Scalable and Efficient Scheduling for Large Scale Kubernetes Clusters

Haifeng Liu, **Yuan Chen**

京东 JD.COM

About JD.com

China's largest online and overall retailer and biggest Internet company by revenue

- 300 million+ active users
- 2018 revenue: $67.2 billion

China's largest e-commerce logistics infrastructure and unrivalled nationwide fulfillment network

- 550+ warehouses
- Covering 99% of population
- Standard same-and next day delivery

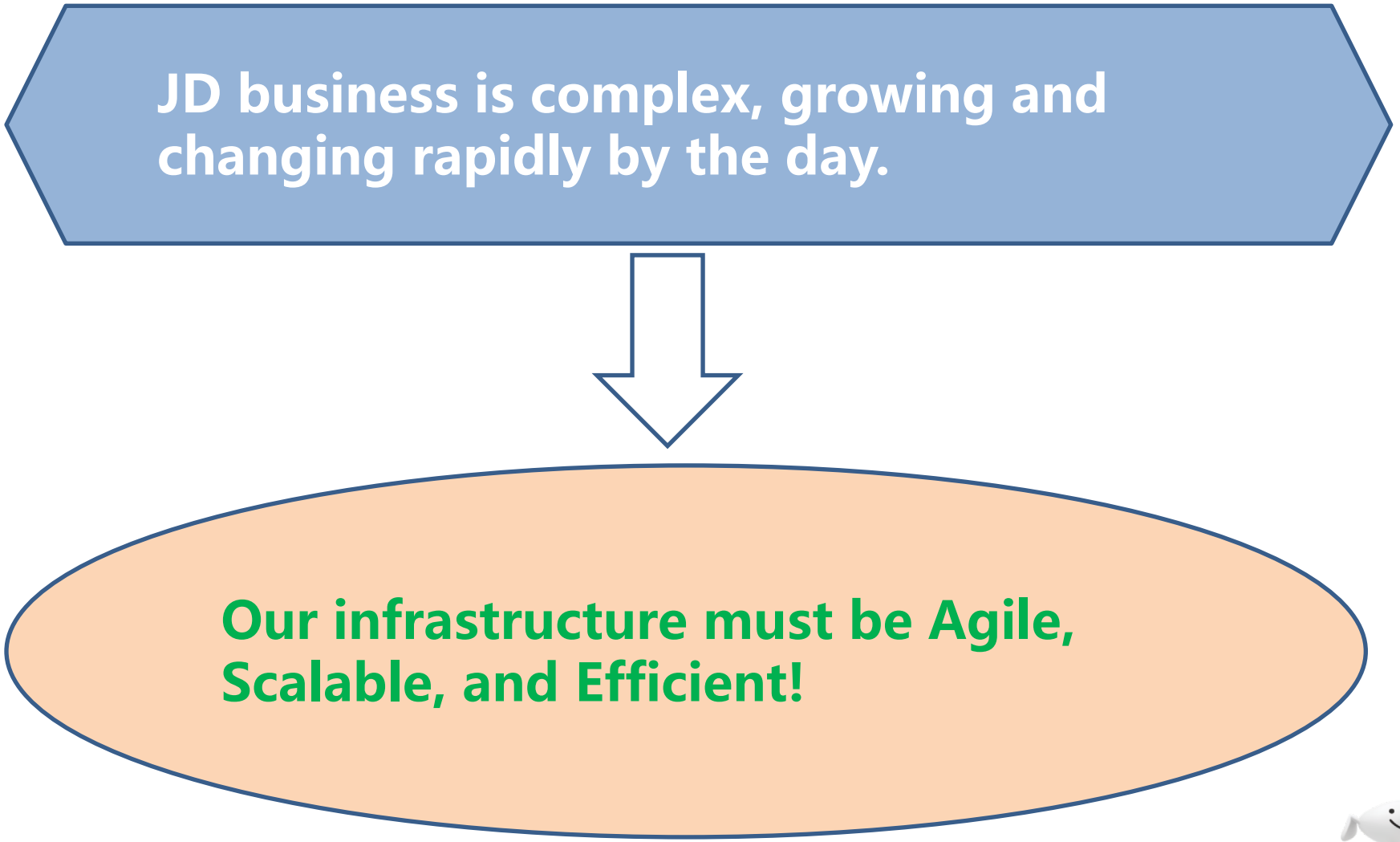
First Chinese internet company to make the Fortune Global 500

Strategic partnerships

Tencent Walmart Google

京东

Challenges



京东

JD Retail Technological Infrastructure Group

Provide and manage hyperscale containerized infrastructures and platforms for all JD services

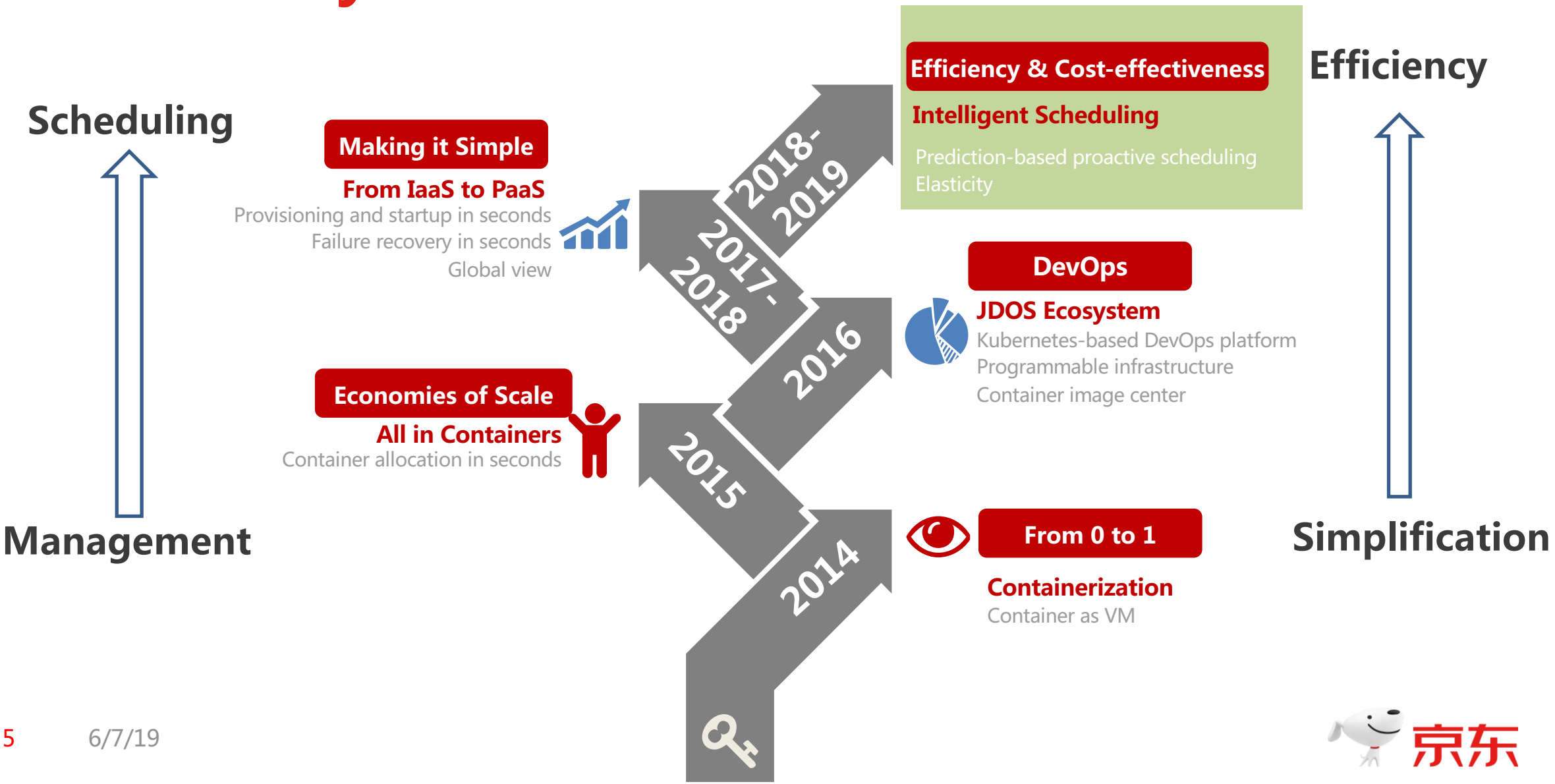
- One of the earliest adopters of Kubernetes
- Run one of the largest Kubernetes clusters in production in the world
- CNCF Platinum Member
- 2018 CNCF End User Award

