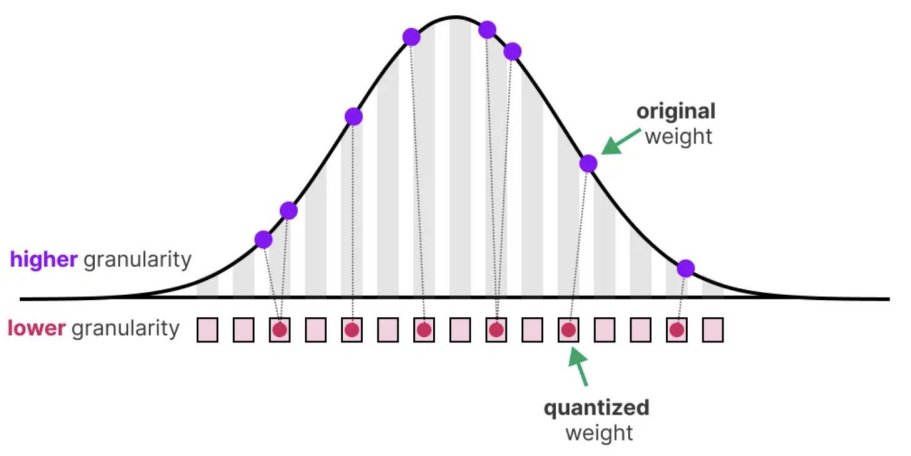
Quantization in vLLM
From Zero to Hero

Eldar Kurtić

Max Dargatz

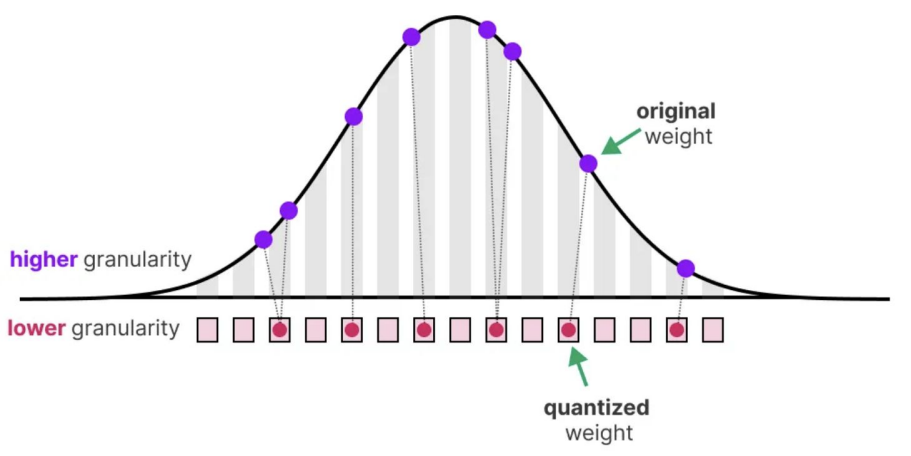
June 3rd, 2025

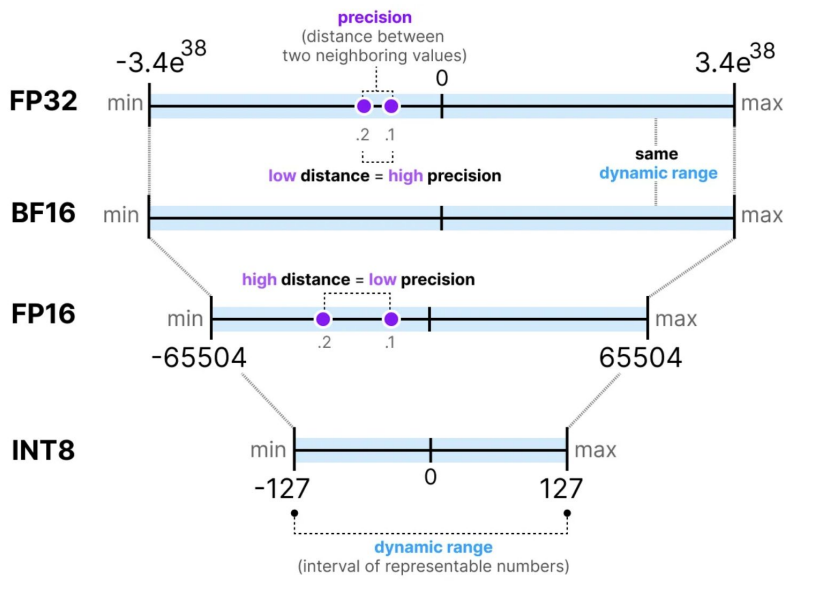
What is quantization?



Source: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization

What is quantization?





Source: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization

Why quantization matters?

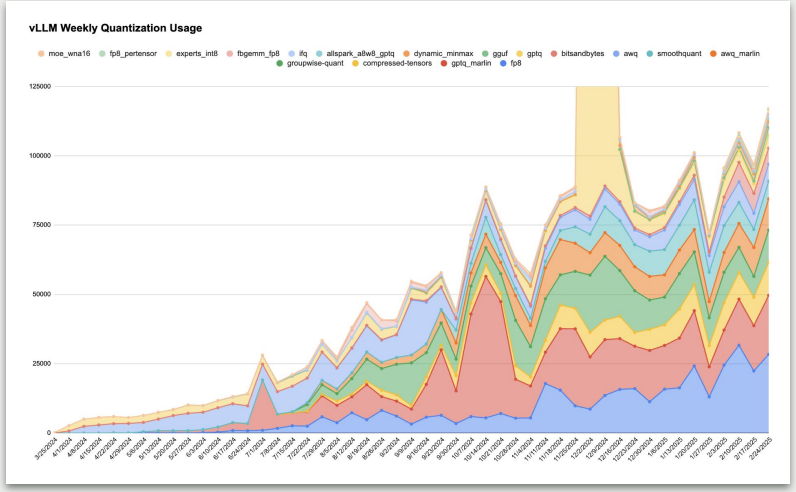
- Reduce memory footprint → lower deployment costs

Why quantization matters?

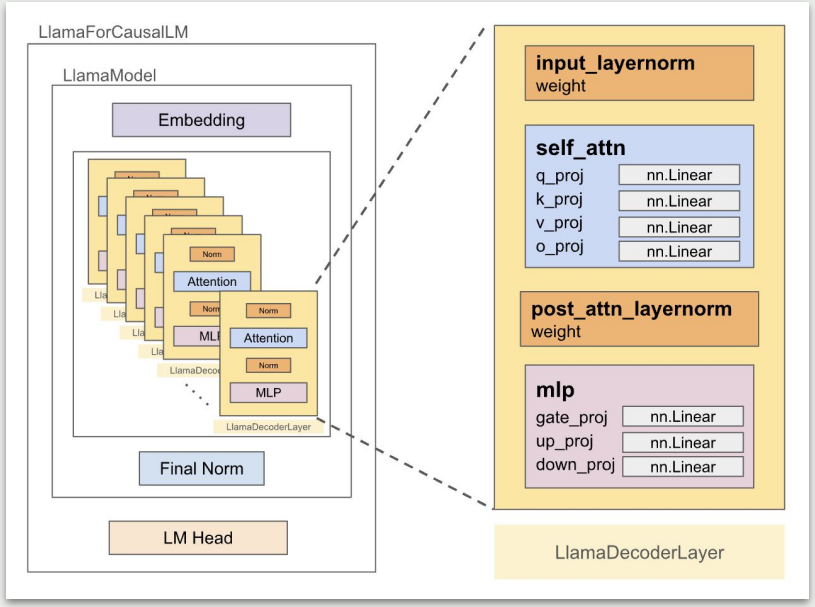
- **Reduce memory footprint → lower deployment costs**
- **Improve inference efficiency → faster forward passes**
  - Via faster weight loading (less data to move around)
  - Via faster matrix-multiplications (lower precision compute)

Why quantization matters?

- **Reduce memory footprint → lower deployment costs**
- **Improve inference efficiency → faster forward passes**
  - Via faster weight loading (less data to move around)
  - Via faster matrix-multiplications (lower precision compute)
- **Real-world adoption of quantized models is steadily increasing**
  - Was 20%, now getting closer to 40%

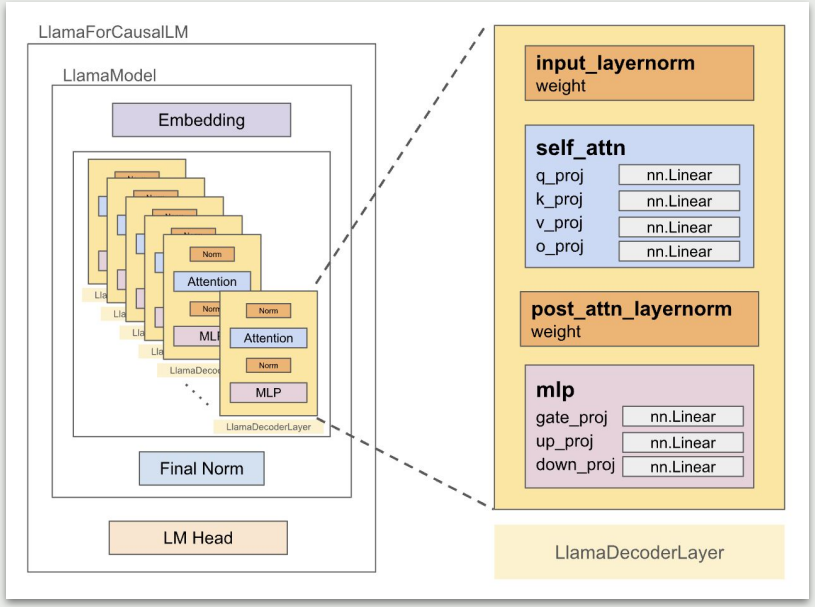


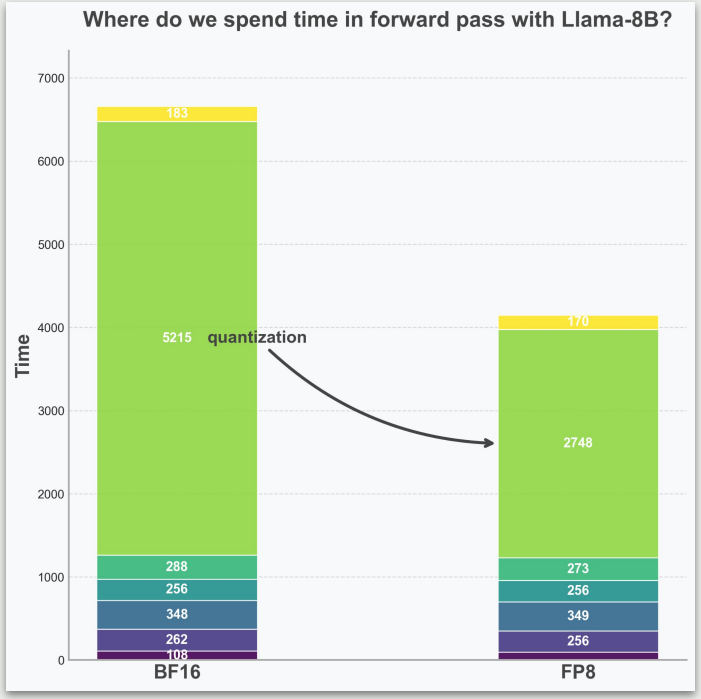
Where does quantization fit inside an LLM?



https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/te_llama/tutorial_accelerate_hf_llama_with_te.html

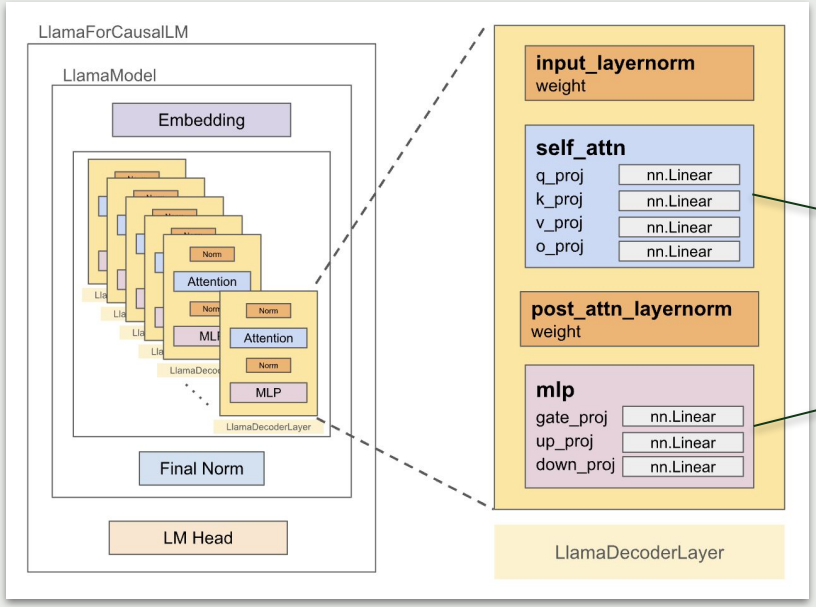
Where does quantization fit inside an LLM?

