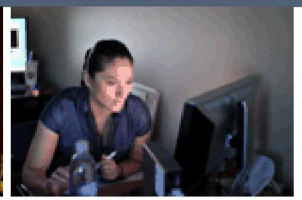
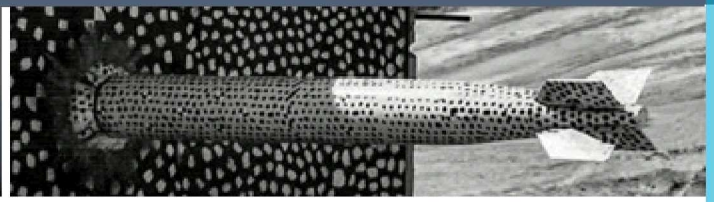
SAND2019-1754PE

CTS-1 Microbenchmarks





PRESENTED BY

Douglas M. Pase, PhD/Sandia National Laboratories

With Anthony M. Agelastos and Joel O. Stevenson

SAND2019-1294 PE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

All data are from gathered from unclassified systems using commercially available, non-proprietary hardware.

All benchmarks are simple variants of common benchmarks and/or public libraries (DAXPY, DGEMM, STREAM, MPI_Sendrecv, etc.)

DAXPY

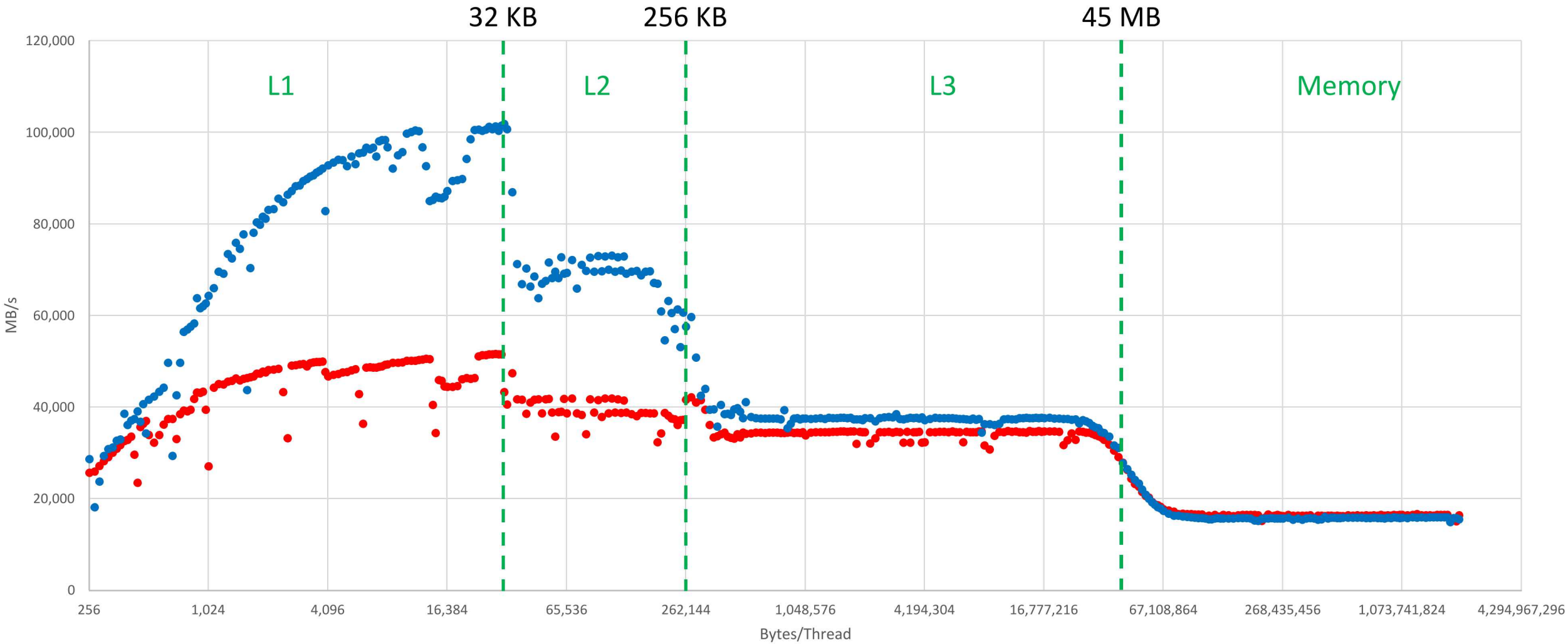
`y[i] = α*x[i] + y[i]`

x, **y** are double-precision floating-point vectors

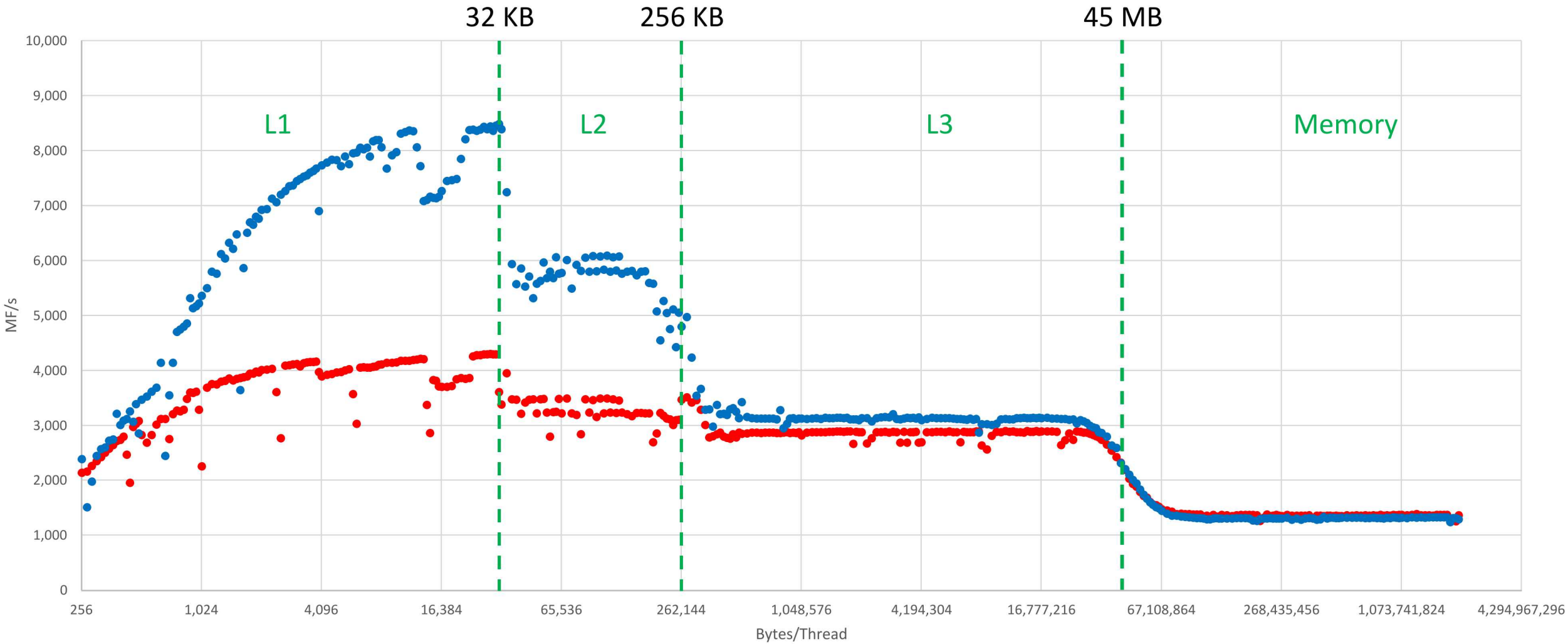
α is a double-precision floating-point scalar

i is an integer index; * is the scalar multiplication operator

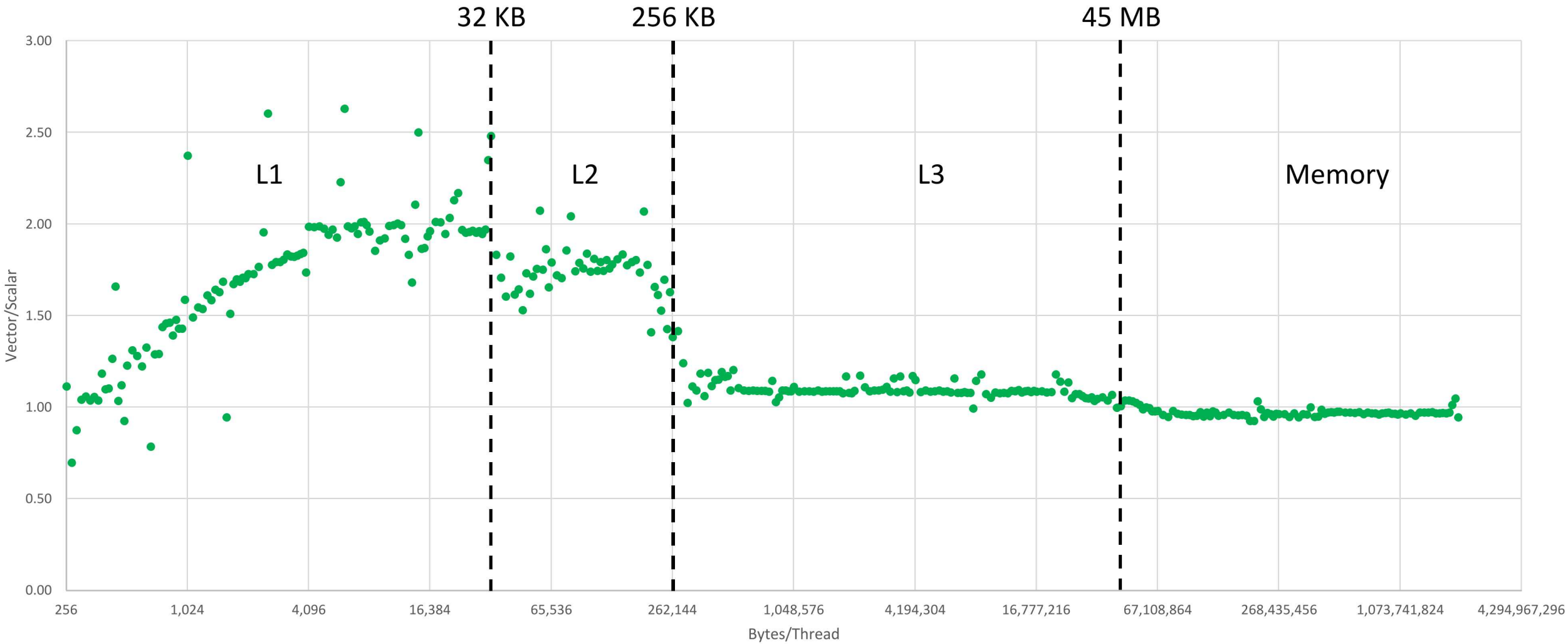
DAXPY Memory Throughput (One Core)



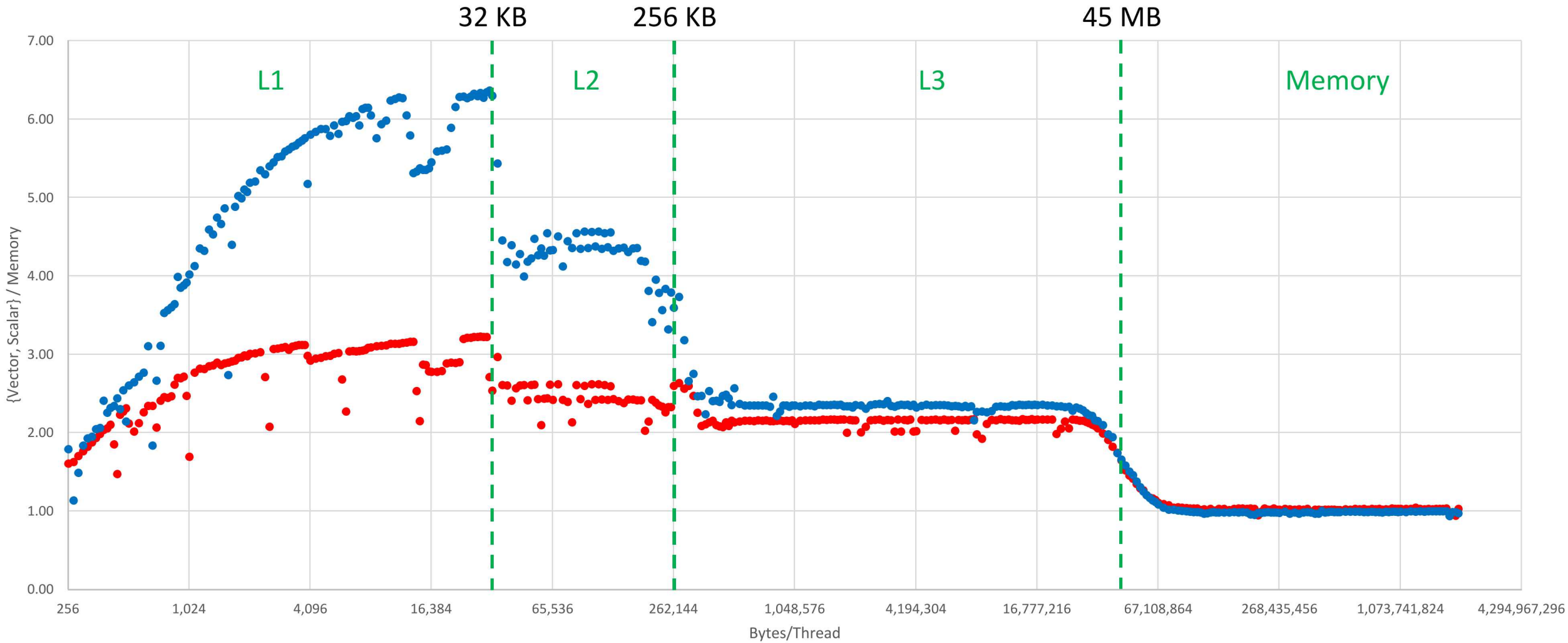
DAXPY Floating-Point Throughput (One Core)



