Decision Model for Cloud Computing under SLA Constraints

Artur Andrzejak\textsuperscript{1}, Derrick Kondo\textsuperscript{2}, Sangho Yi\textsuperscript{2}

\textsuperscript{1}Zuse Institute Berlin, but now at Institute for Infocomm Research (I2R), Singapore

\textsuperscript{2}INRIA Grenoble, France
Trade-offs

- **Cost ($)**
  - Low
  - High

- **Reliability**
  - Low
  - High

- **Performance**

  - **Supercomputers**
  - **Clusters**
  - **Desktop Grids**

- **Cloud Computing**
Market-based Resource Allocation Systems

- Amazon Spot Instances
- “Spot” instance price varies dynamically
- Spot instance provided when user’s bid is greater than current price
- Spot instance terminated when user’s bid ≤ current price
- Amazon charges by the last price at each hour

Synthetic Example:

Real Amazon Price Trace:

cloudexchange.org [tim lossen]
Optimization Problem

• Given job with batch of parallel, independent, divisible tasks

• Deadline and budget constraints

• Objectives

  • Can the job be executed under budget and deadline constraints?

  • What is the bid price and instance type that minimizes the total monetary costs?

  • What is the distribution of monetary costs and execution times for a specific instance type and bid price?
Goal and Approach

• Formulate and show how to apply user decision model

• Characterize relationship between job execution time, monetary cost, reliability, bid price

• Compare costs of different instance types
Outline

- System model
- Decision model
- Simulations method and results
- Related work
- Conclusion & Future work
## User Parameters and Constraints

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_{inst}$</td>
<td>number of instances that process the work in parallel</td>
</tr>
<tr>
<td>$n_{max}$</td>
<td>upper bound on $n_{inst}$</td>
</tr>
<tr>
<td>$W$</td>
<td>total amount of work in the user’s job</td>
</tr>
<tr>
<td>$W_{inst}$</td>
<td>workload per instance ($W/n_{inst}$)</td>
</tr>
<tr>
<td>$T$</td>
<td>task length, time to process $W_{inst}$ on a specific instance</td>
</tr>
<tr>
<td>$B$</td>
<td>budget per instance</td>
</tr>
<tr>
<td>$c_B$</td>
<td>user’s desired confidence in meeting budget $B$</td>
</tr>
<tr>
<td>$t_{dead}$</td>
<td>deadline on the user’s job</td>
</tr>
<tr>
<td>$c_{dead}$</td>
<td>desired confidence in meeting job’s deadline</td>
</tr>
<tr>
<td>$u_b$</td>
<td>user’s bid on a Spot Instance type</td>
</tr>
<tr>
<td>$I_{type}$</td>
<td>EC2 instance type</td>
</tr>
</tbody>
</table>

**Notation**

- **Job parameters**
- **Job constraints**
- **User decision variables**
Random Variables of Model

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ET$</td>
<td>execution time of the job (clock time)</td>
</tr>
<tr>
<td>$AT$</td>
<td>availability time (total time in-bid)</td>
</tr>
<tr>
<td>$EP$</td>
<td>expected price, i.e. (cost per instance)/$AT$</td>
</tr>
<tr>
<td>$M$</td>
<td>monetary cost $AT \cdot EP$ per instance</td>
</tr>
<tr>
<td>$AR$</td>
<td>availability ratio $AT/ET$</td>
</tr>
<tr>
<td>$UR$</td>
<td>utilization ratio $T/ET$</td>
</tr>
</tbody>
</table>

- **$ET$**: Execution time of the job (clock time)
- **$AT$**: Availability time (total time in-bid)
- **$EP$**: Expected price, i.e. (cost per instance)/$AT$
- **$M$**: Monetary cost $AT \cdot EP$ per instance
- **$AR$**: Availability ratio $AT/ET$
- **$UR$**: Utilization ratio $T/ET$
Execution Model Example

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Time (hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Useful computation</td>
<td>Computation happening</td>
<td>0-6</td>
</tr>
<tr>
<td>Checkpoint</td>
<td>Required for successful</td>
<td>3</td>
</tr>
<tr>
<td>Restart</td>
<td>Attempt to restart an</td>
<td>5</td>
</tr>
<tr>
<td>Useful comp.</td>
<td>Computation happening with</td>
<td>6-10</td>
</tr>
<tr>
<td>Failure</td>
<td>Failure occurs</td>
<td>4-5</td>
</tr>
<tr>
<td>Availability</td>
<td>Time available for</td>
<td></td>
</tr>
</tbody>
</table>

