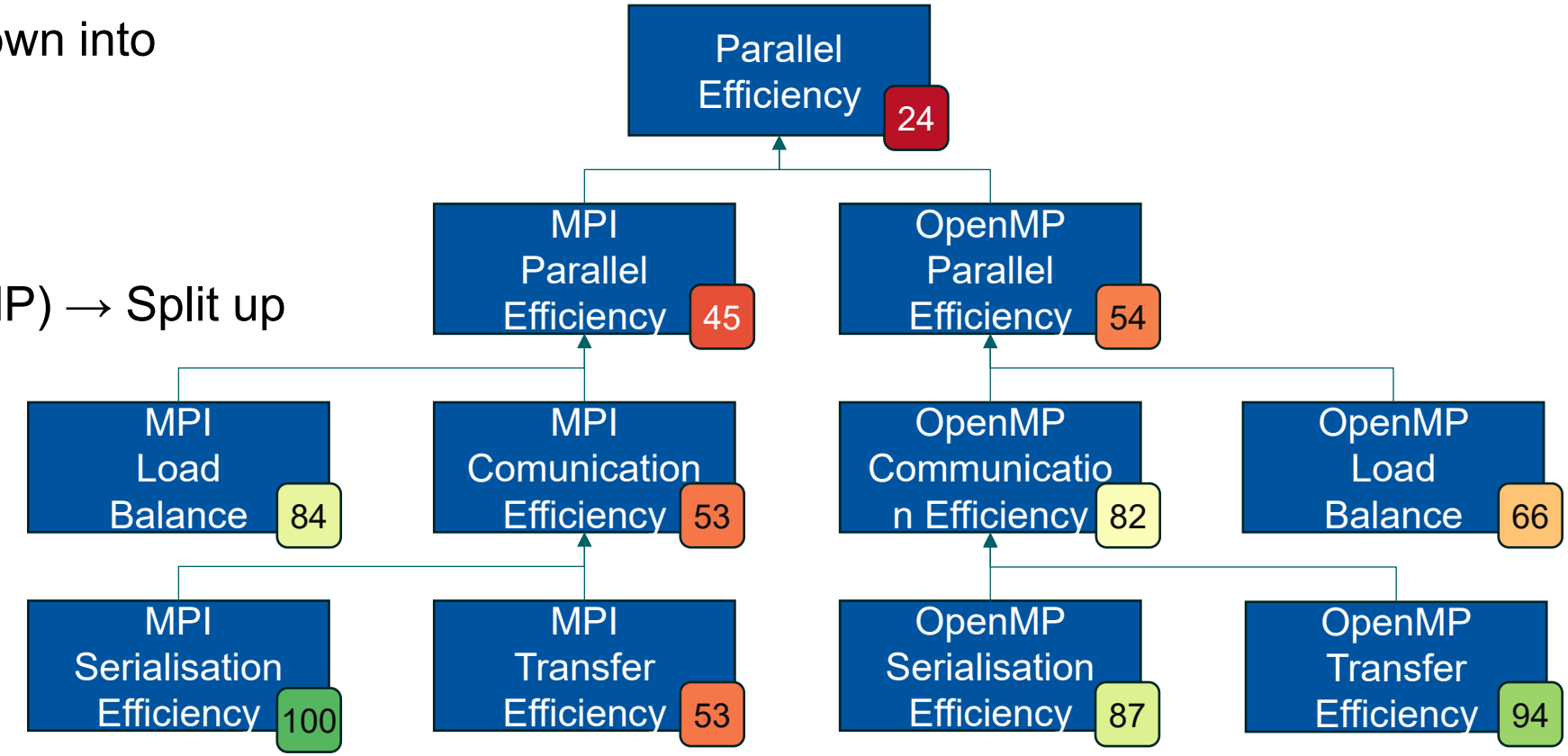
# OTF-CPT

On-the-fly Critical Path Tool


RWTH AACHEN UNIVERSITY

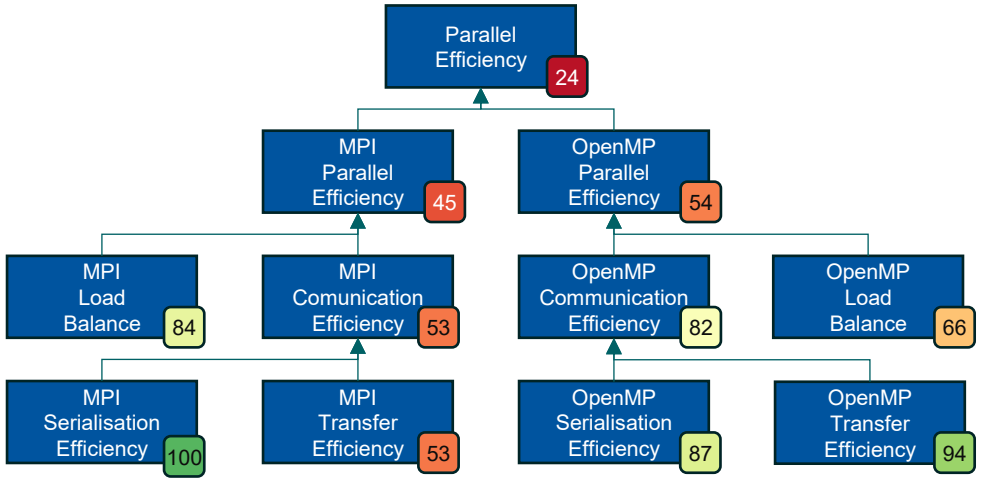
# Hybrid Model Factors

- Highlight issues in the parallel structure of an application
- Parallel Efficiency breaks down into
  - Load balance
  - Serialization
  - Transfer
- Hybrid Setups (MPI+OpenMP) → Split up efficiencies
- Child metrics multiply to parent metric