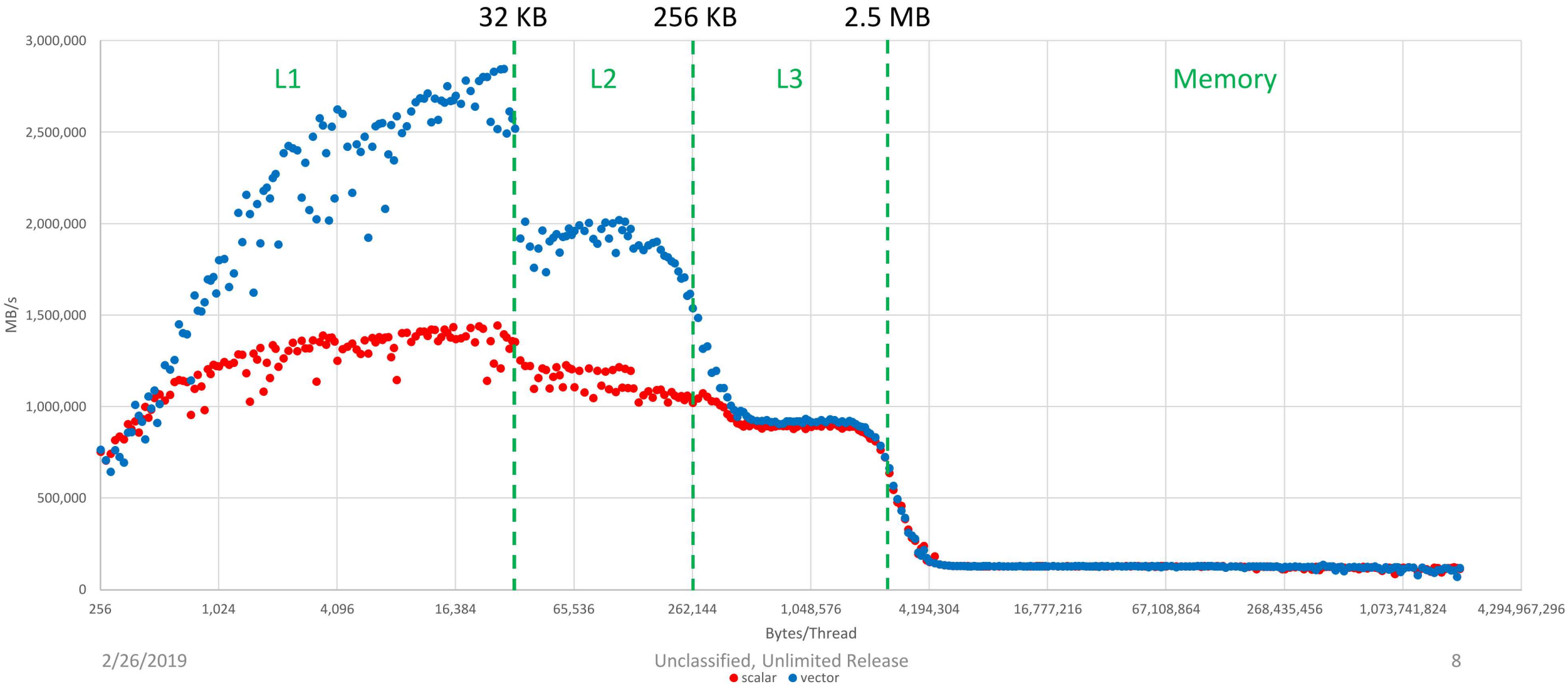
DAXPY Vector/Scalar (One Core)



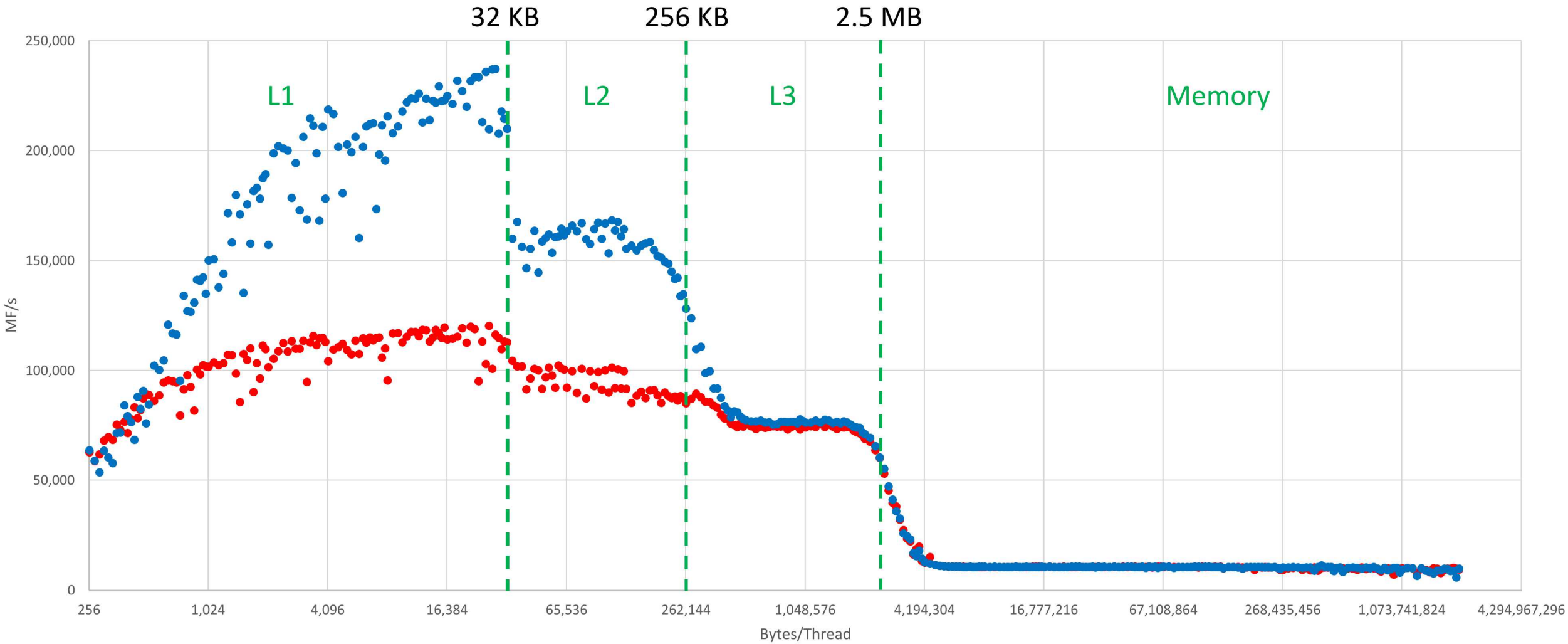
Performance Relative to Memory (One Core)



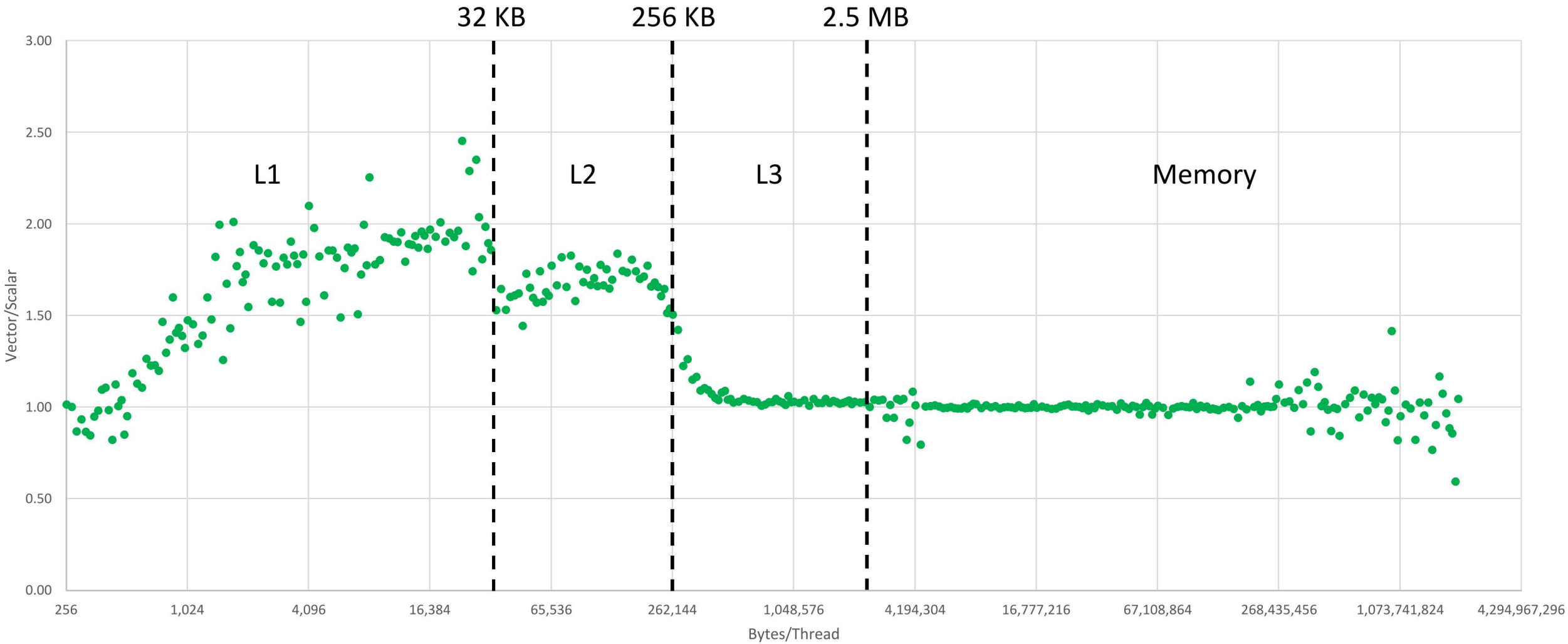
DAXPY Memory Throughput (36 Cores)



