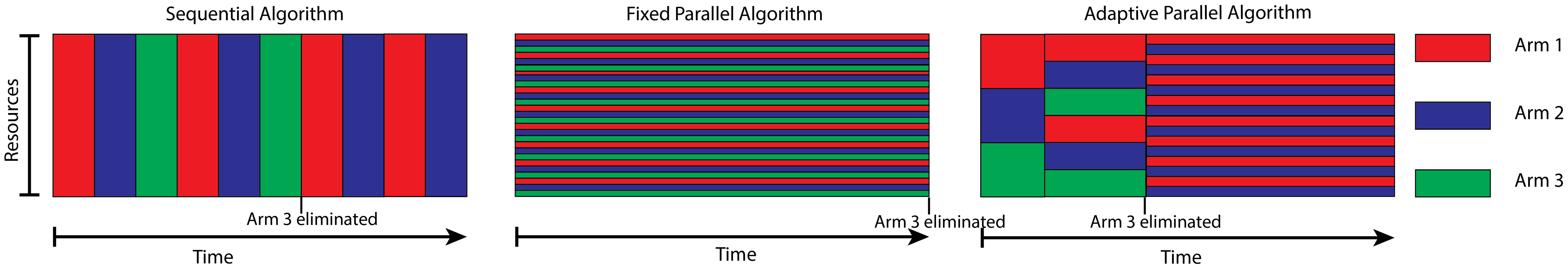
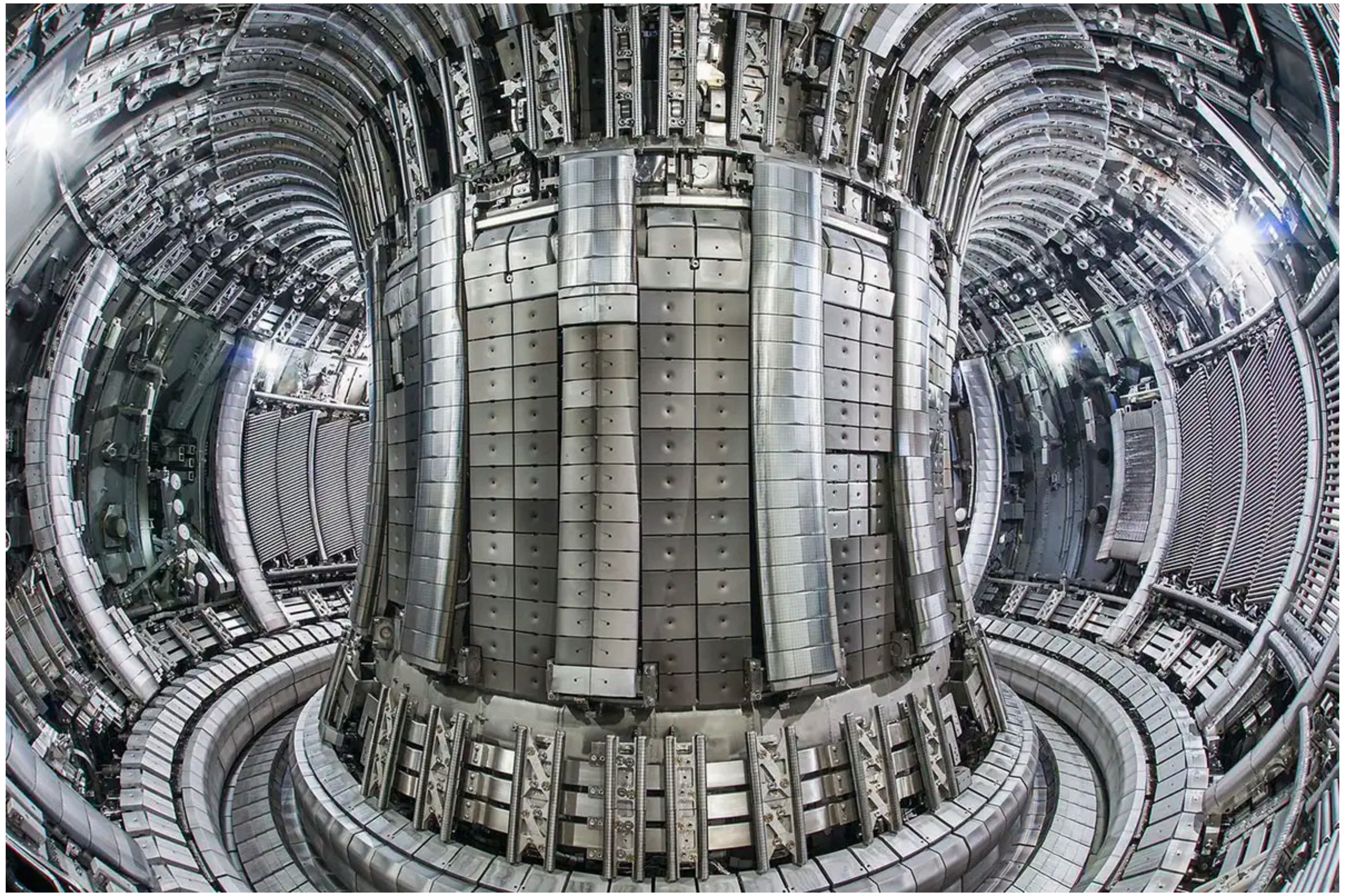
Resource Allocation in Multi-armed Bandit Exploration: Overcoming Nonlinear Scaling with Adaptive Parallelism



Brijen Thananjeyan, Kirthevasan Kandasamy, Ion Stoica, Michael I. Jordan, Ken Goldberg, Joseph E. Gonzalez

Motivating Example: Nuclear Fusion Simulation

- Suppose we have $n = 4$ parameters and we wish to identify the best one.



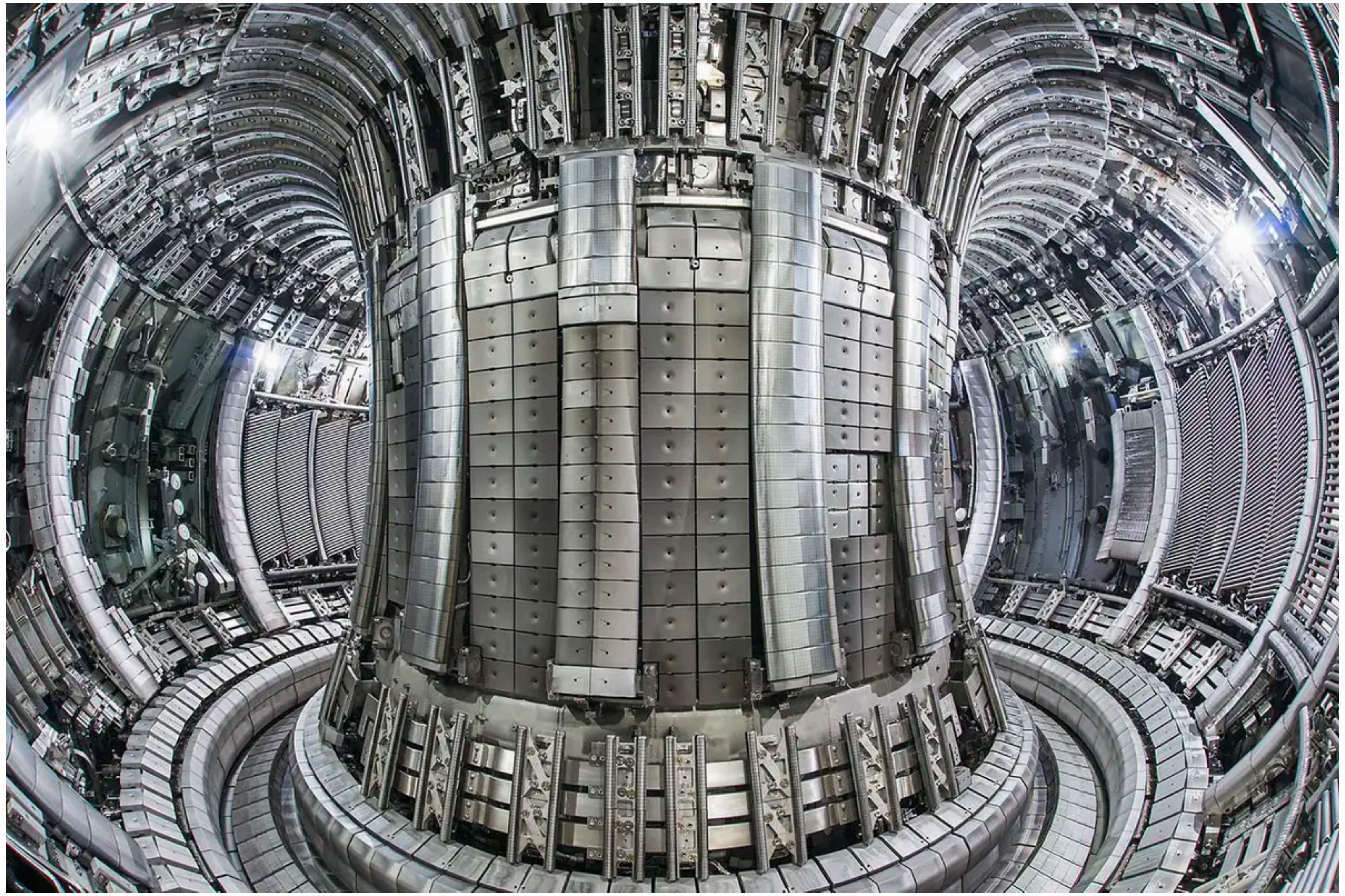
Possible parameters:

Parameter 1
Parameter 2
Parameter 3
Parameter 4

Goal: identify best one

Motivating Example: Nuclear Fusion Simulation

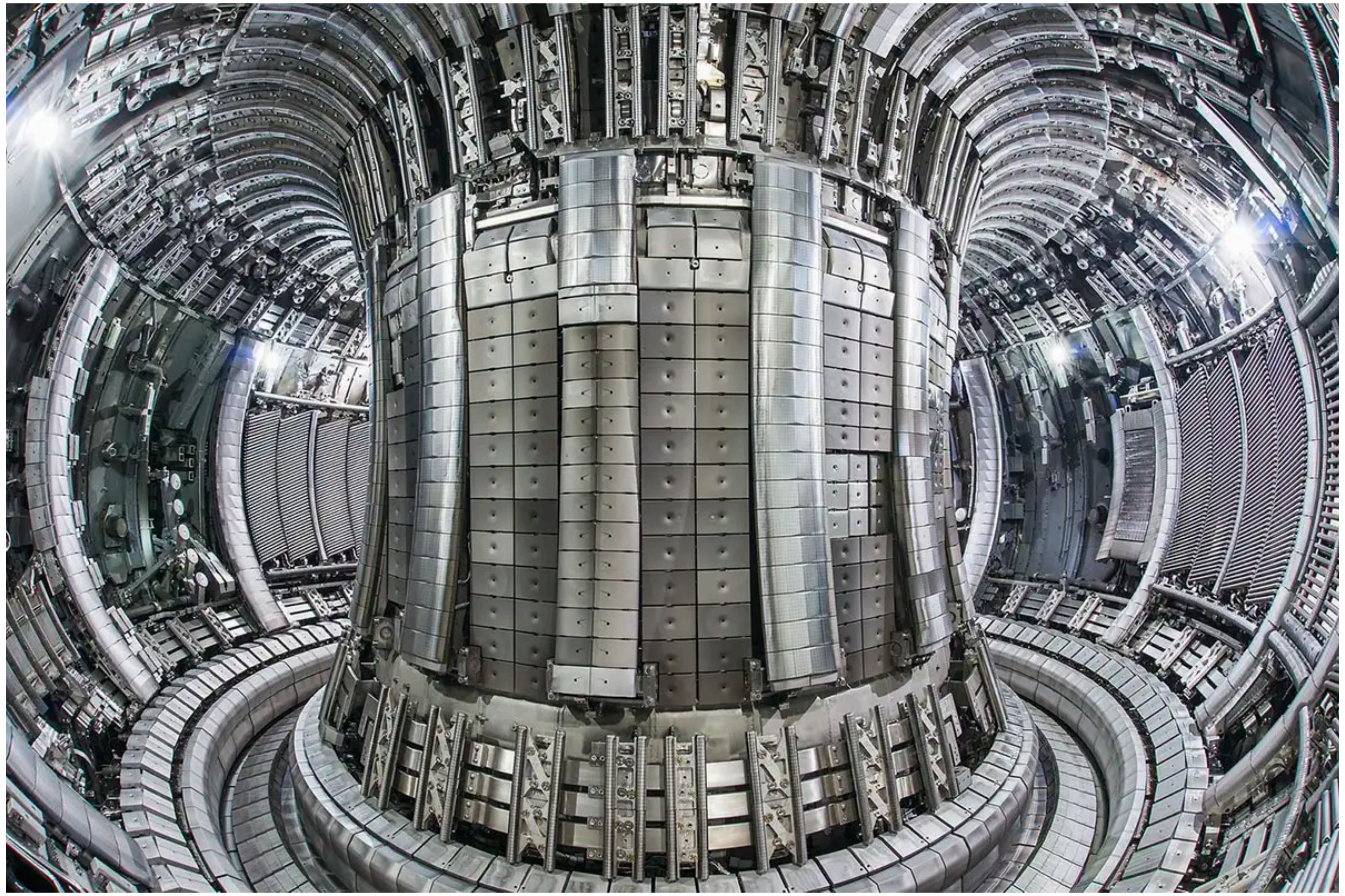
- Could try each parameters once, but simulation is stochastic.
    - Must try repeatedly to be sure we found the best parameters.



Motivating Example: Nuclear Fusion Simulation

We could:

- Try all parameters 100 times and pick the one that is best on average.
- Try all 10 times, pick best 2, try these ones 90 times, then pick the best one.



Motivating Example: Nuclear Fusion Simulation

Suppose we have a set of resources to run simulations.

What is the best way to allocate resources to simulations?

6 GPUs

Motivating Example: Nuclear Fusion Simulation

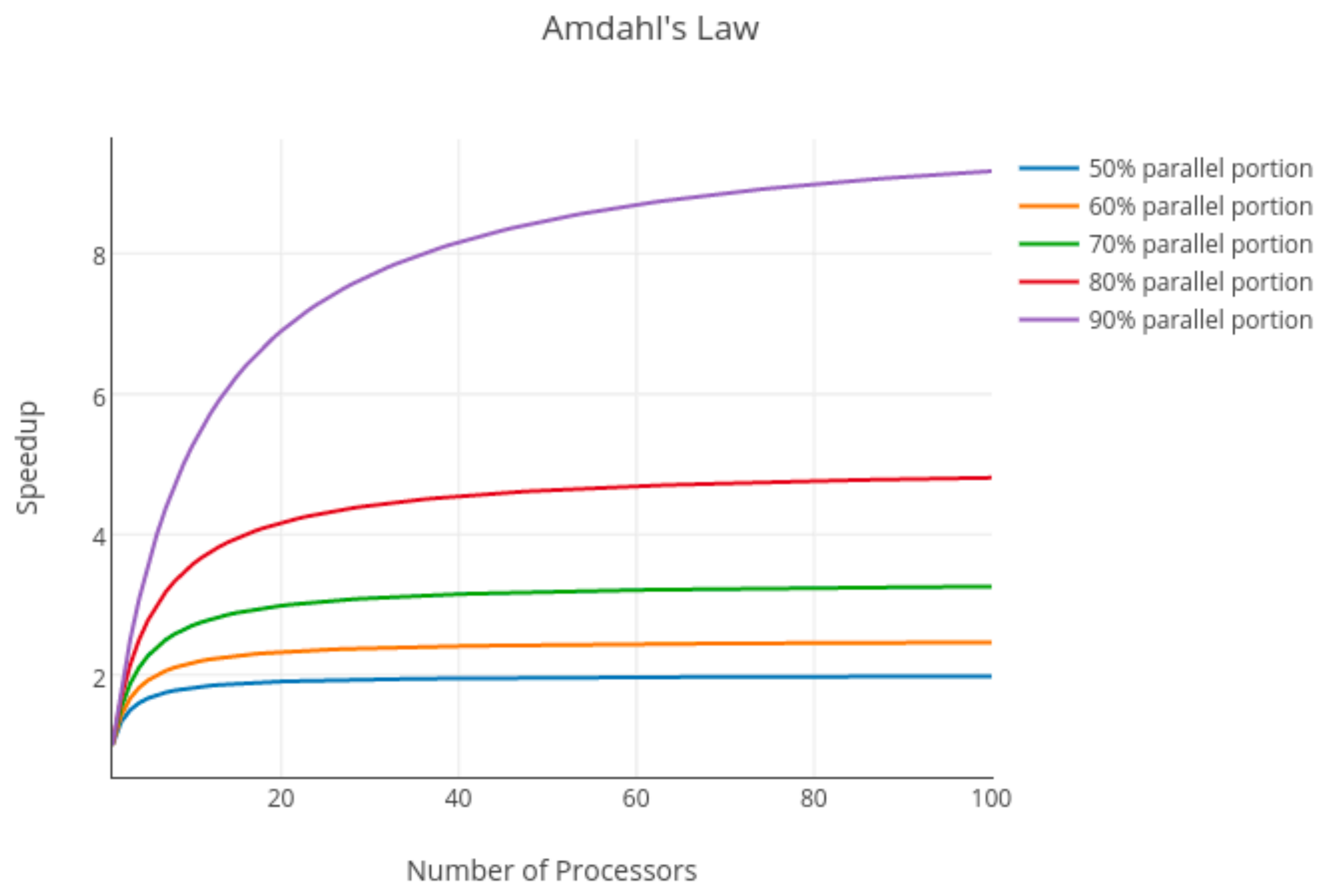
Assigning 1 GPU to a simulation will cause it to take 6 hours.

