TripS: Automated Multi-tiered Data Placement in a Geo-distributed Cloud Environment

Kwangsung Oh, Abhishek Chandra, and Jon Weissman

Department of Computer Science and Engineering

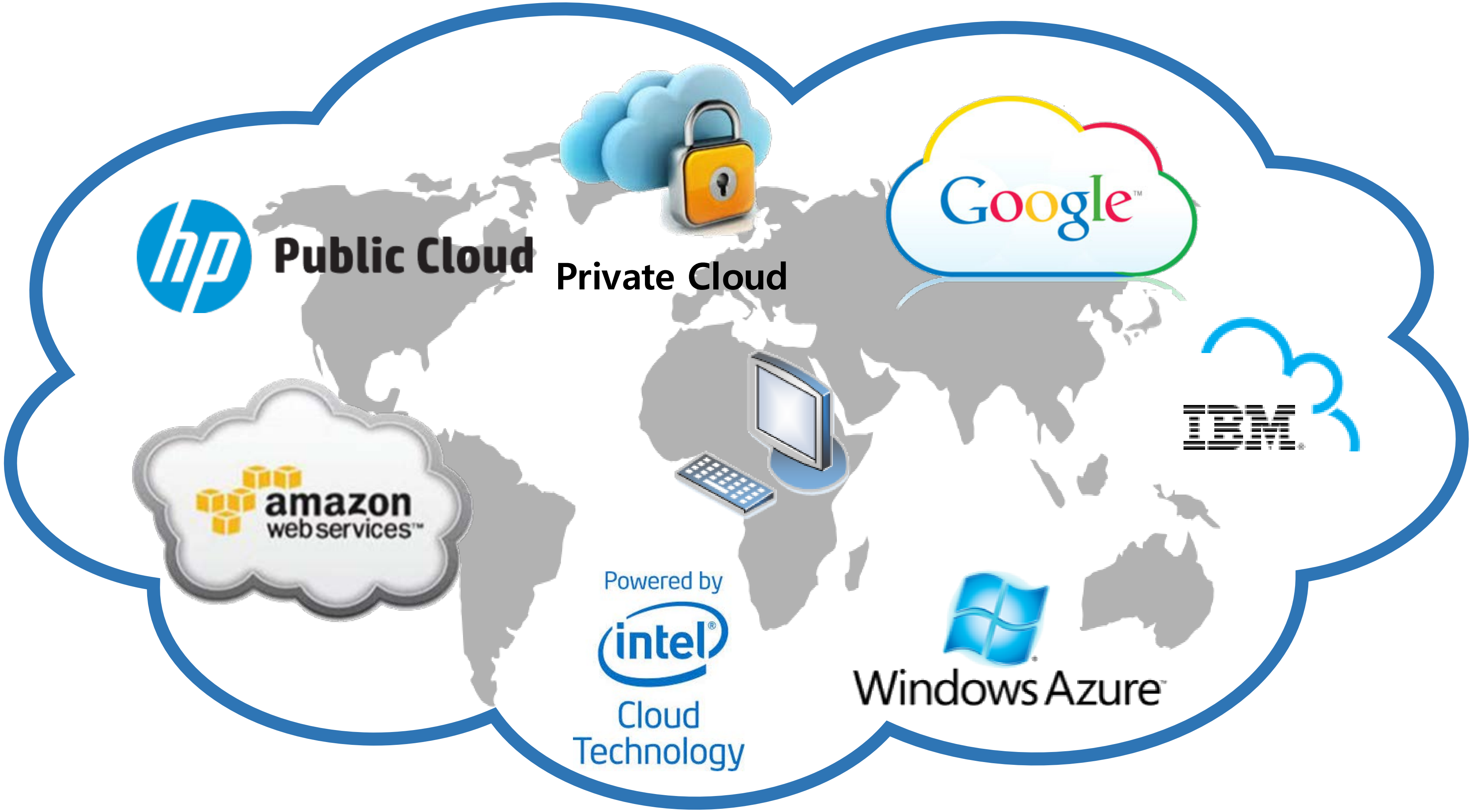
University of Minnesota Twin Cities

Systor 2017

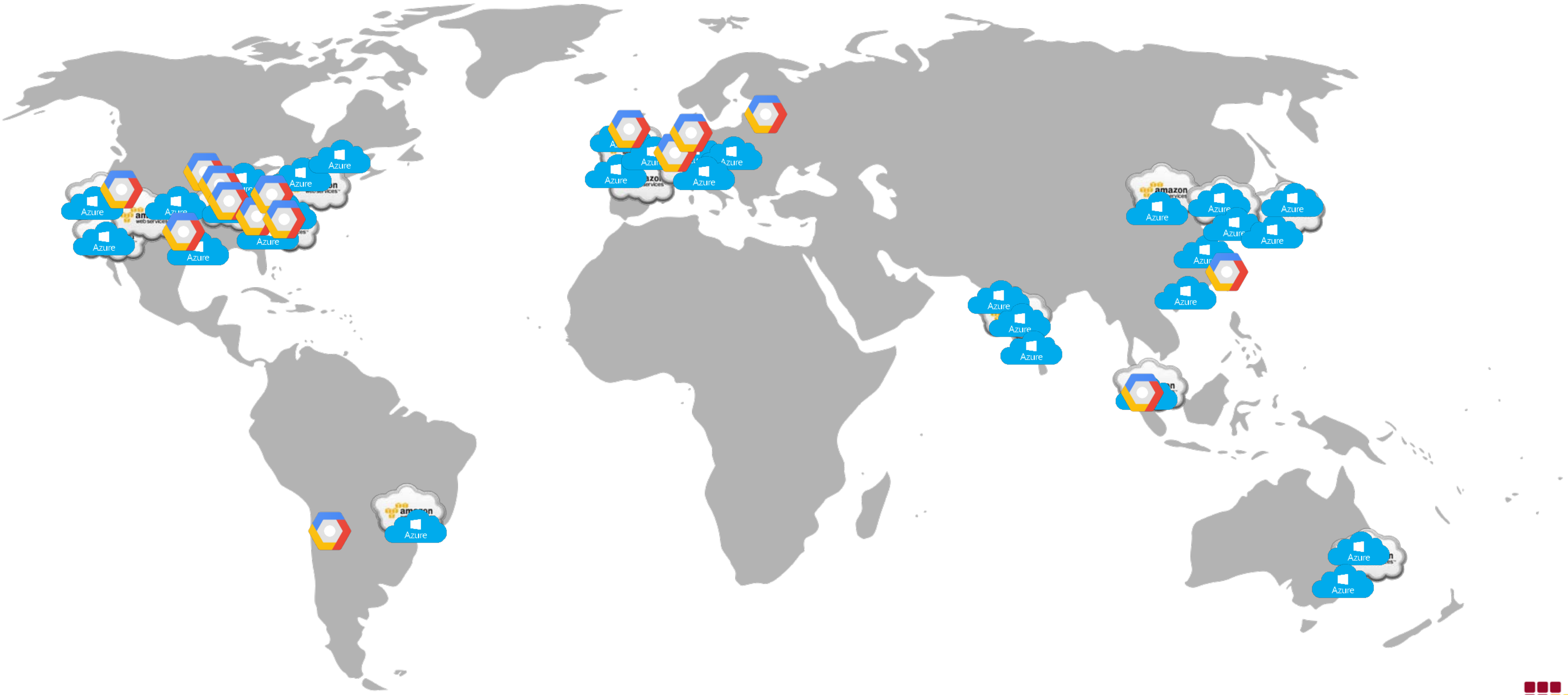
UNIVERSITY OF MINNESOTA
Driven to Discover℠