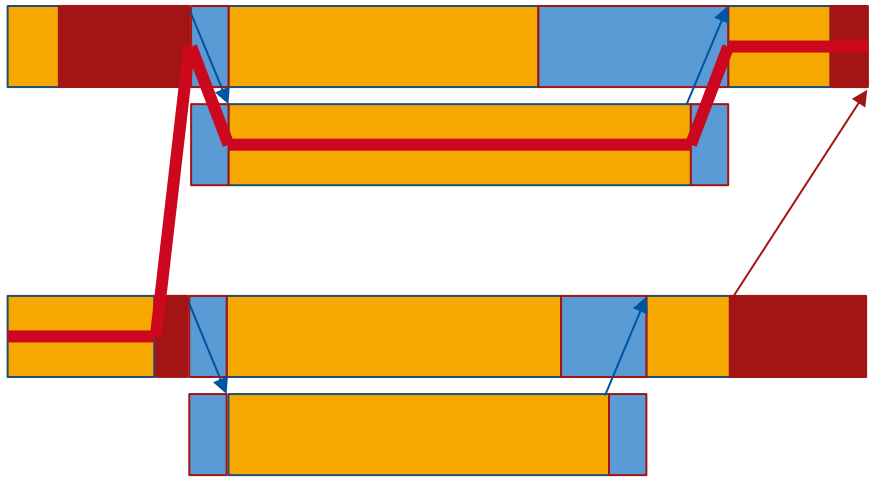
# What to measure?

- *Useful time*: execution time outside parallel runtimes
  - Track execution time on each thread excluding time inside MPI / OpenMP runtimes
- *Real runtime*: observed execution time
  - Track wall clock time from start to end.
- *Ideal runtime*: execution time on an ideal machine with 0 communication cost (inf. BW / 0 lat)
  - Track useful time on critical path → assumes 0 communication cost



