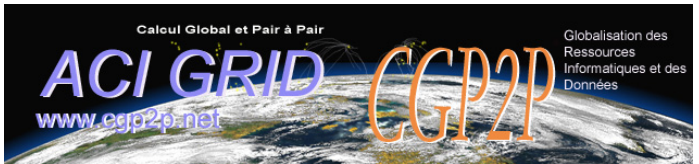
MPICH-V
MPI Implementation for Volatile resources

Grand Large
INRIA

Coordinated Checkpoint Versus Message Log For Fault Tolerant MPI

Aurélien Bouteiller bouteill@lri.fr
joint work with
F.Cappello, G.Krawezik, P.Lemarinier

Cluster&Grid group, Grand Large Project
http://www.lri.fr/~gk/MPICH-V



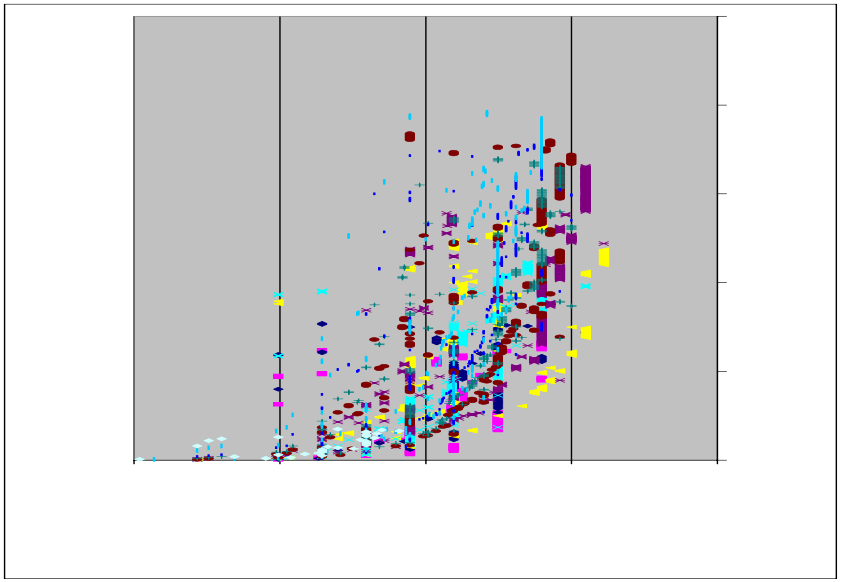
HPC trend: Clusters are getting larger

- High performance computers have more and more nodes (more than 8000 for ASCI Q, more than 5000 for BigMac cluster, 1/3rd of the installations of top500 have more than 500 processors).

  More components increases fault probability

  ASCI-Q full system MTBF is estimated (analytic) to few hours (Petrini: LANL), a 5 hours job with 4096 procs has less than 50% chance to terminate
- Many numerical applications uses MPI library

  Need for automatic fault tolerant MPI

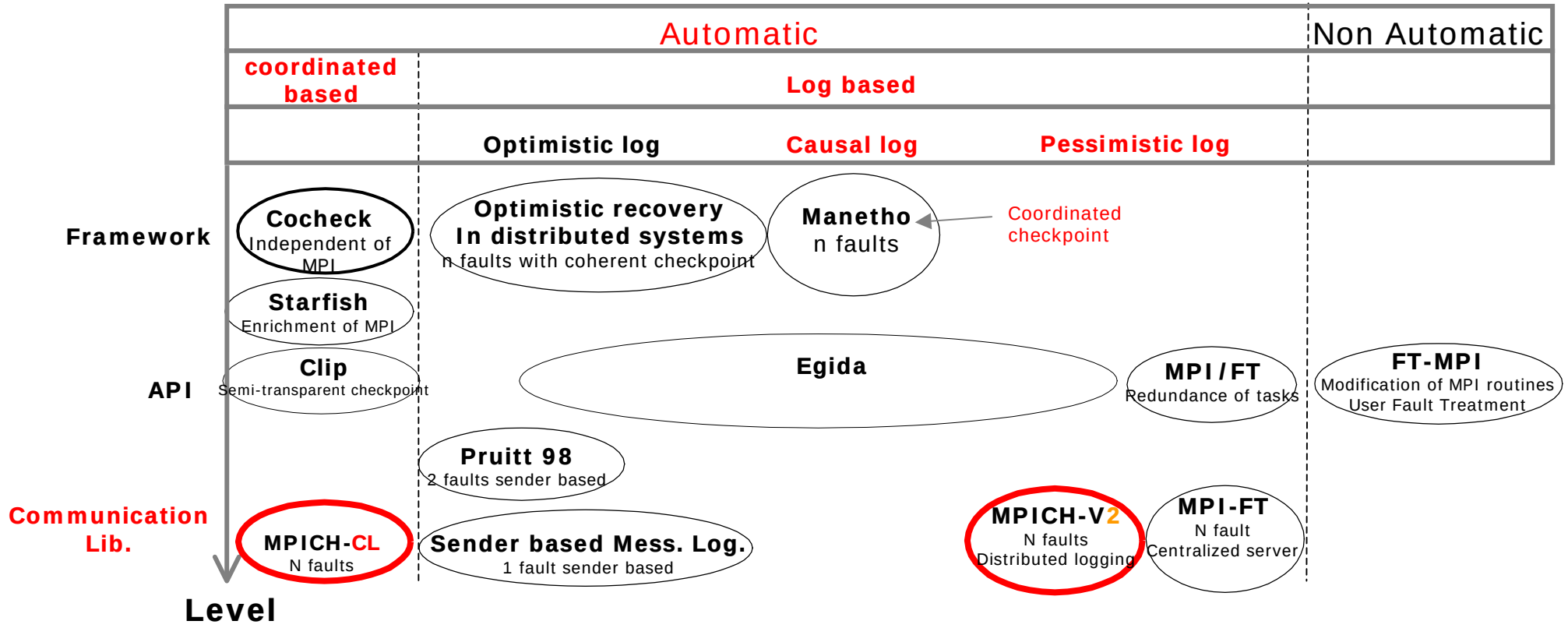


Fault tolerant MPI

A classification of fault tolerant message passing environments considering

A) level in the software stack where fault tolerance is managed and

B) fault tolerance techniques.



Several protocols to perform fault tolerance in MPI applications with N faults and automatic recovery : Global checkpointing, Pessimistic/Causal Message log

