Can I Talk to my Supercomputer?
Conversational AI Interface for HPC Systems

Pouya Kousha, Hari Subramoni, DK Panda

The Network Based Computing Laboratory
The Ohio State University

http://mvapich.cse.ohio-state.edu/
Presentation Outline

• Introduction and Motivation
• Problem Statement
• Proposed Designs
• Performance Evaluation
• Demo
• Concluding Remarks
Introduction

• HPC usage is expanding: New users without HPC background
• Execution of parallel workloads is complex!
  – Learning new interfaces, features, and the terminologies
  – Complex dependencies, installation and execution
• Steep learning curves for executing tasks and utilizing HPC!
• Intuitive Expression: Users naturally convey needs through words & text
• Emergence of science gateways like Open OnDemand
  – GUI-based interface: more accessible and easier
Research Challenges

Creating an AI-enabled conversational interface for HPC faces several challenges, including:

- **Technical HPC terms**: ASR and NLU models lack tailored HPC datasets, hindering accurate interpretation of technical terms and abbreviations.
- **Complex relationships**: Mapping complex HPC component relationships can be time-consuming and complex.
- **Software installation**: HPC software package installation is a challenge, especially for novice users, even with package managers like Spack.
- **Conversational AI integration**: Integrating conversational AI into science gateways requires a modular interface and determining the interface between conversational AI and science gateways.
Proposed Framework for Conversational AI for HPC Tasks

HPC User

Select an HPC Application
Find and Install Dependencies
Install HPC application
Read the documentation
Create the Job script and launch the job

HPC System

Speech Input

Audio

Automatic Speech Recognition
Text

Natural Language Understanding

Entities

Entity/Value Extraction

Text Input

Text

Feedback/Questions

Is Complete?

Yes

SW Installed?

No

No

Yes

KG Inference

Software Installer

Job Script Generator

Select KG

Display Job Output
Training Speech and Text Processing Models for HPC

Generating New HPC Dataset
Same as CAI but extra steps
1. Basic queries and labels
2. Add combinations
3. Develop synonyms and mix
4. Include permutations

Five categories:
• System
• Software
• Model/Algorithm
• Args
• Data

Automatic Speech Recognition
- Model: Speech2Text
- Dataset: TIMIT + HPC-ASR [proposed]
- Pretrained on LibriSpeech

Natural Language Understanding (NLU)
- Model: Bert entity recognition
- Datasets: HPC-NLU [proposed]
- Identify entities and tasks

Speech2Text architecture

BERT architecture
Training Speech and Text Processing Models for HPC

New HPC Ontology

- Capture workload relations
- Create Knowledge Graph (KG) per application

<table>
<thead>
<tr>
<th>Relation Property</th>
<th>Domain</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>canBe</td>
<td>any</td>
<td>any</td>
<td>Defines possible values (OR)</td>
</tr>
<tr>
<td>runs</td>
<td>any</td>
<td>Software or Model</td>
<td>Captures run capability</td>
</tr>
<tr>
<td>depends</td>
<td>Software</td>
<td>System</td>
<td>Captures software dependency</td>
</tr>
<tr>
<td>hasArgs</td>
<td>any</td>
<td>Argument</td>
<td>Defines optional values (defaults)</td>
</tr>
<tr>
<td>hasSoftware</td>
<td>any</td>
<td>Software</td>
<td>Captures software availability</td>
</tr>
</tbody>
</table>

Knowledge Graph Query

- Queries all KGs
- Max-hit KG selection
- Gathers needed/opt Args
- Check against user input
- Query user till completion

Software Installer:

- Checks & installs dependencies via Spack
- Single Spack config and Env.
- Asynchronous installation

OnDemand Integration:

- HPC Integration and accessibility

Deployment modes:

- Passenger: shared resources
- Interactive: exclusive resources
DL Models Performance Evaluation