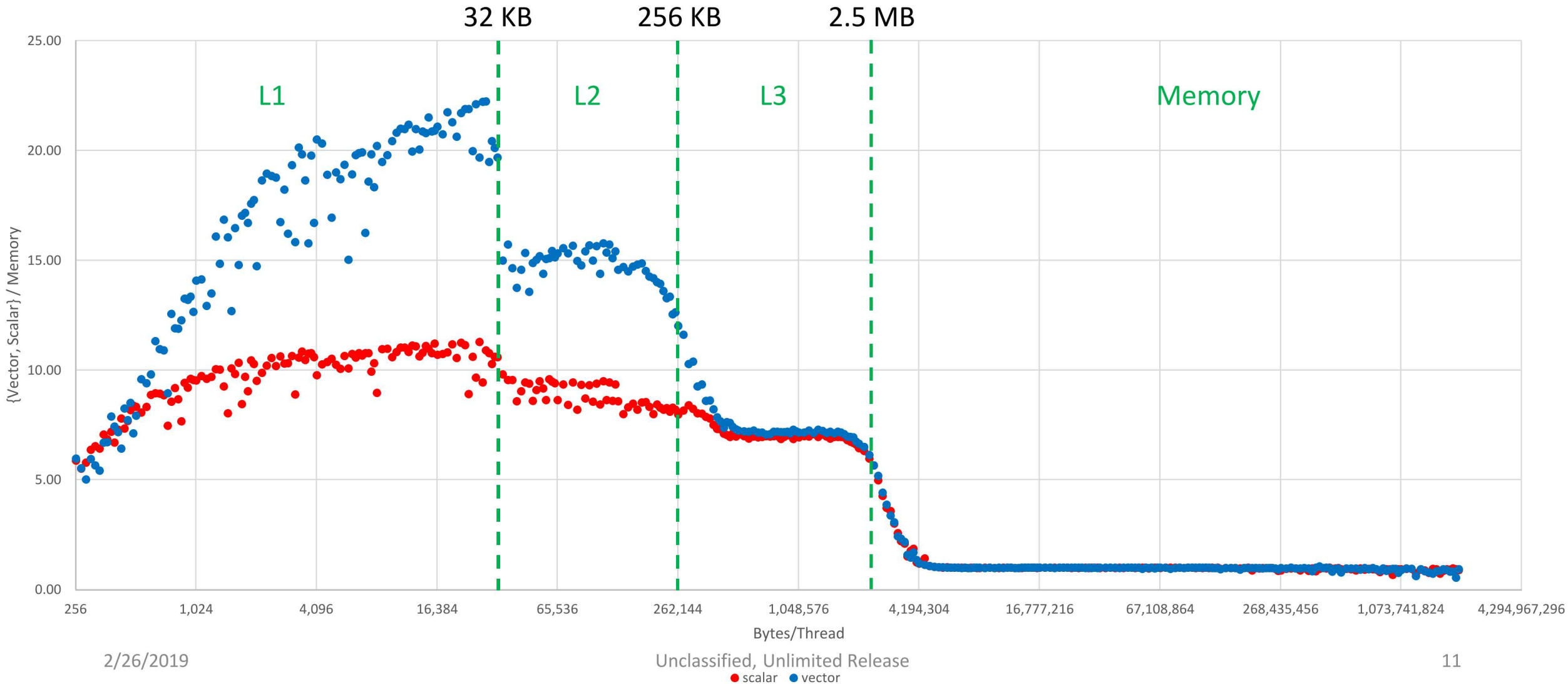
DAXPY Floating-Point Throughput (36 Cores)



DAXPY Vector/Scalar (36 Cores)



Performance Relative to Memory (36 Cores)



DGEMM

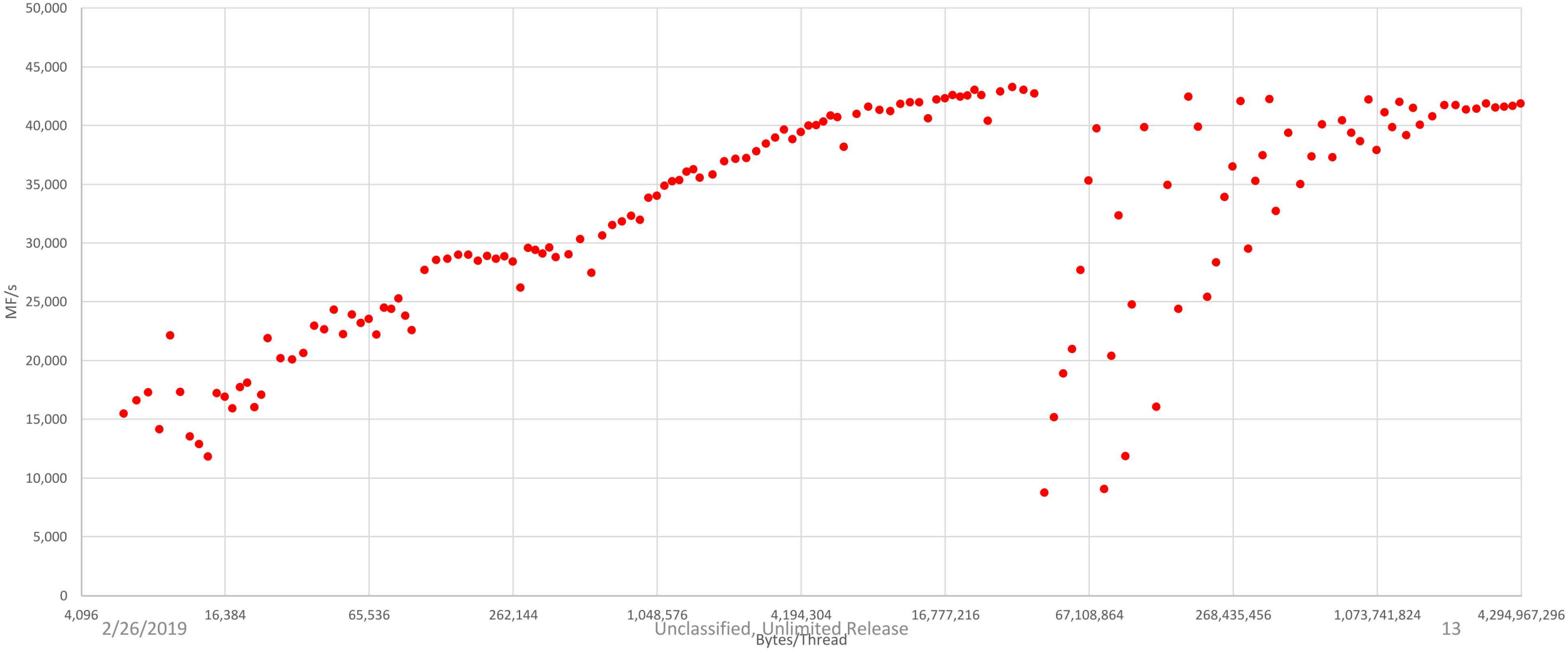
$\underline{C} = \alpha * (\underline{A} \times \underline{B}) + \beta * \underline{C}$

A, **B**, **C** are double-precision floating-point matrices

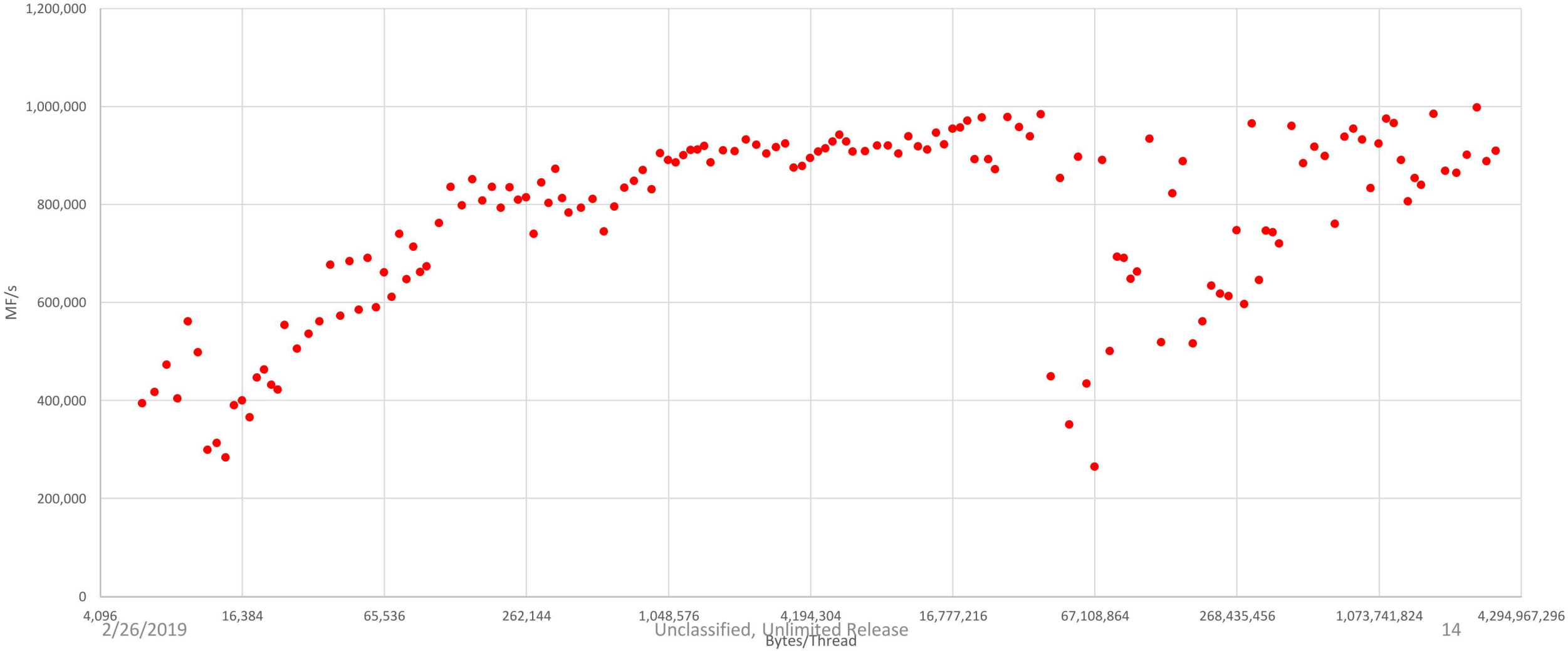
α, β are double-precision floating-point scalars

* is the scalar multiplication operator; x is the matrix multiplication operator

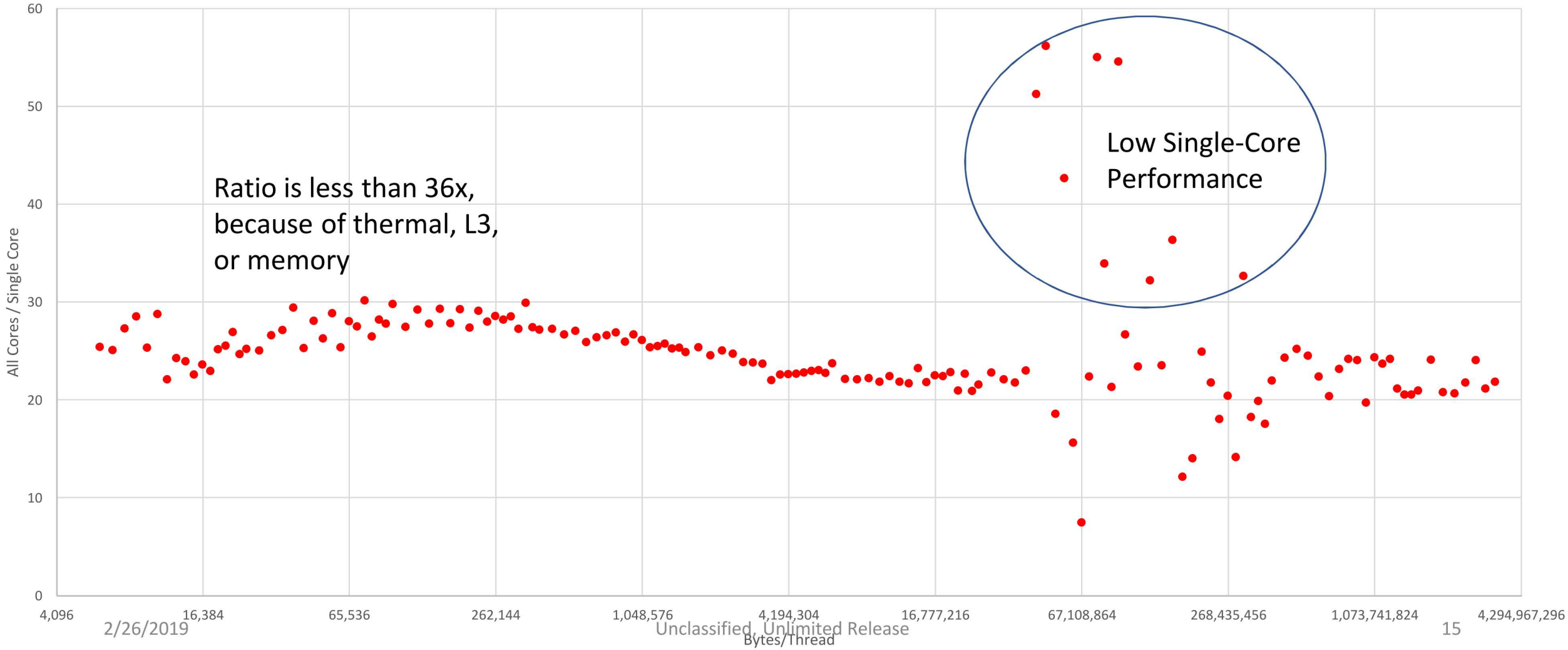
DGEMM MF/s (One Core)



DGEMM MF/s (36 Cores)



