MPICH-V

MPI Implementation for Volatile resources

Grand Large

INRIA

Tolérance aux fautes, impossibilités et solutions

Aurélien BOUTEILLER (aurelien.bouteiller@lri.fr)
joint work with
F.Cappello, P.Lemarinier, T. Herault

Grand Large Project
http://www.mpich-v.net

Cluster & Grid Group at LRI

Calcul Global et Pair à Pair

ACI GRID www.cgp2p.net CGP2P

Globalisation des Ressources Informatiques et des Données

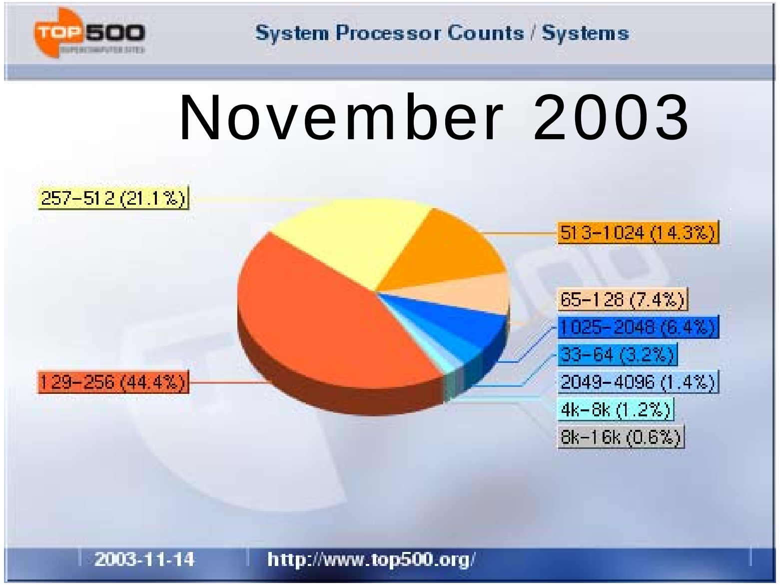
Fault tolerance: Why?

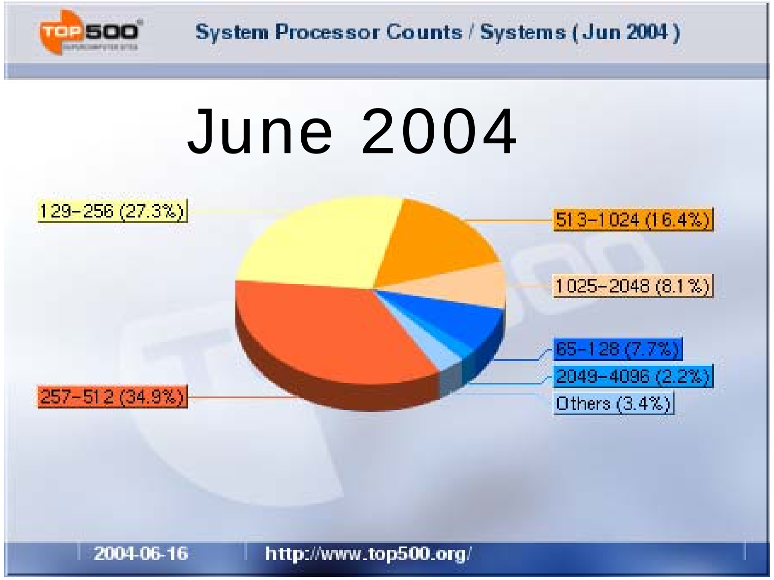
- Current trend: increase of the number of processor

  More components increases fault probability

- Many numerical applications uses MPI (Message Passing Interface) library

  Need for automatic fault tolerant MPI





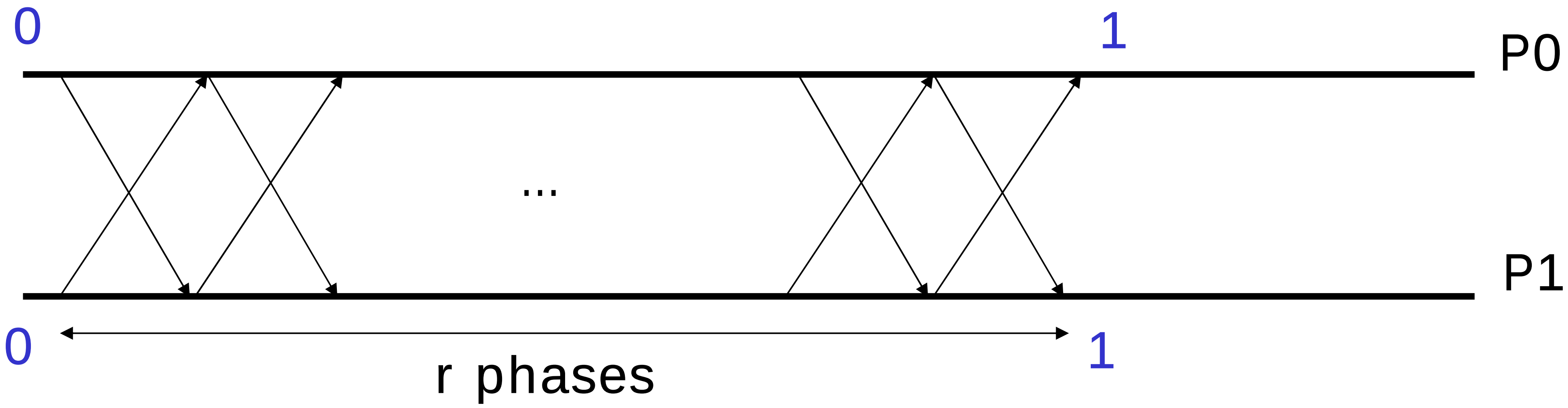
Modèles

- Modèle synchrone
  - Les processus réalisent une étape de calcul de façon synchronisée.
  - A la fin de la phase, tous les messages envoyés ont été reçus
  - Types de fautes : panne de processus, panne de réseau (un message est supprimé, modifié ou retardé arbitrairement longtemps)
- Modèle asynchrone
  - Pas de borne sur la durée d'une étape d'exécution
  - pas d'horloge globale
  - un message peut transiter un temps arbitrairement long dans un canal
  - Types de fautes : panne de processus, panne de réseau (un message est supprimé ou modifié)

Problème du concensus

Synchrone avec pannes réseau

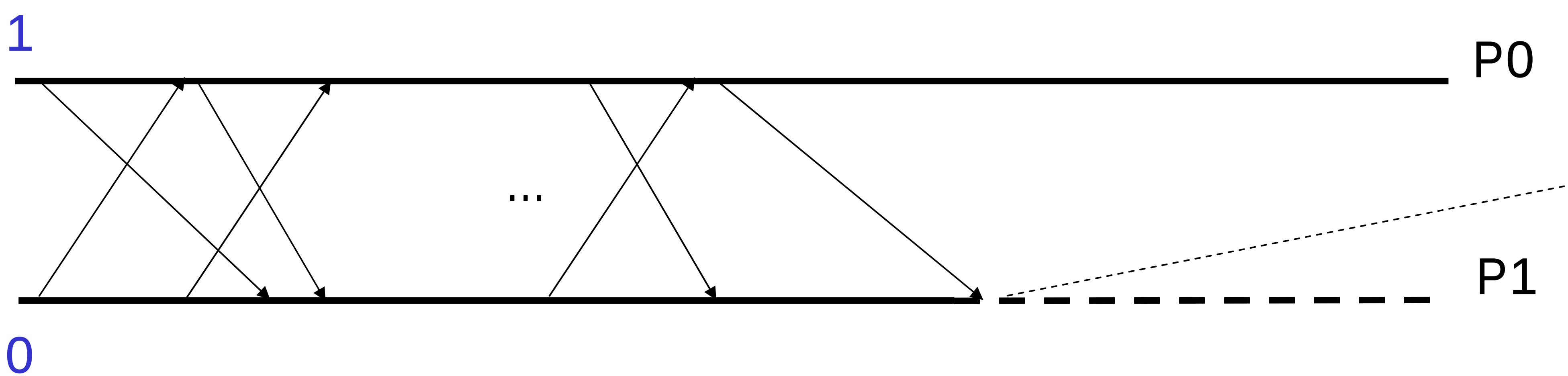
- Les généraux veulent coordonner leur attaque
  - Chaque jour, ils envoient un messager qui informe les autres généraux de leur intention (attaquer ou non)
  - Lorsque tous les généraux souhaitent attaquer, ils attaquent
  - Problème : Les messagers peuvent se faire tuer par l'enemi!
- 3 conditions :
  - Terminaison : Les généraux décident en un temps fini
  - Agrément : les généraux décident tous la même valeur
  - Validité : si initialement, tous les généraux veulent attaquer, ils attaquent. Si initialement aucun général ne veut attaquer, ils n'attaquent pas



Problème du concensus

Asynchrone avec panne crash

- Concensus asynchrone
  - Terminaison : Les processus décident en un temps fini
  - Agrément : tous les processus non défaillants décident d'une même valeur
  - Validité : Si les processus partagent tous une même valeur initiale, ils décident de cette valeur
  - Type de fautes : des processus peuvent s'arrêter totalement
- **Fischer, Lynch, Paterson, Impossibility of distributed concensus with one faulty process (journal of ACM 32(2), April 1985)**



Tolérance aux fautes : Mission impossible ?

