Algorithms for Cloud Computing

Georgios Paschos

amazon

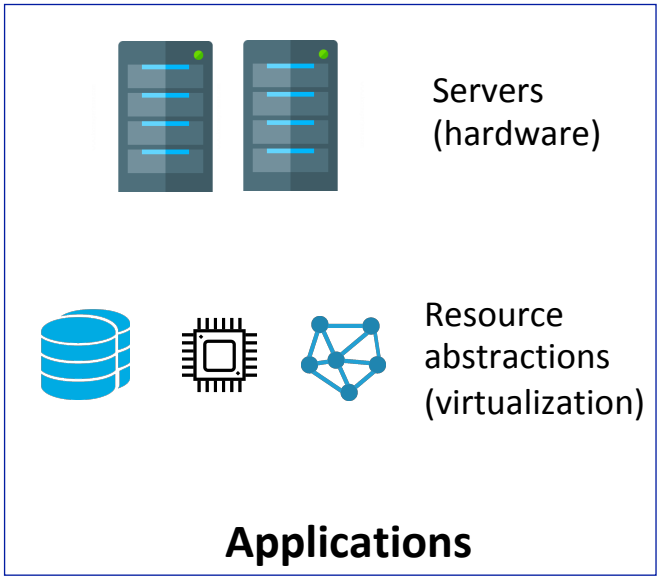
Dr. Georgios Paschos

- '02-'06 Phd - **University of Patras**
- '07-'08 Postdoc - **VTT, Helsinki**
- '08-'12 Researcher & Lecturer - **CERTH + University of Thessaly, Volos**
- '12-'14 Researcher - **MIT, Cambridge**
- '14-'19 Principal Researcher - **Huawei Technologies, Paris**
- '19-now Snr. Manager, Research Science **- Amazon**

amazon

Cloud computing

On-demand availability of computing resources, especially storage and computing power



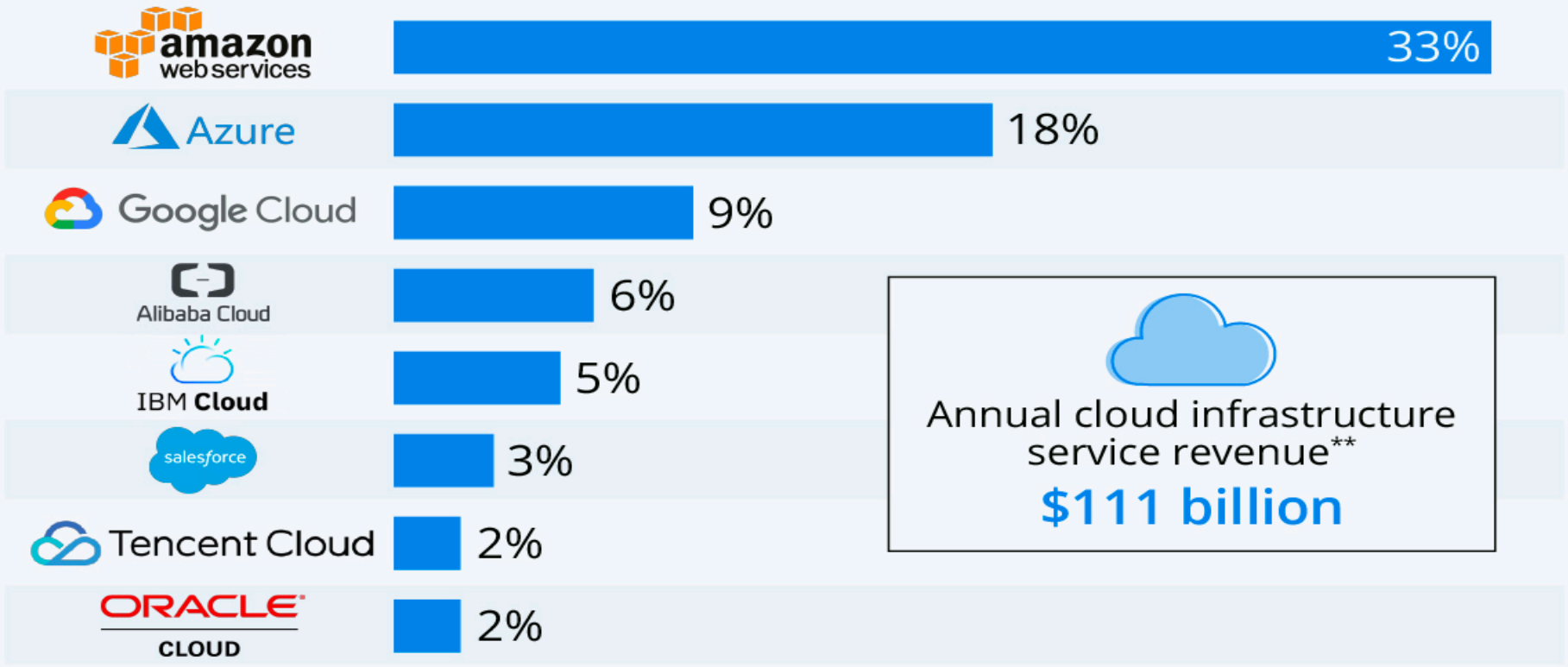
Also describes **data centers** available to many users over the Internet, distributed over multiple locations

Is it important?

amazon

Amazon Leads $100 Billion Cloud Market

Worldwide market share of leading cloud infrastructure service providers in Q2 2020*



* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

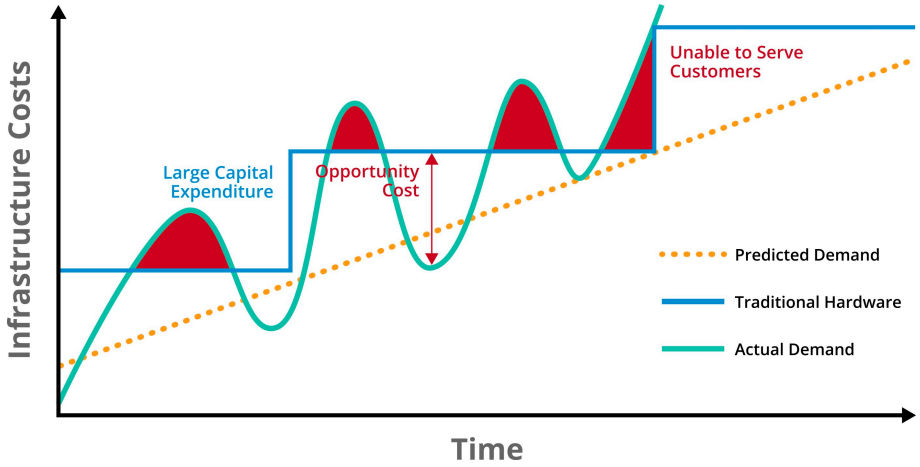
** 12 months ended June 30, 2020

Algorithms for Cloud Computing

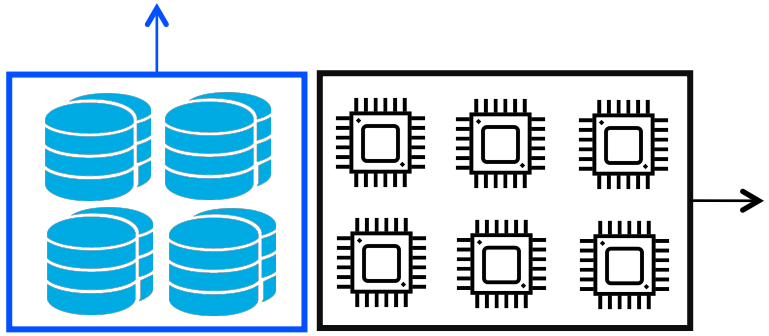
- You will learn how to…
  - Plan resources for future demand
  - Allocate resources in a fair manner
  - Design load balancing algorithms
  - Design network slices
  - How to employ mathematical modeling, optimization, machine learning, and stochastic processes **to design algorithms for cloud systems**

amazon

Resource Planning



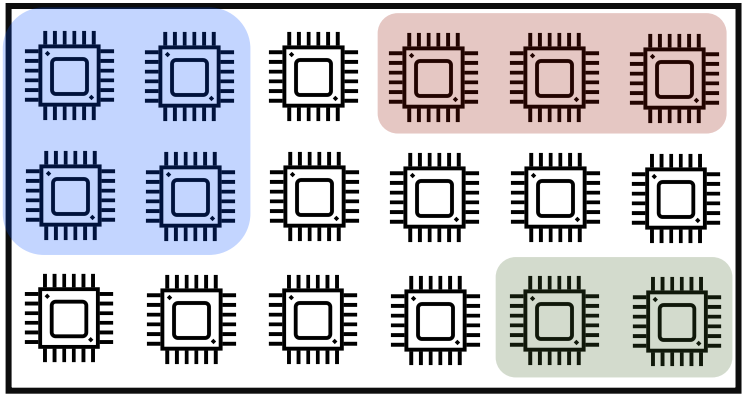
Planning a datacenter



Rightsizing a service

- How many resources should we plan for the future?

