

LA-UR-21-26855

Approved for public release; distribution is unlimited.

Title: MCNP6.2 Benchmarks: Scaling and Hardware Considerations

Author(s): Grieve, Tristan Sumner

Intended for: 2021 MCNP User Symposium, 2021-07-12/2021-07-16 (Los Alamos, New Mexico, United States)

Issued: 2021-08-10 (rev.1)

Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

MCNP6.2 Benchmarks: Scaling and Hardware Considerations

Avery Grieve

XCP-3 (Monte Carlo Codes)

2021 MCNP User Symposium

July 16, 2021

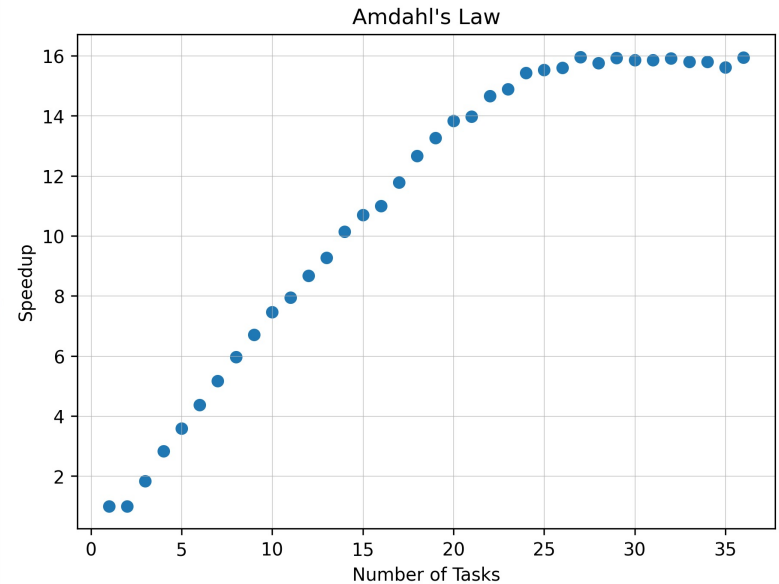
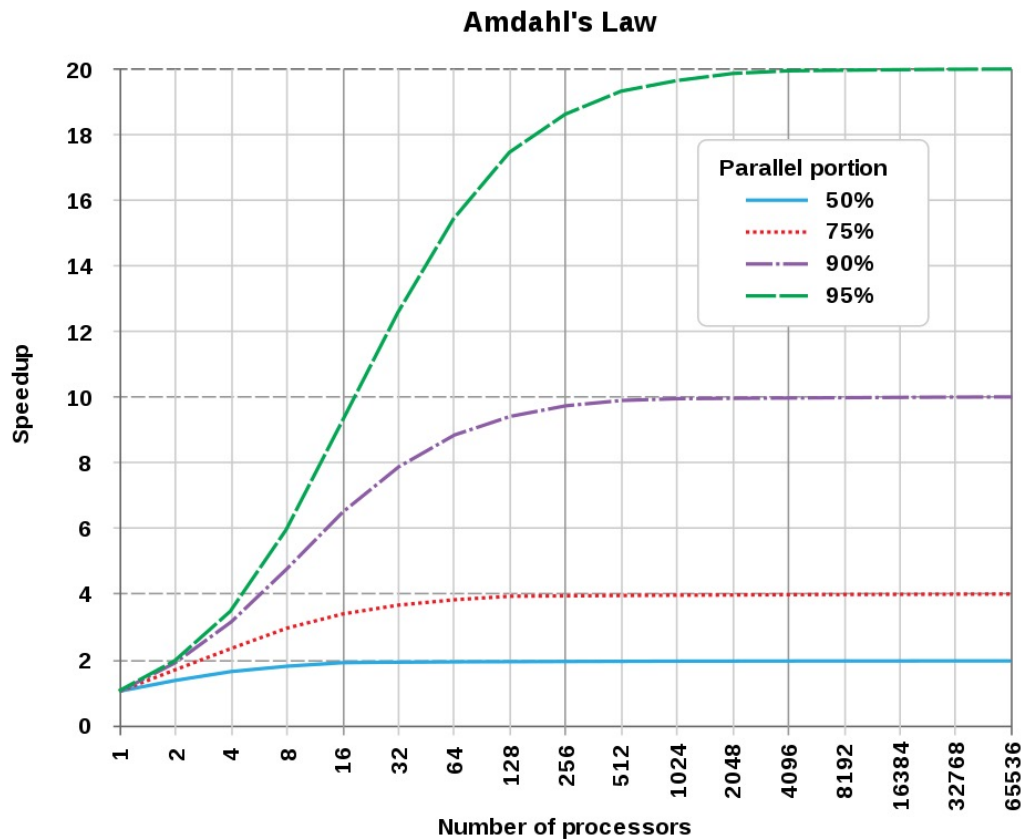
Sections

- Introduction to scaling types
 - Strong and Weak Scaling, Amdahl's and Gustafson's Law
- MCNP Benchmark Tests and Results
 - OMP, MPI results.
 - Scaling falloff with OMP overhead
 - Efficiency Metric
- Memory Benchmark
 - Xeon vs Ryzen
- Key Takeaways

Strong Scaling

- Strong scaling - The number of processors increases while the problem size remains fixed
- For OMP threading:
 - nps = 1e8 (constant)
 - mcnp6 i=input tasks [threads]
- For MPI threading:
 - nps = 1e8
 - mpirun -n [threads] mcnp6.mpi i=input
 - For mpi, the threads are distributed according to sbatch/slurm parameters: --nodes, --ntasks-per-node, etc

Strong Scaling – Amdahl's Law



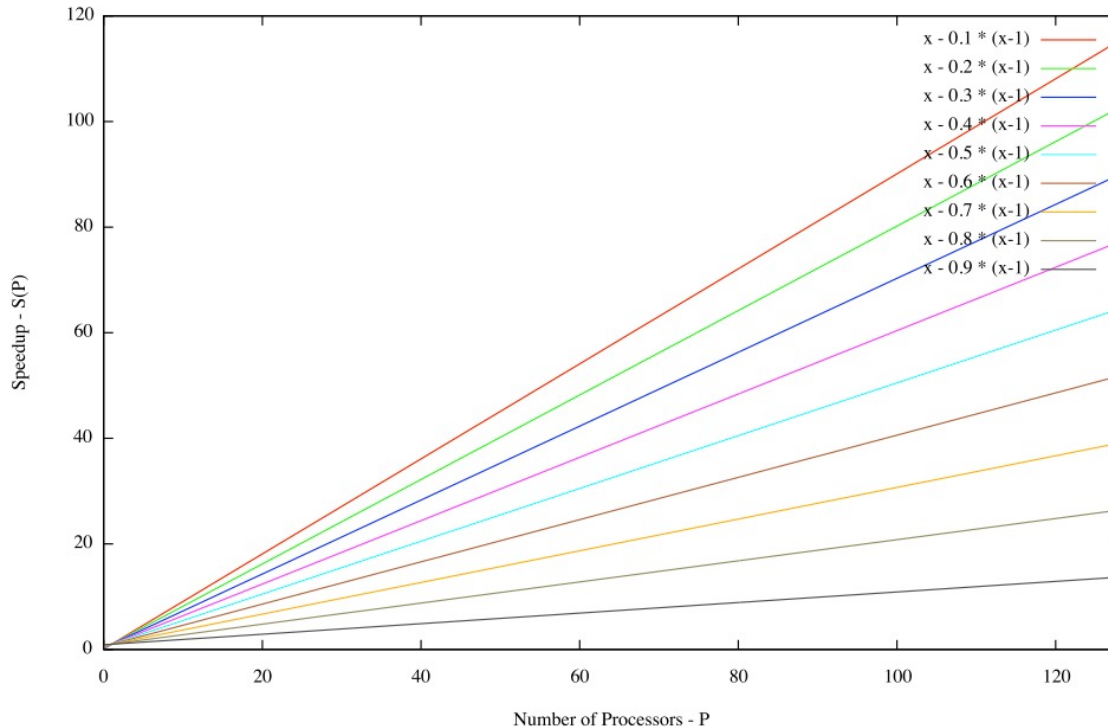
[2]

