ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis

Can Firtina[1], Kamlesh Pillai[2], Gurpreet S. Kalsi[2], Bharathwaj Suresh[2], Damla Senol Cali[3], Jeremie S. Kim[1], Taha Shahroodi[4], Meryem Banu Cavlak[1], Joel Lindegger[1], Mohammed Alser[1], Juan Gómez Luna[1], Sreenivas Subramoney[2], and Onur Mutlu[1,3]

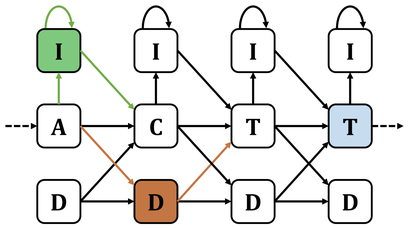
SAFARI [1] ETH zürich [2] intel [3] Carnegie Mellon [4] TU Delft — Delft University of Technology

arXiv Preprint

Source Code

1: Profile Hidden Markov Models (pHMMs)

PHMM Sequence: ...ACTT...
Sequence #1: ...AGGGCTT...
Sequence #2: ...ATT... (Deleted C)
Sequence #3: ...ACTG...



Insertions (I) | **Deletions (D)** | **Substitution or Match**

PHMMs are useful to identify **differences and similarities** between sequences using:

- **Transitions** between states
- Recognition (**emission**) of each character within states
- **Probabilities** assigned to transitions and emissions

2: The Baum-Welch Algorithm

1. Forward Calculation

$$F_t(i) = \sum_{j \in V} F_{t-1}(j)\alpha_{ji}e(S[t], v_i) \quad i \in V, \quad 1 < t \leq n_S$$

2. Backward Calculation

$$B_t(i) = \sum_{j \in V} B_{t+1}(j)\alpha_{ij}e(S[t+1], v_j) \quad i \in V, \quad 1 \leq t < n_S$$

Massive Parallelism

Data Dependency

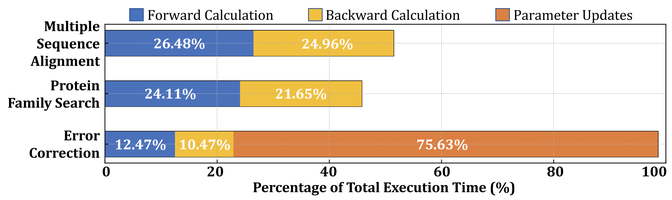
3.1. Updating Transitions

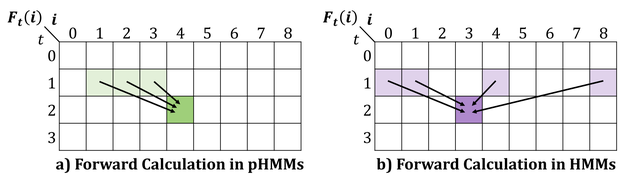
$$\alpha^*_{ij} = \frac{\sum_{t=1}^{n_S-1} \alpha_{ij} e_{S[t+1]}(v_j) F_t(i) B_{t+1}(j)}{\sum_{t=1}^{n_S-1} \sum_{x \in V} \alpha_{ix} e_{S[t+1]}(v_x) F_t(i) B_{t+1}(x)} \quad \forall \alpha_{ij} \in A$$

3.2. Updating Emissions

$$e^*_X(v_i) = \frac{\sum_{t=1}^{n_S} F_t(i) B_t(i)[S[t] = X]}{\sum_{t=1}^{n_S} F_t(i) B_t(i)} \quad \forall X \in \Sigma, \forall i \in V$$

3: Sources of Inefficiencies in the Baum-Welch Algorithm





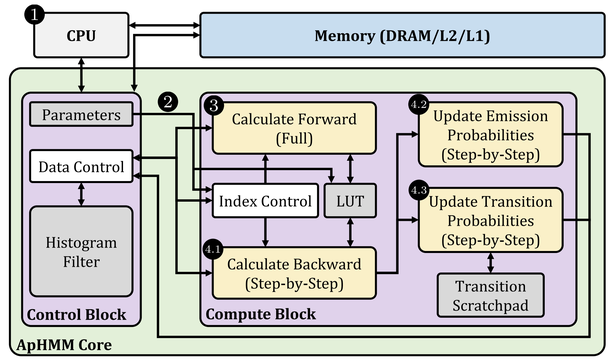
The Baum-Welch algorithm is the **main computational overhead**