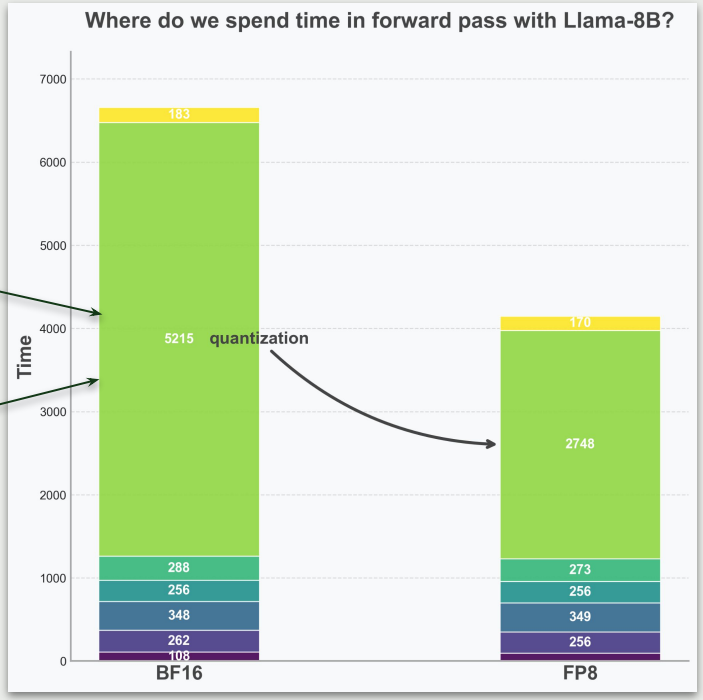




https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/te_llama/tutorial_accelerate_hf_llama_with_te.html

Where does quantization fit inside an LLM?

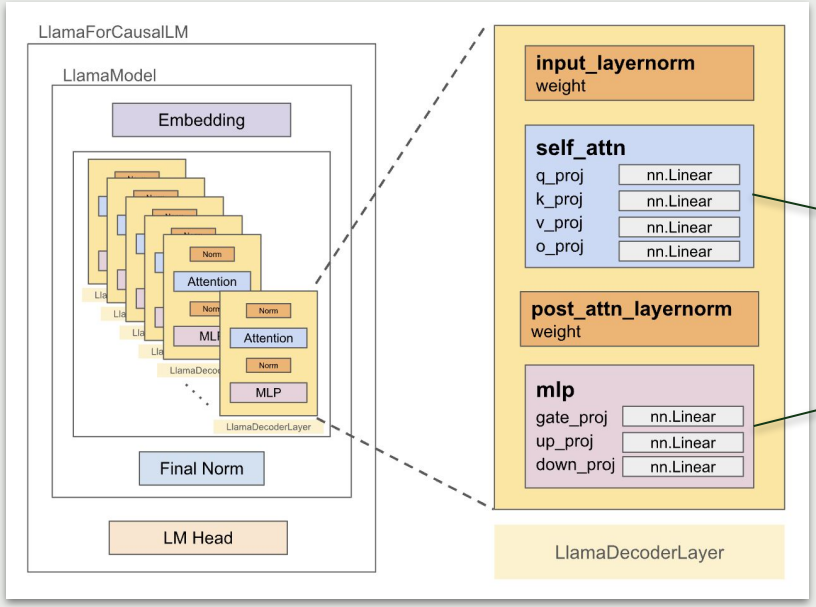


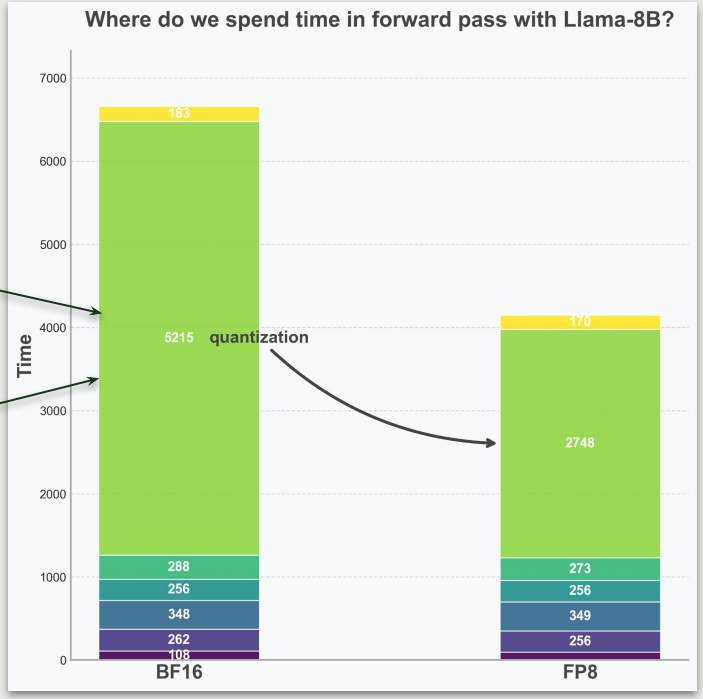


https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/te_llama/tutorial_accelerate_hf_llama_with_te.html

* at very long context lengths, attention changes these trade-offs

Where does quantization fit inside an LLM?



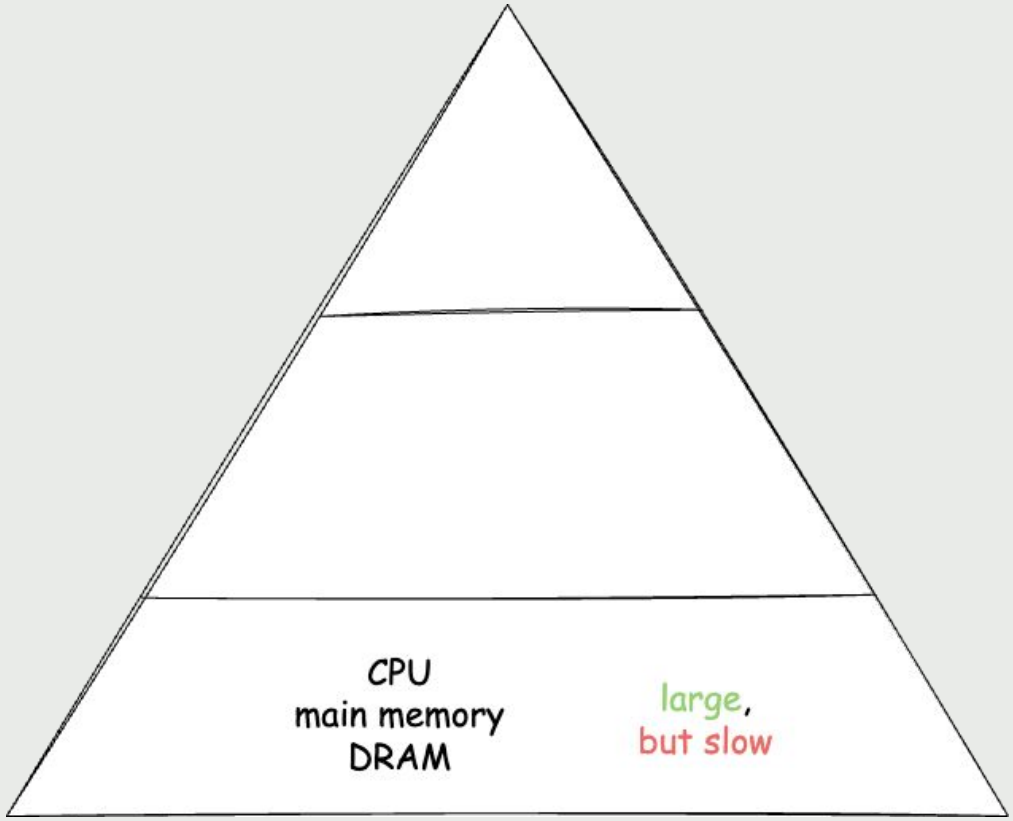


Takeaway: We really want to quantize torch.nn.Linear!

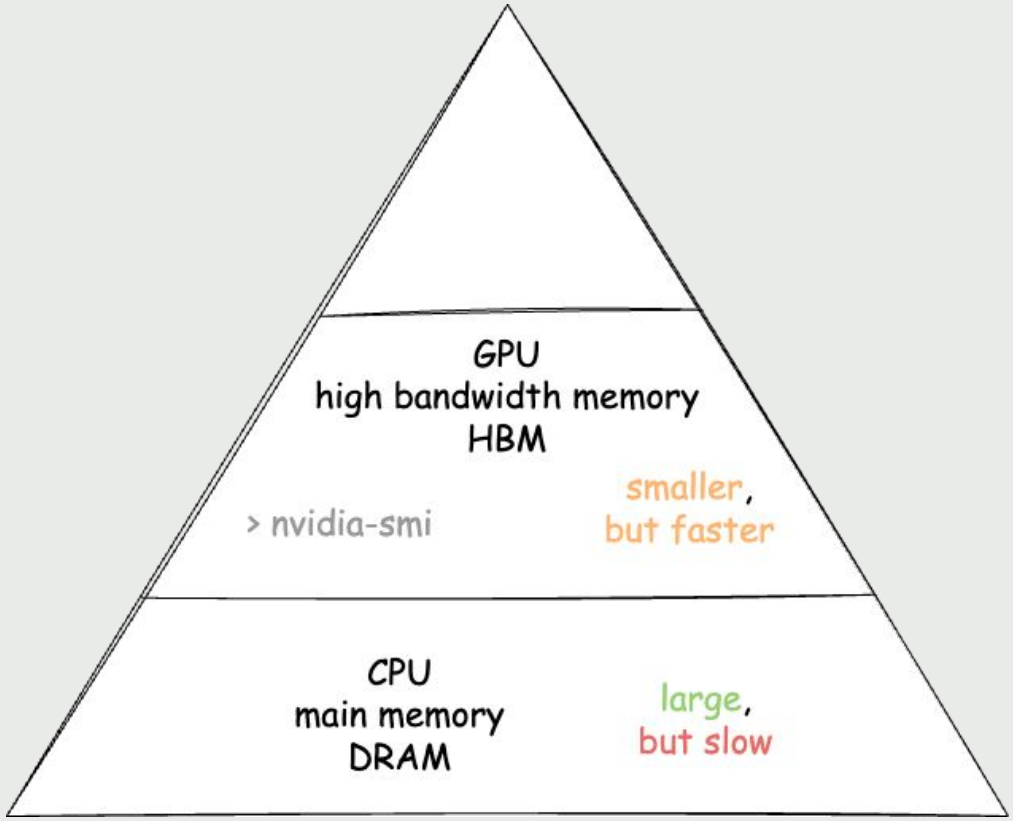
https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/te_llama/tutorial_accelerate_hf_llama_with_te.html

* at very long context lengths, attention changes these trade-offs

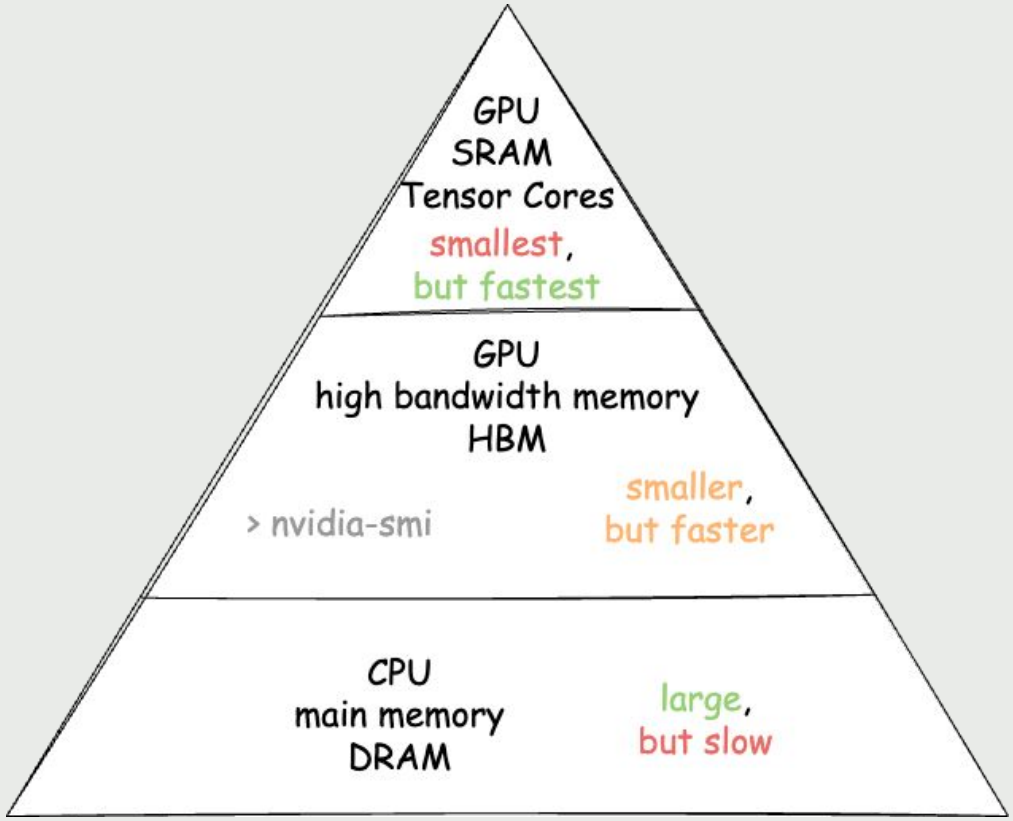
Where does quantization fit inside an LLM?