DCSG
Distributed Computing Systems Group

Cloud Providers Publicly Available

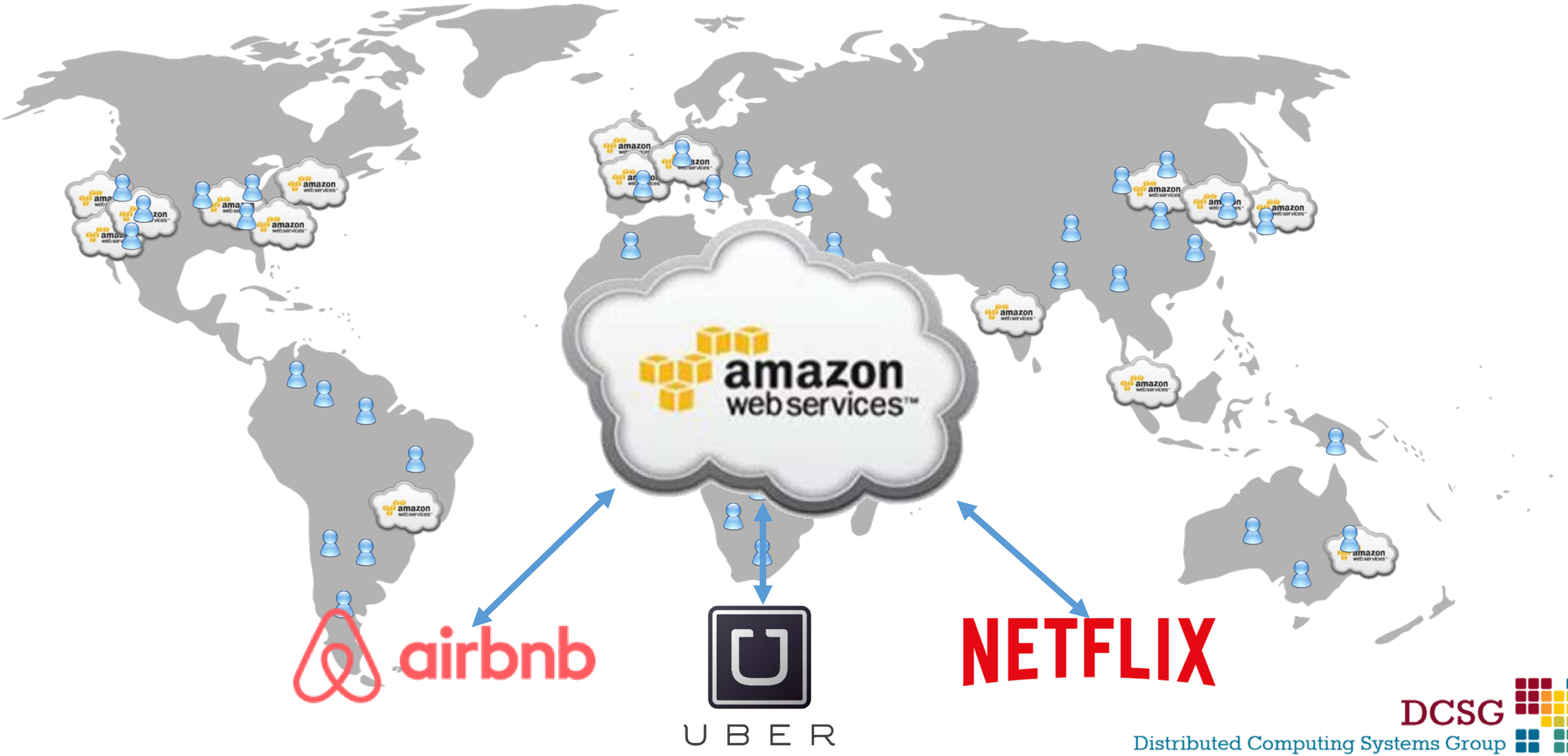


Multiple Data Centers



DCSG
Distributed Computing Systems Group

Users are around the Globe

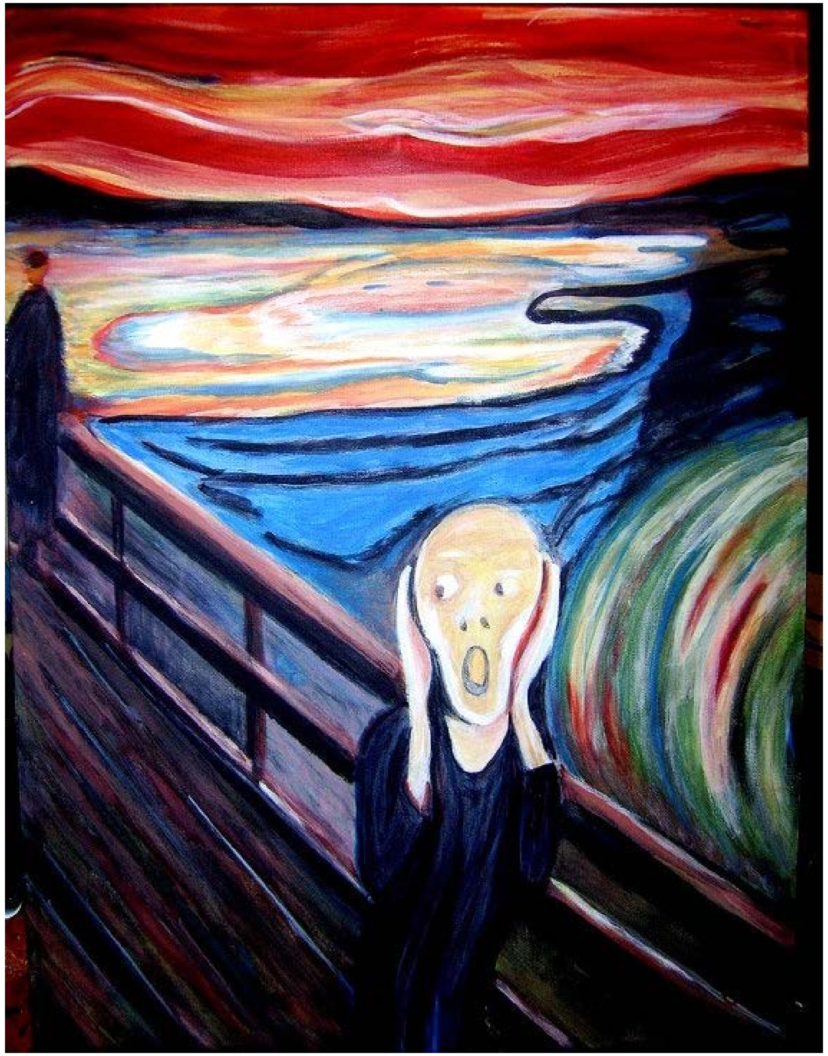


Geo-Distributed Users, DCs and Applications

Where are the ***best locations***
for storing data?

Different Applications’ goals

- SLA
- Consistency Model
- Desired Cost
- Desired Fault Tolerance
- Data Access Pattern
- Users’ Locations
- And many more…

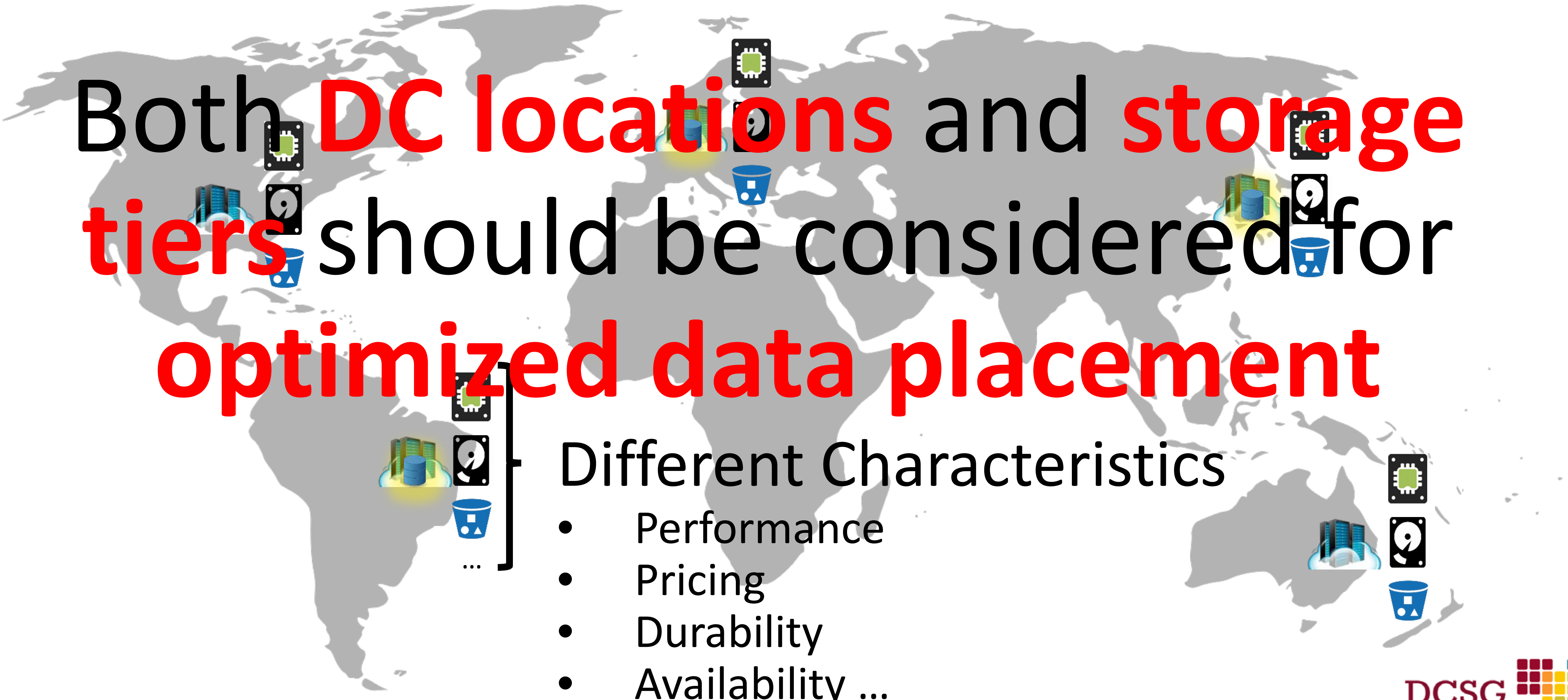


Previous Data Placement Systems

- Volley [Agarwal et al, NSDI ’10]
- Spanner [Dean et al, OSDI ’12]
- SPANStore [Wu et al, SOSP ’13]
- Tuba [Ardekani et al, OSDI ’14]
- Focusing on **data center locations**

Multiple Storage Tiers Available

Both **DC locations** and **storage tiers** should be considered for **optimized data placement**



Different Characteristics

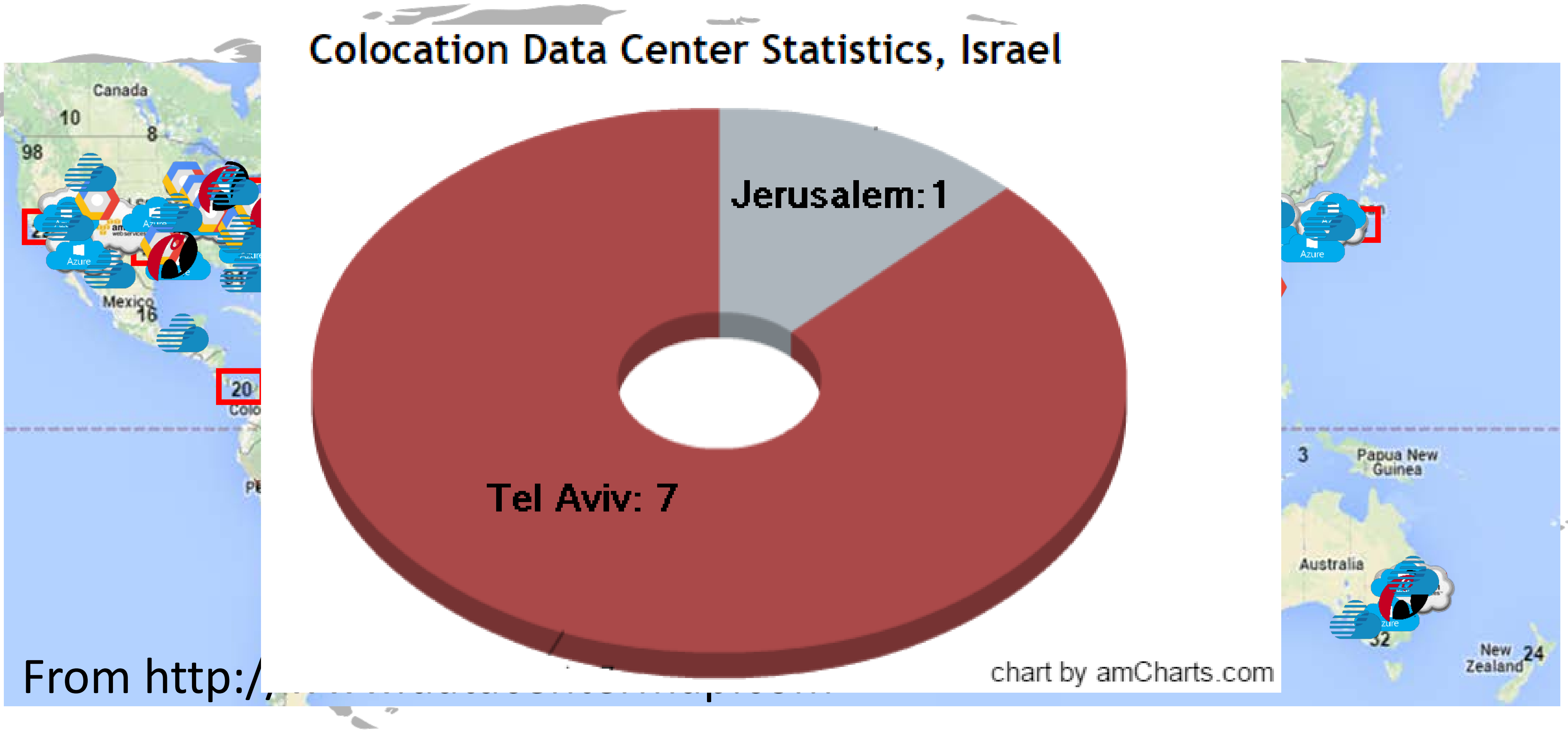
- Performance
- Pricing
- Durability
- Availability …

