Galaxy: A Resource-Efficient Collaborative Edge AI System for In-situ Transformer Inference

Shengyuan Ye$^1$, Jiangsu Du$^1$, Liekang Zeng$^{1,2}$, Wenzhong Ou$^1$, Xiaowen Chu$^2$, Yutong Lu$^1$, Xu Chen$^1$

$^1$Sun Yat-sen University, $^2$HKUST(GZ)
Intelligent edge applications

- **Transformer-based** models driven increasing intelligent applications.

- Personal AI Assistants
- Smart Robotics/UAV
- AR & VR APPs

Shengyuan Ye @ School of CSE, Sun Yat-sen University
Problems of cloud-assisted approaches

- Current Transformer-based applications heavily depend on **cloud services**.

**Benefits of cloud deployment:**
- Powerful and scalable computing resources.

**Raising three game-stopping problems:**
- Data privacy concerns.
- Unreliable WAN connections.
- Network and datacenter pressure.
Problems of on-device deployment

- **On-device deployment** becomes a promising paradigm for intelligent edge APPs.

Transformer-based intelligent applications

Intelligent Edge Devices

Deploy

### TABLE I

<table>
<thead>
<tr>
<th>Model</th>
<th>DistilBert</th>
<th>Bert-L</th>
<th>GPT2-L</th>
<th>OPT-L</th>
<th>OPT-XL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nano-M</td>
<td>0.37s</td>
<td>2.43s</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>Nvidia A100</td>
<td>5ms</td>
<td>20ms</td>
<td>29ms</td>
<td>27ms</td>
<td>38ms</td>
</tr>
<tr>
<td>Memory Footprint</td>
<td>130MB</td>
<td>680MB</td>
<td>1.6GB</td>
<td>2.6GB</td>
<td>5.4GB</td>
</tr>
</tbody>
</table>

- **Check** Protect date privacy.
- **Check** Without WAN transmission.
- **Check** Limited and non-scalable on-board computing resources

121x performance gap. Out of memory error.
Opportunities

- Edge environment often comprise a rich set of trusted idle edge devices.

Smart homes usually have multiple trusted devices, such as mobile phones, laptops, and smart-home devices owned by the same family.

- Utilize nearby edge devices as resource augmentation to render expedited Transformer inference at the edge.

An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.
Challenges

Edge environment often comprise a rich set of trusted idle edge devices.

1. How to distribute sparse Transformer inference, especially **single-sample request**, across multiple edge devices?

2. How to tailor workload partitioning to the resource budget of **heterogeneous** edge devices?

3. How to reduce collaborative inference latency in **bandwidth-limited** edge environments?

An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.
Challenges

1. How to distribute sparse Transformer inference, especially **single-sample request**, across multiple edge devices.

2. How to tailor workload partitioning to the resource budget of **heterogeneous** edge devices.

3. How to reduce synchronization latency in **bandwidth-limited** edge environments.
Hybrid Model Parallelism (HMP) in Galaxy

Solution to Challenge #1: Choosing the most suitable parallelism strategy.

Inference in edge environments is often single-shot (e.g., a single voice command).

Data Parallelism and pipeline parallelism (batch-level parallelism) is unsuitable due to the inability to leverage multiple edge devices concurrently.

Model parallelism (operator-level parallelism) is suitable as it facilitates the concurrent execution of single-shot inference.

Shengyuan Ye @ School of CSE, Sun Yat-sen University
Solution to Challenge #1: Utilizing Hybrid Model Parallelism for orchestration.

A Transformer layer can be divided into three blocks: MHA, MLP, and CONN.
Hybrid Model Parallelism (HMP) in Galaxy

- **Solution to Challenge #1**: Utilizing **Hybrid Model Parallelism** for orchestration.

- **Tensor Model Parallelism**

- **Sequence Model Parallelism**

  - **Tensor model parallelism** can only be applied to **MHA and MLP blocks**, requiring an tensor synchronization at the end of each block.