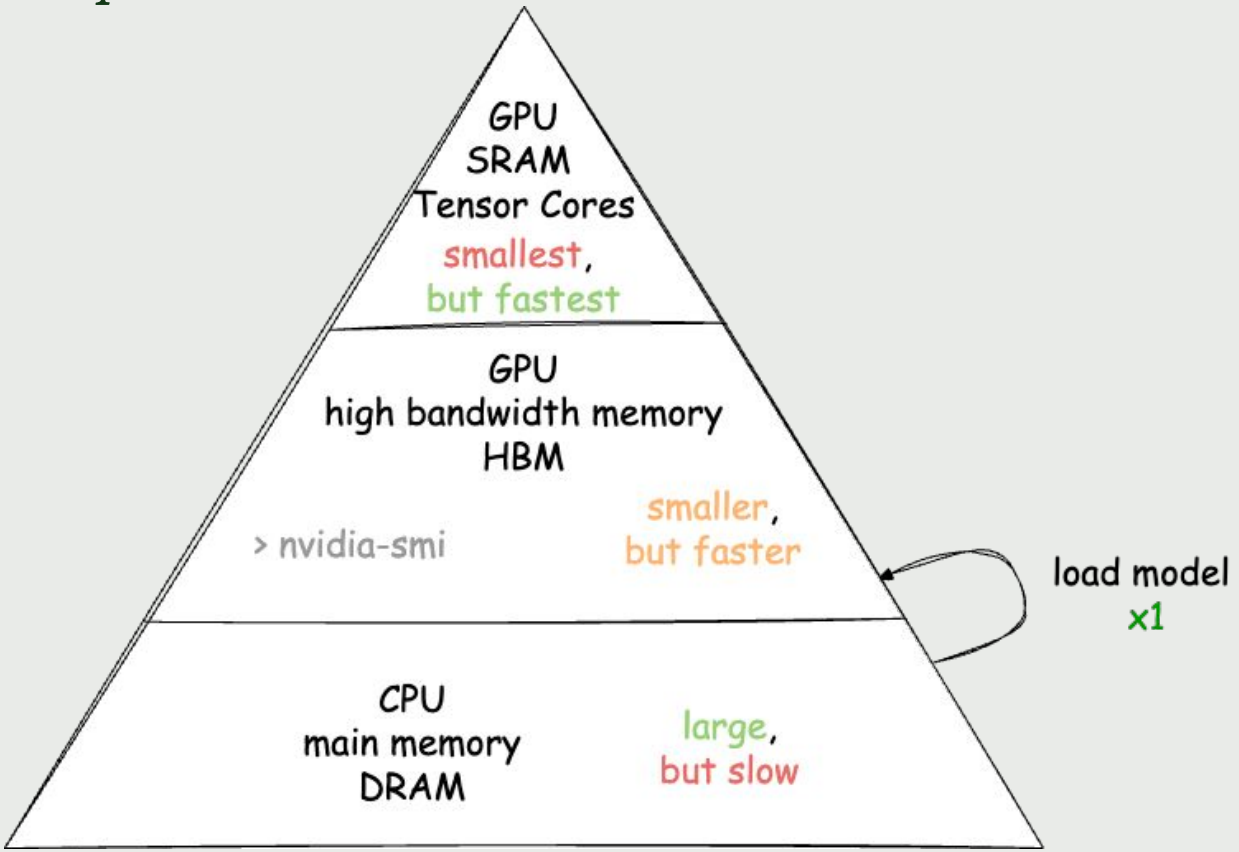
Where does quantization fit inside an LLM?



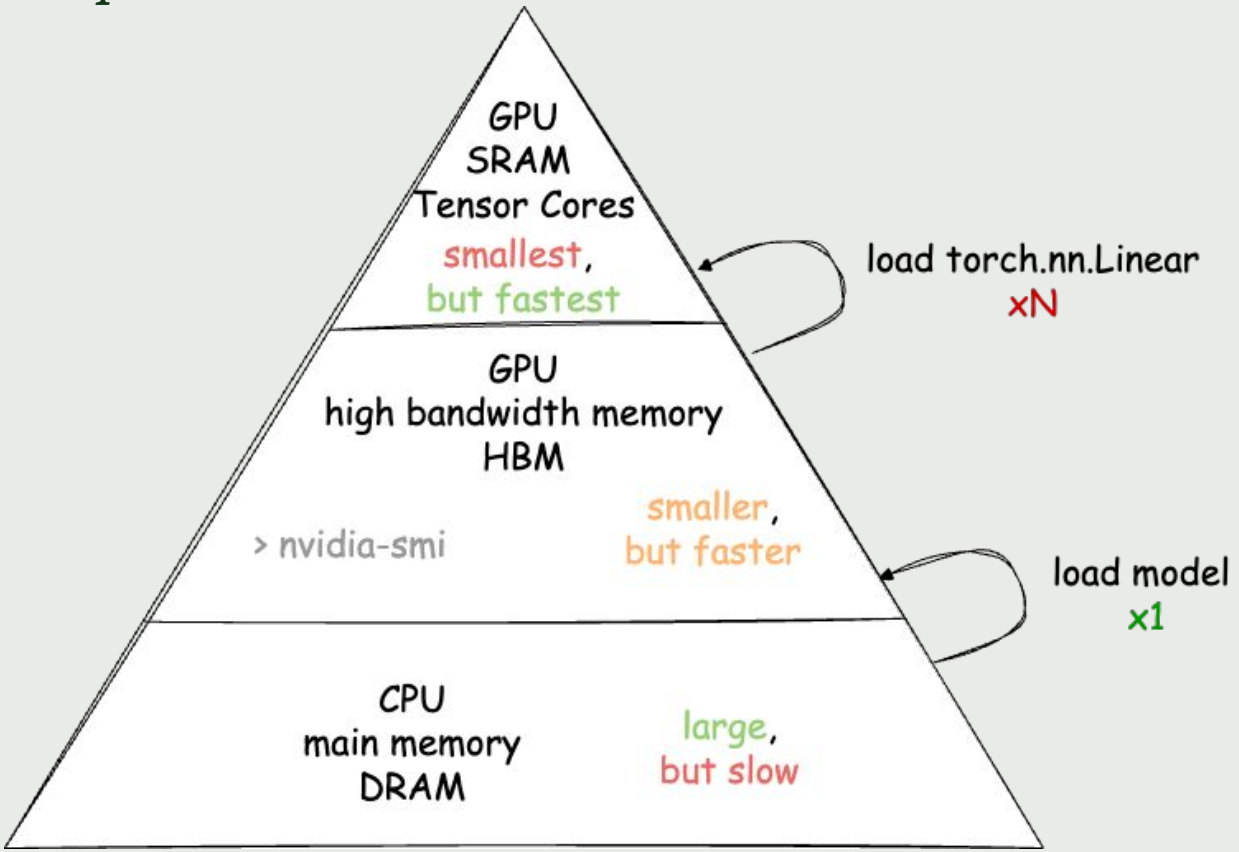
Where does quantization fit inside an LLM?



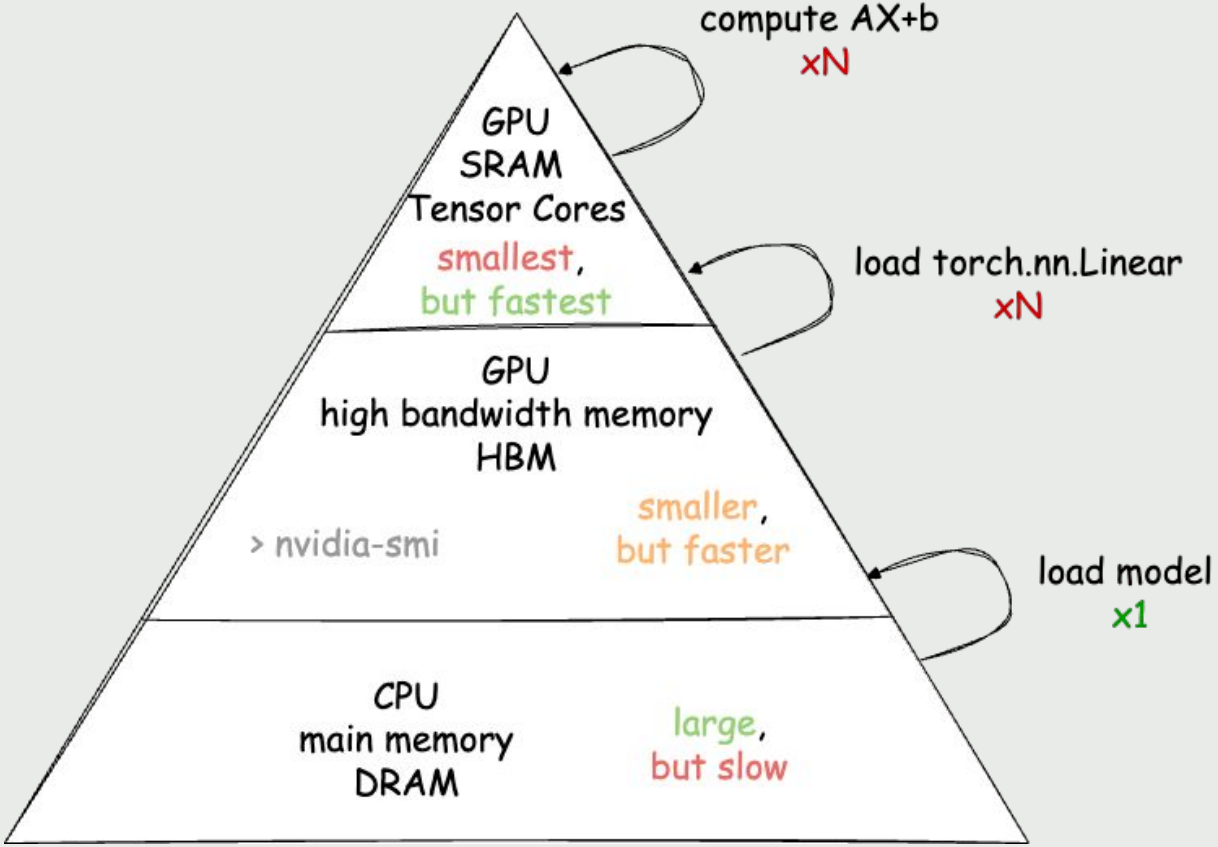
Where does quantization fit inside an LLM?



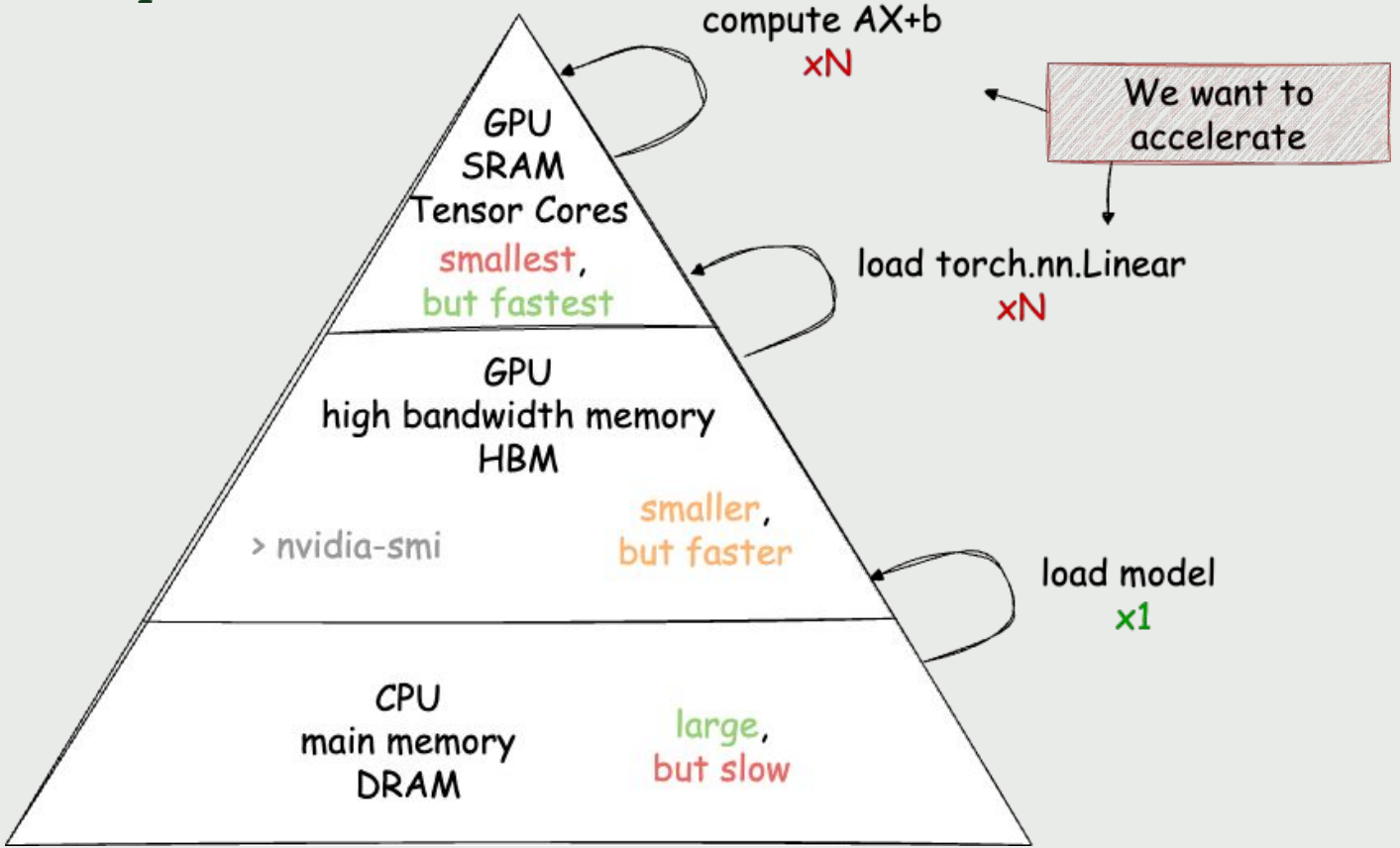
Where does quantization fit inside an LLM?



