Microsoft

人工智能系统 System for AI

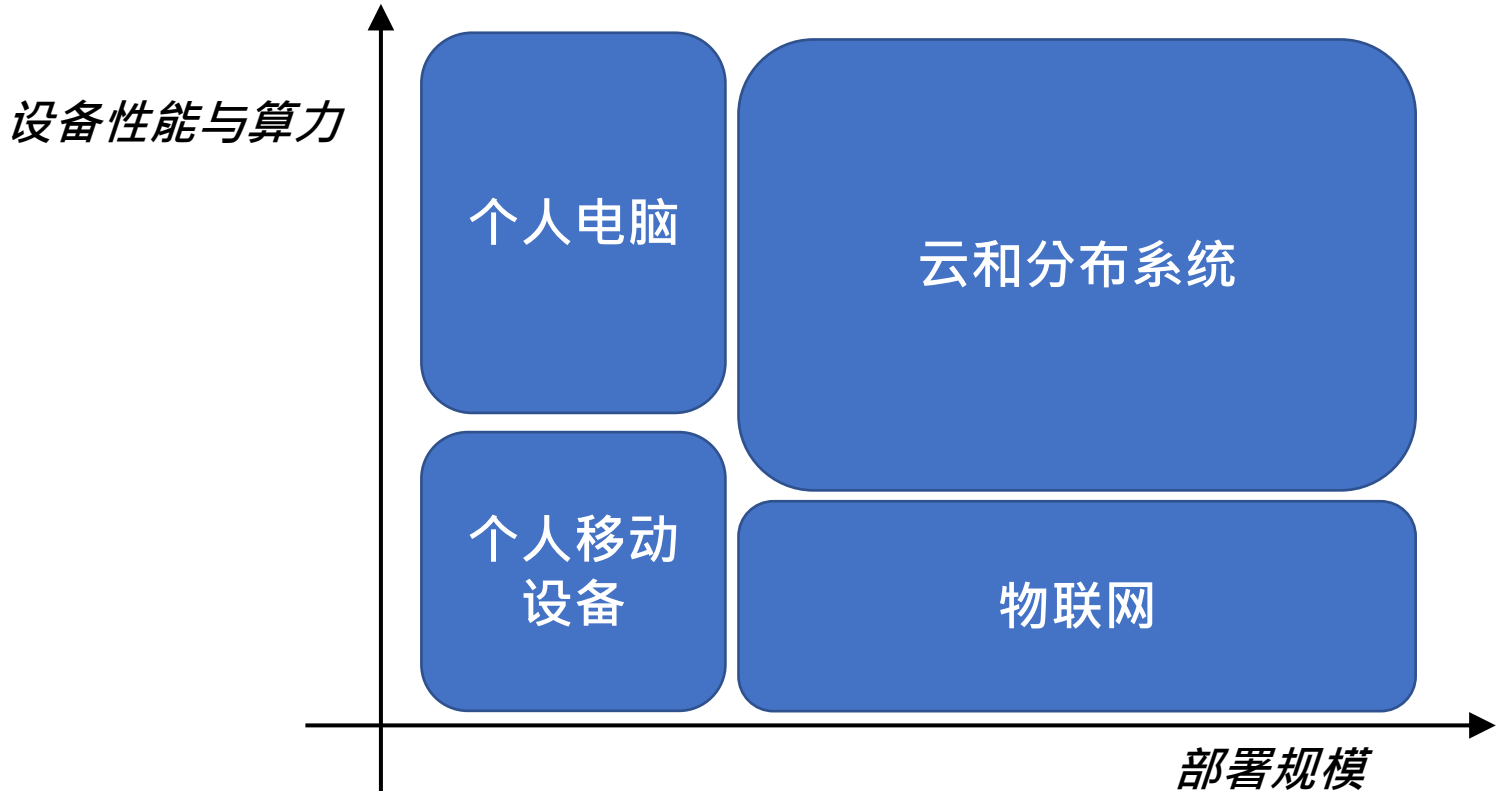
利用人工智能来优化计算机系统
AI-for-Systems

课程主要内容

- **现代系统带来的挑战**
- **应用人工智能来优化现代系统的案例**
  - 案例 #1: 数据库索引
  - 案例 #2: 视频流传输
  - 案例 #3: 系统选项与参数调优
- **落地的考虑要素和痛点**

“Software Runs the World”

- 我们生活中常见的系统

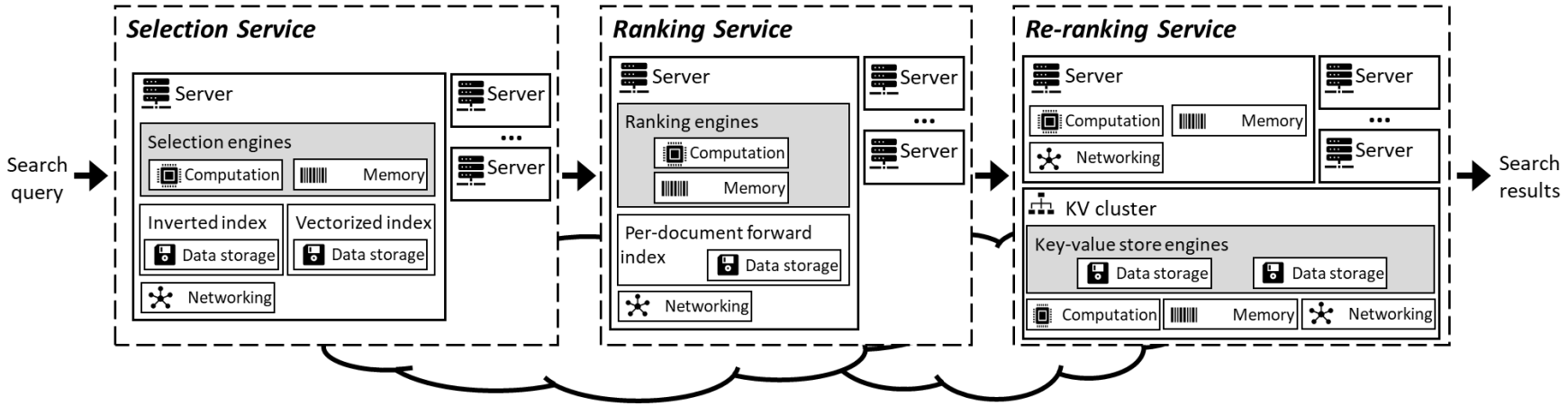


系统设计与运维充满了决策

- **软件**
  - 编译器的策略，操作系统的调度策略，高速缓存里的置换算法，分布系统里资源的分配， 系统参数的调优，微服务的扩容，数据库索引...
- **网路**
  - TCP 的 congestion control 和 flow control，数据压缩策略，网路品质的预测，视频流传输比特率，防火墙的规则匹配策略...
- **硬件与架构**
  - CPU 的数据缓存和预存取，电路布局，数据中心温度与湿度的预测...

