Isolating job for security on high-performance fabrics

Matthieu Perotin – Tom Cornebize

05-01-2017

Bull
atos technologies

Context

Introduction

Security is sometimes mandatory

- Some parallel jobs are too big for a single system…
- … but too confidential for a shared system
- The question is:

  How to be absolutely sure that one job's processes are not communicating with processes from another job

- Critical with offloaded RDMA
- Such communications could be intentional tampering or the result of programming errors
  - No hypothesis is made regarding the attack

How to isolate one job ?

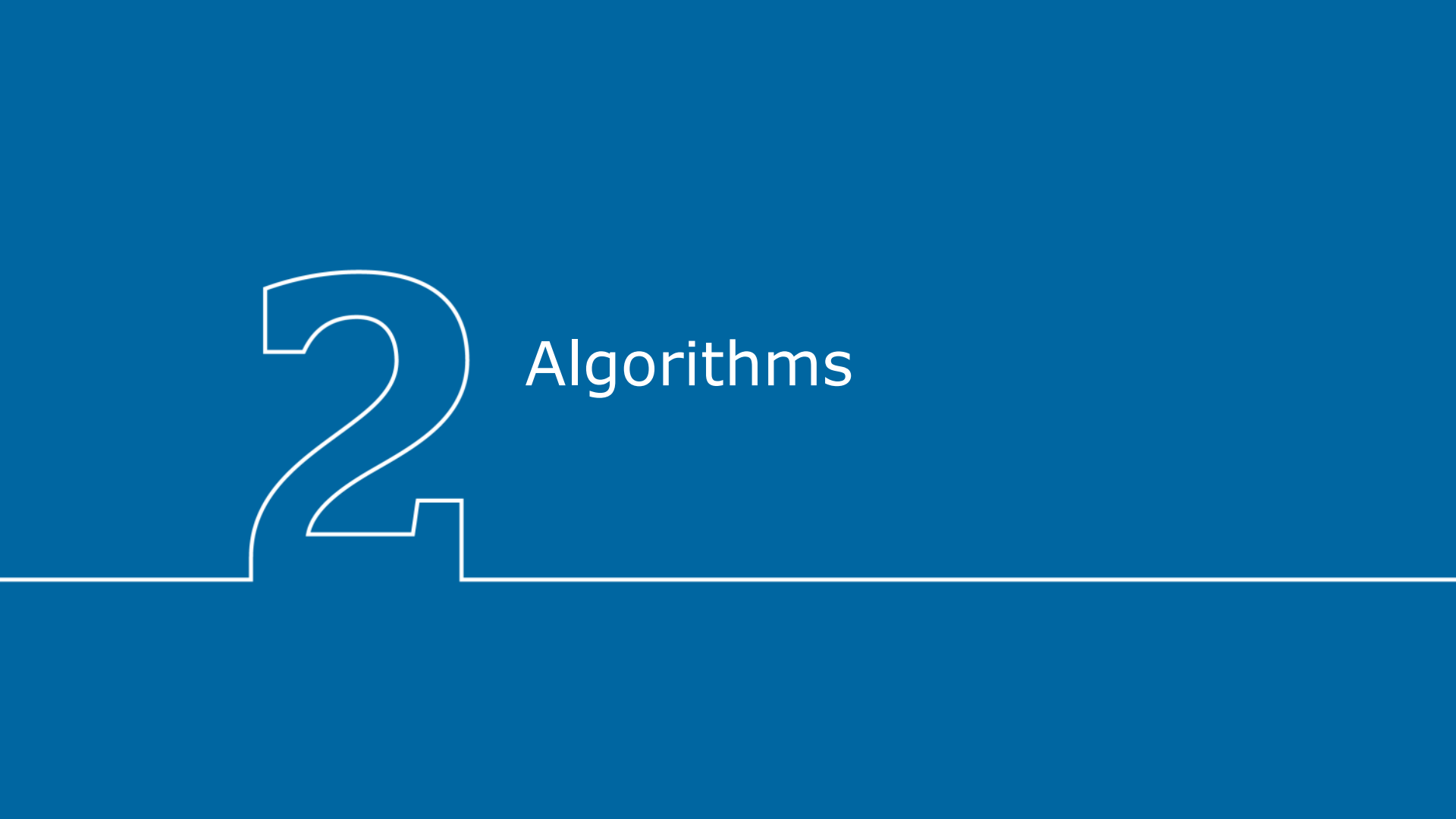
Job isolation is a huge constraint

- The need for isolation is not new
- When such critical jobs are to be run on shared cluster, an “air-gap” is created
  - One part of the fabric is physically separated from the rest
  - The job is run on the isolated part of the cluster
- Such an operation is particularly heavy
  - It is a maintenance operation, that needs to be scheduled
  - It results in two distinct fabrics that must be handled separately by the fabric management
- However, the connexity of the fabric could be broken logically by acting at the routing level
- Logical isolation must be dynamic and fast
  - Done when the job is launched

The BXI Interconnect

How is the routing working ?

- BXI (Bull eXtreme Interconnect): Atos/Bull's own 100Gb/s IC
- Wormhole routing (message based)