- Ignorer le problème : l'expérience du projet Isis
  - Group Membership service sur internet
  - Lorsqu'un processus détecte qu'il a été suspecté, il se suicide
  - Timeout de détection de faute augmenté jusqu'à 20 minutes et plus
  - Fréquents scénarios de suicides collectifs…
- Affaiblir le modèle : pseudosynchronisme du réseau (MPICH-V)
- Chercher à faire quelque chose de moins difficile : RPC stateless (RPC-V)

Pseudosynchronisme

- Modèle synchrone
  - Les processus réalisent une étape de calcul de façon synchronisée (Horloge globale).
  - A la fin de la phase, tous les messages envoyés ont été reçus
  - *On peut résoudre le problème*, mais ne correspond pas au monde réel.
- Modèle asynchrone
  - pas d'horloge globale
  - un message peut transiter un temps arbitrairement long dans un canal
  - Modèle très expressif (Internet), mais on ne peut rien faire!
- **Modèle(s) Pseudosynchrone(s)**
  - **Pas d'horloge globale**
  - **Il existe une borne sur le temps de transit d'un message dans un canal**
  - **Equivalent avec l'existence d'un détecteur de défaillances (appelé Oracle) (Chen, Toueg, Aguilera: On the QoS of failure detectors, proc. Of ICDSN/FTCS-30, June 2000)**
  - **Représente bien les réseaux de diamètre connu constitué de compostants temps réels (typiquement LAN/Clusters), permet de résoudre le problème! (Chen, Toueg: Unreliable failure detectors for reliable distributed systmes, Journal of the ACM 43(2) march 1996)**

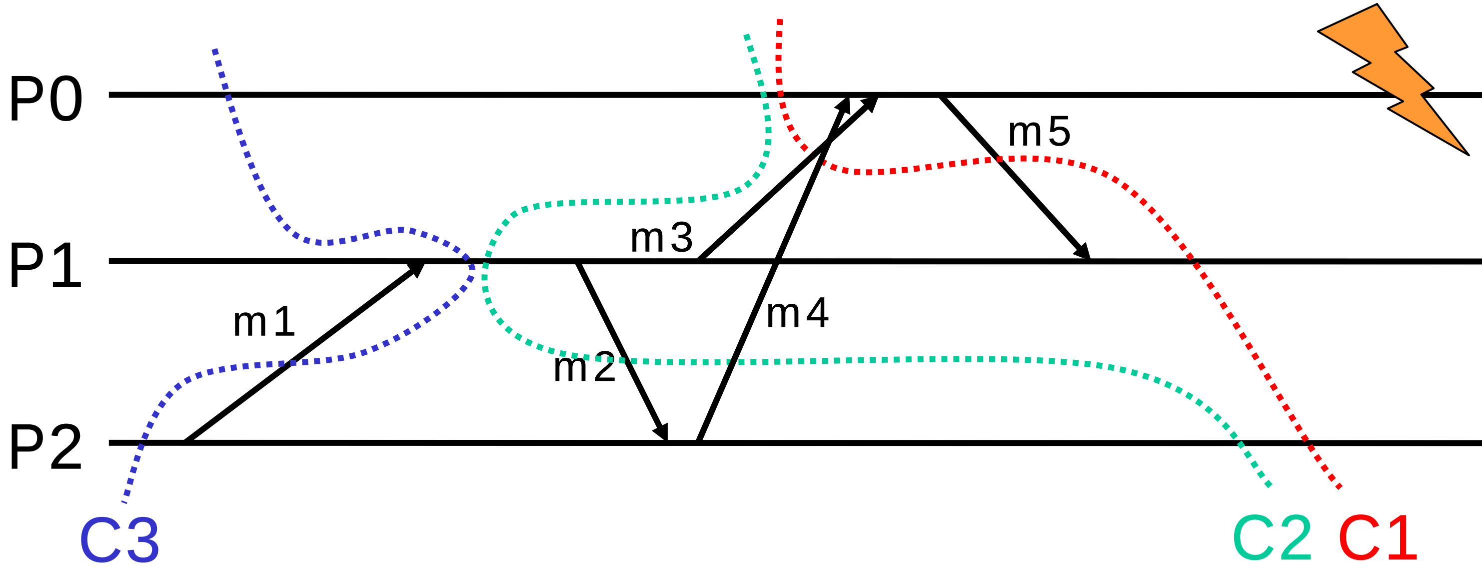
Outline

- **Protocols and Related works**
- MPICH-V Comparison framework
- Performance
- OpenMPI-V
- Conclusion and future works

Fault Tolerant protocols
Problem of inconsistent states

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

Possible loose of the whole execution and unpredictive fault cost



Fault Tolerant protocols
Global Checkpoint

- 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 received by a process are logged on a reliable media before it can causally influence the rest of the system

    Non negligible overhead on network performance in fault free execution : send may be delayed
  - 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

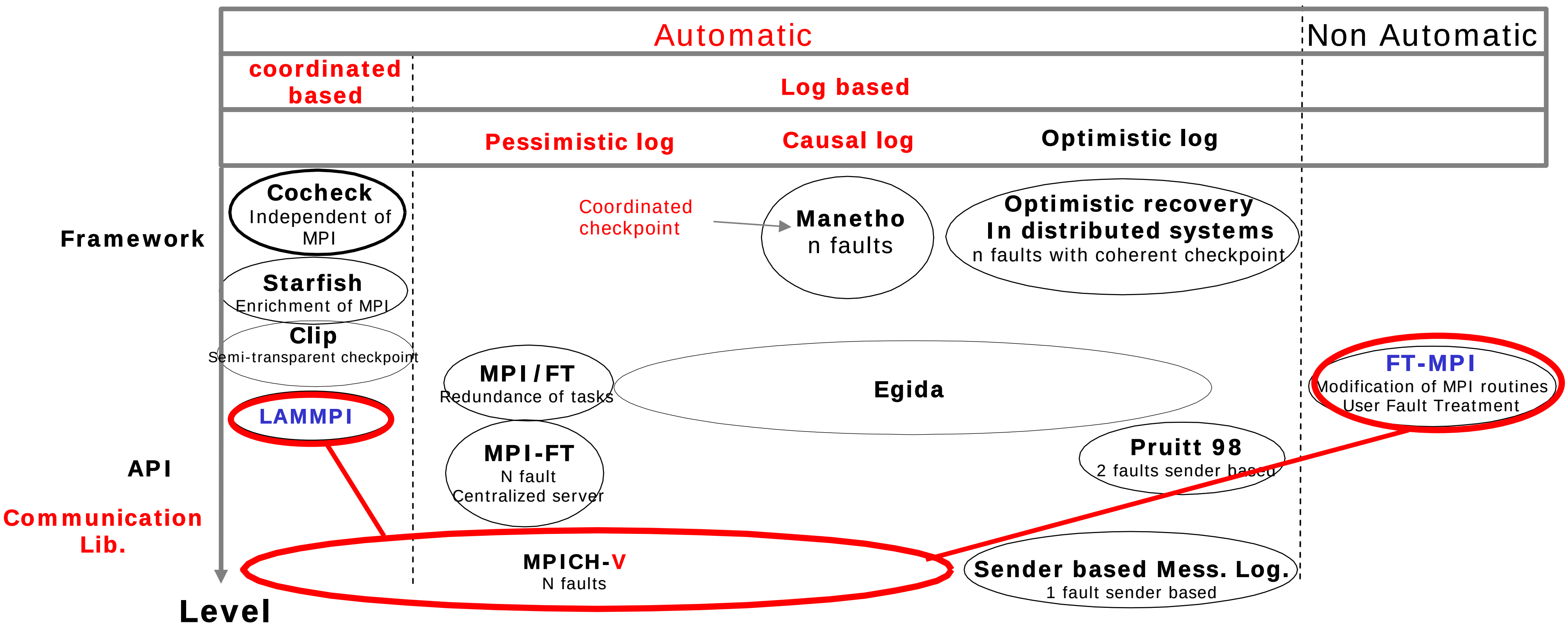
    Non negligible overhead on fault free execution, piggyback is added to messages
  - No global synchronisation

    Does not stress checkpoint server
  - Only failed processes 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

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

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**
  Siram 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

- First comparison of coordinated checkpointing and message logging realized last year (Cluster 2003)