Assigning 6 GPUs to a simulation will cause it to take 2 hours.

6 GPUs

Motivating Example: Nuclear Fusion Simulation

- Algorithms must consider
  - Resources available
  - Scaling of program vs. resources used
    - Typically sublinear due to communication, synchronization, serial components
  - Tradeoff: information accumulation vs. throughput

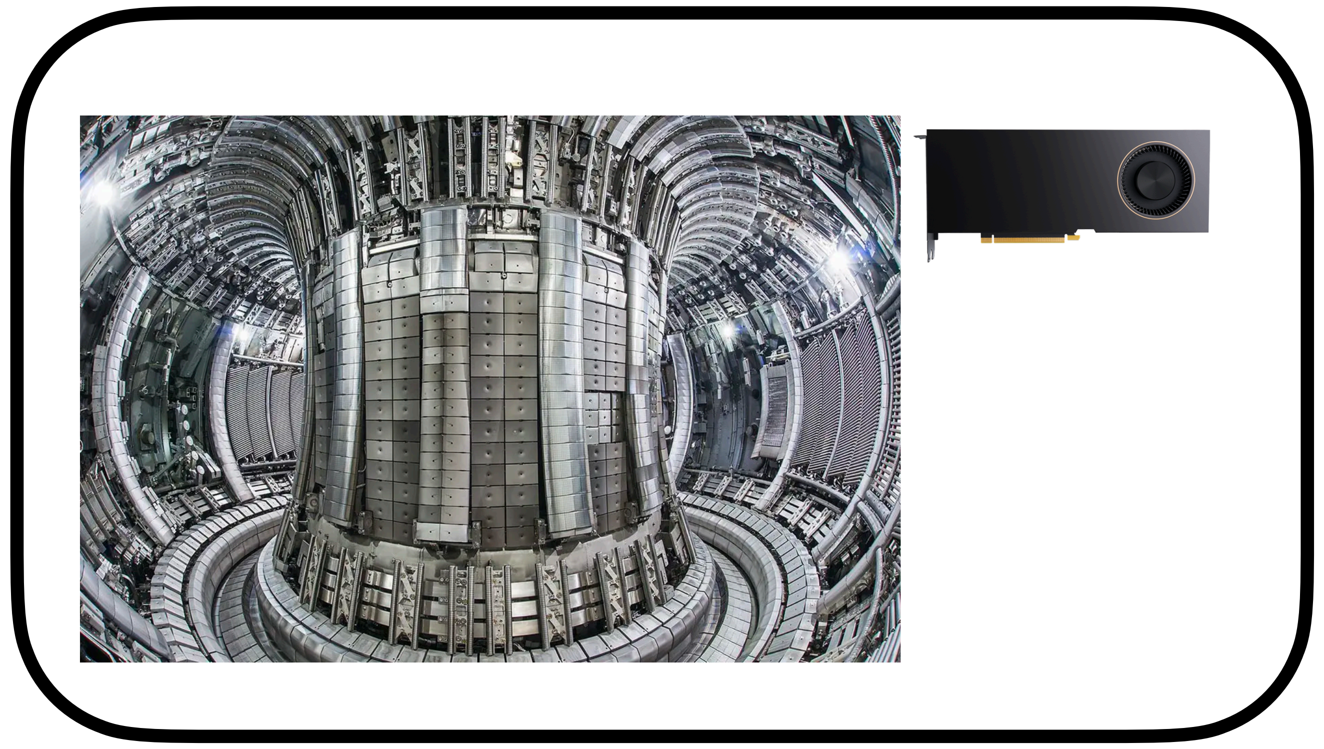


How do we best allocate resources?

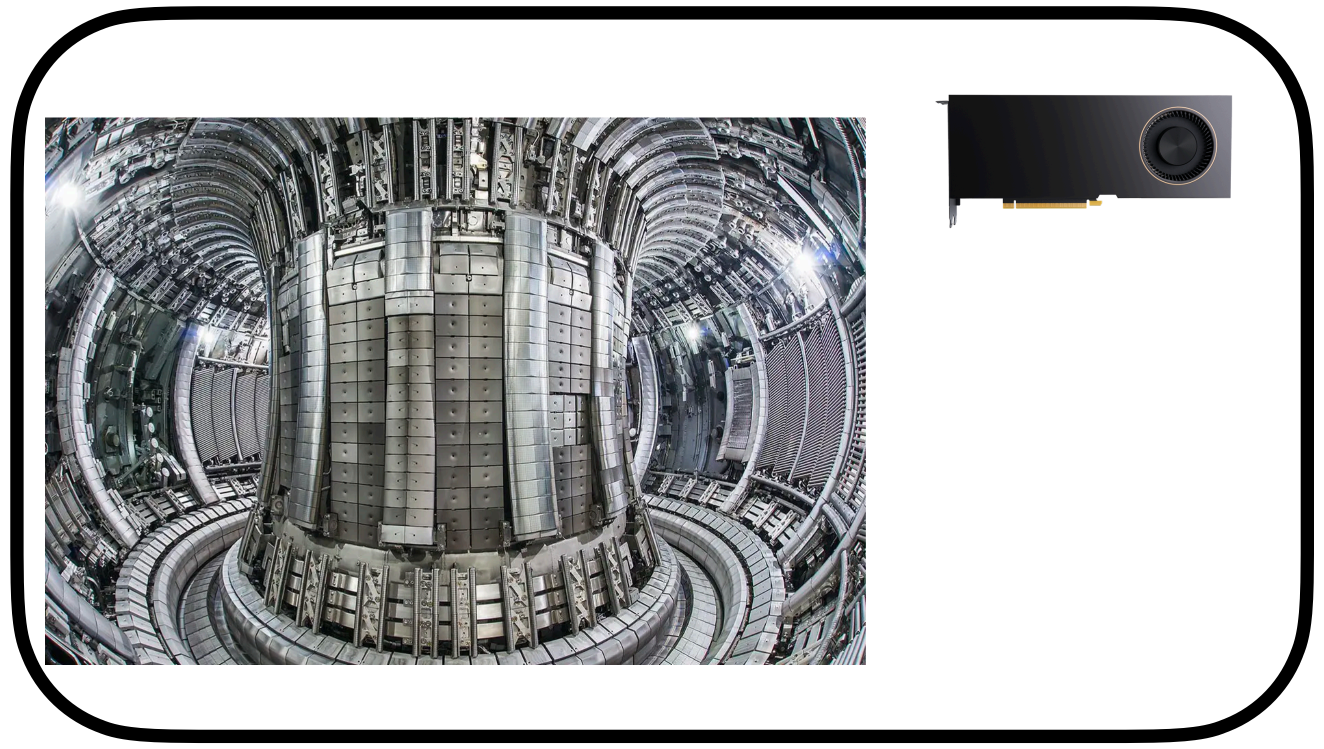
Assign a single GPU to each simulation. We can run 6 simulations at a time.

Suppose with 1 GPU, a simulation takes 6 hours.

