Combining Checkpointing and Replication for Reliable Execution of Linear Workflows

Anne Benoit[1,2], Aurélien Cavelan[3], Florina M. Ciorba[3], Valentin Le Fèvre[1], Yves Robert[1,4]

1. LIP, Ecole Normale Supérieure de Lyon, France
2. Georgia Institute of Technology, Atlanta, GA, USA
3. University of Basel, Switzerland
4. University of Tennessee, Knoxville, TN, USA

http://graal.ens-lyon.fr/~abenoit/

APDCM workshop @ IPDPS'18, Vancouver, May 21, 2018

Linear workflows

- High-performance computing (HPC) application: chain of tasks $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n$
- Parallel tasks executed on the whole platform
- For instance: tightly-coupled computational kernels, image processing applications, ...
- Goal: efficient execution, i.e., minimize total execution time

Linear workflows

- High-performance computing (HPC) application: chain of tasks $T_1 \to T_2 \to \cdots \to T_n$
- Parallel tasks executed on the whole platform
- For instance: tightly-coupled computational kernels, image processing applications, ...
- Goal: efficient execution, i.e., minimize total execution time

Reliable execution

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores

- Failure-prone

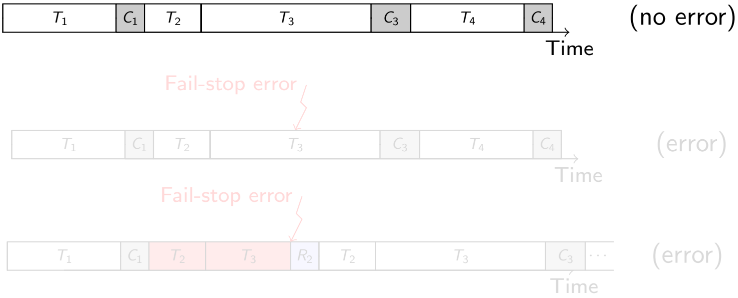
MTBF – one node	1 year	10 years	120 years
MTBF – platform of 10^6 nodes	30sec	5mn	1h

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

Need to ensure that the execution will be reliable, i.e., without failures

Coping with fail-stop errors with checkpoints

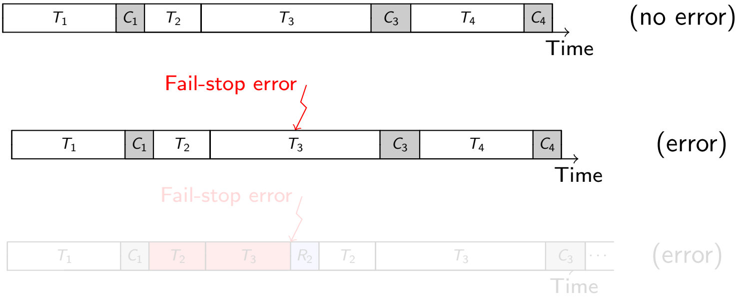
Checkpoint, rollback, and recovery:



- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection
- Rollback to last checkpoint and re-execute

Coping with fail-stop errors with checkpoints

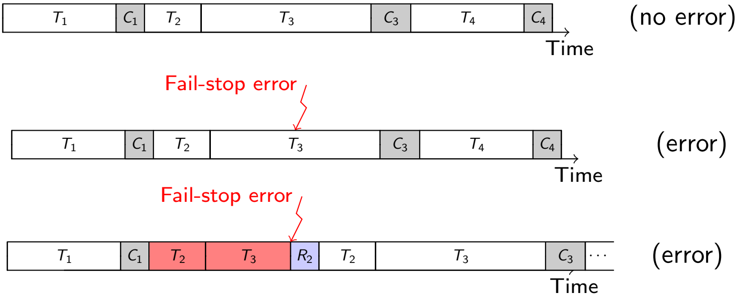
Checkpoint, rollback, and recovery:



- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection
- Rollback to last checkpoint and re-execute

