Effective Straggler Mitigation: Attack of the Clones

Ganesh Ananthanarayanan, Ali Ghodsi,
Scott Shenker, Ion Stoica

Interactive Data Analytics

- Common in today's clusters, expected to grow

- Exploratory and experimental jobs
  - Data analyst querying small sample (**interactive**)
- **Low latency** is crucial for interactive jobs

➢ Interactive jobs are *small*
  - Facebook: 88% of jobs operate on 20GB of data and contain fewer than 50 tasks

Stragglers in Small Jobs

- Small interactive jobs are sensitive to **stragglers**
  - *Tasks that are much slower than the rest in the job*

- Straggler Mitigation:
  - Blacklisting: Eliminate machines with faulty hardware (e.g., erroneous disks)
  - Speculation: **LATE** [OSDI'08], **Mantri** [OSDI'10]…
    - Address the **non-deterministic** stragglers

Despite the mitigation techniques…

- …in production clusters

➢ LATE: The slowest task runs **8 times slower*** than the median task in a job

➢ Mantri: The slowest task runs **6 times slower*** than the median task in a job

- (but they work well for large jobs…)

**we compare progress-rate of tasks, i.e., input-size/duration*

State-of-the-art Straggler Mitigation

Speculative Execution:

1. ***Observe***: measure relative progress of tasks

2. ***Speculate***: launch speculative copies of straggler tasks

Why doesn't this work for small jobs?

1. Consist of just a **few tasks**
   - **Statistically hard** to predict stragglers accurately

2. Run all their tasks **simultaneously**
   - Observing constitutes **large fraction** of job's duration

***Observe & Speculate* is ill-suited to address stragglers in small jobs**

Cloning Jobs

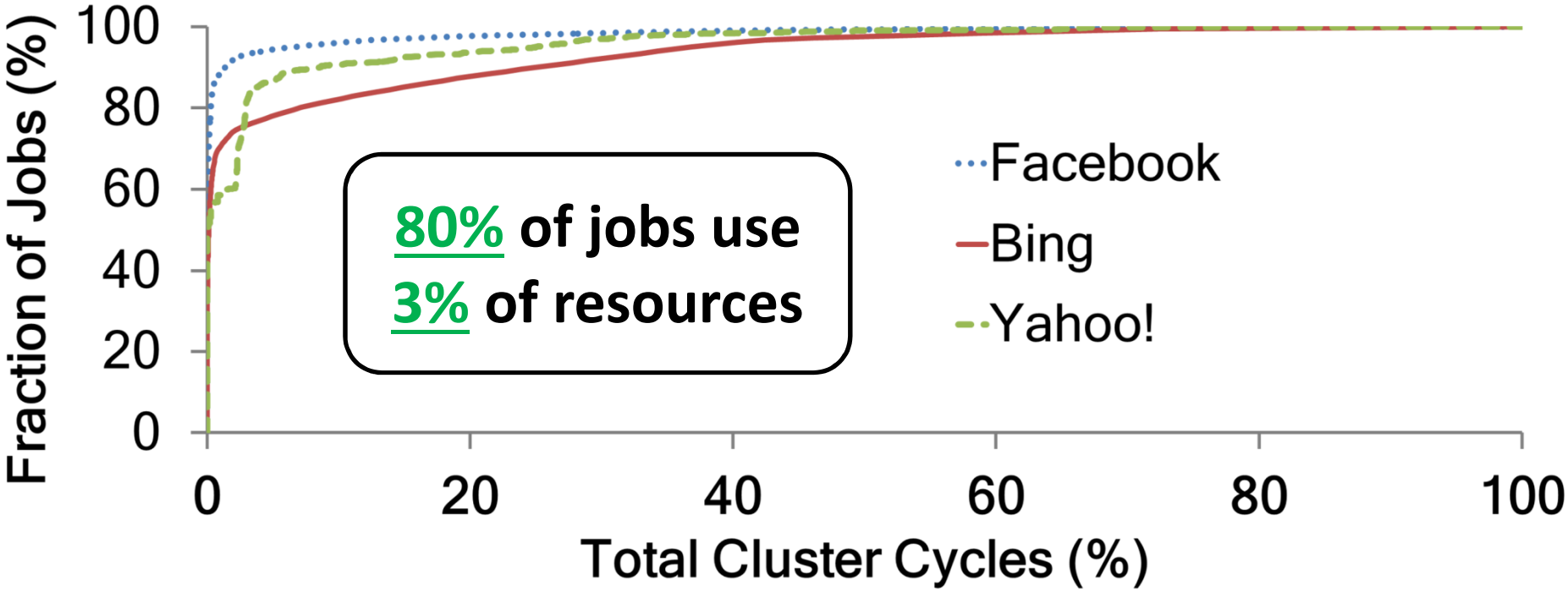
- **Proactively** launch **clones** of a job, just as they are submitted
- Pick the result from the **earliest** clone
  - ***Probabilistically mitigates stragglers***

- Eschews observe & speculate, causal analysis…

Is this feasible in practice?