规模和动态性提高了系统复杂度

- **规模的维度**
  - 一个系统的组成可以有多个子系统，子服务，和子组件
  - 每个系统可能分布在上千个服务器上
    - 例子：Bing 搜索引擎



规模和动态性提高了系统复杂度

- **动态性的维度**
  - 系统负载：譬如，用户搜索关键字随着时间而变...
  - 系统部署：譬如，软件每星期更新，基础架构和硬件规格定期更新，微服务扩容，虚拟机迁移...
  - 外在因素：譬如，网络品质

复杂度使得系统难以被工程师有效地优化

- **人力资源的增长速度跟不上系统规模的增长**
  - 缺乏资深的系统工程师
  - 人的知识与经验的传播需要大量的时间
- **现代的系统能输出大量的系统数据，和具备巨大的优化搜索空间**
  - 数据源：软件，服务器硬件，网路
  - 系统工程师难以理解大量的数据，并找出之中的相关性
- **因为动态性，系统需要持续地被维护和优化**

范式转移：AI-for-Systems

- **过去的计算机系统相对容易理解**
  - 手写的算法或启发法，来对特定的场景，找到近似解
  - *No-free-lunch theorem*

- **但是，机器学习能帮我们更准确地建模系统复杂的行为**
  1. 机器学习擅长于探索和学习大规模数据里复杂的关系
  2. 基础硬件的提高 (譬如 GPU)，和机器学习工具的普及 (譬如 PyTorch 和 Scipy)
  3. 标注数据较易获取：

小结 / 思考

- 系统的复杂度越大，系统工程师就越难总结出客观存在的系统行为规律
- 机器学习可以从海量的系统数据中归纳总结出其内在规律

AI-for-Systems 实现的方式

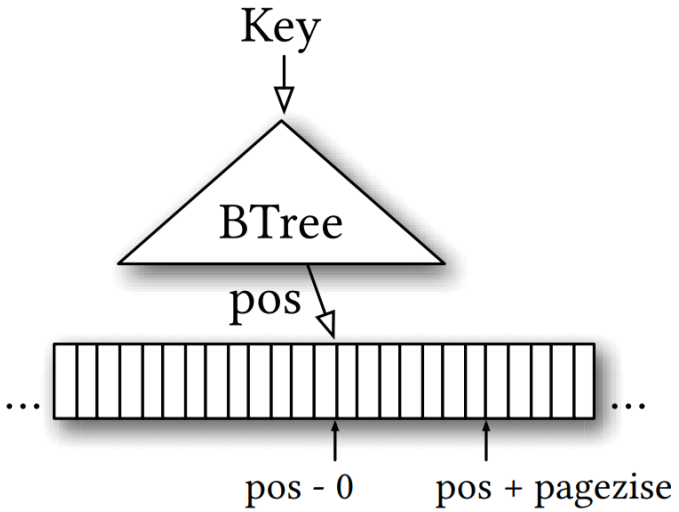
- **“替代”现有的系统元件或决策策略**
  - 案例 #1: 数据库索引
  - 案例 #2: 视频流传输
- **“增强”现有的系统元件**
  - 案例 #3: 系统参数调优

案例 1：数据库索引

- **索引常被用来加速数据库查询**
  - 索引是一种数据结构，存储着索引的值和这个值的数据所在行的物理地址
- **传统索引没有考虑数据的分布特点，往往预先假设了最差的数据分布，从而期望索引具备更高的通用性**
  - 这些索引往往会牺牲大量的存储空间和性能

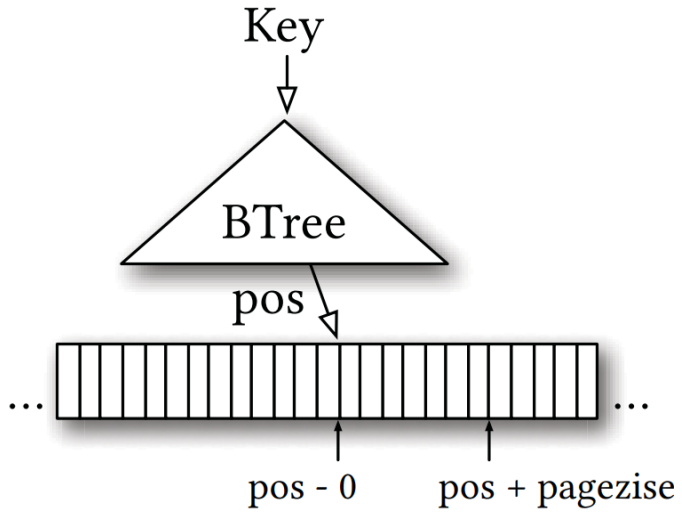
传统索引：B-Tree

- B-Tree 中通常按照 page 来组织数据，每一个 page 对应 B-Tree 中的一个节点
- 基于一个 key 进行查询时，事实上是先通过非叶子节点的索引信息，查找到一个目标 page
  - 搜索时间复杂度：O (log n)
  - 空间复杂度：O (1)



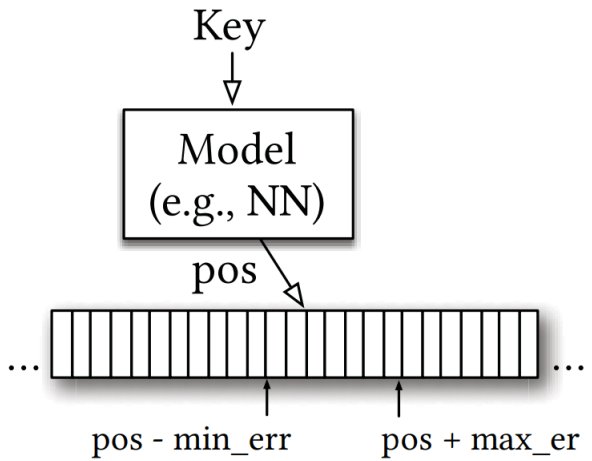
传统索引：B-Tree

- **但是当我们了解数据分布的情况下，B-tree 索引不一定是最好的选择...**
  - 假设我们的数据集就是 1 - 100M 的序列，key 值本身就可以作为偏移量使用。那么时间和空间复杂度可以都是 O (1)

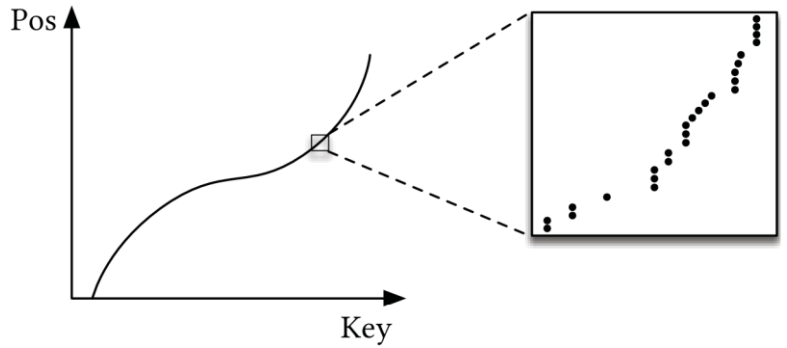


Learned Index

- **用模型来预测 key 的位置**
  - 利用 DNN 学习数据集的分布
  - Key 的位置很大概率在 *pos* – *min_err* 和 *pos* + *max_err* 之间
  - 如果预测错误，则退回到 B-tree



- **对于已排序的数据，预测某个 key 的位置可以被看成一个学习 CDF 曲线的问题**



"The Case for Learned Index Structures"
SIGMOD '18
Kraska et al.

