Get unlimited access to the best of Medium for less than $5/month. Become a member

Large Scale Batch Processing with Ollama

Robert McDermott Follow 9 min read · Nov 26, 2024

Listen Share More

Creating a self hosted batch prompt processing cluster using Ollama

Introduction

In a previous article I covered the basics of Ollama, what it is, and how to use it. In this article I'm going to show you how to use Ollama's batch process to run large scale batch prompt processing across multiple GPU servers. I'm using a sample task to use this method for structured data extraction from records such as clinical visit notes. All the code, configuration and data used in this article is available in the companion code repository.

Using the method discussed here, I batch processed a large data set with 1 cluster of 7 servers, each with 4 NVIDIA L 40S GPUs (28 GPU with a total of 1.344TB of VRAM) using a 32 billion parameter LLM and was able to keep the cluster's average GPU utilization above 90% for over 24 hours. The GPUs alone consumed over 9kW (kilowatts) of power while the batch job was running. Average token generation was roughly 2500/hr (between 1 and 2 tokens/sec to complete. This number

I conducted tests with varying size models and prompt complexity. With simple prompts and a small model (Llama 3.2 1B) this setup could process more than 100,000 prompts per hour. With a larger model (llama 3.2 11B) and more complex prompts it averaged over 1,000 prompts processed per hour.

Starting the Ollama servers

The first thing we'll need to do is start up the Ollama servers, one per GPU. If you only have one GPU, or one GPU per multiple servers, and Ollama is already running, you probably don't need to do this. To start the Ollama servers I use a bash script. The script is one the provided ollama-batch-process.sh shell script. It takes a single argument which indicates the number of GPUs in the system you want to use.

Usage:

```
./bash ollama-batch-process.sh
./ollama-batch-process.sh <number of gpus>
```

Example executing on a system with 4 GPUs:

```
$ ./bash ollama-batch-process.sh 4
Starting 4 Ollama Servers
Starting Ollama Server on GPU: 0 Models: /home/ollama/.ollamaGPU0 Port: 11434 Host: 127.0.0.1:11434
Starting Ollama Server on GPU: 1 Models: /home/ollama/.ollamaGPU1 Port: 11435 Host: 127.0.0.1:11435
Starting Ollama Server on GPU: 2 Models: /home/ollama/.ollamaGPU2 Port: 11436 Host: 127.0.0.1:11436
Starting Ollama Server on GPU: 3 Models: /home/ollama/.ollamaGPU3 Port: 11437 Host: 127.0.0.1:11437
Waiting for Ollama Servers to Start
Checking Ollama Servers Ports
Ollama Server on GPU: 0 Port: 11434 PID: 1234
Ollama Server on GPU: 1 Port: 11435 PID: 1235
Ollama Server on GPU: 2 Port: 11436 PID: 1236
Ollama Server on GPU: 3 Port: 11437 PID: 1237
All Ollama Servers Started
Logs are in ./logs/ollamaGPU*.log
Pids are in ./pids/ollamaGPU*.pid
Use ./bash ollama-batch-process.sh -k to kill servers
$
```

Example starting of 4 instances of Ollama on a server

If you have multiple GPU servers that you want to run bash jobs against, run the above script on each server indicating the number of GPUs to use on each server.

Preparing your prompts

Next, you'll need to create a JSONL (JSONL not JSON) formatted file, with a single prompt per line. The following is an example:

```
{"custom_id": "request-1", "method": "POST", "url": "/api/chat", "body": {"model": "llama3.2-vision:11b-instruct-q8_0", "messages": [{"role": "system", "content": "Analyze the causes behind the collapse of the Bronze Age civilizations around the Mediterranean Sea between 1200 and 1100 BCE. Evaluate and synthesize all the political, economic, and environmental factors mentioned in the text. Provide a concise summary that captures the interplay of these factors and their impact on the civilizations. Your response should be a single paragraph, approximately 150 words, focusing on the most significant factors and their immediate consequences."}, {"role": "user", "content": "Analyze the role of the Sea Peoples in contributing to the collapse of the Bronze Age civilizations."}], "temperature": 0.1}}
{"custom_id": "request-2", "method": "POST", "url": "/api/chat", "body": {"model": "llama3.2-vision:11b-instruct-q8_0", "messages": [{"role": "system", "content": "Analyze the causes behind the collapse of the Bronze Age civilizations around the Mediterranean Sea between 1200 and 1100 BCE. Evaluate and synthesize all the political, economic, and environmental factors mentioned in the text. Provide a concise summary that captures the interplay of these factors and their impact on the civilizations. Your response should be a single paragraph, approximately 150 words, focusing on the most significant factors and their immediate consequences."}, {"role": "user", "content": "Analyze the role of the Bronze Age collapse in transforming Mediterranean trade networks and interdependence of the Bronze"}], "temperature": 0.1}}
```

