# How to prevent tensorflow from allocating the totality of a GPU memory?

Asked 5 years, 1 month ago Active 3 months ago Viewed 184k times

302

119

I work in an environment in which computational resources are shared, i.e., we have a few server machines equipped with a few Nvidia Titan X GPUs each.

For small to moderate size models, the 12 GB of the Titan X is usually enough for 2–3 people to run training concurrently on the same GPU. If the models are small enough that a single model does not take full advantage of all the computational units of the GPU, this can actually result in a speedup compared with running one training process after the other. Even in cases where the concurrent access to the GPU does slow down the individual training time, it is still nice to have the flexibility of having multiple users simultaneously train on the GPU.

The problem with TensorFlow is that, by default, it allocates the full amount of available GPU memory when it is launched. Even for a small two-layer neural network, I see that all 12 GB of the GPU memory is used up.

Is there a way to make TensorFlow only allocate, say, 4 GB of GPU memory, if one knows that this is enough for a given model?

python tensorflow tensorflow2.0 tensorflow2.x nvidia-titan

Share Edit Follow

edited Aug 28 '20 at 9:55
Timbus Calin
7,665 2 16 37

asked Dec 10 '15 at 10:19
Fabien C.
3,175 3 11 6

## 16 Answers

Active Oldest Votes

303

You can set the fraction of GPU memory to be allocated when you construct a `tf.Session` by passing a `tf.GPUOptions` as part of the optional `config` argument:

```
# Assume that you have 12GB of GPU memory and want to allocate ~4GB:
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)

sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
```

The `per_process_gpu_memory_fraction` acts as a hard upper bound on the amount of GPU memory that will be used by the process on each GPU on the same machine. Currently, this fraction is applied uniformly to all of the GPUs on the same machine; there is no way to set this on a per-GPU basis.

Share Edit Follow

edited Jun 8 '17 at 15:00

answered Dec 10 '15 at 11:00
mrry
**118k** 22 370 387

3 Thank you very much. This info is quite hidden in the current doc. I would never have found it by myself :-) If you can answer, I would like to ask for two additional infos: 1- Does this limit the amount of memory ever used, or just the memory initially allocated? (ie. will it still allocate more memory if there is a need for it by the computation graph) 2- Is there a way to set this on a per-GPU basis? – Fabien C. Dec 11 '15 at 1:29

17 Related note: setting CUDA_VISIBLE_DEVICES to limit TensorFlow to a single GPU works for me. See acceleware.com/blog/cudavisibledevices-masking-gpus – rd11 Jan 12 '16 at 15:54

2 it seems that the memory allocation goes a bit over the request, e..g I requested per_process_gpu_memory_fraction=0.0909 on a 24443MiB gpu and got processes taking 2627MiB – jeremy_rutman Sep 23 '17 at 17:15

2 I can't seem to get this to work in a `MonitoredTrainingSession` – Anjum Sayed Oct 13 '17 at 5:34

2 @jeremy_rutman I believe this is due to cudnn and cublas context initialization. That is only relevant if you are executing kernels that use those libs though. – Daniel Feb 20 '19 at 23:26

198

```
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
sess = tf.Session(config=config)
```

https://github.com/tensorflow/tensorflow/issues/1578

Share Edit Follow

answered May 26 '16 at 7:43
Sergey Demyanov
**2,435** 1 13 9

15 This one is exactly what I want because in a multi-user environment, it is very inconvenient to specify the exact amount of GPU memory to reserve in the code itself. – xuancong84 Oct 3 '16 at 1:07

8 Also, if you're using Keras with a TF backend, you can use this and run `from keras import backend as K` and `K.set_session(sess)` to avoid memory limitations – Oliver Jul 1 '19 at 4:52

55

Here is an excerpt from the Book `Deep Learning with TensorFlow`

> In some cases it is desirable for the process to only allocate a subset of the available memory, or to only grow the memory usage as it is needed by the process. TensorFlow provides **two configuration** options on the session to control this. The first is the `allow_growth` option, which attempts to allocate only as much GPU memory based on runtime allocations, it starts out allocating very little memory, and as sessions get run and more GPU memory is needed, we extend the GPU memory region needed by the TensorFlow process.

**1) Allow growth: (more flexible)**

```
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.Session(config=config, ...)
```

The second method is `per_process_gpu_memory_fraction` option, which determines the fraction of the overall amount of memory that `each` visible GPU should be allocated. **Note:** No release of memory needed, it can even worsen memory fragmentation when done.

**2) Allocate fixed memory**:

To only allocate `40%` of the total memory of each GPU by:

```
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.4
session = tf.Session(config=config, ...)
```

**Note:** That's only useful though if you truly want to bind the amount of GPU memory available on the TensorFlow process.

