Computing with Unreliable Resources: Design, Analysis and Algorithms

Da Wang, Ph.D Candidate

Thesis Committee: Gregory W. Wornell (Advisor)
Yury Polyanskiy
Devavrat Shah

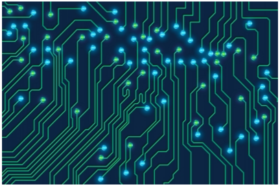
Signals, Information and Algorithms Laboratory

Thesis Defense

May 8, 2014

Computing with unreliable resources: an emerging paradigm

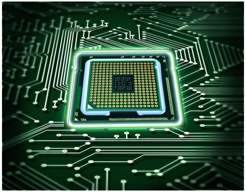
- Large amount of data requires large amount of computing resources
  - VLSI circuit
  - cloud computing/data centers
  - …
- Why are resources unreliable?



technological constraints

cost constraints

A study via concrete applications

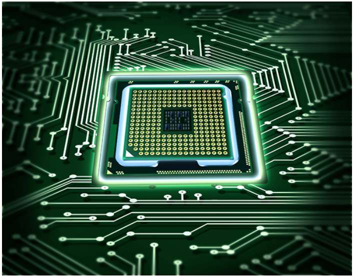


1. Circuit design with unreliable components

2. Scheduling parallel tasks with variable response times

3. Crowd-based ranking via noisy comparisons

Reliable circuit design with unreliable components



AMD claims 20nm transition signals the end of Moore's law

Economic viability comes into question

By **Lawrence Latif**

Tue Apr 02 2013, 15:41

SAN FRANCISCO: CHIP DESIGNER AMD claims that the delay in transitioning from 28nm to 20nm highlights the beginning of the end for Moore's Law.

AMD was one of the first consumer semiconductor vendors to make use of TSMC's 28nm process node with its Radeon HD 7000 series graphics cards, but like every chip vendor it is looking to future process nodes to help it increase performance. The firm told The INQUIRER the time taken to transition to 20nm signals the beginning of the end for Moore's Law.

Famed Intel co-founder and electronics engineer Gordon Moore predicted that total the number of transistors would double every two years. He also predicted that the 'law' would not continue to apply for as long as it has. It was professor Carver Mead at Caltech that coined the term Moore's Law, and now one of Mead's students, John Gustafson, chief graphics product architect at AMD, has said that Moore's Law is ending because it actually refers to a doubling of transistors that are economically viable to produce.

Gustafson said, "You can see how Moore's Law is slowing down. The original statement of Moore's Law is the number of transistors that is more economical to produce will double every two years. It has become warped into all these other forms but that is what he originally said."

According to Gustafson, the transistor density afforded by a process node defines the chip's economic viability. He said, "We [AMD] want to also look for the sweet spot, because if you print too few transistors your chip will cost too much per transistor and if you put too many it will cost too much per transistor. We've been waiting for that transition from 28nm to 20nm to happen and it's taking longer than Moore's Law would have predicted."

Gustafson was pretty clear in his view of transistor density, saying, "I'm saying you are seeing the beginning of the end of Moore's law."

AMD isn't the only chip vendor looking to move to smaller process nodes and has to wait on TSMC and Globalfoundries before it can make the move. Even Intel, with its three year process node advantage over the industry is having problems justifying the cost of its manufacturing business to investors, so it could be the economics rather than the engineering that puts an end to Moore's Law. µ

Chief product architect at AMD:

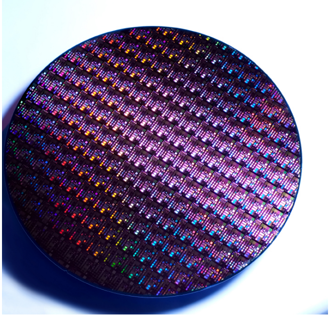
> *"If you print too few transistors your chip will cost too much per transistor . . .*
>
> *. . . and if you put too many it will cost too much per transistor."*

New challenge: fabrication flaws

Fabrication flaws

1. process variations
2. fabrication defects

get worse as
we approach physical limits!

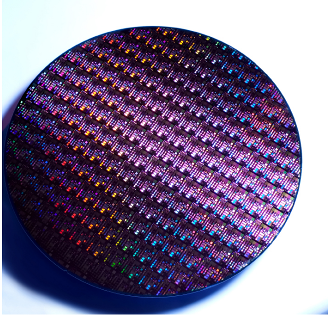


New challenge: fabrication flaws

Fabrication flaws

1. process variations
2. fabrication defects

get worse as
we approach physical limits!



Flash ADC design with imprecise comparators — this talk ✓

- Captures process variations

Digital circuit design with faulty components — in thesis

- Captures fabrication defects

Reliable Analog-to-Digital Converter Design with imprecise comparators

A theoretical framework

Reliable Analog-to-Digital Converter Design with imprecise comparators

A theoretical framework

What are the fundamental performance limits?

Reliable Analog-to-Digital Converter Design with imprecise comparators

A theoretical framework

What are the fundamental performance limits?

Is optimal design for the precise comparators case still optimal?

Reliable Analog-to-Digital Converter Design with imprecise comparators

A theoretical framework

What are the fundamental performance limits?

Is optimal design for the precise comparators case still optimal?

Should we use imprecise comparators?

Reliable Analog-to-Digital Converter Design with imprecise comparators

A theoretical framework

What are the fundamental performance limits?

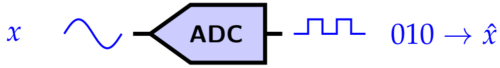
Is optimal design for the precise comparators case still optimal?

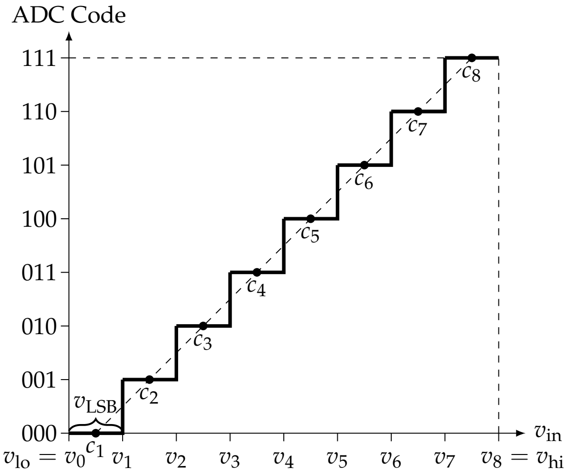
Should we use imprecise comparators?

