Sequential addition of coded tasks for straggler mitigation

Ajay Badita [1], Parimal Parag [1], Vaneet Aggarwal[2]

[1] Indian Institute of Science, Bengaluru, KA, India

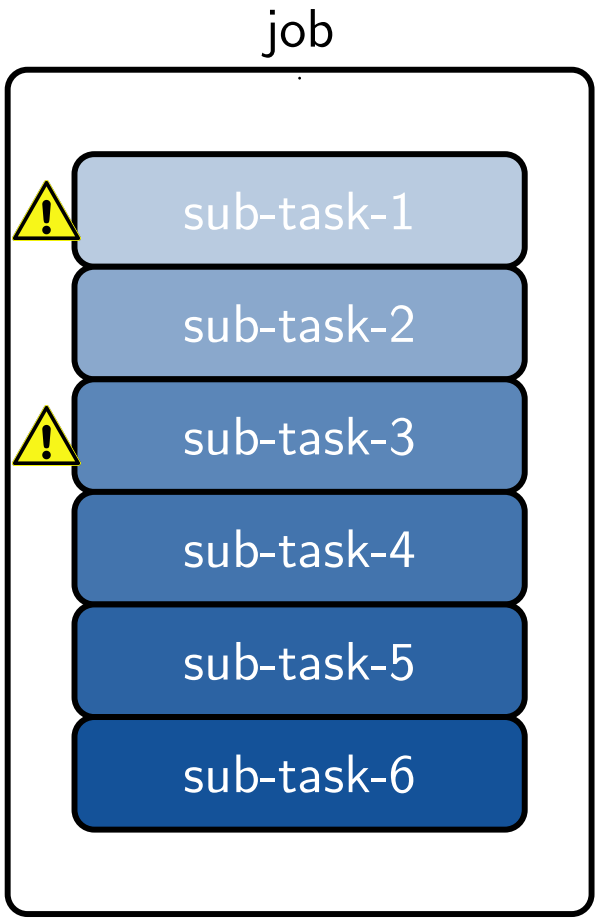
[2] Purdue University, West Lafayette, IN, USA