Ratio of 36 Cores / 1 Core



Stream

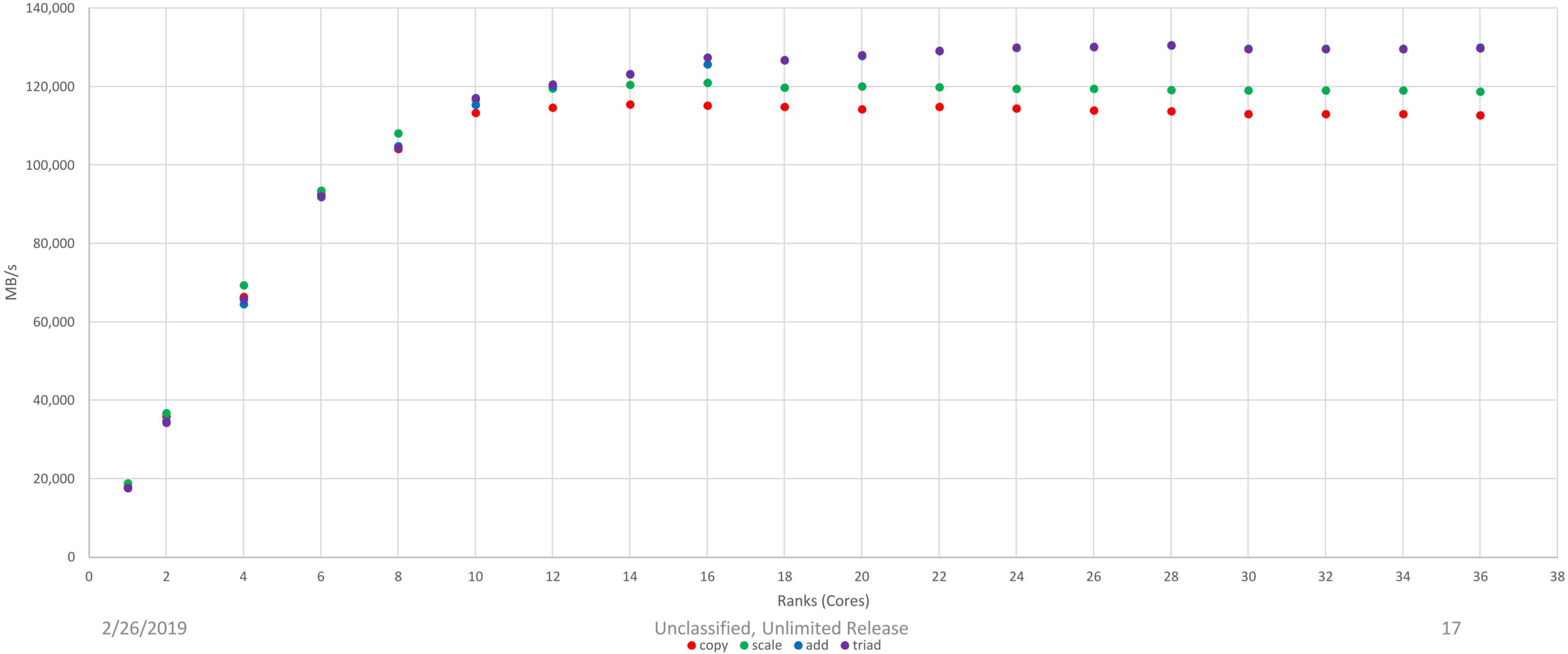
Copy: `y[i] = x[i]`

Scale: `y[i] = α*x[i]`

Add: `z[i] = x[i] + y[i]`

Triad: `z[i] = x[i] + α*y[i]`

Stream Memory Throughput



Gather

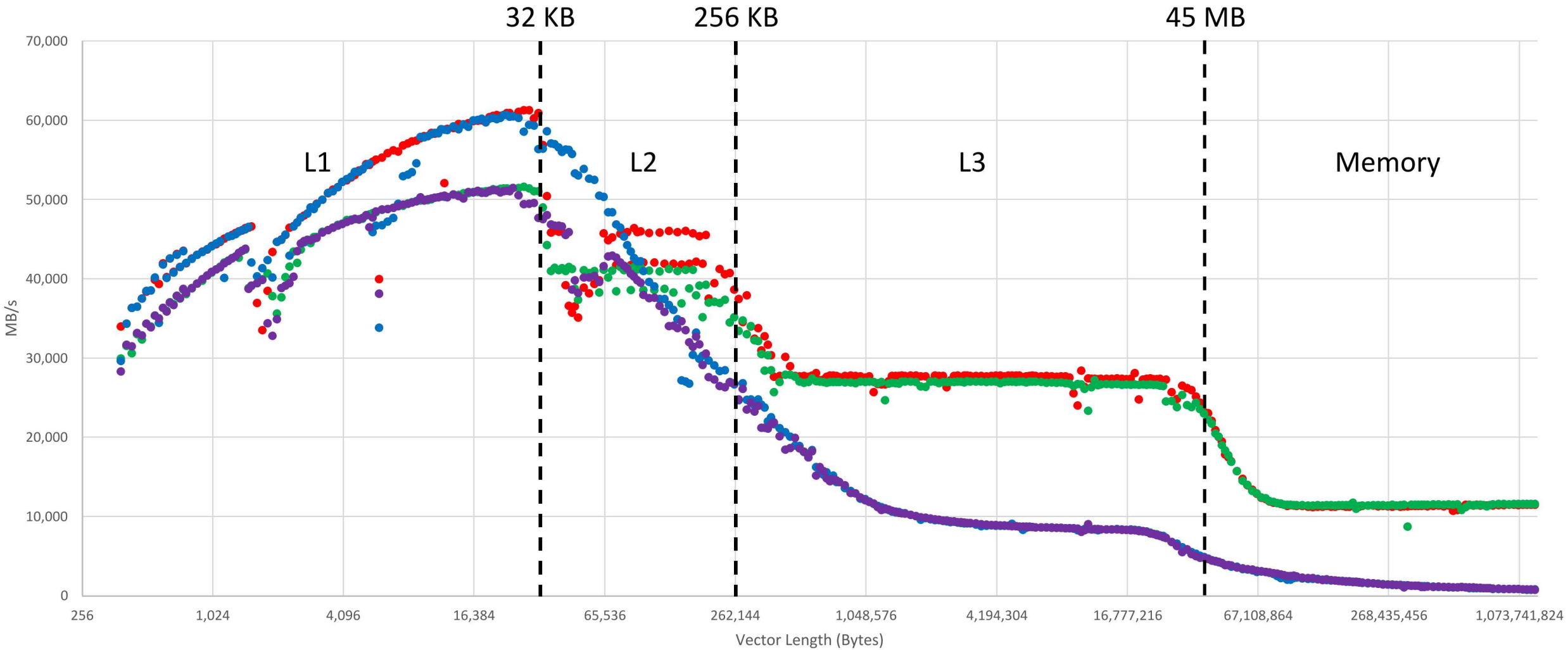
`y[i] = x[v[i]]`

`x`, `y` are double-precision floating-point vectors

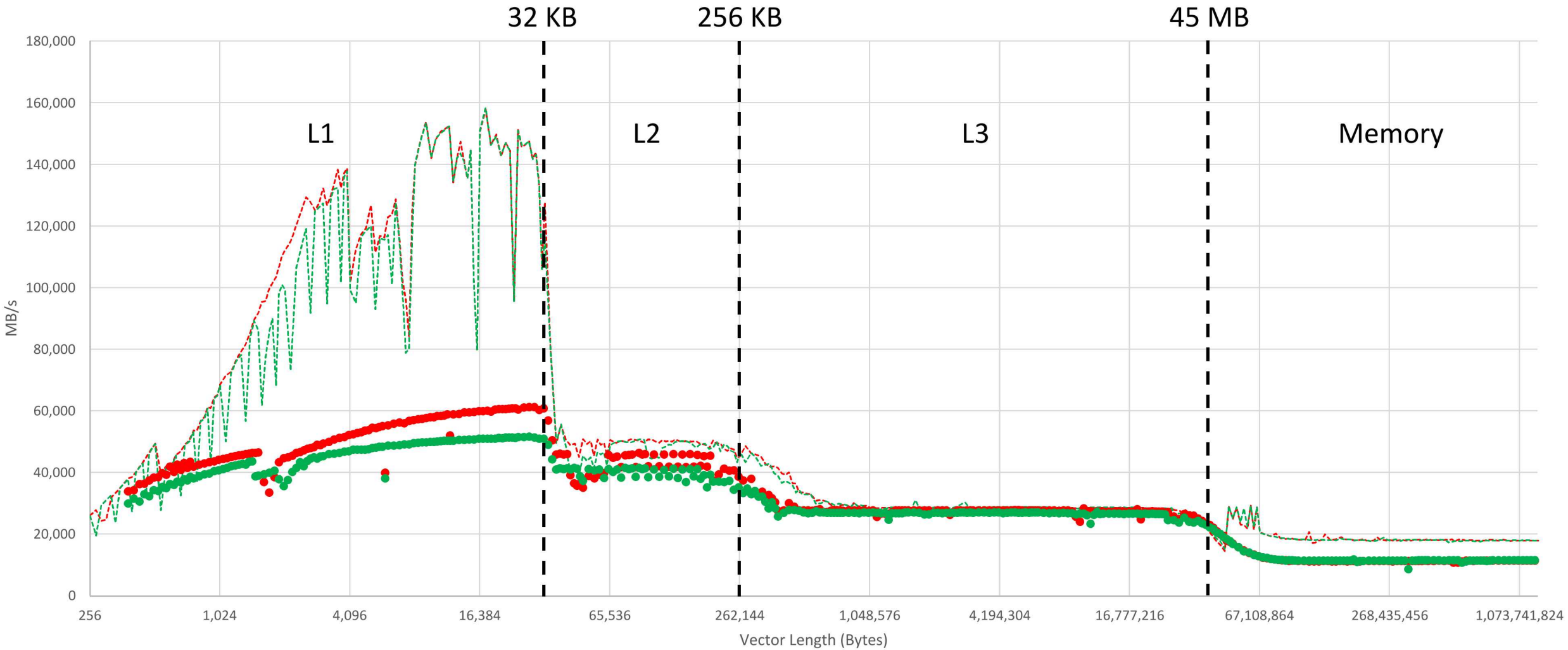
`v` is a 64-bit integer vector index into `x`

`i` is a 64-bit integer index

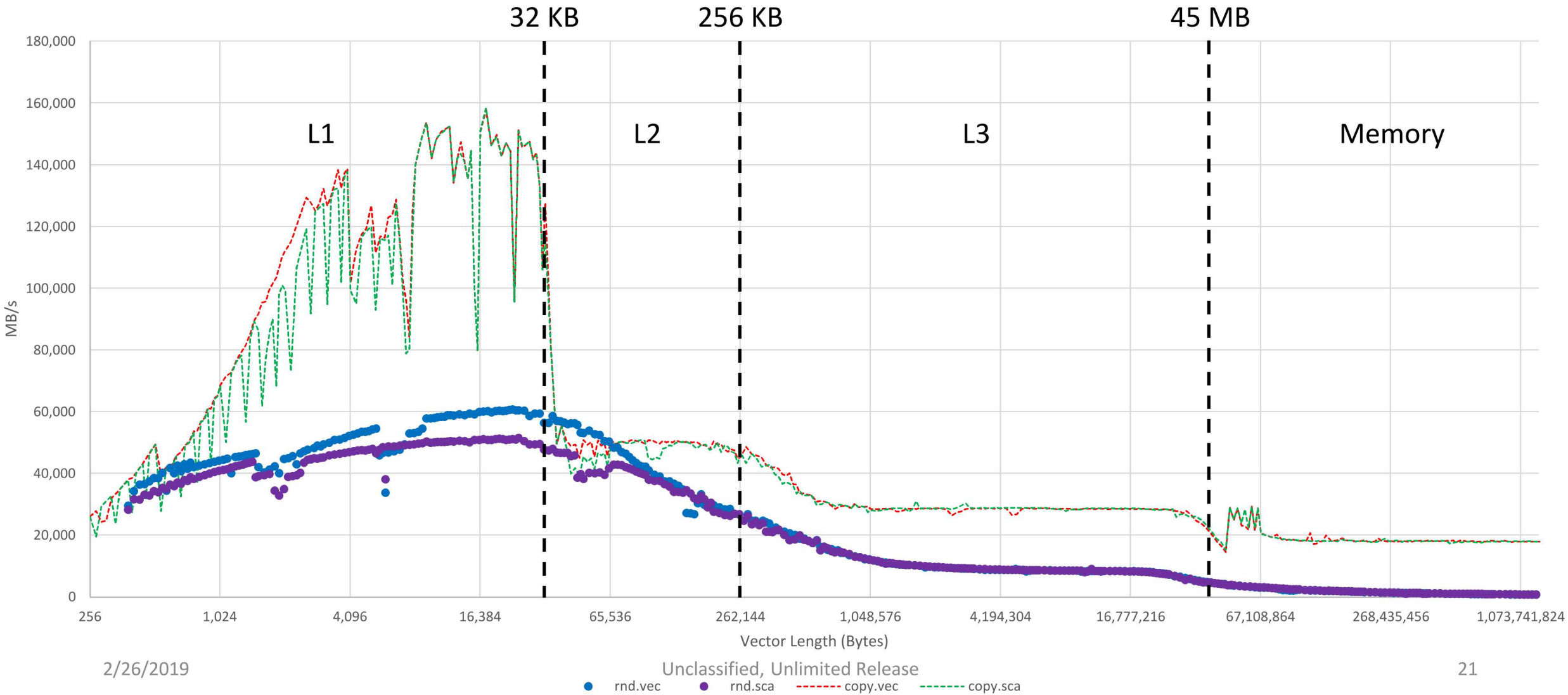
Gather MB/s (One Core)



