Bag-of-Tasks Scheduling under Budget Constraints

Ana-Maria Oprescu, Thilo Kielmann

project co-funded by
the EC 7th Framework Programme

vrije Universiteit  amsterdam
Bags of Tasks

- Example: Parameter sweep applications
- High-throughput computing
- Traditional execution model:
  - find resources (networks of workstations, clusters, grids, ...)
  - sit in a queue
  - run
  - generally no accounting
Clounds

- Elastic computing, get exactly the machines *you* need, exactly when *you* need them...
- ... for the price *they* ask
Assumptions: What’s in a bag?

- Tasks are independent of each other
- Runtimes unknown
- Runtime distribution unknown
- Tasks can be aborted/restarted
- All tasks available for execution when the application starts.
Assumptions: What’s in a cloud?

- Several types of machines
  - different by certain properties, e.g. CPU speed, memory
- Upper limit on the number of machines you can get from a cloud (e.g. self-imposed)
- A machine is charged per Accountable Time Unit (ATU) (e.g. 1 hour)
- We use the term *cluster* for all the machines of the same type you can get from a certain cloud
What’s the problem?

- Goal: Run entire bag on (cloud) clusters, within our budget.
- Bonus goal: Minimize makespan of the whole bag, as much as budget allows.
- Assumptions:
  - some form of runtime distribution exists
  - a ”pay-per-hour” economic model for resource utilization
  - we have all the tasks
1) Start with a set of initial workers from each cluster
2) Run the initial sample on each cluster
3) (Re)configure based on estimates
4) Run tasks
5) At regular monitoring intervals, go back to 3).
Job Profiler: Task runtime estimate

We use estimates to characterize the bag on each machine type

- Statistics for sampling with replacement

For each cluster:

- Keep a moving average
  - initialize the average with a small, initial sample $n$
  - keep collected runtimes of sample set tasks in an ordered list
  - update the moving average during BoT execution
    - Estimate the runtime of running tasks using the average over the "tail" of the sample set.
Reconfigure: How many machines of which types?

- From the average speed of each cluster, (in tasks per minute) we can compute estimates for time/makespan ($T_e$) and budget/cost ($B_e$) for a configuration consisting of nodes from multiple clusters:

  $$T_e = \frac{N}{\sum_{i=1}^{C_{\text{max}}} a_i T_i}$$  
  $$B_e = \left\lceil \frac{T_e}{A \cdot T \cdot U} \right\rceil \cdot \sum_{i=1}^{C_{\text{max}}} a_i \cdot c_i$$

- We minimize $T_e$ while keeping $B_e \leq B$ using a modified Bounded Knapsack Problem (BKP) method.

  - The BKP can be solved in pseudo-polynomial time, as 0-1 knapsack problem via linear programming.

- BaTS chooses the configuration with minimal $T_e$ for $B_e \leq B$. 
Cluster monitoring and BoT execution progress

BaTS regularly re-evaluates the current cluster configuration:

- At each monitoring interval, the problem gets smaller (less tasks left, less budget left).
- Each moving average converges during the run
- Execution on real machines adds some complexity:
  - Individually requested from the cloud provider, startup time until ready
  - Each machine has a different time left of the current ATU
  - Runtime granularity ⇒ paid machine time possibly unused
- Throughout bag execution, BaTS keeps track of
  - Time on machines we already paid for
  - Actual speed (tasks/minute) achieved per cluster
Evaluation Setup - workloads and clouds

- Synthetic workloads
  - N=1000 tasks ⇒ n=30 (sample set size)
  - Normal distribution of runtime: avg=15 min, st. dev.=2.23
  - Iosup et al. show bags typically have some normal distribution
    [The performance of bags-of-tasks in large-scale distributed systems]

- Tasks sleep defined "run" time

- Cloud emulation on DAS-3
  - 2 clouds, 32 machines each
  - Fast/slow machines emulated by modifying the sleep time
  - Allocate through local site scheduler (without competing users)
  - Accountable Time Unit = 1 hour

- Compare BaTS to a self-scheduler (RR)
Evaluation Setup
Profitability: how much faster vs. how much costier

▸ We propose 5 different scenarios: speed and cost of cluster\(_2\) compared to normalized speed and cost of cluster\(_1\).