Challenges

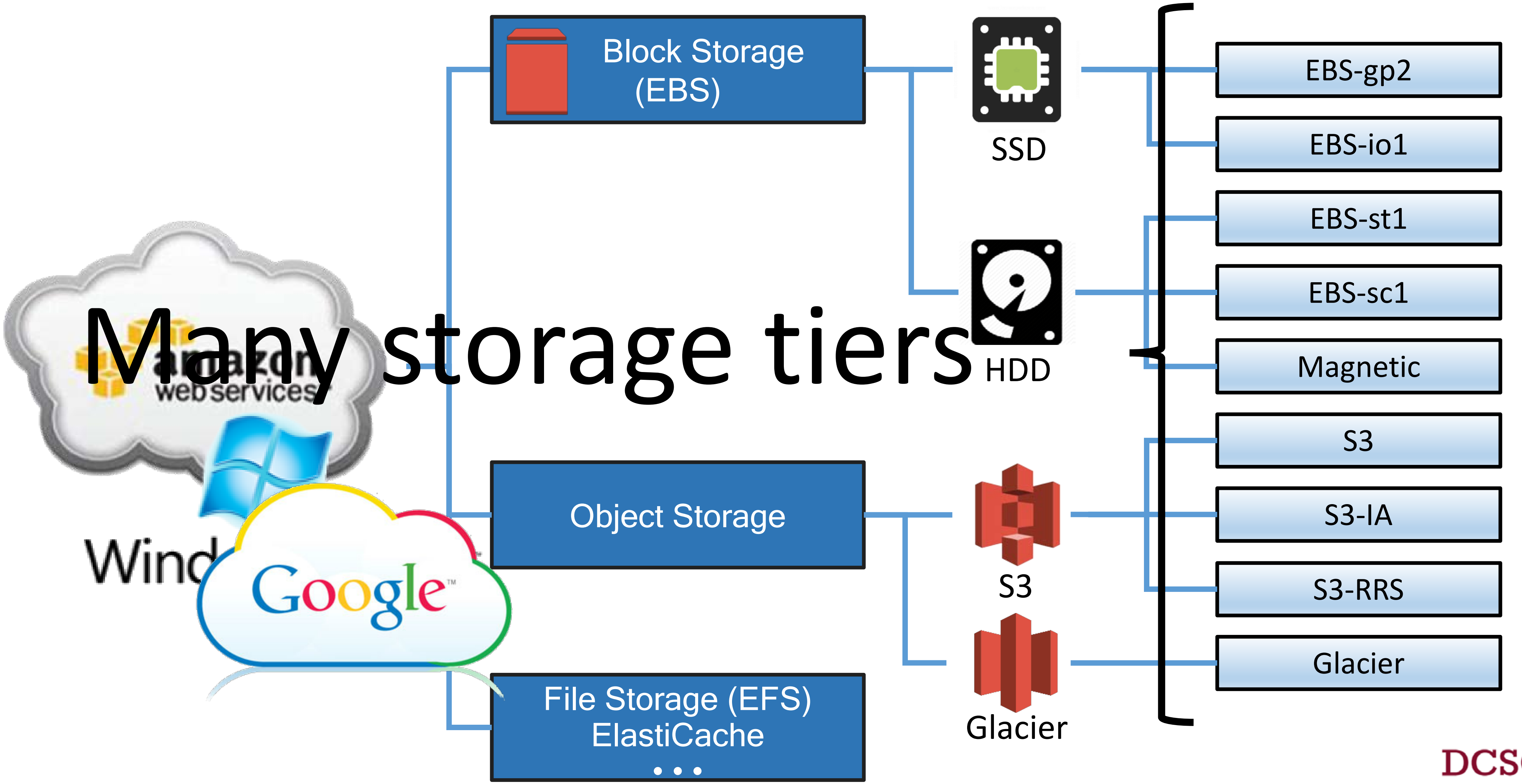
- **Many options** for data center locations and storage tiers

- **Dynamics** from cloud environment

Data Center Locations Options



Storage Services Options



Many storage tiers

Challenges

✓ **Many options** for data center locations and storage tiers

- **Dynamics** from cloud environment

Dynamics from

- Infrastructure
  - Cloud service providers do **not guarantee consistent performance**
  - E.g., transient DCs (or network) failure, burst access pattern, overloaded node and so on
- Applications
  - User locations and access patterns **keep changing**
  - E.g., users are travelling world widely, changes in data popularity

Goal

- **Finding optimized data placement**
  - Exploiting both **DC locations** and **multiple storage tiers**
  - Helping applications handle **dynamics**

Roadmap

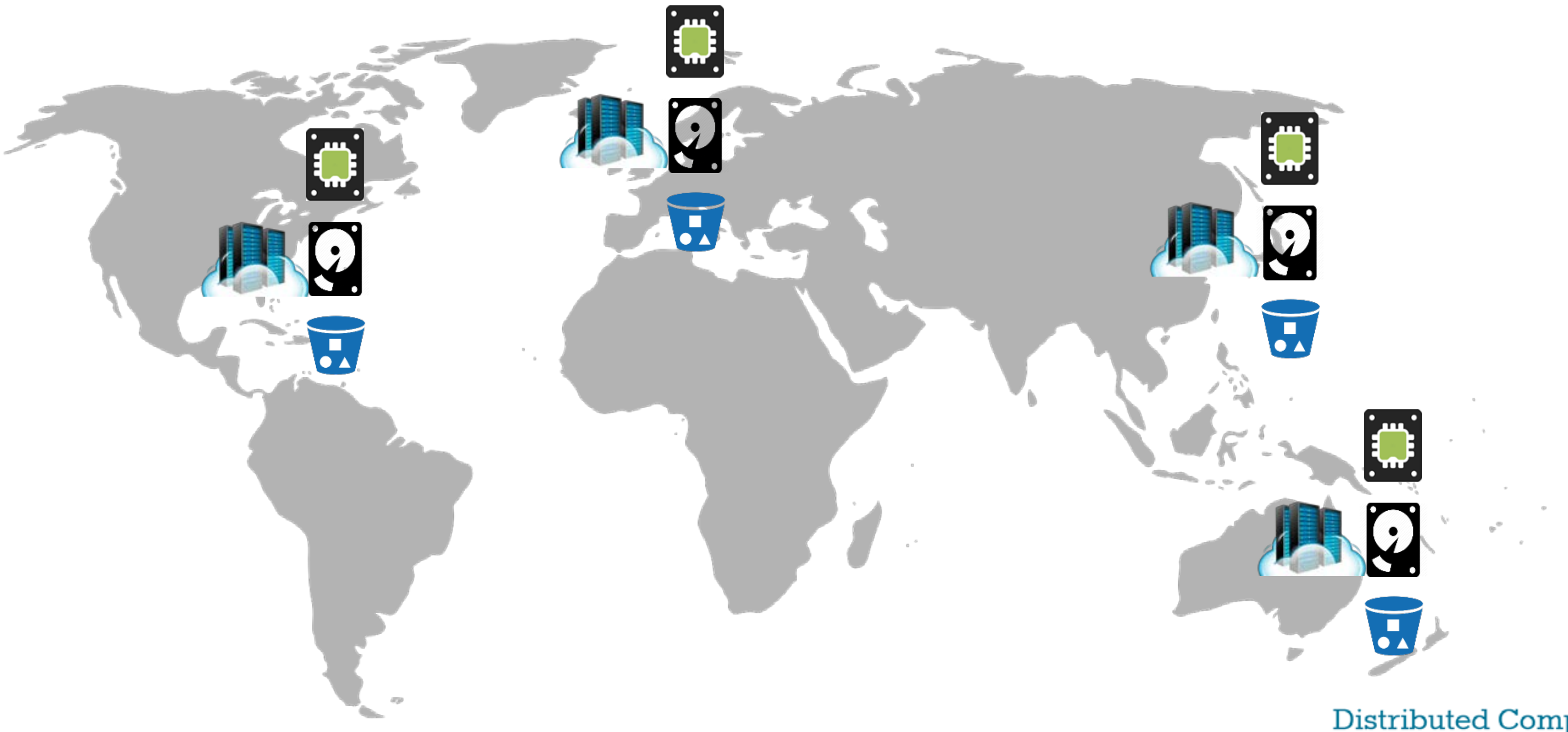
- ✓ Motivations & Goals
- **TripS (Storage Switch System)**
- Handling dynamics
- Experimental Evaluations

TripS

- Light-weight **data placements decision system**; considering both **DC locations** and **storage tiers**
- Helping applications to **handle dynamics**

