NANYANG TECHNOLOGICAL UNIVERSITY

School of Computer Engineering

Presented by:
Yifan Gong, Amelie Chi Zhou, Bingsheng He

Monetary Cost Optimizations for HPC Applications on Amazon Clouds: Checkpoints and Replicated Execution

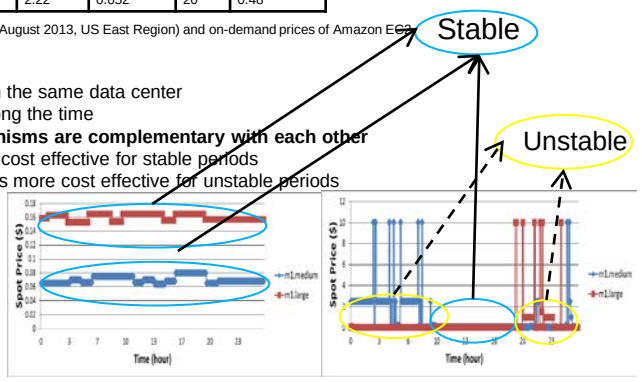
Motivation

- **Spot instances provided by Amazon EC2 can reduce monetary cost for its low prices (Table 1).**

Instance type	Average	stdev	Min	Max	On Demand
m1.small	0.048	0.438	0.007	10	0.06
m1.medium	0.246	1.31	0.0001	10	0.12
m1.large	0.069	0.770	0.026	40	0.24
m1.xlarge	0.413	2.22	0.052	20	0.48

Table 1: Statistics on spot prices ($/hour, August 2013, US East Region) and on-demand prices of Amazon EC2

- **Spot price is dynamic**
  - ❑ Spatial: variance within the same data center
  - ❑ Temporal: variance along the time
- **Traditional fault-tolerant mechanisms are complementary with each other**
  - ❑ Checkpointing is more cost effective for stable periods
  - ❑ Replicated execution is more cost effective for unstable periods



Problem Statement

Minimize monetary cost of an MPI application with deadline requirement.

SOMPI Overview

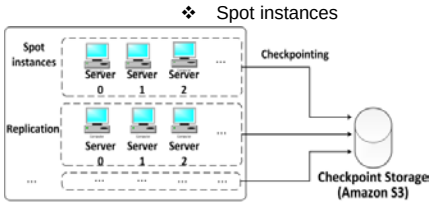
SOMPI is an MPI runtime system that minimizes the monetary cost on Amazon EC2 clouds.

•Runtime system design

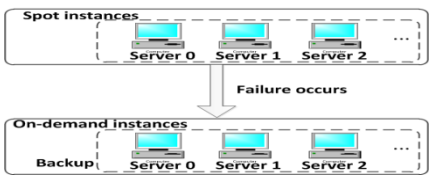
- ❑ Profiler: The profiler extracts application runtime features from profiling run
- ❑ Planner: The planner ddetermines the various design issues of fault-tolerant mechanisms
- ❑ Scheduler: The scheduler is responsible for managing the on-demand and spot instances, including acquiring/releasing instances.
- ❑ Monitor: The monitor tracks the runtime state when the application is running on different replications. We handle the failures on spot instances according to fault-tolerant mechanisms.

•Key Techniques

- ❑ A holistic approach of combining two fault-tolerant techniques
  - ❖ Checkpointing
  - ❖ Replicated execution
- ❑ Hybrid execution: first leverage spot instances if deadline allows, and uses on-demand instances as the last line of defense.
  - ❖ On-demand instances
  - ❖ Spot instances

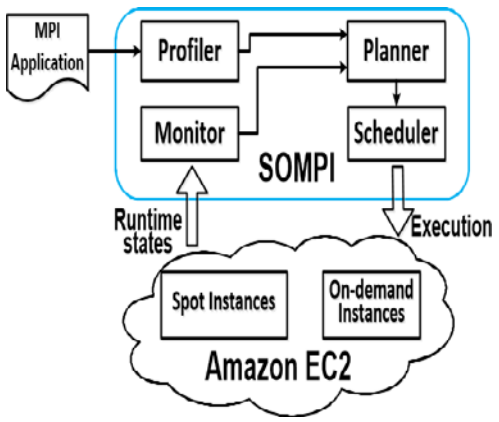


(a) Combining checkpointing and replication techniques



(b) Hybrid execution

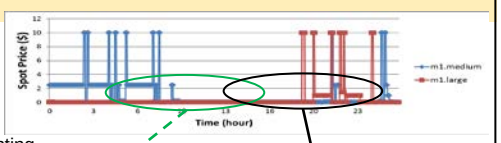
Main techniques of SOMPI



The system design of SOMPI

Technique Details

- **Checkpointing**
- ❑ We use system level coordinated approaches
- ❑ **Design issues**
  - ❖ When to adopt checkpointing techniques
  - ❖ Where to store checkpointing data
  - ❖ How to design the frequency of checkpointing



- **Replicated Execution**
- ❑ We use Active replication
- ❑ **Design issues**
  - ❖ When to use replication
  - ❖ How many replications to use

Replication is useful

Checkpointing is useful

How to resolve those complicated issues?

