Checkpoint Advisory Tool: How often should my application checkpoint, if at all?

Devesh Tiwari

tiwari@ornl.gov

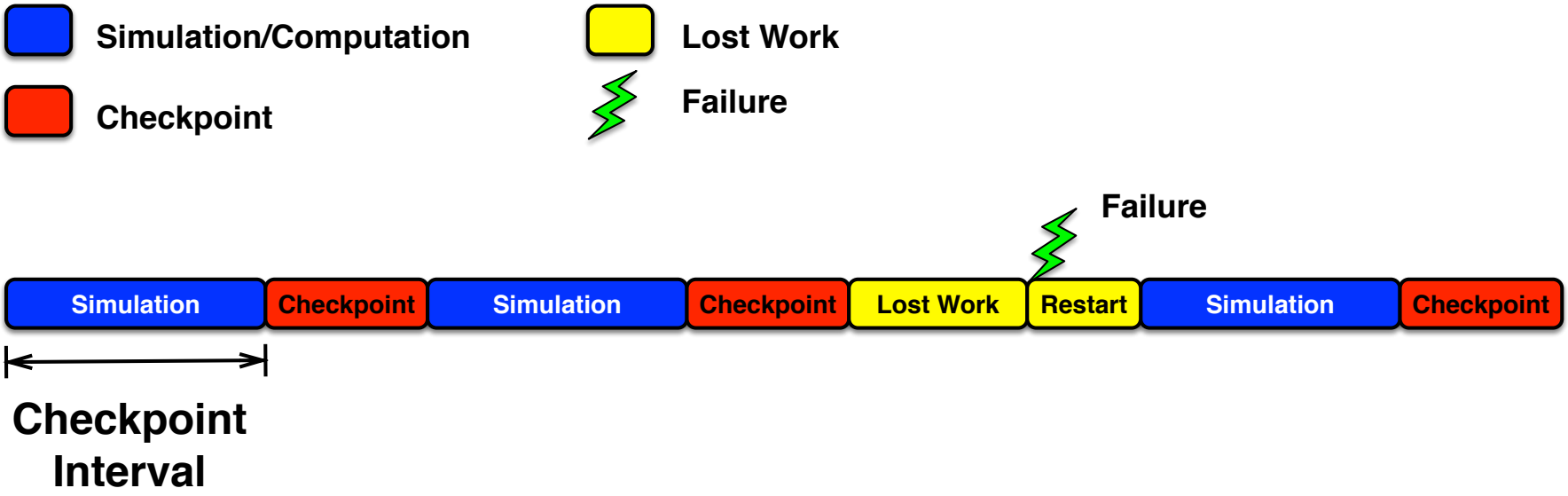
Technology Integration Group

National Center for Computational Sciences

Agenda

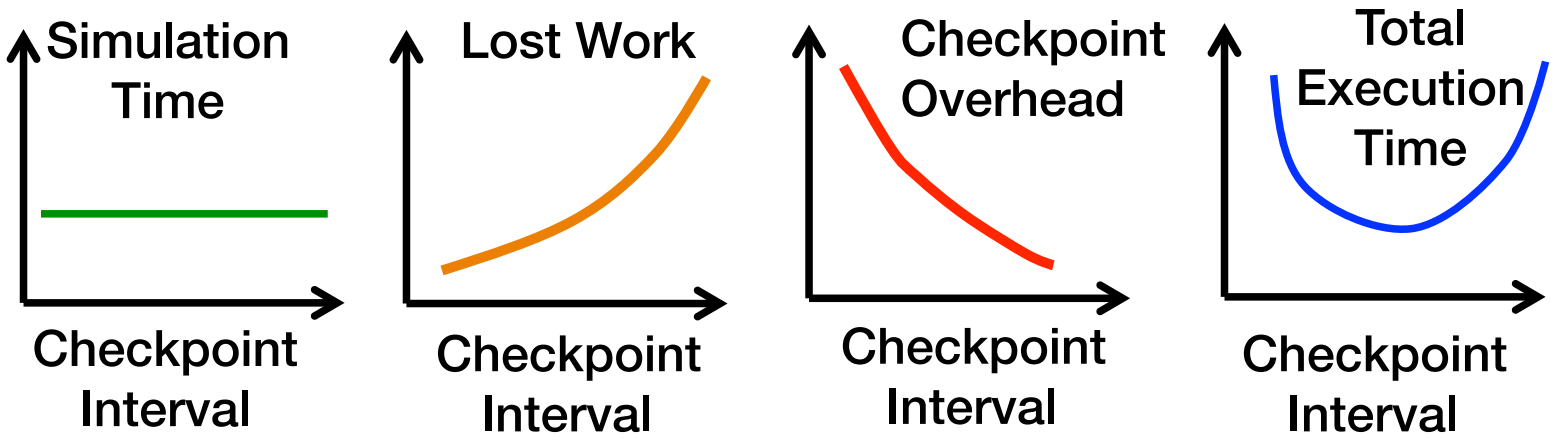
- Background on Checkpointing
- Optimal Checkpointing Interval
- Overview of the Checkpoint Advisory Tool
- How to use it?
- Feedback