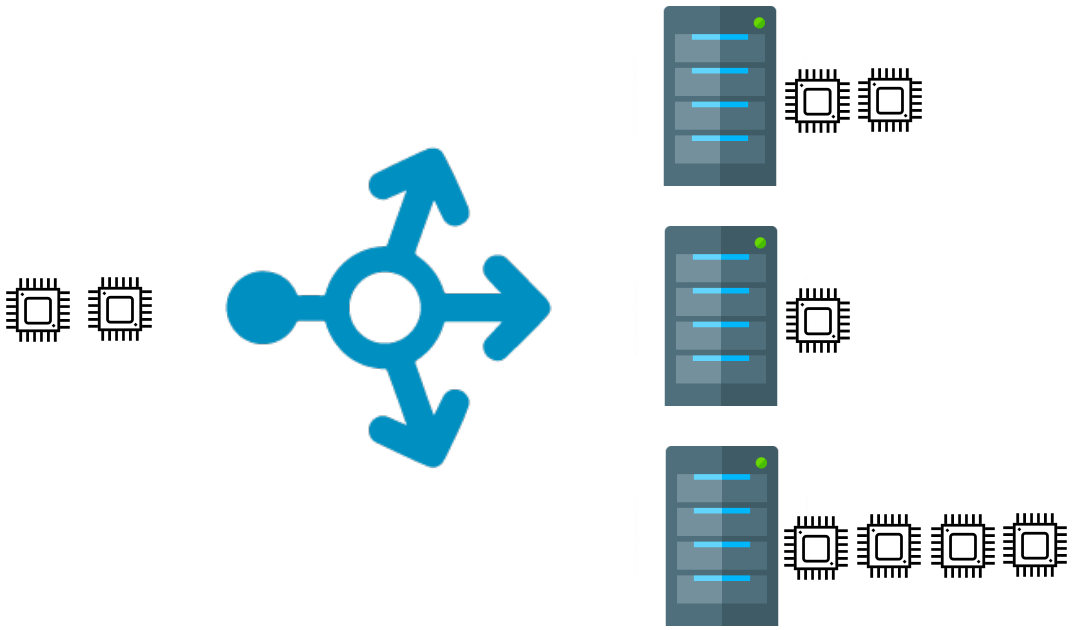
Resource Allocation and Fairness



Sharing resources among users

- How many resources to allocate to each user?

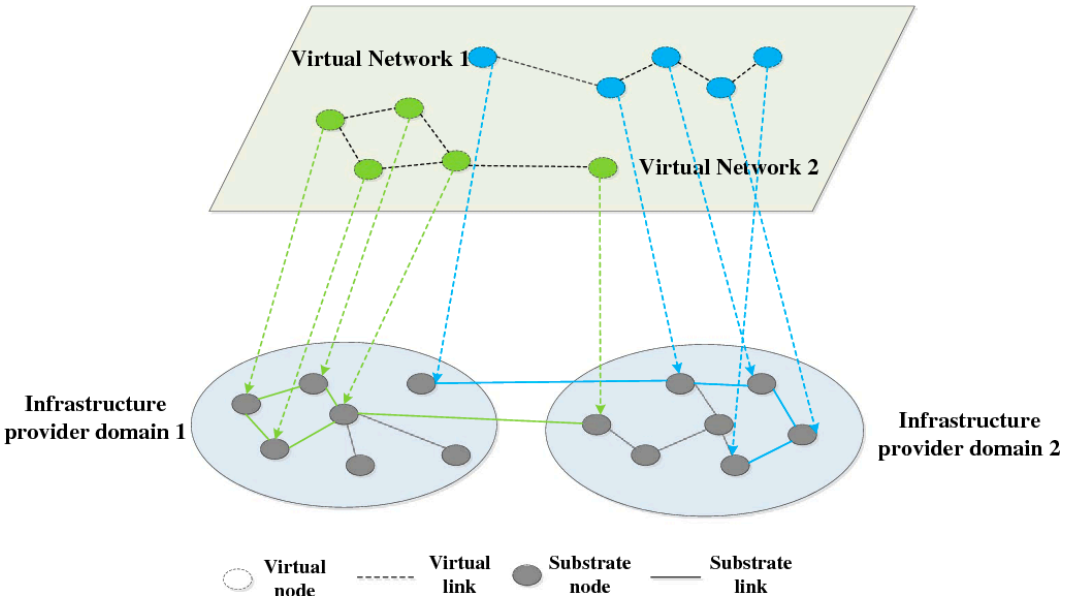
Load Balancing



Routing Jobs to Servers

- Where to route a job?

Network Slicing



Computing and networking

- How to allocate network resources?

Questions?

Katakolo, Greece

amazon

Cloud computing buzzwords

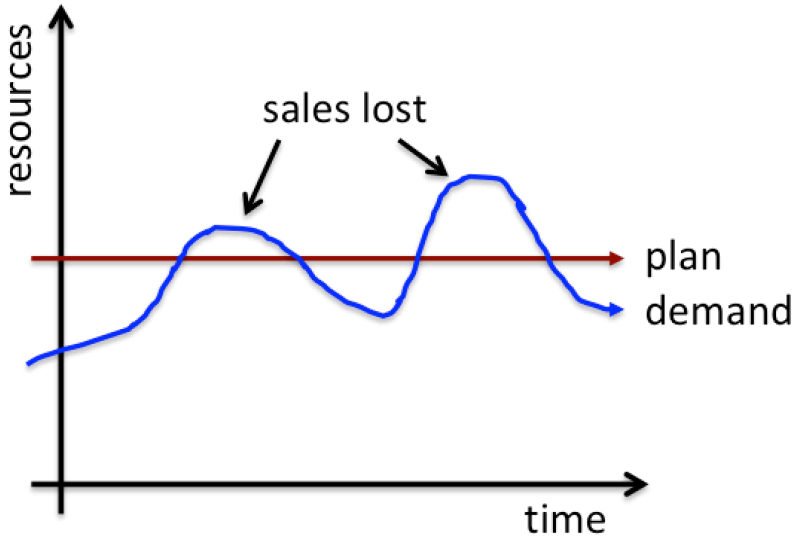
1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Cloud storage
4. Hybrid Cloud
5. Containers
6. Serverless architecture
7. Software Defined Networks
8. Network Slicing
9. Virtual Network Functions
10. Memory disaggregation
11. Map-reduce
12. Devops
13. Edge computing
14. Federated learning

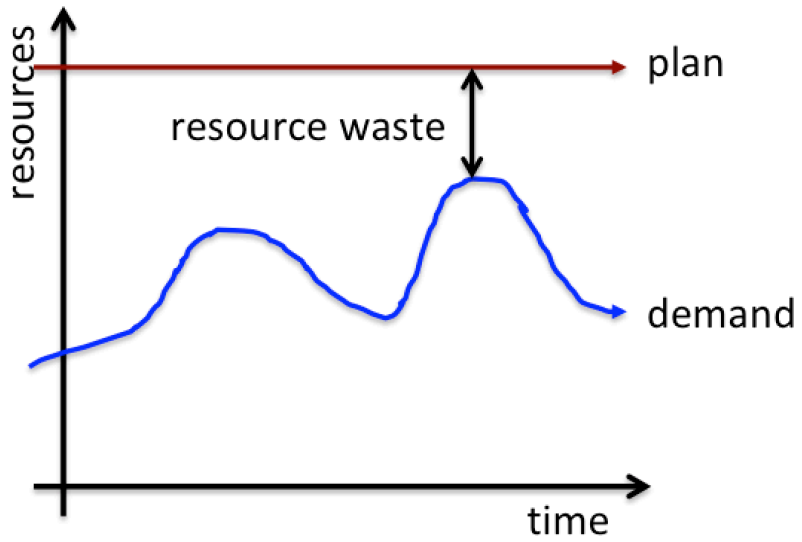
1. Service architecture to provide computing based on VMs (client installs software), eg AWS EC2
2. Service architecture to provide directly applications, Sagemaker
3. Service architecture similar to Dropbox and S3
4. Architecture where private and public cloud are interleaved (eg security products of AWS)
5. Software wrapper/mini OS (Docker)
6. Architecture where we use directly the datacenter resources without need for server concept (Kubernetes)
7. Configurable switches used to optimize datacenter interconnects
8. Application specific virtual networks (5G)
9. Software-based network functionality (firewall, cache, etc)
10. Architecture to interconnect storage units into a single pool
11. Architecture to perform distributed computations in computing clusters
12. A philosophy of quick deployment of services, maintenance and continuous improvement
13. Mini-datacenters in last-mile network connections
14. Architecture for distributed model learning

