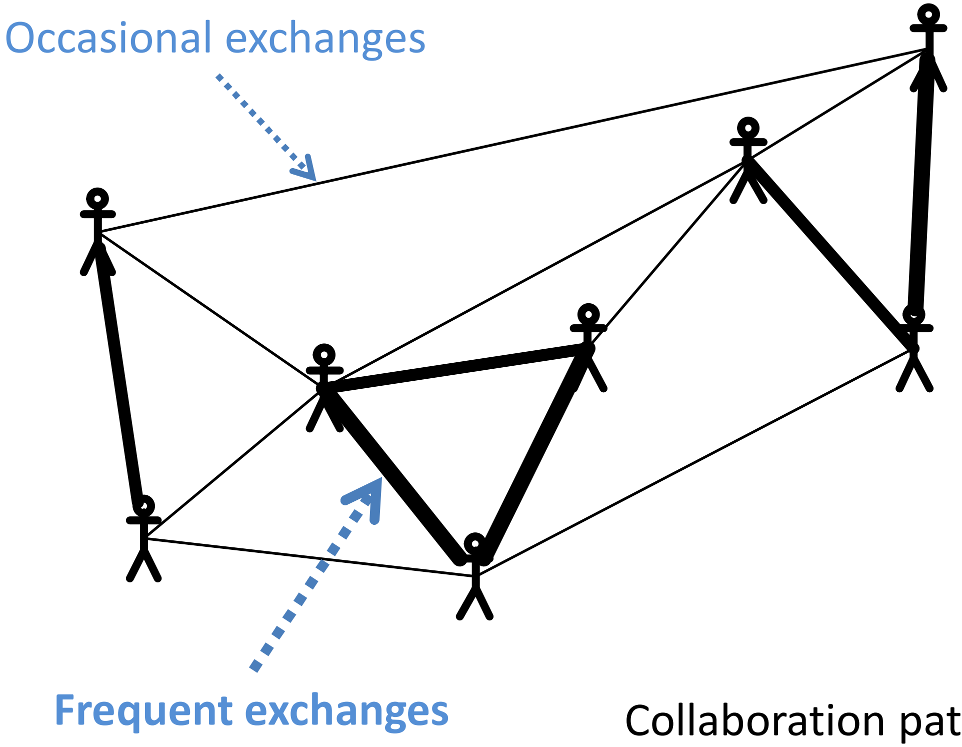
Hermes

Clustering Users in Large-Scale E-mail Services

Thomas Karagiannis, **Christos Gkantsidis**, Dushyanth Narayanan, Antony Rowstron

Microsoft Research
Cambridge, UK

The email social graph



Edges between users that exchange emails

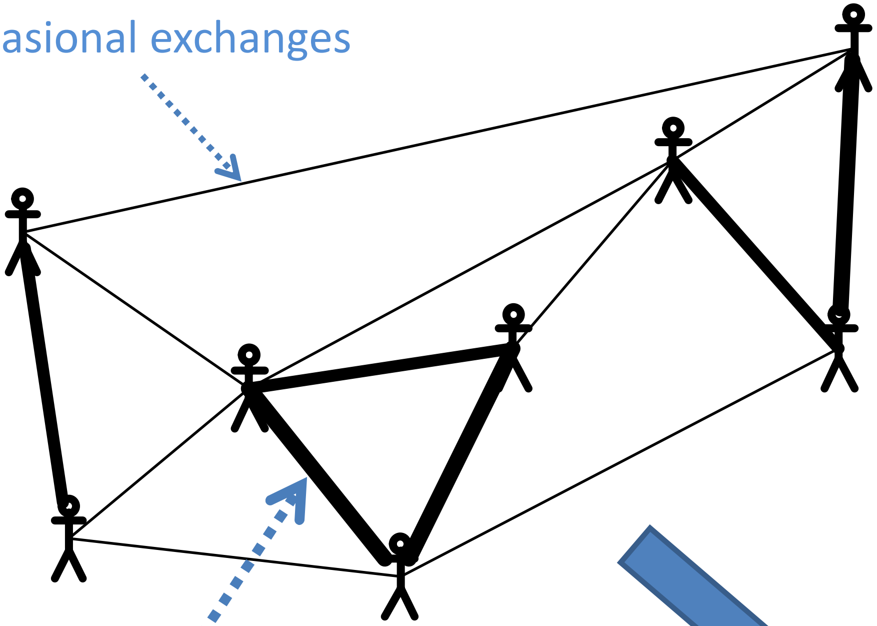
Edges weights capture the frequency of email exchanges, e.g.

