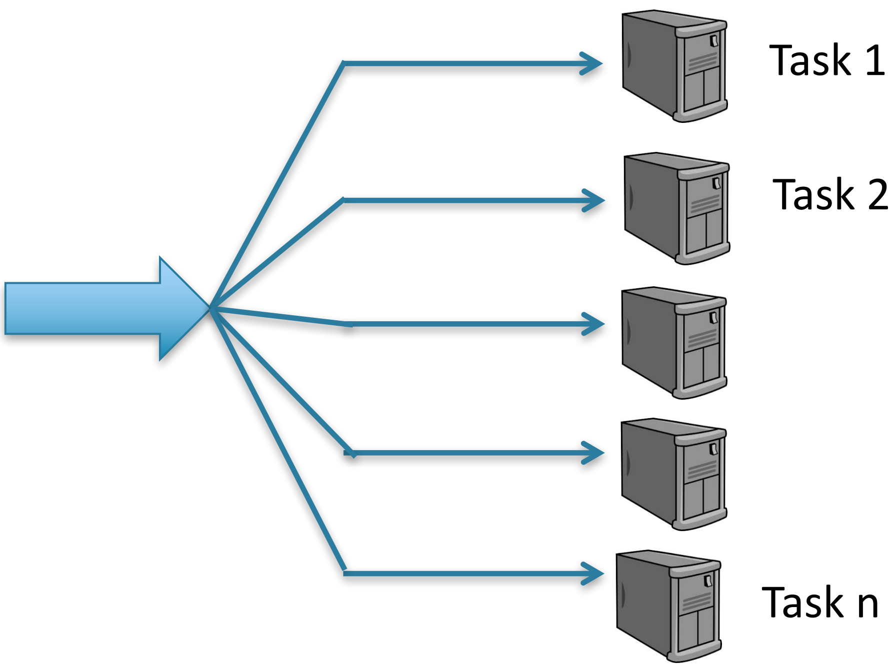
Using Straggler Replication to Reduce Latency in Large-scale Parallel Computing

Da Wang, Gauri Joshi, Gregory Wornell

Problem: Stragglers in Parallel Computing

- A job with hundreds of parallel tasks
- Machine response time can vary due to virtualization, congestion etc.
- The slowest tasks are the bottleneck in job completion

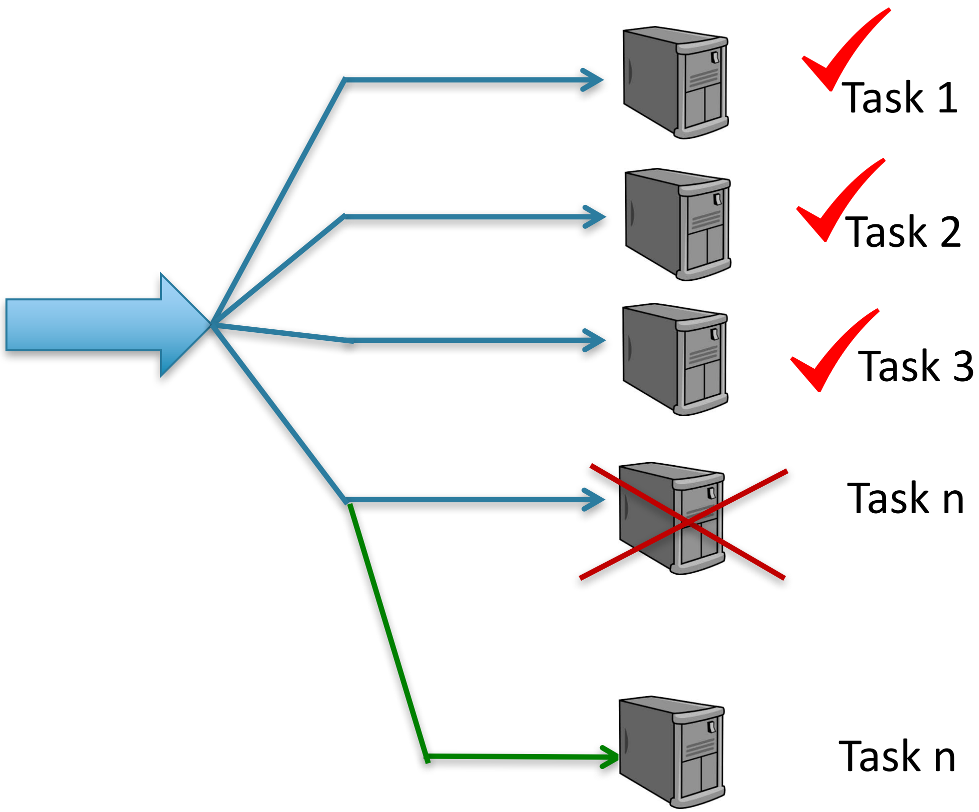


[Dean “Tail at Scale” 2013]

Latency	50%ile	99%ile
1 task finishes	1ms	10ms
All tasks finish	40ms	140 ms

Solution: Replication of Stragglers

Re-run the stragglers when p fraction of tasks are remaining



Related Previous Work

Task Replication in Systems Literature

- First used in MapReduce [Dean 2008] via back-up tasks
- Further developed in [Zaharia 2008], [Ananthanarayanan 2010] etc

Used in practice, but little theoretical analysis so far

Our Contributions

- Provide design insights on how to schedule task replication to reduce delay, with efficient use of additional resources

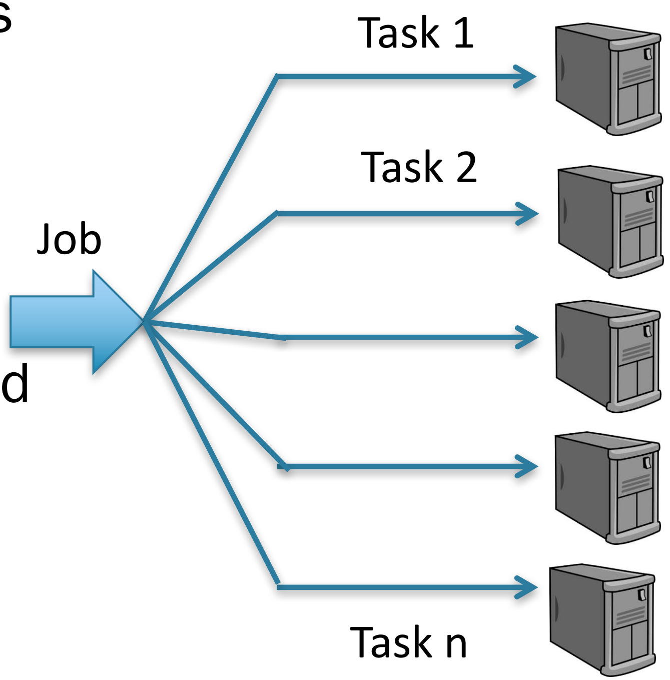
[1] D. Wang, G. Joshi, G. Wornell, " **Efficient Task Replication for Fast Response Times in Parallel Computation** ", SIGMETRICS 2014

[2] D. Wang, "Computing with Unreliable Resources", PhD Thesis, MIT, 2014

[3] G. Joshi, "Efficient Redundancy Techniques to Reduce Delay in Cloud Systems", PhD Thesis, MIT 2016

System Model

- A job with n parallel tasks, n is large
- Finish time of a task, $X \sim F_X$, i.i.d. across machines.



