Cloud and Edge Computing for Mobile Intelligence

Jun ZHANG

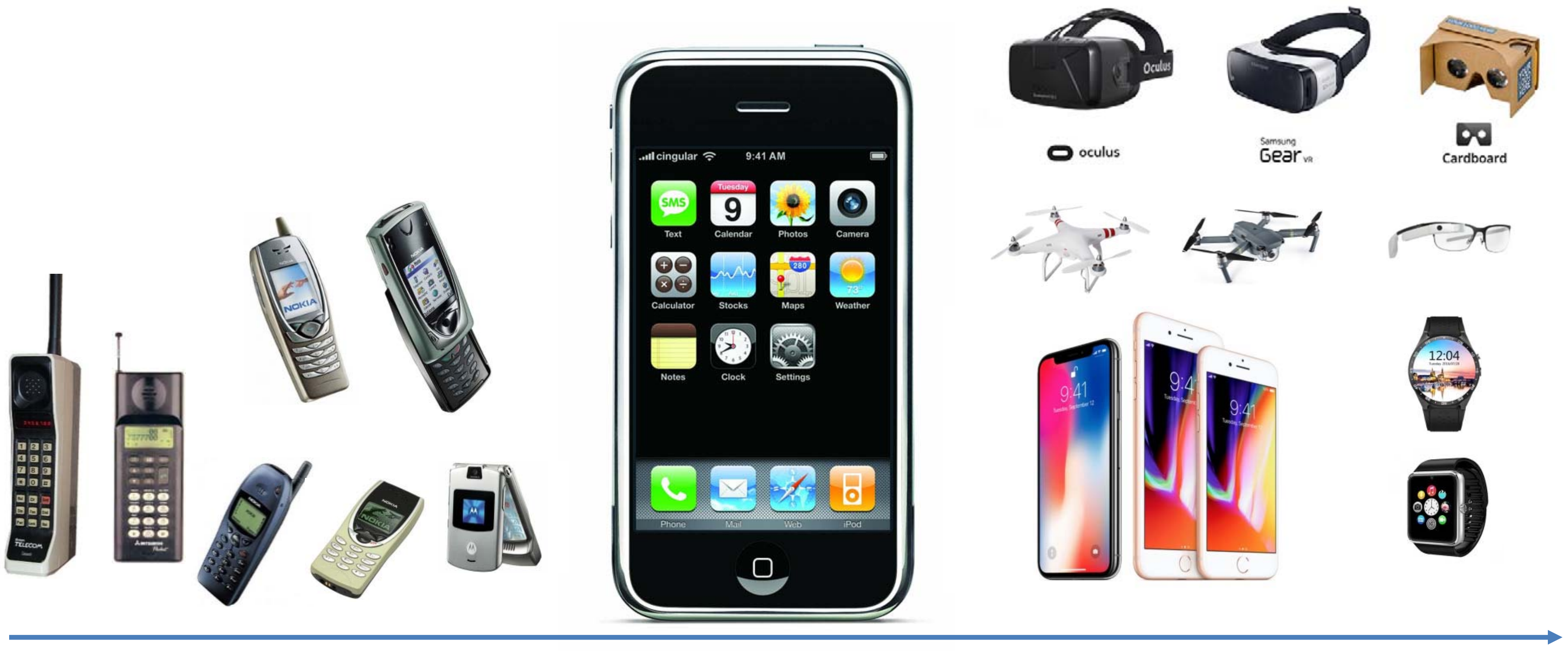
Jan 4, 2018

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Outline

- Introduction
- In-Memory Data Analytics Clusters
- Mobile Edge Computing
- Takeaways

The Tipping Point

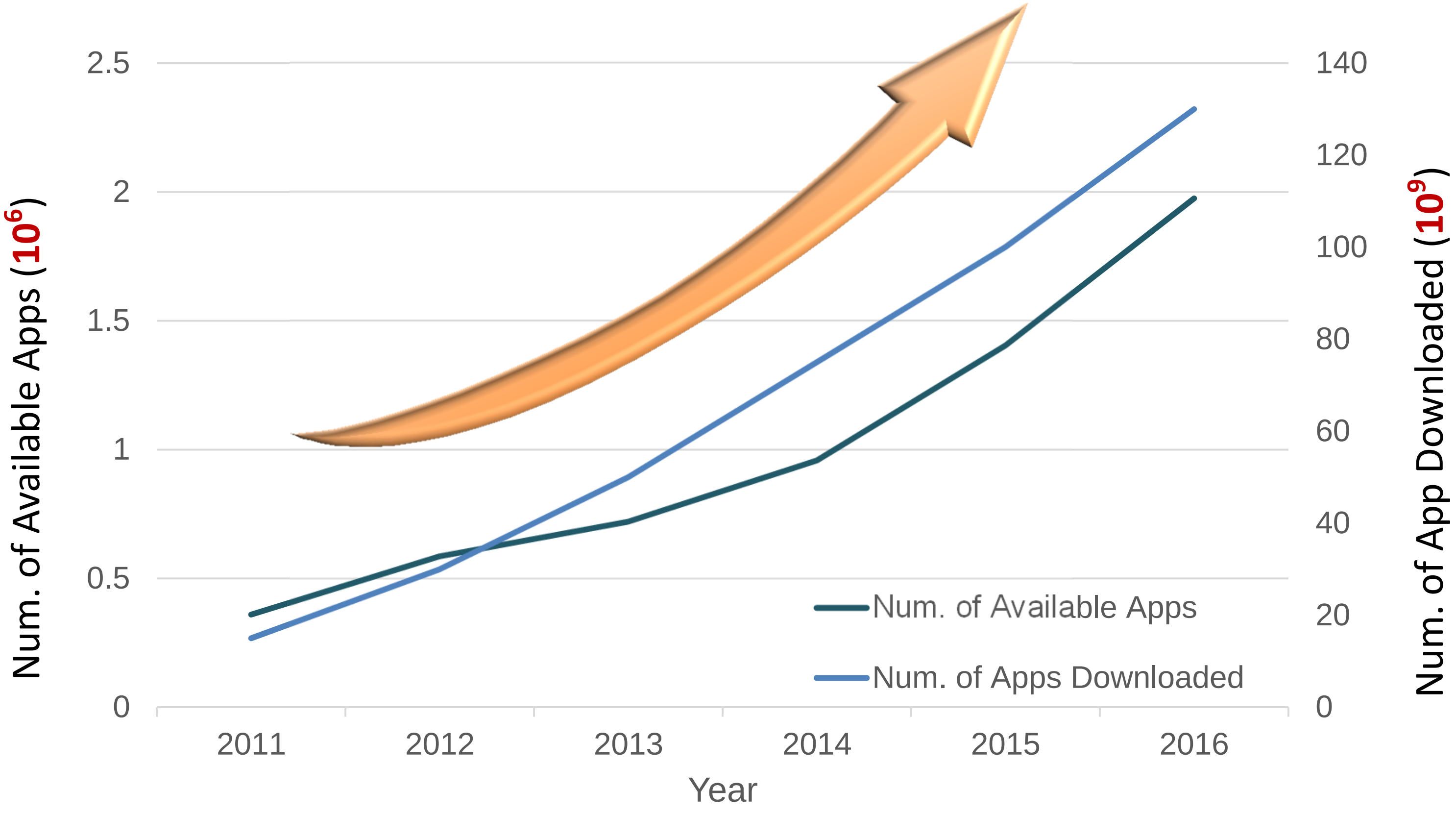


Before

June 29, 2007

After

Growth of Mobile Application Markets



[Source: Statista]

Download on the
App Store

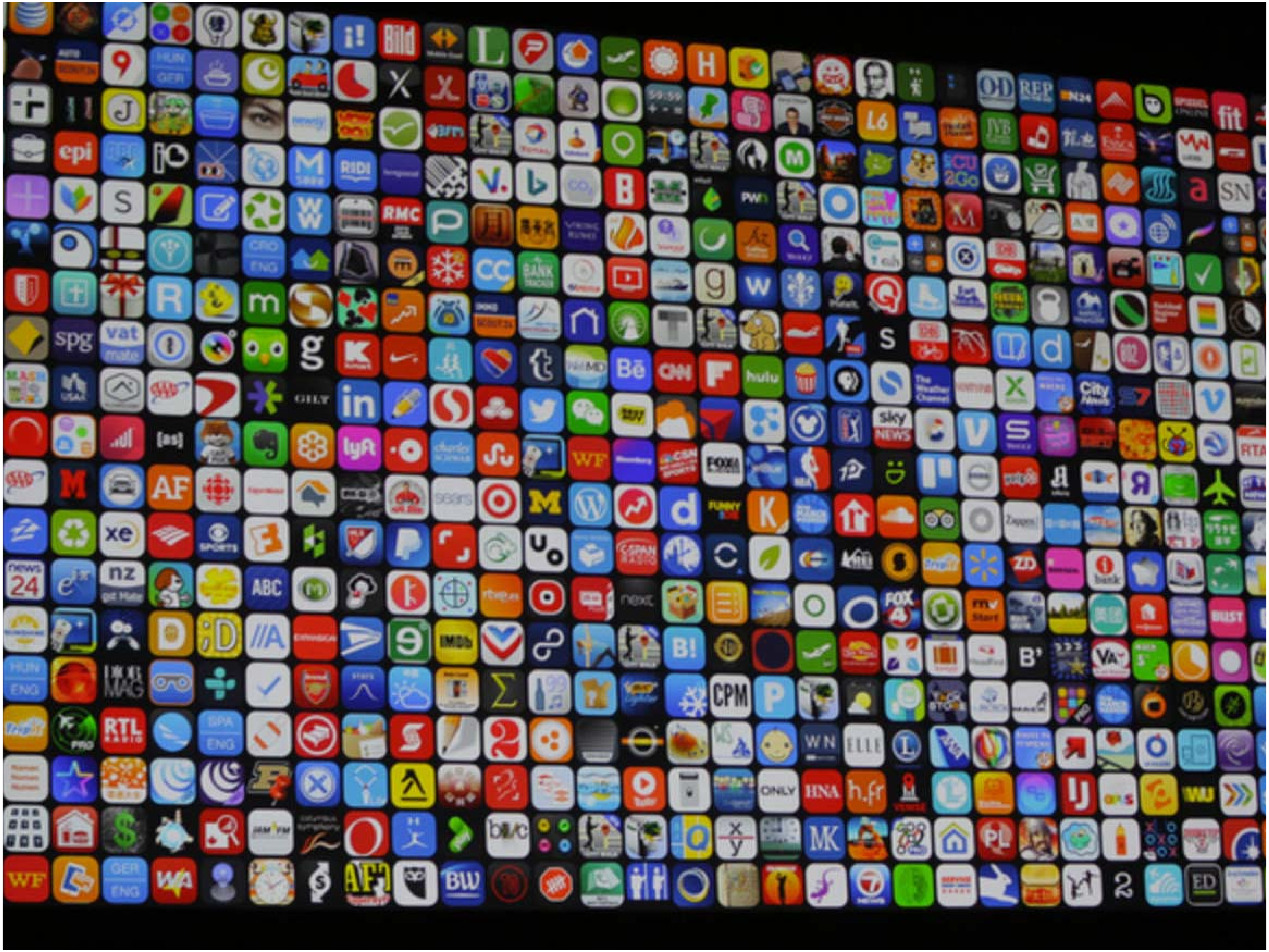
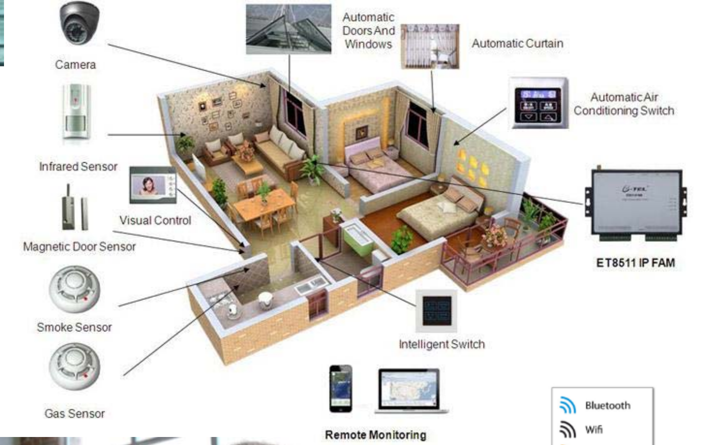
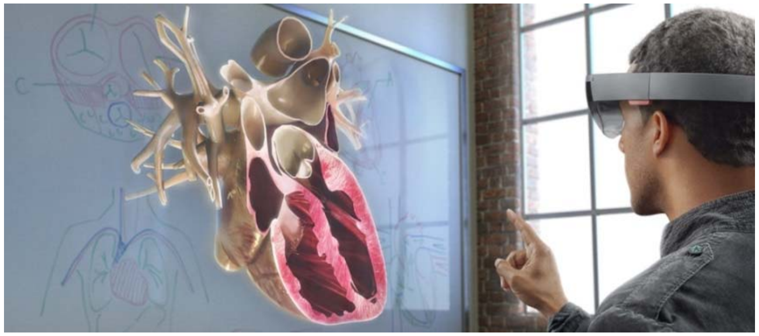


Bild
L
H
OD
REP
fit
J
L6
epi
M
RIDI
S
W
RMC
V.
B
Pwn
a
SN
CC
BANK
g
R
m
W
spg
vat mate
g
K
t
Bē
CNN
hulu
S
d
in
Sky NEWS
V
City
[as]
lyft
WF
FOX
M
AF
sears
M
d
K
Я
xe
CBS SPORTS
C-SPAN
news 24
nz
ABC
next
FOX 4
bank
D
;D
//A
IMDb
B!
B'
BUST
HUN ENG
DIE MAG
CPM
P
RTL RADIO
SPA ENG
2
W N
ELLE
L
ONLY
HNA
h.fr
WU
$
bivc
MK
PL
WF
GER ENG
WA
AFI
BW
2
ED