Joint work with: Yury Polyanskiy, Gregory Wornell

Acknowledgment: Frank Yaul, Anantha Chandrakasan

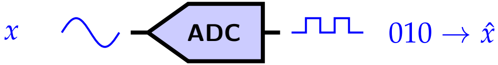
Analog-to-Digital Converter (ADC)

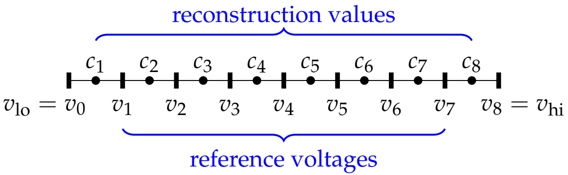




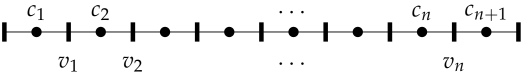
- 2^b reconstruction values
- $n = 2^b - 1$ reference voltages

Analog-to-Digital Converter (ADC)

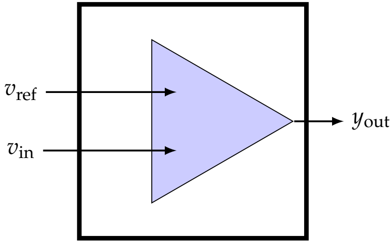




- 2^b reconstruction values
- $n = 2^b - 1$ reference voltages

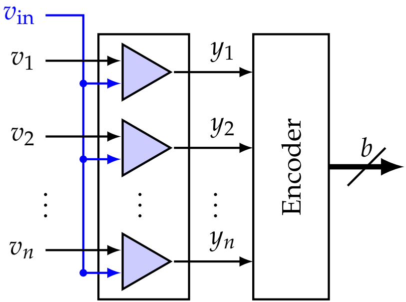


Comparator



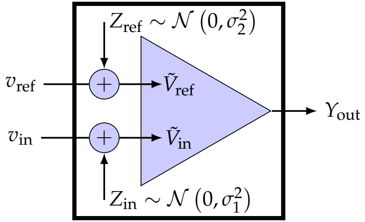
$$y_{\text{out}} = \begin{cases} 1 & v_{\text{in}} > v_{\text{ref}} \\ 0 & v_{\text{in}} \le v_{\text{ref}} \end{cases}$$

The Flash ADC architecture



$$n = 2^b - 1$$

The imprecise comparator due to process variation



Z_{in} and Z_{ref}:

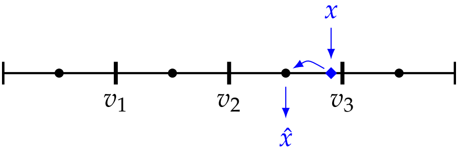
- offsets due to process variation
- variation $\nearrow$ as comparator size $\searrow$
- independent, zero-mean Gaussian distributed [Kinget 2005, Nuzzo 2008]

Note:

- fixed after fabrication
- randomness: over a collection of comparators
- aggregate variation:

$$Z = Z_{\text{ref}} - Z_{\text{in}} \sim \mathsf{N}\left(0, \sigma^2\right)$$

2-bit ADC

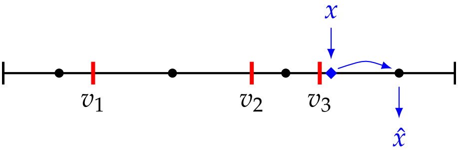


error function:

$$e(x) = x - \hat{x}$$

Reference voltages impacts ADC performance

2-bit ADC



error function:

$$e(x) = x - \hat{x}$$

A call for mathematical framework

Existing theoretical error analysis (e.g., [Lundin 2005])

- assumes small process variation
- does not attempt to change the design

ADC design with imprecise comparators

Practice
- ADC with redundancy [Flynn *et al.*, 2003]
- ADC with redundancy, calibration and reconfiguration [Daly *et al.*, 2008]

A call for mathematical framework

Existing theoretical error analysis (e.g., [Lundin 2005])

- assumes small process variation
- does not attempt to change the design

ADC design with imprecise comparators

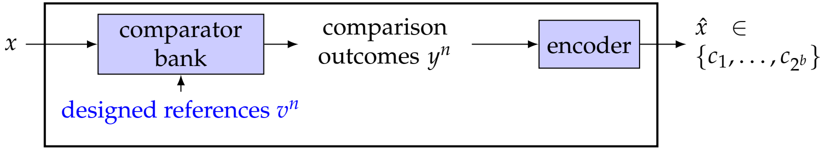
Practice
- ADC with redundancy [Flynn *et al.*, 2003]
- ADC with redundancy, calibration and reconfiguration [Daly *et al.*, 2008]

Theory
- Little prior work
- Related: scalar quantizer with random thresholds for uniform input [Goyal 2011]

System model: ADC with redundancy and calibration

b-bit ADC

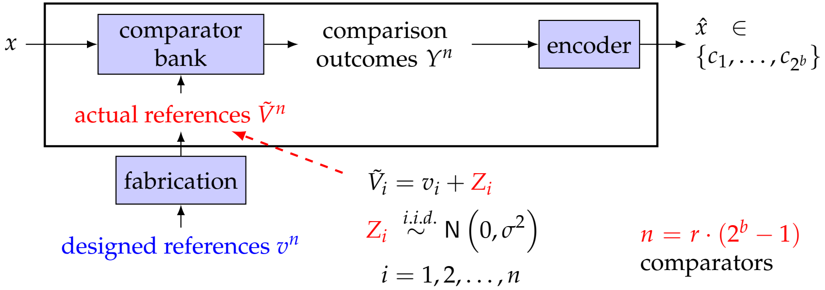
classical



$n = 2^b - 1$ comparators

b-bit ADC

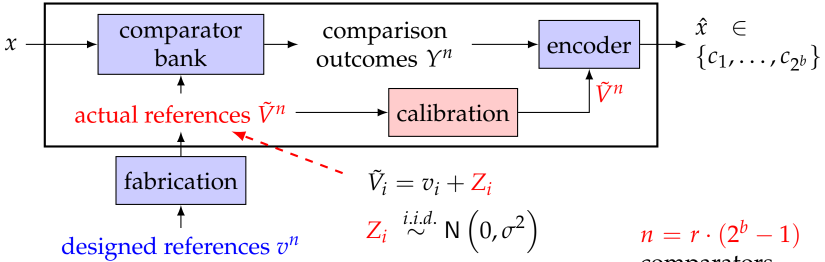
with redundancy



- r: redundancy factor

b-bit ADC

with redundancy and calibration



$$\tilde{V}_i = v_i + Z_i$$

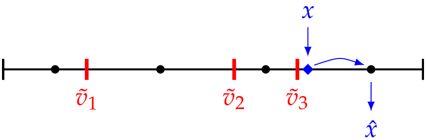
$$Z_i \overset{i.i.d.}{\sim} \mathsf{N}\left(0, \sigma^2\right)$$

$$i = 1, 2, \ldots, n$$

$n = r \cdot (2^b - 1)$ comparators

- r: redundancy factor

Performance measures of ADC



error function $\quad e(x) = x - \hat{x}$

mean-square error

$$\mathrm{MSE} = \mathbb{E}_X\left[e(X)^2\right]$$

maximum quantization error

$$e_{\max} = \max_x |e(x)|$$

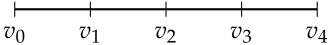
v^n — $\tilde{V}^n$ $\longrightarrow$ analyze performance MSE, $e_{\max}$

$\longrightarrow$ Is uniform $v_1, v_2, \ldots, v_n$ still optimal for uniform input?

$\longrightarrow$ Is scaling down the size of comparators actually beneficial?

