Shed: Optimal Dynamic Cloning to Meet Application Deadlines in Cloud

Sultan Alamro, Maotong Xu, Tian Lan, and Suresh Subramaniam
The George Washington University
Department of Electrical and Computer Engineering

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Introduction

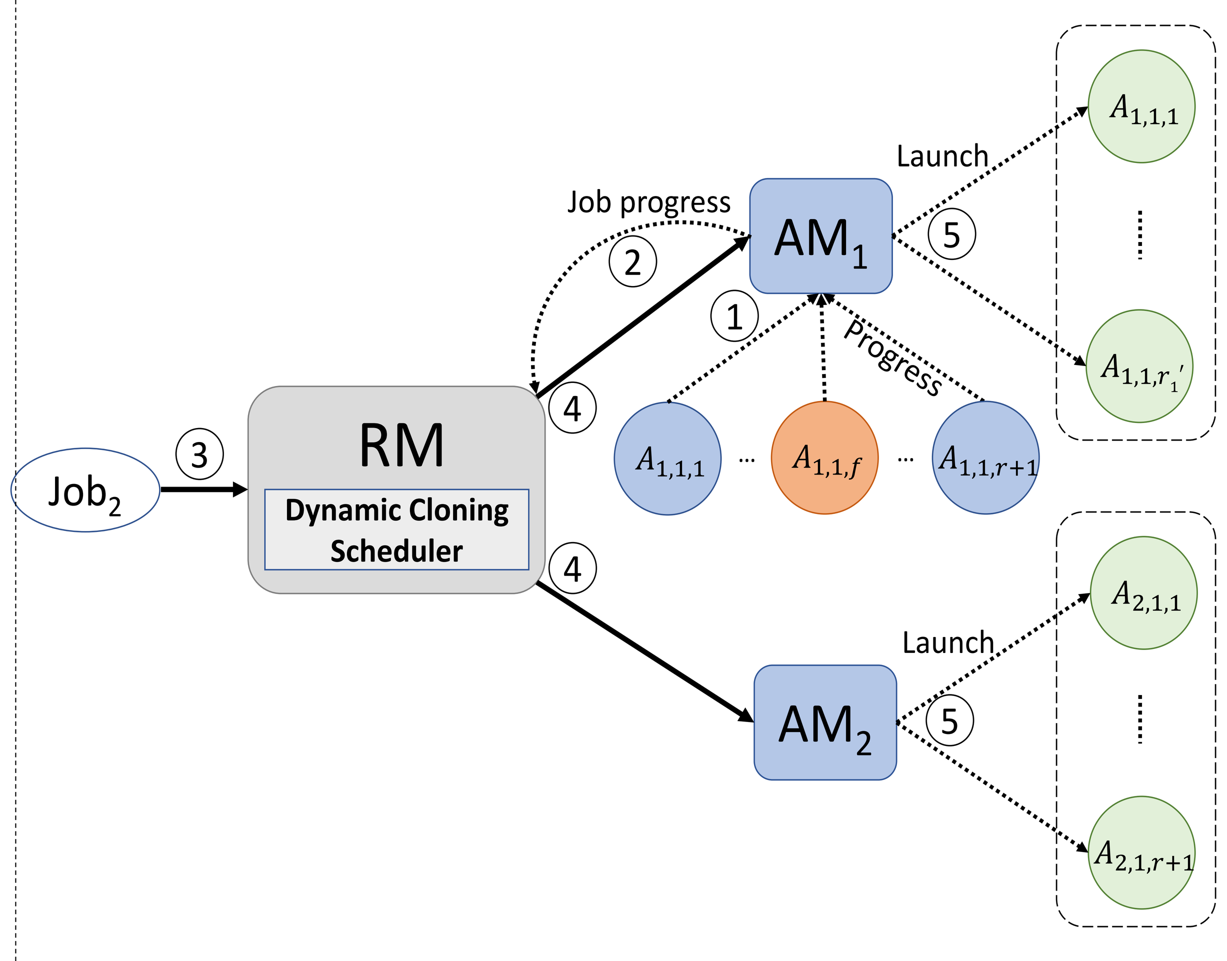
- Demand for cloud-based processing frameworks continues to grow
- Cloud applications is becoming increasingly mission-critical and deadline sensitive, especially in shared clusters
- Cloud providers seek efficient techniques to meet the SLA
- A few slow tasks, called stragglers, could significantly impact job execution time
- Launching extra attempts (clones) for each task upon submission can mitigate stragglers
- This work proposes Shed, an optimization framework that leverages dynamic cloning to jointly maximize jobs' Probability of Completion before Deadline (PoCD) by fully utilizing the available resources

