**Objective**

We explore how different runtime systems can be implemented on the Intel Xeon Phi System on Beacon. This coprocessor has its own Intel MKL library that implements BLAS and LAPACK functionality. For this research, we will first explore how to utilize PLASMA for handling dense linear algebra computations and QUARK for task management and added parallelism to figure out the dependencies between the tasks and the scheduler. Once accomplished, these algorithms will be rigorously tested on the Beacon’s MIC card for performance analysis and comparison with the standard Intel MKL implementation. Another goal is to implement a hybrid Out-of-Core algorithm for Cholesky factorization that can be used in conjunction with the PLASMA/QUARK implementation to see if its performance is efficient and scalable.

**Visual of the overview**

- **PLASMA** – dense algebra algorithms
- **QUARK** – multithreading and task management
- **Intel MKL Library** – optimized math library for comparison with PLASMA

**Modes of execution**

- Native and Offload. The former relies on programming directly into the coprocessor (MIC card). The goal for further optimization is using Offload Mode, which will run on the host processor and “offload” the dense calculations to the co-processor.

**Cholesky Factorization (OOC)**

Cholesky Factorization: \(A = LL^T\)

where \(A\) is SPD (symmetric, positive definite).

\[
\begin{align*}
A_{11} & \quad A_{12} \\
A_{21} & \quad A_{22}
\end{align*}
\]

Recursive Cholesky steps on matrix blocks

- Step 1: \(L_{11} \leftarrow \text{cholesky}(A_{11})\), potrf()
- Step 2: \(L_{21} \leftarrow A_{21} / L_{11}^T\), trsm()
- Step 3: \(A_{22} \leftarrow A_{22} - L_{21} \cdot L_{11}^T, \text{symr()}\) and gemm()
- Step 4: \(L_{22} \leftarrow \text{cholesky}(A_{22})\), potrf()

**Task directed acyclic graph (DAG)**

- Tasks in Cholesky factorization depend on previous tasks if they use the same tiles of data. If we use a node to represent an operation on a tile and use an edge to represent a data dependency, then a DAG is formed.
- Once the DAG is produced and fed into the QUARK runtime system, tasks can be scheduled asynchronously and independently as long as the dependencies are not violated. Here a 4x4 Cholesky example is shown. (i,j,k): operation in the k-th iteration on the (i,j)-th tile.

**Pseudocode for DAG:**

```plaintext
for k=0...n-1
    for j=k...n-1
        for i=j...n-1
            if (i=j+k) potrf(A(i,i,k-1),A(i,i,k))
            if (i=j+k) trsm(A(i,i,k-1),A(i,i,k-1)'A(i,i,k-1)-1,A(i,j,k))
            if (i=j+k) symr(A(i,j,k-1)',A(i,j,k-1),A(i,j,k))
            if (i=j+k) gemm(A(i,i,k-1'),A(i,j,k-1)',A(i,j,k))
```

**Out-of-Core Algorithm (OOC)**

- OOC stores most data on CPU memory and brings small pieces of data into coprocessors for computation, and then writes them back. It takes advantage of the computational efficiency of hardware accelerators without limiting the size of the matrix problem.
- CPU vs coprocessors (GPU, Intel MKL, etc.): GPU is much faster and more energy efficient than CPU but has limited amount of device memory.

**OOC structure**

- The out-of-core part loads parts of the matrix. For example, matrix panels, to device memory, and applies the “left-looking” update from the parts already factorized and written back.
- The In-core part factorizes the parts residing on device memory in which “right-looking” update is involved.

**Proposed methodology**

- Matrix Multiplication and other BLAS routines (QUARK, Intel MKL)
- Hello World Multithreading (QUARK, Intel MKL)
- Performance Testing in seconds and GFLOPS - Giga Floating Operations Per Second (PLASMA, Intel MKL)

**Current progress**

I have modified example code from Dr. Asim Yarkhan, one of the main developers of QUARK, for a matrix multiplication driver that will measure the performance of serial implementation and QUARK multi-threading.

**Preliminary results**

<table>
<thead>
<tr>
<th>Matrix Multiplication Performance Test (GFLOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Matrix Multiplication Performance Test (GFLOPS)" /></td>
</tr>
</tbody>
</table>

- The data will be printed in a user-friendly manner and measure the data in seconds and GFLOPS.
- To generate GFLOPS/sec, under the assumption for multi-threading \(C = A \times B\) that \(A, B, C\) are symmetric matrices \((n \times n)\), then the general formula would be:

\[
\frac{2n^2}{10^6} \times \text{sec}
\]

**Expected goals**

- A driver optimized to implement QUARK and MKL threading to perform matrix multiplication and eventually other BLAS/LAPACK routines. The data measurements should be organized and readable for both new and experienced users.
- Given the current DAG, optimize the Cholesky DAG and eventually replicate the DAG on QUARK.

**References**

- Yarkhan, Asim, Jakub Kurzak, and Jack Dongarra. QUARK Users’ Guide. April 2011

**Team Info**

- **Authors:** Allan Richmond Razon Morales and Tian Chong
- **Mentors:** Dr. Kwai Wong (UTK), Dr. Eduardo D’Azevedo (ORNL)
- **Collaborators:** Dr. Shiquan Su, Dr. Asim Yarkhan, and Ben Chan