

Intel Skylake Microbenchmarks



PRESENTED BY

Douglas M. Pase, PhD/Sandia National Laboratories

With Anthony M. Agelastos and Joel O. Stevenson

SAND2019-XXXX PE

UNCLASSIFIED, UNLIMITED RELEASE

UNCLASSIFIED



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.



Unclassified, Unlimited Release

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.)

Contents

- Processor Description
- DAXPY - Cache and memory throughput (vector and scalar)
- DGEMM - Vector floating-point performance
- Stream - Memory throughput
- Gather - Indirect loads
- Scatter - Indirect stores
- MPI_Sendrecv and MPI_Allreduce - Kernel transition and communication performance

Intel Xeon Platinum 8160 (Sky Lake)

- 14 nm lithography
- 24 cores per socket
- 2.10 GHz - 3.70 GHz clock
- 3 x 10.4(?) GT/s UPI
- 2666 MHz DDR4
- 6 memory channels/socket
- AVX-512
- TDP 150 watts
- 768 KiB L1 inst. cache/core
- 768 KiB L1 data cache/core
- 1 MiB L2 combined cache/core
- 33 MB L3/skt (1.375 MiB/core)
- 128-entry ITLB (4 KiB pages) (?)
- 64-entry DTLB (4 KiB pages) (?)
- 1538-entry STLB (4 KiB pages) (?)

DAXPY

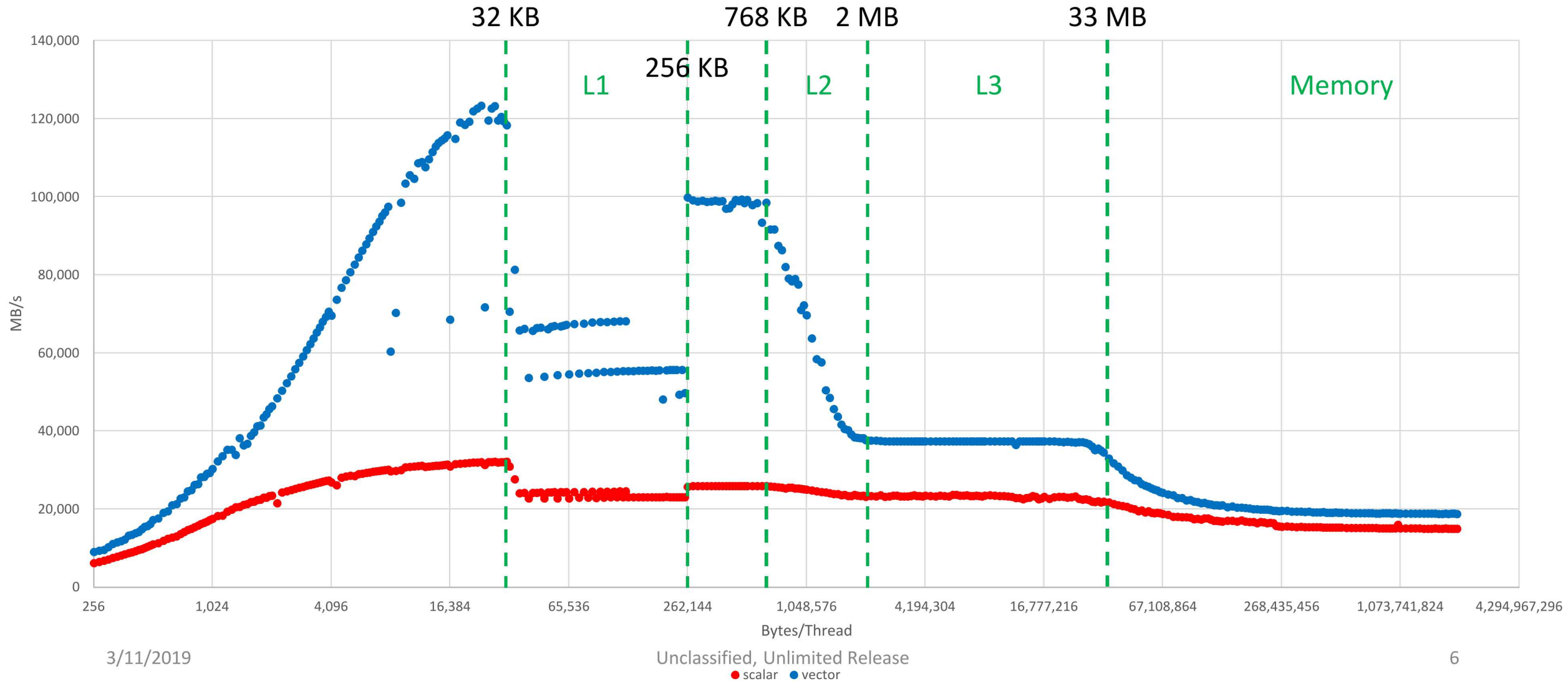
$$y[i] = \alpha * x[i] + y[i]$$

x , y are double-precision floating-point vectors

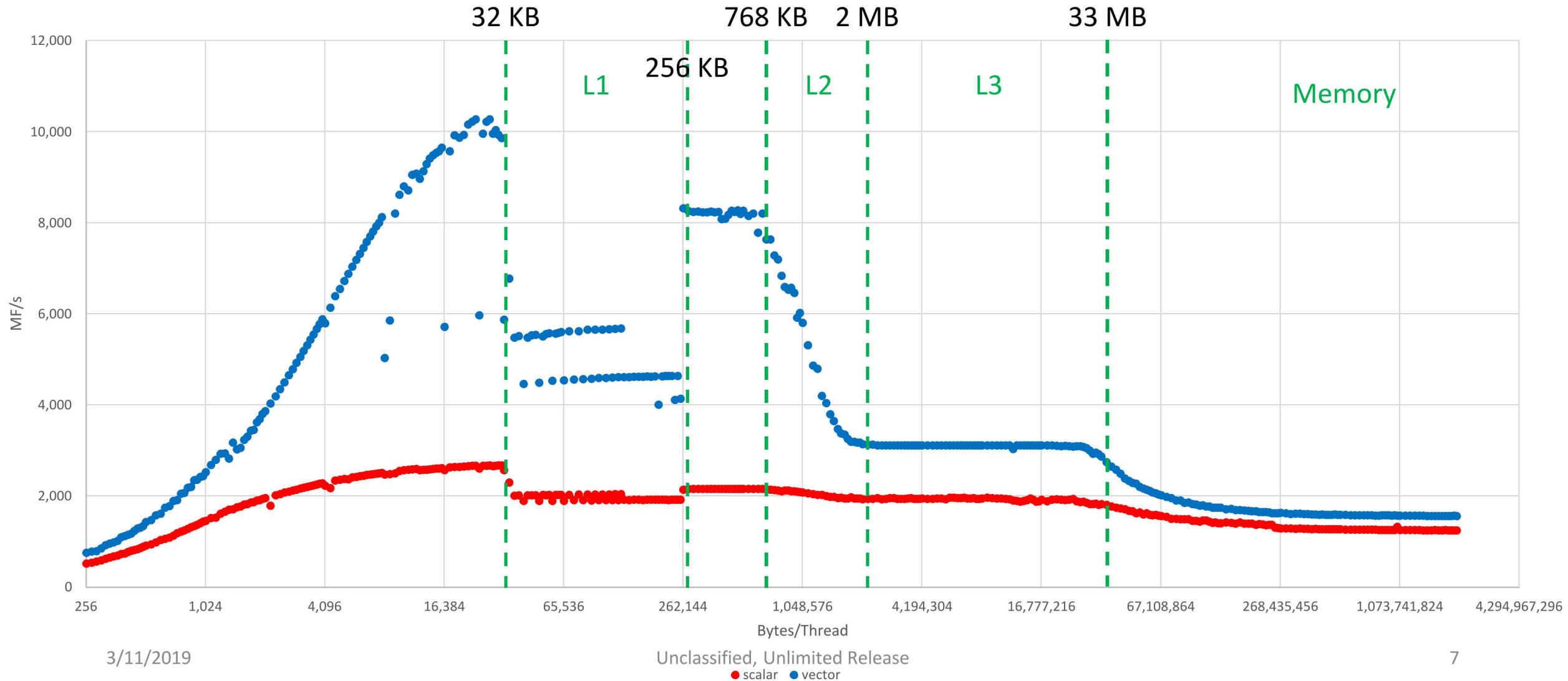
α is a double-precision floating-point scalar

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

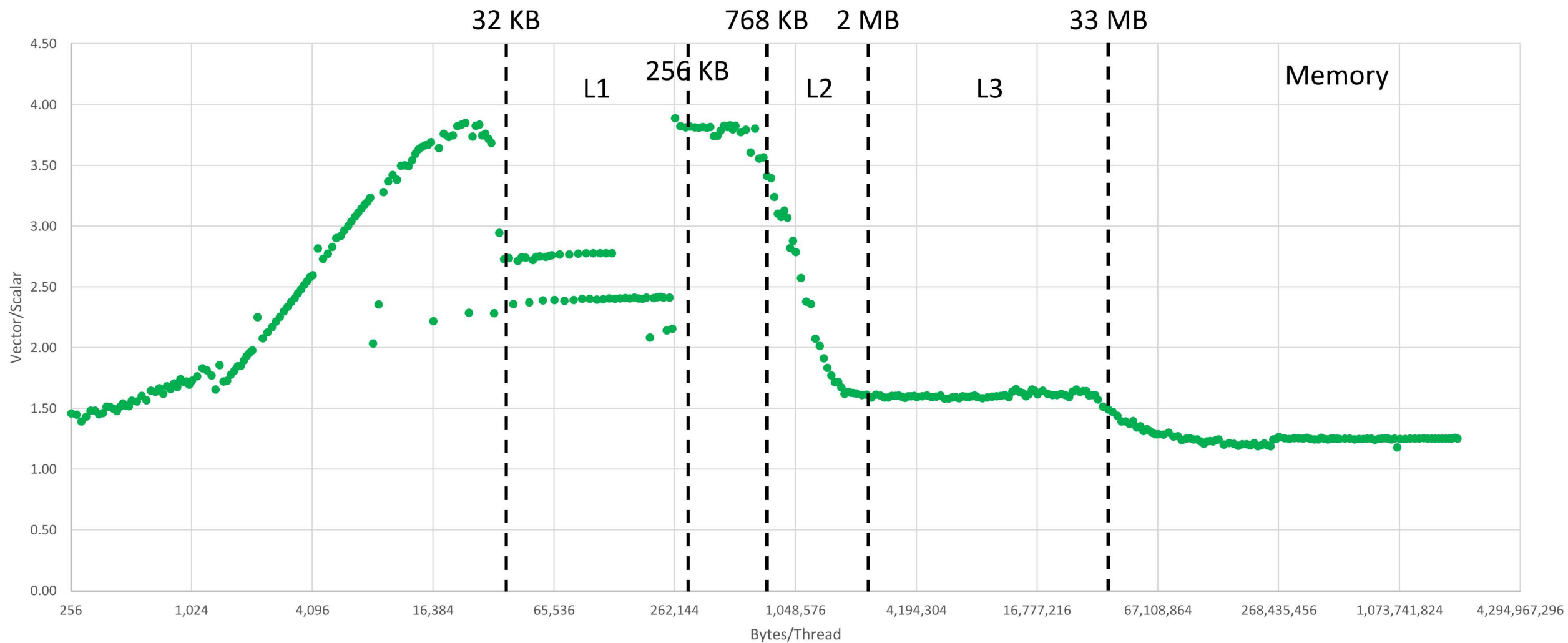
DAXPY Memory Throughput (One Core)



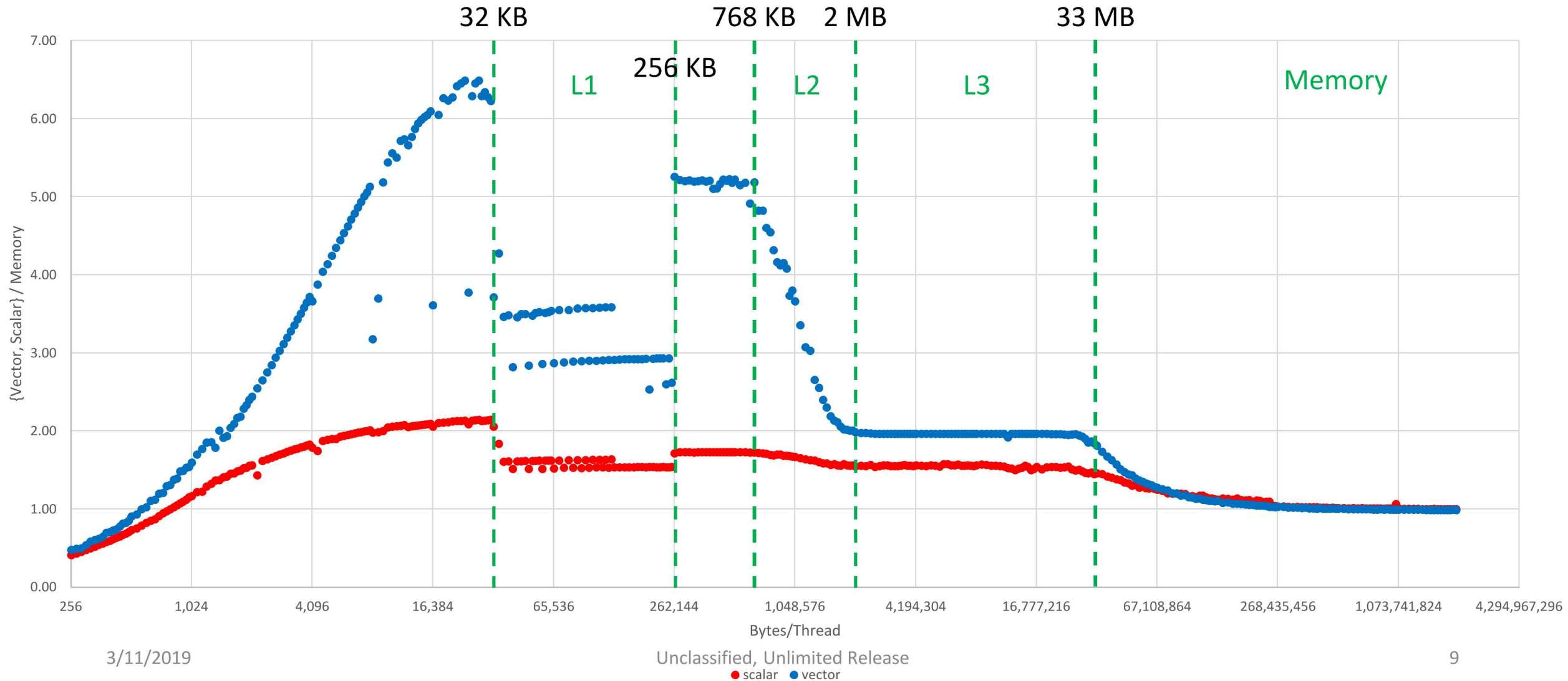
DAXPY Floating-Point Throughput (One Core)



