CNeRF Revisited + The Sim Theory Scheduler: A Case Study

Goal

In December 2024, Simulation Theory integrated the Sim Theory proprietary Scheduler technology into the CNeRF project[1]. This integration resulted in approximately a **39% increase** in CPU occupancy.

Once the performance boost the Simulation Theory Scheduler achieved was clear, the team wanted to find out how the Scheduler compared to other solutions. All publicly available threading and scheduling solutions were assessed to find out which had a feature set closest to the Sim Theory Scheduler for the most fair comparison.

The requirements were:

- ➔ Multiple options for parallelization of work
- ➔ CPU agnostic
- ➔ Readily available
- ➔ Provides a C/C++ API

Intel oneAPI[2] and oneTBB[3] were chosen because they meet the requirements in the following ways:

- ➔ Support a work stealing job system
- ➔ Support parallel for
- ➔ Support x64 and community support for aarch64
- ➔ Readily available and our team has experience using it
- ➔ C++ API

[1] https://github.com/rafaelanderka/cNeRF
[2] https://www.intel.com/content/www/us/en/docs/oneapi/programming-guide/2024-1/intel-oneapi-threading-building-blocks-onetbb.html
[3] https://github.com/uxlfoundation/oneTBB

Hypothesis

Simulation Theory's Scheduler technology can perform better than Intel's oneTBB when integrated with CNeRF at the top-level domain of the application. Neither Simulation Theory's Scheduler nor Intel's oneTBB should disrupt or modify the fork and join nature of OpenMP within PyTorch.

Out of the box, running on the CPU, CNeRF was achieving an effective CPU occupancy of 39.5% using about 25 of the 64 available logical CPU cores. This performance is due to PyTorch, its use of OpenMP, and the parallelization OpenMP provides.

Integration

Simulation Theory approached integration with CNeRF in the following ways:

Simulation Theory's Scheduler

➔ **Subdivide the work into equally-sized blocks.** The Scheduler was initialized with *N* threads. All work was then broken up inside the Scheduler into *N* evenly-sized blocks and processed one block per thread. The result of this is a small number of large tasks processing simultaneously on *N* threads.

➔ **Place all work in a work buffer using an atomic incrementor.** The Scheduler was initialized with *N* threads. An atomic incrementor was used to gate access by the *N* threads to the data buffer. This is the most efficient use of the Scheduler and results in fine-grained tasks processing simultaneously on *N* threads.

Intel's oneTBB

➔ **Use the oneTBB job system to create equally-sized blocks.** oneTBB was initialized with *N* threads. All work was divided into jobs and submitted to oneTBB for execution. The result of this is a small number of large tasks processing simultaneously on *N* threads.

➔ **Use the oneTBB parallel for solution to mimic the approach that was used with the Simulation Theory Scheduler and an atomic incrementor.** This effectively resulted in fine-grained tasks processing simultaneously on *N* threads.

Results

Building on top of Simulation Theory's prior integration of the Scheduler, integrating oneTBB into CNeRF only required a couple of hours of additional work. For all of the following testing, the Simulation Theory Scheduler and Intel's oneTBB were initialized with the same number of threads.

CPU Occupancy

The overall CPU occupancy was **39.5%** on 25.267 out of 64 logical CPU cores in the baseline implementation of CNeRF.