System Model

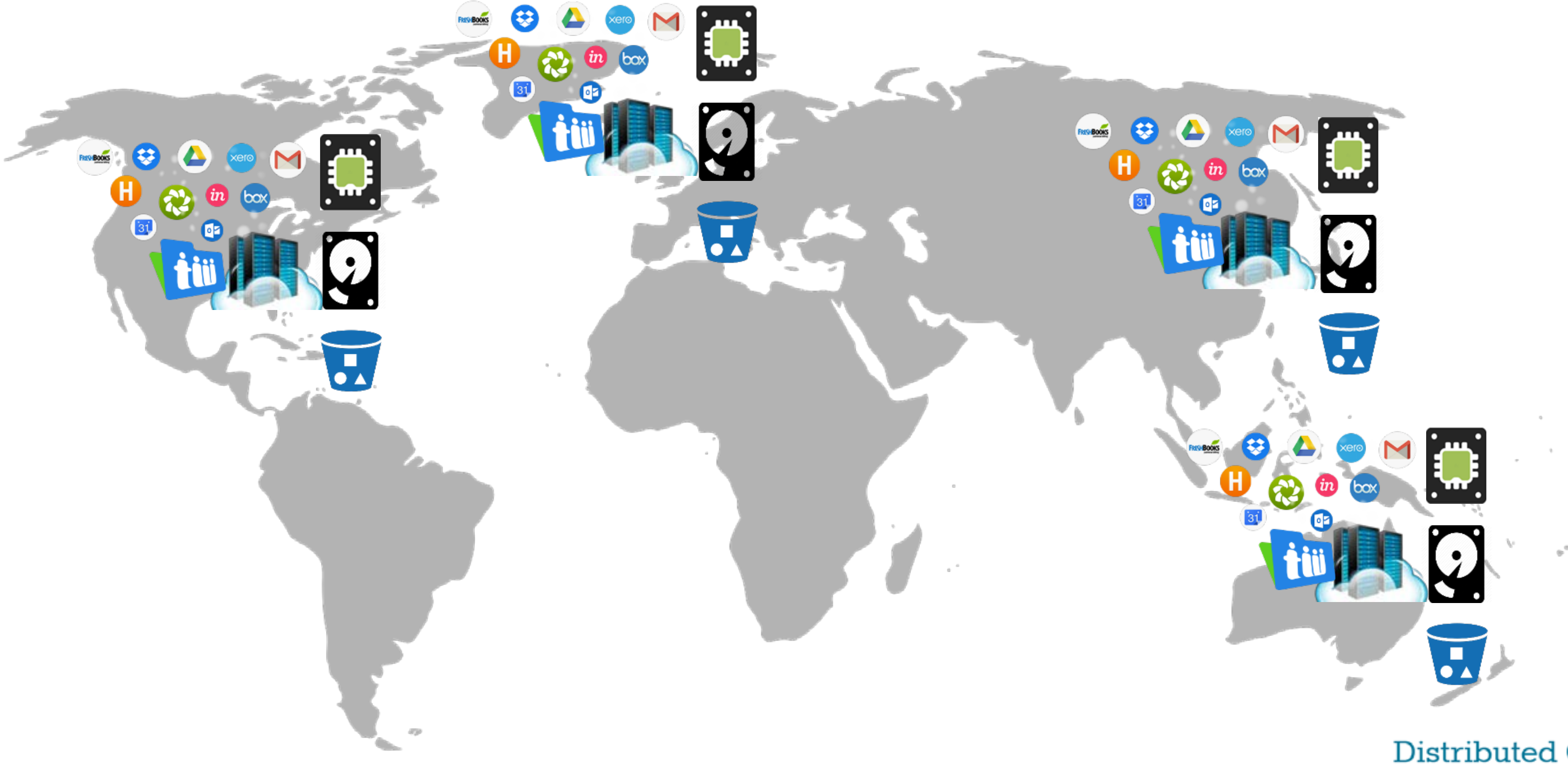
- Geo-distributed storage system (GDSS)
  - Running on multiple DCs (across different cloud providers)
  - Exploiting multiple storage tiers



DCSG
Distributed Computing Systems Group

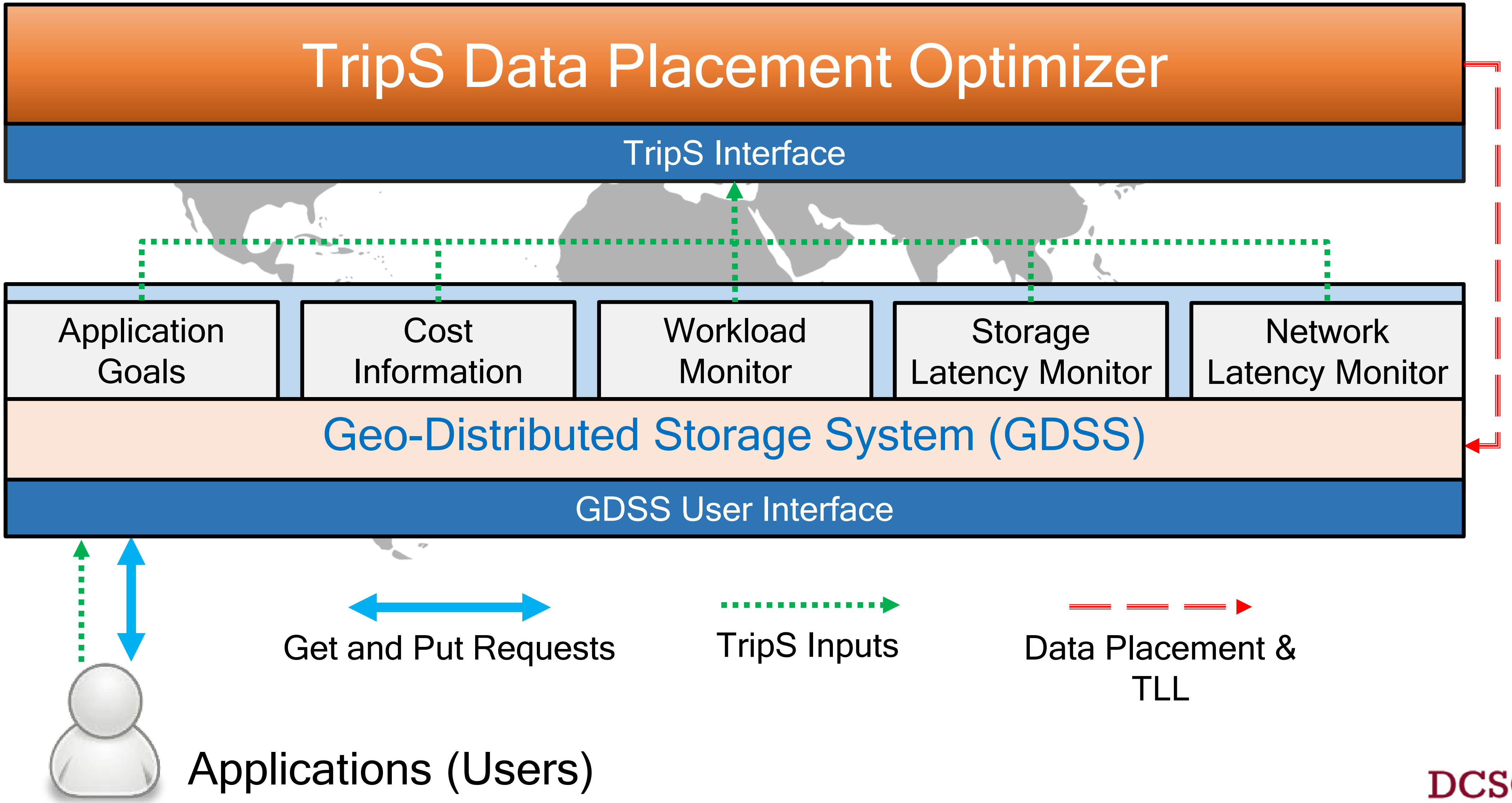
System Model

- Applications are running on GDSS
  - Connecting any GDSS server (possibly the closest server)
  - Using Get/Put API exposed by GDSS



