More for Your Money: Exploiting Performance Heterogeneity in Public Clouds

Presenter: Ting Xie, tingxie2@illinois.edu
Motivation:

• We pay for virtual resources (e.g. computing cycles, storage, virtual machine instance), but logically equal resources may have different performance due to underlying hardware, contention, etc.

• ‘Good’ resources may not stay ‘Good’ all the time.

• Goal:
  Design strategies to dynamically select ‘good’ resources over time in order to maximize performance.
Subject of Analysis

- Subject: Amazon EC2 service in US-East Region
- Resources: EC2 instances
- Pricing: flat hourly rate to run VMs
- Strategy type: placement gaming (remove or add instances to the current working set)
- Trade-off: performance gain versus overhead of launching new instances
- Question: Is performance difference large enough to motivate our strategies?
Experiment on Performance Heterogeneity

• Three types of heterogeneity:
  1. Inter-architecture 2. Intra-architecture 3. Temporal

• BenchMarks used:
  CPU: slurp and Nqueens
  Memory: mcf and sphinx3
  Outbound bandwidth: iperf
  Storage: Bonnie++ on both Elastic Block Store and local disk which uses block reads and block writes

Metric: normalized speedup relative to worst performing architecture
Conclusions

- The maximum speedup, in percent, of best performance over the worst

<table>
<thead>
<tr>
<th>Resource</th>
<th>Inter-arch.</th>
<th>Intra-arch.</th>
<th>Intra-node</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>282%</td>
<td>15%</td>
<td>13%</td>
</tr>
<tr>
<td>Network</td>
<td>88%</td>
<td>125%</td>
<td>25%</td>
</tr>
<tr>
<td>Local Disk</td>
<td>67%</td>
<td>269%</td>
<td>20%</td>
</tr>
<tr>
<td>EBS</td>
<td>33%</td>
<td>106%</td>
<td>41%</td>
</tr>
</tbody>
</table>
Problem Formation

• Placement Model
  1. Quantum: min granularity of replacement intervals
     (here is 1 hour==min billing unit in Amazon EC2)
  2. A placement schedule $P$ is a sequence of sets of instances.
     Let $P[t]$ be the set of instances on time quantum $t$.
  3. Metric of Performance: rate of job execution of instance $S$ at quantum $t$ is $r_t(S)$. (job-specific, assume independence over each instance $S$)
  4. For a placement schedule $P$ of duration $T$, the cost is the sum of set cardinality of each set in the sequence.
  5. time penalty $m$ (overhead of launching a new instance): the fraction of the first quantum consumed
  6. support of a placement schedule $P$: min num of instances preserved in any quantum.
Problem Formation

• Placement Strategies
  For each quantum t:
  1. which instances to keep: $K[t]$ set of remaining instances
  2. which fresh instances to launch: set $F[t]

Input: $P[t-1]$  output: $K[t]+F[t]$ (assume markov)

Various Objectives:
  1. minimize cost or latency given work load
  2. bound the cost and maximize work done
  3. maximize efficiency (work/cost)

(min and max are in the sense of expected value or worst case value)
Problem Formation

Based on restricted strategy space \((A,B)\)-strategies (fixed cost \(TA+B\))

- Types of Strategies
  1. Up-front exploration and Opportunistic replacement
  2. Gray-box strategies versus Black-box strategies
Concrete Design: PERF-M

A black box strategy that combines Up-front and Opportunistic Algorithm:

1. launch A+B instance at t=0

2. Keep top A, kick out those in Top A whose performance fall under threshold $\delta = \text{avg} - \frac{2m}{T-t}$, add the same number of fresh ones to reach the minimum requirement of A instances

3. for subsequent quantum, kick those fall under threshold and adds back the same number of instances.
Synthetic Simulations

• How major changes in cloud affect the strategy

1. Architecture performance variation
   If this dominates, then Up-front works better

2. Instance performance variation
   If this dominates, then opportunistic better

3. Distribution of different architectures
   if ‘good’ machine dominates then up-front best
   if ‘bad’ dominates, up-front performance grows linearly with parameter B
   for opportunistic, complex, but seems bell shaped.
Synthetic Simulations