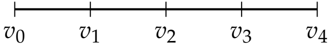
Challenge: randomness in reference voltages

What we design:

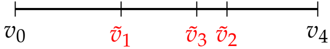


Challenge: randomness in reference voltages

What we design:

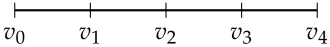


After fabrication:

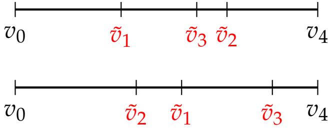


Challenge: randomness in reference voltages

What we design:

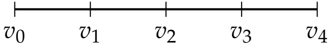


After fabrication:

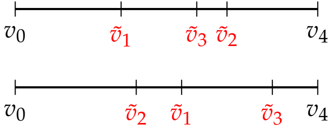


or …

What we design:



After fabrication:



or ...

Observations

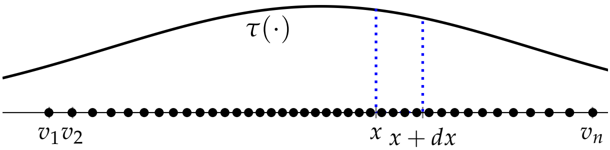
- Ordering may change $\rightarrow$ order statistics
- Random interval sizes $\rightarrow$?

interval sizes $\longleftrightarrow$ density of references $\longleftrightarrow$ high resolution approximation

Assume $n \to \infty$

- Represent v^n by point density functions $\tau(x)$

$$\tau(x)\,dx \approx \frac{\text{number of } v^n \text{ in } [x, x+dx]}{n}$$

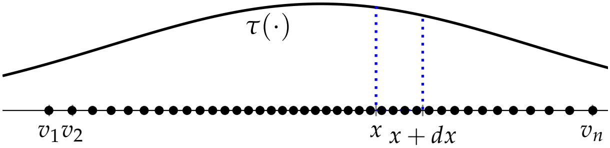


High resolution approximation

Assume $n \to \infty$

- Represent v^n by point density functions $\tau(x)$

$$\tau(x)\,dx \approx \frac{\text{number of } v^n \text{ in } [x, x+dx]}{n}$$



- $\tilde{V}^n$: point density functions $\lambda(x)$

$$\lambda(x)\,dx \approx \frac{\mathbb{E}\left[\text{number of } \tilde{V}^n \text{ in } [x, x+dx]\right]}{n}$$

- Point density function simplifies analysis!

Point density function guides references design

$$\text{reference voltages } v^n \; \underset{\text{given } n\text{, ``sample'' } \tau(\cdot)}{\overset{\text{high res. approx.}}{\rightleftarrows}} \; \text{point density function } \tau(\cdot)$$

Examples

- $\tau \sim \mathsf{Unif}\left([-1, 1]\right)$
- v^n: n-point uniform grid on [-1, 1]

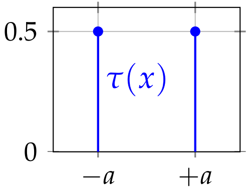
Point density function guides references design

reference voltages v^n $\xrightarrow{\text{high res. approx.}}$ point density function $\tau(\cdot)$

point density function $\tau(\cdot)$ $\xrightarrow{\text{given } n, \text{ “sample” } \tau(\cdot)}$ reference voltages v^n

Examples

- $\tau(x) = 0.5 \cdot \delta(x - a) + 0.5 \cdot \delta(x + a)$
- v^n:
  - $n/2$ reference voltages at $+a$
  - $n/2$ reference voltages at $-a$



With process variation, fabricated references matters

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\lambda(\cdot) \arrow[r, shift right, "\text{sample}"'] & \tilde{V}^n \arrow[l, shift right, "\text{high res. approx.}"'] & \text{what determines performance} \\
\tau(\cdot) \arrow[r, shift right, "\text{``sample''}"'] & v^n \arrow[l, shift right, "\text{high res. approx.}"'] \arrow[u, "\begin{array}{l} \tilde{V}_i = v_i + Z_i \\ Z_i \overset{i.i.d.}{\sim} \phi(\cdot) \sim \mathsf{N}\left(0, \sigma^2\right) \end{array}"'] & \text{what we can design}
\end{tikzcd}$$

- Performance characterization in $\lambda(\cdot)$
- Want to find the optimal $\tau(\cdot)$

With process variation, fabricated references matters

high res. approx.

$\lambda(\cdot) \underset{\text{sample}}{\overset{\text{high res. approx.}}{\rightleftarrows}} \tilde{V}^n$ what determines performance

[**W.**, Polyanskiy & Wornell, ISIT'14]:

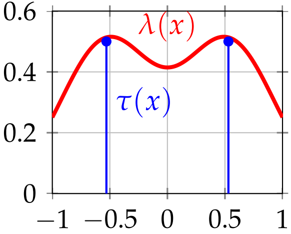
$$\lambda(x) = (\tau * \phi)(x)$$

($*$: convolution)

$$\tilde{V}_i = v_i + Z_i$$

$$Z_i \overset{i.i.d.}{\sim} \phi(\cdot) \sim \mathsf{N}\left(0, \sigma^2\right)$$

$\tau(\cdot) \underset{\text{“sample”}}{\overset{\text{high res. approx.}}{\rightleftarrows}} v^n$ what we can design



- Performance characterization in $\lambda(\cdot)$
- Want to find the optimal $\tau(\cdot)$

Process variation increases MSE 6-fold

Input $X \sim f_X(\cdot)$,

$$\boxed{\text{MSE} = \mathbb{E}_X\left[e(X)^2\right]}$$

classical case [Bennett 1948, Panter & Dite 1951]

$$\text{MSE} \simeq \frac{1}{12n^2}\int \frac{f_X(x)}{\lambda^2(x)}\,dx \qquad\qquad \lambda = \tau$$

with process variations [**W.**, Polyanskiy & Wornell, ISIT'14]

$$\text{MSE} \simeq \frac{1}{2n^2}\int \frac{f_X(x)}{\lambda^2(x)}\,dx \qquad\qquad \lambda = \tau * \phi$$

Why 6 times?

uniform grid vs. random division of an interval
(a topic in order statistics)

Optimal τ

- a necessary and sufficient condition [**W.**, Polyanskiy & Wornell, ISIT'14]

MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1,1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

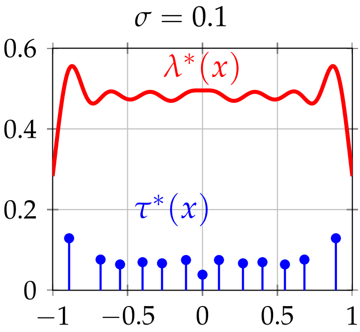
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1,1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

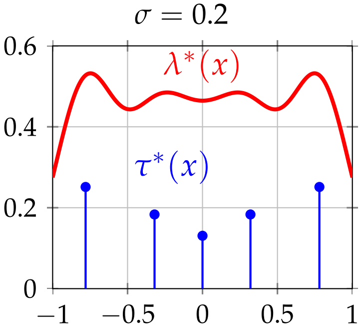
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1, 1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

