Proposal For ASC 25

Team Name: `[Team Name]`

Team Member 1
Team Member 2
Team Member 3
Team Member 4
Team Member 5

January 8, 2025

A proposal submitted to the ASC 25 committee

Contents

1 Brief Background Description of Supercomputing Activities 3
1.1 Hardware and Software Platforms . . . 3
1.2 Example Code Block . . . 3
1.3 Example Figure . . . 3

2 Design of HPC System 4
2.1 Performance Analysis . . . 4

3 Introduction to the University's Activities in Supercomputing 5
3.1 Supercomputing-related Hardware and Software Platforms . . . 5
3.2 Supercomputing-related Courses, Trainings, and Interest Groups . . . 5
3.3 Supercomputing-related Research and Applications . . . 5
3.4 Key Achievements in Supercomputing Research . . . 5

4 Team Introduction 6
4.1 Team Setup . . . 6
4.2 Team Members . . . 6
4.3 Team Motto . . . 6

5 Technical Proposal Requirements 7
5.1 Design of HPC System . . . 7
5.1.1 Theoretical Design of an HPC Cluster . . . 7
5.1.2 Software and Hardware Configurations . . . 7
5.1.3 Interconnection, Power Consumption, Performance Evaluation, and Architecture Analysis . . . 7
5.2 HPL and HPCG Benchmarks . . . 7
5.2.1 Software Environment . . . 7
5.2.2 Performance Optimization and Testing Methods . . . 7
5.2.3 Performance Measurement and Problem/Solution Analysis . . . 7
5.2.4 In-depth Analysis of HPL and HPCG Algorithms and Source Codes 7
5.3 Optimization for AlphaFold3 Inference . . . 7
5.4 GPU Inference Optimization . . . 7
5.4.1 Model Deployment . . . 7
5.4.2 Hardware Configuration . . . 7
5.4.3 Environment Variables Setup . . . 7
5.4.4 Program Execution Command . . . 8
5.4.5 Program Results . . . 8
5.5 Program Optimization . . . 8
5.5.1 Optimization Strategy Overview . . . 8
5.5.2 Optimization Methods . . . 8
5.5.3 Optimization Methods . . . 8
5.5.4 Optimization Results . . . 8
5.6 CPU Inference Optimization . . . 9
5.6.1 Model Deployment . . . 9
5.6.2 Hardware Configuration . . . 9
5.6.3 Environment Variables Setup . . . 9
5.6.4 Program Execution Command . . . 9

5.6.5 Program Results 9
5.7 Program Optimization 9
5.7.1 Optimization Strategy Overview 9
5.7.2 Optimization Methods 10
5.7.3 Optimization Results 10
5.8 Program Execution Process 10
5.8.1 Input Processing 10
5.8.2 Feature Extraction 10
5.8.3 Model Inference 10
5.8.4 Result Processing 10

6 RNA m5C Modification Site Detection and Performance Optimization Challenge 11
6.0.1 Workflow Description 11
6.0.2 m5C Sites File 11
6.0.3 Software Packaging 11
6.0.4 Performance Optimization 11

7 Additional Materials 12

A Additional Technical Details 13
A.1 Configuration Files 13

B References 14

1 Brief Background Description of Supercomputing Activities

1.1 Hardware and Software Platforms

Our university established a high-performance computational (HPC) cluster in 2025 to address the growing demands in scientific research and industrial applications. Here's an example of how to include technical specifications:

Table 1: Hardware Configuration of YSU HPC Cluster

Item	Name	Configuration	Number
Login Node	[Model]	CPU: [Specs]	1
Compute Node	[Model]	CPU: [Specs]	10
GPU Node	[Model]	GPU: [Specs]	2

1.2 Example Code Block

Here's how to include code samples in the document:

Listing 1: HPL Performance Testing Script

```
def test_hpl_performance(problem_size, block_size):
    """
    Test HPL performance with given parameters
    """
    results = []
    for size in problem_size:
        perf = run_hpl_benchmark(size, block_size)
        results.append((size, perf))
    return results
```

1.3 Example Figure

Example of including a figure with caption:

This is a placeholder for the performance graph

Figure 1: HPL Performance Scaling Analysis

2 Design of HPC System

2.1 Performance Analysis

Example of including mathematical equations:

$$R_{peak} = N_{cores} \times N_{flops/cycle} \times F_{clock} \tag{1}$$

Where:

- N_{cores} is the total number of CPU cores
- $N_{flops/cycle}$ is the number of floating-point operations per cycle
- F_{clock} is the clock frequency in Hz

3 Introduction to the University's Activities in Supercomputing

3.1 Supercomputing-related Hardware and Software Platforms

3.2 Supercomputing-related Courses, Trainings, and Interest Groups

3.3 Supercomputing-related Research and Applications

3.4 Key Achievements in Supercomputing Research

4 Team Introduction

4.1 Team Setup

4.2 Team Members

4.3 Team Motto

5 Technical Proposal Requirements

5.1 Design of HPC System

5.1.1 Theoretical Design of an HPC Cluster

5.1.2 Software and Hardware Configurations

5.1.3 Interconnection, Power Consumption, Performance Evaluation, and Architecture Analysis

5.2 HPL and HPCG Benchmarks

5.2.1 Software Environment

5.2.2 Performance Optimization and Testing Methods

5.2.3 Performance Measurement and Problem/Solution Analysis

5.2.4 In-depth Analysis of HPL and HPCG Algorithms and Source Codes

5.3 Optimization for AlphaFold3 Inference

5.4 GPU Inference Optimization

5.4.1 Model Deployment

Since the program requires more than 18GB of video memory, we chose to deploy the model on the YSU HPC supercomputing cluster. Dependencies were installed according to the official dockerfile documentation. Due to the cluster's CUDA driver version being 12.0, which differs from the dockerfile's 12.6, we needed to select a different jax version. According to the jax installation documentation, we used the following commands to install jax:

```
pip install --upgrade pip
pip install --upgrade "jax[cuda12]"
```

After installation, the jax version is 0.5.0, which differs from the version required in the dockerfile but still runs normally.

5.4.2 Hardware Configuration

Using the compute01 node of the supercomputing cluster, which is configured with: CPU: Intel Xeon Gold 5218R @ 2.10GHz, GPU: 2* NVIDIA GeForce 3090 24G, RAM: 125G.

5.4.3 Environment Variables Setup

```
export XLA_PYTHON_CLIENT_MEM_FRACTION=0.95
export JAX_TRACEBACK_FILTERING=off
```

5.4.4 Program Execution Command

```
python ./run_alphafold.py \
? --input_dir=/input_dir \
? --output_dir=/output_dir \
? --model_dir=/model_dir \
? --norun_data_pipeline \
? --num_recycles=3 \
? --flash_attention_implementation=xla
```

5.4.5 Program Results

See cluster files for details.

5.5 Program Optimization

5.5.1 Optimization Strategy Overview

Through code analysis and observation of program execution results, we found that the model inference phase accounts for over 95% of the total runtime. Therefore, we prioritized optimizing the model inference time. Additionally, we identified optimization opportunities in the feature extraction phase.

5.5.2 Optimization Methods

Model Inference Phase: Used jax compilation cache directory to cache compiled functions and model parameters, reducing compilation time during model inference.
Feature Extraction Phase: Defined `FeatureCache` class to cache feature data, reducing repeated computations and memory usage. Implemented `optimize_features` function to optimize data types and memory layout, `compress_features` function to compress feature data, and parallel processing of feature data to reduce runtime.

5.5.3 Optimization Methods

Defined `create_model_runner` function to configure jax environment, including disabling 64-bit operations and setting thread count. Implemented `_post_process_result` function for optimizing result processing and data type conversion. Created `ModelRunner` class with `_split_batch`, `run_inference`, and `_merge_results` functions for dynamic batch processing, parallel model inference, and parallel result merging. Implemented `NumericsHandler` class with `handle_coordinate_numerics`, `handle_general_numerics`, and `check_output_numerics` functions for detecting and handling NaN/Inf values. Developed `CacheManager` class with `_get_cache_key`, `_serialize_value`, `_deserialize_value`, and `put` functions for cache management. Created `MemoryManager` class with `get_memory_usage`, `update`, `cleanup`, and `monitor` functions for memory management.