- Simulating the effect of Gaming Contention

1. First group achieves first-mover advantage
2. As number of participants grows, first group’s performance decreases
3. First group’s gaming strategy decreases the performance of second group
4. Second group can ‘catch up’ by opportunistic replacement if total ‘good’ machines outnumber requested ones

Questions:
Different jobs and perf metrics will have different definition of ‘good’
No overall system performance simulated as m increases
Real World Application

• Strategies used (gray-box and black box)

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Up-front Exploration</th>
<th>Opportunistic replacement rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU</td>
<td>Never</td>
</tr>
<tr>
<td>CPU-M</td>
<td>CPU</td>
<td>$\bar{C} - cpu(S) \geq \delta$</td>
</tr>
<tr>
<td>CPU-MAX</td>
<td>CPU</td>
<td>$cpu(S) \neq \max_i{cpu(S_i)}$</td>
</tr>
<tr>
<td>PERF</td>
<td>Performance</td>
<td>Never</td>
</tr>
<tr>
<td>PERF-M</td>
<td>Performance</td>
<td>$\overline{avg}_1 - r_t(S) \geq \delta$</td>
</tr>
</tbody>
</table>
Real World Application

- Strategies used (gray-box and black box)

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Up-front Exploration</th>
<th>Opportunistic replacement rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU</td>
<td>Never</td>
</tr>
<tr>
<td>CPU-M</td>
<td>CPU</td>
<td>$\bar{C} - cpu(S) \geq \delta$</td>
</tr>
<tr>
<td>CPU-MAX</td>
<td>CPU</td>
<td>$cpu(S) \neq \max_i{cpu(S_i)}$</td>
</tr>
<tr>
<td>PERF</td>
<td>Performance</td>
<td>Never</td>
</tr>
<tr>
<td>PERF-M</td>
<td>Performance</td>
<td>$\overline{avg}_1 - r_t(S) \geq \delta$</td>
</tr>
</tbody>
</table>
Experiments on EC2

• Cost and Performance Model
  
  Def of Performance:
  num of records for NER jobs
  bytes should have delivered for Apache jobs
  
  Estimate of Cost:
  Assume 1 min to stop instance
  Assume 2 min to start instance
  Coarsed-grained billing, by hours
  Costs that are not considered:
  Cost of controller
  Storage cost of launching new instance
Experiments on EC2

• Experiments with PERF-M

approximately 12 hours with A=10 for NER, 12 hours for Apache

<table>
<thead>
<tr>
<th>Runs</th>
<th>Speedup</th>
<th># Migrations</th>
<th>Total migration cost (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER 1</td>
<td>5.7%</td>
<td>13</td>
<td>376</td>
</tr>
<tr>
<td>NER 2</td>
<td>1.5%</td>
<td>12</td>
<td>402</td>
</tr>
<tr>
<td>NER 3</td>
<td>3.8%</td>
<td>16</td>
<td>472</td>
</tr>
<tr>
<td>Apache 1</td>
<td>3.7%</td>
<td>10</td>
<td>283</td>
</tr>
<tr>
<td>Apache 2</td>
<td>23%</td>
<td>7</td>
<td>296</td>
</tr>
<tr>
<td>Apache 3</td>
<td>35%</td>
<td>16</td>
<td>472</td>
</tr>
</tbody>
</table>
My Comments and Opinions

1. The way it combines up-front and opportunistic in its concrete design is not natural to me. use learning?

2. It seems there is no explanation of the heuristic threshold and no detailed test of how this parameter would affect the results.

3. For gray-box, strategies it is essential that one really understands how CPU, storage and network bandwidth affects your job-specific performance. Otherwise I prefer black box.

4. What if everybody does this in Amazon aggressively (trying to beat the highest performance)? Will the individual optimization lead to collective optimal collective.

5. I would use machine learning to deal with the black box parameter setting rather than some predetermined heuristic criteria.
Questions?

• 1. cloud provider blacklist these ‘aggressive’ players?
• 2. should cloud provider provide additional service to avoid people doing this?
PROFILING, WHAT-IF ANALYSIS, AND COSTBASED OPTIMIZATION OF MAPREDUCE PROGRAMS

Presenter: Ting Xie, tingxie2@illinois.edu
Motivation:

• Before submitting a job to a cloud computing framework, say, Hadoop, a number of choices have to be made in order to fully specify how the job should execute.