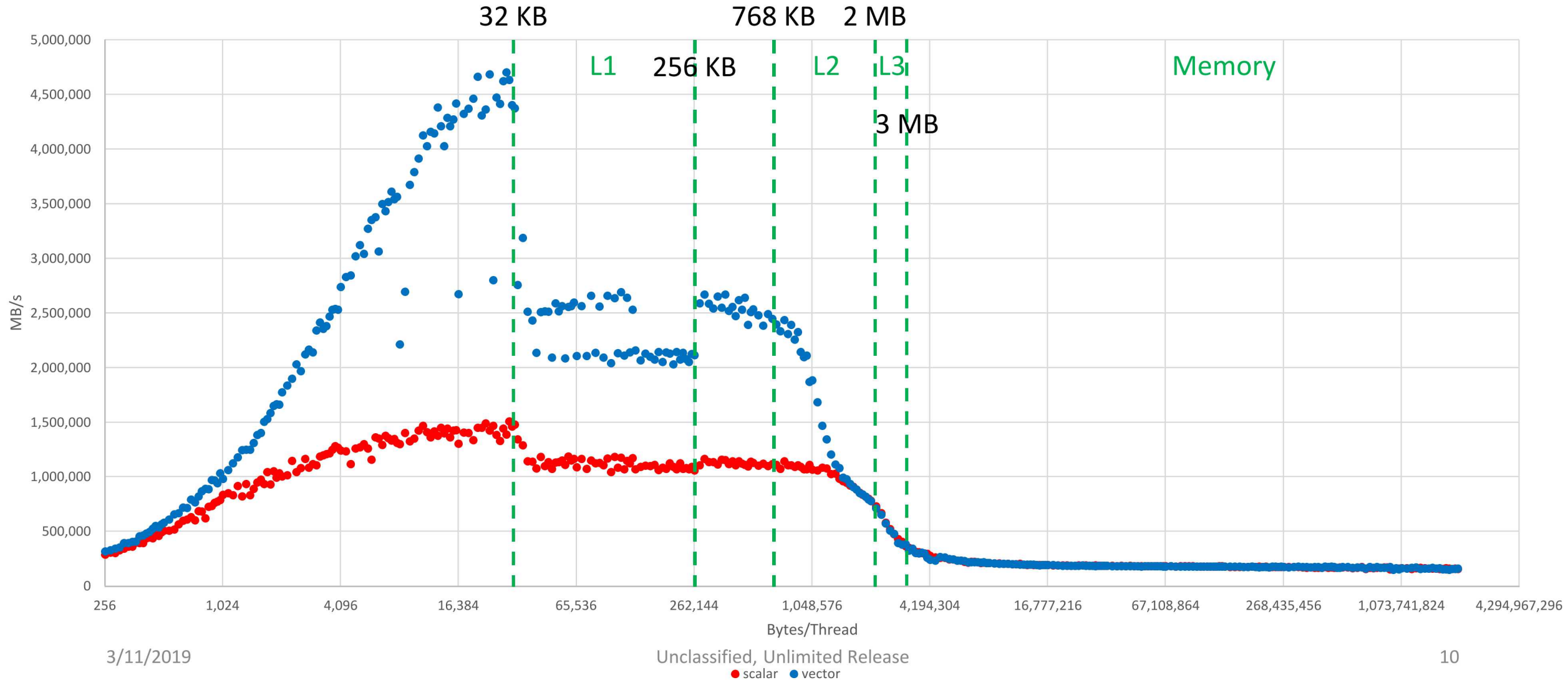
DAXPY Vector/Scalar (One Core)



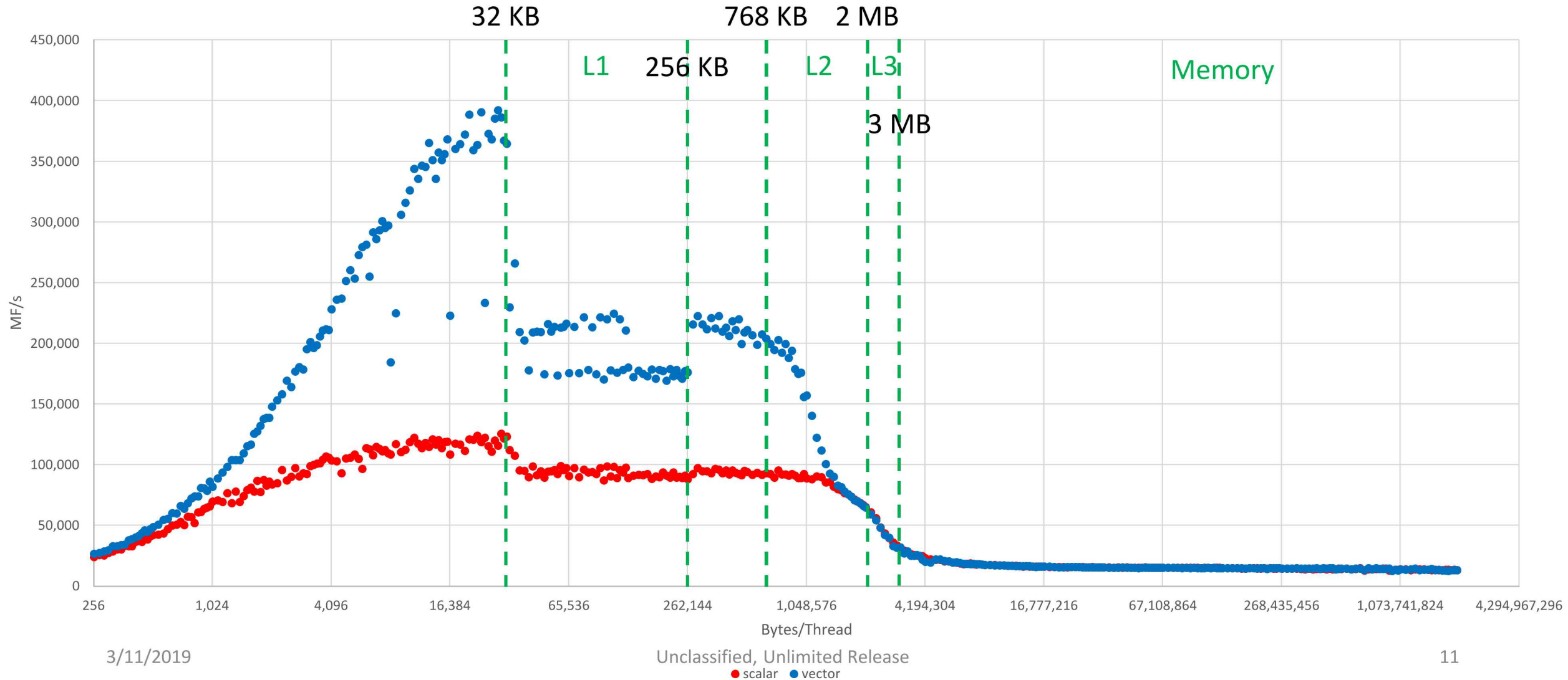
Performance Relative to Memory (One Core)



DAXPY Memory Throughput (48 Cores)



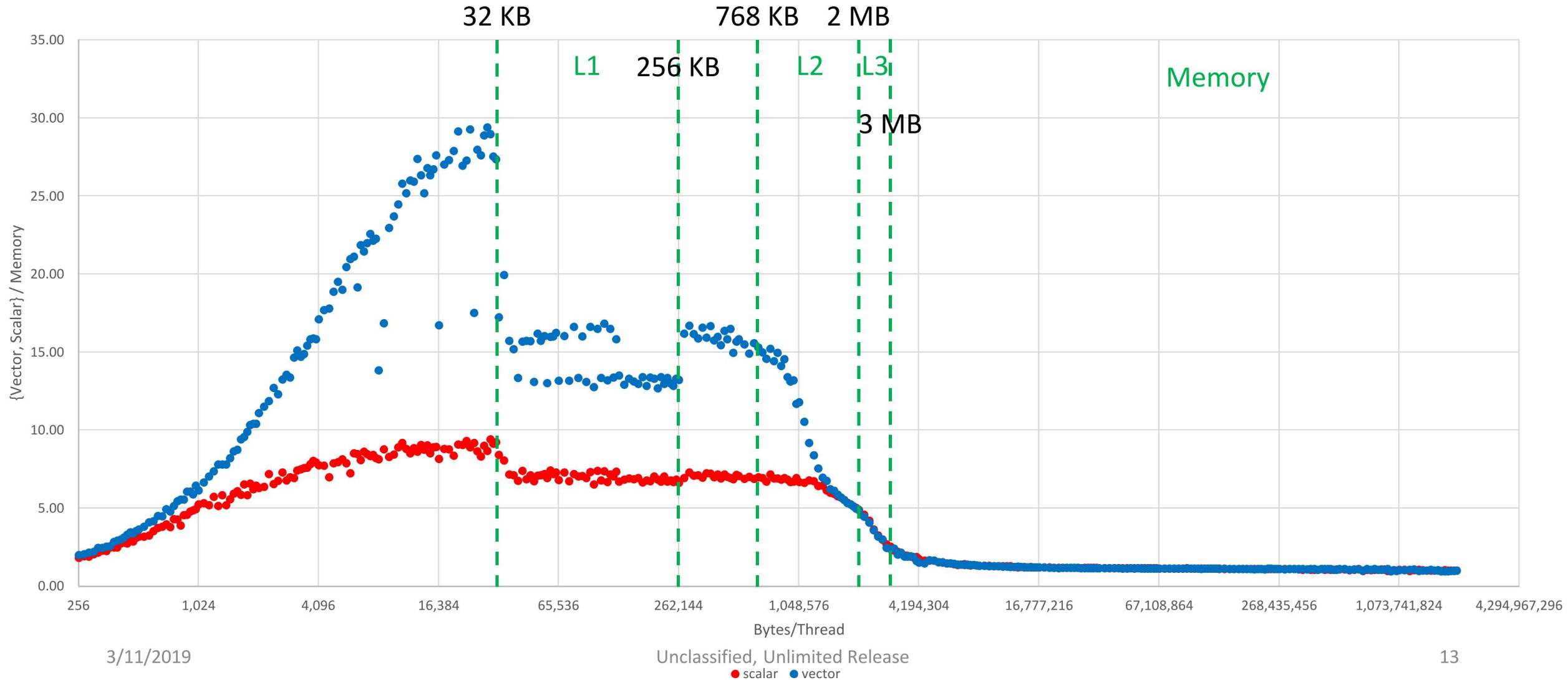
DAXPY Floating-Point Throughput (48 Cores)



DAXPY Vector/Scalar (48 Cores)



Performance Relative to Memory (48 Cores)



DGEMM

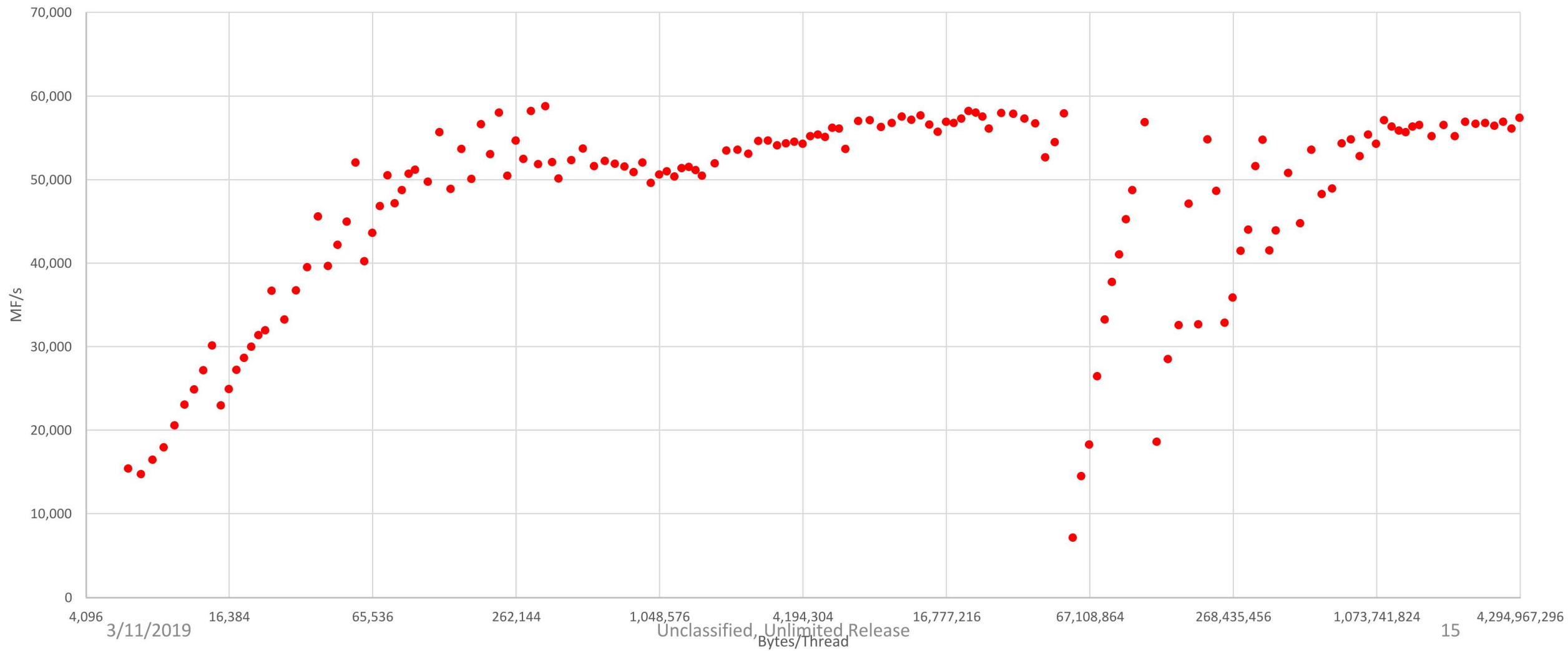
$$\underline{C} = \alpha * (\underline{A} \times \underline{B}) + \beta * \underline{C}$$