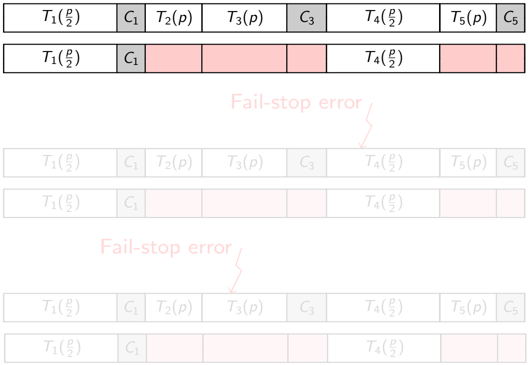
Coping with fail-stop errors with checkpoints

Checkpoint, rollback, and recovery:



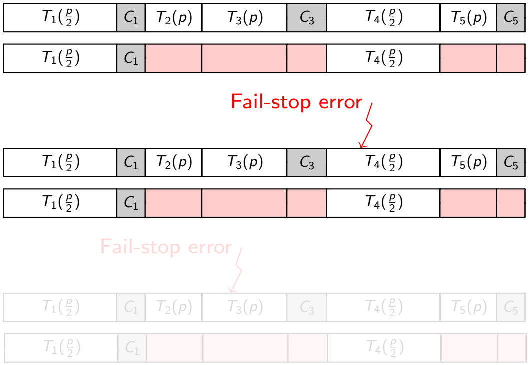
- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection
- Rollback to last checkpoint and re-execute

Coping with fail-stop errors with replication



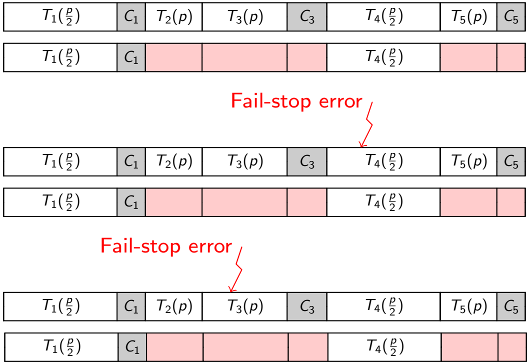
- The whole platform is used at all time, some tasks are replicated
- If failure hits a replicated task, no need to rollback
- Otherwise, rollback to last checkpoint and re-execute

Coping with fail-stop errors with replication



- The whole platform is used at all time, some tasks are replicated
- If failure hits a replicated task, no need to rollback
- Otherwise, rollback to last checkpoint and re-execute

Coping with fail-stop errors with replication



- The whole platform is used at all time, some tasks are replicated
- If failure hits a replicated task, no need to rollback
- Otherwise, rollback to last checkpoint and re-execute

Contributions

- Both checkpointing and replication have been extensively studied
- Combination of both techniques not yet investigated
- Detailed model
- Optimal dynamic programming algorithm
- Experiments to evaluate impact of using both replication and checkpointing during execution
- Guidelines about when to checkpoint only, replicate only, or combine both techniques

Contributions

- Both checkpointing and replication have been extensively studied
- Combination of both techniques not yet investigated
- Detailed model
- Optimal dynamic programming algorithm
- Experiments to evaluate impact of using both replication and checkpointing during execution
- Guidelines about when to checkpoint only, replicate only, or combine both techniques

Outline

Application and platform model

- Application:
  - Chain $T_1 \to T_2 \to \cdots \to T_n$
  - Parallel tasks: (failure-free) execution time of T_i using q_i processors is $w_i \left(\alpha_i + \frac{1-\alpha_i}{q_i}\right)$ (Amdahl's law)
- Platform:
  - Homogeneous platform with p processors P_i, $1 \leq i \leq p$
  - Fail-stop errors, Exponential distribution, error rate λ_{ind}
  - $\mathbb{P}(X \leq T) = 1 - e^{-q\lambda_{ind}T}$ on q processors