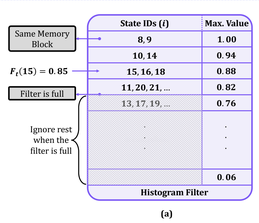
Standard HMM accelerators are **oblivious to the data dependency pattern** in pHMMs

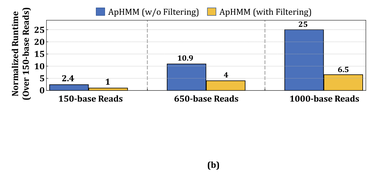
4: Problem and Goal

- **The Baum-Welch algorithm is useful** for many applications but **remains computationally costly due to several sources of inefficiencies**
- Our goal is to
  - **Accelerate the Baum-Welch algorithm** while
  - **Eliminating its inefficiencies**
  - **Using a hardware-software co-design**
- **ApHMM is the first work that** accelerates the Baum-Welch algorithm for pHMMs

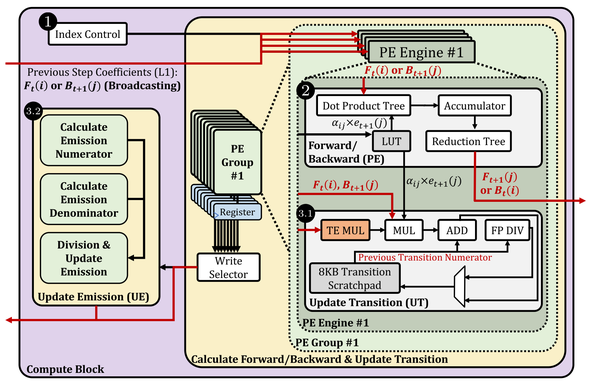
5: ApHMM Overview & Filtering Mechanism







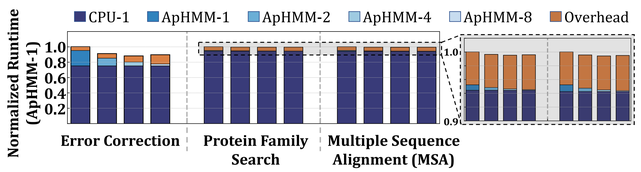
6: Computing the Baum-Welch Algorithm



7: Evaluation Methodology

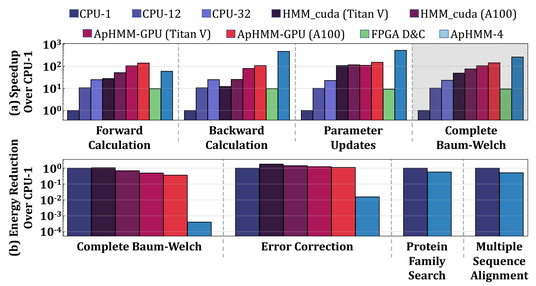
- ApHMM **(ASIC)** and ApHMM-**GPU** implementations
- Compared with **CPU, GPU, and FPGA baselines**
- Use cases
  1. Error correction and polishing
  2. Multiple sequence alignment
  3. Protein Family Search
- Evaluating
  1. Area and power
  2. Performance
- Datasets
  - Error correction: 50,000 random E. coli reads
  - Protein family search: Mapping Mitochondrial carrier to the entire Pfam database
  - Multiple sequence alignment: Aligning commonly used protein families to each other

8: Results



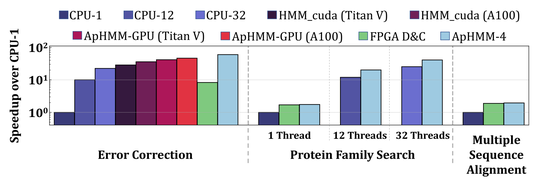
✓ 4-core ApHMM provides the **best overall speedup**

✗ **Limited scalability** for using many cores with ApHMM due to the **data movement overhead**



✓ **Significant improvements** in terms of **performance and energy consumption**

✗ **Limited performance improvements** for the Forward Calculation due to the **data movement overhead**



✓ **ApHMM significantly accelerates the end-to-end execution times** of important applications