- # emails in a week
- total number of bytes / mo
- …

Collaboration pattern:

- **Frequent** email exchanges between **groups of users**

The email social graph



Occasional exchanges

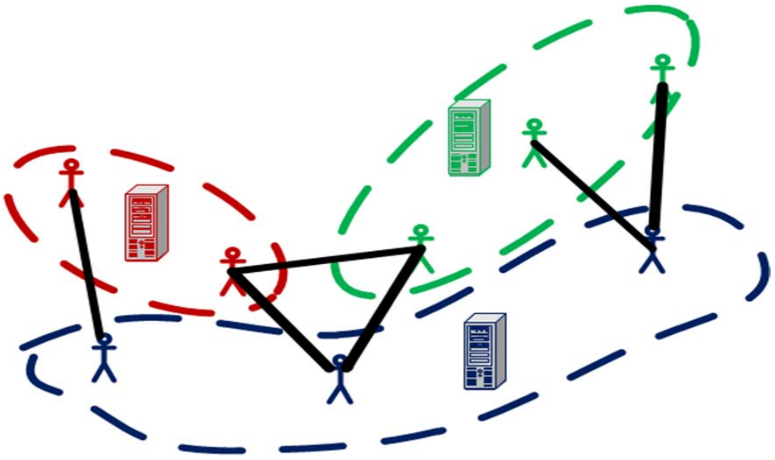
Frequent exchanges

Edges between users that exchange emails

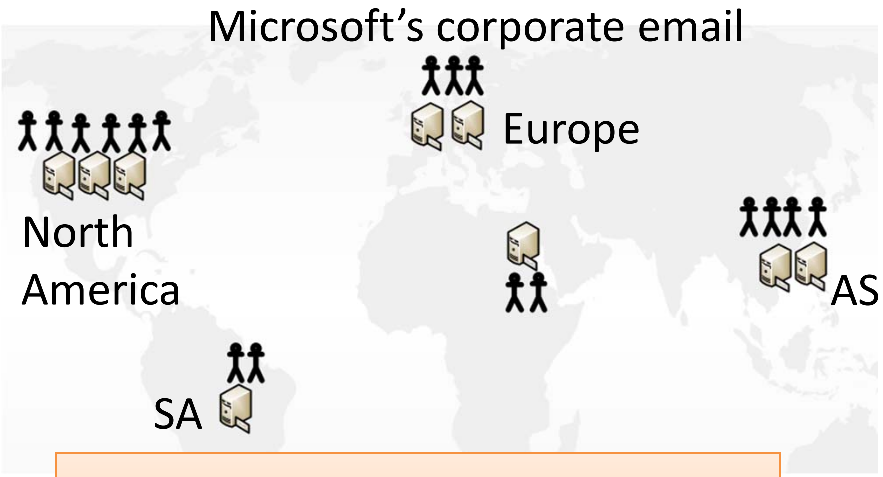
Edges weights capture the frequency of email exchanges, e.g.

- # emails in a week
- total number of bytes / mo
- …

Each user has a home server
User email exchanges
Traffic between servers



System under study



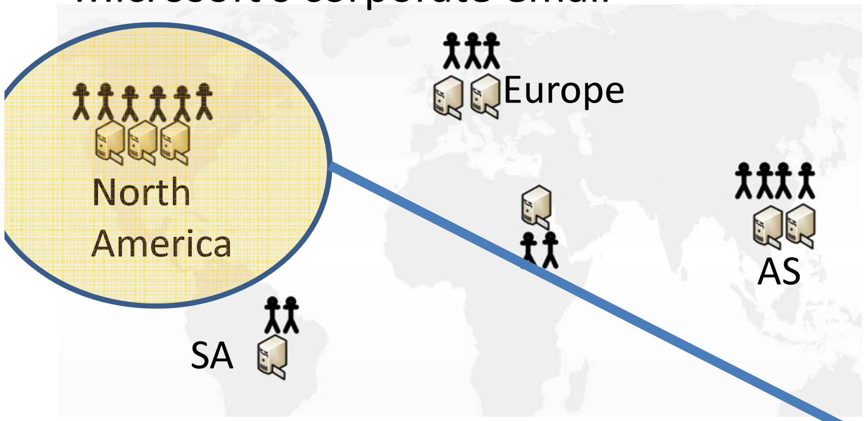
~128K Users
68 exchange servers
⇒ ~1800 users/server

Observed all email activity for 22 weeks (~5 months):

- ~337m emails
- [2.2M emails , 636.3 GB] / day
- **~4 recipients / email + sender**

Current allocation of users to servers

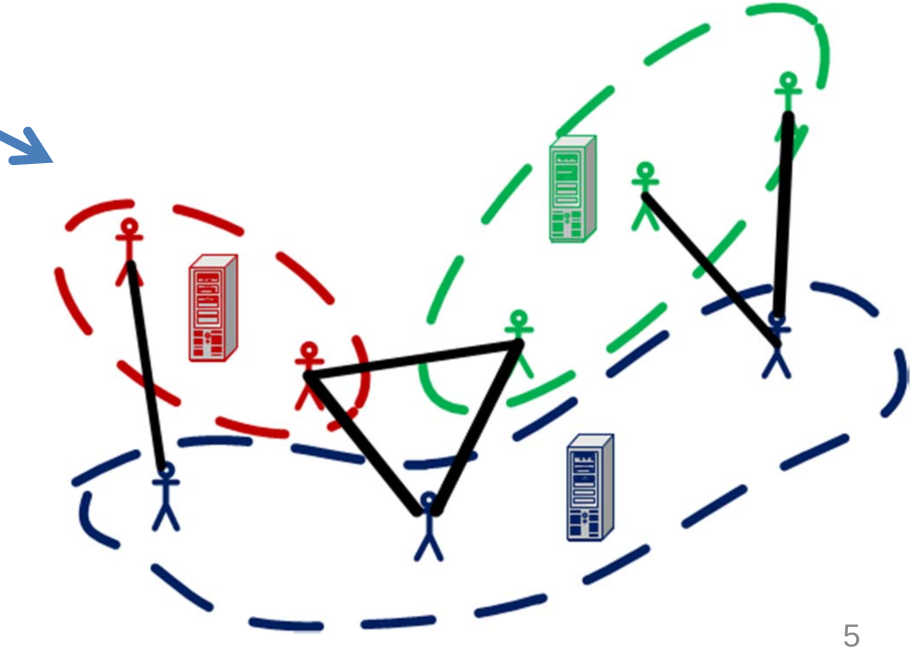
Microsoft’s corporate email



~128K Users,
68 exchange servers
⇒ ~1800 users/server

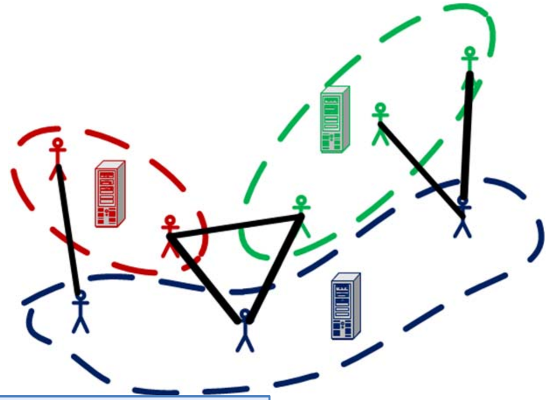
Current user placement:
New users are added to the least loaded server in their region

Hence, agnostic to communication patterns
⇒ **Storage** and network overheads

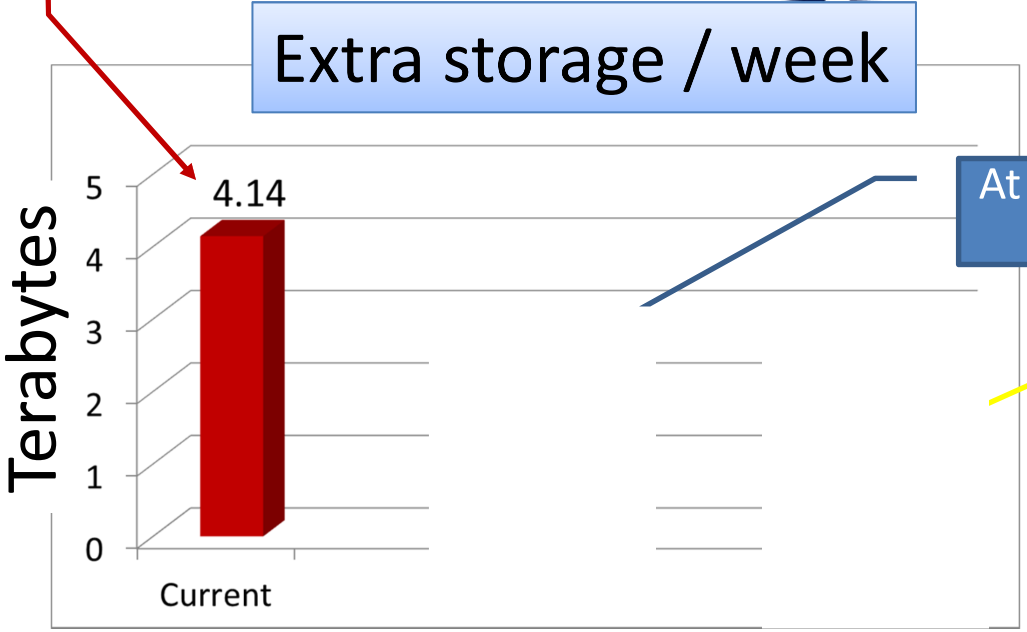


Current allocation of users to servers

Current user placement:
New users are added to the least loaded server in their region



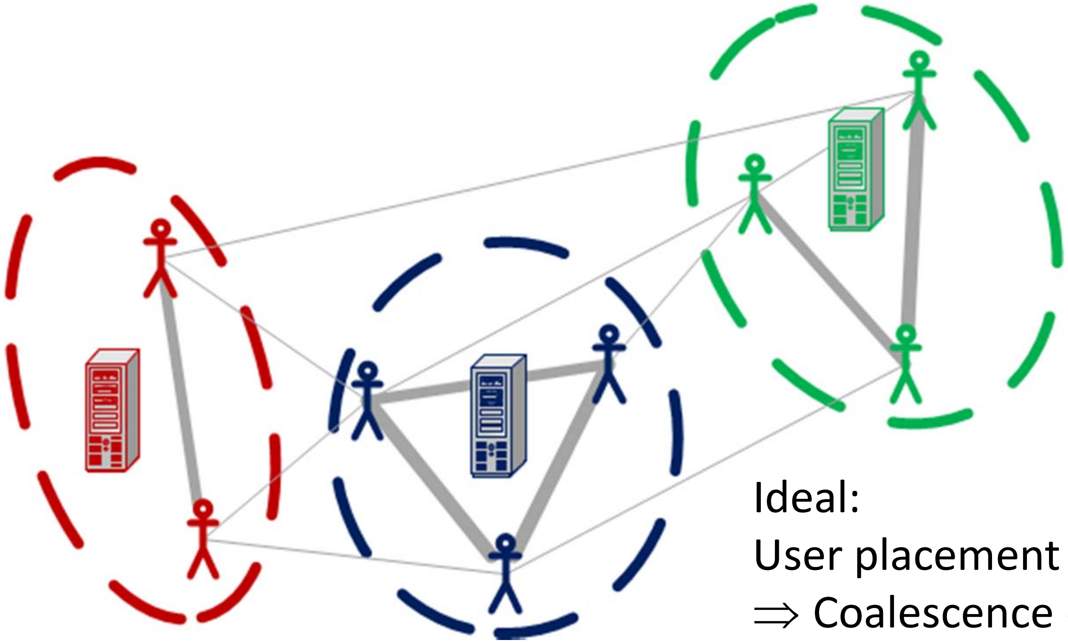
Currently, separate copies per email per user


Extra storage / week

At most one email copy per server

“Optimal”:
Only one copy per email

Better allocation of users to servers



Ideal:

User placement respects communication patterns

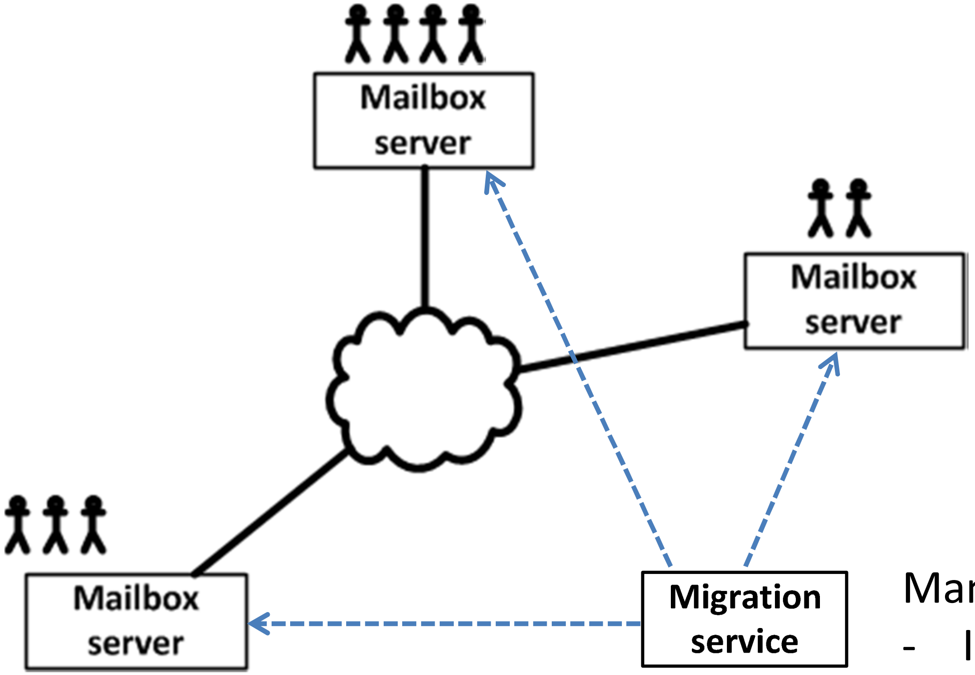
⇒ Coalescence of storage

⇒ Reduces inter-server communication

Goal:

- Detect communication patterns
- Optimize user placement
- Respect current system architecture

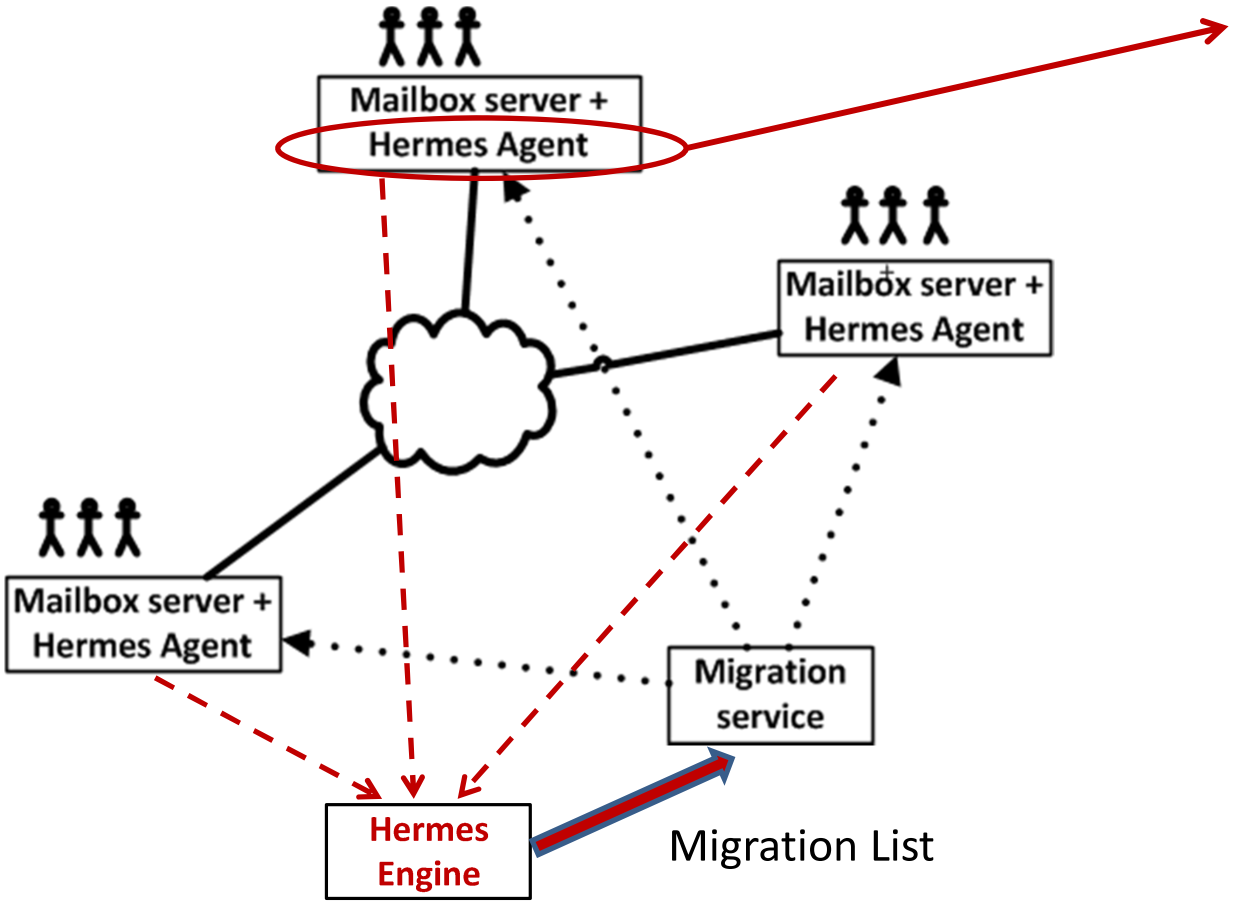
Architecture of email service



Management of infrastructure:

- Inserting / deleting servers
- Moving users

Architecture of email service

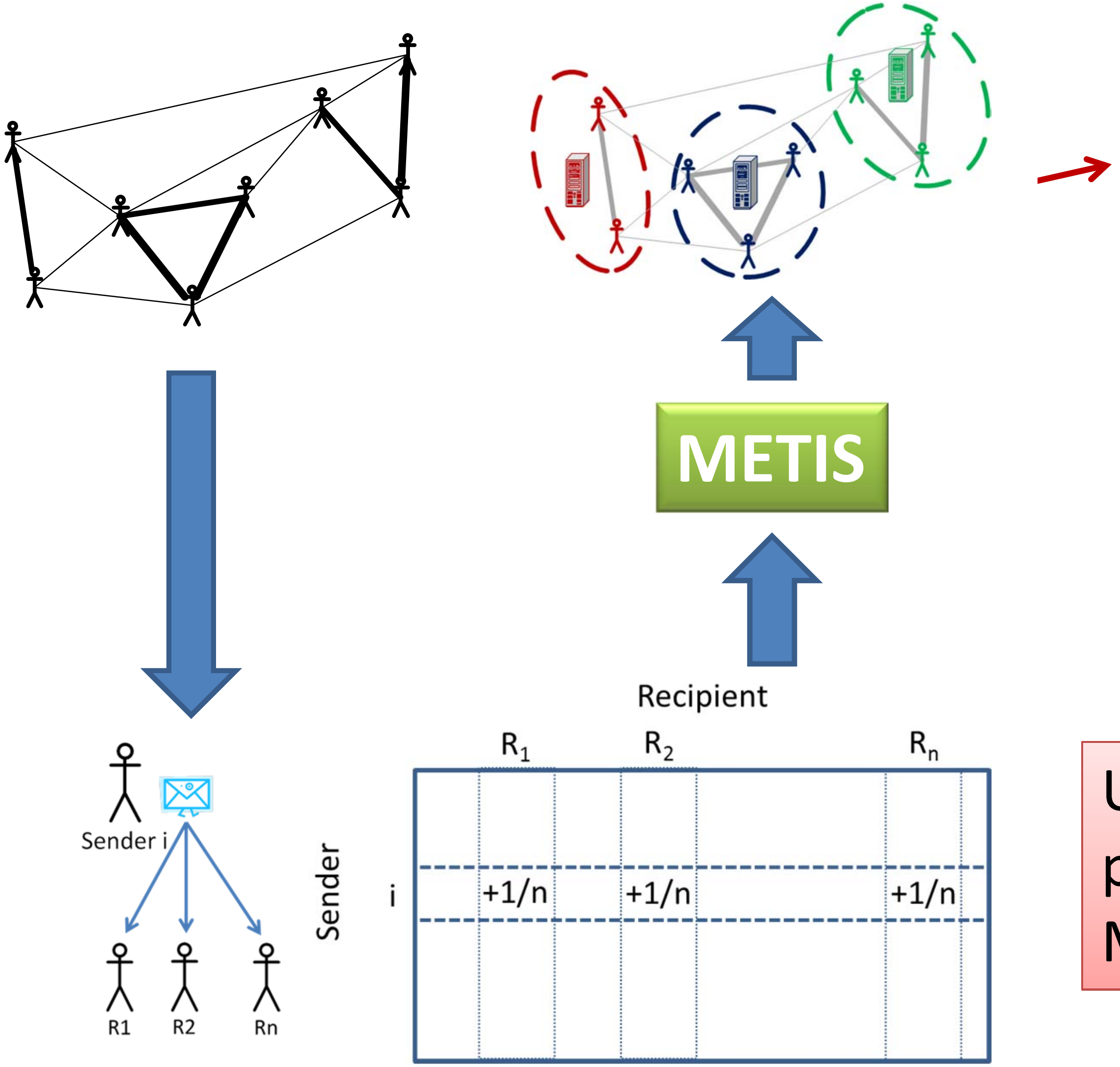


- **Hermes Agent**: Monitors email activity

- **Hermes Engine**: Collects logs and computes "optimal" user placement

Uses standard partitioning algorithms: Metis [Karypis et al.]

Architecture of email service



- **Hermes Agent**:
Monitors email activity

- **Hermes Engine**:
Collects logs and computes optimal user placement

Uses standard partitioning algorithms: Metis [Karypis et al.]

Partitioning

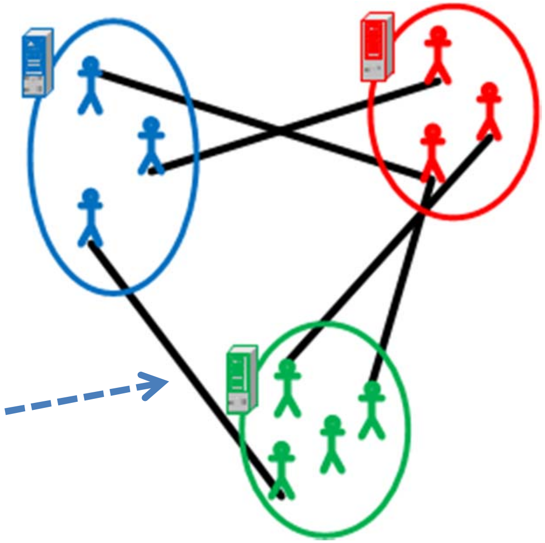
Goal:

- Identify groups of users
- …efficiently

Partitioning

Assign users to partitions s.t.

- min ( )
  for edges with endpoints (i.e. users) on different partitions
- # users per partition is “roughly” balanced



Approach:

Multi-level partitioning

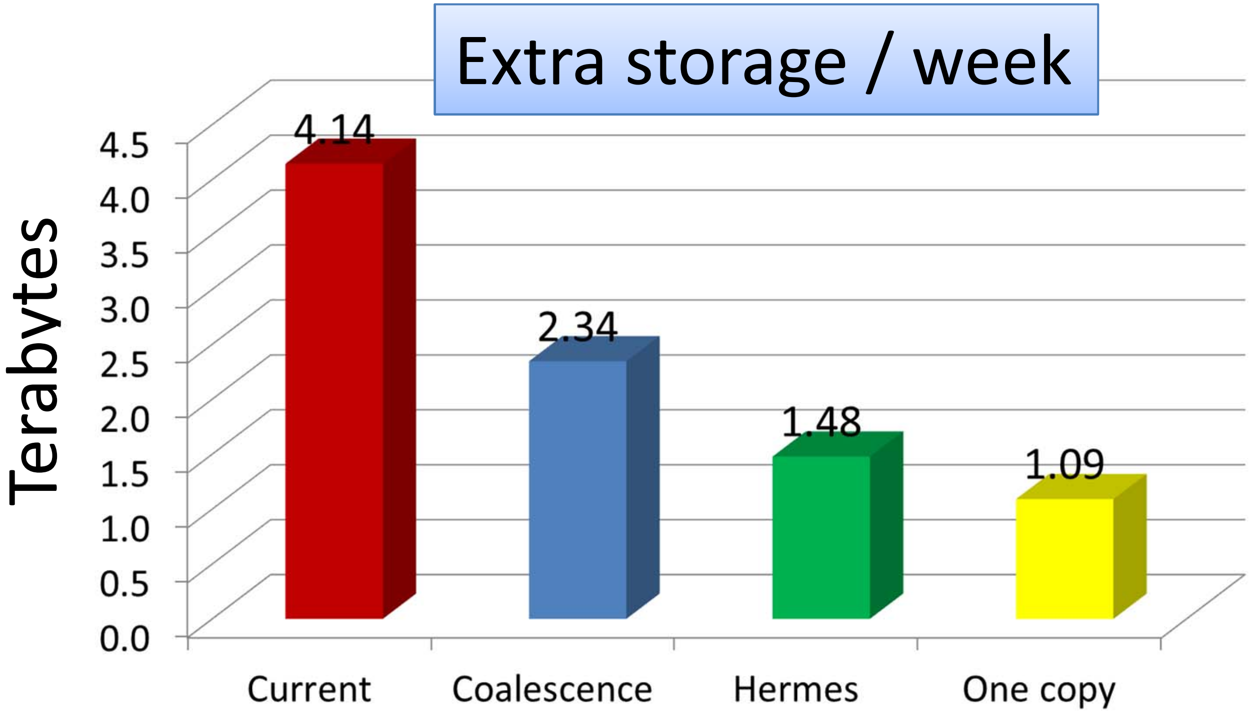
k-Metis & p-Metis

[Karypis et al., ’98]

Evaluation

- Base performance
- Scalability:
  Can it scale to 100's millions of users?
- Capturing changing patterns:
  How often should we re-partition?
- Sensitivity to (# users) / (# servers)
  When should we partition?

Benefits of partitioning



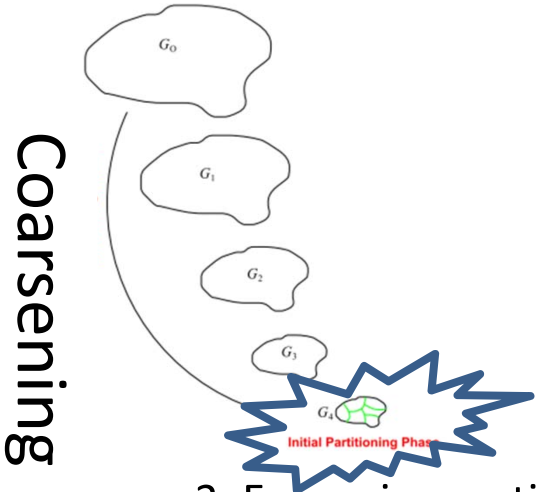
~55 Tbytes of savings in storage (RAID) in 21 weeks

- ✓ 35-40% savings in storage compared to simple coalescence
- ✓ Similar savings in network traffic

Scalability of partitioning

Multilevel partitioning

Source: [Karypis & Kumar, '97]



2. Expensive partitioning step,
but on small graph

1. Coarsening: each step "halves" graph size

3. Un-coarsening: map partitions to original nodes

Metis already efficient (2.66GHz Xeon):

- Available data: 15sec and 250MB in a for 128K nodes and 9-15M edges
- Synthetic model: 10min and 8GB for 4M nodes and 270M edges
  - Memory limited

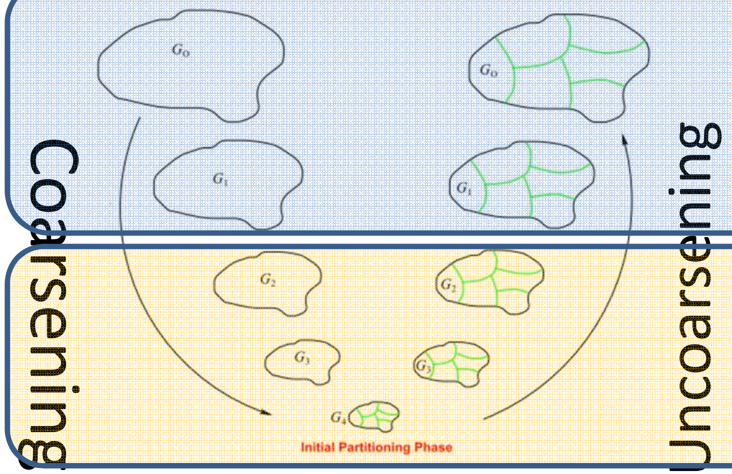
Q) Can we do better?

- Millions of users (e.g. hosted exchange)
- 100's millions (e.g. Hotmail)

Scalability of partitioning

Multilevel partitioning

Source: [Karypis & Kumar, ’97]



Approach: Naïve coarsening and un-coarsening steps to reduce graph size. E.g. 6 naïve steps reduce a graph of size 150M nodes to 2.4M

Run metis as usual on smaller graph

2. Expensive partitioning step, but on small graph

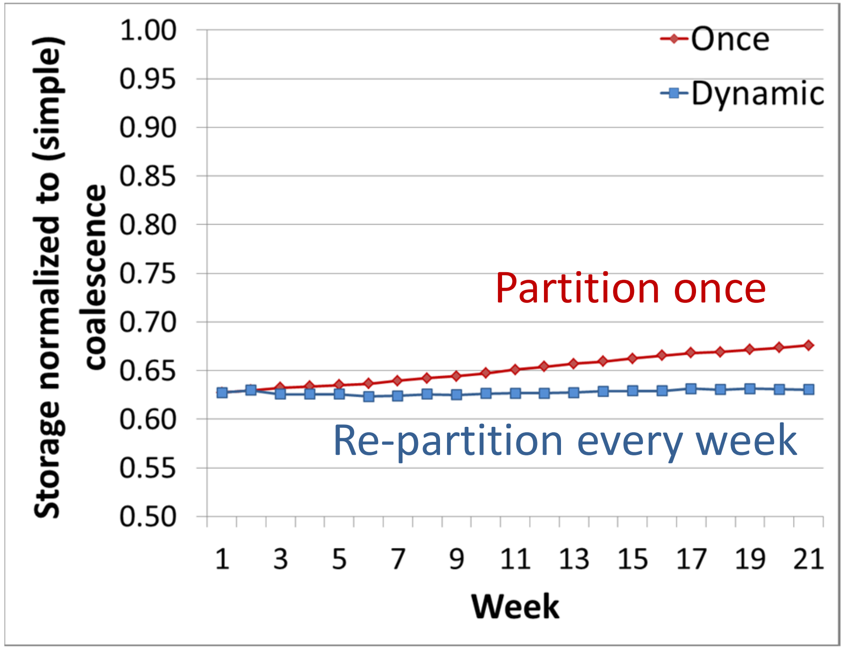
1. Coarsening: each step “halves” graph size

3. Un-coarsening: map partitions to original nodes

Trade-off:
Efficiency of partitioning
(e.g. storage benefit)
reduces by **< 3%**
with **64-fold** reduction in size

How often to re-partition?

- Communication patterns change
- Computing partitions is an efficient background process
- However, moving users (ie mailboxes) around is **expensive**
  - 40-70% of user migrations for each re-partition

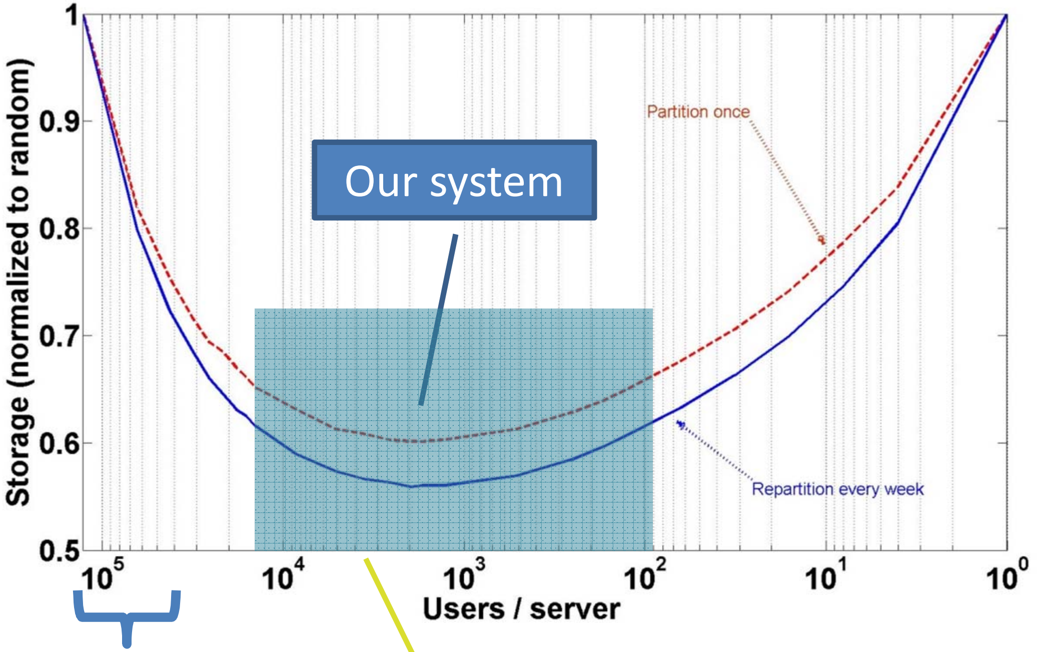


Small loss (<5%) in storage benefits for infrequent re-partitions (eg every few months)

Sensitivity to #users / server

All users in one server

1 user / server



Our system

Very few servers

>200 users per server gives good benefit

Some other observations

- Geography
  - – Easy to incorporate geographical constraints
  - – … very similar results
- Flexibility in setting the optimization goal
  - – This work: minimize storage and net
  - – Can also use I/O load
- Sampling of messages
  - – This work: collected & used all messages
  - – Also, similar results when ignoring emails with large # recipients
  - – Clever sampling techniques?

Related Work

- Spar [*Pujol et al, SigComm 2010*]
  - – Partitioning for online social networks
  - – Evaluation: Twitter, Facebook, and Orkut traces
  - – Algorithm: Modularity Optimization (MO+)

- Volley [*Agarwal etl al, NSDI 2010*]
  - – Data-Placement for Geo-Distributed Cloud Services
  - – Evaluation: Live Mesh and Live Messenger traces
  - – Algorithm: Use geo-information to place users & data, iteratively improve placement

Summary

- Goal: Explore social (graph) patterns to improve online services
  - Hermes: Optimize user placement based on email exchanges
  - 35-50% storage and network savings

- Partitioning has low overhead:
  - No need to do frequent repartitions