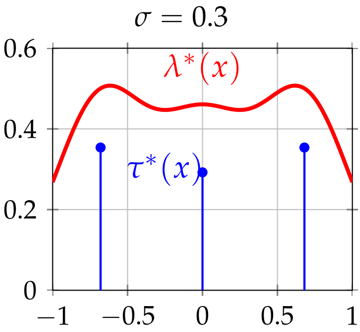
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1,1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

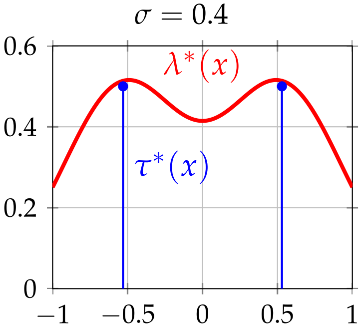
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}([-1,1])$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

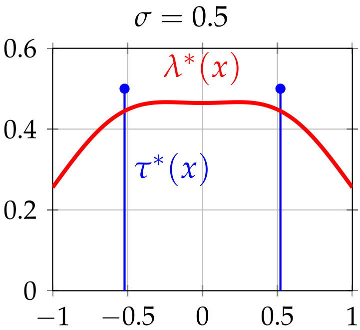
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1,1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

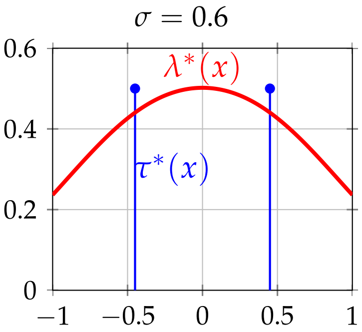
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1,1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

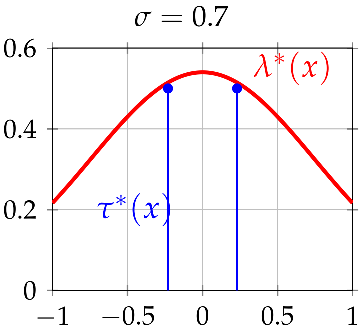
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



MSE-optimal designs can be quite different

Uniform input distribution [**W.**, Polyanskiy & Wornell, ISIT'14]

$$f_X \sim \mathsf{Unif}\left([-1,1]\right)$$

$\sigma_0 \approx 0.7228$

$\sigma < \sigma_0$

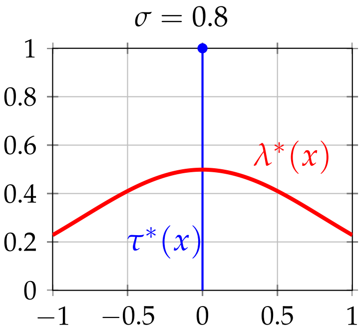
locally optimal iterative optimization

$\Rightarrow \tau^*(x)$

$\sigma \geq \sigma_0$

the necessary and sufficient condition

$\Rightarrow \tau^*(x) = \delta(x)$



Analyzing maximum quantization error

$$\boxed{e_{\max} = \max_x |e(x)|}$$

- High resolution approximation for

$$\begin{aligned} \text{c.d.f.} \quad & \mathbb{P}\left[e_{\max} \leq x\right] \\ \text{Range } [a, b] \quad & \mathbb{P}\left[e_{\max} \in [a, b]\right] \approx 1 \end{aligned}$$

- Accurate for finite n (e.g., when $n \geq 100$)
- $e_{\max}$ often measured in

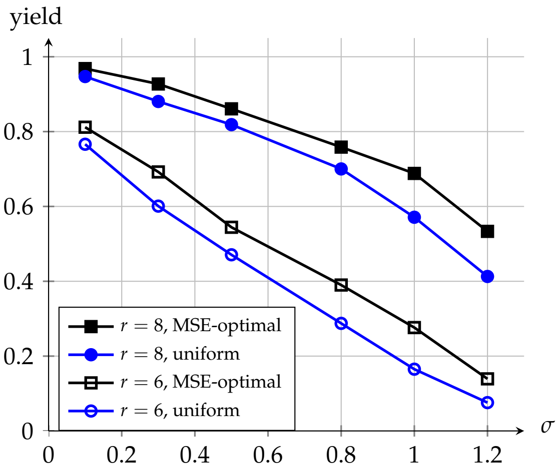
$$\text{LSB} \triangleq \frac{\text{input range}}{2^b}$$

MSE-optimal designs improve yield

- 5%–10% for 6-bit Flash ADC

$$\text{Yield} \triangleq \mathbb{P}\left[e_{\max} \leq \Delta_{\max}\right]$$

$$\text{e.g.} \quad \Delta_{\max} = 1\text{LSB}$$



Scaling down the size of comparators is beneficial

For circuit fabrication [Kinget 2005, Nuzzo 2008],

$$\text{process variation} \qquad \sigma^2 \propto \frac{1}{\text{component area}}$$

Given a fixed silicon area,

$$\text{\# components} \qquad n \propto \frac{1}{\text{component area}}$$

Uniform input distribution, when $\sigma \geq \sigma_0$,

$$\mathrm{MSE} \approx 2\pi\sigma^2/n^2 \qquad\qquad \mathrm{MSE} = \Theta\left(1/n\right)$$

$$e_{\max} \overset{d.}{=} \Theta\left(\sqrt{\pi/2}\,\sigma\frac{b}{n}\right) \quad \xRightarrow{\sigma^2 \propto n} \quad e_{\max} = \Theta\left(\frac{b}{\sqrt{n}}\right)$$

Scaling down the size of comparators is beneficial

For circuit fabrication [Kinget 2005, Nuzzo 2008],

$$\text{process variation} \qquad \sigma^2 \propto \frac{1}{\text{component area}}$$

Given a fixed silicon area,

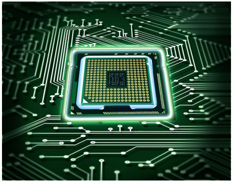
$$\text{\# components} \qquad n \propto \frac{1}{\text{component area}}$$

Uniform input distribution, when $\sigma \geq \sigma_0$,

$$\begin{aligned} \mathrm{MSE} &\approx 2\pi\sigma^2/n^2 \\ e_{\max} &\overset{d.}{=} \Theta\left(\sqrt{\pi/2}\,\sigma\frac{b}{n}\right) \end{aligned} \quad \xRightarrow{\sigma^2 \propto n} \quad \begin{aligned} \mathrm{MSE} &= \Theta\left(1/n\right) \\ e_{\max} &= \Theta\left(\frac{b}{\sqrt{n}}\right) \end{aligned}$$

Building an ADC with more smaller but less precise comparators improves accuracy!