http://tig.jd.com/en

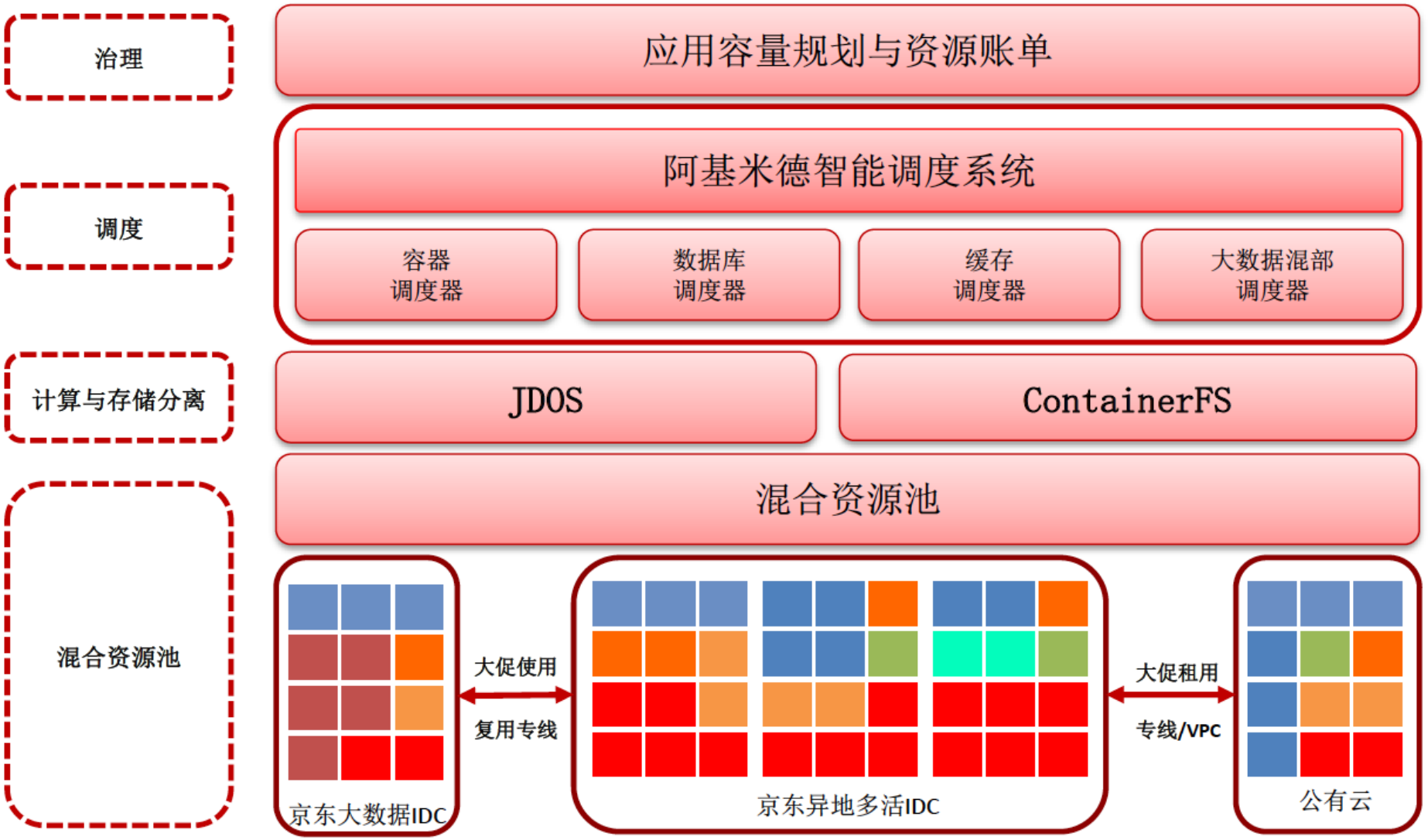
"We are thrilled to have JD... By sharing their Kubernetes experiences and investing directly in the project, JD.com is helping to spread cloud native computing throughout China", -Dan Kohn, Executive Director of the CNCF

京东

Our Journey


Scheduling
Management
Efficiency
Simplification
Making it Simple
From IaaS to PaaS
Provisioning and startup in seconds
Failure recovery in seconds
Global view
Economies of Scale
All in Containers
Container allocation in seconds
Efficiency & Cost-effectiveness
Intelligent Scheduling
Prediction-based proactive scheduling
Elasticity
DevOps
JDOS Ecosystem
Kubernetes-based DevOps platform
Programmable infrastructure
Container image center
From 0 to 1
Containerization
Container as VM
2018-2019
2017-2018
2016
2015
2014

JDOS (JD Data Center OS) Architecture



京东

Resource Management: Challenges

Accuracy

- Users cannot accurately estimate their resource needs (often overprovisioning)
- Resource demand of online services varies a lot. **One size does not fit all!**
  - More - > waste
  - Less -> SLO violations

Complexity

- Stateful pods (e.g., databases)
- Multi-dimensional resources
- Heterogeneous workloads: online service, big data, AI, IoT

Scalability (Performance)

- A single cluster with 10K+ nodes and 100K+ pods

京东

Archimedes: Scalable and Efficient Scheduling

Performance and Scalability

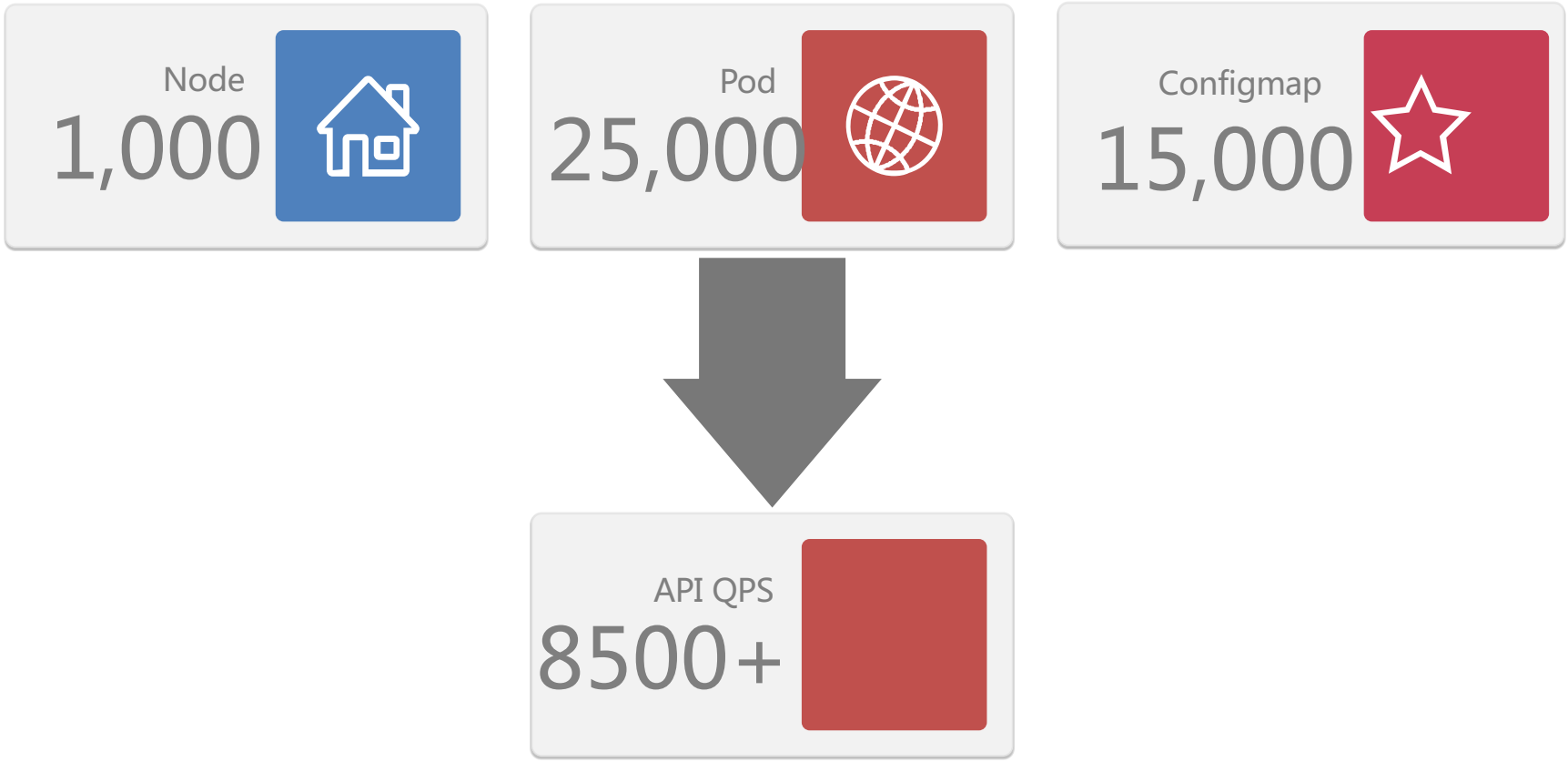
- Customization: API servers, etcd, scheduler, image
- Advanced optimization
  - Preemption and admission control
  - Group scheduling and binding
  - Adaptive scheduling