Share Edit Follow

answered Jan 11 '18 at 18:57
user1767754
**17.6k** 12 106 125

As far as your question is concerned, option 2 might be useful to you. In general if you do not have multiple applications running on GPU and dynamic networks then it makes sense to use 'Allow growth' option. – aniket Jan 31 '20 at 7:13

---

45

**For TensorFlow 2.0 and 2.1** (docs):

```
import tensorflow as tf
tf.config.gpu.set_per_process_memory_growth(True)
```

**For TensorFlow 2.2+** (docs):

```
import tensorflow as tf
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
  tf.config.experimental.set_memory_growth(gpu, True)
```

The docs also list some more methods:

- Set environment variable `TF_FORCE_GPU_ALLOW_GROWTH` to `true`.

- Use `tf.config.experimental.set_virtual_device_configuration` to set a hard limit on a Virtual GPU device.

Share Edit Follow

edited Sep 28 '20 at 19:53
Mateen Ulhaq
**17.1k** 12 68 107

answered Apr 5 '19 at 18:26
Theo
**754** 9 11

1 @AkshayLAradhya no this is only for TF 2.0 and above. The other answers here will work fine for 1.13 and earlier. – Theo May 1 '19 at 21:19

3 Not beyond. For TF 2.2 it's 'tf.config.experimental.set_memory_growth' – Begoodpy Jul 25 '20 at 16:41

2 Since this is a highly upvoted answer, I've updated to the latest version of TF. – Mateen Ulhaq Sep 28 '20 at 19:56

@MateenUlhaq here is a link to the Tensorflow documentation you probably used: tensorflow.org/api_docs/python/tf/config/experimental/… – brethvoice Oct 8 '20 at 13:01

The first part "For TensorFlow 2.0 and 2.1..." is not accurate. It's not in the documentation source referenced and I have TF2.0 and when I tested it I got an error. The second part though works on TF2.0 as well as TF2.2+ – trust_words Oct 13 '20 at 5:31

24

**For Tensorflow version 2.0 and 2.1 use the following snippet**:

```
import tensorflow as tf
gpu_devices = tf.config.experimental.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpu_devices[0], True)
```

**For prior versions** , following snippet used to work for me:

```
import tensorflow as tf
tf_config=tf.ConfigProto()
tf_config.gpu_options.allow_growth=True
sess = tf.Session(config=tf_config)
```

Share Edit Follow

edited Mar 11 '20 at 21:31
CATALUNA84
**422** 4 13

answered Dec 1 '19 at 14:47
Anurag
**361** 2 4

21

All the answers above assume execution with a `sess.run()` call, which is becoming the exception rather than the rule in recent versions of TensorFlow.

When using the `tf.Estimator` framework (TensorFlow 1.4 and above) the way to pass the fraction along to the implicitly created `MonitoredTrainingSession` is,

```
opts = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)
conf = tf.ConfigProto(gpu_options=opts)
trainingConfig = tf.estimator.RunConfig(session_config=conf, ...)
```

```
tf.estimator.Estimator(model_fn=...,
                       config=trainingConfig)
```

Similarly in Eager mode (TensorFlow 1.5 and above),

```
opts = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)
conf = tf.ConfigProto(gpu_options=opts)
tfe.enable_eager_execution(config=conf)
```

**Edit: 11-04-2018** As an example, if you are to use `tf.contrib.gan.train`, then you can use something similar to bellow:

```
tf.contrib.gan.gan_train(........, config=conf)
```

Share Edit Follow

edited Apr 11 '18 at 5:56
GPrathap
5,448 6 51 73

answered Feb 8 '18 at 3:25
Urs
685 5 10

12

You can use

```
TF_FORCE_GPU_ALLOW_GROWTH=true
```

in your environment variables.

In tensorflow code:

```
bool GPUBFCAllocator::GetAllowGrowthValue(const GPUOptions& gpu_options) {
  const char* force_allow_growth_string =
      std::getenv("TF_FORCE_GPU_ALLOW_GROWTH");
  if (force_allow_growth_string == nullptr) {
    return gpu_options.allow_growth();
}
```

Share Edit Follow

edited Jun 17 '19 at 16:44

answered Jun 2 '19 at 17:15
Mey Khalili
121 1 3

11

## Tensorflow 2.0 Beta and (probably) beyond

The API changed again. It can be now found in:

```
tf.config.experimental.set_memory_growth(
    device,
    enable
)
```

Aliases:

- tf.compat.v1.config.experimental.set_memory_growth
- tf.compat.v2.config.experimental.set_memory_growth

**References:**

- https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/config/experimental/set_memory_growth
- https://www.tensorflow.org/guide/gpu#limiting_gpu_memory_growth

**See also:** *Tensorflow - Use a GPU*: https://www.tensorflow.org/guide/gpu

**for Tensorflow 2.0 Alpha see:** this answer

Share Edit Follow edited Nov 7 '19 at 8:12 answered Jun 17 '19 at 13:08
maxstrobel
**437** 4 13

---

6

Shameless plug: If you install the GPU supported Tensorflow, the session will first allocate all GPU whether you set it to use only CPU or GPU. I may add my tip that even you set the graph to use CPU only you should set the same configuration(as answered above:) ) to prevent the unwanted GPU occupation.

