XSW: Accelerating Biological Database Search on Xeon Phi

School of Computer Science and Technology
Shandong University, China

May, 2014
Contents

- Motivation
- Smith-Waterman Algorithm
- Mapping onto the Xeon Phi
- Performance Evaluation
- Conclusions and Future Work
Bio DB Scanning on Xeon Phi: Motivation (1/3)

- Genome sequence databases are growing rapidly
- Growth rate will continue, since multiple concurrent genome projects have begun, with more to come
  - 3699 genomes published
    (http://www.genomesonline.org/ (Sep, 2012))
  - 10031 genome sequencing projects ongoing
Bio DB Scanning on Xeon Phi: Motivation (2/3)

- Discovered sequences need to be analyzed/annotated
- Typical operations
  - Database Scanning
  - Multiple Sequence Alignment
  - Hidden Markov Model training and scoring
  - Computing Evolutionary Trees

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Doubling time (year)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genome databases</td>
<td>1.44</td>
</tr>
<tr>
<td>PC speed (number of transistors)</td>
<td>2.09</td>
</tr>
<tr>
<td>Supercomputer speed (LINPACK)</td>
<td>1.04</td>
</tr>
</tbody>
</table>

- Establishes the need for High Performance Computing (HPC)
- HPC Alternatives
  - Coarse-grained (e.g. Clusters, Grids, Clouds)
  - Fine-grained (e.g. FPGAs, GPUs)
Bio DB Scanning on Xeon Phi: Motivation(3/3)

- High performance/price ratio
- Easy programming
Smith-Waterman Algorithm

- Performs an exhaustive search for the optimal local alignment of two sequences.

- Aligning $S_1$ and $S_2$ of length $l_1$ and $l_2$ using Recurrences:

$$H(i, j) = \max \begin{cases} 
0, 
E(i, j), 
F(i, j), 
H(i-1, j-1) + Sbt(S_1_i, S_2_j)
\end{cases}$$

$$1 \leq i \leq l_1, 1 \leq j \leq l_2$$

$H(i,0) = E(i,0) = 0$

$H(0,j) = F(0,j) = 0$

$E(i,j) = \max \begin{cases} 
H(i, j-1) - \alpha, 
E(i, j-1) - \beta
\end{cases}$

$F(i,j) = \max \begin{cases} 
H(i-1, j) - \alpha, 
F(i-1, j) - \beta
\end{cases}$

AsHES 2014
Smith-Waterman Algorithm

Align \( S1 = \text{ATCTCGTATGATG} \) \( S2 = \text{GTCTATCAC} \)

\[
Sbt(x, y) = \begin{cases} 
2 & \text{if } (x = y) \\
-1 & \text{else}
\end{cases}
\]

\( \alpha = 1, \beta = 1 \)

\[
H(i, j) = \max \begin{cases} 
0 \\
H(i - 1, j) - 1 \\
H(i, j - 1) - 1 \\
H(i - 1, j - 1) + Sbt(S1_i, S2_j)
\end{cases}
\]
Parallel SW on Multi-core CPU

A. Wozniak (1997)


Michael Farrar (2007)

T. Rognes (2011)
- Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation, BMC Bioinformatics, 12:221, 2011. (Impact Factor: 3.02)
T. Oliver, etc. (2005)


W. Liu, etc. (2007)


A. Wirawan, etc. (2008)


Y. Liu, etc. (2013)

- CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions, BMC Bioinformatics, 14:117, 2013. (Impact Factor: 3.02)
Our Algorithm Framework

Biological sequence database

Preprocessed database

Coarse-grained parallel processing by multi-threading

DB subset 1

Thread 1

Fine-grained parallel processing by VPU

Intermediate results

...  

DB subset m

Thread m

Fine-grained parallel processing by VPU

Intermediate results

Result merging and readback to CPU for output
Coarse-grained Parallelism

- Database is partitioned into small subsets
  - Reduce the superfluous computation
  - Achieve better load balancing
Fine-grained Parallelism

The break point info:

\[ \text{Num} = 2; \ C_8, C_{11} \]
Fine-grained Parallelism

A sequence subset which is packed into 2D buffer

Query sequence

16 channels

Searching direction

Block 1  Block 2  Block 3  Block 4  ...