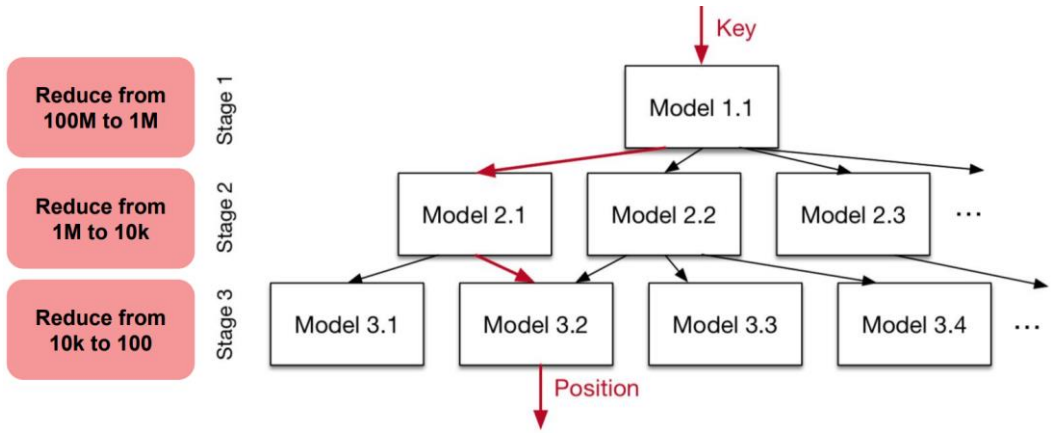
Learned Index 的实现

Naïve Learned Index

- 一个 fully-connected 的 DNN
- 在 2 32-neuron layers 的 DNN 下，*min_err* + *max_err* 大约是 10k
  - 再减低 *min_err* 和 *max_err* 变得越来越困难...

Recursive Model Index (RMI)

- A hierarchy of models



Learned Index vs. B-tree 索引

- Comparison baseline: B-Tree with page size of 128

空间：up to 0.23×
时间：up to 3.08×

空间：up to 0.24×
时间：up to 2.07×

空间：up to 0.24×
时间：up to 1.79×

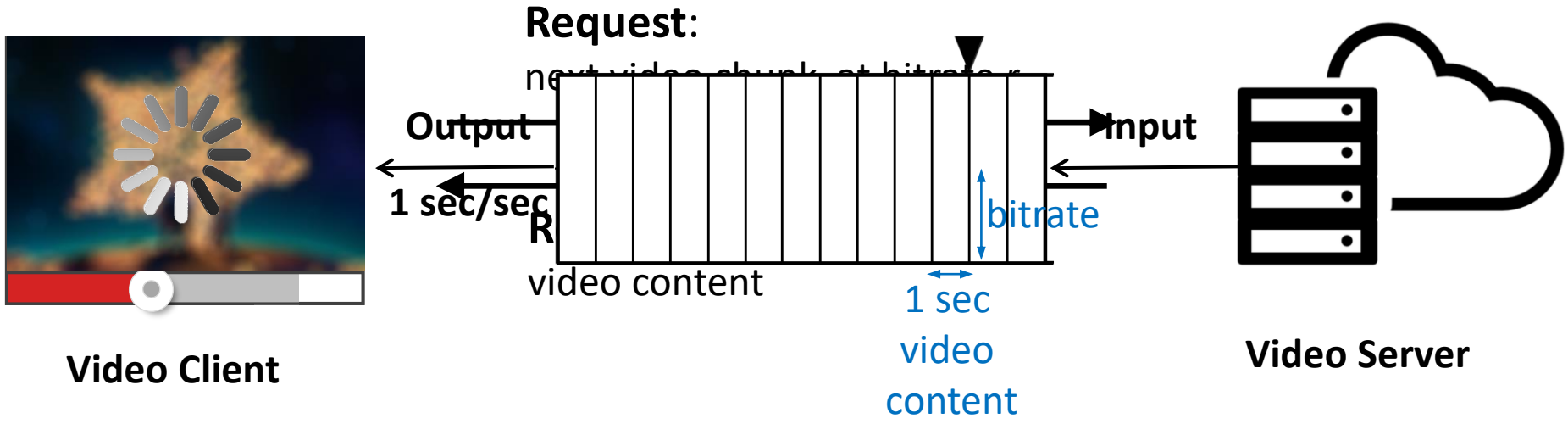
		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 64	26.23 (2.00x)	277 (0.96x)	172 (62.0%)	25.97 (2.00x)	274 (0.95x)	171 (62.4%)	24.92 (2.00x)	274 (0.96x)	169 (61.7%)
	page size: 128	13.11 (1.00x)	265 (1.00x)	134 (50.8%)	12.98 (1.00x)	260 (1.00x)	132 (50.8%)	12.46 (1.00x)	263 (1.00x)	131 (50.0%)
	page size: 256	6.56 (0.50x)	267 (0.99x)	114 (42.7%)	6.49 (0.50x)	266 (0.98x)	114 (42.9%)	6.23 (0.50x)	271 (0.97x)	117 (43.2%)
	page size: 512	3.28 (0.25x)	286 (0.93x)	101 (35.3%)	3.25 (0.25x)	291 (0.89x)	100 (34.3%)	3.11 (0.25x)	293 (0.90x)	101 (34.5%)
Learned Index	2nd stage models: 10k	0.15 (0.01x)	98 (2.70x)	31 (31.6%)	0.15 (0.01x)	222 (1.17x)	29 (13.1%)	0.15 (0.01x)	178 (1.47x)	26 (14.6%)
	2nd stage models: 50k	0.76 (0.06x)	85 (3.11x)	39 (45.9%)	0.76 (0.06x)	162 (1.60x)	36 (22.2%)	0.76 (0.06x)	162 (1.62x)	35 (21.6%)
	2nd stage models: 100k	1.53 (0.12x)	82 (3.21x)	41 (50.2%)	1.53 (0.12x)	144 (1.81x)	39 (26.9%)	1.53 (0.12x)	152 (1.73x)	36 (23.7%)
	2nd stage models: 200k	3.05 (0.23x)	86 (3.08x)	50 (58.1%)	3.05 (0.24x)	126 (2.07x)	41 (32.5%)	3.05 (0.24x)	146 (1.79x)	40 (27.6%)

小结 / 思考

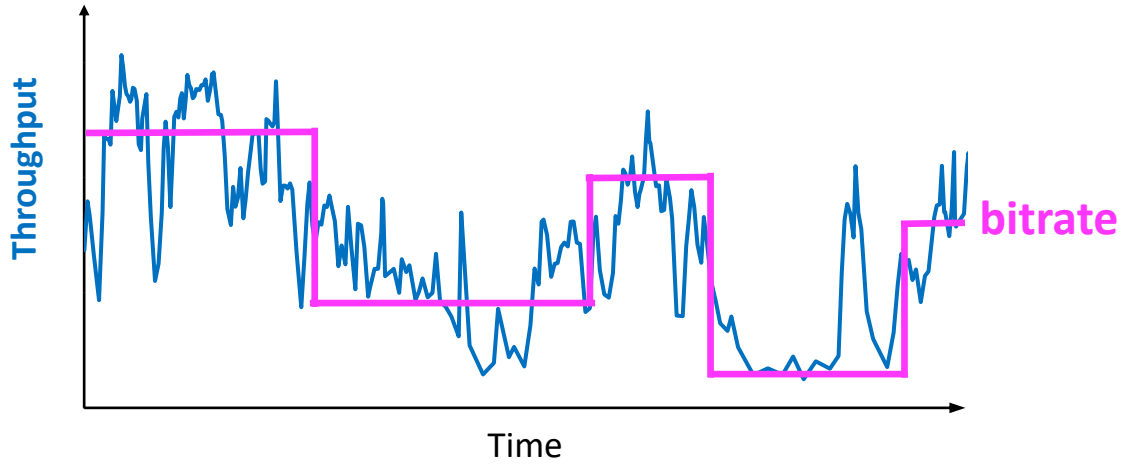
- **如果能把系统决策规划成预测或分类的问题，这些系统决策就有可能变成机器学习的场景**

