Bag-of-Tasks Scheduling under Budget Constraints

Elie Mokheiber

March 12, 2012
Problem

- Rise in commercial cloud offerings
  - Allocate compute resources on demand
  - Charged based on reserved time intervals

- Users lack guidance
  - Several types of machines offered
  - Unclear which machine type is faster for a certain applications

- BaTS (Bags-of-Tasks Scheduler)
Outline

- What is BaTS
- Assumptions
- Architecture
- Profiling task execution time
- Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
Outline

▶ What is BaTS
▶ Assumptions
▶ Architecture
▶ Profiling task execution time
▶ Monitoring plan execution
▶ Algorithm
▶ Test Setup
▶ Test Results
▶ Conclusion
Outline

- What is BaTS
- Assumptions
- Architecture
  - Profiling task execution time
  - Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
Outline

- What is BaTS
- Assumptions
- Architecture
- Profiling task execution time
- Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
Outline

- What is BaTS
- Assumptions
- Architecture
- Profiling task execution time
- Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
Outline

▶ What is BaTS
▶ Assumptions
▶ Architecture
▶ Profiling task execution time
▶ Monitoring plan execution
▶ Algorithm
▶ Test Setup
▶ Test Results
▶ Conclusion
Outline

- What is BaTS
- Assumptions
- Architecture
- Profiling task execution time
- Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
Outline

- What is BaTS
- Assumptions
- Architecture
- Profiling task execution time
- Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
Outline

- What is BaTS
- Assumptions
- Architecture
- Profiling task execution time
- Monitoring plan execution
- Algorithm
- Test Setup
- Test Results
- Conclusion
What is BaTS?

- Budget-constrained scheduler.
- Schedules large bags of tasks onto multiple clouds with different CPU performance and cost.
- Stays within Budget
- Requires no a-priori information about task completion times
- Java-based using Ibis
- Master - Worker principle
- Uses IPL as communication layer
Assumptions: Bag-of-Tasks

- Tasks are independent of each other
- No prior information about run-times
- Tasks can be pre-empted and rescheduled
- All tasks are available when application starts (N is known)
Assumptions: The cloud

- Multiple types of machines
  - Difference in CPU speed, memory, etc.
  - e.g. EC2s Standard Large or High-Memory Double Extra Large
  - Homogeneous

- Maximum number of machines available from cloud

- A machine is charged per Accountable Time Unit (ATU)
  - e.g. 60 minutes

- We define *cluster* as all machines that are from the same type in the cloud
BaTS: System Architecture

Diagram showing the system architecture including:
- Master
- Job Profiler
- Cluster Utilization
- Reconfigure
- Scheduler
- Clusters A, B, and C
Profiling task execution time (1/2)

Use estimates to profile the tasks on each machine type

- Statistical formula for sampling with replacement

For each cluster

- Cumulative moving average
  - Initialize the average with an initial sample set n
  
- Keep collected run-times in an ordered list

- Update the moving average during execution
Profiling task execution time (2/2)

- We use the average estimated speeds of the machine types participating in the execution to compute estimates of the Makespan ($T_e$) and Budget($B_e$) needed for the bag-of-tasks execution:

\[
T_e = \frac{N}{\sum_{i=1}^{C_{nc}} a_i / T_i} ; \quad B_e = \left[ \frac{T_e}{ATU} \right] \times \sum_{i=1}^{C_{nc}} a_i \times c_i ,
\]

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

- BaTS chooses the combination of machines with minimal make-span, where the budget is within the given limit.
Monitoring plan execution

At regular monitoring intervals, BaTS revisits the plan to accommodate the actual progress of the bag of tasks

- Monitoring interval is fraction of the ATU, but ≥ 5 minutes.
- The problem gets smaller (less tasks left, less budget left).
- The moving average gets refined with each completed task
- Check if remaining ATU covers task left
- During execution, BaTS keeps track of
  - Time left on each machine
  - Actual speed (tasks/minute) achieved per cluster
BaTS Algorithm

1: compute \( n = \text{sample size} \)
2: construct initial configuration \( C \)
3: acquire machines according to \( C \)
4: while bag has tasks do
5: \hspace{1em} wait for any machine \( M \) to ask for work
6: \hspace{2em} if \( M \) returned result of task \( T \) then
7: \hspace{3em} update statistics for machine \( M \)
8: \hspace{3em} update the \( rt_{done} \) for \( M \)'s type \( Mt \)
9: \hspace{2em} end if
10: \hspace{1em} if sample set tasks for \( Mt \) finished then
11: \hspace{2em} update cluster stats for \( Mt \)
12: \hspace{1em} end if
13: \hspace{1em} if (monitoring time) \| (first clusters stats) then
14: \hspace{2em} compute estimates
15: \hspace{3em} if constraint violation then
16: \hspace{4em} call BKP to compute a new configuration \( C' \)
17: \hspace{4em} acquire the extra machines required by \( C' \)
18: \hspace{4em} save \( C' \) in \( C \)
19: \hspace{3em} end if
20: \hspace{2em} end if
21: \hspace{1em} if number of machines of \( Mt \) satisfies \( C \) then
22: \hspace{2em} send \( M \) a randomly selected task \( T' \)
23: \hspace{2em} remove \( T' \) from bag and place it in pending
24: \hspace{2em} end if
25: \hspace{1em} if number of machines of \( Mt \) should decrease then
26: \hspace{2em} release \( M \)
27: \hspace{2em} end if
28: end while
Cloud emulation on DAS-3
- 2 clusters, 32 machines in each cluster
- ATU = 1 hour
- Difference in machines is emulated through the `sleep` function

Workload
- 1000 Tasks in total, 30 tasks in sample set
- Run = `sleep`
- Average runtime is 15 Min

BaTS is compared to Round-Robin
Test Setup (2/3)

- 2 clusters: \( \text{cluster}_0 \) and \( \text{cluster}_1 \)
- \( \text{Cluster}_0 \) charges $3 per machine per ATU and average 15 Min runtime
- \( \text{Cluster}_1 \) has 5 scenarios

<table>
<thead>
<tr>
<th>Profitability</th>
<th>Speed</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>0.75</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.33</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Each scenario is evaluated by running:

- Round-robin self scheduler using all 64 machines (32 + 32)
- BaTS on 30+30 machines (initial configuration), with
  - Budget $B_{BaTS_{RR}}$, the cost used by running RR($C_{RR}$)
  - Budget $B_{BaTS_{Min}}$, the cost of running the task on the most profitable machine type
Test Results (1/2)

Elie Mokheiber

Bag-of-Tasks Scheduling under Budget Constraints
Test Results (2/2)

- BaTS successfully schedules bags of tasks within the given user-defined budget constraints
- Three ways to improve BaTS
  - Reduce the overhead incurred by sampling on each cluster
  - A different treatment of the final phase of the computation, both in the reconfiguration and the scheduler modules
  - An additional condition to trigger the search for new configurations further minimizing the make-span and/or saving money
Choosing the best suitable configuration for your application is hard

BaTS stays within the predefined budget ....

although slower

Future Work

Decoupling the initial sampling phase from the main execution

Improving the tail phase of the schedule

**BaTS can help users stay within their budget while still performing reasonably well**