  - **Sequence model parallelism** can be applied to **Connection blocks**, which are element-wise operations and require no additional communication.
Hybrid Model Parallelism (HMP) in Galaxy

- **Solution to Challenge #1**: Utilizing hybrid model parallelism for orchestration.
  - Using **Tensor Model Parallelism** for a Transformer layer.

An instance of tensor model parallelism across two edge devices

Shengyuan Ye @ School of CSE, Sun Yat-sen University
Hybrid Model Parallelism (HMP) in Galaxy

- **Solution to Challenge #1**: Utilizing hybrid model parallelism for orchestration.
  - Using **Tensor Model Parallelism** for MHA and MLP blocks, and **Sequence Model Parallelism** for connection blocks.

An instance of hybrid model parallelism across two edge devices.
**Solution to Challenge #1:** Utilizing hybrid model parallelism for orchestration.

**Opportunities:** Split the original heavy AllReduce into two smaller AllGather and ReduceScatter communications, which provides opportunities for overlapping communication and computation.

An instance of hybrid model parallelism across two edge devices.
Challenges

Edge environment often comprise a rich set of trusted idle edge devices.

1. How to distribute sparse Transformer inference, especially single-sample request, across multiple edge devices.

2. How to tailor workload partitioning to the resource budget of heterogeneous edge devices.

3. How to reduce synchronization latency in bandwidth-limited edge environments.
Resource-Aware Workload Partitioning

**Solution to Challenge #2**: Heterogeneity and Memory Aware Workload Planning.

The initiation of model parallel inference is bound by the completion time of the slowest device (*straggler*)!

- MHA blocks partition results: \( A = \{a_0, a_1, \ldots, a_{D-1}\} \)
- MLP blocks partition results: \( B = \{b_0, b_1, \ldots, b_{D-1}\} \)
- CONN blocks partition results: \( S = \{s_0, s_1, \ldots, s_{D-1}\} \)

**Determined the straggler**

\[
\begin{align*}
\mathcal{L}(\text{MHA}, \mathcal{A}) &= \max_{d \in \{0,1,\ldots,D-1\}} L(\text{MHA}, A_d, d), \\
\mathcal{L}(\text{MLP}, \mathcal{B}) &= \max_{d \in \{0,1,\ldots,D-1\}} L(\text{MLP}, B_d, d), \\
\mathcal{L}(\text{CONN}, \mathcal{S}) &= \max_{d \in \{0,1,\ldots,D-1\}} L(\text{CONN}, S_d, d).
\end{align*}
\]

**Optimization objective**

\[
\min_{\mathcal{A}, \mathcal{B}, \mathcal{S}} \left( \mathcal{L}(\text{MHA}, \mathcal{A}) + \mathcal{L}(\text{MLP}, \mathcal{B}) + \mathcal{L}(\text{CONN}, \mathcal{S}) \right),
\]

subject to

\[
l \cdot \left( M_{\text{att}} \cdot \frac{a_d}{\sum A} + M_{\text{mlp}} \cdot \frac{b_d}{\sum B} \right) < \text{Budget}_d,
\]

where \( d \in \{0,1,\ldots,D-1\} \).
Solution to Challenge #2: Heterogeneity and Memory Aware Workload Planning.

