Exercise Sheet 3 for Lecture Parallel Storage Systems

Deadline: 2024-05-21, 23:59

Prof. Dr. Michael Kuhn (michael.kuhn@ovgu.de)
Michael Blesel (michael.blesel@ovgu.de)
Parallel Computing and I/O • Institute for Intelligent Cooperating Systems
Faculty of Computer Science • Otto von Guericke University Magdeburg
https://parcio.ovgu.de

In the following tasks, you will analyze and optimize the provided application. First, modify the application in a way that after each call of `pwrite`, a synchronization via `fsync` happens. This ensures that the checkpoints are actually written to the storage device. Save the modified application as `checkpoint-fsync.c`. Now analyze the application using the following tools. Make sure to perform all measurements on one of the cluster's compute nodes.

1. SLURM Usage (20 Bonus Points)

After logging in via SSH, you are on the so-called login node of the cluster. This is primarily intended for managing your data and compiling software. More complex tasks should always be performed on compute nodes.

The cluster uses SLURM for job management, which can be used to gain access to the compute nodes. A distinction must be made between non-interactive and interactive use. In the non-interactive case, you submit so-called jobs, which SLURM executes on the compute nodes. In the interactive case, you reserve a compute node and log in to it.

You can display all available partitions and nodes with `sinfo`. The partition used for the lecture or exercises is called `vl-parcio`. For interactive use, nodes can be reserved with `salloc`, which starts a subshell which keeps the reservation until you leave the subshell with `exit`.

```
$ salloc -p vl-parcio
$ srun --pty bash
```

Both commands can also be combined for temporary allocation.

```
$ srun -p vl-parcio --pty bash
```

By default, a single core (with two threads) is allocated. Additional cores can be requested with the `-c` parameter, specifying the number of threads. For example, a value of 4 corresponds to two cores with two threads each. The allocated core or thread count can be easily checked with the `nproc` command.

```
$ srun -p vl-parcio -c 4 nproc
```

Question: Why are four threads allocated when you request three with `-c`? Write a few sentences of justification for this.

A non-interactive job can be submitted using `sbatch job.slurm`. Active jobs can be displayed with `squeue`; output can also be restricted to own jobs with `squeue --me`. If necessary, a job can also be canceled with `scancel <job-id>`. The following script can serve as a template.

```
#!/bin/bash

# Time limit is ten minutes (see "man sbatch")
#SBATCH --time=10:00
# Run one task on one node
#SBATCH --nodes=1
#SBATCH --ntasks=1
# Make all cores available to our task
#SBATCH --cpus-per-task=48
# Use lecture partition
#SBATCH --partition=vl-parcio
# Redirect output and error output
#SBATCH --output=job.out
#SBATCH --error=job.err

srun hostname
srun nproc
srun ./my-application
```

2. iostat (60 Points)

Use `iostat` to display statistics about the application during runtime. Give a detailed explanation about what you can see in the output. Display how many megabytes per second are written by the application and compare this value to the output of `iostat`. Document the results and explain potential differences. Compare the output of `iostat` for the `fsync` version of the application to the original version and explain your observations.

To use `iostat`, the `sysstat` module needs to be loaded on the cluster using the following command (note the space between `.` and `/`).

```
$ . /opt/spack/pss-2024/env.sh
$ module load sysstat
```

Important: To get correct statistics of your program run on the compute nodes of the cluster you need to execute your program in the `/tmp` directory of a node. If you execute it in your home directory the data will not be written to the local nvme SSD of a node.

3. Performance Analysis and Optimization (90 Points)

Implement additional statistics about the time that the application spends in the `fsync` operation. How much of the total I/O time of the program is taken up by `fsync`? Write down explanations for the observed behavior and give a more detailed explanation about what the `fsync` function does. Evaluate the I/O behavior of the application. Can it be optimized?

Think about how the application could be optimized regarding its I/O performance. Implement and explain your optimizations as `checkpoint-optimized.c` (include good documentation). Consider that your optimized version does not necessarily need to call `fsync` after every `pwrite`. It must still be ensured that, for each iteration, the checkpoint is written to the storage device.

4. Performance Comparison (60 Points)

Now benchmark the application with meaningful measurements. Therefore, you should do measurements for the following three application versions:

1. The original application from the materials
2. The original application with synchronization
3. The optimized version of the application from the last task

For your measurements, do application runs ranging from 1–12 threads (measurements for each number of threads) and choose a suitable number of iterations, where the minimum runtime should not be less than 120 seconds. (Note: You can choose different numbers of iterations for the different measurements. Rough guidelines for the number of iterations are: 100,000 for the original application, 4,000 for the original application with synchronization and 10,000 for the optimized application.)

Visualize your results with suitable graphs and write down your insights about the results. Your visualization should make it easy to compare the IOPS and throughput values of the three application versions with varying numbers of threads.

Submission

We will count your last commit on the `main` branch of your repository before the exercise deadline as your submission. In the root directory of the repository, we expect a `PSS-2024-Exercise-03-Materials` directory with the following contents:

- A file `group.txt` with your group members (one per line) in the following format:

  ```
  Erika Musterfrau <erika.musterfrau@example.com>

  Max Mustermann <max.mustermann@example.com>
  ```

- Task 1: A file `slurm-answers.txt` with your answers.
- Task 2: A text file with the output of your application and `iostat` as well as your explanations for potential differences called `iostat.txt`.

- Task 3: The modified source code for the three applications (`checkpoint.c`, `checkpoint-fsync.c` and `checkpoint-optimized.c`). Furthermore, include a file called `analysis.txt` with your reasoning for the optimizations.
- Task 4: A PDF file with graphs for your measurements and explanations called `performance.pdf`.