Linear Programming I

David Woodruff

Outline

- Definition of linear programming and examples
- A linear program to solve max flow and min-cost max flow
- A linear program to solve minimax-optimal strategies in games

Example

- There are 168 hours in a week. Want to allocate our time between
  - studying (S)
  - going to parties (P)
  - everything else (E)
- To survive: $\mathrm{E} \geq 56$
- For sanity: $\mathrm{P} + \mathrm{E} \geq 70$
- To pass courses: $\mathrm{S} \geq 60$
- If party a lot, need to study or eat more: $2S + E - 3P \geq 150$
- Is there a *feasible* solution? Yes, S = 80, P = 20, E = 68
- Happiness is 2P + E. Find a feasible solution maximizing this *objective function*

Linear Program

- This is called a *linear program (LP)*
- All constraints are linear in our variables
- Objective function is linear
- Don't allow $\mathrm{S} \cdot \mathrm{E} \geq 100$, that's a polynomial program. Much harder.

Formal Definition

- Given:
  - n variables $x_1, \dots, x_n$
  - m linear inequalities in these variables
    - E.g., $3x_1 + 4x_2 \leq 6$, $0 \leq x_1$, $x_1 \leq 3$

- Goal:
  - Find values for the x_i's that satisfy constraints and maximize objective
  - In the feasibility problem just satisfy the constraints

  - What would happen if we allowed strict inequalities $x_1 < 3$?
  - max x_1

Time Allocation Problem

- Variables: S, P, E
- Objective: Maximize 2P + E subject to
- Constraints: S + P + E = 168

$E \geq 56$

$S \geq 60$

$2S + E - 3P \geq 150$

$P + E \geq 70$

$P \geq 0$

Operations Research Problem

	labor	materials	pollution
plant 1	2	3	15
plant 2	3	4	10
plant 3	4	5	9
plant 4	5	6	7

What are the variables?

x_1, x_2, x_3, x_4 denote the number of cars at plant i

What's our objective?

maximize $x_1 + x_2 + x_3 + x_4$

- Required to make at least 400 cars at plant 3
- Have 3300 hours of labor and 4000 units of material
- At most 12000 units of pollution
- Maximize number of cars made

	labor	materials	pollution
plant 1	2	3	15
plant 2	3	4	10
plant 3	4	5	9
plant 4	5	6	7

Make at least 400 cars at plant 3
3300 hours of labor and 4000 units of material
At most 12000 units of pollution
Maximize number of cars made

Note: linear programming does not give an integral solution (NP-hard)

What are the variables?

x_1, x_2, x_3, x_4 denote the number of cars at plant i

What's our objective?

maximize $x_1 + x_2 + x_3 + x_4$

Constraints: $x_i \geq 0$ for all i

$$x_3 \geq 400$$
$$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 3300$$
$$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 4000$$
$$15x_1 + 10x_2 + 9x_3 + 7x_4 \leq 12000$$

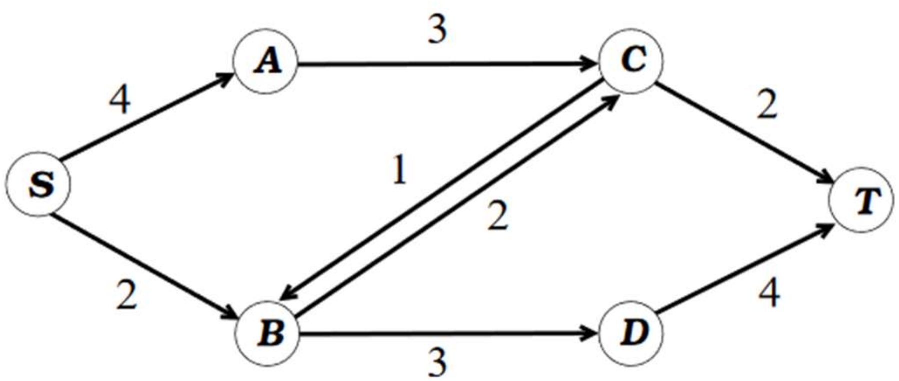
Modeling Network Flow

Variables: f_{uv} for each edge (u,v), representing positive flow

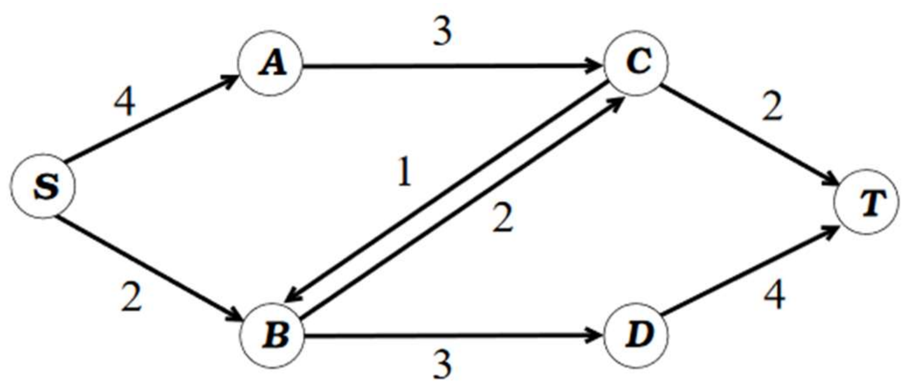
Objective: maximize $\sum_u f_{ut} - \sum_u f_{tu}$

Constraints: For all edges (u,v) $0 \leq f_{uv} \leq c(u,v)$ (capacity constraints)

For all v $\notin \{s,t\}, \sum_u f_{uv} = \sum_u f_{vu}$ (flow conservation)



Modeling Network Flow



In this case, our LP is: maximize $f_{ct} + f_{dt}$ subject to the constraints:

$$0 \leq f_{sa} \leq 4,\ 0 \leq f_{ac} \leq 3, \text{ etc.}$$

$$f_{sa} = f_{ac},\ f_{sb} + f_{cb} = f_{bc} + f_{bd},\ f_{ac} + f_{bc} = f_{cb} + f_{ct},\ f_{bd} = f_{dt}.$$

Min Cost Max Flow

- Edge (u,v) has a capacity c(u,v) and a cost w(u,v)
- Find a max s-t flow of least total cost, where the cost of flow f is

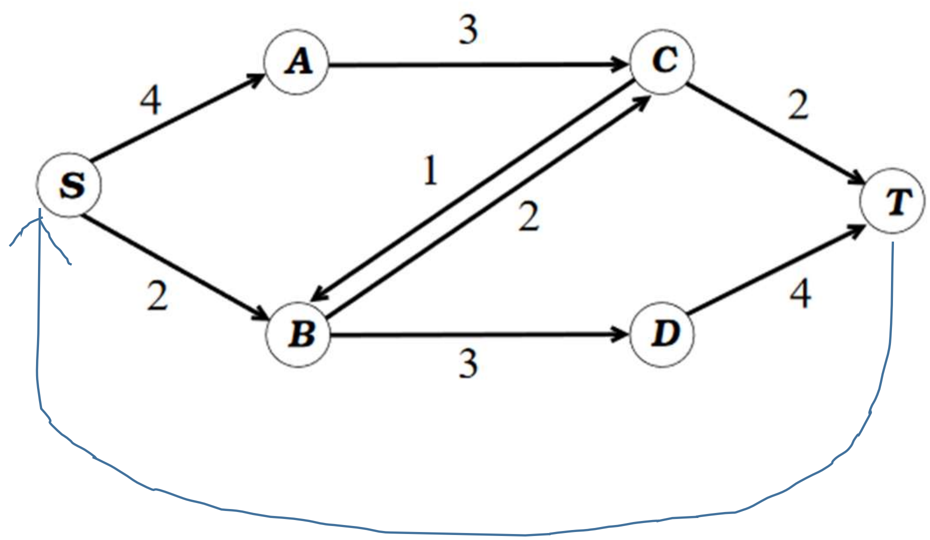
$$\sum_{(u,v)\in E} w(u,v) f_{uv}$$

- *How to solve this?*
- Solution 1: Solve for a maximum flow f
  Add a constraint that flow must equal the flow of f
  Minimize $\sum_{(u,v)\in E} w(u,v) f_{uv}$ also subject to original constraints
- Solution 2: Add an edge (t,s) of infinite capacity and very negative cost
  Minimizing cost automatically maximizes flow

Min Cost Max Flow

$$\min \sum_{(u,v)\in E} w(u,v) f_{uv}$$

$f_{ts} = f_{sa} + f_{sb}$



$f_{ct} + f_{dt} = f_{ts}$

∞

$w(t,s) = -\infty$

Zero Sum Games

Row payoffs:

20	−10	5
5	10	−10
−5	0	10

- Given a zero-sum game with n rows and n columns, compute a minimax optimal strategy for row player

- *What are the variables?*
  - Probabilities $p_1, \ldots, p_n$ on our actions
  - Linear constraints: $\sum_{i=1,\ldots,n} p_i = 1$ and $p_i \geq 0$ for all i
  - Maximize the minimum expected payoff, over all column pure strategies