Heavy-tailed Distribution

- *Production clusters for data analytics*



Can clone small jobs with few extra resources

Cloning for **Stragglers** in **Small** Jobs

- Interactive jobs are important and small
- **Hardest** for straggler mitigation techniques
  - Traditional reactive approach is insufficient
- **Heavy-tailed** distribution → **cloning** is feasible

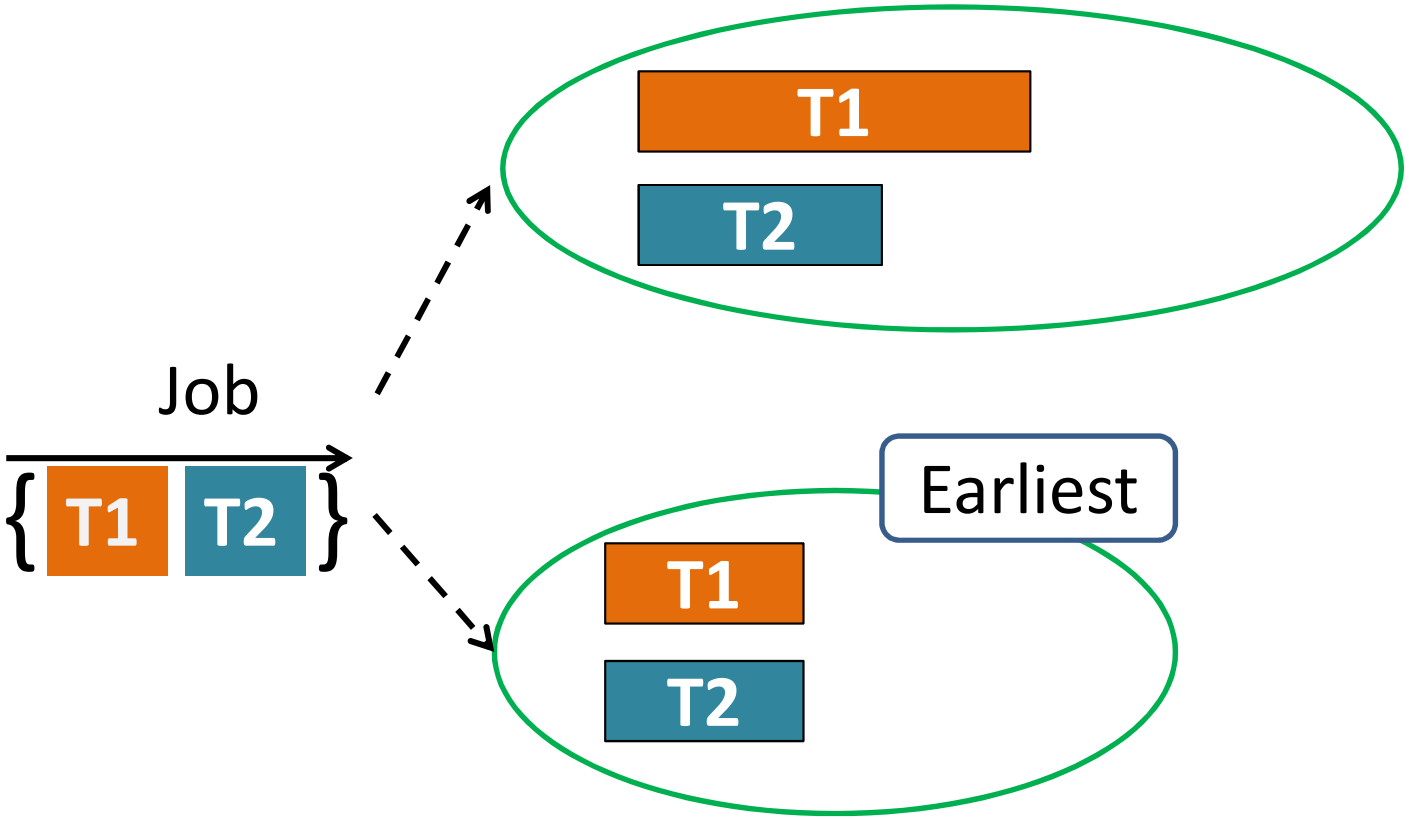
Challenge: Avoid **I/O contention**

- Every clone should get its own copy of data

- **Input** data of jobs
  - Replicated three times (typically) by file system

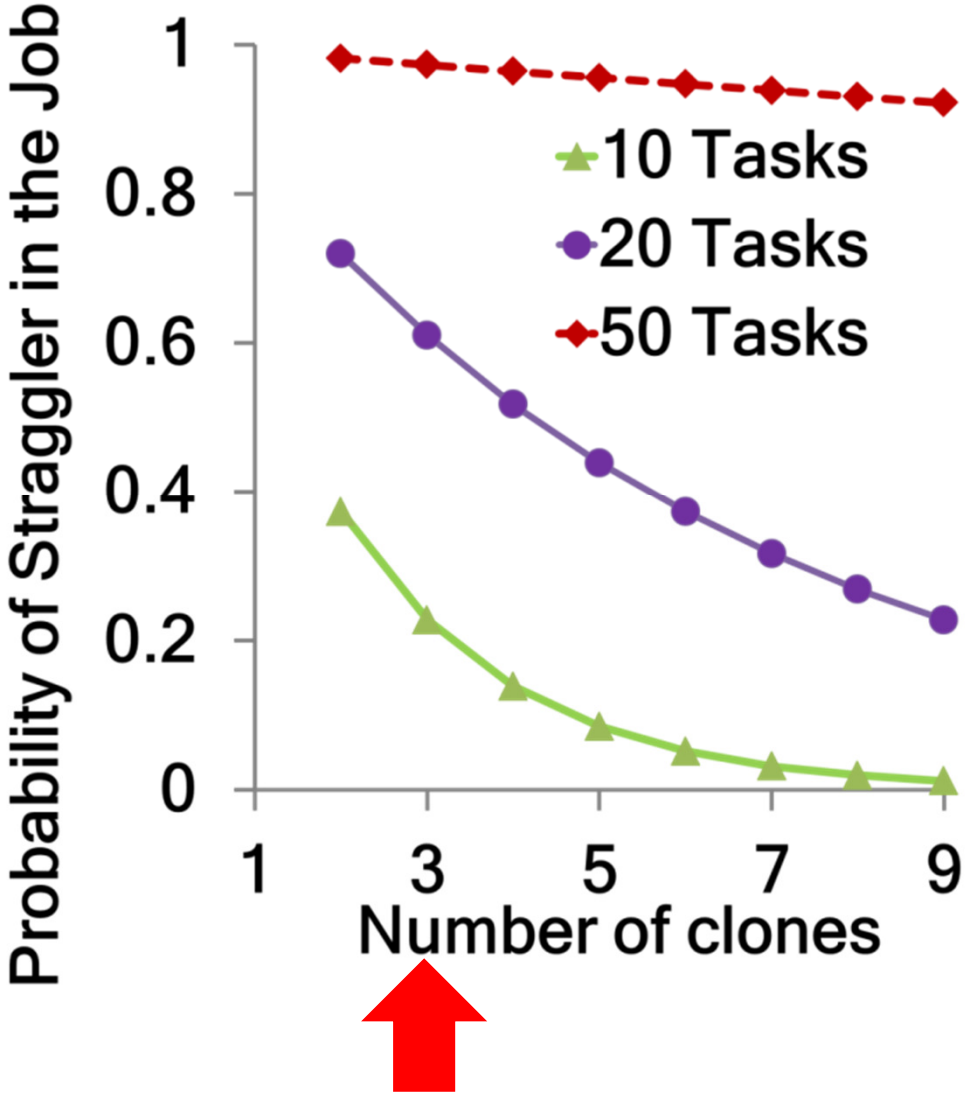
- **Intermediate** data of jobs
  - Not replicated at all, to avoid overheads