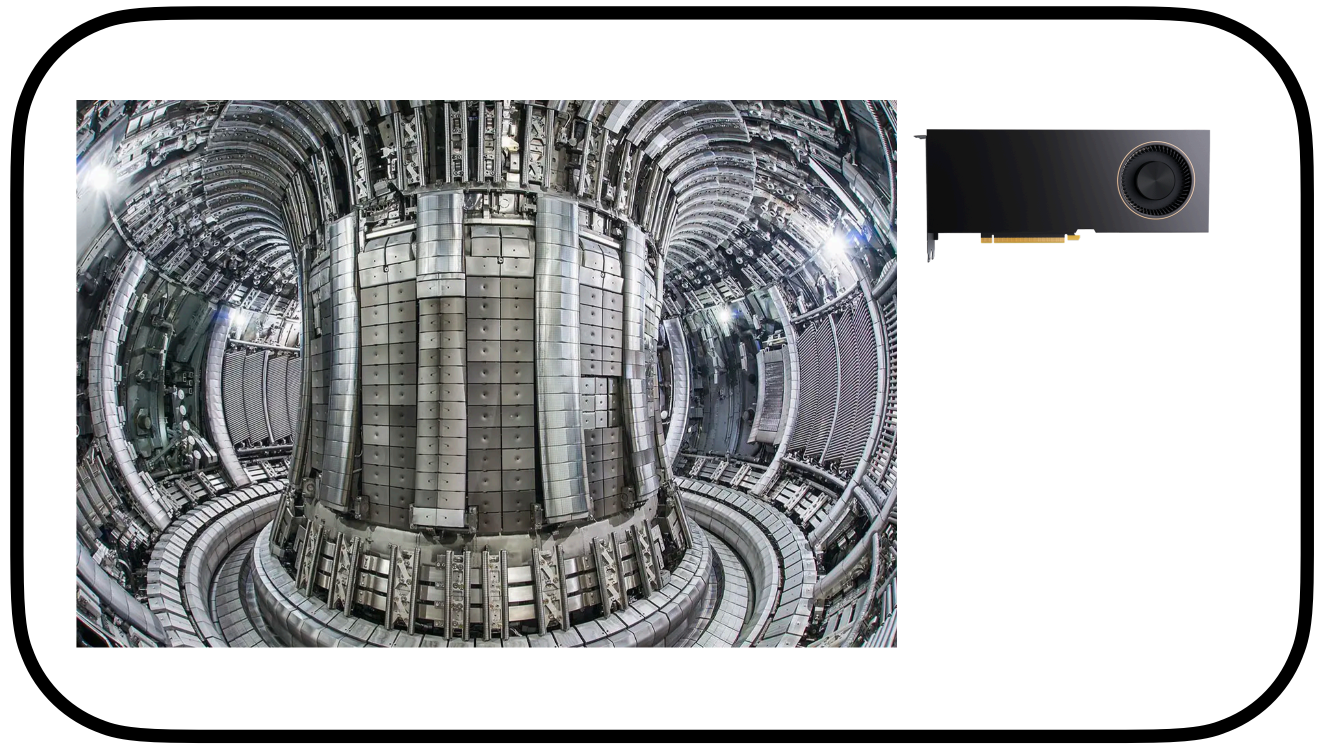
Parameter 1



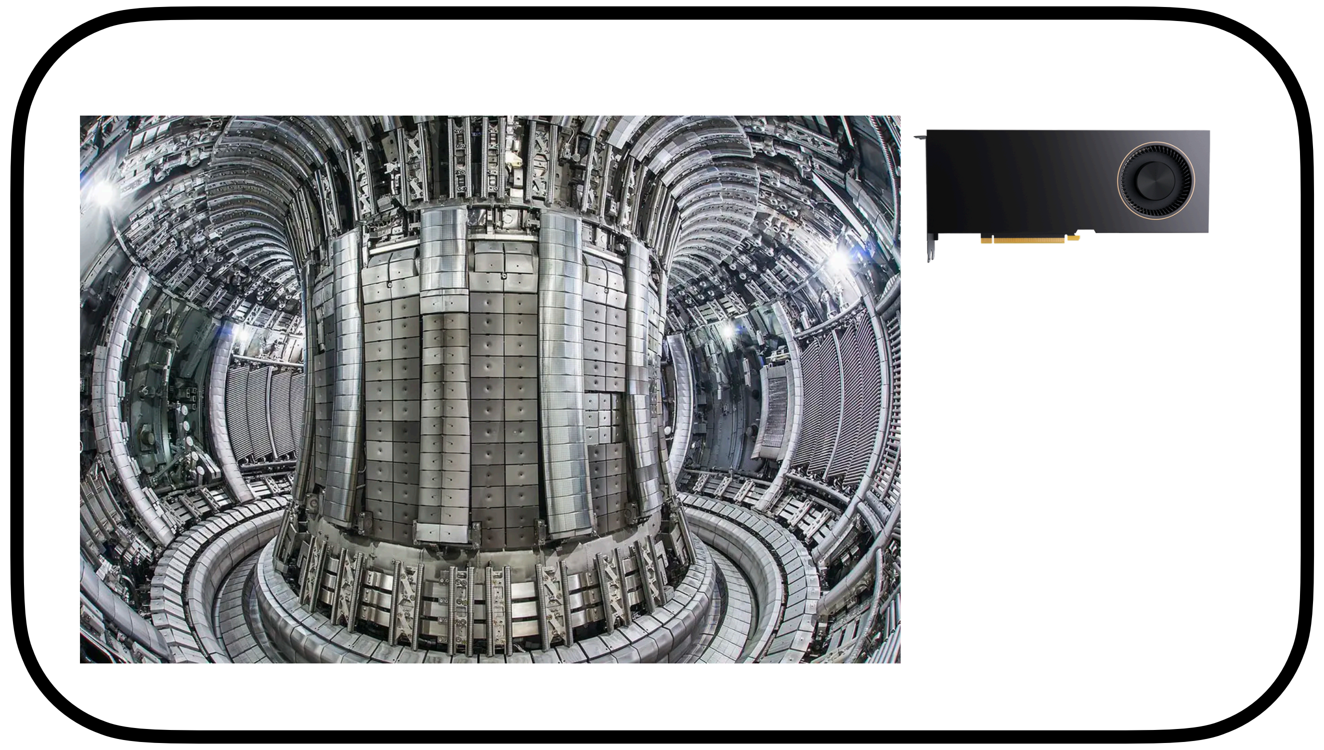
Parameter 2



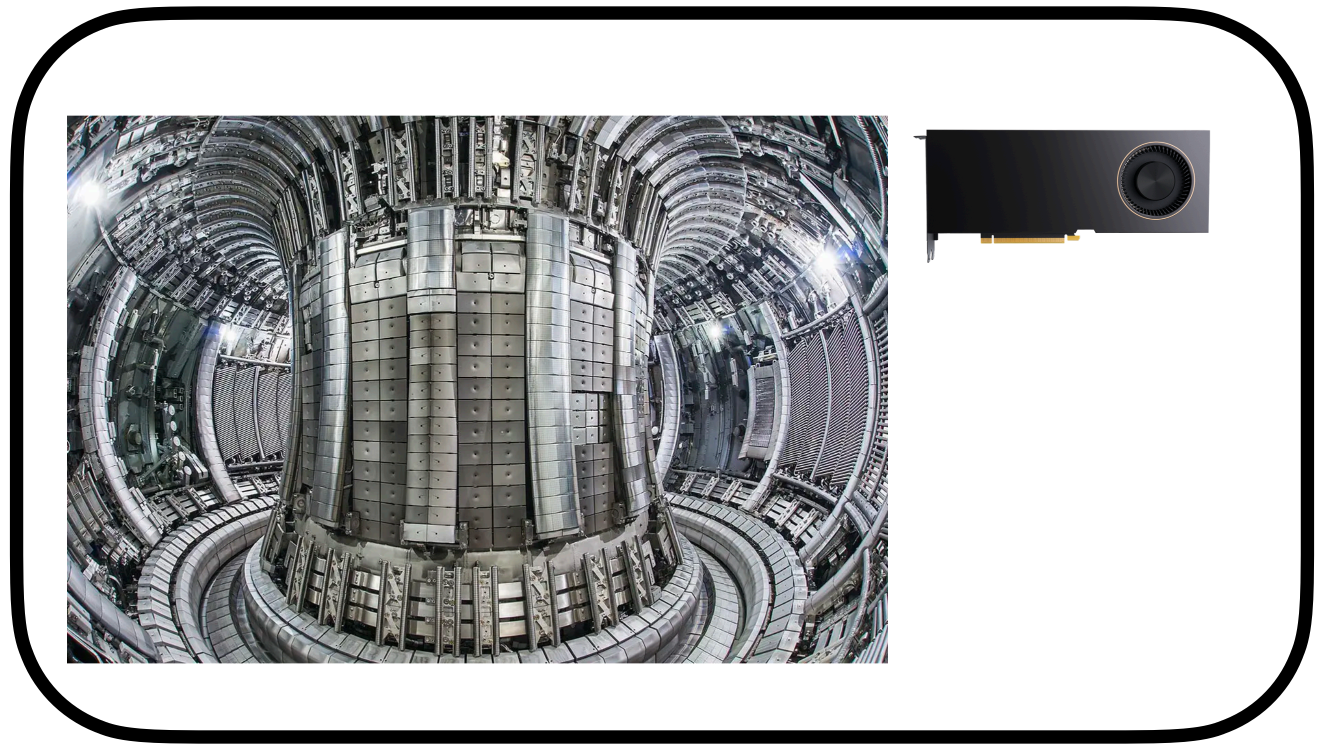
Parameter 4



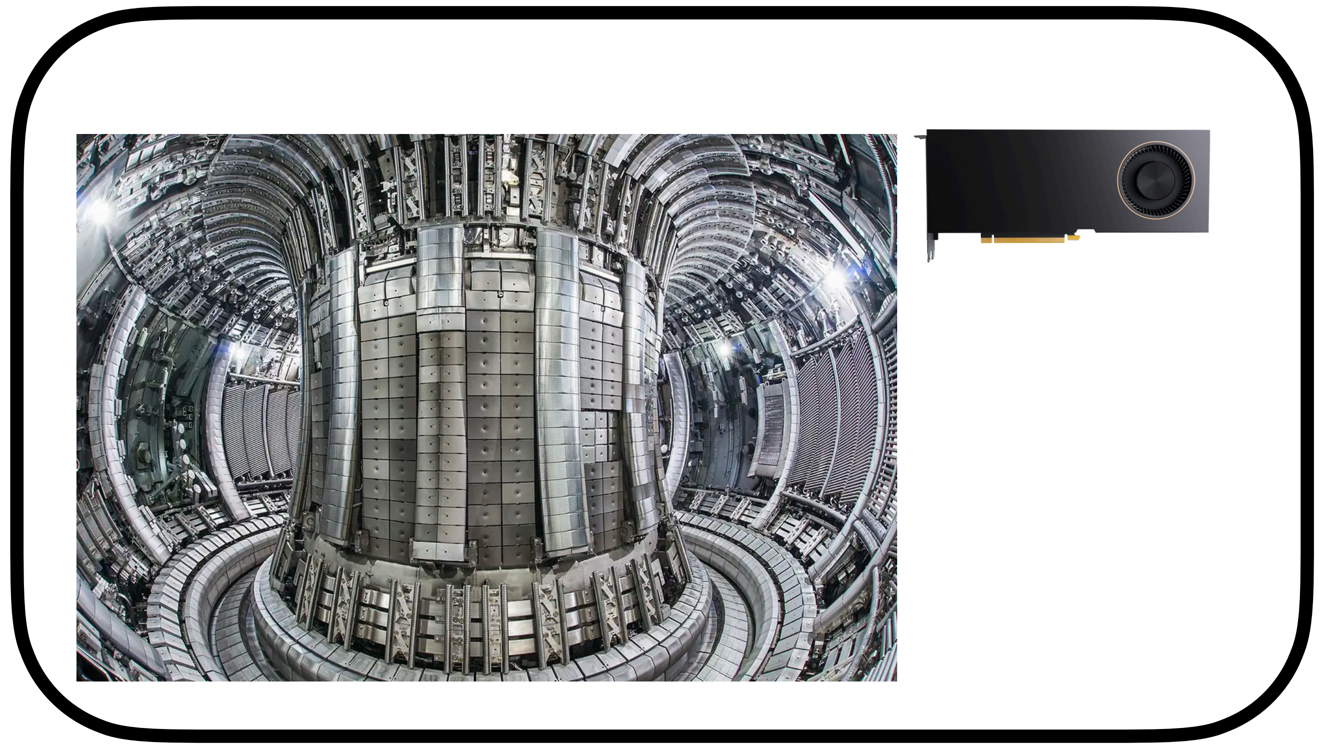
Parameter 1



Parameter 3



Parameter 4

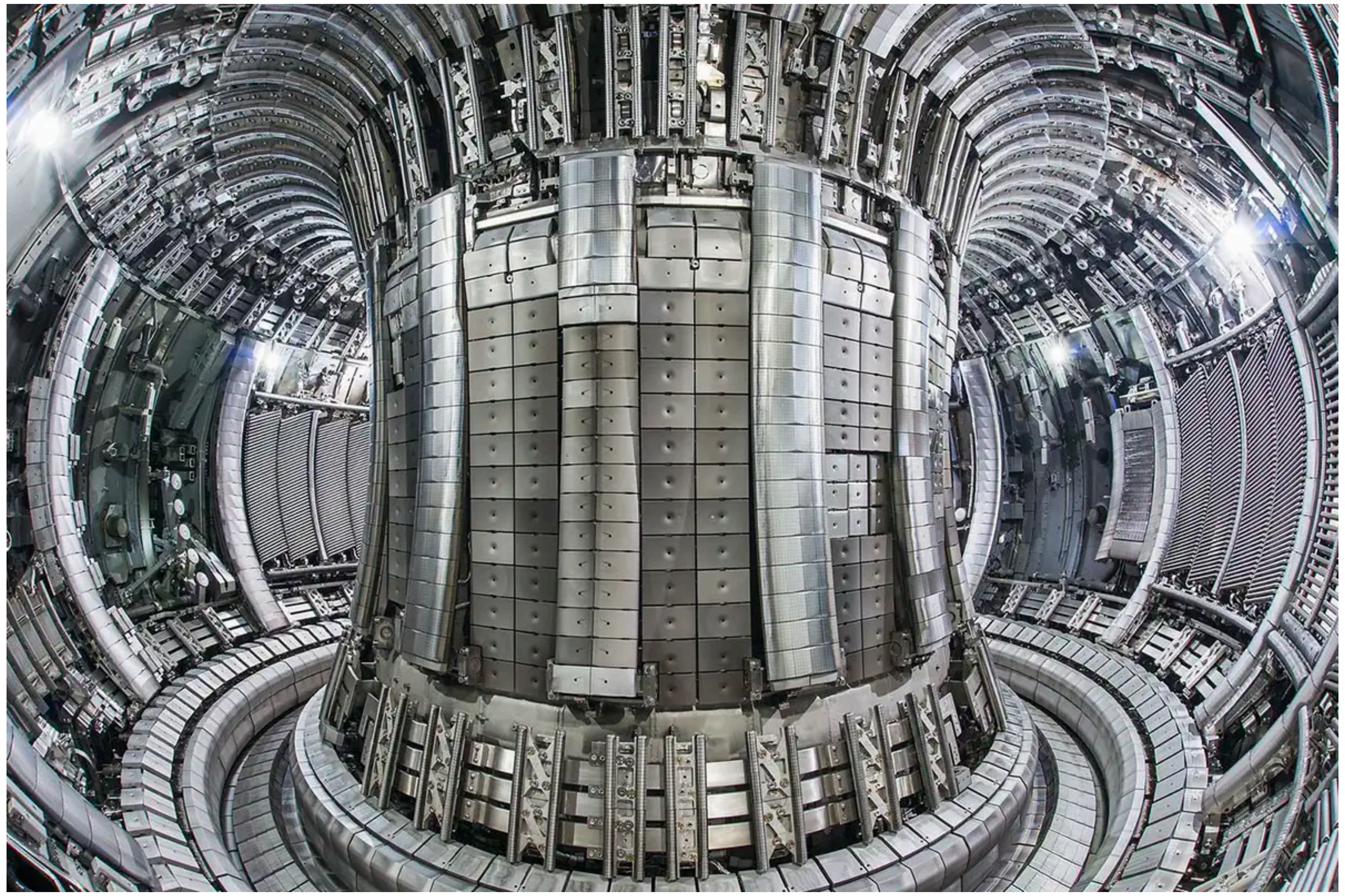


How do we best allocate resources?

Assign all 6 GPUs to each simulation. We can run 1 simulations at a time.

Suppose with 6 GPUs, a simulation takes 2 hours to finish. (Not 1/6th of before).

Parameter 1

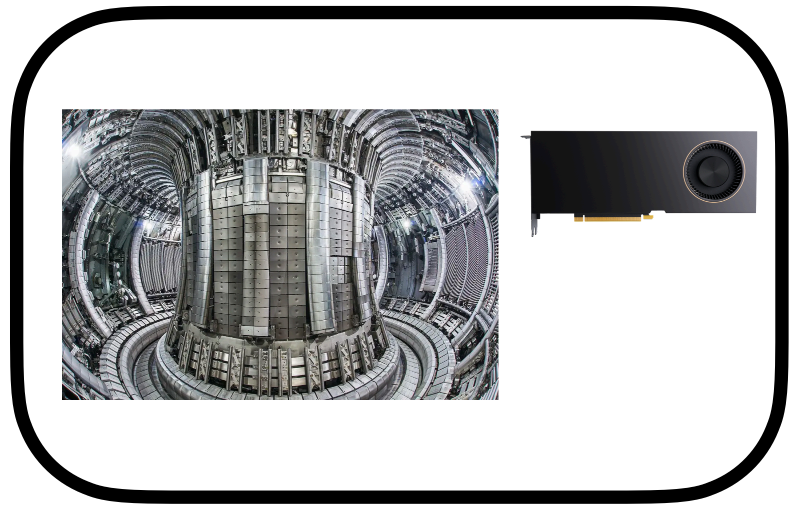


How do we best allocate resources?

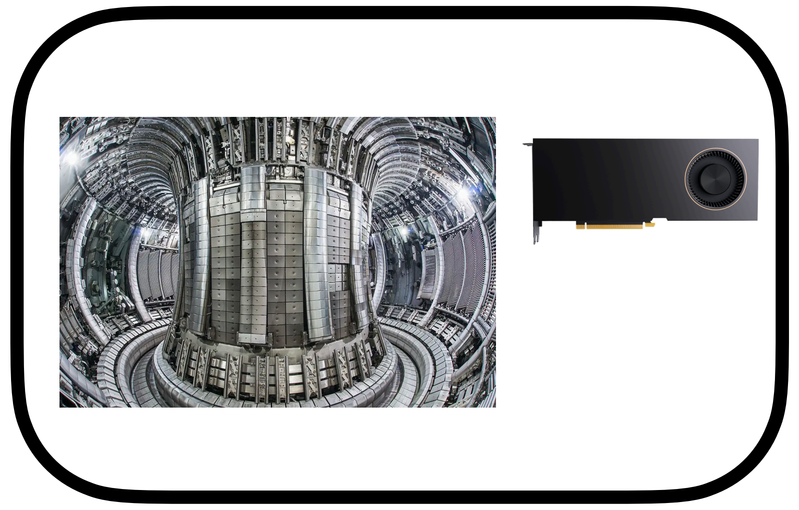
Completely parallel:

- 6 hours/batch
- 1 simulations/hour

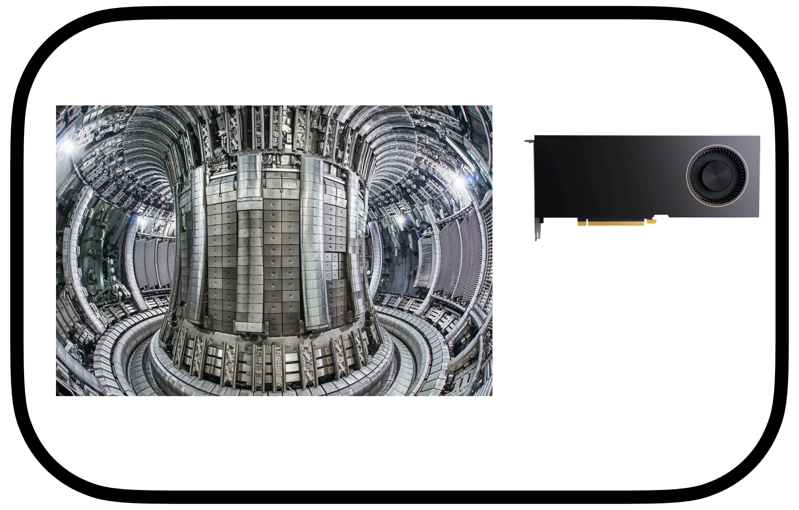
Parameter 1



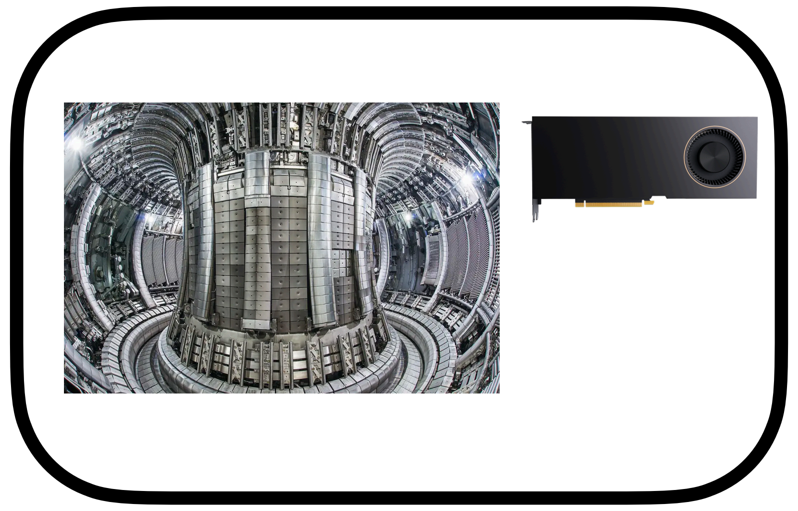
Parameter 2



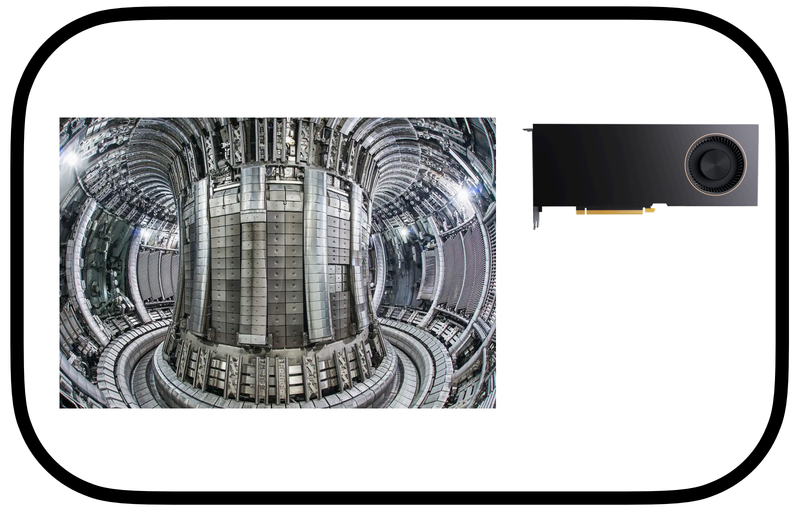
Parameter 4



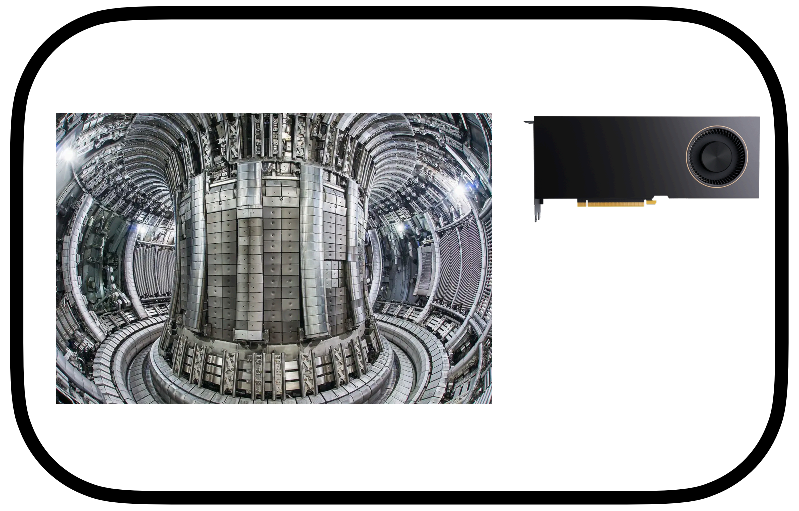
Parameter 1



Parameter 3



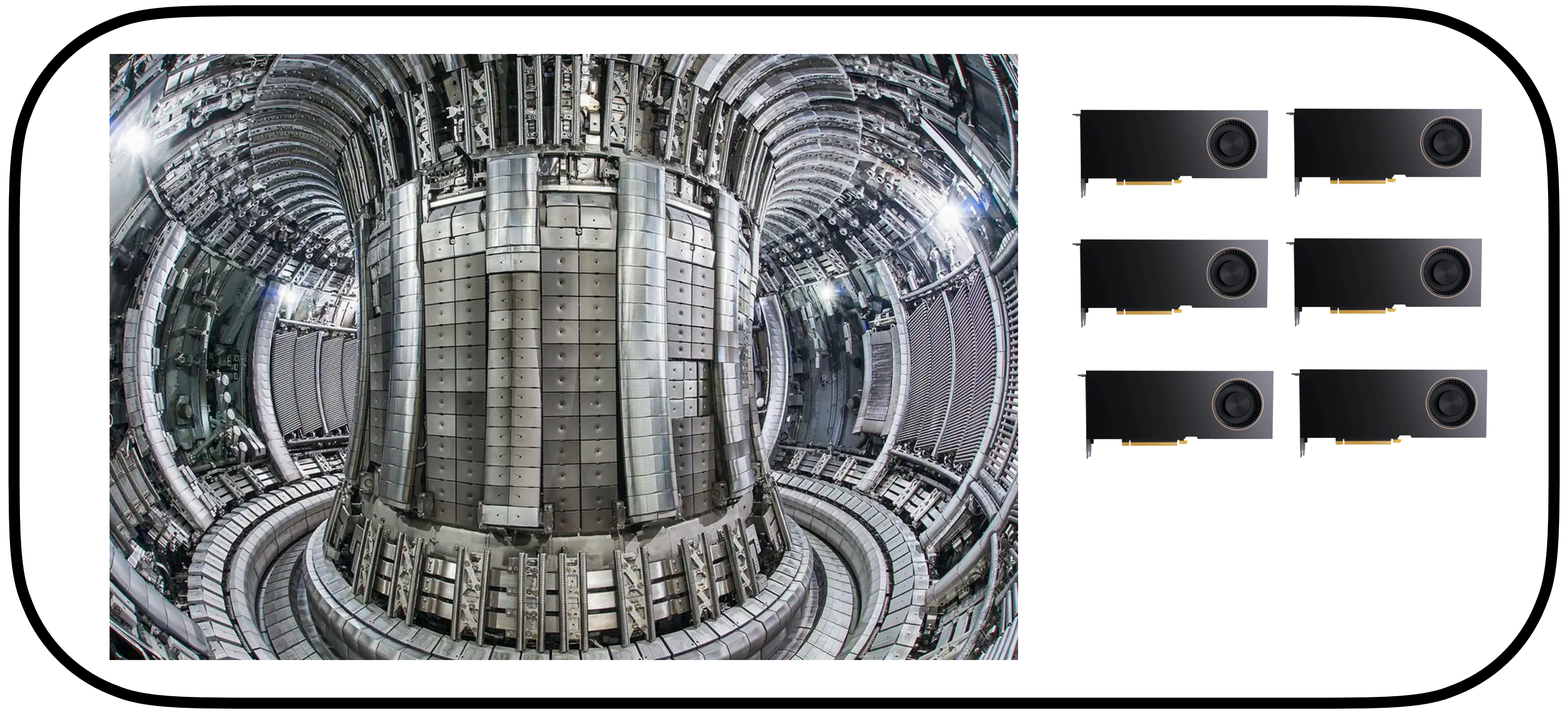
Parameter 4



Completely sequential:

- 2 hours/batch
- 0.5 simulations/hour

Parameter 1



How do we best allocate resources?

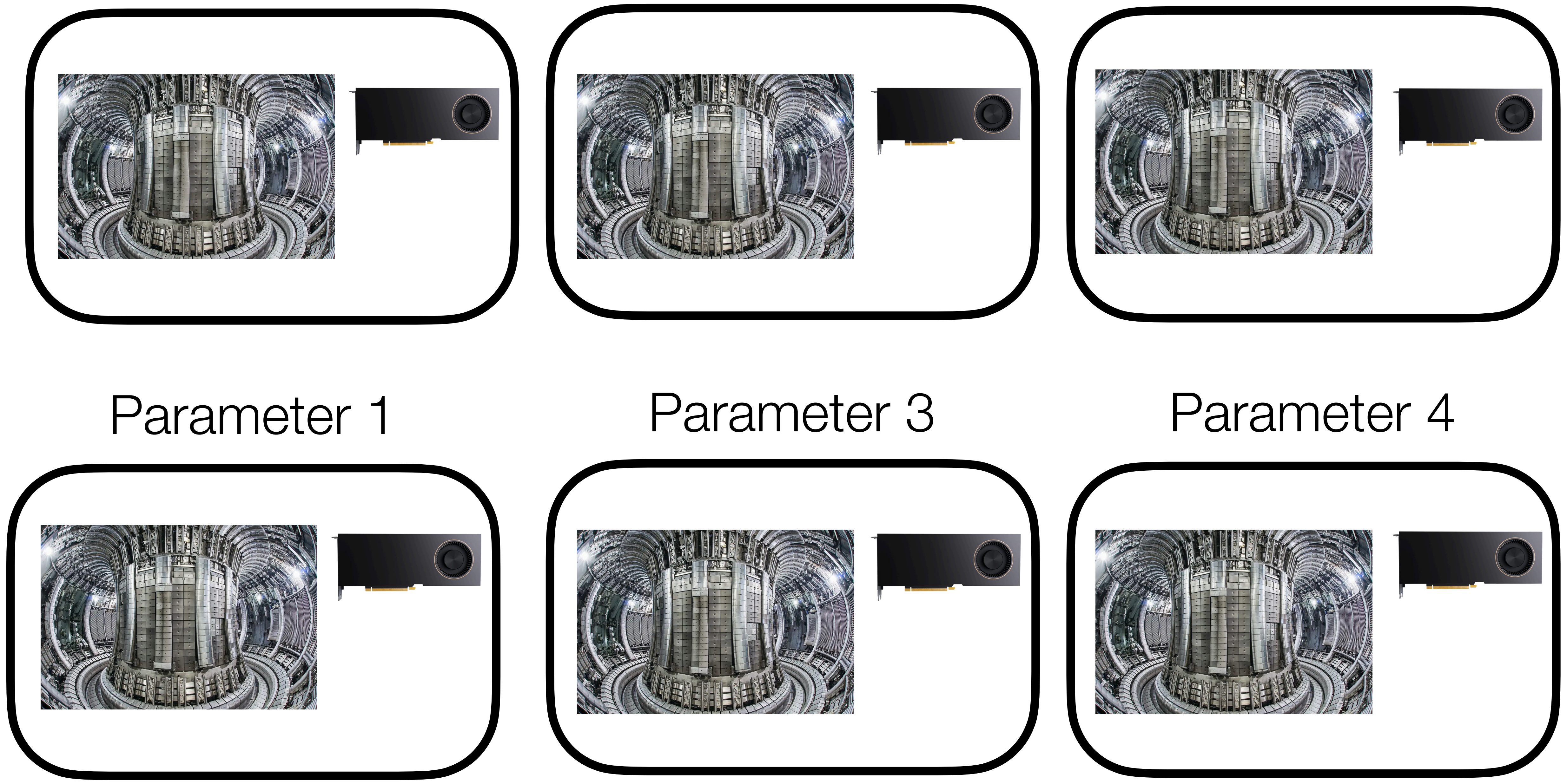
Completely parallel:

- 6 hours/batch
- 1 simulations/hour

More runs,
higher throughput

Parameter 1 | Parameter 2 | Parameter 4

Parameter 1 | Parameter 3 | Parameter 4

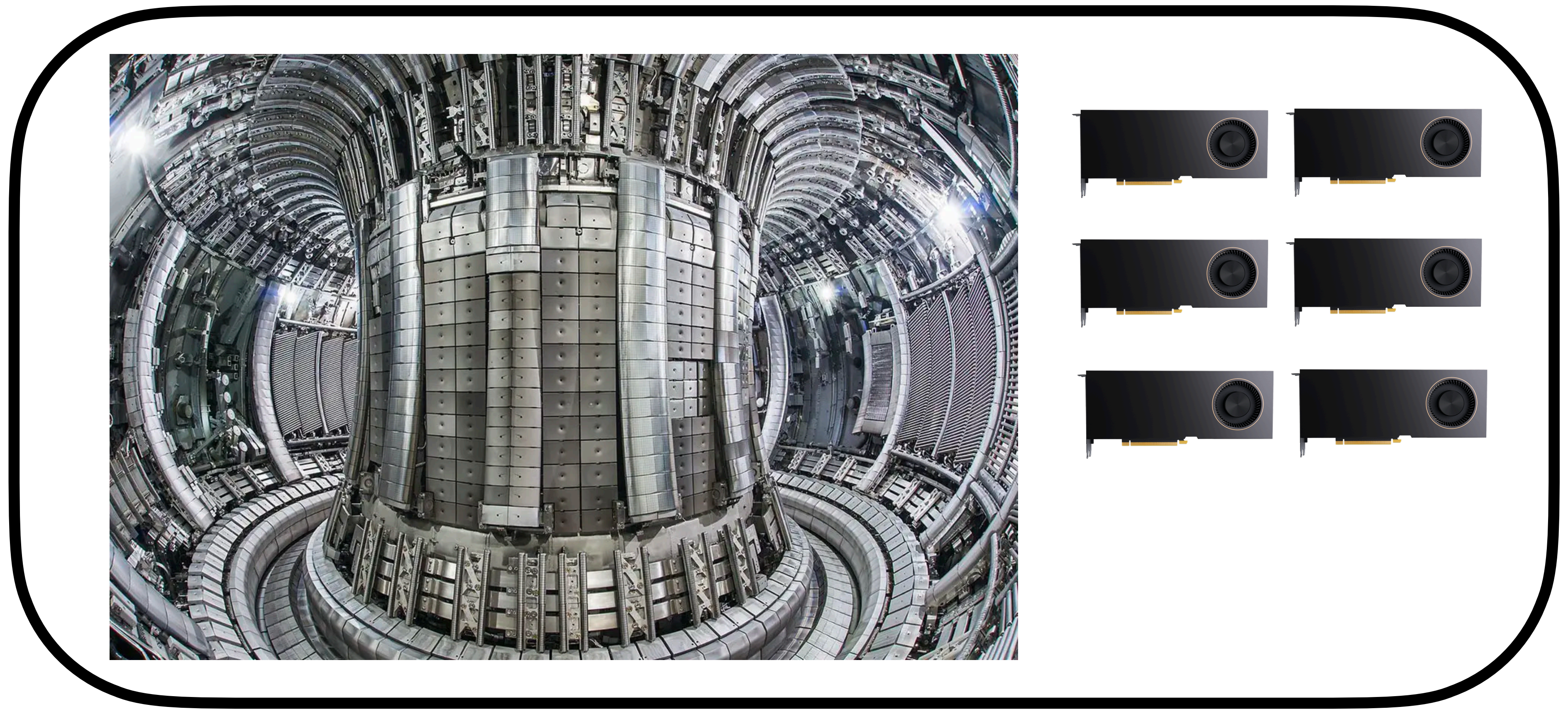


Completely sequential:

- 2 hours/batch
- 0.5 simulations/hour

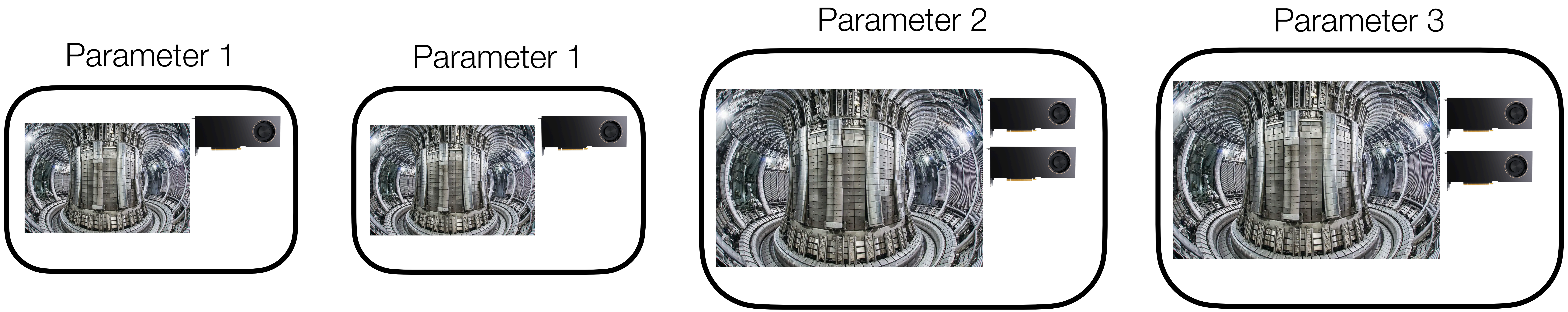
More resources/run,
faster results

Parameter 1



How do we best allocate resources?

Maybe something else?



We will model this as a novel bandit exploration problem.

Real World Problem	Bandit Problem
Simulation parameter	Arm
Simulation, job, run	Arm pull
GPUs, cores, instances, nodes	Resources

This paper contributes:

- Novel setting for best arm identification in multi-armed bandits with time and resource allocation
- A δ-PAC algorithm for the fixed confidence setting
  - Upper bound on runtime
  - Matching lower bound
  - Synthetic experiments

Covered in this talk.

- An algorithm in the fixed deadline setting
  - Upper bound on error probability
  - Synthetic experiments

Covered in the paper, but not in this talk.