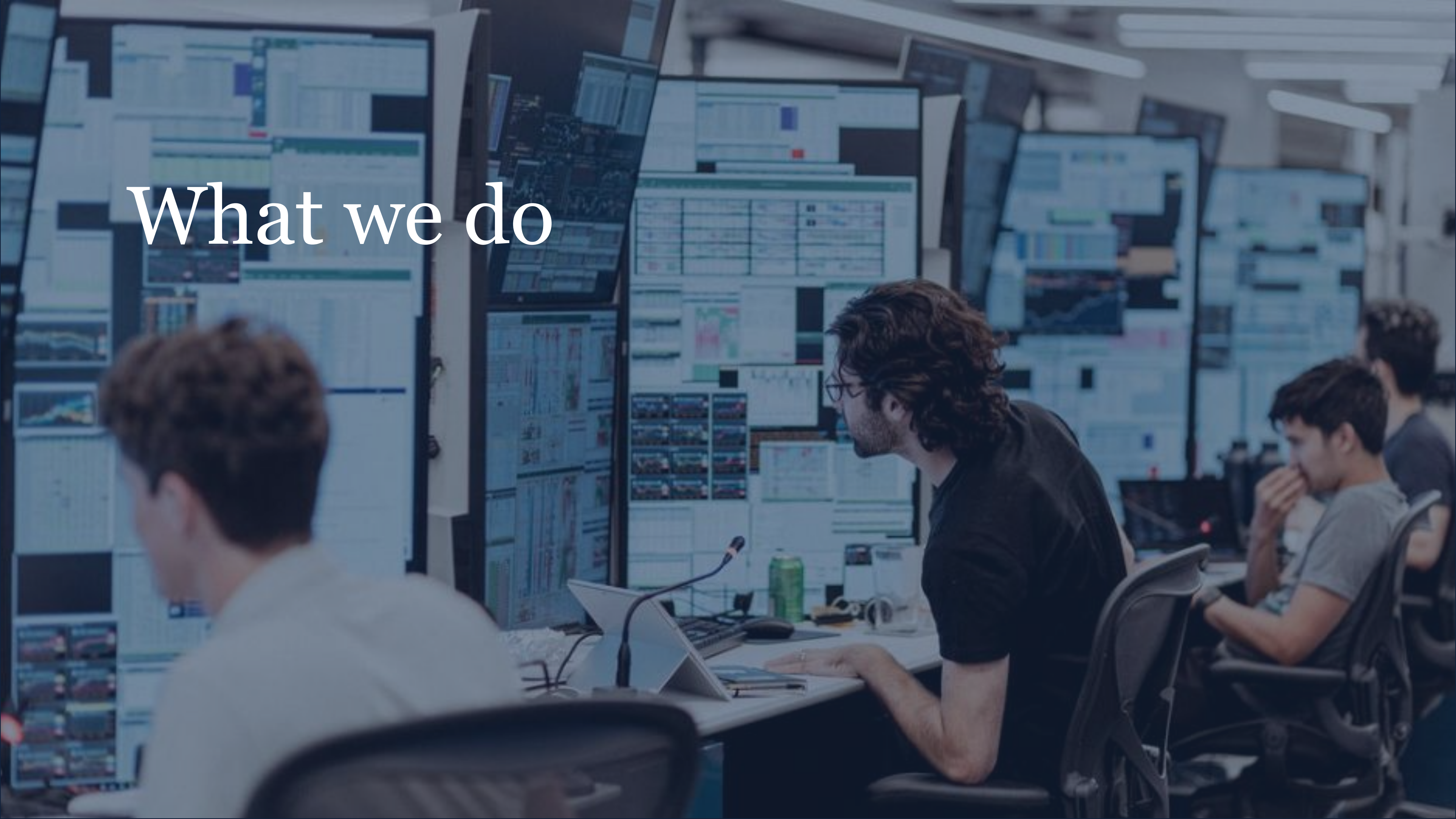
- *How to maximize a minimum with a linear program?*
- Create new "dummy variable" v to represent minimum

Zero Sum Games

- $R_{i,j}$ represents payoff to row player with row player action i and column player action j
- Variables: $p_1, \ldots, p_n$ and v
- Objective: maximize v
- Constraints:
  - $p_i \geq 0$ for all i, and $\sum_i p_i = 1$
  - For all columns j, $\sum_i p_i R_{ij} \geq v$

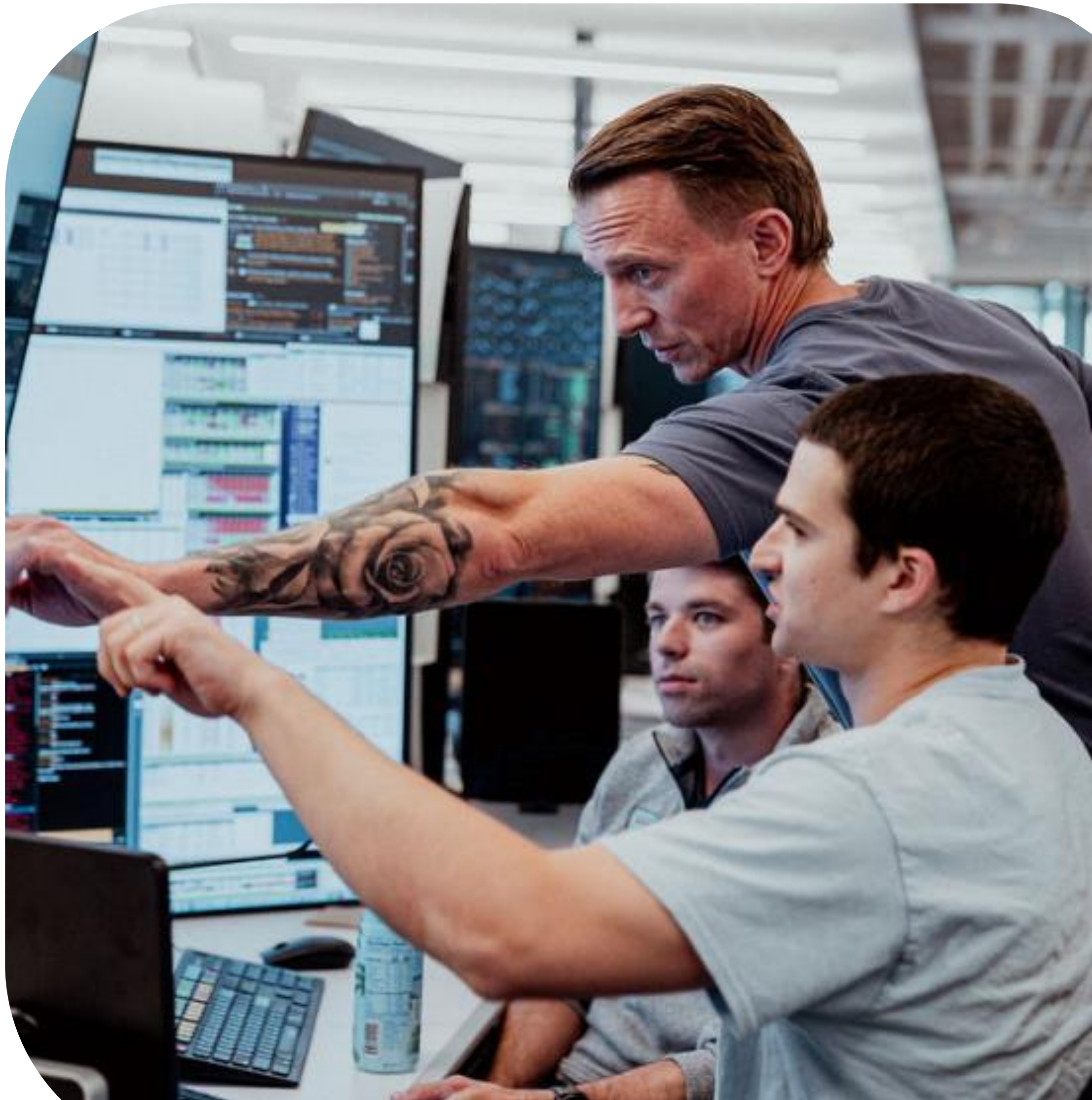
Optiver

What we do



Driven by technology, powered by people

Our mission	We improve the markets
Our culture	We do our best work together
Cutting-edge tech	Best-in-class systems developed in-house
Your growth	There’s no limit to how quickly you can grow your career at Optiver



Opportunities

Software Engineer

- Create, maintain and improve trading systems
- Collaborate cross-functionally
- Solve for complex challenges that have a direct impact on the business

FPGA Engineer

- Accelerate network infrastructure and trading system components
- Identify optimization opportunities
- Utilize or build cutting-edge server hardware

Quantitative Trader

- Manage a trading book as a team
- Continuously refine and improve strategies
- Develop innovative statistical models

Quantitative Researcher

- Use statistical models and machine learning to develop trading algorithms
- Collaborate with a dynamic team
- Build stochastic models to determine the fair value of financial derivatives