→high fault recovery overhead of coordinated checkpoint

→high overhead of message logging on fault free performance

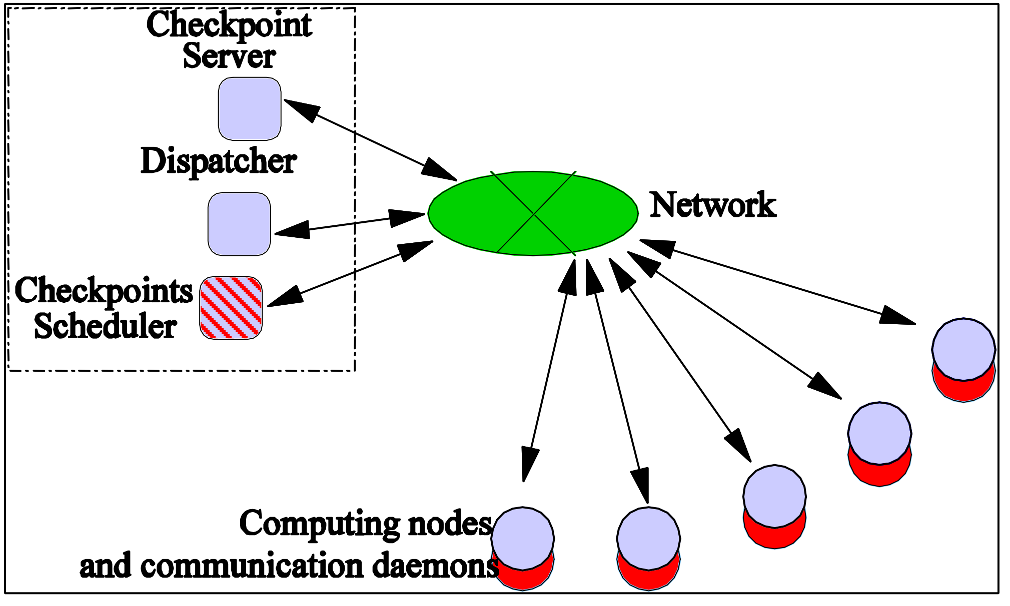
⇨ fault frequency implies tradeoff

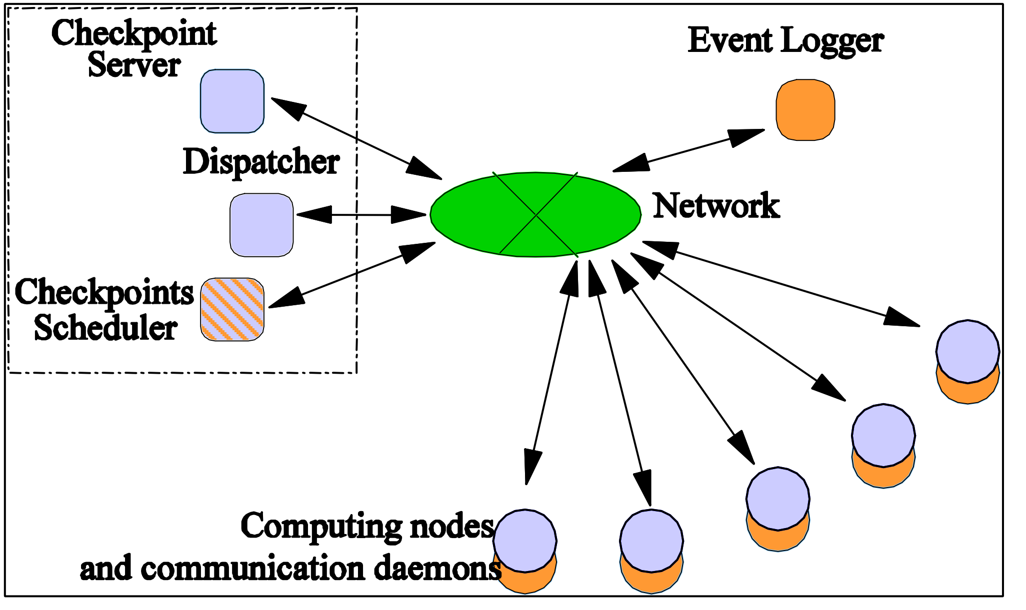
Outline

- Protocols and Related works
- **MPICH-V Comparison framework**
- Performance
- OpenMPI-V
- Conclusion and future works

Architectures

We designed MPICH-V to perform a fair comparison of coordinated checkpoint and pessimistic message log