And in an interactive interface like IPython and Jupyter, you should also set that configure, otherwise, it will allocate all memory and left almost none for others. This is sometimes hard to notice.

Share Edit Follow edited Oct 8 '20 at 13:33 answered May 23 '17 at 7:52
Lerner Zhang
**4,037** 1 30 45

---

5

If you're using Tensorflow 2 try the following:

```
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.compat.v1.Session(config=config)
```

Share Edit Follow answered Mar 12 '20 at 6:44
ssp
**1,356** 7 15

4

For **Tensorflow 2.0** this this solution worked for me. (TF-GPU 2.0, Windows 10, GeForce RTX 2070)

```
physical_devices = tf.config.experimental.list_physical_devices('GPU')
assert len(physical_devices) > 0, "Not enough GPU hardware devices available"
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

Share Edit Follow

answered Oct 5 '19 at 21:05
Sunsetquest
**5,288** 2 27 31

1 I am using TF-GPU 2.0, Ubuntu 16.04.6, Tesla K80. – azar Oct 31 '19 at 20:05

@azar - Thanks for sharing. That's interesting the same issue on both Ubuntu and Windows. Somehow, I always think that the issues are different when getting closer to the hardware. Maybe this is becoming less so as time passes - maybe a good thing. – Sunsetquest Nov 4 '19 at 16:31

1

this code has worked for me:

```
import tensorflow as tf
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.compat.v1.InteractiveSession(config=config)
```

Share Edit Follow

answered Jul 8 '20 at 22:30
Kamil Marczak
**81** 3

1

Well I am new to tensorflow, I have Geforce 740m or something GPU with 2GB ram, I was running mnist handwritten kind of example for a native language with training data containing of 38700 images and 4300 testing images and was trying to get precision , recall , F1 using following code as sklearn was not giving me precise reults. once i added this to my existing code i started getting GPU errors.

```
TP = tf.count_nonzero(predicted * actual)
TN = tf.count_nonzero((predicted - 1) * (actual - 1))
FP = tf.count_nonzero(predicted * (actual - 1))
FN = tf.count_nonzero((predicted - 1) * actual)

prec = TP / (TP + FP)
recall = TP / (TP + FN)
f1 = 2 * prec * recall / (prec + recall)
```

plus my model was heavy i guess, i was getting memory error after 147, 148 epochs, and then I thought why not create functions for the tasks so I dont know if it works this way in tensrorflow, but I thought if a local variable is used and when out of scope it may release memory and i defined the above elements for training and testing in modules, I was able to achieve 10000 epochs without any issues, I hope this will help..

Share Edit Follow

answered Jan 21 '19 at 17:26
Imran Ud Din
**59** 1

I am amazed at TF's utility but also by it's memory use. On the CPU python allocating 30GB or so for a training job on the flowers dataset used in may TF examples. Insane. – Eric M Apr 12 '19 at 15:30

1

```
# allocate 60% of GPU memory
from keras.backend.tensorflow_backend import set_session
import tensorflow as tf
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.6
set_session(tf.Session(config=config))
```

Share Edit Follow

answered Sep 27 '19 at 0:28
DSBLR
**376** 2 9

The provided answer was flagged for review as a Low Quality Post. Here are some guidelines for How do I write a good answer?. This provided answer may be correct, but it could benefit from an explanation. Code only answers are not considered "good" answers. From review. – Trenton McKinney Sep 27 '19 at 4:04

1

i tried to train unet on voc data set but because of huge image size, memory finishes. i tried all the above tips, even tried with batch size==1, yet to no improvement. sometimes TensorFlow version also causes the memory issues. try by using

> pip install tensorflow-gpu==1.8.0

Share Edit Follow

answered Oct 16 '18 at 6:05
Khan
**870** 7 10

0

All the answers above refer to either setting the memory to a certain extent in `TensorFlow 1.X` versions or to allow memory growth in `TensorFlow 2.X`.

The method `tf.config.experimental.set_memory_growth` indeed works for allowing dynamic growth during the allocation/preprocessing. Nevertheless one may like to allocate from the start a specific GPU memory.

The logic behind allocating a specific GPU memory would also be to prevent OOM memory during training sessions. For example, if one trains while opening video-memory consuming Chrome tabs, the `tf.config.experimental.set_memory_growth(gpu, True)` could result in OOM errors thrown, hence the necessity of allocating from the start more memory in certain cases.

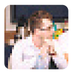
The recommended and correct way in which to allot memory per GPU in TensorFlow 2.X is done in the following manner:

```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  # Restrict TensorFlow to only allocate 1GB of memory on the first GPU
  try:
    tf.config.experimental.set_virtual_device_configuration(
        gpus[0],
        [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=1024)]
```

Share Edit Follow

edited Aug 28 '20 at 10:00

answered Aug 28 '20 at 9:54



Timbus Calin
**7,665** 2 16 37

**Highly active question**. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.