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).

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

<table>
<thead>
<tr>
<th>Instance type</th>
<th>Average</th>
<th>rate</th>
<th>Min</th>
<th>Max</th>
<th>On Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>0.048</td>
<td>0.040</td>
<td>0.003</td>
<td>0.049</td>
<td></td>
</tr>
<tr>
<td>m1.medium</td>
<td>0.240</td>
<td>0.770</td>
<td>0.026</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>m1.large</td>
<td>0.410</td>
<td>2.22</td>
<td>0.052</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>0.410</td>
<td>2.22</td>
<td>0.052</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

- 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
  - 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 determines the various design issues of fault-tolerant mechanisms
  - On-demand and spot instances
  - Acquiring/releasing instances
- Monitor: The monitor tracks the runtime state when the application is running on different replications

**Key Techniques**
- 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

**Prediction Model Overview**
- Hybrid execution design
  - Determines 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 an instance which fails at time t for the first time when bid price is P
  - Estimate the expected monetary cost when we fix the bid price as P and the frequency of checkpointing as F
- Optimization problem formulation
  - Calculates the minimal expected monetary cost with the deadline constraint when we vary P and F
  - We formulate the problem as a non-linear optimization problem with non-linear 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.
- 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.

**Experimental Results**
- NCP and SOMPI have similar monetary cost, with 68% reduction over Baseline
- NR and SOMPI have similar monetary cost, with 77% reductions over Baseline

**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.