Optiver

Elan Biswas

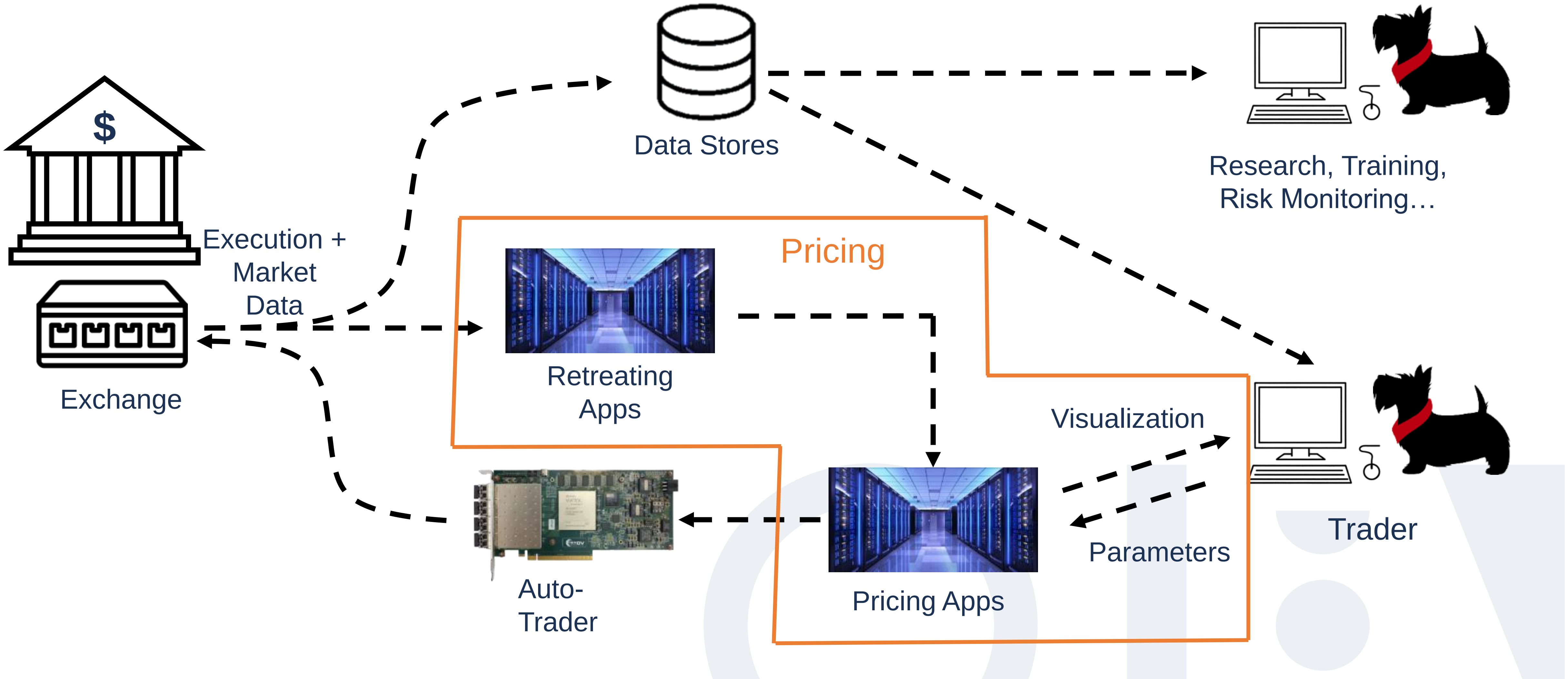
Software Engineer

Career
- **Aug 2023 - Present:** Software Engineer at Optiver
- **Summer 2022:** SWE Intern at Optiver
- **Fall 2021:** SWE Intern at Amazon AWS
- **Summer 2021:** SWE Intern at Pinterest

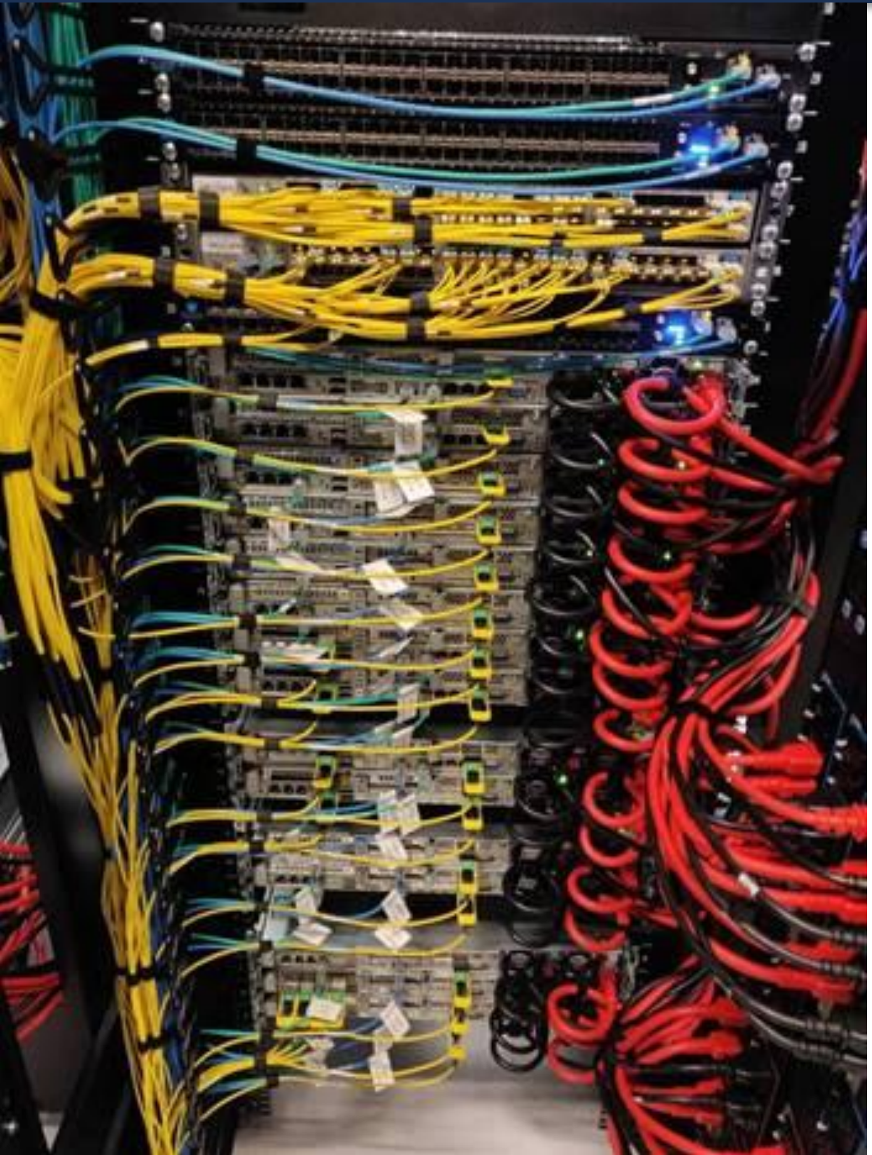
Education
- Carnegie Mellon University - Class of 2023
- Major: Computer Science
- Concentration: Computer Systems

What I do at Optiver
- Pricing Team

From the Trader to the Auto-Trader

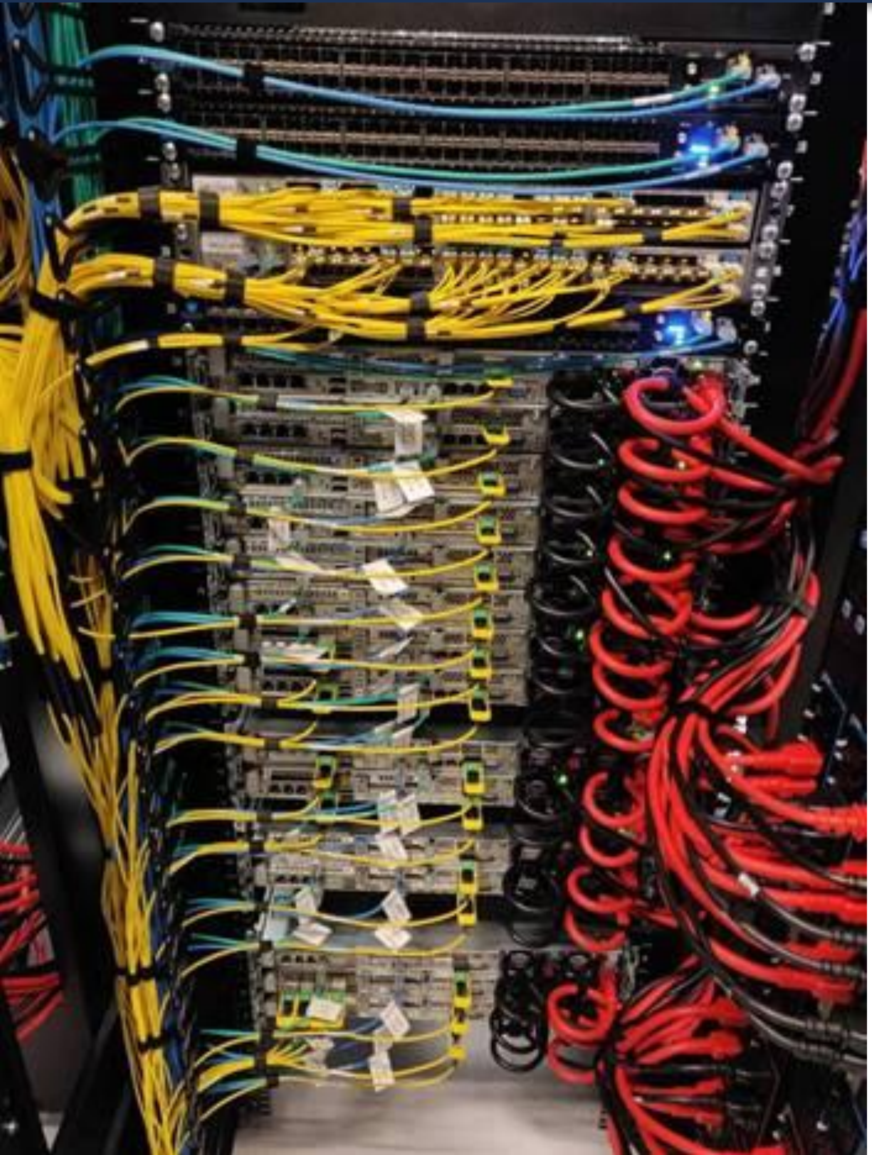


Where do all these apps run?