Where does quantization fit inside an LLM?



Where does quantization fit inside an LLM?



Where does quantization fit inside an LLM?

1) *“Load torch.nn.Linear”* – How to accelerate loading?

Where does quantization fit inside an LLM?

1) *“Load torch.nn.Linear”* – How to accelerate loading?

- Reduce the number of bits to load.

Where does quantization fit inside an LLM?

1) *“Load torch.nn.Linear”* – How to accelerate loading?

- Reduce the number of bits to load. → Solution: **weight quantization**.

Where does quantization fit inside an LLM?

1) *“Load torch.nn.Linear”* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *“Compute AX+b”* – How to accelerate compute?

Where does quantization fit inside an LLM?

1) *“Load torch.nn.Linear”* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *“Compute AX+b”* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Where does quantization fit inside an LLM?

1) *"Load torch.nn.Linear"* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *"Compute AX+b"* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Technical Specifications	
	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core*	989 teraFLOPS
BFLOAT16 Tensor Core*	1,979 teraFLOPS
FP16 Tensor Core*	1,979 teraFLOPS
FP8 Tensor Core*	3,958 teraFLOPS
INT8 Tensor Core*	3,958 TOPS

Where does quantization fit inside an LLM?

1) *"Load torch.nn.Linear"* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *"Compute AX+b"* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Technical Specifications	
	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core*	989 teraFLOPS
BFLOAT16 Tensor Core*	1,979 teraFLOPS
FP16 Tensor Core*	1,979 teraFLOPS
FP8 Tensor Core*	3,958 teraFLOPS
INT8 Tensor Core*	3,958 TOPS

Where does quantization fit inside an LLM?

1) *"Load torch.nn.Linear"* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *"Compute AX+b"* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Technical Specifications	
	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core*	989 teraFLOPS
BFLOAT16 Tensor Core*	1,979 teraFLOPS
FP16 Tensor Core*	1,979 teraFLOPS
FP8 Tensor Core*	3,958 teraFLOPS
INT8 Tensor Core*	3,958 TOPS

Where does quantization fit inside an LLM?

1) *"Load torch.nn.Linear"* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *"Compute AX+b"* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Technical Specifications	
	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core*	989 teraFLOPS
BFLOAT16 Tensor Core*	1,979 teraFLOPS
FP16 Tensor Core*	1,979 teraFLOPS
FP8 Tensor Core*	3,958 teraFLOPS
INT8 Tensor Core*	3,958 TOPS

Where does quantization fit inside an LLM?

1) *"Load torch.nn.Linear"* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *"Compute AX+b"* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Technical Specifications	
	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core*	989 teraFLOPS
BFLOAT16 Tensor Core*	1,979 teraFLOPS
FP16 Tensor Core*	1,979 teraFLOPS
FP8 Tensor Core*	3,958 teraFLOPS
INT8 Tensor Core*	3,958 TOPS

→ Solution: lower precision tensor cores via **weight-and-activation quantization.**

Where does quantization fit inside an LLM?

1) *"Load torch.nn.Linear"* – How to accelerate loading?
- Reduce the number of bits to load. → Solution: **weight quantization**.

2) *"Compute AX+b"* – How to accelerate compute?
- Compute happens in tensor cores. We need faster tensor cores.

Technical Specifications	
	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core*	989 teraFLOPS
BFLOAT16 Tensor Core*	1,979 teraFLOPS
FP16 Tensor Core*	1,979 teraFLOPS
FP8 Tensor Core*	3,958 teraFLOPS
INT8 Tensor Core*	3,958 TOPS

→ Solution: lower precision tensor cores via **weight-and-activation quantization.**

Hopper, Ada Lovelace, and newer

Ampere and older

Let’s see how to **actually** quantize some LLMs.

Meet LLM-Compressor: github.com/vllm-project/llm-compressor

- Quantization (and pruning) for both: practitioners and researchers.

Meet LLM-Compressor:
github.com/vllm-project/llm-compressor

- Quantization (and pruning) for both: practitioners and researchers.
- **For practitioners**: easy to use recipes with heavily optimized default configurations for all quantization formats supported in vLLM (INT W8A8, INT W8A16, INT W4A16, FP W8A8).

Meet LLM-Compressor:
github.com/vllm-project/llm-compressor

- Quantization (and pruning) for both: practitioners and researchers.
- **For practitioners**: easy to use recipes with heavily optimized default configurations for all quantization formats supported in vLLM (INT W8A8, INT W8A16, INT W4A16, FP W8A8).
- **For researchers**: fine-grained control over quantization scales (per-channel, per-tensor, per-group), symmetric/asymmetric, with or without calibration data, activation reordering, dynamic/static activation quantization, etc.

Meet LLM-Compressor: github.com/vllm-project/llm-compressor

- Quantization (and pruning) for both: practitioners and researchers.
- **For practitioners**: easy to use recipes with heavily optimized default configurations for all quantization formats supported in vLLM (INT W8A8, INT W8A16, INT W4A16, FP W8A8).
- **For researchers**: fine-grained control over quantization scales (per-channel, per-tensor, per-group), symmetric/asymmetric, with or without calibration data, activation reordering, dynamic/static activation quantization, etc.
- Models saved into **compressed-tensors** format (extension of safe-tensors), and directly optimized for vLLM inference kernels.

Meet LLM-Compressor: github.com/vllm-project/llm-compressor

fp8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

recipe = QuantizationModifier(targets="Linear", scheme="FP8_DYNAMIC")
oneshot(model=model, recipe=recipe)
model.save_pretrained(save_path)
```

Meet LLM-Compressor: github.com/vllm-project/llm-compressor

fp8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

recipe = QuantizationModifier(targets="Linear", scheme="FP8_DYNAMIC")
oneshot(model=model, recipe=recipe)
model.save_pretrained(save_path)
```

Meet LLM-Compressor: github.com/vllm-project/llm-compressor

fp8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

recipe = QuantizationModifier(targets="Linear", scheme="FP8_DYNAMIC")
oneshot(model=model, recipe=recipe)
model.save_pretrained(save_path)
```

int8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier, SmoothQuantModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

ds = load_dataset("neuralmagic/LLM_compression_calibration").map(
    lambda x: {"text": tokenizer.apply_chat_template(x["messages"], add_generation_prompt=False, tokenize=False)}
)

recipe = [
    SmoothQuantModifier(smoothing_strength=0.8),
    QuantizationModifier(
        targets="Linear",
        scheme="W8A8",
        ignore=["lm_head"]
    )
]

oneshot(
    model=model,
    recipe=recipe,
    dataset=ds,
    num_calibration_samples=128
)
model.save_pretrained(save_path)
```

int8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier, SmoothQuantModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

ds = load_dataset("neuralmagic/LLM_compression_calibration").map(
    lambda x: {"text": tokenizer.apply_chat_template(x["messages"], add_generation_prompt=False, tokenize=False)}
)

recipe = [
    SmoothQuantModifier(smoothing_strength=0.8),
    QuantizationModifier(
        targets="Linear",
        scheme="W8A8",
        ignore=["lm_head"]
    )
]

oneshot(
    model=model,
    recipe=recipe,
    dataset=ds,
    num_calibration_samples=128
)
model.save_pretrained(save_path)
```

int8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier, SmoothQuantModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

ds = load_dataset("neuralmagic/LLM_compression_calibration").map(
    lambda x: {"text": tokenizer.apply_chat_template(x["messages"], add_generation_prompt=False, tokenize=False)}
)

recipe = [
    SmoothQuantModifier(smoothing_strength=0.8),
    QuantizationModifier(
        targets="Linear",
        scheme="W8A8",
        ignore=["lm_head"]
    )
]

oneshot(
    model=model,
    recipe=recipe,
    dataset=ds,
    num_calibration_samples=128
)
model.save_pretrained(save_path)
```

int8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier, SmoothQuantModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

ds = load_dataset("neuralmagic/LLM_compression_calibration").map(
    lambda x: {"text": tokenizer.apply_chat_template(x["messages"], add_generation_prompt=False, tokenize=False)}
)

recipe = [
    SmoothQuantModifier(smoothing_strength=0.8),
    QuantizationModifier(
        targets="Linear",
        scheme="W8A8",
        ignore=["lm_head"]
    )
]

oneshot(
    model=model,
    recipe=recipe,
    dataset=ds,
    num_calibration_samples=128
)
model.save_pretrained(save_path)
```

int8_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier, SmoothQuantModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

ds = load_dataset("neuralmagic/LLM_compression_calibration").map(
    lambda x: {"text": tokenizer.apply_chat_template(x["messages"], add_generation_prompt=False, tokenize=False)}
)

recipe = [
    SmoothQuantModifier(smoothing_strength=0.8),
    QuantizationModifier(
        targets="Linear",
        scheme="W8A8",
        ignore=["lm_head"]
    )
]

oneshot(
    model=model,
    recipe=recipe,
    dataset=ds,
    num_calibration_samples=128
)
model.save_pretrained(save_path)
```

~~int8_quantize.py~~ int4_quantize.py

```python
from transformers import AutoModelForCausalLM, AutoTokenizer
from llmcompressor.modifiers.quantization import QuantizationModifier, SmoothQuantModifier
from llmcompressor.transformers import oneshot

model_stub = "deepseek-ai/DeepSeek-R1-Distill-Llama-8B"
model = AutoModelForCausalLM.from_pretrained(model_stub)
tokenizer = AutoTokenizer.from_pretrained(model_stub)

ds = load_dataset("neuralmagic/LLM_compression_calibration").map(
    lambda x: {"text": tokenizer.apply_chat_template(x["messages"], add_generation_prompt=False, tokenize=False)}
)

recipe = [
    ~~SmoothQuantModifier(smoothing_strength=0.8),~~
    QuantizationModifier(
        targets="Linear",
        ~~scheme="W8A8",~~ scheme ="W4A16",
        ignore=["lm_head"]
    )
]

oneshot(
    model=model,
    recipe=recipe,
    dataset=ds,
    num_calibration_samples=128
)
model.save_pretrained(save_path)
```

Fine-grained control over the quantization process

```
recipe = """
quant_stage:
    quant_modifiers:
        QuantizationModifier:
            targets: ["Linear"]
            ignore: ["lm_head"]
            config_groups:
                group_0:
                    weights:
                        num_bits: 4 | 8
                        type: int | float
                        strategy: tensor | channel | group
                        group_size: 128
                        actorder: false | weight | group
                        symmetric: true | false
                    input_activations:
                        num_bits: 4 | 8
                        type: int | float
                        strategy: tensor | token
                        dynamic: true | false
"""
```

… but what about accuracy?