Notation:
- $T$: Total job time
- $ET$: Expected job time
- $AT$: Actual job time
- $EP$: Expected price
- $M$: Total cost
- $AR$: Availability ratio
- $UR$: Utilization ratio

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Calculation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$6h$</td>
<td></td>
</tr>
<tr>
<td>$ET$</td>
<td>$10h$</td>
<td></td>
</tr>
<tr>
<td>$AT$</td>
<td>$8h$</td>
<td></td>
</tr>
<tr>
<td>$EP$</td>
<td>$0.175 USD/h$</td>
<td></td>
</tr>
<tr>
<td>$M$</td>
<td>$3<em>0.1 + 4</em>0.2 + 1*0.3$</td>
<td>$1.4 USD$</td>
</tr>
<tr>
<td>$AR$</td>
<td>$8/10$</td>
<td>$0.8$</td>
</tr>
<tr>
<td>$UR$</td>
<td>$6/10$</td>
<td>$0.6$</td>
</tr>
</tbody>
</table>
Decision Workflow

Submission with job parameters, and time and budget constraints

Broker applying decision model

Feasible?

Yes, get bid to achieve lowest cost or execution time, then deploy.

No, revise constraints

Amazon EC2 Spot Market
Decision Model

• For a random variable, $X$, we write $X(y)$ for $x$ s.t. $\Pr(X < x) = y$.
  
  • E.g. $\text{ET}(0.50)$ is the median execution time

• Feasibility decisions

  • Deadline constraint achievable with confidence
    $c_{\text{dead}} \Leftrightarrow t_{\text{dead}} \geq \text{ET}(c_{\text{dead}})$

  • Budget constraint achievable with confidence
    $c_B \Leftrightarrow B \geq M(c_B)$

• Among the feasible cases, we choose the one with the smallest $M(c_B)$ or lowest execution time $\text{ET}(c_{\text{dead}})$
Outline

• System model
• Decision model
• Simulations method and results
• Related work
• Conclusion & Future work
Simulation Method

- Determine distributions of model variables via price trace-driven simulation
- Prices: trace of Spot instance prices obtained from Amazon
- Workload model
  - W1: “Big”, based on Desktop Grids, parameters derived from BOINC catalog
  - W2: “Small”, based on Grids, parameters derived from the Grid Workload Archive

<table>
<thead>
<tr>
<th>Workload</th>
<th>$I_{type}$</th>
<th>$n_{max}$</th>
<th>$W_{inst}$</th>
<th>$T$</th>
<th>$t_{dead}$</th>
<th>$c_{dead}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>2.5GHz</td>
<td>20,000</td>
<td>11.5</td>
<td>4.6h</td>
<td>9d</td>
<td>0.9</td>
</tr>
<tr>
<td>W2</td>
<td>2.5GHz</td>
<td>50</td>
<td>6.83</td>
<td>2.7h</td>
<td>17.9h</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Experiments

• 10,000 simulation experiments per set of unique input parameters ($l_{\text{type}}, u_b, T$)

• Experiment corresponds to single task execution

• Starting point randomly selected during period [Jan 11, Mar 18] of Amazon’s Spot Instance price traces
Experiments

- Run simulations for all 7 types of instances
- If not stated otherwise, show results for instance type A, with task length $T$ of 276 minutes (W1) and hourly checkpointing

Table IV

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Class</th>
<th>API Name</th>
<th>Mem. (GB)</th>
<th>Total Units</th>
<th>Num. Cores</th>
<th>Units / Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>hi-cpu</td>
<td>c1.medium</td>
<td>1.7</td>
<td>5</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>B</td>
<td>hi-cpu</td>
<td>c1.xlarge</td>
<td>7</td>
<td>20</td>
<td>8</td>
<td>2.5</td>
</tr>
<tr>
<td>C</td>
<td>std</td>
<td>m1.small</td>
<td>1.7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>std</td>
<td>m1.large</td>
<td>7.5</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>std</td>
<td>m1.xlarge</td>
<td>15</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>hi-mem</td>
<td>m2.2xlarge</td>
<td>34.2</td>
<td>13</td>
<td>4</td>
<td>3.25</td>
</tr>
<tr>
<td>G</td>
<td>hi-mem</td>
<td>m2.4xlarge</td>
<td>68.4</td>
<td>26</td>
<td>8</td>
<td>3.25</td>
</tr>
</tbody>
</table>
Assumptions