Long-running, large-scale scientific applications can be interrupted by system failures on HPC systems.



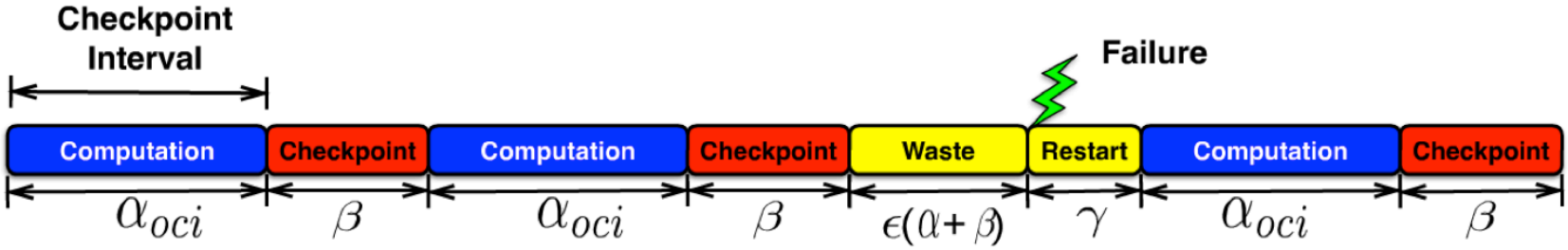
Checkpoint interval is the time period after which an application regularly takes a checkpoint.

Total execution time is the sum of the useful computation/simulation time, the checkpointing overhead and the lost work



More frequent checkpointing potentially reduces the potential lost work but increases the checkpointing overhead and vice-versa.

Deriving Optimal Checkpointing Interval (OCI)



α_{oci}	Optimal Checkpoint Interval (OCI)
β	Time-to-checkpoint
ϵ	Average lost work fraction
γ	Restart overhead
M	Mean Time Between Failure (MTBF)

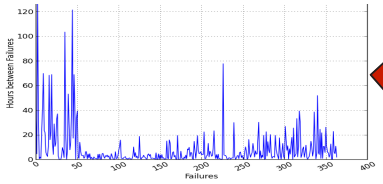
$$\alpha_{oci} = \sqrt{\beta^2 + \frac{\beta\gamma}{\epsilon} + \frac{M\beta}{\epsilon}}$$