Checkpointing

- Checkpointing time: $C_i(q_i) = a_i + \frac{b_i}{q_i} + c_i q_i$
  - $a_i + \frac{b_i}{q_i}$: communication time with latency a_i
  - $c_i q_i$: message passing overhead
- Downtime D
- Recovery cost R_{j+1} (where T_j is the last checkpointed task)
- $R_{i+1}(q_i) = C_i(q_i)$ for $1 \leq i \leq n-1$:
  recovering for $T_{i+1} \approx$ reading C_i
- T_0 with $w_0 = 0$ checkpointed (input time $R_1(q_1)$)
- T_n always checkpointed (output time $C_n(q_n)$)

No replication

- T_i not replicated: costs C_i^{norep} and R_i^{norep}
- Failure-free execution time: $T_i^{norep} = w_i\left(\alpha_i + \frac{1-\alpha_i}{p}\right)$
- Expected execution time $\mathbb{E}^{norep}(i)$:

$$\mathbb{E}^{norep}(i) = \mathbb{P}(X_p \leq T_i^{norep})\Big(T_{lost}^{norep}(T_i^{norep}) + D + R_i^{norep} + \mathbb{E}^{norep}(i)\Big) + (1 - \mathbb{P}(X_p \leq T_i^{norep}))T_i^{norep}$$

- $\mathbb{P}(X_p \leq t) = 1 - e^{-\lambda_{ind} p t}$: probability of failure on one of the p processors before time t
- $T_{lost}^{norep}(T_i^{norep}) = \frac{1}{\lambda_{ind} p} - \frac{t}{e^{\lambda_{ind} p T_i^{norep}} - 1}$
- $\mathbb{E}^{norep}(i) = (e^{\lambda_{ind} p T_i^{norep}} - 1)(\frac{1}{\lambda_{ind} p} + D + R_i^{norep})$
- If T_i is checkpointed, add C_i^{norep}

No replication

- T_i not replicated: costs C_i^{norep} and R_i^{norep}
- Failure-free execution time: $T_i^{norep} = w_i \left(\alpha_i + \frac{1-\alpha_i}{p}\right)$
- Expected execution time $\mathbb{E}^{norep}(i)$:

$$\mathbb{E}^{norep}(i) = \mathbb{P}(X_p \le T_i^{norep})\Big(T_{lost}^{norep}(T_i^{norep}) + D + R_i^{norep} + \mathbb{E}^{norep}(i)\Big) + (1 - \mathbb{P}(X_p \le T_i^{norep}))T_i^{norep}$$

- $\mathbb{P}(X_p \le t) = 1 - e^{-\lambda_{ind} pt}$: probability of failure on one of the p processors before time t
- $T_{lost}^{norep}(T_i^{norep}) = \frac{1}{\lambda_{ind} p} - \frac{t}{e^{\lambda_{ind} p T_i^{norep}} - 1}$
- $\mathbb{E}^{norep}(i) = (e^{\lambda_{ind} p T_i^{norep}} - 1)(\frac{1}{\lambda_{ind} p} + D + R_i^{norep})$
- If T_i is checkpointed, add C_i^{norep}

Replication

- T_i replicated: if a copy fails, downtime + recovery
- Each copy uses $p/2$ processors; costs C_i^{rep} and R_i^{rep}
- Failure-free execution time: $T_i^{rep} = w_i \left(\alpha_i + \frac{1-\alpha_i}{\frac{p}{2}}\right)$
- Expected execution time $\mathbb{E}^{rep}(i)$ **if** T_{i-1} **is checkpointed**:

$$\mathbb{E}^{rep}(i) = \mathbb{P}(Y_p \leq T_i^{rep})\left(T_{lost}^{rep}(T_i^{rep}) + D + R_i^{rep} + \mathbb{E}^{rep}(i)\right) + (1 - \mathbb{P}(Y_p \leq T_i^{rep}))\,T_i^{rep}$$