We have designed a lightweight two-step greedy heuristic algorithm.

```
Algorithm 1: Heterogeneity and Memory Aware Workload Planning

Input: Profiling results of models and devices. \( \mathcal{V} \): The list of computing capacity of devices.
Output: \( \mathcal{A}, \mathcal{B} \): Partition configurations of MHA and MLP block.

Function BalancedPartition \( (T, \mathcal{V}) \):
1. Initialize partition configuration \( C \);
2. Workload \( \leftarrow \) Total workload in block \( T \);
3. foreach \( d \in \{0, 1, 2, ..., D - 1\} \) do
4. \( C_d \leftarrow (\mathcal{V}_d / \sum \mathcal{V}) \cdot \) Workload;
5. Return \( C \);
6. \( \mathcal{A} \leftarrow \) BalancedPartition \( (MHA, \mathcal{V}) \);
7. \( \mathcal{B} \leftarrow \) BalancedPartition \( (MLP, \mathcal{V}) \);
8. Function MemoryAwareBalancing \( (T, C, \mathcal{V}, \mathcal{L}) \):
9. \( OOM\_Devices \leftarrow \) Out-of-memory devices under
10. partition configuration \( C \) in \( \mathcal{L} \);
11. \( Free\_Devices \leftarrow \) Devices retaining available
memory under partition config. \( C \) in \( \mathcal{L} \);
12. if \( OOM\_Devices = \emptyset \) then
13. \( \text{Return } C; \)
14. foreach \( o \in OOM\_Devices \) do
15. \( \text{Waiting}_\text{Shift} \leftarrow \) Overflowing workload on device \( o \);
16. foreach \( f \in Free\_Devices \) do
17. \( \text{Shift} \leftarrow (\mathcal{V}_f / \sum_{i \in Free\_Devices} \mathcal{V}_i) \cdot \text{Waiting}_\text{Shift} \)
workload from \( o \) to \( f \);
18. Remove device \( o \) from \( \mathcal{L} \);
19. \( \text{MemoryAwareBalancing} (T, C, \mathcal{V}, \mathcal{L}); \)
20. \( \mathcal{L} \leftarrow [0, 1, ..., D - 1]; \) \( \triangleright \) List of all devices
21. \( \mathcal{B} \leftarrow \) MemoryAwareBalancing \( (MLP, \mathcal{B}, \mathcal{V}, \mathcal{L}); \)
22. \( \mathcal{A} \leftarrow \) MemoryAwareBalancing \( (MHA, \mathcal{A}, \mathcal{V}, \mathcal{L}); \)
23. if Out-of-memory devices still exist then
24. \( \text{Exit with Fail}; \)
```

Shengyuan Ye @ School of CSE, Sun Yat-sen University
Challenges

1. How to distribute sparse Transformer inference, especially single-sample request, across multiple edge devices.

2. How to tailor workload partitioning to the resource budget of heterogeneous edge devices.

3. How to reduce synchronization latency under bandwidth-limited edge environments.

Edge environment often comprise a rich set of trusted idle edge devices.

An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.
Tile-based Communication Optimization

Each Transformer layer requires 4 tensor synchronization points.

GEMM: General Matrix Multiply
Overlapping communication and computation is an effective optimization strategy.

Tile-based Communication Optimization

AllGather-GEMM overlapping
ReduceSactter-GEMM overlapping
AllGather-GEMM overlapping
ReduceSactter-GEMM overlapping
Tile-based Communication Optimization

**Ring-AllGather Overlapping**

1. Decouple the data dependency between synchronization and GEMM by partitioning the matrix into submatrices.

   \[ E_i = \left[ \frac{H_0 \cdot W_i^D}{H_1 \cdot W_i^D} \right] = \left[ \frac{H_0}{H_1} \right] \cdot W_i^D = D \cdot W_i^D. \]

2. We start the GEMM computation for each submatrix immediately after its synchronization, avoiding the need to wait for the entire matrix.

An illustration of Ring-AllGather overlapping across three edge devices.
Tile-based Communication Optimization

Ring-ReduceScatter Overlapping

1. Decouple the data dependency between synchronization and GEMM by **partitioning the matrix into submatrices**.

\[
\begin{bmatrix}
O_{i,0} \\
O_{i,1} \\
O_{i,2}
\end{bmatrix}
= \begin{bmatrix}
E_{i,0} \cdot W_i^E \\
E_{i,1} \cdot W_i^E \\
E_{i,2} \cdot W_i^E
\end{bmatrix}
= \begin{bmatrix}
E_{i,0} \\
E_{i,1} \\
E_{i,2}
\end{bmatrix}
\cdot W_i^E
= E_i \cdot W_i^E,
\]

2. We start the GEMM computation for each submatrix immediately after its synchronization, **avoiding the need to wait for the entire matrix**.