Weak Scaling

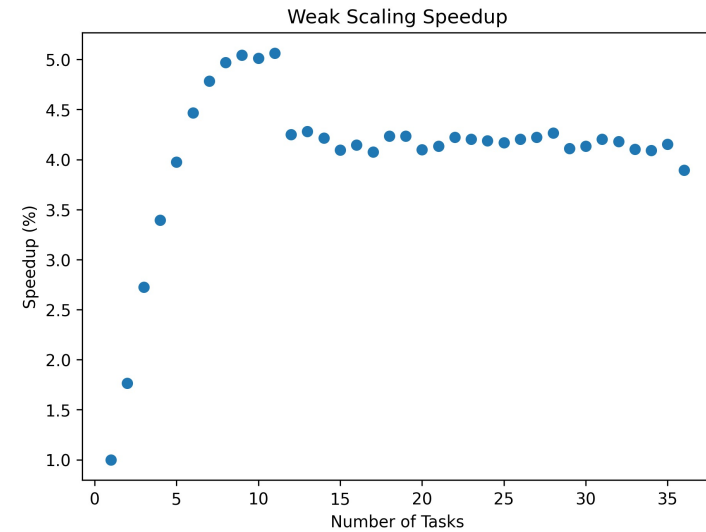
- Weak scaling - The number of processors increases proportionally to the problem size
- For OMP threading:
 - `nps = [threads]*1e6`
 - `mcnp6 i=input tasks [threads]`
- For MPI threading:
 - `nps = [threads]*1e6`
 - `mpirun -n [threads] mcnp6.mpi i=input`
 - For mpi, the threads are distributed according to sbatch/slurm parameters:
--nodes, --ntasks-per-node, etc

Weak Scaling – Gustafson's Law

Gustafson's Law: $S(P) = P \cdot a \cdot (P-1)$



[1]



Test Metrics: fixed source problem

test input for metrics

100 0 2 -999 imp:p=1

200 1 -6.63 -2 imp:p=1

999 0 999 imp:p=0

2 rpp -10 10 -10 10 -2 2

999 so 100

mode p

nps 1e8

sdef par=2 pos=0 0 10 erg=1.3

m1 64157 3 13027 2 31000 3 08016 12

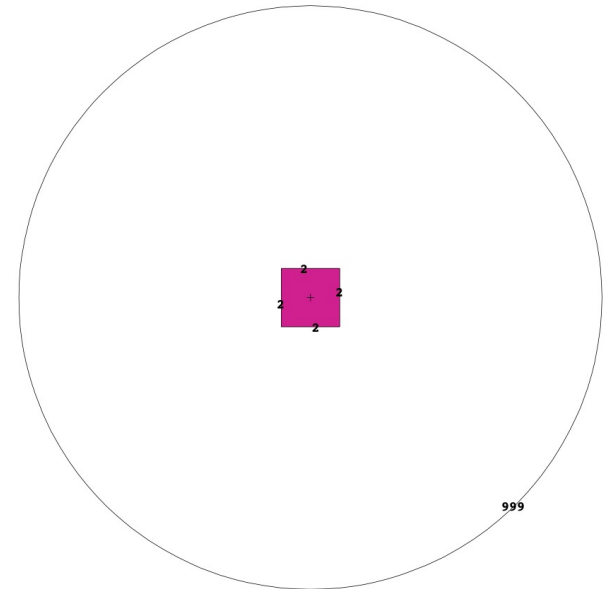
c f8:p 200

c e8 0 700i 1.4

Simply a GAGG(Ce) rectangular prism in a vacuum with a photon source.

```
07/09/21 14:34:33
test input for metrics
```

```
probid = 07/09/21 14:34:13
basis: XY
( 1.000000, 0.000000, 0.000000)
( 0.000000, 1.000000, 0.000000)
origin:
( 0.00, 0.00, 0.00)
extent = ( 100.00, 100.00)
```



Test Metrics: K Code Problem

Lightly modified input from the MCNP Criticality Class.

Longer input summarized:

3x2 array of cans containing plutonium-nitrate solution.

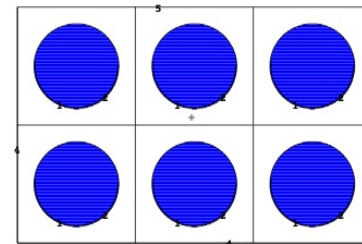
Cans are ~12.5cm ID and ~12.8cm OD.

kcode 20000 1.0 50 150

Note: Weak scaling not performed on K Code problem.