• Checkpointing
  • OPT: optimal checkpointing taken just before failure
  • HOUR: hourly checkpointing taken at each paid hour

• Use only 1 instance type per job
  • Note that deployment time, costs, failures are identical for all instances deployed in parallel.
Execution Time & Monetary Cost

Figure 04 Influence of bid price and percentile $p$ on the execution time $ET(p)$ and monetary cost per instance in USD.

Figure 14 Overhead $(AT(p) - T)$ of the checkpointing for the strategy OPT and HOUR for various bid prices and percentiles.

C. Meeting Deadline and Budgetary Constraints for W1

In this section we study distributions of the execution time and the monetary constraints for the workload $W_8$ (Section II3D) and also demonstrate how these interplay with the constraints introduced in Section II3B4.

Figure ( shows the cumulative distribution function (CDF) of the execution time $ET$ and the monetary costs per instance $M$ according to different values of the bid price $u$ and the task length $T$.

The red vertical lines represent the given deadline $t_{\text{dead}}$ while the blue horizontal lines represent their required confidence $c$. In the results of Figure ( the lowest bid price $(747\text{USD})$ cannot meet the user's given deadline constraints $t_{\text{dead}}$ and $c_{\text{dead}}$, while the two highest bid prices $(747\text{USD})$ cannot meet the budget limit constraints $B$ and $c_B$. Note that the value of $T$ also affects the possible range of bid prices.

The constraints $t_{\text{dead}}$ and $c_{\text{dead}}$ act as a high-pass filter of possible bid prices, and the other constraints $B$ and $c_B$ act as a low-pass filter. Figure ) shows the range of bid prices according to all results in Figure (4. As we can observe from Figure ), some bid prices are not feasible. In these cases we need to either decrease the confidence values or set higher limits on the deadline and the budget.

Figure " shows the CDF of the execution time and the monetary cost for $T = 246$ minutes. Table V shows the lowest monetary costs in the case of this figure according to different values of $c_{\text{dead}}$.

This result demonstrates that the total costs can be significantly affected by changing the degree of the confidence value. By comparing the two cases $c_{\text{dead}} = 0.90$ and $c_{\text{dead}} = 0.82$, we observe that using slightly lower confidence can reduce more than $21\%$ of the monetary costs.
Execution Time & Monetary Cost

Deadline of 200h with conf. 0.80 given bid of $0.076

Execution time

Monetary cost
Execution Time & Monetary Cost

Low bid prices can cause spike in ET due to pricing “band”

Deadline of 200h with conf: 0.80 given bid of $0.076

Execution time

Monetary cost
Low bid prices can cause spike in ET due to pricing “band”

Deadline of 200h with conf: 0.80 given bid of $0.076

Cost 0.40 with conf of 0.99 given bid of $0.079

Execution time

Monetary cost
Execution Time & Monetary Cost

Low bid prices can cause spike in ET due to pricing “band”

Deadline of 200h with conf: 0.80 given bid of $0.076

Cost decreases slightly with bid price

Cost 0.40 with conf of 0.99 given bid of $0.079

Execution time

Monetary cost
Execution Time & Monetary Cost

Low bid prices can cause spike in ET due to pricing “band”

Deadline of 200h with conf: 0.80 given bid of $0.076

Cost decreases slightly with bid price

Cost 0.40 with conf of 0.99 given bid of $0.079

Execution time

Monetary cost

User can save about 10% in costs when bidding low but ET can be much higher
Other Relationships

• Checkpointing overhead versus bid price
  • HOUR checkpointing can amount to over 40% of task length T for low bids and availability confidence p

• AR versus task length T
  • AR is function of task length and bid price, and must consider both factors in decision model

• Coefficient of variation of ET versus task length T
  • Decreases sub-linearly with T. Can be used to determine “slack” between deadline and T.
Figure 2: CDF of execution time and monetary cost for various task lengths on instance type A and workload W.