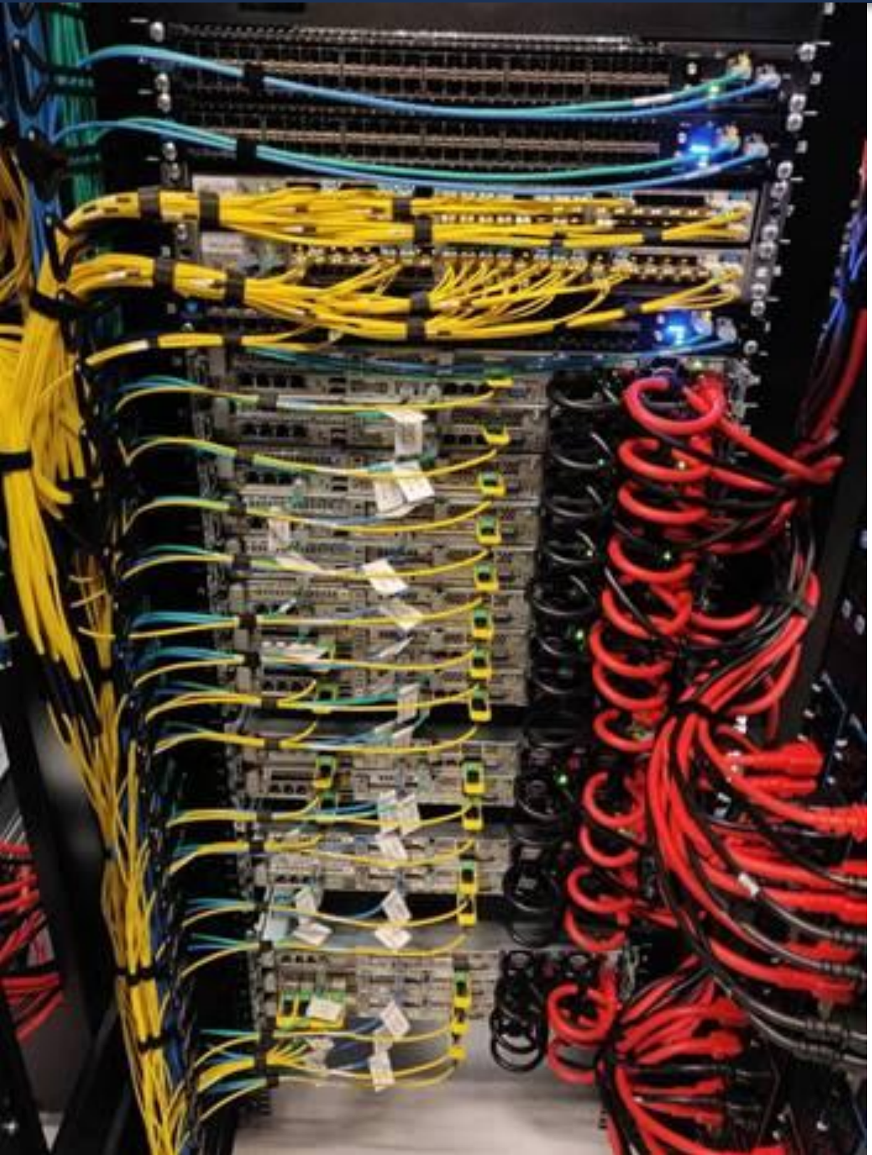
- …on computers, or **hosts**
- **Colocation** – shared server room close to exchange
- Space is limited and costly to rent
- We **partition** the processes based on their configurations
  - e.g. by **desk** – group of products traded by 1 team

Assigning Processes to Hosts



- Each host has a limited # of cores and GBs of RAM
- Each process requires use of some cores and RAM
- **Goal**: isolate **partitions** of the stack
  - Unrelated **desks** ideally shouldn't share a host
- Could we formulate this as an LP?
- Yes! Specifically, we define a **0-1 Integer LP**
  - All variables are binary
  - NP-Complete, but many heuristic solvers exist

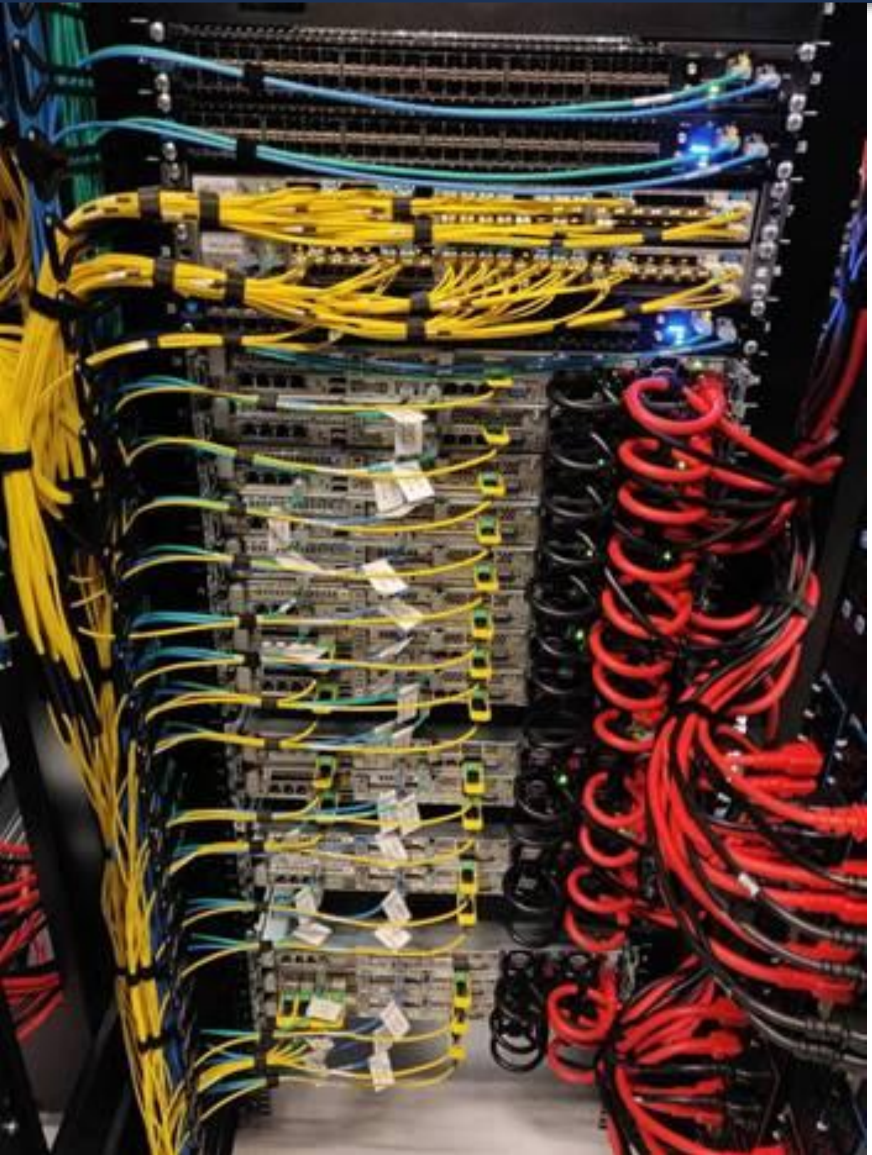
Decision Variables



- $p_{ih} = \begin{cases} 1, & \text{proc. } i \text{ assigned to host } h \\ 0, & OTW \end{cases}$

- $d_{ih} = \begin{cases} 1, & \text{desk } i \text{ configured on host } h \\ 0, & OTW \end{cases}$