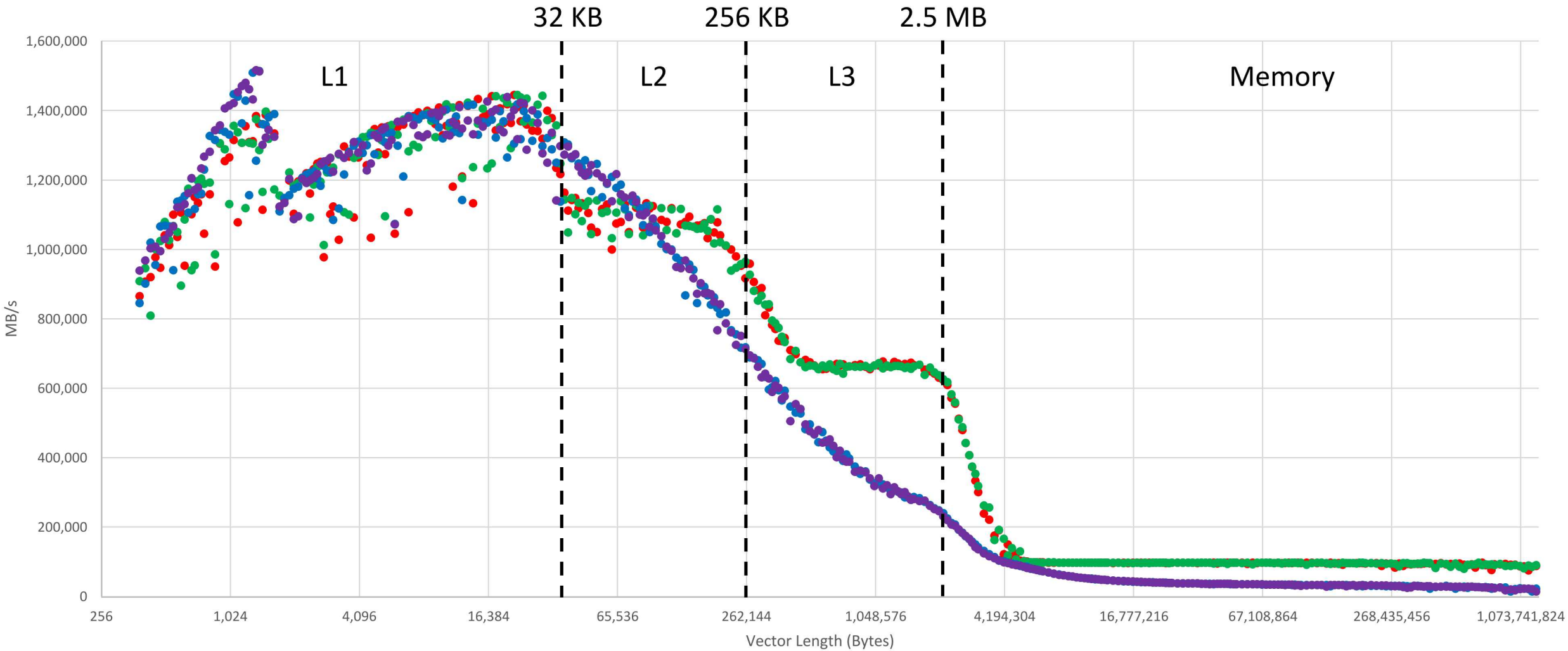
Sequential Gather MB/s (One Core)



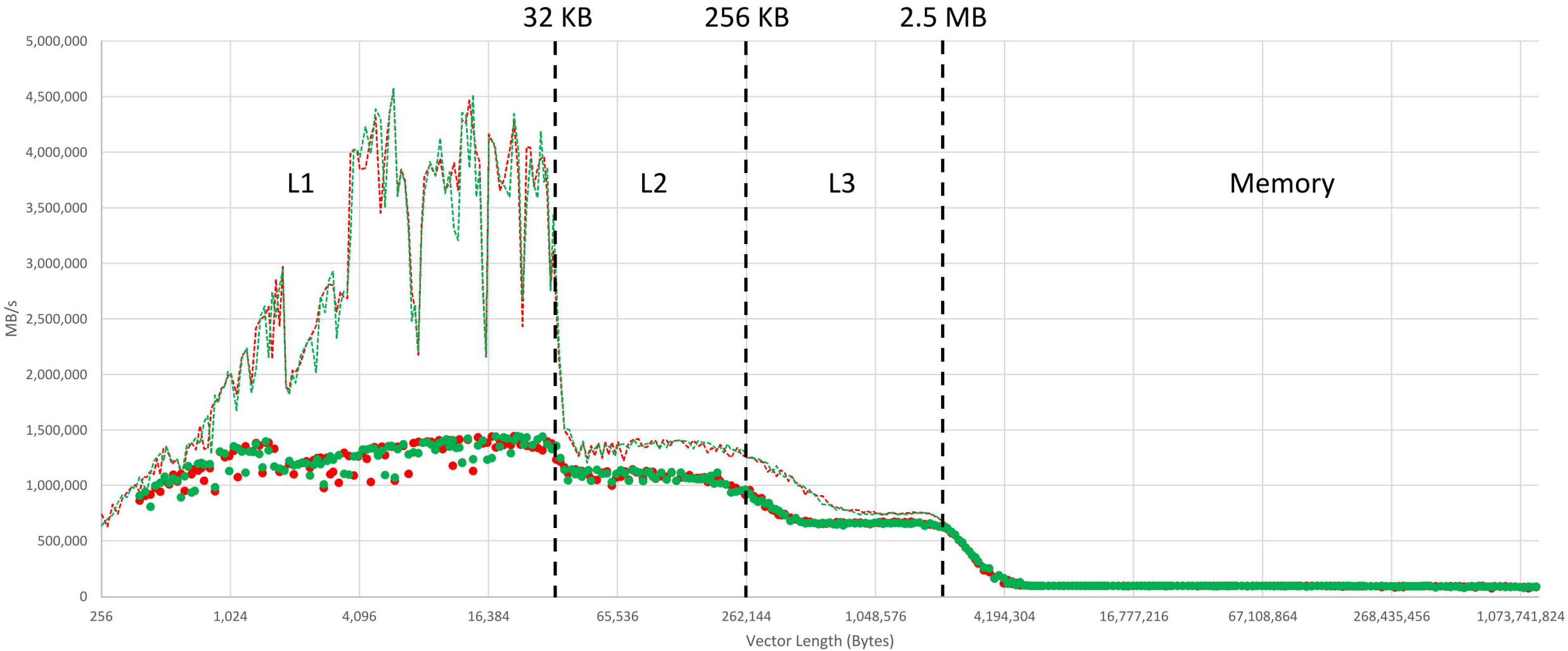
Random Gather MB/s (One Core)



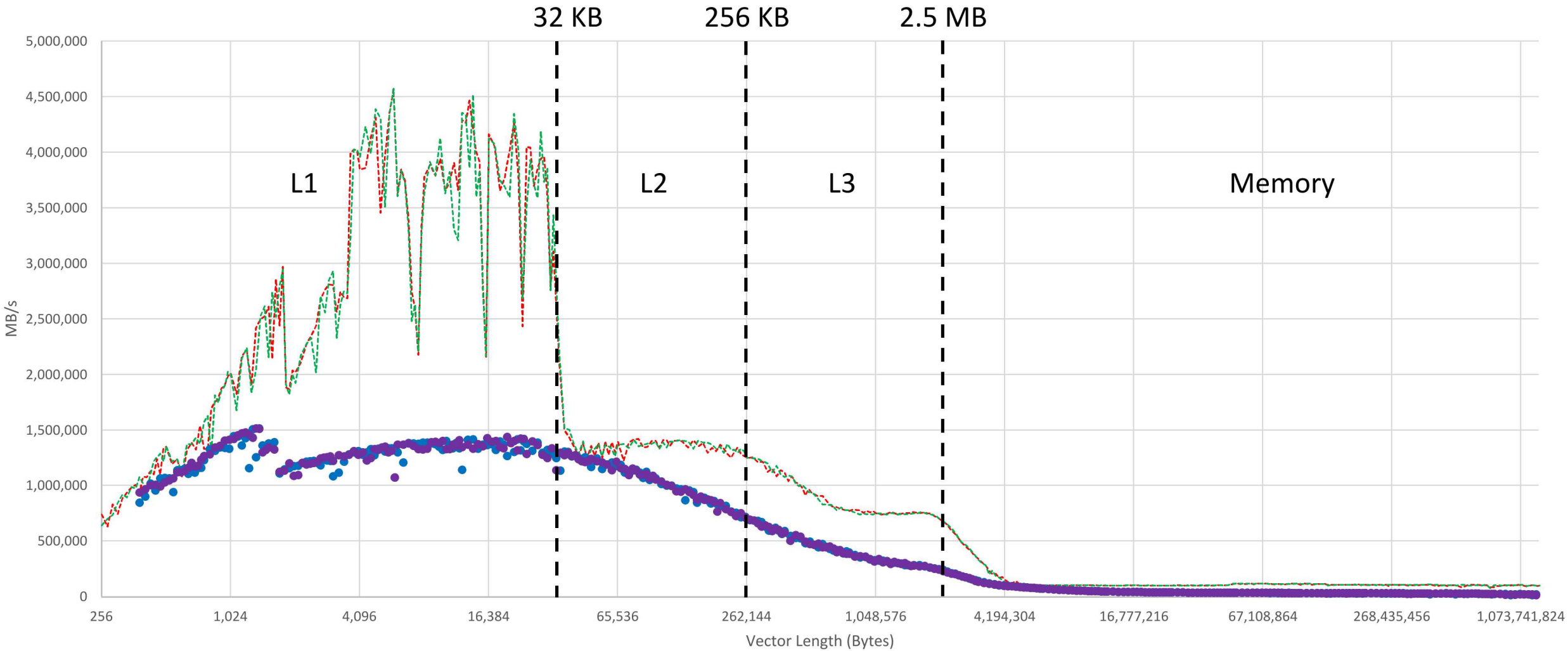
Gather MB/s (36 Cores)



Sequential Gather MB/s (36 Cores)



Random Gather MB/s (36 Cores)