compare fault tolerant protocols for a single MPI implementation

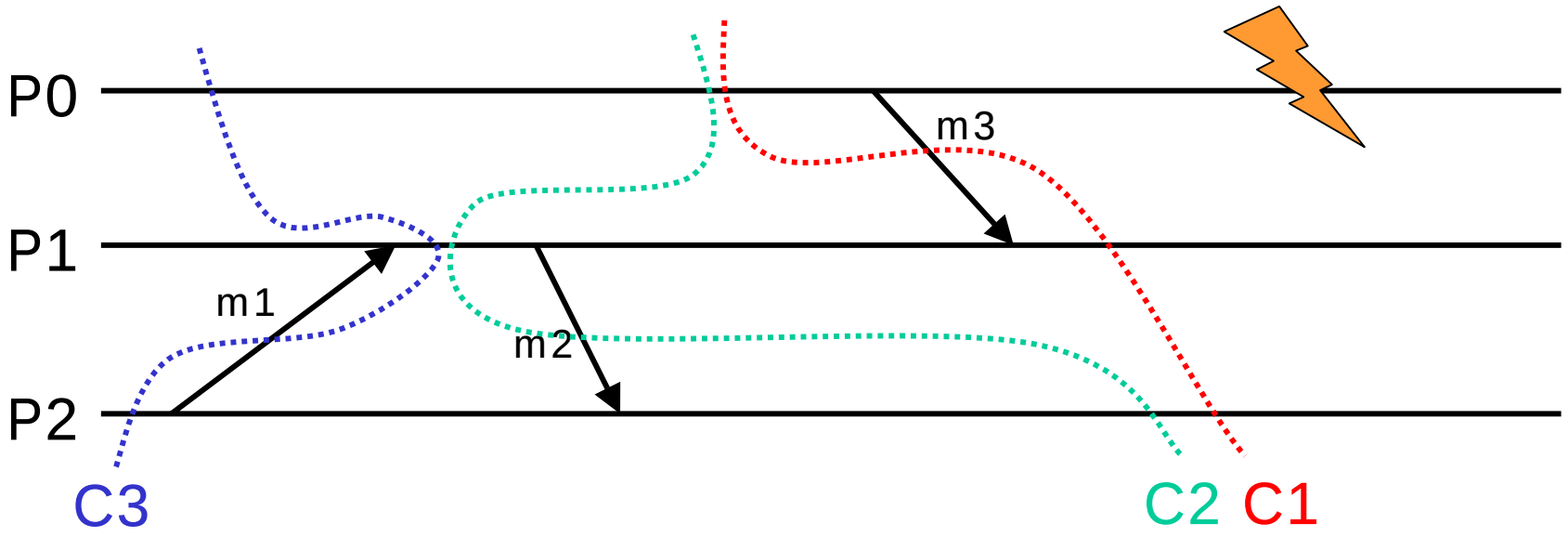
Outline

- Introduction
- **Coordinated checkpoint vs message log**
- Comparison framework
- Performances
- Conclusions and future works

Fault Tolerant protocols
Problem of inconsistent states

- Uncoordinated checkpoint : the problem of inconsistent states
  - Order of message receptions are undeterministic events
    - ➡ message received but not sent are inconsistent
  - **Domino effect** can lead to rollback to the begining of the execution in case of fault

Possible loose of the whole execution and unpredictive fault cost



Fault Tolerant protocols
Global Checkpoint 1/2

- Communication Induced Checkpointing
  - Does not require global synchronisation to provide a global coherent snapshot
  - Drawbacks studied in L. alvisi, E. Elnozahy, S. Rao, S. A. Husain, and A. D. Mel. An analysis of communication induced checkpointing. In *29th symposium on Fault-Tolerant Computing (FTFC'99).* IEEE Press, June 99.
    - number of forced checkpoint increases linearly with number of nodes ⇨ **Does not scale**
    - **Unpredictable frequency of checkpoint** may lead to take an overestimated number of checkpoints
    - Detection of a possible inconsistent state induces blocking checkpoint of some processes

Blocking checkpoint has a dramatic overhead on fault free execution

These protocols may not be used in practice

Fault Tolerant protocols
Global Checkpoint 2/2

- Coordinated checkpoint
  - All processes coordinate their checkpoints so that the global system state is coherent (Chandy & Lamport Algorithm)

    Negligible overhead on fault free execution
  - Requires global synchronization (may take a long time to perform checkpoint because of checkpoint server stress)
  - In the case of a single fault, all processes have to roll back to their checkpoints

    High cost of fault recovery

Efficient when fault frequency is low

Fault tolerant protocols
Message Log 1/2

- Pessimistic log
  - All messages recieved by a process are logged on a reliable media before it can causally influence the rest of the system

    Non negligible overhead on network performances in fault free execution
  - No need to perform global synchronization

    Does not stress checkpoint servers
  - No need to roll back non failed processes

    Fault recovery overhead is limited

Efficient when fault frequency is high

Fault tolerant protocols
Message Log 2/2

- Causal log
  - Designed to improve fault free performance of pessimistic log
  - Messages are logged locally and causal dependencies are piggybacked to messages

    Non negligible overhead on fault free execution, slightly better than pessimistic log
  - No global synchronisation

    Does not stress checkpoint server
  - Only failed process are rolled back
  - Failed Processes retrieve their state from dependant processes or no process depends on it.

    Fault recovery overhead is limited but greater than pessimistic log

Comparison: Related works

Several protocols to perform automatic fault tolerance in MPI applications

- Coordinated checkpoint
- Causal message log
- Pessimistic message log

All of them have been studied theoretically but not compared

- **Egida compared log based techniques**
  Sirram Rao, Lorenzo Alvisi, Harrick M. Vim: The cost of recovery in message logging protocols. In *17th symposium on Reliable Distributed Systems (SRDS),* pages 10-18, IEEE Press, October 1998
  - Causal log is better for single nodes faults
  - Pessimistic log is better for concurrent faults
- No existing comparison of coordinated and message log protocols
- No existing comparable implementations of coordinated and message log protocols
- high fault recovery overhead of coordinated checkpoint
- high overhead of message logging on fault free performance

  ⇨ Suspected : fault frequency implies tradeoff

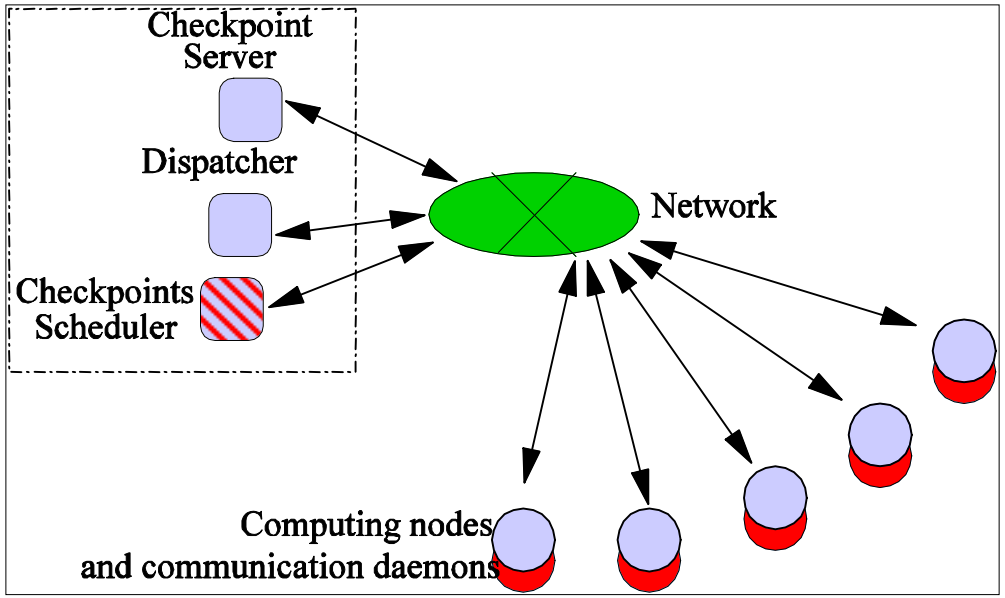
Compare coordinated checkpoint and pessimistic logging