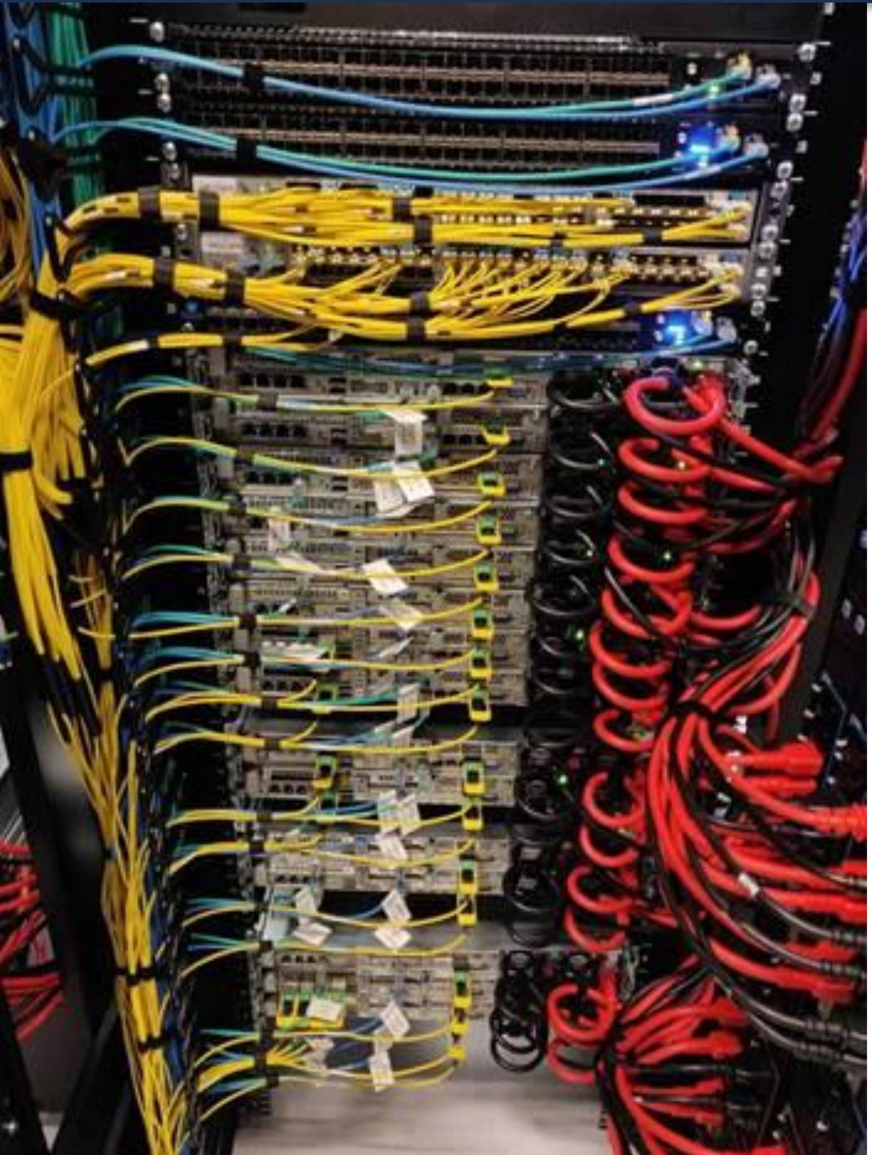
Objective Function



- We want to reduce the surface area of each desk
  - Minimize # of desks on a given host
- Recall d_{ih} indicates desk *i* is configured on host *h*
- We want to **minimize**

$$\sum_{\substack{i \in Desks \\ h \in Hosts}} d_{ih}$$

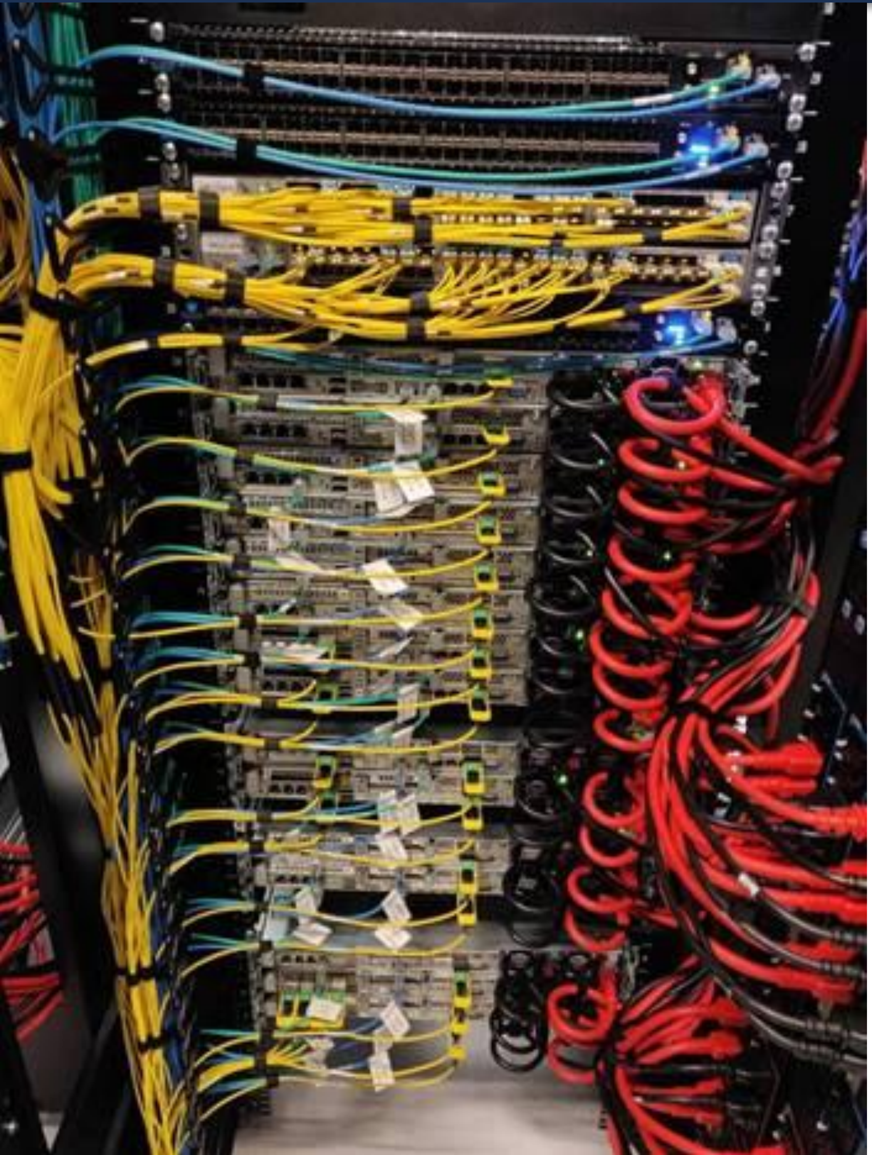
Process Constraints



- Each process must be assigned to exactly 1 host
- For each process *i*

$$\sum_{h \in Hosts} p_{ih} = 1$$

Host Constraints

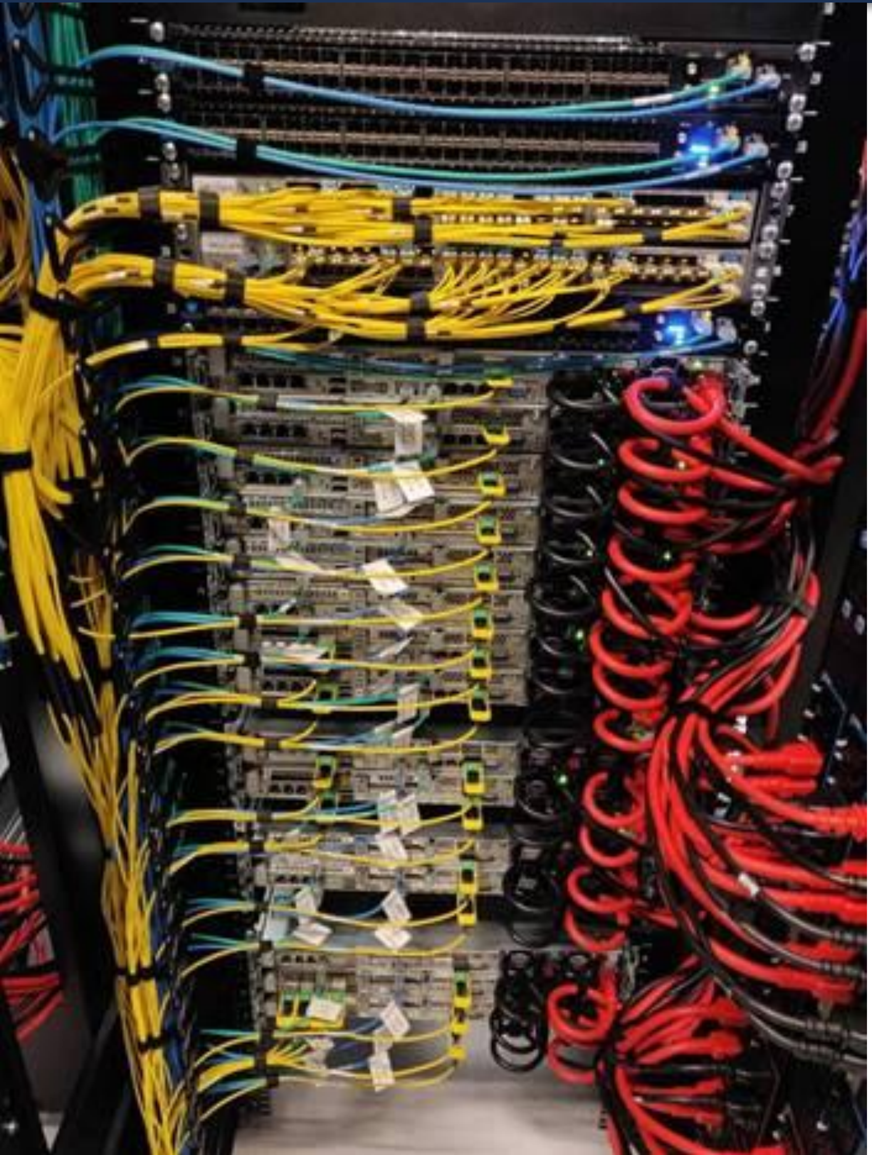


- Each host *h* has m_h GB of RAM and c_h cores
- Each process *i* has a resource usage threshold
    - x_i GBs of RAM
    - y_i cores
- For each host *h*:

$$\sum_{i \in Processes} x_i * p_{ih} \leq m_h$$

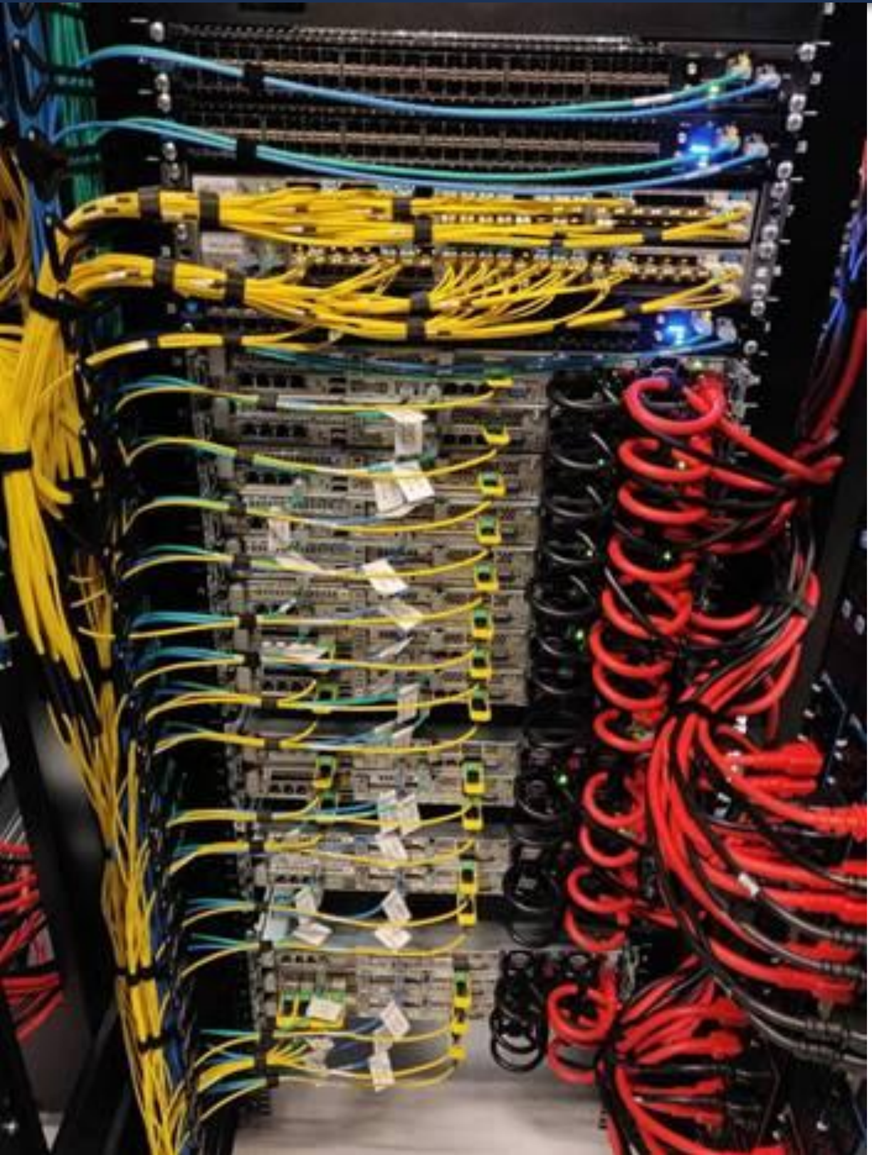
$$\sum_{i \in Processes} y_i * p_{ih} \leq c_h$$

Desk Constraints



- $d_{ih} == 1 \Leftrightarrow$ host *h* has desk *i* configured
- Recall that d_{ih} can only be 0 or 1
- How can we enforce this?

Desk Constraints

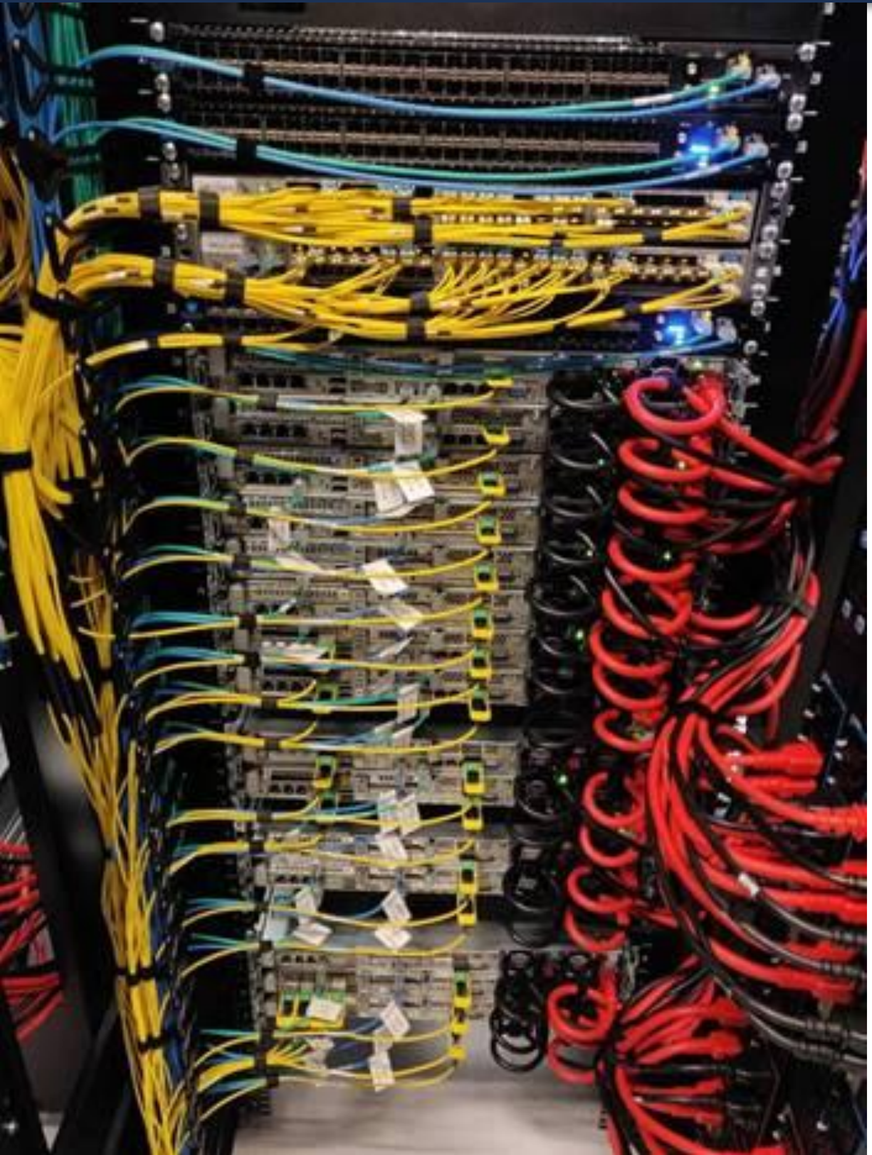


- Let $a_{ij} = \begin{cases} 1, \text{desk } i \text{ is configured for process } j \\ 0, \quad \text{OTW} \end{cases}$
  - Known a priori
- For each desk *i*, host *h*

