OPENFABRICS ALLIANCE

12th ANNUAL WORKSHOP 2016

PREPARING LHCB EVENT BUILDING AT 4TB/S

Sébastien Valat

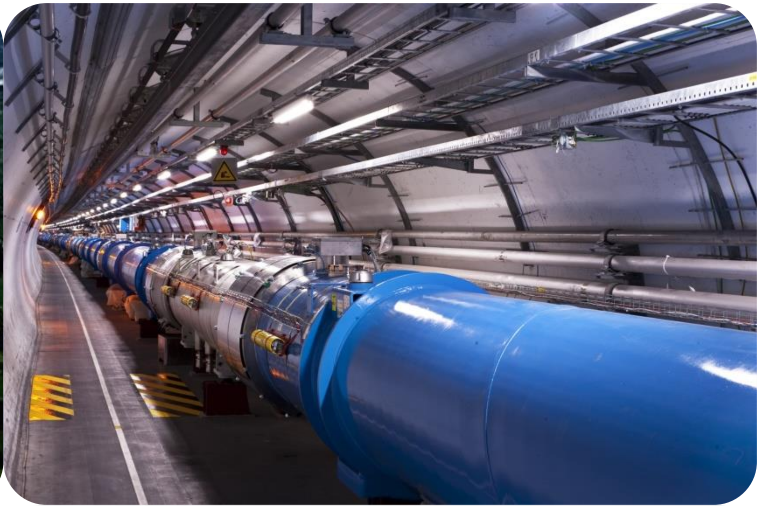
CERN

[April 7th, 2016]

LHCB, THE USE CASE

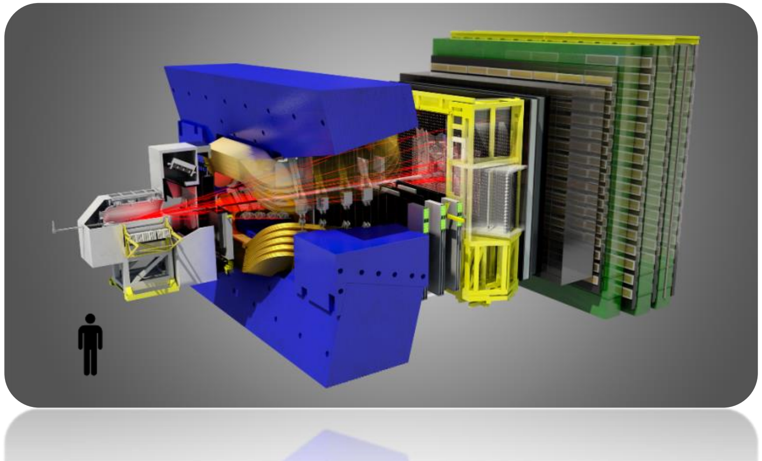
REMINDER ON LHC

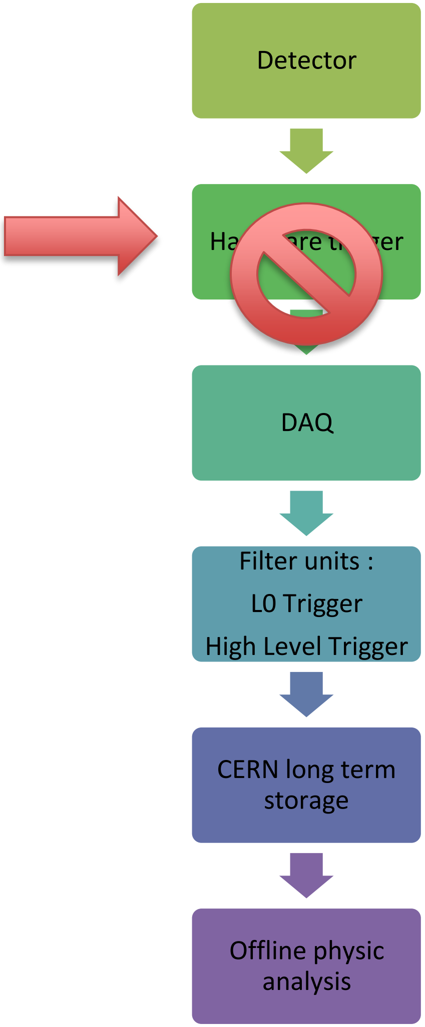
- **Accelerator of 27 km**
- **10 000 superconductive magnets**
- **Collision energy up to 14 TeV**
- **Proton-Proton collisions, but also heavy-ions**
- **4 BIG experiments :**
  - ALICE, ATLAS, CMS, **LHCb**



LHCB, AN UPGRADE FOR 2018-2020

- Update of sub-detectors
- **Removal of hardware trigger**
  - Currently in **custom FPGA**
  - **Hard to maintain** and update
  - In **radiation** area
- **Filter farm will need to handle :**
  - Larger **event rate** (1 Mhz to 40 Mhz)
  - Larger **event size** (50 KB to ~100 KB)
- **Much more data for DAQ & Trigger**
  - It made **4 TB/s (32 Tb/s)**




Detector

Hardware trigger

DAQ

Filter units :

