IEEE CGC 2012, November 1-3, Xiangtan, Hunan, China

Dynamic Grouping Strategy in Cloud Computing

Presenter : Qin Liu[a]

Joint work with

Yuhong Guo[b], Jie Wu[b] , and Guojun Wang[a]

[a] Central South University, China
[b] Temple University, USA

2012-11-2

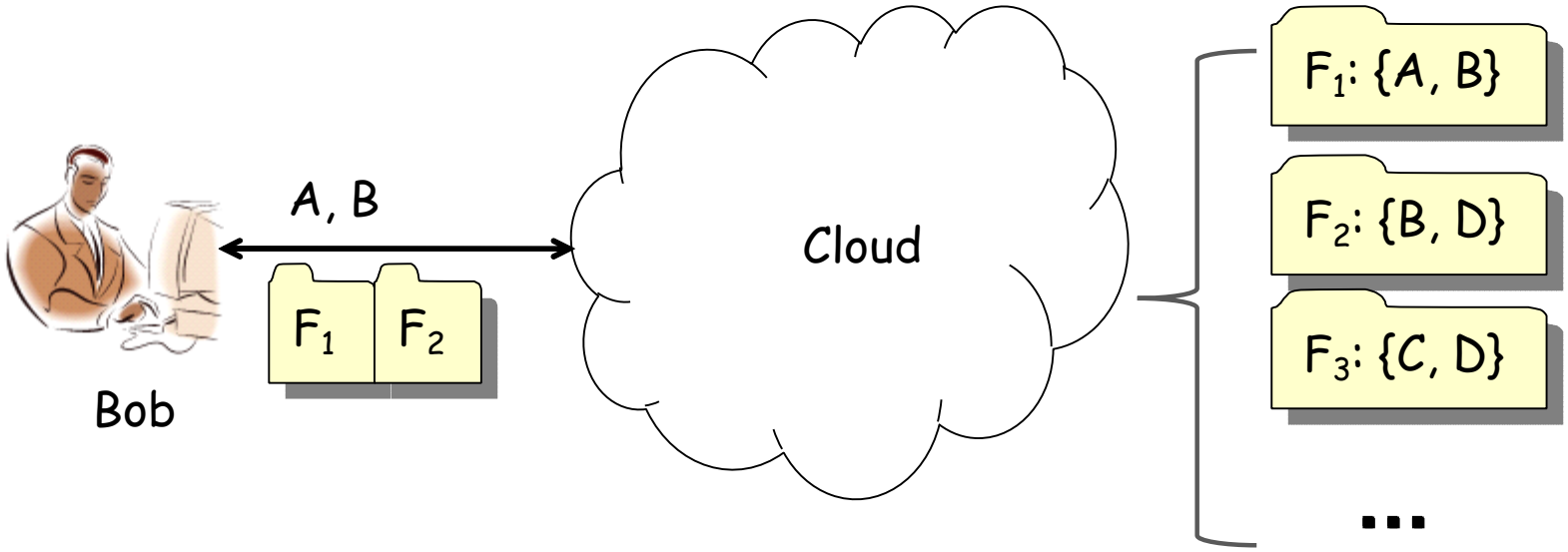
Outline

1. Introduction
2. Preliminaries
3. K-Mean-based Dynamic Grouping
4. Extensions
5. Evaluation
6. Conclusion & Future work

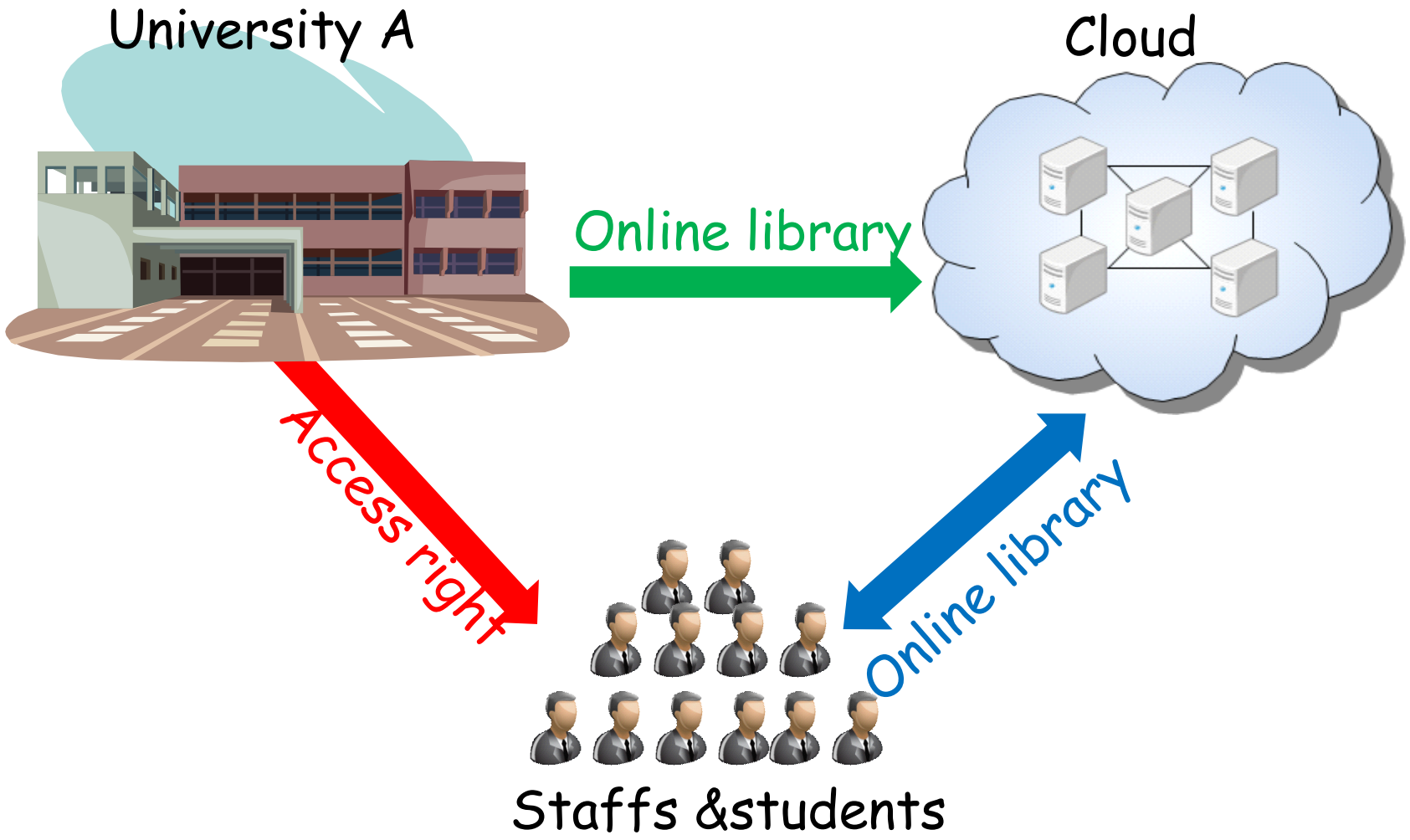
Introduction

Cloud computing model

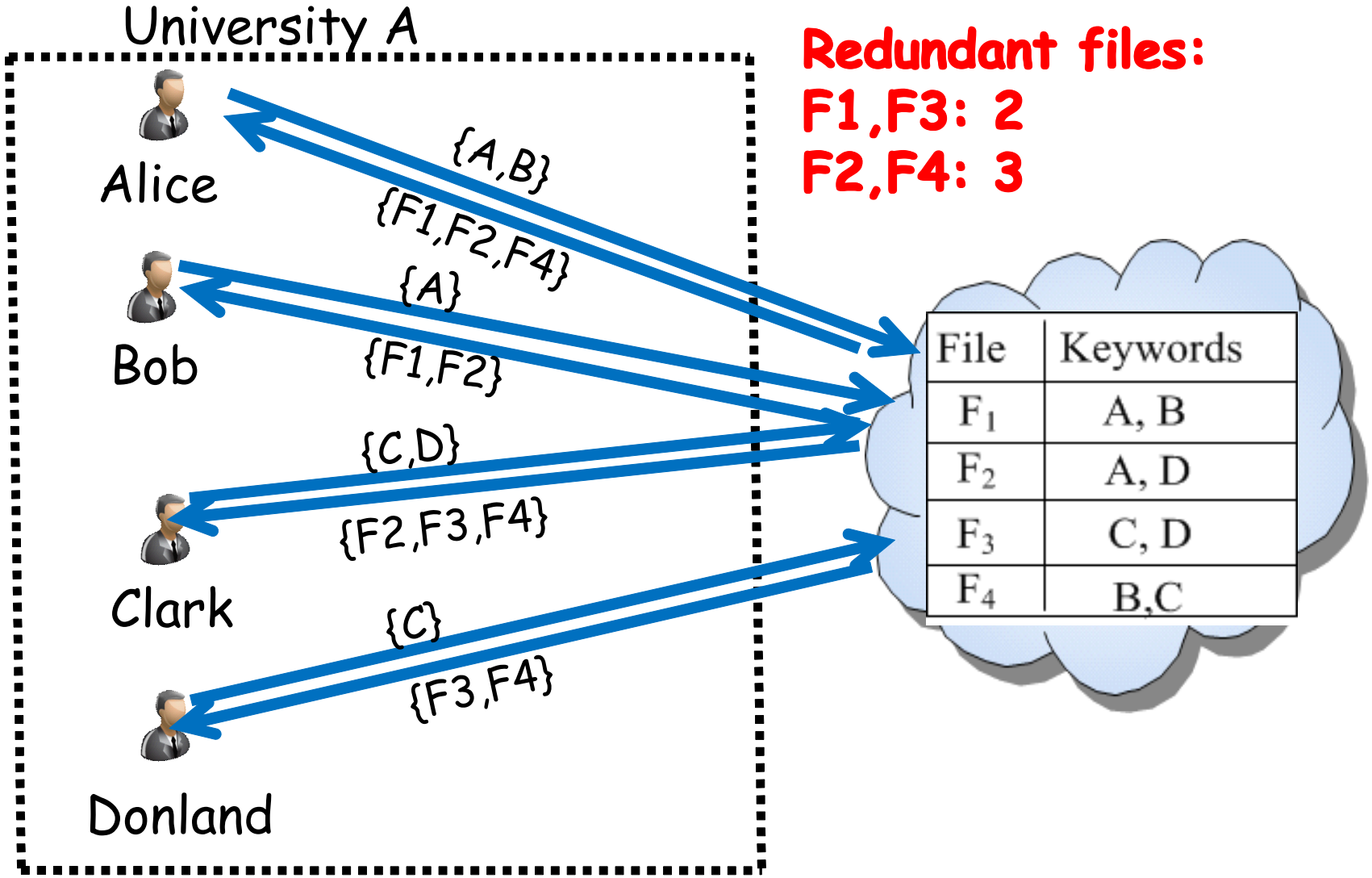
- Cloud computing has emerged as a new type of commercial paradigm due to its overwhelming advantages, such as flexibility, scalability, and cost efficiency