$\underline{A}$, $\underline{B}$, $\underline{C}$ are double-precision floating-point matrices

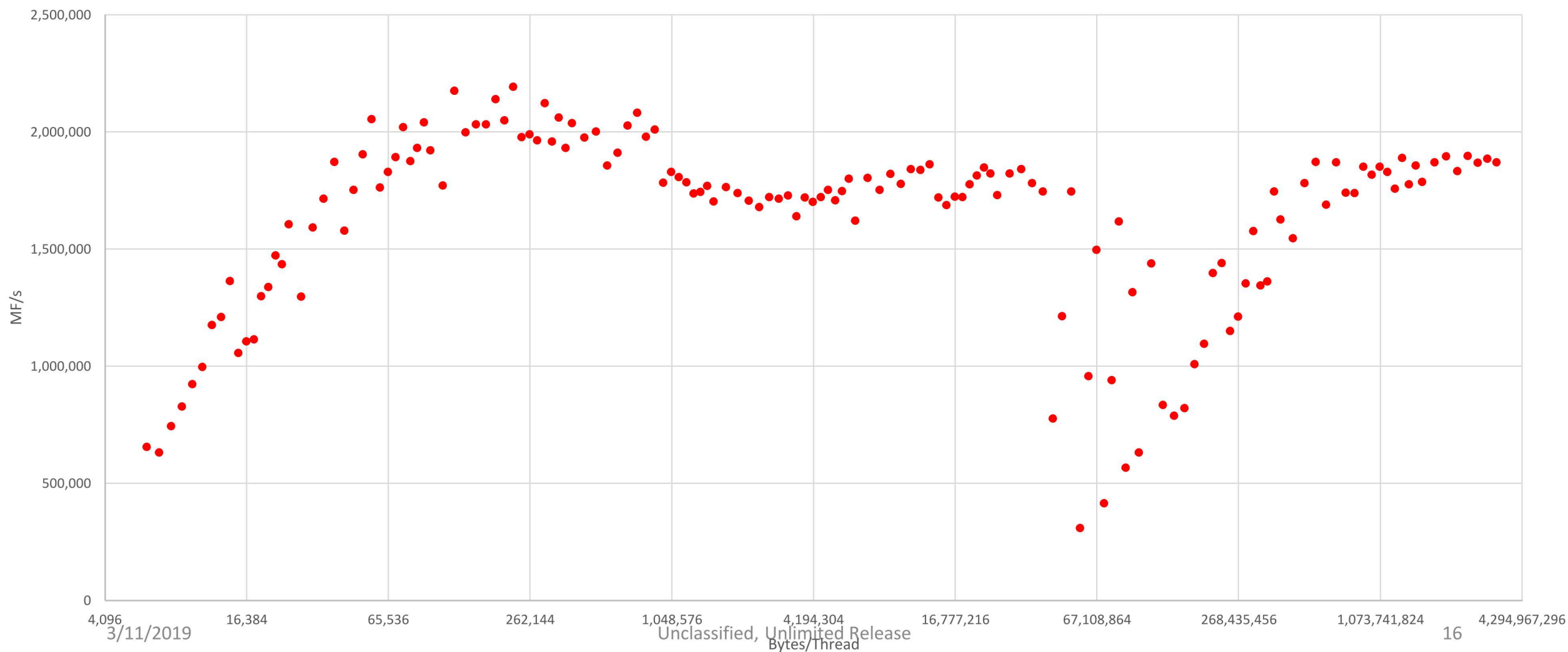
α , β are double-precision floating-point scalars

$*$ is the scalar multiplication operator; $\times$ is the matrix multiplication operator

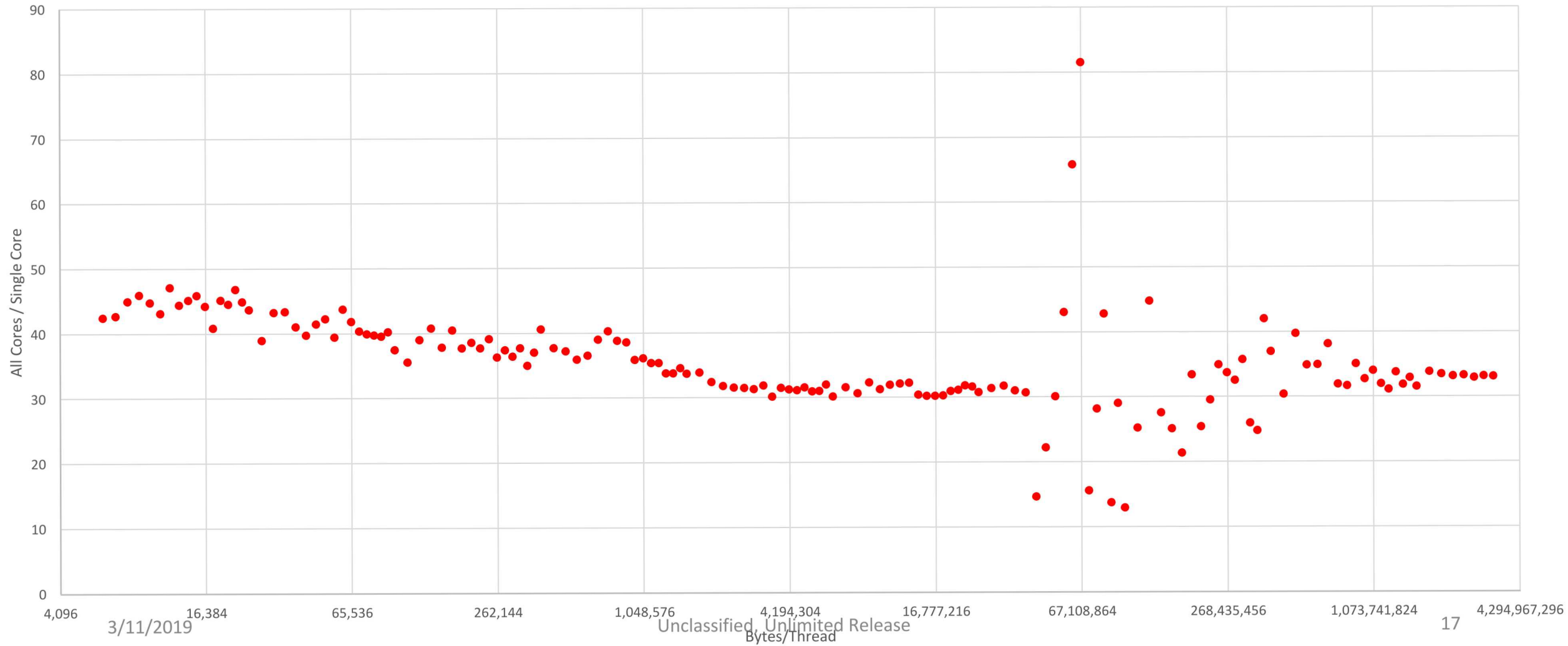
DGEMM MF/s (One Core)



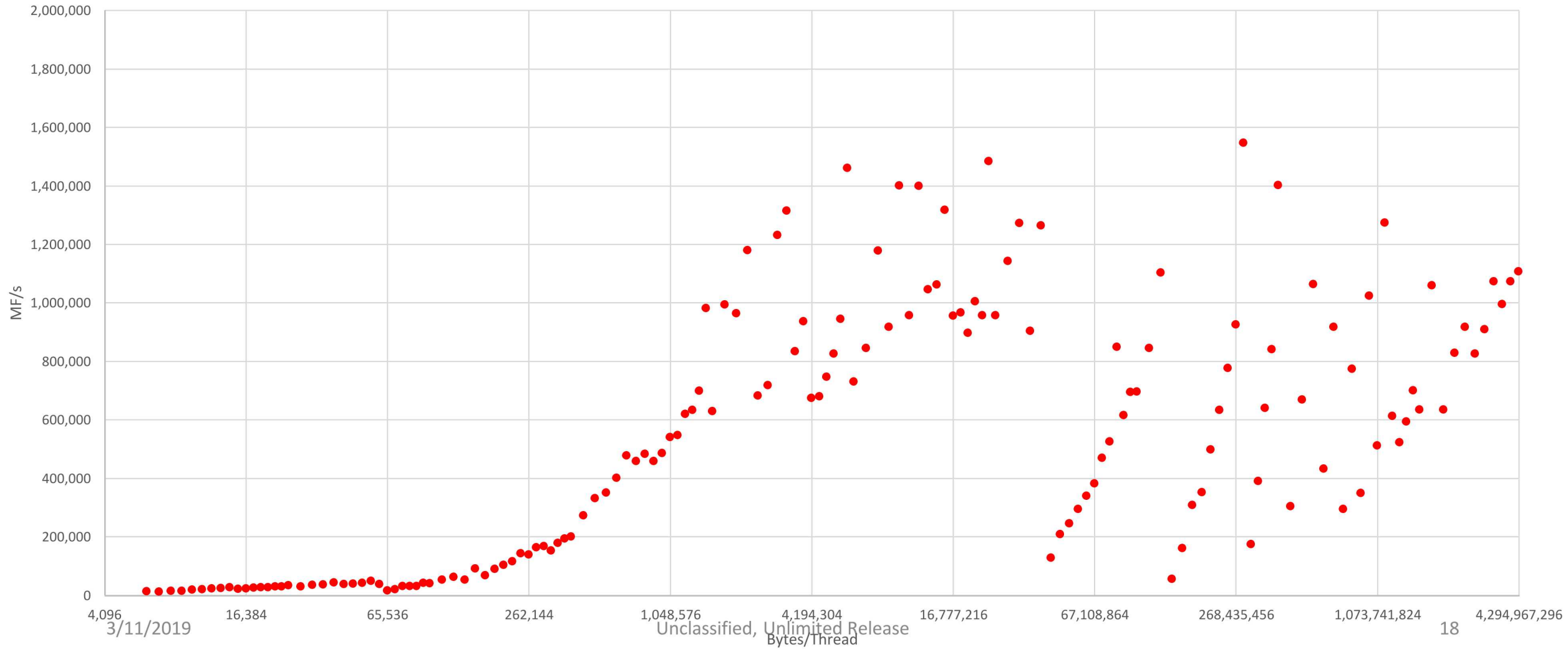
DGEMM MF/s (48 Cores)



Ratio of 48 Cores / 1 Core



OMP Version (48 Cores)