Strawman: Job-level Cloning



- ✓ Easy to implement
- ✓ Directly extends to any framework

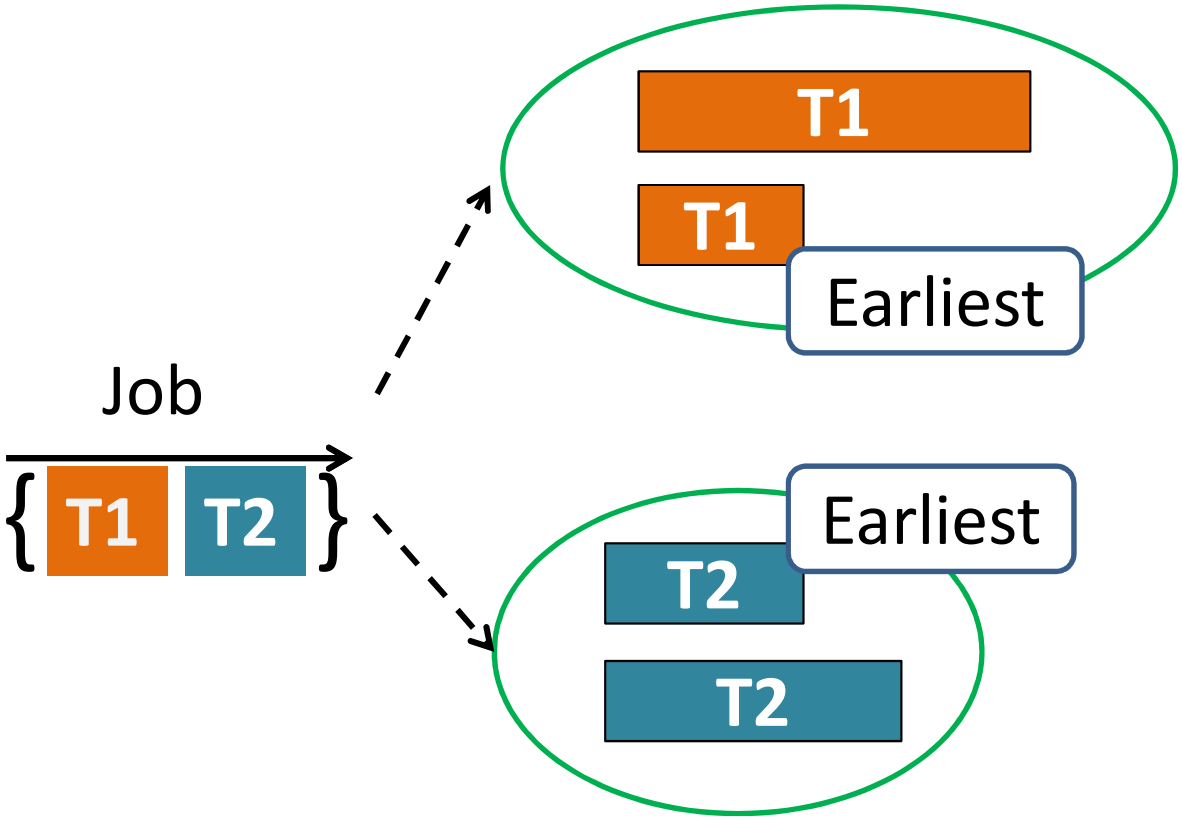
Number of clones



» **3** clones