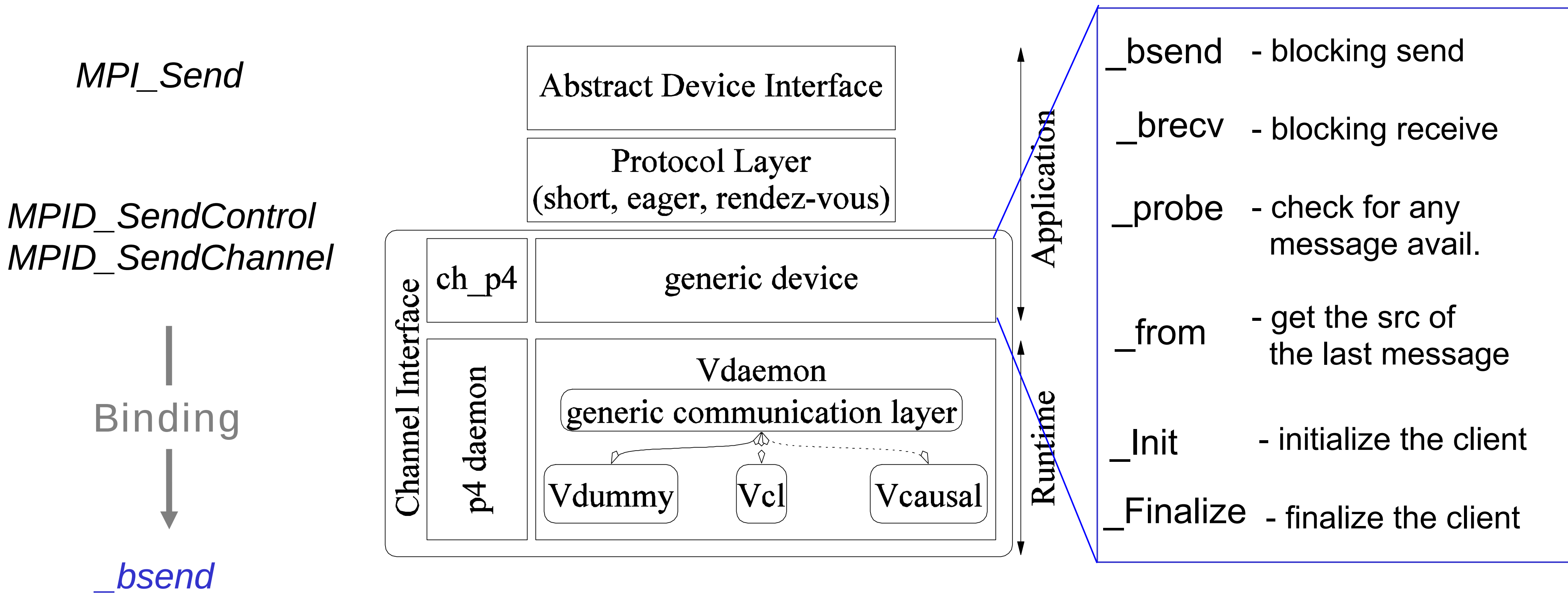


MPICH-Vcl
Chandy&Lamport algorithm
Coordinated checkpoint

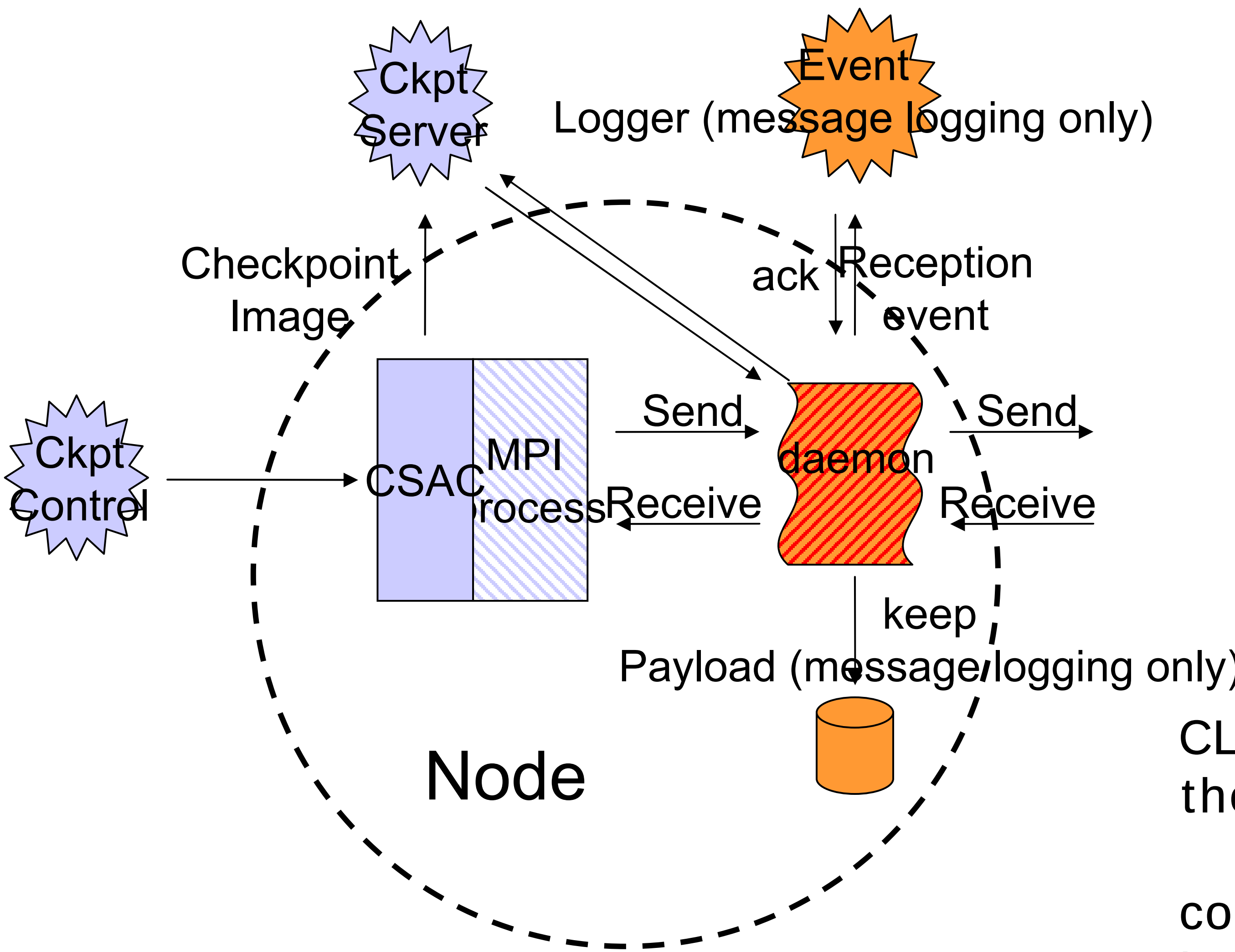
MPICH-V for message logging protocols

Generic device: based on MPICH-1.2.5

- A new device: ‘ch_v’ device
- All ch_v device functions are blocking communication functions built over TCP layer