- **盲目得使用机器学习不一定会有效果**
  1. Learned index 针对只读不写的场景
  2. 模型的复杂度可能提高推断的准确率，但也提高了所需要的时间
     - 比如论文里的 recursive model index

案例 2：视频流传输优化

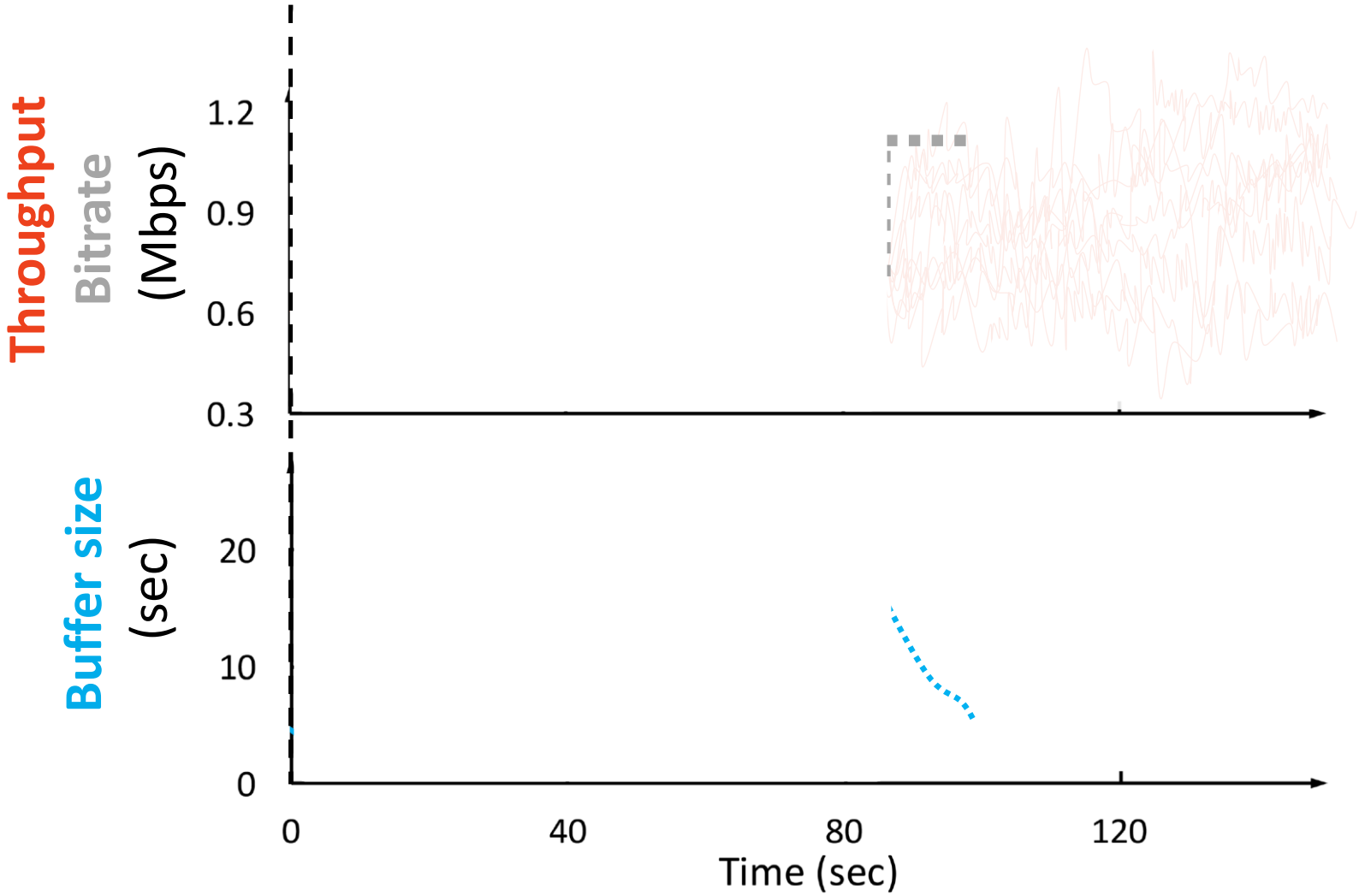


Adaptive Bitrate (ABR) Algorithms



Slide borrowed from Hongzi Mao (SIGCOMM '17) https://web.mit.edu/pensieve/content/pensieve-sigcomm17.pptx

视频流传输优化的难点



Network throughput is variable & uncertain

Conflicting QoE goals
- Bitrate
- Rebuffering time
- Smoothness

Cascading effects of decisions

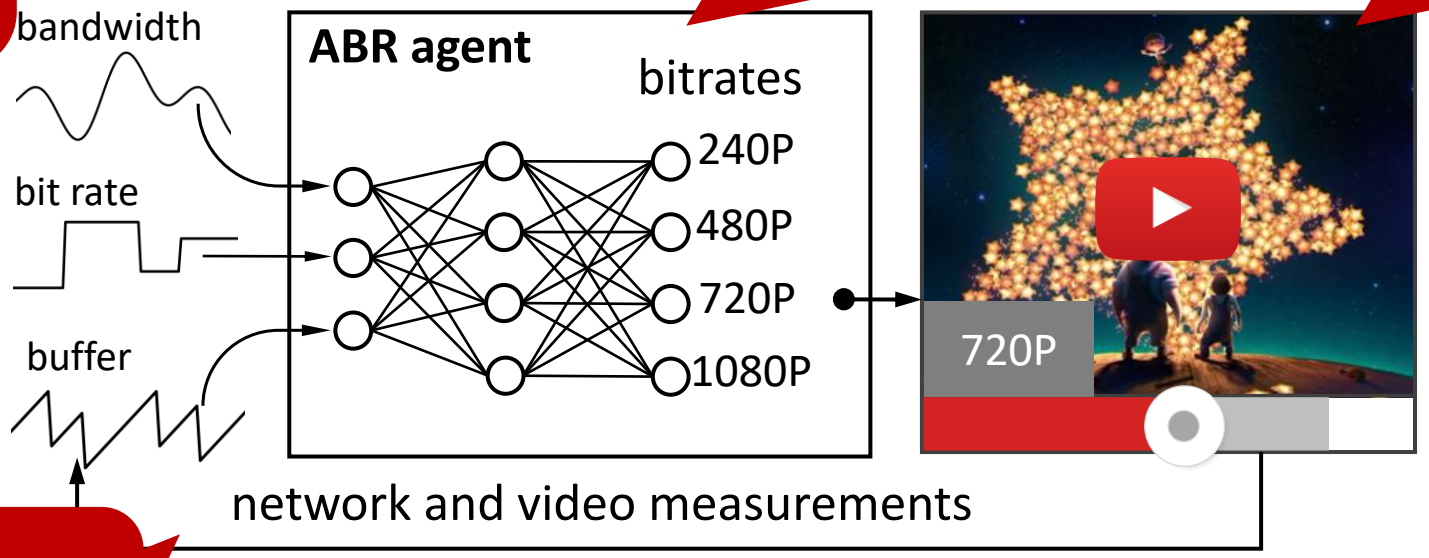
Slide borrowed from Hongzi Mao (SIGCOMM '17) https://web.mit.edu/pensieve/content/pensieve-sigcomm17.pptx

Pensieve

State space:
前面几块的比特率，下一块的不同码率下的大小，客户端当前缓存占用率，...