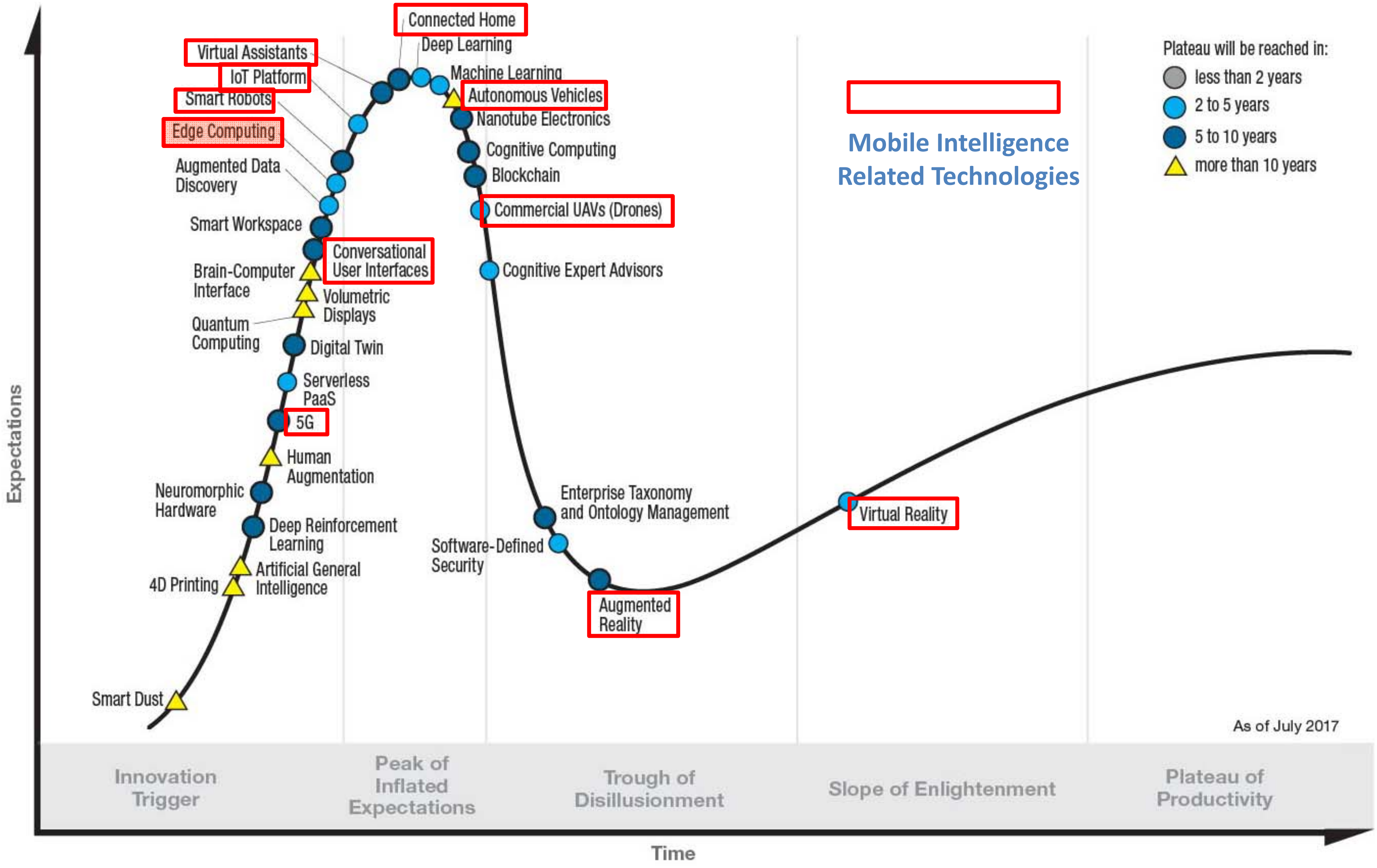
The Era of Mobile Intelligence





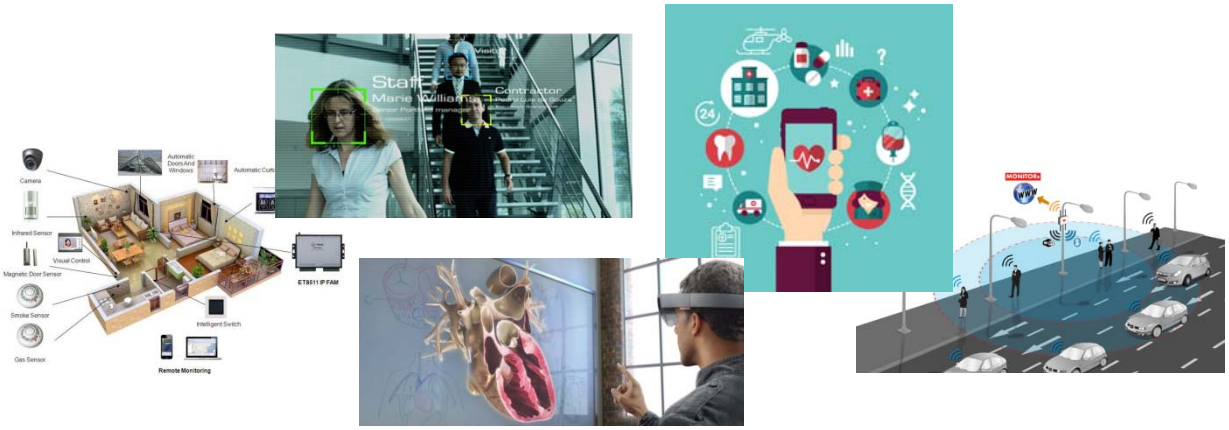
Bluetooth
Wifi
3G/GPRS
MONITOR
WWW

Gartner Hype Cycle for Emerging Technologies, 2017



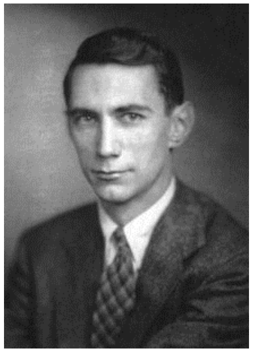
Mobile Intelligence Related Technologies

Mobile Intelligence

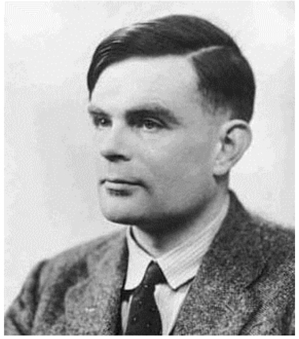


Communications

Computing



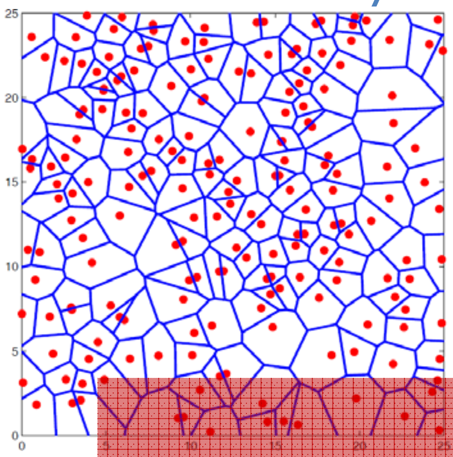
C. E. Shannon
(1916—2001)

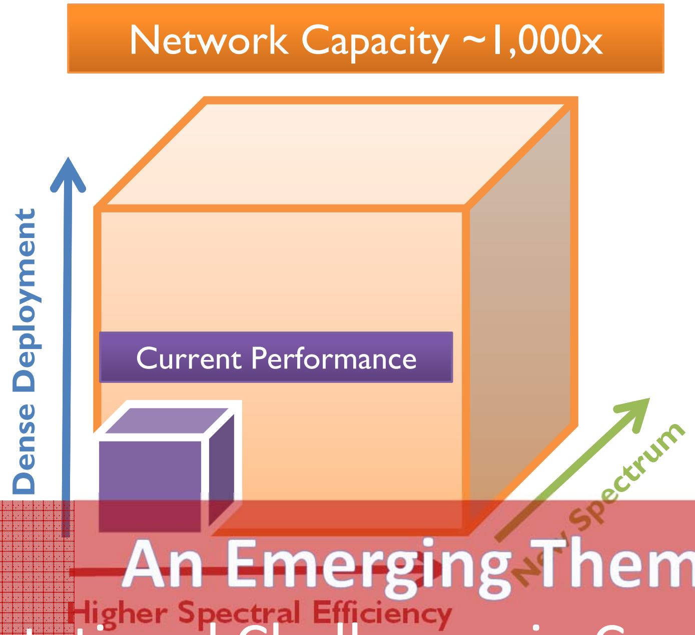


A. Turing
(1912—1954)

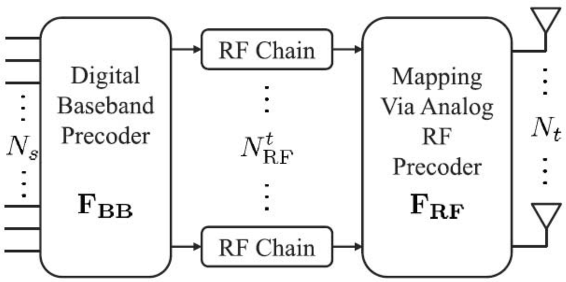
To Address the Communication Challenge – A 3D Picture

Network Analysis via Stochastic Geometry





Hybrid Precoding

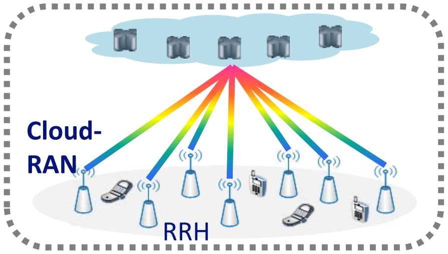


Digital Baseband Precoder
$\mathbf{F}_{\mathrm{BB}}$
RF Chain
RF Chain
N_{RF}^t
Mapping Via Analog RF Precoder
$\mathbf{F}_{\mathrm{RF}}$
N_s
N_t

An Emerging Theme

Computational Challenges in Communications

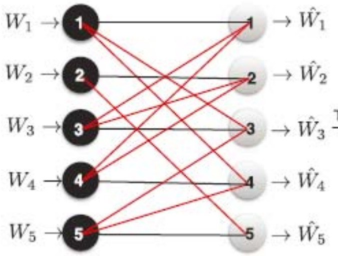
Sparse Beamforming for Network Adaptation



$$\mathbf{v} = [\underbrace{\mathbf{v}_{11}^T,\ldots,\mathbf{v}_{1K}^T}_{\tilde{\mathbf{v}}_1^T},\ldots,\underbrace{\mathbf{v}_{L1}^T,\ldots,\mathbf{v}_{LK}^T}_{\tilde{\mathbf{v}}_L^T}]^T$$

Beamforming coefficients forming a **group**

Low-Rank Optimization for Interference Management



Transmitters

$$\xrightarrow{\text{Transformation}} \mathbf{X} = \begin{bmatrix} 1 & \star & 0 & 0 & \star \\ \star & 1 & 0 & 0 & \star \\ 0 & \star & 1 & \star & 0 \\ 0 & \star & \star & 1 & 0 \\ \star & 0 & \star & \star & 1 \end{bmatrix}$$

Receivers

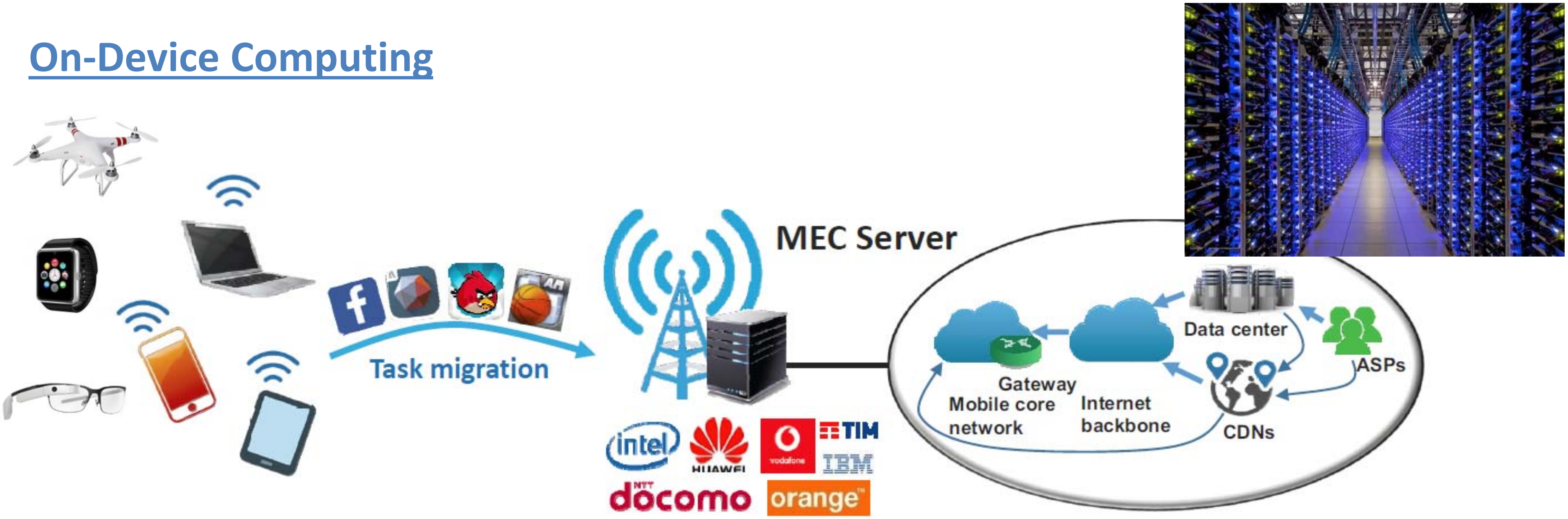
To Address the Computation Challenge
– A 3-Layer Picture