Computing with unreliable resources: circuit design



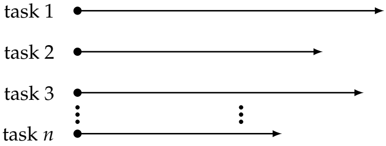
mathematical abstraction	→ asymptotic analysis →	performance trade-offs	→	design guidance
	quantization theory, order statistics	redundancy vs. yield		distribution of fabrication variation

Scheduling parallel tasks with variable response times

Executing parallel tasks: simple yet important

Executing parallel tasks

- “Map” stage of MapReduce
- distributed parallel algorithms
  - ADMM, MCMC, ...
- time series analysis
  - yearly data → weekly data
- crowd sourcing
- ...



In common

Given:

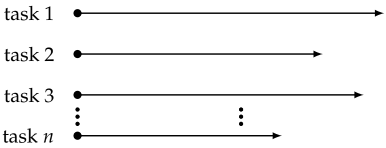
- a large collection of tasks that can be run in parallel

Want:

- results for all tasks

Executing parallel tasks

- "Map" stage of MapReduce
- distributed parallel algorithms
  - ADMM, MCMC, ...
- time series analysis
  - yearly data $\rightarrow$ weekly data
- crowd sourcing
- ...



In common

Given:

- a large collection of tasks that can be run in parallel

Want:

- results for all tasks

But this could be slow!

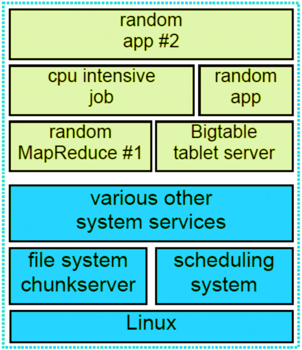
Issue: variability in data centers

Observations

- The response time of a computer in a data center varies

In Google's data center

Shared machine [Dean 2012]



Issue: variability in data centers

Observations

- The response time of a computer in a data center varies
- Latency: determined by the slowest machine
  - worse as we get more machines

In Google's data center

Execution times [Dean 2013] for a large collection of tasks

- median completion time for one: 1ms
- median completion time for all: 40ms

How to reduce latency?

Backup tasks in Google MapReduce

Dean *et al.*, 2008

The backup task option

1. Run n tasks on n machines in parallel
2. Replicate: when 10% of the tasks left
3. Take the earliest results

Effectiveness

- Reduce latency significantly (e.g., 1/3 for distributed sort)
- Handles the issue of “stragglers”

A call for theoretical analysis

Considerable follow-up work in systems

1. [Zaharia *et al.*, USENIX OSDI 2008]: adopted by Facebook
2. [Ananthanarayanan *et al.*, USENIX OSDI 2010]: adopted by Microsoft Bing
3. [Ananthanarayanan *et al.*, USENIX NSDI 2013]

Existing theoretical work inadequate

- Stochastic scheduling: task replication not considered
- Need to understand
  - latency reduction vs. additional resource usage
  - when and how replication could be beneficial

Scheduling problem formulation

Problem

Executing a collection of n parallel tasks.

- n: hundreds or thousands [Reiss *et al.*, 2012]

System model

- Execution time of each task $\overset{i.i.d.}{\sim} F_X$.
- Scheduling actions
  - Send a task to a new machine to run
  - Terminate all machines running a certain task
- Feedback: instantaneous feedback upon completion

The j-th copy of task i

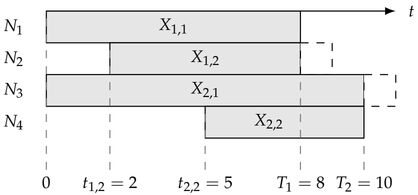
$$\begin{cases} \text{launched at time} & t_{i,j} \\ \text{execution time} & X_{i,j} \overset{i.i.d.}{\sim} F_X \end{cases}$$

- Completion time for task i:

$$T_i \triangleq \min_j (t_{i,j} + X_{i,j})$$

- Latency:

$$T \triangleq \max_i T_i$$



- $T = \max \{T_1, T_2\} = 10$

Total machine time as cost measure

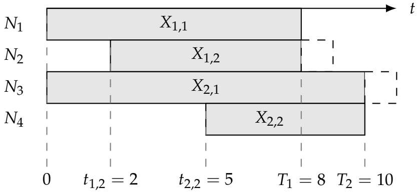
The j-th copy of task i

$$\begin{cases} \text{launched at time} & t_{i,j} \\ \text{execution time} & X_{i,j} \overset{i.i.d.}{\sim} F_X \end{cases}$$

In data centers

Total machine time

$$C \triangleq \sum_{i=1}^{n} \sum_{j=1}^{r_i+1} \left| T_i - t_{i,j} \right|^+$$



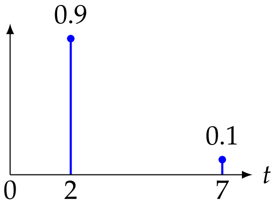
Note: We focus on expected values $\mathbb{E}[T]$ and $\mathbb{E}[C]$

Replication helps!

$$X = \begin{cases} 2 & \text{w.p. } 0.9 \\ 7 & \text{w.p. } 0.1 \end{cases}$$

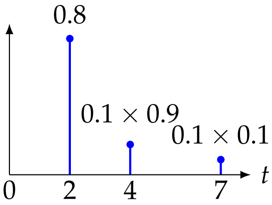
No replication:

$$\mathbb{E}[T] = 2.5$$
$$\mathbb{E}[C] = \mathbb{E}[T] = 2.5$$



Replicate task at $t_2 = 2$:

$$\mathbb{E}[T] = 2.23$$
$$\mathbb{E}[C] = 2.46$$



Discrete random variables [**W.**, Joshi & Wornell, SIGMETRICS'14]

- Arise directly from estimation (quantiles)
- Offers more flexible modeling

Continuous random variables

- Pareto, Exponential
- Analysis: an important class of policies

Execution time modeling

Discrete random variables [**W.**, Joshi & Wornell, SIGMETRICS'14]

- Arise directly from estimation (quantiles)
- Offers more flexible modeling

in thesis

Continuous random variables

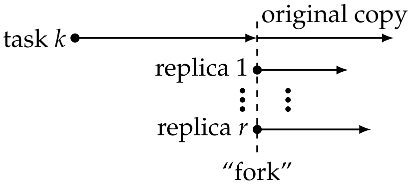
- Pareto, Exponential
- Analysis: an important class of policies

this talk ✓

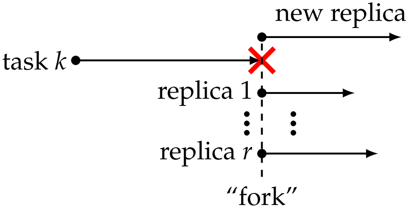
Single-fork policy

- Run n tasks in parallel initially
- When there is p fraction of the tasks left, replicate the unfinished tasks r times
  - Let all the unfinished tasks keep running
  - Relaunch all the unfinished tasks

Without relaunching



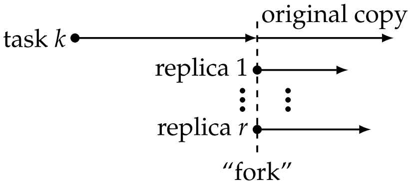
With relaunching



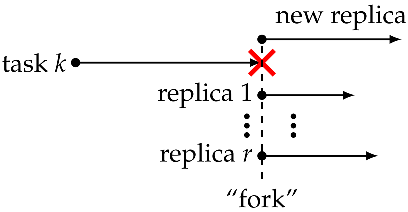
Single-fork policy

- Run n tasks in parallel initially
- When there is p fraction of the tasks left, replicate the unfinished tasks r times
  - Let all the unfinished tasks keep running
  - Relaunch all the unfinished tasks