Remark on the i.i.d assumptions:

- From cloud user's point of view, all rented machines are approx. identical.

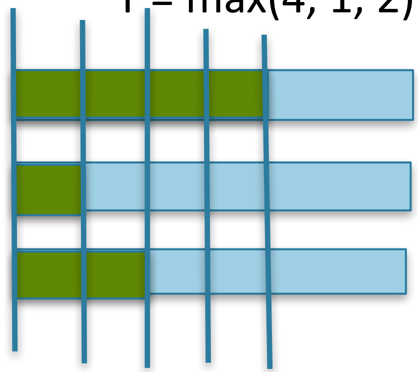
Performance Metrics

- Expected Latency E[T] = Expected Time when all tasks finish
- Expected Cost E[C]= Expected total machine time spent, normalized by # of tasks

T = max(4, 1, 2) = 4

One job with 3 tasks

C = (4
+1
+2)/3
=7/3



Remark on cost metric

- There could be other costs – network, memory usage, etc

Outline

Analysis of E[T] and E[C] using Extreme Value Theory

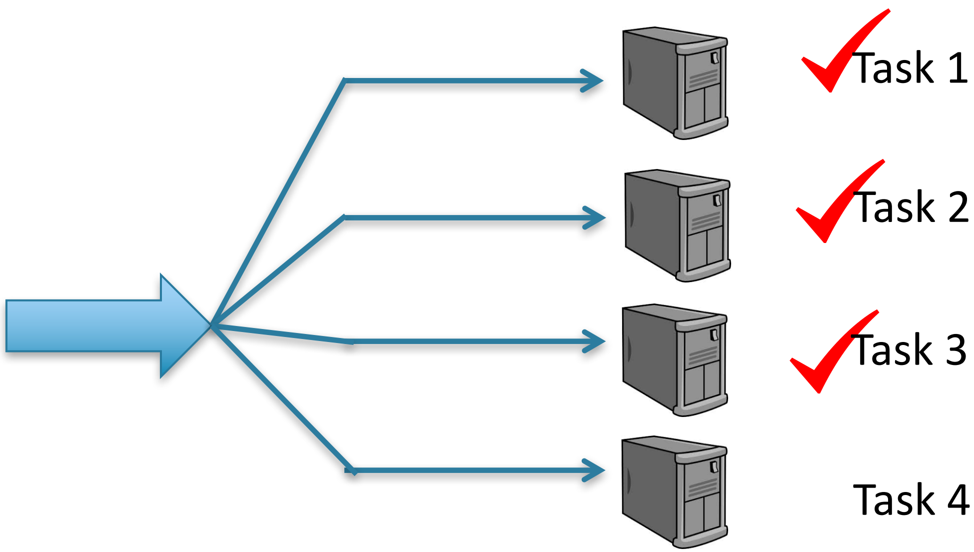
- Tail behavior of F_X is a key factor affecting the E[T]-E[C] trade-off

Heuristic Algorithm to find best replication strategy

- Compare with back-up tasks in MapReduce using Google trace data

Replication Policy: When to re-run?

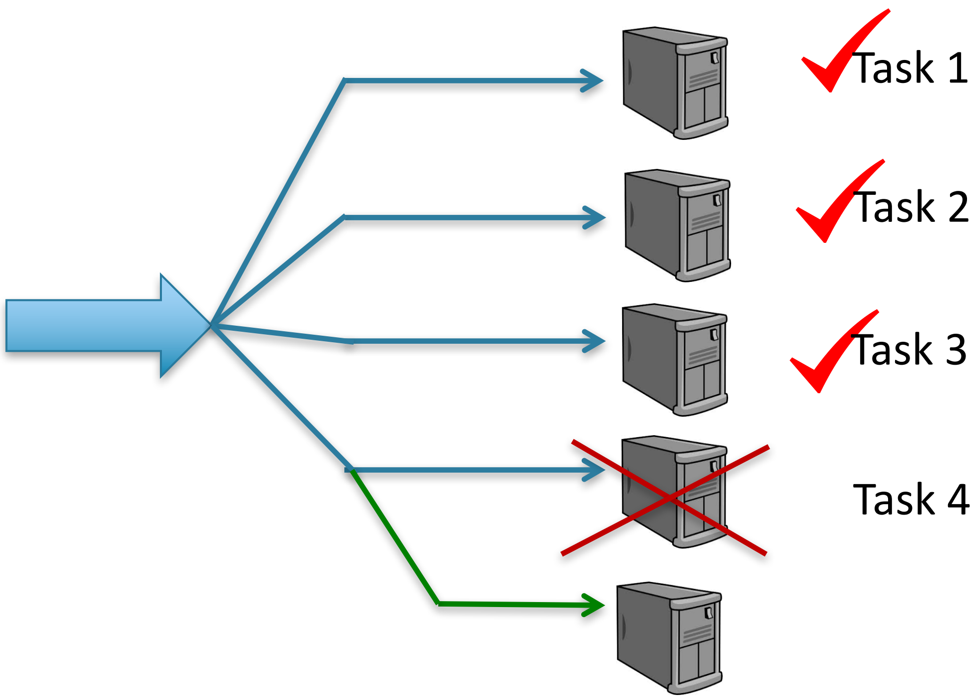
- Re-run the stragglers when *p* fraction of tasks are left



$p = 25\%$

Replication Policy: When to re-run?