Large-scale distributed systems

Distributed Computation

Distributed Storage

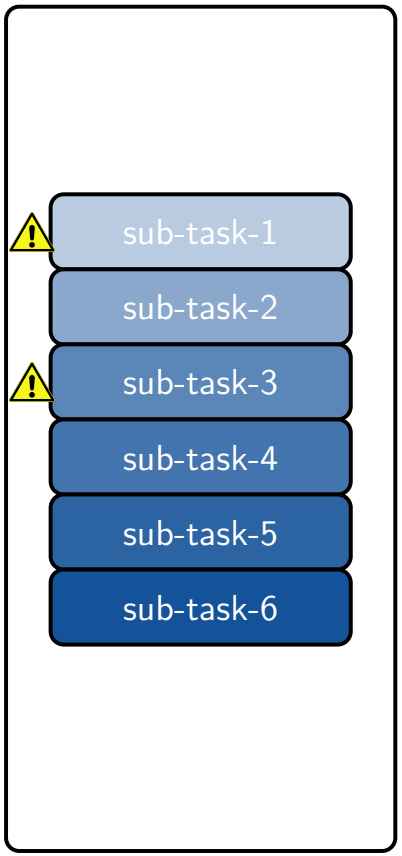
Stragglers in parallel computation

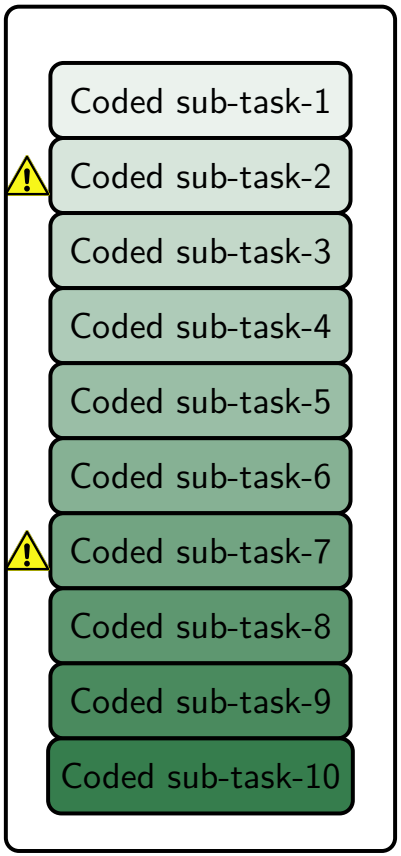


Parallel sub-tasks

Completion: last of sub-task completions at all six servers

Coded redundancy solution





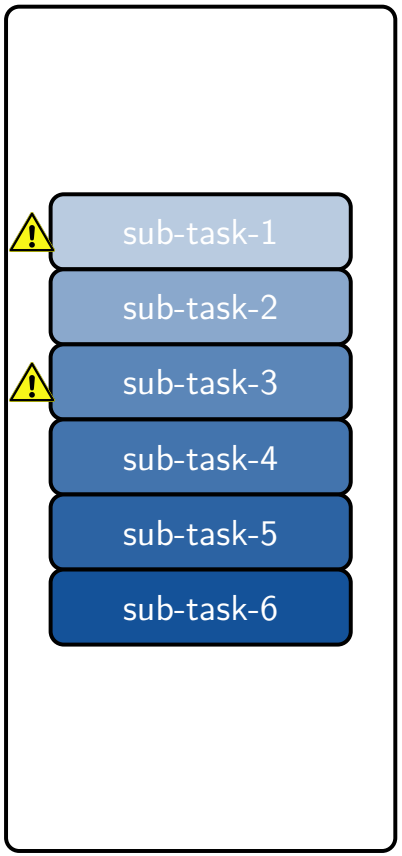
Uncoded sub-tasks

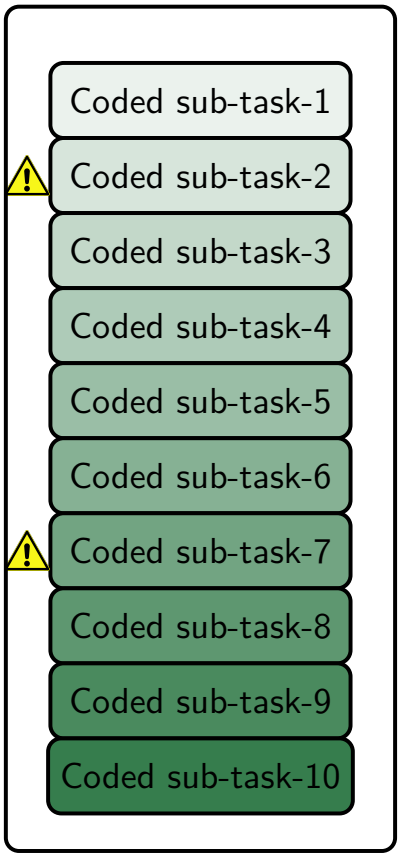
Completion: largest of all six servers

Coded sub-tasks

Completion: sixth largest of all ten

Coded redundancy solution





Uncoded sub-tasks