Action:
下一块的比特率

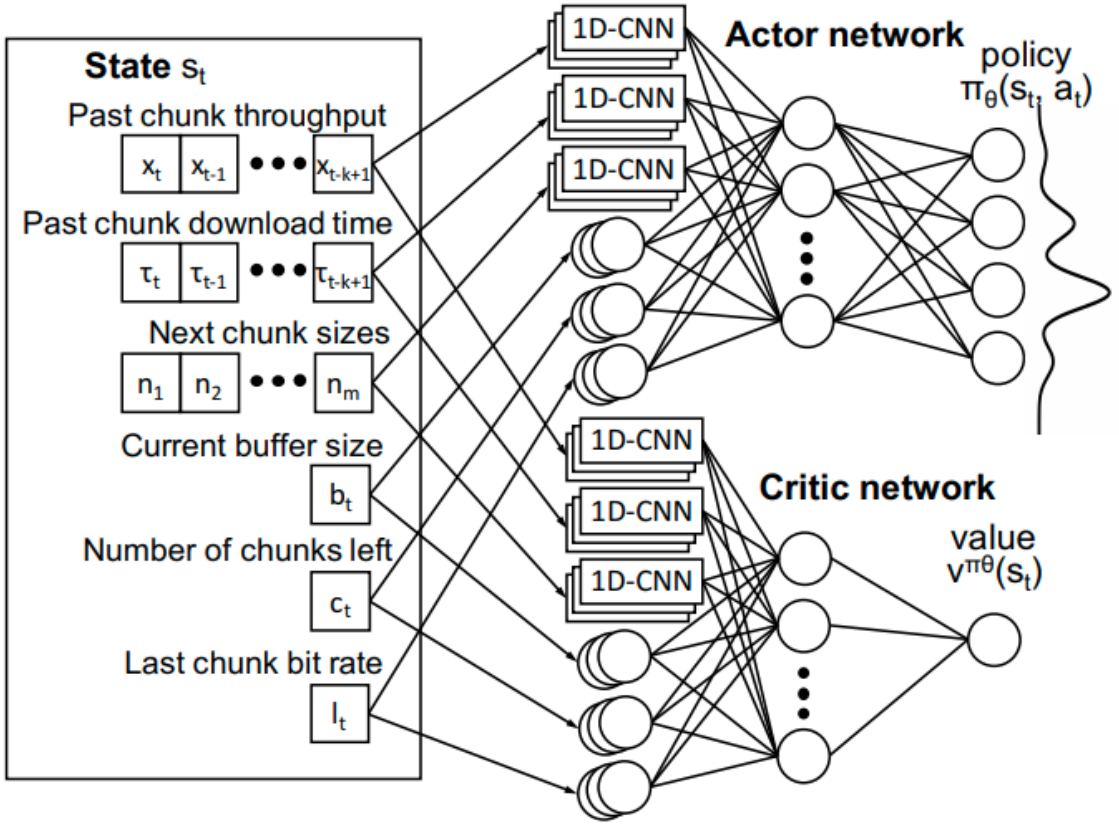
Environment



Reward:
体验质量（比如播放流畅度和画质）

"Neural Adaptive Video Streaming with Pensieve"
SIGCOMM '17
Mao et al.

Pensieve

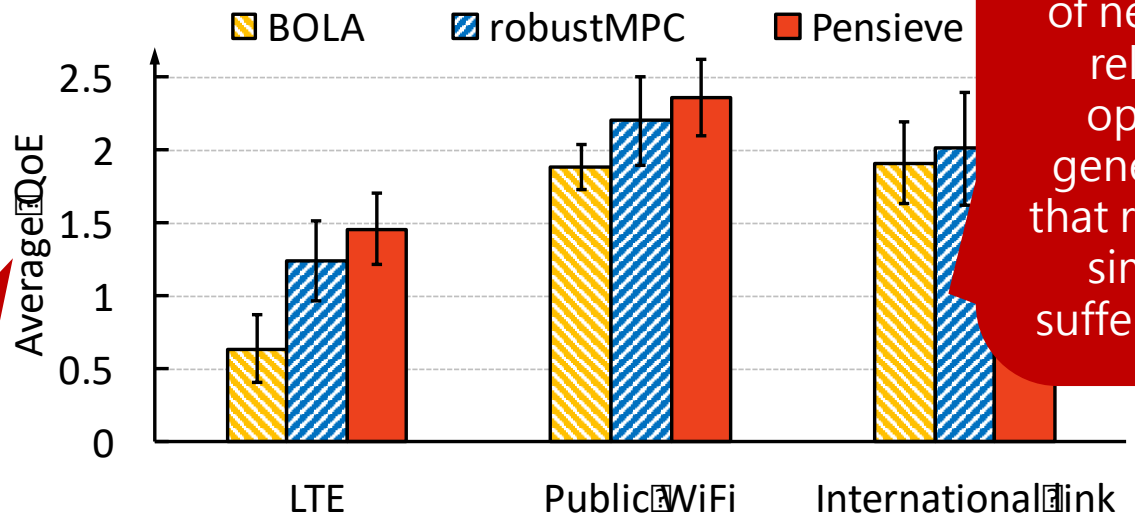


"Neural Adaptive Video Streaming with Pensieve"
SIGCOMM '17
Mao et al.