amazon

Part I, Resource Planning

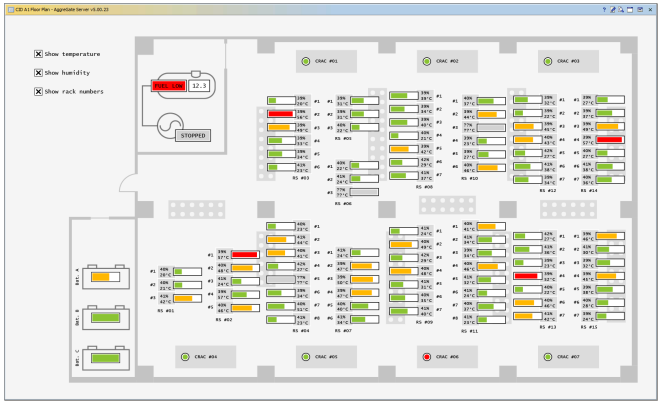
Resource Planning



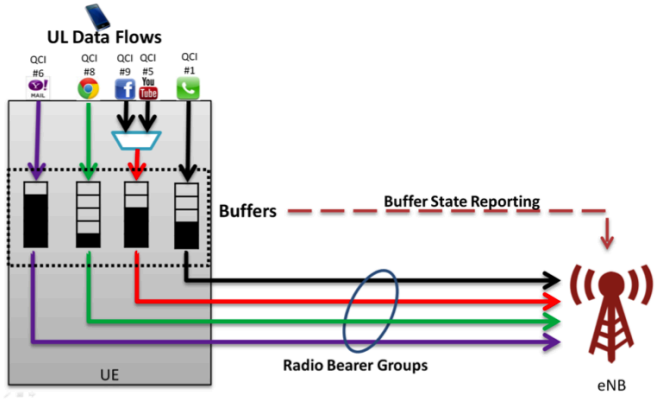


- We want just enough resources
  - Cost from overplanning
  - Disruption from underplanning

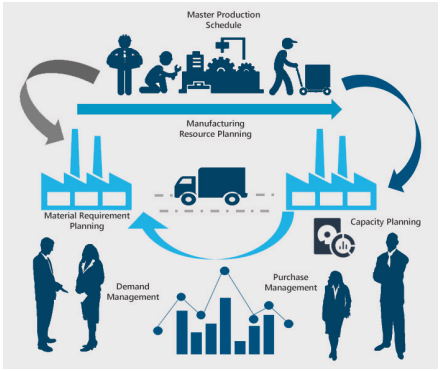
Applications of resource planning



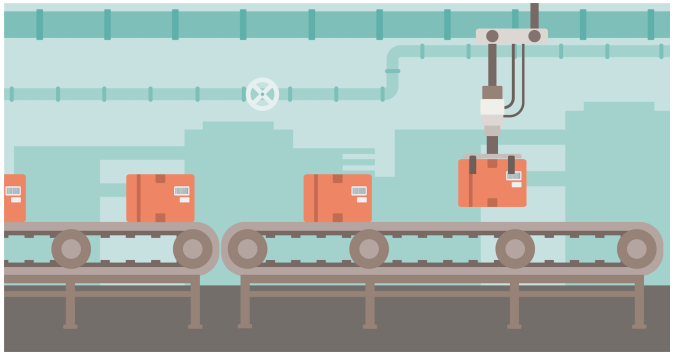
Data Centers



Wireless systems



Supply Chain



Manufacturing

https://cdn10.servertech.com/assets/documents/documents/64/original/Data_Center_Capacity_Planning-DCD_INTELLIGENCE_2019.pdf?1571261229

amazon

Outline

- **Newsvendor:** unknown demand, known distribution
- **Forecasting & robust planning:** unknown demand, normal
- **Online Convex Optimization:** unknown demand, unknown distribution, possibly non-stationary

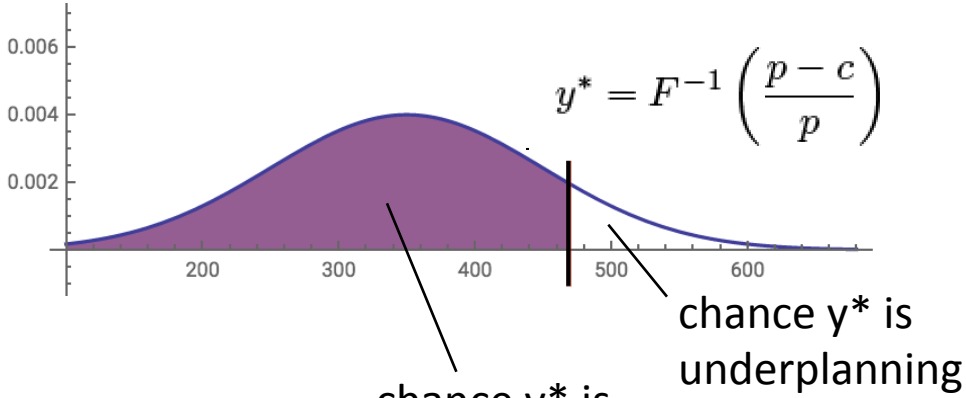
Newsvendor

- How many newspapers to procure, without knowing the day's demand? Unsold newspapers are wasted

- Problem formulation:
  - p: selling price
  - c: procurement cost
  - X: future demand, F(x) = P(X ≤ x)
  - y: #newspapers to order

$$\max_{y \geq 0} p\mathbb{E}[\min(y, X)] - cy$$

- Critical Fractile formula



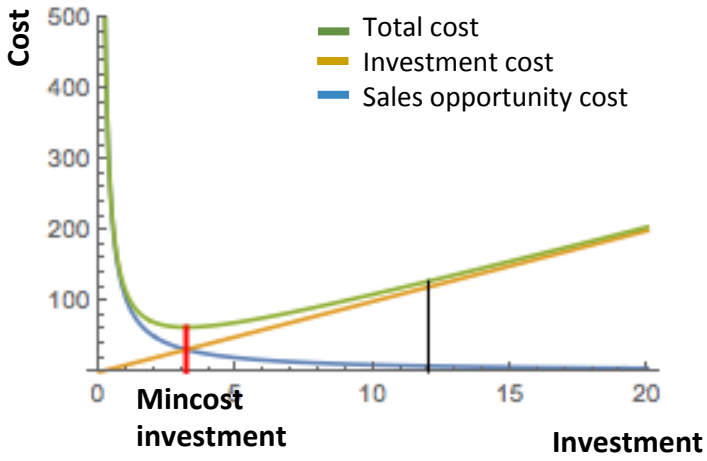
https://demonstrations.wolfram.com/CapacityPlanningForShortLifeCycleProductsTheNewsvendorModel/