Cloud Computing

This Talk

Mobile Edge Computing

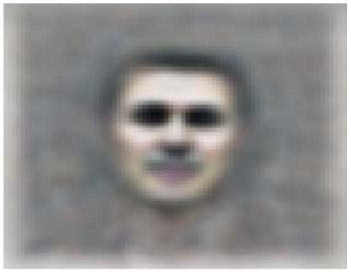
On-Device Computing



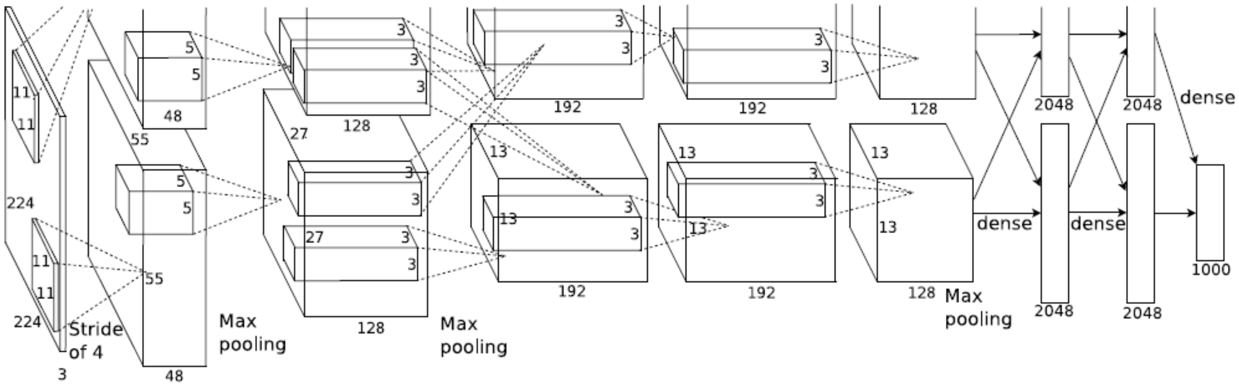
In-Memory Big Data Analytics Clusters

BIG DATA Challenge

- **Training**
  - **DistBelief** (Google) [1]
    - 1 billion connections
    - 1,000 machines for 3 days (16,000 cores)
    - 600 kWatts, $5,000,000



- **Inference** (BIG model size)
  - **AlexNet Caffemodel** > 200MB [2, 3]

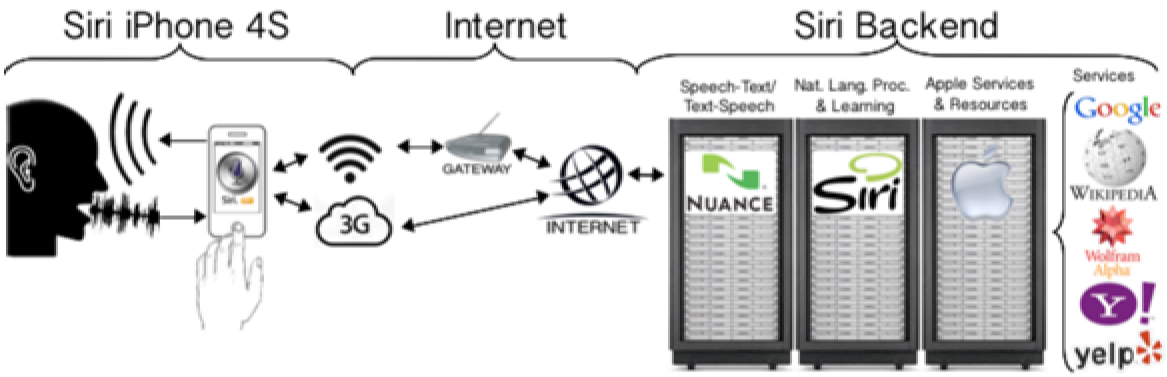


[1] Le, Q, Ranzato, M, el. al. Building high level features using large scale unsupervised learning. ICML 2012.
[2] Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet classification with deep convolutional neural networks. NIPS 2012.
[3] Caffe model zoo. URL http://caffe.berkeleyvision.org/model_zoo.

Go to the Cloud

- Mobile devices are limited in
  - **Processor speed**
  - **Memory size** ☹
  - **Disk capacity**
  - **Battery life**

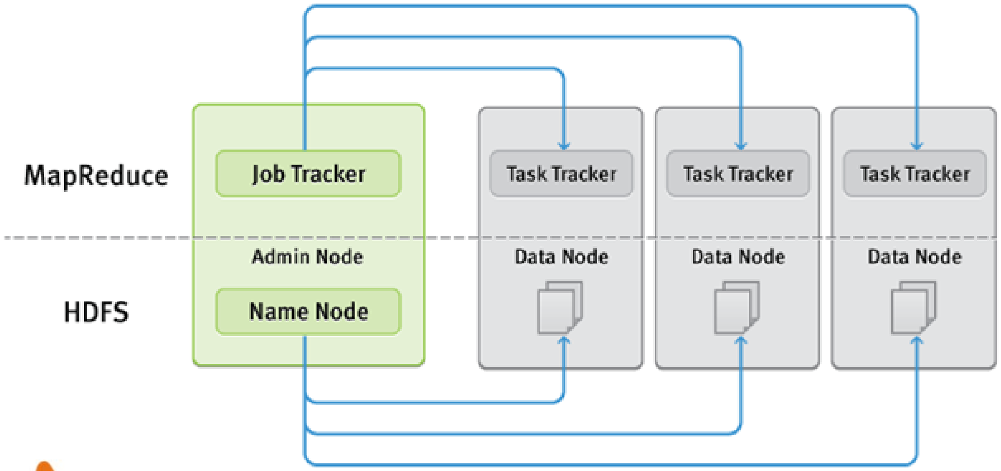
- Solution Example: Siri ☺



http://www.howtechnologywork.com/how-siri-works/

Big Data Analytics in the Cloud

- Cluster-Computing Frameworks
  - hadoop (December 2011) [4]



  - Spark (May 2014) [5]

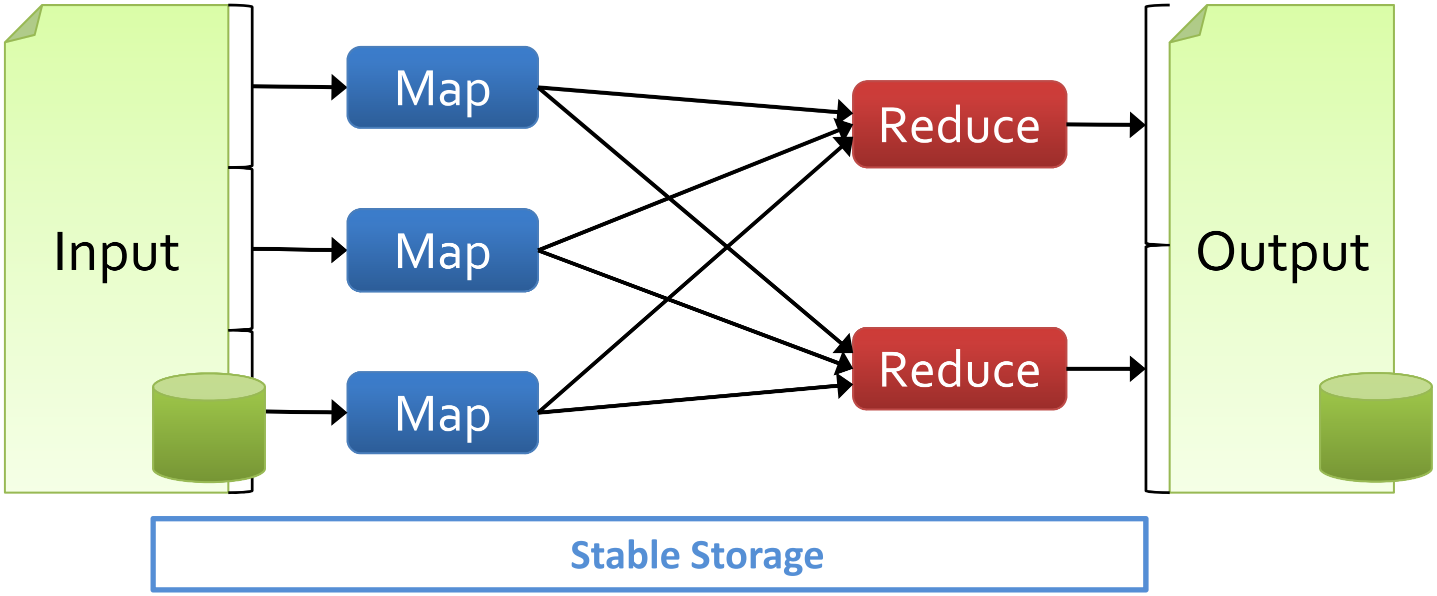
    In-memory cluster computing

[4] J. Dean and S. Ghemawat. “MapReduce: Simplified data processing on large clusters.” In Proc. The 6th Symposium on Operating Systems Design and Implementation (OSDI), pp.137–150, Dec. 2004.

[5] M. Zaharia, M. Chowdhury, et al. “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing.” In NSDI, 2012.

Inefficiency of MapReduce

- MapReduce
  - Write the program state to disk every iteration

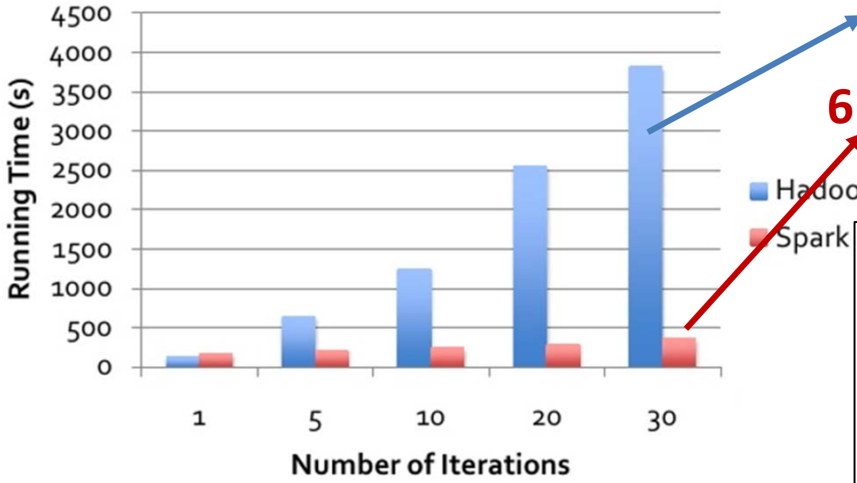


❑ **Inefficient** for

- Iterative algorithms (machine learning, graphs)
- Interactive data mining

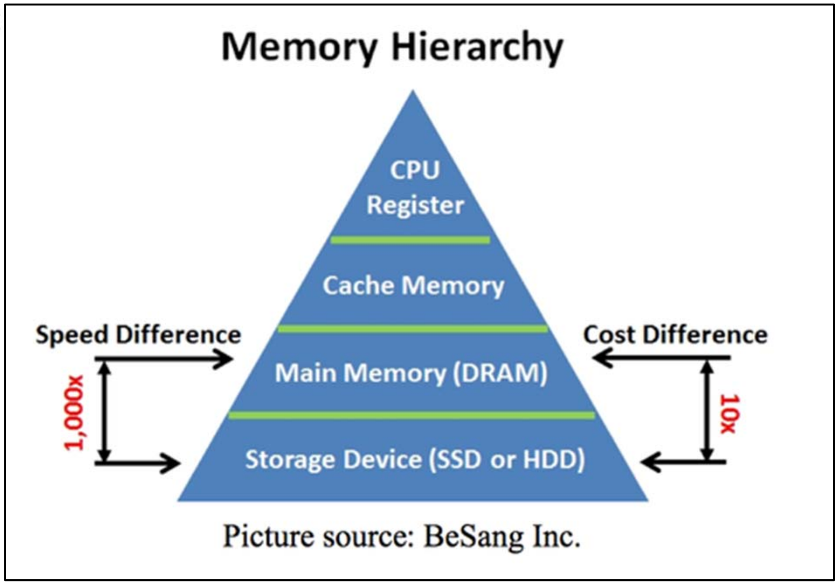
Memory Speeds up Computation

- By caching input data in memory, Spark reduces the runtime by **20 times**. [5]