Communication daemon

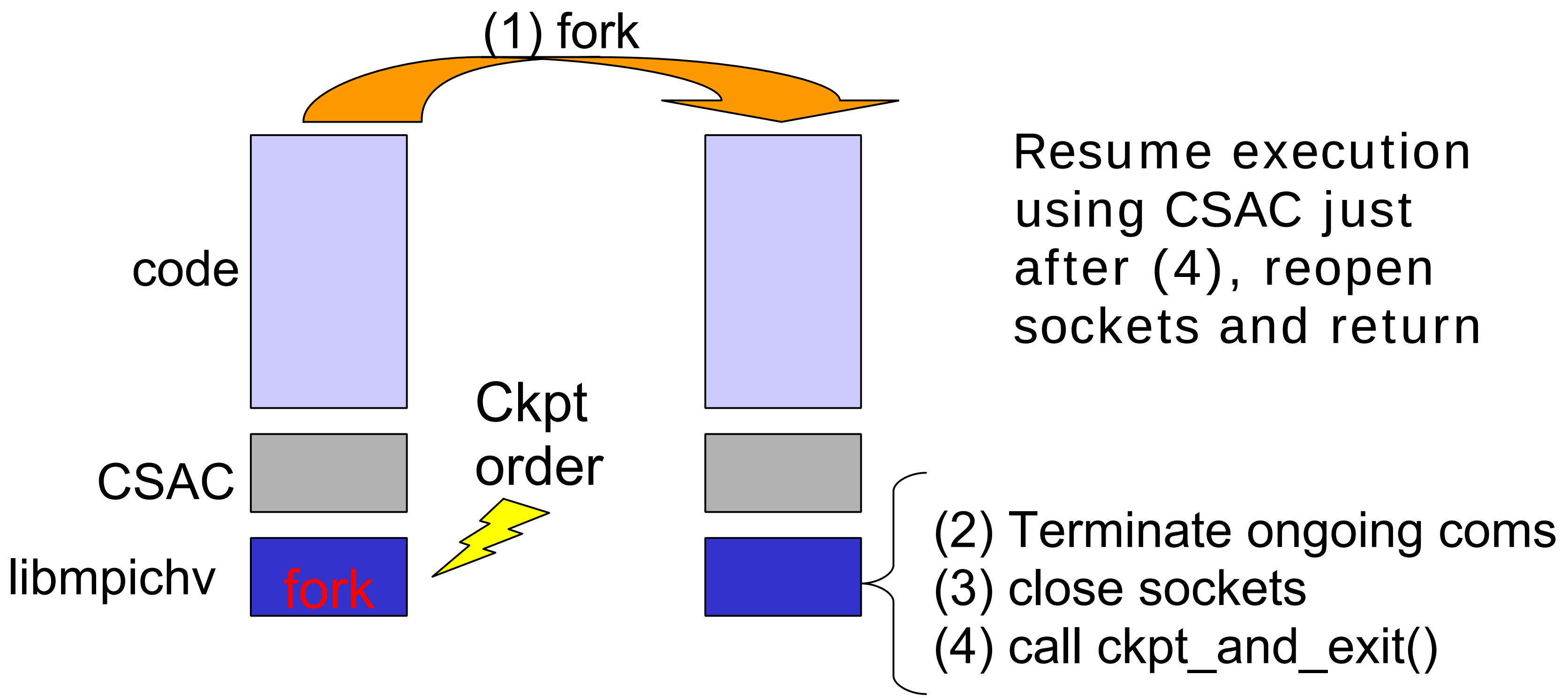


CL and V2/Vcausal share the same architecture

communication daemon includes protocol specific actions

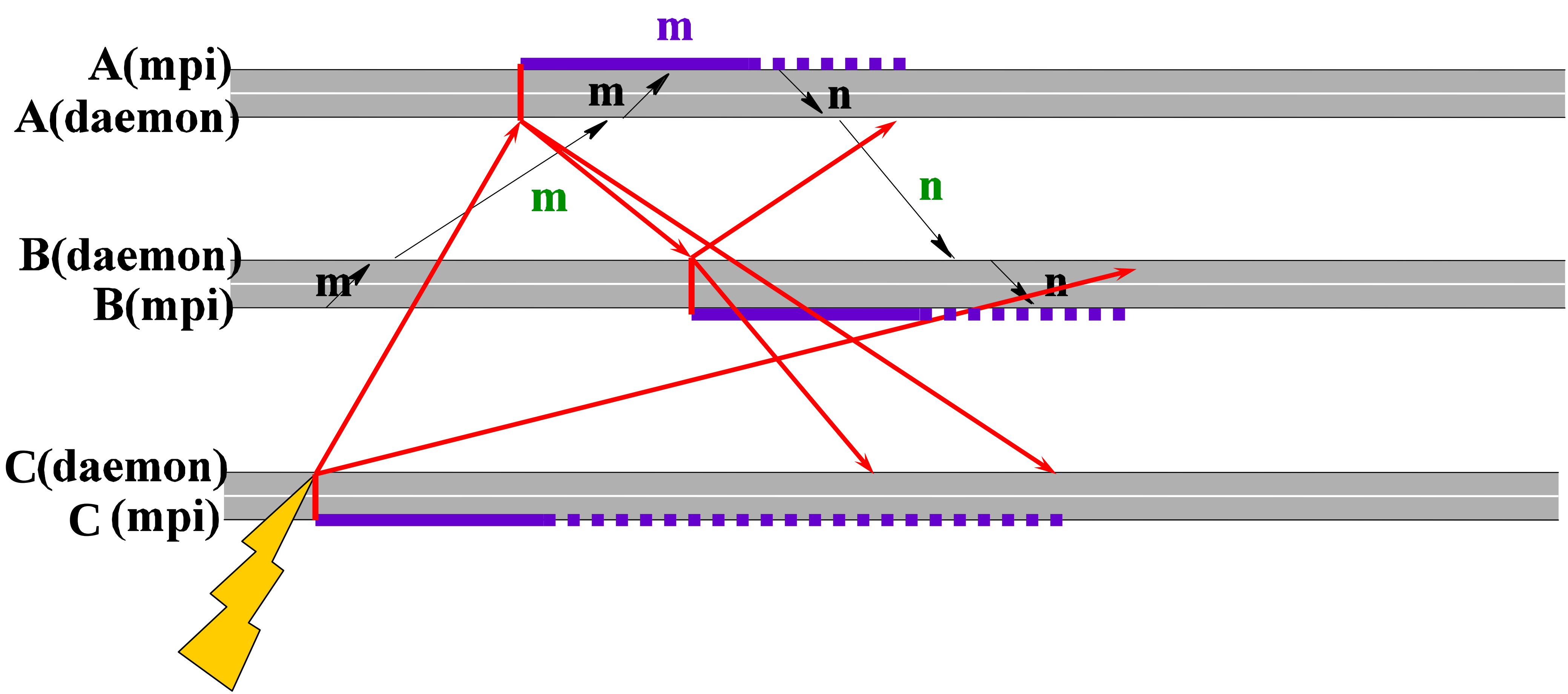
Checkpointing method

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

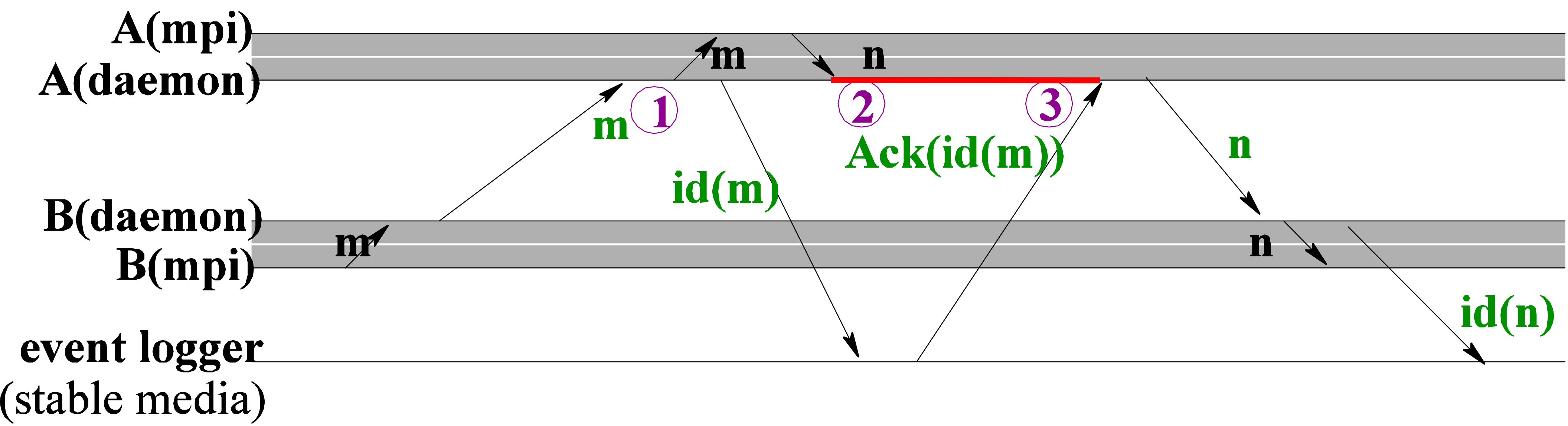


- Checkpoint image is sent to reliable CS on the fly

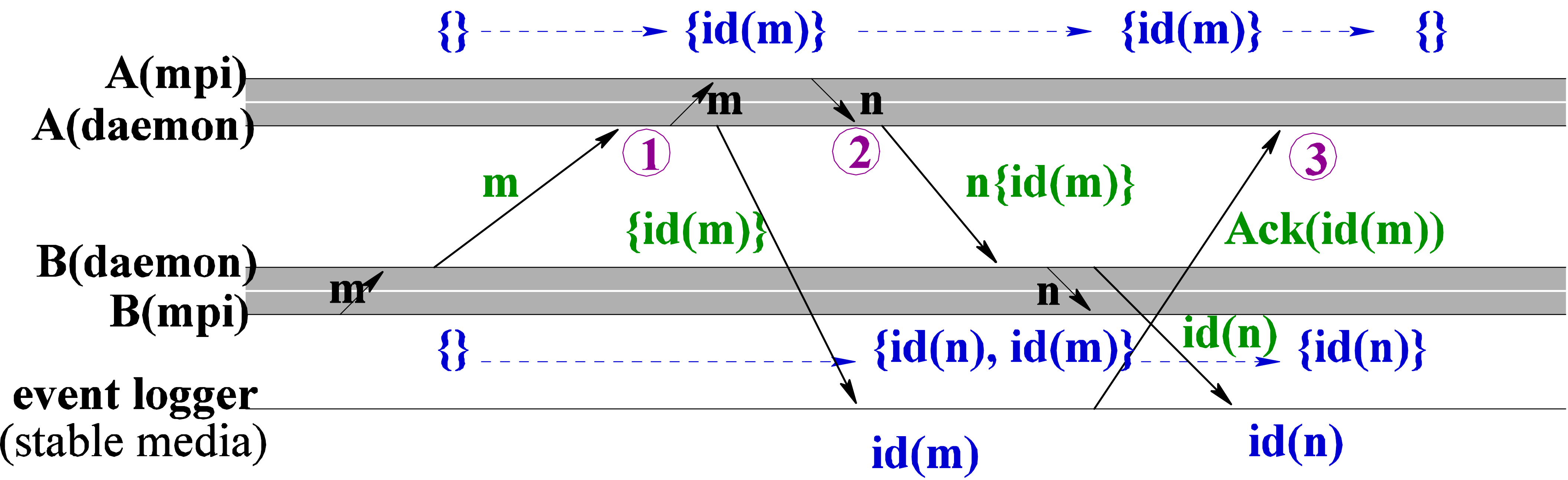
Coordinated checkpoint example



Pessimistic message logging example



Causal message logging

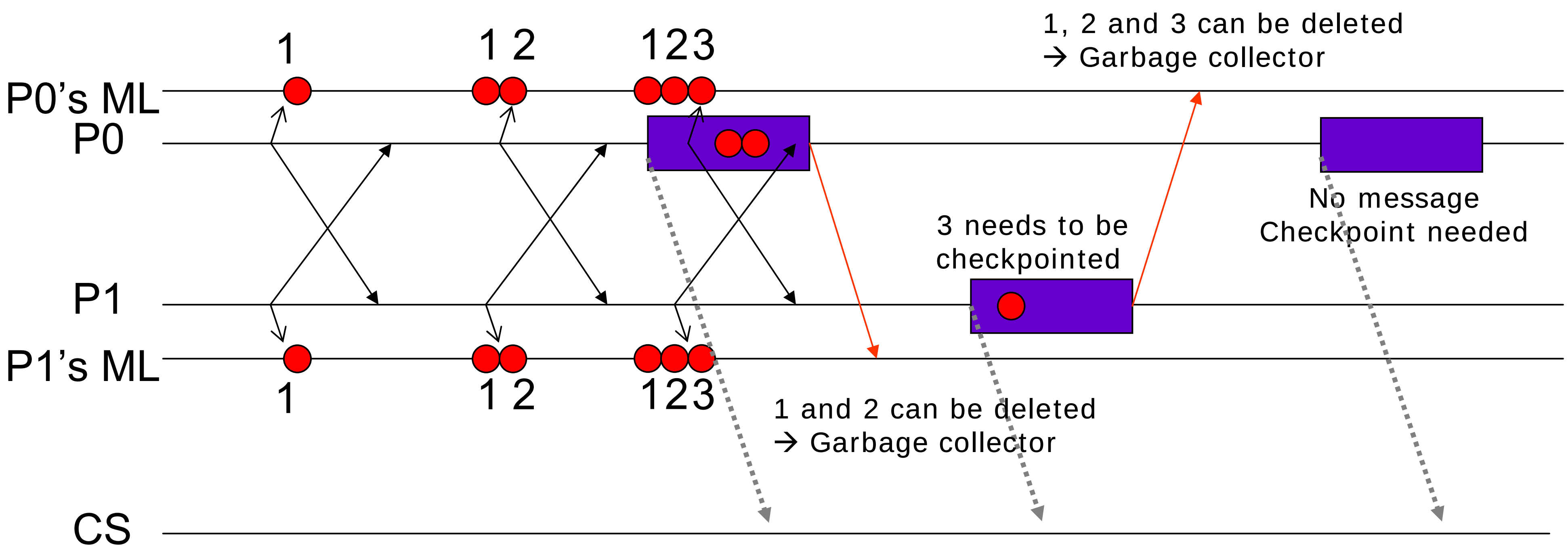


{} : log to piggyback to all messages

n{} message and is piggyback

Scheduling Checkpoint

- Uncoordinated checkpoint lead to log in-transit messages
- Scheduling checkpoint simultaneously will lead to bursts in the network traffic.
- Checkpoint size can be reduced by removing message logs
  - → Coordinated checkpoint (Lamport).
    - →Requires global synchronization
- Checkpoint traffic should be flattened
- Checkpoint scheduling should evaluate the cost and benefit of each checkpoint.