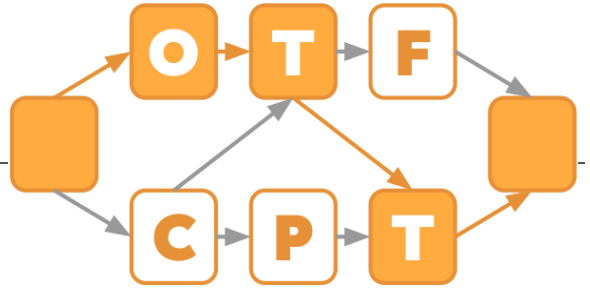
# The Critical Path

- We measure the length of the *Critical Path* of an application
  - Longest chain of dependent useful computations
  - Defines the total runtime → optimizations outside of the critical path do not directly decrease total runtime
  - Useful computation on the Critical Path vs total runtime shows the cost of synchronization

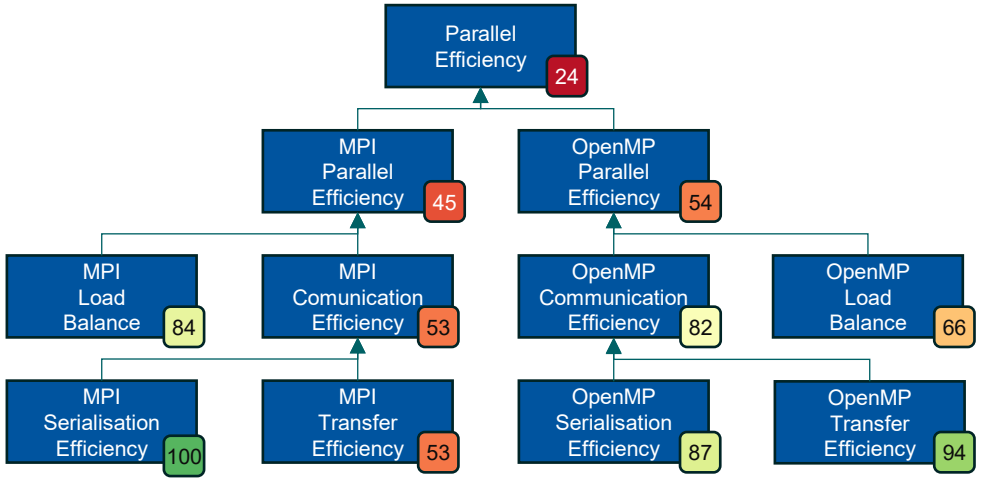


Application
MPI Comm
OpenMP

# On The Fly Critical Path Tool (OTF-CPT)




- Forward-only analysis
  - we only need the metrics of the critical path, but not the concrete path
- Times are measured on thread level and propagated on the critical path
- Relevant metrics: useful computation, time outside the OpenMP runtime
- Relevant critical paths: global, process-local, thread-local
- Calculate hybrid model factors at the end of execution
  - High level overview over parallel application performance






