ハイパフォーマンス コンピューティング

2014/11/10

14M37138

社本秀之

紹介論文

- Detection and Correction of Silent Data Corruption for Large-Scale High-Performance Computing
  - David Fiala, Frank Mueller, Christian Engelmann, Rolf Riesen, Kurt Ferreira, Ron Brightwell
  - SC12

Resilience in HPC

- Faults have become the norm rather than the exception for parallel computation on large systems
  - Jugar (now Titan)
    - Cores: 150,152
    - MTBF: 52 hours

- Recent work shows that:
  - Servers tend to crash twice year
  - 1-5% of disk drives die per year

→ Need of Checkpoint/Restart paradigm for running large-scale jobs

Increase of C/R overhead

- HPC applications required to support C/R paradigm
  - As we enlarge systems, C/R overhead grows exponentially
  - Sandia's study:

TABLE I
168-HOUR JOB, 5 YEAR MTBF

# Nodes	work	checkpt	recomp.	restart
100	96%	1%	3%	0%
1,000	92%	7%	1%	0%
10,000	75%	15%	6%	4%
100,000	35%	20%	10%	35%

- Need of low cost fault tolerant mechanisms

Proposal and Contribution

- Proposal
  - Design and implementation of novel mechanisms for FT in HPC
  - Demonstrate capabilities of SDC protection at communication layer

- Contribution
  - MsgPlusHash, a proposed method, achieves low overheads from 0% to 30% for dual/triple redundancy
    - Runs on ARC cluster at NCSU (108 nodes, 1700+ cores)
    - HPCCG, SWEEP3D, etc.
  - All injected faults are detected by using the proposed method

Silent Data Corruption

- Silent Data Corruption faults
  - Bit flips
  - Some of them are not detectable/correctable
    - Invalid results (applications don't stop)
- Memory bit flips correctable by ECC
  - ECC has upper limit of bit flips
- One of two undetectable errors are expected to occur in a day on ORNL's Jaguar Supercomputer

Related Work

- Redundant MPI implementations:
  - rMPI [K. Ferreira et al.]
    - Built using MPICH
    - Using the MPI profiling layer PMPI
  - MRMPI [C. Engelmann et al.]
    - Not rely on a specific MPI library
    - Using the MPI profiling layer PMPI
  - VolpexMPI [T. LeBlanc et al.]
    - Implemented from scratch
    - Using polling mechanism
    - No support for MPI_ANY_SOURCE

→ All of these implementations don't protect against SDC

Design of RedMPI

- Create replica MPI processes
  - Replicas run same applications
  - Replicas always send same massages when no data corruption
- Dual redundancy
  - Message verification
- Triple redundancy
  - Message verification and correction

	No Redundancy	Dual Redundancy	Triple Redundancy
SDC Detection		✔	✔
SDC Correction			✔

Design Assumptions

- Reliable transport layer (TCP Ethernet/ Infiniband)