127 sec/iteration

6 sec/iteration

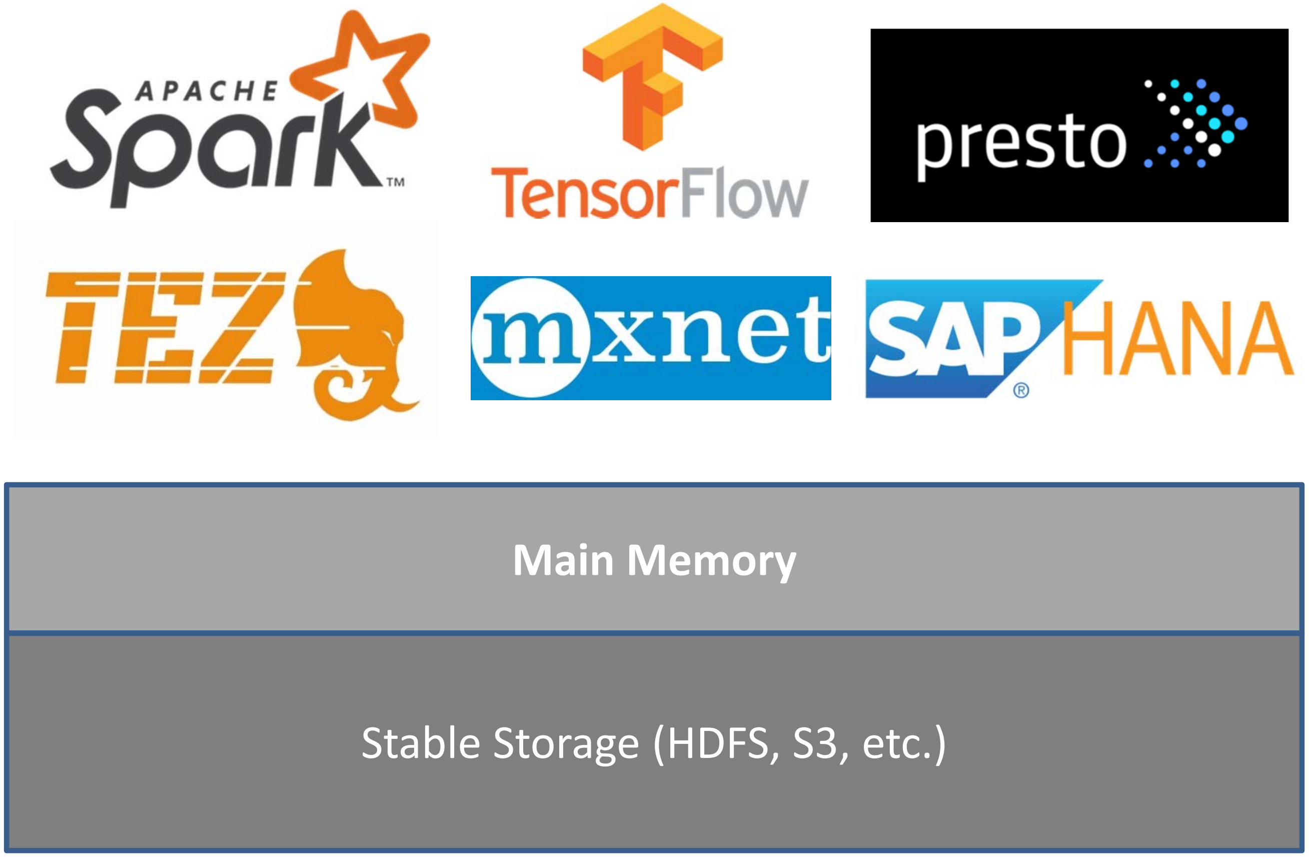


Picture source: BeSang Inc.

[5] M. Zaharia, M. Chowdhury, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." In NSDI, 2012

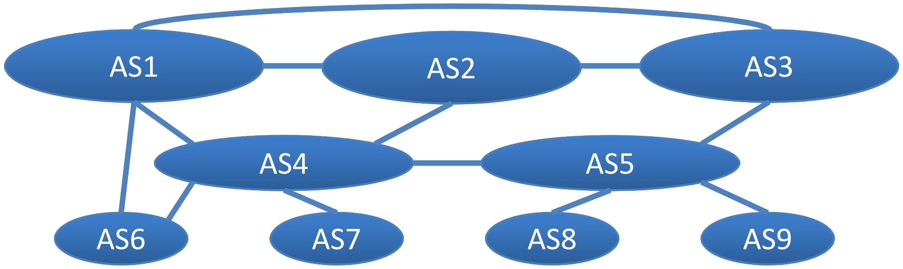
In-Memory Processing

- Data analytics clusters are shifting towards in-memory computations

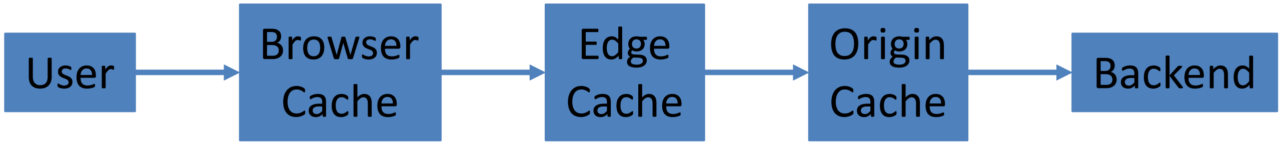


Cache Management

- Crucial for in-memory data analytics systems.
- Well studied in many systems
  - **CDN** (Akamai)



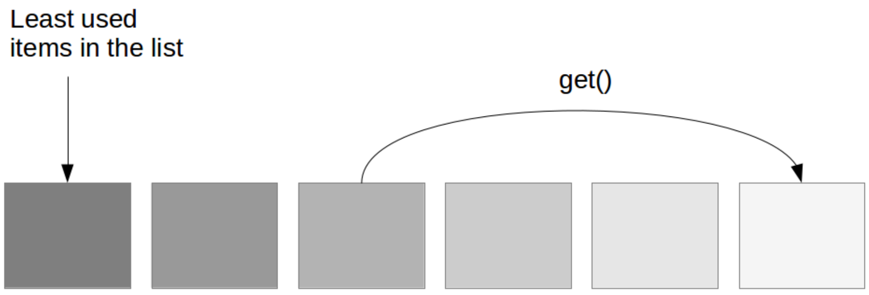
  - **Facebook**



- **Objective**: optimize the ***cache hit ratio***
  - Maximize the chance of in-memory data access.

Existing Solutions

- Least Recently Used (LRU) policy [R. L. Mattson, 1970]
  - Evicts the data block that has not been used for the longest period.
  - Widely employed in prevalent systems, e.g., Spark, Tez and Alluxio.



Calling get() for an item, moves it to the top of the cache

- Least Frequently Used (LFU) policy [M. Stonebraker, 1971]
  - Evicts the data block that has been used the least times.

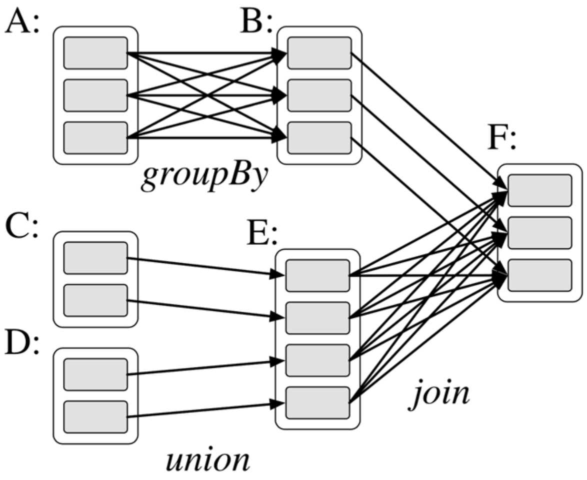
- Summary: "**guessing**" the future data access patterns based on historical information (access recency or frequency).

What’s New for Data Analytics Clusters?

- **Question 1**: Is the future data access completely random and unpredictable?
- **No!**

Data Dependency Reveals Access Patterns

- Application Semantics
  - Data dependency structured as a Directed Acyclic Graph (DAG)



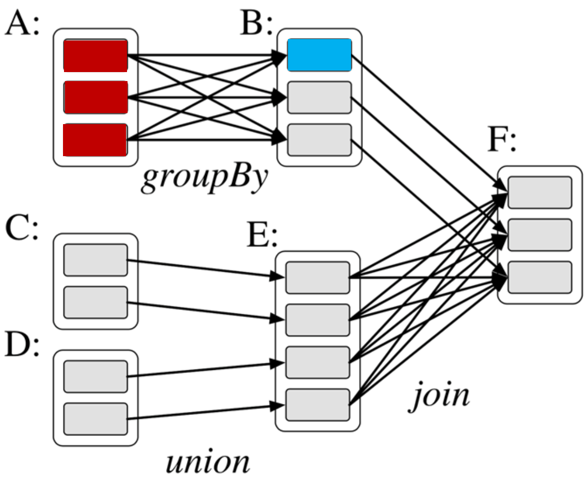
  - Available to the cluster scheduler **before** the job starts
- ***Data access*** follows the dependency DAG.
  - The future is not totally unpredictable.

What’s New for Data Analytics Clusters?

- **Question 2**: Is cache hit ratio still a good metric to evaluate the cache performance?
- **No**

Data Dependency Reveals All-or-Nothing Property

- ***All-or-Nothing***: a computing task can only be sped up when its dependent data blocks are ***all*** cached in memory.
  - E.g. To compute a block in B, all blocks of A are required. Cache hits of only part of the three blocks makes no difference.



  - Cache hit ratio is not appropriate metric for cache performance.

Inefficiency of Existing Cache Polices

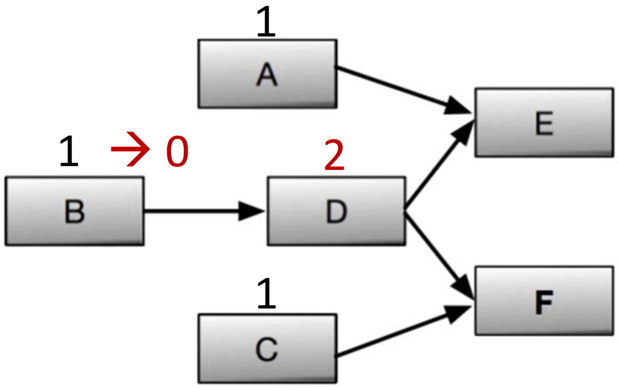
- Oblivious to the data access pattern:
  - Inactive data (no future access) cannot be evicted timely.
  - In our measurement studies, inactive data accounts for **>77%** of the cache space for **>50%** of time.

- Oblivious to the all-or-nothing property:
  - Achieving a high cache hit ratio does not necessarily speed up the computation.

- **Challenge**: How to exploit the data dependency information (DAGs) to clear **the inactive data** efficiently and factor in **the all-or-nothing property**?

LRC: Dependency-Aware Cache Management

- **Reference count**: defined for each data block as the number of downstream tasks depending on it.
  - Dynamically changing over time:



- Least Reference Count (**LRC**) policy [6]: when the cache is full, always evict the data with the least reference count.
  - Inactive data (w/ zero reference count) is evicted first, e.g., block B.

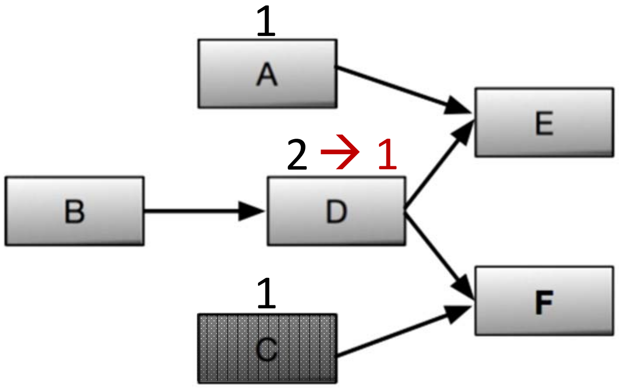
[6] Y. Yu, W. Wang, **J. Zhang**, and K. B. Letaief, "LRC: Dependency-aware cache management in data analytics clusters," in Proc. IEEE INFOCOM 2017, Atlanta, GA, May 2017.

Effective Cache Hit Ratio

- Factor in the all-or-nothing property?
- *Effective cache hit ratio:* A cache hit is effective when it speeds up the computation, i.e., when all the depended blocks are cached.

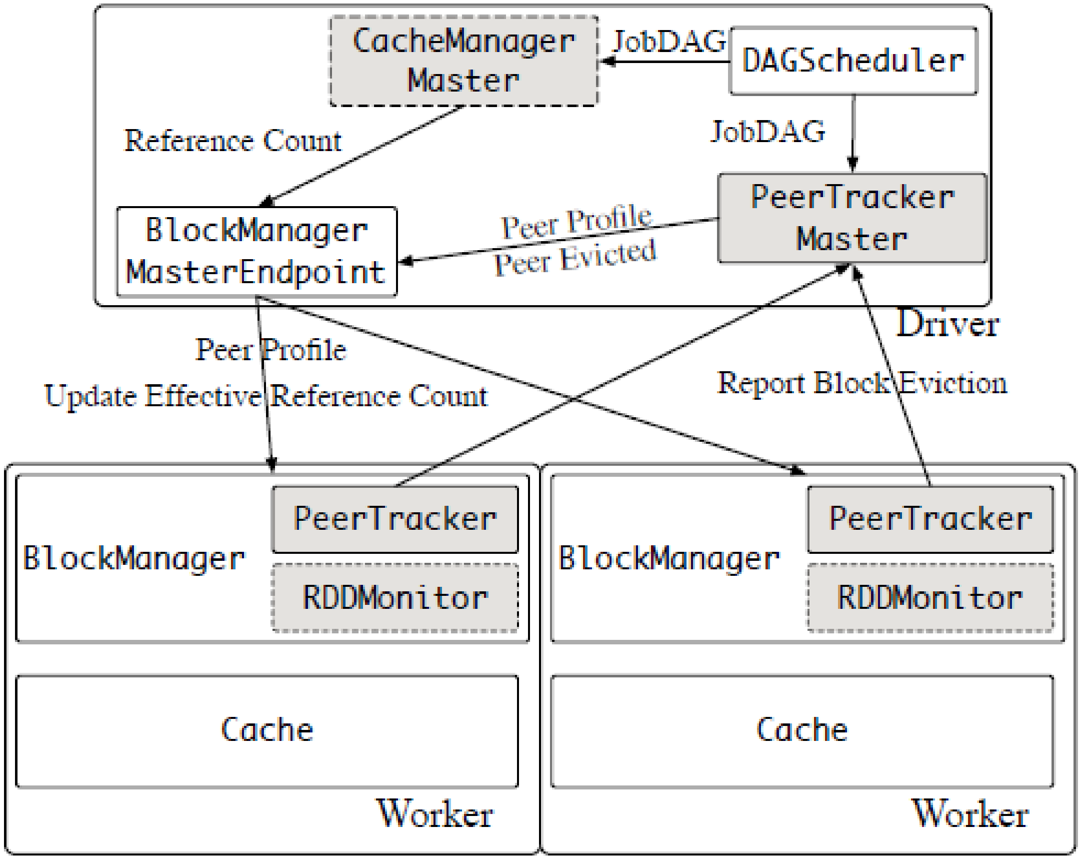
Tailor LRC to Optimize Effective Cache Hit Ratio

- A reference to a data block is only “counted” when it effectively speeds up the computation [7]
  - E.g., the reference to block D for computation of block F is not counted if block C is evicted from the cache.