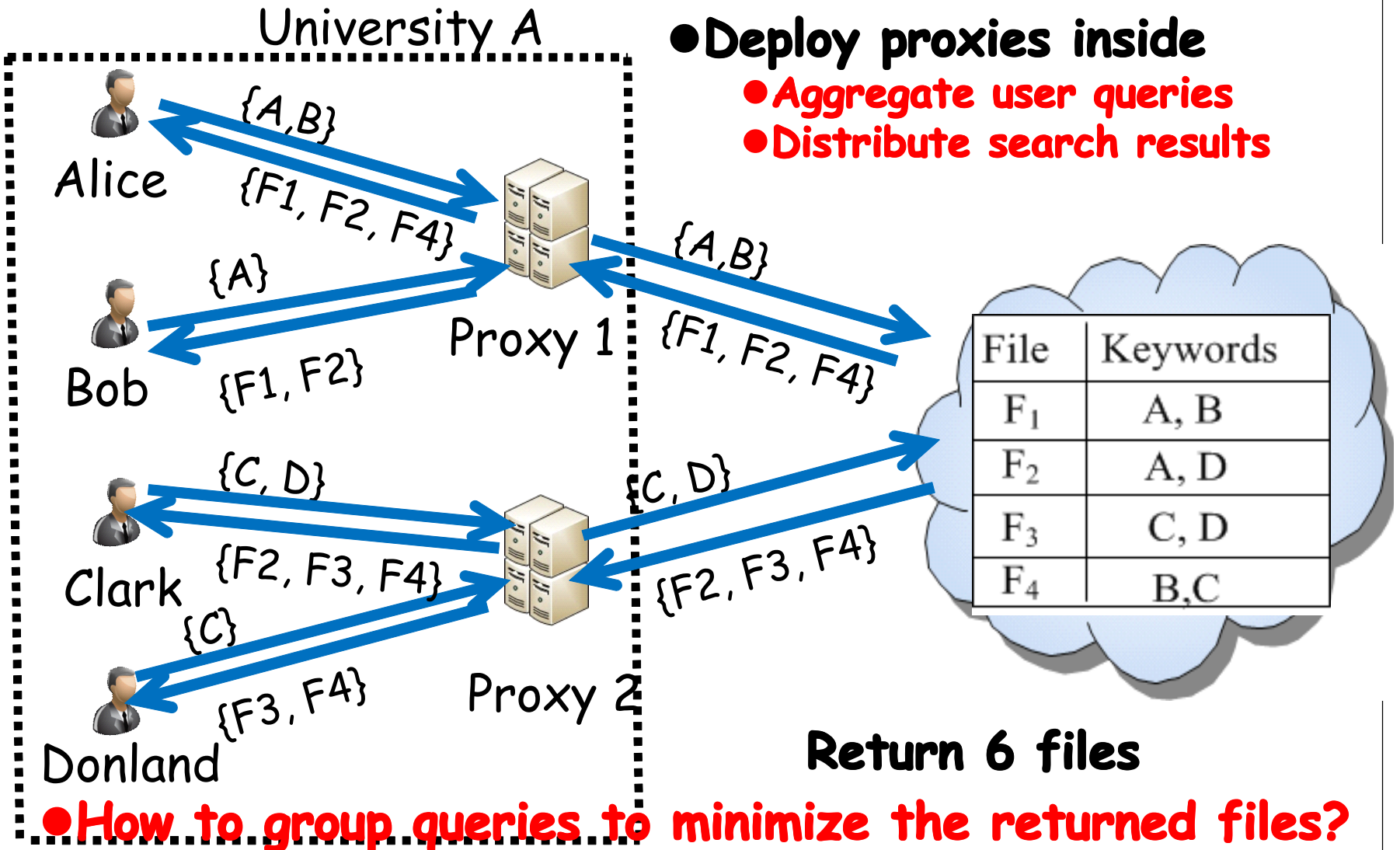
Application scenario



Problem: Cost grows linearly



New chanllenge

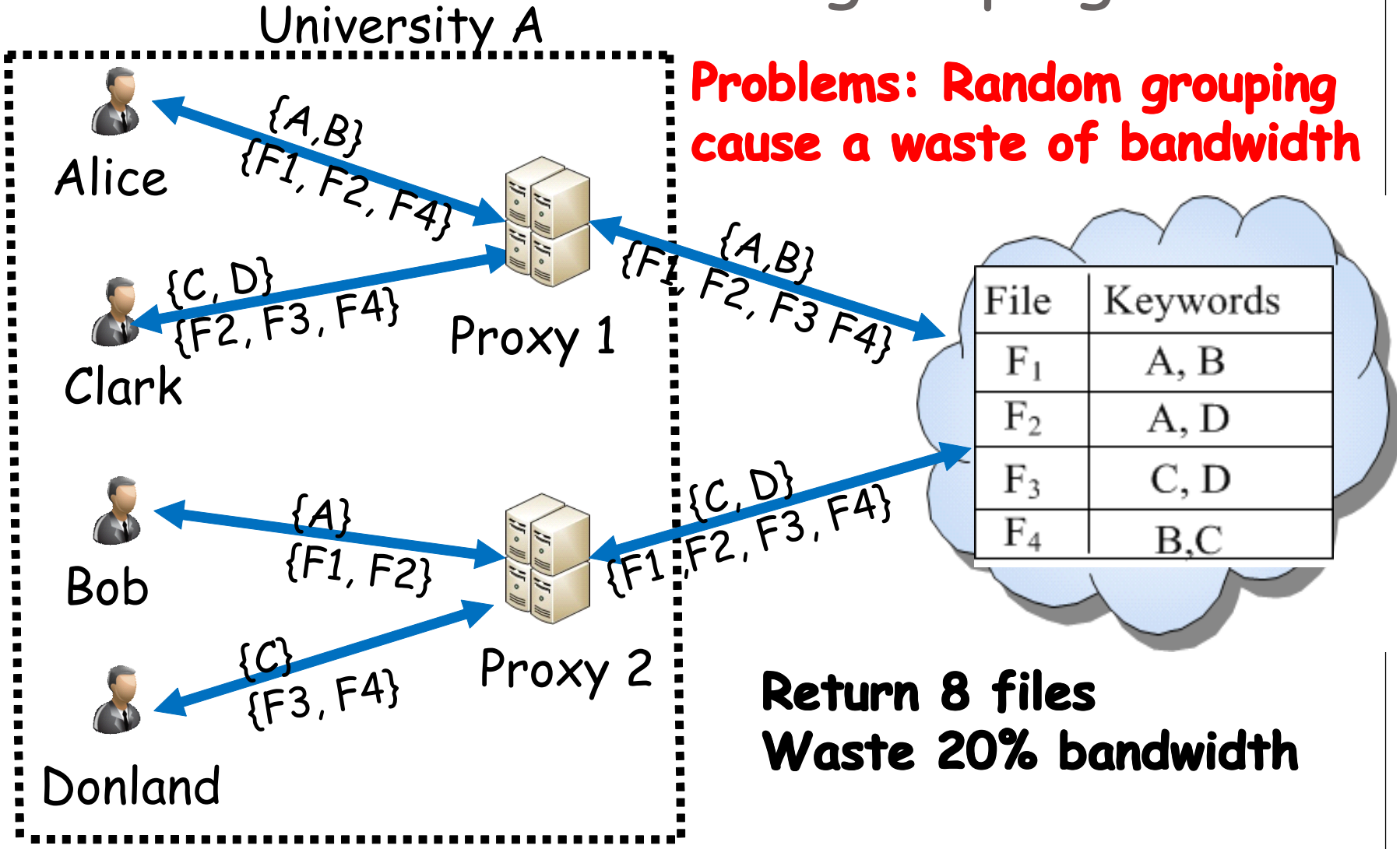


- **Deploy proxies inside**
  - **Aggregate user queries**
  - **Distribute search results**