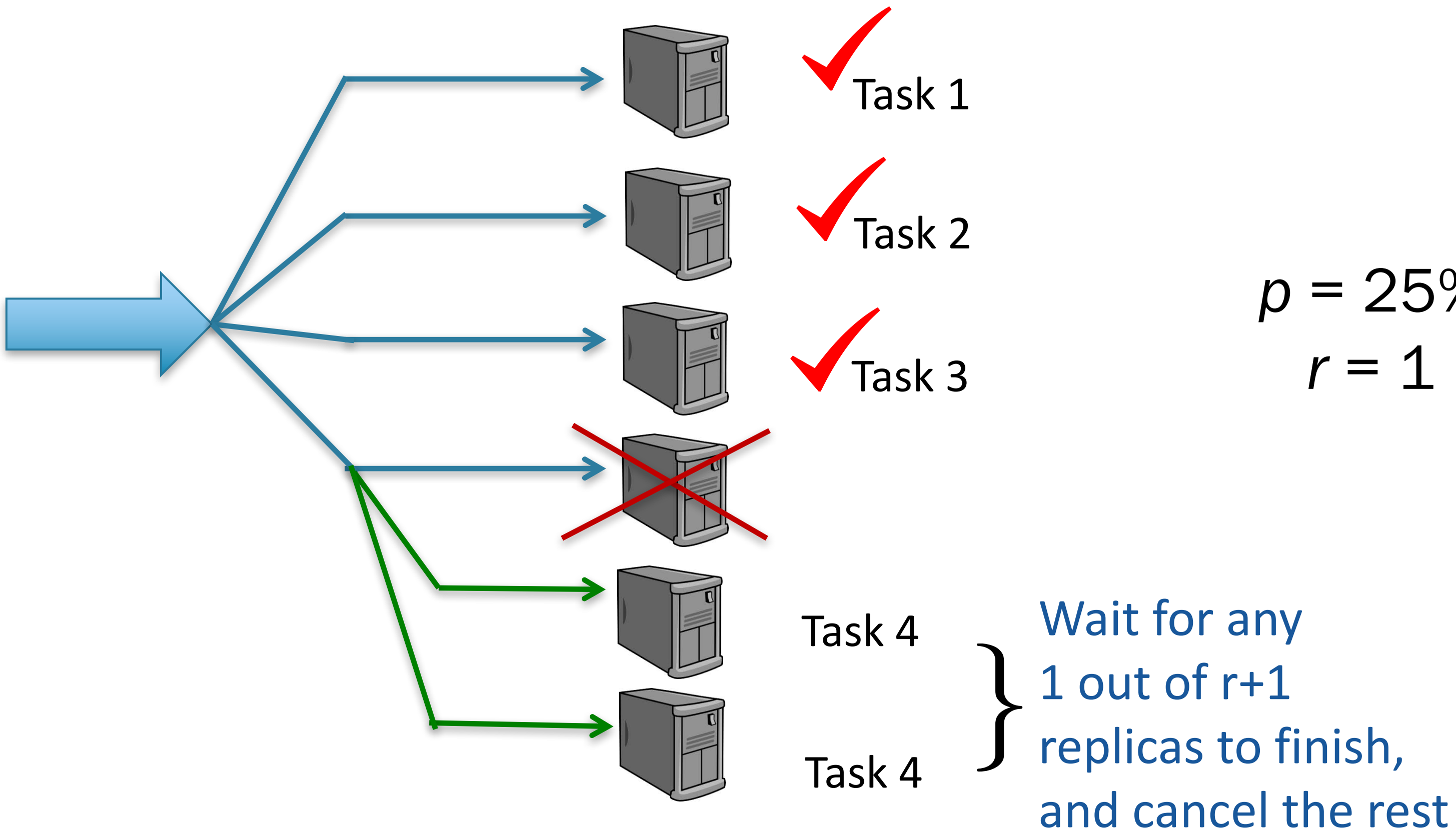
- Re-run the stragglers when p fraction of tasks are left



$p = 25\%$

Replication Policy: How many replicas?

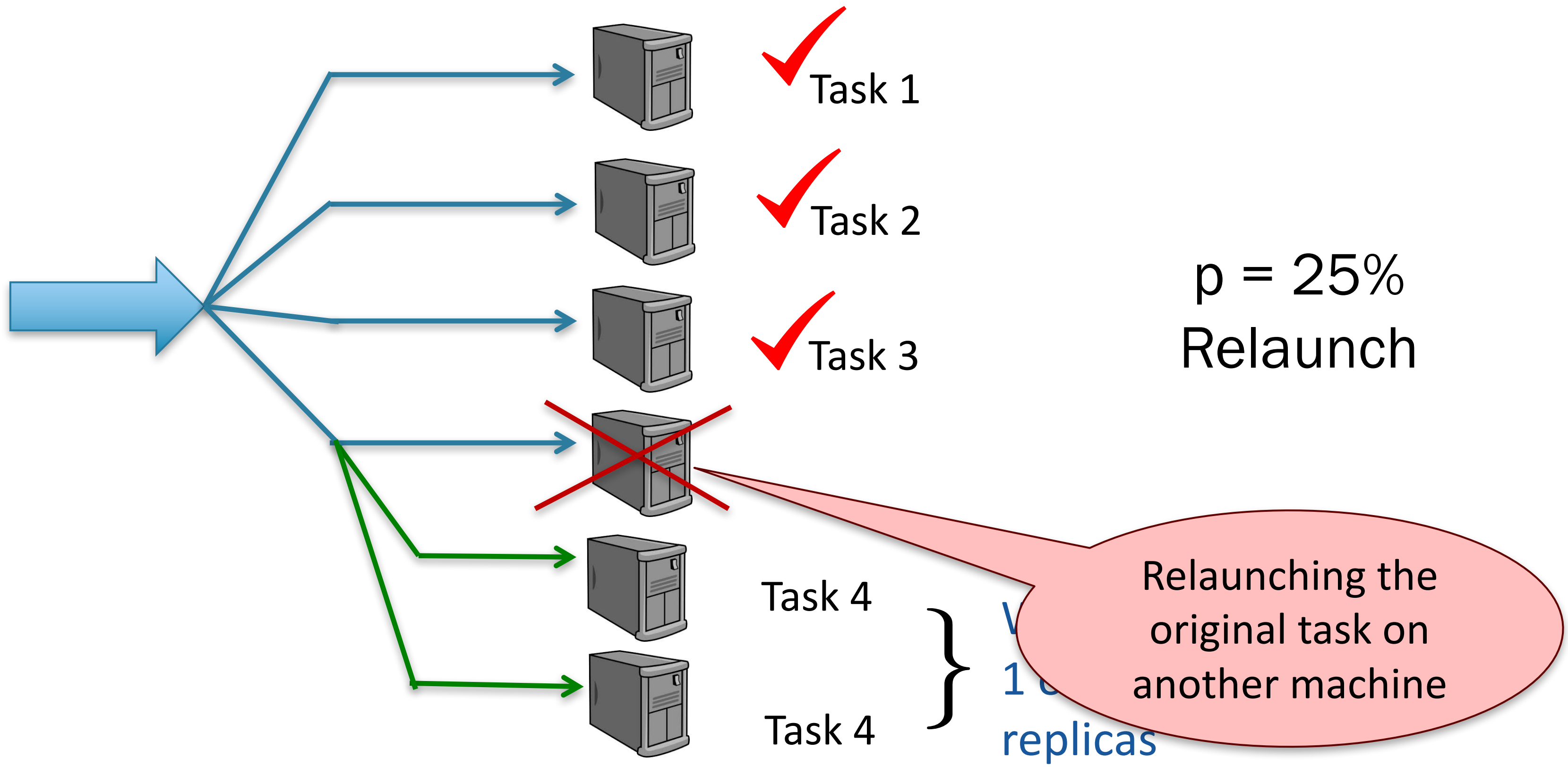
- Re-run the stragglers when *p* fraction of tasks are left
- Run *r* additional replicas