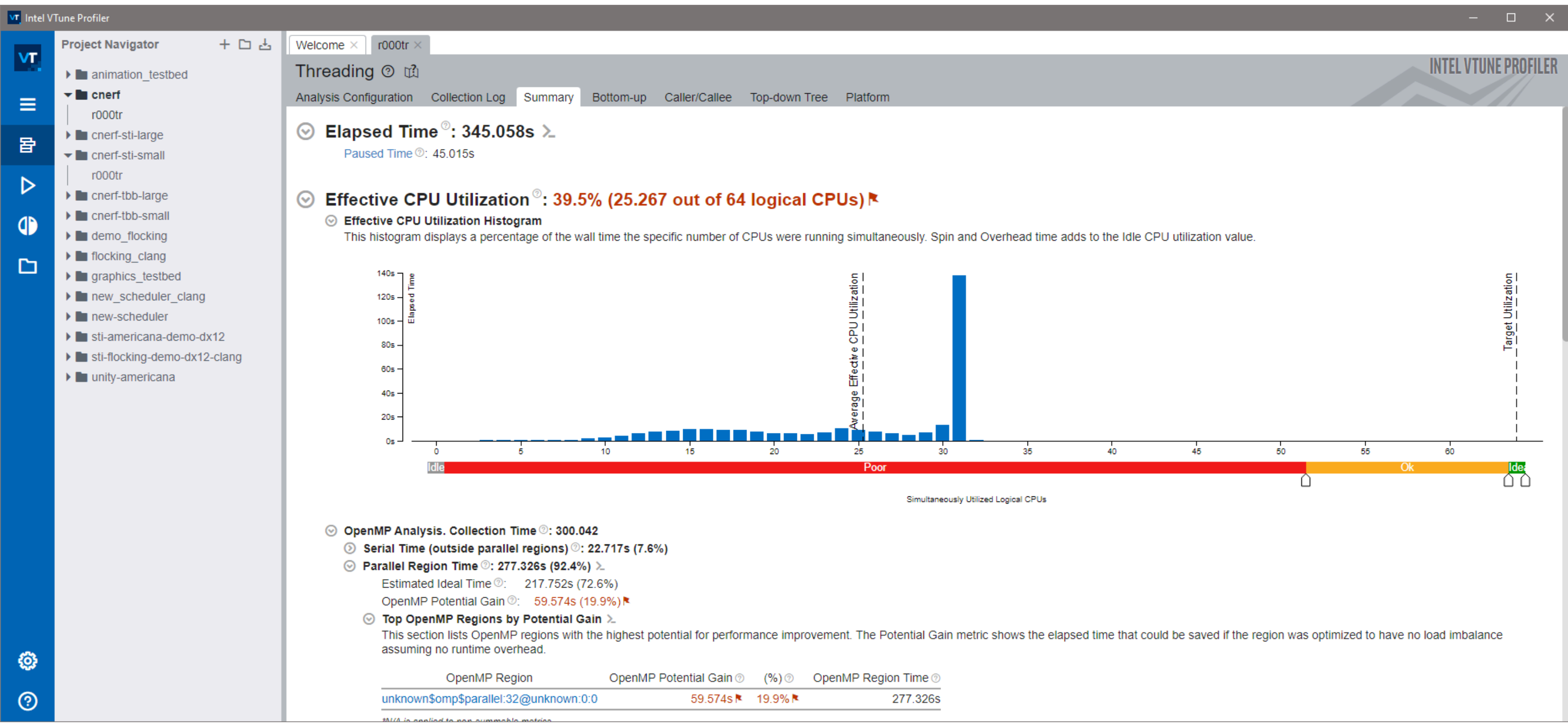
Large-block Work Distribution

- ➔ Using the large-block work distribution model of Intel oneTBB, CPU occupancy **improved** to **56.8%** on 36.361 out of 64 logical CPU cores. This was an **improvement of 30.5%**.
- ➔ Using the large-block work distribution model of the Simulation Theory Scheduler implementation, CPU occupancy **improved** to **57.4%** on 36.735 out of 64 logical CPU cores. This was an **improvement of 31.2%**.
- ➔ Simulation Theory's Scheduler technology **beat** Intel's oneTBB performance **by 1%** using the large-block work distribution method.

Fine-Grained Work Distribution

- ➔ Using the fine-grained work distribution model of Intel oneTBB, CPU occupancy **improved** to **57.4%** of 36.760 out of 64 logical CPU cores. This was an **improvement of 31.3%**.
- ➔ Using the fine-grained work distribution model of the Simulation Theory Scheduler implementation, CPU occupancy **improved** to **65.2%** on 41.743 out of 64 logical CPU cores. This was an **improvement of 39.47%**.
- ➔ Simulation Theory's Scheduler technology **beat** Intel's oneTBB performance **by 11.9%** using the fine-grained work distribution method.