System Model

Consider J jobs submitted to the Mapreduce processing framework. Each job j is associated with a deadline Dj that is determined by a user. Each job j consists of Nj tasks, and it is considered successful if all its Nj tasks are executed and completed before the job deadline Dj. Let Tj denote job j's completion time, and Tj,i for i = 1,…,Nj be the (random) completion times of tasks belongs to job j. Any task whose execution time exceeds the deadline is considered a straggler. Our dynamic cloning approach mitigate the effect of stragglers by proactively launching rj extra attempts for each task. A task is finished once any one of the rj + 1 attempts finishes execution, and then the other copies are killed. Thus, taks i's completion time Tj,i is determined by the completion time of the fastes attempt, i.e.,

$$T_{j,i} = \min_{k=1,\ldots,r_j+1} T_{j,i,k}, \ \forall i, j.$$

System Architecture



Joint PoCD Optimization

$$\text{maximize} \quad \sum_{j=1}^{J} U(p_j),$$

$$\text{s.t.} \quad \sum_{j=1}^{J} N_j \cdot (r_j + 1) + |J| \le \lambda \cdot m$$

$$p_j = R_j(r_j), \ \forall j$$

$$r_j \ge 0, \ \forall j$$

Proposed Online Algorithm

Algorithm 1: Proposed Online Algorithm

1: Upon submission of a new job:
2: Kill all jobs which missed their deadlines
3: $J = \{j_1, j_2, j_3, \ldots\}$
4: **if** $|J| == 1$ **then**
5: $\quad r_{\max} = \left\lfloor \frac{\lambda \cdot m - N_1 - 1}{N_1} \right\rfloor$
6: $\quad r_1 = r_{\max}$
7: **else**
8: $\quad r_j = 0 \ \forall j$
9: $\quad \omega = 0$
10: $\quad \kappa = \lambda \cdot m - \sum_{j=1}^{J} N_j - |J|$
11: $\quad$ Calculate $R_j \ \forall j$
12: $\quad$ **while** $J \ne \{\emptyset\}$ **do**
13: $\quad\quad j' = \arg\min_j \{R_j\}$
14: $\quad\quad$ **if** $N_{j'} + \omega > \kappa$ **then**
15: $\quad\quad\quad J = J - \{j'\}$
16: $\quad\quad$ **else**
17: $\quad\quad\quad r_{j'} = r_{j'} + 1$
18: $\quad\quad\quad \omega = \omega + N_{j'}$
19: $\quad\quad\quad$ Calculate $R_j \ \forall j$
20: $\quad\quad$ **end if**
21: $\quad$ **end while**
22: **end if**

Theorems

$$R_{\text{su}} = \left[1 - \left(\frac{t_{\min}}{D_j} \right)^{\beta \cdot (r_j + 1)} \right]^{N_j}$$

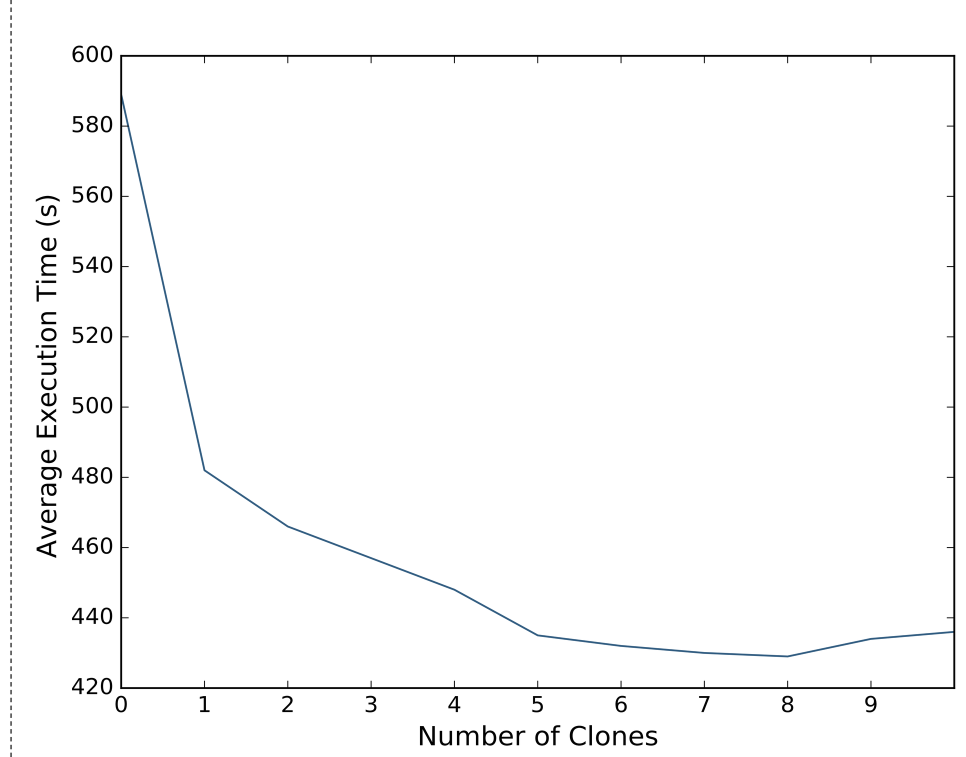
$$R_{\text{ru}} = \left[1 - \left(\frac{(1 - \phi_i) t_{\min}}{D_j - \tau_j} \right)^{\beta \cdot (r_j + 1)} \right]^{N_j}$$