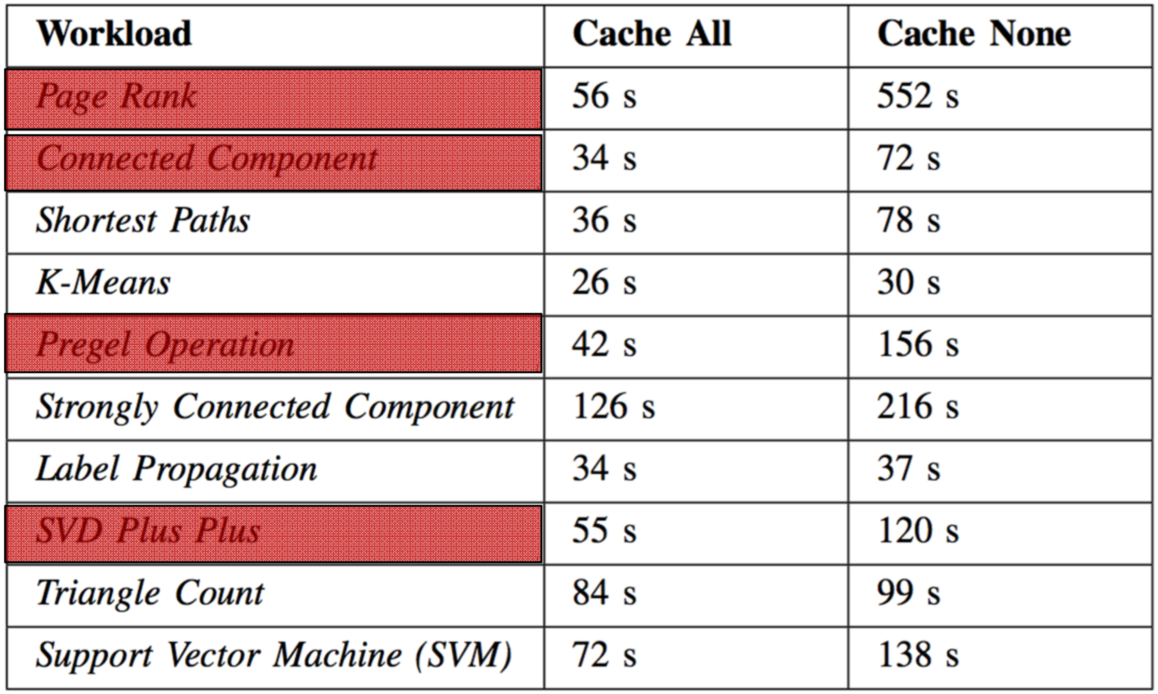
[7] Y. Yu, W. Wang, **J. Zhang**, and K. B. Letaief, "LERC: Coordinated cache management for data-parallel systems," accepted by IEEE Globecom, Singapore, Dec. 2017.

Spark Implementation



Evaluations: Workload Characterization

- **Cluster setup**: 20-node Amazon EC2 cluster.
- **Instance type**: m4.large. Dual-core 2.4 GHz Intel Xeon® E5-2676 v3 (Haswell) processors and 8 GB memory.
- **Workloads**: Typical applications from SparkBench [8].

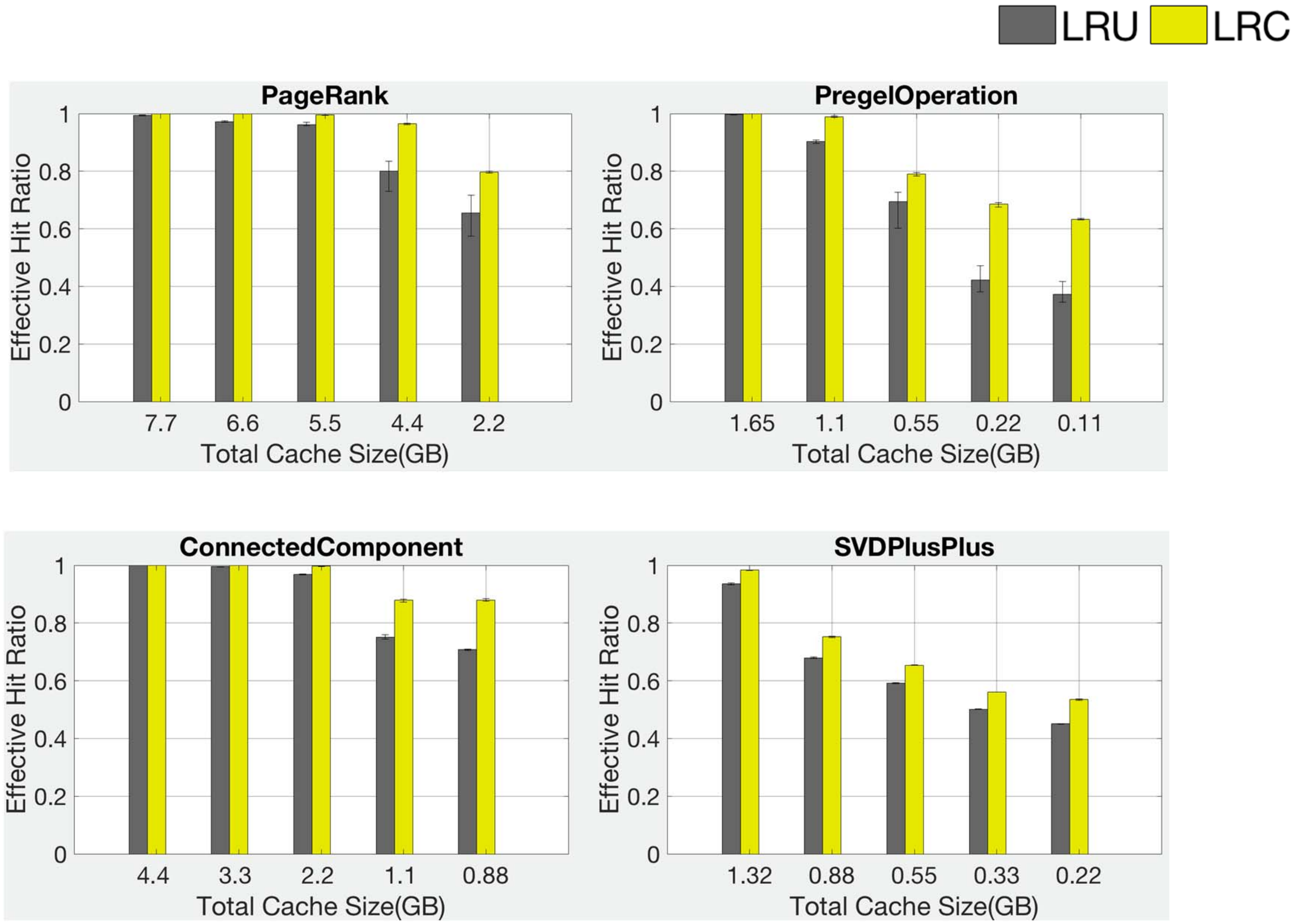


Workload	Cache All	Cache None
Page Rank	56 s	552 s
Connected Component	34 s	72 s
Shortest Paths	36 s	78 s
K-Means	26 s	30 s
Pregel Operation	42 s	156 s
Strongly Connected Component	126 s	216 s
Label Propagation	34 s	37 s
SVD Plus Plus	55 s	120 s
Triangle Count	84 s	99 s
Support Vector Machine (SVM)	72 s	138 s

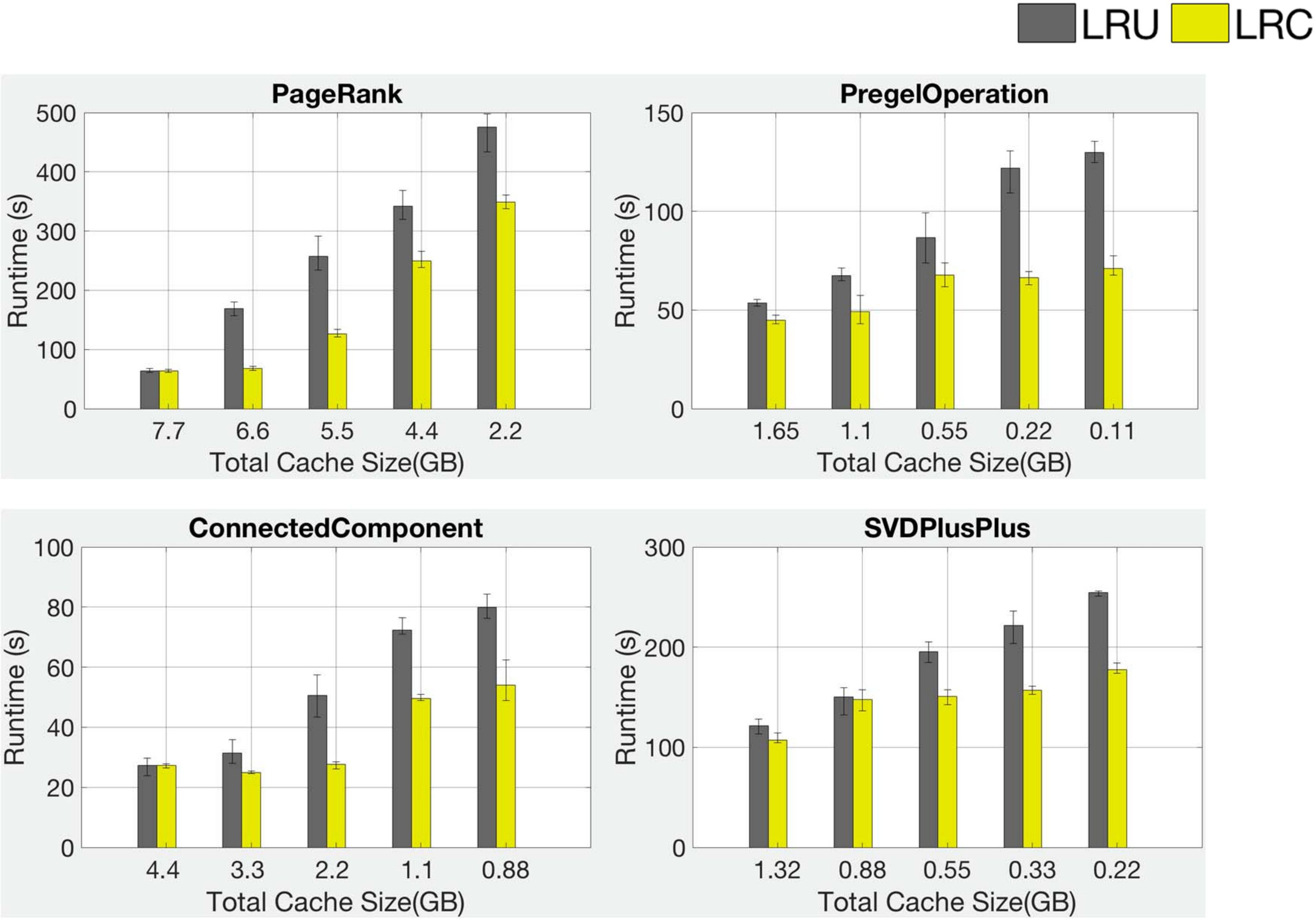
Not all applications benefit from the improvement of cache management.

[8] M. Li, J. Tan, Y. Wang, L. Zhang, and V. Salapura, “Sparkbench: a comprehensive benchmarking suite for in memory data analytic platform spark,” in Proc. 12th ACM International Conf. on Comput. Frontiers, 2015.

Evaluations: Effective Cache Hit Ratio

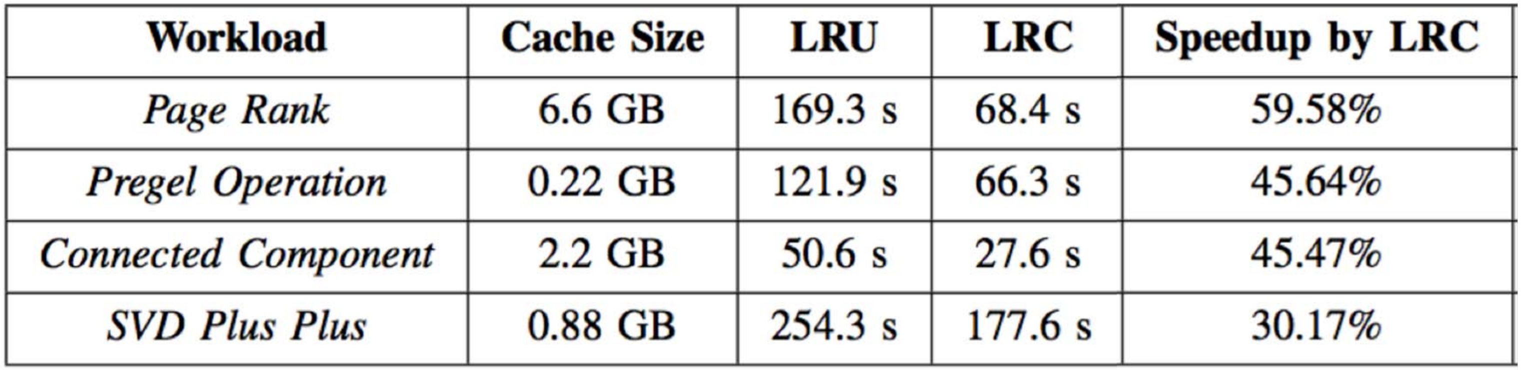


Evaluations: Application Runtime



Evaluations: Summary

- LRC speeds up typical workloads by up to 60%.