Quality and Resource Efficiency: data-driven intelligent scheduling

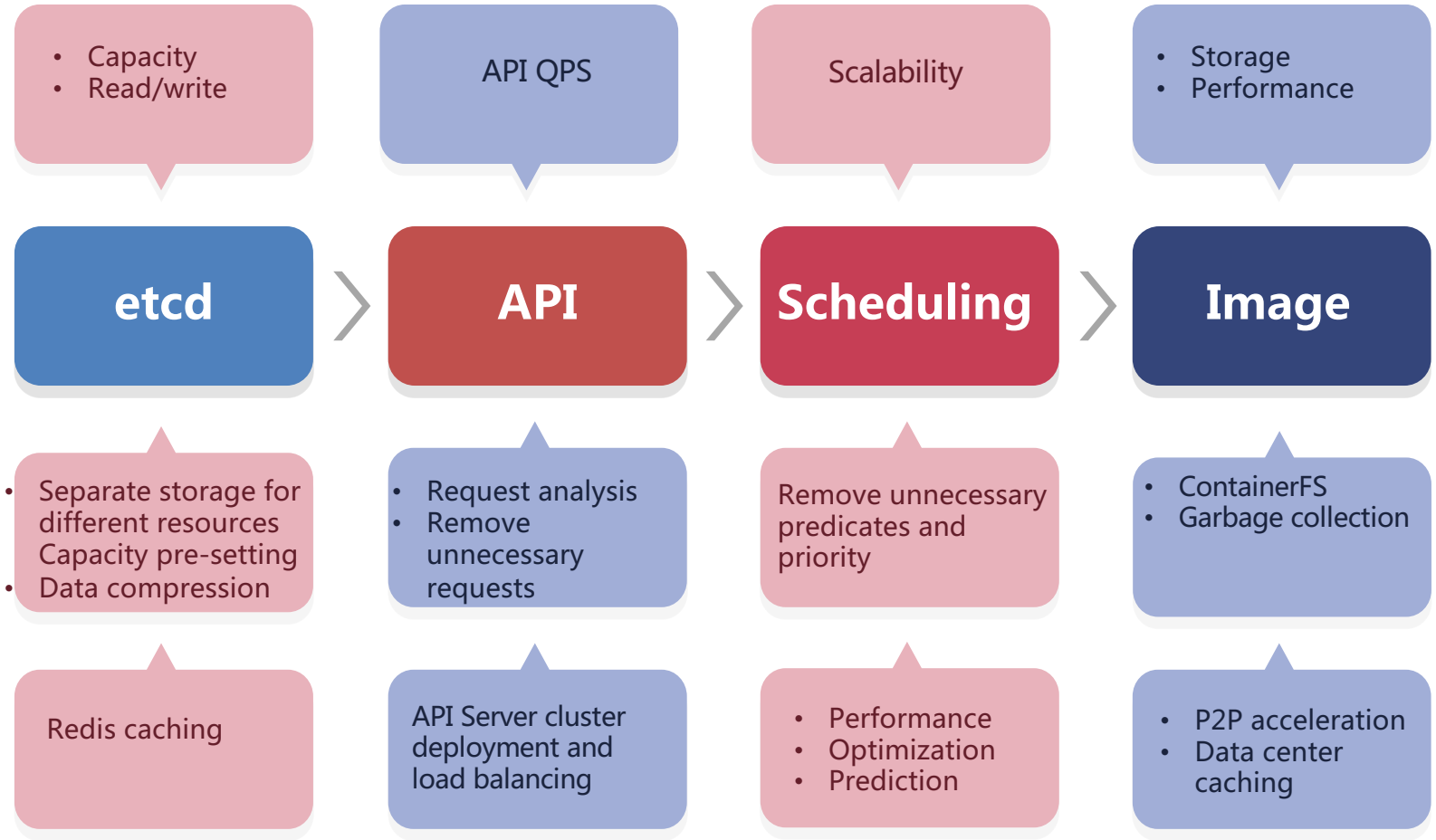
  - historical usage analysis and prediction-driven scheduling
  - mixed workload placement
  - auto scaling with hybrid resource pools

京东

A Typical JDOS Cluster



Bottleneck Analysis



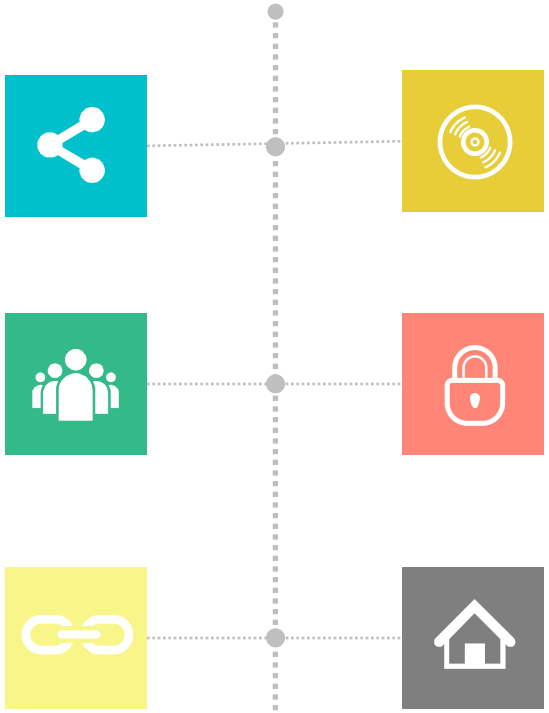
京东

Customization and Optimization

etcd capacity monitoring
compression

different etcd for
different resource-types

heartbeat optimization



APIserver request analysis
configmap requests: 99%

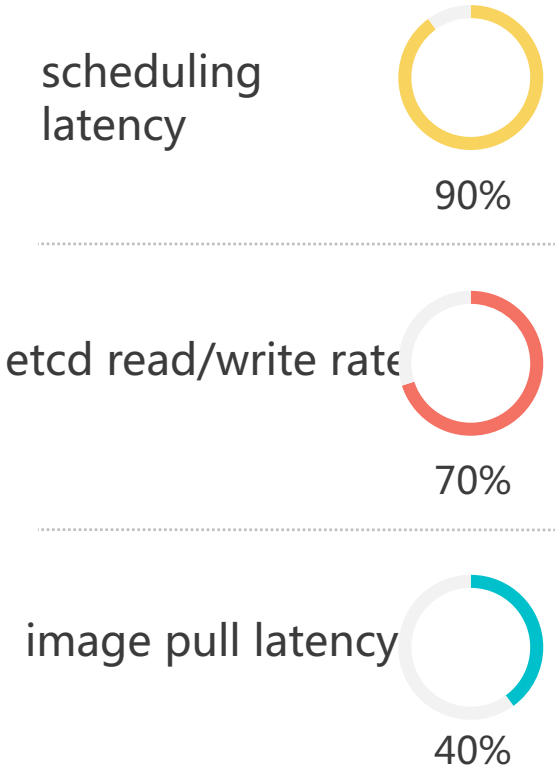
GET requests
Redis cache for etcd

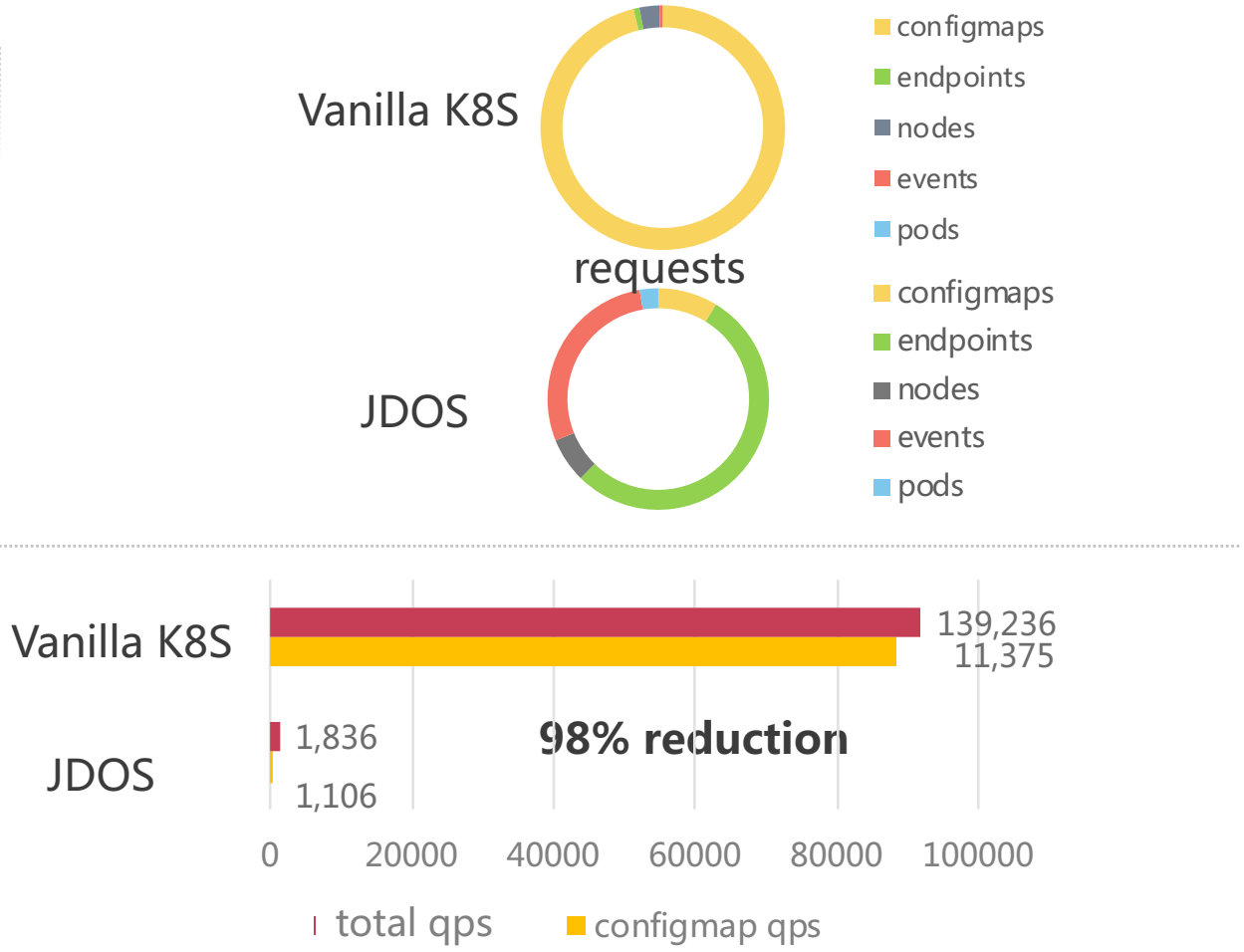
APIServer load balancing

- **Separate storage for different resource types**
- **Compression**
- **Redis cache**