**ASR Evaluation:**
- Improved word error rate for Speech2Text model from 86.2% to 3.7%

**NLU Evaluation:**
- 60K training size, 5M test size
- Achieved 99% accuracy and precision

**ASR + NLU Evaluation:**
- Pipelines ASR+NLU for inference accuracy
- Testing 100 queries from 4 individuals
- M1: Adjust predicted sentence length to match original
- M2: Drop less important/incorrect words to match original content

---

<table>
<thead>
<tr>
<th>Train Dataset</th>
<th>Test Dataset</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (LibriSpeech)</td>
<td>HPC-ASR</td>
<td>86.2</td>
</tr>
<tr>
<td>Base+TIMIT+HPC-ASR</td>
<td>HPC-ASR</td>
<td>3.7</td>
</tr>
</tbody>
</table>

**Improved transition of speech to text**

<table>
<thead>
<tr>
<th>Test Dataset</th>
<th>F1-score</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC-NLU (5M)</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
</tr>
</tbody>
</table>

**Predicting entities correctly!**

<table>
<thead>
<tr>
<th>Metric</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>WER</td>
<td>10.3</td>
<td>8.6</td>
<td>8.3</td>
<td>4.9</td>
<td>8.03</td>
</tr>
<tr>
<td>Accuracy M2</td>
<td>0.97</td>
<td>0.90</td>
<td>0.80</td>
<td>0.95</td>
<td>0.907</td>
</tr>
<tr>
<td>Accuracy M1</td>
<td>0.84</td>
<td>0.81</td>
<td>0.83</td>
<td>0.92</td>
<td>0.849</td>
</tr>
</tbody>
</table>

User #1 and #4 are new to SAI and not used in training.

**High success rate and flexibility recognizing new users’ voices!**

Source: Kousha et al, SAI: AI-Enabled Speech Assistant Interface for Science Gateways in HPC, ISC'23
Overhead of SAI Passenger App Pipeline for Different Queries

End-to-end Overhead: SAI Full Pipeline as passenger App

- Evaluate inference latency for various speech/text queries
- Exclude software installation and execution timing
- Speech latency increases with more query words
- Text latency remains constant

End-to-end Overhead of SAI Passenger App with Multi-Users

- Higher avg. latency for speech/text with more concurrent users
- Speech queries more affected than text queries
- Login node performance degrades significantly with increased passenger mode usage

Source: Kousha et al, SAI: AI-Enabled Speech Assistant Interface for Science Gateways in HPC, ISC'23
Interactive App deployment and Portability

Interactive App: Address performance degradation with scaling users

- Exclusive resources
- User-selected architecture

Observations:
- Lower latency on V100 GPU node
- Improved over passenger deployment

Extending SAI Support to New HPC Software

- Two-step process:
  - Create application KG using SAI-O ontology & supported relationships
  - Add application-specific terms to HPC-ASR and dataset
  - NLU is generic to detect new entities!

- SAI provides scripts for ASR model fine-tuning & NLU performance improvement
- Modular design: KG portability across systems, simplified deployment
- Integration with Open OnDemand for easy porting to new system architectures