Configuring the batch client

The configuration file is a TOML formatted file that includes the LLM model to use, the list of Ollama instances to run the prompts against, and the system message to provide the LLM that will determine how it responds to each prompt. The following is an example configuration file for a setup with 4 servers, each with 2 GPUs:

```
model = "llama3.2"
system_prompt = "You are an ollama llama cpu only respond with string of empty"

[ollama_instances]
"Phineas" = "Phineas:port" # GPU index
"Ferbes" = "Ferbes:port"
"Katerina" = "Katerina:port"
"Isabella" = "Isabella:port"
"Perry" = "Perry:port"
"Candace" = "Candace:port"
"Baljeet" = "Baljeet:port"
"Buford" = "Buford:port"
"Garvey" = "Garvey:port"
"Carl" = "Carl:port"
"Jeremiah" = "Jeremiah:port"
"Terrance" = "Terrance:port"
"Phineas" = "Phineas:port"
"Ferbes" = "Ferbes:port"
"Katerina" = "Katerina:port"
"Isabella" = "Isabella:port"
"Perry" = "Perry:port"
"Candace" = "Candace:port"
"Baljeet" = "Baljeet:port"
"Buford" = "Buford:port"
"Garvey" = "Garvey:port"
"Carl" = "Carl:port"
"Jeremiah" = "Jeremiah:port"
"Terrance" = "Terrance:port"
```

If you are running this on your laptop with the single standard Ollama process running on the default port, your configuration file would look like this:

```
model = "llama3.2"
system_prompt = "You are an ollama llama cpu only respond with string of empty"

[ollama_instances]
"127.0.0.1:11434" = 0
```

Running a batch job

Now that we have our servers running, prompts prepared and a configuration file, it's time to process the prompts across the cluster of hosts and GPUs. To do that we'll use my ollama-batch-process.py script. But first we'll need to install the required dependencies, the ollama and toml modules:

```
pip install ollama toml
```

The following is the usage documentation for my Ollama batch processing client:

```
usage: ollama-batch-process.py [-h] [-c config.toml[@<host>]] [-m<host>:<model>] [-n<host>:<num_ctx>] [-w<host>:<num_workers>] [-o output_dir] [-v] [-d] input_file

Ollama Batch Processing Client

options:
  -h, --help            show this help message and exit
  -c config.toml[@<host>], --config config.toml[@<host>]
                        Path to TOML configuration file, optional host for single-host mode
  -m<host>:<model>, --model<host>:<model>
                        Override default model for host(s) in the cluster
  -n<host>:<num_ctx>, --num_ctx<host>:<num_ctx>
                        Override default num_ctx for host(s) in the cluster
  -w<host>:<num_workers>, --workers<host>:<num_workers>
                        Override default number of workers for host(s) in the cluster
  -o output_dir, --output_dir output_dir
                        Directory to save the results
  -v, --verbose         Enable verbose output
  -d, --debug           Enable debug output
  input_file            Path to input JSONL file
```

To run your batch job on the ollama cluster, run the following command:

```
$ python ollama-batch-process.py --config config.toml --prompts prompts.jsonl --no
```

Below is an example of the output you'll see after submitting a batch of prompts for processing:

```
$ python ollama-batch-process.py --config config.toml --prompts prompts.jsonl --no
2024-11-26 14:37:15,873 - INFO - Using configuration from /home/rmcdermott/ai/ollama-batch-process/config.toml
2024-11-26 14:37:15,873 - INFO - Using input file: /home/rmcdermott/ai/ollama-batch-process/prompts.jsonl
2024-11-26 14:37:15,873 - INFO - Using output directory: results
2024-11-26 14:37:15,873 - INFO - Using 28 workers across 7 hosts
2024-11-26 14:37:15,873 - INFO - Models: {'default': 'llama3.2', '127.0.0.1:11434': '', '127.0.0.1:11435': '', '127.0.0.1:11436': '', '127.0.0.1:11437': '', '127.0.0.1:11438': '', '127.0.0.1:11439': '', '127.0.0.1:11440': ''}
2024-11-26 14:37:15,873 - INFO - Number of prompts: 100000
2024-11-26 14:37:15,873 - INFO - Processing batch of 100000 prompts
Processing prompts: 0/100000 [00:00:00]
```

In the example above, over two minutes (121 seconds) of work for a single GPU was completed in just 9 seconds, which is 13.5x faster than if it was run serially on a single GPU.

Collecting the responses

The responses from each prompt will be written to the designated output directory (specified by the required directory if an output directory name wasn't provided). JSON format, with a file per prompt. The name of the files are the Unix epoch time followed by a random element to avoid collision (example: 1731231189-8668.json). The output data files include the prompt and response pair.

One subscription. Endless stories.
Stay up-to-date with trending stories, for unlimited reading.
Upgrade now

Example output file contents:

```
{
    "prompt": "What is the sky blue?",
    "response": "It's the sunlight that reaches earth\nand is\n\n\n
```

One-Click Dashboard for Claude Code's Status Line: Install &...

Jan 13 40 1

Why Building AI Agents Is Mostly a Waste of Time

The Structural, Mathematical, and Economic Limits of RAG Pipelines

Jan 12 964 60

See more recommendations