Return 6 files

- **How to group queries to minimize the returned files?**

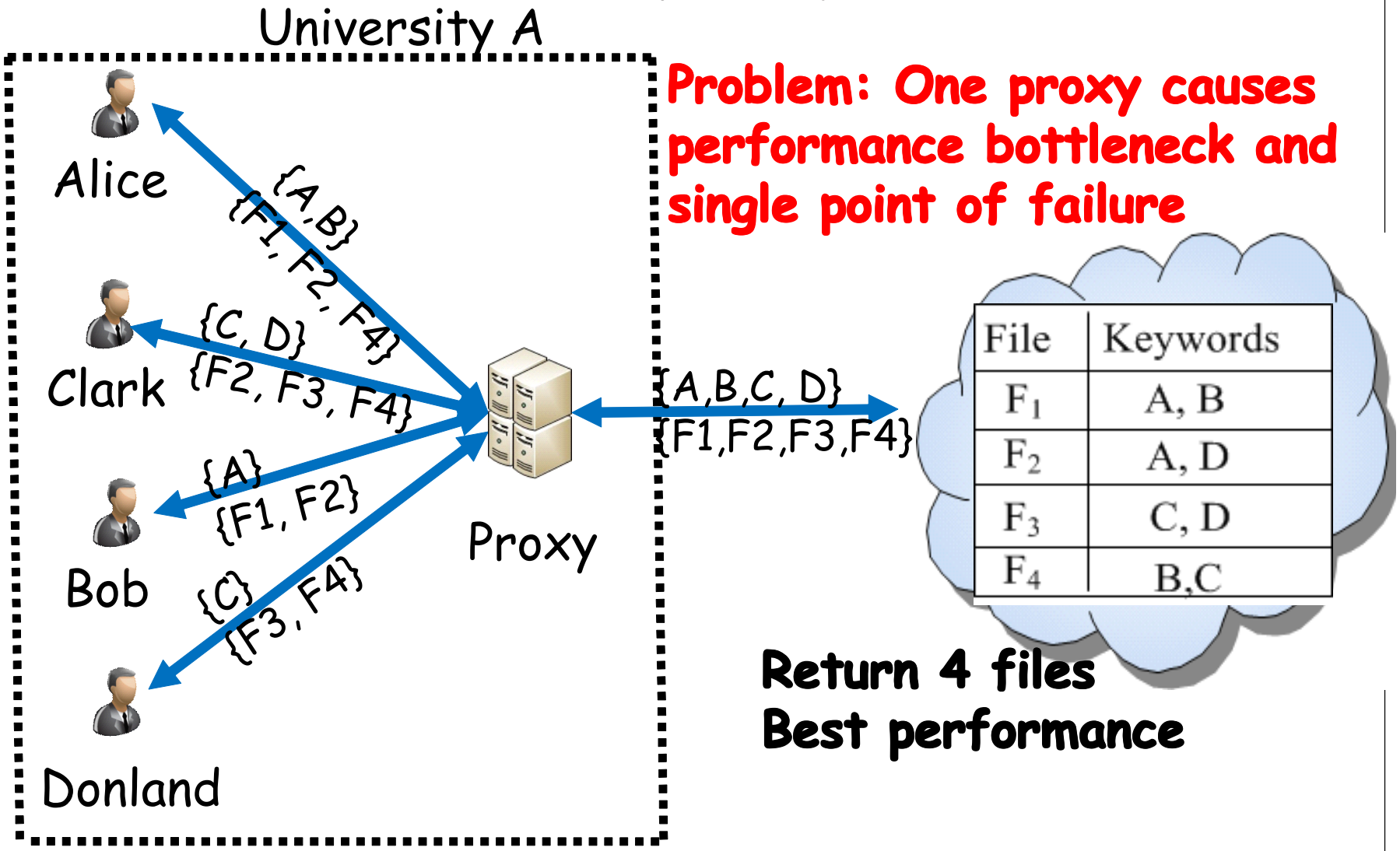
Naïve solution: Random grouping



Problems: Random grouping cause a waste of bandwidth

Return 8 files
Waste 20% bandwidth

Naïve solution: One proxy



Problem: One proxy causes performance bottleneck and single point of failure

Return 4 files
Best performance

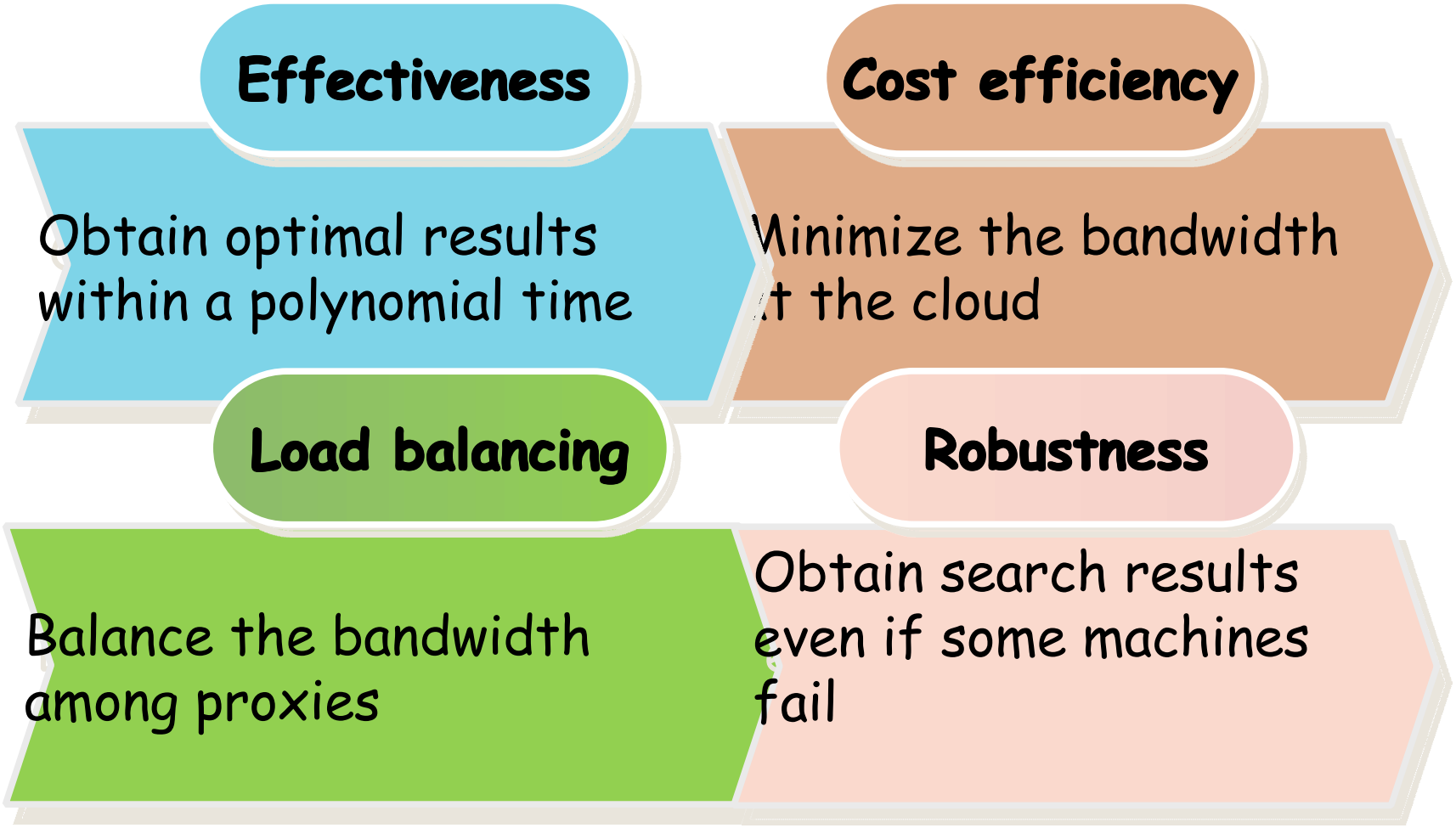
K-Mean-based Dynamic Grouping (KMDG)

- Classify n queries into k groups in the case of k proxy servers, so that each group size equals to n/k and the number of returned files is minimized.

- NP-Hard problem——Heuristic grouping strategy

- Basic strategy: KMDG (based on K-Means)
- Extensions:
  - KMDG1-Robust version
  - KMDG2-Relax the constraint of equal group size