Model = **Llama-3.1-Instruct**

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Recovery %	Average Score	MMLU 5-shot	MMLU CoT 0-shot	ARC-C 0-shot	GSM8k CoT 8-shot	HellaSwag 10-shot	Winogrande 5-shot	TruthfulQA 0-shot
8B	BF16	100.00	74.06	68.3	72.8	81.4	82.8	80.5	78.1	54.5
	W8A8-FP	99.31	73.55	68.0	71.6	81.2	82.0	80.0	77.7	54.3
	W8A8-INT	100.31	74.29	67.8	72.2	81.7	84.8	80.3	78.5	54.7
	W4A16-INT	98.72	73.11	66.9	71.1	80.2	82.9	79.9	78.0	52.8
70B	BF16	100.00	84.40	83.8	86.0	93.3	94.9	86.8	85.3	60.7
	W8A8-FP	99.72	84.16	83.8	85.5	93.5	94.5	86.6	84.6	60.6
	W8A8-INT	99.87	84.29	83.7	85.8	93.1	94.2	86.7	85.1	61.4
	W4A16-INT	99.53	84.00	83.6	85.6	92.8	94.4	86.3	85.5	59.8
405B	BF16	100.00	86.79	87.4	88.1	95.0	96.0	88.5	87.2	65.3
	W8A8-FP	100.12	86.89	87.5	88.1	95.0	95.8	88.5	88.0	65.3
	W8A8-INT	99.32	86.20	87.1	87.7	94.4	95.5	88.2	86.1	64.4
	W4A16-INT	99.98	86.78	87.2	87.7	95.3	96.3	88.3	87.4	65.3

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Recovery %	Average Score	MMLU 5-shot	MMLU CoT 0-shot	ARC-C 0-shot	GSM8k CoT 8-shot	HellaSwag 10-shot	Winogrande 5-shot	TruthfulQA 0-shot
8B	BF16	100.00	74.06	68.3	72.8	81.4	82.8	80.5	78.1	54.5
	W8A8-FP	99.31	73.55	68.0	71.6	81.2	82.0	80.0	77.7	54.3
	W8A8-INT	100.31	74.29	67.8	72.2	81.7	84.8	80.3	78.5	54.7
	W4A16-INT	98.72	73.11	66.9	71.1	80.2	82.9	79.9	78.0	52.8
70B	BF16	100.00	84.40	83.8	86.0	93.3	94.9	86.8	85.3	60.7
	W8A8-FP	99.72	84.16	83.8	85.5	93.5	94.5	86.6	84.6	60.6
	W8A8-INT	99.87	84.29	83.7	85.8	93.1	94.2	86.7	85.1	61.4
	W4A16-INT	99.53	84.00	83.6	85.6	92.8	94.4	86.3	85.5	59.8
405B	BF16	100.00	86.79	87.4	88.1	95.0	96.0	88.5	87.2	65.3
	W8A8-FP	100.12	86.89	87.5	88.1	95.0	95.8	88.5	88.0	65.3
	W8A8-INT	99.32	86.20	87.1	87.7	94.4	95.5	88.2	86.1	64.4
	W4A16-INT	99.98	86.78	87.2	87.7	95.3	96.3	88.3	87.4	65.3

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Recovery %	Average Score	MMLU 5-shot	MMLU CoT 0-shot	ARC-C 0-shot	GSM8k CoT 8-shot	HellaSwag 10-shot	Winogrande 5-shot	TruthfulQA 0-shot
8B	BF16	100.00	74.06	68.3	72.8	81.4	82.8	80.5	78.1	54.5
	W8A8-FP	99.31	73.55	68.0	71.6	81.2	82.0	80.0	77.7	54.3
	W8A8-INT	100.31	74.29	67.8	72.2	81.7	84.8	80.3	78.5	54.7
	W4A16-INT	98.72	73.11	66.9	71.1	80.2	82.9	79.9	78.0	52.8
70B	BF16	100.00	84.40	83.8	86.0	93.3	94.9	86.8	85.3	60.7
	W8A8-FP	99.72	84.16	83.8	85.5	93.5	94.5	86.6	84.6	60.6
	W8A8-INT	99.87	84.29	83.7	85.8	93.1	94.2	86.7	85.1	61.4
	W4A16-INT	99.53	84.00	83.6	85.6	92.8	94.4	86.3	85.5	59.8
405B	BF16	100.00	86.79	87.4	88.1	95.0	96.0	88.5	87.2	65.3
	W8A8-FP	100.12	86.89	87.5	88.1	95.0	95.8	88.5	88.0	65.3
	W8A8-INT	99.32	86.20	87.1	87.7	94.4	95.5	88.2	86.1	64.4
	W4A16-INT	99.98	86.78	87.2	87.7	95.3	96.3	88.3	87.4	65.3

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Recovery %	Average Score	MMLU 5-shot	MMLU CoT 0-shot	ARC-C 0-shot	GSM8k CoT 8-shot	HellaSwag 10-shot	Winogrande 5-shot	TruthfulQA 0-shot
8B	BF16	100.00	74.06	68.3	72.8	81.4	82.8	80.5	78.1	54.5
	W8A8-FP	99.31	73.55	68.0	71.6	81.2	82.0	80.0	77.7	54.3
	W8A8-INT	100.31	74.29	67.8	72.2	81.7	84.8	80.3	78.5	54.7
	W4A16-INT	98.72	73.11	66.9	71.1	80.2	82.9	79.9	78.0	52.8
70B	BF16	100.00	84.40	83.8	86.0	93.3	94.9	86.8	85.3	60.7
	W8A8-FP	99.72	84.16	83.8	85.5	93.5	94.5	86.6	84.6	60.6
	W8A8-INT	99.87	84.29	83.7	85.8	93.1	94.2	86.7	85.1	61.4
	W4A16-INT	99.53	84.00	83.6	85.6	92.8	94.4	86.3	85.5	59.8
405B	BF16	100.00	86.79	87.4	88.1	95.0	96.0	88.5	87.2	65.3
	W8A8-FP	100.12	86.89	87.5	88.1	95.0	95.8	88.5	88.0	65.3
	W8A8-INT	99.32	86.20	87.1	87.7	94.4	95.5	88.2	86.1	64.4
	W4A16-INT	99.98	86.78	87.2	87.7	95.3	96.3	88.3	87.4	65.3

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Academic Benchmarks (Open LLM Leaderboard V2)								Real-World Benchmarks			
		Recovery %	Average Score	IFEval 0-shot	BBH 3-shot	Math lvl 5 4-shot	GPQA 0-shot	MuSR 0-shot	MMLU-Pro 5-shot	Arena-Hard Win-Rate	HumanEval pass@1	HumanEval+ pass@1	RULER Score
8B	BF16	100.0	27.6	77.8	30.1	15.7	3.7	7.6	30.8	25.8	67.3	60.7	82.8
	W8A8-FP	101.2	27.9	77.2	29.6	16.5	5.7	7.5	31.2	26.8	67.3	61.3	82.8
	W8A8-INT	101.5	28.0	77.9	30.9	15.5	5.4	7.6	30.9	27.2	67.1	60.0	82.8
	W4A16-INT	96.1	26.5	76.3	28.9	14.8	4.1	6.3	28.8	24.0	67.1	59.1	81.1
70B	BF16	100.0	41.7	86.4	55.8	26.1	15.4	18.1	48.1	57.0	79.7	74.8	83.3
	W8A8-FP	100.0	41.7	87.6	54.9	28.0	14.6	17.2	47.7	57.7	80.0	75.0	83.0
	W8A8-INT	97.3	40.5	86.6	55.2	23.9	13.6	16.8	47.1	57.0	78.7	74.0	82.5
	W4A16-INT	97.4	40.6	85.7	55.0	24.4	13.8	17.2	47.2	56.3	80.5	74.2	82.2
405B	BF16	100.0	48.7	87.7	67.0	38.9	19.5	19.5	59.7	67.4	86.8	80.1	-
	W8A8-FP	99.9	48.7	86.8	67.1	38.8	18.9	20.8	59.4	66.9	87.0	81.0	-
	W8A8-INT	98.3	47.9	86.9	66.7	35.8	20.4	19.2	58.4	64.6	86.9	80.4	-
	W4A16-INT	98.9	48.2	88.0	67.5	37.6	17.5	19.4	59.3	66.5	85.1	78.9	-

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Academic Benchmarks (Open LLM Leaderboard V2)								Real-World Benchmarks			
		Recovery %	Average Score	IFEval 0-shot	BBH 3-shot	Math lvl 5 4-shot	GPQA 0-shot	MuSR 0-shot	MMLU-Pro 5-shot	Arena-Hard Win-Rate	HumanEval pass@1	HumanEval+ pass@1	RULER Score
8B	BF16	100.0	27.6	77.8	30.1	15.7	3.7	7.6	30.8	25.8	67.3	60.7	82.8
	W8A8-FP	101.2	27.9	77.2	29.6	16.5	5.7	7.5	31.2	26.8	67.3	61.3	82.8
	W8A8-INT	101.5	28.0	77.9	30.9	15.5	5.4	7.6	30.9	27.2	67.1	60.0	82.8
	W4A16-INT	96.1	26.5	76.3	28.9	14.8	4.1	6.3	28.8	24.0	67.1	59.1	81.1
70B	BF16	100.0	41.7	86.4	55.8	26.1	15.4	18.1	48.1	57.0	79.7	74.8	83.3
	W8A8-FP	100.0	41.7	87.6	54.9	28.0	14.6	17.2	47.7	57.7	80.0	75.0	83.0
	W8A8-INT	97.3	40.5	86.6	55.2	23.9	13.6	16.8	47.1	57.0	78.7	74.0	82.5
	W4A16-INT	97.4	40.6	85.7	55.0	24.4	13.8	17.2	47.2	56.3	80.5	74.2	82.2
405B	BF16	100.0	48.7	87.7	67.0	38.9	19.5	19.5	59.7	67.4	86.8	80.1	-
	W8A8-FP	99.9	48.7	86.8	67.1	38.8	18.9	20.8	59.4	66.9	87.0	81.0	-
	W8A8-INT	98.3	47.9	86.9	66.7	35.8	20.4	19.2	58.4	64.6	86.9	80.4	-
	W4A16-INT	98.9	48.2	88.0	67.5	37.6	17.5	19.4	59.3	66.5	85.1	78.9	-

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

			Academic Benchmarks (Open LLM Leaderboard V2)							Real-World Benchmarks			
		Recovery %	Average Score	IFEval 0-shot	BBH 3-shot	Math lvl 5 4-shot	GPQA 0-shot	MuSR 0-shot	MMLU-Pro 5-shot	Arena-Hard Win-Rate	HumanEval pass@1	HumanEval+ pass@1	RULER Score
8B	BF16	100.0	27.6	77.8	30.1	15.7	3.7	7.6	30.8	25.8	67.3	60.7	82.8
	W8A8-FP	101.2	27.9	77.2	29.6	16.5	5.7	7.5	31.2	26.8	67.3	61.3	82.8
	W8A8-INT	101.5	28.0	77.9	30.9	15.5	5.4	7.6	30.9	27.2	67.1	60.0	82.8
	W4A16-INT	96.1	26.5	76.3	28.9	14.8	4.1	6.3	28.8	24.0	67.1	59.1	81.1
70B	BF16	100.0	41.7	86.4	55.8	26.1	15.4	18.1	48.1	57.0	79.7	74.8	83.3
	W8A8-FP	100.0	41.7	87.6	54.9	28.0	14.6	17.2	47.7	57.7	80.0	75.0	83.0
	W8A8-INT	97.3	40.5	86.6	55.2	23.9	13.6	16.8	47.1	57.0	78.7	74.0	82.5
	W4A16-INT	97.4	40.6	85.7	55.0	24.4	13.8	17.2	47.2	56.3	80.5	74.2	82.2
405B	BF16	100.0	48.7	87.7	67.0	38.9	19.5	19.5	59.7	67.4	86.8	80.1	-
	W8A8-FP	99.9	48.7	86.8	67.1	38.8	18.9	20.8	59.4	66.9	87.0	81.0	-
	W8A8-INT	98.3	47.9	86.9	66.7	35.8	20.4	19.2	58.4	64.6	86.9	80.4	-
	W4A16-INT	98.9	48.2	88.0	67.5	37.6	17.5	19.4	59.3	66.5	85.1	78.9	-

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Model = **Llama-3.1-Instruct**

		Academic Benchmarks (Open LLM Leaderboard V2)								Real-World Benchmarks			
		Recovery %	Average Score	IFEval 0-shot	BBH 3-shot	Math lvl 5 4-shot	GPQA 0-shot	MuSR 0-shot	MMLU-Pro 5-shot	Arena-Hard Win-Rate	HumanEval pass@1	HumanEval+ pass@1	RULER Score
8B	BF16	100.0	27.6	77.8	30.1	15.7	3.7	7.6	30.8	25.8	67.3	60.7	82.8
	W8A8-FP	101.2	27.9	77.2	29.6	16.5	5.7	7.5	31.2	26.8	67.3	61.3	82.8
	W8A8-INT	101.5	28.0	77.9	30.9	15.5	5.4	7.6	30.9	27.2	67.1	60.0	82.8
	W4A16-INT	96.1	26.5	76.3	28.9	14.8	4.1	6.3	28.8	24.0	67.1	59.1	81.1
70B	BF16	100.0	41.7	86.4	55.8	26.1	15.4	18.1	48.1	57.0	79.7	74.8	83.3
	W8A8-FP	100.0	41.7	87.6	54.9	28.0	14.6	17.2	47.7	57.7	80.0	75.0	83.0
	W8A8-INT	97.3	40.5	86.6	55.2	23.9	13.6	16.8	47.1	57.0	78.7	74.0	82.5
	W4A16-INT	97.4	40.6	85.7	55.0	24.4	13.8	17.2	47.2	56.3	80.5	74.2	82.2
405B	BF16	100.0	48.7	87.7	67.0	38.9	19.5	19.5	59.7	67.4	86.8	80.1	-
	W8A8-FP	99.9	48.7	86.8	67.1	38.8	18.9	20.8	59.4	66.9	87.0	81.0	-
	W8A8-INT	98.3	47.9	86.9	66.7	35.8	20.4	19.2	58.4	64.6	86.9	80.4	-
	W4A16-INT	98.9	48.2	88.0	67.5	37.6	17.5	19.4	59.3	66.5	85.1	78.9	-