<table>
<thead>
<tr>
<th>cluster(_2) speed</th>
<th>cluster(_1) speed</th>
<th>profitabilty (c_2) w.r.t. (c_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1.33</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

▸ We evaluate each scenario by running:
  ▸ self-scheduler (RR) always using 32+32 machines
  ▸ BaTS on initial config. 30+30 machines provided with
    ▸ budget \(B_{\text{BaTS}_{RR}}\) = cost incurred by running RR (\(C_{RR}\))
    ▸ budget \(B_{\text{BaTS}_{B_{\text{Min}}}}\), computed off-line as the cost incurred by running the bag on a machine of the most profitable type.
Results - Makespan (M), Cost (C) and Budget (B)
Results - Makespan (M), Cost (C) and Budget (B)
Results - Makespan (M), Cost (C) and Budget (B)
Conclusions

- Choosing the cloud resources suitable for your application is tough
- BaTS can help staying within budget while still performing reasonably well
- Limitation: Guessing a proper budget up front
  - Current work: fixing limitation by pre sampling (even smaller)
  - Early results promising
- Future work
  - DAG’s instead of BoT’s (dependencies)
  - BaTS for MapReduce?
Contrail

contrail is co-funded by the EC 7th Framework Programme

Funded under: FP7 (Seventh Framework Programme)
Area: Internet of Services, Software & virtualization (ICT-2009.1.2)
Project reference: 257438
Total cost: 11,29 million euro
EU contribution: 8,3 million euro
Execution: From 2010-10-01 till 2013-09-30
Duration: 36 months
Contract type: Collaborative project (generic)
Related work - Assumptions we *don’t* make

- prior knowledge on task arrival rate, execution time, deadline.
- same complexity class and a calibration step to estimate execution time per machine type.
- prior knowledge of relative complexity classes of tasks
- fixed, one-time cost per machine type.
[amo@fs0 ~]$ preserve -llist
Thu Oct 21 02:40:06 2010

<table>
<thead>
<tr>
<th>id</th>
<th>user</th>
<th>start</th>
<th>stop</th>
<th>state</th>
<th>nhosts</th>
<th>hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1152334</td>
<td>vpopescu</td>
<td>10/15</td>
<td>12/24</td>
<td>00:25</td>
<td>r</td>
<td>node010</td>
</tr>
<tr>
<td>1152611</td>
<td>ppouwels</td>
<td>10/20</td>
<td>10/21</td>
<td>08:00</td>
<td>r</td>
<td>node030</td>
</tr>
<tr>
<td>1152607</td>
<td>ppouwels</td>
<td>10/20</td>
<td>10/21</td>
<td>08:00</td>
<td>r</td>
<td>node059</td>
</tr>
<tr>
<td>1152608</td>
<td>ppouwels</td>
<td>10/20</td>
<td>10/21</td>
<td>08:00</td>
<td>r</td>
<td>node060</td>
</tr>
<tr>
<td>1152633</td>
<td>ppouwels</td>
<td>10/21</td>
<td>10/21</td>
<td>12:22</td>
<td>r</td>
<td>node062</td>
</tr>
<tr>
<td>1152606</td>
<td>ppouwels</td>
<td>10/20</td>
<td>10/21</td>
<td>08:00</td>
<td>r</td>
<td>node068</td>
</tr>
<tr>
<td>1152634</td>
<td>ppouwels</td>
<td>10/21</td>
<td>10/21</td>
<td>13:01</td>
<td>r</td>
<td>node078</td>
</tr>
<tr>
<td>1152604</td>
<td>mcd</td>
<td>10/20</td>
<td>10/21</td>
<td>23:02</td>
<td>r</td>
<td>node076</td>
</tr>
</tbody>
</table>

[amo@fs0 ~]$ finger ppouwels
Login: ppouwels
Directory: /home5/ppouwels
Office: VUMC, PJW.Pouwels@vumc.nl
Never logged in.
No mail.
No Plan.

[amo@fs0 ~]$ finger vpopescu
Login: vpopescu
Directory: /home5/vpopescu
Office: VUMC, v.popescu@vumc.nl
Never logged in.
No mail.
No Plan.