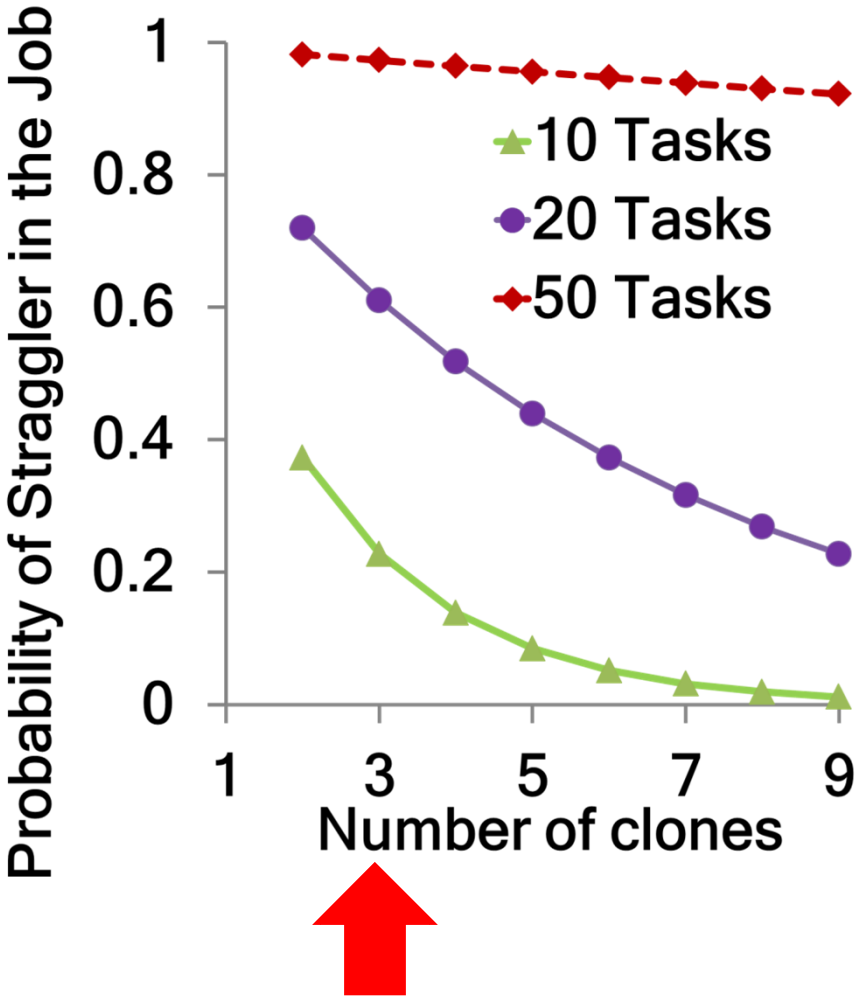
- **Contention** for input data
- Storage crunch, can't replicate more

Task-level Cloning

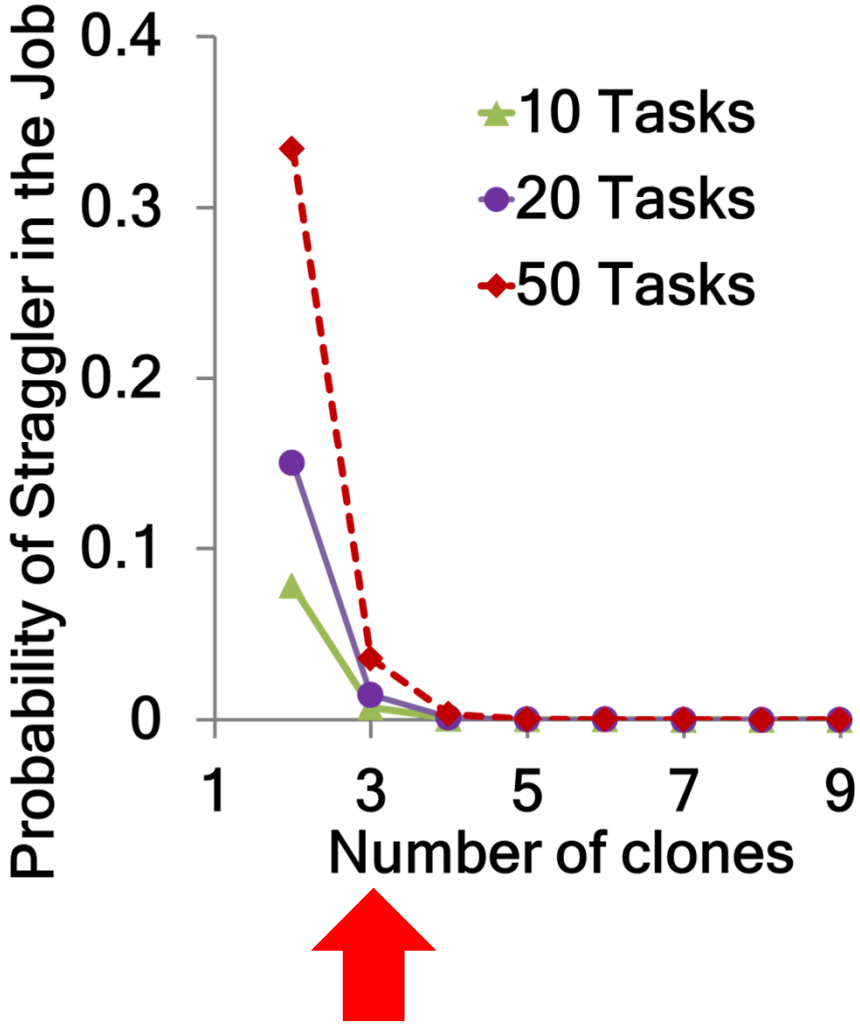


3 clones are plenty!