An illustration of Ring-ReduceScatter overlapping across three edge devices.
Putting It All Together

- **Galaxy system workflow.**

1. **Preprocessing phase:**
   - **Galaxy Profiler** records run-time traces needed for planning using calibration data on edge devices.

2. **Parallelism Planning Phase:**
   - **Galaxy Planner** takes profiling results from Galaxy Profiler as input to generate a parallelism planning configuration.

3. **Execution Phase:**
   - **Galaxy Runtime** applies the planning configuration to target models and edge devices for efficient edge collaborative inference.

Shengyuan Ye @ School of CSE, Sun Yat-sen University
Evaluation

● Testbeds

- Off-the-shelf edge devices: NVIDIA Jetson Nano.
- Simulate **three** heterogeneous computing devices by adjusting the SoC frequency.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Quad Core ARM Cortex-A53 CPU</td>
</tr>
<tr>
<td>GPU</td>
<td>128 Core Maxwell GPU</td>
</tr>
<tr>
<td>CPU Frequency Mode</td>
<td>Nano-S 403MHz</td>
</tr>
<tr>
<td></td>
<td>Nano-M 825MHz</td>
</tr>
<tr>
<td></td>
<td>Nano-L 1.47GHz</td>
</tr>
</tbody>
</table>
Evaluation

● Testbeds

- Using these 3 heterogeneous devices, we simulated **6 different edge clusters**, including both homogeneous and heterogeneous clusters.

<table>
<thead>
<tr>
<th>ID</th>
<th>Homogeneous Edge Env.</th>
<th>ID</th>
<th>Heterogeneous Edge Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2 × Nano-M</td>
<td>D</td>
<td>Nano-L + Nano-M</td>
</tr>
<tr>
<td>B</td>
<td>3 × Nano-M</td>
<td>E</td>
<td>Nano-L + Nano-S</td>
</tr>
<tr>
<td>C</td>
<td>4 × Nano-M</td>
<td>F</td>
<td>Nano-L + Nano-M + Nano-S</td>
</tr>
</tbody>
</table>

● Models and datasets

- 5 prevalent Transformer-based models: DistilBert, Bert, GPT2-L, OPT-L, OPT-XL.
- Evaluate with the input sequences from GLUE dataset.
Evaluation

- **Baselines**
  - Sequence Parallelism (SP) [25]: A state-of-the-art sequence model parallelism method

Maintained high performance **across various network environments**, with up to **46% latency reduction** compared to baseline methods!!!
Evaluation

- System performance under **heterogeneous edge environments**.
  - Galaxy consistently and remarkably outperforms other parallelism methods in heterogeneous edge environments, reducing inference latency by **1.3x to 2.5x**.

Galaxy achieves **86% linear scaling** with parallel inference on 4 Nvidia Jetson Nano devices.
Eco: An Edge Collaborative AI framework for serving miscellaneous AI model at the edge.

We aim to design affordable, accessible, and adaptive AI with your private group of mobile and edge devices.

https://collaborative-edge-ai.github.io/

Features

- **Optimized Computation**
  - Language models
  - Vision perceptrons
  - Graph nets

- **Heterogeneity Awareness**
  - Mobile phones
  - Embedded devices
  - Edge servers

- **Resilient Elasticity**
  - Device breakdown
  - Load variation
  - Bandwidth fluctuation
Thanks for listening

Shengyuan Ye¹, Jiangsu Du¹, Liekang Zeng¹,², Wenzhong Ou¹, Xiaowen Chu², Yutong Lu¹, Xu Chen¹

¹Sun Yat-sen University, ²HKUST(GZ)