Workload	Cache Size	LRU	LRC	Speedup by LRC
Page Rank	6.6 GB	169.3 s	68.4 s	59.58%
Pregel Operation	0.22 GB	121.9 s	66.3 s	45.64%
Connected Component	2.2 GB	50.6 s	27.6 s	45.47%
SVD Plus Plus	0.88 GB	254.3 s	177.6 s	30.17%

Conclusions

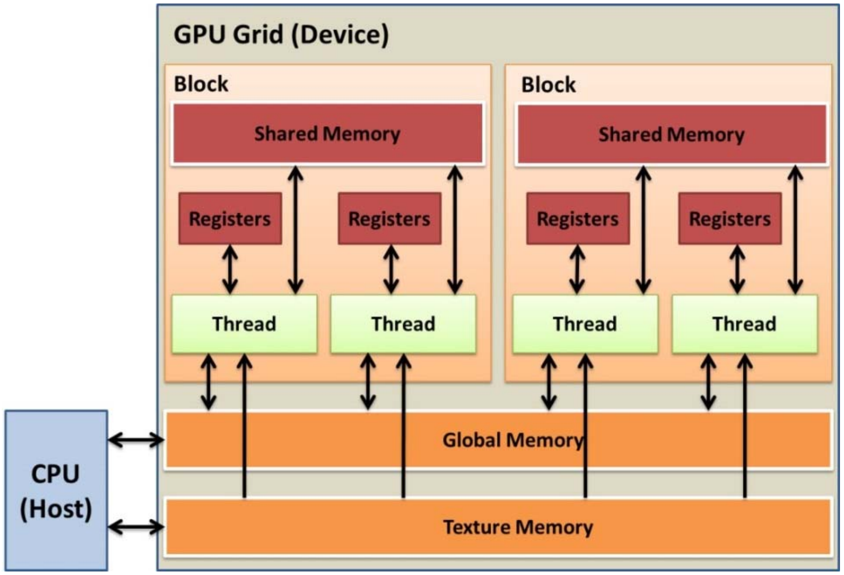
- It is effective to leverage the dependency DAGs to optimize the cache management
- Effective cache hit ratio is a better cache performance metric
  - To account for the all-or-nothing property
- **LRC** – a dependency-aware cache management policy
  - Optimizes the effective cache hit ratio
  - Speeds up typical workloads by up to 60%

Extension: Cache in Distributed Machine Learning Platforms

- Deep Learning Platforms

theano TensorFlow torch Caffe mxnet

- GPU



Mobile Edge Computing

NEED FOR SPEED

- VR/AR
  - Latency < 20 ms
  - Avoid cybersickness

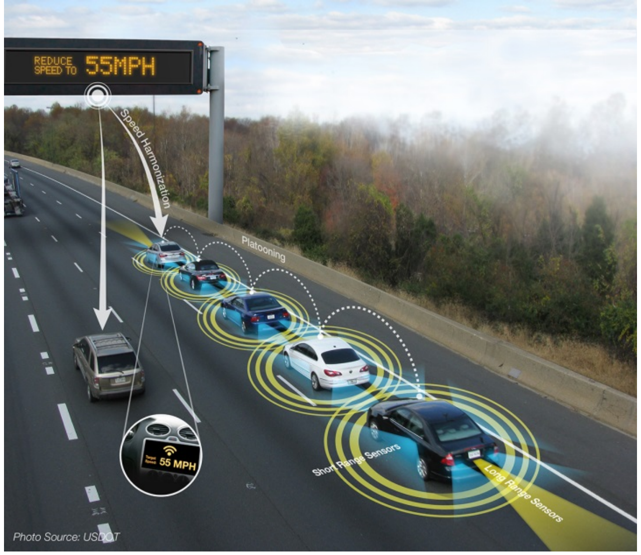
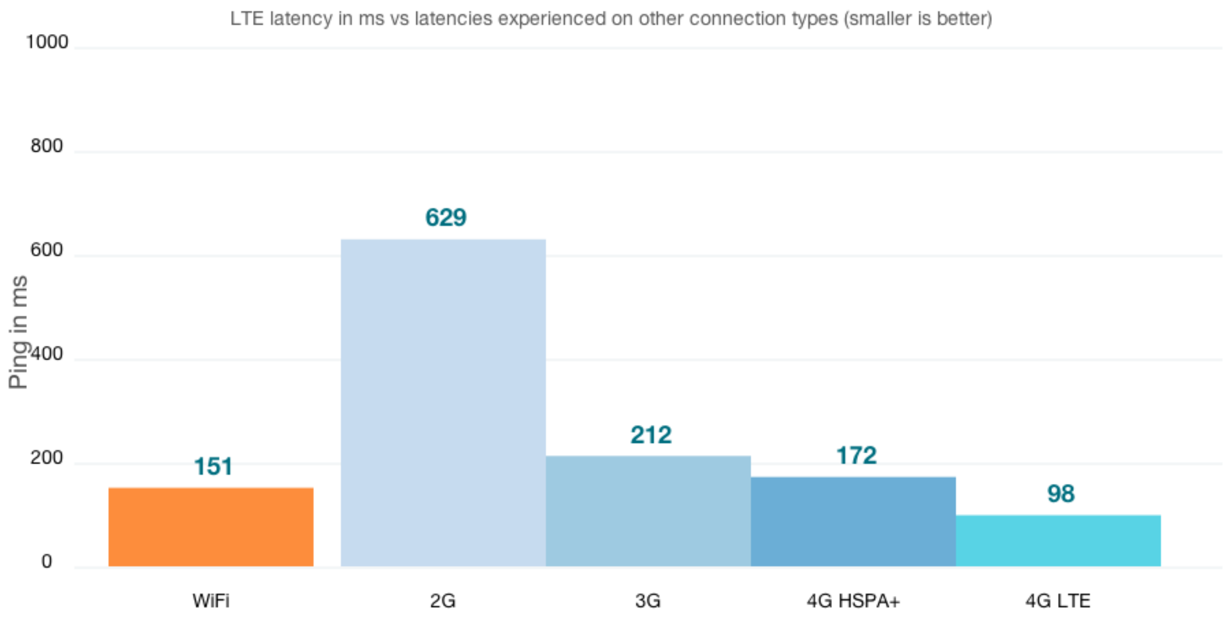


Photo Source: USDOT

- Autonomous Driving
  - For platooning control
  - Latency < 100 ms

Communication Latency



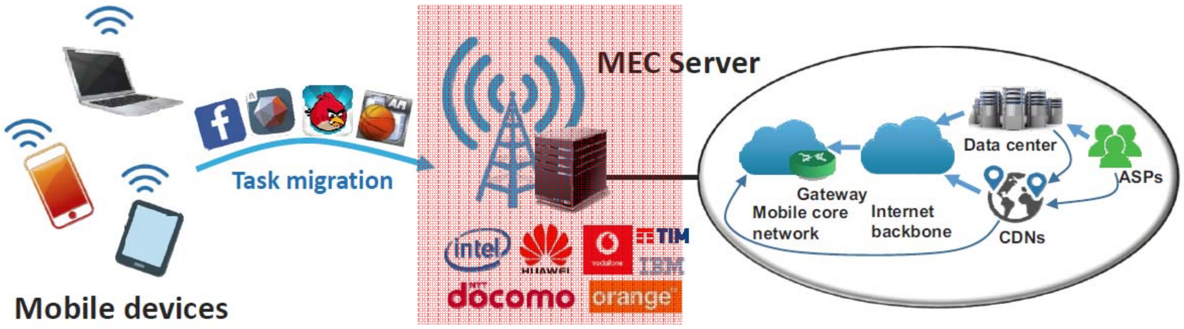
Open Signal, 2014

Communication Latency = **Access Latency** + **Propagation Latency**

5G

Live on the edge

Mobile Edge Computing (MEC)



- *European Telecommunications Standard Institute* (ETSI), 2014
  - MEC “*provides IT and cloud-computing capabilities within the Radio Access Network (RAN) in close proximity to mobile subscribers*”

Ultra-low latency

Mobile energy reduction

Context-awareness

Privacy/Security enhancement

Two Representative Problems in MEC

1. Computation Offloading
   - Which tasks to offload? When?
   - Difficulties: Multipath fading, limited power…

2. Resource Management
   - **Radio resource management**: power control, channel allocation, etc.
     - Communication for computing
   - **Computation resource management**: job scheduling, dynamic voltage and frequency scaling (DVFS)

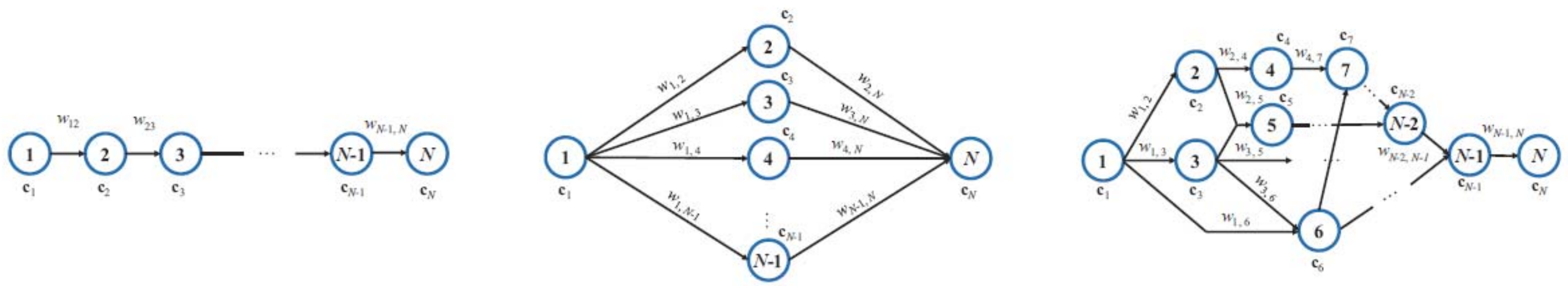
Joint radio and computation resource management is needed

For more research problems:

[9] Y. Mao, C. You, **J. Zhang**, K. Huang, and K. B. Letaief, "A survey for mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322-2358, 4th Quart. 2017.

Offloading Models

- Binary Offloading
  - Task is executed as a whole either locally or remotely
- Data-Partition Model
  - Input bits are bit-wise independent and can be arbitrarily divided into different groups
- Task-Call Graph Model
  - Most general, not well investigated



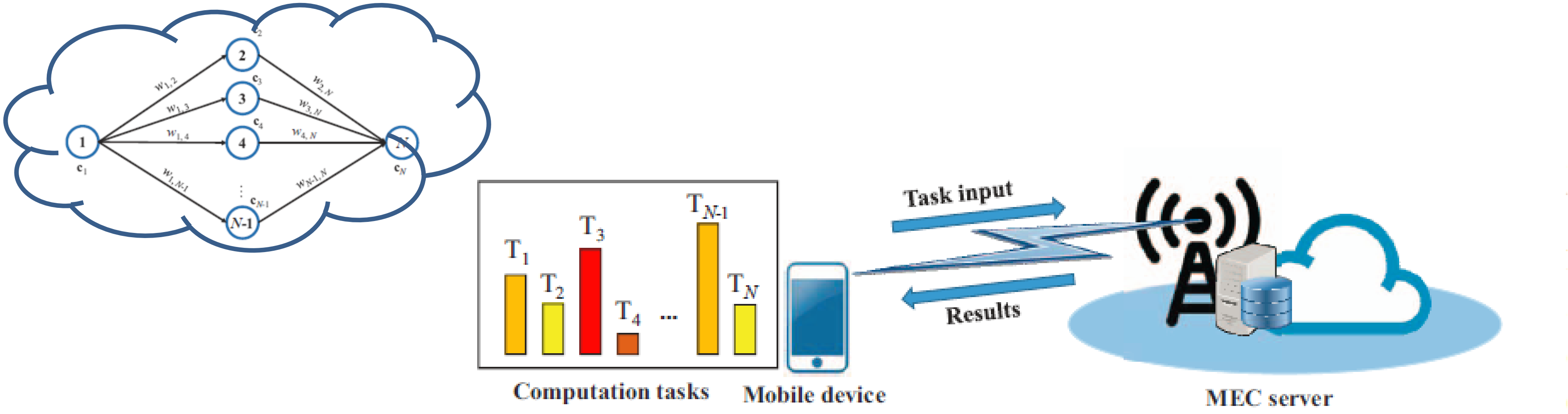
(a) Sequential dependency

(b) Parallel dependency

(c) General dependency

Problem 1: Task Offloading Scheduling

System Model



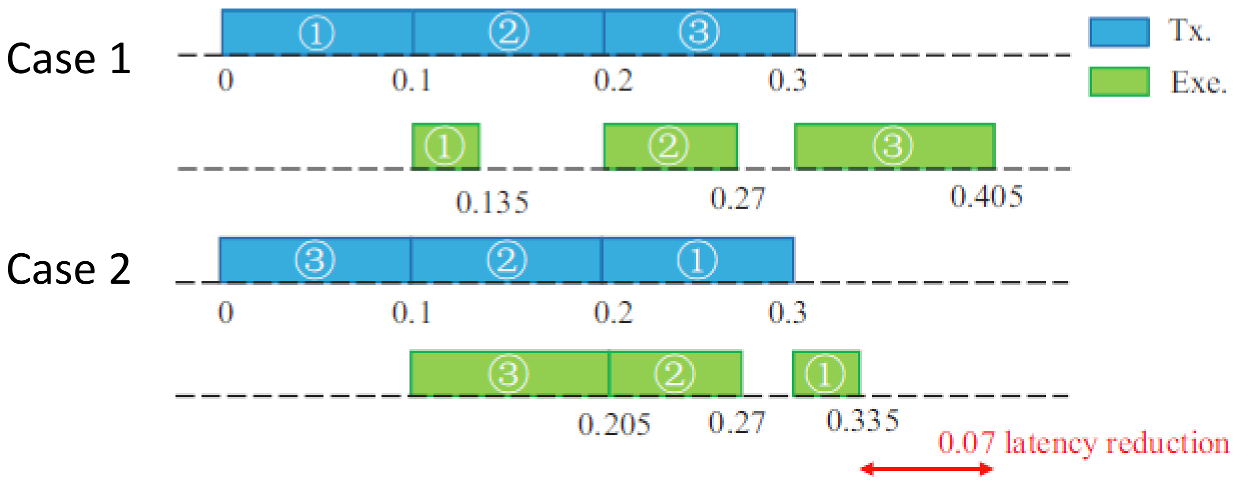
- Device has scarce computation resource

  → all tasks are offloaded

- Limited resources
  - A single communication channel
  - A single-core CPU at the edge server (FIFO)

Impact of the Scheduling Order

- Different tasks have
  - Different offloading data sizes (Communication latency)
  - Different computation intensities (Computation latency)
- Affected by both communication and computation resources.