Conclusion

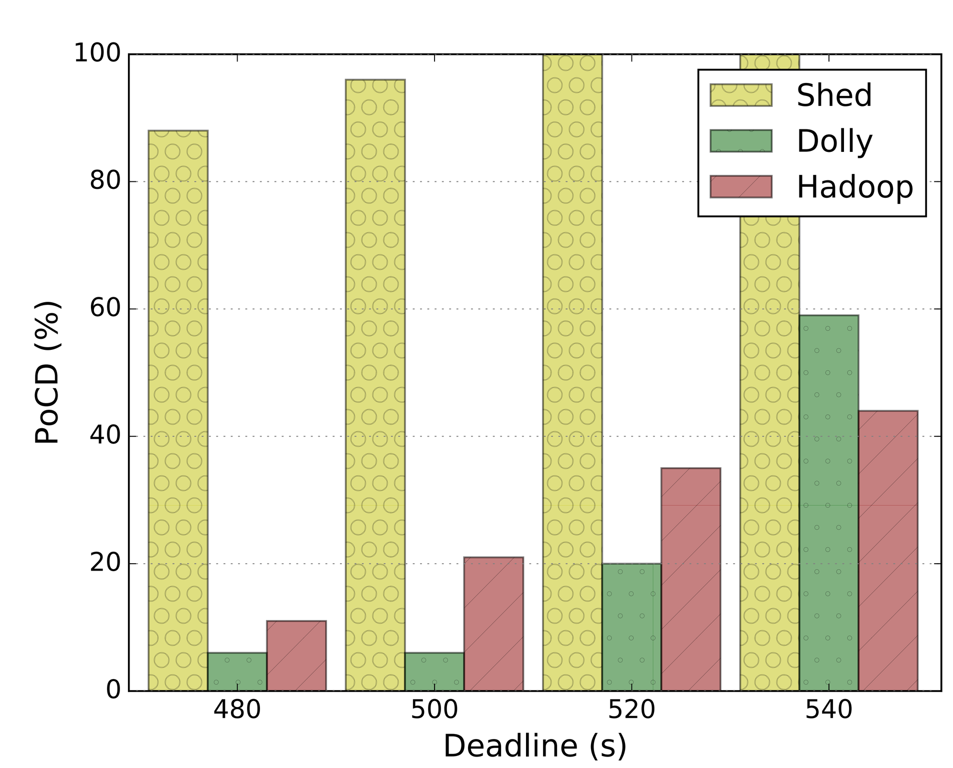
In this work, we propose Shed, an optimization framework that leverages dynamic cloning to jointly maximize PoCD and cluster utilization. We also present an online scheduler that dynamically optimize resources upon new job arrival. Our solution includes an online greedy algorithm to find the optimal number of clones needed for each job. Our results show, in some cases, that Shed can achieve 100% PoCD compared to Dolly and Hadoop with speculation. The proposed algorithm is able to achieve more than 90% utilization of available cloud resources, whereas Dolly and Hadoop achieves only about 22%.