The typical price range ["Low Bid", "High Bid"] has been determined on the price history from January 2887 to March 2007. We plotted this history as in Figure 8, removed obviously anomalous prices (high peaks or long intervals of constant prices), and took the minimum "Low Bid" and maximum "High Bid". The last column "Ratio in s" shows $(H - L)/L * 100$ per instance type, i.e., the range of bid prices divided by "Low Bid" in s. This answers the first question—the variation of the typical bid prices is only about 10 s to 12 s across all instance types.

In Table VI, column "Low 4 Unit" shows the "Low Bid" price divided by the total number of EC0 computing units (units) of this instance type. The column "High 4 Unit" is computed analogously. For workload types assumed here, this allows one to estimate the cost of processing one unit/hour in US cents, disregarding the checkpointing and failure-recovery overheads. Obviously, instance types within the high-CPU class [:] have lowest cost of unit/hour, only about 40 s of the standard class. For the high-memory instance types, a user has to pay a small premium, approx 8 s more than for the standard class.

Interestingly, all instance types within each class have almost identical cost of one unit/hour. In summary, switching to a high-CPU class (if amenable to the workload typew can reduce the cost of unit/hour by approx 60 s while bidding low saves only 10 s of the cost, with a potentially extreme increase of the execution time.

**Distribution of Execution Time and Costs**

**(Instance Type A and Workload W1)**

Deadline = 12960 minutes (9 days)

Budget per instance = 0.40 USD

Confidence = 0.9

(b) when task length T = 276 minutes
Distribution of Execution Time and Costs (Instance Type A and Workload W1)

Deadline = 12960 minutes (9 days)

Pr (ET <= 4800m) = 0.90 with bid of 0.082

Budget per instance = 0.40 USD

Confidence = 0.9

(b) when task length T = 276 minutes
Distribution of Execution Time and Costs
(Instance Type A and Workload W1)

- Deadline = 12960 minutes (9 days)
- Pr (ET ≤ 4800m) = 0.90 with bid of 0.082
- Pr (M ≤ 0.38) = 0.90 with bid of 0.076

(b) when task length T = 276 minutes
Distribution of Execution Time and Costs (Instance Type A and Workload W1)

(b) when task length $T = 276$ minutes
The typical price range ["Low Bid" - "High Bid"] has been determined on the price history from January 28, 88 to March 28, 87 (we plotted this history as in Figure 8, removed obviously anomalous prices - high peaks or long intervals of constant prices - and took the minimum \( L \) "Low Bid" and maximum \( H \) "High Bid"). The last column "Ratio in s" shows \( \frac{H - L}{L} \times 100 \) per instance type, i.e., the range of bid prices divided by "Low Bid" in "s". This answers the first question: the variation of the typical bid prices is only about 10 "s to 12 "s across all instance types. In Table VI, column "Low 4 Unit" shows the "Low Bid" price divided by the total number of EC0 Computing Units "units" of this instance type. The column "High 4 Unit" is computed analogously. For workload types assumed here, this allows one to estimate the cost of processing one unit "hour" in US "cent" disregarding the checkpointing and failure "recovery" overheads. Obviously, instance types within the high "CPU" class have lowest cost of unit "hour" - only about 40 "s of the standard class. For the high "memory" instance types, a user has to pay a small premium - approx 8 "s more than for the standard class. Interestingly, all instance types within each class have almost identical cost of one unit "hour". In summary, switching to a high "CPU" class (if amenable to the workload type) can reduce the cost of unit "hour" by approx 60 "s while bidding low saves only 10 "s of the cost, with a potentially extreme increase of the execution time.

Distribution of Execution Time and Costs (Instance Type A and Workload W1)
The "typical" price range \([\text{"Low Bid"}, \text{"High Bid"}]\) has been determined on the price history from Jan 2888 to March 2007 (we plotted this history as in Figure 8, removed obviously anomalous prices (high peaks or long intervals of constant prices) and took the minimum \(L\) ("Low Bid") and maximum \(H\) ("High Bid"). The last column "Ratio in s" shows \((H - L) / L \times 100\) per instance type, i.e., the range of bid prices divided by "Low Bid" in s. This answers the first question—the variation of the typical bid prices is only about 10 s to 12 s across all instance types.