Motivation

Problem Setup

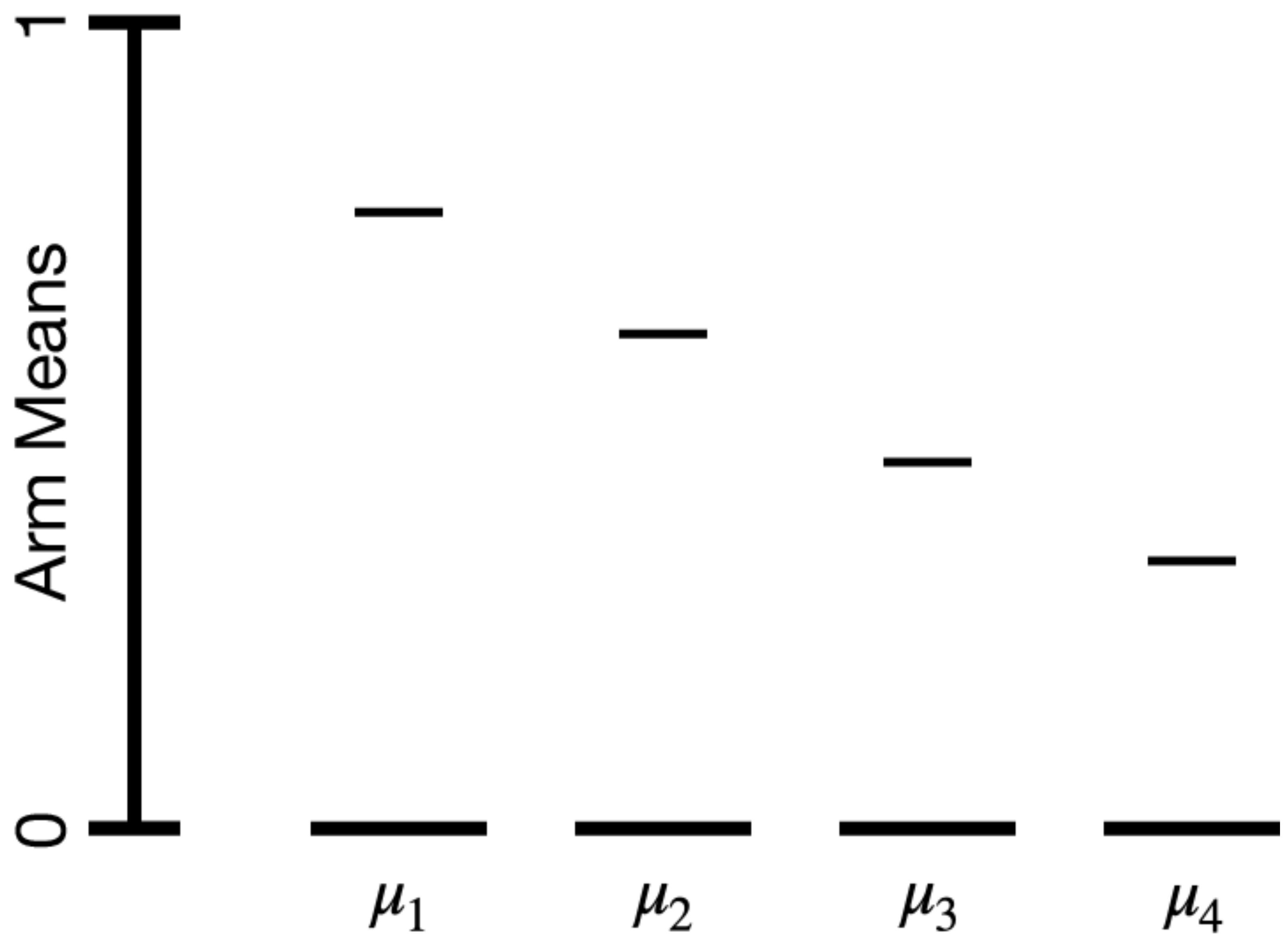
Fixed Confidence Setting: Results

Best Arm Identification (BAI): Prior Work

- Sequential BAI:
    - Karnin, Zohar, Tomer Koren, and Oren Somekh. "Almost optimal exploration in multi-armed bandits." In *International Conference on Machine Learning*, pp. 1238-1246. 2013.
    - Kaufmann, Emilie, Olivier Cappé, and Aurélien Garivier. "On the complexity of best-arm identification in multi-armed bandit models." *The Journal of Machine Learning Research* 17, no. 1 (2016): 1-42.Parallel setting:
- Parallel BAI:
    - Jun, Kwang-Sung, Kevin G. Jamieson, Robert D. Nowak, and Xiaojin Zhu. “Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls.” In *AISTATS* pp.139-148, 2016.
    - Grover, Aditya, Todor Markov, Peter Attia, Norman Jin, Nicholas Perkins, Bryan Cheong, Michael Chen, Zi Yang, Stephen Harris, William Chueh, Stefano Ermon. Best arm identification in multi-armed bandits with Delayed Feedback. In *AISTATS*, pp. 833-842. PLMR, 2018.
- **This paper:**
    - Augment prior settings by adding time and resource allocation to BAI

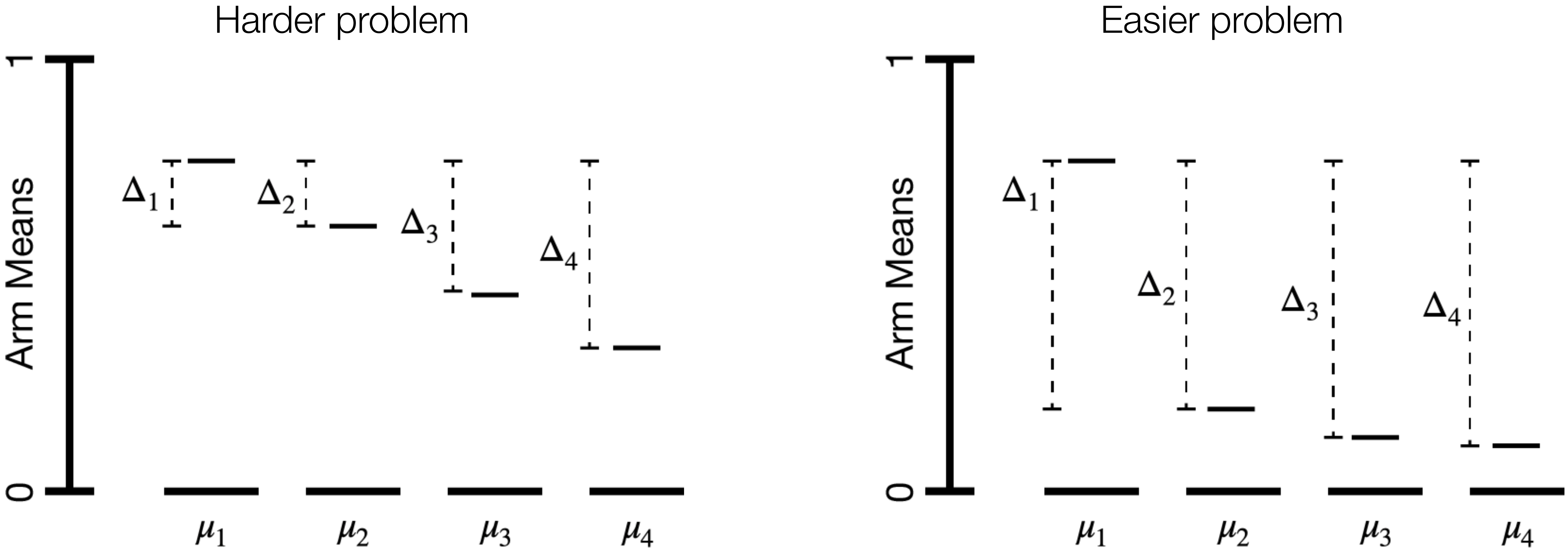
Best Arm Identification

- n arms
  - Samples independent, bounded in [0, 1]
  - Arm i has mean μ_i
    - WLOG: assume $\mu_1 > \mu_2 \geq \cdots \geq \mu_n$
- **Goal: identify the arm with the highest mean**



Best Arm Identification

Define arm gap $\Delta_i = \mu_1 - \mu_i$ for $i > 1$, and $\Delta_1 = \mu_1 - \mu_2$



Best Arm Identification (BAI): Settings

Prior work

Fixed confidence setting:

- **Given:** error probability tolerance δ
- **Goal:** identify best arm within error tolerance, minimize number of pulls

Fixed budget setting:

- **Given:** budget B of arm pulls
- **Goal:** identify best arm within B pulls, minimize error probability δ

This paper

Fixed confidence setting:

- **Given:** error probability tolerance δ
- **Goal:** identify best arm within error tolerance, minimize **time**

Fixed deadline setting:

- **Given: time deadline T**
- **Goal:** identify best arm within **T time**, minimize error probability δ

Best Arm Identification (BAI): Settings

Prior work

Fixed confidence setting:

- **Given:** error probability tolerance δ
- **Goal:** identify best arm within error tolerance, minimize number of pulls

Fixed budget setting:

- **Given:** budget B of arm pulls
- **Goal:** identify best arm within B pulls, minimize error probability δ

This paper

Fixed confidence setting:

- **Given:** error probability tolerance δ
- **Goal:** identify best arm within error tolerance, minimize **time**

Fixed deadline setting:

- **Given: time deadline T**
- **Goal:** identify best arm within **T time**, minimize error probability δ

New Idea: allocate a fraction of resources to each pull.

λ: scaling function, **known,** indicates how **resource allocation affects pull time**

- *e.g. allocating 2 GPUs to a job vs. 1 causes it to run 1.5x faster*

How do we model time taken per pull?

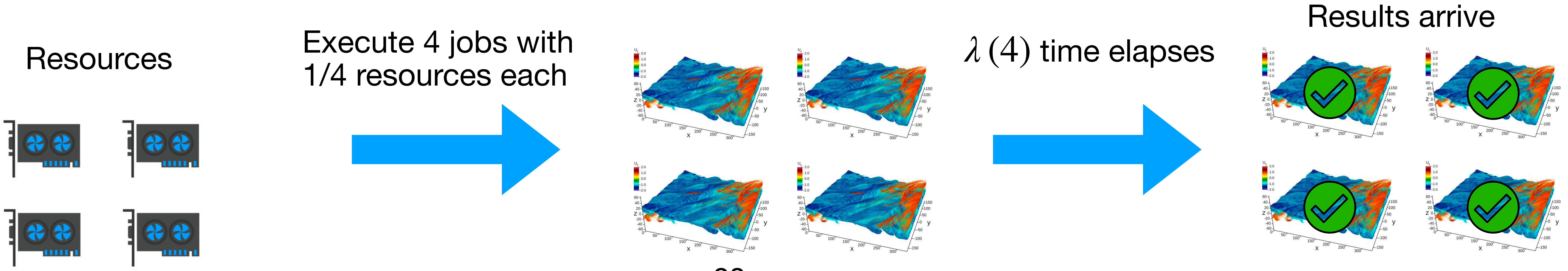
Should depend on the number of resources allocated to the pull.

Define scaling function λ

- If fraction $\alpha \in [0,1]$ of resources allocated to a pull, it takes $\lambda(1/\alpha)$ time.

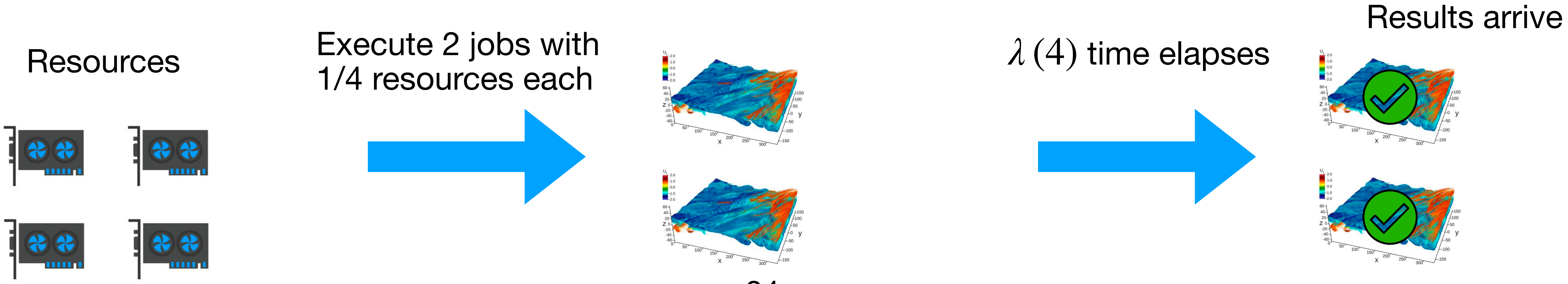
Scaling Function Properties

- If fraction $\alpha \in [0,1]$ of resources allocated to a pull, it takes $\lambda(1/\alpha)$ time.
- Suppose a batch of b pulls are executed in parallel with $1/b$ resources each.
  - Batch takes $\lambda(b)$ time to finish.



Scaling Function Properties

- If fraction $\alpha \in [0,1]$ of resources allocated to a pull, it takes $\lambda(1/\alpha)$ time.
- Suppose a batch of b pulls by dividing $\eta \in [0,1]$ resources evenly.
  - Batch takes $\lambda(b/\eta)$ time to finish.



Core Assumption on Scaling Function λ

Diminishing returns (sublinear scaling): allocating more resources doesn't proportionally speed up sampling time

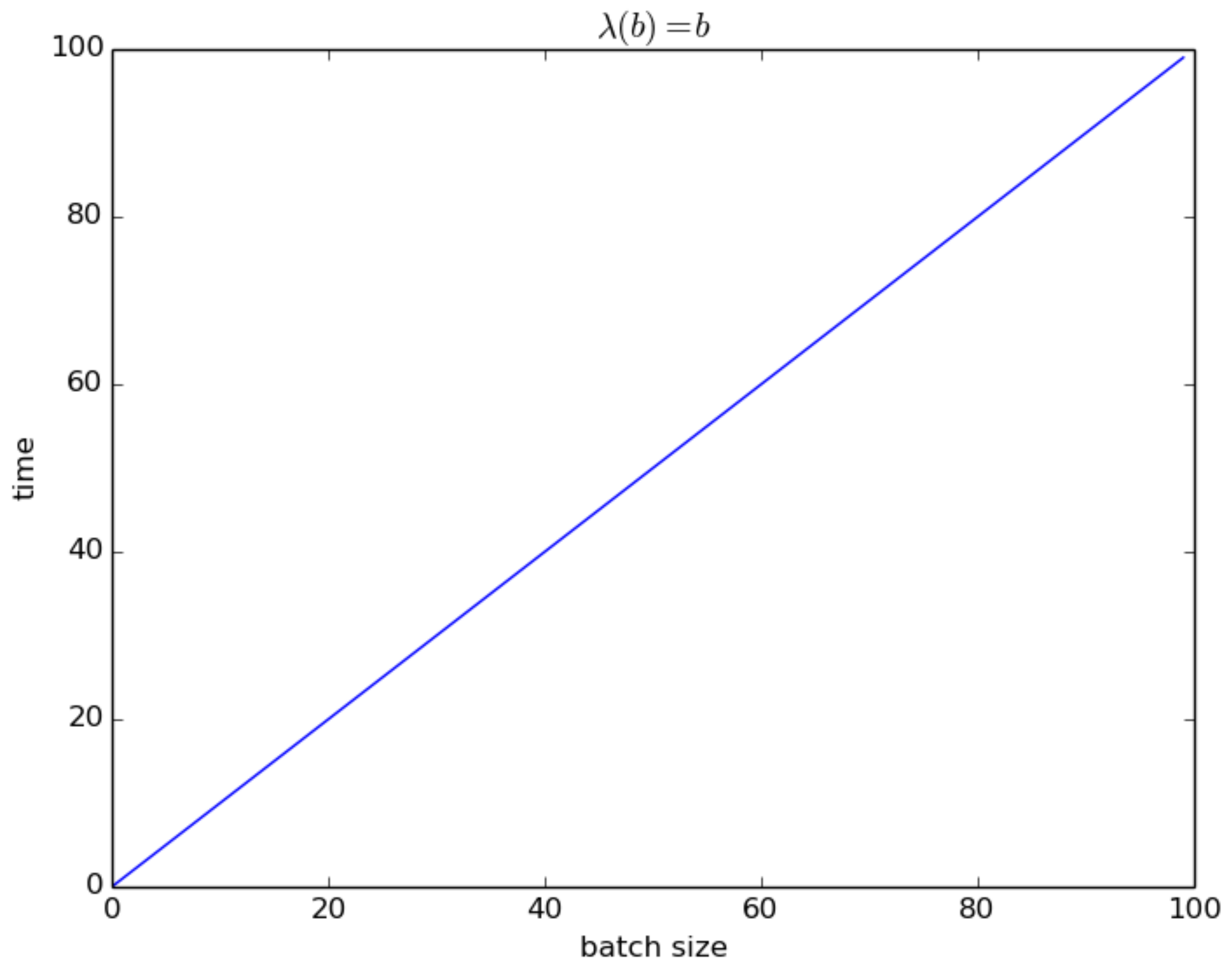
- *e.g. allocating 2 GPUs to a job vs. 1 causes it to run 1.5x faster not >2x*

Linear vs. Sublinear Scaling Function λ

$$\lambda(b) = b$$

A batch of 100 jobs takes 100 hours.