Stream

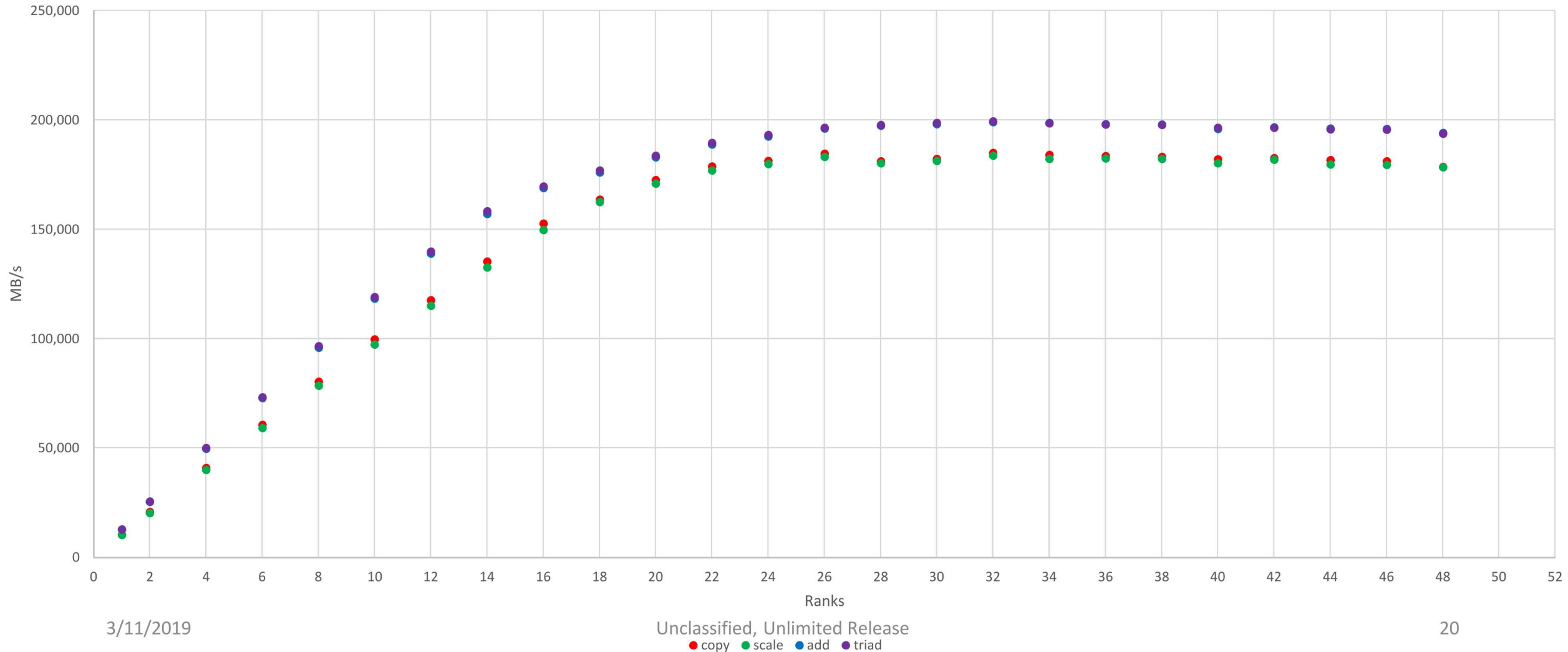
Copy: $y[i] = x[i]$

Scale: $y[i] = \alpha * x[i]$

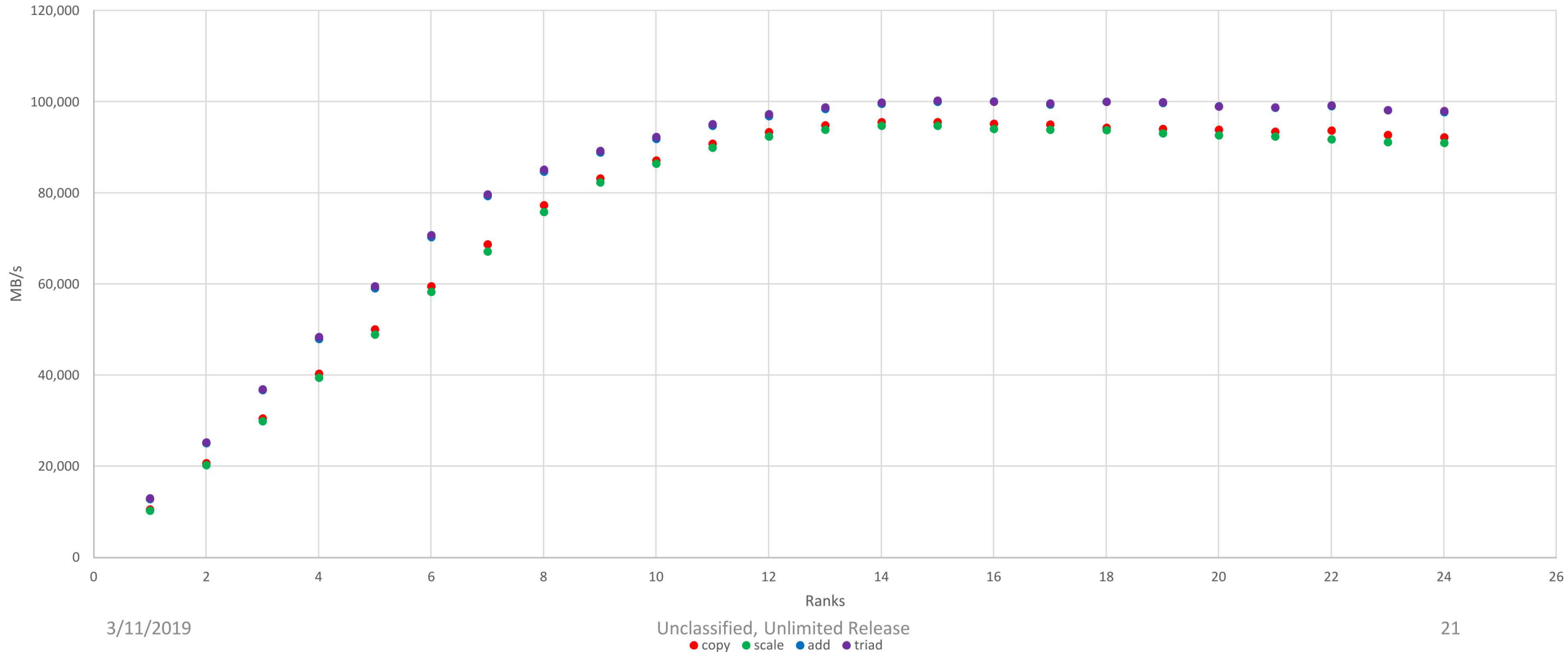
Add: $z[i] = x[i] + y[i]$

Triad: $z[i] = x[i] + \alpha * y[i]$

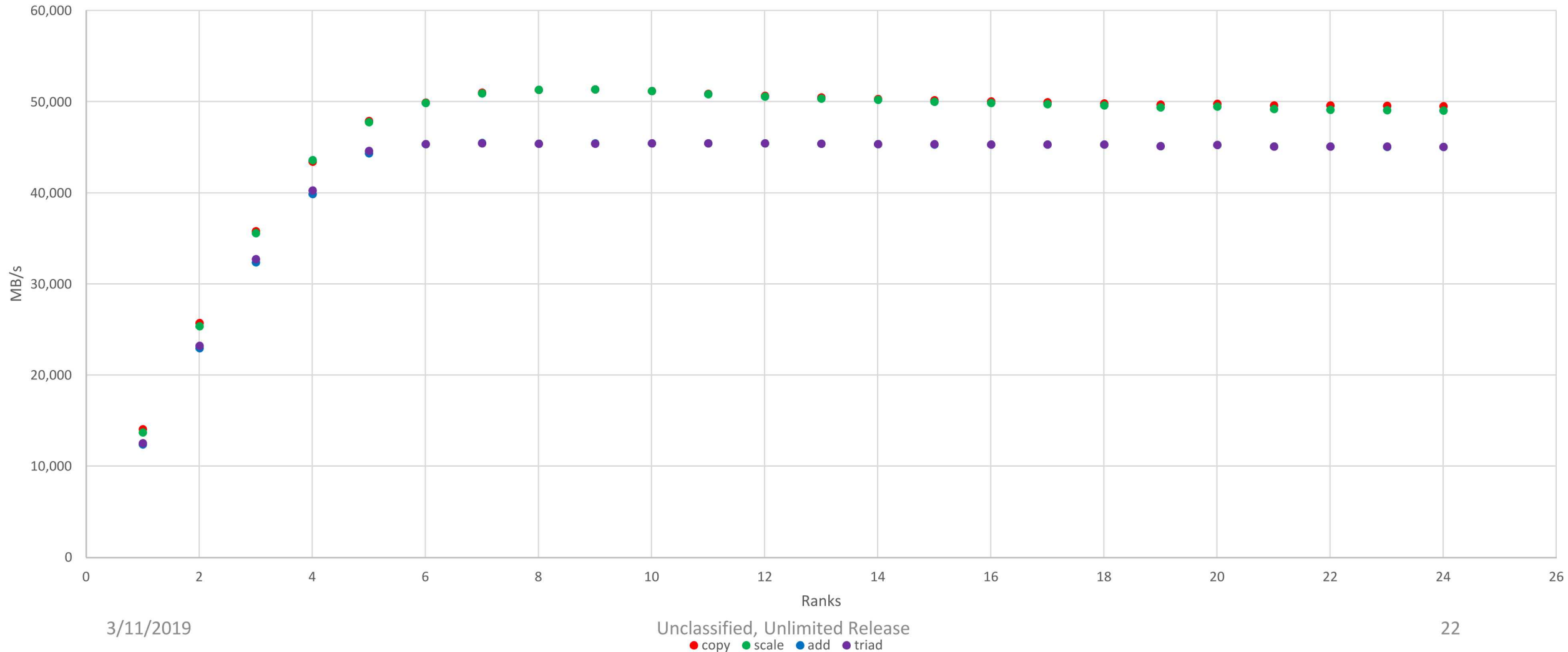
Stream Memory Throughput (2 Sockets)



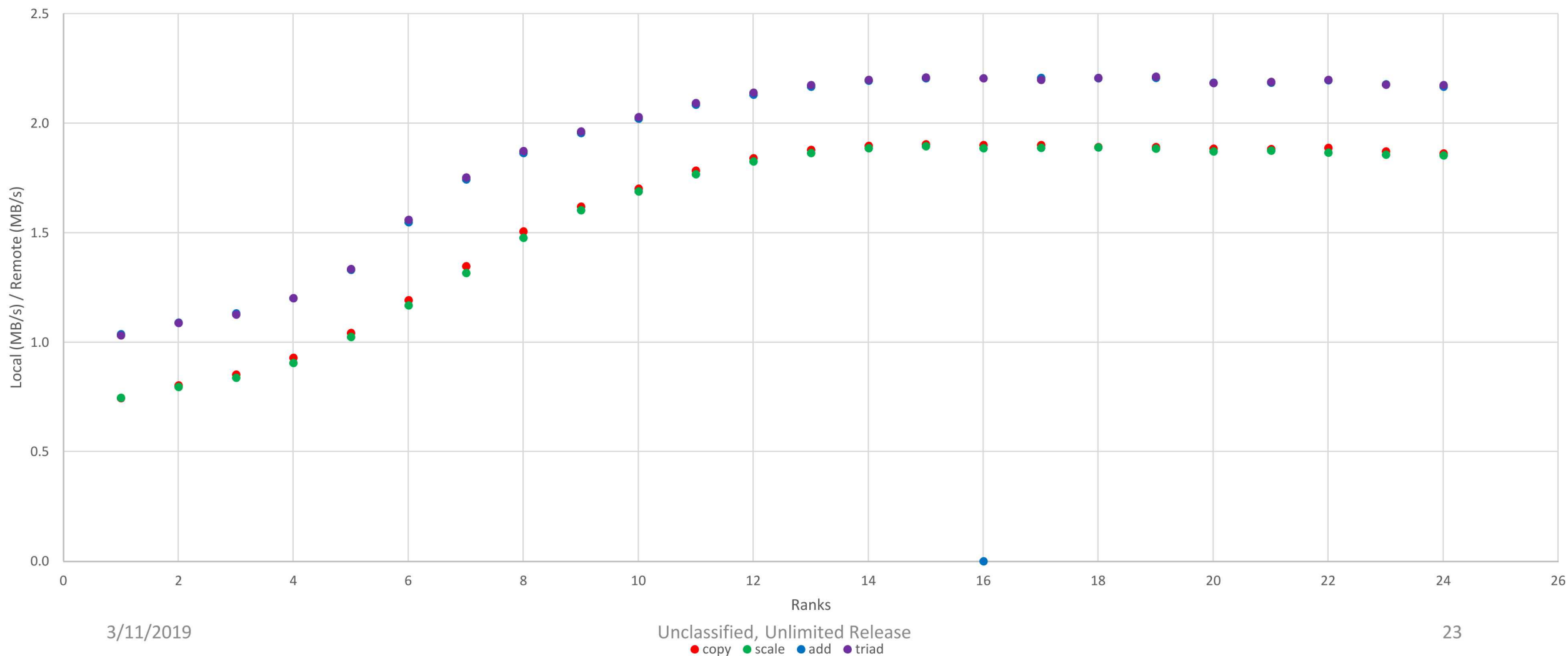
Local Memory Throughput (1 Socket)



Remote Memory Throughput (1 Socket)



Local vs. Remote Memory B/W (1 Socket)