Completion: largest of all six servers

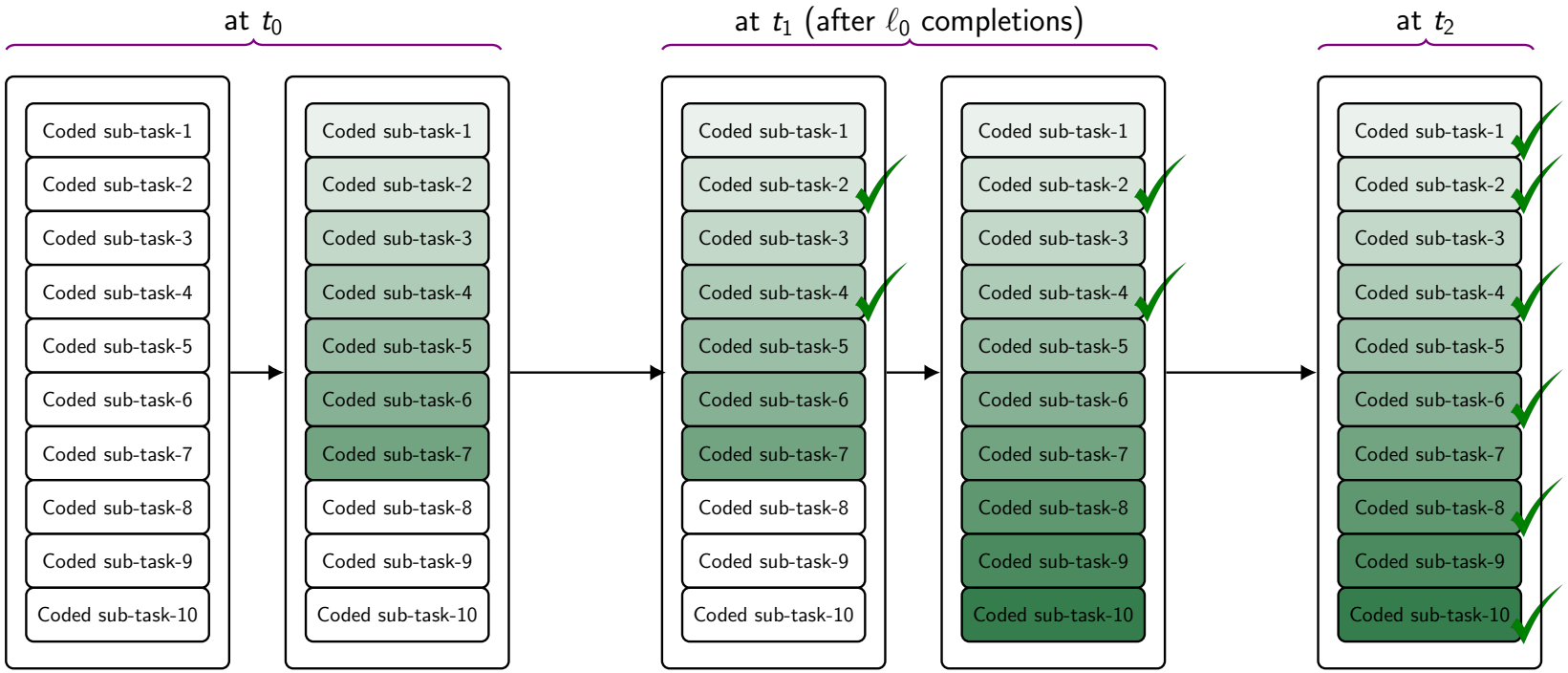
Maximum six servers in use

Coded sub-tasks

Completion: sixth largest of all ten

Maximum ten servers in use

Delayed relaunch



Problem

Find the number of initial servers and fork-task threshold that maximally trade-off between means of completion time and utilization cost.

Prior Work

Wang, TOMPECS2019

- ⋆ Initial servers: number of sub-tasks k
- ⋆ Fork-task threshold: a fraction of initial servers
- ⋆ Launch r new replicas for each remaining task

