Performance Analysis on CPU nodes

Case study: The ISOKANN machine learning script

Dr. Christian Tuma

Zuse Institute Berlin

NATIONALES HOCHLEISTUNGS RECHNEN

ZIB ZUSE INSTITUTE BERLIN

NHR Graduate School Course Week 2022

Preparation

- Log in to “Lise”

  ```
  $ ssh [-Y] [-I keyfile] username@blogin.hlrn.de
  ```

- Allocate a compute node

  ```
  $ salloc -N 1
  ```

- Connect to the allocated compute node

  ```
  $ ssh [-Y] $SLURM_NODELIST
  ```

- Create and enter a new sub-directory

  ```
  $ mkdir ML; cd ML
  ```

- Take a copy of the ML script

  ```
  $ cp /sw/ml.py .
  ```

- Load the Tensorflow environment module

  ```
  $ module load tf/2.3.0
  ```

Initial run

- Execute the given script (in the background)

```
$ python ml.py &
...
... StreamExecutor device (0): Host, Default Version
... Creating new thread pool with default inter op setting: 192.
Tune using inter_op_parallelism_threads for best performance.
```

While the script is running, have a look at your processes … how many, system load?

```
$ ps -Lfu $USER
$ top
$ htop
```

- Wait for the script to finish, remember its timing result

```
...
Runtime in seconds: 584.704
```

Manage TF threading

- Option A: Limit the number of CPU cores visible to TF

  ```
  $ numactl -C 0-3 -l python ml.py &
  ```

  Now, how many threads, timing?
- Option B: Explicit thread numbers via calls of the TF API (lines 9—12 of the ML script)

  ```
  https://www.tensorflow.org/api_docs/python/tf/config/threading
  ```

  - first script argument: number of inter-parallel threads (default = 0 = automatic)

  - second script argument: number of intra-parallel threads (default = 0 = automatic)

  ```
  $ python ml.py 2 1 &
  ```

  (How many threads, timing?)
- Combination of Options A and B

  ```
  $ numactl -C 0 -l python ml.py 1 1
  ```

Impact of parameters in `model.fit()`

- Locate line 103 of the ML script

  ```
  model.fit(x..., y... , epochs=350, verbose=0, batch_size=32)
  ```

- Increase `batch_size` and repeat the run time measurement

  e.g. `batch_size=100, batch_size=300, batch_size=1000`

  ```
  $ numactl -C 0 -l python ml.py 1 1
  ```

  Impact of `batch_size` ?

- (Impact of `epochs` ?)

TF scaling tests

- Scaling behavior – run time as a function of the number of threads

  (use improved `batch_size` here)

```
$ numactl -C 0   -l python ml.py 1 1
$ numactl -C 0,1 -l python ml.py 2 1
$ numactl -C 0,1 -l python ml.py 1 2
$ numactl -C 0,1 -l python ml.py 2 2
```

  Speed-up?

- Scaling behavior with increased ML workload

  (use, e.g., `n_neurons_per_layer = 800` and `epochs=50`)

  Speed-up?

Code metrics

- Quick Performance Overview with Intel Application Performance Snapshot (APS)

```
$ module load vtune/2021
$ aps numactl -C 0,1 -l python ml.py 2 2
$ firefox aps_report_date_time.html
```