- **Problem**: ***<u>How to determine the optimal scheduling order to minimize the overall completion time?</u>***

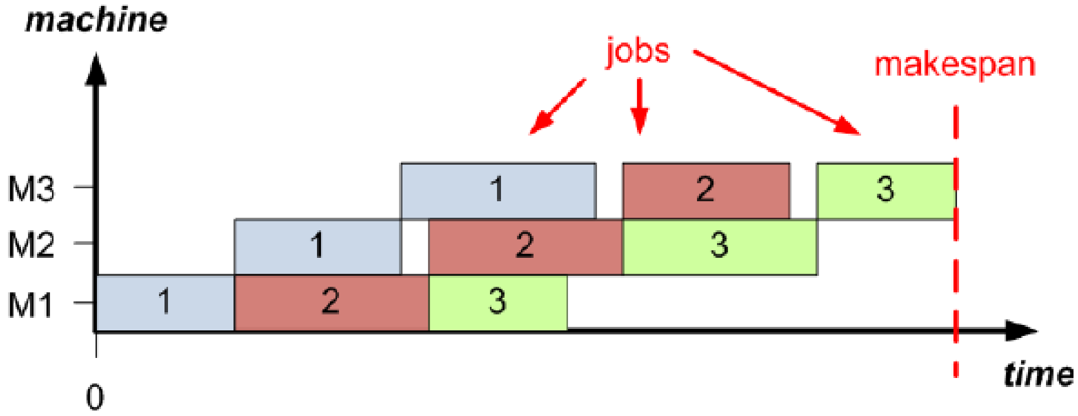
Problem 1: Task Offloading Scheduling

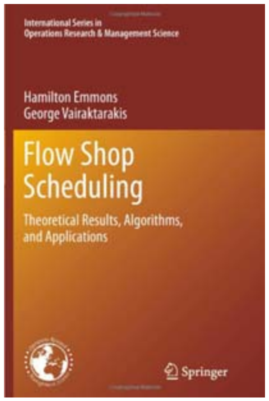
Flow Shop Scheduling

- Large design space: N!

- Not NP-Hard!
  - Offloading time → processing time at machine 1
  - Execution time → processing time at machine 2

➢ **(Two Machine) Flow Shop Problem**




International Series in Operations Research & Management Science

Hamilton Emmons
George Vairaktarakis

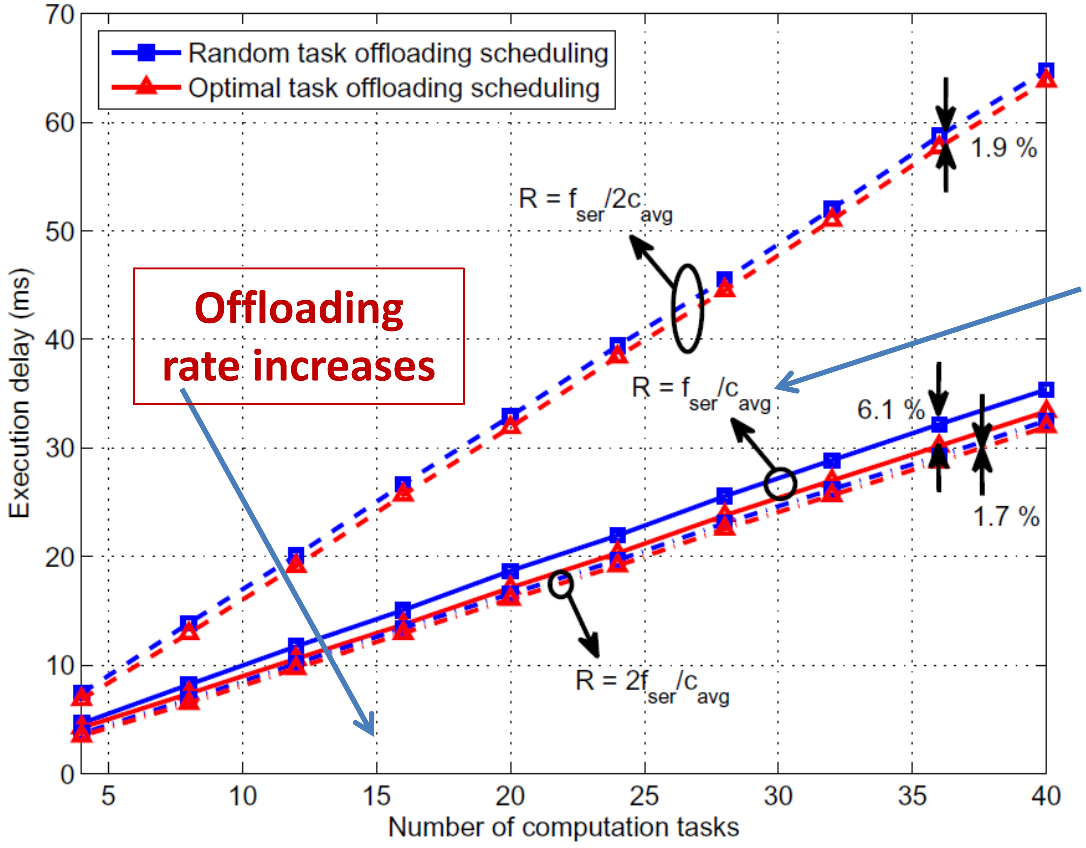
Flow Shop Scheduling

Theoretical Results, Algorithms, and Applications

Springer

- Optimal solution: **Johnson's Algorithm**

Simulation Results



Offloading rate increases

average transmission time = average processing time

- Optimal scheduling is more critical when radio resource and computational resource are balanced.

[10] Y. Mao, **J. Zhang**, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Networking Conf. (WCNC)*, San Francisco, CA, Mar. 2017.

Problem 1: Task Offloading Scheduling

Difficult to Generalize

- Most of extensions of the flow shop scheduling problems are NP-Hard [Garey et al. 1976]

✓ Multiple users, 1 edge server (Not NP-Hard)

❑ Consider feeding back computation results

  – 3-machine flow shop (NP-Hard)

❑ Multiple edge servers

  – Hybrid flow shop model with lags/machine assignment (NP-Hard)

❑ Enable mobile execution

  – Offloading decision/order optimization (NP-Hard)

Problem 2: Stochastic Resource Management

Stochastic Models

- Limitations of previous works
  - Assume task offloading and execution within one coherent block
  - However, typically
    - Offloading process ~ tens of milliseconds
    - Channel coherence block ~ a few milliseconds

  → need to consider **stochastic** channels
  - Assume one task in each slot for each user

  → need to consider **stochastic** task arrivals

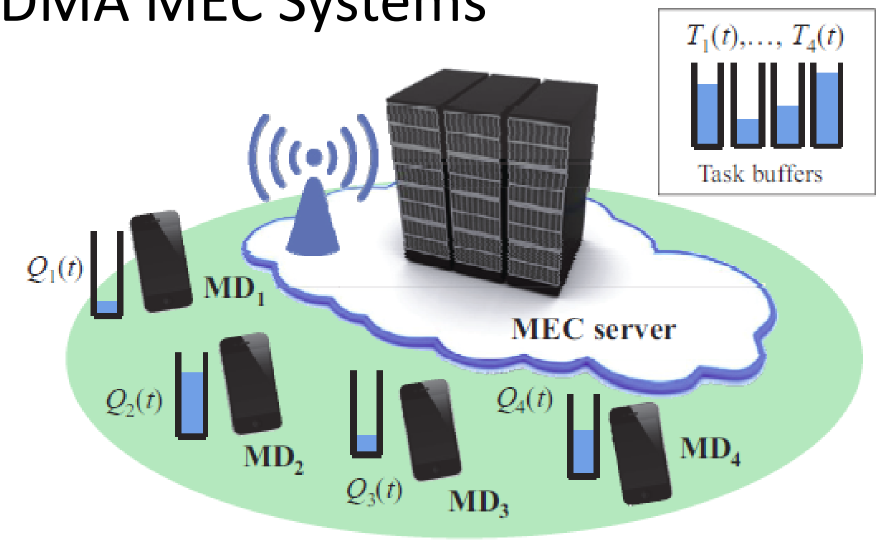
- Stochastic joint radio and computation resource management in multiuser MEC systems [11]
  - Radio resource management: power control and bandwidth allocation
  - Computation resource management: MEC scheduling, DVFS

[11] Y. Mao, **J. Zhang**, S.H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994-6009, Sept. 2017.

Problem 2: Stochastic Resource Management

System Model

- Multi-user FDMA MEC Systems



  - Queuing model
    - Mobile side: $Q_i(t+1) = (Q_i(t) - D_{\Sigma,i}(t))^+ + A_i(t)$ Task arrival (bits)
    - Server side: $T_i(t+1) = (T_i(t) - D_{s,i}(t))^+ + \min\{(Q_i(t) - D_{l,i}(t))^+, D_{r,i}(t)\}$

$\propto f_{l,i}(t)$ ↑

Power-rate function

CSI $\Gamma_i(t)$

  - Mobile/server CPU speeds, $f_{l,i}(t)/f_{C,m}(t)$
  - MEC scheduling decision, $D_{s,i}(t)$
  - Transmit power and bandwidth allocation, $p_{tx,i}(t)$ and $a_i(t)$

Problem 2: Stochastic Resource Management

Problem Formulation

- Average weighted sum power minimization

$$\mathcal{P}_2: \min_{\{\mathrm{X}(t)\}} \lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{N}} w_i\left(p_{\mathrm{tx},i}(t)+p_{l,i}(t)\right)+w_{N+1}p_{\mathrm{ser}}(t)\right]$$

$$\mathbf{X}(t) \triangleq [\mathbf{f}(t), \mathbf{p}_{\mathrm{tx}}(t), \boldsymbol{\alpha}(t), \mathbf{f}_C(t), \mathbf{D}_s(t)],$$

Local execution power

Server execution power

$$\text{s.t} \quad 0 \le f_{l,i}(t) \le f_{i,\max}, i\in\mathcal{N}, t\in\mathcal{T}$$

$$0 \le f_{C,m}(t) \le f_{C_m,\max}, m\in\mathcal{M}, t\in\mathcal{T}$$

CPU speed constraints

$$0 \le p_{\mathrm{tx},i}(t) \le p_{i,\max}, i\in\mathcal{N}, t\in\mathcal{T}$$

$$\boldsymbol{\alpha}(t)\in\mathcal{A}, t\in\mathcal{T}$$

Tx power and bandwidth allocation constraints

$$\mathcal{A} = \{\boldsymbol{\alpha}|\alpha_i \ge \epsilon_A, \sum\nolimits_{i\in\mathcal{N}} \alpha_i \le 1\}, \epsilon_A \searrow 0^+$$

$$\sum_{i\in\mathcal{N}} D_{s,n}(t)L_n \le \sum_{m\in\mathcal{M}} f_{C,m}(t)\tau, t\in\mathcal{T}$$

$$D_{s,i}(t)\ge 0, i\in\mathcal{N}, t\in\mathcal{T}$$

Server scheduling constraints

$$\lim_{T\to+\infty}\frac{\mathbb{E}[|Q_i(T)|]}{T} = 0, i\in\mathcal{N}$$

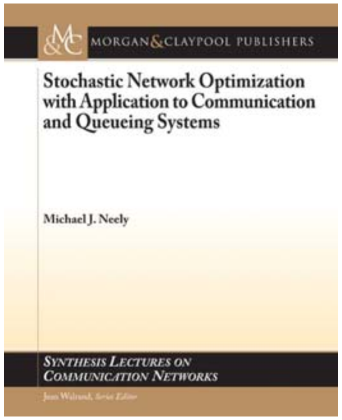
$$\lim_{T\to+\infty}\frac{\mathbb{E}[|T_i(T)|]}{T} = 0, i\in\mathcal{N}$$

Mean rate stability

A challenging stochastic optimization problem!