Outline

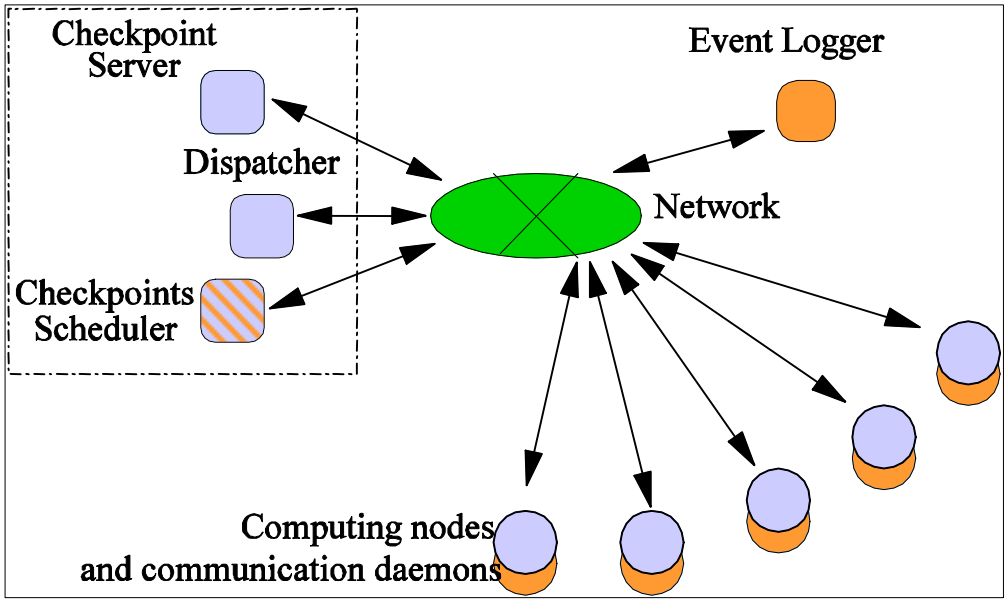
- Introduction
- Coordinated checkpoint vs Message log
- Related work
- **Comparison framework**
- Performances
- Conclusions and future works

Architectures

We designed MPICH-CL and MPICH-V2 in a shared framework to perform a fair comparison of coordinated checkpoint and pessimistic message log

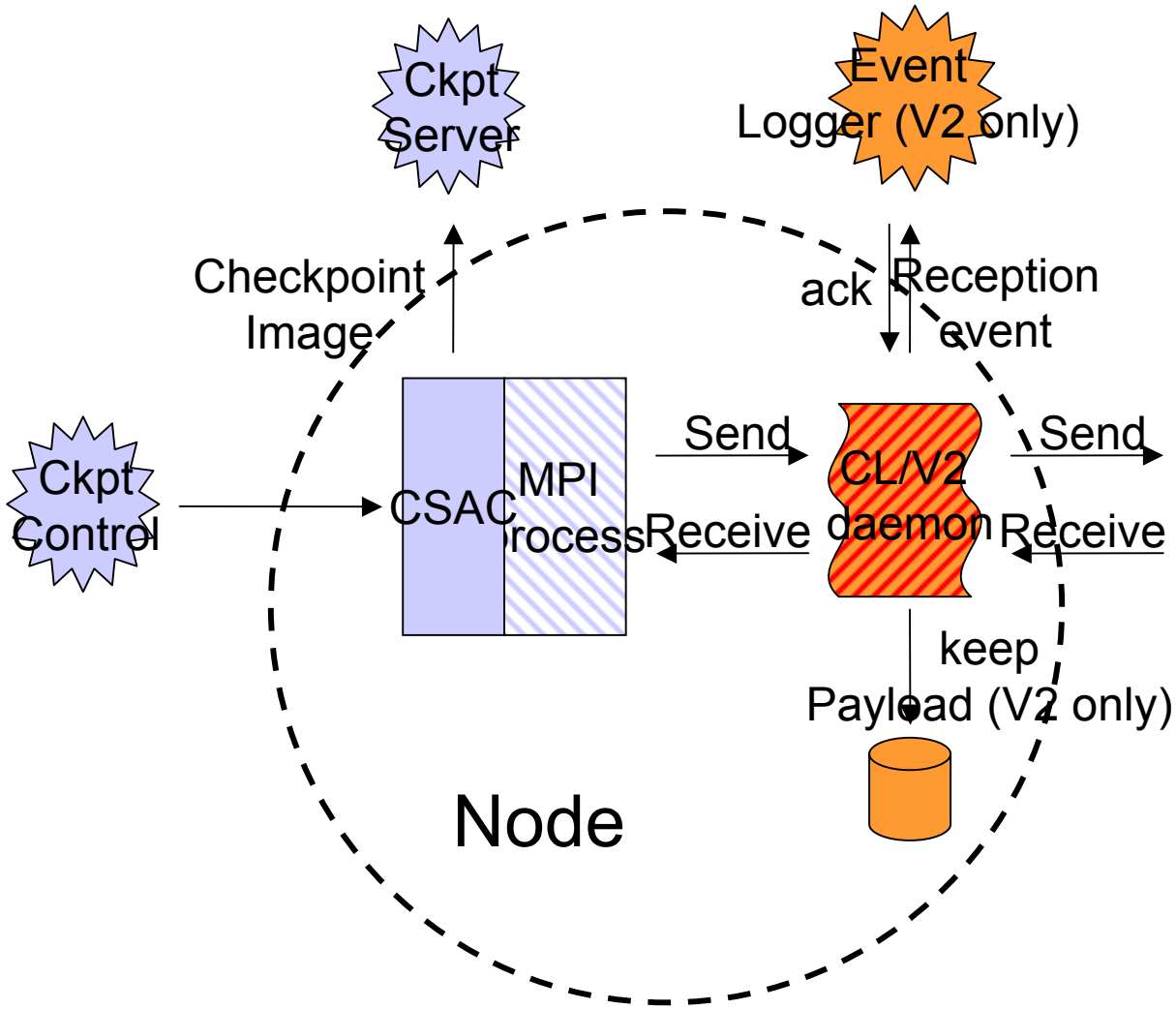


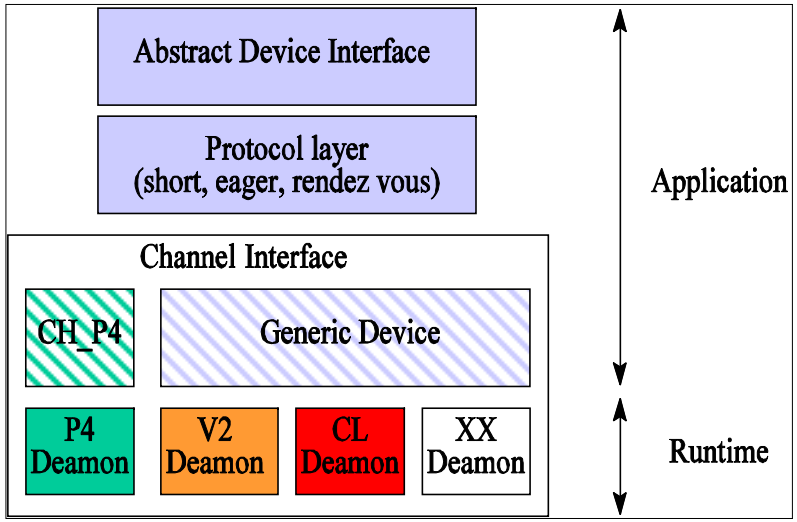
MPICH-CL
Chandy&Lamport algorithm
Coordinated checkpoint