Without relaunching



With relaunching



Note:

- after replication, $r + 1$ tasks in total
- when $r = 0$: only the relaunching case is interesting

Order statistics

Given random variables $X_1, X_2, \ldots, X_n$

$$X_{\min} = X_{1:n} \leq X_{2:n} \leq \ldots \leq X_{n:n} = X_{\max}$$

Order statistics

Given random variables $X_1, X_2, \ldots, X_n$

$$X_{\min} = X_{1:n} \leq X_{2:n} \leq \ldots \leq X_{n:n} = X_{\max}$$

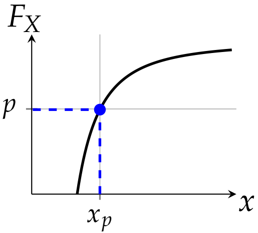
Latency

- Replicated tasks: new execution time distribution F_Y
  - $F_Y = g(F_X, r, \text{relaunch or not})$
- Replicate for the last p fraction of the tasks:

$$T = \underbrace{X_{(1-p)n:n}}_{\text{before forking}} + \underbrace{Y_{pn:pn}}_{\text{after forking}}$$

p-quantile of a distribution F_X

$$x_p \triangleq F_X^{-1}(p)$$



As $n \to \infty$,

$$X_{pn:n} \to \mathsf{N}\left(x_p, \frac{1}{n}\frac{p(1-p)}{f_X^2(x_p)}\right)$$

Time before forking:

$$\mathbb{E}\left[X_{(1-p)n:n}\right] = x_{1-p} \triangleq F_X^{-1}(1-p)$$

Theorem (Fisher-Tippett-Gnedenko theorem)

Given $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} F_X$*, if there exist sequences of constants* $a_n > 0$ *and* $b_n \in \mathbb{R}$ *such that*

$$\mathbb{P}\left[\frac{X_{n:n} - b_n}{a_n} \leq z\right] \rightarrow G(z)$$

as $n \rightarrow \infty$ *and* G *is a non-degenerate distribution, then* G *belongs to one of the following families:*

$$\begin{array}{ll}
\textit{Gumbel law} & G(z) = \exp\left\{-\exp\left(-z\right)\right\} \textit{ for } z \in \mathbb{R}, \\
\textit{Fréchet law} & G(z) = \begin{cases} 0 & z \leq 0 \\ \exp\left\{-z^{-\alpha}\right\} & z > 0 \end{cases}, \\
\textit{Weibull law} & G(z) = \begin{cases} \exp\left\{-\left(-z\right)^{\alpha}\right\} & z < 0 \\ 1 & z \geq 0 \end{cases},
\end{array}$$

where $\alpha > 0$*.*

Theorem (Fisher-Tippett-Gnedenko theorem)

Given $X_1, X_2, \ldots, X_n \overset{i.i.d.}{\sim} F_X$*, if there exist sequences of constants* $a_n > 0$ *and* $b_n \in \mathbb{R}$ *such that*

$$\mathbb{P}\left[\frac{X_{n:n} - b_n}{a_n} \leq z\right] \to G(z)$$

as n ... *e*

foll

Given F_X, the distribution of $X_{n:n}$ can be characterized as $n \to \infty$.

Key: tail behavior of $1 - F_X$

Also applicable to $X_{1:n}$

Weibull law $G(z) = \begin{cases} \ldots \\ 1 & z \geq 0 \end{cases},$

where $\alpha > 0$.

execution time before forking: F_X $\xrightarrow{\text{central value theorem}}$ time before forking: $X_{(1-p)n:n} \to F_X^{-1}(1-p)$

execution time before forking: F_X $\xrightarrow{\text{extreme value theorem}}$ $X_{n:n} \to G_X$, $X_{1:r} \to G'_X$

$Y = g(F_X, r, \text{relaunch or not})$

execution time after forking: F_Y $\xrightarrow{\text{extreme value theorem}}$ time after forking: $Y_{pn:pn} \to G_Y$

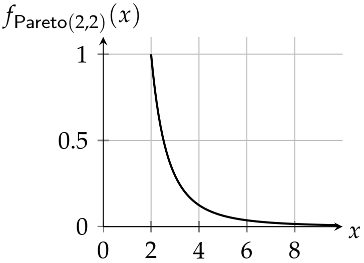
$$T = X_{(1-p)n:n} + Y_{pn:pn}$$

Cost can be analyzed similarly.

Execution time: Pareto distribution

Pareto (α, x_m).

$$F_X(x; \alpha, x_m) \triangleq \begin{cases} 1 - \left(\frac{x_m}{x}\right)^{\alpha} & x \geq x_m \\ 0 & x < x_m \end{cases}$$



- heavy-tail distribution
- observed in data centers [Reiss *et al.*, 2012].

Extremes

$$X_{1:n} \sim \text{Pareto}(n\alpha, x_m)$$

$$\frac{X_{n:n}}{x_m n^{1/\alpha}} \sim \text{Fréchet}$$

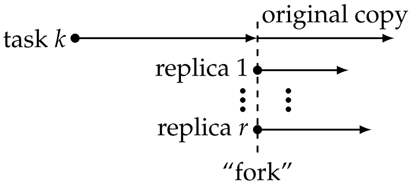
$$\mathbb{E}[X_{n:n}] = x_m n^{1/\alpha} \Gamma(1 - 1/\alpha) \propto n^{1/\alpha} \cdot \mathbb{E}[X]$$