Outline

- Protocols and Related works
- MPICH-V Comparison framework
- **Performance**
- OpenMPI-V
- Conclusion and future works

Experimental conditions

Ethernet experiments:

- 32 2800+ AthlonXP CPU, 1 GB DDR SDRAM, 70GB ATA100 IDE Disc
  100Mbits/s Ethernet card connected by a single Fast Ethernet Switch

Myrinet experiments:

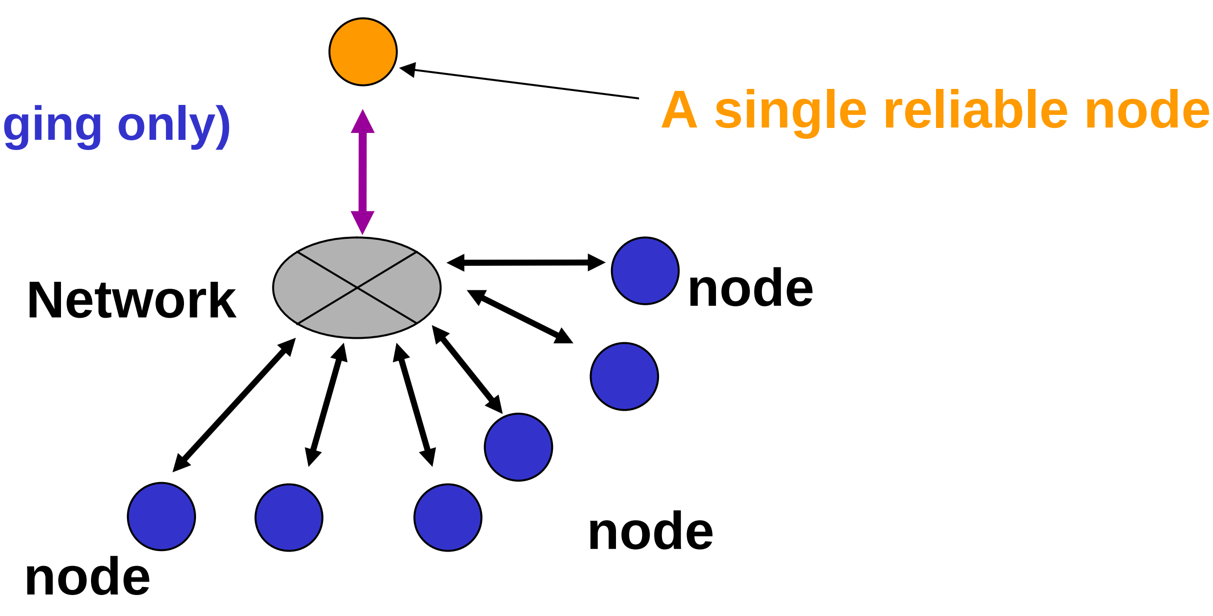
- 8 2200+ AthlonXP-MP CPU, 1 GB DDR SDRAM, 70GB ATA100 IDE Disc
  Myrinet2000 connected by a single 8-port myrinet switch

SCI experiments

- 32 2800+ AthlonXP CPU, 1 GB DDR SDRAM, 70GB ATA100 IDE Disc
  2D-torus topology SCI

Linux 2.4.20, GCC 2.96 (-O3), PGI Fortran <5 (-O3, -tp=athlonxp)

Checkpoint Server
+Event Logger (message logging only)
+Checkpoint Scheduler
+Dispatcher

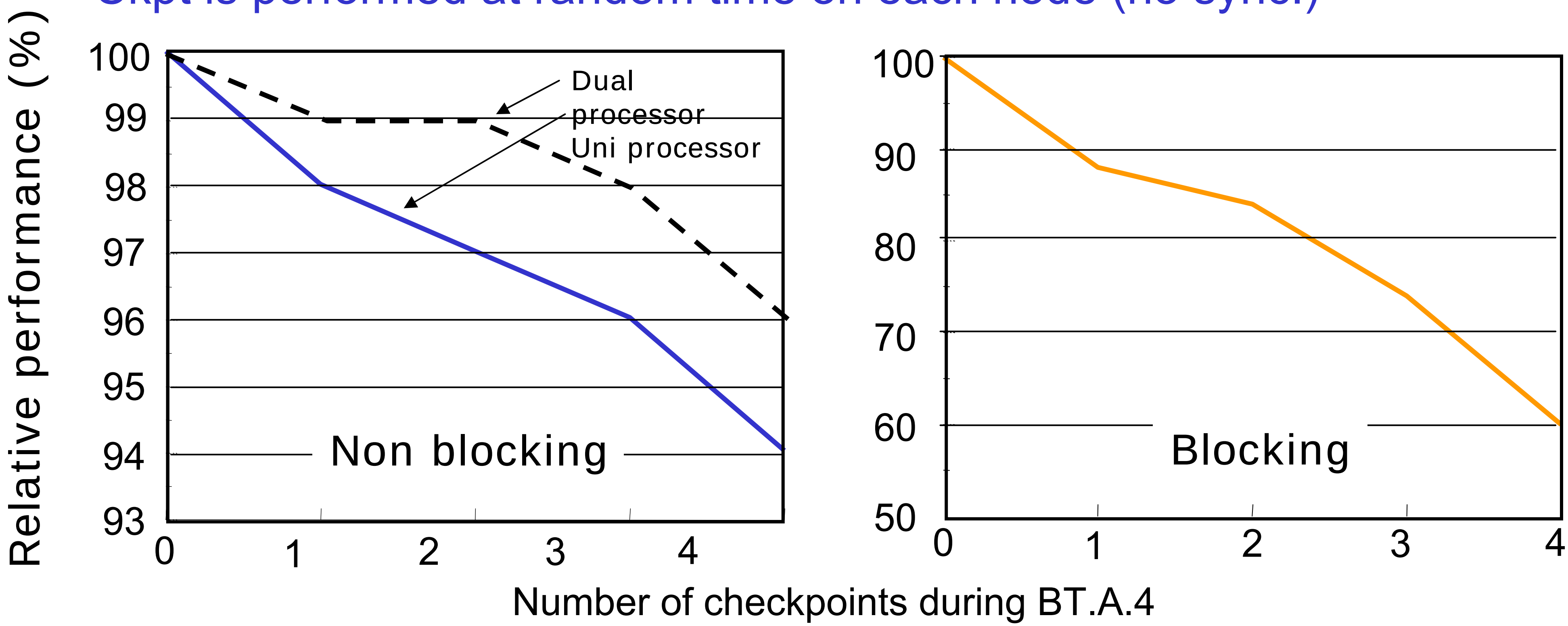


Impact of checkpointing on application performance

Performance reduction for NAS BT.A.4 according to the number of consecutive checkpoints

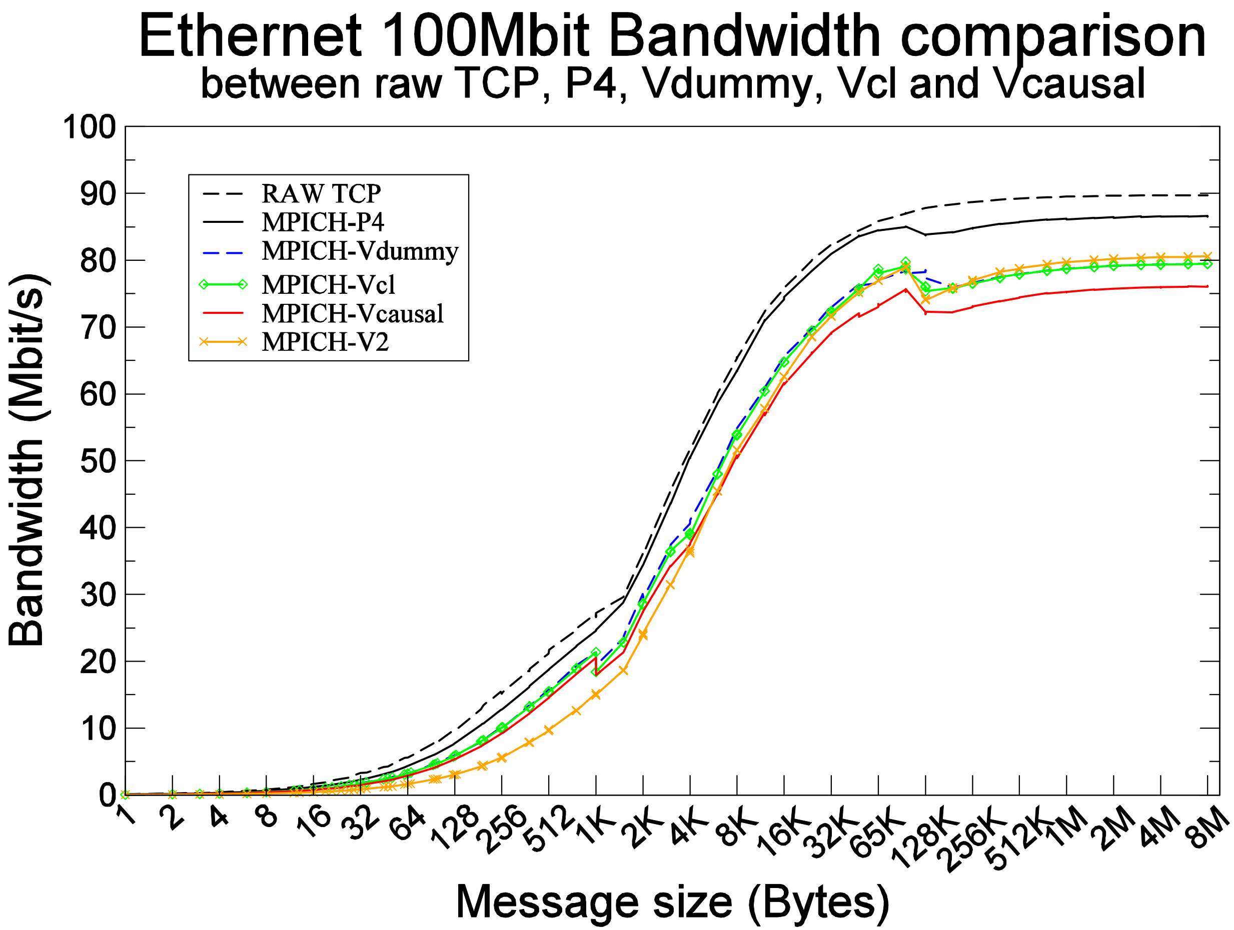
A single checkpoint server for 4 MPI tasks (P4 driver)

Ckpt is performed at random time on each node (no sync.)



