EPICS Collaboration Meeting, DLS, Spring 2013

Performance and micro benchmarking EPICS V4

Matej Sekoranja

Introduction

COSYLAB

- In EPICS everything is about performance…
- Everybody wants to see performance figures…
- … here they are!
- C++ pvAccess performance figures actually (there is no C++ IOC implementation, yet).

Test setup

COSYLAB

- Server:
  - Intel Core i7-3820 Processor (3.6GHz, 10MB, 4C/8T)
  - 2x 4GB DDR3-1600
  - 1Gbit Intel 82579L(M)
  - Debian Wheezy 64-bit
- Client:
  - MacBook 13” Pro, Intel® Core™ Duo (2.66GHz, 3MB, 2C)
  - 2x 4GB DDR3-1067
  - 1Gbit Broadcom BCM5701
  - Mac OS X 10.6.8
- 1Gbit HP V1410-8G Unmanaged Switch
- SFTP Cat6 cable
- Latest mercurial ‘tip’ EPICSv4 C++ modules, 64k buffers

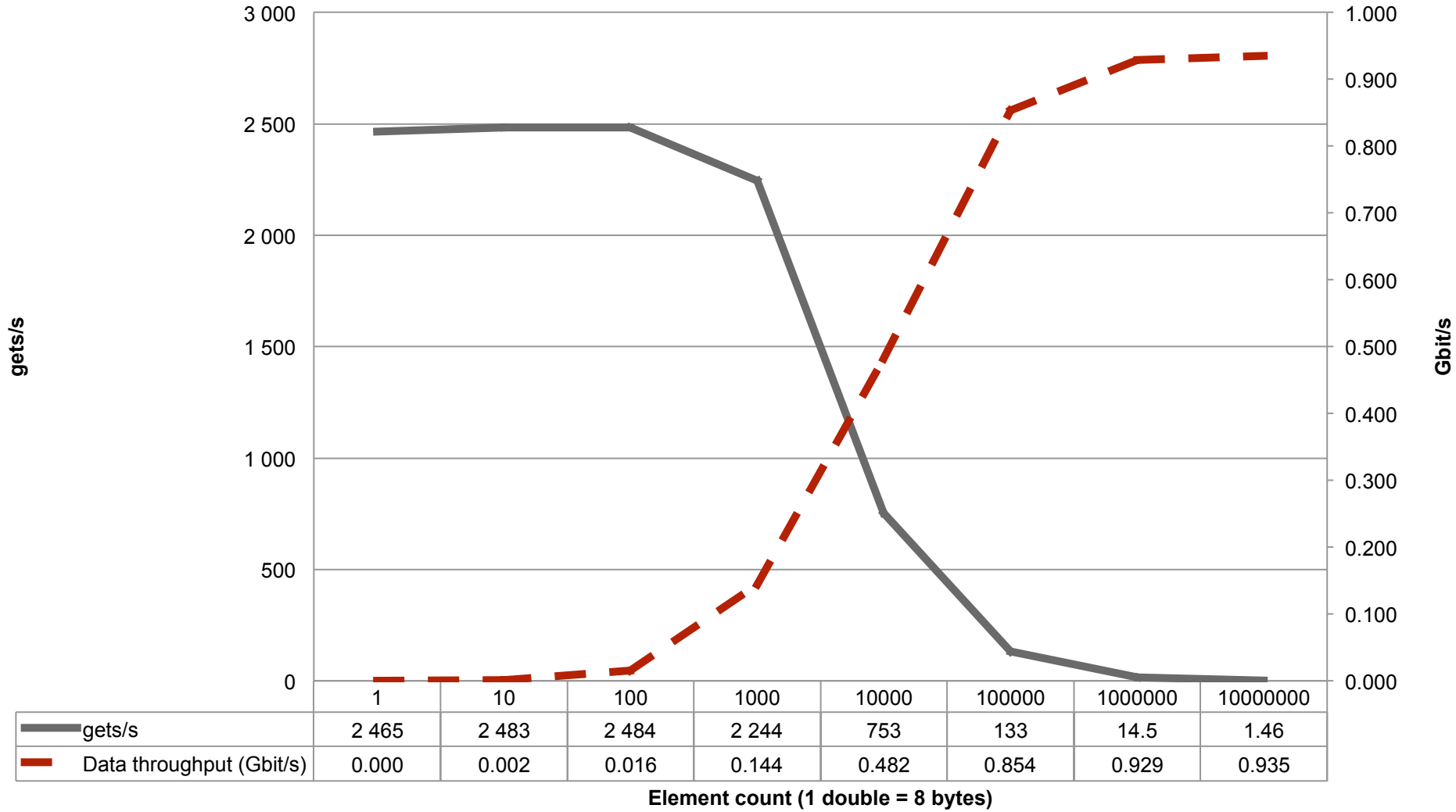
Test setup (cntd.)