- $\mathbb{P}(Y_p \leq t) = (1 - e^{-\frac{\lambda_{ind}\, p}{2} t})^2$: probability of failure on both replicas of $\frac{p}{2}$ processors before time t
- $T_{lost}^{rep}(T_i^{rep})$ computed as before
- ...
- Formula for $\mathbb{E}^{rep}(i)$

Replication

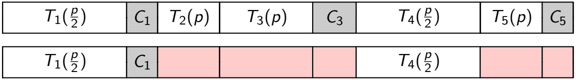
- T_i replicated: if a copy fails, downtime + recovery
- Each copy uses $p/2$ processors; costs C_i^{rep} and R_i^{rep}
- Failure-free execution time: $T_i^{rep} = w_i \left(\alpha_i + \frac{1-\alpha_i}{\frac{p}{2}}\right)$
- Expected execution time $\mathbb{E}^{rep}(i)$ **if** T_{i-1} **is checkpointed**:

$$\mathbb{E}^{rep}(i) = \mathbb{P}(Y_p \leq T_i^{rep})\left(T_{lost}^{rep}(T_i^{rep}) + D + R_i^{rep} + \mathbb{E}^{rep}(i)\right) + (1 - \mathbb{P}(Y_p \leq T_i^{rep}))\, T_i^{rep}$$

- $\mathbb{P}(Y_p \leq t) = (1 - e^{-\frac{\lambda_{ind}\, p}{2} t})^2$: probability of failure on both replicas of $\frac{p}{2}$ processors before time t
- $T_{lost}^{rep}(T_i^{rep})$ computed as before
- ...
- Formula for $\mathbb{E}^{rep}(i)$

Optimization problem

- ChainsRepCkpt optimization problem
- Minimize the expected makespan of the workflow
- Four possibilities for each task:
  checkpoint or not, and replicate or not



Outline

Optimization problem

Theorem

The optimal solution to the ChainsRepCkpt *problem can be obtained using a dynamic programming algorithm in $O(n^2)$ time, where n is the number of tasks in the chain.*

- Recursively computes expectation of optimal time required to execute tasks T_1 to T_i and then checkpoint T_i
- Distinguish whether T_i is replicated or not

- $T^{rep}_{opt}(i)$: knowing that T_i is replicated
- $T^{norep}_{opt}(i)$: knowing that T_i is not replicated

- Solution: $\min\left\{T^{rep}_{opt}(n) + C^{rep}_n, T^{norep}_{opt}(n) + C^{norep}_n\right\}$

Computing $T_{opt}^{rep}(j)$: j is replicated

$$T_{opt}^{rep}(j) = \min_{1 \le i < j} \begin{cases} T_{opt}^{rep}(i) + C_i^{rep} + T_{NC}^{rep,rep}(i+1,j), \\ T_{opt}^{rep}(i) + C_i^{rep} + T_{NC}^{norep,rep}(i+1,j), \\ T_{opt}^{norep}(i) + C_i^{norep} + T_{NC}^{rep,rep}(i+1,j), \\ T_{opt}^{norep}(i) + C_i^{norep} + T_{NC}^{norep,rep}(i+1,j), \\ R_1^{rep} + T_{NC}^{rep,rep}(1,j), \\ R_1^{norep} + T_{NC}^{norep,rep}(1,j) \end{cases}$$

- T_i: last checkpointed task before T_j
- T_i can be replicated or not
- T_{i+1} can be replicated or not
- $T_{NC}^{A,B}$: no intermediate checkpoint, first/last task replicated or not, previous task checkpointed
- Similar equation for $T_{opt}^{norep}(j)$