Pensieve

- **Comparison baselines:**
  1. BOLA – 考虑了 buffer occupancy observations
  2. MPC (Model Predictive Control) – 考虑了 buffer occupancy observations and network throughput prediction on 5 future chunks

QoE 考虑了 *(1)* 比特率大小，*(2)* 缓冲时间 (卡顿)，*(3)* 片段与片段之间的比特率平滑性



"MPC lacks an accurate model of network dynamics – thus it relies on simple and sub-optimal heuristics… More generally, any ABR algorithm that relies on fixed heuristics or simplified system models suffers from these limitations "

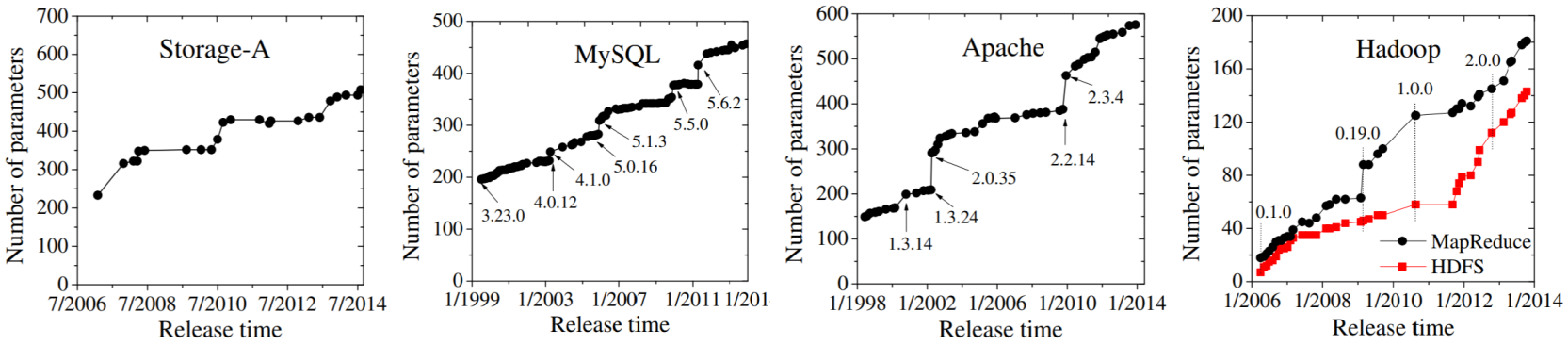
小结 / 思考

- **机器学习提供了一种与系统自我交互过程中学习的策略，并使得现代系统能实时地自适应环境**

- **盲目得使用机器学习不一定会有效果**
  - 当所需要学习的行为空间增大，建模的代价也会增加
  - 比如模型的准确率，数据集大小，模型复杂度，模型训练的时长
  - 系统工程师可以从经验，来制定学习的行为空间

案例 3：系统设定与参数调优

- 现代系统里很多的决策是系统工程师透过设定与参数来调整
- 多维的优化目标：系统作业处理延迟，系统资源平均使用率...
- 现代系统有越来越多的设定与参数

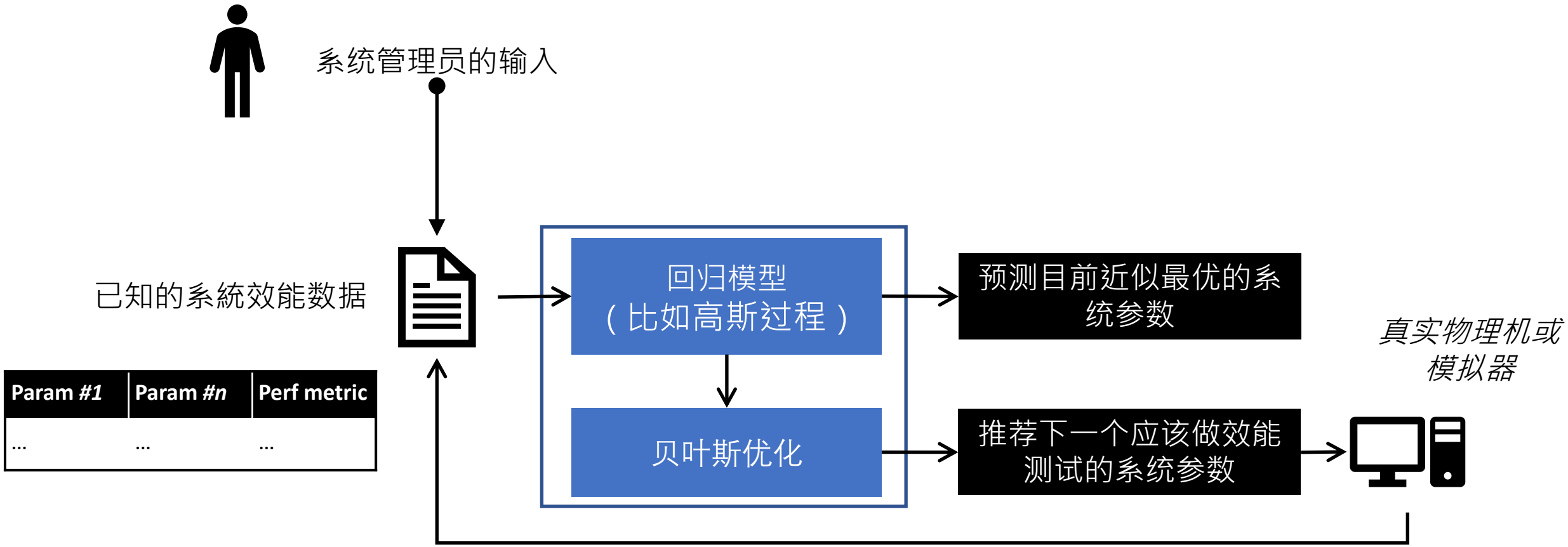


"Understanding and Dealing with over-Designed Configuration in System Software"
FSE '15
Xu et al.

系统调参的相关工作

	应用场景	机器学习算法
"OtterTune: Automatic Database Management ystem Tuning Through Large-scale Machine Learning", SIGMOD '17	数据库调优	1. Factor analysis 和 k-means clustering for workload characterization 2. Lasso for identifying important configuration knobs 3. Bayesian optimization and Gaussian process models for blackbox knob tuning
"CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics", NSDI '17	大数据分析机器的配置与花费的调优	Bayesian optimization and Gaussian process models
""Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms", OSDI '17	Azure 集群的负载特征来提高资源管理	1. Random forests for CPU utilization 2. Extreme gradient boosting trees for deployment size, VM lifetime, and workload class
"An end-to-end automatic cloud database tuning system using deep reinforcement learning", SIGMOD '19	数据库调优	Reinforcement learning
"AutoSys: The Design and Operation of Learning-Augmented Systems", ATC '20	Bing 搜索引擎的优化	Bayesian optimization and Gaussian process models
"MLGO: A Machine Learning Guided Compiler Optimizations Framework", '21	针对编译后代码大小的编译器优化	Reinforcement learning
…	…	…