- MPI functions supported
  - point-to-point, collectives, wildcards…

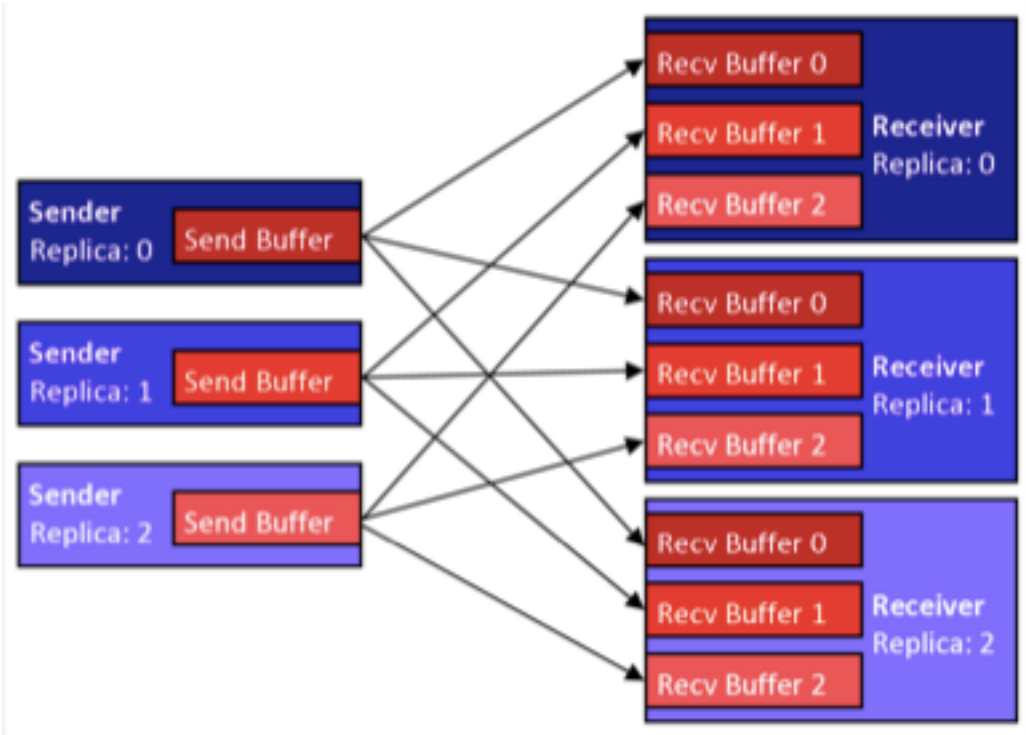
Redundant MPI Ranks

- Transparently creates r replicas per normal MPI process

- Virtual rank
  - seen by applications
- Native rank
  - seen by MPI
- Replica rank
  - given 0~r-1 to identify replicas

SDC Detection method 1 (All-to-all)

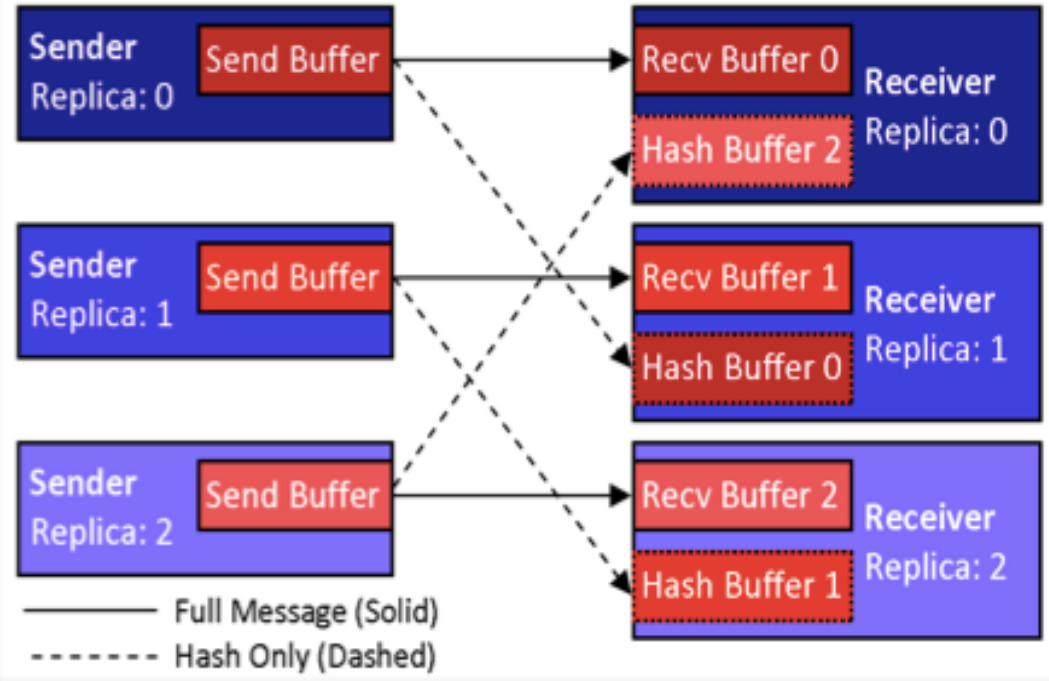
- Each sender sends full copy of a message to other receiver
- r receive buffers
- r^2 messages



SDC Detection method 2 (MsgPlusHash)