Computing $T_{opt}^{rep}(j)$: j is replicated

$$T_{opt}^{rep}(j) = \min_{1 \le i < j} \left\{ \begin{array}{l} T_{opt}^{rep}(i) + C_i^{rep} + T_{NC}^{rep,rep}(i+1,j), \\ T_{opt}^{rep}(i) + C_i^{rep} + T_{NC}^{norep,rep}(i+1,j), \\ T_{opt}^{norep}(i) + C_i^{norep} + T_{NC}^{rep,rep}(i+1,j), \\ T_{opt}^{norep}(i) + C_i^{norep} + T_{NC}^{norep,rep}(i+1,j), \\ R_1^{rep} + T_{NC}^{rep,rep}(1,j), \\ R_1^{norep} + T_{NC}^{norep,rep}(1,j) \end{array} \right\}$$

- T_i: last checkpointed task before T_j
- T_i can be replicated or not
- T_{i+1} can be replicated or not
- $T_{NC}^{A,B}$: no intermediate checkpoint, first/last task replicated or not, previous task checkpointed
- Similar equation for $T_{opt}^{norep}(j)$

Computing $T_{NC}^{A,B}(i,j)$

$$T_{NC}^{A,B}(i,j) = \min\left\{ T_{NC}^{A,rep}(i,j-1), T_{NC}^{A,norep}(i,j-1) \right\} + T^{A,B}(j \mid i)$$

- $T^{A,B}(j \mid i)$: time needed to execute task T_j, knowing that a failure during T_j implies to recover from T_i:

$$T^{A,norep}(j \mid i) = \left(1 - e^{-\lambda T_j^{norep}}\right)\Big(T_{lost}^{norep}(T_j^{norep}) + D + R_i^A \\ + \min\left\{ T_{NC}^{A,rep}(i,j-1), T_{NC}^{A,norep}(i,j-1) \right\} + T^{A,norep}(j \mid i)\Big) \\ + e^{-\lambda T_j^{norep}}\left(T_j^{norep}\right)$$

$$T^{A,rep}(j \mid i) = \left(1 - e^{-\frac{\lambda T_j^{rep}}{2}}\right)^2 \Big(T_{lost}^{rep}(T_j^{rep}) + D + R_i^A \\ + \min\left\{ T_{NC}^{A,rep}(i,j-1), T_{NC}^{A,norep}(i,j-1) \right\} + T^{A,rep}(j \mid i)\Big) \\ + \left(1 - \left(1 - e^{-\frac{\lambda T_j^{rep}}{2}}\right)^2\right)\left(T_j^{rep}\right)$$

Complexity

- Compute $O(n^2)$ intermediate values $T^{A,B}(j \mid i)$ and $T_{NC}^{A,B}(i,j)$ for $1 \leq i,j \leq n$ and $A, B \in \{rep, norep\}$
- Each of these take constant time
- $O(n)$ values $T_{opt}^{A}(i)$, for $1 \leq i \leq n$ and $A \in \{rep, norep\}$
- Minimum over at most $6n$ elements: $O(n)$
- Overall complexity: $O(n^2)$

Outline

Experimental setup

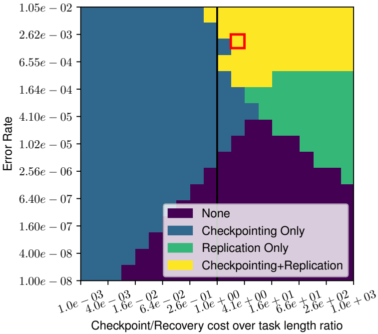
- Total work: $W = 10,000$ seconds
- Fully parallel tasks: $\alpha_i = 0$ (worst case for replication)

- Five work distributions:
  - UNIFORM: Identical tasks, $\frac{W}{n}$
  - INCREASING: length increases: $i\frac{2W}{n(n+1)}$
  - DECREASING: length decreases: $(n - i + 1)\frac{2W}{n(n+1)}$
  - HIGHLOW: $\lceil \frac{n}{10} \rceil$ big tasks (60% of work) followed by small tasks
  - RANDOM: random lengths between $\frac{W}{2n}$ and $\frac{3W}{2n}$, reduced if it exceeds W

- $C_i^{rep} = \alpha C_i^{norep}$ and $R_i^{rep} = \alpha R_i^{norep}$, where $1 \leq \alpha \leq 2$