AsHES 2014
Fine-grained Parallelism

\[
H(i, j) = \max\begin{cases}
0 \\
E(i, j) \\
F(i, j) \\
H(i-1, j-1) + Sbt(S1_i, S2_j)
\end{cases}, \quad 1 \leq i \leq l1, 1 \leq j \leq l2
\]

\[
\begin{align*}
H(i,0) &= E(i,0) = 0 \\
H(0,j) &= F(0,j) = 0 \\
E(i,j) &= \max\begin{cases}
H(i, j-1) - \alpha \\
E(i, j-1) - \beta
\end{cases}, \quad F(i,j) = \max\begin{cases}
H(i-1, j) - \alpha \\
F(i-1, j) - \beta
\end{cases}
\end{align*}
\]

\[
\begin{align*}
1. & \quad \text{vH[i]} = _\text{mm512_add_epi32}(\text{vH[i]}, \text{vSbt}); \quad // H = H + sbt[q] \\
2. & \quad \text{vH[i]} = _\text{mm512_max_epi32}(\text{vH[i]}, \text{vF[i]}); \quad // H = \max(H, F) \\
3. & \quad \text{vH[i]} = _\text{mm512_max_epi32}(\text{vH[i]}, \text{vE}); \quad // H = \max(H, E) \\
4. & \quad \text{vH[i]} = _\text{mm512_max_epi32}(\text{vH[i]}, \text{vZero}); \quad // H = \max(H,0) \\
5. & \quad \text{vS} = _\text{mm512_max_epi32}(\text{vS}, \text{vH[i]}); \quad // S = \max(S, H) \\
6. & \quad \text{vF[i]} = _\text{mm512_sub_epi32}(\text{vF[i]}, \beta); \quad // F = F - \beta \\
7. & \quad \text{vE} = _\text{mm512_sub_epi32}(\text{vE}, \beta); \quad // E = E - \beta \\
8. & \quad \text{vN[i]} = _\text{mm512_mask_mov_epi32}(\text{vN[i]}, 0xffff, \text{vH[i]}); \quad // N = H \\
9. & \quad \text{vH[i]} = _\text{mm512_sub_epi32}(\text{vH[i]}, \alpha); \quad // H = H - \alpha \\
10. & \quad \text{vE} = _\text{mm512_max_epi32}(\text{vH[i]}, \text{vE}); \quad // E = \max(H, E) \\
11. & \quad \text{vF[i]} = _\text{mm512_max_epi32}(\text{vH[i]}, \text{vF[i]}); \quad // F = \max(H, F)
\end{align*}
\]
Fine-grained Parallelism

Score Matrix

Database Sequences

Shuffling procedure
Fine-grained Parallelism

Mask \( dbSeq < 16 \)

\[
\begin{array}{cccccccccccccccccccc}
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1
\end{array}
\]

Database sequence

\[
\begin{array}{cccccccccccccccccccc}
8 & 6 & 3 & 20 & 7 & 19 & 15 & 4 & 18 & 22 & 5 & 11 & 21 & 17 & 2 & 9
\end{array}
\]

Mask \( dbSeq \geq 16 \)

\[
\begin{array}{cccccccccccccccccccc}
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0
\end{array}
\]

Database sequence

\[
\begin{array}{cccccccccccccccccccc}
8 & 6 & 3 & 20 & 7 & 19 & 15 & 4 & 18 & 22 & 5 & 11 & 21 & 17 & 2 & 9
\end{array}
\]

Vector register
Performance Evaluation

- XSW: Implemented using C, Pthreads, and KCI.

- Performance evaluation on a PC server with an Intel E5-2620 six-core 2.0GHz CPU and an Intel Xeon Phi 7110P card. The server has 16GB RAM and runs Linux Red Hat 6.3.

- Performance comparison to SWIPE and CUDASW++ 3.0 running on a K20 GPU which is installed on the same PC server.

- Two biological databases are used: Swiss-Prot (541,954 sequences) and Environmental NR (6,165,520 sequences).
Performance Evaluation

- Performance comparison for scanning the Swiss-Prot.
Performance Evaluation

- Performance comparison for scanning the Environmental NR.
Conclusion

• Xeon Phi offers a flexible solution with a very good price/performance ratio for the SW algorithm (http://sdu-hpcl.github.io/XSW/)

• Achieved better performance than SWIPE and CUDASW++ 3.0 on an Xeon Phi 7110P

• Since the performance of many-core architectures grows faster than multi-core CPU, Xeon Phi-centric HPC will become even more important in the future
Future Work

XOmics
- Design and develop Omics-related algorithms on Xeon Phi

XFILE
- an Efficient File System for Processing Large-scale Data using Xeon Phi

XMR
- A Heterogeneous Architecture-based MapReduce Framework for Large-scale Data Processing

XDC
- Xeon Phi Accelerated Compression Framework for Large-scale Data
New Results: XSW 2.0

- Scanning large-scale databases using the offload programming model.
New Results: XSW 2.0

- Performance comparison to SWAPHI for scanning the Environmental NR.
New Results: XSW 2.0

- Performance for scanning large-scale DB (NR + TrEMBL, totally 36GB).