- BXI switches have per port destination based routing tables
- Every time a message enters an *input port*
  - The destination NID (Node ID) is extracted from the message header
  - The *output port* is chosen from the input port's routing table
    - f(destination NID) = output port
  - Adaptive routing is possible
    - If the chosen *output port* is overloaded, another port is chosen among a list of possible ports


2
Algorithms

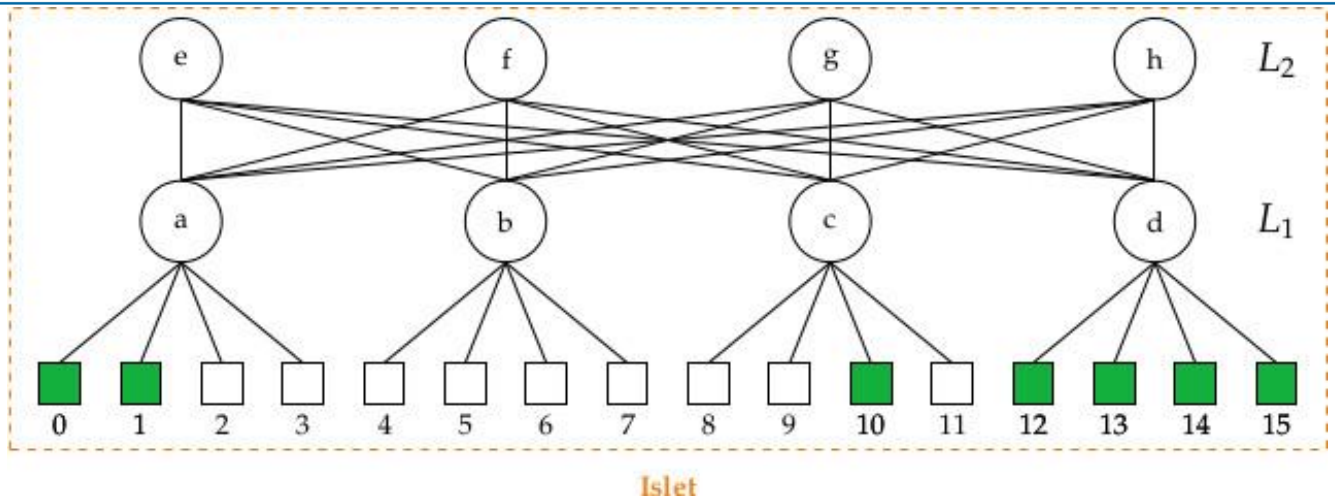
Job isolation

The idea behind the proposed solution

- Act on the routing tables in the *input ports*
  - f(destination NID) = {output port}
- Remove all entries from the set of output ports that would allow a message to be forwarded to a node without a process from the same job
- Hypothesis: no shared nodes
- How would that work ?
  1. Schedule the job
  2. Modify the routing tables (isolation)
  3. run job
  4. Modify the routing tables (back to normal)

Isolating jobs

A naïve algorithm



- Forbid incoming messages
  - Invalidate green destinations
    - switch a, ports 2 and 3
    - switch b, ports 4, 5, 6 and 7
    - switch c, ports 8, 9 and 11
- Forbid outgoing messages
  - Invalidate white destinations
    - switch a, ports 0 and1
    - switch c, port 10
    - switch d, ports 12 to 15

Job Isolation

A naïve algorithm

- Works on any topology
- Independent from the routing algorithm
  - And thus from re-routing operations
- Polynomial: $O(n^2)$
- Can adding hypothesis lead to fewer invalidations ?