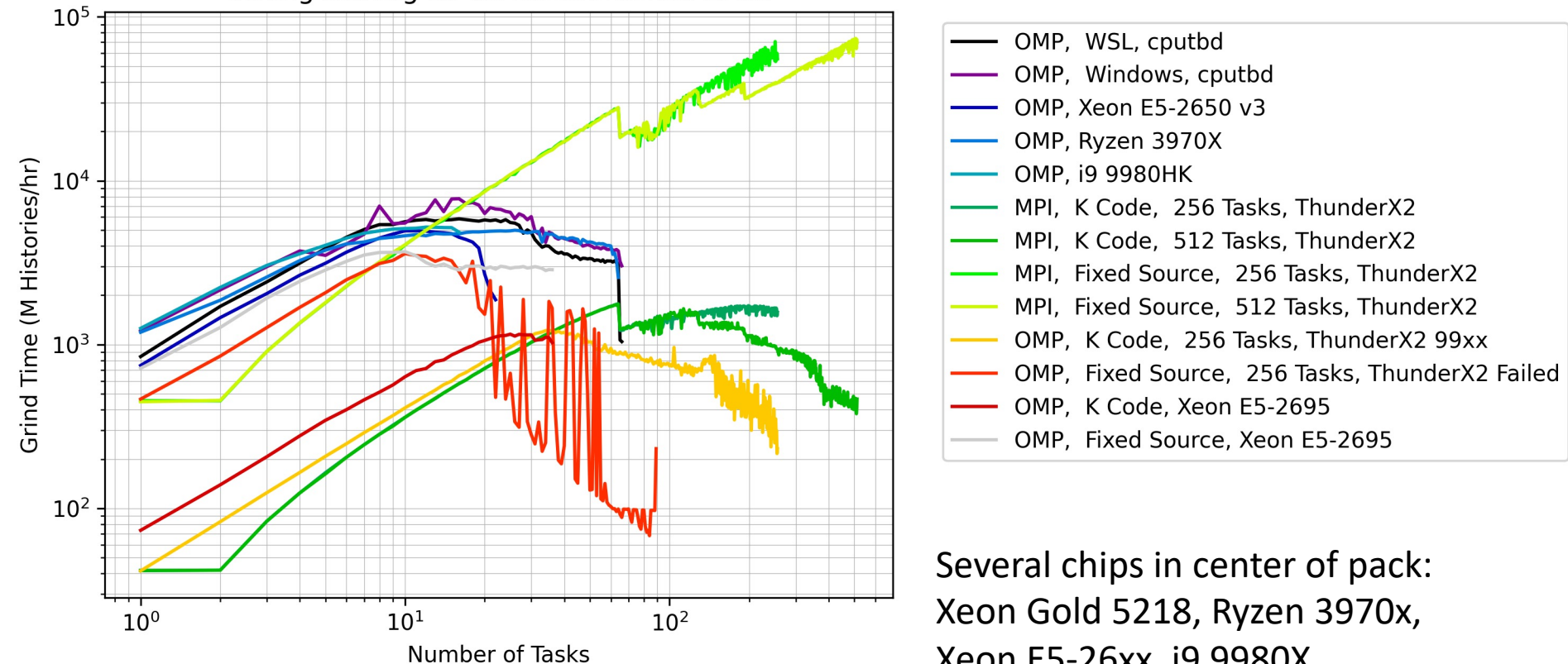
```
07/11/21 23:06:22  
puc6 3 x 2 array of cans
```

```
probid = 07/11/21 23:05:53  
basis: XY  
( 1.000000, 0.000000, 0.000000)  
( 0.000000, 1.000000, 0.000000)  
origin:  
( 34.73, 20.10, 0.00)  
extent = ( 73.96, 73.96)
```



Strong Scaling Data At-A-Glance: Real Time

Strong Scaling for all Benchmarked CPUs

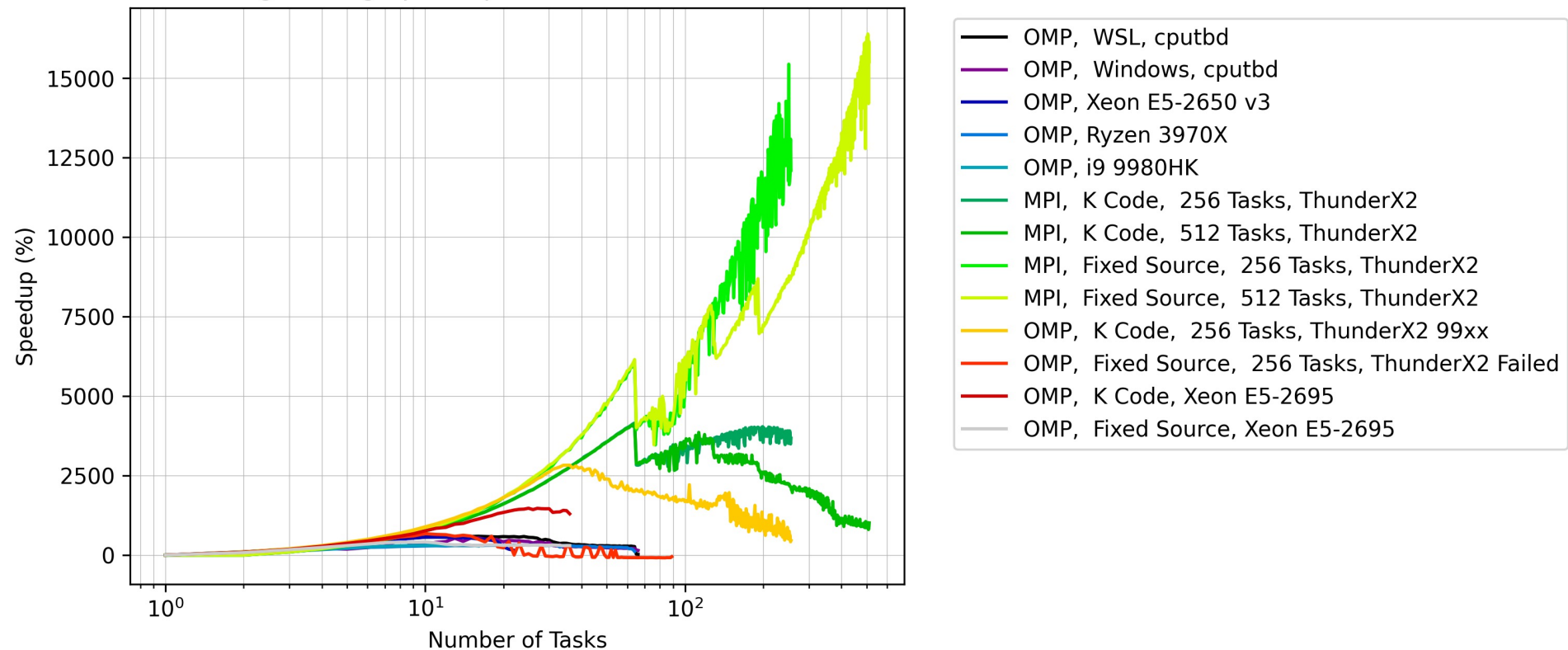


Several chips in center of pack:
Xeon Gold 5218, Ryzen 3970x,
Xeon E5-26xx, i9 9980X

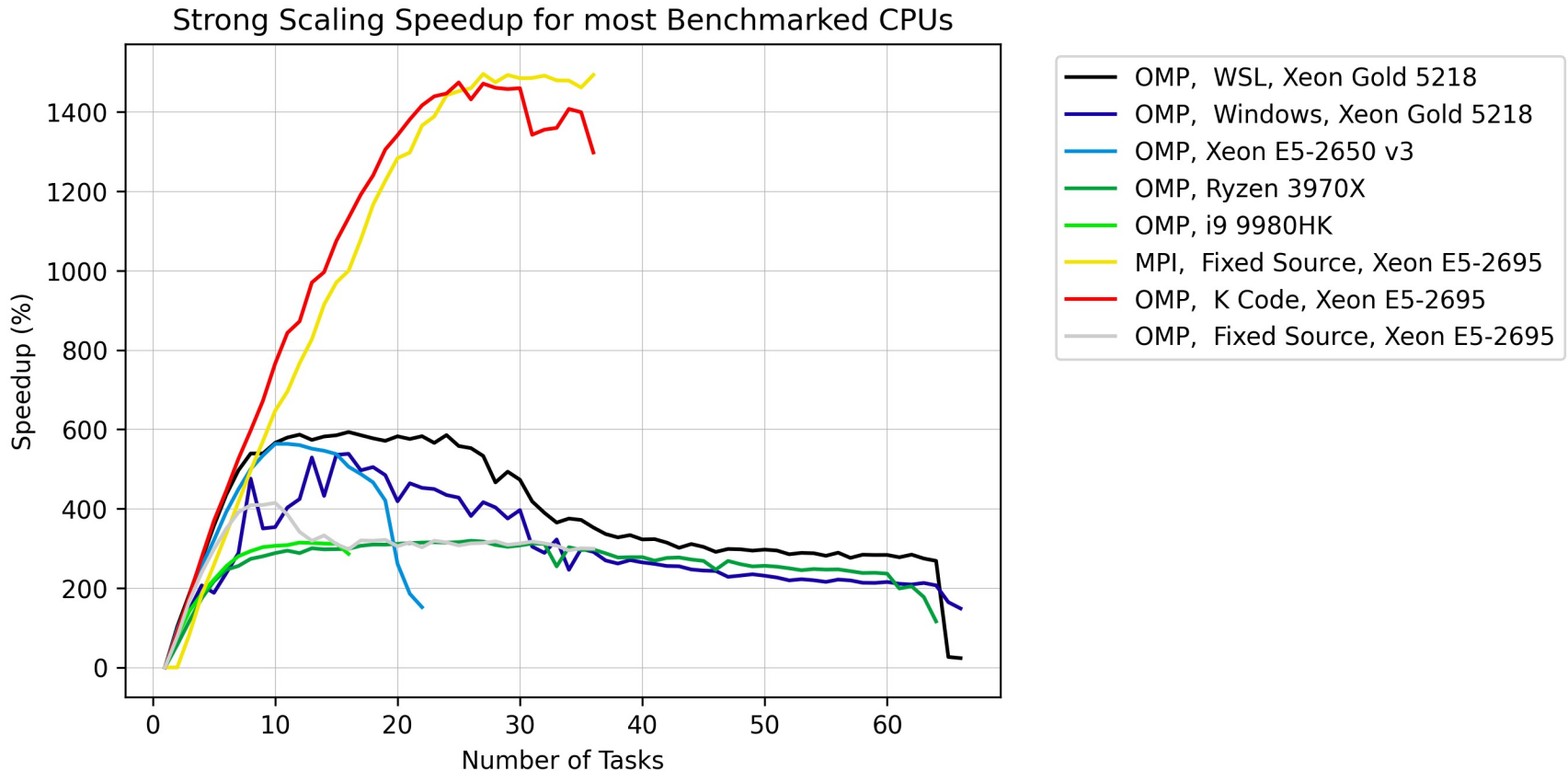
Upper and lower outlier is the
same chip: ThunderX2 (ARM)

Strong Scaling Data At-A-Glance: Speedup

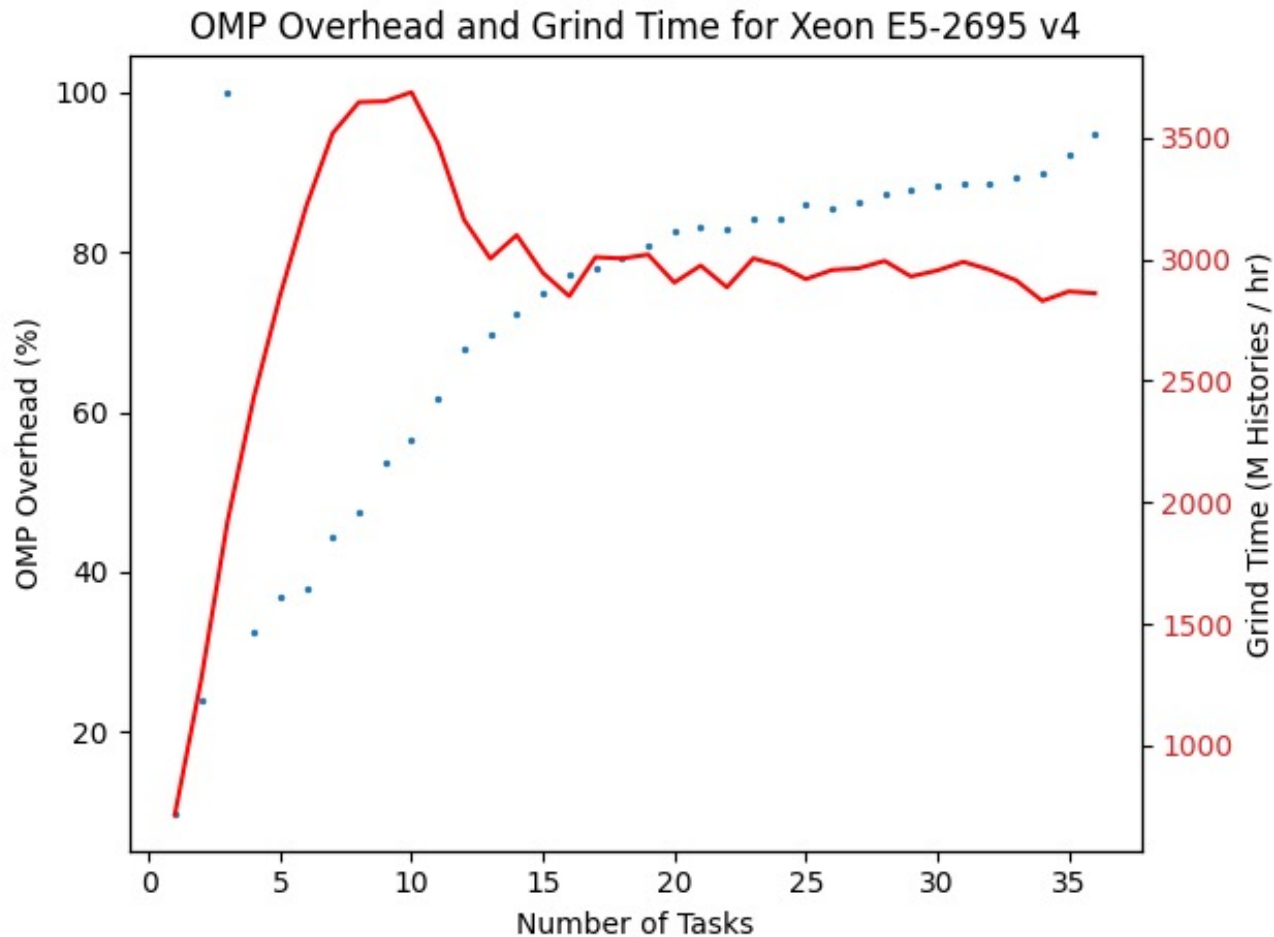
Strong Scaling Speedup for all Benchmarked CPUs



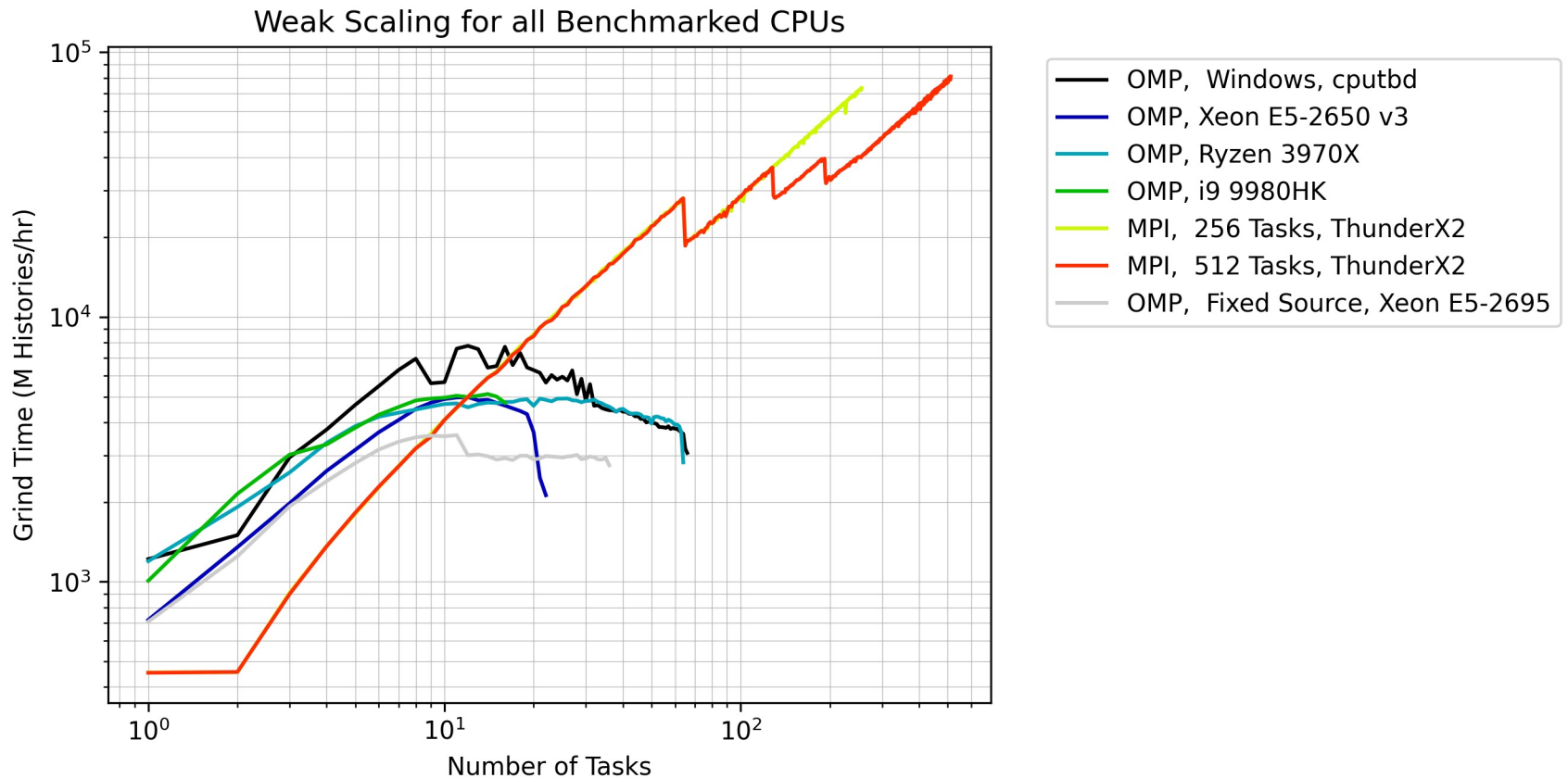
Strong Scaling Data At-A-Glance: Speedup (minus the outliers)



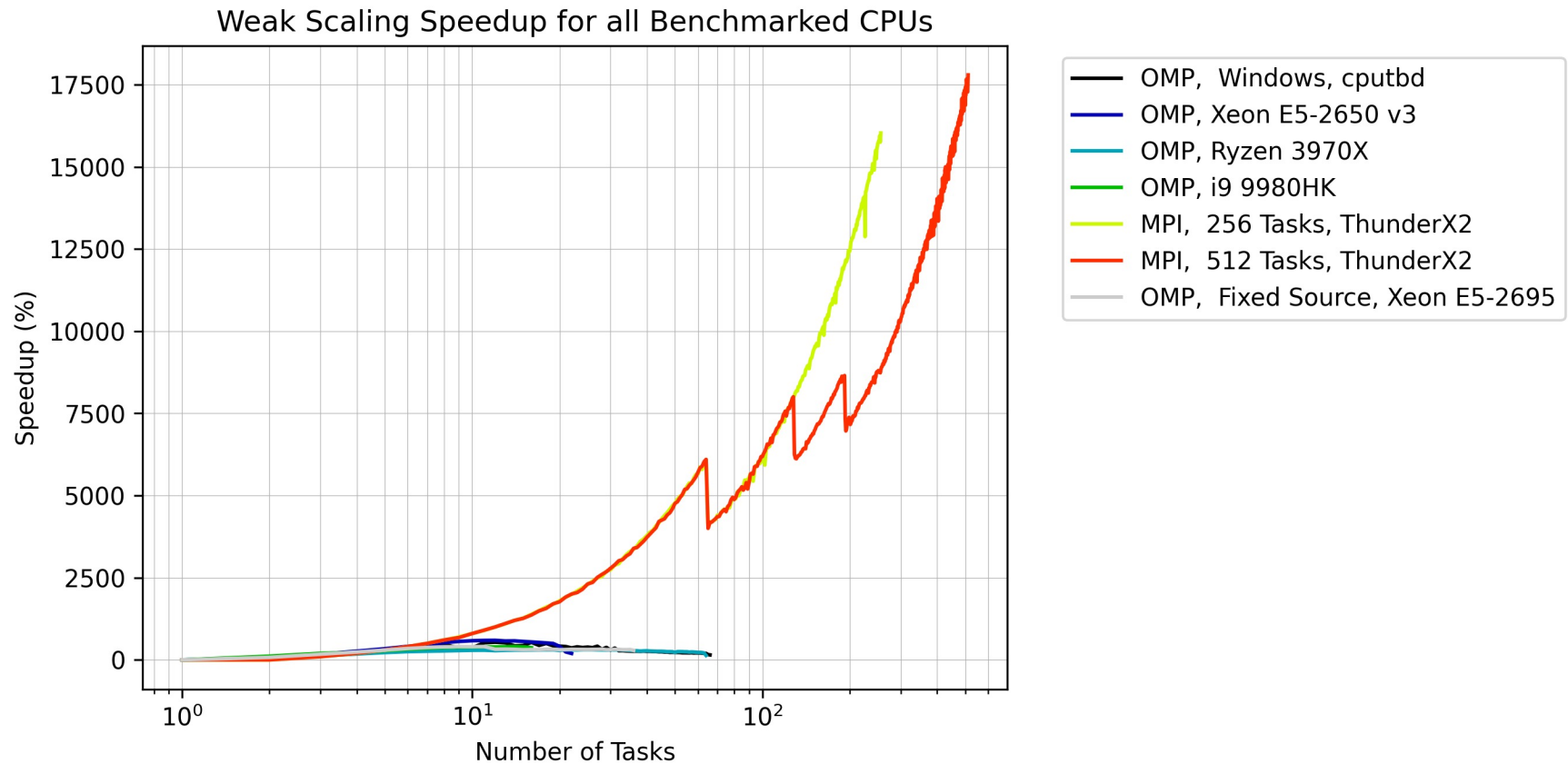
Strong Scaling Performance Falloff



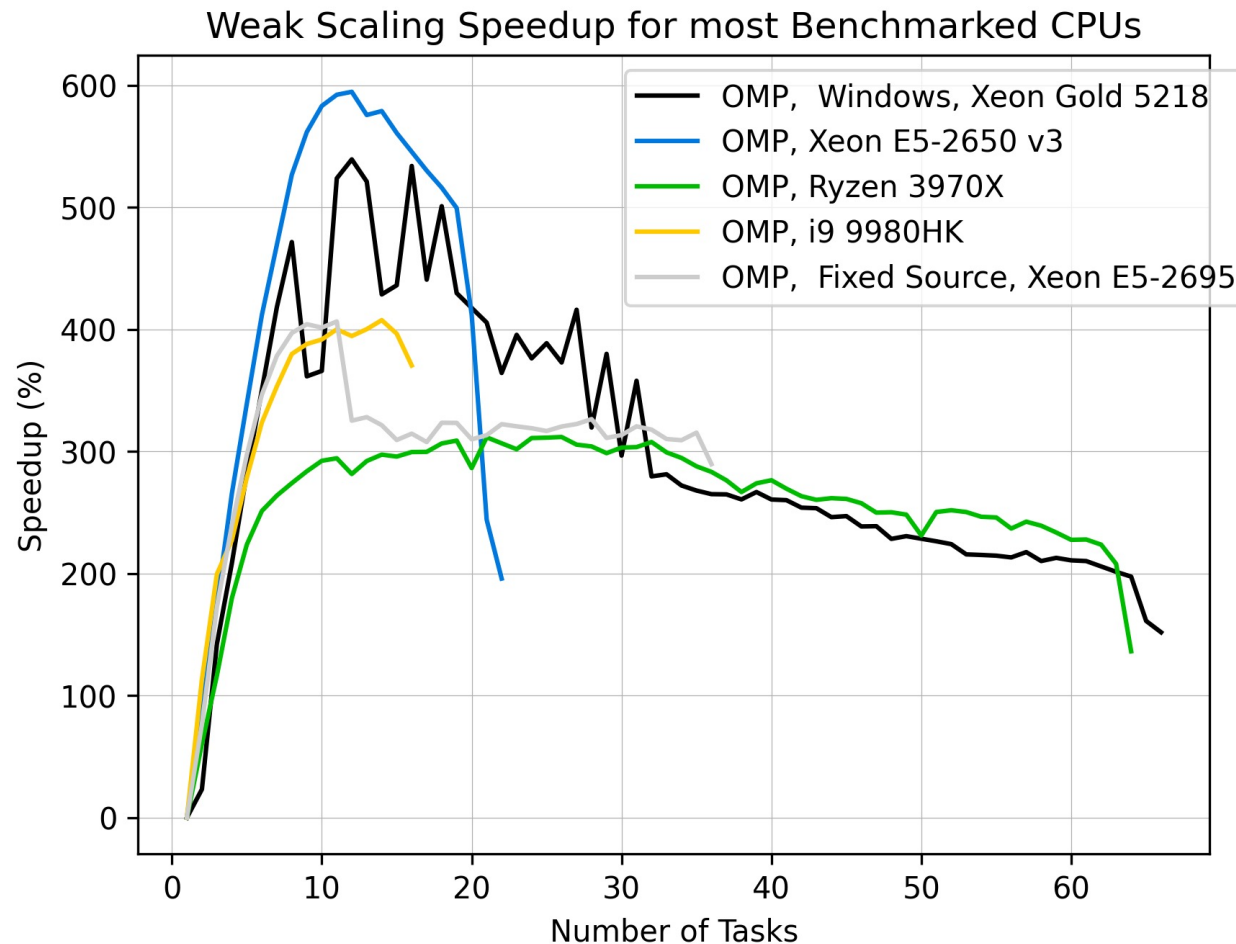
Weak Scaling Data At-A-Glance: Grind Time



Weak Scaling Data At-A-Glance: Speedup

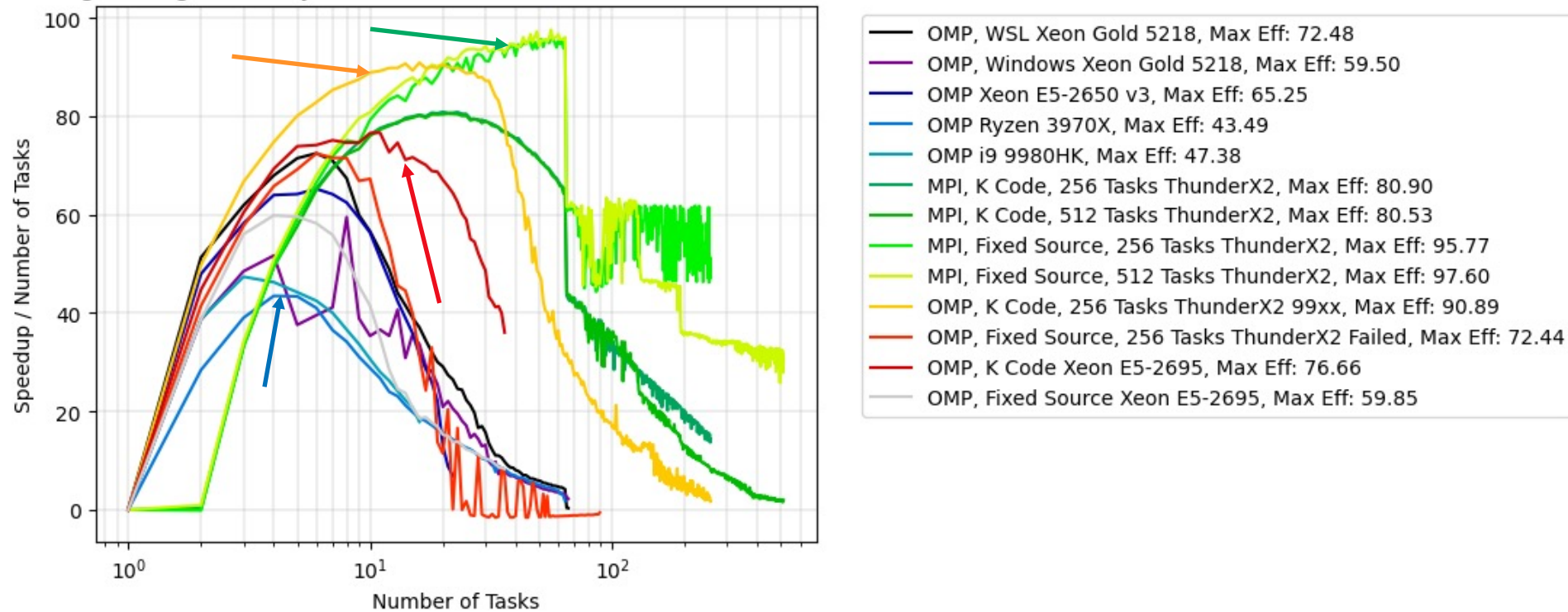


Weak Scaling Data At-A-Glance: Speedup (minus the outliers)