- Reducing the total data transfer overhead compared to the previous method

- 1 receive buffer
- 1 hash buffer



Experimental Setup

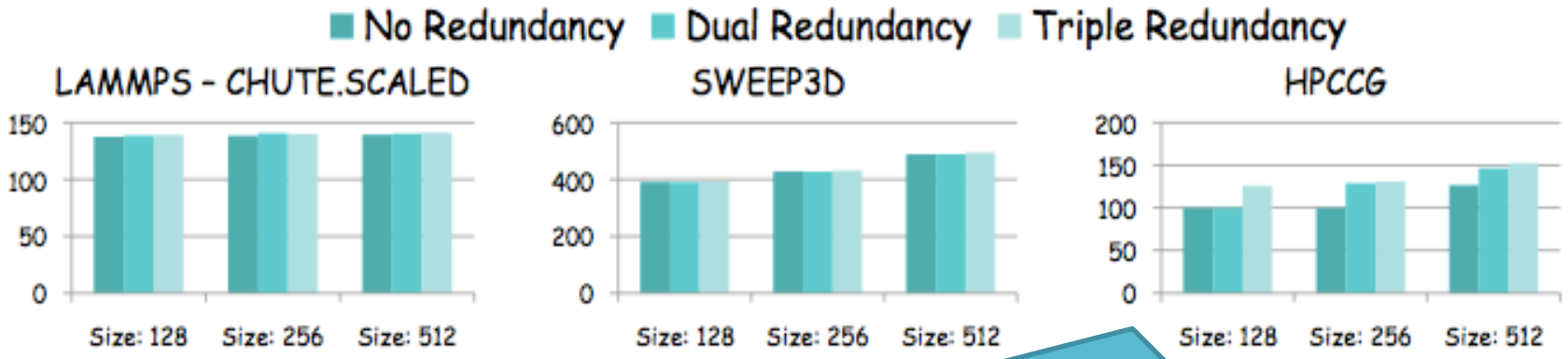
- ARC cluster at NCSU
  - 108 nodes, over 1700 cores
  - 32GB DRAM per node
  - 8GB/s Infiniband interconnect

→ Using at most 1536 processes

- OpenMPI 1.5

- Applications
  - LAMMPS, SWEEP3D, HPCCG, etc.

Weak Scaling

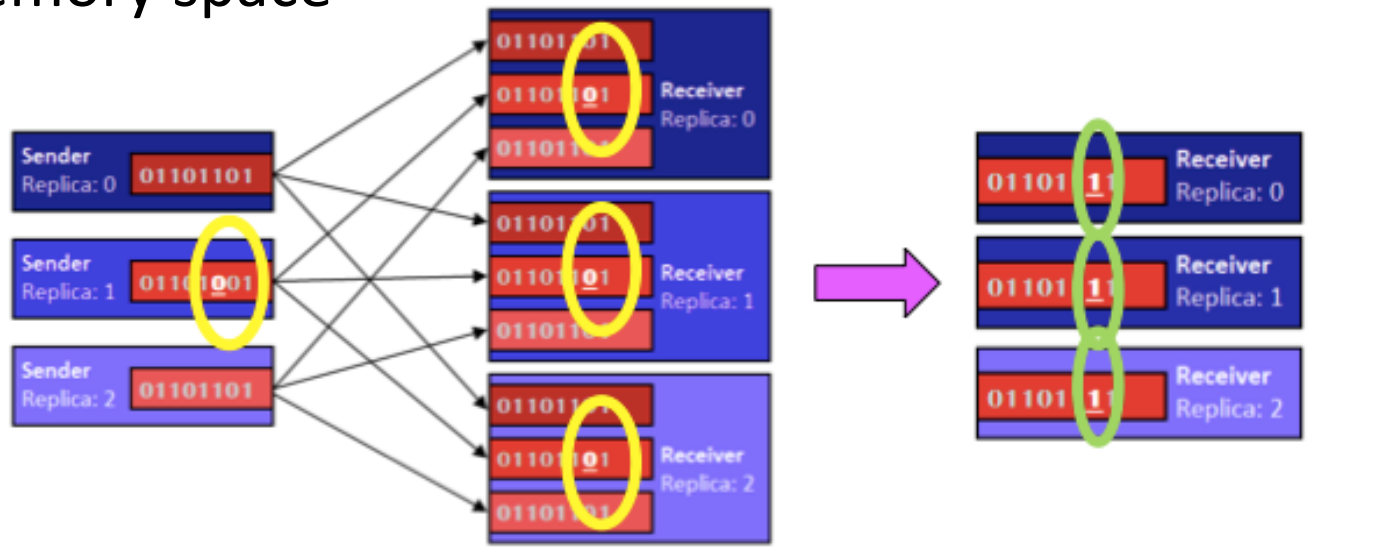


# of processes	No Redundancy	Dual Redundancy	Triple Redundancy
128	99.8[s]	99.8[s]	125.8[s]
256	99.6[s]	128.8[s]	131.0[s]
512	126.4[s]	146.2[s]	152.3[s]

- At most approx. 30% overhead
- The overhead ratio is still modest as the # of processes grows