Gather

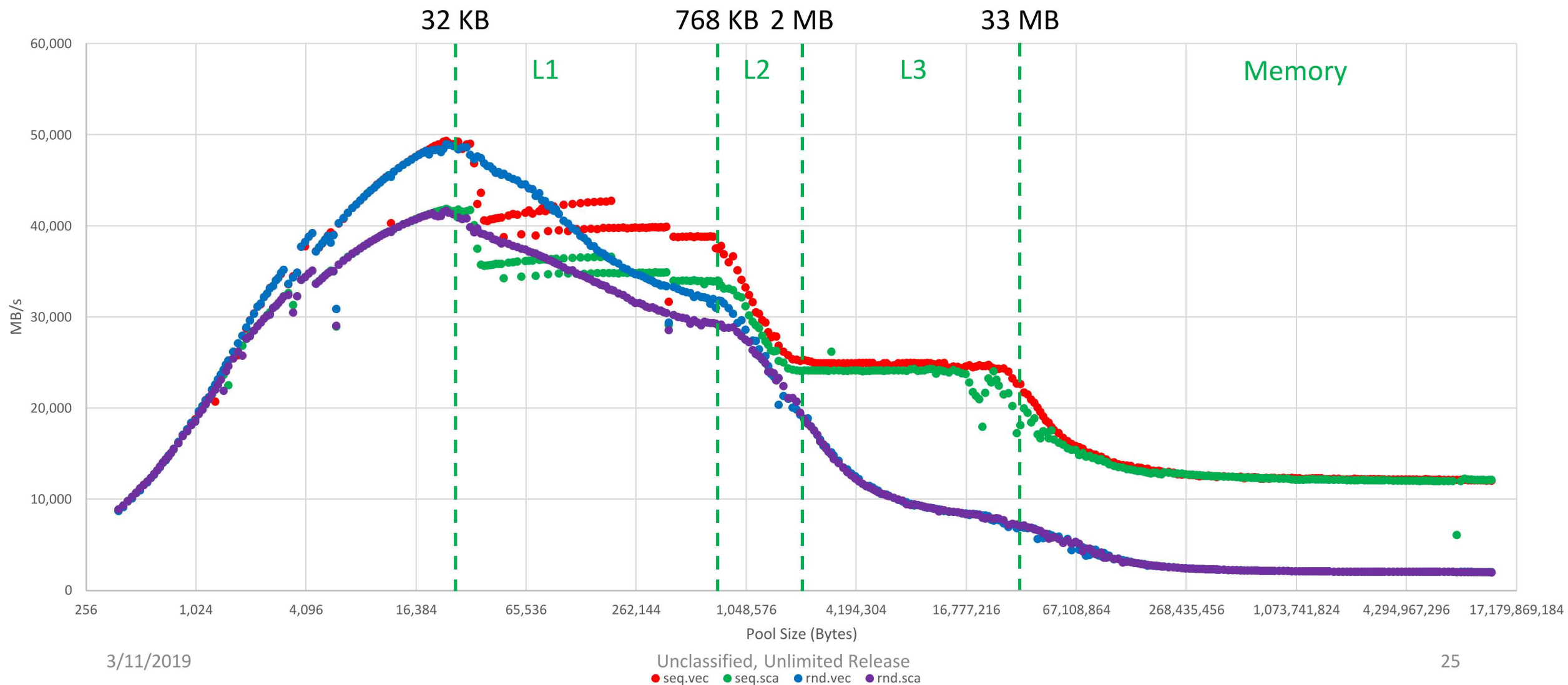
$y[i] = x[v[i]]$

x, y are double-precision floating-point vectors

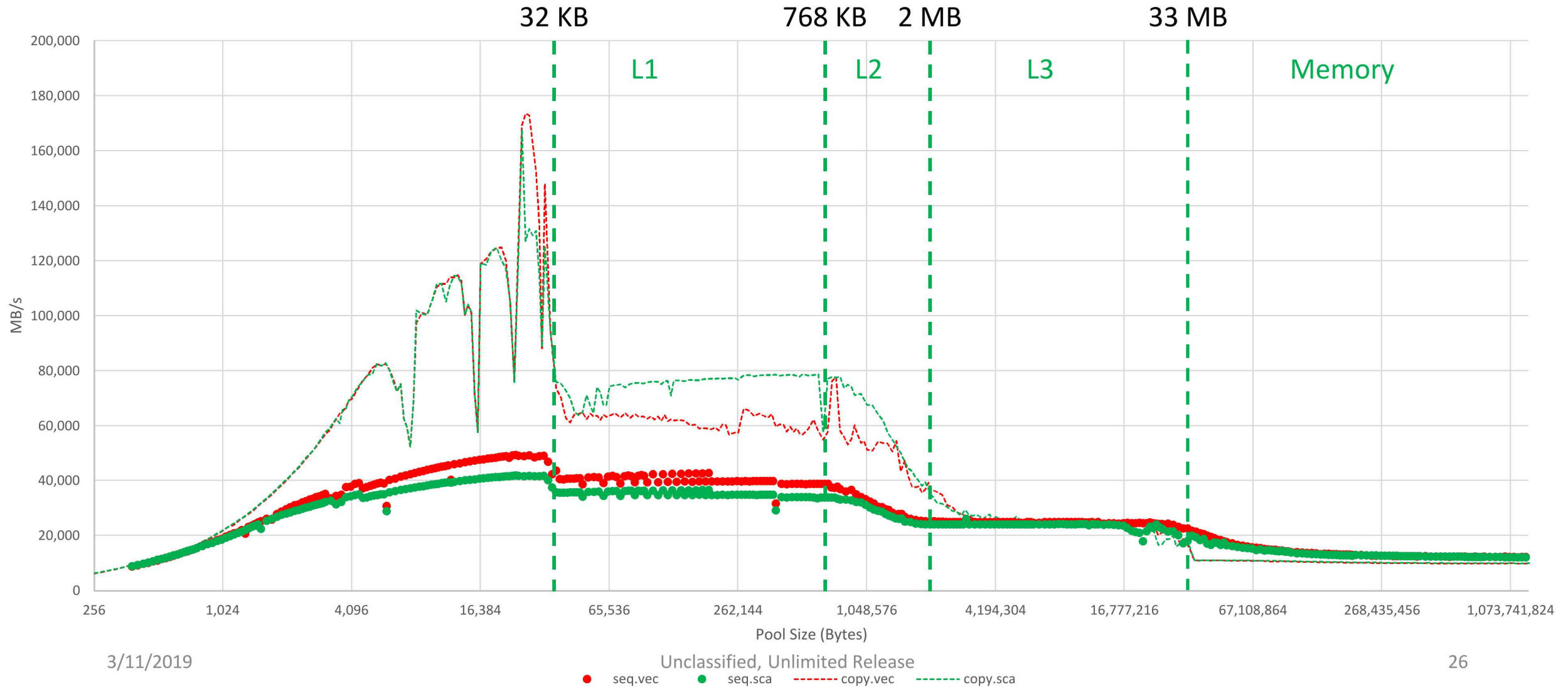
v is a 64-bit integer vector index into x

i is a 64-bit integer index

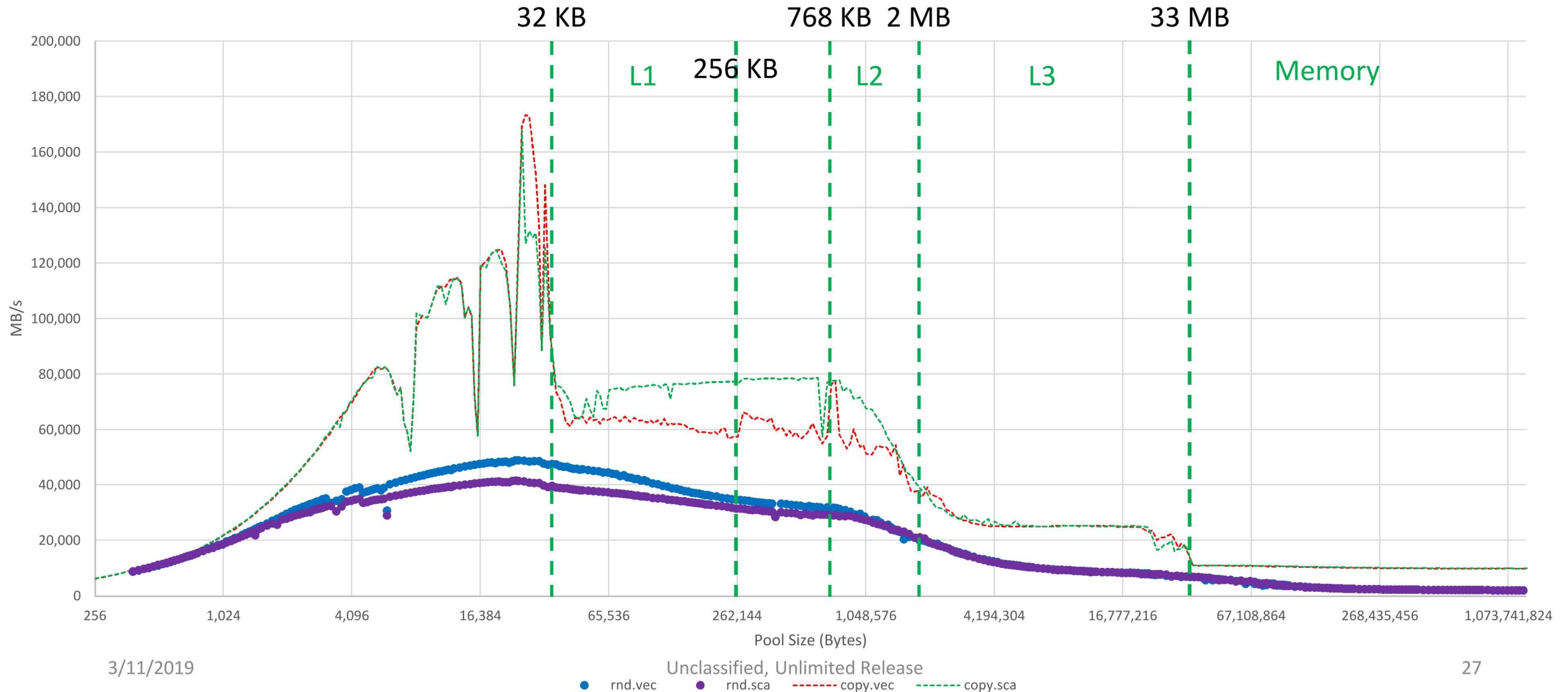
Gather MB/s (One Core)



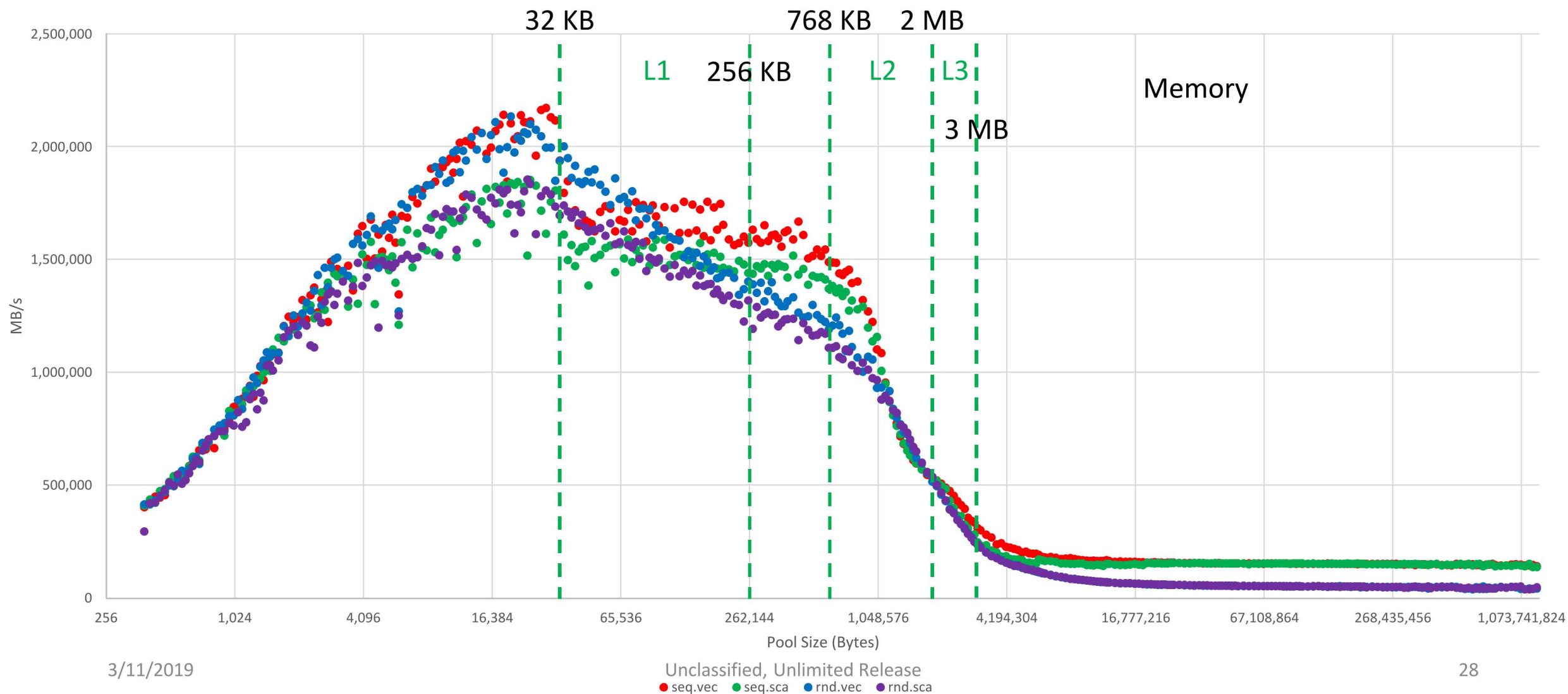
Sequential Gather MB/s (One Core)



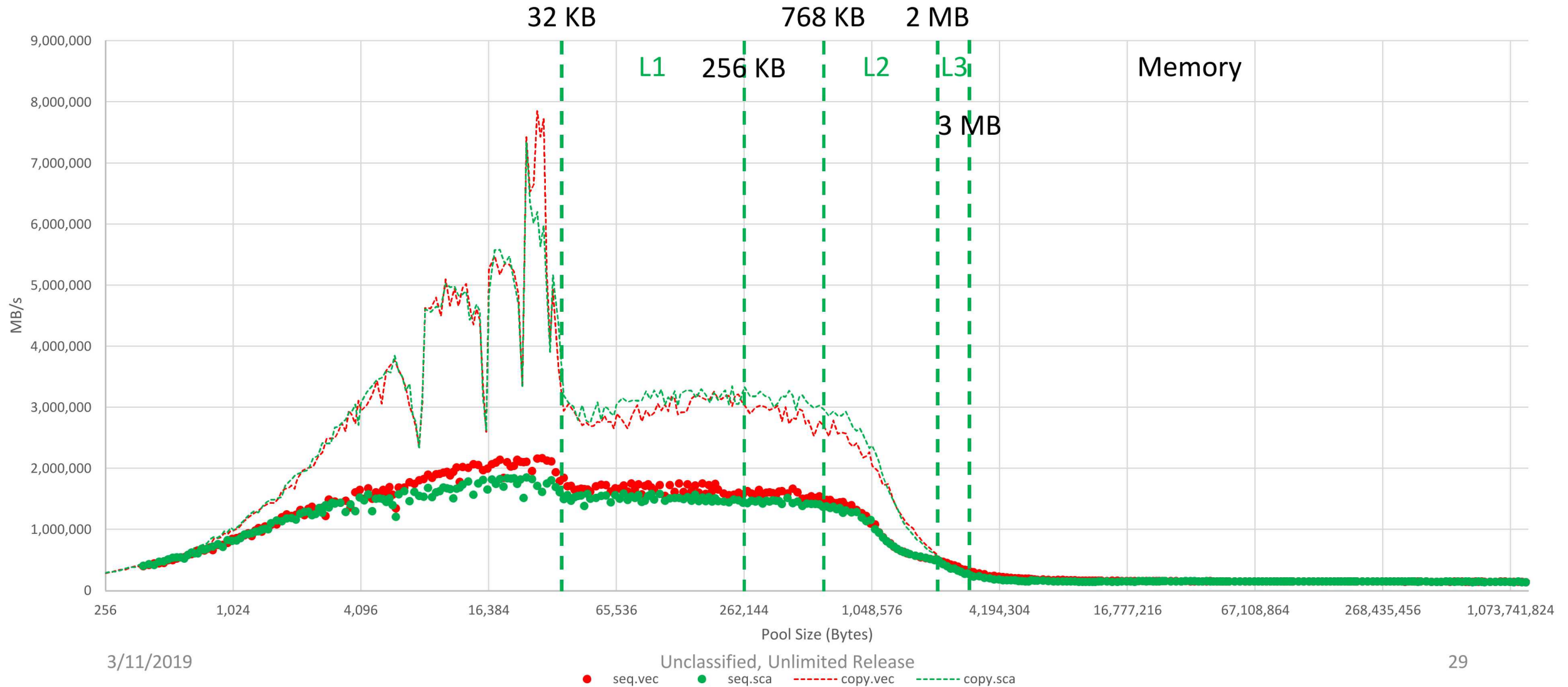
Random Gather MB/s (One Core)