COSYLAB

- Channel get operation under the test
  - For each channel issue a get, wait for all responses
  - Double scalar/array
- 1 channel under the test = measures latency
  - Expected limits/bottlenecks:
    - min: TCP/IP ping-pong latency
    - max: 1Gbit ethernet throughtput (- TCP/IP overhead)
- 1000 channels under the test = measures handling of multiple parallel requests

COSYLAB

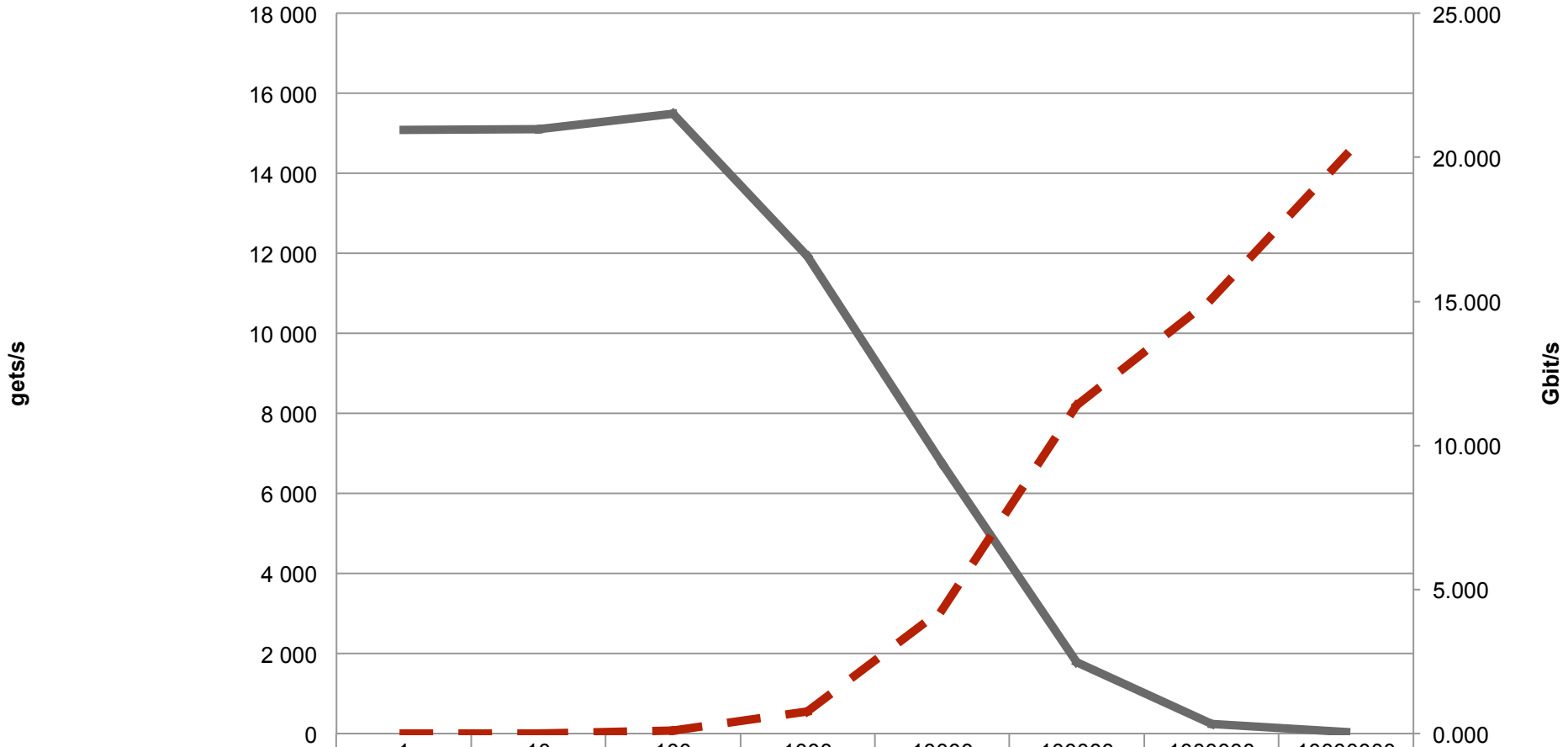
Get performance on double array, one channel



	1	10	100	1000	10000	100000	1000000	10000000
gets/s	2 465	2 483	2 484	2 244	753	133	14.5	1.46
Data throughput (Gbit/s)	0.000	0.002	0.016	0.144	0.482	0.854	0.929	0.935