5.5.4 Optimization Results

Before optimization: Total runtime: 3651.96s, Model inference: 3353.36s, Feature extraction: 298.60s.

After optimization: Total runtime: 3149.68s, Model inference: 3042.80s, Feature extraction: 106.88s.
Total improvement: 13.7%, Model inference improvement: 9.3%, Feature extraction improvement: 66.9%.

5.6 CPU Inference Optimization

5.6.1 Model Deployment

Since the CPU version requires over 100GB of memory for large inputs, we chose to deploy the model on the login node of the YSU HPC supercomputing cluster, with the same deployment process as the GPU version.

5.6.2 Hardware Configuration

Using the login node configured with: CPU: Intel Xeon Gold 5218R @ 2.10GHz, RAM: 125G.

5.6.3 Environment Variables Setup

```
export MKL_DEBUG_CPU_TYPE=5
export MKL_ENABLE_INSTRUCTIONS=AVX2
export KMP_AFFINITY="granularity=fine,compact,1,0"
export MKL_DYNAMIC=FALSE
```

```
import os
os.environ['JAX_PLATFORMS'] = 'cpu'
os.environ['JAX_SKIP_ROCM_TESTS'] = '1'
os.environ['JAX_SKIP_TPU_TESTS'] = '1'
os.environ['JAX_LOG_COMPILES'] = '0'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

5.6.4 Program Execution Command

Same as GPU version.

5.6.5 Program Results

See cluster files for details.

5.7 Program Optimization

5.7.1 Optimization Strategy Overview

Through code analysis, we identified optimization opportunities in memory usage and CPU communication. We also found NaN/Inf values during program execution that needed handling.

5.7.2 Optimization Methods

Defined create_model_runner function to configure jax environment, including disabling 64-bit operations and setting thread count. Implemented _post_process_result function for optimizing result processing and data type conversion. Created ModelRunner class with _split_batch, run_inference, and _merge_results functions for dynamic batch processing, parallel model inference, and parallel result merging. Implemented NumericsHandler class with handle_coordinate_numerics, handle_general_numerics, and check_output_numerics functions for detecting and handling NaN/Inf values. Developed CacheManager class with _get_cache_key, _serialize_value, _deserialize_value, and put functions for cache management. Created MemoryManager class with get_memory_usage, update, cleanup, and monitor functions for memory management.

5.7.3 Optimization Results

Before optimization: Total runtime: 44102.1s After optimization: Total runtime: 43929.5s Total improvement: 0.4%, potentially limited by memory bandwidth based on CPU usage during runtime.

5.8 Program Execution Process

5.8.1 Input Processing

First, receives input in JSON format describing target molecule composition and experimental conditions, then sets parameters such as cycle count and diffusion sample number based on command-line arguments.

5.8.2 Feature Extraction

Performs MSA, filters structure templates based on sequence similarity and publication date, loads CCD and RDKit for non-standard residue processing and small molecule ligand 3D conformation generation, then encodes these into multidimensional vectors.

5.8.3 Model Inference

Processes sequence and pairing features, generates atomic coordinates, then performs iterative optimization through multiple cycles, using previous prediction results as input for each iteration.

5.8.4 Result Processing

Decodes atomic coordinates from model output Frame Transforms, calculates local bond lengths and angles, normalizes results, performs confidence assessment, and finally outputs 3D structures as PDB files.

6 RNA m5C Modification Site Detection and Performance Optimization Challenge

6.0.1 Workflow Description

6.0.2 m5C Sites File

6.0.3 Software Packaging

6.0.4 Performance Optimization

7 Additional Materials

A Additional Technical Details

A.1 Configuration Files

Example of including configuration files:

Listing 2: HPL Configuration File

```
# Sample HPL.dat
HPL.out      output file name
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
29000        Ns
1            # of NBs
256          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
2            Ps
2            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
```

B References

References

[1] Author, *Title of the Book*, Publisher, Year.