DCSG
Distributed Computing Systems Group

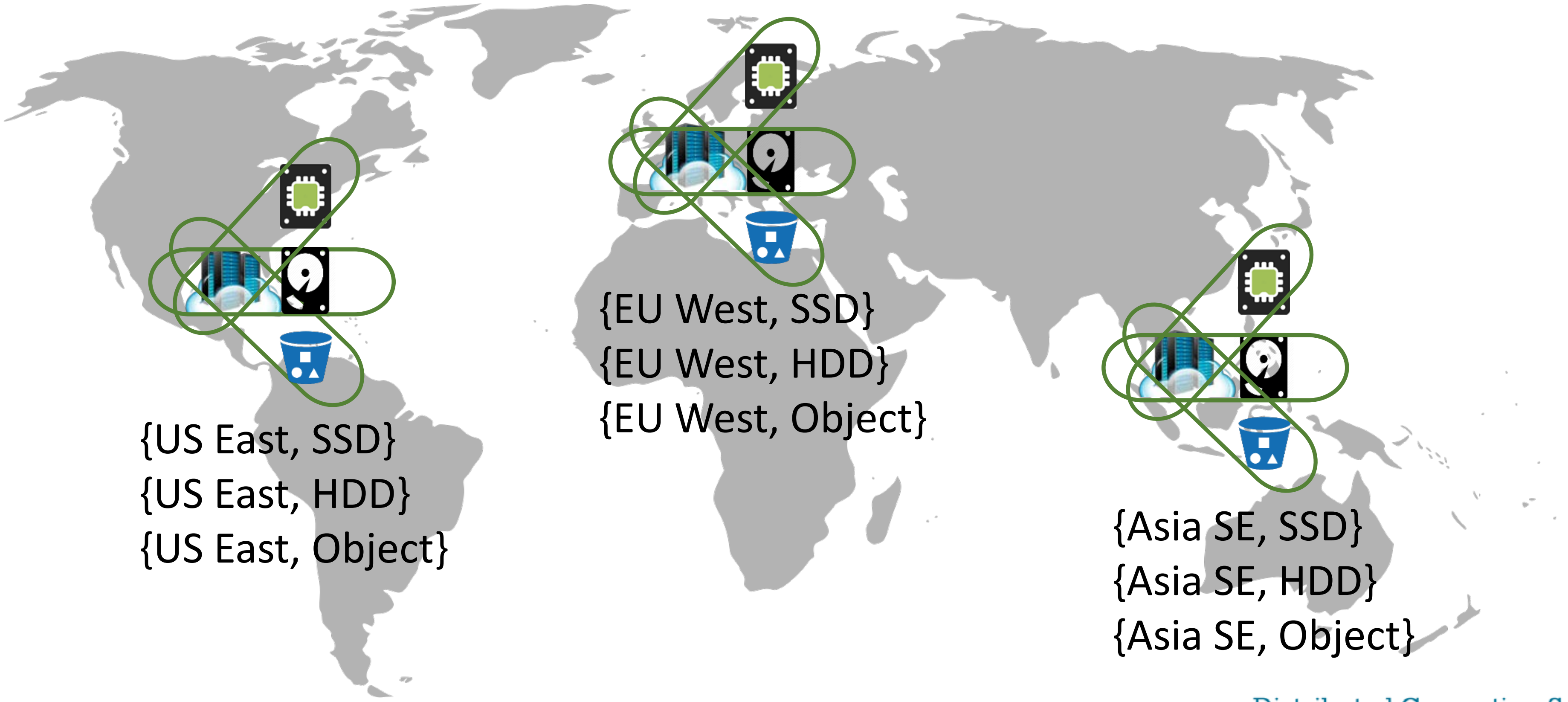
TripS Architecture



DCSG
Distributed Computing Systems Group

Locale

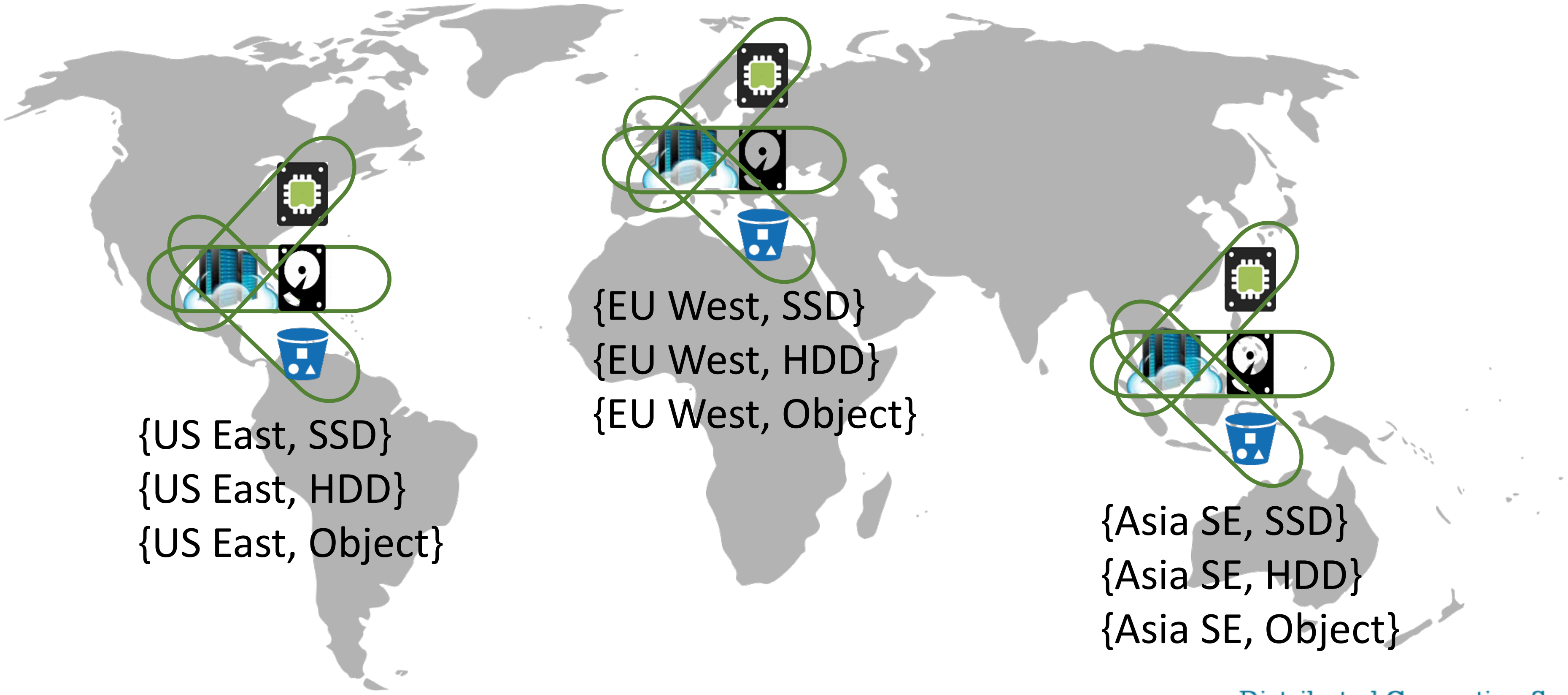
- {DC location, storage tier} tuple
- E.g., 9 locales are available



DCSG
Distributed Computing Systems Group

Data Placement Problem

- **Determining set of *locales*** to store data
  - Satisfying all applications' goals



DCSG
Distributed Computing Systems Group

TripS Inputs

- **Application desired goals**
  - SLA
  - Consistency model
  - Degree of fault tolerance
  - Locale count (LC)
- **Cost** information
  - Storage and Network cost
- **Latency** information
  - Storage and network (between DCs)
- **Workload** information
  - Number of Requests (Get and Put)
  - Average data size

Input	Description
D	Set of DCs
D_iS	Set of storage tiers in DC i
$C_{ij}^{network}$	Network cost between DC i and DC j
$C_{it}^{storage}$	Storage Tier t provisioned storage cost in DC i
$C_{it}^{get\_/put\_req}$	Get/Put request cost for storage tier t in DC i
$C_{it}^{ret/write}$	Data retrieval/write cost from/to storage tier t in DC i
$SLA^{get/put}$	Get/Put operation SLA from each DC
$LC\ (>\ 0)$	Locale count in the TLL that can be accessed within SLA from each DC location
F	Minimum number of DC faults handled
$Consistency$	Consistency Model
$Size_i$	Average object size in DC i
$Center$	Centralized DC location for a Global Lock (in strong Consistency)
$L_{ij}^{network}$	Network latency from DC i to DC j
$L_{it}^{get/put}$	Get/Put latency for storage tier t in DC i
$A_i^{get/put}$	Number of Get/Put requests for DC i

Optimized Data Placement

- **Solving data placement problem** with given inputs as MILP (Mixed Integer Linear Problem)
- **Minimized**

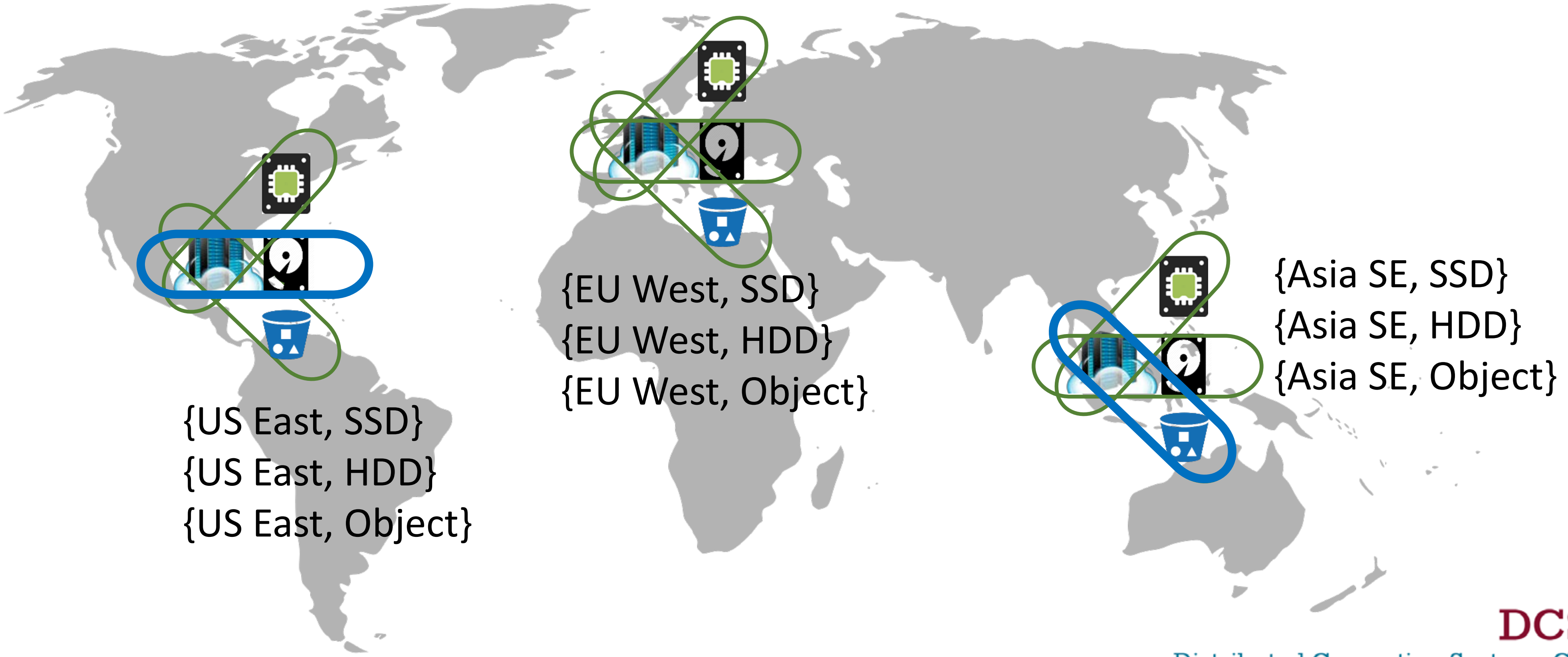
Total cost = Get Cost + Put cost
+ Broadcast Cost + Storage Cost

- Get Cost: $\sum_i \cdot A_i^{get} \cdot \sum_j \sum_t T_{ijt} \cdot (Size_i \cdot (C_{ji}^{network} + C_{jt}^{ret}) + C_{jt}^{get\_req})$
- Put Cost: $\sum_i \cdot A_i^{put} \cdot \sum_j \sum_t T_{ijt} \cdot (Size_i \cdot (C_{ij}^{network} + C_{jt}^{write}) + C_{jt}^{put\_req})$
- Broadcast Cost: $\sum_i \cdot A_i^{put} \sum_j \sum_k \sum_l B_{ijkt} \cdot (Size_i \cdot (C_{jk}^{network} + C_{kt}^{write}) + C_{kt}^{put\_req})$
- Storage Cost: $\sum_i \sum_t P_{it} \cdot Size_i \cdot C_{it}^{storage}$

DCSG
Distributed Computing Systems Group

Data Placement Example

- TripS decides to store data in 2 locales
  {US East, HDD}, {Asia SE, Object}



Roadmap

- ✓ Motivations & Goals
- ✓ TripS (Storage Switch System)
- **Handling dynamics**
- Experimental evaluations

Dynamics

- Long-term dynamics
  - E.g., diurnal access pattern, user lo
  - From hour(s) to week(s)
  - Lazy re-evaluating the data placement is enough

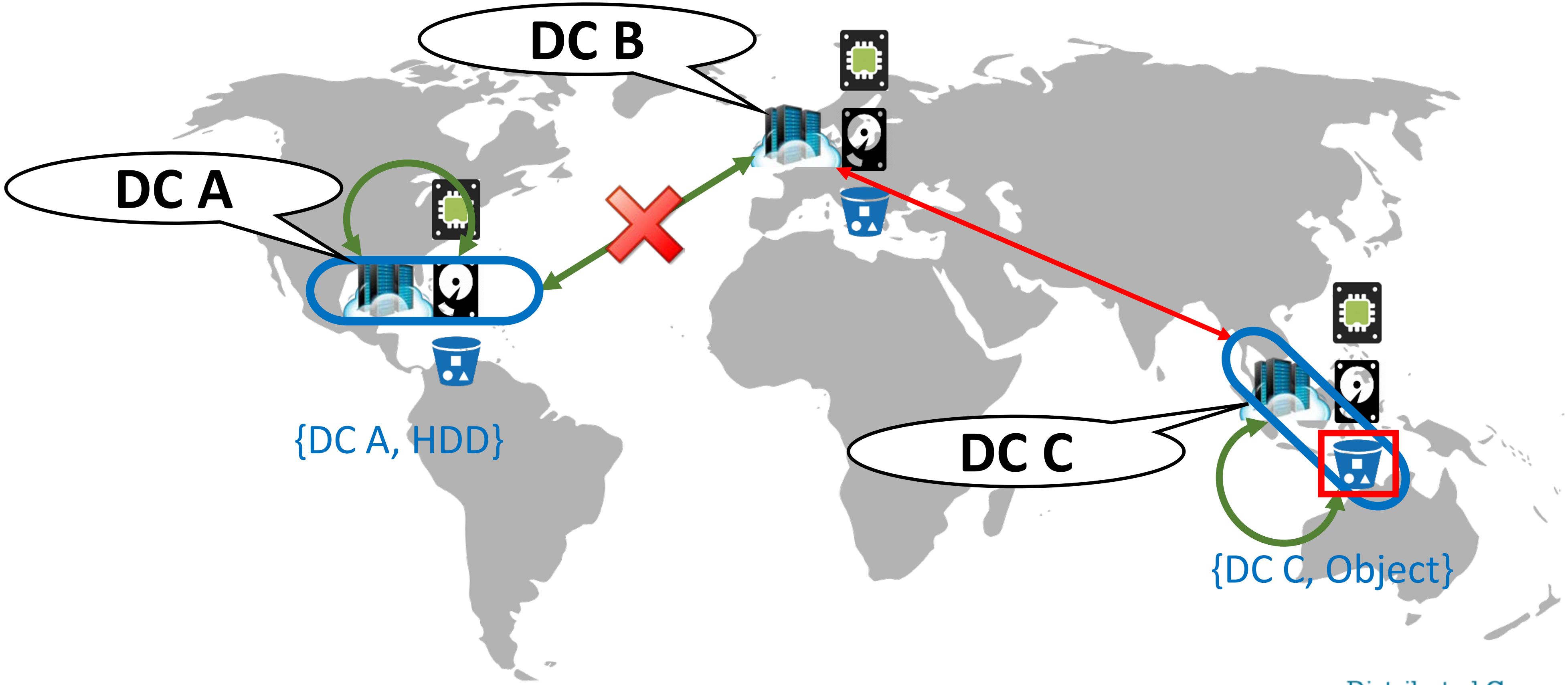
Like other systems, TripS can handle long-term dynamics