MPICH-V2
Pessimistic sender Based
message log

Communication daemon





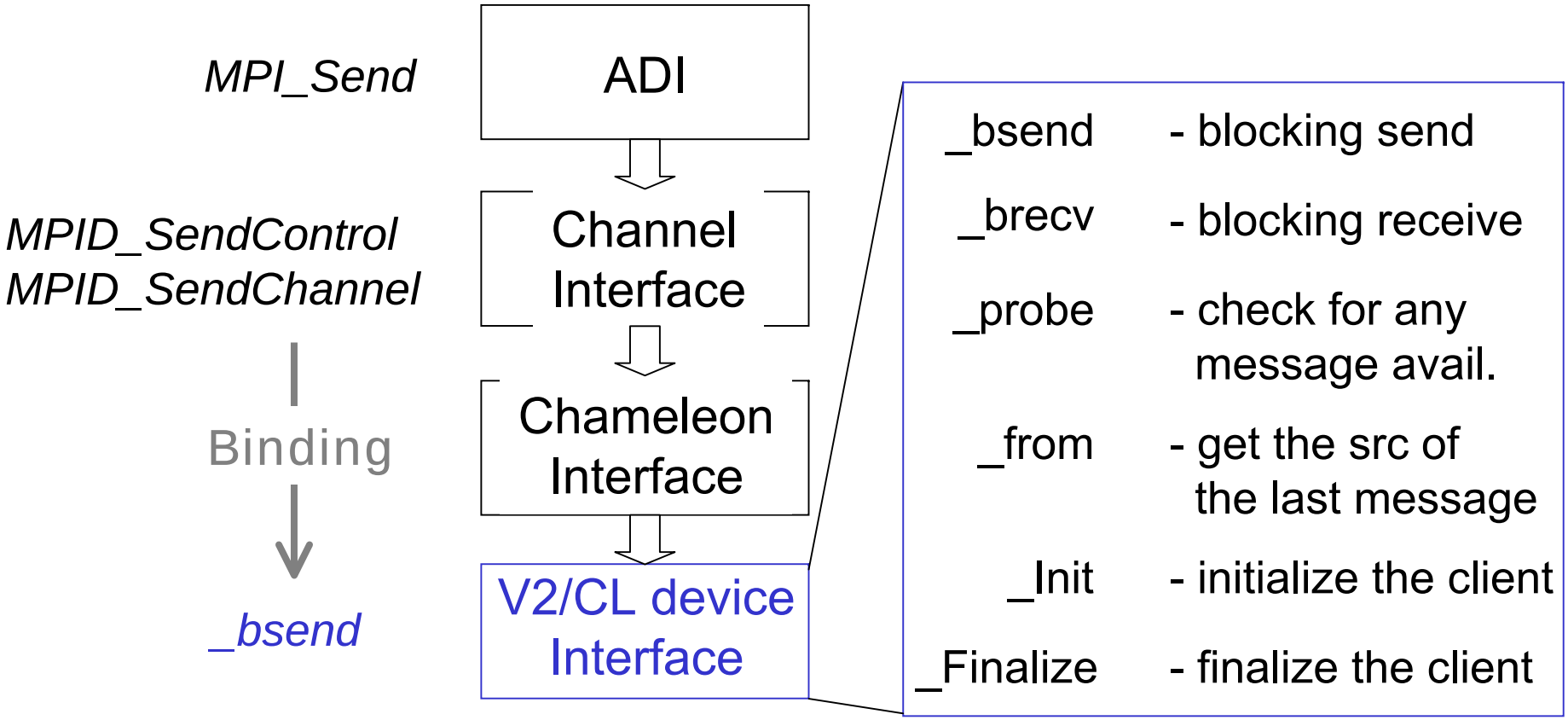
CL and V2 share the same architecture

communication daemon includes protocol specific actions

Generic device: based on MPICH-1.2.5

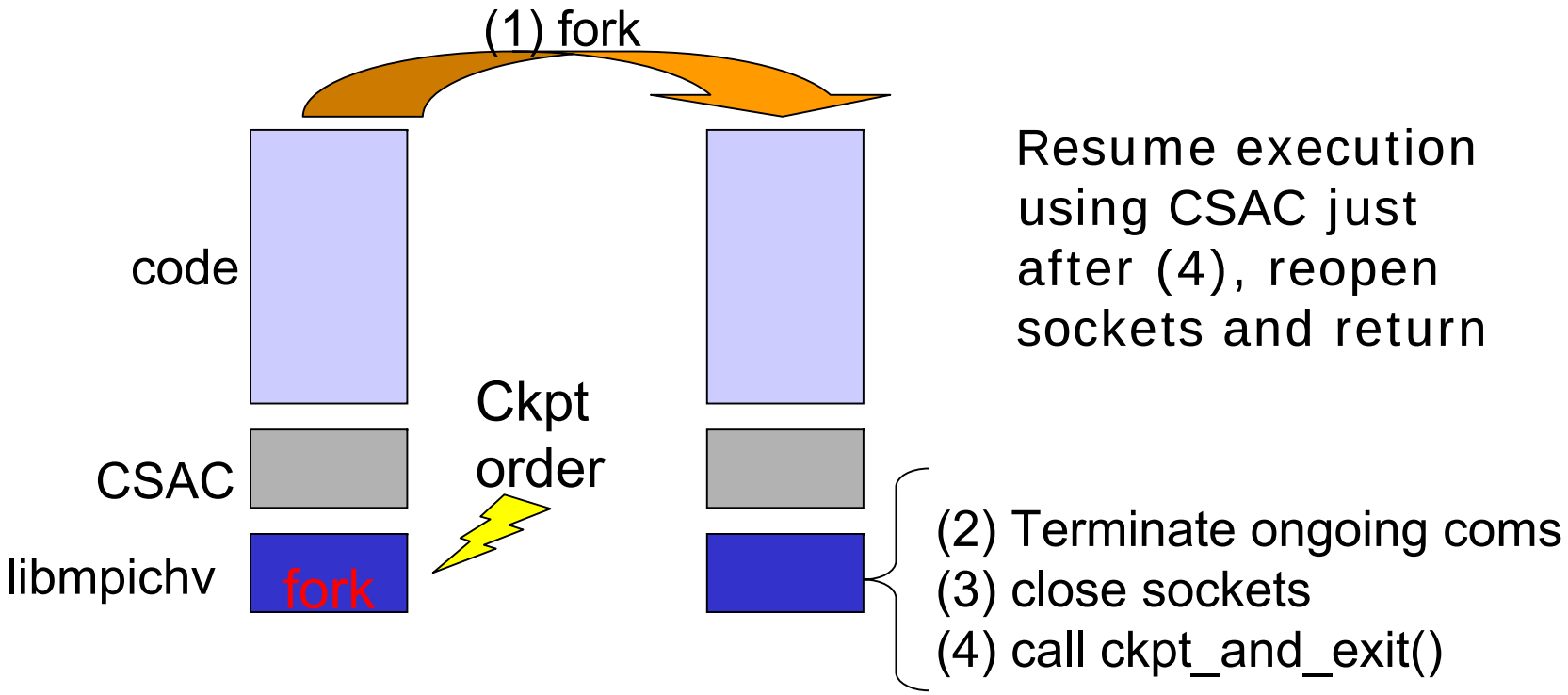
– A new device: ‘ch_v2’ or ‘ch_cl’ device

– All ch_xx device functions are blocking communication
functions built over TCP layer