p = 25%
r = 1

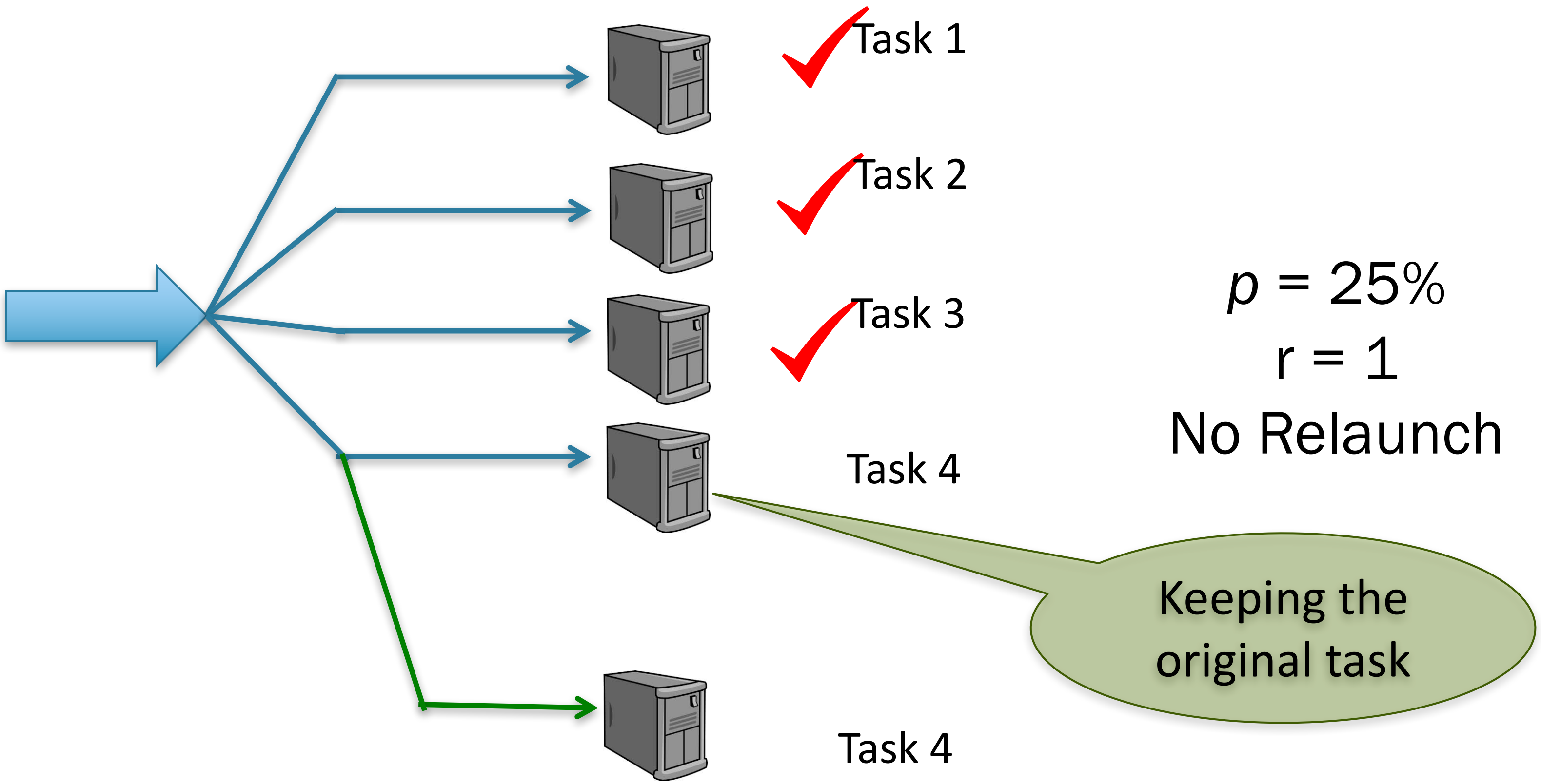
Replication Policy: Relaunch or not?

- Re-run the stragglers when p fraction of tasks are left
- Run r additional replicas



Replication Policy: Relaunch or not?

- Re-run the stragglers when *p* fraction of tasks are left
- Run *r* additional replicas



p = 25%

r = 1

No Relaunch

Problem Formulation

Given n tasks, and task finish time distribution F_X,

Design Parameters

- p: Fraction of tasks left when we replicate
- r: Number of additional replicas
- Relaunch original straggling task or not

Performance Metrics

- Latency E[T]
- Cost E[C]

Evaluating Expected Latency E[T]

- Wait for (1-p)n tasks to finish
- Launch replicas of the pn stragglers
  - Time for 1 out *r+1* copies to finish Y ~ F_Y =g(F_X, r, kill/keep)
  - For e.g. r= 1 with task-killing → $(1-F_Y) = (1-F_X)^2$

$$T = X_{(1-p)n:n} + Y_{pn:pn}$$

Wait for (1-p)n out of n tasks to finish

Maximum of finish times of the *pn* stragglers after replication

Notation $X_{k:n}$: k^{th} smallest of n i.i.d. rvs $X_1, X_2, .. X_n$

Evaluating Expected Latency E[T]

$$\mathbb{E}[T] = \mathbb{E}[X_{(1-p)n:n}] + \mathbb{E}[Y_{pn:pn}]$$

Central Value Theorem — n -> ∞ :

$$F_X^{-1}(1-p)$$

Extreme Value Theorem — n -> ∞ :

Different behavior for Exponential, Light or Heavy tailed Y

Asymptotic approx for n -> ∞ is close to simulation even for n ~ 300

Exercise: Task Execution Time X ~ Exp(μ)