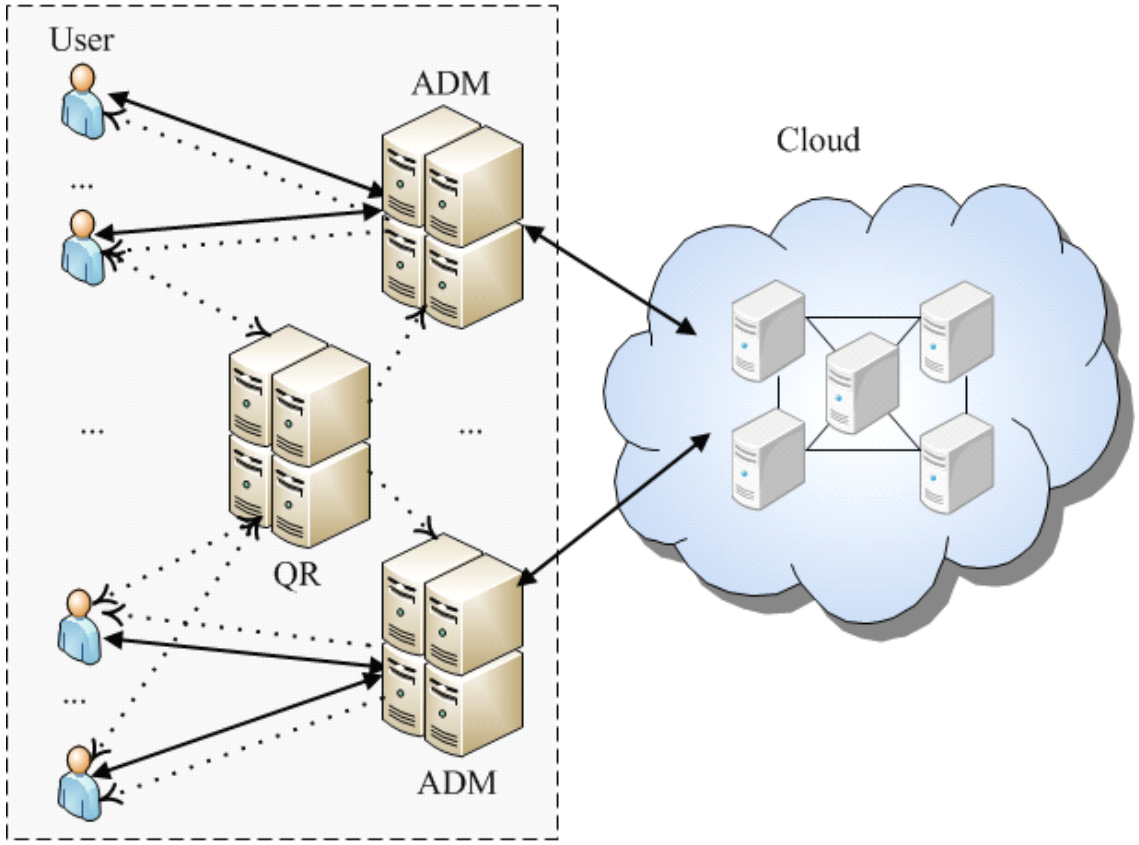
Design goals



Preliminaries

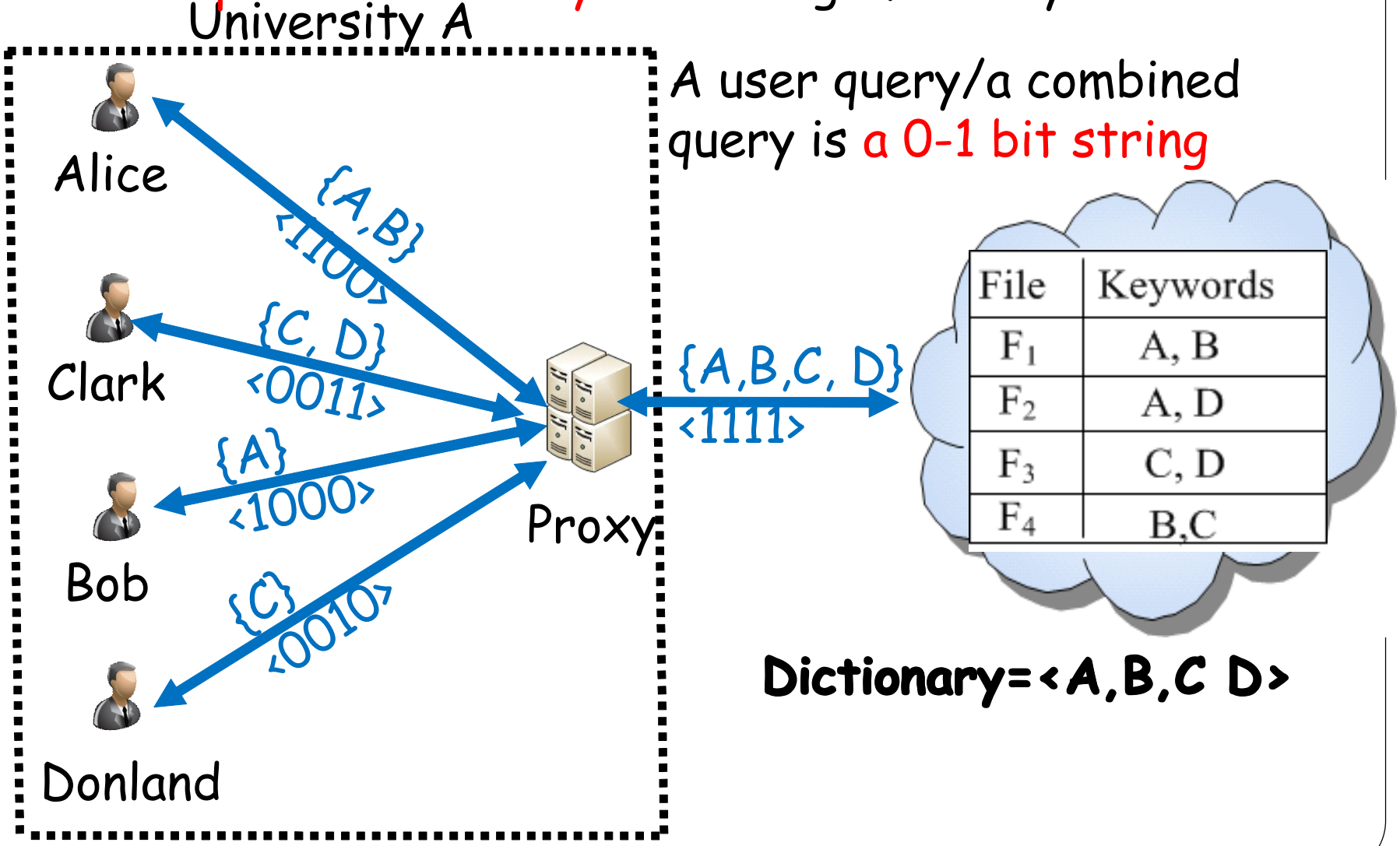
System model

- The cloud, many users, and many proxy servers
  - Query router (QR)
  - Aggregation and distribution machines (ADMs)

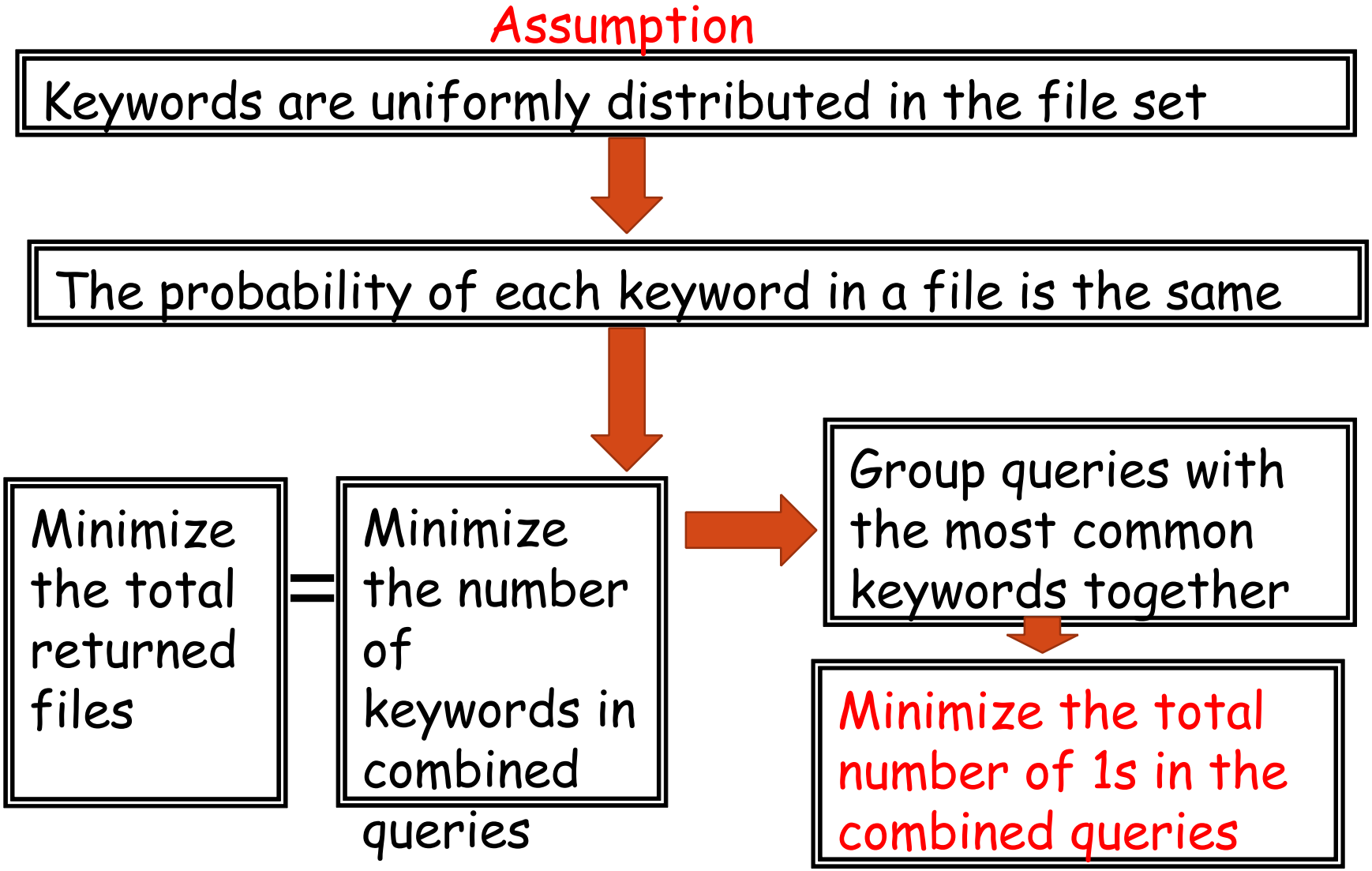


System model

Given a public dictionary consisting of all keywords



Problem formulation



Parameter analysis

- The expected value of the number of returned files can be calculated with Eq.1

$$\sum_{j=1}^{k} t \cdot (1 - (1 - \gamma/d)^{\hat{S}_j}). \quad (1)$$

d: the number of keywords in the dictionary
γ : the average number of keywords in a file
k: the number of groups/proxies
t: the number of files in the cloud
$\hat{S}_j$ (grouping cost) : the average number of keywords in the j-th combined query

K-Mean-based Dynamic Grouping (KMDG)

Definitions

Group seed. *For $1 \leq j \leq k$, the group seed s_j is the center and first member of group g_j.*