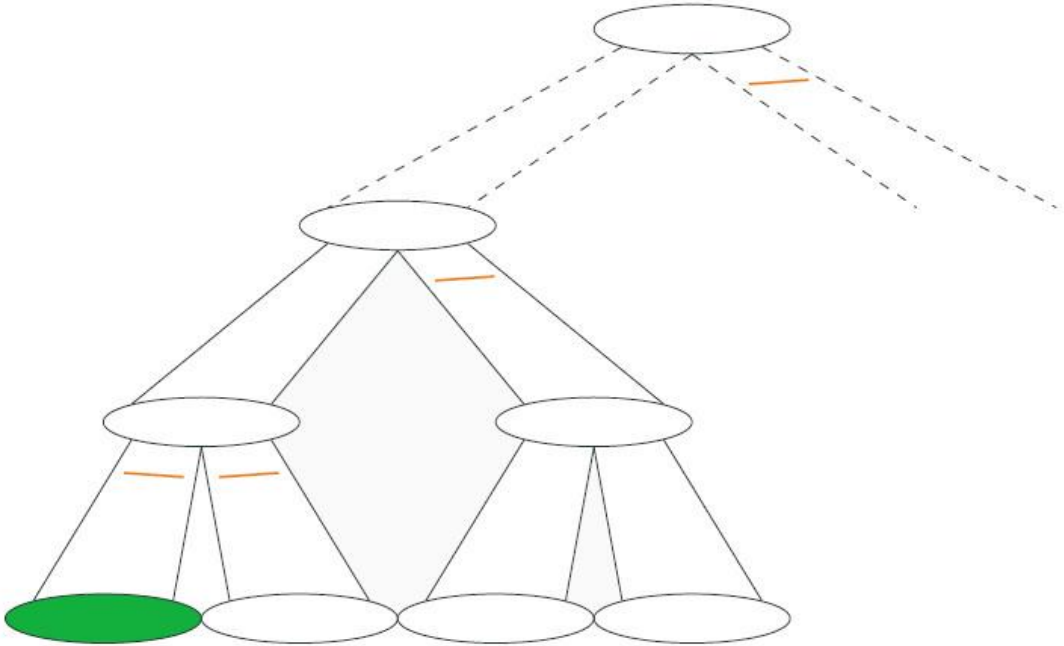
Working with fat-trees

Reducing the number of invalidated entries

- Upper Layers algorithm
  - Use the tree structure of the fat-tree and try to cut whole branches rather than individual leaves