$$\mathbb{E}[T] = \mathbb{E}[X_{(1-p)n:n}] + \mathbb{E}[Y_{pn:pn}]$$

Central Value Theorem

n -> ∞

$$F_X^{-1}(1-p)$$

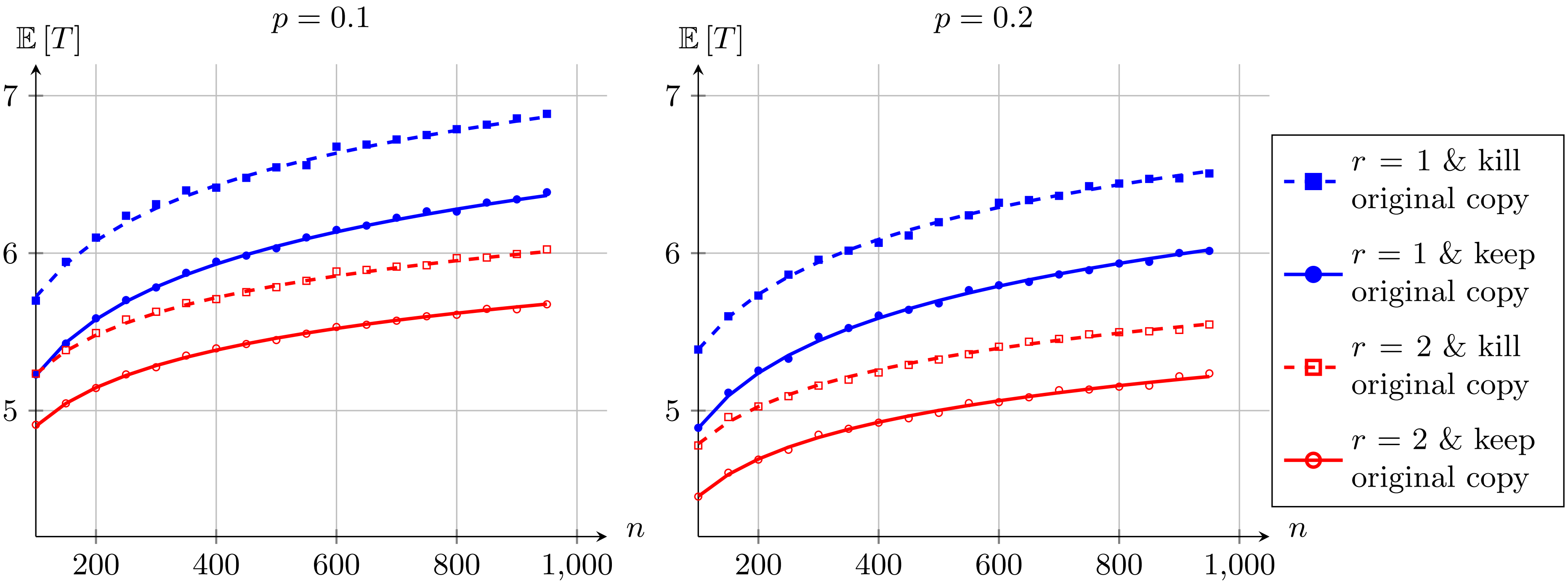
Extreme Value Theorem

n -> ∞

Different behavior for Exponential, Light or Heavy tailed Y

Comparing Theoretical Analysis with Simulations

X ~ 1+ Exp(1), and n = 400



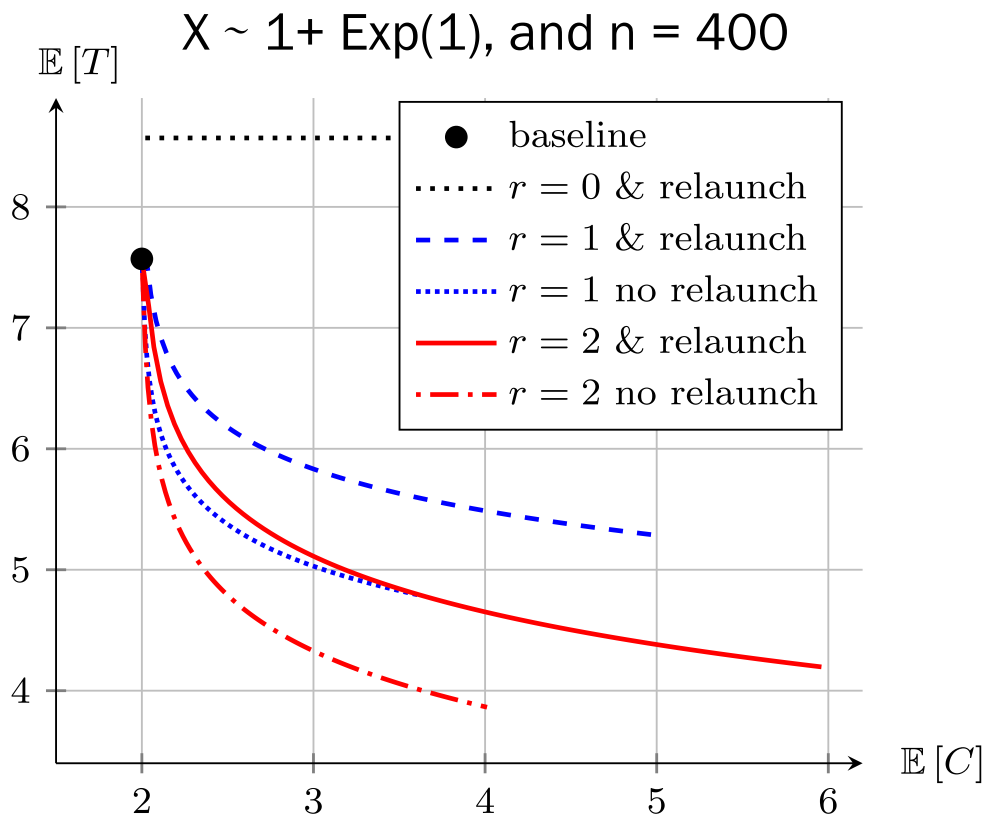
Evaluating Expected Cost E[C]

$$\mathbb{E}[C] = \frac{1}{n} \sum_{i=1}^{(1-p)n} \mathbb{E}[X_{i:n}] + \frac{np}{n} \mathbb{E}[T^{(1)}] + \frac{1}{n} \sum_{j=1}^{pn} (r+1)\mathbb{E}[Y]$$

$$= \int_0^{1-p} F_X^{-1}(h)dh + pF_X^{-1}(1-p) + (r+1)p\mathbb{E}[Y] + O(1/n)$$

By Central
Value Theorem

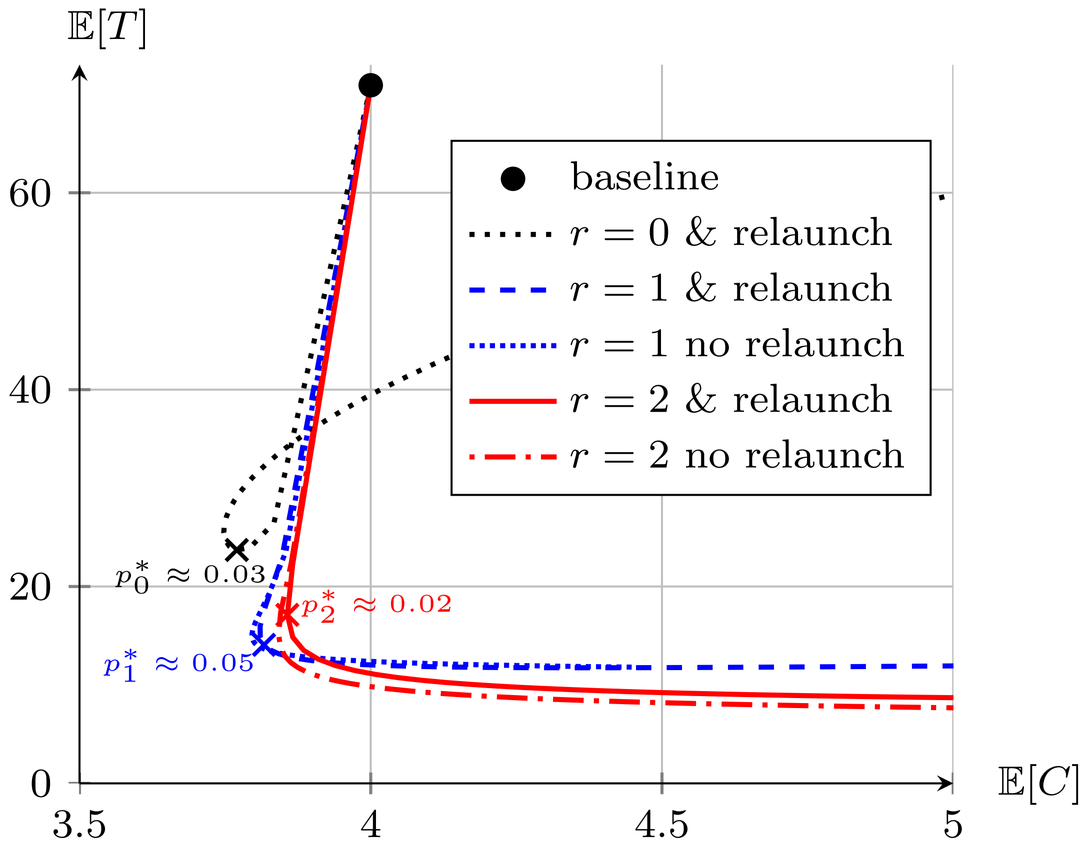
Case: Shifted Exponential (Exp. tail)