CNeRF Original Performance Graph[4]



Elapsed Time: 345.058s

Paused Time: 45.015s

Effective CPU Utilization: 39.5% (25.267 out of 64 logical CPUs)

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

OpenMP Analysis. Collection Time: 300.042

Serial Time (outside parallel regions): 22.717s (7.6%)

Parallel Region Time: 277.326s (92.4%)

Estimated Ideal Time: 217.752s (72.6%)

OpenMP Potential Gain: 59.574s (19.9%)

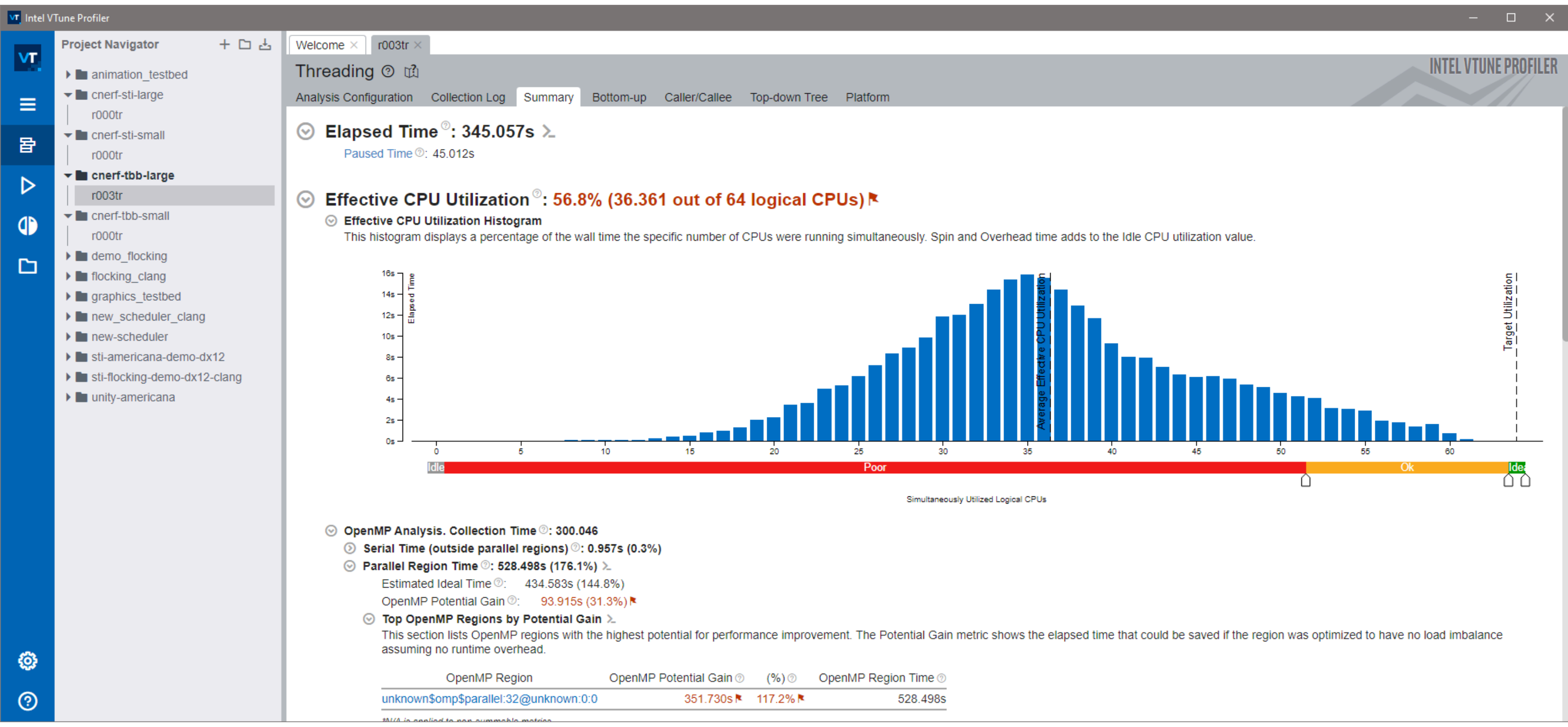
Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
unknown$omp$parallel:32@unknown:0:0	59.574s	19.9%	277.326s

[4] Simulation Theory used Intel V-Tune to capture these and the following performance metrics from the testing.