<table>
<thead>
<tr>
<th>Architecture /Model</th>
<th>Deployment type</th>
<th>Total latency</th>
<th>ASR module</th>
<th>NLU module</th>
<th>KG module</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDW speech</td>
<td>Interactive</td>
<td>0.4919</td>
<td>0.23865</td>
<td>0.02275</td>
<td>0.22655</td>
</tr>
<tr>
<td></td>
<td>Passenger</td>
<td>0.50245</td>
<td>0.2366</td>
<td>0.0217</td>
<td>0.2274</td>
</tr>
<tr>
<td>BDW text</td>
<td>Interactive</td>
<td>0.2665</td>
<td>0.24105</td>
<td>0.0174</td>
<td>0.1754</td>
</tr>
<tr>
<td></td>
<td>Passenger</td>
<td>0.27125</td>
<td>N/A</td>
<td>0.0218</td>
<td>0.24795</td>
</tr>
<tr>
<td>SKX speech</td>
<td>Interactive</td>
<td>0.44085</td>
<td>0.24105</td>
<td>0.0172</td>
<td>0.224</td>
</tr>
<tr>
<td></td>
<td>Passenger</td>
<td>0.22095</td>
<td>N/A</td>
<td>0.0242</td>
<td>0.19585</td>
</tr>
<tr>
<td>SKX text</td>
<td>Interactive</td>
<td>0.40735</td>
<td>0.16585</td>
<td>0.0172</td>
<td>0.224</td>
</tr>
<tr>
<td>V100 speech</td>
<td>Interactive</td>
<td>0.2664</td>
<td>N/A</td>
<td>0.0225</td>
<td>0.2433</td>
</tr>
<tr>
<td>V100 text</td>
<td>Passenger</td>
<td>0.2676</td>
<td>N/A</td>
<td>0.0225</td>
<td>0.2448</td>
</tr>
</tbody>
</table>

Compare breakdown & total latency on different architectures, 8-word text/speech query (100 speech/400 text iterations)

Source: Kousha et al, SAI: AI-Enabled Speech Assistant Interface for Science Gateways in HPC, ISC'23
SAI Demo
Concluding Remarks

• Proposed SAI, a conversational AI-enabled interface for science gateways in HPC, with Automatic Speech Recognition and Entity detection and classification model
  • Created an HPC speech and text dataset, defined a new ontology called SAI-O, and used knowledge graphs to check and validate user tasks, allowing for a general approach for any HPC application
  • Demonstrated capability by supporting three different HPC applications, and integrated SAI in Open OnDemand, deploying it on real HPC systems
  • Evaluated performance and functionality, with positive feedback from early users

• As future work we plan on releasing various components developed
  – HPC-ASR and HPC-NLU datasets
  – The retrained ASR and NLU models
  – Preform user survey
Thank You!

Panda@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/
What about ChatGPT?

• ChatGPT is a general-purpose language model. It’s versatile and it can handle a wide range of language tasks relatively well. However, it may not excel in any specialized task compared to a more dedicated tool.

• ChatGPT can be very effective in communicating with humans, but it’s far from the best when it comes to interacting with HPC systems.

• Therefore, SAI is proposed and fine-tuned with the specific objective of effective and reliable interaction between both HPC users and HPC systems.

A Swiss army knife falls short if what we need is a multi-head screwdriver!
Comparison with State of the Art: ChatGPT
Qualitative Observations:

- Sensitivity to user inputs
- Lack of consistent answers
  - Same question had different answers
  - Can lead to reproducibility issues
- Incorrect answers
  - Data is not distributed among GPUs!
  - Compilation issues

Limitations

- ChatGPT sometimes writes plausible-sounding but incorrect or nonsensical answers. Fixing this issue is challenging, as: (1) during RL training, there’s currently no source of truth; (2) training the model to be more cautious causes it to decline questions that it can answer correctly; and (3) supervised training misleads the model because the ideal answer depends on what the model knows, rather than what the human demonstrator knows.

ChatGPT is sensitive to tweaks to the input phrasing or attempting the same prompt multiple times. For example, given one phrasing of a question, the model can claim to not know the answer, but given a slight rephrase, can answer correctly.

Courtesy: https://openai.com/blog/chatgpt

```python
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=32, shuffle=True, num_workers=2)
criterion = nn.CrossEntropyLoss()
```

Screenshot of ChatGPT generated code – missing data distribution among processes
• ChatGPT produce irrelevant information
  - Provided code and extra parameters
  - Used CIFAR dataset without inquiring user

• Not best practice always!
  - Disabled InfiniBand for NCCL
  - No knowledge of best practices for performance

• Lack of specific HPC system configuration and knowledge