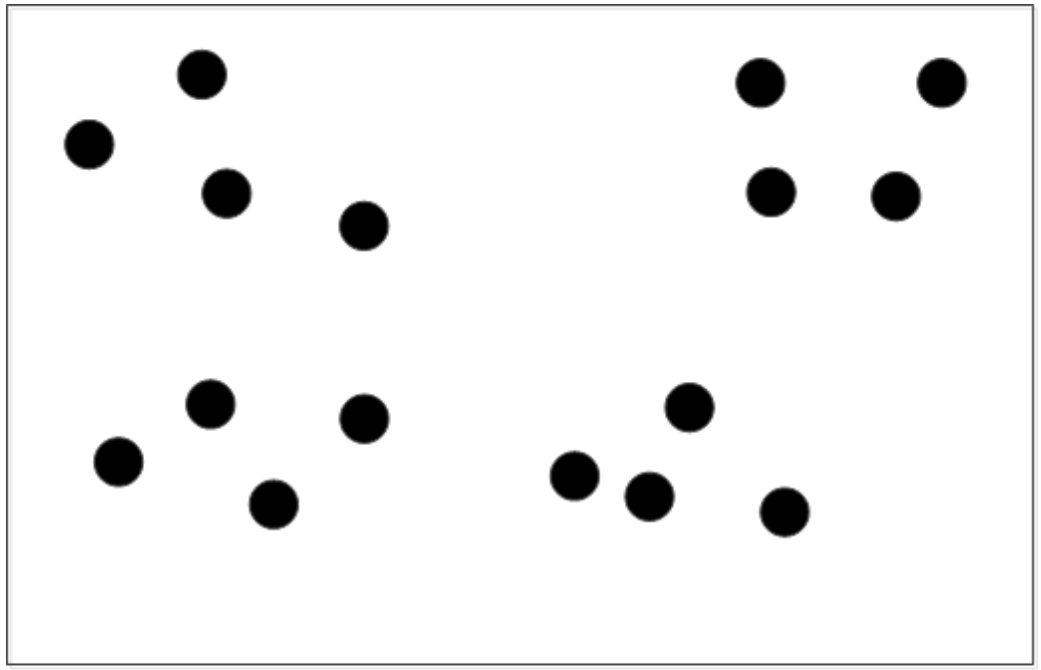
Distance. *The distance between query Q_i and query Q_j, denoted as $Dist(Q_i, Q_j)$, is the number of increased 1s for Q_i after combining with Q_j.*

For example, if $Q_1 = \langle 11100000 \rangle$ and $Q_2 = \langle 11000000 \rangle$, $Dist(Q_1, Q_2) = 0$, and $Dist(Q_2, Q_1) = 1$.

Nearest neighbor. *Given a query Q_j, query Q_i with the minimal distance from Q_j is called as Q_j's nearest neighbor.*

High level ideas

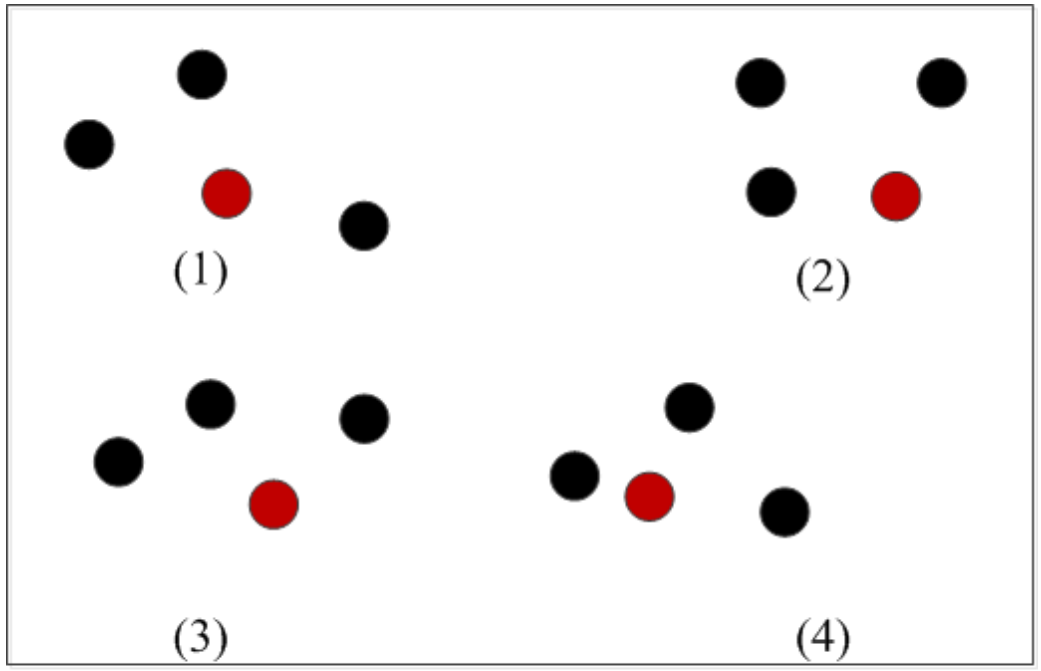
- Suppose each query denotes a node



Given *n* queries

High level ideas

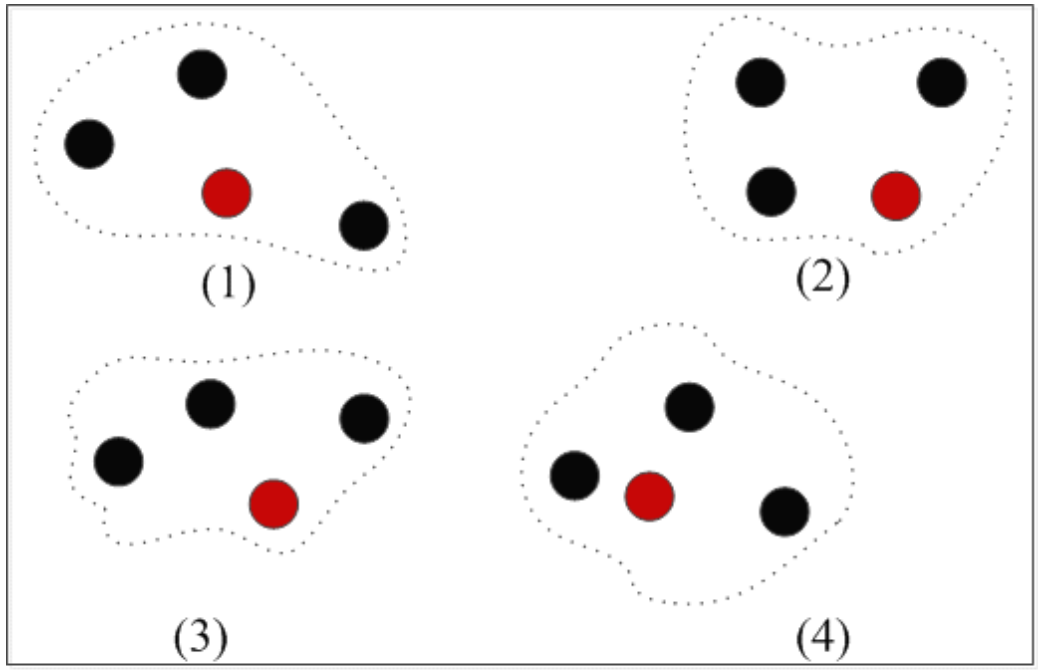
- An improvement: First choose a random node as the seed. For the *i-th* seed, choose the one with the total maximal distance with all *i-1* seeds



Step 1. Randomly choose *k* queries as the seed

High level ideas

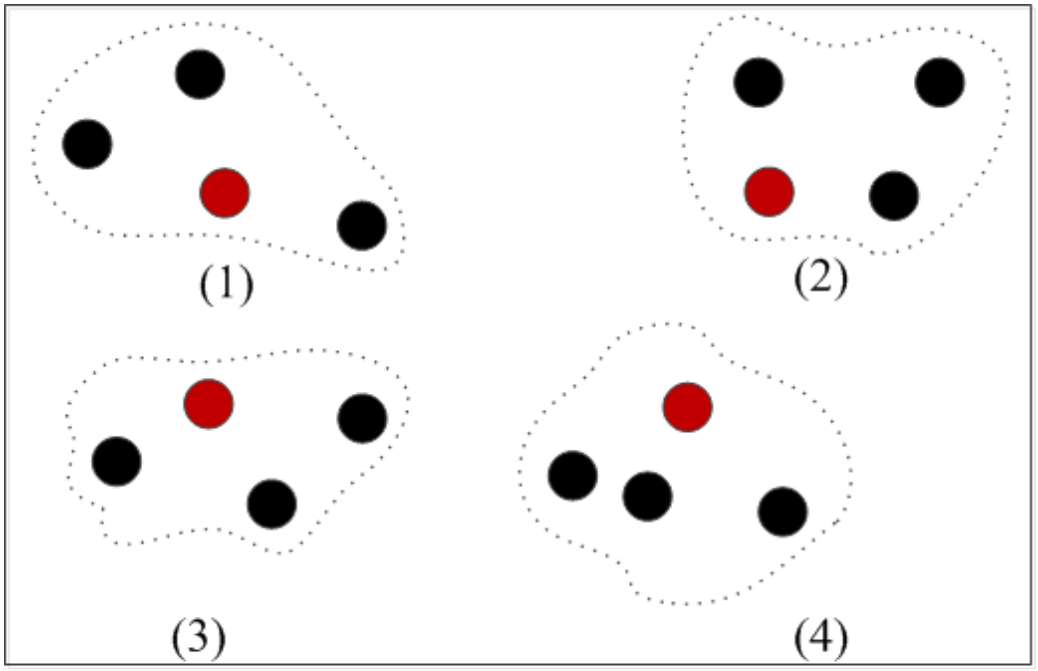
- Closet: the minimal number of increased 1s after being combined with the seed
- s1=1100, s2=0011, s3=1001, s4=0101; Q=1100