Node Checkpointing

- User-level Checkpoint : Condor Stand Alone Checkpointing
- Clone checkpointing + non blocking checkpoint



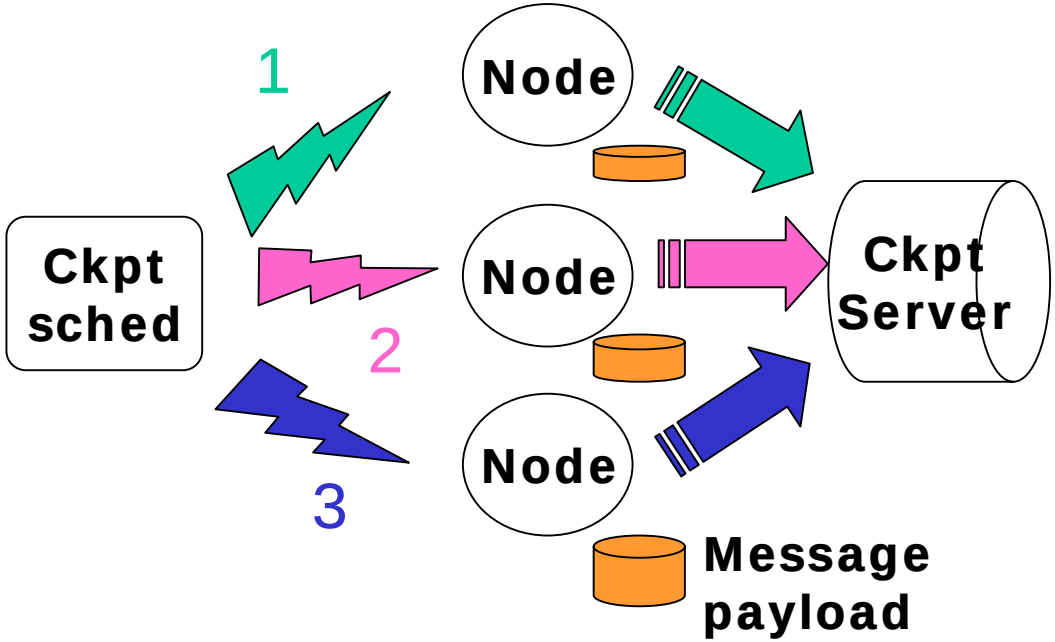
- Checkpoint image is sent to reliable CS on the fly

  →Local storage does not ensure fault tolerance

Checkpoint scheduler policy

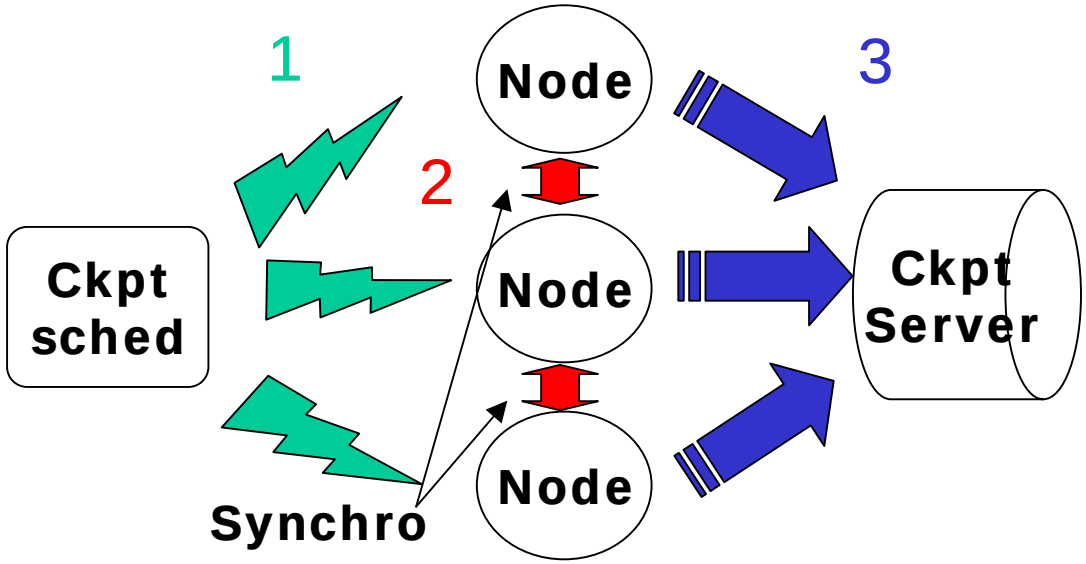
- In MPICH-V2, checkpoint scheduler is not required by pessimistic protocol. It is used to minimize size of checkpointed payload using a best effort heuristic.

  Policy : permanent individual checkpoint



- In MPICH-CL, checkpoint scheduler is used as a dedicaded process to initiate checkpoint.