L0 Trigger

High Level Trigger

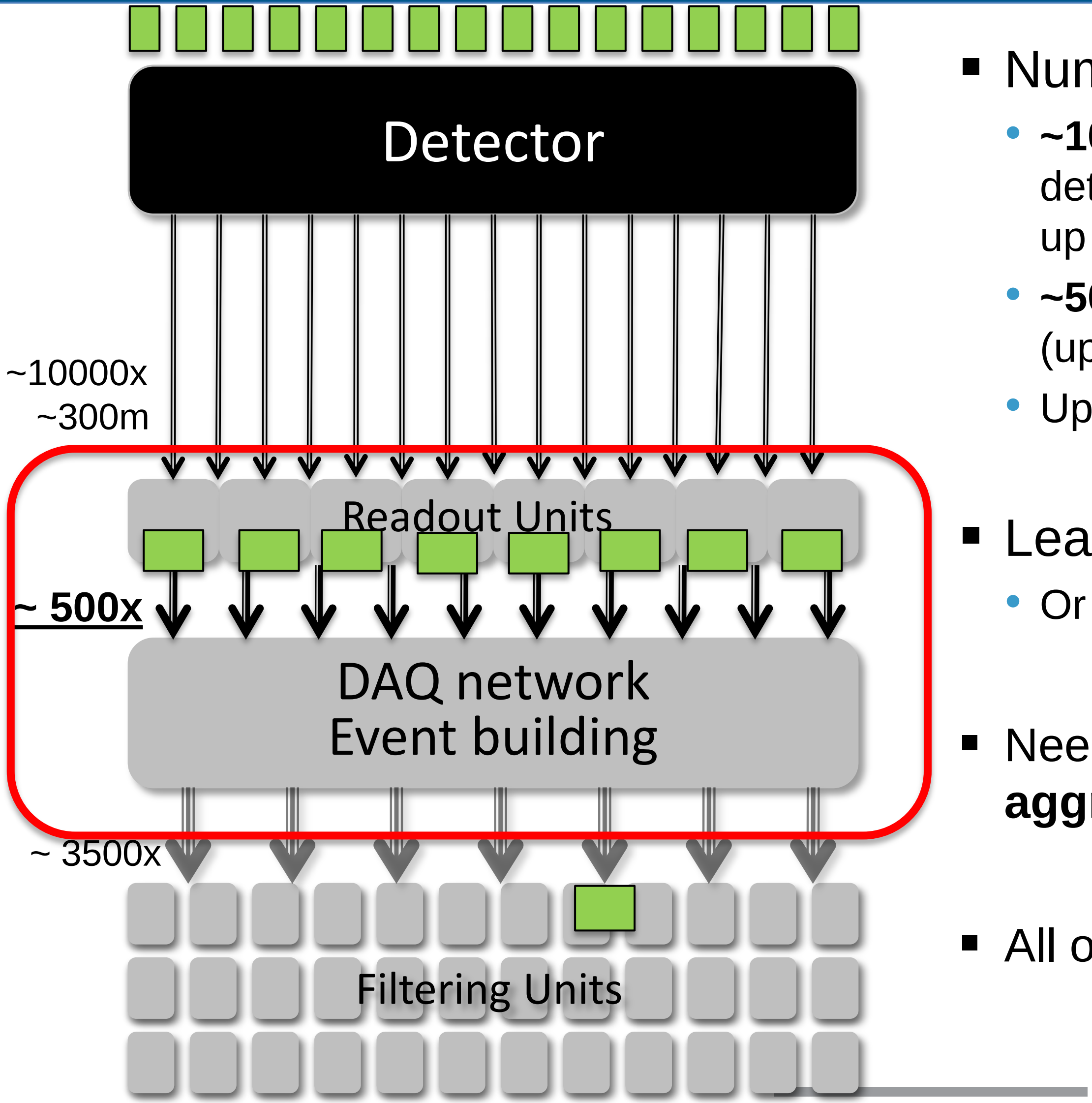
CERN long term storage

Offline physic analysis

WHY TRIGGERING ?

- **We cannot store all of the collisions !**
  - Far too much data !
- **Most collisions produce already well known physics**
- **We keep only interesting events for new physics**
- **Challenge for upgrade: need to trigger in software only**
  - Need to improve current software performance
  - A factor of **100** (hardware + software)
- **For some costly functions**
  - Look at **GPU**
  - Look at possible **CPU embedded FPGA** for some costly functions

DATAFLOW



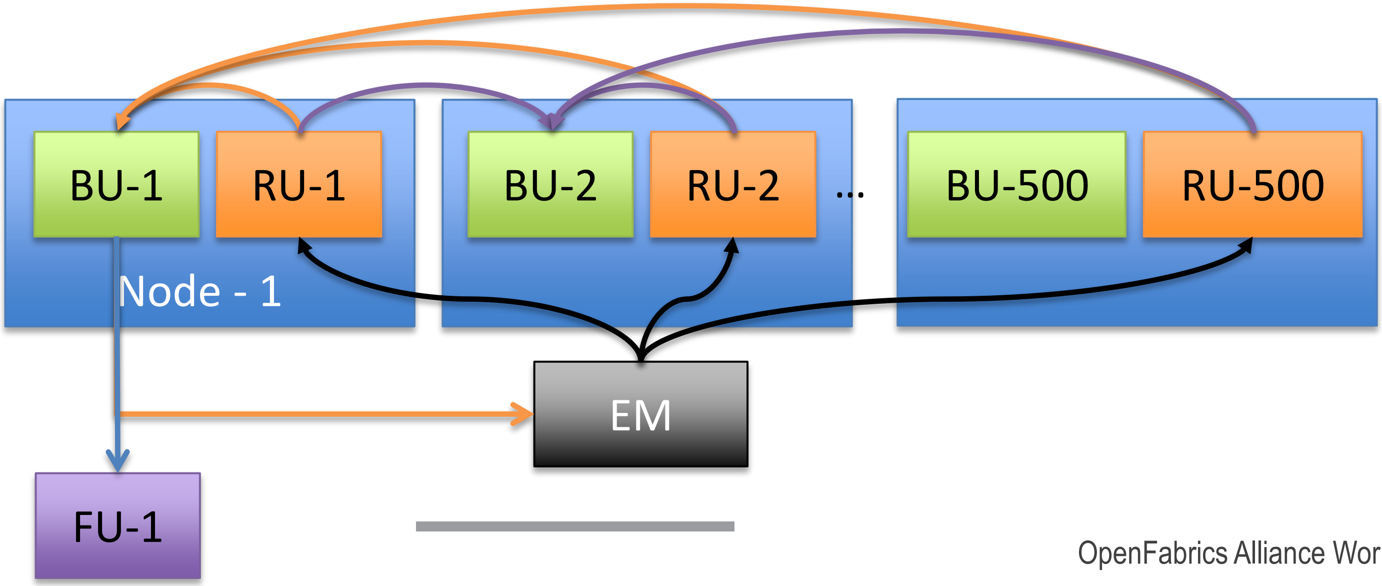
- Numbers
  - **~10 000** optical links going out from detector to the surface (**~300 m)** and up to **~4.5 Gb/s** each.
  - **~500** readout nodes (up to 48 input links each)
  - Up **100 Gb/s** incoming per node
- Lead to a total of **~4 TB/s**
  - Or **32 Tb/s**
- Need a **100 Gb/s** network to **aggregate the data**
- All of this in **real-time**