System Failure and I/O Load Aware Checkpointing

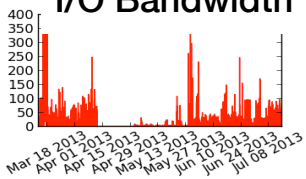
Titan Compute Nodes

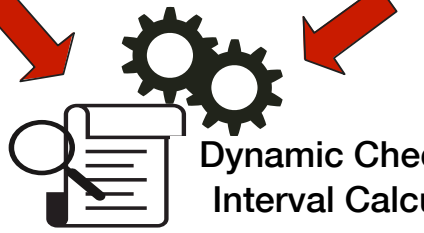
Spider II Storage System

System Failure
Inter-arrival Distribution

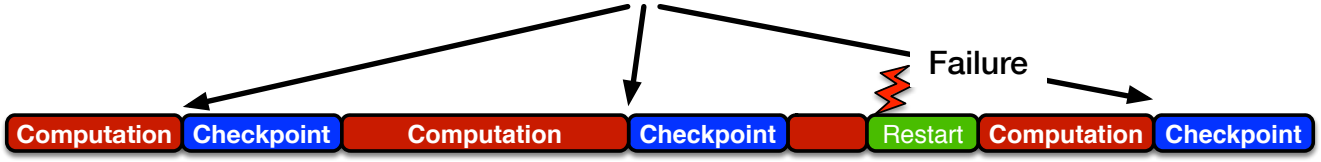


I/O Bandwidth





Dynamic Checkpoint
Interval Calculation



Checkpoint advisory tool to help users place checkpoints optimally

- reduce I/O overhead and increase resource utilization efficiency

Improving application performance via combining system resource usage state/history with application I/O requirements

Checkpoint Advisory

Checkpoint Advisory -- v1.0 (July 2015) usage and command line options

module load checkpointadvisory
Usage: checkpoint_advisory arguments and options
Usage: checkpoint_advisory -c <Time To Checkpoint> -s <Job Size> -n <Job Name> -i <Job ID> -t <Requested Wall Clock Time>

Required Arguments:
-c --checkpoint_time Time to write one checkpoint (in minutes)
-s --job_size Job size (number of nodes)

Optional Arguments:
-n --job_name Name of the job
-i --job_id Job ID
-t --requested_time Requested wall clock time

Notes:
Time to checkpoint is a required argument. The input should be in minutes. It can be an estimation based on prior runs.
Job size is the number of nodes to be allocated for next job (aprun command).
Job size argument is not necessarily the number of nodes requested by the batch job (i.e., $PBS_NUM_NODES).
Job name and ID are optional parameters.
If you are running multiple apruns within one batch job, supplying application's name is recommended every time checkpoint advisory is invoked.

Examples:

```
checkpoint_advisory --checkpoint_time 20 --job_size 100
checkpoint_advisory -c 5 -s 4096
checkpoint_advisory --checkpoint_time 2 --job_size 10000 -n $PBS_JOBNAME -i $PBS_JOBID --requested_time $PBS_WALLTIME
```

Job script Integration

```
#!/bin/bash

#PBS -N Job_Name
#PBS -A Account_Number
#PBS -l walltime=05:00:00
#PBS -l nodes=1000
#PBS -o outputlog
#PBS -e errorlog

#Time to checkpoint in minutes

#My OCI in minutes

module load checkpointadvisory

OIC=$(eval ”checkpoint_advisory --checkpoint_time 0.20 --job_size 1000")

# between multiple apruns

echo "Optimal Interval for Checkpointing (Oh-I-See):" $OIC

# Change directory

cd $MEMBERWORK/my_directory

aprun -n 1200 -N 16 ./app_name

exit 0
```

As Plug-In Module

```
mbpro97062:Tool.Checkpoint.Advisory dgt$ python Cp_Advisory_V1.0.py
-------------entering <module>-------------

Parameters:
System Size (number of nodes): 18688
System MTBF: 25.0 hours
-----
Job Size (number of nodes): 3750
Job MTBF: 124.586666667 hours
Time-to-Write-One-Checkpoint: 10.0 minutes
-----

-------------entering optimal_checkpoint_interval-------------
Optimal Checkpoint Interval 6.33 hours
Optimal Checkpoint Interval 380.02 minutes
-----------------------------------------------

-------------exiting <module>-------------
```

Checkpoint Advisory Tool: How often should my application checkpoint, if at all?

Devesh Tiwari

tiwari@ornl.gov

Technology Integration Group

National Center for Computational Sciences

This work used the resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. Department of Energy, under the contract No. DEAC05-00OR22725.