Step 2. Classify the queries that are closet to the seed into a group

High level ideas

- After choosing k seeds, grouping process is performed in the same way in next round



Step 3. Randomly choose a query in a group as the seed in the next round

Algorithm

Algorithm 1 KMDG

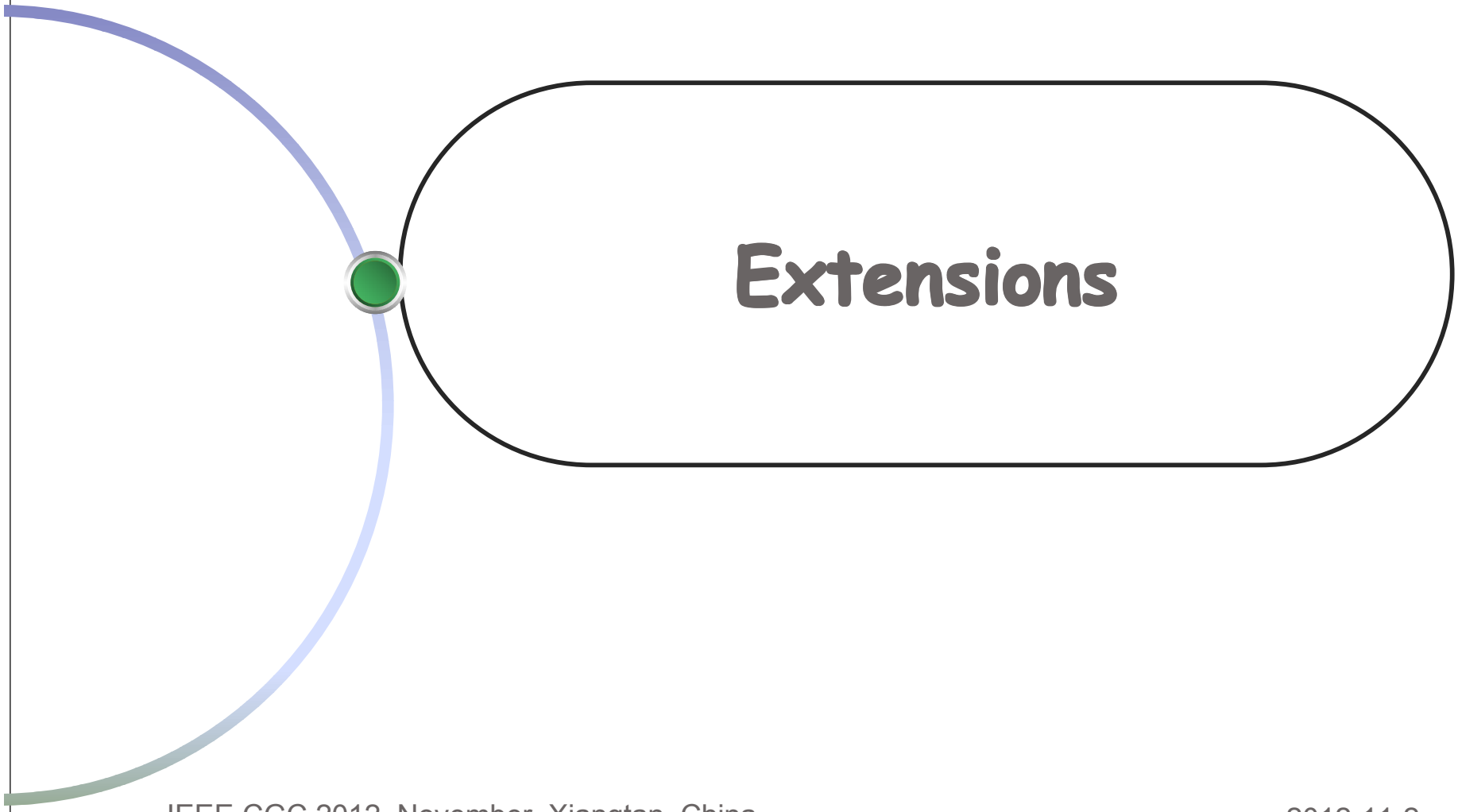
1: Construct a set, $CandiQ$, with the universal user queries
2: Randomly choose k distinct queries from $CandiQ$ as seeds $s_1, \ldots, s_k$ for groups $g_1, \ldots, g_k$, and remove them from $CandiQ$
{Runs the following process multiple rounds}
3: **while** $CandiQ$ is not empty **do**
4: **for** $j = 1$ to k **do**
5: $Neighbor_j$ is a subset of $CandiQ$ that accommodate s_j's nearest neighbors
6: Choose a random element $Q_i \in Neighbor_j$ into g_j and remove it from $CandiQ$
7: Initialize $CandiQ$ with the universal user queries
8: **for** $j = 1$ to k **do**
9: Randomly choose a query from g_j as the seed s_j for the next round and remove it from $CandiQ$

Example

Sample Queries

$Q_1 = \langle 11100000 \rangle \Rightarrow$ (A,B,C)	$Q_5 = \langle 00000111 \rangle \Rightarrow$ (F,G,H)
$Q_2 = \langle 11000000 \rangle \Rightarrow$ (A,B)	$Q_6 = \langle 00000011 \rangle \Rightarrow$ (G,H)
$Q_3 = \langle 11000000 \rangle \Rightarrow$ (A,B)	$Q_7 = \langle 00000011 \rangle \Rightarrow$ (G,H)
$Q_4 = \langle 00010000 \rangle \Rightarrow$ (D)	$Q_8 = \langle 00001000 \rangle \Rightarrow$ (E)

	CandiQ	g_1	g_2	g_3	g_4
Round 1	Q_3 11000000 Q_4 00010000 Q_7 00000011 Q_8 00001000	Q_1 11100000 Min(Dist)=1 $Neighbor_1=\{Q_4\}$	Q_2 11000000 Min(Dist)=0 $Neighbor_2=\{Q_3\}$	Q_5 00000111 Min(Dist)=1 $Neighbor_3=\{Q_8\}$	Q_6 00000011 Min(Dist)=0 $Neighbor_4=\{Q_7\}$
Round 2	NULL	Q_1 11100000 Q_4 00010000	Q_2 11000000 Q_3 11000000	Q_5 00000111 Q_8 00001000	Q_6 00000011 Q_7 00000011


Extensions

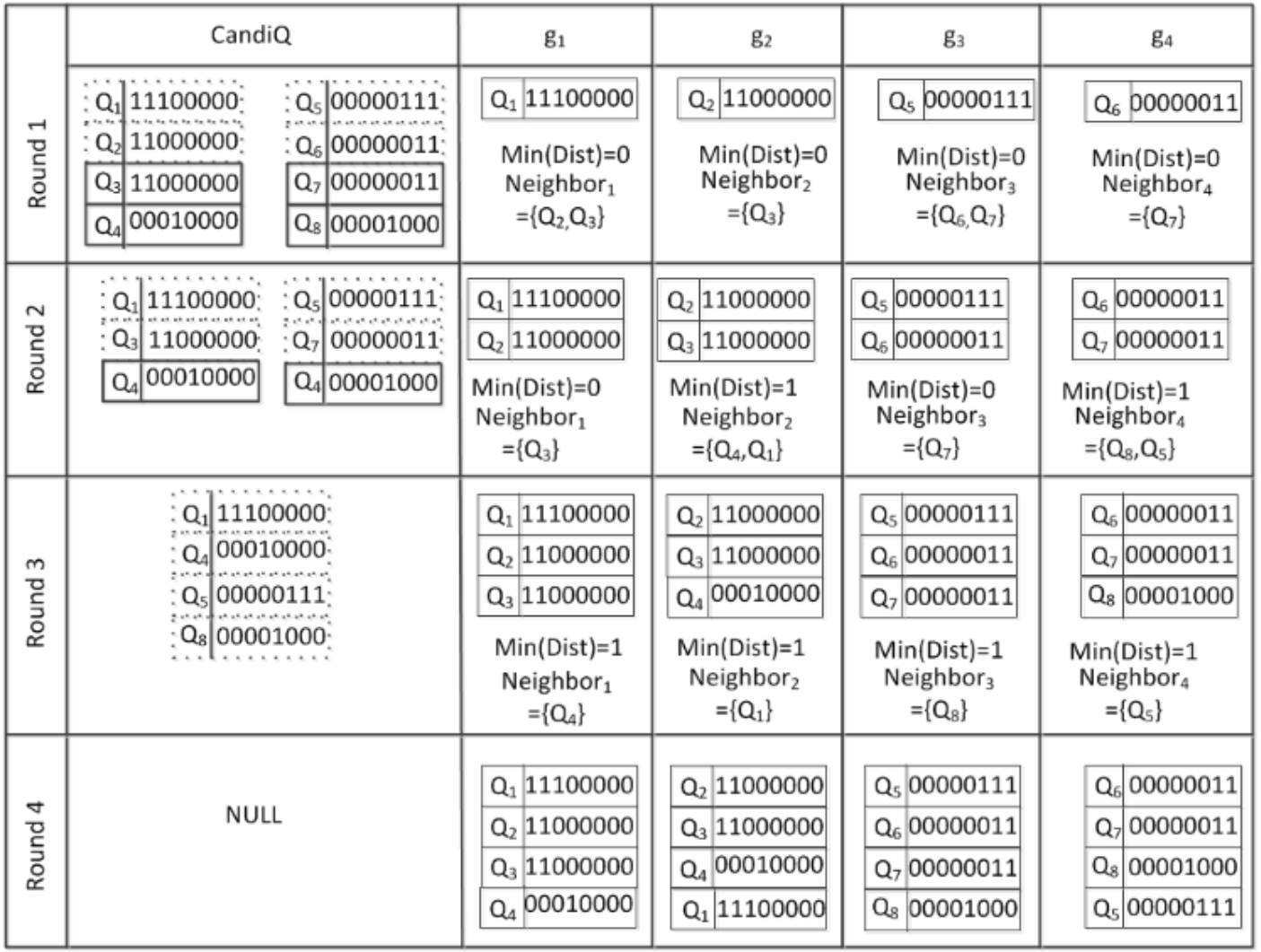
KMDG1-Algorithm

- Each user generates $2 \leqslant \alpha \leqslant k$ query copies with the constraint that α copies are in different