贝叶斯优化和高斯回归的黑盒优化



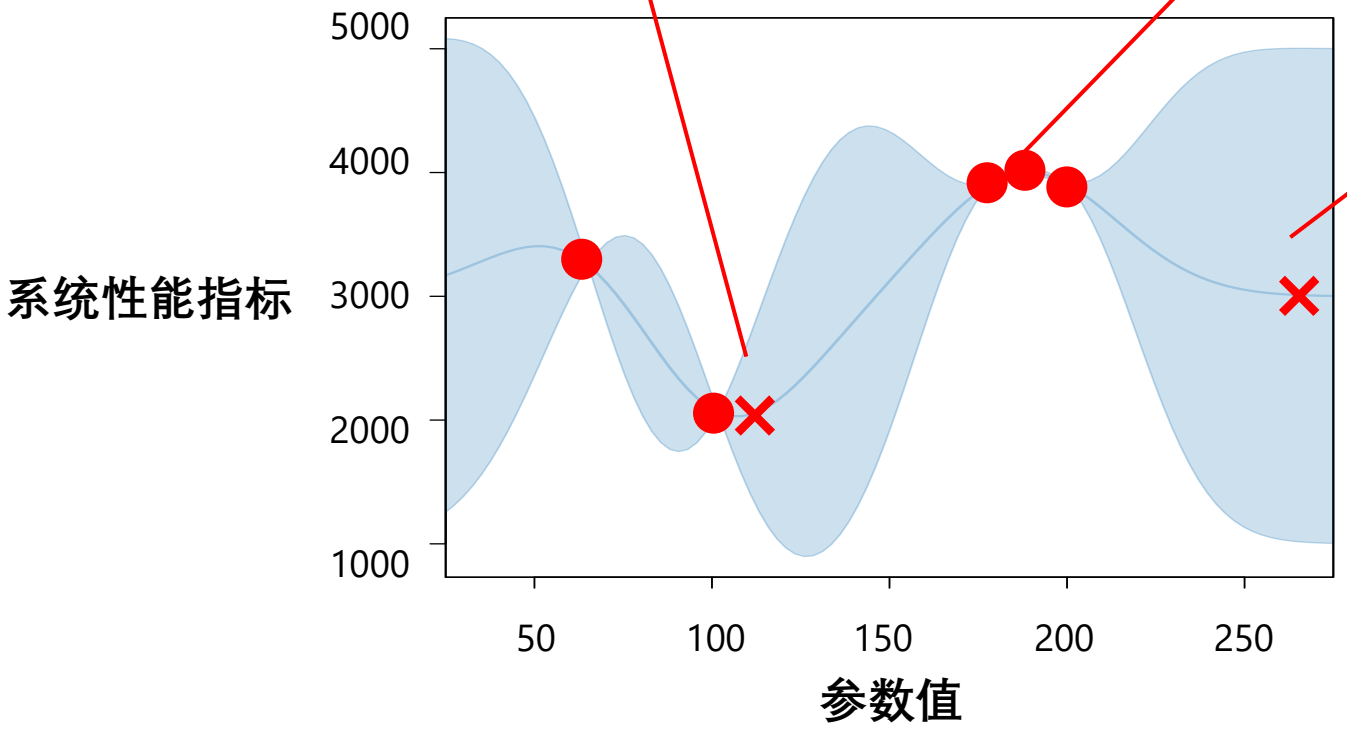
Exploitation, Exploration, and Re-sampling

Exploitation, Exploration, and Re-sampling

Exploitation: 选择现在可能最佳的点来采样

Re-sampling: 选择有可能有噪声或异常值的点来重新采样

Exploration: 选择现在不确定但未来可能会有高收益的点来采样



"OtterTune: Automatic Database Management ystem Tuning Through Large-scale Machine Learning"
SIGMOD '17
Van Aken et al.

"CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics"
NSDI '17
Alipourfard et al.

"Metis: Robustly Optimizing Tail Latencies of Cloud Systems"
ATC '18
Liang et al.