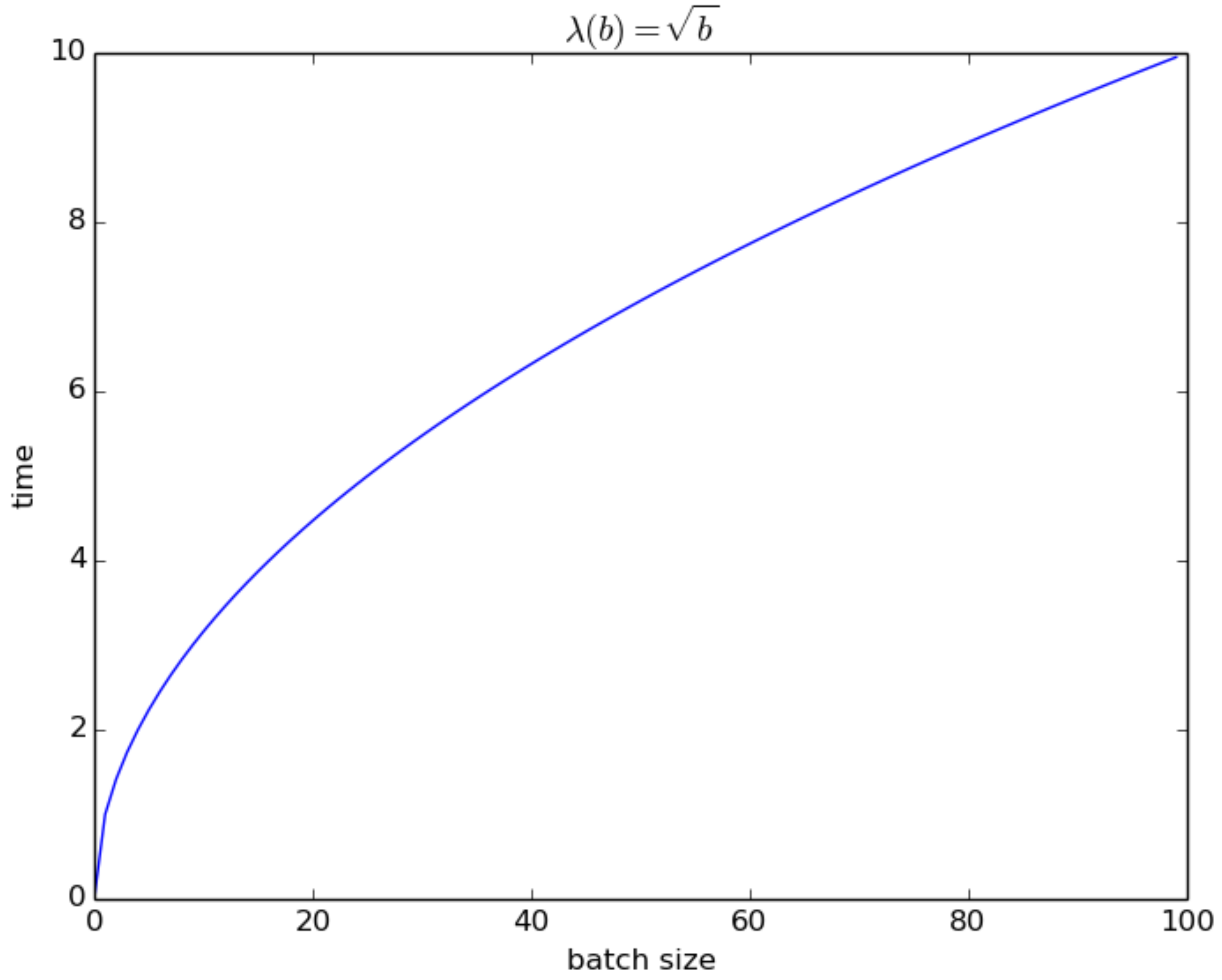
A batch of 1 job takes 1 hour. **(100x speedup)**



$$\lambda(b) = \sqrt{b}$$

A batch of 100 jobs takes 10 hours.

A batch of 1 job takes 1 hour. **(10x speedup)**



Implications

Need to trade off between information accumulation and throughput

Example:

- Let $\lambda(b) = \sqrt{b}$
- Pulling arms sequentially:

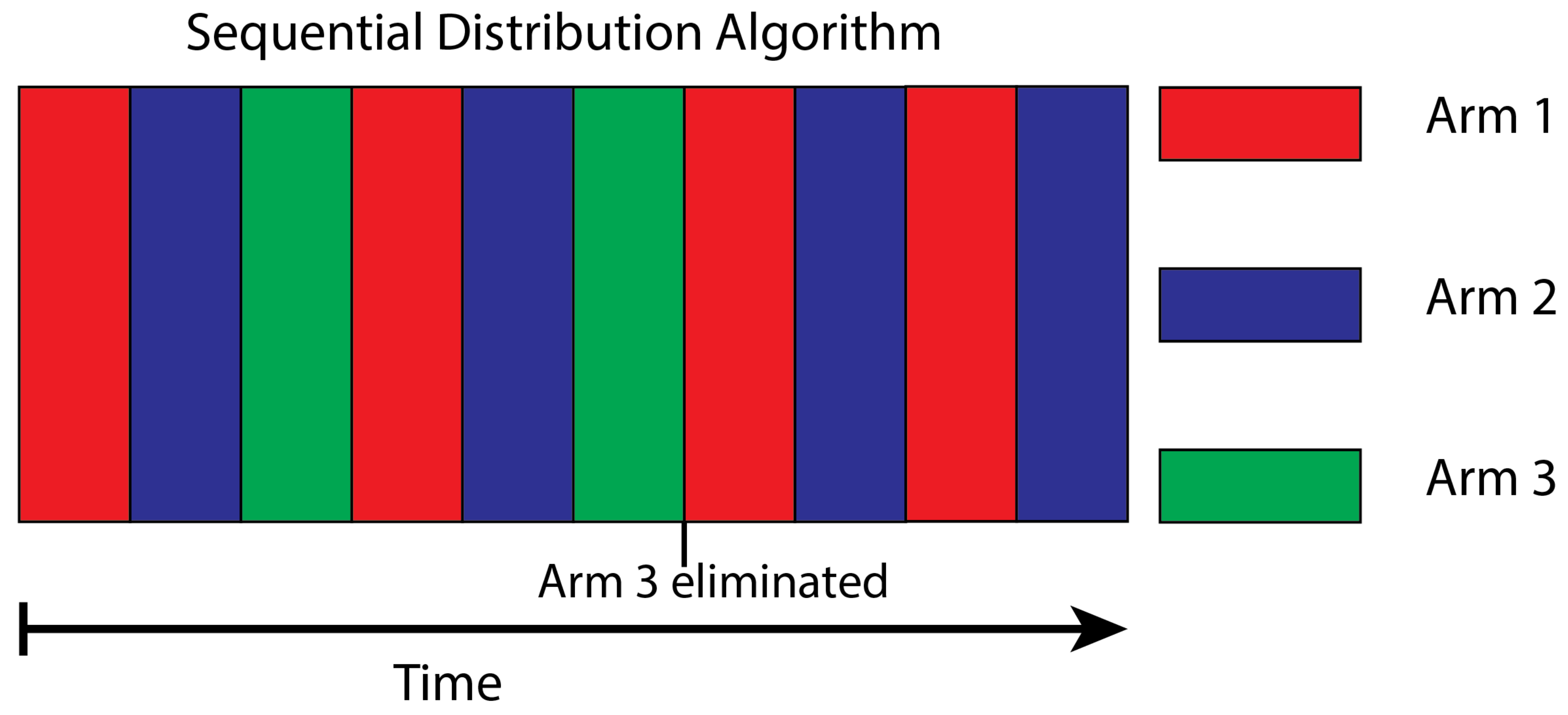
  Throughput: 1 pull/hour

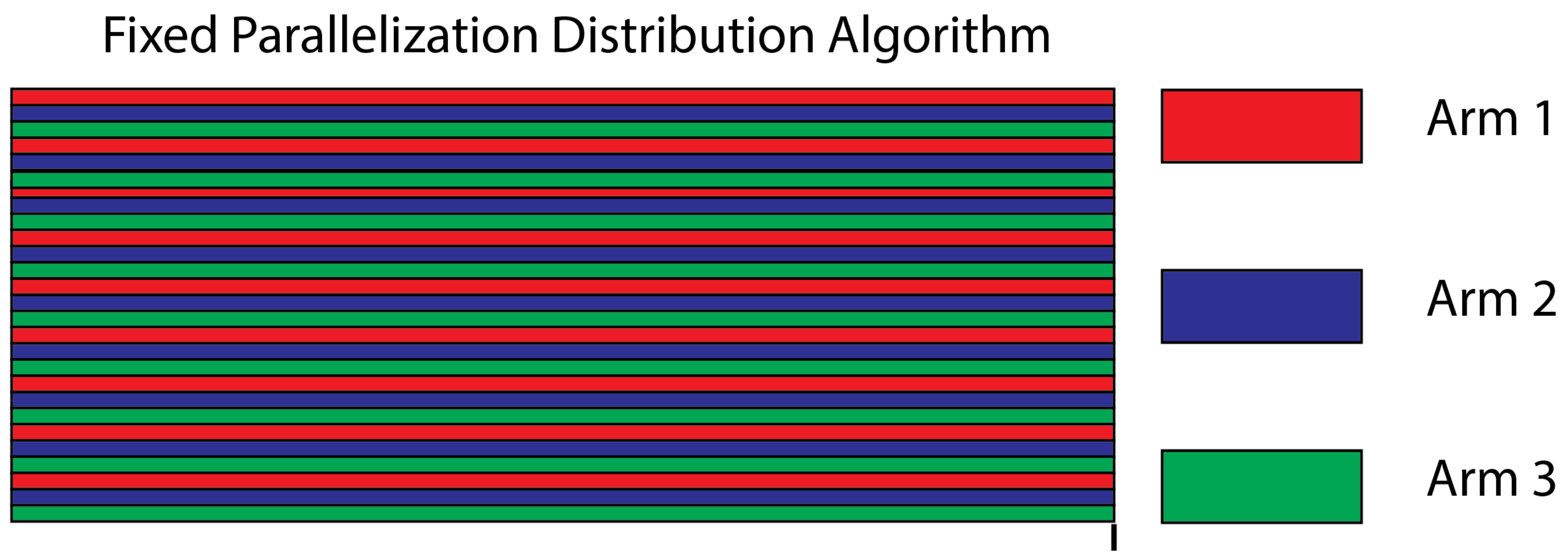
  Sampling time: 1 hour

- Pull resources with 1/16 resources each (16 at a time)

  Throughput: 4 pulls/hour

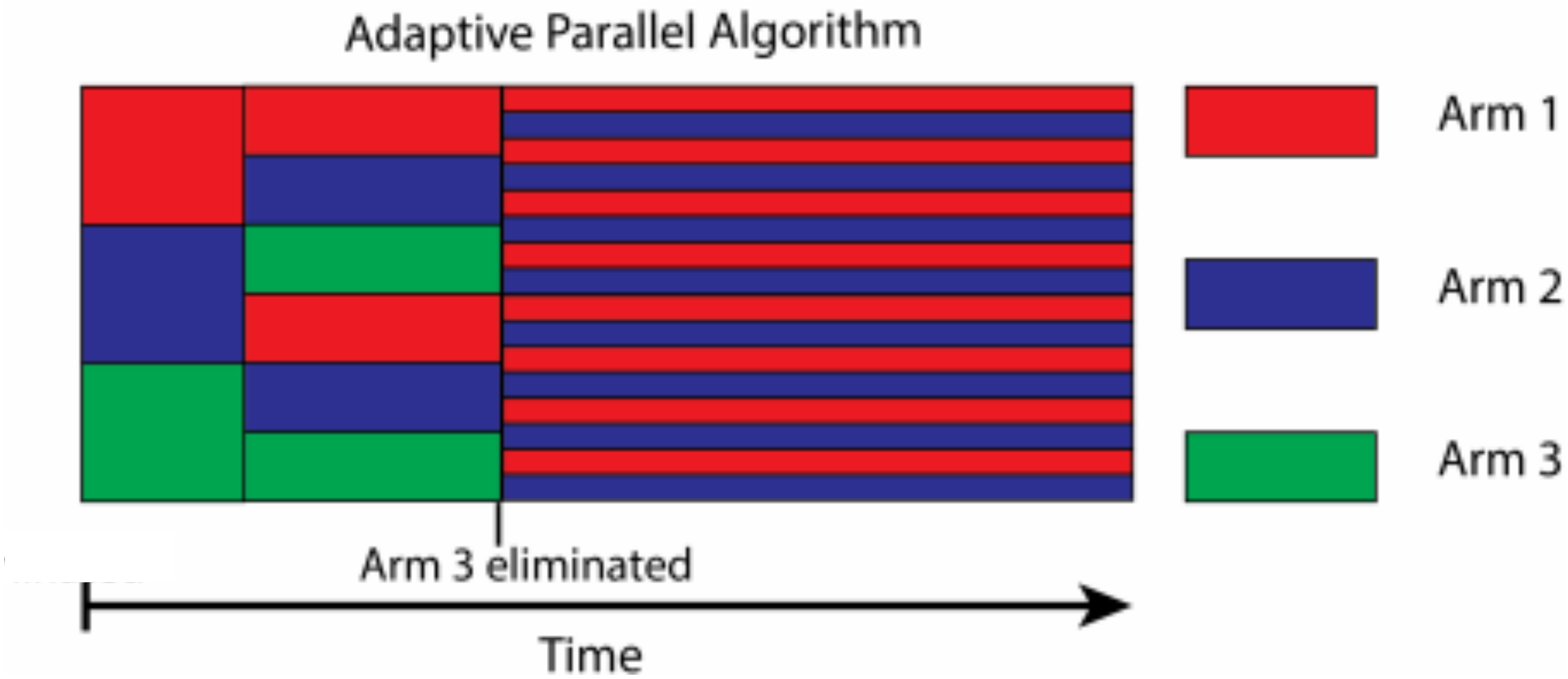
  Sampling time: 4 hours





Efficient algorithms must adaptively balance parallelism.