Fault Injector

- When sending messages, 1/x messages randomly receive 1 random bit flip
- Modifies not only send buffer, but also the original memory space



bit is permanently flipped in sender's buffer ➔ passed to receivers

Receivers detect corruption

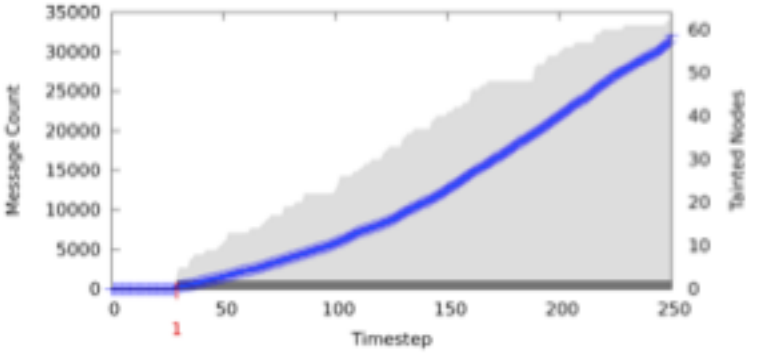
Retains only correct msg

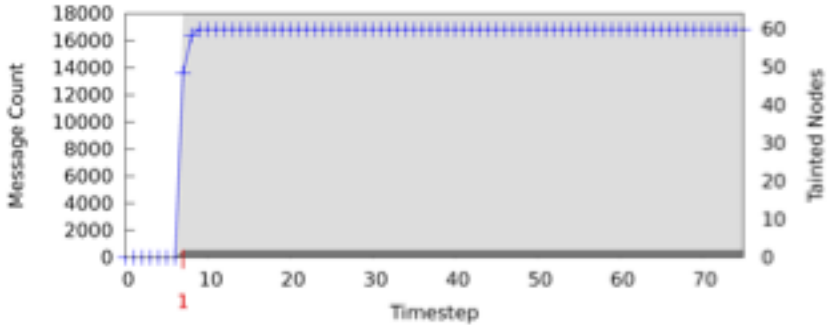
Fault Injection Experiments

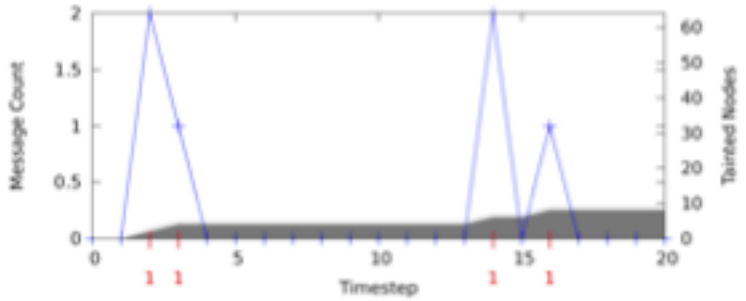
- Propagation
  - Investigate how quickly do SDC injections propagate to other processes via communication
    - NPB (LU, BT, SP, etc.)
  - Dual redundancy
    - Allow application progress to continue when detecting corruptions
- Protection
  - Investigate the effectiveness of RedMPI's SDC detection/ correction
    - CG benchmark
  - Triple redundancy

Experiments: Propagation

- Progressive
  - Communicate with their grid neighbors
- Explosion
  - Use collectives or send msgs to all nodes
- Localized
  - Corrupted data is neither reused nor transmitted







Experiments: Protection #1

- Configuration
  - Corruption frequency (1 bit flip): 1/5,000,000
  - virtual ranks: 64, physical ranks: 192
  - Inject corruption to only the process whose replica rank is 0

- 10 times runs in total
  - 1 occasion with two injections
  - 4 occasions with one injection
  - 5 occasions without injections

→ All runs pass benchmark's built-in verification

Experiments: Protection #2

- Configuration
  - Corruption frequency (1 bit flip): 1/2,500,000
  - virtual ranks: 64, physical ranks: 192
  - Doubling the odds for and injection
  - Remove the process selection restriction

- 10 times runs in total
  - 2.5 injections on average & several thousand invalid messages per run

→ RedMPI forced corrupted job to fail

Conclusion

- Design and develop
  - MsgPlusHash, a proposed method, achieves low overheads from 0% to 30% for dual/triple redundancy
    - Runs on ARC cluster at NCSU (108 nodes, 1700+ cores)
    - HPCCG, SWEEP3D, etc.
- All injected faults are detected by using the proposed method

→ Redundancy may be worth the cost to protect and ensure correct output

Discussion

- How does the overhead change when the datasets we handle become huge?
  - Redundancy use much memory

- How much power consumption when using triple redundancy?