- Short-term dynamics
  - E.g., burst access, transient failures or overload
  - From second(s) to minute(s)
  - Frequent re-evaluating the data placement is **expensive!!**

Can be handled **pro-actively** with **Target Locales List (TLL)**

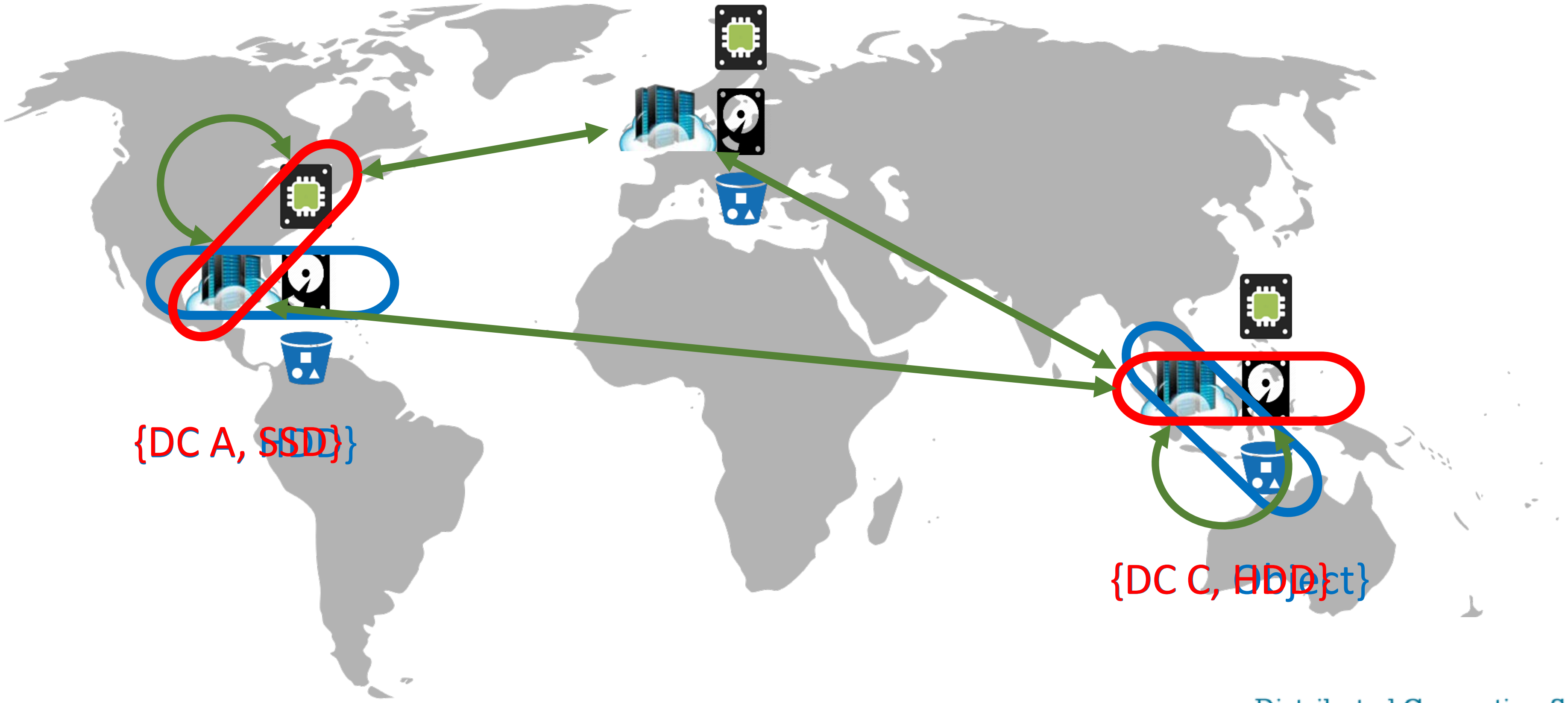
Target Locale List (TLL)

- **List of locales satisfying the SLA goal**
  - *Locale count* (LC) parameter = 1 (as an application's goal)



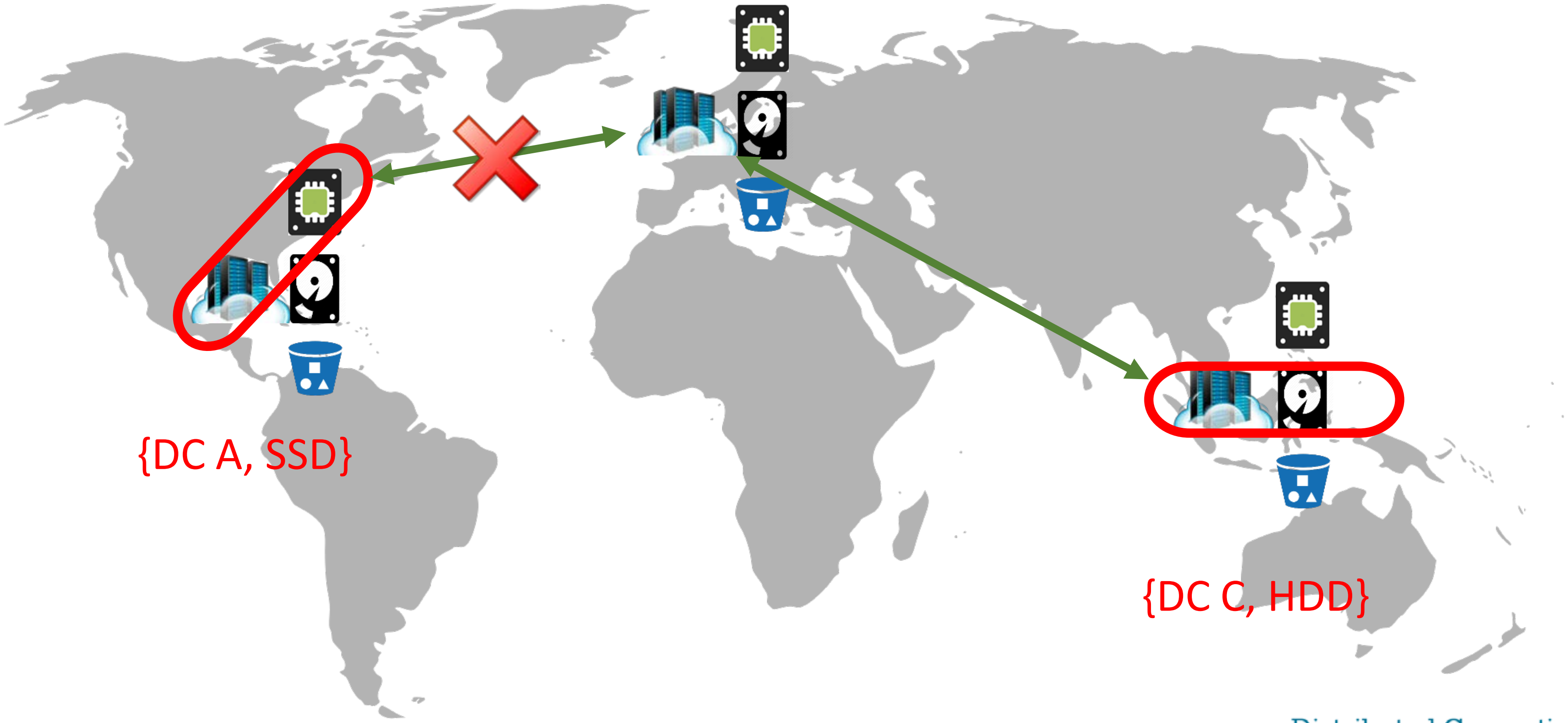
Target Locale List (TLL)

- **List of locales satisfying the SLA goal**
  - *Locale count* (LC) parameter = **2** (as an application's goal)



Locale Switching

- Avoiding SLA violation
- Tradeoff cost for performance



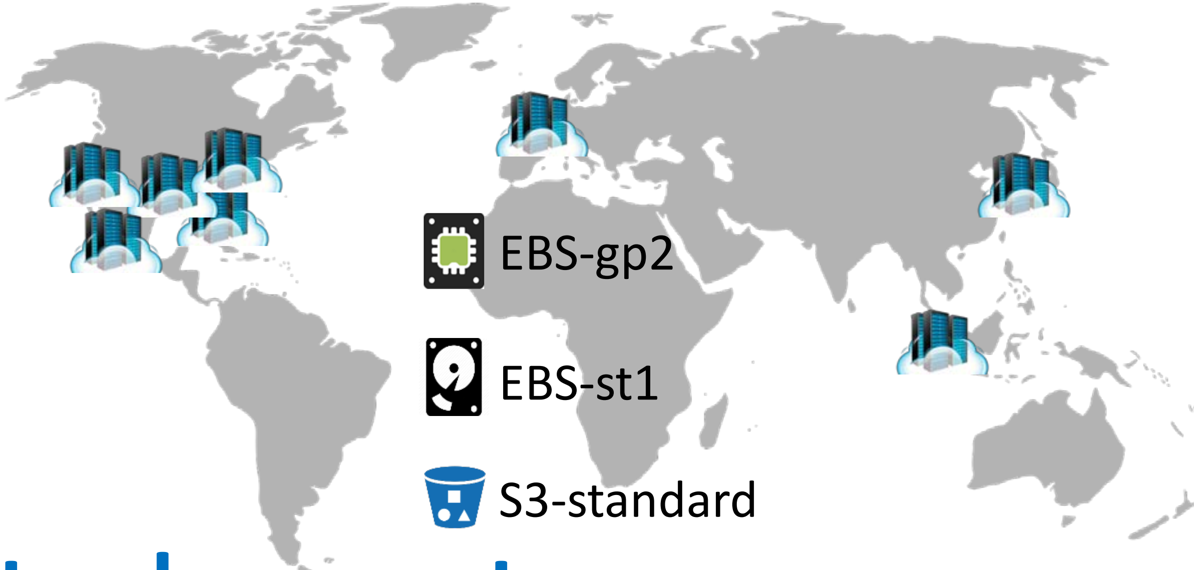
DCSG
Distributed Computing Systems Group