- **Remove unnecessary requests**
- **Load balancing**

Results

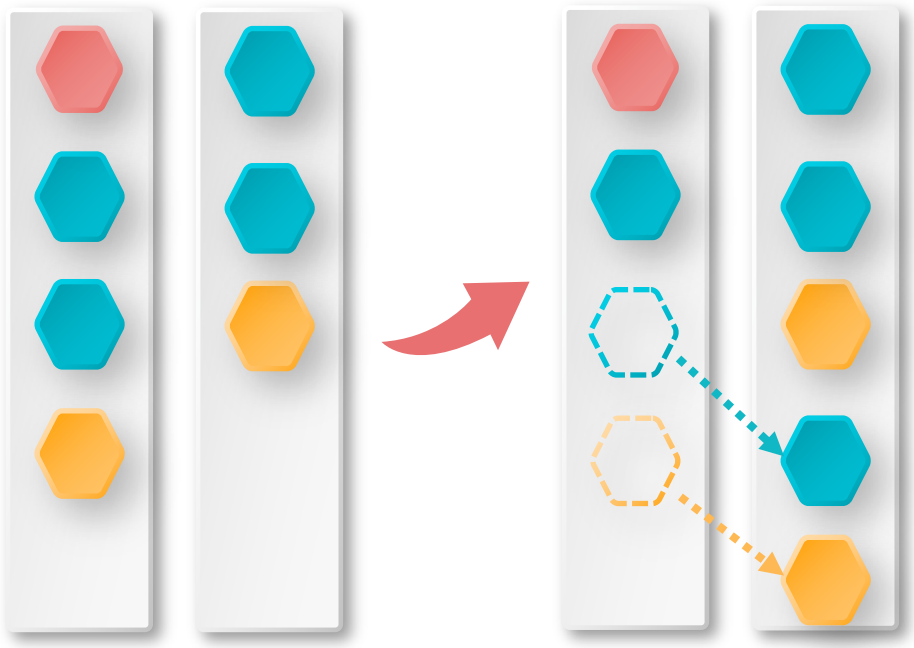

scheduling latency
90%
etcd read/write rate
70%
image pull latency
40%


Vanilla K8S
JDOS
requests
configmaps
endpoints
nodes
events
pods
configmaps
endpoints
nodes
events
pods
Vanilla K8S
JDOS
139,236
11,375
1,836
1,106
98% reduction
0
20000
40000
60000
80000
100000
total qps
configmap qps

京东

Preemption and Admission Control

Preemption Optimization



- Resource demand prediction
- Priority and SLOs-based preemption
- Proactive preemption
- Preempted apps in scheduling queues
- Overloaded host locked

Admission Control

Reject low priority request.

No requeue after a certain number of retries.

京东

Group Scheduling and Binding

Problem

- A large number of pod requests from an application (e.g., horizontal auto-scaling) cause a significant scheduling delay.

Solution : group scheduling

- Schedule an application group consisting of a number of pods with same/similar resource requirements at the same time.
- Use a representative pod resource requirement to evaluate nodes' priorities to obtain top k (k<=m) candidates.
- Assign and bind the m pods to the best k nodes simultaneously.

Other solution: Poseidon

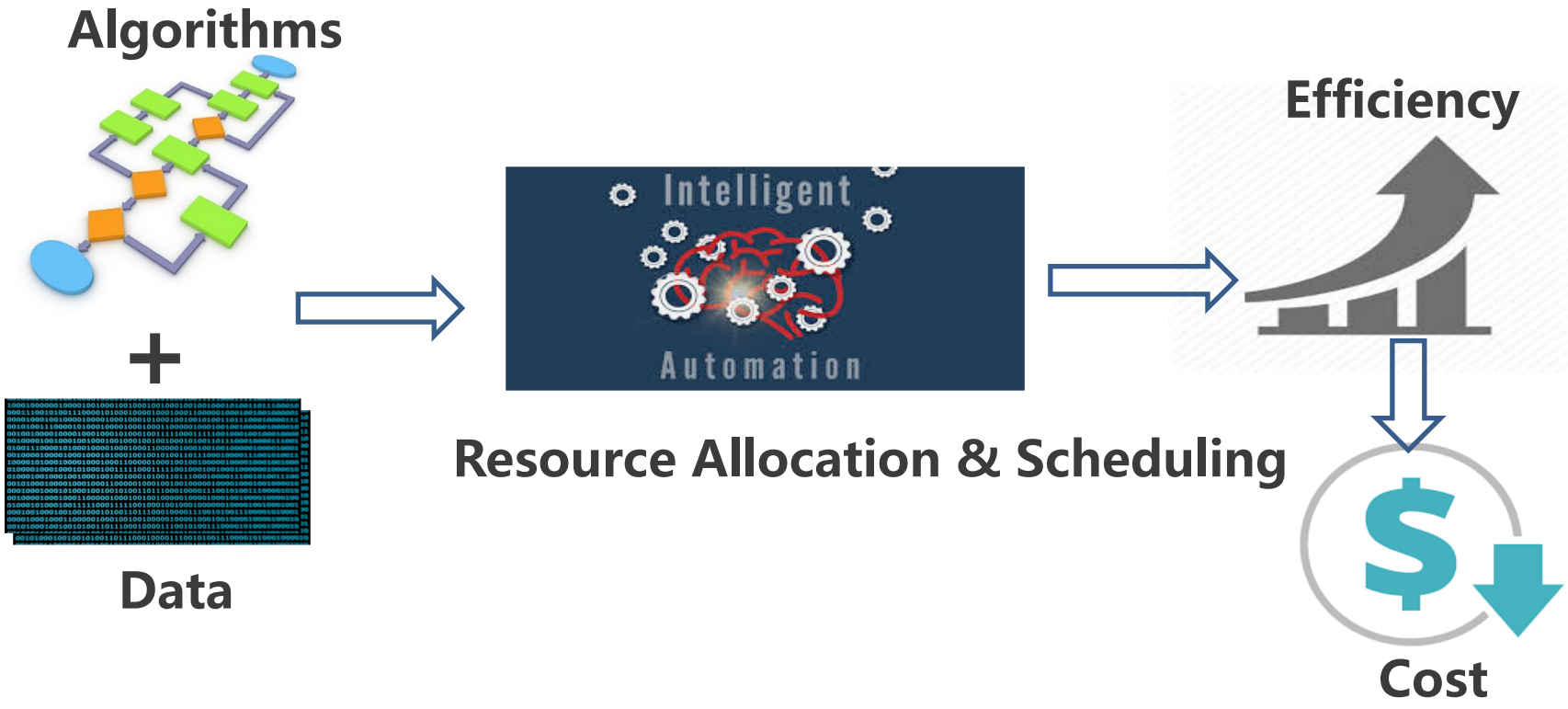
Adaptive Scheduling

Different algorithms and/or parameters for different workloads and scenarios.

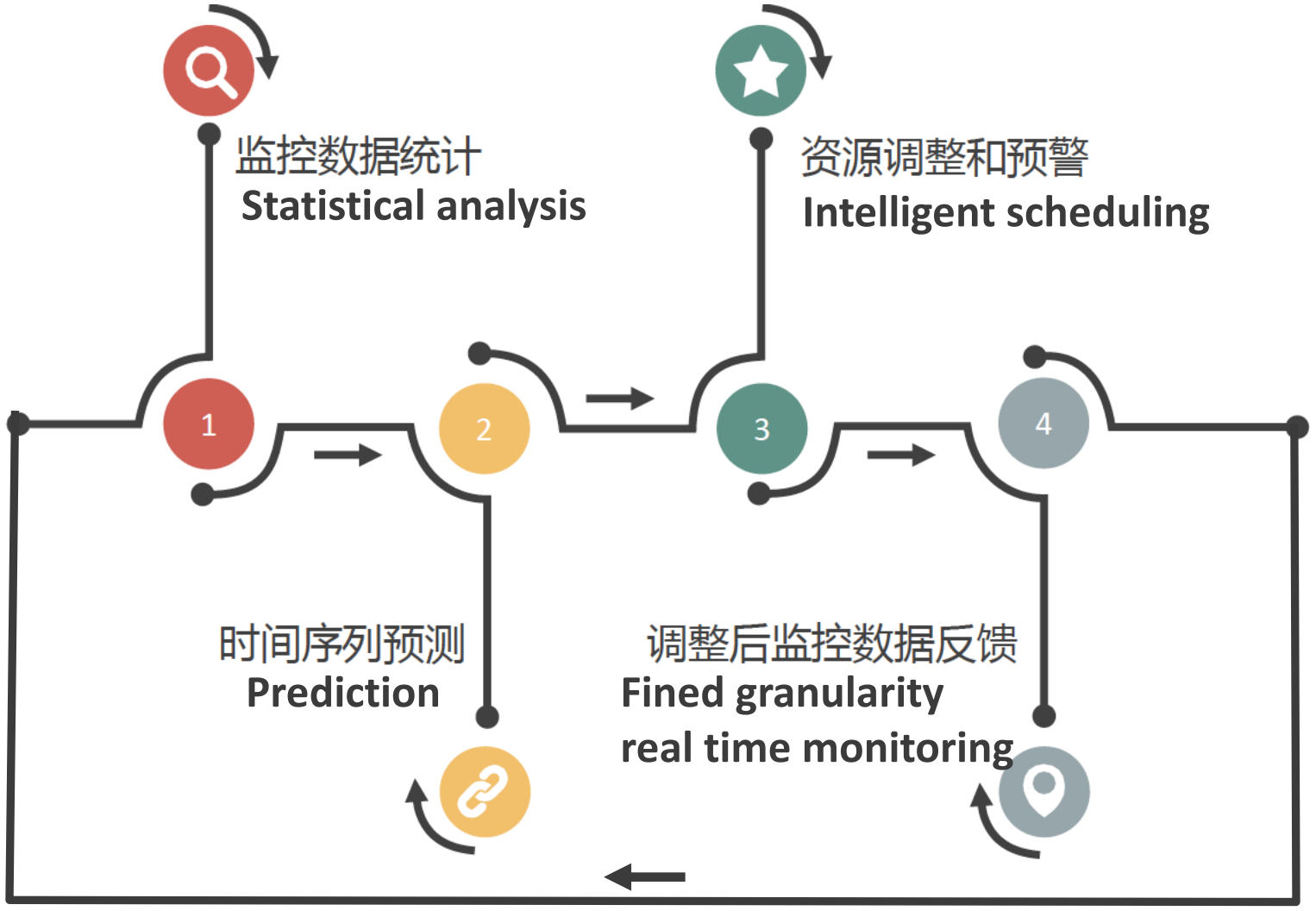
- Scheduling/rescheduling long running critical services
  - Advanced and accurate scheduling
  - A large number of candidate nodes
- Auto-scaling and one-time batch jobs
  - Simple and group scheduling
  - A small number of candidate nodes
- Cache scheduling decisions

Archimedes: Intelligent Resource Management

Utilize advanced model and machine learning algorithms to **automatically** and **accurately** allocate, schedule and adjust Pod resources to maximize the resource utilization while satisfying the SLOs.