In Table VIIf column "Low 4 Unit" shows the "Low Bid" price divided by the total number of EC2 Computing Units (units) of this instance type. The column "High 4 Unit" is computed analogously. For workload types assumed here, this allows one to estimate the cost of processing one unit/hour in US cents disregarding the checkpointing and failure-recovery overheads. Obviously, instance types within the high-CPU class [:] have lowest cost of unit/hour only about 40 s of the standard class. For the high-memory instance types a user has to pay a small premium, approx 8 s more than for the standard class. Interestingly, all instance types within each class have almost identical cost of one unit/hour. In summary, switching to a high-CPU class (if amenable to the workload typew can reduce the cost of unit/hour by approx 60 s while bidding low saves only 10 s of the cost, with a potentially extreme increase of the execution time.

**Distribution of Execution Time and Costs (Instance Type A and Workload W1)**

(b) when task length \(T = 276\) minutes
Distribution of Execution Time and Costs (Instance Type A and Workload W1)

(b) when task length $T = 276$ minutes

Possible range of bid prices $B$ and $c_B$

(b) $T = 276$
Distribution of Execution Time and Costs (Instance Type A and Workload W1)

(b) when task length $T = 276$ minutes

$T_{dead}$, $c_{dead}$: high-pass filter

B, $c_B$: low-pass filter

(b) $T = 276$
Gains in Adjusting Budget or Deadline Constraints

• Slightly changing confidence value for deadline can significantly reduce costs

• For $c_{\text{dead}} = 0.90$ and $c_{\text{dead}} = 0.82$, we observe that using slightly lower confidence can reduce more than 21% of the monetary costs.

• Slightly changing budget or its confidence can have significant effect on execution time
Comparing Instance Types

Bidding price comparison across instance types (in US-cents)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Class</th>
<th>Total Units</th>
<th>Low Bid</th>
<th>High Bid</th>
<th>Low / Unit</th>
<th>High / Unit</th>
<th>Ratio in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>hi-cpu</td>
<td>5</td>
<td>7.6</td>
<td>8.4</td>
<td>1.52</td>
<td>1.68</td>
<td>10.5</td>
</tr>
<tr>
<td>B</td>
<td>hi-cpu</td>
<td>20</td>
<td>30.4</td>
<td>33.6</td>
<td>1.52</td>
<td>1.68</td>
<td>10.5</td>
</tr>
<tr>
<td>C</td>
<td>std</td>
<td>1</td>
<td>3.77</td>
<td>4.2</td>
<td>3.77</td>
<td>4.2</td>
<td>11.4</td>
</tr>
<tr>
<td>D</td>
<td>std</td>
<td>4</td>
<td>15</td>
<td>16.8</td>
<td>3.75</td>
<td>4.2</td>
<td>12.0</td>
</tr>
<tr>
<td>E</td>
<td>std</td>
<td>8</td>
<td>30.4</td>
<td>33.6</td>
<td>3.8</td>
<td>4.2</td>
<td>10.5</td>
</tr>
<tr>
<td>F</td>
<td>hi-mem</td>
<td>13</td>
<td>53.3</td>
<td>58.8</td>
<td>4.1</td>
<td>4.52</td>
<td>10.3</td>
</tr>
<tr>
<td>G</td>
<td>hi-mem</td>
<td>26</td>
<td>106</td>
<td>118</td>
<td>4.08</td>
<td>4.54</td>
<td>11.3</td>
</tr>
</tbody>
</table>

Ratio in % = \( \frac{(H-L)}{L} \times 100 \)

Price variation is about 10-12% (likely due to regulation)

High CPU instances have lowest cost per unit hour and saves \(~60\%\) wrt standard instance
Related Work

• Batch schedulers
  • Don’t have decision models that consider costs, performance, and reliability for scheduling

• Cloud services and monitoring
  • Don’t make bidding recommendations

• Cloud economics
  • Don’t take into account dynamic and tunable pricing or availability
Summary

• Largest cost savings achieved by using high-CPU instance types instead of standard or high-memory instance types

• Bidding low prices yields cost savings of about 10%, but can lengthen execution time significantly

• Our model allows a user to turn several knobs (parameters) to achieve desired balance between monetary costs and service levels (job deadlines or reliability).

• Data, simulator, results available at:
  • http://spotmodel.sourceforge.net
Future Work

• Include mixes of instances (wrt size, availability zones, dynamically adjusting number of instances, rebidding and restarting)

• Price prediction or reverse engineering of pricing scheme

• Implement web service and middleware that applies it
Thank you