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

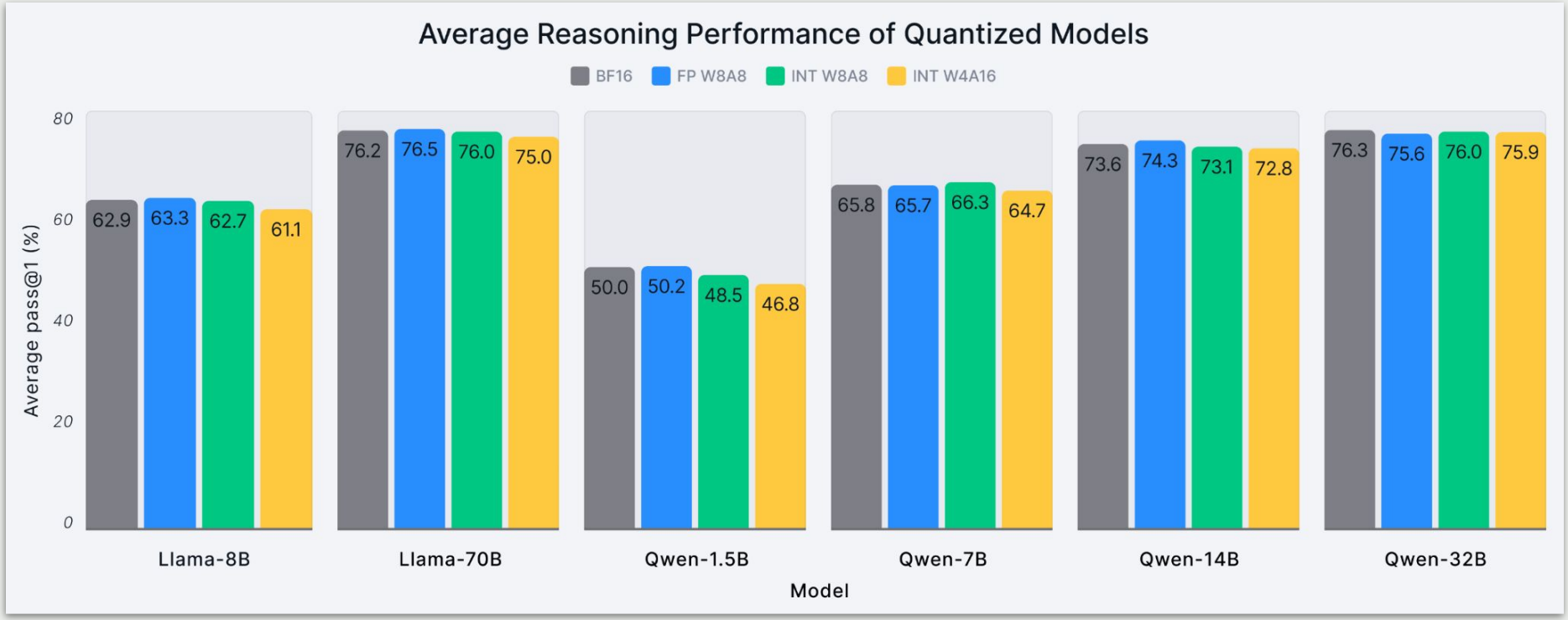
Model = **Llama-3.1-Instruct**

		Academic Benchmarks (Open LLM Leaderboard V2)								Real-World Benchmarks			
		Recovery %	Average Score	IFEval 0-shot	BBH 3-shot	Math lvl 5 4-shot	GPQA 0-shot	MuSR 0-shot	MMLU-Pro 5-shot	Arena-Hard Win-Rate	HumanEval pass@1	HumanEval+ pass@1	RULER Score
8B	BF16	100.0	27.6	77.8	30.1	15.7	3.7	7.6	30.8	25.8	67.3	60.7	82.8
	W8A8-FP	101.2	27.9	77.2	29.6	16.5	5.7	7.5	31.2	26.8	67.3	61.3	82.8
	W8A8-INT	101.5	28.0	77.9	30.9	15.5	5.4	7.6	30.9	27.2	67.1	60.0	82.8
	W4A16-INT	96.1	26.5	76.3	28.9	14.8	4.1	6.3	28.8	24.0	67.1	59.1	81.1
70B	BF16	100.0	41.7	86.4	55.8	26.1	15.4	18.1	48.1	57.0	79.7	74.8	83.3
	W8A8-FP	100.0	41.7	87.6	54.9	28.0	14.6	17.2	47.7	57.7	80.0	75.0	83.0
	W8A8-INT	97.3	40.5	86.6	55.2	23.9	13.6	16.8	47.1	57.0	78.7	74.0	82.5
	W4A16-INT	97.4	40.6	85.7	55.0	24.4	13.8	17.2	47.2	56.3	80.5	74.2	82.2
405B	BF16	100.0	48.7	87.7	67.0	38.9	19.5	19.5	59.7	67.4	86.8	80.1	-
	W8A8-FP	99.9	48.7	86.8	67.1	38.8	18.9	20.8	59.4	66.9	87.0	81.0	-
	W8A8-INT	98.3	47.9	86.9	66.7	35.8	20.4	19.2	58.4	64.6	86.9	80.4	-
	W4A16-INT	98.9	48.2	88.0	67.5	37.6	17.5	19.4	59.3	66.5	85.1	78.9	-

Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about accuracy?

Reasoning Performance = average score on **AIME 2024**, **MATH-500**, and **GPQA-Diamond**
Models = **DeepSeek-R1-Distill**



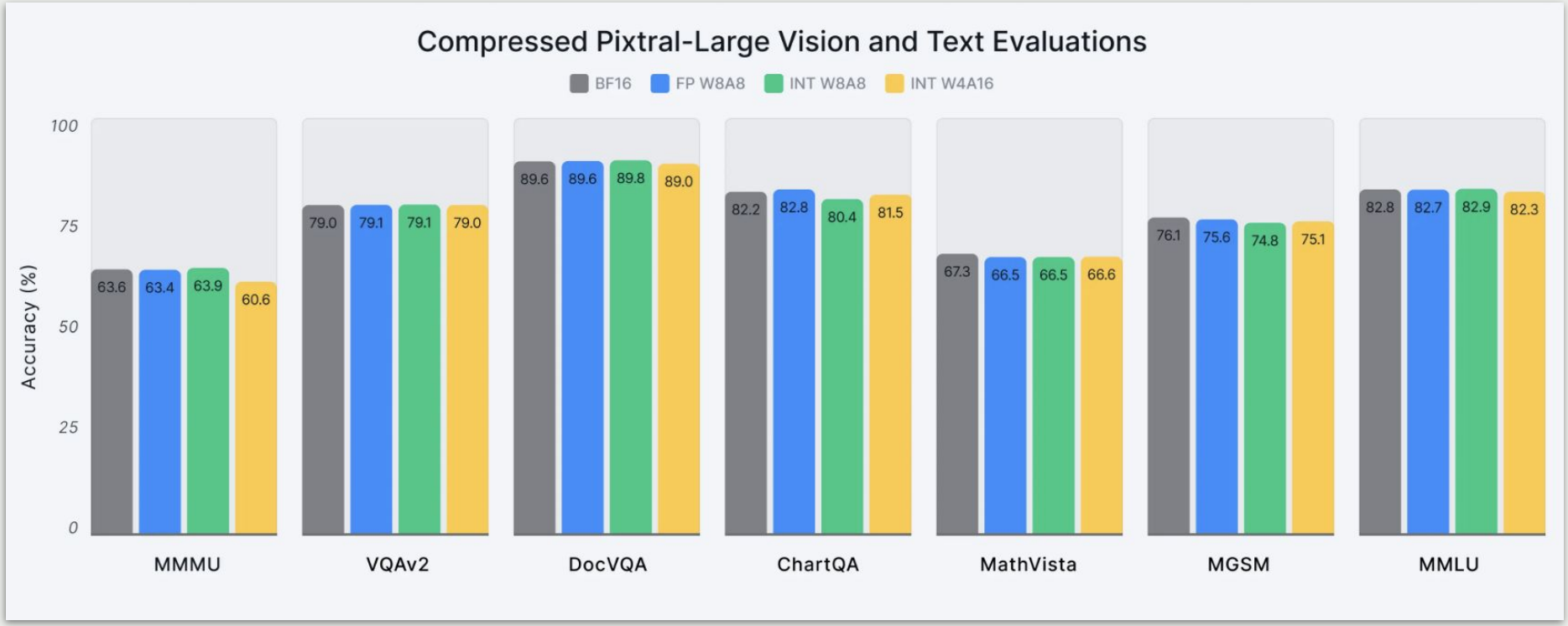
Kurtić, E., et al. (2025). "Quantized DeepSeek-R1 Models: Deployment-Ready Reasoning Models."

Fresh off the press: INT4 quantized DeepSeek-R1-0528

	Recovery (%)	deepseek/DeepSeek-R1-0528	RedHatAI/DeepSeek-R1-0528-quantized.w4a16 (this model)
AIME 2024 pass@1	98.50	88.66	87.33
MATH-500 pass@1	99.88	97.52	97.40
GPQA Diamond pass@1	101.21	79.65	80.61
Reasoning Average Score	**99.82**	**88.61**	**88.45**

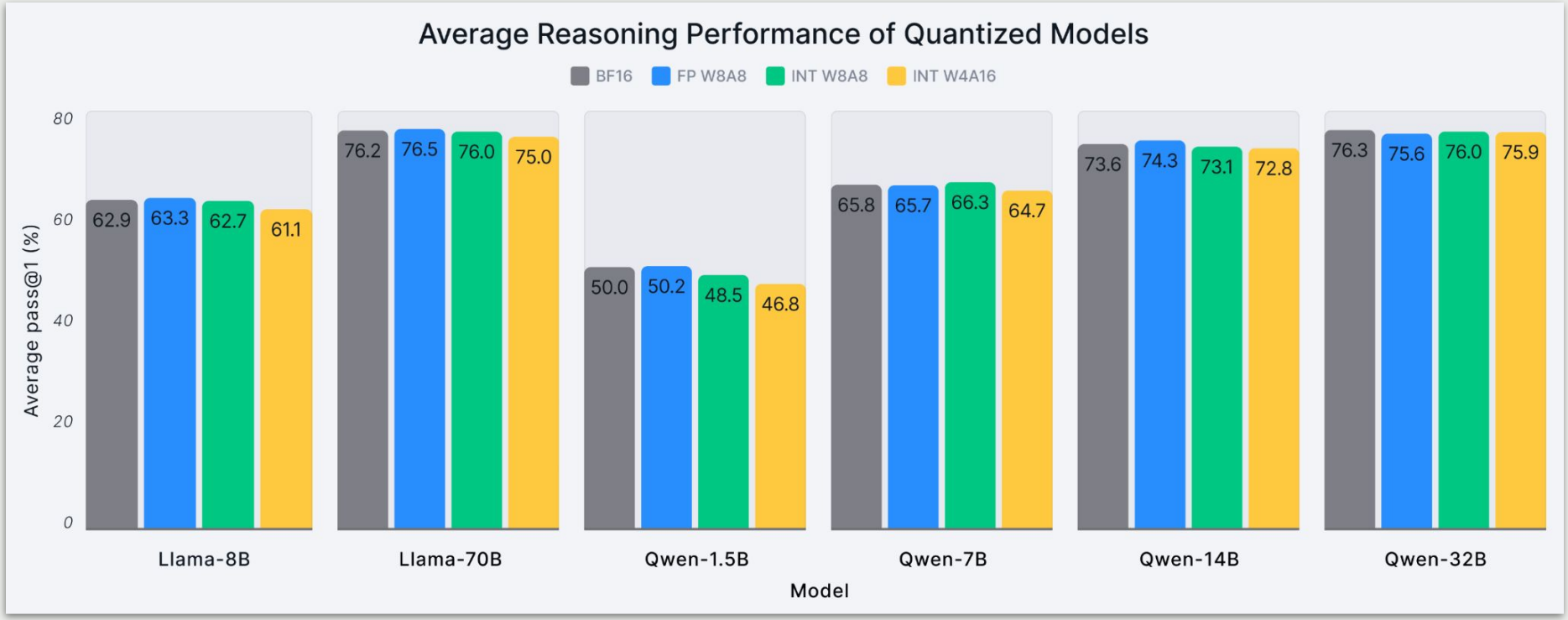
Kurtić, E., et al. (2025). "Quantized DeepSeek-R1 Models: Deployment-Ready Reasoning Models."