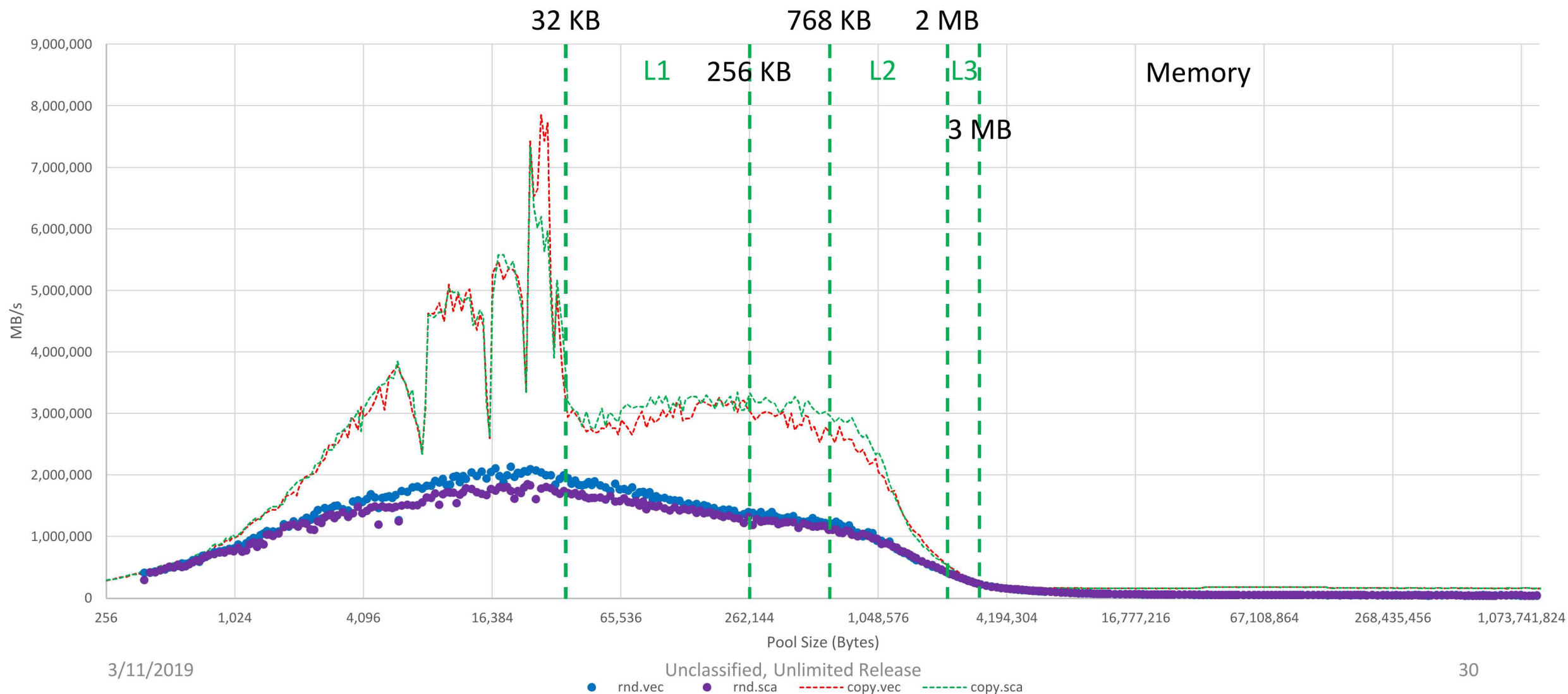
Gather MB/s (48 Cores)



Sequential Gather MB/s (48 Cores)



Random Gather MB/s (48 Cores)



Scatter

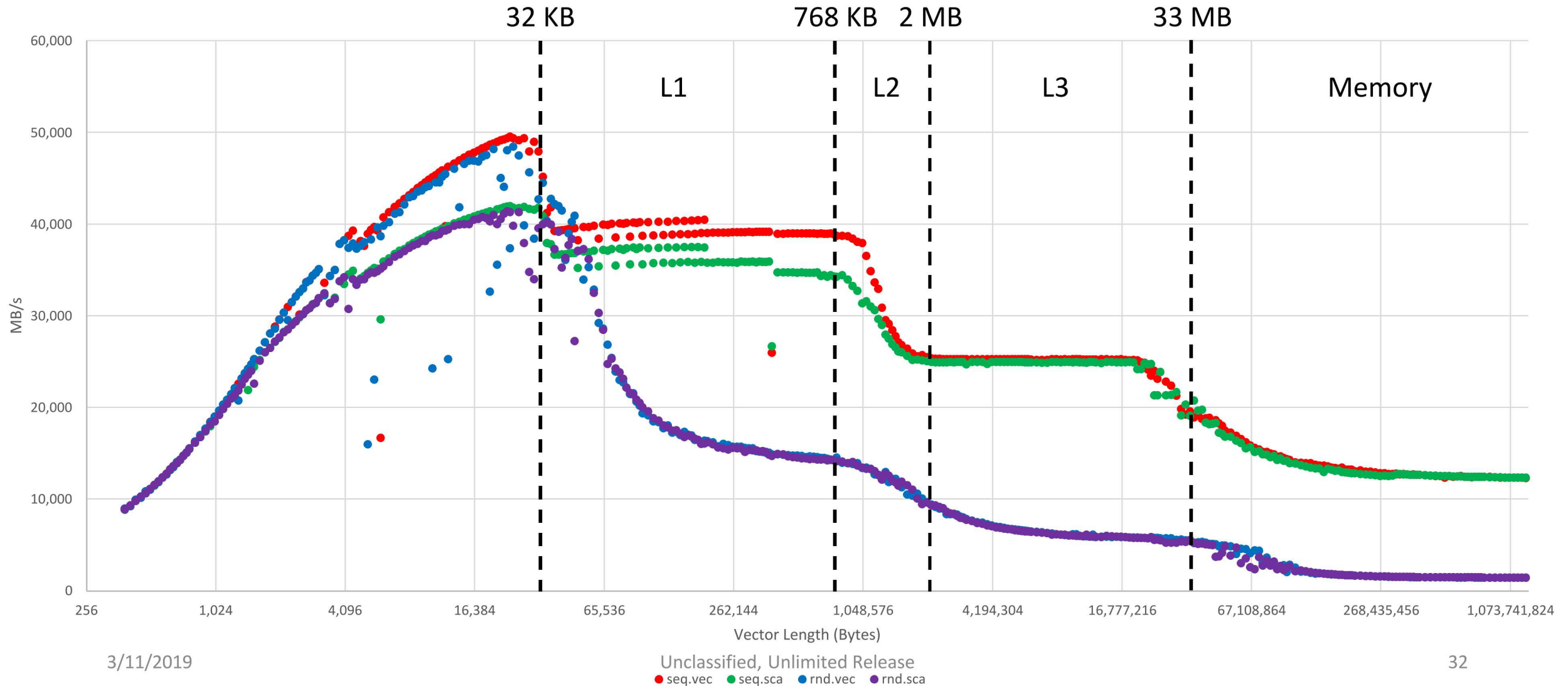
$y[v[i]] = x[i]$

x, y are double-precision floating-point vectors

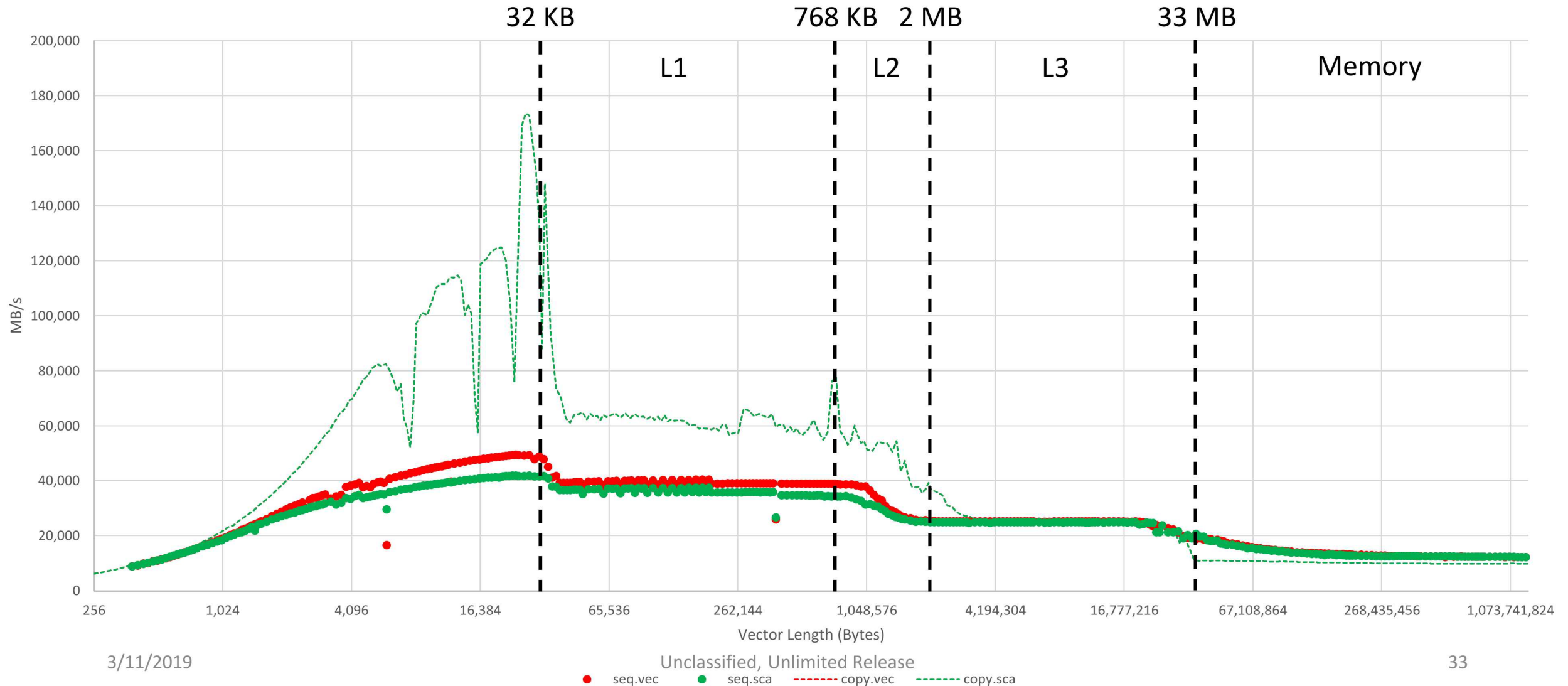
v is a 64-bit integer vector index into x

i is a 64-bit integer index

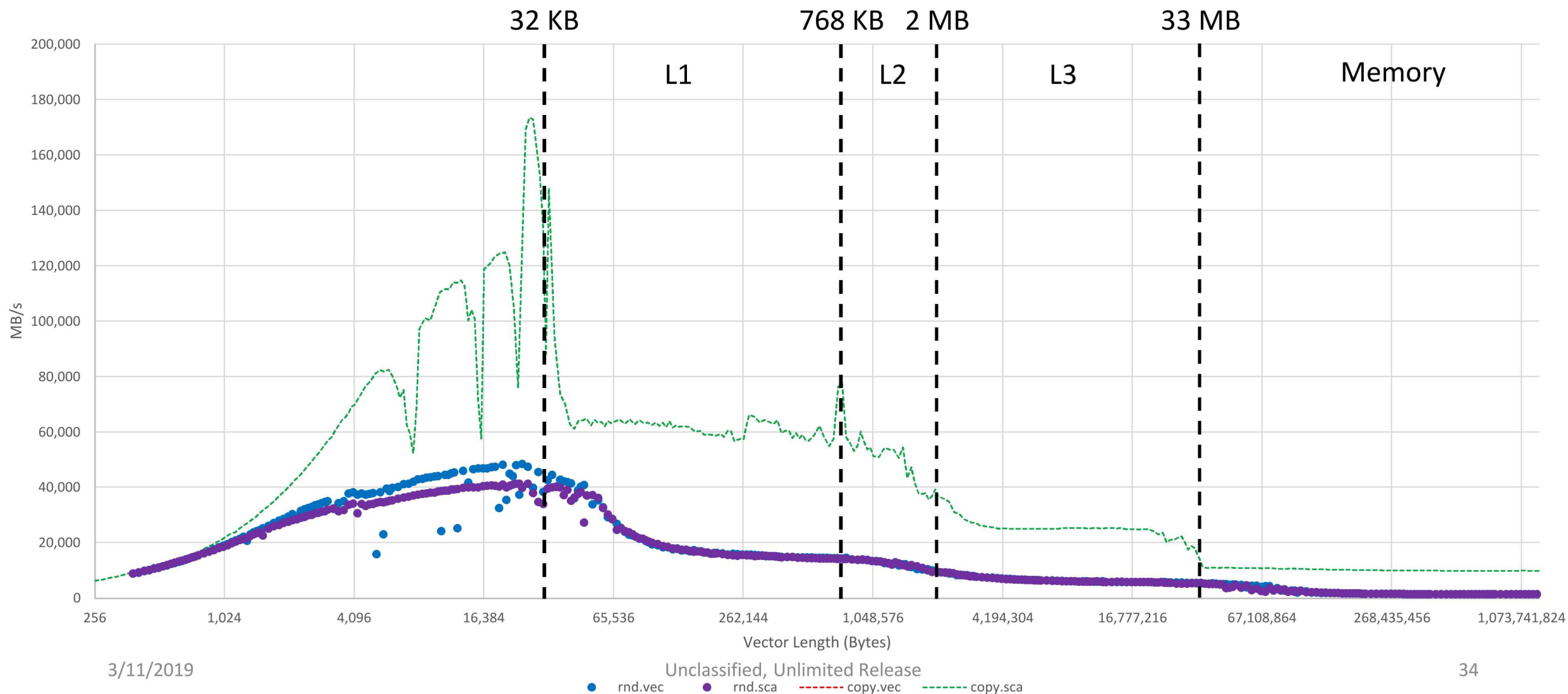
Scatter MB/s (One Core)



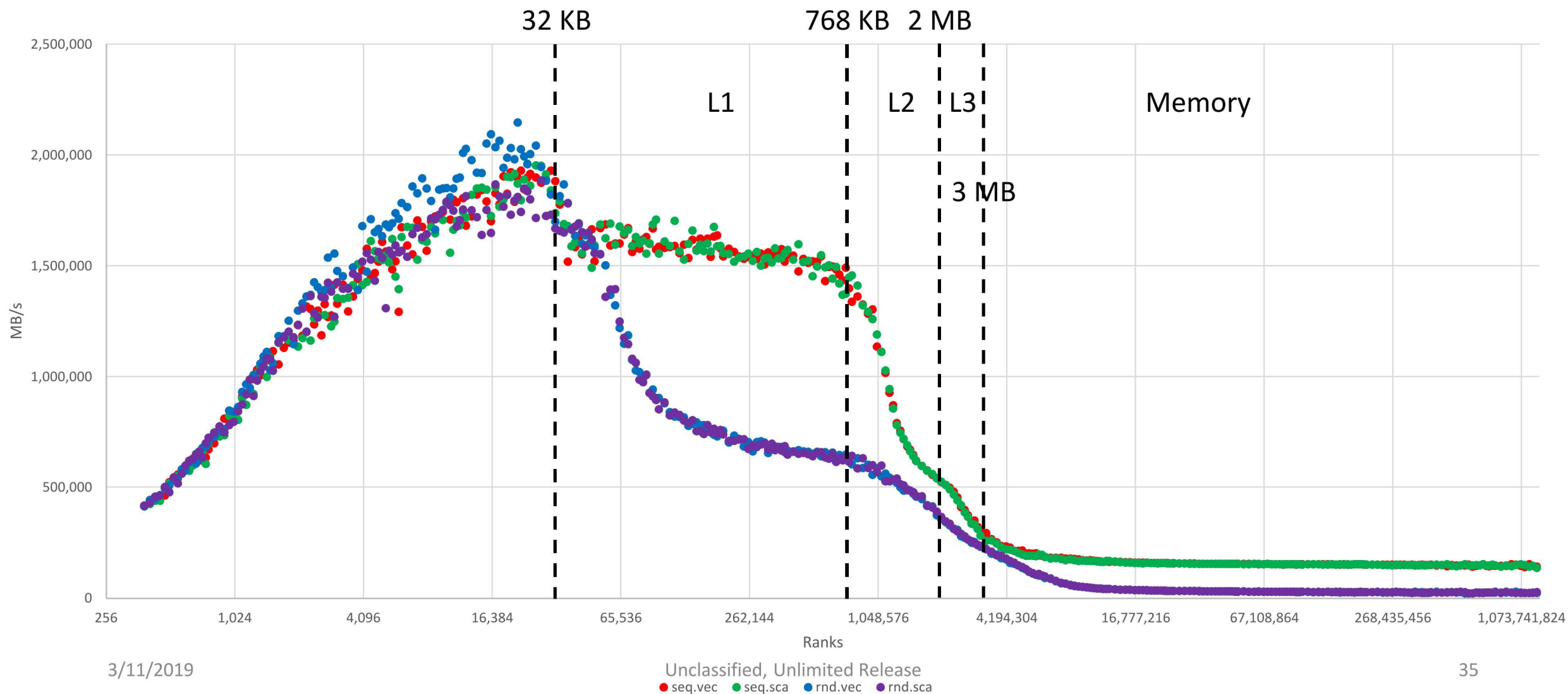
Sequential Scatter MB/s (One Core)