  Policy : checkpoint every n seconds where n is a runtime parameter

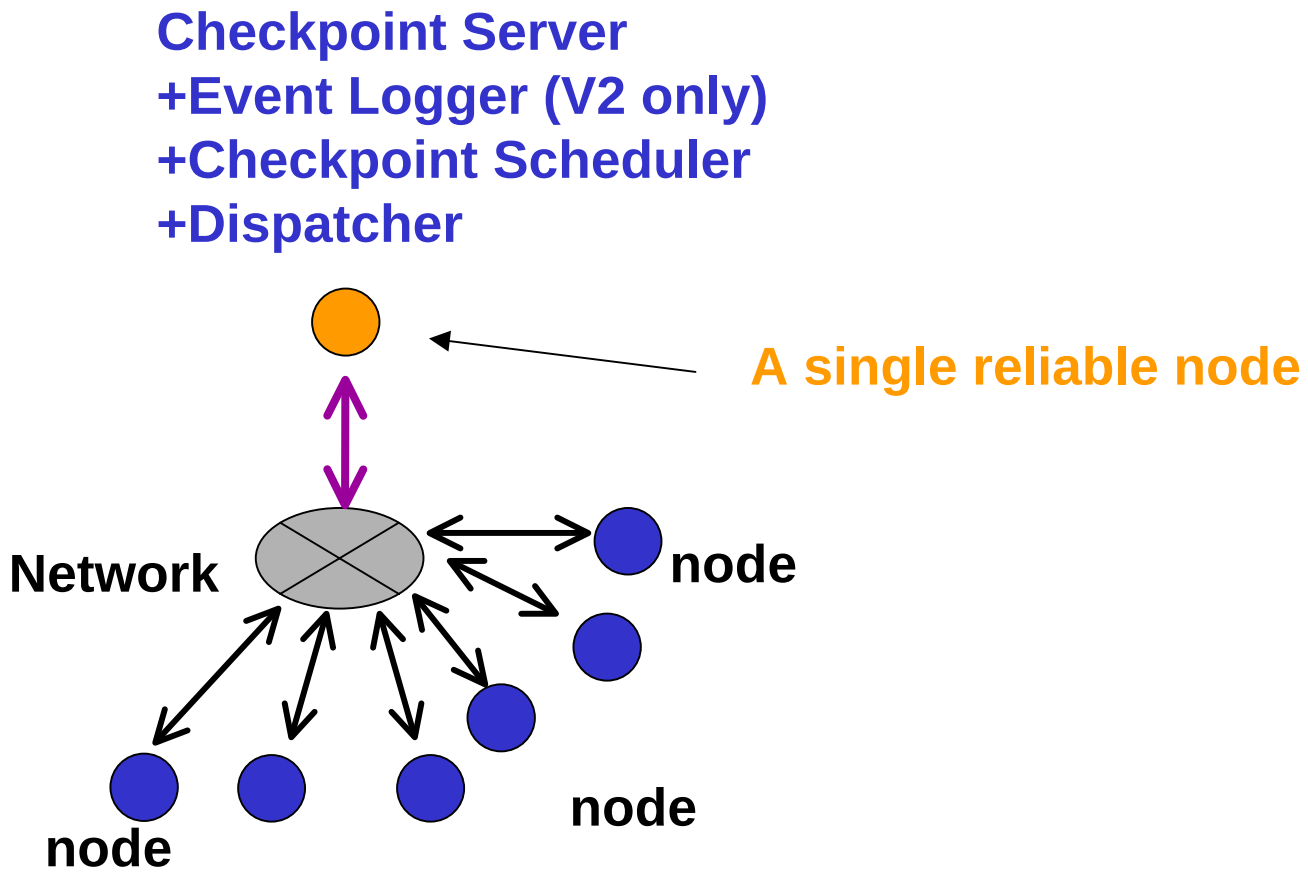


Outline

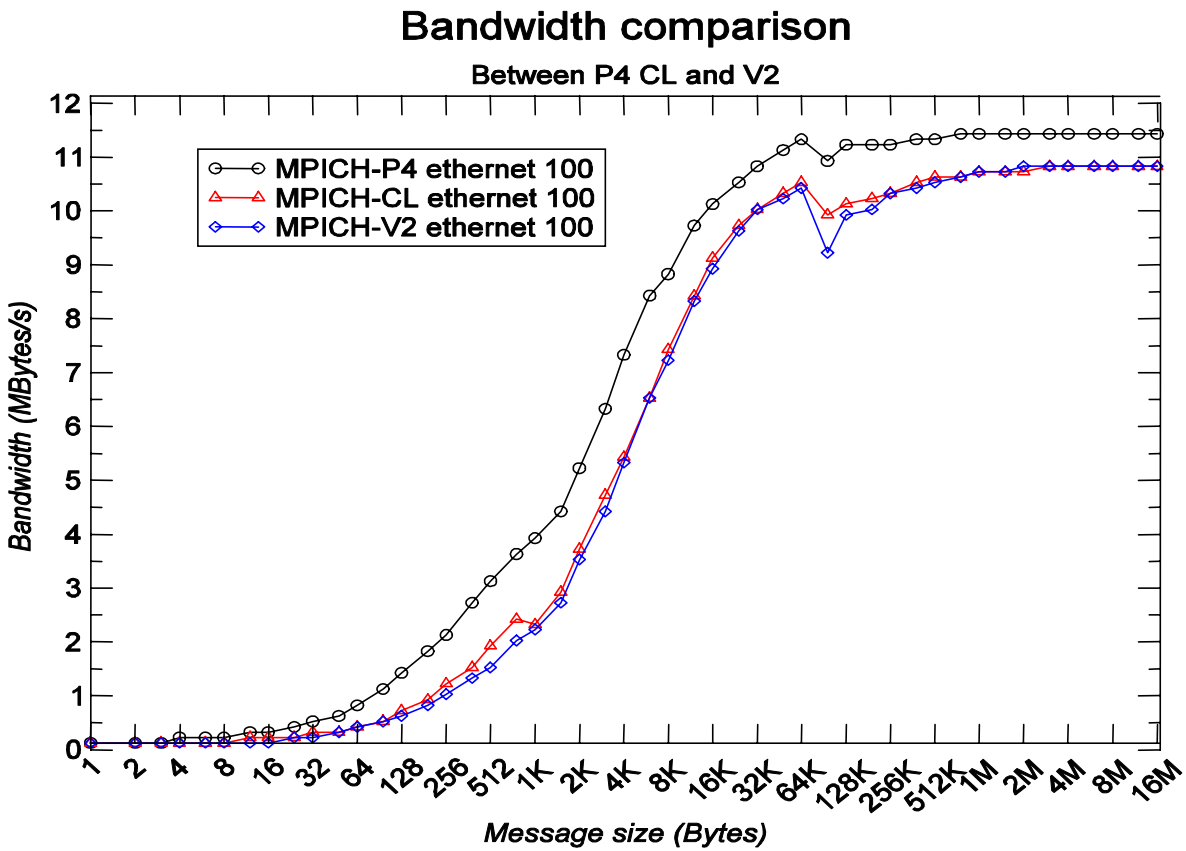
- Introduction
- Coordinated checkpoint vs Message log
- Comparison framework
- **Performances**
- Conclusions and future works

Experimental conditions

Cluster: 32 1800+ Athlon CPU, 1 GB, IDE Disc
+ 16 Dual Pentium III, 500 Mhz, 512 MB, IDE Disc
+ 48 ports 100Mb/s Ethernet switch
Linux 2.4.18, GCC 2.96 (-O3), PGI Frotran <5 (-O3, -tp=athlonxp)



Bandwith and latency



Latency for a 0 byte MPI message :
MPICH-P4 (77us),
MPICH-CL (154us),
MPICH-V2 (277us)

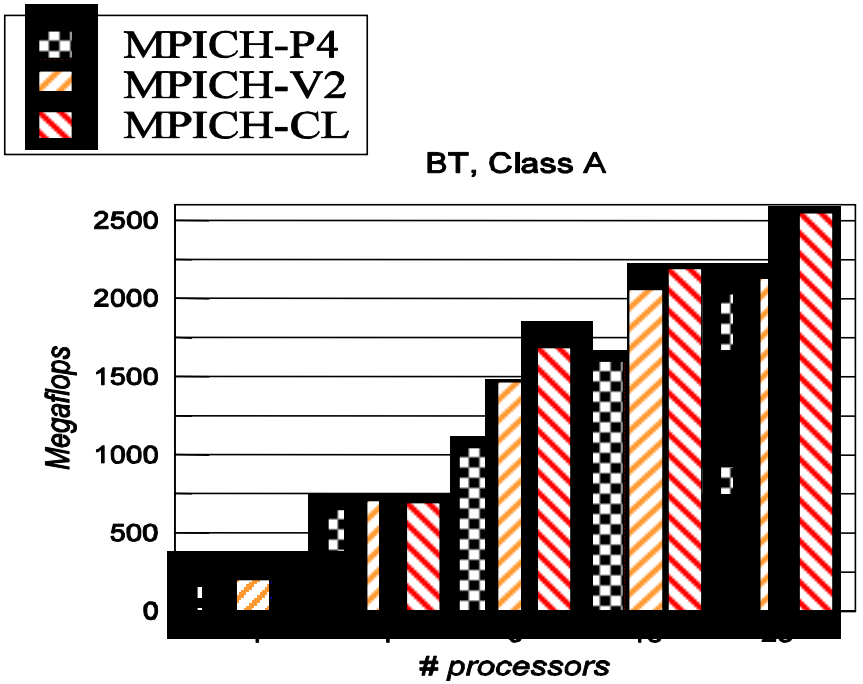
Latency is high in MPICH-CL due to more memory copies compared to P4
Latency is even higher in MPICH-V2 due to the event logging.