$$\sum_{j \in Processes} a_{ij} * p_{jh} \leq d_{ih} * |Processes|$$

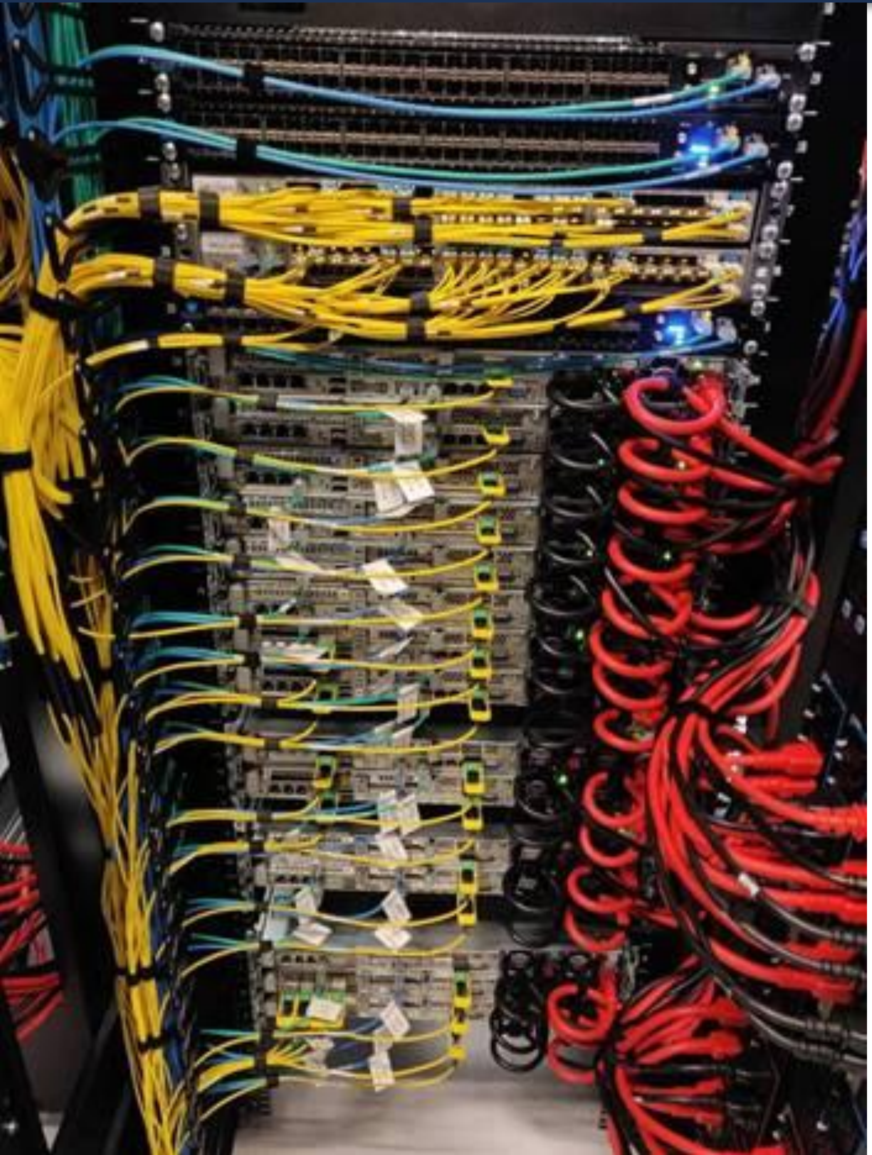
$$\mathrm{d}_{ih} \leq \sum_{j \in Processes} a_{ij} * p_{jh}$$

Load Balancing



- The solution may pile all apps for a desk on one host
- Heuristic: leftover space of the most burdened host
- We could add buffer variables
  - b_m - minimum leftover RAM among all hosts
  - b_c - minimum # of leftover cores among all hosts
- Note: no longer 0-1 ILP, but that's okay

Load Balancing

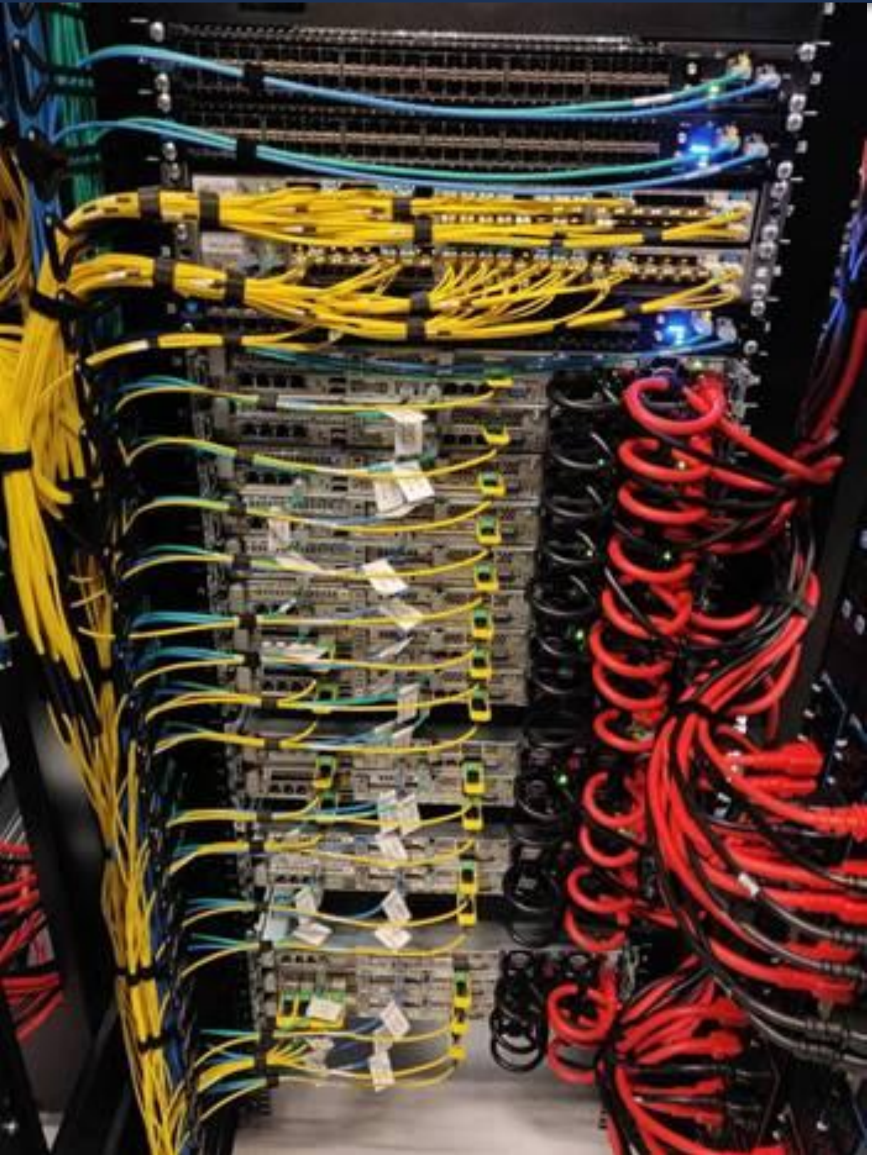


- Recall our objective was to **minimize**

$$\sum_{\substack{i \in Desks \\ h \in Hosts}} d_{ih}$$

- We want to **maximize** the leftover memory and core space of the most burdened host

Load Balancing

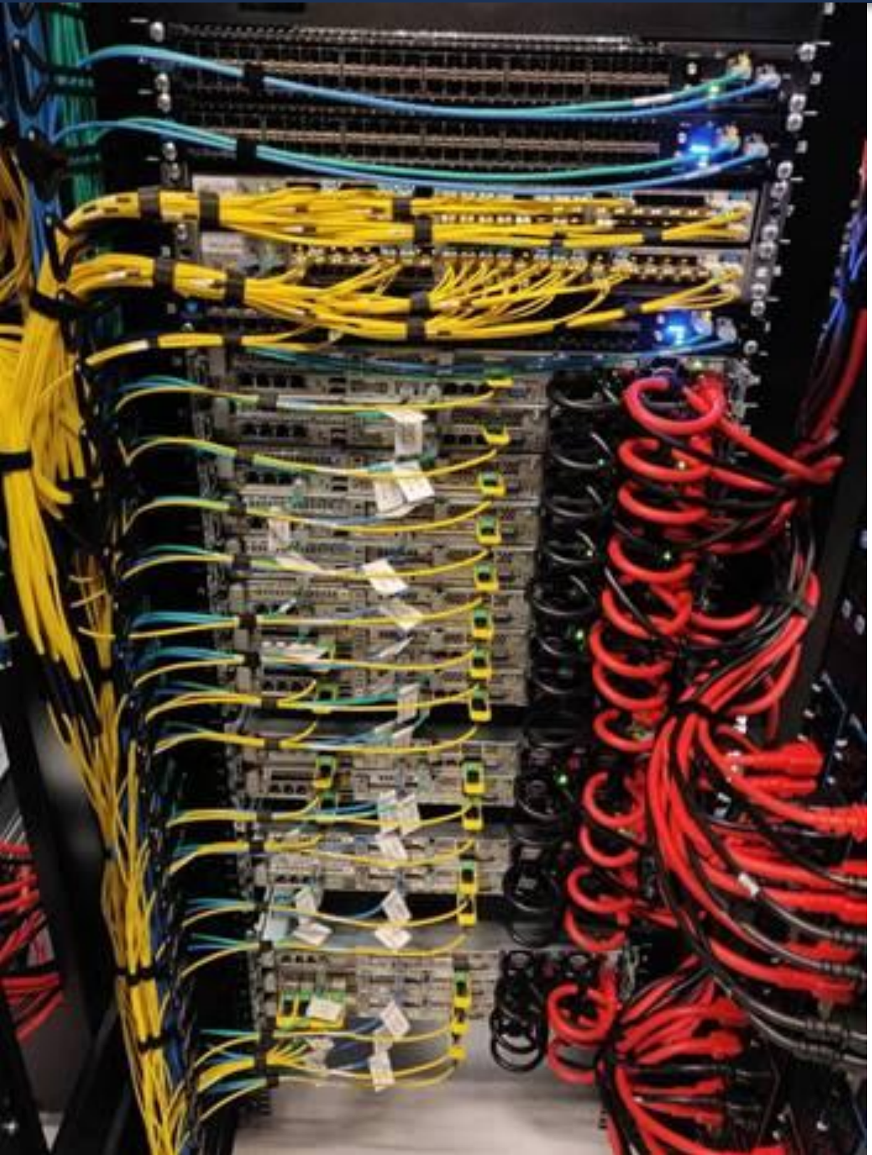


- Recall our objective was to **minimize**

$$\sum_{\substack{i \in Desks \\ h \in Hosts}} d_{ih} - b_m - b_c$$

- We want to **maximize** the leftover memory and core space of the most burdened host