Reference
- J. Dean *et al*, *2008*
- G. Ananthanarayanan *et al*, 2013

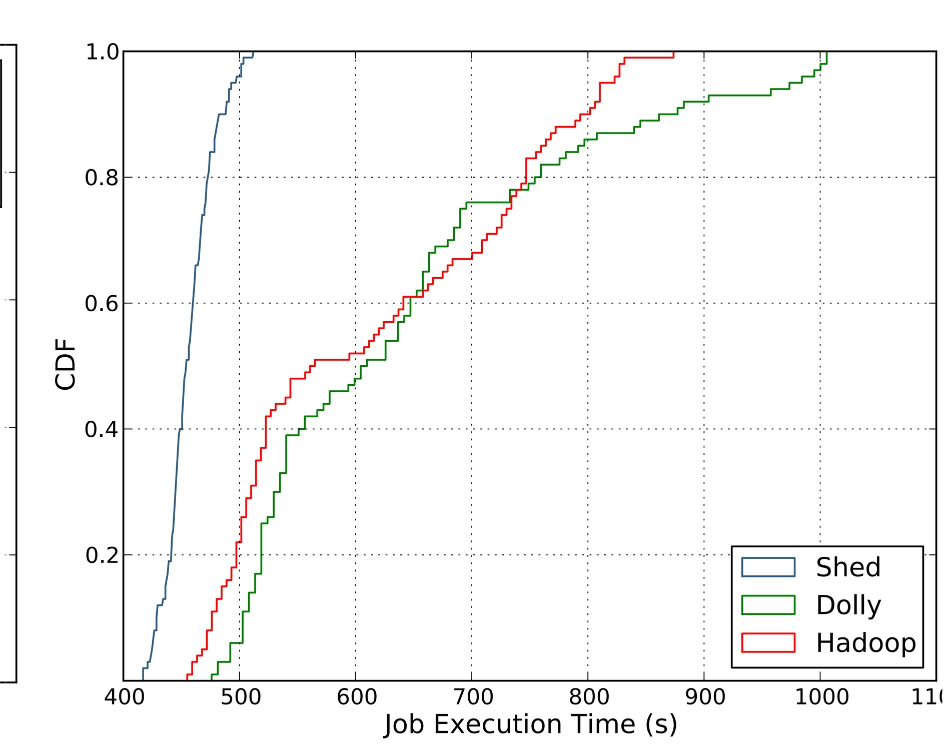
Evaluation



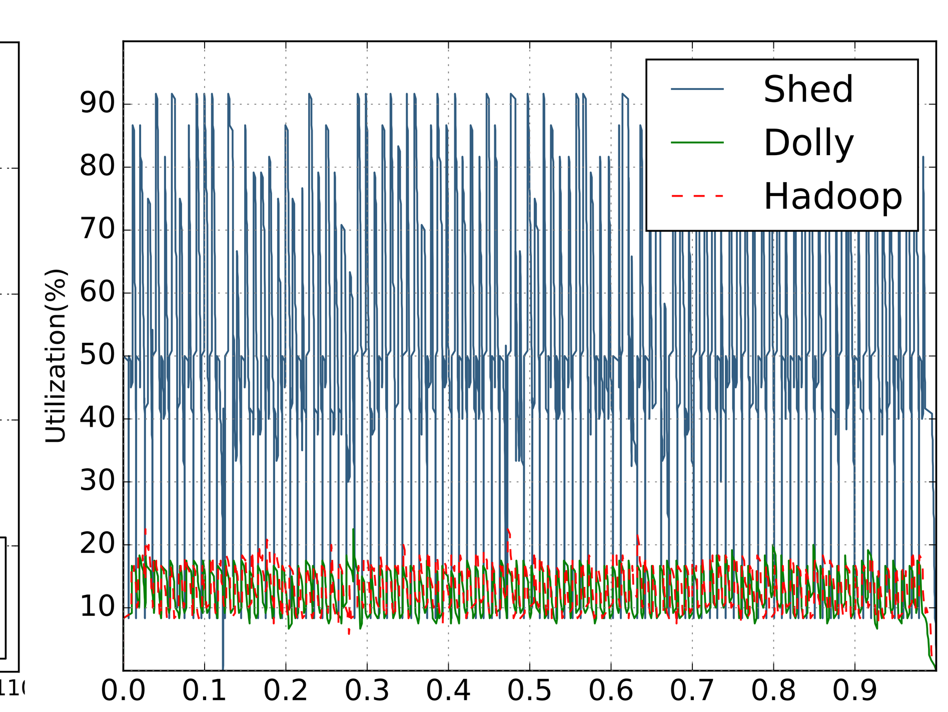
Average job execution time versus number of clones



PoCD with of 10-tasks jobs using WordCount benchmark



CDF for 10-tasks jobs of WordCount benchmark



Cluster utilization for 1—tasks jobs of WordCount benchmark