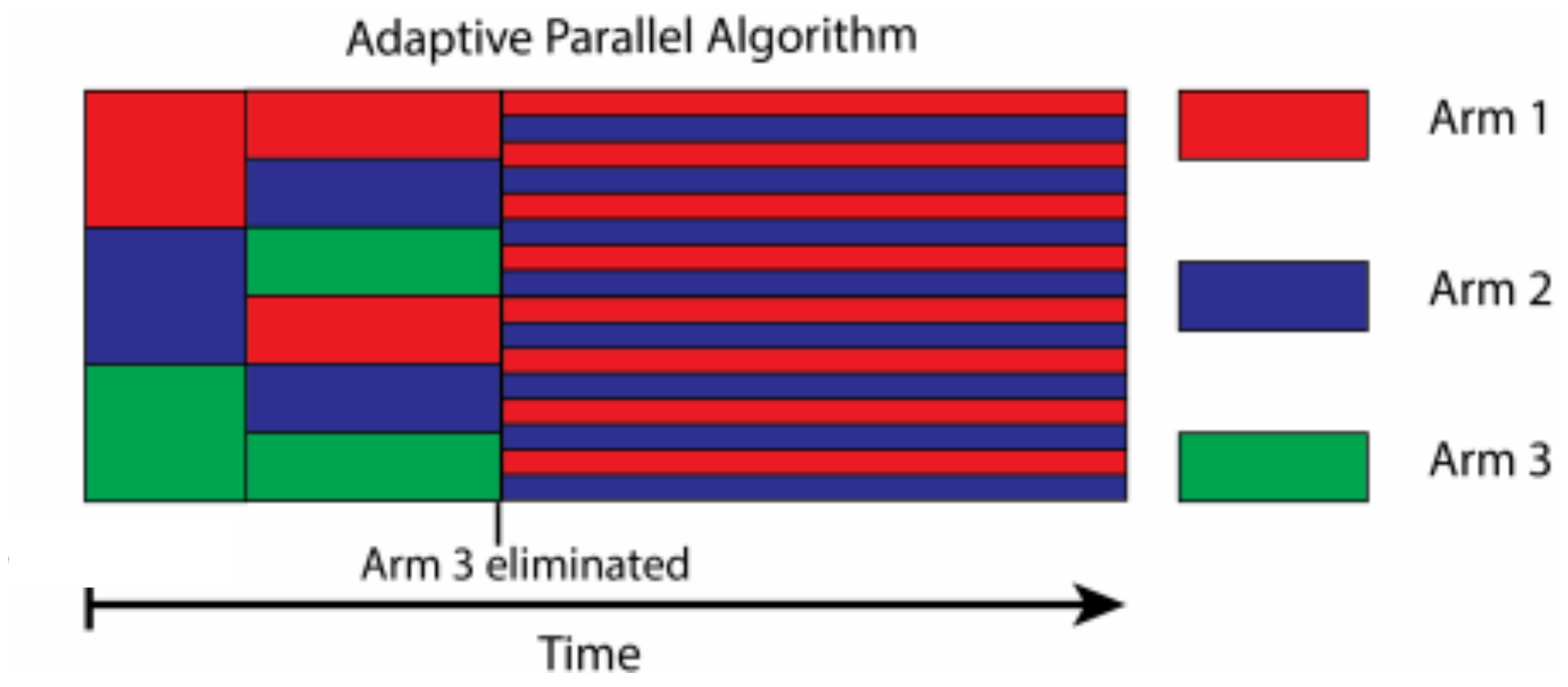
Algorithms must adaptively assess the difficulty of the problem.



Motivation
Problem Setup
Fixed Confidence Setting Results

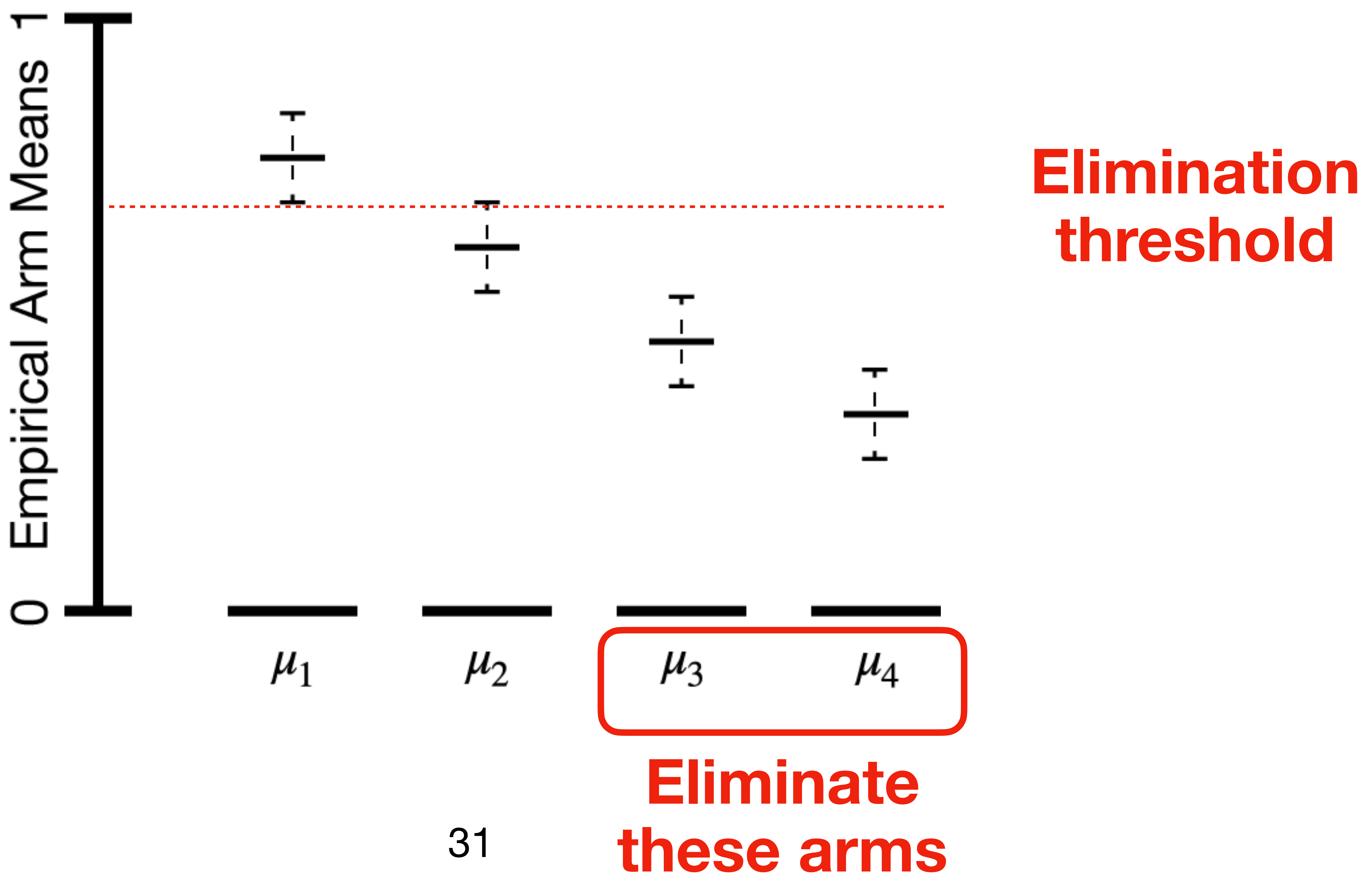
Fixed Confidence Setting

- Identify best arm with prob $\geq 1 - \delta$, minimize time *T*
- Algorithm: ***Adaptive-Parallel-Racing (APR)***
  - Maintain confidence bounds for each arm
    - eliminate when confidence bounds disjoint from top confidence bound
  - Adaptively increase parallelism during execution



APR maintains confidence bounds for each arm.

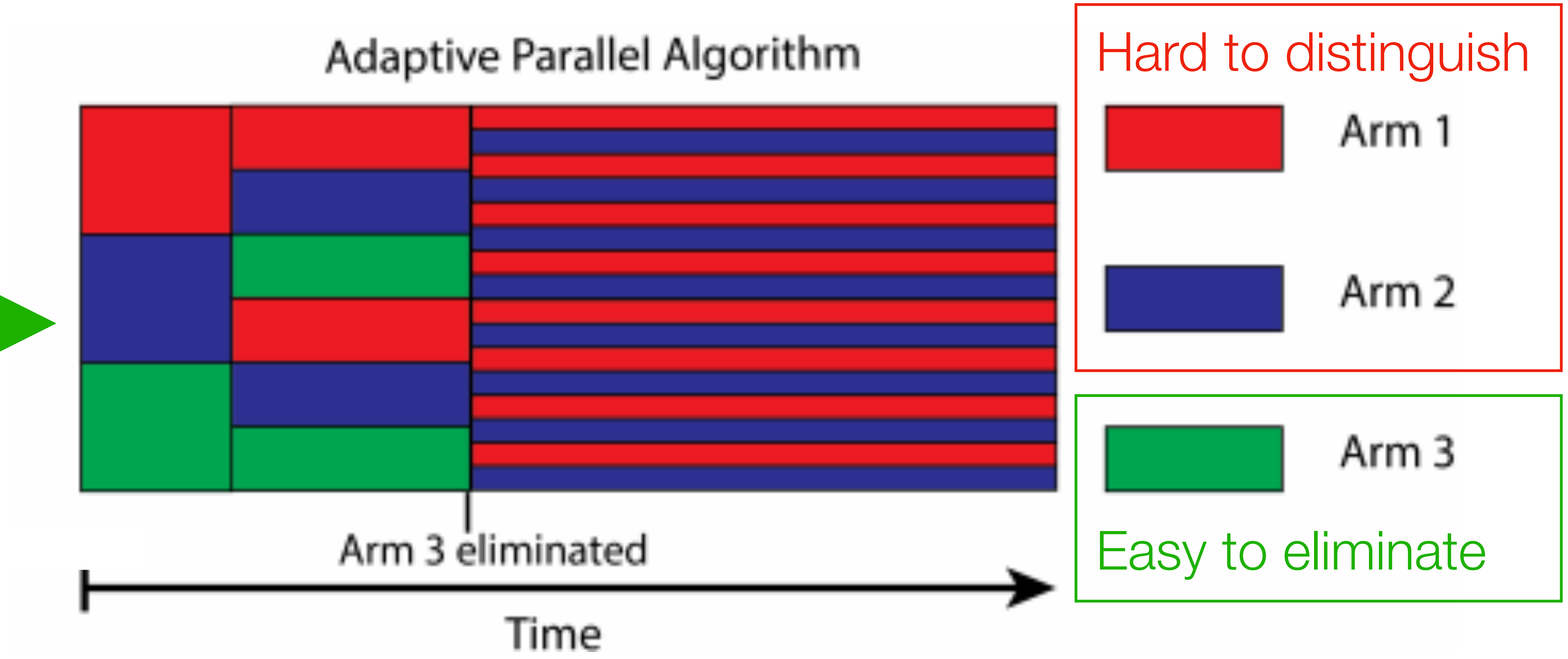
Eliminate when confidence bounds disjoint from top confidence bound.



APR adaptively increases parallelism for surviving arms.

Longer surviving, more samples required.

Increase throughput over time



Oracle Dynamic Program Sketch

- Suppose we can eliminate arm $i \geq 2$ after pulling all arms N_i times
  - Prior lower bounds suggest that $N_i \approx \frac{1}{\Delta_i^2}$
- Suppose we knew the N_i values but not the arms they correspond to.
- **How do we optimally schedule batches of arm pulls to minimize time?**

Oracle Dynamic Program Sketch

Example: $n = 3$ arms

- To eliminate arm 3, we need 100 pulls for each arm.
- To eliminate arm 2, we need 1000 pulls for each surviving arm.
- Option 1:
  - Pull all arms 100 times, eliminate arm 3, then pull arms 1 and 2, 900 times.
- Option 2:
  - Pull all arms 1000 times, eliminate arms 2 and 3.
- Optimal choice depends on scaling function.

Oracle Dynamic Program Sketch

- General n, N_i case:
  - Dependent on N_i values and scaling λ
- Define a dynamic program $\mathcal{T}\left(\{N_i\}_{i\in[n]}\right)$ that finds the optimal time $T^\star$
- Lower bound:
  - Show we cannot beat the DP solution by much
- Upper bound
  - Show APR gets close to the DP solution

Theorem 2: Fixed Confidence Lower Bound

- Show no δ-PAC algorithm cannot beat the DP solution $T^\star$ by much
- Uses a change of measure argument as in Kaufmann et al. 2016, and reduces BAI into the scheduling DP.
- Result: for any δ-PAC algorithm, expected time $\mathbb{E}T \in \tilde{\Omega}\left(T^\star\right)$

Emilie Kaufmann, Olivier Cappé, Aurélien Garivier. On the Complexity of Best Arm Identification in Multi-Armed Bandit Models. JMLR 2016.

Theorem 1: Fixed Confidence Upper Bound

- Show APR cannot lose to the DP solution $T^\star$ by much.
- Must show that we neither
  - Take too long to "ramp up" parallelism.
  - "Overshoot" by over-pulling arms.
- Proof shows that w.p. $\geq 1 - \delta$, neither of these events can occur often.
- W.p. $\geq 1 - \delta$, APR identifies the best arm in time $T \in \tilde{O}\left(T^\star\right)$, where $\tilde{O}$ ignores sub polynomial terms.

Comparison of Theoretical Results

- Lower bound: for any δ-PAC algorithm, expected time $\mathbb{E}T \in \tilde{\Omega}\left(T^\star\right)$.

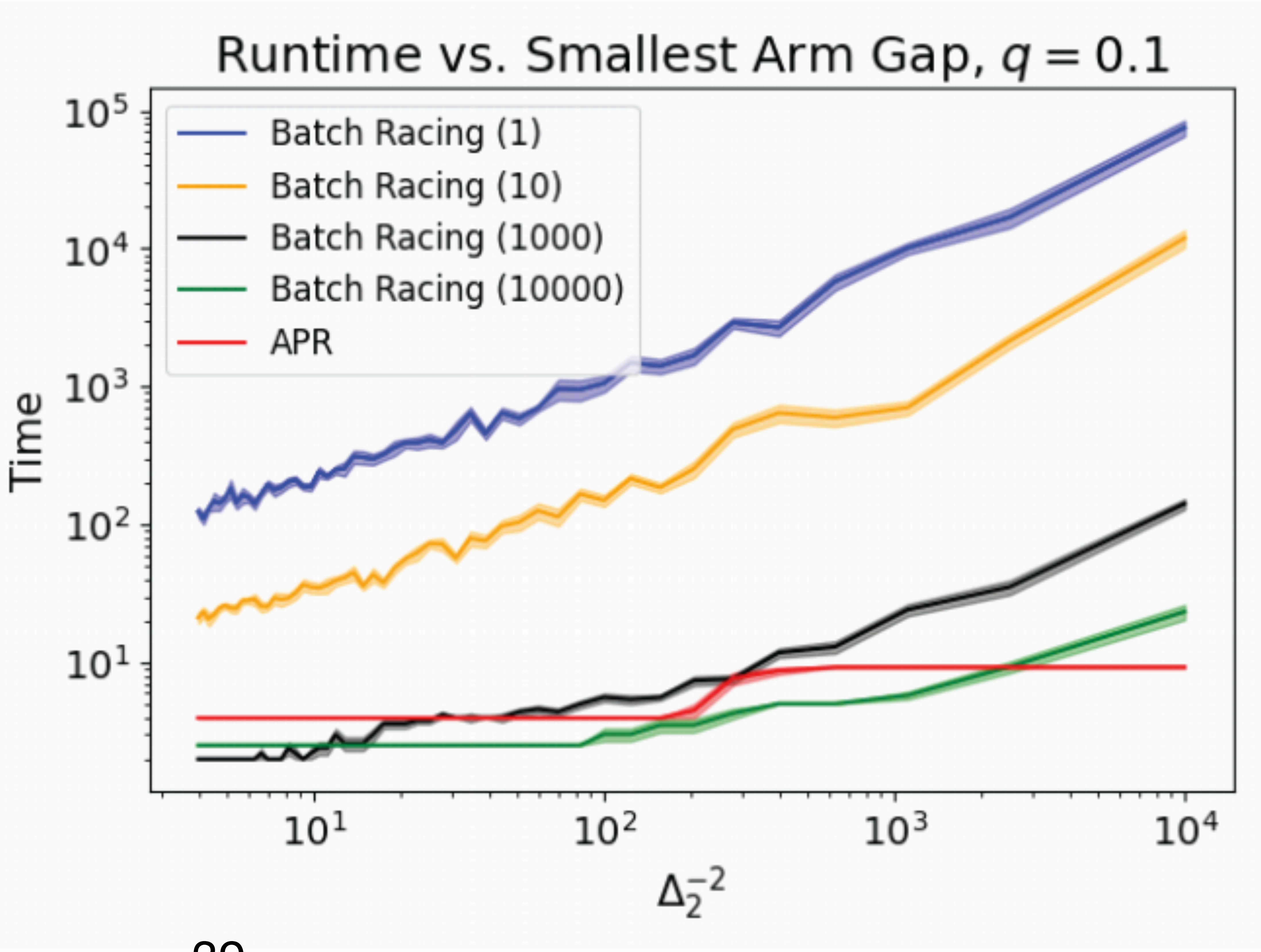
- Upper bound: APR is δ-PAC, and has time $T \in \tilde{O}\left(T^\star\right)$ w.p. at least $1 - \delta$.