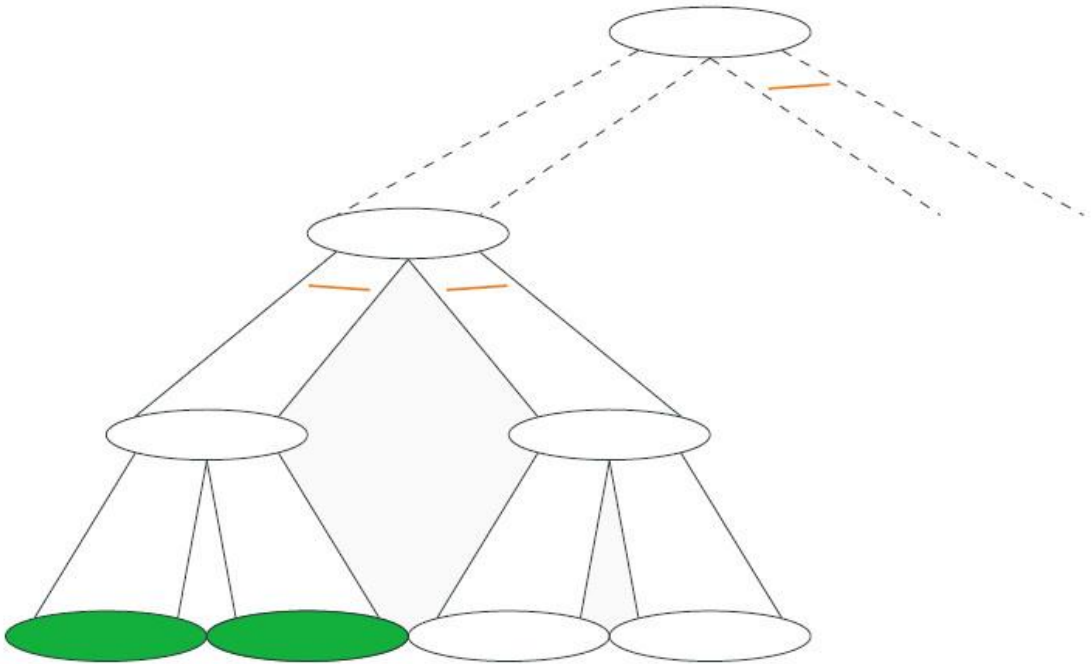
Job isolation

The upper layer algorithm



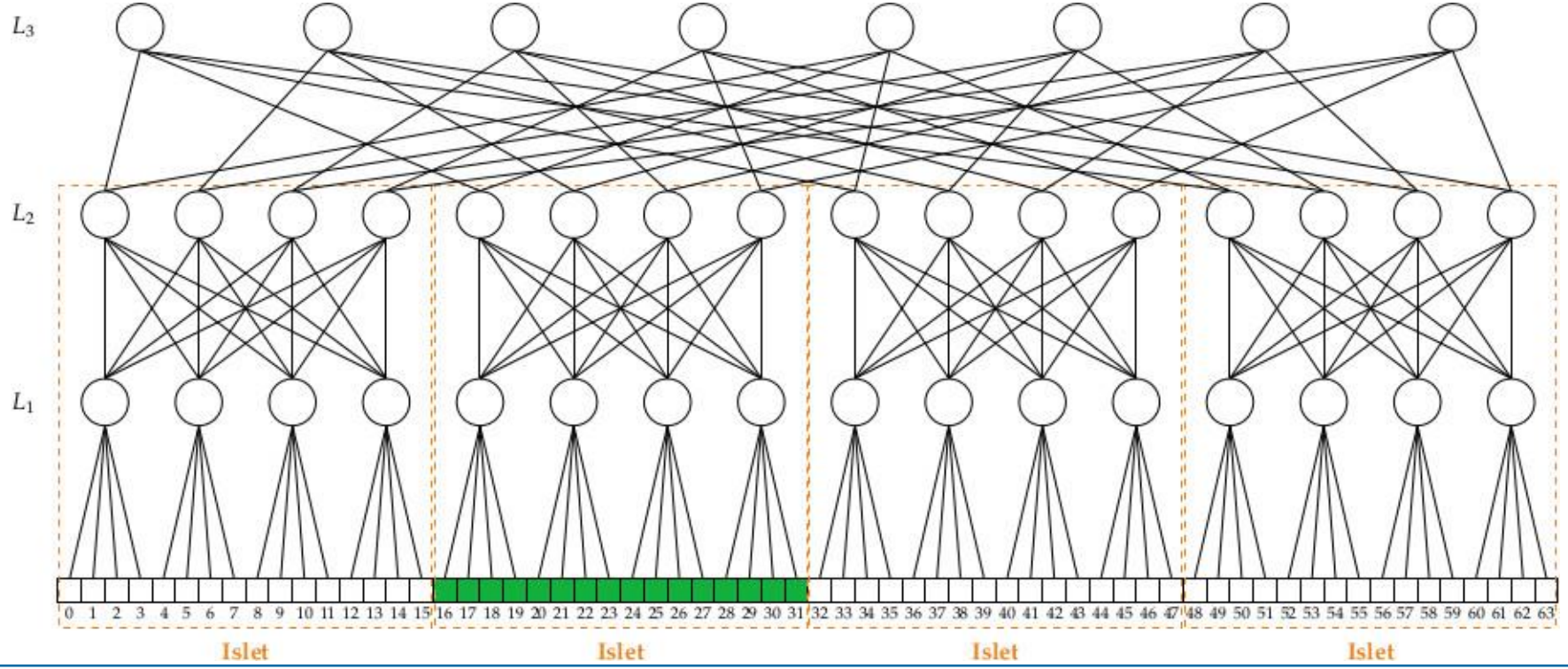
Job isolation

The upper layer algorithm



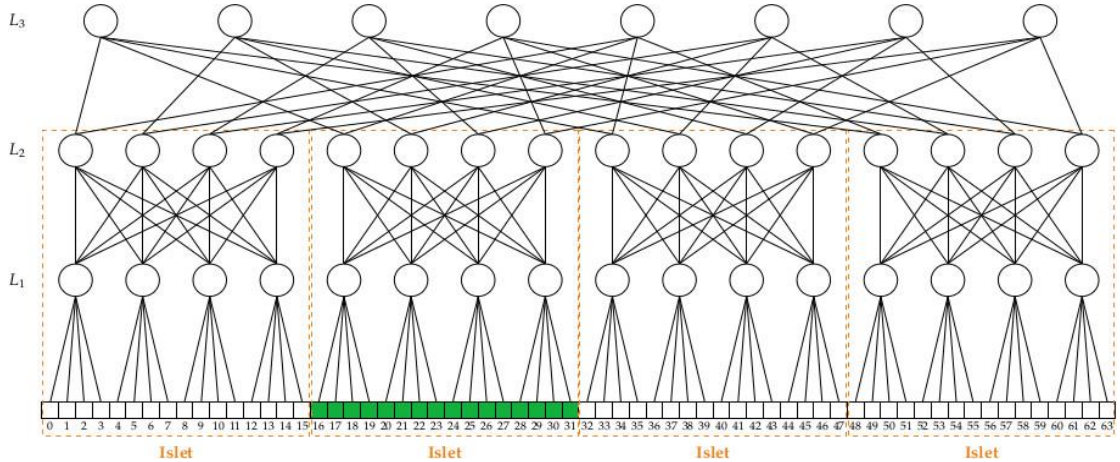
Job isolation

The upper layer algorithm



Job isolation