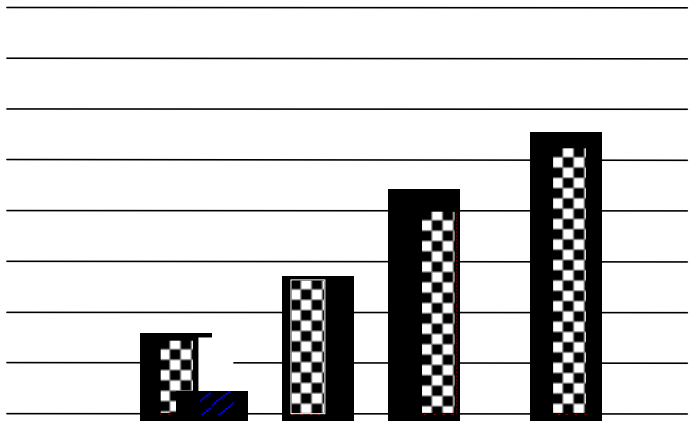
→ A receiving process can send a new message only when the reception event has been successfully logged (3 TCP messages for a communication)

Benchmark applications

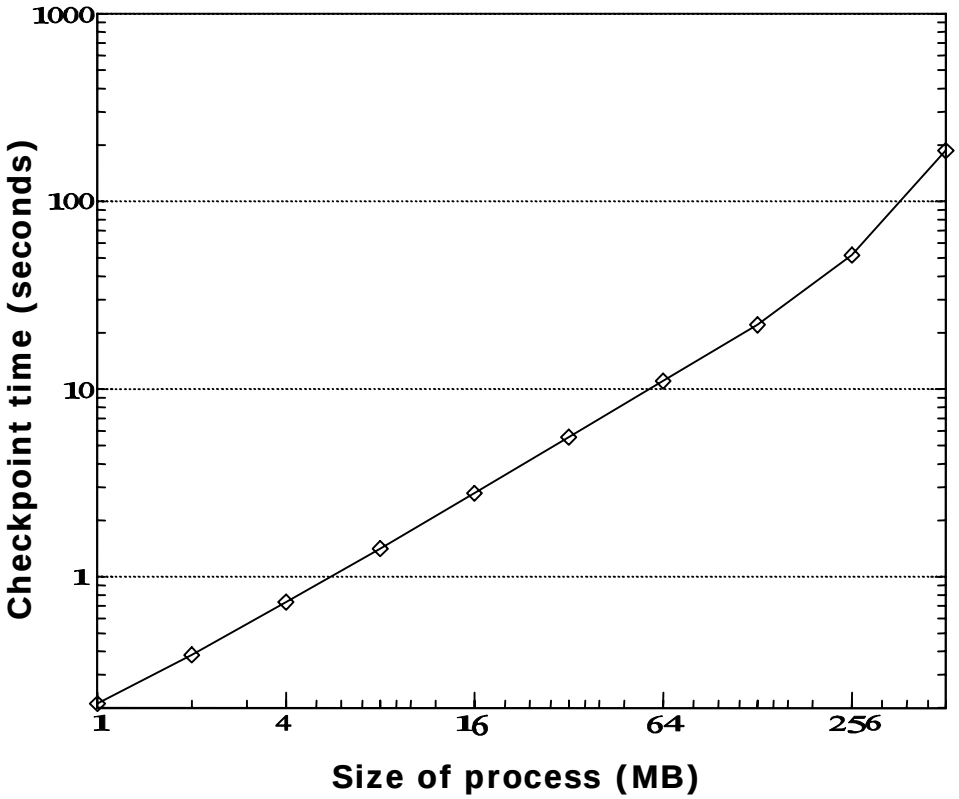
Validating our implementations on NAS BT Benchmark class A and B shows comparable performances to P4 reference implementation.

As expected MPICH-CL reaches better fault free performances than MPICH-V2





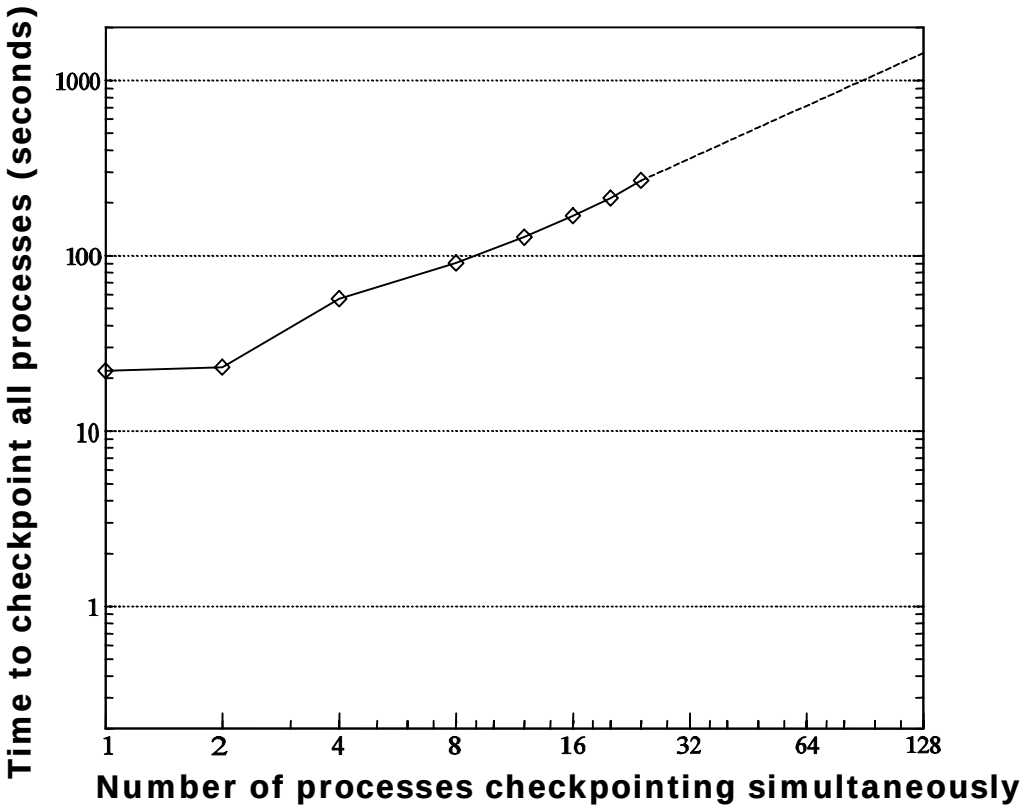
Checkpoint server Performance



Time to checkpoint a process according to its size.

Checkpoint time increases linearly with checkpoint size.

Memory swap overhead appears at 512MB (fork).