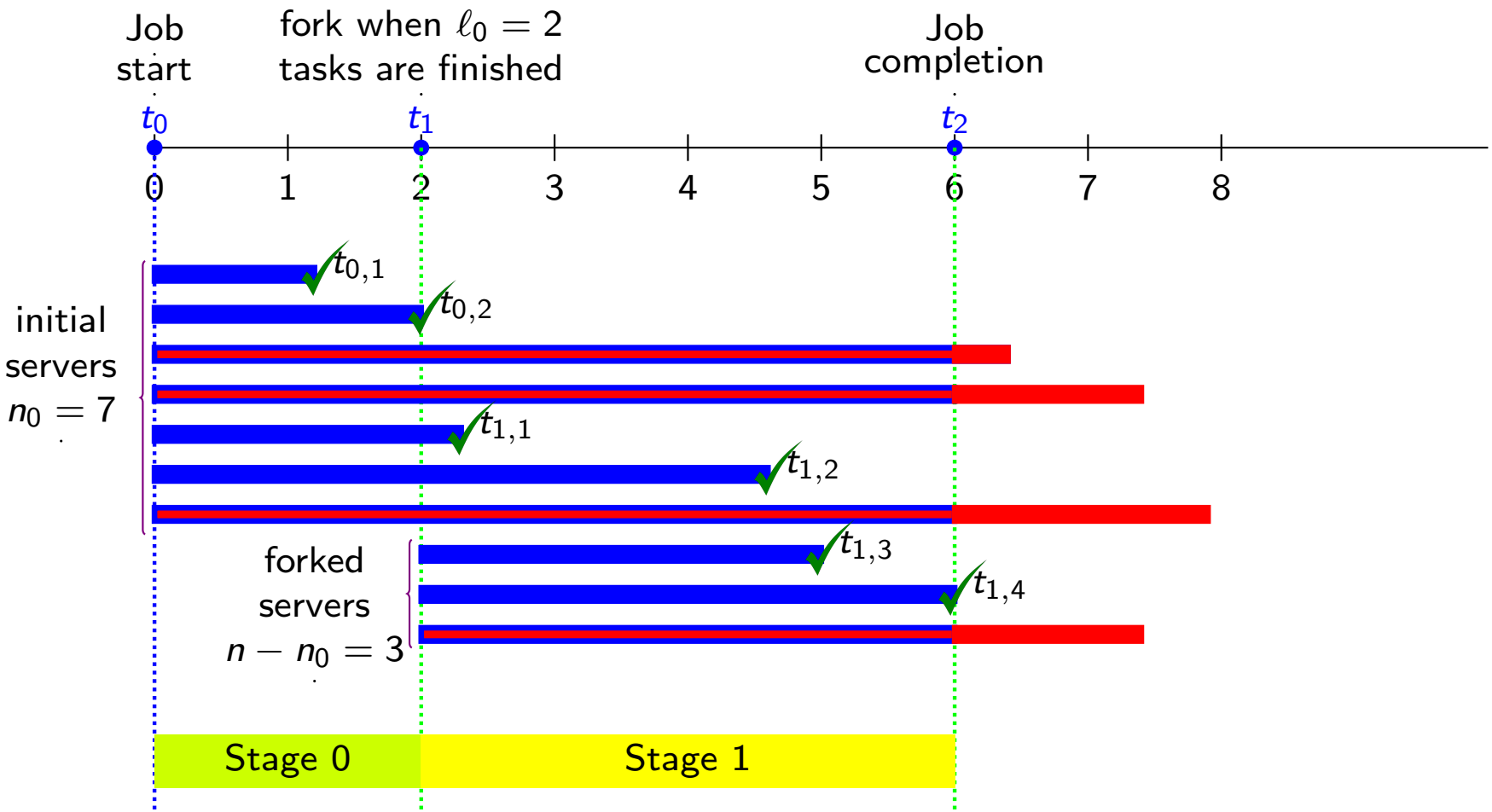
Aktas, IEEE/ACM TNET2019

- ⋆ Initial servers: number of sub-tasks k
- ⋆ Fork-time threshold: Fixed duration
- ⋆ Launch remaining coded sub-tasks

This work: k subtasks coded to n MDS coded subtasks

- ⋆ Initial servers: any number of available servers $n_0 \leqslant n$
- ⋆ Fork-task threshold: $\ell_0 \leqslant \min\{n_0, k\}$
- ⋆ Launch remaining coded sub-tasks at ℓ_0 completions
- ⋆ The choice of initial number of server matters