Roadmap

- ✓ Motivations & Goals
- ✓ TripS (Storage Switch System)
- ✓ Handling dynamics
- **Experimental evaluation**

Evaluation

- Running on Wiera [Oh et al, HPDC '16] as **GDSS**
- 8 Amazon DCs and
  3 storage tiers
- Evaluation illustrates
  - TripS finds **optimized data placement**
  - TripS helps applications **handle dynamics** (e.g., network delays or transient failures)

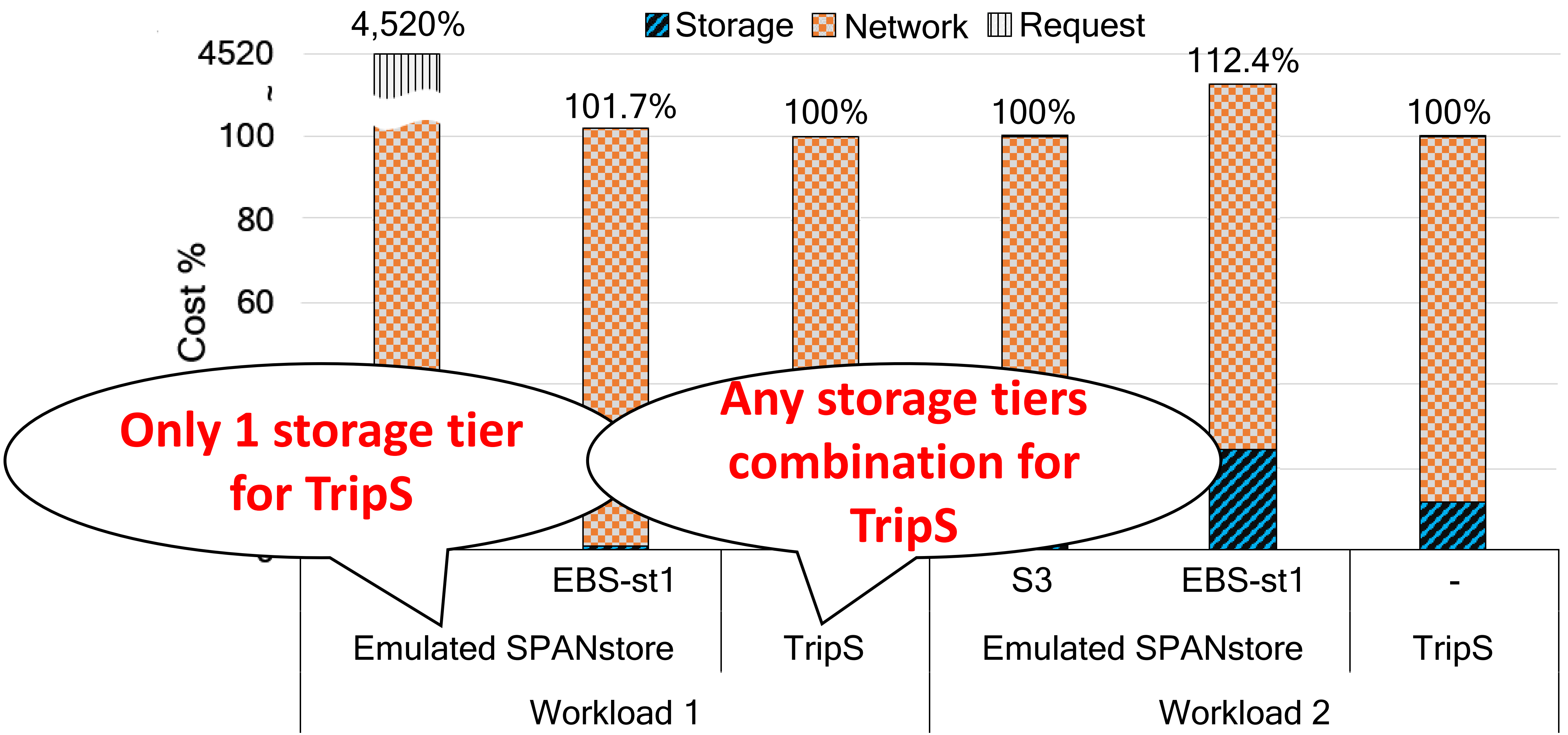


TripS Finds Optimized Data Placement

- Two synthetic workloads
  - Latency sensitive Web applications
  - Data analytic applications
- Compare with emulated SPANStore [Wu et al, SOSP '13]
  - Only one storage tier (S3 or EBS) on TripS

	Average Data Size	# Get / Put Request	Get / Put SLA
Workload 1	8 KB (small data)	10,000 / 1,000 (frequent accessed)	200 ms / 350 ms (latency sensitive)
Workload 2	100 MB (big data)	1,000 / 100 (less frequent accessed)	500 ms / 800 ms (bandwidth sensitive)

Optimized Data Placement for Both Workload

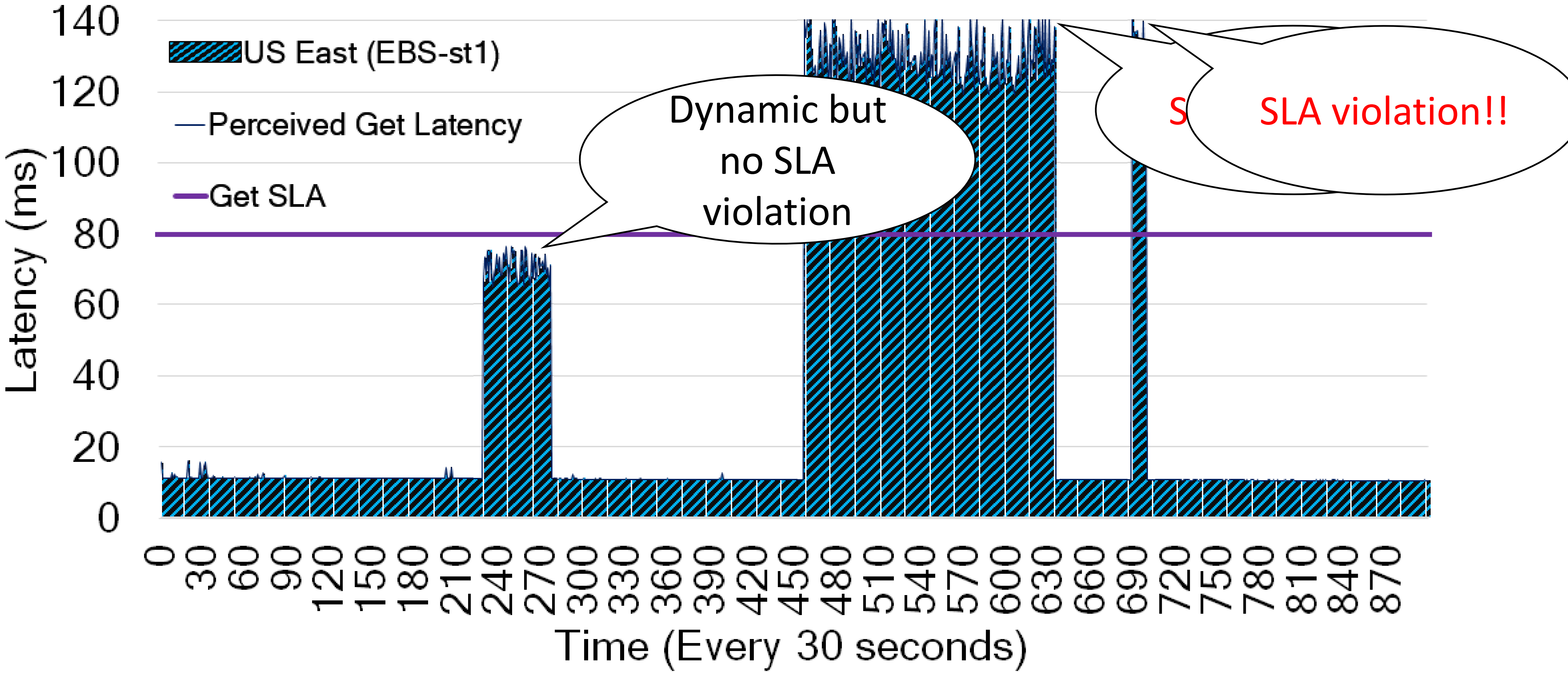


Handling Short-term Dynamics

- 5 DCs on North America region
- Workload
  - YCSB Workload B
    - 95% Read, 5% Write
  - Average data size: 8 KB
  - 80 ms (Get) / 200ms (Put)
- Varying LC parameter