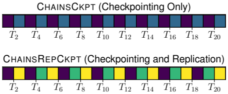
Comparison to checkpoint only

- UNIFORM distribution
- Reports occ. of checkpoints and replicas in optimal solution
- Checkpointing cost $\leq$ task length $\Rightarrow$ no replication

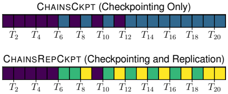


Optimal solutions with both strategies

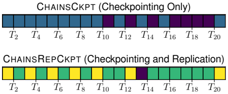
- Scenario of the red square on the previous slide
- Less checkpoints when replication is used
- Optimal solution combines both techniques
- Rule of thumb: replication preferred for small tasks



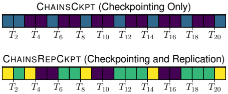
(a) Uniform



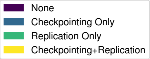
(b) Increasing

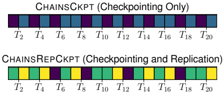


(c) Decreasing



(d) HighLow

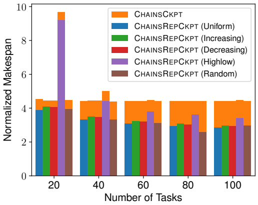


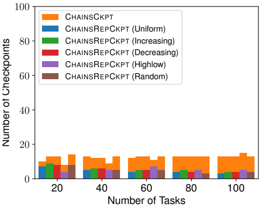


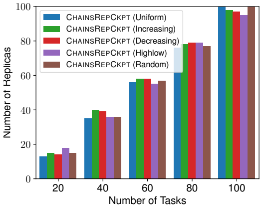
(e) Random

Comparison, different numbers of tasks

- Performance of ChainsRepCkpt compared to ChainsCkpt
- Expensive checkpoints (limited to ≈ 17) $\Rightarrow$ makespan of ChainsCkpt remains constant
- ChainsRepCkpt can replicate increasing number of small tasks

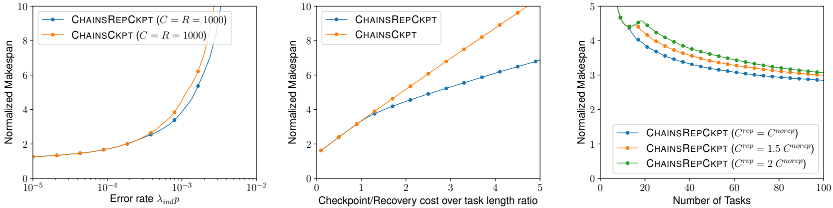






Impact of error rate and checkpoint cost

- Larger error rate ⇒ using replication helps
- Replication not needed for small checkpointing costs
- Replication more efficient when no increase in checkpoint cost



Further experiments

- With increasing number of processors and variable checkpointing costs: improvement up to 80.5% with $p = 10,000$ processors
- Impact of number of checkpoints and replicas: the optimal solution always matches minimum value obtained in simulations
- When both checkpointing cost and error rate are high, small deviation from optimal solution leads to large overhead

Outline

Conclusion

- Combination of checkpointing and replication
- Goal: Minimize execution time of linear workflows
- Decide which task to checkpoint and/or replicate
- Sophisticated dynamic programming algorithm: optimal solution
- Experiments: Gain over checkpoint-only approach quite significant, when checkpoint is costly and error rate is high

- Extend to more complicated workflows
- Experiments on real application workflows
- Cope with silent errors as well as fail-stop errors

Conclusion

- Combination of checkpointing and replication
- Goal: Minimize execution time of linear workflows
- Decide which task to checkpoint and/or replicate
- Sophisticated dynamic programming algorithm: optimal solution
- Experiments: Gain over checkpoint-only approach quite significant, when checkpoint is costly and error rate is high

- Extend to more complicated workflows
- Experiments on real application workflows
- Cope with silent errors as well as fail-stop errors