→When 4 checkpoints are performed per process performance is about 94% the one of a non checkpointed execution.
→Several nodes can use the same CS

Bandwidth and latency (Ethernet)



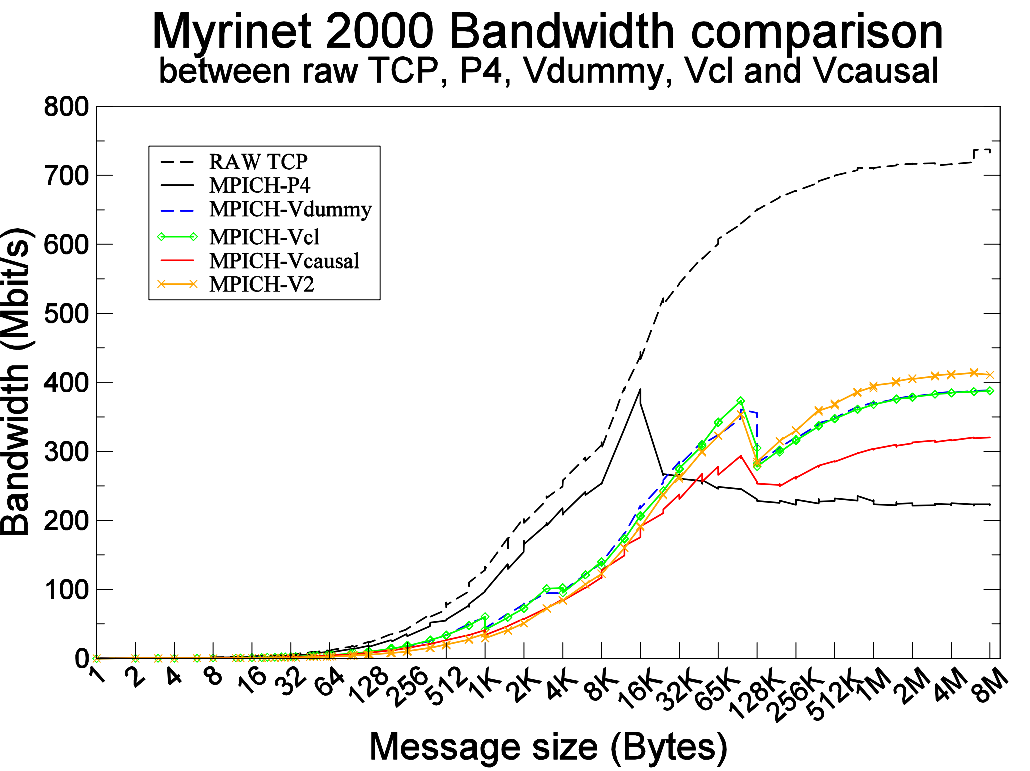
Latency for a 1 byte MPI message :

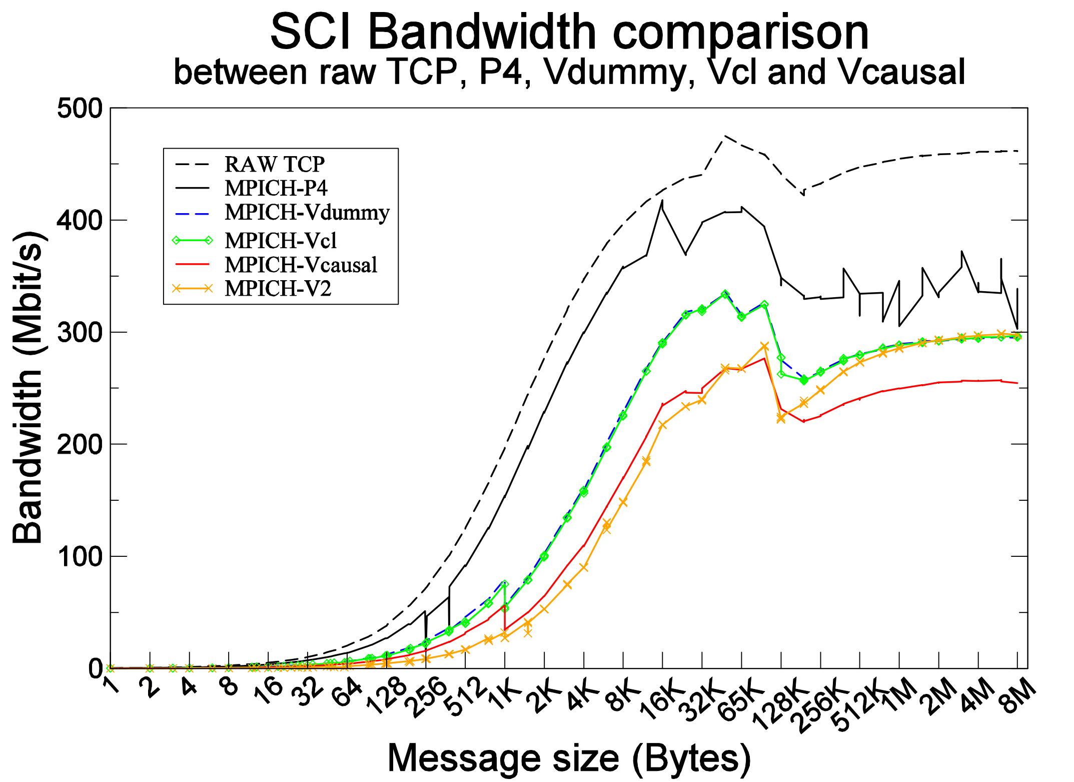
TCP (75us)
MPICH-P4 (100us)
MPICH-V (135us)
MPICH-Vcl (138us)
MPICH-Vcausal (157us)
MPICH-V2 (291us)

Latency is high in MPICH-Vcl 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)

Bandwidth and latency
(high speed networks)

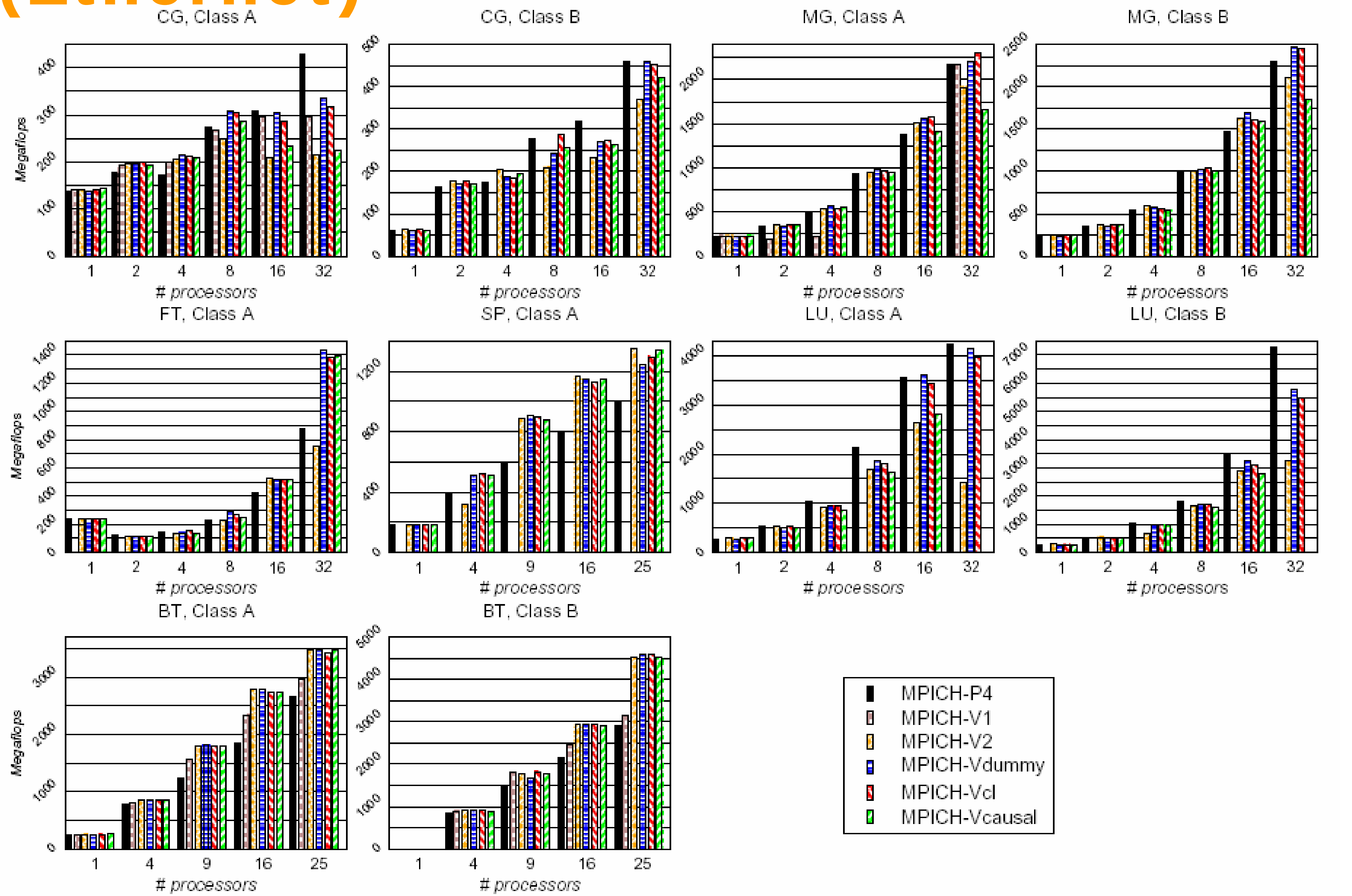




Latency for a 1 byte MPI message :