Scatter

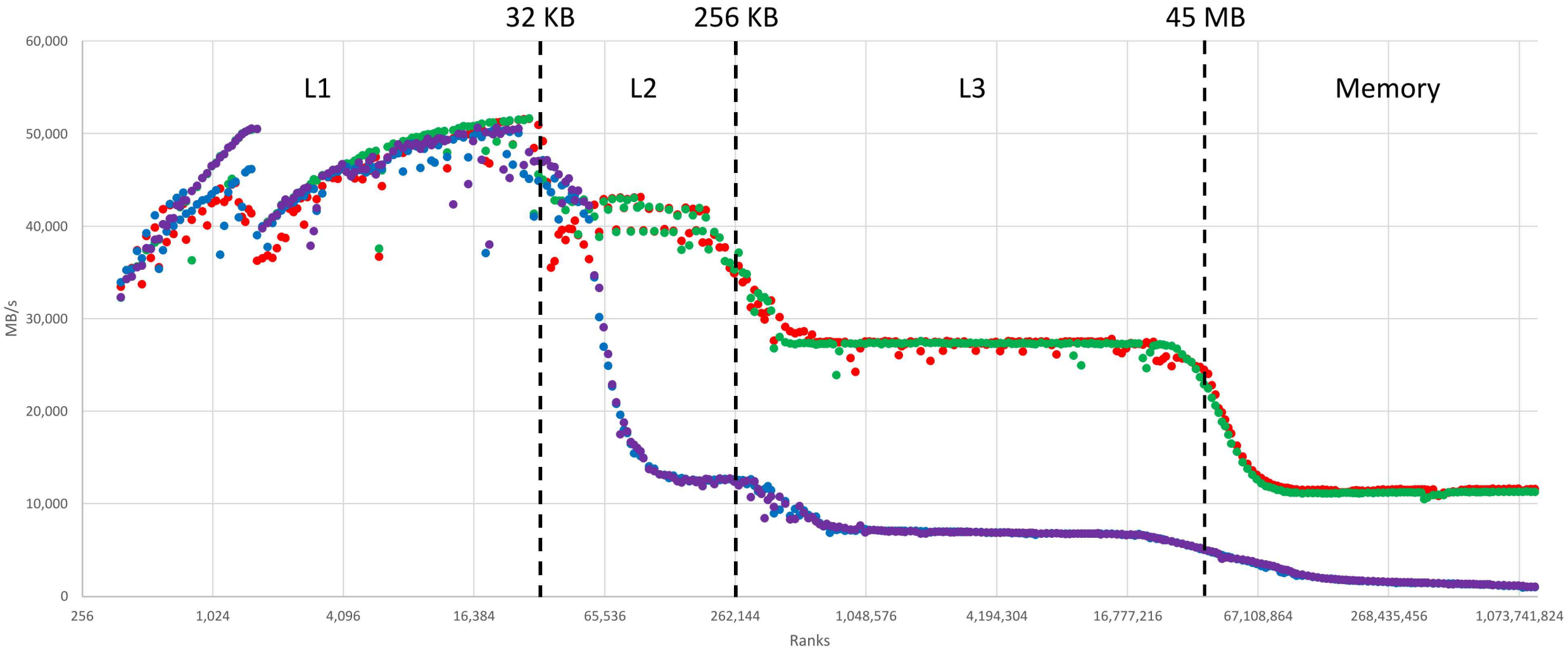
`y[v[i]] = x[i]`

`x`, `y` are double-precision floating-point vectors

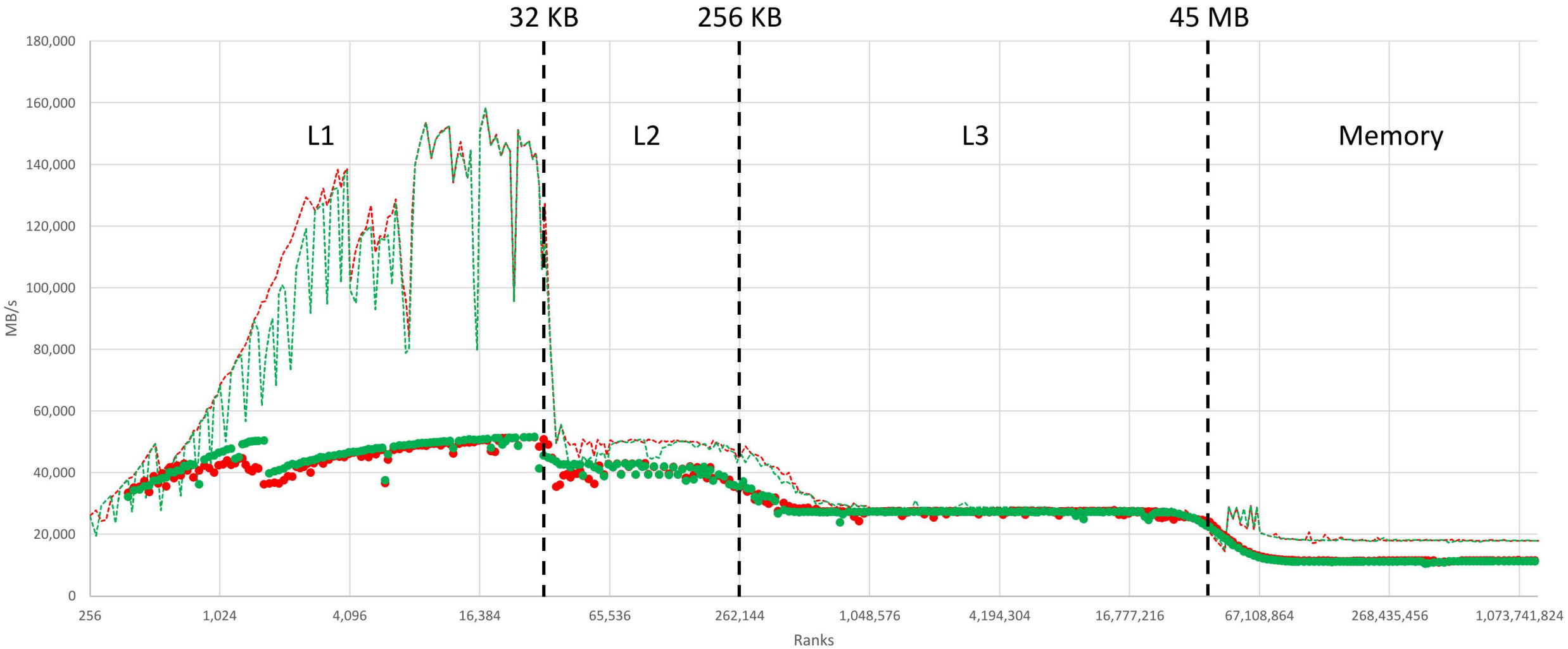
`v` is a 64-bit integer vector index into `x`

`i` is a 64-bit integer index

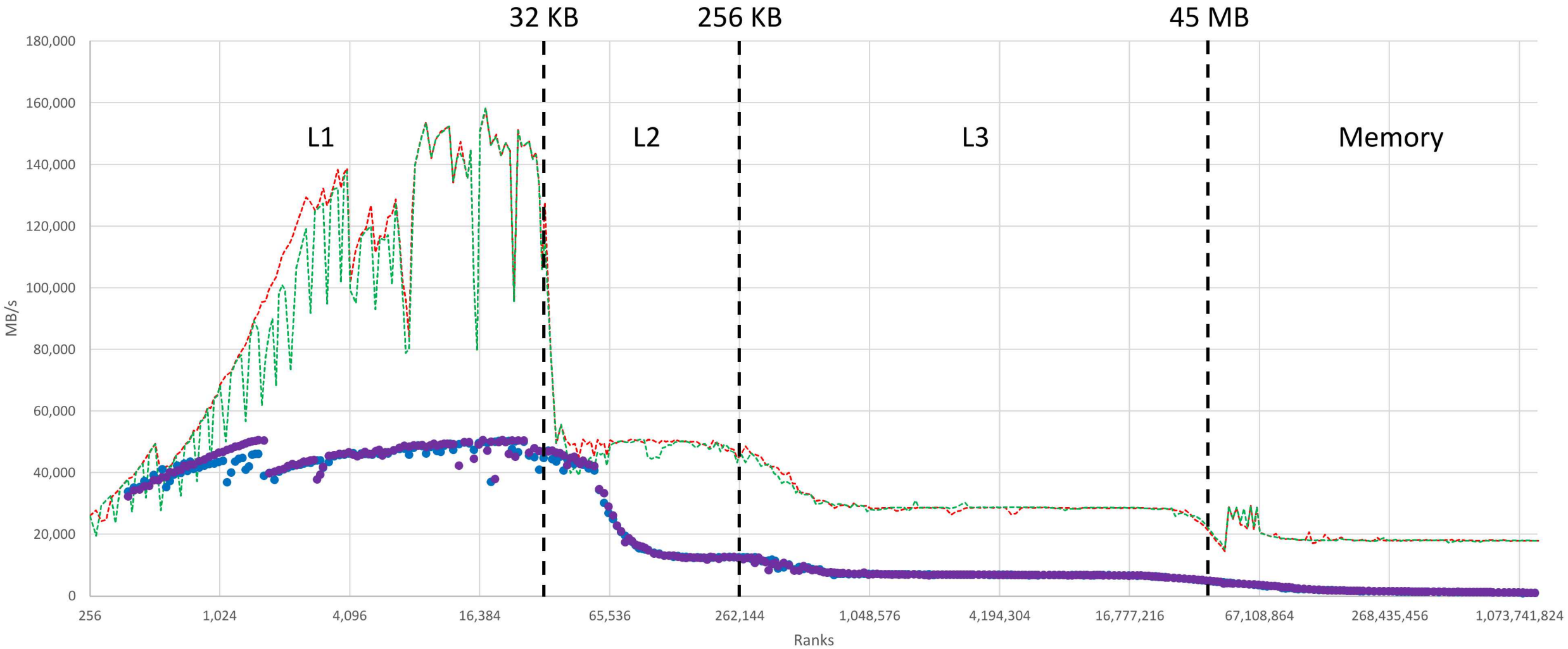
Scatter MB/s (One Core)



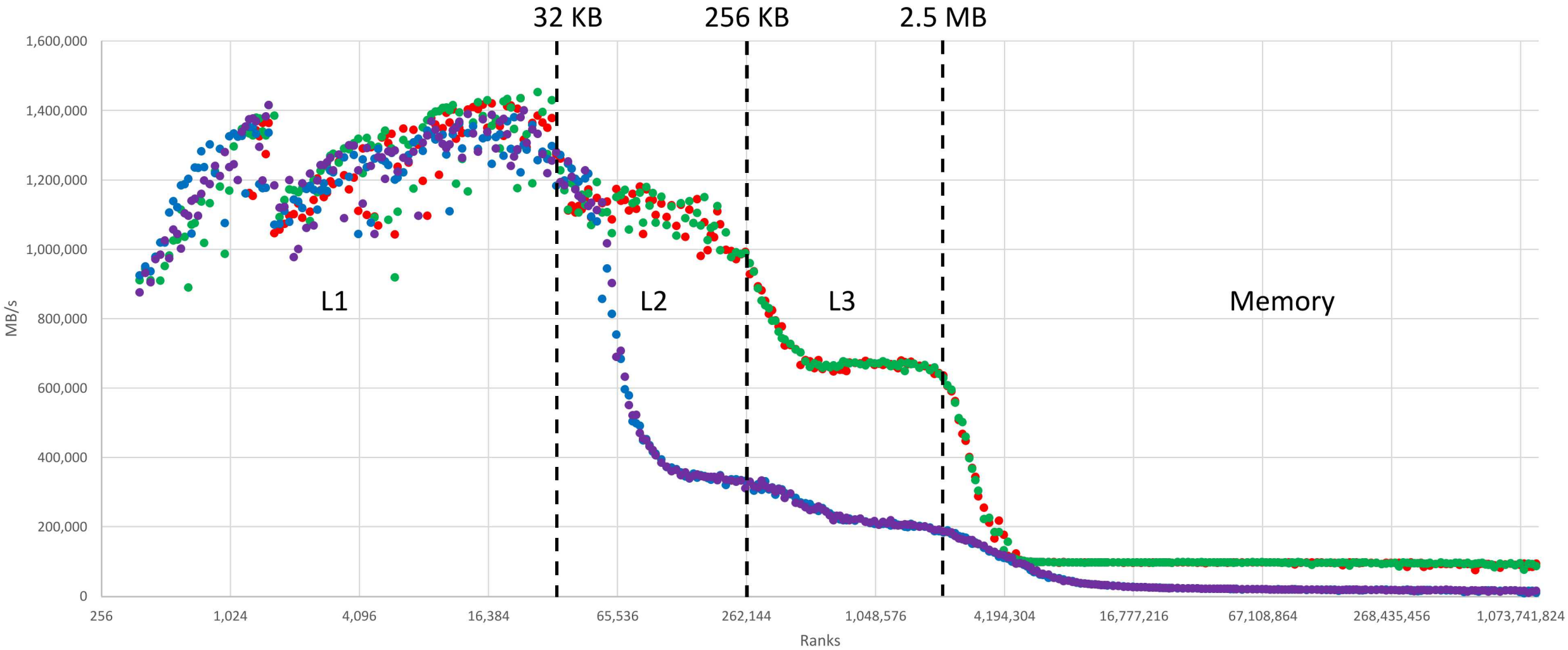
Sequential Scatter MB/s (One Core)