The upper layer algorithm



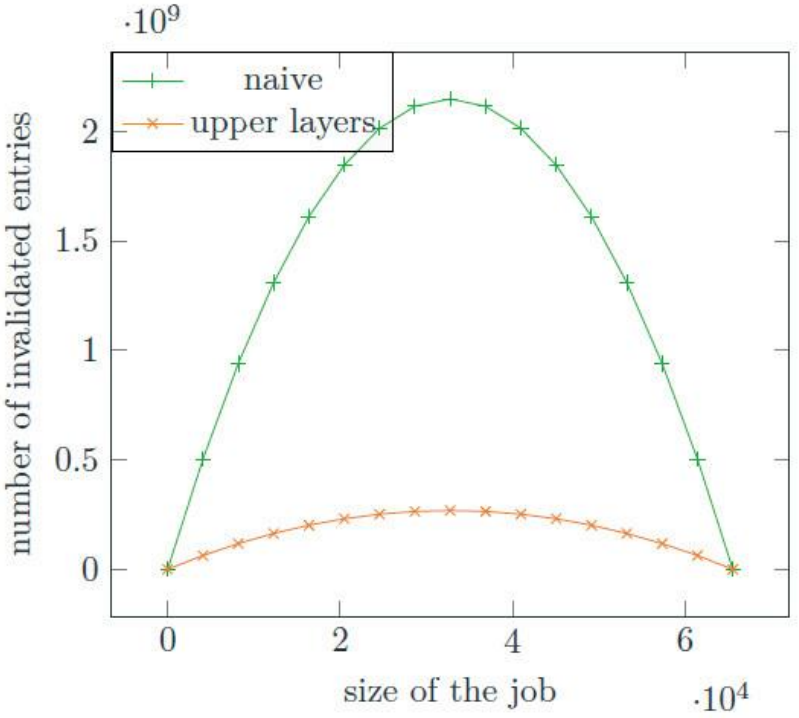
- Naïve Algorithm: 1536 entries invalidated
- Upper Layer: 768
- Ratio: 0.5 (For this specially tailored, ideal case. Works at least as well.)
- The algorithm is still independent from the routing algorithm