EVENT BUILDING NETWORK EVALUATION

COMMUNICATION PATTERN

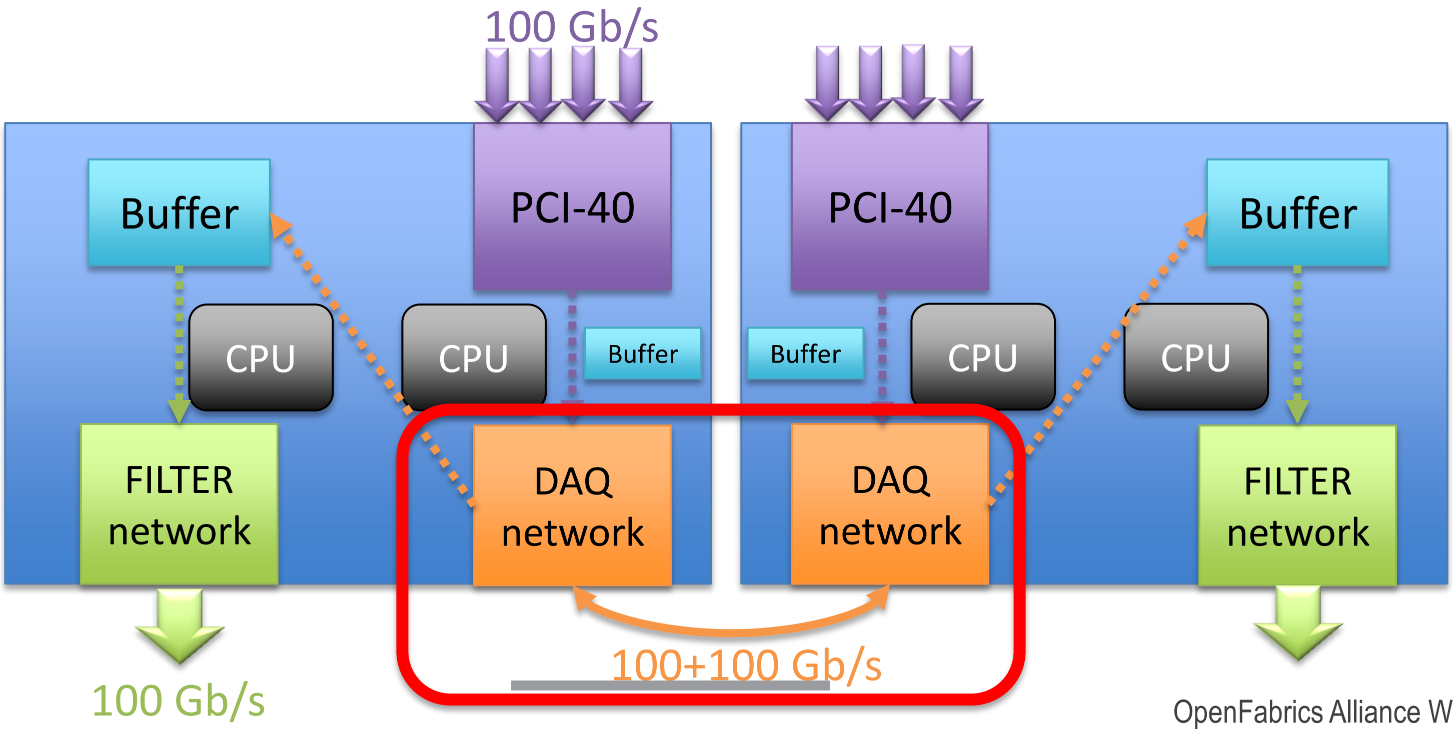
- Work with **4 units:**
  - **Readout unit (RU)** to **read** data from PCI-E readout board
  - **Event manager (EM)** to dispatch the work over **Builder unit** (credits)
  - **Builder unit (BU)** to **merge** data from **Readout unit** and send to **Filter unit.**
  - **Filter Unit (FU)** to **select the interesting** collisions.

- **RU/BU mostly does a kind of "many gather"**
  - To **aggregate** the data chunks from each collision



IO NODES HARDWARE

- **Three IO boards at 100 Gb/s per node:**
  - PCI-40 for fiber input
  - Event building network
  - Output to filter farm
- **Also stress the memory (400 Gb/s of total traffic)**



THE DAQ NETWORK TECHNOLOGIES

- **We need 100 Gb/s per node (RU/BU)**
  - Some margins, 80 Gb/s might be fine

- **Not as HPC apps**
  - We need to **fully fill the network continuously**
  - Bad pattern : many all-gathers (**all-to-all**) !

- **Think of using a fat-tree topology**

- **Technologies we looked on:**
  - **InfiniBand** EDR
  - Intel® **Omni-Path**
  - 100 Gb/s **Ethernet**

INTERFACES TO EXPLOIT THEM

- **MPI**
  - **Support all networks**
  - **Optimized** for **IB/**Intel® Omni-Path
  - How to support **fault tolerance ?**
    - We need to run 24h/24h and 7d/7

- **InfiniBand VERBS**
  - For IB only
  - Low level
  - **Might** be **OK** for **fault tolerance**

- **Libfabric**
  - For IB, Intel® Omni-Path, and TCP
  - Low level
  - Node **failure** and **recovery** support **to be studied**.

- **TCP/UPD** (we don’t depend on latency)

DAQPIPE

- **A benchmark to evaluate event building solutions**

- **Three message size on the network**
  - Command : **~64 B**
  - Meta-data : **~10 KB**
  - Data : **~1 MB**

- **Manage communication scheduling models**
  - **Barrel shift** ordering (with N on-fly)
  - **Random** ordering (with N on-fly)
  - Send **all in one go**

- **Support various APIs**
  - MPI
  - TCP
  - Libfabric

FIRST FEELING WITH LIBFABRIC AS USER

- **Lack of some simple (one file) example**
  - 🙂 Fabtest fine
  - 😐 but codes split in **multiple functions** and **handle all cases**
    - Good: we get **something full** to run
    - Less good: it **took time to dig in** to learn

- ~~**Lack of beginner guide** 😐~~ **=>** 🙂 (I started on 1.1.0rc2)
  - => Thanks to Jianxin Xiong

- **Easy to develop codes thanks to TCP support**
  - 🙂🙂 **Write using TCP**, then test over VERBS or **PSM**

ONCE GOING THROUGH THE INIT STEP

- **Quite quickly to get running**
  - Using fi_send / fi_remote_write

- **I come from OS/HPC memory management PhD.**
  - So: not already an expert on fabrics
  - Ok, I was in a thread-based MPI researching group
  - I did not previously know the app

- **Some rounded numbers**
  - **~1-2 week** of **sandbox** playing
  - **~1-2 week** to get **init** running in my app
  - **~3-4 days** to get **communications in place**
  - +X days of debugging (I also changed some other stuff)…..

DUAL SUPPORT MSG / RDM

- 🙂 **Yes it is managed by the same library**
  - But different semantic.
- ☹ **In practice need to duplicate a big part of the code**
  - At least the init
- 🙂 **But also if statements for the communications**
  - **Address vector** VS. **endpoints**
  - Different **tagging approach** (4 bytes vs 8 bytes)
- 🙂 **Get some issues to see my IB board in fi_info**
  - That’s my fault but (~3 days)…
  - Not sure to completely understand the filtering mechanism
- 🙂 **That’s fine, but I naively expected less diff**

MY LIBFABRIC USAGE

- **Use RDM (OPA) of MSG (IB) protocol**

- **Using fi_send/fi_recv**
  - For **command fixed size** channel
  - Pre-post N **(~6)** recv buffers
  - Re-post immediately on receive.

- **Using fi_remote_write**
  - For **meta-data/data** exchanges
  - Remote **key/addr sent via command** channel
  - Using **tag** to **match** and **notify** received messages

- **Using only one thread**
  - I'm not sure to completely understand how to use more threads (but I didn't tried yet).