A Closed Loop System



京东

Advanced Scheduling

CPU and Memory sizing

- Metadata data, historical resource usage statistics and future demand prediction
- Proactive adjustment

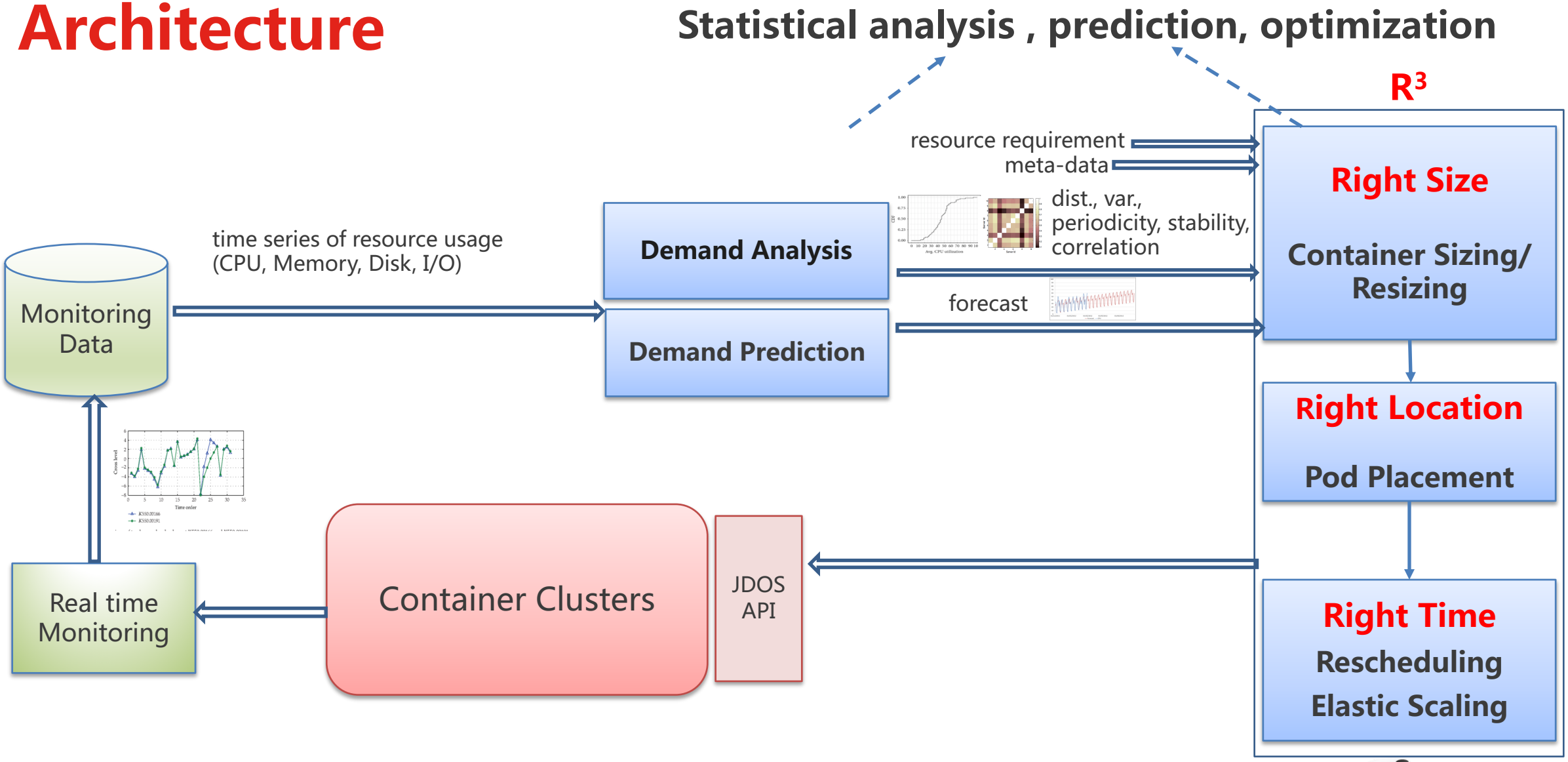
Host selection :resource balance, resource availability, pod affinity

Mixed placement of workloads and preemption optimization

Resource scaling across multiple hybrid resource pools

京东

Architecture



Right Sizing: Oversubscription with Performance Guarantees

K8S

- HPA alone does not solve the resource efficiency problem.
  - Right size is needed to save resources.
  - Not work for stateful pod.
- VPA implementation is too simple.
  - Request only
  - Recreate pod upon update
  - Thrashing
  - OOM

Archimedes

Advanced models and algorithms for setting correct size for Pods

- Meta data based initial size
- Statistical features of historical data + future demand prediction based (re)sizing
  - CPU: request and limit + headroom
  - Memory: peak and outlier prediction + hierarchical sub-clusters
  - Disk resource management for databases
- **High utilization, low SLOs violations, no OOM, stable**

京东

Right Sizing

- Estimate the workload demand.
  - New workload
    - Resource requirements
    - Meta-data based classification + group resource requirements
  - Existing workload
    - Historical usage analysis + prediction (ARIMA, LSTM, …)
- Allocate or adjust the workload resource based on the estimates.
- Reschedule the workload if needed.

Right Sizing of CPU Resource

Adaptive tail-bound based resource request and limit

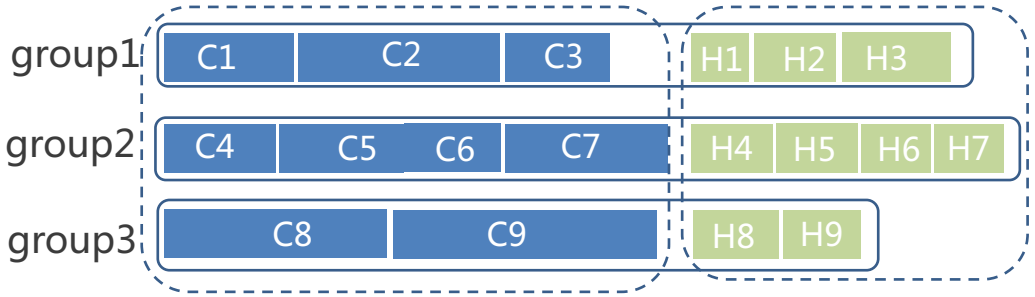
- Guaranteed resource (request): $request_i$ = 90%ile
- Maximum resource(limit): $headroom_i$ = maximum (99%ile)
- Opportunistic headroom: $headroom_i = limit_i - request_i$

Shared headroom on host (max_of_sum < sum_of _max)

- Correlated containers: $headroom = headroom_i + headroom_j$
- Uncorrelated containers: $headroom = max(headroom_i, headroom_j)$
- Shared headroom: $shared\_headroom <= \sum_{i=1}^{N} headroom_i$

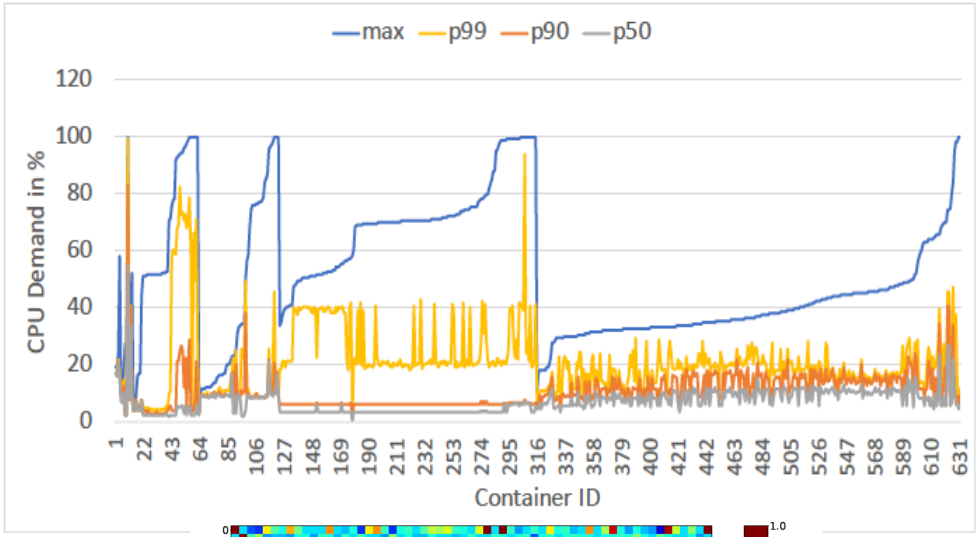
Scheduling constraint

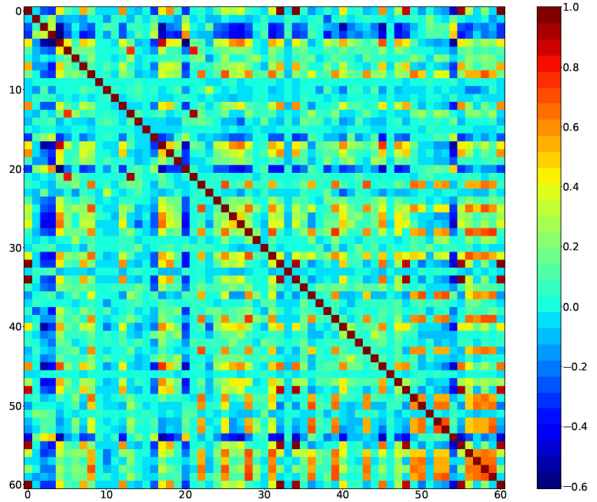
- $\sum_{i=1}^{M} request_i + shared\_headroom <= capacity$