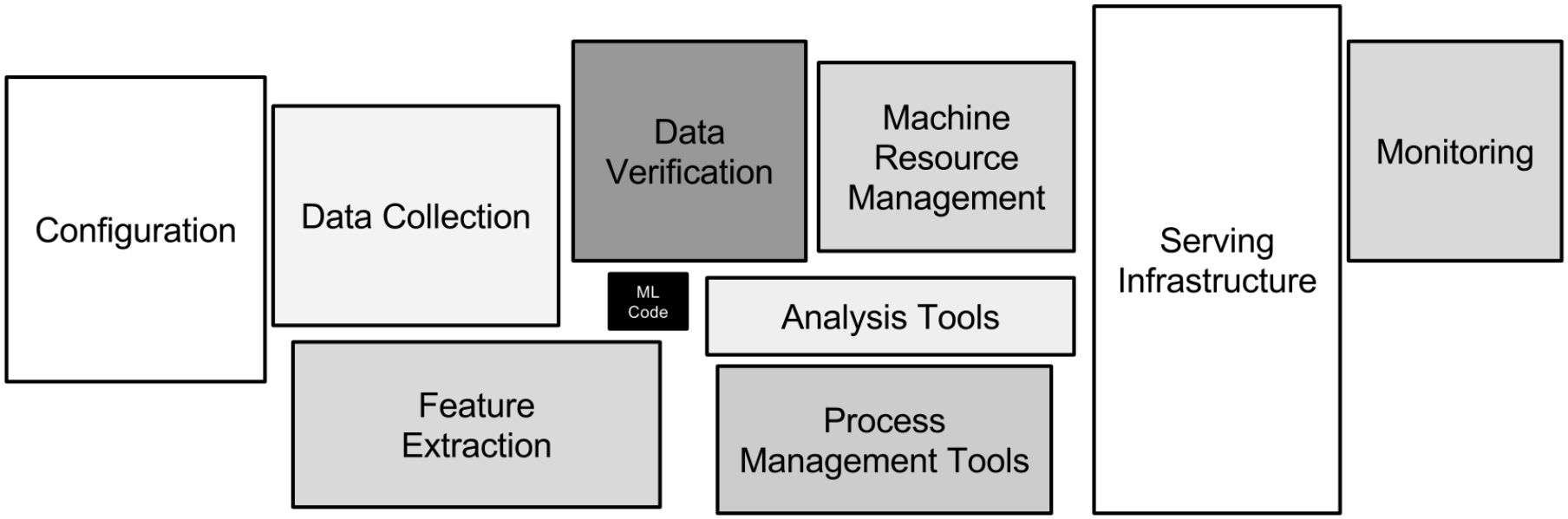
小结 / 思考

- **深度学习不是唯一的工具，传统机器学习方法也能优化系统场景**

- **系统正确性**
  - 模型不确定性会影响系统正确性
  - 机器学习太过于激进会影响系统的稳定性

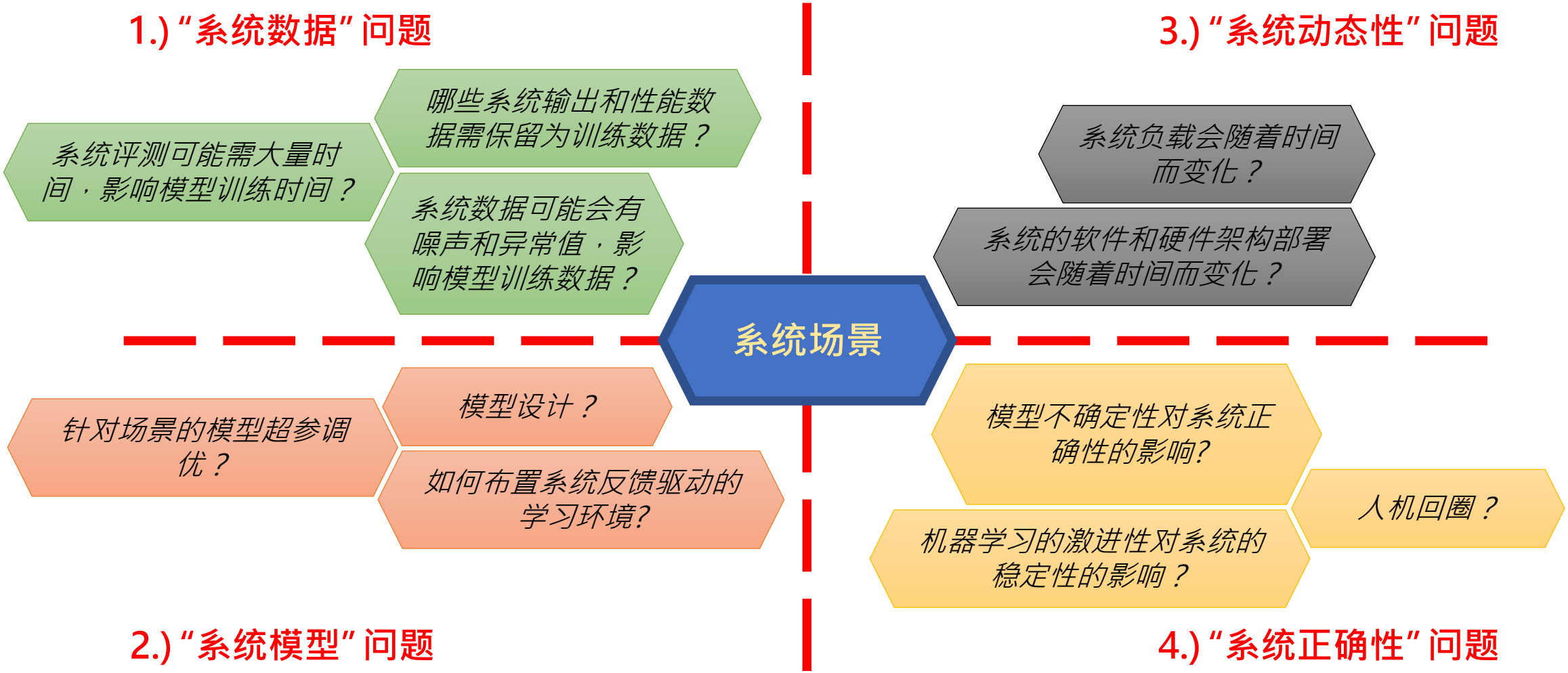
- **如何利用自动化的思维来弥补系统工程师在机器学习经验上的不足？**

利用人工智能来优化计算机系统
不只是选取模型



"Hidden Technical Debt in Machine Learning Systems"
NIPS 2015
D. Sculley et al.

落地的考虑要素和痛点


1.)“系统数据”问题
哪些系统输出和性能数据需保留为训练数据？
系统评测可能需大量时间，影响模型训练时间？
系统数据可能会有噪声和异常值，影响模型训练数据？
3.)“系统动态性”问题
系统负载会随着时间而变化？
系统的软件和硬件架构部署会随着时间而变化？
系统场景
模型设计？
针对场景的模型超参调优？
如何布置系统反馈驱动的学习环境?
模型不确定性对系统正确性的影响?
人机回圈？
机器学习的激进性对系统的稳定性的影响？
2.)“系统模型”问题
4.)“系统正确性”问题

“系统数据”问题

- **系统评测可能需大量时间...**
  - 比如：资料库的缓存需要暖机
  - 研究课题：如何降低系统评测的次数（比如 exploration-exploitation）？
- **系统能输出大量的系统数据...**
  - 比如：RocksDB 有 50+ 设定参数，能记录 100+ 性能指标
  - 研究课题：如何判断哪些系统数据需保留为训练数据集？
- **系统数据可能会有噪声和异常值...**
  - 比如：时间类性能指标
  - 研究课题：如何学习系统的正常方差

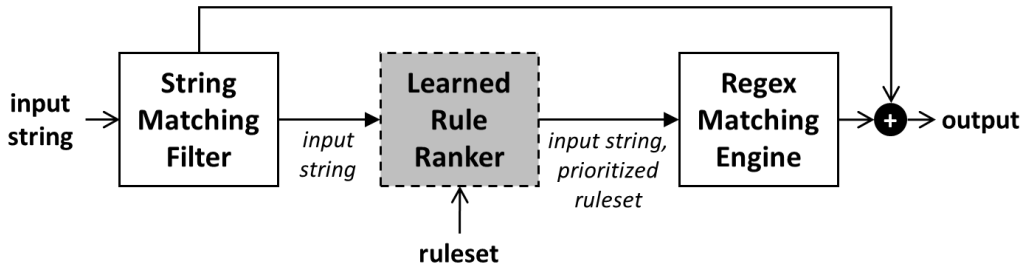
“系统模型”问题

- **系统反馈驱动的学习环境...**
  - 比如：线上/线下真实系统环境，仿真/模拟系统环境
  - 研究课题：如何布置这些环境？如何迁移不同环境下学到得知识？
- **针对场景的模型超参调优...**
  - 比如：模型的推断延迟，准确率，资源使用率
  - 研究课题：自动机器学习（AutoML）
- **针对场景的模型设计...**
  - 比如：（非深度/深度）模型架构的选择，有效模型输入的选择
  - 研究课题：自动机器学习（AutoML），特征工程

“系统模型”问题案例：Learned Ranker

- 规则匹配引擎的效率能影响防火墙的性能
  - 基本上，防火墙为每个网络数据检查每一条过滤规则，直到
- Learned ranker 可学习过滤规则的特征，来预测一个
  符合的过滤规则，进而为规则匹配引擎预排序规则

复杂的模型可以提升排序的准确率。但是系统整体的性能必须考虑模型的额外延迟



Model	Top-1	Top-3	Top-5	Latency (μs)
DNN(128)	81.85%	94.39%	97.05%	11.65
DNN(256)	83.37%	95.18%	97.45%	14.68
DNN(512)	83.72%	95.61%	97.52%	21.44
RNN(128)	89.44%	97.51%	98.82%	33.43
RNN(64)	92.98%	98.55%	99.30%	39.88
RNN(32)	95.02%	99.23%	99.71%	48.02
LR	67.69%	82.89%	88.18%	48.25

“Accelerating Rule-matching Systems with Learned Rankers”
ATC ‘19
Li et al.

“系统动态性”问题

- **系统负载会随着时间而变化？**
  - 比如：用户搜索关键字
  - 研究课题： 终身学习，增量学习，迁移学习

- **系统的软件和硬件架构部署会随着时间而变化？**
  - 比如：多租户伺服器，系统横向扩展，软件/硬件升级
  - 研究课题： 针对机器学习的系统抽象

“系统正确性”问题

- **模型对系统正确性的影响?**
  - 比如：模型推断结果的不确定性
  - 研究课题：模型测试和验证，模型可解释性
- **机器学习的激进性对系统的稳定性的影响？**
  - 比如：频繁的改动系统设定可能造成系统无法处于稳定状态
  - 研究课题：人机回圈
- **人机回圈？**
  - 比如：专家的经验可能存在偏差和误差
  - 研究课题：如何采样专家的经验？

小结 / 思考

- **利用人工智能来优化计算机系统不只是选取模型**
  - “系统数据” 问题
  - “系统模型” 问题
  - “系统动态性” 问题
  - “系统正确性” 问题
- **利用自动化机器学习的思维来帮助系统工程师**
  - 一套针对 AI-for-Systems 的方法论，框架，和工具链

课程主要内容回顾

- **现代系统带来的挑战**
- **应用人工智能来优化现代系统的案例**
  - 案例 #1: 数据库索引
  - 案例 #2: 视频流传输
  - 案例 #3: 系统选项与参数调优
- **落地的考虑要素和痛点**