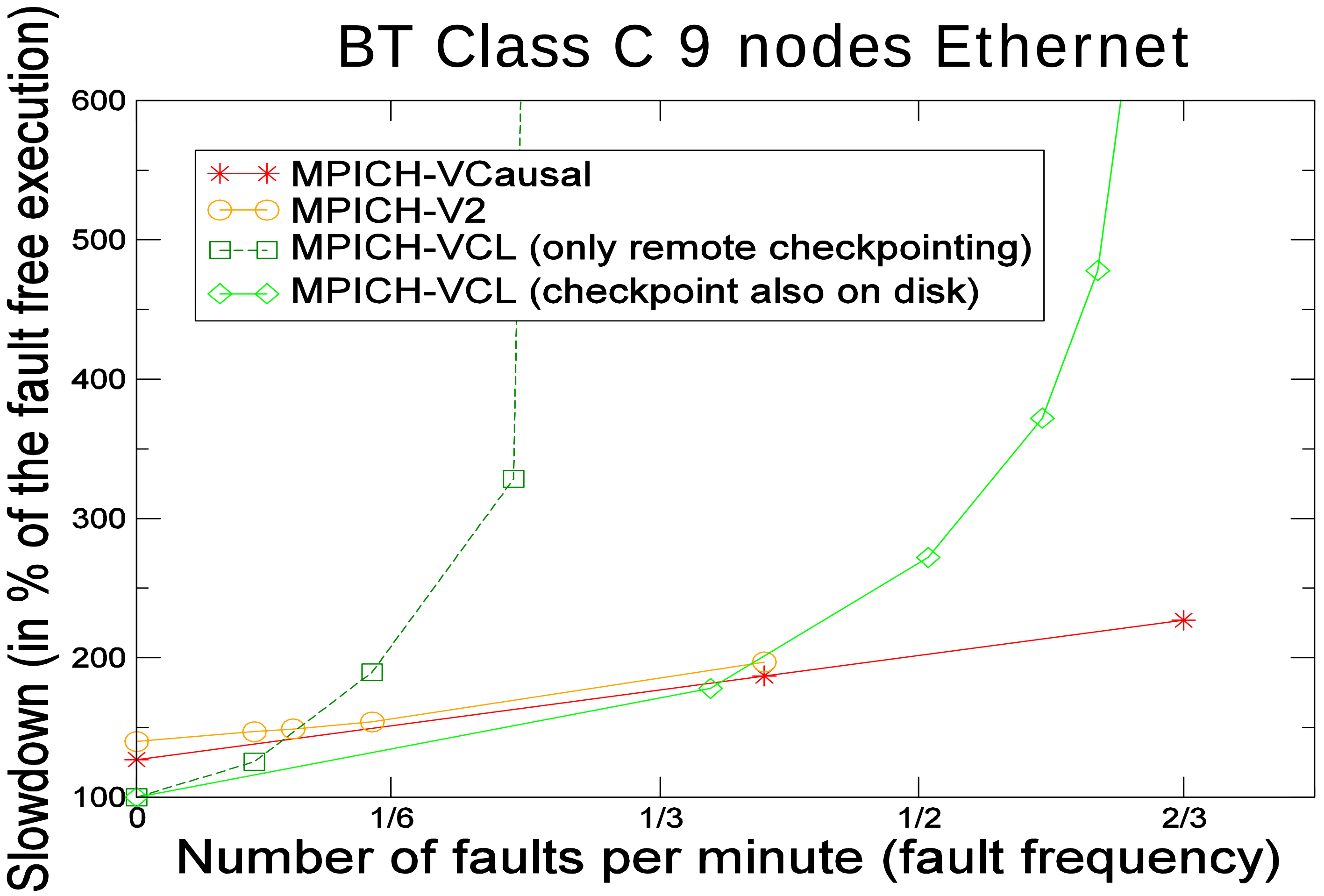
	Myrinet 2000	SCI
TCP	(43us)	(23us)
MPICH-P4	(53us)	(34us)
MPICH-V	(94us)	(76us)
MPICH-Vcl	(99us)	(81us)
MPICH-Vcausal	(112us)	(116us)
MPICH-V2	(183us)	(355us)

NAS Benchmark Class A and B (Ethernet)



NAS Benchmark Class A and B (Ethernet)

Fault impact on performance



- 20% overhead for fault free execution of Vcausal. (40% for pessimistic implementation). Crosspoint between Vcl and Vcausal at 0.006 faults per second (0.002 for the crosspoint between pessimistic and remote-checkpoint Vcl)
- If we consider a 1GB memory occupation for every process, an extrapolation expects the crosspoint to appear around one fault every 9 hours.
- Message logging implementation can tolerate a high fault rate. MPICH-Vcl cannot ensure termination of the execution for a high fault rate.

What we have learned from MPICH-V

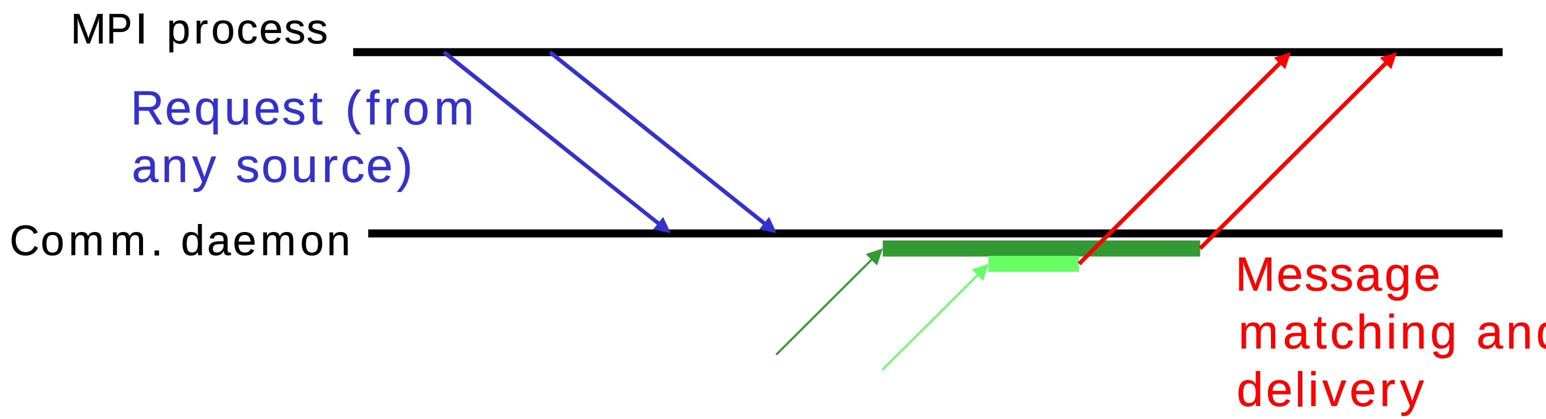
- MPICH-Vcl, MPICH-V2 and MPICH-Vcausal are comparable implementations of fault tolerant MPI from the MPICH-1.2.5, using respectively coordinated checkpoint, pessimistic message logging and causal message logging
- We have compared the overhead of these techniques according to fault frequency
- The recovery overhead is the main factor differentiating performance
- We have found a crosspoint from which message logging becomes better than coordinated checkpoint. On our test application this crosspoint appears near 1 per 3 minutes. The crosspoint for a 1GB dataset application should be around 9 hours. Considering MTBF of cluster lower than 9 hours, the coordinated checkpoint appear to be appropriate.
- MPICH-V framework is not as efficient as expected : much overhead lies in framework and not in protocols !
- MPICH-V framework is not suitable for high performance networks:
  - Overhead of the framework is too high for a production platform
  - As a research tool, overhead is “flatening” performance comparisons of the various protocols when using HP networks

Outline

- Protocols and Related works
- MPICH-V Comparison framework
- Performance
- **OpenMPI-V**
- Conclusion and future works

Ongoing work: OpenMPI-V Zero copy high perf implementation

MPICH-V



In green: incoming messages

OpenMPI-V