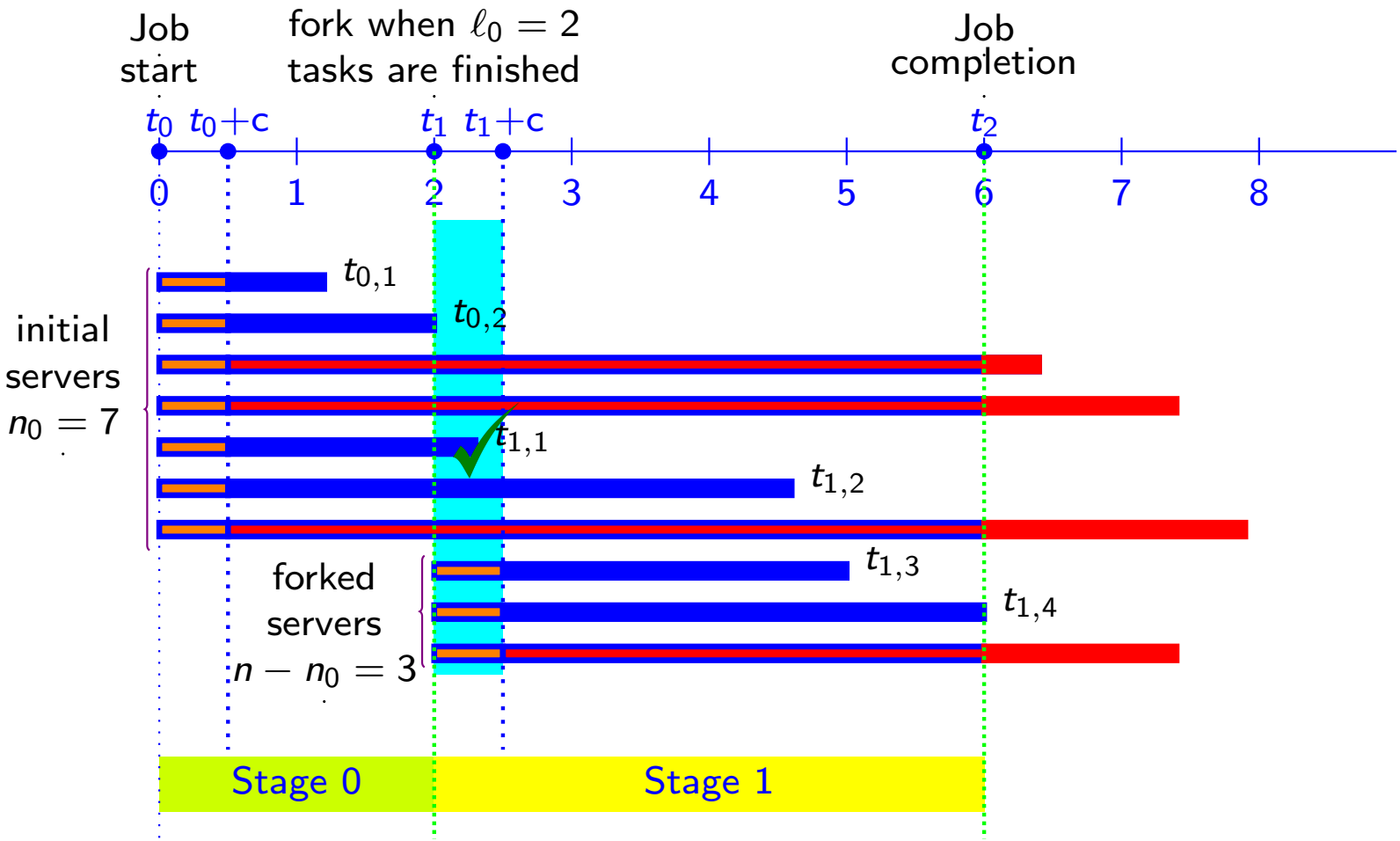
Performance metric computation



★ Completion time $t_{1,4}$: $t_{0,1} + (t_{0,2} - t_{0,1}) + (t_{1,1} - t_{0,2}) + (t_{1,2} - t_{1,1}) + (t_{1,3} - t_{1,2}) + (t_{1,4} - t_{1,3})$

★ Cost: $7(t_{0,1}) + 6(t_{0,2} - t_{0,1}) + 8(t_{1,1} - t_{0,2}) + 7(t_{1,2} - t_{1,1}) + 6(t_{1,3} - t_{1,2}) + 5(t_{1,4} - t_{1,3})$

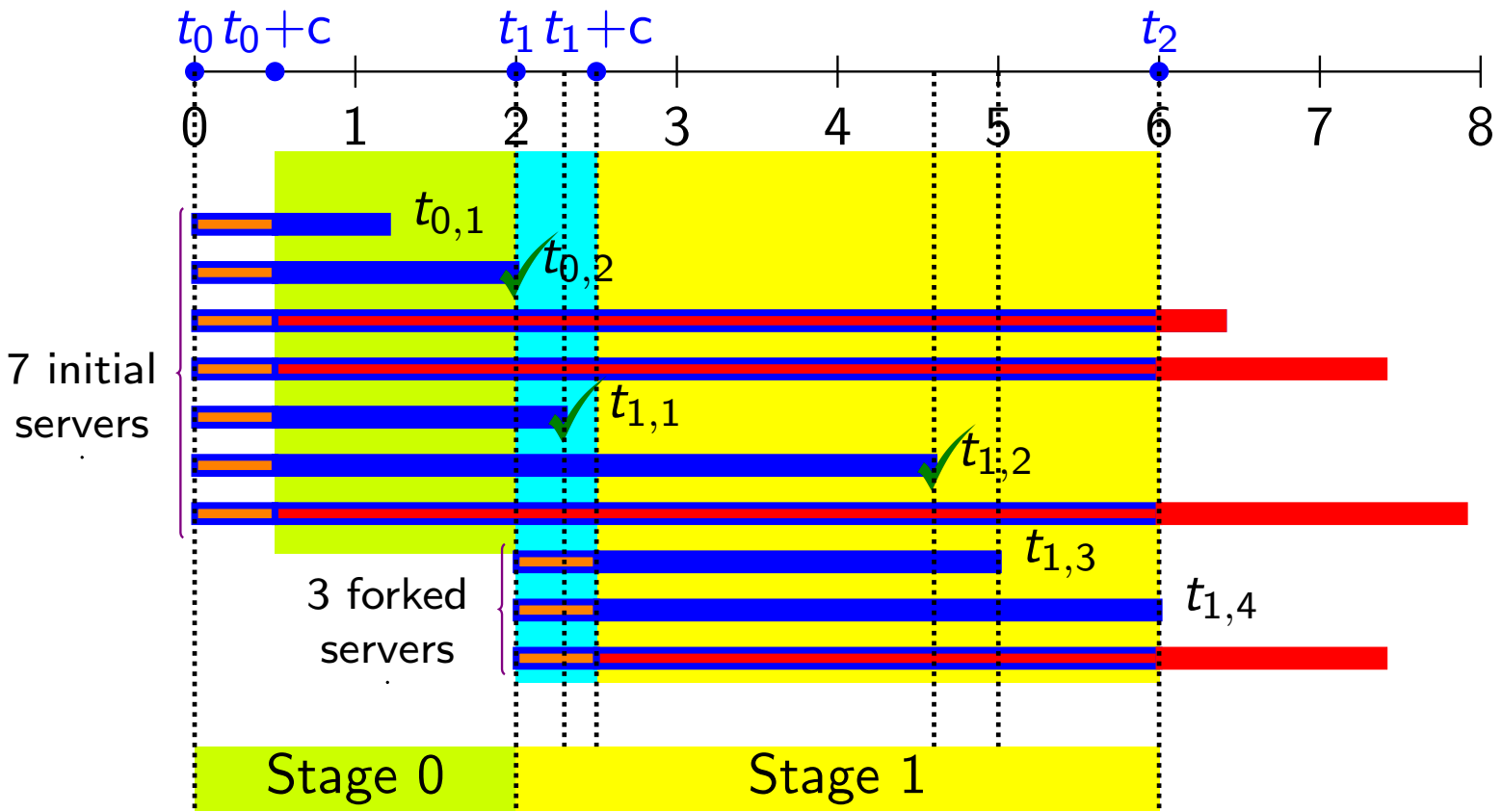
Shifted exponential service distribution