Request= C1+C2+...+C9

Headroom= MAX (H1+H2+H3, H4+H5+H6+H7, H8+H9)

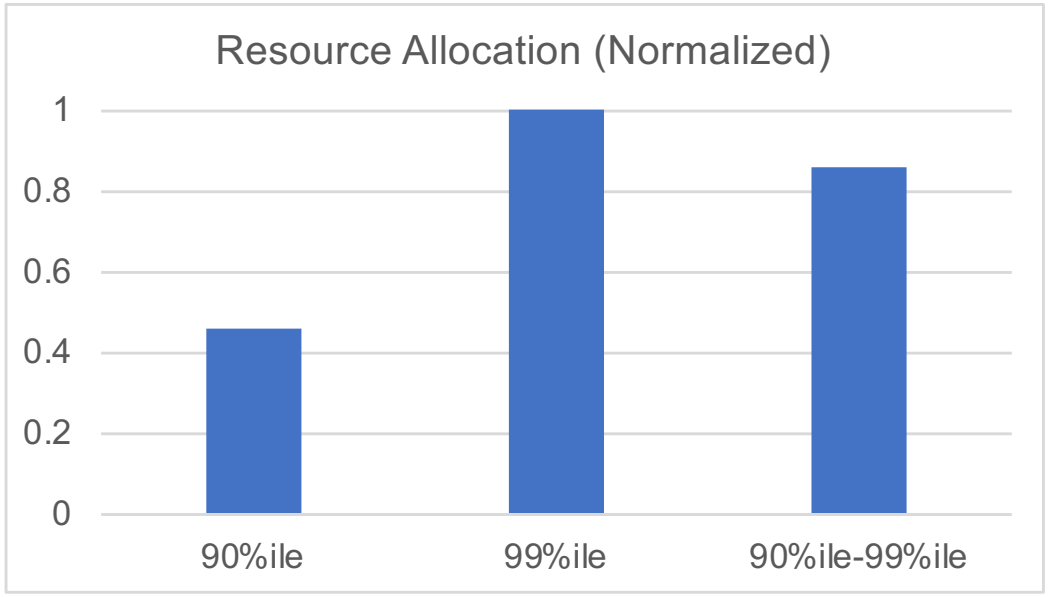


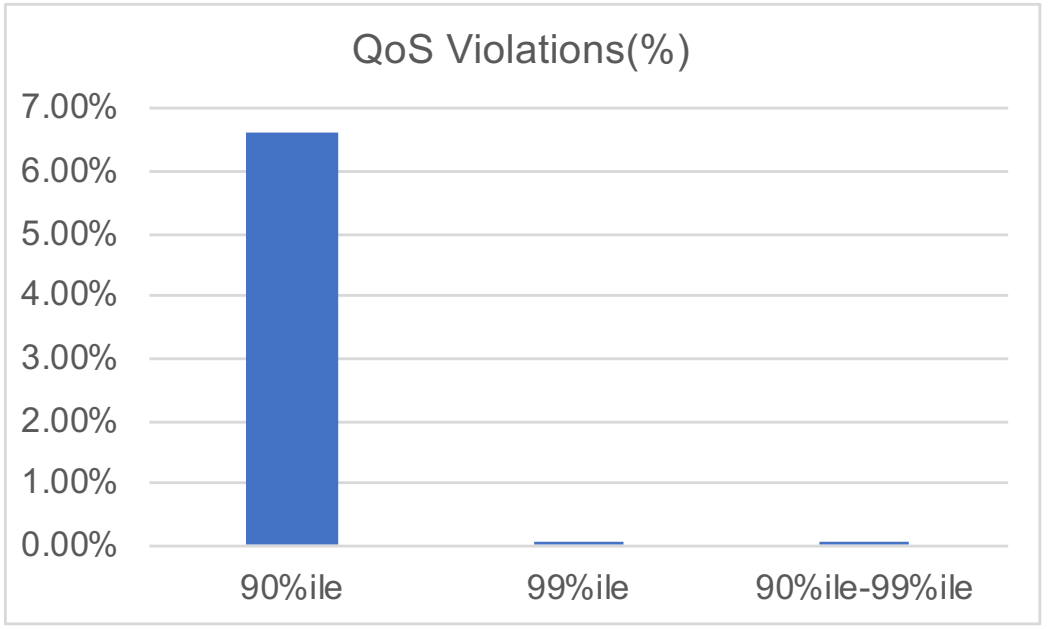


2-3X improvement without performance degradation!