• Seems to give users the flexibility, but the performance will be greatly affected if these parameters are not properly set,

• Goal:

  . Tune these parameters automatically before running or after a short period of profiling of your code.
Parameters for Tuning

1. The number of map tasks and workers
2. The number of reduce tasks and workers
3. The amount of memory to allocate to each map (reduce) task to buffer its outputs (inputs).
4. The settings for multi-phase external sorting used by most MapReduce frameworks to group map output values by key.
5. Whether the output data from the map (reduce) tasks should be compressed before being written to disk (and if so, then how).
6. Whether a program-specified Combiner function should be used
Difficulties

1. Black-box map and reduce functions:
   language like Java and C++ is not as restrictive or declarative like SQL, hard to analyze

2. Lack of schema and statistics about the input data:
   User only provides a interface or URL of the input data

3. Differences in plan spaces:
   The execution plan space of configuration parameter settings for MapReduce programs is very different from the plan space for SQL queries.
Important Terms

Job Profile:
1. Dataflow fields
2. Cost fields
3. Dataflow Statistics fields
4. Cost Statistics fields

Cluster Node Homogeneity

Data Flow Proportionality (unless informed by system)
(usually 2 and 4 contains all factors in performance metric)
Components

- Profiler (dynamic instrumentation)
- What if Engine
- CBO (cost-based optimizer)

Given profile, CBO efficiently search through the setting pool to find optimal

(i) subspace enumeration
(ii) search within each enumerated subspace
Generating Profiles or Virtual Profiles

• Via Measurement
  Task-level sampling to generate approximate profiles

• Via Estimation
What-If Engine

- A complex module created by experts that answers questions like:

(detailed example formulas given in the appendix)

<table>
<thead>
<tr>
<th>What-if Questions on MapReduce Job Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WIF_1$</td>
</tr>
<tr>
<td>$WIF_2$</td>
</tr>
<tr>
<td>$WIF_3$</td>
</tr>
</tbody>
</table>

$$mapOutputPairs = \frac{mapInputSize \times mapPairsSelectivity}{mapInputPairWidth}$$
Cost-Based Optimizer (CBO)

Question: trust the profile captured by different system or not?

1. Subspace Enumeration
   - cluster elements in the space first

2. Search optimal within subspace
   - 1) Gridding (discretization)
   - 2) Recursive Random Search (RRS)

   It first samples randomly to identify promising regions that contain the optimal setting with high probability. Then recursively sample a point in this region, as a starting point, to find local optimal.

3. Global optimal from all cluster optimals
Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts to tune MapReduce jobs.
Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts or past experience to tune MapReduce jobs.