Load Balancing

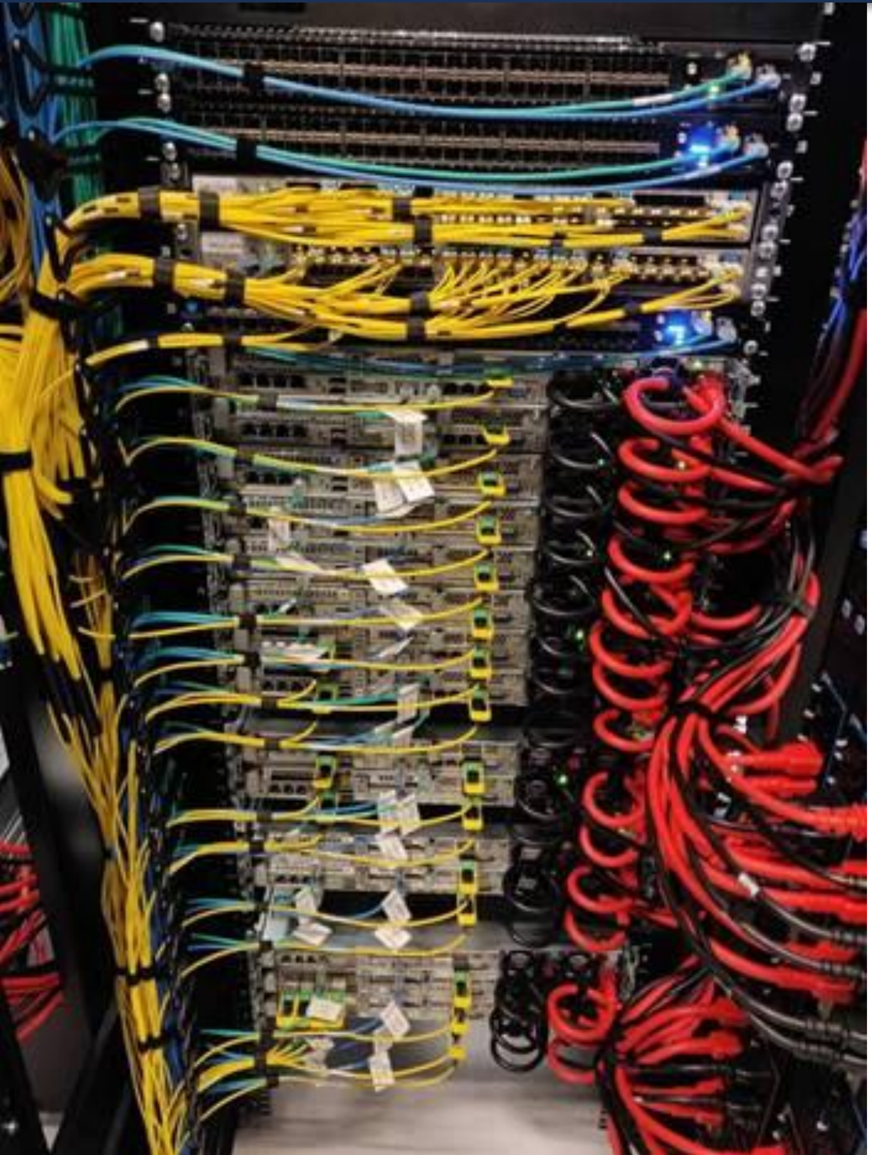


- Need to update constraints to set b_m and b_c
- Recall our host constraints. For each host *h*

$$\sum_{i \in Processes} x_i * p_{ih} \leq m_h$$

$$\sum_{i \in Processes} y_i * p_{ih} \leq n_h$$

Load Balancing

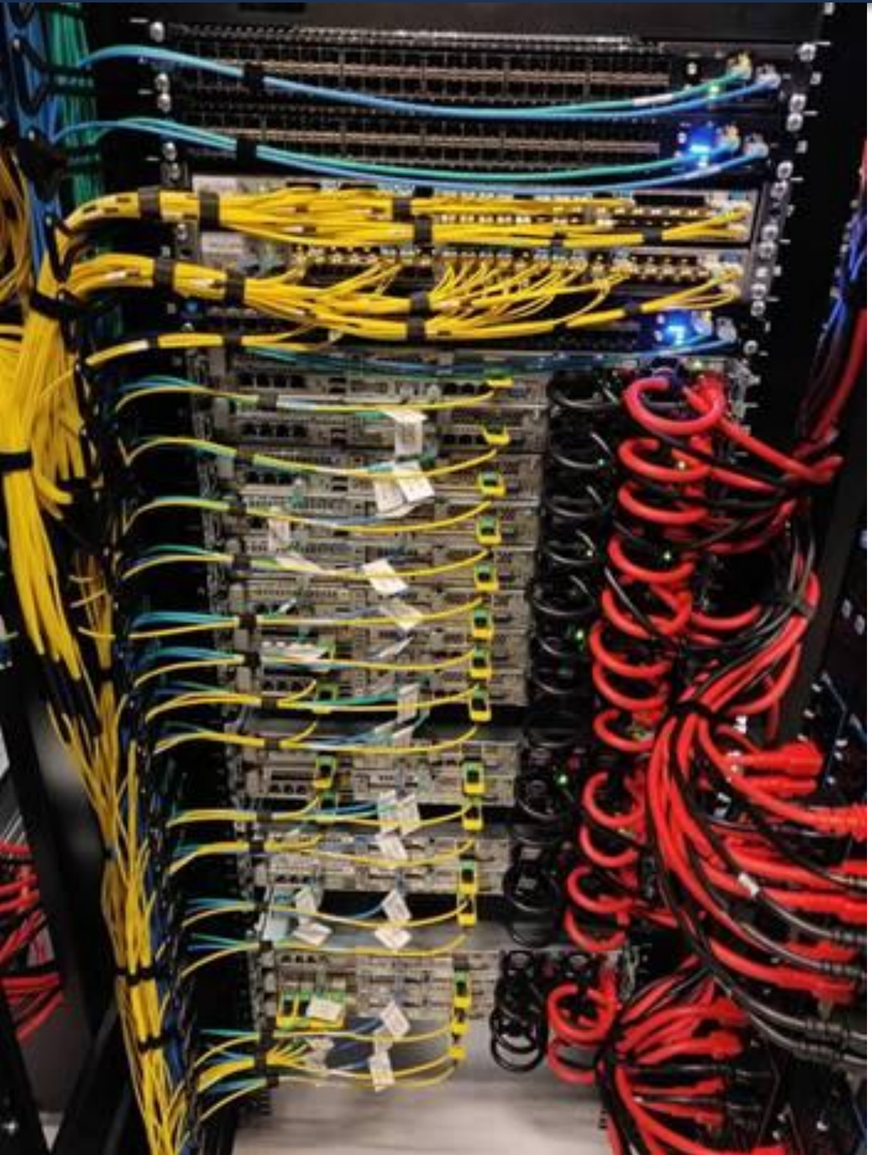


- Need to update constraints to set b_m and b_c
- Recall our host constraints. For each host *h*

$$\sum_{i \in Processes} x_i * p_{ih} \leq m_h - b_m$$

$$\sum_{i \in Processes} y_i * p_{ih} \leq c_h - b_c$$

Putting it All Together



- Minimize

$$\sum_{i,h} d_{ih} - b_m - b_c$$

- Subject to constraints

$$\sum_{h \in Hosts} p_{ih} = 1, \qquad \forall i \in Processes$$

$$\sum_{j \in Procs} a_{ij} p_{jh} \leq d_{ih} * |Processes|, \forall i \in Desks, h \in Hosts$$

$$\mathrm{d}_{ih} \leq \sum_{j \in Procs} a_{ij} p_{jh}\,, \qquad \forall i \in Desks, h \in Hosts$$

$$\sum_{i \in Procs} x_i * p_{ih} \leq m_h - b_m, \qquad \forall h \in Hosts$$

$$\sum_{i \in Procs} y_i * p_{ih} \leq c_h - b_c, \qquad \forall h \in Hosts$$

Questions?