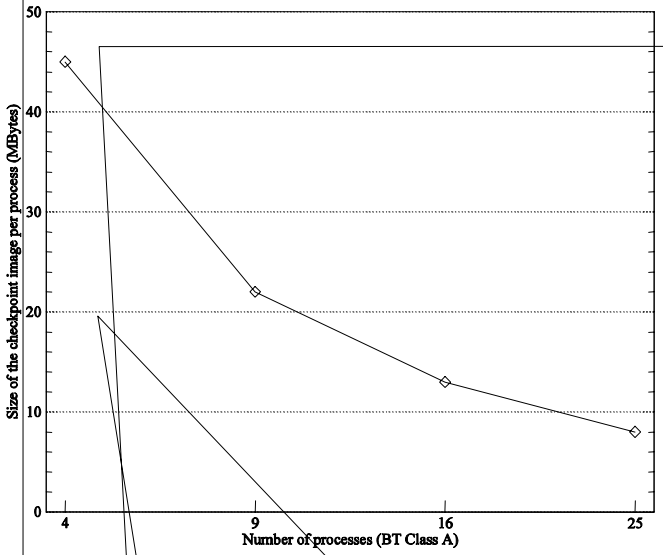


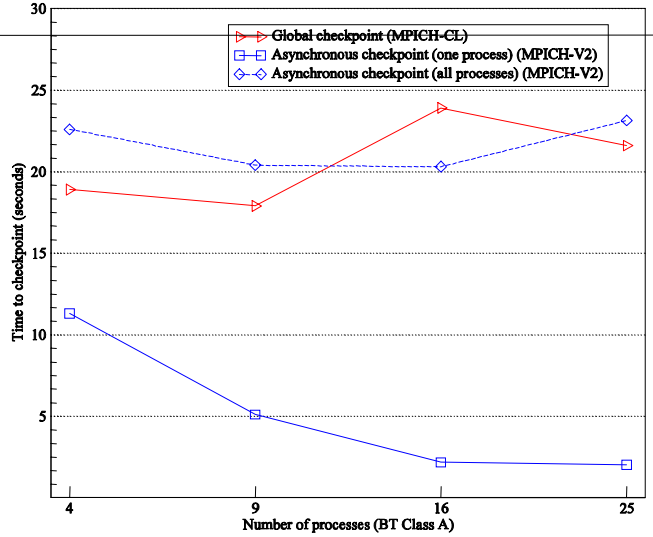
Time to checkpoint all processes concurently on a single checkpoint server.

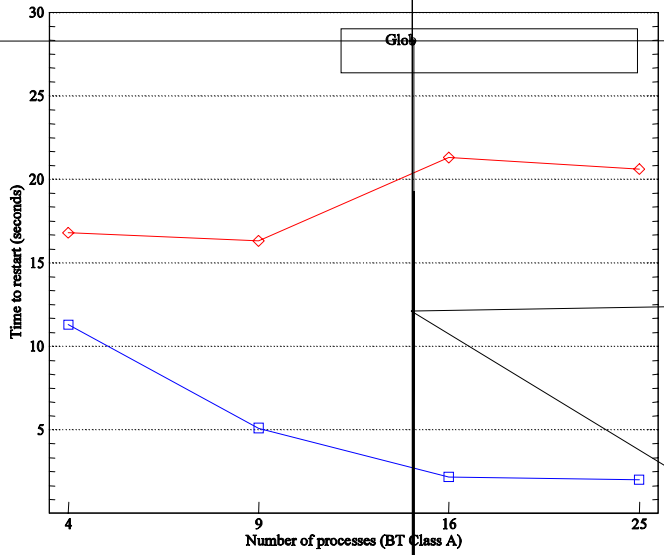
2nd process does not increase checkpoint time, filling unused bandwith.

More processes increase checkpoint time linearly.

BT Checkpoint and Restart Performance



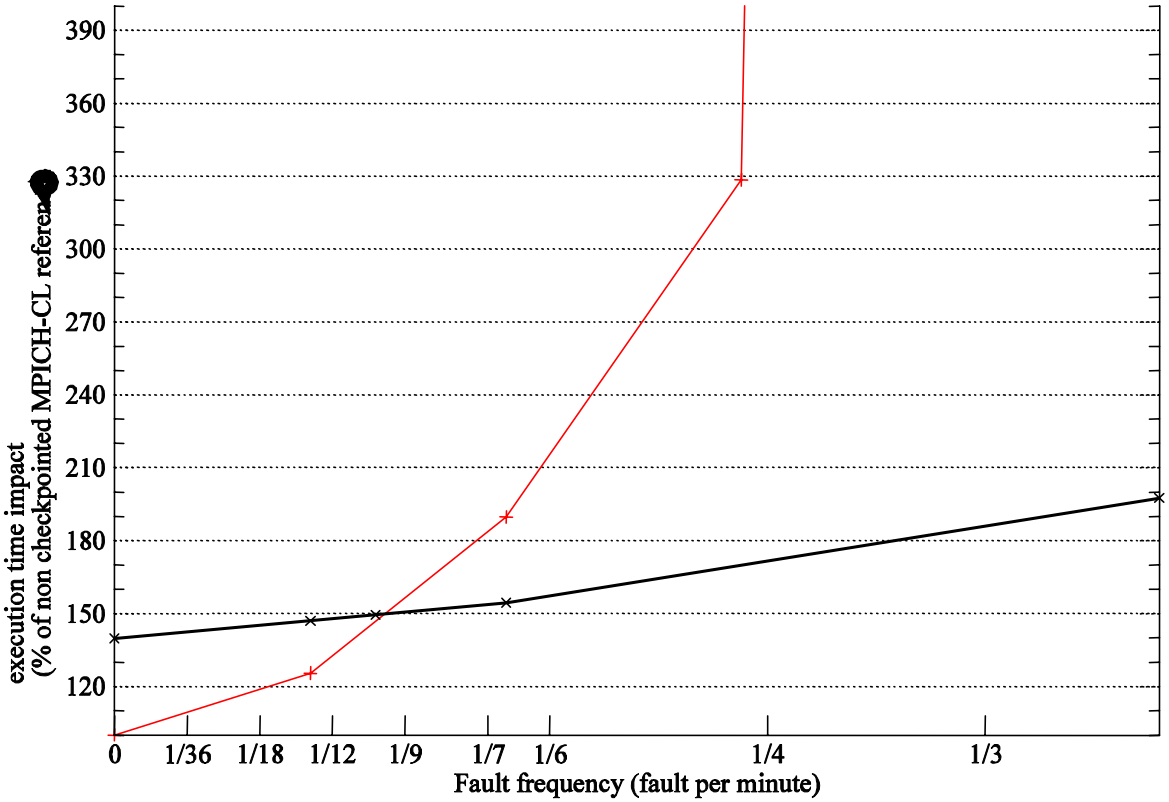




- Considerating the same dataset, per process image size decreases when number of processes increases
- As a consequence time to checkpoint remains constant with increasing number of processes
- Performing a complete asynchronous checkpoint takes as much time as coordinated checkpoint
- Time to restart after a fault is decreasing with the number of nodes for V2 and not changing for CL

Fault impact on performances

- NAS Benchmark BT B 25 nodes (32MB per process image size)
- Average time to perform Checkpoint
  - MPICH-CL : 68s
  - MPICH-V2 : 73.9s
- Average time to recover from failure
  - MPICH-CL : 65.8s
  - MPICH-V2 : 5.3s



Outline

- Introduction
- Coordinated checkpoint vs Message log
- Comparison framework
- Performances
- **Conclusions and future works**

Conclusion

- MPICH-CL and MPICH-V2 are two comparable implementations of fault tolerant MPI from the MPICH-1.2.5, one using coordinated checkpoint, the other pessimistic message log
- We have compared the overhead of these two techniques according to fault frequency
- The recovery overhead is the main factor differentiating performances
- We have found a crosspoint from which message log becomes better than coordinated checkpoint. On our test application this cross point appears near 1 per 10 minutes. With 1GB application, coordinated checkpoint does not ensure progress of the computation for one fault every hour.

Perspectives

- Larger scale experiments
- Use more nodes and applications with realistic amount of memory
- High performance networks experiments

  Myrinet – Infiniband
- Comparison with causal log