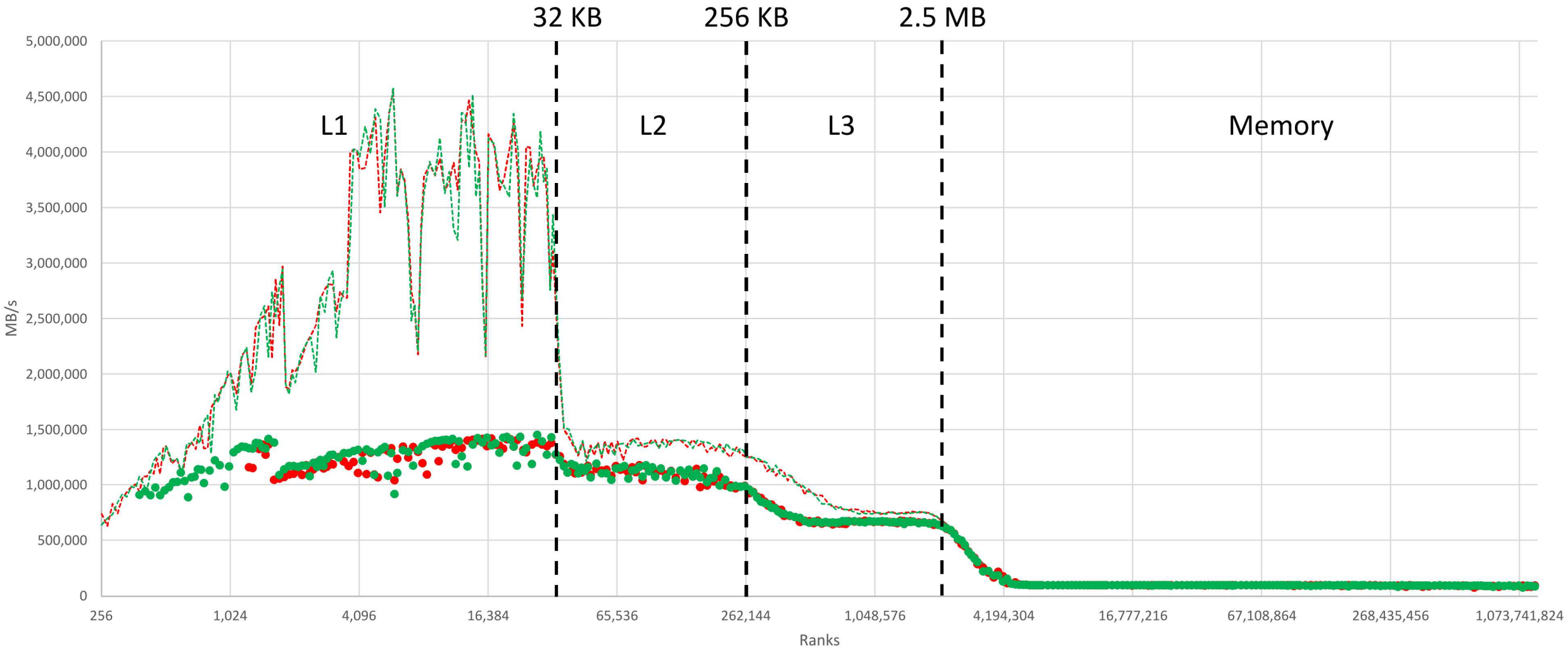
Random Scatter MB/s (One Core)



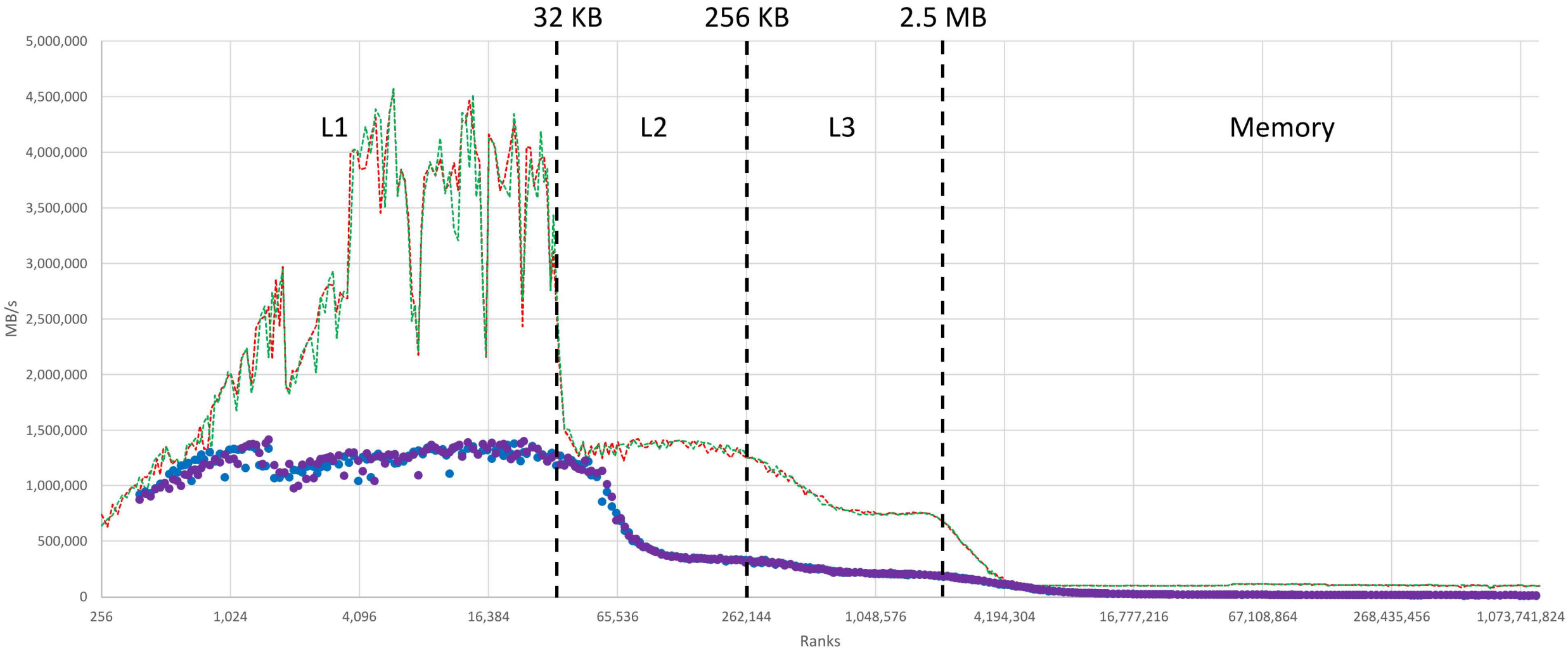
Scatter MB/s (36 Cores)



Sequential Scatter MB/s (36 Cores)

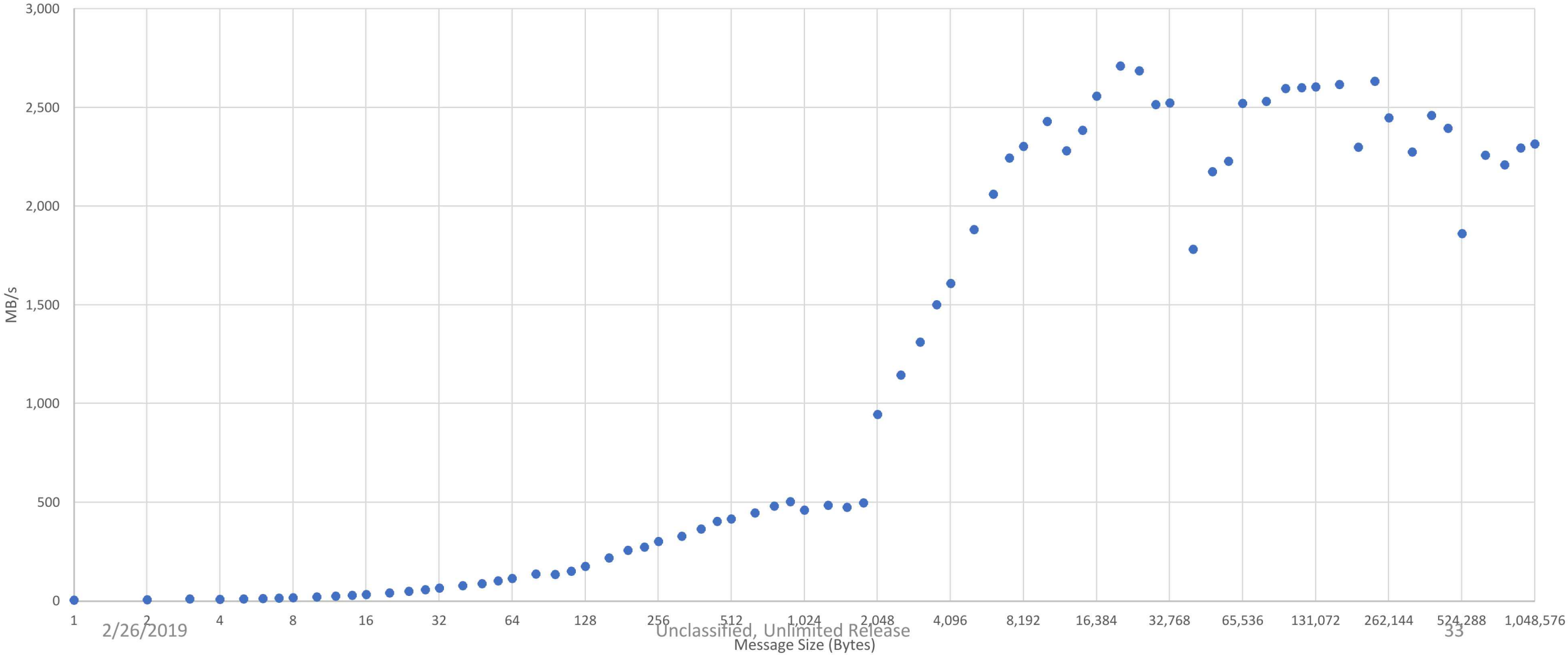


Random Scatter MB/s (36 Cores)

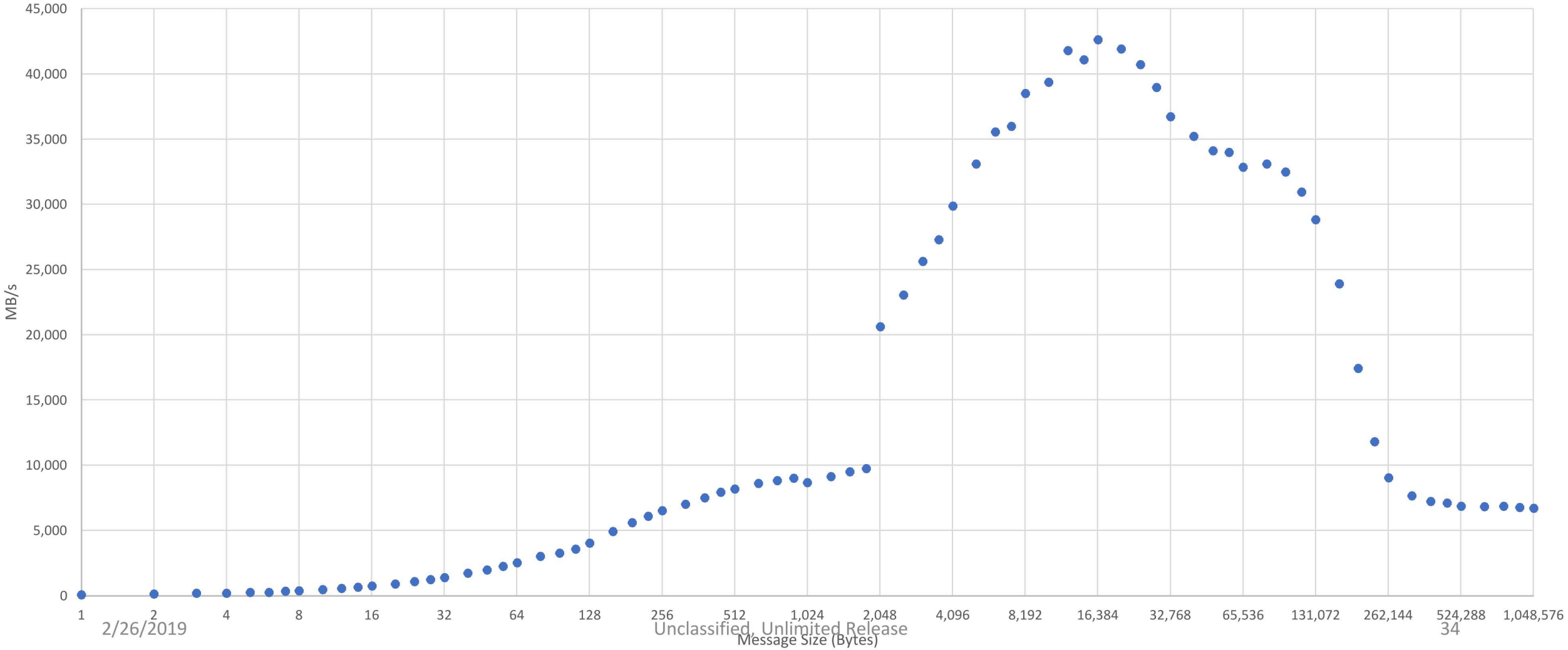


MPI_Sendrecv() & MPI_Allreduce()