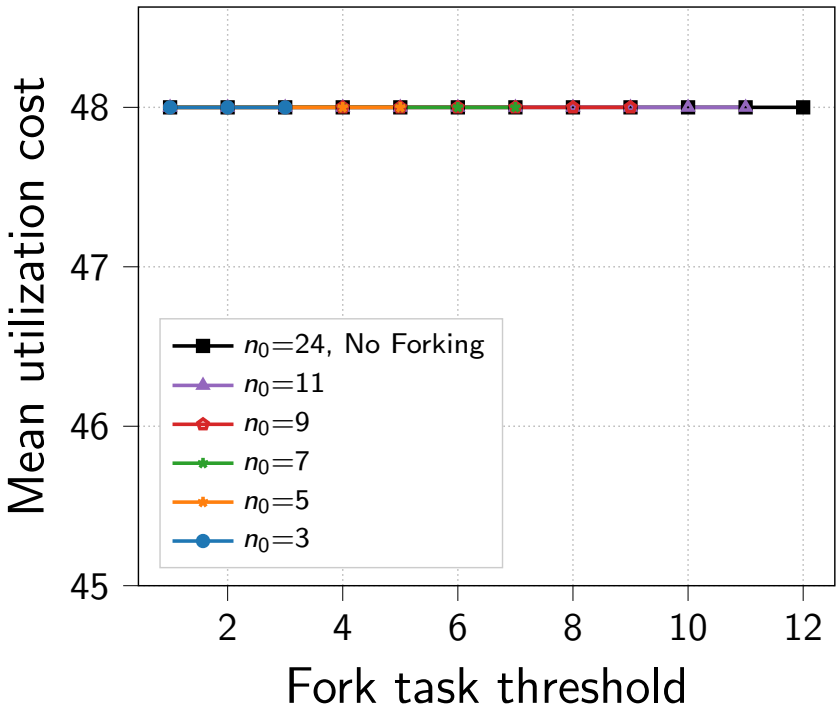
Beginning of stage 1

Mean time between two coded sub-task completions in stage 1 conditioned on the event of m completions in $[t_1, t_1 + c)$ denoted by E_m

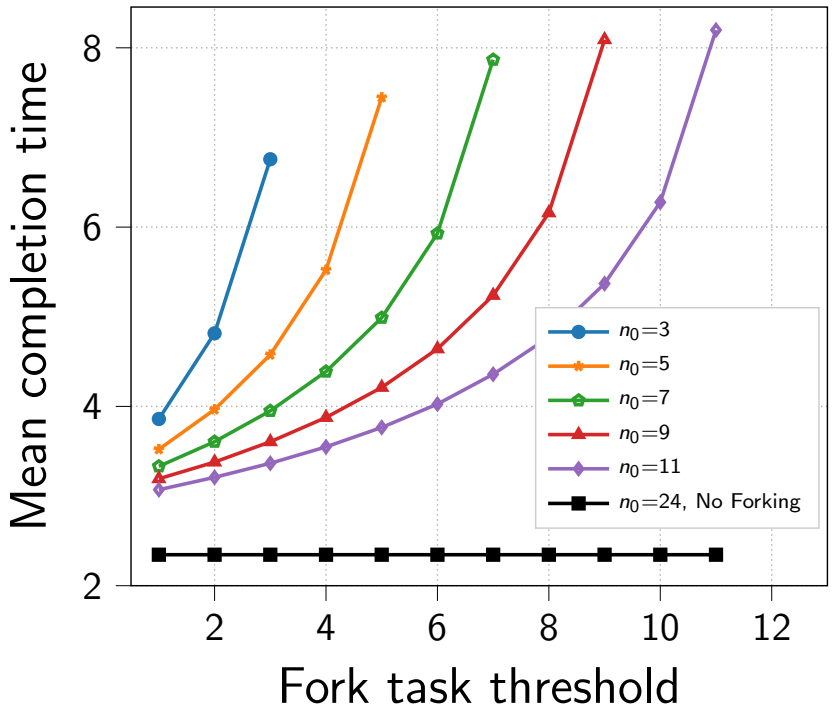
$$\mathbb{E}[(t_{1,r} - t_{1,r-1})|E_m] = \begin{cases} {}_2F_1\left(\begin{smallmatrix}1,r\\ m+2\end{smallmatrix}; \alpha\right) \frac{r\alpha}{\mu(m+1)}, & r < m+1, \\ c\left[\frac{1}{\alpha^{(r-1)}} - \sum_{i=1}^{r-1} \frac{\alpha^{i-r+1}}{ic\mu}\right] + \frac{1}{\mu(n-m-\ell_0)}, & r = m+1. \end{cases}$$