Algorithm 2 KMDG1 (Robust version of KMDG)

1: Construct a set, $CandiQ$, with $2 \leq \alpha \leq k$ query copies
2: Line 2 in Alg. 1
{Runs the following process multiple rounds}
3: **while** $CandiQ$ is not empty **do**
4: **for** $j = 1$ to k **do**
5: $Neighbor_j$ is a subset of $CandiQ$ that accommodate s_j's nearest neighbors
6: Choose a random element $Q_i \in Neighbor_j$ and $Q_i \notin g_j$ into g_j and remove it from $CandiQ$
7: Initialize $CandiQ$ with $2 \leq \alpha \leq k$ query copies
8: Line 8 to line 9 in Alg. 1

KMDG1-Example



KMDG2-Algorithm

- Relax the constraint of equal group size to further reduce bandwidth

Friendliest neighbor. *Given a set of queries in g_j, query $Q_i \notin g_j$, which is the nearest neighbor of s_j and causes the minimal cost after being grouped into g_j, is called the friendliest neighbors of group seed s_j.*

Algorithm 3 KMDG2

1: Line 1 to 2 in Alg. 1
2: **for** $i = 1$ to r **do**
3: **while** $CandiQ$ is not empty **do**
4: **for** $j = 1$ to k **do**
5: $Friend_j$ is a subset of $CandiQ$ that accommodate friendliest neighbors of s_j and $\hat{S}_j$ is the related cost
6: $\hat{S}_j$ is the minimal cost
7: Add a random query in $Friend_j$ to g_j
8: Line 8 to 10 in Alg. 1

KMDG2-Example

	CandiQ	g_1	g_2	g_3	g_4
Round 1	Q_3 11000000 Q_4 00010000 Q_7 00000011 Q_8 00001000	Q_1 11100000 Min(Cost)=3 $Friend_1=\{Q_3\}$	Q_2 11000000 Min(Cost)=2 $Friend_2=\{Q_3\}$	Q_5 00000111 Min(Cost)=3 $Friend_3=\{Q_7\}$	Q_6 00000011 Min(Cost)=2 $Friend_4=\{Q_7\}$
Round 2	Q_4 00010000 Q_7 00000011 Q_8 00001000	Q_1 11100000 Min(Cost)=4 $Friend_1=\{Q_4,Q_8\}$	Q_2 11000000 Q_3 11000000 Min(Cost)=3 $Friend_2=\{Q_4,Q_8\}$	Q_5 00000111 Min(Cost)=3 $Friend_3=\{Q_7\}$	Q_6 00000011 Min(Cost)=2 $Friend_4=\{Q_7\}$
Round 3	Q_4 00010000 Q_8 00001000	Q_1 11100000 Min(Cost)=4 $Friend_1=\{Q_4,Q_8\}$	Q_2 11000000 Q_3 11000000 Min(Cost)=3 $Friend_2=\{Q_4,Q_8\}$	Q_5 00000111 Min(Cost)=4 $Friend_3=\{Q_4,Q_8\}$	Q_6 00000011 Q_7 00000011 Min(Cost)=3 $Friend_4=\{Q_4,Q_8\}$
Round 4	Q_4 00010000	Q_1 11100000 Min(Cost)=4 $Friend_1=\{Q_4\}$	Q_2 11000000 Q_3 11000000 Min(Cost)=3 $Friend_2=\{Q_4\}$	Q_1 11100000 Min(Cost)=4 $Friend_3=\{Q_4\}$	Q_6 00000011 Q_7 00000011 Q_8 00001000 Min(Cost)=4 $Friend_4=\{Q_4\}$
Round 5	NULL	Q_1 11100000	Q_2 11000000 Q_3 11000000 Q_4 00010000	Q_1 11100000	Q_6 00000011 Q_7 00000011 Q_8 00001000

Evaluation

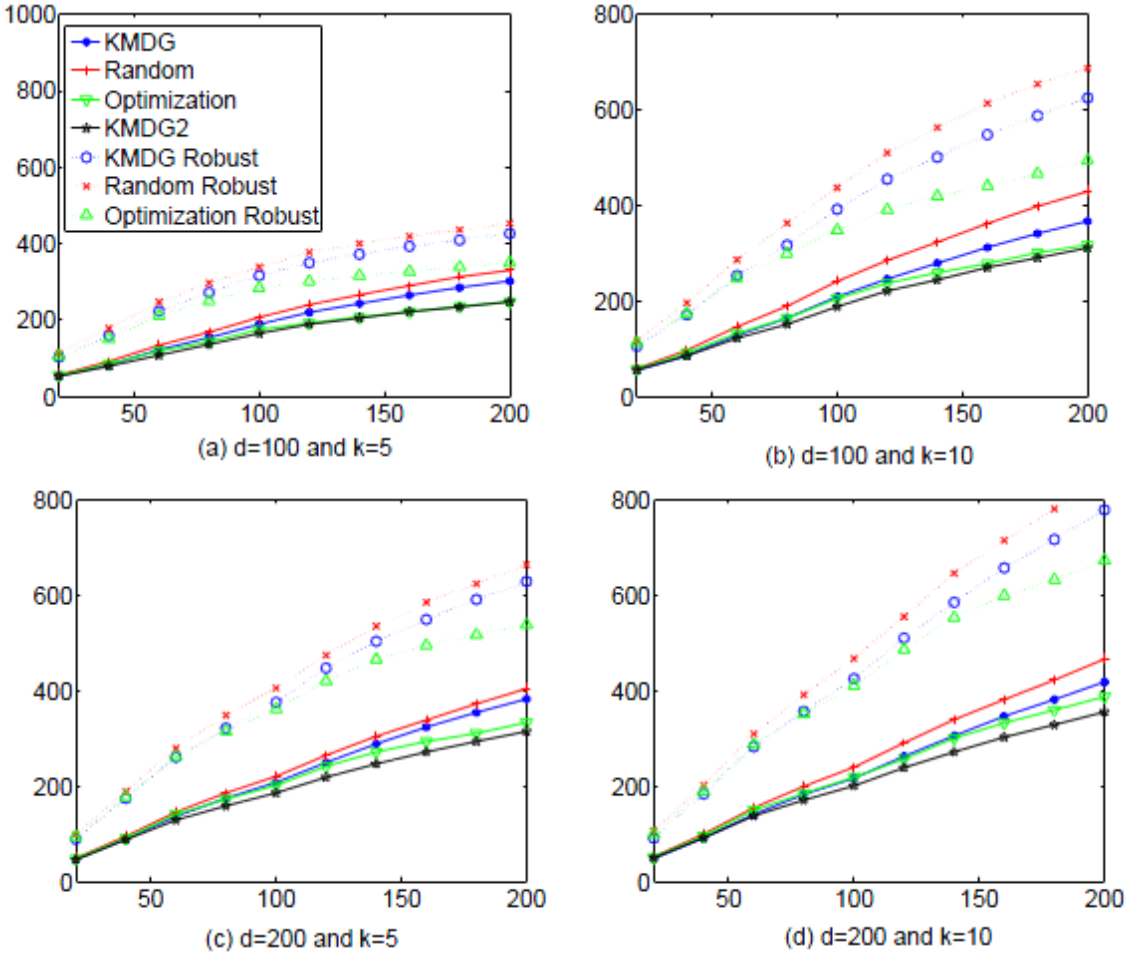
Parameters

- Simulations are conducted with MATLAB R2010a, running on a local machine with an Intel Core 2 Duo E8400 3.0 GHz CPU and 8 GB RAM