<table>
<thead>
<tr>
<th>MapReduce Config. Parameter in Hadoop</th>
<th>Brief Description and Use</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>io.sort.mb</td>
<td>Size (MB) of map-side buffer for storing and sorting key-value pairs produced by the map function</td>
<td>100</td>
</tr>
<tr>
<td>io.sort.record.percent</td>
<td>Fraction of io.sort.mb for storing metadata for every key-value pair stored in the map-side buffer</td>
<td>0.05</td>
</tr>
<tr>
<td>io.sort.spill.percent</td>
<td>Usage threshold of map-side memory buffer to trigger a sort and spill of the stored key-value pairs</td>
<td>0.8</td>
</tr>
<tr>
<td>io.sort.factor</td>
<td>Number of sorted streams to merge at once during multiphase external sorting</td>
<td>10</td>
</tr>
<tr>
<td>mapreduce.combiner.class</td>
<td>The (optional) Combiner function to preaggregate map outputs before transfer to reduce tasks</td>
<td>null</td>
</tr>
<tr>
<td>min.num.spills.for.combine</td>
<td>Minimum number of spill files to trigger the use of Combiner during the merging of map output data</td>
<td>3</td>
</tr>
<tr>
<td>mapred.compress.map.output</td>
<td>Boolean flag to turn on the compression of map output data</td>
<td>false</td>
</tr>
<tr>
<td>mapred.reduce.slowstart.completed.maps</td>
<td>Proportion of map tasks that need to be completed before any reduce tasks are scheduled</td>
<td>0.05</td>
</tr>
<tr>
<td>mapred.reduce.tasks</td>
<td>Number of reduce tasks</td>
<td>1</td>
</tr>
<tr>
<td>mapred.job.shuffle.input.buffer.percent</td>
<td>% of reduce task’s heap memory used to buffer output data copied from map tasks during the shuffle</td>
<td>0.7</td>
</tr>
<tr>
<td>mapred.job.shuffle.merge.percent</td>
<td>Usage threshold of reduce-side memory buffer to trigger reduce-side merging during the shuffle</td>
<td>0.66</td>
</tr>
<tr>
<td>mapred.inmem.merge.threshold</td>
<td>Threshold on the number of copied map outputs to trigger reduce-side merging during the shuffle</td>
<td>1000</td>
</tr>
<tr>
<td>mapred.job.reduce.input.buffer.percent</td>
<td>% of reduce task’s heap memory used to buffer map output data while applying the reduce function</td>
<td>0</td>
</tr>
<tr>
<td>mapred.output.compress</td>
<td>Boolean flag to turn on the compression of the job’s output</td>
<td>false</td>
</tr>
</tbody>
</table>
Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts or past experience to tune MapReduce jobs.

<table>
<thead>
<tr>
<th>Conf. Parameter (described in Table 4)</th>
<th>RBO Settings</th>
<th>CBO Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>io.sort.factor</td>
<td>10</td>
<td>97</td>
</tr>
<tr>
<td>io.sort.mb</td>
<td>200</td>
<td>155</td>
</tr>
<tr>
<td>io.sort.record.percent</td>
<td>0.08</td>
<td>0.06</td>
</tr>
<tr>
<td>io.sort.spill.percent</td>
<td>0.80</td>
<td>0.41</td>
</tr>
<tr>
<td>mapred.compress.map.output</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>mapred.inmem.merge.threshold</td>
<td>1000</td>
<td>528</td>
</tr>
<tr>
<td>mapred.job.reduce.input.buffer.percent</td>
<td>0.00</td>
<td>0.37</td>
</tr>
<tr>
<td>mapred.job.shuffle.input.buffer.percent</td>
<td>0.70</td>
<td>0.48</td>
</tr>
<tr>
<td>mapred.job.shuffle.merge.percent</td>
<td>0.66</td>
<td>0.68</td>
</tr>
<tr>
<td>mapred.output.compress</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>mapred.reduce.tasks</td>
<td>27</td>
<td>60</td>
</tr>
<tr>
<td>min.num.spills.for.combine</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Use of the Combiner</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts to tune MapReduce jobs.

Figure 7: Map and reduce time breakdown for two CO jobs run with configuration settings suggested by RBO and CBO.
Efficiency and Effectiveness of CBO

Figure 11: (a) Running times for all MapReduce jobs running with Hadoop’s Default, RBO-suggested, and CBO-suggested settings; (b) Optimization time, and (c) Number of what-if calls made (unique configuration settings considered) by the six CBOs.
Figure 8: Map and reduce time breakdown for CO jobs from (A) an actual run and (B) as predicted by the What-if Engine.

Figure 9: Actual Vs. Predicted (by What-if Engine) running times for CO jobs run with different configuration settings.
Approximate Profiles through Sampling
Future Work

Challenge:

Some systems submit several jobs together in the form of job workflows which exhibit data dependencies.

In addition, the optimization space now grows to include logical decisions such as selecting the best partitioning function, join operator, and data layout.

Future Direction:

Integrate the What-if Engine with tools like data layout and dynamic resource allocators for complex MapReduce workflows.
My Comments and Opinions

1. Prediction could be done at the cloud computing provider side using learning (assume the system is stable over time, namely the history record is reliable till now). Current optimal setting may be far from global optimal, there will be some other profile-equivalent programs with different setting achieve higher performance.

2. The assumption of data flow proportionality and node homogeniety is controversial.

3. Level of sampling really matters, there is a trade-off between finer level sampling vs. overhead brought by it.
Questions?