EPICS Collaboration Meeting, SLAC, Spring 2012

pvAccess Network Protocol

Matej Sekoranja
presented by Marty Kraimer

Introduction

COSYLAB

- pvAccess is a high-performance network communication protocol for signal monitoring and scientific data services interconnect
- Designed to support the structured data types – pvData
- Combines CPU and wire data size considerations to optimize overall control network throughput and minimum latency

Status

COSYLAB

- First full Java and C++ implementations available from Feb 2012 (1.0-BETA release)
- After this Java implementation refactored
  - "codec" based – abstract class that takes care of all the encoding/decoding
  - transport implementations (e.g. TCP, UDP) simply use the codec and contain no protocol specific knowledge
  - extensive JUnit tests implemented to verify codec implementation
- C++ implementation refactoring in progress

Status (cntd.)

COSYLAB

- pvAccess Protocol Specification Second Public Draft released
  - Lots of discussion, expert reviews…
  - Comments, reviews welcome and appreciated
- Last change: unsigned integer support!
- Currently no implementation (Java, C++) fully implement the specification

Performance

COSYLAB

- "Everybody" wants to see some performance figures…
  - A simple test was performed to give a feeling what pvAccess performance capabilities are

- What was tested:
  - C++ implementations tested
  - Only protocols tested (aka their "portable server" implementations)

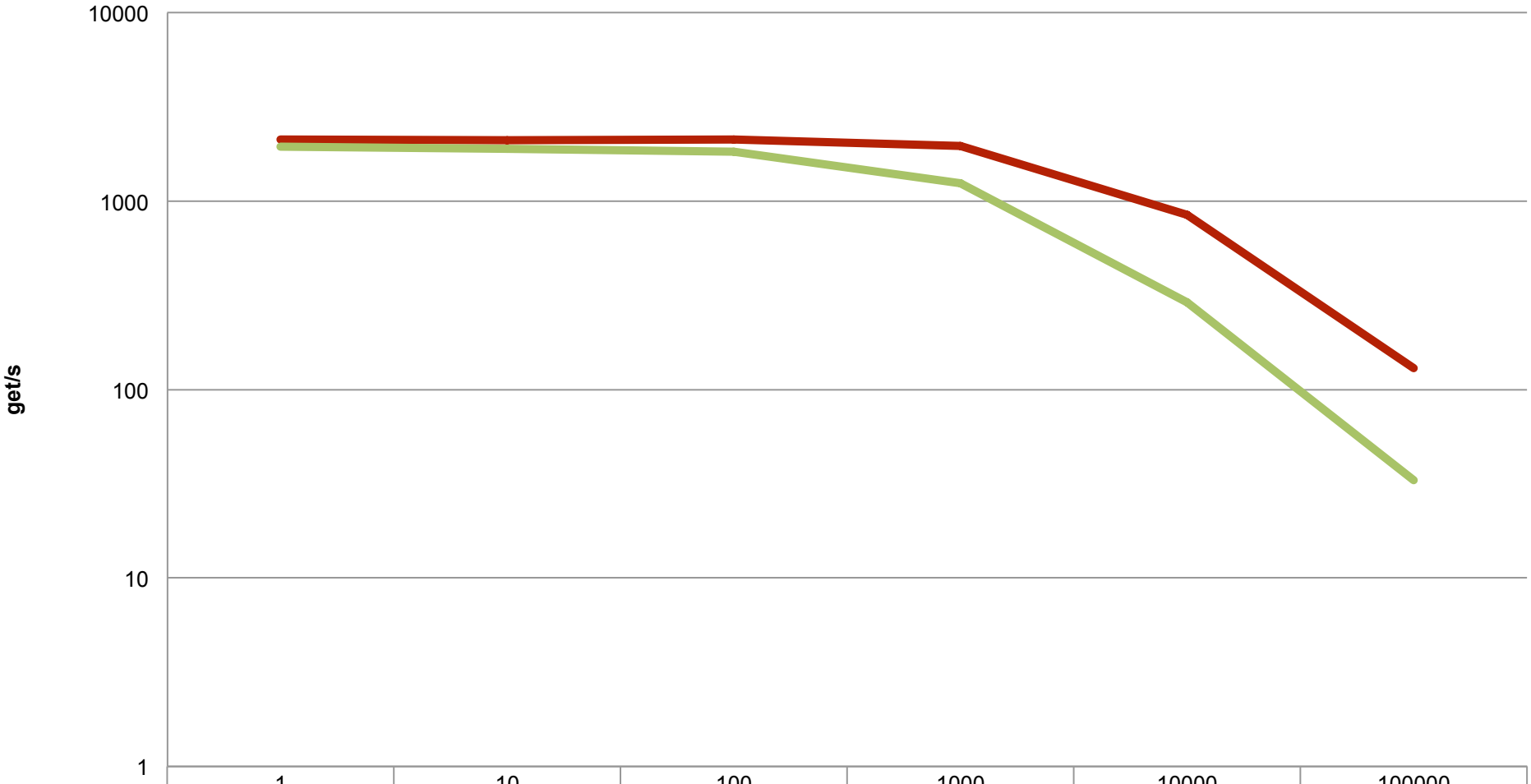
  - pvAccess C++ (1.0.1-BETA)
  - EPICS 3.14.12 Channel Access

Performance (cntd.)

COSYLAB

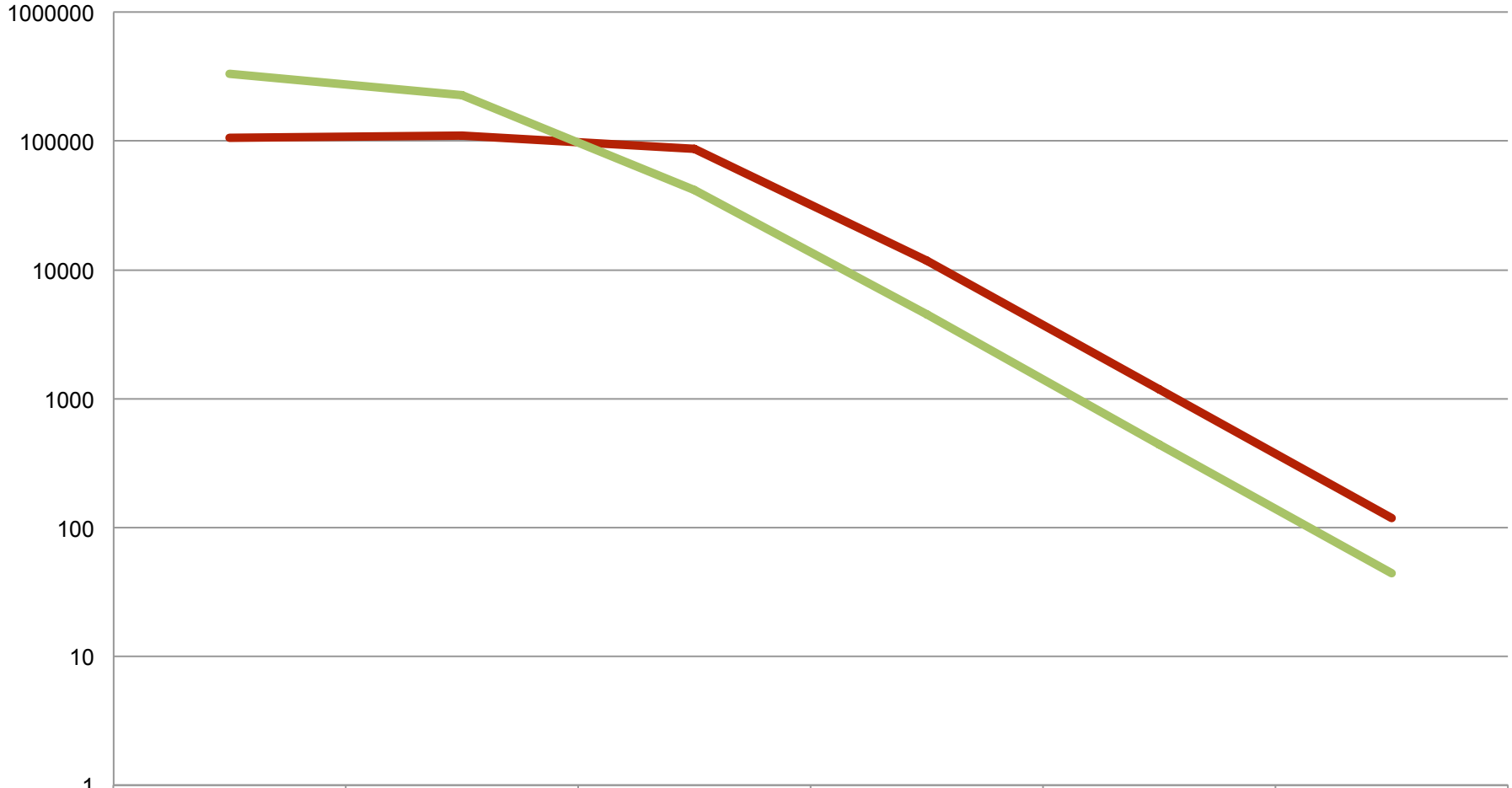
- Clean isolated environment:
  - 2 MacBook Pro's (~2GHz Intel Core Duo-s, 4GB memory)
  - 1GBit Ethernet point-to-point connection (no switch between)
  - Fair test – avoid test-cases where pvAccess can optimize data transfer (i.e. transfer only changed fields in a structure)
  - The test:
    - GET request on a double array value channel (aka DBR_DOUBLE)
    - variable number of channels
    - variable value array size

One channel, double array element count on x axsis



	1	10	100	1000	10000	100000
pvAccess	2120	2110	2120	1960	850	130
CA	1945	1900	1835	1245	291	33

1000 channels, double array element count on x axsis



	1	10	100	1000	10000	100000
pvAccess	106000	111000	86600	11800	1183	118
CA	333000	228000	41570	4510	443	44

Performance (cntd.)

COSYLAB

- pvAccess handles large arrays better
  - it's design allows to send GByte arrays using only 16kB buffers, while CA requires entire array to fit the buffer
- CA handles small messages better
  - pvAccess does not yet implement user-controlled flushing, aka ca_flush_io()
    - API design in progress
  - new codec-based implementation is also expected to provide better performance figures

THANK YOU!