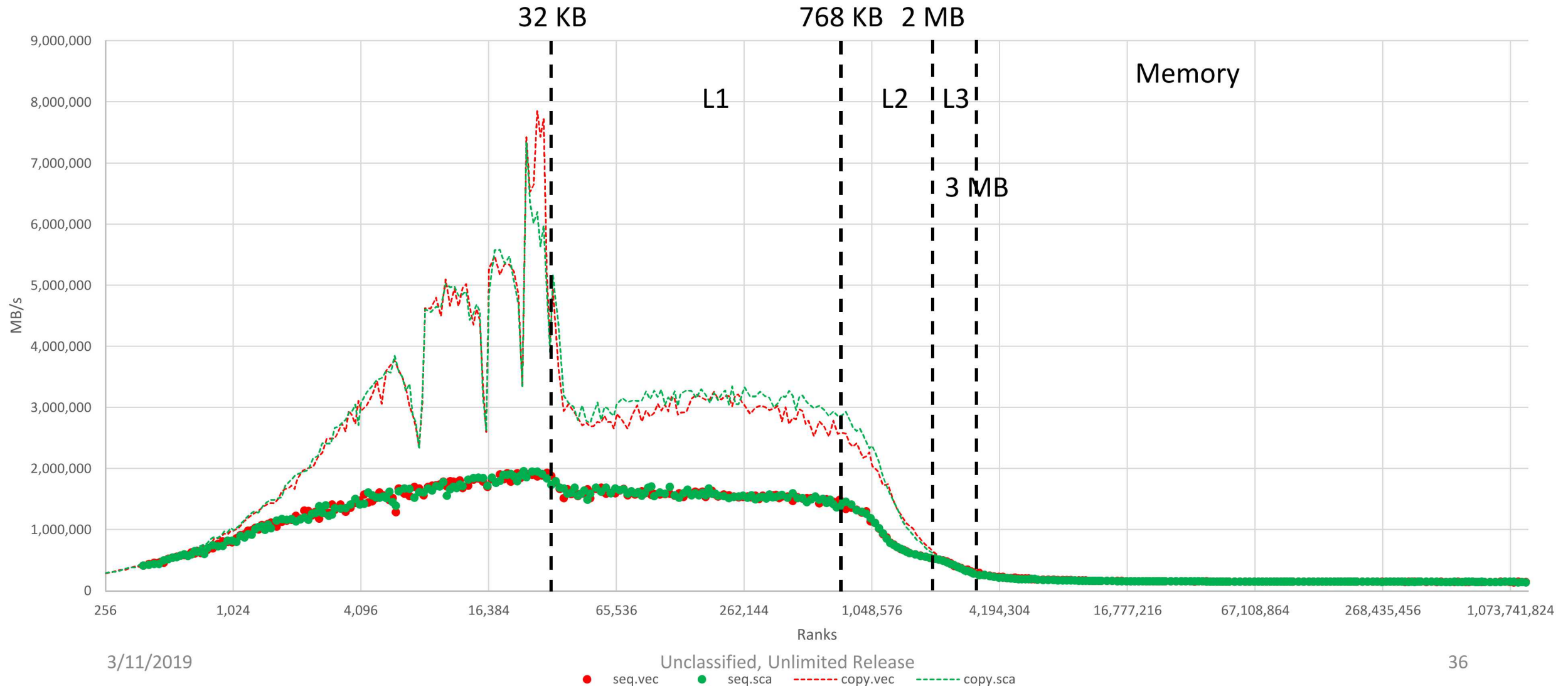
Random Scatter MB/s (One Core)



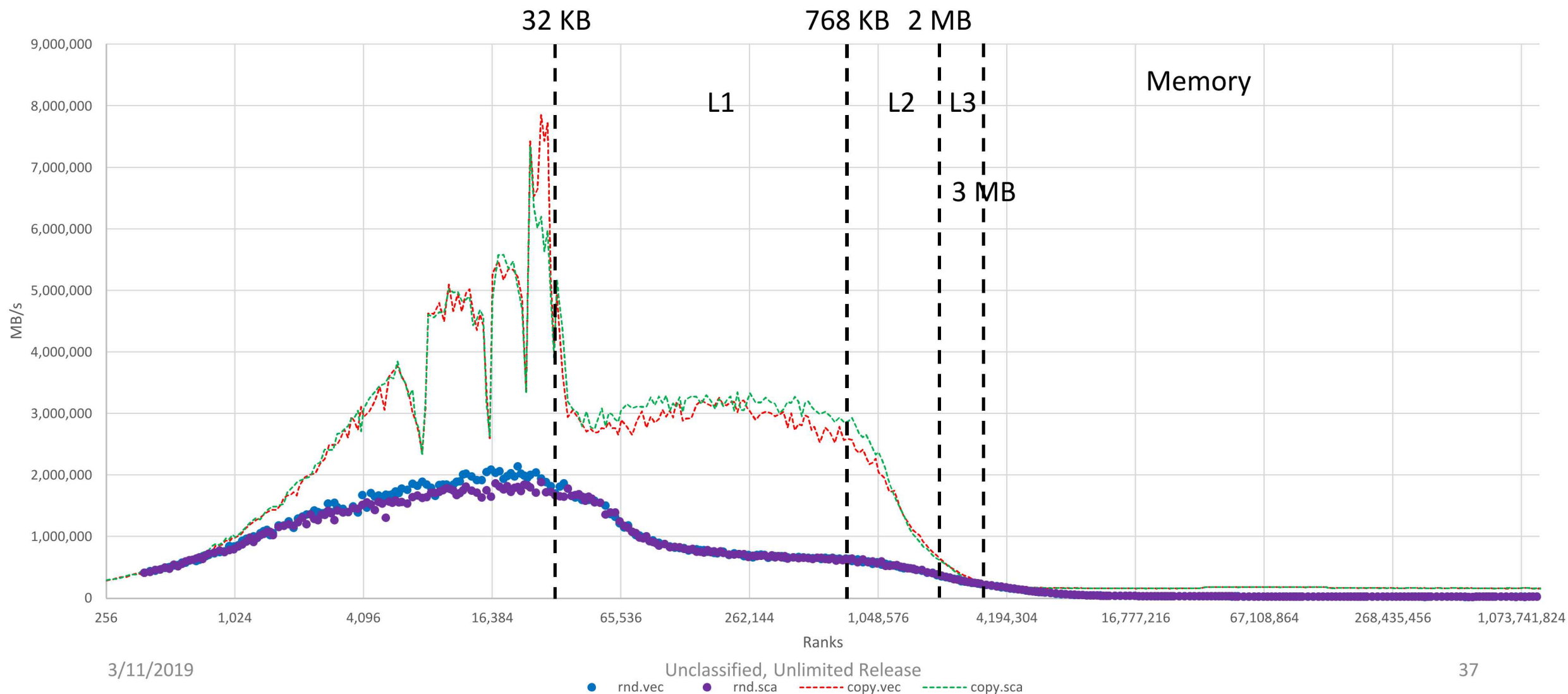
Scatter MB/s (48 Cores)



Sequential Scatter MB/s (48 Cores)

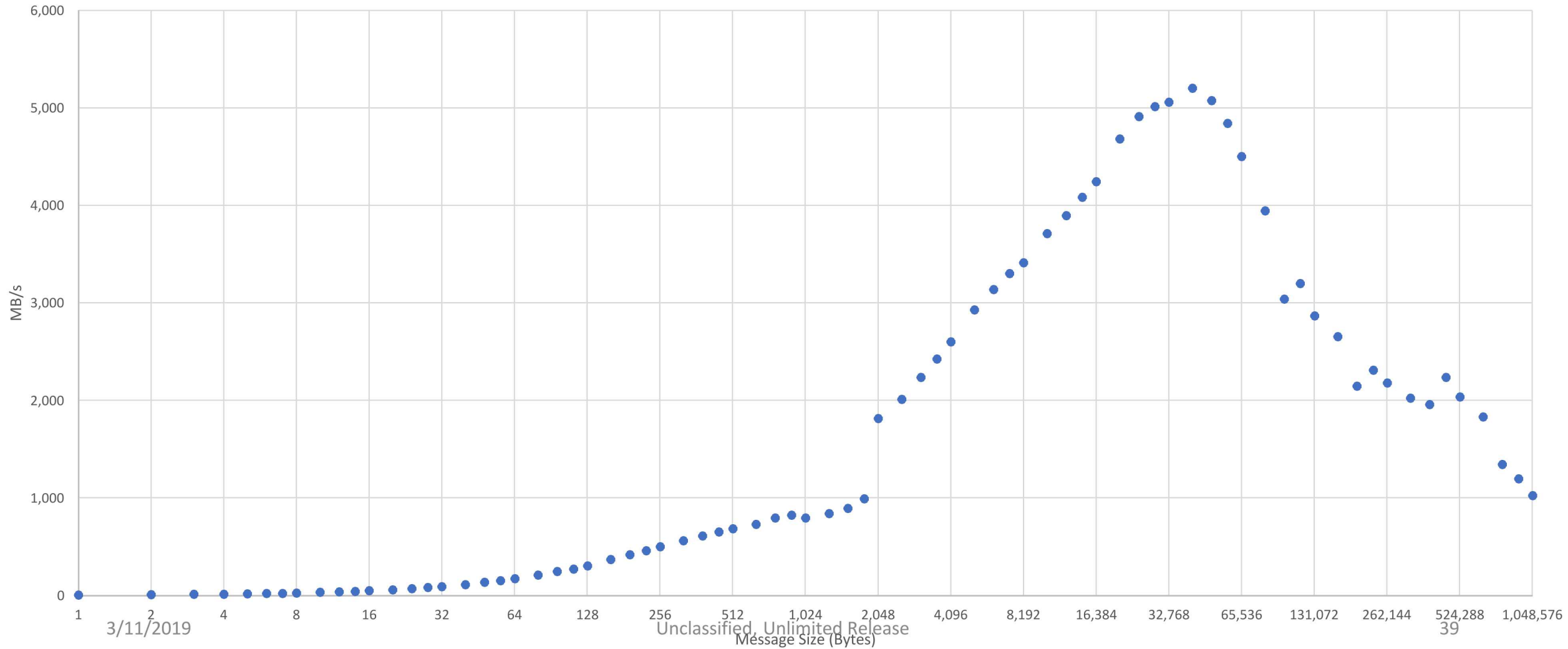


Random Scatter MB/s (48 Cores)

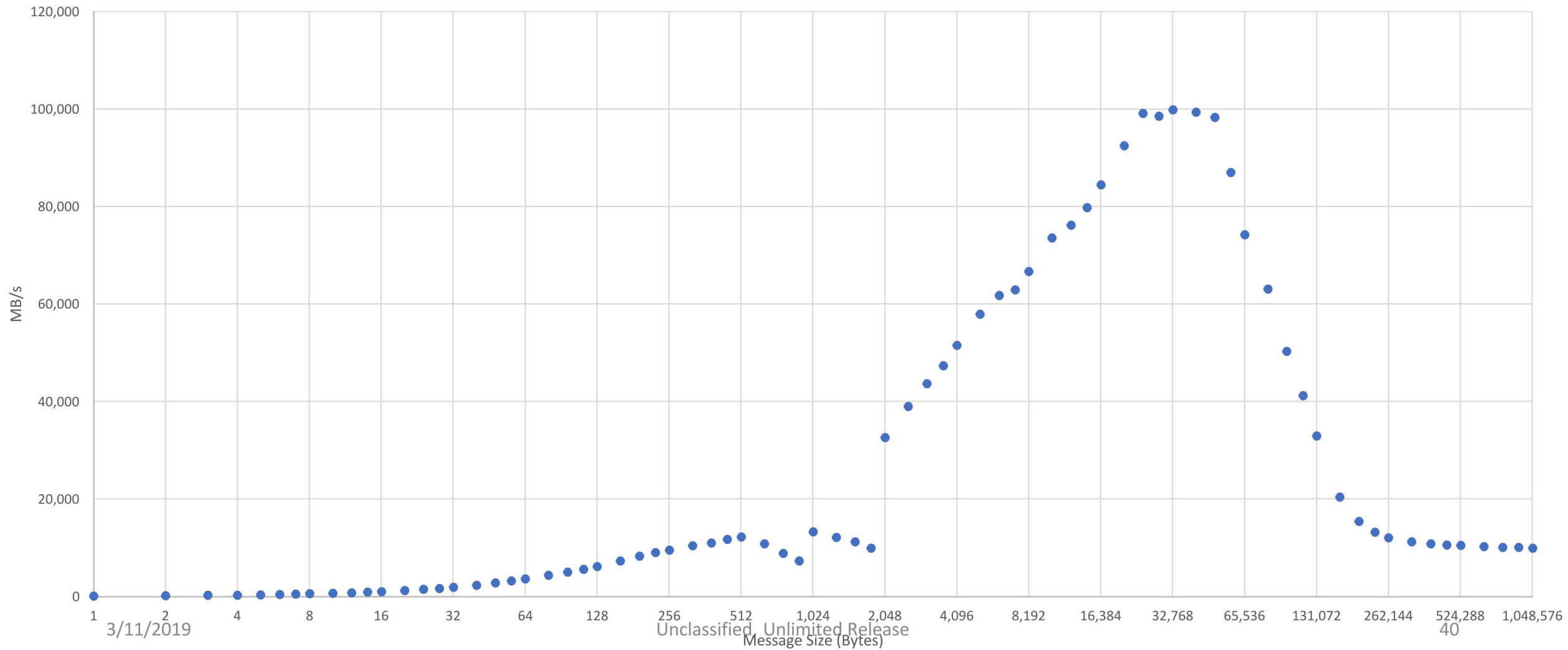


MPI_Sendrecv() &
MPI_Allreduce()