IT Center | RWTH AACHEN UNIVERSITY

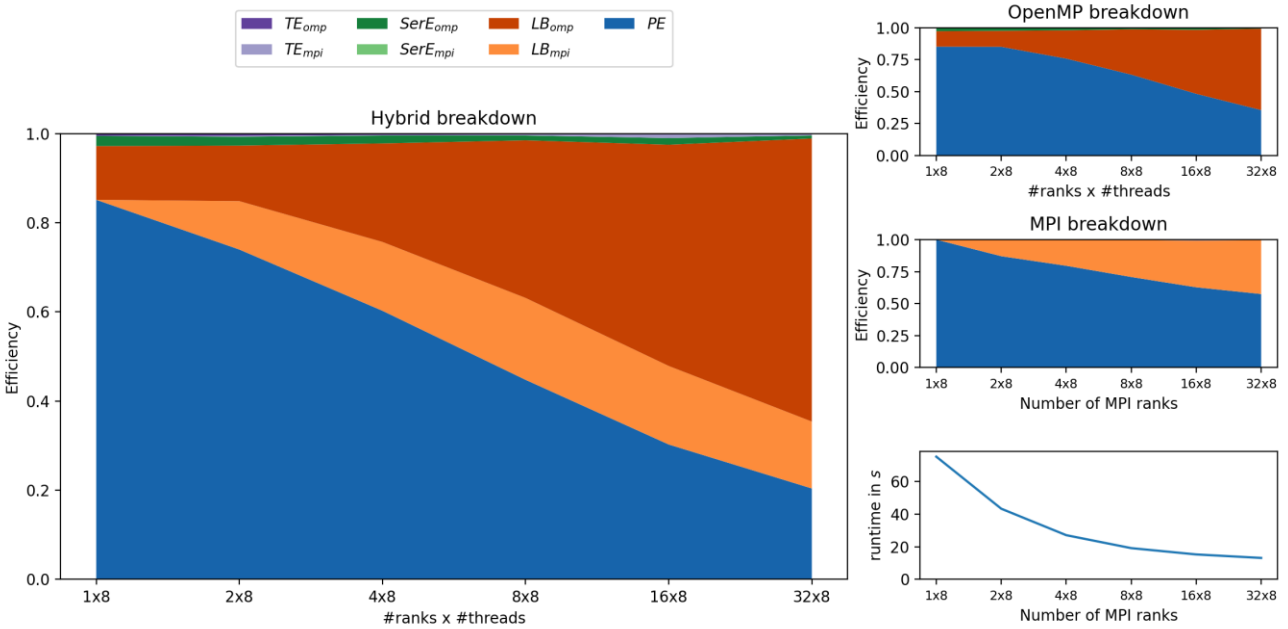
# OTF-CPT: Usage

- Run your parallel application normally
  - No recompilation necessary
- Tool provided as library: libOTFCPT.so
- Use LD_PRELOAD to run application with the tool
- Set OMP_TOOL_LIBRARIES for OpenMP-Support
- Outputs statistics and the hybrid model factors

```
$ export OMP_NUM_THREADS=8
$ mpirun -np 32 \
env LD_PRELOAD=./libOTFCPT.so \
OMP_TOOL_LIBRARIES=./libOTFCPT.so \
./hybrid-application

... APP OUTPUT
--------CritPath Analysis Tool results:--------
=> Number of processes:           32
=> Number of threads:             256
=> Average Computation (in s):    34.797
=> Total runtime (in s):          37.146

----------------POP metrics----------------
Parallel Efficiency:                  0.937
  Load Balance:                       0.976
  Communication Efficiency:           0.960
    Serialisation Efficiency:         0.989
    Transfer Efficiency:              0.971
  MPI Parallel Efficiency:            0.956
    MPI Load Balance:                 0.981
    MPI Communication Efficiency:     0.974
      MPI Serialisation Efficiency:   0.997
      MPI Transfer Efficiency:        0.977
  OMP Parallel Efficiency:            0.980
    OMP Load Balance:                 0.995
    OMP Communication Efficiency:     0.986
      OMP Serialisation Efficiency:   0.992
      OMP Transfer Efficiency:        0.994
-------------------------------------------
```

# Visualization

- Outputs can be visualized using a provided python script
  - Expects run outputs with the name <PREFIX>-<RANKS>x<THREADS>.* in a single folder.
  - Will search for the OTF-CPT output and parse it
  - Usage: `python CPT-plot.py -o <OUT_DIR> -p <PREFIX> experiment_directory`