CNeRF Intel oneTBB Large, Equally-Sized Tasks Performance Graph



Elapsed Time: 345.057s

Paused Time: 45.012s

Effective CPU Utilization: 56.8% (36.361 out of 64 logical CPUs)

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

OpenMP Analysis. Collection Time: 300.046

Serial Time (outside parallel regions): 0.957s (0.3%)

Parallel Region Time: 528.498s (176.1%)

Estimated Ideal Time:	434.583s (144.8%)
OpenMP Potential Gain:	93.915s (31.3%)

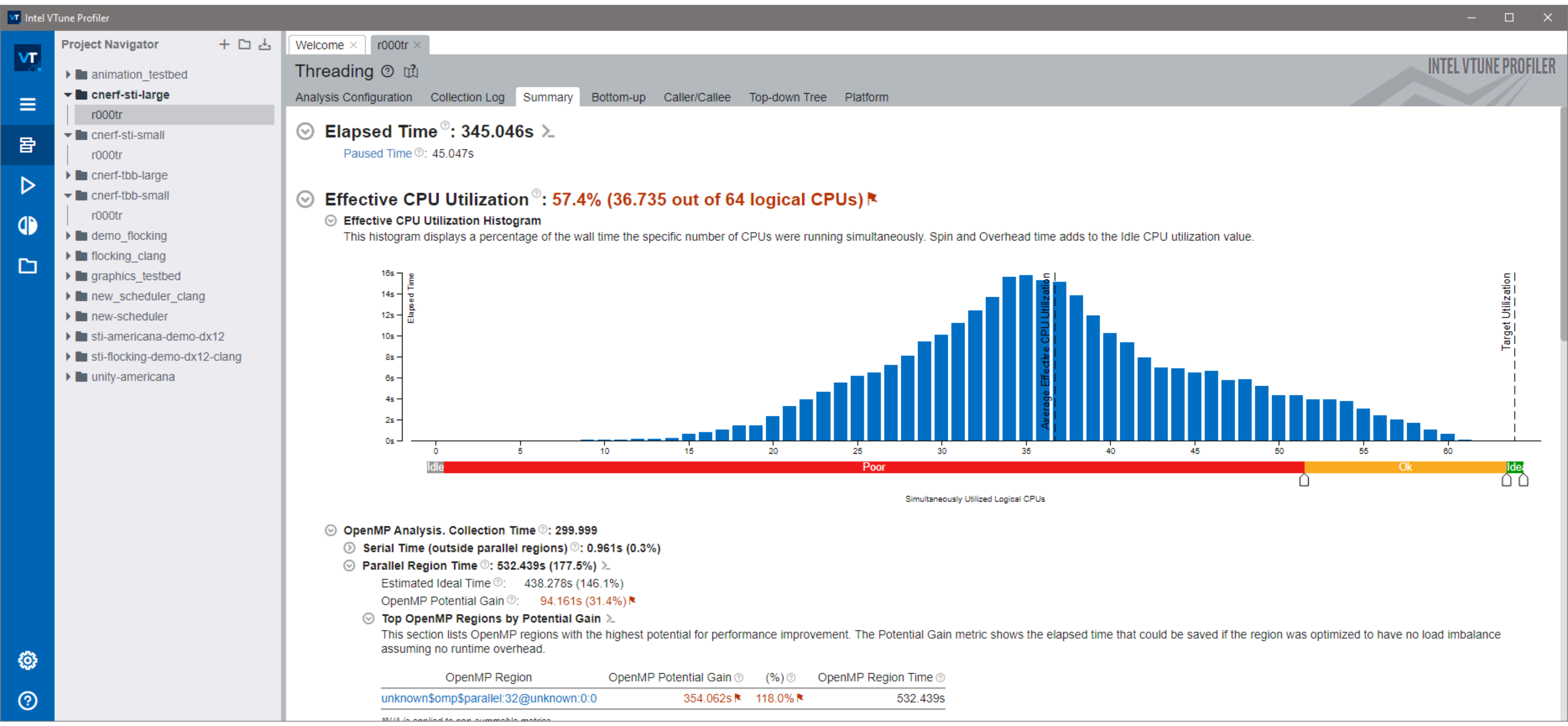
Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
unknown$omp$parallel:32@unknown:0:0	351.730s	117.2%	528.498s

*N/A is applied to non-summable metrics

CNeRF Simulation Theory Large, Equally-Sized Tasks Performance Graph



Elapsed Time: 345.046s

Paused Time: 45.047s

Effective CPU Utilization: 57.4% (36.735 out of 64 logical CPUs)

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

OpenMP Analysis. Collection Time: 299.999

Serial Time (outside parallel regions): 0.961s (0.3%)

Parallel Region Time: 532.439s (177.5%)

Estimated Ideal Time: 438.278s (146.1%)