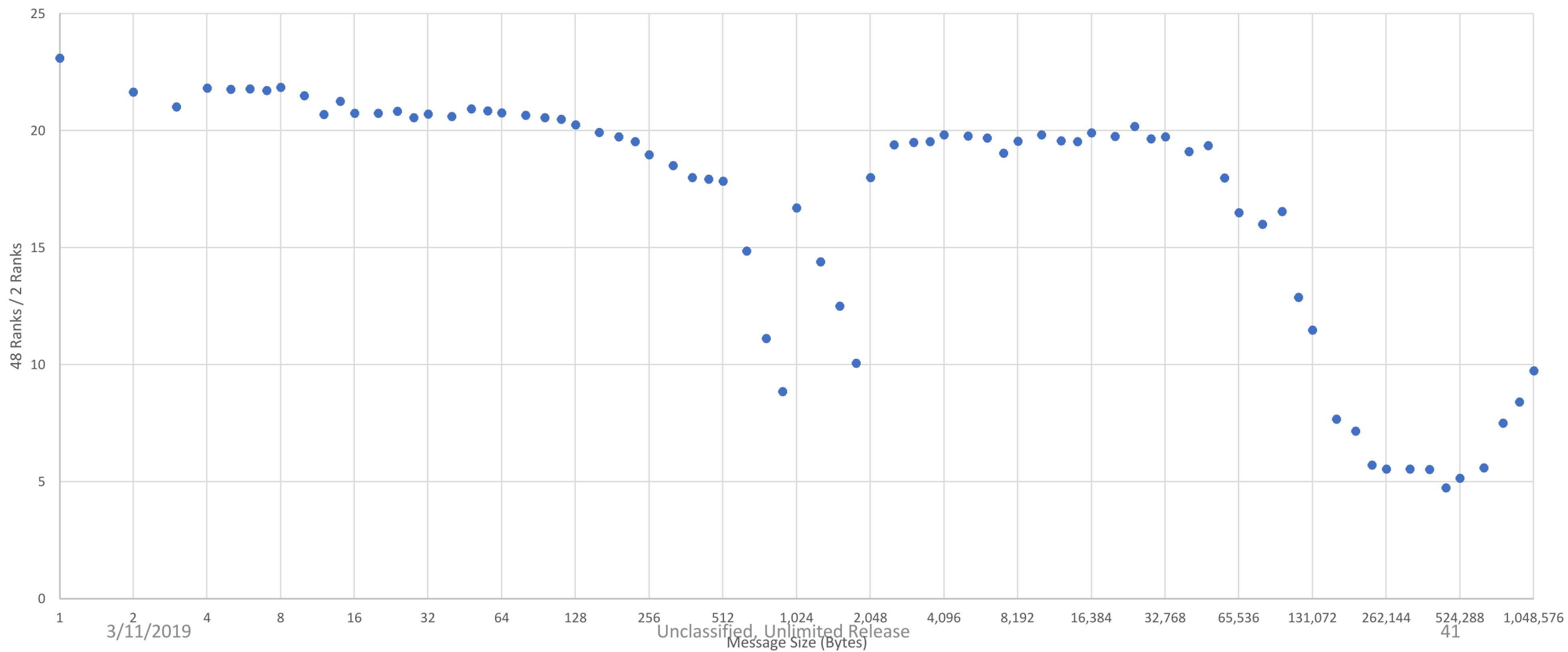
MPI_Sendrecv (1 Node, 2 Ranks)



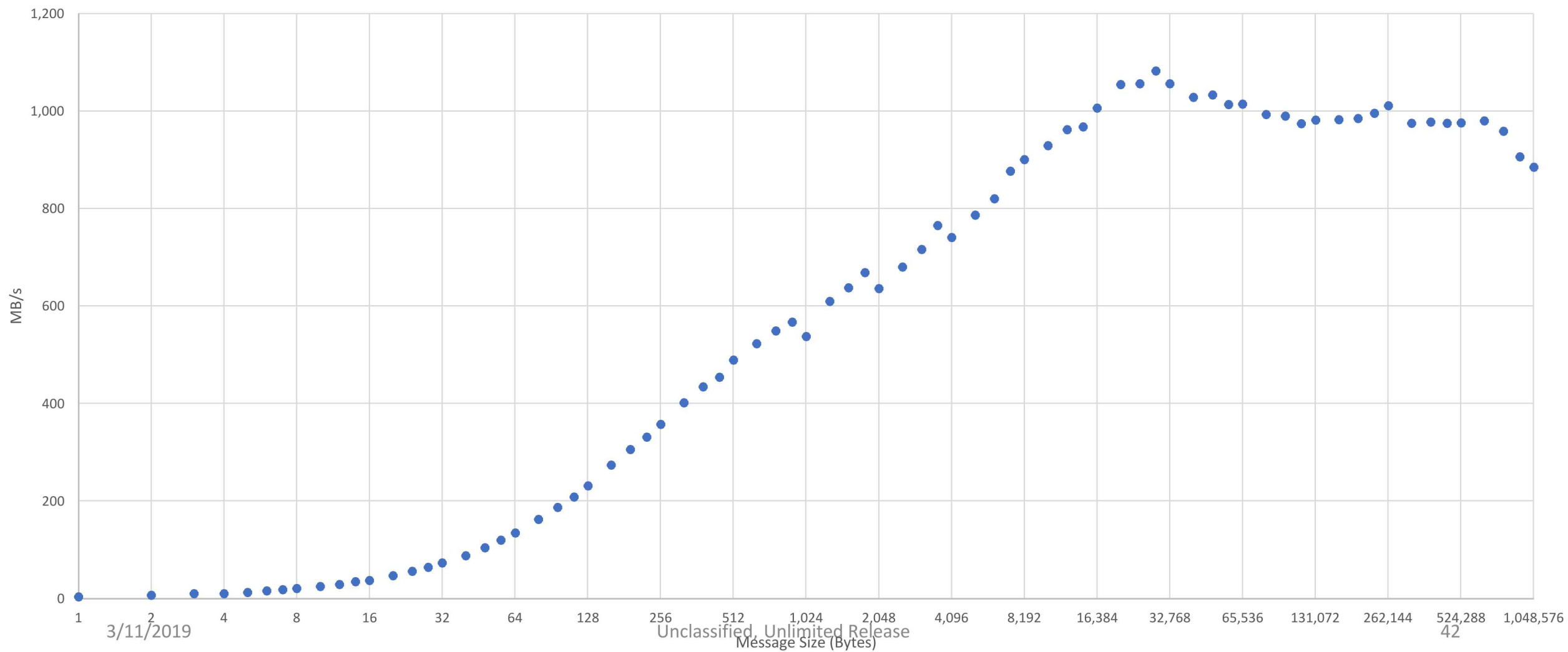
MPI_Sendrecv (1 Node, 48 Ranks)



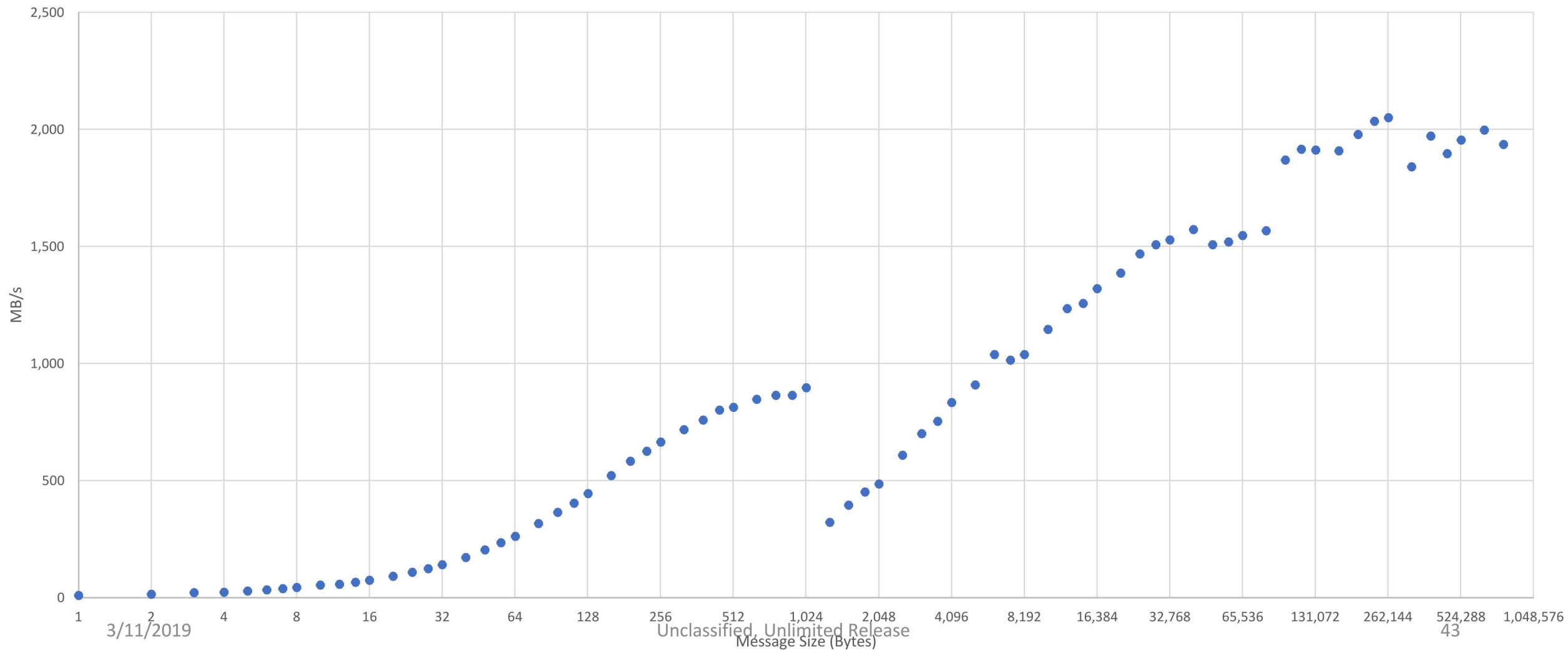
MPI_Sendrecv (48 Ranks / 2 Ranks)



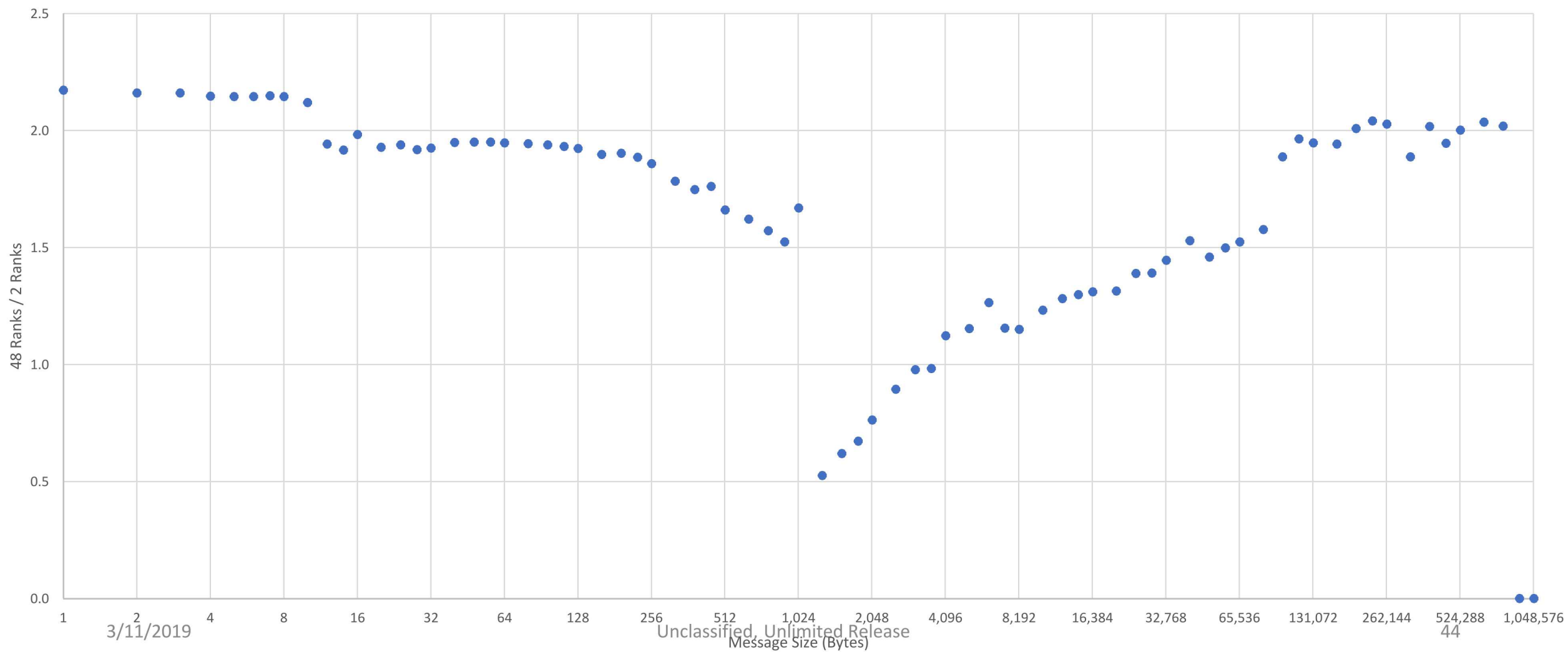
MPI_Allreduce (1 Node, 2 Ranks)



MPI_Allreduce (1 Node, 48 Ranks)



MPI_Allreduce (48 Ranks / 2 Ranks)



Questions?