| #Ranks x #Threads | 1x8 | 2x8 | 4x8 | 8x8 | 16x8 | 32x8 |
|---|---|---|---|---|---|---|
| Global Efficiency | 85.2 | 73.9 | 59.1 | 41.8 | 26.2 | 15.2 |
| Parallel Efficiency | 85.2 | 74.0 | 60.2 | 44.7 | 30.3 | 20.4 |
| Load Balance | 87.6 | 76.0 | 61.6 | 45.4 | 31.0 | 20.6 |
| Communication Efficiency | 97.2 | 97.3 | 97.8 | 98.3 | 97.5 | 98.8 |
| Serialisation Efficiency | 97.7 | 97.9 | 98.2 | 98.8 | 98.5 | 99.2 |
| Transfer Efficiency | 99.5 | 99.4 | 99.6 | 99.5 | 99.0 | 99.6 |
| MPI Parallel Efficiency | 100.0 | 87.1 | 79.4 | 70.7 | 62.7 | 57.5 |
| MPI Load Balance | 100.0 | 87.2 | 79.6 | 70.9 | 63.2 | 57.6 |
| MPI Communication Efficiency | 100.0 | 99.8 | 99.8 | 99.7 | 99.2 | 99.7 |
| MPI Serialisation Efficiency | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| MPI Transfer Efficiency | 100.0 | 99.8 | 99.9 | 99.8 | 99.2 | 99.7 |
| OMP Parallel Efficiency | 85.2 | 85.0 | 75.8 | 63.2 | 48.3 | 35.5 |
| OMP Load Balance | 87.6 | 87.2 | 77.4 | 64.1 | 49.1 | 35.8 |
| OMP Communication Efficiency | 97.2 | 97.5 | 97.9 | 98.6 | 98.3 | 99.1 |
| OMP Serialisation Efficiency | 97.7 | 98.0 | 98.2 | 98.9 | 98.5 | 99.3 |
| OMP Transfer Efficiency | 99.5 | 99.5 | 99.7 | 99.8 | 99.8 | 99.9 |
| Computational Scalability | 100.0 | 99.9 | 98.1 | 93.5 | 86.5 | 74.7 |

# Live Demo: NPB

# Options and Region of Interest

- Environment variable OTFCPT_OPTIONS controls options

export OTFCPT_OPTIONS="verbose=1 stopped=1"

- Start and Stop of tool
  - MPI_Pcontrol or omp_control_tool (for OpenMP-only applications)
  - Currently only a single pair of start/stop markers possible

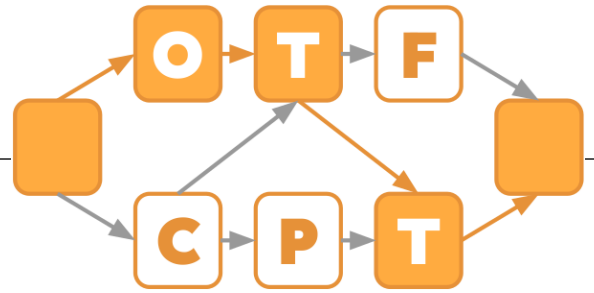
```
MPI_Pcontrol(1); // start
// region of interest
MPI_Pcontrol(0); // stop

omp_control_tool(omp_control_tool_start, 0, NULL); // start
// region of interest
omp_control_tool(omp_control_tool_stop, 0, NULL); // stop
```

| OTFCPT_OPTIONS | | |
|---|---|---|
| **Flag Name** | **Default Value** | **Description** |
| stopped | 0 | Delay the start of measurement until a start marker is encountered |
| data_path | stdout | Write metric data to "<data_path>-<#procs>x<#threads>.txt". Special values are "stdout" and "stderr". Overwrites the file without checking. |
| log_path | stdout | Write logging output to "<log_path>.<pid>". Special values are "stdout" and "stderr". Only relevant with verbose=1 |
| verbose | 0 | Print additional statistics. |
| enable | 1 | Use OTF-CPT during execution. |

# Future Developments

- GPU support
  - target regions using OMPT
  - CUDA events using CUPTI
- Dependent metrics
  - E.g. Hardware counters propagated along the critical path
- Additional OpenMP focused metrics
  - Focus on tasking
- More profiling information
  - Region markers
- Multiple starts and stops

# Available on Github and MN5




- The OTF-CPT is available on github

- Also available on MN5 for this workshop
  - Available in /gpfs/scratch/nct_362/RWTH/OTF-CPT
  - Modules in /gpfs/scratch/nct_362/RWTH/modules
    - **`module use /gpfs/scratch/nct_362/RWTH/modules`**
    - sets paths so you can use libOTFCPT.so
  - Prebuilt for:
    - intel/2023.2.0, impi/2021.10.0
    - intel/2025.2, impi/2021.10.0
    - Intel/2023.2.0, openmpi/4.1.5
  - Please ask for other compiler/MPI combinations (or build yourself)

https://github.com/RWTH-HPC/OTF-CPT