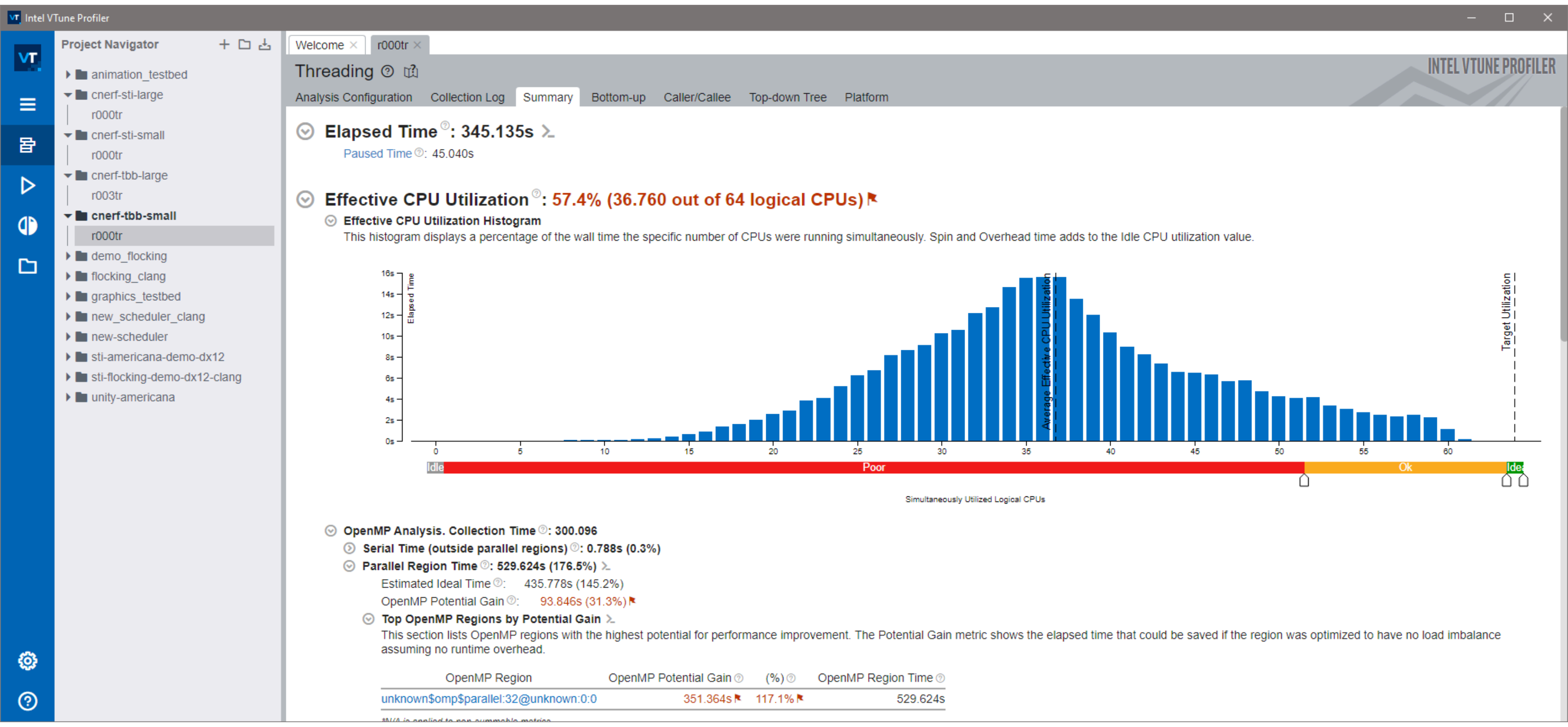
OpenMP Potential Gain: 94.161s (31.4%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
unknown$omp$parallel:32@unknown:0:0	354.062s	118.0%	532.439s

CNeRF Intel oneTBB Small Tasks Performance Graph



Elapsed Time: 345.135s

Paused Time: 45.040s

Effective CPU Utilization: 57.4% (36.760 out of 64 logical CPUs)

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

OpenMP Analysis. Collection Time: 300.096

Serial Time (outside parallel regions): 0.788s (0.3%)

Parallel Region Time: 529.624s (176.5%)

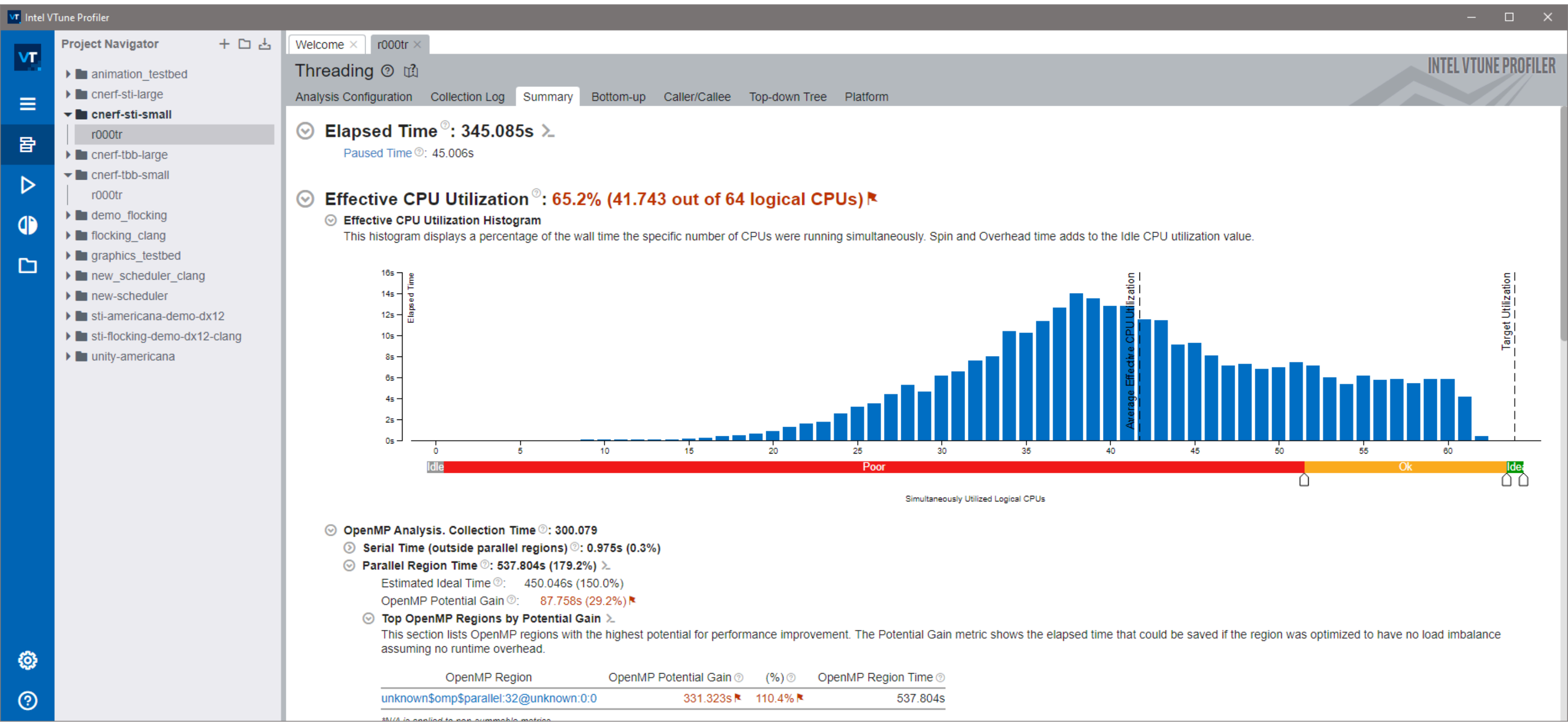
Estimated Ideal Time:	435.778s (145.2%)
OpenMP Potential Gain:	93.846s (31.3%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
unknown$omp$parallel:32@unknown:0:0	351.364s	117.1%	529.624s

CNeRF Simulation Theory Small Tasks Performance Graph



Elapsed Time: 345.085s

Paused Time: 45.006s

Effective CPU Utilization: 65.2% (41.743 out of 64 logical CPUs)

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

OpenMP Analysis. Collection Time: 300.079

Serial Time (outside parallel regions): 0.975s (0.3%)

Parallel Region Time: 537.804s (179.2%)

Estimated Ideal Time: 450.046s (150.0%)

OpenMP Potential Gain: 87.758s (29.2%)

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
unknown$omp$parallel:32@unknown:0:0	331.323s	110.4%	537.804s

Author: Randy Culley, CTO
sales@simtheoryinc.com
simtheoryinc.com