Recall: single-fork policy

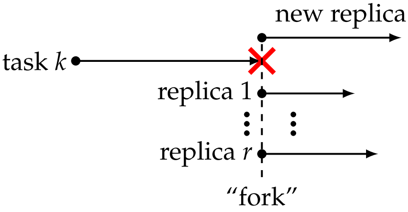
Parameters

- number of tasks: n
- fraction to replicate: p
- additional replicas: r

Without relaunching

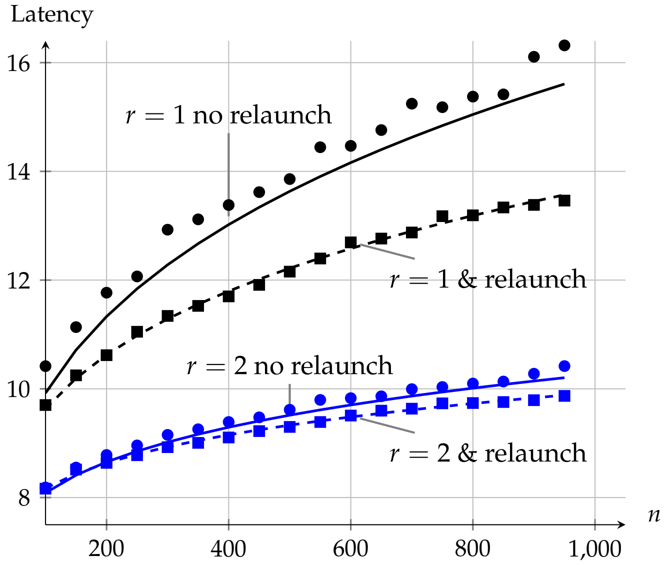


With relaunching



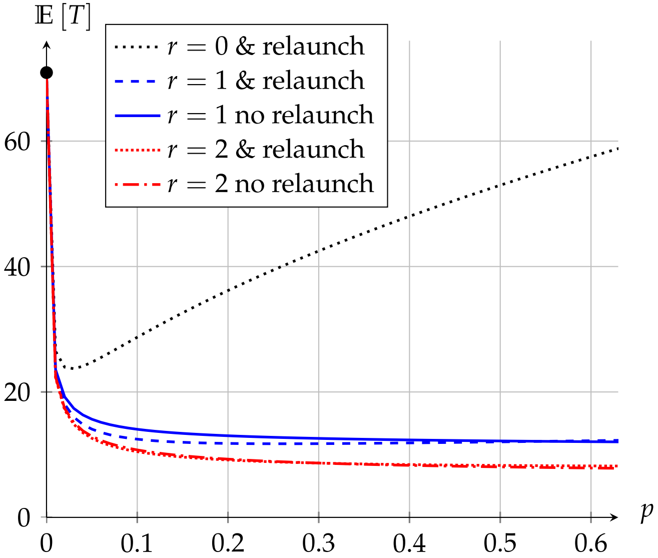
Asymptotic characterization accurate in finite regime

$X \sim$ Pareto $(2, 2)$ and replication fraction $p = 0.2$



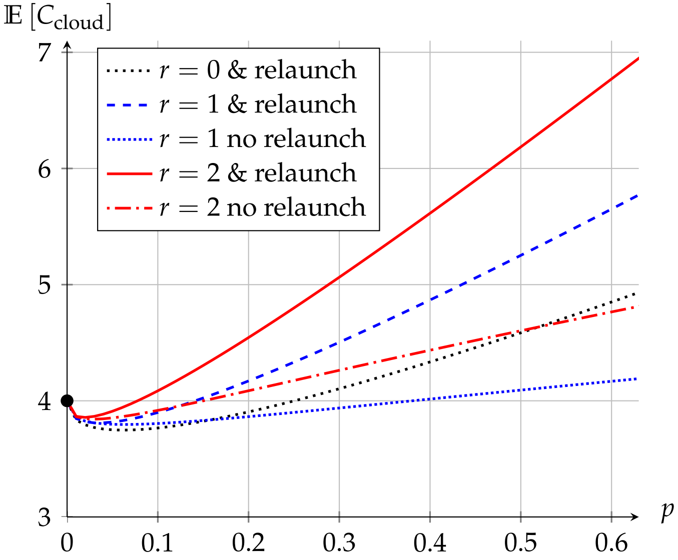
Latency & Cost vs. replication fraction p

$X \sim \text{Pareto}\,(2,2)$ with $n = 400$



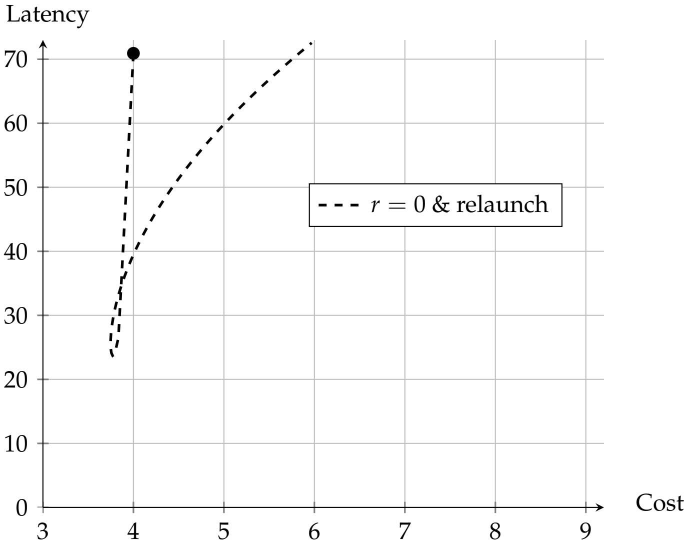
Latency & Cost vs. replication fraction p