京东

Resource Usage and Performance Tradeoff





京东

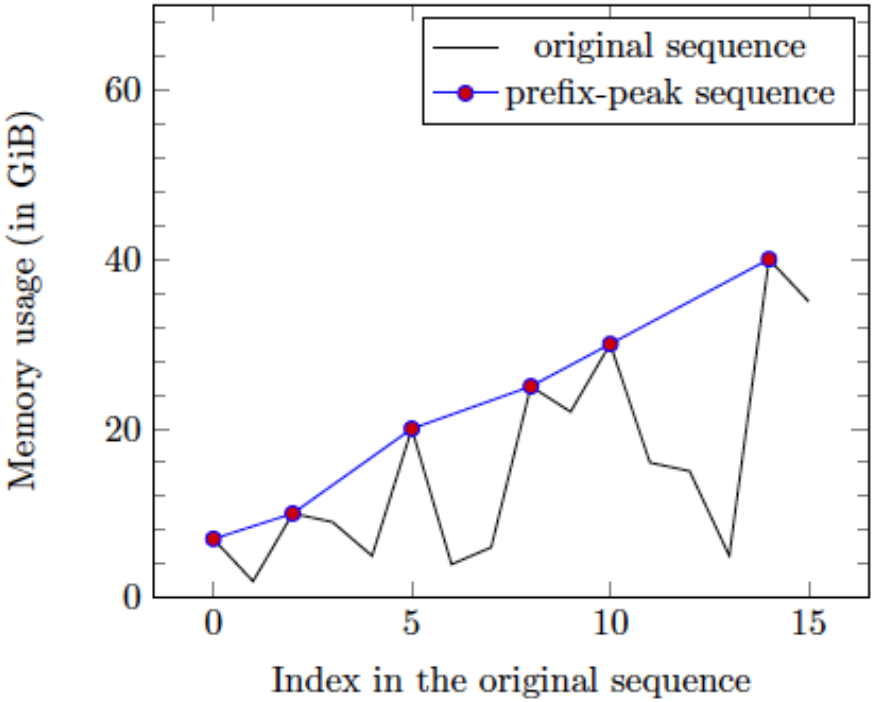
Right Sizing of Memory Resource

Goal: Maximize memory utilization without OOMs

- **Estimate the next peak demand**

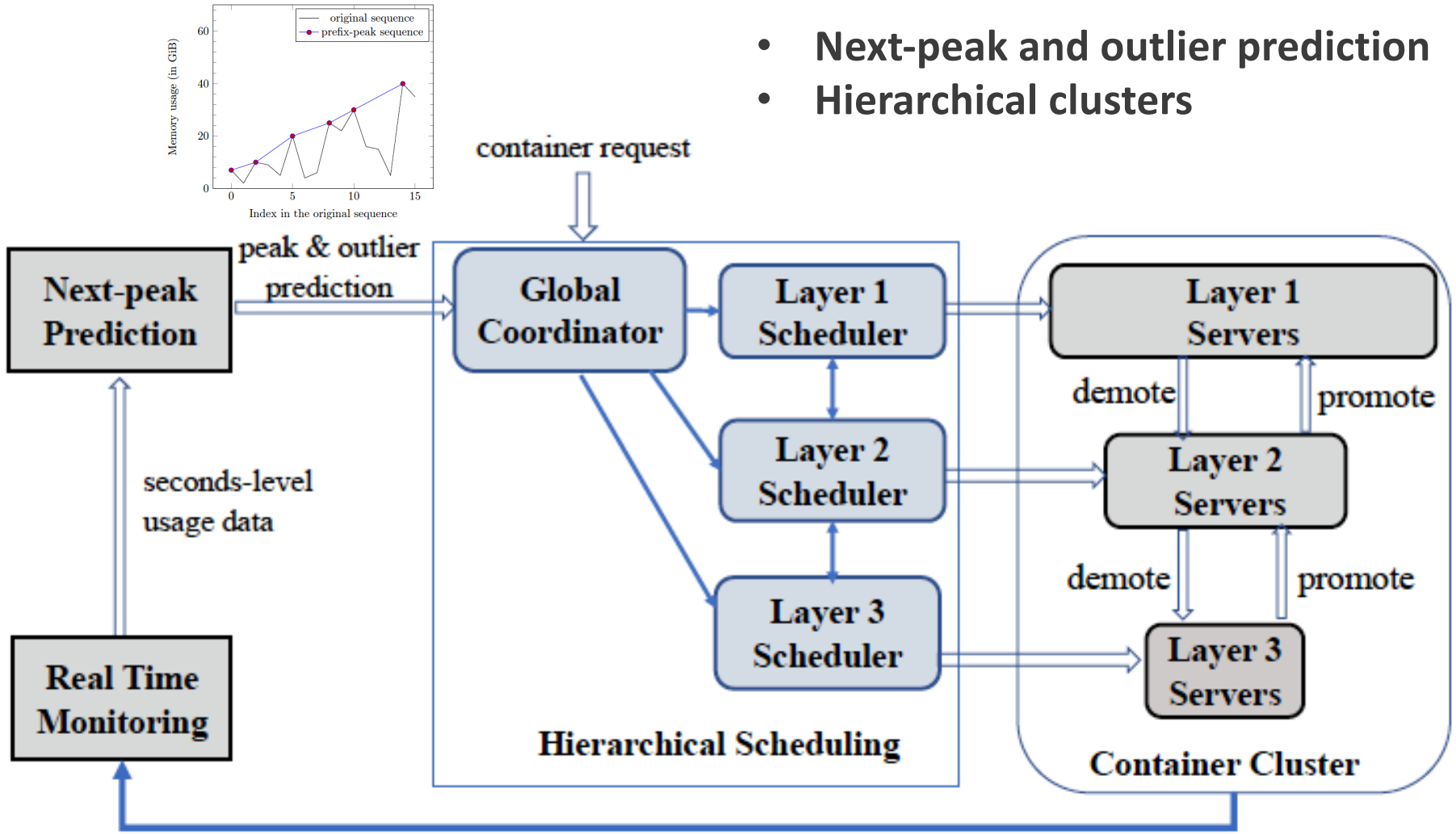
  New workload
  - Directly use memory limit specified in the submission.
  - Classify the workload based on its meta-data and use the statistic measures of the group.

  Existing Workload
  - Predict the next peak demand
- **Adjust or reschedule the workload if needed**



Memory Scheduling for Online Services

- Next-peak and outlier prediction
- Hierarchical clusters



Memory Utilization of Online Services

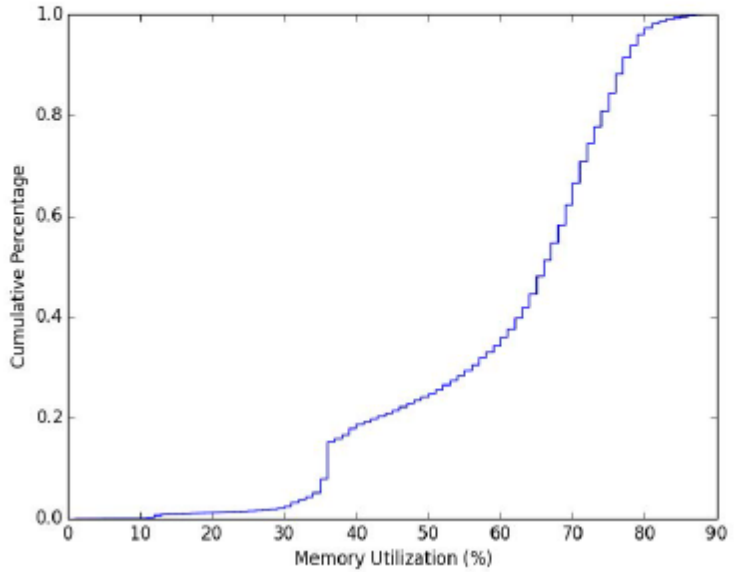
80% utilization without OOM

memory utilization	*#OOM*	$\frac{\#sched\_fail}{\#workload}$	$\frac{\#migration}{\#promotion}$
80.45%	0	3.19%	2.26%
85.02%	3	5.32%	4.24%
89.48%	226	5.77%	4.76%

Real-trace Driven Simulation

61% utilization without OOM

#Servers	Memory Utilization				
	Avg.	Min	Median	99%ile	Max
8,126	61.0%	1.0%	66.0%	83.0%	90.0%



Production Systems

京东

Host Selection Optimization

Host Priority	Vanilla Kubernetes	JDOS
Multi-resource balance	Variance between remaining CPU, memory and disk	Similarity between the host remaining resources and the request
Resource availability	Equal weights for CPU and Memory	Workload-aware weights Resource usage dynamics
Pod affinity	N/A	Pod demand correlation-aware

京东

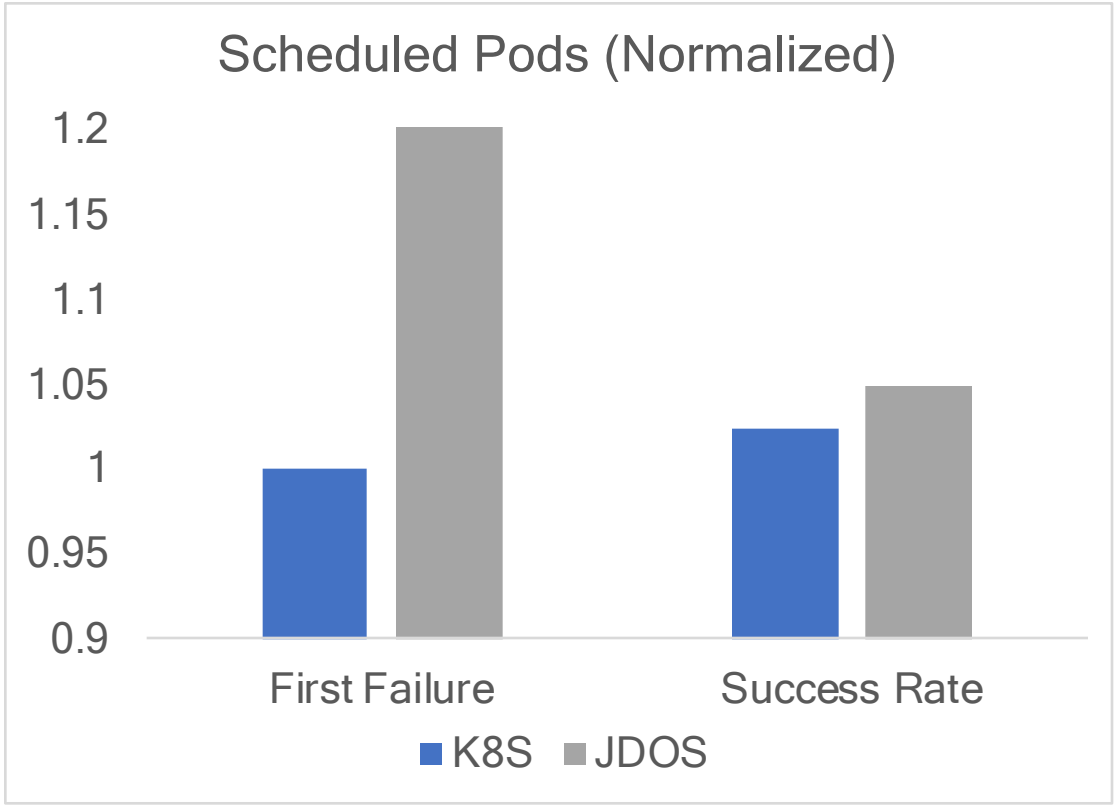
Comparison with K8S

Setup

- 4,857 servers (8 configurations)
- 25,000 pods (120 configurations)

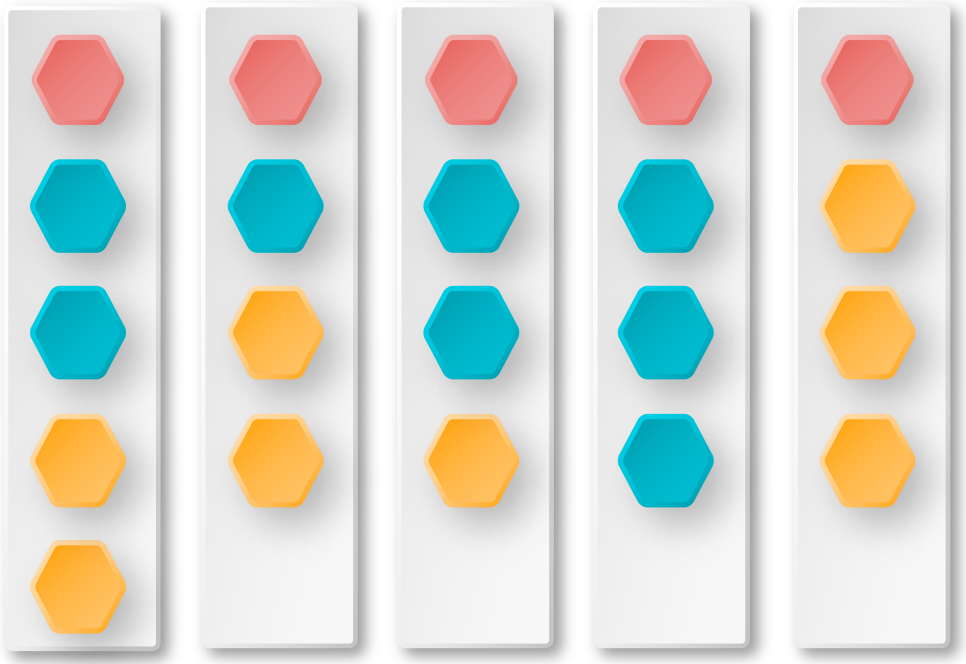
Results

- Schedule 5-20% more pods than K8S.
- Comparable scheduling performance with K8S.