amazon

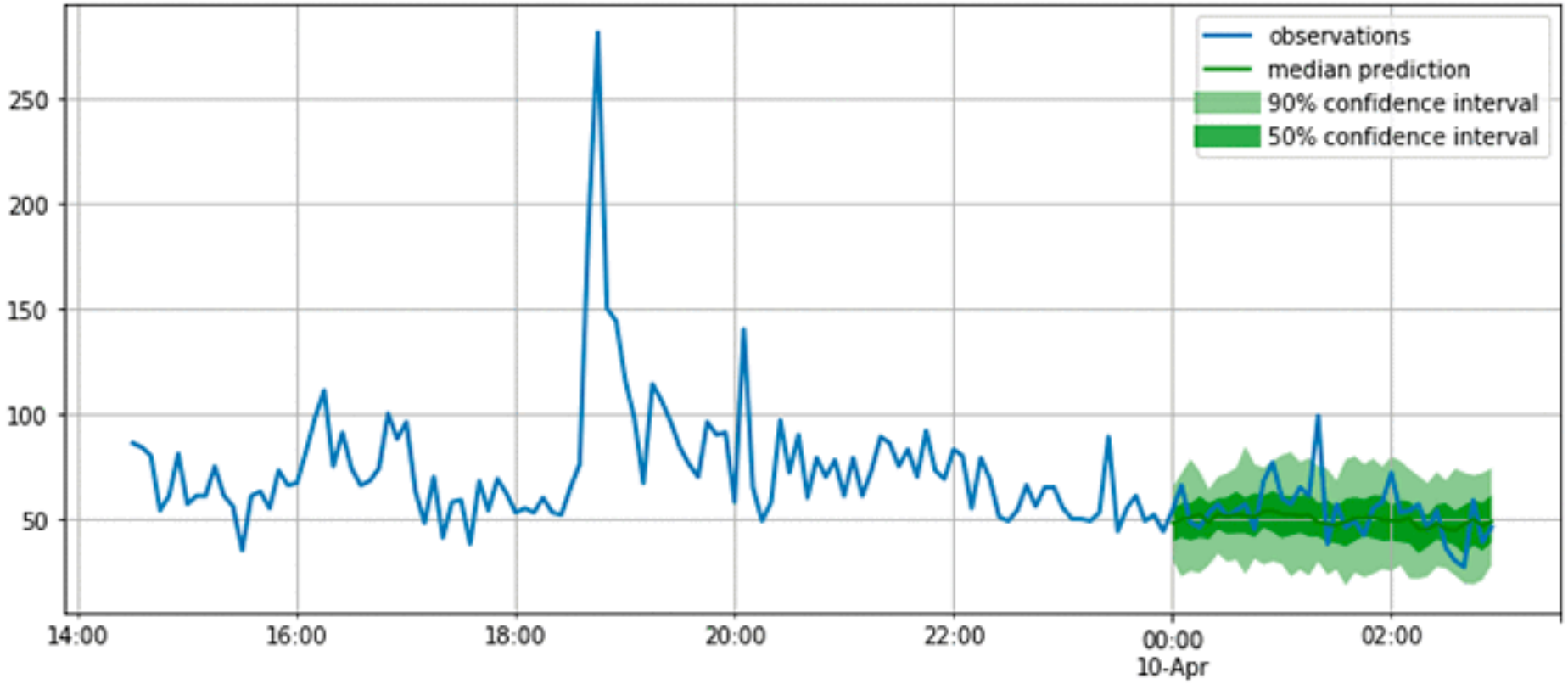
Proof

$$\mathbb{E}[\min(y,X)] = \mathbb{P}(X \le y)\mathbb{E}[\min(y,X)|X \le y] + \mathbb{P}(X > y)\mathbb{E}[\min(y,X)|X > y]$$
$$= \int_{x \le y} x\varphi(x)dx + (1 - F(y))y$$
$$\frac{\partial \mathbb{E}[\min(y,X)]}{\partial y} = y\varphi(y) + (1 - F(y)) - y\varphi(y) = 1 - F(y)$$

$$U'(y) = p - c - pF(y)$$



Demand Forecasting



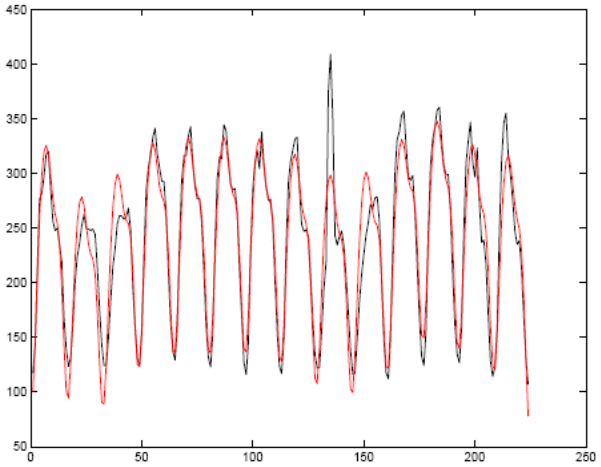
[Hyndman18] R. Hyndman and G. Athanasopoulos, *"Forecasting Principles and Practice"*, 2018.

amazon

Time-Series Forecasting

- Observations $x_1, x_2, \ldots, x_{t-1}$
- Predict next observation F_t

Forecasting: Can be viewed as computing the distribution of x_t conditional on $x_1, \ldots, x_{t-1}$



- Forecasting Error: $E_t = F_t - x_t$
- **Objective:** Choose prediction to minimize a loss function, e.g., MAPE:

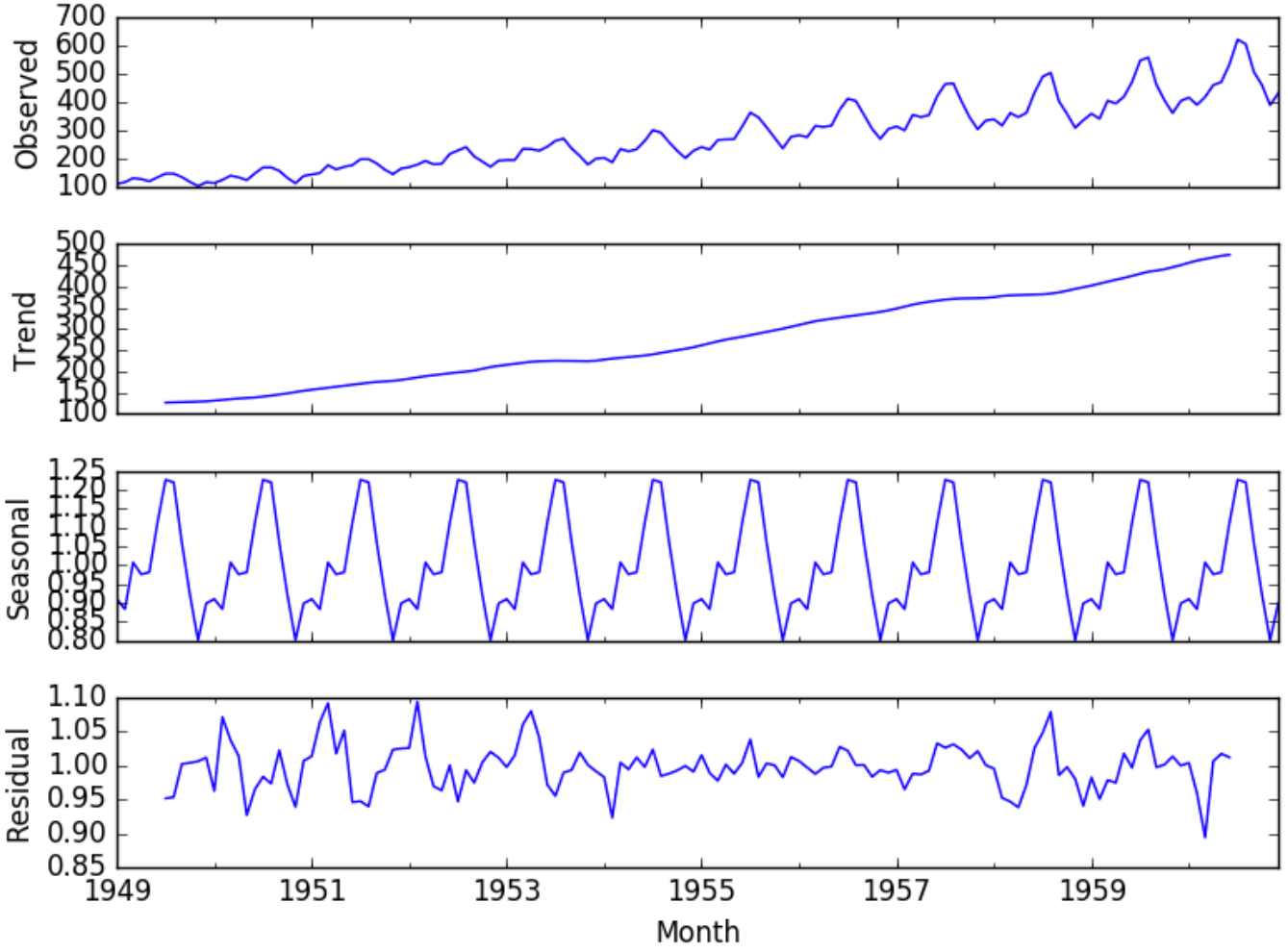
$$MAPE = 100 \cdot \frac{\sum_{i=t}^{t+N-1} |E_i / x_i|}{N}$$

amazon

Basic Methods

- Naïve forecast: $F_t = x_{t-1}$ (forgets, good in changes)
- Mean estimator: $F_t = \frac{1}{t-1}\sum_{i=1}^{t-1} x_i$ (estimates stationarity)
- Exponential smoothing: $F_t = \alpha x_{t-1} + (1-\alpha)\alpha x_{t-2} + \cdots + (1-\alpha)^{t-2}\alpha x_1$
  - α in [0,1]
  - α=1 yields naïve forecast
  - α➔0 yields mean estimator

Seasonal decomposition



ARIMA forecasting

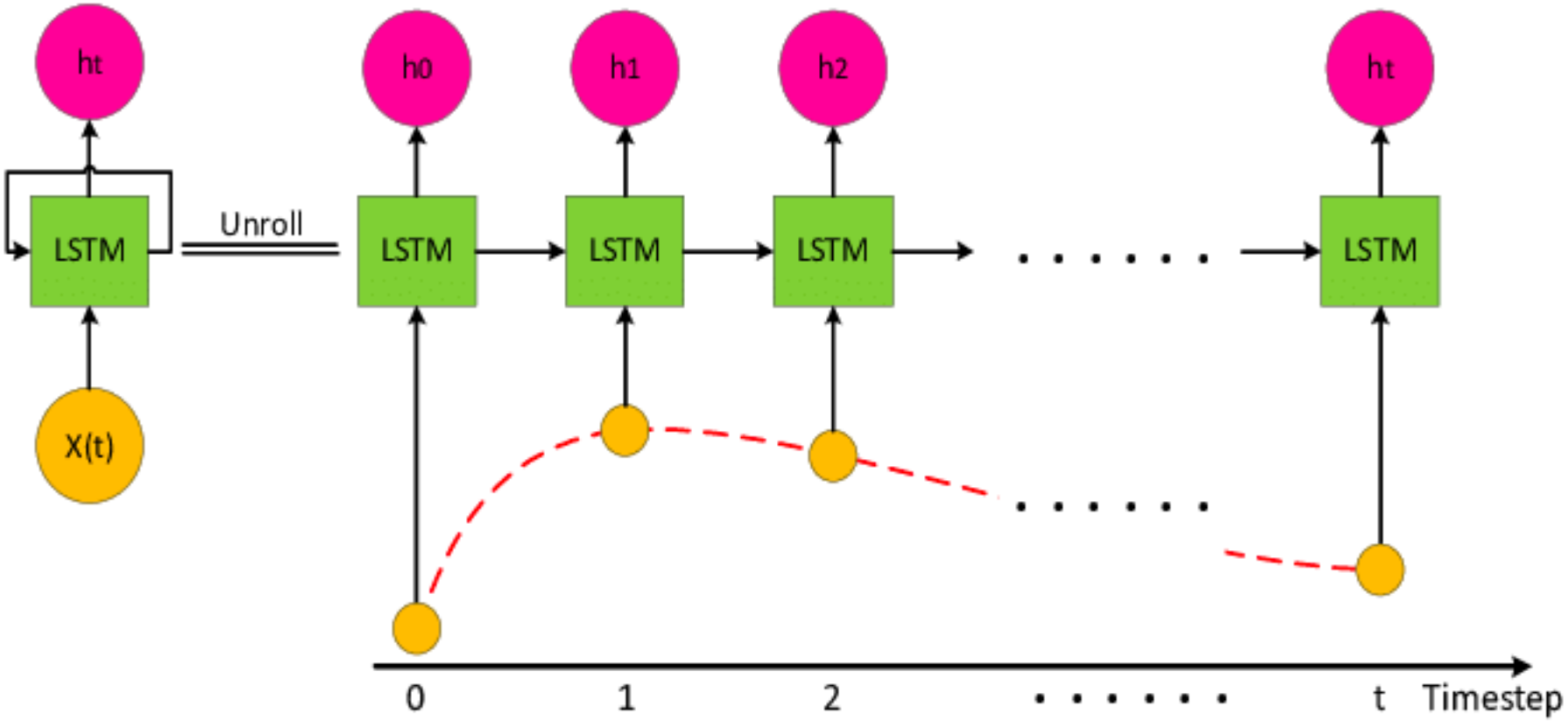
- p AR (Auto-Regressive) $F_t = \alpha x_{t-1} + (1-\alpha)\alpha x_{t-2} + \cdots + (1-\alpha)^{t-2}\alpha x_1$
- q MA (Moving Average)
- ARMA: $F_t = \mathbb{E}[F_t] + E_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j E_{t-j}$

actuals

white noise terms representing residuals

- ARIMA, Seasonal ARIMA
- Auto-ARIMA: auto discovers orders p,q and differencing

LSTM



https://ts.gluon.ai/

amazon

Stochastic Planning

- **Assumption:** residuals are normal distributed

$$\begin{cases} x_k = \mathbb{E}[F_k] + E_k \\ E_k \sim \mathcal{N}(0, \sigma^2) \end{cases}$$

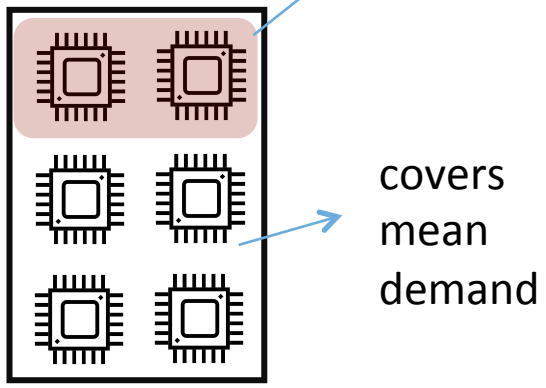
- **Chance constraint:**

$$P(y_k < x_k) \leq \epsilon$$

$$P(E_k > y_k - \mathbb{E}[F_k]) \leq \epsilon$$

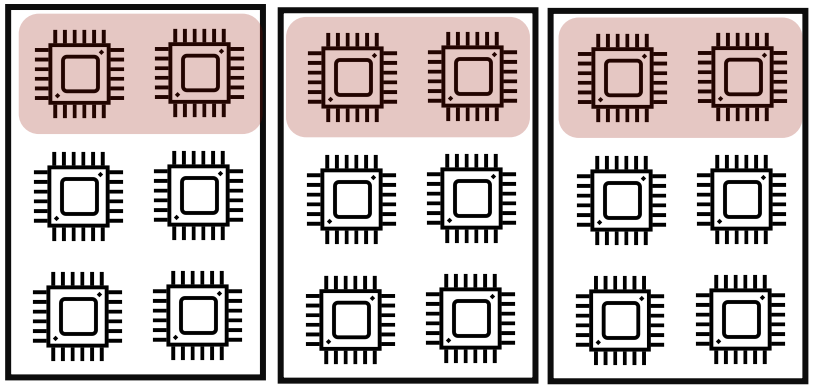
$$Q\left(\frac{y_k - \mathbb{E}[F_k]}{\sigma}\right) \leq \epsilon$$

$$y_k \geq \underbrace{\mathbb{E}[F_k]}_{\text{mean demand}} + \underbrace{\sigma Q^{-1}(\epsilon)}_{\text{uncertainty buffer}}$$

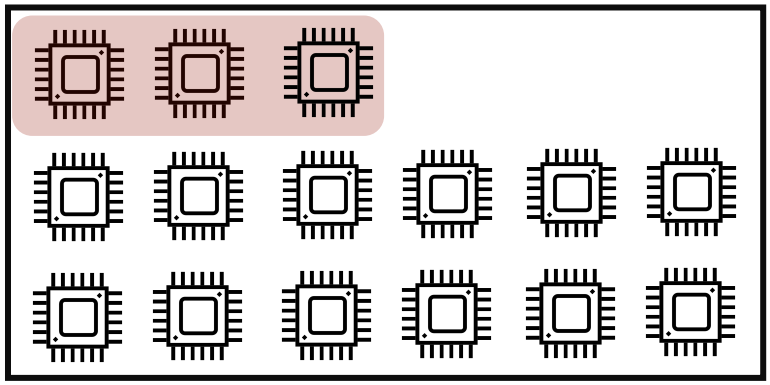


Resource Pooling

- How to plan for multiple users?



Common mistake



Correct

$$P\left(\sum_k y_k < \sum_k x_k\right) \leq \epsilon \Leftrightarrow P\left(\sum_k E_k > \sum_k y_k - \sum_k \mathbb{E}[F_k]\right) \leq \epsilon$$

$$\Leftrightarrow Q\left(\frac{\sum_k y_k - \sum_k \mathbb{E}[F_k]}{\sqrt{K}\sigma}\right) \leq \epsilon$$

$$\Leftrightarrow \sum_k y_k \geq \sum_k \mathbb{E}[F_k] + \sqrt{K}\sigma Q^{-1}(\epsilon)$$

James Hamilton, AWS Chief Scientist

- "The cost of supporting the efficiency of the aggregate workload is wildly better than any individual workload. Super cool. As a cloud provider, I win before I even start thinking."

https://review.firstround.com/Head-of-Amazon-Web-Services-on-Finding-the-Next-Great-Opportunity
https://mvdirona.com/jrh/work/

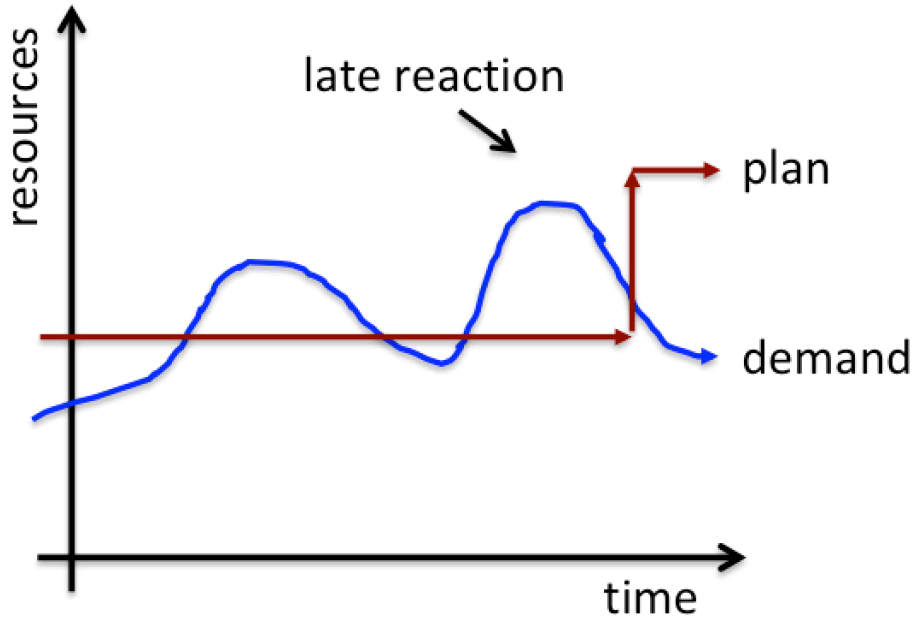
amazon

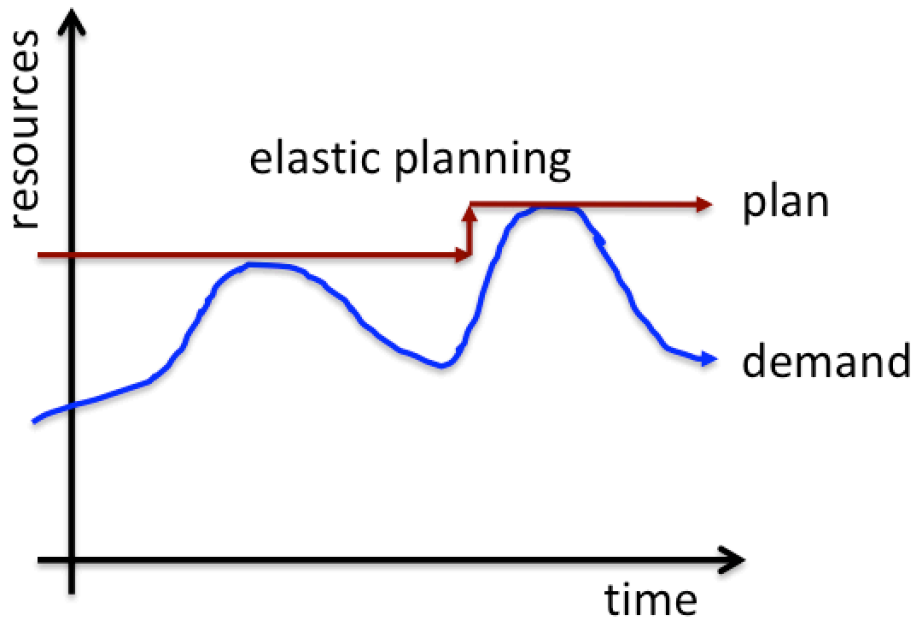
James on how to manage cost in datacenters

Go where the **user pain** is (service optimize for cost). And, as a test, if the very first thing the largest users do is shut off auto-management, the feature isn't yet right. We should be implementing auto-management systems that the very biggest users actually chose to use. ***These very large customers prioritize stability over the last few percentage of optimization***. They don't want to get called in the middle of the night when a plan changes. My recommendation is to adopt a **do-no-harm mantra** and, failing that, detect and correct harm before it has broad impact. Be able to revert back a failed optimization fast. Focus on the problems where human optimization is not possible. For example resource allocation is extremely dynamic. The correct amount of buffer pool, sort heap, and hash join space varies with the workload and can't be effectively human set. This type of problem is perfect for auto-management.

Focus on optimizations that are 1) stable (do no harm) or 2) dynamic where you can do better than a static, human chosen setting.

Adaptive resource planning

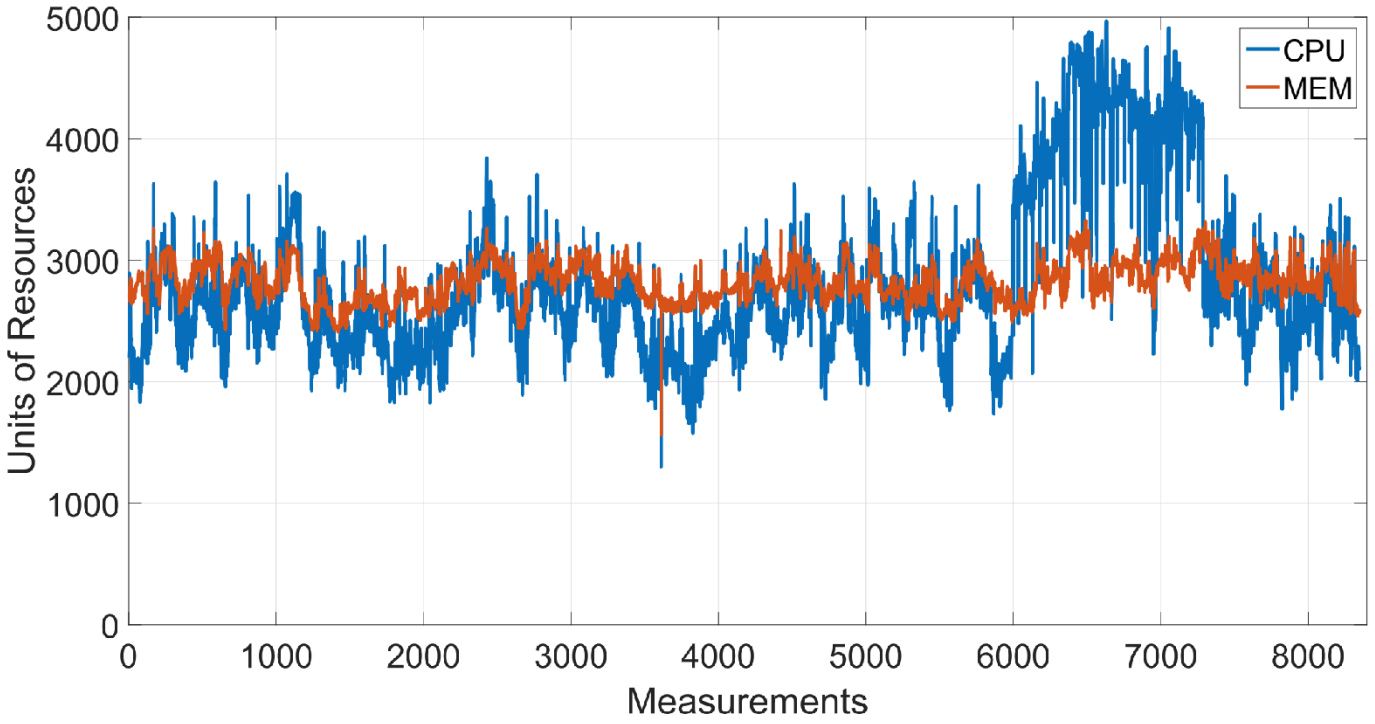




amazon

Planning for unforcastable demand

- Example: demand for cloud resources in Google cluster

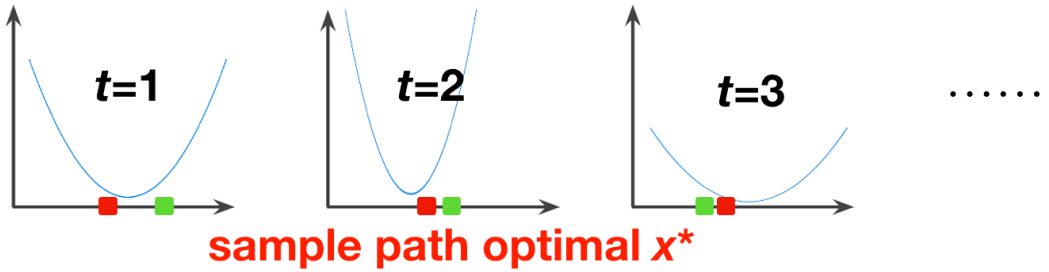


[Reiss12] C. Reiss et al. "*Heterogeneity and dynamicity of clouds at scale: Google trace analysis*," ACM SoCC, 2012.

amazon

Online Convex Optimization

- Algorithm chooses x_t
- Receives unknown loss $f_t(x_t)$
- Find algorithm to minimize $\sum_{t=1}^{T} f_t(x_t)$



Key idea: compare **algorithm losses** vs ***x*** losses*

[S-Schwartz12] I. S.-Schwartz, "*Online learning and Online Convex Optimization*,", 2012.
[Hazan14] E. Hazan, "*Introduction to Online Convex Optimization*," 2014.

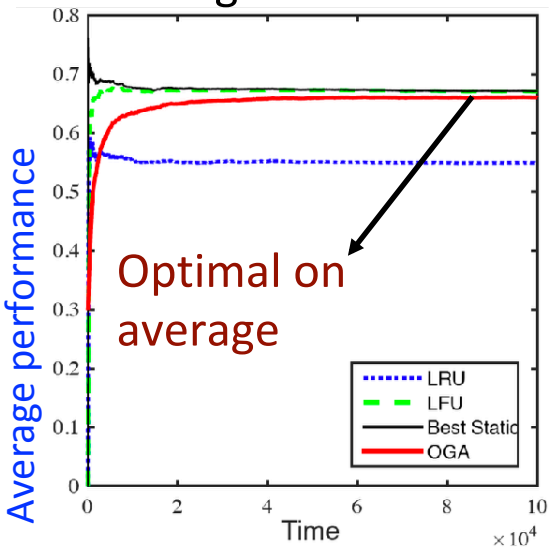
The metric of Regret

- **Definition** (Regret): $\mathbf{Reg}^{\pi} = \max_{Pr(f_1,\ldots,f_T)} \underbrace{\sum_{t=1}^{T} f_t(y_t)}_{\text{algorithm cumulative loss}} - \underbrace{\sum_{t=1}^{T} f_t(y^*)}_{\text{benchmark cumulative loss}}$

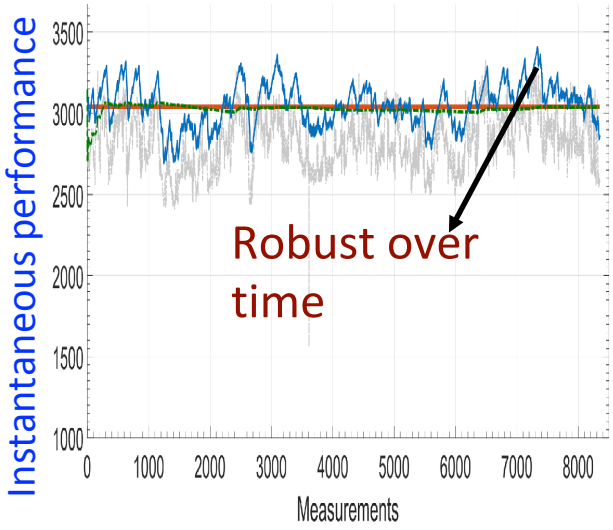
Adversary chooses f

- An algorithm with $\mathbf{Reg}^{\pi} = o(T)$ is called "no regret algorithm"

no regret cache hits



no regret resource plan

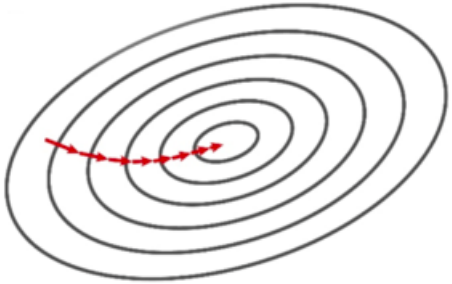


amazon

Zinkevich's Online Gradient

- Take a step in the direction of previous gradient:

$$x_t = \underbrace{\Pi_{\mathcal{X}}}_{\text{Eucl. projection}}\Big[x_{t-1} + \underbrace{\eta_t}_{\text{stepsize}}\underbrace{\nabla f_{t-1}(x_{t-1})}_{\text{previous gradient}}\Big]$$



Theorem: Online gradient descent has regret $O(\sqrt{T})$ [Zinkevich03]

Regret Lower bound = faster possible learning rate

Theorem: If f is convex, any algorithm has regret: $\Omega\left(\sqrt{T}\right)$ [Abe08]

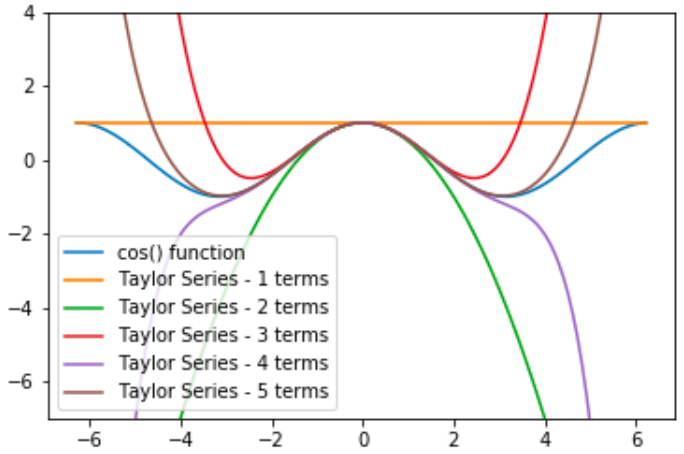
Online gradient is optimal!

[Abe08] J. Abernethy, "*Optimal Strategies and Minimax Lower Bounds for Online Convex Games,*" COLT, 2008.

[Zinkevich03] M. Zinkevich, "*Online Convex Programming and Generalized Infinitesimal Gradient Ascent,*" ICML, 2003.

Intuition: Why gradient?

Estimate next function using Taylor expansion of previous



Intuition #1 (Linear approximation):

$$\hat{f}_t \approx f_{t-1} + \nabla f_{t-1}(x_{t-1})(x_t - x_{t-1})$$

Intuition #2 (Filtering):

Consider minimizing the quadratic function iteratively

$$x_1 = x_0(1-\eta)$$
$$x_2 = x_1(1-\eta) = x_0(1-\eta)^2$$
$$\vdots$$
$$x_n = x_0(1-\eta)^n$$

initial conditions are forgotten: difficult to fool such a learner

amazon

OCO for newsvendor

unknown demand

Expected profit: $$f_t(y_t^\pi) = \underbrace{-c\sum_{k=1}^{K} y_{t,k}^\pi}_{\text{investment cost}} - \underbrace{\sum_{k=1}^{K}(x_{t,k} - y_{t,k}^\pi)^2 \mathbb{1}_{\{x_{t,k} > y_{t,k}^\pi\}}}_{\text{sales loss}}$$

$\delta_k(y_k) = x_{t,k} - y_k$

Subgradient: $$\frac{\partial f_t}{\partial y_k} = -c + 2(x_{t,k} - y_k)\mathbb{1}_{\{x_{t,k} > y_k\}} = \begin{cases} 2\delta_k(y_k) - c & \text{if } y_k < x_{t,k}, \\ -c & \text{otherwise} \end{cases}$$

Online Subgradient Ascent: $$\begin{cases} y_{t+1,k} = (y_{t,k} + \eta(2\delta_k(y_k) - c))^+ & \text{if } y_k < x_{t,k}, \\ y_{t+1,k} = (y_{t,k} - \eta c)^+ & \text{otherwise.} \end{cases}$$

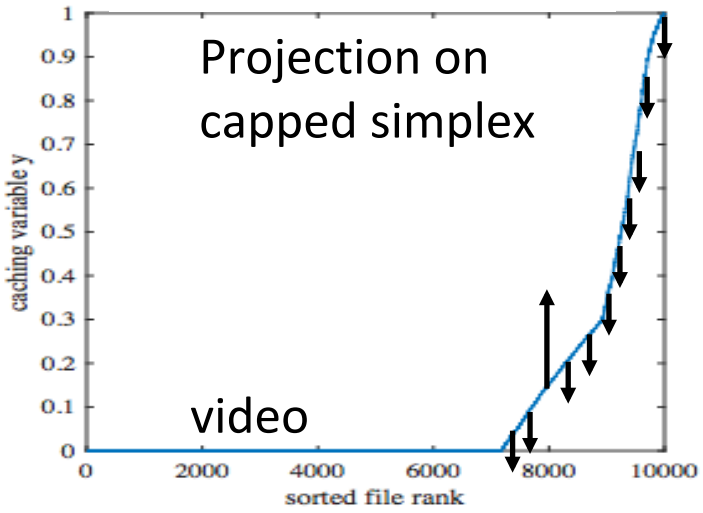
increase if sold everything,
decrease when waste

Many restaurants provision food in this way!

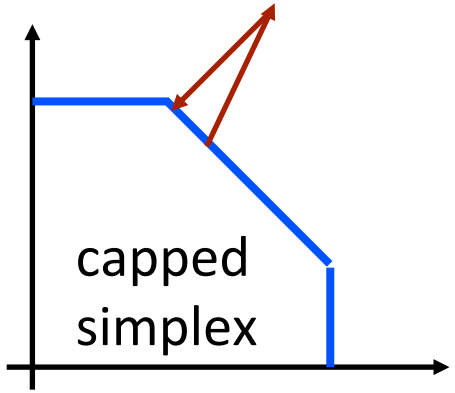
amazon

OCO for caching

- **I can store C videos. Which ones?**
- Fraction of video i cached at time t: $y_t(i)$
- Decision vector: $y_t \in [0,1]^I : \sum_i y_t(i) \leq C$
- Video i requests: $p_t(i)$
- Expected profit: $f(y_t) = \sum_i y_t(i) p_t(i)$



- *Optimal action:* $y_t(i) = \Pi\left[y_{t-1}(i) + \eta p_{t-1}(i)\right]$



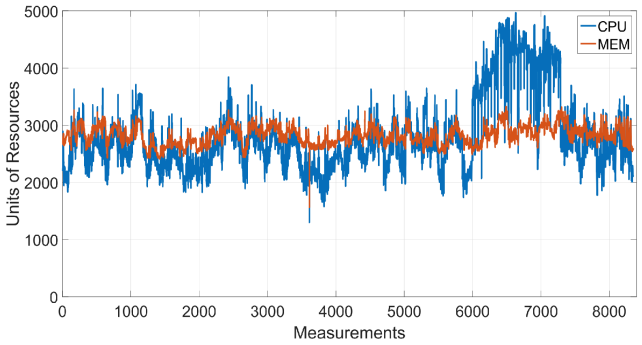
[Paschos19] G. Paschos et al. "*Learning to cache with no regrets*," IEEE INFOCOM 2019.
[Wang15] W. Wang and C. Lu, "*Projection onto the capped simplex*," arXiv:1503.01002v1, 2015.

amazon

OCO for resource planning

Violation probability: $g_t(y_t) = \mathbf{P}\left(y_t > x_t\right) - \epsilon$

Horizon violations: $\mathbf{Ctr} = \sum_{t=1}^{T} g_t(y_t)$



Definition: an algorithm is *feasible* if $\mathbf{Ctr} = o(T)$

Theorem: Online Lagrangian is feasible no regret policy for K-benchmark [Pas19]

[Pas19] N. Liakopoulos et al., "*Cautious Regret Minimization: Online Optimization with Long-term Budget Constraints*," ICML, 2019.
[Tsi09] S. Mannor et al., "*Online Learning with Sample Path Constraints*," Journal of Machine Learning Research, 2009.

amazon

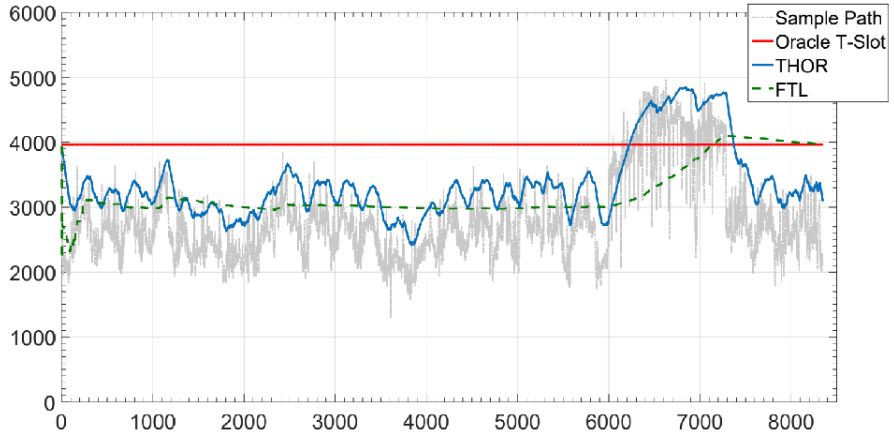
Online Lagrangian

Plan $y_t = \left[y_{t-1} - \frac{1}{2\eta} \left(V \frac{\partial f_{t-1}}{\partial y_{t-1}} + Q(t) \frac{\partial g_{t-1}}{\partial y_{t-1}} \right) \right]^+$

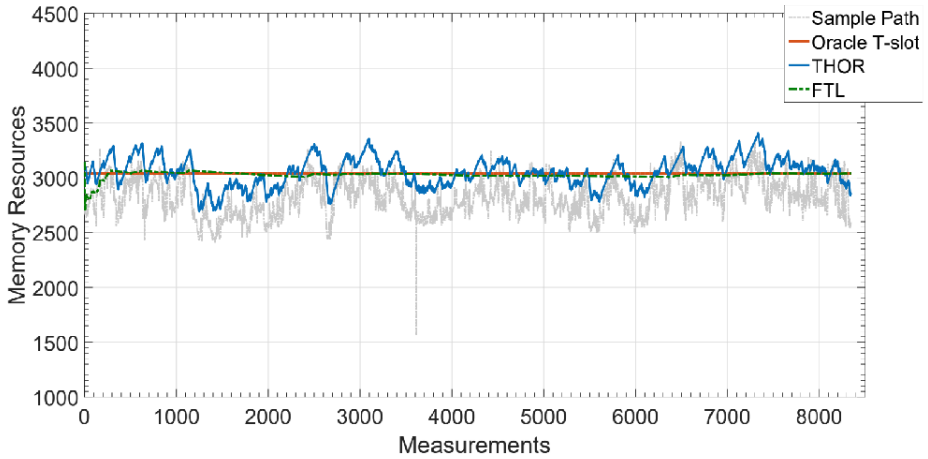
gradient of f_{t-1} ~Lagr. multiplier gradient of g_{t-1}

where $Q(t+1) = \left[Q(t) + b_{t-1}(y_t) \right]^+$

Accumulates (linear) predictions of violations



(a) CPU Reservations



(b) Memory Reservations

amazon

Questions?

paschosg@amazon.com

- For questions about the course
- For questions about internship opportunities

https://paschos.net/

- Course material & relevant papers