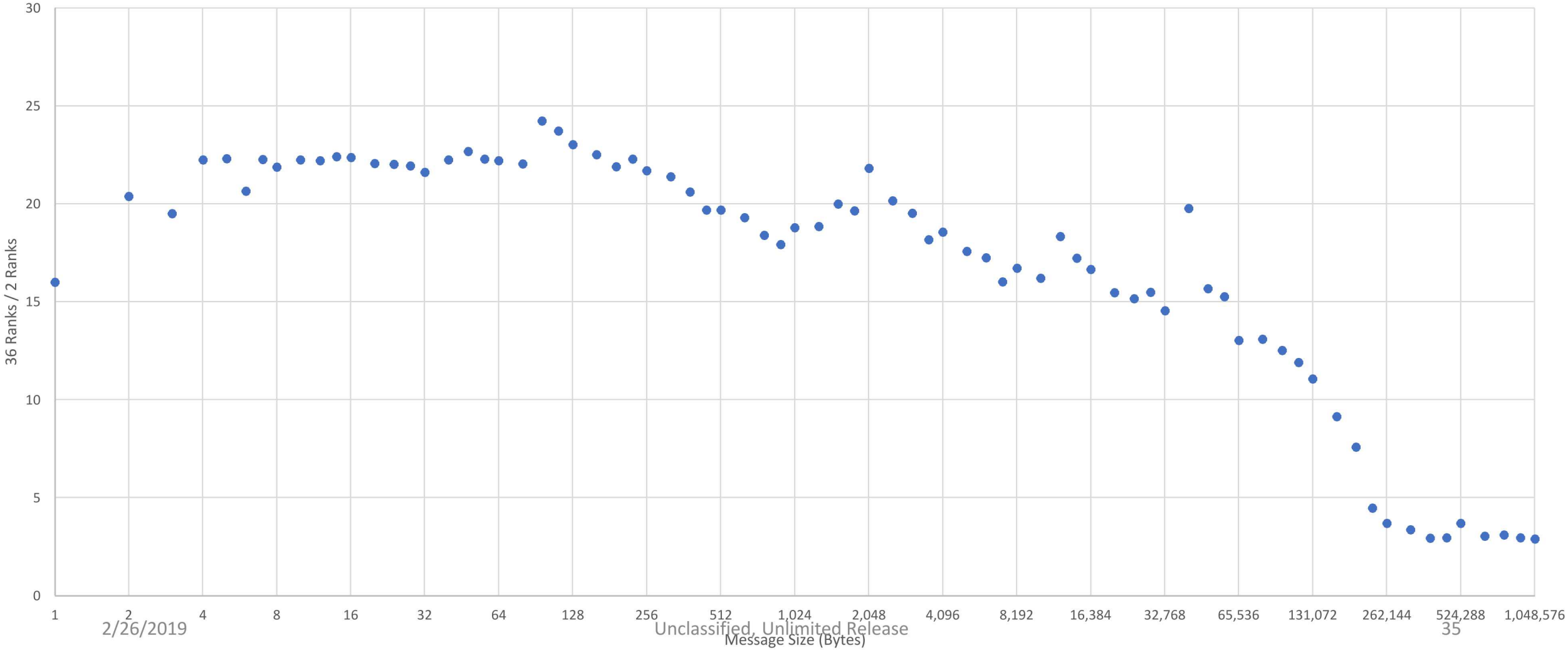
MPI_Sendrecv (1 Node, 2 Ranks)



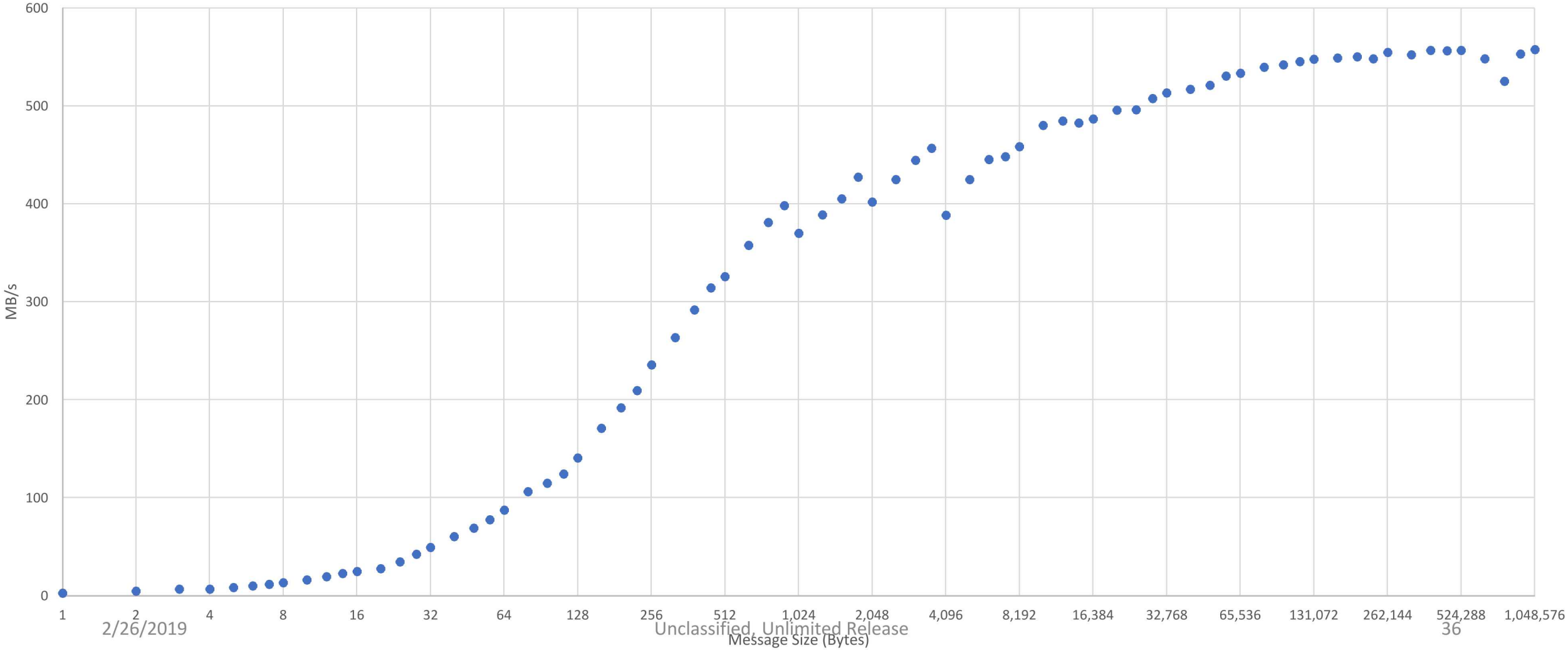
MPI_Sendrecv (1 Node, 36 Ranks)



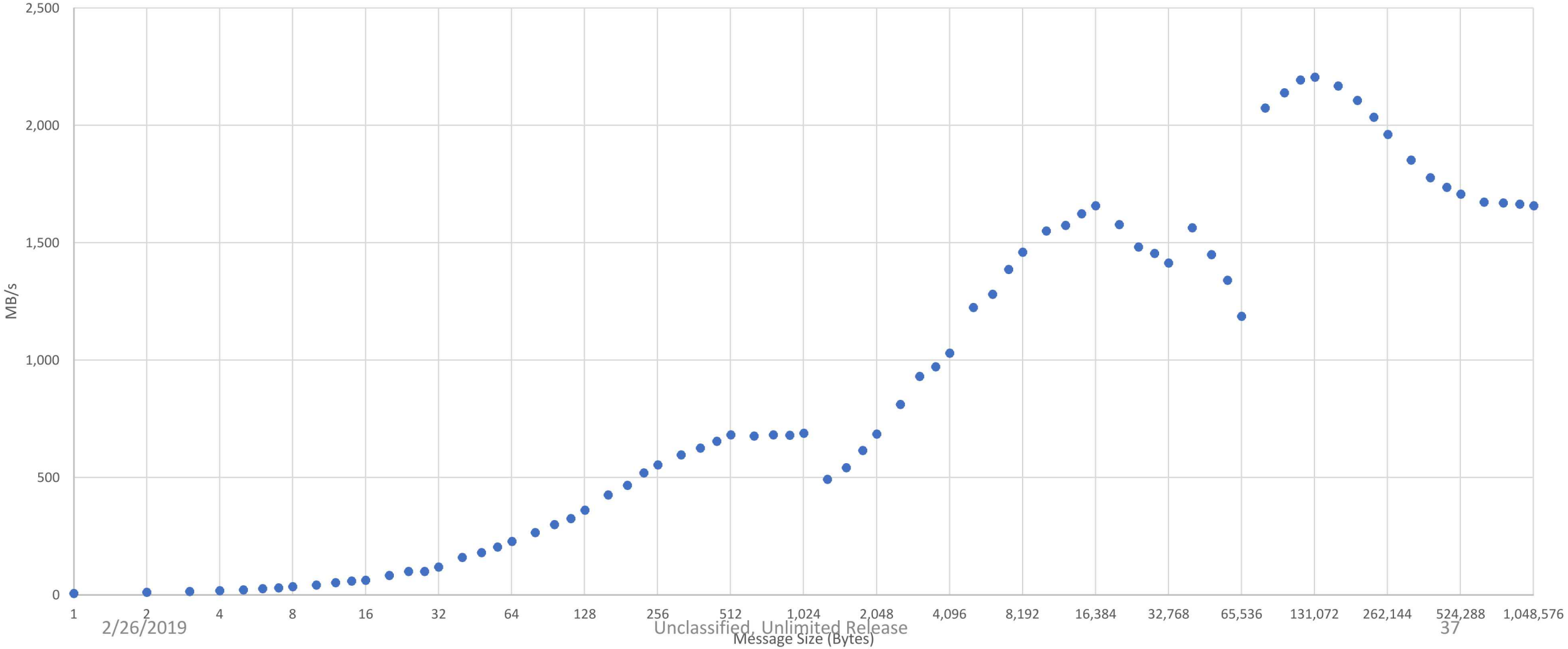
MPI_Sendrecv (36 Ranks / 2 Ranks)



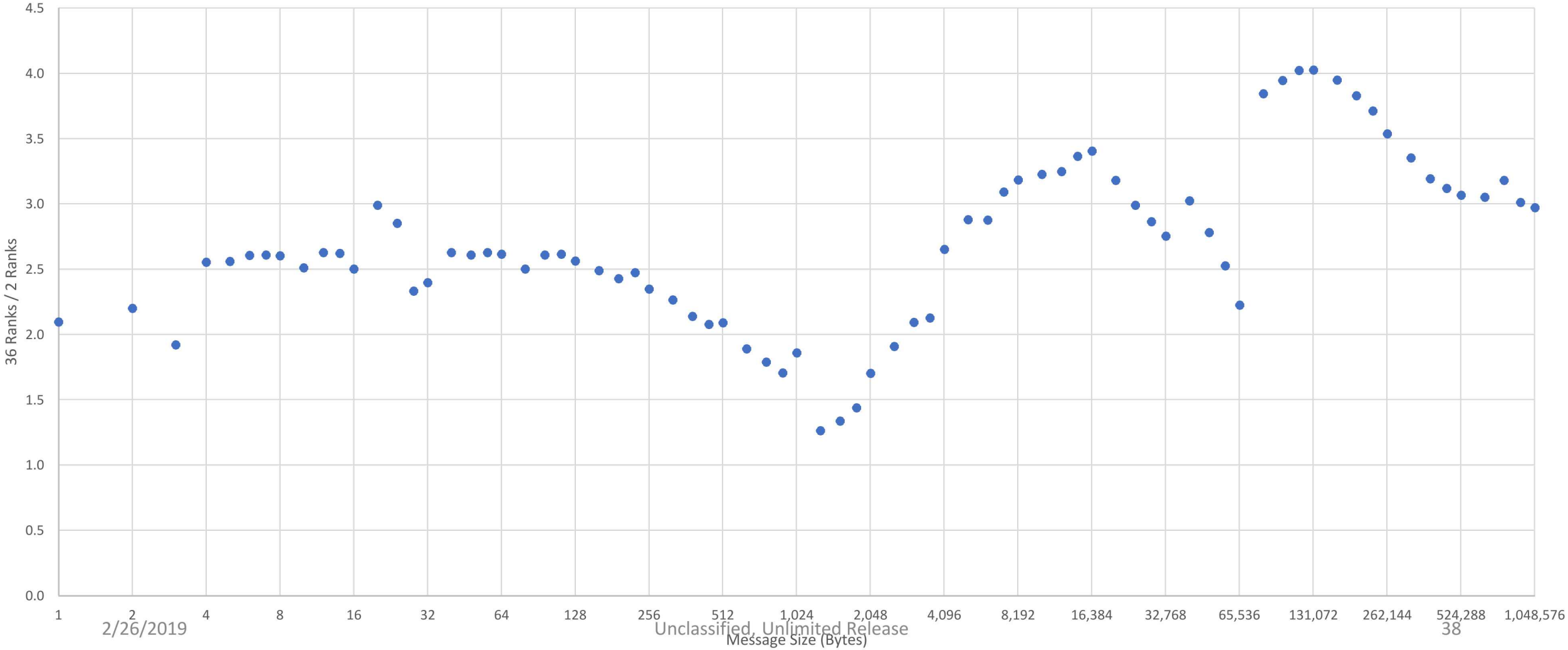
MPI_Allreduce (1 Node, 2 Ranks)



MPI_Allreduce (1 Node, 36 Ranks)



MPI_Allreduce (36 Ranks / 2 Ranks)



Questions?