- Increasing p and r reduces latency but increases cost
- Killing a straggling task never helps!

Case: Pareto (Heavy tail)

X ~ Pareto (2, 2) and n =400



Latency and cost both reduce for small p!

Outline

Analysis of E[T] and E[C] using Extreme Value Theory

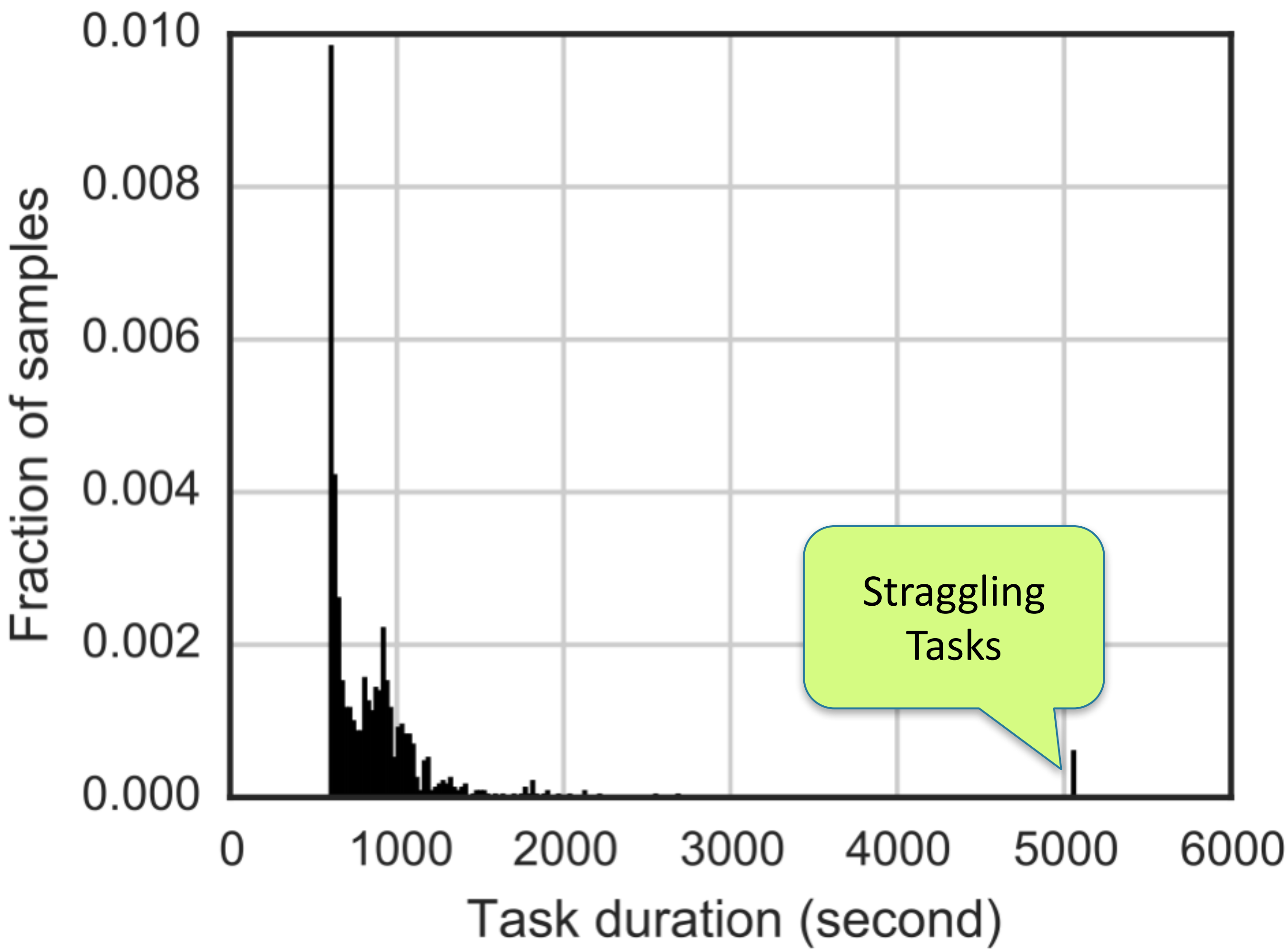
- Tail behavior is a key factor affecting the E[T]-E[C] trade-off

Heuristic Algorithm to find best replication strategy

- Comparing with back-up tasks in MapReduce using Google trace data

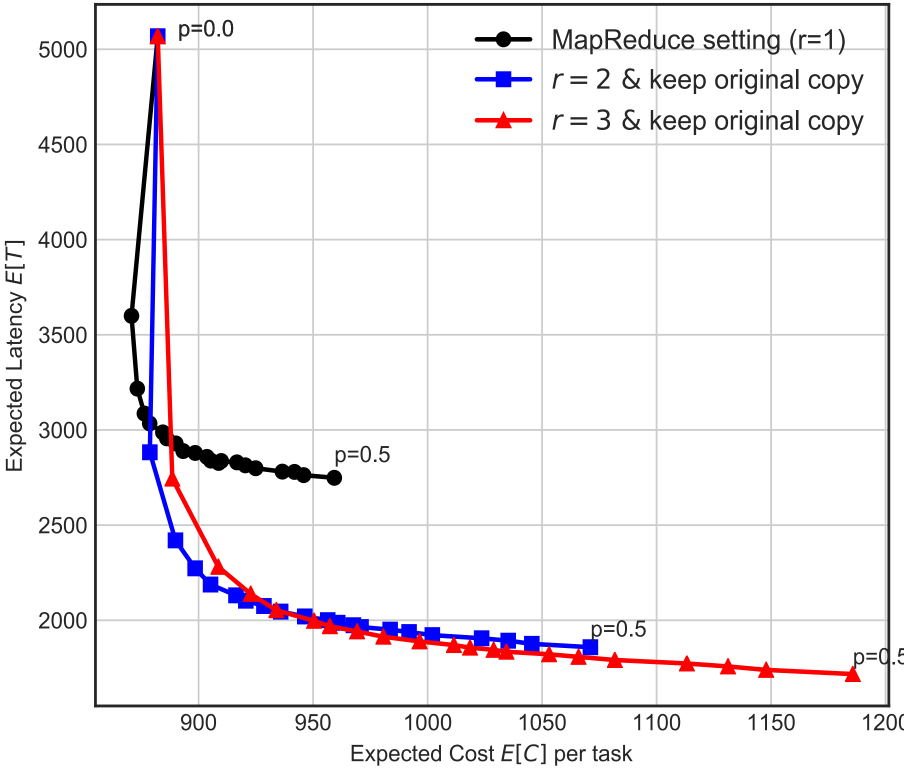
Example: Job with 1026 tasks

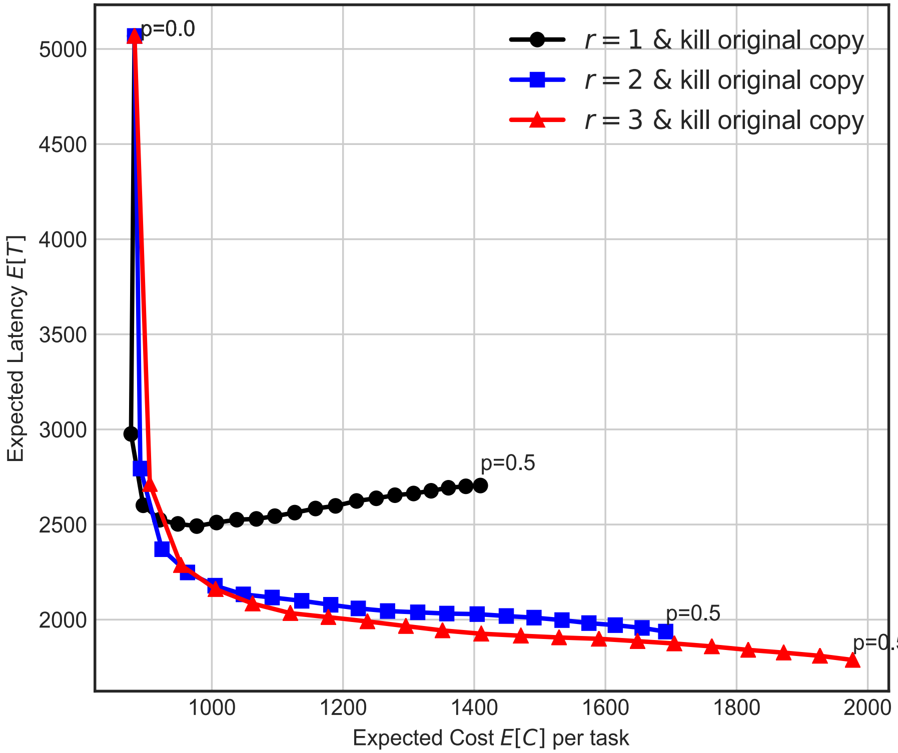
Google Cluster Data



Simulations using Google Cluster Data

Latency-Cost Trade-off

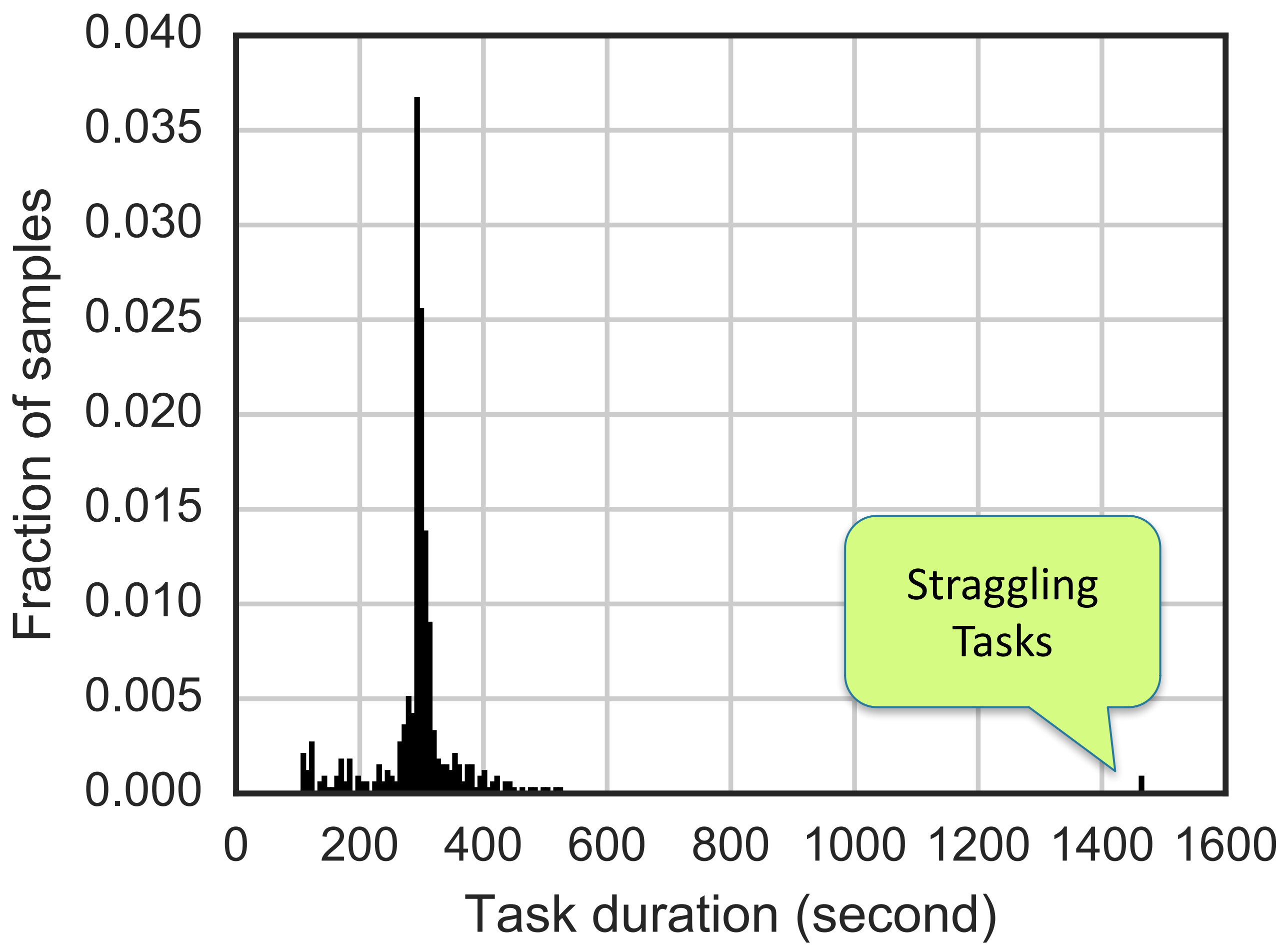




Careful choice of replication strategy can be better than the default in MapReduce

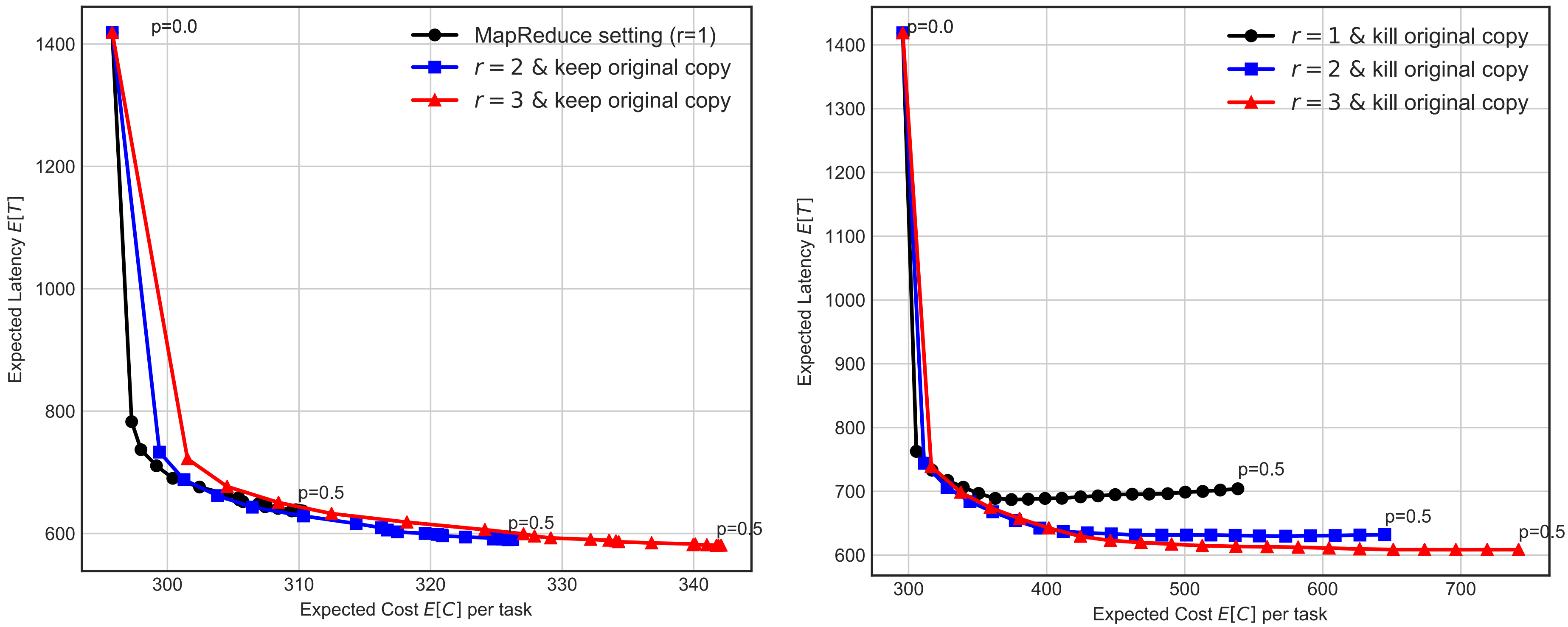
Example: Job with 488 tasks

Google Cluster Data



Simulations using Google Cluster Data

Latency-Cost Trade-off



Heuristic Search of the Best Strategy

May be hard to use our analysis to optimize the strategy for any F_X

- Analysis of E[T] and E[C] after replication can be hard

ESTIMATION

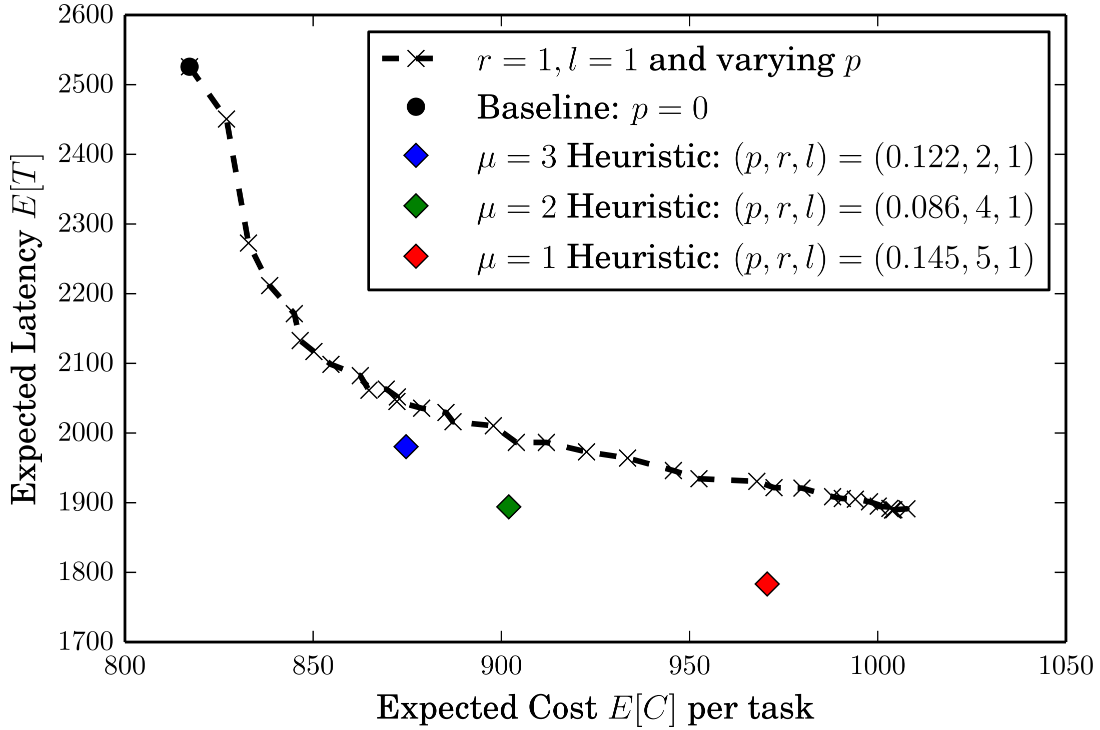
- Estimate F_X from traces of task execution time
- Use empirical F_X to estimate $J = E[T] + \mu E[C]$ for given p, r, relaunch/not

HEURISTIC ALGORITHM

1. For given p, choose r, and the kill/keep strategy that minimizes J
2. Perform gradient descent on p

Heuristic Algo: Resulting E[T] and E[C]

- Run heuristic algorithm with different μ, to minimize J = E[T] +μ E[C]
- r= 1, without relaunch (l=1): Back-up tasks option in MapReduce



Concluding Remarks

SUMMARY

- Tail behavior is important in choosing the right policy
  - Pareto, Shifted Exponential etc.
- Heuristic algorithm to find good replication policy given traces of execution time

RELATED AND FUTURE WORK

- Queueing of jobs (next class)
- Online algorithm to learn F_X and schedule simultaneously