Summary of parameters

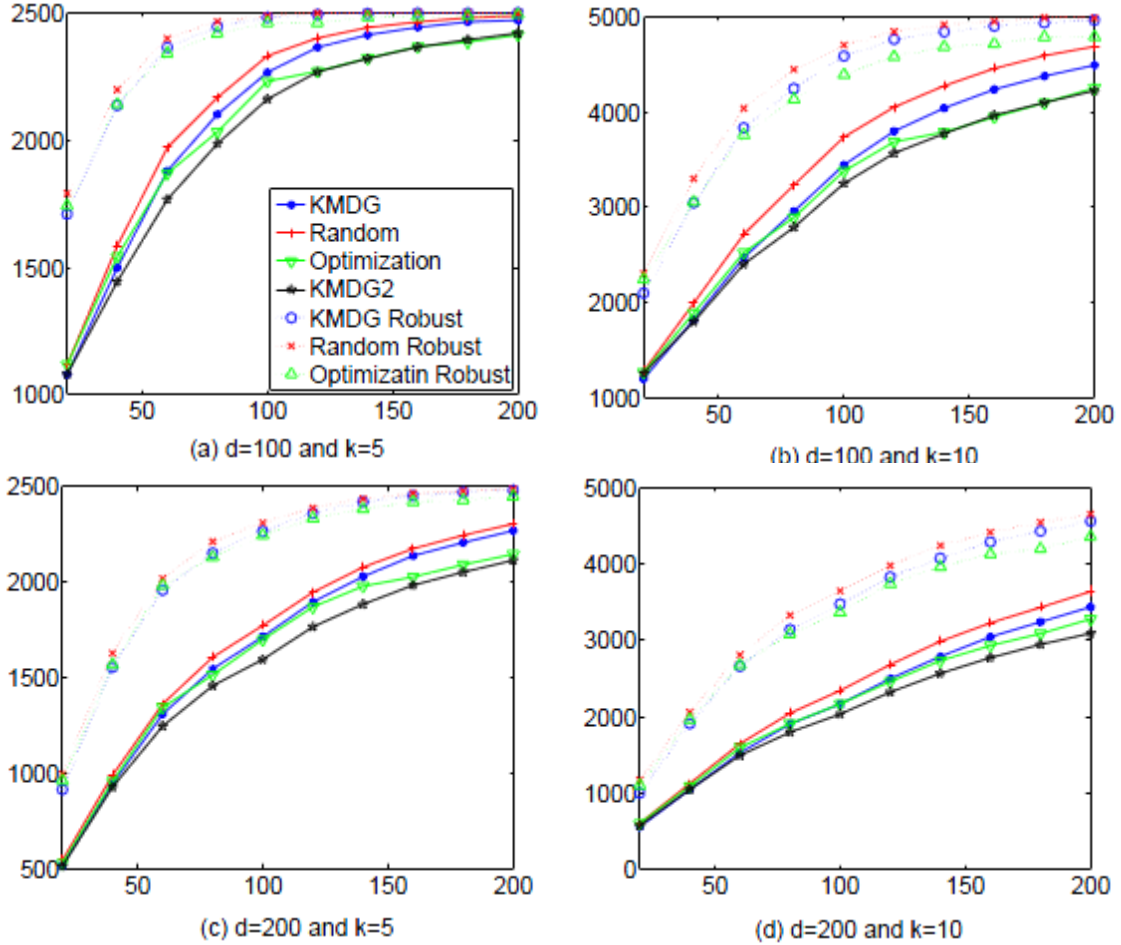
Notation	Description	Value
$\|F\|$	File size	$500\,KB$
n	Number of users in a batch	1-200
d	Number of keywords in the dictionary	100
t	Number of files stored in the cloud	1,000
k	Number of groups	5, 10
r	Number of rounds	500
S_i	Number of 1s/keywords in Q_i	1-5
α	Number of query copies	2
γ	Number of keywords in each file	5

Performance



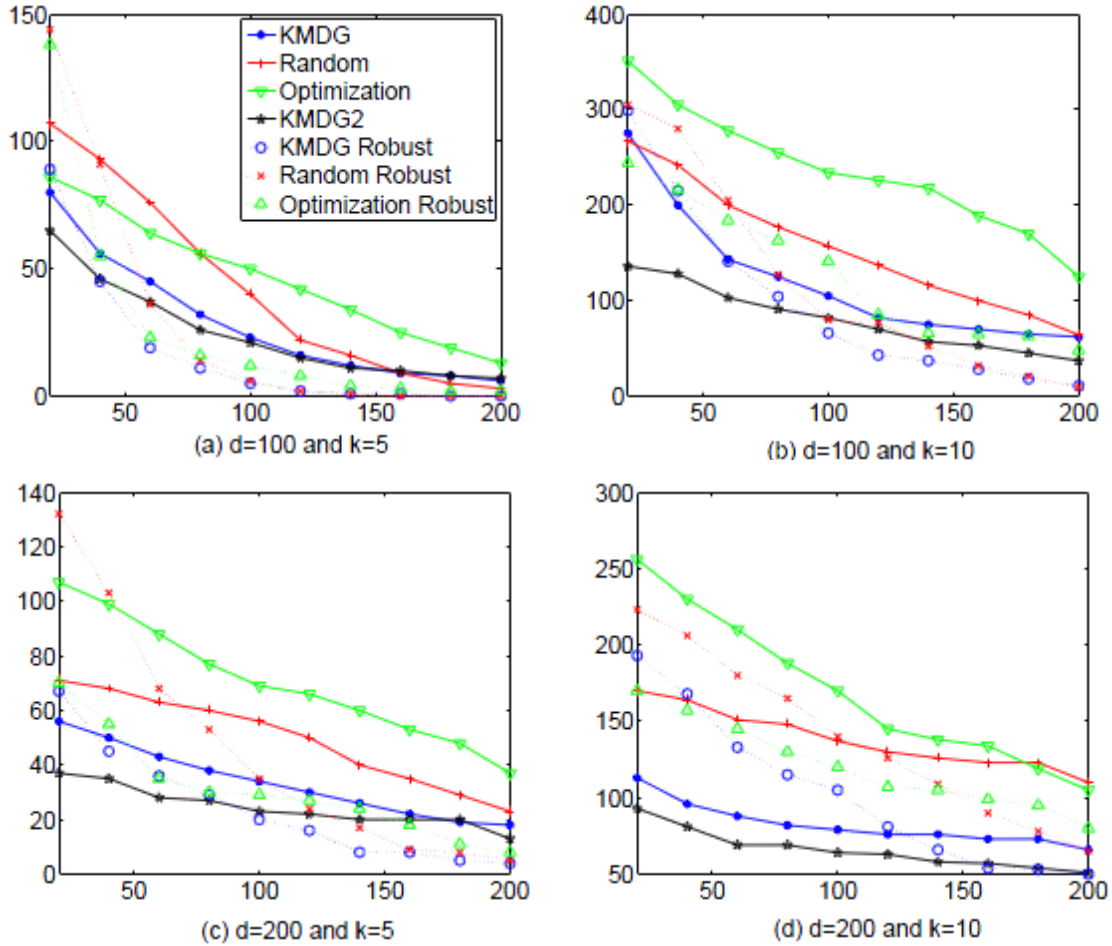
Comparison of total number of 1s. X-axis denotes the number of users and Y-axis denotes the total number of 1s.

Performance



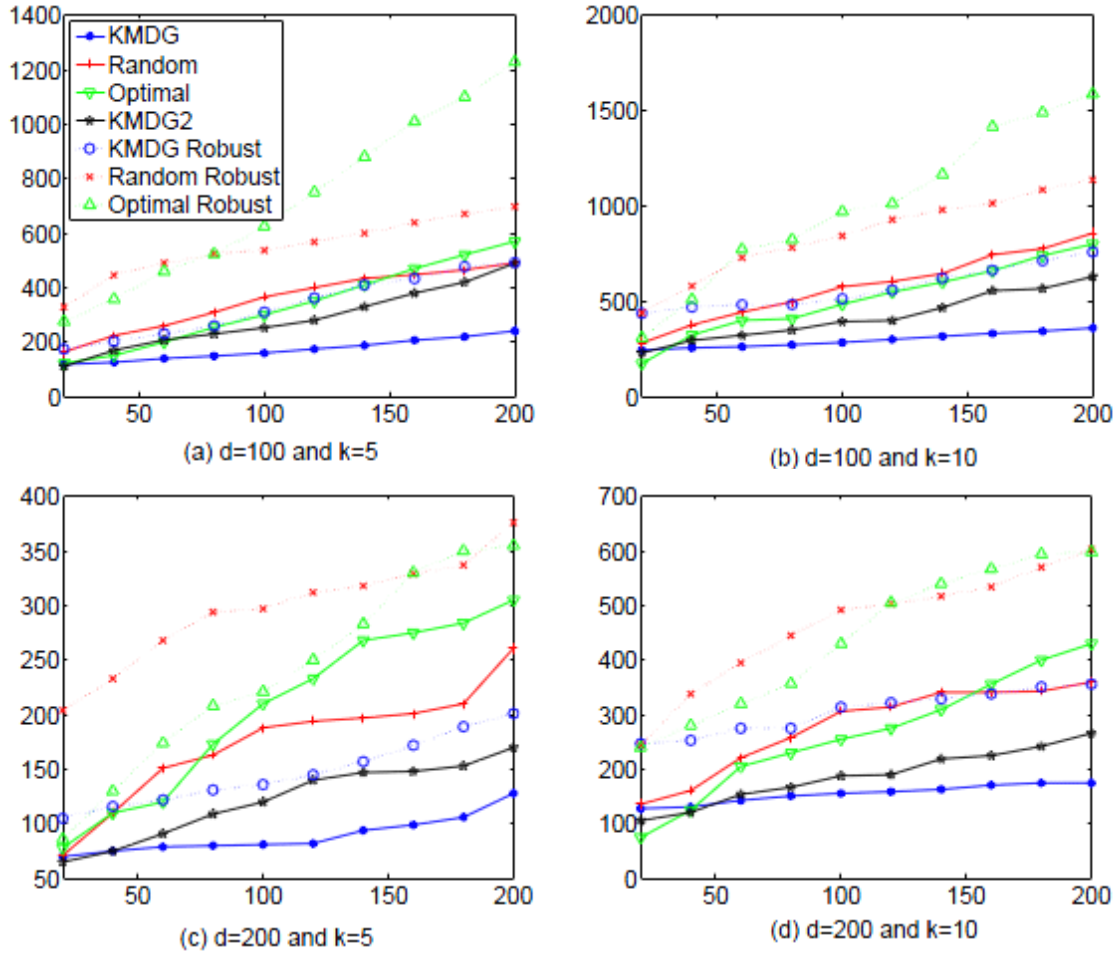
Comparison of bandwidth at the cloud. X-axis denotes the number of users and Y-axis denotes bandwidth at the cloud (MB)

Load balancing



Comparison of imbalanced transfer-in bandwidth. X-axis denotes the number of users and Y-axis denotes the imbalanced transfer-in bandwidth (MB)

Load balancing



Comparison of imbalanced transfer-out bandwidth. X-axis denotes the number of users and Y-axis denotes the imbalanced transfer-out bandwidth (MB)

Conclusion & Future work

1. A dynamic grouping strategy is proposed to achieve cost efficiency, load balancing, and robustness in cloud computing
2. Experiment results show that KMDG can largely reduce the bandwidth incurred at the cloud
3. Conduct experiments on other keyword distributions to verify the effectiveness of KMDG

Thank you!