Quantization plays well with VLMs as well



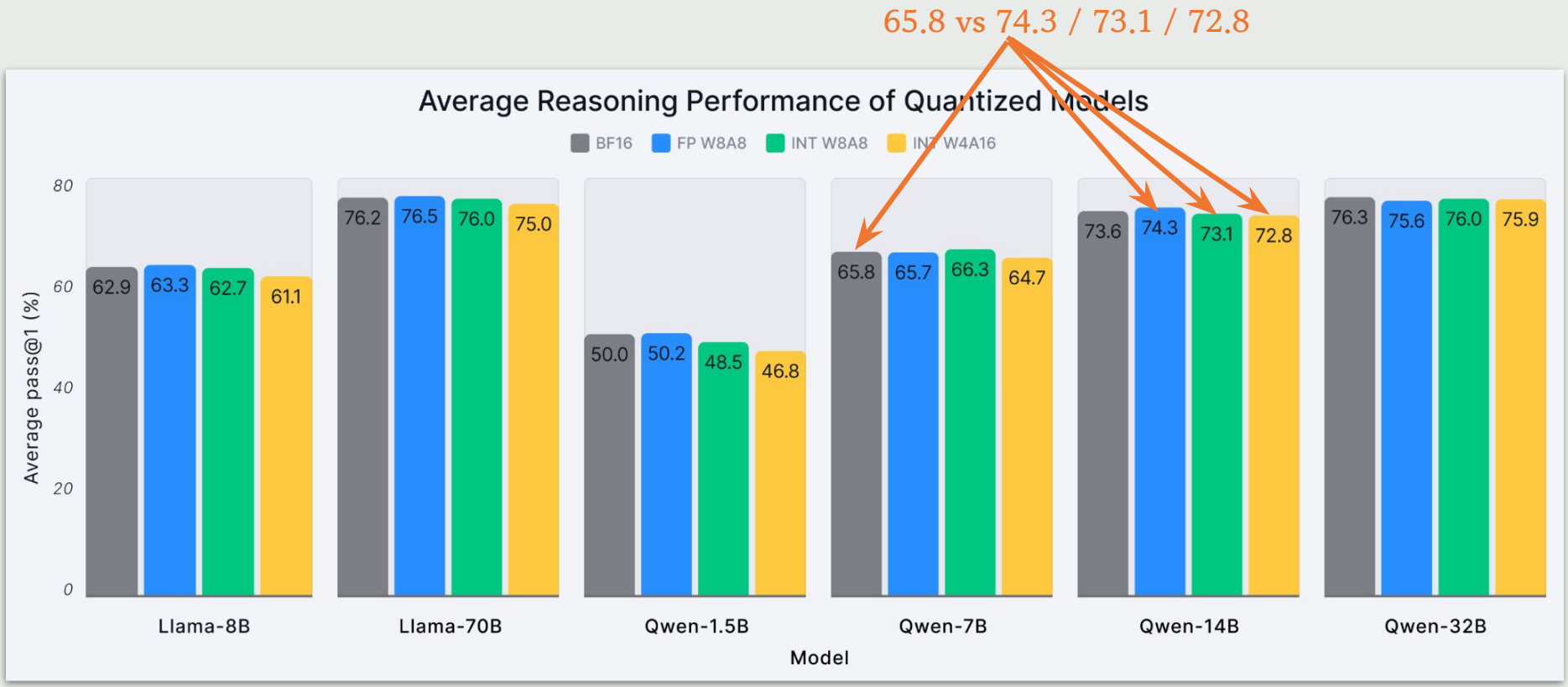
Larger quantized is always better than smaller unquantized

Larger quantized is always better than smaller unquantized



Kurtić, E., et al. (2025). "Quantized DeepSeek-R1 Models: Deployment-Ready Reasoning Models."

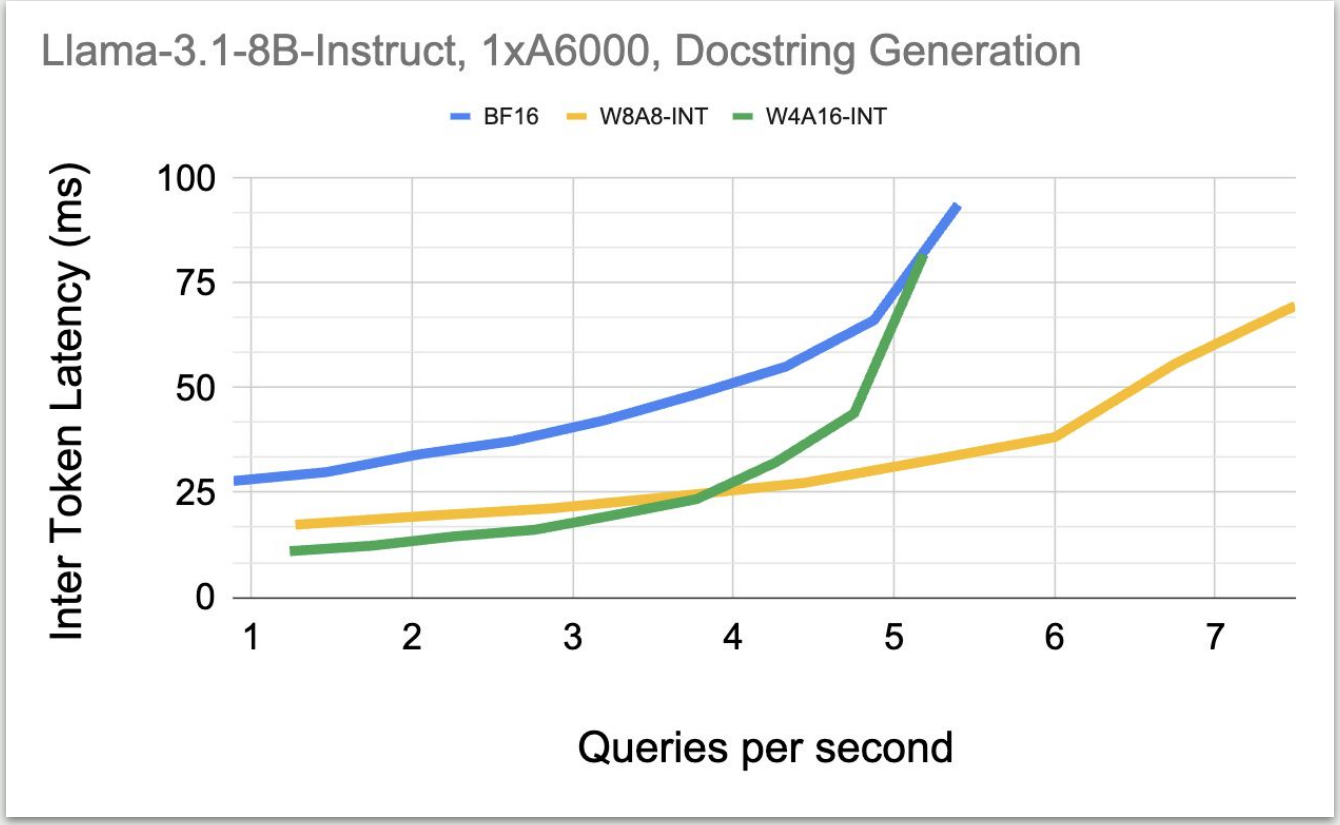
Larger quantized is always better than smaller unquantized



Kurtić, E., et al. (2025). "Quantized DeepSeek-R1 Models: Deployment-Ready Reasoning Models."

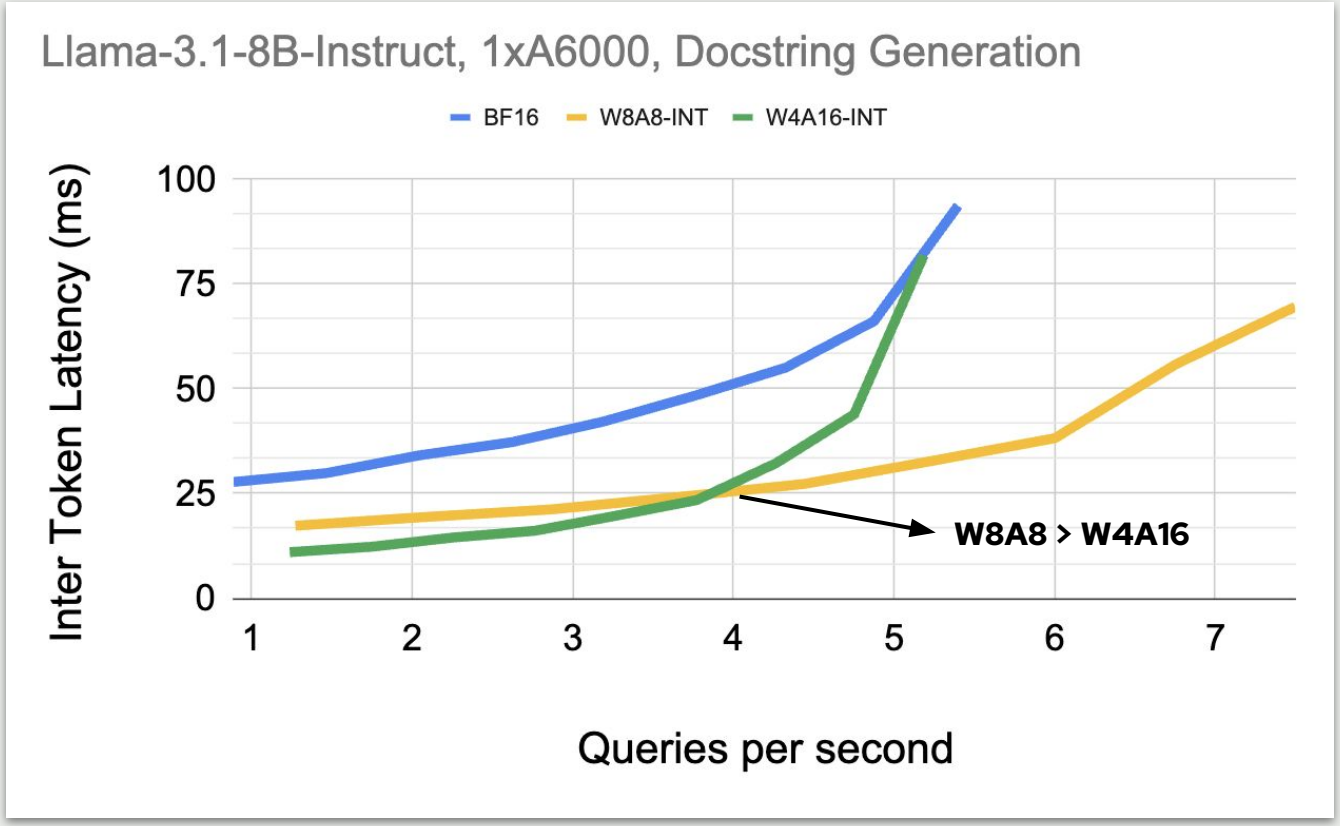
… but what about speedups?

… but what about speedups?



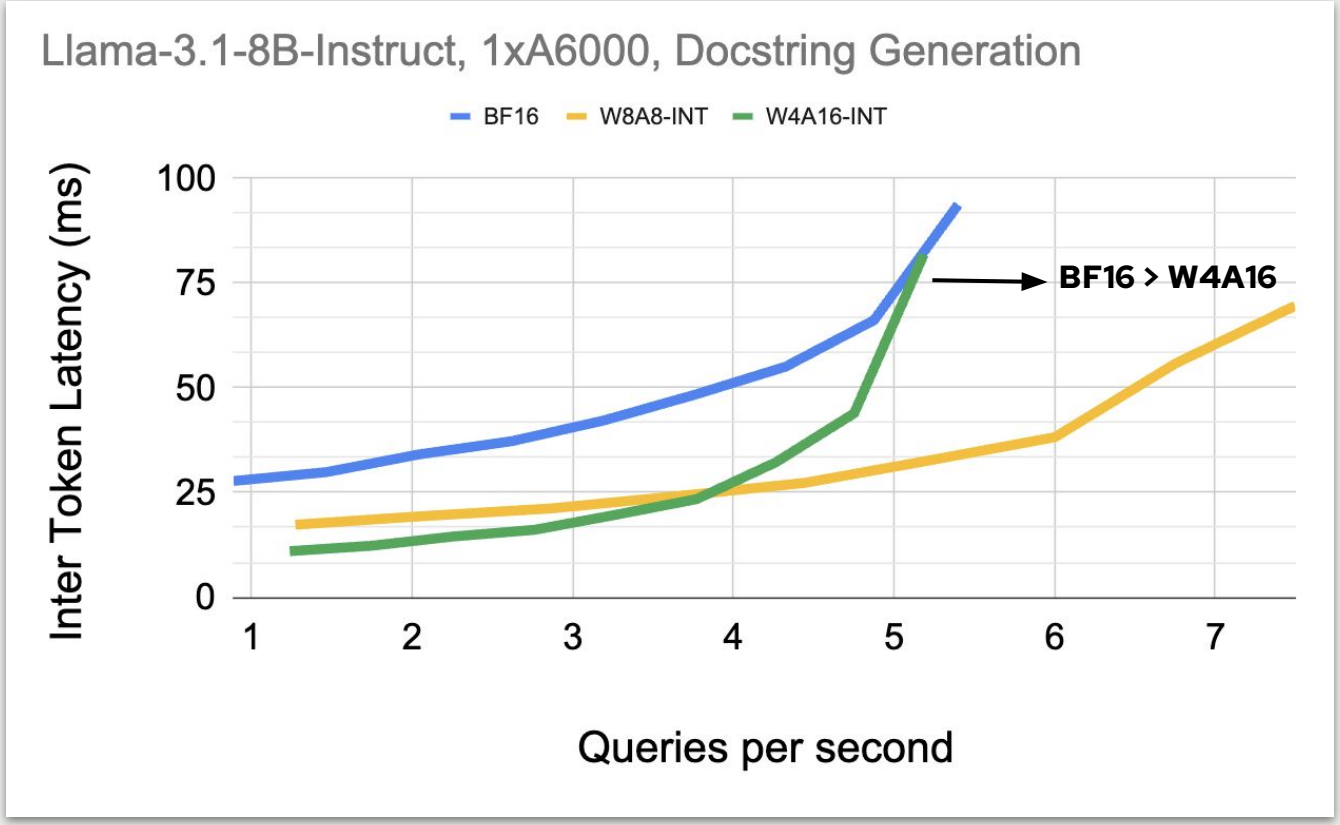
Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about speedups?



Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about speedups?

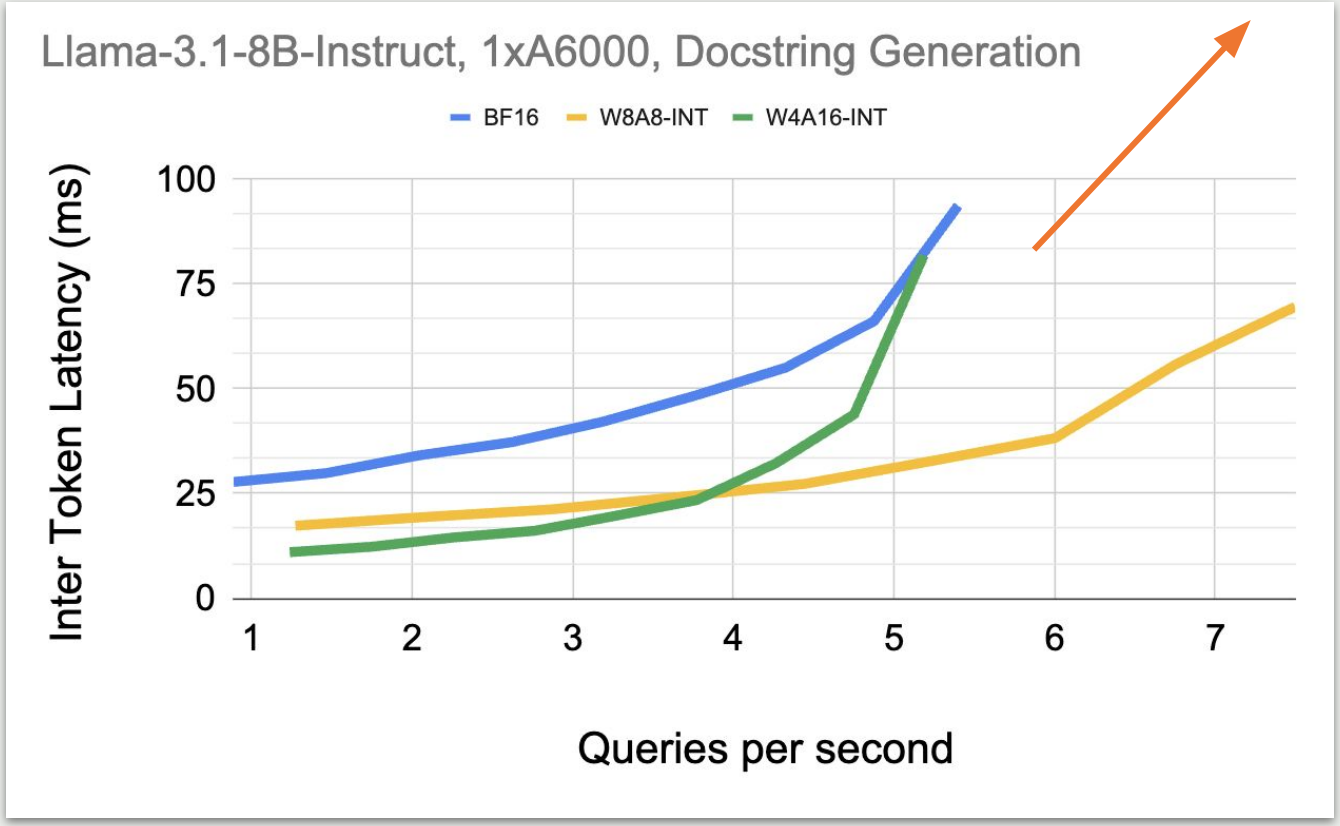


Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

… but what about speedups?

GuideLLM

github.com/neuralmagic/guidellm



Kurtić, E., et al. (2024). "Give Me BF16 or Give Me Death: Accuracy-Performance Trade-Offs in LLM Quantization."

What next?

1. How to quantize? → **LLM-Compressor**
   https://github.com/vllm-project/llm-compressor

2. How to benchmark? → **GuideLLM**
   https://github.com/neuralmagic/guidellm

3. Where to find high-quality quantized models? → **Red Hat AI's Hub**
   https://huggingface.co/RedHatAI

4. How to deploy? → **vLLM**
   https://github.com/vllm-project/vllm

Some more evidence ...

Inferencing - what to consider?

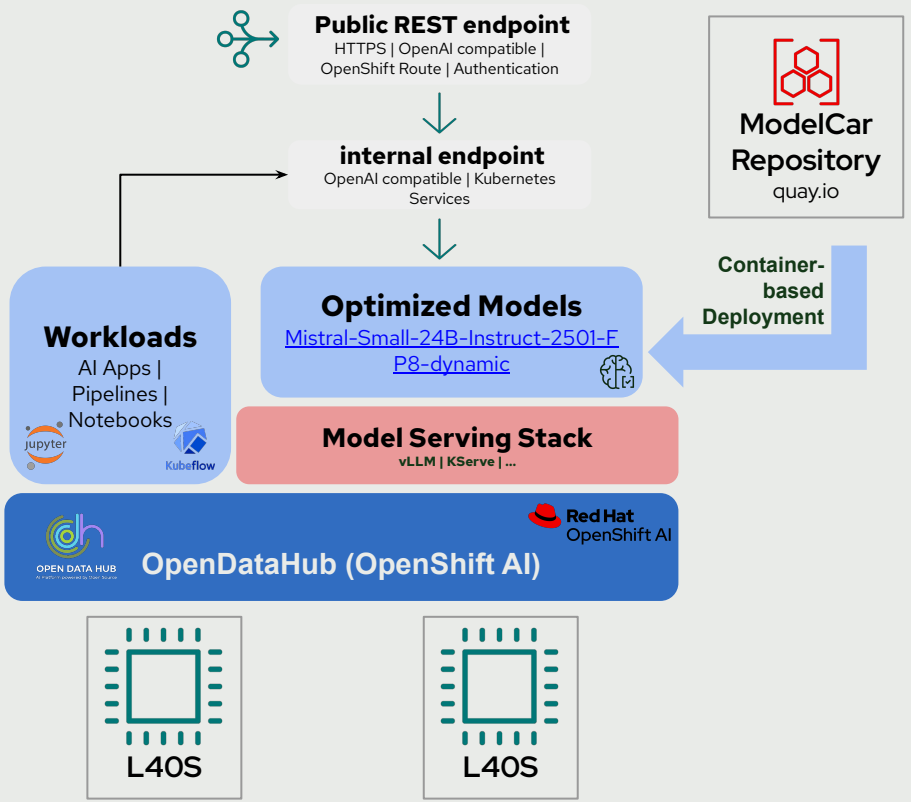
Why to run the tests?

1. **Capacity Planning & Hardware Utilization**
   We needed to know how many concurrent users our service could support on a single L40S, and how to push that GPU to its limits without running out of memory or throughput. We also compared a smaller, lighter model (Mistral 7B) to reveal any trade-offs—particularly its tendency to hallucinate more often—against the larger 24 B variants.

2. **The Power (and Pitfalls) of Quantization**
   Quantizing a model can halve or even quarter its memory footprint, dramatically boosting throughput and concurrency. However, a naive quantization approach can introduce unacceptable accuracy loss. We highlight Red Hat's FP8-dynamic method, which trims memory use by ≈ 50 % while retaining over 98 % of the full-precision model's reasoning and QA performance.

3. **Quality-to-Cost & Sustainability Trade-Offs**
   Ultra-large models deliver top-tier accuracy but demand huge GPU budgets and deliver diminishing returns on quality per dollar. In many real-world RAG and reasoning tasks, mid-sized checkpoints strike the sweet spot—offering robust instruction-following, multi-step logic, and hallucination resistance without the prohibitive cost of 100-plus-billion-parameter giants.

Inferencing Stack for performance tests



Models deployed on an AI stack, leveraging optimized models from Red Hat:

- currently Mistral Small 24B in an FP8 quantized version is deployed
- Model Serving stack is based on vLLM and KServe to provide scalable deployments
- OpenShift (AI) provide the GPU integration for dedicated and shared GPU usage
- the model can be exposed through external HTTPS and cluster internal endpoints
- An external endpoint can be integrated in an API Gateway

Model Benchmarks (1/3) - General Information

		mistralai/Mistral-Small-24B-Instruct-2501	RedHatAI/Mistral-Small-24B-Instruct-2501-FP8-dynamic	mistralai/Mistral-7B-Instruct-v0.3
Model Description		Instruction-tuned variant of Mistral-Small ("Instruct-2501")	the same "Instruct-2501" checkpoint, dynamically quantized	Instruction-tuned Mistral model (v0.3)
Model Licence		Apache License 2.0 (Huggingface Token required)	Apache License 2.0 (open)	Apache License 2.0 (Huggingface Token required)
Model Size		24 Billion params	24 Billion params	7 Billion Params
Context Length		32 768	32 768	32 768
Precision		BF16	FP8	BF16
GPU RAM required	Model Weights	~44GiB	~22GB	~13.6GB
	Reserve (Cuda Allocation, Pytorch Cache)	~3GB	~3GB	~3GB
	Left for KV-Cache*	N/A	~23GB	~31GB

*Assuming L40S NVIDIA GPUs

Model Benchmarks (2/3) - Quality

	mistralai/Mistral-Small-24B-Instruct-2501	RedHatAI/Mistral-Small-24B-Instruct-2501-FP8-dynamic	mistralai/Mistral-7B-Instruct-v0.3
IFEval (Inst-and-Prompt Level Strict Acc, 0-shot)	73.27	73.53	54.65
BBH (Acc-Norm, 3-shot)	45.18	44.39	25.57
MMLU-Pro (Acc, 5-shot)	38.83	37.28	23.06
Average Score	**52.42**	**51.73**	**19.23**

Above 3 benchmarks stress the model's ability to parse open-ended instructions, perform multi-step reasoning, and generalize under few-shot prompting—exactly what you need for **RAG and Agents** that must plan, call tools, loop over results, etc.

- **IFEval** is pure instruction-following with no examples—your baseline for "can it do what I just told it?"
- **BBH** tests a wide variety of hard reasoning tasks in a few-shot context—your proxy for "can it chain together reasoning steps under guidance?"
- **MMLU-Pro** zeroes in on tougher academic/factual challenges—your yardstick for "does it still know its facts when you give it 5 examples?"
- The **Average Score** neatly summarizes the three into one number you can track.

Between the three models:

- Both 24B checkpoints sit at ~73 % IFEval, ~44–45 % BBH and ~37–39 % MMLU-Pro → an **≈ 52 %** average
- The 7B comes in at **54.7 % IFEval, 25.6 % BBH, 23.1 % MMLU-Pro → a 19.2 % average**

The **Red Hat** quantized Mistral 24B heavily **outpaces** its little relative, while **only slightly losing** ground compared to its sibling. At the same time, the **FP8 quantization** make it a great choice when **balancing quality to infra requirements**!

Model Benchmarks (3/3) - Performance

Model	Test Strategy	Requests/sec	Concurrent Requests	Tokens/sec
Mistral-7B-Instruct-v0.3	synchronous	0.37	1.0	~47.4
	throughput (max)	10.93	176.94	~1 398.9
Mistral-Small-24B-Instruct-2501-FP8-dynamic	synchronous	0.19	1.0	~24.7
	throughput (max)	8.66	209.33	~1 108.6

Tokens /sec is the average number of **output** tokens the model emits per second under each load profile.

At **peak throughput**, the 7B model pushes out ~1 400 tokens/sec, while the quantized 24 B model does ~1 100 tokens/sec—consistent with their relative request/sec figures.

Single-Request Latency

- **Workload per inference:** Mistral-7B (7 B params at BF16) does ~7 B matrix ops on 733 TFLOPS cores; Mistral-24B-FP8 (24 B params at FP8) does ~3× more ops on 1 466 TFLOPS cores.
  Result: Even with FP8's double-speed cores, the 24 B model's larger size makes its per-call latency ~2× that of the 7 B.

High Concurrency

1. **Amortized overhead:** Kernel launches, memory setup, etc., get shared across many parallel requests.
   Memory efficiency: FP8 uses half the memory, so more data stays on-chip, reducing stalls.
2. **Pipelined attention:** Modern kernels (e.g. FlashAttention-3) overlap compute and data movement, pushing FP8 hardware toward peak throughput.

While the Mistral 7B performs **more responsive under low load,** under heavy load, the 24B FP8 model **"catches up",** narrowing the latency gap to only ~1.3× slower at max throughput.

In summary, **at a more than ~2.5× quality improvement, latency increases by much less** which makes the quantized model a great choice!

Q&A

Eldar Kurtić

Max Dargatz

June 3rd, 2025