京东

Mixed Workload Placement



HPLTS: High Priority Long Term Service

LPLTS: Low Priority Long Term Service

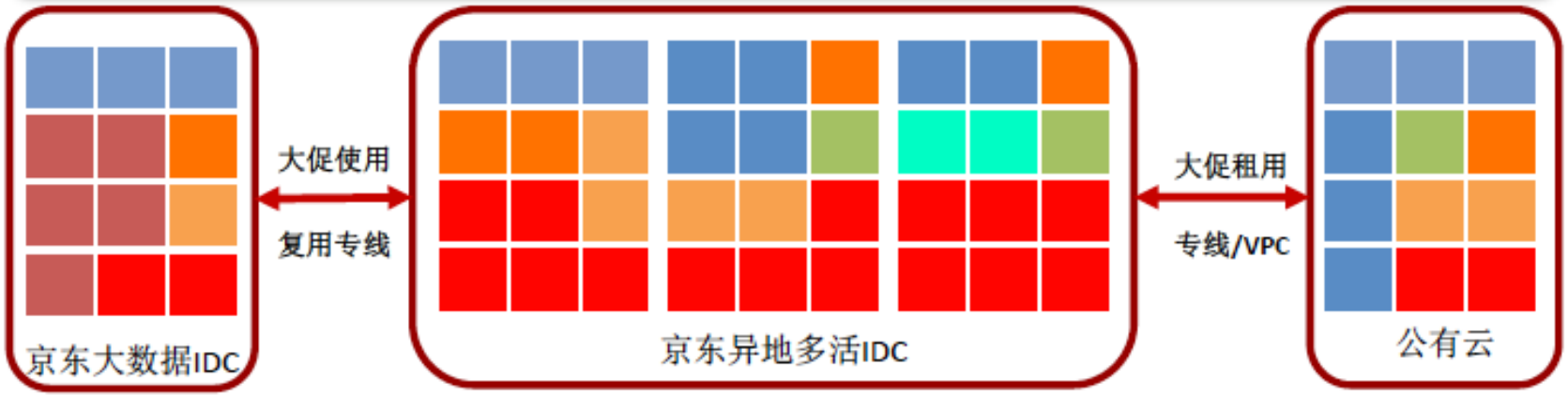
BJ: Batch Job

1. Even distribution of HPLTS
2. A Mix of LTS and BJ
3. Anti-affinity placement

京东

Hybrid Resource Pools

Resource Scaling across hybrid resource pools



Co-location of online services and batch jobs is not a silver bullet

- **Performance and fault isolation concerns.**
- **Memory safety margin:** batch jobs are not memory intensive.
- **Cost issue**: batch jobs can run in "cheap" and "remote" data centers, but internet services need to be located close to users with low latency.

京东

Singles Day Shopping Festival (11/11/2018)

17%

3-5X

hundreds of millions

- $24.7 billion in transaction volume
- Scheduled 460,000 containers (Pod) and 3,000,000 CPU cores

京东

Conclusions

- JD is one of the earliest adopters of Kubernetes and container technologies.
- We have customized Kubernetes and built a modern system on top of it.
- This entire ecosystem of Kubernetes plus our own optimizations dramatically reduce complexity of operations and make our systems stable and scalable.
- By integrating and optimizing the complete software stack on Kubernetes platform, especially through advanced scheduling, we significantly improve resource utilization and reduce costs.

京东

Acknowledgements

Thanks a lot for the following colleagues!

Jiangang Fan, Min Li, Yongcheng Bao, Xiaoping Liang, Huaxia Wang, Zhengxuan Wu, Shugang Chen, Wanli Yang, Xinkun Xu, Feng Wang, Huasong Shan, Liying Zhang, Shanshan Sun

京东

Thanks for your time!

Contact:

Yuan Chen (Email: yuan.chen@jd.com, WeChat: yuan_gt)
Haifeng Liu (Email: bjliuhaifeng@jd.com)

京东

Right Size, Right Location and Right Time

R^3: Right Size, Right Location, Right Time

- Pod sizing (**Right Size)**
- Host Selection (**Right Location)**
- Pod migration and rescheduling (**Right Time & Right Location**)
- Co-scheduling of mixed workloads across hybrid resource pool (**Right Time, Right Location, Right Size**)

京东

Host Selection: Multi-Resource Balancing

K8S: BalancedResourceAllocation = 10 – abs(cpuRemaingFraction-memoryRemainingFraction)*10

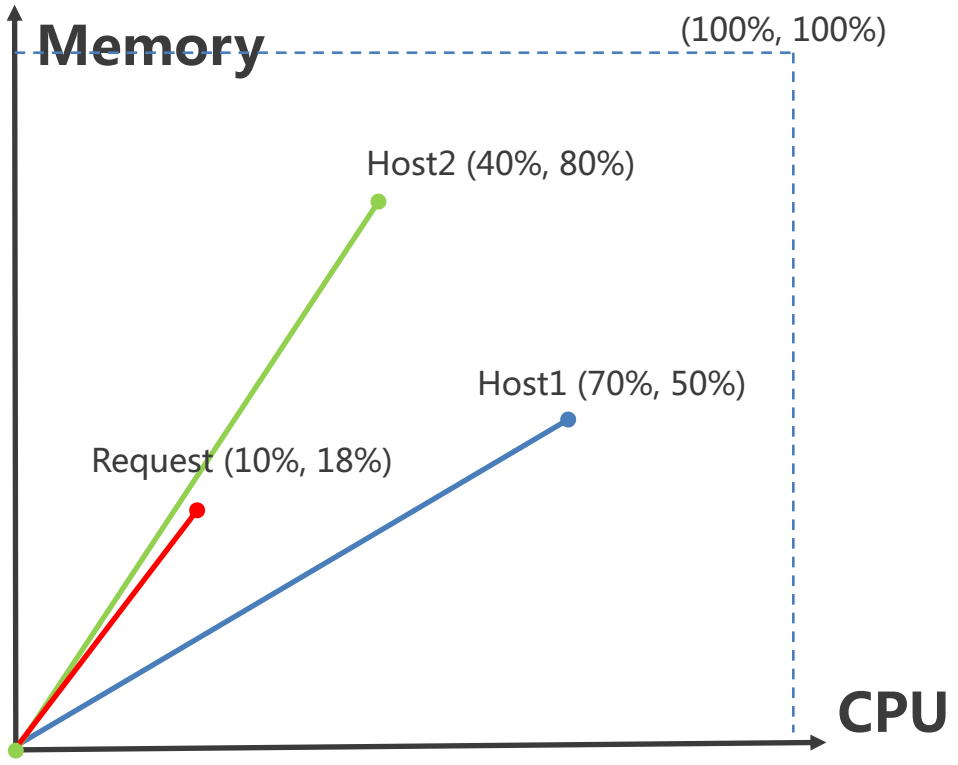
Optimization：similarity between the request resource and available resources

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Host available resources： (U_{CPU}, U_{MEM})

Pod resource request： (R_{CPU}, R_{MEM})

Metric: $$\frac{(R_{CPU}U_{CPU} + R_{MEM}U_{MEM})}{\sqrt{R_{CPU}^2 + R_{MEM}^2}\sqrt{U_{CPU}^2 + U_{MEM}^2}}$$



Host Selection: Resource Availability

K8S :

LeastRequestedPriority = cpu((capacity – sum(requested)) 10 / capacity) + memory((capacity – sum(requested)) 10 / capacity) / 2

Optimization: dynamic variable weights

$$w_{cpu} * \text{cpuAvailFraction} + w_{mem} * \text{memAvailFraction}$$

Algorithm 1:

Weighted sum of CPU : $\text{cpuFraction} = \sum_i f_i * \text{cpuFraction}_i$

Weight sum of memory : $\text{memFraction} = \sum_i f_i * \text{memFraction}_i$

$$w_{cpu} = \frac{\text{cpu}Fraction}{cpuFraction + memFraction}$$

$$w_{mem} = \frac{memFraction}{cpuFraction + memFraction}$$

Algorithm 2: Hosts with stable resource usage are more usable.

MAD- Median Absolute Deviation

$$w_{cpu} = 1 - \text{median}(|\text{cpuUtil}_i - median(\text{cpuUtil}_i)|)$$

$$w_{mem} = 1 - \text{median}(|\text{memUtil}_i - median(\text{memUtil}_i)|)$$

京东

Host Selection: Correlation-awareness

X : existing pods resource usage

Y : new pod resource usage

Anti-affinity: If Y can be predicted by Y, X and Y are highly correlated. We should avoid placing them on the same host.

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,k} & \cdots & x_{1,K} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \cdots & x_{n,k} & \cdots & x_{n,K} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,k} & \cdots & x_{N,K} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{bmatrix}$$

Multi-regression analysis $\mathbf{b} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \quad \hat{\mathbf{y}} = \mathbf{X}\mathbf{b}$

Correlation $$\frac{\sum_{i=1}^{n}(\mathbf{y}_i - m_Y)^2(\hat{\mathbf{y}}_i - m_{\hat{Y}})^2}{\sum_{i=1}^{n}(\mathbf{y}_i - m_Y)^2 \sum_{i=1}^{n}(\hat{\mathbf{y}}_i - m_{\hat{Y}})^2}$$