Finding Optimal Checkpoint Interval for Inter-dependent parallel processes in Volunteer PC grids

Mohammad Tanvir Rahman and Dr Jaspal Subhlok

The Problem

- To minimize the completion time for inter-dependent parallel processes running in a Volunteer environment.

Key idea: By adopting optimal checkpoint interval considering any level of replication of the clients.

Key Contribution:
- Designing a mathematical model for finding optimal check point interval
- Implementing the algorithm and evaluating the performance.

What is Volunteer Environment

- A network of idle computers
- Can be in different physical location and connected through a volunteer framework.
- Provides free CPU power
- Frequent failure
- Checkpoint and replication needed for successful parallel execution

Motivation

- Checkpoint and replication needed for effective parallel execution
- Optimal Checkpoint Interval needed for minimizing completion time and increase collective throughput
- Too frequent checkpoints: waste a lot of time and resources
- Infrequent checkpoints: increase a work-loss in case of client failure

Challenges in Finding Optimal Checkpoint Interval:
- Highly failure characteristics of volunteer nodes
- Replication of processes

Experiment setup

Testbed: Around 300 volunteer nodes. Roughly 25% on our university campus and 75% distributed worldwide

Test Application: Replica Exchange Molicular Dynamics (REMD)

Parameters:
- Number of processes: 16, 32
- Level of replication : 1,2,3
- Checkpoint interval: 12 to 24000 sec
- Checkpoint size: 50KB
- Host selection policy: None or Active
- Number of Run: 10 times

Algorithm

Input:
- Success distribution of a single process, p = f(t)
- Number of processes, n
- Number of replica for each process, r
- Time to create a checkpoint, T_s

Output:
Optimal Checkpoint interval, T_c

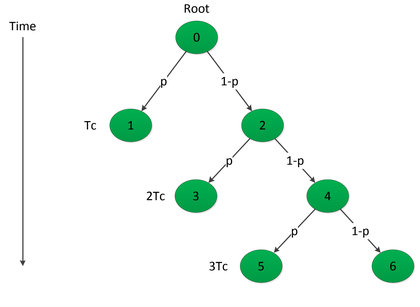

Fig 1: Tree structure to avg time to reach next check point

Starting from root, after time T_c, probability to reach next checkpoint (node 1) is p and probability to fail is (1-p).

If failed, probability to reach to next checkpoint (node 3) will be p*(1-p). Average time to reach to next check point can be measured using the tree in Fig 1.

For n processes each with r replica, Optimal checkpoint interval, $T_{c(op)}$ =

$$\min_{T_c = 1 \ldots \infty} \frac{1}{(1-(1-f(T_c)^r)^n)} + \frac{T_s}{T_c}$$

Result

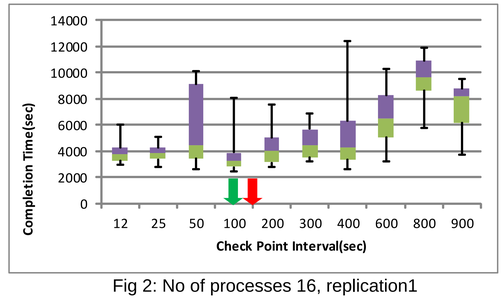

Fig 2: No of processes 16, replreplication1

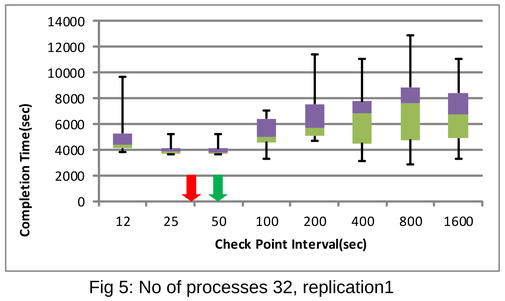

Fig 5: No of processes 32, replication1

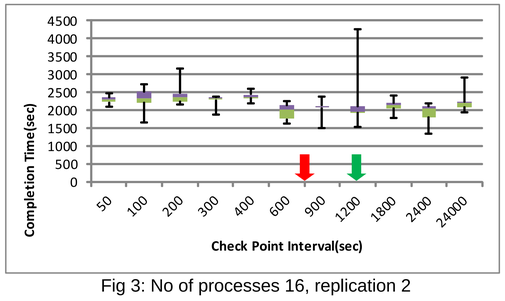

Fig 3: No of processes 16, replication 2

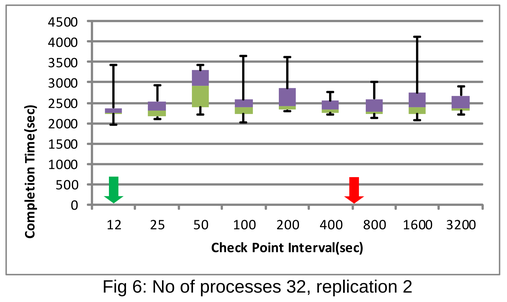

Fig 6: No of processes 32, replication 2

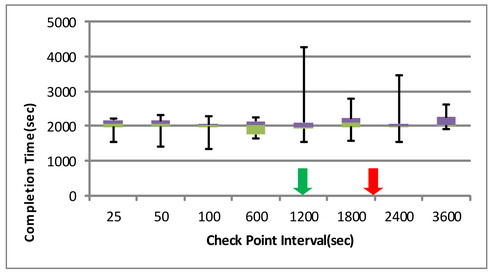

Fig 4: No of processes 16, replication 3

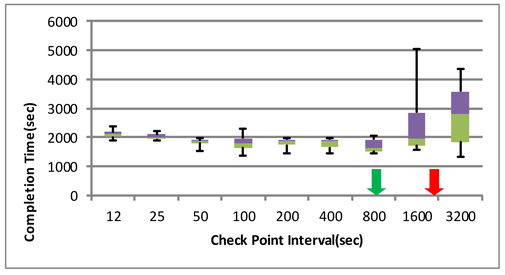

Fig 7: No of processes 32, replication 3

Legend:
- (Red Arrow) indicates Optimal CPI calculated by our algorithm
- (Green Arrow) indicates CPI for measured lowest execution time

Conclusion

TABLE I: optimal checkpoint interval calculated by our algorithm

No of Process, n	No of Replica, r	CPI Calculated by our Algorithm, T (s)	Avg Completion time for T (s)	Actual Optimal CPI, Tc (s)	Avg Completion time for Tc (s)	Diff in CPI (%)	Diff in Performance increase (%)
16	1	125	3025	100	3276.50	25.00	-7.68
16	2	789	2080	1200	1966.00	-34.25	5.80
16	3	1826	2089	1200	1966.00	52.17	6.26
32	1	29	3701	50	3813.00	-42.00	-2.94
32	2	605	2107	12	2252.00	4941.67	-6.44
32	3	1988	1876	800	1620.00	148.50	15.80

- The Check point Interval (CPI) calculated by our algorithm minimizes the completion time of the job.
- Even if the CPI calculated by our algorithm is not close to the Actual Optimal Check Point Interval, the performance of minimizing completion time with our calculated CPI is within 15% range with that of Optimal CPI.