Problem 2: Stochastic Resource Management

Proposed Solution



MORGAN & CLAYPOOL PUBLISHERS

Stochastic Network Optimization with Application to Communication and Queueing Systems

Michael J. Neely

SYNTHESIS LECTURES ON COMMUNICATION NETWORKS

Jean Walrand, Series Editor

- Online resource management (Lyapunov optimization)
  - Solve a deterministic optimization problem at each time slot

$$\min_{\mathrm{X}(t)} \quad -\sum_{i\in\mathcal{N}} Q_i(t)\, D_{\Sigma,i}(t) - \sum_{i\in\mathcal{N}} T_i(t)\left(D_{s,i}(t) - D_{r,i}(t)\right) + V \cdot P_{\Sigma}(t)$$

$$\text{s.t} \quad \text{All constraints in } \mathcal{P}_2 \text{ except the stability constraints}$$

An UB of the Lyapunov drift-plus-penalty

- The average weighted sum power consumption satisfies

$$\overline{P}_{\Sigma}^{\star} \le P_{\Sigma,\mathcal{P}_2}^{\mathrm{opt}} + \frac{C}{V}$$

- Average sum queue length of the task buffer satisfies

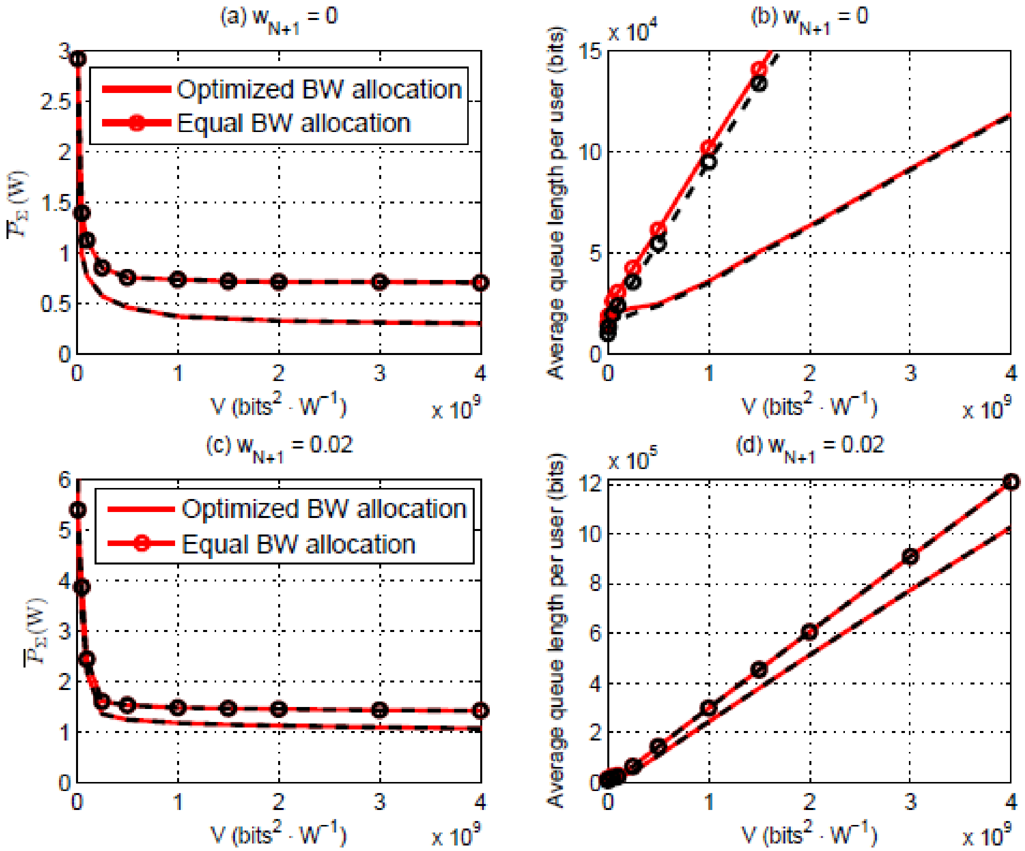
$$\lim_{T\to+\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i\in\mathcal{N}} \left(Q_i(t) + T_i(t)\right)\right] \le \frac{C + V\cdot\left(\Psi(\epsilon) - P_{\Sigma,\mathcal{P}_2}^{\mathrm{opt}}\right)}{\epsilon}$$

Power-delay tradeoff: [O(1/V), O(V)]

<u>Problem 2: Stochastic Resource Management</u>

Simulation Results

- Benchmark: Equal bandwidth allocation



N = 5, λ_i = 4 kbits/slot

Verify the [O(1/V), O(V)] power-delay tradeoff

Benefits of joint resource management on power and delay performance for MEC

Conclusions

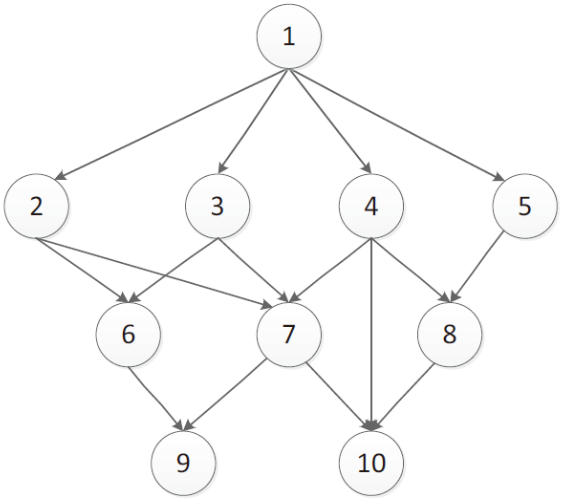
- Critical to **jointly** consider radio and computation resources

- General offloading models are practically important
  - More efforts are needed

- Stochastic models are necessary
  - Efficient online algorithms are needed

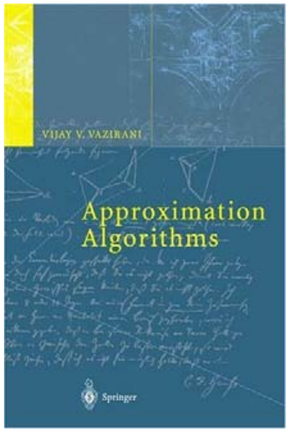
Extension: General Task Models (i)

- General dependency is Hard!



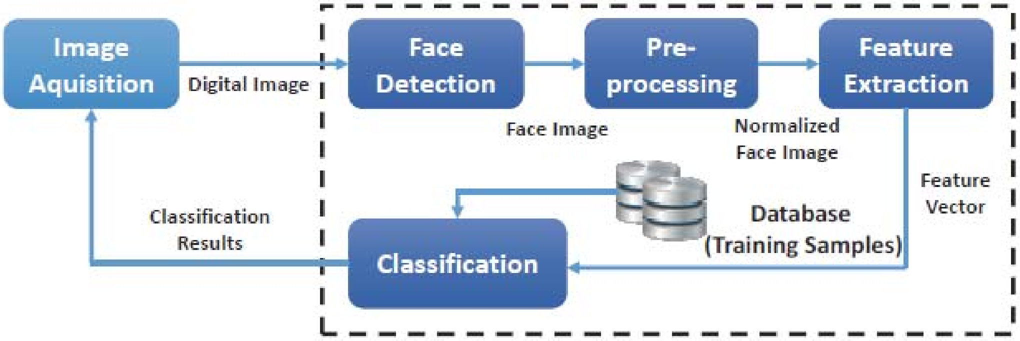
"I can't find an efficient algorithm, but neither can all these famous people."

Extension: General Task Models (ii)



- Option I
  - Approximation algorithms

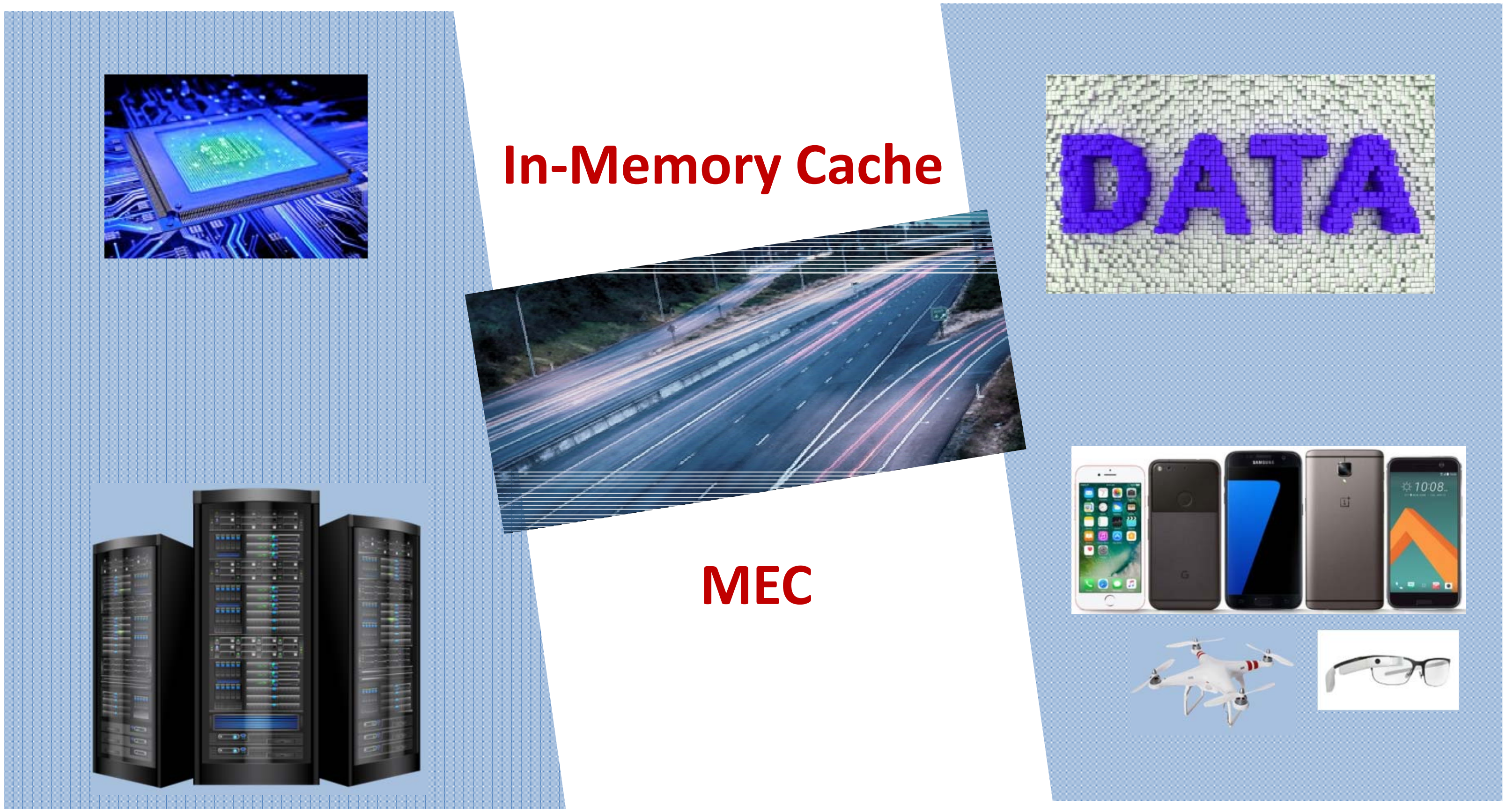
- Option II
  - Focus on important and interesting cases
  - Example: face recognition



Offloadable Tasks

- **NP-hardness does not prevent developing practically useful algorithms**

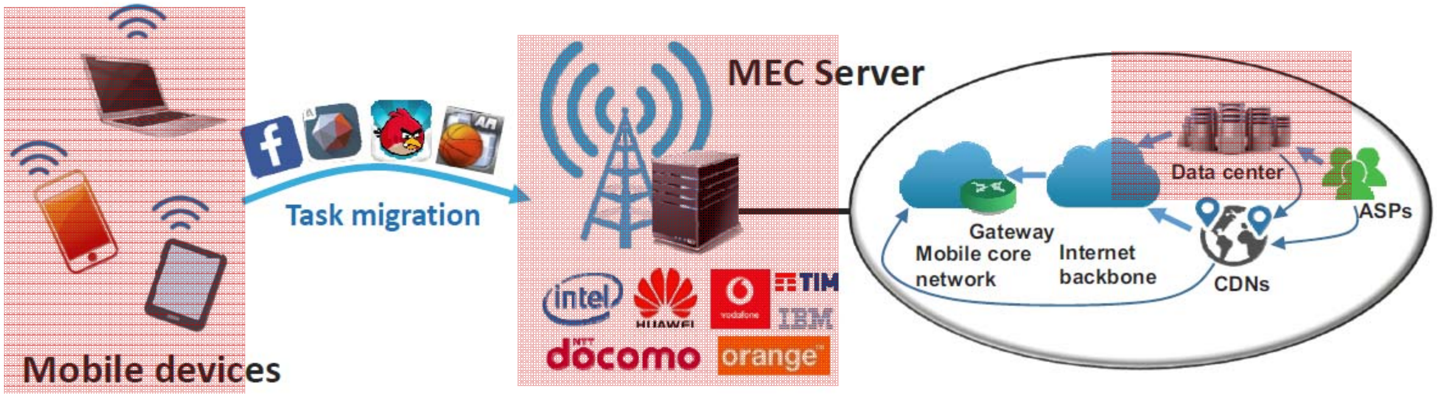
Overcome Long Distance



In-Memory Cache

MEC

Takeaways



- Different computing platforms are needed to support mobile intelligence
  - **Cloud** + **Edge** + **On-Device**
  - A holistic view is needed

- **Communication** + **computing** + **data** + **algorithm**
  → **mobile intelligence**
  - Pay attention to the **bottleneck**

My Research Interest

- Wireless Communications
  - Dense cooperative networks
  - Network analysis via stochastic geometry
  - Millimeter-wave communications
  - Wireless caching
- Distributed Computing Systems
  - Big data analytics systems
  - Mobile edge computing

- For more information
  - http://www.ece.ust.hk/~eejzhang/

Thank you!