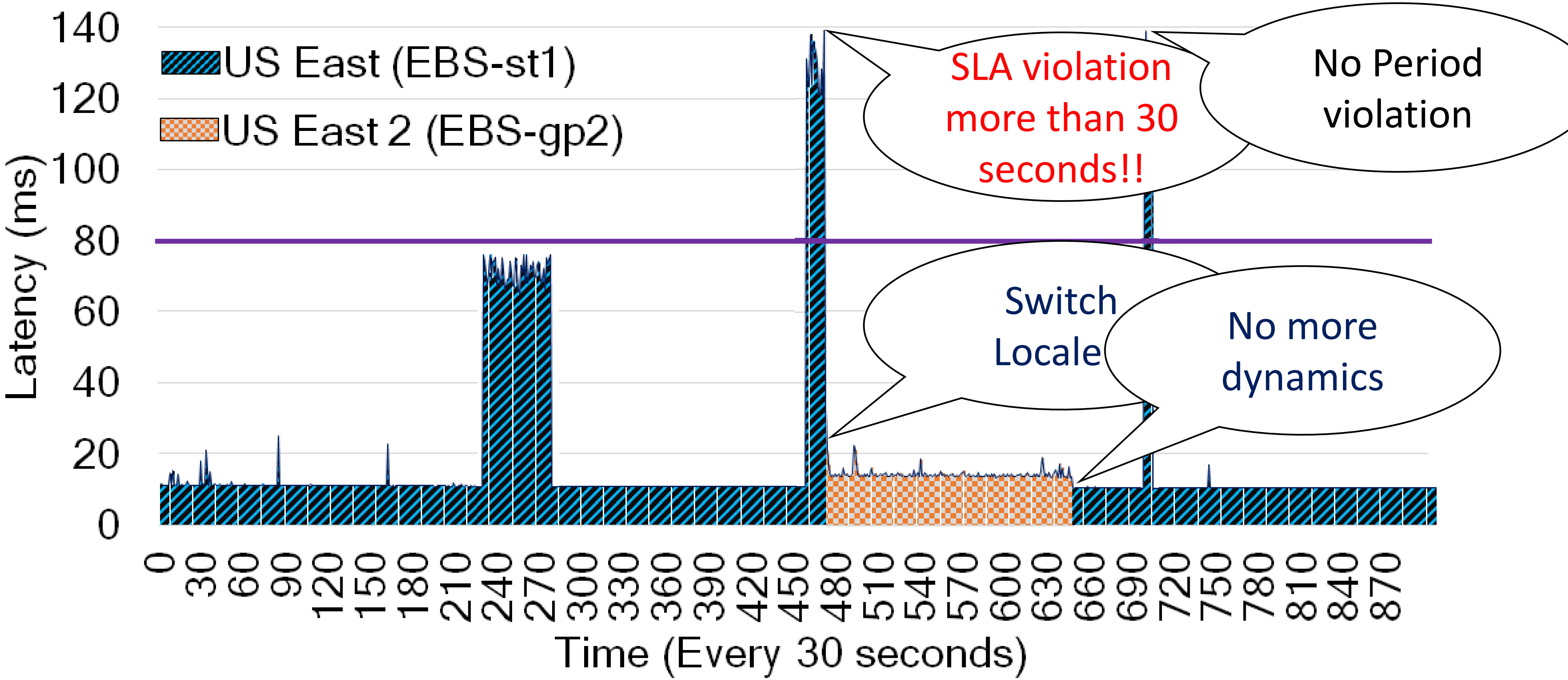
DCSG
Distributed Computing Systems Group

Transient Network Delays with LC = 1



DCSG
Distributed Computing Systems Group

Transient Network Delays with LC = 2



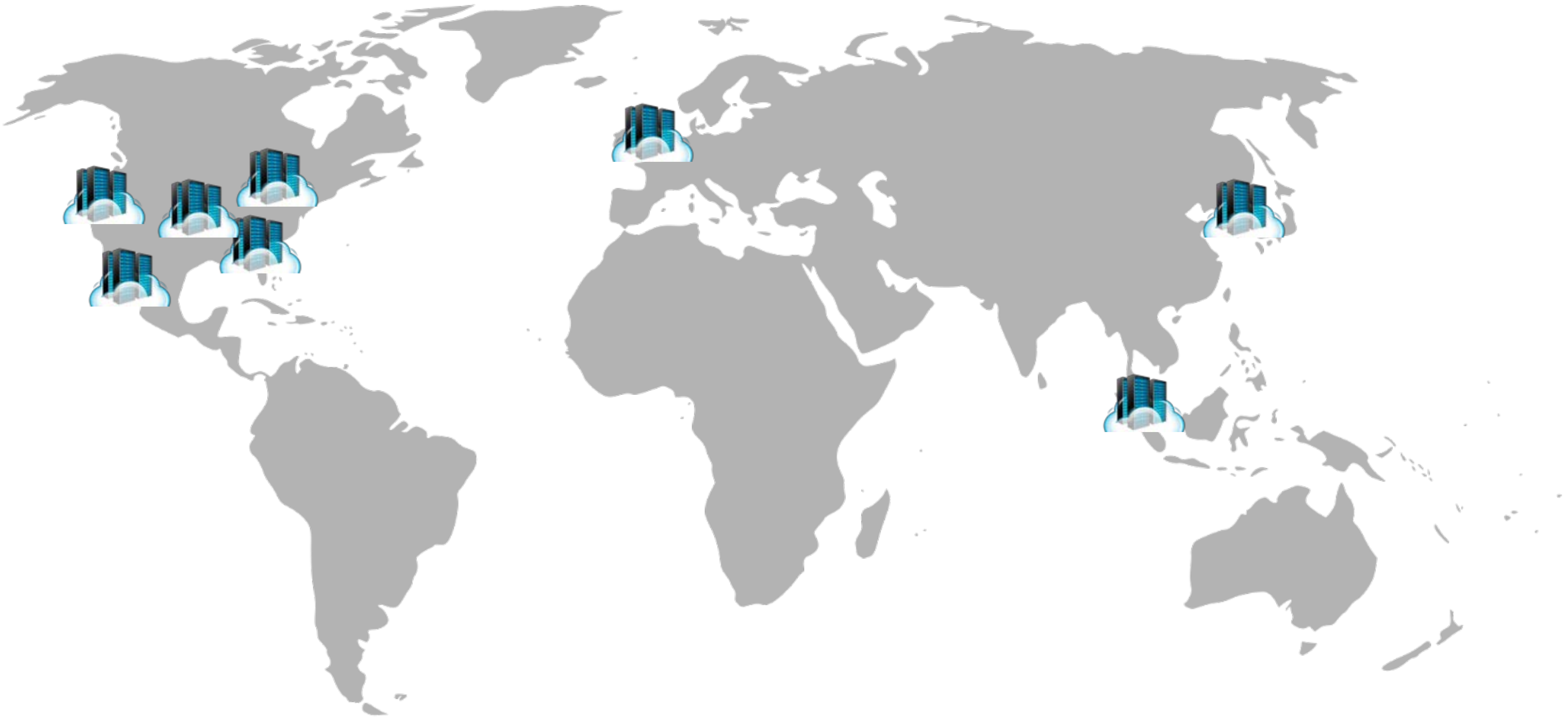
Tradeoff Cost for Performance by LC

- As LC increases, total cost also increases
- Tradeoff cost for performance

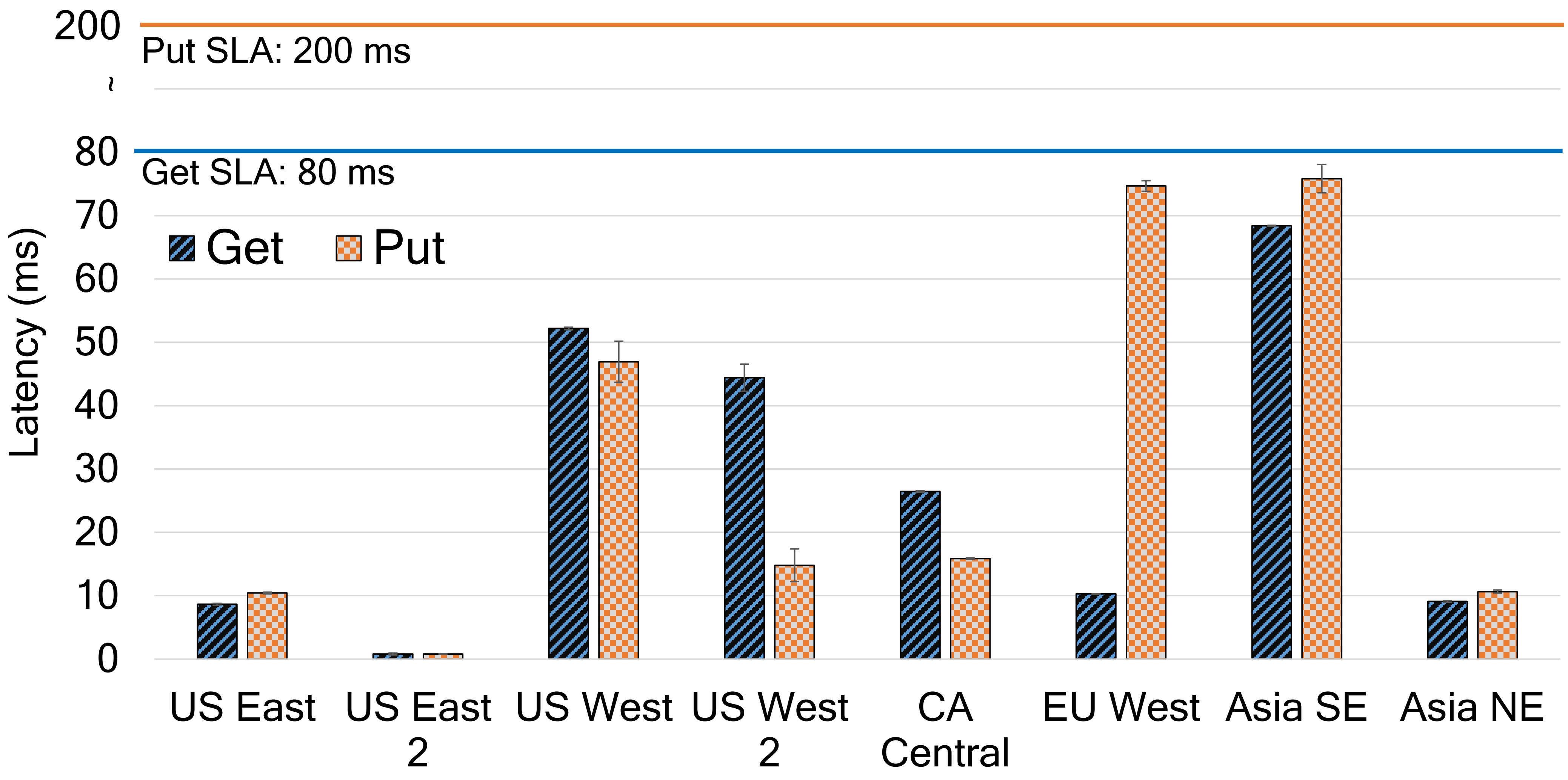
LC parameter	Data placement	Storage	Network	Total
1	{US East, EBS-st1}, {US East 2, EBS-st1}, {US West 2, EBS-st1}	100%	100%	**100%**
2	{US East, EBS-st1}, {US East 2, **EBS-gp2**}, {US West 2, EBS-st1}	140.7%	100%	**105.3%**
3	{US East, **EBS-gp2**}, {US East 2, EBS-gp2}, {**US West**, EBS-st1}	188.1%	100%	**111.5%**
4	{US East, EBS-gp2}, {US East 2, EBS-gp2}, {US West, EBS-st1}, {**CA central, EBS-gp2**}	269.6%	166.7%	**180.1%**

Real Application Scenario - Retwis

- Twitter like Web application
- Using TripS-enabled Wiera instead of Redis



Satisfying SLA Goals



1K users: 125 Users per each location

DCSG
Distributed Computing Systems Group

Conclusion

- TripS finds optimized data placement with a consideration both **DC locations** and **storage tiers** with **minimized cost**
- TripS helps applications **handle dynamics** especially **short-term dynamics** with Target Locale List (TLL)

Thank You!