Strawman

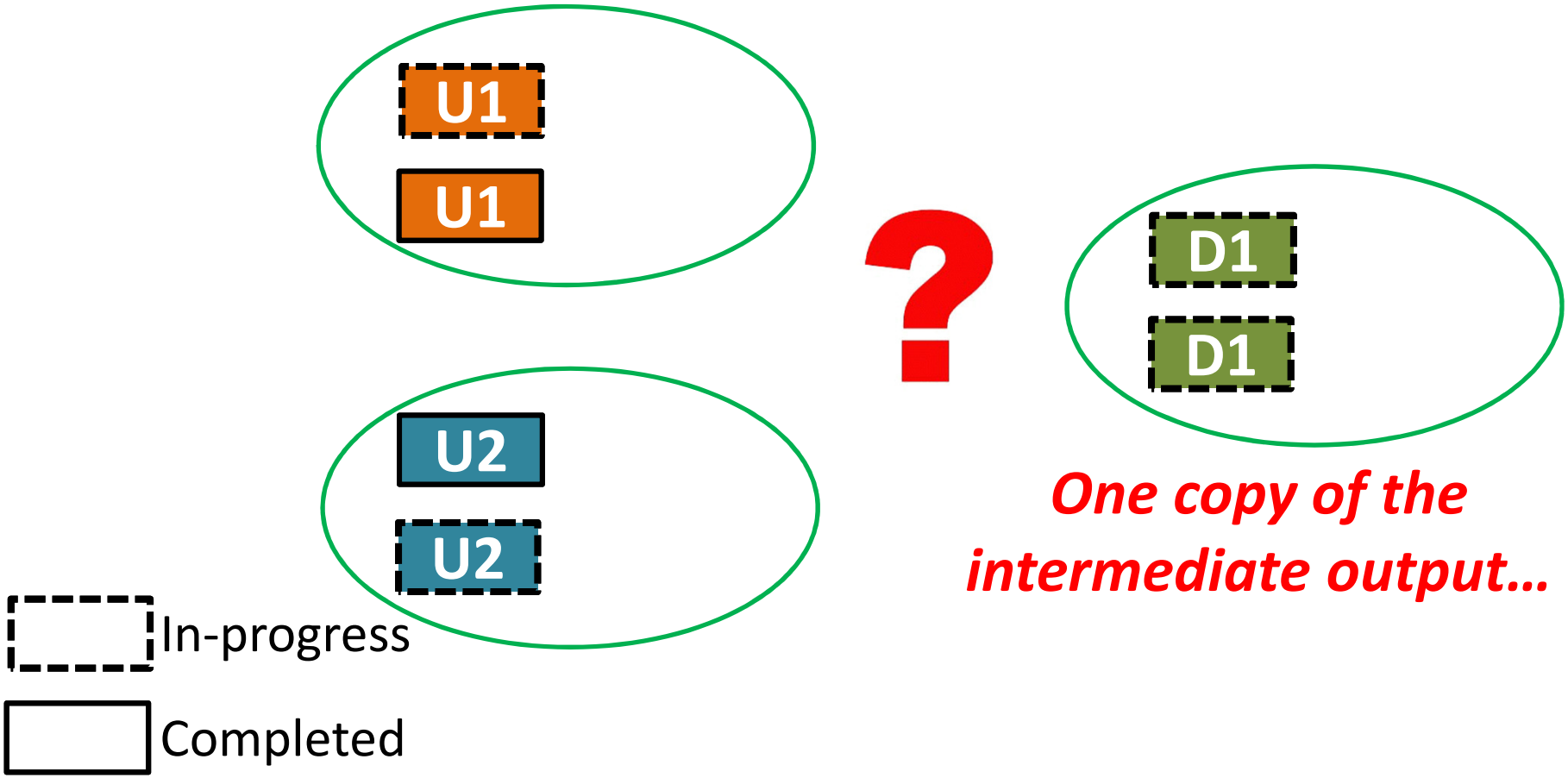


Task-level Cloning



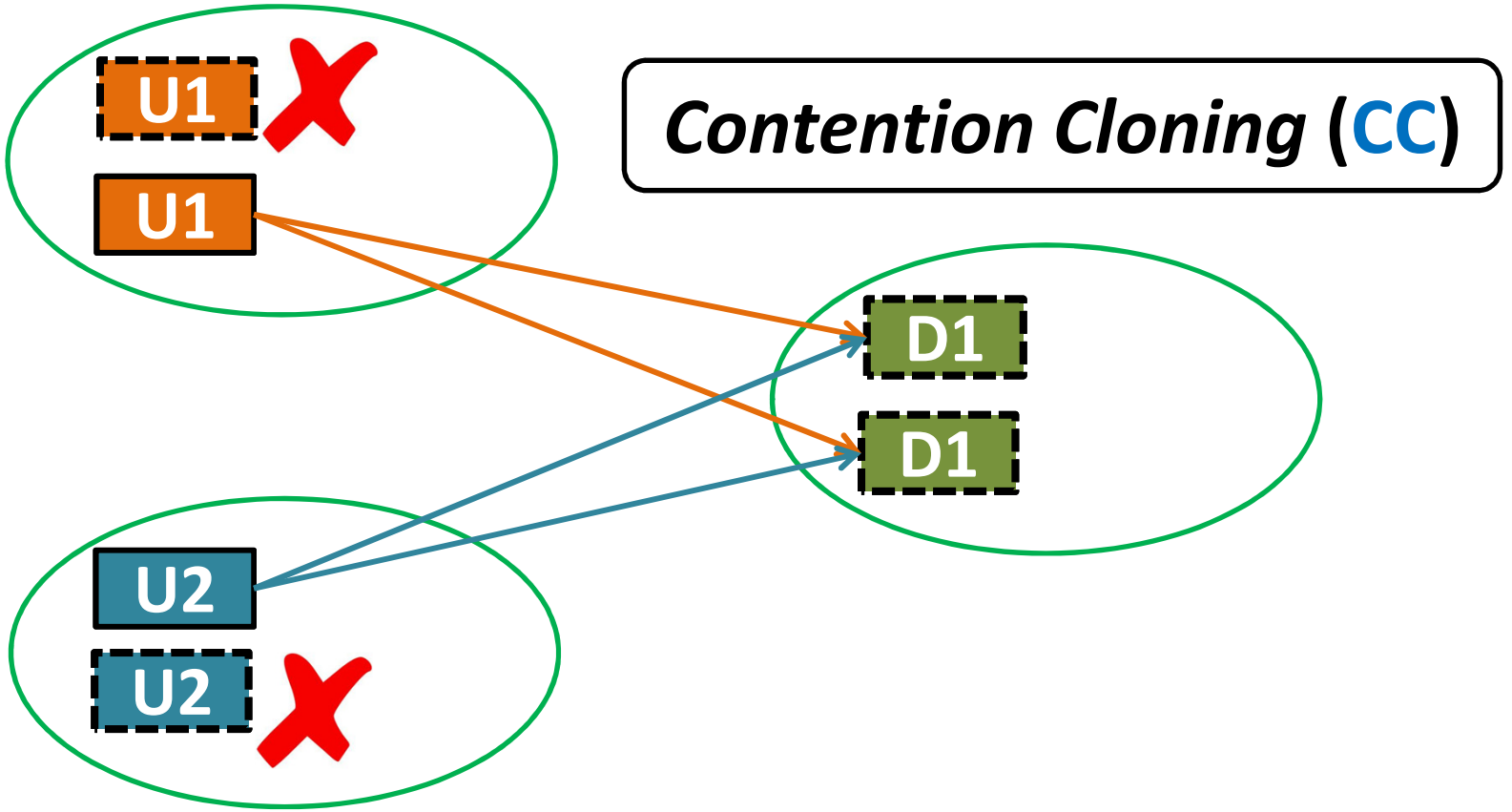
Intermediate data reads?