3
Implementation

Proof of concept implementation

Number of invalidated entries

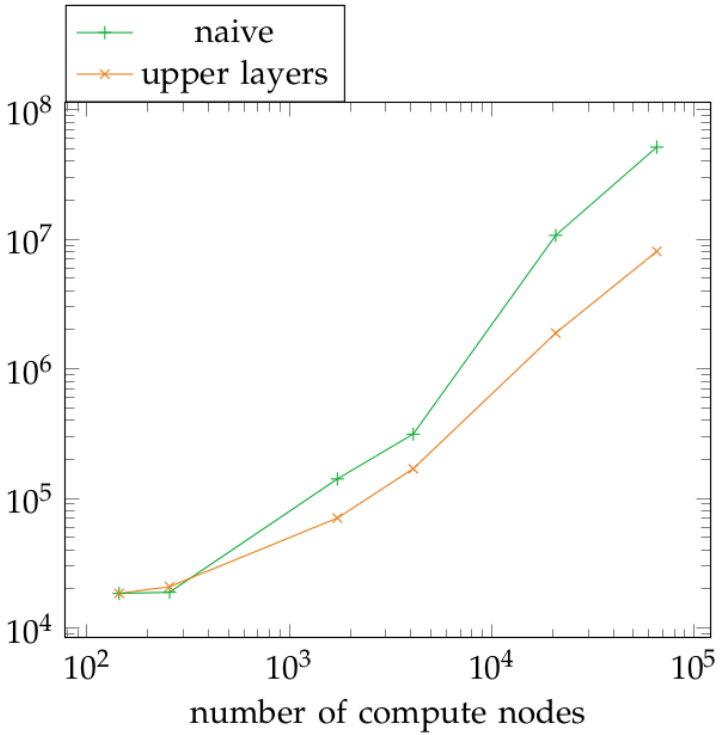
- 4 level, 64k nodes PGFT
- Job are randomly chosen union of sub-trees of level 3
- Worst case, when a job is half the size of the cluster
  - naïve: 2.147G entries invalidated
  - Upper Layers: 268M



Proof of concept implementation

Implementation results – Memory consumption

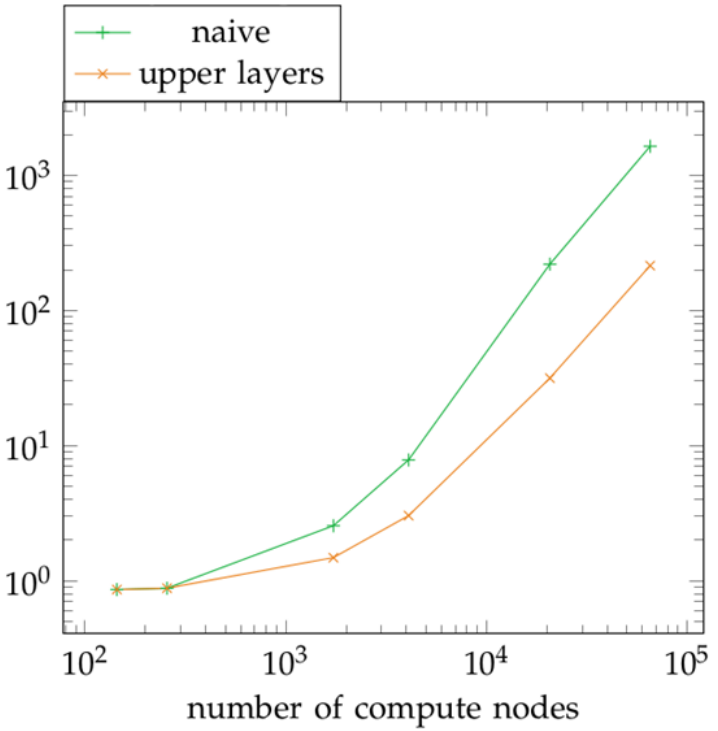
- Naïve
  - From 18MB to 51GB
- Upper Layers
  - From 18MB to 8 GB



Proof of concept implementation

Implementation results – Runtime (12 Xeon cores)

- Naïve
  - From 0.8s to 1649s
- Upper Layers
  - From 0.8s to 214s



Concluding remarks

Concluding remarks

And Future works

- Two algorithms for logical isolation were presented
  - One which is independent from routing algorithms and topologies
  - Another which is independent from routing algorithms but specific to fat-trees
- The implementation results showed that dynamic isolation is possible with the BXI interconnect
- Further studies will consider routing algorithm dependent isolation
  - spoiler: good preliminary results
- A practical first step in scheduling aware routing for security
- Performance oriented isolation is a field that remains to be explored
  - How can a lower bound on the performances for a job’s communication be obtained ?

Thanks

For more information please contact:
matthieu.perotin@atos.net

Bull
atos technologies