conversely, our GBPR implementation ($\alpha = 1$) achieved search throughputs of 62–371 Gbps on the experimental GPU, providing a memory bandwidth of 336.5 GB/s. Here efficiency ranged from 2% to 14%, thus we conclude that our GBPR implementation ($\alpha = 1$) is highly competitive with the original version.

REFERENCES

[1] C. H. Lin, G. H. Wang, and C. C. Huang, "Hierarchical parallelism of bit-parallel algorithm for approximate string matching on GPUs," in *Proc. Symp. Comput. Appl. and Commun.*, Jul. 2014, pp. 76–81.