- Jobs consist of DAG of tasks
  - Downstream tasks read outputs of upstream tasks

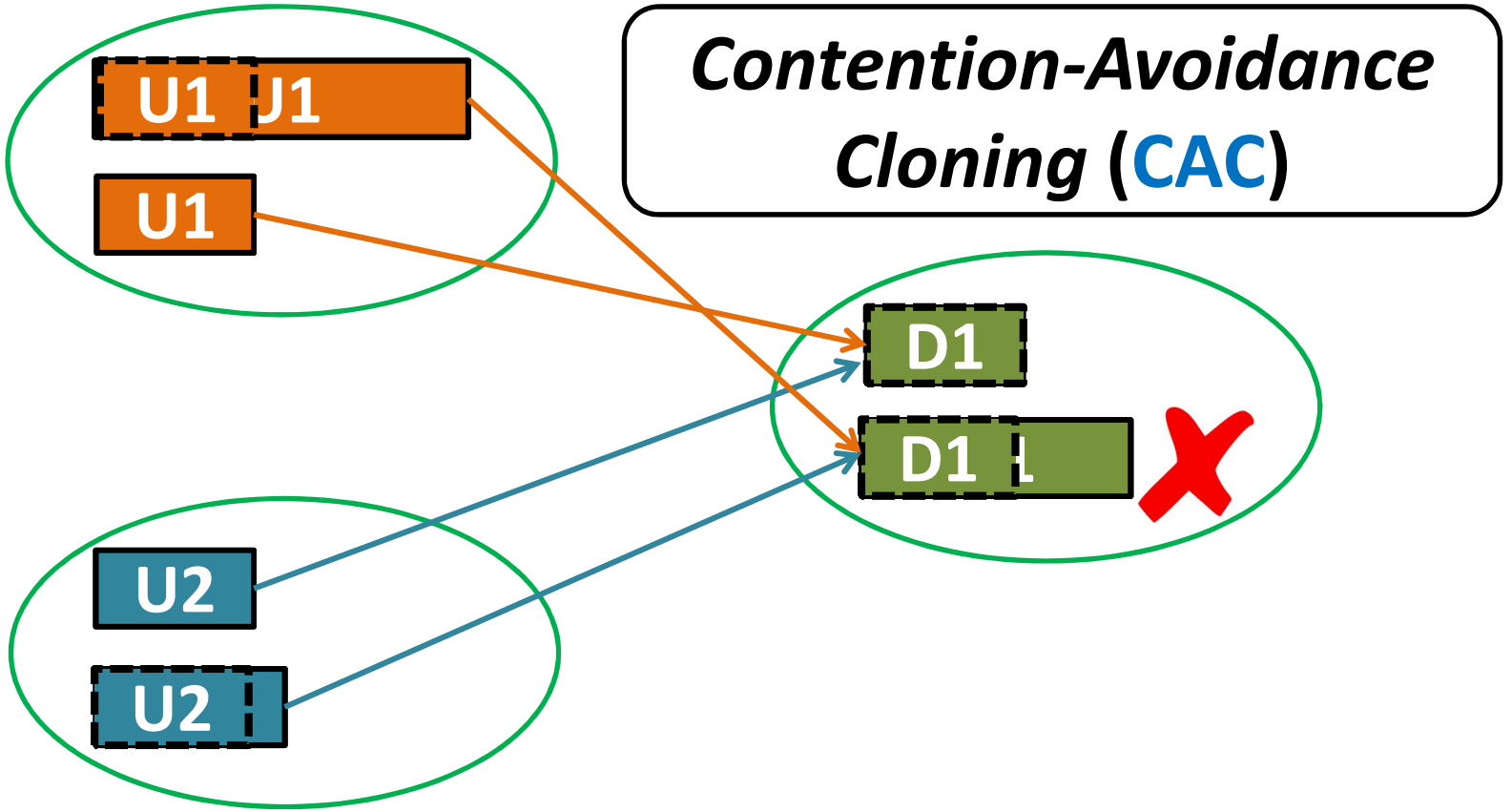


One copy of the intermediate output…

Assign **Earliest** Copy



Assign **Exclusive** Copy



Jobs are more vulnerable to stragglers

CAC vs. CC

- **CAC avoids contentions** but **increases vulnerability to stragglers**
  - *Straggler probability in a job increases by >10%*
- **CC mitigates stragglers** in jobs but **creates contentions**
  - *Intermediate data transfer takes ~50% longer*

How to minimize contention without straggling downstream tasks?