Metric – Strong Scaling

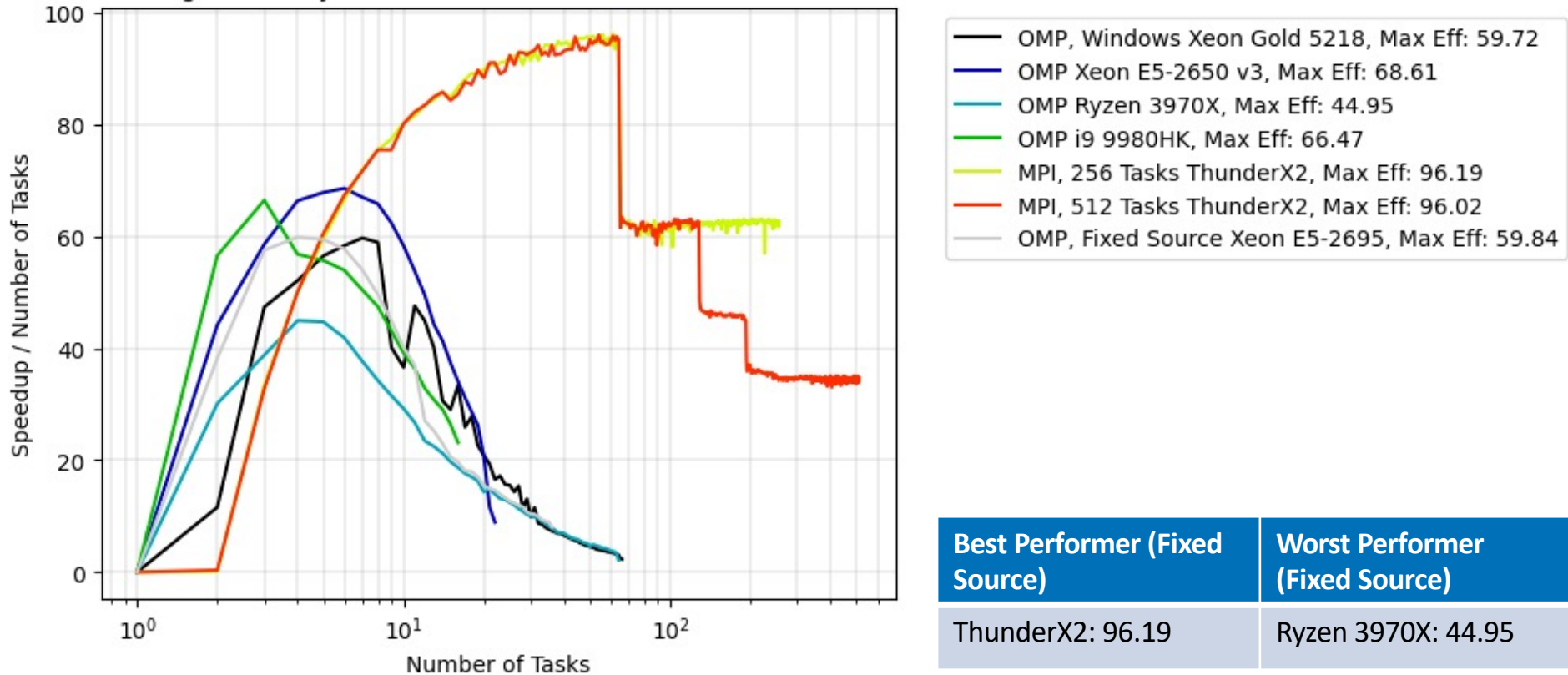
Strong Scaling Efficiency vs Number of Tasks for each Benchmarked CPU



Best Performer (Fixed Source)	Worst Performer (Fixed Source)	Best Performer (K Code)	Worst Performer (K Code)
ThunderX2: 97.60	Ryzen 3970X: 43.49	ThunderX2: 90.89	Xeon E5-2695: 76.66

Metric – Weak Scaling

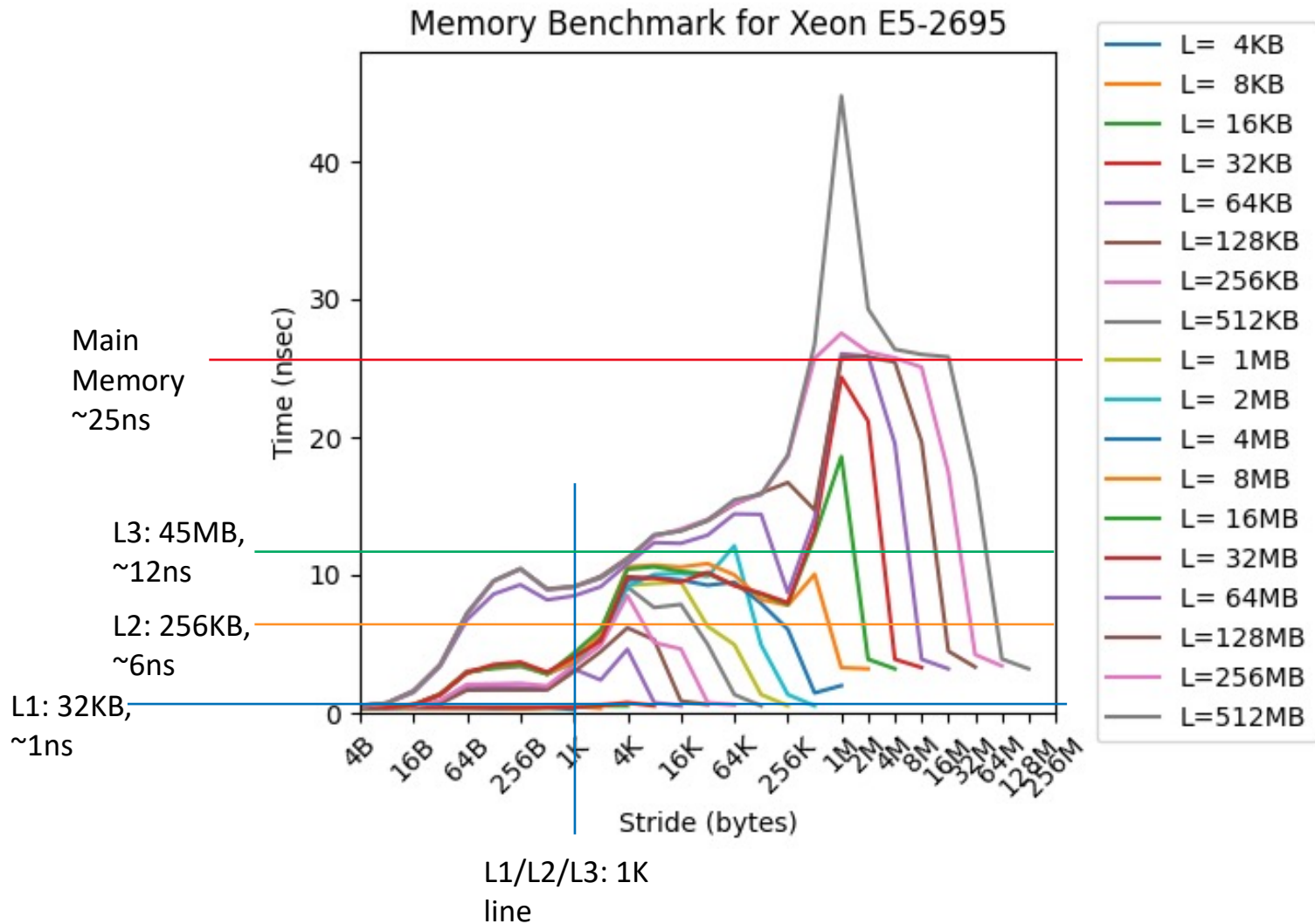
Weak Scaling Efficiency vs Number of Tasks for each Benchmarked CPU



Membench: Memory benchmarking

- Membench is a simple memory benchmark available at <https://github.com/bkochuna/membench>
 - Created by a grad student in the late 90s, "maintained" by Prof. Brendan Kochunas at UMich.
- Used during University of Michigan NERS 570 course (scientific computing).
- Source lightly modified to set maximum write to 0.5 GB.
- Program measures cache and memory access times and memory structure can be deduced from resulting graphs.

Membench: Example Result



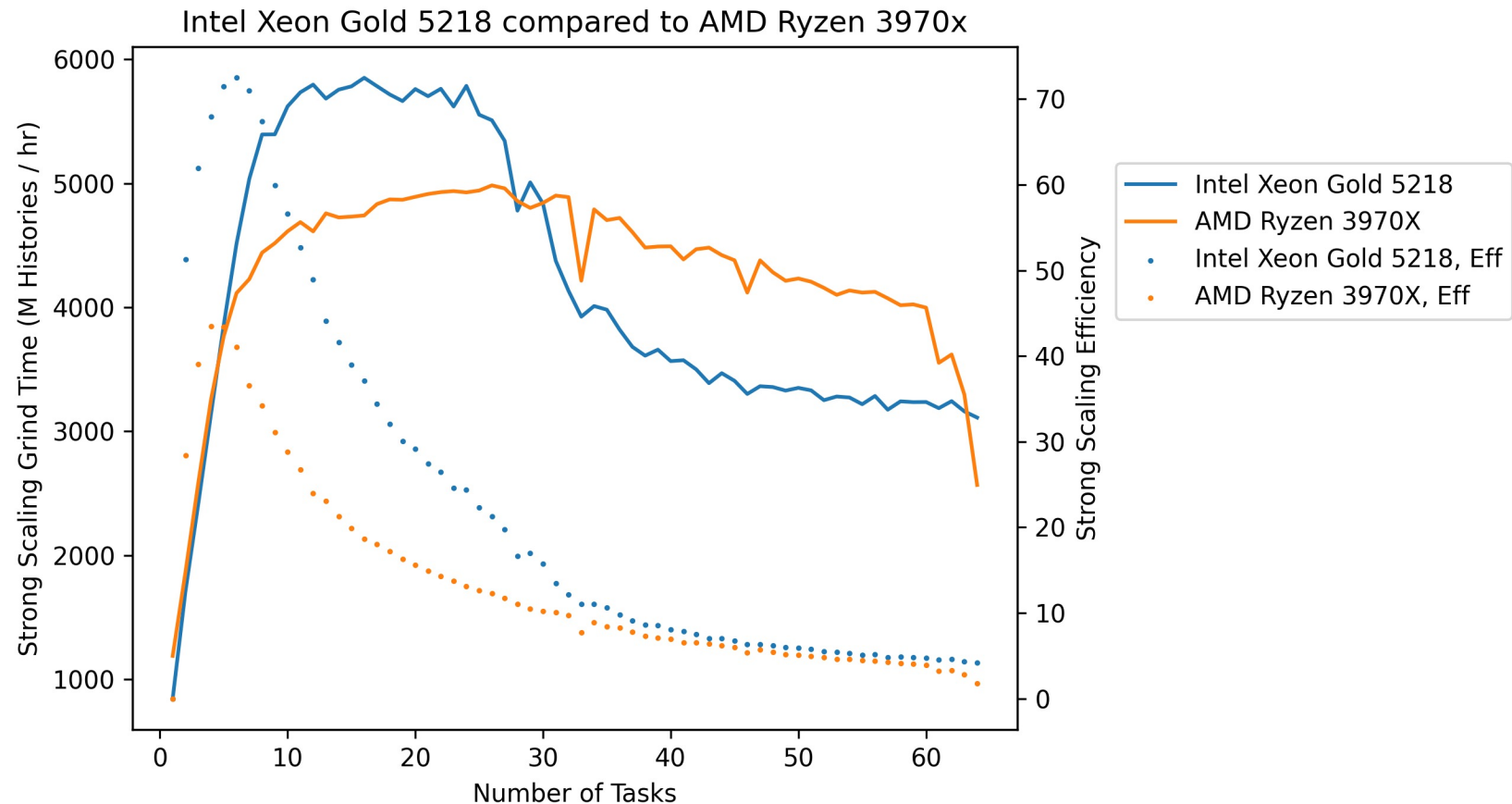
Membench Results

Chip	L1 Size, Line length, Access time	L2 Size, Line length, Access Time	L3 Size, Line length, Access Time	Main Memory Access Time
Intel Xeon E5-2695v4	32K, 1K, ~1ns	256K, 1K, ~6ns	45M, 1K, ~12ns	~25ns
Intel i9-9980HK	512K, 1K, ~5ns	2M, 1K, ~8ns	16M, 1K, ~11ns	~20ns
Intel Xeon E5-2650v3	32K, 1K, ~2ns	2M, 1k, ~10ns	25M, 1K, ~15ns	~24ns
Intel Xeon Gold 5218	32K, 64B, ~2ns	1M, 256B, ~4ns	10M, 1K, ~8ns	~20ns
AMD Ryzen 3970X	2M, 1K, ~2ns	16M, 4K, ~8ns	128M, 1K, ~13ns	~18ns
Cavium ThunderX2	32K, 32B, ~2ns	256K, 64B, ~3ns	32M, 4K, ~15ns	~27ns

Xeon Gold 5218 and Ryzen 3970X are both 32c/64t chips at 2.3 and 3.7 GHz base clock respectively.. Let's compare performance.

Xeon Gold 5218 vs Ryzen 3970X

clock speed vs cache speed



Key Takeaways

- Most off the shelf chips have similar performance (Xeon, AMD Threadripper were comparable)
- Maximum performance (and efficiency) appears related to cache performance, speed is most important.
- Scaling is better on “fatter” processors (Xeon vs Threadripper), but may be irrelevant when maximum performance is considered
- MPI performance is better than OMP threading, if you have MCNP source, compile with MPI.
 - OMP good within socket or prior to hyperthreading, MPI better across sockets