COSYLAB

Local get performance on double array, one channel

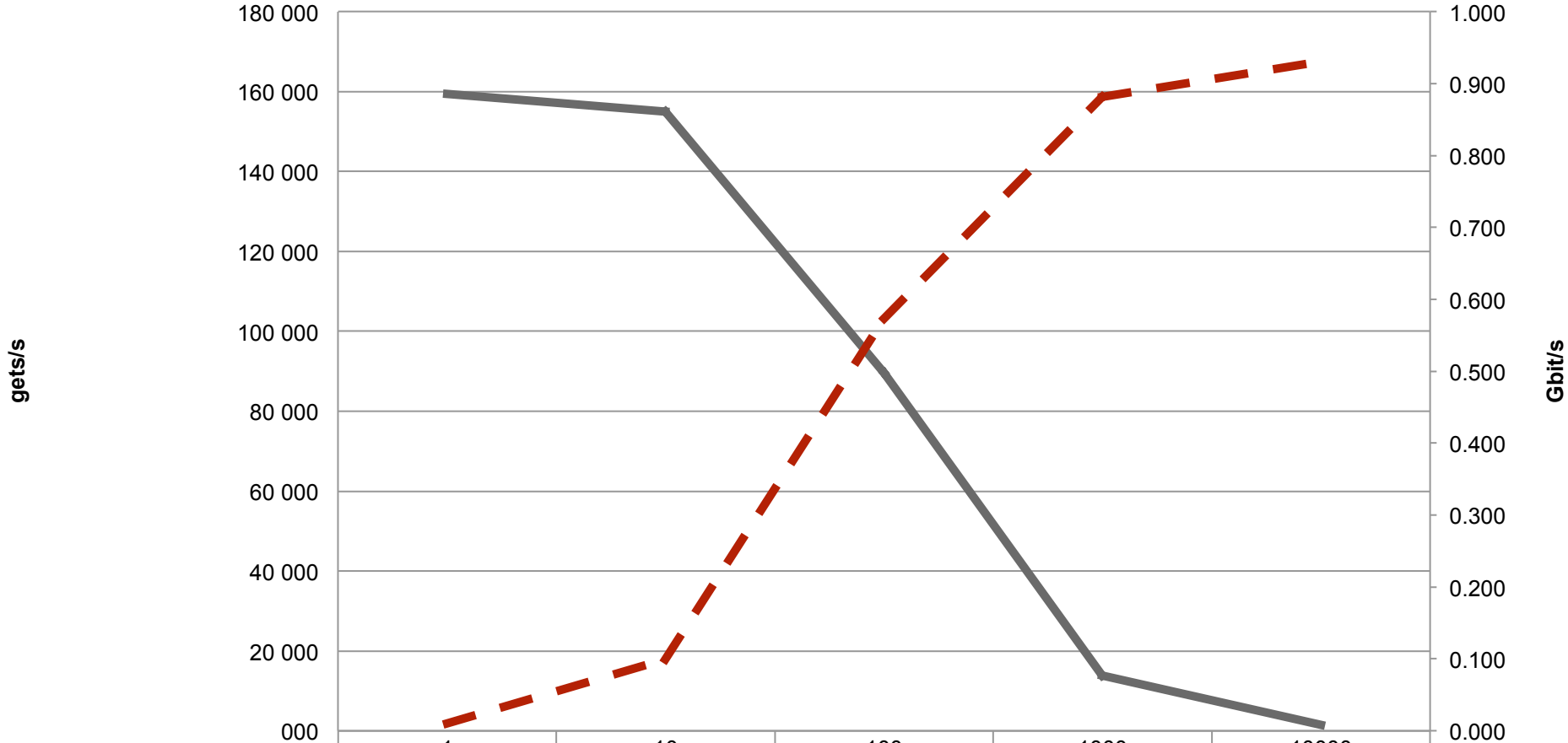


	1	10	100	1000	10000	100000	1000000	10000000
gets/s	15 086	15 103	15 500	11 927	6 744	1 780	235.9	31.38
Data throughput (Gbit/s)	0.001	0.010	0.099	0.763	4.316	11.391	15.100	20.082

Element count (1 double = 8 bytes)

COSYLAB

Get performance on double array, 10000 channels

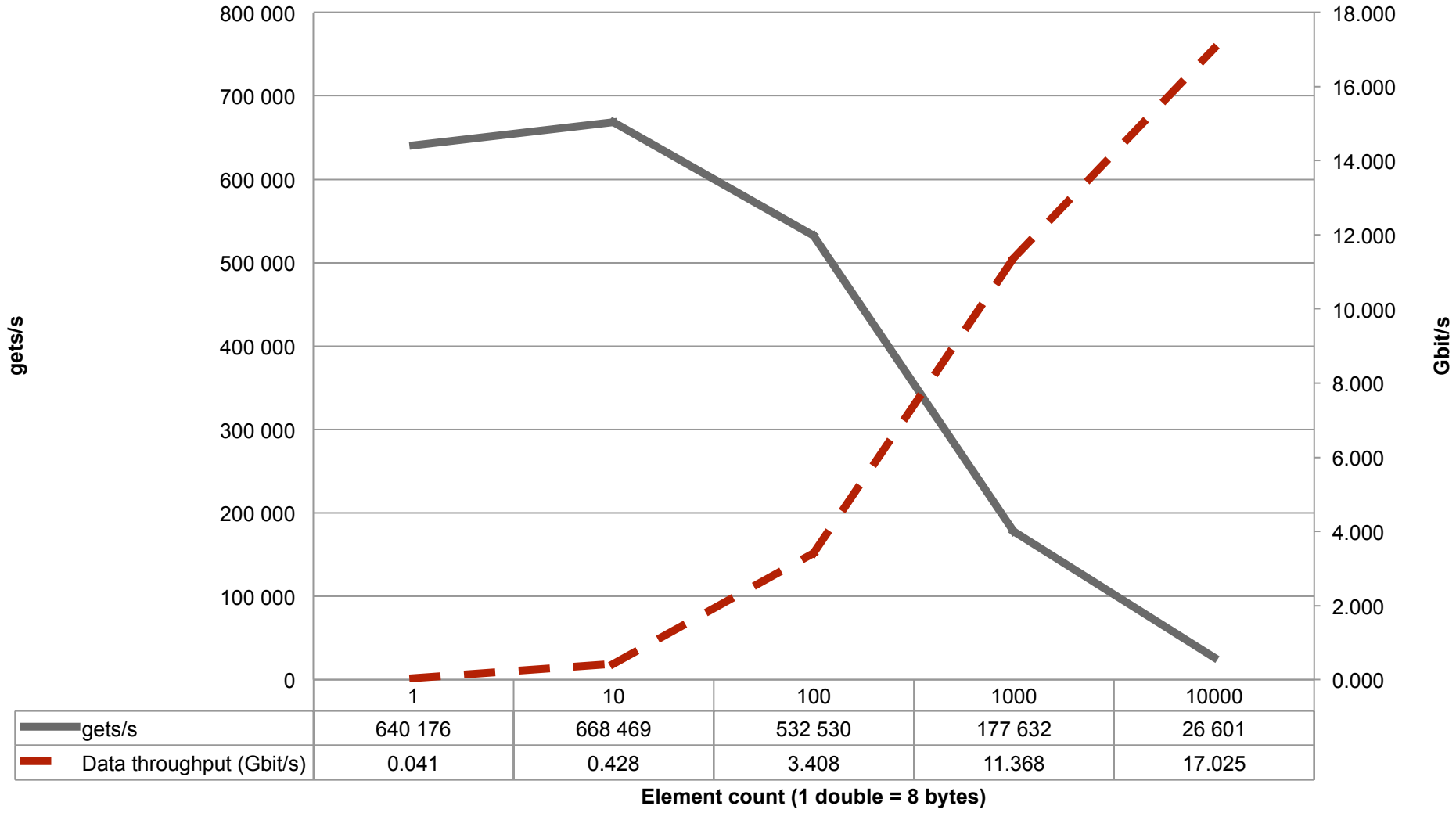


	1	10	100	1000	10000
gets/s	159 497	154 934	89 621	13 780	1 454
Data throughput (Gbit/s)	0.010	0.099	0.574	0.882	0.931

Element count (1 double = 8 bytes)

COSYLAB

Local get performance on double array, 10000 channels



	1	10	100	1000	10000
gets/s	640 176	668 469	532 530	177 632	26 601
Data throughput (Gbit/s)	0.041	0.428	3.408	11.368	17.025

Micro-benchmark of a get

- Developed to measure time spent at the different stages of a processing pipeline
- Works over different threads, processes and conditionally also different hosts (if clocks are in sync)
- Design to help developers to sport performance bottlenecks

Micro-benchmark results

COSYLAB

	mean	stddev	
from channel->get() until the start of serialization	11.71	1.12	us
get request message serialization	0.57	0.20	us
from serialization to deserialization (local TCP/IP socket send)	20.46	2.05	us
get request message deserialization	0.94	0.07	us
the actual get request	0.39	0.17	us
until the start of response serialization	10.34	2.00	us
get response serialization	1.63	0.26	us
from serialization to deserialization (local TCP/IP socket send)	21.63	2.79	us
get response deserialization	1.72	0.37	us
	69.39	**5.90**	us

THANK YOU!

Your TRUSTED Control System Partner

COSYLAB