Delay Assignment

➢ *Distinguish* ***intrinsic variations in task completions*** *from* ***stragglers***

- **Small delay** to get *exclusive* copy before *contending* for the available copy
  - **Probabilistic model** of task durations and read b/w
  - (*Similar to delay scheduling* [EuroSys'10])

- Delay updated automatically and periodically

Dolly: Cloning Jobs

- **Task-level** cloning of jobs
- **Delay Assignment** to manage intermediate data
- Works within a **resource budget**
  - Clone only if resources are available



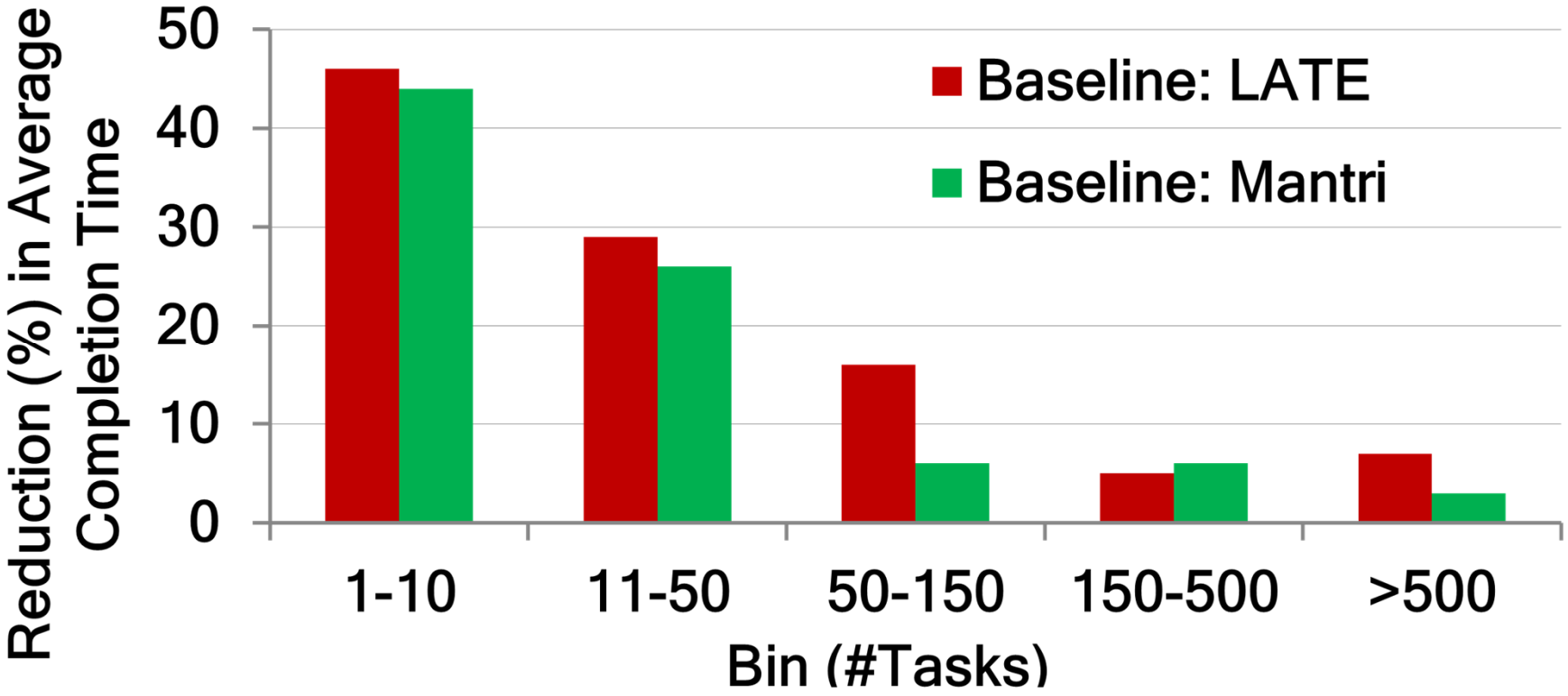
How effective is **Dolly**?

- Baselines: **LATE** or **Mantri**, + **blacklisting**
- Cloning **budget: 5%**

- Workload from Facebook and Bing traces
  - 1000's of nodes, Hadoop and Dryad jobs
- Implemented in Hadoop, 150 node deployment