$X \sim \text{Pareto}(2, 2)$ with $n = 400$



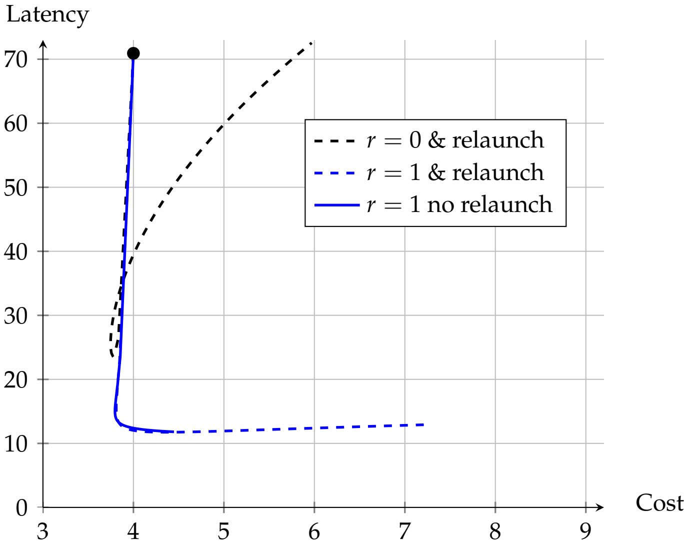
Latency-cost trade-off

$X \sim \text{Pareto}(2, 2)$ with $n = 400$



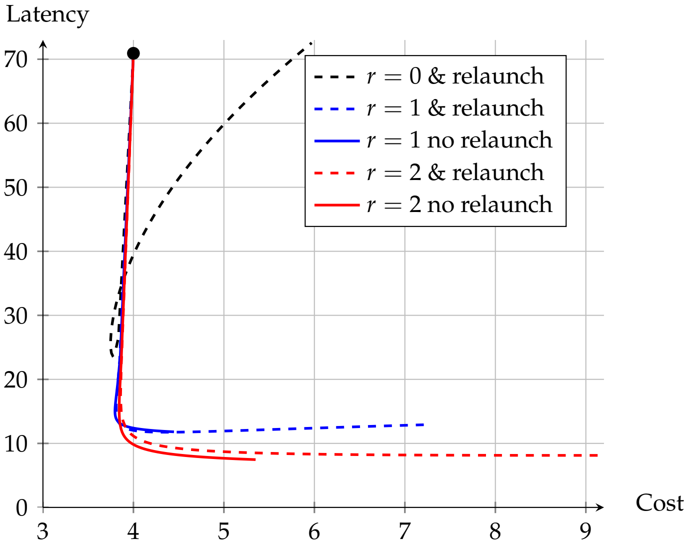
Latency-cost trade-off

$X \sim \text{Pareto}\,(2, 2)$ with $n = 400$



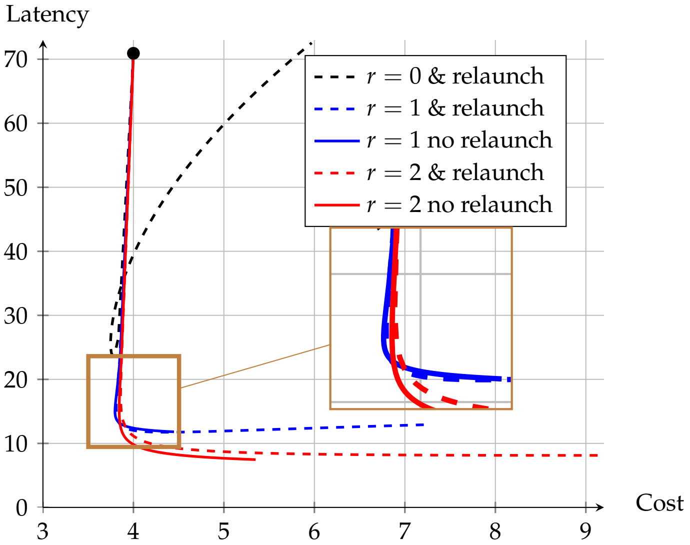
Latency-cost trade-off

$X \sim \text{Pareto}\,(2,2)$ with $n = 400$



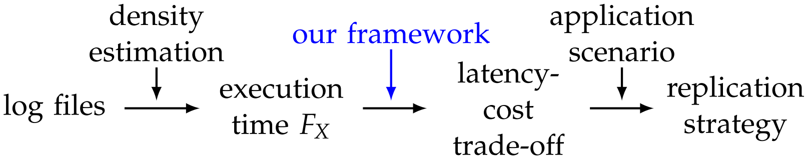
Latency-cost trade-off

$X \sim \text{Pareto}\,(2, 2)$ with $n = 400$



Recap

- A framework for analyzing single-fork policies
  - Can be extended to multi-fork
- Applicable to any distributions that can be analyzed via EVT
  - Pareto, Exponential, Erlang, etc..



Computing with unreliable resources: scheduling

mathematical abstraction → asymptotic analysis → performance trade-offs → design guidance

order statistics, extreme value theory

latency vs. cost

tail behavior of execution time distribution

Crowd-based ranking via noisy comparisons

Rank by score assignment

- Examples: `IMDb.com`, `Yelp.com`

Rank by pairwise comparison

- Examples: admission/recruiting, knockout stage of tournaments
- Challenges:
  - human comparisons are noisy: answer flipped w.p. ε
  - human comparisons are expensive (economic cost, time, ...)

What is the fundamental trade-off between ranking accuracy and the number of comparisons?

approximate sorting via (noisy) comparisons

Information-theoretic lower bounds on #comparisons

Distortion measure

ℓ_1 distance of permutations $\quad d_{\ell_1}(\pi_1, \pi_2) \triangleq \sum_{i=1}^{n} |\pi_1(i) - \pi_2(i)|$

To achieve distortion $D \leq \Theta\left(n^{1+\delta}\right)$

- Approximate sorting with noiseless comparisons
  - [**W.**, Mazumdar & Wornell, ISIT'14]: at least
    $$(1-\delta) n \log n \quad \text{comparisons}$$
  - Tight: the *multiple selection* algorithm in [Kaligosi 2005] achieves this bound

Information-theoretic lower bounds on #comparisons

Distortion measure

ℓ_1 distance of permutations $\quad d_{\ell_1}(\pi_1, \pi_2) \triangleq \sum_{i=1}^{n} |\pi_1(i) - \pi_2(i)|$

To achieve distortion $D \leq \Theta\left(n^{1+\delta}\right)$

- Approximate sorting with noiseless comparisons
  - [**W.**, Mazumdar & Wornell, ISIT'14]: at least
    $$(1-\delta) n \log n \quad \text{comparisons}$$
  - Tight: the *multiple selection* algorithm in [Kaligosi 2005] achieves this bound
- Approximate sorting with noisy comparisons
  - [**W.**, Mazumdar & Wornell, ISIT'14]: at least
    $$\frac{(1-\delta)}{1 - H_b(\varepsilon)} n \log n \quad \text{comparisons}$$
  - Existing algorithms only known to be $O(n \log n)$

More distortion measures [**W.**, Mazumdar & Wornell, ISIT'14]

- Kendall tau distance, Chebyshev distance, ...
- Relationships among distortion measures

Other distributional model

- Mallows distributional model

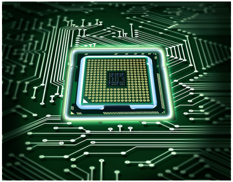
mathematical abstraction → asymptotic analysis → performance trade-offs → design guidance

rate-distortion theory, combinatorics

accuracy vs. #comparisons

error probability of noisy comparisons

Computing with unreliable resources



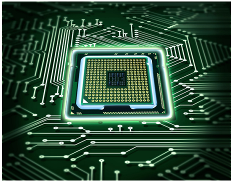
mathematical abstraction → asymptotic analysis → performance trade-offs → design guidance

suitable for large scale systems!

reliability vs. resource usage

statistical properties of unreliability

Computing with unreliable resources



mathematical abstraction → asymptotic analysis → performance trade-offs → design guidance

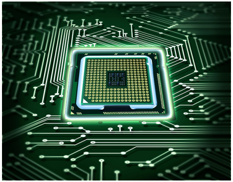
↑ suitable for large scale systems!

reliability vs. resource usage

statistical properties of unreliability

A unified "coding theory" for these computing problems?

Computing with unreliable resources



mathematical abstraction → asymptotic analysis → performance trade-offs → design guidance

suitable for large scale systems!

reliability vs. resource usage

statistical properties of unreliability

A unified "coding theory" for these computing problems?

A journey of a thousand miles begins with a single step.

— Lao Tzu

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat

This thesis is impossible without . . .

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …
  - friends

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …
  - friends
  - colleagues

This thesis is impossible without . . .

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, . . .
    - friends
    - colleagues
    - constructive criticisers

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …
  - friends
  - colleagues
  - constructive criticisers
  - poker opponents

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …
    - friends
    - colleagues
    - constructive criticisers
    - poker opponents
    - …

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …
  - friends
  - colleagues
  - constructive criticisers
  - poker opponents
  - …
- My parents

This thesis is impossible without …

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, …
  - friends
  - colleagues
  - constructive criticisers
  - poker opponents
  - …
- My parents
- Hengni

This thesis is impossible without . . .

- Thesis committee: Greg, Yury and Devavrat
- Collaborators: Gauri Joshi, Arya Mazumdar
- The community: SiA, RLE, LIDS, CSAIL, MTL, EECS, GSA, GSC, MIT, . . .
  - friends
  - colleagues
  - constructive criticisers
  - poker opponents
  - . . .
- My parents
- Hengni

Thank you!