ABOUT FAILURE RECOVERY OPEN QUESTIONS (TO ME)

- **With libfabric**
  - **How** are we **notified** of a **node failure** ?
  - **Can** we **re-setup** the **connection** ?
  - Mostly an open question for RDM PSM mode ?

- **Can we lose messages ?**
  - In theory we don’t need to care at our level

- **Internal software side status**
  - We need to **stop pending** connections
  - **Update internal status** to pursue
  - Be able to **re-register the failed node**

LIBFABRIC & LAUNCHER

- **We need to share addresses**

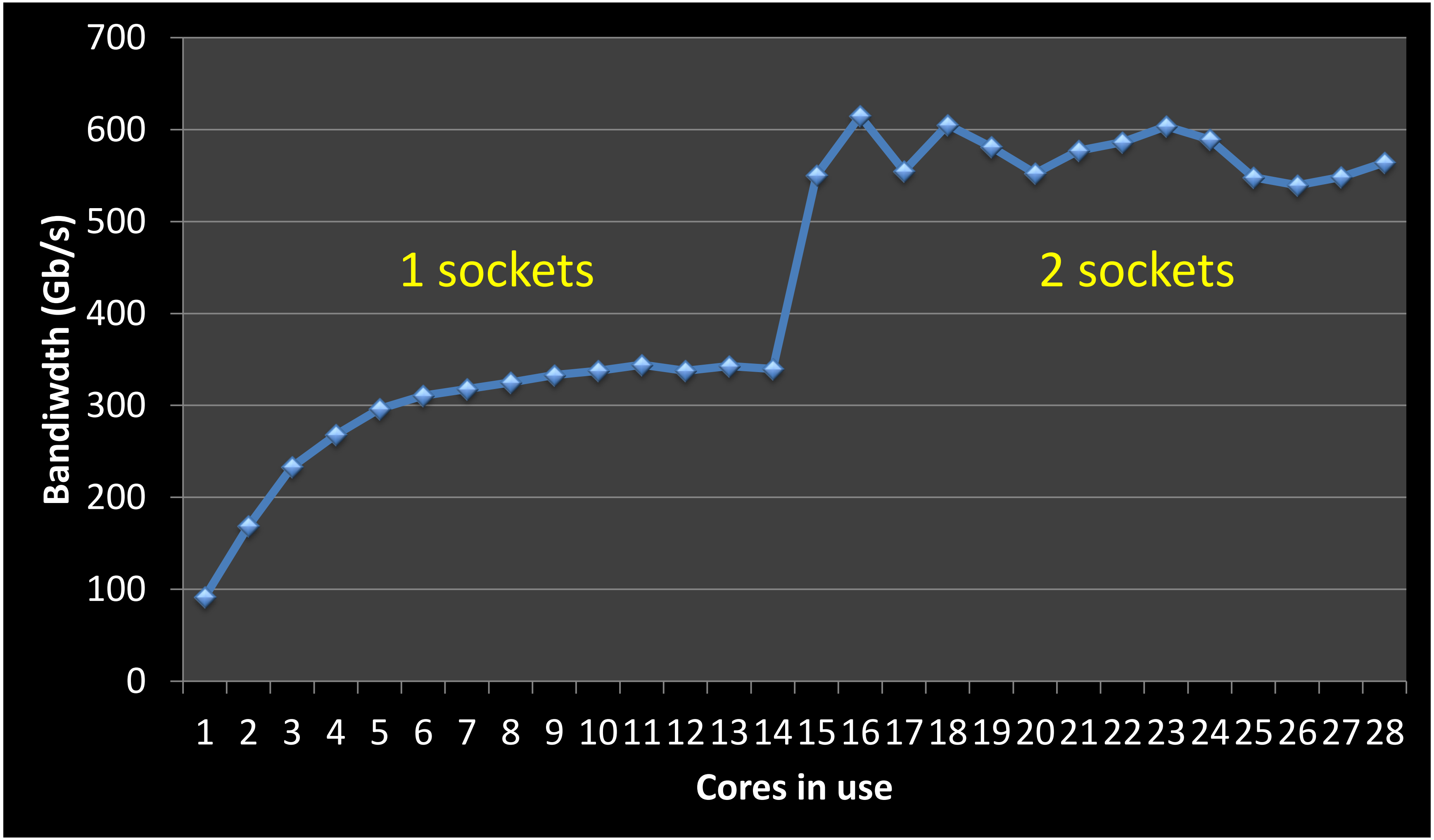
- **I use mpich hydra launcher**
  - **MPI** like **launching**
  - Support most supercomputer’s **job manager**
  - Maybe issues for **node failure recovery, to be checked**.

- **It is not so much code**
  - Could be interesting **to point** it **in libfabric** doc/examples (fabtest ?)

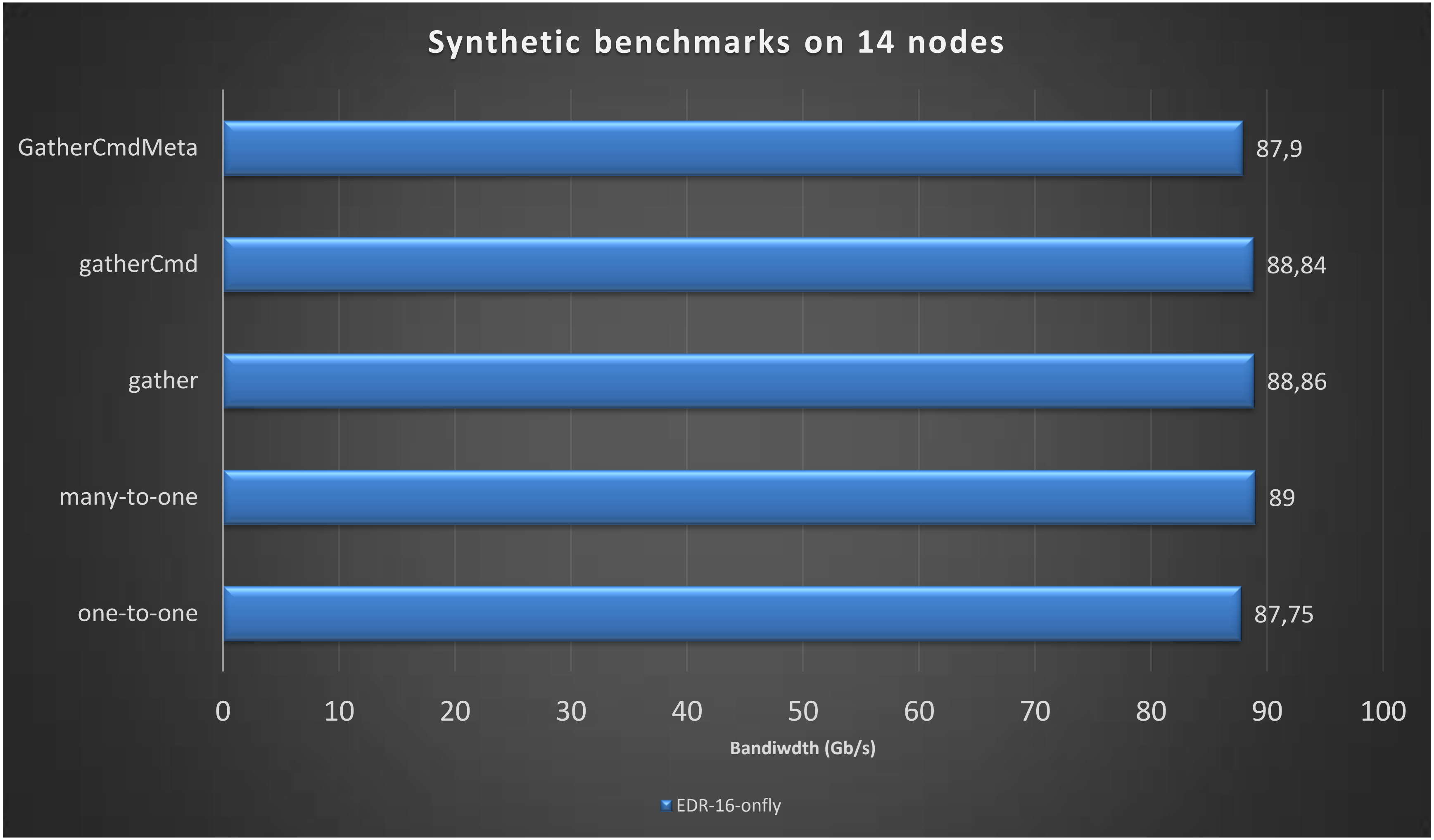
- **Missing:**
  - Some interface files are **missing in hydra package**
  - Need to extract the **hdyra-simple** part **from mpich**
  - Or missing doc to use it ?
  - It’s using **V1 API**
  - If I understand mpich use **V2 API**, how to use it **outside of mpich** ?

EXPERIMENTAL RESULTS

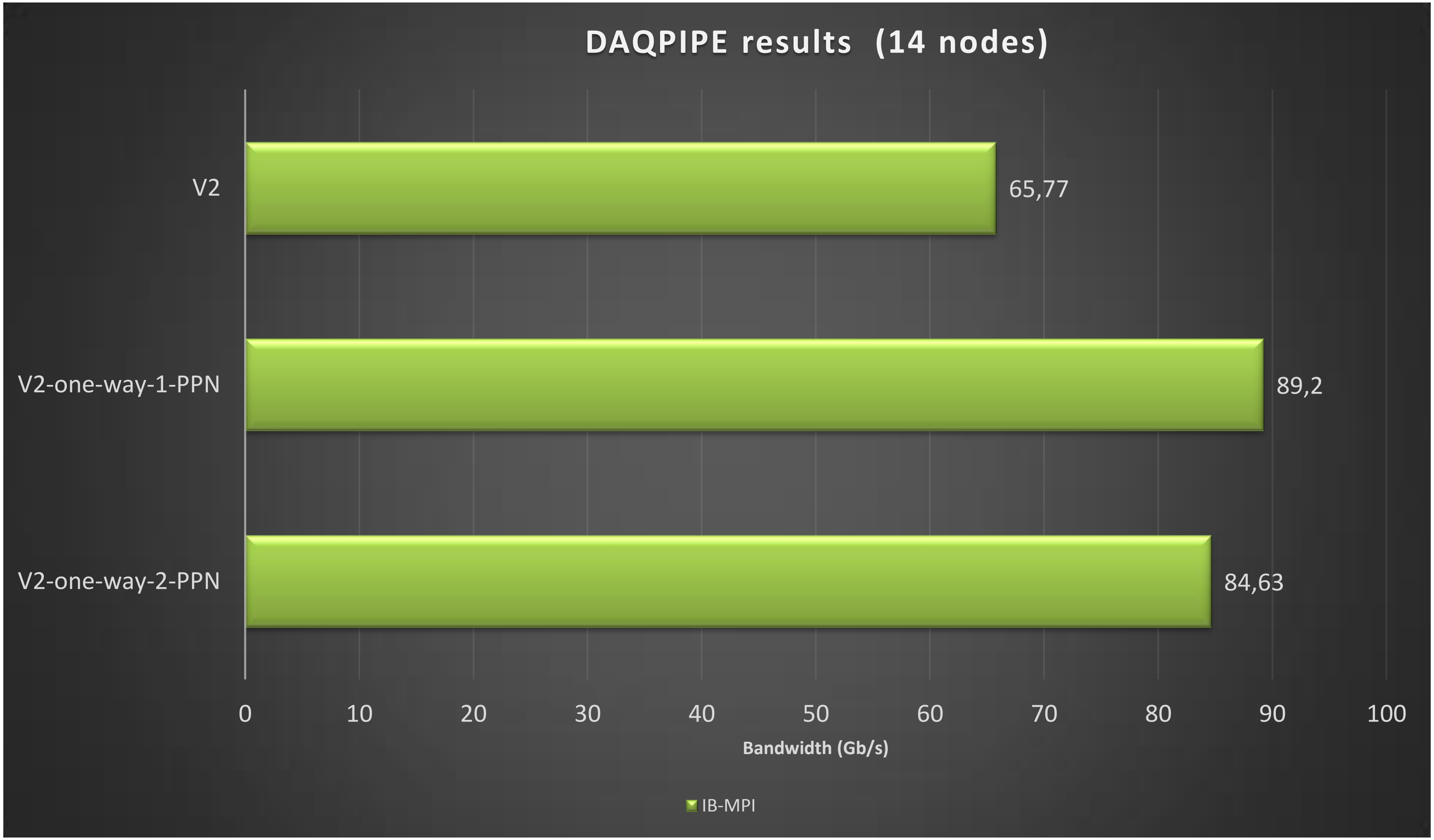
CPU MEMORY BANDWIDTH
STREAM BENCHMARK ON BI-XEON E5-2690



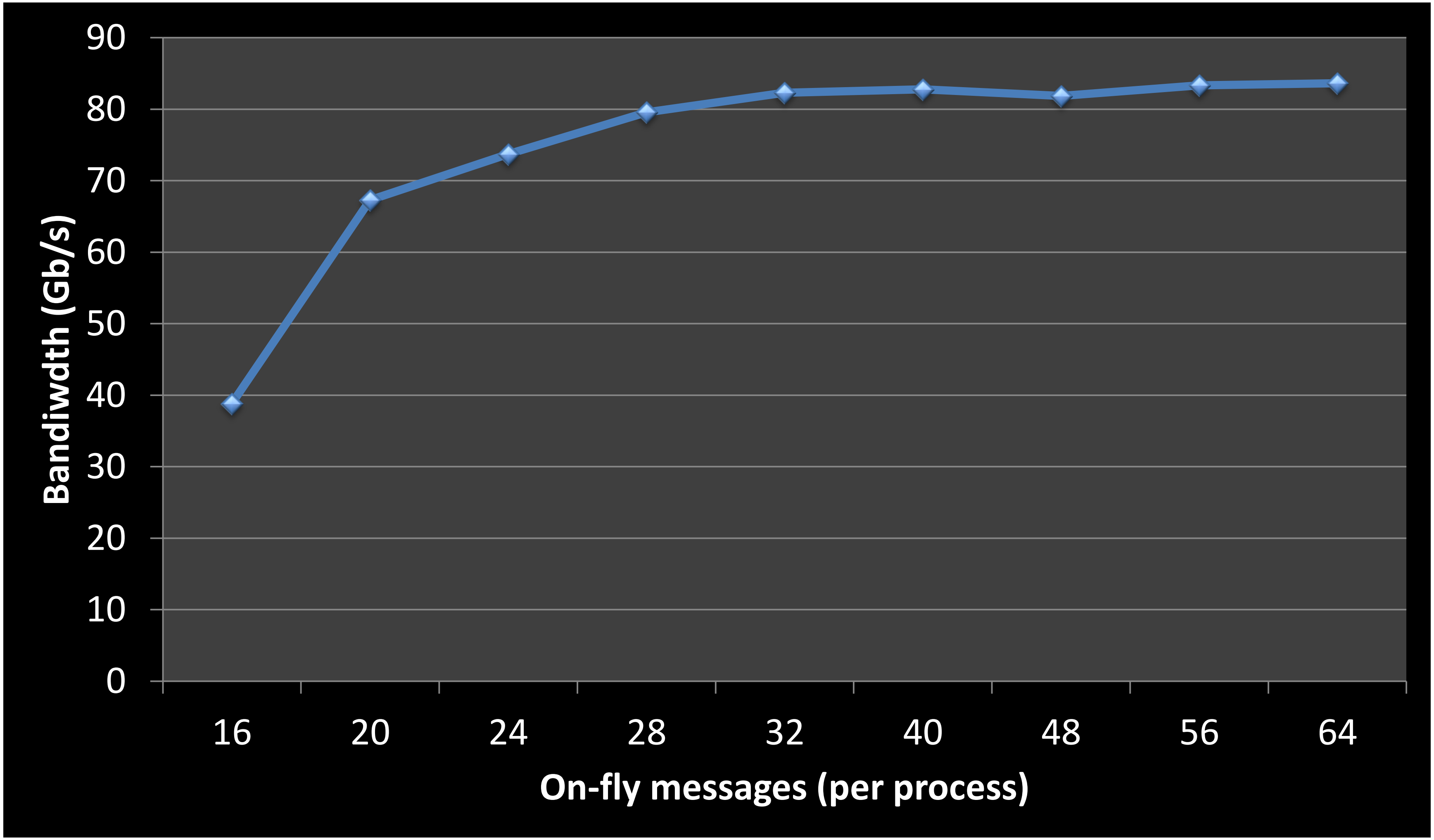
SIMPLE BENCHMARKS OVER IB-EDR



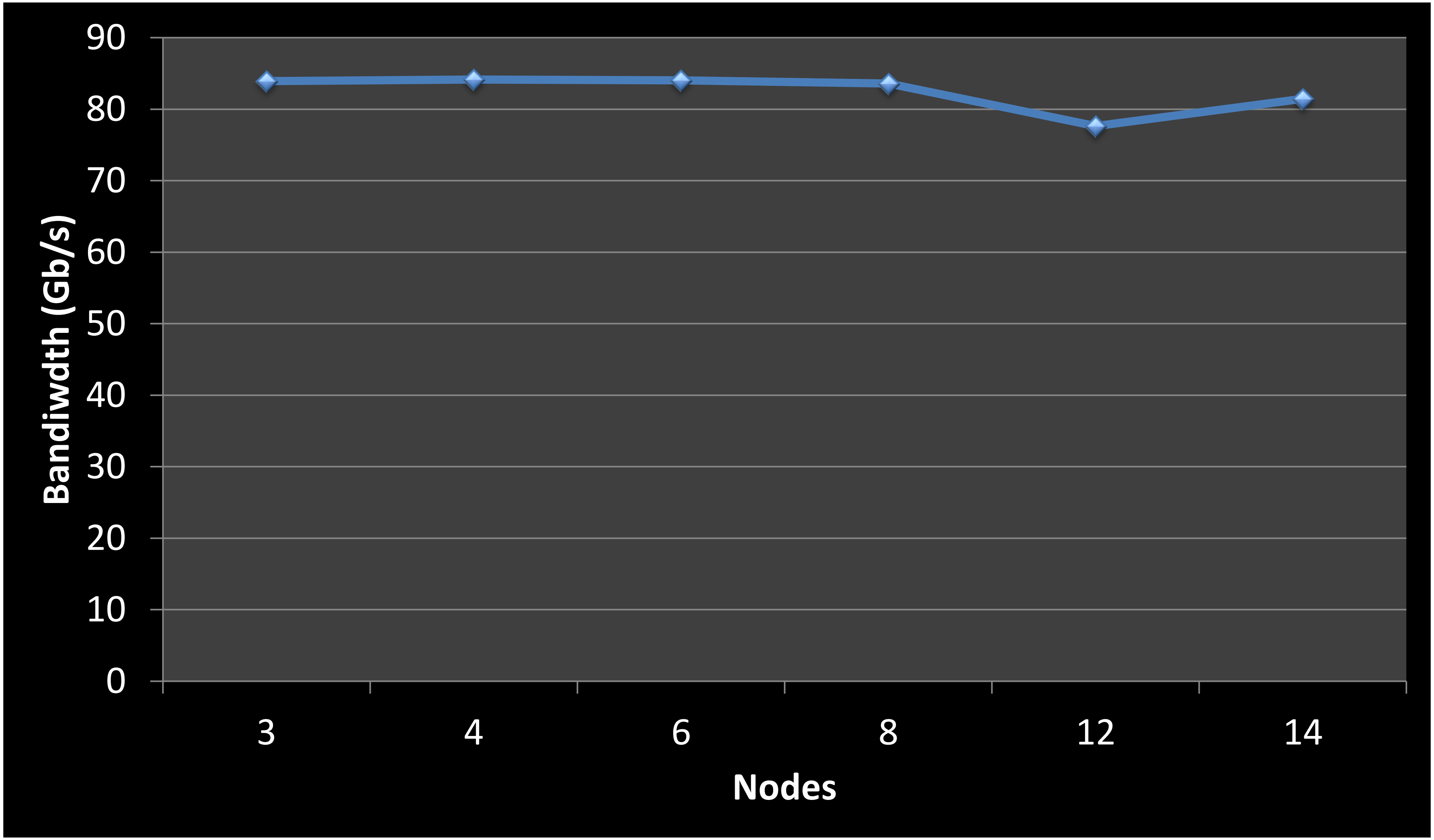
DAQPIPE OVER IB



NUMBER OF ON-FLY MESSAGES

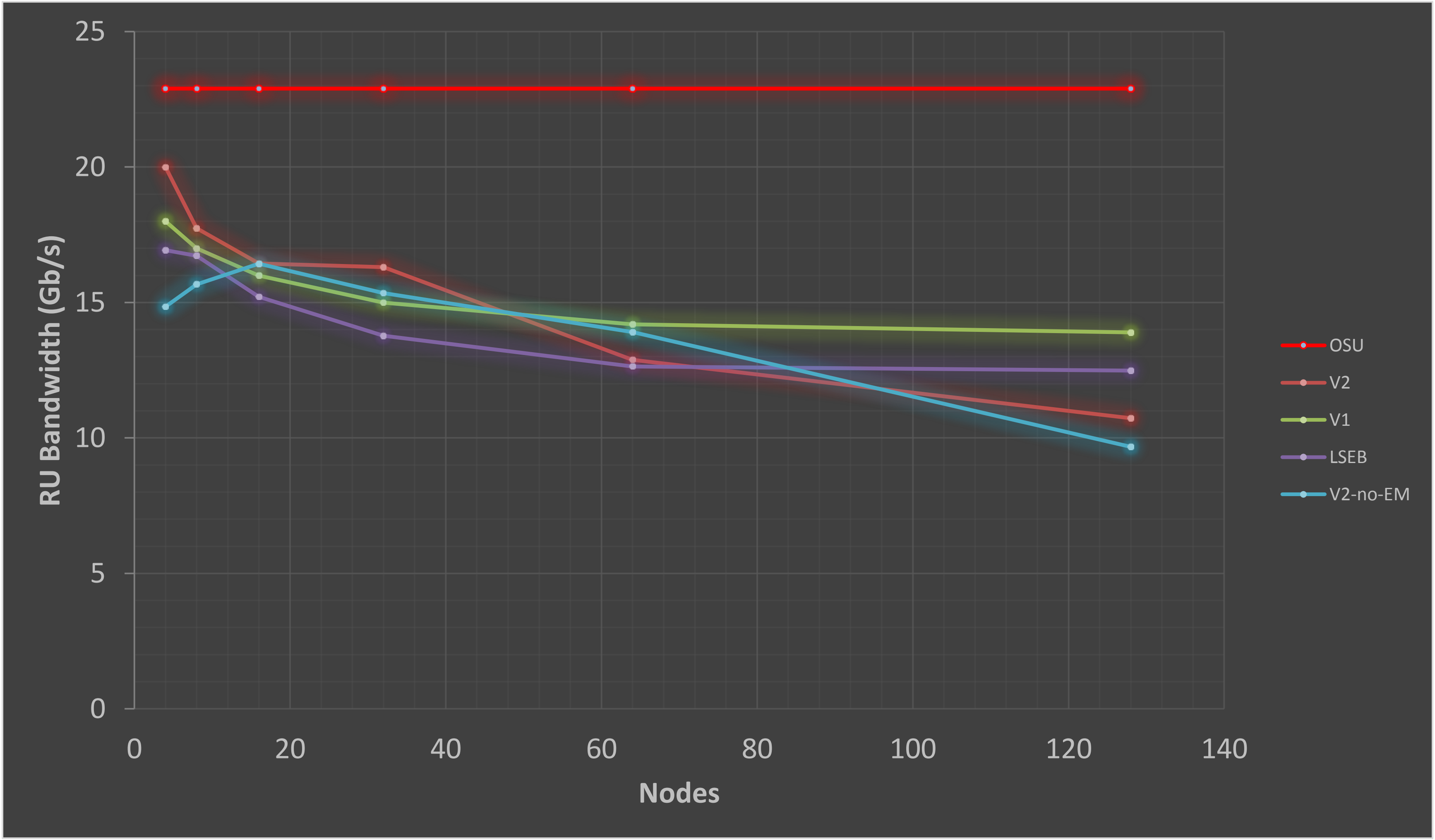


SCALABILITY ON EDR



SCALABILITY ON QDR

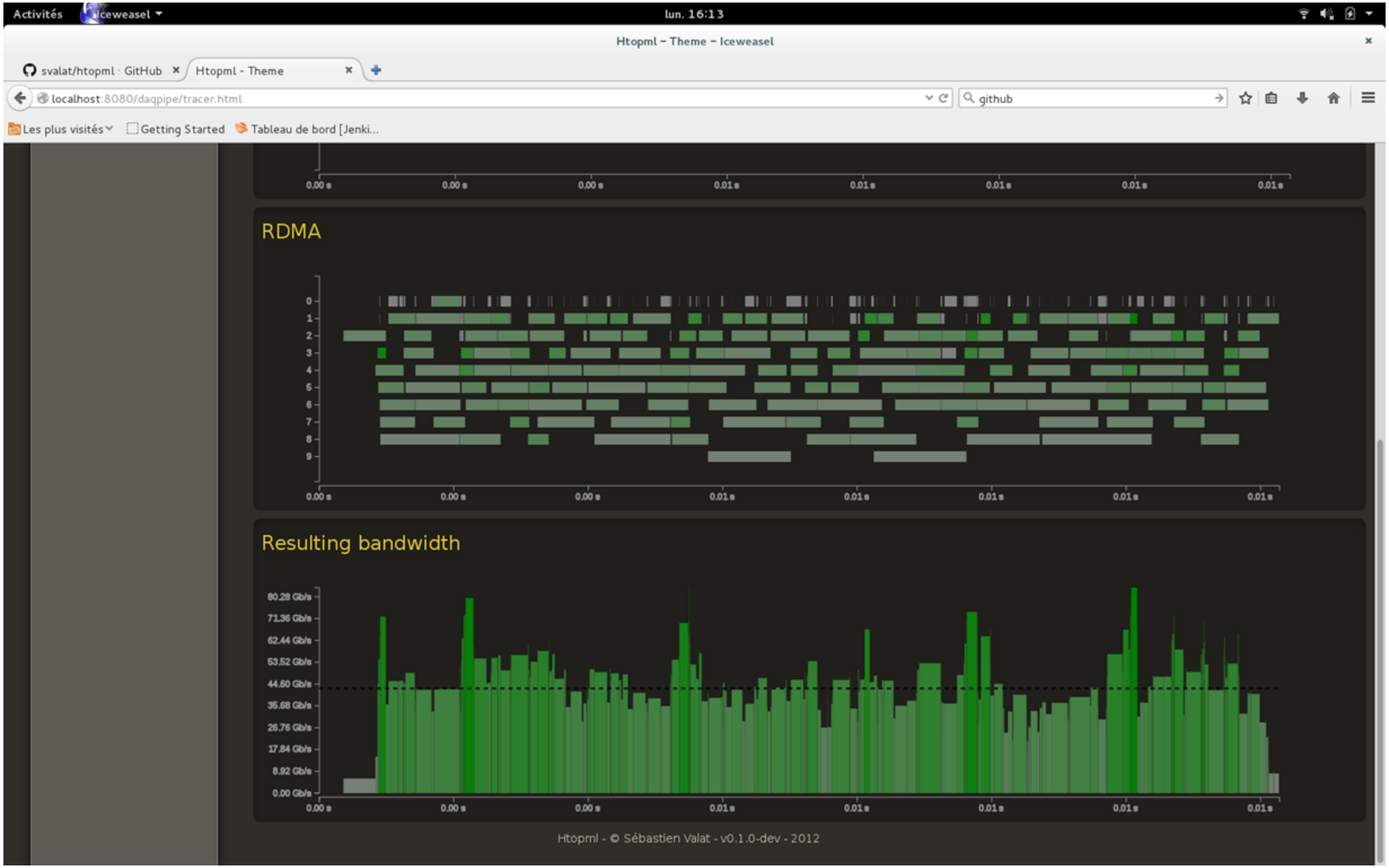
Gallileo supercomputer



OPENFABRICS ALLIANCE

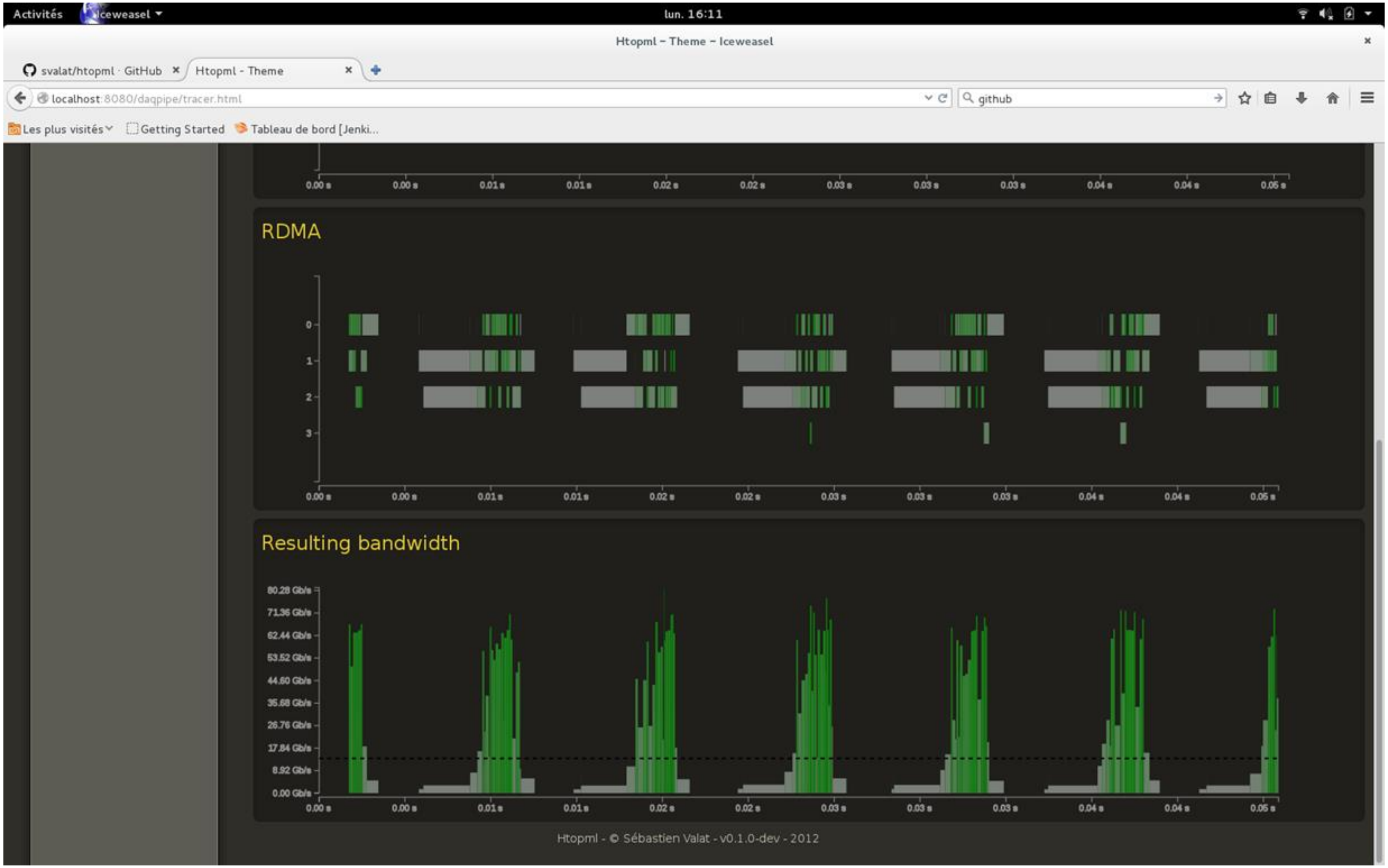
MONITORING FOR OPTIMIZATION

MONITORING COMMUNICATION SCHEDULING



Htopml - © Sébastien Valat - v0.1.0-dev - 2012

MONITORING COMMUNICATION SCHEDULING



CONCLUSION

CONCLUSION

- **Performance**
  - We **need** to achieve **~80-100 Gb/s**
  - On IB EDR, also see **80 Gb/s**, and sustained on 14 nodes.
  - **How it will scale at 500 ?**

- **Libfabric**
  - We succeed quite easily to use API on PSM :
    - Qlogick QDR
    - Intel® Omni-Path
  - Still have some issues to use on IB
  - Maybe init can be simplified ?
  - Plan to test failure recovery support soon

OPENFABRICS ALLIANCE

12th ANNUAL WORKSHOP 2016

THANK YOU

Sébastien Valat

CERN

REFS

- **Code in use :**
  - https://gitlab.cern.ch/svalat/lhcb-daqpipe-v2
  - Used tag : *bench-opa-2016-02*
  - Also see simpler benchmark in benchmarking/ubenchmark subdir
  - Libfabric used : 1.1.0 and 1.2.0
  - MPI : OpenMPI (with “-mca mtl psm2 -mca pml cm” for Intel® Omni-Path)

- **Documents**
  - [1] LHCb Trigger and Online Upgrade Technical Design Report (https://cds.cern.ch/record/1701361?ln=en)
  - [2] Current 128 node results from https://indico.cern.ch/event/382495/session/34/contribution/20/attachments/1153728/1657396/Large-scale_DAQ_tests.pdf