Experiments

APR: Red

Baselines: have fixed batch size, each good on specific problems

Poor scaling $\lambda(b) = b^{0.1}$

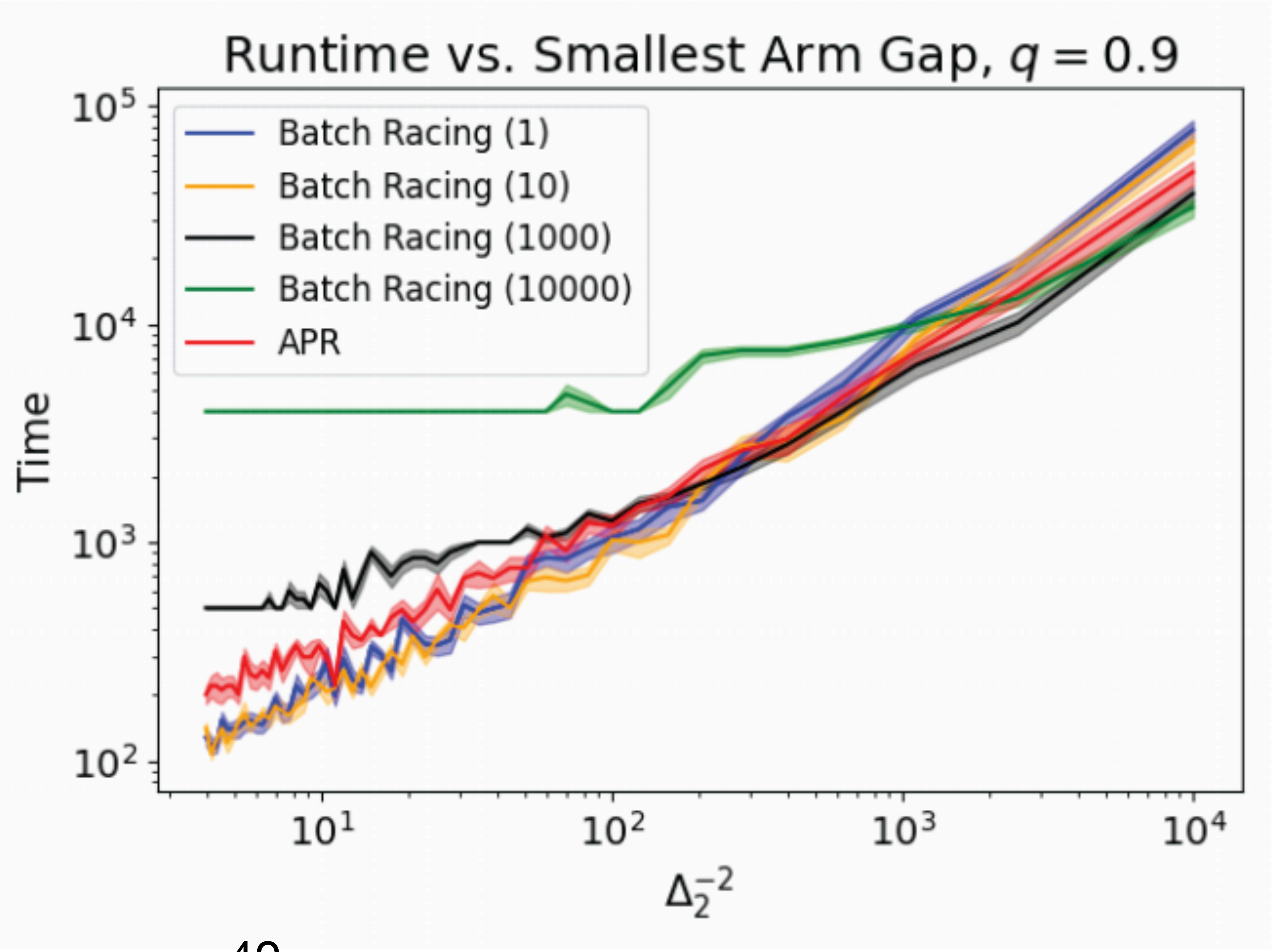


Experiments

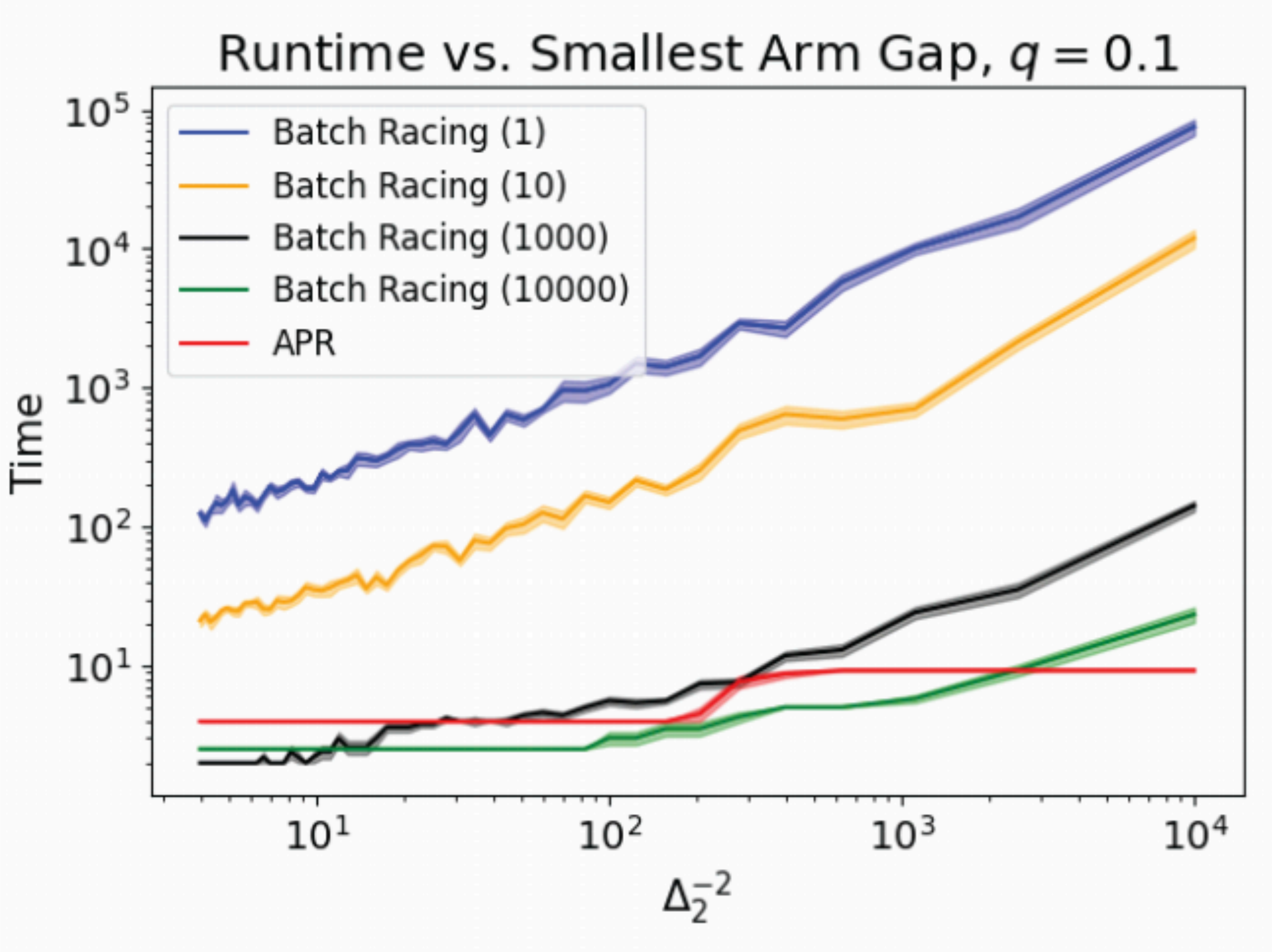
APR: Red

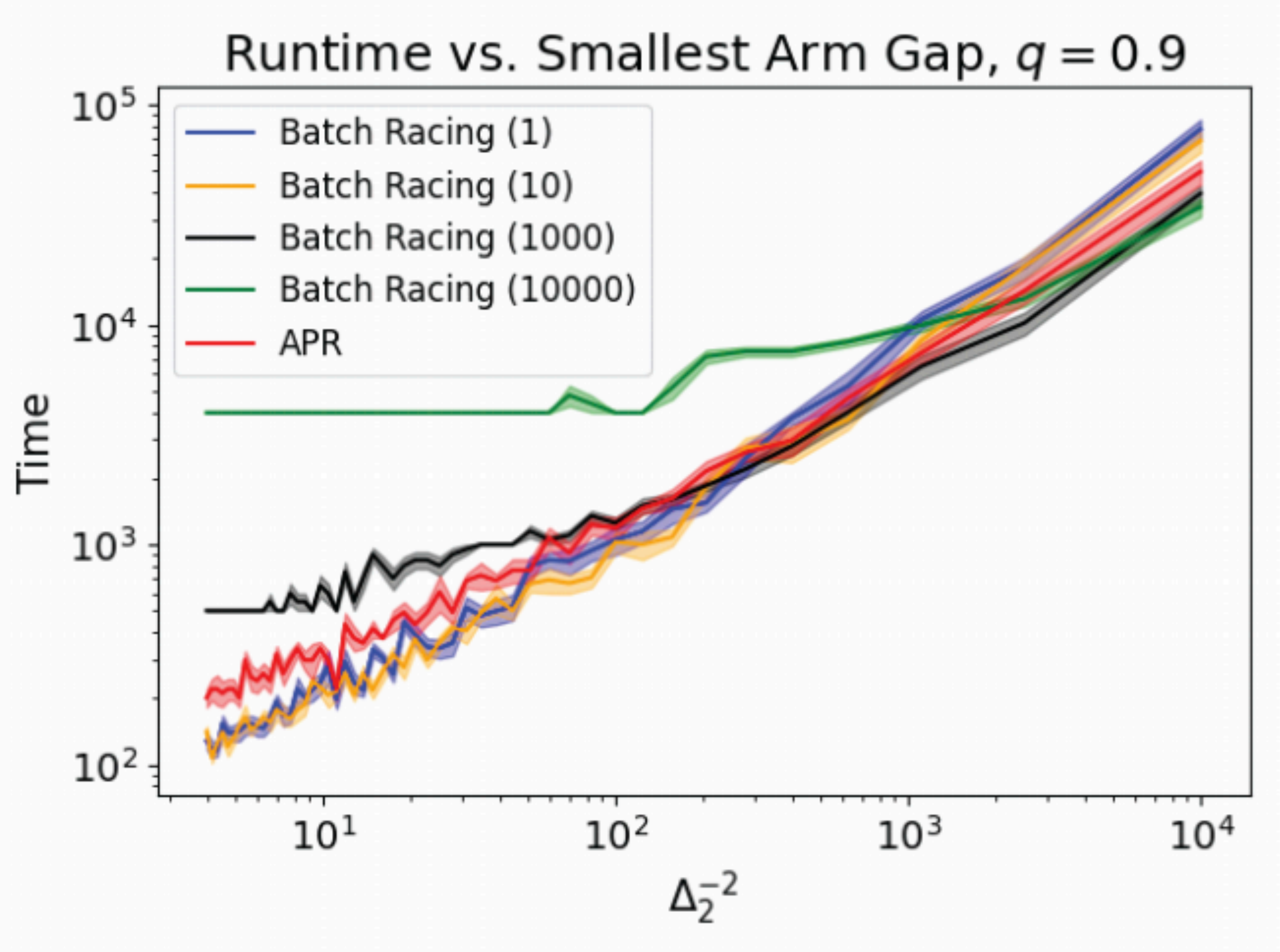
Baselines: have fixed batch size, each good on specific problems

Good scaling $\lambda(b) = b^{0.9}$



APR consistently matches the best baselines for each scaling function



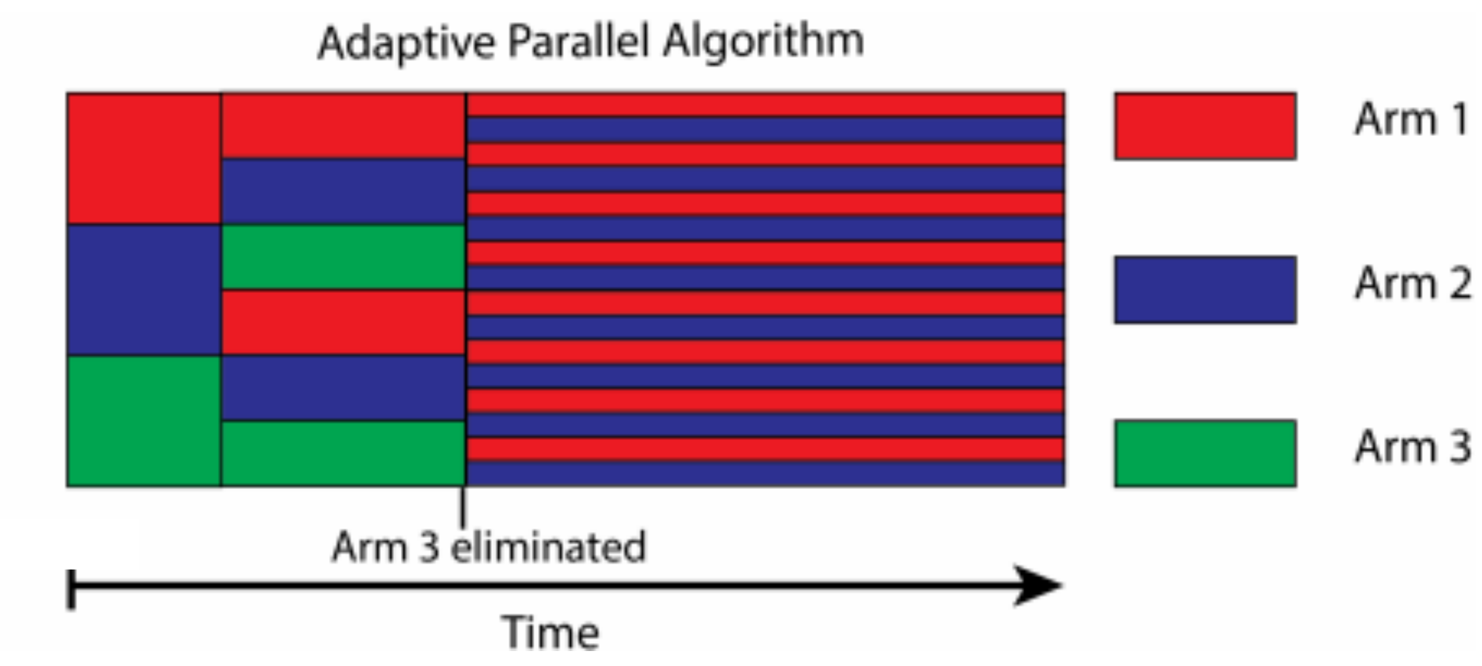


Dirty Laundry and Future Work

- In fixed confidence setting: lower bound in expectation, upper bound w.h.p.
- Lower bounds for fixed deadline setting
- Real-world experiments
- Core assumption: each arm has **same, known** scaling function λ_i
- Elastic resource that can grow and shrink:
  - https://arxiv.org/pdf/2106.03221.pdf

Summary

- Novel problem for parallel best arm identification
  - Considers time and parallel resource scaling
- Matching upper and lower bounds in the fixed confidence setting
- Upper bound in the fixed deadline setting
- Contact: bthananjeyan@berkeley.edu



riselab UC Berkeley

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH