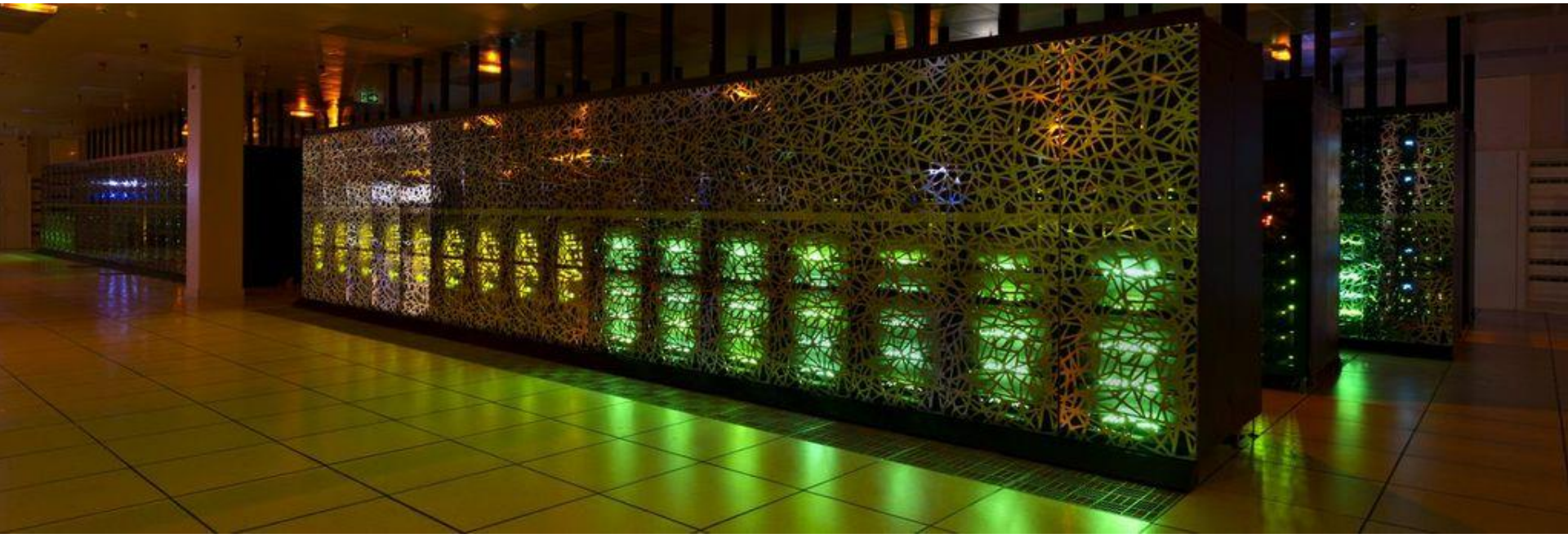
Sur le chemin de l'exascale: comment doper la performance des communications

BXI: Bull eXascale Interconnect

June 24, 2015



Bull Exascale program

bullx S6000 series

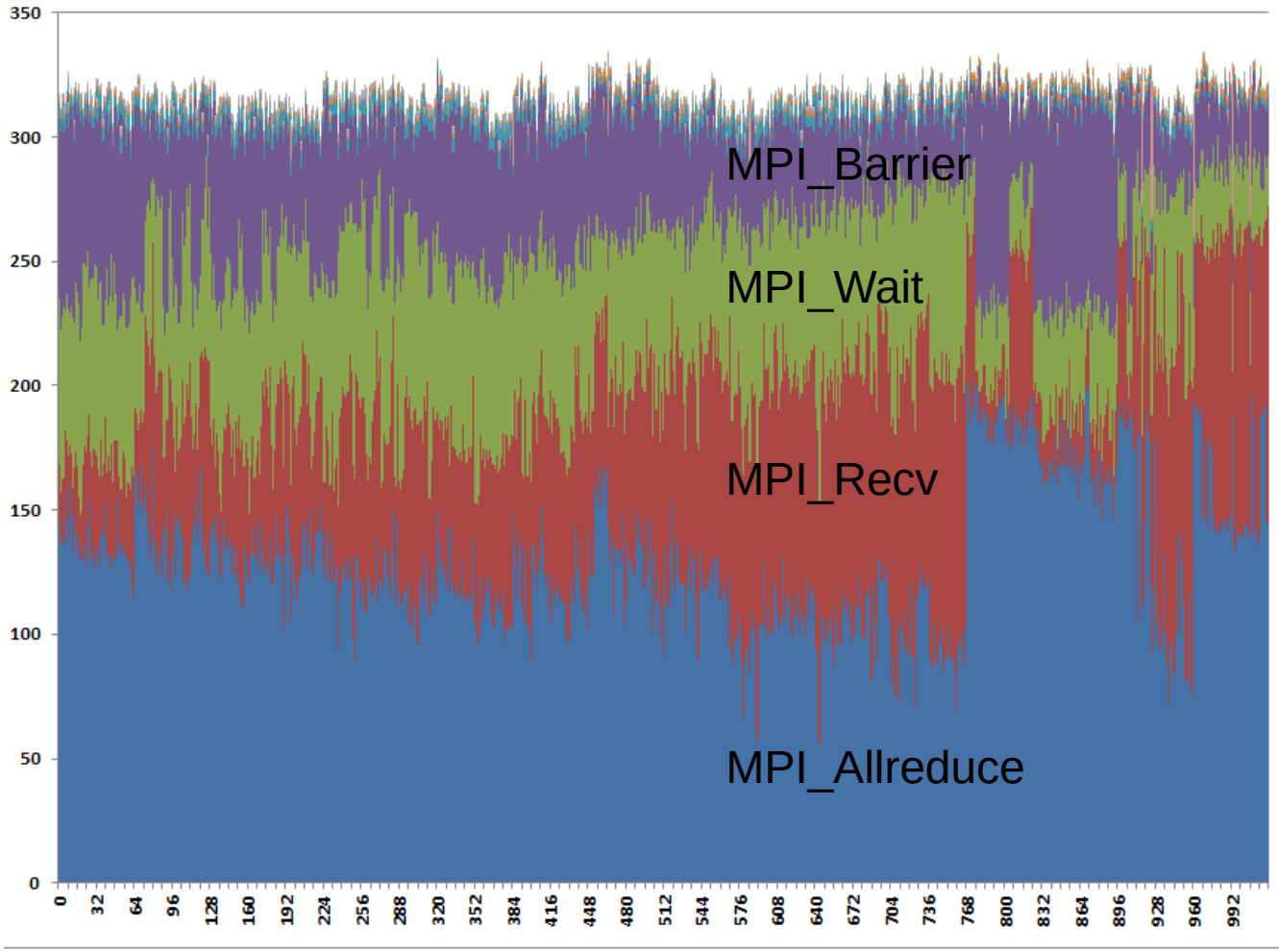
SEQUANA

BXI

Parallel programming

bullx super computer suite

Yales – MPI profile on 1024 threads



with FDR 56Gb/s

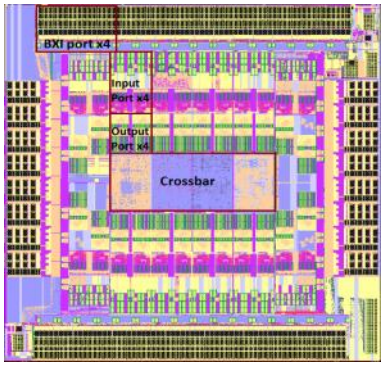
Total time	700s
MPI time	320s

with EDR 100Gb/s

MPI time	250s(*)
Total time	630s(*)

(*) estimations

BXI: Efficient communications for Exascale applications



- **Bull Interconnect for the Exascale**
  - HW acceleration →sustained performance under heavy load
  - High Bandwidth, Low latency, High message rate
  - Exascale scalability
- **BXI full acceleration in hardware**
  - Portals 4 (unconnected protocol) - Minimum constant memory footprint
  - HW support for MPI communications (send/recv, collectives, asynchronous)
  - PGAS – Partitioned Global Address Space – new programming languages
- **BXI highly scalable, efficient and reliable**
  - up to 64k nodes
  - Adaptive Routing
  - QoS / 16 virtual channels; typically 2 virtual networks (data + IO)
  - Reliable: end-to-end error checking + link level CRC & ECC

BXI Network is based on 2 ASICs

NIC ASIC

PCI Express
16x Gen 3

MPI Latency
<1 µs

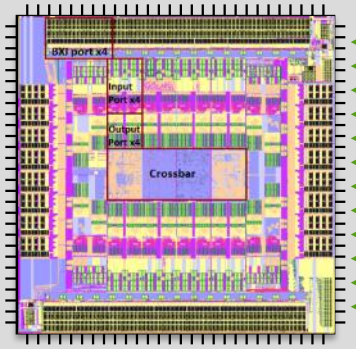
Issue rate
100 Mmsg/s

BXI Link
100 (4x25) Gb/s

Lutetia

switch ASIC

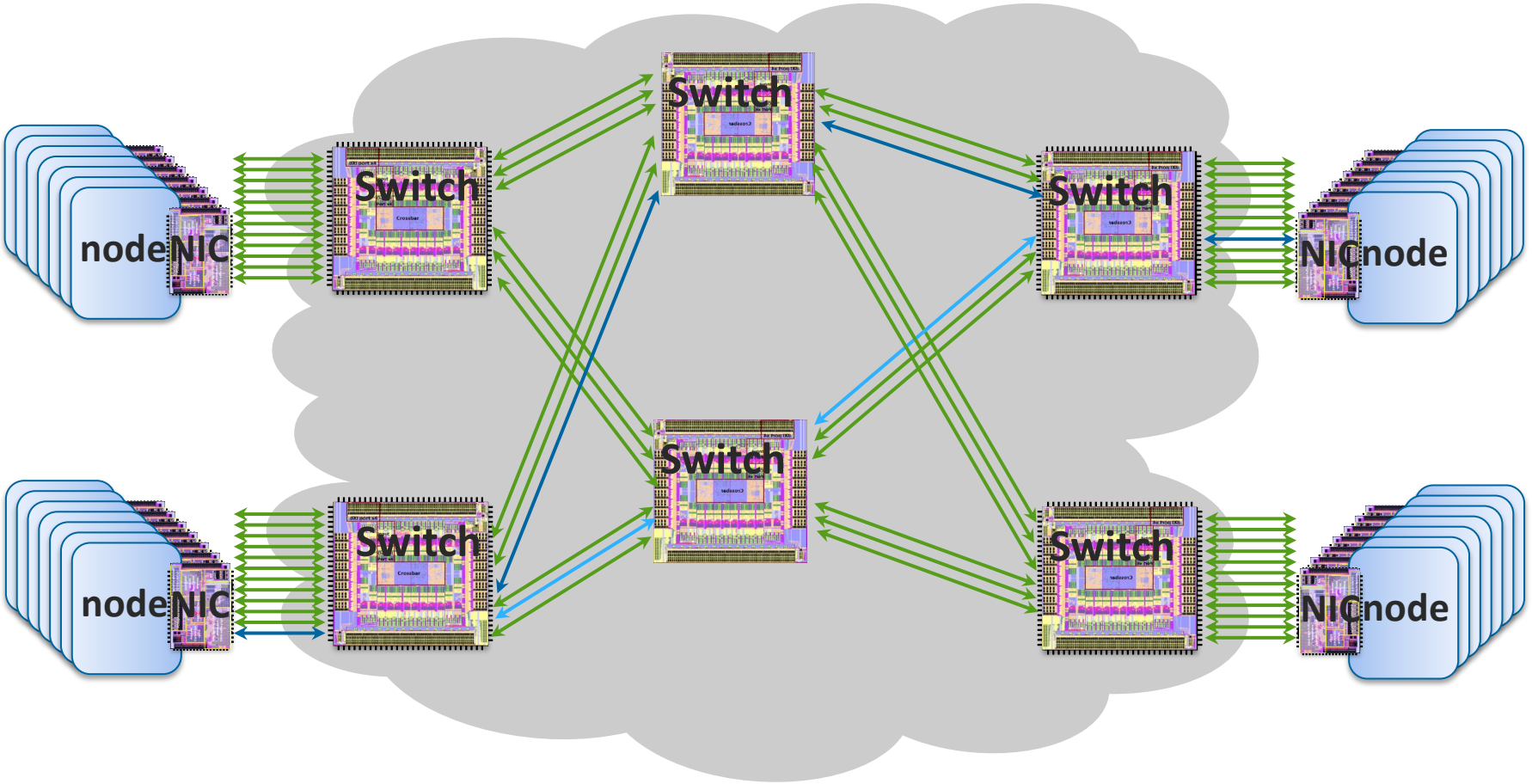
48 ports
BXI Link



9600 Gb/s
bandwidth

Divio

BXI Network: indirect connections



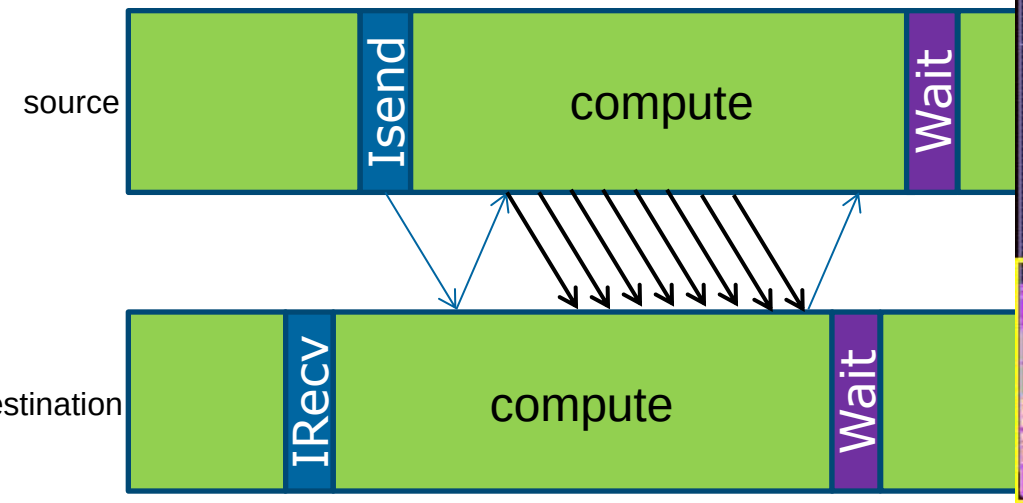
BXI: offloading MPI communication in HW (1/2)

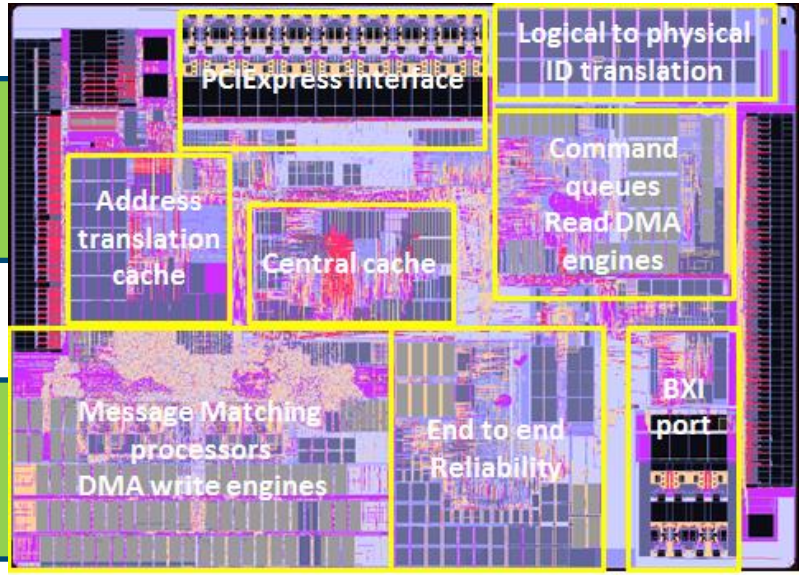
address V2P

size

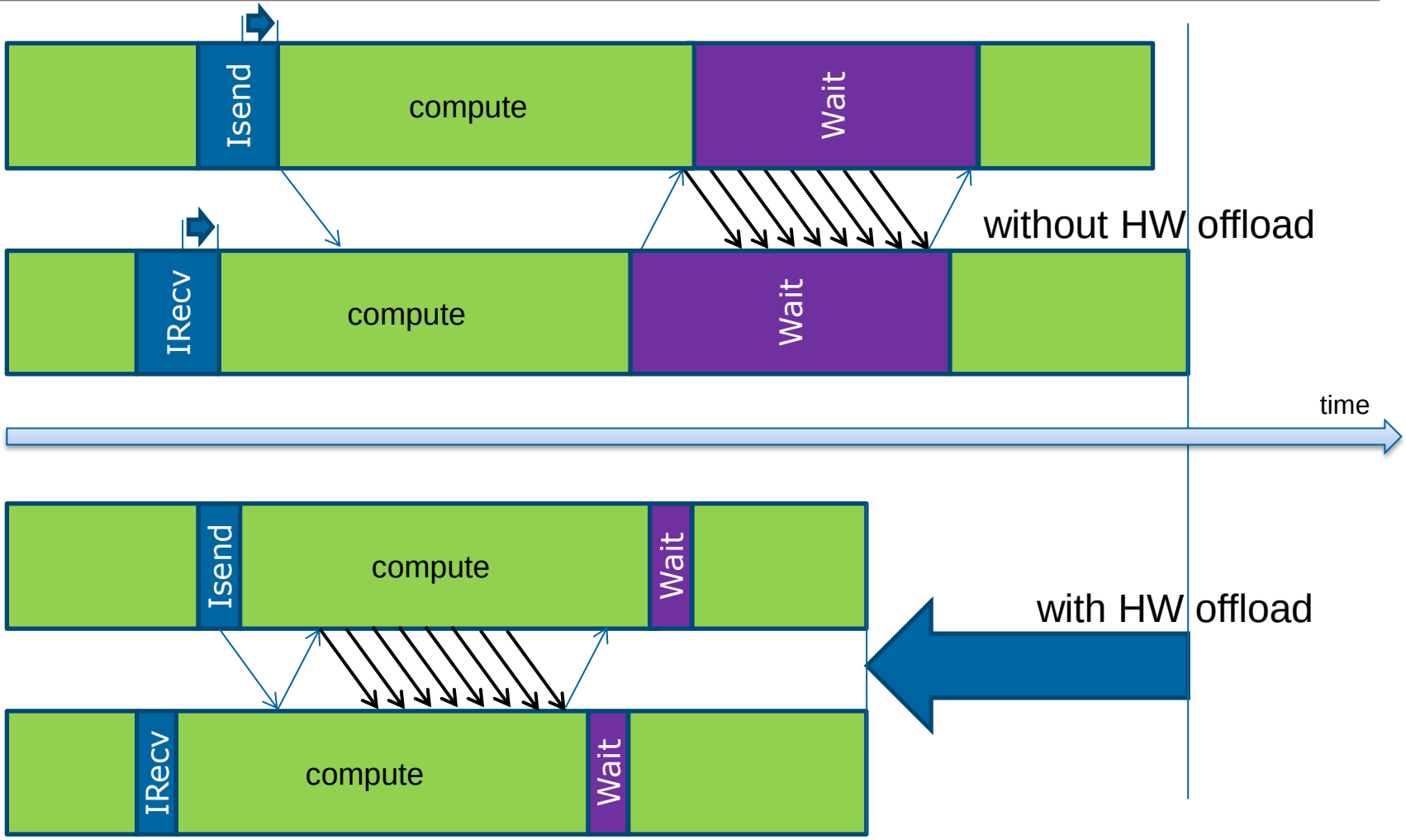
rank L2P

```
#include <mpi.h>

int MPI_Isend( const void *buf, int count, MPI_Datatype datatype, int dest,    int tag, MPI_Comm comm, MPI_Request *request)

int MPI_IRecv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Wait(MPI_Request *request, MPI_Status *status)
```

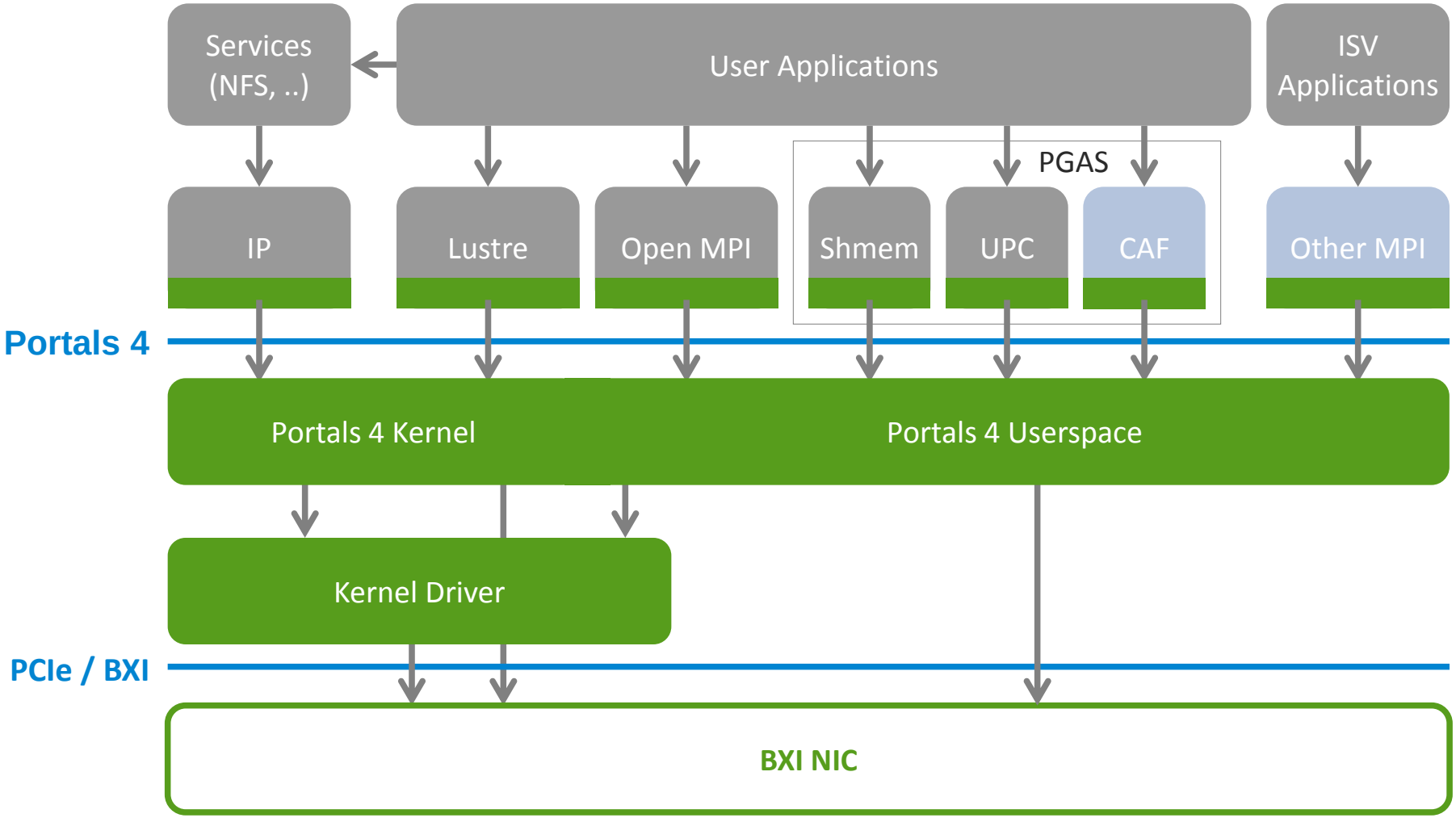




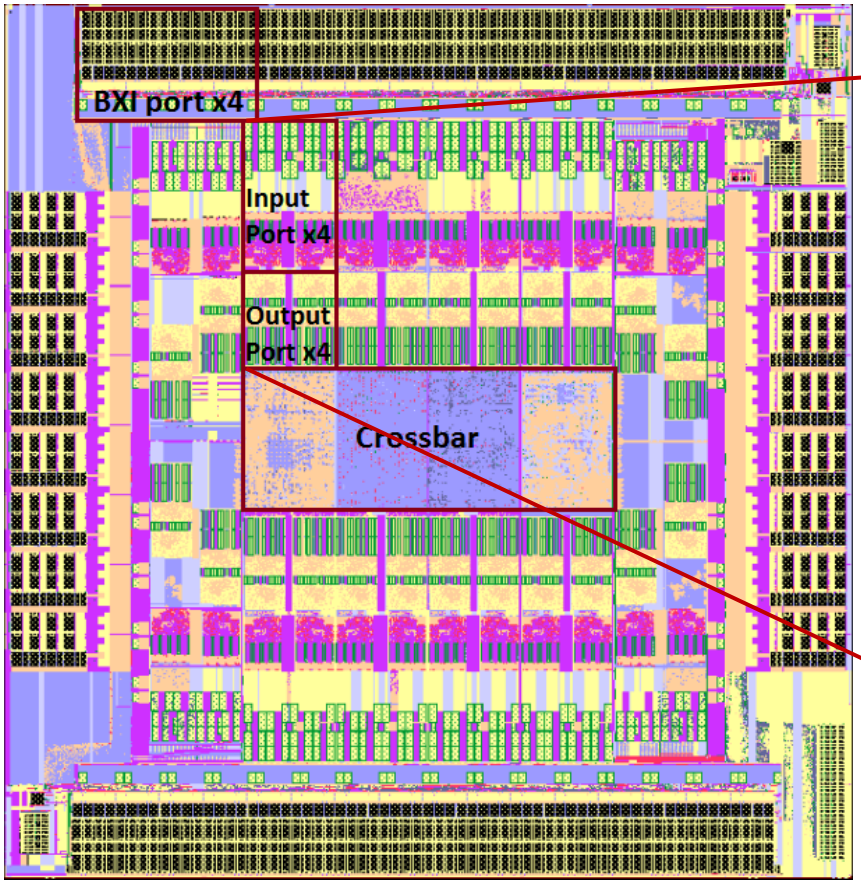
BXI: offloading MPI communication in HW (2/2)

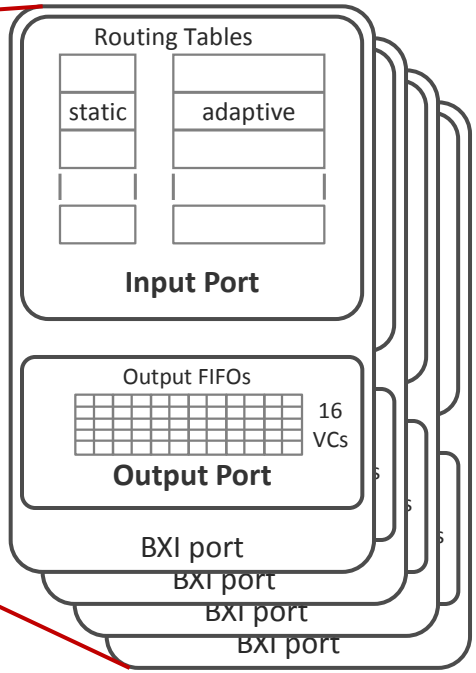


BXI Software compute stack



BXI: 48 port switch ASIC





48 ports

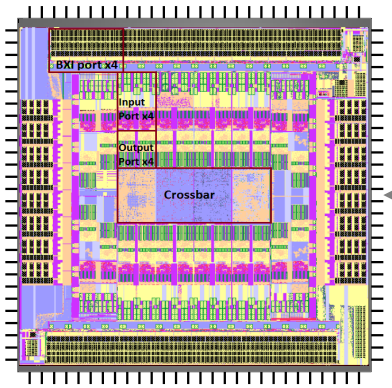
Adaptive Routing

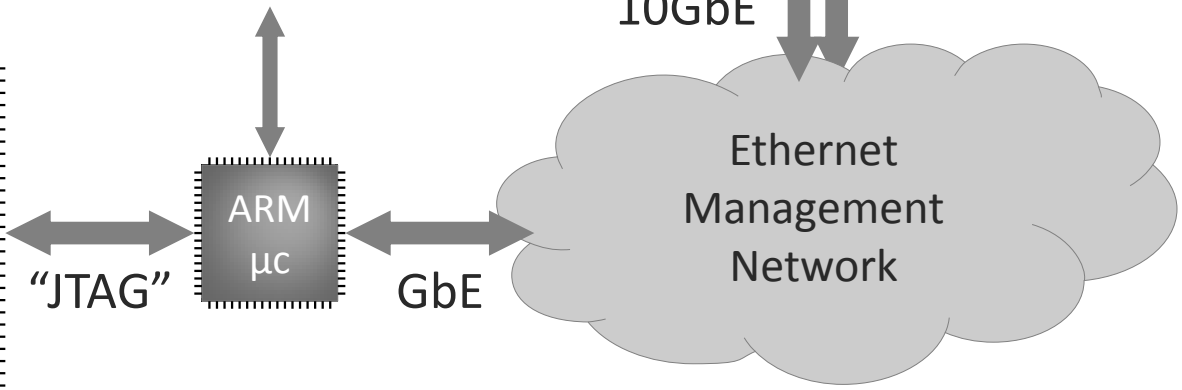
1 routing table per port

16 VCs

BXI Fabric Management

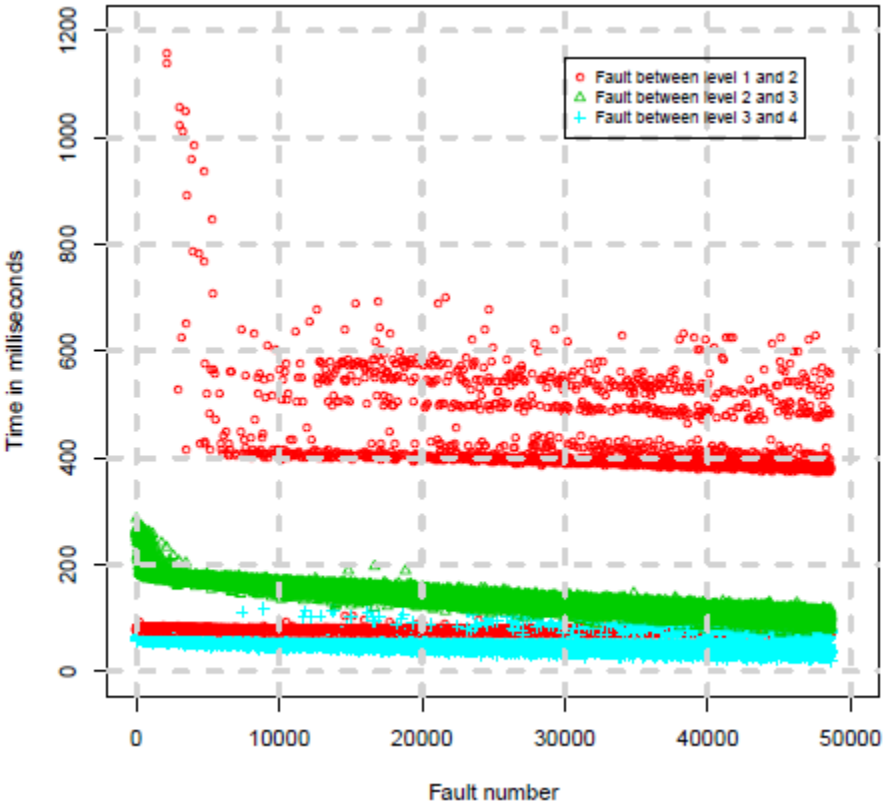
Routing	→	Up to 64k nodes
Supervision	→	Failures handling
Topology	→	Cable checking
Performance	→	Counters access

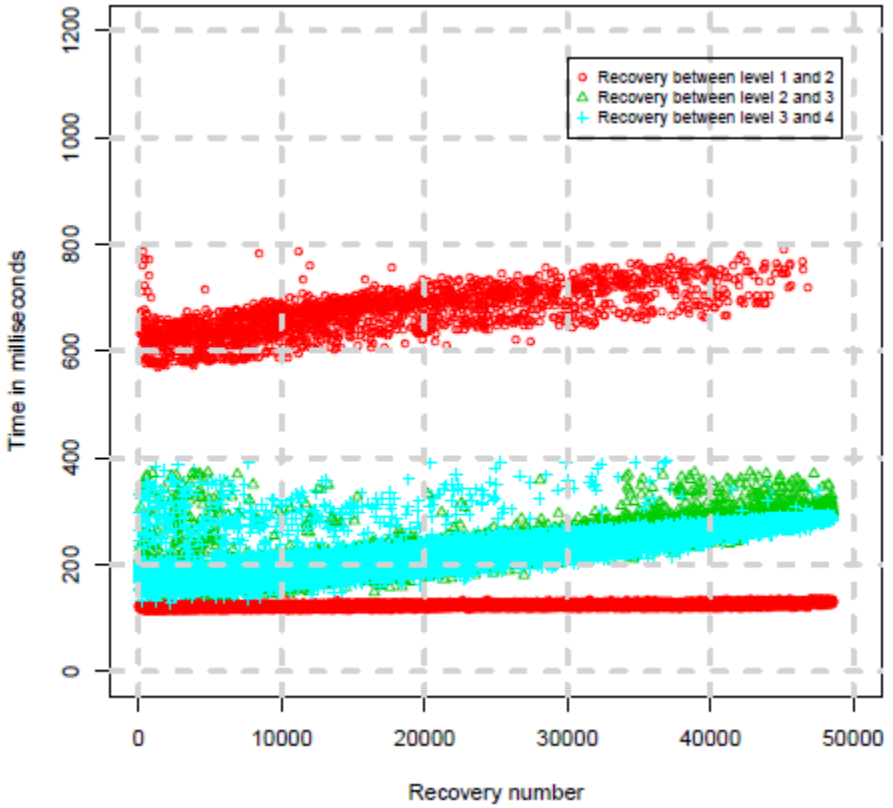




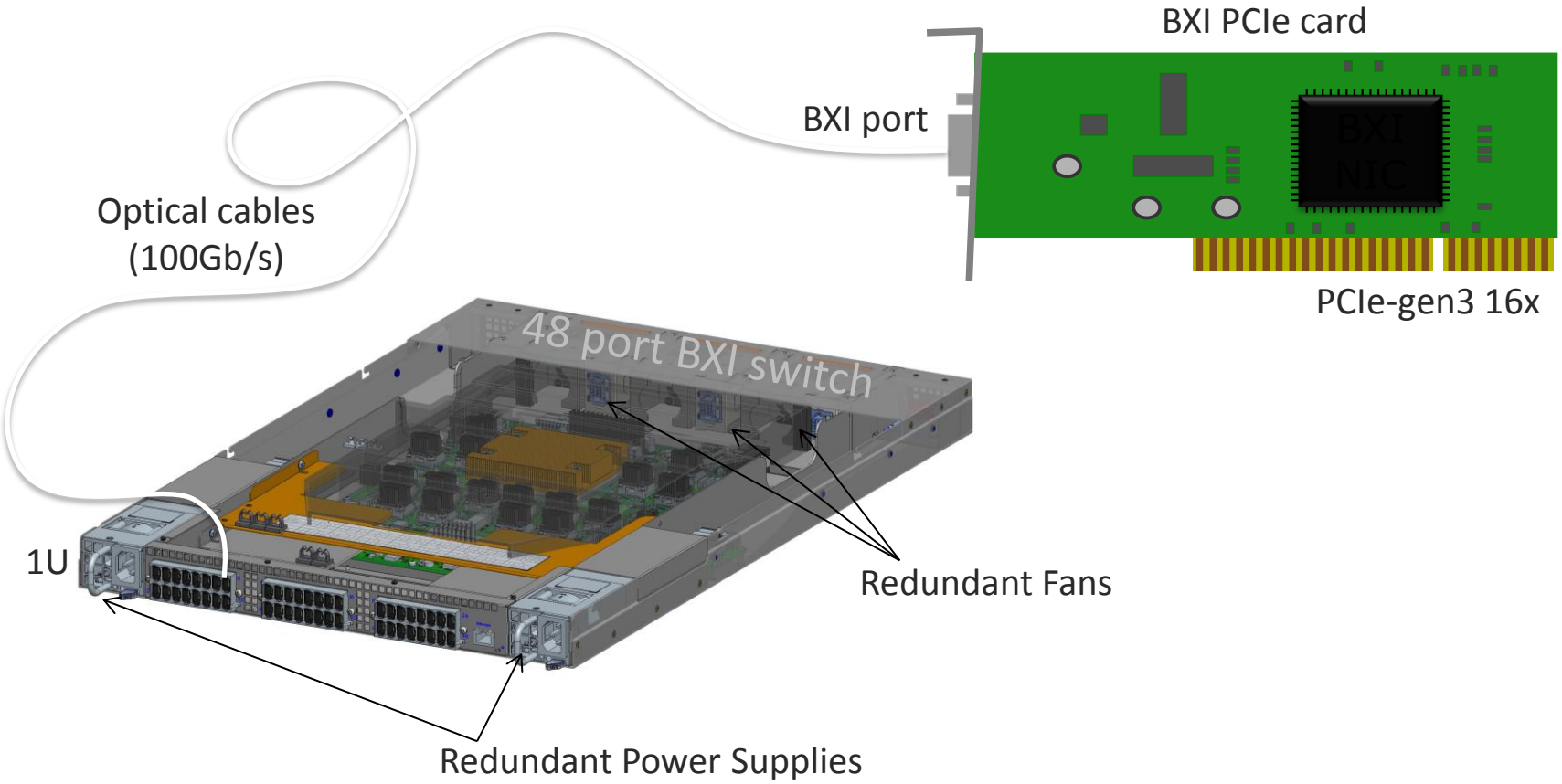
BXI Fabric Management QuickRepair

- 64k nodes
- Full routing tables computation takes >5minutes, but goal is <5s … routing algorithm cost is O(N^4)
- BXI QuickRepair computes local tables update in case of link failure and re-activation in < 1s

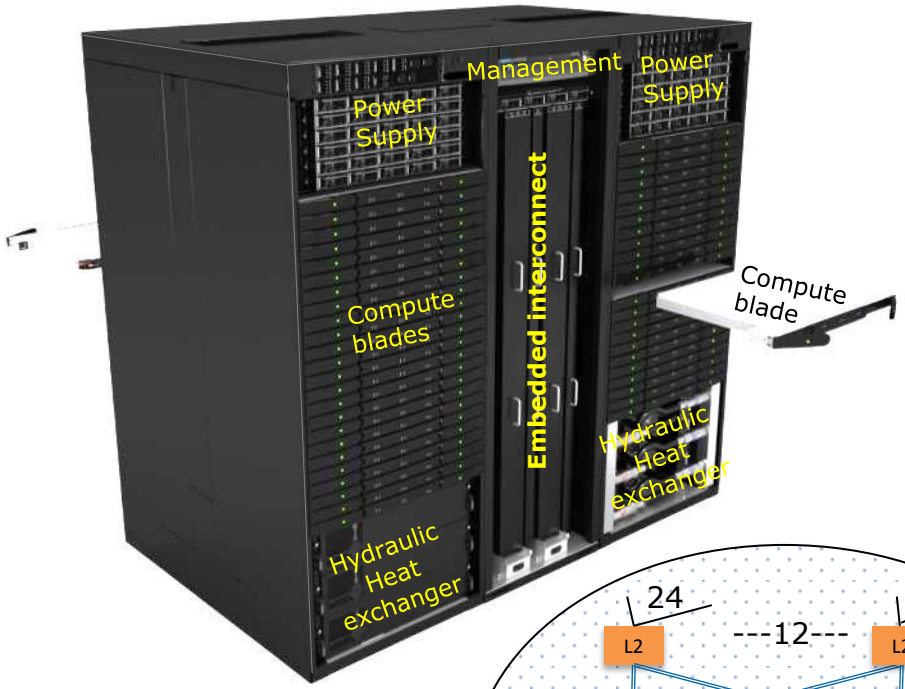


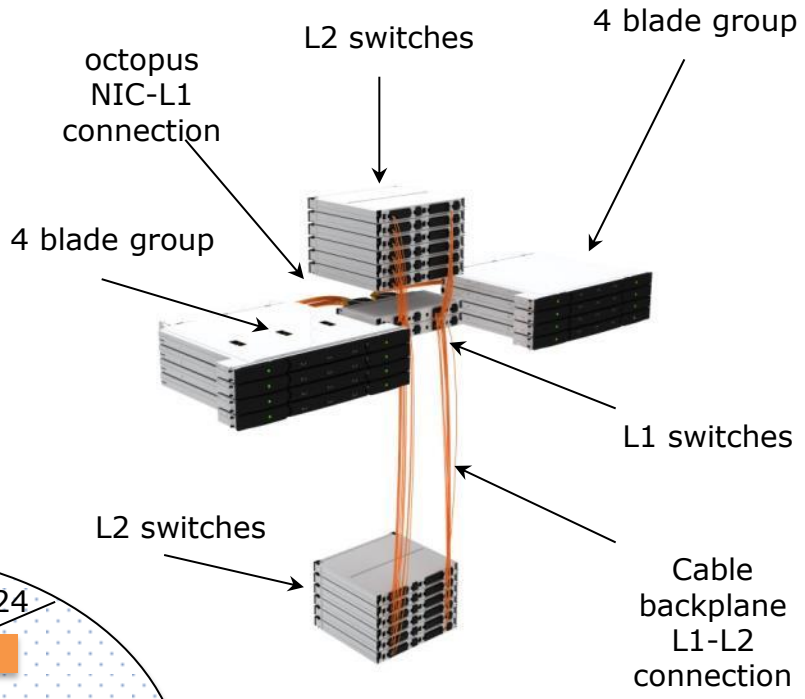


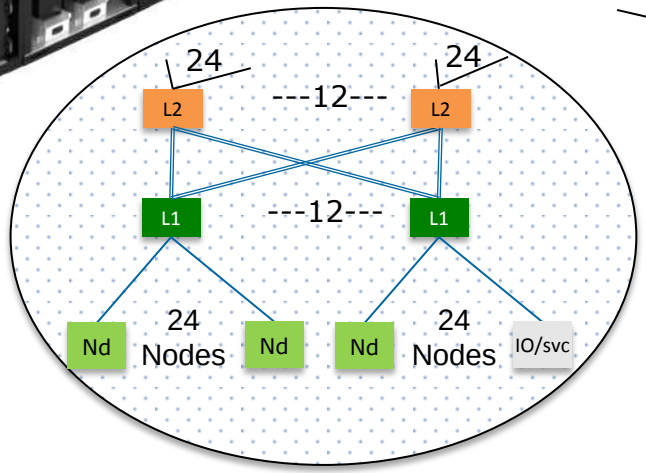
BXI
PCI adapter card and 48p standalone switch



“Sequana” – Embedded interconnect

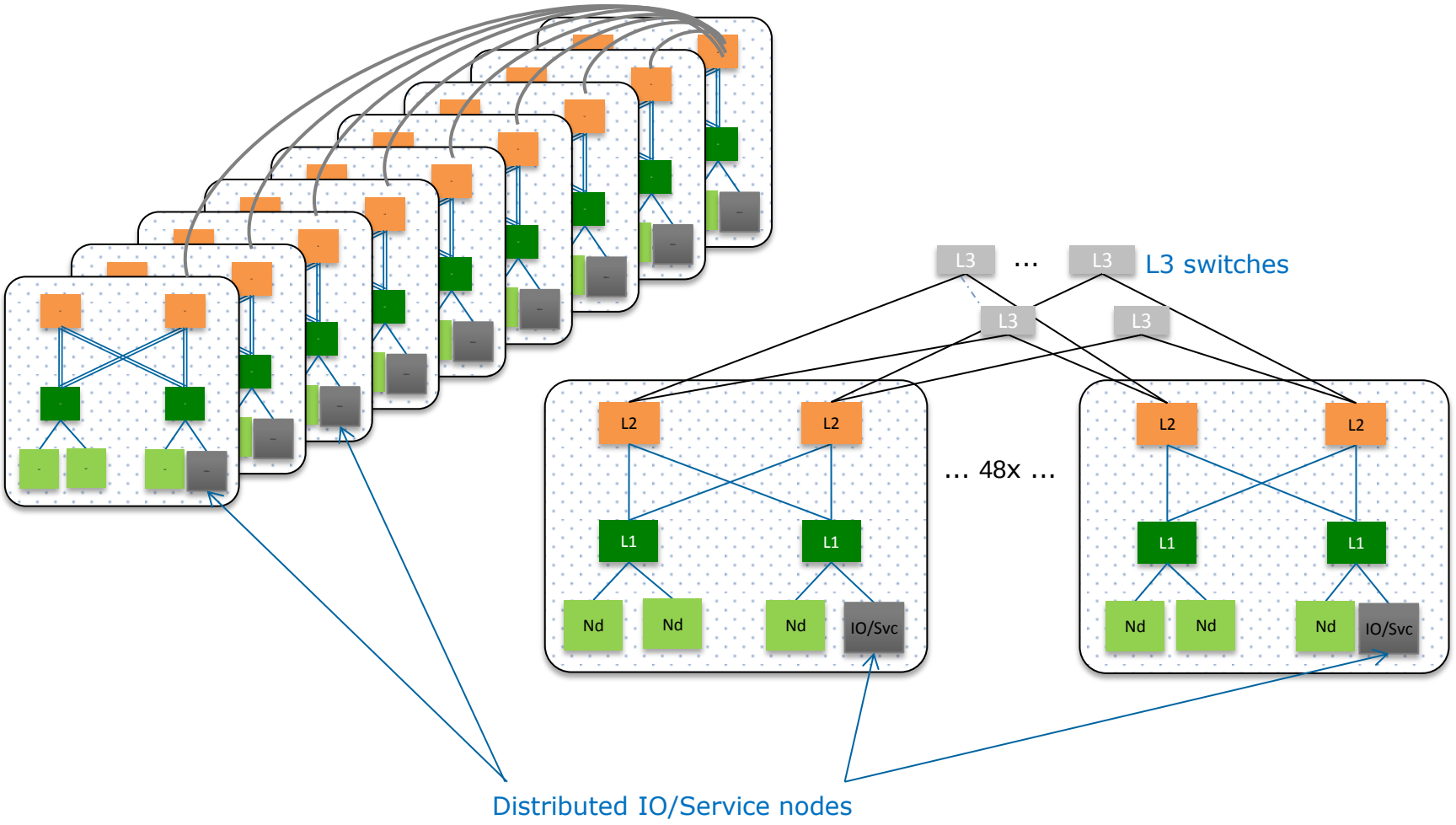






Fast Interconnect layout

Sequana cells interconnection



Questions ?

10^18 Bull exascale program

Bull
atos technologies