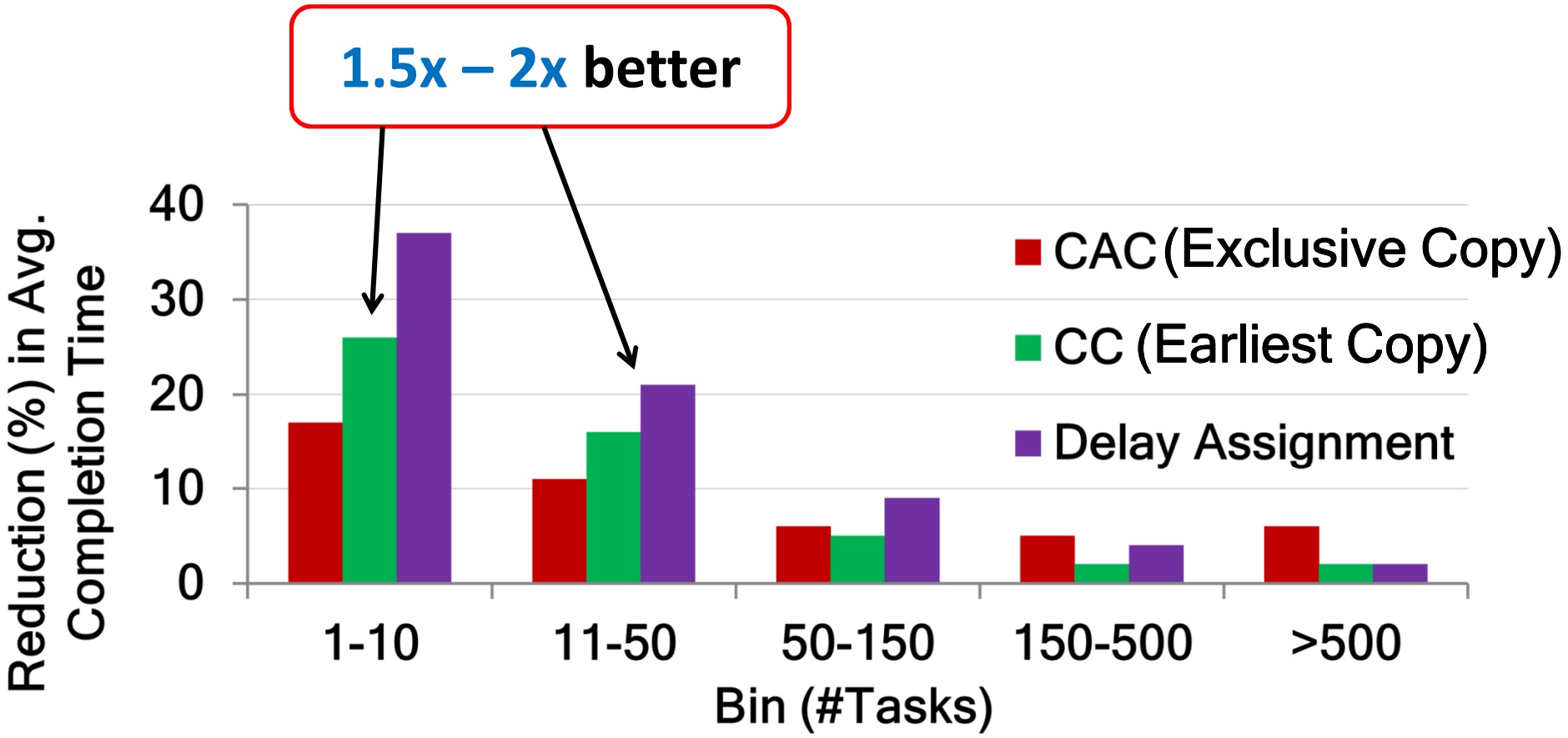
Average job completion time

Jobs are 44% and 42% faster w.r.t. LATE and Mantri



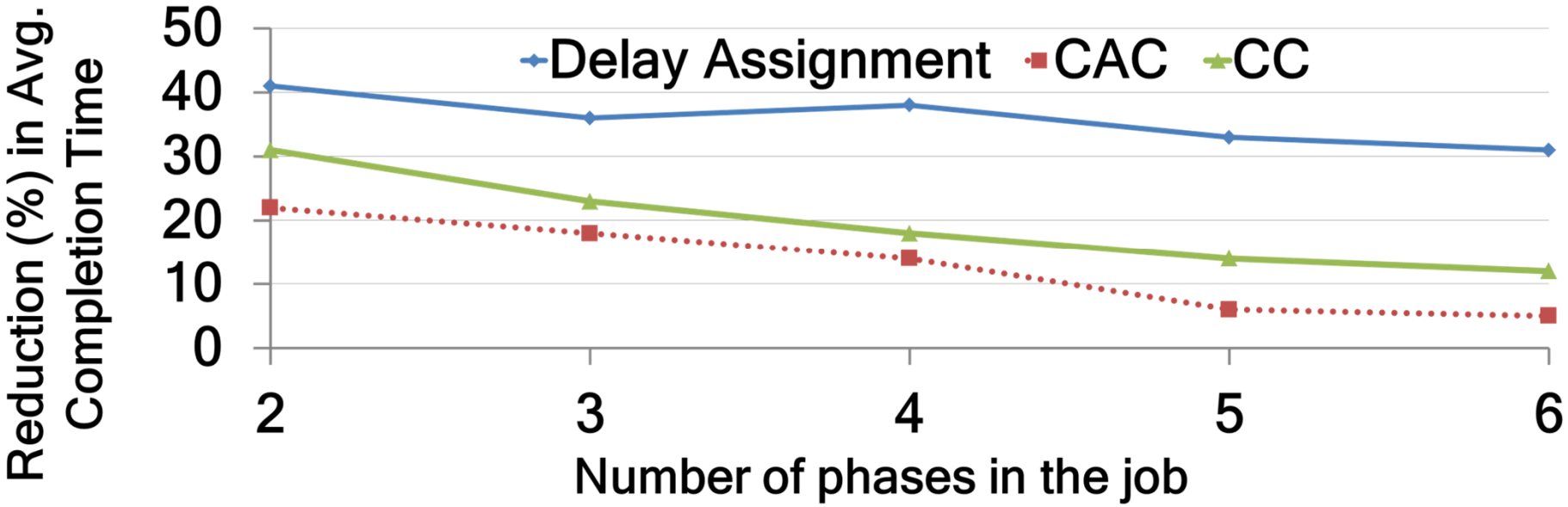
Effective Mitigation: Slowest task is 1.06x slower (down from 8x)

Delay Assignment is crucial



Impact on #phases in job?

- *Job DAGs can have many (> 2) phases*



Growing gap w.r.t. CAC and CC

Conclusions

- Traditional straggler mitigation techniques ill-suited for small interactive jobs

- **Dolly**: Proactive **Cloning** of jobs
  - **Heavy-tail** → Small **cloning budget (5%)** suffices
  - Effective Mitigation: eliminates *nearly* all stragglers

- **Power-law + Latency-sensitivity → Cloning**