We have developed a prediction model to determine those design issues for cost effectiveness.

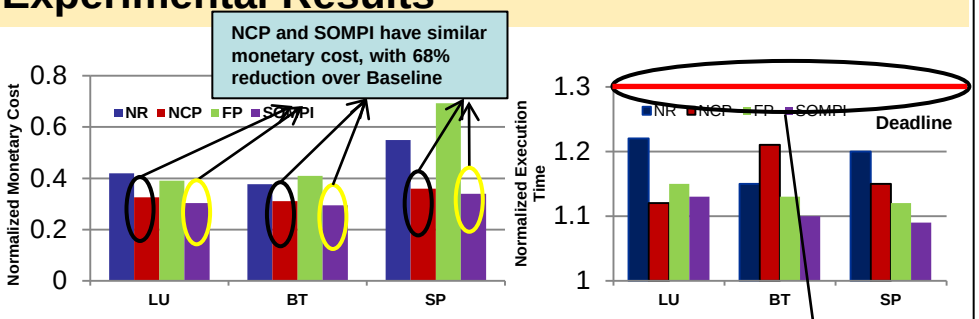
Prediction Model Overview

- **Hybrid execution design**
  - ❑ Determine the instance type for spot instances and on-demand instance for MPI executions so that the deadline is satisfied.
- **Replication design**
  - ❑ Estimate the number of ***circle groups*** for replications.
- **Bid price and frequency of checkpointing**
  - ❑ Predict the **Failure rate function F(P,t)** based on the trace. F(P,t) is the probability of a instance which fails at time t for the first time when bid price is P
  - ❑ Estimate the **expected monetary cost** when we fixe the bid price as P and the frequency of checkpointing as F
- **Optimization problem formulation**
  - ❑ Calculate the minimal expected monetary cost with the deadline constrain when we vary P and F.
  - ❑ **We formulate the problem as a nonlinear optimization problem with nonlinear constraints,** which can be solved in a timely manner.
- **With the model, SOMPI automatically decides the usage of two fault-tolerant techniques**
  - ❑ When P=0, we do not use replications. when F is 0, we do not use checkpointing.
  - ❑ In other cases, we combine checkpointing and replicated execution techniques together

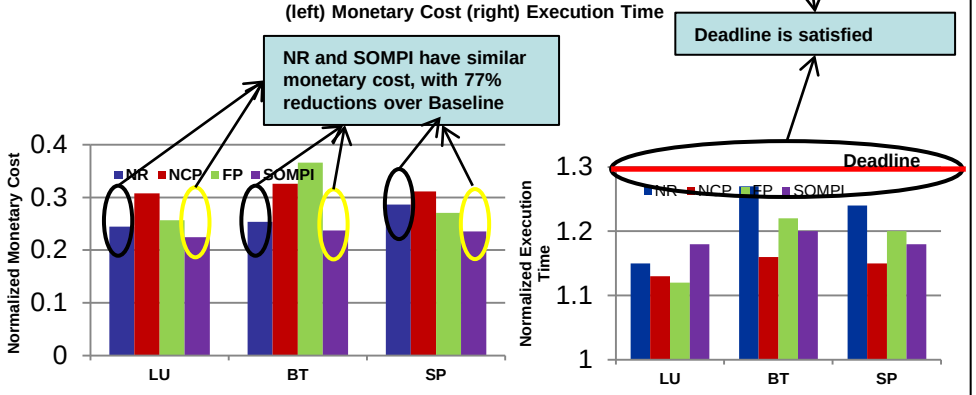
Experimental Setup

- **Platform:** We conduct the experiment on Amazon EC2 with 128 processes
- **Applications:** We apply our middleware system to NAS Parallel Benchmarks (NPB) kernels version 2.4. All the applications are compiled with Class E.
  - ❑ **BT:** Block Tri-diagonal solver; **SP**: Scalar Penta-diagonal solver; **LU:** Lower-upper Gauss-Seidel solver
- **Comparisons:**
  - ❑ **Baseline** executes the applications on on-demand m1.medium instances
  - ❑ **FP** fixes the bid price of the spot instances as the on-demand price of the same type.
  - ❑ **NR** does not use the replication.
  - ❑ **NCP** executes the applications without checkpoint.
  - ❑ **SOMPI** combines checkpointing and replicated execution techniques. The design is guiled by a cost model

Experimental Results



The overall comparison for different approaches (normalized to Baseline) in stable trace: (left) Monetary Cost (right) Execution Time



The overall comparison for different approaches (normalized to Baseline) in unstable trace: (left) Monetary Cost (right) Execution Time

Conclusion

- We propose a practical runtime system SOMPI with monetary cost optimizations on Amazon EC2 clouds. We take advantages of Amazon EC2 spot instances and on-demand instances.
- We propose a cost model to guide the combination of checkpointing and replication techniques to improve the cost effectiveness.
- More details in our project site: http:///pdcc.ntu.edu.sg/xtra/tr/2014-TR-120-SOMPI.pdf