Initial servers n_0 smaller than sub-tasks k



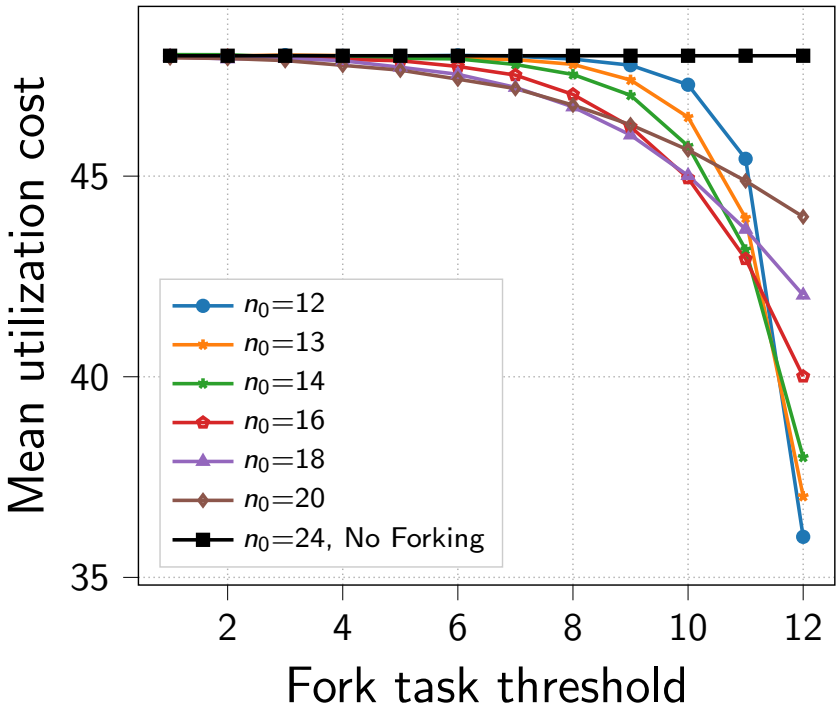
Constant for any choice



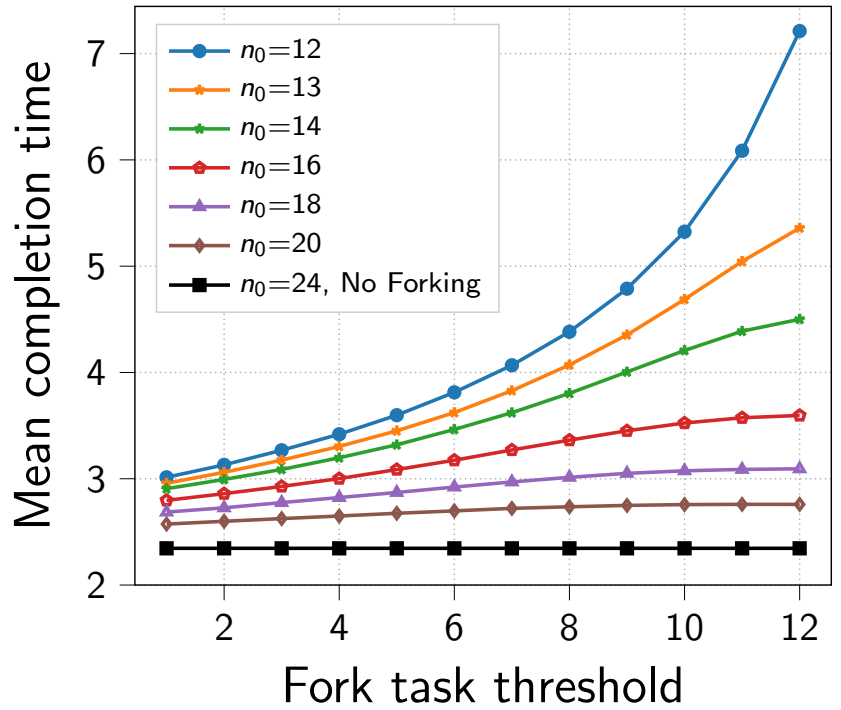
Increasing in fork-task threshold

⋆ Mean utilization cost is identical to that of no-forking case **!**

Initial servers n_0 greater than or equal to sub-tasks k

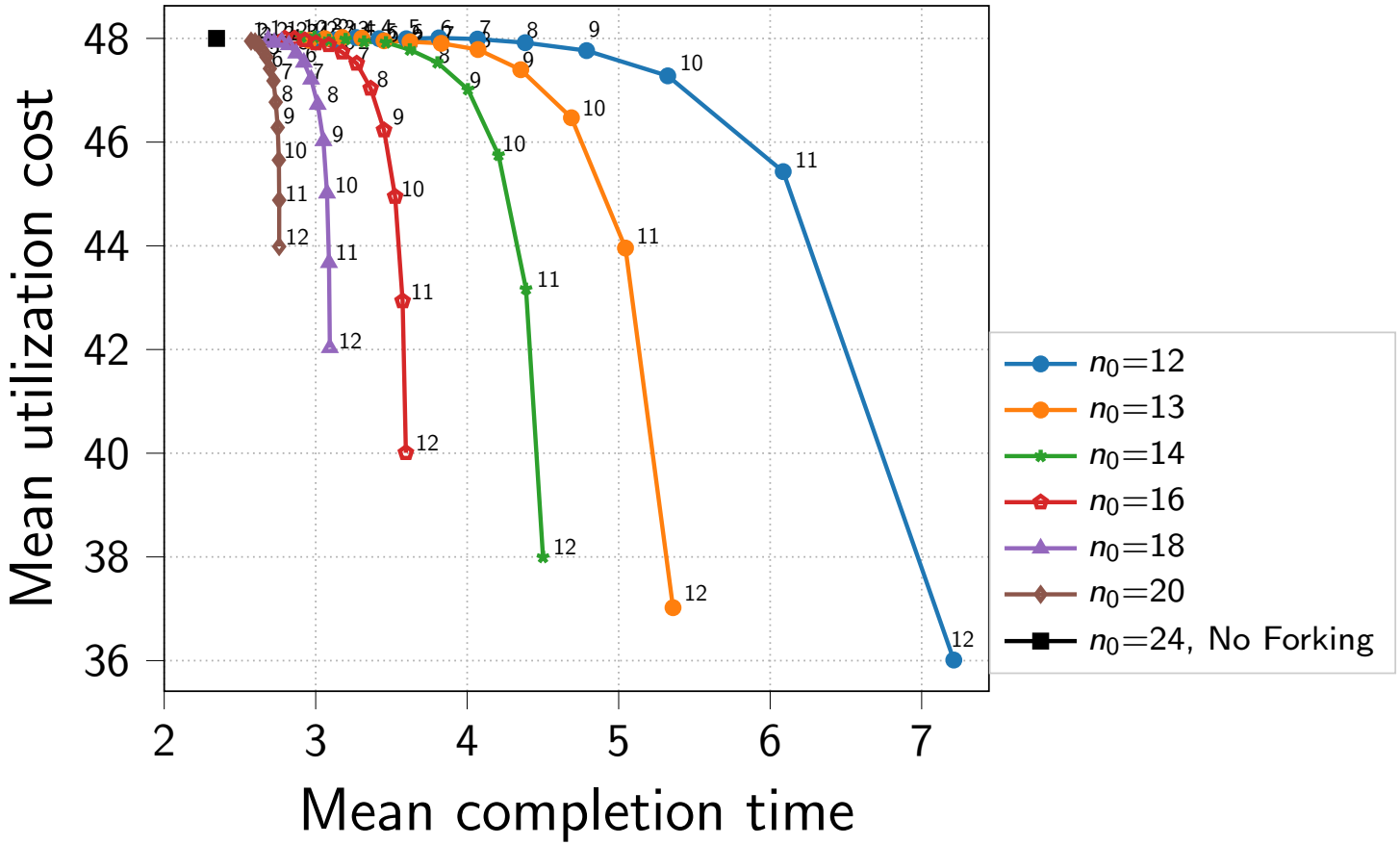


Decreasing in fork-task threshold



Increasing in fork-task threshold

Tradeoff when $n_0 \geqslant k$



- ⋆ Choice of initial servers matters
- ⋆ Fork-task threshold gives a true tradeoff when $n_0 \geqslant k$
- ⋆ Performance improvement !

Final remarks

Summary

- ⋆ Analyzed delayed relaunch
- ⋆ Single-fork analysis with initial servers, fork-task threshold
- ⋆ Computed mean completion time, mean utilization cost
- ⋆ Characterized the impact of initial servers and fork-task threshold on the two metrics
- ⋆ The choice of initial number of server matters

Future work

- ⋆ Multi-fork analysis with coded-redundancy
- ⋆ Forking for sequence of job arrivals